DevOps on AWS:

Deep Dive on Continuous Delivery and the AWS Developer Tools

Michael Needham, Snr Manager Solution Architecture – RMEA

July 2016


Why are we here today?
https://secure.flickr.com/photos/mgifford/4525333972

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:

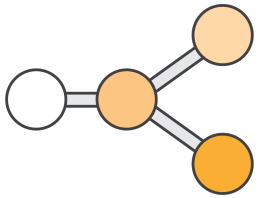
- Tools to manage the flow of your software development release process
- Tools to properly test and inspect your code for defects and potential issues
- Tools to deploy your applications

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

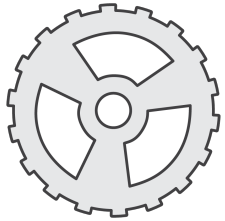
https://www.flickr.com/photos/jurvetson/5201796697/

Release processes have four major phases

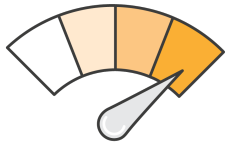
- Check-in source code such as .java files.
- Peer review new code



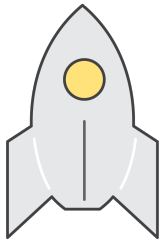
- Compile code
- Unit tests
- Style checkers
- Code metrics
- Create container images



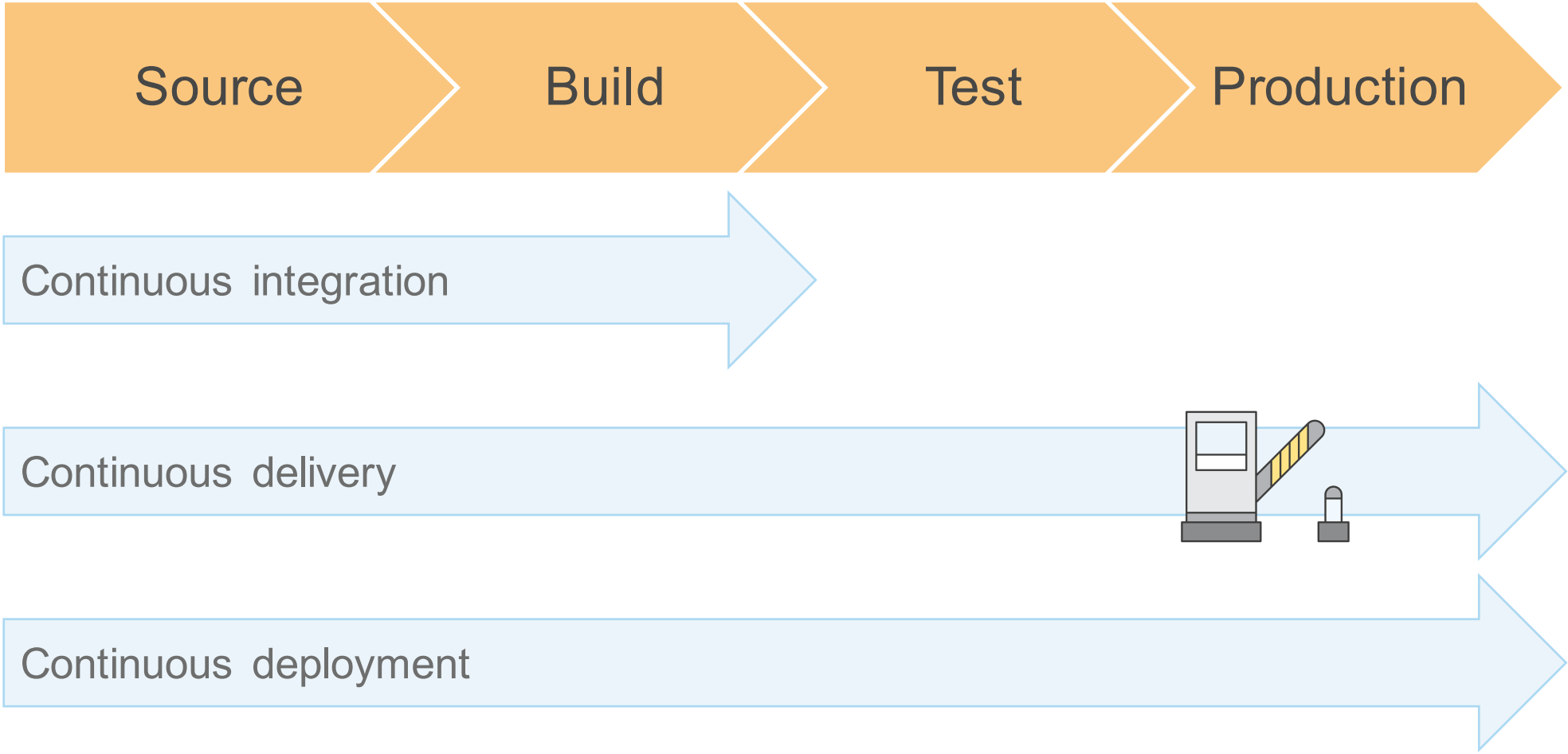
- Integration tests with other systems
- Load testing
- UI tests
- Penetration testing



- Deployment to production environments



Release processes levels



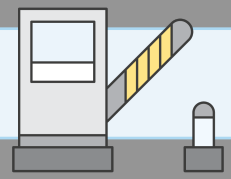
Release Processes levels

Source Build Test Production

Our focus today

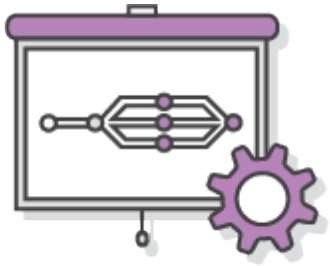
Continuous integration

Continuous delivery



Continuous deployment

Continuous Delivery Benefits



Automate the software release process

Improve developer productivity

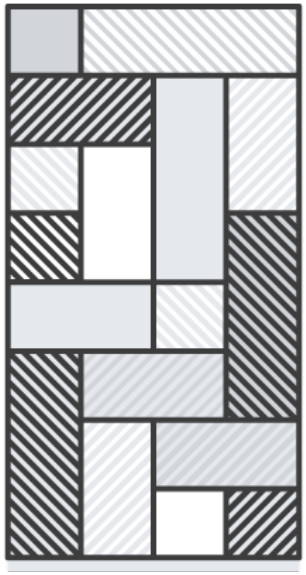
Find and address bugs quickly

Deliver updates faster


A look back at development at Amazon..
https://secure.flickr.com/photos/pixelthing/15806918992/

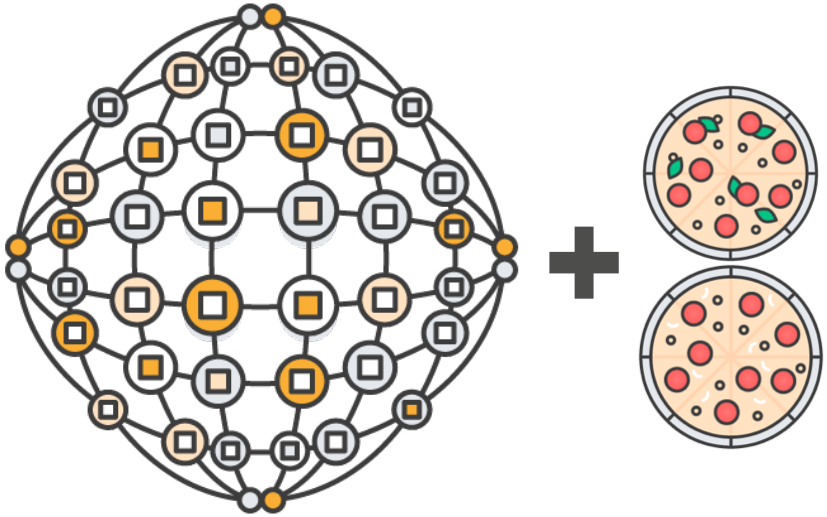
Development transformation at Amazon: 2001-2009

