Splunk 7.x Quick Start Guide

Gain business data insights from operational intelligence

Packt>

www.packt.com

James H. Baxter

Splunk 7.x Quick Start Guide

Gain business data insights from operational intelligence

James H. Baxter

Packt>

BIRMINGHAM - MUMBAI

Splunk 7.x Quick Start Guide

Commissioning Editor: Amey Varangaonkar
Acquisition Editor: Reshma Raman
Content Development Editor: Roshan Kumar
Technical Editor: Jinesh Topiwala
Copy Editor: Safis Editing
Project Coordinator: Hardik Bhinde
Proofreader: Safis Editing
Indexer: Pratik Shirodkar
Graphics: Alishon Mendonsa
Production Coordinator: Jisha Chirayil

First published: November 2018

Production reference: 1281118

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham
B3 2PB, UK.

ISBN 978-1-78953-109-1

www.packtpub.com

I'd like to dedicate this book to my kids, who make my life much more complete, and to my girlfriend, Cathy, who patiently endures my work style.

– James H. Baxter

mapt.io

Mapt is an online digital library that gives you full access to over 5,000 books and videos, as well as industry leading tools to help you plan your personal development and advance your career. For more information, please visit our website.

Why subscribe?

- Spend less time learning and more time coding with practical eBooks and Videos from over 4,000 industry professionals
- Improve your learning with Skill Plans built especially for you
- Get a free eBook or video every month
- Mapt is fully searchable
- Copy and paste, print, and bookmark content

Packt.com

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.packt.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at customercare@packtpub.com for more details.

At www.packt.com, you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on Packt books and eBooks.

Contributors

About the author

James H. Baxter is the owner/CEO of Machine Data Insights, Inc., a certified Splunk architect, and a developer and machine learning practitioner with over 35 years of experience in various engineering and analysis disciplines, including radio/satellite; networks; capacity and performance modelling; speech technology; packet-level analysis; programming; and Splunk architecture, administration, and machine learning solutions for companies including MCI, IBM, BP, Disney, and AMEX. James is also a private pilot and holds an Extra class amateur radio and FCC Radiotelephone license.

You can reach him at LinkedIn at `James H. Baxter`.

About the reviewer

Yogesh Raheja is a certified DevOps and cloud expert with a decade of IT experience. He has expertise in technologies including OS, source code management, build and release tools, continuous integration/deployment/delivery tools, containers, configuration management tools, monitoring, logging tools, and public and private clouds. He loves to share his technical expertise worldwide at various forums, conferences, webinars, blogs, and on LinkedIn. He has authored *Effective DevOps with AWS*, *Second Edition*, *Automation with Puppet 5*, and *Automation with Ansible*, and has published his online courses on Udemy. He has reviewed multiple books for Packt, including *Implementing Splunk 7*, *Third Edition*, and *Splunk Operational Intelligence Cookbook*, *Third Edition*.

Packt is searching for authors like you

If you're interested in becoming an author for Packt, please visit `authors.packtpub.com` and apply today. We have worked with thousands of developers and tech professionals, just like you, to help them share their insight with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Table of Contents

Preface 1

Chapter 1: Introduction to Splunk 8
- **What is Splunk?** 9
  - Splunk products 10
  - The history of Splunk 11
- **Installing Splunk for free** 12
- **Splunk components** 13
- **Splunk processing tiers** 17
- **Splunk events** 19
- **Splunk information resources** 22
- **Summary** 23

Chapter 2: Architecting Splunk 24
- **Selecting a Splunk configuration** 24
  - Data collection – data inputs 25
  - Data collection – concurrent searches 27
  - Distributed versus clustered Splunk environments 28
  - Replication and search factor 29
    - Replication factor 30
    - Search factor 30
  - Hot/warm and cold buckets 31
  - Search head clusters 32
  - Making a design decision 33
- **Selecting Splunk hardware options** 34
  - Performance considerations 34
  - Making a hardware selection 37
  - Disk-sizing calculations 38
- **Summary** 40

Chapter 3: Installing and Configuring Splunk 41
- **Installing Splunk Enterprise** 41
  - Installing Splunk on Linux 42
    - Linux settings 43
      - User–group – environment settings 43
      - ulimits 43
      - Transparent huge pages 44
    - Starting Splunk 45
      - Starting on reboot 46
      - Stopping Splunk 47
  - Installing Splunk on Windows server 47
    - Disabling antivirus software 47

Installing Splunk with a short pathname 48
Installing Splunk via the GUI 48
Stopping and starting Splunk on Windows 49
Synchronization of system clocks 49
Configuring Splunk components 50
Splunk directory structure 50
Configuration file precedence 52
Splunk installation checklist 54
Component and IP address list 54
Installation steps 55
Individual component configurations 56
License master and cluster master 57
Forwarding Splunk's internal logs to the indexers 59
Pointing servers to the license master 59
Indexing cluster 60
Configuring a TCP input 60
Deployer 61
Search heads 62
Designating and starting a search head captain 63
Checking search head cluster status 63
Deployment server 64
Multisite environments 65
Cluster master 65
Indexers 66
Search heads 66
Cross-environment search 67
Documenting your Splunk deployment 68
Summary 70

Chapter 4: Getting Data into Splunk 71
Installing Splunk universal forwarder 72
Installation steps 72
Starting/stopping the universal forwarder 74
Configuring outputs.conf 74
Configuring inputs.conf 75
Setting up a heavy forwarder 78
Configuring other data source inputs 80
Configuring an HTTP Event Collector 81
Testing the HTTP Event Collector 85
Introduction to apps 86
Using the deployment server 87
Configuring a deployment client 89
Configuring the deployment server 90
Creating deployment apps 90
Creating a serverclass.conf file 91
Using forwarder management in Splunk web 94
Managing Splunk Indexes 95

Creating an index 96
Deleting index data 98
Summary indexes 98
Metrics indexes 99
Splunk sourcetypes 100
Creating custom source types 102
Using the cluster master 104
Distributing the configuration bundle 105
Summary 108

Chapter 5: Administering Splunk Apps and Users 109
Using the deployer 109
Deploying new or updated apps 114
Configuring users and roles 114
Splunk authentication 114
LDAP authentication 116
SAML authentication 117
Managing Splunk roles 118
Search restrictions 119
Capabilities 119
Indexes 119
authorize.conf 120
Working with authentication.conf and authorize.conf 120
Best practices for administering Splunk 121
Index naming conventions 122
Source type naming conventions 122
Location of indexes.conf, props.conf, and transforms.conf 123
Supporting your Splunk Deployment 124
Splunk support personnel 124
Funding Your Splunk deployment 125
Splunk resource cost calculations 126
Summary 127

Chapter 6: Searching with Splunk 128
The Splunk Web interface 129
Search controls 131
Timeline and events 134
Creating Splunk searches 137
Basic search commands 138
Index 138
Time-range selection 139
Search filters 141
Search commands 143
Eval 143
Stats 144
Dedup 146
Rex 146

Where 147
Formatting commands 147
Rename 148
Sort/reverse 148
Head/tail 148
Top/rare 149
Visualizing search results 149
Table/fields 149
Chart/timechart 150
Chart 151
Timechart 152
Visualizations in Splunk web 153
Advanced search commands 154
Subsearches 154
Join 155
Transaction 156
Streaming versus transforming commands 158
Optimizing searches 159
Optimizing search jobs 159
Job inspector 160
Summary 161

Chapter 7: Splunk Knowledge Objects 162
Field extractions 163
Index-time field extractions 163
Search-time field extractions 163
Using the extract fields interface 165
Other knowledge objects 167
Event types – tags – aliases 168
Event type 168
Tags 169
Field aliases 169
Lookups 170
Macros 172
Datasets and data models 173
Datasets 173
Data models 174
Using data models in search 176
Data model acceleration 177
Pivot tables 177
Summary 180

Chapter 8: Splunk Reports, Dashboards, and Alerts 181
Introduction 181
Creating reports 184
Scheduling a report 186
Creating a dashboard 191
Adding a new panel with inline search 194

Editing panel characteristics 195
Using dashboard forms 196
Using tokens 197
Working with Simple XML 198
Improving dashboard performance 200
Using JavaScript and CSS within a dashboard 201
Event-handlers 202
Creating an alert 203
Summary 205

Chapter 9: Splunk Applications 206
Splunk apps and add-ons 207
Creating a Splunk app 208
App context and permissions 210
Using Splunkbase 215
Splunk app and add-on for Unix and Linux 215
Machine learning toolkit 220
Splunk DB Connect 223
Requirements and installation 223
Hardware requirements 224
Java runtime 224
Installing DB connect 224
Database JDBC drivers 225
Configuring DB Connect 226
Configuring task server 226
Database drivers 228
Configuring database input 229
Identities and roles 229
Connections 231
Input 233
Output 236
Lookups 238
Troubleshooting DB Connect 240
HEC port conflicts 241
Splunk Premium apps 241
IT service intelligence 242
Enterprise security and UBA 243
Summary 244

Chapter 10: Advanced Splunk 245
Troubleshooting Splunk 246
Splunk logs 246
btool 248
diag 249
Opening a Splunk support case 250
Locked license issue 251
Performance and capacity 251

REST API endpoints 253
Splunk Monitoring Console 254
Configuring the monitoring console 256
Using the Monitoring Console 258
Data rebalancing 260
Indexer clustering and bucket status 261
Upgrading Splunk Enterprise 262
Splunk development 262
Software Development Kits 263
Using the Python SDK 264
The REST API 266
Additional study topics 266
Summary 268
Other Books You May Enjoy 269
Index 272

Preface

Splunk is an increasingly popular platform for collecting, searching, monitoring, and analyzing ever-growing amounts of big data from applications, network devices, and Internet of Things sensors. Aggregating, centralizing, and analyzing log and event data with Splunk turns that data into answers regarding the health of machines and applications, counts and trends in customer transactions, security threats, and a multitude of other insights that may be valuable to a particular company or industry.

Over the last five years, Splunk has more than doubled its number of customers, which now totals over 13,000 in 110 countries, including 89 of the Fortune 100. Given the increasing trend and opportunity to profit from the valuable insights derived by leveraging machine learning (ML) techniques on large data sets, Splunk has positioned itself well for further growth by building ML into its premium applications, such as IT Service Intelligence, Enterprise Security, User Behavior Analytics, and Industrial Asset Intelligence, in order to provide real-time and predictive analytics in those environments, as well as providing the Machine Learning Toolkit for developing custom solutions. So, as far as the IT professional who likes to work with data and data systems is concerned, learning how to architect, implement, administer, and/or use Splunk for analyzing data is a safe and valuable career investment for the foreseeable future. This book was written with a view to helping you embark on that journey and learn the landscape as quickly as possible.

Who this book is for

This book is intended for experienced IT personnel who are just getting started working with Splunk and who want or need to quickly get to the heart of the matters of architecting, implementing, and administering Splunk, as well as working with Splunk search and several key apps to extract value from collected data and/or help their user base to do so. Business users who need to leverage Splunk to extract useful data from Splunk by building reports, dashboards, and alerts, and perhaps even utilize the Machine Learning Toolkit or one of Splunk's premium apps, such as Enterprise Security, will also benefit from this book, as it is a quick read that will help them better understand and work with the underlying technology from which their rich datasets and solutions are derived, and gain insights that may foster ideas for extracting even more value from their data.

This book is obviously too short to provide complete coverage of all of Splunk's features, possible configurations, and related options (that would require a few thousand pages), but it does strive to provide an introduction to all of the most crucial topics, sufficient information and examples to get the immediate job done, and sufficient insights to support intelligent and efficient research to fill in any gaps and successfully complete customizations of a Splunk environment to suit any business environment or situation.

What this book covers

`Chapter 1`, *Introduction to Splunk*, introduces Splunk to the newcomer, with a high-level overview of Splunk components, features, and capabilities, along with the basics of how Splunk works, so as to serve as a solid foundation when going into further detail in subsequent chapters.

`Chapter 2`, *Architecting Splunk*, provides guidance and examples for selecting the appropriate Splunk configuration for a variety of business environments, choosing and sizing the hardware Splunk will run on, and how to calculate the amount of disk space and number of indexers you'll need to accommodate your anticipated data ingestion volume.

`Chapter 3`, *Installing and Configuring Splunk*, covers installing Splunk Enterprise and configuring each of the required components to perform their specific functions. This chapter includes a checklist for implementing a complete Splunk environment, working examples of the essential configuration file settings, and guidance for documenting the final Splunk solution.

`Chapter 4`, *Getting Data into Splunk*, gets to the heart of managing a Splunk environment. This chapter provides working examples of all of the key parameters and settings used to configure data inputs from Universal Forwarders for various log types, inputs from other data sources, and using the HTTP Event Collector for getting data into Splunk. We also cover parsing and storing the data in the various types of indexes, and how they're configured.

`Chapter 5`, *Administering Splunk Apps and Users*, wraps up the administration tasks by discussing how to manage the apps and search capabilities that users will need in order to find and extract the data stored in Splunk. Since Splunk is usually implemented as a distributed/clustered solution for reliability and scalability purposes, the focus will be on managing this more complex type of environment. Threaded throughout this chapter will be tips and strategies to help develop and apply the best standards and practices for managing and supporting a Splunk solution in a typical business environment.

`Chapter 6`, *Searching with Splunk*, is perhaps the most important part of the entire book, as this chapter covers all the crucial skills needed to get data out of Splunk indexes, reduce it to its essential elements, and transform and format the results into a dataset and visualizations that provide real value and powerful insights. The important features of the user interface—Splunk web—are leveraged in working examples of the more basic Search Processing Language (SPL) commands, which serve as the foundation for a gentle and logical progression to using the more advanced commands and visualization options.

`Chapter 7`, *Splunk Knowledge Objects*, covers the various ways you can powerfully enhance and enrich machine data with user-defined fields and additional data to help harness that information in a smarter and more focused way. Event types, tags, and aliases allow you to classify and normalize similar events; field extractions create fields from otherwise unlabeled segments of an event. Lookups enhance your data with additional information, such as the meaning of HTTP status codes. Data models are pre-prepared representations of one or more datasets created to drive pivot tables and allow business users to create complex reports and visualizations without having to use the SPL. These capabilities help make Splunk a much more useful and valuable business analysis tool, and you will want to know how it all works.

`Chapter 8`, *Splunk Reports, Dashboards, and Alerts*, builds on the search skills developed in the previous chapter to help you quickly and easily create effective reports and dashboards from saved searches that provide status indicators, charts, graphs, tables, and complex visualizations that can be viewed directly or scheduled for delivery by email with embedded PDFs. You'll also learn how to configure alerts to let support and business line personnel know when something isn't right.

`Chapter 9`, *Splunk Applications*, explains how to combine the knowledge objects, saved searches, and reports/dashboards/alerts you built from previous chapters into a Splunk app—a packaged solution that makes Splunk more useful and relevant to specific technologies or use cases. It also covers in detail how to install and configure several of the more useful (and free!) apps and add-ons available from Splunkbase – one that collects OS-level data from all your Linux and Windows servers, and another very popular app that allows you to query relational databases and ingest that data into Splunk. Finally, we'll install and review the Splunk Machine Learning Toolkit, as well as introduce Splunks' premium apps – ITSI, ES, and UBA—and see how they fit into comprehensive monitoring and situational detection solutions.

`Chapter 10`, *Advanced Splunk*, is an overview and reference for several important topics and skills that any Splunk administrator will want to include in their tool chest. While Splunk is inherently stable and reliable, there will be times when you have to troubleshoot problems; this chapter covers the most useful Splunk logs and tools for determining what's working and what isn't. Then, we segway into using the Monitoring Console to keep tabs on overall Splunk health, as well as providing working examples of searches that can be built for monitoring disk and index sizes versus configured capacity, search concurrency and performance, and other factors than an administrator will be interested in. As a finale for this chapter and book, the reader is introduced to the essential concepts and references for taking Splunk to the next level – using API endpoints and the Splunk SDKs and frameworks for developing powerful customized solutions on top of the Splunk platform.

The coverage of functionality and the examples provided in this book are based on Splunk 7.1.1, which was current at the time of writing. Splunk is aggressively expanding and improving its product, so there will inevitably be new features and capabilities released in the future that are not covered, but the functions and configurations that are covered in this book are central to the Splunk platform, meaning that the information should remain relevant and useful for quite some time.

To get the most out of this book

To get the most out of this book, you will need to install the free version of Splunk Enterprise on your desktop or laptop so that you can investigate Splunk's directory structure and configuration files and options, and follow along in each chapter by experimenting with the configurations, searches, apps, and report/dashboard/alert examples provided.

If you want to develop your architect and administration skills with Splunk and don't have admin-level access to a Splunk sandbox environment at your workplace, you may want to consider building a small Splunk environment on cloud-based servers; the cost is not too great if you manage your up-time carefully, and you can configure and run a clustered solution using the free Splunk Enterprise trial license for up to 30 days.

Downloading the extra material

You can download a file that contains the data collection forms and indexer disk space calculator spreadsheets featured in `Chapter 2`, *Architecting Splunk*, clickable links to all the URLs providing additional information, and the search strings from each chapter, which you can copy/paste and alter to meet your requirements by logging into your account at `www.packtpub.com`. If you purchased this book elsewhere, you can visit `www.packtpub.com/ support and register to have the file emailed to you.`

Download the example code files

You can download the example code files for this book from your account at `www.packt.com`. If you purchased this book elsewhere, you can visit `www.packt.com/support` and register to have the files emailed directly to you.

You can download the code files by following these steps:

1. Log in or register at `www.packt.com`.
2. Select the **SUPPORT** tab.
3. Click on **Code Downloads and Errata**.
4. Enter the name of the book in the **Search** box and follow the onscreen instructions.

Once the file is downloaded, please make sure that you unzip or extract the folder using the latest version of:

- WinRAR/7-Zip for Windows
- Zipeg/iZip/UnRarX for Mac
- 7-Zip/PeaZip for Linux

The code bundle for the book is also hosted on GitHub at `https://github.com/PacktPublishing/Splunk-7.x-Quick-Start-Guide`. In case there's an update to the code, it will be updated on the existing GitHub repository.

We also have other code bundles from our rich catalog of books and videos available at `https://github.com/PacktPublishing/`. Check them out!

Download the color images

We also provide a PDF file that has color images of the screenshots/diagrams used in this book. You can download it here: `http://www.packtpub.com/sites/default/files/downloads/9781789531091_ColorImages.pdf`.

Conventions used

There are a number of text conventions used throughout this book.

`CodeInText`: Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles. Here is an example: "The `rpm` will install Splunk in the `/opt/splunk` directory"

A block of code is set as follows:

```
index=<index> <filter> <"text string to match">
| command1 <arguments>
| command2 <arguments>
| visualization commands & arguments
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
hot bucket (files being written to)
/opt/splunk/var/lib/splunk/myindex/db/hot_v1_41
warm bucket (closed for writing, searchable)
/opt/splunk/var/lib/splunk/myindex/db/db_1530043376_1529957920_40/
cold bucket (searchable, may reside on different storage)
/opt/splunk/var/lib/splunk/myindex/colddb/db_1508276979_1508276438_0/
```

Any command-line input or output is written as follows:

```
$ sudo su - splunk                 don't forget this step!
$ cd $SPLUNK_HOME/bin
$ ./splunk start --accept-license
```

Bold: Indicates a new term, an important word, or words that you see on screen. For example, words in menus or dialog boxes appear in the text like this. Here is an example: "You can now click **Settings** | **Fields** | **Field extractions** and view the list of all the field extractions, including the one you just created."

Warnings or important notes appear like this.

Tips and tricks appear like this.

Get in touch

Feedback from our readers is always welcome.

General feedback: If you have questions about any aspect of this book, mention the book title in the subject of your message and email us at `customercare@packtpub.com`.

Errata: Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you would report this to us. Please visit `www.packt.com/submit-errata`, selecting your book, clicking on the Errata Submission Form link, and entering the details.

Piracy: If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at `copyright@packt.com` with a link to the material.

If you are interested in becoming an author: If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit `authors.packtpub.com`.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions, we at Packt can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about Packt, please visit `packt.com`.

1
Introduction to Splunk

Welcome to the first chapter of the Splunk 7.x Quick Start Guide! This chapter introduces Splunk to the newcomer and guides them progressively toward understanding the reasons why Splunk is so popular. It introduces all the powerful capabilities and solutions it offers for collecting and analyzing machine data from a wide variety of devices and environments. This chapter also includes a high-level overview of how Splunk works to serve as a foundation for digging into more details in the chapters to come.

The topics that are covered in this chapter include the following:

- Understanding what Splunk is and what problems it solves
- Installing a free version of Splunk Enterprise
- Becoming familiar with the major components of a Splunk solution and their functions
- Becoming aware of the major processing tiers of a Splunk deployment—data input, parsing, indexing, and search
- Learning about the four key Splunk fields for every event—`_time`, host, source, and sourcetype—and why they're important
- Becoming aware of the Splunk community and all the information and resources available to learn more about configuring and using Splunk

What is Splunk?

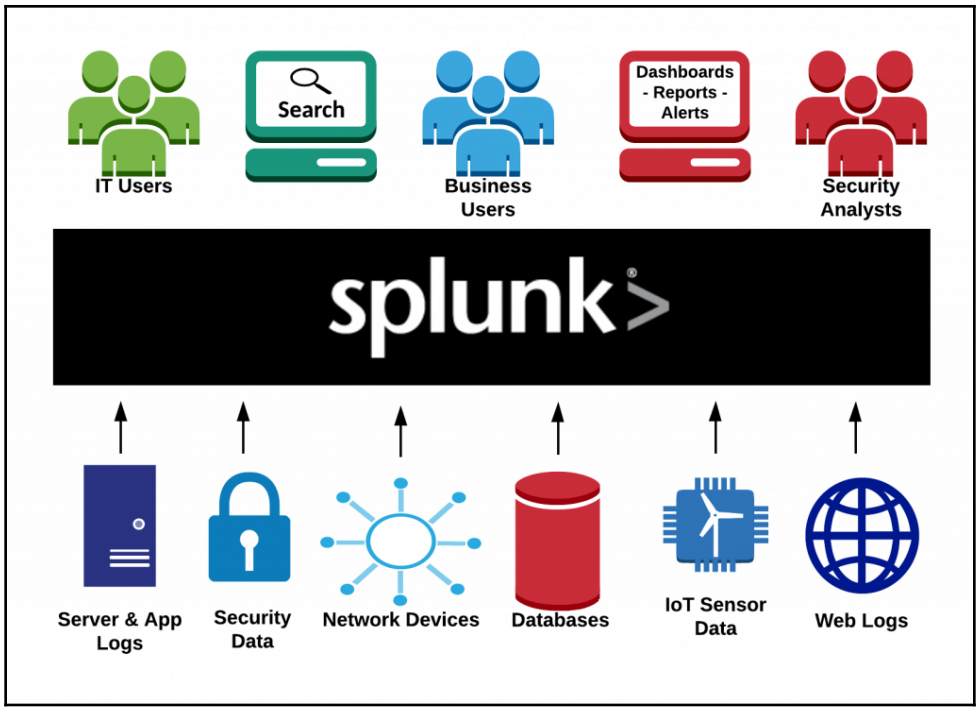
Okay—so what is Splunk, anyway? How do you explain this product to your peers, friends, and family in a way that is easy to comprehend without watering down its awesome capabilities? Here's how I explain it, with an introductory setting and then increasing levels of powerful uses—until I notice their eyes starting to glaze over, at which point I stop and summarize again: "It's like Google for all kinds of machine data!"

Every company will have tens, hundreds, or maybe thousands of application and web servers, databases, and network devices such as switches, routers, and firewalls; all kinds of sensors, and so on—and all of these create log files or data streams that record their activities and statuses over time. Now, imagine needing to troubleshoot a problem that might be caused by any one of several parts of a system, and having to log into each of these machines one at a time, manually dig through its log file looking for clues, then log into the next, and so on—you can see how tedious and time-consuming this can become. Or maybe you want to monitor critical processes to make sure things are running well—how do you do that for a lot of machines?

Splunk is a software platform that collects and stores all this machine data in one place. It makes it as easy to search through and investigate that data as using Google. Basically, it's Google for log files! Beyond troubleshooting, you can use this search capability to build reports and dashboards to monitor performance, reliability, or other metrics across a whole collection of related servers and devices, and even create alerts to warn you by text or email when something is going wrong. It's also used to detect security threats, and since you have all this data in one place, you can do event correlation across devices and apply machine learning to it for the purposes of anomaly detection, user behavior analytics, and even predictive analytics to identify potential problems before they happen.

Splunk has a media kit brochure that covers the spectrum of ways Splunk helps companies extract value from their machine data, which can be found at: `https://www.splunk.com/en_us/newsroom/media-kit.html`.

The following diagram illustrates the spectrum of data you can collect with Splunk, and captures the essence of what Splunk does:



Splunk data sources and use cases

Splunk products

There are several types and licensing models of Splunk available to suit the needs of its customer base.

Splunk Enterprise is designed for on-premise deployments; it can be scaled to support an unlimited number of users and ingestion volumes by adding the necessary number and types of Splunk functional software components (indexers and search heads) on customer-supplied servers. The cost of a Splunk Enterprise license is based on daily ingestion volume. A wide range of applications for Splunk Enterprise, written by both Splunk and the user community, that add value to the product are available for free from the Splunkbase website. Splunk also sells several sophisticated premium solution applications such as IT Service Intelligence, Enterprise Security, and User Behavior Analytics on an individual license basis, and a Machine Learning Toolkit is available for free from Splunkbase if you want to roll out your own ML solutions.

Splunk Cloud is a cloud-based **software as a service** (**SaaS**) version of Splunk Enterprise; it is also licensed based on daily ingestion volume, and offers all the functionality of the on-premise Splunk Enterprise product. The core Splunk infrastructure is provided and managed by Splunk, while data inputs, approved apps, reports, dashboards, alerts, and so on are managed by the customer.

Splunk Light is designed to be a small-scale solution. It allows up to 20 GB/day of ingestion volume, five users, and reports/dashboards/alerts, but does not provide support for distributed deployments, Splunkbase or premium solution apps (except for an app for **Amazon Web Services** (**AWS**)), and several other limitations.

Splunk Free is a free version of the core Splunk Enterprise product that has limits on users (one user), ingestion volume (500 MB/day), and other features; also, it cannot be used for a deployed/clustered configuration.

You can compare the available features of each product here: `https://www.splunk.com/en_us/software/features-comparison-chart.html`.

The history of Splunk

Splunk was designed from the beginning to allow people who were troubleshooting IT problems to search through their log files as easily as if using a search engine like Google. As a product, Splunk was conceived by founders Rob Das and Erik Swan between 2002 and 2004 after asking a number of people at 60-70 companies, how do you find problems in your infrastructure today? The answer was consistently that they would go through log files, and when asked about what tools they used, they tended to answer that they would write their own scripts and that they do everything manually. They also commented that they used Google to search for posts from other people who had solved the same problem. Rob and Erik thought, why not create a tool that allows people to search through log files as easily as they search the web using something like Google? So they built a prototype to demo at Linux World with the tagline of Google for log files, and since they offered a free download, people tried it, liked it, and told their friends—from there, Splunk spread virally from person to person and company to company.

The name **Splunk** was derived from asking people, what is it like to troubleshoot in your environment? One answer was that it was like digging through caves with headlamps and helmets on, and crawling through the muck trying to find the problem, and it took forever. A common term for going through caves is spelunking, so they decided to call the product **Splunk**. From that beginning, Splunk has evolved into the powerful big data and business analytics tool that the founders envisioned from the start.

You can watch Rob and Erik tell their story here: `https://www.splunk.com/view/SP-CAAAGBY`.

Installing Splunk for free

It will be very helpful while you're learning about Splunk to have an environment where you can navigate through directories, inspect and modify configuration files, and experiment freely with Splunk menu options and features without jeopardizing a Splunk installation in your Enterprise, even if you have the access rights to do so. You can install a free version of Splunk on your laptop or personal server that provides this flexibility from this link: `https://www.splunk.com/en_us/download/splunk-enterprise.html`.

Complete the form to create a Splunk account (or log in if you already have one), click to agree to the Splunk **Software License Agreement**, and click **Create your account**. From the downloads page, click the Windows, Linux, or macOS tabs at the top and select the version appropriate for your platform. You will need to agree to the software license again, then click **Start Your Download Now**, and save the file to disk. The next web page that appears provides links to installation videos and documents for your chosen environment; the installation process is very straightforward, so you shouldn't have any trouble.

If you choose to **Launch browser with Splunk Enterprise** when the installation finishes, you will log in with the default username of **admin**, and password of **changeme**. Change the password upon the first login.

And as they say at the end of the installation videos—dig in and start exploring!

Splunk components

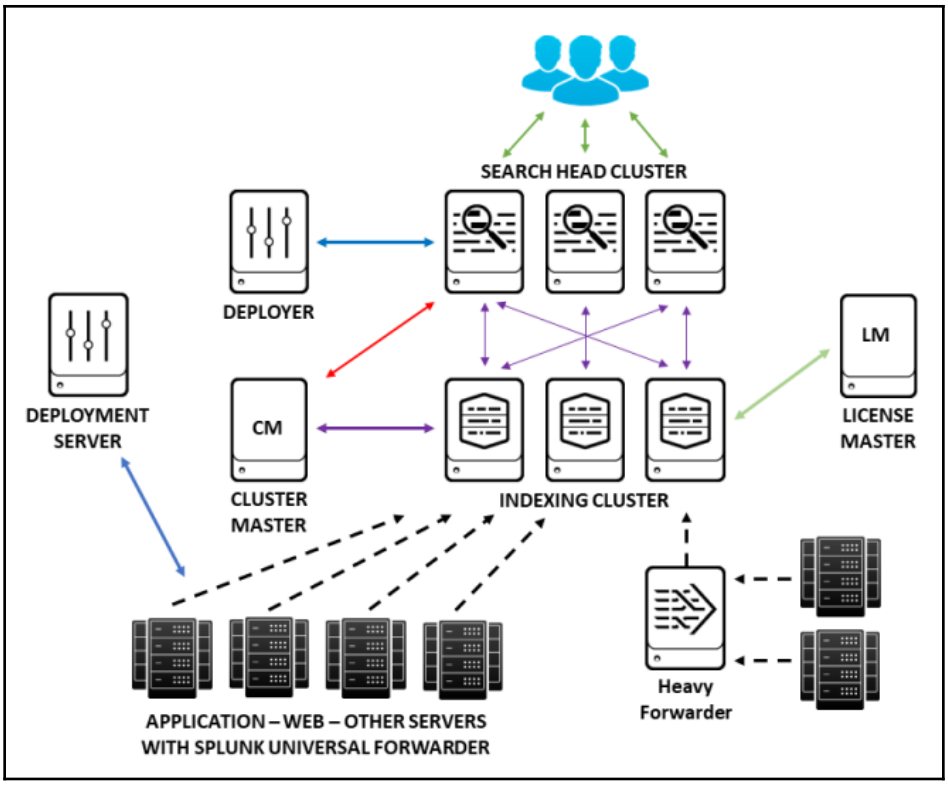
One of the first and most important things you need to learn about Splunk in order to work with it effectively is what the functional components are and how they work together. Here is a list:

- Universal forwarder
- Indexer and indexer clusters
- Search head and search head clusters
- Deployment server
- Deployer
- Cluster master
- License master
- Heavy forwarder

Universal forwarders, indexers, and search heads constitute the majority of Splunk functionality; the other components provide supporting roles for larger clustered/distributed environments. We'll summarize each of these here and dig into more details in chapters to come.

In very small installations, you can install Splunk Enterprise on a single server, which will provide all the indexing and search functionality in one instance. However, in most cases, but in most cases you will need to scale Splunk up to handle higher data volumes and more users, so the functions that we just listed will be broken out onto multiple servers to support scalability and provide redundancy for higher reliability.

The following diagram shows all the Splunk components involved in a larger Splunk deployment and how they fit together:



Splunk components in a distributed deployment

The **universal forwarder** (**UF**) is a free small-footprint version of Splunk Enterprise that is installed on each application, web, or other type of server (which may be running various flavors of Linux or Windows operating systems) to collect data from specified log files and forward this data to Splunk for indexing (storage). In a large Splunk deployment, you may have hundreds or thousands of forwarders that consume and forward data for indexing.

An **indexer** is the Splunk component that creates and manages indexes, which is where machine data is stored. Indexers perform two main functions: parsing and storing data, which has been received from forwarders or other data sources into indexes, and searching and returning the indexed data in response to search requests.

An indexing cluster is a group of indexers that have been configured to work together to handle higher volumes of both incoming data to be indexed and search requests to be serviced, as well as providing redundancy by keeping duplicate copies of indexed data spread across the cluster members. Be aware that index cluster members can be called both peer nodes and search peers in Splunk documentation, depending on context, which can be a bit confusing until you get used to the nomenclature.

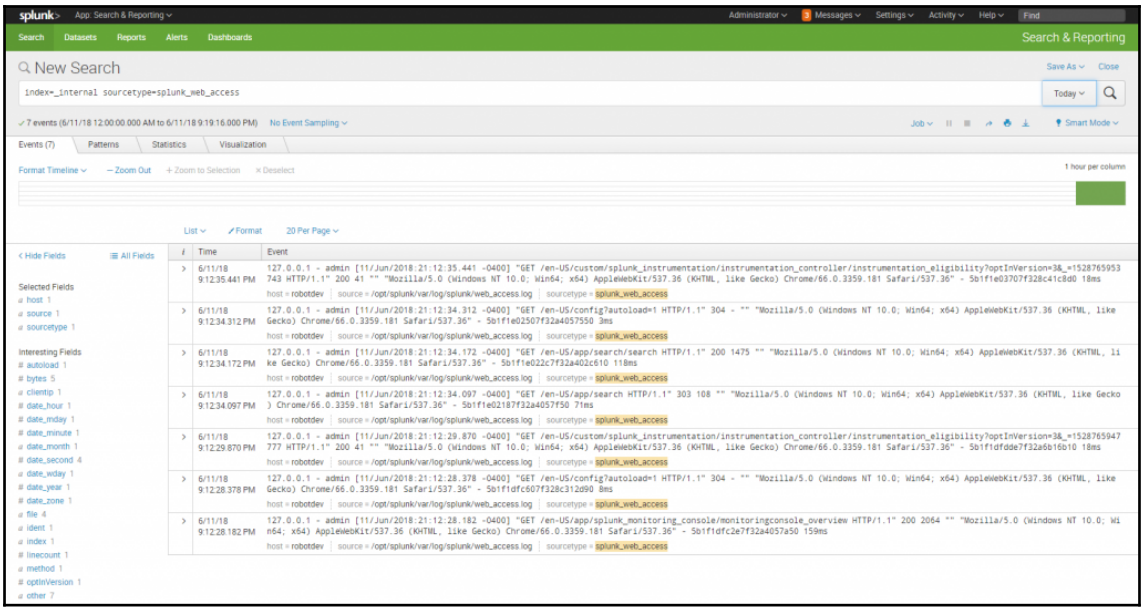
A **search head** is an instance of Splunk Enterprise that handles search management functions. This includes providing a web-based user interface called Splunk Web, from which users issue search requests in what is called **Search Processing Language** (**SPL**). Search requests initiated by a user (or a report or dashboard) are sent to one or more indexers to locate and return the requested data; the search head then formats the returned data for presentation to the user.

A search head cluster is a group of multiple search heads configured to work together to service larger numbers of users and provide redundancy for servicing search requests. Search head cluster members are called **cluster members** in Splunk documentation.

The following screenshot is an example of executing a simple search in Splunk Web. The SPL specifies searching in the `_internal` index, which is where Splunk saves data about its internal operations, and provides a count of the number of events in each log for **Today**. The SPL command specified an `index`, and then pipes the returned results to the `stats` command to return a `count` of all the events by their `source` and `sourcetype` (we'll discuss all this a bit further along in this chapter):

```
index=_internal | stats count by source, sourcetype
```

Following is the screenshot for simple search in Splunk:



Simple search in Splunk Web

A **deployment server** is a Splunk Enterprise instance that acts as a centralized configuration manager for a number of Splunk components, but which in practice is used to manage UFs. For instance, all of the hundreds or thousands of UFs that may be installed on servers in an Enterprise environment can periodically "phone home" to the deployment server to pick up new configuration information, making the task of managing all these UFs much easier.

A **deployer** is a Splunk Enterprise instance that is used to distribute Splunk apps and certain other configuration updates to search head cluster members.

A **cluster master** is a Splunk Enterprise instance that coordinates the activities of an indexing cluster. In an index cluster, data is distributed across multiple indexer instances to provide redundancy; the cluster master takes care of both controlling how that data is distributed, and helping search heads know where to direct their search requests to find specific sets of data. A cluster master is also called a *master node* in Splunk documentation.

A **license master** is a single Splunk Enterprise instance that provides a licensing service for the multiple instances of Splunk that have been deployed in a distributed environment. If you have a standalone instance of Splunk Enterprise, or a standalone indexer, you can install a license locally on that machine; otherwise, you will need a license master.

A **heavy forwarder** is an instance of Splunk Enterprise that can receive data from other forwarders or data sources and parse, index, and/or send data to another Splunk instance for indexing. A heavy forwarder has some features disabled so that it doesn't consume as many resources as an indexer, but retains most of an indexer's capability except that it cannot service search requests. A heavy forwarder is often used to parse incoming data and route it to various destinations for indexing depending upon the source and type of data received, and/or to speed up processing and reduce the processing load on indexers, but its use is optional.

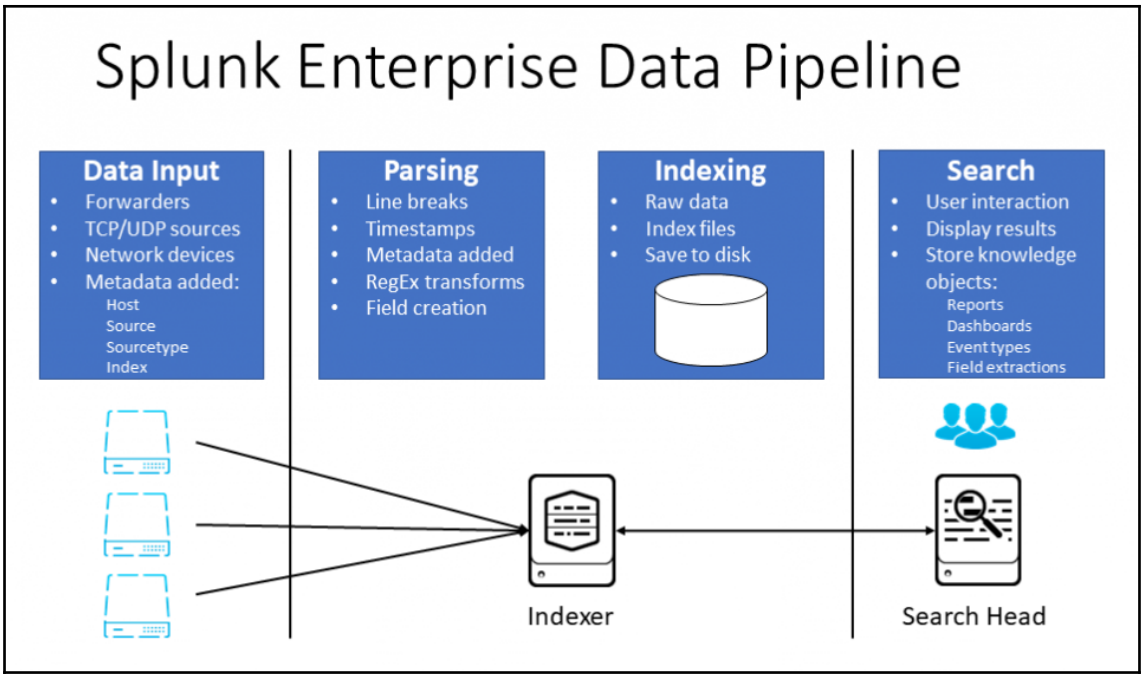
Splunk Enterprise also has a monitoring tool function called the **monitoring console**, which lets you view detailed topology and performance information about your entire distributed deployment from one interface. This function can be configured to run on one of the other Splunk components such as the cluster master or deployer if resources allow.

If this seems like a lot of functionality to absorb when you're just getting started, don't worry—you only really need to focus on getting comfortable with the data input, indexing, and search functions for now; the other pieces play smaller roles in day-to-day Splunk administration activities.

Splunk processing tiers

It is also helpful to become familiar with the Splunk processing tiers and what is called the data pipeline, which consists of the transformation functions that occur as data traverses through Splunk software, in order to—better understand how the various components are configured to work together.

As data that Splunk receives from forwarders or other data sources moves through the data pipeline, Splunk transforms it into searchable events that get indexed. The data pipeline has four segments, as illustrated in the following diagram:



Splunk data pipeline

In the **Data Input** segment, Splunk consumes the raw data stream from forwarders or other data sources, breaks it into 10K blocks, and annotates each block with metadata such as host, source, sourcetype, and the index the data is destined for per the settings in a configuration file.

In the **Parsing** segment, Splunk breaks data in these blocks into individual events, and identifies, parses, and assigns timestamps to each event. The metadata from the block of data the event came from is assigned to each event, and any required transformations to the event data or metadata are completed.

A Splunk event is a single piece of data in Splunk. Typically, an event represents a single line entry in a log file or data stream, but an event can contain multiple lines such as from a Java stacktrace. We'll cover events in more detail in the next section.

In the **Indexing** phase, Splunk writes the individual events to the specified index on disk. It writes both compressed raw data and index files; index files contain pointers to the raw data as well as some metadata to facilitate and speed up the search process.

In a distributed deployment, parsing and indexing are usually referenced together as the indexing process, as these functions are both handled on an indexer. This is okay at a high level, but if you need to inspect the processing of data more closely or determine how to scale and allocate Splunk components, you may need to consider the two segments individually.

The **Search** segment manages all the aspects of how a user searches, views, and uses the indexed data. It parses and interprets the SPL search commands, requests and receives data from indexers, and formats and presents the results to the user. The search function in Splunk also stores and uses user-created *knowledge objects* such as event types, tags, macros, field extractions, and so on.

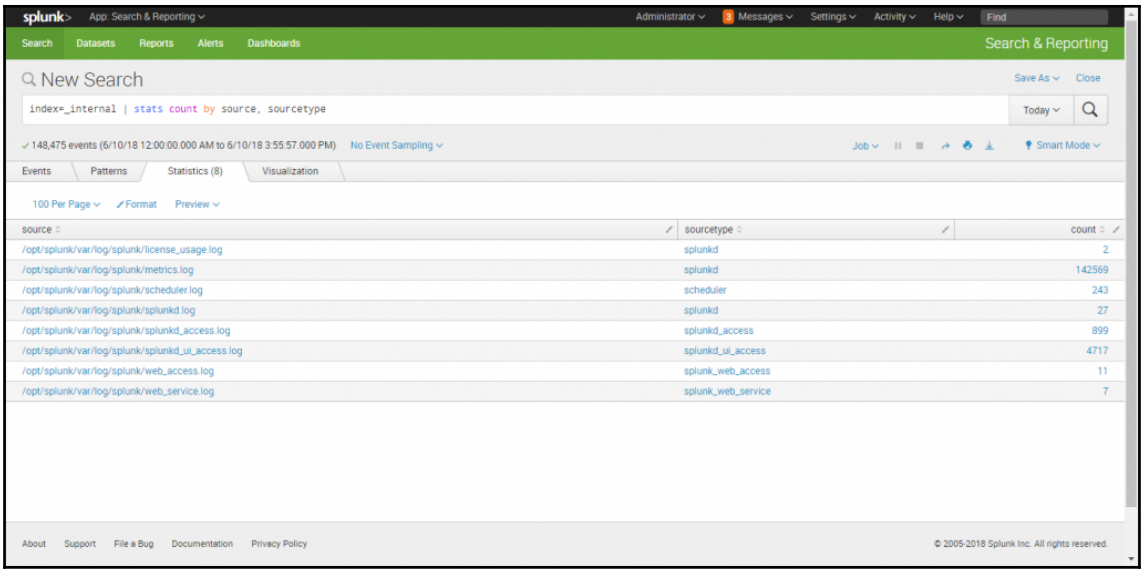
In a distributed deployment, the search function is handled on a search head; in a single-instance Splunk Enterprise server, the input, parsing, indexing, and search functions are all accomplished on that single instance, as you might expect. We will cover all these functions, and how they're configured and interact, in much more detail in the next several chapters.

Splunk events

As we discovered in the previous section, Splunk creates events from each entry in a log file or data stream. You can search for specific types of events, within specified time frames, using SPL in Splunk Web. For example, let's say you create a search on the instance of Splunk on your laptop using the SPL command:

```
index=_internal sourcetype=splunk_web_access
```

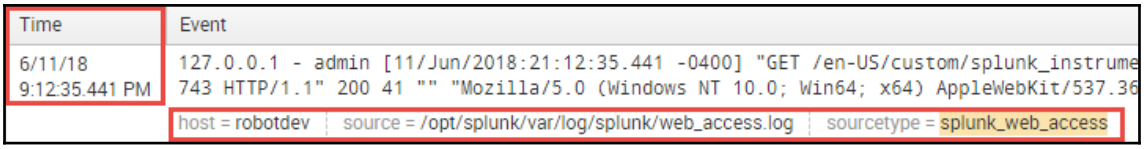
if you press *Enter*, Splunk will return a number of events for **Today** or any other time frame you've selected in the **Time Range** drop-down. These events come from Splunk's internal web server, and reflect the format and fields that are typical of a web log:



Splunk events from a simple search

We'll cover all of the features and details of using Splunk Web in `Chapter 6`, *Searching with Splunk*, so for now let's focus on some of the most important and useful fields in the events themselves.

The following is a screenshot of a typical Splunk event:



A typical Splunk event

Regardless of the data source or type, Splunk always tags each event with a number of *default* fields; some of these come from the metadata mentioned in the discussion about the data pipeline in the previous section, and others are added at index time. There are four of these fields that you will want to become familiar with right away, as they are used extensively for filtering arguments in your SPL commands to return the events of interest. In the preceding screenshot, these key fields have been circled in red – they are as follows:

- `_time` (timestamp)
- host
- source
- sourcetype

The date and time reflected in the **Time** column is the **timestamp** assigned to the event, which Splunk stores in a `_time` field. When an event contains a timestamp, as this one does, that is, **[11/Jun/2018:21:12:35.441 -0400],** Splunk will parse that timestamp and save it in the `_time` field as an epoch value (number of seconds since 00:00:00 **coordinated universal time** (**UTC**), Thursday, 1 January 1970). If an event does not contain a timestamp, Splunk will assign the time the event was indexed to the `_time` field. Splunk displays this `_time` value in the date-time format, as seen previously, corrected for the time zone specified in the Splunk Web account settings—we'll cover this in more detail in the chapter on Splunk search.

The **host** field is the name or IP address of the physical device from which an event originates. You can use this field to create filters to return events from a specific host. In the preceding example, the host is a Splunk server called **robotdev**.

The **source** field identifies where the event originated. In the case of data obtained from log files, the source consists of the full pathname of the file; in the case of a network-based source, this field contains the protocol and port, such as UDP: `514`. In this example, the event came from a Splunk log file in the `/opt/splunk/var/log/splunk` directory called `web_access.log`.

The **sourcetype** field identifies the data structure of an event (what fields the event contains, where they are, and how they're formatted), and determines how Splunk parses the data into specified fields during the indexing process. Splunk Enterprise comes with a large set of predefined source types for known data source types, and will assign the correct sourcetype to your data if it recognizes the format. You can use the sourcetype field in searches to find all the data of a certain type, regardless of the source. In the preceding example, the sourcetype is a Splunk-specific type called `splunk_web_access`.

The other important field that is not displayed in search results, but is essential for writing SPL commands to perform searches is the **index**, which as you can see was specified in the SPL command used to return the example events previously. Splunk has four internal indexes: `_audit`, `_internal`, `_introspection`, and `_telemetry`; you can view the data in these to get familiar with events in the short term. You will create and use custom indexes to store data from your company's host and device logs, and specify those indexes in your search strings.

Splunk information resources

The Splunk website has a landing page with links to a vast array of additional information on configuring and using Splunk: `https://www.splunk.com/en_us/community.html`.

Some of the more useful of these sources include the following:

Splunk documentation is, as you might expect, an extensive collection of documentation and tutorials that covers all aspects of all of Splunk's products. In the Splunk Enterprise documentation landing page, for example, you'll find tabs for **Get started**, **Search and report**, **Administer**, and so on with links to manuals for related topics. The **Admin Manual** and **Getting Data In** manuals under the **Administer** tab are particularly useful for administering Splunk.

Splunk answers is a collection of questions posted by Splunk users and administrators with answers provided by experts in the field who have solved those particular problems. If you use a search engine such as Google to look for an answer to an issue working with Splunk, here or in Splunk Docs is usually where you'll end up.

Splunk education offers a number of Learning Paths for users, Splunk administrators, architects, developers, and so on, which I highly recommend. While this Quick Start Guide is intended to introduce and guide you through the most important and useful features of Splunk from one source, and provide some practical experience and examples of common configurations and challenges you'll encounter, you will want to pursue a formal education path if you wish to obtain a deeper understanding of Splunk and achieve the various levels of Splunk certifications such as Certified User, Power User, Admin, Architect, or Developer.

I also recommend that you download, print, and peruse the Splunk Quick Reference Guide: `https://www.splunk.com/pdfs/solution-guides/splunk-quick-reference-guide.pdf`.

This is an excellent resource both to accompany the chapters that follow and to keep by your side as you perform day-to-day Splunk administration tasks, as it covers the most significant concepts and information about Splunk as well as a great deal of reference material that is helpful for performing searches using the Search Processing Language.

Summary

That wraps up Chapter 1! We've covered what Splunk is and why it's great, you got a free copy of Splunk installed on your laptop so that you can experiment freely with its files and features, and most importantly, started building a basic mental model of how Splunk works and introduced the key concepts around getting data in, parsed, and indexed. We also performed a basic search and got familiar with events and their four key fields. Finally, we covered all the resources that are available to expand your knowledge of Splunk far beyond what can be covered in a Quick Start guide.

In the next chapter, we'll go into the details of designing and implementing a Splunk Enterprise environment from scratch!

2
Architecting Splunk

If you have just started learning Splunk, it is unlikely that you would be expected to architect and implement a complex Splunk solution, especially for a larger enterprise. Typically, such projects are advisedly executed with the assistance of experienced architects from Splunk professional services, a Splunk partner consultancy, or your own in-house architects. However, you may be championing an introductory Splunk sandbox or solution at your company, or joining an existing team and need to come up to speed quickly—in which case, this overview should be helpful.

The topics that will be covered in this chapter include the following:

- Collecting the data needed for choosing an appropriate Splunk configuration
- Understanding the different types of Splunk environments
- Understanding replication and search factors
- Introduction to indexing buckets
- Considerations for search head clusters
- How to select and size the appropriate hardware options for each Splunk component
- Disk-sizing calculations

Let's start with determining the type of Splunk deployment you will build and work with.

Selecting a Splunk configuration

At a high level, the type and size of the Splunk solution you build will depend mostly on two factors:

- The volume of data you will be indexing each day
- The peak number of concurrent searches that will be running

The volume of data indexed will be the average daily total of all the inputs from your various data sources—server logs, network and security devices, and so on. The number of concurrent searches includes both ad hoc searches from users, and the number of scheduled saved searches that will be running periodically to populate reports, dashboards, and alerts. To determine these factors you will need to collect some data from your user communities and do some fairly straightforward calculations.

Data collection – data inputs

You can obtain an estimate of the amount of daily ingestion volume that you will need to handle by polling your application and other data source teams for this information; the findings of this collection will affect the number of nodes and disk storage factors involved in sizing the indexing tier of the Splunk solution.

You can collect this information with a spreadsheet or web form distributed to your application and other data source teams to complete and return to you for calculating an initial sizing; the same forms can be employed for submitting requests to ingest new data inputs down the road. The data fields collected for application and web server log files, for example, should include the following:

- **Environment**: There is typically a development and test environment, and one or more production environments, for example.
- **Application name**: This is the full name of the application for reference purposes.
- **Application ID**: There may be a short alphanumeric ID for each application in use at your company.
- **Host**: The DNS-resolvable (fully qualified domain name) hostname and/or IP address.
- **OS**: Windows, Linux, or other OS. This is helpful if you plan to install any of the Splunk apps to collect metrics from specific operating systems.
- **Middleware**: Any middleware used by this host, such as Apache Tomcat, IIS, JBoss, WebSphere, WebLogic, and so on. This is used to determine the Splunk sourcetype that will be applied for parsing the log file contents when ingesting the data.
- **Log location path**: This is the full path to the directory containing the log file(s) that will be ingested—this is needed for configuring the inputs on universal forwarders.
- **Log filename**: There may be more than one log file type and naming scheme in a logging directory; specify each naming scheme individually.

- **Daily log size** (**MB**): This can be obtained by inspecting a historical list of log file sizes (before compression and archiving) and calculating the average per-day logging volume. It is obviously important that this number is as accurate as possible.
- **PCI/PII data in the log file**: This (yes/no) field indicates the presence of **payment card information** (credit card numbers) or **personally identifiable information** (names, addresses, phone, email, and so on) in the log data that will indicate that these fields need to be masked and/or the data stored in restricted access indexes to prevent exposing confidential data to unauthorized personnel.
- **Data retention period** (**days**): The period of time the data should be stored. Typical periods are 7-30 days for dev/test, and 30-90 days, or even a year or longer for production, depending on the nature of the data, business reporting needs, and/or any regulatory requirements. This affects indexer disk storage requirements, so it should be carefully considered.

An example of a data inputs collection spreadsheet for application log files is depicted in *Fig 2.1* as shown in the following screenshot:

Splunk Data Inputs - Log Files

Environment[1]	ApplicationFullName	AppID	HostnameIP	OS	Middleware	LogPathName[2]	Avg Daily Log Size MB[3]	PCI/PII	Data Retention Days	# Hosts	Total Daily Ingestion Volume MB	Total Data Retention Volume MB	Notes
Dev/Test	AppServices	APPS00316	10.14.212.15, 10.14.212.16, 10.14.212.17	RHEL7	Tomcat8	/var/opt/apps/app_services/log/app_services.log	25	No	14	3	75	1050	includes some stack traces
Dev/Test	Product Ordering Website	EXA01577	win-dt-035, win-dt-036	WS2016	IIS	D:\inetpub\logs\LogFiles\W3SVC4*_daily.log	10	No	14	2	20	280	
									Total Ingestion Volumes:		**95**	**1330**	
Production	AppServices	APPS00316	10.26.17.201, 10.26.17.202, 10.26.17.205, 10.26.17.212	RHEL7	Tomcat8	/var/opt/apps/app_services/log/app_services.log	15	No	90	4	60	5400	
Production	Product Ordering Website	EXA01577	win-prod-52, win-prod-55	WS2016	IIS	D:\inetpub\logs\LogFiles\W3SVC4*_daily.log	10	No	30	2	20	600	
									Total Ingestion Volumes:		**80**	**6000**	

NOTES
[1] Typial entries are Dev/Test or Production
[2] Provide a separate entry for each unique log file location / name. Wildcards can be specified for variable parts of log file names
[3] Specify the average daily log file size for each application, log type, & log file such that the total logging volume for each host can be determined

Fig 2.1: Splunk data inputs—log files

Note that this spreadsheet includes cells on the right-hand side for calculations of the total daily ingestion volume and the total data volume for the specified retention period; this will be useful for calculating indexer sizing, and helps your user community to develop an appreciation for the logging volume they are imposing on the Splunk environment (and hopefully, only log what is truly needed). The formulas for these cells are as follows:

- *Total daily ingestion volume MB = (Avg daily log size MB) × (# Hosts)*
- *Total data retention volume MB = (Total daily ingestion volume MB) × (Data retention days)*

Data collection – concurrent searches

If you are building a new Splunk deployment for your company, you and your user communities may have limited estimates of how many ad hoc and scheduled searches for reports, dashboards, and alerts you can expect to employ, so this factor may be a bit difficult to nail down exactly. It may help to understand the various types of searches that are available from a Splunk perspective. We will cover searches more thoroughly in `Chapter 6`, *Searching with Splunk*, but for the purpose of gathering an estimate of the number of concurrent searches expected, you only need to be aware of the basic types of searches so that you can discuss this information with your user community:

- **Ad hoc**: These are run by a user from Splunk Web for troubleshooting, one-time investigative reports, and so on.
- **Scheduled historical search**: These are searches from a scheduled report or alert, or a dashboard that updates its panels periodically, which run against already-indexed data. This will likely represent the vast majority of the searches on your system.
- **Real-time search**: These are searches that are run against events as they are received for indexing, typically for time-sensitive monitoring. Real-time searches can be run against events before they are indexed, or just after they are indexed with a slight delay. The number of concurrent real-time searches running can greatly affect indexing performance; for this reason, only users with the admin role can run and save them by default (this ability can also be assigned to specific users or roles), and it is best to limit their use.
- **Summary indexing search**: This is a frequently running search that extracts information of interest (specific fields out of each event, for example) and saves the results into a designated summary index. You can then run searches and reports for longer reporting periods against this significantly smaller summary index (instead of against the entire range of full-sized events for the given time period) for increased performance.

Again, it is usually difficult to determine the number of expected concurrent searches because there are so many variables, and there are no hard and fast rules of thumb for estimating them. It is a good idea to sit down with each of your user communities and discuss the various types and number of reporting products they may want to leverage from the Splunk deployment. The *Fig 2.2* depicts an example of a Splunk reporting products document that you may want to prepare to aid in these discussions, and to let your user base provide some useful planning feedback:

Splunk Reporting Planner

Information to guide report - dashboard - alert planning and Splunk concurrent search calculations

			Ad-Hoc Search	Scheduled Reports		Dashboards		Alerts		Summary Indexing		Real-Time Searches	
Environment	ApplicationFullName	AppID	# Concurrent Users[1]	#	Run Time / Frequency[2]	#	Refresh Frequency[3]	#	Run Time / Frequency	#	Run Time / Frequency	#	Active Period	Run Time(s)[4]
Dev/Test	AppServices	APPS00316	4	6	Daily	2	15 Min	3	5 Min			1	2 Hours	Test Runs
Dev/Test	Product Ordering Website	EXA01577	3	2	As needed	3	5 Min	3	5 Min			1	2 Hours	Test Runs
Production	AppServices	APPS00316	2	8	Daily	3	15 Min	6	5 Min	1	5 Min	2	24 Hrs	N/A
Production	Product Ordering Website	EXA01577	3	8	Daily	3	5 Min	8	5-60 Min	1	5 Min	1	24 Hrs	N/A

[1] Number of users actively using Splunk Web to run searches at the same time

[2] How often the report/dashboard/alert is scheduled to run - Daily, Hourly, etc. - and at what time it is started each time

[3] Dashboards can be configured to automatically refresh on a periodic basis, which re-runs the searches that populate each panel

[4] What time the RT search is started/stopped

Provide a list and notes for each type of reporting product, for each Environment & Application

Environment	ApplicationFullName	AppID
Dev/Test	AppServices	APPS00316

Reports	Dashboards	Alerts	Summary Indexing	Real-Time Searches
Test Performance Summary	Performance - # logins, 200/fail, RT	Servers Down		Perf - # logins, 200/fail, RT
Testing Error Details	# Users per location - abandon rate	RT Performance Degraded		

Fig 2.2: Splunk report planning worksheet

Don't let this part of the data collection process discourage you; assuming you're planning to implement a reasonable amount of search capability to start with, and your user community isn't wanting to run an exceptionally high number of concurrent searches, this part of the data collection process isn't nearly as critical as determining the volume of data you should expect to ingest. As a general rule, you will need to add more indexers before you run out of processing capability in search heads, and if you are expanding an existing Splunk environment, you can run reports to measure the number of searches by type and compare this to the numbers of users and data sources to establish a user-to-concurrent-search ratio for better capacity planning and management.

Before we dive into using the information we've just collected to choose Splunk hardware options, there are a few more topics that need to be covered. By the way, in the discussion and examples of implementing Splunk in this book, we are going to assume the use of a **Splunk Enterprise license** since that is the most likely scenario you will be working with eventually.

Distributed versus clustered Splunk environments

It may also be helpful to define the difference between distributed and clustered Splunk deployments before we go further.

In a non-distributed, non-clustered environment, you will have Splunk Enterprise installed on a single server instance—this single machine handles all of the indexing of data and searches of that data (as well as all the other Splunk functions).

A *distributed* environment describes the separation of indexing and searching logic in Splunk. In the simplest example of a distributed environment, the indexing and search functions are divided across at least two machines—an indexer on one server that receives and indexes data, and a search head on a separate server that communicates with the indexer to service search requests – two instances, each performing a different function.

In a *clustered* environment, you would combine multiple indexers and/or search heads into an indexing/search head cluster for high availability (in case a server goes down) and data redundancy (storing more than one copy of the data across the indexing cluster). If you want to provide even better disaster recovery, you can build a multisite cluster wherein you have two indexing and/or search head clusters at different physical locations, or sites in Splunk terminology.

So, a distributed Splunk deployment does not necessarily indicate a clustered one, but a clustered deployment does infer a distributed one, since there are multiple instances (clusters) of indexers and/or search heads that perform separate functions.

If you implement a distributed and clustered environment, you will also need to implement a license master to provide licensing services to all of the separate Splunk components. If you implement a clustered indexing tier, you will need a cluster master to distribute configuration files that affect parsing and indexing operations across the indexing tier. In a similar fashion, if you implement a search head cluster, you will want to provide a deployer that manages the distribution of applications and configuration files across the search cluster. Finally, you will also want to use a deployment server to manage and distribute data input configuration files to all the universal forwarders. We will cover the specifics of all these components in later sections of this chapter.

For the examples of implementing and administering Splunk in this book, we are going to assume that we have a distributed environment that employs an indexing cluster and search head cluster, as this is the most common configuration in use at most companies.

Replication and search factor

Splunk Enterprise provides high reliability in terms of data duplication and redundant search capability by offering the ability to specify a replication factor and search factor in configuration settings for clustered environments.

Replication factor

Setting a **replication factor** is how you specify the number of *raw* data copies of indexed data you want to maintain across the indexing cluster. Indexers store incoming data in buckets, and the cluster will maintain copies of each bucket distributed across the nodes in the indexing tier (as many copies as you specify for the replication factor) so that if one or more individual indexers go down, the data still resides elsewhere in the cluster. This provides both the ability to search all the data in the presence of one or more missing nodes, and to redistribute copies of the data to other nodes and so maintain the specified number of duplicate copies.

The indexing cluster can tolerate a failure of (replication factor -1) indexers (or peer nodes, in Splunk nomenclature). If you are using a **replication factor** (**RF**) of two, the cluster maintains two copies of the data, so you can lose one peer node and not lose the data altogether; if you use an RF of three, you can lose up to two nodes and still maintain at least one copy; and so on.

The trade-off is that your cluster will need to store and process multiple copies of data. The replicating activity doesn't consume much processing power, but as the replication factor is increased, you will need more indexers and more disk storage for the indexed data.

Search factor

When you configure the cluster master node to specify the replication factor, you also designate a search factor. The search factor determines the number of *searchable* copies of data the indexing cluster maintains. The default value for a search factor is 2, meaning that the cluster maintains two searchable copies of all the data buckets. The search factor must be less than or equal to the replication factor.

The difference between a searchable and a non-searchable copy of a data bucket is that the searchable copies include both the raw data itself and the index files the indexer uses to search the data; a non-searchable copy just contains the raw data alone. Data stored as a non-searchable copy is saved in a form that makes it possible to recreate the index files later, if necessary, but the data won't be immediately searchable otherwise.

The trade-off with the search factor setting is that keeping searchable copies of data takes more disk storage than non-searchable data. The default search factor of 2 is sufficient for most situations, keeping in mind that, if a single node goes down, the cluster master works quickly to create and redistribute searchable copies elsewhere in the cluster.

In summary, the replication factor simply represents the number of copies of the raw data maintained across the indexing tier, and the search factor represents the number of copies of the index files used for searching that data that is maintained.

Hot/warm and cold buckets

There is a lot of discussion around index buckets when administering Splunk, for good reason—but it's a bit of a difficult subject to get your head around when you're just getting started, so here is a simplified, but accurate, introductory description of these concepts that you'll use the most in daily administration work, and which may warrant some consideration in the installation configuration process (regarding what disk storage to use). We'll also cover this subject again in the next chapter.

Again, incoming data is stored in indexes. Indexes have buckets, which is where event data is stored; buckets are directories organized by age. **Hot buckets** are the current files that are open and being written to; hot buckets eventually reach a size or age where they are closed and placed in a date-ranged directory, at which point they become **warm buckets**. Hot and warm buckets reside in the `. . ./myindex/db` directory; warm buckets that reach a certain age are moved to the `/colddb` directory and become **cold buckets**. Note that the cold buckets directory could reside on cheaper storage off the indexer, which comes into play when we look at sizing an indexing cluster. These cold buckets are still searchable, but searches will take longer – typically, older data is searched for less frequently, so this is not a huge issue. A final stage in the bucket lifespan is when cold buckets exceed a configured age. Then, they are moved to a frozen state and are either stored or deleted. If they were stored, they can be retrieved; when frozen buckets are opened and decompressed, they move to the `. . ./thaweddb` directory.

