

# 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, Splunk>, Listen to Your Data, The Engine for Machine Data, Splunk Cloud, Splunk Light and SPL are trademarks and registered trademarks of Splunk Inc. in the United States and other countries. All other brand names, product names, or trademarks belong to their respective owners. © 2018 Splunk Inc. All rights reserved.

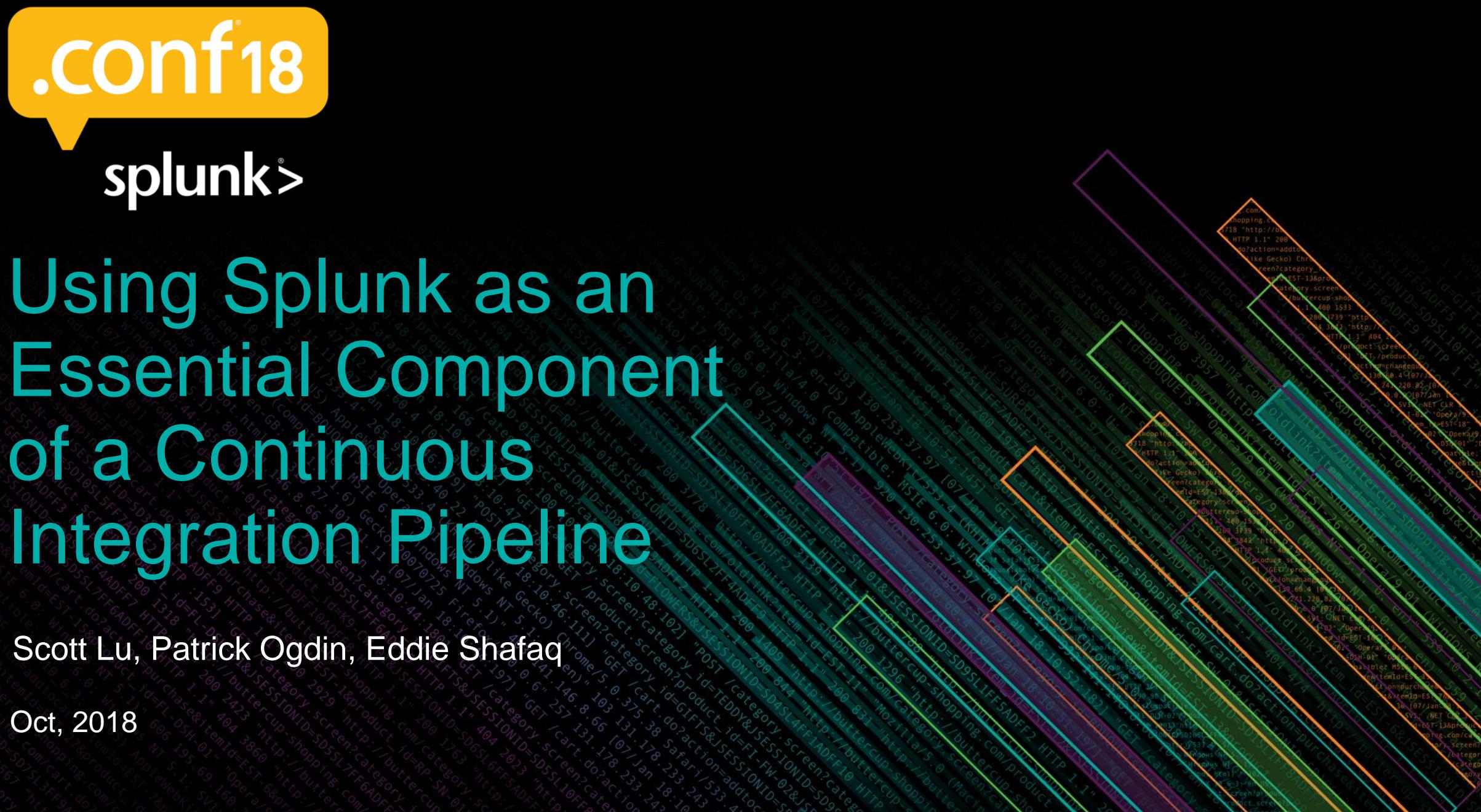


splunk>

# 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

# 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.

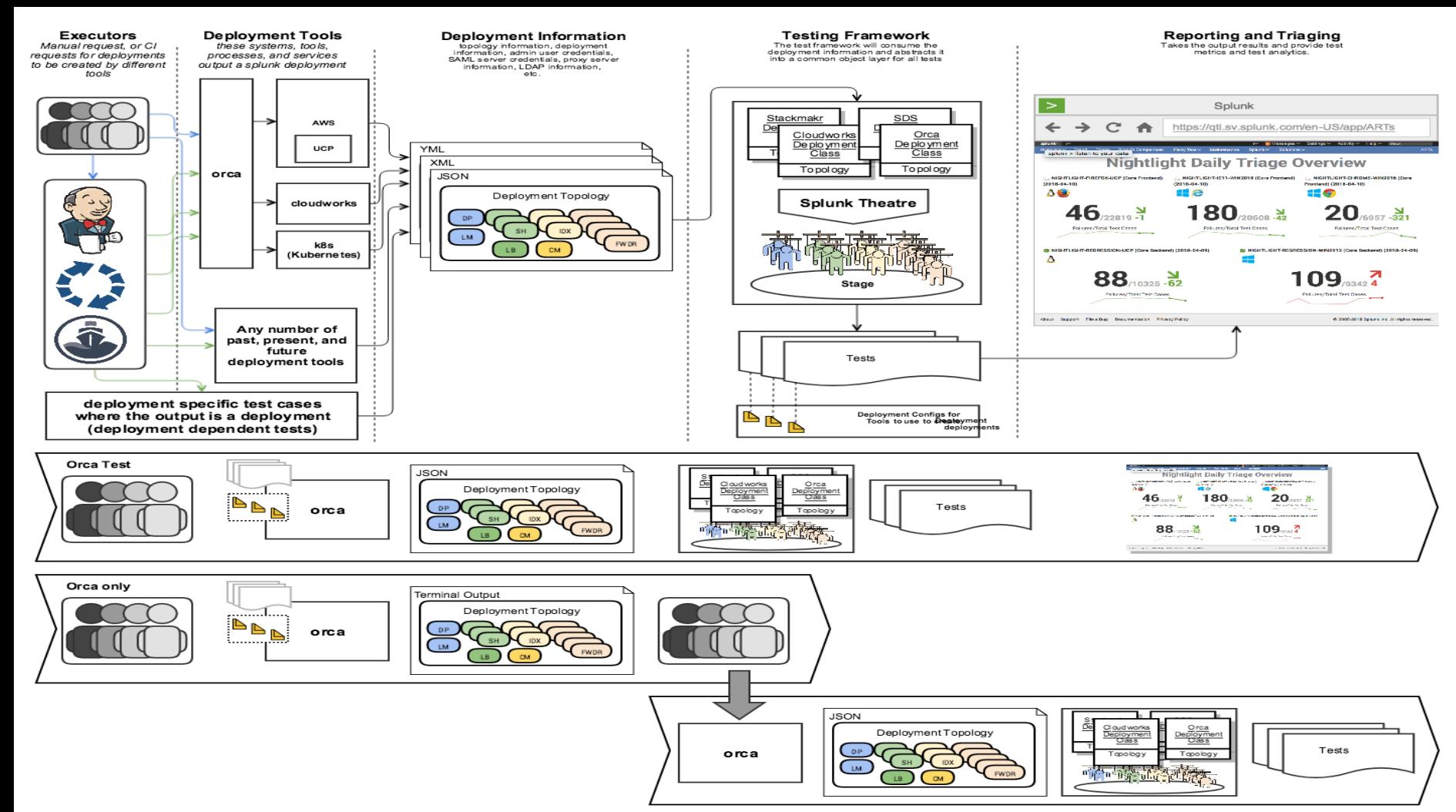
# Introduction

## 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

Open stories/tasks/bugs

# Problem: Building Software is Complicated



# Internal Customer CI/CD Splunk Scenarios

# 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?

Skipping 27 KB.. <a href="#">Full Log</a>						
Y1U2W5pzzohpoGApnsACe8u1nyAAAA[	Omdeclare	-rx	TEST_TYPE=			
0	0	0	#1532	Sep 24, 2018 4:12 AM		
0	0	0	#1531	Sep 24, 2018 12:12 AM		
0	0	0	#1530	Sep 23, 2018 8:12 PM		
0	0	0	#1529	Sep 23, 2018 4:12 PM		
0	0	0	#1528	Sep 23, 2018 12:12 PM		
0	0	0	#1527	Sep 23, 2018 8:12 AM		
0	0	0	#1526	Sep 23, 2018 4:12 AM		
0	0	0	#1525	Sep 23, 2018 12:12 AM		
0	0	0	#1524	Sep 22, 2018 8:12 PM		
0	0	0	#1523	Sep 22, 2018 4:12 PM		
0	0	0	#1522	Sep 22, 2018 12:12 PM		
00:30:05	drwxr-xr-x	2	root root	4096 Sep 7 00:29 .		
00:30:05	drwxr-xr-x	3	root root	4096 Sep 7 00:29 ..		

# Pipeline Visualization

## Solution

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

## Build Information

Build Summary

Variable Name	Value
Build Number	466346
Host	jenkins-core-dom-dev
Job Duration	00:49:40.61
Job Name	Docker/pytest_docker
Job Result	FAILURE
Job Started At	2018-09-05 10:32:33
Job Type	Pipeline
Node	(master)
Queue Time	00:07:03.706
Scm	git
Trigger By	Started by upstream project "CoreQA/backend-for-dom" build number 1,436

 Splunk App for Jenkins

Ucp Container Flavor	sys_default
Aggregate Report Name	clustering_index_replication_suite_2[-platform=linux...longrunning_2]
Dockerargs	{"imageName":"repo.splunk.com/splunk/products/splunk-core-backend-test:2.3","remotePath":"/tmp/","script":null,"portBindings":null,"containerEnv":[{"test_report=test-results-230.xml"}],"ports":null}
Jobconfig	{"NUM_OF_VM":15,"files":"\$HOME/.ssh/testcube_id_rsa","gitRepoUrl":"ssh://git@git.splunk.com:7999/splcore/qa.git","branchName":"release/orangeswirl","runCounter":1,"PYTEST_MAX_THREADS":1}
Jobtimeouthours	3
Os	container
Product	splunk
Scm	git
Splunk Build Branch	orangeswirl
Splunk Build Commit	fd78ef8c8aea
Test Count	1
Test Path	clustering/index_replication/suite_2

