



Paul Maddox, AWS
Developer Technologies



A tale of two pizzas: Accelerating Software Delivery with AWS Developer Tools

June 2017

AWS



@paulmaddox

S U M M I T

A tale of two pizzas

Accelerating Software Delivery with AWS Developer Tools

Paul Maddox, Developer Technologies
Kelvin Zhu, Lead Developer, Okta

June 2017



Agenda

- Common challenges
- 100mph history of Amazon's journey
- Amazon's internal practices and tooling
- AWS Developer Tools



15yrs

The **average lifespan**
of an **S&P company**
dropped from 67 years in
the 1920s to 15 years today

2/3

More than two-thirds of
IT budgets go toward
keeping the lights on

77%

of **CEOs** believe **security risk** has increased in the
last few years and **65%**
believe their **risk management**
capability is **falling behind**



How This Affects You

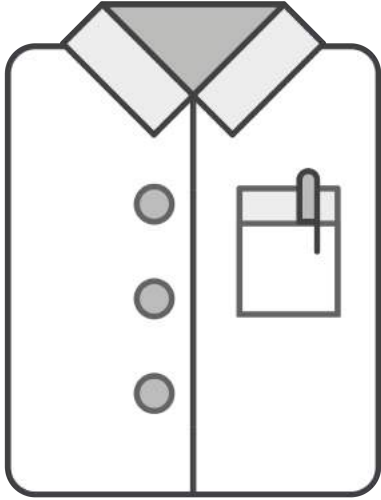
You're left **without the necessary resources** to pursue critical business initiatives required to maintain a competitive advantage

Your traditional IT model **lacks the agility** you need to keep pace with innovative startups

Insufficient security, compliance and availability can hamper your ability to compete and open the door to sophisticated, hard-to-identify attacks

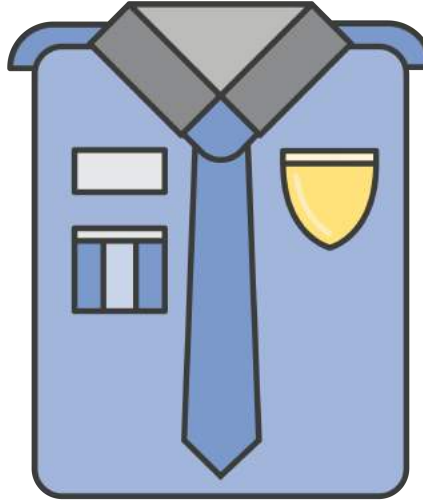
*“Finding time for innovation is hard,
we’re just **too busy**...”*

A world of conflicting priorities



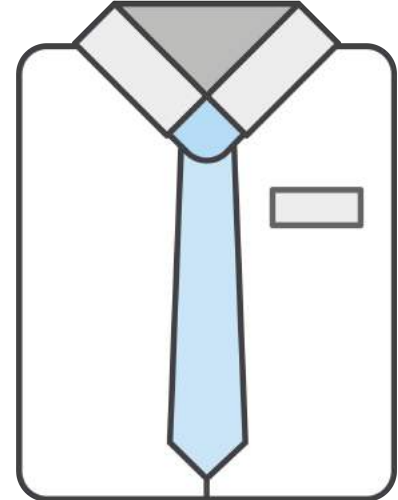
Developers

Paid to change things



Security

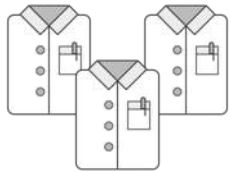
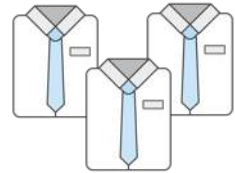
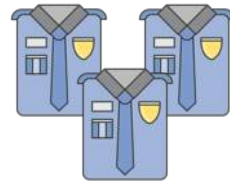
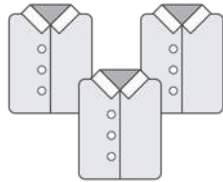
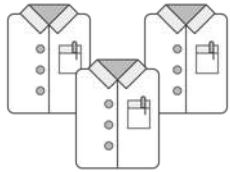
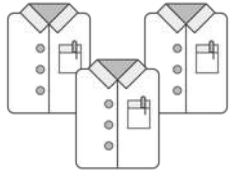
Paid to prevent risk



Operations

Paid to ensure stability

...and of bottlenecks

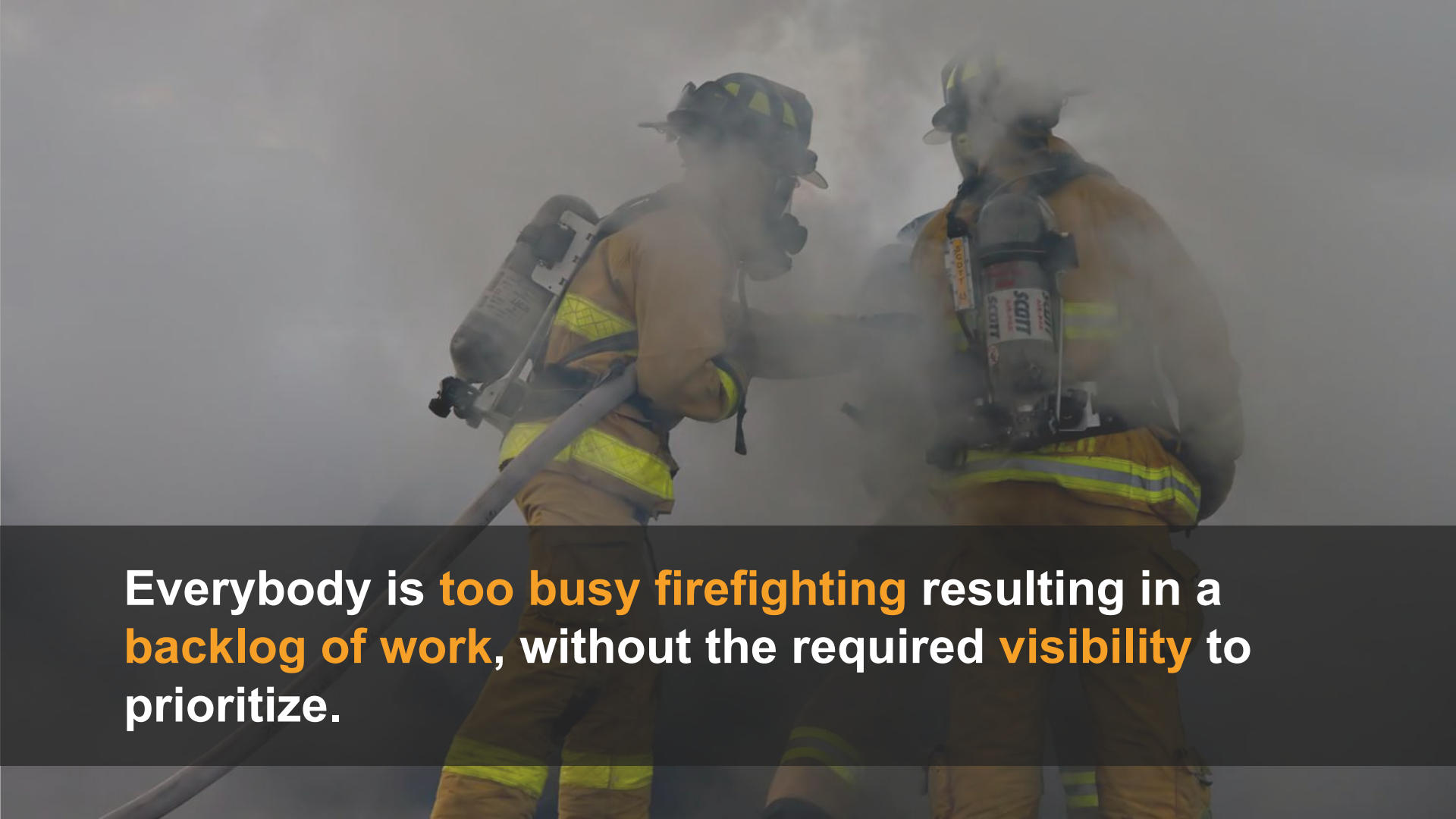


Development

Testing

Security

Operations



Everybody is **too busy firefighting** resulting in a **backlog of work**, without the required **visibility** to prioritize.



Large releases contain so many pieces, that it's easy to lose track and revert to **fear of change**, resulting in **analysis paralysis**.

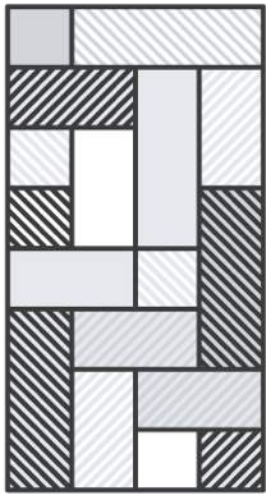
The image shows the interior of a large, historic cathedral. The architecture features high vaulted ceilings, stone pillars, and a series of large, arched windows at the top. A wide, central staircase with stone balustrades leads up to a higher level. Several people are seen walking on the stairs and in the lower areas, providing a sense of scale. The lighting is warm, coming from the windows and interior lamps.

Retrospective: Development @ Amazon



Development transformation at Amazon: 2001-2009

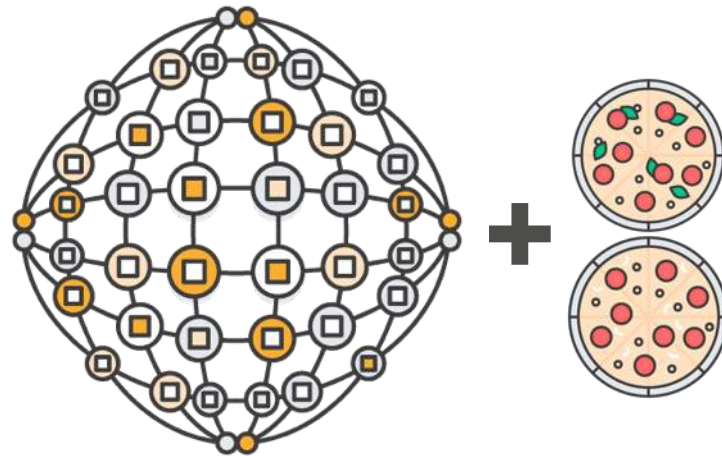
2001



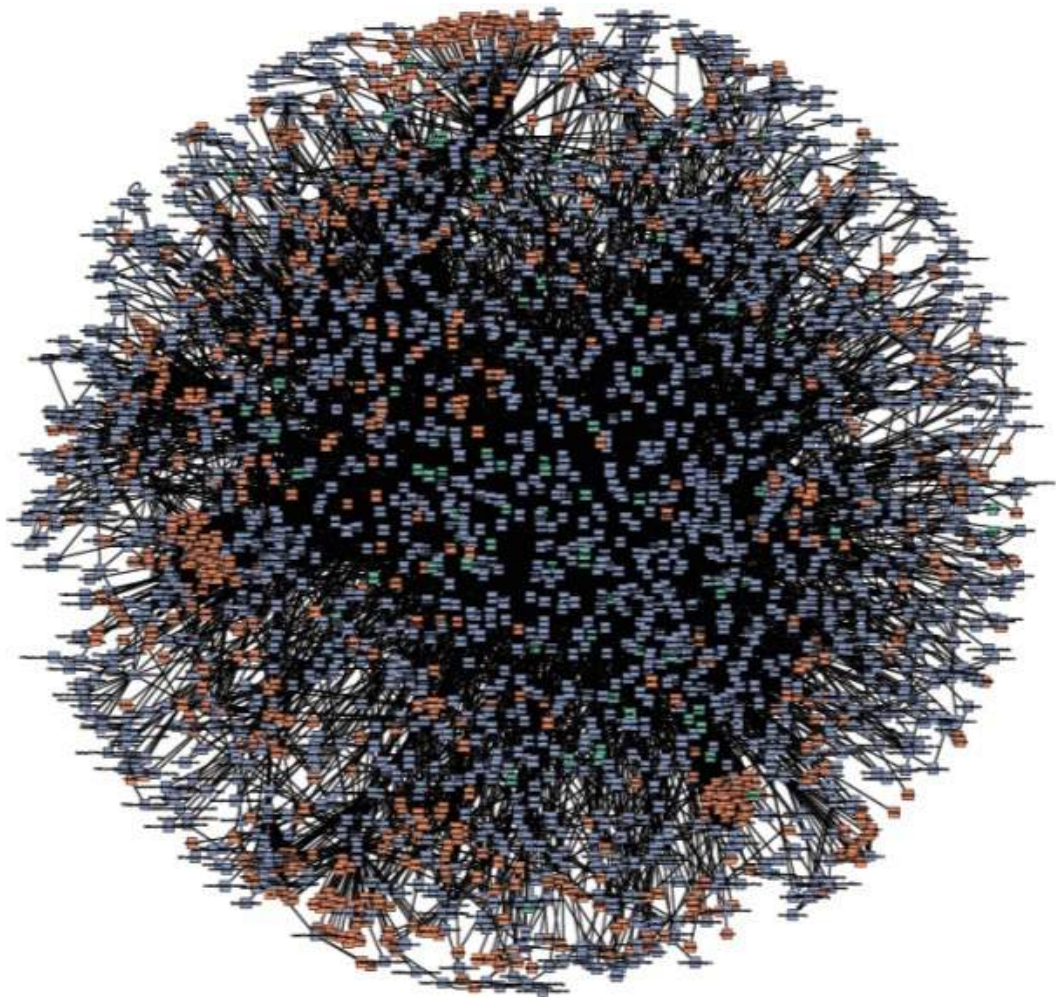
monolithic
application + teams



2009

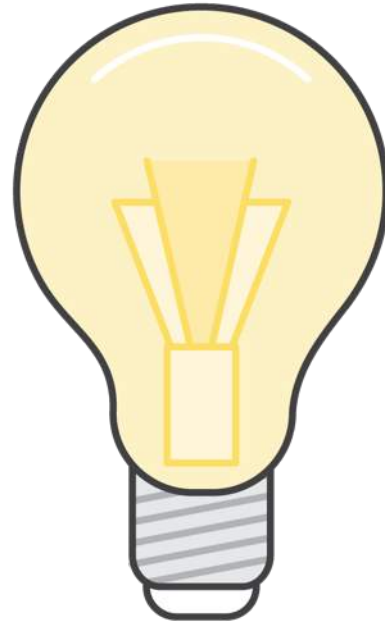


microservices + 2 pizza teams



Amazon Retail Platform (2009)

