



DevOps on AWS: Deep Dive on Continuous Delivery and the AWS Developer Tools

Chris Munns, Business Development Manager - DevOps

August 11, 2016



A photograph of a weathered brick wall. The wall is made of reddish-brown bricks and shows signs of age and wear, including discoloration and small holes. A large, bold, white sans-serif font is overlaid on the wall, containing the text "Why are we here today?". The text is centered and occupies the upper half of the image.

Why are we
here today?

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 tools:

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



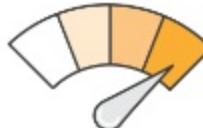
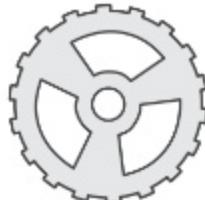
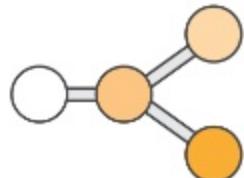
First, we need to understand a little bit about software release processes



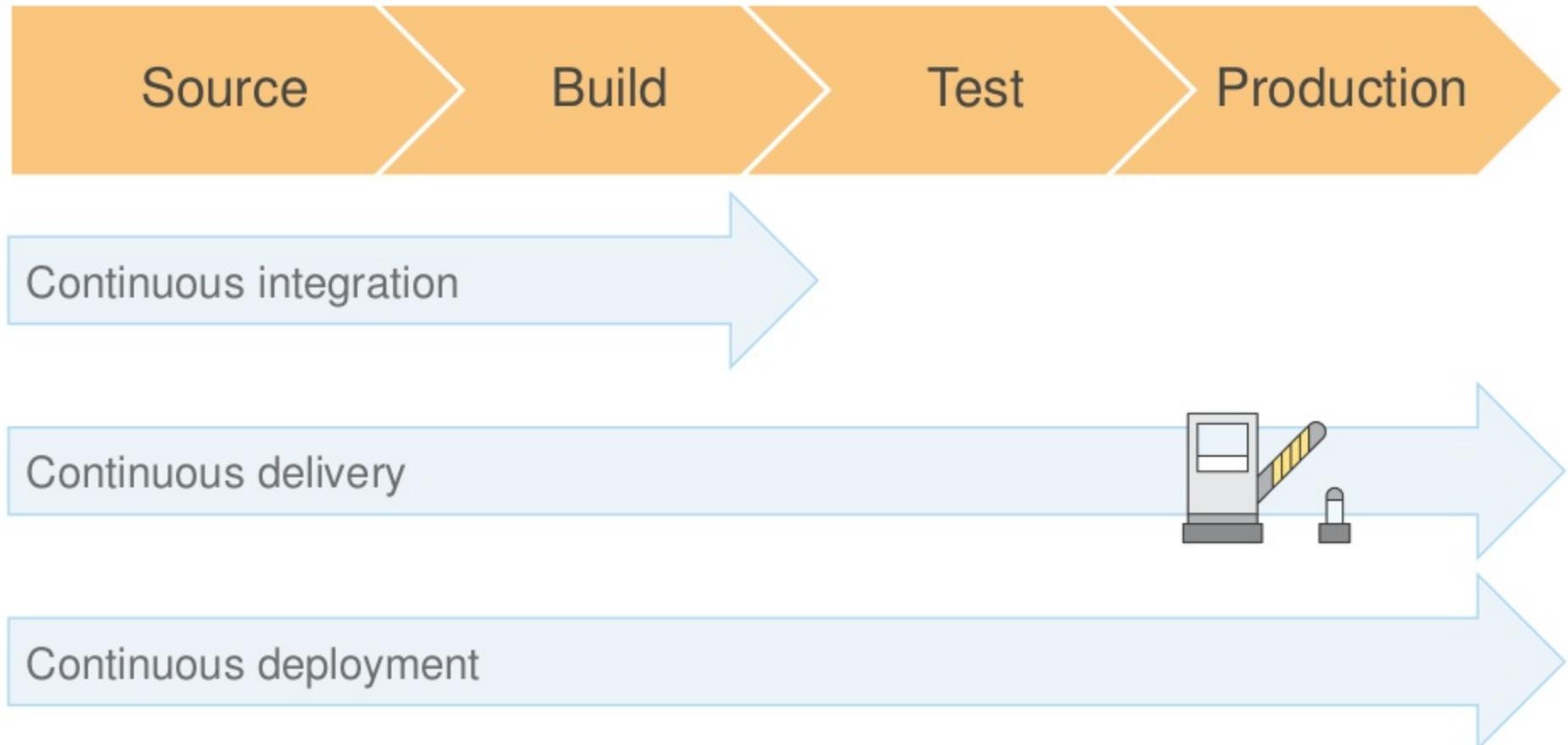
Release processes have four major phases



