Session 3-1: Getting Data in

Contents

Session 3-1 - Getting Data In

In this chapter, we will cover the basic ways to get data into **Splunk**. You will learn about the following recipes:

- Indexing files and directories
- Getting data through network ports
- Using scripted inputs
- Using modular inputs
- Using the Universal Forwarder to gather data
- Loading the sample data for this book
- Defining field extractions
- Defining event types and tags

Introduction

The machine data that facilitates operational intelligence comes in many different forms and from many different sources. Splunk is able to collect and index data from many different sources, including log files written by web servers or business applications, syslog data streaming in from network devices, or the output of custom-developed scripts. Even data that looks complex at first can be easily collected, indexed, transformed, and presented back to you in real time.

This **session will walk you through the basic recipes that will act as the building blocks to get the data you want into Splunk**. The session will further serve as an introduction to the sample datasets that we will use to build our own operational intelligence Splunk app. The datasets will be coming from a hypothetical, three-tier, e-commerce web application and will contain web server logs, application logs, and database logs.

Splunk Enterprise can index any type of data; however, it works best with time-series data (data with timestamps). When Splunk Enterprise indexes data, it breaks it into events, based on timestamps and/or event size, and puts them into indexes. Indexes are data stores that Splunk has engineered to be very fast, searchable, and scalable across a distributed server environment; they are commonly referred to as indexers. This is also why we refer to the data being put into Splunk as being indexed.

All data indexed into Splunk is assigned a source type. The source type helps identify the data format type of the event and where it has come from. Splunk has a number of preconfigured source types, but you can also specify your own. The example source types include `access_combined`, `cisco_syslog`, and `linux_secure`. The source type is added to the data when the indexer indexes it into Splunk. It is a key field that is used when performing field extractions and when conducting many searches to filter the data being searched.

The **Splunk community plays a big part in making it easy to get data into Splunk.** The ability to extend Splunk has provided the opportunity for the development of inputs, commands, and applications that can be easily shared. If there is a particular system or application you are looking to index data from, there is most likely someone who has developed and published relevant configurations and tools that can be easily leveraged by your own Splunk Enterprise deployment.

Splunk Enterprise is designed to **make the collection of data very easy,** and it will not take long before you are being asked or you yourself try to get as much data into Splunk as possible—at least as much as your license will allow for!

Indexing files and directories

File and directory-based inputs are the most commonly used ways of getting data into Splunk. The primary need for these types of inputs will be to index logfiles. Almost every application or system produces a logfile, and it is generally full of data that you want to be able to search and report on.

Splunk is able to continuously monitor for new data being written to existing files or new files being added to a directory, and it is able to index this data in real time. Depending on the type of application that creates the logfiles, you would set up Splunk to either monitor an individual file based on its location or scan an entire directory and monitor all the files that exist within it. The latter configuration is more commonly used when the logfiles being produced have unique filenames, such as filenames containing a timestamp.

This recipe will show you how to configure Splunk to continuously monitor and index the contents of a rolling logfile located on the Splunk server. The recipe specifically shows how to monitor and index a Red Hat Linux system's messages logfile (`/var/log/messages`). However, the same principle can be applied to a logfile on a Windows system, and a sample file is provided. Do not attempt to index the Windows event logs this way, as Splunk has specific Windows event inputs for this.

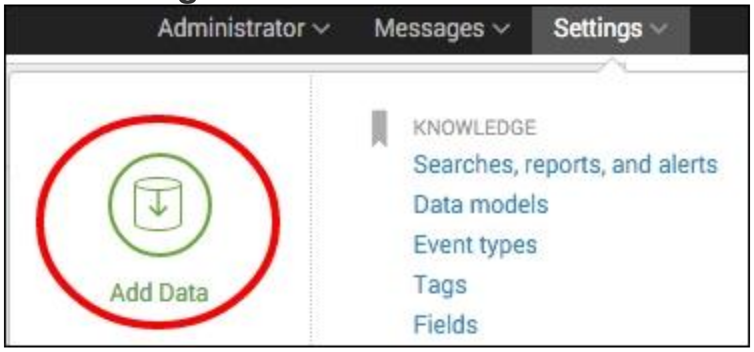
Getting ready

To step through this recipe, you will need a running Splunk Enterprise server and access to read the `/var/log/messages` file on Linux. No other prerequisites are required. If you are not using Linux and/or do not have access to the `/var/log/messages` location on your Splunk server, use the `cp01_messages.log` file that is provided and upload it to an accessible directory on your Splunk server.

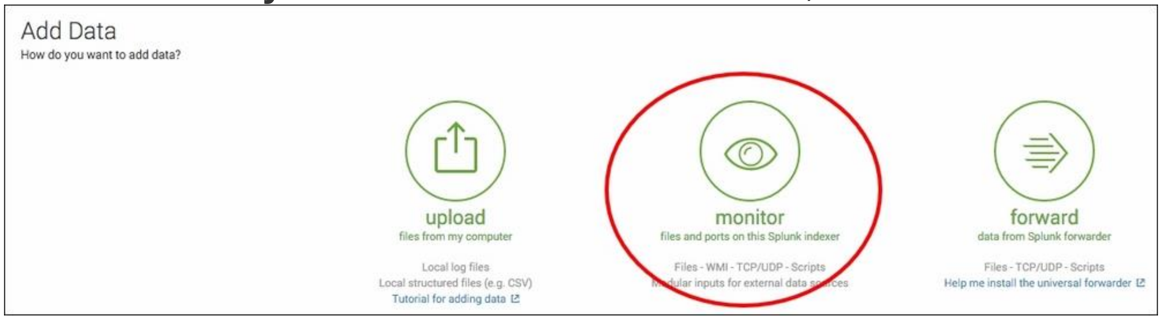
How to do it…

Follow the steps in the recipe to monitor and index the contents of a file:

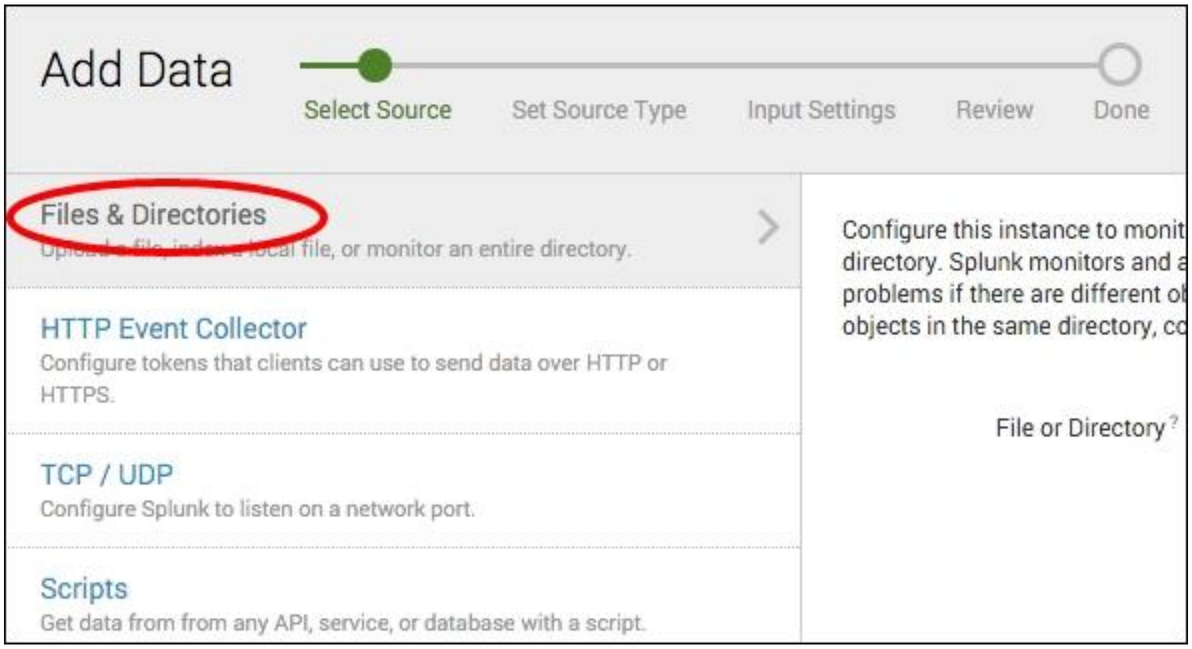
1. Log in to your Splunk server.
2. From the menu in the top right-hand corner, click on the **Settings** menu and then click on the **Add Data** link.



3. If you are prompted to take a quick tour, click on **Skip**.
4. In the **How do you want to add data?** section, click on **monitor**.


Add Data
How do you want to add data?
upload
files from my computer
Local log files
Local structured files (e.g. CSV)
Tutorial for adding data
monitor
files and ports on this Splunk indexer
Files - WMI - TCP/UDP - Scripts
Modular inputs for external data sources
forward
data from Splunk forwarder
Files - TCP/UDP - Scripts
Help me install the universal forwarder

5. Click on the **Files & Directories** section.



Files & Directories
Upload a file, index a local file, or monitor an entire directory.

HTTP Event Collector
Configure tokens that clients can use to send data over HTTP or HTTPS.

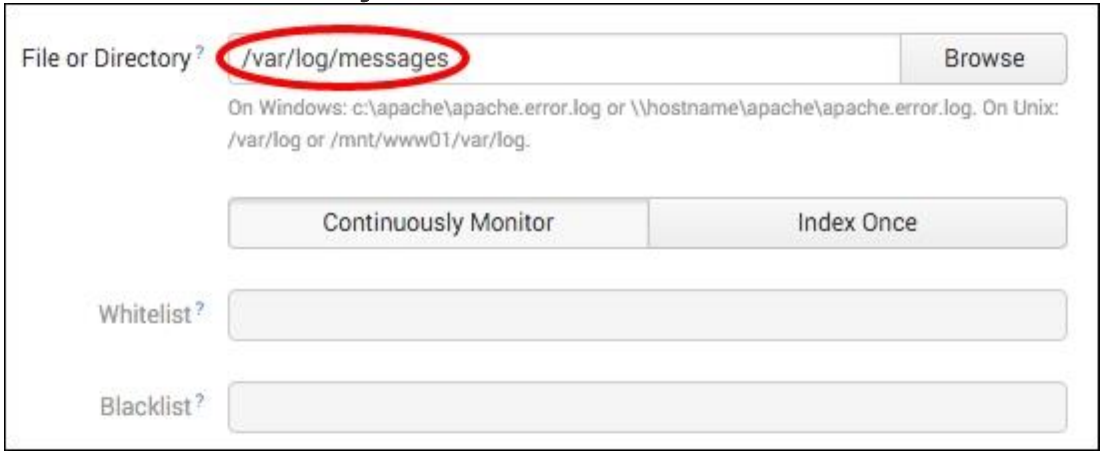
TCP / UDP
Configure Splunk to listen on a network port.

Scripts
Get data from from any API, service, or database with a script.

Configure this instance to monit
directory. Splunk monitors and a
problems if there are different ob
objects in the same directory, co

File or Directory ?

6. In the **File or Directory** section, enter the path to the logfile (`/var/log/messages` or the location of the `cp01_messages.log` file), ensure **Continuously Monitor** is selected, and click on **Next**.



File or Directory ? /var/log/messages

On Windows: c:\apache\apache.error.log or \\hostname\apache\apache.error.log. On Unix: /var/log or /mnt/www01/var/log.

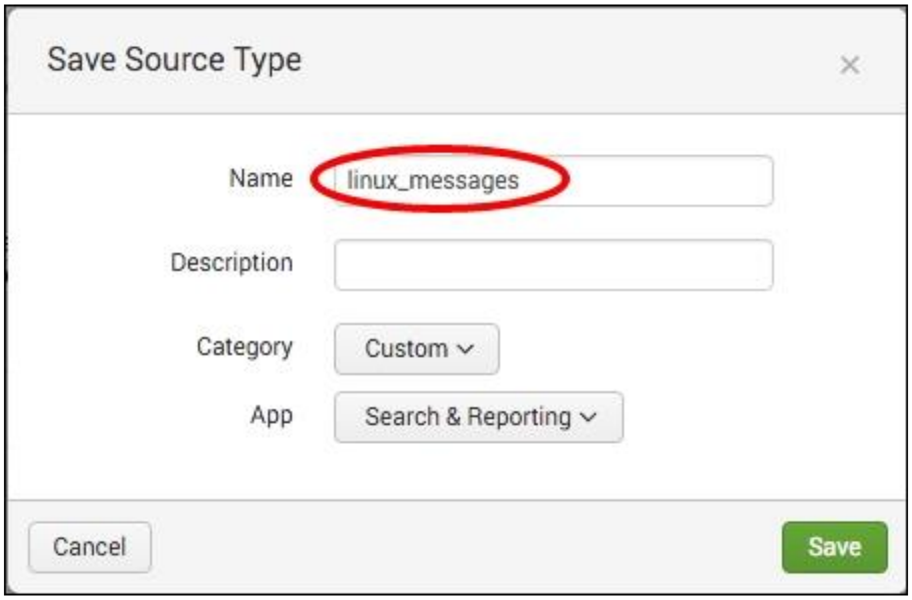
TIP

If you are just looking to do a one-time upload of a file, you can select **Index Once** instead. This can be useful to index a set of data that you would like to put into Splunk, either to backfill some missing or incomplete data or just to take advantage of its searching and reporting tools.

7. Assuming that you are using the provided file or the native `/var/log/messages` file, the data preview will show the correct

line breaking of events and timestamp recognition. Click on the **Next** button.

8. A **Save Source Type** box will pop up. Enter `linux_messages` as the **Name** and then click on **Save**.



Save Source Type

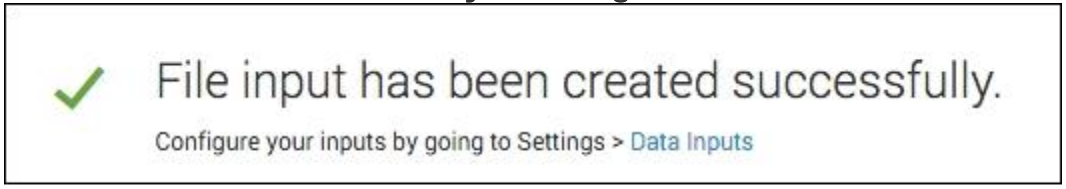
Name: linux_messages

Description:

Category: Custom

App: Search & Reporting

9. On the **Input Settings** page, leave all of the default settings, and click **Review**.
10. Review the settings and if everything is correct, click **Submit**.
11. If everything was successful, you should see a **File input has been created successfully** message.