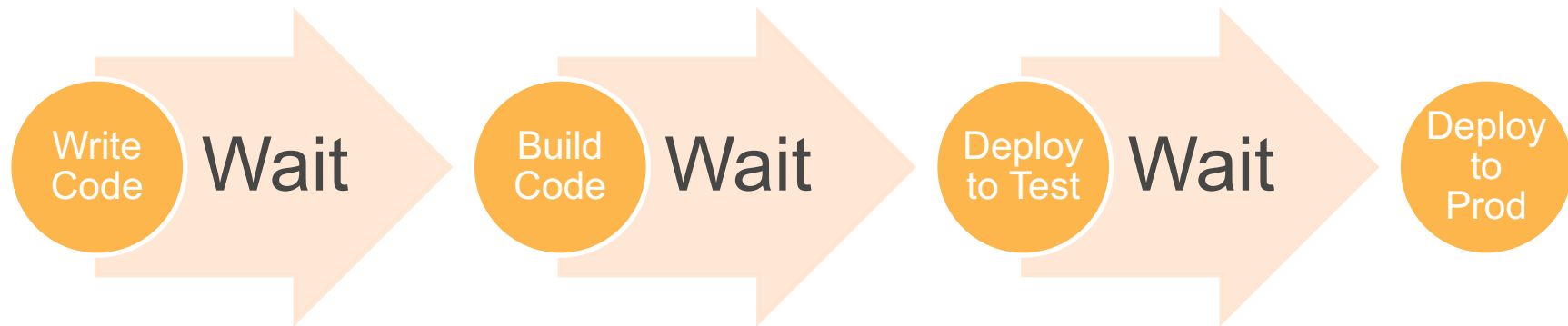
Things went much better under this model and teams were releasing faster than ever, but we felt that we could still improve.



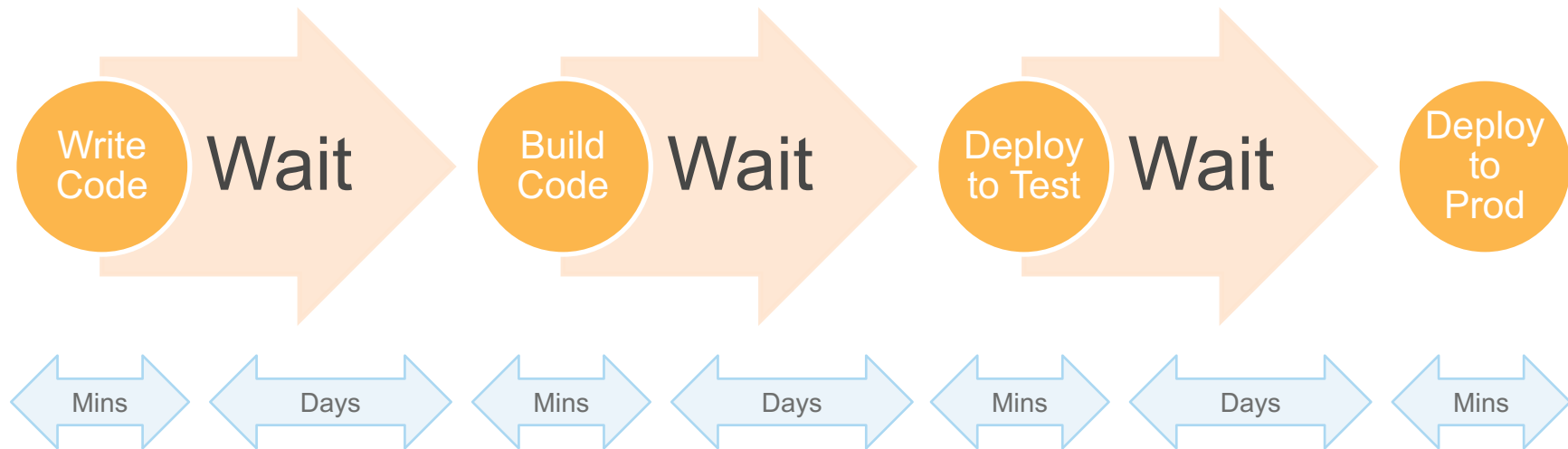


In 2009, we
ran a study to
find out where
inefficiencies
might still exist

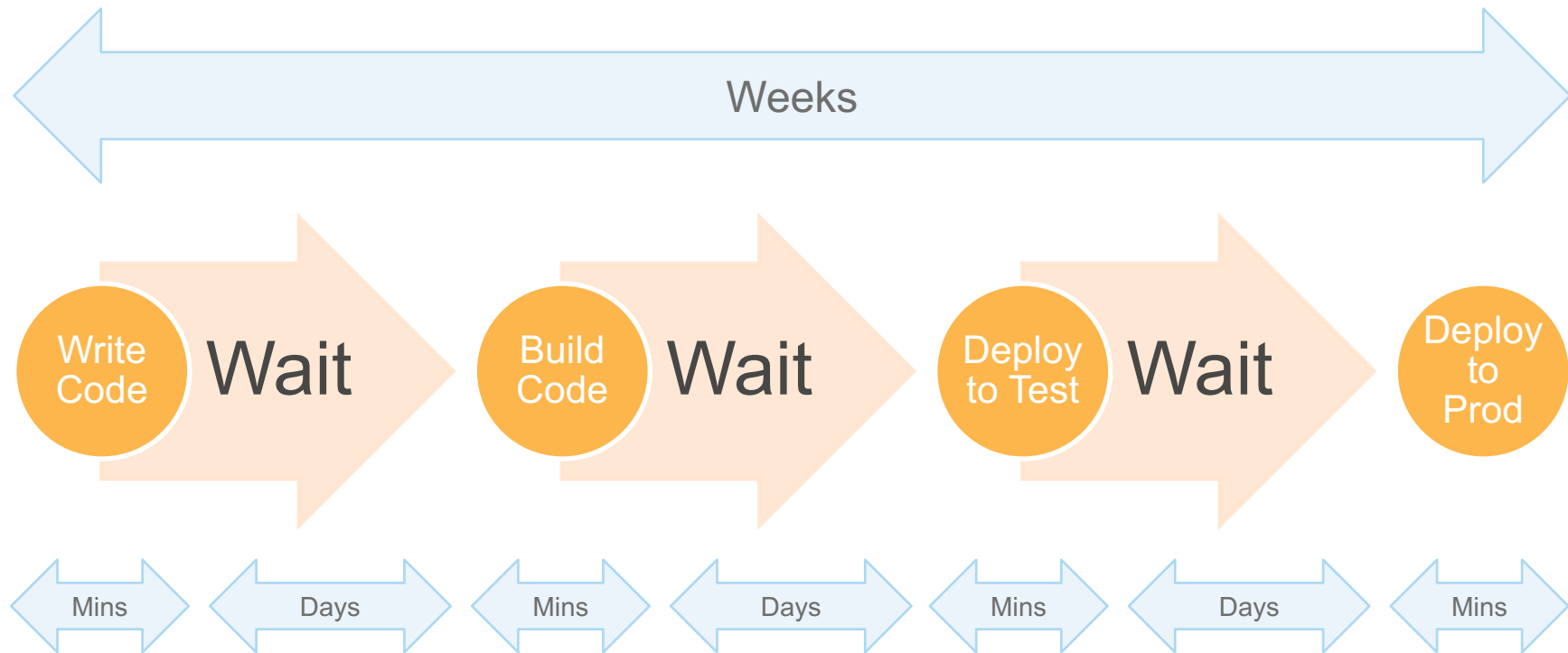
We were just waiting.



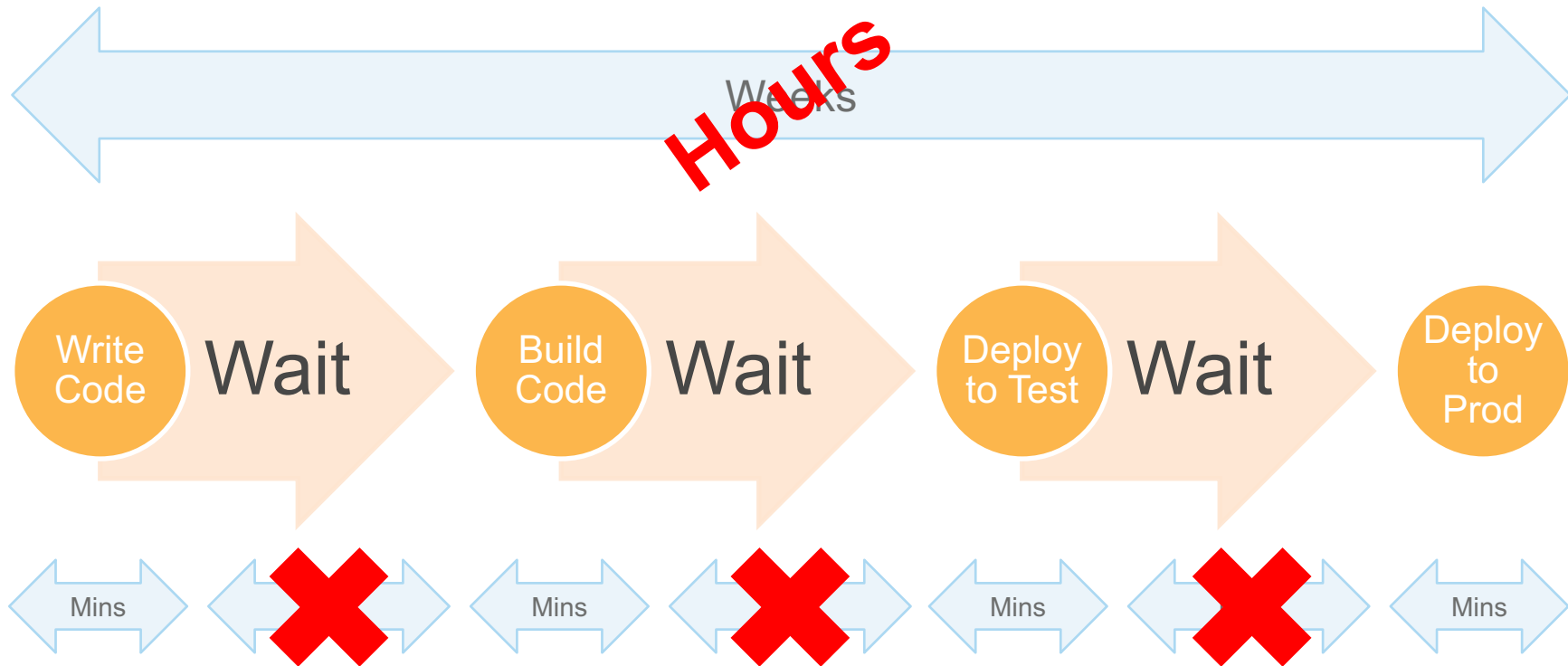
We were just waiting.

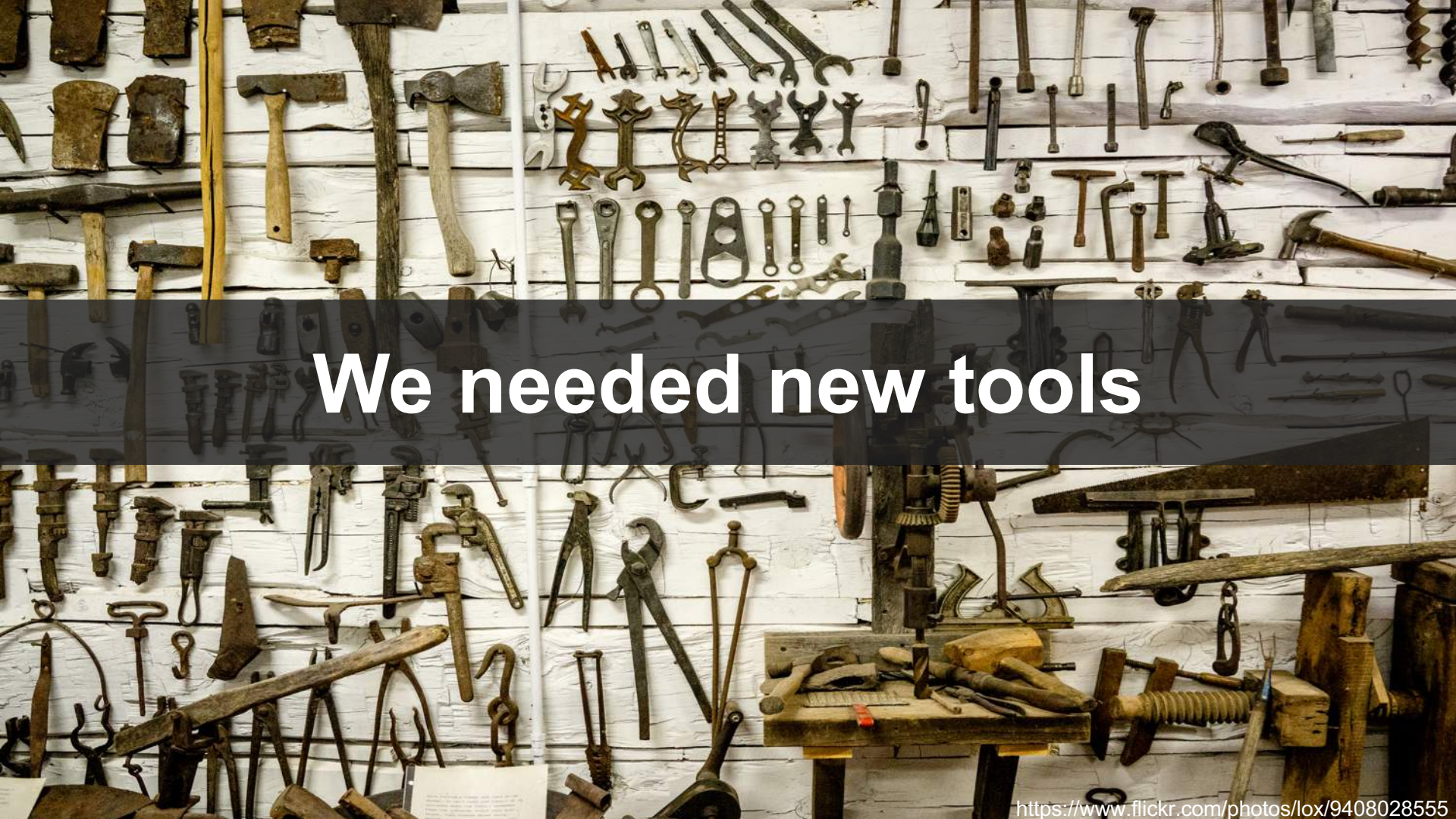


We were just waiting.