- 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



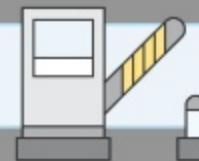
Release Processes levels

Our focus today

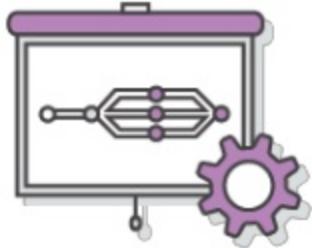
Continuous integration

Continuous delivery

Continuous deployment



Continuous delivery benefits



Automate the software release process



Improve developer productivity



Find and address bugs quickly



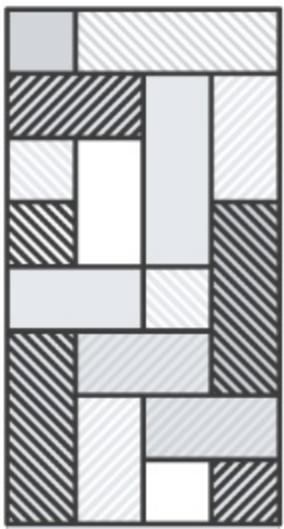
Deliver updates faster



A look back at development at Amazon

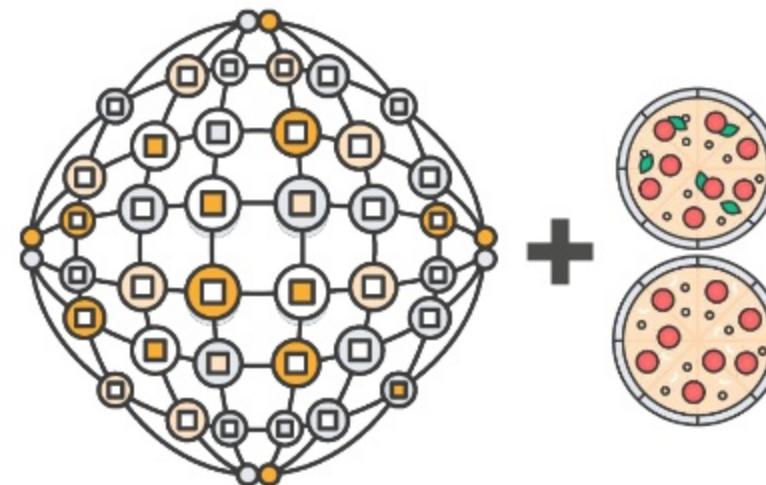
Development transformation at Amazon: 2001-2009

2001



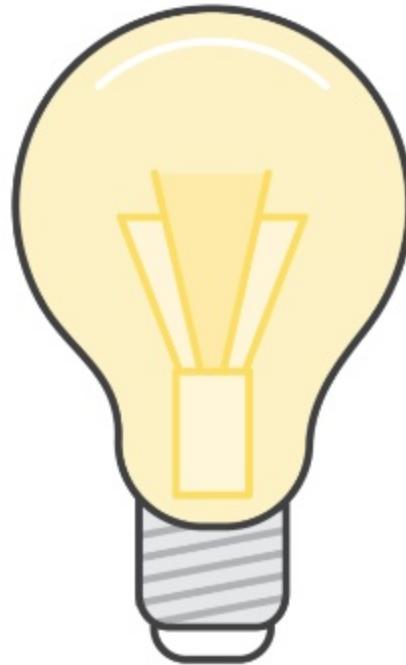
monolithic
application + teams

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



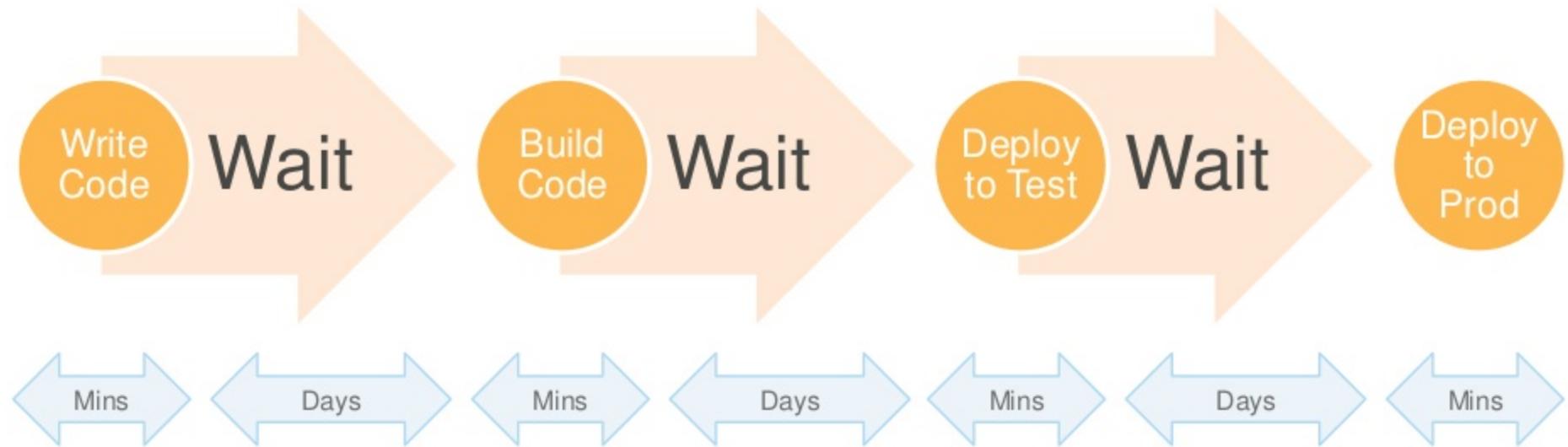


In 2009, we ran a study to find out where inefficiencies might still exist

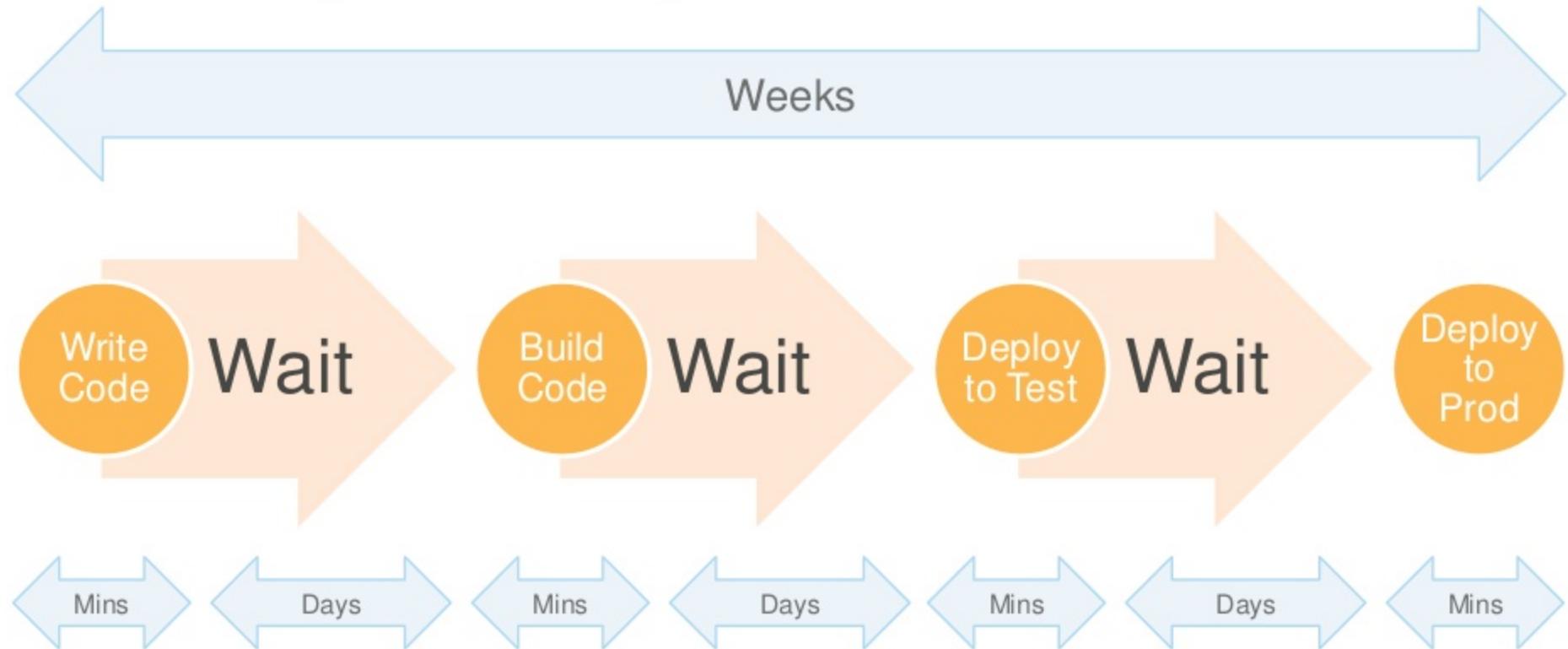
We were just waiting.



