

Essential Component of a Continuous Integration Pipeline

Bill Houston, Eddie Shafaq



Forward-Looking Statements



During the course of this presentation, we may make forward-looking statements regarding future events or plans 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 may differ materially. The forward-looking statements made in the this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, it may not contain current or accurate information. We do not assume any obligation to update any forward-looking statements made herein.

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 functionalities described or to include any such feature or functionality in a future release.

Splunk, Splunk>, Turn Data Into Doing, 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. © 2019 Splunk Inc. All rights reserved.

Our Speakers

Our Speakers



EDDIE SHAFaq

Splunk Infrastructure



BILL HOUSTON

Splunk Tooling

Our Speakers

Eddie Shafaq

Splunk Infrastructure

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 in an DevOps role around Engineering Effectiveness and Infrastructure Engineering services.

Bill Houston

Splunk Tooling

Bill started his career as an analog hardware engineer designing professional recording equipment. Currently he is a senior release engineer at Splunk working on improvements to the Jenkins CI systems. Prior to Splunk he spent 16 years at Adobe working in various roles; the last four were spent using Jenkins to build and test Adobe Flash.

Agenda

Introduction

- The Challenge - Give Devs the results of their CI faster
- Overview - Splunk's CI environment
 - Splunk/Jenkins Plug-in
- Investigation - Using Splunk
 - Using Splunk - Collecting Build Data
 - Build Parallelization
 - DistCC, PIGZ
 - Test Parallelization
 - Multiple test instances
- Notifying Developers
- Jenkins App
- Monitoring Essential Service

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

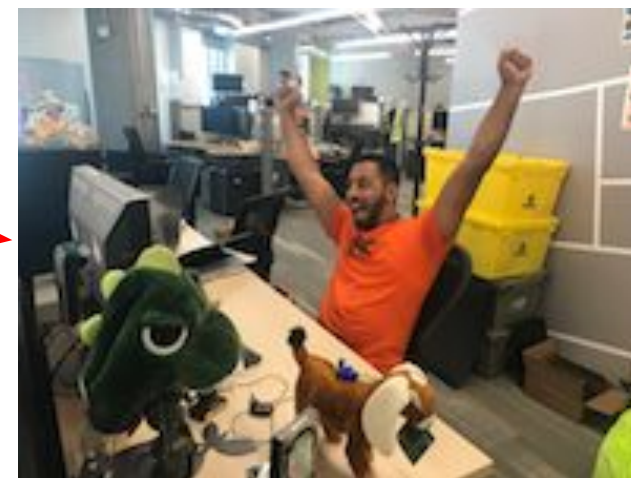
Engineering Productivity

- Test results
- Open stories/tasks/bugs

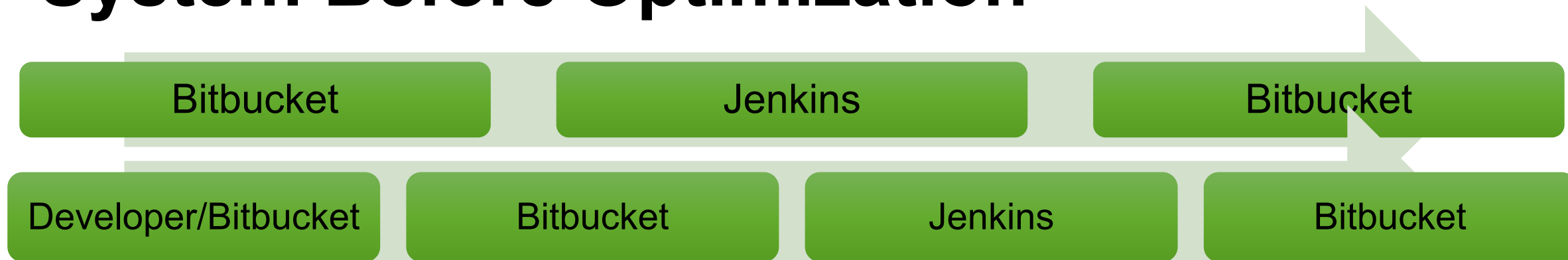
The Challenge

Improve Engineering Productivity

- Our contribution: Get CI test results to developers faster



System Before Optimization



The system has **30 dedicated Linux Docker agents** to perform continuous integration testing

Each job ran for approximately **90 minutes**, performing a build of Splunk and running a set of validation tests

That meant it could perform an approximate average of **20 jobs per hour**

System Before Optimization

If more than **20 triggers** were received in a **one hour** period the excess **triggers were queued** waiting for a Docker 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 their work



Our Analysis

Our Analysis

Understanding the Situation

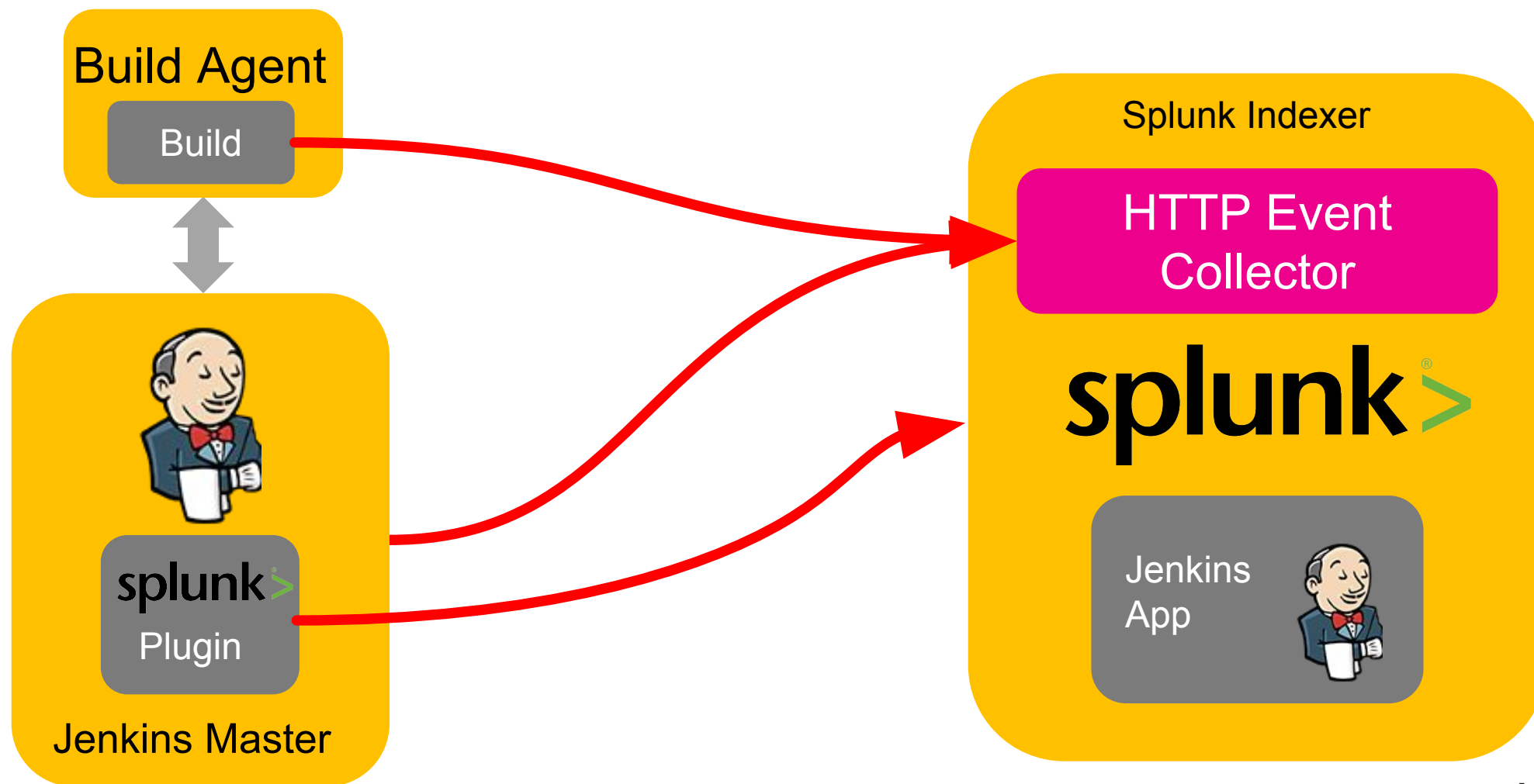
There were four factors that affected the delay developers experienced while waiting for test results

- 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 each phase of the builds start to run
- Notification – how long before developers know the test results

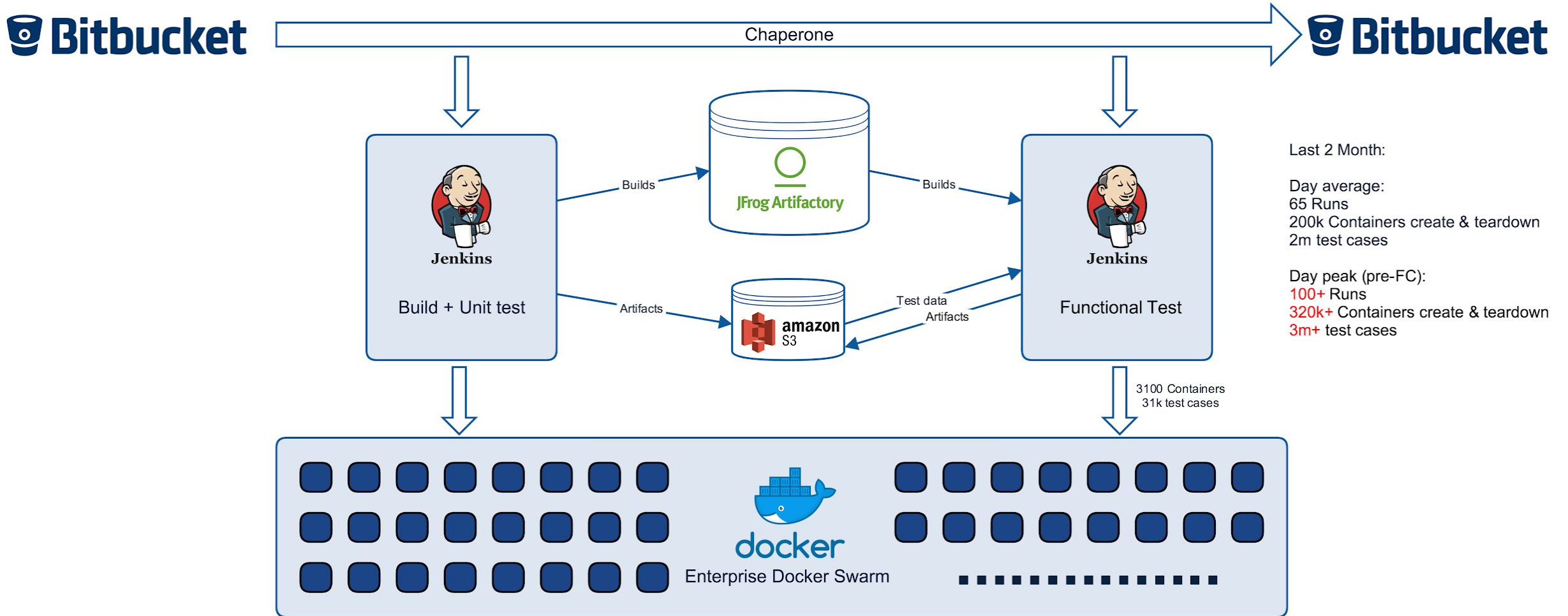
We need to quantify each of those factors and determine what we could do to mitigate their effects on the overall time