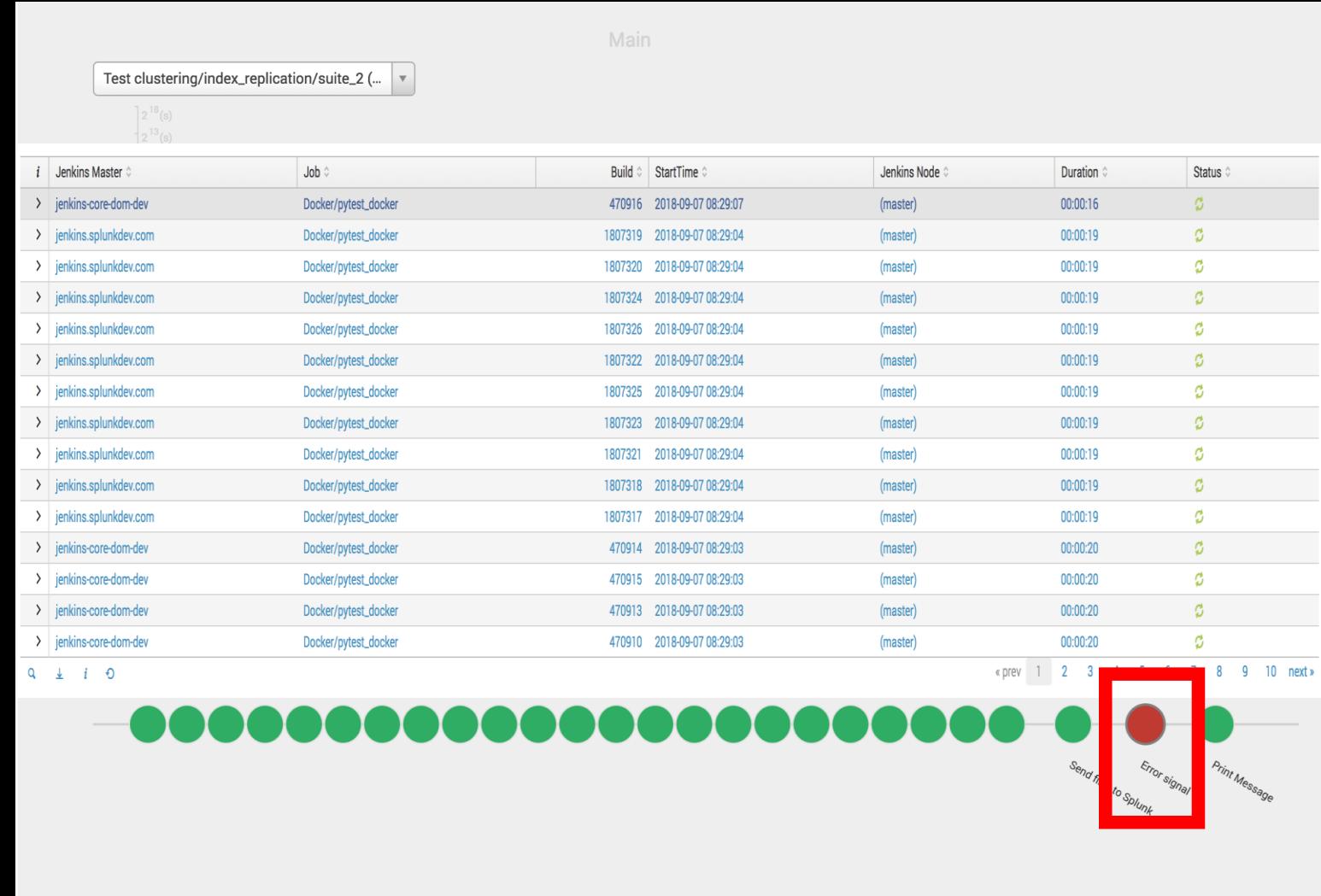
# Splunk App for Jenkins



# Pipeline Visualization

# Highlights

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



# Test Result Triage System

ARTs



# Test Triage System

## Problem

 **Splunk**

Jenkins Master for UI QA team.

All	Prior Releases	UI branch triggers	bugfix-eng	develop	galaxy	honeybuzz	ivory	jackhammer	scrum - analytics	scrum - dashboards
scrum - framework	scrum - search ui	scrum - sg-1	scrum - sg-2	scrum - splunk light	scrum - table ui	scrum - ui framework	scrum-cloud			
scrum-common-criteria	sustaining									

S	Name ↓	Last Duration	Last Failure	Passed Tests	Failed Tests	Total Tests	Build time	Cron Trigger
	<a href="#">develop » client » lin64-downloadtrial-chrome lin64</a>	3.4 sec	7 mo 24 days - #12	<a href="#">0 passed</a>	<a href="#">0 failed</a>	<a href="#">0 total</a>	2016-05-22 20:07	
	<a href="#">develop » client » lin64-downloadtrial-firefox lin64</a>	3 hr 47 min	7 days 17 hr - #727	<a href="#">18,753 passed</a>	<a href="#">139 failed</a>	<a href="#">18,892 total</a>	2017-01-11 06:20	Build periodically: H 6 * * *
	<a href="#">develop » client » lin64-downloadtrial-firefox lin64-disable-introspection</a>	8 hr 26 min	1 mo 19 days - #27	<a href="#">17,956 passed</a>	<a href="#">141 failed</a>	<a href="#">18,097 total</a>	2016-11-23 15:48	
	<a href="#">develop » client » lin64-downloadtrial-firefox lin64-non-ascii-non-default-user</a>	3 hr 48 min	2 days 13 hr - #65	<a href="#">16,832 passed</a>	<a href="#">211 failed</a>	<a href="#">17,043 total</a>	2017-01-09 10:33	
	<a href="#">develop » client » lin64-downloadtrial-firefox lin64-reverse-proxy-off</a>	18 hr	1 mo 24 days - #52	<a href="#">17,220 passed</a>	<a href="#">719 failed</a>	<a href="#">17,939 total</a>	2016-11-18 18:47	
	<a href="#">develop » client » lin64-downloadtrial-firefox lin64-root-endpoint</a>	3 hr 36 min	1 mo 19 days - #48	<a href="#">17,925 passed</a>	<a href="#">230 failed</a>	<a href="#">18,155 total</a>	2016-11-23 00:19	
	<a href="#">develop » client » lin64-downloadtrial-firefox lin64-smoke-saml-sso</a>	1 hr 19 min	1 mo 19 days - #465	<a href="#">0 passed</a>	<a href="#">0 failed</a>	<a href="#">0 total</a>	2016-11-23 16:36	
	<a href="#">develop » client » lin64-downloadtrial-firefox lin64-ssl</a>	13 hr	1 mo 24 days - #58	<a href="#">17,794 passed</a>	<a href="#">471 failed</a>	<a href="#">18,265 total</a>	2016-11-18 19:22	
	<a href="#">develop » client » lin64-downloadtrial-firefox lin64-burpsuite</a>	1 hr 9 min	1 yr 3 mo - #38	<a href="#">8,094 passed</a>	<a href="#">2,061 failed</a>	<a href="#">10,155 total</a>	2015-12-15 16:23	
	<a href="#">develop » client » lin64-downloadtrial-saucelabs-chrome win2012</a>	9 hr 36 min	4 mo 22 days - #27	<a href="#">11,771 passed</a>	<a href="#">813 failed</a>	<a href="#">12,584 total</a>	2016-08-22 14:23	
	<a href="#">develop » client » lin64-downloadtrial-saucelabs-firefox win2008</a>	16 hr	5 mo 2 days - #122	<a href="#">12,127 passed</a>	<a href="#">735 failed</a>	<a href="#">12,862 total</a>	2016-08-12 11:05	

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

# 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)

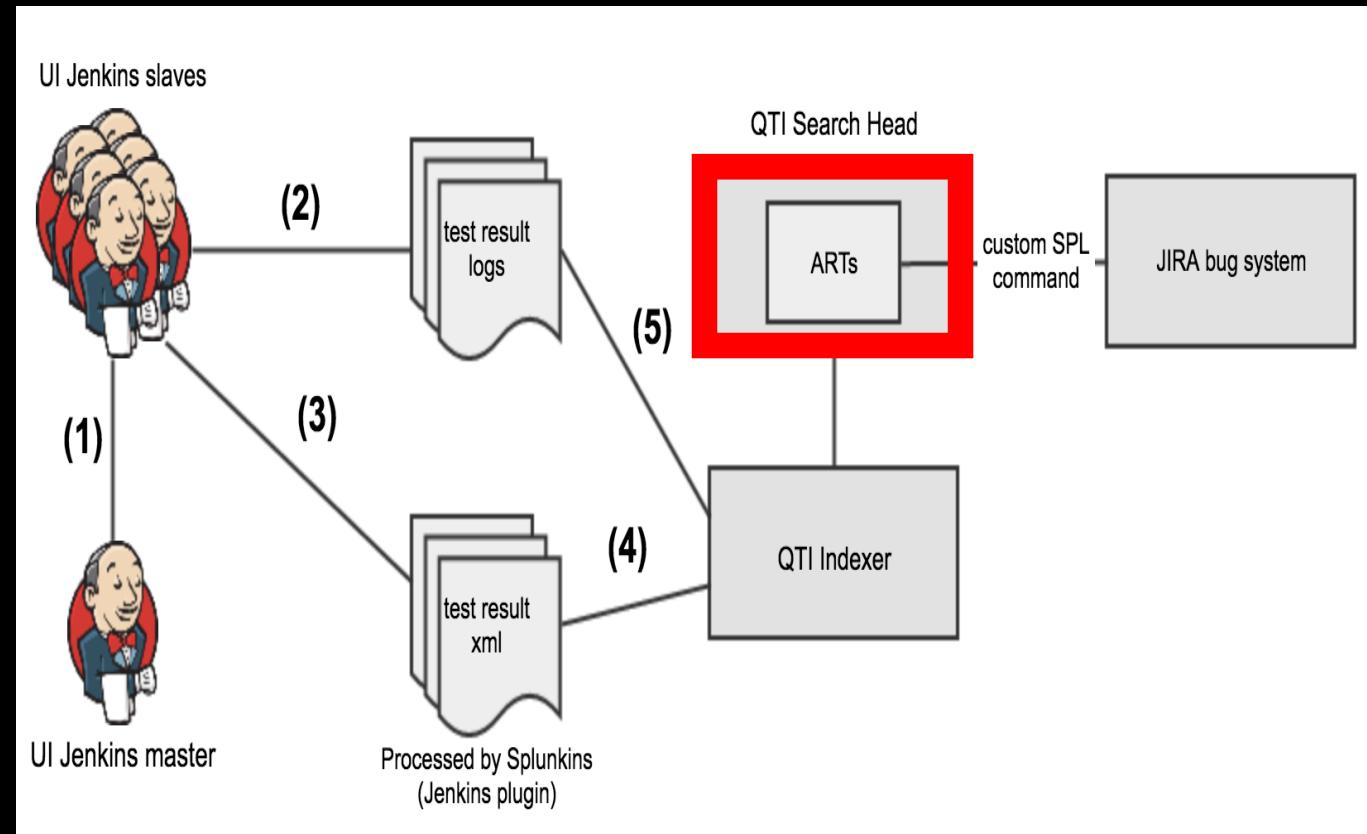
Job Name ↓	All	Failed (Aug.3rd)	Failed (Aug.11th)	Failed with JIRA	Status	Owner	Comments
<a href="#">client_clustering_webdriver_master</a>	80	<a href="#">8</a>	<a href="#">7</a>				
<a href="#">client_clustering_webdriver_searchhead</a>	21	<a href="#">1</a>	<a href="#">1</a>		Jian		Fixed two timing test script issues: <a href="#">pull-request 875</a> merged
<a href="#">client_clustering_webdriver_slave</a>	30	<a href="#">3</a>	<a href="#">3</a>				
<a href="#">client_forwarder_mgmt_webdriver</a>	175	<a href="#">10</a>	<a href="#">10</a>	4	Jian		<p><a href="#">pull-request 874</a> merged</p> <p>6 of the failures are due to pre-created serverclasses have 0 mapped (should be 1), cannot be reproduced locally</p> <p>3 tests fixed, <a href="#">pull-request 911</a> merged</p> <p>Passed locally: <code>test_unsupported_different_restartsplunkd</code>, <code>test_unsupported_different_stateonclient</code>, <code>test_unsupported_no_unsur</code></p> <p> <a href="#">SPL-104695</a> - "Learn more" links return an error page <span style="border: 1px solid green; padding: 2px;">CLOSED</span></p> <p> <a href="#">SPL-103719</a> - There is no error message when repositoryLocation is used in Apps <span style="border: 1px solid green; padding: 2px;">UNTRIAGED</span></p>

- ▶ Heavy manual work
- ▶ Stale information
- ▶ Not scalable

# Test Triage System

## Solution

- ▶ ARTs (Automation Result Triage system)