2001



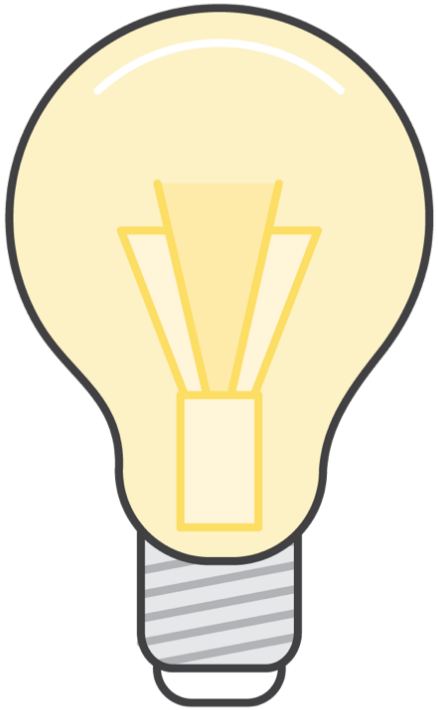
monolithic
application + teams

2009



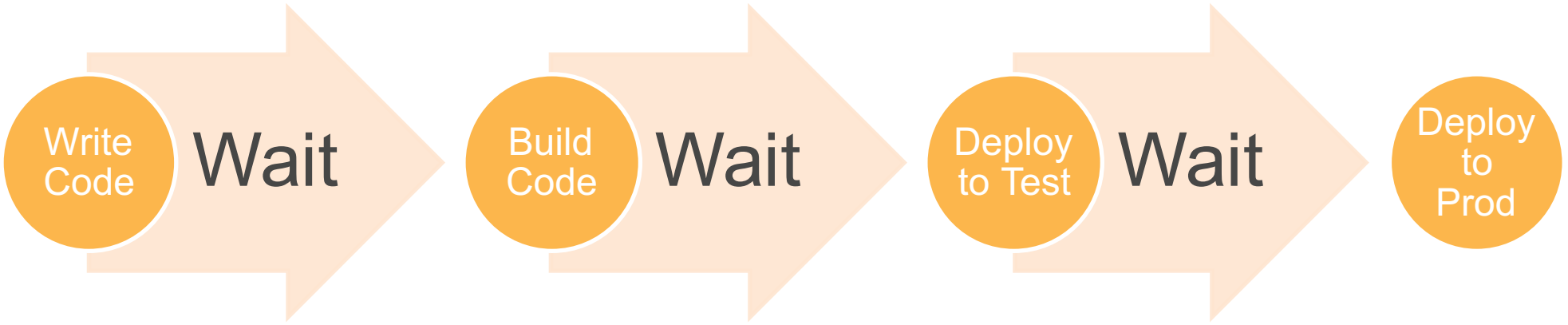
microservices + 2 pizza teams

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

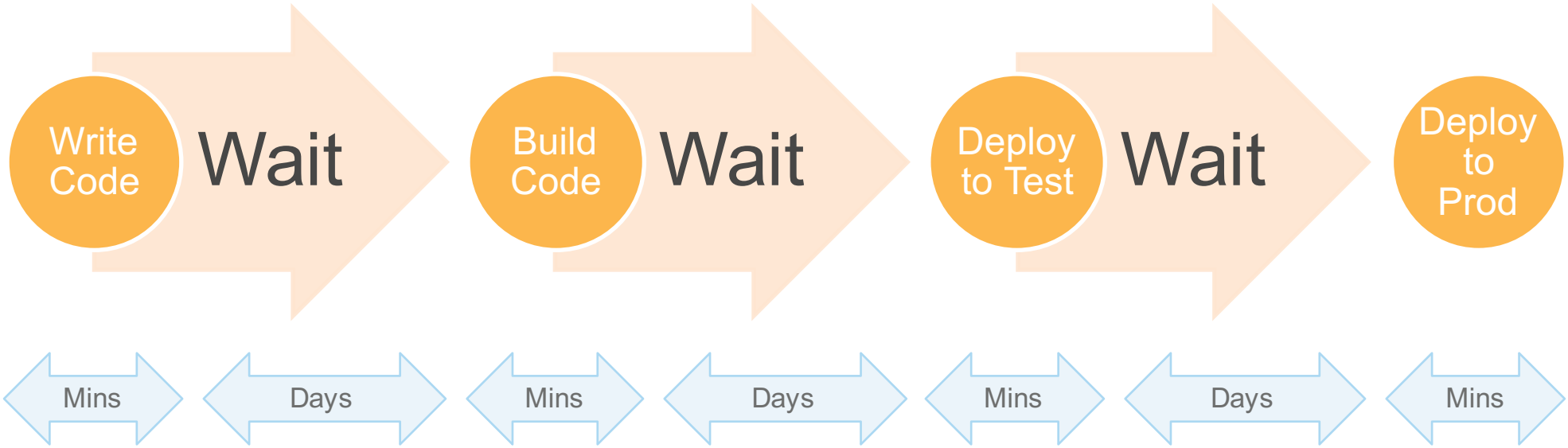


In 2009, we ran a study to find out where inefficiencies might still exist

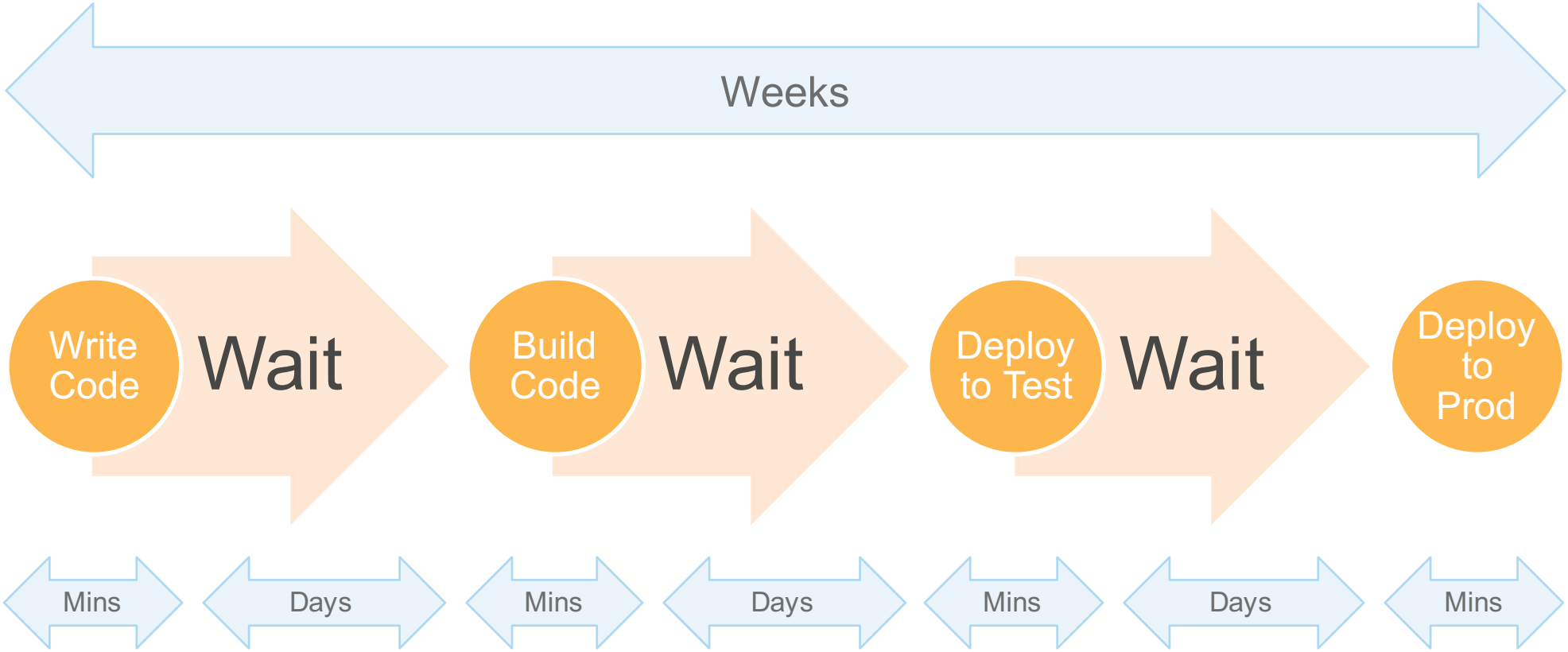
We were just waiting.