How We Collected The Data

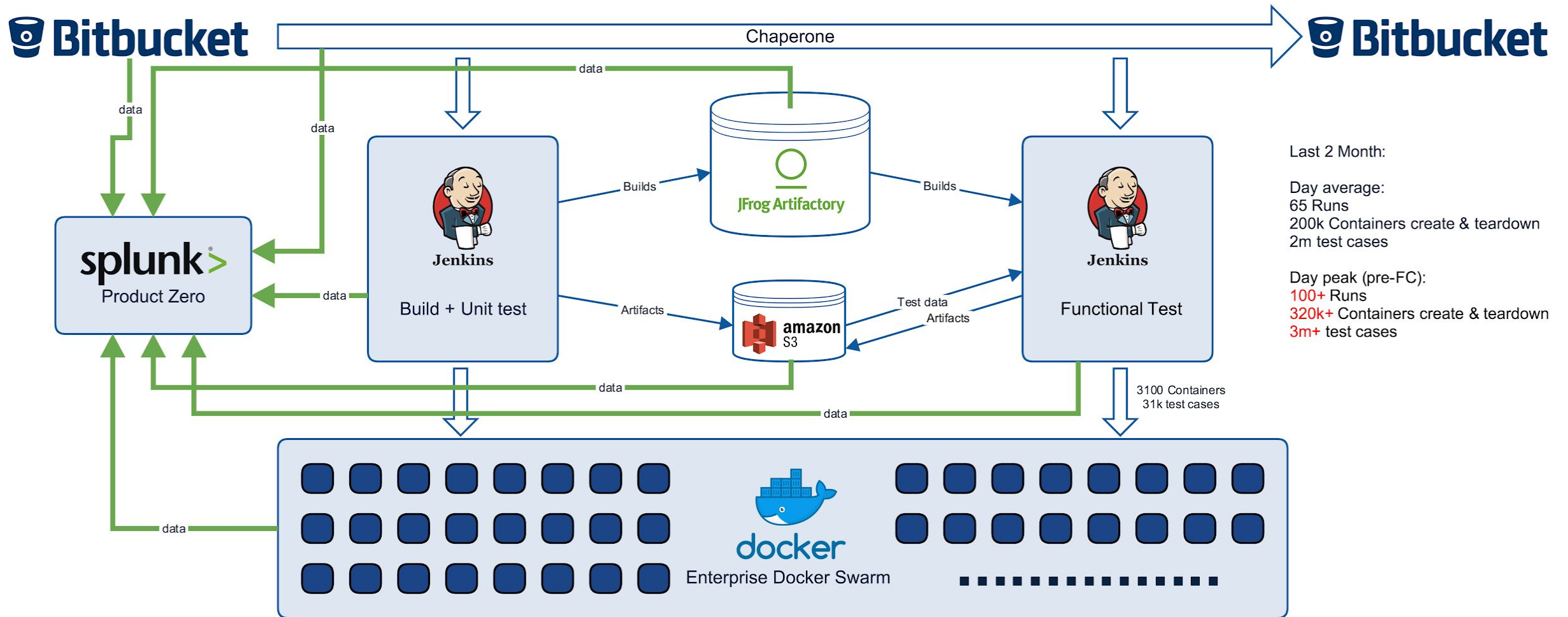
Our jobs are well connected to Splunk



Internal CI Infrastructure - Overview



highlevel



The Splunk Plugin for Jenkins

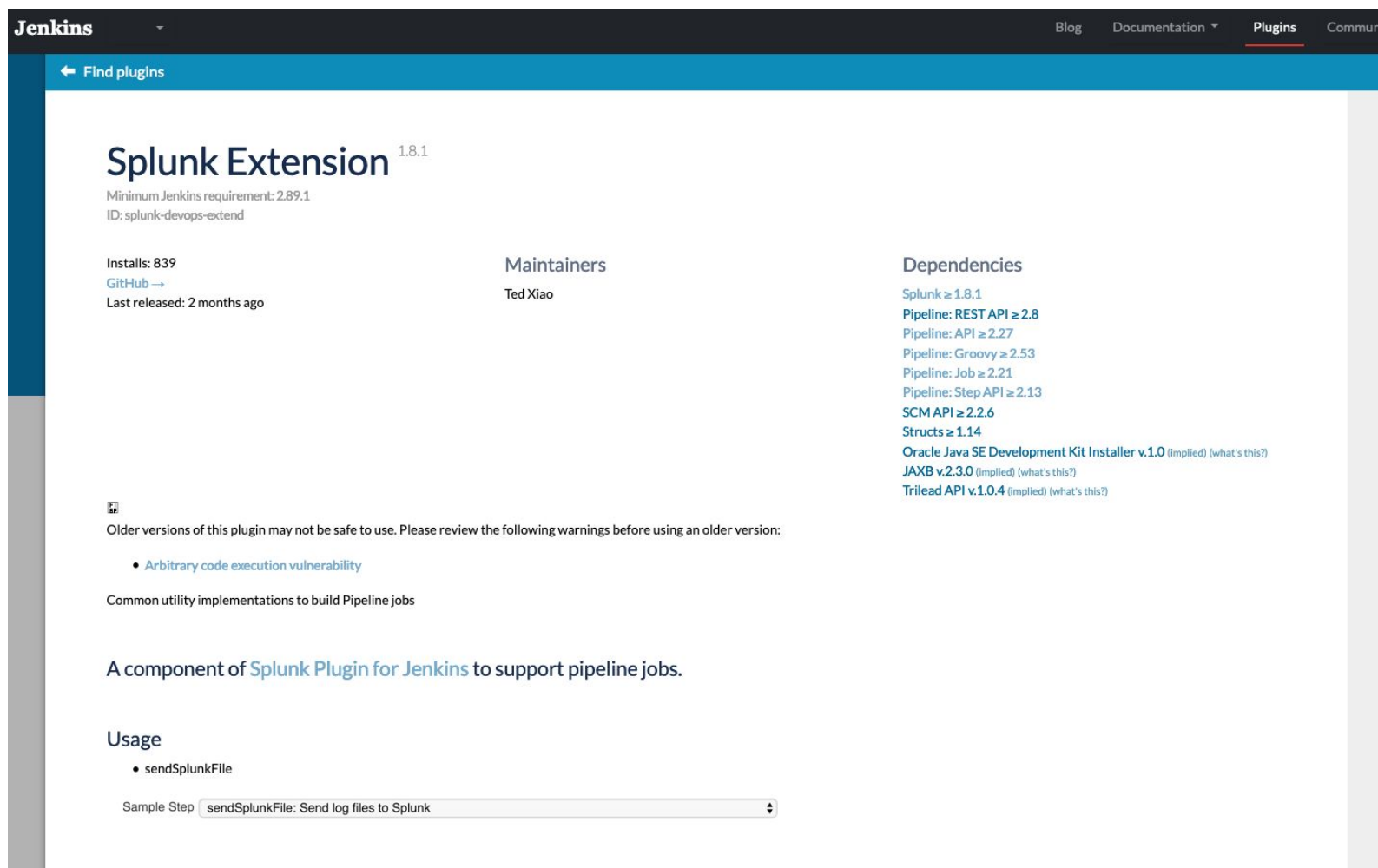
Makes it Easy to Send **Your** CI/CD Data to Splunk

The screenshot shows the Jenkins web interface with the 'Plugins' tab selected. The 'Find plugins' search bar is at the top. The 'Splunk' plugin (version 1.8.1) is displayed. It includes details such as 'Minimum Jenkins requirement: 2.60.3', 'ID: splunk-devops', 'Installs: 2121', 'GitHub →', and 'Last released: 2 months ago'. The maintainers listed are Danielle Jenkins, Kashyap Jotwani, and Ted Xiao. A list of dependencies is provided, including Script Security, Clover, Cobertura, Cucumber, JaCoCo, JUnit, TestNG, Command Agent Launcher, Oracle Java SE Development Kit Installer, JAXB, and Trilead API. A security warning section states that older versions may not be safe to use and lists 'Sandbox Bypass' and 'Arbitrary code execution vulnerability in rare circumstances'. The Splunk logo is shown, followed by a description of the plugin's functionality. The right sidebar contains an 'Archives' section with a line graph showing install trends from October to September, and a 'Links' section with links to GitHub, Javadoc, Labels, Build notifiers, Build reports, and Clean-up actions. At the bottom of the sidebar, there is a section for 'Previous Security Warnings' which repeats the vulnerabilities mentioned in the main content.

<https://wiki.jenkins.io/display/JENKINS/Splunk+plugin+for+Jenkins>

The Splunk Extension Plugin for Jenkins

Makes it Easy to Send Your Pipeline Data to Splunk



The screenshot shows the Jenkins web interface with the 'Plugins' tab selected. The 'Find plugins' search bar is at the top. The 'Splunk Extension' plugin (version 1.8.1) is displayed. It includes details such as 'Minimum Jenkins requirement: 2.89.1', 'ID: splunk-devops-extend', 'Installs: 839', 'GitHub →', and 'Last released: 2 months ago'. The 'Maintainers' section lists 'Ted Xiao'. The 'Dependencies' section lists various requirements like 'Splunk ≥ 1.8.1', 'Pipeline: REST API ≥ 2.8', 'Pipeline: API ≥ 2.27', 'Pipeline: Groovy ≥ 2.53', 'Pipeline: Job ≥ 2.21', 'Pipeline: Step API ≥ 2.13', 'SCM API ≥ 2.2.6', 'Structs ≥ 1.14', 'Oracle Java SE Development Kit Installer v.1.0 (implied) (what's this?)', 'JAXB v.2.3.0 (implied) (what's this?)', and 'Trilead API v.1.0.4 (implied) (what's this?)'. A warning section states: 'Older versions of this plugin may not be safe to use. Please review the following warnings before using an older version: • Arbitrary code execution vulnerability'. Below this, it says 'Common utility implementations to build Pipeline jobs'. A description states: 'A component of [Splunk Plugin for Jenkins](#) to support pipeline jobs.' The 'Usage' section lists '• sendSplunkFile'. At the bottom, a 'Sample Step' dropdown shows 'sendSplunkFile: Send log files to Splunk'.

Jenkins Blog Documentation **Plugins** Communi

← Find plugins

Splunk Extension ^{1.8.1}

Minimum Jenkins requirement: 2.89.1
ID: splunk-devops-extend

Installs: 839
[GitHub →](#)
Last released: 2 months ago

Maintainers

Ted Xiao

Dependencies

Splunk ≥ 1.8.1
Pipeline: REST API ≥ 2.8
Pipeline: API ≥ 2.27
Pipeline: Groovy ≥ 2.53
Pipeline: Job ≥ 2.21
Pipeline: Step API ≥ 2.13
SCM API ≥ 2.2.6
Structs ≥ 1.14
Oracle Java SE Development Kit Installer v.1.0 (implied) (what's this?)
JAXB v.2.3.0 (implied) (what's this?)
Trilead API v.1.0.4 (implied) (what's this?)

Older versions of this plugin may not be safe to use. Please review the following warnings before using an older version:

- Arbitrary code execution vulnerability

Common utility implementations to build Pipeline jobs

A component of [Splunk Plugin for Jenkins](#) to support pipeline jobs.

Usage






- sendSplunkFile

Sample Step

The Splunk Plugin for Jenkins

Easy to install

Splunk for Jenkins Configuration

Enable	<input checked="" type="checkbox"/>	
Indexer hostname	<input type="text"/>	
HTTP Input Port	<input type="text"/>	
HTTP Input Token	<input type="text"/>	
SSL Enabled	<input checked="" type="checkbox"/>	
Jenkins Master Hostname	<input type="text"/>	

Test Connection

The Splunk Plugin for Jenkins

Easy to configure

Splunk for Jenkins Configuration

Enable	<input checked="" type="checkbox"/>	
Indexer hostname	<input type="text" value="splunkidx.example.com"/>	?
HTTP Input Port	<input type="text" value="8088"/>	?
HTTP Input Token	<input type="text" value="DF789112-B454-4991-ABD3-B42CF73B3458"/>	?
SSL Enabled	<input checked="" type="checkbox"/>	?
Jenkins Master Hostname	<input type="text" value="ci-server.example.com"/>	?

Splunk connection verified

Test Connection

The Splunk Plugin for Jenkins

Easy to customize

Data Source ?

Config item ?

Value

Delete

Data Source ☒ Build Event ?

Config item ?