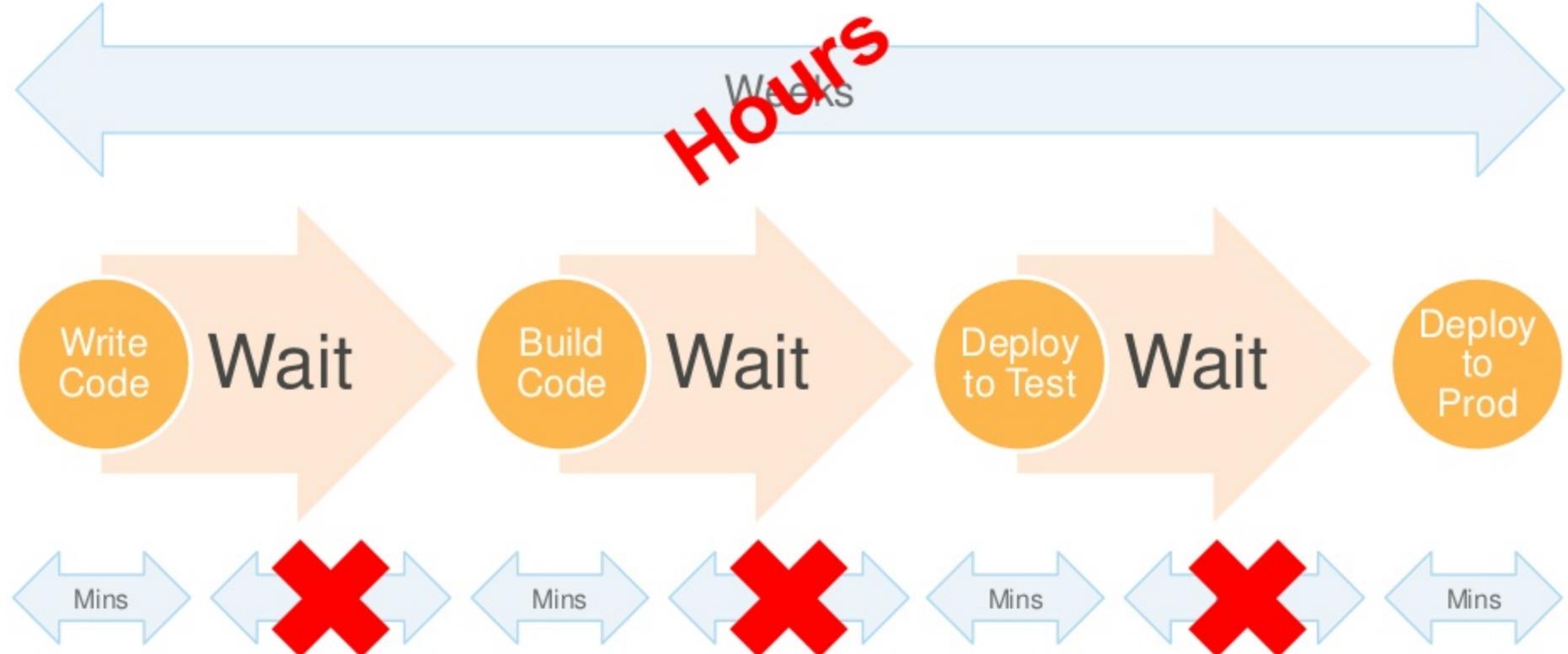
We were just waiting.



We were just waiting.



We were just waiting.



A photograph showing a collection of large-diameter metal pipes, possibly steel, stacked in a somewhat haphazard manner on a bed of dark gravel or small stones. The pipes have a weathered, light-colored surface with visible rust and paint peeling. Some have black caps or fittings attached to their ends. The lighting suggests a bright day, casting long shadows of the pipes onto the gravel.

We built tools to
automate our software
release process



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/service could be correlated statistically with happier developers:



Our pipelines service!

A close-up photograph of a yellow apple with red spots. A hand-drawn smiley face is on the apple, featuring two small red heart shapes for eyes and a curved red line for a mouth. Overlaid on the image is the text "continuous delivery" in large white letters at the top and "happier developers!" in large white letters below it, separated by two horizontal black lines.

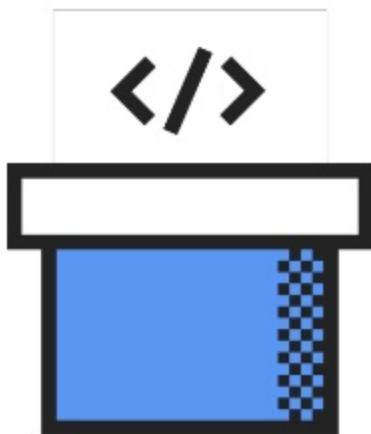
continuous delivery

==

happier developers!

