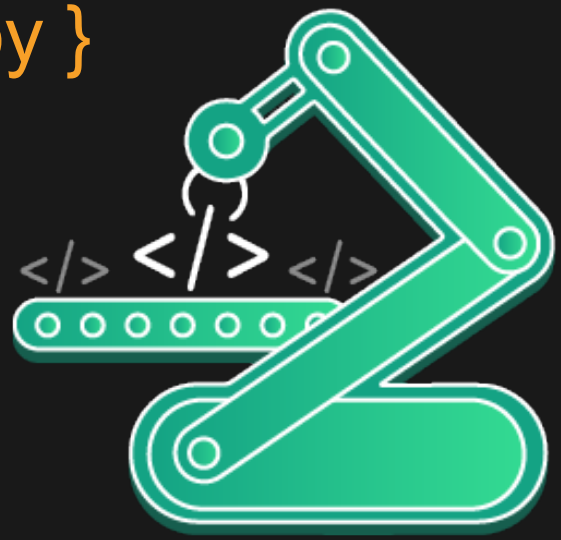
Accelerating Software Delivery

with AWS Code { Pipeline | Build | Deploy }



Glauber Gallego, Solutions Architect, AWS

Feb 2017

amazon web services | Webinars

Software moves faster today

Software creation and distribution is easier and faster than ever:

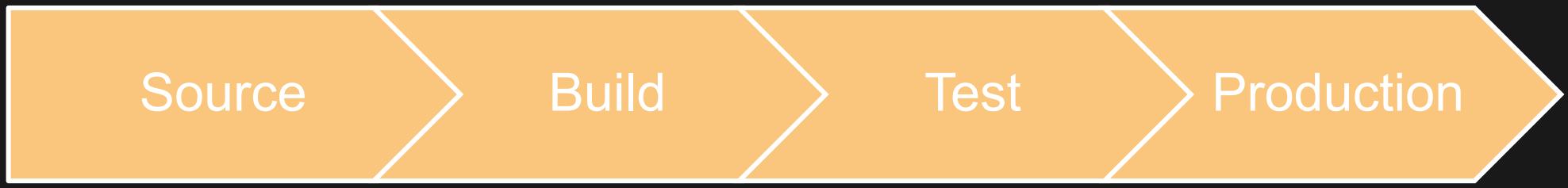
- Startups can now take on giants with little to no funding ahead of time
- Getting your software into the hands of millions is a download away
- Your ability to move fast is paramount to your ability to fight off disruption

What tools do you need to move fast?

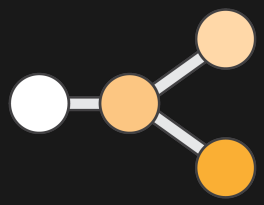
Releasing software in this new software driven world requires a number of tools:

- 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

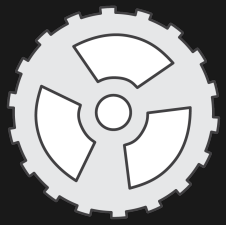
Release processes have four major phases



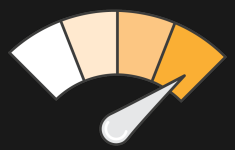
- Check-in source code such as .java files.
- Peer review new code



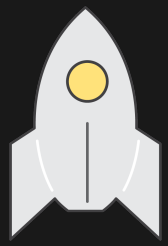
- Compile code
- Unit tests
- Style checkers
- Code metrics
- Create container images



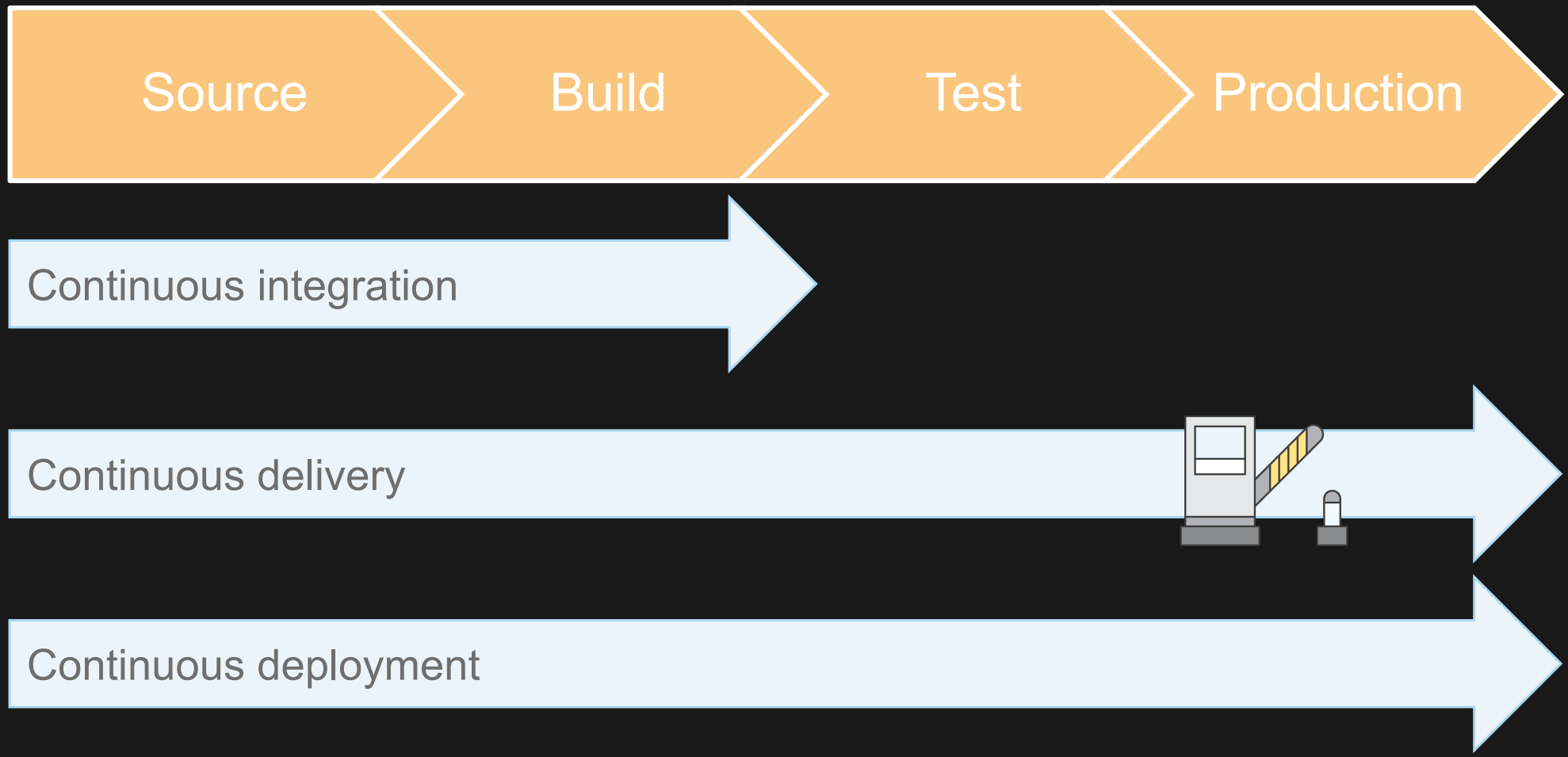
- Integration tests with other systems
- Load testing
- UI tests
- Penetration testing



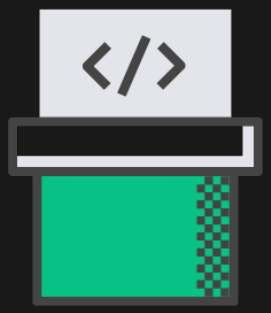
- Deployment to production environments



Release processes levels

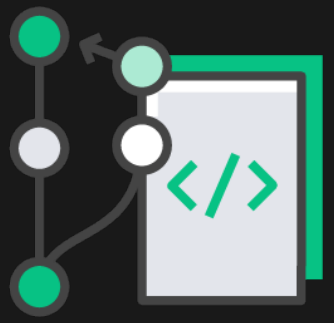


AWS Code* Services



AWS CodePipeline

AWS CodeDeploy



AWS CodeCommit

NEW!