Value

Add

- ✓ Build Event
- Build Report
- Console Log
- Jenkins Config
- Log File
- Queue Information
- Slave Information
- Default

The Splunk Plugin for Jenkins

Makes it Easy to Send Your CI/CD Data to Splunk

The screenshot shows the Jenkins Plugins page for the Splunk plugin. The page layout includes a top navigation bar with links to Blog, Documentation, Plugins (active), Community, Subprojects, About, English, and a Download button. Below the navigation bar is a search bar labeled 'Find plugins'. The main content area displays the details for the Splunk 1.8.1 plugin. On the left, it shows the plugin name, version, minimum Jenkins requirement (2.60.3), ID (splunk-devops), install count (2121), a link to the GitHub repository, and the last release date (2 months ago). In the center, it lists the maintainers: Danielle Jenkins, Kashyap Jotwani, and Ted Xiao. On the right, it lists dependencies: Script Security ≥ 1.6.1, Clover ≥ 4.7.1 (optional), Cobertura ≥ 1.9.8 (optional), Cucumber json test reporting ≥ 0.9.7 (optional), JaCoCo ≥ 2.1.0 (optional), JUnit ≥ 1.18 (optional), TestNG Results ≥ 1.14 (optional), Command Agent Launcher v.1.0 (implied) (what's this?), Oracle Java SE Development Kit Installer v.1.0 (implied) (what's this?), JAXB v.2.3.0 (implied) (what's this?), and Trilead API v.1.0.4 (implied) (what's this?). Below the dependencies, there is a warning section titled 'Older versions of this plugin may not be safe to use. Please review the following warnings before using an older version:' with two bullet points: 'Sandbox Bypass' and 'Arbitrary code execution vulnerability in rare circumstances'. The Splunk logo is displayed below the warning. A paragraph describes the plugin's functionality: 'Splunk plugin for Jenkins provides deep insights into your Jenkins master and slave infrastructure, job and build details such as console logs, status, artifacts, and an incredibly efficient way to analyze test results.' Another paragraph mentions that the plugin is used together with a 'Splunk App for Jenkins' for out-of-the-box dashboards and search capabilities. On the right side of the page, there is an 'Archives' section with a 'Get past versions' link and a line graph showing the number of installations over time from October to September. Below the graph are links to GitHub, Javadoc, and Labels. There are also links for 'Build notifiers', 'Build reports', and 'Clean-up actions'. A section titled 'Are you maintaining this plugin?' asks the user to visit the 'Jenkins Plugin Wiki' to edit the content. Finally, a 'Previous Security Warnings' section lists two warnings: 'Arbitrary code execution vulnerability in rare circumstances' (affects version 1.5.2 and earlier) and 'Sandbox Bypass' (affects version 1.7.4 and earlier).

Jenkins 1.8.1
Minimum Jenkins requirement: 2.60.3
ID: splunk-devops

Installs: 2121
GitHub →
Last released: 2 months ago

Maintainers
Danielle Jenkins
Kashyap Jotwani
Ted Xiao

Dependencies
Script Security ≥ 1.6.1
Clover ≥ 4.7.1 (optional)
Cobertura ≥ 1.9.8 (optional)
Cucumber json test reporting ≥ 0.9.7 (optional)
JaCoCo ≥ 2.1.0 (optional)
JUnit ≥ 1.18 (optional)
TestNG Results ≥ 1.14 (optional)
Command Agent Launcher v.1.0 (implied) (what's this?)
Oracle Java SE Development Kit Installer v.1.0 (implied) (what's this?)
JAXB v.2.3.0 (implied) (what's this?)
Trilead API v.1.0.4 (implied) (what's this?)

Older versions of this plugin may not be safe to use. Please review the following warnings before using an older version:

- Sandbox Bypass
- Arbitrary code execution vulnerability in rare circumstances

splunk

Splunk plugin for Jenkins provides deep insights into your Jenkins master and slave infrastructure, job and build details such as console logs, status, artifacts, and an incredibly efficient way to analyze test results.

The plugin is used together with a [Splunk App for Jenkins](#) that provides out-of-the-box dashboards and search capabilities to enable organizations to run a high performing Jenkins cluster and bring operational intelligence into the software development life cycle.

Archives
Get past versions

Links
[GitHub](#)
[Javadoc](#)
[Labels](#)
[Build notifiers](#)
[Build reports](#)
[Clean-up actions](#)

Are you maintaining this plugin?
Visit the [Jenkins Plugin Wiki](#) to edit this content.

Previous Security Warnings

- Arbitrary code execution vulnerability in rare circumstances
 - Affects version 1.5.2 and earlier
- Sandbox Bypass
 - Affects version 1.7.4 and earlier

The Splunk HTTP Event Collector

Simple to send custom data to your Splunk instance

In Bash

Format your data as a JSON string:

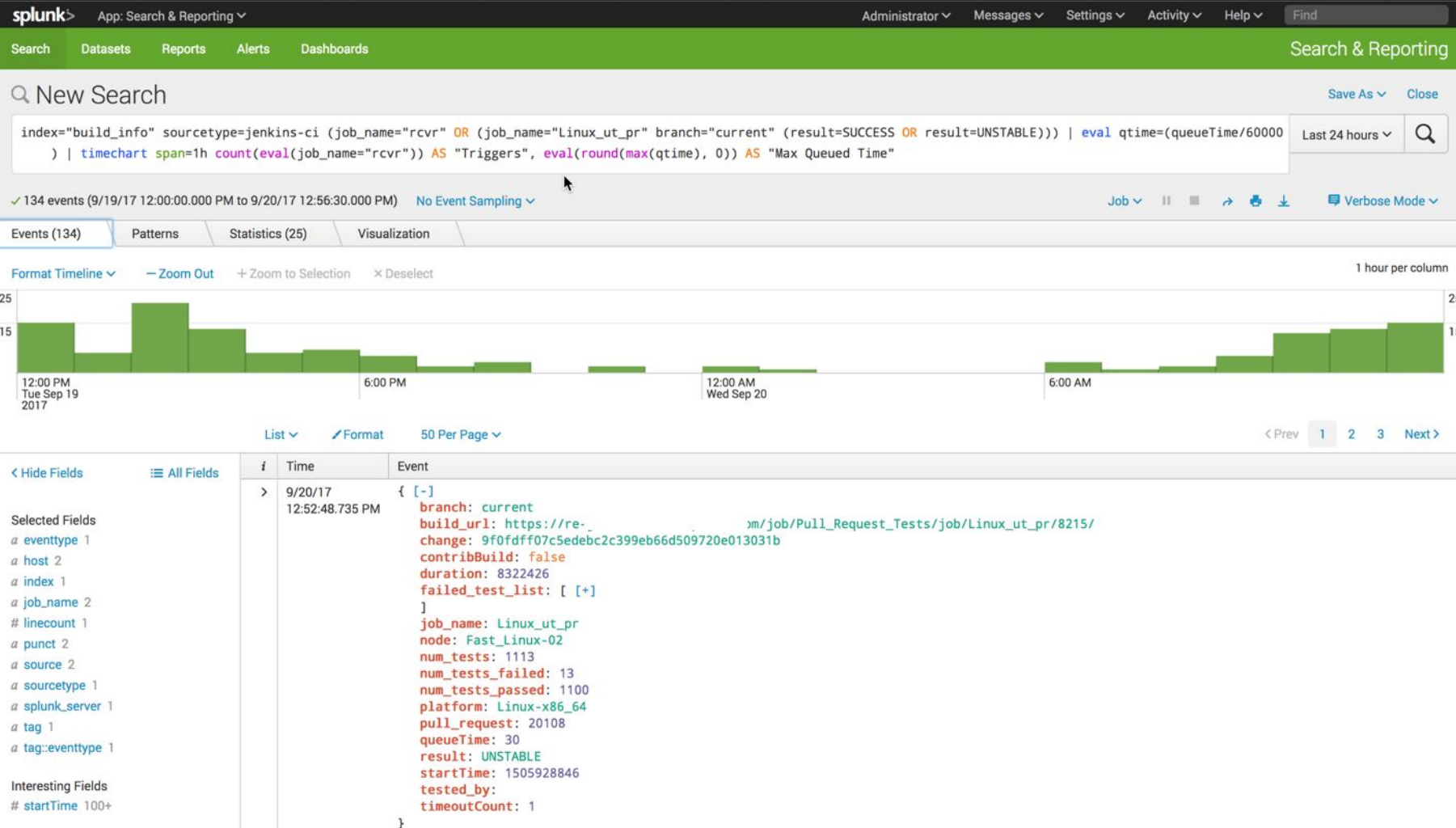
```
jsonData="{\"time\": 12345, \"index\": \"YourIndex\", \"sourcetype\": \"YourSourceType\",  
\"source\": \"YourSource\", \"event\": {\"YourFieldName\": \"SomeData\", more json formatted data  
goes here}}\""
```

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 old 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 NEW Search We Use to Analyze Jobs

Build the final table with user friendly names for display

```
index="jenkins_statistics" job_name="pr-coreblid"  
host="jenkins-core-delivery.splunkeng.com"  
build_number=* type=completed  
"metadata.BRANCH"=current | eval fields =  
mvzip('stages{}.name','stages{}.duration') |  
mvexpand fields | rex field=fields  
"(?<stagePlatName>[^,]+),(?<stageSeconds>[^,]+)" |  
rex field=stagePlatName  
"(?<plat>[^\n-]+)[\n-](?<stageName>.+)$" | eval  
stageMinutes=stageSeconds/60 | timechart span=1d  
eval(round(avg(stageMinutes), 2)) by stageName
```

The NEW Search We Use to Analyze Jobs

Build the final table with user friendly names for display

```
index="jenkins_statistics" job_name="pr-corebld"  
host="jenkins-core-delivery.splunkeng.com"  
build_number=* type=completed  
"metadata.BRANCH"=current
```

The NEW Search We Use to Analyze Jobs

Correctly associate each builds stage data

```
| eval fields =  
mvzip(mvzip('stages{}.name','stages{}.duration'),  
'stages{}.start_time')
```

```
2019-10-21 08:41:01.328
```

```
intel_linux_64_2.6-Sync,86.021,1571667908  
intel_linux_64_2.6-Build contrib,70.814,1571667994  
intel_linux_64_2.6-Build,996.183,1571668065  
intel_linux_64_2.6-Show Warnings,9.532,1571669061  
intel_linux_64_2.6-Package,1771.809,1571669071  
intel_linux_64_2.6-Publish,1006.698,1571670843  
intel_linux_64_2.6-Backend Fast Testing,606.131,1571671850  
intel_linux_64_2.6-Backend Unit Results,1.13,1571672456  
intel_linux_64_2.6-Postbuild,0.057,1571672457
```


The NEW Search We Use to Analyze Jobs

rexpand into stage specific fields with associated time stamps

| mvexpand fields

_time ↕	fields ↕
2019-10-21 08:41:01.328	intel_linux_64_2.6-Sync,86.021,1571667908
2019-10-21 08:41:01.328	intel_linux_64_2.6-Build contrib,70.814,1571667994
2019-10-21 08:41:01.328	intel_linux_64_2.6-Build,996.183,1571668065
2019-10-21 08:41:01.328	intel_linux_64_2.6-Show Warnings,9.532,1571669061
2019-10-21 08:41:01.328	intel_linux_64_2.6-Package,1771.809,1571669071
2019-10-21 08:41:01.328	intel_linux_64_2.6-Publish,1006.698,1571670843
2019-10-21 08:41:01.328	intel_linux_64_2.6-Backend Fast Testing,606.131,1571671850
2019-10-21 08:41:01.328	intel_linux_64_2.6-Backend Unit Results,1.13,1571672456
2019-10-21 08:41:01.328	intel_linux_64_2.6-Postbuild,0.057,1571672457

The NEW Search We Use to Analyze Jobs

Build the final table with user friendly names for display

```
|rex field=fields
```

```
"(?<stagePlatName>[^,]+),(?<stageSeconds>[^,]+)" |
```

```
rex field=stagePlatName
```

```
"(?<plat>[^\n-]+)[\n-](?<stageName>.+)$" | eval
```

```
stageMinutes=stageSeconds/60
```

The NEW Search We Use to Analyze Jobs

Build the final table with user friendly names for display

_time ↕	stageName ↕	stageMinutes ↕
2019-10-21 08:41:01.328	Sync	1.4337
2019-10-21 08:41:01.328	Build contrib	1.1802
2019-10-21 08:41:01.328	Build	16.6031
2019-10-21 08:41:01.328	Show Warnings	0.1589
2019-10-21 08:41:01.328	Package	29.53015
2019-10-21 08:41:01.328	Publish	16.77830
2019-10-21 08:41:01.328	Backend Fast Testing	10.1022
2019-10-21 08:41:01.328	Backend Unit Results	0.0188
2019-10-21 08:41:01.328	Postbuild	0.00095