File input has been created successfully.

Configure your inputs by going to Settings > Data Inputs

12. Click on the **Start searching** button. The **Search & Reporting** app will open with the search already populated based on the settings supplied earlier in the recipe.

TIP

In this recipe, we could have simply used the common syslog source type or let Splunk choose a source type name for us; however, starting a new source type is often a better choice. The syslog format can look completely different depending on the data source. As knowledge objects, such as field extractions, are built on top of source types, using a single syslog source type for everything can make it challenging to search for the data you need.

How it works…

When you add a new file or directory data input, you are basically adding a new configuration stanza into an `inputs.conf` file behind the scenes. The Splunk server can contain one or more `inputs.conf` files, and these files are either located in `$SPLUNK_HOME/etc/system/local` or in the local directory of a Splunk app.

Splunk uses the monitor input type and is set to point to either a file or a directory. If you set the monitor to point to a directory, all the files within that directory will be monitored. When Splunk monitors files, it initially starts by indexing all the data that it can read from the beginning. Once complete, Splunk maintains a record of where it last read the data from, and if any new data comes into the file, it reads this data and advances the record. The process is nearly identical to using the `tail` command in Unix-based operating systems. If you are monitoring a directory, Splunk also provides many additional configuration options, such as blacklisting files you don't want Splunk to index.

TIP

For more information on Splunk's configuration files, visit http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles.

There's more…

While adding inputs to monitor files and directories can be done through the web interface of Splunk, as outlined in this recipe, there are other approaches to add multiple inputs quickly. These allow for customization of the many configuration options that Splunk provides.

ADDING A FILE OR DIRECTORY DATA INPUT VIA THE CLI

Instead of going via the GUI, you can add a file or directory input via the Splunk **CLI** (**command-line interface**). Navigate to your `$SPLUNK_HOME/bin` directory and execute the following command (replacing the file or directory to be monitored with your own):

For Unix:

```
./splunk add monitor /var/log/messages -sourcetype
linux_messages
```

For Windows:

```
splunk add monitor c:\filelocation\cp01_messages.log -
sourcetype linux_messages
```

There are a number of different parameters that can be passed along with the file location to monitor.

NOTE

See the Splunk documentation for more on data inputs using the CLI (http://docs.splunk.com/Documentation/Splunk/latest/Data/MonitorfilesanddirectoriesusingtheCLI).

ADDING A FILE OR DIRECTORY INPUT VIA INPUTS.CONF

Another common method of adding the file and directory inputs is to manually add them to the `inputs.conf` configuration file directly. This approach is often used for large environments or when configuring Splunk forwarders to monitor for files or directories on endpoints.

Edit `$SPLUNK_HOME/etc/system/local/inputs.conf` and add your input. After your inputs are added, Splunk will need to be restarted to recognize these changes:

For Unix:

```
[monitor:///var/log/messages]

sourcetype = linux_messages
```

For Windows:

```
    [monitor://c:\filelocation\cp01_messages.log]
    sourcetype = linux_messages
```

NOTE

Editing `inputs.conf` directly is often a much faster way of adding new files or directories to monitor when several inputs are needed. When editing `inputs.conf`, ensure that the correct syntax is used and remember that Splunk will need a restart for modifications to take effect. Additionally, specifying the source type in the `inputs.conf` file is best practice to assign source types.

ONE-TIME INDEXING OF DATA FILES VIA THE SPLUNK CLI

Although you can select **Upload and Index a file** from the Splunk GUI to upload and index a file, there are a couple of CLI functions that can be used to perform one-time bulk loads of data.

Use the `oneshot` command to tell Splunk where the file is located and which parameters to use, such as the source type:

```
    ./splunk add oneshot XXXXXXX
```

Another way is to place the file you wish to index into the Splunk `spool`directory, `$SPLUNK_HOME/var/spool/splunk`, and then add the file using the `spool` command:

```
    ./splunk spool XXXXXXX
```

TIP

If using Windows, `omit ./` that is in front of the Splunk commands mentioned earlier.

INDEXING THE WINDOWS EVENT LOGS

Splunk comes with special `inputs.conf` configurations for some source types, including monitoring the Windows event logs. Typically, the Splunk **Universal Forwarder** (**UF**) would be installed on a Windows server and configured to forward the Windows events to the Splunk indexer(s). The configurations for `inputs.conf` to monitor the Windows security, application, and event logs in real time are as follows:

```
[WinEventLog://Application]
disabled = 0
[WinEventLog://Security]
disabled = 0
[WinEventLog://System]
disabled = 0
```

By default, the event data will go into the main index, unless another index is specified.

Getting data through network ports

Not every machine has the luxury of being able to write logfiles. Sending data over network ports and protocols is still very common. For instance, sending logs via syslog is still the primary method to capture network device data such as firewalls, routers, and switches.

Sending data to Splunk over network ports doesn't need to be limited to network devices. Applications and scripts can use socket communication to the network ports that Splunk is listening on. This can be a very useful tool in your back pocket, as there can be scenarios where you need to get data into Splunk but don't necessarily have the ability to write to a file.

This recipe will show you how to configure Splunk to receive syslog data on a UDP network port, but it is also applicable to the TCP port configuration.

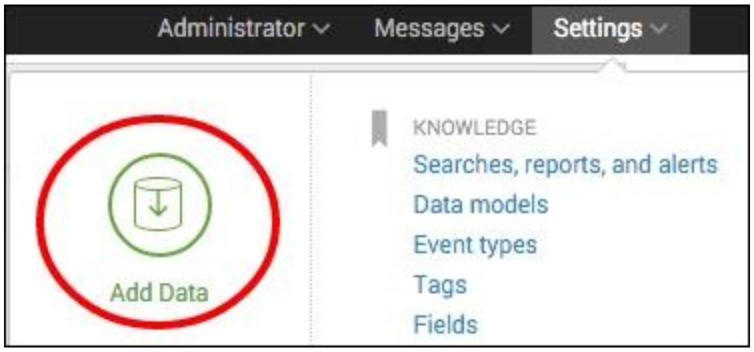
Getting ready

To step through this recipe, you will need a running Splunk Enterprise server. No other prerequisites are required.

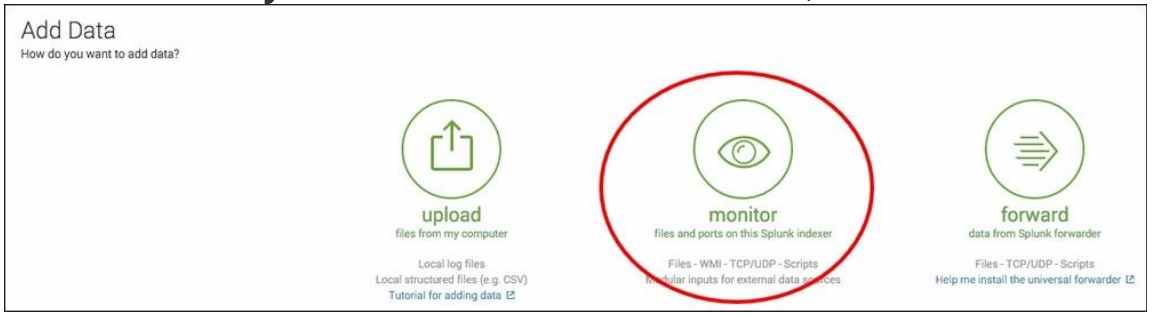
How to do it…

Follow the steps in the recipe to configure Splunk to receive network UDP data:

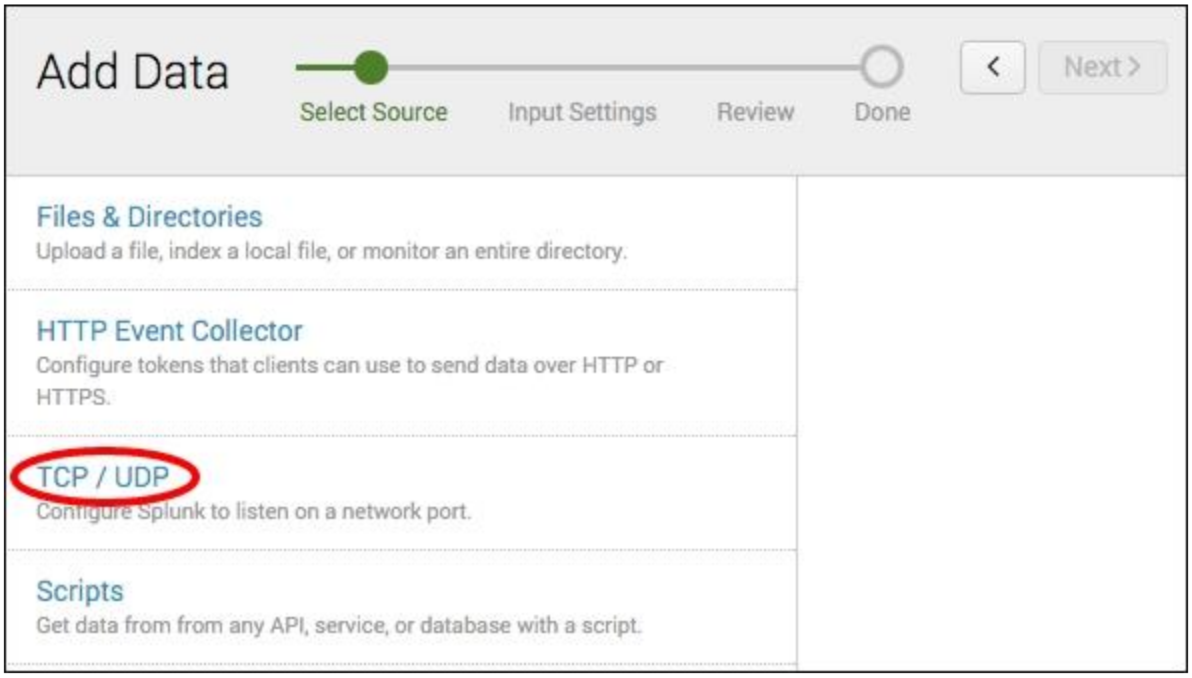
1. Log in to your Splunk server.
2. From the menu in the top right-hand corner, click on the **Settings** menu and then click on the **Add Data** link.



3. If you are prompted to take a quick tour, click on **Skip**.
4. In the **How do you want to add data?** section, click on **monitor**.

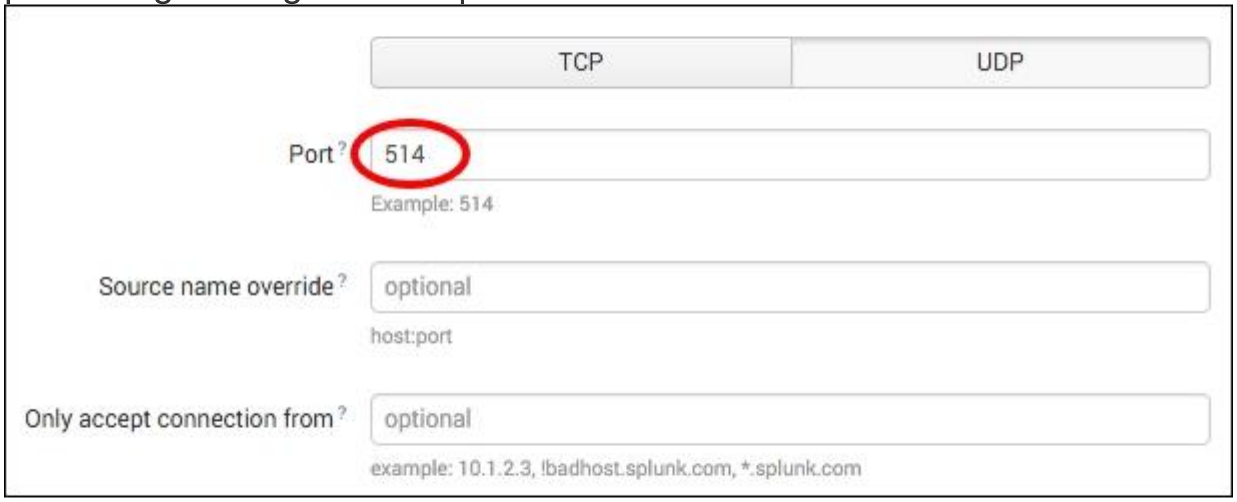

Add Data
How do you want to add data?
upload
files from my computer
Local log files
Local structured files (e.g. CSV)
Tutorial for adding data
monitor
files and ports on this Splunk indexer
Files - WMI - TCP/UDP - Scripts
Modular inputs for external data sources
forward
data from Splunk forwarder
Files - TCP/UDP - Scripts
Help me install the universal forwarder

5. Click on the **TCP / UDP** section.


Add Data
Select Source
Input Settings
Review
Done
Next >
Files & Directories
Upload a file, index a local file, or monitor an entire directory.
HTTP Event Collector
Configure tokens that clients can use to send data over HTTP or HTTPS.
TCP / UDP
Configure Splunk to listen on a network port.
Scripts
Get data from from any API, service, or database with a script.

6. Ensure the **UDP** option is selected and in the **Port** section, enter `514`. On Unix/Linux, Splunk must be running as root to access privileged ports such as 514. An alternative would be to specify a

higher port such as port 1514 or route data from 514 to another port using routing rules in iptables. Then click on **Next**.

