Best Practices for Developing Splunk Apps and Add-ons

Jason Conger

Staff Solutions Architect, Splunk

.conf2016

splunk>

Disclaimer

whoami

jason.conger@splunk.com

@JasonConger

http://www.linkedin.com/in/JasonConger

blogs.splunk.com/author/jconger/
www.JasonConger.com

4+ years at Splunk

Created or consulted on numerous Splunkbase applications

Staff Solutions Architect
Global Strategic Alliances

Agenda

1. Creating a Splunk Application
2. Getting data into Splunk
3. Asking questions of your data with Splunk

"I wish I knew these things before I ever built my first Splunk Application"

- Jason Conger

2

2

2

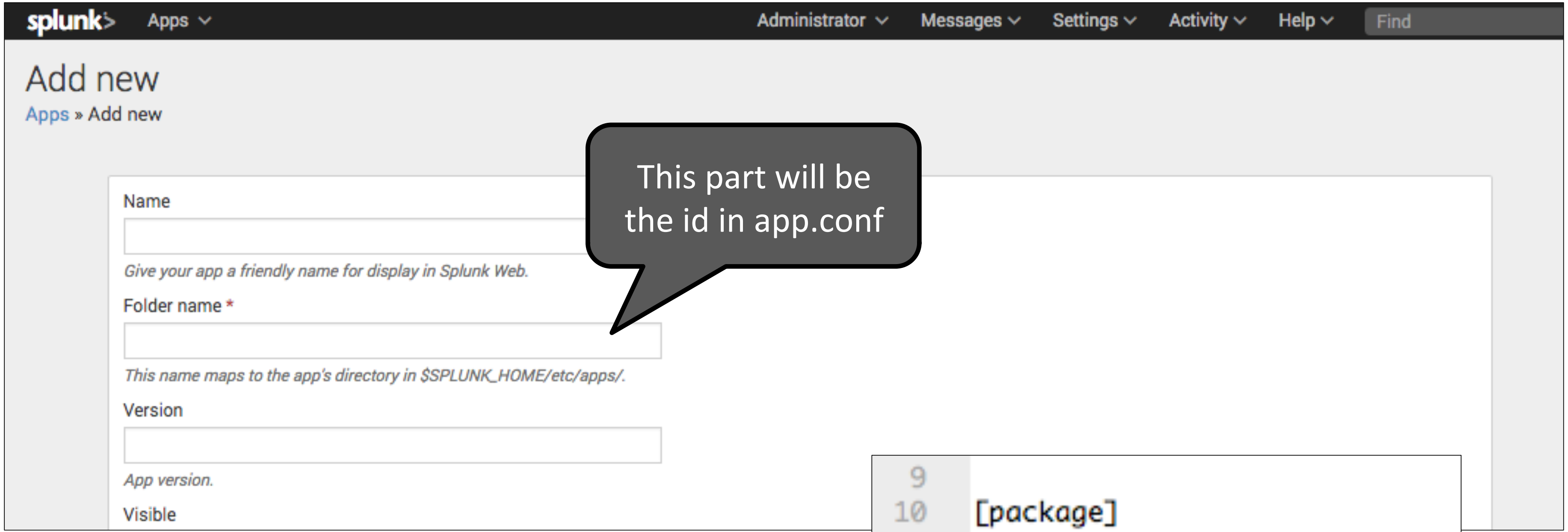
1

Creating An Application

.conf2016

splunk>

Naming the Directory for Your App or Add-on



Add new

Apps » Add new

Name

Give your app a friendly name for display in Splunk Web.

Folder name *

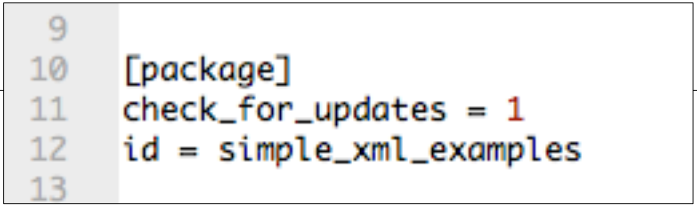
This name maps to the app's directory in $SPLUNK_HOME/etc/apps/.

Version

App version.

Visible

This part will be the id in app.conf



```
[package]
check_for_updates = 1
id = simple_xml_examples
```

Naming the <u>Directory</u> for Your App or Add-on

- ❑ For applications (dashboards, forms, alerts, etc.):
  - Vendor-app-product (example = acme-app-widget)

- ❑ For add-ons (data collection with no dashboards):
  - TA_vendor-product (example: TA_acme-widget)

- ❑ For Enterprise Security add-ons:
  - TA-<datasource> (example: TA-snort)

Note: you may see some other naming standards such as SA or DA out there.

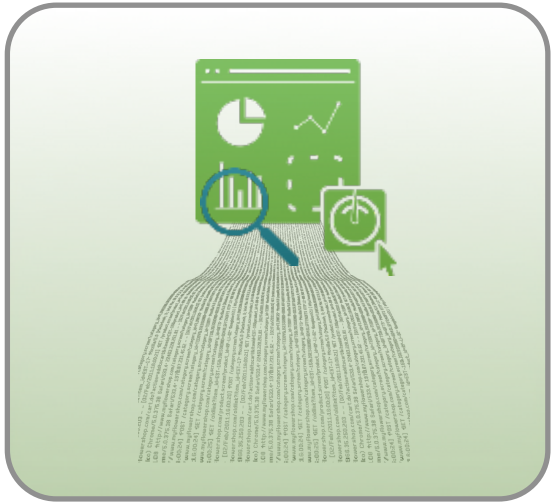
Naming Your App or Add-on

Note: after uploading an application to Splunkbase, the directory name and the “id” parameter in app.conf cannot be changed.

The actual name of the application displayed on the Splunk start screen and on Splunkbase is controlled by a file named app.conf and is independent of the directory name mentioned previously.

App naming guidelines -> http://docs.splunk.com/Documentation/Splunkbase/latest/Splunkbase/Namingguidelines

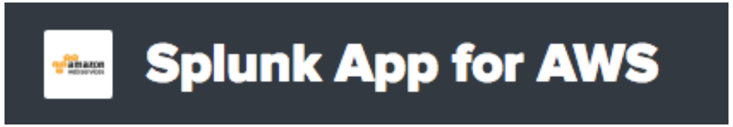
Should You Break Up Your App?


Consolidated App

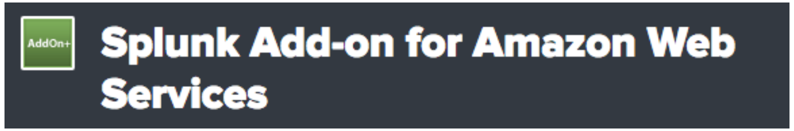
Distributed App

Should You Break Up Your App?

- Do you need to collect data from forwarders?
- Need to share knowledge objects with multiple apps?
- Distributed Environment?
- The Splunk App for AWS is a good example

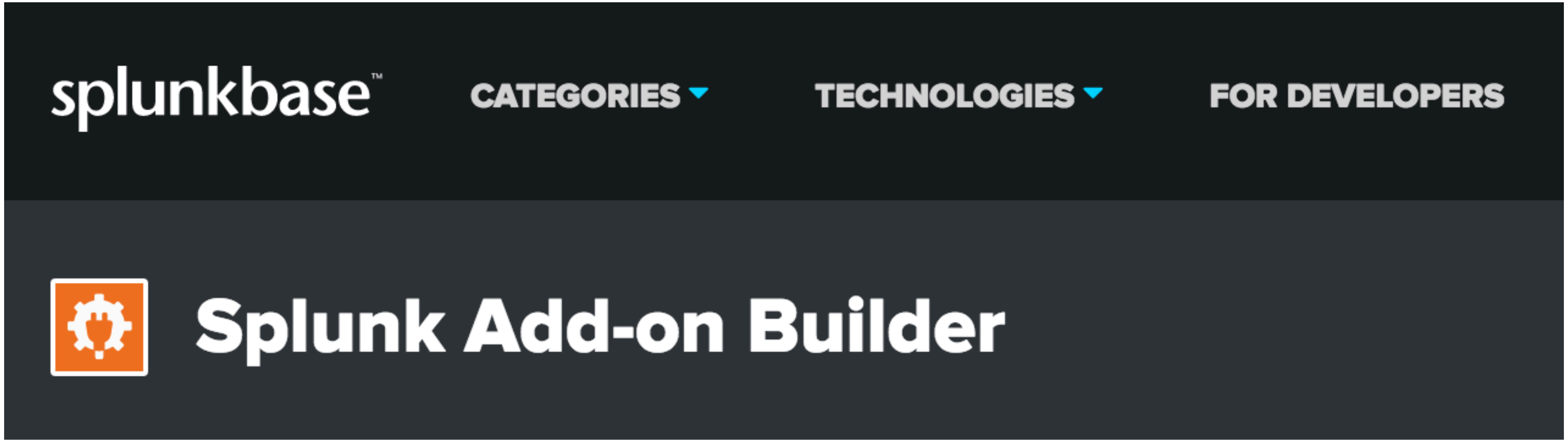

Splunk App for AWS

depends on



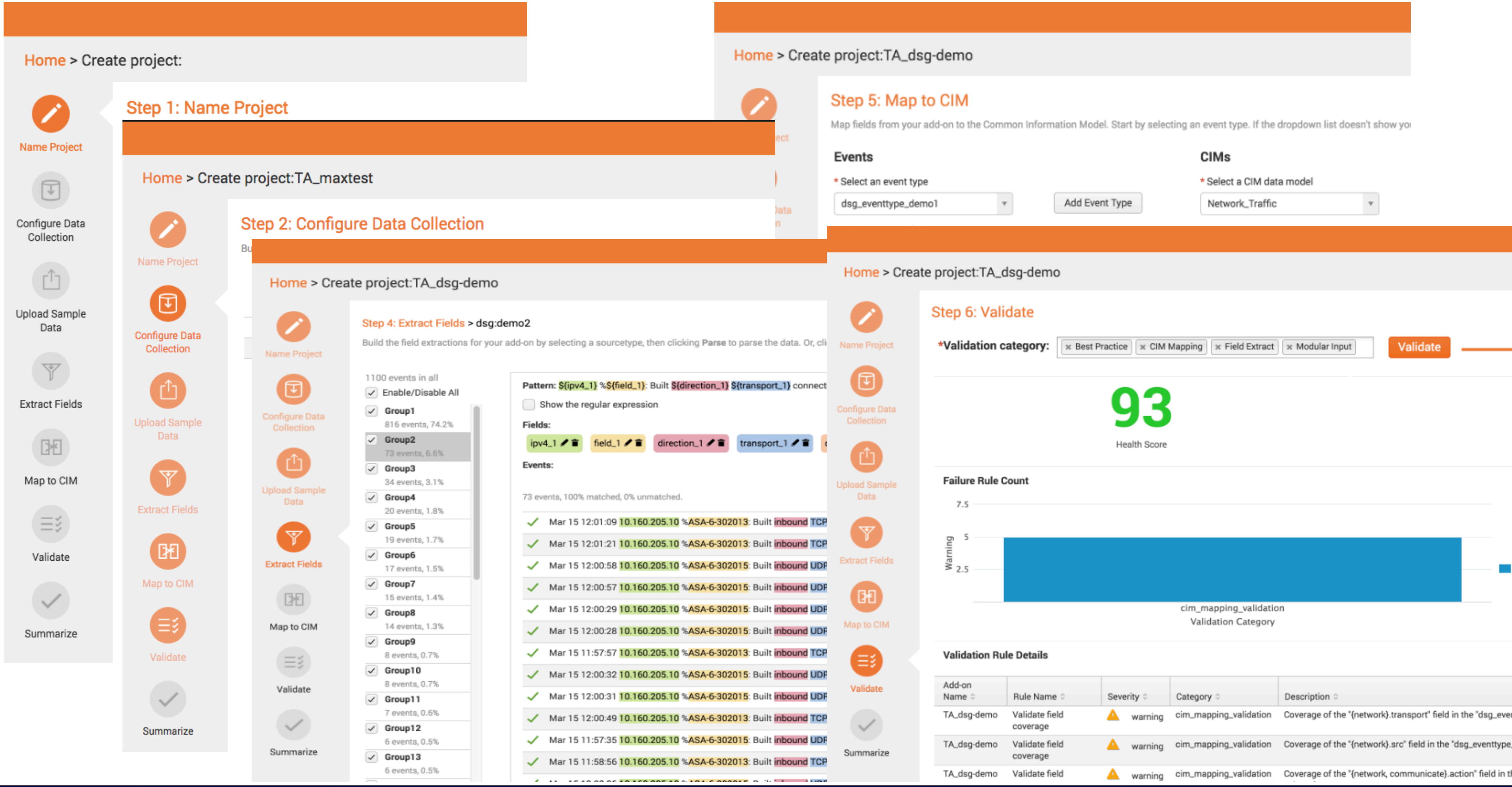
Splunk Add-on for Amazon Web Services

Quick Start = Splunk Add-on Builder



Splunk Add-on Builder

https://splunkbase.splunk.com/app/2962/



Home > Create project:

