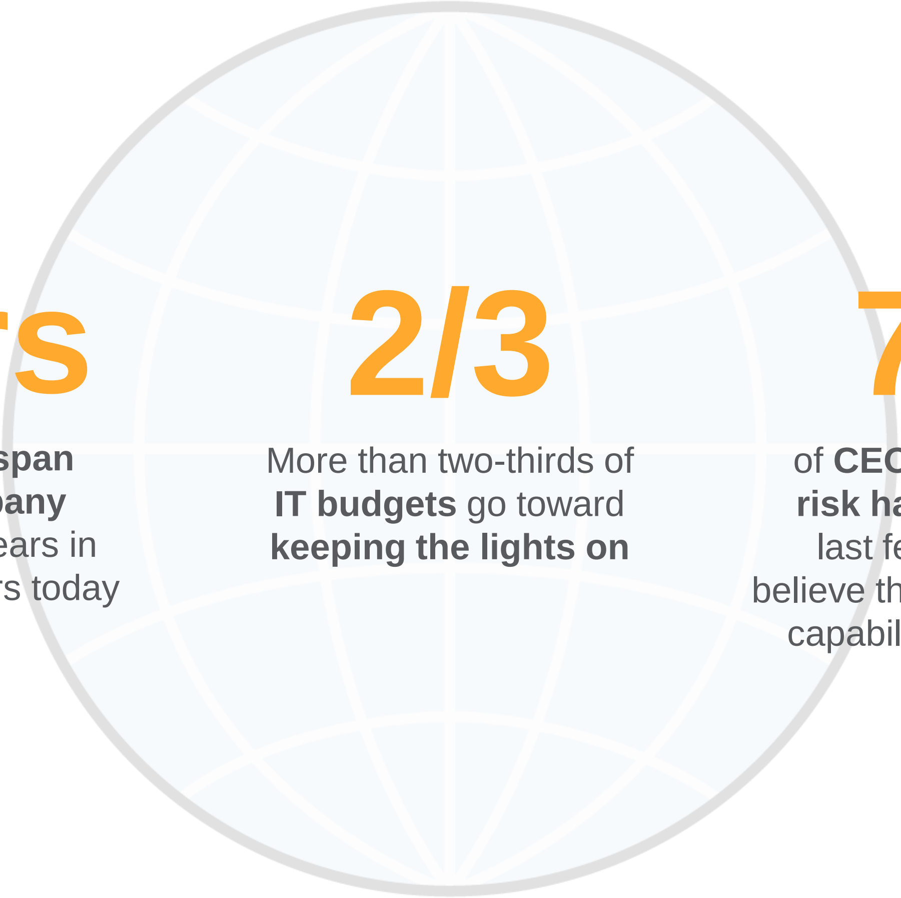
DevOps Tooling

March 2017

Paul Maddox, Specialist Solutions Architect
DevOps & Developer Technologies

@paulmaddox



15yrs

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

2/3

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

77%

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

How This Affects You

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

Your traditional IT model **lacks the agility** you need to keep pace with market disruptors

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

Responding requires a new model

Focus on differentiating your company

Innovate at speed

Reduce risk

“Finding time for innovation is hard,

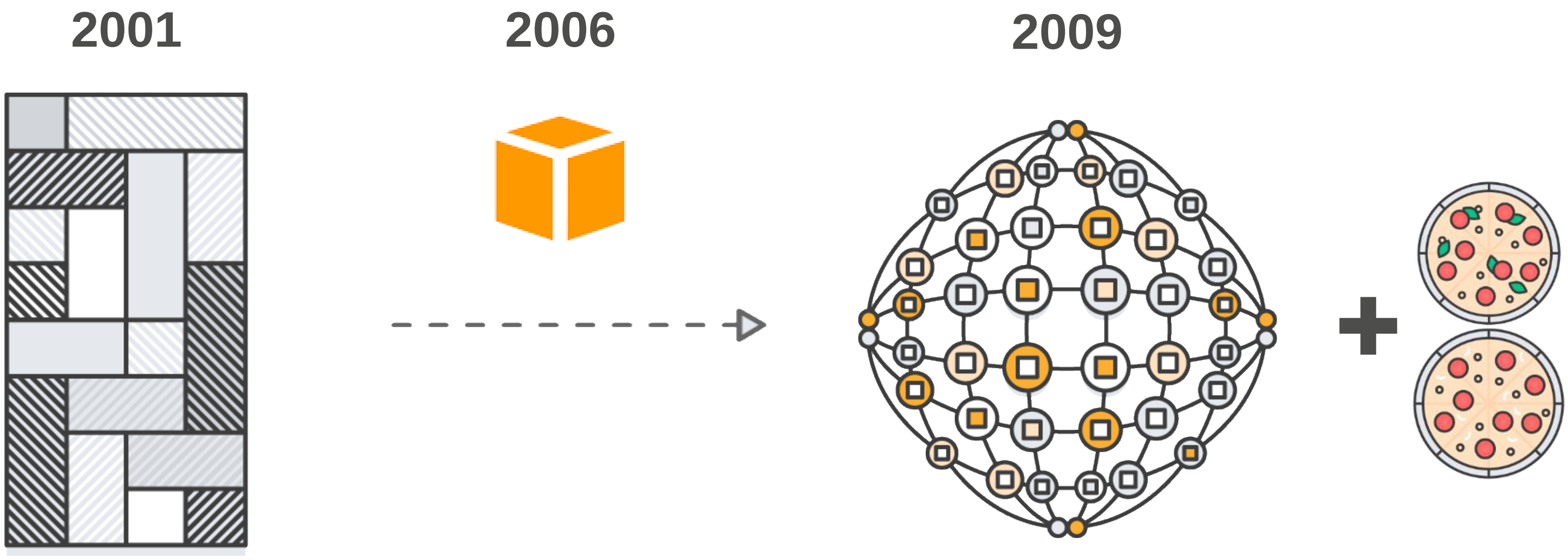
we’re just ***too busy****…”*

“But, we have ***too much legacy****…”*

Amazon.com used to be a monolith…

A 1GB executable that took 18hrs to compile, with a centralised deployment team

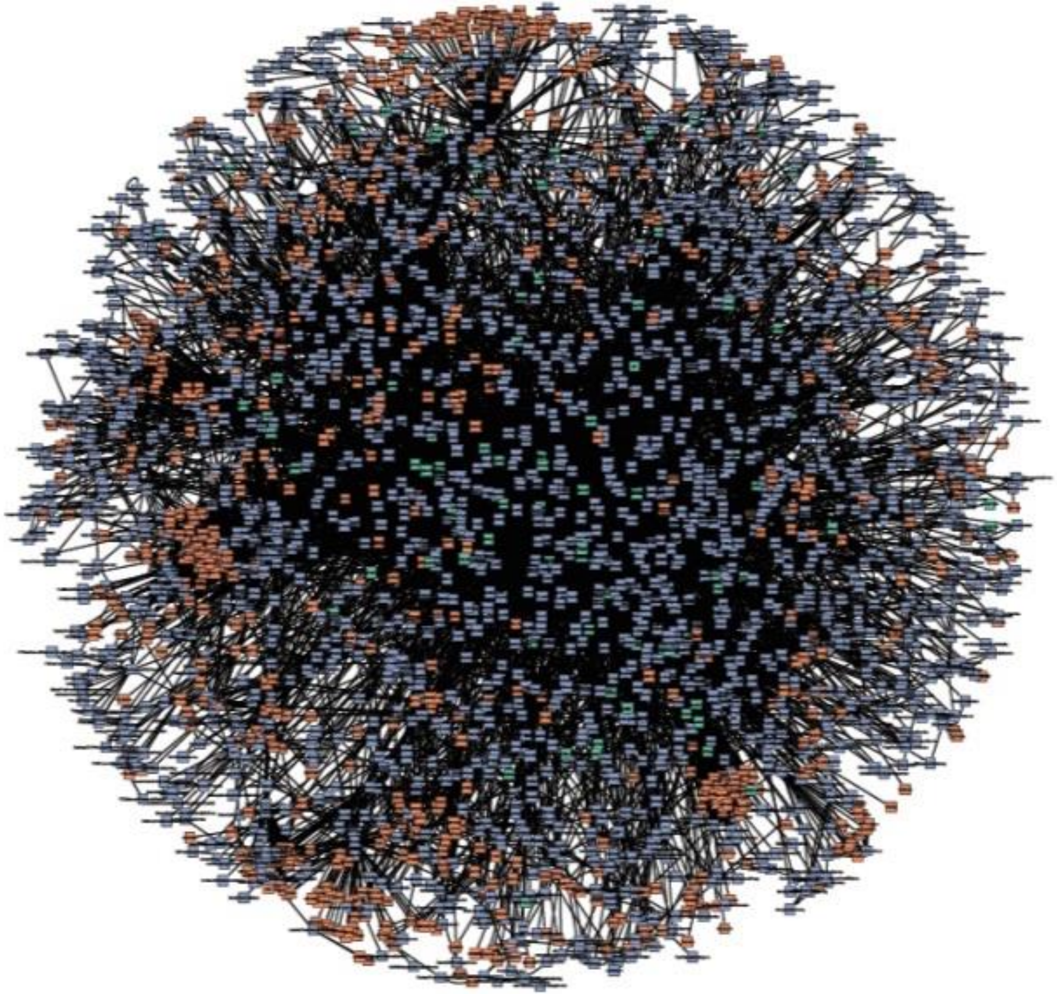
Development transformation at Amazon: 2001-2009



2001

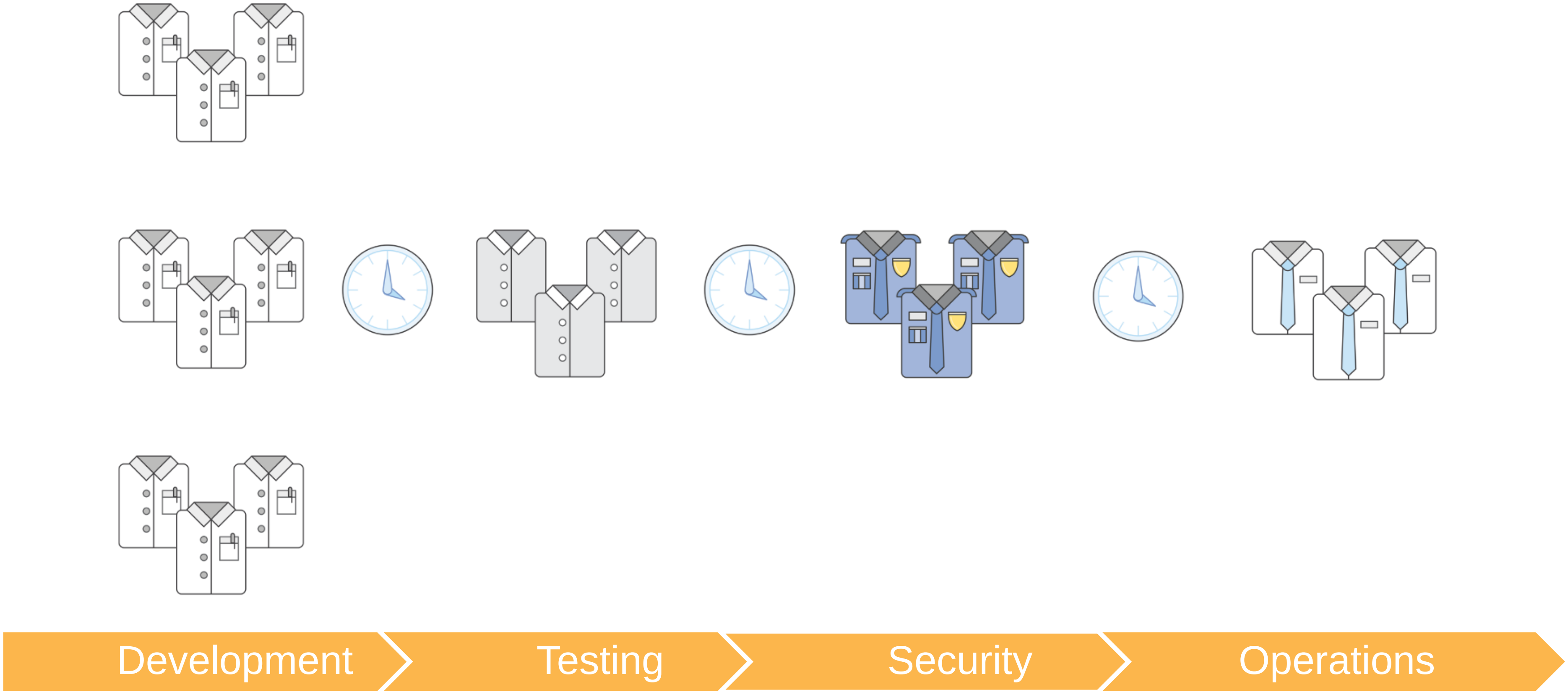
2006

2009



Amazon Retail Platform (2009)

Where is time consumed?