AWS CodeBuild

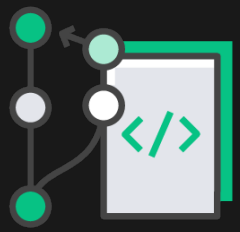
NEW!

AWS Code Services

Software Release Steps:

AWS Code Services

Software Release Steps:



AWS CodeCommit

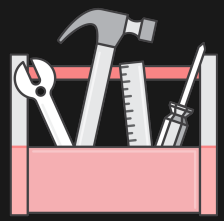
AWS Code Services

Software Release Steps:

AWS CodeBuild

AWS Code Services

Software Release Steps:



Third Party
Tooling

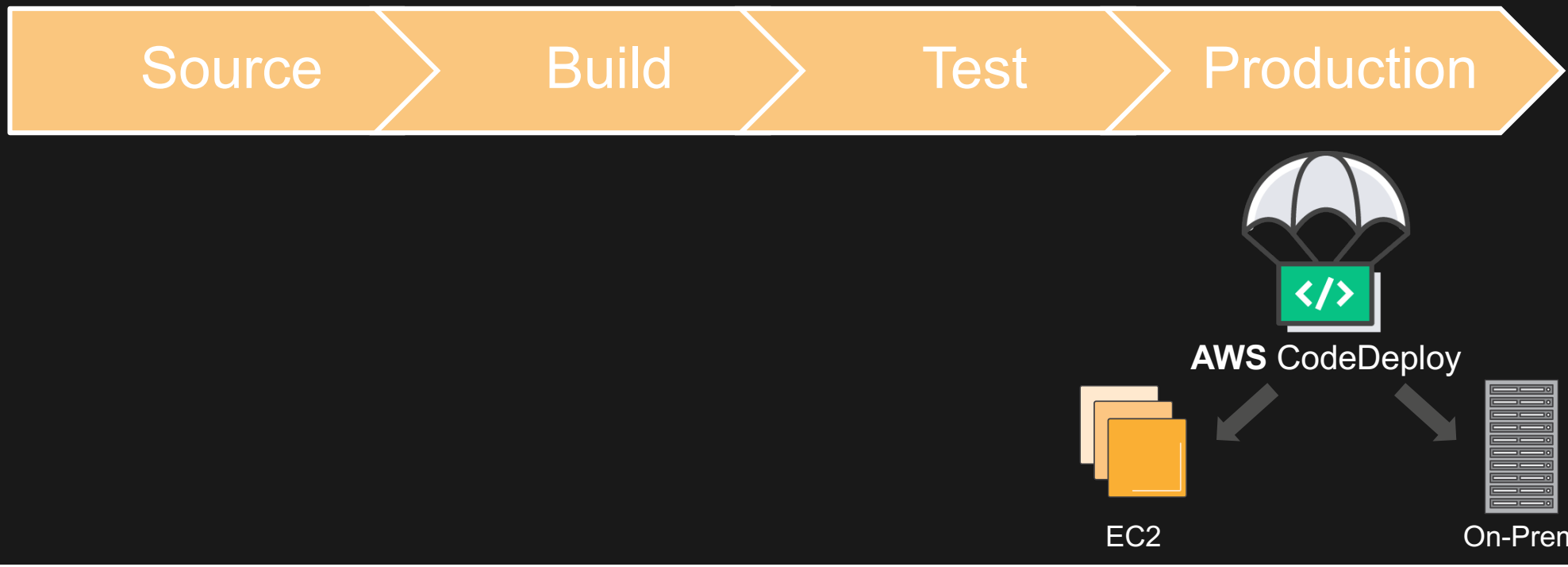
AWS Code Services

Software Release Steps:

AWS CodeDeploy

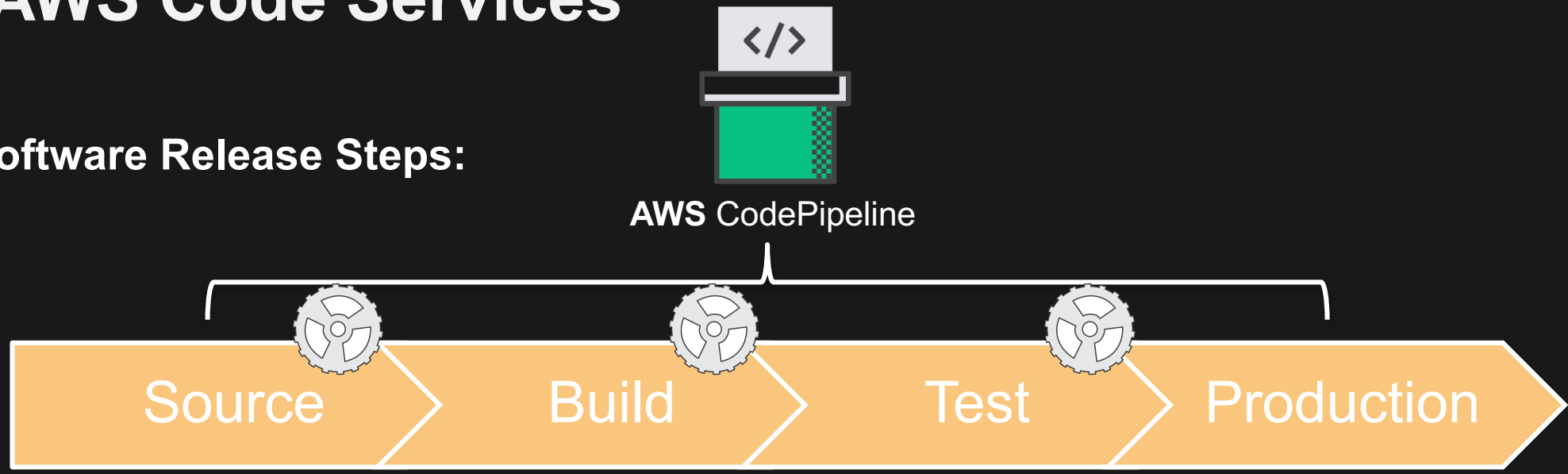
AWS Code Services

Software Release Steps:



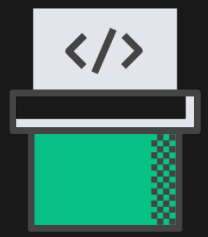
AWS Code Services

Software Release Steps:



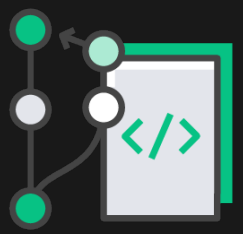
AWS Code Services

Software Release Steps:



AWS CodePipeline

Source → Build → Test → Production



AWS CodeCommit

AWS CodeBuild

Third Party Tooling

AWS CodeDeploy


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

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

Only pay by the minute for the compute resources you use

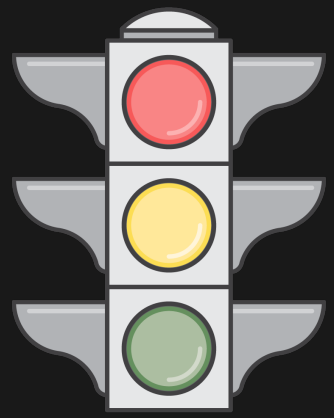
Launched with CodePipeline and Jenkins integration

How does it work?