Hot and warm buckets are, by default, stored in `/opt/splunk/var/lib/splunk` (Linux) or `C:\Program Files\Splunk\var\lib\splunk` (Windows). Indexes reside in directories under this initial path, and under that are directories for hot and warm buckets (`. . ./<index>/db/`), cold buckets (`. . ./<index>/colddb/`), and a few other directories we won't worry about for now. For example:

```
hot bucket (files being written to)
/opt/splunk/var/lib/splunk/myindex/db/hot_v1_41
warm bucket (closed for writing, searchable)
/opt/splunk/var/lib/splunk/myindex/db/db_1530043376_1529957920_40/
cold bucket (searchable, may reside on different storage)
/opt/splunk/var/lib/splunk/myindex/colddb/db_1508276979_1508276438_0/
```

Search head clusters

A search head cluster is a group of Splunk Enterprise search heads that share configurations, search job scheduling, and search artifacts, which are the results and associated metadata from a completed search job. You will want to utilize a search head cluster in a distributed Splunk deployment to handle more users and concurrent searches, and to provide multiple search heads so that search capability is not lost if one or more search members goes down. There are a few design-related factors about search head clusters you need to be aware of while architecting and especially administering your Splunk solution, which we'll cover here.

In a search head cluster, one of the members has the role of **captain**, whose duties include, among others:

- Coordinating job scheduling: Scheduled search jobs (reports, alerts, and so on) are assigned to one of the search members by the captain when it is time for them to be run.
- Coordinating replication activities across the members: That changes to saved searches, lookup tables, dashboards, and so on made by a user on one search member are replicated to all the other members so that all of them are working with the same knowledge objects.
- Sending knowledge bundles to the search peers: A knowledge bundle is a set of knowledge objects residing on the search head that is sent to the search peers so that they can properly process search requests.

The captain will also handle its own replication tasks and assign search jobs to itself unless purposely configured not to, so it shares the search head loads just like any of the other members.

Any search member can perform the role of captain, but there is only one captain at any given time, and the search head member that is to perform the role of captain is selected by an election process between the members. If a failure occurs, the election process may be repeated and the captain can change, although a given member will usually remain the captain for quite some time unless that member itself goes down. The election process involves a member winning a majority vote of all members, and involves a random timer on each member – the member whose timer runs out first stands for the election; the other members usually comply and that member becomes the new captain. Note that Splunk uses the Raft Consensus Protocol from Stanford to manage leader election and auto-failover.

Because of how the majority vote process works, *a search head cluster should consist of a minimum of three members*. If either member of a two-member cluster fails, a captain cannot be elected and you lose the benefits of the captain's role in a search head cluster. I've introduced this topic here because you need to be aware of this constraint, and why we need three search heads to build a minimal cluster.

If you employ a search head cluster, you will also need to provide a **deployer** that distributes Splunk apps and other configurations to the search cluster members. You may also want to place the search cluster members behind a **load balancer** so that users can access the search heads through a single URL, without the need to specify a specific search head.

Splunk provides a document entitled Search head clustering architecture that you can read if you want to gain a more through understanding of clustered search head functions and considerations—it is available here: `http://docs.splunk.com/Documentation/Splunk/latest/DistSearch/SHCarchitecture`.

Making a design decision

By now, you should be fairly convinced that unless you are planning a small Splunk Enterprise deployment on a single stand-alone server, or perhaps several stand-alone indexers for point-solutions with a single search head to search across all of them, you will need to design a distributed, clustered environment that provides higher reliability and scalability.

Remember that a distributed/clustered Splunk environment can be scaled as needed by adding additional indexers and/or search heads, and you should assume that there is going to be some amount of growth over time; you may also find that your ingestion volume shortly after initial turn-up exceeds the volumes your business units tell you about, and the peak concurrent number of ad hoc and scheduled searches may exceed initial expectations as well. However, you can build a conservatively sized initial deployment with this possibility in mind, so don't worry *too* much about trying to get an exact assessment.

Depending on the findings from your poll of the user community, it may be a good idea to design an initial Splunk deployment that is quite a bit larger than your ingestion volume calculations—500 GB or even 1 TB/day of ingestion volume, for example—and let your usage grow into this solution. You can then monitor ingestion volumes and concurrent search counts and add indexers and search heads if and when needed as you gain a better feel for the particular needs of your business environment.

In the next section, we will cover how to select the appropriate hardware and disk-sizing options to accommodate your Splunk deployment, based on the decisions you have made so far.

Selecting Splunk hardware options

If you plan to implement and support your Splunk infrastructure internally (versus leveraging Splunk Cloud, wherein Splunk provides and manages the needed infrastructure for you), you will be building your solution on your own hardware platforms (physical or virtual servers), or on cloud-based instances (AWS, Azure, Google Cloud, and so on). For non-cloud environments, your organization will likely have a preferred hardware vendor that provides support options, so you will be selecting servers within the needed range of CPU, memory, and disk storage options from your vendor's offerings; cloud providers similarly offer a variety of sizing options.

Performance considerations

Splunk provides an excellent guide to all the factors and considerations for hardware options selection in their capacity planning manual; this is a ~35-page document that you will want to read before making a final choice regarding hardware options: `https://docs.splunk.com/Documentation/Splunk/latest/Capacity/IntroductiontocapacityplanningforSplunkEnterprise`.

There are several takeaways from this document that warrant emphasizing:

- If you anticipate needing to scale your Splunk deployment to any extent, you will likely be building a distributed environment with an indexing cluster and a search head cluster.
- As discussed in previous sections, the primary factors affecting the performance of your Splunk solution are the volume of data being indexed and the type and number of concurrent searches that will be encountered.
- A Splunk indexer has to split its disk resources between indexing incoming data and servicing search requests, both of which can be I/O-intensive. An indexer dedicates a CPU core to each search request, as well as taking care of other system requests and the indexing function. When the available number of available cores is consumed, all activities slow down as processing time is split between indexing and servicing search requests, and disk I/O is involved in both activities, so the practical solution to improving performance is adding more indexers.

- Splunk identifies four types of searches that impact the indexers in different ways; **dense** and **sparse** search types return between roughly 1 to 10 percent of the searched events, and tax the CPU from decompressing the raw data, while **super-sparse** and **rare** searches return a relatively smaller number of events and taxes disk I/O from the need to search through so many buckets looking for events that match the search criteria.
- Search heads aren't as disk-I/O-intensive, but in such a fashion to indexers, search heads do dedicate a CPU core to each search job. If all of the available cores are consumed, searches will be queued and search results will return more slowly; Splunk will warn you when the system reaches the maximum number of queued saved searches allowed, and it is possible that saved searches will be skipped on an overtaxed system.
- You can add more search heads to accommodate more concurrent searches, but indexer capability will likely need to be increased as well (perhaps before adding more search heads) to handle the higher number of concurrent search requests.
- If you plan to install any Splunk apps, you will want to review their installation manuals and consider the unique resource requirements for these apps in your design. A Splunk app is a prebuilt collection of dashboards and other UI elements and the saved searches that power them, packaged together for a specific technology. There are Splunk apps for a wide variety of technologies, including the Splunk app for Unix and Linux, the Splunk app for VMware, Splunk Enterprise Security, and so on. You can peruse all the Splunk apps that are available here: `https://splunkbase.splunk.com/`.

Splunk promotes the concept of a **reference server specification for dedicated search heads** and includes the following:

- 16 CPU cores at 2GHz or better per core on Intel x86 64-bit CPUs
- 12 GB of RAM
- A 1 GB Ethernet NIC
- 300 GB of RAID 1 (mirrored) disk storage
- A standard 64-bit distribution of Linux or Windows

The **reference server specifications for indexers** includes various performance levels (basic, midrange, and high-performance), such as the following:

- 12-48 CPU cores at 2GHz or better
- 12-128 GB of RAM
- A 1 GB Ethernet NIC
- A disk subsystem offering at least 800-1,200 average IOPS (input/output operations per second) with RAID 1+0 (mirroring and striping)
- A standard 64-bit distribution of Linux or Windows

For the best indexing performance, you could employ a solid-state disk (SSD) for storing hot and warm index buckets (which are instantly searchable), and relegate cold buckets (which can be searched but it takes longer) to a slower, cheaper, rotating disk.

A few more pertinent notes from the capacity manual include the following:

- You can also use **Elastic Block Storage** (**EBS**) on AWS instances, if you take care to select volume types and sizes that provide the necessary IOPS to handle Splunk Enterprise operations for searching and indexing; generally, the larger the EBS volume selected, the better the IOPS performance. You should also select an **EBS-optimized** EC2 instance to ensure adequate network bandwidth (10 GB) to support the needed throughput between the instance and EBS volume.
- The ratio of **universal forwarders** (**UFs**) to indexers that can be accommodated depends heavily on indexer performance in terms of CPUs and disk I/O, and the amount of data being sent to the indexers. Splunk conducted tests that indicated a ratio of between 2,000-5,000 UFs per indexer could be handled (keeping in mind the data ingestion volumes); performance increased when the server was configured for a higher number of Unix file descriptors – that is, three to four times the number of forwarders the indexer could accept. We will touch on Unix file descriptors later in this chapter.
- If you have indexers and search heads that exceed the reference server specifications in terms of the number of CPUs and disk I/O performance, you may be able to use *parallelization settings* that improve search and indexing performance and better leverage your hardware investment.

Making a hardware selection

After reviewing the dizzying array of specifications and performance considerations presented thus far, you may be thinking, "I just want to select the right hardware and get on with it!" So, let's narrow this down to the significant criteria that will influence your options the most for an initial deployment, make some decisions, and finalize a hardware shopping list based on the following assumptions:

- You are going to build a distributed, clustered Splunk Enterprise solution.
- You are going to deploy at least a three-search head cluster (the minimum number recommended to implement an SH cluster)—this will accommodate up to 100+ users and concurrent searches based on a rule of thumb of 20-40 concurrent search jobs per member, depending on the number of CPU cores.
- You are going to deploy enough indexers to accommodate the anticipated ingestion volume, rounded up to some increment of roughly 250 GB/day, which is the rule of thumb ingestion capability of a typical indexer; this assumption may need to be modified by your disk subsystem options – see as follows.
- You will use the Splunk reference server specifications as a guide for selecting from the hardware options available to you for search heads and indexers.
- You will use the search head reference server specifications for the various other Splunk components—license master, deployer, cluster master, and deployment server.
- Depending on the size of your deployment, you may be able to combine some of the supporting Splunk components onto one machine—the license master with the cluster master; the cluster master can also host the monitoring console, which we'll discuss in a later chapter.

The only decision factor remaining is to determine the number of indexers, which will depend on the ingestion volume, replication factor, search factor, data retention periods, and your available disk subsystem size options. Let's do that now.

Disk-sizing calculations

From the information you gathered from your line of business users, you hopefully have a general idea of how much ingestion volume your Splunk deployment will need to accommodate, and have rounded that up to some 250 GB/day increment. You also know what the retention periods will be for all the data inputs; these are typically 7-14 days for dev/test data, and 30 days or more for production data, depending on the business case. Finally, you have decided on a replication factor and search factor—we'll assume that you have selected an RF of 2 and an SF of 2. Assuming you have a list of sizing options for a suitable disk subsystem solution that meets the required IOPS and RAID specification (800-1,200) for the indexers, you can now perform the needed calculations and trade-offs to determine how many indexers you will need.

There are a number of indexer disk space calculators available on the web, but here we will again employ a spreadsheet and do the calculations ourselves. The significant factors and formulas for calculating the disk space usage for one *group* of data inputs with the same retention period (14 days) is as follows:

- RF = 2
- SF = 2
- GB/day ingestion rate = 125
- Retention in days = 14

Splunk documentation suggests that a ballpark figure for calculating the actual disk space occupied by compressed and indexed data is 50%:

- 15% for the raw data file
- 35% for index files

The formulas for calculating disk space occupancy, in progressive order and with values rounded up, are as follows:

- Base size of raw data = GB/day * .15 * Retention in days: 125 * .15 * 14 = 263 GB
- Base size of index data = GB/day * .35 * Retention in days: 125 * .35 * 14 = 613 GB
- Replicated size on disk = RF * Raw data + SF * Index data: 2 * 263 + 2 * 613 = 1,752 GB

Repeat the preceding calculations for each input group with a different retention period, then add them together to obtain the total disk occupancy. You can then divide the total disk storage needs by the per-indexer disk array size (after accounting for any RAID level reductions) to obtain the number of indexers needed to store the required volume.

A fairly fancy version of a disk-sizing calculator, which I used for calculating this example, is depicted in the *Fig 2.3*. Assuming a 500 GB/day total ingestion volume from several data input groups with differing retention periods, the ability to handle the loss of one indexer without exceeding our disk storage capability after all the missing buckets are redistributed, and with the availability of a 12-TB disk array on each indexer, we will need at least four indexers to store the 500 GB/day of incoming data for the desired retention periods.

Splunk Indexer Disk Sizing Calculator

	User Fields in Blue
Replication Factor	2
Search Factor	2
Number of Days in Hot/Warm	14
GB/Day Indexing factor	250

Data Source	GB per day	Raw Compression Rate	Index Compression Rate	Retention in Days	Base Size of Raw	Base Size of Index Files	Replicate (Y or blank)	Replicated Size on Disk in GB	Hot and Warm GB	Cold GB
Application Group 1	125.0	0.15	0.35	14	263	613	Y	1,752	1,752	-
Application Group 2	50.0	0.15	0.35	90	675	1,575	Y	4,500	700	3,800
Application Group 3	250.0	0.15	0.35	30	1,125	2,625	Y	7,500	3,500	4,000
Network Device Group 1	50.0	0.15	0.35	30	225	525	Y	1,500	700	800
Sensor Group 1	25.0	0.15	0.35	30	113	263	Y	752	351	401
Totals	500 GB/Day				2,401	5,601		16,004	7,003	9,001
Number of Indexers	4		125	ingestion volume per indexer (GB/day)						
Recovery space	if number of indexers lost =		1					4,001	1,751	2,250
Overhead	20%	for peak usage, report acceleration, summary indexes						4,001	1,751	2,250
Total Disk (GB)								24,006	10,505	13,501
							Disk Space per Indexer (GB)	6,002	2,627	3,375
							Disk Space per Indexer for Raid 1+0 (GB)	12,004	5,254	6,750

Fig 2.3: Splunk indexer disk-sizing calculator

Note that in the preceding example, the per-indexer ingestion volume is only 125 GB/day, which is roughly half (or less) of what a typical reference server indexer can handle, depending on its specifications—it is common for disk storage, and not indexer processing capabilities, to be the limiting factor that drives indexer counts up. But there is a solution — the preceding spreadsheet depicts the calculations of hot and warm versus cold storage sizes (leveraging the number of days with a hot/warm value of 14), which reveals that by utilizing some slower, cheaper, off-indexer disk storage for cold data, you can reduce your per-indexer storage needs from 12 TB to ~5.25 TB, which would allow for a smaller number of indexers and/or increased indexer utilization. The search performance for events older than 14 days would be slower, but the cost savings in terms of both storage and indexer hardware would very likely warrant the performance hit.

Once you've made your final hardware selections, and completed your disk size and indexer count calculations, you're ready to procure your Splunk servers.

Summary

Whew! We've accomplished a lot of difficult work in this chapter—collecting data from our user groups, selecting the best Splunk configuration to build, learning enough about replication and search factors, indexing buckets, and search head clusters to inform our design decisions, and finally, selecting our hardware options and calculating how much disk space we'll need and how many indexers will be needed to support out expected data ingestion volumes.

After you take a short break, we'll get started installing and configuring our Splunk Enterprise deployment. Don't be long!

3
Installing and Configuring Splunk

Now that we've made our hardware decisions and have a handful of shiny new servers just waiting for something to do, let's install Splunk Enterprise on them!

The topics we'll cover in this chapter include the following:

- How to install Splunk Enterprise on Linux and Windows platforms
- Becoming familiar with Splunk's directory structure
- Understanding Splunk's configuration file precedence and how it affects option settings
- How to configure each Splunk component to perform its specific function
- Using a checklist to ensure that a Splunk environment is properly configured
- Implementing Splunk in multiple environments—zones and cross-environment search
- How to document a Splunk solution

There's a lot to do, so let's get started!

Installing Splunk Enterprise

Installing Splunk is a fairly easy and straightforward process if you've make adequate preparations, as we did in the previous chapter. You will find a list of system requirements for your choice of operating system, as well as full instructions for installing Splunk for each type in the Installation Manual: `http://docs.splunk.com/Documentation/Splunk/latest/Installation/Whatsinthismanual`.

That is a fairly long document to read through – make sure that you at least read the system requirements section: `http://docs.splunk.com/Documentation/Splunk/latest/Installation/Systemrequirements`.

Note that if you are utilizing AWS EC2 instances for your Splunk deployment, Amazon offers an AMI (Amazon Machine Image) that includes Splunk Enterprise 7.x (at the time of writing this book), so if you pick this AMI as you spin up your instance, it will come up with Splunk installed and ready to go!

We will go through some basic installation steps in this section, and then move on to configuring Splunk on each server to perform its specific function.

Installing Splunk on Linux

You can get Splunk Enterprise for Linux on the Splunk website, starting at this URL: `https://www.splunk.com/en_us/download/splunk-enterprise.html`.

Create a free account with Splunk from this page, or log in if you already have one. On the **Choose Your Download** page, click the tab for the operating system (Linux, in this case), and select one of the packaging options—`.deb`, `.tgz`, or `.rpm`. We will choose an `.rpm` for this example, as the OS is Red Hat Enterprise Linux (RHEL) Server release 7.5 (obtained by typing `cat /etc/redhat-release` in a Terminal).

Clicking the **Download** button next to `.rpm` will start a download process, but it also reveals a link you can click to download the `rpm` using the command line (`wget`) – we'll use this option. Clicking the link opens a message box where you can copy the `wget` command (in this case, your exact filename of the rpm will vary, depending on the version of Splunk you're downloading):

```
wget -O splunk-7.1.1-8f0ead9ec3db-linux-2.6-x86_64.rpm
'https://www.splunk.com/bin/splunk/DownloadActivityServlet?architecture=x86
_64andplatform=linuxandversion=7.1.1andproduct=splunkandfilename=splunk-7.1
.1-8f0ead9ec3db-linux-2.6-x86_64.rpmandwget=true'
```

Logged in with a Terminal as root on your Linux server, from any directory (I used `/root`), paste the preceding command and press *Enter*. If you get an error message **command not found**, you'll need to install `wget` by typing `yum install wget -y`.

After verifying that `rpm` downloaded successfully, install it:

```
rpm -i splunk-7.1.1-8f0ead9ec3db-linux-2.6-x86_64.rpm
```

The `rpm` will install Splunk in the `/opt/splunk` directory, and all files should have the owner and group as Splunk. That's it!

Linux settings

Before we start Splunk for the first time, we'll need to verify and modify a few Linux settings.

User–group – environment settings

You will want to run Splunk as the `splunk` user, and the installation `rpm` should have created the `splunk` user and group, but let's check to make sure: typing the command `cut -d: -f1 /etc/passwd` will display a list of local users. If `splunk` isn't in the list, create this user and group with the `adduser splunk` and `groupadd splunk` commands.

You should be able to change the user to `splunk`: `su - splunk`. Then, ensure that the `$SPLUNK_HOME`
environment variable is set: `cd $SPLUNK_HOME`

Typing `pwd` should reveal that you're now in the `/opt/splunk` directory.

ulimits

Splunk has some specific requirements regarding the maximum file size, number of open files, user processes, and data segment sizes on the Linux platform—known as `ulimits`—that you will want to verify. From a Terminal command, type these commands (as root) and verify that the numbers meet or exceed the values given:

```
ulimit -f unlimited
ulimit -n 64000
ulimit -u 16000
ulimit -d 1073741824 (or unlimited)
```

(Or just type `ulimit -a` and read from the list.)

If your settings don't meet or exceed those values, make the following changes (for RHEL 7 running the `systemd` service):

Edit the `/etc/systemd/system.conf` file, un-comment (if needed), and change the following settings:

```
[Manager]
DefaultLimitFSIZE=-1
DefaultLimitNOFILE=64000
DefaultLimitNPROC=16000
```

Save the file and restart, then verify the settings with the `ulimit -a` command. If you're running something other than RHEL 7, you will need to search the Splunk installation and troubleshooting documents and/or the web for ulimit instructions for your installation.

Transparent huge pages

Some Linux distributions typically have a *transparent huge pages* feature that is enabled by default; this causes some fairly significant performance losses with Splunk Enterprise (reportedly 30% or more), so Splunk suggests this feature be disabled. The process for doing this varies between distributions and versions, so you may need to search the web for instructions for your particular distribution. The following link has instructions for Red Hat Enterprise Linux versions 6 and 7: `https://access.redhat.com/solutions/46111`.

The following process works for RHEL 7.5:

Check to see if THP is enabled by using (root) `cat /proc/cmdline`. Look for `transparent_hugepage=never` in the results.

If this is missing, edit the `/etc/sysconfig/grub` file and add the`transparent_hugepage=never` to the `GRUB_CMDLINE_LINUX` entry, as follows:

```
GRUB_CMDLINE_LINUX="console=ttyS0,115200n8 console=tty0 net.ifnames=0
crashkernel=auto transparent_hugepage=never"
```

Then, run the: `grub2-mkconfig -o/boot/grub2/grub.cfg` command and restart the server. When it comes up, run the `cat /proc/cmdline` command again and verify the entry.

Starting Splunk

We are now ready to start Splunk for the first time! You will want to run Splunk Enterprise as a *Splunk* user by using the: `su - splunk` command.

Then, navigate to the `/opt/splunk/bin` directory with the: `cd /opt/splunk/bin` command and type `./splunk start`.

You will be presented with a lengthy license agreement that you can spacebar through and then accept by typing `y`. Alternatively, you can start Splunk with an argument to just accept the license:

```
./splunk start --accept-license
```

You may be prompted to enter and verify an admin password, and then Splunk will go through the startup process; when it is finished, it will tell you where the Splunk Web interface is. Enter the given string into a new browser window (be sure to include the `http://` before the IP address and `:8000` at the end) and you will be presented with the Splunk login screen:

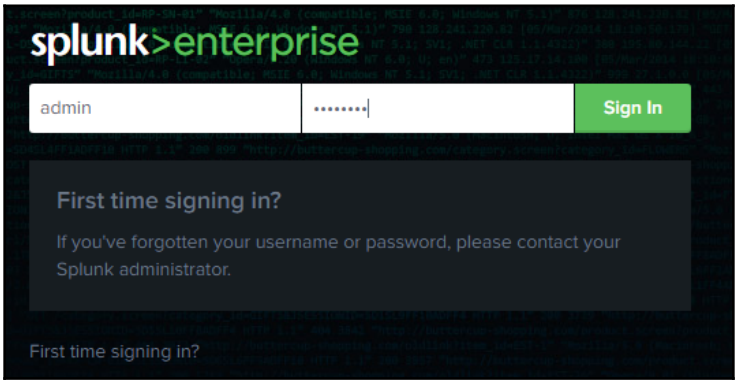


Fig 3.1: First Splunk login screen

After logging in (using the admin password provided when you started Splunk), you may be presented with a *Help us improve Splunk software* window; you can check or uncheck those options as you choose or skip it. When the page loads, click the *Search and Reporting* icon on the left-hand side; you will be prompted to **Take a quick tour** alternatively, you can skip that as well.

Congratulations!! You have a fully operational installation of Splunk Enterprise on Linux. Before we move on, there is one more task to perform at the command line.

Starting on reboot

By default, Splunk does not start when the server is rebooted; you will typically want it to do so. You can have Splunk create a script that starts it by executing an 'enable boot-start' command (as root, as this will alter OS level files):

```
[root@ip-172-31-28-164 ~]# cd /opt/splunk/bin
[root@ip-172-31-28-164 bin]# ./splunk enable boot-start -user splunk
Init script installed at /etc/init.d/splunk.
Init script is configured to run at boot.
```

Then, edit the `/etc/init.d/splunk` file and add `USER=splunk` right after the `RETVAL=0` entry near the top of the file:

```
#!/bin/sh
#
# /etc/init.d/splunk
# init script for Splunk.
# generated by 'splunk enable boot-start'.
#
# chkconfig: 2345 90 60
# description: Splunk indexer service
#
RETVAL=0
USER=splunk

. /etc/init.d/functions

splunk_start() {
 echo Starting Splunk...
 "/opt/splunk/bin/splunk" start --no-prompt --answer-yes
 RETVAL=$?
 [ $RETVAL -eq 0 ] andand touch /var/lock/subsys/splunk
}
...
```

Be sure to specify `-user splunk` when you execute the enable boot-start command, and make the noted change to the `/etc/init.d/splunk` file, or the script will start Splunk as root upon startup and cause you all kinds of file ownership issues! You can verify that Splunk is running as the *splunk user* by executing the: `ps -ef | grep splunk` command and checking to see which user (root or splunk) owns the splunk processes. If you **DO** accidentally start splunk as root, stop Splunk, cd to the `/opt/splunk` directory, and run `chown -R splunk:splunk ./` (as root) to change the ownership of all the files back to the *splunk* user. Don't fret—we've all done it, and it's easily fixed.

Stopping Splunk

Note that you should always stop Splunk from the command line before rebooting the server (as 'splunk' user):

```
/opt/splunk/bin/splunk stop
```

You can also check to see if Splunk is running:

```
/opt/splunk/bin/splunk status
```

Installing Splunk on Windows server

Installing Splunk Enterprise on Windows Server is fairly straightforward, but it does require some preparation in terms of deciding whether you're going to install it as a local system user or a member of your existing domain, in which case there are a number of considerations for access rights to directories and files. You will also need to know if you will collect data from remote Windows servers via Splunk universal forwarders and/or the **Windows Management Instrumentation** (**WMI**), as this will affect your decision as well. The Splunk Enterprise Installation Manual includes a section covering Windows installation that covers all of these factors; the discussion starts at this link: `http://docs.splunk.com/Documentation/Splunk/7.1.1/Installation/ChoosetheuserSplunkshouldrunas`.

As you read through the specifics in the installation manual, I suspect you will decide to run Splunk as a local system user, and use forwarders to collect remote data—a much more performant and simpler solution to install, manage, and troubleshoot.

The documentation discusses a couple more factors we will touch on before getting started with the installation.

Disabling antivirus software

Splunk Enterprise components (especially indexers) require high disk throughput, so you should disable or restrict any anti-virus software installed on the Windows server from scanning Splunk directories and processes.

You can remove the Windows Defender Features (Windows defender and GUI for Windows defender) on Windows Server 2016, for example, by starting **Server Manager**, selecting **Local Server** in the left-hand menu, scrolling down to **Roles and Features**, locating the Windows Defender feature, then clicking the down-arrow on **Tasks** (on the far right) and selecting **Remove Roles and Features**. You will then need to work through a series of steps in the wizard; when you get to the **Features** section, uncheck **Windows Defender Features**, click **Next**, then click **Remove**. The server will remove the software, after which you will need to do a reboot.

If you have other anti-virus packages installed (McAfee, Symantec, and so on), you will have to consult the administration manuals for those solutions. Note that you should ensure that removing or limiting the anti-virus software on your Windows platforms is acceptable with your organization's security rules before taking any action!

Installing Splunk with a short pathname

The Windows API has a path limitation of `MAX_PATH`, which is defined as being 260 characters, including the drive letter, colon, backslash, 256 characters for the path, and a null terminating character. Windows cannot address a file path that is longer than this, and if Splunk software creates a file with a path length that is longer than that, it won't be able to retrieve the file.

To avoid this problem, it would be a good practice to install the software into a directory with a short path length such as `C:\Splunk`.

Installing Splunk via the GUI

For the installation example we'll cover here, we'll install Splunk as a local system user, using the GUI instead of the command line since that's the easiest way to do it. You can get Splunk Enterprise for Windows on the Splunk website starting at this URL: `https://www.splunk.com/en_us/download/splunk-enterprise.html`.

Create a free account with Splunk from this page, or log in if you already have one. On the **Choose Your Download** page, click the tab for the operating system (Windows, in this example), and click the **Download Now** button to select the 64-bit `.msi` package. You can save the `.msi` file on the desktop or other location of preference, then just double-click it to start the installation process.

You will need to check a box to accept the license agreement, and then click `Customize Options`. The first customization step gives you the chance to change the installation directory to `C:\Splunk` instead of `C:\Program Files\Splunk`, as we discussed previously. Clicking **Next** gives you the option of changing the user Splunk will run under—leave this at *local system* unless you have made other preparations to join a domain. The next screen prompts you for a password – you must provide the Splunk admin password at this step if the installation is going to be successful – then click **Install**. When the installation is complete, you can click **Finish** with the option checked to **Launch browser with Splunk Enterprise**; the browser will start and you can login with the **admin** user and the password you provided during the installation. You may be presented with a **Help us improve Splunk software** window; you can check or uncheck those options as you choose or skip them. When the page loads, click the **Search and Reporting** icon on the left-hand side; you will be prompted to **Take a quick tour** alternatively, you can **Skip** that as well.

Congratulations!! You have a fully operational installation of Splunk Enterprise on Windows.

Stopping and starting Splunk on Windows

You can stop Splunk on Windows via the Services application (**Windows Administrative Tools** | **Services** | **Splunk Service** | **Right-Click** | **Stop**) before shutting down or rebooting your Windows server; starting Splunk is the same process but for selecting **Start**. Note that, by default, Splunk will start automatically after a reboot in Windows, unless you configure it otherwise.

Synchronization of system clocks

It is important that the system clocks on all machines that are running Splunk Enterprise and participating in a cluster—your cluster master node, peer nodes, and search heads—are time-synchronized. Otherwise, issues can arise such as timing problems between the master and peer nodes, search failures, or premature expiration of search artifacts. Most environments employ the **Network Time Protocol** (**NTP**) for this purpose; you will want to ensure that this function is configured and working correctly across all your machines.

Configuring Splunk components

If you have installed Splunk Enterprise on all of the servers you've planned for in your deployment, you're now ready to perform the changes needed to a few key files that will configure each component to perform its specific function—search head, indexer, and so on.

Before we get started with that, though, it would be a good idea to do a quick review of where Splunk keeps the files you'll be working with the most, and touch on something called 'precedence,' which is important for understanding how the configuration files work together in a complex system.

Splunk directory structure

Splunk's directory structure is identical, whether you're running on Windows or Linux, once you get past the initial installation path - `C:\Splunk` for Windows (if you took the advice on shortening the path), and `/opt/splunk` on Linux. Beyond that, there are a number of directories, as follows (Linux and Windows follow the exact same directory layout):

```
bin/        binaries, python and shell scripts
etc/        numerous - see below
include/     python include file
lib/         various links to libraries
openssl/     openssl files
share/       most of the Splunk Web UI code resides here
var/         default indexes and Splunk logs
```

The Splunk documentation often references `$SPLUNK_HOME` (as do some scripts), which is the base location for the Splunk installation – you'll want to have a mental picture that `$SPLUNK_HOME` is `/opt/splunk` for most Linux systems, and `C:\Splunk` (if you installed it that way) or `C:\Program Files\Splunk` to make such references easier to work with.

Nearly all of the configuration work you will do as you administer Splunk on a day-to-day basis will be on `.conf` files (variously named files with a `.conf` extension) residing in several directories under the `$SPLUNK_HOME/etc` directory.

Let's explore the most prominent of these locations, as you'll want to get familiar with them as soon as possible. We will be working in these directories for configuring Splunk components and all kinds of administration tasks:

- `$SPLUNK_HOME/etc/system/`: This is where system-wide configuration files reside; they control what specific function a Splunk server performs (indexer, search head, and so on) and a number of other important system settings. Note that changes you make to Splunk's configuration using Splunk Web or the CLI will be stored in various `.conf` files located in the `/opt/splunk/etc/system/local` directory.
- `$SPLUNK_HOME/etc/apps/`: This is where Splunk apps reside, including default apps like the search function, apps you might install from Splunkbase, or apps you create yourself. A location you might want to peruse to get familiar with the contents of these directories is the search app: `$SPLUNK_HOME/etc/apps/search`.

Note that under the `/system` and `/apps` directories, there is always a `/default` directory, and usually a `/local` directory. Splunk places all of its default operational settings in the `.conf` files in the `/default` directory upon installation – you should never alter any file in the default directory. Configuration changes made by an administrator or user – either by using Splunk Web, the CLI, or by editing the `.conf` files directly – should be done within `.conf` files located in the `/local` directory (create one if it doesn't already exist). Splunk then merges the contents of the files in `/default` and `/local` using precedence, which we'll cover in the next section.

- `$SPLUNK_HOME/etc/auth/`: Location of security certificates
- `$SPLUNK_HOME/etc/users`: User-specific configurations

I recommend that you explore these locations – open and read through some of the files to start getting a feel for what they contain. As you do, note that in each `.conf` file, the convention is to use *stanzas* such as [settings] and *attributes* entries under each stanza for settings pertaining to that stanza, such as this example from a `$SPLUNK_HOME/etc/system/local/web.conf` file:

```
[settings]
enableSplunkWebSSL = 1
httpport = 8000
```

Also, there is the following:

`$SPLUNK_HOME/var/log/splunk/`: This is where splunk creates and updates its own internal logs. You'll want to be familiar with the logs that are most useful for troubleshooting: `splunkd.log`, `metrics.log`, `audit.log`, and various access logs.

`$SPLUNK_HOME/var/log/introspection/`: This is where Splunk creates logs related to its `kvstore`, `splunk_disk_objects` (indexes and search-related artifacts), and `splunk_resource_usage`, which is a record of usage and performance-related metrics.

Splunk indexes its own internal log files, so you can search through them using Splunk Web – that's exactly what Splunk was made for! You can discover what source (log locations) and sourcetypes (types of log data) you can view with these search commands:

```
index=_internal | stats count by source, sourcetype
index=_introspection | stats count by source, sourcetype
```

To view the contents of a log, use one of the source or sourcetypes as a search filter:

```
index=_internal  sourcetype=splunkd
```

Configuration file precedence

Splunk uses configuration (`.conf`) files to control nearly every aspect of its operation. There are numerous configuration files *with the same name* layered inside of different directories that affect users, an app, or the system as a whole, so upon startup, Splunk merges the contents of these files based upon a directory location-based prioritization scheme to achieve an overall working configuration in memory. The rules Splunk follows when merging these files are as follows:

- When different copies of the same file have conflicting attribute values (when they set the same attribute to different values), it uses the value from the file with the highest precedence.
- The precedence of configuration files is determined by their location in the directory structure – system, app, or user directory, *in that order*. To determine priority among the numerous collections of files in the app directory, it uses lexicographical order—app directories starting with *A* have higher priority than apps starting with *B* and so on.

There is also a concept of **global** (system) versus **app or user** contexts. App and user activities, such as searching, take place in an app (search) and user (your ID or role) context to allow control over access to knowledge objects and allowable actions. Activities like indexing data work within a global context, independent of any app or user.

You will most often need to consider precedence order within the global context:

- System local directory—highest priority
- App local directories
- App default directories
- System default directory—lowest priority

Precedence order within an app or user context will make more sense as you work with Splunk apps in later chapters:

- User directories for current user – highest priority
- App directories for currently running app (local, then default)
- App directories for all other apps (local, then default)
- System directories (local, then default) – lowest priority

Configuration file precedence is a fairly complex subject, covering more areas than discussed here; if you need to understand how it works in greater depth, you can read more in the Splunk admin manual starting at this link: `http://docs.splunk.com/Documentation/Splunk/7.1.1/Admin/Wheretofindtheconfigurationfiles`.

Most of the time, you'll just be working with system and app-level configuration files, so in summary, there are two simple rules that you should follow regarding configuration file precedence that will keep you out of trouble:

- **Never** change a `.conf` file in any `/default` directory. Splunk will overwrite these files (and your changes) when you upgrade Splunk.
- **Always** create a new, empty, `.conf` file, or edit an existing one, in the `/local` directory – this copy will have precedence over a file with the same name in the `/default` directory, so changes reflected here will always supersede identical settings in a `.conf` file in the `/default` directory, and Splunk upgrades don't touch any files in the `/local` directory. Make sure that Splunk has write permissions to any files you create in the `/local` directory, as this is also where any changes to the configuration from commands you run from the command-line interface (CLI) or by using Splunk Web will be stored here.

Splunk installation checklist

Ok—we should be ready to set up all of our Splunk servers! Assuming you're configuring each Splunk component to perform its unique function manually, you'll want a checklist of the common configuration settings that all the servers receive, followed by the configuration items that are specific to each function (search head, indexer, and so on) to help ensure your configuration is correct and consistent. Let's start with a list of the components and their IP addresses, which you'll need to reference often during this process.

For this example, we're building a distributed/clustered Splunk Enterprise dev/test environment with a three search head cluster, three indexer cluster, a cluster master and license master combined on one server, a deployer, and a deployment server—a total of nine nodes.

Component and IP address list

Since you will be referencing each server's IP address fairly often during the configuration process, it helps to create a list of the servers, the IP addresses, and each server's function in a spreadsheet or other document. The following example also includes the shared security keys for various Splunk functions that will be used in the configuration steps, as well as the list of ports that will be used (and must be opened in the applicable firewalls):

Splunk Enterprise Dev/Test Configuration

Splunk Version:	7.1.1
Search Head LB URL:	https://splunk-testdev.mydomain.com:8443
Splunk Admin Password	**Splunk1t2me**
Shared Security Key for Cluster Nodes	**!Sp1unkCM!**
Shared Security Key for Search Nodes	**!Sp1unkSH!**
Shared Security Key for Forwarder Inde	**!Sp1unkID!**

Instances

Function	IP Address
CM/LM	172.31.18.102
D	172.31.28.225
DS	172.31.17.204
IDX1	172.31.28.223
IDX2	172.31.39.185
IDX3	172.31.13.169
SH1	172.31.28.137
SH2	172.31.46.250
SH3	172.31.1.45

Common/Default Splunk Ports

8000	Splunk Web port (HTTP)
8080	Splunk indexer replication port
8089	Splunk management port
8090	Splunk search head replication port
8443	Splunk Web port (HTTPS)
9997	Splunk indexer receiving port
514	Syslog
22	SSH

Splunk Enterprise Functions

CM	Cluster Master
D	Deployer
DS	Deployment Server
HF	Heavy Forwarder
IDX	Indexer
LM	License Master
MC	Monitoring Console
SH	Search Head
UF	Universal Forwarder

Fig 3.2: Component IP address and function list

Installation steps

The next part of the document is a checklist that contains the installation steps that pertain to all of the Splunk servers—in this example, for Linux. We'll continue the spreadsheet that we've just started with this list:

Installing & Configuring Splunk

Default Configuration Tasks

☐	Login via SSH, set password (if applicable)	<password>
☐	Verify root access	**sudo su -**
☐	Verify Splunk recommended ulimits	**ulimit -a** edit (uncomment) /etc/systemd/system.conf & reboot
		-f DefaultLimitFSIZE=-1 or unlimited
		-n DefaultLimitNOFILE=64000
		-u DefaultLimitNPROC=16000
☐	Verify Transparent Huge Pages disabled	**cat /proc/cmdline** "transparent_hugepage=never" in results.
		edit /etc/sysconfig/grub
		add "transparent_hugepage=never" to GRUB_CMDLINE_LINUX
		grub2-mkconfig -o /boot/grub2/grub.cfg
		reboot
☐	Install wget	**yum install wget**
☐	Get Splunk Enterprise rpm	https://www.splunk.com/en_us/download/splunk-enterprise.html
		Login -> Select OS & package type & click Download -> Copy wget command string
	Paste wget command into terminal:	**wget -O splunk-7.1.1-8f0ead9ec3db-linux-2.6-x86_64.rpm 'https://www.splunk.com/bin/splunk/**
☐	Install Splunk	**rpm -i splunk-7.1.1-8f0ead9ec3db-linux-2.6-x86_64.rpm**
		complete
☐	Verify splunk user	**cut -d: -f1 /etc/passwd**
☐	Change user to splunk	**sudo su - splunk**
☐	Verify $SPLUNK_HOME env set	**cd $SPLUNK_HOME**
		pwd /opt/splunk
☐	Verify splunk owership	**ls -al** drwxr-xr-x. 4 splunk splunk 4096 Jul 8 14:14 bin
☐	Start Splunk	**cd ./bin**
		./splunk start --accept-license
		Please enter a new password: **Splunk1t2me**
		... The Splunk web interface is at http://172.31.18.102:8000
☐	Verify processes running as 'splunk'	**ps -ef | grep splunk**
		splunk 1524 1519 0 14:31 ? 00:00:00 [splunkd pid=1519] splunkd -p 8089 start [process-runner]
☐	Login to Splunk Web from browser	**http://<ip address>:8000**
		username: admin
		password: Splunk1t2me

Fig 3.3: Default installation task checklist

After the basic installation is completed successfully, complete the post-installation tasks:

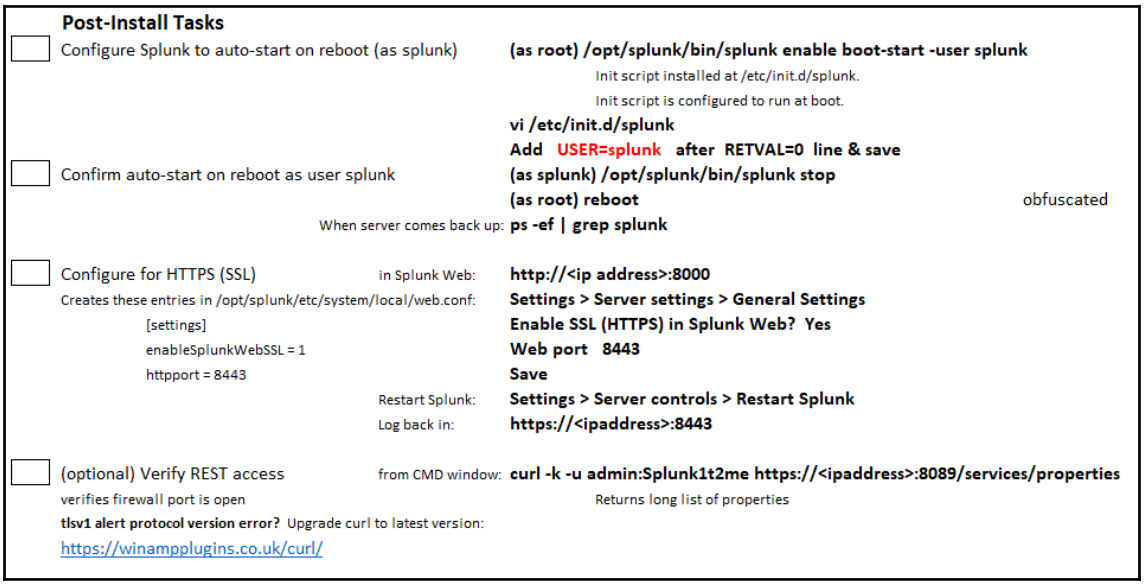


Fig 3.4: Post-installation task checklist

Configuring Splunk to use HTTPS (SSL) for Splunk Web is the first configuration task we didn't cover previously, and is the first example of configuring Splunk from the GUI interface. From Splunk Web, select **Settings** | **Server settings** | **General Settings**. The following screenshot shows the part of the page where you will click **Yes** on the question *Enable SSL (HTTPS) in Splunk Web?* and set the Web port to `8443`; then, click **Save** and the offered button to restart:

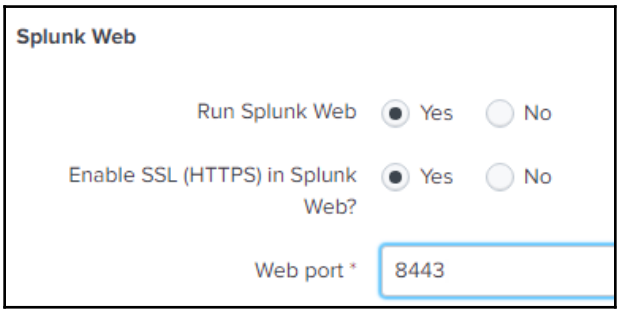


Fig 3.5: Configuring Splunk Web to use SSL

Individual component configurations

The configuration steps and settings to be completed next depend on the Splunk function that a particular server will perform – let's go through these settings to complete the initial Splunk deployment, component by component. Most configuration settings can be accomplished from the Splunk Web GUI or via the CLI, but some can only be accomplished by editing a specific `.conf` file. We will do as much as is practical from the GUI.

You may want to add these steps to your checklist, if you choose to make one, so that you have a complete record of how your Splunk deployment was configured for future reference. I like to copy/paste the contents of especially the `server.conf`, `inputs.conf`, and `outputs.conf` files off to the side of my configuration checklist spreadsheet, and any other files altered during the process.

License master and cluster master

You can create a **license master**, which is a centralized license server for the entire deployment, simply by installing the Splunk Enterprise license you obtained from Splunk sales on it:

1. (Splunk Web)(license master): **Settings** | **Licensing** | **Add new license** | **Choose File** | **Install**
2. Click the provided button to restart Splunk

After logging back in, navigate to **Settings** | **Licensing** and verify the license was installed. You'll also note at the top of this page that *This server is acting as a master license server* – so there's nothing else to do in terms of configuring a license master. During this operation, the license file you imported was saved in `/opt/splunk/etc/licenses/`.

To enable this node as a **cluster master**, perform the following steps in Splunk Web:

- **Settings | Indexer clustering** —click **Enable indexer clustering**
- Select **Master node** (default) and click **Next**
- Set the replication factor to **2** and leave the search factor at **2** (or the RF/SF of your choice)
- Enter `!Sp1unkCM!` in the Security Key field (or a security key of your choice)
- Enter `DevTestIndexers` in the Cluster Label field (or an entry of your choice)
- Click **Enable Master Node** and **Restart Now**

When the server comes back up, you can view the `/opt/splunk/etc/system/local/server.conf` file and note the following settings that weren't there before; in most cases, you can tell what function a Splunk server is performing just by inspecting the `server.conf` file. The `[lmpool:...] and [license]` stanzas reflect the default license pool settings and the fact that you're using a Splunk Enterprise license; the [clustering] stanza reflects the choices made for clustering. Note that the `pass4SymmKey` contains an *obfuscated* version of the security key you entered (`!Sp1unkCM!`) during setup; Splunk does this to all plain text security keys upon startup:

```
[lmpool:auto_generated_pool_enterprise]
description = auto_generated_pool_enterprise
quota = MAX
slaves = *
stack_id = enterprise

[license]
active_group = Enterprise

[clustering]
cluster_label = DevTestIndexers
mode = master
pass4SymmKey = $1$J2nsIHghS61r0NU=
replication_factor = 2
```

Forwarding Splunk's internal logs to the indexers

On ***all Splunk nodes except the indexers***, we want to have all of Splunk's internal logs forwarded to the indexers instead of indexing them locally; this reduces disk space usage and it makes the internal logs for all the Splunk nodes searchable without having to log into each node individually – you can determine which node a specific log entry came from by the host field.

In the `/opt/splunk/etc/system/local` directory, create an `outputs.conf` file and add the following entries (replacing the example `<ipaddress>:9997` entries with the correct IP addresses and receiving port, if different) for your indexers. We'll let this node pick up this configuration upon the next restart of Splunk *after* you've set up the indexers. Remember to perform this step on all of your nodes (except indexers):

```
[indexAndForward]
index = false

[tcpout]
defaultGroup = dev_test_indexers
forwardedindex.filter.disable = true
indexAndForward = false

[tcpout:dev_test_indexers]
server=172.31.28.223:9997,172.31.39.185:9997,172.31.13.169:9997
```

We're done with this node for now. We needed these components to be in place first so that we can point all the other servers to the license master for licensing, and point the indexers to this cluster master node during their setup.

Pointing servers to the license master

For ***all Splunk nodes except the license master*** (and cluster master, in this example, since they reside on the same server), you'll need to configure the node to get its licensing from the license master. This can be easily done from Splunk Web, as follows:

1. Click **Settings | Licensing | Change to slave | Designate a different Splunk instance as the license master**
2. Enter the license master URI: (`scheme:IPAddress:mgmt_port`) `https://172.31.18.102:8089`
3. Save and restart

This creates the following stanza and setting in the `/opt/splunk/etc/system/local/server.conf` file:

```
[license]
master_uri = https://172.31.18.102:8089
```

Indexing cluster

For each of the indexers (peer nodes) in the indexing cluster, perform the following steps in Splunk Web:

1. **Settings** | **Indexer clustering—**click **Enable indexer clustering**
2. Click **Peer node** and **Next**

In the Peer node configuration screen, complete the following fields (examples provided):

1. Master URI: `https://172.31.18.102:8089`
2. Peer replication port: `8080`
3. Security key: `!Sp1unkCM!`
4. Click **Enable peer node**

This creates the following entries in the `/opt/splunk/etc/system/local/server.conf` file:

```
[replication_port://8080]

[clustering]
master_uri = https://172.31.18.102:8089
mode = slave
pass4SymmKey = $1$bIw25oPe24yYuqk=
```

Configuring a TCP input

On ***all of the indexers***, you will need to configure a TCP input for receiving the forwarded internal logs from the other Splunk servers; this input can service forwarded data from universal forwarders as well:

1. **Settings | Forwarding and receiving | Configure receiving | New Receiving Port (button)**
2. Listen on this port: `9997`
3. **Save**

Splunk will create or append an `inputs.conf` file in `/opt/splunk/etc/system/local/` with the following content:

```
[default]
host = 172.31.28.223

[splunktcp://9997]
connection_host = ip
```

That completes the cluster master and indexing tier—let's set up the clustered search environment next.

Deployer

The deployer is used to to distribute apps and configuration files to the search head cluster, and when a search head that was down comes back up, it contacts the deployer to see if it has missed any updates. So, let's build that component before setting up the search heads.

Assuming you've configured the deployer to get its licensing from the license master and to forward internal logs to the indexers, there is only a minor configuration change to be made on the `server.conf` file in `/opt/splunk/etc/system/local` to prepare this node to be a deployer. These settings cannot be accomplished from Splunk Web:

```
[shclustering]
shcluster_label = DevTestSearchHeads
pass4SymmKey = !Sp1unkSH!
```

The `shcluster_label` entry is a label of your choice that identifies the search cluster in the Monitoring Console (covered in a later chapter). `pass4SymmKey` is entered in plain text, and Splunk will obfuscate this entry upon the next startup; this is the secret key the search head cluster members will use to communicate with the deployer and each other.

Search heads

After configuring the licensing and forwarding the internal logs, you can configure a node to be a member of a search head cluster, either by executing a CLI command, or editing the `server.conf` file directly—there is no provision to make all these settings in Splunk Web. Let's try the CLI command `splunk init shcluster-config` first, providing all the needed elements in the arguments:

- `auth`: The admin-level username and password.
- `mgmt_uri`: `scheme:ipaddress:port` of the node you're configuring.
- `replication_port`: A port you've selected for the search heads to replicate knowledge artifacts with each other.
- `replication_factor`: The number of searchable copies of data (index files) to be retained on the indexers (default is 2).
- `conf_deploy_fetch_url`: The `scheme:ipaddress:port` of the deployer – this is so that the search head knows who to contact to get updates if it was previously down.
- `secret`: The same `pass4SymmKey` password you configured on the deployer - use this same secret key on all the search heads, too. If you use certain non-text characters in your password, you may need to wrap this entry in single quotes.
- `shcluster_label`: The same cluster label you configured in the deployer.
- Here's what the command looks like fully filled out, with entries for our example deployment:

```
./splunk init shcluster-config -auth admin:Splunk1t2me -mgmt_uri
https://172.31.28.137:8089 -replication_port 8090 -replication_factor 2 -
conf_deploy_fetch_url https://172.31.18.102:8089 -secret '!Sp1unkSH!' -
shcluster_label DevTestSearchHeads
```

The preceding command creates the following entries in the `/opt/splunk/etc/system/local/server.conf` file. Note that the `disabled = 0` entry is needed because search head clustering is disabled by default in `/opt/splunk/etc/system/default/server.conf`:

```
[replication_port://8090]

[shclustering]
conf_deploy_fetch_url = https://172.31.28.225:8089
mgmt_uri = https://172.31.28.137:8089
```

```
pass4SymmKey = $1$q3Fg5DtBkC6yGZA=
replication_factor = 2
disabled = 0
shcluster_label = DevTestSearchHeads
```

Next, you need to configure the search heads to communicate with the cluster master, and thus join them to the indexing cluster. The cluster master provides a list of the search peers (indexers) to the search heads so that they can contact the active indexers for search requests. Here's the CLI command to run on each search head:

```
./splunk edit cluster-config -mode searchhead -master_uri
https://172.31.18.102:8089 -secret '!Sp1unkCM!'
```

Note that you're specifying the cluster master's URI, and providing the **index cluster** secret key. Running this command will add the following entries to the `server.conf` file:

```
[clustering]
master_uri = https://172.31.18.102:8089
mode = searchhead
pass4SymmKey = $1$q3Fg5DtBkG23GZA=
```

You can configure nodes to be a clustered search head by directly editing the `server.conf` file with the correct entries; sometimes, this is easier if you're sure of the entries to be made. After restarting Splunk on these servers, you will have a search head cluster!

Designating and starting a search head captain

The first time you bring up the search head cluster, you will need to execute a command on one of the search heads to trigger the cluster into electing a captain; the `server_list` in this command is a `scheme:ipaddress:mgmt_port` list of all your search heads, separated by commas and enclosed in quotes:

```
./splunk bootstrap shcluster-captain -servers_list
"https://172.31.28.137:8089,https://172.31.46.250:8089,https://172.31.1.45:
8089" -auth admin:Splunk1t2me
```

Checking search head cluster status

You can check the status of your search cluster with the following command, executed on any of the search heads:

```
./splunk show shcluster-status -auth admin:Splunk1t2me
```

If your cluster is running properly, you will get back a status for the captain and each of members (again, including the server that is performing the captain's role)—for example:

```
-bash-4.2$ ./splunk show shcluster-status

 Captain:
                              dynamic_captain : 1
                              elected_captain : Mon Jul  9 02:48:52 2018
                                           id : 14E5BDB5-5B74-4D50-92ED-86A2CD00E020
                                initialized_flag : 1
                                        label : ip-172-31-28-137.ec2.internal
                                     mgmt_uri : https://172.31.28.137:8089
                           min_peers_joined_flag : 1
                            rolling_restart_flag : 0
                              service_ready_flag : 1

 Members:
        ip-172-31-46-250.ec2.internal
                                        label : ip-172-31-46-250.ec2.internal
                           last_conf_replication : Pending
                                     mgmt_uri : https://172.31.46.250:8089
                                mgmt_uri_alias : https://172.31.46.250:8089
                                        status : Up
        ip-172-31-28-137.ec2.internal
                                        label : ip-172-31-28-137.ec2.internal
                                     mgmt_uri : https://172.31.28.137:8089
                                mgmt_uri_alias : https://172.31.28.137:8089
                                        status : Up
        ip-172-31-1-45.ec2.internal
                                        label : ip-172-31-1-45.ec2.internal
                           last_conf_replication : Pending
                                     mgmt_uri : https://172.31.1.45:8089
                                mgmt_uri_alias : https://172.31.1.45:8089
                                        status : Up
```

Fig 3.6: Search head cluster status

Deployment server

The deployment server is most often used to manage apps and configuration files on universal forwarders. We will cover how to configure and use this node in the next `Chapter 4`, *Getting Data into Splunk*, so for now all you need to do is complete the configuration changes to use the license master and to forward internal logs to the indexers.

Multisite environments

Implementing a multisite deployment for increased redundancy requires a few additional considerations and a few configuration changes on the nodes in three areas: the cluster master, the indexing cluster, and the search head cluster. Although there are CLI commands to configure some of these settings, it is generally easier just to edit the `server.conf` directly.

Cluster master

On the cluster master, add the following to `server.conf`:

Under the `[general]` stanza:

- Identify the site the cluster master is in with an `entry: site = site1`

Under the `[clustering]` stanza:

- Enable the cluster for multisite with an entry: `multisite = true`
- Set the number of sites for the `cluster: available_sites = site1,site2`
- Set the multisite replication factor :`site_replication_factor = origin:2,total:3`
- Set the multisite search factor: `site_search_factor = origin:1,total:2`

For the site replication and search factors, the origin is the site where the data first entered the cluster; the number associated with the origin is how many copies of each bucket are to be kept on that site, and the total is the total number of copies of each bucket that are across all the sites in the cluster. If you have more than two sites, you can specify the number of copies kept at each site (see the Splunk docs). From the preceding example, the `server.conf` file would contain the following:

```
[general]
site = site1

[clustering]
mode = master
multisite = true
available_sites = site1,site2
site_replication_factor = origin:2,total:3
site_search_factor = origin:1,total:2
pass4SymmKey = !Sp1unkCM!
cluster_label = DevTestIndexers
```

Indexers

You can also modify the `server.conf` on each indexer – under the `[general]` stanza:

- Identify the site the indexer is in with a `site = site1` (or site2) entry

The settings under the `[clustering]` stanza remain the same as those used on single-site peers:

```
[general]
site = site1

[replication_port://8080]

[clustering]
master_uri = https://172.31.18.102:8089
mode = slave
pass4SymmKey = !Sp1unkCM!
```

Search heads

Finally, modify the `server.conf` on each search head—under the `[general]` stanza:

- Identify which site the search head belongs to (it can only belong to one): `site = site1`

Under the `[clustering]` stanza:

- Indicate that this search head is running in a multisite environment: `multisite = true`

The resultant entries in the `server.conf` (along with other previous entries) are as follows:

```
[general]
site = site1

[replication_port://8090]

[shclustering]
conf_deploy_fetch_url = https://172.31.18.102:8089
mgmt_uri = https://172.31.28.137:8089
pass4SymmKey = $1$q3Fg5DtBkC6yGZA=
replication_factor = 2
shcluster_label = DevTestSearchHeads

[clustering]
```

```
multisite = true
master_uri = https://172.31.18.102:8089
mode = searchhead
pass4SymmKey = $1$q3Fg5DtBkG23GZA=
```

There is a feature called *search affinity*, where searches normally run only across search peers in the same site as the search heads. This occurs naturally if you have copies of searchable data in both sites, as the configuration in the preceding examples provides.

Also note that if you are deploying a search head cluster across two sites, your primary site should contain a majority of the search head nodes. If there is a network disruption between the two sites, only the site with the majority of search heads can elect a new captain, and you'll want the primary site to retain a captain node if it is still operational.

You can review all of the considerations for multisite configurations in Splunk's docs before setting this up:

- `http://docs.splunk.com/Documentation/Splunk/latest/Indexer/Multisitedeploymentoverview`
- `http://docs.splunk.com/Documentation/Splunk/latest/Indexer/Multisiteconffile`
- `http://docs.splunk.com/Documentation/Splunk/latest/Indexer/Multisitesearchaffinity`

Cross-environment search

If you want to enable searching across more than one Splunk environment—say, two or more different production environments that may have their own indexing and search clusters (but not a multisite situation), you simply point the search head(s) at the cluster master(s) for each additional environment in `server.conf`, using multiple linked stanzas, as in the following example:

```
[clustering]
mode = searchhead
master_uri = clustermaster:prod1, clustermaster:prod2

[clustermaster:prod1]
master_uri=https://Master1.example.com:8089
pass4SymmKey=secretKeyForTheProd1IndexingCluster

[clustermaster:prod2]
master_uri=https://Master2.example.com:8089
pass4SymmKey=secretKeyForTheProd2IndexingCluster
```

To configure searching across multisite clusters, you will need to set the two multisite-specific attributes – site and multisite – in the `[general]` and `[clustermaster]` stanzas; the placement varies depending on which of various configurations you have, so I will refer you to the Splunk docs for specifics and examples: `https://docs.splunk.com/Documentation/Splunk/latest/Indexer/Configuremulti-clustersearch`.

Documenting your Splunk deployment

It is essential to create and maintain documentation for your Splunk deployment so that administration personnel can understand and effectively troubleshoot the environment, and assist in further architectural planning. This documentation should include, as a minimum, an accurate diagram depicting all the Splunk components and how they are connected, and a record of the significant configuration files (`server.conf`, `web.conf`, and so on) that governs the specific functionality of each component.

The diagram of the Splunk deployment should include each environment (dev/test and production, and so on) and depict the different sites if applicable. Within each environment, you can use an icon and/or text box representing each of the Splunk components with the following information associated with each component (as a minimum):

- Function(s) (search head, indexer, and so on)
- FQDN hostname and IP address

Index and search head clusters can be grouped together; multiple environments or multisite configurations should be clearly depicted. The diagram should also depict any load balancers that are placed in front of search head clusters, any heavy forwarders placed between universal forwarders or other data sources, and so on – a comprehensive picture of the Splunk environment.

An example diagram of the dev/test Splunk deployment featured in the examples used in this book is provided in *Fig 3.7*:

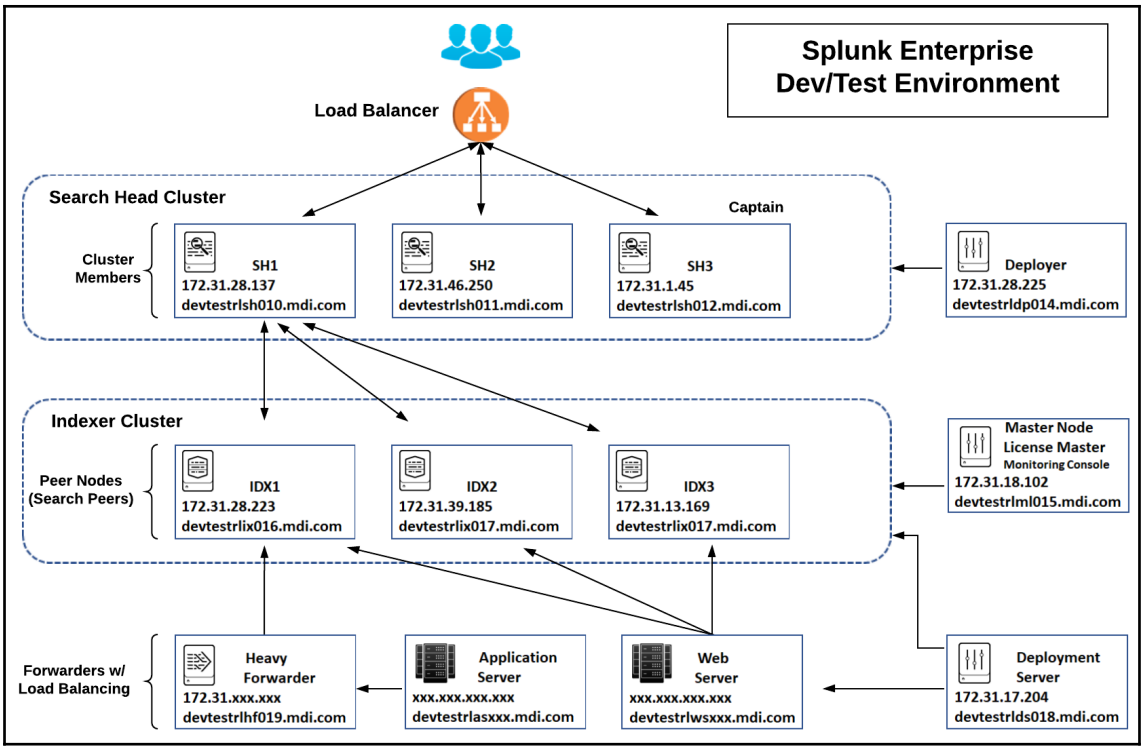


Fig 3.7: Splunk Enterprise dev/test deployment diagram

As we mentioned previously it is helpful to keep copy/paste or file captures of the contents of the significant configuration files—mostly those located in `/opt/splunk/etc/system/local`—someplace where they can be referenced later. Larger companies will likely want to leverage their CMDB and/or employ an automation solution to record and control their Splunk configurations.

Summary

We've accomplished a lot in this chapter as well—installing, configuring, and documenting an entire distributed/clustered Splunk Enterprise deployment! I hope that this process, as well as the explanations about some key aspects of Splunk functionality, has helped you start to build a solid foundational understanding of how all the Splunk components work together.

In the next two chapters, we will dive into administering Splunk data inputs, indexes, apps, and users – this will be fun!

4
Getting Data into Splunk

In this and the following chapter, you will learn how to administer your Splunk deployment, starting with getting data into Splunk. In this context, the term *administration* includes a wide range of initial setup as well as the day-to-day tasks involved in getting data into Splunk, properly parsed and indexed; managing the indexing and search head clusters so that the data can be searched, deploying Splunk apps, and setting up users and their roles so that they can access the data.

The topics covered in this chapter include the following:

- Installing and configuring universal forwarders to send log data to Splunk
- Setting up a heavy forwarder
- Configuring inputs from other types of data sources
- How to configure an HTTP Event Collector (HEC) to input data
- How to configure Splunk to properly parse nonstandard data formats
- How to distribute configuration files to forwarders from a deployment server
- How to configure and manage event and metrics indexes
- How to manage an indexing cluster from a cluster master

There's a lot to cover, but before we get started, let me mention that while you can configure most Splunk features using either Splunk Web or the command-line interface (CLI), any changes will result in new entries or modification to existing entries in various `.conf` files, which becomes the configuration record that Splunk will read into memory upon a restart. Furthermore, if you are working with a distributed/clustered Splunk environment, which is the primary assumption for this book, there are numerous settings that cannot (or should not) be set using Splunk Web (index configurations, for example), and for some of these the configurations have to be made in the appropriate `.conf` files and distributed to the search head cluster members or search peers (indexers) via CLI commands. For this reason, I will generally focus on how to configure Splunk features directly in `.conf` files, how to deploy them (if applicable) via CLI commands, and only cover using Splunk Web for those functions where it makes the most sense to do so in terms of time or convenience.