We were just waiting.





We needed new tools

**Tooling should be decentralised,
encouraging self service.**

**It should promote best practices
WITHOUT being restrictive.**

It should be technology agnostic.

#1: It should be the path of least resistance.



Pipelines

Continuous Delivery

From check-in to
production

CI/CD + Release
Automation

>90% of Amazon teams



Rolling Deployments
(zero downtime)

Health Checking

Versioned Artifacts &
Rollbacks

General Best Practices used by Amazon Developers

- CI/CD is a MUST!
 - Commit frequently
 - Builds on every commit
 - Build once in a given execution flow
 - Deploy to a running environment for further testing



General Best Practices used by Amazon Developers

- CI/CD is a MUST!
 - Commit frequently
 - Builds on every commit
 - Build once in a given execution flow
 - Deploy to a running environment for further testing
- Everything that is code (application, infrastructure, documentation) goes into a repository
 - If its not in a repository, it doesn't go into production environments!



General Best Practices used by Amazon Developers

- CI/CD is a MUST!
 - Commit frequently
 - Builds on every commit
 - Build once in a given execution flow
 - Deploy to a running environment for further testing
- Everything that is code (application, infrastructure, documentation) goes into a repository
 - If its not in a repository, it doesn't go into production environments!
- Start with continuous delivery ("gated" promotion) and build up to continuous deployment once evidence of a high-level of excellence in testing is clear



General Best Practices used by Amazon Developers

- CI/CD is a MUST!
 - Commit frequently
 - Builds on every commit
 - Build once in a given execution flow
 - Deploy to a running environment for further testing
- Everything that is code (application, infrastructure, documentation) goes into a repository
 - If its not in a repository, it doesn't go into production environments!
- Start with continuous delivery ("gated" promotion) and build up to continuous deployment once evidence of a high-level of excellence in testing is clear
- Deploy to canaries, test, deploy to an AZ, test, deploy to a Region, test



**Thousands of teams +
Microservices architectures +
Multiple environments +
Continuous delivery?**

**= 60 million deployments a year
= 1.9 deployments / second**

...but how does this help me?

Step 1: Shrink your deployments

A measure of agility

How many **deployments** am I performing?

“We have a quarterly release cycle”

“Too many to count”

How many are done **out of hours**?

“We minimize customer impact”

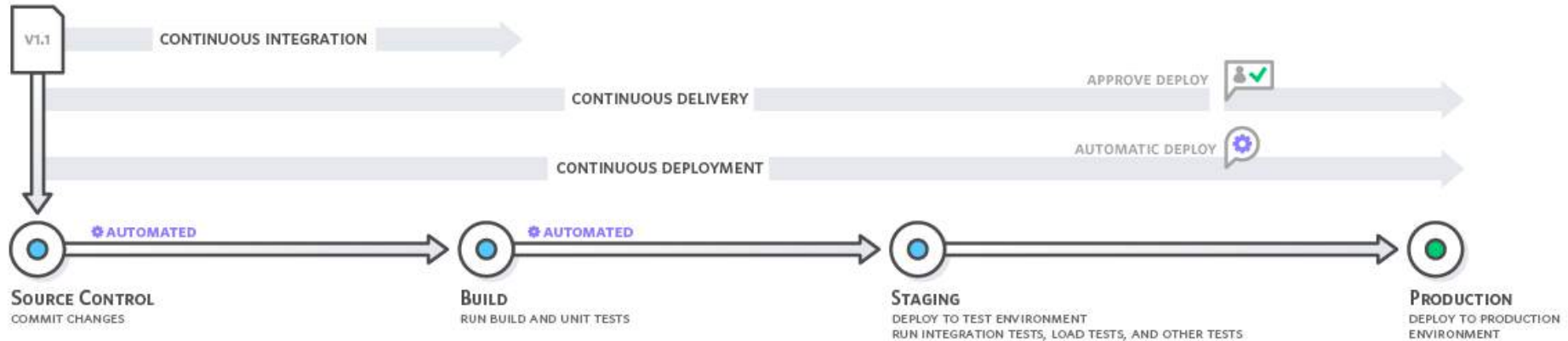
“Time is irrelevant”

How many suffer emergency **roll backs**?

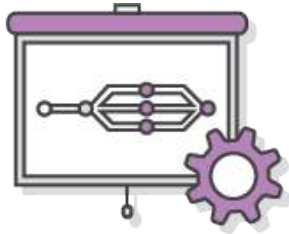
“We frequently catch problems too late and need to rollback from pre-release backups”

“We roll forwards not back”

Strive for **continuous deployment**.
Use **metrics and tooling** to gain trust.



Continuous Delivery Benefits



Automate the software
release process



Improve developer
productivity



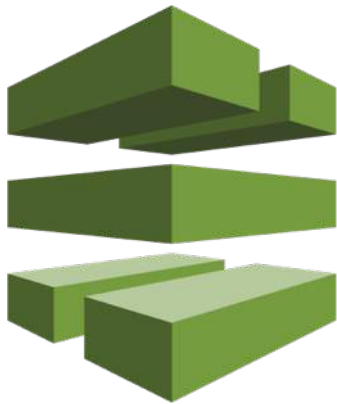
Find and address
bugs quickly



Deliver updates faster

Step 2: Improve Visibility

AWS CodePipeline

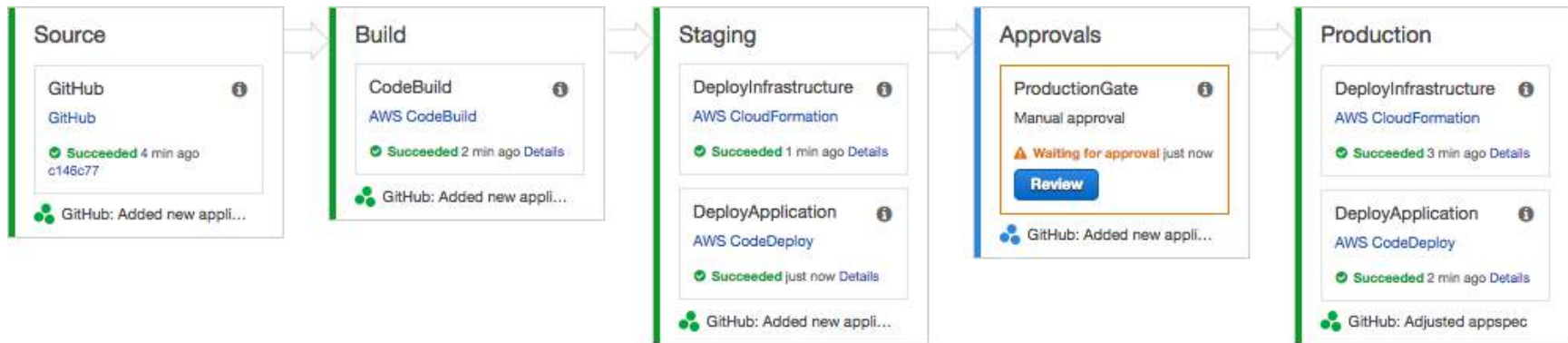


Continuous delivery service for fast and reliable application updates

Model and visualize your software release process

Builds, tests, and deploys your code every time there is a code change

Integrates with third-party tools and AWS



Step 3: Automate all of the things!

- Build & test
- Infrastructure & deployment



“This is our **build server**...

I mean, I think it is.
Someone else set it
up. They’ve left now.

Don’t break it.”

AWS CodeBuild



Fully managed build service that compiles source code, runs tests, and produces software packages

Scales continuously and processes multiple builds concurrently

You can provide **custom build environments** suited to your needs via Docker images

Pay by the minute for the compute resources you use

Integrated with AWS CodePipeline and Jenkins

Configure a Build Project

Environment: How to build

Environment image* ☒ Use an image managed by AWS CodeBuild
☐ Specify a Docker image

Operating system*

Ubuntu ▼

Runtime*

Choose a runtime environment ▼

Build specification

Base

Android

Java

Python

Ruby

Golang

Node.js

Artifacts: Where to put the artifacts from the build

```
1  version: 0.1
2
3  phases:
4
5      install:
6          commands:
7              - go get -u github.com/golang/lint/golint
8
9      pre_build:
10         commands:
11
12             # Ensure code passes all lint tests
13             - golint -set_exit_status
14
15             # Run all tests included with our application
16             - go test
17
18     build:
19         commands:
20
21             # Build our application
22             - go build -o app
23
24     artifacts:
25         files:
26             - app
27
```

buildspec.yml

- Sits in source repo alongside your project.
- Defines the commands to be run for each phase of the build, along with the output artifacts.
- Any errors will be reported back as a build failure, and the logs visible in the AWS CodeBuild console.

See build results

Phase details

	Name	Status	Duration	Completed
▶	SUBMITTED	Succeeded		Feb 25, 2017 12:04:11 AM UTC
▶	PROVISIONING	Succeeded	42 secs	Feb 25, 2017 12:04:54 AM UTC
▶	DOWNLOAD_SOURCE	Succeeded	4 secs	Feb 25, 2017 12:04:59 AM UTC
▶	INSTALL	Succeeded	21 secs	Feb 25, 2017 12:05:20 AM UTC
▶	PRE_BUILD	Succeeded	3 secs	Feb 25, 2017 12:05:23 AM UTC
▶	BUILD	Succeeded		Feb 25, 2017 12:05:24 AM UTC
▶	POST_BUILD	Succeeded		Feb 25, 2017 12:05:24 AM UTC
▶	UPLOAD_ARTIFACTS	Succeeded		Feb 25, 2017 12:05:25 AM UTC
▶	FINALIZING	Succeeded	5 secs	Feb 25, 2017 12:05:30 AM UTC
▶	COMPLETED	Succeeded		

Build logs

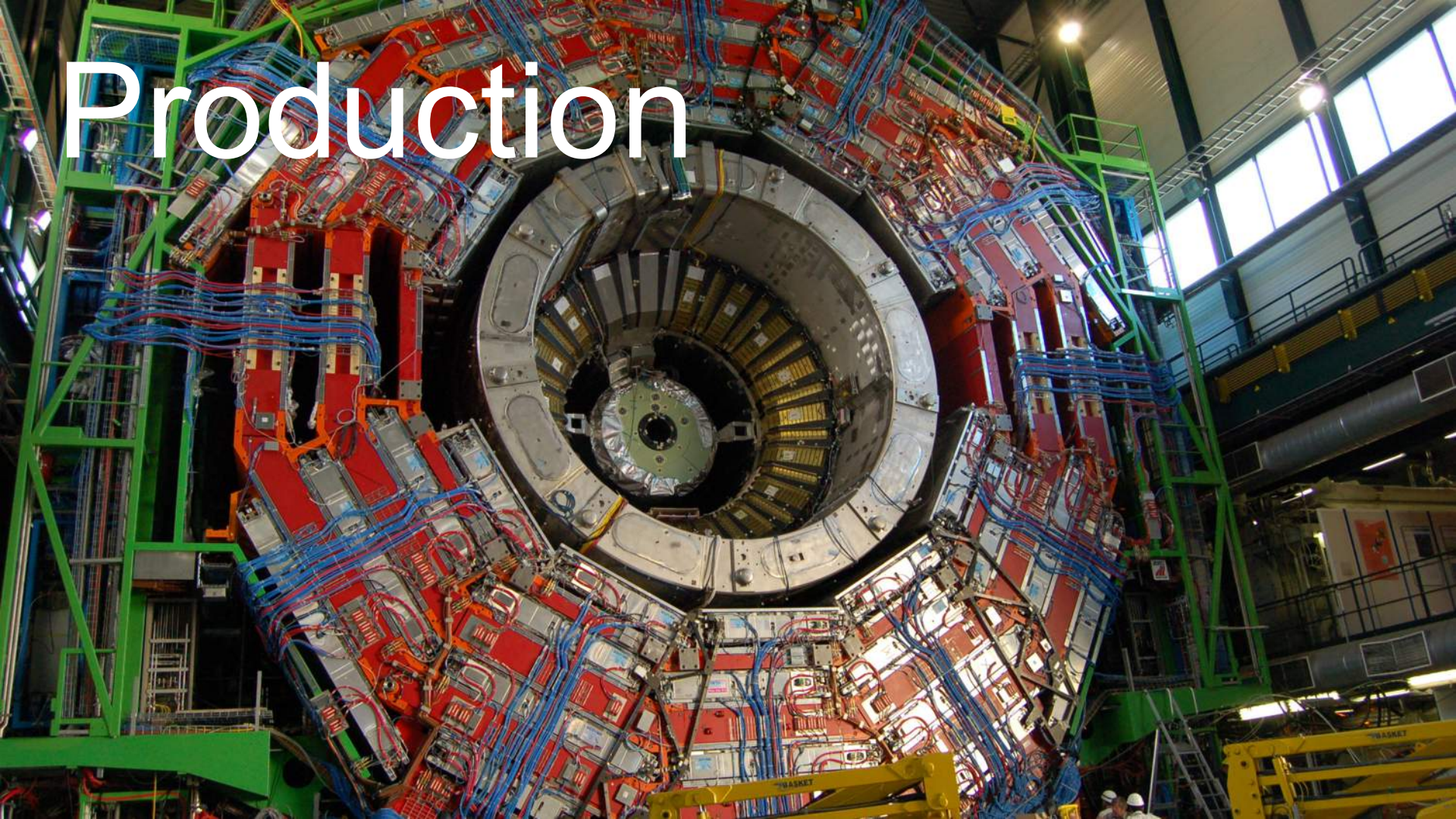
Showing the last 20 lines of build log below. [View entire log](#)

```
[Container] 2017/02/25 00:05:23 Phase context status code: Message:
[Container] 2017/02/25 00:05:23 Entering phase BUILD
[Container] 2017/02/25 00:05:23 Running command go build -o app
[Container] 2017/02/25 00:05:24 Phase complete: BUILD Success: true
[Container] 2017/02/25 00:05:24 Phase context status code: Message:
[Container] 2017/02/25 00:05:24 Preparing to copy artifacts
[Container] 2017/02/25 00:05:24 Expanding base directory path
[Container] 2017/02/25 00:05:24 Assembling file list
[Container] 2017/02/25 00:05:24 Expanding .
[Container] 2017/02/25 00:05:24 Expanding artifact file paths for base directory .
```

Step 3: Automate all of the things!