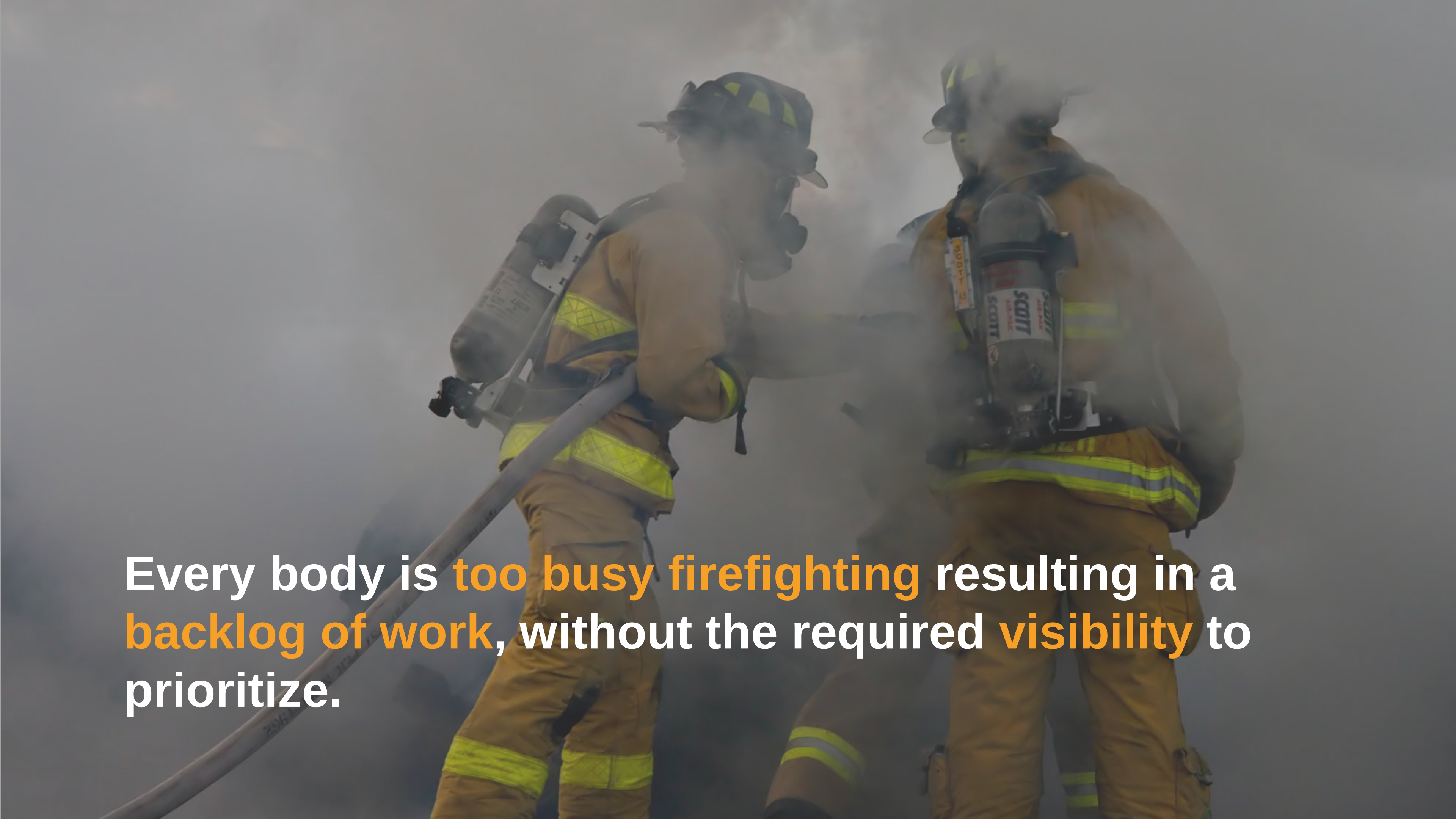


Development

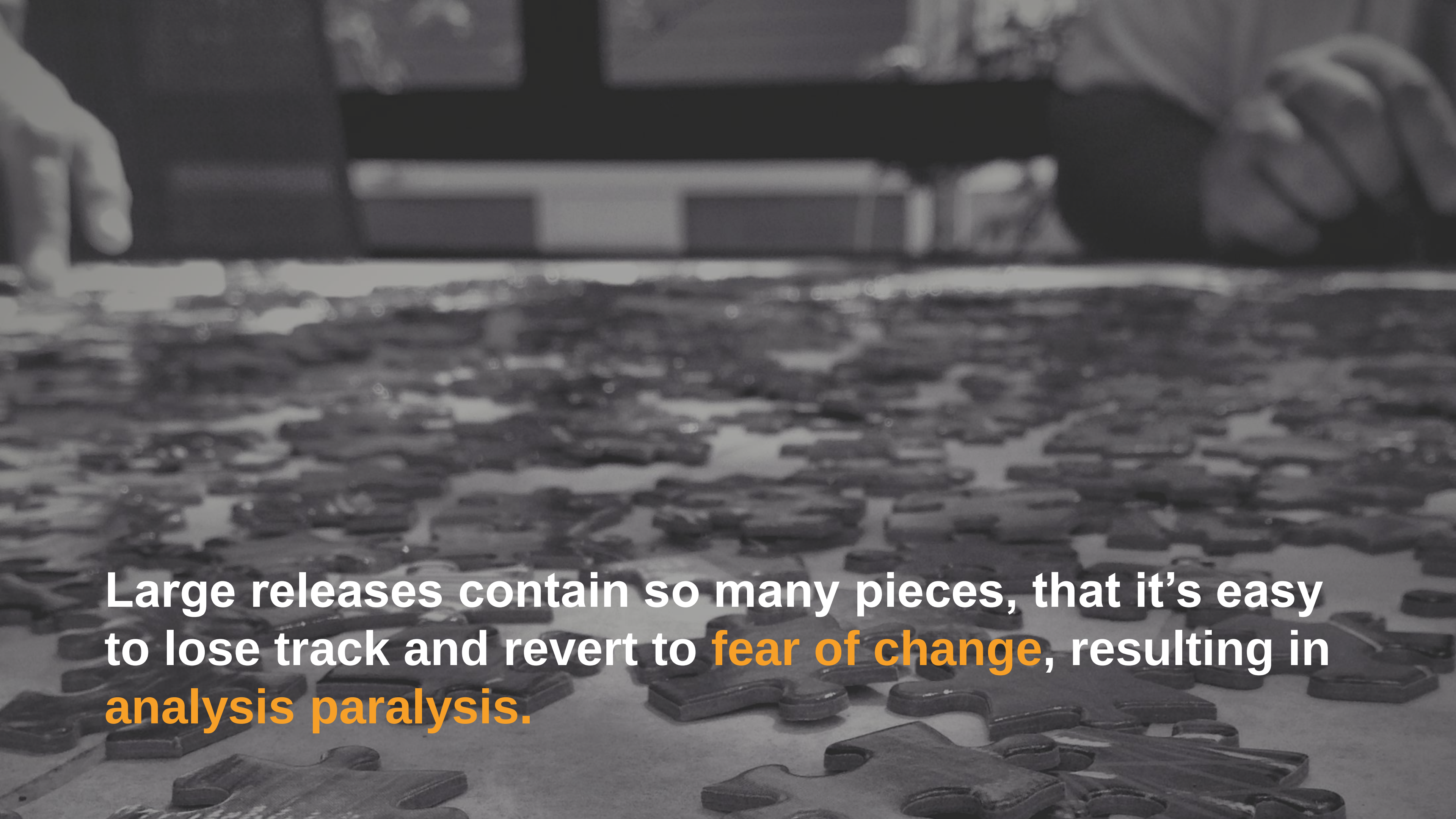
Testing

Security

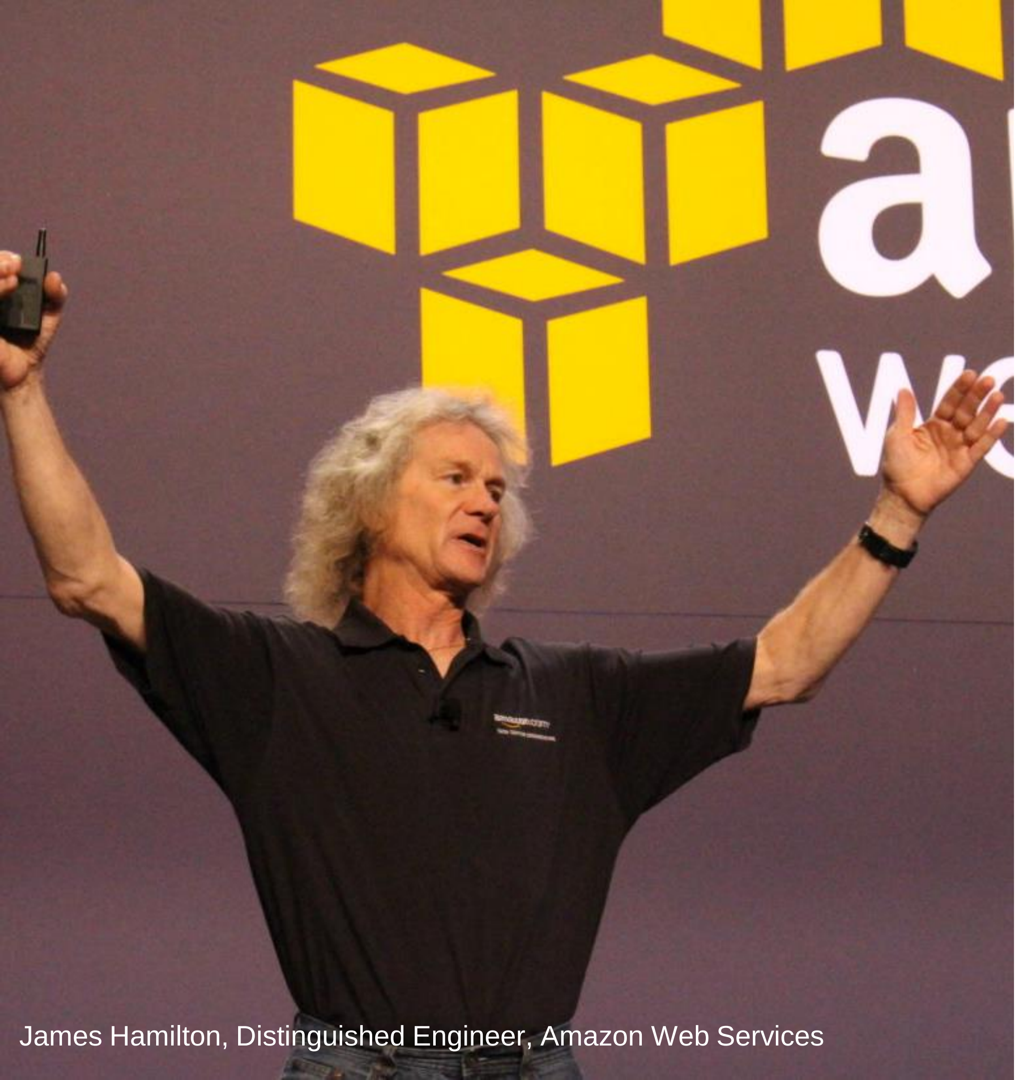
Operations



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



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



James Hamilton, Distinguished Engineer, Amazon Web Services

“We have long believed that 80% of operations issues originate in design and development…

Most operations issues, however, either have their genesis in design and development or are best solved there.

If the development team is frequently called in the middle of the night, automation is the likely outcome. If operations is frequently called, the usual reaction is to grow the operations team.”

Step 1: Shrink your deployments

A measure of innovation agility

How many deployments am I performing?

“We have a quarterly release cycle”

“Too many to count””

How many are done out of hours?

“We minimize customer impact”

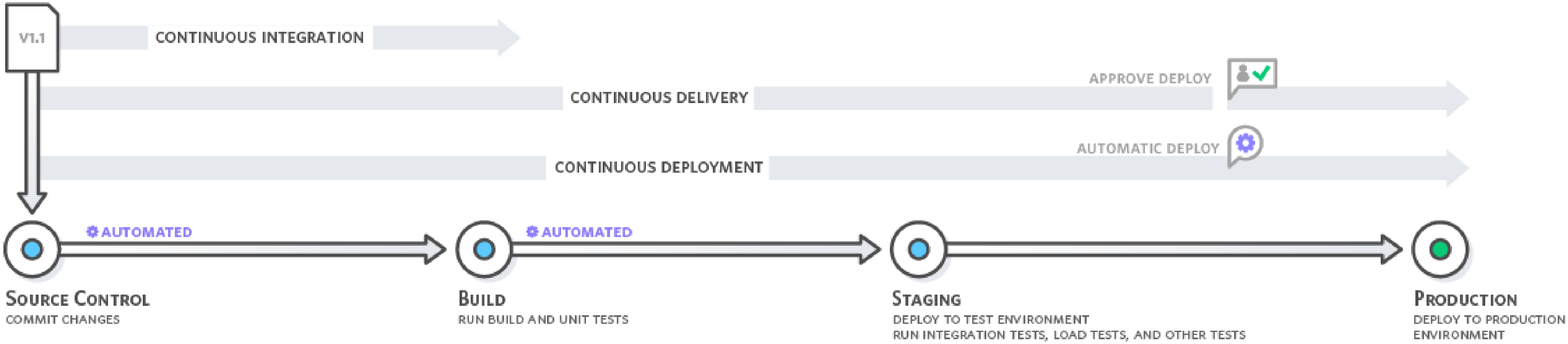
“Time is irrelevant”

How many suffer emergency roll backs?