Name Project
Configure Data Collection
Upload Sample Data
Extract Fields
Map to CIM
Validate
Summarize

Step 1: Name Project

Home > Create project:TA_maxtest

Step 2: Configure Data Collection

Home > Create project:TA_dsg-demo

Step 4: Extract Fields > dsg:demo2

Build the field extractions for your add-on by selecting a sourcetype, then clicking **Parse** to parse the data. Or, cli

1100 events in all

- ✓ Enable/Disable All
- ✓ Group1 — 816 events, 74.2%
- ✓ Group2 — 73 events, 6.6%
- ✓ Group3 — 34 events, 3.1%
- ✓ Group4 — 20 events, 1.8%
- ✓ Group5 — 19 events, 1.7%
- ✓ Group6 — 17 events, 1.5%
- ✓ Group7 — 15 events, 1.4%
- ✓ Group8 — 14 events, 1.3%
- ✓ Group9 — 8 events, 0.7%
- ✓ Group10 — 8 events, 0.7%
- ✓ Group11 — 7 events, 0.6%
- ✓ Group12 — 6 events, 0.5%
- ✓ Group13 — 6 events, 0.5%

Pattern: ${ipv4_1} %${field_1}: Built ${direction_1} ${transport_1} connect

Show the regular expression

Fields: ipv4_1 · field_1 · direction_1 · transport_1

Events:

73 events, 100% matched, 0% unmatched.

```
✓ Mar 15 12:01:09 10.160.205.10 %ASA-6-302013: Built inbound TCP
✓ Mar 15 12:01:21 10.160.205.10 %ASA-6-302013: Built inbound TCP
✓ Mar 15 12:00:58 10.160.205.10 %ASA-6-302015: Built inbound UDP
✓ Mar 15 12:00:57 10.160.205.10 %ASA-6-302015: Built inbound UDP
✓ Mar 15 12:00:29 10.160.205.10 %ASA-6-302015: Built inbound UDP
✓ Mar 15 12:00:28 10.160.205.10 %ASA-6-302015: Built inbound UDP
✓ Mar 15 11:57:57 10.160.205.10 %ASA-6-302013: Built inbound TCP
✓ Mar 15 12:00:32 10.160.205.10 %ASA-6-302015: Built inbound UDP
✓ Mar 15 12:00:31 10.160.205.10 %ASA-6-302015: Built inbound UDP
✓ Mar 15 12:00:49 10.160.205.10 %ASA-6-302013: Built inbound TCP
✓ Mar 15 11:57:35 10.160.205.10 %ASA-6-302015: Built inbound UDP
✓ Mar 15 11:58:56 10.160.205.10 %ASA-6-302013: Built inbound TCP
```

Home > Create project:TA_dsg-demo

Step 5: Map to CIM

Map fields from your add-on to the Common Information Model. Start by selecting an event type. If the dropdown list doesn't show yo

Events

* Select an event type: dsg_eventtype_demo1

Add Event Type

CIMs

* Select a CIM data model: Network_Traffic

Home > Create project:TA_dsg-demo

Step 6: Validate

***Validation category:** Best Practice · CIM Mapping · Field Extract · Modular Input

Validate

93

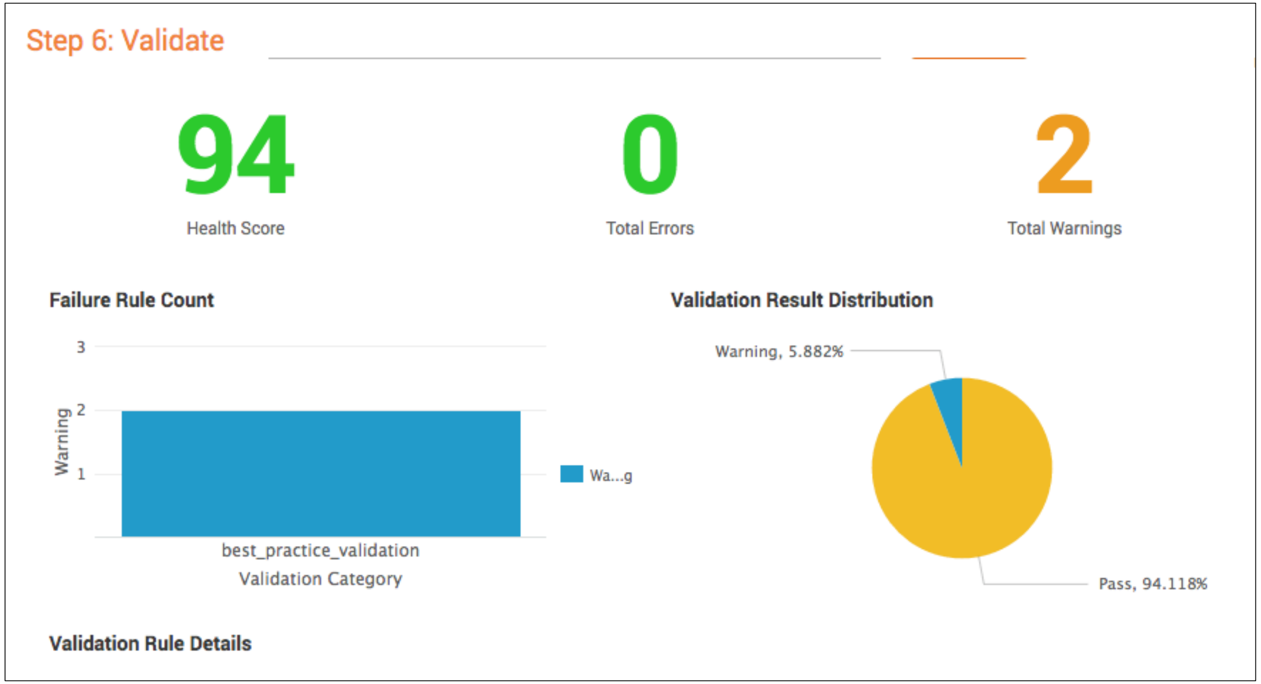
Health Score

Failure Rule Count

Validation Rule Details

Add-on Name	Rule Name	Severity	Category	Description
TA_dsg-demo	Validate field coverage	⚠ warning	cim_mapping_validation	Coverage of the "{network}.transport" field in the "dsg_eve
TA_dsg-demo	Validate field coverage	⚠ warning	cim_mapping_validation	Coverage of the "{network}.src" field in the "dsg_eventtype_
TA_dsg-demo	Validate field	⚠ warning	cim_mapping_validation	Coverage of the "{network, communicate}.action" field in th

Use the Builder on Existing Content Too


Step 6: Validate

94
Health Score

0
Total Errors

2
Total Warnings

Failure Rule Count

Validation Result Distribution

Validation Rule Details

Note: you may get some inapplicable warnings for apps since this version is mainly about add-ons.

2

Getting Data In

.conf2016

splunk>

Getting Data In

Reaching out to get data

- Reading files on a disk
- Windows Inputs
  - Perfmon
  - Event Logs
  - Registry
  - WMI
- Scripts*
- Modular inputs*

Listening for data

- TCP
- UDP
- HTTP
- Stream
- Scripts*
- Modular inputs*

* Scripts and modular inputs can really do either depending on what you code

Best Practices for Logging Data to be Consumed by Splunk

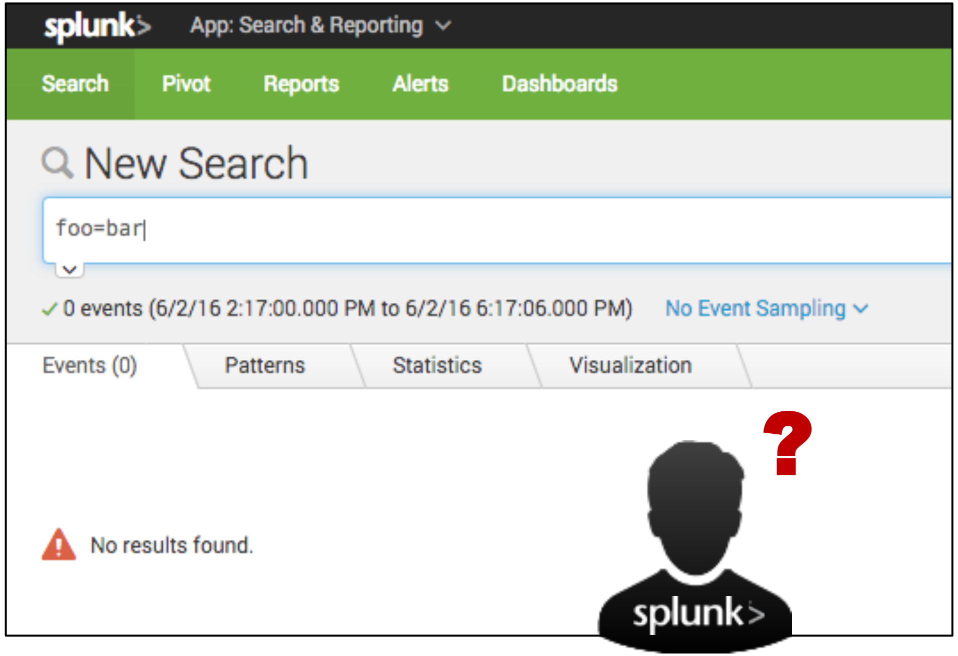
- Log in text format
- Start the log line event with a time stamp
- Use clear key-value pairs
- Create events that humans can read
- Use unique identifiers
- Keep multi-line events to a minimum
- Use JSON (JavaScript Object Notation) format

http://dev.splunk.com/view/logging-best-practices/SP-CAAADP6

Best Practices for Writing Data to an Index

Write to the default "main" index

main



```
foo=bar
```

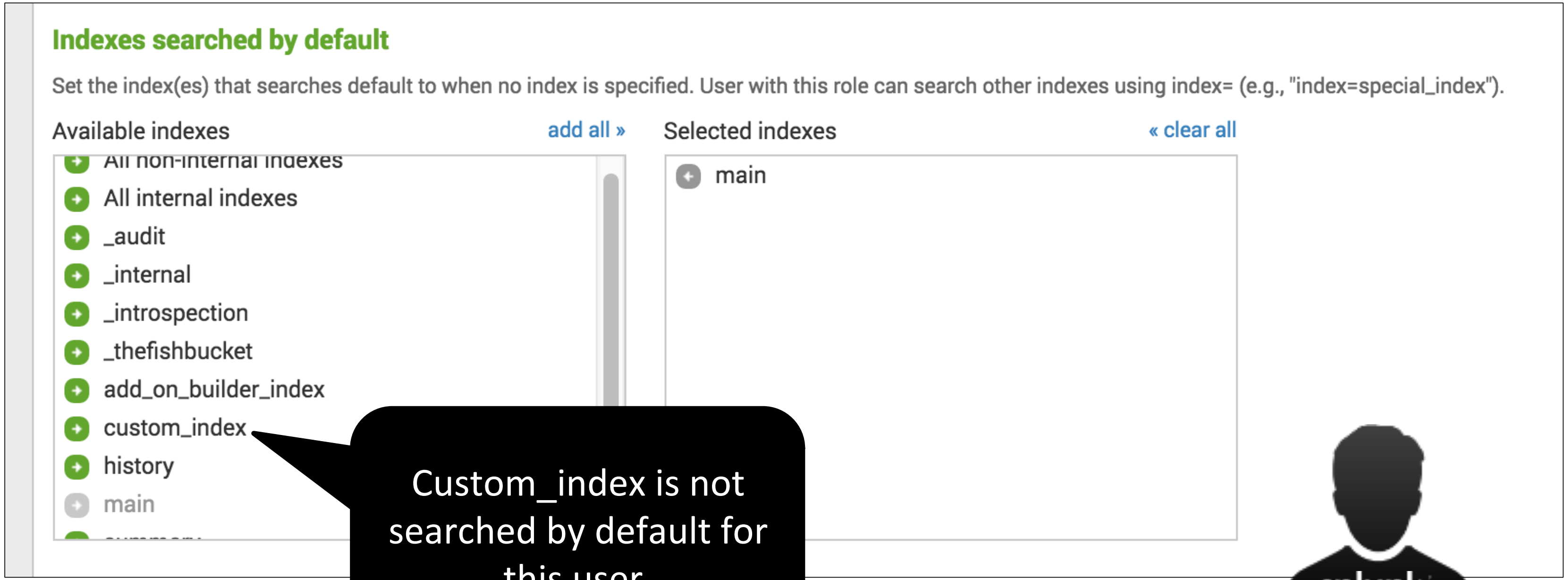
✓ 0 events (6/2/16 2:17:00.000 PM to 6/2/16 6:17:06.000 PM)

No results found.

custom index
contains foo=bar
data

Best Practices for Writing Data to an Index

Write to the default "main" index



Indexes searched by default

Set the index(es) that searches default to when no index is specified. User with this role can search other indexes using index= (e.g., "index=special_index").