# ARTs Highlights

## Overview

✓ ORANGESWIRL-CHROME-UCP (2018-10-01)



76 / 23684 76 ↑

Failures/Total Test Cases

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



80 / 10244

Failures/Total Test Cases

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



75 / 21715

Failures/Total Test Cases

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

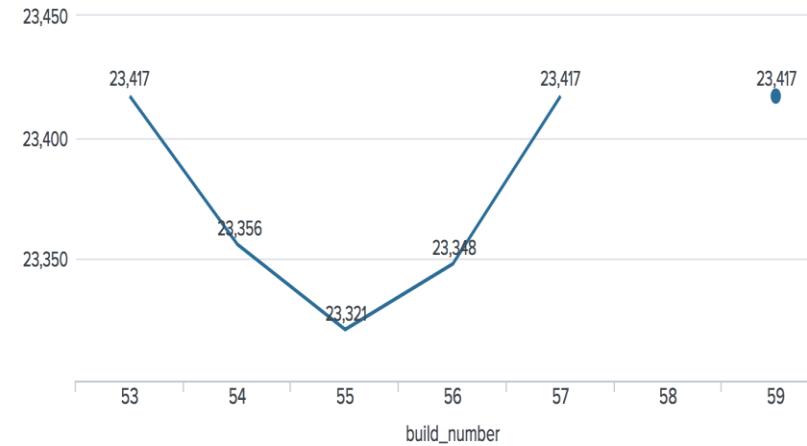
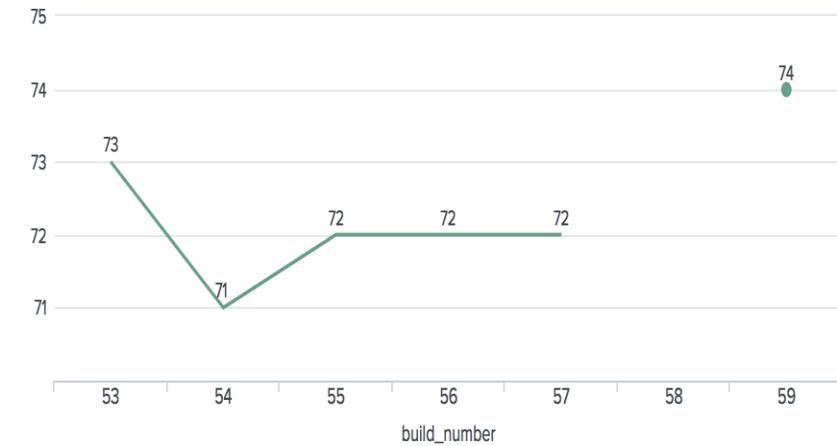
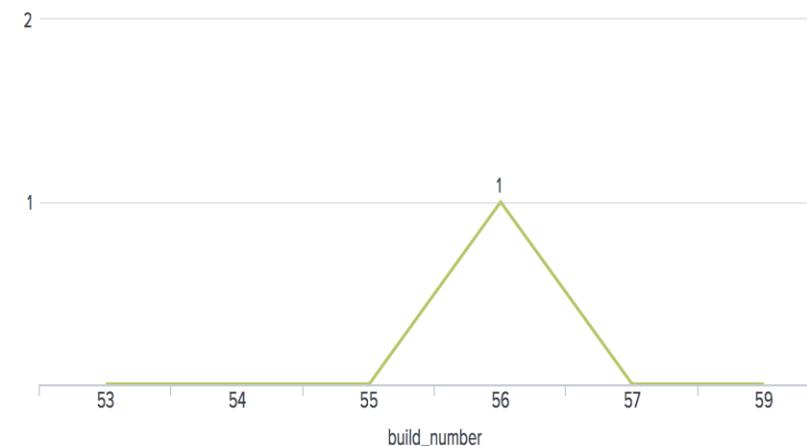
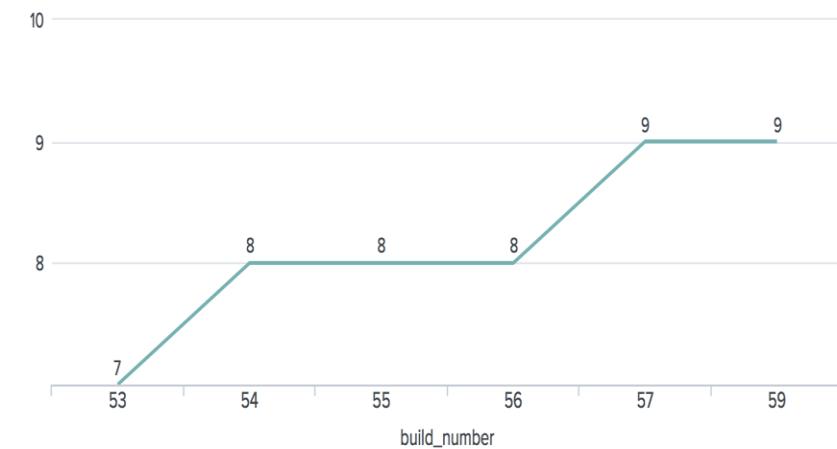


80 / 9224

Failures/Total Test Cases

# ARTs Highlights

## Trend

**TOTAL TESTS TREND****TOTAL FAILURES TREND****INFRASTRUCTURE (SETUP/START/RESTART/CONNECTION)  
FAILURES TREND****TEST FAILURES TREND**

# ARTs Highlights

## Triage

### Edit Triage Notes

Existing

\_fixed\_by\_zhe\_IE\_9\_26

86

start failures Connection

Intermittency: 0.09

Severity: 1



[notes:\_fixed\_by\_zhe\_IE\_9\_26]==>dashboards.test  
 [notes:\_fixed\_by\_zhe\_IE\_9\_26]==>dashboards.test

[notes:\_fixed\_by\_zhe\_IE\_9\_26]==>dashboards.test\_creating\_dashboards

J	R	Feature	Failed	Triaging	Untriaged	Failure type
⌚	⌚	client_distributed_clustering_master_bucket	3(0)	0	0	IndexError
⌚	⌚	client_distributed_forwarder_gdi_data_input	2(0)	0	0	AssertionError

# ARTs Highlights

## Branch Compare

**Branch Comparison**

Last 7 days    splunk     Show Job Info     Show Results

Summary Table Fields

Feature	Total	Added	Removed	Fixed
Failed	Existing	Regression	New	

Source (Jenkins master)    Job Name    Operating System

apps-jenkins	CoreFrontend/oran...	container
--------------	----------------------	-----------

Build\_No

36 (2018-09-06)
-----------------

vs. Source (Jenkins master)    vs. Job Name    vs. Operating System

apps-jenkins	CoreFrontend/dev...	container
--------------	---------------------	-----------

vs. Build\_No

313 (2018-09-06)
------------------

Job Name: CoreFrontend/orangeswirl-daily-triage-firefox-lin64

Total tests	Pass rate	Total failures	Skipped tests	Job time	Total tests	Pass rate	Total failures	Skipped tests	Job time
23350	99.90%	24	61	3h33m	23381	99.91%	22	61	4h22m

Product bug failures	Setup failures	Start/Restart failures	Connection failures	Flaky failures	Test failures	Product bug failures	Setup failures	Start/Restart failures	Connection failures	Flaky failures	Test failures
17	0	0	0	3	4	17	0	0	0	1	4

Summary  Show Jobs with No Tests

Feature	Total	Added	Removed	Fixed	Failed	Existing	Regression	New
client_webdriver_simplexml_pdf	145(-1)	0	1	0	1(+1)	0	1	
client_distributed_clustering	32(-1)	0	1	0	0(0)	0	0	

# 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

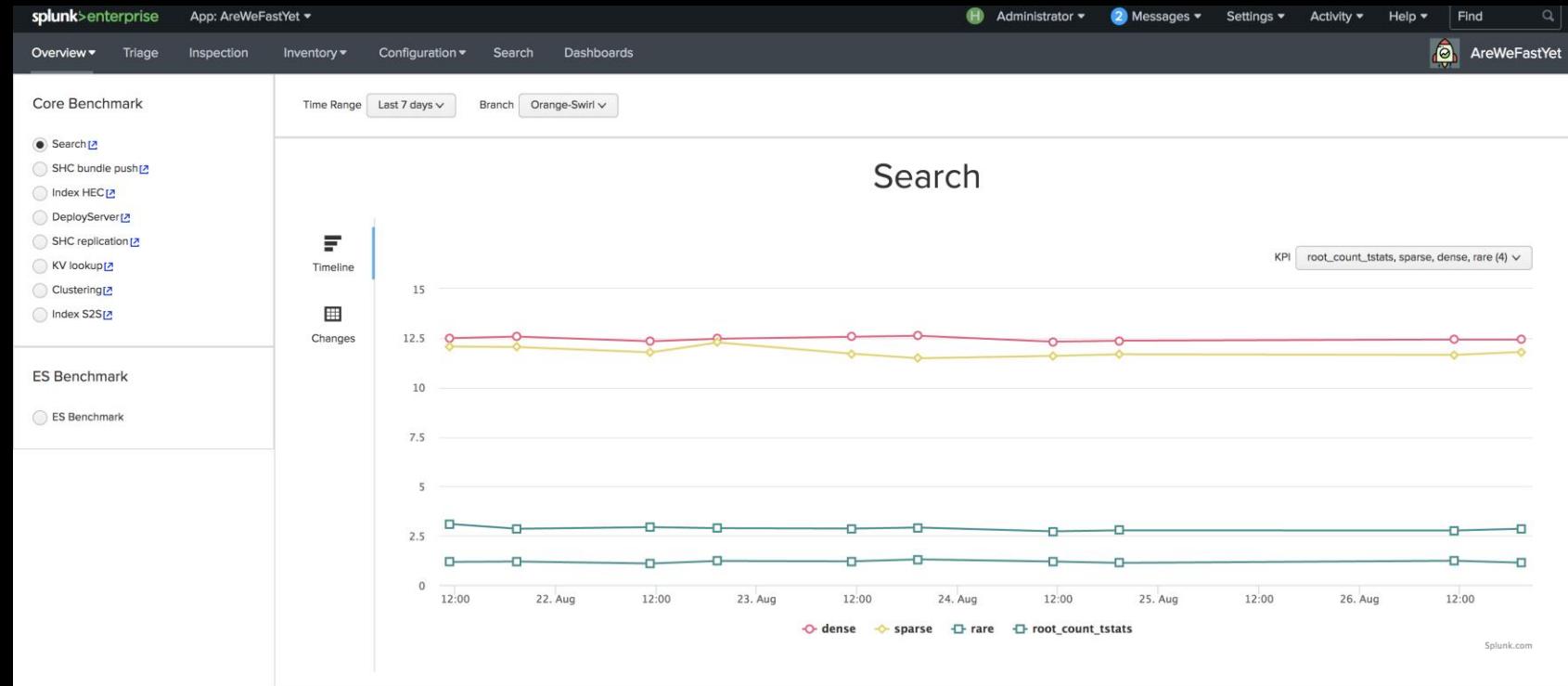
- ▶ AreWeFastYet 
    - One-stop-shop for scheduled and on-demand testing results
    - Interface for user defined rules to set KPIs and perform regression checks
    - Anomaly detection and alerting
    - Automatically generated visualization for test results and trending
    - Enable easy triage and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