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

“We roll forwards not back”

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



Continuous Delivery Benefits

Automate the software release process

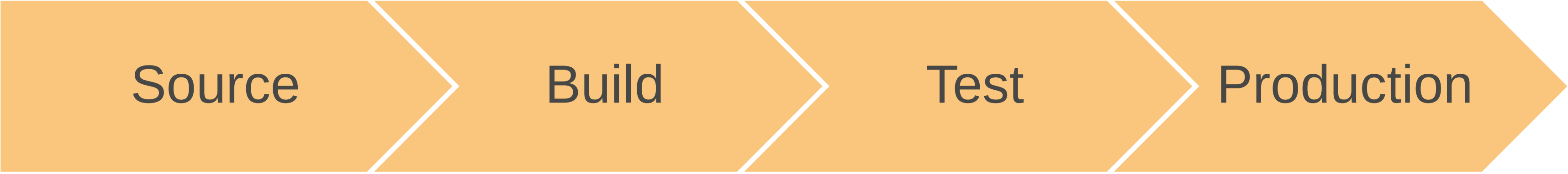
Improve developer productivity

Find and address bugs quickly

Deliver updates faster

Step 2: Improve Visibility

Release processes have four major phases



Source

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

Build

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

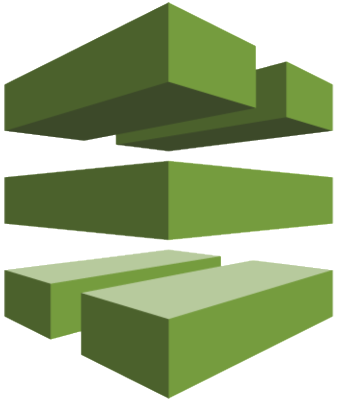
Test

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

Production

- Deployment to production environments

AWS CodePipeline

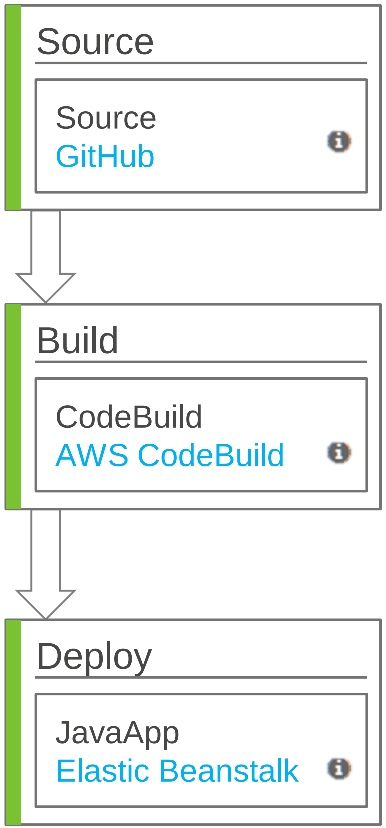


Continuous delivery service for fast and reliable application updates

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



Source

Source
GitHub

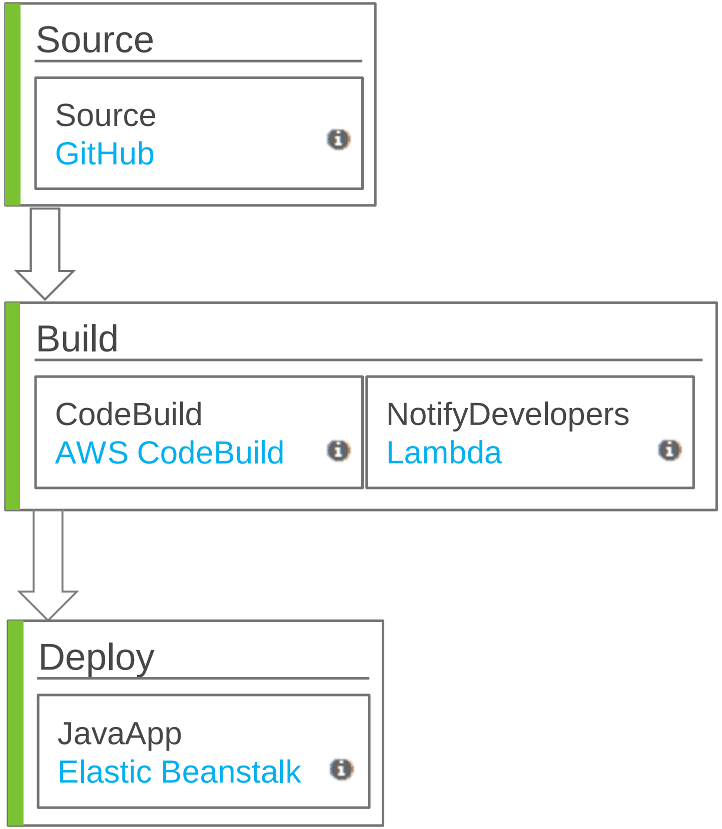
Build

CodeBuild
AWS CodeBuild

Deploy

JavaApp
Elastic Beanstalk

AWS CodePipeline automatically picks up new source revisions from **AWS CodeCommit, GitHub or S3** and takes them through your build and release process.



Source

Source
GitHub

Build

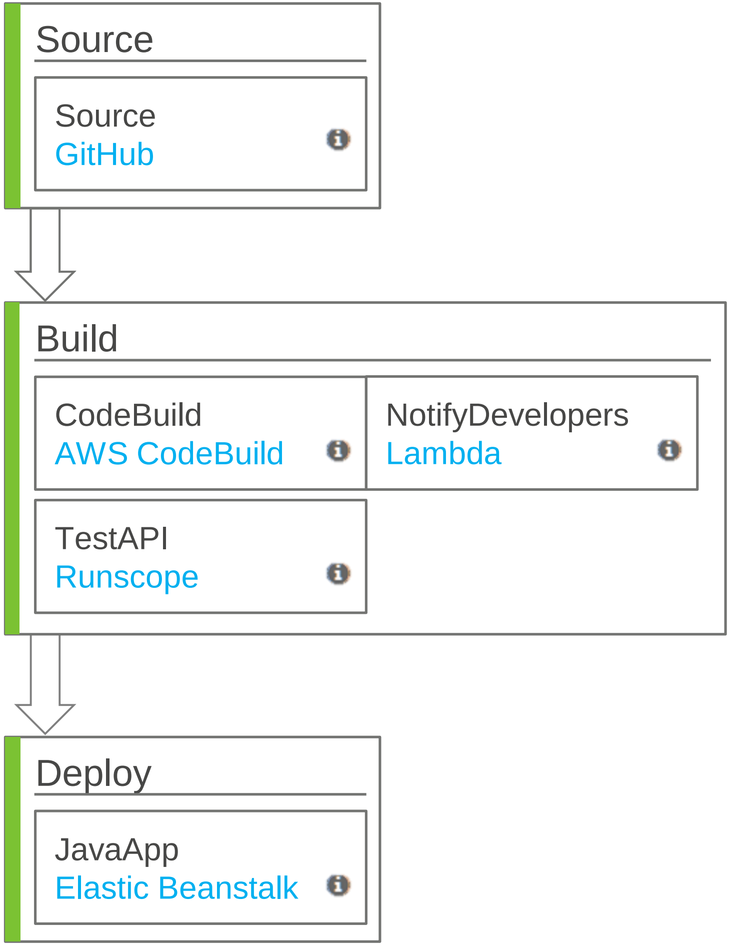
CodeBuild
AWS CodeBuild

NotifyDevelopers
Lambda

Deploy

JavaApp
Elastic Beanstalk

Parallel Actions



Source

Source
GitHub

Build

CodeBuild
AWS CodeBuild

NotifyDevelopers
Lambda

TestAPI
Runscope

Deploy

JavaApp
Elastic Beanstalk

Sequential Actions

Step 3: Automate all of the things!

- Build & test
- Infrastructure & deployment

https://www.flickr.com/photos/adamgiles/119754762

“This is our build server…

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

Don’t break it.”

AWS CodeBuild

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

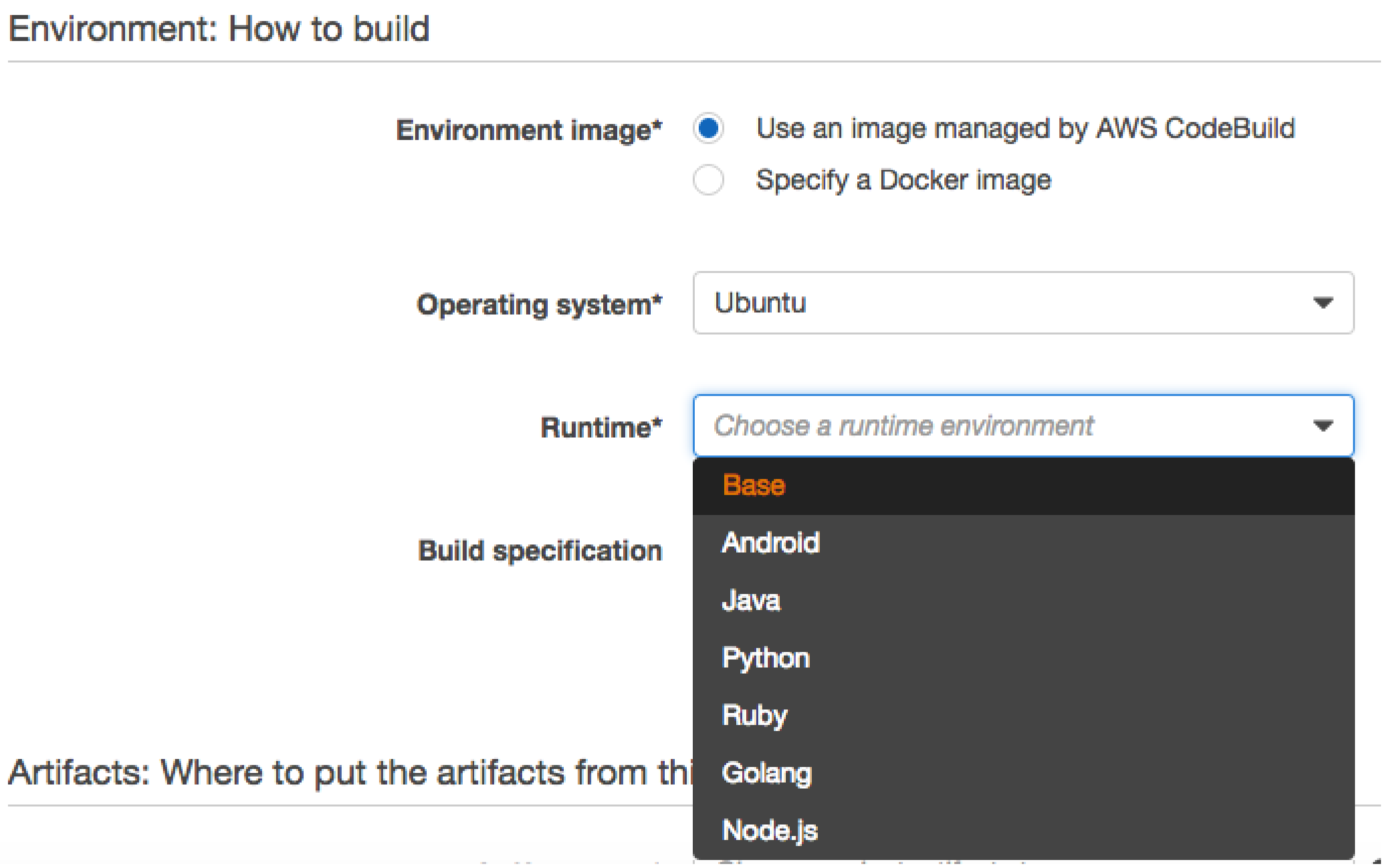
Scales continuously and processes multiple builds concurrently

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

Pay by the minute for the compute resources you use

Integrated with CodePipeline and Jenkins

Configure a Build Project



Environment: How to build

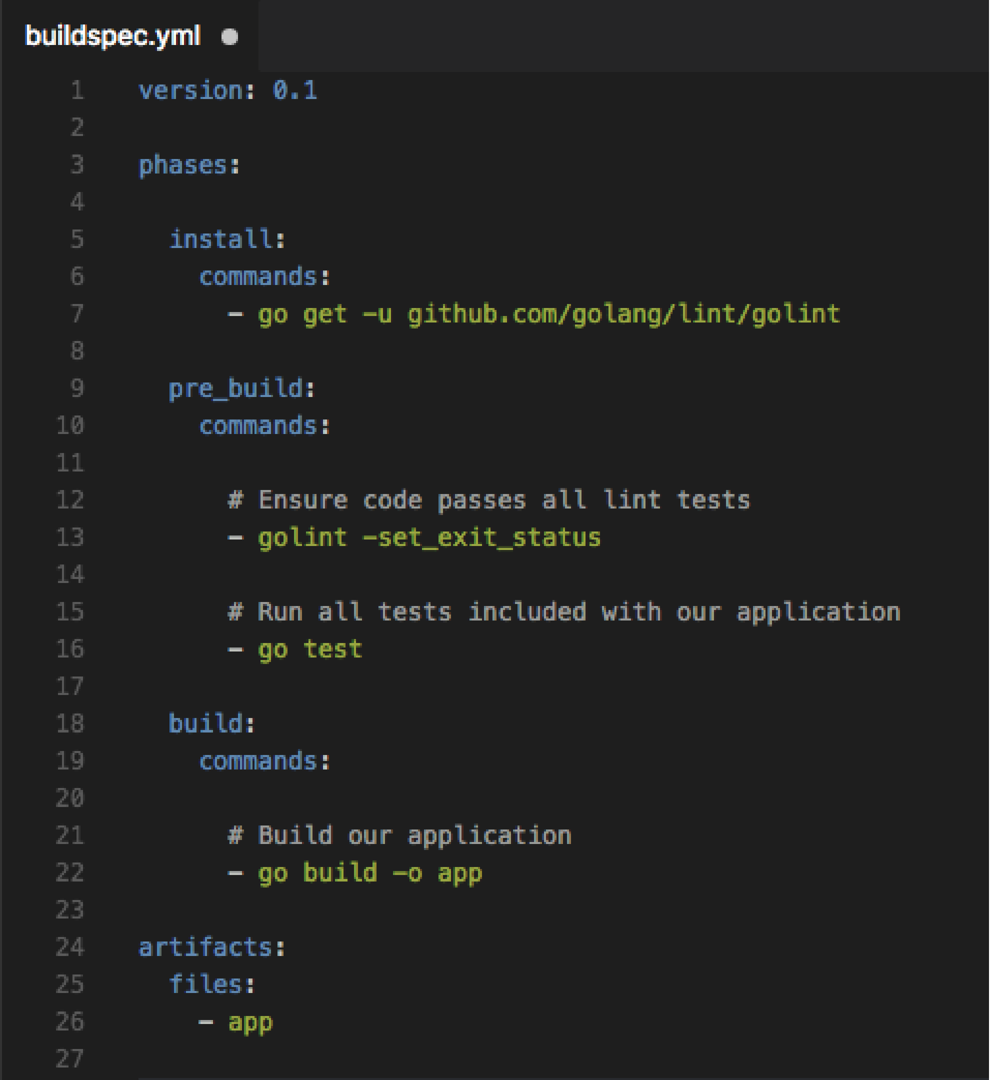
Environment image* ● Use an image managed by AWS CodeBuild
○ Specify a Docker image