It is my experience and opinion that having a good grasp on how Splunk configuration options are reflected in the various `.conf` files will serve you better in the long run.

I'll also note that this chapter covers a lot of topics that are interdependent; it is difficult to decide which subjects to cover first to avoid having to reference topics we haven't fully discussed yet, such as apps and indexes, for example, when those topics have to be included when setting up inputs. You may want to read through this chapter to pick up the concepts and then reread some or all of the topics to solidify your understanding of how this all works together.

Installing Splunk universal forwarder

Since much of the data you will ingest into Splunk will likely come from the logs of application and web servers, we'll need to install a universal forwarder on a server and configure it to *monitor* specific logs and send that data to Splunk for indexing, along with some configuration settings that tell Splunk how to parse the logs and which index to store the data in.

Installation steps

The Splunk universal forwarder is basically a specialized instance of Splunk Enterprise with most features disabled, and it is a separate binary, but you can follow the same process as was used for installing Splunk Enterprise in `Chapter 3`, *Installing and Configuring Splunk,* for both Linux and Windows installs.

You can download the Splunk Enterprise universal forwarder from this link: `https://www.splunk.com/en_us/download/universal-forwarder.html`.

Select the Windows or Linux tab, click the download button for the 64-bit version, and format as appropriate for your operating system. For Linux, when installing Splunk Enterprise, you can download and save the `rpm` or other binary, but you can also copy the `wget` command to install the package from your server by clicking on **Download via Command Line (wget)**:

```
wget -O splunkforwarder-7.1.2-a0c72a66db66-linux-2.6-x86_64.rpm
'https://www.splunk.com/bin/splunk/DownloadActivityServlet?architecture=x86
_64&platform=linux&version=7.1.2&product=universalforwarder&filename=splunk
forwarder-7.1.2-a0c72a66db66-linux-2.6-x86_64.rpm&wget=true'
```

On a Linux server, `cd` to `/tmp` or similar location (as root) and paste the preceding command. When the download is complete, install the forwarder:

```
rpm -i splunkforwarder-7.1.2-a0c72a66db66-linux-2.6-x86_64.rpm
```

The installation is quick, and you'll only see a **Complete** message when it's done. Navigating to the `/opt` directory should reveal a `splunkforwarder` directory owned by *Splunk*.

On Windows machines, you simply download the `.msi` and double-click to install; accept the License Agreement, provide an admin password, and work through the wizard steps. You will be prompted to enter the IP or DNS and port for the deployment server (first prompt) or a receiving indexer (next prompt); in this case, we'll provide the IP and port for the deployment server, and click **Next**:

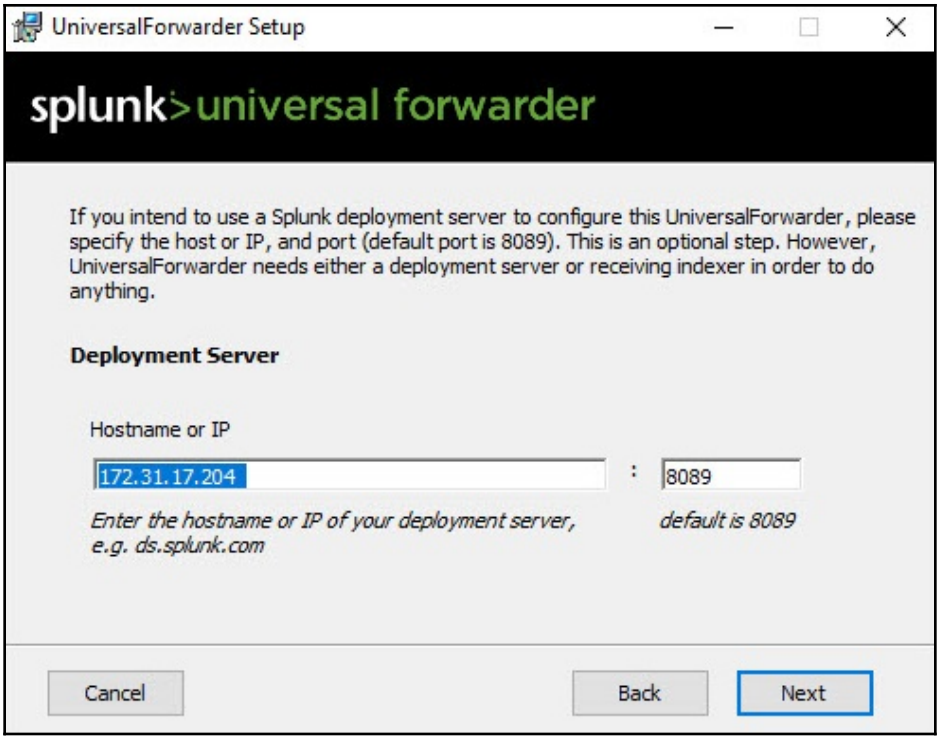


Fig 4.1- Specifying a deployment server

Leave the **Receiving Indexer** fields blank in the next panel, click **Next**, then **Install**, and then **Finish** when it's done.

Starting/stopping the universal forwarder

You start, stop, and restart the forwarder the same way as you start/stop/restart Splunk Enterprise. On Linux, note that we'll avoid having to read and accept the license during the startup with the `--accept-license` argument:

```
$ sudo su - splunk        don't forget this step!
$ cd $SPLUNK_HOME/bin
$ ./splunk start --accept-license
```

You will be prompted for the admin password, and after a few messages where the UF creates some additional directories and checks some configuration files, there should be a *done* report.

On Windows, the UF should already be running after the installation, and a quick check of the services app should also confirm that it is configured to start automatically, so there's nothing else to do.

On Linux, however, you will need to configure Splunk to auto-start upon reboot, by executing the following command (as root, usually):

```
./splunk enable boot-start -user splunk
```

Then, you must edit the `/etc/init.d/splunk` file to add the `USER=splunk` line under the RETVAL entry.

Over the next few sections, we'll discuss configuring the `outputs.conf`, `inputs.conf`, and `deploymentclient.conf` files for use on a forwarder. If you plan to manage your forwarders using the deployment server, you don't need to configure `outputs.conf` and `inputs.conf` on the forwarder itself; these can be distributed later using the deployment server—but we'll cover them now just in case. You will, however, need to configure the `deploymentclient.conf` file on the forwarder in either case.

Configuring outputs.conf

We now need to tell the forwarder which index(ers) to send their data to; you can do this from the command line:

```
./splunk add forward-server <indexer hostname or IP>:9997 -auth
<username>:<password>
```

Alternatively, you can do this by creating and configuring the `outputs.conf` file that this command would create in the `$SPLUNK_HOME/etc/system/local` directory on the forwarder. In this example, we'll configure the forwarder to send data to all three of our indexers in a load balanced (one indexer at a time, round-robin) fashion. Note that we also specify the `useACK` option to tell the forwarder to wait for an acknowledgement from the indexer that the data was received before sending any more. Also note that the receiving port on the indexers is, by default, `9997`, although you can configure it to use any port you like as long as the `inputs.conf` on the indexers are looking for your forwarded data on the same port:

```
[tcpout]
defaultGroup = indexer_cluster
useACK = true

[tcpout:indexer_cluster]
server = 172.31.28.223:9997,172.31.39.185:9997,172.31.13.169:9997
```

After configuring this file, you can restart the forwarder; although we haven't specified any inputs yet, the forwarder will still send its internal logs to the indexers. You can confirm this is working by running a search that lists all the Splunk servers you're ingesting log data from and making sure your fowarder hostname or IP shows up:

```
index=_internal | stats count by host
```

Configuring inputs.conf

Now, we will configure an `inputs.conf` file to monitor log files and send the entries to Splunk for indexing. In this example case, we have installed a forwarder on an Apache web server running on a Linux host; the web server log files reside in the `/var/log/httpd` directory – you can see that there is an active and a rolled-over log for both the access and error logs:

```
[root@ip-172-31-39-242 httpd]# pwd
/var/log/httpd
[root@ip-172-31-39-242 httpd]# ls -l
-rw-r--r-- 1 root root 86698 Aug 5 22:24 access_log
-rw-r--r-- 1 root root 576950 Aug 5 02:43 access_log-20180805
-rw-r--r-- 1 root root 1000 Aug 5 14:08 error_log
-rw-r--r-- 1 root root 3374 Aug 5 03:16 error_log-20180805
```

To monitor these logs, we need to edit the `inputs.conf` file that was created in the `/opt/splunkforwarder/etc/system/local` directory when the Splunk forwarder was installed, with the following contents:

```
[default]
host = ip-172-31-39-242

[monitor:///var/log/httpd/access_log]
index = weblogs_90d_eidx
sourcetype = access_combined
ignoreOlderThan = 30d

[monitor:///var/log/httpd/error_log]
index = weblogs_90d_eidx
sourcetype = apache_error
ignoreOlderThan = 30d
```

The `[default]` stanza contains a host entry for this forwarder; this is what Splunk will display in the `host` field of each event for logs originating from this server. Note that if you decide to manage forwarders with a deployment server, you will need the `inputs.conf` file to remain where it is as it contains the host field that gets sent with events – in other words, don't delete this `inputs.conf` file.

The `[monitor]` stanza specifies what logs to monitor; this also becomes the *source* metadata field you will see in events. The format for log locations in Linux is:

```
[monitor://(path to log file(s))
Ex:
[monitor:///var/log/httpd/access_log]
```

For Windows, the format is the same, except you use backslashes instead of forward slashes for the path specifier:

```
[monitor://C:\Windows\System32\WindowsUpdate.log]
```

You can specify the index the monitored events are sent to, and the sourcetype to assign to each log type after the monitor stanza. The sourcetype tells Splunk what format the log entries take, so it knows how to parse out the timestamp and the various fields. The `ignoreOlderThan` parameter tells the forwarder not to send events older than 30 days, which avoids ingesting data older than your intended retention period and incurring the licensing charge, only to discard the data because it exceeds your retention period:

```
index = web_90d_eidx
sourcetype = access_combined
ignoreOlderThan = 90d
```

You'll note that the entries in `inputs.conf` sets the four imperative fields found in every Splunk event: host, index, source, and sourcetype. If the index isn't specified, Splunk will send the events to the default *main* index. If the sourcetype isn't specified, Splunk will attempt to recognize the appropriate sourcetype from the fields format of the event and assign one of a set of predefined sourcetypes that Splunk knows about.

In the preceding exapmle, I actually determined the applicable sourcetypes by ingesting some events without specifying them in `inputs.conf`, observed what sourcetype Splunk assigned to each type (`access_log` was `access_combined`; `error_log` was `apache_error`), and added the applicable entries in `inputs.conf` afterwards. For well-known log types such as these, you don't *have* to provide the sourcetype, as Splunk will try to match the log format to one of its known sourcetypes, but it is good technical practice to do so.

Ellipsis and asterisk wildcards can be used if there is more than one subdirectory to monitor logs in, and/or more than one log in a given directory – the monitor stanza that follows uses both methods for the example case where the log paths vary, but the same sourcetype could be assigned to both logs, and both logs are to be stored in the same index:

```
/var/log/myapp/system/default/scripts/runtime.log
/var/log/myapp/system/instance/bin/cpp/run.log

# both of the above will be picked up by the following entry
[monitor:///var/log/myapp/.../*.log]
```

You can discover more options for `inputs.conf` files in Splunk docs at the following links:

- `http://docs.splunk.com/Documentation/SplunkCloud/latest/Data/Monitorfilesanddirectorieswithinputs.conf`
- `http://docs.splunk.com/Documentation/SplunkCloud/latest/Data/Specifyinputpathswithwildcards`
- `http://docs.splunk.com/Documentation/Splunk/latest/Data/Listofpretrainedsourcetypes`

As I mentioned before, if you are going to use a deployment server to manage your forwarders (recommended), you will NOT want to configure the `inputs.conf` and `outputs.conf` files in `$SPLUNK_HOME/etc/system/local` on your forwarder, as outlined previously. You will instead configure the `inputs.conf` and `outputs.conf` files in one or more *apps* that are placed on the deployment server; these apps will be downloaded to your forwarder and placed in the `$SPLUNK_HOME/etc/apps` directory. So, at this point, you may want to set up the *deployment client* settings on your forwarder so that it contacts the deployment server periodically to pick up the applicable app(s). I'll introduce the concept of *apps* shortly, and cover the deployment client settings in the *Using the deployment server* section, further along in this chapter.

As a final note—if you have problems getting the Splunk forwarder to read log files on *nix systems due to access permissions, you may need to execute the following command (as root):

```
sudo setfacl –R –m u:splunk:rX /path/to/logs
The –R switch will apply permissions recursively
The –m is to modify the existing ACL
The u:splunk specifies the splunk user
The rX grants read access to everything, and sets the execute bit only on
files with an existing execute bit flipped.
```

Setting up a heavy forwarder

A heavy forwarder is presented in Splunk docs and other sources as being a type of forwarder that sends data to another Splunk Enterprise instance or to a third-party system. It is also suggested that a heavy forwarder can have a smaller footprint than a Splunk Enterprise indexer (by disabling some services such as Splunk Web), but retains most of the capabilities of an indexer. Finally, the docs will state that a heavy forwarder parses data before forwarding it and can route data based on criteria such as a source or type of event, and that it can index data locally while forwarding data to another indexer. However, you won't find a download file for a *heavy forwarder* on the Splunk site, nor will you find a specific document or direct references to setting up a heavy forwarder.

I'll offer another perspective on the subject of heavy forwarders that might help clear up any confusion resulting from such verbiage: a heavy forwarder is just another instance of Splunk Enterprise. There is nothing unique about this instance except that it is not configured to be a pure indexer, nor a search head, by entries in the `server.conf` file. Nor is it used (although it unadvisedly could be) to provide license master, deployer, or deployment server services. A heavy forwarder can, however, be configured to do any or all of the functions stated previously by the proper configuration of its `inputs.conf` file (to allow it to accept data from universal forwarders or other data sources), and `outputs.conf` file (to send data it has collected to a single indexer or a cluster of indexers in load-balanced fashion, or even a third-party non-Splunk receiver), along with possibly some other settings depending on the application.

A heavy forwarder can be configured to parse the data it receives— as an indexer would before indexing the events—to perform the masking of sensitive PII or PCI data, extract, discard, and/or only forward specific data or events to reduce indexing volume, or any number of other specialized preprocessing tasks before sending the data on to the indexing tier, or another destination based on specified criteria. You *could* index data on a heavy forwarder, and then forward that data on to an indexing tier, but since Splunk licensing is accounted for by indexing volume, this would result in accruing double ingestion charges, so I suspect a viable use case for this is rare.

Another common and valuable use for heavy forwarders is to host specialized Splunk applications such as DB Connect (which can query and index database tables, or perform Splunk searches and send the results to databases), the Splunk App for AWS (interfaces with various AWS services), or any number of other apps available from Splunkbase. For most of these applications, you will want to leave Splunk Web enabled on the heavy forwarder so that you can administer the application.

So, in summary, a heavy forwarder is a specially configured instance of Splunk Enterprise. If your user case calls for using a heavy forwarder, you can install Splunk Enterprise on a properly sized server using the steps provided in the previous chapter (remember to set ulimits, disable transparent huge pages, and enable boot-start), and then proceed to install the Splunk app and/or configure the forwarders' inputs and outputs as needed for your use case. The specifications for your heavy forwarder (CPU, memory, disk) will depend upon the app(s) you install, and how you plan to use it, but you should build it to *reference server* specs as a minimum (16 cores, 12 GB RAM, 300 GB disk). We will cover several scenarios that call for a heavy forwarder in later sections and chapters of this book.

One final note about heavy forwarders: unlike universal forwarders, which simply monitor files and forward the unparsed data to the indexing tier, a heavy forwarder that is monitoring local files must be configured to parse the log data into events using `props.conf`, and optionally, `transforms.conf`, just as an indexer would. The parsed events are then simply forwarded to the indexing tier for storage in the specified index. This note will make more sense when we cover custom sourcetypes and indexing later in this chapter – just remember that a heavy forwarder does the job an indexer normally would do in a universal forwarder scenario.

Configuring other data source inputs

Some devices such as firewalls, routers, and switches do not generate logs directly; instead, they send their log and event data over network ports to, typically, a syslog server that stores the data in log files in some directory. The best practice for handling this scenario is to install a universal forwarder on the syslog server so that you can configure multiple inputs (one for each type of data) in `inputs.conf` to assign the appropriate index and sourcetype for each data source type. You will also need to assign a host in the `inputs.conf` file for each input, or do some research on how to leverage DNS to identify the proper hostname for each data source (which is outside the scope of this book). However, if you simply must stream this data from the devices directly to Splunk, there are two approaches that will work.

One approach is to send the device data to a Splunk heavy forwarder, and let the heavy forwarder send the data on to the indexing cluster in a load-balanced fashion. If you have data from multiple types of devices, and you use the standard UDP:514 syslog socket, you will have to do some research to configure Splunk to recognize each data type and assign the appropriate sourcetype, as well as the host assignment.

Another, less desirable approach is to send the device data directly to an indexer. This has the disadvantage of data loss if you restart the indexer (you'll likely be restarting indexers more often than you would a heavy forwarder), as well as the possibility of causing an indexing tier storage imbalance on the target indexer. If you have multiple indexers and multiple data sources, you may be tempted to point each source to a different indexer–which would work, but you'll have to carefully (and manually) manage the `inputs.conf` file on each indexer as they will not be consistent.

Regardless of the method you choose to terminate your streaming data sources, you will configure Splunk to accept syslog data by making the appropriate entries in the `inputs.conf` on either the heavy forwarder or indexer(s), as follows:

```
[udp://514]
index = cisco_30d_eidx
sourcetype = syslog
```

You may have to run Splunk as `root` instead of `splunk` on Linux to be able to access privileged ports such as `514`, or route data from `514` to another/higher port using routing rules in `iptables`. Did I mention it's much easier to use a universal forwarder on a syslog server?

You can also set up data inputs using Splunk Web by selecting **Settings** | **Data inputs** | **UDP** | **New**, selecting **UDP** (or **TCP**, if appropriate), entering the **port number**, and clicking **Next**. On the next **Input Settings** page, click the **Select Source Type** drop-down and select from the various categories and selections, or click **New** and configure a custom sourcetype; then, click **Review** and **Submit**.

You can also configure an input from the CLI by using the following format:

```
$SPLUNK_HOME/bin/splunk add udp 514 –sourcetype syslog
```

Configuring a syslog or other streaming data input scenario obviously has a number of factors that should be considered –you will want to search the web for solutions that most closely fit your situation, as well as review the subject in the *Getting Data in Splunk* document: `https://docs.splunk.com/Documentation/Splunk/latest/Data/Monitornetworkports`.

Finally, you should consider using a technology-specific Splunk app such as the Cisco Networks Add-on to help manage your syslog inputs from network or other devices. We will cover Splunk apps and add-ons in more detail in `Chapter 9`, *Splunk Applications*.

Configuring an HTTP Event Collector

Another very versatile and highly scalable way of getting data into Splunk is via the HTTP Event Collector (HEC), which is a solution that listens for HTTP requests containing JSON objects. The HTTP Event Collector can collect data at extremely high volumes from many devices and data sources, all on a single port. Another interesting feature of using HEC is that the host, index, source, and sourcetype associated with a given data source can be specified within the JSON object of each received event.

The HTTP Event Collector uses a token-based authentication model; you configure a new token in Splunk, and give that token to your application developers who then include it in each event sent to Splunk.

To set up the HTTP Event Collector, you first configure the global settings, which includes the HTTP endpoint it will listen on – by default, this is port `8088`. You can also (and should) configure a default index, and optionally a default sourcetype. This is one activity that is best done in Splunk Web the first time, so you can see all the options available to you; later, we'll look at where those option selections end up in the associated `.conf` files. To edit the global settings from Splunk Web, click **Settings** | **Data inputs** | **HTTP Event Collector**, then click **Global Settings**. Click **Enabled**, select a **Default Index** (such as main), and click **Save:**

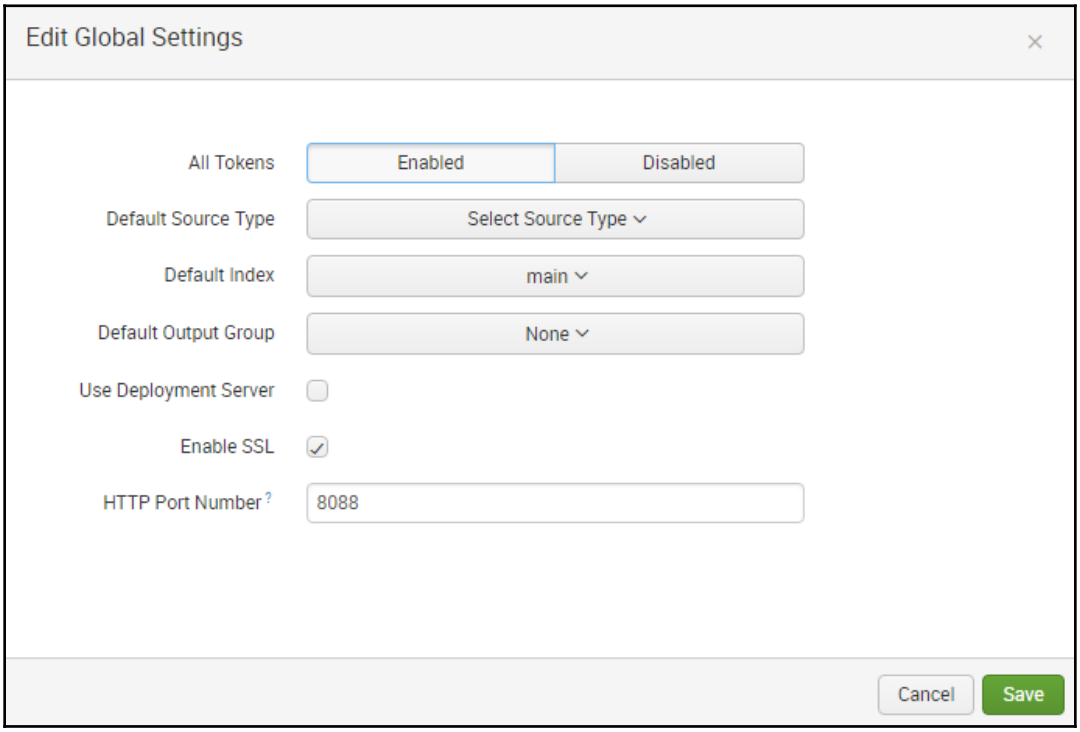


Fig 4.2: HTTP Event Collector Global Settings

Next, click **New Token** and enter the **Name** and optional **Description** for this input, then click **Next**. If you are configuring the HEC on a heavy forwarder and this data is going to be forwarded to a specific indexing tier (and you have configured the indexer(s) in an `outputs.conf` with one or more `[tcpout:indexer_cluster]` or similar group stanzas), you can select the desired output group (in case there's more than one) here as well. Finally, clicking **Enable indexer acknowledgement** will acknowledge receipt of each bundle of data, in case your sending application is configured to utilize this feature:

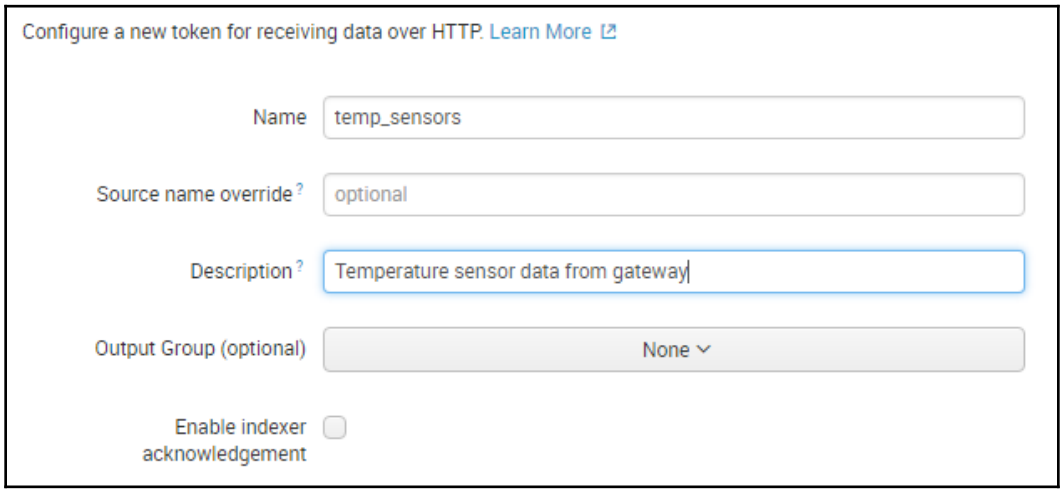


Fig 4.3: HTTP token name

On the **Input Settings** page, you can click the drop-down and select the sourcetype to be associated with the incoming events, allow Splunk to automatically assign a sourcetype, or click **New** and configure a unique sourcetype; we'll discuss how to configure unique sourcetypes later in this chapter. For this example, I clicked **Select** | **Select Source Type** | **Metrics** | **metrics_csv**. From the **App Context** drop-down, I selected **Search & Reporting (search)** as the app directory that this token's settings are to reside in.

Finally, I want to make sure that this sensor data can only go into an index named **sensors_180d_midx,** so I clicked on that entry in the **Available item(s)** box so that it appeared in the **Selected item(s)** box, then clicked on **sensors_180d_midx** in the **Default Index** drop-down:

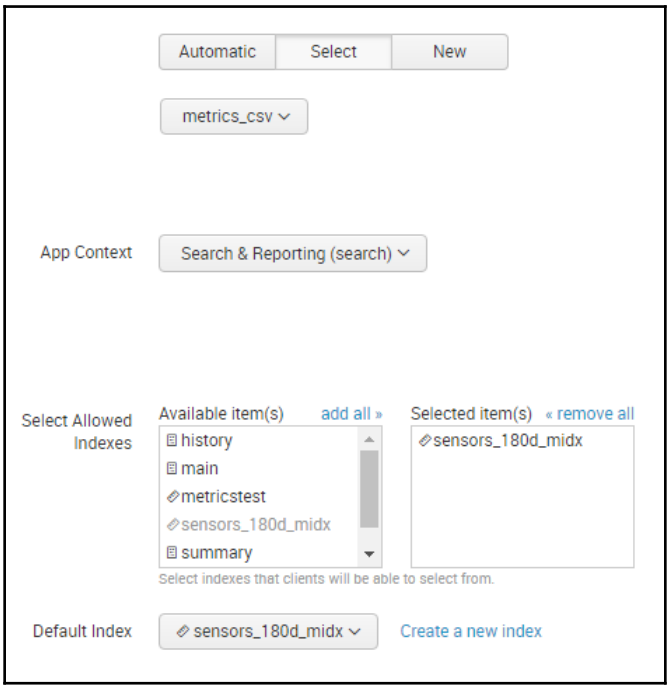


Fig 4.4: HTTP token setup – Input Settings

Finally, click **Review**, ensure that the settings are what you want, and click **Submit**. You will be presented with the token value, which you can give to your developers:

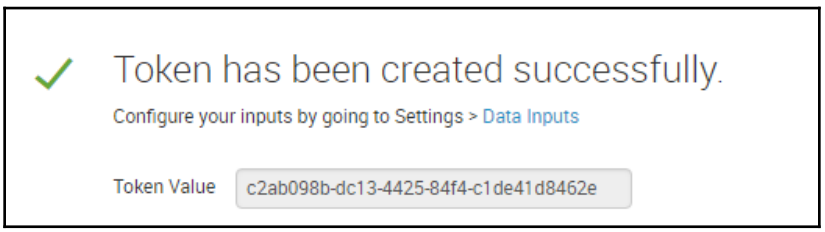


Fig 4.5: HTTP token

The settings we have just completed resulted in the creation of two `inputs.conf` files.

The global settings were preserved in `/opt/splunk/etc/apps/splunk_httpinput /local inputs.conf`; basically, entries to enable HEC (`disabled = 0`), and the default index. In this example, I have purposely altered the HEC port to `8888` to depict how a non-default port is reflected in this file; typically, there is no port entry because you're using the default port of `8088`:

```
[http]
disabled = 0
index = main
port = 8888    // this is usually 8088
```

The token-specific settings are, in this case, located in `/opt/splunk/etc/apps/search /local/inputs.conf`, because we chose **Search & Reporting (search)** as the app that is to contain the configuration. You can see that all of our selections appear here: `index =` is the selected default index, whereas `indexes =` is a list of the only indexes the data is allowed to go to, just in case the developer mistakenly specifies an index you *don't* want the data to go into in the event JSON:

```
[http://temp_sensors]
description = Temperature sensor metrics
disabled = 0
index = sensors_180d_midx
indexes = sensors_180d_midx
sourcetype = metrics_csv
token = c2ab098b-dc13-4425-84f4-c1de41d8462e
```

Testing the HTTP Event Collector

You can verify and test your HEC settings with the `curl` command, which is usually available on most Linux distributions and can be installed on Windows. Replace the `<token>` with the token you obtained during the setup; when you run the `curl` command, you should be creating an event in the default index for that HEC token:

```
curl -k https://<host>:8088/services/collector -H 'Authorization: Splunk
<token>' -d '{"sourcetype": "mysourcetype", "event":"Hello, World!"}'

example:
curl -k https://192.168.1.8:8088/services/collector -H 'Authorization:
Splunk 985fdede-1bcf-4ae8-a7f5-8d6b89dd2c56' -d '{"sourcetype":"test",
"event":"Hello World!"}'
```

Note that you'll get errors in Windows when you use single quotes (`'`), so change them to double quotes (`"`) and escape the other double quotes with a backslash (`\"`):

```
curl -k https://192.168.1.8:8088/services/collector -H "Authorization:
Splunk 985fdede-1bcf-4ae8-a7f5-8d6b89dd2c56" -d
"{\"sourcetype\":\"mysourcetype\", \"event\":\"Hello World!\"}"

You should get the following response:
{"text":"Success","code":0}
```

If you run a search for the default index the HEC token is configured for, you will see your test event.

You can read more about the HTTP Event Collector at the following link, which also provides information that your developers will need to reference for configuring their application to use HEC: `http://dev.splunk.com/view/event-collector/SP-CAAAE6M`.

Introduction to apps

We will cover Splunk *apps* in much more detail in `Chapter 9`, *Splunk Applications*, but because they are such a significant part of many Splunk administration tasks, I need to provide at least an introduction to what Splunk apps represent to avoid confusion when I reference apps in the tasks we'll be discussing shortly. Also, Splunk's use of the term may not reflect what you're accustomed to thinking of as an *app* – so it's important to address this sooner rather than later.

An app created by a Splunk user, or any of the apps that come with Splunk, contain a collection of all the objects that make up the app and how it functions within the Splunk environment. These objects can include configuration (`.conf`) files to control inputs, indexes, saved searches, reports, dashboards, alerts, and more, as well as static files for creating web browser views and menu navigation items. Each of these objects have permissions associated with them to determine who can view or alter them.

From an administrative point of view, a Splunk app is a collection of files that resides in a folder under `/opt/slunk/etc/apps/`; each app has its own folder under this path. The best way to grasp how this works is to inspect the contents of one of folders for an app that comes with Splunk. For instance, the *launcher*, which is a fairly simple app that controls the view and menu when you first log in to Splunk (unless you've selected another Default Dashboard), or the `search` folder, which contains all the files used to manifest the **Search & Reporting** interface.

Let's take a quick look at the *launcher*, as it contains many of the typical folders and contents for a basic app:

```
/opt/splunk/etc/apps/launcher
   /appserver        // html, css, images to support a view
   /default          // default .conf and navigation, view files
   /local            // .conf files from user settings
   /metadata         // permissions and app owner info
   /README           // additional info on the app
```

The more complex search app, by comparison, contains these and several other folders for lookups, scripts, and so on. But an app can be as simple as just an app folder, with a `/local` folder inside, that contains a single `inputs.conf` or `indexes.conf` file–there's no minimal requirements for contents, although there is a certain structure to follow as the app grows in size and complexity. For example, you will see *apps* referenced in the discussion of using the deployment server in the next section–these too are folders under `.../apps`, but they may just contain a few configuration files that are distributed to universal forwarders. So, as you can see, Splunk apps can vary significantly in size and contents depending on the complexity of the functionality they support.

A Splunk app is fully contained within its directory–if you delete an app directory, the app is gone–there are no other files in other locations to be concerned about.

Using the deployment server

A deployment server is the Splunk system that distributes apps, configurations, and other assets to universal or heavy forwarders, and in some environments, other Splunk Enterprise instances. You'll typically use the deployment server just to distribute and maintain the `inputs.conf`, `outputs.conf`, and possibly a number of other configuration files and apps on all your universal forwarders so that you don't have to maintain them on a server-by-server basis manually.

It is not a requirement that you use a deployment server; you can use an external tool such as Windows System Center Configuration Manager, or chef, puppet, or salt if your deployment runs on *nix servers. However, a deployment server is the fastest and most native way to get apps and configurations deployed to your Splunk universal forwarders.

Again, the most typical use of a deployment server is to distribute configuration files to the hundreds or thousands of universal forwarders you may find in many companies. You can also use the deployment server to distribute config files to non-clustered indexers, but you cannot use it to update indexer cluster peer nodes (that's done with the cluster master) or search head cluster members (accomplished with the deployer), so we'll just focus on managing universal forwarders in this discussion. You can, however, use the deployment server to distribute updates to the master node (cluster master), which then uses what is known as "the configuration bundle method" to distribute them to the peer nodes, but we won't do that here.

First, a few definitions:

- Splunk defines a remotely configured Splunk instance such as a universal forwarder as a **deployment client**
- A deployment client can belong to one or more **server classes**, which is a configuration category such as the OS or application
- A unit of content that gets deployed to members of a server class is called a **deployment app**

You need to understand the update process that occurs when using the deployment server, which is as follows:

1. Each deployment client periodically polls the deployment server – this is called *phoning home* (configured in `deploymentclient.conf` on each client/forwarder)
2. The deployment server checks the set of deployment apps that should reside on the client, based on which server classes the client belongs to (configured in `serverclass.conf` on the deployment server)
3. The deployment server sends the client the list of apps that it should have, along with those apps' current checksums
4. The client compares the app info list from the deployment server with its own app info to determine if there are any new or changed apps that it should download
5. If there are new or updated apps, the deployment client downloads them from the deployment server
6. Depending on the configuration for a given app, the client might restart itself so that the app changes will take effect

To use the deployment server, you will need to configure both the deployment server and the deployment clients. Let's start with setting up the deployment clients.

Configuring a deployment client

On the deployment client (universal or heavy forwarder), create a `deploymentclient.conf` file containing the entries as in the following example; this file resides in `$SPLUNK_HOME/etc/system/local`. You could also perform this function from the CLI on the deployment client the first time, if you want to verify the settings that get created. Run this command as the user `splunk`; you will be prompted to authenticate with the Splunk admin and password credentials you set when you installed the forwarder, unless you include the `-auth` argument:

```
$SPLUNK_HOME/bin/splunk set deploy-poll
<deploymentserverhostnameorIP>:<mgmtport> -auth username:password
Example:
$SPLUNK_HOME/bin/splunk set deploy-poll 172.31.17.204:8089 -auth
admin:changeme
$SPLUNK_HOME/bin/splunk restart
```

If you run into permission errors running this command, check to ensure that your Splunk forwarder is running as `splunk`:

```
ps -ef | grep splunk
```

If you're running Splunk as root, change to user root and stop Splunk (`$SPLUNK_HOME/bin/splunk stop`), then change to `splunk` user, and restart Splunk. If you have trouble with file permissions at this point, drop back to root and reset the ownership of all the files in `$SPLUNK_HOME` to `splunk` (`chown -R splunk:splunk *`) and try again.

Running the set deploy-poll CLI command will create a `deploymentclient.conf` file with the following minimal entries; the targetUri is the `<hostname or ip>:<mgmt_port>` of the deployment server:

```
[target-broker:deploymentServer]
targetUri = 172.31.17.204:8089
```

To this, you can optionally add a [deployment-client] stanza and set a client name, or change the polling cycle time from the default of 60 seconds, which is useful if you need to service hundreds or thousands of forwarders. The following example includes a few commented-out options for reference:

```
[deployment-client]
clientName = MDIWebServer
# phoneHomeIntervalInSecs = 300
# disabled = true

[target-broker:deploymentServer]
```

```
targetUri = 172.31.17.204:8089
```

You have to restart Splunk on the deployment client after any change. The forwarder will now start polling the deployment server every 60 seconds for updates. The apps that will be downloaded from the deployment server will be placed in the `$SPLUNK_HOME/etc/apps` folder on the forwarder; we'll cover the configuration of these apps in the next section.

Configuring the deployment server

You will need to configure two items on the deployment server to enable its functionality:

1. Create/configure the deployment apps that will contain the `.conf` files and other content to be deployed to the deployment clients in the `$SPLUNK_HOME/etc/deployment-apps` directory
2. Create/configure a `serverclass.conf` file in `$SPLUNK_HOME/etc/system/local` to map which apps get distributed to which deployment clients

A deployment app consists of any content you want to download to a set of deployment clients. This content can include the following:

- A Splunkbase app (such as the Splunk app for Linux and Unix or Splunk app for Microsoft Windows)
- A set of configurations (`inputs.conf`, `outputs.conf`, and so on)
- Other content, such as scripts and supporting files

The great thing about using the deployment server is that you can add or modify this content at any time, and the deployment clients will pick up and implement the changes upon their next phone-home cycle, making the management of any number of forwarders much easier.

Creating deployment apps

To get started and serve as an example, let's create a couple of apps on the deployment server to distribute the `outputs.conf` file to all of the forwarders, and then set up an `inputs.conf` for an example web server.

On the deployment server, create an `outputs.conf` file in a new folder called `forwarder_outputs` and a `/local` folder under that, in `$SPLUNK_HOME/etc/deployment-apps`. You'll note this is the same content we configured in the section on installing a universal forwarder – we're just going to manage and distribute this file from the deployment server instead of having to do it manually on each forwarder. This makes it possible to update this file easily if you add more indexers to your cluster down the road: `/opt/splunk/etc/deployment-apps/forwarder_outputs/local/outputs.conf`

```
[tcpout]
defaultGroup = indexer_cluster
useAck = true

[tcpout:indexer_cluster]
server = 172.31.28.223:9997,172.31.39.185:9997,172.31.13.169:9997
```

Next, we'll create another "app" to hold the `inputs.conf` file for monitoring our web server logs; this is again the same content from when we set up the forwarder configuration, minus the `[default]` stanza and host identifier: `/opt/splunk/etc/deployment-apps/webserver_inputs/local/inputs.conf`

```
[monitor:///var/log/httpd/access_log]
index = weblogs_90d_eidx
sourcetype = access_combined
ignoreOlderThan = 30d

[monitor:///var/log/httpd/error_log]
index = weblogs_90d_eidx
sourcetype = apache_error
ignoreOlderThan = 30d
```

Creating a serverclass.conf file

The interesting thing about a deployment server is that the act of creating a `serverclass.conf` file in the `$SPLUNK_HOME/etc/system/local` directory is all it takes to prepare any Splunk Enterprise instance to serve this function. While you can use almost any non-clustered Splunk member as a deployment server for very small (< 30 deployment clients) environments, you'll want to use a dedicated deployment server for higher counts because of the resource overhead involved. You can service up to ~300 clients with a single deployment server, depending upon its capabilities, and ~1,000 or more if you decrease the phone-home time on the deployment clients (forwarders).

The `serverclass.conf` file is where we map the apps we want to deploy to specific groups of deployment clients that should receive them. The best way of explaining how this is done is by example, so let's configure a very basic `serverclass.conf` to distribute the `forwarder_outputs` app to all of our deployment clients, and the `webserver_inputs` app to just our web server:

`/opt/splunk/etc/system/local/serverclass.conf`

```
[serverClass:all_forwarders]
whitelist.0 = *
restartSplunkd = true
[serverClass:all_forwarders:app:forwarder_outputs]


[serverClass:webserver]
whitelist.0 = 172.31.39.242
restartSplunkd = true
[serverClass:webserver:app:webserver_inputs]
```

In the preceding example, you'll note that there is a `serverClass` level, and a `serverClass:app` level, and that you can apply certain properties to either that will override any [global] properties that may be set (a `[global]` stanza is not shown in this example as it's not needed).

After creating this file, or any time you change it in the future, you need to tell the deployment server to read the file and implement the changes:

```
$SPLUNK_HOME/bin/splunk reload deploy-server
```

You'll be prompted for your username and password (unless you use the `-auth <username>:<passwd>` switch), then you should see **Reloading serverclass(es)**. After this, within a minute or so, you should be able to see that the `all_forwarders` and `webserver_inputs` apps folders have appeared in the `$SPLUNK_HOME/etc/apps` directory on your forwarder(s), and that you can run a search to confirm you're receiving current log events from the target server(s).

A slightly larger example of typical entries and format/usage in a `serverclass.conf` file can seen here:

```
[serverClass:app_name]
# Set the attribute to ipAddress, hostname, DNSname, or clientName
whitelist.0 = servername(0001|0002|0003).yourfqdn.com
whitelist.1 = servername(0027|0028|0030).yourfqdn.com
whitelist.2 = *.fflanda.com
```

```
blacklist.0=printer.fflanda.com
blacklist.1=scanner.fflanda.com

restartSplunkd = true
[serverClass:app_name:app:app_directory]

# There is also a machineTypesFilter which will match any machine types
# in a comma-delimited list - for example:
# linux-x86_64, windows-x64, linux-i686
```

Each `[serverClass:app_name]` stanza represents a `server class`; the `serverclass.conf` file can contain multiple server classes, and hosts can belong to one or more server classes as needed. The **whitelist/blacklist** entries specify the hosts to include/exclude in specific classes. `restartSplunkd` causes the deployment client to restart `splunkd` after an update. The `[serverClass:app_name:app:app_directory]` stanza specifies the app directories located in `$SPLUNK_HOME/etc/deployment-apps` on the deployment server that are to be sent to the deployment clients. There can be multiple stanzas in each server class, one for each app to include.

You'll recall that in the first `serverclass.conf` example, only the web server belonged to the `webserver` class, but all forwarders belong to the `all_fowarders` class so that they can pick up the `outputs.conf` file.

An important point to remember about using the deployment server to manage the apps and configurations across your deployment clients is that if you delete an app or configuration file from the `.../deployment-apps` folder on your deployment server, that app or configuration file will be deleted from all of the deployment clients that previously had them installed. Similarly, if you delete or comment out the entries for a specific server class in `serverclass.conf`, the deployment clients will delete the app(s) that were referenced in that server class, regardless of whether the app and its contents still reside in the `.../deployment-apps` folder.

Obviously, you'll want to manage the content within the `deployment-apps` folder carefully. The good thing is that if you accidentally mis-alter your `serverclass.conf` file or delete an app from `.../deployment-apps`, you can always repair/replace the error and the deployment clients will pick up and reinstall the missing app upon their next phone home.

Using forwarder management in Splunk web

You can use the Forwarder Management app in Splunk Web to create and manage your server classes, assign apps to specific clients, and so on by clicking **Settings** | **Forwarder management** on the deployment server. There are three tabs: **Apps**, **Server Classes**, and **Clients**. The Forwarder Management app is useful for verifying that deployment clients are phoning home and picking up their apps, and if you have your apps built, it can be a quick and easy way to add additional server classes and configure them now that you understand how these are manifested in the `serverclass.conf` file. Also, if you create a new server class in the Forwarder Management app, the changes are applied to the deployment server and rolled out after you click **Save** without having to do a `reload deploy-server` CLI command:

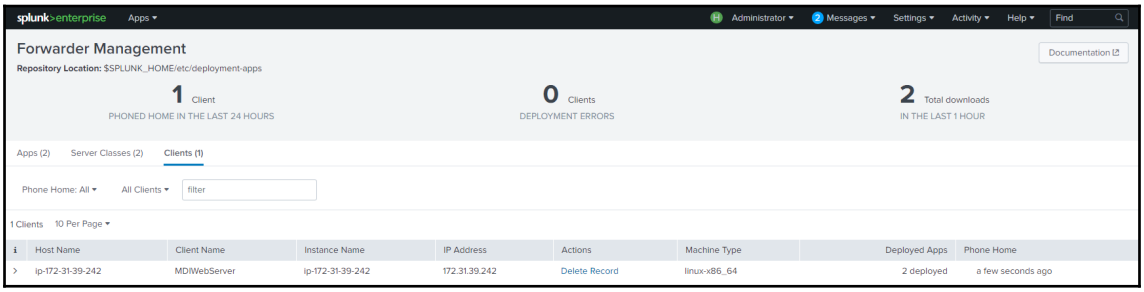


Fig 4.6: Forwarder Management app on a deployment server

You can read more about configuring and using the deployment server in the Splunk Wiki and docs:

- `https://wiki.splunk.com/Deploy:DeploymentServer`
- `http://docs.splunk.com/Documentation/Splunk/latest/Updating/Aboutdeploymentserver`

Managing Splunk Indexes

Indexes are where the data that is sent to Splunk is stored. Each index is a directory, and as we discussed in previous sections, each directory contains a subdirectory for host/warm buckets (in `/db`), cold buckets (in `/colddb`), and a `datamodel_summary` , and `thaweddb` directory, which may be empty if not used for datamodels and thawed buckets, respectively. The folder structure for Splunk's `_internal` index looks like this:

```
/opt/splunk/var/lib/splunk/_internaldb
   /colddb
   /datamodel_summary
   /db
   /thaweddb
```

Index buckets are the files and directories inside each of the above; for example, the `/db` directory will contain directories of both hot buckets (open and being written to, usually named `hot_xxx`), and warm buckets (closed for writing, but quickly searchable, usually named `db_xxx` or `rb_xxx`). The `db_xxx` buckets are those that were created on this particular indexer; the `rb_xxx` buckets are replicated copies sent to this indexer from other indexers to meet the replication factor.

You will want to create Splunk indexes that are used to store specific sets of data based on your applications, line of business, or numerous other criteria that fit the needs of your business. Using a number of specific indexes, instead of one or just a few very large indexes, offers several advantages:

- You can control who has access to indexes that contain sensitive data on a per-index basis with Splunk user role settings
- You can create indexes with differing data retention periods, thereby optimizing retention and disk storage factors
- Using smaller indexes provides improved search performance – you can specify the needed index(es) in your search strings
- You can control where the data is stored on a per-index basis, allowing cheaper storage of older event data

In this section, we'll cover the options used in the configuration file that creates and manages indexes – `indexes.conf`. This file is usually located either in `/opt/splunk/etc/system/local` or, more commonly, within an app folder such as `/opt/splunk/etc/apps/devtest/local`. Obviously, there can be numerous `indexes.conf` files scattered across numerous apps; Splunk will combine these upon startup and use the collective settings to manage indexes. Many apps you might install from Splunkbase may specify the indexes and settings those apps use with `indexes.conf` files located inside their respective app folders; for the indexes you create for your business apps, you can either create apps for each use case and configure `indexes.conf` files for each app, or maintain a centralized file for all your indexes either within `$SPLUNK_HOME/etc/system/local`, or within an app you create such as `$SPLUNK_HOME/etc/apps /splunk_indexes/local`. The specific approach you take will depend on your preference and whether you apply an administration automation scheme to your Splunk deployment.

Creating an index

The most common settings used in an `indexes.conf` file are listed as follows for an example of a normal *events* index called `weblogs_90d_eidx`; we'll touch on metrics indexes shortly:

```
[weblogs_90d_eidx]
repFactor=auto
homePath = $SPLUNK_DB/weblogs_90d_eidx/db
coldPath = $SPLUNK_DB/weblogs_90d_eidx/colddb
thawedPath = $SPLUNK_DB/weblogs_90d_eidx/thaweddb
# Set the Maximum Index Size in MB
# 51200 MB = 50 GB
maxTotalDataSizeMB = 51200
# Set Index Retention to 90 Days
frozenTimePeriodInSecs = 7776000
```

The significant settings include the following:

- `repFactor = auto`: This tells Splunk to replicate this index across the indexing cluster – important!
- `homePath/coldPath/thawedPath`: The disk locations where this indexes' buckets get stored in their various states

- `maxTotalDataSizeMB = 51200`: The maximum size this index can become before the oldest events are discarded
- `frozenTimePeriodInSecs = 7776000`: Events older than this period are either rolled to the frozen storage location, or if this isn't specified (by using a `coldToFrozenDir` and/or `coldToFrozenScript` setting (see the docs)), the data is discarded.

Basically, you're telling Splunk where to store the index buckets, and when to get rid of old data, either because the index becomes too big, or preferably, when the data *ages out* and is discarded because it exceeds the `frozenTimePeriodInSecs` time period. The `maxTotalDataSizeMB` setting should be used more as a fail-safe across your indexes to avoid exceeding your disk storage space in case the number or size of events is excessive. Note the use of a `#` to include comments in any `.conf` file.

In the preceding example, the index storage paths for homePath (hot/warm buckets), coldPath, and thawedPath uses the `$SPLUNK_DB` environment variable that, by default, is typically set to `/opt/splunk/var/lib/splunk` for Linux, or `C:\Program Files\Splunk\opt\splunk\var\lib\splunk` on Windows machines. If you are storing your index data in a different partition or drive than the one that contains your OS and Splunk files, you would explicitly specify different homePath and coldPath locations.

You can peruse all the options for configuring `indexes.conf` files in the Splunk docs:

```
https://docs.splunk.com/Documentation/Splunk/latest/Admin/Indexesconf#PER_
INDEX_OPTIONS
http://docs.splunk.com/Documentation/Splunk/latest/Indexer/
Configureindexstorage
```

Although you can view your indexes in Splunk Web by clicking **Settings** | **Indexes**, and create and configure a new index (on a stand-alone indexer only!!) by clicking **New Index**, if you try this approach, you'll note that *you cannot configure a retention period from the index options page in Splunk Web*. If you don't want to use the default 6-year retention period, you'll have to set the retention value by adding a `frozenTimePeriodInSecs` entry and value to the `indexes.conf` file—so you may has well do all of your configuration settings with manual edits and save the extra step.

If you are administering an indexing cluster, you must not create or configure indexes from Splunk Web on an individual indexer; you have to create or modify the indexes by configuring an `indexes.conf` file on the cluster master, and distribute the index configurations to the search peers (indexers) from there. We will cover using a cluster master for this purpose in a later section of this chapter.

Deleting index data

If you want to remove an index and delete the data that is in it, you can remove the index's stanza and entries, and then delete that index's directory from your `$SPLUNK_DB` location. You can also delete specific events from an index without deleting the index by using Splunk's `delete` command, which involves creating a search to identify the events you want to get rid of, and piping those search results to the delete command. Note that this action requires that your role (even if you're an admin) include the `can_delete` capability, and that **this activity is dangerous**, for obvious reasons. I highly recommend you carefully read the Splunk documentation on this subject before deleting indexes or specific events within an index:

`http://docs.splunk.com/Documentation/Splunk/latest/Indexer/RemovedatafromSplunk`

Summary indexes

A summary index is a designated Splunk index that stores the results of an ad hoc search piped to a `collect` command, or a scheduled report when you enable summary indexing for the report. Summary indexing lets you run fast searches over large datasets by spreading out the cost of a computationally expensive report over time. To achieve this, the search that populates the summary index runs on a frequent, recurring basis and extracts the specific data that you require. You can then run fast and efficient searches against this smaller subset of data in the summary index versus running the search across all the events in the source index. Because the footprint of a summary index is typically much smaller, you can keep the summarized data for much longer periods of time without consuming excessive disk space, and pull data out of the summary index for historical reporting. Summary indexes are also useful for storing historical time series data for statistical analysis, anomaly detection, and related machine learning efforts.

A summary index is created just like any other Splunk index–by creating entries in an `indexes.conf` file with settings to configure an adequate size and retention period for the data to be stored. Note that sending data from search results to a summary index does *not* incur any additional ingestion charges to your Splunk license—so feel free to leverage this powerful capability where needed.

Metrics indexes

The indexes we have discussed so far are created to contain standard *events* that can accommodate a wide variety of log or device data formats and include any number of fields. With the release of Splunk Enterprise 7.x, Splunk introduced "metrics" indexes, which, as the name suggests, are used to store numerical values that pertain to any activity or process measured over an interval of time—metrics from applications, servers, network devices, IoT sensors, and so on. Splunk natively supports metrics from some well-known industry metrics-gathering tools, as well as the use of the HTTP Event Collector for collecting metrics:

- **collectd agent**: A Unix-based daemon, using the write_HTTP plugin
- **StatsD line protocol:** This is used by a wide range of client libraries and open source tools

You can also feed metrics data into Splunk from comma-separated value (CSV) files.

Metrics indexes store a limited number of field types, the most prevalent/necessary of which include the following:

- `metric_timestamp`: A timestamp of the metric in UNIX time notation (2018-08-31 18:23:32.000)
- `metric_name`: The metric being recorded, in dot-notation format (`os.cpu.user`)
- `_value`: The 64-bit floating point ("Double" type) numeric value of the metric (72.12345)
- `<some other parameter>`: Optional, additional field(s) that are stored as a *dimension* (Ex: `device_type`)

If you want to feed metric data into Splunk using a CSV file, the preceding fields are the minimum fields you will need to provide (`metric_timestamp`, `metric_name`, and `_value`) in the first (header) line of the file. Metrics indexes also record the standard source, sourcetype, host, and index metadata fields, and you can optionally leverage "dimensions" fields for storing additional information for organizing and filtering the metrics data. Metrics indexes are configured in `indexes.conf` files just as events indexes are; the only difference is the inclusion of a "datatype = metric" entry:

```
[sensors_180d_midx]
repFactor = auto
homePath = $SPLUNK_DB/sensors_180d_midx/db
coldPath = $SPLUNK_DB/sensors_180d_midx/colddb
thawedPath = $SPLUNK_DB/sensors_180d_midx/thaweddb
datatype = metric
# max size = 50 GB
maxTotalDataSizeMB = 512000
# retention period is 180 days
frozenTimePeriodInSecs = 15552000
```

You cannot write search strings to view metrics events in Splunk Web as you normally would for standard events; Splunk provides the `mstats` reporting command for extracting and visualizing metrics data, using various metadata and/or dimensions for filtering. An example of using the `mstats` command to visualize sensor data is as follows:

```
| mstats count(_value) WHERE metric_name=* AND index="sensors_180d_midx"
AND source="sensor_test_data.csv"
```

You will find more details about configuring, using, and reporting from metrics indexes in the Splunk docs:

`https://docs.splunk.com/Documentation/Splunk/latest/Metrics/Overview`

Splunk sourcetypes

Splunk uses specific configuration files for telling the indexers how to parse incoming data to properly extract the timestamps and specific event fields, and how to identify when one event ends and the next one starts (linebreaks); this is done with a `props.conf` file. In some cases, you'll need to tell Splunk how to transform (modify or add) data from existing fields into new fields and/or modify or mask those fields–this is done with a `transforms.conf` file. The `props.conf` file contains stanzas that define a sourcetype and how it behaves; entries in the `props.conf` stanzas can also reference stanzas in a `transforms.conf` file to perform transformation operations.

A sourcetype is one of the default metadata fields that Splunk software assigns to all incoming data. It tells Splunk what kind of data is being ingested so that it can identify the various fields in the data properly during indexing.

Splunk software comes with a large number of predefined source types. When consuming industry standard data, Splunk software will usually select the correct sourcetype automatically. If Splunk doesn't automatically identify the format of your data, you might need to select a predefined sourcetype or create a new custom sourcetype manually in a `props.conf` file with entries that identify the location of line breaks, timestamps, and perhaps some key fields. Finally, if your data source contains heterogeneous (diverse in character or content) data, you might need to assign the sourcetype on a per-event (rather than a per-source) basis.

You can use the sourcetype field as a filter to search event data once the data has been indexed; the sourcetype is a key way to categorize your data, so you'll end up using it a lot.

An example of some of the common source types that Splunk software automatically recognizes includes the following:

- `access_combined`: NCSA format HTTP web server logs
- `apache_error`: Standard Apache web server error logs
- `iis`: Windows Internet Information Services logs
- `cisco_syslog`: The standard syslog produced by Cisco network devices (including PIX firewalls, routers, and ACS), usually via remote syslog to a central log host, which as we discussed previously you can forward to Splunk using a universal forwarder that's installed on that log host

You can view these built-in sourcetypes from Splunk Web by clicking **Settings** | **Source types** (there are multiple pages–click through them with the prev # next buttons, and/or select the **100 per page** option from the drop-down). You can investigate the configuration details of each sourcetype by clicking **Edit** | **Advanced**, and all of the settings for each of these predefined source types are stored in:

```
/opt/splunk/etc/system/default/props.conf
```

If you create or modify a sourcetype using Splunk web, the resultant props.conf file will reside in the `/local` folder of the current app context (search, loader, and so on) or the app you specify during the edit operation. If you manually create/edit custom `props.conf` and `transforms.conf` files, you will store these inside the `/local` folder of an existing or custom `Splunk app` folder:

```
/opt/splunk/etc/<my_app>/local/props.conf
```

Creating custom source types

If you need to create a *custom* sourcetype for your unusual log type or other data source, you can clone an existing sourcetype and modify it for your purposes if the log you're wanting to ingest is similar to an existing known format. Otherwise, you will need to create a `props.conf` file that gets stored in your `$SPLUNK_HOME/etc/apps/<app folder>/local` directory, which contains the needed entries to properly parse the incoming data.

You can peruse the examples in the `.../default/props.conf` file, as well as all the options for `props.conf` and `transforms.conf` files in the following Splunk docs:

- `http://docs.splunk.com/Documentation/Splunk/latest/Data/Whysourcetypesmatter`
- `https://docs.splunk.com/Documentation/Splunk/latest/Admin/Propsconf` `https://docs.splunk.com/Documentation/Splunk/latest/Admin/Transformsconf`

The most common entries you may need to use in a `props.conf` file to create a custom sourcetype will have to do with specifying whether these are single or multiline events, identifying where in the event data a linebreak should occur, the proper timestamp location and format (especially in cases where an event contains more than one timestamp), and perhaps an indicator of the content format (JSON, XML, and so on). A sampling of some of the most common parameters you will use in a `props.conf` file is listed below as follows, understanding that you'll not use all of these in most configurations, or at least not all together. The entry in the stanza defines the sourcetype name; the parameters that follow are parsing parameters that are assigned to that sourcetype. The `//` comments are not allowed in a real file:

```
[my_custom_sourcetype]
SHOULD_LINEMERGE = false        // combines several lines into a single
line event. Default = true
LINE_BREAKER = ([\r\n]+)        // defines how to identify the start of the
next event (using RegEx)
BREAK_ONLY_BEFORE = \{\"name    // another way to identify the start of the
next event (using RegEx)
MAX_TIMESTAMP_LOOKAHEAD = 40    // how far into an event to look for a
timestamp (default = 128)
TIME_PREFIX = \"time\":\"       // RegEx of text that preceeds the start of
a timestamp
TIME_FORMAT=%Y/%m/%d %H:%M:%S   // defines how to parse an unusual
timestamp format
TRUNCATE = 5000                 // tells Splunk not to accept more than
<integer> bytes per event (default = 10000)(0 = don't truncate)
```

```
MAX_EVENTS = 1024               // specifies the max number of input lines
to add to a multi-line event (default = 256)
description = my node.js app log // <string> used to describe the
sourcetype. Doesn't affect parsing
TZ = US/Eastern                 // specifies the time zone to apply if not
specified in the timestamp
ANNOTATE_PUNCT = false          // turns off 'punctuation' if you don't
plan to use it in searches
KV_MODE = none                  // if you don't have KV pairs or other
structured format, disable it
SEDCMD-remove_header = s/(\{\s*\"records\":\s*\[)//g    //  RegEx replace:
s/<pattern_to_match>/<replace>/g

[another_sourcetype]
SHOULD_LINEMERGE = false
MAX_TIMESTAMP_LOOKAHEAD = 40
LINE_BREAKER = ([\r\n]+)
TIME_FORMAT = %Y-%m-%dT%H:%M:%S.%3N%Z    // this timestamp includes a
timezone specifier
```

The date and time format variables that Splunk recognizes are defined here:

`https://docs.splunk.com/Documentation/Splunk/latest/SearchReference/Commontimeformatvariables`

Sometimes, you may need to extract or transform event data fields; this is accomplished using a combination of the `props.conf` and `transforms.props` files, as follows:

```
# props.conf
[my_custom_sourcetype]
SHOULD_LINEMERGE = false
...

[sourcetype::my_custom_sourcetype]
TRANSFORMS-appServerHostname = appserver_hostname
```

In the preceding example, following the custom sourcetype definition and its entries is another stanza that references your custom sourcetype (`[sourcetype::my_custom_sourcetype]`) with a `TRANSFORM` pointer (`TRANSFORMS-appServerHostname`) that maps to a transform operation (`appserver_hostname`).

The transform operation itself is specified in a stanza (`[appserver_hostname]`) in a `transforms.conf` file; this stanza reflects/matches the `TRANSFORMS` pointer value in the `props.conf` file, and contains entries that will extract a value from a `hostname` field in each event and assign that hostname to the `host` metadata field that Splunk stores with every event as it is indexed:

```
# transforms.conf
[appserver_hostname]
REGEX = MachineName="(?<hostname>.*?)"
FORMAT = hostname::$1
DEST_KEY = MetaData:Host
```

The `transforms.prop` file should reside in a `/local` folder, along with its companion `props.conf` file. The `props.conf` and `transforms.conf` files, along with any `indexes.conf` file(s), must be distributed to the indexers, the process for which is discussed in the next section.

Using the cluster master

Now that you have defined your indexes in `indexes.conf` file(s), and where needed, defined your custom source types in `props.conf` and perhaps a `transforms.conf` file, you need to get these files distributed to all the indexers in the cluster so that they can use them to parse the incoming data properly from forwarders and/or other data sources that reference those indexes and sourcetypes in their `inputs.conf` files, and store the parsed data in the appropriate index. Each indexer in the cluster must be working with identical configuration files – this configuration file distribution is accomplished by using the cluster master.

In similar fashion as is done to distribute apps from a deployment server, you distribute apps containing `indexes.conf` and any other configuration files by creating apps directories in a `$SPLUNK_HOME/etc/master-apps` folder on the cluster master. This folder already has a `_cluster` directory in it, which contains a `/default/indexes.conf` file that should not be altered. You will create your new app folders as per the following illustration of a `webserver` and `sensors` app; each of these have `indexes.conf` files containing the example entries you saw in the preceding sections on creating indexes:

```
$SPLUNK_HOME/etc/master-apps/
                              _cluster/
                                     default/
                                      local/
                              <webserver>/
                                      local/
```

```
                                indexes.conf
                                props.conf
                                transforms.conf
                    <sensors>/
                            local/
                                indexes.conf
                     ...
```

Note that you cannot distribute individual files that are not within a subdirectory under `.../master-apps`; if you need to distribute a lone file to all the indexers, you can place it in the `.../master-apps/_cluster/local` folder.

Distributing the configuration bundle

The collection of directories in `$SPLUNK_HOME/etc/master-apps` is called a `configuration bundle`; when the cluster master distributes this bundle to the indexing peers, it overwrites the entire contents of any configuration bundle previously distributed. On the indexing peers, the configuration bundle contents received from the cluster master is stored in the `$SPLUNK_HOME/etc/`**`slave-apps`** folder.

Here is the recommended process for distributing your configuration bundle from the cluster master using the CLI:

```
1. Validate the cluster-bundle
$SPLUNK_HOME/bin/splunk validate cluster-bundle -auth username:password
2. Inspect the results to ensure a valid configuration bundle
$SPLUNK_HOME/bin/splunk show cluster-bundle-status
3. Apply the new configuration bundle to all the indexers
$SPLUNK_HOME/bin/splunk apply cluster-bundle
4. Monitor the indexer restart process (if one was needed) and validate all
indexers are up
$SPLUNK_HOME/bin/splunk show cluster-bundle-status
```

The results of a `show cluster-bundle-status` command before running the `apply cluster-bundle` command are depicted in *Fig 4.7* as follows. You can see that the `last_validated_bundle` checksum is different than the `latest_bundle` checksum, and most importantly, `last_validation_succeeded=1` is present.