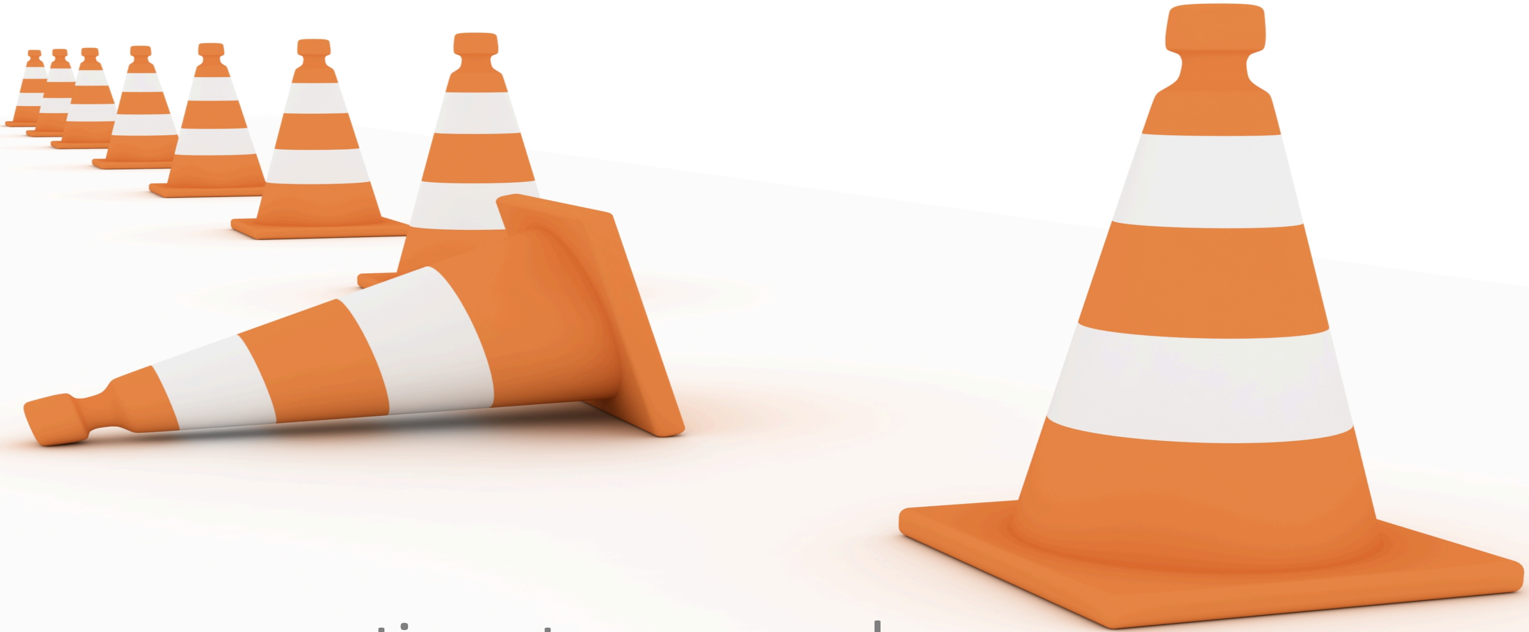
Available indexes — add all »

- All non-internal indexes
- All internal indexes
- _audit
- _internal
- _introspection
- _thefishbucket
- add_on_builder_index
- custom_index
- history
- main

Selected indexes — « clear all

- main

Custom_index is not searched by default for this user.



There are an exceptions to every rule

Exceptions to using the “main” Index

- Testing – writing data to a test index during development allows the developer to quickly and easily clear out all events in the index without impacting other events elsewhere.

```
$SPLUNK_HOME/bin/splunk clean eventdata custom_index
```

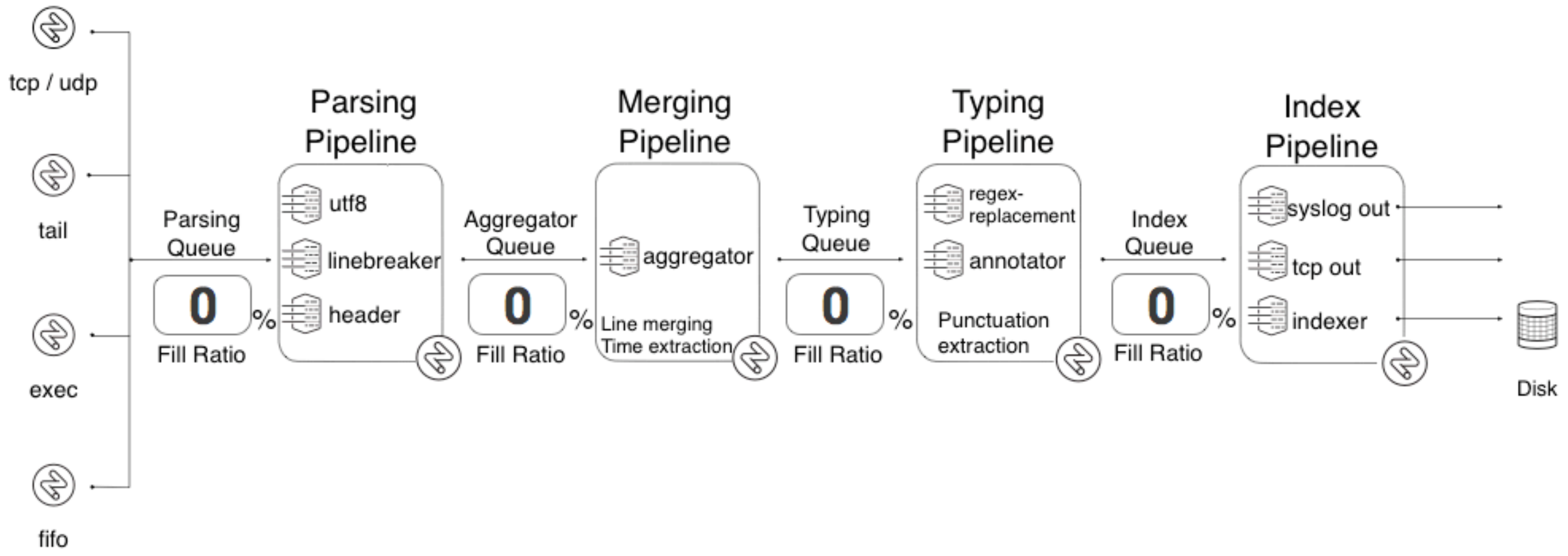
- Retention – data retention/aging is controlled on the index level. Some administrators may want to have custom retention policies based on the type of data.
- Security – using Splunk’s RBAC, the administrator can control who sees what data.

Exceptions to using the "main" Index

- Testing – writing data to a test index during development allows the developer to quickly and easily clear out all events in the index without impacting other events elsewhere.
- Retention – data retention/aging is controlled on the index level. Some administrators may want to have custom retention policies based on the type of data.
- Security – using Splunk's RBAC, the administrator can control who sees what data.

The last 2 exception decisions should be made by the Splunk admin – not the developer.

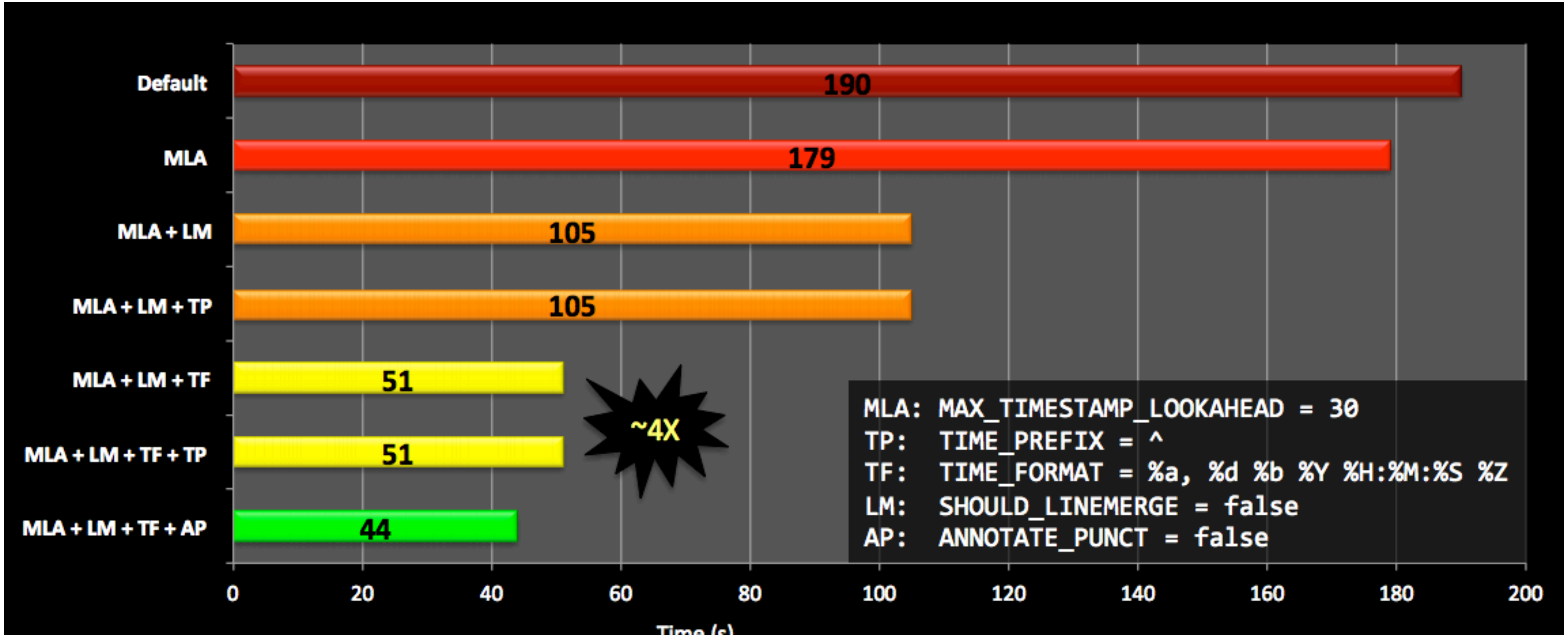
Get to Know Your Pipelines



Useful Index Time Processing Attributes

Event Breaking	LINE_BREAKER	`<where to break the stream>`
	SHOULD_LINEMERGE	`<enable/disable merging>`
Timestamp Extraction	MAX_TIMESTAMP_LOOKAHEAD	`<# chars in to look for ts>`
	TIME_PREFIX	`<pattern before ts>`
	TIME_FORMAT	`<strptime format string to extract ts>`
Typing	ANNOTATE_PUNCT	`<enable/disable punct:: extraction>`

Useful Index Time Processing Attributes



```
MLA: MAX_TIMESTAMP_LOOKAHEAD = 30
TP:  TIME_PREFIX = ^
TF:  TIME_FORMAT = %a, %d %b %Y %H:%M:%S %Z
LM:  SHOULD_LINEMERGE = false
AP:  ANNOTATE_PUNCT = false
```

HT: Dritan Bitincka

Adding Inputs

Scripted versus Modular Inputs

Feature	Scripted Inputs	Modular Inputs
End user configuration	Inline arguments Often requires editing text configuration files	User interface provided in the Splunk Web interface. This makes the input “look and feel” as if it were a native Splunk feature.
Multi-platform support	No	Yes You can package your script to include versions for separate platforms.
Custom REST endpoints	No	Yes Modular inputs can be access and manipulated using Splunk REST endpoints.
Endpoint permissions	N/A	Access implemented using Splunk Enterprise capabilities.

More complete information can be found on the Splunk documentation page.