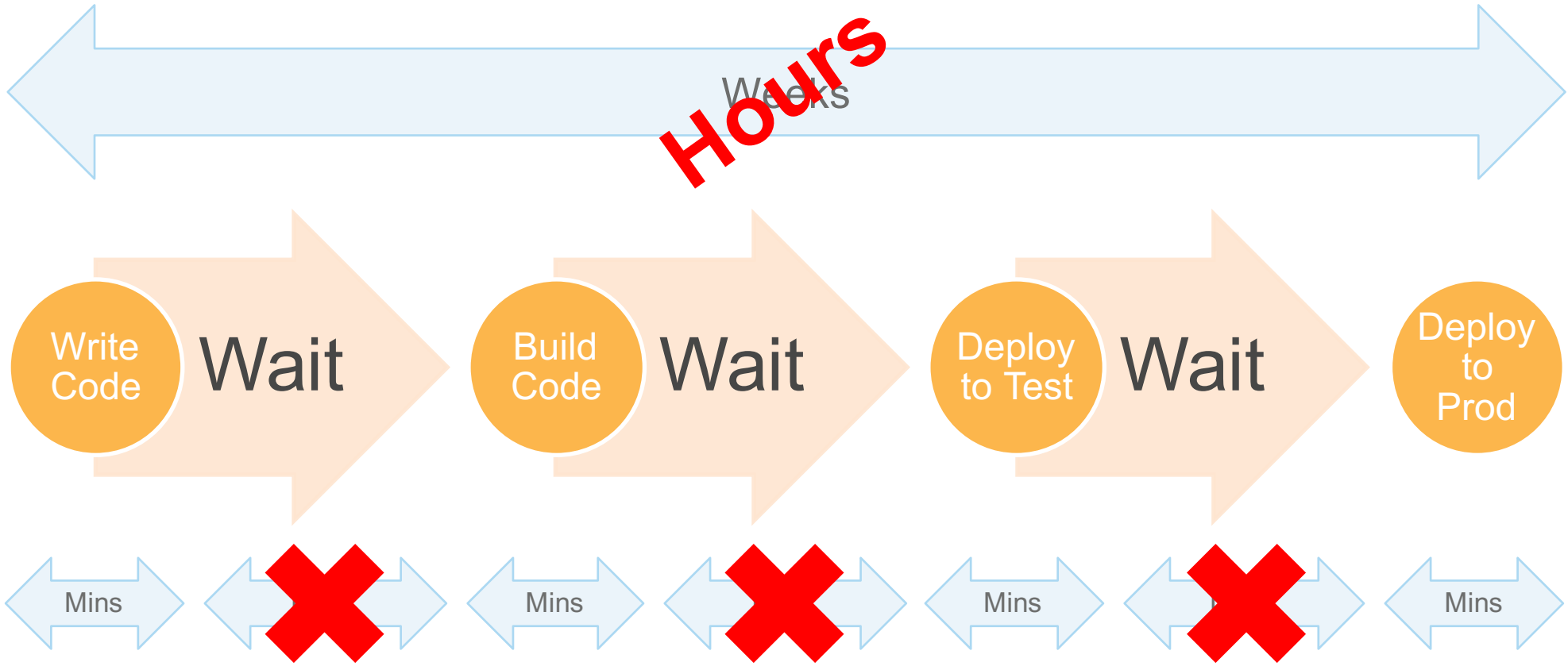
We were just waiting.



We were just waiting.



We were just waiting.




We built tools to automate our software release process
https://secure.flickr.com/photos/lindseygee/5894617854/

Automated actions and transitions; from check-in to production

Development benefits:

- Faster
- Safer
- Simplification & standardization
- 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 deploys

This has continued to work out really well:

This equates to roughly:

- 1000+ deploys every hour
- 10,000 hosts deployed to simultaneously.Max hosts 30,000 hosts

Every 11.6 seconds (production only)

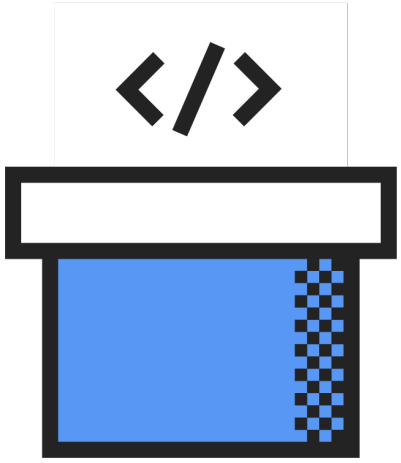
This has continued to work out really well:

Every year at Amazon, we perform a survey of all our software developers. The 2014 results found only one development tool/service could be correlated statistically with happier developers:

Our pipelines service!

continuous delivery == happier developers!

AWS CodePipeline



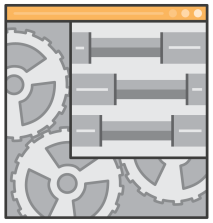
Continuous delivery service for fast and reliable application updates

Model and visualize your software release process

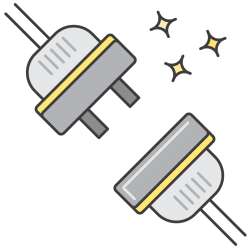
Builds, tests, and deploys your code every time there is a code change