Operating system* Ubuntu

Runtime* *Choose a runtime environment*

Build specification

Artifacts: Where to put the artifacts from thi

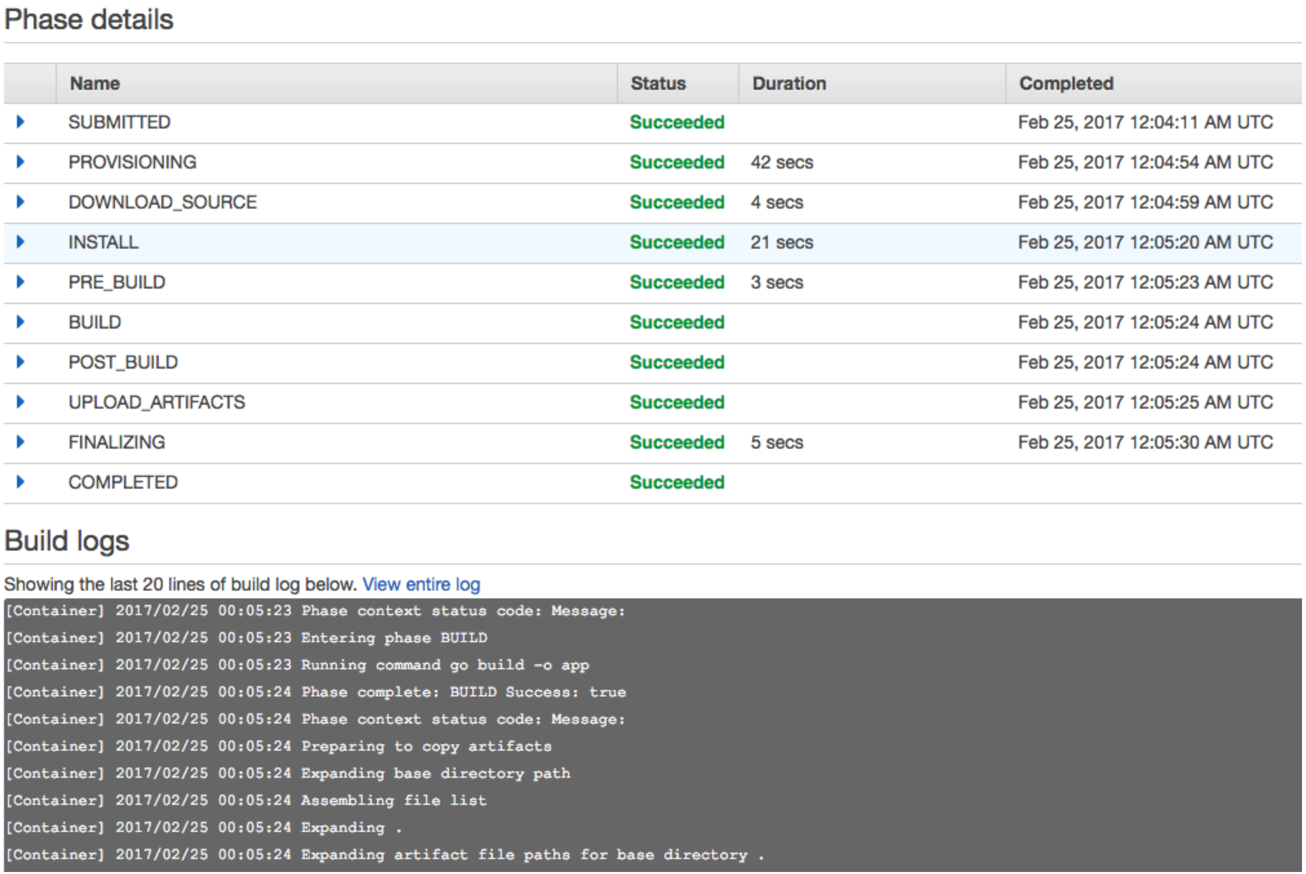


```yaml
version: 0.1

phases:

  install:
    commands:
      - go get -u github.com/golang/lint/golint

  pre_build:
    commands:

      # Ensure code passes all lint tests
      - golint -set_exit_status

      # Run all tests included with our application
      - go test

  build:
    commands:

      # Build our application
      - go build -o app

artifacts:
  files:
    - app

```

buildspec.yml

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

See build results



Phase details

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

Build logs

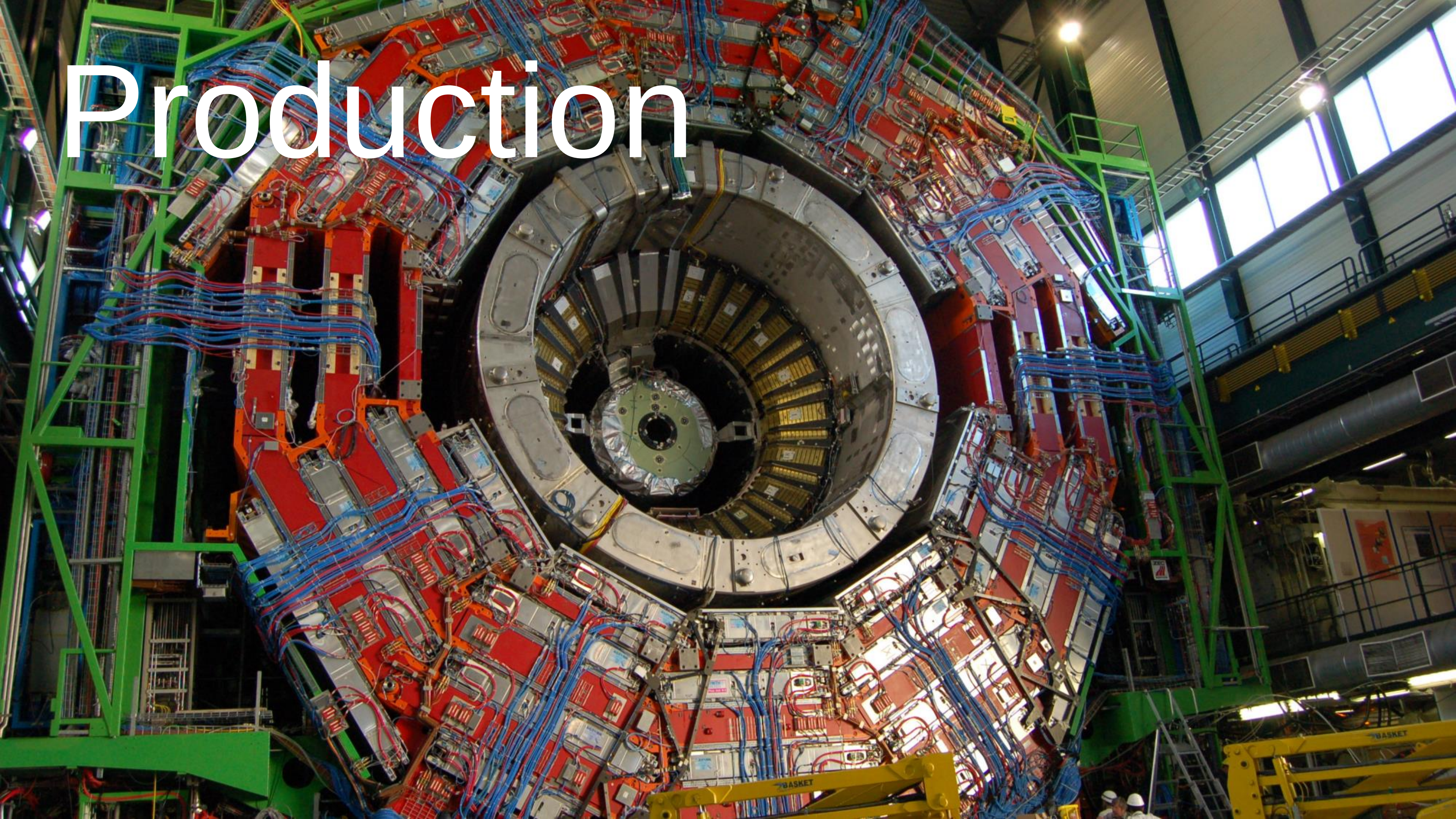
Showing the last 20 lines of build log below. View entire log

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

Step 3: Automate all of the things!

- Build & test
- Infrastructure & deployment

Production

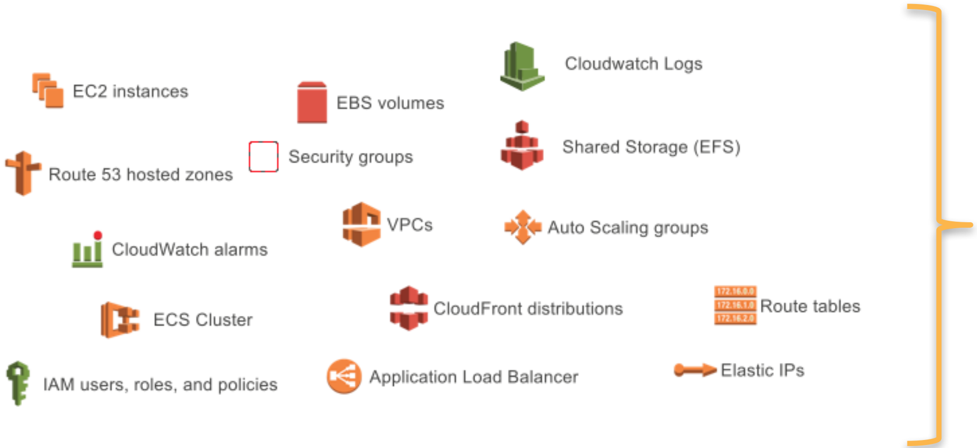


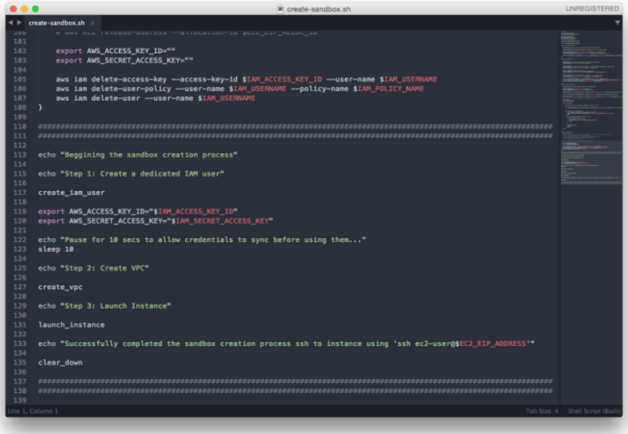
Staging / Test

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

Consistent environments build trust

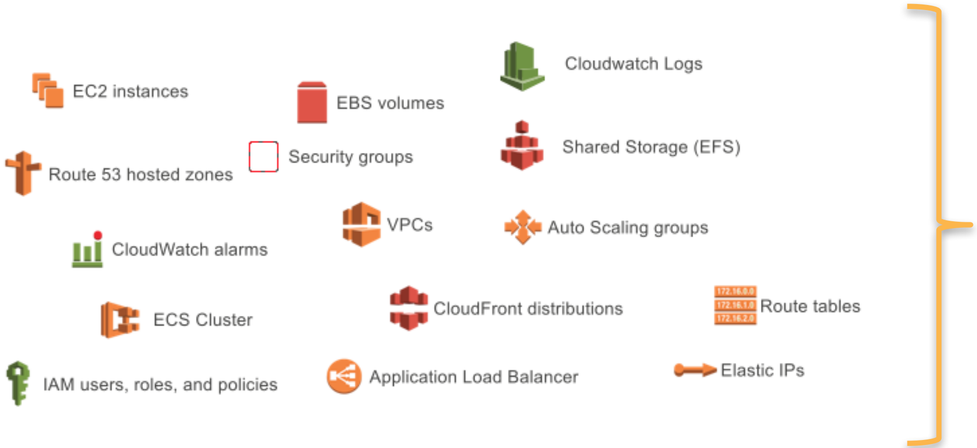
Write a script?

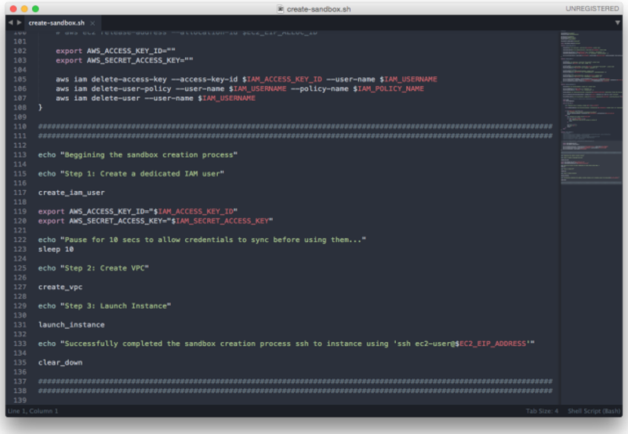




```
    # aws ec2 release-address --allocation-id $EC2_EIP_ALLOC_ID

    export AWS_ACCESS_KEY_ID=""
    export AWS_SECRET_ACCESS_KEY=""

    aws iam delete-access-key --access-key-id $IAM_ACCESS_KEY_ID --user-name $IAM_USERNAME
    aws iam delete-user-policy --user-name $IAM_USERNAME --policy-name $IAM_POLICY_NAME
    aws iam delete-user --user-name $IAM_USERNAME
}

##########################################################################################################################
##########################################################################################################################

echo "Beggining the sandbox creation process"

echo "Step 1: Create a dedicated IAM user"

create_iam_user

export AWS_ACCESS_KEY_ID="$IAM_ACCESS_KEY_ID"
export AWS_SECRET_ACCESS_KEY="$IAM_SECRET_ACCESS_KEY"

echo "Pause for 10 secs to allow credentials to sync before using them..."
sleep 10

echo "Step 2: Create VPC"

create_vpc

echo "Step 3: Launch Instance"

launch_instance

echo "Successfully completed the sandbox creation process ssh to instance using 'ssh ec2-user@$EC2_EIP_ADDRESS'"

clear_down

##########################################################################################################################
##########################################################################################################################

```

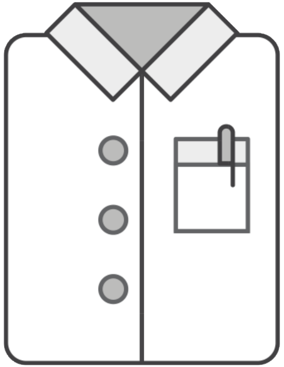
Write a script?





```
    # aws ec2 release-address --allocation-id $EC2_EIP_ALLOC_ID

    export AWS_ACCESS_KEY_ID=""
    export AWS_SECRET_ACCESS_KEY=""

    aws iam delete-access-key --access-key-id $IAM_ACCESS_KEY_ID --user-name $IAM_USERNAME
    aws iam delete-user-policy --user-name $IAM_USERNAME --policy-name $IAM_POLICY_NAME
    aws iam delete-user --user-name $IAM_USERNAME
}

####################################################################################################
####################################################################################################

echo "Beggining the sandbox creation process"

echo "Step 1: Create a dedicated IAM user"

create_iam_user

export AWS_ACCESS_KEY_ID="$IAM_ACCESS_KEY_ID"
export AWS_SECRET_ACCESS_KEY="$IAM_SECRET_ACCESS_KEY"

echo "Pause for 10 secs to allow credentials to sync before using them..."
sleep 10

echo "Step 2: Create VPC"

create_vpc

echo "Step 3: Launch Instance"

launch_instance

echo "Successfully completed the sandbox creation process ssh to instance using 'ssh ec2-user@$EC2_EIP_ADDRESS'"

clear_down

####################################################################################################
####################################################################################################

