Worst Practices for Building Splunk Apps and Add-ons

…and how to avoid them

Jason Conger | Splunk

October 2018 | Version 1.0

Forward-Looking Statements

During the course of this presentation, we may make forward-looking statements regarding future events or the expected performance of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results could differ materially. For important factors that may cause actual results to differ from those contained in our forward-looking statements, please review our filings with the SEC.

The forward-looking statements made in this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, this presentation may not contain current or accurate information. We do not assume any obligation to update any forward-looking statements we may make. In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only and shall not be incorporated into any contract or other commitment. Splunk undertakes no obligation either to develop the features or functionality described or to include any such feature or functionality in a future release.

splunk> .conf18

whoami

jason.conger@splunk.com
@JasonConger
http://www.linkedin.com/in/JasonConger
https://www.splunk.com/blog/author/jconger.html

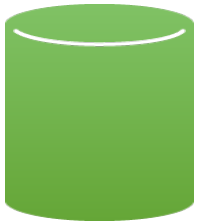
Staff Solutions Architect
Global Strategic Alliances

6+ years at Splunk

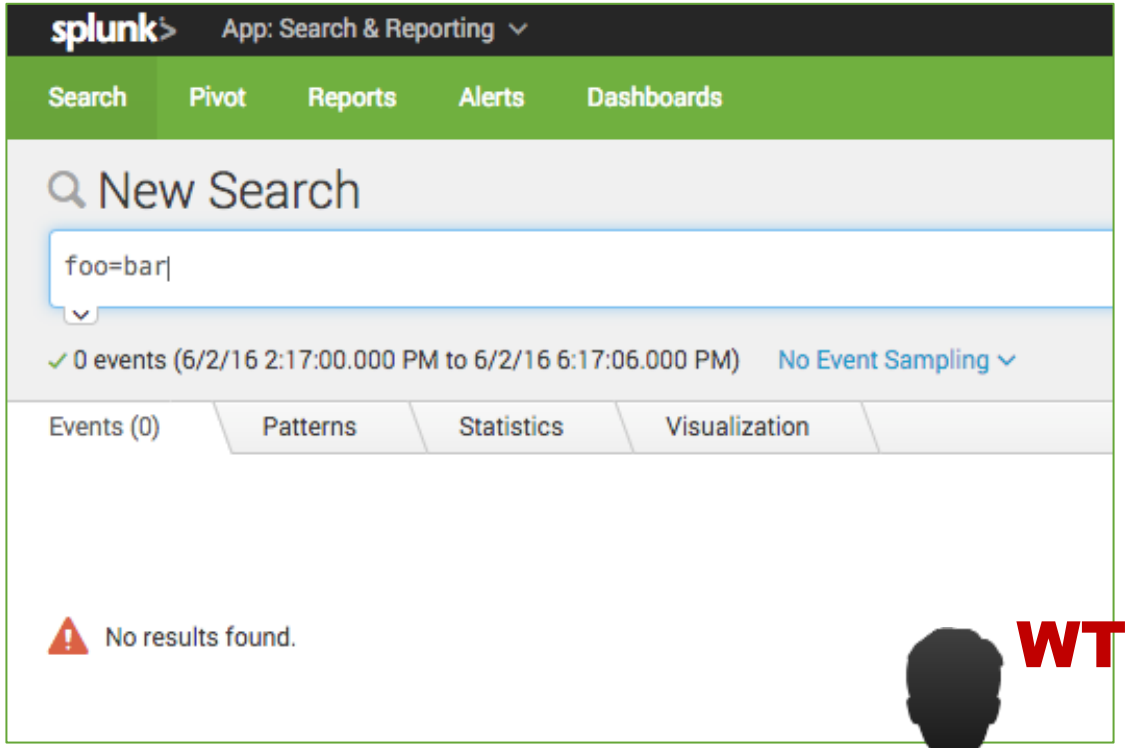
Created or consulted on 25+ Splunkbase applications

Worst Practice

Using Lots of Custom Indexes

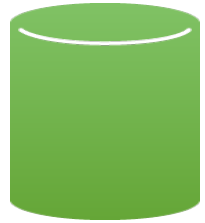


main



WTF?

