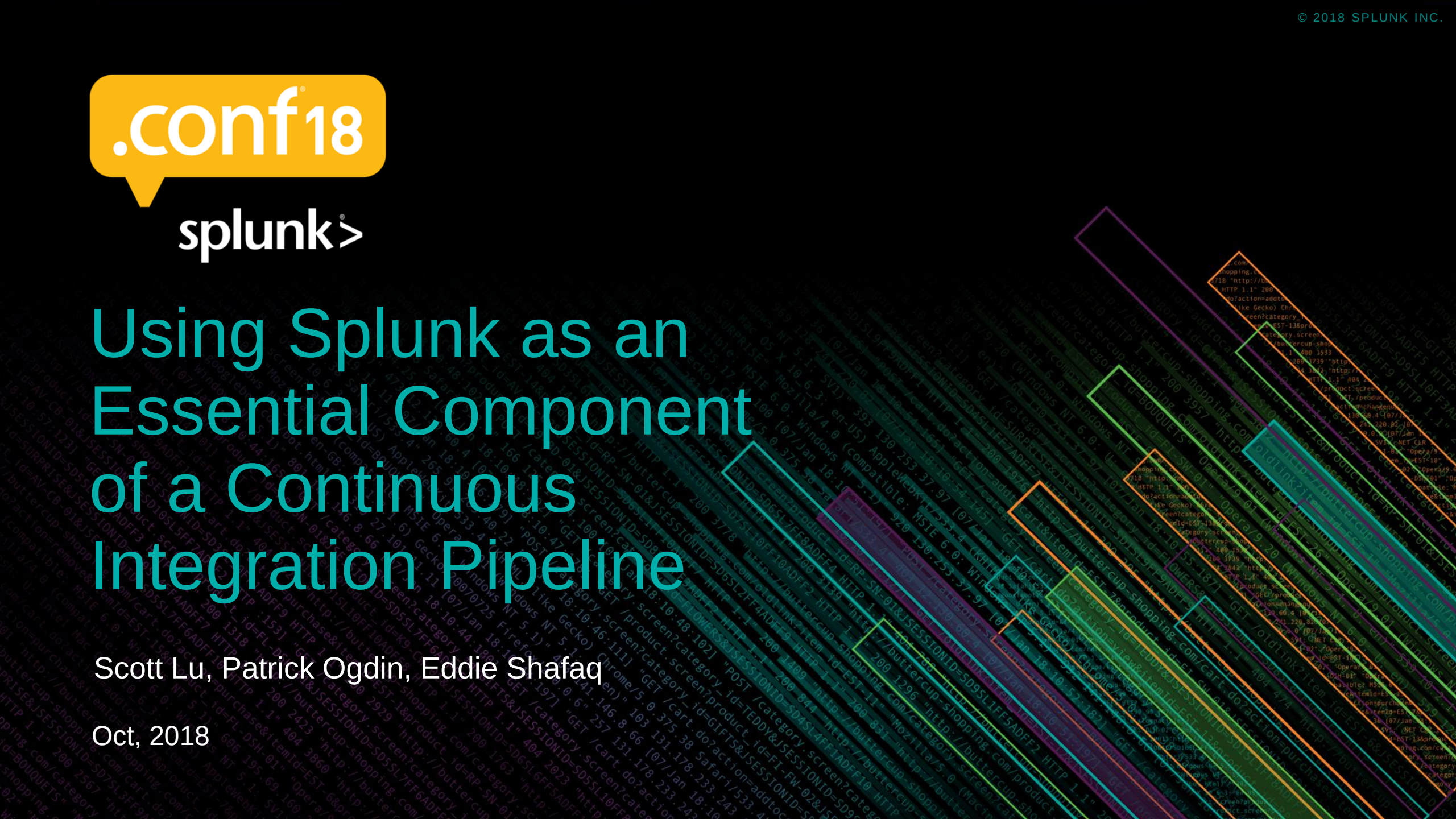
Forward-Looking Statements

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



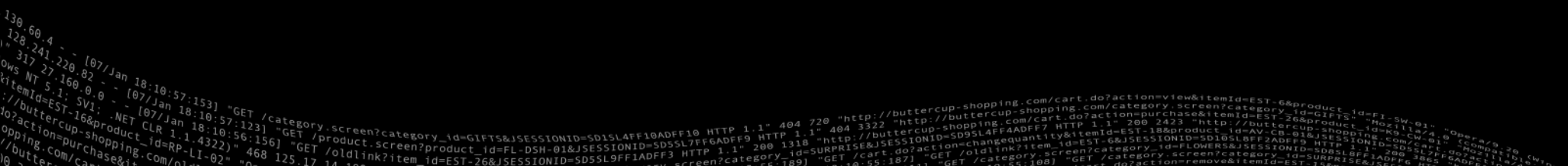
Using Splunk as an Essential Component of a Continuous Integration Pipeline

Scott Lu, Patrick Ogdin, Eddie Shafaq

Oct, 2018

Agenda

- Introduction (Scott & Patrick)
- Internal Customer CI/CD Splunk Scenarios
  - Pipeline Visualization
  - Test Result Triage System (ARTs)
  - Performance Testing Results
  - Developer Portal
- Introduction (Eddie)
- Infrastructure Engineering CI/CD Splunk Scenarios
  - Splunk/Jenkins
  - Distcc
  - Build Parallelization
  - Test Parallelization



Our Speakers

PATRICK OGDIN

Principal Product Manager

SCOTT LU

Principal Software Engineer

MODERATED BY GREEN TRACKSUIT

splunk> .conf18

Our Speakers

Patrick Ogdin

Principal Product Manager

Splunker since 2008. Working on Splunk Performance Testing and internal Reporting for our Continuous Integration Pipeline. Previously Product Manager for Splunk Enterprise.

Scott Lu

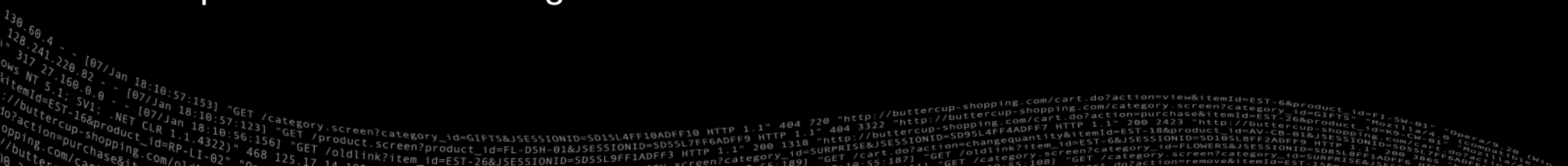
Principal Software Engineer

Joined Splunk since 2015. Mainly focusing on improving test framework and designing automation test cases for UI testing. Also developed an effective test result triage system using Splunk.

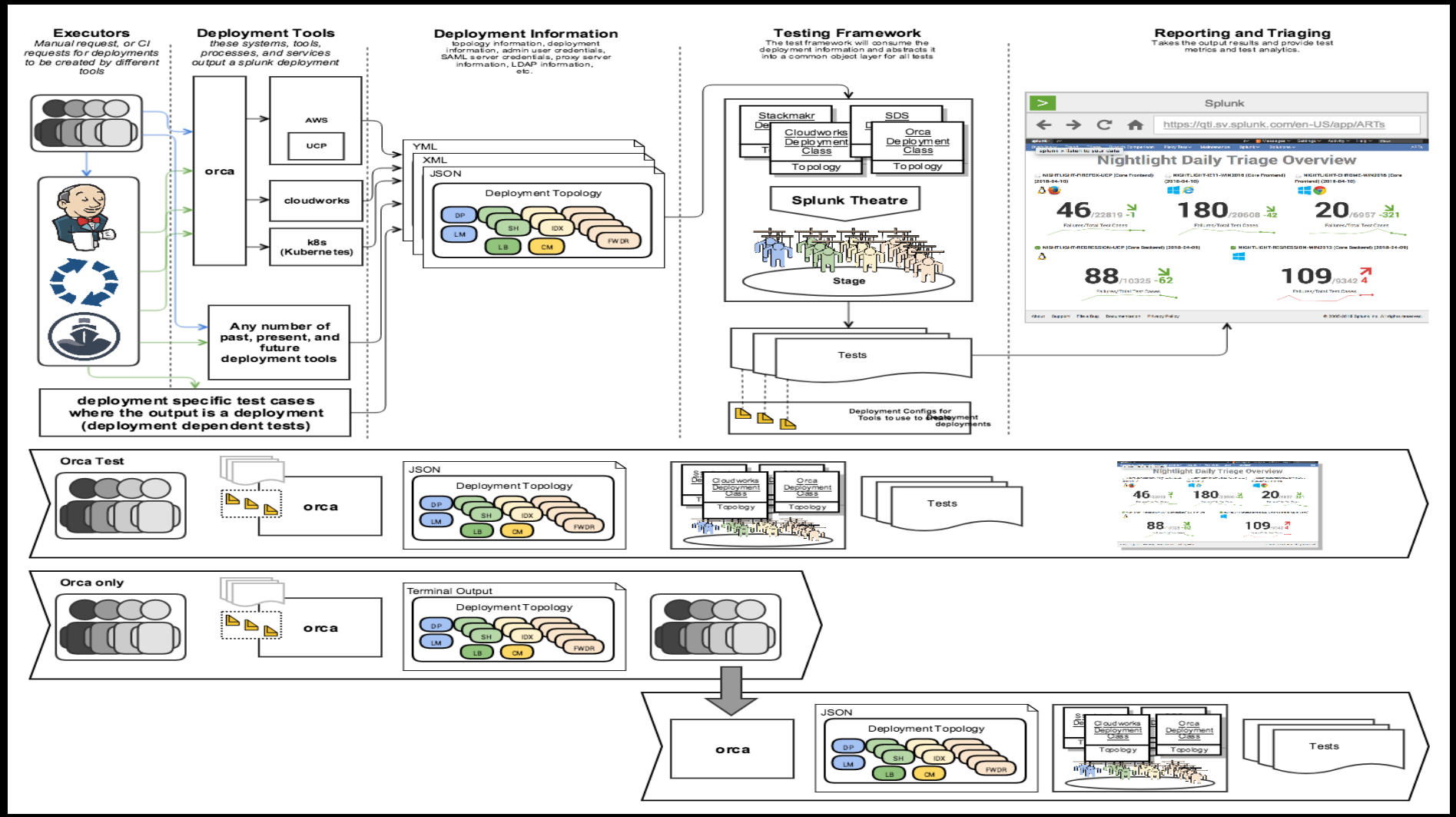
Introduction

Why Splunk for CI Use Cases

- Multiple critical systems working together
  - Some built and hosted internally
  - Others consumed as Cloud services
  - Diverse data sources
- Need Health & Performance Data (standard troubleshooting)
  - CPU/Mem/IO
- Need Application/Service Level Visibility
  - Build process
  - Test platform
  - Storage
- Engineering Productivity
  - Test results
  - Open stories/tasks/bugs



Problem: Building Software is Complicated



Internal Customer CI/CD Splunk Scenarios

splunk> .conf18

Splunk CI/CD Use Cases

Pipeline Visualization

Test Result Triage System (ARTs)

Performance

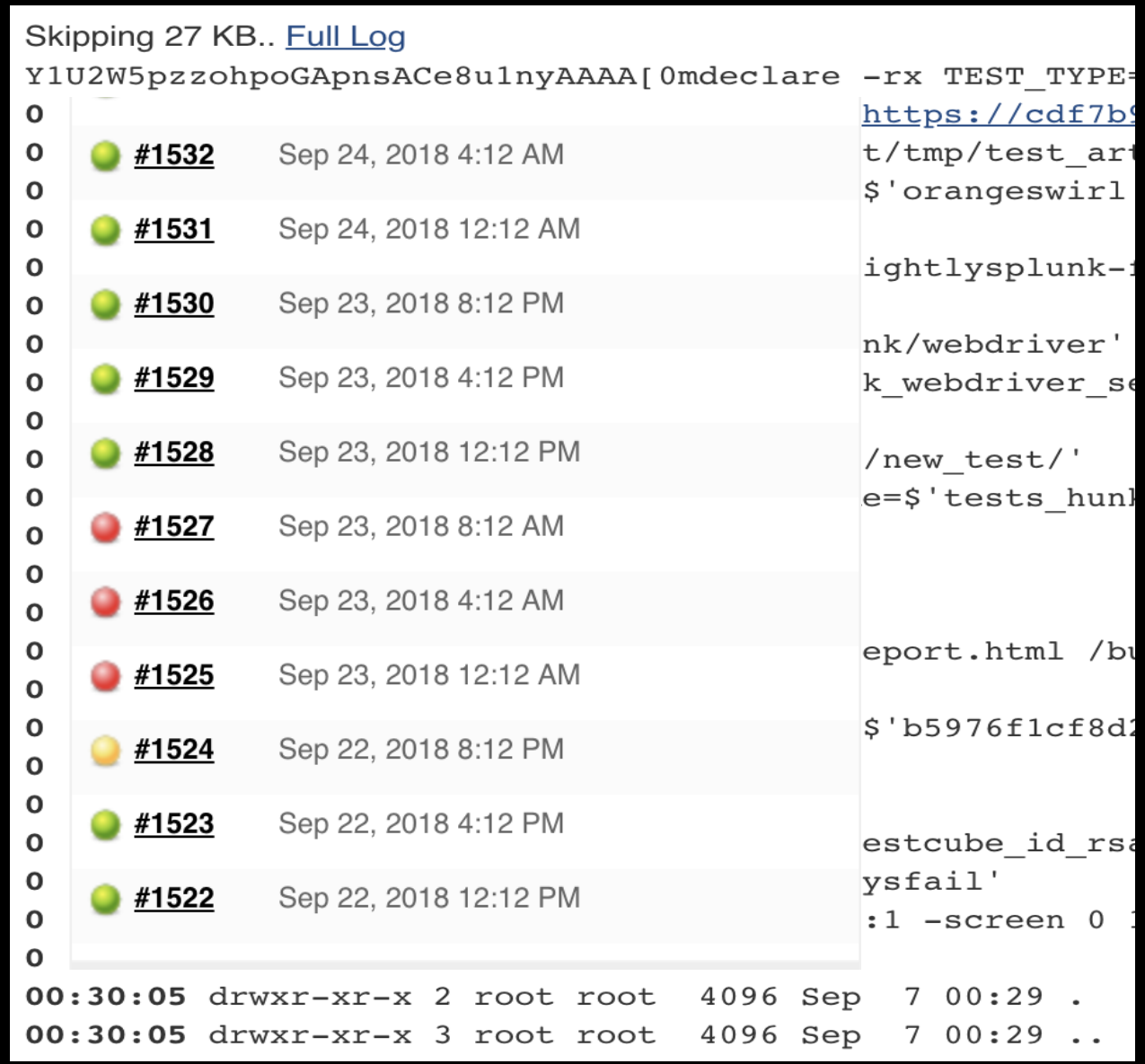
Developer Portal

Pipeline Visualization

Pipeline Visualization

Problem

- Hard to track jobs in Jenkins
- Lost in console logs
- How to take actions accordingly?



