aws RE:INFORCE

aws RE:INFORCE

SDD308

Integrating security testing into your container build pipeline

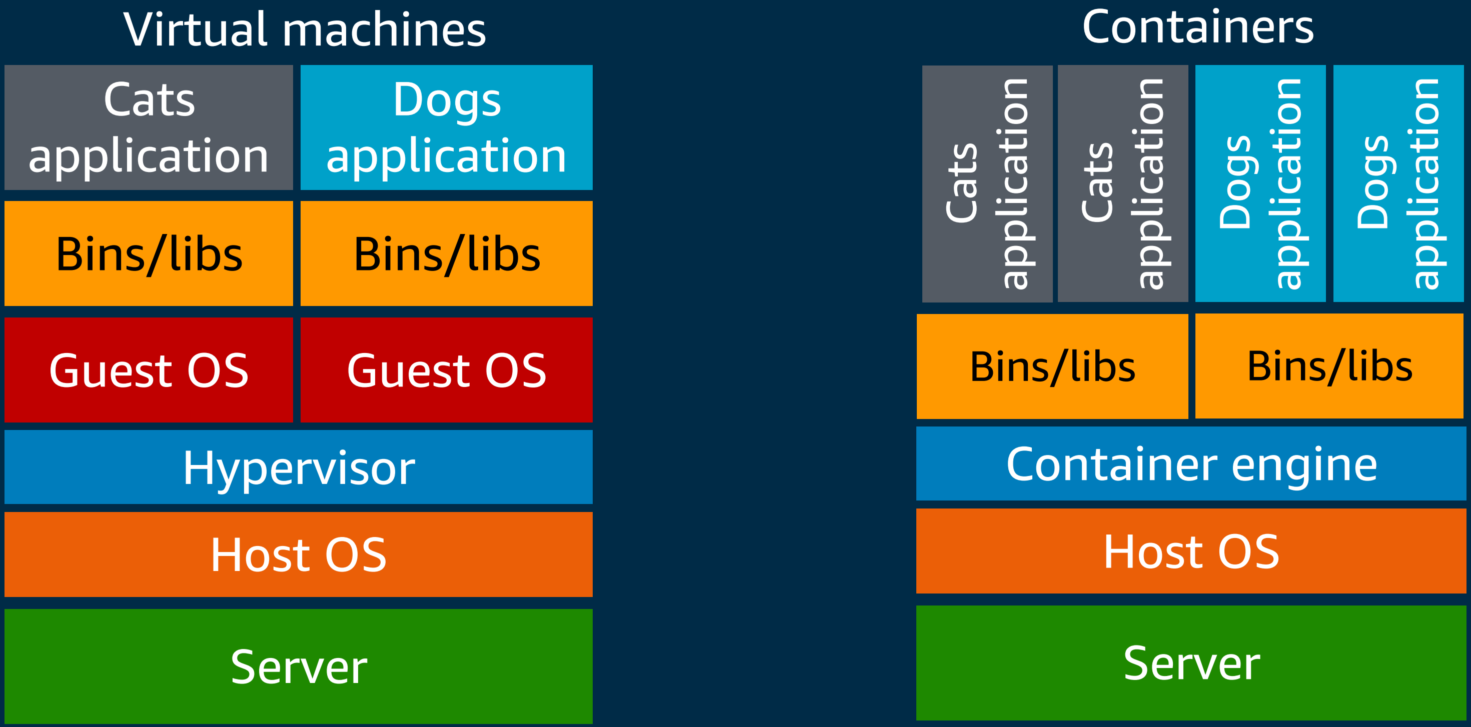
Aditya Patel
Security Architect
AWS

Avik Mukherjee
Senior Consultant
AWS

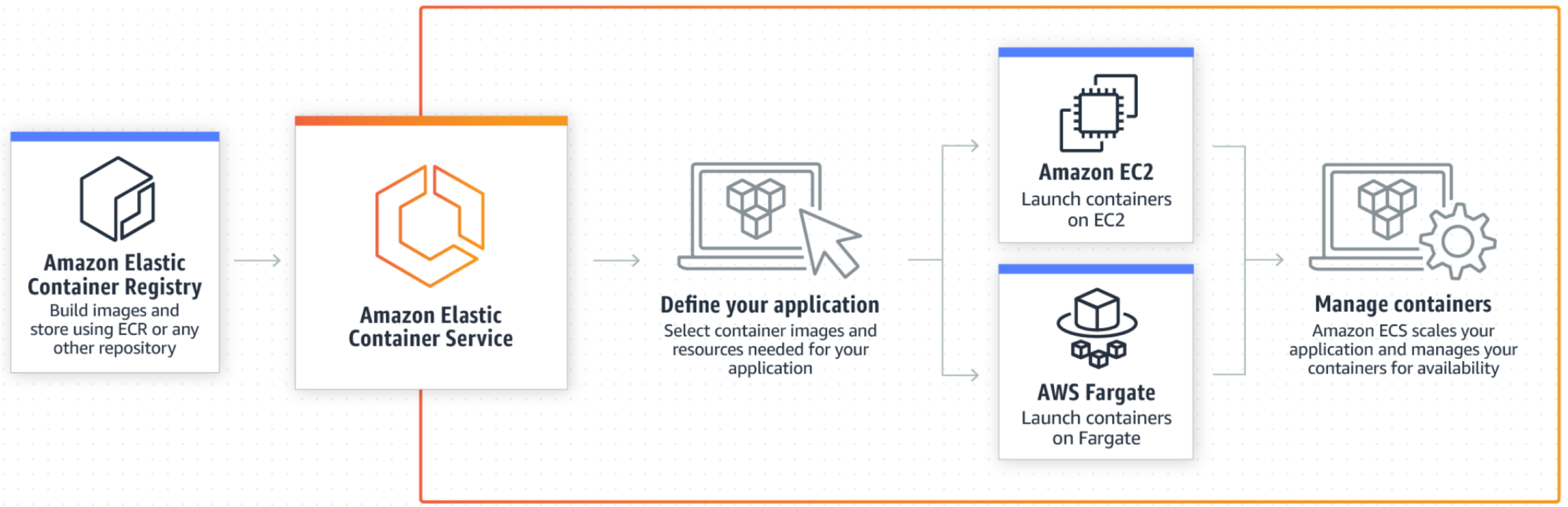
Goals