# Performance Testing Results

## Highlights

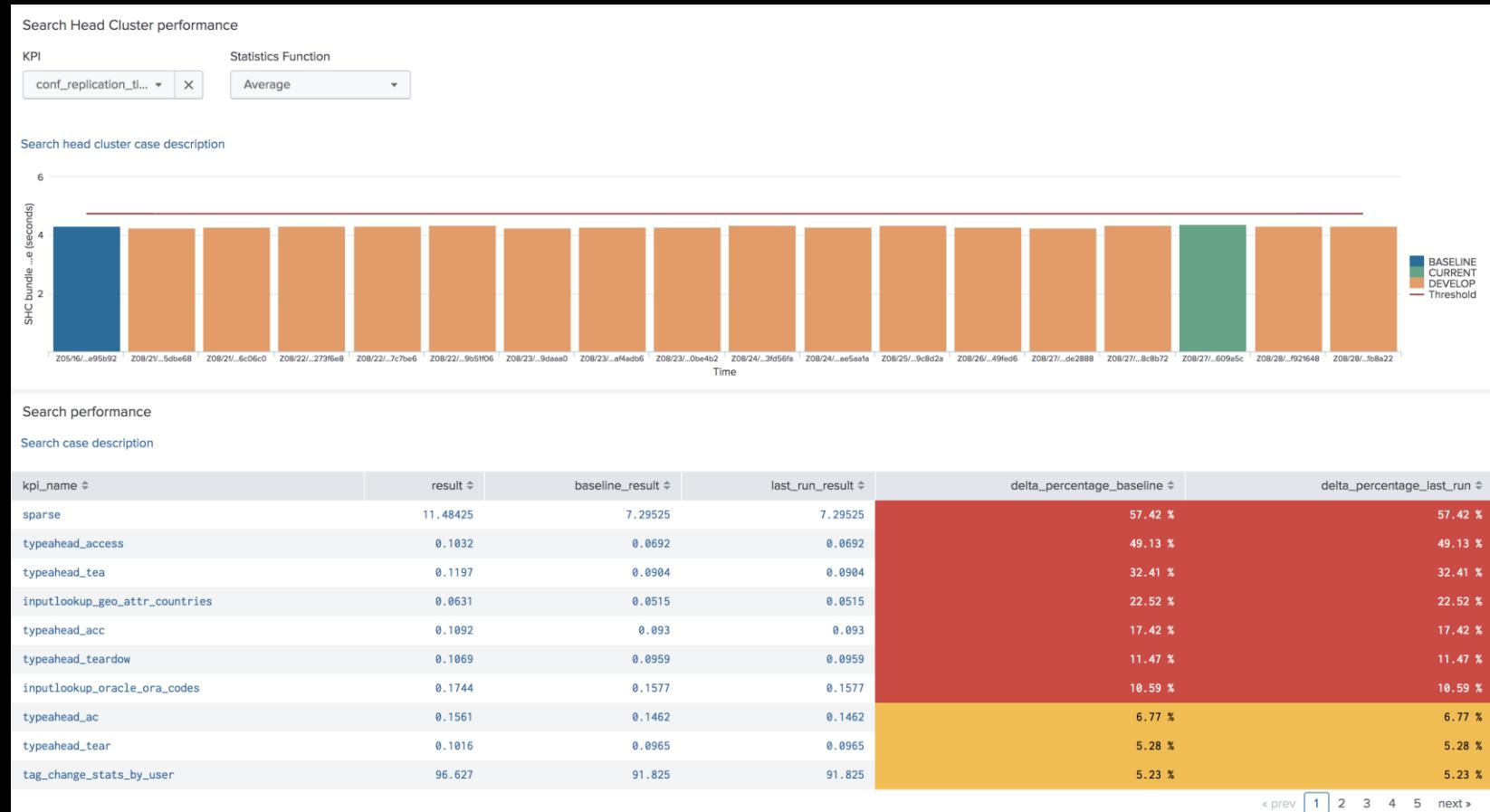
- ▶ Daily Benchmark Results Trending



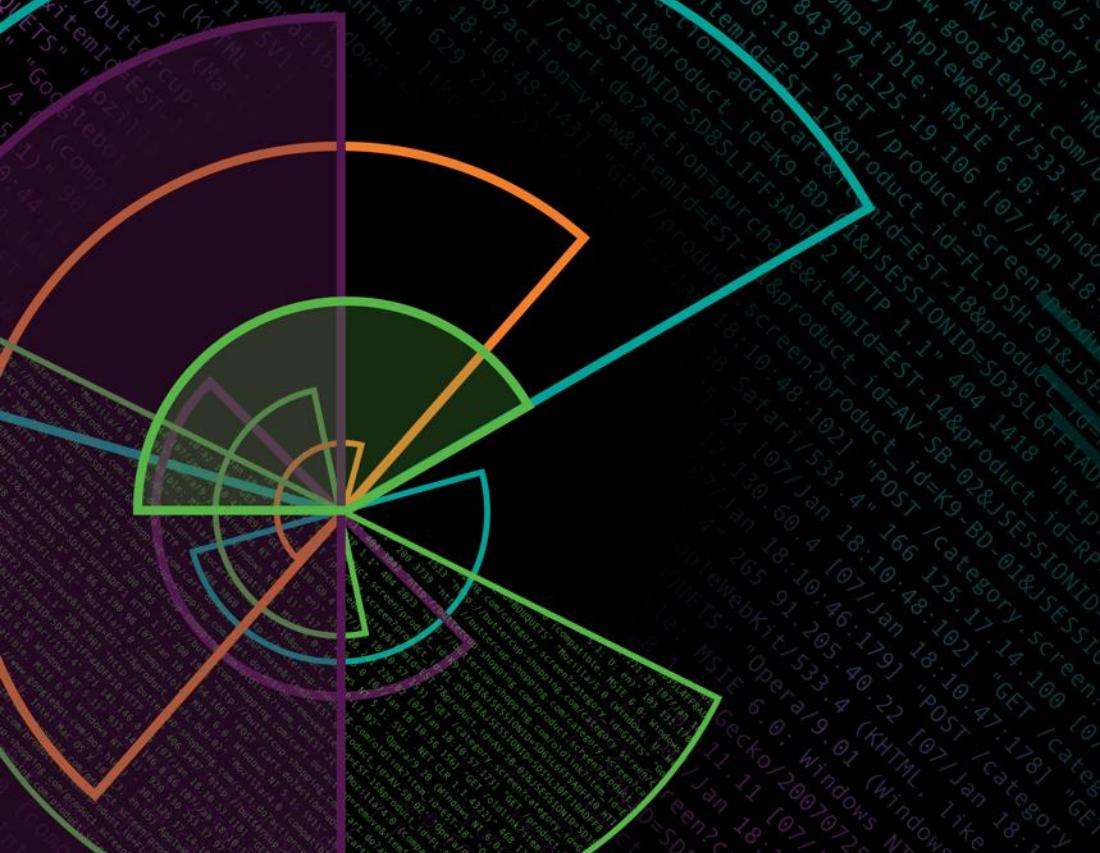
# Performance Testing Results

## Highlights

### Test Results Comparison and Triage



# Developer Portal

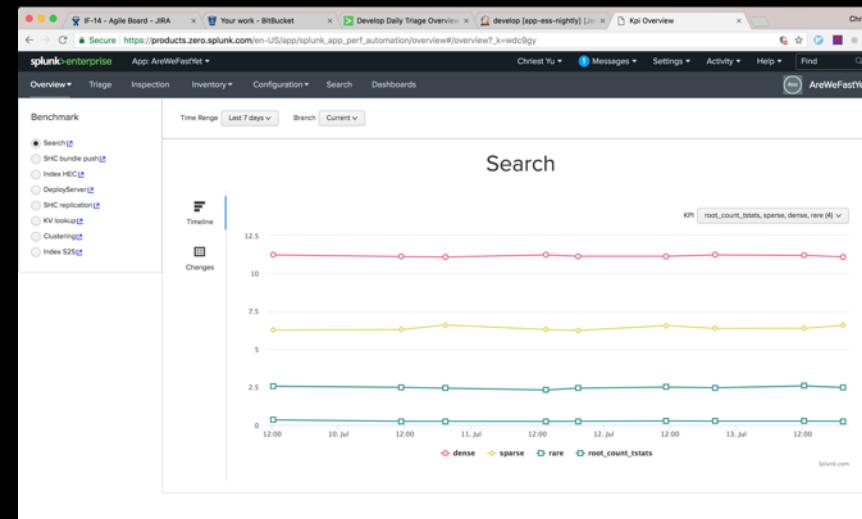


# Use Case - Developer Portal Problem

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

As a Splunker...

Morning



Afternoon

```

13 + if sys.platform == "win32":
14 +     import msvcrt
15 +     # Binary mode is required for persistent mode on Windows.
16 +     msvcrt.setmode(sys.stdin.fileno(), os.O_BINARY)
17 +     msvcrt.setmode(sys.stdout.fileno(), os.O_BINARY)
18 +     msvcrt.setmode(sys.stderr.fileno(), os.O_BINARY)
19 +
20 + from splunk.persistconn.application import PersistentServerConnectionApplication
21 + import os.path
22 +
23 + sys.path.insert(0, os.path.join(os.path.dirname(os.path.abspath(__file__)), "libs"))

```

# 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.

The screenshot displays the InfraHome developer portal interface. At the top, there's a navigation bar with links for Home, Task, Trigger, Links, and Auth. Below the navigation is a search bar with the placeholder "InfraHome".

**Sprint Status:** Sprint 27, 0% May 18 5:30 PM - 23 May 18 5:30 PM, 5 Workdays Left. It shows metrics: To Do: 1, In Progress: 2, Done: 1, Closed: 0. Below this is a list of tickets:

- INFRA-2990: As a Splunk Developer I would like to avoid plane crash
- FAST-3010: Docker service throw errors when initializing
- INFRA-3219: Frige: build a solid foundation for future usage
- INFRA-1319: Streaming voices avoiding crusade

**Pull Requests:** A list titled "Pull Request" showing two items under "Reviewing" and one under "Created By Me".

- Feature/INFRA-2345: Tools & Infrastructure/app-infra-home (Status: 0/0)
- Feature/INFRA-3367: Tools & Infrastructure/app-infra-home (Status: 1/0)
- Bugfix/INFRA-3181: Tools & Infrastructure/app-infra-home (Status: 0/2)

**Search Results:** A search results page for "jingboc" showing three JIRA issues:

Tagged	JIRA	JIRA Status	Repository	Branch	PR ID	PR Status	Latest Pipeline	Owner	Links
★	INFRA-6375	To Do						jingboc	Action
★	INFRA-7295	Closed	app-infra-home	bugfix/INFRA-7295	#52	MERGED		jingboc	Action
★	INFRA-7264	Closed	app-infra-home	bugfix/INFRA-7264	#50	MERGED		jingboc	Action

**Tickets I Care:** A list of tickets tagged with "jingboc":

Tagged	JIRA	JIRA Status	Repository	Branch	PR ID	PR Status	Latest Pipeline	Owner	Links
★	INFRA-6375	To Do						jingboc	Action
★	INFRA-7295	Closed	app-infra-home	bugfix/INFRA-7295	#52	MERGED		jingboc	Action
★	INFRA-7264	Closed	app-infra-home	bugfix/INFRA-7264	#50	MERGED		jingboc	Action

- ▶ 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

# 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

The screenshot displays the Splunk Developer Portal interface across three main sections:

- Home Page:** Shows a "Personal" dashboard with sections for "Sprint 27" (INFRA-S27), "5 Workdays Left", and "To Do". It lists four tickets: INFRA-2990, FAST-3010, INFRA-3219, and INFRA-1319, each with a brief summary.
- Task Page:** Titled "Pull Request", it shows a list of pull requests:
  - Reviewing: Feature/INFRA-2345 (0 green, 0 orange)
  - Feature/INFRA-3367 (1 green, 0 orange)
  - Created By Me: Bugfix/INFRA-3181 (0 green, 2 orange)
- Notification:** Shows two notifications:
  - DEVELOP-IE11-WIN2016 (2018-05-17) with a Windows logo icon.
  - DEVELOP-REGRESSION-WIN2012 (Core Backend) (2018-05-16) with a Windows logo icon.

