



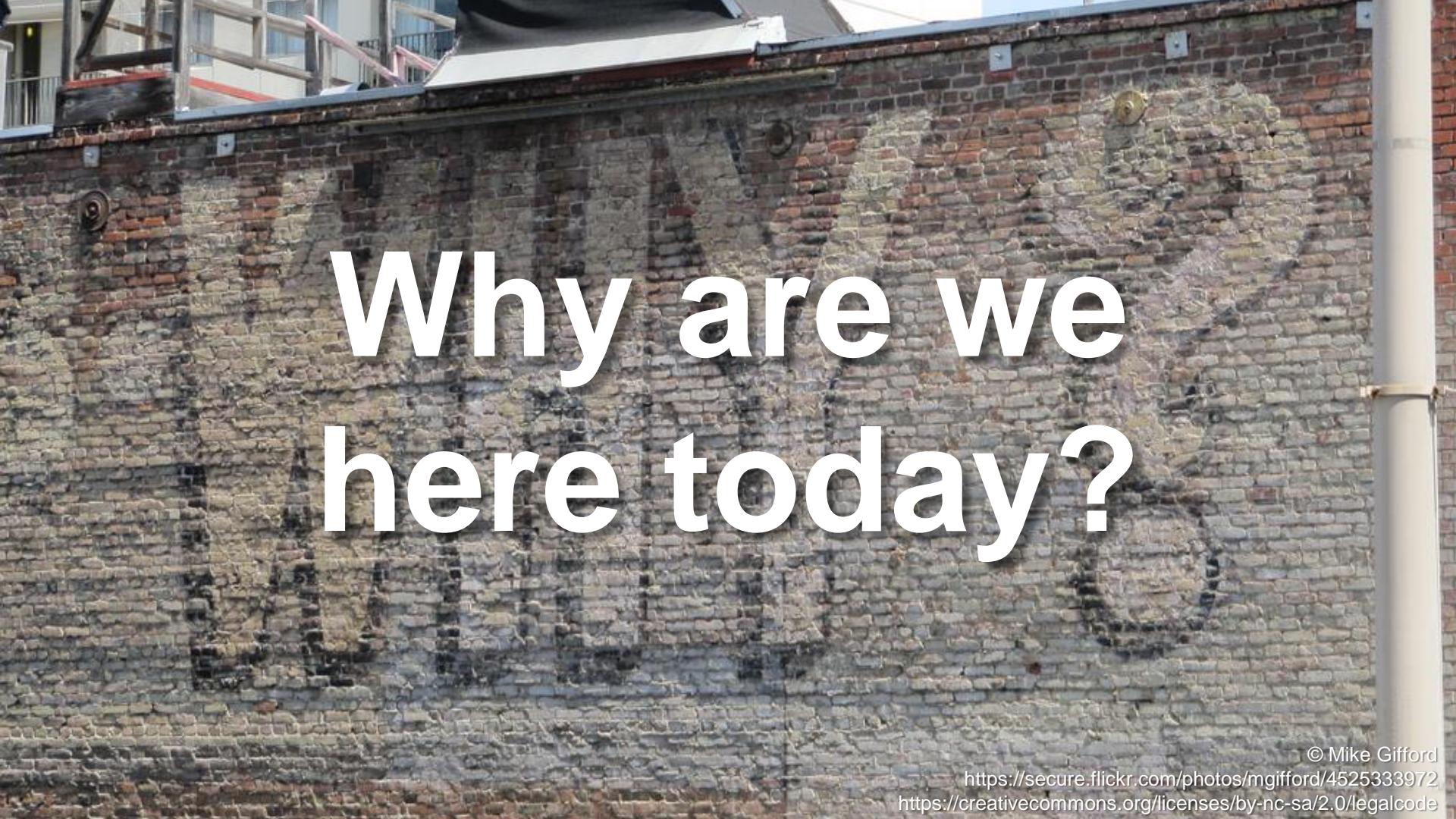
# DevOps on AWS

## Deep Dive on Continuous Delivery and the AWS Developer Tools

Matthew Trescot, Manager, Solutions Architecture

July 2016



A photograph of a weathered brick wall. A white cylindrical pipe runs vertically along the right side. In the top left corner, there's a metal railing and some structural elements. The brickwork is made of various shades of brown, tan, and grey.

# Why are we here today?

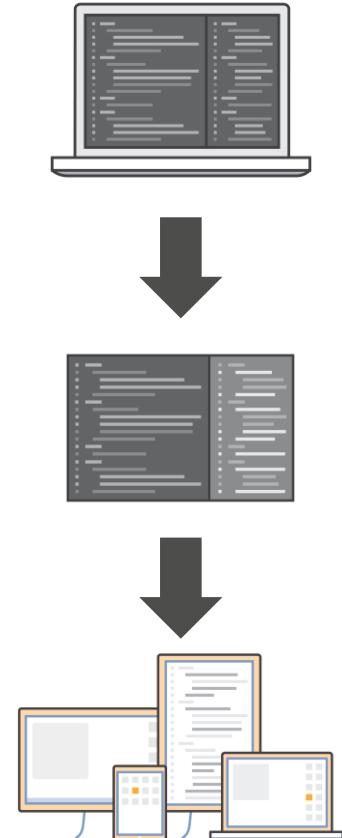
© Mike Gifford

<https://secure.flickr.com/photos/mgifford/4525333972>  
<https://creativecommons.org/licenses/by-nc-sa/2.0/legalcode>

# Software moves faster today

Software creation and distribution is easier and faster than ever

- Startups can now take on giants with little to no funding ahead of time
- Getting your software into the hands of millions is a download away
- Your ability to move fast is paramount to your ability to fight off disruption



# The software delivery model has drastically changed

## Old software delivery model



## New software delivery model



# What tools do you need to move fast?

Releasing software in this new software-driven world requires a number of things

- Tools to manage the flow of your software development release process
- Tools to properly test and inspect your code for defects and potential issues
- Tools to deploy your applications

A photograph of a donut production line. A long metal conveyor belt curves from the foreground to the background, with numerous golden-brown donuts spaced evenly along its surface. In the background, there are stacks of paper plates on the left and a worker in a red shirt and tan cap at a workstation on the right. Various pieces of industrial equipment, including pipes and containers, are visible. The overall scene is a busy food processing facility.

# First, we need to understand a little bit about software release processes

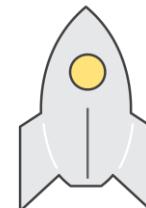
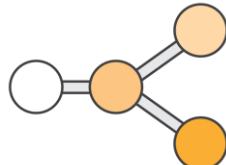
© Steve Jurvetson

<https://www.flickr.com/photos/jurvetson/5201796697/>  
<https://creativecommons.org/licenses/by-nc-sa/2.0/legalcode>