AWS CodePipeline

Continuous delivery service for fast and reliable application updates

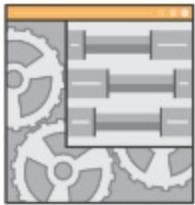


Model and visualize your software release process

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

Integrates with third-party tools and AWS

AWS CodePipeline benefits



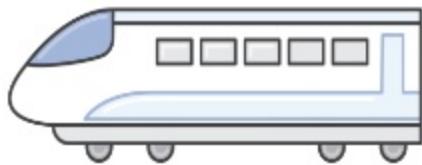
Configurable workflow



Easy to integrate



Improved quality



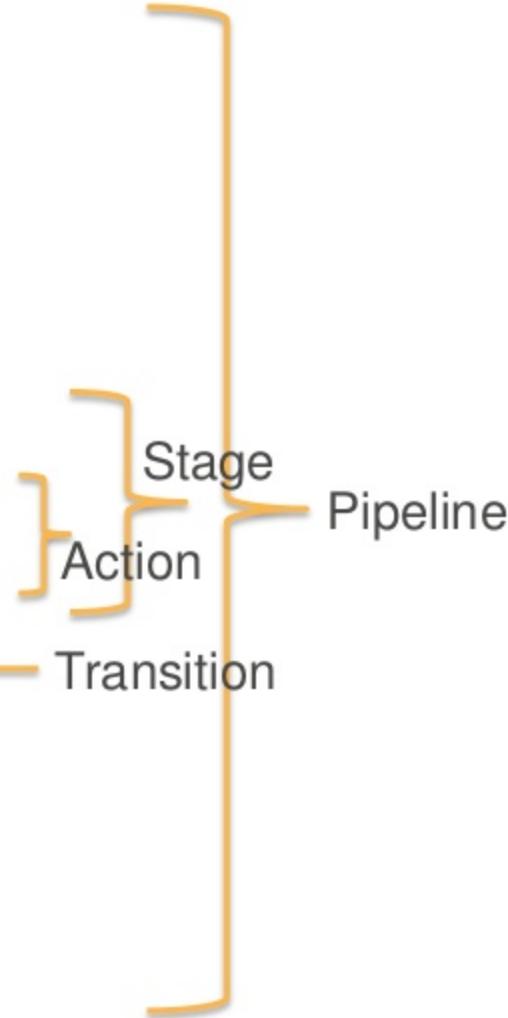
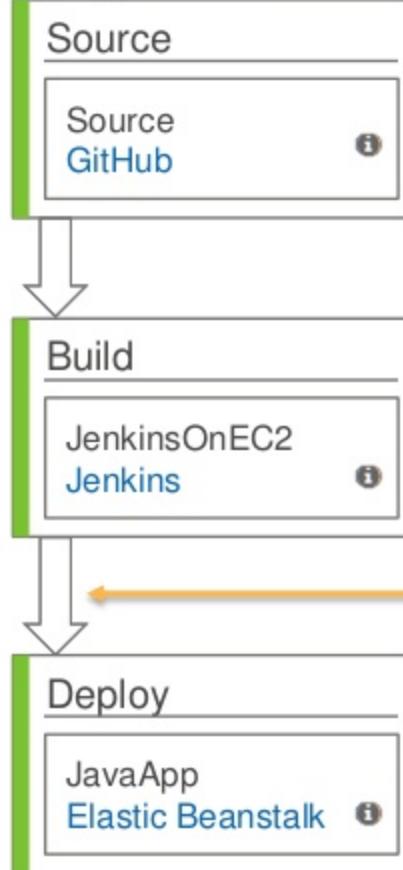
Rapid delivery