The NEW Search We Use to Analyze Jobs

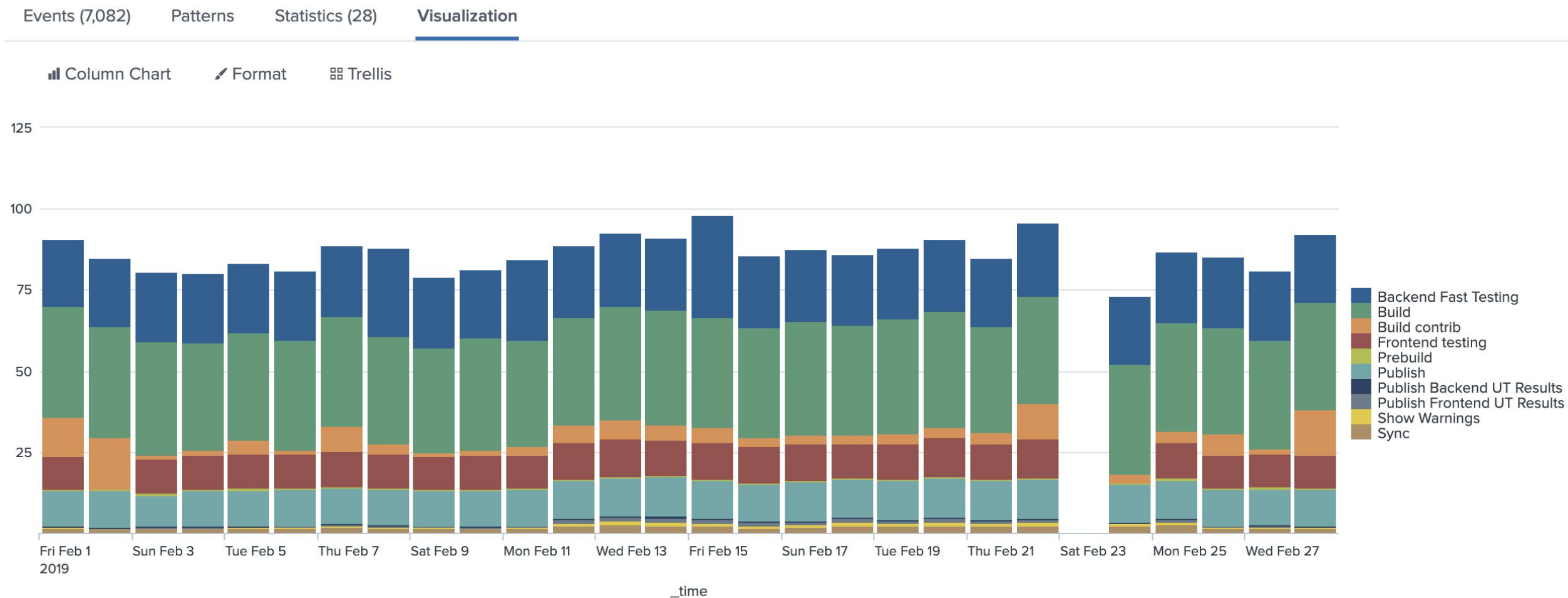
Build the final time chart showing each builds results

```
| timechart span=1d eval(round(avg(stageMinutes),  
1)) by stageName
```

Our Analysis

How long does each step of a job take?

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



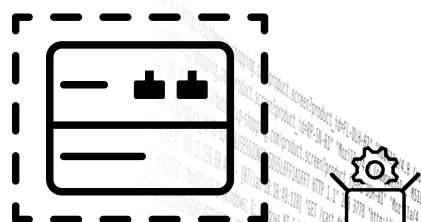


Speeding up the Splunk Build

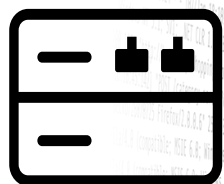
Distcc Architecture

Containers:

12 x 20 Core VMs

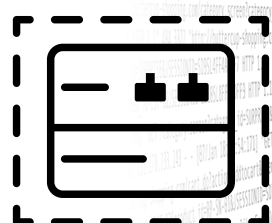


All DistCC Build Clients and servers use the same build toolchain and chroot



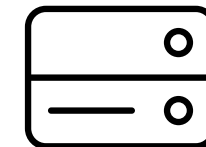
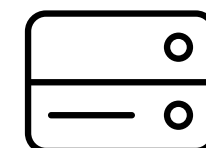
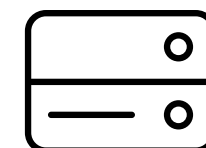
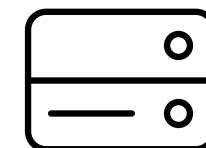
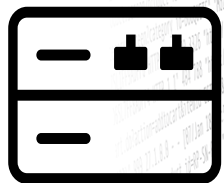
30 Docker agents

12 compile servers



DistCC server is used ONLY for compile

make -j80



DistCC VS Normal Build

▶ 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

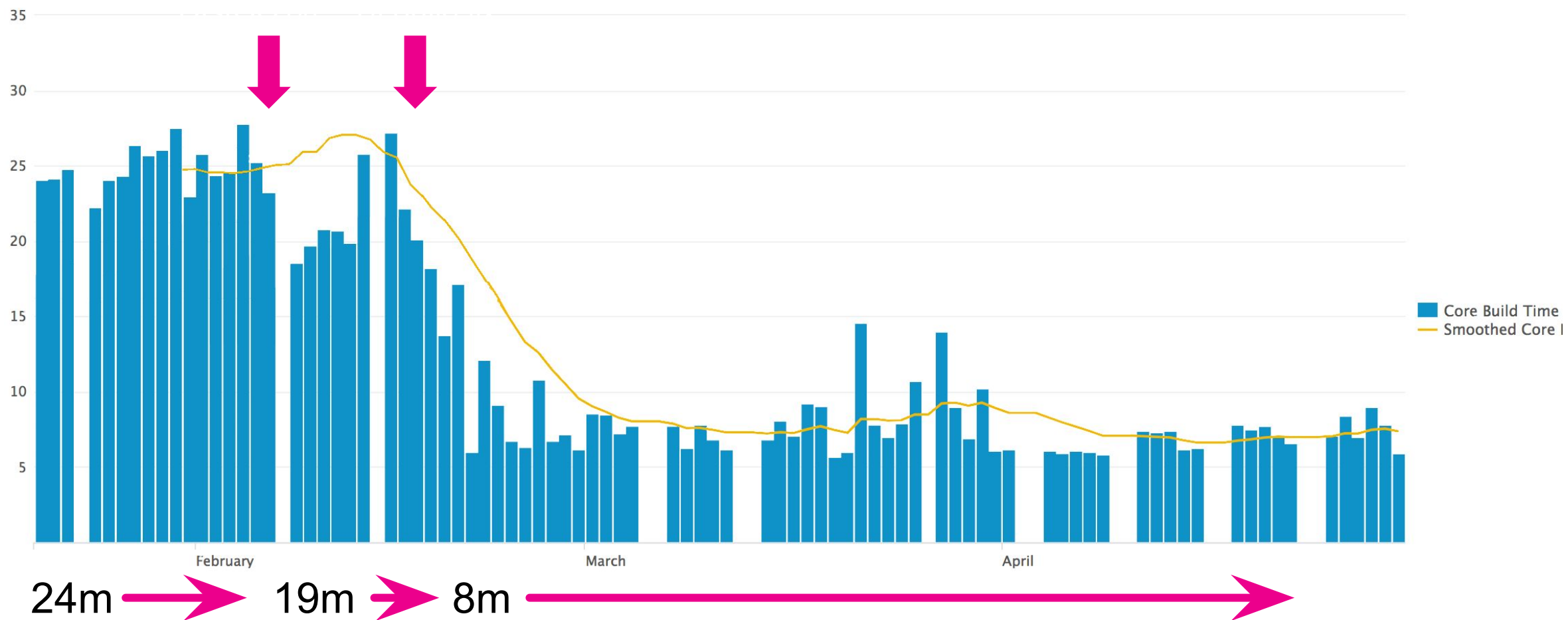
- `make -j 48`
- **Web UI -j 6**
- Optimal 24 core VM agent
 - 12 DistCC hosts

8m →



Build Time Improvement - Results

Dramatic reduction in the overall build time





Speeding up Testing

Our Analysis

Reducing the Build Testing Time

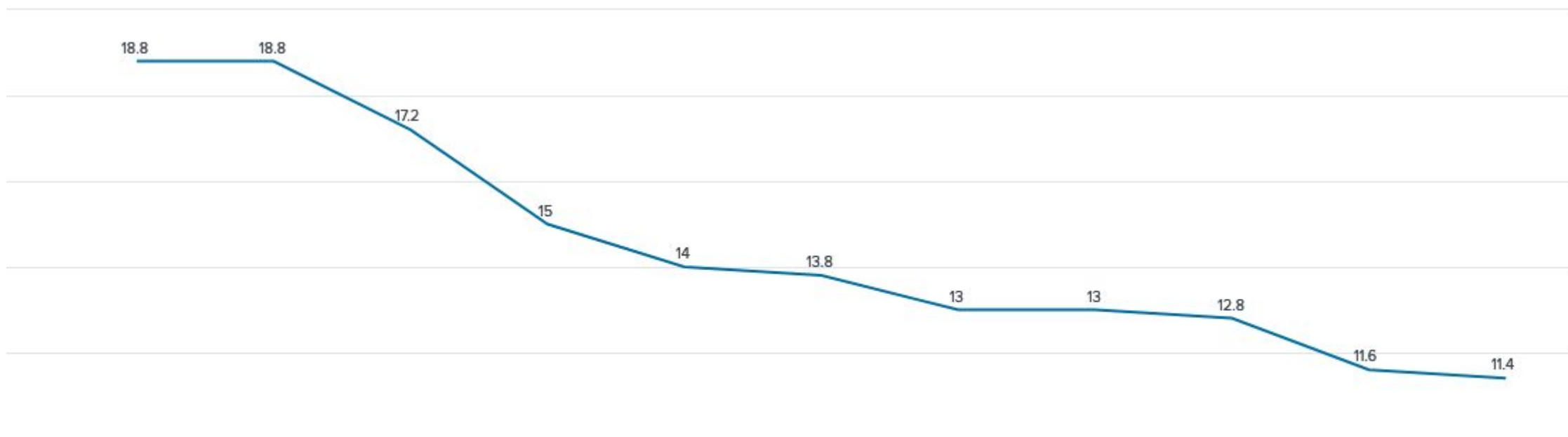
Analysis:

- Based on our previous analysis the backend unit tests were executed in parallel across 16 test instances on the build agents
- We used Splunk to measure the overall timing of each test configuration as well as the individual tests

Mitigation:

- Over time we increased the number of parallel test instances from 16 to 40 measuring the overall test time
- We stopped at 40 parallel tests because the build containers have 48 cores