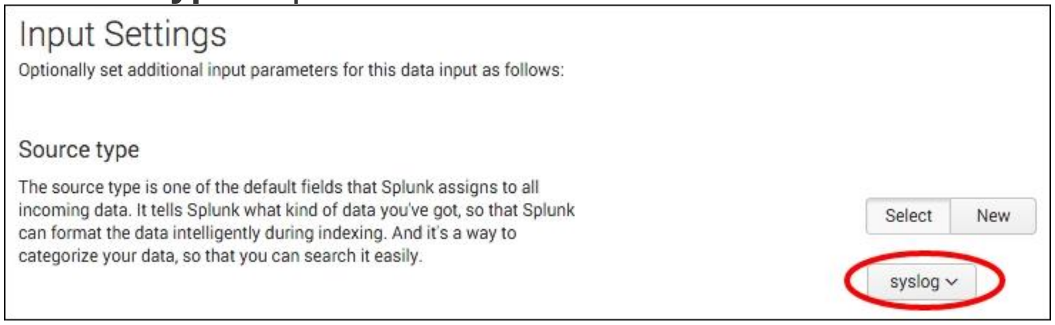


Port? 514
Example: 514

Source name override? optional
host:port

Only accept connection from? optional
example: 10.1.2.3, !badhost.splunk.com, *.splunk.com

7. In the **Source type** section, select **From** list from the **Set sourcetype**drop-down list, and then, select **syslog** from the **Select Source Type**drop-down list and click **Review**.



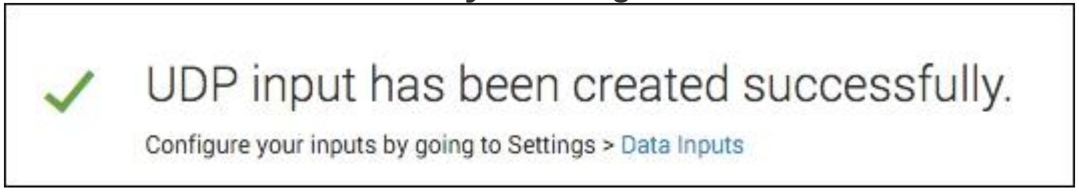
Input Settings

Optionally set additional input parameters for this data input as follows:

Source type

The source type is one of the default fields that Splunk assigns to all incoming data. It tells Splunk what kind of data you've got, so that Splunk can format the data intelligently during indexing. And it's a way to categorize your data, so that you can search it easily.

8. Review the settings and if everything is correct, click **Submit**.
9. If everything was successful, you should see a **UDP input has been created successfully** message.



UDP input has been created successfully.

Configure your inputs by going to Settings > Data Inputs

10. Click on the **Start searching** button. The **Search & Reporting** app will open with the search already populated based on the settings supplied earlier in the recipe. Splunk is now configured to listen on UDP port 514. Any data sent to this port

now will be assigned the syslog source type. To search for the syslog source type, you can run the following search:

```
source="udp:514" sourcetype="syslog"
```

Understandably, you will not see any data unless you happen to be sending data to your Splunk server IP on UDP port 514.

How it works…

When you add a new network port input, you basically add a new configuration stanza into an `inputs.conf` file behind the scenes. The Splunk server can contain one or more `inputs.conf` files, and these files are either located in the `$SPLUNK_HOME/etc/system/local` or the local directory of a Splunk app.

To collect data on a network port, Splunk will set up a socket to listen on the specified TCP or UDP port and will index any data it receives on that port. For example, in this recipe, you configured Splunk to listen on port 514 for UDP data. If data was received on that port, then Splunk would index it and assign a syslog source type to it.

Splunk also provides many configuration options that can be used with network inputs, such as how to resolve the host value to use on the collected data.

TIP

For more information on Splunk's configuration files, visit http://docs.splunk.com/Documentation/Splunk/latest/Admin/Aboutconfigurationfiles.

There's more…

While adding inputs to receive data from network ports can be done through the web interface of Splunk, as outlined in this recipe, there are other approaches to add multiple inputs quickly; these inputs allow for customization of the many configuration options that Splunk provides.

ADDING A NETWORK INPUT VIA THE CLI

You can also add a file or directory input via the Splunk CLI. Navigate to your `$SPLUNK_HOME/bin` directory and execute the following command (just replace the protocol, port, and source type you wish to use):

For Unix:

```
./splunk add udp 514 -sourcetype syslog
```

For Windows:

```
splunk add udp 514 -sourcetype syslog
```

There are a number of different parameters that can be passed along with the port. See the Splunk documentation for more on data inputs using the CLI (http://docs.splunk.com/Documentation/Splunk/latest/Data/MonitorfilesanddirectoriesusingtheCLI).

ADDING A NETWORK INPUT VIA INPUTS.CONF

Network inputs can be manually added to the `inputs.conf` configuration files. Edit `$SPLUNK_HOME/etc/system/local/inputs.conf` and add your input. You will need to restart Splunk after modifying the file:

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

TIP

It is best practice to not send syslog data directly to an indexer. Instead, always place a forwarder between the network device and the indexer. The Splunk forwarder would be set up to receive the incoming syslog data (`inputs.conf`) and will load balance the data across your Splunk indexers (`outputs.conf`). The forwarder can also be configured to cache the syslog data in the event communication to the indexers is lost.

Using scripted inputs

Not all data that is useful for operational intelligence comes from logfiles or network ports. Splunk will happily take the output of a command or script and index it along with all your other data.

Scripted inputs are a very helpful way to get that hard-to-reach data. For example, if you have third-party-supplied command-line programs that can output data you would like to collect, Splunk can run the command periodically and index the results. Typically, scripted inputs are often used to pull data from a source, whereas network inputs await a push of data from a source.

This recipe will show you how to configure Splunk on an interval to execute your command and direct the output into Splunk.

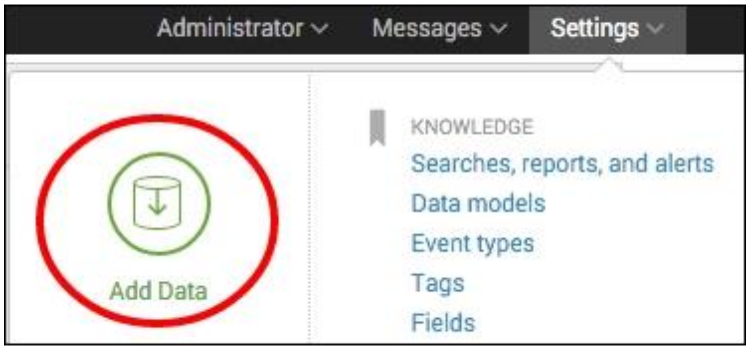
Getting ready

To step through this recipe, you will need a running Splunk server and the provided scripted input script suited to the environment you are using. For example, if you are using Windows, use the `cp01_scripted_input.bat`file. This script should be placed in the `$SPLUNK_HOME/bin/scripts`directory. No other prerequisites are required.

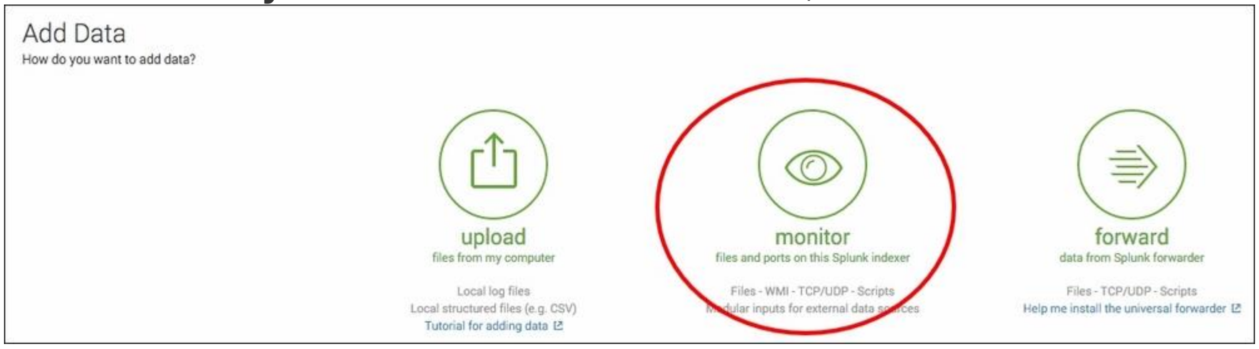
How to do it…

Follow the steps in the recipe to configure a scripted input:

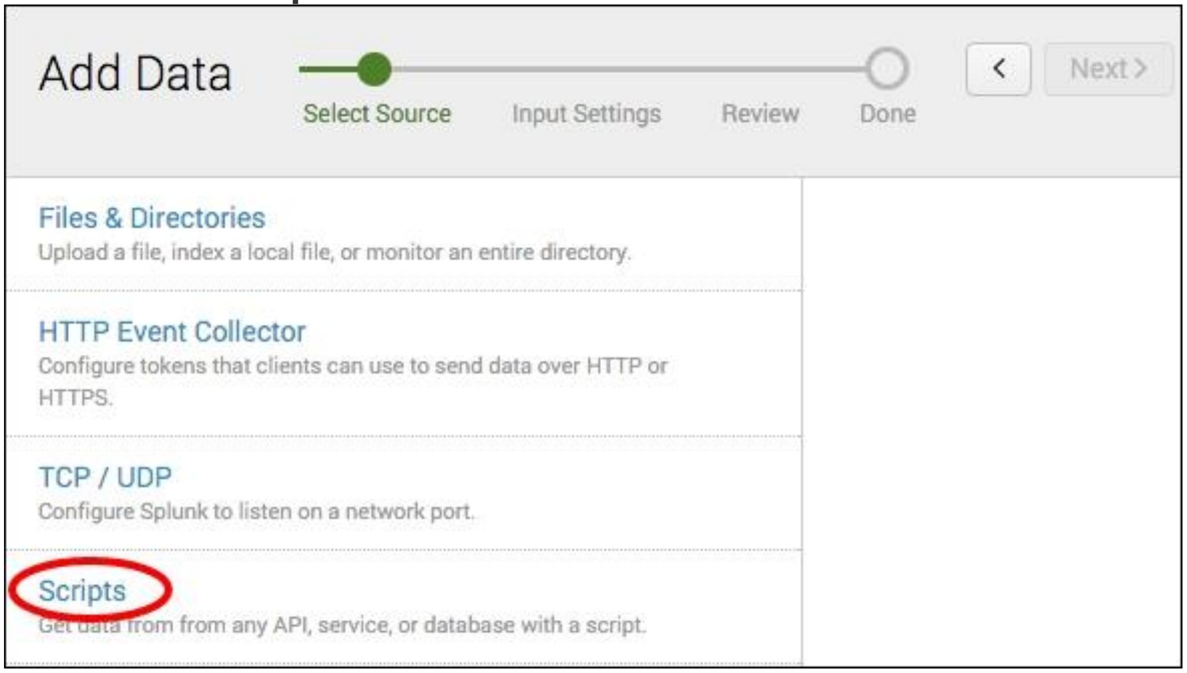
1. Log in to your Splunk server.
2. From the menu in the top right-hand corner, click on the **Settings** menu and then click on the **Add Data** link.



3. If you are prompted to take a quick tour, click on **Skip**.
4. In the **How do you want to add data?** section, click on **monitor**.


Add Data
How do you want to add data?
upload
files from my computer
Local log files
Local structured files (e.g. CSV)
Tutorial for adding data
monitor
files and ports on this Splunk indexer
Files - WMI - TCP/UDP - Scripts
Modular inputs for external data sources
forward
data from Splunk forwarder
Files - TCP/UDP - Scripts
Help me install the universal forwarder

5. Click on the **Scripts** section.



Files & Directories
Upload a file, index a local file, or monitor an entire directory.

HTTP Event Collector
Configure tokens that clients can use to send data over HTTP or HTTPS.

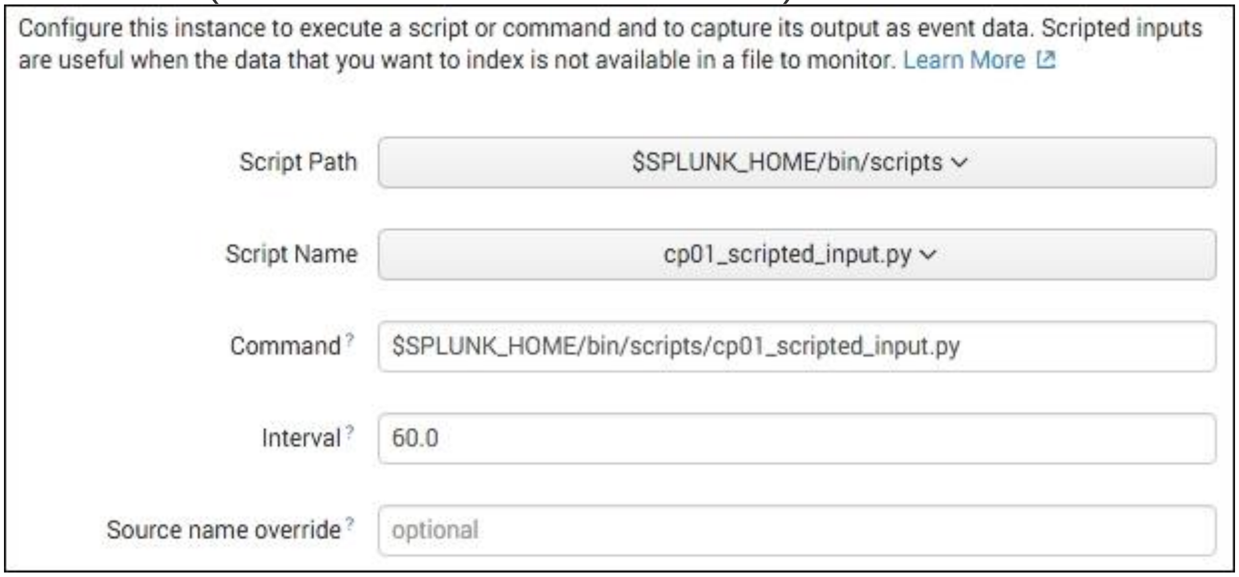
TCP / UDP
Configure Splunk to listen on a network port.

Scripts
Get data from from any API, service, or database with a script.

6. A form will be displayed with a number of input fields. In the **Script Path**drop-down, list select the location of the script. All scripts must be located in a Splunk `bin` directory, either

in `$SPLUNK_HOME/bin/scripts` or an appropriate bin directory within a Splunk app, such as `$SPLUNK_HOME/etc/apps/search/bin`.