Pipeline Visualization

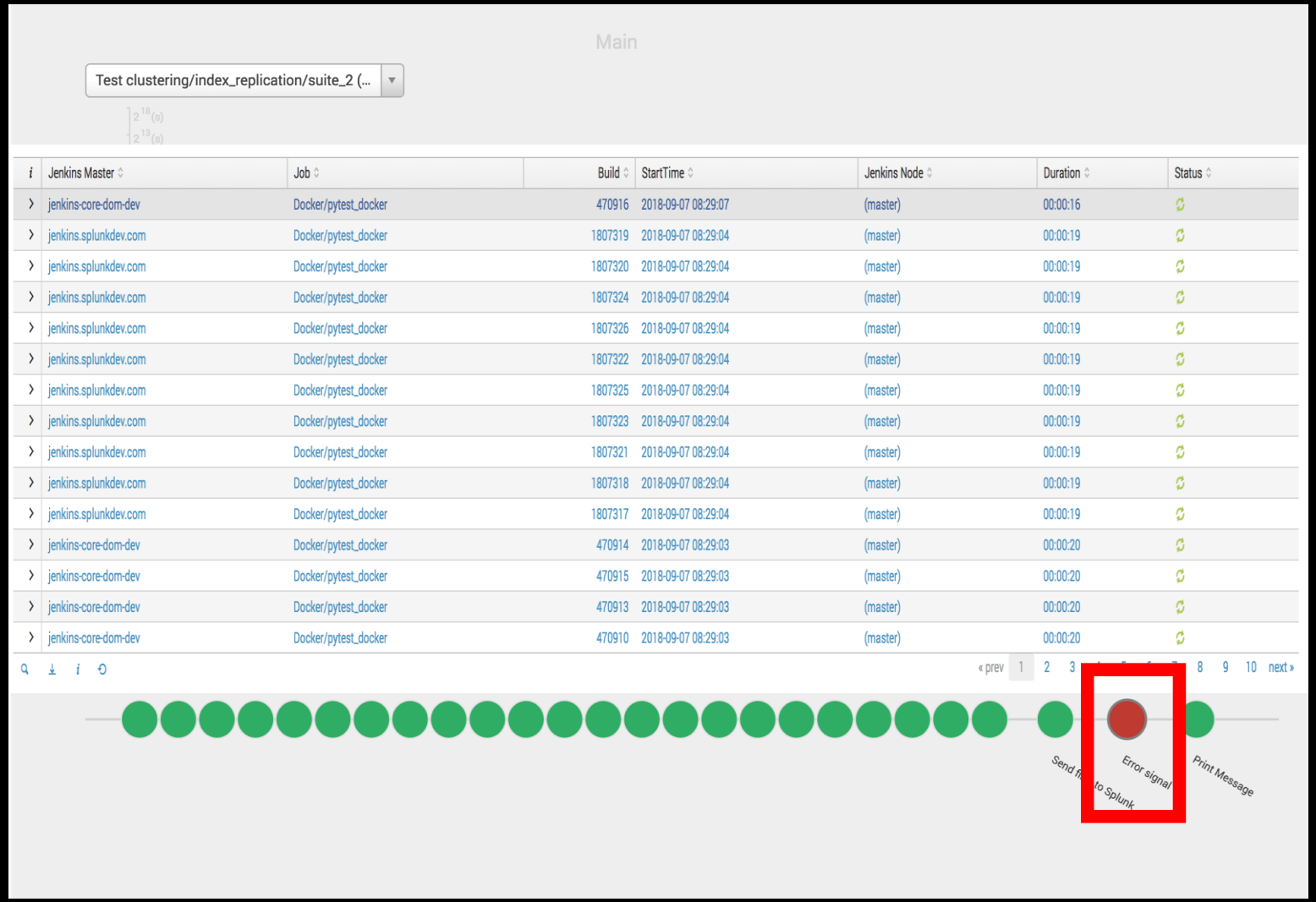
Solution

- Make it flat and easy to access
- Make it obvious and intuitive
- Make it informative and comprehensive

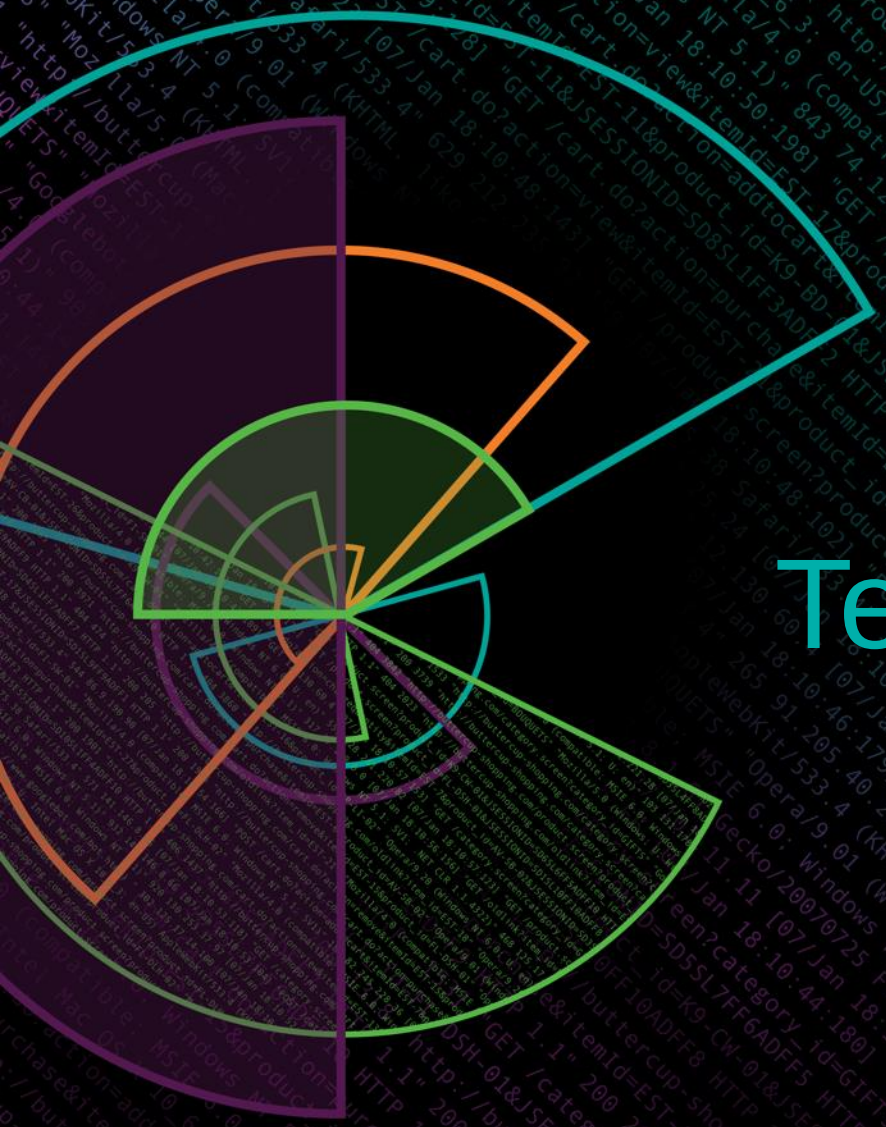
Pipeline Visualization

Highlights

- Mimic the nature of Jenkins workflow
- Easy to pinpoint the issue
- Take action accordingly



splunk> .conf18



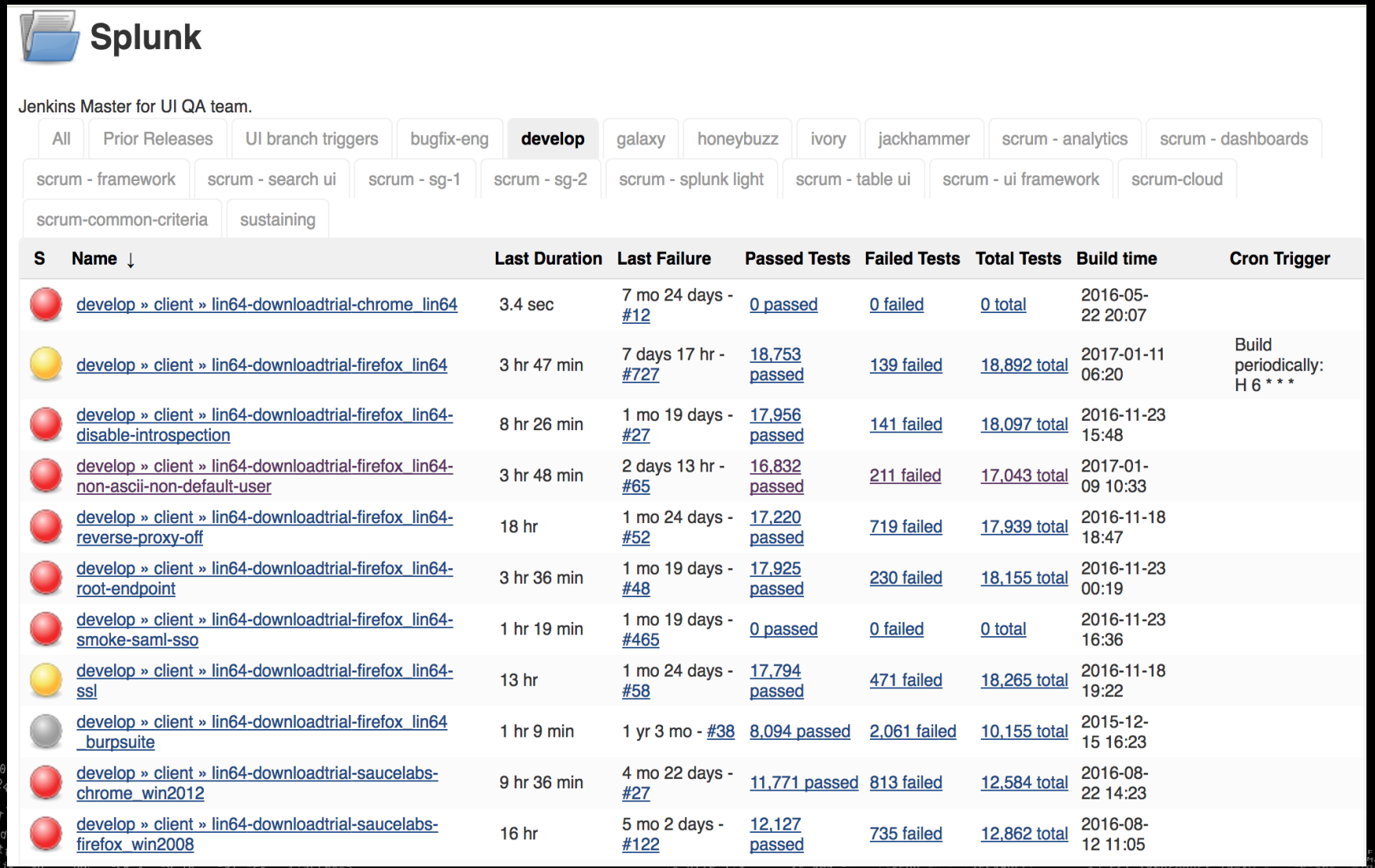
Test Result Triage System

ARTs

splunk> .conf18

Test Triage System

Problem



S	Name ↓	Last Duration	Last Failure	Passed Tests	Failed Tests	Total Tests	Build time	Cron Trigger
	develop » client » lin64-downloadtrial-chrome_lin64	3.4 sec	7 mo 24 days - #12	0 passed	0 failed	0 total	2016-05-22 20:07	
	develop » client » lin64-downloadtrial-firefox_lin64	3 hr 47 min	7 days 17 hr - #727	18,753 passed	139 failed	18,892 total	2017-01-11 06:20	Build periodically: H 6 * * *
	develop » client » lin64-downloadtrial-firefox_lin64-disable-introspection	8 hr 26 min	1 mo 19 days - #27	17,956 passed	141 failed	18,097 total	2016-11-23 15:48	
	develop » client » lin64-downloadtrial-firefox_lin64-non-ascii-non-default-user	3 hr 48 min	2 days 13 hr - #65	16,832 passed	211 failed	17,043 total	2017-01-09 10:33	
	develop » client » lin64-downloadtrial-firefox_lin64-reverse-proxy-off	18 hr	1 mo 24 days - #52	17,220 passed	719 failed	17,939 total	2016-11-18 18:47	
	develop » client » lin64-downloadtrial-firefox_lin64-root-endpoint	3 hr 36 min	1 mo 19 days - #48	17,925 passed	230 failed	18,155 total	2016-11-23 00:19	
	develop » client » lin64-downloadtrial-firefox_lin64-smoke-saml-sso	1 hr 19 min	1 mo 19 days - #465	0 passed	0 failed	0 total	2016-11-23 16:36	
	develop » client » lin64-downloadtrial-firefox_lin64-ssl	13 hr	1 mo 24 days - #58	17,794 passed	471 failed	18,265 total	2016-11-18 19:22	
	develop » client » lin64-downloadtrial-firefox_lin64_burpsuite	1 hr 9 min	1 yr 3 mo - #38	8,094 passed	2,061 failed	10,155 total	2015-12-15 16:23	
	develop » client » lin64-downloadtrial-saucelabs-chrome_win2012	9 hr 36 min	4 mo 22 days - #27	11,771 passed	813 failed	12,584 total	2016-08-22 14:23	
	develop » client » lin64-downloadtrial-saucelabs-firefox_win2008	16 hr	5 mo 2 days - #122	12,127 passed	735 failed	12,862 total	2016-08-12 11:05	

- Too many jobs to look at
- Lack of interaction
- Hard to collaborate among teams

splunk> .conf18

Test Triage System

Old Approach

Ember Test Cases Triage

Created by Scott Lu, last modified by Jian Zhang on Sep 16, 2015

This page is used to keep track of the failures and status of the test cases in **epic/client-qa-regression-tests** branch for each features.