Improvement in the overall test time as parallel instances were increased

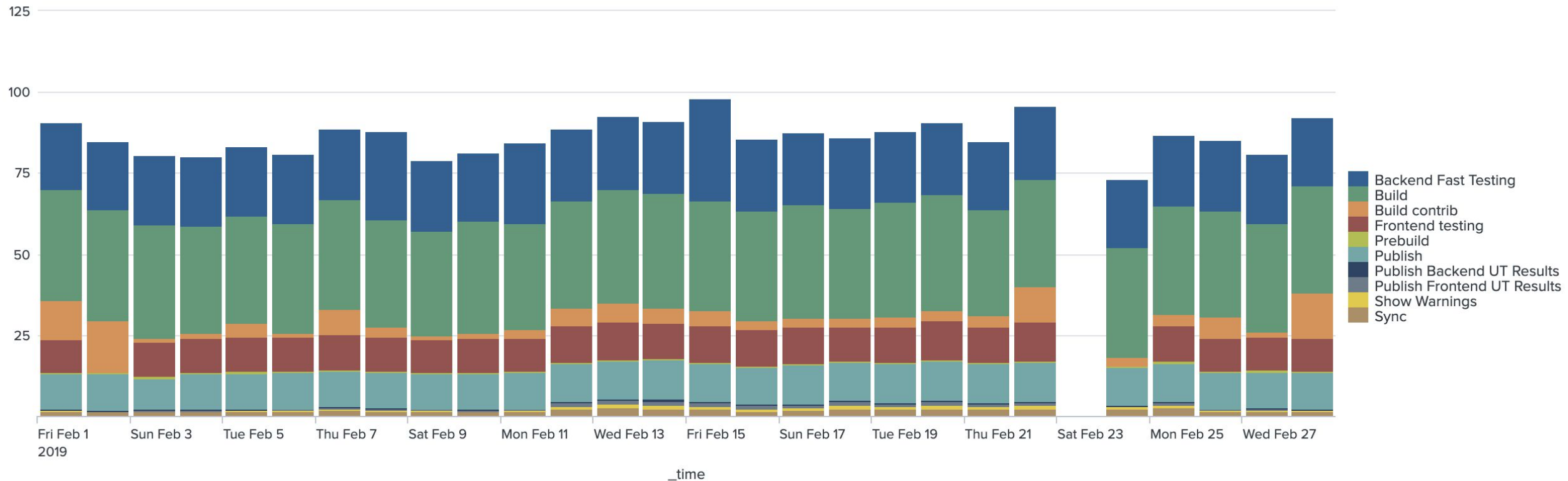


Before Optimization

Peak load period – > significant delays

Events (7,082) Patterns Statistics (28) **Visualization**

Column Chart Format Trellis

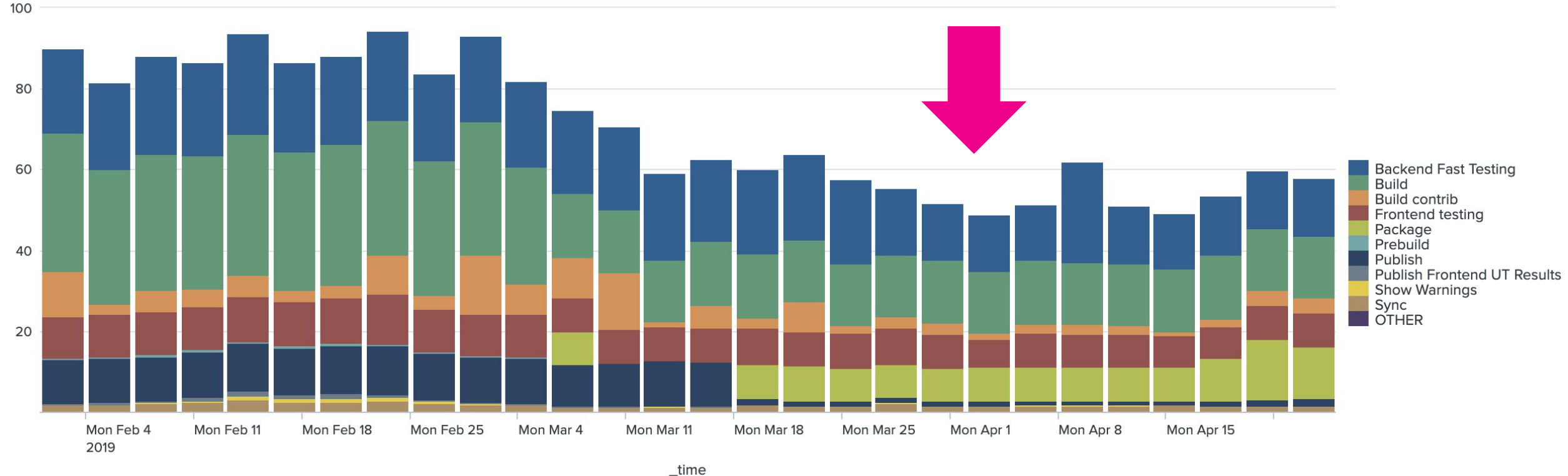


After Optimization

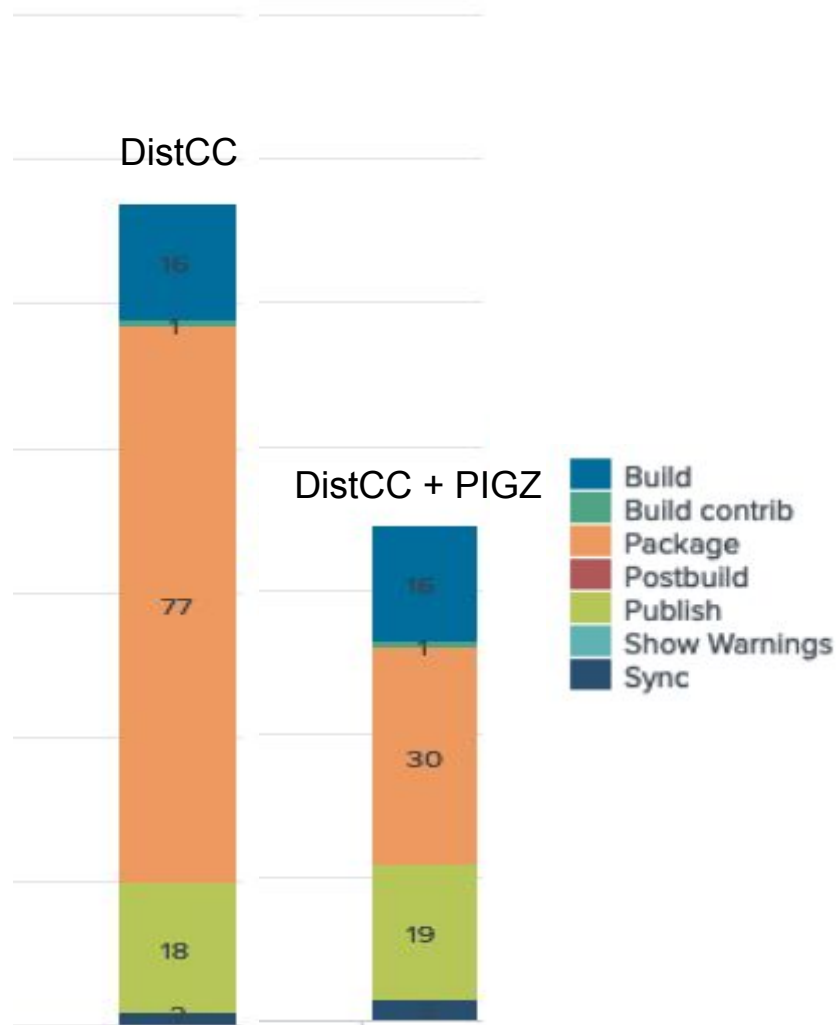
Recent similar trigger conditions

Events (14,403) Patterns Statistics (28) **Visualization**

Column Chart Format Trellis

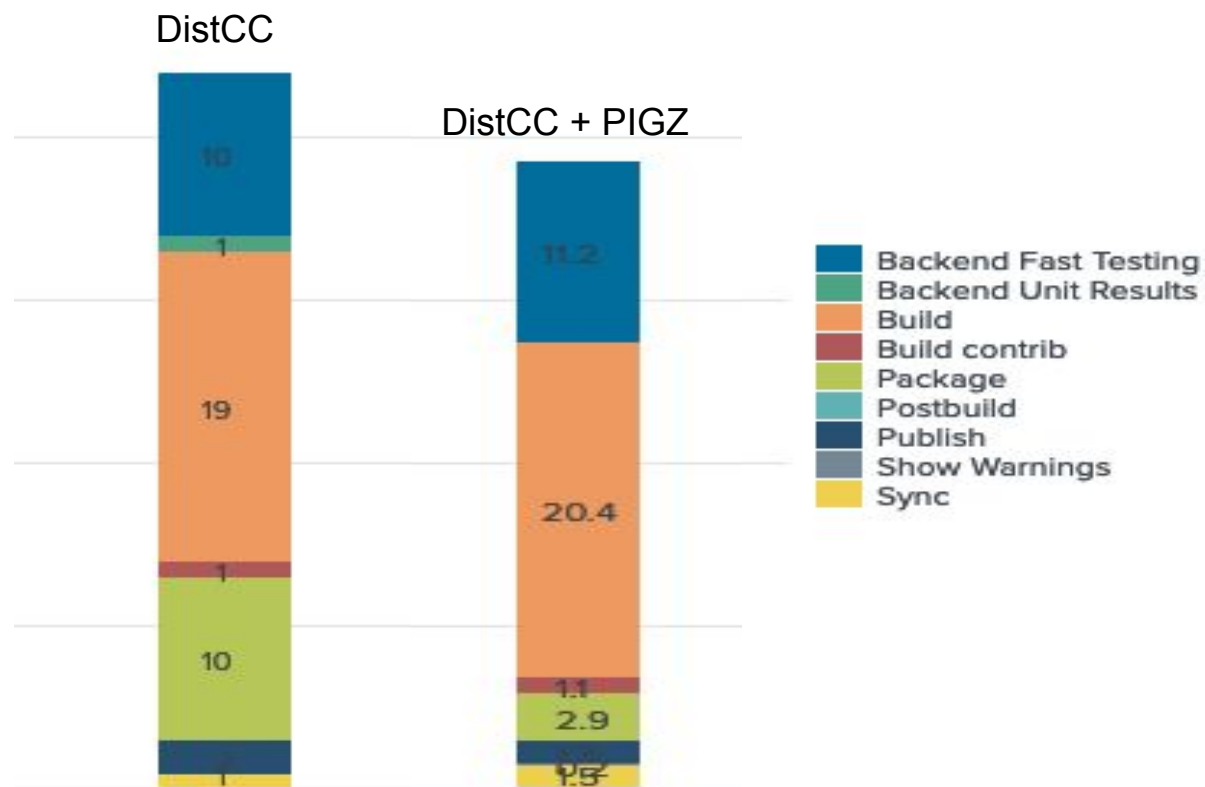


Production release build improvement with **DistCC**



- Production release build
 - 5 Components
 - 10 archives created
- Multicore archiving program
 - Utilizes multiple cores on the container
- 2x time reduction for packaging

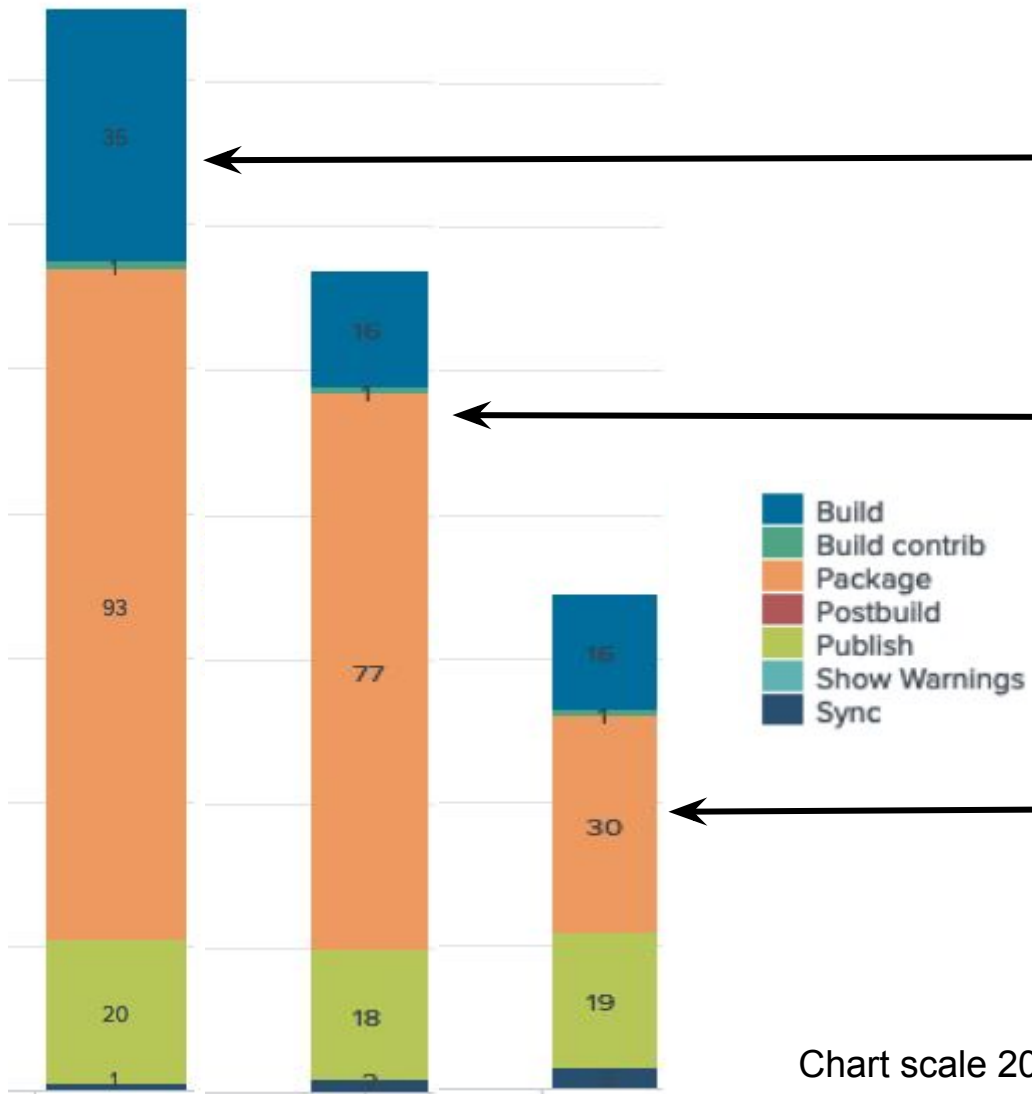
Pull Request Build Improvement with PIGZ