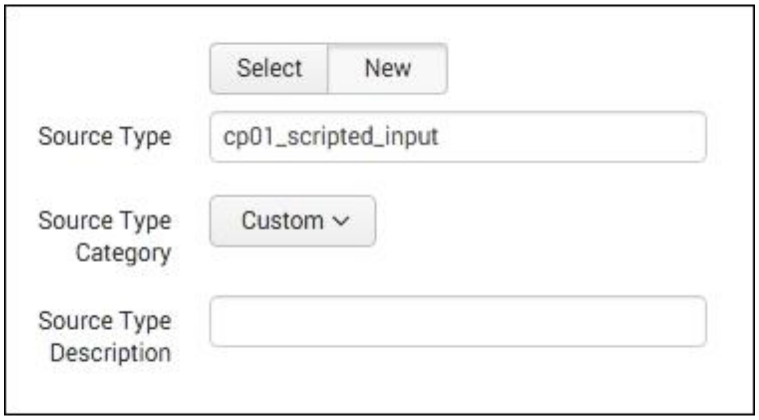
7. In the **Script Name** drop-down list, select the name of the script. In the **Commands** field, add any command-line arguments to the auto-populated script name.
8. Enter the value in the **Interval** field (in seconds) in which the script is to be run (the default value is **60.0** seconds) and then click **Next**.



Configure this instance to execute a script or command and to capture its output as event data. Scripted inputs are useful when the data that you want to index is not available in a file to monitor. Learn More

Script Path	$SPLUNK_HOME/bin/scripts
Script Name	cp01_scripted_input.py
Command ?	$SPLUNK_HOME/bin/scripts/cp01_scripted_input.py
Interval ?	60.0
Source name override ?	optional

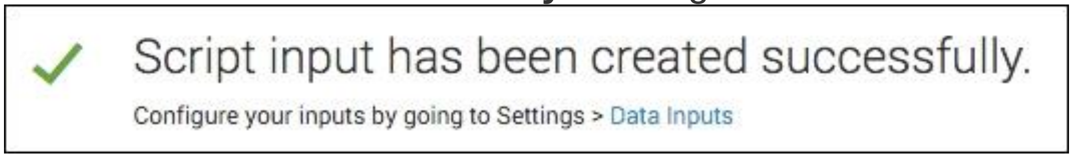
9. In the **Source Type** section, you have the option to either select a predefined source type or select **New** and enter your desired value. For the purpose of this recipe, select **New** as the source type and enter `cp01_scripted_input` as the value for the source type. Then click **Review**.



Source Type	cp01_scripted_input
Source Type Category	Custom
Source Type Description	

Data will be indexed into Splunk's default index, which is main. To change the destination index, you can select the desired index from the drop-down list in the **Index** section.

10. Review the settings. If everything is correct, click **Submit**.
11. If everything was successful, you should see a **Script input has been created successfully** message.



Script input has been created successfully.

Configure your inputs by going to Settings > Data Inputs

12. Click on the **Start searching** button. The **Search & Reporting** app will open with the search already populated based on the settings supplied earlier in the recipe. Splunk is now configured to execute the scripted input you provided every 60 seconds, in accordance with the specified interval. You can search for the data returned by the scripted input using the following search over all time:

```
sourcetype=cp01_scripted_input
```

How it works…

When adding a new scripted input, you are directing Splunk to add a new configuration stanza into an `inputs.conf` file behind the scenes. The Splunk server can contain one or more `inputs.conf` files, located either in `$SPLUNK_HOME/etc/system/local` or the local directory of a Splunk app.

After creating a scripted input, Splunk sets up an internal timer and executes the command that you have specified, in accordance with the defined interval. It is important to note that Splunk will only run one instance of the script at a time, so if the script gets blocked for any reason, it will cause the script to not be executed again, until after it has been unblocked.

Since Splunk 4.2, any output of the scripted inputs that are directed to `stderr` (causing an error) are captured to the `splunkd.log` file, which can be useful when attempting to debug the execution of a script. As Splunk indexes its own data by default, you can search for that data and alert on it if necessary.

For security reasons, Splunk does not execute scripts located outside of the bin directories mentioned earlier. In order to overcome this limitation, you can use a wrapper script (such as a shell script in Linux or batch file in Windows) to call any other script located on your machine.

Using modular inputs

Since Splunk 5.0, the ability to extend data input functionality has existed such that custom input types can be created and shared while still allowing for user customization to meet needs.

Modular inputs build further upon the scripted input model. Originally, any additional functionality required by the user had to be contained within a script. However, this presented a challenge, as no customization of this script could occur from within Splunk itself. For example, pulling data from a source for two different usernames needed two copies of a script or meant playing around with command-line arguments within your scripted input configuration.

By leveraging the modular input capabilities, the developers are now able to encapsulate their code into a reusable app that exposes parameters in Splunk and allows for configuration through processes familiar to Splunk administrators.

This recipe will walk you through how to install the **Command Modular Input**, which allows for periodic execution of commands and subsequent indexing of the command output. You will configure the input to collect the data output by the `vmstat` command in Linux and the `systeminfo`command in Windows.

Getting ready

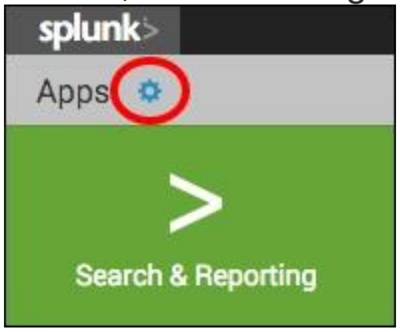
To step through this recipe, you will need a running Splunk server with a connection to the Internet. No other prerequisites are required.

How to do it…

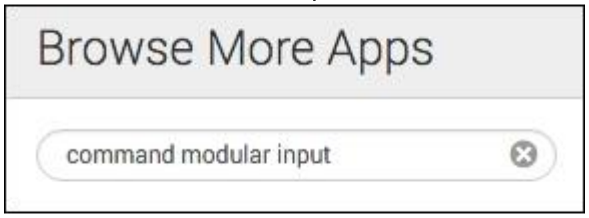
Follow the steps in this recipe to configure a modular input:

1. Log in to your Splunk server.

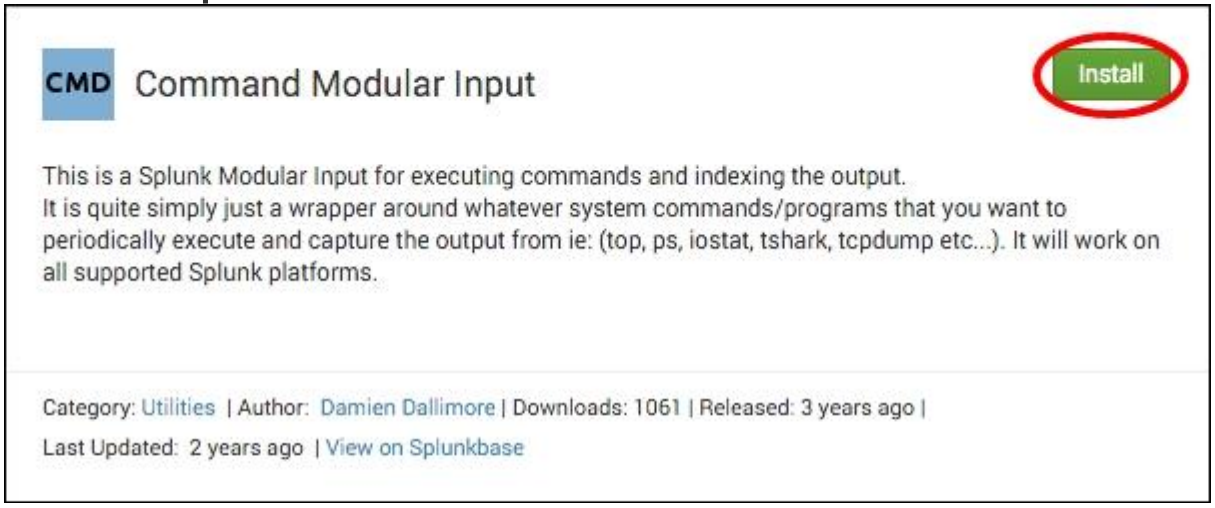
2. From the **Apps** menu in the upper left-hand corner of the home screen, click on the gear icon.



3. The **Apps settings** page will load. Then click on the **Browse More Apps**button.
4. In the search field, enter `command modular input` and press *Enter*.



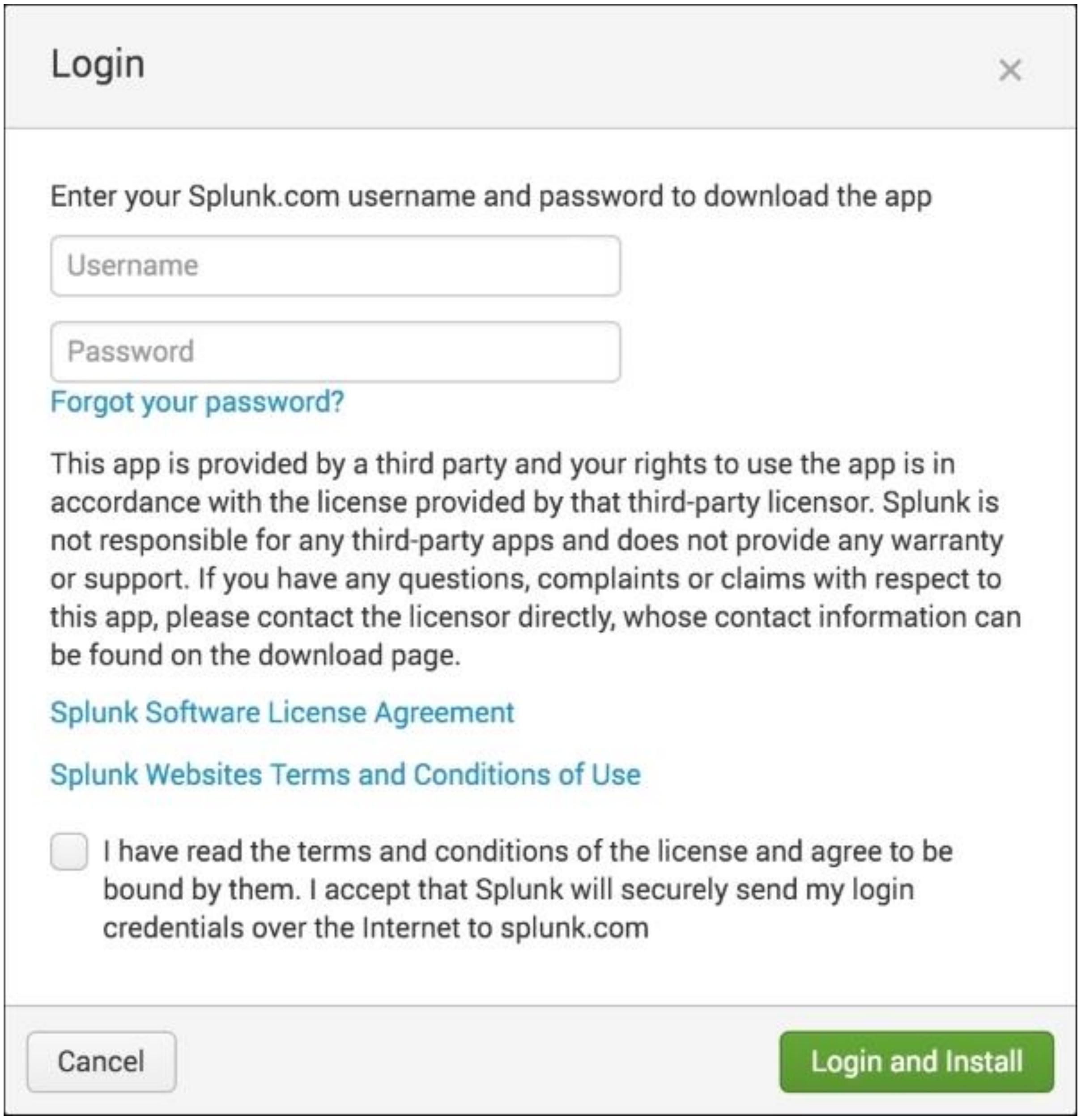
5. In the search results, click on the **Install** button for **Command Modular Input**.



This is a Splunk Modular Input for executing commands and indexing the output.
It is quite simply just a wrapper around whatever system commands/programs that you want to periodically execute and capture the output from ie: (top, ps, iostat, tshark, tcpdump etc...). It will work on all supported Splunk platforms.

Category: Utilities | Author: Damien Dallimore | Downloads: 1061 | Released: 3 years ago |
Last Updated: 2 years ago | View on Splunkbase

6. Enter your Splunk.com credentials, check the checkbox to accept the terms and conditions, and click on **Login and Install**. Splunk should return with a message saying that the app was installed successfully.


Login

Enter your Splunk.com username and password to download the app

Username

Password

Forgot your password?

This app is provided by a third party and your rights to use the app is in accordance with the license provided by that third-party licensor. Splunk is not responsible for any third-party apps and does not provide any warranty or support. If you have any questions, complaints or claims with respect to this app, please contact the licensor directly, whose contact information can be found on the download page.