Get started fast

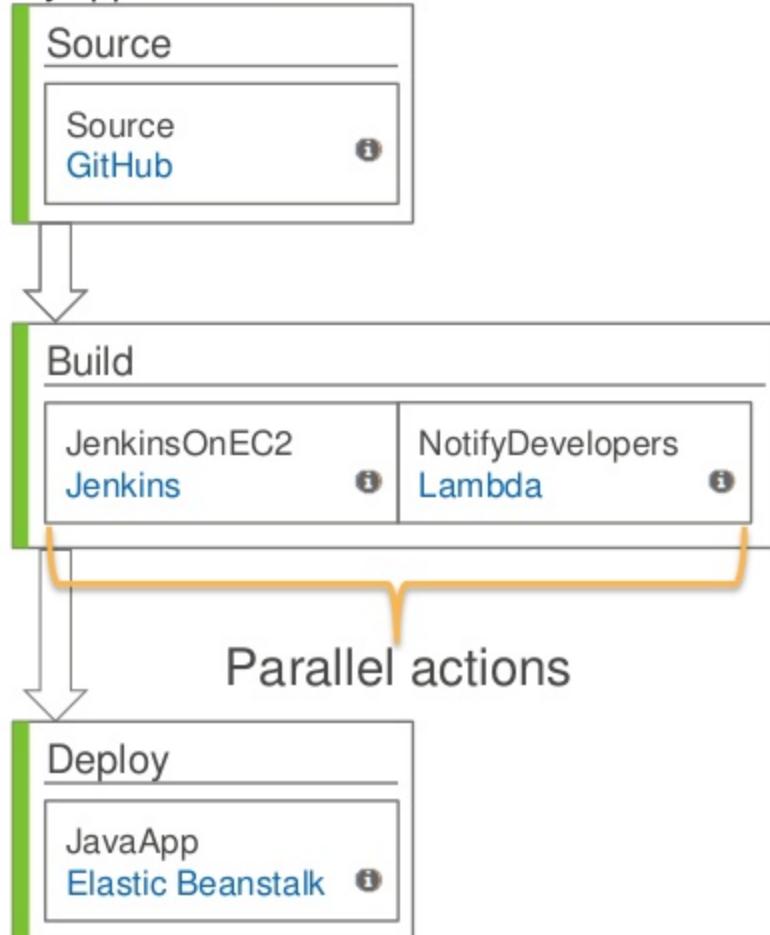


MyApplication



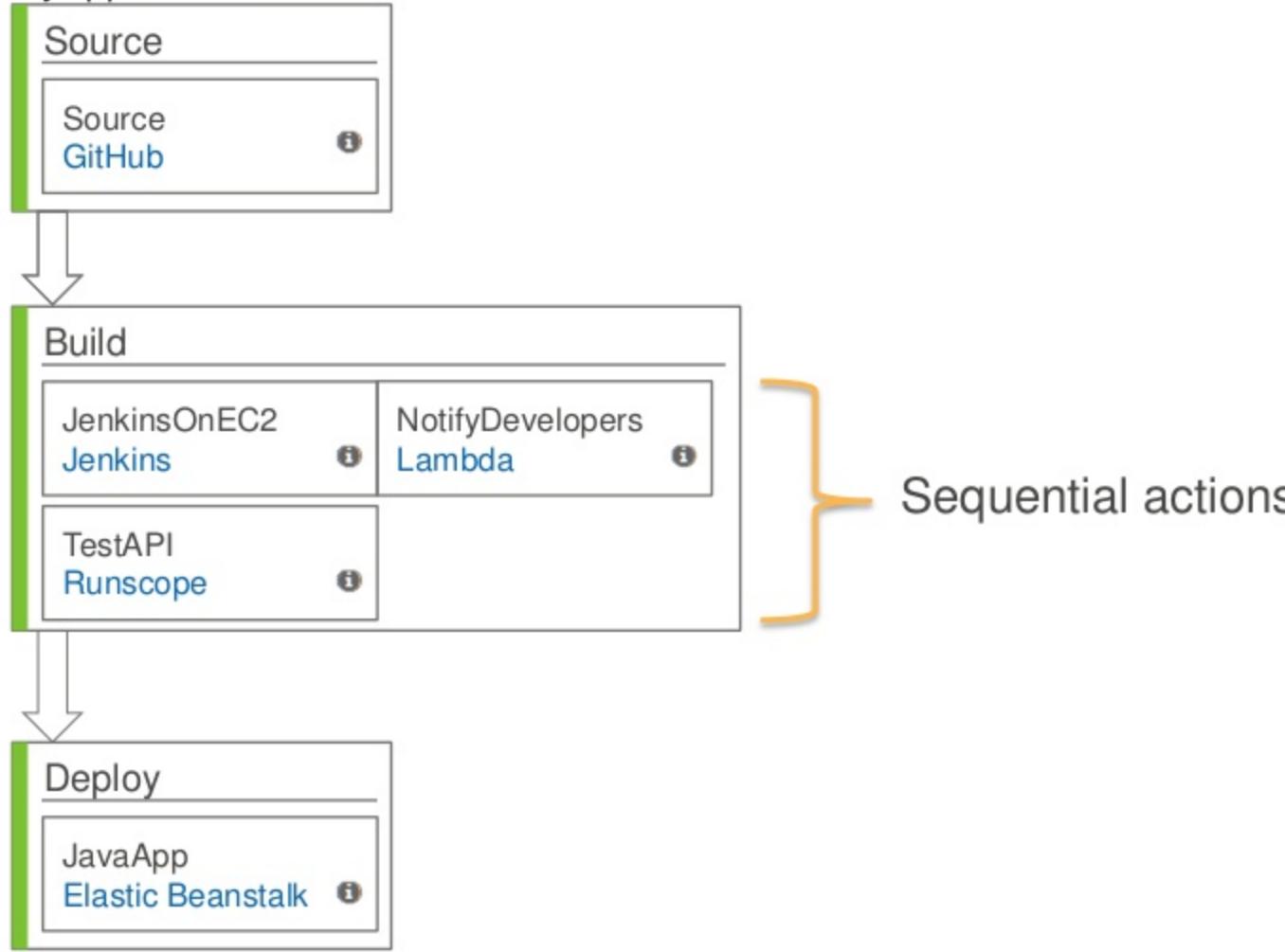


MyApplication





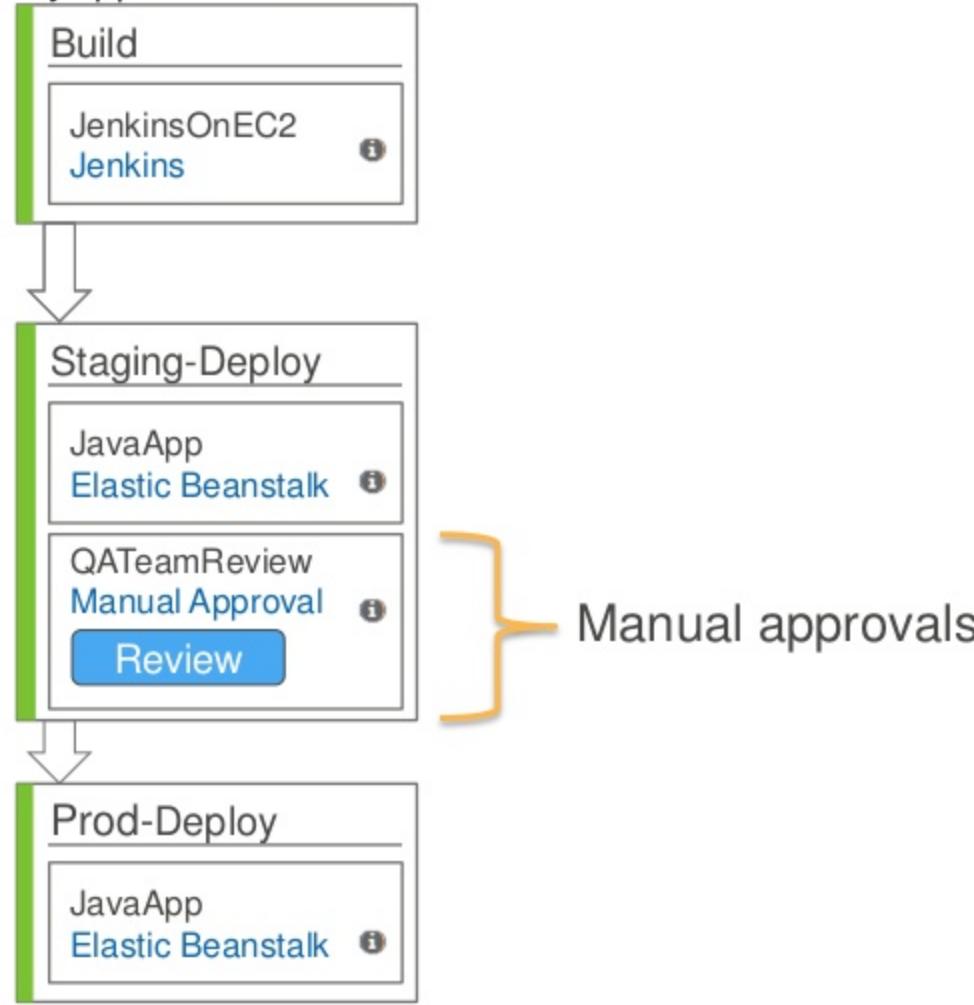
MyApplication



Sequential actions

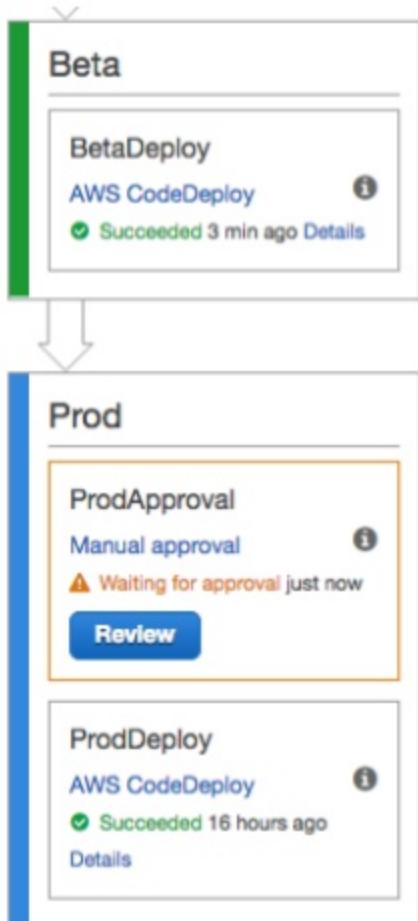