- Pull Request Build build
 - 3 Components
 - 3 archives created
- Multicore archiving program
 - Utilizes multiple cores on the container
- 2x time reduction for packaging

Splunk Production Build Time

Overall average build time reduced by 50 percent
Improved ~50%



Initial containerized builds did not use distcc, and used standard tar compression: 35 Minute build time

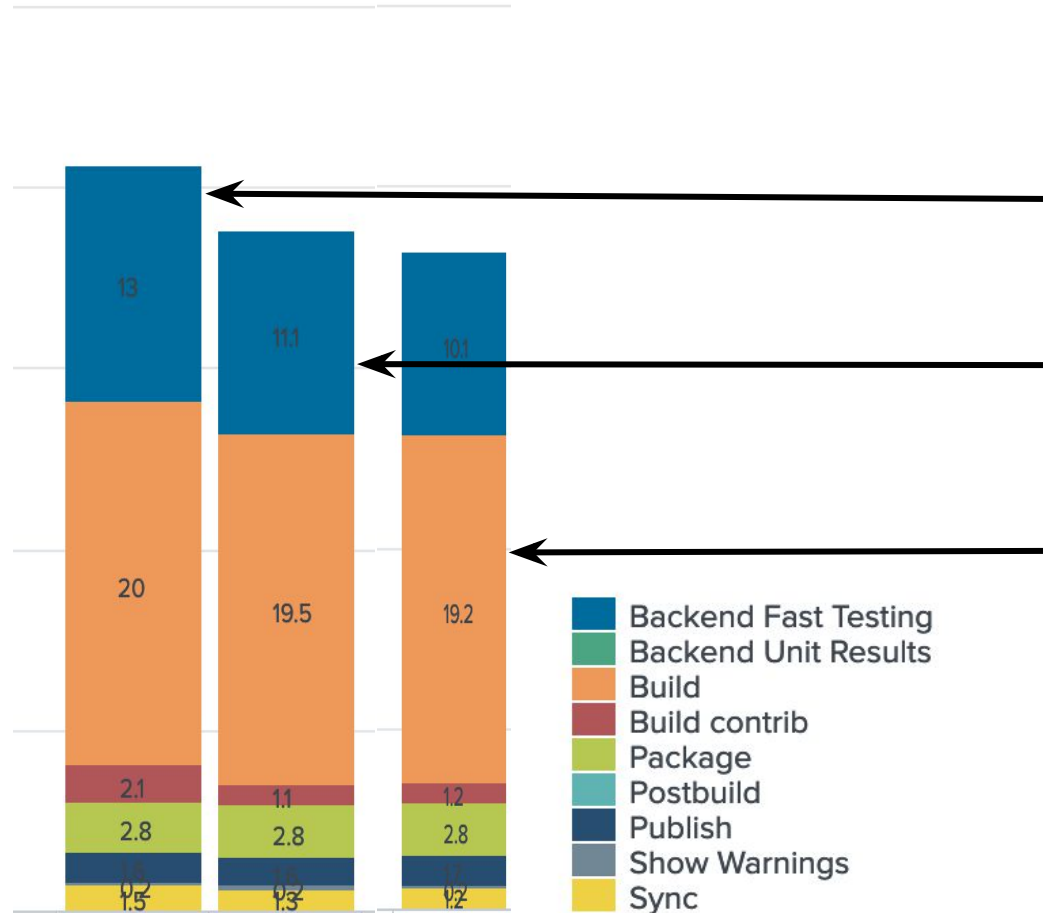
Adding distcc support cut the build times in half: 16 minute build time

Converting our packaging to use multiple core compression reduced packaging time by more than 50 percent

Chart scale 20 minutes/division

Splunk Pull Request Time Improvements

Overall average running time reduced by 50 percent



Initial test configuration with 16 tests running in parallel takes 13 minutes

Mid change, 25 tests running in parallel takes 11.1 minutes

Final test configuration has 40 backend unit tests running in parallel, with distcc and pigz performance improvements takes 10.1 minutes, final total running time 36 minutes

Chart scale 10 minutes/division



Improving Notifications

Our Analysis

Speeding up Developer Notification

Analysis:

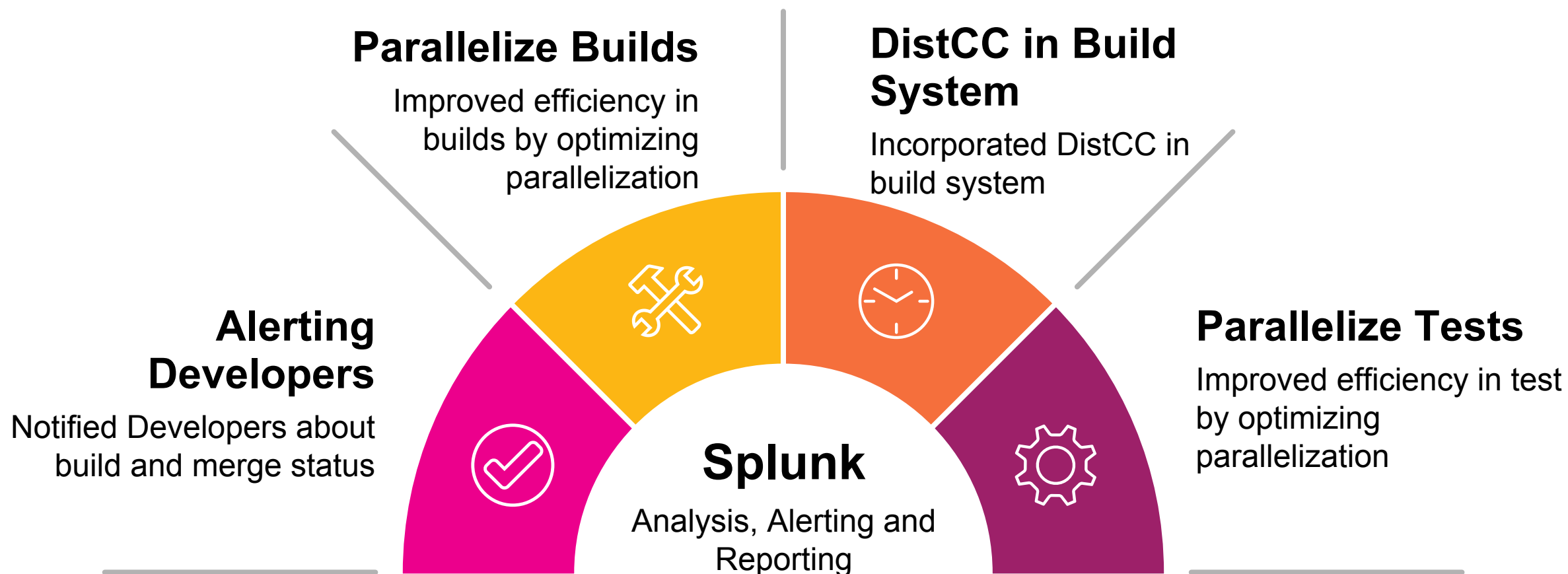
- Test results only available as comments in pull request UI
- Developers need to go to Bitbucket frequently to check for results
 - All employees are connected via Slack

Mitigation:

- Added personal Slack messages
 - On receipt of trigger
 - When job is completed → includes test results