Integrates with 3rd party tools and AWS

AWS CodePipeline Benefits

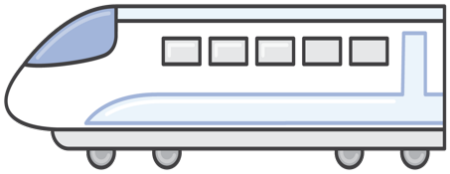


Configurable workflow



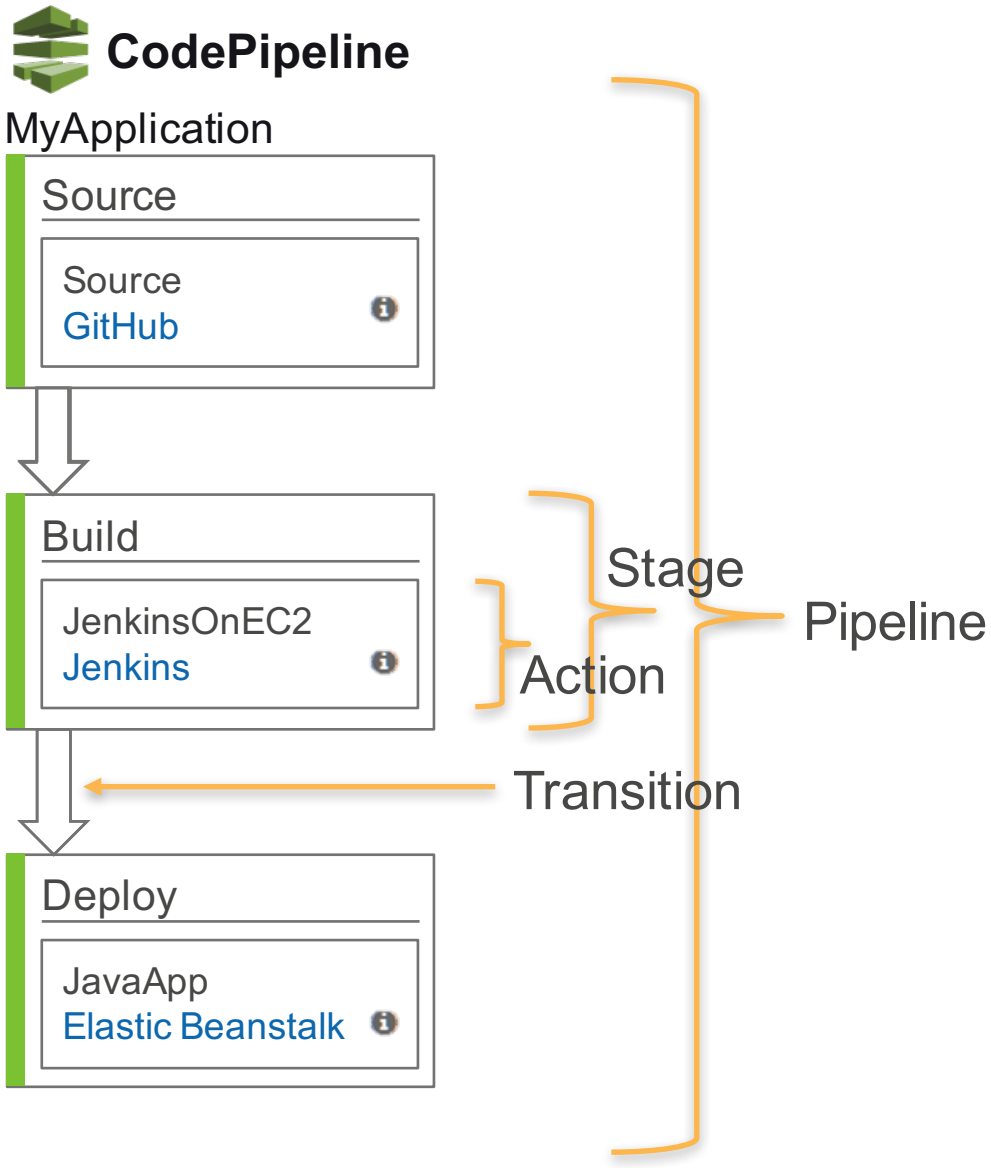
Easy to integrate

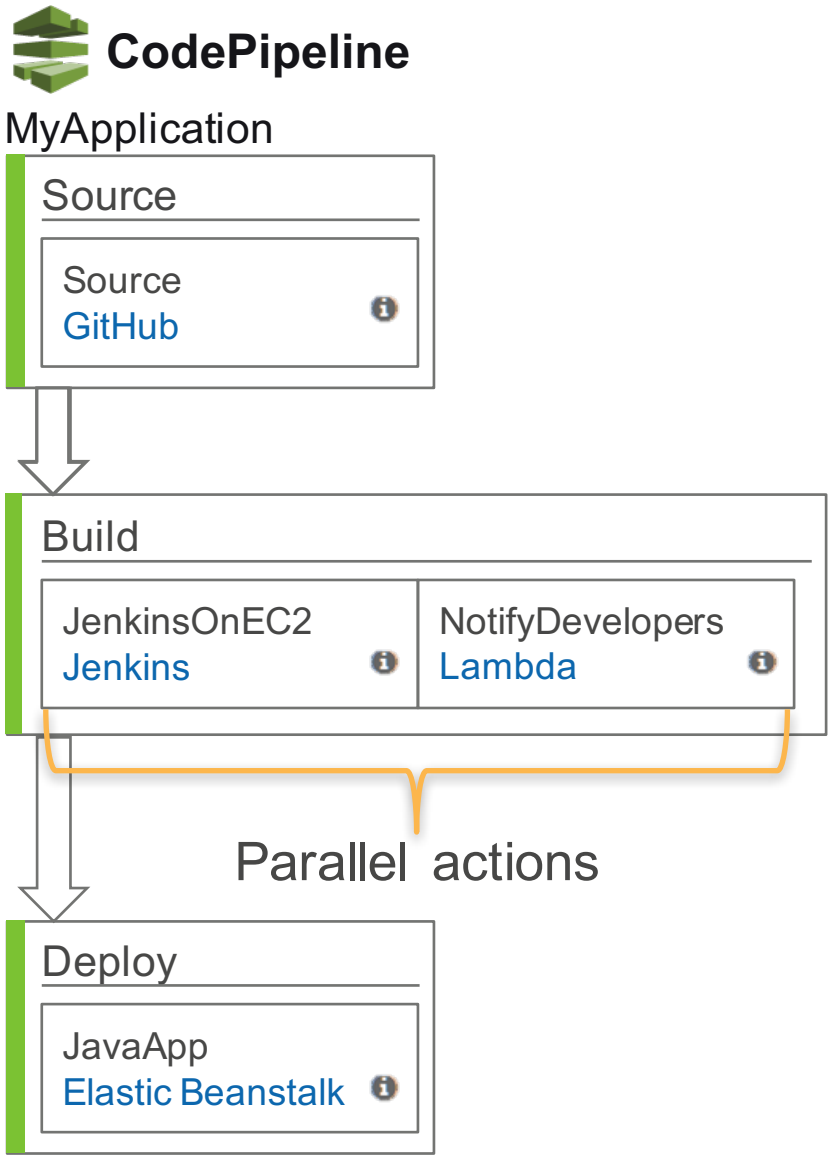
Improved quality

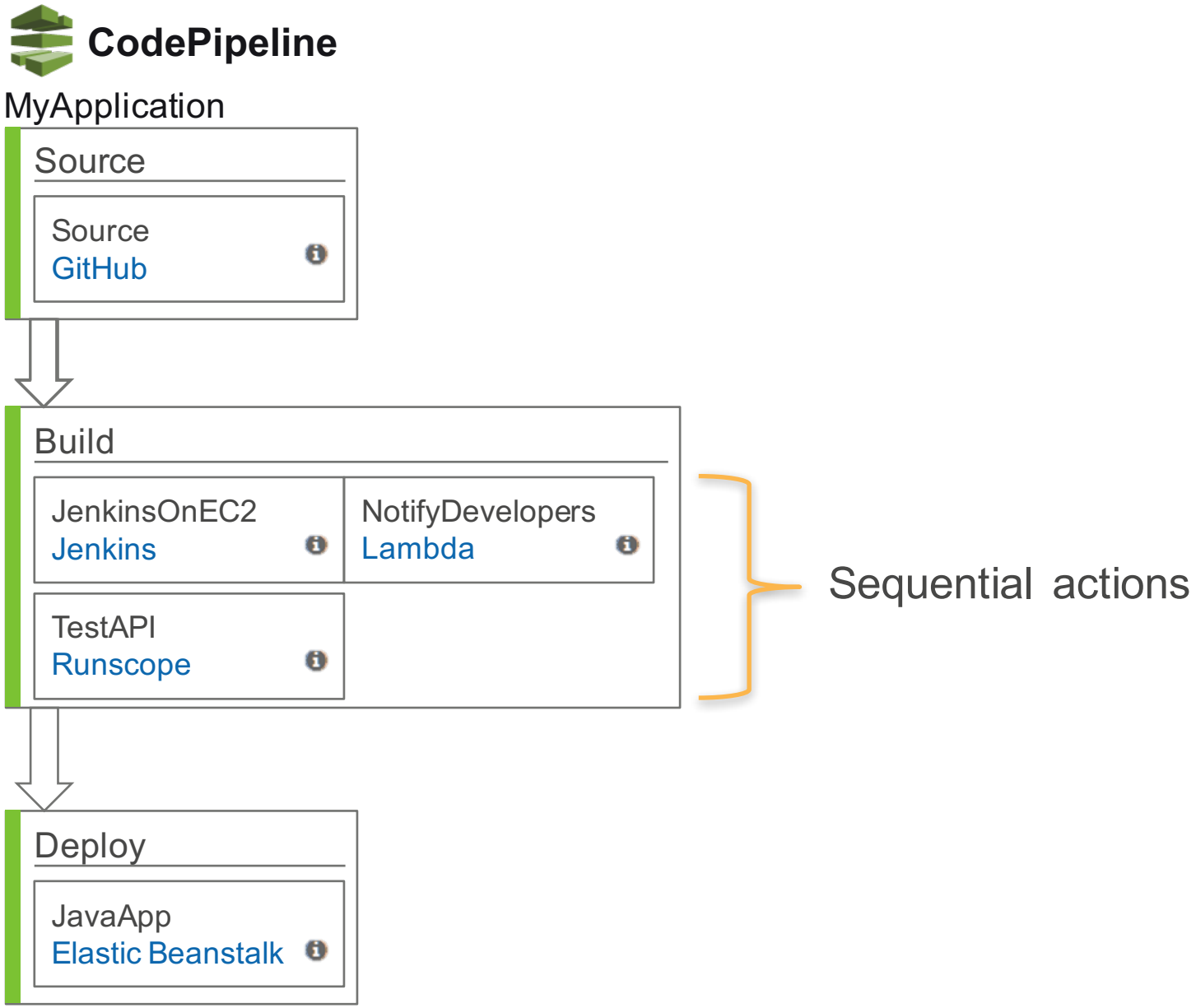


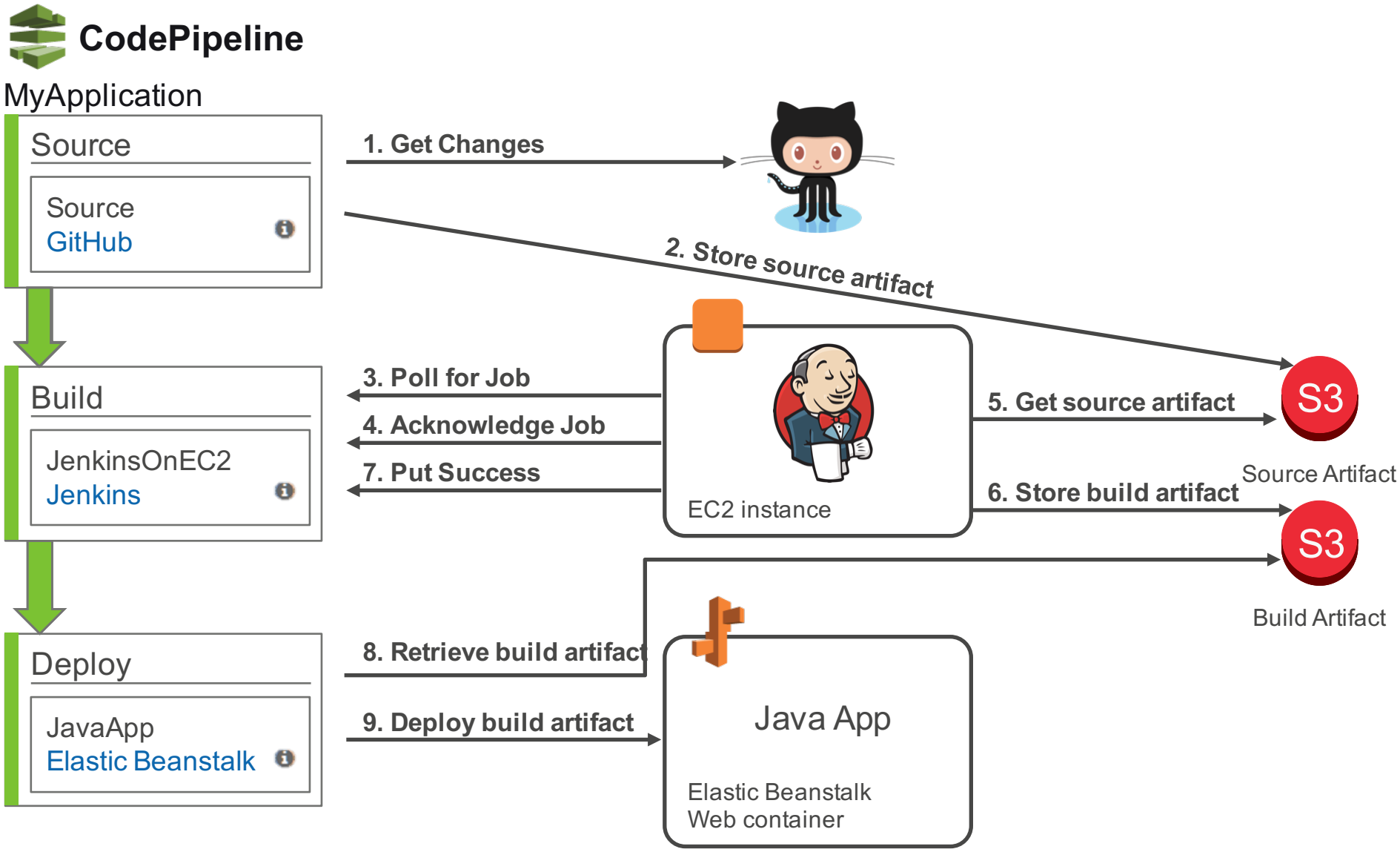
Rapid delivery

Get started fast


CodePipeline
MyApplication
Source
Source
GitHub
Build
JenkinsOnEC2
Jenkins
Deploy
JavaApp
Elastic Beanstalk
Stage
Action
Pipeline
Transition


CodePipeline
MyApplication
Source
Source
GitHub
Build
JenkinsOnEC2
Jenkins
NotifyDevelopers
Lambda
Parallel actions
Deploy
JavaApp
Elastic Beanstalk


CodePipeline
MyApplication
Source
Source
GitHub
Build
JenkinsOnEC2
Jenkins
NotifyDevelopers
Lambda
TestAPI
Runscope
Sequential actions
Deploy
JavaApp
Elastic Beanstalk


CodePipeline
MyApplication
Source
Source
GitHub
Build
JenkinsOnEC2
Jenkins
Deploy
JavaApp
Elastic Beanstalk
1. Get Changes
2. Store source artifact
3. Poll for Job
4. Acknowledge Job
7. Put Success
EC2 instance
5. Get source artifact
6. Store build artifact
S3
Source Artifact
S3
Build Artifact
8. Retrieve build artifact
9. Deploy build artifact
Java App
Elastic Beanstalk
Web container

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

GitHub

CloudBees

Apica

XebiaLabs
Deliver Faster

BlazeMeter

Jenkins

Ghost Inspector

Solano Labs

HPE StormRunner

Runscope

AWS service integrations

Source	Invoke Logic	Deploy
Amazon S3	AWS Lambda	AWS Elastic Beanstalk
Code Commit		AWS CodeDeploy


Building your application development release pipeline
https://www.flickr.com/photos/seattlemunicipalarchives/12504672623/

DEMO!