- Build & test
- Infrastructure & deployment

Production





Staging / Test

<https://www.flickr.com/photos/argonne/8569247592>

Consistent environments
build trust

AWS CloudFormation

Infrastructure-as-Code

Resources

Description:

This is an example template.
Now we've got some resources deployed!

Resources:

MyApplicationServer

Type: `AWS::EC2::Instance`

Properties:

InstanceType: `t2.micro`

ImageId: `ami-6bb2d67c`

SecurityGroups:

- `!Ref MySecurityGroup`

MySecurityGroup

Type: `AWS::EC2::SecurityGroup`

Properties:

...

**But what about deploying
applications?**

AWS CodeDeploy



Automated deployments

Deploy to Amazon EC2 and/or On-premise

Minimize downtime

Supports rolling in-place deployments, as well as blue/green

Stop and roll back

You can automatically or manually stop and roll back deployments if there are errors.

Centralized control

You can launch and track the status of your deployments through the AWS CodeDeploy console or the AWS CLI. You will receive a report that lists when each application revision was deployed and to which Amazon EC2 instances.

Easy to adopt

Supports Windows and Linux. Works with any application. Also integrates with your CI/CD tooling or AWS CodePipeline.

```
1  version: 0.0
2  os: linux
3
4  files:
5    - source: /app
6      destination: /opt
7
8  hooks:
9    BeforeInstall:
10     - location: codedeploy/BeforeInstall.sh
11    AfterInstall:
12     - location: codedeploy/AfterInstall.sh
13    ApplicationStop:
14     - location: codedeploy/ApplicationStop.sh
15    ApplicationStart:
16     - location: codedeploy/ApplicationStart.sh
17    ValidateService:
18     - location: codedeploy/ValidateService.sh
19
20
21
22
23
24
```

appspec.yml

- Sits in source repo alongside your project (similar to buildspec.yml for AWS CodeBuild)
- Specify hook scripts for each phase
- Make sure to include the validate hook. This is how AWS CodeDeploy verifies a deployment was successful.

Deployment: d-I1OIHCXNK



✔ Deployment Succeeded

Deployment progress



Installing application on your instances

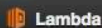
2 of 2 instances updated

Deployment details

Instance activity

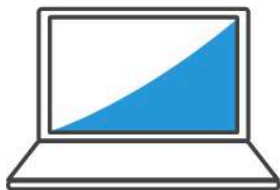
Filter Status ^						
				Instances per page	10	< Viewing 1 to 2 of 2 instances >
Instance ID	Start time	End time	Duration	Status	Most recent event	Events
i-08560c56aaef28103	Feb 25, 2017 12:07:02 AM UTC	Feb 25, 2017 12:07:11 AM UTC	9 secs	Succeeded	ValidateService	View events
i-0aa27af98b810c2a8	Feb 25, 2017 12:07:03 AM UTC	Feb 25, 2017 12:07:12 AM UTC	9 secs	Succeeded	ValidateService	View events

Putting it all together...



AWS CodeStar

AWS CodeStar lets you quickly develop, build and deploy applications on AWS.

[Start a project](#)

Create new applications

Start new software projects on AWS in minutes using templates for web applications, web services and more.



Work across your team securely

Manage project access for your team. Visualize, operate, and collaborate on your projects in one place.



Manage software delivery easily

Iterate quickly with everything you need including source, build, test, and deployment tools.

[AWS CodeStar Documentation & Support](#)

[Getting Started](#) | [Documentation](#) | [Support](#) | [Forums](#)





Select template

Set up tools

Start coding

Filter

Application category

- ☐ Web application
- ☐ Web service
- ☐ Alexa Skill
- ☐ Static Website

Programming languages

- ☐ Ruby
- ☐ Node.js
- ☐ Java
- ☐ Python
- ☐ PHP
- ☐ HTML 5

AWS services

- ☐ AWS Elastic Beanstalk
- ☐ Amazon EC2
- ☐ AWS Lambda

Choose a project template

Start a new software project on AWS in minutes using a project template. [Help me choose](#)

Ruby on Rails



Web application

AWS Elastic Beanstalk
(runs in a managed application environment)

Ruby on Rails



Web application

Amazon EC2
(runs on virtual servers that you manage)

Java Spring



Web application

AWS Elastic Beanstalk
(runs in a managed application environment)

Java Spring



Web application

Amazon EC2
(runs on virtual servers that you manage)

Node.js



Web application

AWS Lambda
(running serverless)

Node.js



Web application

AWS Elastic Beanstalk
(runs in a managed application environment)

Node.js



Web application

Amazon EC2
(runs on virtual servers that you manage)

Python (Django)



Web application

AWS Elastic Beanstalk
(runs in a managed application environment)

Python (Django)



Web application

Amazon EC2
(runs on virtual servers that you manage)

Express.js



Web application

AWS Elastic Beanstalk
(runs in a managed application environment)

Express.js



Web application

Amazon EC2
(runs on virtual servers that you manage)

PHP (Laravel)



Web application

AWS Elastic Beanstalk
(runs in a managed application environment)



Select template

Set up tools

Start coding

Project name

DemoApp

Project ID ⓘ

demoapp

[Edit](#)

AWS CodeStar includes all of the tools and services you need for a development project.
This project includes an **AWS CodePipeline** connected with the following tools:



Source



Build



Test



Deploy



Monitoring

ⓘ AWS CodeCommit

ⓘ AWS CodeBuild

ⓘ AWS CloudFormation

ⓘ Amazon CloudWatch

☒ AWS CodeStar would like permission to administer AWS resources on your behalf. [Learn more](#)[Previous](#)[Create Project](#)



Provisioning 83% complete

Choose how you want to edit your project code

You can always change this choice after the project has been created



Visual Studio

Configure the AWS Toolkit for Visual Studio to edit your AWS CodeStar project code in Microsoft Visual Studio 2015 (or higher.)



Eclipse

Configure the AWS Toolkit for Eclipse to edit your AWS CodeStar project code in Eclipse.



Command line tools

Edit AWS CodeStar project code by connecting directly to your project's Git source repository.

Clone repository URL

HTTPS ▾

`https://git-codecommit.eu-west-1.amazonaws.com/v1/repos/demoapp`

[Copy](#)

[Credential details](#)

Previous

Skip



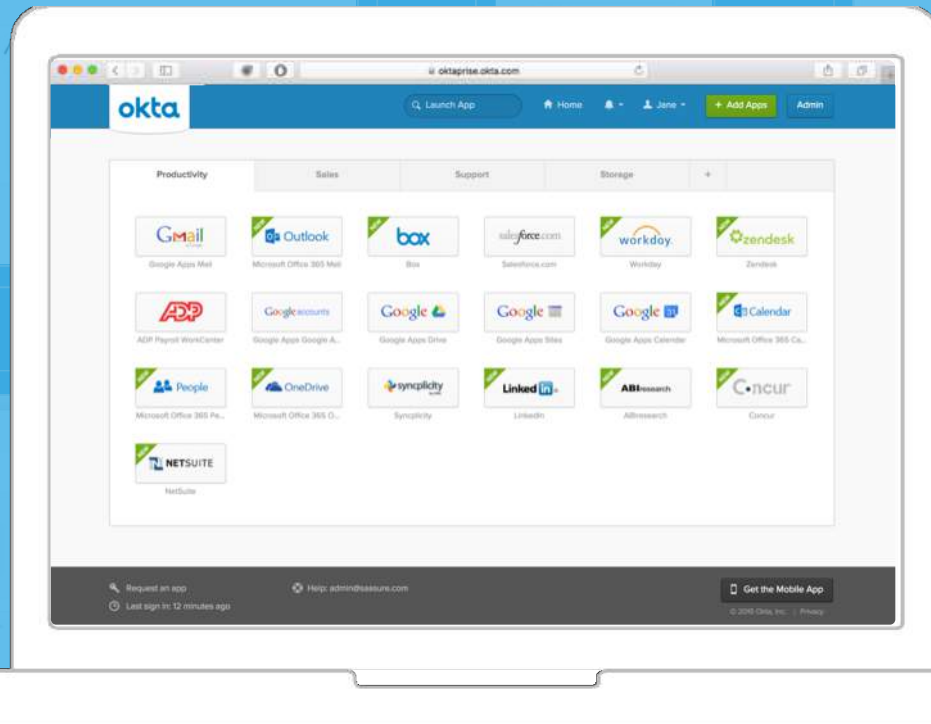
Summary

- **Innovation requires agility**. Teams busy firefighting, and large releases are the enemies of agility
- Shrink Deployments and strive for **continuous delivery**
- **Improve visibility** with AWS CodePipeline
- Automate and **scale your builds** with AWS CodeBuild
- Maintain **consistent environments** with AWS CloudFormation
- Implement **safe, zero-downtime deployments** with AWS CodeDeploy
- **Get started today** with AWS CodeStar

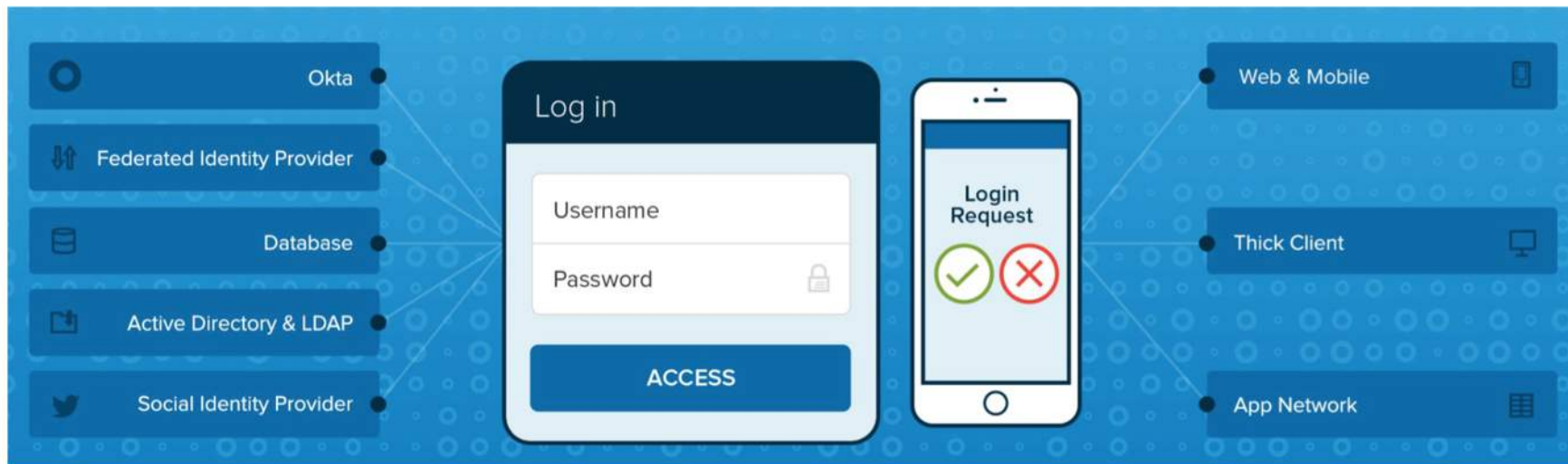
Topics

- Who is Okta
- Okta Engineering—How Do We work, how do we ship our code?
- The Challenges of the Developer Productivity Team
- A CI System with Amazon EC2 Container Service and Docker

Millions of people use Okta every day



An identity platform for developers



Learn more at: developer.okta.com

Engineering

Okta Engineering—How Do We Work, How Do We Ship Our Code?

- 300 engineers, split into teams with embedded specialists
- 1 week sprints, and deploy to production weekly
- Capability to do more than one hotfix per day
- Every merge to master is a potential release candidate

Okta Engineering—How Do We Test Our Code?

- Every topic branch goes through the same amount of rigor in testing as release candidate.
- Passing automated tests is enforced at commit time.
- Largest repo: 45K tests, takes 40 minutes (upwards of 200 parallel runs, but configurable)
- Smallest repo: 100 tests, 5 minutes
- The Developer Productivity team is responsible for supporting engineering.