Scripted versus Modular Inputs

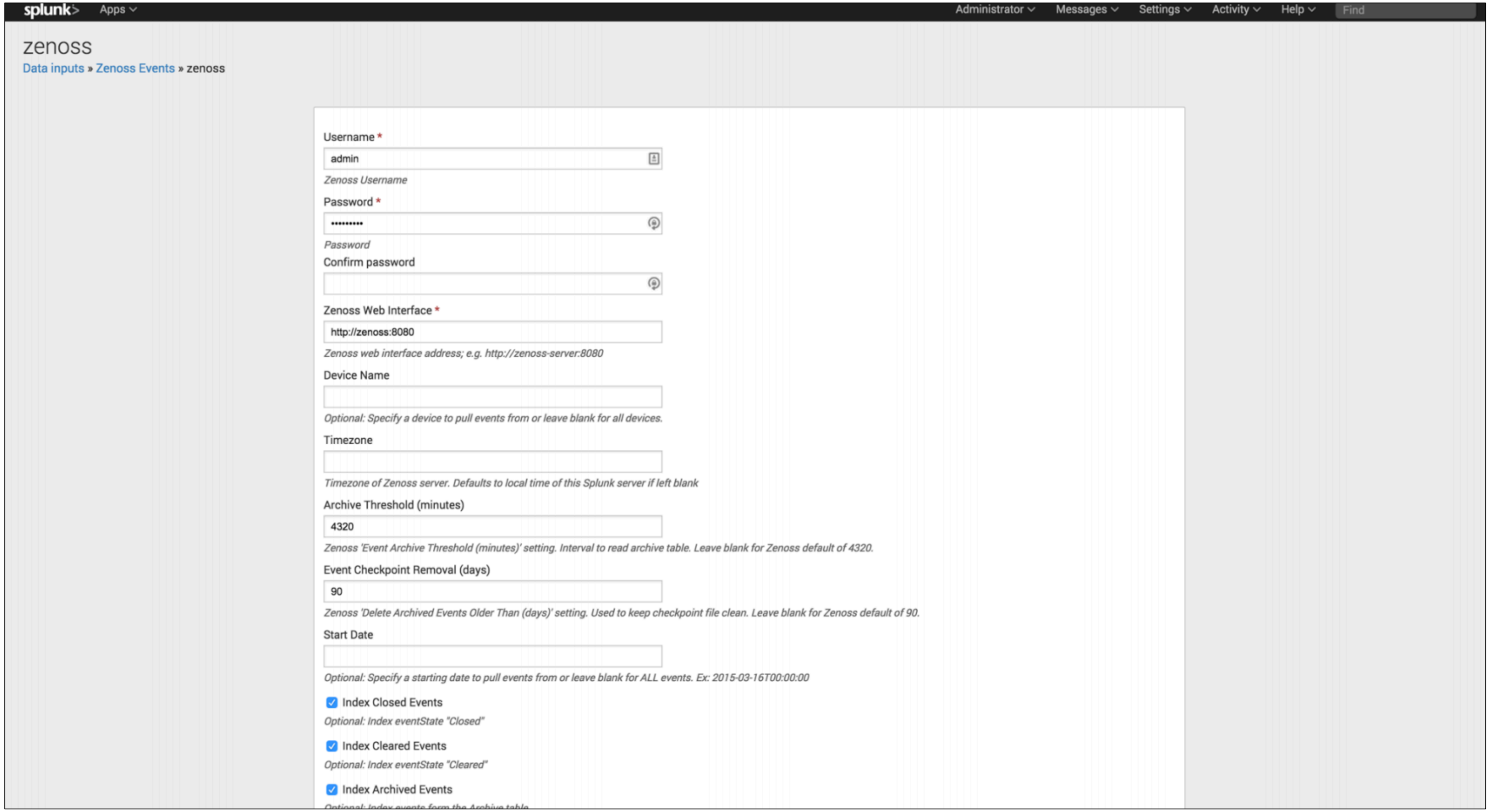
Scripted inputs are more suited for trivial tasks such as running an OS command (like ***top*** for *nix or ***Get-Process*** from Windows PowerShell) and sending the output to Splunk.

Modular inputs are more suited for tasks that require end user setup or more advanced event processing. Calling a REST API with parameters is a good example of when to use a modular input.

This or…

```
script://./bin/zenoss_wrapper.sh -u admin -p password -a h8p://
zenoss:8080 -z America/Los_Angeles -t 4320 -r 90 -s
2015-03-16T00:00:00 -index-closed-events 1 -index-cleared-events 1 -
index-archived-events 1 -index-suppressed-events 1 -index-
repeatevents 1]

sourcetype = zenoss-events
interval = 60
index = zenoss
```



zenoss

Data inputs » Zenoss Events » zenoss

Username *

admin

Zenoss Username

Password *

•••••••••

Password

Confirm password

Zenoss Web Interface *

http://zenoss:8080

Zenoss web interface address; e.g. http://zenoss-server:8080

Device Name

Optional: Specify a device to pull events from or leave blank for all devices.

Timezone

Timezone of Zenoss server. Defaults to local time of this Splunk server if left blank

Archive Threshold (minutes)

4320

Zenoss 'Event Archive Threshold (minutes)' setting. Interval to read archive table. Leave blank for Zenoss default of 4320.

Event Checkpoint Removal (days)

90

Zenoss 'Delete Archived Events Older Than (days)' setting. Used to keep checkpoint file clean. Leave blank for Zenoss default of 90.

Start Date

Optional: Specify a starting date to pull events from or leave blank for ALL events. Ex: 2015-03-16T00:00:00

✓ Index Closed Events

Optional: Index eventState "Closed"

✓ Index Cleared Events

Optional: Index eventState "Cleared"

✓ Index Archived Events

Optional: Index events form the Archive table

HT: Scott Haskell

Scripted and Modular Input Best Practices

Do not hard code paths

Example (Python):

os.path.join(os.environ["SPLUNK_HOME"],'etc','apps', APP_NAME)

Example (PowerShell):

Join-Path -path (get-item env:\SPLUNK_HOME).value "Splunk\etc
\apps"

Scripted and Modular Input Best Practices

Use Error Trapping (so that you can search them in the _internal index)

```
import logging
try:
      Some code that may fail like opening a
file
except IOError, err:
      logging.error('%s - ERROR - File may not
exist %s\n' % (time.strftime("%Y-%m-%d %H:%M:
%S"), str(err)))
      pass
```

Scripted and Modular Input Best Practices

<u>Error Trapping</u> (you can use stderr too)

```
try:
      Some code that may fail like opening a
file

except IOError, err:
      sys.stderr.write('%s - ERROR - File may
not exist %s\n' % (time.strftime("%Y-%m-%d %H:
%M:%S"), str(err)))
      pass
```

Scripted and Modular Input Best Practices

<u>Use Splunk methods to read cascaded settings</u>

Example (Python):

```
import splunk.clilib.cli_common

def __init__(self,obj):
      self.object = obj
      self.settings =
splunk.clilib.cli_common.getConfStanza("acme",
"default")
```

- Give more explanation on previous slide
- Mention someone trying to read from default and write to local
- Maybe mention btool too

Scripted and Modular Input Best Practices

<u>Disable any inputs by default</u>

inputs.conf:

```
[my_stanza]
disabled = 1
```

Scripted Inputs Best Practices

<u>Test Scripts using Splunk CMD</u>

Mac:

```
/Applications/Splunk/bin/splunk cmd python /Applications/
Splunk/etc/apps/<your app>/bin/<your script>
```

Windows:

```
C:\Program Files\Splunk\bin\splunk.exe cmd C:\Program Files
\Splunk\etc\apps\<your app>\bin\<your script>
```

Modular Inputs Best Practices

Use Splunk SDKs (these abstract a lot of code for you)

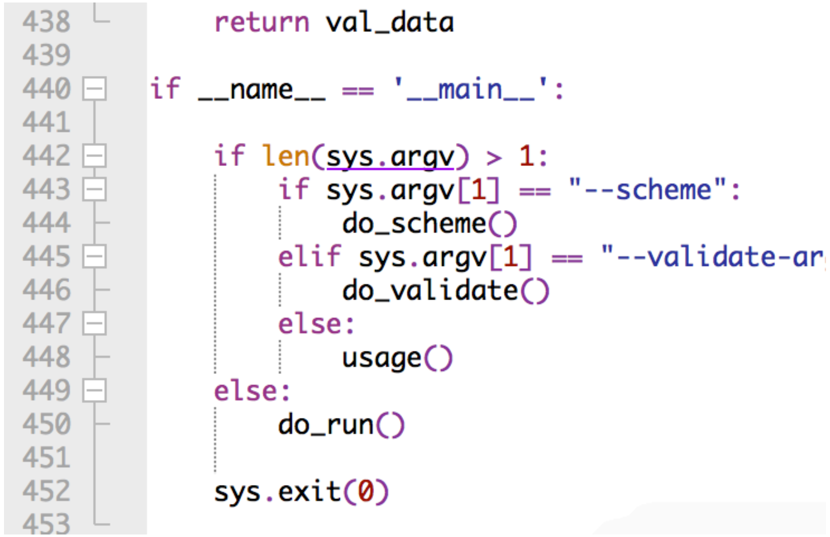
Python http://dev.splunk.com/view/python-sdk/SP-CAAAER3

C# http://dev.splunk.com/view/csharp-sdk/SP-CAAAEQH

Java http://dev.splunk.com/view/java-sdk/SP-CAAAER2

Modular Input SDKs

Before = 453 lines



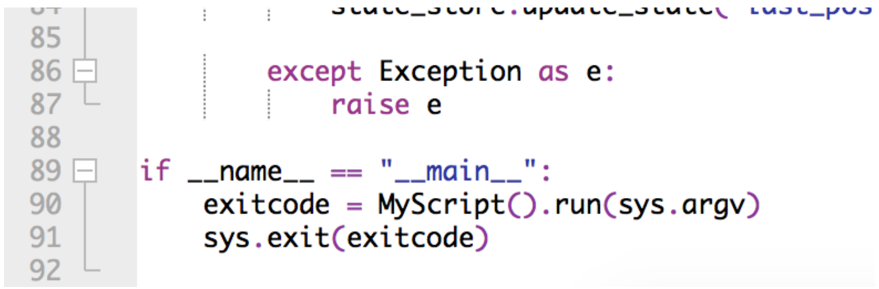
```
    return val_data

if __name__ == '__main__':

    if len(sys.argv) > 1:
        if sys.argv[1] == "--scheme":
            do_scheme()
        elif sys.argv[1] == "--validate-ar
            do_validate()
        else:
            usage()
    else:
        do_run()

    sys.exit(0)
```

After = 92 lines



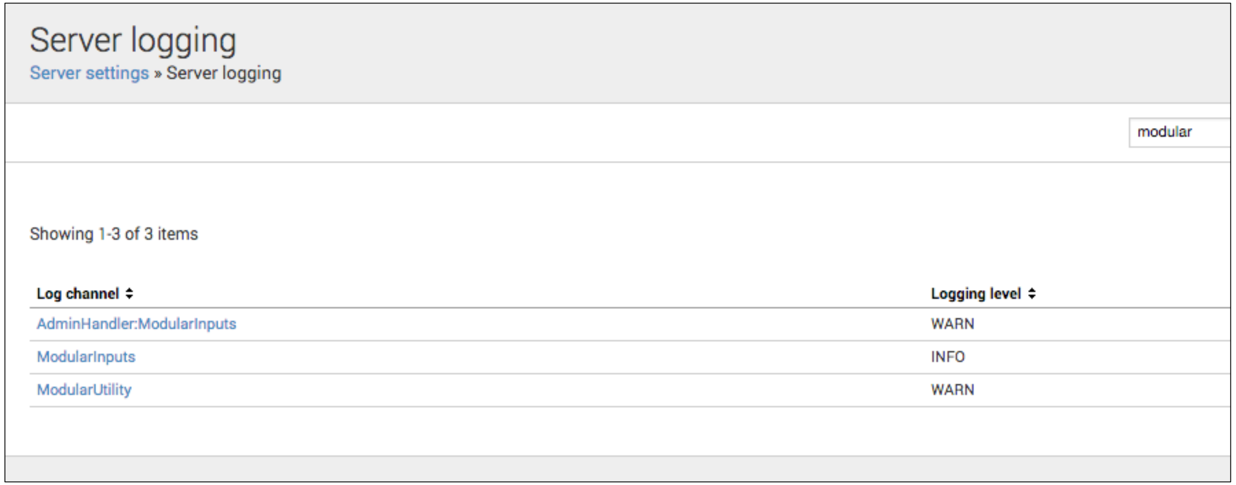
```
                state_store.update_state( last_pos

        except Exception as e:
            raise e

if __name__ == "__main__":
    exitcode = MyScript().run(sys.argv)
    sys.exit(exitcode)
```

Modular Input SDK Logging

By default, only INFO and higher events are logged to _internal.