```

Lets imagine for a minute

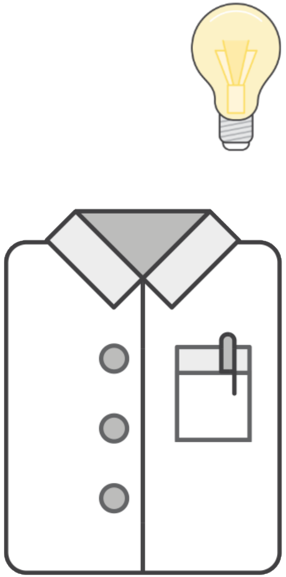


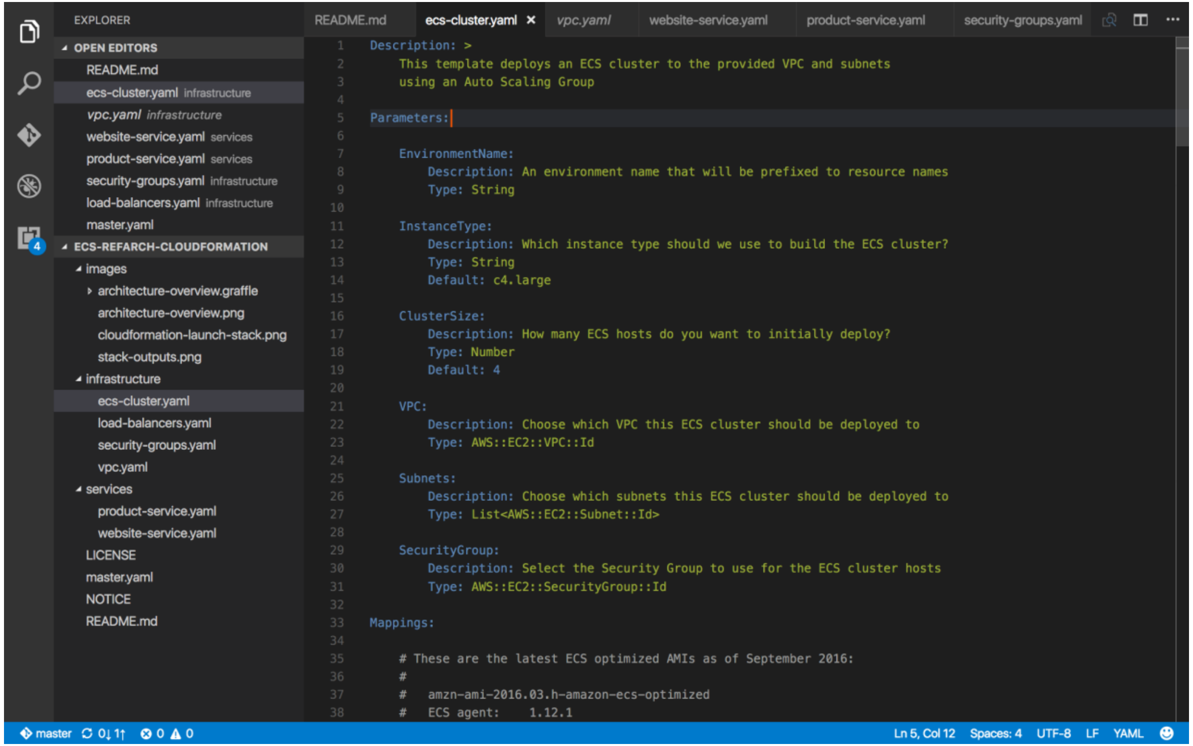
This is Alice, she needs to build a new environment.

It will:

- Contain infrastructure & applications to deploy.
- Need to be repeatable; new test & QA stacks are required all the time.
- Need to be auditable; her security teams are often left out of the loop.

Alice knows about CloudFormation…





```
Description: >
    This template deploys an ECS cluster to the provided VPC and subnets
    using an Auto Scaling Group

Parameters:

    EnvironmentName:
        Description: An environment name that will be prefixed to resource names
        Type: String

    InstanceType:
        Description: Which instance type should we use to build the ECS cluster?
        Type: String
        Default: c4.large

    ClusterSize:
        Description: How many ECS hosts do you want to initially deploy?
        Type: Number
        Default: 4

    VPC:
        Description: Choose which VPC this ECS cluster should be deployed to
        Type: AWS::EC2::VPC::Id

    Subnets:
        Description: Choose which subnets this ECS cluster should be deployed to
        Type: List<AWS::EC2::Subnet::Id>

    SecurityGroup:
        Description: Select the Security Group to use for the ECS cluster hosts
        Type: AWS::EC2::SecurityGroup::Id

Mappings:

    # These are the latest ECS optimized AMIs as of September 2016:
    #
    #   amzn-ami-2016.03.h-amazon-ecs-optimized
    #   ECS agent:    1.12.1
```

AWS CloudFormation

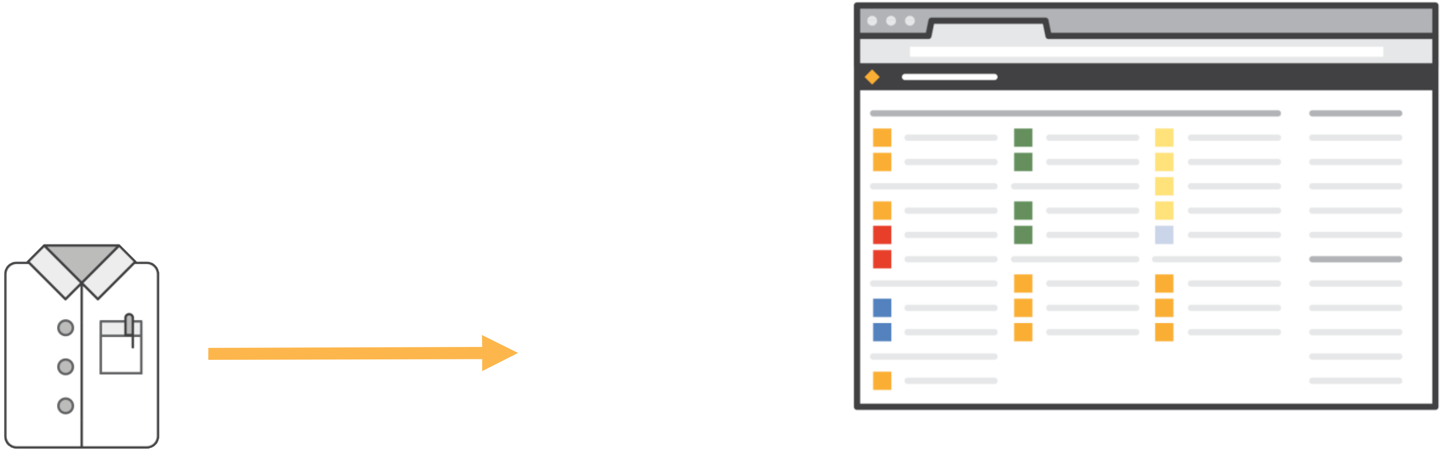
Infrastructure-as-Code

Time to deploy!

…or…

```
alice@macbook:~$ aws cloudformation create-stack
    --stack-name preprod
    --template-body file://Users/alice/env.yaml
```

A new environment is required…



…or…

```
alice@macbook:~$ aws cloudformation create-stack
      --stack-name development
      --template-body file://Users/alice/env.yaml
```

Template Anatomy

Description:

This is an example template.
It doesn't do much yet...

Parameters:

Resources:

Outputs:

Parameters

```
Description:

    This is an example template.
    All it has so far are input parameters...

Parameters:

    Type:
        Description: Is this a dev, test or prod environment?
        Type: String
        AllowedPattern: [dev|test|prod]

    VPC:
        Description: Pick a VPC to deploy to
        Type: AWS::EC2::VPC::Id
```

Resources

```
Description:

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

Resources:

    MyApplicationServer
        Type: AWS::EC2::Instance
        Properties:
            InstanceType: t2.micro
            ImageId: ami-6bb2d67c
            SecurityGroups:
                - !Ref MySecurityGroup

    MySecurityGroup
        Type: AWS::EC2::SecurityGroup
        Properties:
            ...
```

Outputs

```
Description:

    This is an example template.
    Now we’ve got some resources deployed and the public IP
    of our instance as an output of the template.

Outputs:

    MyPublicIpAddress
        Description: instance
        Value: !Sub https://${MyApplicationServer.PublicIp}
```

Helpful links

http://aws.amazon.com/cloudformation/aws-cloudformation-templates/

- Google: “AWS CloudFormation Templates”
- Helpful solution-based templates (eg: Active Directory domain forest, or LAMP stack

http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/CHAP_TemplateQuickRef.html

- Google: “AWS CloudFormation Snippets”
- Handy snippets for common AWS services (eg: Autoscaling group)

But what about deploying applications?

CloudFormation includes bootstrapping

```
AWS::CloudFormation::Init:
    config:

        users:
            # add local users
        groups:
            # add local groups
        packages:
            # install packages from repos (both Windows & Linux)
        sources:
            # fetch sources from source control
        files:
            # deploy files from S3 and set permissions
        commands:
            # write bash / powershell / python etc scripts
        services:
            # enable / disable OS services
```

AWS CodeDeploy

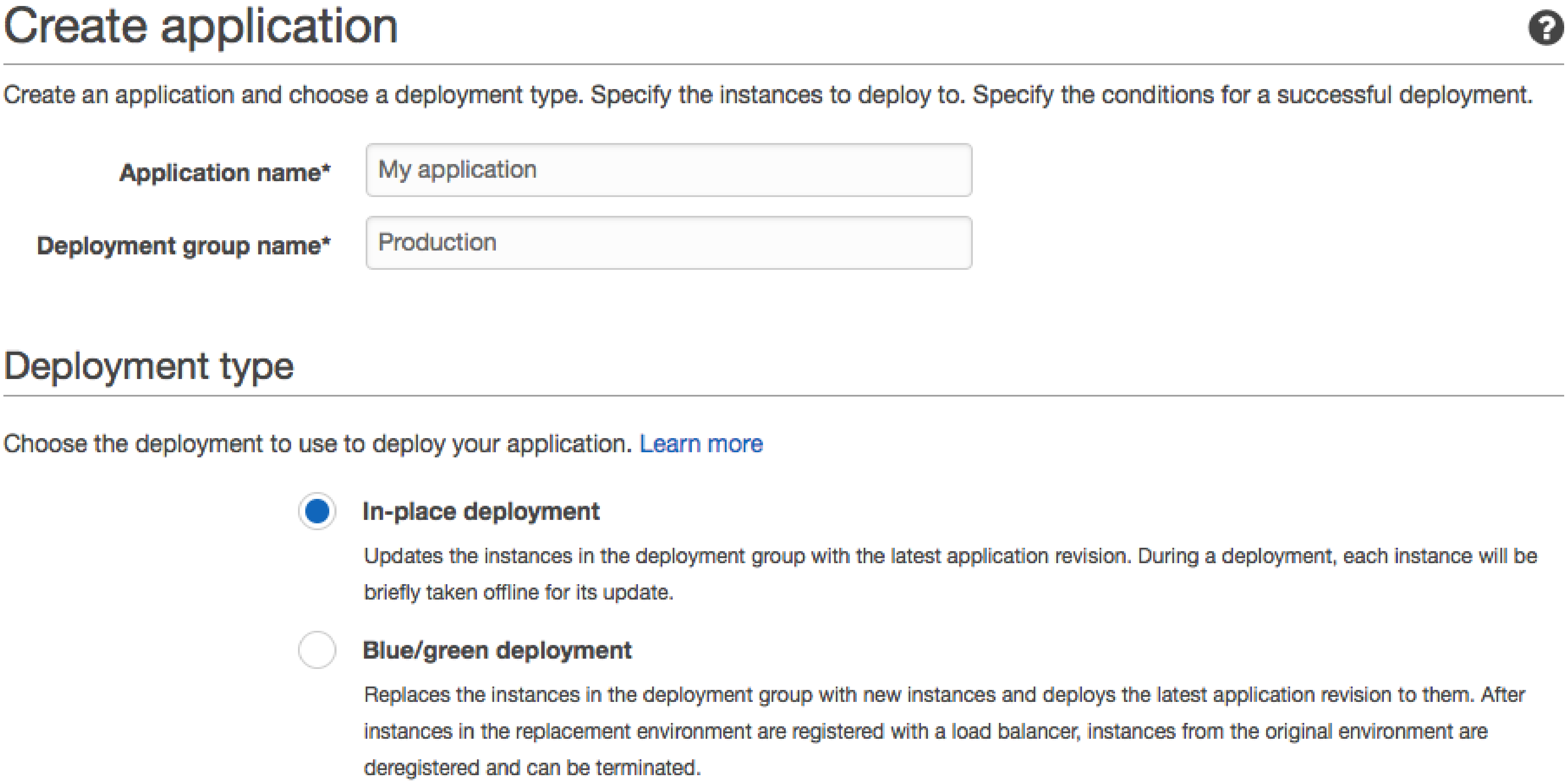
Automated deployments
Deploy to Amazon EC2 and/or On-premise

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

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

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

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



Create application

Create an application and choose a deployment type. Specify the instances to deploy to. Specify the conditions for a successful deployment.