Splunk Software License Agreement

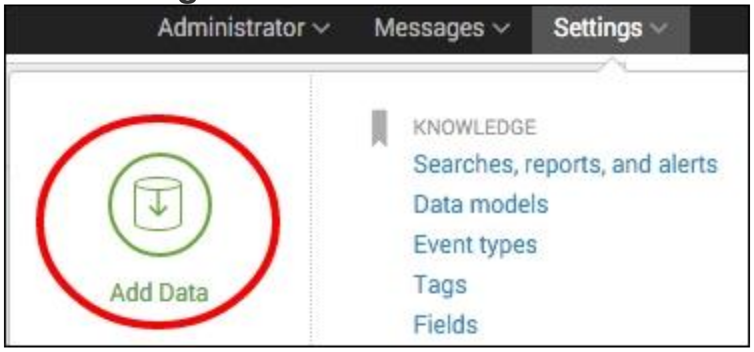
Splunk Websites Terms and Conditions of Use

☐ I have read the terms and conditions of the license and agree to be bound by them. I accept that Splunk will securely send my login credentials over the Internet to splunk.com

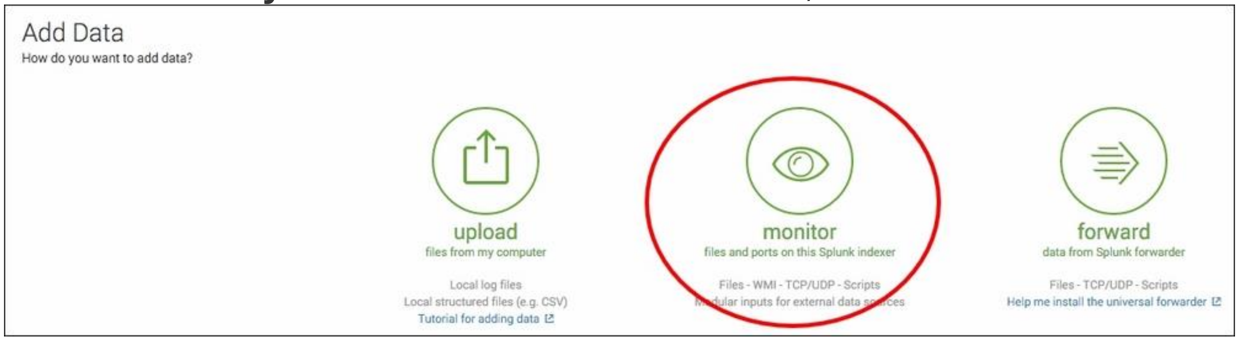
Cancel

Login and Install

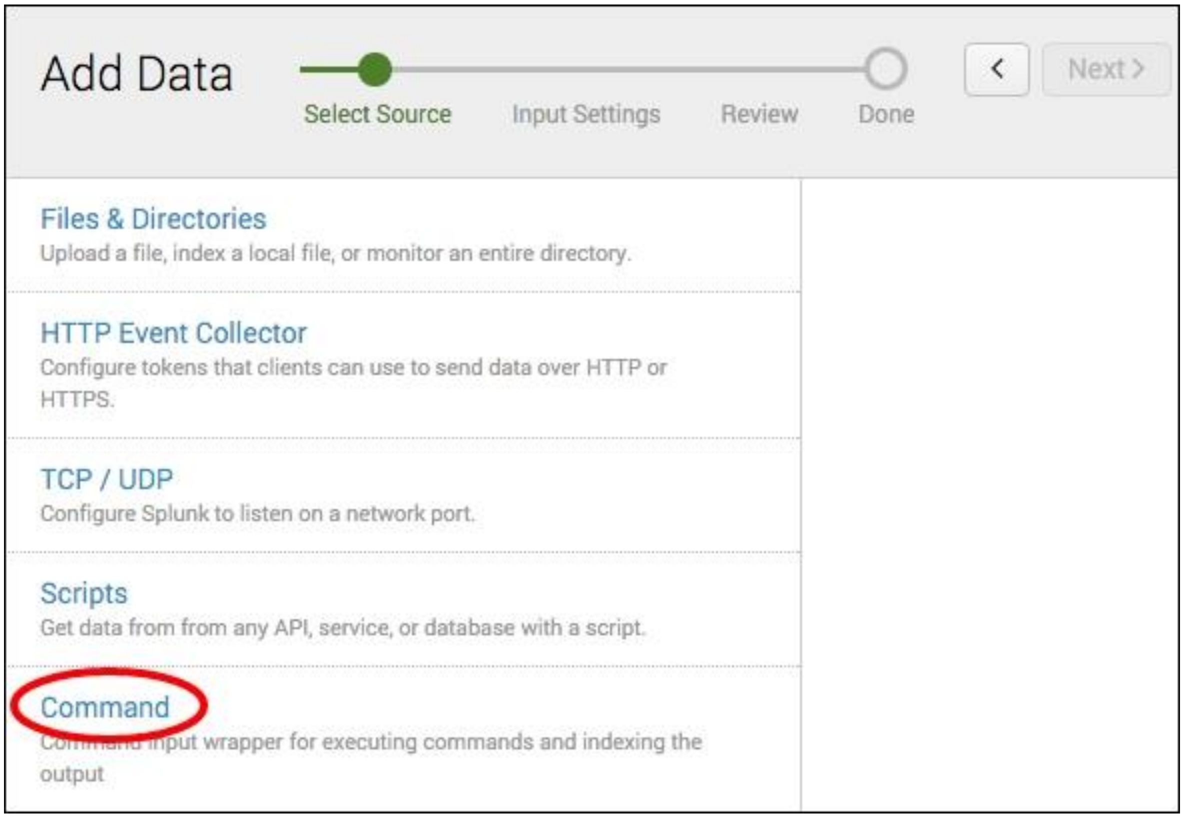
7. From the menu in the top right-hand corner, click on the **Settings** menu and then click on the **Add Data** link.



8. If you are prompted to take a quick tour, click on **Skip**.
9. In the **How do you want to add data?** section, click on **monitor**.


Add Data
How do you want to add data?
upload
files from my computer
Local log files
Local structured files (e.g. CSV)
Tutorial for adding data
monitor
files and ports on this Splunk indexer
Files - WMI - TCP/UDP - Scripts
Modular inputs for external data sources
forward
data from Splunk forwarder
Files - TCP/UDP - Scripts
Help me install the universal forwarder

10. Click on the **Command** section.



Files & Directories
Upload a file, index a local file, or monitor an entire directory.

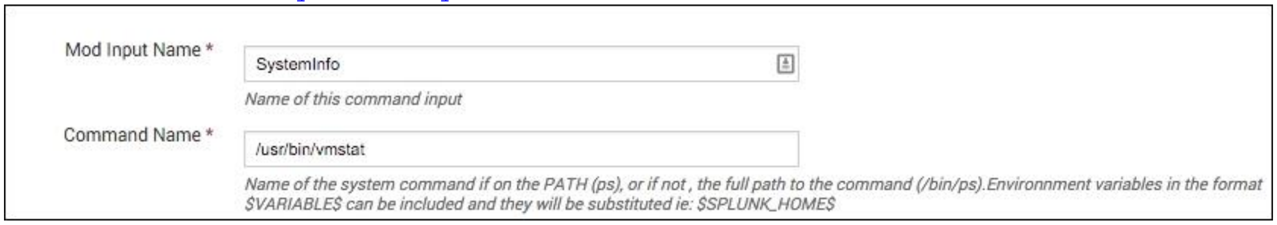
HTTP Event Collector
Configure tokens that clients can use to send data over HTTP or HTTPS.

TCP / UDP
Configure Splunk to listen on a network port.

Scripts
Get data from from any API, service, or database with a script.

Command
Command input wrapper for executing commands and indexing the output

11. In the **Mod Input Name** field, enter a name for the input of `SystemInfo`. If you are using Linux, enter `/usr/bin/vmstat` in the **Command Name**field. If you are using Windows, enter `C:\Windows\System32\systeminfo.exe` in the **Command Name**field.

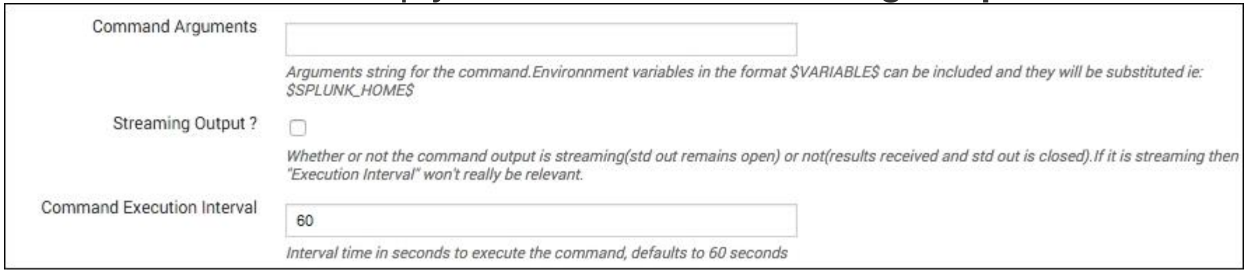


Mod Input Name * SystemInfo
Name of this command input

Command Name * /usr/bin/vmstat
Name of the system command if on the PATH (ps), or if not , the full path to the command (/bin/ps).Environnment variables in the format $VARIABLE$ can be included and they will be substituted ie: $SPLUNK_HOME$

Use the full path if the command to be executed cannot be found on the system `PATH`.

12. In the **Command Arguments** field, enter any argument that needs to be passed to the command listed in the **Command Name** field. In the **Command Execution Interval** field, enter a value in seconds for how often the command should be executed (in this case, we will use 60 seconds). If the output is streamed, then leave this field empty and check the **Streaming Output** field.



Command Arguments

Arguments string for the command.Environnment variables in the format $VARIABLE$ can be included and they will be substituted ie: $SPLUNK_HOME$

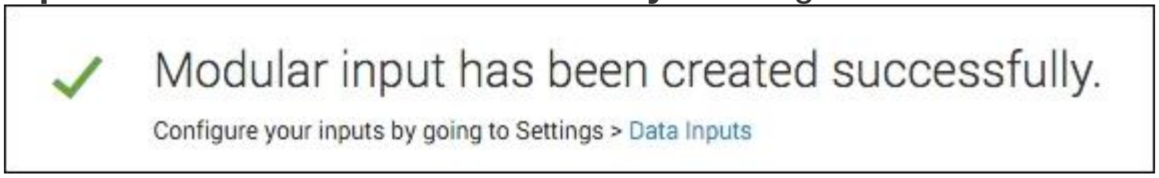
Streaming Output ? ☐

Whether or not the command output is streaming(std out remains open) or not(results received and std out is closed).If it is streaming then "Execution Interval" won't really be relevant.

Command Execution Interval: 60

Interval time in seconds to execute the command, defaults to 60 seconds

13. In the **Source type** section, you have the option to either select a predefined source type or select **Manual** and enter a value. For the purpose of this recipe, select **Manual** as the source type and enter `cp01_modular_input` as the value for the source type.
14. Click **Next**.
15. If everything was successful, you should see a **Modular input has been created successfully** message.



✓ Modular input has been created successfully.

Configure your inputs by going to Settings > Data Inputs

16. Click on the **Start searching** button. The **Search & Reporting** app will open with the search already populated based on the settings supplied earlier in the recipe. Splunk is now configured to execute the modular input you provided, every 60 seconds, in accordance with the specified interval. You can search for the data returned by the scripted input using the following search over all time:

```
sourcetype=cp01_modular_input
```

How it works…

Modular inputs are bundled as Splunk apps and, once installed, contain all the necessary configuration and code to display them in the **Data**

inputssection of Splunk. In this recipe, you installed a modular input application that allows for periodic execution of commands. You configured the command to execute every minute and index the results of the command each time, giving the results a source type of `cp01_modular_input`.

Modular inputs can be written in a number of languages and need to follow only a set of interfaces that expose the configuration options and runtime behaviors. Depending on the design of the input, they will either run persistently or run on an interval and will send data to Splunk as they receive it.

TIP

You can find several other modular inputs, including REST API, SNMP, and PowerShell, on the Splunk Apps site (http://splunkbase.splunk.com).

There's more…

To learn how to create your own modular input, refer to the *Modular Inputs*section of the *Developing Views and Apps for Splunk* Web manual located at http://docs.splunk.com/Documentation/Splunk/latest/AdvancedDev/ModInputsIntro.

Using the Universal Forwarder to gather data

Most IT environments today range from multiple servers in the closet of your office to hundreds of endpoint servers located in multiple geographically distributed data centers.

When the data we want to collect is not located directly on the server where Splunk is installed, the Splunk UF can be installed on your remote endpoint servers and used to forward data back to Splunk to be indexed.

The UF is similar to the Splunk server in that it has many of the same features, but it does not contain Splunk Web and doesn't come bundled with the Python executable and libraries. Additionally, the UF cannot process data in advance, such as performing line breaking and timestamp extraction.

This recipe will guide you through configuring the Splunk UF to forward data to a Splunk indexer and will show you how to set up the indexer to receive the data.

Getting ready

To step through this recipe, you will need a server with the Splunk UF installed but not configured. You will also need a running Splunk server. No other prerequisites are required.

TIP

To obtain the UF software, you need to go to http://www.splunk.com/en_us/download.html and register for an account if you do not already have one. Then, either download the software directly to your server or download it to your laptop or workstation and upload it to your server via a file-transfer process such as SFTP.

How to do it…

Follow the steps in the recipe to configure the Splunk Forwarder to forward data and the Splunk indexer to receive data:

1. On the server with the UF installed, open a command prompt if you are a Windows user or a terminal window if you are a Unix user.
2. Change to the `$SPLUNK_HOME/bin` directory, where `$SPLUNK_HOME` is the directory in which the Splunk Forwarder was installed.

   For Unix, the default installation directory will be `/opt/splunkforwarder/bin`. For Windows, it will be `C:\Program Files\SplunkUniversalForwarder\bin`.

   TIP

   If using Windows, `omit ./` in front of the Splunk command in the upcoming steps.

3. Start the Splunk Forwarder if not already started, using the following command:

   ```
   ./splunk start
   ```

4. Accept the license agreement.
5. Enable the UF to autostart, using the following command:

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

6. Set the indexer that this UF will send its data to. Replace the host value with the value of the indexer as well as the username and password for the UF:

   ```
   ./splunk add forward-server <host>:9997 -auth
   <username>:<password>
   ```

   The username and password to log in to the Forwarder (default is `admin:changeme`) is `<username>:<password>`.