You will also know if a rolling restart of the indexers will be required:

```
-bash-4.2$ ./splunk validate cluster-bundle
Validating new bundle. Please run 'splunk show cluster-bundle-status' to check the status of the bundle validation.
Created new bundle with checksum=A609FFBB016C56E982F5D5E685F79ED3
-bash-4.2$
-bash-4.2$ ./splunk show cluster-bundle-status

master
        cluster_status=None
        active_bundle
               checksum=AD63783A794BE1C6A3D27CFC6EFC639E
               timestamp=1531075572 (in localtime=Sun Jul  8 18:46:12 2018)
        latest_bundle
               checksum=AD63783A794BE1C6A3D27CFC6EFC639E
               timestamp=1531075572 (in localtime=Sun Jul  8 18:46:12 2018)
        last_validated_bundle
               checksum=A609FFBB016C56E982F5D5E685F79ED3
               last validation succeeded=1
               timestamp=1534628241 (in localtime=Sat Aug 18 21:37:21 2018)
        last_check_restart_bundle
               last_check_restart_result=restart not required
               checksum=
               timestamp=0 (in localtime=Thu Jan  1 00:00:00 1970)

 ip-172-31-13-169.ec2.internal   1FA5C915-5ABF-419A-8D26-A4B00F0E87D4    default
        active_bundle=AD63783A794BE1C6A3D27CFC6EFC639E
        latest_bundle=AD63783A794BE1C6A3D27CFC6EFC639E
        last_validated_bundle=A609FFBB016C56E982F5D5E685F79ED3
        last_bundle_validation_status=success
        restart_required_apply_bundle=0
        status=Up

 ip-172-31-28-223.ec2.internal   60B63B6B-1C66-4592-84B6-1DA13B452F6F    default
        active_bundle=AD63783A794BE1C6A3D27CFC6EFC639E
        latest_bundle=AD63783A794BE1C6A3D27CFC6EFC639E
        last_validated_bundle=A609FFBB016C56E982F5D5E685F79ED3
        last_bundle_validation_status=success
        restart_required_apply_bundle=0
        status=Up

 ip-172-31-39-185.ec2.internal   B38F17CE-F3C9-449C-A729-850E1AC9B179    default
        active_bundle=AD63783A794BE1C6A3D27CFC6EFC639E
        latest_bundle=AD63783A794BE1C6A3D27CFC6EFC639E
        last_validated_bundle=A609FFBB016C56E982F5D5E685F79ED3
        last_bundle_validation_status=success
        restart_required_apply_bundle=0
        status=Up
```

Fig 4.7: cluster-bundle-status results

You should *always* run `validate cluster-bundle` and then the `show cluster-bundle-status` command and inspect the results for the presence of `last_validation_succeeded=1` *before* running the apply command to make sure there are no typos or other errors in the files in the new configuration bundle. Not doing so can result in distributing a corrupt bundle that may cause all of your indexers to go down and stay that way until you manually log in to each one, fix the error, and restart the indexer. I know this from painful experience.

If you still have Splunk Web enabled on your indexers, you can log in to one of them and select **Settings** | **Index** to view the indexes and their configured size. Remember **not** to edit index configurations from Splunk Web on an indexer!

You can also control the distribution of the configuration bundle from Splunk Web by selecting (on the cluster master only) **Settings** | **Indexer clustering**. From this dashboard, you can view a list and the status of peers (indexers), indexes, and search heads. Clicking **Edit** | **Configuration Bundle Actions** will allow you to validate the configuration bundle, push it to the search peers (indexing cluster), or roll back the last change if the configuration has a problem (but heed my preceding warning – if the indexers are all down, you can't roll back anything). You can also perform a rollback from the CLI by using **`./splunk rollback cluster-bundle`** if needed:

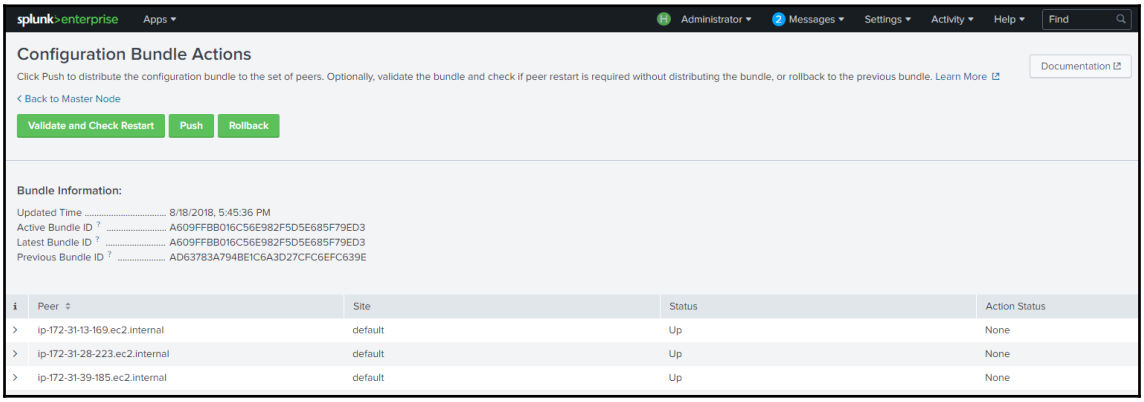


Fig 4.8: Configuration Bundle Actions dashboard

You can perform searches from the cluster master (click **Apps** | **Search & Reporting** to the top-left), although it is not and must not be configured to be part of a search head cluster. The cluster master is also a good location to configure the Monitoring Console, which is a good tool for monitoring the health of your indexing cluster; we'll cover the MC in `Chapter 10`, *Advanced Splunk*.

Summary

The topics we've covered so far include activities you'll perform on a fairly regular basis in your Splunk environment. We covered installing and configuring universal and heavy forwarders, and how to configure Splunk to input data from forwarders and other sources, including how to use the increasingly popular HTTP Event Collector. We then reviewed how to manage forwarders with the deployment server, got an introduction to Splunk apps, and learned how to create indexes, custom sourcetypes, and how to deploy those configurations to the indexing peers using the cluster master.

Now that we've got data flowing into our indexers, let's learn about a few other admin tasks in the next chapter!

5
Administering Splunk Apps and Users

—In this chapter, we will round out our administrative duties by learning how to distribute Splunk apps and set up our users and their roles so that they can access the data in Splunk, and have a place (a Splunk app) to save and use the reports, dashboards, and alerts they'll build using this data. We'll also spend a little time discussing how to best manage and support your Splunk environment in terms of financial and administrative resources so that it keeps running in top shape.

The topics covered in this chapter include:

- How to distribute Splunk apps from a deployer
- Configuring users and roles to grant access to Splunk functionality
- Best practices for managing data in Splunk
- Supporting your Splunk deployment

Let's go!

Using the deployer

The deployer is used to distribute apps and user files to search head-cluster members. This occurs when you execute a command to apply a new or updated configuration bundle that you have prepared; it also occurs when a search-cluster member joins or rejoins the cluster—it contacts the deployer to see whether there are any updates it needs to download—so that all search-cluster members always have identical configurations.

The deployer function in a small Splunk installation can reside on another supporting member, such as a cluster master or license master. In larger deployments, it should be a dedicated instance, mostly because the other dedicated instances will be pretty busy performing their respective functions.

The configuration bundle created and distributed to search cluster members by the deployer is not the same as the configuration bundle that gets distributed from the cluster master to indexing peers; the cluster-master configuration bundle distributes apps that `contain indexes.conf`, `props.conf`, `transforms.conf`, and perhaps some other files that the indexers use to parse and store data in indexes.

Conversely, the configuration bundle distributed by the deployer includes app directories destined to reside in the `$SPLUNK_HOME/etc/apps` folder on the search heads, which contain configuration and other files that implement reports, dashboards, and the saved searches that drive them, navigation and view files. It can also contain files that will reside in the `$SPLUNK_HOME/etc/users` folder on the search heads— `local.meta` and other files that define roles and preferences for Splunk users.

To use the deployer effectively, it's important to understand what the distribution process actually does and how it interacts with a working search head cluster, so let's review some basics.

In each member of a search-head cluster, there are various app directories that reside inside the `$SPLUNK_HOME/etc/apps` folder, and within each of those apps' folders are various other folders, such as appserver, bin, default, and metadata. Within each app directory is also a `/local` directory—this is where the `.conf` files containing the details of *user-created* changes are stored. If a user creates a search and saves it as a report, for example, the report definitions are saved in a `savedsearches.conf` file in the `/local` directory *for the app context* the report was created in. The app context is the app you are working in when the report is created. If you select Search & Reporting when you log in, and then create and save a report, the `savedsearches.conf` file that gets created or updated will reside in the `$SPLUNK_HOME/etc/apps/search/local` directory. If you had selected another app (say, DevTest) before creating and saving the report, the `savedsearches.conf` file would get saved in the `.../etc/apps/devtest/local` folder.

In a search-head cluster, a user may be working from any one of the cluster members. If they create a new report, the `savedsearches.conf` file containing that report definition is saved in the `<app>/local` folder *on that search head*.

Every search-head cluster has a Captain (more on that in a bit); it's the Captain's job to detect that the `savedsearches.conf` file has been created or updated, and to distribute the new/updated copy to the other search-cluster members (placing it in the `/local` folder under the applicable app directory on each search head) so that users that happen to be logged into a different search head can see and run that report—this happens fairly quickly. With that info fresh in your mind, let's discuss the deployer now.

To distribute a new or updated app to the search head cluster, you place a copy of that app directory, containing all its subdirectories and files (you cannot distribute a `.tar.gz` file—the files must be un-archived/uncompressed), in the `$SPLUNK_HOME/etc/shcluster/apps` folder on the deployer. Here is the directory structure you'll work with on the deployer:

```
$SPLUNK_HOME/etc/shcluster
                        /apps
                            /<devtest>
                            /<ready_player_one>
                            ...
                        /users
                            /wade
                            /samantha
                            ...
```

Most of the time, you'll just use the deployer to distribute new or updated apps, but if you need to distribute a specific user settings or you're migrating one or more apps that were previously residing on a standalone search head or a search pool (a set of search heads that weren't configured as a cluster), you might also copy the directories from the `$SPLUNK_HOME/etc/users` folder from the original search head(s) to the deployer – these will go in the `$SPLUNK_HOME/etc/shcluster/users` folder on the deployer.

When you are ready, run the following command from the deployer to distribute the new/updated apps and users info to the search head cluster; the `-target` points to any one of the search-head cluster members:

```
$SPLUNK_HOME/bin/splunk apply shcluster-bundle -target
<URI>:<management_port> - auth <username>:<password>
For example:
./splunk apply shcluster-bundle -target https://172.31.46.250:8089 -auth
admin:Splunk1t2me

You will see a message:
Warning: Depending on the configuration changes being pushed, this command
might initiate a rolling restart of the cluster members. Please refer to
the documentation for the details. Do you wish to continue? [y/n]: y

When the push has completed you will get a message:
Bundle has been pushed successfully to all the cluster members.
```

If you want to determine which cluster member is the Captain and/or check the status of the rolling-restart that may occur after a configuration bundle deployment, you can run this command on one of the cluster members (not on the deployer):

```
./splunk show shcluster-status –auth <username>:<password>
```

When distributing the configuration bundle to the search cluster members, the deployer treats the folders in `/shcluster/apps and /shcluster/users` differently. If there are new or updated app folders in the `/shcluster/apps` directory, the deployer will create a tarball for *each* app directory and send those tarballs to each member of the search-head cluster. Each member will then decompress and update the appropriate app(s) in their `.../etc/apps` directory.

If there were new or updated files in the `.../shcluster/users` folder on the deployer, the deployer will create a *single* tarball of *all* of the user folders to the search-cluster Captain; the Captain takes care of distributing the updates to the `/users` folder to all the other cluster members.

There are a few additional things you should be aware of regarding the use of the deployer:

- The deployer only distributes new apps or apps that have changed since the last push.
- The deployer won't distribute standalone files that are outside of the `/apps` or `/users` folders within `.../shcluster`; you must place all files within a subdirectory.
- Any change to any file in the `.../shcluster/users` directory will result in a tarball of the entire `/users` folder being distributed.
- If you delete an app directory from `.../shcluster/apps` and do a push, *that app will be deleted on all the cluster members, so be careful*!
- If you attempt to push a large tarball (> 200 MB), the deployment may fail due to timeouts. You can delete some (or the largest) app and do a push, then add the other/larger app(s) back into the `.../shcluster/apps` folder and push again, since the deployer won't try to push any apps that have already been deployed and haven't changed since the last push.

- If you tried to deploy an empty `.../shcluster/apps or /users` folder, the deployer would effectively delete all the existing apps or user folders on the cluster members – for this reason, Splunk will abort and print an error message unless you override this behavior with a `-force true` switch in the apply `shcluster-bundle` command.
- You can prevent the search cluster from automatically doing a rolling restart by using a variation of the apply command: `splunk apply shcluster-bundle -action stage && -action apply`, then run an `apply shcluster-bundle -action send` command. I'm not sure why you would do this, but there it is.
- You can then invoke the `./splunk rolling-restart shcluster-members` command from the Captain. A rolling restart of your search-head cluster is also occasionally helpful if you need to ensure all the members are consistent.
- Finally, the deployer stages the tarballs it deploys to the cluster members in its `$SPLUNK_HOME/var/run/splunk/deploy` directory if you want to inspect the contents for troubleshooting purposes or just out of curiosity.

There is one more facet of using the deployer that you should consider when updating apps across the search-head cluster: if the app that you copy to `.../shcluster/apps` contains a `/local` folder with configuration files, the settings in those configuration files will be moved to the `.../etc/apps/default` folder before they are tarballed and distributed to the cluster members. If there are existing `.conf` files and entries in the `.conf` files in the `/default` folder, the settings from the `.conf` files that were in the `/local` folder will be merged using standard Splunk precedence (any settings to stanzas in a `/local .conf` file will take precedence over the same setting in a `.conf` file in the `/default` folder). In this way, any settings in an app's `/local` folder on the deployer are not lost, but you avoid overwriting any user-created configurations stored in the `.../apps/<app>/local` folder on the cluster members, which would otherwise result in those settings being lost and upsetting a lot of users.

In a similar fashion, stanza settings in any file in the `/local` folder in the `.../shcluster/users` directories are merged with any existing settings of the same type on the cluster members using Splunk precedence.

Deploying new or updated apps

If you want to create a new Splunk app, update an existing app, or download and install an app from Splunkbase and distribute that app to all of your search head cluster members, you will do the initial app install or update from the deployer using the following steps (from Splunk Web). We'll cover creating and installing Splunk apps in much more detail in `Chapter 9`, *Splunk Applications*, so I'll just provide generic steps here:

1. Create the app: **Apps** | **Manage Apps** | **Create App** and/or install the app from Splunkbase: **Apps** | **Find More Apps** and/or update the files in an existing app directory
2. Copy the new/updated app directory(s) from `.../etc/apps` to `.../etc/shcluster/apps`
3. Deploy the new configuration bundle using the `splunk apply cluster-bundle` command

You can read more about using the deployer to distribute apps to search head-cluster members in the Splunk doc on this subject: `http://docs.splunk.com/Documentation/Splunk/latest/DistSearch/PropagateSHCconfigurationchanges`.

Configuring users and roles

Now that you have a working Splunk Enterprise environment, you will need to configure users to give them access to Splunk search heads, and assign roles to each user to control what levels of access and control they have in the environment. For small installations, you may choose to configure users with only Splunk own authentication controls; for larger installations, you'll likely use **Lightweight Directory Access Protocol** (**LDAP**), SAML (Security Assertion Markup Language), or one of the other available authentication methods.

Splunk authentication

Splunk authentication is on by default, and remains on even if you select one of the other authentication methods. This means that even if you have a LDAP or SAML username that can log into Splunk, you can also still log in using admin or any other Splunk-defined username.

You create and configure users within Splunk itself by clicking **Settings** | **Access controls** on one of the search heads. Clicking **Users** and then **New User** allows you to create a new user and assign one of the defined Splunk roles:

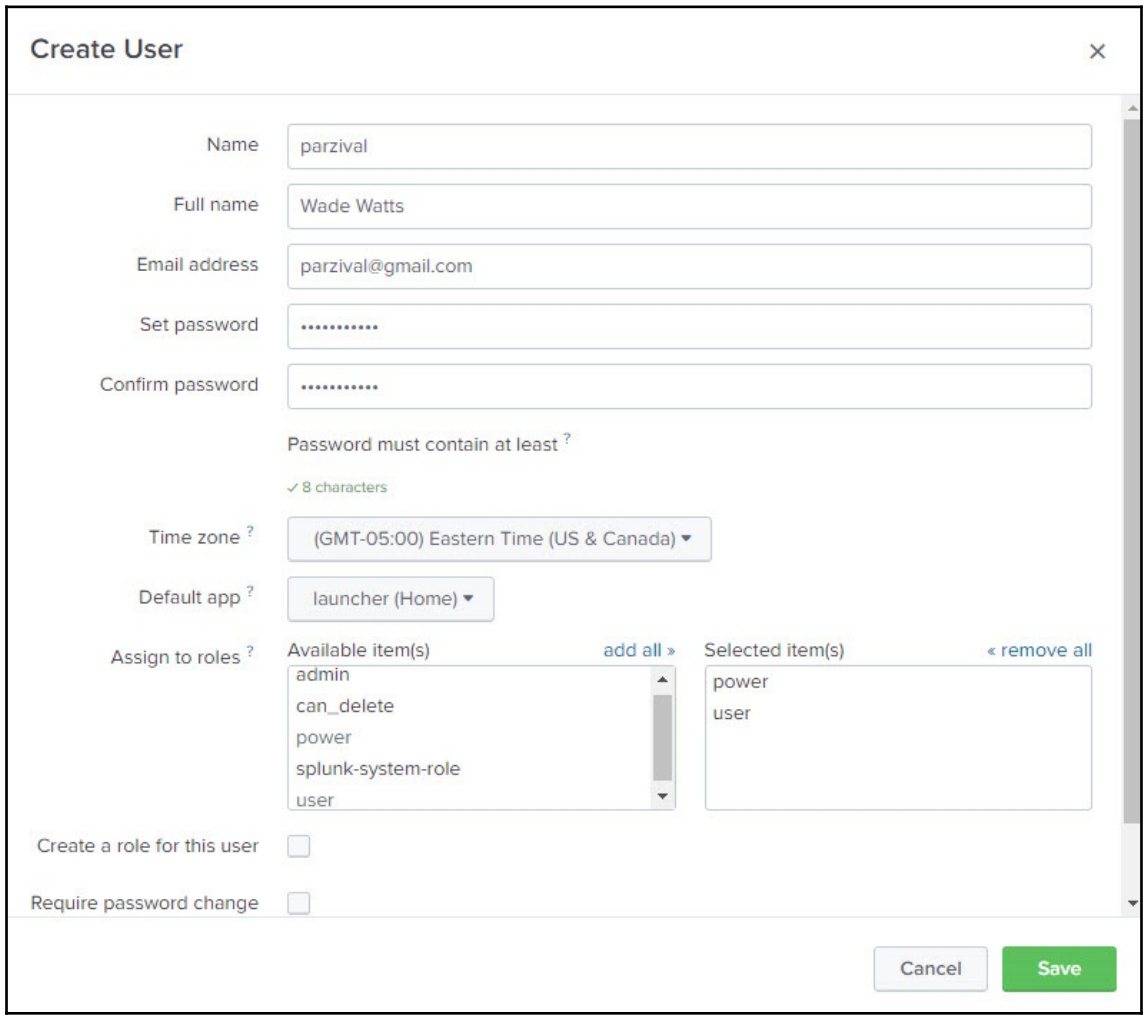


Fig 5.1: Creating a Splunk user

Out of the box, Splunk comes with several user roles defined. The most basic role is user; the other roles add capabilities to this basic set to various degrees. I will cover roles in more depth later in this chapter. The following are some of the roles:

- **user**: Basic set of capabilities for using Splunk to do searches.
- **can_delete**: This role must be assigned to allow a user (even admin) to delete indexes or events within indexes, or to schedule a real-time search. Typically, only admin-level users should be given this capability.
- **power**: This role allows a user to schedule saved searches for reports, dashboards, and to edit some knowledge objects.
- **splunk-system-role**: This is a unique user that all Splunk system jobs, such as saved searches and alerts, run as.
- **admin**: This role assumes all capabilities by default (except **can_delete**, which can be assigned).

The basic information about users defined using Splunk authentication is stored in a `passwd` file in the `$SPLUNK_HOME/etc/` directory; this includes each user's full name, a hashed version of their password, email address, and their assigned role(s) (such as user, power, or admin). If you need to migrate existing user settings from a standalone search head or search-head pool to a search-head cluster, you'll need to copy this `passwd` file to each member in the new environment (and possibly merge its contents with contents in the existing `passwd` files).

LDAP authentication

You configure Splunk to use LDAP authentication by clicking **Settings** | **Access controls** | **Authentication method** and selecting **LDAP**, at which time a configure Splunk to use LDAP link will appear. Clicking this will take you to a form to enter LDAP strategies; click **New LDAP**. A new form will open where you can enter a LDAP strategy name and configure all the settings required to access a LDAP server. You will need to get these settings and the Bind DN password from your LDAP system administrator. When you complete and save this form, these settings are saved in an `authentication.conf` file in the `$SPLUNK_HOME/etc/system/local` directory. Here are some fictional example settings:

```
#### Sample Configuration for Active Directory (AD)
[authentication]
authSettings = AD
authType = LDAP

[AD]
```

```
SSLEnabled = 1
bindDN = ldap_bind@splunksupport.kom
bindDNpassword = ldap_bind_user_password
groupBaseDN = CN=Groups,DC=splunksupport,DC=kom
groupBaseFilter =
groupMappingAttribute = dn
groupMemberAttribute = member
groupNameAttribute = cn
host = ADbogus.splunksupport.kom
port = 636
realNameAttribute = cn
userBaseDN = CN=Users,DC=splunksupport,DC=kom
userBaseFilter =
userNameAttribute = sAMAccountName
timelimit = 15
network_timeout = 20
anonymous_referrals = 0

[roleMap_AD]
admin = SplunkAdmins
power = SplunkPowerUsers
user = SplunkUsers
```

SAML authentication

You configure Splunk to use SAML single sign-on authentication by clicking **Settings** | **Access controls** | **Authentication method** and selecting **SAML**, at which time a configure Splunk to use SAML link will appear. Clicking this will take you to a form to enter SAML configuration settings. You will need to obtain the correct settings from the person who administers your SAML accounts. When you complete and save this form, these settings are saved in an `authentication.conf` file in the `$SPLUNK_HOME/etc/system/local` directory. Here are some fictional example settings:

```
[authentication]
authSettings = samlv2
authType = SAML

[samlv2]
attributeQuerySoapPassword = changeme
attributeQuerySoapUsername = test
entityId = test-splunk
idpAttributeQueryUrl = https://exsso/idp/attrsvc.ssaml2
idpCertPath = /home/splunk/etc/auth/idp.crt
idpSSOUrl = https://exsso/idp/SSO.saml2
idpSLOUrl = https://exsso/idp/SLO.saml2
```

```
signAuthnRequest = true
signedAssertion = true
attributeQueryRequestSigned = true
attributeQueryResponseSigned = true
redirectPort = 9332
cipherSuite = TLSv1 MEDIUM:@STRENGTH
nameIdFormat = urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress

[roleMap_SAML]
admin = SplunkAdmins
power = SplunkPowerUsers
user = all

[userToRoleMap_SAML]
samluser = user::Saml Real Name::samluser@domain.com

[authenticationResponseAttrMap_SAML]
role = "http://schemas.microsoft.com/ws/2008/06/identity/claims/groups"
mail = "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress"
realName = "http://schemas.microsoft.com/identity/claims/displayname"
```

You can find a great deal more information about setting up Splunk authentication by reading through the Splunk doc on `authentication.conf`—I've provided a link at the end of this section.

Managing Splunk roles

Regardless of which authentication method you employ, as your environment grows, you will want to create and assign specific roles to your various user groups to manage the capabilities they can exercise; which indexes, reports, and dashboards they can access, and whether they can only view the data or have the ability to create and edit Splunk knowledge objects and reports, and schedule searches to populate them. In addition, you can set parameters for the various roles to control how many concurrent searches a user or role can run, how much disk space their searches can consume, and a number of other limitations to help manage the capacity and performance of your Splunk environment.

The best way to get familiar with the various role settings is to review them in Splunk Web by clicking **Settings** | **Access controls** | **Roles** and clicking on one of the existing roles, such as **admin**. Let's briefly review these sections one by one.

Search restrictions

In the search restrictions section, you can manage settings to restrict search terms, time ranges, and a number of values for user-level and role-level concurrent real-time and adhoc or saved-search (from reports, for example) search jobs, as well as a total jobs disk-usage quota. Since each concurrent search consumes a CPU core on a search head, as well as consuming CPU on indexers, you may need to reduce the allowed number of concurrent searches as your user population grows, although you really should just increase the number of search heads and indexers if the search load is legitimate. When users begin to hit the concurrent search limits, they will receive Splunk warnings to this effect, and if the problem gets bad enough, saved searches will begin to be aborted during peak load periods. It's better to manage search-head and indexer performance and capacity as your environment grows and avoid the problem altogether—I'll touch on this further in `Chapter 10`, *Advanced Splunk*.

Capabilities

You can select from a wide range of Splunk capabilities that you want to assign to each role. The default capabilities assigned to each default Splunk role (user and power, for example) provide a good starting point. The go-to reference for understanding what each of these capabilities represent is the Splunk doc on `authorize.conf`—a link is provided at the end of this section.

Indexes

You can control which indexes a given role will search by default if no index is specified in a search, as well as the indexes this role is allowed to search. It is a very good idea to educate your user community to always specify an index (or indexes, if applicable) for their search, as well as to specify default indexes and limit the indexes each role can access, if you are going to develop specific roles for your various lines of business or other groups.

authorize.conf

All of your custom roles and the capabilities you assign to each role are stored in an `authorize.conf` file that is located, by default, in `$SPLUNK_HOME/etc/system/local`. Some typical entries for this file might include settings to prevent a user from scheduling a real-time search (consumes a lot of resources—use sparingly), specifying a default index, and limiting which index(es) a role can access. You'll notice in the following examples that there are three general categories of entries in an `authorize.conf` file:

- Changes to default role settings
- New roles that inherit from existing roles but add specific capabilities/limititations
- New roles where all capabilities are explicitly specified

```
# Changes to default role settings
[role_admin]
importRoles = can_delete;power;user
schedule_rtsearch = disabled
srchIndexesDefault = main
srchMaxTime = 8640000

[role_power]
schedule_rtsearch = disabled

# New role - inherits from existing role + specific settings
[role_sensor-admin]
importRoles = power
srchFilter = index=sensors_180d_midx

# New role - all capabilities specified
[role_api]
accelerate_datamodel = enabled
accelerate_search = enabled
admin_all_objects = enabled
...
```

Working with authentication.conf and authorize.conf

As in any other Splunk administration endeavor, it will serve you well to understand what the various entries in the `authentication.conf` and `authorize.conf` files do, and how these two files work together to control authentication and access to Splunk functions.

There is an `authentication.conf` and `authorize.conf` file located in `$SPLUNK_HOME/etc/system/default` that establishes the default settings for Splunk-based authentication and its default roles. When you modify any default setting using Splunk web, these changes are stored in the respective files in `.../etc/system/local`. You can, of course, edit these `.conf` files directly instead of using Splunk Web. You can also create smaller versions of these files in the `/local` folder of your Splunk apps, as needed, to configure settings that apply to those specific apps and the users that will be accessing them; Splunk will combine all the settings of the various `.conf` files upon startup.

Let's summarize how the two `.conf` files work together:

- Entries in `authentication.conf` specify the authentication method, the authentication server settings (if using non-Splunk authentication), and provide a mapping between various roles (which are configured in `authorize.conf`) to active directory or other authentication groups for your selected authentication solution.
- Entries in `authorize.conf` provide modifications to default roles, define custom roles, and specify the capabilities and limitations assigned to each role.

The Splunk docs are a must-read when setting up your authentication options and creating/assigning custom roles:

- `https://docs.splunk.com/Documentation/Splunk/latest/Admin/Authenticationconf`
- `https://docs.splunk.com/Documentation/Splunk/latest/Admin/authorizeconf`
- `http://docs.splunk.com/Documentation/Splunk/latest/Security/SetupuserauthenticationwithLDAP`
- `http://docs.splunk.com/Documentation/Splunk/latest/Security/HowSAMLSSOworks`

Best practices for administering Splunk

To avoid distractions in the previous sections of this chapter and in `Chapter 4`, *Getting Data into Splunk*, I've reserved a few additional comments on topics you will want to consider for establishing some best practices as you administer your Splunk environment, understanding of course that these have to be tailored to your particular organization's culture, needs, and IT environment.

Let's first talk about developing a naming convention for indexes and source types. A search of the web will provide a number of discussions and ideas on the topic; here are a few options I've settled on.

Index naming conventions

When creating indexes, use lowercase names (Splunk changes uppercase index names to lowercase anyway) with underscores between segments:

```
Index name convention 1:  organization_app
imdx_eservice

(Preferred) Index name convention 2: organization_app__retention_indextype
imdx_eservice_90d_eidx

This convention allows users to know what the retention period is, as well
as the type of index. The indextypes are:
eidx - events index
midx - metrics index
sidx - summary index

For os-level logs: os_<os>_retention_idxtype
os_nix_120d_eidx
os_nix_90d_midx  (using collectd)
os_win_120d_eidx
```

Source type naming conventions

The naming of custom source types should really reflect the source of the data, including (as applicable) the vendor, device type, app, and the type of data being indexed. Source type elements can be separated by underscores or colons, as exemplified by a couple of Splunk-provided source types and some variations; I personally like using colons between the elements:

```
vendor:product:type
cisco:asa
websphere_core
websphere:app:core
db2:diag or ibm:db2:diag
```

Location of indexes.conf, props.conf, and transforms.conf

There are two schools of thought regarding where to keep `indexes.conf` files on the cluster master:

- Place them in their associated app directory's `/local` folder along with that app's props, transforms, and other files. This keeps all the files for a given app together, but means you must navigate to each app to change something that might need to be altered across a lot of indexes (such as a frozen storage location, for instance).
- Configure a single `indexes.conf` file and keep it in `/splunk_indexes` or a similar app directory dedicated to this purpose. This makes it easier to manage all the indexes in one place. You could do the same with `props.conf` and `transforms.conf`, as well. The downside of this approach is you have to search through a lot of entries to find the specific ones you want to modify.

I imagine the answer to the question of which approach to use comes down to personal preference, and/or if you plan to use some level of automation to manage your environment, and what approach that solution might take to administering the files.

Here are some other tips and thoughts:

- Document everything you do—you'll thank yourself three months later when you've forgotten what you did and why.
- It is not practical to expect yourself, or anyone who comes in behind you to manage your Splunk environment, to remember all the possible settings in the various `.conf` files if you're not working on those files every day. To ease this burden and ensure all configuration settings are used appropriately in future, you can add comments to your `.conf` files (using a # before the comment line) for non-obvious entries, and/or entries that alter a setting from its default, and note what the default value was.
- Don't let users or management make you rush your configuration or implement anything you don't fully understand. You'll just get rushed into yet another project without mastering this one, and eventually you'll end up with an environment that behaves strangely, and you won't have a clue as to why or where to start looking for a problem that may never get fixed properly.

- On that note, if at all possible, create a small clustered Splunk solution in a sandbox where you can test configuration settings and new apps without fear of crashing a production environment. You can build a test environment in the cloud and shut it down when you're not using it; you only get charged for server instances when they're running. There may be a monthly charge for disk volumes, permanent IP addresses, loadbalancers, or other dedicated services, but the overall cost is relatively small if you're careful.
- If and when it's possible and practical (the earlier the better), you should consider implementing some level of automation for many of your Splunk administration tasks. This can free up a lot of time that could be better spent with your users, implementing improvements to your environment, reporting objects, or even applying some machine learning solutions for enhanced analysis or predictive analytics. While many automation solutions focus on manipulating `.conf` files on GitHub or similar solutions, you also want to consider that almost anything you can do from Splunk Web or the CLI can also be accomplished using REST API calls.

Supporting your Splunk Deployment

As important and challenging as setting up an efficient Splunk environment is, it is equally important - and sometimes just as challenging - to ensure that you have taken adequate measures to fund and support your Splunk solution so that it continues to provide value well into the future. We'll address several of the most important points in this section.

Splunk support personnel

As a rule of thumb, Splunk recommends a minimum of three Splunk support personnel for any significant Splunk deployment. With the following roles and related training/certification and experience, these roles and duties can overlap or be altered depending on the needs of the organization and skill sets within the Splunk team:

- **Architect**: Designs and helps to implement and support Splunk infrastructure and related configurations, monitors Splunk capacity and performance metrics, and plans and implements upgrades to hardware and Splunk software versions.
- **Administrator**: Configures and manages universal forwarders and data inputs, creates and manages indexes, installs and deploys Splunk apps, and manages indexing and search head cluster configuration files and users/roles.

- **Knowledge object manager**: Works with application teams and other data source groups to create and manage reports, dashboards, alerts, field extractions, event types, tags, data models, pivots, and related business information objects.

Another rule of thumb that can be applied in conjunction with the preceding staffing recommendation is that you'll need about one additional Splunk support person for each ~1 TB/day of ingestion volume. Assuming you have a three-person team, once you start to exceed approximately 3 TB/day you'll want to acquire an additional Splunk support person for each new TB/day of volume. If your Splunk environment is growing at a steady or accelerating rate, it is better to hire each additional person well before you reach the next TB milestone, instead of afterwards, to allow time for the new person to get up to speed and avoid an endless cycle of not-quite-enough support to maintain an efficient solution and the associated technical debt that can result.

As your Splunk user community expands, it is also advisable to train and develop Splunk power users within each group that can create and maintain the knowledge objects, reports, dashboards, and alerts needed by business unit, as well as provide support for lesser-skilled users. These resident power users have the distinct advantage of being more familiar with the data they are working with, as well as a better understanding of the needs of their part of the organization.

Funding Your Splunk deployment

While not a typical topic for a technical book, it would be remiss of me if I didn't include an urgent reminder that you need to plan for how the initial implementation and licensing, and, more importantly, the ongoing support, maintenance, and growth of your Splunk environment will be funded within your organization. This factor is *more* critical for ensuring a reliable Splunk platform and return on your business investment than any technical detail we'll discuss in this book.

It is common for Splunk to be introduced independently by several business units, and because it's a great product, to see its use grow beyond what these groups can, or want to, manage. Other factors (usually financial) can come into play such that ownership of these smaller Splunk environments is reassigned and combined under a focused logging/monitoring group. Unfortunately, these transfers do not always include (if they ever have) a workable plan for funding this integration and the ongoing support and growth of the overall environment (beyond repeatedly begging management for money).

The risk inherent in this scenario is a lack of adequate and timely funding to support necessary hardware and licensing growth, resulting in an overtaxed and poorly performing infrastructure. It could also cause a shortage of the number of adequately trained and experienced Splunk personnel needed to keep the environment running at optimal performance and reliability to provide support to end users, and help to extract the tangible business value that Splunk was implemented to provide in the first place.

One method of obtaining the needed funding for Splunk support is to implement a charge-back system, where the business units responsible for providing critical services to your company are required to budget and allocate to the Splunk team a calculated sum on a monthly or annual basis, based on the measured (or estimated, for new demands) average daily data volumes they are sending to Splunk for ingestion from log files or other sources. This approach has two advantages: since the Splunk resource needs and licensing costs are generally based on the volume of data being ingested, and there is a reasonable correlation of numbers of users and searches to this volume, the business is directly paying for the resources it is consuming, and the Splunk team can procure new hardware and other resources as needed to proactively support the growing environment. In addition, because sending Splunk more data costs them more money, this approach incentivizes the business units to optimize and limit the volume of data they index; otherwise, you're more likely to see log files bloated with unnecessary or duplicate entries, or large numbers of multiline stack trace entries, even in production environments.

If your user community creates a very large number of reports, dashboards, and scheduled searches, such that you find yourself needing to add search heads and indexers just to support the search load, you may need to impose a surcharge for this as well.

Splunk resource cost calculations

The significant items to include in Splunk resource-cost calculations include:

- **Hardware**, such as CPU/memory/disk, or equivalent cloud instances, and any network, firewall, and load-balancer services
- **Splunk licensing**: This includes the initial volume-based licensing and annual support fees (again, by volume) for both Splunk Enterprise or Splunk Cloud, and any Splunk Premium applications (such as Enterprise security or IT service intelligence)
- **Administration and support**: Splunk personnel and training costs

Since there is an initial up-front investment for the Splunk license and any incremental increases, the calculated ingestion-volume-based charges in a charge-back or other funding plan should be cost-averaged over a three-to-five year-term, perhaps to align with any other CapEx schedules, such as for hardware. Splunk searches can be created to periodically report indexing volume and concurrent search counts to provide feedback and validation to your data sources for billing/budgeting purposes, and to assist in further planning for resource growth and cost projections.

Again, a well-thought-out and well-implemented charge-back, or equivalent, funding solution will ensure you have the physical and human resources needed to provide an effective, reliable, and valuable big data environment for your business.

Summary

This was a relatively short but important chapter! After learning how to deploy apps to search-cluster members with the deployer, we discovered how to set up user authentication and define the various roles to control access to Splunk and its various capabilities. Finally, I offered a few best practices on administering your Splunk environment and some key considerations for supporting it into the future.

If you've waded through (and hopefully experimented with) all the functionality in this chapter and `Chapter 4`, *Getting Data into Splunk*, you now know how to expertly administer *all* of the Splunk components in a clustered, distributed environment. In addition, you understand how the various configuration settings for each function are represented in the `.conf` files for each component, as well as a great deal about how Splunk actually executes the administration tasks, which should be helpful in troubleshooting scenarios.

These first five chapters have had a heavy focus on architect- and administrator-level knowledge. Starting with the next chapter, we'll move into working with Splunk more from a user perspective – creating searches, knowledge objects, reports/dashboards/alerts, and Splunk apps. See you there!

6
Searching with Splunk

This chapter is perhaps the most important part of the entire book as it covers how to use Splunk's Search capability to get data out of Splunk indexes, reduce it to its essential components, and then transform and format the results into a dataset and visualizations that provide real value and useful insights. The important features of the user interface — Splunk Web—are leveraged in working examples of the more basic **Search Processing Language** (**SPL**) commands, which serve as the foundation for a gentle and logical progression to using the more advanced commands and visualization options.

Topics we will cover in this chapter include:

- Essential SplunkWeb interface features and controls
- How to format basic **Search Processing Language** (**SPL**) search strings
- How to create filters to reduce the amount of data returned from searches
- How to use the two most powerful search commands — `eval` and `stats` – to create calculated fields
- More search commands – `chart` and `timechart`, `table` and `fields`, `rename`, `rex`, `sort`, `top`, and `where`
- How to do subsearches and use the join and transaction commands for complex result sets
- How to select and configure the visualization controls
- How to optimize searches – search jobs and the job inspector

If you haven't already, now is a good time to get the Splunk Quick Reference Guide as you will be referring to it often to help create search strings:

`https://www.splunk.com/pdfs/solution-guides/splunk-quick-reference-guide.pdf`

The Splunk Web interface

You can log into Splunk Web using the appropriate URL and the default port of `8000`: `http://<hostname or ip address>:8000`. If you have configured Splunk to use HTTPS (a good idea), you will need to reflect the port you configured in the `web.conf` file. Another example, using a fully-qualified domain name, is `https://mysplunkserver.mycompany.com:8443`. After logging in with your username and password, you will be presented with the Home interface (unless you'd selected another default page in the past), which features a list of clickable icons for all the installed applications along the left-side. One of the icons along the left side menu will be **Search & Reporting** – click that icon to open the **Search** interface:

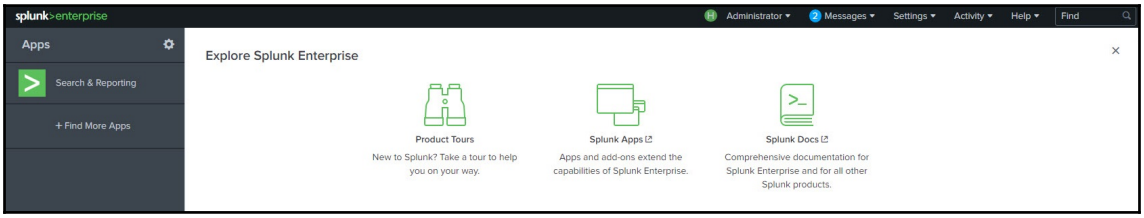


Fig 6.1: Splunk launch page

At this point, you're ready to enter a search command using **Search Processing Language** (**SPL**), which consists of a series of commands and arguments that specifies the data you're looking for.

Let's enter a basic command that will return the data from a series of events ingested from a web server's logs. This data reflects the configurations discussed in `Chapter 4`, *Getting Data into Splunk* where we set up a universal forwarder to monitor an Apache web server and ingest the logs into an index called `weblogs_90d_eidx` with the `access_combined` and `access_error` sourcetypes for the access and error logs, respectively. The initial search command is `index=weblogs_90d_eidx` with **Last 24 hours** selected in the time-range drop-down menu on the right. This is a good time to make the point that, in essence, *all* of your searches will initially be based around these two factors:

- Using SPL commands to find and filter the data to include just the events you're looking for
- Selecting the shortest time range that the events occurred in, or the time range you want to inspect if you're looking for something

Regardless of how complex your searches become, they will always come back to these two criteria.

Executing the preceding search returned 216 events for the last 24-hour period; usually a search on an index with no other filtering criteria will return thousands or millions of events:

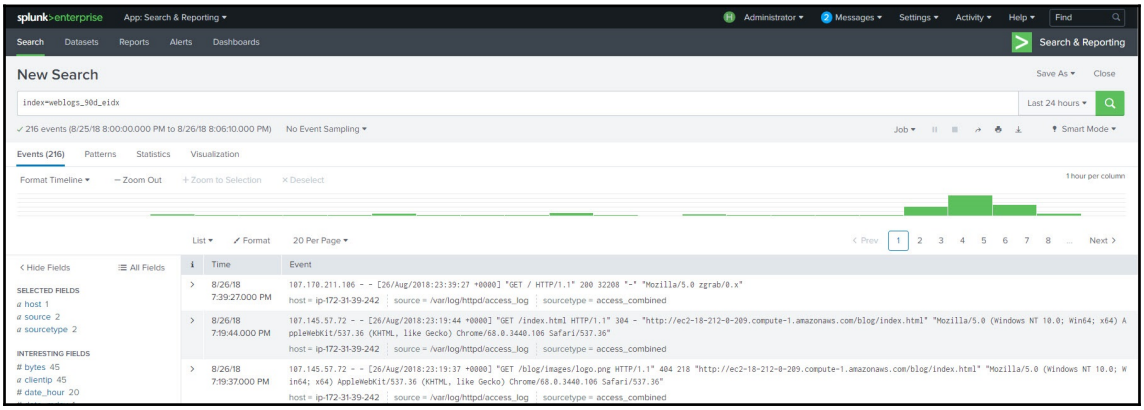


Fig 6.2: Basic search results in Splunk Web

If you have admin access to Splunk and want to display some events so you can follow the interface controls discussion in your own environment, you can execute a search on `index=_internal`. If you don't have admin access, and don't know what indexes are in use in your environment, the following search string will provide a list of all the non-internal indexes and a count of the number of events in each index for the time range selected (click and select **Last 15 minutes** to reduce the search load) and then pick one of the indexes to use with the `index=` command (without the `| stats count by index` part) to display some events:

```
index = * | stats count by index
```

Search controls

The top section of the search interface provides a variety of selections that focus mostly on search commands and controls. For the sake of thoroughness, I'll touch on all of them briefly and expand on the controls you'll use frequently:

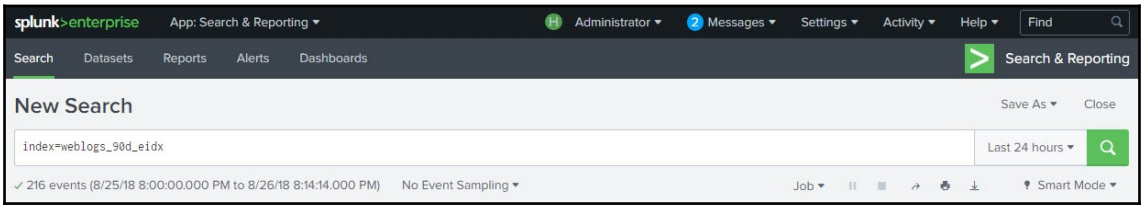


Fig 6.3: Search controls

The very top row of the Splunk Web interface holds the applications drop-down menu and the Splunk bar, which includes the following controls:

- **splunk>enterprise**: Clicking on this logo will take you back to the homepage.
- **App: Search & Reporting**: Clicking this drop-down menu will allow you to select other Splunk apps, manage apps, or find more apps on Splunkbase.
- **(H) icon**: New with Splunk 7.x, clicking this icon will give you a quick overview of the health of the Splunk instance you're logged into.
- **Administrator (or your username)**: Clicking this and selecting **Preferences** allows you to select (on the **Global** tab) the timezone you want events displayed in and the default app when you log in. Under the **SPL Editor/Global** tab, you can select the **Advanced editor** options, which include various levels of color-coding and search assistance (hints provided while you're writing SPL), and whether you want line numbers and help formatting your SPL strings. I highly recommend enabling all of these options. The **Themes** tab allows you to change the appearance of the search command bar:

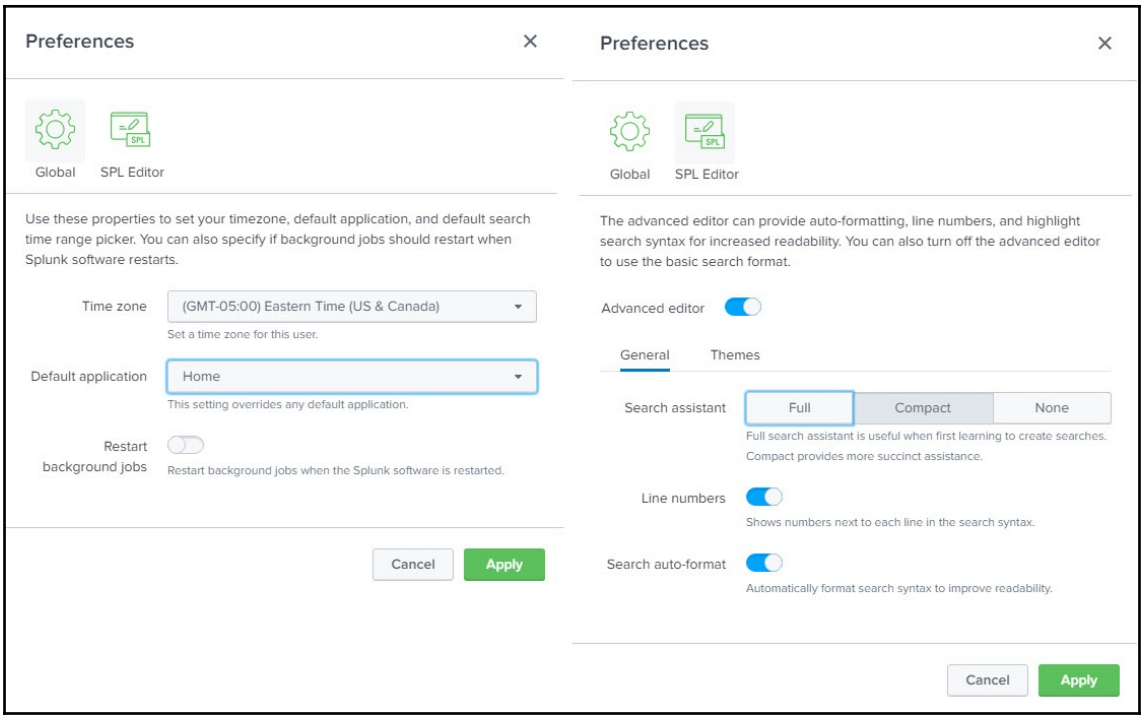


Fig 6.4: User preferences

- **(#) Messages**: Clicking this drop-down menu displays the most current system-level health messages Splunk has generated. These are well worth reading, and then can be deleted. The basis of these messages will have been reflected in splunkd or other internal log events.
- **Settings**: You will have seen many of these when performing administrative tasks, and we'll cover others in chapters to come.
- **Activity**: You can view details about specific search jobs and triggered alerts—we'll cover these in later sections.
- **Help**: This provides a variety of links to things such as documentation, tutorials, and Splunk support. The **Help** | **About** page is useful for identifying the name of the server your Splunk Web interface is being served from (especially if you attached to a search head from a Virtual IP (VIP) URL and want to know which server in the cluster you're actually on), the Splunk version, and—if you're working within a Splunk app—the name and version of that app.
- **Find**: Search for Splunk items that match the entered string; this is mostly a keyword search function.

Under the Splunk bar is the **search bar**, where you enter your search criteria using SPL commands.

Above and to the right of the search bar is a **Save As** drop-down menu for saving searches as reports, dashboards, or alerts, and saving a search string as an event type. We'll cover all these in later chapters. The **Close** link clears any previous search strings and results, and resets the page for a new search.

To the right of the search bar is the **Time Range Picker**, where you select the time range for your search, such as the **Last 60 minutes**, **Last 24 hours** (the default), or the **Last 7 days**.

Finally, the **magnifying glass icon** can be clicked to initiate a search – but you can also just press *Enter* to start a search:

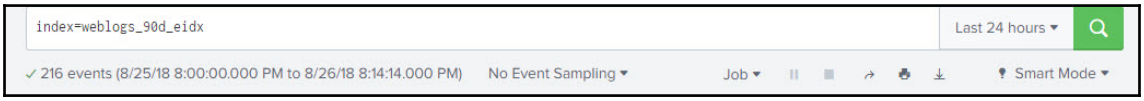


Fig 6.5: Search bar controls

Below the search bar, Splunk will indicate the number of events that matched the search criteria, and the time range that the search covered – these are useful metrics to watch as you build search filters and/or during troubleshooting sessions.

If you are doing some statistical analysis, you may want to select an **Event Sampling** option of **1:xxx** – but for most searches, you'll leave this in its default No **Event Sampling setting**.

Next over: when Splunk runs a search, it is called a job and the **Job** drop-down control allows you to edit Job settings, inspect a job, or delete a job. When Splunk runs a search job, it saves all the results, such as the search criteria, in files in a search artifact directory on the search head for a default of 10 minutes (in the `$SPLUNK_HOME/var/run/splunk/dispatch` directory); the artifact directory is deleted after it expires. However, if a search contains some valuable/shareable results, you can click **Edit Job settings**, change the save time to **7 days**, expand the permissions, and click to copy a link that you can send to others to view the search results (without rerunning the search). We'll cover more about the Job inspector in the last part of this chapter.

Next to the Job drop-down list are controls to:

- **Pause** or **Resume** the search.
- **Stop** the search.
- **Share** the search (the same controls as Edit Job Settings except it auto-extends the save time).

- **Print** the search result pages.
- **Export** the search result events as raw text, or in CSV, XML, or JSON formats. You can name the file and specify how many events to export.
- Select **Fast, Smart**, or **Verbose** modes of displaying search results. The Fast mode is faster, but Splunk won't extract and display any more than just a few very basic metadata fields in the left-hand-side field column (we'll cover that next). Splunk will extract fields from events in the Smart or Verbose settings (Verbose for use with stats searches), and these extracted fields can be *very* handy for analysis and troubleshooting scenarios, so I usually leave this setting on at least Smart mode (default).

Timeline and events

Below the search bar and search controls is the timeline. When the **Events** tab is selected, this visually depicts the number of events observed for each incremental period of time; the time period represented by each time column is automatically adjusted depending on the time range selected in the time-range picker. Peaks, valleys, and gaps observed in the timeline indicate changes in activity or server downtime; if you are using one or more search filters, you can see when events occurred that match the filter.

If you click the **Patterns** tab, you can see the results of some behind-the-scenes analysis Splunk does using the cluster command to identify common characteristics in the returned events – this can be useful for identifying the most common types of events in a search result dataset.

The **Statistics** tab displays search results in tabular format if you use what is called a transforming command in your search string, which creates a data series output; otherwise this and the **Visualization** tab only display additional data-formatting options. The **Visualization** tab provides options for charting data – more on this a bit later in the chapter.

You can click **Format Timeline** to change the size of the timeline or even hide it, and/or change from linear to log scales. The **Full** timeline displays date-times on the **X**-axis, and a scale on the **Y**-axis; the **Compact** timeline does not, but you can hover over a column to see the date-time and number of events. The following screenshot illustrates the full timeline display:

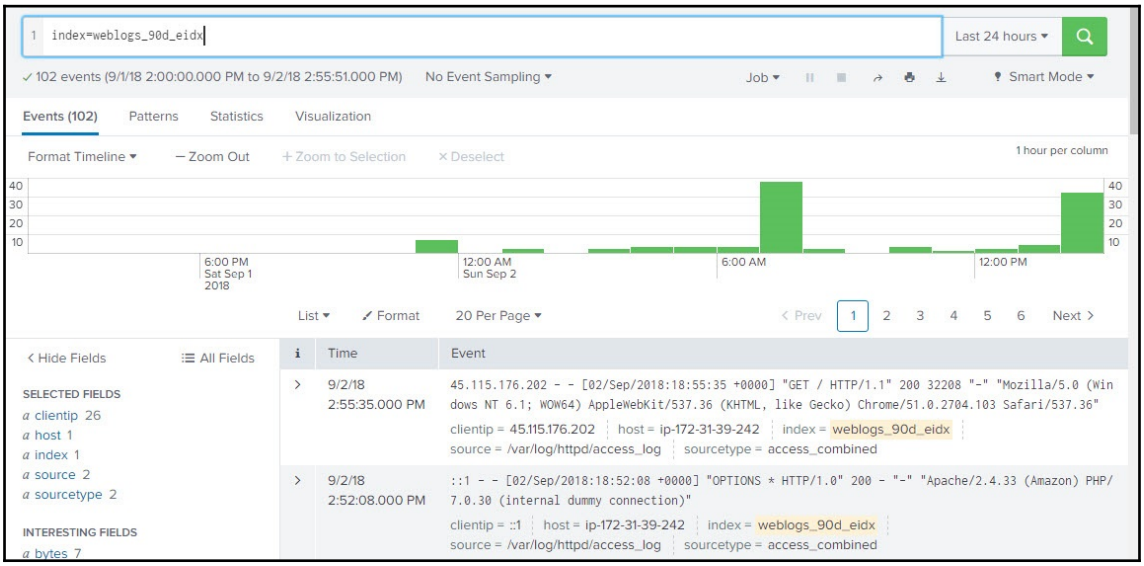


Fig 6.6: Splunk timeline and events

Under the timeline is the **Events** section. You can click **Raw**, **List**, or **Table** from the drop-down menu to select how you want to view the contents of each event; **List** is the default and probably the best view most of the time. Clicking **Format** allows you to select several display options – Row Numbers, Wrap Results, and Max Lines per displayed event. The **Click Selection** control determines the granularity and related selection behavior of the clickable objects inside each event for adding, excluding, or running a new search from the selected text. The functionality of the **per-page** and **page selector** controls operate as you would expect.

The **Fields** section to the left of the events (which can be hidden by clicking **Hide Fields**, but you'll usually want this displayed) provides some very useful tools for working with events – you'll want to experiment with this until you're comfortable with using it as soon as possible. Under the **SELECTED FIELDS** area, you will always see the four key metadata fields that Splunk associates with every event as it is indexed: host, index, source, and sourcetype. Beside each entry is the number of unique values observed for that field.

If **Smart Mode** or **Verbose Mode** is selected under the time-range picker, in the **INTERESTING FIELDS** section, you will see a list of all the other fields Splunk has recognized within the entire set of events returned from the previous search. You can peruse this alphabetical list looking for fields of interest, and then click on that field to display a variety of additional information about that particular field, such as the number of different values for that field, what percentage of events that field appears in, the **Top 10 Values**, **Count**, and percentage of each unique value. You can click **Yes** from the **Selected** buttons to have that field appear under the **SELECTED FIELDS** list, and have that field and its value appear in the field summary at the bottom of each event. Note that if a particular event does not have a given selected field, that field will not be represented in the summary:

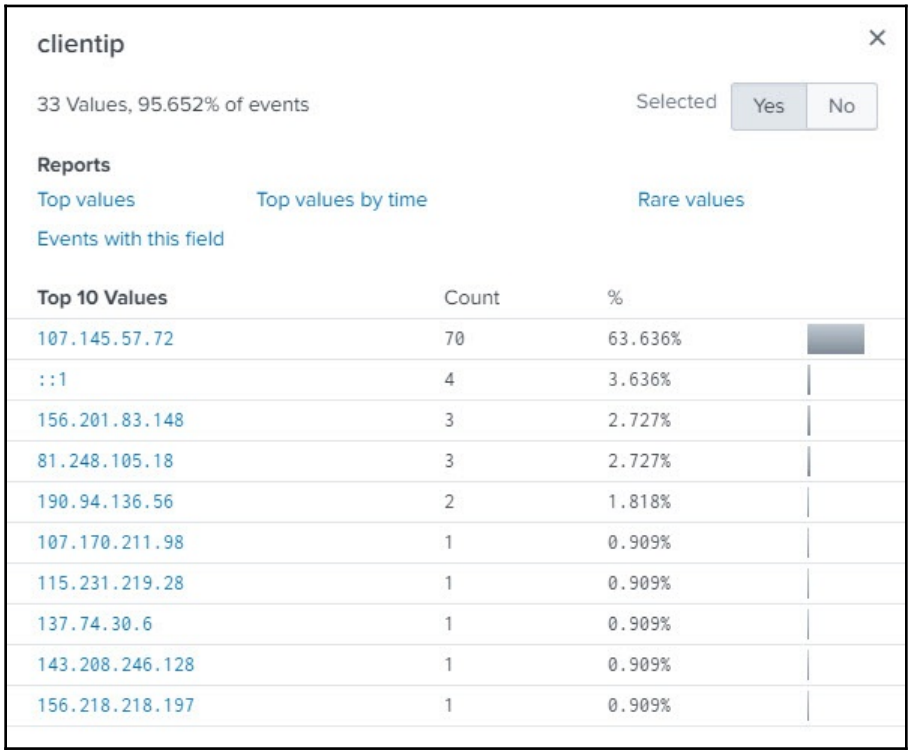


Fig 6.7: Event field details

That covers the most important/useful controls in Splunk for viewing search results; let's start creating some search commands so we have some events to look at!

Creating Splunk searches

A Splunk search is created from a series of commands and arguments using SPL. The commands and arguments are chained together using the pipe character (`|`) such that the output of one command is fed into the next command to the right.

Be aware that Splunk assumes the very first command on the search bar is **search** – you don't have to add it (unless you're creating a sub-search, in which case **search** is the first command to be given). I like to use *Ctrl + Enter* to stack my commands within the search bar—this makes it much easier to read and comprehend search strings, such as the following:

```
index=<index> <filter> <"text string to match">
| command1 <arguments>
| command2 <arguments>
| visualization commands & arguments
```

You can create searches most effectively by following some basic steps. You will find it most effective to build search commands progressively, performing the following general steps and executing the search for each step to make sure you get the results you expected, and, most importantly, that you fully understand what each command does. If you're not sure how a command or its particular arguments work, search through the Splunk docs or answers, review the examples given, and experiment with the command and a few of its arguments until you have mastered the command. Especially when you're first learning to work with SPL, I can't stress the value of this approach enough; follow it, and soon you'll be typing in SPL commands with confidence and minimal research. Here are the basic steps to building a SPL search string:

1. Specify an index and time range that will retrieve all of the events you're interested in.
2. Add filters to better specify and reduce the number of events to just those you want to work with, and eliminate the rest.
3. Progressively pipe the events/data to the next command plus any applicable arguments and check that you get what you expected from each step.
4. Pipe the final dataset to a table, chart, or time chart for visualization and adjust the visualization settings.
5. When the search is working as desired, save it to a report, dashboard panel, or alert as desired.

This process is also applicable if you're troubleshooting (or just trying to understand) an existing search – make a copy (or create a new duplicate browser tab, leaving the original SPL intact elsewhere to reference), then strip off all of the search commands except just the index and perhaps a filter or two (or back those out as well). View the results, then progressively add sections of the search string back in, making sure it works and that you understand each part. Eventually you'll confirm the search is working as intended, or identify and resolve any errors.

Basic search commands

The two topics in this section—index and time range—cover the basic commands you will want to employ in almost all of your searches to ensure that you get all the events you're looking for, but no more than necessary—this helps improve the performance of your searches and keep resource usage to a minimum.

Index

The first and most basic search command that you can execute is to specify an index. Unless you are looking for events containing a particular text string and you have no idea what index or sourcetype those events might be found in, you should *always specify an index* at the beginning of your search string to avoid searching the entire set of indexes, which is terribly wasteful.

Splunk has internal indexes, such as `_internal`, `_introspection`, and `_audit`, that contain events from the internal, which are kept in `$SPLUNK_HOME/var/log/introspection` and `$SPLUNK_HOME/var/log/splunk`. By default, you must have admin privileges to view the events in these indexes.

Then there are the custom indexes that are created by various Splunk apps and the indexes you will configure in `indexes.conf` for apps that you create; you will need to know what these indexes are or do a search to find out.

If you don't know which indexes, sources, or sourcetypes are available in your Splunk environment, you can run a search that will list all of them, and then pick an index to start with; hopefully, the index-naming convention in use will reflect the type of the data in each index. Keep in mind that the ability to see and retrieve events from a given index may be limited by your user role:

```
index=_* | stats count by index, source, sourcetype     // Splunk internal
logs
index=* | stats count by index, source, sourcetype      // app or custom
indexes
```

Time-range selection

The next criterion that you should carefully select is the date-time range for the events you want to retrieve. Obviously, you want a wide enough range to retrieve all the events of interest, but only wide enough—it takes more time and processing resources to retrieve events for longer periods of time. You may want to create a report for the previous hour, 24 hours, or the last 7 days—but searches for 30 days or longer should only be run if you really need the data.

You can select one of the numerous settings from the **Time-Range Picker**, which has numerous sections:

- **Presets**: Pick from a wide range of selections. You should avoid using **All time** unless you really need to use it.
- **Relative**: Specify a relative time in the past to now, or the beginning of the current minute.
- **Real-time**: View events as they arrive to be indexed in a sliding window of the selected size. Note that real-time displays pick up events before they are indexed, and have an effect on indexing performance, so use this option judiciously.
- **Date Range** and **Date & Time Range**: Specify a date or date and time range.
- **Advanced**: Specify an earliest and latest date or time range using **Time modifiers**. These time modifiers can be used in your search strings (instead of using the time-range picker); if you enter your time modifier entries into the fields in this tab, you can view the true date and time ranges that the specifiers you created will cover – the ranges are displayed in the shaded areas below the entry fields.

Splunk time modifiers use the following syntax:

```
earliest=[+|-]<time_integer><time_unit>@<snap_time_unit>
latest=[+|-]<time_integer><time_unit>@<snap_time_unit>
now()

where
time_integer is an Epoch value (number of seconds since January 1, 1970 at
12:00:01 AM)

time_unit is a time unit abbreviation such as s for seconds, m for minutes,
h for hours, d for days, etc.

@snap_time_unit is a 'snap-to' unit - set the time rounded down to the most
recent minute, start of hour, start of day, etc.
```

In the following example, the events for the entire 24-hour period, from two days prior to one day prior, are verified in the **Advanced** tab, and then utilized within the search string:

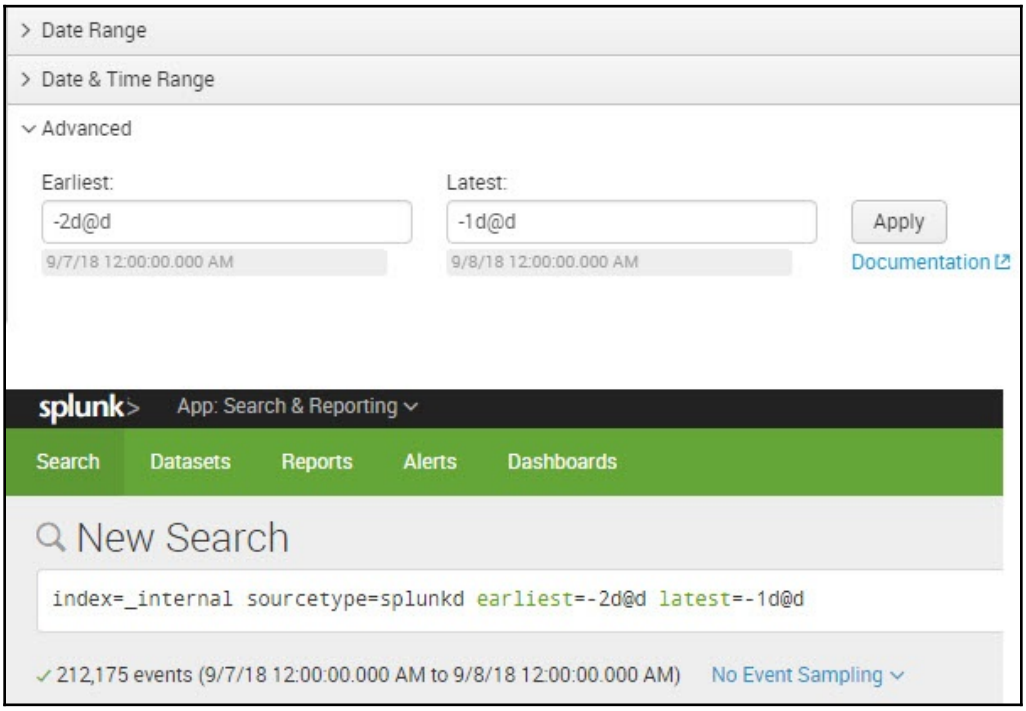


Fig 6.8: Use of time modifiers in the time-range picker and search bar

The supported time units and their abbreviations are listed in this table:

Time unit	Unit abbreviations
Second	s, sec, secs, second, seconds
Minute	m, min, minute, minutes
Hour	h, hr, hrs, hour, hours
Day	d, day, days
Week	w, week, weeks
Month	mon, month, months
Quarter	q, qtr, qtrs, quarter, quarters
Year	y, yr, yrs, year, years

Note that if you are using time modifiers in a search string in Splunk web, the time-range picker selection must be wide enough to accommodate the time modifier range or you will get incomplete results. Time modifiers utilized for search strings in saved reports, dashboards, alerts, and so on will of course not be affected by a Splunk Web time-range consideration.