Based on the results of **Build 6.3.0 GA RC3** (a52c23427a13) (total of **124** failures, 38 failures caused by known bugs)

Job Name ↓	All	Failed (Aug.3rd)	Failed (Aug.11th)	Failed with JIRA	Status	Owner	Comments
client_clustering_webdriver_master	80	8	7				
client_clustering_webdriver_searchhead	21	1	1			Jian	~~Fixed two timing test script issues: pull-request-875 merged~~
client_clustering_webdriver_slave	30	3	3				
client_forwarder_mgmt_webdriver	175	10	10	4		Jian	~~pull-request-874 merged~~ 6 of the failures are due to pre-created serverclasses have 0 mapped (should be 1), cannot be reproduced locally ~~3 tests fixed, pull-request-911 merged~~ Passed locally: test_unsupported_different_restartsplunkd, test_unsupported_different_stateonclient,test_unsupported_no_unsu ~~SPL-104695~~ - "Learn more" links return an error page CLOSED SPL-103719 - There is no error message when repositoryLocatior is used in Apps UNTRIAGED

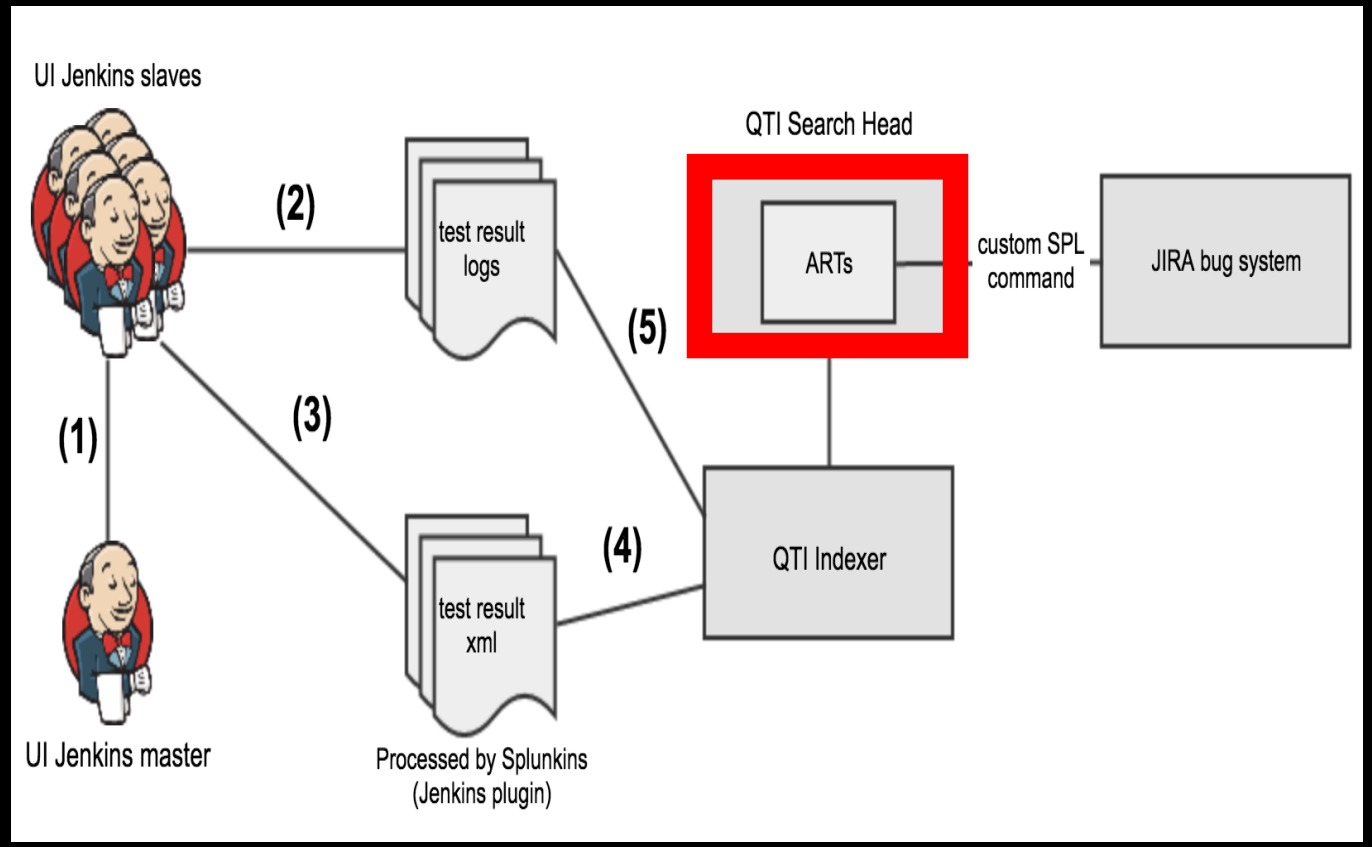
- Heavy manual work
- Stale information
- Not scalable

splunk> .conf18

Test Triage System

Solution

- ARTs (Automation Result Triage system)



ARTs Highlights

Overview

ORANGESWIRL-CHROME-UCP (2018-10-01)

76/236876

Failures/Total Test Cases

ORANGESWIRL-IE11-WIN2016 (2018-09-26)

75/21715

Failures/Total Test Cases

ORANGESWIRL-REGRESSION-UCP (Core Backend) (2018-09-26)

80/10244

Failures/Total Test Cases

ORANGESWIRL-REGRESSION-WIN2012 (Core Backend) (2018-09-26)

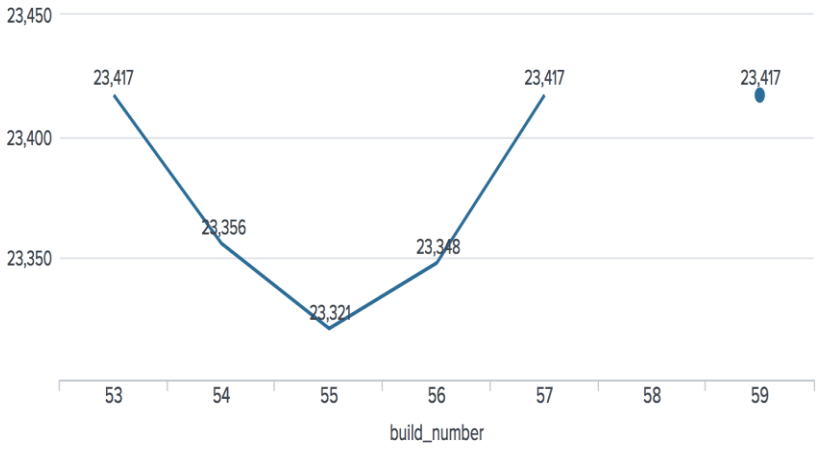
80/9224

Failures/Total Test Cases

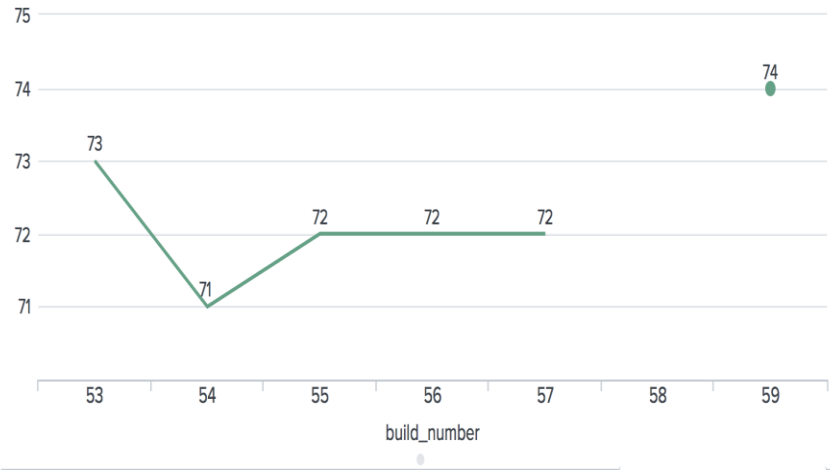
ARTs Highlights

Trend

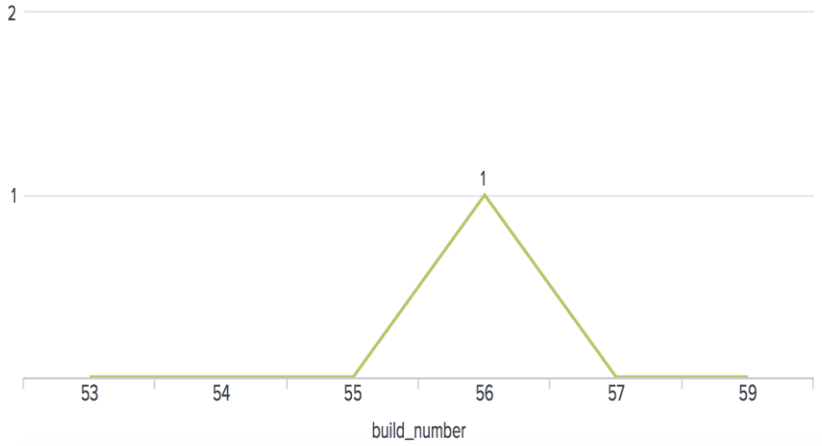
TOTAL TESTS TREND



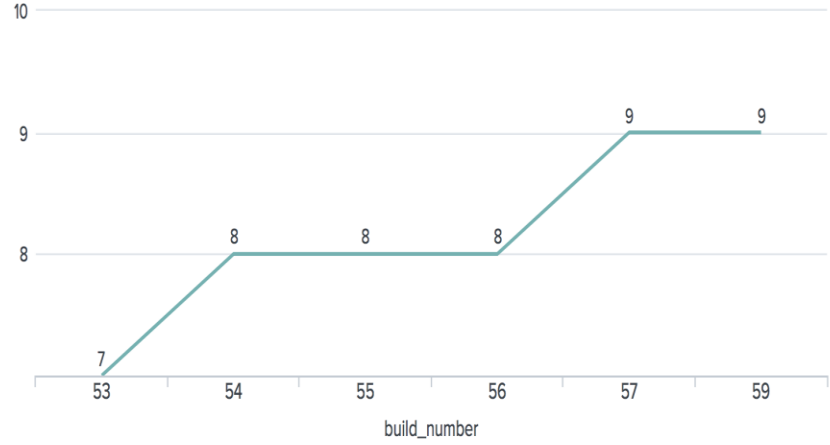
TOTAL FAILURES TREND



INFRASTRUCTURE (SETUP/START/RESTART/CONNECTION) FAILURES TREND



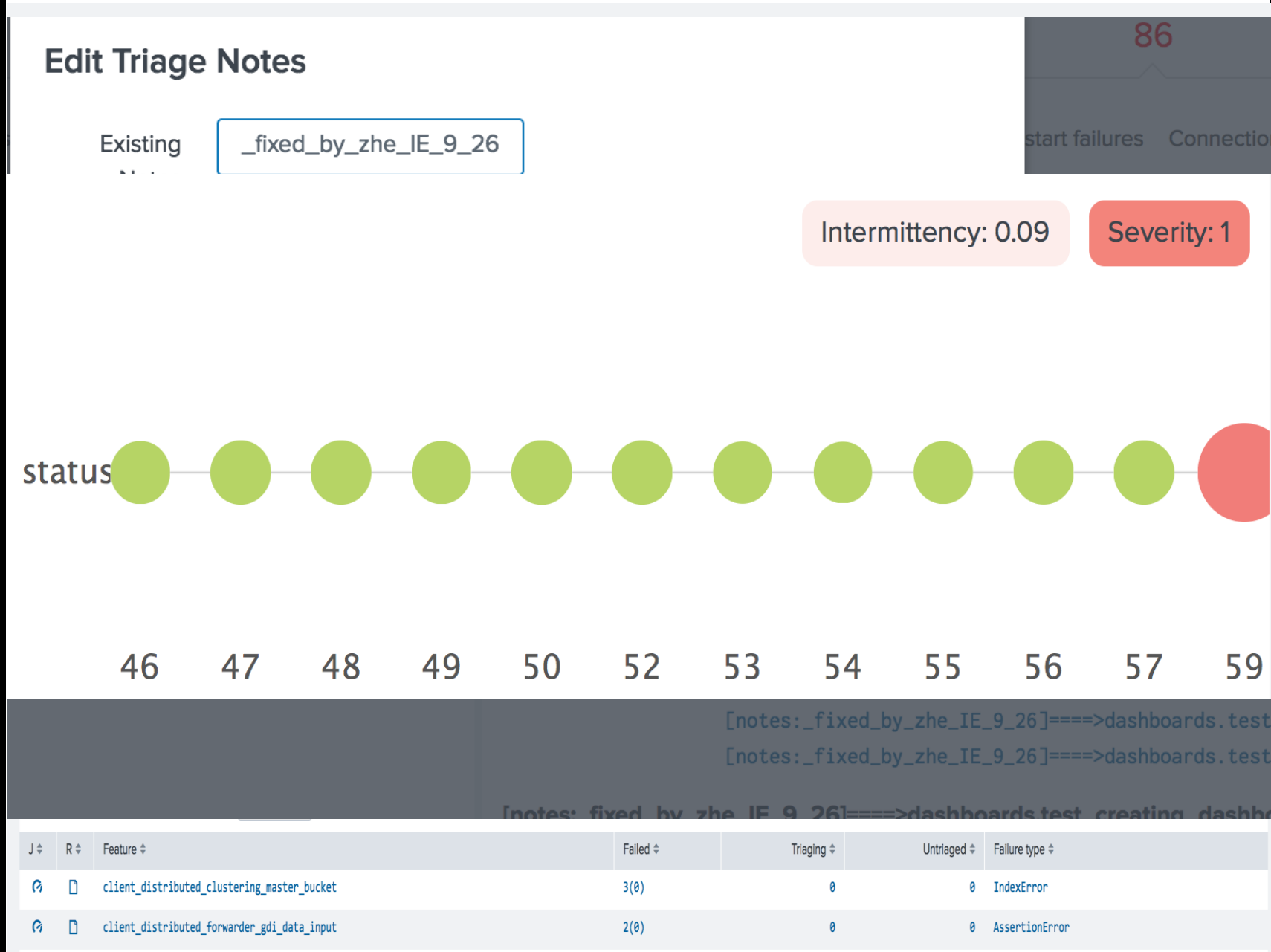
TEST FAILURES TREND



splunk> .conf18

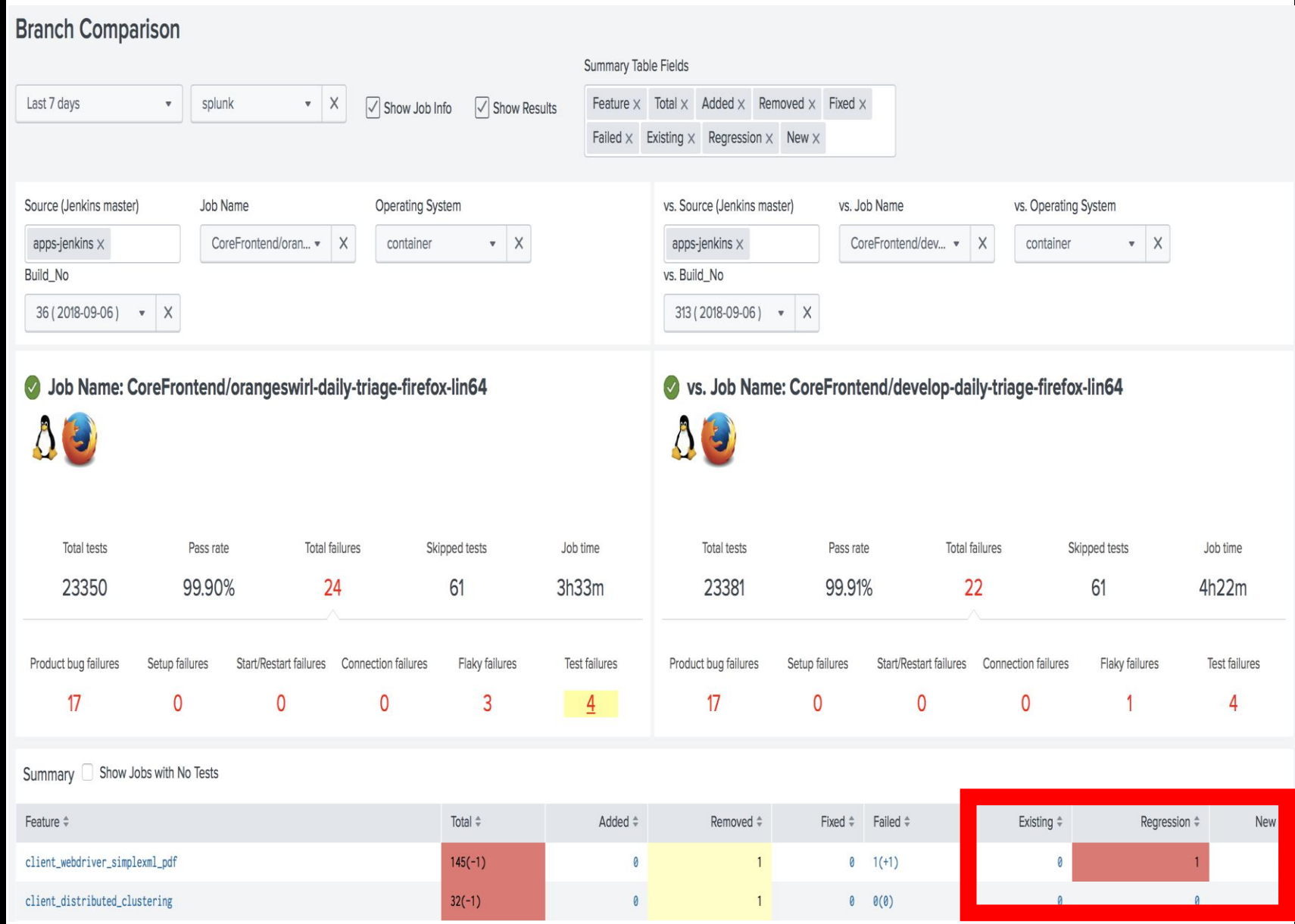
ARTs Highlights

Triage



ARTs Highlights

Branch Compare



splunk> .conf18

Performance

Use Case - Performance Testing Results

Problem

- Performance Testing yields lots of results across several Splunk products
  - Splunk Enterprise feature regression
    - Ex. Is Index Clustering replication getting slower?
  - Enterprise Security
    - System level benchmarks to find performance regressions across versions
  - Splunk Cloud
    - Pushing limits on Splunk features
- Proliferation of result formats
- Lots of manual work to compare and analyze test results
- Visualization and Trendspotting is a nightmare
- Variable Testing Schedules
  - Performance tests run on-demand
  - Performance tests run on a fixed schedule