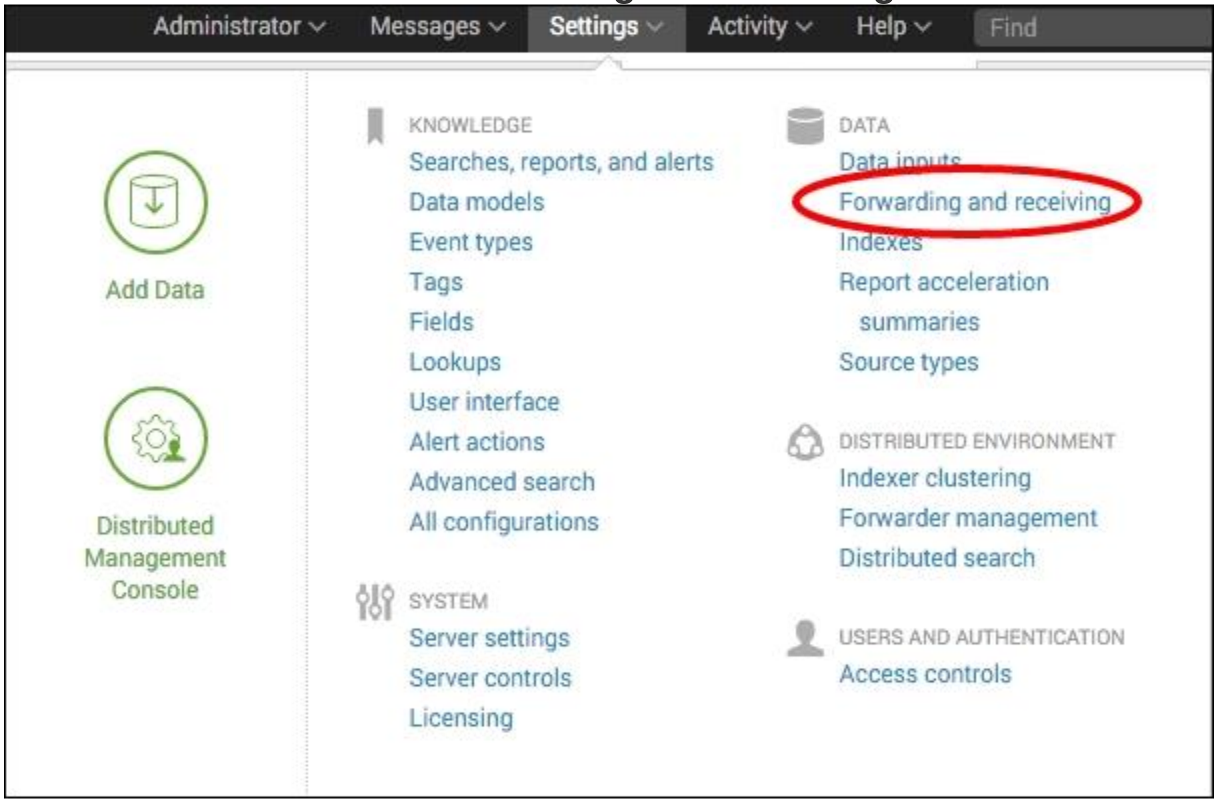
   TIP

   Additional receiving indexers can be added in the same way by repeating the command in the previous step with a different indexer host or IP. Splunk will automatically load balance the forwarded data if more than one receiving indexer is specified in this manner. Port 9997 is the default

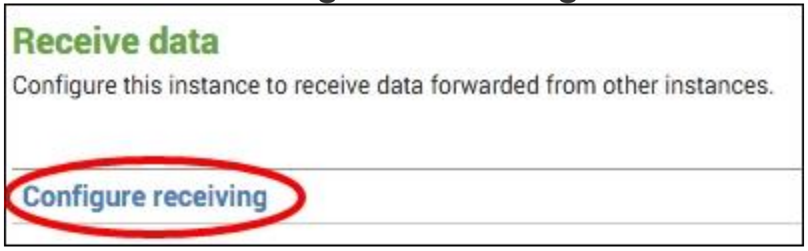
Splunk TCP port and should only be changed if it cannot be used for some reason.

On the receiving Splunk indexer servers:

1. Log in to your receiving Splunk indexer server. From the home launcher, in the top right-hand corner click on the **Settings** menu item and then select the **Forwarding and receiving** link.



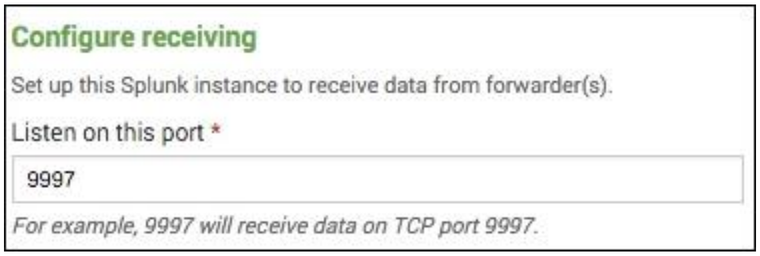
2. Click on the **Configure receiving** link.


Receive data

Configure this instance to receive data forwarded from other instances.

Configure receiving

3. Click on **New**.
4. Enter `9997` in the **Listen on this port** field.


Configure receiving

Set up this Splunk instance to receive data from forwarder(s).

Listen on this port *

9997

For example, 9997 will receive data on TCP port 9997.

5. Click on **Save** and restart Splunk. The UF is installed and configured to send data to your Splunk server, and the Splunk server is configured to receive data on the default Splunk TCP port 9997.

How it works…

When you tell the forwarder which server to send data to, you basically add a new configuration stanza into an `outputs.conf` file behind the scenes. On the Splunk server, an `inputs.conf` file will contain a `[splunktcp]` stanza to enable receiving. The `outputs.conf` file on the Splunk forwarder will be located in `$SPLUNK_HOME/etc/system/local`, and the `inputs.conf`file on the Splunk server will be located in the local directory of the app you were in (the launcher app in this case) when configuring receiving.

Using forwarders to collect and forward data has many advantages. The forwarders communicate with the indexers on TCP port 9997 by default, which makes for a very simple set of firewall rules that need to be opened. Forwarders can also be configured to load balance their data across multiple indexers, increasing search speeds and availability. Additionally, forwarders can be configured to queue the data they collect if communication with the indexers is lost. This can be extremely important when collecting data that is not read from logfiles, such as performance counters or syslog streams, as the data cannot be re-read.

There's more…

While configuring the settings of the UF can be performed via the command-line interface of Splunk, as outlined in this recipe, there are several other methods to update the settings quickly and allow for customization of the many configuration options that Splunk provides.

ADD THE RECEIVING INDEXER VIA OUTPUTS.CONF

The receiving indexers can be directly added to the `outputs.conf` configuration file on the UF. Edit `$SPLUNK_HOME/etc/system/local/outputs.conf`, add your input, and then, restart the UF. The following example configuration is provided, where two receiving indexers are specified. The `[tcpout-server]` stanza can be leveraged to add output configurations specific to an individual receiving indexer:

```
[tcpout]
defaultGroup = default-autolb-group

[tcpout:default-autolb-group]
disabled = false
server = mysplunkindexer1:9997,mysplunkindexer2:9997

[tcpout-server://mysplunkindexer1:9997]
[tcpout-server://mysplunkindexer2:9997]
```

TIP

If nothing has been configured in `inputs.conf` on the UF, but `outputs.conf` is configured with at least one valid receiving indexer, the Splunk forwarder will only send internal forwarder health-related data to the indexer. It is, therefore, possible to configure a forwarder correctly and be detected by the Splunk indexers, but not actually send any real data.

Loading the sample data for this Course

While most of the data you will index with Splunk will be collected in real time, there might be instances where you have a set of data that you would like to put into Splunk, either to backfill some missing or incomplete data, or just to take advantage of its searching and reporting tools.

This recipe will show you how to perform one-time bulk loads of data from files located on the Splunk server. We will also use this recipe to load the data samples that will be used throughout the subsequent sessions as we build our operational intelligence app in Splunk.

There are two files that make up our sample data.

- The first is `access_log`, which represents the data from our web layer and is modeled on an Apache web server.
- The second file is `app_log`, which represents the data from our application layer and is modeled on `log4j` log data from our custom middleware application.

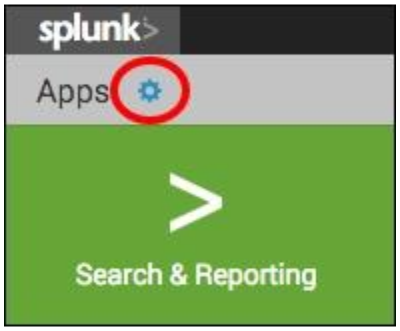
Getting ready

To step through this recipe, you will need a running Splunk server and you should have a copy of the sample data generation app (`OpsDataGen.spl`) for this book.

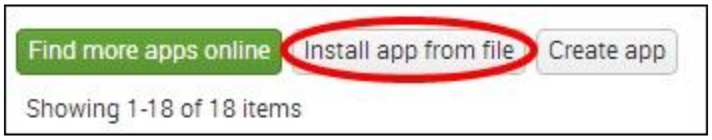
How to do it…

Follow the given steps to load the sample data generator on your system:

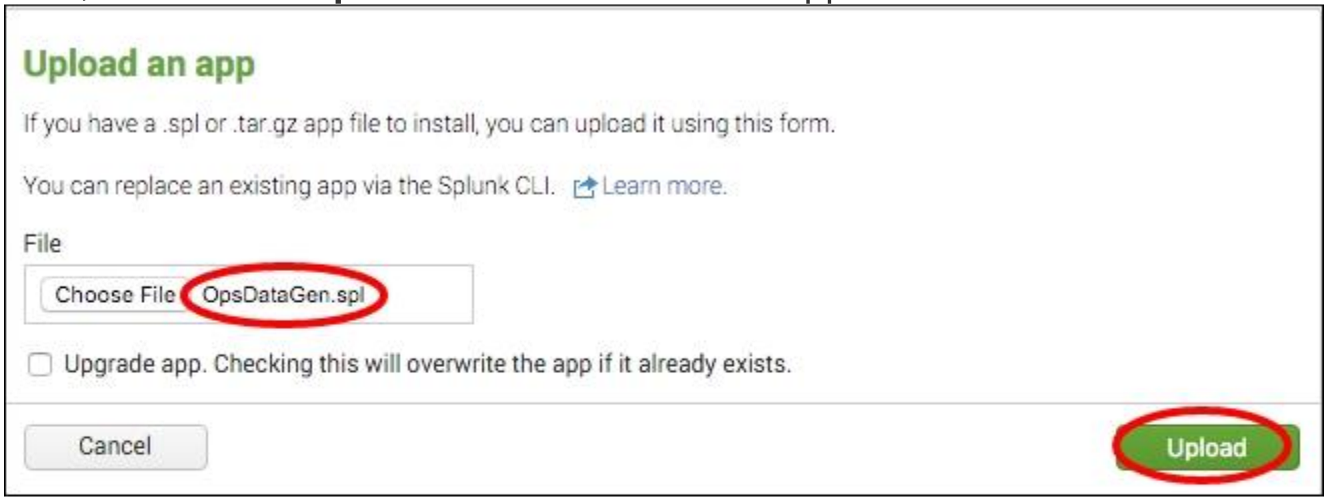
1. Log in to your Splunk server using your credentials.
2. From the **Apps** menu in the upper left-hand corner of the home screen, click on the gear icon.



3. The **Apps settings** page will load. Then click on the **Install app from file**button.



4. Select the location of the `OpsDataGen.spl` file on your computer, and then, click on the **Upload** button to install the application.



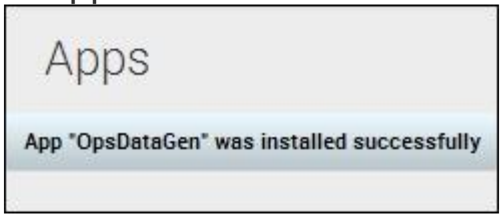
Upload an app

If you have a .spl or .tar.gz app file to install, you can upload it using this form.

You can replace an existing app via the Splunk CLI. Learn more.

File
Choose File
OpsDataGen.spl
Upgrade app. Checking this will overwrite the app if it already exists.
Cancel
Upload

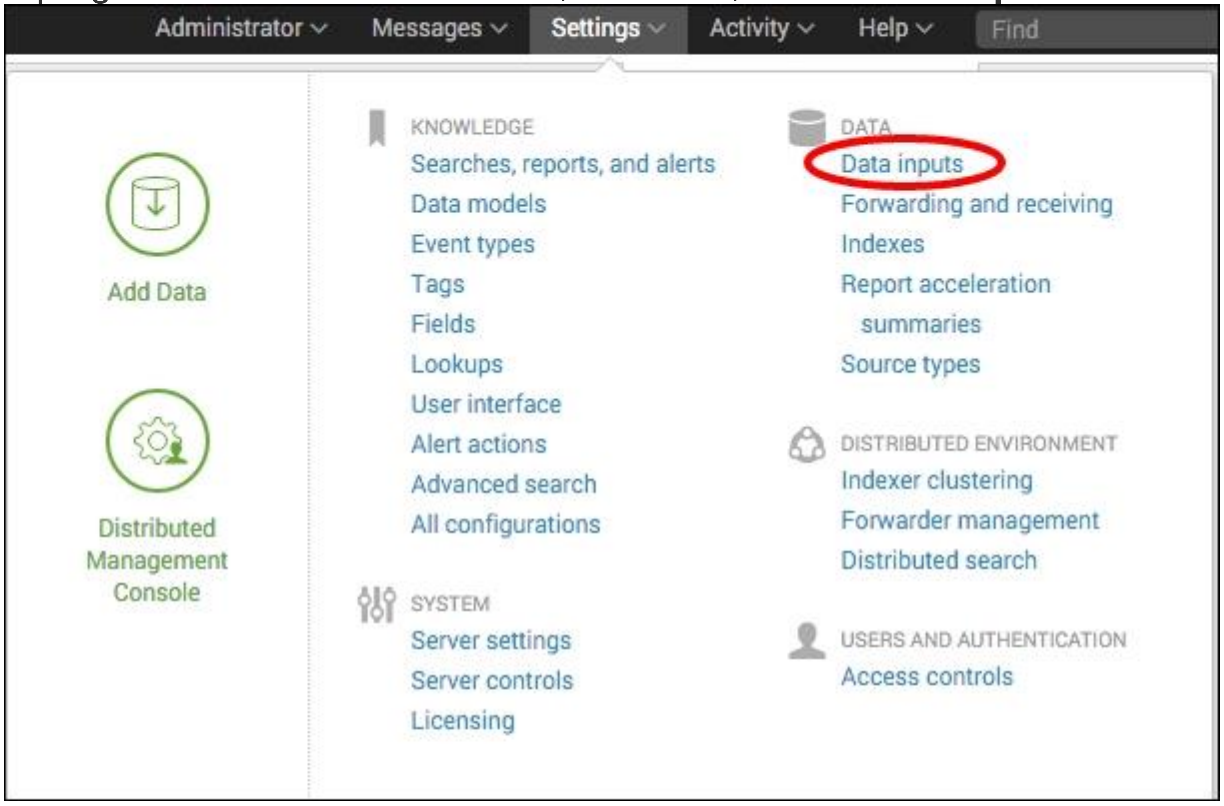
5. After installation, a message should appear in a blue bar at the top of the screen, letting you know that the app has installed successfully. You should also now see the `OpsDataGen` app in the list of apps.