1. Downloads source code
2. Executes commands configured in the buildspec in temporary compute containers (created fresh on every build)
3. Streams the build output to the service console and CloudWatch logs
4. Uploads the generated artifact to an S3 bucket

How can I automate my release process with CodeBuild?

- Integrated with AWS CodePipeline for CI/CD
- Easily pluggable (API/CLI driven)
- Bring your own build environments
    - Create Docker images containing tools you need
- Open source Jenkins plugin
    - Use CodeBuild as the workers off of a Jenkins master



buildspec.yml Example

```
version: 0.1

environment_variables:
  plaintext:
    JAVA_HOME: "/usr/lib/jvm/java-8-openjdk-amd64"

phases:
  install:
    commands:
      - apt-get update -y
      - apt-get install -y maven
  pre_build:
    commands:
      - echo Nothing to do in the pre_build phase...
  build:
    commands:
      - echo Build started on `date`
      - mvn install
  post_build:
    commands:
      - echo Build completed on `date`
artifacts:
  type: zip
  files:
    - target/messageUtil-1.0.jar
  discard-paths: yes
```

buildspec.yml Example

```
version: 0.1

environment_variables:
  plaintext:
    JAVA_HOME: "/usr/lib/jvm/java-8-openjdk-amd64"

phases:
  install:
    commands:
      - apt-get update -y
      - apt-get install -y maven
  pre_build:
    commands:
      - echo Nothing to do in the pre_build phase...
  build:
    commands:
      - echo Build started on `date`
      - mvn install
  post_build:
    commands:
      - echo Build completed on `date`
artifacts:
  type: zip
  files:
    - target/messageUtil-1.0.jar
  discard-paths: yes
```

- Variables to be used by phases of build
- Examples for what you can do in the phases of a build:
  - You can install packages or run commands to prepare your environment in ”install”.
  - Run syntax checking, commands in “pre_build”.
  - Execute your build tool/command in “build”
  - Test your app further or ship a container image to a repository in post_build
- Create and store an artifact in S3

Building Your Code

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

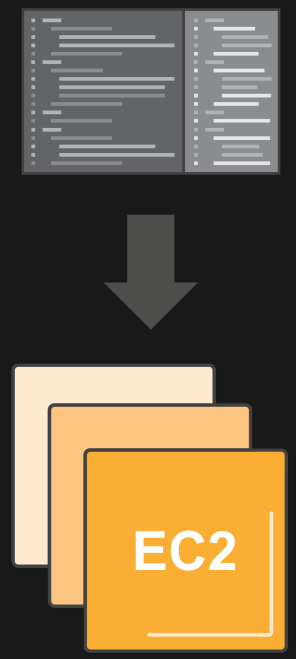
- .NET languages: C#, F#, VB.net, etc.
- Java and JVM languages: Java, Scala, JRuby
- Go
- iOS languages: Swift, Objective-C

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!

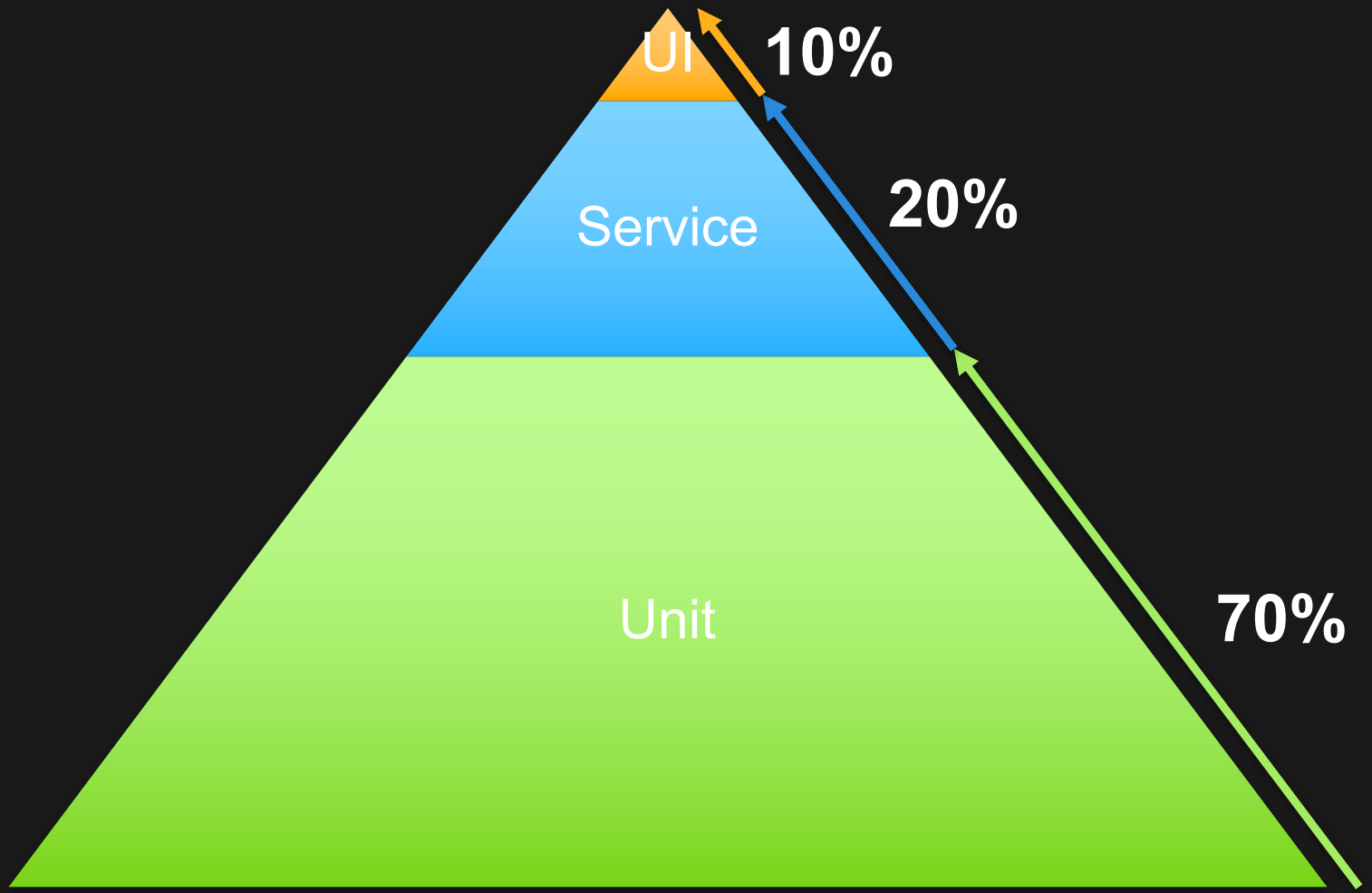
Testing Your Code

Testing is both a science and an art form!