An epic drama
of adventure
and exploration
MGM PRESENTS A STANLEY KUBRICK PRODUCTION
2001
a space
odyssey
STARRING
KEIR DULLEA · GARY LOCKWOOD
SCREENPLAY BY
STANLEY KUBRICK AND ARTHUR C. CLARKE
PRODUCED AND DIRECTED BY
STANLEY KUBRICK
SUPERPANAVISION®
and METROCOLOR
MGM

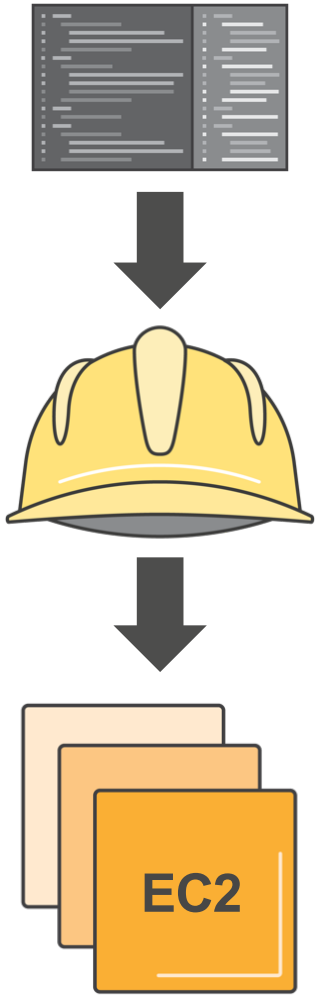

Build & test your application
https://secure.flickr.com/photos/spenceyc/7481166880

Building Your Code

"Building" code typically refers to languages that require compiled binaries:

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

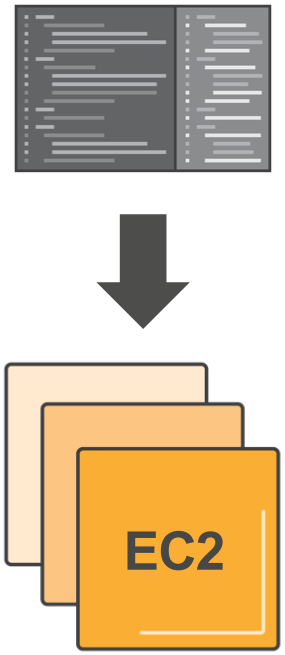
We also refer to the process of creating Docker container images as "building" the image.



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!

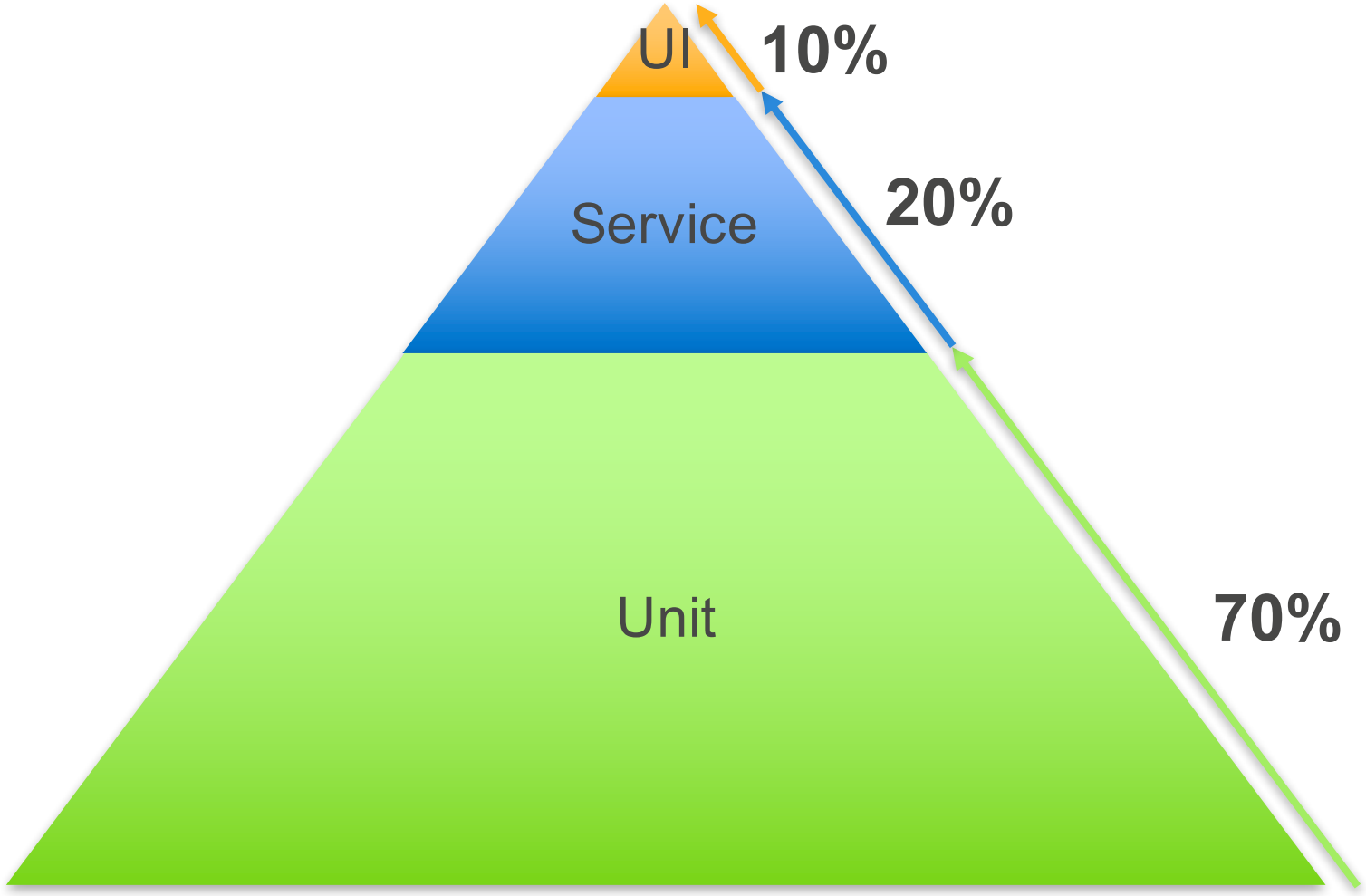
Goals for testing your code:

- 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

Where to Start with Testing

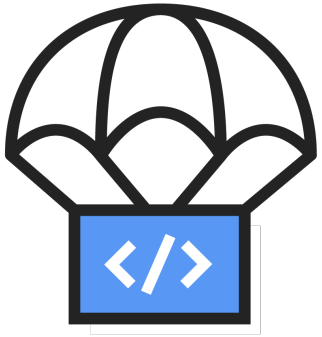
1. Start with a generic syntax linter
    - Are you missing brackets, commas, etc.?
2. Work on the top of the pyramid initially to test user facing experience/functionality
    - Are your pages rendering properly?
3. Move on to deeper UI/service checks
    - Are these two components in sync?
4. Start unit tests against basic user functions and move deeper into application logic
    - Invest more time and effort here as you find bugs or failures in production

Where to Focus Your Tests:




Deploying your applications
https://secure.flickr.com/photos/simononly/15386966677