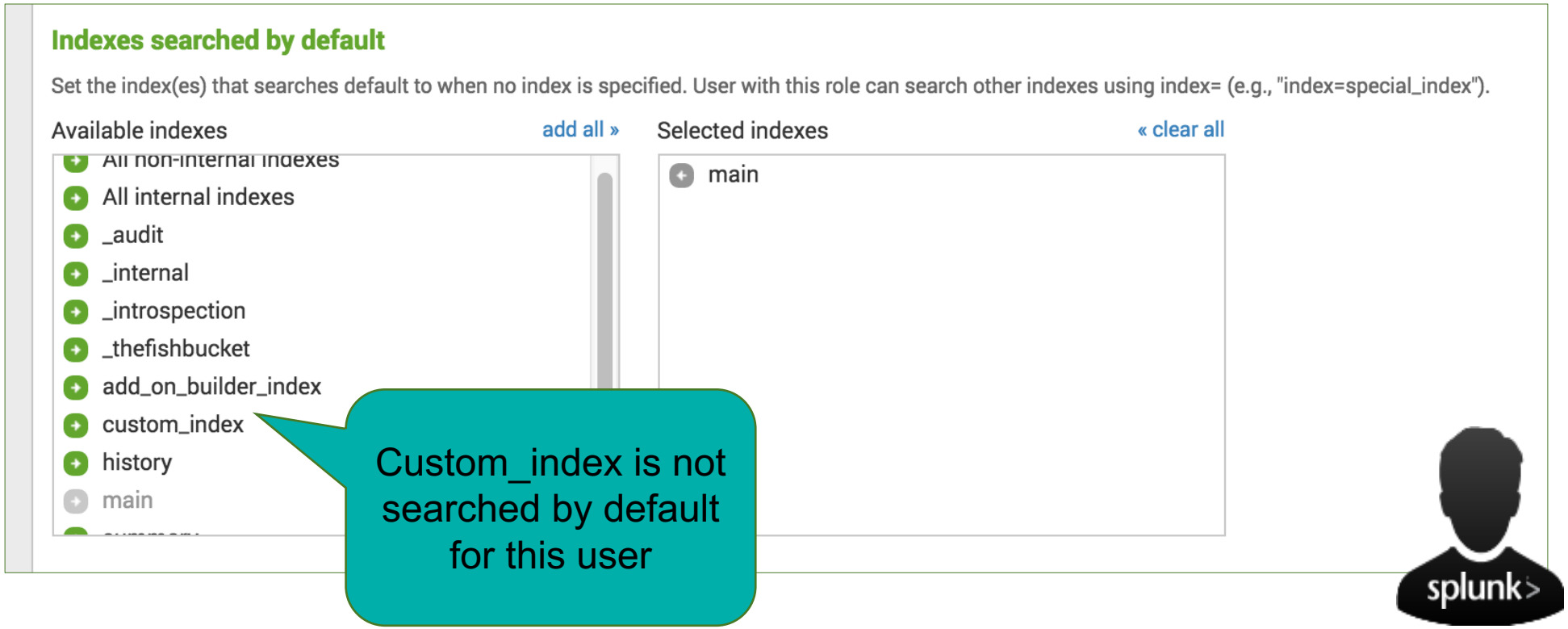
custom index



contains foo=bar data

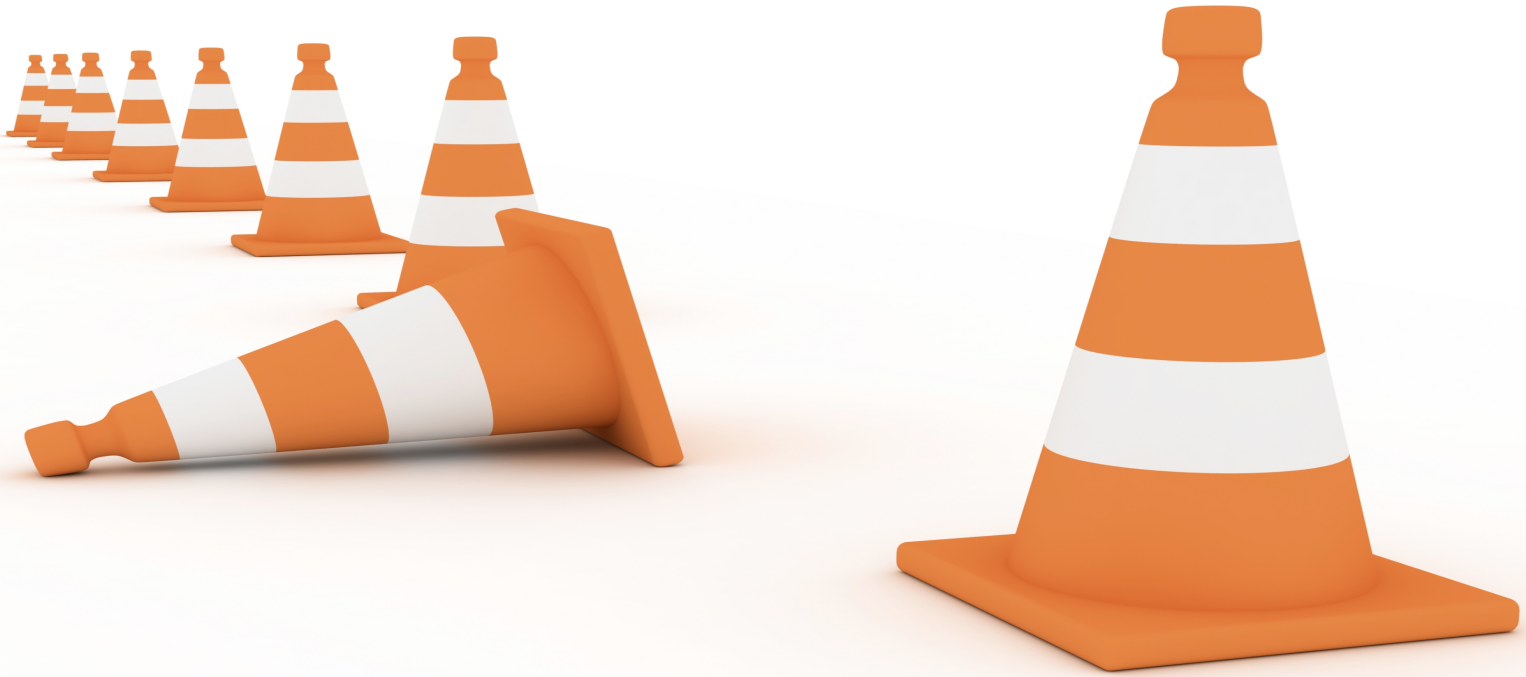
Worst Practice

Using Lots of Custom Indexes



Best Practice

Write Data To The “main” Index*



* there are exceptions to every rule

Exceptions To Using “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
- Performance – certain types and volumes of data may necessitate higher performing disk architectures

Exceptions to Using “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.
- Performance – certain types and volumes of data may necessitate higher performing disk architectures.

The last 3 exception decisions should be made by the Splunk admin – not the developer.

Worst Practice

Relying on Automatic sourcetype Recognition

inputs.conf

```
[monitor://<path>]
# living dangerously with no sourcetype
```

- If sourcetype is unset in inputs.conf, Splunk picks a source type based on various aspects of the data.
- Requires RegEx matching = additional processing
- “too_small” sourcetypes