Performance Testing Results

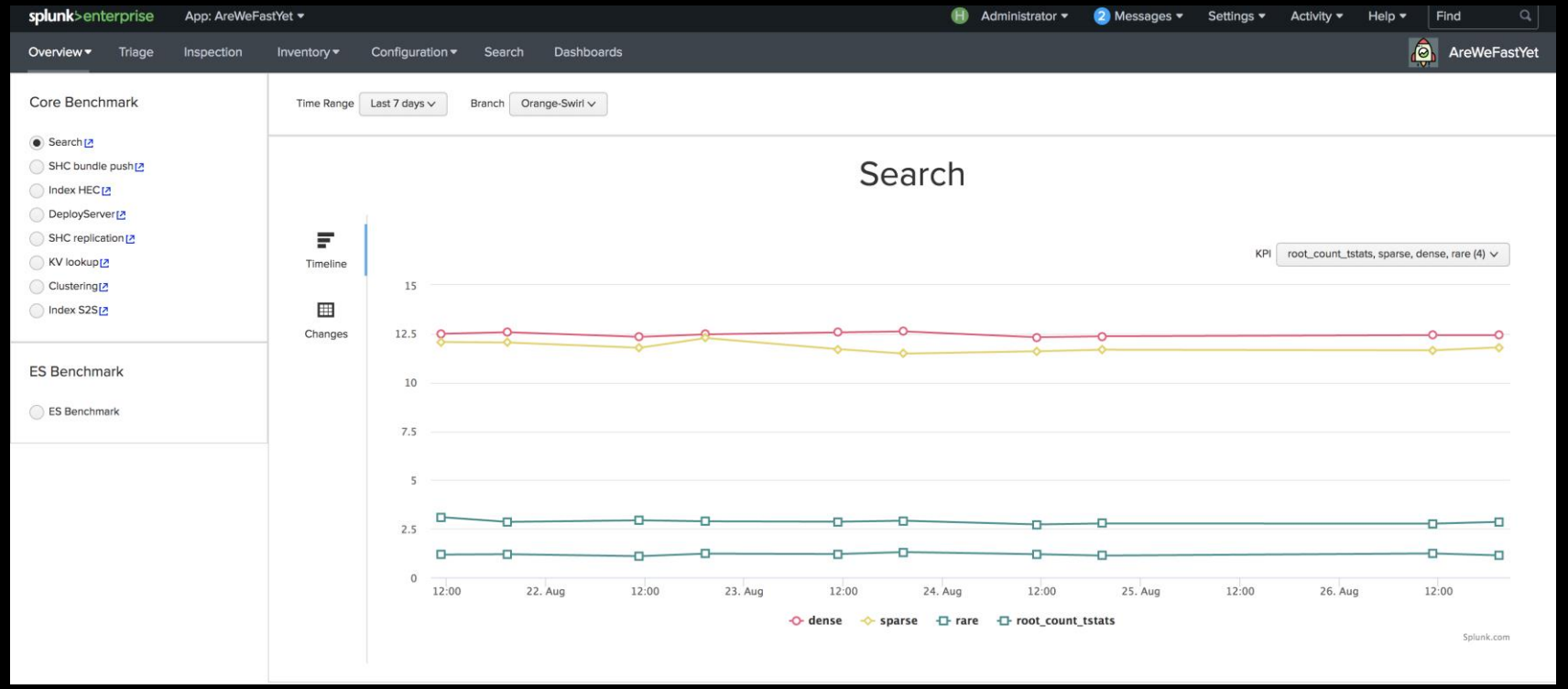
Solution

- AreWeFastYet 
  - One-stop-shop for scheduled and on-demand testing results
  - Interface for user defined rules to set KPIs and perform regression checks
  - Anomaly detection and alerting
  - Automatically generated visualization for test results and trending
  - Enable easy triage and profiling (future work)

Performance Testing Results

Highlights

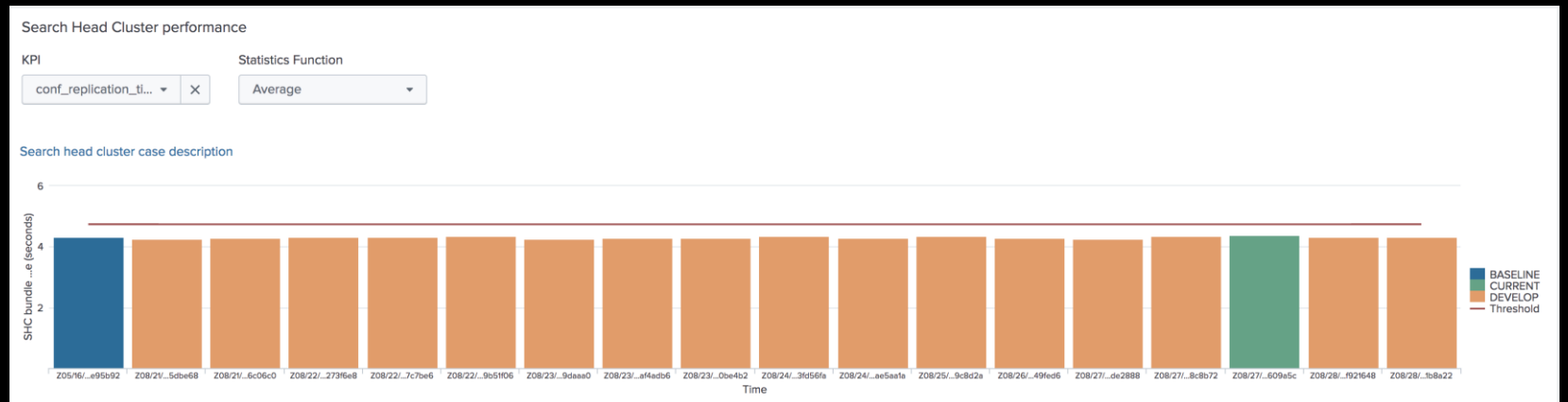
- Daily Benchmark Results Trending



Performance Testing Results

Highlights

- Test Results Comparison and Triage



kpi_name	result	baseline_result	last_run_result	delta_percentage_baseline	delta_percentage_last_run
sparse	11.48425	7.29525	7.29525	57.42 %	57.42 %
typeahead_access	0.1032	0.0692	0.0692	49.13 %	49.13 %
typeahead_tea	0.1197	0.0904	0.0904	32.41 %	32.41 %
inputlookup_geo_attr_countries	0.0631	0.0515	0.0515	22.52 %	22.52 %
typeahead_acc	0.1092	0.093	0.093	17.42 %	17.42 %
typeahead_teardow	0.1069	0.0959	0.0959	11.47 %	11.47 %
inputlookup_oracle_ora_codes	0.1744	0.1577	0.1577	10.59 %	10.59 %
typeahead_ac	0.1561	0.1462	0.1462	6.77 %	6.77 %
typeahead_tear	0.1016	0.0965	0.0965	5.28 %	5.28 %
tag_change_stats_by_user	96.627	91.825	91.825	5.23 %	5.23 %

« prev 1 2 3 4 5 next »

Developer Portal

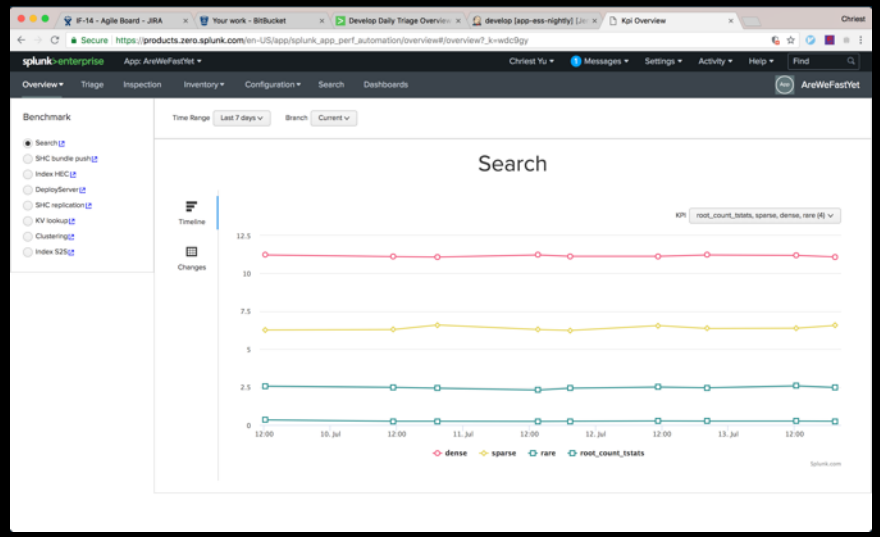
Use Case - Developer Portal

Problem

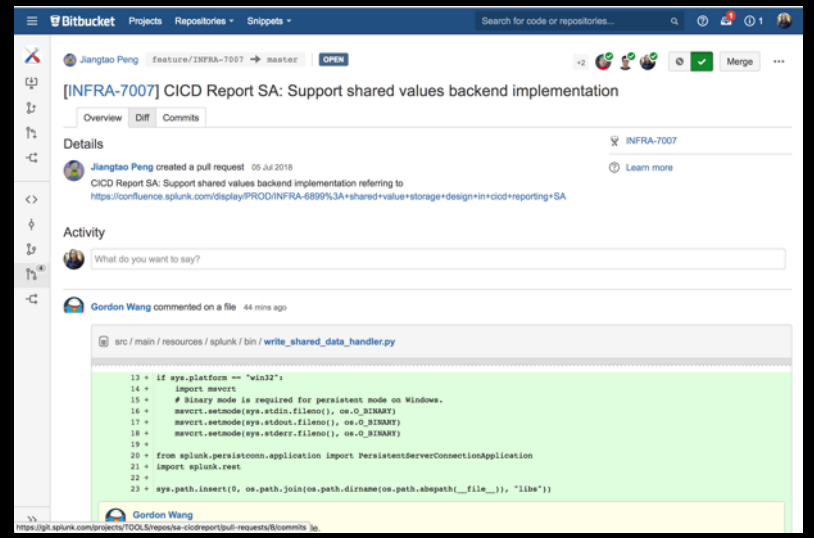
- Daily development info is distributed across many systems
- Difficult to onboard new engineers to CI/CD platform

As a Splunker…

Morning



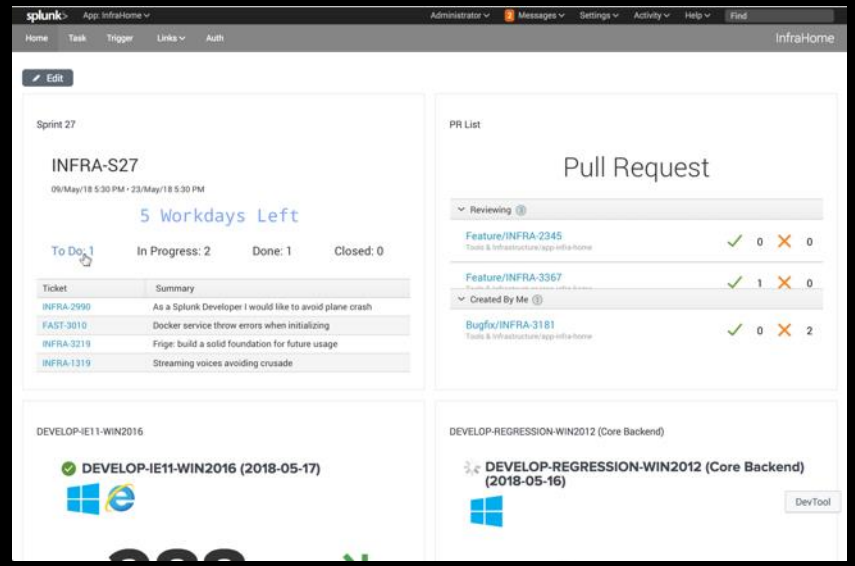
Afternoon

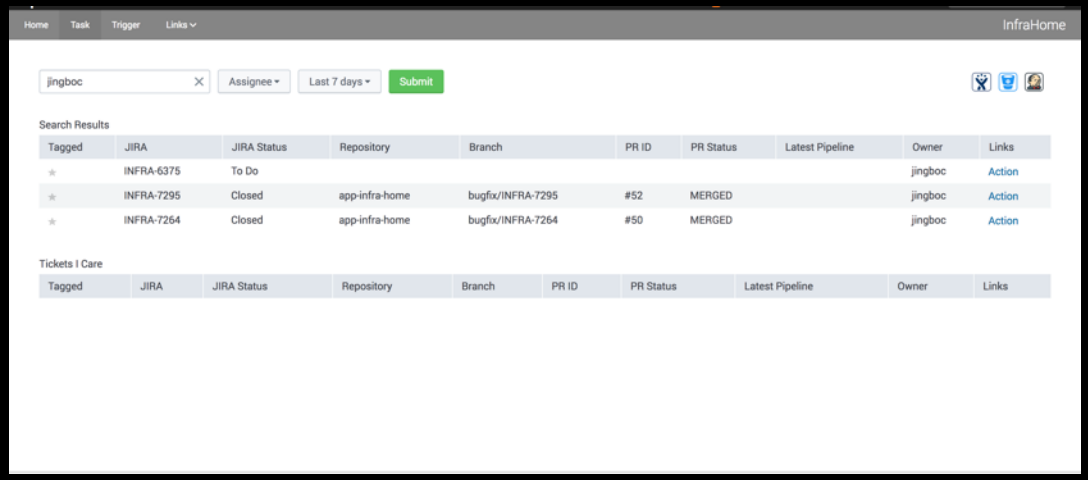


Developer Portal

Solution

- Centralized and consolidated portal
- Dashboards and tools for different personas
- Easy to work with other systems, such as JIRA, Bitbucket, Jenkins etc.