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 3rd party tools and AWS

appspec.yml Example

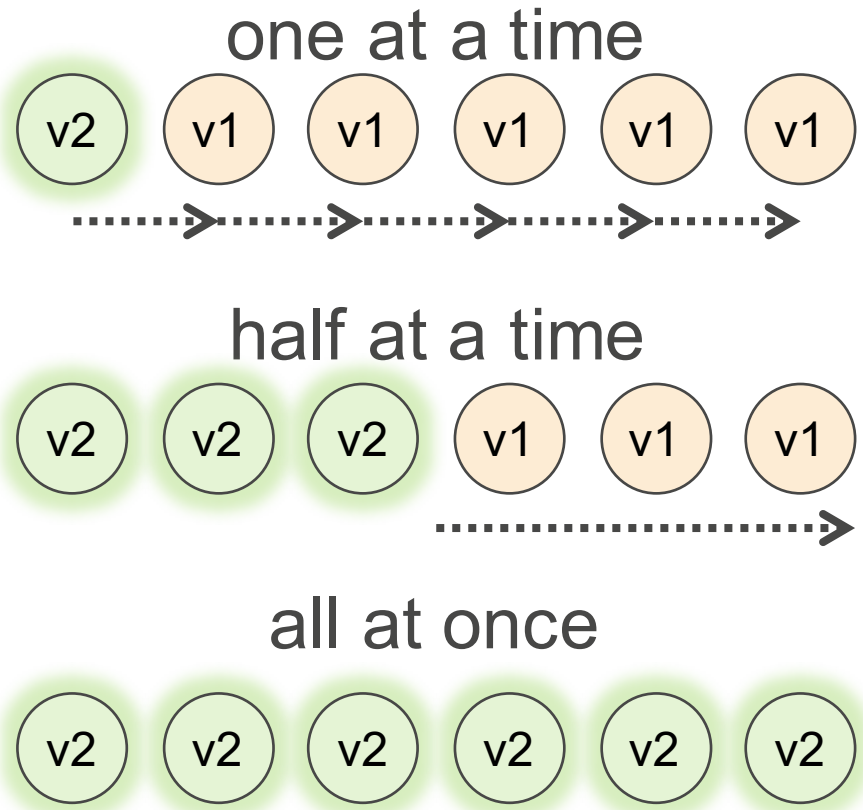
```
version: 0.0
os: linux
files:
  - source: /
    destination: /var/www/html
permissions:
  - object: /var/www/html
    pattern: “*.html”
    owner: root
    group: root
    mode: 755
hooks:
  ApplicationStop:
    - location: scripts/deregister_from_elb.sh
  BeforeInstall:
    - location: scripts/install_dependencies.sh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

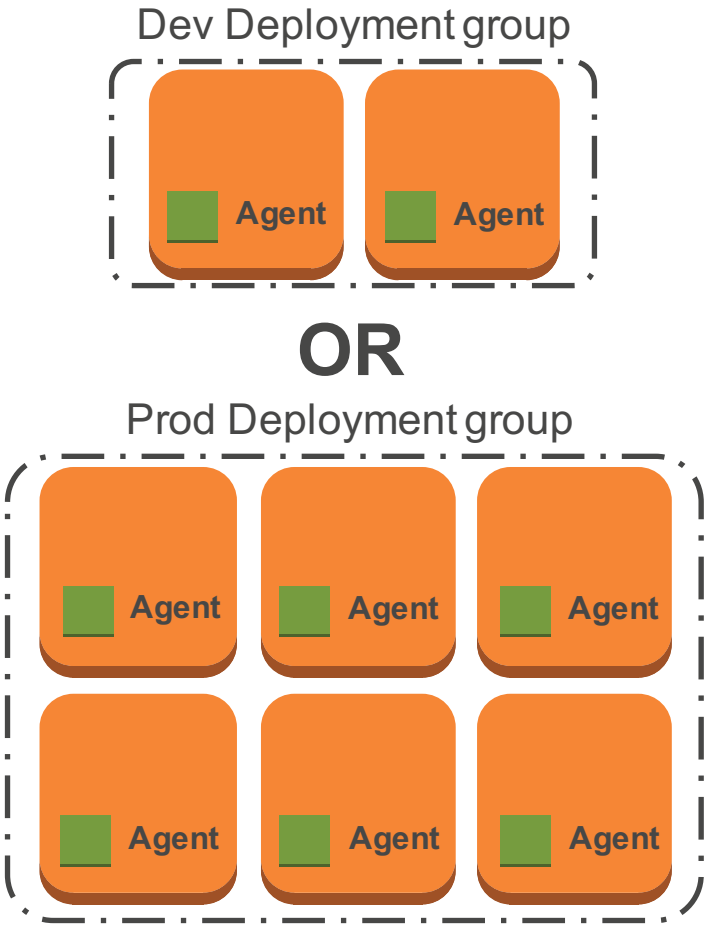
appspec.yml Example

```
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 & files
- Remove/add instance to ELB
- Install dependency packages
- Start Apache
- Confirm successful deploy
- More!

Choose Deployment Speed and Group





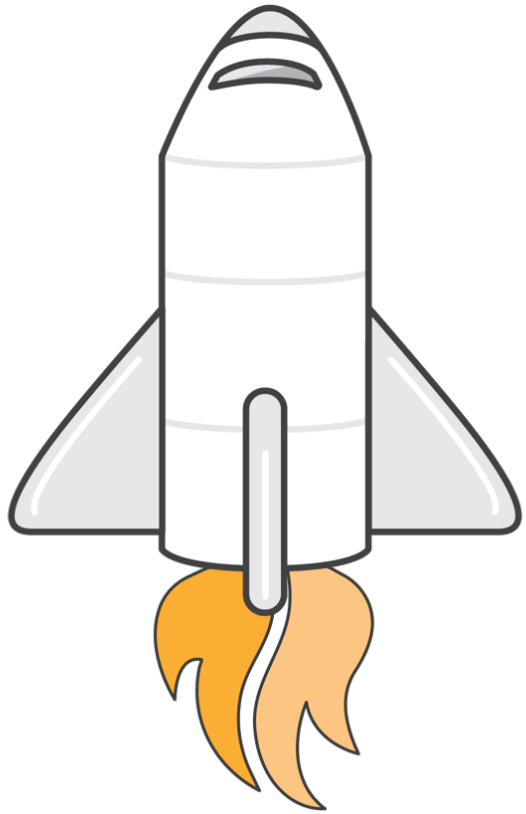

Launching to
Production
https://www.flickr.com/photos/spacex/16510243060/

Launching to Production

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

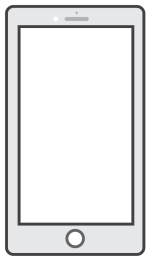
You'll want think about:

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

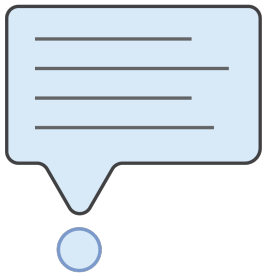


How can we track these and communicate deploys?

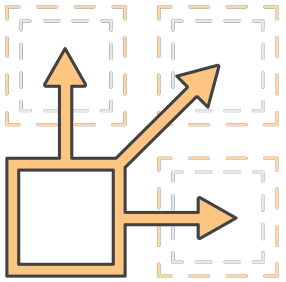
Extend AWS CodePipeline Using Custom Actions