Worst Practice

Neglecting Timestamp Extraction, Event Breaking, and Typing in props.conf

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>

splunk> .conf18

Best Practice

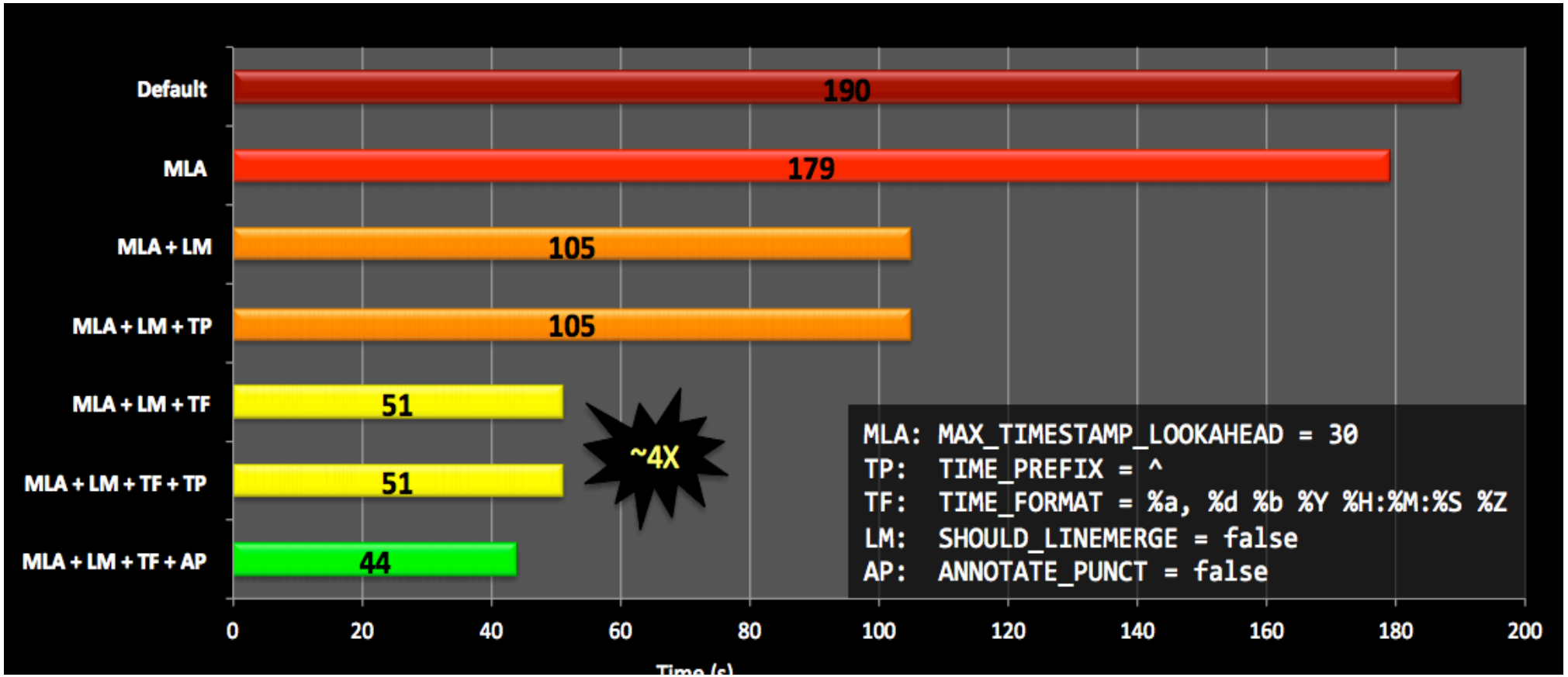
Explicitly Define sourcetype Settings

```
[mysourcetype]
TIME_PREFIX = ^
MAX_TIMESTAMP_LOOKAHEAD = 25
TIME_FORMAT = %Y-%m-%d %H:%M:%S.%3N %z
LINE_BREAKER = ([\r\n]+)\d{4}-\d{2}-
\d{2}\s\d{2}:\d{2}:\d{2}.\d{3}
SHOULD_LINEMERGE = False
ANNOTATE_PUNCT = false
```