Application name*	My application
Deployment group name*	Production

Deployment type

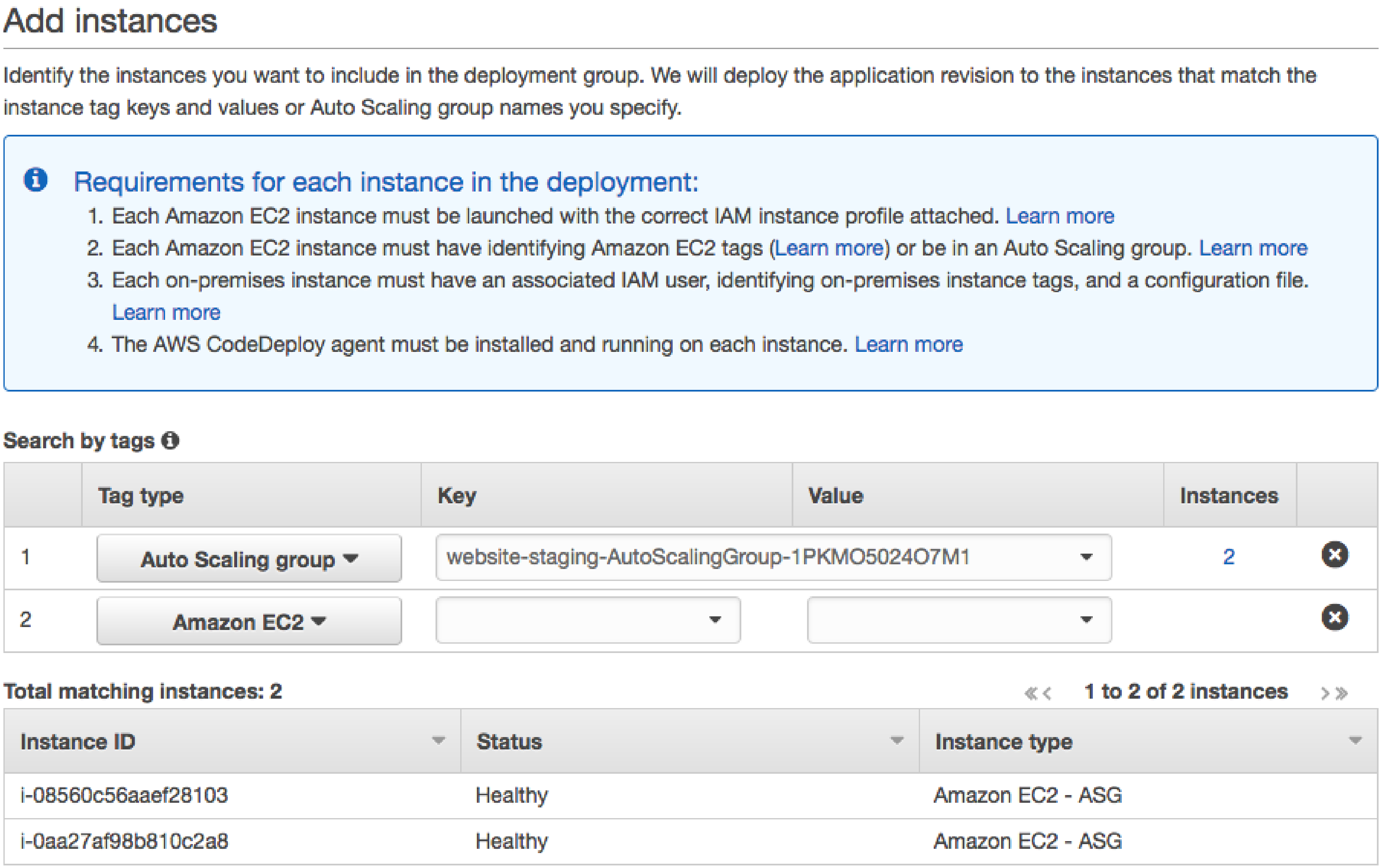
Choose the deployment to use to deploy your application. Learn more

● **In-place deployment**

Updates the instances in the deployment group with the latest application revision. During a deployment, each instance will be briefly taken offline for its update.

○ **Blue/green deployment**

Replaces the instances in the deployment group with new instances and deploys the latest application revision to them. After instances in the replacement environment are registered with a load balancer, instances from the original environment are deregistered and can be terminated.



Add instances

Identify the instances you want to include in the deployment group. We will deploy the application revision to the instances that match the instance tag keys and values or Auto Scaling group names you specify.

Requirements for each instance in the deployment:

1. Each Amazon EC2 instance must be launched with the correct IAM instance profile attached. Learn more
2. Each Amazon EC2 instance must have identifying Amazon EC2 tags (Learn more) or be in an Auto Scaling group. Learn more
3. Each on-premises instance must have an associated IAM user, identifying on-premises instance tags, and a configuration file. Learn more
4. The AWS CodeDeploy agent must be installed and running on each instance. Learn more

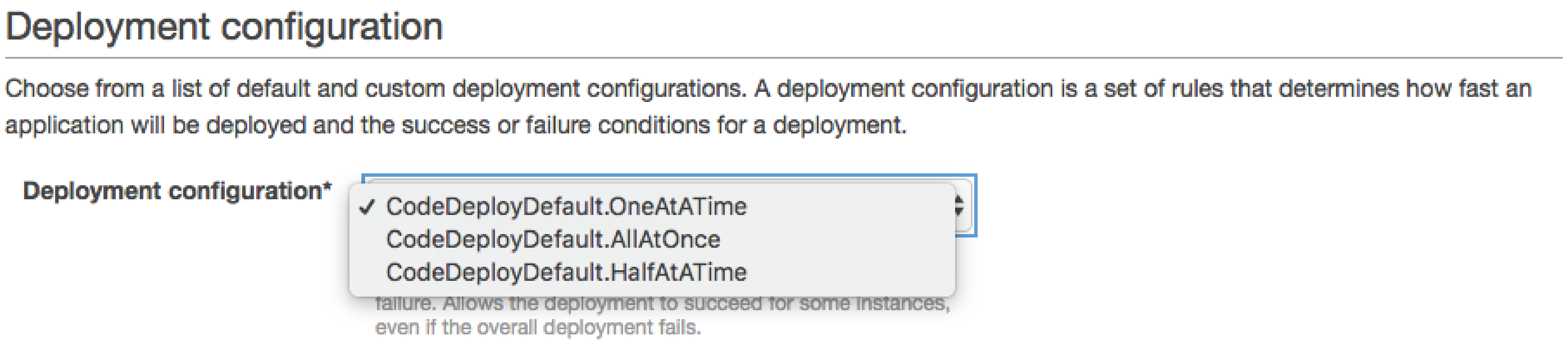
Search by tags

	Tag type	Key	Value	Instances	
1	Auto Scaling group ▾	website-staging-AutoScalingGroup-1PKMO5024O7M1 ▾		2	⊗
2	Amazon EC2 ▾	▾	▾		⊗

Total matching instances: 2

« ‹ **1 to 2 of 2 instances** › »

Instance ID	Status	Instance type
i-08560c56aaef28103	Healthy	Amazon EC2 - ASG
i-0aa27af98b810c2a8	Healthy	Amazon EC2 - ASG



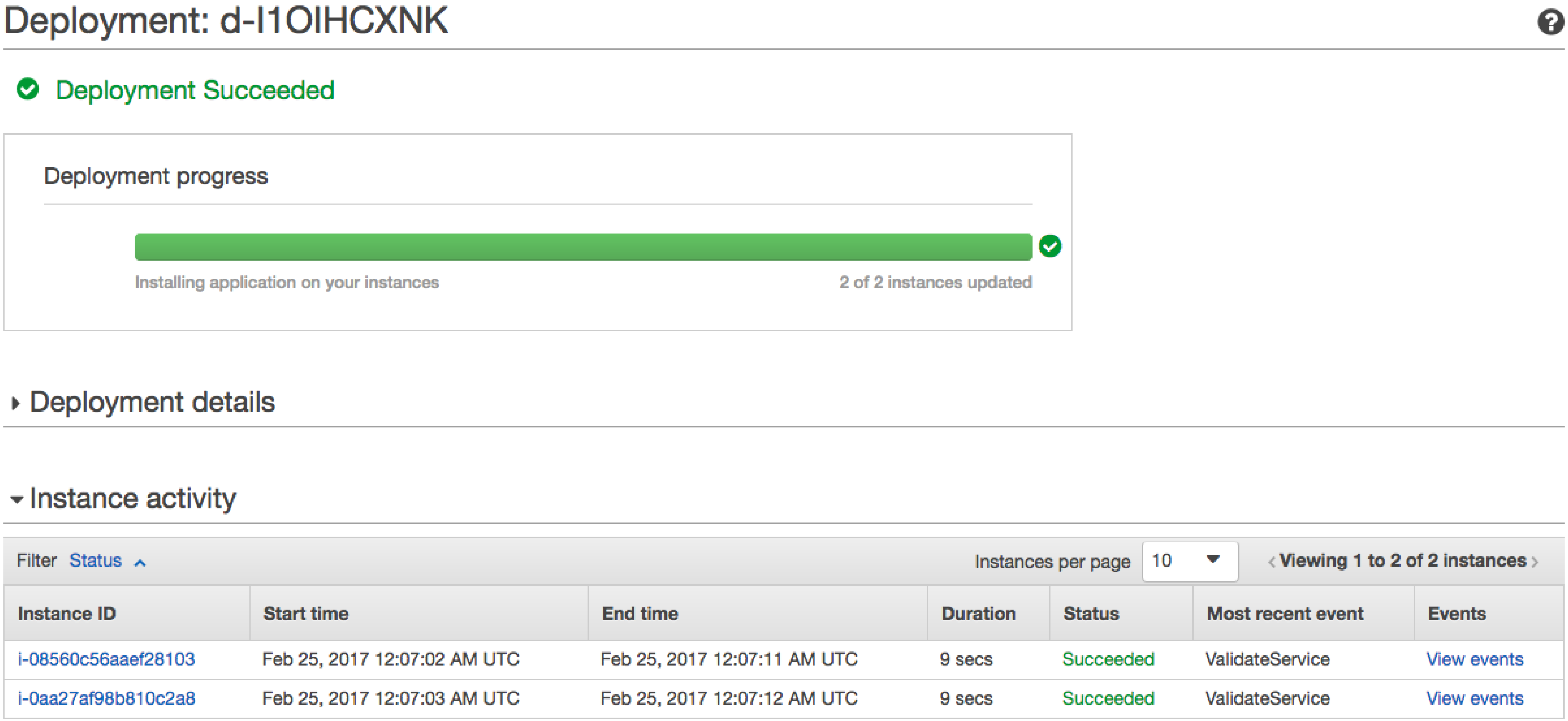
Deployment configuration

Choose from a list of default and custom deployment configurations. A deployment configuration is a set of rules that determines how fast an application will be deployed and the success or failure conditions for a deployment.

Deployment configuration*

✓ CodeDeployDefault.OneAtATime
CodeDeployDefault.AllAtOnce
CodeDeployDefault.HalfAtATime

failure. Allows the deployment to succeed for some instances, even if the overall deployment fails.



Deployment: d-I1OIHCXNK

✓ Deployment Succeeded

Deployment progress

✓

Installing application on your instances

2 of 2 instances updated

▸ Deployment details

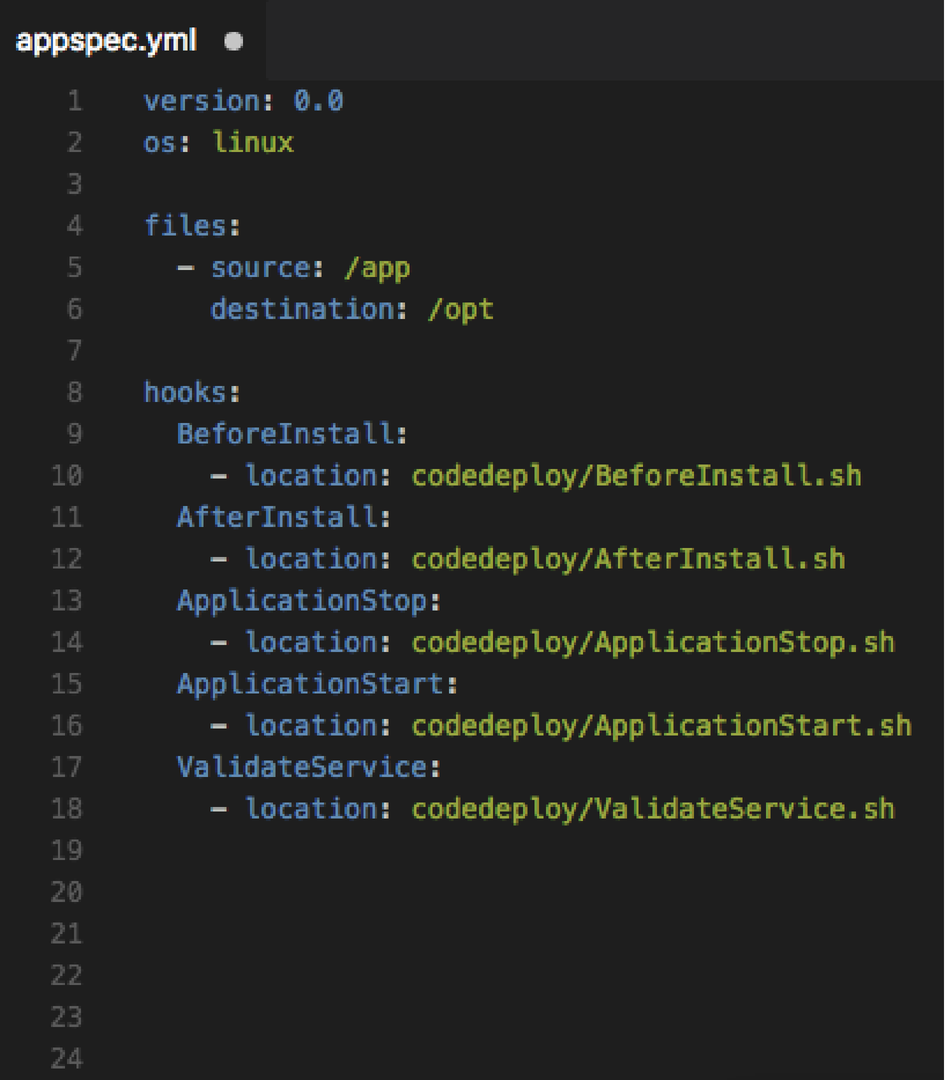
▾ Instance activity

Filter Status ▴

Instances per page 10 ▾

‹ Viewing 1 to 2 of 2 instances ›

Instance ID	Start time	End time	Duration	Status	Most recent event	Events
i-08560c56aaef28103	Feb 25, 2017 12:07:02 AM UTC	Feb 25, 2017 12:07:11 AM UTC	9 secs	Succeeded	ValidateService	View events
i-0aa27af98b810c2a8	Feb 25, 2017 12:07:03 AM UTC	Feb 25, 2017 12:07:12 AM UTC	9 secs	Succeeded	ValidateService	View events