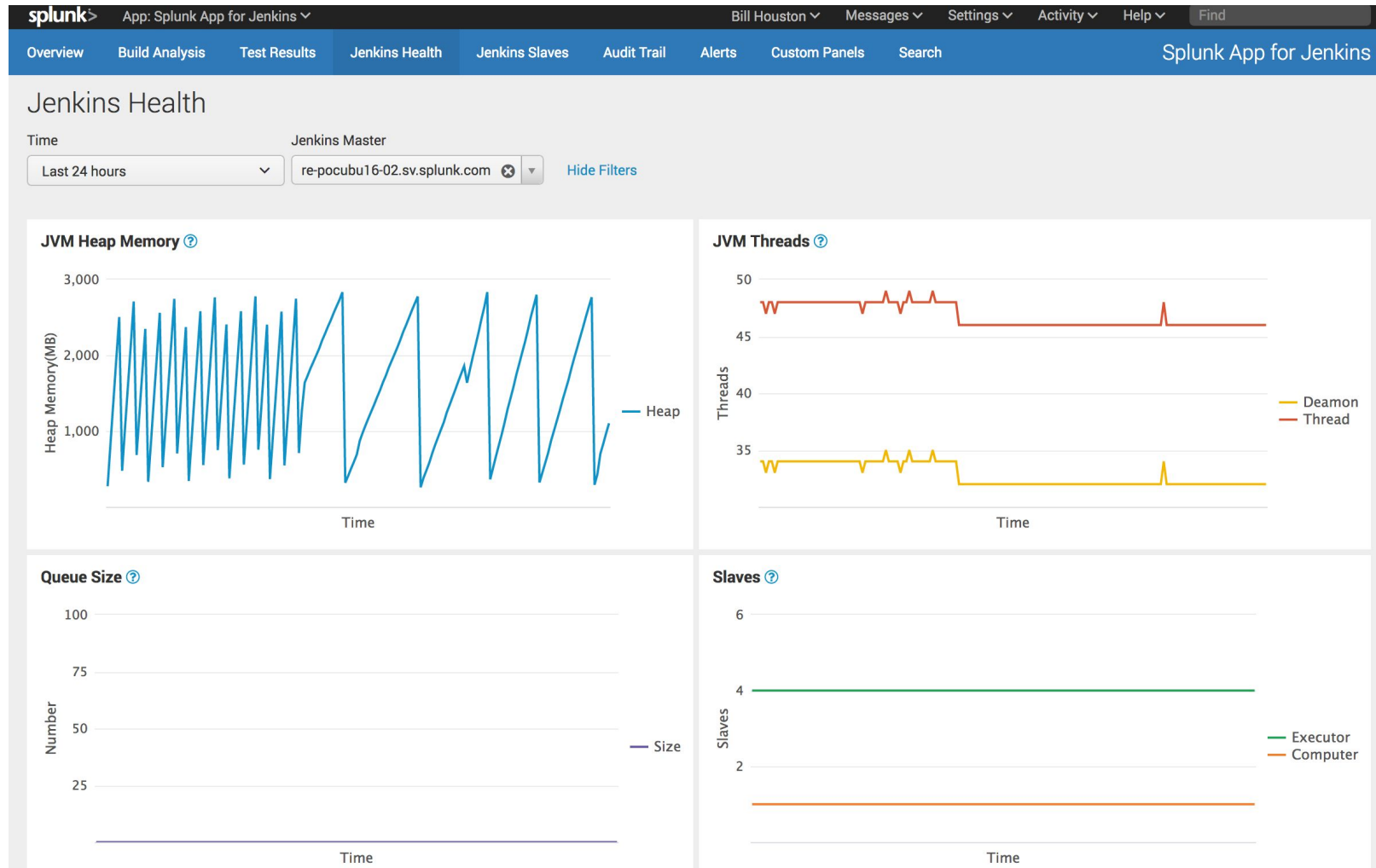
Overall Improvement

Key Takeaways



The Jenkins App for Splunk

Seamlessly collect, monitor and analyze Your Jenkins Data

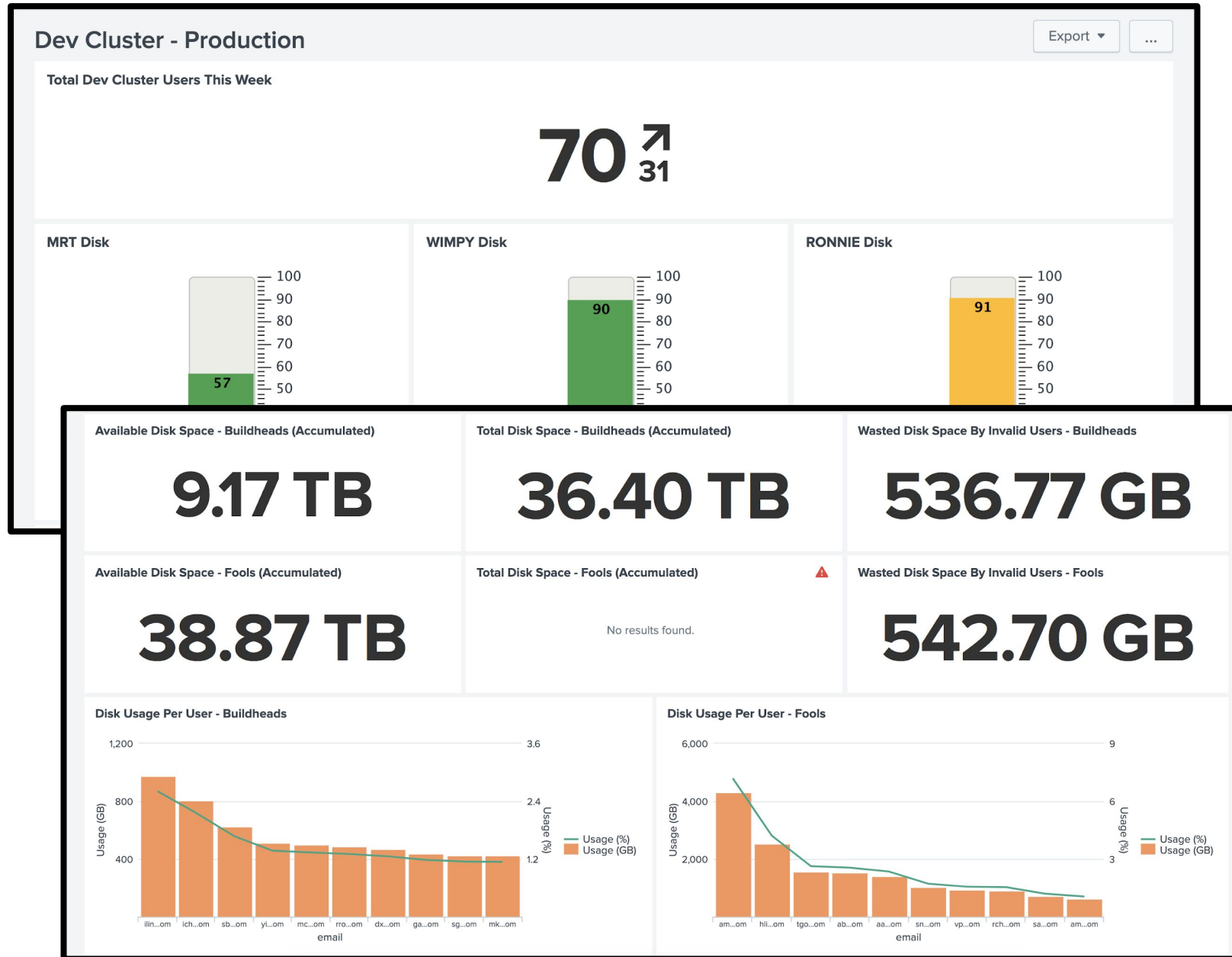


DevOps

→ DevCluster

□ Artifactory

□ Jenkins

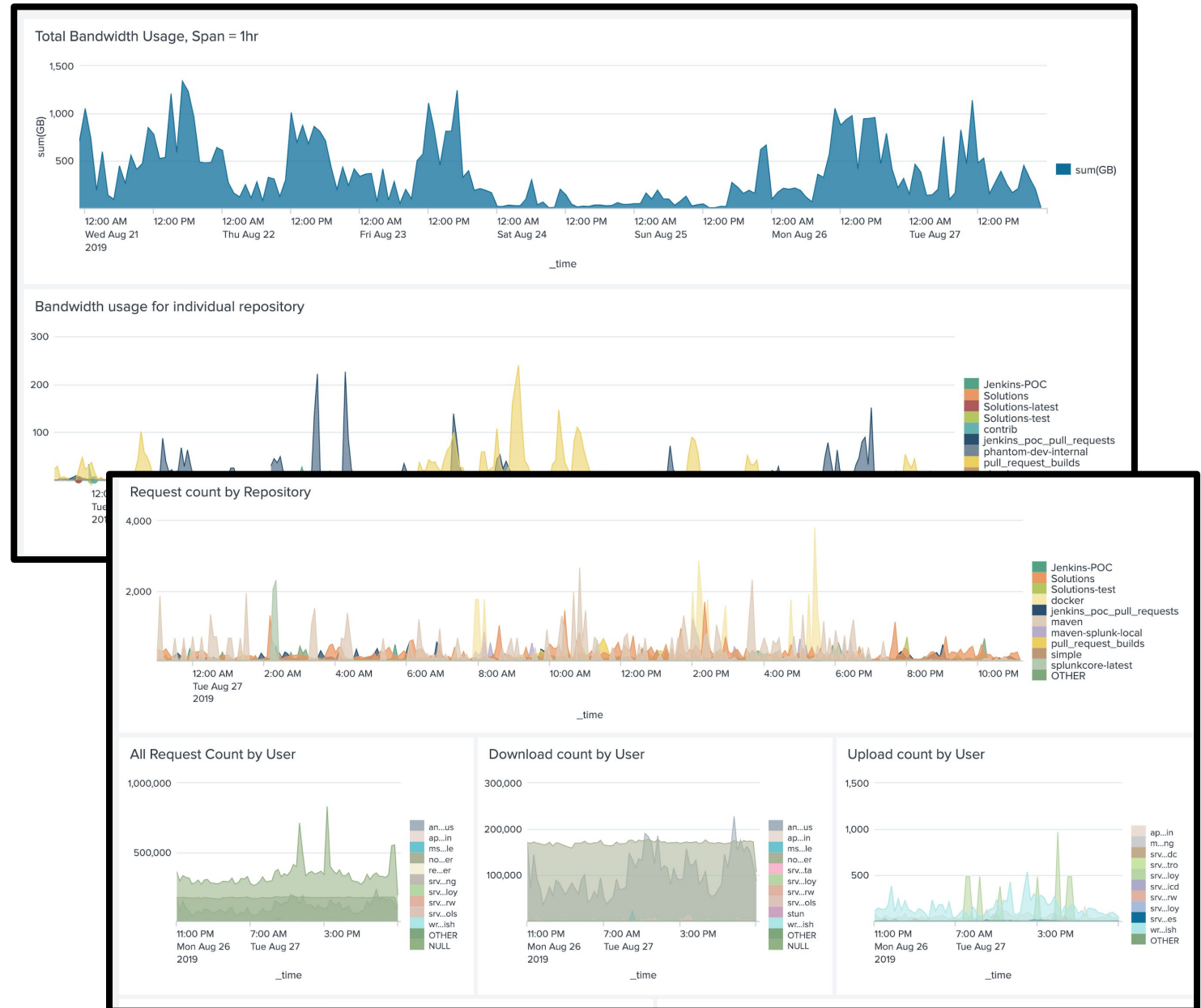


DevOps

□ DevCluster

➔ Artifactory

□ Jenkins

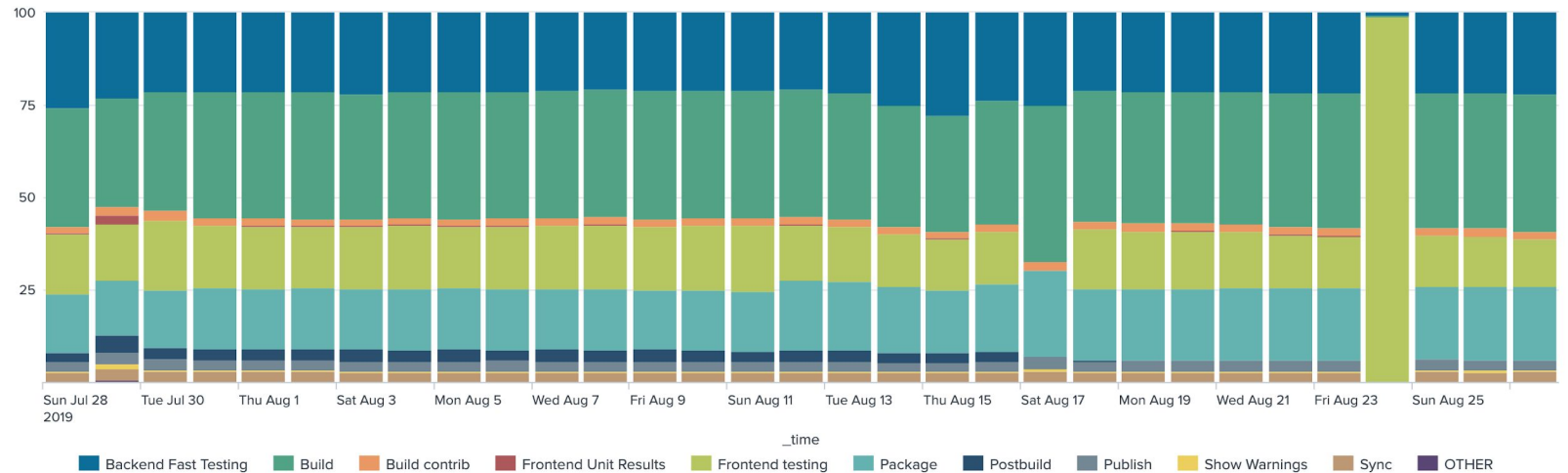


DevOps

- DevCluster

Artifactory

➔ Jenkins



_time	Backend Fast Testing	Build	Build contrib	Frontend Unit Results	Frontend testing	Package	Postbuild	Publish	Show Warnings	Sync
2019-07-28	14.52710	18.16523	1.07140	0.06992	9.37882	8.86540	1.45635	1.48620	0.21534	1.46695
2019-07-29	14.79525	18.87707	1.46140	1.47374	9.77394	9.72591	3.01183	1.97306	0.81786	1.93698
2019-07-30	12.33035	18.54812	1.57295	0.06602	10.76380	9.14457	1.72032	1.68032	0.29207	1.65183
2019-07-31	11.63593	18.59426	1.13495	0.07270	9.17360	8.93835	1.71763	1.52153	0.21970	1.56025
2019-08-01	11.59110	18.72688	1.12073	0.07693	9.25633	8.83682	1.68110	1.50230	0.17630	1.62465
2019-08-02	11.39035	18.57178	1.03100	0.07520	9.01565	8.81320	1.68557	1.51050	0.17315	1.54210

_time	Backend Fast Testing	Backend Unit Results	Build contrib	Build contrib	Frontend Unit Results	Frontend testing	Package	Publish	Show Warnings	Sync
2019-08-20	11.3508	0.02635	19.07747	1.1017	0.06639	7.76889	10.4945	1.5506	0.16790	1.46550
2019-08-21	11.45780	0.03121	19.17198	1.14760	0.07337	8.16260	10.44740	1.56330	0.18310	1.49430
2019-08-22	11.66310	0.03453	19.35802	1.14270	0.06850	7.72937	10.43055	1.56635	0.18563	1.48000
2019-08-23	11.43480	0.03097	19.13514	1.10740	0.07262	7.26130	10.31465	1.55950	0.17743	1.45500
2019-08-24	20.75055	0.02574	10.54810	0.97258	0.0153	2886.8770	6.63977	1.14820	0.20972	1.3419
2019-08-25	11.40320	0.02920	19.16045	1.1126	0.07637	7.16380	10.29730	1.64860	0.18903	1.59880
2019-08-26	11.43260	0.02990	19.29505	1.15685	0.07487	7.16932	10.41955	1.55145	0.20684	1.48000