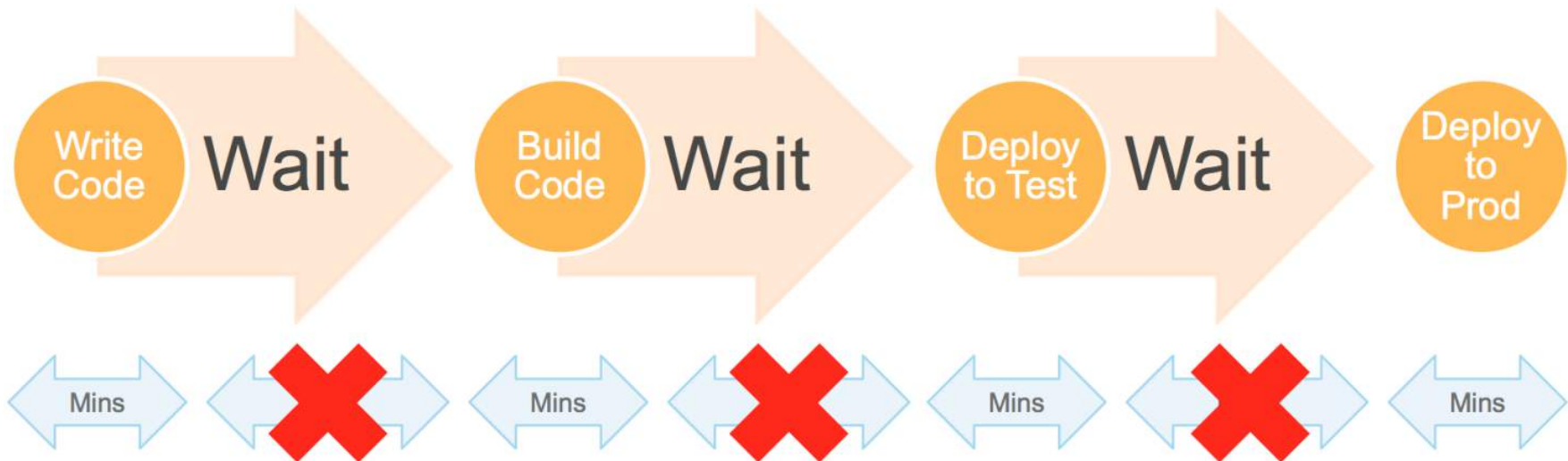
Challenges of Developer Productivity Team

- 1.Speed to Master
- 2.Cost
- 3.Managing Automated Tests



Hours

Weeks



Solutions

Bacon, Flakinizer, and Boomerang

- **Bacon** is Okta's own home-grown CI system, tailored specifically to Okta Engineering's needs.
- **Flakinizer** detects false positives in tests, and ranks tests by flakiness.
- **Boomerang** is a tool and a process to identify and fix flakey tests quickly.
- Custom Java app running with Amazon Web Services.
- Immutable and Disposable build workers in docker—created for one-time use, destroyed when job is done



master

6/16 10:37am • 183cd4c • okta-core
rlucero@okta.com



Passed ⓘ



✓ build	-- / --	16 mins
✓ components-unit	6163 / 6164	8 mins
✓ parallel-unit	13143 / 13144	17 mins
✓ serial-unit	767 / 767	7 mins
✓ js-unit	-- / 1	8 mins
✓ js-lint	-- / --	5 mins
✓ pmd	-- / --	4 mins
✓ checkstyle	-- / --	3 mins
✓ l10-check	-- / --	2 mins
✓ cycles	-- / --	2 mins
✓ feature-flag-report	-- / --	1 mins
✓ APP_PLATFORM	2363 / 2363	24 mins
✓ ADMIN_EXPERIENCE	8 / 8	7 mins
✓ ADMIN_MANAGED_TABS	9 / 9	7 mins
✓ ADMIN_DASHBOARD	63 / 63	13 mins
✓ AUTHENTICATION	976 / 977	18 mins
✓ APP_METADATA	210 / 210	18 mins
✓ ADMIN_NOTIFICATIONS	24 / 24	12 mins
✓ APP_EDITOR	216 / 216	31 mins
✓ ACTIVE_DIRECTORY	883 / 884	33 mins
✓ APPS	481 / 481	25 mins
✓ CVD	1314 / 1314	21 mins
✓ CPC	185 / 185	24 mins

as_mysqlDockerImageUpdates

6/16 10:34am • 3c0d935 • okta-core
ajinkya.suryawanshi@okta.com



Failed



✓ build	-- / --	16 mins
✓ components-unit	6082 / 6083	9 mins
✓ parallel-unit	13047 / 13048	16 mins
✓ serial-unit	712 / 712	9 mins
✓ js-unit	-- / --	7 mins
✓ js-lint	-- / --	6 mins
✓ pmd	-- / --	5 mins
✓ checkstyle	-- / --	7 mins
✓ l10-check	-- / --	2 mins
✓ cycles	-- / --	2 mins
✓ feature-flag-report	-- / --	3 mins
✗ APP_PLATFORM	1851 / 1860	41 mins
✓ ADMIN_EXPERIENCE	8 / 8	12 mins
✓ ADMIN_MANAGED_TABS	9 / 9	10 mins
✗ ADMIN_DASHBOARD	-- / 1	-1 mins
✗ AUTHENTICATION	793 / 795	38 mins
✗ APP_METADATA	208 / 208	19 mins
✗ ADMIN_NOTIFICATIONS	16 / 17	22 mins
✗ APP_EDITOR	203 / 204	42 mins
✗ ACTIVE_DIRECTORY	662 / 663	41 mins
✗ APPS	374 / 375	45 mins
✗ CVD	935 / 936	49 mins
✓ CPC	184 / 184	29 mins

nv-OKTA-128710

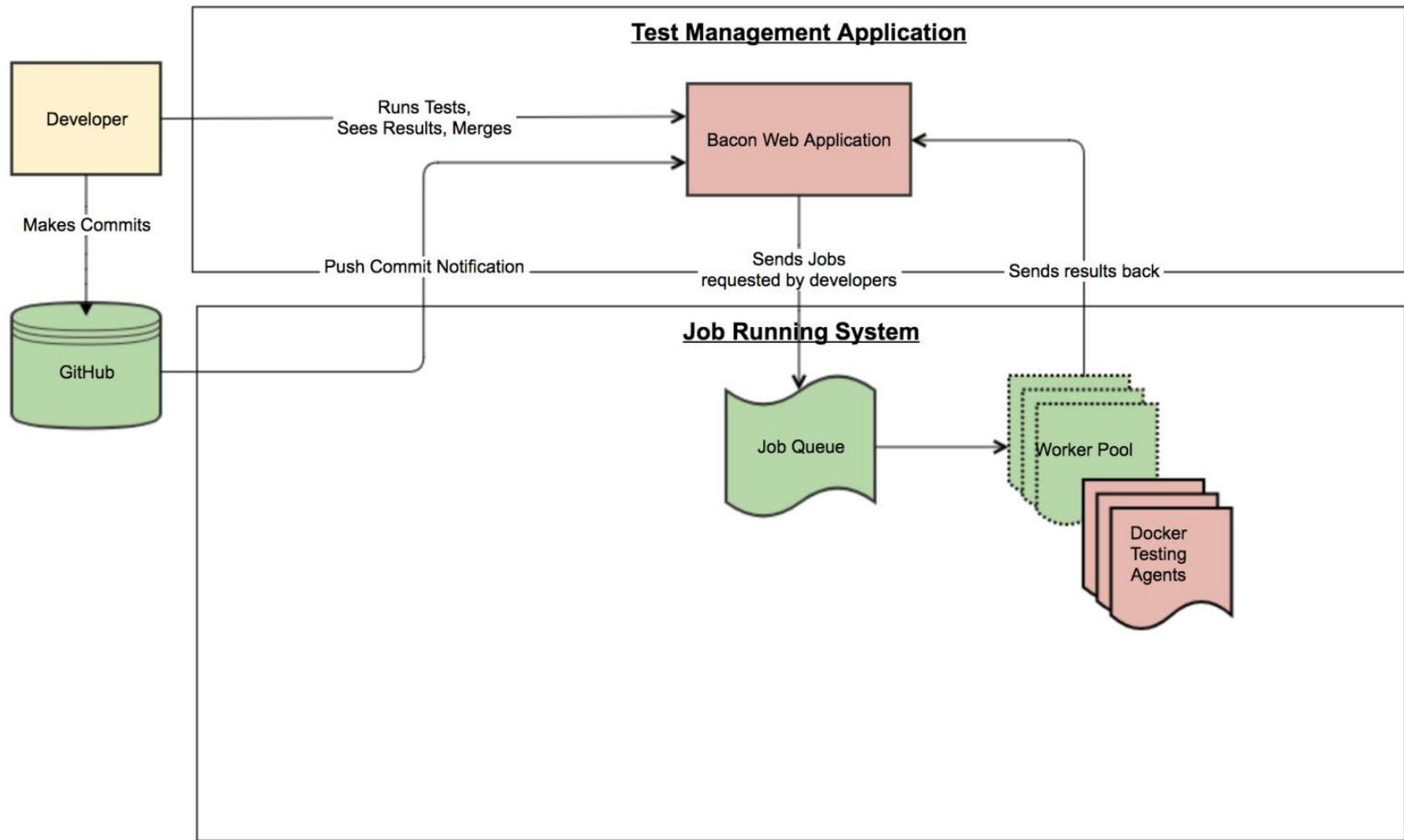
6/16 10:23am • 1b6d877 • okta-core
nikhil.venkatraman@okta.com



Passed

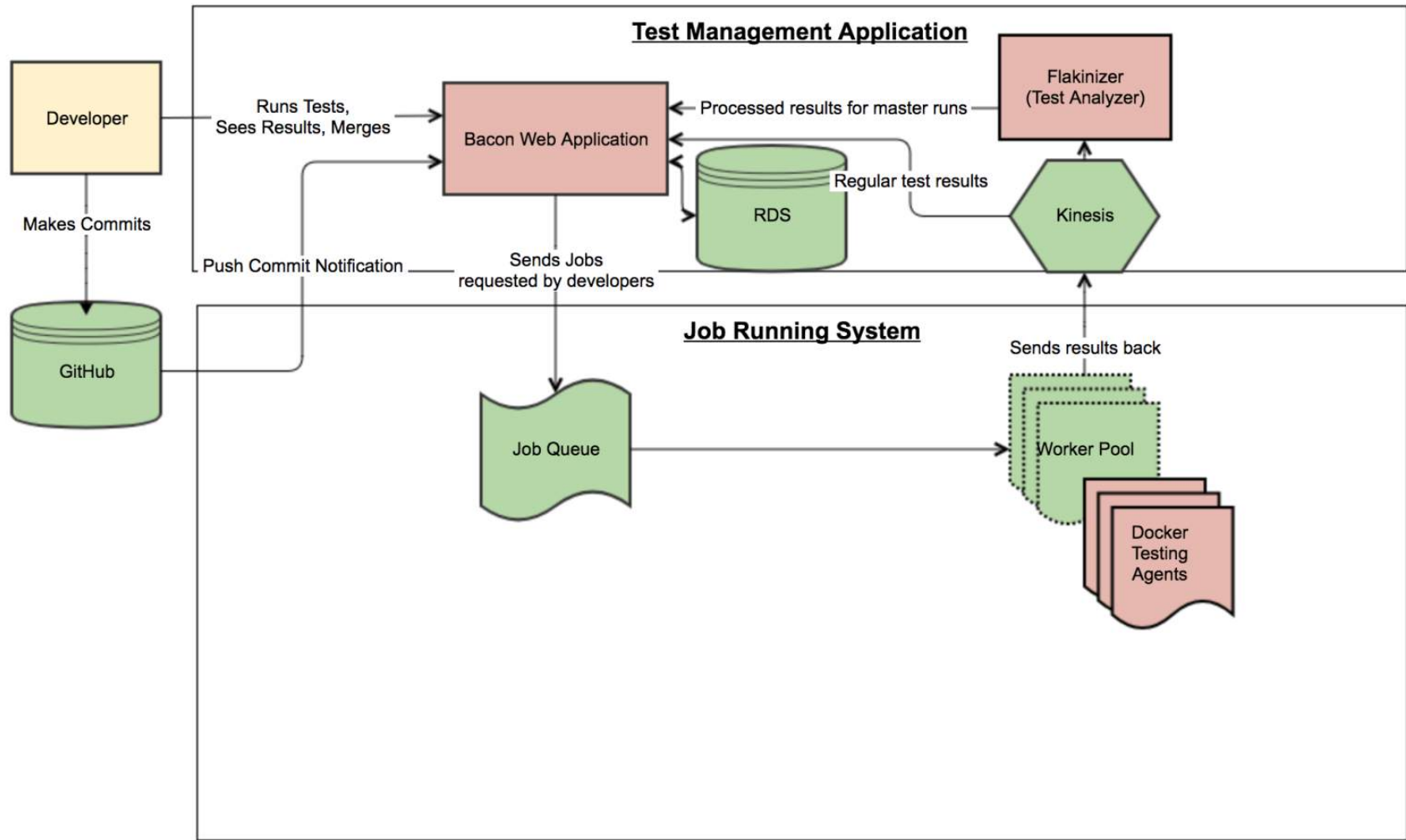


✓ build	-- / --	14 mins
✓ components-unit	6163 / 6164	8 mins
✓ parallel-unit	13143 / 13144	18 mins
✓ serial-unit	767 / 767	7 mins
✓ js-unit	-- / --	7 mins
✓ js-lint	-- / --	9 mins
✓ pmd	-- / --	4 mins
✓ checkstyle	-- / --	3 mins
✓ l10-check	-- / --	5 mins
✓ cycles	-- / --	2 mins
✓ feature-flag-report	-- / --	1 mins
✓ APP_PLATFORM	2363 / 2363	21 mins
✓ ADMIN_EXPERIENCE	8 / 8	6 mins
✓ ADMIN_MANAGED_TABS	9 / 9	8 mins
✓ ADMIN_DASHBOARD	63 / 64	9 mins
✓ AUTHENTICATION	976 / 977	20 mins
✓ APP_METADATA	210 / 210	17 mins
✓ ADMIN_NOTIFICATIONS	24 / 24	13 mins
✓ APP_EDITOR	216 / 216	28 mins
✓ ACTIVE_DIRECTORY	884 / 884	30 mins
✓ APPS	481 / 481	36 mins
✓ CVD	1314 / 1314	17 mins
✓ CPC	185 / 185	19 mins



High Level Flow











- Developer writes code and commits to github
- Bacon web application receives commit and processes into test runs
- Test runs inserted as messages to an SQS Queue
- Docker containers on slave hosts pick up messages from SQS queue and run them
- Test results reported back to Bacon web application