- Learn about container security using DevSecOps
- Learn about open-source container security tools and standards
- Learn about AWS development tools and DevOps services
- *Have fun while you’re at it!*

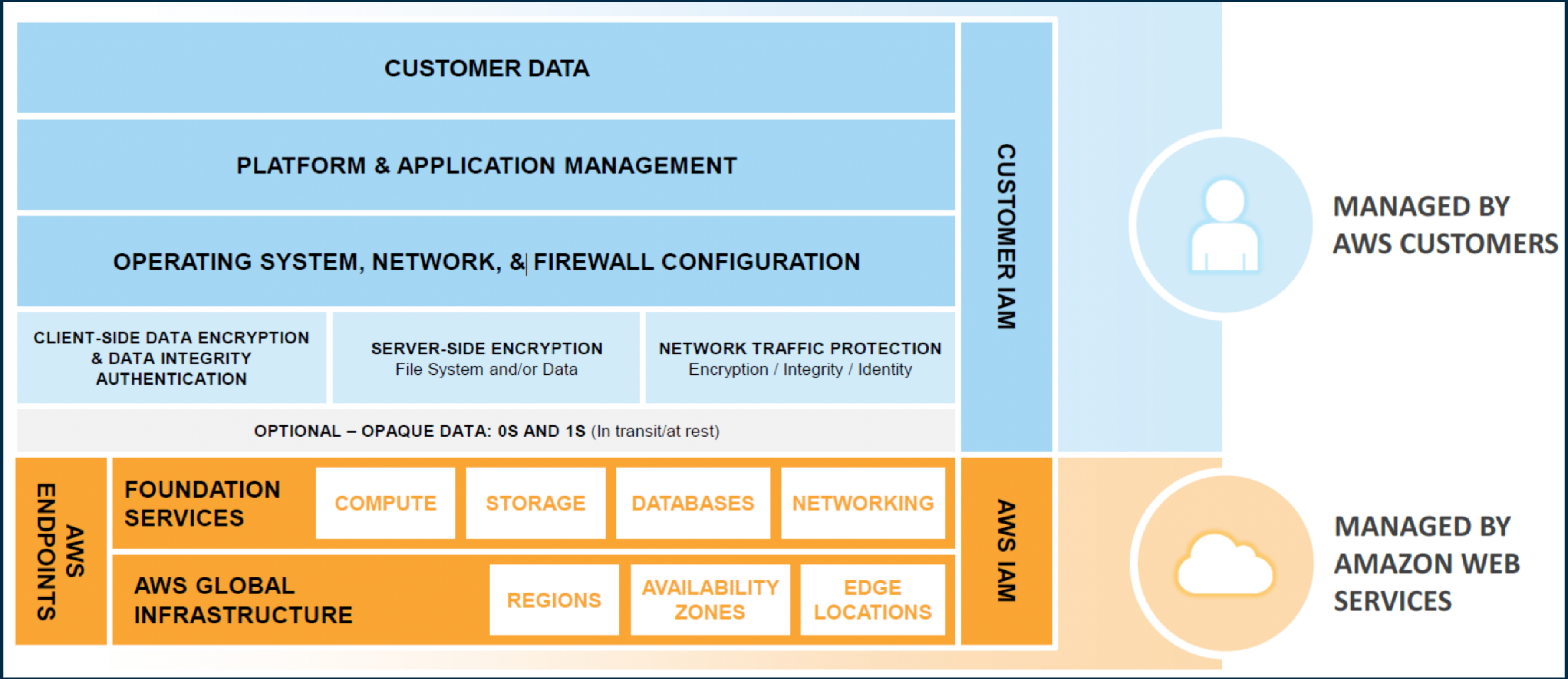
Why is container security different?