Server logging

Server settings » Server logging

modular

Showing 1-3 of 3 items

Log channel	Logging level
AdminHandler:ModularInputs	WARN
ModularInputs	INFO
ModularUtility	WARN

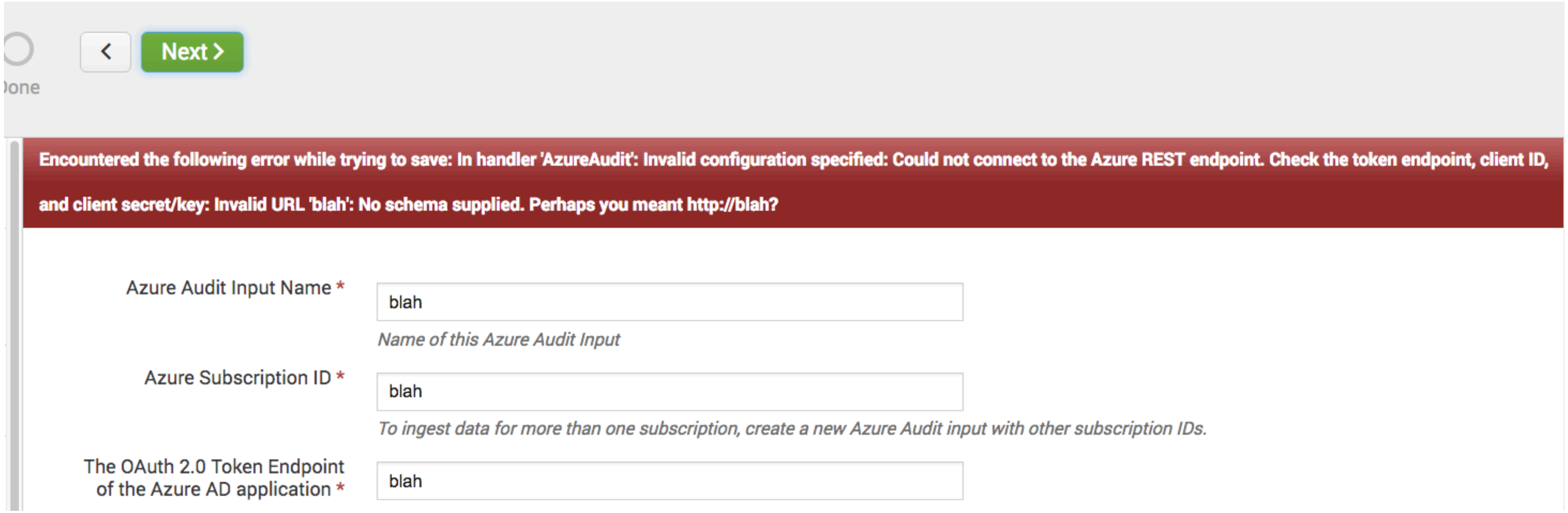
Modular Inputs Best Practices

Validate User Input

```
# Try to connect to the Azure API to validate the given arguments
work
try:
    access_token = get_token_from_client_credentials(
        endpoint = val_data["token_endpoint"],
        client_id = val_data["client_id"],
        client_secret = val_data["client_secret"])
except Exception, e:
    raise Exception, "Could not connect to the Azure REST endpoint.
Check the token endpoint, client ID, and client secret/key: %s" %
str(e)
```

Modular Inputs Best Practices

Validate User Input



Encountered the following error while trying to save: In handler 'AzureAudit': Invalid configuration specified: Could not connect to the Azure REST endpoint. Check the token endpoint, client ID, and client secret/key: Invalid URL 'blah': No schema supplied. Perhaps you meant http://blah?

Azure Audit Input Name *	blah *Name of this Azure Audit Input*
Azure Subscription ID *	blah *To ingest data for more than one subscription, create a new Azure Audit input with other subscription IDs.*
The OAuth 2.0 Token Endpoint of the Azure AD application *	blah

Modular Inputs Best Practices

Test Inputs using Splunk CMD

Example (real):

```
/Applications/Splunk/bin/splunk cmd splunkd print-modinput-config AzureDiagnostics AzureDiagnostics://gsa1892 | /Applications/Splunk/bin/splunk cmd python /Applications/Splunk/etc/apps/TA_Azure/bin/AzureDiagnostics.py
```

Modular Inputs Best Practices

Test Inputs using Splunk CMD

Name of the input

Instance of the input

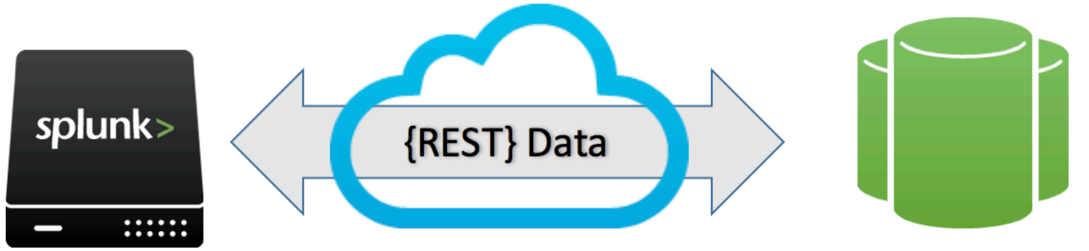
Example (real):

```
/Applications/Splunk/bin/splunk cmd splunkd print-modinput-
config AzureDiagnostics AzureDiagnostics://gsa1892 | /
Applications/Splunk/bin/splunk cmd python /Applications/
Splunk/etc/apps/TA_Azure/bin/AzureDiagnostics.py
```

Input code

Modular Inputs Best Practices

<u>Use the checkpoint parameter to persist data</u>



<u>Checkpoint File</u>

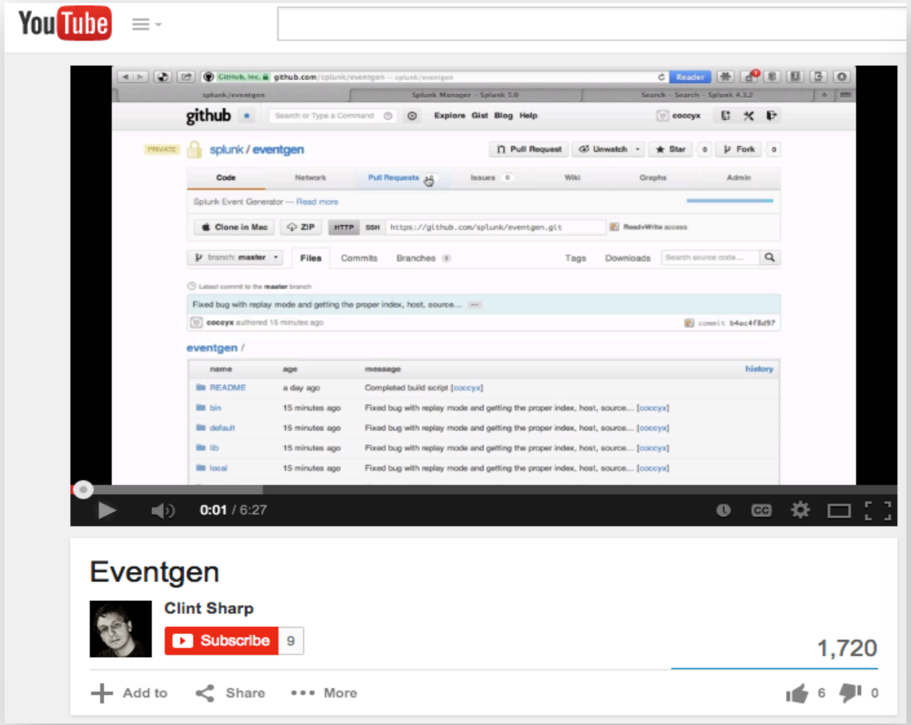
1st run: position = 0
2nd run: position = 5
3rd run: position = 10
Nth run: position = x

<u>3rd Party Data</u>

1st run: returned 0 – 4
2nd run: returned 5 – 9
3rd run: returned 10 - x

http://blogs.splunk.com/2016/05/11/splunking-continuous-rest-data/

Eventgen



splunk / eventgen

Splunk Event Generator — Read more

https://github.com/splunk/eventgen.git

Fixed bug with replay mode and getting the proper index, host, source...

coccyx authored 15 minutes ago

commit b4ec4f8d97

eventgen /

name	age	message
README	a day ago	Completed build script [coccyx]
bin	15 minutes ago	Fixed bug with replay mode and getting the proper index, host, source... [coccyx]
default	15 minutes ago	Fixed bug with replay mode and getting the proper index, host, source... [coccyx]
lib	15 minutes ago	Fixed bug with replay mode and getting the proper index, host, source... [coccyx]
local	15 minutes ago	Fixed bug with replay mode and getting the proper index, host, source... [coccyx]

0:01 / 6:27

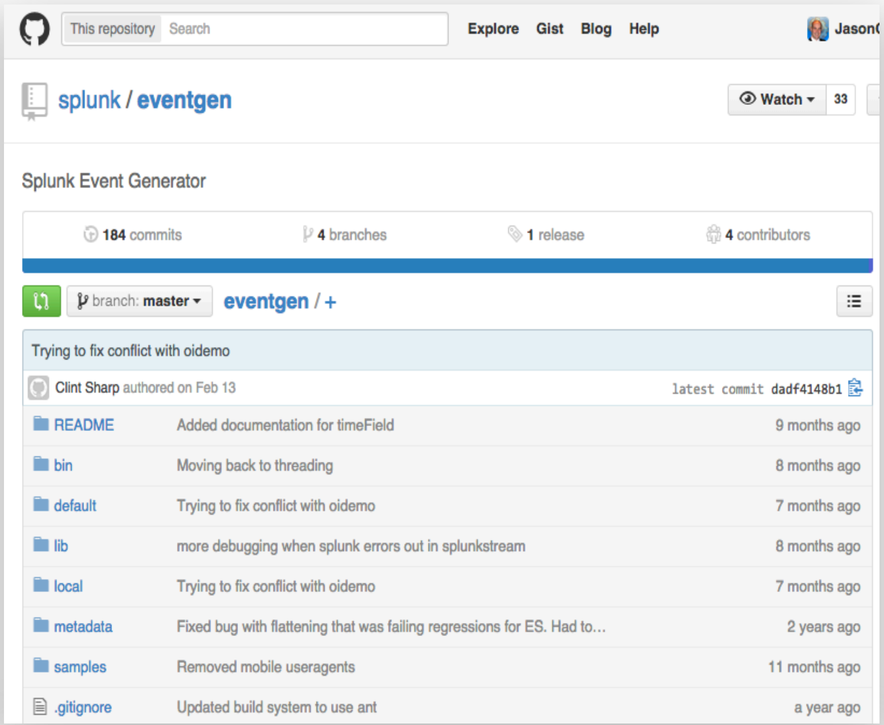
Eventgen

Clint Sharp

Subscribe 9

1,720

Add to · Share · More · 6 · 0



splunk / eventgen

Watch 33

Splunk Event Generator

184 commits · 4 branches · 1 release · 4 contributors

branch: master · eventgen / +

Trying to fix conflict with oidemo

Clint Sharp authored on Feb 13 · latest commit dadf4148b1

README	Added documentation for timeField	9 months ago
bin	Moving back to threading	8 months ago
default	Trying to fix conflict with oidemo	7 months ago
lib	more debugging when splunk errors out in splunkstream	8 months ago
local	Trying to fix conflict with oidemo	7 months ago
metadata	Fixed bug with flattening that was failing regressions for ES. Had to...	2 years ago
samples	Removed mobile useragents	11 months ago
.gitignore	Updated build system to use ant	a year ago

Anonymize Eventgen Samples

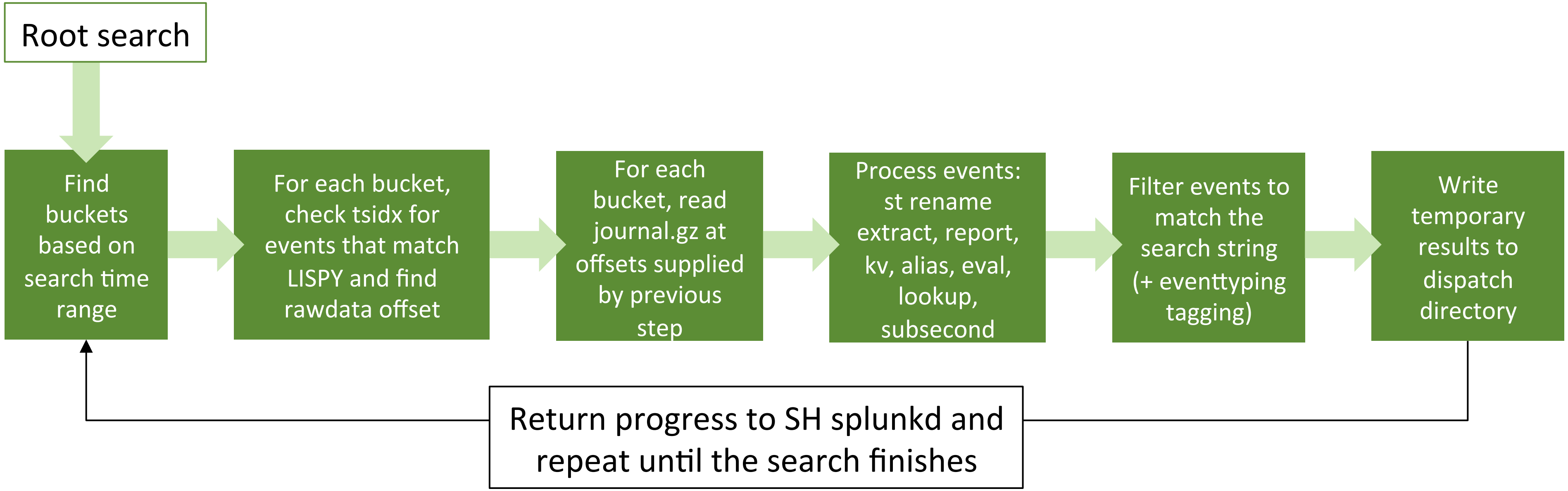
Regex Find and Replace Tools are Your Friend!