# 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.



# EDDIE SHAFAQ

Splunk Infrastructure



# Our Speaker

# Splunk Infrastructure

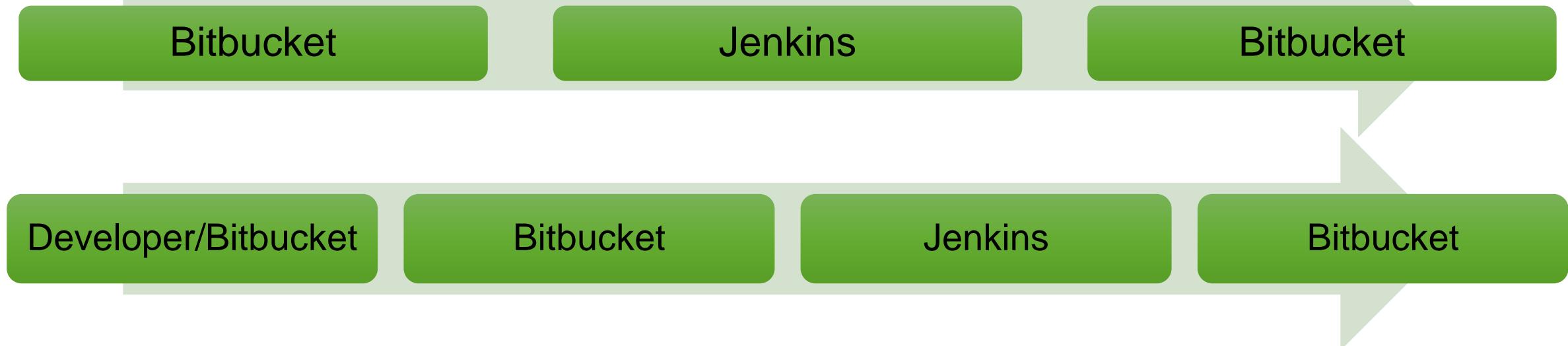
First joined Splunk in August 2011 as a Systems Administrator. Aided in expanding engineering support in "exotic operating system" (AIX, HPUX, S390X and PowerLinux). Served as a member of release engineering to address operational and infrastructure support for products team. Currently serving an infrastructure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

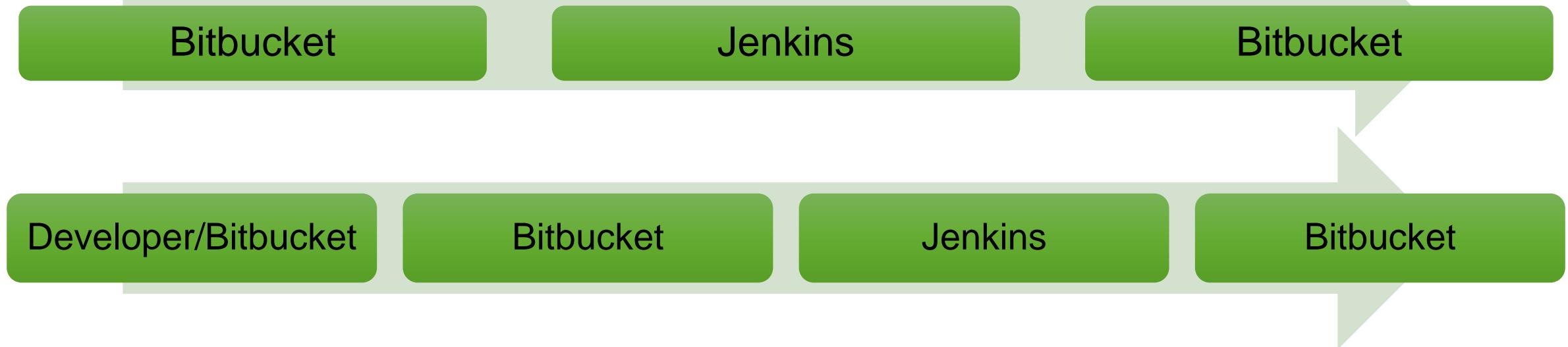


The system had **15 dedicated Linux agents** to perform continuous integration testing

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

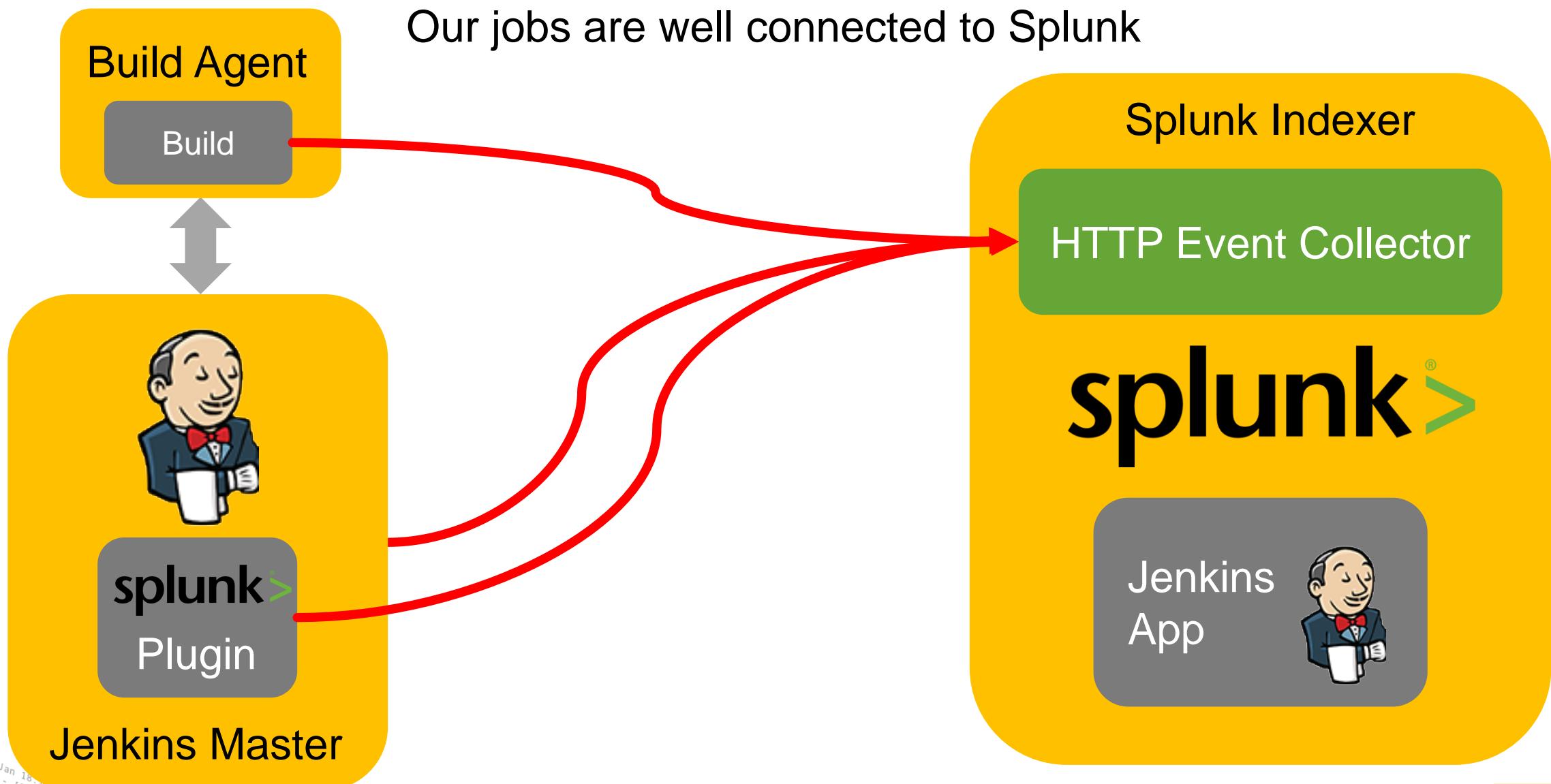
- Build time – how long it takes to build the Splunk executables
- Test time – how long it takes to perform the required set of tests
- Queue time – how long before the test actually started to run
- Notification – how long before developers know the test results

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

130.60.4 - - [07/Jan 18:10:57:153] "GET /category.screen?category\_id=GIFTS&JSESSIONID=SD15LAFF10ADFF10 HTTP 1.1" 404 72@ "http://buttercup-shopping.com/cart.do?action=view&itemId=EST-6&product\_id=F1-SW-01" "Opera/9.80 (Windows NT 5.1; SV1; .NET CLR 1.1.4322)" 128.241.220.82 - - [07/Jan 18:10:57:156] "GET /product.screen?product\_id=FL-DSH-01&JSESSIONID=SD55L7FF6ADFF9 HTTP 1.1" 404 332@ "http://buttercup-shopping.com/cart.do?action=view&itemId=EST-26&product\_id=S095L4FFAADDFF1 HTTP 1.1" 200 2423 "http://buttercup-shopping.com/cart.do?action=view&itemId=EST-18&product\_id=AU-CUP-18&JSESSIONID=SD10SLBFF2ADEF0F" Microsoft Internet Explorer/10.0.10240.3855 317.27.160.0.0 - - [07/Jan 18:10:56:156] "GET /oldlink?item\_id=EST-26&JSESSIONID=SD55L9FF1ADFF3 HTTP 1.1" 200 1318 "http://buttercup-shopping.com/cart.do?action=view&itemId=EST-18&product\_id=AU-CUP-18&JSESSIONID=SD10SLBFF2ADEF0F" Microsoft Internet Explorer/10.0.10240.3855 128.192.168.10 - - [07/Jan 18:10:55:187] "GET /cart.do?action=changeQuantity&itemId=EST-6&JSESSIONID=SD10SLBFF2ADEF0F" Microsoft Internet Explorer/10.0.10240.3855 128.192.168.10 - - [07/Jan 18:10:55:187] "GET /category.screen?category\_id=GIFTS&JSESSIONID=SD15LAFF10ADFF10 HTTP 1.1" 404 72@ "http://buttercup-shopping.com/cart.do?action=view&itemId=EST-6&product\_id=F1-SW-01" "Opera/9.80 (Windows NT 5.1; SV1; .NET CLR 1.1.4322)" 128.192.168.10 - - [07/Jan 18:10:55:188] "GET /category.screen?category\_id=SURPRISE&JSESSIONID=SD08SLBFF2ADEF0F" Microsoft Internet Explorer/10.0.10240.3855

# 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...

The image shows a laptop screen displaying the Splunk interface. The search bar contains the following query:

```
index="build_info" sourcetype=jenkins-ci (job_name="rcvr" OR (job_name="Linux_ut_pr" branch="current" (result=SUCCESS OR result=UNSTABLE))) | eval qtime=(queueTime/60000) | timechart span=1h count(eval(job_name="rcvr")) AS "Triggers", eval(round(max(qtime), 0)) AS "Max Queued Time"
```