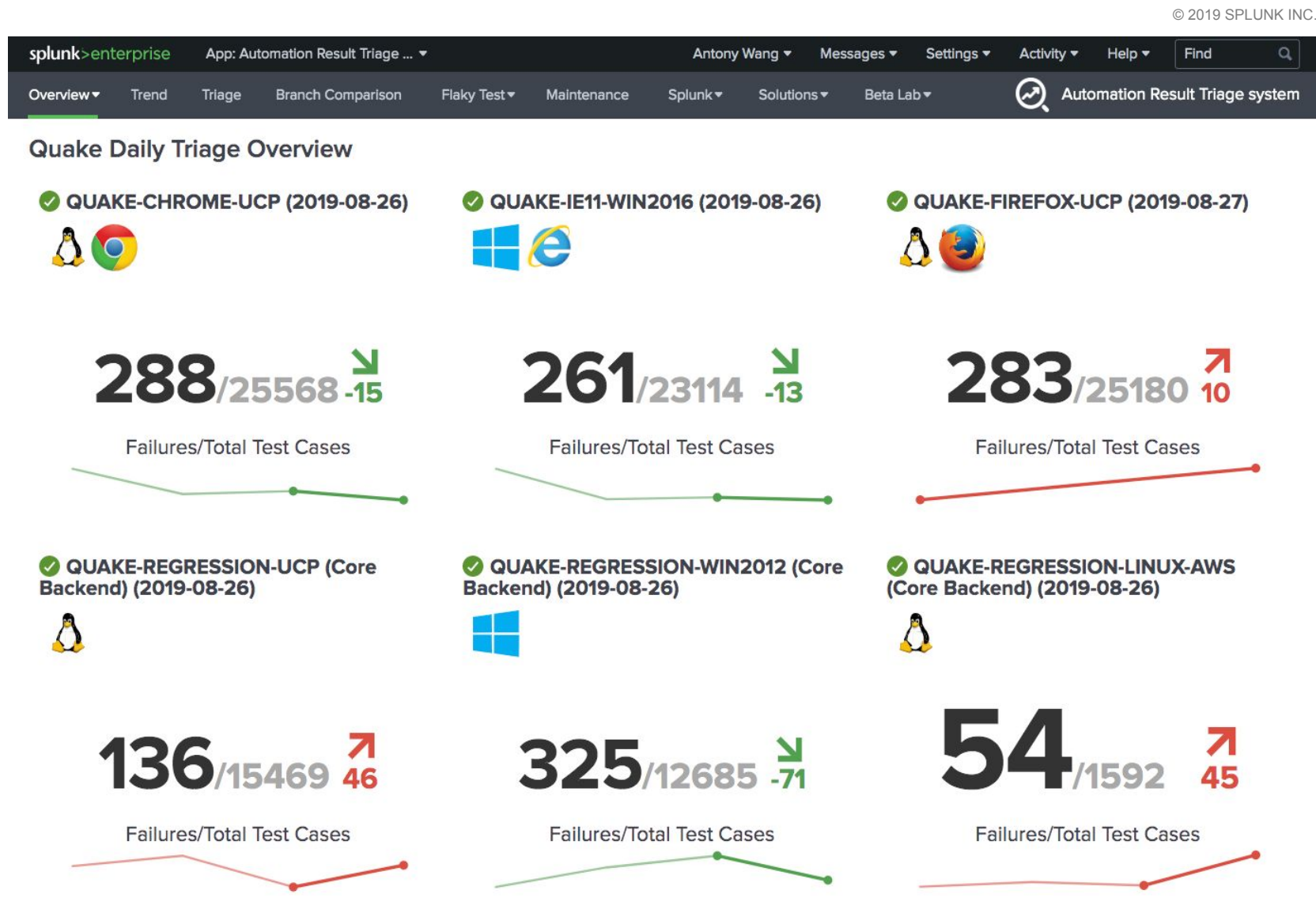
Test, CI & more

Golden Branch Health

Test Triage (ARTs)

Code Coverage

PR Cycle Time



Test, CI & more

Golden Branch Health

Test Triage (ARTs)

Code Coverage

PR Cycle Time

splunk>enterprise App: Automation Result Triage ... Antony Wang Messages Settings Activity Help Find


Overview Trend Triage Branch Comparison Flaky Test Maintenance Splunk Solutions Beta Lab Automation Result Triage system

Branch Comparison

Between Date-times ☐ Show Job Info ☒ Show Results

Summary Table Fields: Total Added Removed Fixed Failed Existing Regression New


✓ Job Name: CoreFrontend/PR_DOM_frontendregression



Total tests	Pass rate	Total failures	Skipped tests	Job duration
20875	99.69%	64	4	1h56m

Product bug failures	Setup failures	Start/Restart failures	Connection failures	Flaky failures	Test failures
1	0	0	0	0	63

✓ vs. Job Name: CoreFrontend/Commit_DOM_frontendregression



Total tests	Pass rate	Total failures	Skipped tests	Job duration
20880	100.00%	0	4	1h47m

Product bug failures	Setup failures	Start/Restart failures	Connection failures	Flaky failures	Test failures
0	0	0	0	0	0

Component Included: * X Component Excluded: X

Summary ☐ Show Jobs with No Tests

Feature	Total	Added	Removed	Fixed	Failed	Existing	Regression	New
client_webdriver_simplexml_panels	2047(0)	0	0	0	27(+27)	0	27	0
client_webdriver_manager_data	476(0)	0	0	0	13(+13)	0	13	0
client_webdriver_components_inputs	346(0)	0	0	0	10(+10)	0	10	0
client_webdriver_simplexml_vizeditor	3213(0)	0	0	0	9(+9)	0	9	0
client_webdriver_default	104(-4)	0	4	0	3(+3)	0	3	0

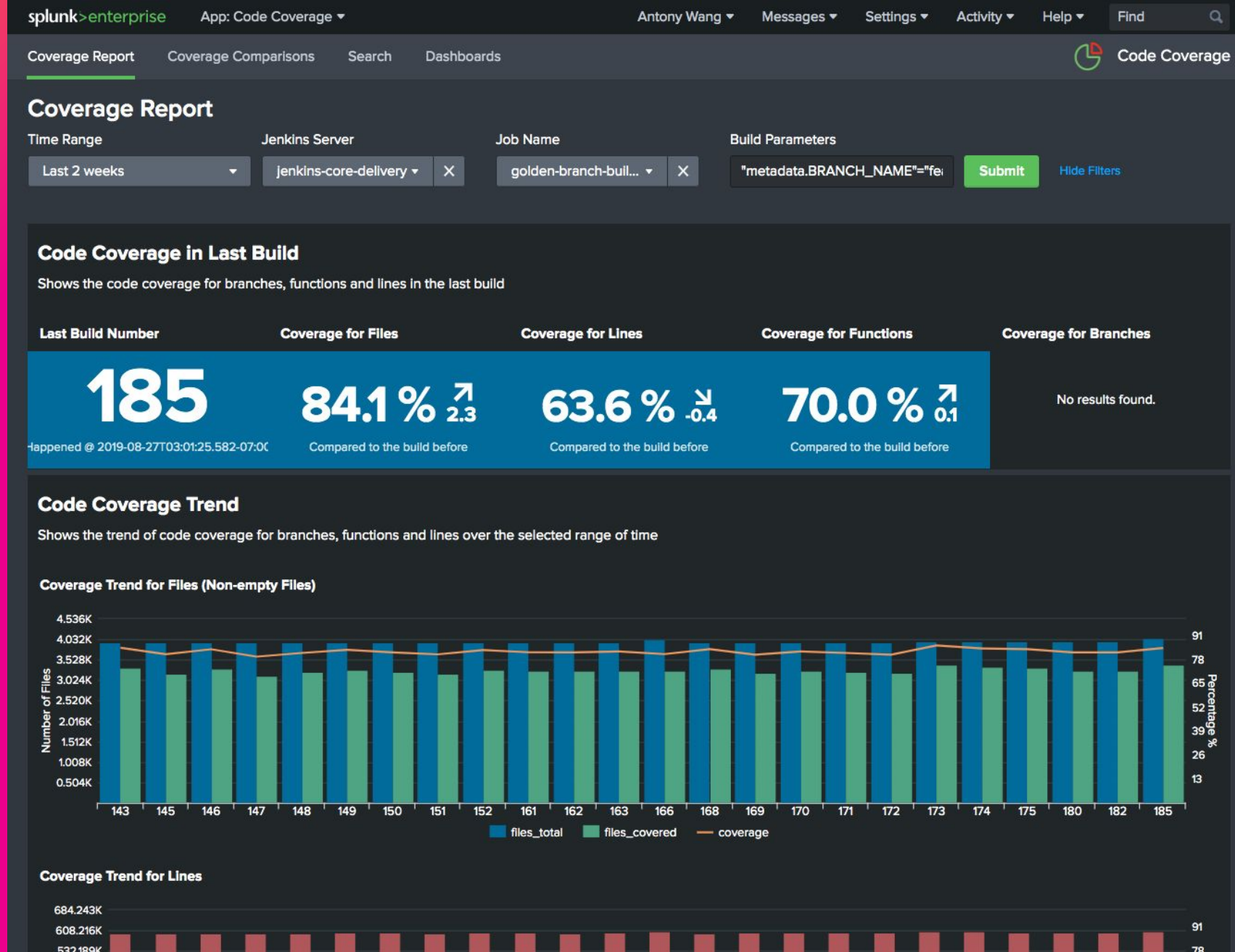
Test, CI & more

Golden Branch Health

Test Triage (ARTs)

Code Coverage

PR Cycle Time



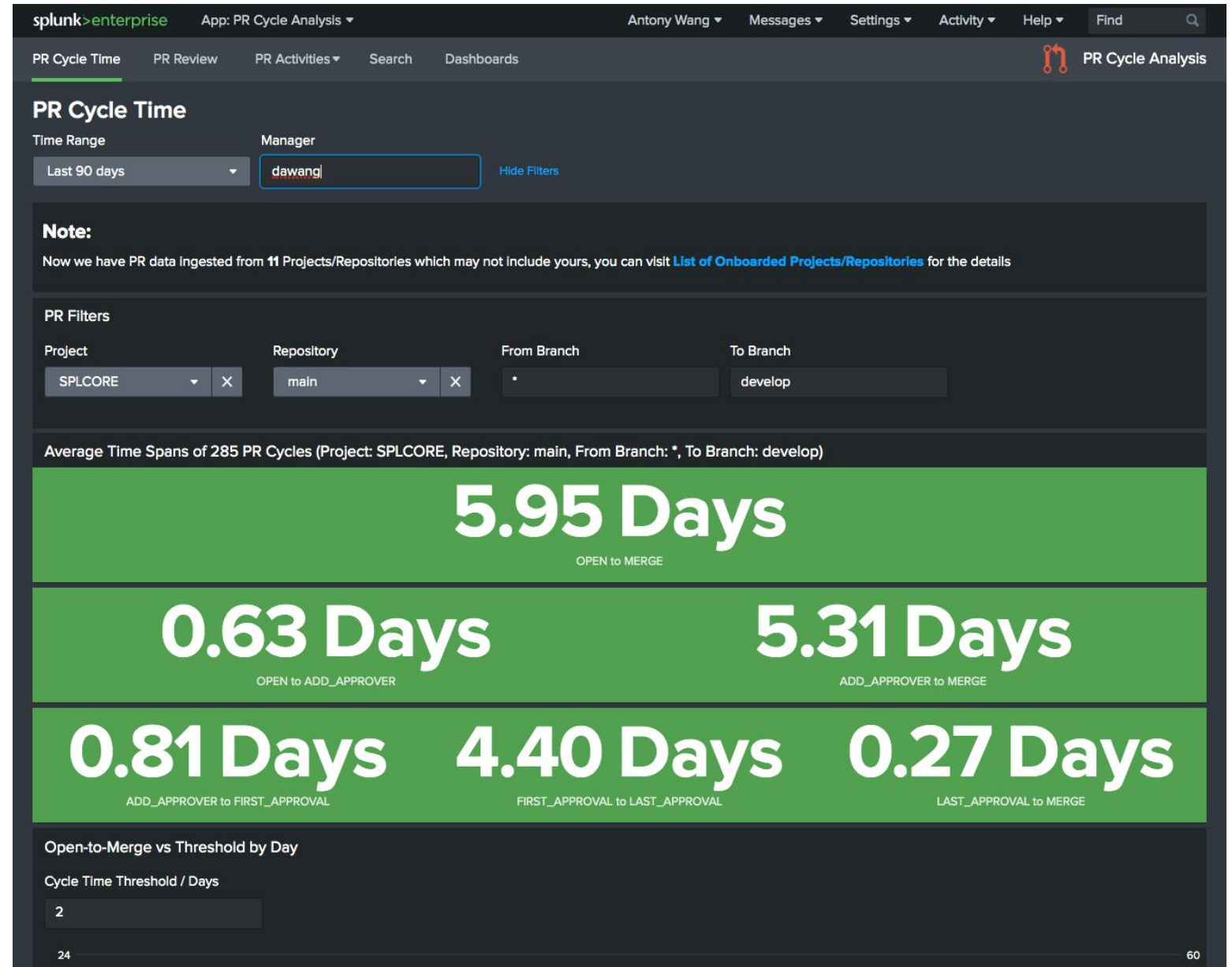
Test, CI & more

□ Golden Branch Health

□ Test Triage (ARTs)

□ Code Coverage

→ PR Cycle Time





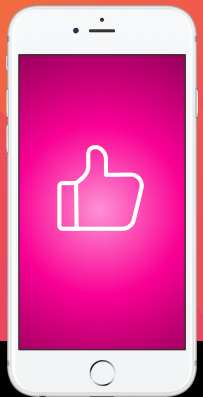
Thank

You



Go to the .conf19 mobile app to

RATE THIS SESSION





Q&A