splunk> .conf18

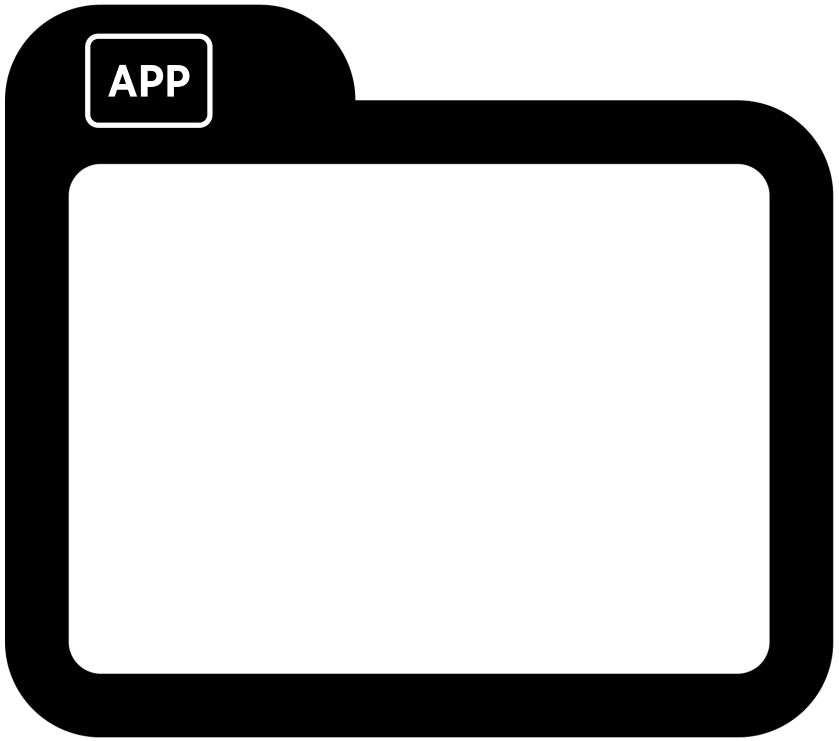
Best Practice

Explicitly Define sourcetype Settings



Worst Practice

Not Including Dependencies In Your App's bin Folder



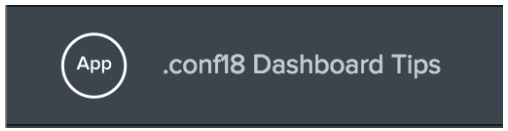
Best Practice

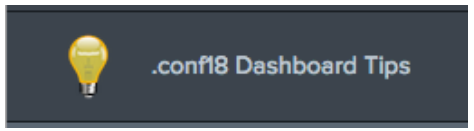
Include Everything Your App Needs

- Test on a clean Splunk install – Docker can be your friend.