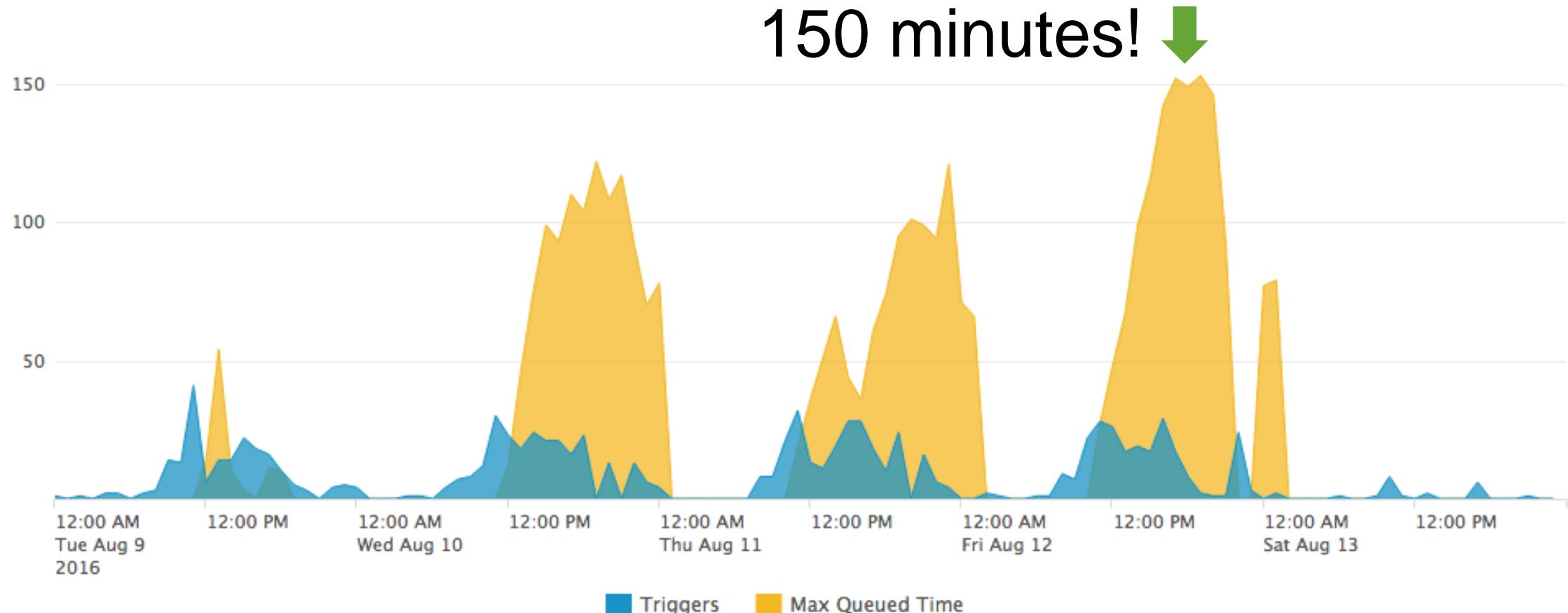
The search results show 134 events from September 19, 2017, to September 20, 2017. A histogram visualization displays the number of triggers over time, with a maximum value of 25. The event list table shows the following details for one event:

	i	Time	Event
< Hide Fields	i	9/20/17 12:52:48.735 PM	> { [-] branch: current build_url: https://re...om/job/Pull_Request_Tests/job/Linux_ut_pr/8215/ change: 9f0fdf07c5edebc2c399eb66d509720e013031b contribBuild: false duration: 8322426 failed_test_list: [ [+] ] job_name: Linux_ut_pr node: Fast_Linux-02 num_tests: 1113 num_tests_failed: 13 num_tests_passed: 1100 platform: Linux-x86_64 pull_request: 20108 queueTime: 30 result: UNSTABLE startTime: 1505928846 tested_by: timeoutCount: 1 }

The sidebar on the left lists selected fields and interesting fields. The selected fields include eventtype, host, index, job\_name, linecount, punct, source, sourcetype, splunk\_server, and tag. The interesting fields include startTime.

# Queue Times Before Optimization

**Peak load period had significant delays**



# 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; 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/(?<build number>.*)/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 < 5 | search sst > 0 | search pst < 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")
```

# The Search We Use to Analyze Jobs

# Extract the job number

```
| rex field=source  
"job/Pull Request Tests/job/Linux ut pr/(?<build nu  
mber>.*)/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 <  
5 | search sst > 0 | search pst < 25
```

# The Search We Use to Analyze Jobs

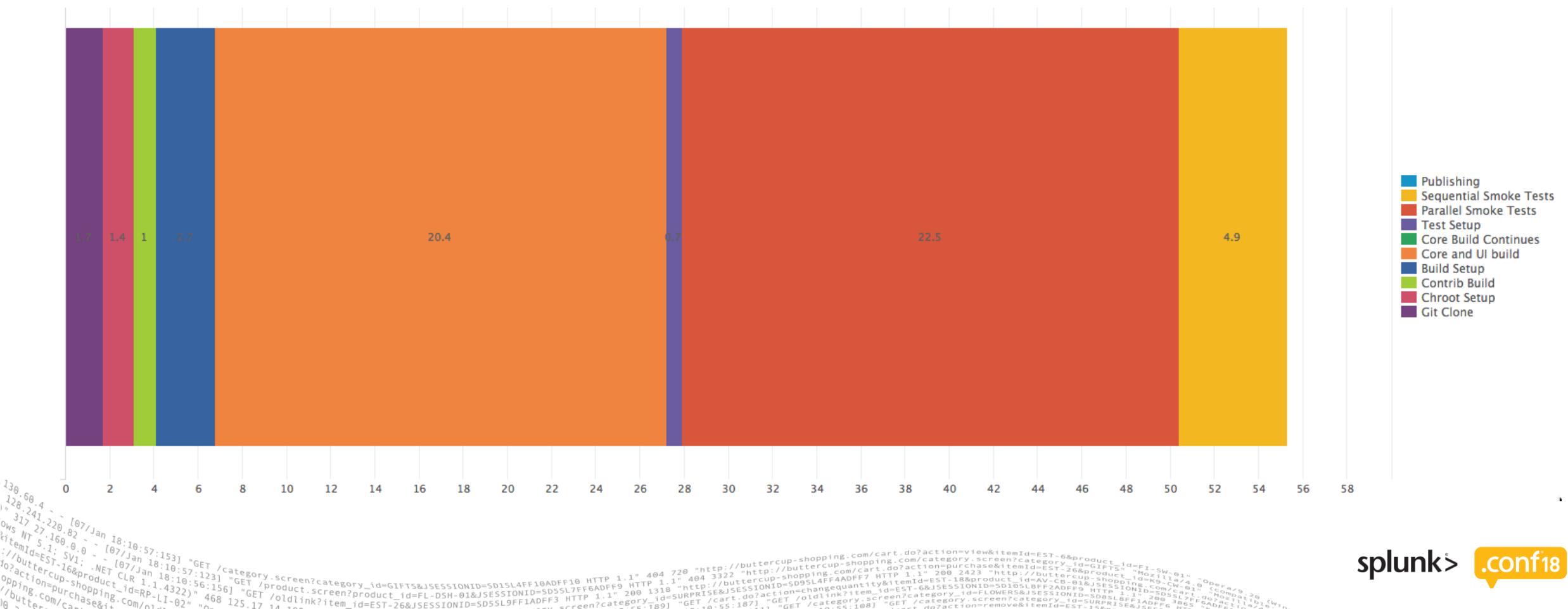
Build the final table with user friendly names for display

```
|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?

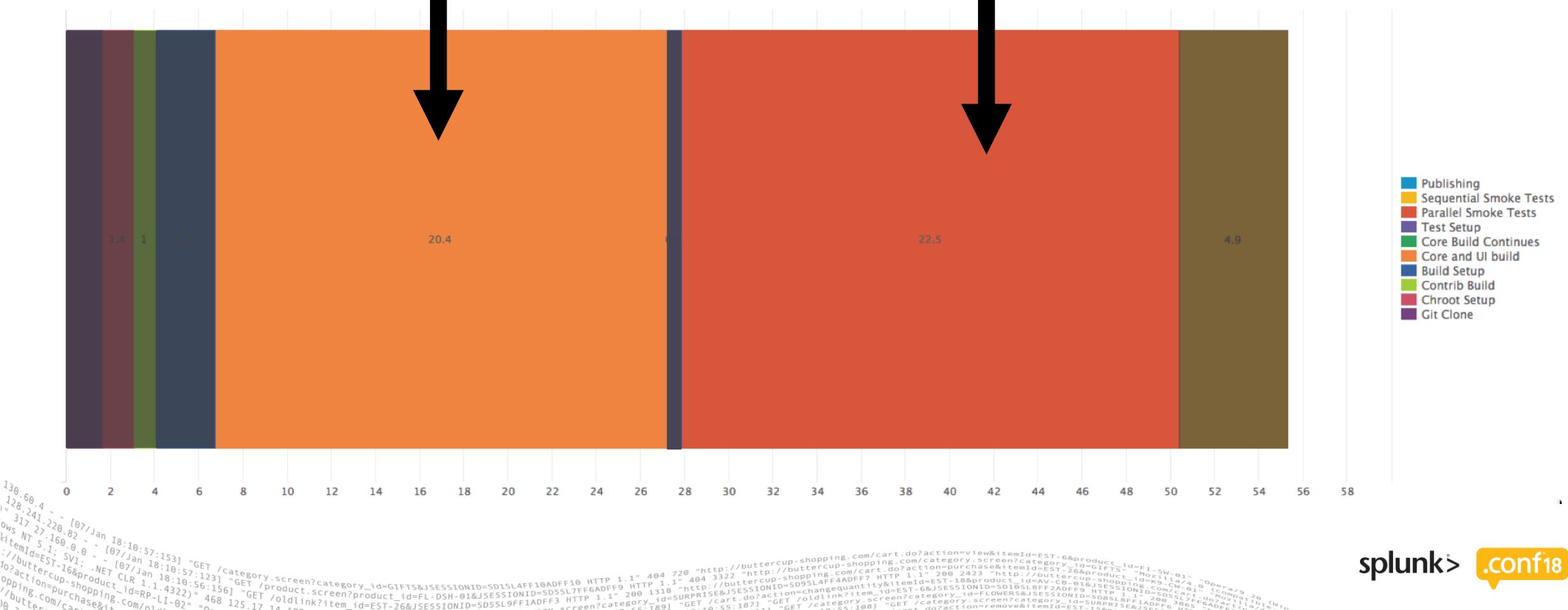
- We used the previous search to chart the time each step took



# 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)

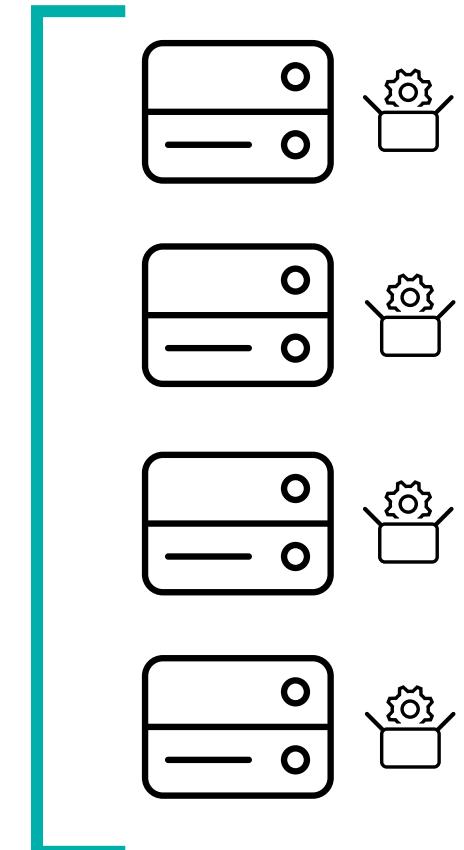


# Speeding up the Splunk Build

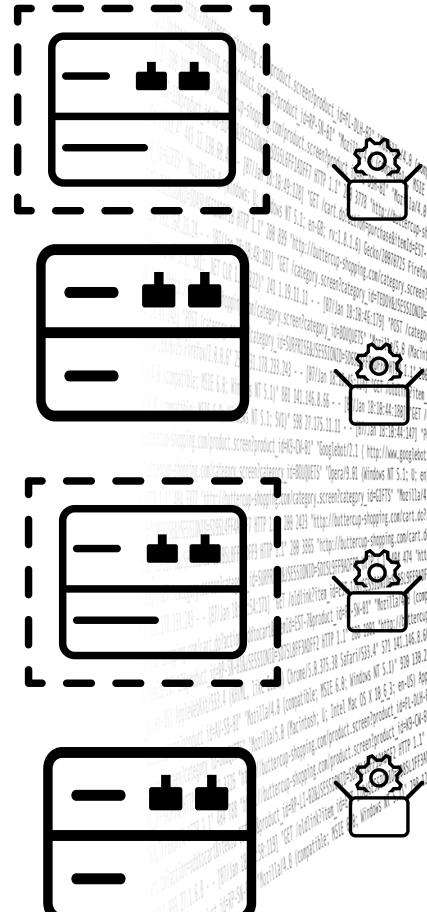
# Distcc Architecture

Physical Machines and VMs:

12 x 20 Core 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

A massive amount of log data is displayed diagonally across the slide, illustrating the high volume of requests and responses handled by the DistCC system during a build process. The logs show numerous HTTP requests for product screens, category screens, and cart actions, all originating from various clients and being processed by the DistCC server. The logs are filled with details like client IP addresses, user agent strings, session IDs, and specific URLs for products like "GIFTS&JSESSIONID=SD15LAFF10ADFF10", "NET CLR 1.1.4322", and "GET /oldlink?item\_id=EST-26&JSESSIONID=SD55L9FF1ADFF3". The sheer volume of logs emphasizes the efficiency and scale of the distributed build infrastructure.