MyApplication



Manual approvals

Manual approvals – New!

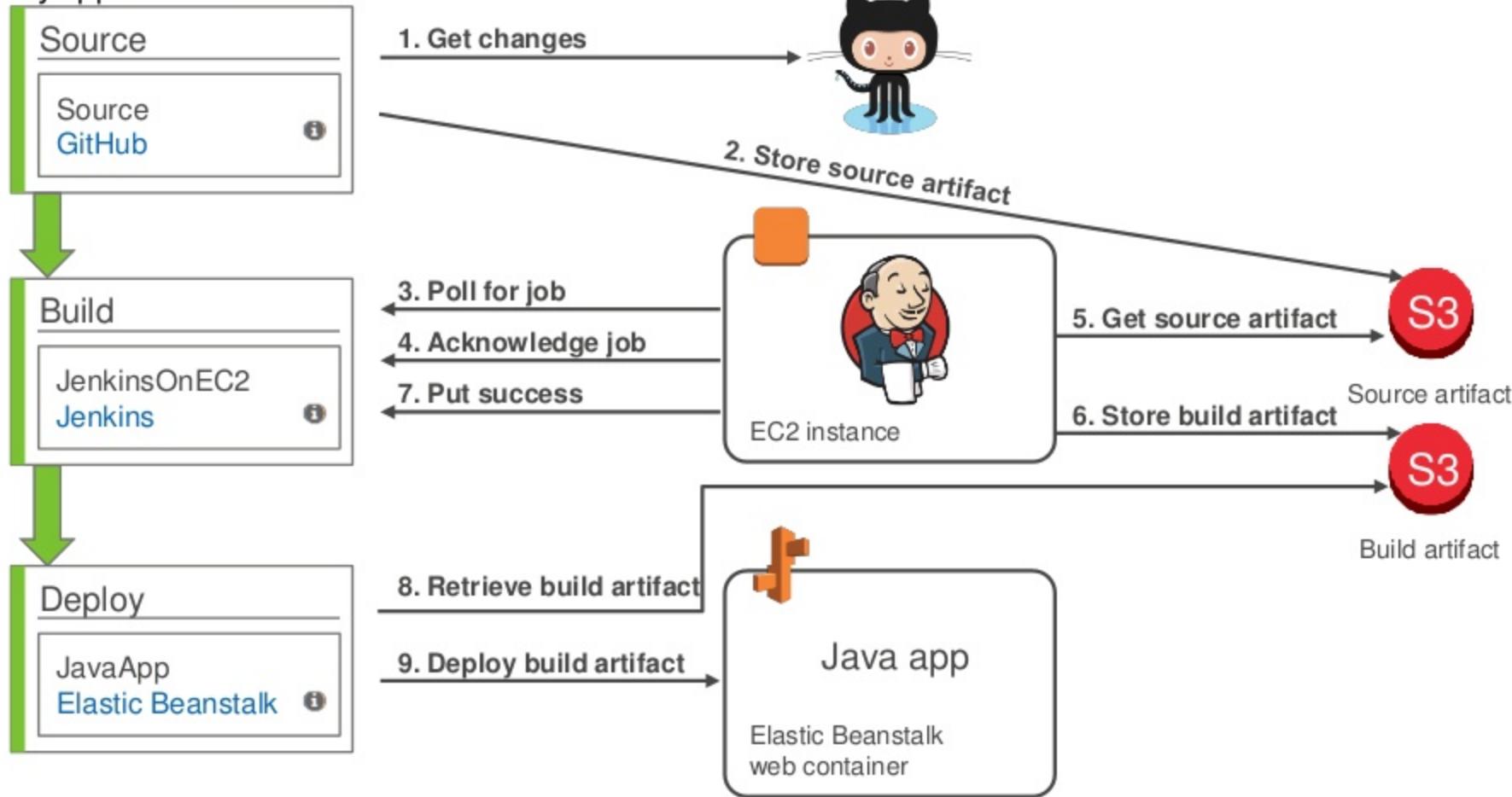


You can add a manual approval at the point where you want the pipeline to stop running until someone approves or rejects the revision in progress.