There is a wide range of time modifiers you can use; the preceding examples are just a sample. You can see more examples in the Splunk Quick Reference Guide, and all the time modifiers in the Splunk docs:

`https://docs.splunk.com/Documentation/Splunk/latest/SearchReference/SearchTimeModifiers`

Search filters

After specifying an index and time range, you can (and should, anywhere applicable) apply filters to reduce the number of returned events, and isolate the result set to just the events of interest.

The most-commonly used filters are to specify a sourcetype (which will typically reflect a particular log type, and data from numerous sources), followed by one or more field specifiers or text strings to search for. For example, the search string in the following figure returned events from the weblogs_`90d_eidx` index, where the sourcetype is `access_combined`, the status field contains values of 400 and above, and the event contains a text string of logins. This was for a search that covered the last 30 days; using this filtering criteria reduced the event count from what could have been multiple thousands of events to only **32**:

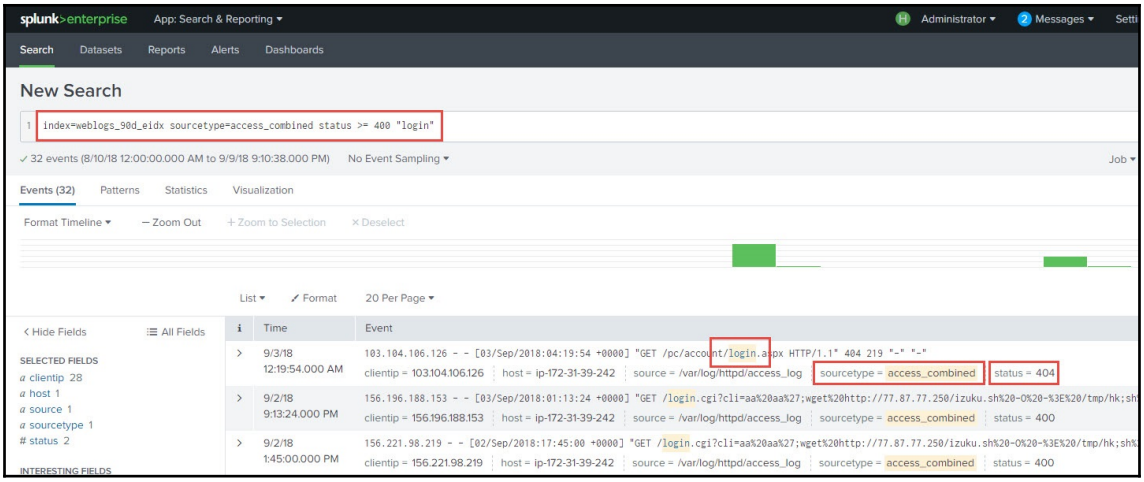


Fig 6.9: Using search filters

There is an implied `AND` between the filter criteria in a search string. The preceding search is interpreted the same as in the following search; in fact, you *could* write the search this way and it would work the same:

```
index=weblogs_90d_eidx AND sourcetype=access_combined AND status >= 400 AND
"login"
```

You can specify an `OR` or a `NOT` in your search filters, using all capitals; use parentheses for grouping if needed. You can also utilize wildcards (asterisks). You should avoid using the `NOT` operator, and broad usage of wildcards, to reduce the search-processing overhead:

```
index=weblogs_90d_eidx sourcetype=*access* (status = 400 OR status = 404)
NOT "login"
```

Finally, be aware that specifying that a particular field exists and contains a value using `<field>=*` will display only those events that contain the given field and actually have a value for that field; this can be a useful filter as well in some circumstances.

Search commands

At this point, you've used an index and various filters to retrieve the events you want to work with. You're now ready to pipe those events to other search commands to modify, supplement, or format the data for reporting or visualization. The next several sections introduce many of the most useful search commands you will use, starting with the most-commonly used: `eval` and `stats`.

Eval

The `eval` command calculates an expression and puts the resulting value into a field; this can be used to create a new field, or to replace the value in an existing field. You can use arithmetic operators (+,-,*,/,%), string concatenations (`eval name = last.",".first`), and Boolean operations (AND, OR, NOT, XOR, <, >, <=, >=, !=, =, ==, and LIKE).

The syntax is `| eval <new_field_name> = function(arguments).`

Some of the most useful examples include:

Function	Description	Examples
`if(X,Y,Z)`	If X is TRUE, the result is Y; otherwise Z	`| eval status = if(code==200,"OK","Error")`
`len(X)`	Returns the character length of X	`| eval bytes=len(_raw)`
`like(X,"Y")`	Returns TRUE if X is like the SQLite pattern in Y	`| eval match = like(field, "addr%")`
`max(X,...)`	Returns the maximum value of X and optional additional fields	`| eval maxdelay = max(delay)` `| eval maxdelay = max(fetchtime, proctime)`
`min(X,...)`	Returns the minimum value of X and optional additional fields	`| eval mindelay = min(delay)`
`now()`	Returns the current time in Unix (Epoch) time	`| eval current_time = now()`
`round(X,Y)`	Returns X rounded to the Y number of digits	`| eval delay = round(delay, 3)`
`strftime(X,Y)`	Returns Epoch time value X rendered using the pattern Y	`| eval current_time = strftime(now(), "%Y-%m-%d %H:%M:%S")`

Stats

The `stats` command is used to perform statistical functions on numeric values in event fields. The `stats` functions listed here are also used with chart and timechart commands, which we'll cover shortly. Some useful examples of the `stats` functions include:

Function	Description	Examples
`avg(X)`	Returns the average of the values in field X	`| stats avg(bytes) as avg_length`
`count(X)`	Returns the number of occurrences of field X	`| stats count(status) by status`
`max(X)` / `min(X)`	Returns the maximum/minimum value of field X	`| stats max(proctime) as max_proctime`
`perc<X>(Y)`	Returns the X-th percentile value of Y	`| stats perc95(proctime) as p95`
`stdev(X)`	Returns the sample standard deviation of X	`| stats stdev(proctime) as std_pt`
`sum(X)`	Returns the sum of the values in field X	`| stats sum(bytes) as tot_bytes`

Note that the value returned from the stats operation can be assigned a new field name by using as `<new_field_name>` after the stats function, as in the preceding example of stats avg(bytes) as `avg_length`; otherwise, the calculated value is assigned the name of the function, such as `avg(bytes)`.

You can combine multiple `eval` or `stats` commands separated by commas, and use the by operator to have the `stats` command calculate and/or display values over one or more fields, for example, to create a statistical table of a web server's processing times by host and URI:

```
...
| eval proctime = (end_time - start_time), rtsec = (proctime/1000)
| stats min_rt = min(proctime), max_rt = max(proctime), avg_rt =
avg(proctime) by host, uri
```

In the following example of using the `eval` and `stats` commands, stats is used to get the average number of bytes of data sent to each clientip for each user agent (browser) a particular client might be using (Chrome, iphone, ipad, and so on). The eval command is used to round the fractional average bytes value down to a whole number:

```
index=weblogs_90d_eidx sourcetype=access_combined
| stats avg(bytes) as avglength by clientip, useragent
| eval avglength = round(avglength, 0)
```

You can add the sparkline option to the stats command to include a small graph of event counts over time in the results table:

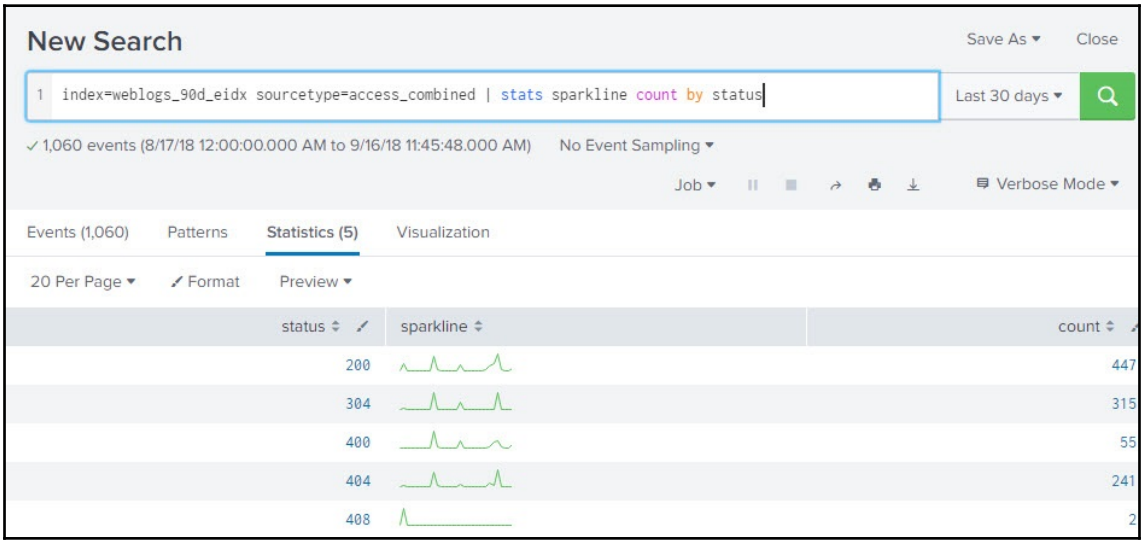


Fig 6.10: Adding a sparkline to stats results

When you use a transforming command, such as stats, a table of values is created with multiple rows and columns, and you can view the results in Splunk Web in the **Statistics** tab; Splunk will automatically switch to the **Statistics** tab if you execute search commands that result in tabular output.

The `eval` and `stats` commands are probably the two most useful SPL commands for manipulating data in your events; you'll use them a lot, so they warrant some experimentation to increase your comfort level with them. See the Quick Reference Guide and the Splunk docs for more examples and information about eval and stats:

- `http://docs.splunk.com/Documentation/Splunk/latest/SearchReference/Eval`
- `http://docs.splunk.com/Documentation/Splunk/latest/SearchReference/Stats`

Dedup

Dedup removes events that contain an identical combination of values for the specified field(s).

To eliminate all the events but one for a given host, or to eliminate duplicate events altogether, perform the following:

```
... | dedup host
... | dedup _raw
```

You can specify the number of duplicate events to keep for each value of a single field, or for each combination of values among several fields. In this example, for events that have the same source `AND` host values, we keep the first three that occur and remove all subsequent events:

```
... | dedup 3 source host
```

Rex

The `rex` command is used to perform field extractions using regular expression named groups, or to replace or substitute characters in a field using sed expressions.

The `rex` command matches the value of the specified field against the unanchored regular expression and extracts the named groups into fields of the corresponding names. If a field is not specified, the regular expression is applied to the `_raw` field.

Applying a rex command to an entire event (`_raw`) instead of a specific field may have a performance impact for larger result sets.

In the following example, a new field, called starttime, is created from a `sys_created_on=text` entry in the event followed by a date-time stamp; the new starttime field will contain the timestamp:

```
| rex field=_raw "(sys_created_on=\"(?<starttime>[0-9-: ]+)\")"
```

When `mode=sed`, the given `sed` expression will replace or substitute characters in the value of the chosen field. If a field is not specified, the `sed` expression is applied to `_raw`, with the same potential performance impact.

In the following example, the `sed` syntax is used to match a regex to a series of credit-card numbers and replace them with Xs and the last four digits; in the second line, an alpha-numeric sequence, such as `AADE364B-FDF6-4AA3-9581-15BAF0D12CE7`, is replaced with `<anXX>` to represent the entry:

```
... | rex mode=sed field=ccnumber "s/(\d{4}-){3}/XXXX-XXXX-XXXX-/g"
... | rex mode=sed field=exception_message "s/(\/[0-9A-Za-z-
]+\/)/\/<anXX>\//g"
```

Where

The `where` command is used to filter search results. It uses Boolean eval-expressions, where the expression returns either **true** or **false**, and returns only the results for which the eval expression returns true:

```
... | where proctime > 1500
... | where proctime > maxproctime
```

The where command uses the same syntax as the eval command, and both commands interpret quoted strings as literals. If the string is not quoted, it is treated as a field name. Because of this, you can use the `where` command to compare two different fields, which you cannot use the other search commands for. Note that if you are using the where command to compare two numbers, and one or both are actually strings, you must convert them to numbers with the `tonumber()` command to get this to work:

```
... | eval proctime = tonumber(proctime) | where proctime > 5
```

Formatting commands

After you have retrieved and massaged your events to contain just the data you want to inspect, you may need to format the output data in preparation for presentation; the following sections cover the most common formatting commands.

Rename

You can rename a field using the `rename <fieldname> as <newfieldname>` syntax. If you rename a field with whitespace or special characters, you'll need to enclose the field in quotes for both the rename command and anywhere else you use the field *after* the rename operation:

```
... | rename proctime as ProcessingTime | table URL, ProcessingTime, ...
... | rename proctime as "Processing Time (ms)" | table URL, "Processing
Time (ms)", ...
```

Sort/reverse

The `sort` command, as you probably guessed, allows you to sort your search output data by one or more fields. You can sort descending by putting a – in front of any of the fields. You can also reverse an entire set of events using the `reverse` command:

```
...| table url, proctime | sort -proctime
...| reverse
```

Head/tail

The `head` command returns the last N number of results (default = 10) in descending received order.

You can also utilize an eval-type argument in parentheses with the `head` command to display all initial results up until the first result where the eval-expression evaluates to false; there is a `keeplast=t` option to display the first non-match event.

You can also specify a `limit='X'` value to limit the return set size (but you can't use an integer and the limit options together). If both a numeric limit and an eval expression are used, the smaller of the two constraints applies:

```
index = weblogs_90d_eidx sourcetype=access* | head   // first 10 events
... | head limit=20                            // first 20 events
... | head keeplast=t (status <= 200)          // all events until eval
(status <= 200) is false
... | head limit=20 keeplast=t (status <= 200) // first 20 events or until
the eval is false

... | tail 5   // last 5 events in reverse order
```

The `tail` command returns the last N number of results (default = 10), starting at the end of the result set.

Top/rare

The `top` command displays the X most common values for the fields in the given field list, and calculates a count and a percentage for the frequency that the values occur. The default is the top/rare 10 events, but you can specify the count. You can also group the results by field by specifying `<by-clause>`:

```
index = weblogs_90d_eidx sourcetype=access_combined | top clientip
...| top 5 uri, uri_path by clientip
...| rare 3 status
```

The `rare` command returns the least-common values for the specified field – the opposite of `top`.

Visualizing search results

Now that your search results are formatted, you are ready to display the data as a table or a chart.

Table/fields

Splunk can display the fields contained in each event in table format by piping the events to the `table` command along with a list of the fields to display, in the order in which they are to be arranged from left to right. The resultant table will contain one row per event and a column for each specified field. Splunk will automatically switch to the **Statistics** tab to display the resulting table.

Displaying a table of all of the fields in an event is not usually desirable; you can use the `table` command to specify the fields that the table is to include, and to specify the order in which the fields are displayed. The following examples show how to use the `table` command; the first example includes using an asterisk to create a table that displays all the fields in each event, which can make it easier to view and select the fields of interest. You can then add those fields behind the `table` command and eliminate the asterisk when you're happy with the results.

When you create a search that includes a transforming command such as `stats`, the results are by necessity presented as a table containing the output fields specified in the `stats` command; you can pipe the output of the `stats` command to the `table` command to exclude some of these fields or change the display order.

The `fields` command tells Splunk which fields to keep or remove from each event; this can be used with sub-search or `join` commands to return just the fields of interest. Note that, by default, Splunk includes the `_time` and `_raw` fields in results when you use the fields command. You can remove a field (including `_time` and _raw) from the results by using a *– and a space* in front of a field name or list of names. The use of commas between field names is optional. Here are some examples:

```
displays all fields in a table
index=weblogs_90d_eidx sourcetype=access_combined | table *
displays just the listed fields in a table
index=weblogs_90d_eidx sourcetype=access_combined | table _time clientip,
status, useragent, uri

displays just the listed fields plus _time and _raw in a table
index=weblogs_90d_eidx sourcetype=access_combined | fields clientip status
useragent uri | table *
removes the clientip and _raw fields from search results
index=weblogs_90d_eidx sourcetype=access_combined | fields – clientip _raw
```

Chart/timechart

Along with the `stats` command, the `chart` and `timechart` commands transform one or more specified fields from a series of events into a tabular format (a data series) that can be used to create chart visualizations. A *data series* is a sequence of related data points. These data series can be plotted on a chart; each line in a line chart displays the data points for one series. If these data points reflect values that change over time (and are associated with timestamps), it is known as *time series data* or just a time series.

Chart

Chart is a transforming command that returns your results in a table format. The results can then be used to display the data in a chart, such as a bar, column, line, area, or pie chart. You have to use one of three statistical functions with the `chart` command: `stats-agg-term` (as in the stats command), `sparkline-agg-term` (a **sparkline** aggregation function), or `eval-expression` (as in the eval command). The chart command is similar to the `stats` command, with some differences.

The basic syntax is as follows:

```
... | chart <chart-options> <stats or aggregation function> BY <column-
split>
or
... | chart <stats or agg function> OVER <row-split> BY <column-split>
```

Some examples of the `chart` command include the following:

```
index=weblogs_90d_eidx sourcetype=access_combined | chart limit=3
useother=f count by useragent
index=_internal source=*metrics.log group=*thruput | chart avg(kbps) over
group by host
```

The limit argument reduces the number of columns displayed to the value specified plus an additional OTHER column that captures all the other column values unless you use the `useother=false` option.

The allowable **sparkline** aggregation functions include: `c()` | `count()` | `dc()` | `mean()` | `avg()` | `stdev()` | `stdevp()` | `var()` | `varp()` | `sum()` | `sumsq()` | `min()` | `max()` | `range()`. These and a number of other statistical functions can be used with the chart and timechart commands as well – see the Splunk Quick Reference Guide or Splunk docs for a full list.

Timechart

A **timechart** is a statistical aggregation applied to a field to produce a chart, with time used as the X-axis. You can specify a column split-by field, where each distinct value of the split-by field becomes a series in the chart. If you use an `eval` expression, the split-by clause is required.

Here is an example of using the timechart command (and an example of a timechart visualization is depicted in *Fig 6.11*):

```
index=_internal source=*metrics.log group=per_source_thruput | timechart
span=5m avg(kbps) by host
```

You can use a span option with timechart to control the time increments for each bin of values the statistical function operates over.

Note that you can pipe a properly formatted table to the stats, chart, and timechart commands; if you pipe a table to the timechart command, the `_time` (or other time value) field must be the first field in the table.

There are many options for all of these commands – you can peruse the following Splunk docs and Answers links to get all the details if you're going to be doing a lot of charting:

- `http://docs.splunk.com/Documentation/Splunk/latest/SearchReference/Chart`
- `http://docs.splunk.com/Documentation/Splunk/latest/SearchReference/Timechart`
- `https://answers.splunk.com/answers/32001/difference-between-stats-and-chart.html`
- `http://docs.splunk.com/Documentation/SplunkCloud/latest/SearchReference/Xyseries`

Visualizations in Splunk web

You can select the **Statistics** tab in Splunk Web to view your data series, then click the **Visualization** tab, select a charting option, and optionally any formatting options. An example of a search that uses a line chart is depicted in *Fig 6.11*; we will cover some additional visualization topics in `Chapter 8`, *Splunk Reports, Dashboards, and Alerts*:

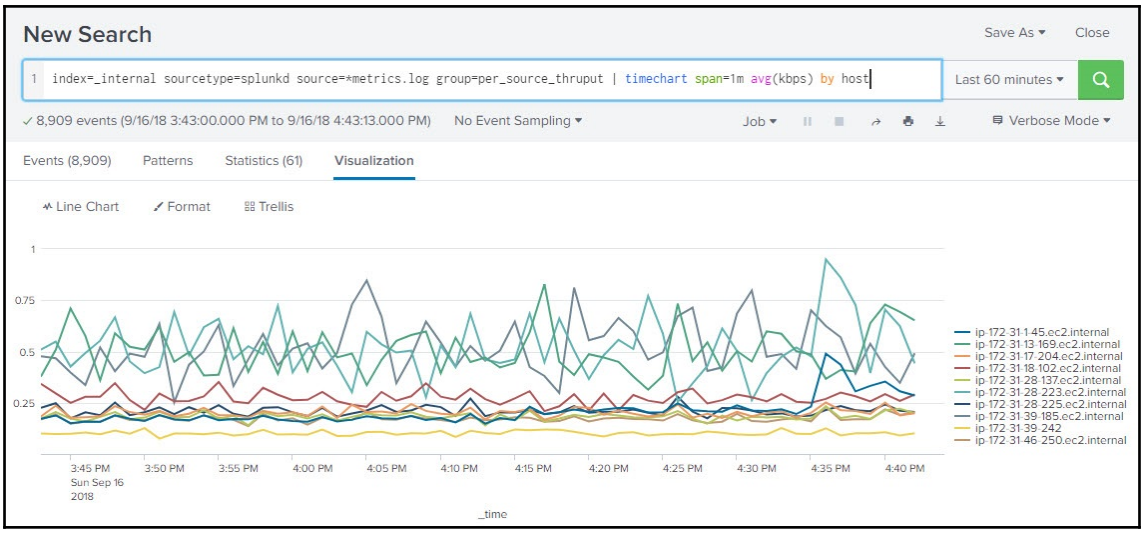


Fig 6.11: Line Chart in Splunk web

Some types of charts lend themselves better to single or multiple numbers of data series; there are also visualizations for displaying a single value, such as a temperature. The following table provides guidelines for picking the best chart type for one or more data series, as well as listing the single-value chart options:

Chart type	Use for a single data series?	Optimized for multiples series?	Notes
Pie	Yes	No	Single series only
Single Value, Radial, Filler, Marker Gauge	Yes	No	Single values only
Bar or Column	Yes	Yes	Can be stacked for multiple series

Line/Area	Yes	Yes	Most common options for multiple data series
Scatter	No	Yes	Works best with two data series
Bubble	No	Yes	Works best with three data series

Advanced search commands

In some situations, you may need to utilize one of the more complex commands, such as `sub-searches`, `join`, or `transaction`. These commands allow you to handle dynamic events and values that would otherwise be difficult or impossible to do within a single search.

Subsearches

A **subsearch** is a search that runs within a primary, or outer, search, and is intended to return results to the primary/outer search to provide data that the primary search needs.

When a search contains a subsearch, the subsearch is run first. Subsearches must be enclosed in square brackets in the primary search, and the first term must be an event-generating command such as `search`, `eventcount`, or `tstats` (usually, you'll use search). An example of a simple subsearch that identifies the top user by **clientip** and then lists the uris that client visited is given here; note that the table command is used to limit the results passed back to the calling search to just the **clientip** field:

```
index=weblogs_90d_eidx sourcetype=access_combined status=200
[search index=weblogs_90d_eidx sourcetype=access_combined status=200 | top
1 clientip | table clientip]
| stats count by clientip, uri
```

Time ranges selected from the time-range picker apply to both the main search and subsearches, but you can use the earliest/latest options within either layer to control the search timeframe.

You can use more than one subsearch in a search; if they are nested, the innermost subsearch is run first, then the next innermost, and so on. If they are sequential, the subsearch closest to the beginning of the search (the first subsearch) is run first, then the next, and so on.

By default, subsearches return a maximum of 10,000 results and have a maximum runtime of 60 seconds. You can change these defaults in `limits.conf`.

Join

You can use the `join` command to combine the results of a subsearch with the results of a main search based on one or more fields that are common within each result set; this allows you to combine search results from more than one source. The field names that are common for the main and subsearch must match in terms of both name and case; you can rename a field in a subsearch to exactly match the name in the main search if needed to make this work.

The `join` command uses a SQL-like join `type`; the `type` specified in the SPL command dictates how the events in the main search that do not match any events in the subsearch are treated. The results of a left (that is, an outer) join include *all* of the events in the main search and only those events in the subsearch that have field values matching the same field values in the main search. With an inner join, the result set only includes events in the main search plus events in the subsearch that have matching field values. The types of joins are depicted in *Fig 6.12*:

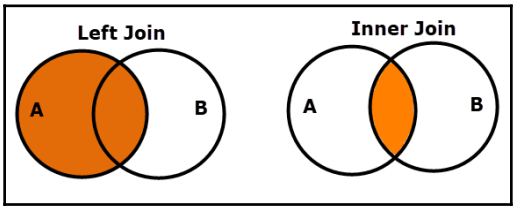


Fig 6.12: Types of joins

Here is the basic syntax and a simple example of a join command:

```
... | join type=[inner | left | outer] [field-list] [<subsearch>]

example:
...
| join type=left product_num
[search index=vendor_logs
| rename prod_id AS product_num
| table product_num price quantity]
...
```

The field values in the table in the subsearch (`product_num`, `price`, `quantity`) are returned and added to each matching event in the main search. You can then continue working with each main search event as if it had always had all of the combined events.

Transaction

The `transaction` command is used to find and group together related events that meet various criteria. Here are some of the things you can use the transaction command to do:

- Group events together using a field value, such as an ID or IP address.
- Group events that begin and end with specific field values.
- Break up groups of events that span longer than a given duration. For example, if a transaction does not explicitly end with a message, you can specify a maximum span of time after the start of a transaction to mark the end of the group of events for that transaction.
- To display raw event data for grouped events.

Some of the most common transaction arguments include:

- `<field-list>`: Field name(s) used to group events into transactions
- `startswith=<string>`: If found, marks the start of a transaction
- `endswith=<string>`: If found, marks the end of a transaction
- `maxspan = <int>[s|m|h|d]`: Maximum length of time a transaction can span
- `maxpause = <int>[s|m|h|d]`: Maximum pause between events within one transaction
- `maxevents = <int>`: Maximum number of events in a transaction

The transaction command adds two fields to the results: duration, which shows the amount of time between the first and last events in the transaction, and `eventcount`, which shows the number of events in the transaction.

An example of using the transaction command to group events together for a user interaction with a website is shown in the following screenshot; this displays the various files and images fetched for this interaction, which included 12 events over a 3 ms timeframe. The search string used was as follows:

```
index=weblogs_90d_eidx sourcetype=access_combined
| transaction clientip maxspan=30s maxpause=5s
| where eventcount > 3
| table _time clientip eventcount duration uri
```

Here are the results of the preceding transaction command:

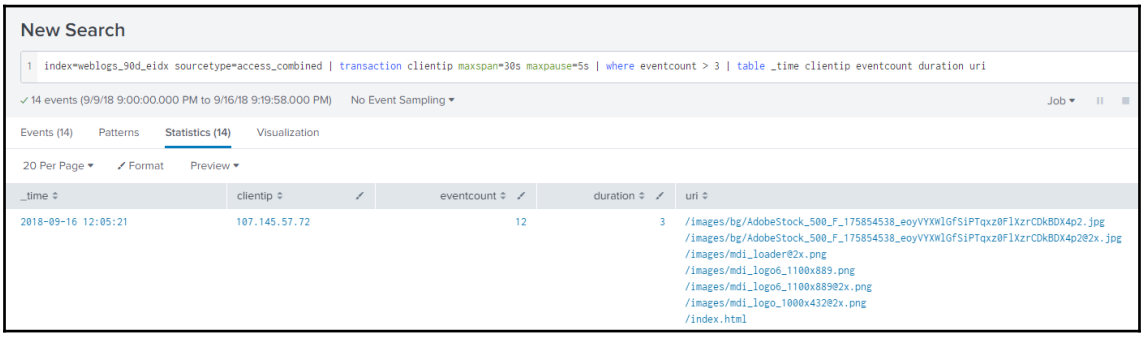


Fig 6.13: Transaction command results

There are a lot of options for the subsearch, join, and transaction commands; you can get all the details in the Splunk docs:

- `http://docs.splunk.com/Documentation/SplunkCloud/latest/SearchTutorial/Useasubsearch`
- `http://docs.splunk.com/Documentation/Splunk/latest/SearchReference/Join`
- `http://docs.splunk.com/Documentation/Splunk/latest/Search/Abouttransactions`

Streaming versus transforming commands

Splunk defines a number of general types of search commands; the most-commonly discussed are:

- Distributable and centralized streaming
- Non-streaming
- Transforming
- Generating

A `streaming` command performs some type of operation on each event as it is returned by a search, without regard to the other events. An example is the `eval` command, which can create a new field from one or more existing fields in each event, and add that new field to each event:

```
... | eval tempf = (tempc * 9/5 + 32)
```

A `non-streaming` command requires that all the events are returned from the indexers before an operation is performed on the entire set of events. An example is the `sort` command: all the events must be returned by the indexers so the search head can do a final sort on the results.

The key concept about streaming commands you should consider is that when the command completes, you still have a set of events with all the fields intact upon which you can execute other commands; this is not the case with transforming commands.

A `transforming` command converts the specified fields from a set of events into a data table consisting of rows and columns, just as in a spreadsheet. Transforming commands include the stats, chart, timechart, top, and the other commands that require a dataset to work with. After a transforming command, you no longer have events to work with, just the data fields – so these commands have to be run *after* you have collected and filtered the desired events, added new fields, and are ready to format the data into a table or visualization.

Note that the difference between a **distributable** versus **centralized** streaming command is that a distributable command may be run on the indexer(s) and can be applied to subsets of the indexed data in a parallel fashion, thus improving performance. A centralized streaming command applies a transformation to each event in the result set, but only works on the search head since all the results must be returned before the command can be applied.

Generating commands generate events without performing an operation or transformation on them—basically, delivering unaltered events. These events can come from an index using the implied or explicit search command, or from other sources by using the other generating commands, which include `metadata`, `inputcsv`, `inputlookup`, `dbinspect`, `datamodel`, `pivot`, and `tstats`.

When you are just starting to learn Splunk search commands, you don't need to worry too much about what type each command is, but as you begin to consider search performance, it is helpful to understand and consider the implications of each type. You can read more about the different types of search commands, as well as how to improve the performance of your searches, at the following links:

- `http://docs.splunk.com/Documentation/Splunk/latest/Search/Typesofcommands`
- `http://docs.splunk.com/Documentation/Splunk/latest/SearchReference/Commandsbytype`
- `http://docs.splunk.com/Documentation/Splunk/latest/Search/Writebettersearches`

Optimizing searches

Because search is the biggest and most important function provided by Splunk, it is prudent to pay attention to the processing costs of the various search commands and the impact on performance.

Optimizing search jobs

As mentioned previously, there are a number of factors that should be considered when creating search commands to optimize the speed and efficiency of our searches; here are the most significant ones:

- Limit the amount of data that needs to be pulled from the indexers to a minimum by careful use of index, sourcetype, and other search filters in the initial search parameters.
- Limit the time range for searches to the smallest window necessary to get the desired result sets. When you are creating your initial search strings, start with a small window and expand as necessary when you have the search working as desired.

- Search as specifically as possible, using the most efficient filters; for example, use *fatal error* instead of **error**, and avoid the use of `NOT <string>` unless absolutely necessary.
- Use post-processing searches in dashboards (we'll cover this in `Chapter 8`, *Splunk Reports, Dashboards, and Alerts*) to avoid running multiple searches for the same data.
- Become familiar with the Job inspector, and integrate it into fine-tuning your more complex search strings.

Job inspector

Every search performed in Splunk results in a search job being created, and each job has a Search ID, which is referenced as a SID. You can inspect the performance of every search using the Job inspector. After running a search, you can click the **Job** drop-down menu and select **Inspect Job** to view a new window that displays the search performance metrics associated with every search. There are two major sections to every job inspection: **Execution costs** and **Search job properties**.

The **Execution costs** displays each execution-processing component of a search job, how many invocations of each component were incurred, the input and output event counts, and, most importantly, the **Duration** in seconds that each component consumed. This can be inspected to determine whether a given search string was operating efficiently, and, with a little work, to compare the overall processing costs of the various ways you might execute a search to realize the desired results.

The other section of the job inspector is a clickable drop-down list labeled **Search job properties**, which displays a great deal of information about the specifics of the search job, including which indexers were involved, the selected parameter status of the job, and how the search string was parsed and delivered/executed in the various Splunk components.

An example of a job inspector window for a fairly simple search, `index= weblogs_90d_eidx sourcetype=access_combined | stats count by clientip`, is depicted in the following screenshot; the search returned **149** results from **545** events in **2.031 seconds**:

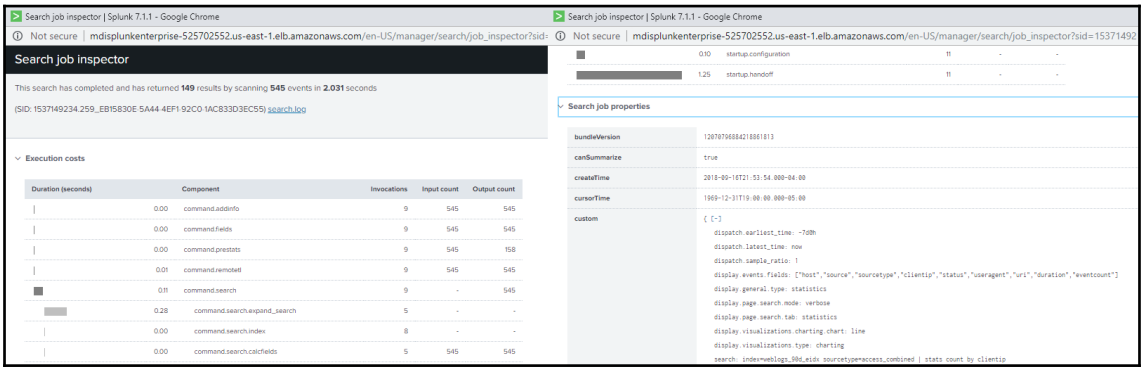


Fig 6.14: Job inspection metrics

Summary

This wraps up the chapter on searching with Splunk. We've covered the Splunk Web interface, how to create basic Splunk searches, and how to create filters to reduce the result set to just the events we're interested in. We then discussed various search commands that can be used to create new calculated fields from the existing fields in the events, commands to manipulate and format the datasets, and how to visualize the results in tables and charts. Finally, we reviewed how to inspect and optimize the performance of our search commands. In the next chapter, we'll cover Splunk knowledge objects to supplement, enhance, and provide even more value to our data.

7
Splunk Knowledge Objects

In this chapter, we cover the various ways you can powerfully enhance and enrich machine data with user-defined fields and datasets to help harness that information in a smarter and more focused way. Event types, tags, and aliases allow you to classify and normalize similar events; field extractions create fields from otherwise-unlabeled segments of an event. Lookups enhance your data with additional information, such as the meaning of HTTP status codes. Data models are pre-prepared representations of one or more datasets created to drive pivot tables and allow business users to create complex reports and visualizations without having to use the Search Processing Language (SPL). All of these enhancements create what are known as knowledge objects, and these capabilities help make Splunk a much more useful and valuable business analytics tool, so you'll want to know how it all works.

The specific topics covered in this chapter include the following:

- Fields and how to perform field extractions to create new fields from events
- How to use event types, tags, and aliases to group your data types
- Using macros to combine filters and commands
- How to configure and use lookups to enhance your data
- How to configure and use data models and pivot tables

Creating and managing Splunk knowledge objects requires that you become more familiar with the data you will be working with; you will need to work closely with your application and other data source teams, and/or help subject matter experts in those teams to understand how they can assist with developing and managing these objects with Splunk.

Let's get started!

Field extractions

Regardless of how you are using the data in Splunk to solve business problems, you'll be working with the values in various fields within each event. Splunk extracts event fields in three ways:

- **Index-time**: Fields are extracted and stored when events are indexed
- **Search-time**: Fields are automatically extracted from key-value pairs in each event
- **Explicitly extracted fields**: Fields are created from specified locations within each event at search time

And as discussed in `Chapter 6`, *Searching with Splunk,* you can use Splunk commands, such as eval and stats, to create new fields from data in existing fields.

Index-time field extractions

Splunk extracts and stores several default metadata fields, such as timestamp (`_time`), host, source, and `sourcetype`, for each event at index time. Splunk can also extract custom fields at index time that you have explicitly configured for index-time extraction; this is accomplished with appropriate entries in `props.conf` and `transforms.conf` files that get distributed to the search peers (indexers), and a `fields.conf` file that is distributed to the search heads. There are some significant performance implications and other considerations related to using index-time field extractions, so you should use search-time field extractions in most cases. For details on the appropriate application and configuration of index-time field extractions, check out the Splunk docs: `http://docs.splunk.com/Documentation/Splunk/latest/Data/Configureindex-timefieldextraction`.

Search-time field extractions

Most of the events you'll be working with came from logs or other sources that stored their data either with an ordered field placement, such as an Apache or IIS log file, or from structured sources with key-value identified fields, such as `"field"="value"` or `"field":"value"` from JSON sources, or tagged fields from XML sources.

At search time, Splunk automatically identifies and extracts the first 50 fields that it finds in the event data that match obvious key-value pairs; these extracted fields will appear in the fields sidebar on the left-hand side of the events table if you have selected Smart Mode or Verbose Mode in the field discovery mode selection dropdown.

You can modify the 50-field limit by editing the `[kv]` stanza in `limits.conf`, but if you're only going to use a few of the fields in these large events, this may impose unnecessary processing requirements. Splunk can extract any field explicitly mentioned in the filtering section of a search string that it might have otherwise found through automatic extraction but is not among the first 50 fields, for example:

```
index=someapplogs sourcetype=type90  the62ndfield = *  the77thfield = * |
...
```

One caveat of this approach is that if there are events in which the fields specified in the search do not exist, or do not have a value, *these events won't show up in the search results at all*. If there are other fields of interest in each event and you were expecting to get a full set of events regardless of whether every specified field existed or had a value, this can cause confusing results.

The behavior described here, where the use of `<fieldname> = *` in the filtering section of a search only returns events for which the specified field exists and has a value, can be a useful filtering technique in certain circumstances where you only want events that *do* match this criteria.

Another method of ensuring that you extract specific fields at search time that may lie beyond the first-50-field limit is to use `rex` to explicitly extract a labeled field, even if you assign the same field name back to the extracted data; this has the advantage of not eliminating an event from the search results if this field is missing or doesn't have a value. In the following example, there is a standard start time field in each event that only has an entry if the event marks the beginning of a series of related events. Note that we're extracting the value from the `starttime` field and assigning it to a new field with the same name, effectively leaving the field unchanged but making sure we extract it at search time, regardless of its position in the result events:

```
... | rex field=_raw "(starttime=\"(?<starttime>[0-9-: ]+)\")"
```

Using the extract fields interface

Sometimes, you will want to extract specific parts of a larger field or other blob of data within an event as an individual field. You can use the field extractor wizard to create new fields from event data using two extraction methods: **regular expression** for unstructured event data, and **delimiters** for structured data where the fields in each event are separated by a common delimiter, such as a comma, space, or tab, in cases where Splunk doesn't know how to label these fields.

To get started, enter a search string and retrieve a sample of the events you want to perform a field extraction on. You can then start the field extractor wizard from four locations:

- Click **Extract New Fields** at the bottom of the fields sidebar.
- Click **All Fields** (top of fields sidebar) | **Extract New Fields** (top-right of **Select Fields** form).
- Click the down-arrow to the left of an event and select **Event Actions** | **Extract Fields**.
- Click (menu bar) **Settings** | **Fields** | **Field Extractions** | and open **Field extractor**.

The steps involved in working with the field extractor wizard are: **Select Sample** | **Select Method** | **Select Fields** | **Validate** | **Save**. Let's walk through these steps in detail:

- **Select Sample**: The initial sequence of actions will vary depending on where you entered the field extractor, and whether you included a `sourcetype` filter in your search; you may be prompted to select a `sourcetype` from a dropdown, and from a list of events, to select a sample event to work with. Then, click **Next**.
- **Select Method**: You will be presented with the option of using **Regular Expressions** or **Delimiters**. If your event fields are reliably separated by spaces or other delimiters, select **Delimiters**; otherwise, click **Regular Expressions** (the typical choice) and click **Next**.
- **Select Fields**: If you're using a RegEx extraction, highlight the value in the sample event that you want to extract. You will be presented with a **Field Name** field in which you can enter the name of the field you want to extract, then click **Add Extraction**:

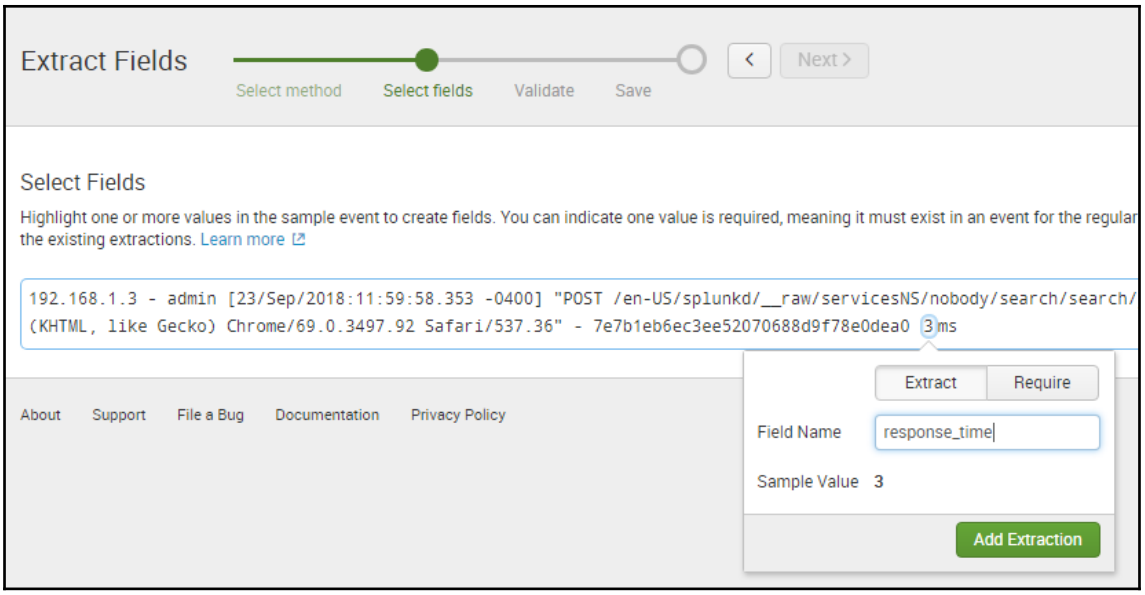


Fig 7.1: Extracting a field

At this point, you will be presented with a preview where you can see (in the **Events** tab) the sample of events with each of the entries for the new field highlighted. Clicking the **<your field name>** tab displays a statistical summary of the number and distribution of values for the new field. Further up, in the **Select fields** pane, you can also click **Show Regular Expression** to view, and optionally edit, the RegEx Splunk has created for this field. You'll notice that this is a very positional-dependent RegEx; you may want to edit it to just include a reliable prefix and/or suffix string. In this example, I clicked **Show Regular Expression** and edited the entry to reflect the fact that the response time would always have `ms` as a suffix to the Regex Splunk had created (and deleted everything else), and clicked **Save**:

```
...\s+(?P<response_time>\d+ms)
```

When you are satisfied with the extraction criteria, click **Next**.

- **Validate**: If you are using regular expression extraction, this screen is similar to the previous validation view, but in this view you can select a **Matches and Non-Matches** button to see how well your field extraction matches across a selectable number of events. If you had selected the **Delimiters** method in a previous step, you won't see this step. When you are satisfied with the results, click **Next**, or click the **<** button to go back and adjust your extraction settings.
- **Save**: After you click **Save**, you will be given the option to rename the extraction and set the permissions for this extraction. If this extraction is useful across a wide range of users and apps, you can click **All apps** and click the boxes for the user roles that should be able to view and/or edit the extraction, then click **Finish.**

You can now click **Settings** | **Fields** | **Field extractions** and view the list of all the field extractions including the one you just created. You'll note that the field extraction names are a combination of the `sourcetype` and the extraction name:

Fig 7.2: Field extractions list

A good starting point for learning more about all the ways to extract fields from various data formats is this Splunk doc: `http://docs.splunk.com/Documentation/Splunk/latest/Search/Extractfieldswithsearchcommands`.

Other knowledge objects

The remaining topics in this chapter, along with fields, are collectively called knowledge objects, which are the user-defined entities that enrich the existing data in Splunk. I will provide a brief description and example of some of the most common knowledge objects in this chapter; you can learn more about all of the knowledge objects in the Knowledge Manager Manual: `http://docs.splunk.com/Documentation/Splunk/latest/Knowledge/WhatisSplunkknowledge`.

Event types – tags – aliases

Splunk event types, tags, and aliases are used together or separately to associate search criteria and various field values to a common name. This can simplify search strings and allow for one-location modification of search criteria without having to change these to numerous saved searches, reports, dashboards, alerts, and so on.

Event type

An **event type** is a user-defined field that represents a category of events that can all be matched by the same search string. When you run a search that returns a useful set of events, you can save that search string—or the reusable parts of it—as an event type, and use that event type in future searches. This is particularly useful for leveraging event types in things such as saved searches, reports, and dashboards if the event type contains one or more indexes, sources, or source types that can be changed in one place (the event type) instead of updating all the tools that use this same search criteria.

You can view the event types that Splunk provides out of the box by clicking **Settings** | **Event types**; one example you'll see is an event type named `splunkd-log` with a search string that specifies an index and a source in Linux and Windows path formats:

```
Name            Search string
splunkd-log     index=_internal source=*/splunkd.log OR
source=*\\splunkd.log
```

You can click **New** to create an event type in this view, or you can create an event type after running a search by clicking **Save as** and selecting **event type**. In the form which appears, complete the fields to give the event type a name, assign optional tags, and select a color that gets displayed next to events of this event type, as well as a priority from the drop-down selector which Splunk will use if an event matches more than one event type.

The definition for event types that you create are stored in an `eventtypes.conf` file in `$SPLUNK_HOME/etc/users/<your-username>/<app>/local/`, where `<app>` is your current app context, such as search (the app you were in when you created the event type). If you set the permissions on the event type to make it available to all users (either in the app, or globally to all apps), Splunk will move the `eventtypes.conf` file to `$SPLUNK_HOME/etc/apps/<app>/local/`.

You can use an event type in a search as in the following example, in lieu of explicitly specifying the index and sources or other criteria represented in the event type definition:

```
eventtype=splunkd-log | ...
```

Tags

A **tag** is a knowledge object that you can assign to specific field and value combinations, such as IP addresses or ID numbers. For example, you can tag a server with a specific IP address or hostname with its function, for example, tagging a host field that contains the value of `10.15.32.15` as the cluster master in your Splunk deployment. If you had events that reported a host by its IP address, and other events that reported the name in the host field, you can use a tag for the host field that associates the IP address with the hostname so you only had to search for the hostname to get both sets of events.

You can create or edit a tag by clicking **Settings** | **Tags** | **List by field-pair value** (or one of the other two selections) and clicking either the tag you want to edit or **New**.

You can also create a tag by clicking the down arrow to the left of an event, finding the field in the list you want to tag, clicking the down arrow to the right, and clicking **Edit Tags**. You will be prompted to give the tag a name and **Save**.

You can use a tag in a search filter as you would any field; for example, a tag created to identify a `hostname`, `robotdev`, as `DevSplunkServer` could be used in a search as in the following example. Splunk will display both labels in the host field for the returned events:

```
Search:
index=_internal tag=DevSplunkServer
Host field as displayed in the events:
host = robotdev DevSplunkServer
```

Field aliases

A **field alias** is an alternate name that you can assign to a field, allowing you to use that alias to search for events that contain that field. A field can have multiple aliases, but a single alias can only apply to one field. An alias does not replace or remove the original field name. For example, you may have a field called `hosttype` in one data source, and a field called `function` in another data source that means the same thing. You can assign a `hosttype` alias to the `function` field and only have to search for `hosttype` in your search string to capture all the desired events.

You can create an alias by selecting **Settings** | **Fields** | **Field aliases** | **New**. You will be prompted to give the alias a name, associate it with a `sourcetype`, source, or host by name, and provide the `field=value` alias values. You can then use that alias in searches.

For example, if you create an alias that associates the host field with a `server` alias, you can use the term `server` as a search field in your search instead of the host field if you had events that intertwined the two terms:

```
index=_internal server=robotdev
```

Lookups

A **lookup** is a powerful feature that provides data enrichment by mapping a select value in an event to a field from another data source, such as a CSV file, and adding the matched results to the original event. For example, you can use a lookup to match an HTTP status code value and return a new field containing a status description. Data sources for lookups can include search results, CSV files, `geospatial .kmz` files, KVStore collections, data obtained by running a script, and external database connections (using the DB Connect app).

As a simple but useful example, you can create a comma-separated value file containing HTTP status codes and their short descriptions, which includes a header line at the top; save the file as `HTTPStatusCodes.csv`:

```
status,description
100,Continue
...
200,OK
...
304,Not Modified
...
404,Not Found
...
500,Internal Server Error
...
```

Upload the CSV file by clicking **Settings** | **Lookups** | **Lookup table files**. From this page, you can change the app you want the lookup file to belong to (such as Search & Reporting), then click **New Lookup Table File**. Click **Choose File**, navigate and select your CSV file, enter the desired **Destination filename** (the destination filename can be the same as the source filename), and click **Save**. The lookup file will be imported and saved in the `$SPLUNK_HOME/etc/users/<user>/search/lookups/` folder on the search head you're using, and if this is in a search head cluster, the lookup file will be distributed to the other search heads by the **Captain**. When you are returned to the list of **Lookup** table files, you'll want to click **Permissions** and select **This app only** or **All apps**, and who can read/write (I suggest everyone can read, admin and power can write).

This will result in the lookup file being moved to `$SPLUNK_HOME/etc/apps/search/lookups` (if you were in the search app context). I mention these file locations as you change these settings because it helps you understand how and where Splunk manages and stores knowledge objects as you change app and user ownership.

To use the lookup in a search, pipe the events to the lookup command, and specify the lookup filename and the field in the events (status, in this case) that you want to map to a field in the lookup file (in this case, the lookup field is also called status). Splunk will return the field(s) that correspond to the lookup field and add them as field(s) to each event. You can select the field(s) to output (in this case, we only have `description`, but there could be multiple fields in the lookup file) and even rename the output field with the `OUTPUT <field>` as `<new field name>` option, as seen in the following example `[w]`:

```
index=weblogs_90d_eidx sourcetype=access_combined
| lookup HTTPStatusCodes.csv status OUTPUT description as "HTTP Status
Code"
| table _time clientip status "HTTP Status Code"
```

You can see the results of using the lookup in the following screenshot; you now have the status and a code description:

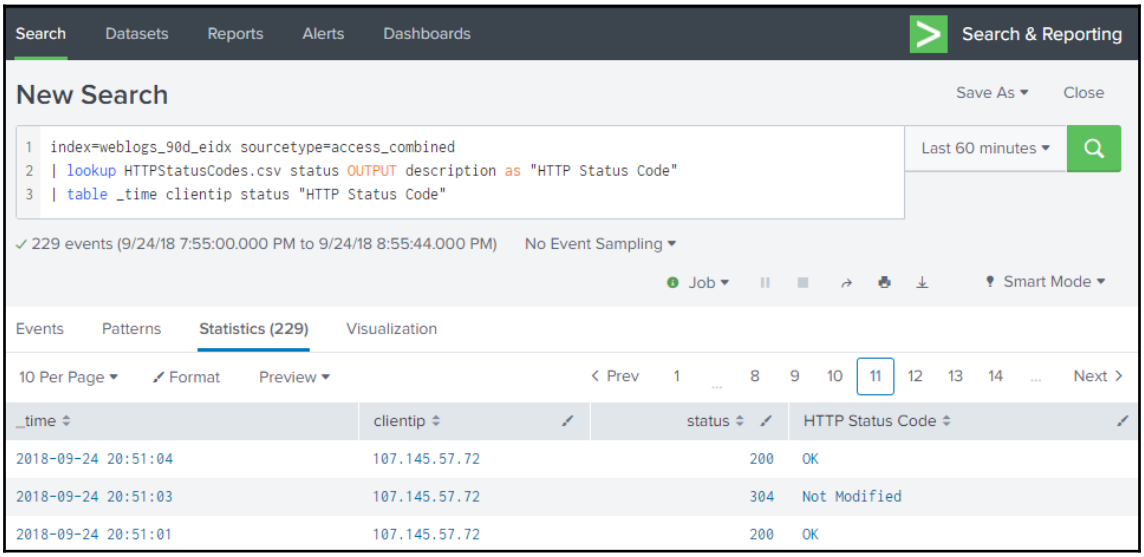


Fig 7.3: Using a lookup file

You can also click **Settings** | **Lookups** | **Lookup definitions** | **New Lookup Definition** to create a lookup definition with a name you can use in the search instead of a filename; this would allow you to introduce new filenames and not break any reports or dashboards that are referencing the definition.

Finally, you can use the **Automatic lookups** feature to map lookups to a specific `sourcetype`, source, or host, and define input and output fields; after configuring an automatic lookup (remember to set **Permissions**), you don't have to use the lookup command to obtain the looked-up fields; you just reference those fields as you would any other in the event.

You can learn more about all the ways to implement lookups in the Splunk docs:

`http://docs.splunk.com/Documentation/Splunk/latest/SearchReference/Lookup`

Macros

Search macros are reusable chunks of SPL that you can insert into any part of your searches using the `<macro_name>` format (using backticks). Macros can be as simple as grouping several commonly used search commands together, or used to create complex calculation macros that accept arguments. You can peruse the macros that come with Splunk by clicking **Settings** | **Advanced search** | **Search macros** (select all from the app dropdown) to better understand how they're used.

You can then create your own macros by clicking **New Search Macro** and completing the form. For example, you could create a new macro called `weblogs` whose definition is `index=weblogs_90d_eidx sourcetype=access_combined`. After saving the macro, click **Permissions** and set the app context (search, for example) and who can use and edit the macro (everyone and admin/power, respectively). To use this macro in a search, just replace the part of the search string the macro contains:

```
`weblogs` | stats count by clientip, status, uri
```

If the macro starts with a command that normally is preceded by a pipe symbol, you'll need to put the pipe symbol in front of the macro, not inside it. Finally, if you build a macro that takes arguments, you give the argument a name when you create it and insert that argument inside the macro with `$<argument_name>$` tokens. If we added an argument to the preceding macro to provide an additional filter string, you would create a new macro named `weblogs(1)` (indicating there is one argument) with the following search string:

```
index=weblogs_90d_eidx sourcetype=access_combined $filter$
In use you have to use quotes around the filter argument string:
`weblogs("status=200")` | stats count by clientip, uri
```

For more on configuring macros, see the Splunk documentations:

`http://docs.splunk.com/Documentation/Splunk/latest/Knowledge/Definesearchmacros`

Datasets and data models

Understanding and working with datasets and data models, and the terms commonly used to describe their features, may be hard to wrap your head around unless you have data modeling experience. I'll provide a number of conceptual statements and explanations in this section that should enable you to put the pieces together in such a way that you get that *Aha!* moment and it all starts to make sense. From that point, only working with a few data models will take you to proficiency.

Datasets

A Splunk dataset is a collection of data from lookups, searches, and so on—a set of data. You can view the datasets that come with Splunk by clicking the **Datasets** menu option in most apps, such as **Search**; this lists all the currently-configured dataset types (lookups, data models, and so on). You can click on a lookup table, such as **geo_attr_us_states.csv**, to get a tabular view of this file's contents, or on a data model dataset to see a list of events with the selected fields associated with that model and their values from a search.

Data models

Data models are a hierarchy of related datasets. Data models are comparable to relational database schemas, if that is a familiar concept to you; they are a structured collection of related tables. Data models are a way to enable less technical users to create reports and dashboards (using **Pivots,** which we'll cover in the next section) without having to write the searches that generate the data that feeds them. The domain knowledge needed to build the searches that populate the datasets is encoded into the data model by people familiar with both the data and Splunk searches, or by the combined efforts of people in both camps. Let's walk through a couple of examples.

From the Search & Reporting app, click **Settings** | **Data models**. You should see two of Splunk's provided data models; by clicking **Splunk's Internal Server Logs - SAMPLE**, for example, you can peruse this model to get a feel for how they are built and displayed. In the left-hand **Datasets** pane, the datasets are listed in a hierarchical fashion. Click **Splunk Server** (this is the root object) and note the contents of the right-hand pane; it consists of a section called **Constraints**, which in this case is a SPL search string with an index and initial filters:

```
index=_internal source=*scheduler.log* OR source=*metrics.log* OR
source=*splunkd.log* OR source=*license_usage.log* OR
source=*splunkd_access.log*
```

Under the **Constraints** section is the unlabeled **Attributes** (fields) section, which contains **Inherited** and **Extracted** event fields. Inherited fields are those that are common to all events: host, source, `sourcetype`; extracted fields are those fields that Splunk automatically extracts from the events. Take a look through these sections, then click **Performance and System Data** (this is a child object) in the left pane. Note that an additional filter has been added to the **Constraints** section to isolate just the performance events:

```
source=*metrics.log*
```

Finally, click **Thruput** and note an additional **group=thruput** filter has been added to the constraints, and on the right, this constraint has been added to the inherited constraints from the higher-level datasets.

The event fields in the **Attributes** section for this dataset are all auto-extracted fields. If you click **Daily Usage Summary** under the **Licenser** child object and scroll to the bottom of the fields, you will see three new fields in the **CALCULATED** section: **Pool Size (GB)**, **License Used (GB)**, and **stack size (GB)**. These are fields created with an **Eval Expression**; you can click **Edit** on the right to see the formula used to create these fields. These were created by clicking the **Add Field** dropdown, selecting **Eval Expression**, and completing and saving the fields in the form that appears. You'll see that you can also create new fields using a **Lookup**, **Regular Expression**, or **Geo IP** (if there are IPv4 fields in the events to work with).

To serve as an example that should increase your comfort level with data models, we'll build a simple one for working with web access log events; if you have access to some web logs, you can follow this example in your own environment, modifying details as needed.

From the data models page, click **New Data Model**. Give it a name, **Web Logs**, select the **Search & Reporting** app, and click **Create**. Click **Add Dataset | Root Event** in the left pane; give the dataset a name of **Web Access**, and in the **Constraints** field, enter the following (altered for your `index` and `sourcetype`):

```
index=weblogs_90d_eidx sourcetype=access_combined
```

Now, click **Save**. Note the standard **INHERITED** fields appear on the right. Now, click **Add Field | Auto-Extracted**; Splunk will run a quick sample search and populate the form that appears with all the extracted fields it can find. You can select specific fields, or click the box at the top to select all fields, then click **Save**; all of the selected fields are now added to the data model. Now, click **Add Fields | Eval Expression**, name the `status_category` field, enter the following in the **Eval Expression** field, and click **Save**:

```
case(like(status,"1%"), "Informational", like(status,"2%"), "Success",
like(status,"3%"), "Redirect", like(status,"4%"), "Client Error",
like(status,"5%"), "Server Error")
```

Let's add a **child dataset** to the web logs root object. Click **Add Dataset | Child**, give this dataset a name of **Web Errors**, and enter the **Additional Constraints** field as shown here:

```
status >= 400
```

Before you click **Save**, click **Preview** and ensure that events matching the new constraints appear. When you are returned to the data models page, the **Web Errors** dataset will appear under **Web Usage**.

Note that you could have also selected **Root Transaction** or **Root Search** from the **Add Dataset** dropdown; these allow you to create a dataset based on the transaction command, or a straightforward SPL search string, respectively.

Using data models in search

Data models are used in Splunk searches and for **Pivots**, which we'll discuss shortly. You can also use your data model to perform searches by using the `from` command after a pipe at the beginning of the search, and indicating which data model you want to apply. You can specify by using a child dataset in the search by using the dot operator (`.`) and then the child dataset's name:

```
| from datamodel:Web_Logs.Web_Errors
```

You can also launch a search using a data model by finding the data model listing in the **Settings** | **Data models** page and clicking **Explore** | **Investigate in Search**. In the following screenshot, you can see this in use. Only events with a status of 400 or greater are returned, and the fields listed on the left are just those fields selected to be included in the data model/dataset:

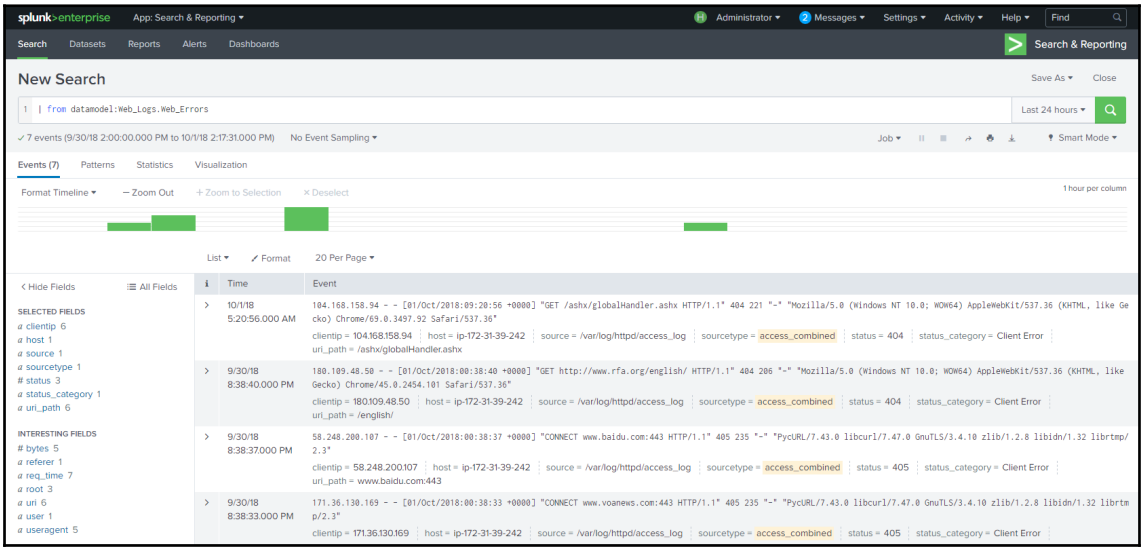


Fig 7.4: Using a Data Model in Search

These were simple examples of using datasets and data models, but data models can become very complex as data from multiple sources is combined.

In case you are curious, data models created in the search app context are saved in a JSON format file located in `$SPLUNK_HOME/etc/apps/search/local/data/models` (in Linux); acceleration settings are in a `datamodels.conf` file located in `.../search/local/`.

Data model acceleration

Data models can be accelerated to vastly improve search performance; selecting this option creates `tsidx` files on the indexers containing the fields you extract in the data model to speed up search results. To configure your data model to be accelerated, start on the **Settings** | **Data Models** page, and click **Edit** | **Edit Acceleration** for that data model. Note that private data models cannot be accelerated; you may have to adjust the **Permissions**. Just click the **Accelerate** checkbox, select a **Summary Range**, and **Save**.

Note that in a distributed environment, accelerated data models are not replicated across indexers by default; you have to configure that in `server.conf` on each indexer. See this document for more details:

```
https://docs.splunk.com/Documentation/Splunk/latest/Indexer/
Clustersandsummaryreplication
```

And finally, you should also consider that accelerated data models exact an indexing performance and storage cost, so they should be used judiciously. You can learn more about data models from these links:

- `http://docs.splunk.com/Documentation/Splunk/7.1.3/Knowledge/Aboutdatamodels`
- `https://conf.splunk.com/session/2014/conf2014_DavidClawson_Splunk_WhatsNew.mp4`

Pivot tables

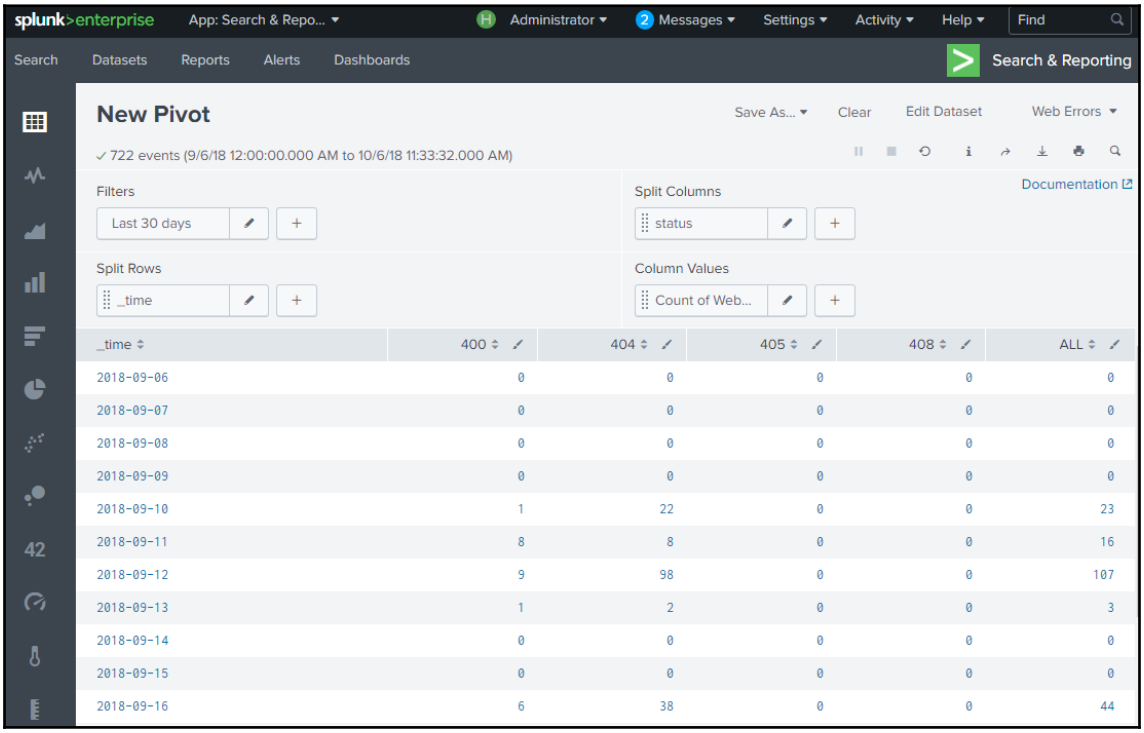
As mentioned, a Pivot (also known as a **pivot table**) is a way for less technical users to work with data models and datasets (which have been prepared by knowledge object and data managers) to investigate data and create business reports and dashboards without having to know Splunk SPL. Describing pivots with words is counterproductive; an example is a much better approach, so let's walk through creating a simple pivot table.