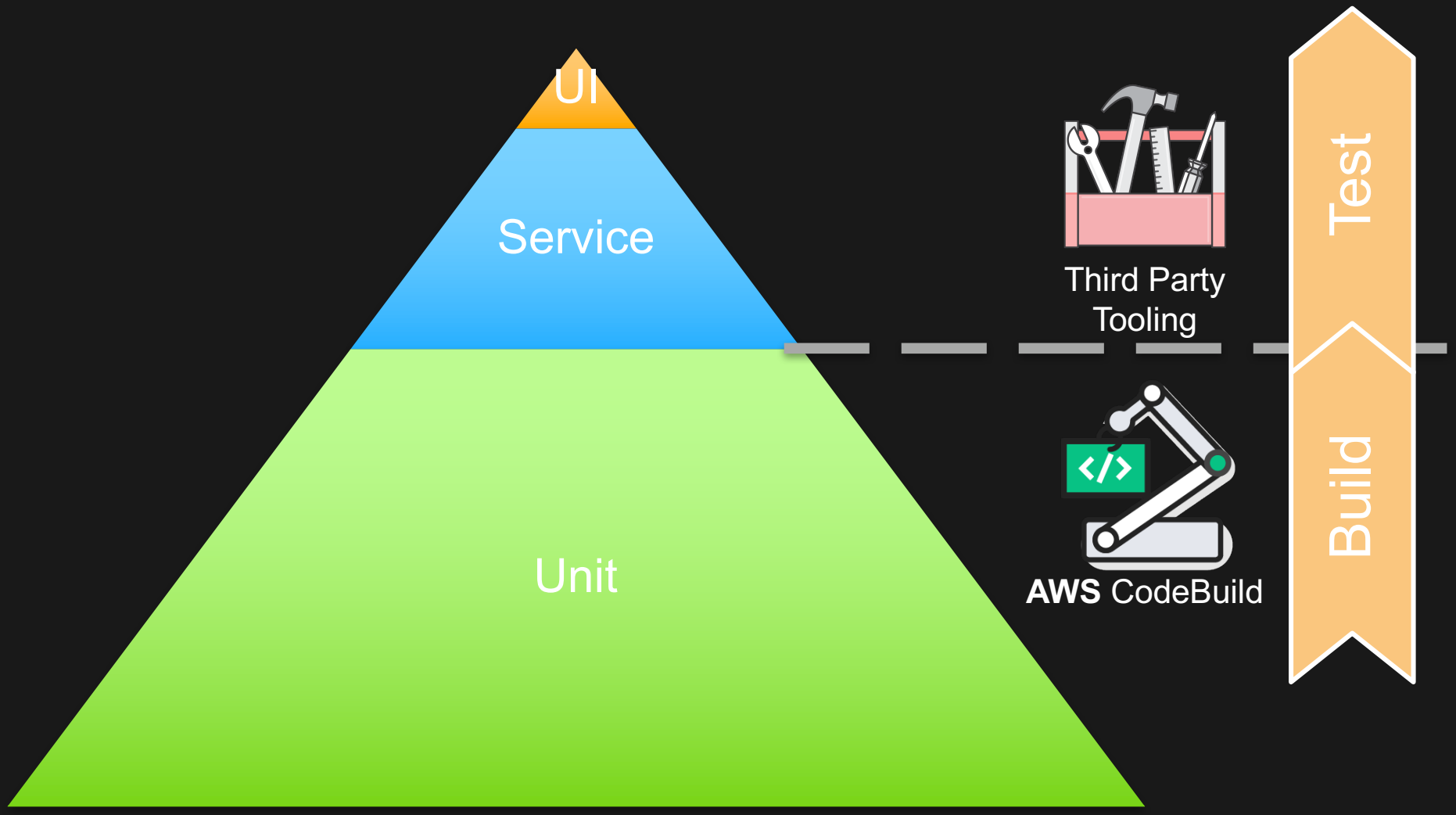
Goals for testing your code:

- 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

Where to Focus Your Tests:



What service and release step corresponds with which 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