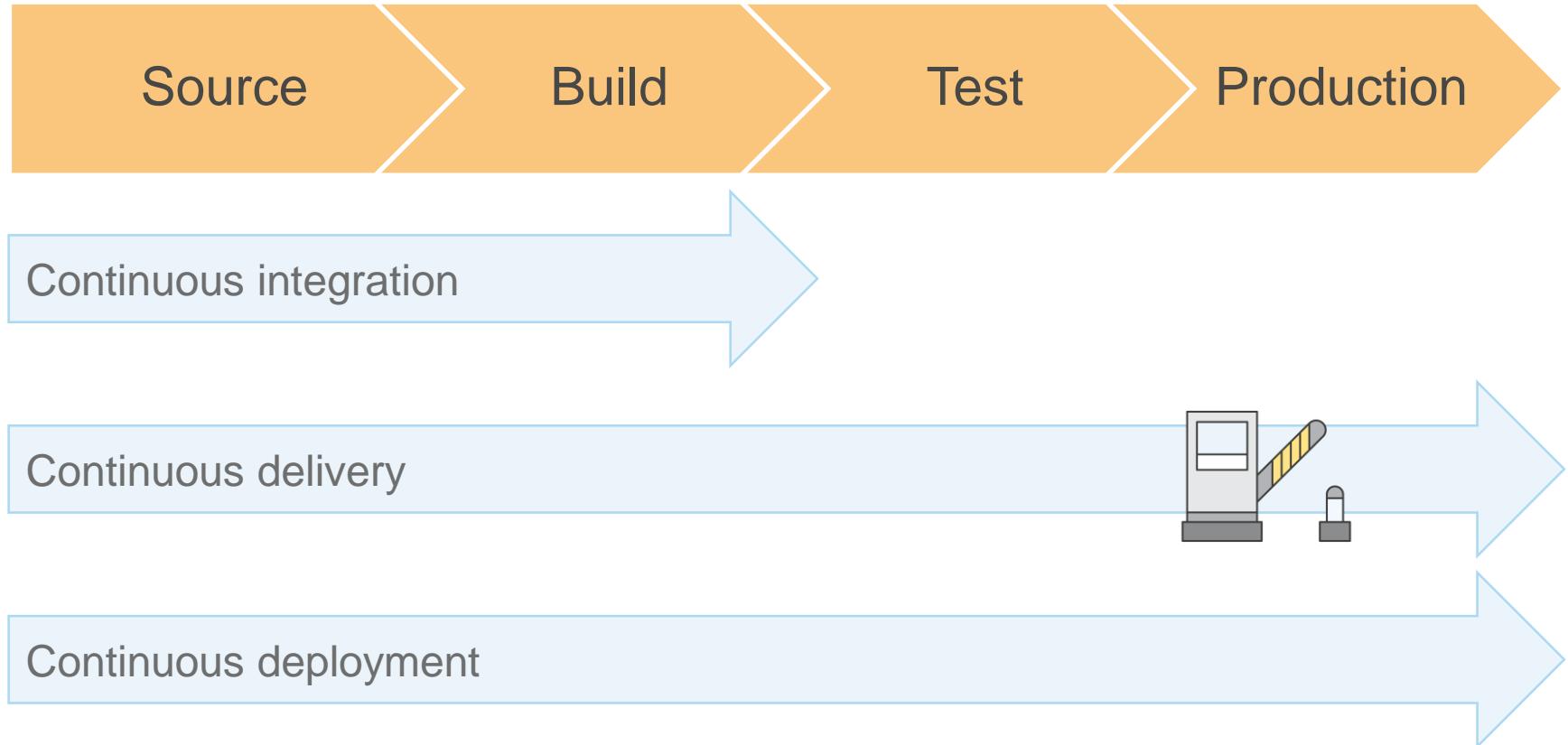
# Release processes have four major phases



- Check-in source code such as .java files.
- Peer review new code
- Compile code
- Unit tests
- Style checkers
- Code metrics
- Create container images
- Integration tests with other systems
- Load testing
- UI tests
- Penetration testing
- Deployment to production environments



# Release processes levels



# Release Processes levels

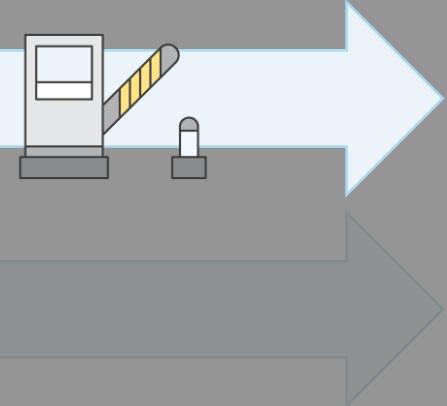
Source → Build → Test → Production

# Our focus today

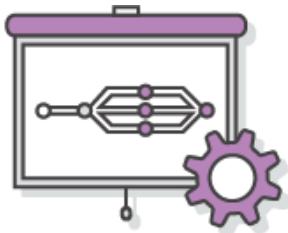
Continuous integration

Continuous delivery

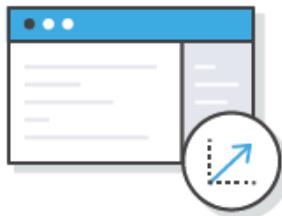
Continuous deployment



# Continuous delivery benefits



Automate the software release process



Improve developer productivity



Find and address bugs quickly



Deliver updates faster

The background image shows the grand interior of the Natural History Museum in London, featuring its iconic arched architecture and a wide staircase. The lighting is warm, highlighting the stone walls and the architectural details.

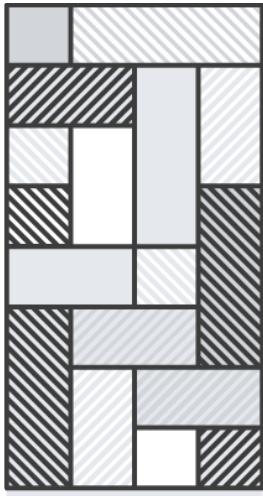
# A look back at development at Amazon

© Craig Morey

<https://secure.flickr.com/photos/pixelthing/15806918992/>  
<https://creativecommons.org/licenses/by-nc-sa/2.0/legalcode>

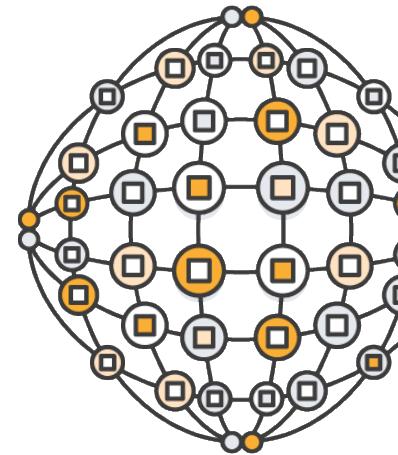
# Development transformation at Amazon: 2001–2009