- Easy to use personal dashboard
- Widgets for specific need

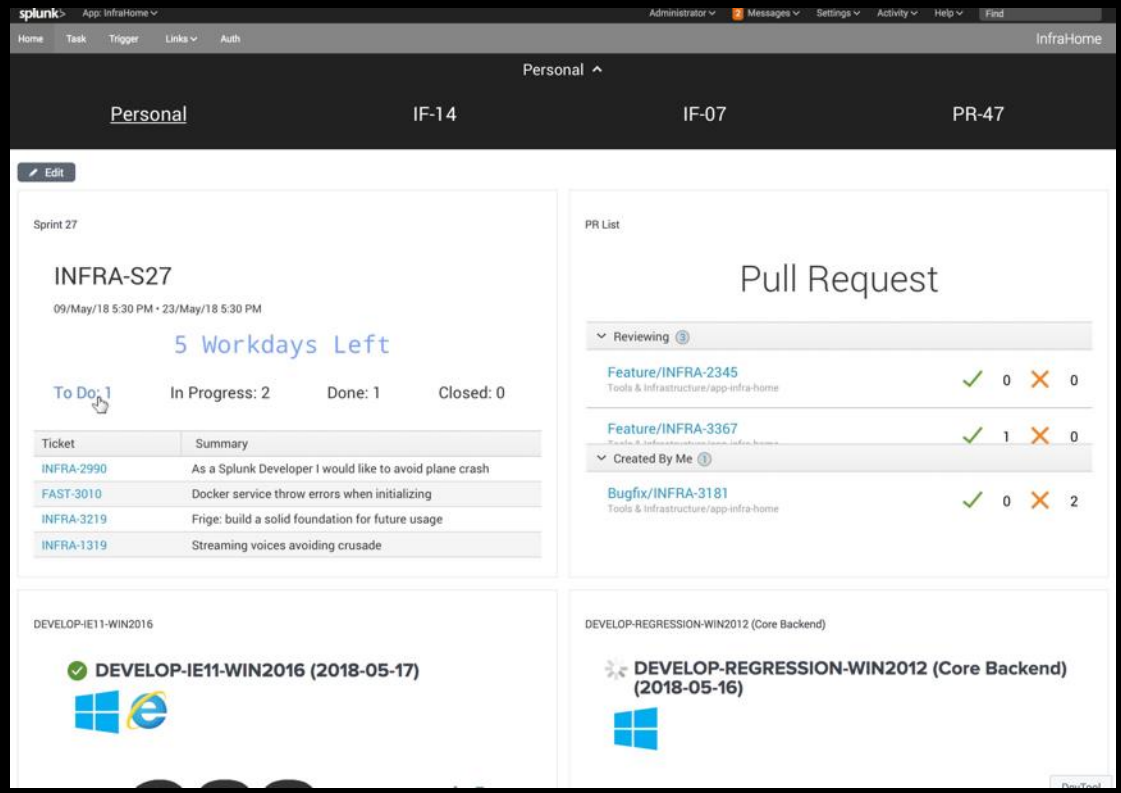
- Search for ad-hoc connections
- Follow up action links navigating to other systems
- Bookmark items care most

splunk> .conf18

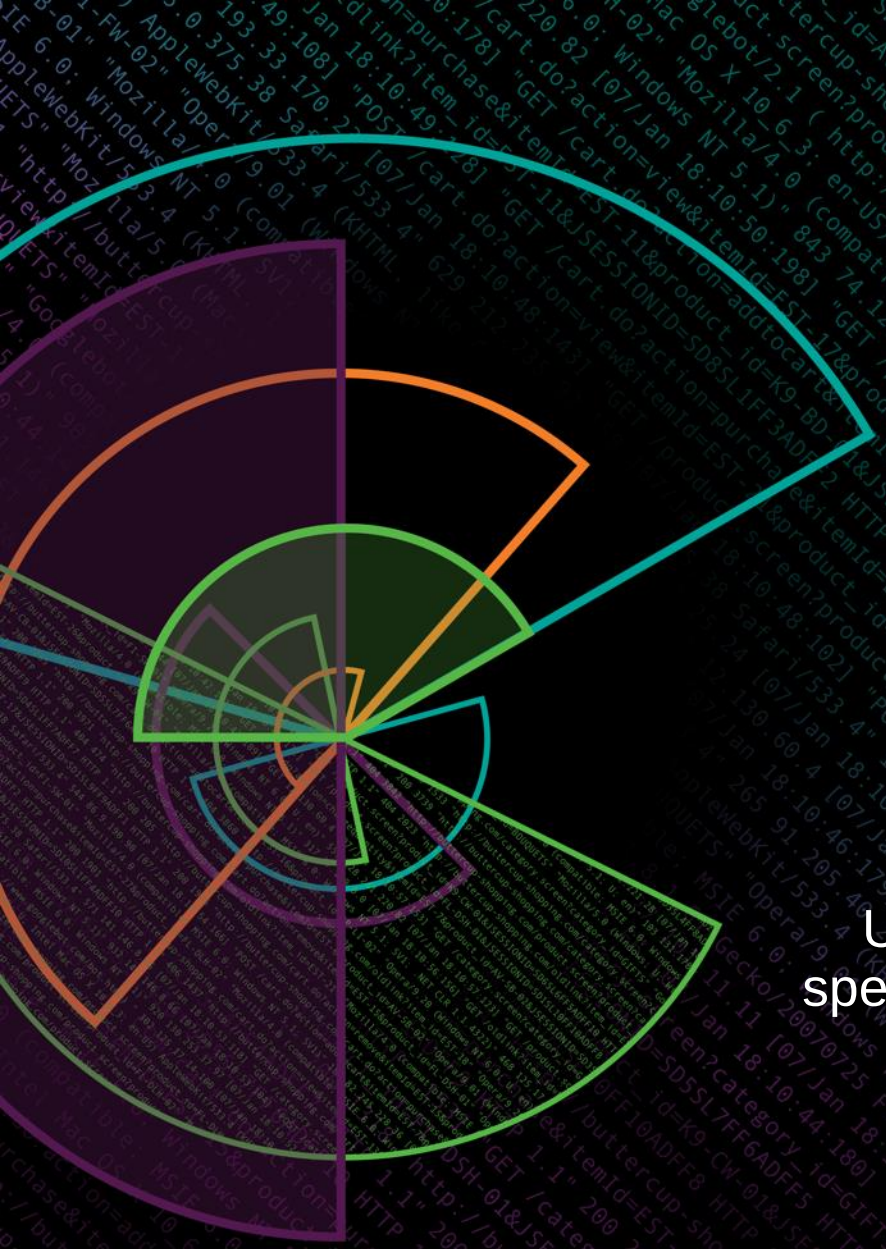
Developer Portal

Highlights

- Home Page
  - Onboard Wizard
  - Team View
  - Widgets
- Task Page
  - Integration with Jira, Bitbucket, Jenkins, etc
  - More integration is coming, like GitLab
- Notification
  - Slack integration, TODO list



splunk> .conf18



Infrastructure Engineering CI/CD Splunk Scenarios

Using Splunk to measure development productivity and solve for speed in developer teams by collecting build data. See how Splunk "builds Splunk with Splunk," and how Splunk helps developers collaborate through tools such as Git, JIRA, Jenkins and Slack.

splunk> .conf18

EDDIE SHAFAQ

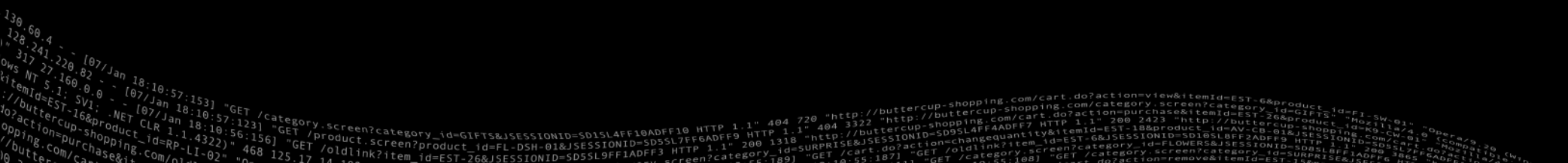
Splunk Infrastructure

splunk> .conf18

Our Speaker

Splunk Infrastructure

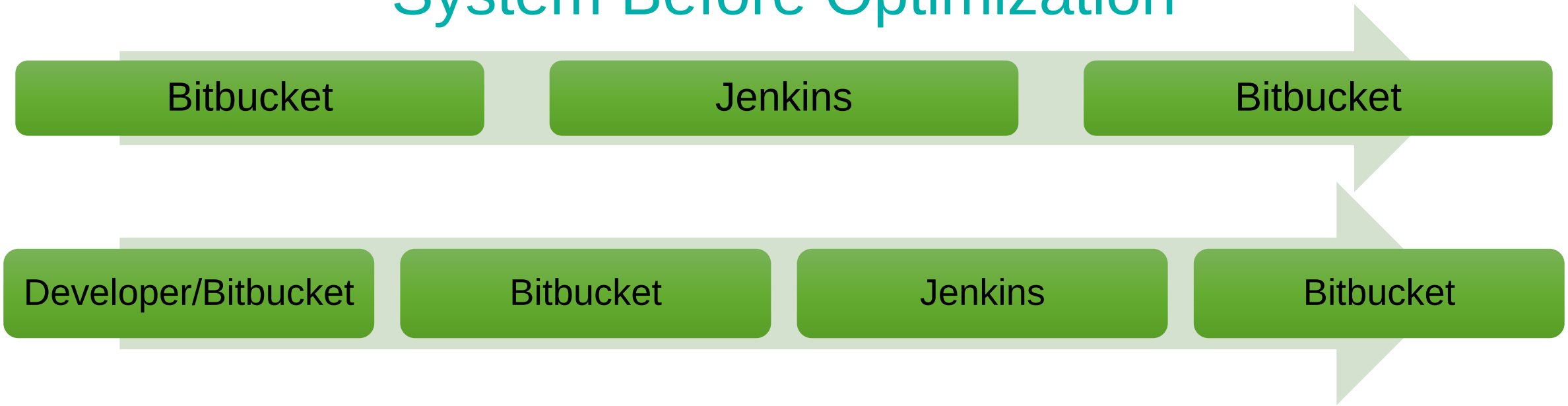
First joined Splunk in August 2011 as a Systems Administrator. Aided in expanding engineering support in "exotic operating system" (AIX, HPUX, S390X and PowerLinux). Served as a member of release engineering to address operational and infrastructure support for products team. Currently serving an infrastructure leadership role around Core Engineering and Release Engineering services.



The Goal Set by Management

- Improve Developer Productivity
  - Our contribution: Get CI test results to developers faster

System Before Optimization

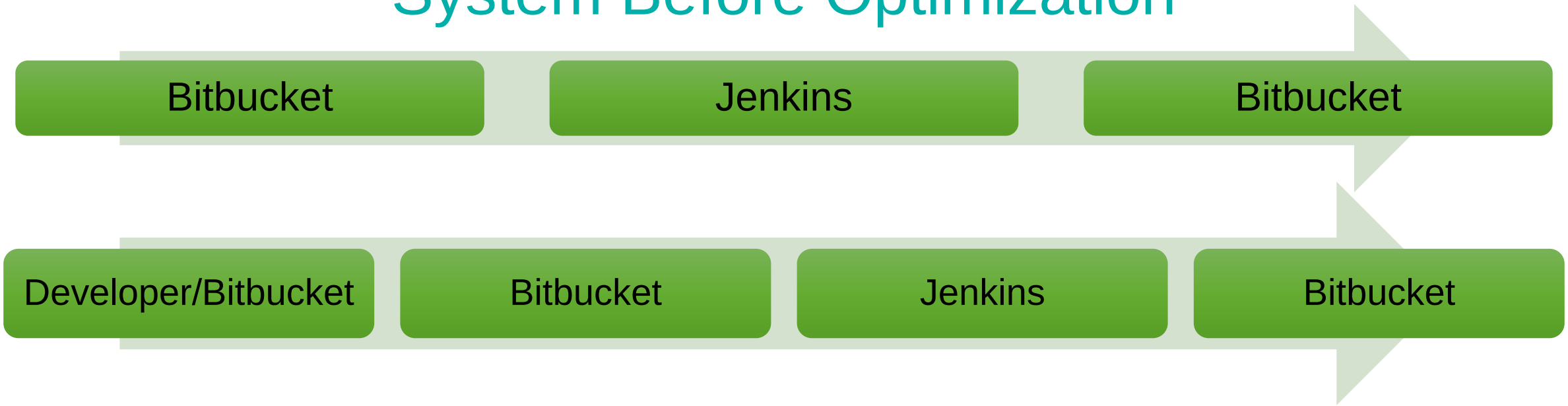


The system had **15 dedicated Linux agents** to perform continuous integration testing

Each job ran for approximately **54 minutes**, performing a build of Splunk and running a set of validation tests

That meant it could perform an approximate average of **17 jobs per hour**

System Before Optimization



The system had **15 dedicated Linux agents** to perform continuous integration testing

Each job ran for approximately **54 minutes**, performing a build of Splunk and running a set of validation tests

That meant it could perform an approximate average of **17 jobs per hour**

System Before Optimization

If more than **17 triggers** were received in a **one hour** period the excess **triggers were queued** waiting for a Linux agent to run on

Under “normal” circumstances the system operated with minimum delays, however during peak load periods when the pressure on developers was the highest…

We experienced **significant delays** resulting in frustration and phone calls as the **engineers waited** for results of the validation test jobs they were required to run before they could commit thier work