From the **Search & Reporting** app, click **Settings** | **Data models**, and click the **Pivot** link on the right-hand side of the **Web Usage** entry created in the previous section. This will take you to the **Select a Dataset** page; select **Web Errors**. You can also get here by clicking the **Datasets** menu option, filtering/finding the **Web Logs** | **Web Usage** | **Web Errors** dataset from the list, and clicking **Explore** | **Visualize with Pivot** from the right-hand link. You will now be on the **New Pivot** page, where you will see the four selectors that you will use to design your pivot. I'll describe them in the comparative context of a spreadsheet:

- **Filters**: Click to select a time range for events to include in the pivot report, for example, Last 30 days.

- **Split rows**: This becomes the rows in the pivot table. You can click the **+** button, **select _time**, and then leave the **Period** setting at **Auto** or select another time category, such as **minutes**, **hours**, or **days** (we'll select **Days** for this example), and then select whether the rows are organized in ascending or descending order; leave this at **Auto** or the results will be sorted by count, which is not what we want. You could select other row division criteria from the list of available fields, **but _time** is usually the right choice for most reports.
- **Split columns**: This controls the columns of the pivot table. Click the + button, select **status**, and leave all the default settings as they are except the Totals option at the bottom – click Yes to add a total count column; click Add to Table.
- **Column values**: This field is automatically configured to use **Count** of **Web Errors** in this example, which is appropriate, but you could click to select it and add additional field or calculated values if the report or visualization type called for it.

At this point, you have a nice table of the daily counts of web events with a status code of 400 or greater:



_time	400	404	405	408	ALL
2018-09-06	0	0	0	0	0
2018-09-07	0	0	0	0	0
2018-09-08	0	0	0	0	0
2018-09-09	0	0	0	0	0
2018-09-10	1	22	0	0	23
2018-09-11	8	8	0	0	16
2018-09-12	9	98	0	0	107
2018-09-13	1	2	0	0	3
2018-09-14	0	0	0	0	0
2018-09-15	0	0	0	0	0
2018-09-16	6	38	0	0	44

Fig 7.5: Pivot Table: Daily Web Errors by Status Code

Now, do you see the visualization icons in the left-hand sidebar? This is where pivot tables really shine. Your user has quickly and easily created a table of data tailored to their reporting purpose without writing a line of SPL, and they can now click on an icon to create a chart of any type from their data. Clicking the second icon in the preceding example will display a line chart and all of its visualization controls:

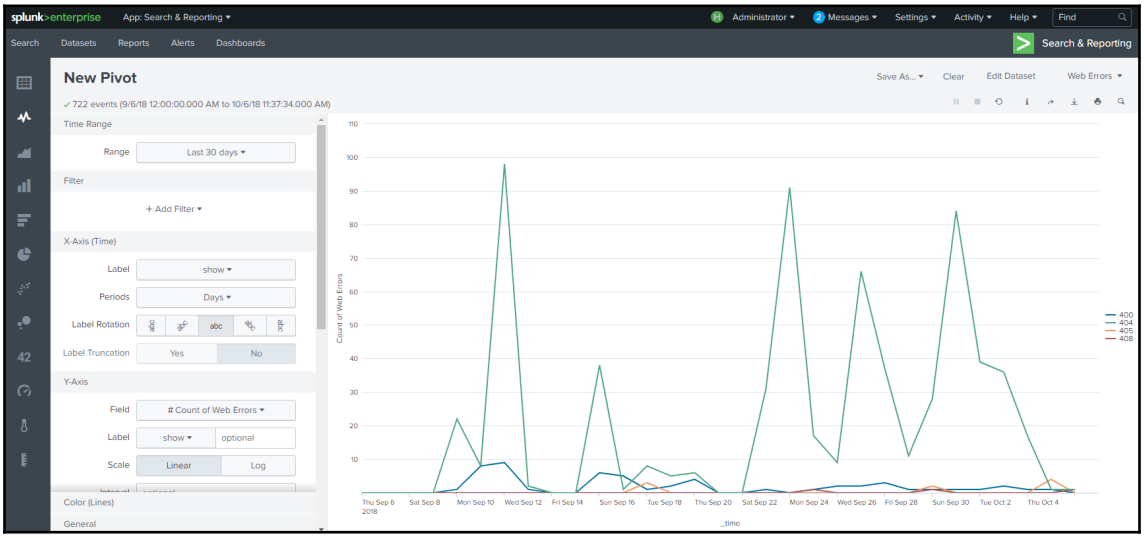


Fig 7.6: Creating a visualization from a pivot table

Once you have the visualization configured to your liking, you can click the **Save As...** dropdown at the top of the page and save your table or visualization as a **Report** or **Dashboard Panel**. If you select **Report**, give the report a name, such as **Daily Web Errors by Status** and an optional description, click **Save**, and you're done!

Splunk's data models, datasets, and pivots are a powerful data inspection and reporting tool in the hands of your business users; you can learn more about pivots in the Splunk docs:

`http://docs.splunk.com/Documentation/Splunk/latest/Pivot/IntroductiontoPivot`

Summary

This chapter provided an introduction to the most common knowledge objects Splunk users can leverage to enhance their data. We learned how to extract and create new fields in events, how to group and/or replace search criteria with event types and macros, how to tag and categorize event fields with tags and aliases, and how to enhance the data in event fields with lookups. Then, we looked at creating datasets and data models to be used in pivot tables so our less technical users can leverage the power of the data provided by Splunk in their reports and dashboards. In the next chapter, we'll cover how to create reports, dashboards, and alerts – see you there!

8
Splunk Reports, Dashboards, and Alerts

This chapter builds on the skills developed in the previous two chapters to help you quickly and easily develop effective reports and dashboards from saved searches that provide at-a-glance status indicators, charts, graphs, tables, and complex visualizations that can be scheduled for delivery by email or other transmission means. You'll also learn how to configure alerts to monitor critical events and let support personnel know when something isn't right—and where to look first to fix it.

The topics covered in this chapter include the following:

- Creating reports and saved searches for scheduled reports
- Using reports to create and populate a Splunk dashboard
- Adding and integrating controls into a dashboard
- Enhancing Splunk dashboards with JavaScript and other visualization options
- Creating and configuring alerts

Introduction

This chapter starts with creating reports from a search, which is the foundation for creating dashboards and alerts as well. When you have created and executed a search using SPL, inspected the results to ensure they include all the data you wanted them to, and selected and configured the visualization options so that the output is meaningful and useful, you have several options for saving the search and search results so they can be shared and viewed by others, and/or run again on an ad-hoc or scheduled basis to support a business function.

The choices for accomplishing this are available by clicking the **Save As...** link to the top-right of Splunk Web and selecting the desired option from the following list:

- Report
- Dashboard panel
- Alert
- Event type

We'll cover each of these options in turn, and, in the process, leverage a useful search that provides an assortment of useful information about your Splunk servers, including the amount of available disk space versus disk usage. Note that in a distributed environment, this search should be run from a cluster master to get information on all of your indexers (in case Splunk Web is disabled on the indexers), and should be run from any search head to get information on your search heads. You can select **Last 15 minutes** in the time-range picker, although this setting doesn't matter since this REST search is a snapshot of the current configuration; you can see the entire search string in the code block below:

```
| rest services/server/status/partitions-space
| eval pct_disk_free=round(available/capacity*100,2),
pct_disk_used=round(100-(available/capacity*100),2)
| eval disk_capGB=round(capacity/1024, 3),
disk_availGB=round(available/1024, 3), disk_usedGB = disk_capGB -
disk_availGB

| join type=left splunk_server
[
| rest /services/server/info
| eval UpSince = strftime(startup_time,"%Y-%m-%d %H:%M:%S")
| eval upSec = (now() - startup_time)
| eval upHrs = round(upSec/3600, 1)
| eval upDays = round(upSec/86400, 1)
| eval memGB = round(physicalMemoryMB/1024,3)
| rename numberOfCores as #cores, numberOfVirtualCores as #virtCores,
version as splunk_version
| table splunk_server UpSince upSec upHrs upDays cluster_label
shcluster_label cpu_arch #cores #virtCores os_name os_version memGB
server_roles splunk_version activeLicenseGroup activeLicenseSubgroup
]

| table splunk_server UpSince upHrs upDays cpu_arch #cores #virtCores memGB
fs_type mount_point disk_capGB disk_usedGB disk_availGB pct_disk_used
pct_disk_free server_roles *cluster_label os_name os_version splunk_version
```

Here is an example of the results obtained from running the preceding search:

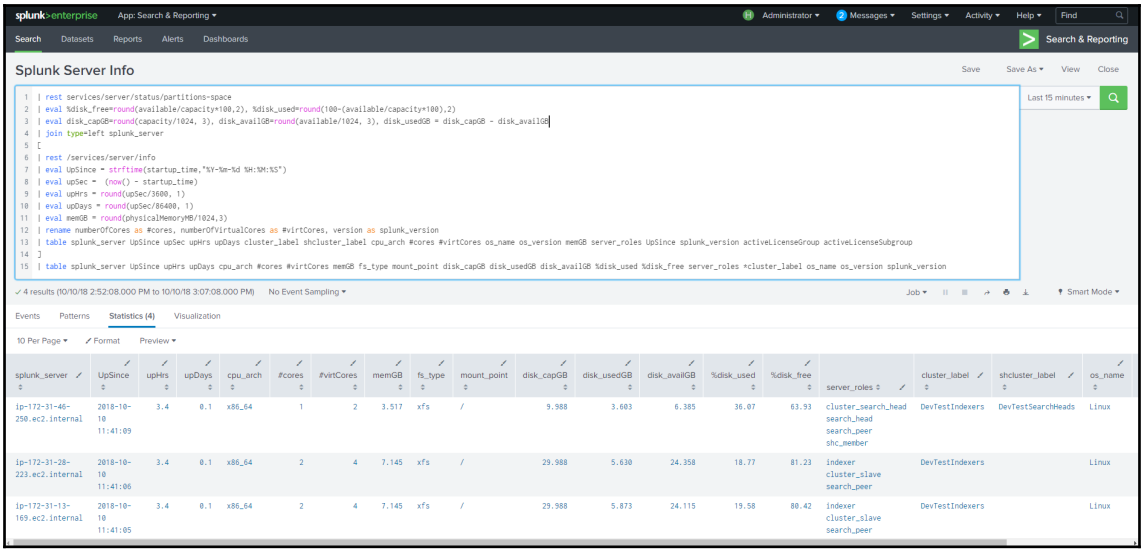


Fig 8.1: Server information search

This search uses REST API calls to obtain disk and server information from the Splunk servers. We'll introduce the REST API properly in `Chapter 10`, *Advanced Splunk*; for now, it is enough for us to know that piping to the REST command (`| rest`) with the specified Splunk API endpoints returns a row for each server containing the fields and values for a variety of Splunk internal metrics. In the preceding search, we obtain information about disk capacity and usage from one endpoint (`/services/server/partitions-space`), and join that with a number of other information fields (from `/services/server/info`) using `splunk_server` as the common field for the join.

You can break the preceding search into smaller parts, remove the table commands to see all the fields that are returned, and then rebuild the table command to list the specific fields and their display order to your liking. When you're ready, let's use this data to create some useful reports, dashboard panels, and an alert.

Creating reports

To save the search in the previous section as a tabular report, follow these steps:

1. Click on the **Save As...** | **Report** link.
2. In the form that opens, give the report a title, such as `Splunk Server Info`, and an optional description. In this case, we will set the option to display the **Time Range Picker** in the report to **No** because the REST calls are an immediate snapshot, but you'll probably want this option displayed for most of your reports.
3. Click **Save.**

You'll be presented with a new form with the options to set **Permissions**, **Schedule** or **Accelerate** the report, or **Embed** it in an external website, as shown in following screenshot:

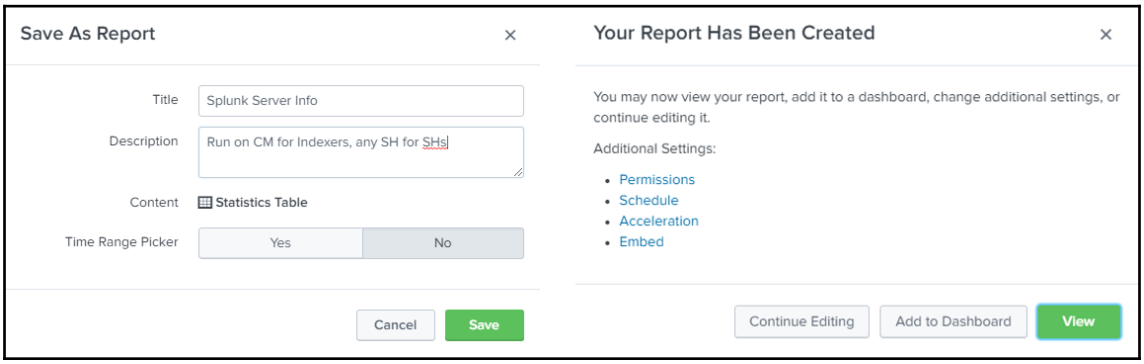


Fig 8.2: Saving a report and additional settings

You can now click **Permissions** and change the default **Display For** setting from **Owner** to **App** (usually the best option) or `All apps`.

The **Run As** setting determines whether the search runs with the permissions of the report **Owner** (the person who defined the search and report) or the permissions of the search `user` (the person who is running the search). Reports run as **Owner** by default, and scheduled reports can *only* run as **Owner**, so you may as well use this setting unless needs dictate otherwise. Finally, you can give read permissions to everyone and write permissions to admin and power users, as shown in the following screenshot. Click **Save** to save your choices:

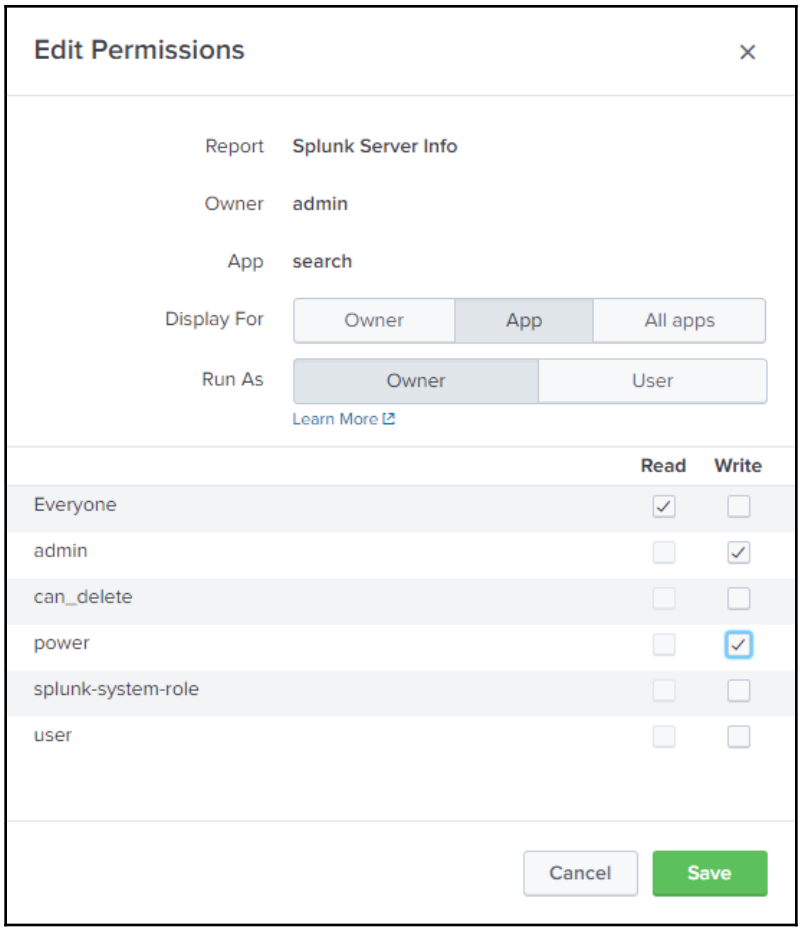


Fig 8.3: Setting report permissions

You can now see your new report listed along with all the other reports by clicking **Reports** from the menu options (from within the **Search & Reporting** app). Clicking on the Splunk **Server Info** report will run the report, as you would expect. Clicking **Open in Search** opens (and runs) the report while allowing you to view and edit the SPL. You may choose to make some modifications and **Save** them to the existing report, or create a new report using the **Save As** | **Report** option and giving the new report another name.

You can also view all the reports by clicking **Settings** | **Searches, reports, and alerts** and selecting **Search & Reporting (search)** from the app dropdown. Clicking on the Splunk **Server info** report from this context allows you to view and edit the Splunk search SPL for this report.

Clicking the **Edit** dropdown in either of these contexts allows you to perform a number of additional setup activities on the report:

- **Permissions**: Set app context and read/write permissions.
- **Schedule**: Schedule report to run periodically along with associated options.
- **Acceleration**: Enable report acceleration if applicable (must be scheduled).
- **Summary Indexing**: Have tabular results sent to a summary index (must be scheduled).
- **Disable**: Disable the report.
- **Advanced Edit**: Configure a vast number of detailed reporting, visualization, and alert action options.
- **Clone**: Make a copy of this report—the default name will be `<report name>` clone.
- **Embed**: Embed a URL into this (scheduled) report in an external website.
- **Move**: Move the report to another app. This changes the location of the `savedsearches.conf` file containing the report details, as well as some related metadata, to the specified app.
- **Delete**: Delete the report.

In order to configure several of these options, you must first configure the report to be *scheduled* to run on a periodic basis – so let's create a report we might want to see on a regular basis and schedule it to run.

Scheduling a report

Running a recurring report that displays all the static features (memory, CPU, and so on) of your Splunk servers probably isn't useful, but a periodic look at disk usage is—so let's modify the preceding report to display just the disk used and the remaining available disk space. On the existing report, click **Edit** | **Open in Search** and eliminate most of the previous SPL so that you get a table of server names, disk space used, and available GB as shown in following code:

```
| rest services/server/status/partitions-space
| eval disk_capGB=round(capacity/1024, 3),
disk_availGB=round(available/1024, 3), disk_usedGB = disk_capGB -
disk_availGB
| table splunk_server disk_usedGB disk_availGB
```

After running the report to ensure it works, you can set the visualization options by going through the following steps :

1. Click the **Visualization** tab
2. Under **Select visualization**, choose **Column Chart**
3. Under **Format** | **General**, click the middle **stacked** button, and click **On** for **Show Data Values**
4. Click the **Y-Axis** tab and select **Custom** from the **Title** dropdown, and enter **Disk Space** – GB or similar in the blank field
5. Click the top-right **X** to dismiss the form
   The form will look similar to the following image:

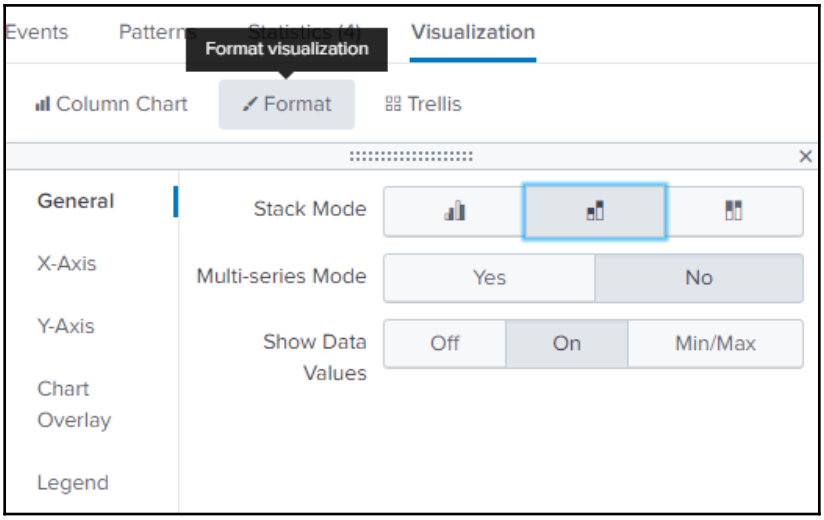


Fig 8.4: Visualization options

You can now click **Save As** | Report and give the report the name of `Splunk Disk Space Usage` or similar and click **Save**. Change the permissions to assign the report to the Search app with the preferred permissions and click **Save** again. Depending on how many servers you have, you should see a report similar to the following image. The top of each bar reflects the total disk capacity for each server:

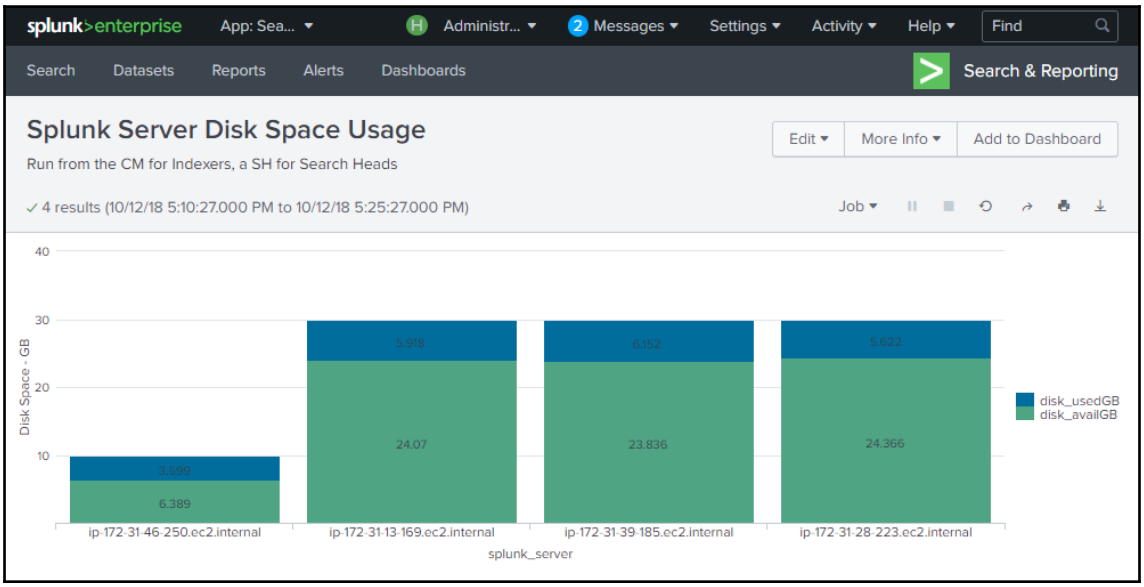


Fig 8.5: Splunk Server Disk Usage report

Note that in the preceding example, the Splunk servers have less than the recommended reference server disk capacity of 300 GB—this is a development Splunk environment that does not ingest significant volumes of data.

Let's schedule this **report** to run on a weekly basis and send the results to our email address by going through the following steps:

1. Click **Edit** | **Edit Schedule**.
2. Click the **Schedule Report** checkbox; the form will expand to provide a number of scheduling options. We could just keep the default settings of Run every week on **Monday** at **6:00**, but just to make it interesting, we'll use a cron schedule.
3. Click **Schedule**, select **Run on Cron Schedule**, and accept the provided **0 6 * * 1** in the **Cron Schedule** field, which is, again, every **Monday** at **6 AM**.
4. Set the **Time Range** to **Last 15 minutes.**
5. Keep the defaults **Schedule Priority** as **Default** and **Schedule Window** as **No window**; you can hover over the **?** next to these headings to see what they're about.

For reference, here is an example of the cron format:

```
More info at: https://en.wikipedia.org/wiki/Cron

┌──────────────────────── minute (0 - 59)
|  ┌──────────────────────── hour (0 - 23)
|  |  ┌──────────────────────── day of the month (1 - 31)
|  |  |  ┌──────────────────────── month (1 - 12)
|  |  |  |  ┌──────────────────────── day of the week (0 - 6) (Sun to Sat)
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
* * * * *

*/5 * * * *       Every 5 minutes
0 */12 * * *      Every 12 hours, on the hour
*/20 * * * 1-5    Every 20 minutes, Monday through Friday
0 9 1-7 * 1       First Monday of each month, at 9 AM
0 9 25-31,1 * *   Every 25th through 31st, and the 1st, day of each month,
at 9 AM
```

At this point, the report is scheduled, but you would have to click on **View Recent** from the reports list to see the results. Let's configure it to send the report to our email address by going through the following steps:

1. Under **Trigger Actions** | **Add Actions,** click **Send email**.
2. Enter the destination email address (or multiple email addresses separated by commas).

3. Click **Inline (Table)** and **Attach PDF** and then click **Save**. The form will look similar to the following image:

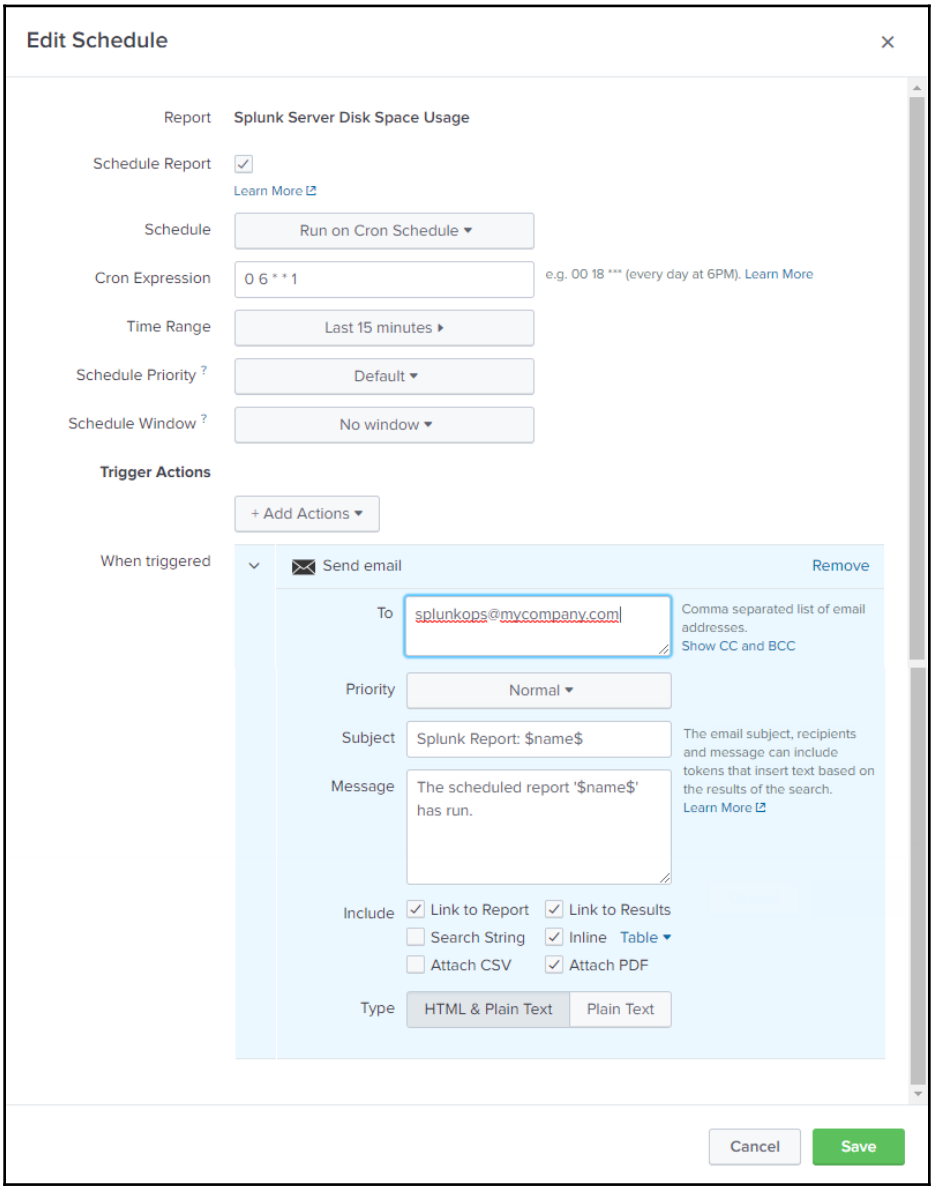


Fig 8.6: Settings for scheduling a report

The report will now be run at the specified time, and an email will be sent with an attached PDF report that reflects the visualization in Splunk web.

Note that if you built the report under the **Search & Reporting** app, then all of the report settings will be saved in `/opt/splunk/etc/apps/search/local/savedsearches.conf` (on Linux); the access permissions and other metadata are stored in `/opt/splunk/etc/apps/search/metadata/local.meta`.

You should not alter these files directly if you are using a search-head cluster, as files that have been manually edited/updated on one search head are *not* automatically distributed to the other search heads by the captain, so version mismatches are likely. It is not a good practice to manually edit any configuration files directly on clustered search heads for this reason.

You can learn much more about Splunk reports from the documentation, which can be found at the following link:

`https://docs.splunk.com/Documentation/Splunk/latest/Report/Aboutreports`

Creating a dashboard

Splunk dashboards provide a way to group a set of tables and graphs together into one useful and informative view. A dashboard consists of one or more panels that contains a table, graph, or other visualization; each can run a different search query to populate the panel. Dashboards can also exist as forms, which provide user-selectable options, such as a time-picker or drop-down lists, check boxes, or radio buttons, to alter the search parameters.

The quickest and easiest way to create a new dashboard is to start with an existing report; we'll use the two reports we created in the previous section of this chapter to populate a simple Splunk server status and information dashboard using the following steps:

1. In the **Search & Reporting** app, click the **Reports** menu button.
2. From the reports list, click **Splunk Server Disk Space Usage**.
3. In the top-right corner of the page, click **Add to Dashboard**; a **Save As Dashboard Panel** form will appear.
4. Since we're creating a new dashboard, leave the **Dashboard** setting at **New**.
5. Enter a **Dashboard Title** and optional **Description**–these will appear as a header for the entire dashboard.

6. On **Dashboard Permissions**, click **Shared in App** and enter a description for this panel in the **Panel Title** field; this will appear as a header for the first panel.
7. Change the **Panel Powered By** setting to **Report**, which will retain all the report metadata, such as the schedule, acceleration, and permissions.
8. Change the **Panel Content** to **Column Chart**.
9. Click **Save**. This form will appear similar to the following image:

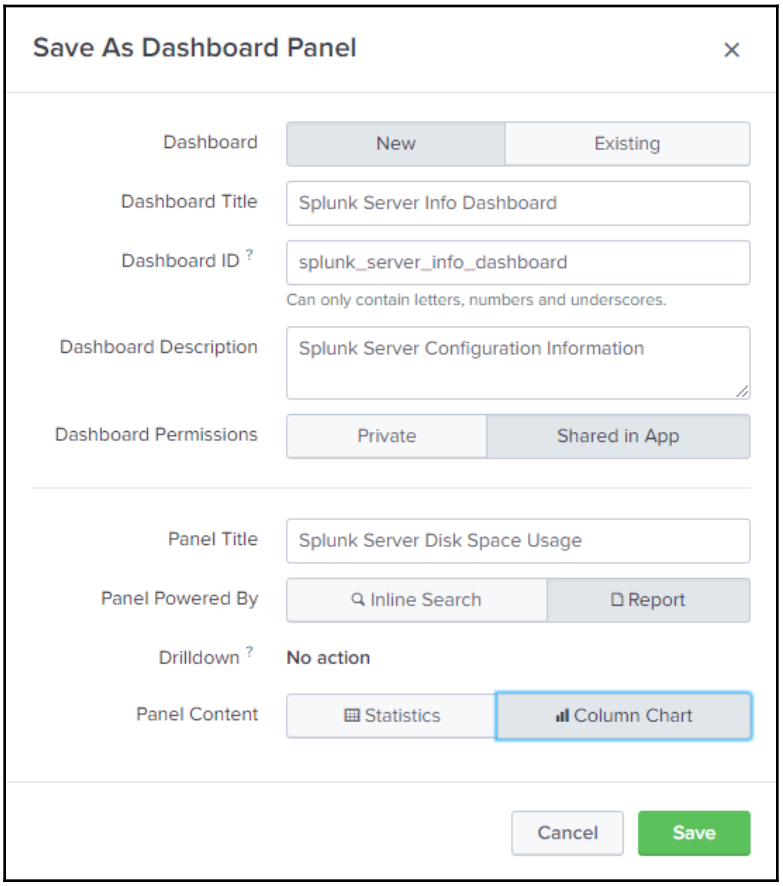


Fig 8.7: New dashboard panel options

In the next window, click **View Dashboard**. The dashboard will appear, and at this point, it won't look that different than the original report—a graph of disk utilization. Let's now add all the other Splunk server information from that report under the graph by going through the following steps:

1. Click **Edit**.
2. In the **Edit Dashboard** section that appears at the top, click **+ Add Panel**. The **Add Panel** selector will appear on the right.
3. Click **New from Report** and select **Splunk Server Info**. A **Preview** panel will appear that reflects the SPL that was used to create the report and the search results.
4. Click **Add to Dashboard** and the new panel will be added under the disk usage graph.
5. You'll notice that the new panel reflected the original report title in the lower title field; you can fill in the upper title field with the same Splunk server information text and delete it from the lower field if you want it to be consistent with the first panel.
6. Click **Save**.

You now have a two-panel dashboard; you can click **Dashboards** from the top menu and find your new dashboard among the list. You can click **Export** on the top -right of the page and export the dashboard as a PDF, schedule a PDF delivery (using a form with options similar to scheduling a report), or **Print** the dashboard.

The following is a screenshot of a new two-panel dashboard:

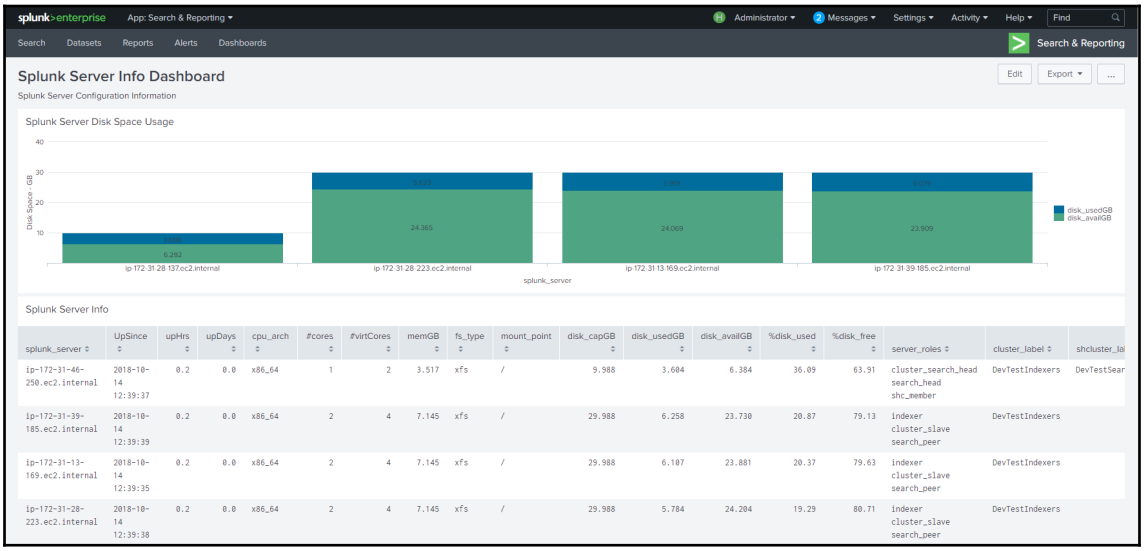


Fig 8.8: New two-panel dashboard

Note that since this dashboards' panels were derived from existing reports, the panels reference those reports, and *any modifications to the search string for either panel must be done within the originating report*. The upside of this approach is that if you modify a report, your dashboard panels are automatically updated as well; the downside is that if someone else modifies the report, your dashboard panels will be affected without your knowing, and perhaps not to your liking. The alternative is to build dashboard panels that use inline searches, where the search string is set within the panel itself.

Splunk dashboards offer a wide range of features and options; an exhaustive review of all these options is beyond the scope of a book of this size, so I'll briefly outline some of the most important features–you can experiment with these to familiarize yourself with how they work.

Adding a new panel with inline search

Adding a new panel to the existing dashboard is fairly straightforward. Starting within the existing dashboard, You can create a new panel from scratch by going through the following steps:

1. Click **Edit** | **+ Add Panel** | **New**
2. Select a visualization **–Line Chart**
3. Select the **Time-Range options**: **Use time picker** and select **Last 60 minutes**
4. Provide a title: **Throughput by Series**
5. Enter the SPL search string (see the following code block)
6. Click **Add to Dashboard**

This creates a simple line chart panel–the search SPL is shown in the following code:

```
index=_internal sourcetype=splunkd source=*metrics.log
group="per_sourcetype_thruput" series=splunkd
| timechart span=5m avg(kbps) as avg_kbps, max(kbps) as max_kbps,
perc95(kbps) as p95_kbps by series
```

After creating this new panel, click on the double-dot lines at the top of the panel, drag it to the top of the dashboard, above the other two panels, and then click **Save**. We will look at this new panel in more detail later in this section to illustrate form controls and other options.

Editing panel characteristics

If you click **Edit** on a dashboard panel, and if you're in the default **UI** (versus **Source**) mode, you will notice four icons to the top-right of each panel—let's cover what those are.

The first icon varies depending on whether the panel was derived from an existing report or is using an inline search that was provided when the panel was built. Clicking the report icon gives you a summary of the report's configuration and some options, such as **Edit in Search**. If your panel uses an inline search, the first icon will be a magnifying glass; clicking this allows you to alter the search **SPL** and **Time-Range** options, as well as select an **Auto-Refresh Delay** to automatically re-run the search and update the panel periodically.

Clicking the second icon allows you to change the type of **visualization** (line chart, bar chart, and so on). The icon will reflect the chart type.

The next paintbrush icon allows you to change the **visualization options**—this is the same format we used to edit a visualization in our disk usage report.

The last (triple dots) icon allows you to configure a **Drilldown** action for when you click on something inside the panel, or to configure a **Trellis** configuration that, for example, splits a multiple-series graph into multiple graphs, one for each series.

Also, while in **Edit** | **UI** mode, you can **drag a panel** to another location within the dashboard; dragging a panel to the side of another panel will place them side by side rather than using the default stacked-panel arrangement, and will size them accordingly.

If your panel is a **Statistics Table** view, you can click the paintbrush alongside each column header to configure the variable colors based on the values or other contents of each cell. In the following example, the threshold value ranges for `pct_disk_free` are obviously skewed to unrealistic numbers to illustrate the color range effect:

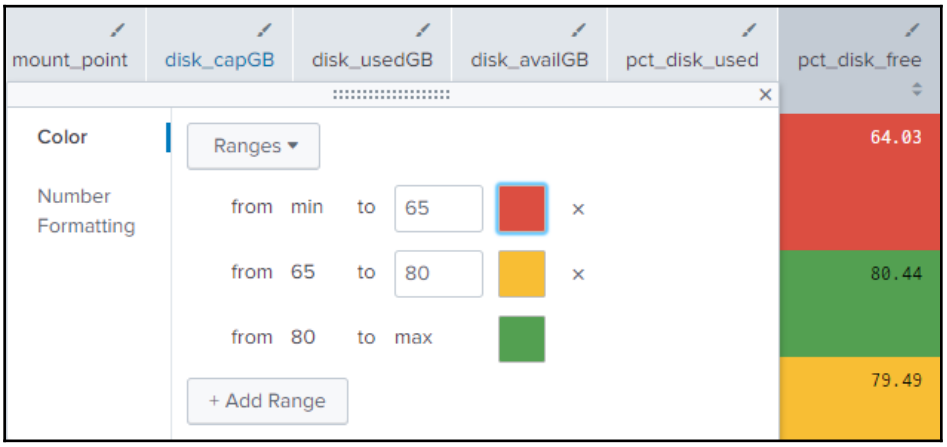


Fig 8.9: Selecting color ranges for table cells

Again, the best way of learning this using is to experiment with all these options. Be aware that most visualizations require a search using transforming commands, such as `stats`, `chart`, or `timechart`, to render properly. You can reference the Splunk documentation for more information about the data-formatting needs of each visualization.

Using dashboard forms

You can also add controls to a dashboard so that users can select variable time ranges or other criteria that affect the data views. Let's go through a simple scenario as shown in the following steps:

1. In **Edit (UI)** mode, click **+ Add Input**
2. Select **Time** to add a **Time-Range picker** control to the top of the dashboard
3. Click the paintbrush icon on the control and add **Time Range Selection** to the **Label** field
4. Click to enable **Search on Change**
5. Enter **timerange** in the **Token** field—we'll learn more about how to use tokens in a bit.

For most of the input controls, you can specify the selections that are available to the users and what values to assign to those selections in the **Name** and **Value** fields of the **Static Options** section of the control. But you can also have the control self-populate the available values by running a search; this is useful as the selections may vary over time. Let's configure a self-populating input by going through the following steps:

1. Click **+ Add Input**
2. Select **Dropdown**
3. Click the paintbrush at the top of the control
4. Enter **Select a Series** in the **Label** field
5. Click to enable **Search on Change**
6. Enter **series** in the **Token** field
7. Drag the vertical scroll control down until you get to the **Dynamic Options** section and enter the SPL from the following code block in the **Search String** field
8. Select **Last 60 minutes** in the **time-range** dropdown
9. Enter **series** in both the **Label** and **Value** fields
10. Click **Apply**
11. Click to **Save** the dashboard. Here's the SPL for the search field:

```
index=_internal sourcetype=splunkd source=*metrics.log
group="per_sourcetype_thruput" | stats count by series | table series
```

This new control will now run a search and populate itself with all the various types of series associated with splunkd metrics events when the dashboard loads, and the user can select a time range and a series from these controls that can be applied to one or more dashboard panels. These time-range and series selections will be reflected in the tokens we gave a name to in each control. Now we just need to configure the dashboard panels to use the selections in their search strings by referencing these tokens.

Using tokens

To apply the tokens in the preceding section to the last panel we built using an inline search, click **Edit (UI)**, and then click the magnifying glass icon for the throughput by series panel to edit its search string, as shown in the following code:

```
index=_internal sourcetype=splunkd source=*metrics.log
group="per_sourcetype_thruput" series=$series$
earliest=$timerange.earliest$ latest=$timerange.latest$
| timechart span=5m avg(kbps) as avg_kbps, max(kbps) as max_kbps,
perc95(kbps) as p95_kbps
```

As you can see in the preceding **bold** entries, you can use a token from a form control to represent the values from user selections by surrounding the token name from each control with `$`, as you can see in the `series=$series$` entry. For a time-range token, you can use the native `$timerange.earliest$` and `$timerange.latest$` Splunk tokens to specify the time range to be used in the search. You can use these tokens anywhere in the dashboard, but you have to make sure each token name is unique across your controls. It's just that easy!

You can also use a token in the title fields, by entering `Series: $series$` (for example)into the lower `Title` field in the panel—the selected series will be displayed above the line graph. In the following screenshot, you can see that the user-selected **Time Range** and **Series** is reflected in the chart:

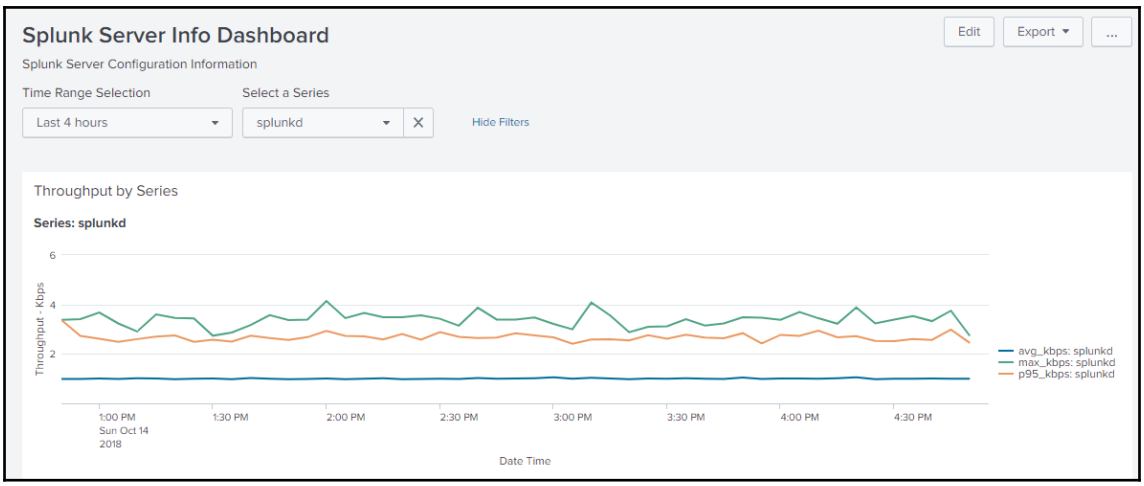


Fig 8.10: Using form controls in dashboards

Working with Simple XML

So far, we've been using the UI method of configuring a dashboard, but beneath the covers, Splunk converts our selections to what it calls **Simple XML**, which it saves in the `/opt/splunk/etc/apps/search/local/data/ui/views` directory (on Linux) in a file called `splunk_server_info_dashboard.xml` for this example dashboard. As you work more with dashboards, you will find that you sometimes might need to work with the XML itself–so let's take a look.

Click **Edit** and then **Source**; you will see that all the XML that makes up the dashboard configuration. An excerpt reflecting the initial tags, the form controls, and the first row and panel within that row are depicted in the following code block. I've marked some of the tags in bold to highlight their placement and use. Note, for example, that the **inputs** has a **type** attribute, as well as attributes for the tokens and the `searchWhenChanged` setting. Within the Select a **Series** input, you can see the search and query tags for the self-populating search. Further down, you can see a `<row>` tag that contains the **Throughput by Series** panel, within which you find the search query, and the various charting options specified:

```
<form>
  <label>Splunk Server Info Dashboard</label>
  <description>Splunk Server Configuration Information</description>
  <fieldset submitButton="false">
    <input type="time" token="timerange" searchWhenChanged="true">
      <label>Time Range Selection</label>
      <default>
        <earliest>-24h@h</earliest>
        <latest>now</latest>
      </default>
    </input>
    <input type="dropdown" token="series" searchWhenChanged="true">
      <label>Select a Series</label>
      <fieldForLabel>series</fieldForLabel>
      <fieldForValue>series</fieldForValue>
      <search>
        <query>index=_internal sourcetype=splunkd source=*metrics.log
group="per_sourcetype_thruput" | stats count by series | table
series</query>
        <earliest>-60m@m</earliest>
        <latest>now</latest>
      </search>
    </input>
  </fieldset>
  <row>
    <panel>
      <title>Throughput by Series</title>
      <chart>
        <title>Series: $series$</title>
        <search>
          <query>index=_internal sourcetype=splunkd source=*metrics.log
group="per_sourcetype_thruput" series=$series$
earliest=$timerange.earliest$ latest=$timerange.latest$
| timechart span=5m avg(kbps) as avg_kbps, max(kbps) as max_kbps,
perc95(kbps) as p95_kbps</query>
          <earliest>$timerange.earliest$</earliest>
          <latest>$timerange.latest$</latest>
```

```
        <refresh>1h</refresh>
        <refreshType>delay</refreshType>
      </search>
      <option name="charting.axisTitleX.text">Date Time</option>
      <option name="charting.axisTitleX.visibility">visible</option>
      <option name="charting.axisTitleY.text">Throughput - Kbps</option>
      <option name="charting.axisTitleY.visibility">visible</option>
      <option name="charting.axisTitleY2.visibility">visible</option>
      <option name="charting.chart">line</option>
      <option name="charting.drilldown">none</option>
      <option name="charting.legend.placement">right</option>
      <option name="refresh.display">progressbar</option>
    </chart>
  </panel>
</row>
```

Spend a little time looking through this XML and you'll soon see that it's fairly straightforward; as you work more with dashboards, you may find it easier to copy/paste and modify panel entries rather than create each one from scratch. There are also some advanced charting options that have to be set within the XML, as there aren't UI controls for setting them.

As an example of an XML-only setting, you can add a simple attribute to the opening `<form>` tag, (or `<dashboard>` tag if your dashboard doesn't have any controls) to make a dashboard auto-refresh periodically, as in the following example, which will cause it to refresh every 300 seconds (5 minutes):

```
<form refresh="300">
```

You'll note that if you don't add any input controls to your new dashboard, the opening/closing tags will be `<dashboard></dashboard>` instead of `<form></form>`; otherwise, there is no difference between the two.

Improving dashboard performance

If you have a dashboard running several searches that are similar, you can save search resources (and time) by creating a **base search** (also known as a **Global search**) for the dashboard, and configuring the applicable panels to use a **post-process search** to modify the results of the base search. The following code shows a simple example—and note that you can use HTML-format comments in Simple XML:

```
<! -- Base search -->
<dashboard>      # (or <form>)
  <label>Dashboard with post-process search</label>
```

```
  <search id="baseSearch">
    <query>
      index=_internal source=*splunkd.log | stats count by component,
log_level
    </query>
  </search>

... (tags below compressed to save space)

<!-- post-process search # 1 -->
 <row> <panel> <chart> <title>Event count by log level</title>
        <search base="baseSearch">
          <query> stats sum(count) AS count by log_level </query>
        </search> </chart> </panel>
...
<!-- post-process search # 2 -->
       <panel> <chart> <title>Error count by component</title>
        <search base="baseSearch">
          <query> search log_level=error | stats sum(count) AS count by
component </query>
        </search>
...
```

You have to use transforming searches in the base search to avoid returning raw events, and be aware that there are some limitations for base and post-process searches that may arise from the following situations, which will cause confusing results in your dashboards:

- Base searches cannot return more than 500,000 events (the rest are discarded).
- Splunk Web can timeout from search operations that take more than 30 seconds to complete.
- Base searches cannot pass more than 10,000 events to post-process searches.

Using JavaScript and CSS within a dashboard

Since the Simple XML used to format Splunk dashboards ultimately gets converted to a standard web page, you can use custom JavaScript and CSS files to control objects on the dashboard page by adding script and/or stylesheet attributes to the opening `<dashboard>` or `<form>` tag, as shown in the following code:

```
<dashboard script="my_javascript_code.js" stylesheet="my_stylesheet.css">
```

Search the web for more information and examples—the opportunities are almost limitless.

Event-handlers

The last feature I should introduce (although there are many others, but space prevents me from mentioning them all) is **event-handlers**, which let you listen for and define responses to state changes and behavior, such as a mouse click or input form selection in a dashboard. This is accomplished using the `<change>` and `<condition>` elements in Simple XML within a control. The best explanation is an example, such as the following code, which creates a radio control to let you hide or display the **Metrics Throughput** panel. There are two `<condition>` elements that capture any changes to a new radio setting: one sets the token, the other unsets the `show_panel` token, which is used with a `depends` attribute in the `<panel>` element to control whether it is visible:

```
...
    <input type="radio" token="show_panel" searchWhenChanged="true">
      <label>Show Metrics Throughput</label>
      <choice value="show">Show</choice>
      <choice value="hide">Hide</choice>
      <change>
        <condition label="Show">
          <set token="show_panel">true</set>
        </condition>
        <condition label="Hide">
          <unset token="show_panel"></unset>
        </condition>
      </change>
      <default>show</default>
      <initialValue>show</initialValue>
    </input>
...
    <!-- 'depends' controls whether the panel is visible -->
    <panel depends="$show_panel$">
...
```

We've only scratched the surface of the many uses and configuration options of Splunk dashboards, and creating stunning and performant dashboards is a skill set in and of itself. You can learn much more about configuring dashboards from the Splunk documentation at the following links:

- `https://docs.splunk.com/Documentation/Splunk/latest/SearchTutorial/Createnewdashboard`
- `http://docs.splunk.com/Documentation/Splunk/latest/Viz/Aboutthismanual`

Creating an alert

Let's build an alert that notifies us when available disk space falls below 15% of the total capacity so we can avert any issues that can be caused by running out of disk space.

First, create a search (in **Search & Reporting**) using a modification of the SPL we used to create the disk usage report earlier in this chapter, as shown in the following code:

```
| rest services/server/status/partitions-space
| eval pct_disk_free=round(available/capacity*100,2),
pct_disk_used=round(100-(available/capacity*100),2)
| eval disk_capGB=round(capacity/1024, 3),
disk_availGB=round(available/1024, 3), disk_usedGB = disk_capGB -
disk_availGB
| where pct_disk_free <= 15
| table splunk_server disk_capGB disk_usedGB disk_availGB pct_disk_used
pct_disk_free
```

Note the use of the `where` command to filter only the events where the calculated available disk space is less than or equal to 15%; you can alter the threshold value to a higher number so that you get some results in the table to ensure that this is working in your environment, and then put the threshold back to 15% (or the number of your choosing). The idea is that if everything is good, then no events will pass through the filter; if we're running low on disk space, an event will bypass the filter and we'll use that to trigger an alert email. Here are the remaining steps:

1. Set the **Time Range** to **Last 15 minutes**.
2. Click **Save As** | **Alert.** A form will appear that is very similar to the one that we used for setting up a scheduled report, except that this one has a few additional settings.

2. Click **Shared in App** to set the **Permissions** and leave the **Alert type** as **Scheduled**. The **Time Range** settings are the same as those that you saw in the scheduled report, as shown in the following screenshot:

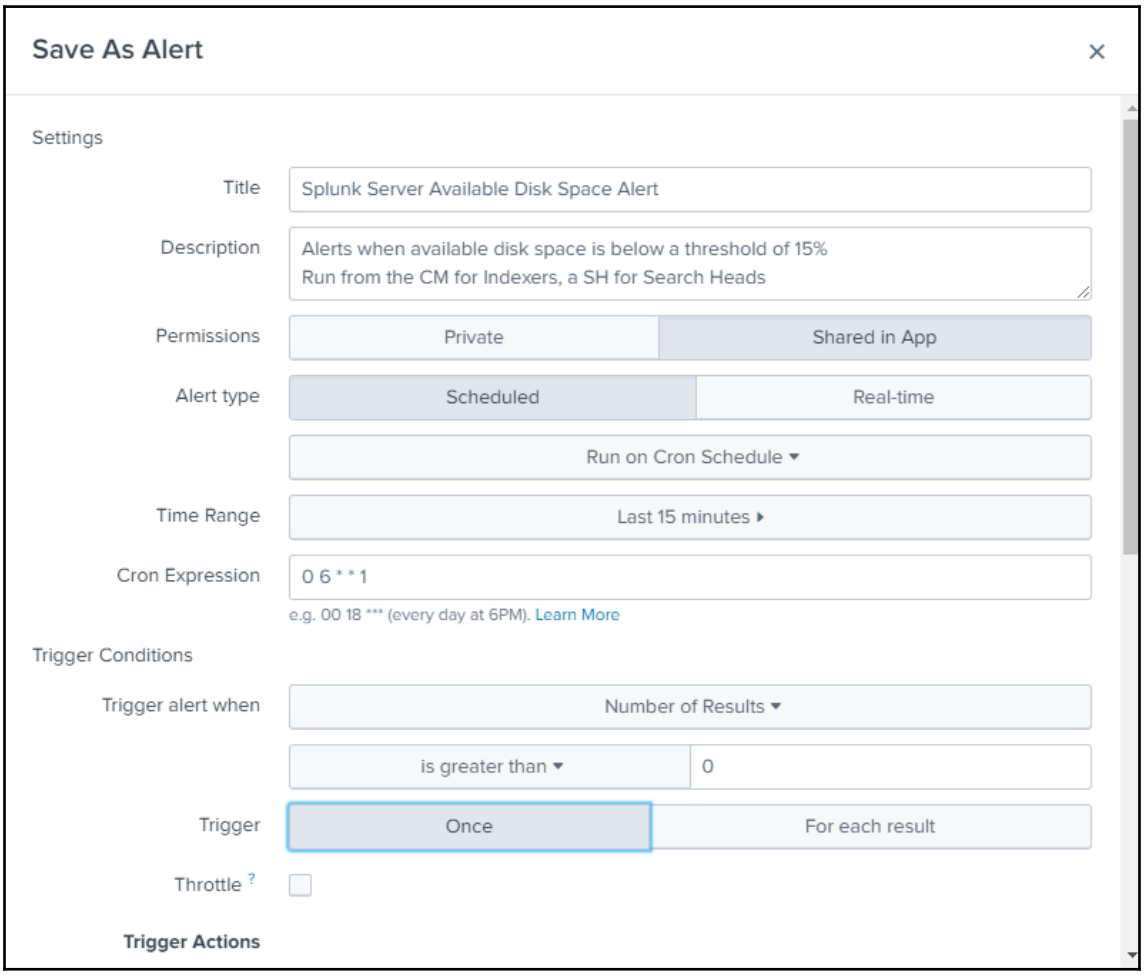


Fig 8.11: Alert settings form

Under the **Trigger Conditions** section, you can set the alert to trigger in a variety of ways; we'll use the default of **Trigger alert when Number of Results is greater than 0**, and set it up so that it only **triggers Once**. Under that option, the **Trigger Actions** options allow you to send an email or take other actions, which are the same that were available for scheduled reports.

The following example alert email was created by temporarily setting the percent available disk space threshold to 65%:

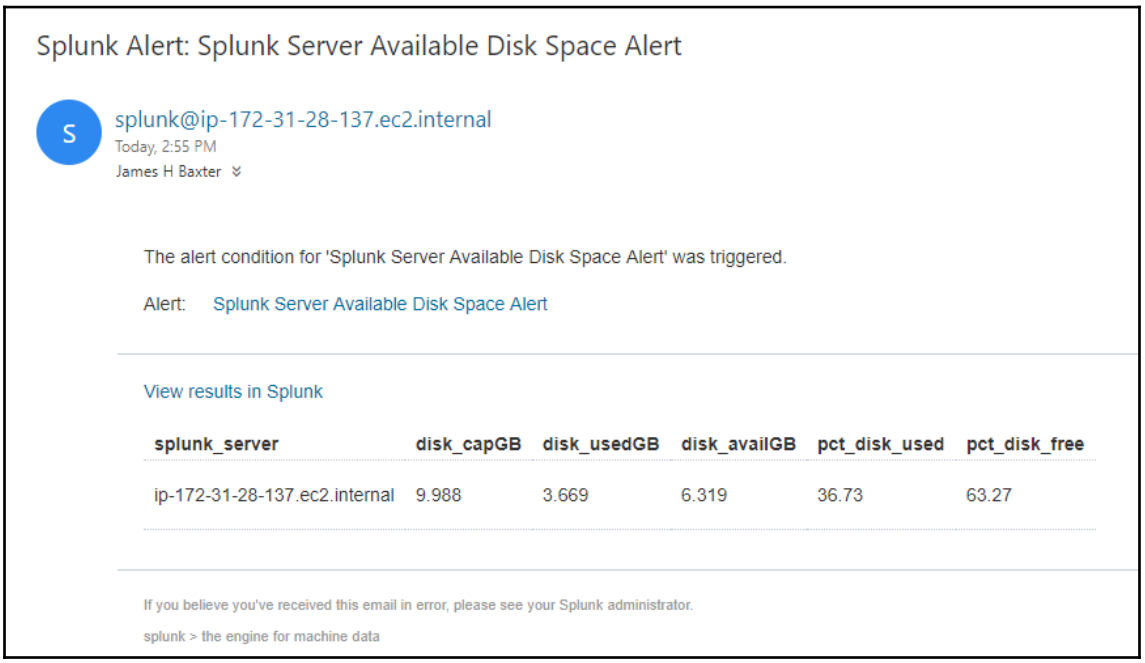


Fig 8.12: Alert email

You can learn more about configuring Splunk alerts from the documentation at the following link:

`https://docs.splunk.com/Documentation/Splunk/latest/Alert/Aboutalerts`

Summary

In this chapter, we covered how to create reports from searches, how to schedule them to run on a periodic basis, and how to use reports to create panels in a new dashboard. We then learned how to add more panels to our dashboard and configure a number of useful features within it. We finished up with learning how to use a search to create an alert that can notify us when things are amiss.

In the next chapter, we'll take a close look at several Splunk apps that you'll want to be familiar with.

9
Splunk Applications

In this chapter, we'll cover how to combine configuration files, scripts, knowledge objects, and reports/dashboards into packages called apps that make Splunk more useful and relevant to specific technologies or business-driven use cases. This chapter will also introduce several of the most useful (and mostly free) apps and add-ons available from Splunkbase that further extend the value of Splunk by providing optimized data collection and management functions for a wide variety of technologies, including Linux and Windows servers, databases, and various logs and metrics from AWS, to give just a few examples. Finally, we'll review the Splunk Machine Learning Toolkit, DB Connect, and Splunk's premium apps – IT Service Intelligence, Enterprise Security, and User Behavior Analytics, and see how they fit into comprehensive monitoring and situational-detection solutions.

The specific topics discussed in this chapter include the following:

- Apps and Add-Ons
- How to create Splunk apps from templates
- Using Splunkbase to find and install free apps
- Using Linux and Windows TA applications to monitor the infrastructure
- Installing and configuring Splunk DB Connect to work with data from relational databases
- Installing and getting familiar with the ML toolkit
- Becoming aware of the Splunk Premium apps

Let's get started with Splunk apps!

Splunk apps and add-ons

Apps and add-ons extend the functionality of the Splunk platform. A Splunk **app** is a collection of knowledge objects, and as you know, a knowledge object is a broad term that is applied to configuration files, saved searches, macros, lookups, and so on. An app can also include scripts that are used to retrieve data from external sources and/or HTML, CSS, XML, image, and other files to create user interfaces and visualizations that expand and increase Splunk's functionality to meet user needs.

By default, the Splunk platform includes one basic app that enables you to work with your data: **Search & Reporting.** To expand Splunks' functionality, you can install other apps from Splunkbase or create your own. Most of the apps provided by Splunk or other users on Splunkbase are fairly sophisticated and greatly extend the functionality of the Splunk platform.

An **add-on**, on the other hand, is generally an app that enables the Splunk platform to collect and ingest a particular type of data from other technologies or vendors. An Add-on will typically include a script or code and related configuration files to support the data-collection process and task-specific saved searches and macros, and many do not include a user interface—they play a supporting role only. Examples include the Splunk Add-on for Unix and Linux and the Add-on for Microsoft Windows, both of which collect OS-level logs and metrics, and the Splunk Add-on for Amazon Web Services, which interfaces with various AWS technologies to collect and store logs and metrics data into Splunk indexes.

To generalize, apps offer user interfaces and tools that enable you to work with your data, and they often rely on add-ons to ingest various types of data. This will all make more sense as you actually work with apps—let's get started.

Creating a Splunk app

As we mentioned, you can create your own apps in Splunk. In practice, user-created apps—or more specifically, the app directories and their contents—are typically used as a container for your saved searches, reports, dashboards, and configuration files that pertain to the data for a specific technology, application, environment, or business unit. These apps can be as simple as a few `.conf` files (such as `indexes.conf` or `inputs.conf`) to configure Splunk to import and store data, or a sophisticated collection of knowledge objects, scripts, and a full-featured user interface to allow data collection, visualization, analysis, and reporting. All of the files within an app are in plain text (and can be edited) and Splunk provides full documentation on all of its `.conf` files—including the stanzas, attributes, and possible values—so that the purpose of each configuration setting is transparent.