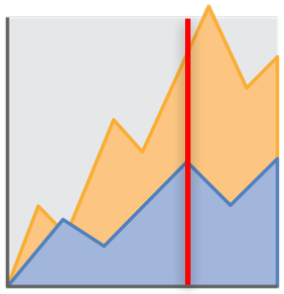
Mobile testing



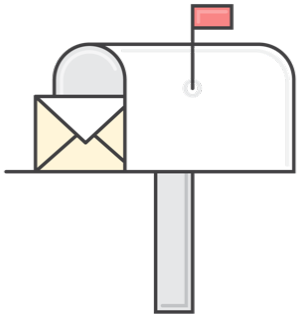
Update tickets



Provision resources

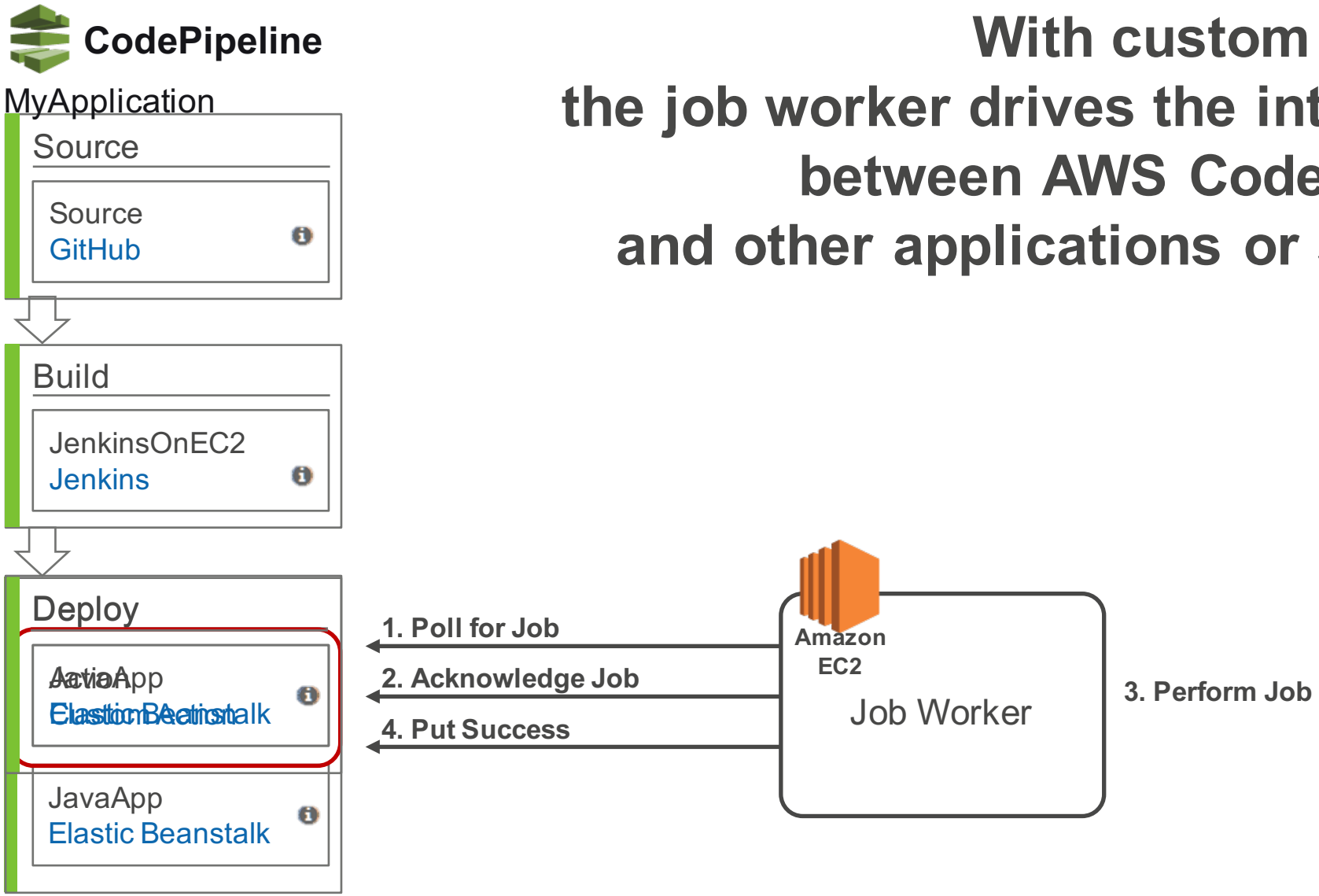


Update dashboards

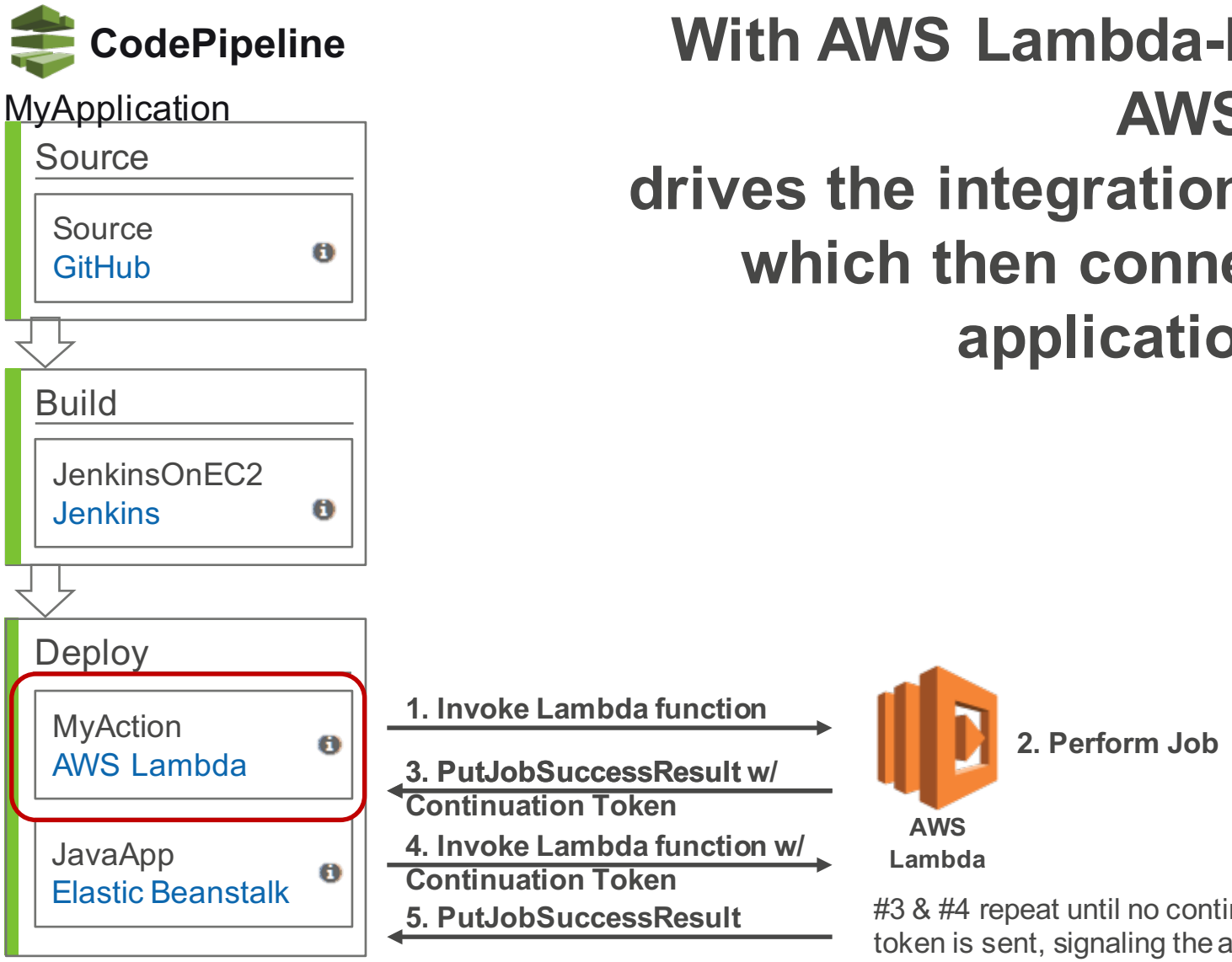


Send notifications

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

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
- 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

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!

amazon web services

Customer

amazon web services

LifeQ Cloud Platform

LifeQ is a health informatics company with a cloud-based, bio-mathematical engine and API. The engine can ingest various physiological data streams to construct a highly personalized, digital representation of an individual's unique physiology.

LifeQ Cloud Platform

A platform consisting of hosted internet services that enables the bio-mathematical algorithms and Virtual Human Model (VHM) to receive data from integrated data sources and makes the analytics available to other parties in the ecosystem.

LifeQ Cloud Platform

A platform consisting of hosted internet services that enables the bio-mathematical algorithms and Virtual Human Model (VHM) to receive data from integrated data sources and makes the analytics available to other parties in the ecosystem.

Challenges

- Scaling mathematical models (multiple languages)
- IoT data stream scaling (many small files)
- Speed to market (New data models monthly)
- Rapidly changing infrastructure requirements
- Compliance (HIPAA)

AWS Usage

- EC2 instances with ELBs & RDS
- EC2 Container Registry
- Prebuilt AMIs with docker containers (100s startup)
- Certificate Manager
- S3
- IAM cross-account roles & policies
- CodeDeploy
- SQS for Logstash

Benefit

- Stable
- DevOps Tooling
- Automation - APIs for everything
- Spin up testing environment in 15minutes (~35 servers)
- AWS is HIPAA compliant
- Cost per model (business decides SLA)

The Future

- Route53 geo-redundency
- Elastic Cache
- Mesos+Marathon or Kubernetes clusters