Worst Practice

Not Including High-res 2x App Icons

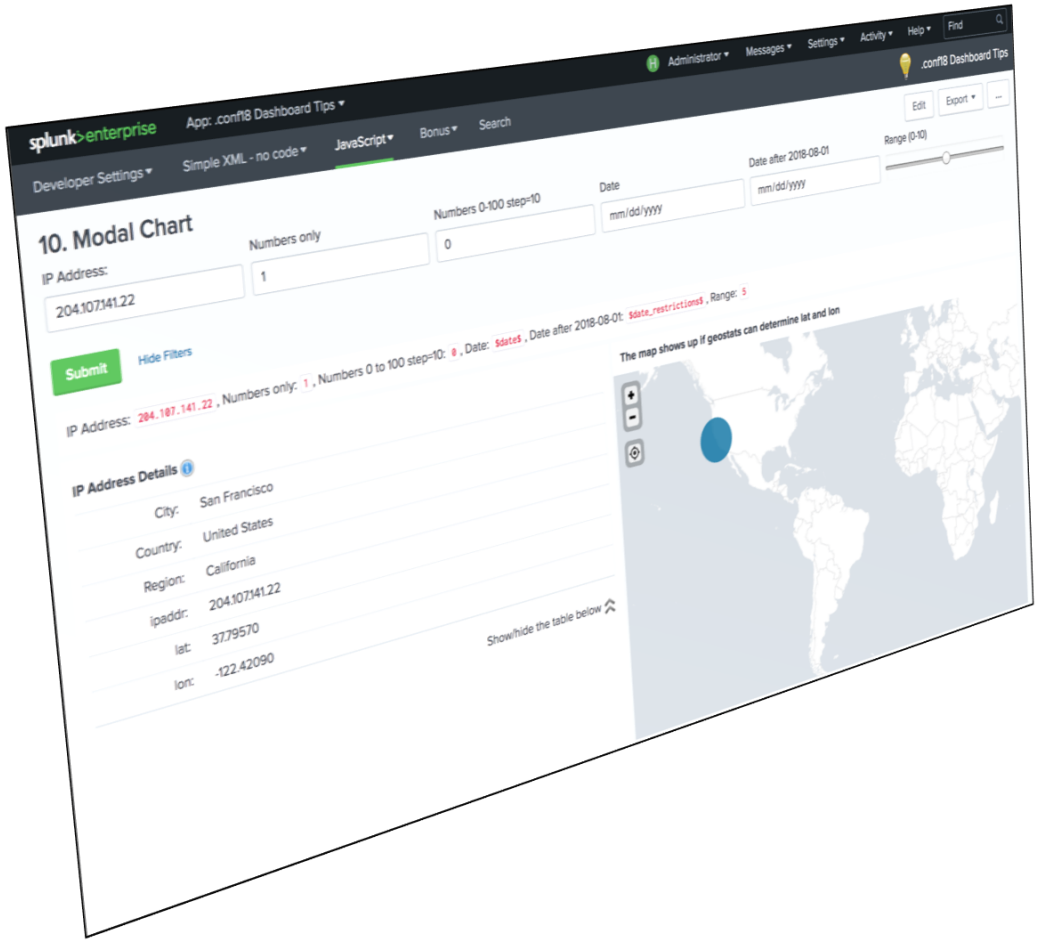




- But, the default icon is better than it used to be…

Worst Practice

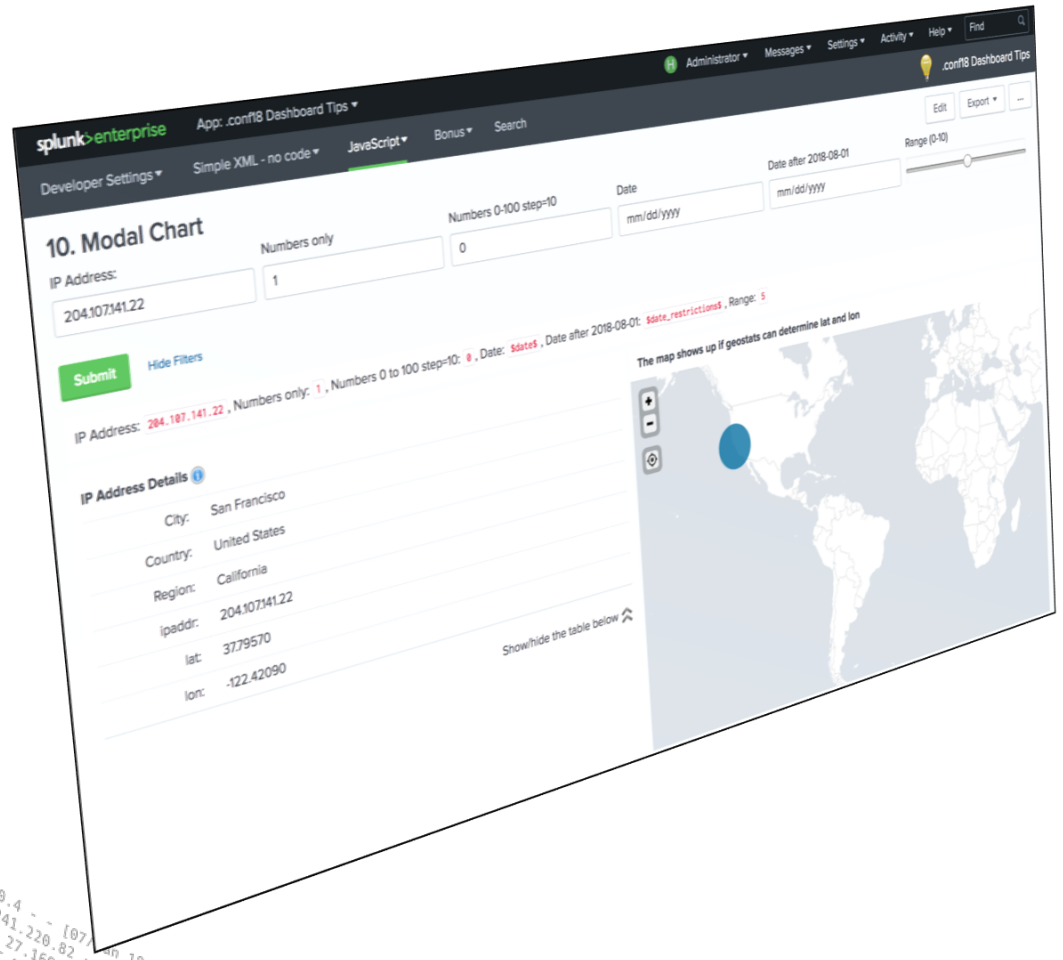
Not Parameterizing Base Searches




```
<search>
  <query>
    index = my_index sourcetype
= mysourcetype | stats count
  </query>
</search>
```