You can create a new Splunk app from Splunk Web yourself by going through the following steps:

1. Click the **Apps** dropdown
2. Select **Manage Apps**
3. Click **Create app**
4. In the form that appears, give the app a **name** that will be displayed in the left-hand menu, as well as a recognizable, OS-friendly **folder name**
5. The **Version** can be 1.0 (it's your first one!)
6. If you want your app to have an icon with its name listed on the left-hand side with all the other Splunk apps, provide a user interface for selecting reports, dashboards, and so on, and set **Visible** to **Yes**—if it is just going to be a container for some configuration files, set this to **No**
7. Author and description are self-explanatory

The following screenshot shows an example form:

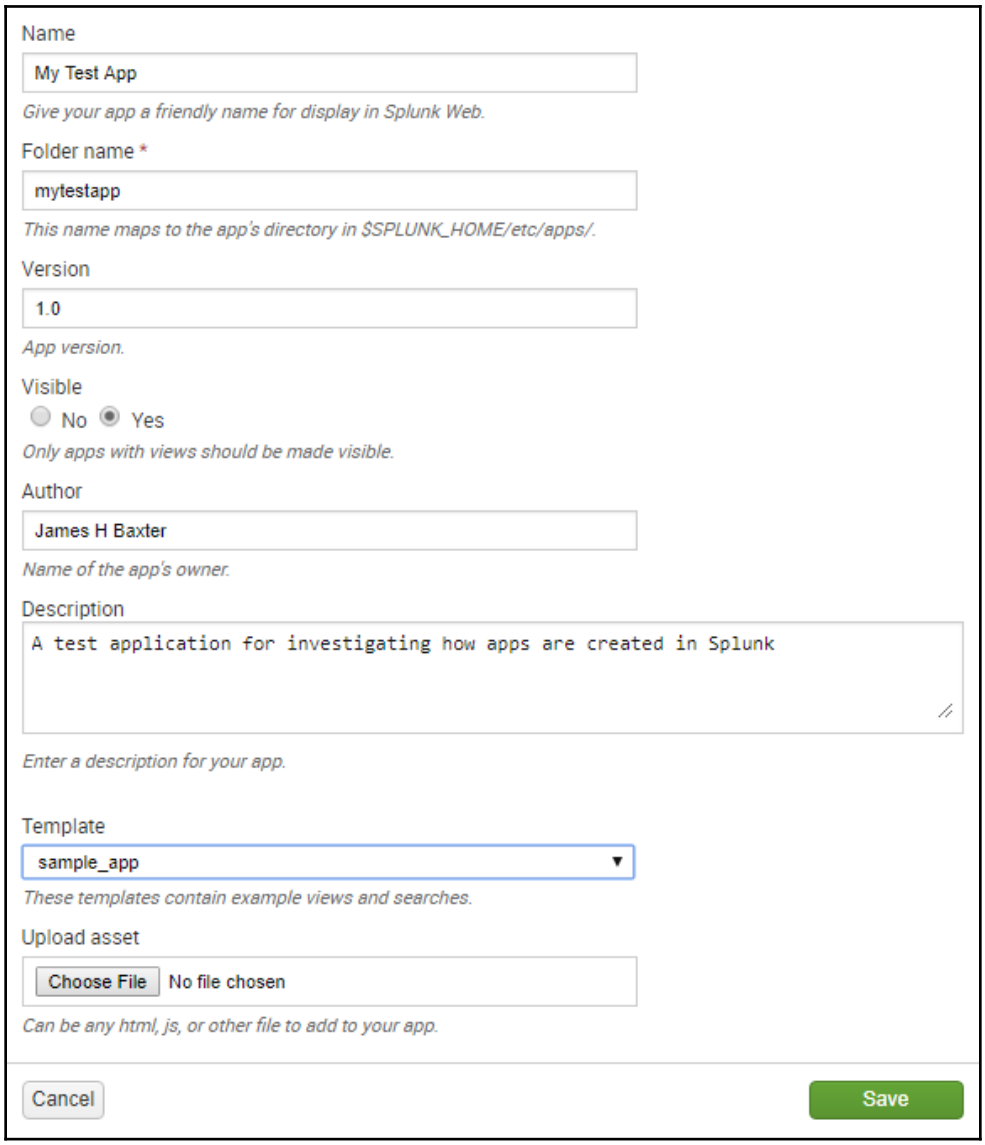


Fig 9.1: Creating a new Splunk app

Splunk uses one of two selectable **Ttemplate** for creating the starting structure of your new app: **barebones** and `sample_app`. You can select **barebones** if your app is not going to have a user interface; otherwise, choose **sample_app**. Finally, you can click **Choose File** to upload any user interface files (HTML, CSS, JS, images) to be used with your app. Then click **Save**.

After you have created your app, your app directory (for Linux), its subdirectories, and the general contents of each folder will be shown here as shown in the following code; if you chose the **barebones** template, the `appserver` folder will be missing:

```
/opt/splunk/etc/apps/mytestapp/

  appserver/static  # images, html, css, etc. files for the user interface
  bin/        # script files that collect or manipulate data
  default/    # app.conf and several other default conf files
  local/      # where you and Splunk put .conf files for inputs, indexes,
props, etc.
  metadata/   # default.meta and local.meta files - stores access
permissions
```

The `app.conf` file in `.../mytestapp/default` contains the entries you made when you created the app—whether it's visible, the label, author, description, and version. If your app is going to have a user interface, then the `.../appserver/static/` folder is where you can put your own image files to use in the app, as well as alter the provided `application.css` file to customize the look of your app; this file is nicely commented to help you find and alter its effects on the app's appearance. Also of note is a `default.xml` file in `.../mytestapp/local/data/ui/nav/` that configures the navigation bar across the top of your app, which you can alter to suit your needs. There is also a set of sample dashboards in `.../mytestapp/local/data/ui/views` that you can view and alter from the dashboards button in the navigation bar for your app— you can delete these if you want to.

App context and permissions

It is important that you understand how Splunk permissions work within app contexts, because this affects where Splunk stores `.conf` and other files, reports, dashboards, and alerts, as well as how it controlls who can see and use or alter these reports/dashboards/alerts. A few sample scenarios using the example of creating a new report would probably be helpful for you to underastand these concepts.

First of all, if you are going to create a new report, you should consider who you would like to be able to find and use it. If it is generic in nature and meant to be visible to the Splunk user population at large, you should perhaps create it within the **Search & Reporting** app, since that's where most people usually start when they log into Splunk—they can click **Reports** and select your report from the list. In that case, you would click the **Search & Reporting** icon (if you started out in the Launcher after logging in), or select **Search & Reporting** from the **App** dropdown if you're somewhere else, before you create the new report. This puts you in the **search app context**.

Or perhaps your report is about financial transaction volumes for a specific service and should only be visible to a select business unit within your organization. In that case, you might create a new app called *Company Financials*, and configure it to use a financials folder under `.../etc/apps` to store its files. Before you create your new report, you should select this app from the left-hand icons or the dropdown so that you start off in the **financials app context**.

Now you're ready to build a new report in the financials app context. To do this, go through the following steps:

1. Click **Search** from the menu bar if you haven't not already
2. Enter and edit your SPL search string and test to confirm that you get the results you're looking for
3. Adjust the **Visualization** settings
4. Click **Save As** | **Report**
5. Give the report a name
6. Click **Save**
7. Click **Reports**, and click **Yours** (instead of all or this app's) to eliminate all the sample reports that were included when you created the app from the list and any others that have global visibility.

Note that your report is listed with an **Owner**, an **App**, and a **Sharing** column, as shown in the following screenshot:

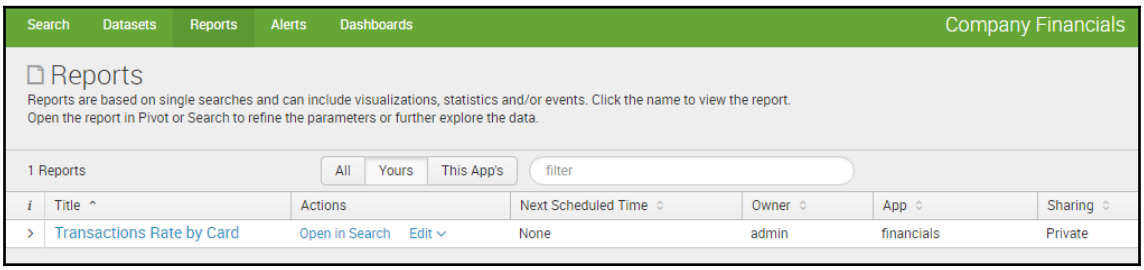


Fig 9.2: New transactions rate report

The **Owner** is you, the **admin**, in this example; otherwise the *Owner* would reflect the user ID of the person who created the report. The **App** is **financials**, because that was the app you were in when you created the report. Finally, the **Sharing** is **Private**, because at this point you haven't changed the permissions for this report from their defaults, so no one else can see it yet.

Here's the interesting thing: if you inspect the `/opt/splunk/etc/apps/financials/` folder, you might expect to find the configuration details of your new report reflected in a `savedsearches.conf` file in the `.../local` folder—but that's not where it is! Because this report is private, it currently resides in the `/opt/splunk/etc/users/admin/financials/local/` folder, and only you (or anyone with an admin role) can view and edit it. There is also an entry in the `.../etc/users/admin/metadata/local.meta` file that reflects your new report, the version of Splunk it was created under, and the last epoch time it was modified. If you look through the `.../etc/users/<my userID/` folders, you'll find a `/history` folder that contains a `.csv` file with a record of the searches you've run in this context.

Now, if you click **Edit** | **Edit Permissions**, you can change the **Display For** setting from **Owner** to **App** and, as we've done before, give **Everyone** permission to **Read**, and the admin and power roles permission to **Write**. If this is a report that only members of a Financials group should be able to view, you have probably created a Splunk role for that group, in which case you'd only give **Read** access to that role, and perhaps a subset of this group would have a role belonging to Financials power users—only they should be allowed to **Write** (create and edit) reports and other knowledge objects in that app.

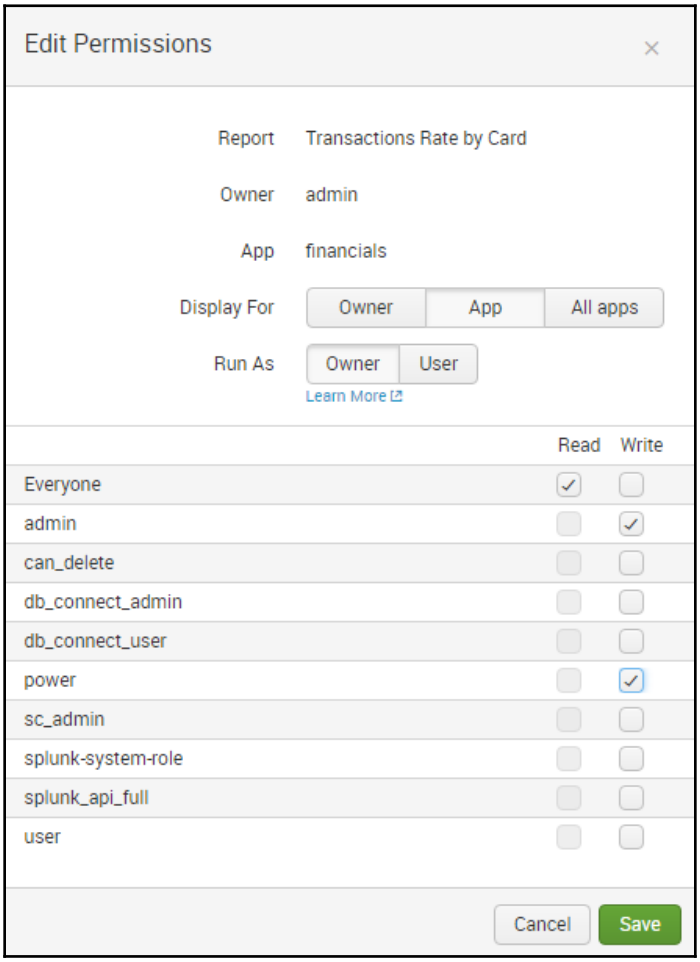


Fig 9.3: Changing permissions

In any case, changing the **Display For to App** will result in the following three changes:

- The `savedsearches.conf` file containing the report configuration will be moved from `.../etc/users/admin/local` to `.../etc/apps/financials/local`, where you expected to find it in the first place.
- An entry now appears in the `.../etc/apps/financials/metadata/local.meta` file that reflects the report and its permissions. There is also an `export = none` entry in that file that indicates this that report does not have global visibility.
- When you click **Reports** and view the list of reports, the **Sharing** field will now show **App**.

This change also means that when you are in the *Company Financials* (financials) app, and you click **Reports**, you and others can see the report (assuming they have role-level rights as well), as you would expect. But if you're in any other app, say, **Search & Reporting**, and click **Reports**, you won't see the **Transactions Rate by Card** report listed.

Changing the **Display For** permission setting to **All apps** results in the following observations:

- The `savedsearches.conf` file doesn't move—it's still in `.../apps/financials/local`
    - The export entry in the `.../etc/apps/financials/metadata/local.meta` file changes from **none** to **system** to reflect a global visibility
- The Sharing field in the app list will show **Global**

You and others will now be able to see the **Transactions Rate by Card** report when you click **Reports** and view the list from *any* app. You probably noticed that regardless of which app you're in, if you click **Reports** and view the list, you'll see reports that have an **Owner** that is listed as nobody or the user that created that report belonging to the search or other apps, with a **Sharing** level of **Global**.

You can read more about creating and configuring Splunk apps in the documentation at the following links:

- `http://dev.splunk.com/view/quickstart/SP-CAAAFDC`
- `http://dev.splunk.com/view/webframework-developapps/SP-CAAAEUC`

Next, we'll look at how to install Splunk apps that others have written.

Using Splunkbase

Splunkbase is a site where users post and share apps and add-ons with the Splunk community. You can browse and install apps and add-ons from Splunkbase on any running Splunk instance, or download the file to your personal computer and install it from there if the Splunk instance doesn't have internet access. Splunkbase has over 1,000 apps and add-ons from Splunk, Splunk partners, and the user community. You can get to Splunkbase from a Splunk instance by clicking **Apps | Browse More Apps** (or **Find More Apps**). If you want to get to Splunkbase from your PC, the URL is `https://splunkbase.splunk.com/`.

You can browse though all the available apps by category, vendor, or other groupings, or click **See All Apps**, which takes you to a page where you can select various filters that you can apply to narrow your search and a search field where you can enter a few keywords and search for apps that match. Most apps are free; a few others must be purchased by contacting Splunk sales or the author of the app. Sone examples of free, popular apps include **Splunk Dashboard Examples**, **Splunk DB Connect** (database access), and the **Splunk Machine Learning Toolkit** (all highly recommended). When you find an app you'd like to download or install, you have to click the **Login to Download** button and log in via your (free) Splunk account.

We'll cover how to install a couple of the free apps available on Splunkbase in the next section.

Splunk app and add-on for Unix and Linux

The Splunk Technology **add-on** for Unix and Linux collects OS-level data from Unix hosts and stores it in a Splunk index; the Splunk **app** for Unix and Linux provides visualizations, alerts, and so on from that data to provide insights and operational visibility in your Unix and Linux environments. You don't have to install or use the Splunk App for Unix and Linux to use the Add-on; you could create dashboards and alerts yourself using the data collected and indexed by the add-on.

There is a similar app—add-on pair for Windows environments as well, which will collect data from perfmon, registry, WMI, and other sources. The installation process is similar, so we'll just cover the installation of the Unix and Linux apps in this section.

To install the Splunk Add-on for Unix and Linux in a distributed/clustered environment, go through the following steps:

1. Install the Add-on on the cluster master via Splunk Web and then restart Splunk
2. Click the **Splunk Add-on for Unix and Linux** icon
3. Click **Continue** to go to the app setup page
4. In the setup page, enable the **File/Directory** and **Scripted** input you wish to gather data for and click **Save**

The entries for the input that you enable (the default is disabled) are saved in an `inputs.conf` file in the`$SPLUNK_HOME/etc/apps/Splunk_TA_nix/local/` file.

The form for selecting **File** and **Directory Inputs** will look similar to the following screenshot:

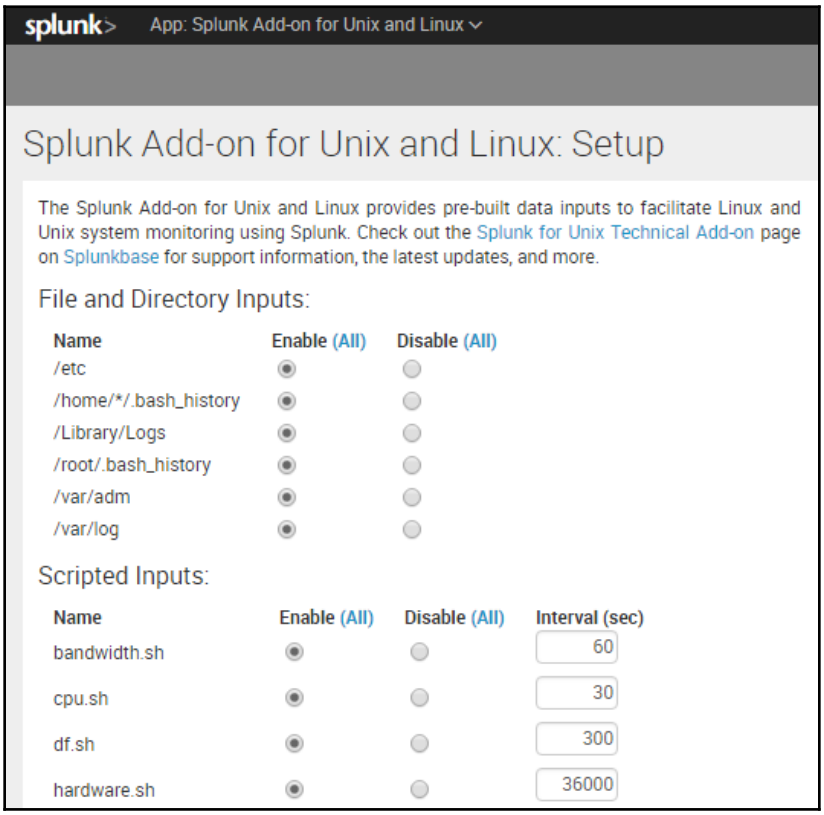


Fig 9.4: Selecting input

Be aware that if you enable *all* of the inputs, you will ingest approximately 200 MB/day per monitored host, and this ingestion counts against your daily Splunk license capacity, so you'll want to tailor the number of enabled inputs to just those that you intend to use.

Also, by default, the data collected by the enabled inputs will be sent to the default index, which is typically `main`. You may wish to use a more specific index, such as `os_nix` (or `os_win` if you repeat this process for Windows servers) and specify a retention period and perhaps a size limit. This is easily accomplished by adding an `index = os_nix` entry under each stanza of `inputs.conf`:

```
[monitor:///etc]
disabled = false
index = os_nix

[monitor:///home/*/.bash_history]
disabled = false
index = os_nix
...
```

You'll also need to create an indexes.conf file to hold the data. Note that if this is a distributed/clustered Splunk deployment, then this `indexes.conf` file should only be deployed to the *index cluster* by placing it in the `$SPLUNK_HOME/etc/master-apps/Splunk_TA_nix/local` directory on the cluster master (see the following steps) so that it gets rolled out to the index cluster only:

```
[os_nix]
homePath = $SPLUNK_DB/os_nix/db
coldPath = $SPLUNK_DB/os_nix/colddb
thawedPath = $SPLUNK_DB/os_nix/thaweddb
# Set Index Retention to 18 months
frozenTimePeriodInSecs = 46656000
# This index gets replicated across index cluster
repFactor=auto
```

You can install the **Splunk App for Unix and Linux** from Splunk Web in the same way that you did for the Add-on. After installation and a restart, upon clicking the App icon, you will be prompted to do an initial setup wherein you will want to change the **Unix Index(s)** setting from the default of `os` to reflect the proper index—either `os_nix`, if you followed the preceding example, or main, since that's where the data is going to go if you don't direct it otherwise—then click **Save** as shown in the following screenshot:

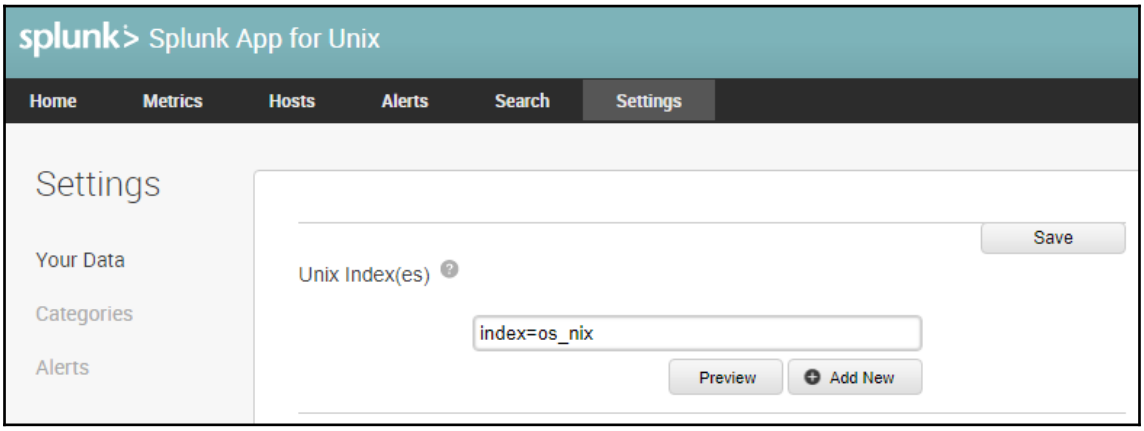


Fig 9.5: Splunk app for Unix index setting

If you're going to deploy this add-on to all the Splunk nodes (and, optionally, the app to the master node and search heads), as well as all of your hosts with universal forwarders, a workable process may include the following general steps:

1. Install the Splunk **add-on** and **app** for Unix and Linux on the **cluster master** (master node) using Splunk Web, and then restart.
2. Select the file/directory and script inputs to enable using the Add-on setup page. You can also change the polling interval if the defaults are not satisfactory.
3. Edit `.../local/inputs.conf` to add an `index = os_nix` for each input stanza.
4. Copy the entire `Splunk_TA_nix` folder (do *not* copy the `splunk_app_for_nix` folder) to `$SPLUNK_HOME/etc/master_apps`.
5. Add the `indexes.conf` file to create the `os_nix` index to the `.../master_apps/Splunk_TA_nix/local` folder—or add its entries to your preferred `indexes.conf` file in some other location.
6. Run validate cluster-bundle and apply cluster-bundle to distribute the add-on to the indexers (search peers).

7. Confirm that the new index was created and that you're getting indexer OS-level metrics with the following search: `index = os_nix | stats count by host and index=os_nix | stats count by sourcetype`.
8. Click on the Splunk app for Unix and Linux icon, then click **Configure**, change the **Unix index(es)** entry to `os_nix` and **Save**. This completes our work on the cluster master.
9. Install the add-on and app on the **deployer** using Splunk Web and restart your machine. Configure the setup modifications outlined in the preceding steps for both the add-on and the app.
10. Copy the `Splunk_TA_nix` and `splunk_app_for_nix` folders to the `.../etc/shcluster/apps` folder and deploy to the search-head cluster using `$SPLUNK_HOME/bin/splunk apply shcluster-bundle -target https://<searchheadhostname>:8089`.
11. Install just the add-on and configure the `inputs.conf` as needed on the license master, any along with heavy forwarders or other **standalone Splunk nodes** wherein the apps can't be distributed by other means.
12. Install just the Add-on on the **deployment server** using Splunk Web, select inputs, edit `inputs.conf`, and restart your machine.
13. Copy the `Splunk_TA_nix` folder to `$SPLUNK_HOME/deployment_clients`.
14. Edit the `serverclass.conf` file in `$SPLUNK_HOME/etc/system/local` to include the Add-on for the applicable forwarder clients, then run `$SPLUNK_HOME/bin/splunk` to reload `deploy-server` . The forwarders will pick up the new app the next time they phone home. You can do a simple search to make sure the forwarders got the add-on (`index=os_nix | stats count by host`), and that they're picking up all the desired metrics with `| stats count by source`.

A simple example of the `serverclass.conf` entries for a single white-listed forwarder node is shown in the following code:

```
[serverClass:os_nix]
whitelist.0 = 172.31.39.242
restartSplunkd = true

[serverClass:os_nix:app:Splunk_TA_nix]
```

Once you have OS-level data indexed, using it for reports and dashboards is as easy as identifying the metrics you want to report on. Run the search against `os_nix`, inspect the fields listed on the left, and choose/investigate those of interest. The following screenshot shows a search for the `df` metric, which shows disk capacity and usage:

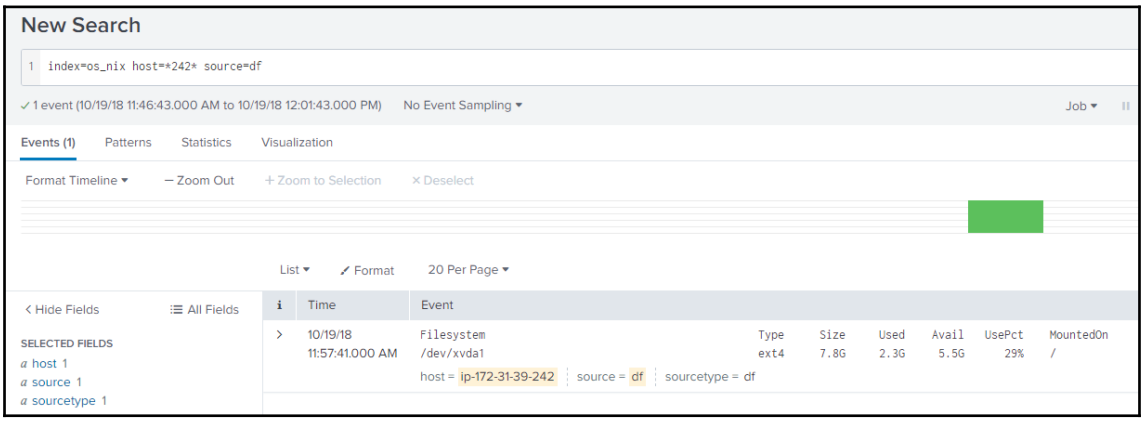


Fig 9.6: Disk metrics from OS-level data

The following links will take you to the Splunk documnetations for the app and add-on respectively:

- `http://docs.splunk.com/Documentation/UnixApp/latest/User/AbouttheSplunkAppforUnix`
- `http://docs.splunk.com/Documentation/AddOns/released/UnixLinux/About`

Machine learning toolkit

If you're not investigating and experimenting with a machine learning and its cousin artificial intelligence, then you should be—this technology is changing the world in very significant ways, and offers an opportunity to obtain a *great* deal of additional value from the data residing in your Splunk environment. Using Splunk, you can collect, search, analyze, and clean your data, then build and deploy machine learning models that detect anomalies and outliers, predict performance or reliability issues before they adversely affect your customers, and then visualize, set up alerts for, and share the results within your organization.

You may not be inclined to immerse yourself in the world of writing algorithms and building artificial neural networks, and the statistics, probability, linear algebra, calculus, and programming disciplines required to navigate this landscape. However, you can become a citizen data scientist by learning about the various categories of machine learning, the most common algorithms used for each one, the hyper-parameters (settings) each requires, and experimenting with how to apply them to the vast array of real-world datasets available to you right in your Splunk environment! I highly recommend that you install and start working with this app, and that you encourage your Splunk user community to do the same.

The MLTK is installed on search heads. You must install the appropriate **Python for scientific computing add-on** for your environment (Mac, Linux, Windows 32- or 64-bit) *before* installing the Machine Learning Toolkit – both are available on Splunkbase. The Python add-on provides the machine learning libraries and supporting utilities used by the MLTK. To install the Python package and MLTK to all the search heads in a distributed environment, go through the following steps:

1. Log in to Splunk Web on the **deployer**.
2. Click the **Apps** gear icon | **Browse more apps**, and search for **Python**.
3. Click **Install** on the appropriate package (Python for scientific computing (for Linux 64-bit), for example).
4. When the package is done installing, accept the **restart** prompt.
5. Log back in, and search for machine learning on Splunkbase.
6. Click to install the **Splunk Machine Learning Toolkit**.
7. When the installation completes, copy the `Splunk_SA_Scientific_Pythong_linux_x86_64` (or the appropriate folder) *and* `Splunk_ML_Toolkit` folders to the `$SPLUNK_HOME/etc/shcluster/apps` folder.
8. Deploy the apps to the search-head cluster using the `apply shcluster-bundle` command.

When you next log in to a search head, you will see the Splunk machine learning toolkit icon in the apps menu; clicking this will present you with the first of several slides of a tour that pops up the first time you start the app. The first description describes the MLTK environment quite well:

Welcome to the Splunk Machine Learning Toolkit. This app consists of five main components: (1) the Showcase, which provides working examples on real datasets, (2) Experiments, which offer a guided ML experience, (3) Models, where you can find ML models created outside the Experiments workflow, (4) the "Classic" workflow, consisting of Alerts, Scheduled Training, and Assistants that fall outside of the Experiments workflow, and (5) the Settings page, which allows you to configure your ML algorithms.

Upon completing the tour, or the next time you start the MLTK app, you'll be presented with the **Showcase** page, which portrays six of the more common and useful classes of machine learning applications. It also provides assistant links under the examples header, which will walk you through how apply machine learning models to various example datasets—these are quite good, and once you get the hang of how to work with the models and SPL commands using the provided data sets, you can apply the techniques to data from your own environment:

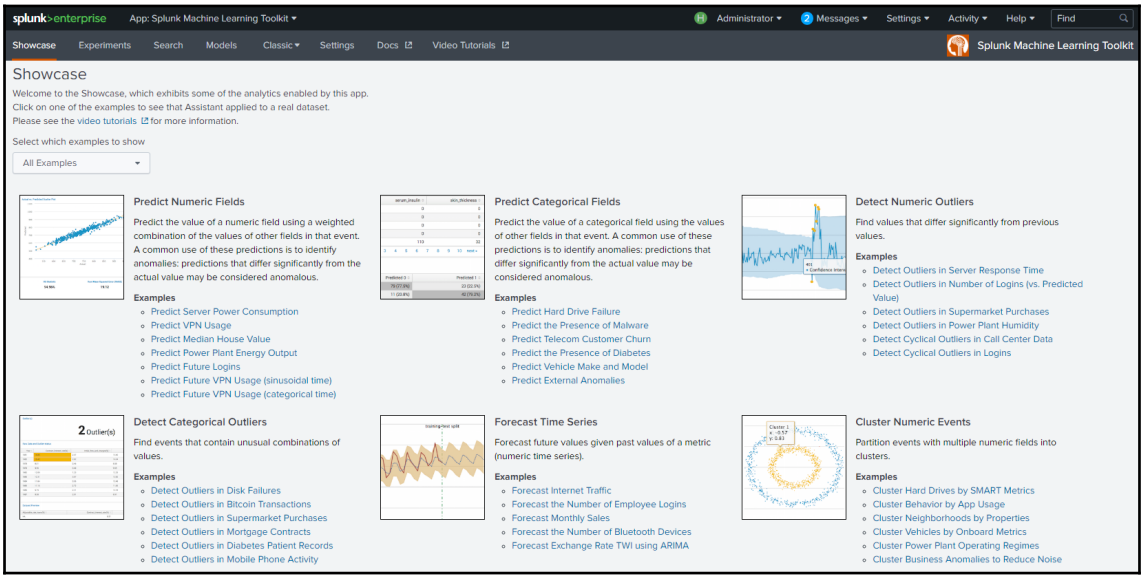


Fig 9.7: Machine learning toolkit showcase page

The Splunk documentation on the MLTK will guide you through all the features and functionality of the MLTK; a thorough read of this document coupled with experimentation with the toolkit (along with some books and lots of web searches on the subject) should give you a great start at working within this wondrous field. The following links include the MLTK documentation and my current favorite book on the subject of machine learning and artificial neural networks, which can help supplement your understanding of the topics:

- `http://docs.splunk.com/Documentation/Splunk/latest/Search/MachineLearning`
- `https://www.amazon.com/Hands-Machine-Learning-Scikit-Learn-TensorFlow/dp/1491962291/`

Splunk DB Connect

Splunk's DB Connect enables you to use scheduled **input** to query traditional relational databases and retrieve and store that data in indexes to be combined with other data sources in Splunk searches to provide a more comprehensive view of, and insights into, all the data across the Enterprise. You can also configure Splunk searches from other sources and use an **output** from DB Connect to store the tabled results in your database tables or perform **lookups** from database tables to add to and enrich search results from other sources. Finally, you can perform ad-hoc queries to add database records to search results.

This app is available for free on Splunkbase, and is a very common addition to most Splunk Enterprise environments, so I'll cover it in a bit more detail.

Requirements and installation

At the time of writing, the current version of Splunk DB Connect is 3.1.3. In previous (2.x) versions of DB Connect, the app could be installed on all the search heads in a cluster, and the captain would assume the role of actually performing the database queries; this sometimes resulted in performance and reliability issues if the search heads got too busy, so as of version 3.x, Splunk started enforcing the need to install DB Connect on a standalone node such as a heavy forwarder. You can still install DB Connect on clustered search heads to allow users to perform ad-hoc database queries, but if you want to configure it to perform scheduled queries or output, you'll need to install it on a heavy forwarder or, as I have done quite often, on a deployer, as this is a full Splunk Enterprise node that typically doesn't have a heavy background workload—so it will work quite nicely if equipped with sufficient CPU cores and memory.

This is one application for which you simply must download and read carefully through the manual before, and as, you install and configure it—this is a fairly complex application to work with, and you'll find yourself referencing the manual often. You can find the manual at `http://docs.splunk.com/Documentation/DBX/latest/DeployDBX/AboutSplunkDBConnect`.

Hardware requirements

The hardware requirements for supporting DB connect vary depending on the amount of input, output, or lookups being performed, but one example that is referenced in the Splunk manual for a performance benchmark utilized 8 CPU cores at 2.6 GHz and 16GB of RAM – this can be considered a minimal configuration. Since the queried data is stored in the indexing tier, disk-space consumption is just the amount that is needed for installing the app, which isn't a significant amount.

Java runtime

DB Connect requires that the server it is running on has the Java Runtime Environment (JRE) version 8 from Oracle installed. You can obtain the JRE from the following link; you should install it on the Splunk server and ensure that the `$JAVA_HOME` environment variable has been set, *before* installing DB Connect: `https://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html`.

Accept the license agreement and download the appropriate `.rpm`, `.tar.gz`, or `.exe` for your environment; you'll need to reference the Oracle documentation for instructions on how to install the JRE on your particular operating system.

Installing DB connect

Log into the heavy forwarder (or deployer, if you go that route) and install DB Connect from Splunkbase as you did for the other apps: click the **Apps** dropdown or icon, select **Browse more apps**, search on Splunkbase for **DB Connect**, click **Install**, and then accept the **Restart** prompt.

When the installation is complete, Splunk will have installed this app in `$SPLUNK_HOME/etc/apps/splunk_app_db_connect`. There's quite a few folders in there, but you'll only need to concern yourself with the `/drivers` and `/local` directories in most cases. You should now see the DB Connect icon in the **Apps** list, as shown in the following screenshot:

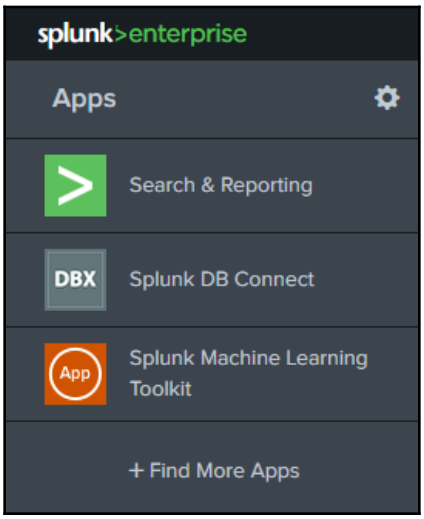


Fig 9.8: Splunk DB Connect and machine learning toolkit App icons

Database JDBC drivers

DB Connect supports DB2/Linux, Informix, MemSQL, MySQL, AWS Aurora, Microsoft SQL Server, Oracle, PostgreSQL, AWS RedShift, SAP SQL Anywhere, Sybase ASE, Sybase IQ, and Teradata databases. You will need to download the JDBC drivers for the database(es) you intend to interface with. The following list shows the are links to a few of the more common database drivers; if these are outdated, a search for `<database type> jdbc drivers` should get you where you need to go:

- **Oracle**: `https://www.oracle.com/technetwork/database/application-development/jdbc/downloads/jdbc-ucp-183-5013470.html`
- **Microsoft SQL Server**: `https://www.microsoft.com/en-us/download/details.aspx?id=11774`
- **MySQL**: `https://dev.mysql.com/downloads/connector/j/5.1.html`

After downloading each file, un-archive it and copy *just* the `<drivername>.jar` file (not all the other files that may come in the archive) to the `/drivers` directory under `splunk_app_db_connect`.

Configuring DB Connect

If you've installed the Java Runtime Environment, installed DB Connect, and copied the database drivers to the `/drivers` folder, then you're ready to start DB Connect and set it up. When you click the DB Connect icon the first time, you will be prompted to allow or prevent Splunk from collecting anonymized usage information on DB Connect—you can click **No** or **OK**. You can then click the **Setup** button to start configuring the app—this will take you to the **Configuration** | **Settings** | **General** tab, or click **Skip Setup** and navigate there yourself.

Configuring task server

As you enter the **Configuration** | **Settings** | **General** page, you may see an error message that says *Cannot communicate with task server, please check your settings*. You can ignore this for now, as that's exactly what we're going to do next. The task server is a a Java Virtual Machine that DB Connect runs in the background and uses for its polling and other operations. If the `$JAVA_HOME` environment setting is properly set, DB Connect will usually be able to detect that the path exists and populate the **JRE Installation Path** accordingly; if not, you'll have to manually enter it. The **JVM Options** field may be blank; DB Connect has a routine that usually completes this field when you click the **Save** button, so you can leave it blank for now; if that doesn't happen, troubleshoot the JRE installation or determine and enter the appropriate JVM memory and other settings yourself.

The **Task Server Port** will be set to the default of **9998**, and DB Connect may alter that as well when you click **Save**, as you can see in the following screenshot. If that doesn't work, only then should you try to manually configure another port, such as **9999** or **10000** (you can change this in both the **JVM Options** and the **Task Server Port** fields), to find an open port that works.

Note that the first time you configure this page and click **Save**, the Restarting **Task Server** countdown may expire and DB Connect will complain that it can't start—you should restart Splunk and try again before you spend too much time troubleshooting; you'll likely find that upon opening the DB Connect app after the restart that it is working after all.

An example of the **General Settings** form is shown in the following screenshot:

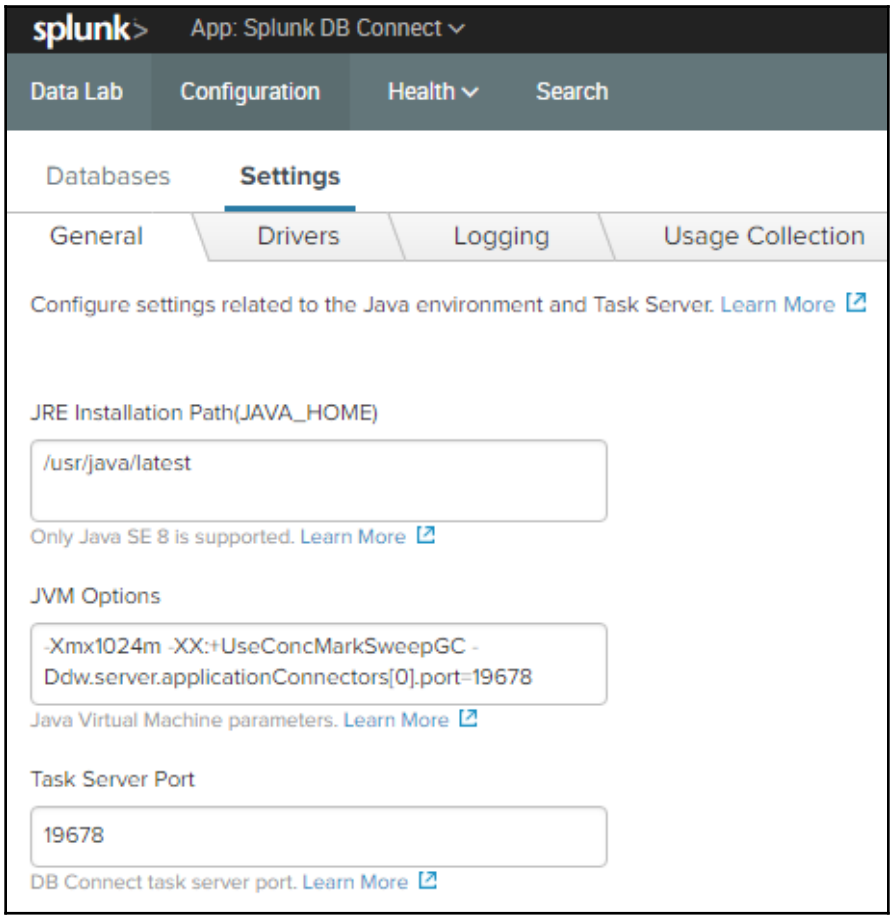


Fig 9.9: DB Connect general setup

Database drivers

Clicking the **Configuration** | **Settings** | **Drivers** tab should reveal a list of the database drivers that you installed in previous steps, and their version. If this doesn't occur, click the **Reload** button. If that doesn't resolve the issue, verify that you have the correct (`.jar`) drivers installed in the `/drivers` folder. There is no real configuration to be set up here, just a confirmation that the drivers are present and recognized, as shown in the following screenshot:

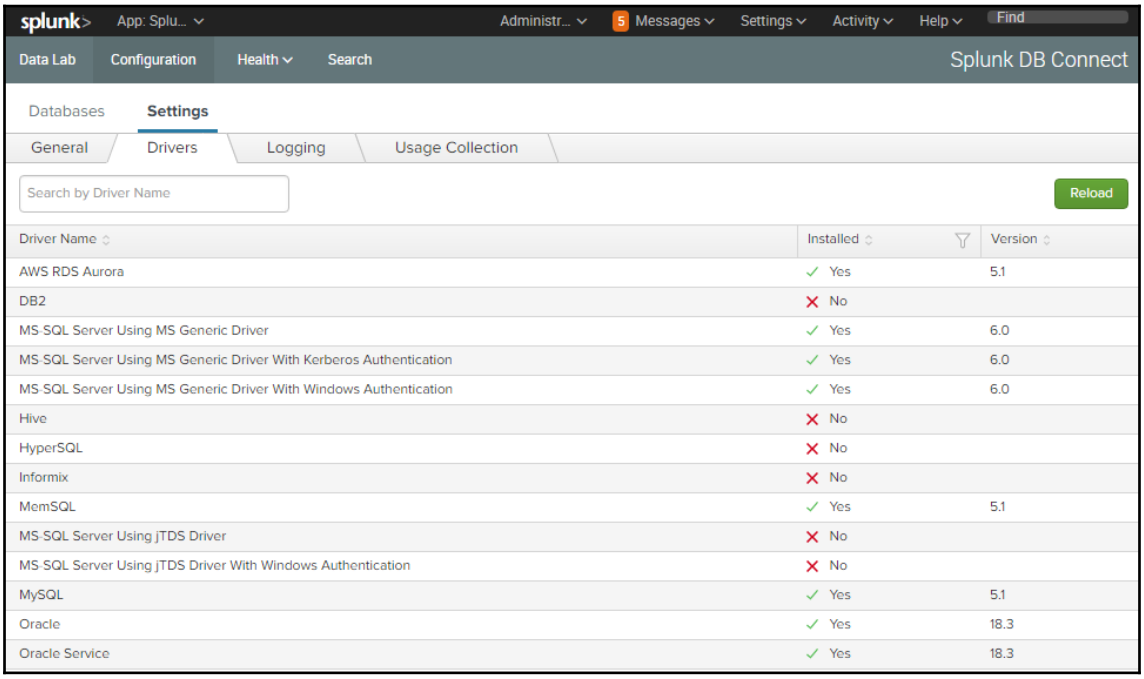


Fig 9.10: DB Connect drivers page

Configuring database input

Now that DB Connect is installed and configured, you're ready to set up an **input** from one of your databases.

This process includes configuring an **identity**, which is a username and password for accessing the database (not the same as your Splunk credentials), and a **connection**, which uses the selected identity along with the host, port, and other settings to connect to a particular database.

Identities and roles

The first step in setting up DB Connect for any purpose is to create the identity; you'll need to get a **username** and **password** that will allow you to access the target database from your DBA before proceeding. You can do this by going through the following steps:

1. Click **Configuration** | **Databases** | **Identities**
2. Click **New Identity**
3. Give the **Identity** a unique **name** (which you'll reference when setting up **Connections**).
4. Enter the **Username** and **Password** that had been previously configured for granting access to a particular database.
5. (Optional) If this is a Microsoft SQL Server instance using Windows AD, click the box and provide the **Domain** information as well.
6. Click the **Permissions** tab. You'll see that Splunk DB Connect provided two additional user roles when you installed the app: `db_connect_admin and db_connect_user`. These provide admin and read-only permissions, as you would expect from their names; you can give the normal Splunk admin role and the `db_connect_admin` role both **Read** and **Write** permissions, and the `db_connect_user` **Read** access only.
7. Click **Save**; you'll be returned to the **Databases** | **Identities** page, where you should see your new **Identity** listed. Note that you can **Enable**/**Disable** this **Identity** from this page, and alter the **Permissions**.

An example of the **Identity Settings** and **Permissions** forms is shown in the following screenshot:

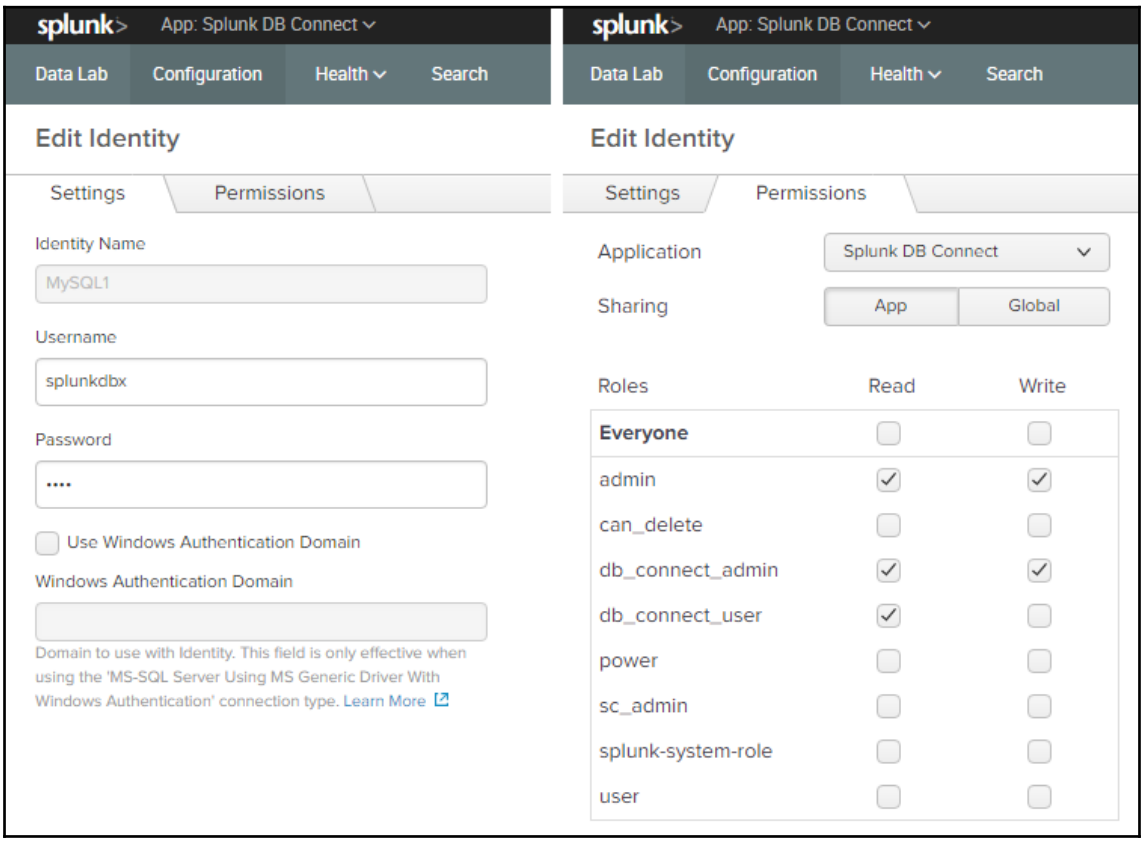


Fig 9.11: DB Connect identity settings and permissions

Connections

The next step in setting up DB Connect is to create at least one **Connection**. You'll need to obtain the **host**, **port**, and **default database** information for the database you're going to be connecting to from your DBA before proceeding with the following steps:

1. Click the **Databases** | **Connections** tab and then click **New Connection**.
2. Give the **Connection** a unique **Name** (you'll reference this when setting up **Input**, **Output**, or **Lookups**).
3. Select the previously-configured **Identity**.
4. Select a **Connection Type** from the list of database types, such as **MS SQL Server**, Oracle or MySQL.
5. (Optional) Set a **Timezone** to apply to the **Timestamp** fields (or allow the JVM timezone setting to be applied)—this might apply if your timestamps don't include a TZ indicator and your DB Connect instance is in a different timezone than your database.
6. Complete the fields for the **Host** (or IP address) and the **Port** used to access the database.
7. Select the **Default Database**.
8. The **SSL** and **Read Only** settings are self-explanatory; if you're only going to use DB Connect for input and lookup queries, you might want to click the **Read Only** box; if you're going to write to tables using the **Output** feature, then this will need to be left unchecked.
9. (Optional) Note that as you completed the host- and port-default database entries, the JDBC URL preview was populated with the combined URL using colons (`:`); in some cases, the query may require the use of slashes or other notation—if this is the case (your DBA can tell you), you can click the **Edit JDBC URL** checkbox and manually alter the URL in this field as needed.
10. Click **Save.**

Your new **Connection** will now be listed on the **Connections** page. You can **Enable**/**Disable** this **Connection**, and edit the **Permissions** as needed.

An example of the form entries you'll need to complete for setting up a **Connection** are shown in the following screenshot; I've altered the image to place the form fields side by side to save space:

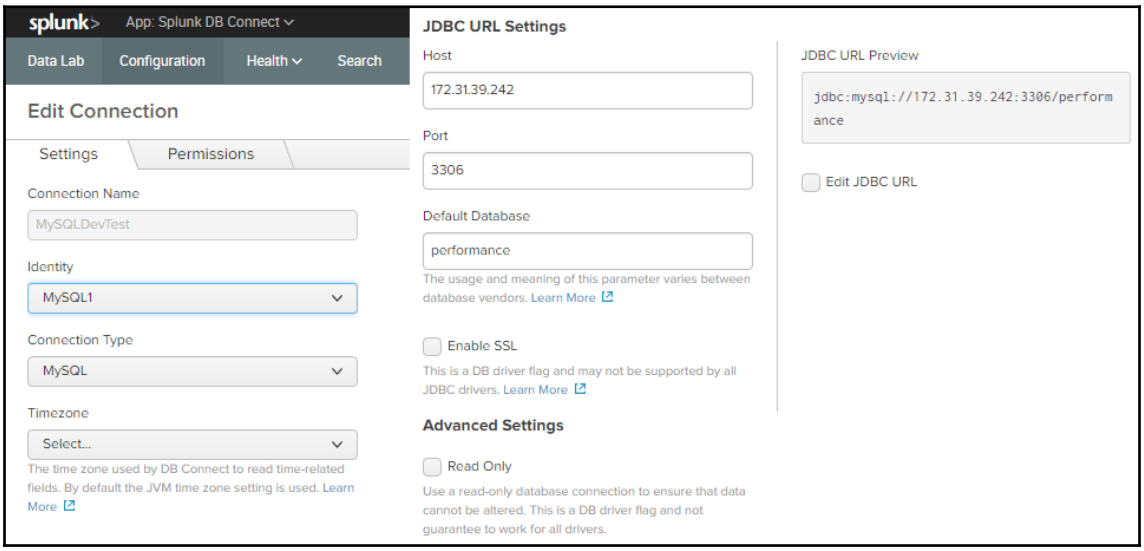


Fig 9.12: DBX Connection settings

When you clicked **Save**, DB Connect will confirm that the settings are correct by connecting to the database and running a version query; if this fails, you'll obviously need to troubleshoot any problems before proceeding. Note that your DBA may need to configure the database to accept a connection from your `'username'@'<your splunk hostname>'` or similar settings on the databases before you can successfully connect. On MySQL, for example, you may also need to add or edit a bind-address entry in `/etc/my.cnf` on the database server to allow a remote connection (preferably from a specified IP address, as follows):

```
bind-address = 0.0.0.0
```

Input

Most often, you'll be using DB Connect to poll databases to fetch data to ingest in Splunk using an **Input**—let's set one up. You can look at the screenshot after the following steps to see how they were applied:

1. Click **Data Lab** | **Inputs** | **New Input**. You'll arrive on the **Set SQL Query** page.
2. Select a **Connection** (**MySQLDevTest**).
3. Select a **Catalog** (performance).
4. Select either a **Schema** or **Table** (`webbytes`, for this example).
5. Splunk will create a generic SQL query (or you can edit/create one yourself in the **SQL Editor**)—click **Execute SQL** to run it. The rows and columns that are returned from the database are used to support and continue the configuration.
6. Select an **Input Type** of either **Batch** or **Rising**. Batch input is a bulk download of all of the table entries, while the rising column is an incremental download; you'll typically want to use the rising column method to avoid duplicate records being ingested.
7. If you select **Rising**, you must select a **Rising Column** containing a value that DB Connect records to keep track of the last record it fetched; this should be either a timestamp with millisecond resolution (to avoid missing newer records that were stored within the same second), or a unique auto-incrementing integer value *that doesn't roll over* (to avoid a halt in fetching new records if the value suddenly drops below the last recorded value when it rolls over).
8. (Optional) You can also provide a **Checkpoint Value** (a recent epoch value, for example) to avoid fetching *all* the older records if you only want to pick up new records going forward.
9. Select whether to use the **Current Index Time** for a timestamp for new records that are indexed or **Choose a Column** to obtain a timestamp from.
10. If you chose **Rising** Column, Splunk will provide a template of how to modify the SQL query to support the rising column factor. You will need to make the recommended modification and click **Execute SQL** to verify that it works before DB Connect will allow you to proceed.
11. When this is all working, click **Next**, which will take you to the **Set Properites** page**.**

An example of the settings for the **Set SQL Query** page is shown as the following screenshot:

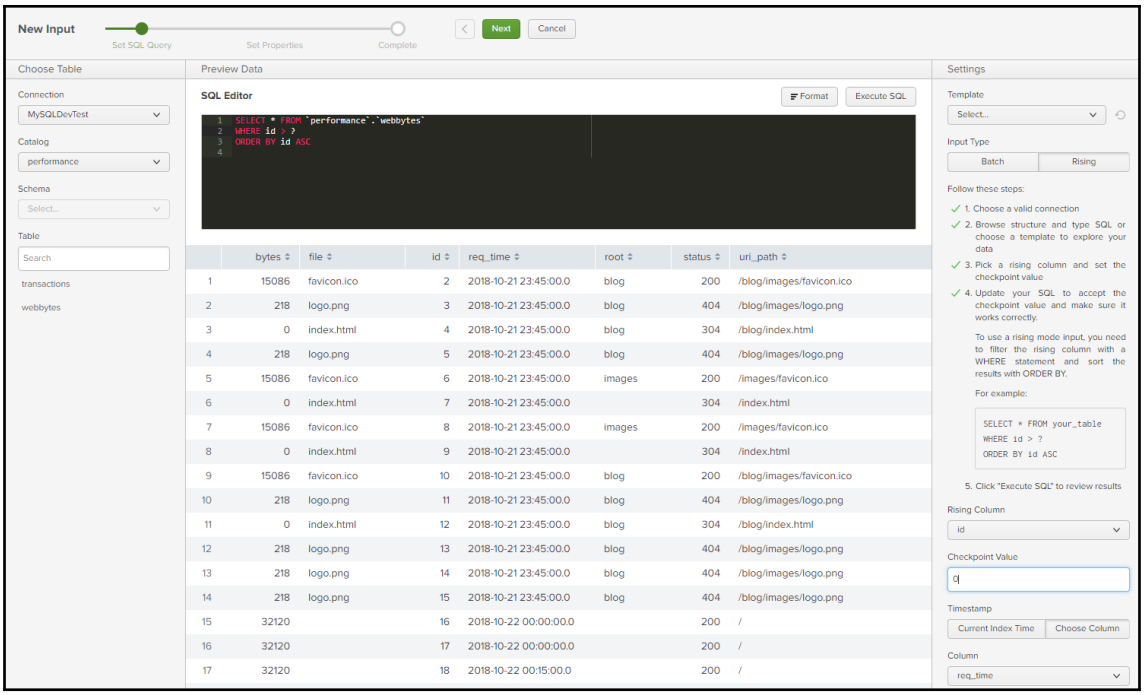


Fig 9.13: Set SQL Query page

On the **Set Properties** page, you configure settings that tell Splunk how often to perform queries and what to do with the data:

1. Input a **Name** and **Description**.
2. Set the **Execution Frequency** in seconds or use a cron entry. The execution interval should not be set too short or DB Connect won't be able to keep up, especially if you have a lot of input running. A setting of every 2 minutes (120 seconds) to 5 minutes (300 seconds) is a good minimum, but it should be set for longer periods if possible.
3. (Optional) Enter values for **Max Rows to Retrieve** and **Fetch Size**—the defaults are usually fine.
4. (Optional) Enter a **Host** if you want to specify something other than the host that is specified in the connection; I usually leave this blank.
5. (Optional) Enter a **Source** parameter; if you leave this blank, the input name will be used, which is usually preferable.
6. Enter a **Sourcetype** that reflects the type of database and the table the data comes from—this is a value that you make up, such as `mysql_webbytes`.

7. Select an **Index**. If you're running DB Connect on a heavy forwarder, deployer, or other Splunk node that doesn't have any way of knowing what all the configured indexes are, you can enter the index name manually—just make sure you type in the name correctly, and that the index actually exists before you set up the input! otherwise the events will go to whatever the default index is for your environment (such as `main`).
8. Click **Finish**. You will be returned to the **Input** page and your new input should be listed. Again, you can **Enable/Disable** the **Input** and set the **Permissions** as needed.

The following screenshot shows an example of the entries used in the **Set Properties** page:

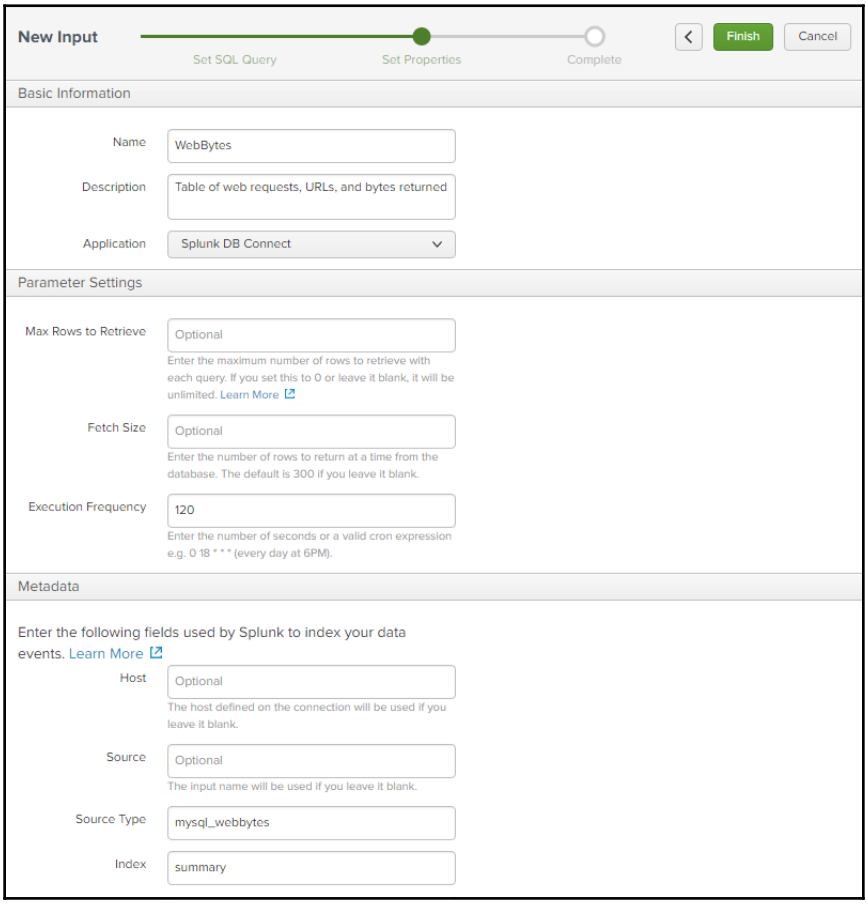


Fig 9.14: Set properties page

You can confirm that the input is working by performing a search on the appropriate index and sourcetype; note in the results, the `source` is the name of your input (`WebBytesInput`, in this example), and the `sourcetype` is as you named it during configuration as shown in the following screenshot:

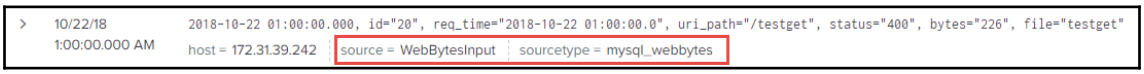


Fig 9.15: Indexed DB Connect input record

Output

In this very brief example of configuring an **output**, we're going to search our weblog's index for the sizes (in bytes) of files that have been served up for user requests and store them in a webbytes table of a performance database running on MySQL. As a side note, this is how I populated the webbtyes table to use for the previous input configuration examples, so please don't be confused by the similarity.

Note that if you are going to use the output feature of DB Connect, the node that it is running on must be capable of running searches. You can add this ability by editing `$SPLUNK_HOME/etc/server/local/server.conf` to add a `[clustering]` stanza that will enable a search head mode and point to the cluster master to obtain a list of the search peers (indexers) as per the following example. Note that the `pass4SummKey` entry contains the password for accessing the cluster master:

```
[clustering]
master_uri = https://172.31.18.102:8089    # Cluster Master IP:Port
mode = searchhead
pass4SymmKey = !Sp1unkCM!
```

To create an output, go through the following steps:

1. Click **Configuration** | **Outputs** | **New Output**
2. In the **Set up Search** page, create an SPL search string
3. Select a **Time Range** (**Last 15 minutes**, for example).
4. Click **Next** to go to the **Choose Table** page.
5. Select a **Connection** (**MySQLDevTest**).
6. Select a **Catalog** (performance).
7. Select a **Schema** or an entry under the **Table** list (**WebBytes**, in this example).
8. Click **Next** to go to the **Fields Mapping** page.
9. Select the **search fields** and the **table columns** that the data should go into in the database.
10. Click **Next** to go to the **Set Properties** page.
11. Give the output a **Name** (**WebBytes**, in this example) and **Description**.
12. If you want to schedule the output, click **Schedule this output** and enter an **Execution Frequency** in seconds or a valid cron expression, such as ***/15 * * * *** for every 15 minutes.