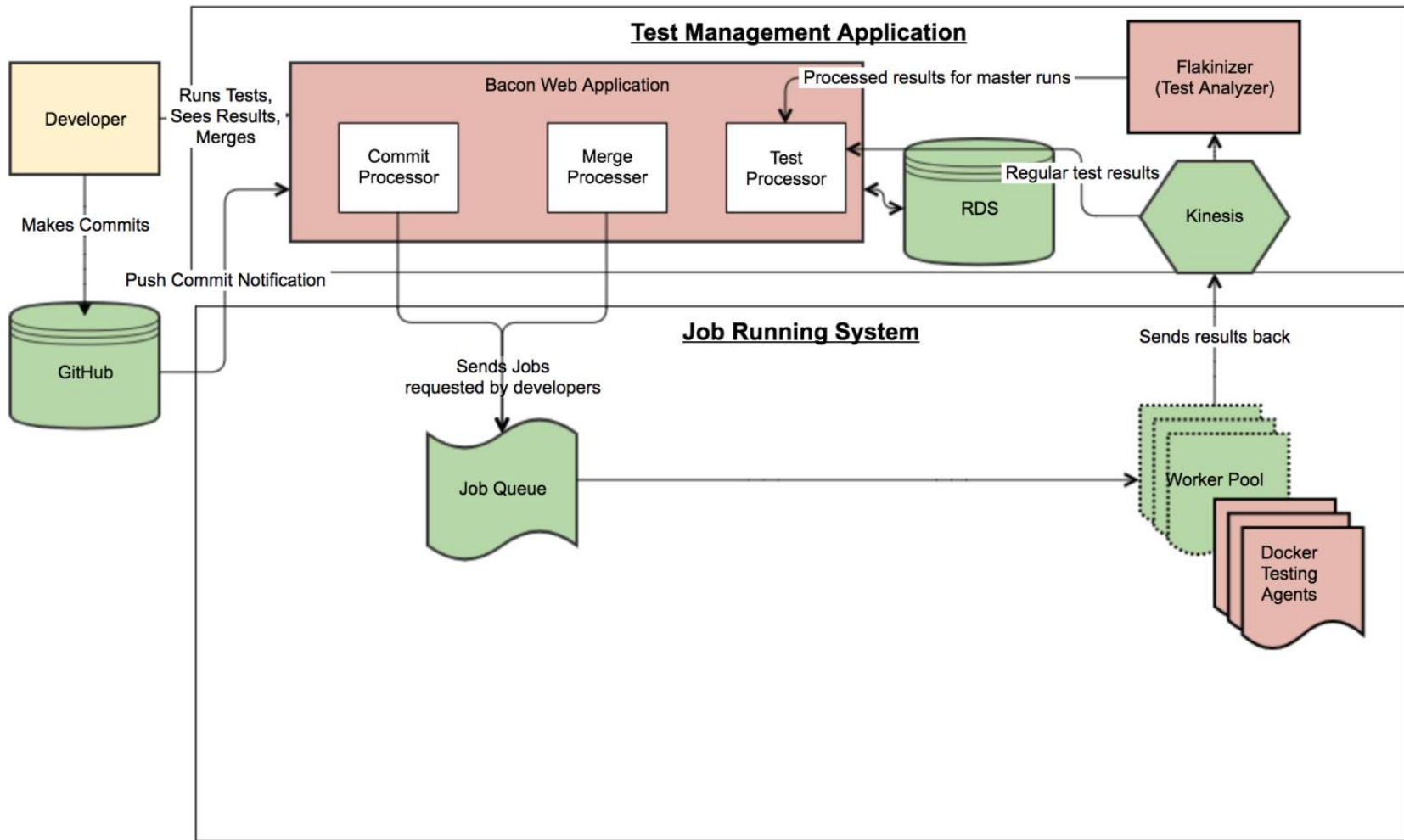


Test Result Processing

- Results from all workers dumped into a Kinesis stream to allow for robustness against server failures
- All results inserted into Bacon for display to developers
- Master branch runs also piped through Flakinizer, where pass/fail ratio calculated for each individual test
- On display of test results for topic branches, still show that flakey tests failed, but do not block ability to merge if only tests failed are flakey ones

com.okta.webdriver.tests.apps.google.ConfigureGoogleUserManagementWD

Test	Data	Recent Avg	Long Avg	Owner	Test Suite	Ignored	Type	Jira
com.okta.webdriver.tests.apps.google.ConfigureGoogleUserManagementWD#setUpBaseBrowserTestCase		1	1	Team: OAN Advanced Apps	TOP_APPS		SELENIUM	+
com.okta.webdriver.tests.apps.google.ConfigureGoogleUserManagementWD#setUpBaseWebDriverTestCase		1	1	Team: OAN Advanced Apps	TOP_APPS		SELENIUM	+
com.okta.webdriver.tests.apps.google.ConfigureGoogleUserManagementWD#tearDownBaseBrowserTestCase		1	1	Team: OAN Advanced Apps	TOP_APPS		SELENIUM	+
com.okta.webdriver.tests.apps.google.ConfigureGoogleUserManagementWD#tearDownBaseWebDriverTestCase		1	1	Team: OAN Advanced Apps	TOP_APPS		SELENIUM	+
com.okta.webdriver.tests.apps.google.ConfigureGoogleUserManagementWD#setUpRunAsDynamicOrgWebDriverTestCase		1	1	Team: OAN Advanced Apps	TOP_APPS		SELENIUM	+
com.okta.webdriver.tests.apps.google.ConfigureGoogleUserManagementWD#setUpAbstractRunAsOrgWebDriverTestCase		1	1	Team: OAN Advanced Apps	TOP_APPS		SELENIUM	+
com.okta.webdriver.tests.apps.google.ConfigureGoogleUserManagementWD#tearDownRunAsDynamicOrgWebDriverTestCase		1	1	Team: OAN Advanced Apps	TOP_APPS		SELENIUM	+
com.okta.webdriver.tests.apps.google.ConfigureGoogleUserManagementWD#tearDownAbstractRunAsOrgWebDriverTestCase		1	1	Team: OAN Advanced Apps	TOP_APPS		SELENIUM	+
com.okta.webdriver.tests.apps.google.ConfigureGoogleUserManagementWD#setUp		0.86	0.89	Team: OAN Advanced Apps	TOP_APPS		SELENIUM	+
com.okta.webdriver.tests.apps.google.ConfigureGoogleUserManagementWD#testGoogleUserManagement		0.86	0.89	Team: OAN Advanced Apps	TOP_APPS		SELENIUM	+



Parallelized Test Processing

- Configurable parallelized test processing--based on desired speed, cost, branch name, time of day, etc.
 - e.g. hot fixes need to be done <30 min
 - weekends builds can run slower, at 90 minutes
- Tests are split into suites on individual build workers (the faster the speed we need, the more parallelized suites).
- If tests failed, selective retry of only the tests that failed

master

6/22 12:08pm • 1b091e5 • okta-core
frank.montanaro@okta.com



Passed 



✓ DEVICE_TRUST	13 / 13	10 mins
✓ DB_MIGRATIONS	1 / 1	6 mins
✓ DOMAINS	3 / 3	10 mins
✓ DOWNLOAD_PAGE	19 / 19	17 mins
✓ EMAIL	53 / 53	17 mins
✓ ENCRYPTION	43 / 43	22 mins
✓ END_USER_APP	417 / 417	24 mins
✓ EMAIL_CUSTOMIZATION	53 / 53	12 mins
✓ FRAMEWORK	766 / 766	22 mins
✓ FEATURES	189 / 189	23 mins
✓ FILE_STORE	17 / 17	14 mins
✓ GROUPS	812 / 812	32 mins
✓ GROUP_PUSH	290 / 290	29 mins
✓ GETTING_STARTED	4 / 4	11 mins
✓ IWA	29 / 29	27 mins
✓ IN_PRODUCT_HELP	21 / 21	11 mins
✓ IAA	28 / 28	15 mins
✓ ILM	374 / 374	31 mins
✓ IDP	736 / 736	44 mins
✓ JOBS	105 / 105	25 mins
✓ KEYSTORE	6 / 6	10 mins
✓ LOGGING	11 / 11	16 mins

Brought to you with <3 by eng_productivity@okta.com

Merge Processing

- At merge, display to users tests that failed for their commit
- Allow bypass of failed tests, but keep track of who bypassed what
- If tests fail in following master runs, 4 hour SLA for developer who bypassed tests to fix problem
- Merging code is just another type of job sent to queues which can only have one run going at a time

#NotMyFault:



Following tests have **failed**:

	Test Suite	Test	Product Area	NMF/week	History
❶	FUNCTIONAL-SERIAL	com.okta.api.internal.mediation.radius.RadiusAgentMediationServiceFuncTest#checkRadiusMFANetworkCondition	AUTHENTICATION	0	
❶	SELENIUM-PARALLEL	com.okta.webdriver.tests.platform.admin.devices.policies.OMMEnrollmentPolicyPageTest#testBasicAndroidRuleValidation	MOBILE	0	
❶	SELENIUM-PARALLEL	com.okta.webdriver.tests.enduser.EnduserDevicesUITest#testDevicesIsEmptyWhenNoDevicesAdded	MOBILE	0	

☐ I hereby solemnly swear that none of these failures are my fault.

Continue to Merge

Boomerang Summary Report: 16 Jun 2017 15:00 to 22 Jun 2017 17:15

+ Set Time Range

Distinct Tests Skipped

Users

Tests Skipped By Product Area

Classes Skipped By Product Area

Tests Skipped Only Once

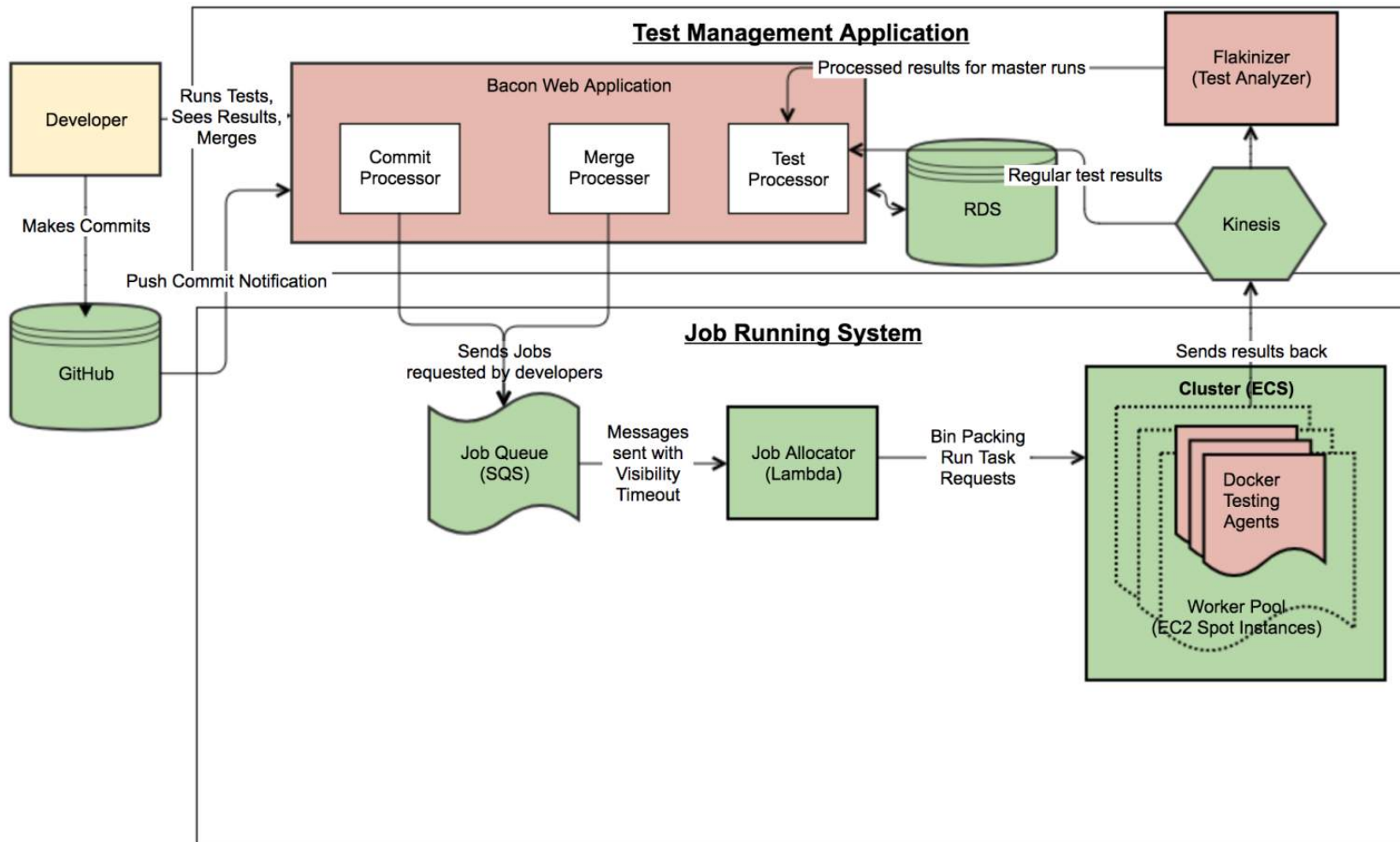
Tests Skipped in over 50% of SHAs

Tests not skipped in previous period

Tests not skipped in previous 10 periods

Distinct Tests Skipped

Test Class	Test Method	Product Area	Team	Status in Master	Times Skipped
com.okta.client.api.internal.su.ImplicitAssignmentSuClientTest	testToggleImplicitAssignment	SYNC	eng_sync	Flakey	8
com.okta.webdriver.tests.cvd.UDGroupAppAssignmentTest	testAddAppWithoutUserManagementThenCreateUsers	CVD	eng_udcore	Passed	1
com.okta.webdriver.tests.cvd.UDGroupAppAssignmentTest	testAssignBookmarkAppGAA	CVD	eng_udcore	Passed	1
com.okta.webdriver.tests.cvd.UDGroupAppAssignmentTest	testAssignSWAAppGAA	CVD	eng_udcore	Passed	1
com.okta.webdriver.tests.cvd.UDGroupAppAssignmentTest	testAssignUMAppGAA	CVD	eng_udcore	Passed	1
com.okta.webdriver.tests.cvd.UDGroupAppAssignmentTest	testGAAUMAppNotSavingEdits	CVD	eng_udcore	Passed	1
com.okta.webdriver.tests.cvd.UDGroupAppAssignmentTest	testGAAUMAppSavingEdits	CVD	eng_udcore	Passed	1
com.okta.webdriver.tests.enduser.NewEndUserUIAppsTest	testAppSettingsExternalSync	END_USER_APP	eng_up	Flakey	1