Rollback automatically if failure detected

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

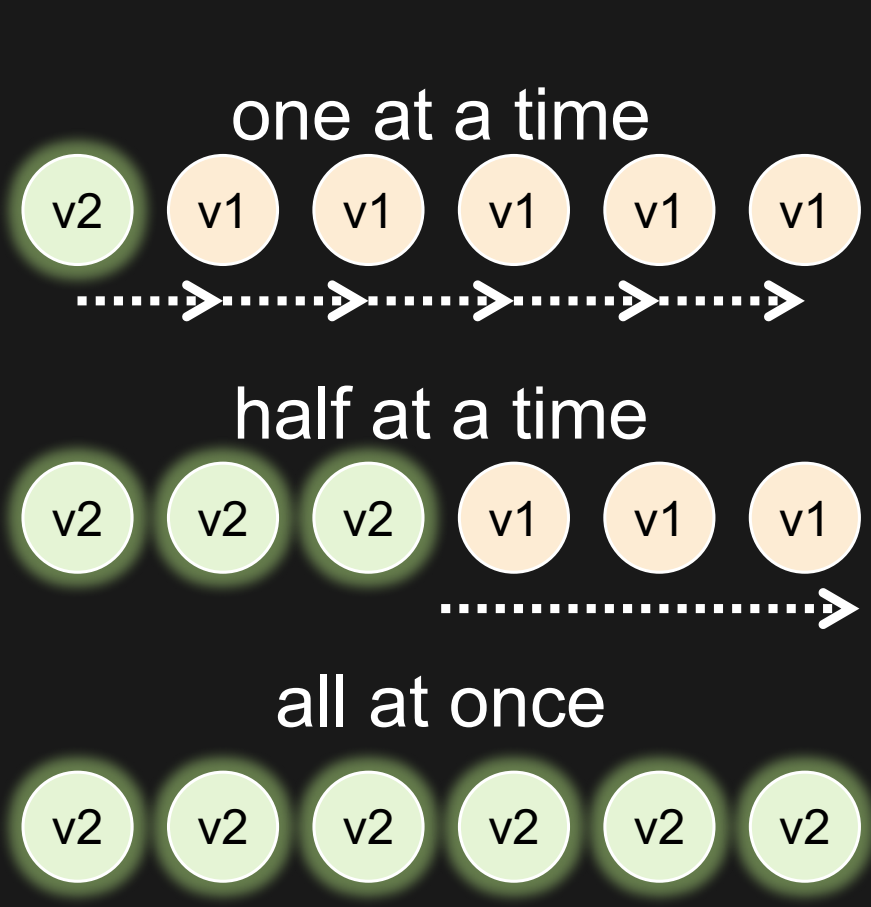
```
version: 0.0
os: linux
files:
  - source: /
    destination: /var/www/html
permissions:
  - object: /var/www/html
    pattern: “*.html”
    owner: root
    group: root
    mode: 755
hooks:
  ApplicationStop:
    - location: scripts/deregister_from_elb.sh
  BeforeInstall:
    - location: scripts/install_dependencies.sh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

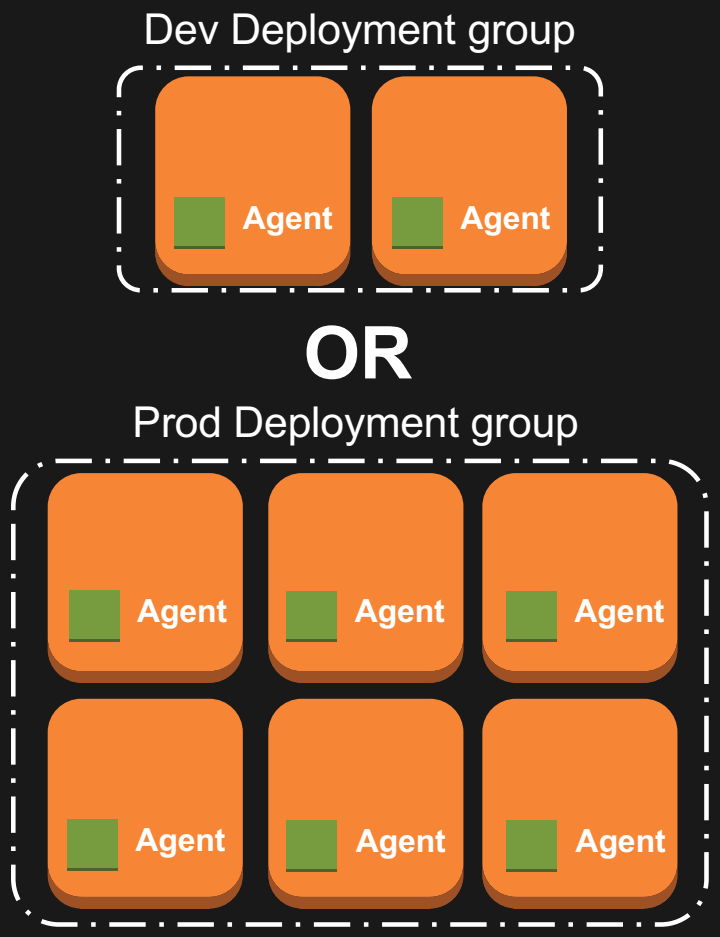
appspec.yml Example

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

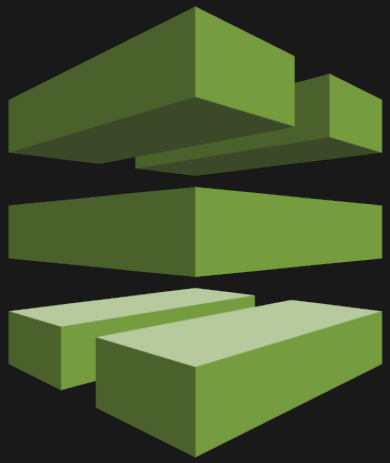
Choose Deployment Speed and Group






Orchestrating build and deploy with a pipeline
https://www.flickr.com/photos/seattlemunicipalarchives/12504672623/

AWS CodePipeline

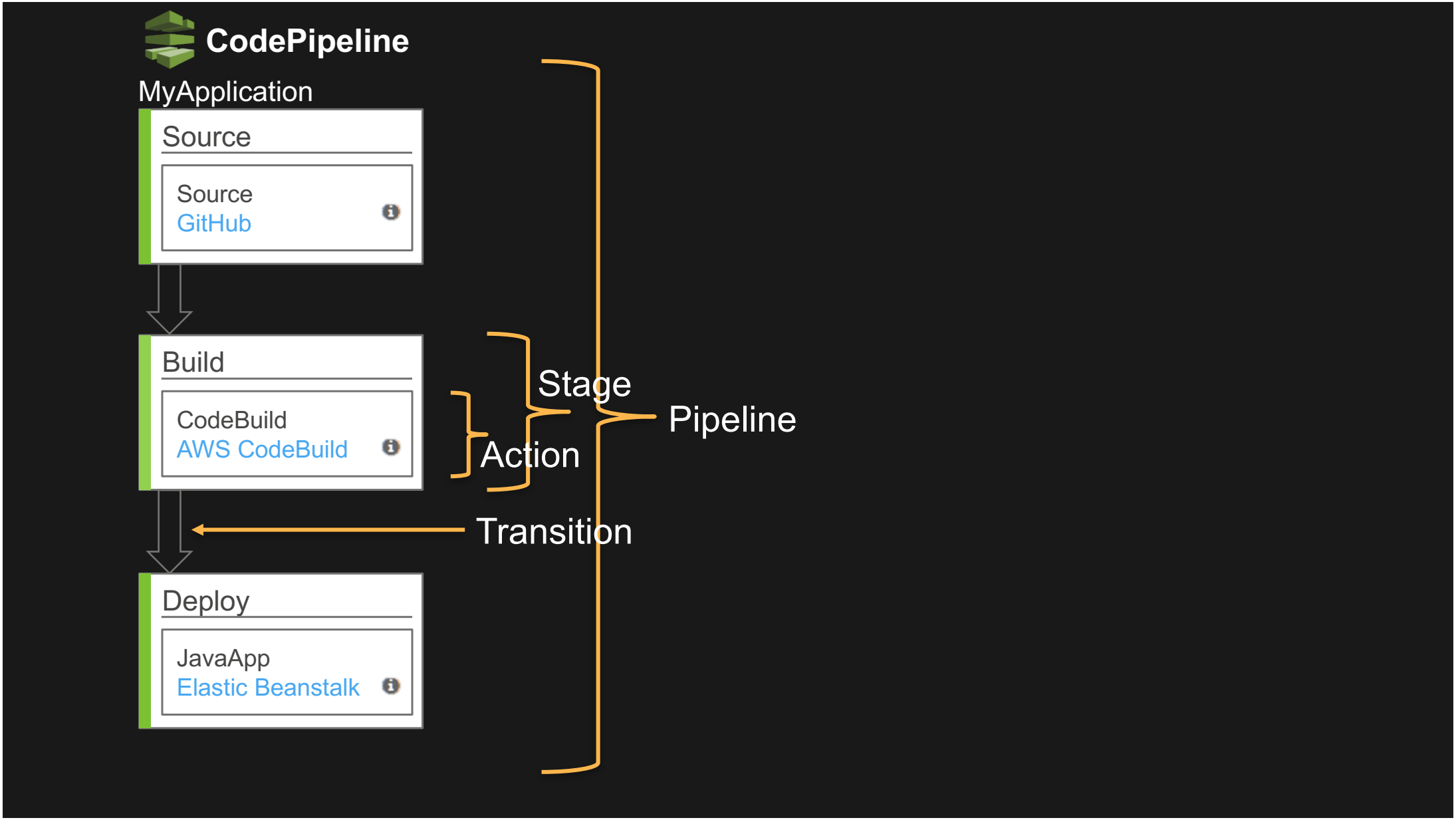


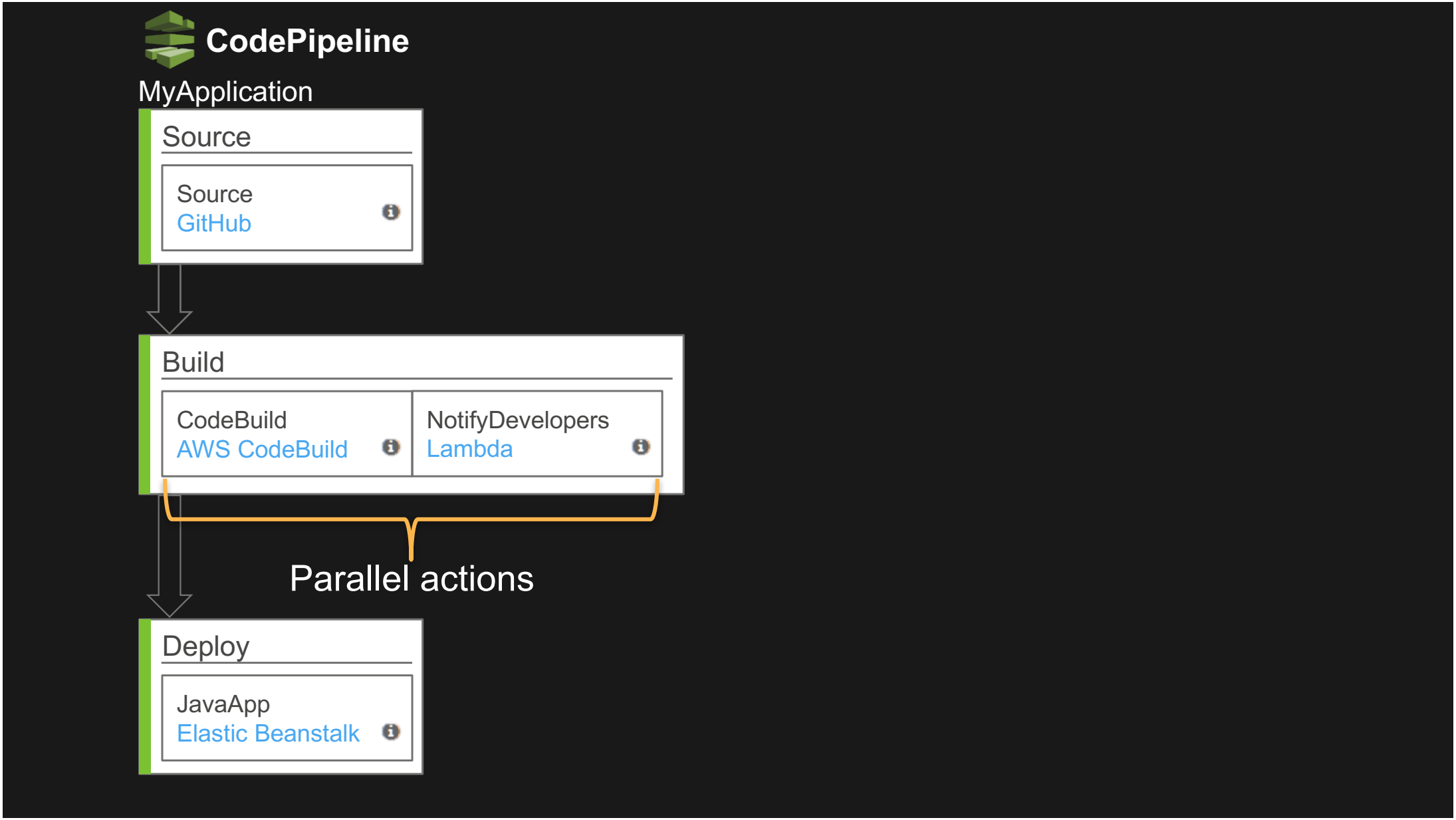
Continuous delivery service for fast and reliable application updates

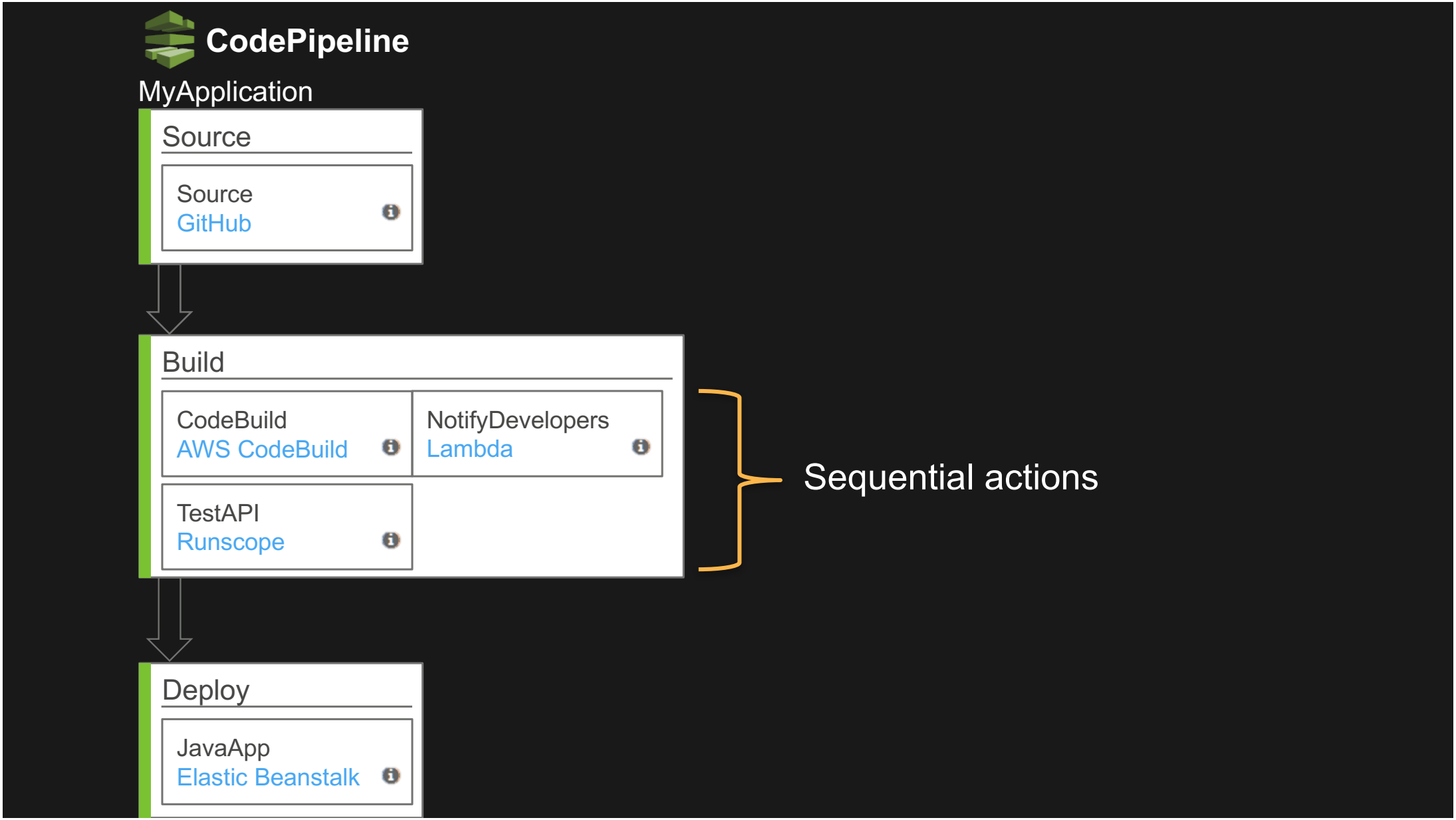
Model and visualize your software release process

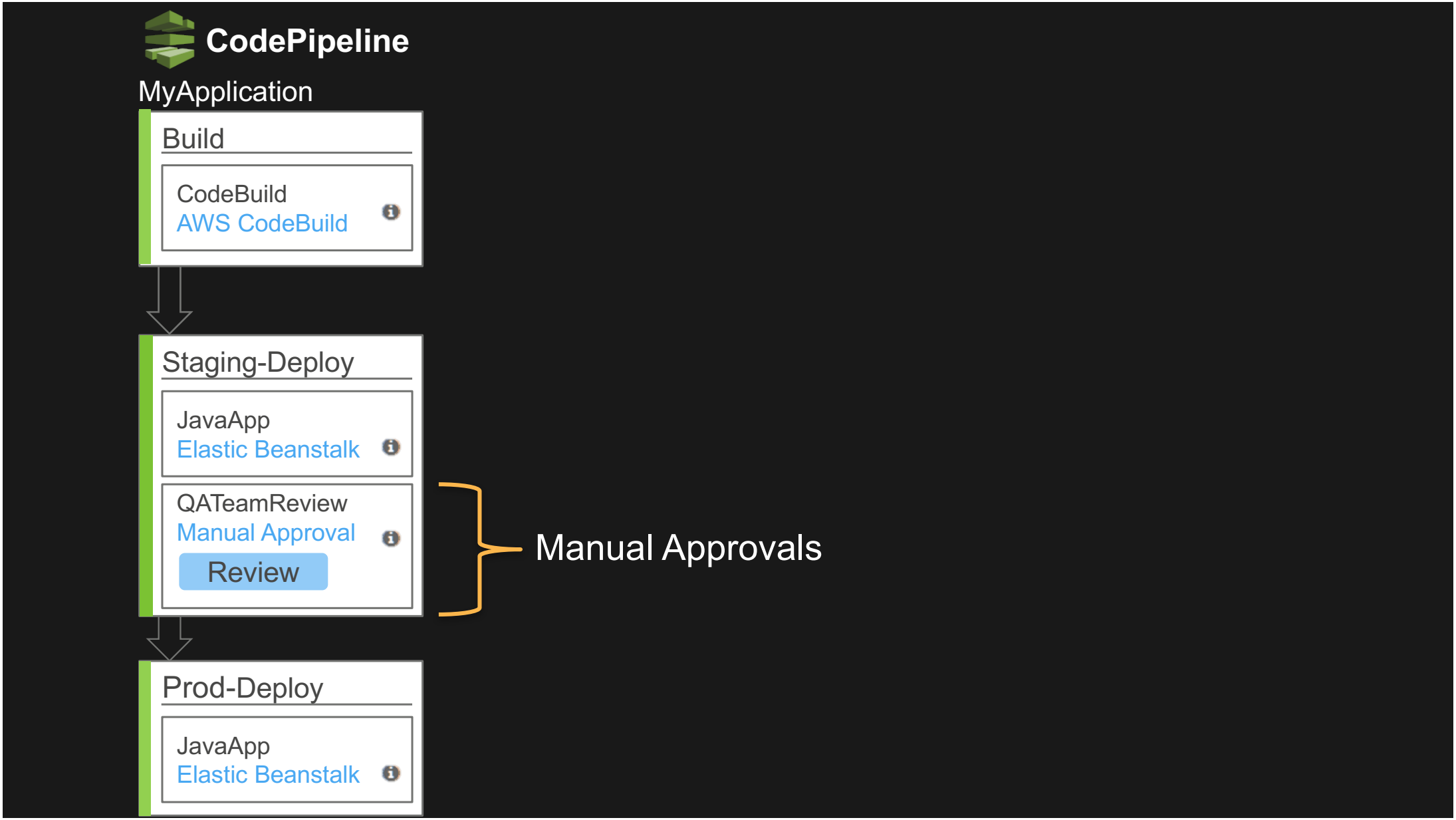
Builds, tests, and deploys your code every time there is a code change

Integrates with third-party tools and AWS


CodePipeline
MyApplication
Source
Source
GitHub
Build
CodeBuild
AWS CodeBuild
Deploy
JavaApp
Elastic Beanstalk
Stage
Action
Pipeline
Transition


CodePipeline
MyApplication
Source
Source
GitHub
Build
CodeBuild
AWS CodeBuild
NotifyDevelopers
Lambda
Parallel actions
Deploy
JavaApp
Elastic Beanstalk


CodePipeline
MyApplication
Source
Source
GitHub
Build
CodeBuild
AWS CodeBuild
NotifyDevelopers