The following DB Connect output SPL example that ensures the **Bytes** field has a valid number and converts the web log timestamp (22/Oct/2018:01:00:00 +0000) into a format that can be easily understood by the database (2018-10-22 01:00:00); we then use the `table` command to convert and format the output from the events into multiple records with fields that can be sent to the database as shown in the following code:

```
index=weblogs_90d_eidx uri_path=* bytes=* req_time=*
| replace "-" with 0 in bytes
| eval req_epoch = strptime(req_time,"%d/%b/%Y:%H:%M:%S")
| eval req_datetime = strftime(req_epoch, "%Y-%m-%d %H:%M:%S")
| table uri_path status bytes root file req_datetime
```

You can verify that your output is working by inspecting your database table for new records from your Splunk data, and you can confirm that the output jobs run (or troubleshoot any issues) by inspecting the Splunk logs using the following code:

```
index=_internal sourcetype=dbx_server
```

The following screenshot shows an example event that reflects a successfully completed DBX output job:

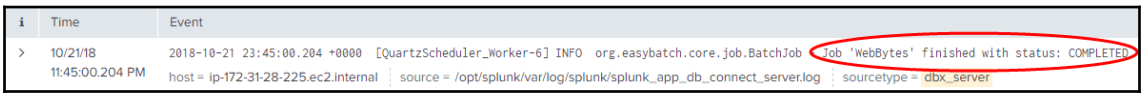


Fig 9.16: Completed output job log entry

Lookups

Lookups are very useful for enriching data from Splunk searches with information that may be residing in enterprise databases—the opportunities are enticing. You can create a lookup by clicking **Data Lab** | **Lookups** | **New Lookup**. You start on the **Set Reference Search** page by creating an initial search and clicking the magnifying glass icon or pressing *Enter*; when the results appear, click **Next** and you should see the following code:

```
index=weblogs_90d_eidx  | table uri_path status
```

On the **Set Lookup SQL** page, select a **Connection**, **Catalog**, and **Schema** or **Table**, and create an SQL query that will return the results from the table that you want to use in your lookup. In this example, assume that we have an http status table containing HTTP status codes and a `shortdef` and `longdef` field. The query is given here; click the **Execute SQL** button, and when the results come back, click **Next**. You should see the following code:

```
select * from performance.httpstatus;
```

On the **Field Mapping** page, select a **Search Field** and map it to the corresponding **Table Column** field in the database table to do the lookup on. as shown in the following screenshot. Then select the **Table Columns** you want to return from the lookup, and assign them an Alias you want to. Then click **Next**:

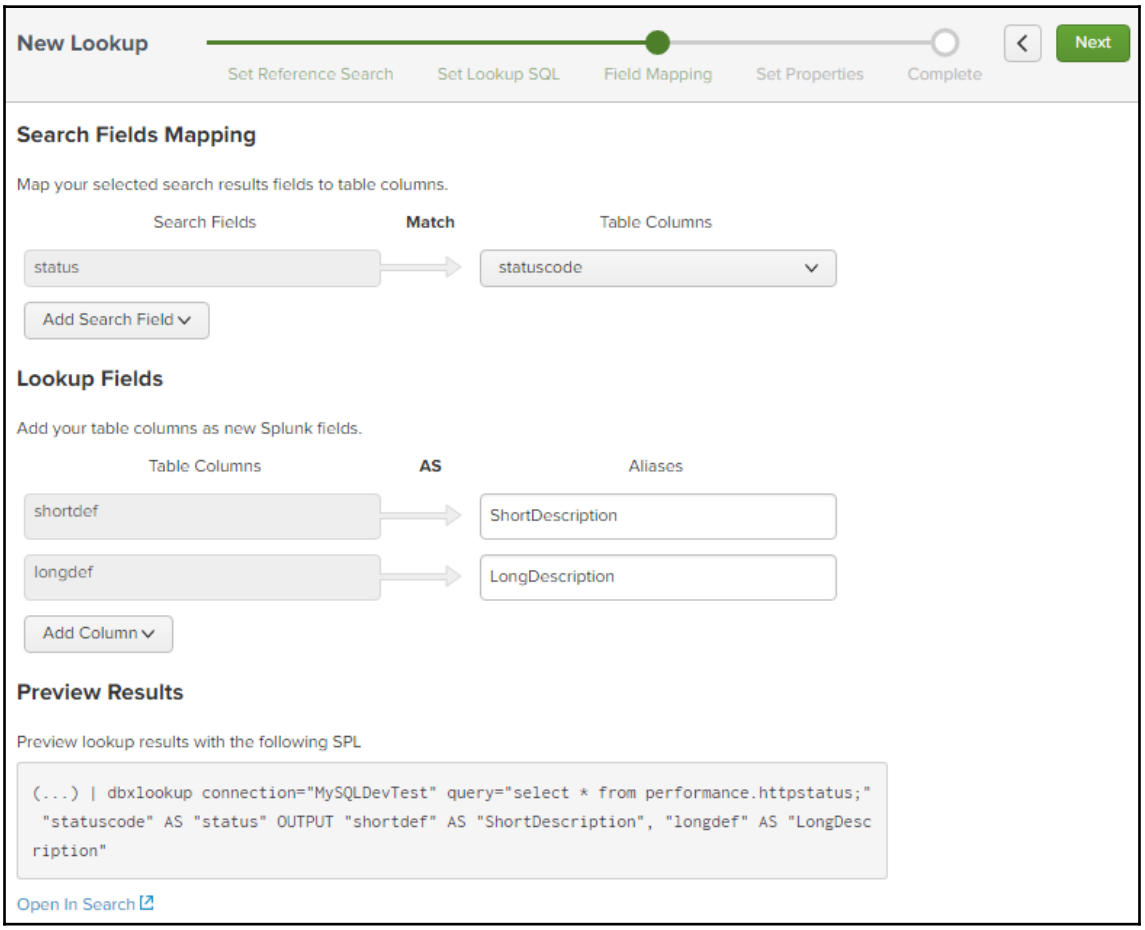


Fig 9.17: Lookup field mapping

On the **Set Properties** page, give the lookup a **Name** and **Description**. In the **Summary** section, you will be provided the SPL to use with this lookup, as shown in the following code:

```
| dbxlookup lookup="StatusCodeLookup"
```

You can apply the example to a search that uses the lookup, as shown in the following code:

```
index=weblogs_90d_eidx
| dbxlookup lookup="StatusCodeLookup"
| rename uri_path as "URI Path", status as Status
| table "URI Path" Status ShortDescription LongDescription
```

The looked-up fields will be added to the list of usable fields in your search, as shown in the following screenshot:

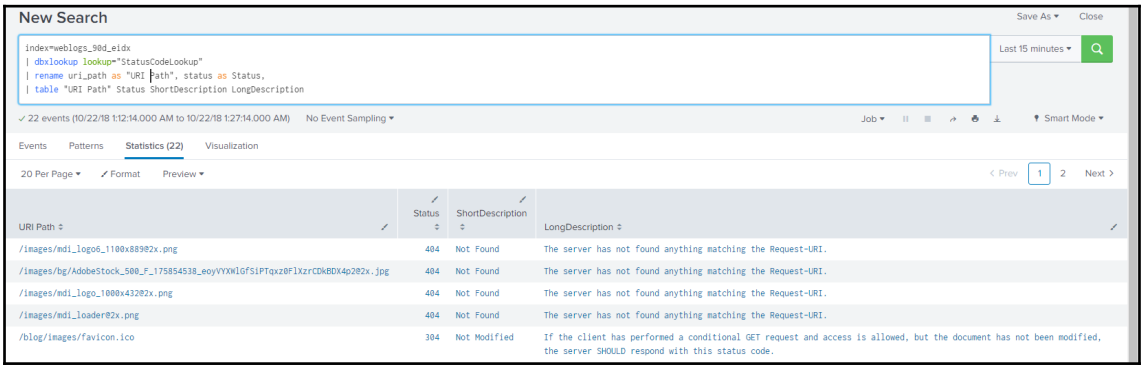


Fig 9.18: Performing a search using a DB Connect lookup

Troubleshooting DB Connect

Troubleshooting DB Connect issues is usually a matter of working through and noting the messages on the various configuration pages, and/or looking for errors or other hints in the DBX logs, which are indexed with the following sourcetypes:

- `dbx_audit`
- `dbx_connection_pool_metrics`
- `dbx_health_metrics`
- `dbx_server`
- `dbx_server_access`

A good starting-point search just looks for errors, as shown in the following code:

```
index=_internal sourcetype=dbx* error
or most often:
index=_internal sourcetype=dbx_server error
```

HEC port conflicts

DB Connect uses the **HTTP Event Collector** (on its default port of `8088`) to index data received from inputs. If this port is already in use, you may see errors in the DBX logs indicating that there is a problem when trying to write records and a non successful HTTP status code:

```
... ERROR org.easybatch.core.job.BatchJob - Unable to write records
java.io.IOException: HTTP Error 403: Forbidden ...
```

If this appears to be an issue, you can change the port by clicking **Settings** | **Data inputs** | **HTTP Event Collector** | **Global Settings** and trying another port. The changes will be stored in `$SPLUNK_HOME/etc/apps/splunk_httpinput/local/inputs.conf`.

The DB Connect manual has a troubleshooting section that may be helpful for database-related issues as well.

Splunk Premium apps

Splunk provides a number of Premium Apps, which are sophisticated value-added solutions tailored to IT operations, security, and Internet of Things environments. These are paid solutions in that they require the purchase of an additional license; you will need to talk to a Splunk salesperson to get pricing. The following is a partial list of Premium apps; I'll briefly introduce a few of them:

- **Splunk IT Service Intelligence** (**ITSI**)
- **Splunk Enterprise Security** (**ES**)
- **Splunk User Behavior Analytics** (**Splunk UBA**)
- **Splunk App for PCI Compliance – Splunk Enterprise Security**
- **Splunk App for Microsoft Exchange**

IT service intelligence

Splunk **IT Service Intelligence** (**ITSI**) is a monitoring and analytics app that provides actionable insight into the performance and behavior of your IT services. ITSI utilizes machine learning to dynamically baseline the normal behavior of selected **key performance indicators** (**KPIs**) from the data you have ingested into Splunk, detect and alert you when operational metrics fall outside of a normal threshold, and leverage a **predictive analytics** feature to predict service outages so that corrective action can be taken before your end users are affected. The key components of ITSI are as follows:

- **Entities**: These correspond to IT assets that can be tracked with a metric, such as a service, device, app, or process.
- **KPIs**: These are metrics for entities from a search, such as CPU load, memory, or response times.
- **Services**: These are a group of KPIs that generate health scores, perform root-cause analysis, and receive alerts.

The IT Service Intelligence application offers a number of powerful features; the following are a few of the most commonly used ones:

- **A Service Analyzer view**: Displays the health scores of your various IT services in color-coded tiles. When a tile turns yellow or red, you can drill down into those metrics to investigate the source of the discrepancy.
- **Glass Tables**: Visualizes and monitors interrelationships and dependencies across IT and business services. You can design a very appealing high-level view (summarized metrics displayed over images or icons) of the overall health of your services, derived from various service KPIs, health scores, or the results of ad-hoc searches, with a drill-down capability called **deep dives**, which lets you view multiple KPIs to speed troubleshooting.
- **Correlation Searches**: Identifies patterns across multiple data sources and generates alerts when notable events occur.

ITSI is a fairly complex application to install, configure, and maintain; you should have a decent amount of Splunk knowledge and experience, take the training class, and/or ask for assistance from a Splunk consultant before implementing it. You can learn more about ITSI from the details link of the Splunkbase app page at `https://splunkbase.splunk.com/app/1841/`.

Enterprise security and UBA

Splunk **Enterprise Security** (**ES**) is an advanced analytics-driven **security information and event management** (**SIEM**) solution that can serve as the nerve center of a security ecosystem, giving teams the ability to discover, monitor, investigate, respond, and report on threats, attacks, and other abnormal activity found across the enterprise. It simplifies threat management, eases alert overload, and provides executives with a window into their business risks from security concerns.

Much like ITSI, ES continuously monitors the security-related data stored in Splunk Enterprise to visualize a security posture with dashboards that feature key security indicators that utilize static and dynamic thresholds and trending information. You can optimize, centralize, and automate incident-response workflows with alerts and predefined reports and correlations; conduct investigations using ad-hoc searches and correlations to detect malicious activities; and conduct multistep investigations using trace activities on compromised systems, and apply what is called the kill-chain methodology (a military-derived model to identify, locate, and disable an attacker) to reduce or eliminate security threats.

ES can be integrated with and leverage Splunk's **User Behavior Analytics** (**UBA**) app, which is a machine-learning-driven solution that helps identify threats and anomalous behavior across users, devices, and applications. It identifies threats and provides risk ratings and supporting evidence by leveraging machine learning algorithms for anomaly-detection and correlation, and provides visual mappings of these anomalies over various phases of an attack life cycle. ES and the UBA product work together by sharing anomalies and threats to allow threat-incident assessments, risk scores, notable events, workflow management, and automated responses.

As with ITSI, Splunk ES and UBA are premium security solutions requiring a paid license and a fair amount of both Splunk and security expertise. You can get more information from their respective web pages and product briefs at the following links:

- `https://www.splunk.com/en_us/software/enterprise-security.html`
- `https://www.splunk.com/en_us/software/user-behavior-analytics.html`
- `https://www.splunk.com/pdfs/technical-briefs/splunk-for-advanced-analytics-and-threat-detection-tech-brief.pdf`

Summary

We've covered a *lot* in this chapter, but I think and hope that this coverage of how to create Splunk apps install and configure some of the free apps available from Splunkbase, configure permissions, and understand where and how these apps are manifested in the Splunk directory structure will help you to solidify your overall knowledge of how Splunk works. I especially hope you will install and configure the Add-on for your particular operating system(s), and work with DB Connect and the MLTK, as these are both very useful and valuable additions to your Splunk deployment.

In the next, and final, chapter, we'll cover how to monitor and troubleshoot your Splunk deployment, and introduce a few advanced topics.

10
Advanced Splunk

This chapter will serve as a brief overview and reference for several important topics and skills that any Splunk administrator will want to have in their toolkit. While Splunk is inherently stable and reliable, there will be times when you have to troubleshoot problems; this chapter covers the most useful Splunk logs and tools for determining what's working and what isn't. Then, we'll segue into using the Monitoring Console to keep tabs on Splunk health, as well as providing examples of searches that can be built for monitoring disk and index usage versus configured capacity, search performance, and several other factors that an administrator will be interested in. As a finale for this chapter, and book, the reader will be introduced to the essential concepts and references for taking Splunk to the next level—using API endpoints and the Splunk SDKs and frameworks for developing powerful customized solutions on top of the Splunk platform.

The specific topics covered in this chapter includes the following:

- Troubleshooting with Splunk logs
- Using the `btool` and `diag` utilities
- Searches to monitor Splunk performance and capacity
- Configuring and using the Monitoring Console
- Getting started with Splunk SDKs and the RESTful API

Before seeing you off to finish becoming a Splunk expert, I'll also provide some additional topics for further study. Let's go!

Troubleshooting Splunk

A Splunk Enterprise deployment is a fairly complex system, and while it is by nature a stable, reliable, and intuitive platform to work with, things can go wrong usually because of configuration or loading issues. We'll cover a few of the most common problems you'll run across in this section, and how to detect and resolve them. However, if you're going to be maintaining a larger Splunk environment, I highly recommend that you read through the **Troubleshooting Manual** and take the **Troubleshooting Splunk Enterprise** training class, as both do an excellent job of explaining more of the technical details of how Splunk works under the covers and the various logs and other sources of information to help with troubleshooting. Here are the links to these two resources:

```
https://docs.splunk.com/Documentation/Splunk/latest/Troubleshooting/Whatsin
here
https://www.splunk.com/en_us/training/courses/troubleshooting-splunk-
enterprise.html
```

Splunk logs

Splunk logs all of its activities in the `$SPLUNK_HOME/var/log/splunk/ and .../var/log/introspection` directories. Splunk ingests its own activity logs and stores them in the `_internal` (for 30 days) and `_audit` (6 years) indexes; resource usage, storage data, and KV store performance are indexed in `_introspection` for 14 days.

The `_internal` index and `splunkd` sourcetype is your go-to place for most troubleshooting. If you run a search for the sourcetypes and sources (source log files) in this index, you'll get a hint about the spectrum of information stored there; the most common sources are `metrics.log` and `splunkd.log` and messages with `log_level` of `ERROR`, `WARN`, or `WARNING`:

```
index=_internal | stats count by sourcetype, source
index=_internal sourcetype=splunkd source=*metrics.log
index=_internal sourcetype=splunkd source=*splunkd.log
index=_internal (log_level=error OR log_level=warn*)
```

`metrics.log` is where Splunk stores metrics information for various groups of splunk activities, such as throughputs, queue sizes, response times, and jobs count. These measurements are taken every 30 minutes, and by default report the top 10 results for each type or group so they are not all-inclusive. These groups include the following (this isn't a full list):

- `pipeline`: Metrics for the various code modules that are connected together to process and manipulate events flowing through Splunk. Group names include `indexerpipe`, parsing, typing, merging, `exec`, `dev-null`, and `fschangemanager`.
- `queue`: Metrics for the queues that serve as buffers between pipeline modules. Watch for `current_size=` values to identify queues that are full or blocked. Group names include `parsingqueue`, `tcpin_queue`, and `httpinputq`.
- `thruput`: Throughput values are reported in kbps, which means kilo*bytes* per second. Groups include `per_index_thruput`, `per_sourcetype_thruput`, `per_host_thruput`, and `per_source_thruput`. Ignore the `eps` values as they have accuracy issues.
- `tcpout_connections`: Output metrics for a Splunk component that is sending data to another module or system.

`splunkd.log` provides an overview of what `splunkd` is doing. Events are logged for all the Splunk components. You can get a list by running a search that lists of all the components in alphabetical order—you can look through this list to find components that may be related to the issue you're investigating and build filters accordingly:

```
index=_internal source=*splunkd.log | stats count by component
```

If you are working in a clustered environment, you can look for `ERROR/WARN*` events related to clustering components by adding `component=` filters for the following components:

- `CMMaster`: Cluster master functionality
- `CMPeer`: Cluster peer activities
- `CMBundleMgr`: Cluster bundle-related functionality for master and peers
- `CMRepJob`: Replication jobs and functionality
- `CMBucket`: Bucket-related activities
- `CMSlave`: General slave functionality
- `BucketReplicator` **(send side) and** `S2SFileReceiver` **(receive side)**: Bucket replication activities

The following search will help you get an overview of what is going on in your Splunk environment; the results display events with various ranges of log levels and messages. Set the **Time-Range Picker** for the last 5 or 15 minutes or so to limit the number of messages to something you can scan:

```
index=_internal source=*splunkd.log | table host component log_level
message splunk_server
and
index=_internal source=*splunkd.log (error OR warn*) | table host component
log_level message splunk_server
```

Here are some other logs of interest:

- `audit.log`: The information about user activity, such as logins and searches run (by search ID).
- `license_usage.log`: The indexed volume in bytes per license pool, index, source, sourcetype, and host.
- `remote_searches.log`: The information on indexers regarding searches they are participating in.
- `splunkd_access.log`: Any action done from `splunkd` through the UI is logged here, including `splunkweb`, the CLI, all `POST` or `GET` actions, deleted saved searches, and other programs accessing the REST endpoints, as well as the time taken to respond to the requests.
- `_introspection index`: This stores data about the splunk instance and operating system from the introspection logs. You can also get related information from REST endpoints.

btool

Splunk configuration files have many overlapping settings in many different locations; determining the final configuration that Splunk will use from the precedence-based merged values of all of these settings can be an arduous task. Splunk provides a CLI tool, called `btool`, in the `$SPLUNK_HOME/bin` directory which displays the merged on-disk configuration values from any of the .conf files in the system. Note again that `btool` displays the *on-disk* configuration file settings—if you change a setting and don't restart Splunk, btool will reflect the new settings, but that's not what will necessarily be running in-memory:

```
splunk help btool     # display usage and list of options
splunk btool check    # check all configurations for typos/errors
```

```
splunk btool <conf file name> list [options]    # conf file name w/o
'.conf'
```

For example, to get a list of all of the configured indexes and their configuration settings, perform the following:

```
./splunk btool indexes list
./splunk btool indexes list | grep '\['    # list just the stanzas
```

You can specify an app, user, directory, and various other parameters as options. To list the `savedsearches.conf` in the search app, do the following:

```
./splunk btool --app=search savedsearches list
```

Finally, you can ask `btool` to list all of the settings for a given stanza in a provided `.conf` file; in this example, we have a list of all of the configuration settings (across all of the `server.conf` files in *all* of the `/default` and `/local` directories throughout the system) for the `[clustering]` stanza in `server.conf`:

```
./splunk btool server list clustering

To locate all the server.conf files in Linux:
cd $SPLUNK_HOME
find . -name server.conf
```

diag

Splunk provides a `diag` utility that captures a snapshot of your Splunk configuration and conditions up to the point that the diag was run into a `tar.gz` file. If you engage with Splunk Support to help troubleshoot a persistent issue, they will likely ask you to run `diag` on one or more of your Splunk servers and upload the files to their support site.

To run diag, execute `$SPLUNK_HOME/bin/splunk diag`.

`diag` will provide messages of what it's getting as it runs and, when done, will let you know that it created a file in `$SPLUNK_HOME`; the filename reflects the host it was run on and the date-time, example:

```
Splunk diagnosis file created: /opt/splunk/diag-
ip-172-31-1-45.ec2.internal-2018-10-24_21-27-24.tar.gz
```

You can copy the `diag` file to another location and unzip it, which will create a directory of the same name:

```
$ tar xfzv diag*
```

You can then navigate through and inspect its contents:

```
composite.xml       # Splunk server processor configurations
diag.log            # Record of what files were captured
/dispatch           # Search artifacts
/etc                # Config files, apps, etc.
excluded_filelist.txt    # list of what was NOT collected
/introspection      # Introspection logs
/log                # Splunk logs
/_raft              # Current search Captain
splunk-7.1.1-8f0ead9ec3db-linux-2.6-x86_64-manifest    # installation
manifest
systeminfo.txt      # Long list of OS-level & Splunk system info
/var                # List of index directories (not the indexes
themselves)
```

`diag` collects all your configuration and apps files and logs. `diag` does *not* collect your `splunk.secret` file (used to obfuscate `pass4SymmKey` and other passwords) or any of the other files in `/etc/auth or /etc/password`, and it strips out any password hashes and salts in the `/etc/passwd` file. It also doesn't collect indexed data, although it does gather information about the configured indexes. Finally, diag provides a list of excluded files in `excluded_filelist.txt` and a record of what it collected in the `diag.log` file.

If `diag` takes too long to run, creates too large a file, or crashes, you can specify options to limit what it collects. For example, to just collect everything in the `/etc` directory before a Splunk upgrade, perform the following:

```
./splunk diag --collect=etc
```

You can review the Splunk documentation for all of the options available for the diag utility: `https://docs.splunk.com/Documentation/Splunk/7.1.2/Troubleshooting/Generateadiag`.

Opening a Splunk support case

You can open a support case with Splunk by logging in to your splunk account at `www.splunk.com`, clicking **Support | Support Portal**, logging in, and clicking **Submit a New Case**. Be prepared to provide information about your environment and a good problem description, attach any supporting documents or screenshots/images if applicable, and click **Submit**. You'll be assigned a **Case Number** and Splunk will be in touch with you via email, usually within a day or less, depending on the nature and priority of the issue.

Locked license issue

Note that if you somehow find you cannot search and are receiving error messages indicating a licensing issue, you can open a Splunk case to receive a reset key file that you can install on your license master, which will unlock your license and allow search activities to resume but I would escalate to your Splunk sales representative by phone and/or email for quicker service.

Performance and capacity

In this section, I'll provide a few searches that are helpful for interrogating some performance and capacity aspects of your Splunk deployment; these are in addition to a number of useful displays you can get from the Monitoring Console, which we'll cover in the 'Splunk monitoring console' section of this chapter shortly:

- **How quickly are indexers responding to search heads?** (run on a search head) This could identify a search that is poorly written. The results indicate the search ID (`search_id`), and you can match the values in the results to directory names in the `$SPLUNK_HOME/var/run/splunk/dispatch` directory if you want to inspect the search artifacts. Here's the search:

```
index=_internal source=*remote_searches.log
| stats max(elapsedTime) as MaxElapsedTime by server, search_id
| convert num(MaxElapsedTime) | where MaxElapsedTime > 1
```

- **What is the size of your search artifacts**? (run on a search head) If you are running out of allowable disk space usage in the `$SPLUNK_HOME/var/run/splunk/dispatch` directory or receiving errors that search artifacts are too large (2 GB by default) to send to the search peers (indexers), this can help identify the search(es) that are to blame. Search heads replicate their knowledge objects, such as fields, tags, and lookups, used in a search to the indexers to support search activities. You can match the excessive values in the search results from this search with directory names in the dispatch directory and go looking for lookup files or other knowledge objects in those directories that are of excessive size by leveraging the results of this search:

```
index=_internal sourcetype=splunkd_access method=GET jobs
| stats sum(bytes) as TotBytes by uri | convert num(TotBytes)
```

- **Am I receiving reports of skipped searches?** (run on search head) If you have more than the configured allowable saved searches scheduled to run simultaneously, some searches may be skipped. Use this search to start your investigation:

```
index=_internal sourcetype=scheduler status=skipped | eval scheduled_time =
strftime(scheduled_time, "%Y-%m-%d %H:%M:%S") | where skipped = 1 | table
app savedsearch_name search_type scheduled_time
```

- **How many concurrent searches are running and who are the users generating them?** (run on search head and view a Line Chart visualization) Use this search to identify the number and primary users behind your concurrent search counts:

```
index=_internal source=*metrics.log group="search_concurrency" | timechart
span=1m sum(active_hist_searches) as concurrent_searches by user
```

- **Index size and retention** (run on cluster master) You can see how much of your allocated index size is being used and the retention period using this search:

```
| rest /services/data/indexes
| table title maxTotalDataSizeMB currentDBSizeMB frozenTimePeriodInSecs
totalEventCount homePath
```

- **What is your disk usage versus capacity?** (run on a search head for SHs and cluster master for indexers) Splunk crashes, indexer detentions, and an inability to search are often related to disk exhaustion – you should keep an eye on this if disk space is getting low by using this search:

```
| rest /services/server/status/partitions-space
| eval pct_free=round(available/capacity*100,2), pct_used=round(100-
(available/capacity*100),2), capacity_gb=round(capacity/1024,0), gb_used =
round((capacity-available)/1024,0), gb_avail=round(available / 1024,0)
| rename title as "Partition #"
| table splunk_server mount_point "Partition #" capacity_gb gb_used
gb_avail pct_used pct_free
| sort pct_free
```

- **Check a user's authorization capabilities** (run on a search head) The combined capabilities across all roles for the given user can be obtained with this search:

```
| rest /servicesNS/<user>/user-prefs/authorization/capabilities | table id
capabilities
example:
| rest /servicesNS/guest/user-prefs/authorization/capabilities | table id
capabilities
```

- **When and why did Splunk crash?** You may want to inspect `splunkd_crash_log` to determine the cause of an unplanned outage - this search displays the events in the crash log:

```
index=internal sourcetype=splunkd_crash_log
| stats count by host
```

- **Determine when Splunk was last restarted** with this search**:**

```
index=_internal source=*splunkd.log "Splunkd starting"
```

- **ulimits and Transparent Huge Pages:** If you get error messages that suggest you're exceeding the allowable number of files or processes, you can display the configured per-host **ulimit** values and a check of the **Transparent Huge Pages** setting with the following search. The **Time Range** must include the period when Splunk was last started/restarted, as this is a one-time post-startup collection (use the preceding search to determine when the last start occurred):

```
index=_internal source=*splunkd.log component=ulimit
```

REST API endpoints

You may have noticed that, in some of the search examples in this book, I've used REST endpoints to get information. This is a very useful feature that you'll want to become comfortable with, as these endpoints sometimes provide an easier and in some cases, the only way to get some configuration information.

The best starting point for investigating REST endpoints is **`/services/properties`**; enter the following in the search bar:

```
| rest /services/properties
| rest /services/server
| rest /services/server/info    # lots of good HW info here

Try these servicesNS searches:
| rest /servicesNS/admin/search/data
| rest /servicesNS/-/-/data
```

The `/services` series provides access to system-wide information endpoints. The `/servicesNS` (namespace) series in the preceding code block requires that you provide an **owner** and **app** to access app-specific objects, or use the – wildcard for these entries where applicable.

You can use the `btool` utility to provide a complete list of all of the available REST endpoints on your Splunk server:

```
$SPLUNK_HOME/bin/splunk btool web list expose
```

You can also query a REST API endpoint on a local or remote Splunk server using its management port (`8089` by default) and `curl`, which is usually installed on Linux environments and can be installed on Windows platforms. From the command line, enter the following:

```
curl –k –u <userid>:<passwd>
https://<hostname_or_ip>:<mgmt_port>/<splunk_endpoint>
For example:
curl –k –u admin:Splunk1t2me https://18.210.79.20:8089/services/properties
To get results in json (or csv if supported):
curl –k –u admin:Splunk1t2me
https://18.210.79.20:8089/services/properties?output_mode=json
```

You will receive results in Atom format XML; as per the preceding example, you can also request results in JSON format or CSV if the endpoint supports it.

To obtain documentation for all of the Splunk REST API endpoints and their descriptions, as well as a tutorial on using REST to access (and update use with caution!) Splunk Enterprise configurations, see the following link: `http://dev.splunk.com/restapi`.

Splunk Monitoring Console

The Monitoring Console is a tool for viewing detailed topology and performance information for your Splunk Enterprise deployment. You can access the **Monitoring Console** (you must have admin privileges) by clicking the **Monitoring Console** icon in **Settings** on the node where the MC has been configured, which is usually (preferably) a Cluster Master(CM) with sufficient hardware resources (minimum of search head reference specs) to support both the master node and monitoring console demands. The CM automatically has the search capability needed to support MC functions, but without otherwise being a member of the search-head cluster.

If your MC has already been configured, you'll be taken to the **Overview** page, where you can get a quick status of your distributed environment, CPU and memory consumption, and a number of other function-specific metrics:

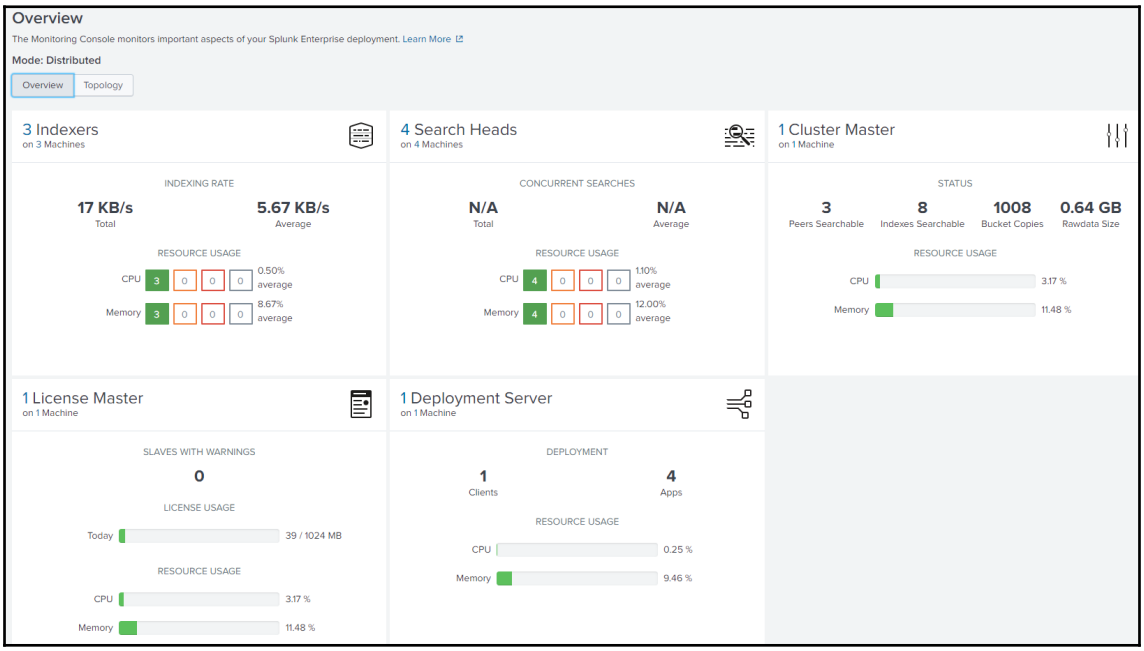


Fig 10.1: Monitoring Console Overview page

Clicking the **Topology** button takes you to another summary view that gives you a per-node status and metrics you can select from the drop-down above each category of servers, such as **Indexers**, **Search Heads**, and **Other**, which covers the other functions (**License Masters**, **Cluster Masters**, **Search Head** deployers, and **Deployment Servers**):

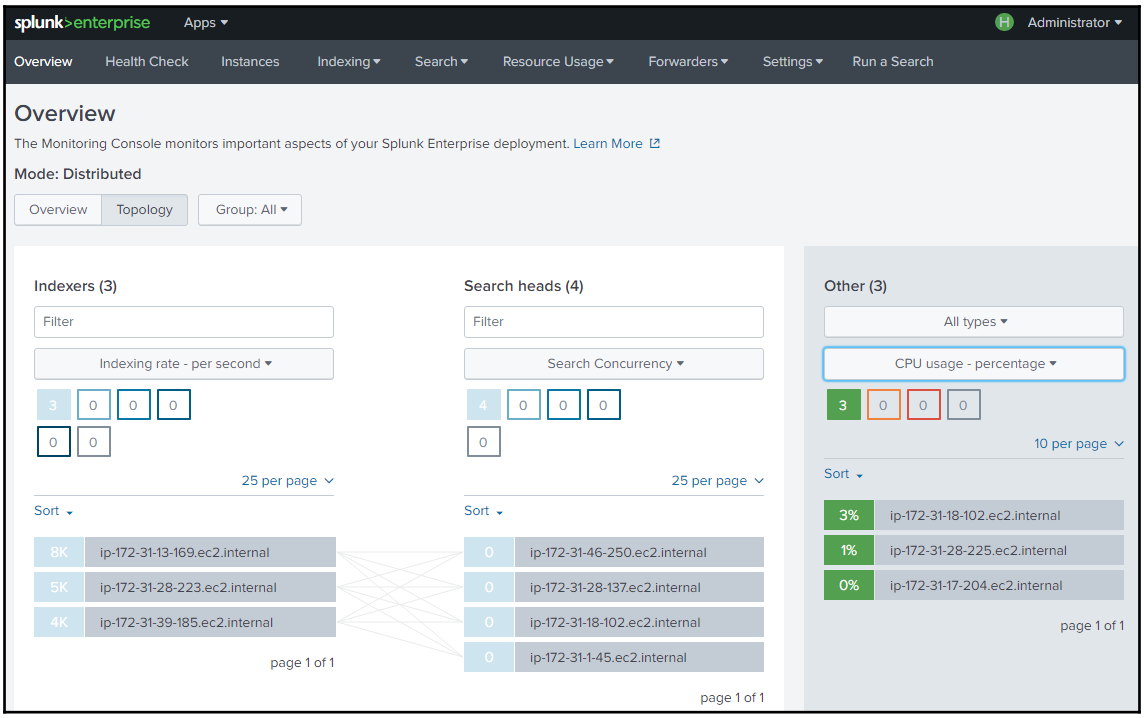


Fig 10.2: Monitoring Console Topology View

Configuring the monitoring console

If your Monitoring Console has not yet been configured, you should configure your **Distributed Search settings** first so that your MC can become aware of all of your Splunk servers. Click **Settings** | **Distributed search**, then **Distributed search setup**. Click **Yes** for **Turn on distributed search?**, then click **Save** and let Splunk restart.

Log in to the CM again, click **Settings** | **Distributed search** | **Search peers**. You should see your Splunk indexers listed. Now click **New Search Peer**, complete the **Peer URI**, **username**, and **password** fields, and click **Save** for each search head, your license master, deployer, deployment server, and any non-clustered indexers or other nodes in your environment that you want to be able to monitor. Note that the **Peer URI format** is `https://<servername_or_ip>:<mgmt_port> (usually 8089)`, and that the **Remote username** is a Splunk-authorized user with sufficient (admin or equivalent) rights to connect to the target node and perform API calls to the management port. When you are done, you should have a list of all of your Splunk servers, the **State** should report **Up** and the **Health status** should be **Healthy**. If you have configured your cluster labels for search heads and indexers, these should be reflected in the list as well; note that your search heads will probably reflect the indexers label, but they will be displayed properly in the MC pages. An example of the 'Add search peers' form is portrayed in the image below:

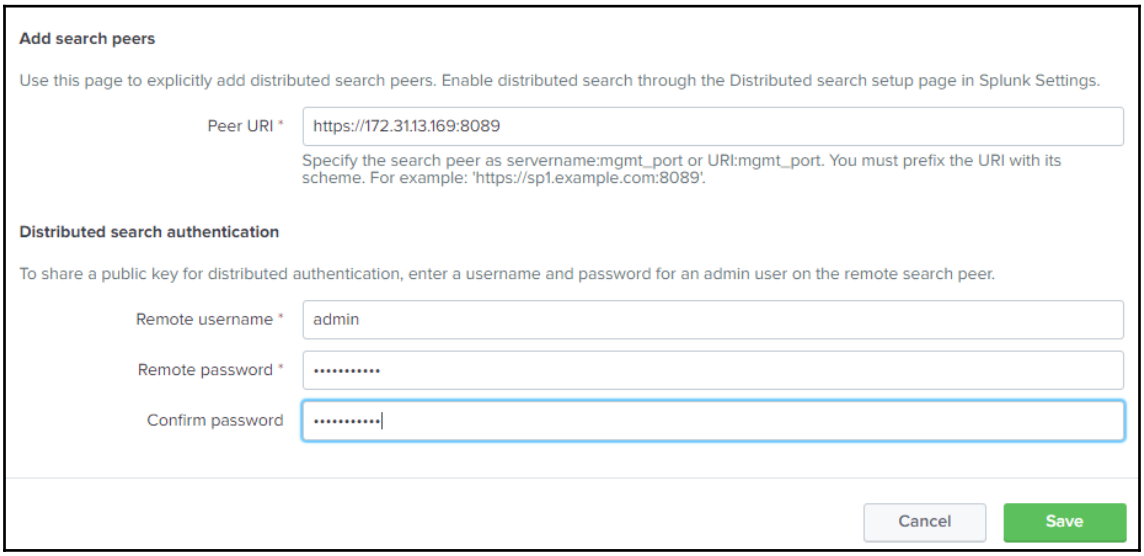


Fig 10.3: Adding search peers

You're now ready to configure the monitoring console itself. Click **Settings** | **Monitoring Console icon**, and then **Settings** | **General Setup** from the MC top menu bar. The default **Mode** will be **Standalone**; assuming you have a distributed environment, click the **Distributed** button, then **Apply Changes**. You should now see all of your Splunk servers listed; you may need to click **Edit** | **Edit Server Roles** and correct some entries.

Information about the distributed search peers and the categories/labels they belong to is recorded in the `distsearch.conf` file in `$SPLUNK_HOME/etc/system/local`, the `splunk_monitoring_console_assets.conf` file in `$SPLUNK_HOME/etc/apps/splunk_monitoring_console/local`, and the `assets.csv` and `dmc_forwarder_assets.csv` files in `.../apps/splunk_monitoring_console/lookups`. You can peruse or edit these files to correct entries if needed – see the Splunk docs on `distsearch.conf` and the monitoring console for more information.

Using the Monitoring Console

The monitoring console provides a wide range of performance- and capacity-monitoring tools – too many to cover in depth in a book this size, but we'll take a whirlwind tour so you're aware of what you might want to look into further:

- **Overview | Triggered Alerts**: At the bottom of the **Overview** page, you can enable various alerts that will inform you of critical issues with indexers, event-processing queues, memory, disk space, license usage, search peers not responding, and missing forwarders. You can edit the thresholds on most of these and, by clicking the **Advanced Edit** button, the **Edit** button next to **Actions**, and then the **Add Actions** button in the **Edit Alert** form, you can configure the alert to send you an email or other actions.
- **Health Check**: Clicking this menu option takes you to a page where you can click **Start** to run a series of checks across all of your Splunk Enterprise servers. These health checks are impressive in their coverage, and any discrepancies result in a warning or error icon appearing in the **Results** column; clicking on the icon provides a pop-up message explaining the nature of the issue and what steps to take to resolve it. You can click **Settings | Health Check Items** to disable or add other check routines.
- **Instances**: This provides a list of your Splunk servers, their roles, cluster labels, OS, CPU cores, RAM, Splunk version, and a **Views** drop-down menu under **Action**, which opens a new MC page to view Resource Usage, Search Activity, Search Usage Statistics, and information on the KV Store, as applicable.

- **Indexing**: This menu has numerous submenus to view dashboards related to indexer Performance, Clustering, Indexes and Volumes, Input, and License Usage. One dashboard that is particularly useful if you have a busy index cluster is the **Indexing | Indexer Clustering | Indexer Clustering: Status** view, as seen in the following screenshot; you can monitor the status of your indexers and whether the search and replication factors are being met:

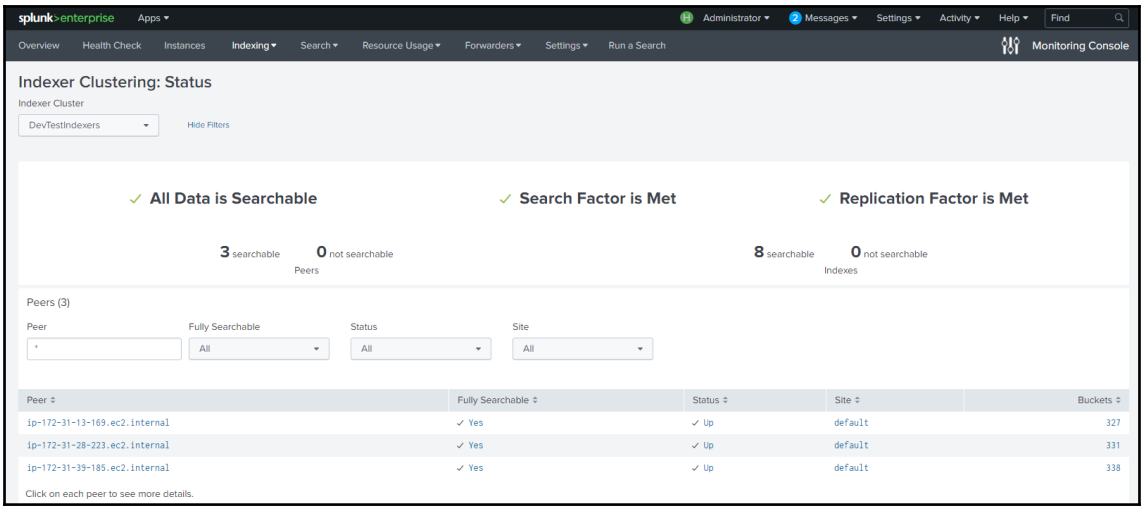


Fig 10.4: Indexer Clustering Status

Another interesting view is the **Indexing Performance: Instance** view, which provides a graphic view of the indexing pipeline components and the status of their queues. You should keep an eye on **Indexing | Inputs** for your applicable data input volumes in a busy environment, and **Indexing | Inputs | Data Quality** can give you insights into event-processing problems. In the Indexer Clustering: Status view portrayed in the image below, you can see that all data is searchable because the Search and Replication Factors have been met:

- **Search**: This menu option provides a series of submenus to view dashboards for search Activity, Distributed Search, Clustering, Scheduler Activity, and KV Store information; artifact replication and scheduler delegation are important search metrics to watch.

You'll notice by now that most of these submenu options allow you to view information on an **Instance** or **Deployment** basis and that, by scrolling down each dashboard page, you'll find a variety of graphs and tables with a wealth of status and performance information.

- **Resource usage**: This provides a view of physical resource usage across an instance, machine, or the entire deployment. The metrics reported include CPU, process class, memory, disk usage and disk I/O.
- **Forwarders**: If you have enabled **Forwarder Management** in the **Settings** | **Forwarder Monitoring Setup** page, you can view a list and the status and range of metrics for your Universal Forwarders, such as data volume and connection counts.
- **Settings**: This is obviously where you can configure various settings for the monitoring console; you've seen several of these, and the remaining are fairly self-explanatory.

Configuring and using the monitoring console (previously known as the Distributed Management Console, hence the reference to DMC in the URL) is another activity for which you simply must read through the manual to become aware of, and proficient with, all of its powerful features. Considering that studying all of the dashboards in the monitoring console provides a thorough overview of all of Splunk's functional components, it also serves as a very good training experience – it's well worth the time it takes to become more familiar with this powerful tool! Check out the following link for more information: `https://docs.splunk.com/Documentation/Splunk/latest/DMC/DMCoverview`.

Data rebalancing

Splunk creates primary buckets on the indexers receive data from the forwarders that are sending data to that indexer at the time, and distributes replicated buckets across an indexing tier based on bucket counts, not index size. Therefore, it is possible that over time, the disk usage across all, your indexers will differ significantly, and may cause some disks to get too close to full for comfort. If this is the case, you may need to perform a **data rebalance** across your indexing tier. This is performed from the Cluster Master.

Although you can execute and monitor a data rebalance from the Splunk CLI, I recommend doing so from the Cluster Master GUI so that you can more easily monitor the rebalance progress. Click **Settings** | **Indexer clustering**. In the **Indexer Clustering**: **Master Node** page, click **Edit** | **Data Rebalance**. You can adjust the **Threshold**, **Max Runtime**, and **Index** selections or leave them at their defaults, and click **Start**. You will see a progress bar appear at the bottom of this form. You can allow the rebalance to run until it finishes, or click the Cancel button to stop the rebalance; there is no harm in cancelling the operation and any progress made during the run will not be undone.

If you want to start/monitor/stop a rebalance from the CLI (on the cluster master), use these commands:

```
splunk rebalance cluster-data -action start [-index index_name] [-
max_runtime interval_in_minutes]
splunk rebalance cluster-data -action status
splunk rebalance cluster-data -action stop
```

Indexer clustering and bucket status

While you're in the **Indexer Clustering: Master Node** page, note that you can click on the **Peers**, **Indexes**, and **Search Heads** tabs to view the status of your distributed nodes and **Indexes**. In the **Peers** and **Search Heads** tabs, you can see the status (**Up/Down**) of your nodes and the number of buckets.

In the **Indexes** tab, you can view a bar-type display of the status of your **Searchable Data Copies** and **Replicated Data Copies** compared to the configured replication factors for both categories. In addition, there is a **Bucket Status** button you can click that will take you to a Bucket Status page.

There are three tabs on the Bucket Status page as well: **Fixup Tasks – In Progress**, **Fixup Tasks – Pending**, and **Indexes With Excess Buckets**. In the **Indexes** with **Excess Buckets** tab, you can click the **Remove All Excess Buckets** button or the **Remove** button for a select index.

Excess bucket copies are those that exceed the cluster's replication or search factor. They can result from peers leaving the cluster and then returning to it. When a peer goes down, the cluster master initiates bucket-fixing activities to compensate for any now-missing copies that were residing on that peer; the cluster master starts coping buckets across nodes to restore the replication-factor number of copies and a search-factor number of searchable copies. If the peer later returns to the indexer cluster, any bucket copies that the peer retained while down are once again available to the cluster. This can result in the cluster having excess copies of some buckets.

Excess copies do not interfere with the operation of the cluster, but they aren't needed and consume extra disk space, so if you observe a large number of excess buckets, you can safely remove them. I will note that if you often observe a high number of excess buckets, you should inspect your indexing cluster for reliability issues.

Upgrading Splunk Enterprise

To upgrade Splunk, perform the following steps:

1. Back up your indexed data
2. Back up your configuration: `splunk diag --collect=etc`
3. Stop Splunk.
4. Install the new version into the existing Splunk directory.
5. Start Splunk.
6. Answer **n** to the **Perform migration and upgrade without previewing configuration** *changes?* prompt, and preview the proposed migration changes on your first upgrade.
7. Review `$SPLUNK_HOME/var/log/splunk/migration.log.<timestamp>` after the first upgrade to ensure you understand what was done.

See the Splunk docs before your first upgrade; be sure to select the correct version you're upgrading to, and read through the **READ THIS FIRST** section before performing the upgrade: `http://docs.splunk.com/Documentation/Splunk/latest/Installation/HowtoupgradeSplunk`.

Splunk development

As we covered in `Chapter 9`, *Splunk Applications*, you can build apps and add-ons to add functionality to your Splunk investment. This discussion mostly focused on creating an app to serve as a container for configuration files and for adding reports and dashboards and perhaps a script to pull data from an outside data source. However, Splunk provides Software Development Kits (**SDKs**) for a variety of languages that, coupled with the Splunk REST API, allow you to further customize and extend the power of Splunk.

The gateway to Splunk development is the **splunk>dev** website, which offers a plethora of resources, tutorials, code examples, and reference material to help with your development project. Here are the links to the relevant topics:

- `http://dev.splunk.com/`
- `http://dev.splunk.com/sdks`
- `http://dev.splunk.com/restapi`
- `http://dev.splunk.com/python`
- `http://docs.splunk.com/Documentation/PythonSDK`

From the **dev.splunk.com** page, clicking the **Get started** menu link and then the **Overview** and **Quick Start** tabs will take you to an introduction about how to create Splunk apps and add-ons and about using the **AppInspect** tool to validate your app in case you want to upload it to Splunkbase for the rest of the Splunk community to use.

Clicking the **I want to** menu item offers tabs for **Develop and Release an app or add-on**; the **Integrate with the Splunk platform** tab takes you to an introduction of the Splunk REST API, SDKs, and developer tools. The **Extend premium features** tab provides links for developing custom modules for ITSI and integrations for Enterprise Security.

The **Resources** menu item has tabs for Tutorials and code examples. The **Downloads** tab provides links for several tools; all of the SDKs; logging libraries for Java, JavaScript, and C#; and IDE plugins for Eclipse and Visual Studio. The **Reference** tab provides links to web documents for the various tools, REST API, SDKs, and the **SplunkJS Web Framework**, which you can use to work with Splunk dashboards after you have converted them from Simple XML into HTML or you develop a dashboard or other UI from scratch.

Software Development Kits

The focus of this section will be on using the **Splunk SDKs**, which are available for Python, Java, C#, and JavaScript, as a layer on top of the REST API for accessing Splunk resources.

You can use the SDKs to do the following:

- Run Splunk searches from external applications.
- Create Splunk-like data visualizations (charts, tables, and dashboards) in other business tools.
- Build mobile applications using dashboards and alerts powered by Splunk.
- Send data directly to Splunk from remote devices and applications using API calls.
- Manage a Splunk environment, such as users, input, and indexes, from an application outside of Splunk.

Using the Python SDK

As an example of using the Python SDK, navigate to the `dev.splunk.com/sdks` page and click the button to download the Python SDK; this will provide a `splunk-sdk-python-x.x.x.zip` file that you can save in your development folder. Unzipping the file will create a similarly-named directory that contains folders for the library files, examples, and so on.

You will need to have a Python 2.7 environment set up on your development environment; Splunk only works with the 2.7 version of Python at present, and you'll want your code to be compatible in case you want to run it on a Splunk server instead of on a remote machine. You will need to set your `PYTHONPATH` environment variable to point to the Splunk Python SDK directory.

To work with the examples from the SDK, you'll also want to create a `.splunkrc` file in your home directory; this is discussed in the Splunk SDK for Python Utilities page – here are some examples for a Windows environment:

```
PYTHONPATH=C:\Dropbox\Splunk\Automation\SplunkSDK\splunk-sdk-python

Splunk SDK for Python Utilities (explains .splunkrc file)
http://dev.splunk.com/view/python-sdk/SP-CAAAEFC

Example .splunkrc file:
# Splunk host (default: localhost)
host=192.168.1.8
# Splunk admin port (default: 8089)
port=8089
# Splunk username
username=admin
# Splunk password
password=Splunk1t2me
# Access scheme (default: https)
scheme=https
# Your version of Splunk
version=7.0
```

Once you have your initial development environment set up, you can try one of the examples. Open a CMD window (Windows) or Terminal (Mac), navigate to the /examples folder of the Python SDK, and run the `index.py` file – you should get a list of all of the indexes configured on the Splunk server you pointed to in the `.splunkrc` file and how many events they contain:

```
C:\Dropbox\Splunk\Automation\SplunkSDK\splunk-sdk-python-1.6.5\examples>python -V
Python 2.7.13 :: Anaconda 4.3.0 (64-bit)

C:\Dropbox\Splunk\Automation\SplunkSDK\splunk-sdk-python-1.6.5\examples>python index.py
_audit (3669729)
_internal (7553149)
_introspection (1095511)
_telemetry (354)
_thefishbucket (0)
devtest (336)
history (0)
main (24048)
os_nix (158304)
splunklogger (0)
summary (0)
unix_summary (0)
web_services_90d (0)

C:\Dropbox\Splunk\Automation\SplunkSDK\splunk-sdk-python-1.6.5\examples>
```

Fig 10.5: Python SDK example-listing indexes

From this point, you can poke through the other example `.py` files and read through the reference material to get familiar with working with the SDK in a Splunk environment. In particular, you'll need to learn about the various **modules** (you'll mostly use the **client** module), the **Service class**, **Endpoint Entities** and **Collections**, and how to work with Splunk **Namespaces**; you'll recognize these as being defined by an *owner*, *app*, and *sharing mode* – just as we work with when setting up permissions for apps and knowledge objects.

The REST API

You will use the SDK for your chosen language with the REST API to access and manipulate Splunk resources by using GET and POST operations.

From the `dev.splunk.com/restapi` web page, you will find links to all three of the REST API reference documents, which include many `curl` command examples of using the API; you can test the endpoints with curl as you work to integrate them with your SDK code:

- **REST API reference manual**: This is a thorough coverage of all of the API endpoints, separated into categories such as Access, Application, Cluster, Input, and Search endpoints.
- **REST API user manual**: This is a short must-read of the basic concepts for working with REST APIs.
- **REST API tutorials**: This provides examples of using API calls to manage Splunk apps and knowledge objects, Splunk configurations, and creating and accessing searches.

As mentioned before, you'll want to get familiar with Splunk namespace parameters. These are manifested as **Extensible Administration Interface** (**EAI**)and **Access Control List** (**ACL**) `[eai:acl]` parameters when working with the REST API programmatically; you'll recognize these from when you've used a `| rest` call in a Splunk search and looked through the results, which is a good way to get familiar with how they're presented and used.

Again, here is the entry link to the API information: `http://dev.splunk.com/restapi`.

Additional study topics

If you plan to make Splunk part of your career path and goals, you'll want to expand your Splunk knowledge and experience by reading books (such as this one!), the Splunk docs, and getting some hands-on experience. You will also want to take as many of the Splunk training classes as you can afford or justify – they are excellent and essential for obtaining a more rounded knowledge base and those oh-so-important certifications.

The list of topics and documentation links provided offer good coverage for much of the Splunk landscape; I suggest you at least browse through these just to make yourself aware of all of the subjects – the real learning happens when you really need to look up and use a feature or configuration setting. After a while, it all starts to gel together and your confidence and efficiency both climb dramatically. Check these out; it's worth the time:

- **Splunk Community - the springboard to everything Splunk**
  https://www.splunk.com/en_us/community.html
- **Splunk Book**
  `https://www.splunk.com/goto/book`
- **Search and Reporting**
  `http://docs.splunk.com/Documentation/Splunk/latest/SearchReference/WhatsInThisManual`
- **Knowledge Objects**
  `http://docs.splunk.com/Documentation/Splunk/latest/Knowledge/WhatisSplunkknowledge`
- **Dashboards and Visualizations**
  `http://docs.splunk.com/Documentation/Splunk/latest/Viz/Aboutthismanual`
  `http://dev.splunk.com/view/dev-guide/SP-CkAAAE3C`
- **Splunk Admin**
  `https://docs.splunk.com/Documentation/Splunk/latest/Admin/Howtousethismanual`
- **Indexers and Cluster of Indexers**
  `http://docs.splunk.com/Documentation/Splunk/latest/Indexer/Aboutindexesandindexers`
- **Splunk Training & Certification Classes**
  `https://www.splunk.com/en_us/training.html`
  `https://www.splunk.com/en_us/training/free-courses/splunk-fundamentals-1.html`

Summary

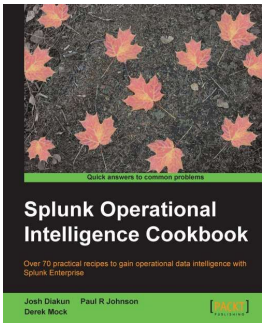
Well, that's all folks! I hope you found this last chapter enlightening, and I especially hope you'll use it as a reference guide for remembering and performing some of the more advanced troubleshooting and maintenance activities on your Splunk environment. And, in your spare time, dabble with the REST API endpoints and try out one of the SDKs.

Thank you very much for reading my book! I hope you found it worth your time. Splunk is one of the most exciting technology solutions in the industry, and it has a lot of potential for the future. I'm looking forward to watching and helping it develop; I trust you are too, and that this book helps you along your journey.

Happy Splunking!

Other Books You May Enjoy

If you enjoyed this book, you may be interested in these other books by Packt:



Splunk Operational Intelligence Cookbook

Paul R Johnson, Josh Diakun, Et al

ISBN: 978-1-84969-784-2

- Search, report on, and visualize operational intelligence data
- Enrich operational data with lookups and workflows
- Model and accelerate data and perform pivot-based reporting
- Build real-time, scripted, and other intelligence-driven alerts
- Summarize data for longer term trending, reporting, and analysis
- Build a fully featured Splunk operational intelligence application
- Integrate advanced JavaScript charts and leverage Splunk's API

Implementing Splunk 7 - Third Edition

James Miller

ISBN: 978-1-78883-628-9

- Focus on the new features of the latest version of Splunk Enterprise 7
- Master the new offerings in Splunk: Splunk Cloud and the Machine Learning Toolkit
- Create efficient and effective searches within the organization
- Master the use of Splunk tables, charts, and graph enhancements
- Use Splunk data models and pivots with faster data model acceleration
- Master all aspects of Splunk XML dashboards with hands-on applications
- Create and deploy advanced Splunk dashboards to share valuable business insights with peers

Leave a review - let other readers know what you think

Please share your thoughts on this book with others by leaving a review on the site that you bought it from. If you purchased the book from Amazon, please leave us an honest review on this book's Amazon page. This is vital so that other potential readers can see and use your unbiased opinion to make purchasing decisions, we can understand what our customers think about our products, and our authors can see your feedback on the title that they have worked with Packt to create. It will only take a few minutes of your time, but is valuable to other potential customers, our authors, and Packt. Thank you!

Index

A

alert
  creating 203, 205
Alerts 153
Amazon Web Services (AWS) 11

B

best practices, for administering Splunk
  about 121
  index naming conventions 122
  location of indexes.conf 123
  location of props.conf 123
  location of transforms.conf 123
  source type naming conventions 122
btool 248, 249

C

centralized streaming command 158
chart command 150, 151
cluster master
  about 57, 58
  configuration bundle, distributing 105, 107
  using 104
clustered Splunk environments
  versus distributed Splunk environments 28, 29
cold buckets 31
components, Splunk
  cluster master 16
  deployer 16
  deployment server 16
  heavy forwarder 17
  indexer 15
  license master 17
  search head 15
configuration bundle
  distributing 105, 107
Coordinated Universal Time (UTC) 21
cross-environment search 67, 68
CSS
  using, within dashboard 201

D

dashboard forms
  using 196, 197
dashboard
  CSS, using within 201
dashboards
  about 153
  creating 191, 192
  JavaScript, using within 201
  new panel, adding with inline search 194
  panel characteristics, editing 195
  performance, improving 200
data model acceleration 177
data models
  about 174, 175
  reference 177
  using, in search 176
data source inputs
  configuring 80, 81
data-rebalancing
  about 260
  bucket status 261
  indexer clustering 261
datasets 173
dedup command 146
deployer
  about 61
  using 109, 110, 111, 112
deployment client
  configuring 89, 90
deployment server, Splunk
  reference 94

deployment server
about 64
apps, creating 90, 91
configuring 90
serverclass.conf, creating 91, 93
using 87, 88
diag utility
about 249, 250
reference 250
distributable streaming command 158
distributed Splunk environments
versus clustered Splunk environments 28, 29

E

Elastic Block Storage (EBS) 36
Enterprise Security (ES) 243
eval command
about 143
reference 146
event type 168
event-handlers 202
extract fields interface
using 165, 166

F

field alias 170
field extractions
about 163
index-time field extractions 163
search-time field extractions 164
file precedence
configuring 52, 53
formatting commands
about 147
head command 148
rare command 149
rename command 148
reverse command 148
sort command 148
tail command 148
top command 149
forwarder management
using, in Splunk Web 94

H

head command 148
heavy forwarder
setting up 78, 80
hot buckets 31
HTTP Event Collector
configuring 81, 83, 85
testing 85

I

index cluster 63
index-time field extractions
about 163
reference 163
indexes.conf configuration, Splunk
references 97
information resources, Splunk
Splunk answers 22
Splunk documentation 22
Splunk education 22
inputs.conf options, Splunk
references 77
IT Service Intelligence (ITSI)
about 242
features 242
key components 242

J

JavaScript
using, within dashboard 201
join command 155, 156

K

Key Performance Indicators (KPIs) 242
knowledge objects
about 167
data models 174, 175
datasets 173
event type 168
field aliases 169
lookup 170, 171, 172
macros 172
pivot tables 177, 179
reference 167

tags 169

L

LDAP authentication 116
license master 57, 58
Linux settings
about 43
group, environment settings 43
transparent huge pages 44
ulimits 43, 44
user, environment settings 43
locked license issue 251
lookup
about 170
using 171, 172

M

machine learning toolkit
about 220
reference 223
macros
about 172
reference 173
media kit, Splunk
reference 9
metrics indexes 99, 100
monitoring console 17
multisite environments
about 65
cluster master 65
indexers 66
search heads 66

N

Network Time Protocol (NTP) 49
new apps
deploying 114

P

page selector 135
payment card information 26
per-page 135
personally identifiable information 26
pivot tables 177, 179
pivots
reference 179
predictive analytics 242
processing tiers, Splunk
Data Input segment 18
Indexing phase 19
Parsing segment 18
Search segment 19
products, Splunk
Splunk Cloud 11
Splunk Enterprise 10
Splunk Free 11
Splunk Light 11
Python SDK
using 264

R

reference guide, Splunk
reference 22
rename command 148
reports
creating 184, 185
scheduling 186, 188, 189, 191
REST API endpoints 253, 254
REST API
used, for accessing Splunk resources 266
rex command 146, 147
robotdev 21
roles
configuring 114

S

SAML Authentication 117
search bar 133
search command
about 138, 143, 154
dedup command 146
eval command 143
index 138, 139
join command 155, 156
rex command 146, 147
searches, optimizing 159
stats command 144, 145, 146
streaming command, versus transforming command 158, 159

subsearch 154
time-range selection 139, 140, 141
transaction command 156, 157
where command 147
search filters 141
search head cluster 32
search head
captain, designating 63
captain, initiating 63
cluster status, checking 63, 64
search heads 62, 63
search jobs
optimizing 159, 160
Search Processing Language (SPL) 15, 128, 129
search results
chart command 150, 151
fields command 149, 150
Splunk web, visualizing 153
table command 149, 150
timechart command 150, 152
visualizing 149
search-time field extractions 164
searches
ad hoc 27
data models, using in 176
job inspector 160, 161
optimizing 159
real-time searches 27
scheduled historical search 27
summary indexing search 27
Server Manager 48
servers
pointing, to license master 59
Simple XML
working with 198, 200
software as a service (SaaS) 11
Software Development Kits (SDKs)
about 263
Python SDK 264
sort command 148
Splunk Add-on
about 207
for Linux 215, 217, 218
for Unix 215, 217, 218
Splunk app
about 86, 87, 207
context 211
creating 208, 210
permissions 213, 214
reference 35
Splunk authentication 114, 116
Splunk Cloud 11
Splunk community
reference 22
Splunk components
about 52
cluster master 57, 58
configuring 50, 57
cross-environment search 67, 68
deployer 61
deployment server 64
file precedence, configuring 52, 53
indexing cluster 60
license master 57, 58
multisite environments 65
search heads 62, 63
Splunk directory structure 50, 51, 52
Splunk installation checklist 54
Splunk configuration
concurrent searches 27, 28
data inputs 25, 26
design decision, making 33
replication factor 30
search factor 30
selecting 24
Splunk Dashboard Examples 215
Splunk DB Connect
about 215, 223
configuring 226
connections, creating 231, 232
database drivers 228
database input, configuring 229
database JDBC drivers 225
hardware requisites 224
HEC port conflicts 241
identities, creating 229
input 233, 234
installing 224
Java runtime 224
lookups 238, 240

output 236, 237
reference 224
requisites 223
roles 230
task server 226
troubleshooting 240
Splunk deployment
capacity 251
documenting 68, 69
funding 125, 126
performance 251
supporting 124
Splunk development 262
Splunk directory structure 50, 51
Splunk docs
references 152
Splunk Enterprise deployment 246
Splunk Enterprise, for Windows
reference 48
Splunk Enterprise
about 10
antivirus software, disabling 47, 48
initiating 45, 47
initiating, on Windows 49
installing 41, 42
installing, on Linux 42
installing, on Windows server 47
installing, via GUI 48, 49
installing, with pathname 48
Linux settings 43
reboot, initiating 46
reference link 42
stopping, on Windows 49
system clocks, synchronization 49
upgrading 262
Splunk Free 11
Splunk free
installing 12
Splunk hardware options
disk-sizing calculations 38, 39
performance considerations 34, 35
selecting 34, 37
Splunk index data
deleting 98
Splunk indexes
creating 96, 97
managing 95, 96
metrics indexes 99, 100
summary indexes 98
Splunk installation checklist
about 54
component and IP address list 54
steps, installing 55, 56
Splunk Light 11
Splunk logs 246, 247, 248
Splunk Machine Learning Toolkit 215
Splunk monitoring console
about 254, 256
configuring 256, 257
using 258, 259
Splunk Premium apps
about 241
Enterprise Security (ES) 243
IT Service Intelligence (ITSI) 242
references 243
User Behavior Analytics (UBA) 243
Splunk Reports 153
Splunk resource-cost calculations 126
Splunk roles
authentication.conf 121
authorize.conf 120, 121
capabilities 119
indexes 119
managing 118
search restrictions 119
Splunk searches
creating 137
formatting commands 147
search command 138, 143
search filters 141, 142
Splunk support case
opening 250
Splunk support personnel 124
Splunk universal forwarder
initiating 74
inputs.conf, configuring 75, 77
installation steps 72, 73
installing 72
outputs.conf, configuring 74
Splunk Web interface

about 129, 130, 133, 134
search controls 131, 132
Splunk web interface
timeline and events 136
Splunk Web interface
timeline and events 134, 135
Splunk Web
forwarder management, using 94
Splunk's internal logs
forwarding 59
Splunk
about 9
components 13, 15, 17
custom source types, creating 102, 104
events 19, 21
history 11
information resources 22
processing tiers 17
products 10
references 267
sourcetypes 100, 101
troubleshooting 246
Splunkbase
reference 215
using 215
Statistics tab 149
stats command
about 144, 145, 146
reference 146
streaming command
versus transforming command 158, 159
summary indexes 98
system clocks
synchronization 49

T

table command 149, 150
tags 169
TCP input
configuring 60, 61
time modifier 139
timechart command 150, 152
tokens
using 197
top command 149
transaction command 156, 157
transforming command
versus streaming command 159

U

universal forwarder (UF) 14, 36
updated apps
deploying 114
User Behavior Analytics (UBA) 243
users
configuring 114

V

Visualization tab 134

W

warm buckets 31
where command 147
Windows Management Instrumentation (WMI) 47