App "OpsDataGen" was installed successfully

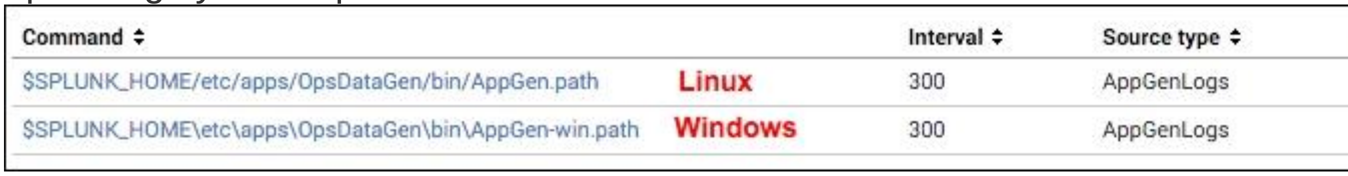
6. By default, the app installs with the data-generation scripts disabled. In order to generate data, you will need to enable either a Windows or Linux script, depending on your Splunk operating

system. To enable the script, select the **Settings** menu from the top right-hand side of the screen, and then, select **Data inputs**.



7. From the **Data inputs** screen that follows, select **Scripts**.
8. On the **Scripts** screen, locate the **OpsDataGen** script for your operating system and click on **Enable**.
   - For Linux, it will be `$SPLUNK_HOME/etc/apps/OpsDataGen/bin/AppGen.path`
   - For Windows, it will be `$SPLUNK_HOME\etc\apps\OpsDataGen\bin\AppGen-win.path`

   The following screenshot displays both the Windows and Linux inputs that are available after installing the **OpsDataGen** app. It also displays where to click to enable the correct one based on the operating system Splunk is installed on.



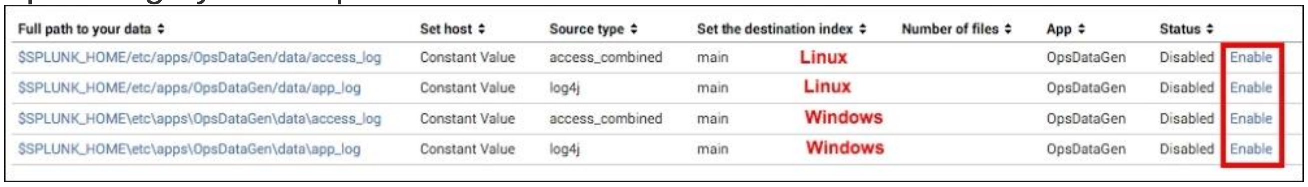
Command ⇕		Interval ⇕	Source type ⇕
$SPLUNK_HOME/etc/apps/OpsDataGen/bin/AppGen.path	Linux	300	AppGenLogs
$SPLUNK_HOME\etc\apps\OpsDataGen\bin\AppGen-win.path	Windows	300	AppGenLogs

9. Select the **Settings** menu from the top right-hand side of the screen, select **Data inputs**, and then select **Files & directories**.

10. On the **Files & directories** screen, locate the two **OpsDataGen** inputs for your operating system and for each click on **Enable**.

- For Linux, it will be `$SPLUNK_HOME/etc/apps/OpsDataGen/data/access_log` and `$SPLUNK_HOME/etc/apps/OpsDataGen/data/app_log`
- For Windows, it will be `$SPLUNK_HOME\etc\apps\OpsDataGen\data\access_log` and `$SPLUNK_HOME\etc\apps\OpsDataGen\data\app_log`

The following screenshot displays both the Windows and Linux inputs that are available after installing the **OpsDataGen** app. It also displays where to click to enable the correct one based on the operating system Splunk is installed on.

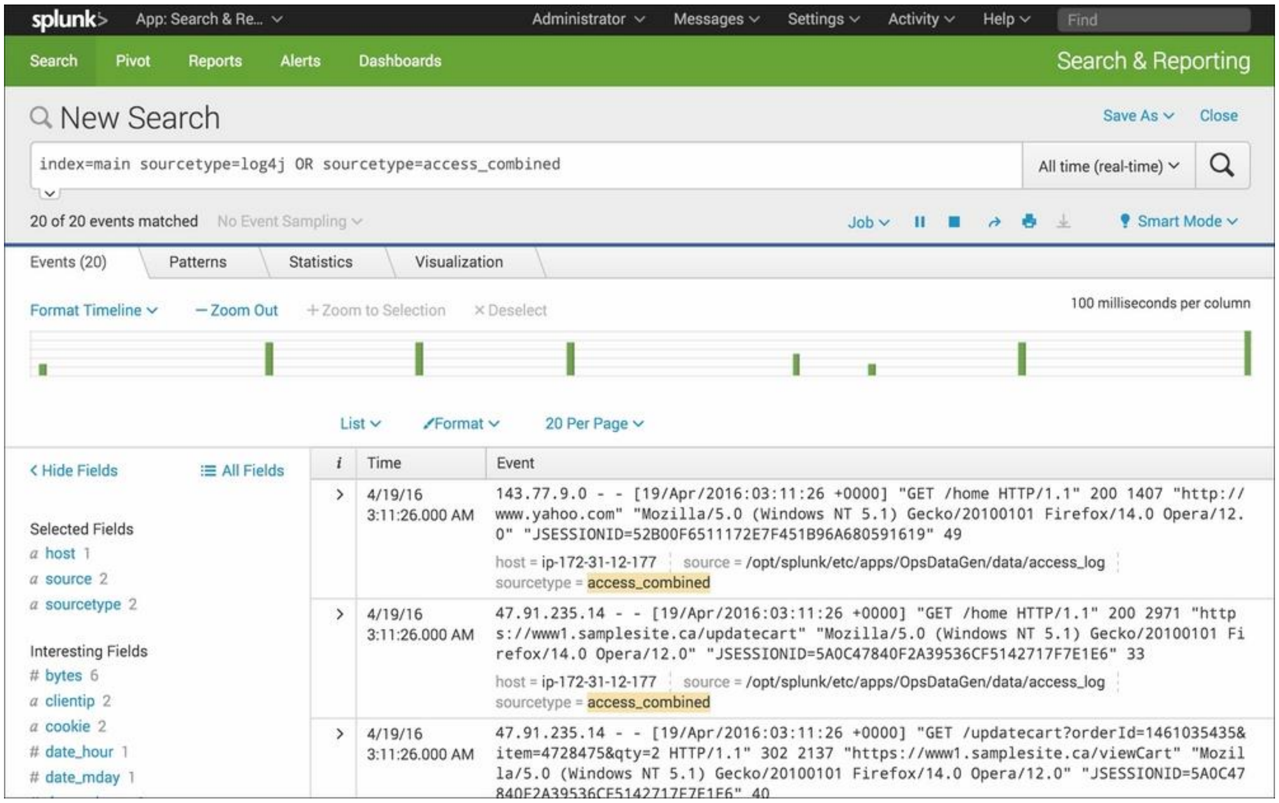


Full path to your data	Set host	Source type	Set the destination index	Number of files	App	Status	
$SPLUNK_HOME/etc/apps/OpsDataGen/data/access_log	Constant Value	access_combined	main	Linux	OpsDataGen	Disabled	Enable
$SPLUNK_HOME/etc/apps/OpsDataGen/data/app_log	Constant Value	log4j	main	Linux	OpsDataGen	Disabled	Enable
$SPLUNK_HOME\etc\apps\OpsDataGen\data\access_log	Constant Value	access_combined	main	Windows	OpsDataGen	Disabled	Enable
$SPLUNK_HOME\etc\apps\OpsDataGen\data\app_log	Constant Value	log4j	main	Windows	OpsDataGen	Disabled	Enable

11. The data will now be generated in real time. You can test this by navigating to the Splunk search screen and running the following search over an All time (real-time) time range:

```
index=main sourcetype=log4j OR sourcetype=access_combined
```

12. After a short while, you should see data from both the source types flowing into Splunk. The data generation is now working, as displayed in the following screenshot:



```
index=main sourcetype=log4j OR sourcetype=access_combined
```

20 of 20 events matched

i	Time	Event
>	4/19/16 3:11:26.000 AM	`143.77.9.0 - - [19/Apr/2016:03:11:26 +0000] "GET /home HTTP/1.1" 200 1407 "http://www.yahoo.com" "Mozilla/5.0 (Windows NT 5.1) Gecko/20100101 Firefox/14.0 Opera/12.0" "JSESSIONID=52B00F6511172E7F451B96A680591619" 49` host = ip-172-31-12-177 source = /opt/splunk/etc/apps/OpsDataGen/data/access_log sourcetype = access_combined
>	4/19/16 3:11:26.000 AM	`47.91.235.14 - - [19/Apr/2016:03:11:26 +0000] "GET /home HTTP/1.1" 200 2971 "https://www1.samplesite.ca/updatecart" "Mozilla/5.0 (Windows NT 5.1) Gecko/20100101 Firefox/14.0 Opera/12.0" "JSESSIONID=5A0C47840F2A39536CF5142717F7E1E6" 33` host = ip-172-31-12-177 source = /opt/splunk/etc/apps/OpsDataGen/data/access_log sourcetype = access_combined
>	4/19/16 3:11:26.000 AM	`47.91.235.14 - - [19/Apr/2016:03:11:26 +0000] "GET /updatecart?orderId=1461035435&item=4728475&qty=2 HTTP/1.1" 302 2137 "https://www1.samplesite.ca/viewCart" "Mozilla/5.0 (Windows NT 5.1) Gecko/20100101 Firefox/14.0 Opera/12.0" "JSESSIONID=5A0C47840F2A39536CF5142717F7E1E6" 40`

How it works…

In this case, you installed a Splunk application that leverages a scripted input. The script we wrote generates data for two source types. The `access_combined` source type contains sample web access logs, and the `log4j` source type contains application logs. These data sources will be used throughout the recipes in this course. Applications will also be discussed in more detail later on.

Defining field extractions

Splunk has many built-in features, including knowledge on several common source types, which lets it automatically know which fields exist within your data. Splunk, by default, also extracts any key-value pairs present within the log data and all the fields within the JSON-formatted logs. However, often the fields within raw log data cannot be interpreted out of the box, and this knowledge must be provided to Splunk in order to make these fields easily searchable.

The sample data that we will be using in subsequent sessions contains data we wish to present as fields to Splunk. Much of the raw log data contains key-value fields that Splunk will extract automatically, but there is one field we need to tell Splunk how to extract, **representing the page response time**. To do this, we will be adding a custom field extraction, which will tell Splunk how to extract the field for us.

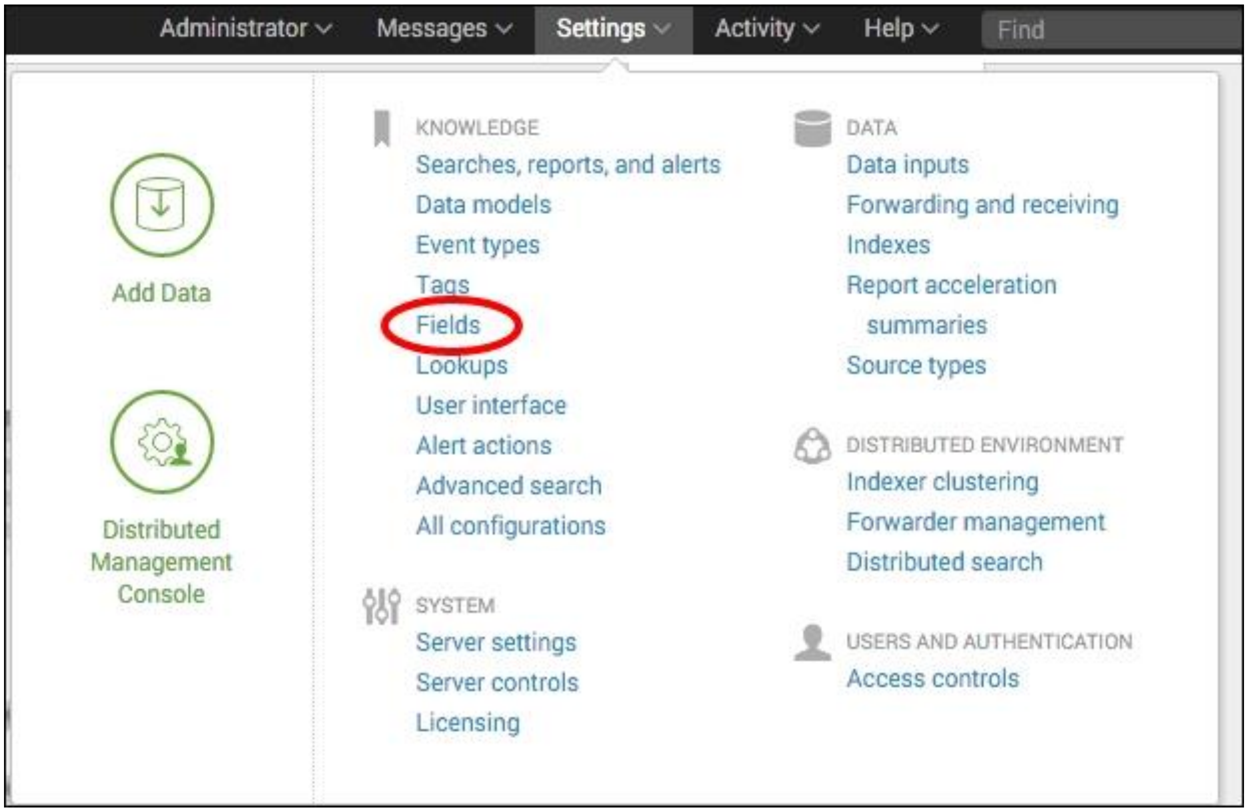
Getting ready

To step through this recipe, you will need a running Splunk server with the operational intelligence sample data loaded. No other prerequisites are required.