2001



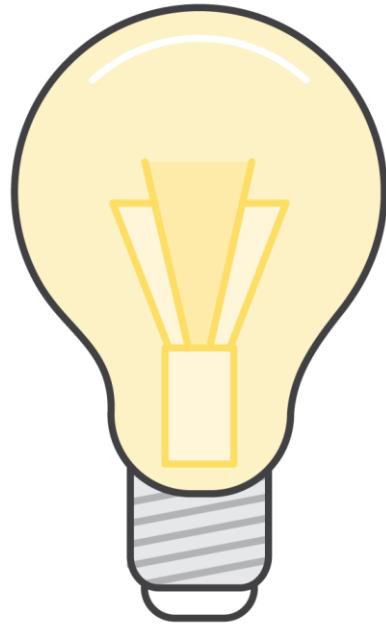
monolithic  
application + teams

2009



microservices + 2 pizza teams

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



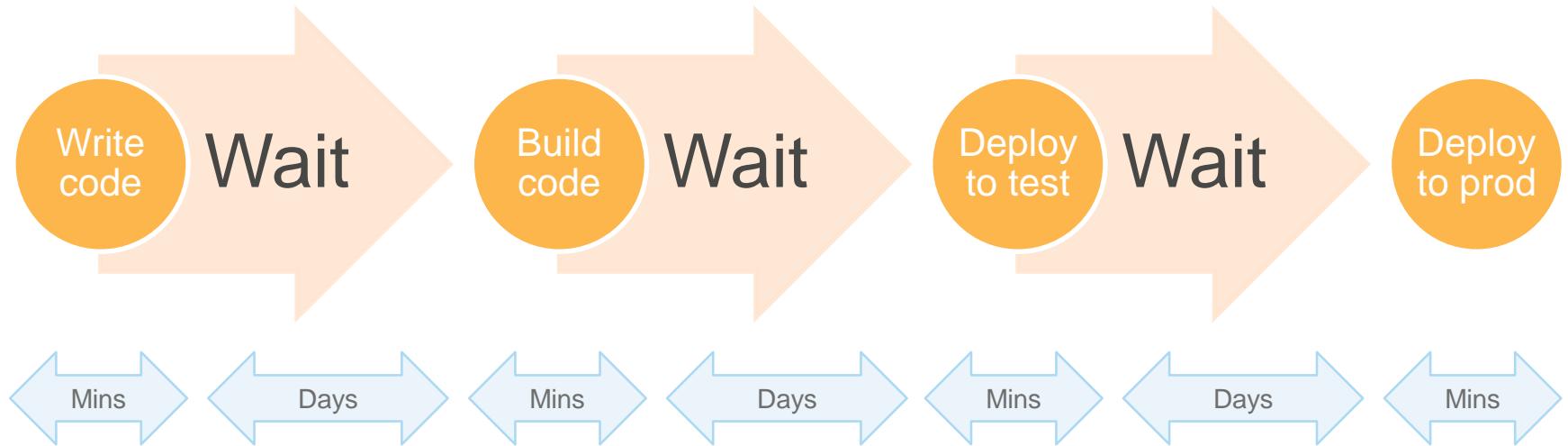


In 2009, we ran a study to find out where inefficiencies might still exist

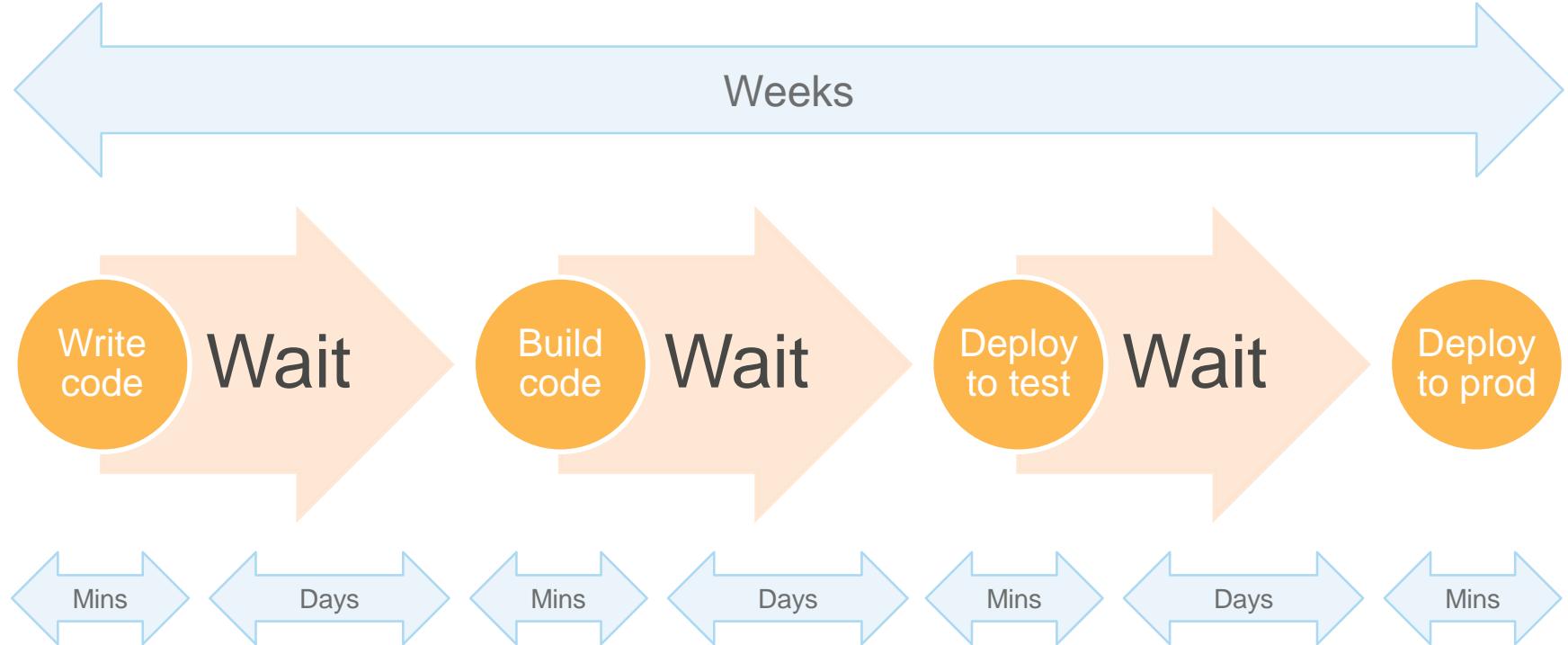
# We were just waiting



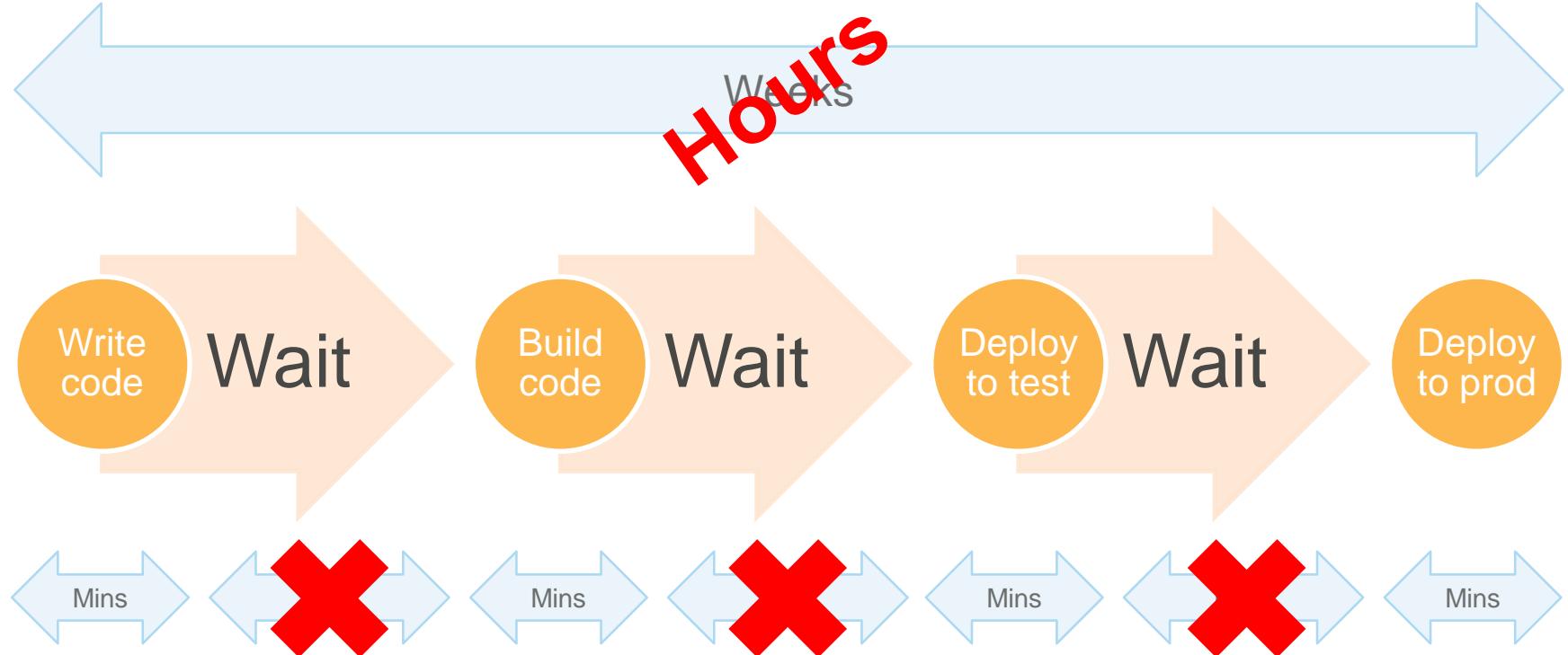
# We were just waiting.