Our Analysis

Our Analysis

Understanding the Situation

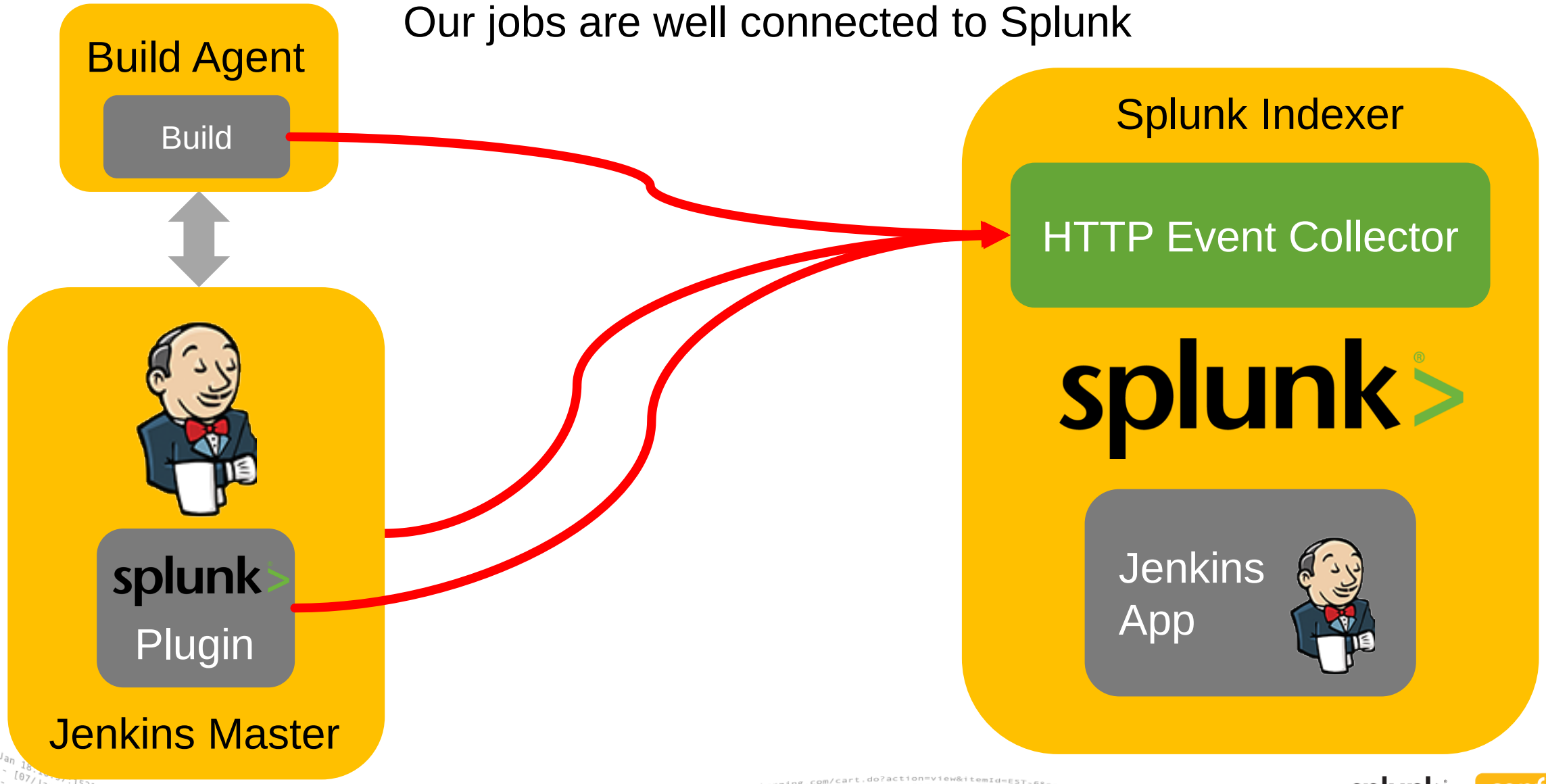
There were four factors that affected the delay developers experienced while waiting for test results

- Build time – how long it takes to build the Splunk executables
- Test time – how long it takes to perform the required set of tests
- Queue time – how long before the test actually started to run
- Notification – how long before developers know the test results

We need to quantify each of those factors and determine what we could do to mitigate their effects on the overall time

How We Collected The Data

Our jobs are well connected to Splunk



The Splunk HTTP Event Collector

Simple to send custom data to your Splunk instance

In Bash

Format your data as a JSON string:

```
jsonData="{\"time\": 12345, \"index\": \"YourIndex\", \"sourcetype\": \"YourSourceType\", \"source\":
\"YourSource\", \"event\": \{\"YourFieldName\": \"SomeData\", more json formatted data goes here}}”
```

Include as much json formatted information as you need in the event section

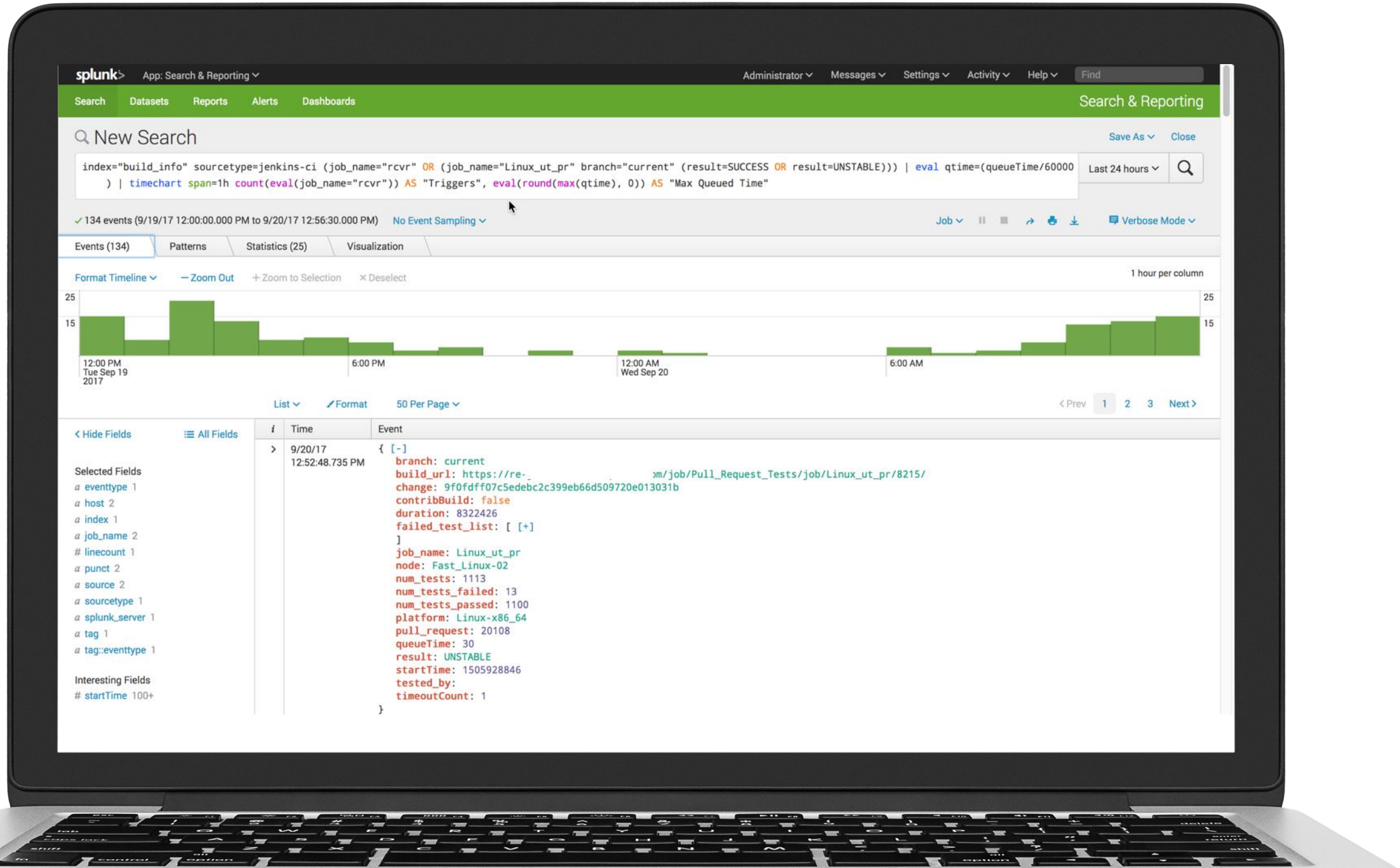
Then execute a curl call:

```
curl \
  --tlsv1.2  --header "Authorization: <Splunk_auth_token_goes_here>" \
  --header "Content-Type: application/json" \
  --request 'POST' \
  --data $jsondata \
```

https://YourSplunkInstance/services/collector/event

Its that simple…

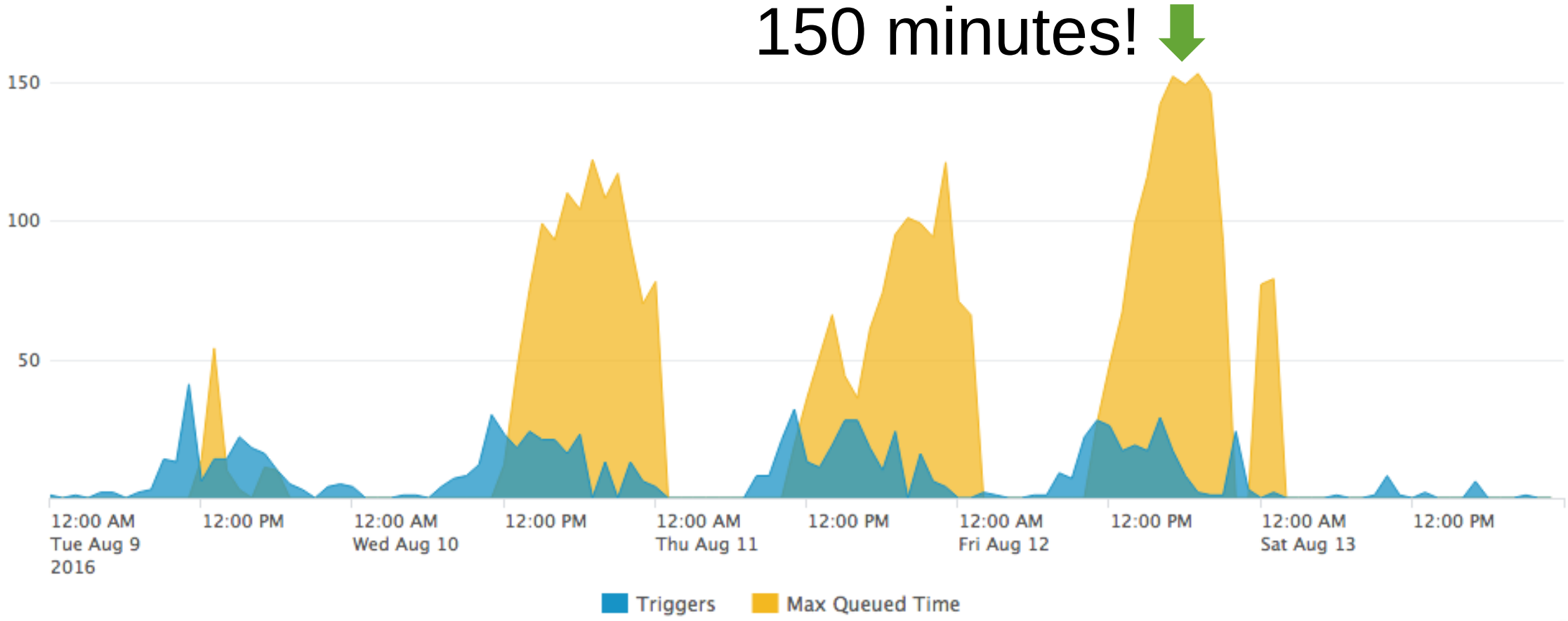
splunk> .conf18



Queue Times Before Optimization

Peak load period had significant delays

150 minutes! ⬇



splunk> .conf18

The Search We Use to Analyze Jobs

Of course this won't work for you, but…

```
index="jenkins_console"  host=”aJenkins.ourco.com" source="*Linux_ut_pr*" ("make -j48 || exit 0" OR "Install the project..." OR "Core build is done" OR "run the tests again" OR "Starting backend unit tests" OR "Package and publish Splunk" OR "starting Linux 64 test" OR "fetch the jenkins scripts directory" OR ("nodes run &gt;&gt;&gt;&gt; STARTING ACTION" AND "Write splunk-version.txt") OR ("STARTING COMMAND" AND "Running the contrib command") OR "Done all requested steps") | rex field=source "job/Pull_Request_Tests/job/Linux_ut_pr/(?&lt;build_number&gt;.*)/console" | eval buildStep=case(searchmatch("fetch the jenkins scripts directory"),"start", searchmatch("starting Linux 64 test"),"clone", searchmatch("Running the contrib command"),"chroot", searchmatch("Write splunk-version.txt"),"contrib", searchmatch("make -j48 || exit 0"),"setup", searchmatch("Core build is done"),"build_1", searchmatch("Install the project..."), "build_2", searchmatch("Starting backend unit tests"), "package", searchmatch("run the tests again"), "tests_1", searchmatch("Package and publish Splunk"), "tests_2", searchmatch("Done all requested steps"), "publish") | chart values(_time) by build_number, buildStep    | eval gc = round(('clone' - 'start')/60) |eval cs = round((chroot - 'clone')/60) | eval "cb" = round((contrib - chroot)/60) | eval "bs" = round((setup - contrib)/60) | eval "cub" = round((build_1 - setup)/60) | eval "cbc" = round((build_2 - build_1)/60) | eval "ts" = round((package - build_2)/60) | eval "pst" = round((tests_1 - package)/60) | eval "sst" = round(('tests_2' - 'tests_1')/60) | eval "pub" = round(('publish' - 'tests_2')/60) | search cb &lt; 5 | search sst &gt; 0 | search pst &lt; 25 |chart values(pub) as Publishing, values(sst) as "Sequential Smoke Tests", values(pst) as "Parallel Smoke Tests", values(ts) as "Test Setup", values(cbc) as "Core Build Continues", values(cub) as "Core and UI build", values(bs) as "Build Setup", values(cb) as "Contrib Build", values(cs) as "Chroot Setup", values(gc) as "Git Clone" by build_number
```

The Search We Use to Analyze Jobs

Collect the specific log file lines we will use in our analysis

```
index="jenkins_console"  host="_aJenkins.ourco.com
" source="*Linux_ut_pr*" ("make -j48 || exit 0" OR
"Install the project..." OR "Core build is done" OR
"run the tests again" OR "Starting backend unit
tests" OR "Package and publish Splunk" OR "starting
Linux 64 test" OR "fetch the jenkins scripts
directory" OR ("nodes run >>>> STARTING ACTION" AND
"Write splunk-version.txt") OR ("STARTING COMMAND"
AND "Running the contrib command") OR "Done all
requested steps")
```

splunk> .conf18

The Search We Use to Analyze Jobs

Extract the job number

```
| rex field=source
"job/Pull_Request_Tests/job/Linux_ut_pr/(?<build_number>.*)/console
```

The Search We Use to Analyze Jobs

Build a table of time stamps by job number and job step

```
| eval buildStep=case(searchmatch("fetch the jenkins scripts directory"),"start", searchmatch("starting Linux 64 test"),"clone", searchmatch("Running the contrib command"),"chroot", searchmatch("Write splunk-version.txt"),"contrib", searchmatch("make -j48 || exit 0"),"setup", searchmatch("Core build is done"),"build_1", searchmatch("Install the project..."), "build_2", searchmatch("Starting backend unit tests"), "package", searchmatch("run the tests again"), "tests_1", searchmatch("Package and publish Splunk"), "tests_2", searchmatch("Done all requested steps"), "publish")

| chart values(_time) by build_number, buildStep limit=50
```

The Search We Use to Analyze Jobs

Calculate the deltas between job steps

```
| eval gc = round(('clone' - 'start')/60) |eval cs
= round((chroot - 'clone')/60) | eval "cb" =
round((contrib - chroot)/60) | eval "bs" =
round((setup - contrib)/60) | eval "cub" =
round((build_1 - setup)/60) | eval "cbc" =
round((build_2 - build_1)/60) | eval "ts" =
round((package - build_2)/60) | eval "pst" =
round((tests_1 - package)/60) | eval "sst" =
round(('tests_2' - 'tests_1')/60) | eval "pub" =
round(('publish' - 'tests_2')/60) | search cb &lt;
5 | search sst &gt; 0 | search pst &lt; 25
```