From Queues to Workers

- Messages on SQS Queue picked up by Lambda
- Pool of EC2 hosts in ECS cluster associated with queue
- Lambda uses bin packing algorithm to run tasks on ECS cluster to minimize number of workers used (enables scaling down)
- Visibility timeouts used on SQS queue to guarantee each run will be completed at least once (allows for use of spot instances)

<input type="checkbox"/>	Name	Queue Type	Content-Based Deduplication	Messages Available	Messages in Flight	Created
<input type="checkbox"/>	ci-queue-productionJenga-API-Linux-S	Standard	N/A	0	137	2017-05-12 15:46:47 GMT-07:00
<input checked="" type="checkbox"/>	ci-queue-productionJenga-Func-Linux-S	Standard	N/A	0	777	2017-05-12 16:02:51 GMT-07:00
<input type="checkbox"/>	ci-queue-productionJenga-Okta-Rum	Standard	N/A	0	0	2017-05-31 03:10:23 GMT-07:00
<input type="checkbox"/>	ci-queue-productionJenga-Releng	Standard	N/A	0	0	2016-09-30 14:56:41 GMT-07:00
<input type="checkbox"/>	ci-queue-productionJenga-Selenium	Standard	N/A	0	17	2016-05-27 14:26:30 GMT-07:00
<input type="checkbox"/>	ci-queue-productionJenga-Selenium-Linux-M	Standard	N/A	0	124	2017-05-12 15:51:43 GMT-07:00
<input type="checkbox"/>	ci-queue-productionJenga-Selenium-Small	Standard	N/A	0	0	2016-11-01 16:53:44 GMT-07:00
<input type="checkbox"/>	ci-queue-productionJenga-Topic-Check-In	Standard	N/A	0	0	2016-09-07 17:11:29 GMT-07:00

Cluster : productionJenga-Func-Linux-S-NodeCluster-1GYDS1Q4Q2GOL

Delete Cluster

Get a detailed view of the resources on your cluster.

Status **ACTIVE**

Registered container instances 746

Pending tasks count 2

Running tasks count 642

Services

Tasks

ECS Instances

Metrics

Scheduled Tasks

Run new Task

Stop

Stop All

Last updated on June 16, 2017 10:18:49 AM (more than 1 hour ago)

↺

?

Desired task status:

Running

Stopped

Filter in this page

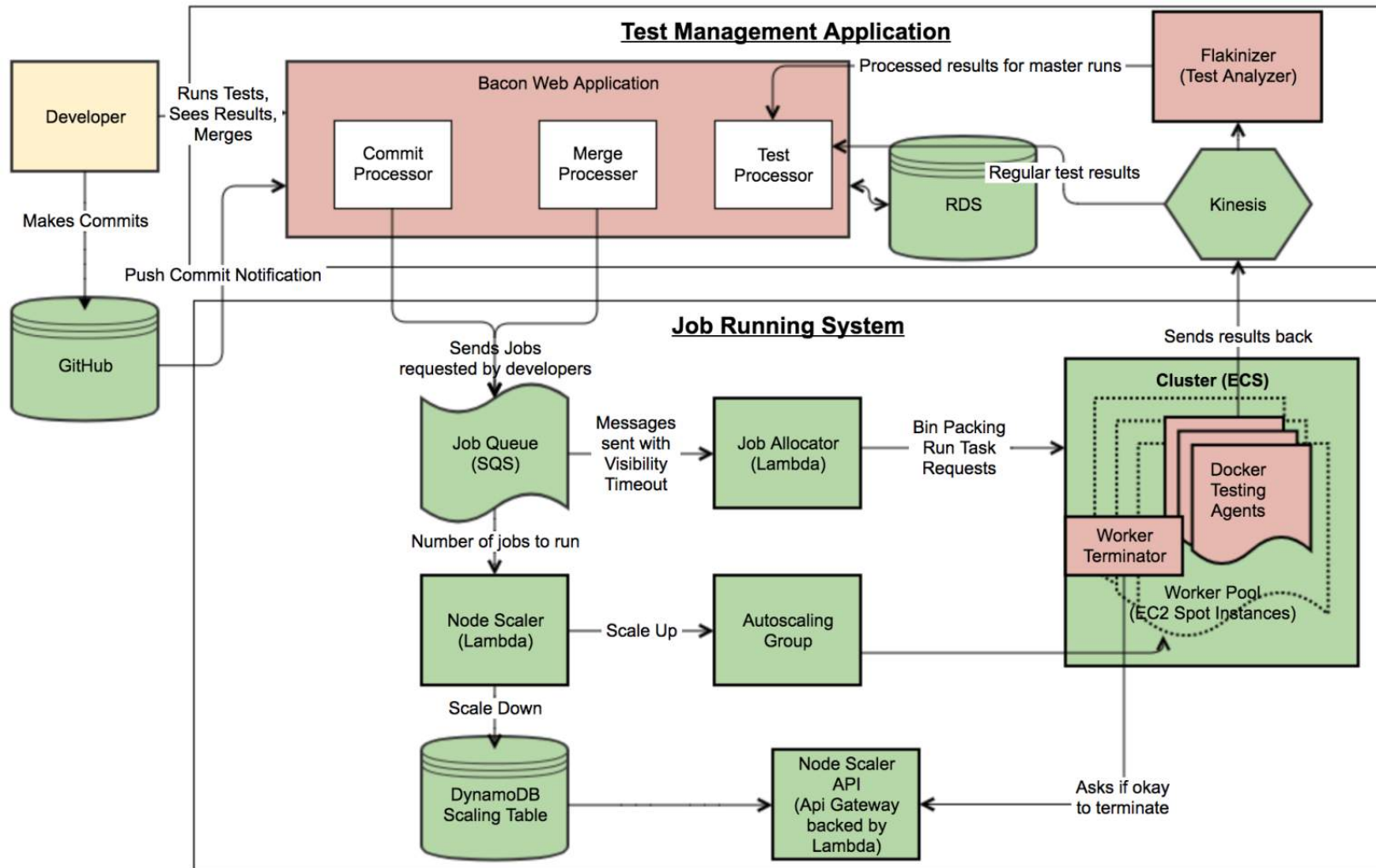
< 1-50 >

Page size

50

▼

<input type="checkbox"/>	Task	Task Definition	Container Instance	Last status	Desired status	Started By	Group
<input type="checkbox"/>	00248d80-6dad-441...	productionJenga-Fu...	e0084e6c-9e40-4b1...	RUNNING	RUNNING		family:productionJe...
<input type="checkbox"/>	00e3684b-2c54-44a...	productionJenga-Fu...	b5e5a8b4-d719-469...	RUNNING	RUNNING		family:productionJe...
<input type="checkbox"/>	019790c8-207c-4ef...	productionJenga-Fu...	283b84e1-03cd-441...	RUNNING	RUNNING		family:productionJe...

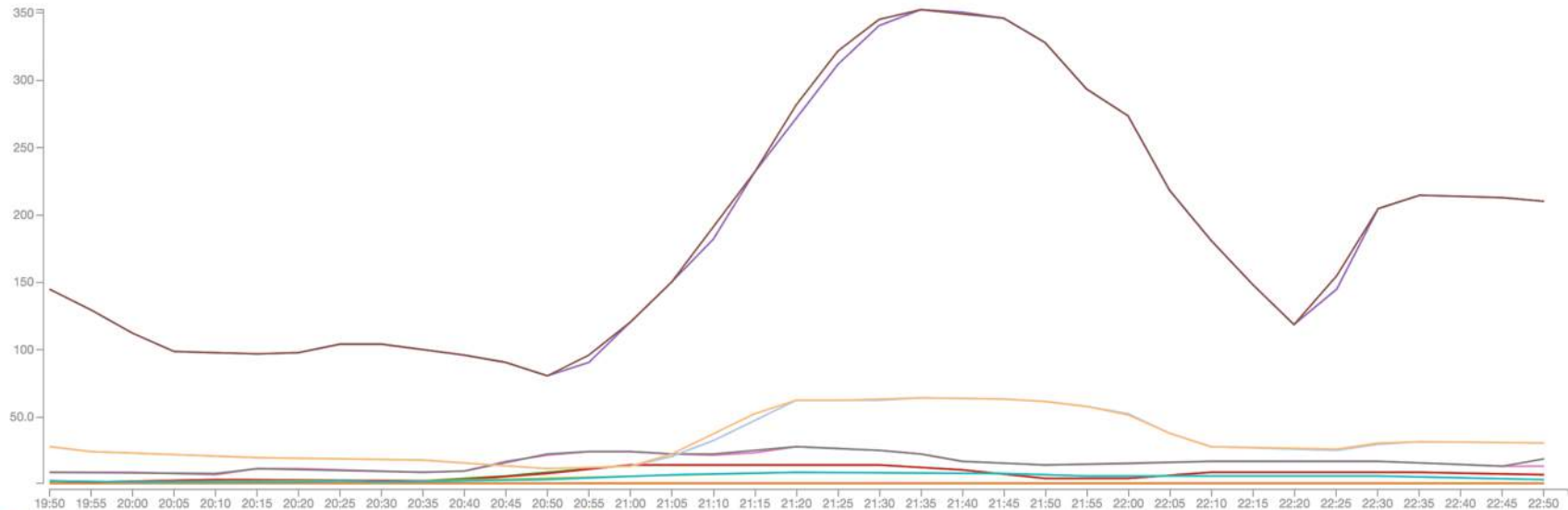


Scaling Our Worker Count

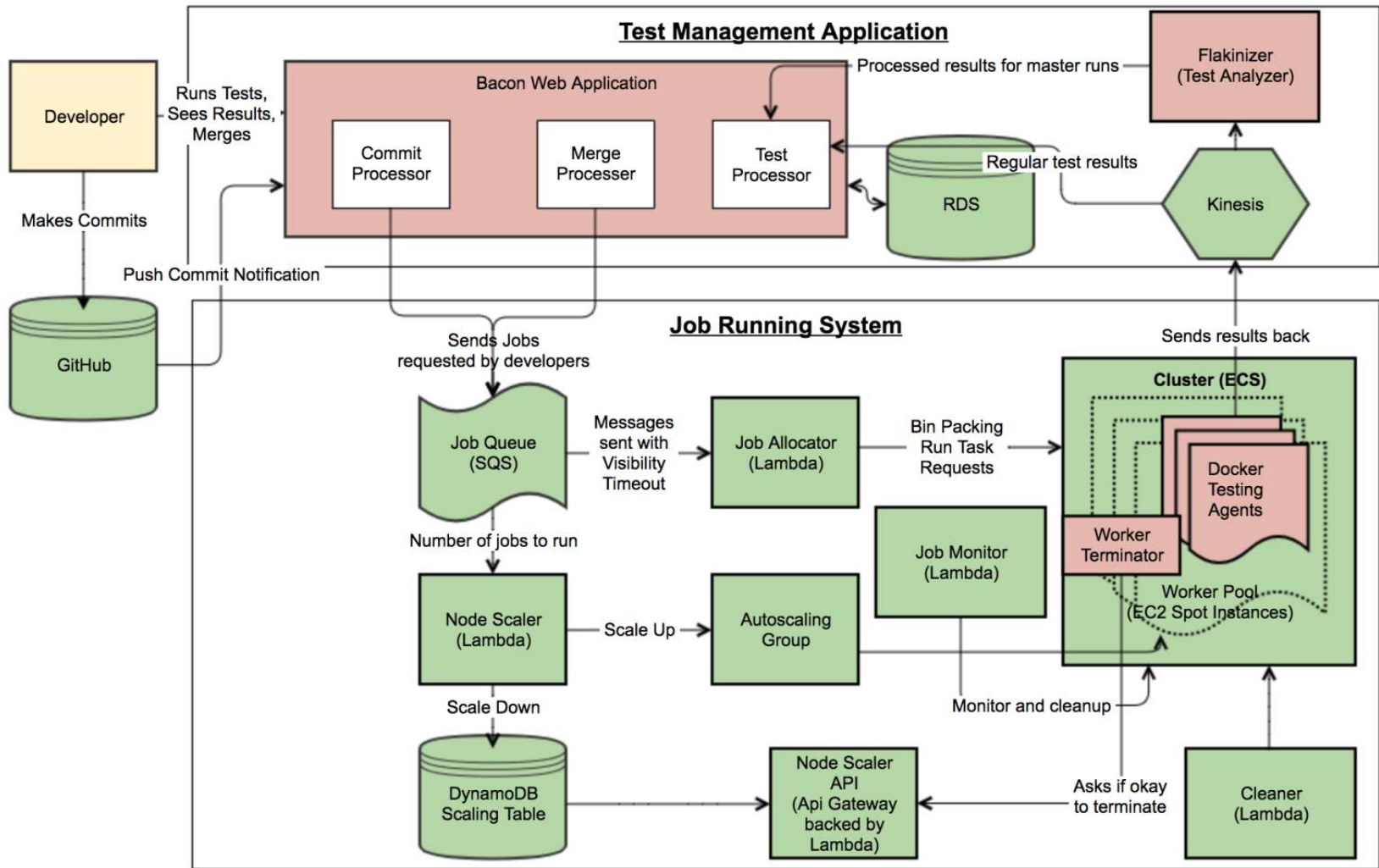
- Lambda function uses number of messages and resource usage to calculate number of workers
- If scale up, simple call to ASG for more workers
- Increase to number of workers purposely slowed down slightly to avoid overshooting (cost savings)
- Also have special “lazy” queues that we can send our messages to which will not scale up more workers (cost savings)