# We were just waiting



# We were just waiting





# We built tools to automate our software release process

© Lindsey G

<https://secure.flickr.com/photos/lindseygee/5894617854/>

<https://creativecommons.org/licenses/by/2.0/legalcode>



# Pipelines

Automated actions and transitions; from check-in to production

## Development benefits

- Faster
- Safer
- Consistent and standardized
- Visualization of the process

# This has continued to work out really well

In 2014

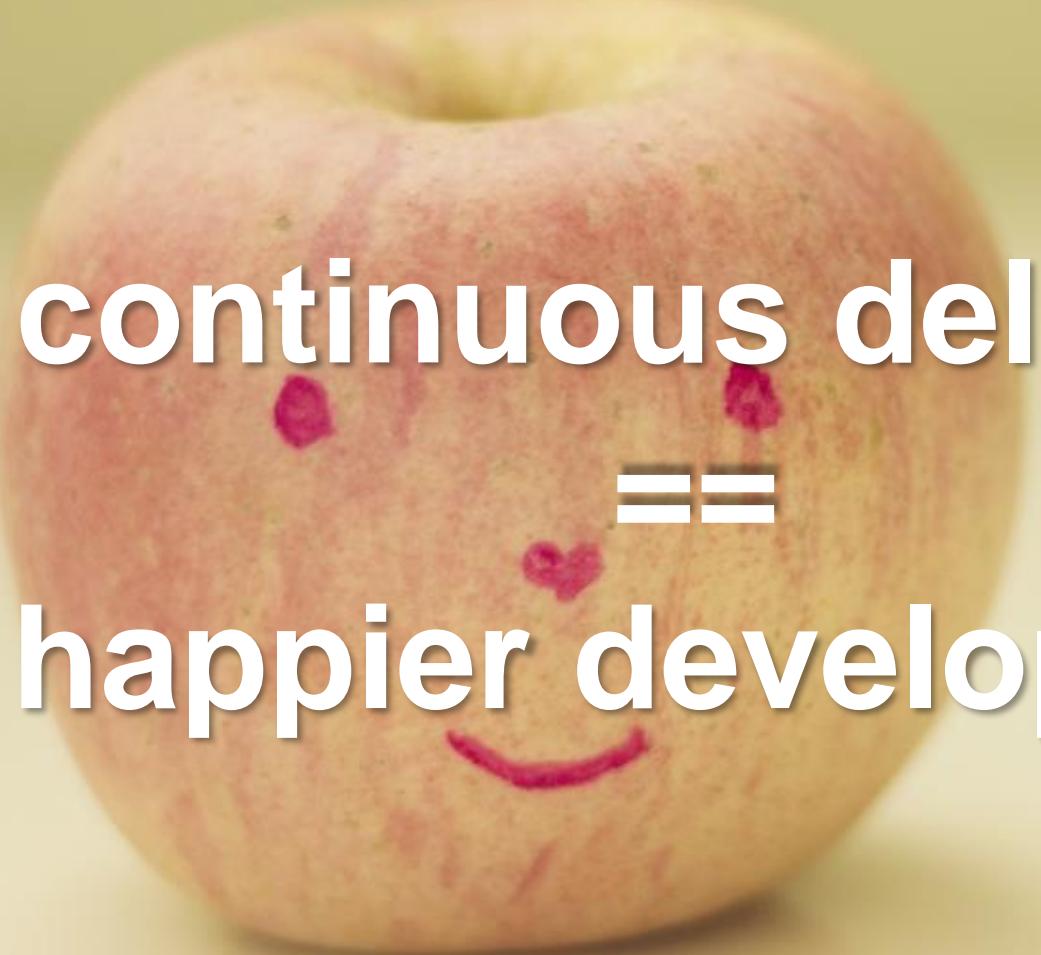
- Thousands of service teams across Amazon
- Building microservices
- Practicing continuous delivery
- Many environments (staging, beta, production)

**50 million deployments**

We continue to survey our software developers every year and in 2014 results found only one development tool or service could be correlated statistically with happier developers:



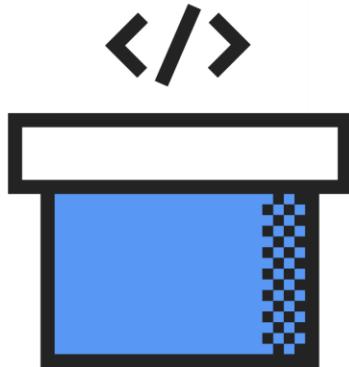
Our pipelines service!



continuous delivery  
≡  
happier developers!

# AWS CodePipeline

Continuous delivery service for fast and reliable application updates

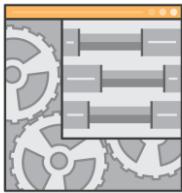


Model and visualize your software release process

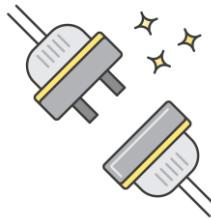
Builds, tests, and deploys your code every time there is a code change