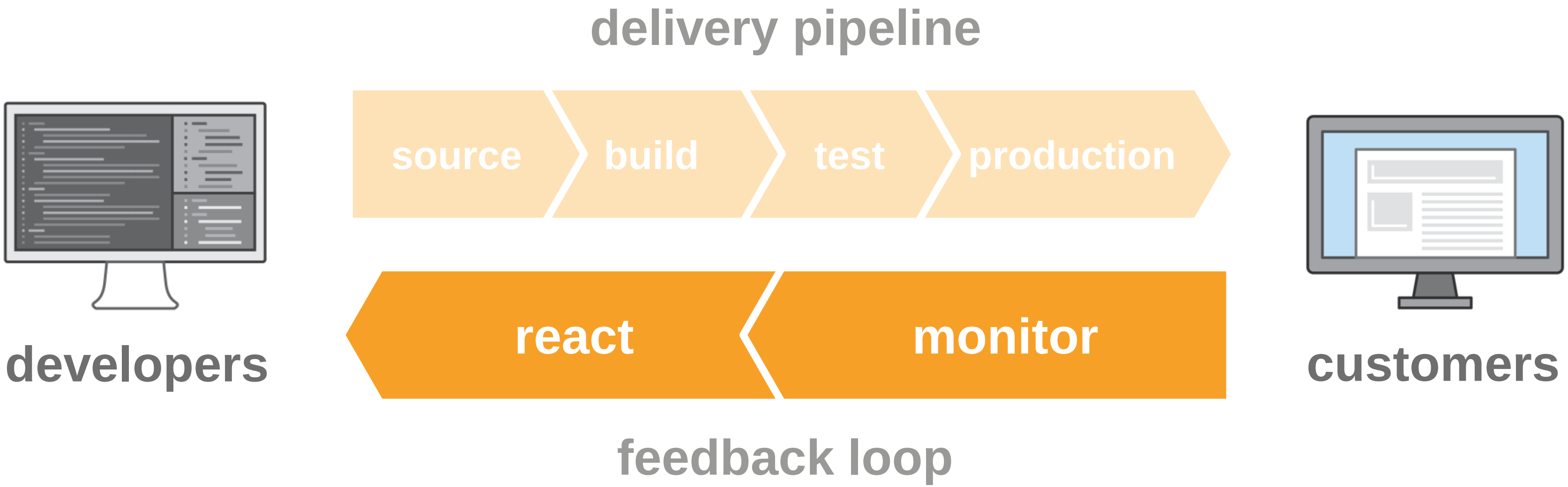


```
version: 0.0
os: linux

files:
  - source: /app
    destination: /opt

hooks:
  BeforeInstall:
    - location: codedeploy/BeforeInstall.sh
  AfterInstall:
    - location: codedeploy/AfterInstall.sh
  ApplicationStop:
    - location: codedeploy/ApplicationStop.sh
  ApplicationStart:
    - location: codedeploy/ApplicationStart.sh
  ValidateService:
    - location: codedeploy/ValidateService.sh
```

appspec.yml

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

Step 4: Feedback Loop



delivery pipeline

source

build

test

production

developers

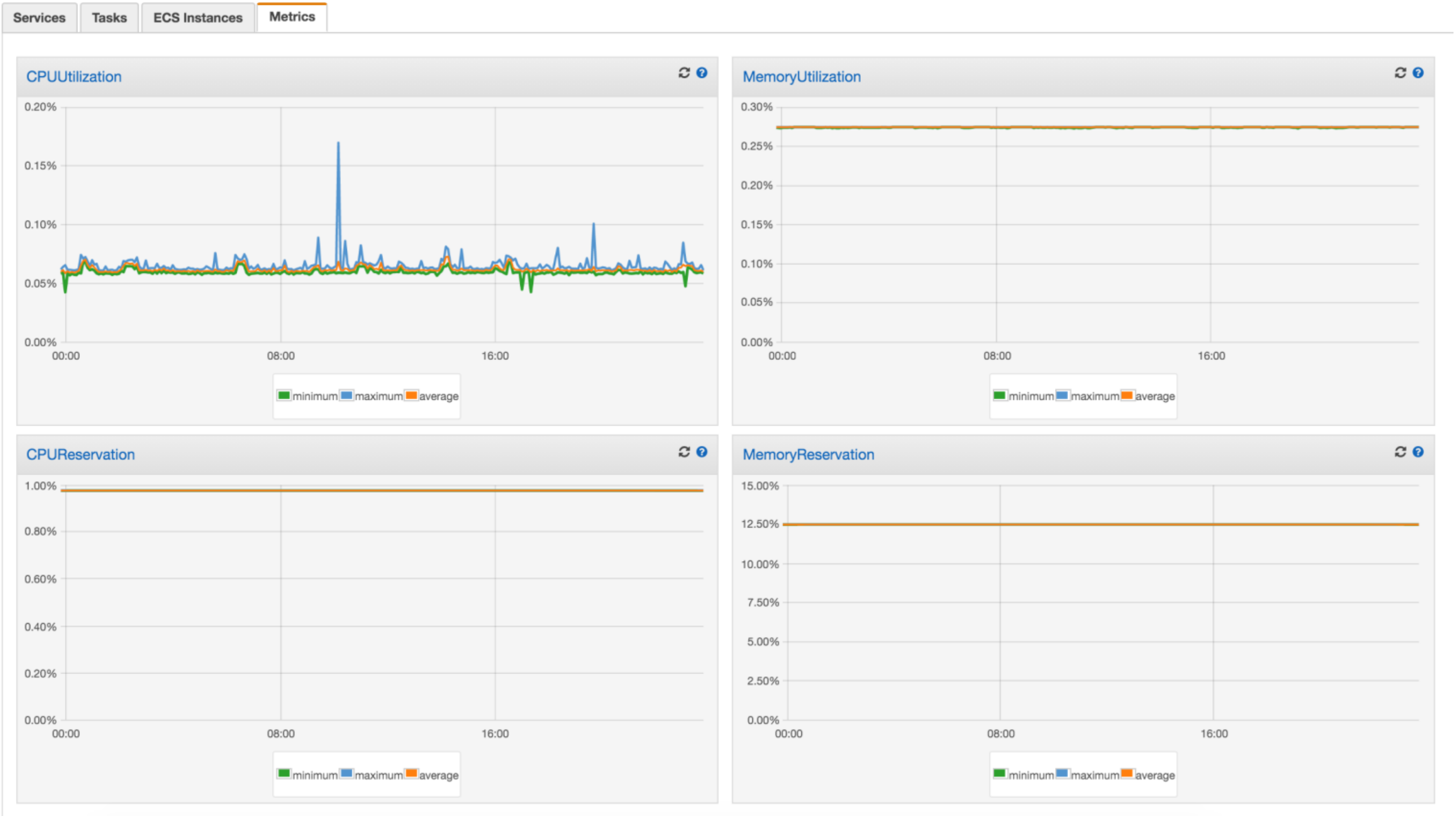
react

monitor

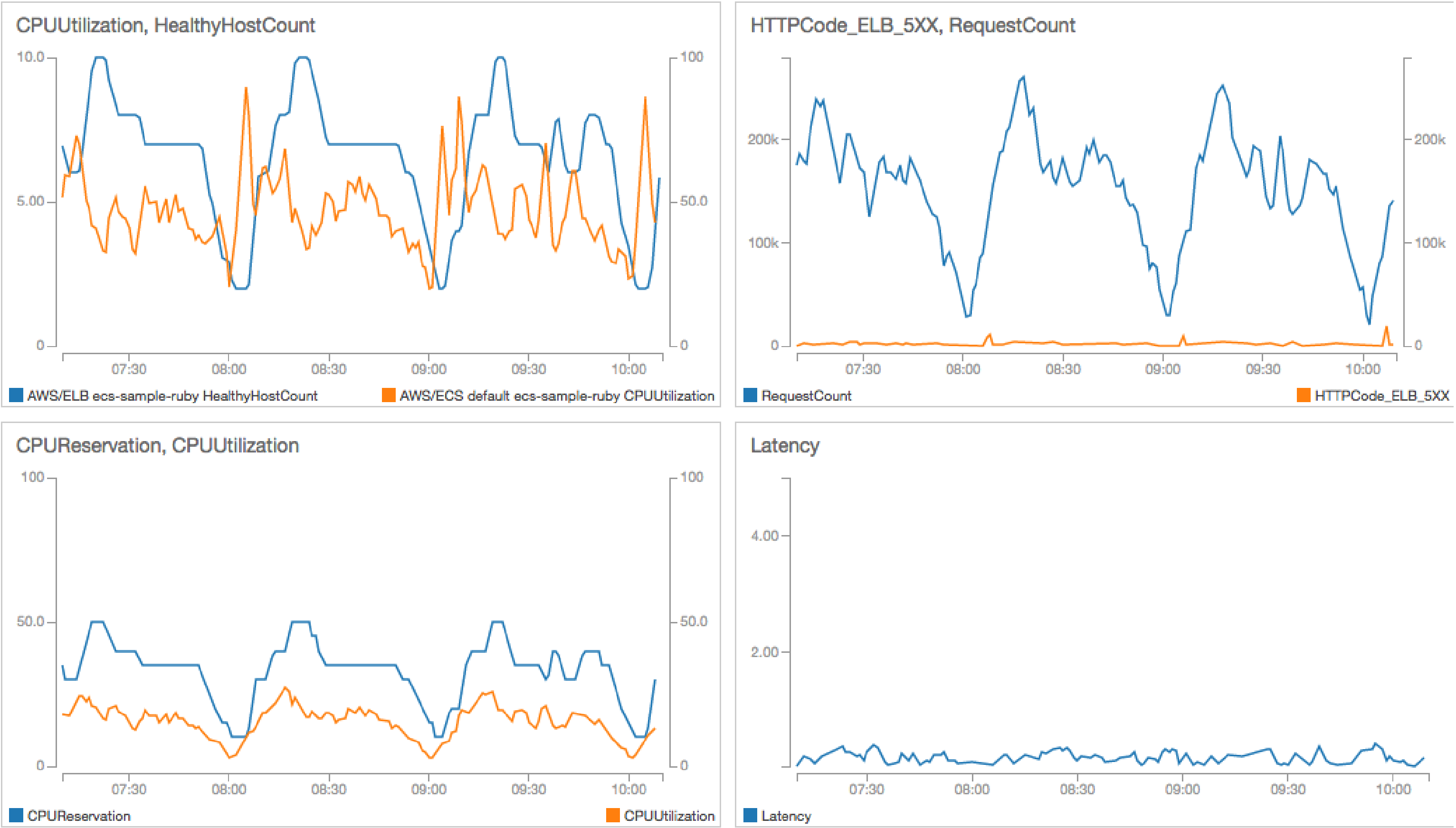
customers

feedback loop

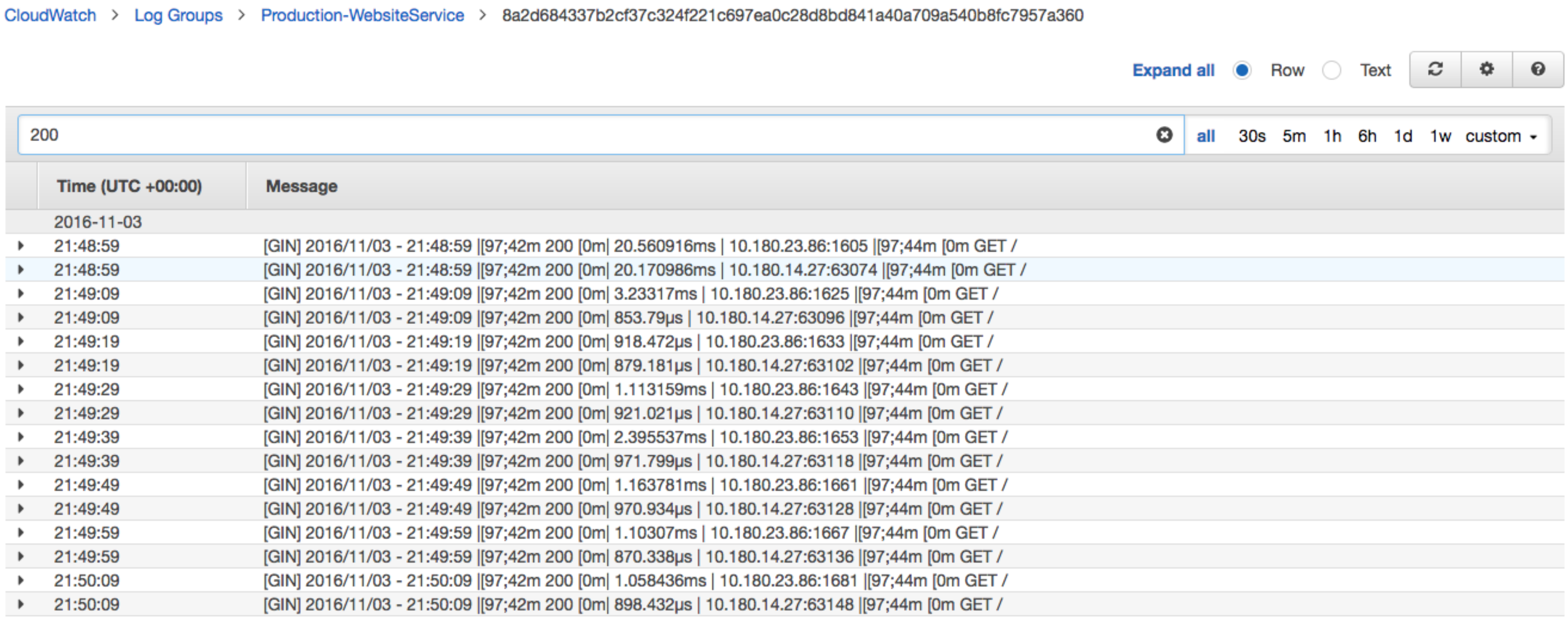
Monitoring with CloudWatch



Monitoring with CloudWatch



Centralized Logging with CloudWatch Logs

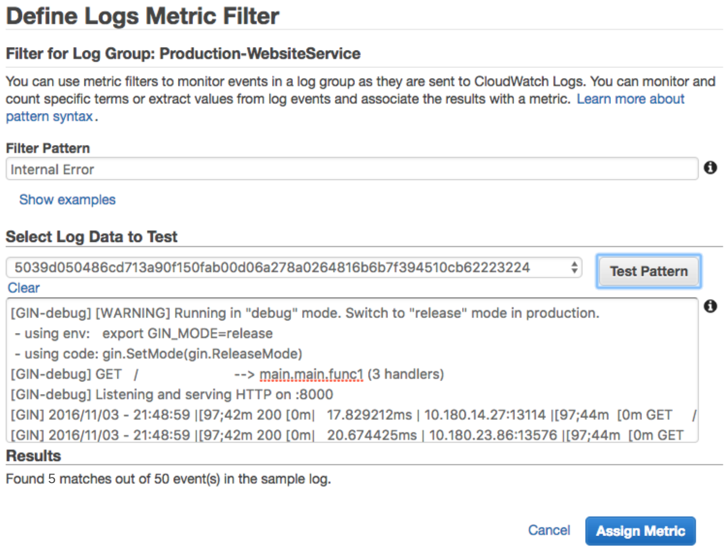


CloudWatch > Log Groups > Production-WebsiteService > 8a2d684337b2cf37c324f221c697ea0c28d8bd841a40a709a540b8fc7957a360