Scaling Our Worker Count

- If scale down, store desired number in DynamoDB
- Each worker runs a termination application that checks for idleness and nearness to billing cycle, calls API if conditions match
- API handled by API Gateway backed by a Lambda function that checks desired number in DynamoDB and returns true or false based on that
- API based system allows for scale up to large amounts of workers without hitting AWS API call rate limits

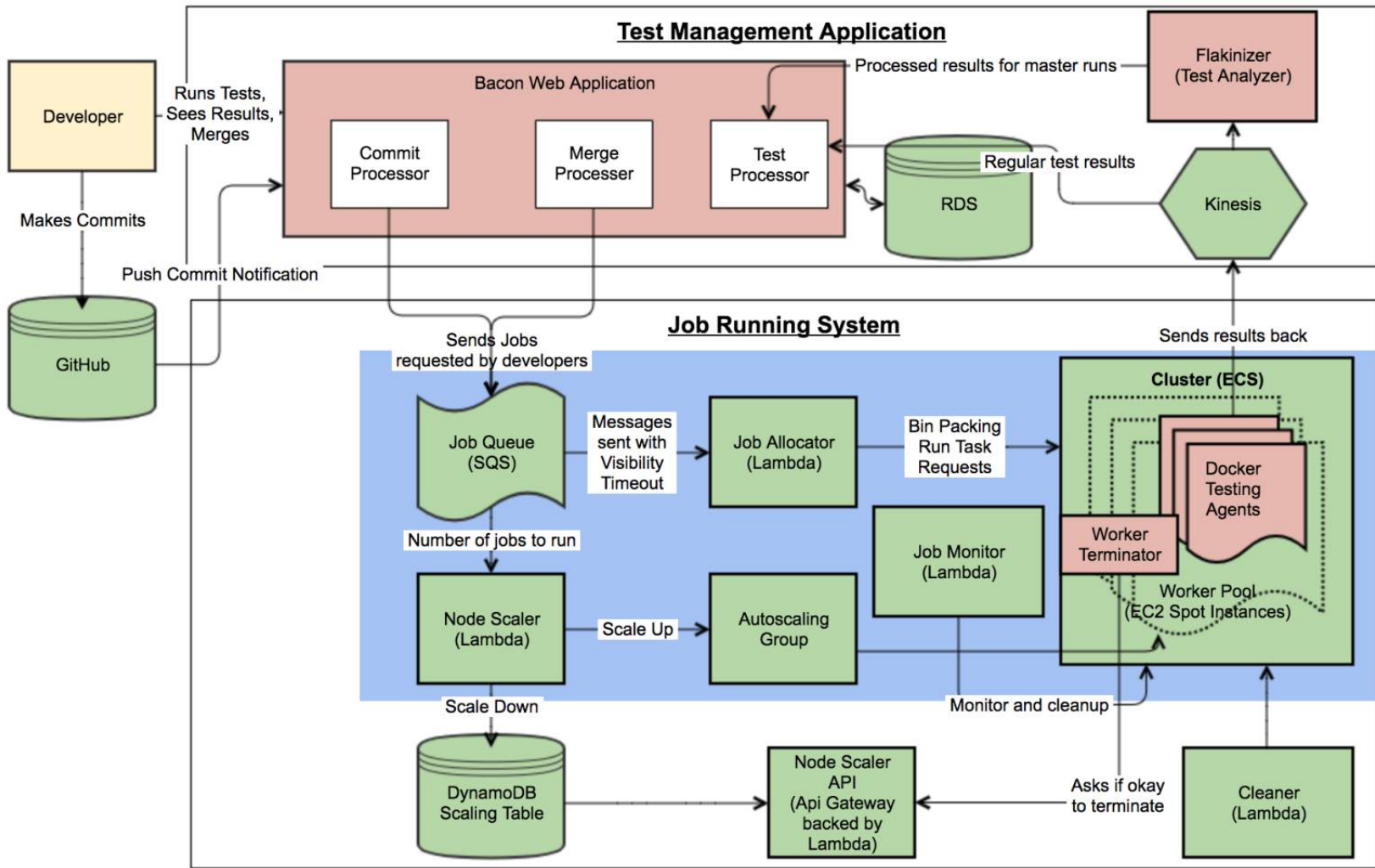


- Jenga-API-c32xlarge-v1-AutoScalingGroup-YQ0ONG9O0ESZ GroupTotalInstances
- Jenga-API-c32xlarge-v1-AutoScalingGroup-YQ0ONG9O0ESZ GroupDesiredCapacity
- Jenga-Build-m32xlarge-v1-AutoScalingGroup-13RWWYBSJX4H6 GroupDesiredCapacity
- Jenga-Build-m32xlarge-v1-AutoScalingGroup-13RWWYBSJX4H6 GroupTotalInstances
- Jenga-Functional-AutoScalingGroup-G2JYCT1BDPVJ GroupTotalInstances
- Jenga-Functional-AutoScalingGroup-G2JYCT1BDPVJ GroupDesiredCapacity
- Jenga-Unit-v3-AutoScalingGroup-1LOJH0Z7BXKKW GroupTotalInstances
- Jenga-Unit-v3-AutoScalingGroup-1LOJH0Z7BXKKW GroupDesiredCapacity
- Jenga-Unit-v4-Parallel-Safe-AutoScalingGroup-110G2D3SGJTXU GroupTotalInstances
- Jenga-Unit-v4-Parallel-Safe-AutoScalingGroup-110G2D3SGJTXU GroupDesiredCapacity
- productionJenga-Selenium-AutoScalingGroup-1LZWSWAKK9CAK GroupTotalInstances
- productionJenga-Selenium-AutoScalingGroup-1LZWSWAKK9CAK GroupDesiredCapacity



Cleanup and Monitoring

- Splunk logging for each of our applications
- Cloudwatch gives visibility into lambda run logs and worker/queue performance
- Job monitor lambda function that monitors cluster for tasks stuck in pending too long (problem for visibility timeout) and ones alive too long (stuck)
- Cleaner lambda function that kills EC2 worker hosts that live too long and ensures parity between ASG and ECS



Specialized Queues

- All of the infrastructure behind a queue has a cloudformation template
- If need a specialized kind of worker for running jobs on, as simple as spinning up a new cloudformation stack
- Our application just needs to send messages to new queue and stack resources will scale to process it

Amazon Web Services



EC2 Container Service



Lambda



Simple Storage Service



Relational Database Service



Elastic Compute Cloud



CloudFormation



CloudWatch



EC2 Container Registry



EC2 Spot Instances



CloudTrail



Simple Notification Service



Kinesis



Simple Queue Service

Future

Expand Use

- Use EC2 Container Service for more services
- Allow Developers to control their test suites and Docker images more directly
- Developer Environments
 - Aim to enable running CI containers right out of the box

Result: Happy Engineering Team

- Developers can write more tests quicker.
- Happy devs, timely build/test status feedback.
- Happy quality team, all tests are run at each commit.
- Happy ops team, release candidate produced quickly.
- Happy management, infra budget is under control.

Thank You

stackshare.io/okta/okta



okta

Join us @Okta - www.okta.com/company/careers/



AWS

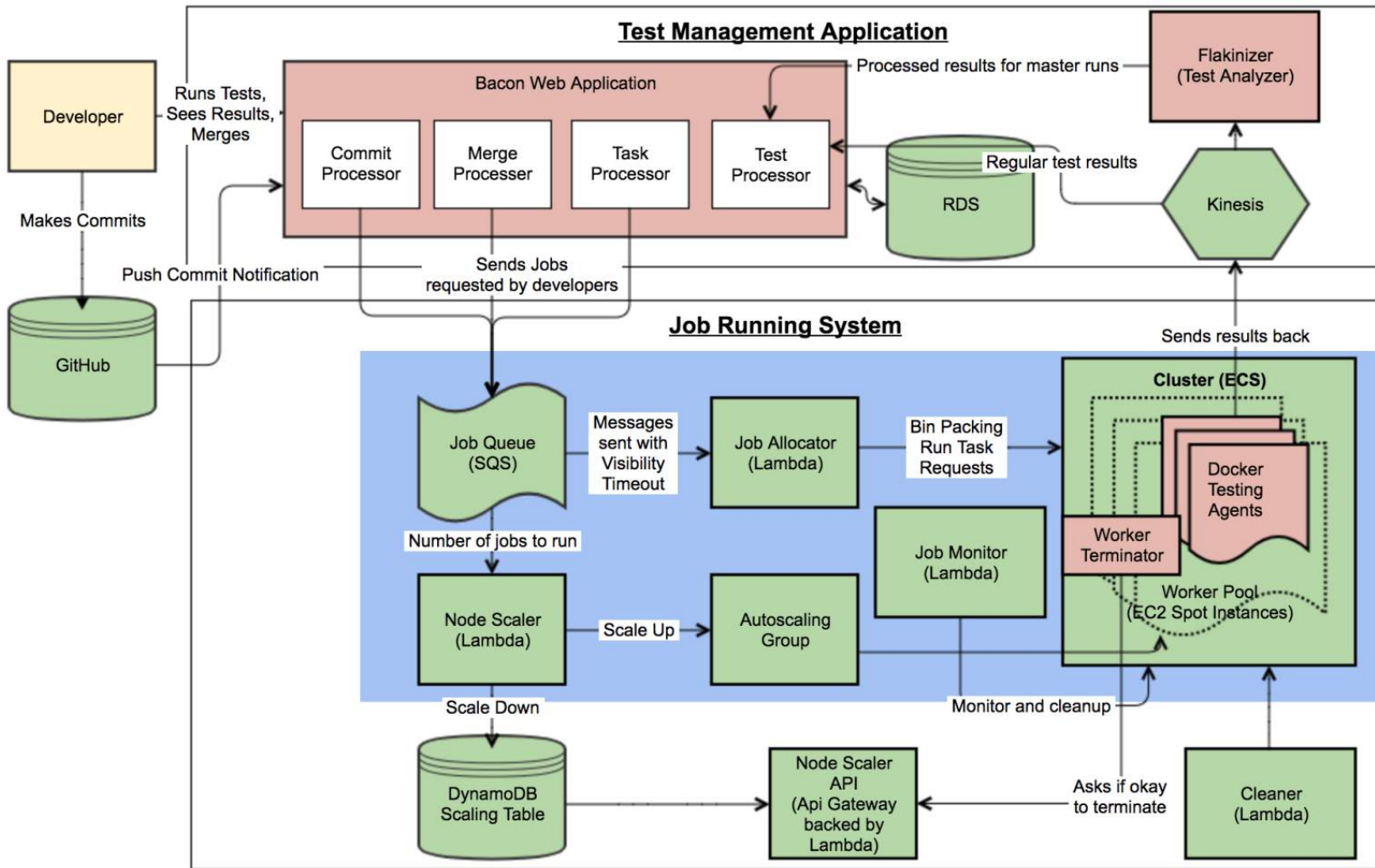
S U M M I T





**Remember to complete
your evaluations!**

Appendix



Task Processing

- Some jobs don't fall under test suites (deploys, promotions, etc)
- Require ability to rerun for same parameters even if successful (i.e. deploy rollback)
- Separate configuration, display, and running system for generating messages to the queues for running

<div> <div>Run Task</div> <div>Create New Task Type</div> <div>Edit Task Type</div> <div>Q Search...</div> </div>											
<div>FILTERS</div> <div>ANSIBLE_TOWER</div> <div>APERTURE_VPC_DEPLOYMENT</div> <div>API_AUTO_TESTS</div> <div>ARMAGEDDON</div> <div>A_TRIGGER_NIR</div> <div>BACON_CREATE_DEPLOY_ENV</div> <div>BACON_DEPLOY_TO_ENVIRO...</div> <div>BACON_DESTROY_DEPLOY_E...</div> <div>Blah Test</div> <div>CHECK_PKI</div> <div>COMPARE_SCREEN</div> <div>CREATE_APERTURE_TEST_CL...</div> <div>CREDZILLA_DEPLOY</div>	User	Sha	Task ID	Status	Result	Tests Fail/Pass/Run	Schedule	Error Message	Created time	Duration	
	ugur.aktepe	ea9fcc9	cfe00b1	FINISHED	✓	-- / -- / --		NONE	06/16/2017 6:32:31	2m 58s	Action ▾
	alex.pletnev	b26bd69	7d1beaf	FINISHED	✓	-- / -- / --		NONE	06/14/2017 11:04:9	3m 15s	Action ▾
	alex.pletnev	9a5d190	c97294c	FINISHED	✓	-- / -- / --		NONE	06/14/2017 10:45:49	2m 39s	Action ▾
	alex.pletnev	e2db64f	1c67f42	FINISHED	✓	-- / -- / --		NONE	06/14/2017 7:03:19	2m 49s	Action ▾
	alex.pletnev	fb20172	636ee86	FINISHED	✓	-- / -- / --		NONE	06/14/2017 6:08:13	2m 56s	Action ▾
	alex.pletnev	be4003a	1307b29	FINISHED	✓	-- / -- / --		NONE	06/13/2017 10:15:18	3m 29s	Action ▾
	alex.pletnev	86dbd34	740331b	FINISHED	✓	-- / -- / --		NONE	06/12/2017 12:03:13	2m 44s	Action ▾
	tsecor	3dfbf32	bd2197b	FINISHED	✗	1 / 0 / 1		BUILD_FAILURE	06/12/2017 10:21:39	2m 31s	Action ▾
	alex.pletnev	4bf18d4	93edcfa	FINISHED	✗	1 / 0 / 1		BUILD_FAILURE	06/12/2017 9:05:20	2m 26s	Action ▾
	alex.pletnev	08bdd25	ec4553d	FINISHED	✓	-- / -- / --		NONE	06/09/2017 10:20:8	2m 43s	Action ▾
	alex.pletnev	664a948	9cd991d	FINISHED	✓	-- / -- / --		NONE	06/09/2017 9:57:25	3m 5s	Action ▾
	alex.pletnev	625317d	aef0049	FINISHED	✓	-- / -- / --		NONE	06/08/2017 10:22:1	3m 3s	Action ▾



Thank you.

Paul Maddox, AWS
Developer Technologies