Integrates with third-party tools and AWS

# AWS CodePipeline benefits



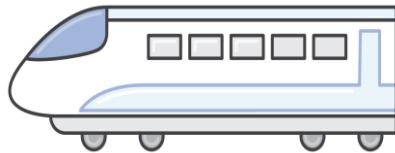
Configurable workflow



Easy to integrate



Improved quality



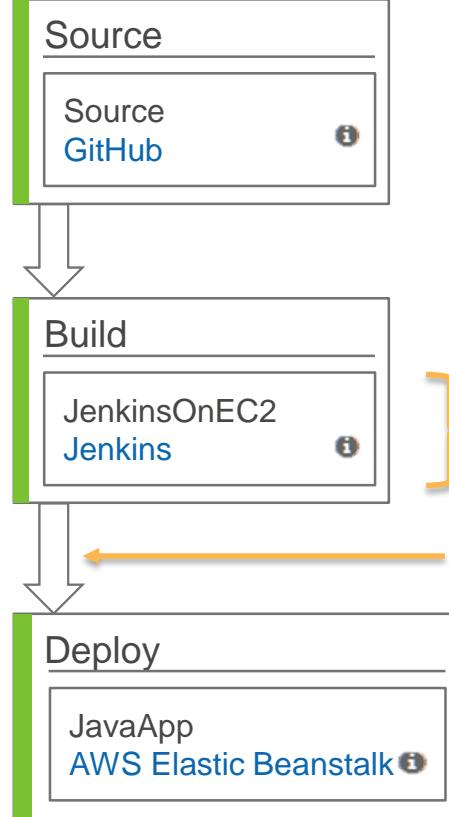
Rapid delivery



Get started fast



MyApplication



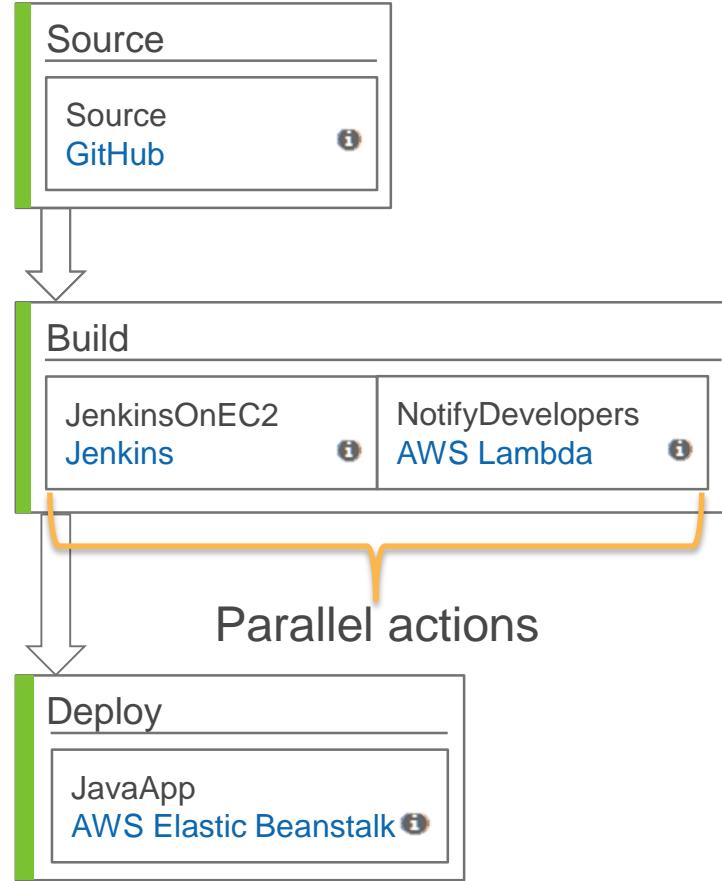
Stage  
Action

Transition

Pipeline

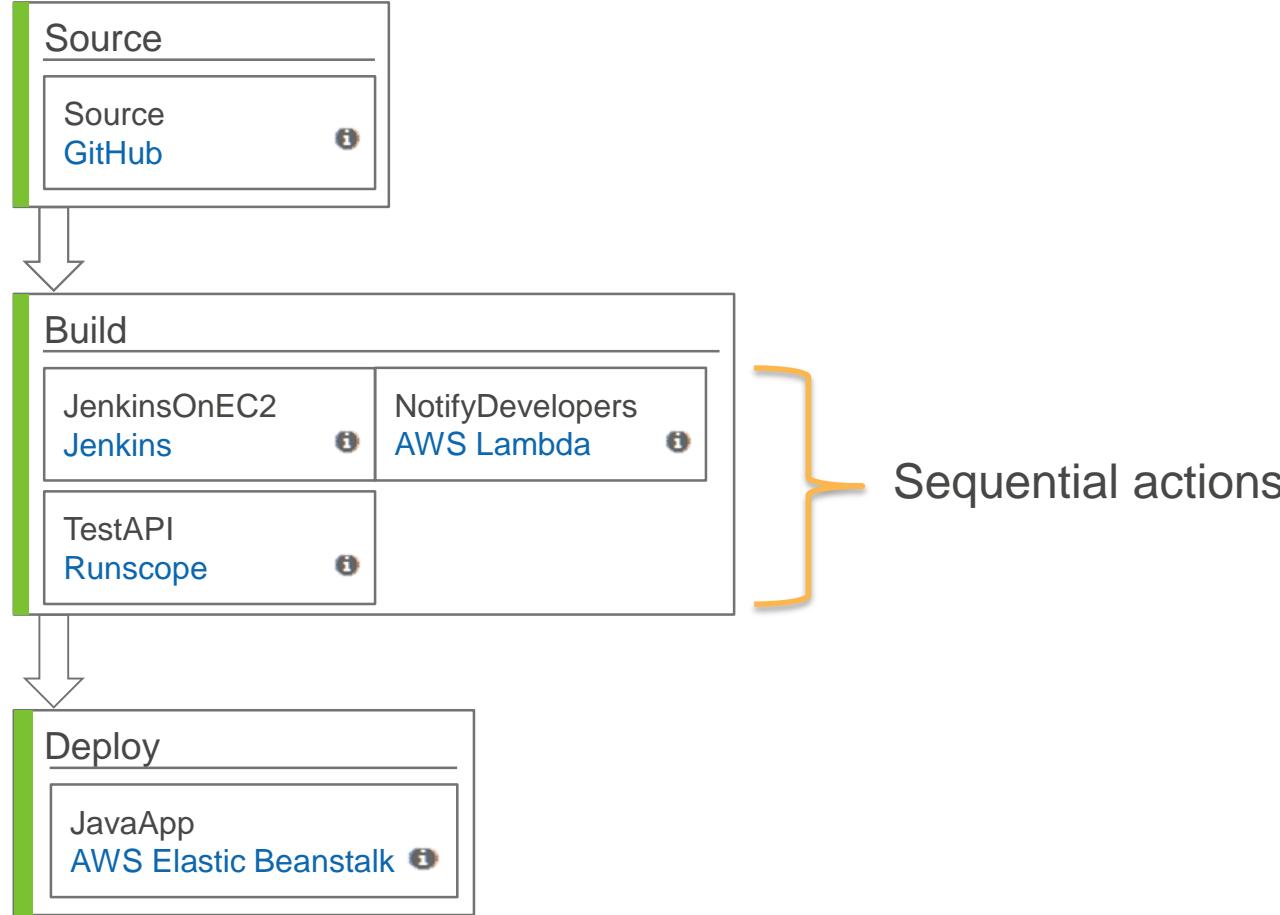


## MyApplication





## MyApplication

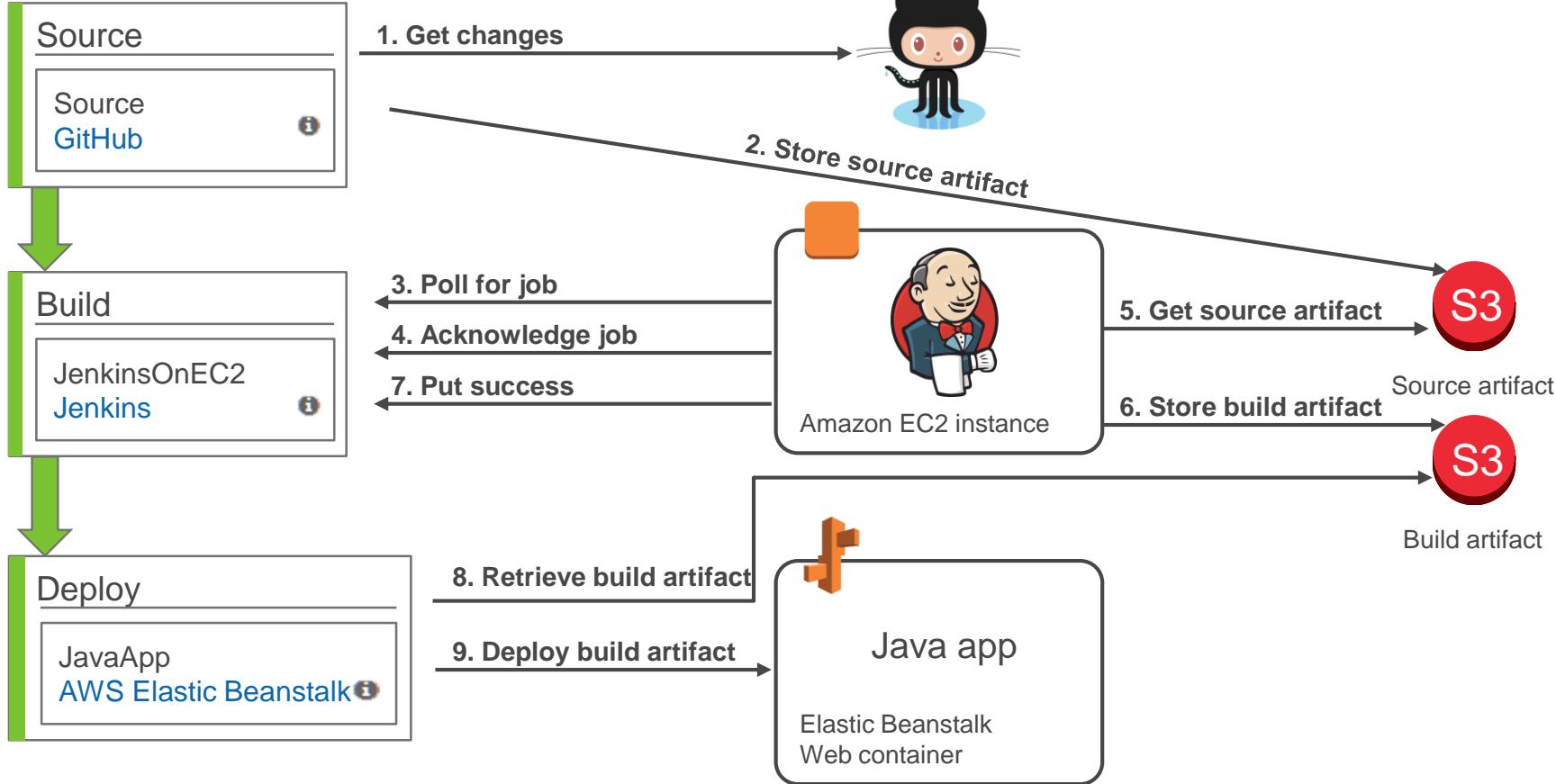


Sequential actions



# CodePipeline

MyApplication



# We have a strong partner list, and it's growing

Source

Build

Test

Deploy

**GitHub**

  
**CloudBees**

 **Apica**

  
**XebiaLabs**  
Deliver Faster

 **Jenkins**

 **BlazeMeter**

  
**Solano Labs**

 **Ghost Inspector**

 **HPE StormRunner**

 **Runscope**

# AWS service integrations

Source

Invoke logic

Deploy

Amazon S3

AWS Lambda

AWS CodeDeploy

AWS CodeCommit

AWS Elastic Beanstalk

AWS OpsWorks

# Building your application development release pipeline

A black and white historical photograph showing several workers in a trench laying large, corrugated metal pipes. One worker in the foreground is kneeling, holding a long pipe section. Another worker is standing behind him, also working on the pipe. A third worker is visible further down the trench. The ground is uneven dirt and rocks. The pipes are large and have a distinct ribbed or corrugated texture.

© Seattle Municipal Archives

<https://www.flickr.com/photos/seattlemunicipalarchives/12504672623/>

<https://creativecommons.org/licenses/by/2.0/legalcode>

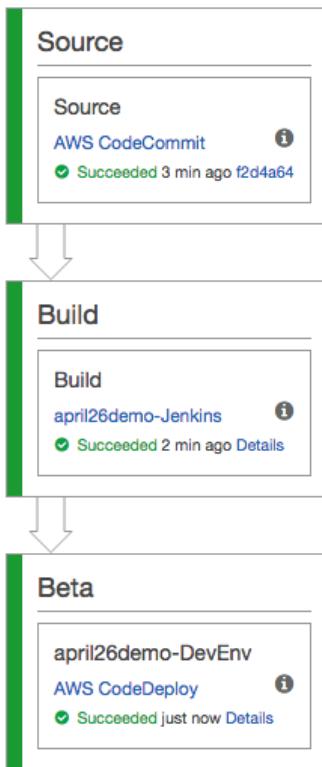


**DEMO!**

# Webinar-Pipeline



View progress and manage your pipeline.

[Edit](#)[Release change](#)

# Webinar-Pipeline

View progress and manage your pipeline.

Edit

Release change

## Source

### Source

AWS CodeCommit

✓ Succeeded 3 min ago f2d4a64

## Build

### Build

april26demo-Jenkins

✓ Succeeded 2 min ago Details

## Beta

april26demo-DevEnv

AWS CodeDeploy

✓ Succeeded just now Details

Dashboard

Code

Settings

Triggers

## Code: Suits4Dogs

Suits for Dogs demo application

Branch: master Clone URL

### Suits4Dogs

scripts

src

appspec.yml

LICENSE

pom.xml

README.md

README.md

### aws-codedeploy-sample-tomcat

A sample Tomcat application integrated with CodeDeploy.



Jenkins

Jenkins / april26demo / #2

- [Back to Project](#)