# 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**

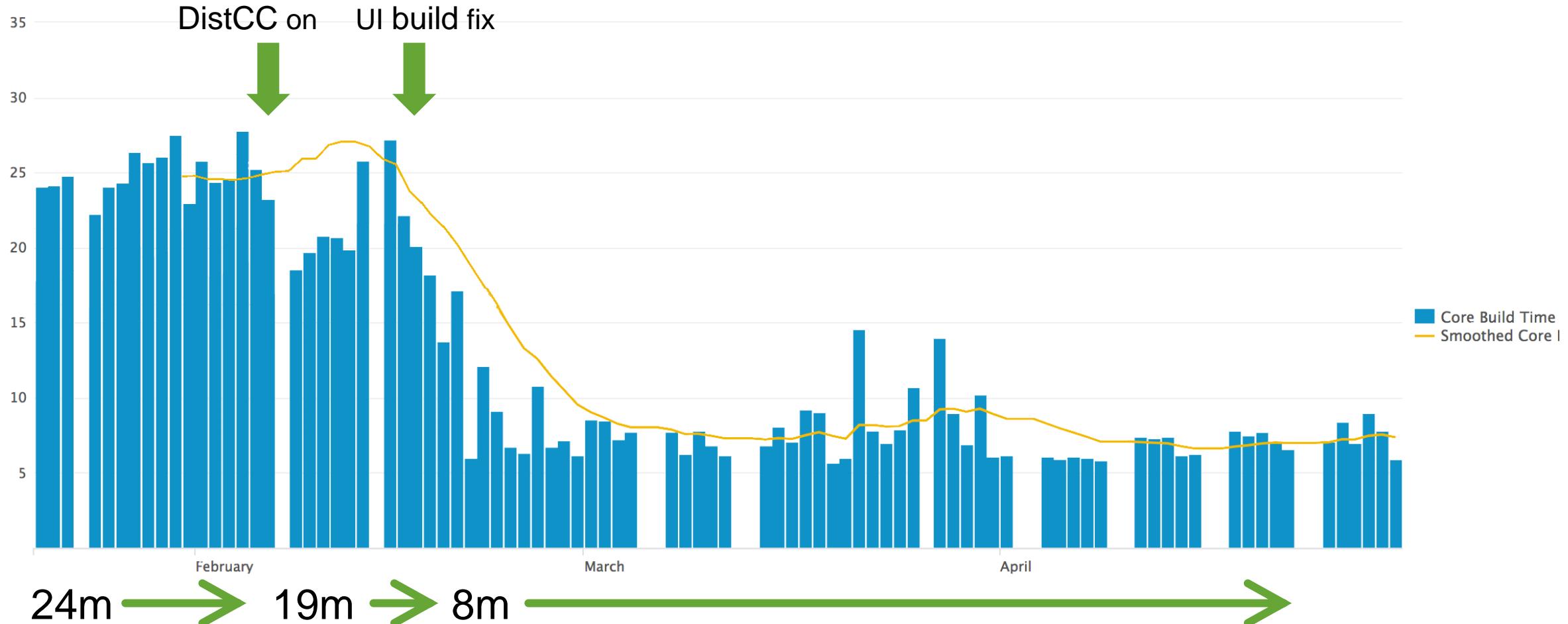
- Optimal 24 core VM agent

- 12 DistCC hosts

8m → 

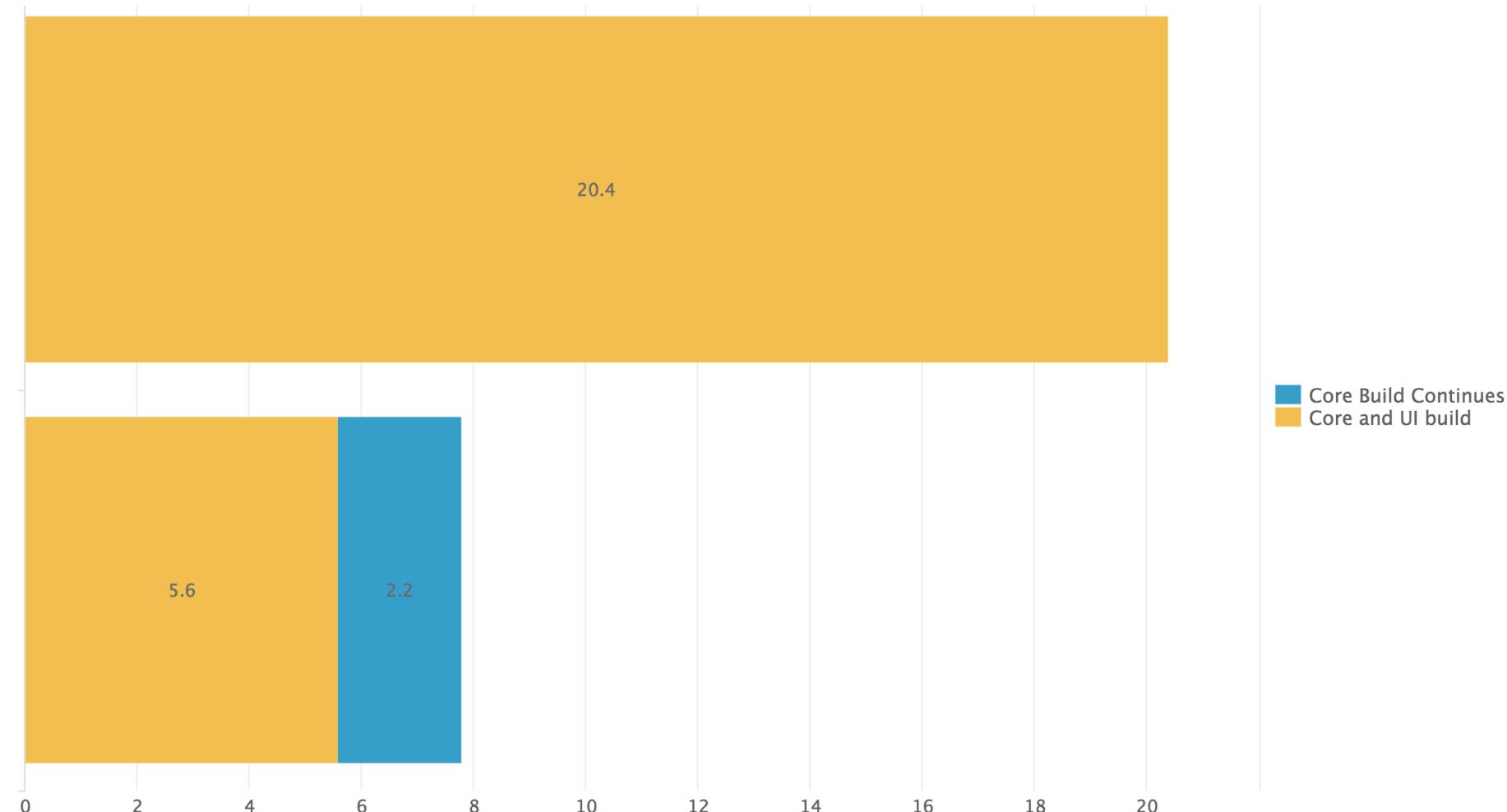
# Build Time Improvement - Results

Dramatic reduction in the overall build time



# Build Time Improvement - Results

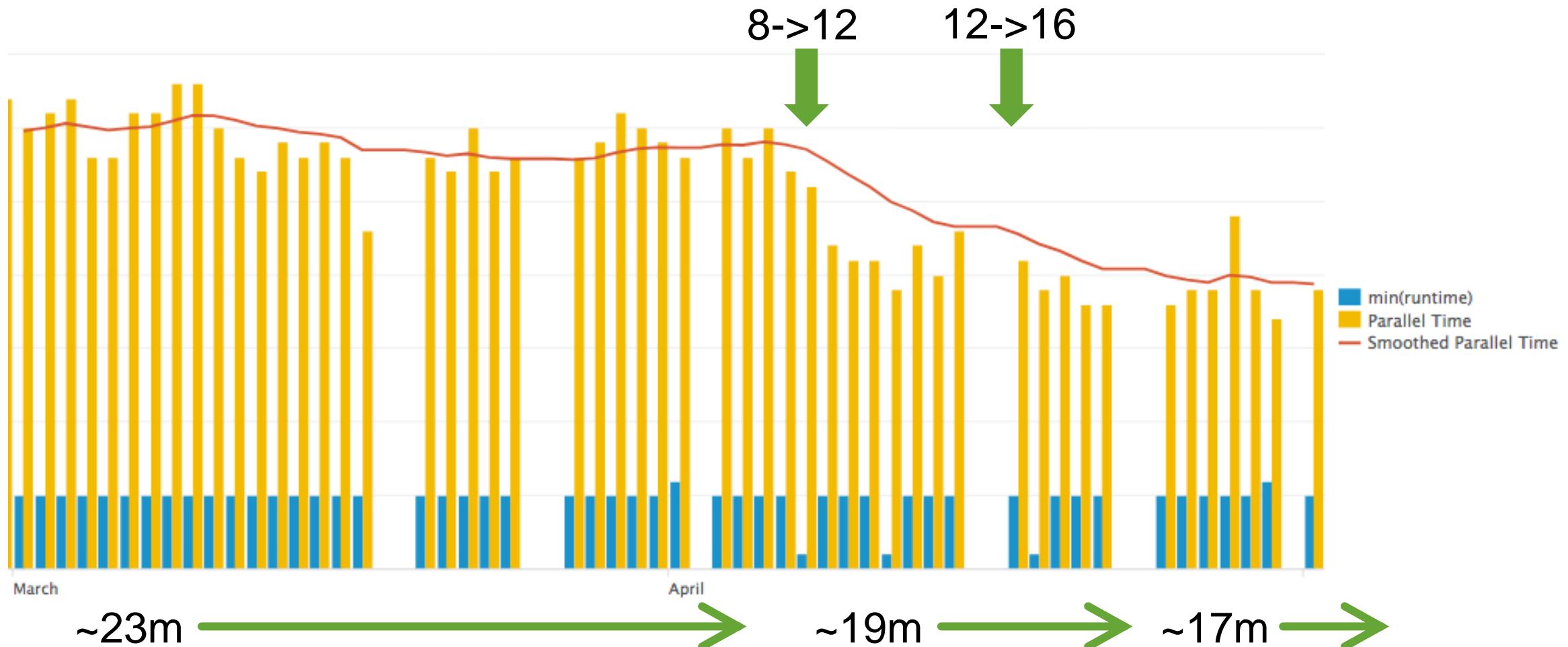
## Dramatic reduction in the core compile time