Lambda
TestAPI
Runscope
Sequential actions
Deploy
JavaApp
Elastic Beanstalk


CodePipeline
MyApplication
Build
CodeBuild
AWS CodeBuild
Staging-Deploy
JavaApp
Elastic Beanstalk
QATeamReview
Manual Approval
Review
Manual Approvals
Prod-Deploy
JavaApp
Elastic Beanstalk


DEMO!

Demo:

1. Start with a repository (github.com/awslabs/aws-codedeploy-sample-tomcat)
2. Add buildspec.yml
3. Create CodePipeline pipeline with a Source and Build stage
4. Do a build
5. Add a deploy stage
6. Do a full execution of the pipeline

Demo:

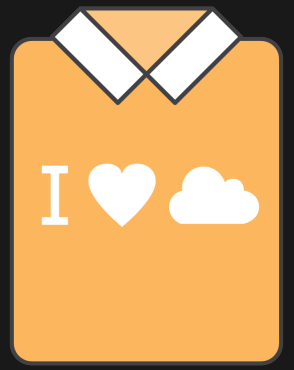
- ✓ Start with a repository (github.com/awslabs/aws-codedeploy-sample-tomcat)
- ✓ Add buildspec.yml
- ✓ Create CodePipeline pipeline with a Source and Build stage
- ✓ Do a build
- Add a deploy stage
- Do a full execution of the pipeline

Demo:

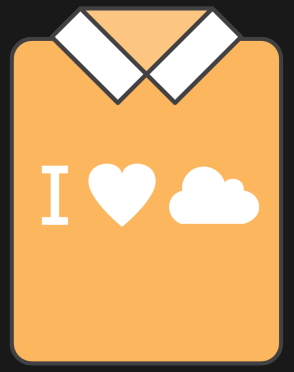
- ✓ Start with a repository (github.com/awslabs/aws-codedeploy-sample-tomcat)
- ✓ Add buildspec.yml
- ✓ Create CodePipeline pipeline with a Source and Build stage
- ✓ Do a build
- ✓ Add a deploy stage
- ✓ Do a full execution of the pipeline

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

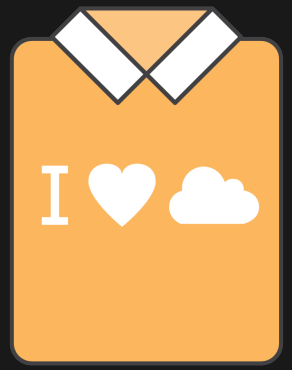


General Best Practices used by Amazon Developers



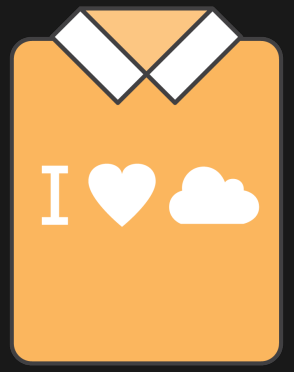
- CI/CD is a MUST!
  - Commit frequently
  - Builds on every commit
  - Build once in a given execution flow
  - Deploy to a running environment for further testing
- Everything that is code (application, infrastructure, documentation) goes into a repository
  - If its not in a repository, it doesn’t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