- Pipeline will stop 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
- Useful for manual QA actions or as part of “Canary” deploy models



MyApplication



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

Source

Build

Test

Deploy



AWS service integrations

Source

Invoke Logic

Deploy

Amazon S3

AWS Lambda

AWS CodeDeploy

AWS CodeCommit

AWS Elastic Beanstalk

AWS OpsWorks

A black and white photograph showing several workers in a trench, laying large, corrugated metal pipes. One worker in the foreground is kneeling, working on a pipe. Another worker's legs are visible standing on top of a pipe. The trench is deep and filled with dirt and debris. The pipes are massive, with visible longitudinal ribs.

Building your application development release pipeline

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

Code: Suits4Dogs

Code for Dogs demo application
Branch: master Clone URL:

Suits4Dogs

- script
- src
- appspec.yml
- LICENSE
- pom.xml
- README.md

README.md

aws-codedeploy-sample-tomcat
A sample Tomcat application integrated with CodeDeploy.

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)

Started 3 min 52 sec ago Took 20 sec

[add description](#)

Module Builds

SampleMavenTomcatApp Maven Webapp 16 sec

AWS CodeDeploy Deployments > Deployment d-29NPUR1FF

Deployment: d-29NPUR1FF

View information about your deployment.

Deployment Details		Revision
Deployment Succeeded		Revision Location: s3://codepipeline-us-east-1-025342911114/Wbinaer-Pipeline/b4ppblus3t4N9VSD1tag-4ba40bd23e117139ac072569fb03e-6
Application	april26demo	Revision Created: 3 minutes ago
Deployment Group	april26demo-DevEnv	Description: Application revision registered by Deployment ID: d-29NPUR1FF
Deployment ID	d-29NPUR1FF	
Deployment Config	CodeDeployDefault.OnAtTime	
Minimum Healthy Hosts	0 of 1 instances	

Instances per page: 10 Viewing 1 to 1 instances

Instance ID	Start Time	End Time	Duration	Status	Most Recent Event	Events
i-0c101bbf	3 minutes ago	3 minutes ago	32 secs	Succeeded	ValidateService	View Events

A photograph of an industrial assembly line, likely for car manufacturing. Numerous bright orange KUKA brand industrial robots are positioned along a conveyor belt, each working on a different part of a white car chassis. The robots have black grippers and are connected by a complex network of red and black cables. The background shows more machinery and equipment, with some yellow shelving units labeled with letters like 'TR', 'BR', '9R', '10R', '10L', '9L', '8L', '7L', and '6L'. The lighting is dramatic, with strong highlights on the robots and the metallic parts of the cars.

Build and test your
application

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

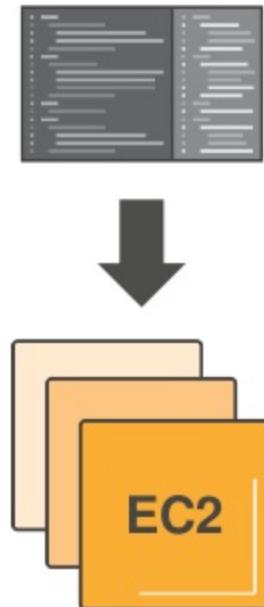
We also refer to the process of creating Docker container images as “building” the image.



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 non-desired application usage and logic failures
- Make applications more secure



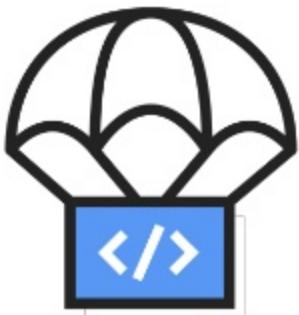
A large wooden trebuchet, a medieval siege engine, is positioned on a stone pier overlooking a calm sea under a clear blue sky. The trebuchet's arm is angled upwards, and its wooden frame is detailed with metal hardware and rivets. In the background, a stone wall runs along the pier, and a few people are walking on the grassy area behind it.