The Search We Use to Analyze Jobs

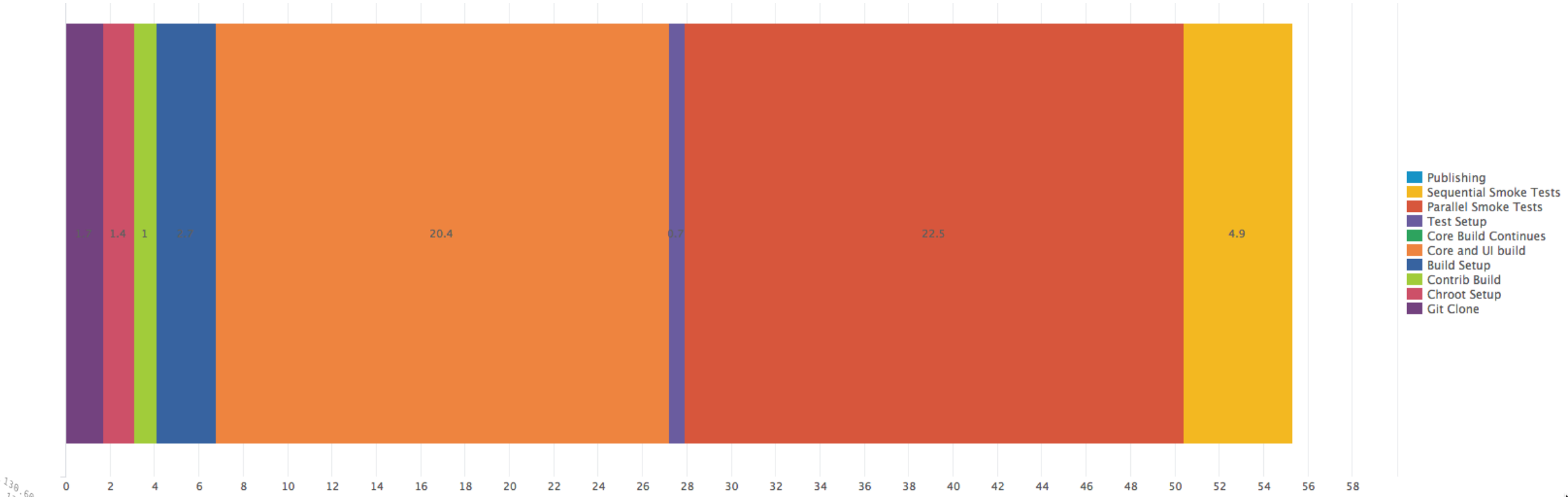
Build the final table with user friendly names for display

```
|chart values(pub) as Publishing, values(sst) as
"Sequential Smoke Tests", values(pst) as "Parallel
Smoke Tests", values(ts) as "Test Setup",
values(cbc) as "Core Build Continues", values(cub)
as "Core and UI build", values(bs) as "Build
Setup", values(cb) as "Contrib Build", values(cs)
as "Chroot Setup", values(gc) as "Git Clone" by
build_number limit=50
```

Our Analysis

How long does each step of a job take?

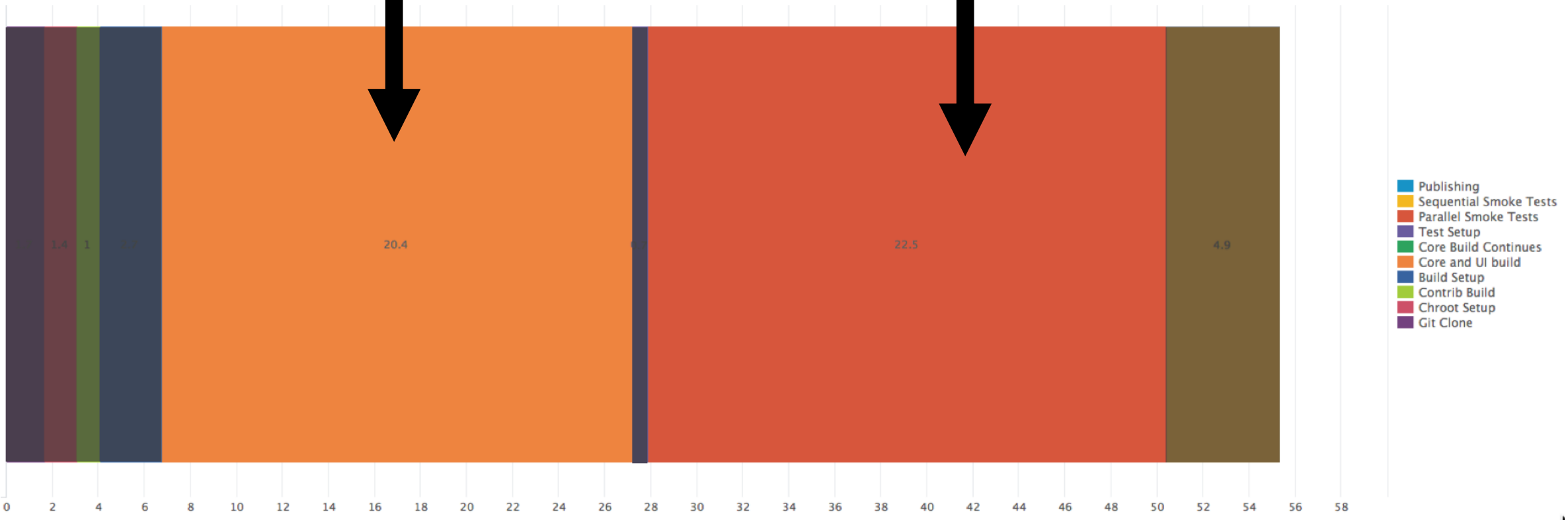
- We used the previous search to chart the time each step took



Our Analysis

How long does each step of a job take?

- We Identified the two portions of the job that took the longest
- The Splunk build (in orange) and the validation tests (in red)

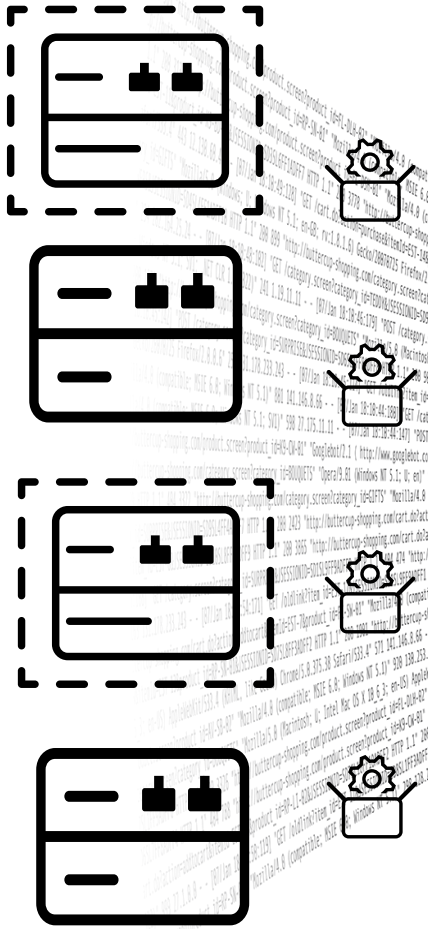


Speeding up the Splunk Build

splunk> .conf18

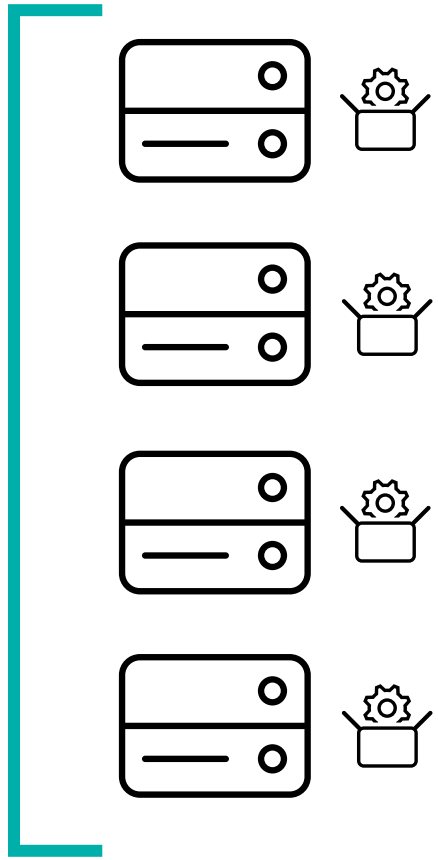
Distcc Architecture

Physical Machines and VMs:



- All DistCC Build Clients and servers use the same build toolchain and chroot
- 15 Build agents
- 12 compile nodes
- DistCC server is used ONLY for compile
- `make -j48`

12 x 20 Core VMs



DistCC VS Normal Build

Building Splunk with DistCC

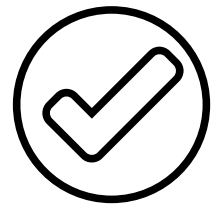
- 24 Min Build
  - `make -j 24`
  - Web UI –j 1
  - Optimal 24 core VM agent

24m →

- 19 Min Build
  - `make -j 48`
  - Web UI –j 1
  - Optimal 24 core VM agent
  - 12 DistCC hosts

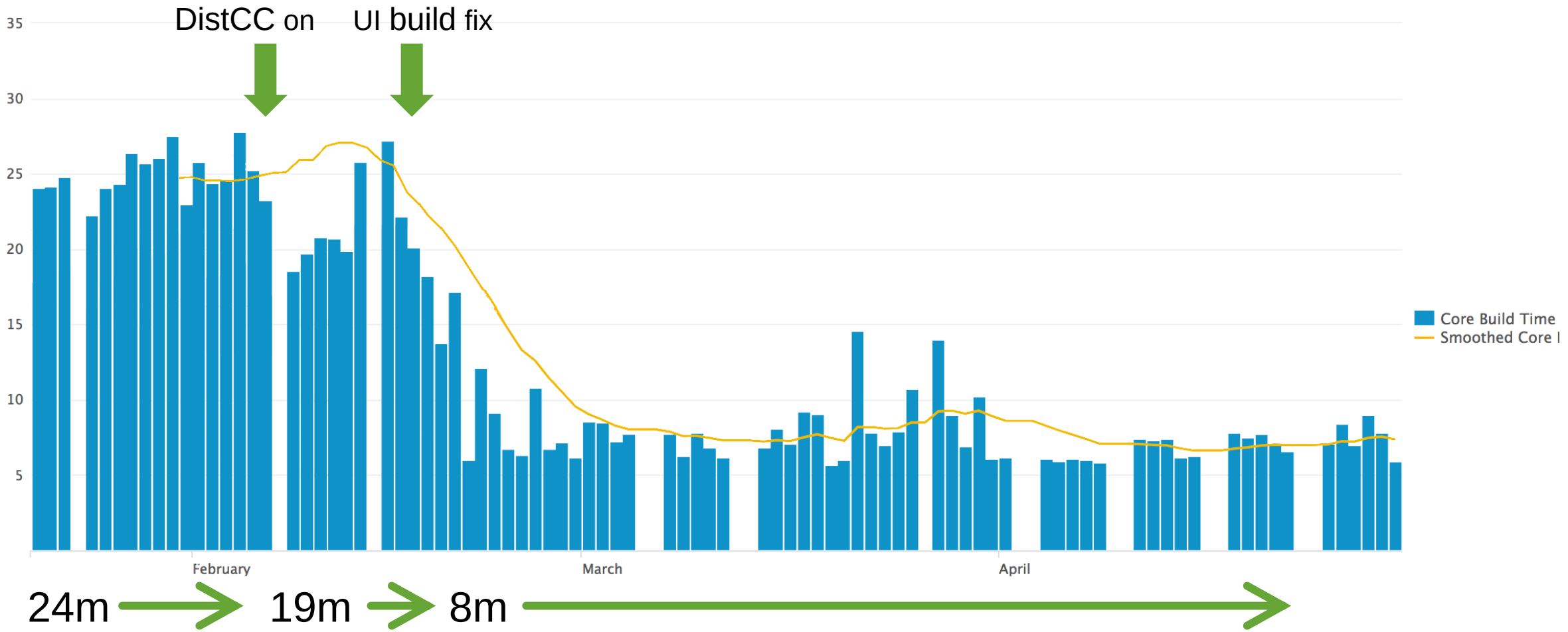
19m →

- 8 Min Build
  - **`make -j 48`**
  - **Web UI –j 6**
  - Optimal 24 core VM agent
    - 12 DistCC hosts