200

Time (UTC +00:00)	Message
2016-11-03	
21:48:59	[GIN] 2016/11/03 - 21:48:59 |[97;42m 200 [0m| 20.560916ms | 10.180.23.86:1605 |[97;44m [0m GET /
21:48:59	[GIN] 2016/11/03 - 21:48:59 |[97;42m 200 [0m| 20.170986ms | 10.180.14.27:63074 |[97;44m [0m GET /
21:49:09	[GIN] 2016/11/03 - 21:49:09 |[97;42m 200 [0m| 3.23317ms | 10.180.23.86:1625 |[97;44m [0m GET /
21:49:09	[GIN] 2016/11/03 - 21:49:09 |[97;42m 200 [0m| 853.79µs | 10.180.14.27:63096 |[97;44m [0m GET /
21:49:19	[GIN] 2016/11/03 - 21:49:19 |[97;42m 200 [0m| 918.472µs | 10.180.23.86:1633 |[97;44m [0m GET /
21:49:19	[GIN] 2016/11/03 - 21:49:19 |[97;42m 200 [0m| 879.181µs | 10.180.14.27:63102 |[97;44m [0m GET /
21:49:29	[GIN] 2016/11/03 - 21:49:29 |[97;42m 200 [0m| 1.113159ms | 10.180.23.86:1643 |[97;44m [0m GET /
21:49:29	[GIN] 2016/11/03 - 21:49:29 |[97;42m 200 [0m| 921.021µs | 10.180.14.27:63110 |[97;44m [0m GET /
21:49:39	[GIN] 2016/11/03 - 21:49:39 |[97;42m 200 [0m| 2.395537ms | 10.180.23.86:1653 |[97;44m [0m GET /
21:49:39	[GIN] 2016/11/03 - 21:49:39 |[97;42m 200 [0m| 971.799µs | 10.180.14.27:63118 |[97;44m [0m GET /
21:49:49	[GIN] 2016/11/03 - 21:49:49 |[97;42m 200 [0m| 1.163781ms | 10.180.23.86:1661 |[97;44m [0m GET /
21:49:49	[GIN] 2016/11/03 - 21:49:49 |[97;42m 200 [0m| 970.934µs | 10.180.14.27:63128 |[97;44m [0m GET /
21:49:59	[GIN] 2016/11/03 - 21:49:59 |[97;42m 200 [0m| 1.10307ms | 10.180.23.86:1667 |[97;44m [0m GET /
21:49:59	[GIN] 2016/11/03 - 21:49:59 |[97;42m 200 [0m| 870.338µs | 10.180.14.27:63136 |[97;44m [0m GET /
21:50:09	[GIN] 2016/11/03 - 21:50:09 |[97;42m 200 [0m| 1.058436ms | 10.180.23.86:1681 |[97;44m [0m GET /
21:50:09	[GIN] 2016/11/03 - 21:50:09 |[97;42m 200 [0m| 898.432µs | 10.180.14.27:63148 |[97;44m [0m GET /

Tip: Use Metric Filters with CloudWatch Logs



Define Logs Metric Filter

Filter for Log Group: Production-WebsiteService

You can use metric filters to monitor events in a log group as they are sent to CloudWatch Logs. You can monitor and count specific terms or extract values from log events and associate the results with a metric. Learn more about pattern syntax.

Filter Pattern

Internal Error

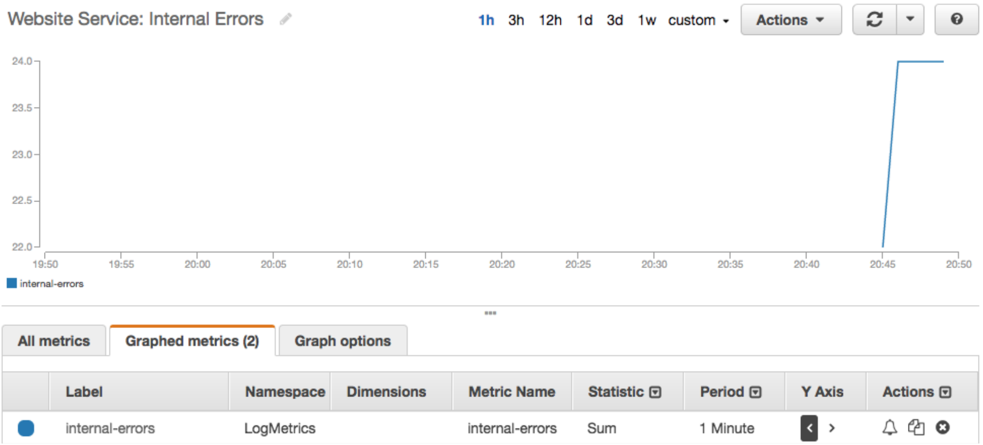
Select Log Data to Test

5039d050486cd713a90f150fab00d06a278a0264816b6b7f394510cb62223224

```
[GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in production.
 - using env:   export GIN_MODE=release
 - using code: gin.SetMode(gin.ReleaseMode)
[GIN-debug] GET   /                         --> main.main.func1 (3 handlers)
[GIN-debug] Listening and serving HTTP on :8000
[GIN] 2016/11/03 - 21:48:59 |[97;42m 200 [0m|   17.829212ms | 10.180.14.27:13114 |[97;44m  [0m GET     /
[GIN] 2016/11/03 - 21:48:59 |[97;42m 200 [0m|   20.674425ms | 10.180.23.86:13576 |[97;44m  [0m GET
```

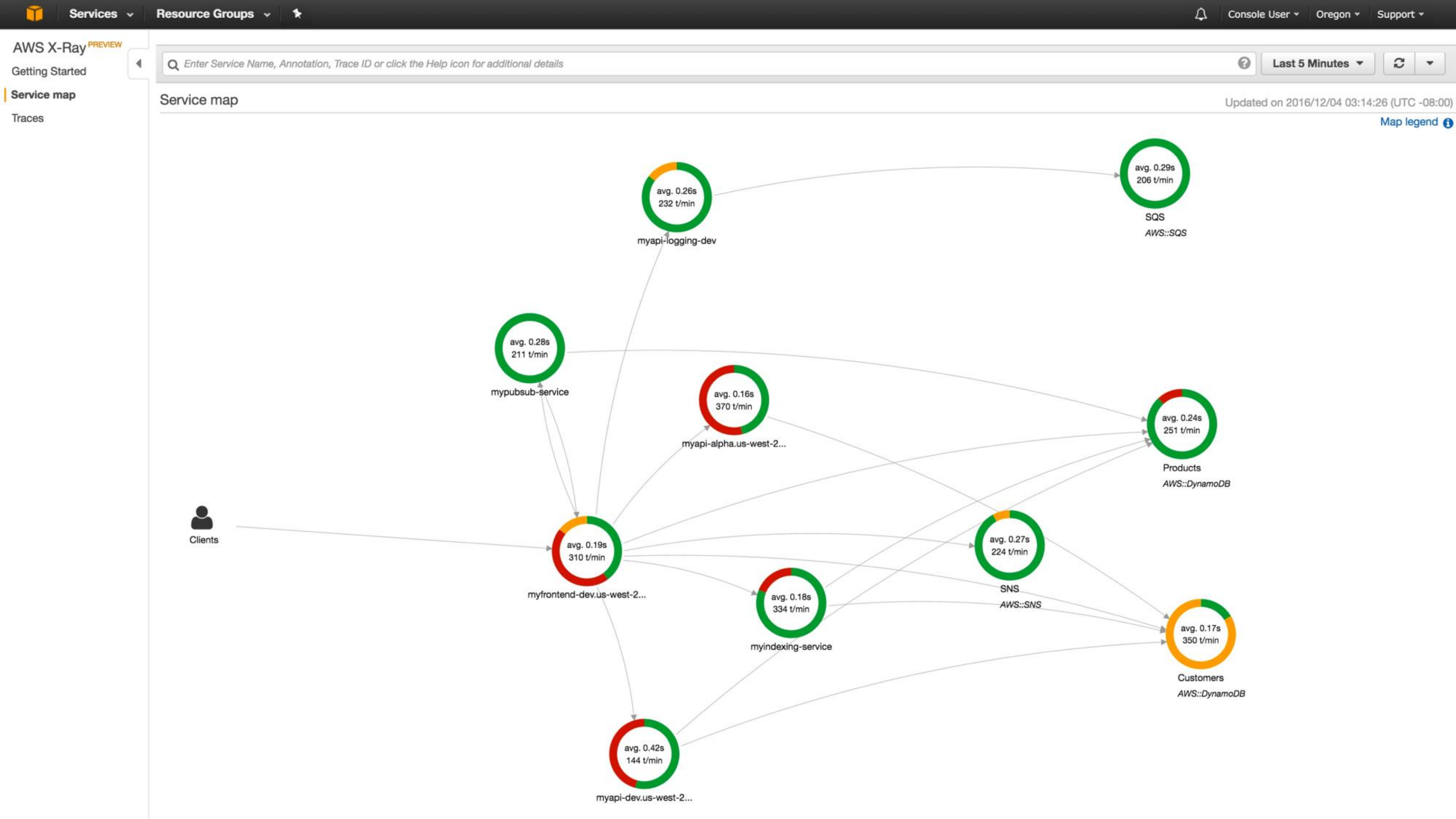
Results

Found 5 matches out of 50 event(s) in the sample log.



Website Service: Internal Errors

	Label	Namespace	Dimensions	Metric Name	Statistic	Period	Y Axis	Actions
●	internal-errors	LogMetrics		internal-errors	Sum	1 Minute	‹ ›	

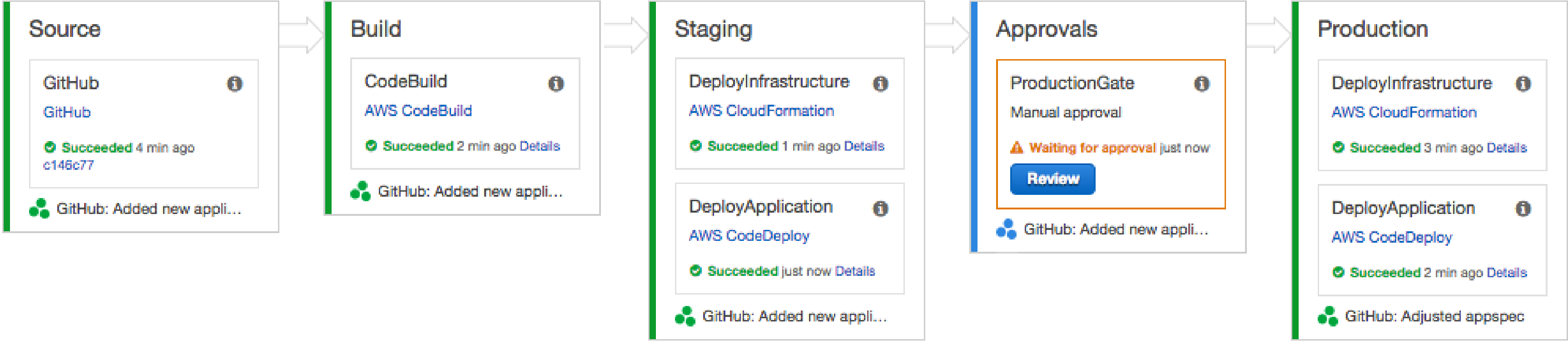


Enter Service Name, Annotation, Trace ID or click the Help icon for additional details

Service map

Updated on 2016/12/04 03:14:26 (UTC -08:00)

Putting it all together…



Source

GitHub
GitHub
Succeeded 4 min ago
c146c77

GitHub: Added new appli…

Build

CodeBuild
AWS CodeBuild
Succeeded 2 min ago Details

GitHub: Added new appli…

Staging

DeployInfrastructure
AWS CloudFormation
Succeeded 1 min ago Details

DeployApplication
AWS CodeDeploy
Succeeded just now Details

GitHub: Added new appli…

Approvals

ProductionGate
Manual approval
Waiting for approval just now

GitHub: Added new appli…

Production

DeployInfrastructure
AWS CloudFormation
Succeeded 3 min ago Details

DeployApplication
AWS CodeDeploy
Succeeded 2 min ago Details

GitHub: Adjusted appspec

Ready-to-go Examples

Building and deploying containers with CodeBuild & ECS:

https://github.com/awslabs/ecs-refarch-continuous-deployment

Example pipeline for Serverless (Lambda/API Gateway):

https://aws.amazon.com/blogs/compute/continuous-deployment-for-serverless-applications/

Example delivery pipeline for Go apps:

https://github.com/awslabs/golang-deployment-pipeline

Summary

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

amazon
web services

Love DevOps?

We’re hiring in Israel.

✉ pmaddox@amazon.com

@paulmaddox

Remember to complete your evaluations!