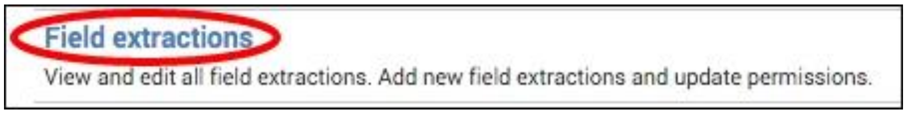
How to do it…

Follow the given steps to add a custom field extraction for response:

1. Log in to your Splunk server.
2. In the top right-hand corner, click on the **Settings** menu and then click on the **Fields** link.



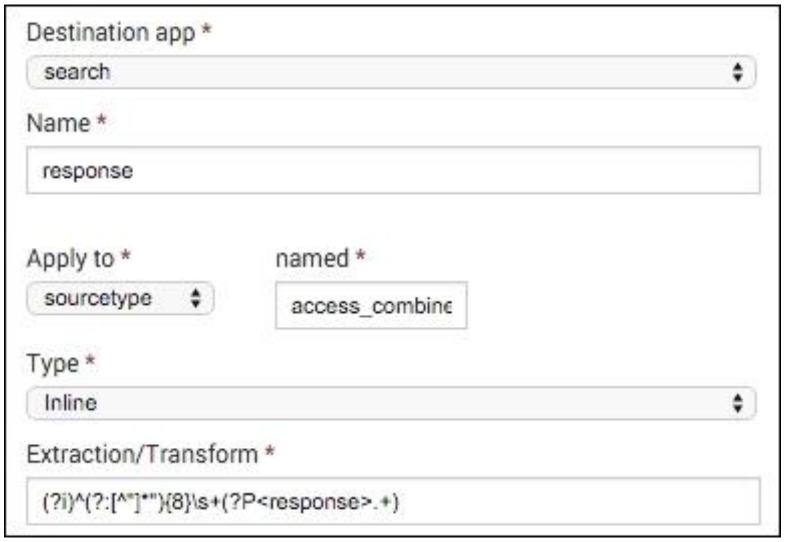
3. Click on the **Field extractions** link.



Field extractions

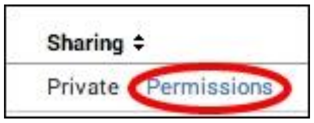
View and edit all field extractions. Add new field extractions and update permissions.

4. Click on **New**.
5. In the **Destination app** field, select the **search** app, and in the **Name**field, enter `response`. Set the **Apply to** drop-down list to **sourcetype**and the **named** field to `access_combined`. Set the **Type** drop-down list to **Inline**, and for the **Extraction/Transform** field, carefully enter the `(?i)^(?:[^"]*"){8}\s+(?P<response>.+)` regex.



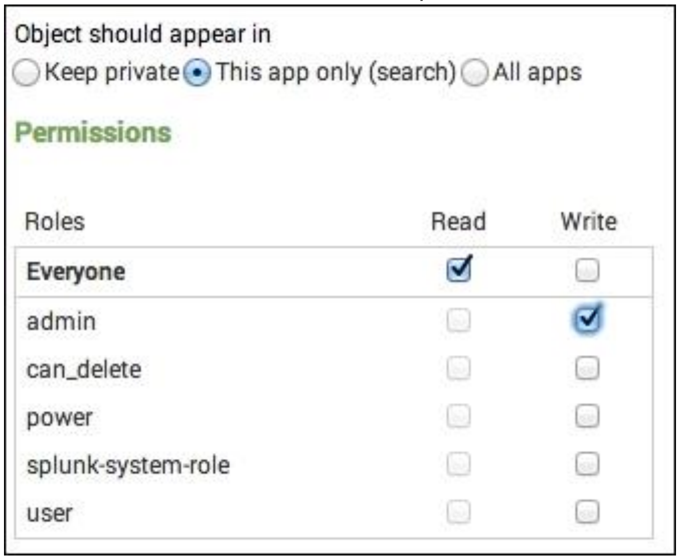
Destination app *	search
Name *	response
Apply to *	sourcetype
named *	access_combine
Type *	Inline
Extraction/Transform *	(?i)^(?:[^"]*"){8}\s+(?P<response>.+)

6. Click on **Save**.
7. On the **Field extractions** listing page, find the recently added extraction, and in the **Sharing** column, click on the **Permissions** link.



Sharing	
Private	Permissions

8. Update the **Object should appear in** setting to **All apps**. In the **Permissions** section, for the **Read** column, check **Everyone**, and in the **Write** column, check **admin**. Then, click on **Save**.



Object should appear in

Keep private [✓ This app only (search)] All apps

Permissions

Roles	Read	Write
Everyone	✓	
admin		✓
can_delete		
power		
splunk-system-role		
user		

9. Navigate to the Splunk search screen and enter the following search over the **Last 60 minutes** time range:

```
index=main sourcetype=access_combined
```

10. You should now see a field called **response extracted** on the left-hand side of the search screen under the **Interesting Fields** section.

How it works…

All field extractions are maintained in the `props.conf` and `transforms.conf` configuration files. The stanzas in `props.conf` include an extraction class that leverages regular expressions to extract field names and/or values to be used at search time. The `transforms.conf` file goes further and can be leveraged for more advanced extractions, such as reusing or sharing extractions over multiple sources, source types, or hosts.

Defining event types and tags

Event types in Splunk are a way of categorizing common types of events in your data in order to make them easier to search and report on. One advantage of using event types is that they can assist in applying a common classification to similar events. Event types essentially turn chunks of search criteria into field/value pairs. Tags help you search groups of event data more efficiently and can be assigned to any field/value combination, including event types.

For example, Windows logon events could be given an event type of `windows_logon`, Unix logon events be given an event type of `unix_logon`, and VPN logon events could be given an event type of `vpn_logon`. We could then tag these three event types with a tag of `logon_event`. A simple search for `tag="logon_event"` would then search across the Windows, Unix, and VPN source types and return all the logon events. Alternatively, if we want to search only for Windows logon events, we will search for `eventtype=windows_logon`.

This recipe will show how to define event types and tags for use with the sample data. Specifically, you will define an event type for successful web server events.

NOTE

For more information on event types and tags in Splunk, check out:

- http://docs.splunk.com/Documentation/Splunk/latest/Knowledge/Abouteventtypes
- http://docs.splunk.com/Documentation/Splunk/latest/Knowledge/Abouttagsandaliases

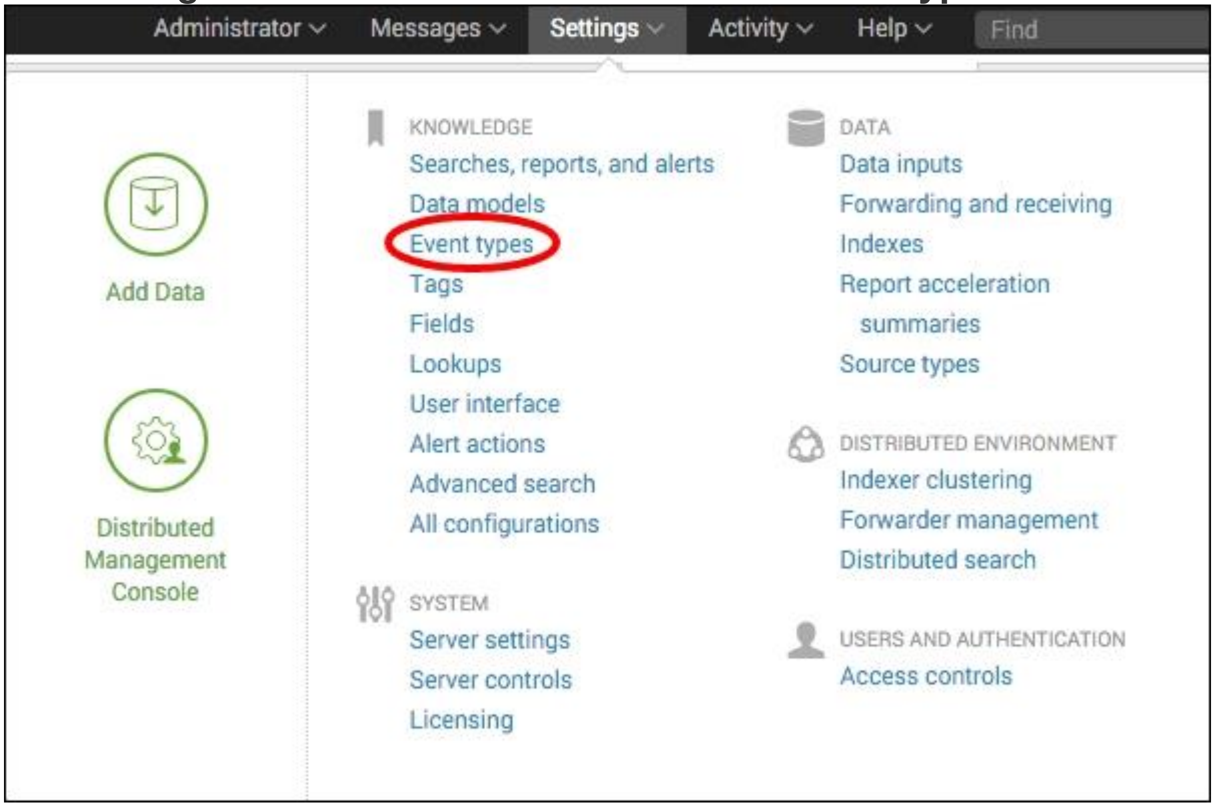
Getting ready

To step through this recipe, you will need a running Splunk server with the operational intelligence sample data loaded. No other prerequisites are required.

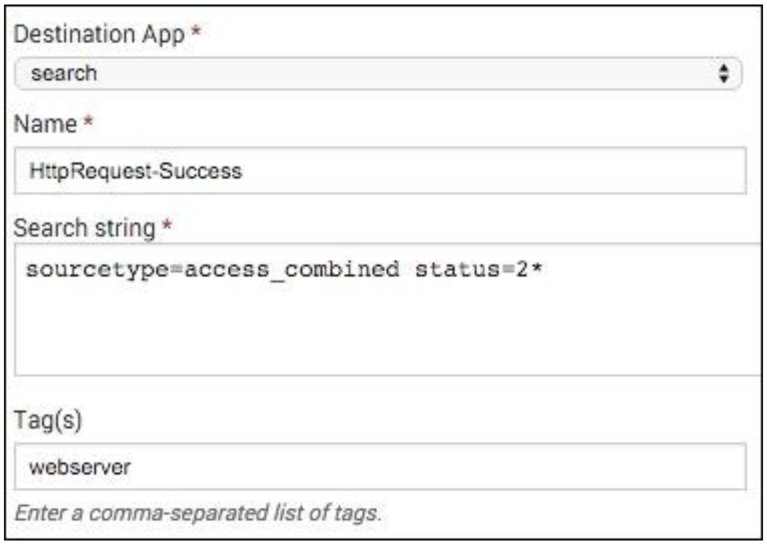
How to do it…

Follow the given steps to define an event type and associated tag:

1. Log in to your Splunk server.
2. From the home launcher in the top right-hand corner, click on the **Settings** menu item and then click on the **Event types** link.



3. Click on the **New** button.
4. In the **Destination App** drop-down list, select **search**. Enter `HttpRequest-Success` in the **Name** field. In the **Search string** text area, enter `sourcetype=access_combined status=2*`. In the **Tag(s)** field, enter `webserver`, and then click on **Save**.


Destination App *

search

Name *

HttpRequest-Success

Search string *

```
sourcetype=access_combined status=2*
```

Tag(s)

webserver

Enter a comma-separated list of tags.

5. The event type is now created. To verify that this worked, you should now be able to search by both the event type and the tag that you created. Navigate to the Splunk search screen in the **Search & Reporting** app and enter the following search over the **Last 60 minutes** time range to verify that the `eventtype` is working:

   ```
   eventtype="HttpRequest-Success"
   ```

6. Enter the following search over the **Last 60 minutes** time range to verify that the tag is working:

   ```
   tag="webserver"
   ```

How it works…

Event types are applied to events at search time and introduce an `eventtype` field with user-defined values that can be used to quickly sift through large amounts of data. An event type is essentially a Splunk search string that is applied against each event to see if there is a match. If the event type search matches the event, the `eventtype` field is added, with the value of the field being the user-defined name for that event type.

The common tag value allows for a grouping of event types. If multiple event types had the same tag, then your Splunk search could just search for that particular tag value, instead of needing to list out each individual event type value.

Event types can be added, modified, and deleted at any time without the need to change or reindex your data, as they are applied at search time.

Event types are stored in `eventtypes.conf` in either the `$SPLUNK_HOME/etc/system/local/` or a custom app directory.

There's more…

While adding of event types and tags can be done through the web interface of Splunk, as outlined in this recipe, there are other approaches to add them in bulk quickly and allow for customization of the many configuration options that Splunk provides.

ADDING EVENT TYPES AND TAGS VIA EVENTTYPES.CONF AND TAGS.CONF

Event types in Splunk can be manually added to the `eventtypes.conf`configuration files. Edit or create `$SPLUNK_HOME/etc/system/local/eventtypes.conf` and add your event type. You will need to restart Splunk after this:

```
[HttpRequest-Success]
search = status=2*
```

Tags in Splunk can be manually added to the `tags.conf` configuration files. Edit or create `$SPLUNK_HOME/etc/system/local/tags.conf` and add your tag. You will need to restart Splunk after this:

```
[eventtype=HttpRequest-Success]
webserver = enabled
```

TIP

In this recipe, you tagged an event type. However, tags do not always need to be associated with event types. You can tag any field/value combination found in an event. To create new tags independently, click on the **Settings** menu and select **Tags**.