8m → 

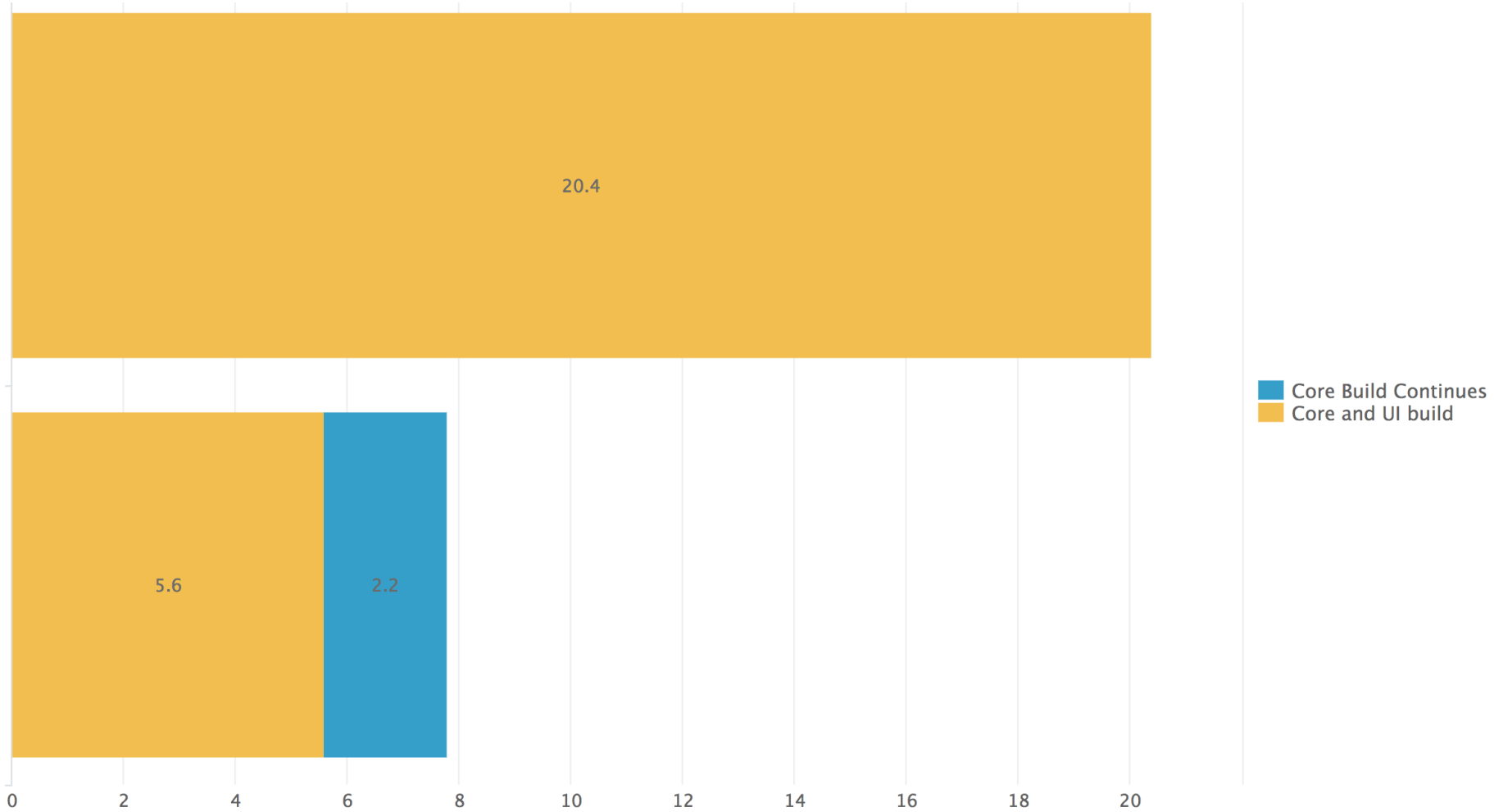
Build Time Improvement - Results

Dramatic reduction in the overall build time



Build Time Improvement - Results

Dramatic reduction in the core compile time

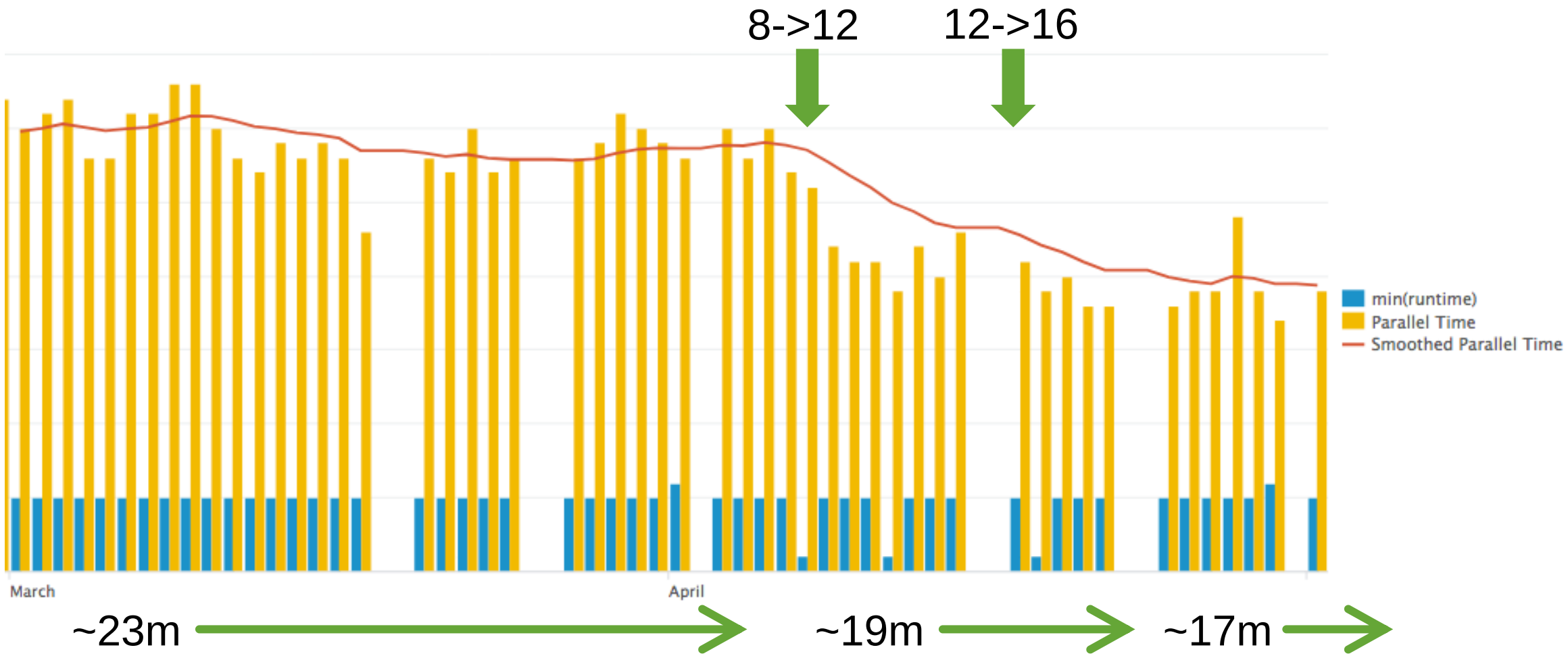


Speeding up Testing

splunk> .conf18

Increasing Test Parallelization

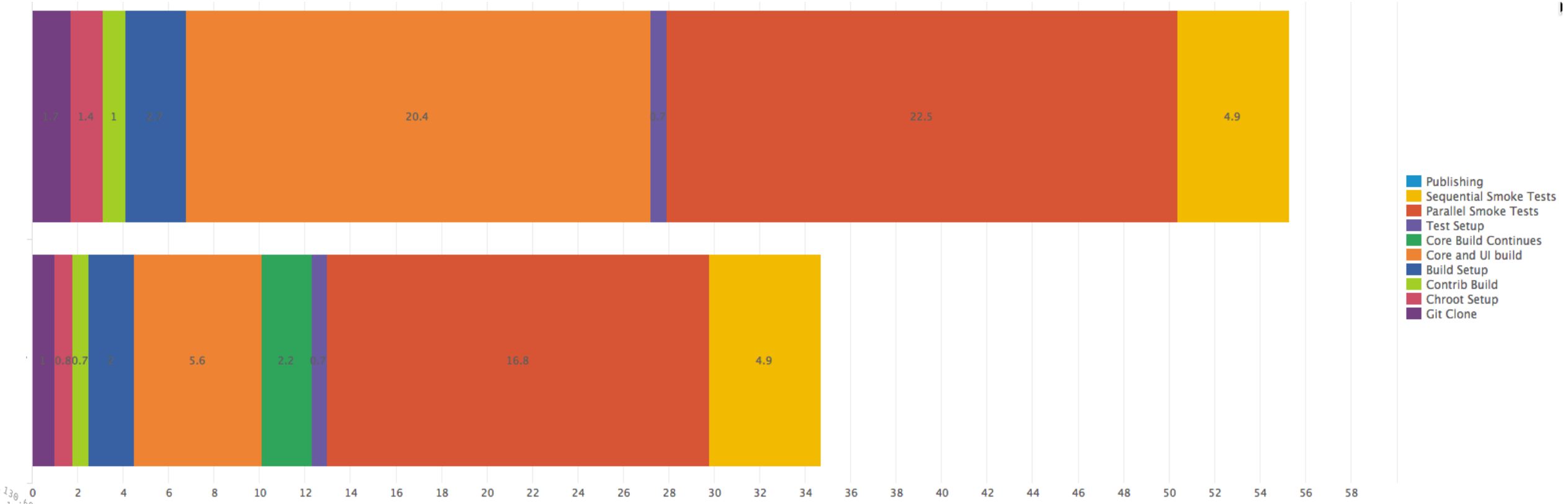
Improvement in the overall test time as parallel instances were increased



The Final Overall Result

Overall job time reduced to ~35 minutes

- Build time reduced to under 8 minutes
- Test time reduced to under 22 minutes



splunk> .conf18

Managing the Agents

splunk> .conf18

Our Analysis

Breaking Down The Timing

Reducing the Pull Request Queue time

Analysis:

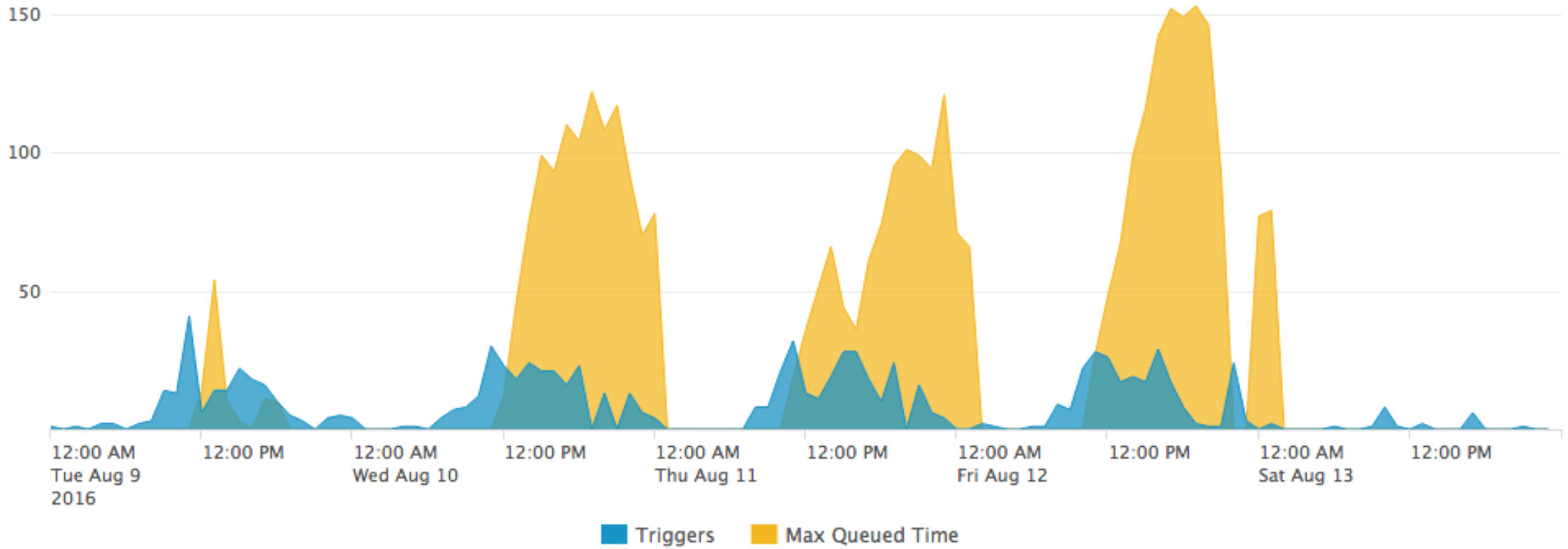
- Developers are only waiting for results from pull request tests
- Triggers on commits to some branches and all pull requests

Mitigation:

- Use more agents for the pull request tests
- Manage the allocation of agents to tasks -> shift resources to pull requests when the queue starts to climb
- Add 5 “standby agents” with reduced capabilities that are powered up on demand -> smaller footprint on VM hosts

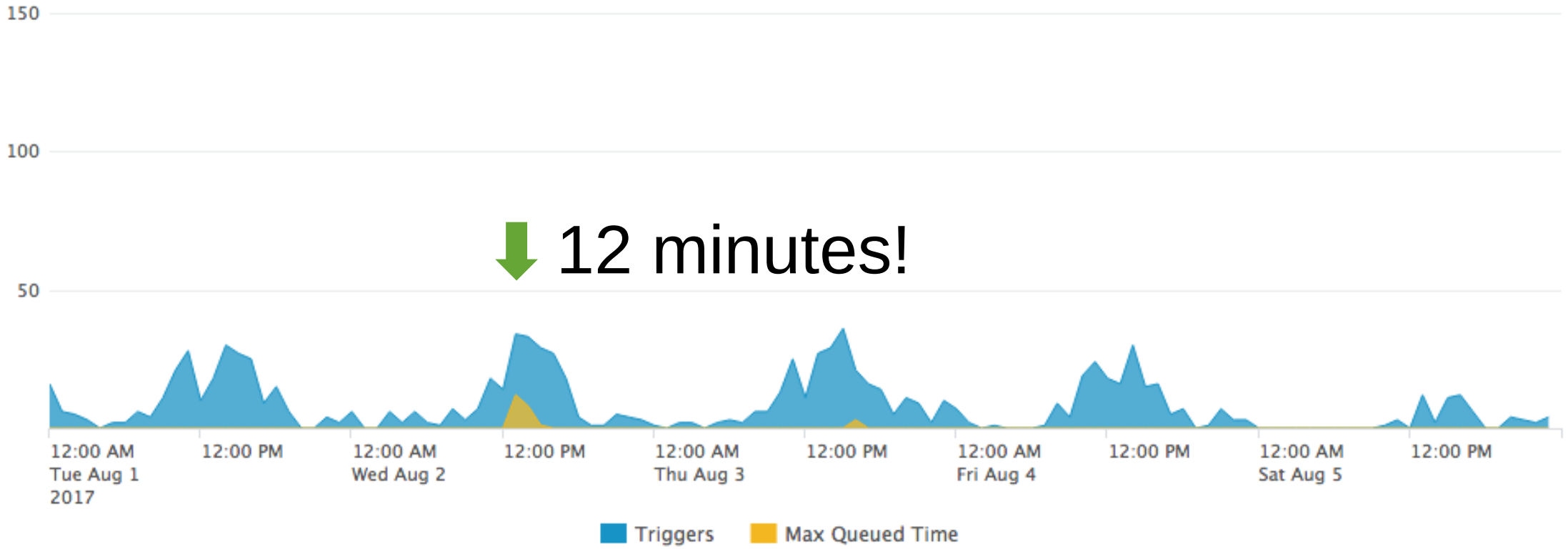
Before Optimization

Peak load period –> significant delays



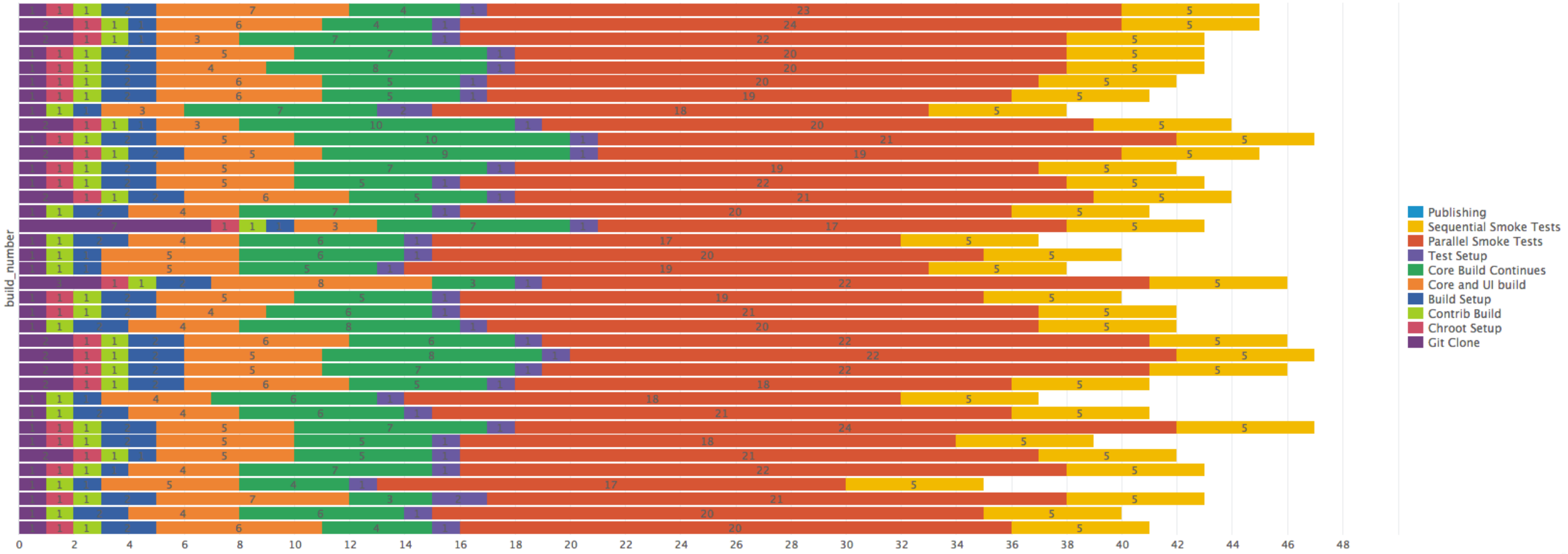
After Optimization

Recent similar trigger conditions



The Search We Used to Analyze Builds

Here is what the full output of the search we presented earlier looks like



Thank You

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

Q&A

splunk> .conf18