Best Practice

Use macros (preferred) or eventtypes to parametrize searches



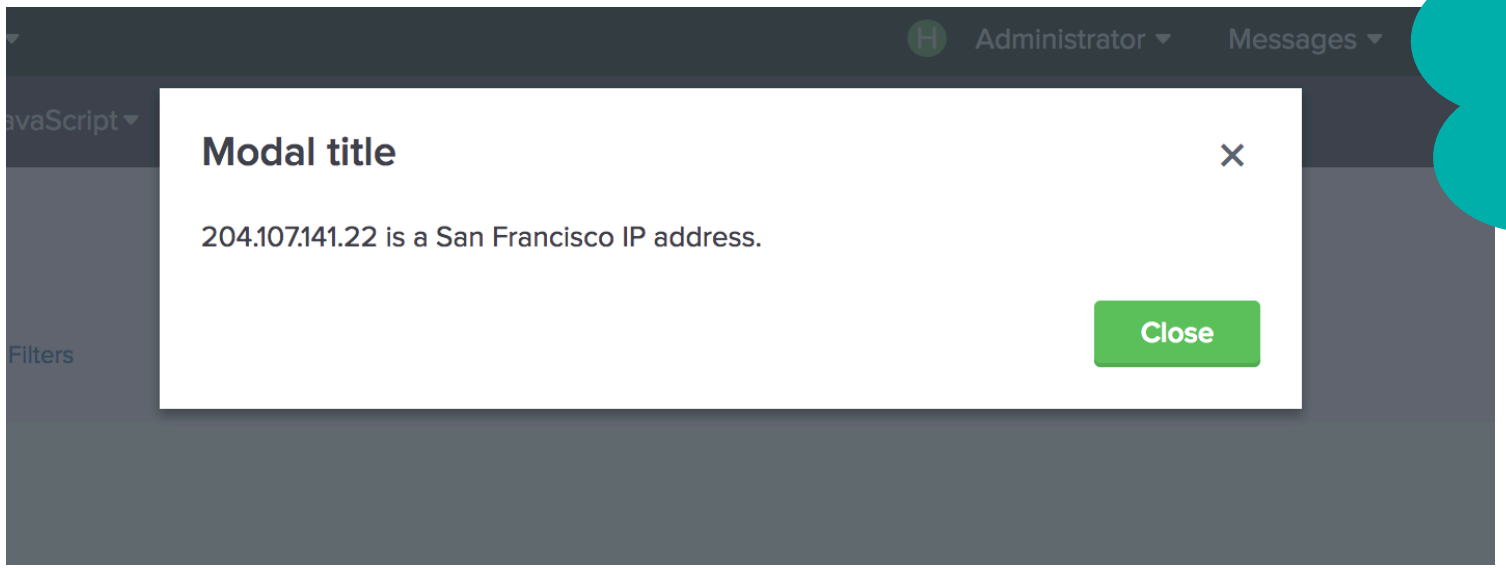

```
macros.conf
[my_index]
definition = index=my_index

[my_sourcetype]
definition = `my_index` sourcetype =
my_sourcetype
```

```
<search>
  <query>
    `my_sourcetype` | stats count
  </query>
</search>
```