- [Status](#)
- [Changes](#)
- [Console Output](#)
- [Edit Build Information](#)
- [Delete Build](#)
- [Polling Log](#)
- [Redeploy Artifacts](#)
- [See Fingerprints](#)

## Build #2 (Apr 26, 2016 3:30:10 PM)



Failed to determine (log)



Started by an SCM change

### Module Builds

SampleMavenTomcatApp Maven Webapp 16 sec

## Deployment: d-29NPUR1FF

View information about your deployment.

### Deployment Details

✓ Deployment Succeeded

Application april26demo

Deployment Group april26demo-DevEnv

Deployment ID d-29NPUR1FF

Deployment Config CodeDeployDefault.OneAtATime

Minimum Healthy Hosts 0 of 1 instances

### Revision

Revision Location s3://codepipeline-us-east-1-63534231114/Webinar-Pipeline/MyAppBuild/d-29NPUR1FF/etag=ab a485bd23e117139de072565f6db35e-6

Revision Created 3 minutes ago

Description Application revision registered by Deployment ID: d-29NPUR1FF

Filter Status Instances per page 10 Viewing 1 to 1 instances

Instance ID	Start Time	End Time	Duration	Status	Most Recent Event	Events
i-2c1018b7	3 minutes ago	3 minutes ago	32 secs	Succeeded	ValidateService	<a href="#">View Events</a>



A photograph of an industrial factory floor, likely an automotive plant. Numerous orange KUKA brand industrial robots are positioned around a series of car chassis. The robots are performing welding tasks, with bright sparks flying from their work points. The factory is filled with complex machinery, conveyor belts, and structural elements. In the background, more car frames are visible on the assembly line. The lighting is industrial, with overhead lights reflecting off the metal surfaces.

# Build and test your application

© Spencer Cooper

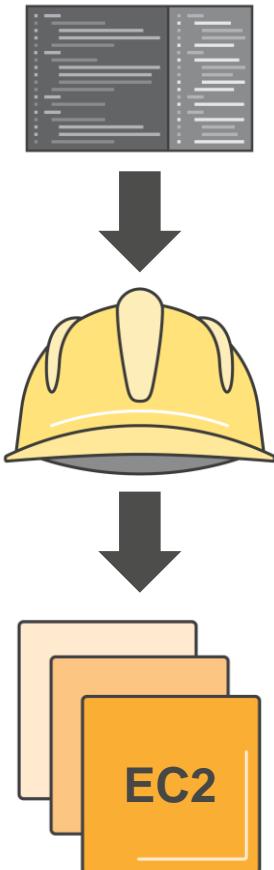
<https://secure.flickr.com/photos/spenceyc/7481166880>  
<https://creativecommons.org/licenses/by/2.0/legalcode>

# Building your code

“Building” code typically refers to languages that require compiled binaries

- .NET languages: C#, F#, VB.net, etc
- Java and JVM languages: Java, Scala, JRuby
- Go
- iOS languages: Swift, Objective-C

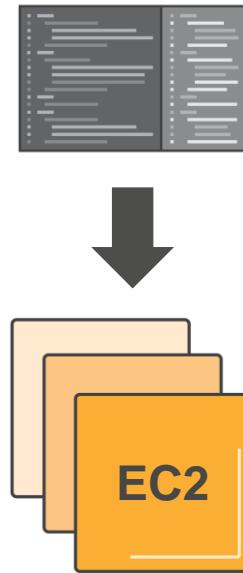
We also refer to the process of creating Docker container images as “building” the image



# No building required!

Many languages don't require building; these are considered interpreted languages

- PHP
- Ruby
- Python
- Node.js



You can just deploy your code!