Deploying your applications

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

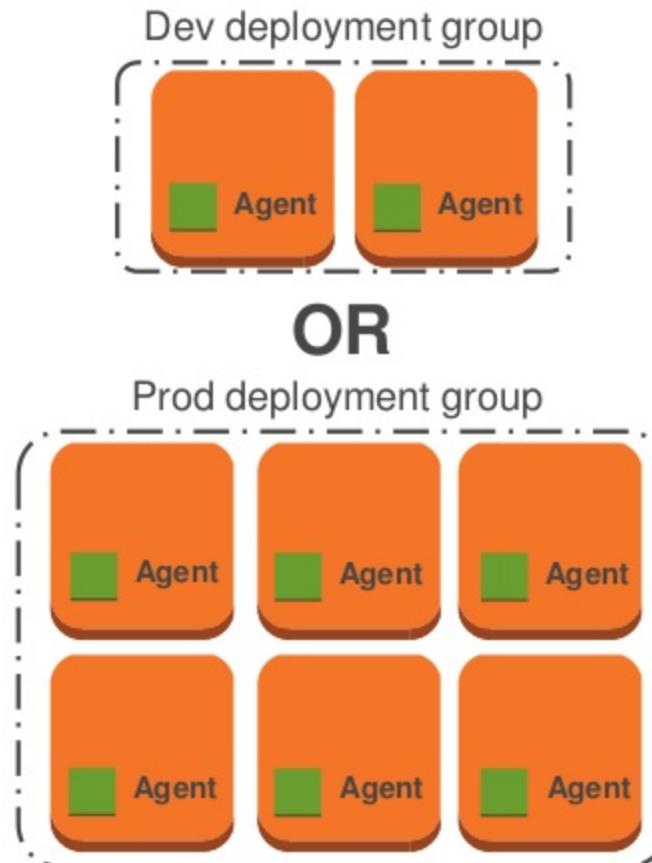
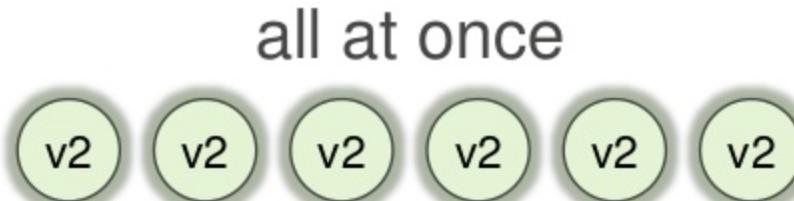
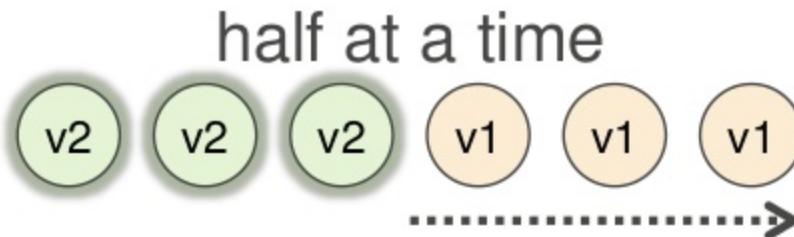
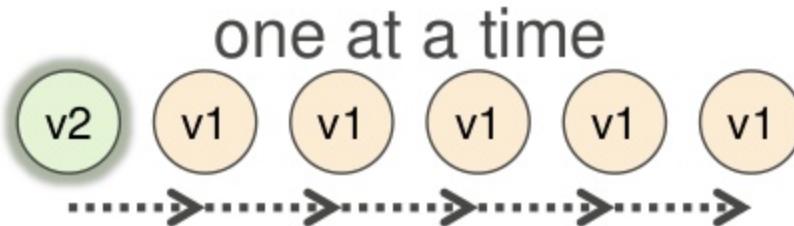
```
version: 0.0
os: linux
files:
  - source: /
    destination: /var/www/html
permissions:
  - object: /var/www/html
    pattern: "*.html"
    owner: root
    group: root
    mode: 755
hooks:
  ApplicationStop:
    - location: scripts/deregister_from_elb.sh
  BeforeInstall:
    - location: scripts/install_dependencies.sh
  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 ELB
- Install dependency packages
- Start Apache
- Confirm successful deploy
- More!

Choose deployment speed and group





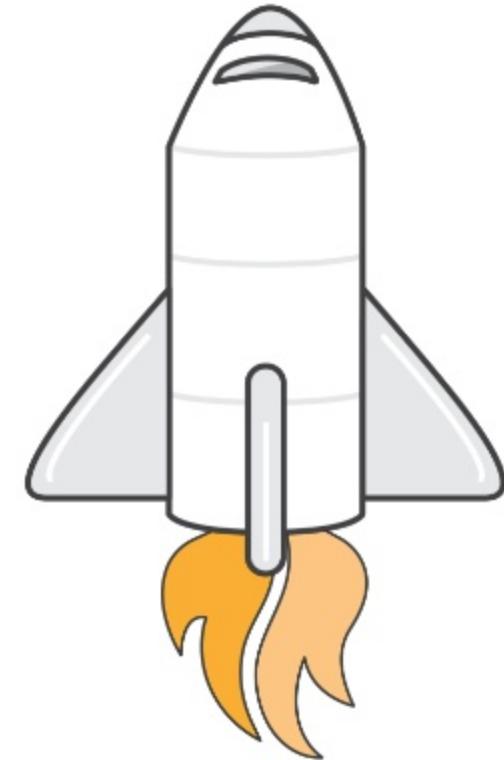
Launching to production

Launching to production

After you've built and tested your code and hopefully gone through a few pre-production deploys, it's time for the real thing!

You'll want to think about:

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



How can we track these and communicate deploys?

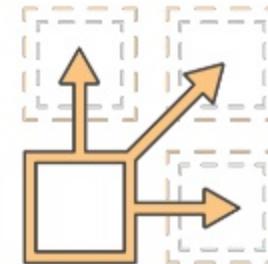
Extend AWS CodePipeline using custom actions



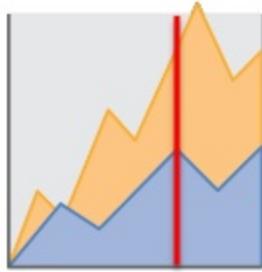
Mobile testing



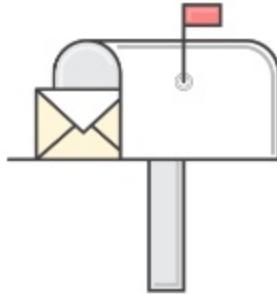
Update tickets



Provision resources



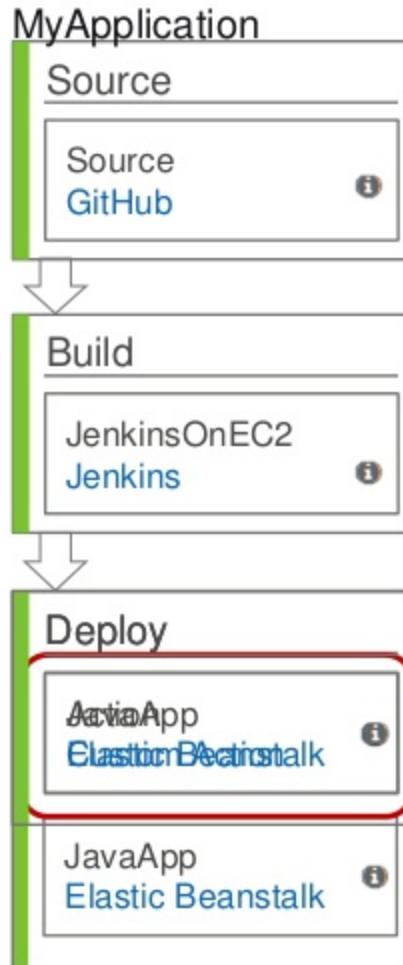
Update dashboards



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



1. Invoke Lambda function →
3. PutJobSuccessResult w/
continuation token ←
4. Invoke Lambda function w/
continuation token →
5. PutJobSuccessResult ←

2. Perform job

#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

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
- AWS 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 AWS CodePipeline and spin up a full continuous delivery pipeline using the Starter Kit

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/>
- AWS CodePipeline
 - <https://aws.amazon.com/codepipeline/>
 - <https://aws.amazon.com/documentation/codepipeline/>
- AWS CodeDeploy
 - <https://aws.amazon.com/codedeploy/>
 - <https://aws.amazon.com/documentation/codedeploy/>
 - <https://github.com/awslabs/aws-codedeploy-samples>
- AWS Code Services Starter Kit: <http://bit.ly/AWSCodeStarterKit>



Thank you!