Worst Practice

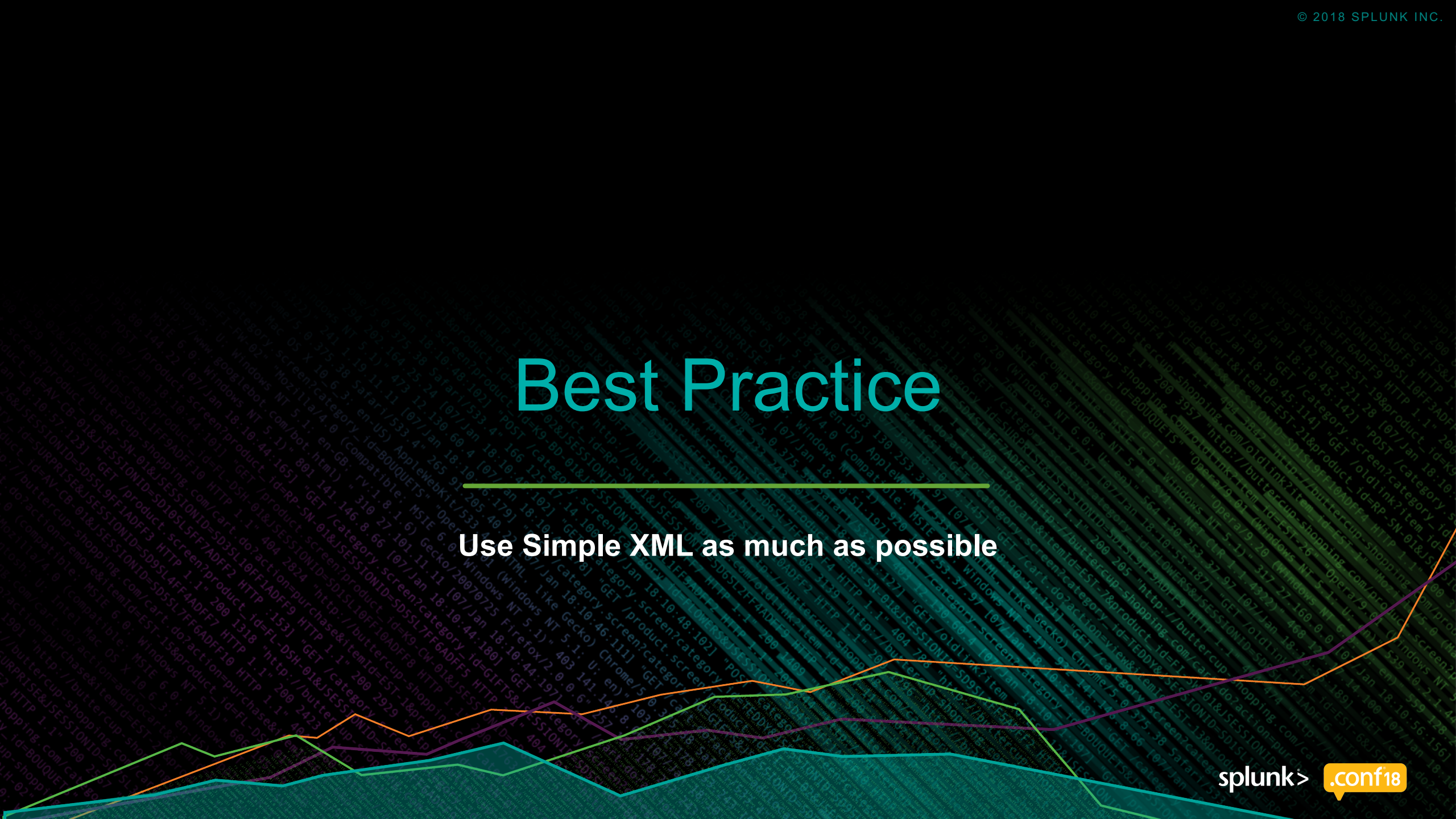
Converting to HTML for Basic Dashboard Extension



I need to add a modal popup, I guess I need to convert to HTML…

Best Practice

Use Simple XML as much as possible



More Splunkiness

- Meh practice – not using TERM() for IP Address searches
  - https://docs.splunk.com/Documentation/Splunk/latest/Search/UseCASEandTERMtomatchphrases
- Worst practice – no debug logging in add-ons
  - Use debug logging so the use of the _internal index can be used for troubleshooting
- Worst practice – not using AppInspect
  - Use AppInspect even if you are not uploading to Splunkbase – it doesn't hurt

Thank You

Don't forget to rate this session in the .conf18 mobile app