# Testing your code

Testing is both a science and an art form!

Goals for testing your code

- Want to confirm desired functionality
- Catch programming syntax errors
- Standardize code patterns and format
- Reduce bugs due to undesired application usage and logic failures
- Make applications more secure



A large wooden trebuchet, a medieval siege engine, is positioned on a stone pier overlooking a body of water with distant hills. The trebuchet's arm is angled upwards, and its wooden frame is detailed with metal hardware and rivets. In the background, a few people are walking on a grassy area near a stone wall.

# Deploying your applications

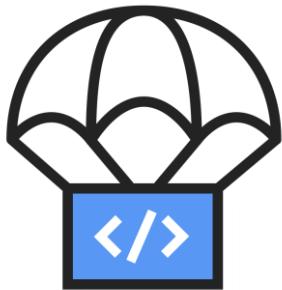
© Simon Q

<https://secure.flickr.com/photos/simononly/15386966677>  
<https://creativecommons.org/licenses/by/2.0/legalcode>

# AWS CodeDeploy

Automates code deployments to any instance

Handles the complexity of updating your applications



Avoid downtime during application deployment

Deploy to Amazon EC2 or on-premises servers in any language and on any operating system

Integrates with third-party tools and AWS

# appspec.yml Example

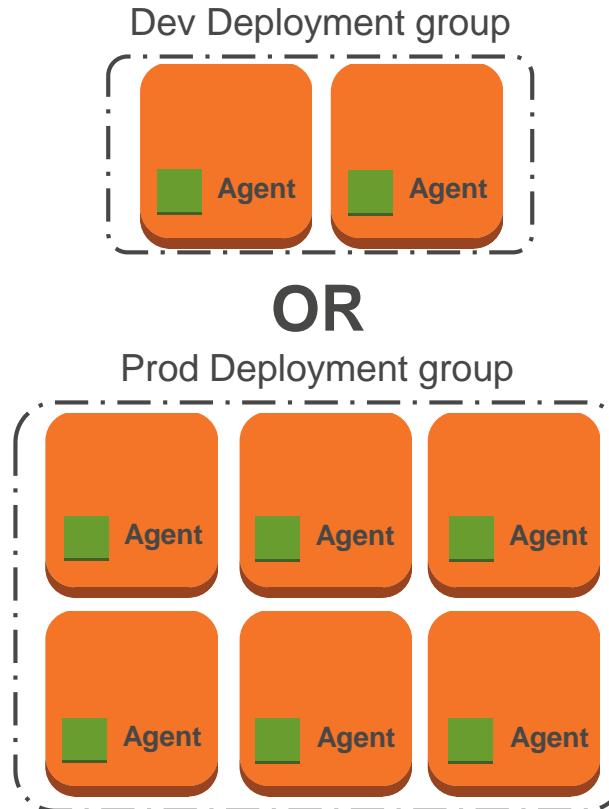
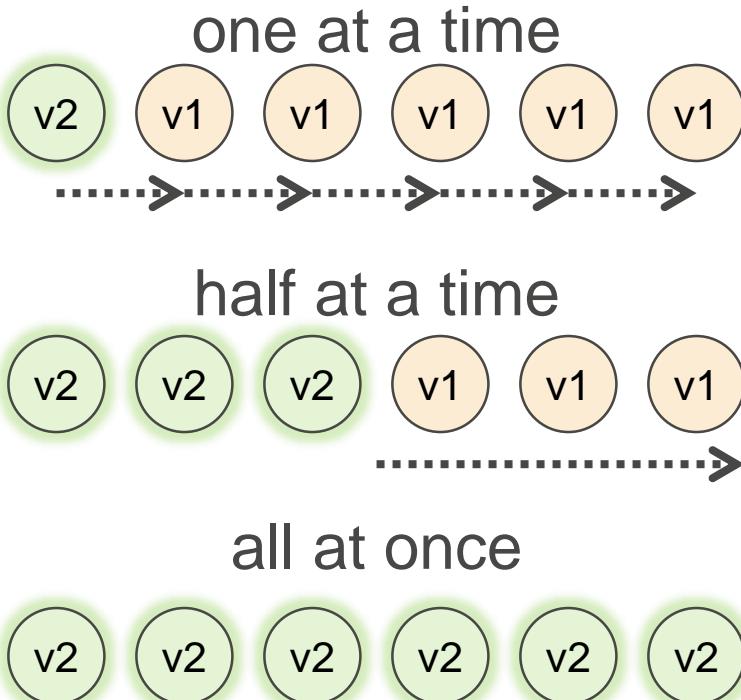
```
version: 0.0
os: linux
files:
- source: /
  destination: /var/www/html
permissions:
- object: /var/www/html
  pattern: “*.html”
  owner: root
  group: root
  mode: 755
hooks:
ApplicationStop:
- location: scripts/deregister_from_elb.sh
BeforeInstall:
- location: scripts/install_dependencies.sh
ApplicationStart:
- location: scripts/start_httpd.sh
ValidateService:
- location: scripts/test_site.sh
- location: scripts/register_with_elb.sh
```

# appspec.yml Example

```
version: 0.0
os: linux
files:
  - source: /
    destination: /var/www/html
permissions:
  - object: /var/www/html
    pattern: "*.html"
    owner: root
    group: root
    mode: 755
hooks:
  ApplicationStop:
    - location: scripts/deregister_from_elb.sh
  BeforeInstall:
    - location: scripts/install_dependencies.sh
  ApplicationStart:
    - location: scripts/start_httpd.sh
  ValidateService:
    - location: scripts/test_site.sh
    - location: scripts/register_with_elb.sh
```

- Send application files to one directory and configuration files to another
- Set specific permissions on specific directories and files
- Remove/add instance to Elastic Load Balancing
- Install dependency packages
- Start Apache
- Confirm successful deploy
- More!

# Choose deployment speed and group



A photograph of a rocket launch at night. A bright, orange-yellow arc of light curves upwards across the dark blue sky, starting from the bottom left and ending near the top right. The horizon line is visible at the bottom, showing some lights and structures from the launch site.

# Launching to production

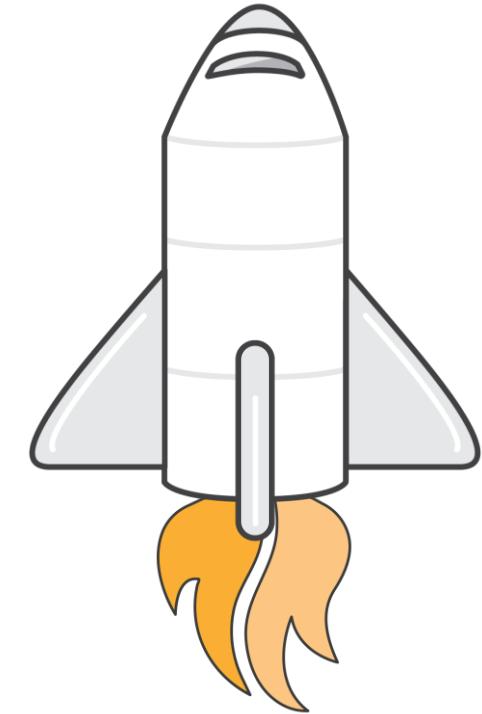
# Launching to production

After you've built and tested your code and hopefully gone through a few preproduction deploys, its time for the real thing!

You'll want think about the following

- Impact to customers
- Impact to infrastructure
- Impact to business

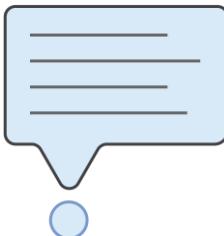
How can we track these and communicate deployments?