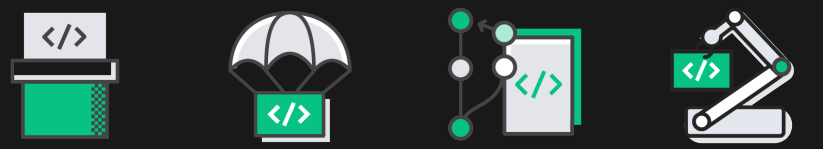
General Best Practices used by Amazon Developers



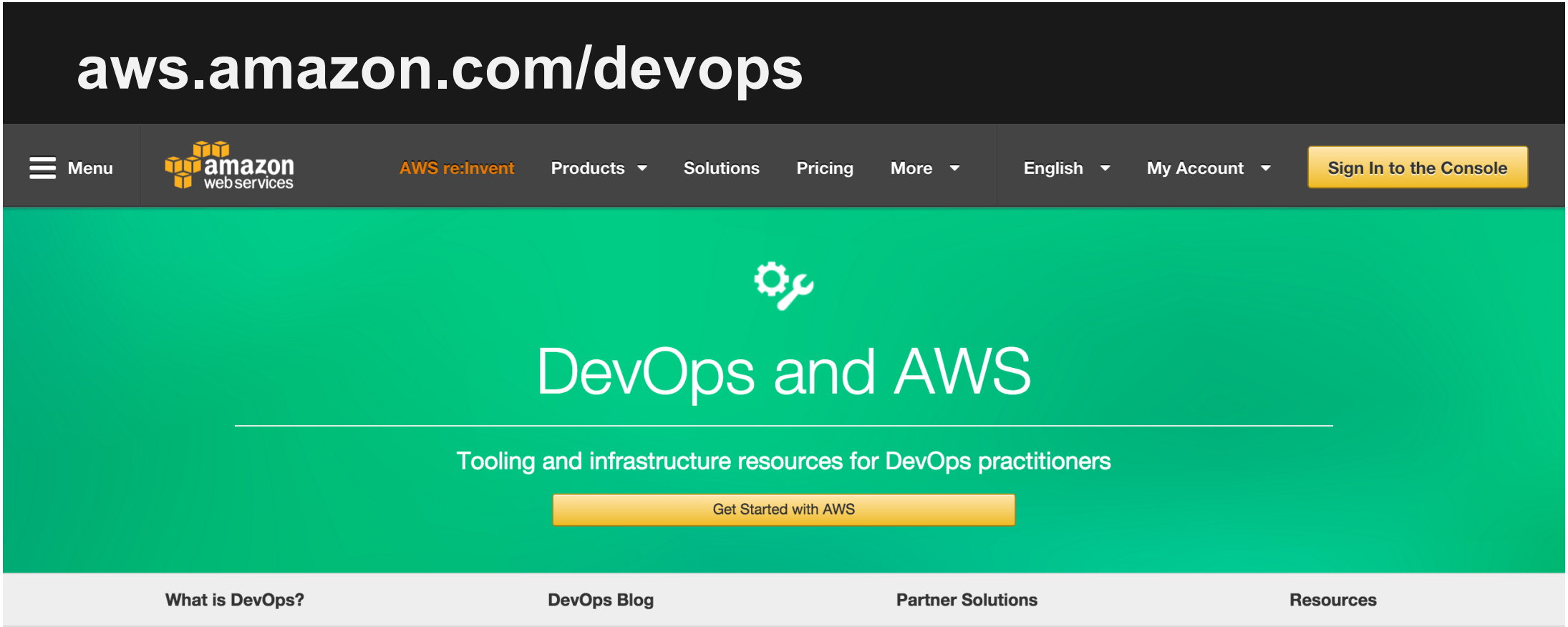
- CI/CD is a MUST!
  - Commit frequently
  - Builds on every commit
  - Build once in a given execution flow
  - Deploy to a running environment for further testing
- Everything that is code (application, infrastructure, documentation) goes into a repository
  - If its not in a repository, it doesn't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

Code* Tips and Tricks

- All Code* products can(and should) be provisioned and managed with AWS CloudFormation!
    - You could literally store the CloudFormation templates that provision your Code* resources in CodeCommit and update them via CodePipeline (It's like Code* Inception!)
- Deep integration with IAM. You can assign permissions on who can commit code, approve manual approvals, deploy to certain deployment groups and more!
- Integrate with AWS Lambda to do almost anything:
    - CodeCommit has Repository Triggers
    - CodeDeploy has Event Notifications
    - CodePipeline has native Lambda invoke



aws.amazon.com/devops



AWS provides a set of flexible services designed to enable companies to more rapidly and reliably build and deliver products using AWS and DevOps practices. These services simplify provisioning and managing infrastructure, deploying application code, automating software release processes, and monitoring your application and infrastructure performance.

DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management

AWS DevOps Blog

Menu | amazon web services | AWS re:Invent | Products | More | My Account | Sign In to the Console