Notepad++

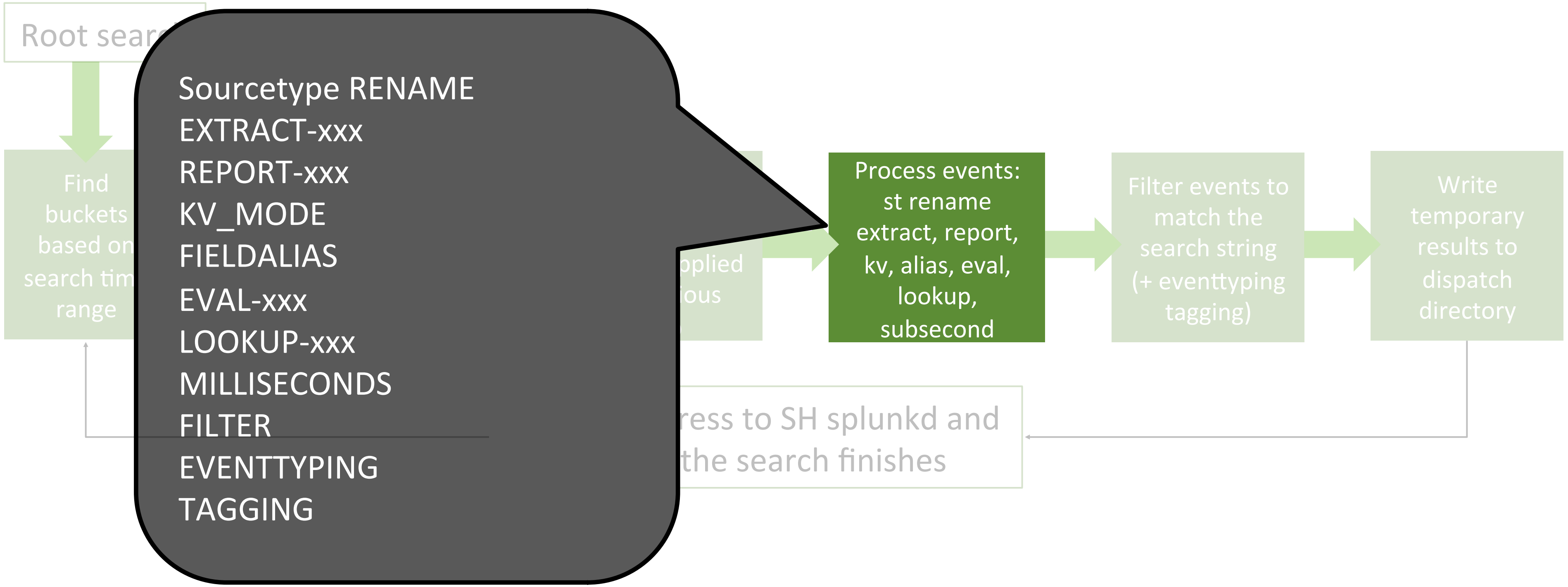
3

Asking Questions of your Data

Get to Know the Search Pipeline


Root search
Find buckets based on search time range
For each bucket, check tsidx for events that match LISPY and find rawdata offset
For each bucket, read journal.gz at offsets supplied by previous step
Process events: st rename extract, report, kv, alias, eval, lookup, subsecond
Filter events to match the search string (+ eventtyping tagging)
Write temporary results to dispatch directory
Return progress to SH splunkd and repeat until the search finishes

Get to Know the Order of Operations


Root sear
Find buckets based on search tim range
pplied
ous
Process events: st rename extract, report, kv, alias, eval, lookup, subsecond
Filter events to match the search string (+ eventtyping tagging)
Write temporary results to dispatch directory
ress to SH splunkd and
the search finishes

Sourcetype RENAME
EXTRACT-xxx
REPORT-xxx
KV_MODE
FIELDALIAS
EVAL-xxx
LOOKUP-xxx
MILLISECONDS
FILTER
EVENTTYPING
TAGGING

Parameterize Root Searches

marcros.conf example:

```
[acme_index]
definition = index=acme
```

Example search using macro:

```
`acme_index` sourcetype=widiget |
stats count
```

Remember that main index thing earlier?

Get to Know Distributed Search

<u>macros.conf</u>

```
[my_index]
definition = index=main
```

<u>eventtypes.conf</u>

```
[my_eventtype]
search = `my_index` sourcetype="foo"
```

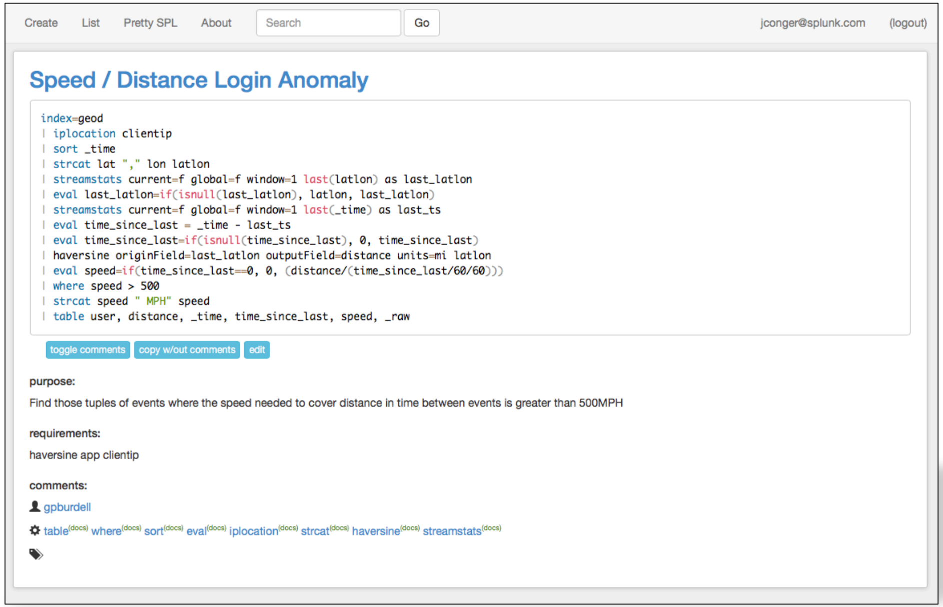
Example search:

```
eventtype=my_eventtype | stats count
```

This will not work in a distributed environment by default.

Get to Know the Big Book of Search

www.bbosearch.com



Speed / Distance Login Anomaly

```
index=geod
| iplocation clientip
| sort _time
| strcat lat "," lon latlon
| streamstats current=f global=f window=1 last(latlon) as last_latlon
| eval last_latlon=if(isnull(last_latlon), latlon, last_latlon)
| streamstats current=f global=f window=1 last(_time) as last_ts
| eval time_since_last = _time - last_ts
| eval time_since_last=if(isnull(time_since_last), 0, time_since_last)
| haversine originField=last_latlon outputField=distance units=mi latlon
| eval speed=if(time_since_last==0, 0, (distance/(time_since_last/60/60)))
| where speed > 500
| strcat speed " MPH" speed
| table user, distance, _time, time_since_last, speed, _raw
```

purpose:

Find those tuples of events where the speed needed to cover distance in time between events is greater than 500MPH

requirements:

haversine app clientip

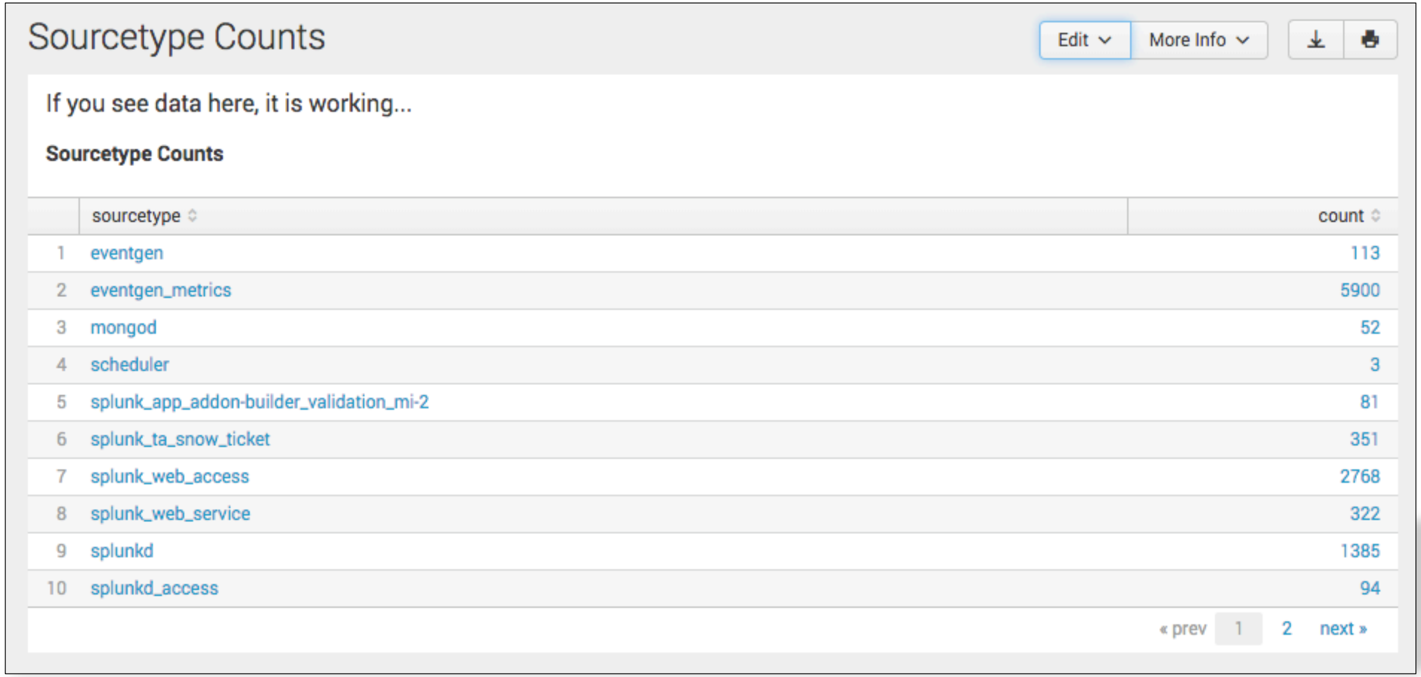
comments:

gpburdell

table(docs) where(docs) sort(docs) eval(docs) iplocation(docs) strcat(docs) haversine(docs) streamstats(docs)

Include Prebuilt Panels

Even if it just to verify the thing is working



Sourcetype Counts

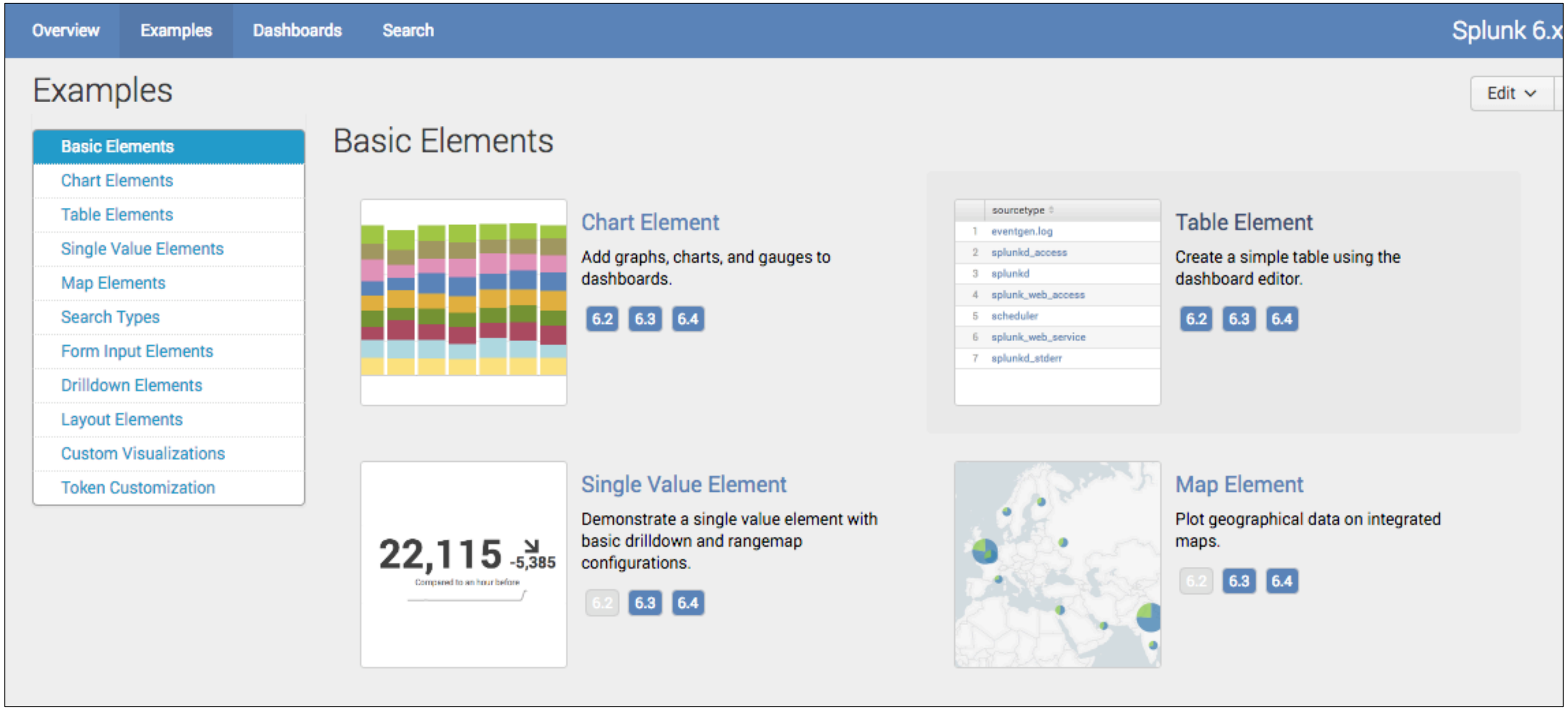
If you see data here, it is working...