# Speeding up Testing

# Increasing Test Parallelization

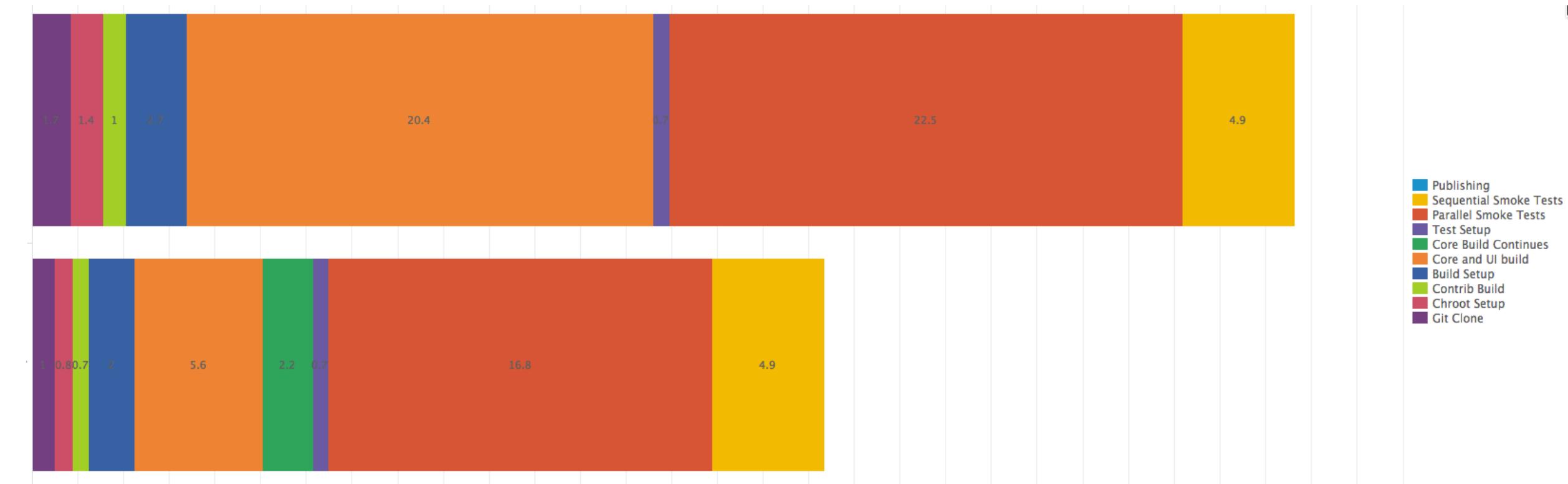
Improvement in the overall test time as parallel instances were increased



# 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



# Managing the Agents

# Our Analysis

## Breaking Down The Timing

# Reducing the Pull Request Queue time

## Analysis:

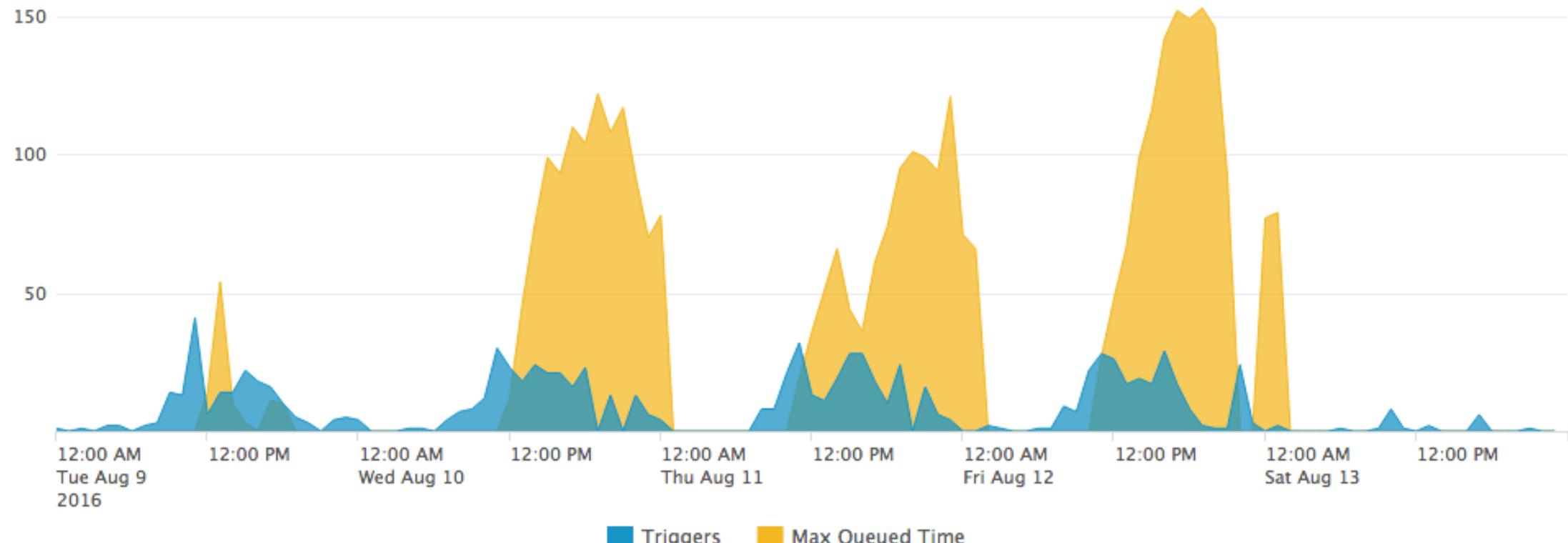
- Developers are only waiting for results from pull request tests
  - Triggers on commits to some branches and all pull requests

## 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

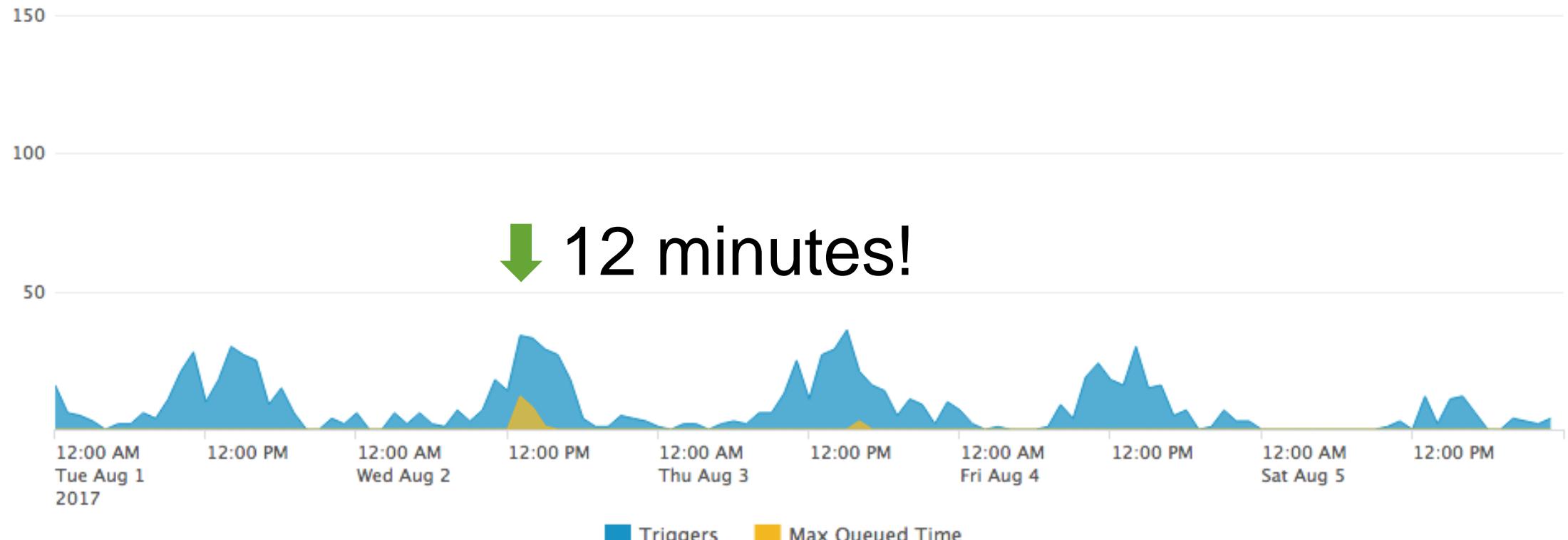
# Before Optimization

Peak load period → significant delays



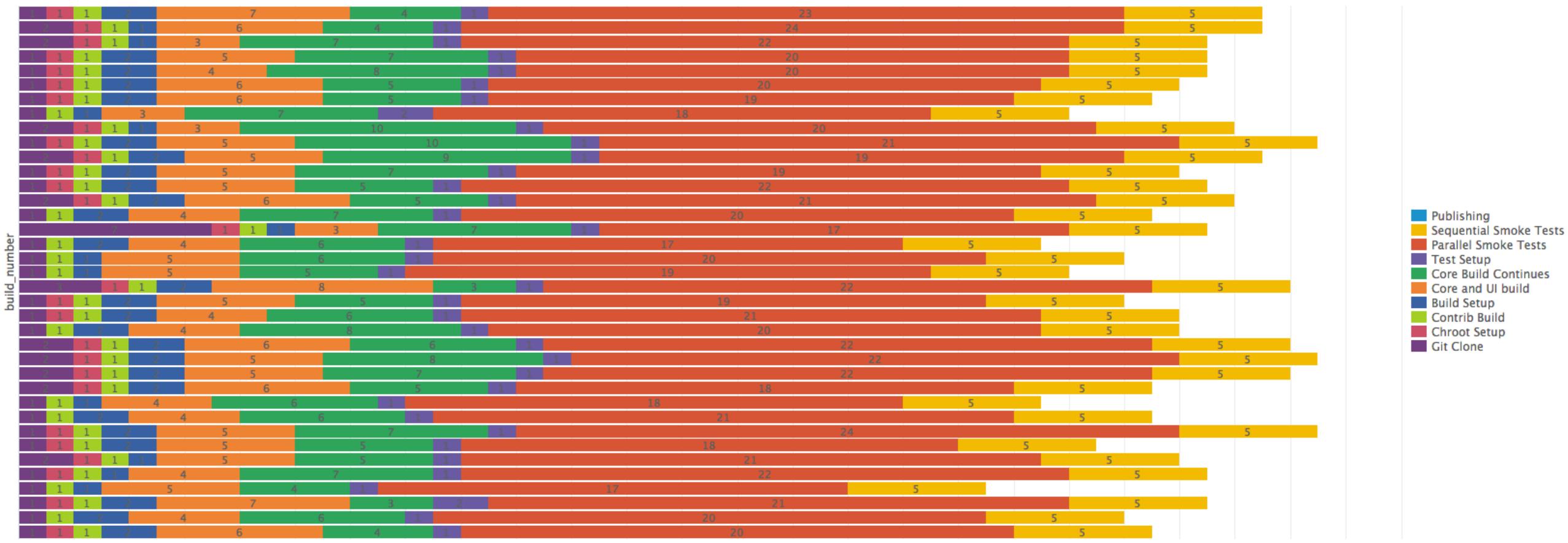
## After Optimization

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

---