AWS DevOps Blog

Building a Cross-Region/Cross-Account Code Deployment Solution on AWS

by BK Chaurasiya | on 01 NOV 2016 | in How-To | Permalink | Comments

Many of our customers have expressed a desire to build an end-to-end release automation workflow solution that can deploy changes across multiple regions or different AWS accounts.

In this post, I will show you how you can easily build an automated cross-region code deployment solution using AWS CodePipeline (a continuous delivery service), AWS CodeDeploy (an automated application deployment service), and AWS Lambda (a serverless compute service). In the Taking This Further section, I will also show you how to extend what you've learned so that you can create a cross-account deployment solution.

We will use AWS CodeDeploy and AWS CodePipeline to create a multi-pipeline solution running in two regions (Region A and Region B). Any update to the source code in Region A will trigger validation and deployment of source code changes in the pipeline in Region A. A successful processing of source code in all of its AWS CodePipeline stages will invoke a Lambda function as a custom action, which will copy the source code into an S3 bucket in Region B. After the source code is copied into this bucket, it will trigger a similar chain of processes into the different AWS CodePipeline stages in Region B. See the following diagram

Search the DevOps Blog

Search

Resources

AWS Developer Tools

AWS Management Tools

DevOps and AWS

Resources for DevOps on AWS


QUESTION
EVERYTHING
?
https://secure.flickr.com/photos/dullhunk/202872717/