Sourcetype Counts

	sourcetype	count
1	eventgen	113
2	eventgen_metrics	5900
3	mongod	52
4	scheduler	3
5	splunk_app_addon-builder_validation_mi-2	81
6	splunk_ta_snow_ticket	351
7	splunk_web_access	2768
8	splunk_web_service	322
9	splunkd	1385
10	splunkd_access	94

« prev 1 2 next »

Use the Dashboards Example App



Add graphs, charts, and gauges to dashboards.

Create a simple table using the dashboard editor.

	sourcetype
1	eventgen.log
2	splunkd_access
3	splunkd
4	splunk_web_access
5	scheduler
6	splunk_web_service
7	splunkd_stderr

22,115 -5,385

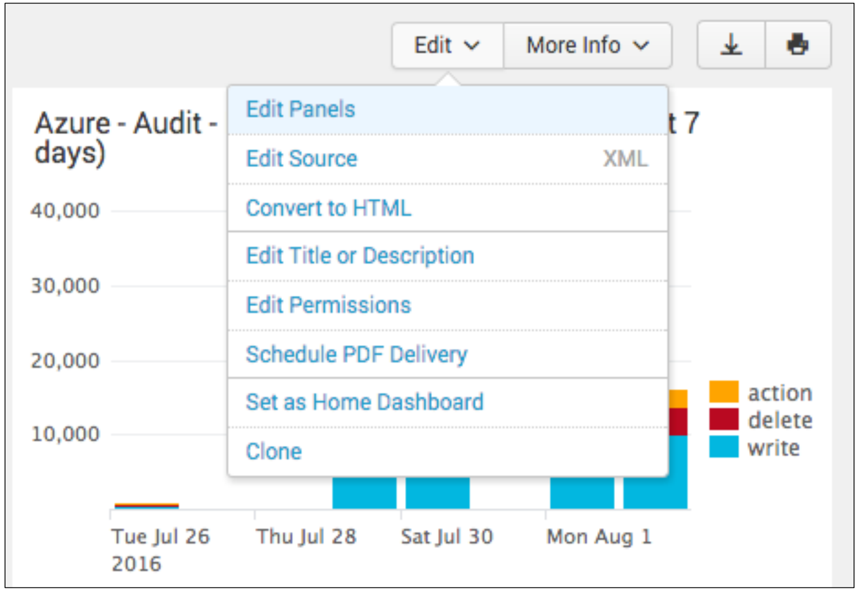
Compared to an hour before

Demonstrate a single value element with basic drilldown and rangemap configurations.

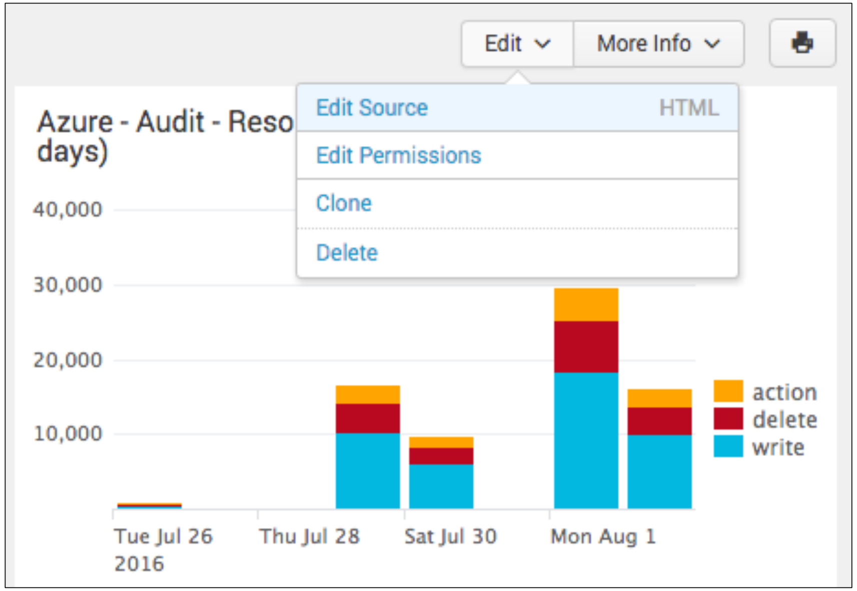
Plot geographical data on integrated maps.

https://splunkbase.splunk.com/app/1603/

Use Simple XML as much as possible

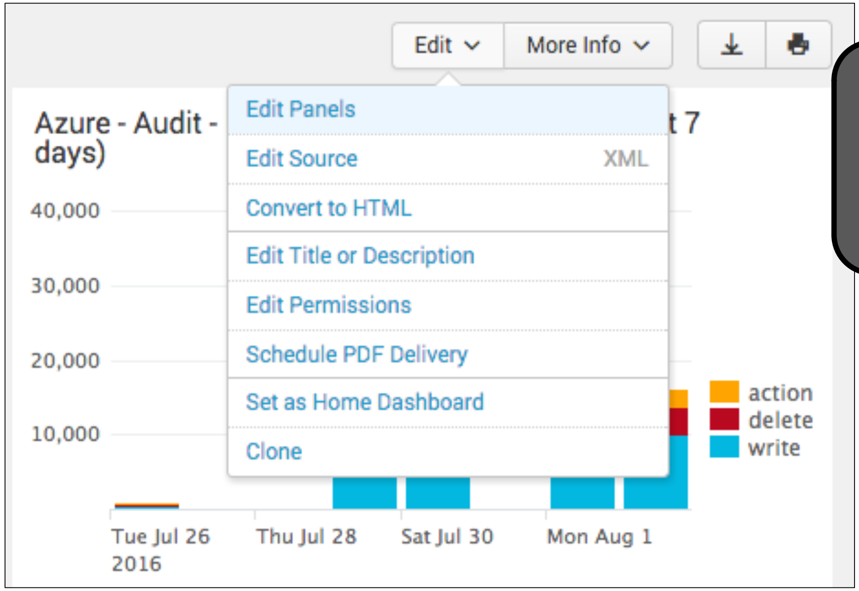


Simple XML

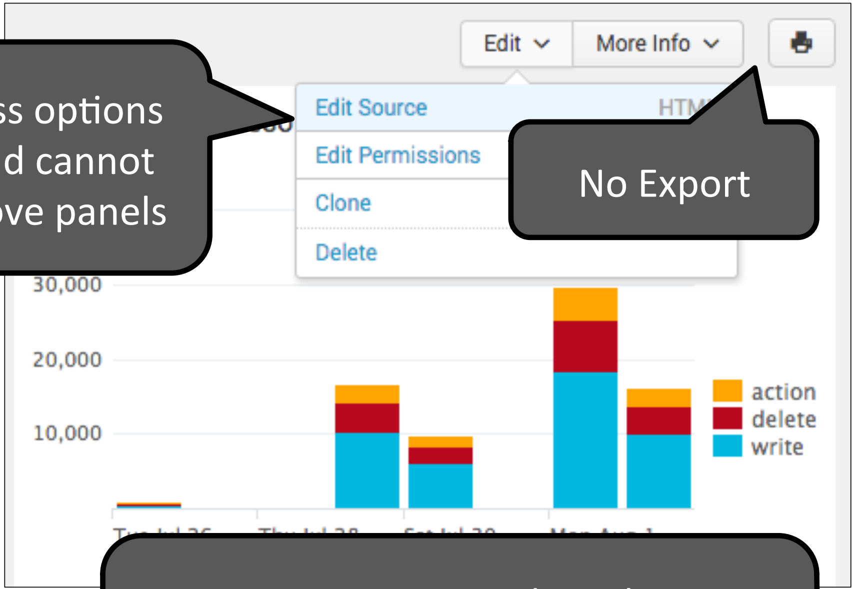


HTML

Use Simple XML as much as possible



Simple XML



Less options and cannot move panels

No Export

But, it is HTML and you have complete control of the look and feel.

Get to know JavaScript and jQuery

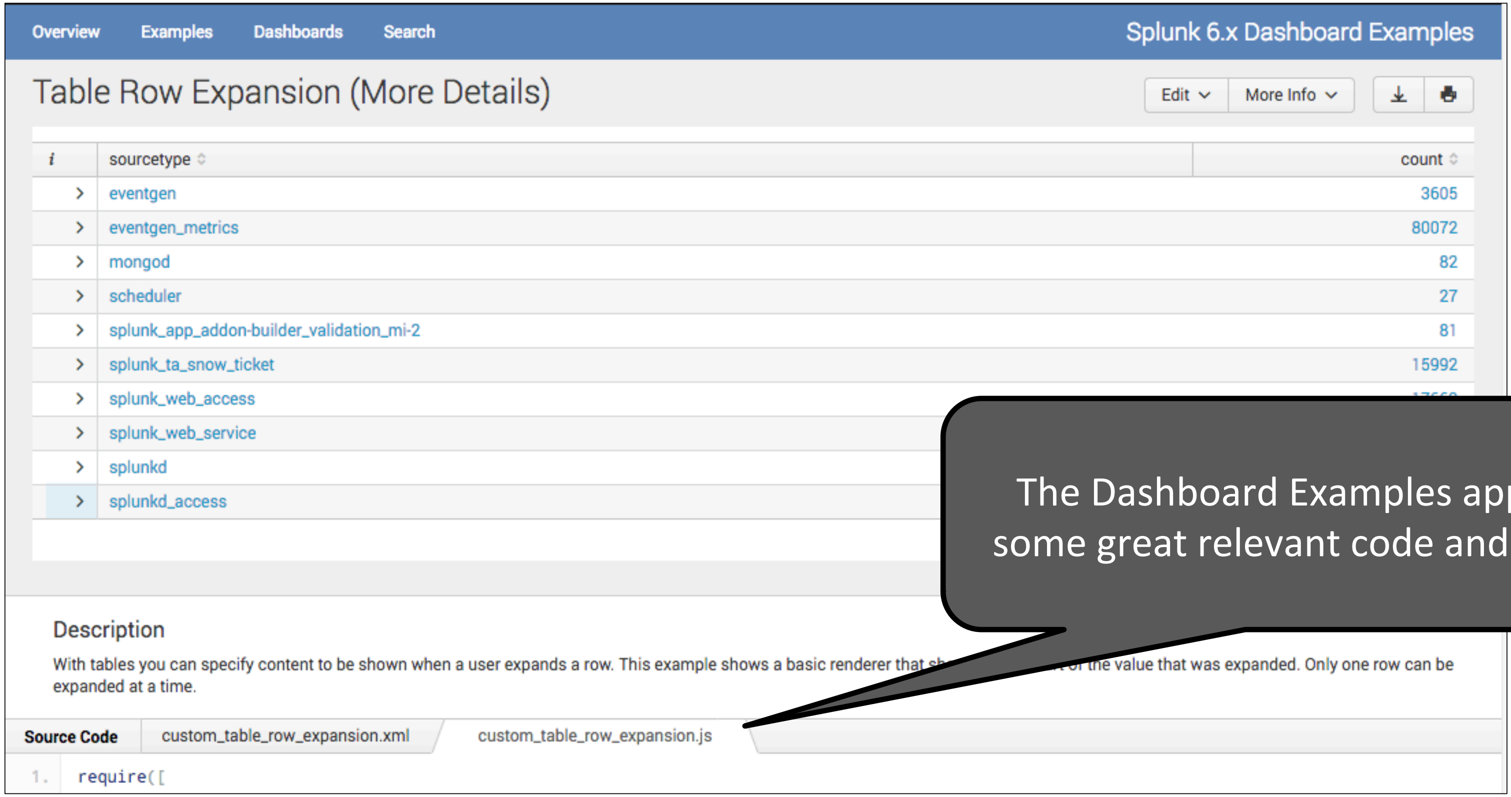


Table Row Expansion (More Details)

i	sourcetype	count
›	eventgen	3605
›	eventgen_metrics	80072
›	mongod	82
›	scheduler	27
›	splunk_app_addon-builder_validation_mi-2	81
›	splunk_ta_snow_ticket	15992
›	splunk_web_access	[illegible]
›	splunk_web_service	
›	splunkd	
›	splunkd_access	