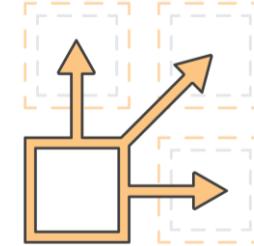
# Extend AWS CodePipeline using custom actions



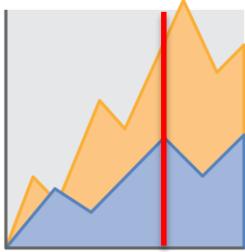
Mobile testing



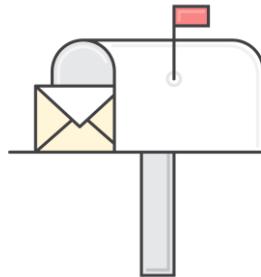
Update tickets



Provision resources



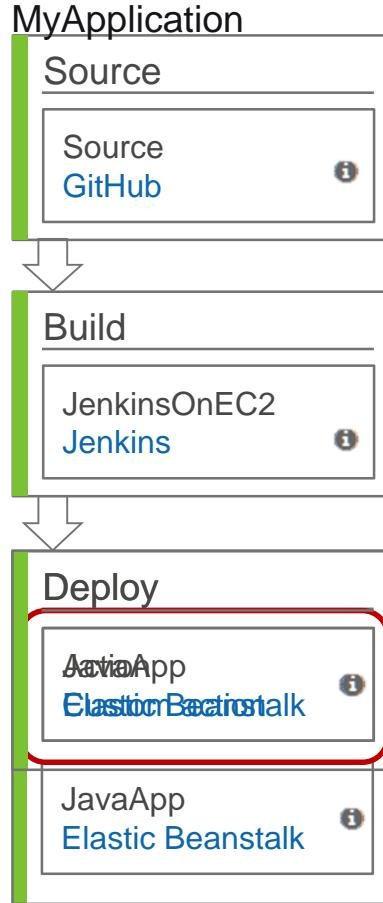
Update dashboards



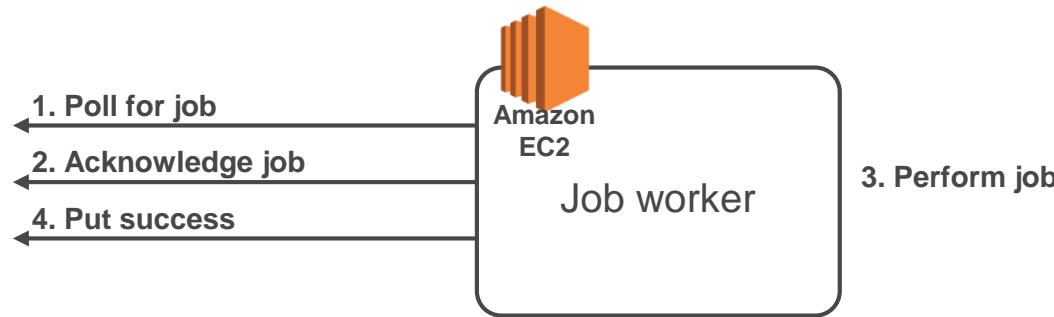
Send notifications

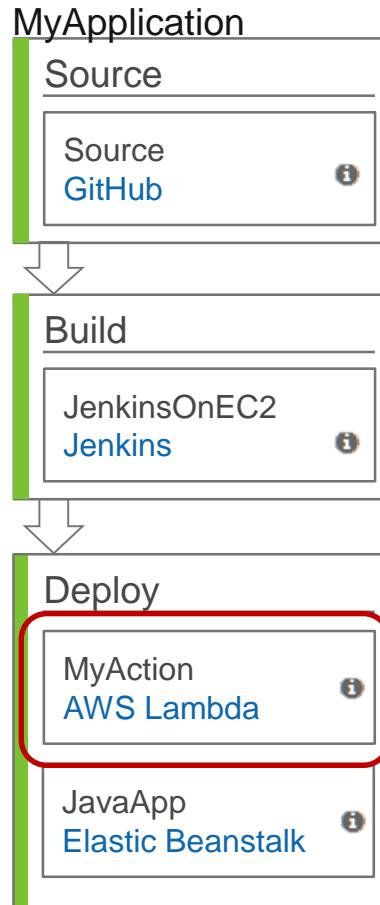


Security scan



With custom actions,  
the job worker drives the interaction  
between AWS CodePipeline  
and other applications or services

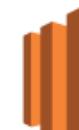




With AWS Lambda-based actions,  
AWS CodePipeline  
drives the integration with Lambda,  
which then connects with other  
applications or services



2. Perform job



1. Invoke Lambda function

3. PutJobSuccessResult w/  
continuation token

4. Invoke Lambda function w/  
continuation token

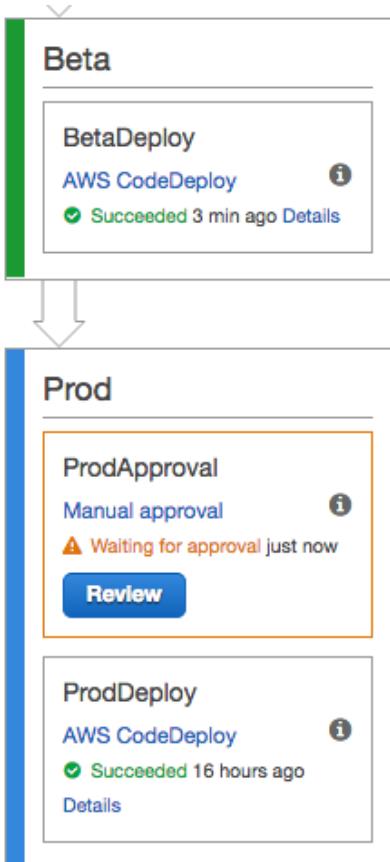
5. PutJobSuccessResult

#3 and #4 repeat until no continuation  
token is sent, signaling the action has  
been completed (#5)

# What extension method should I use?

Lambda	Custom action
Short-running tasks are easy to build	Can perform any type of workload
Long-running tasks need more work	Control over links displayed in console
Node.js, Python, and Java support	Any language support
Runs on AWS	Can run on premises
No servers to provision or manage	Requires compute resources

# Manual approvals – new!



You can add a manual approval at the point where you want the pipeline to stop running until someone approves or rejects the revision in progress

- Pipeline stops executing when it has reached the point at which you set the approval action
- Pipeline execution resumes only when the action has been approved
- Approval action managed with AWS Identity and Access Management (IAM) permissions
- Notify approvers in several ways including email, SMS, webhooks, and more

# FIN, ACK

We've seen a quick run through today of the benefits of continuous delivery on our software release process

- Continuous integration (build/test) helps shrink our feedback loop greatly
- We can get our software out in front of our users much more rapidly
- By moving faster we can actually ensure better quality
- CodePipeline allows for integration with almost any service or tool you can think of!
  - Plus visualization of what's going on!

# Try it out today

Test out CodePipeline and spin up a full continuous delivery pipeline using the starter kit

**[bit.ly/AWSCodeStarterKit](https://bit.ly/AWSCodeStarterKit)**

# But wait, there's more!

## Resources to learn more

- Continuous integration: <https://aws.amazon.com/devops/continuous-integration/>
- Continuous delivery: <https://aws.amazon.com/devops/continuous-delivery/>
- CodePipeline
  - <https://aws.amazon.com/codepipeline/>
  - <https://aws.amazon.com/documentation/codepipeline/>
- CodeDeploy
  - <https://aws.amazon.com/codedeploy/>
  - <https://aws.amazon.com/documentation/codedeploy/>
  - <https://github.com/awslabs/aws-codedeploy-samples>
- Code services starter kit: <http://bit.ly/AWSCodeStarterKit>



# Thank You!