Description

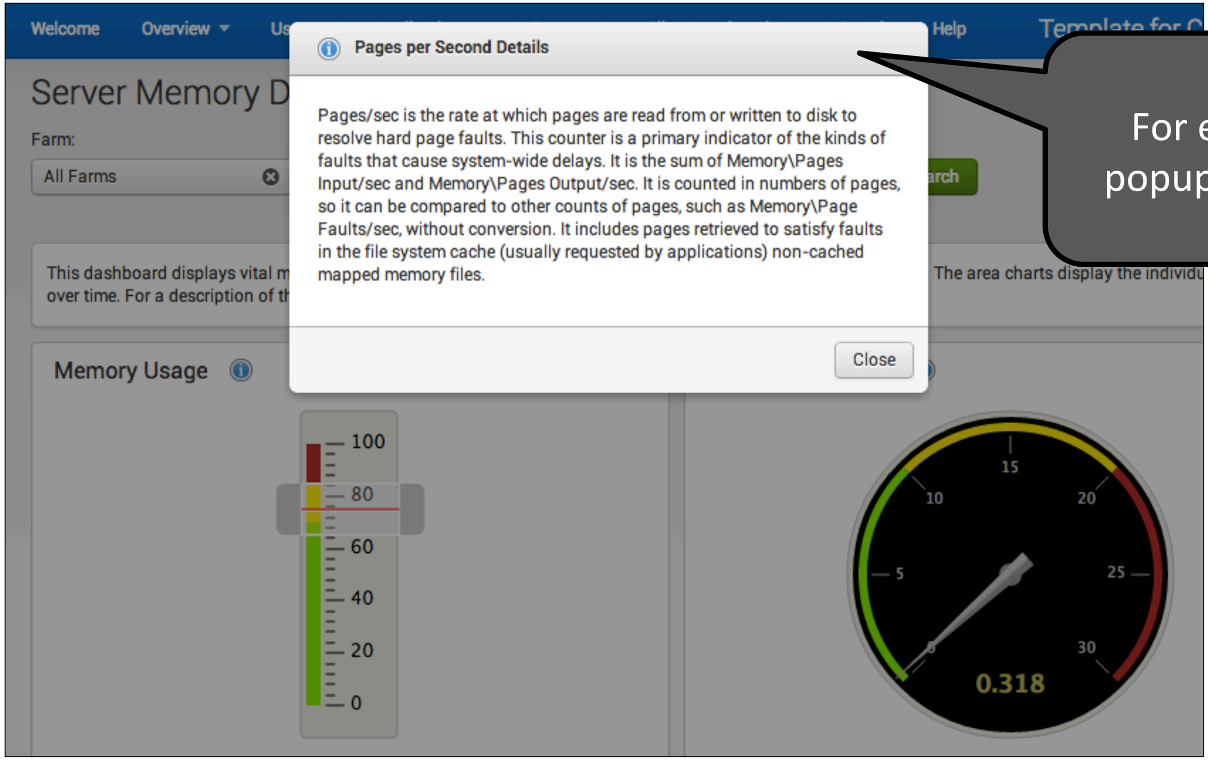
With tables you can specify content to be shown when a user expands a row. This example shows a basic renderer that sh [illegible] t of the value that was expanded. Only one row can be expanded at a time.

Source Code | custom_table_row_expansion.xml | custom_table_row_expansion.js

```
require([
```

The Dashboard Examples app has some great relevant code and ideas.

Bootstrap can add functionality



Pages per Second Details

Pages/sec is the rate at which pages are read from or written to disk to resolve hard page faults. This counter is a primary indicator of the kinds of faults that cause system-wide delays. It is the sum of Memory\Pages Input/sec and Memory\Pages Output/sec. It is counted in numbers of pages, so it can be compared to other counts of pages, such as Memory\Page Faults/sec, without conversion. It includes pages retrieved to satisfy faults in the file system cache (usually requested by applications) non-cached mapped memory files.

Close

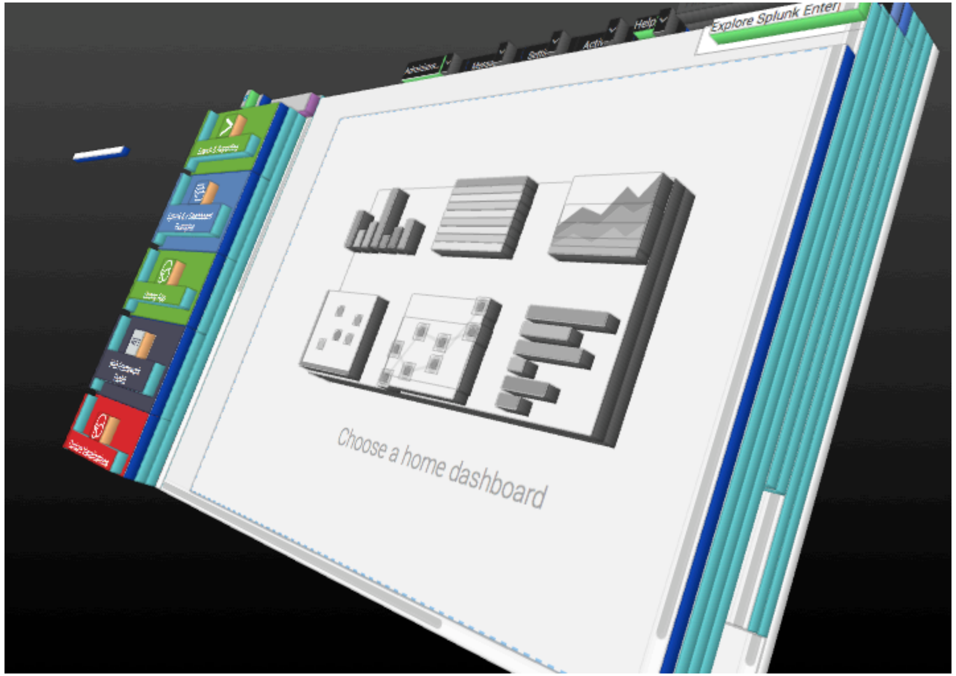
This dashboard displays vital m
over time. For a description of th

The area charts display the individu

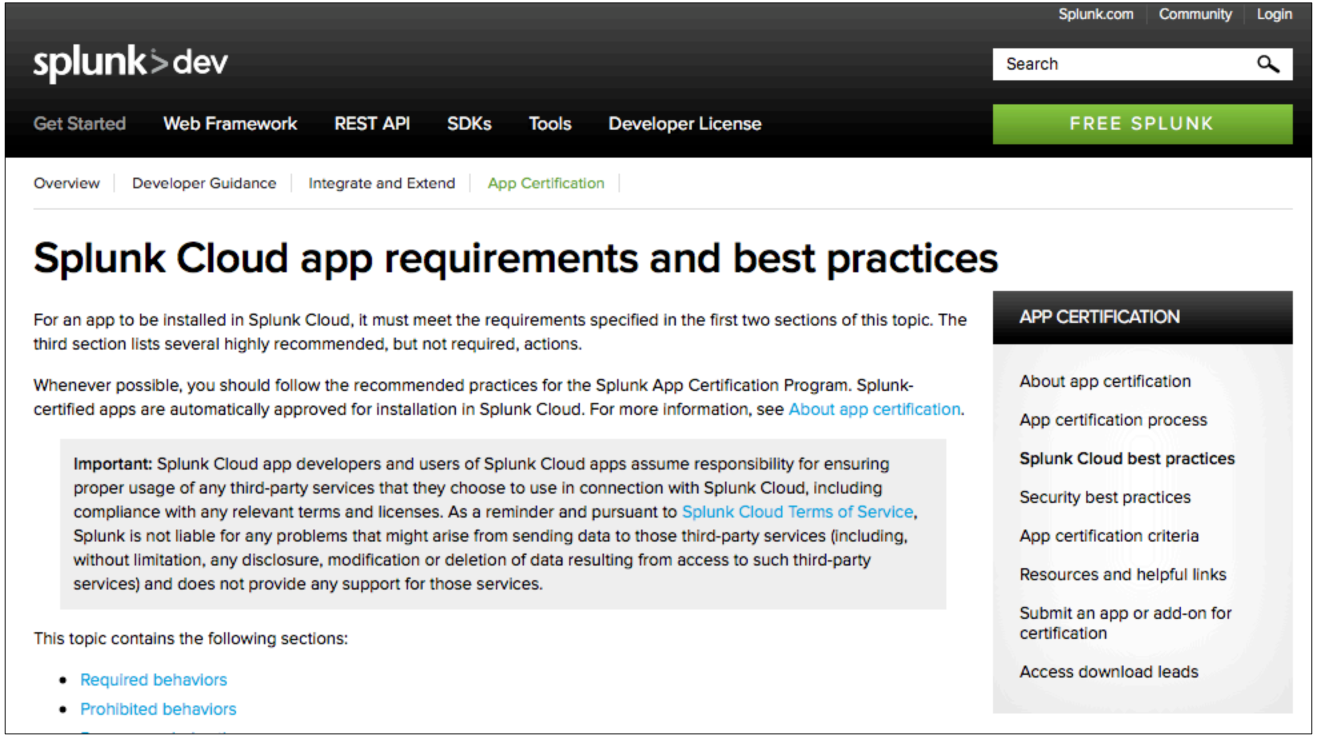
For example, easily add a modal popup to a Simple XML dashboard!

Get to know CSS

- All Splunk elements have an id now
- Check out Firefox's 3D view for layering



Splunk Cloud has Best Practices too



Splunk Cloud app requirements and best practices

For an app to be installed in Splunk Cloud, it must meet the requirements specified in the first two sections of this topic. The third section lists several highly recommended, but not required, actions.

Whenever possible, you should follow the recommended practices for the Splunk App Certification Program. Splunk-certified apps are automatically approved for installation in Splunk Cloud. For more information, see About app certification.

Important: Splunk Cloud app developers and users of Splunk Cloud apps assume responsibility for ensuring proper usage of any third-party services that they choose to use in connection with Splunk Cloud, including compliance with any relevant terms and licenses. As a reminder and pursuant to Splunk Cloud Terms of Service, Splunk is not liable for any problems that might arise from sending data to those third-party services (including, without limitation, any disclosure, modification or deletion of data resulting from access to such third-party services) and does not provide any support for those services.

This topic contains the following sections:

- Required behaviors
- Prohibited behaviors

APP CERTIFICATION
About app certification
App certification process
Splunk Cloud best practices
Security best practices
App certification criteria
Resources and helpful links
Submit an app or add-on for certification
Access download leads

http://dev.splunk.com/view/app-cert/SP-CAAAE85

Do's and Don'ts – Packaging Applications

Do	Don't
Follow the guidelines found at http://docs.splunk.com/Documentation/Splunk/latest/AdvancedDev/PackageApp	Leave any hidden files in the app such as Mac's ._ files.
Include a screen shot of your application in the correct location.	
Let the user choose which inputs are enabled for their environment.	Enable all inputs by default if not necessary.
Use a build automation tool such as Apache Ant if necessary to ensure a clean build/ package.	Leave anything in: $SPLUNK_HOME/etc/apps/<app>/local directory $SPLUNK_HOME/etc/apps/<app>/metadata/local.meta
Ensure the appropriate settings are set in app.conf	
Document your app with a README.txt file	
Test your application on a clean system	

Do’s and Don’ts – Data Collection

Do	Don’t
Support multiple platforms.	Code for a single OS.
Use scripting language utilities such as os.path.join() and the special environment variable $SPLUNK_HOME to construct paths in scripts.	Hard code script paths.
Write data to the “main” index. This ensures that your data is searchable by default.	Hard code index names in searches if you must use a custom index.
Use key=value pairs in writing to log files (if you have control of the logging output).	Use name abbreviations.
Throttle how much data is collected at one time from an API.	Overwhelm a system by pulling exorbitant amounts of data at one time from an API.
Use logging and error trapping in scripts and inputs.	

Do's and Don'ts - Applications

Do	Don't
Use setup.xml or a mod input configs to allow the end user to configure the app	Make users manually enter information such as API credentials into configuration files.
Encrypt user input passwords. http://blogs.splunk.com/2011/03/15/storing-encrypted-credentials/	Store clear text passwords in .conf files.
Parameterize indexes so that they can be easily changed	Hard code indexes in your searches
Use the CIM add-on http://docs.splunk.com/Documentation/CIM/latest/User/Overview	
Place all .conf files in default **$SPLUNK_HOME/etc/apps/<your_app>/default**	Leave any content in $SPLUNK_HOME/etc/apps/<your_app>/local
Set default permissions in: **$SPLUNK_HOME/etc/apps/<your_app>/metadata/default.meta**	Have a local.meta file located in: $SPLUNK_HOME/etc/apps/<your_app>/metadata

THANK YOU

.conf2016

splunk>