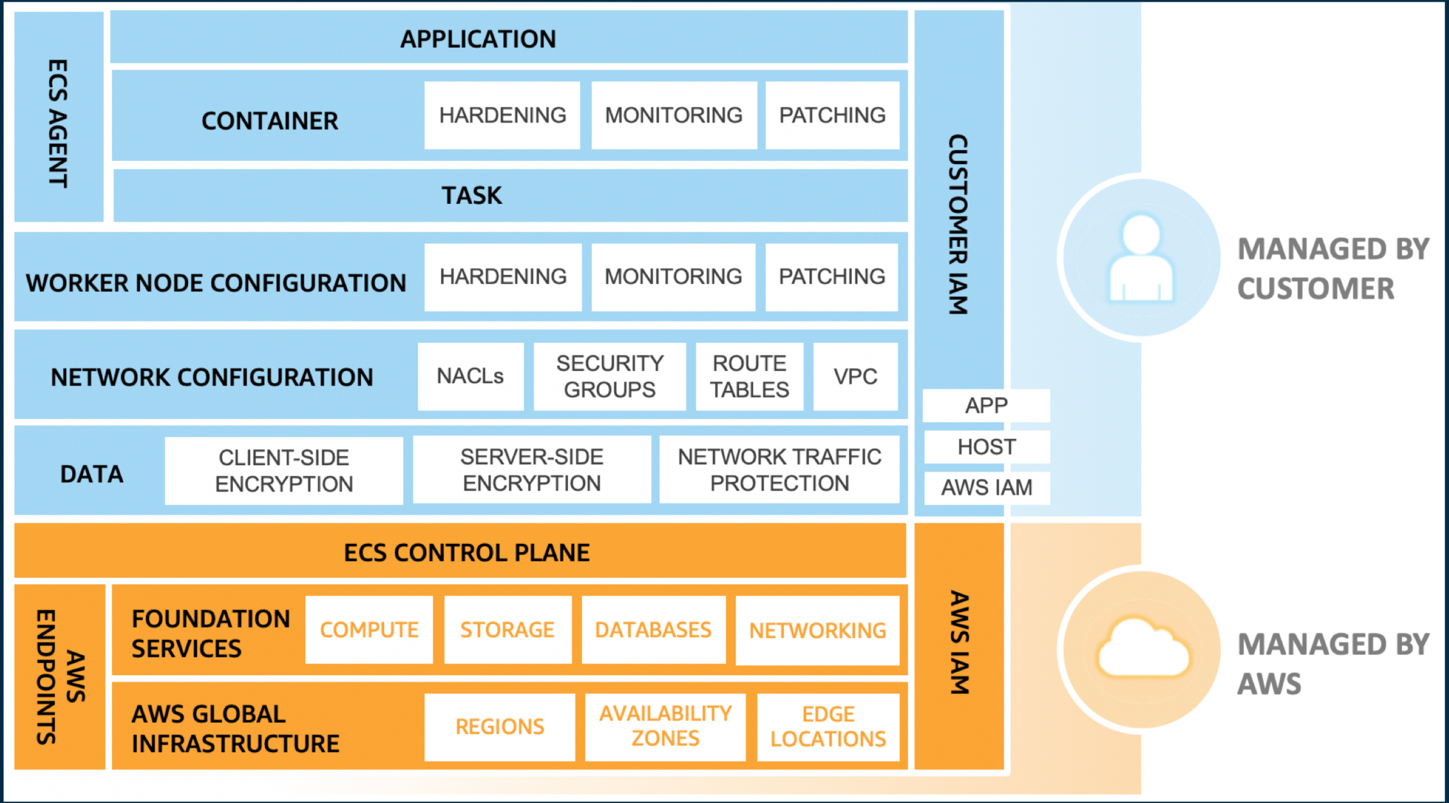
Containers on AWS


Amazon Elastic Container Registry
Build images and store using ECR or any other repository
Amazon Elastic Container Service
Define your application
Select container images and resources needed for your application
Amazon EC2
Launch containers on EC2
AWS Fargate
Launch containers on Fargate
Manage containers
Amazon ECS scales your application and manages your containers for availability

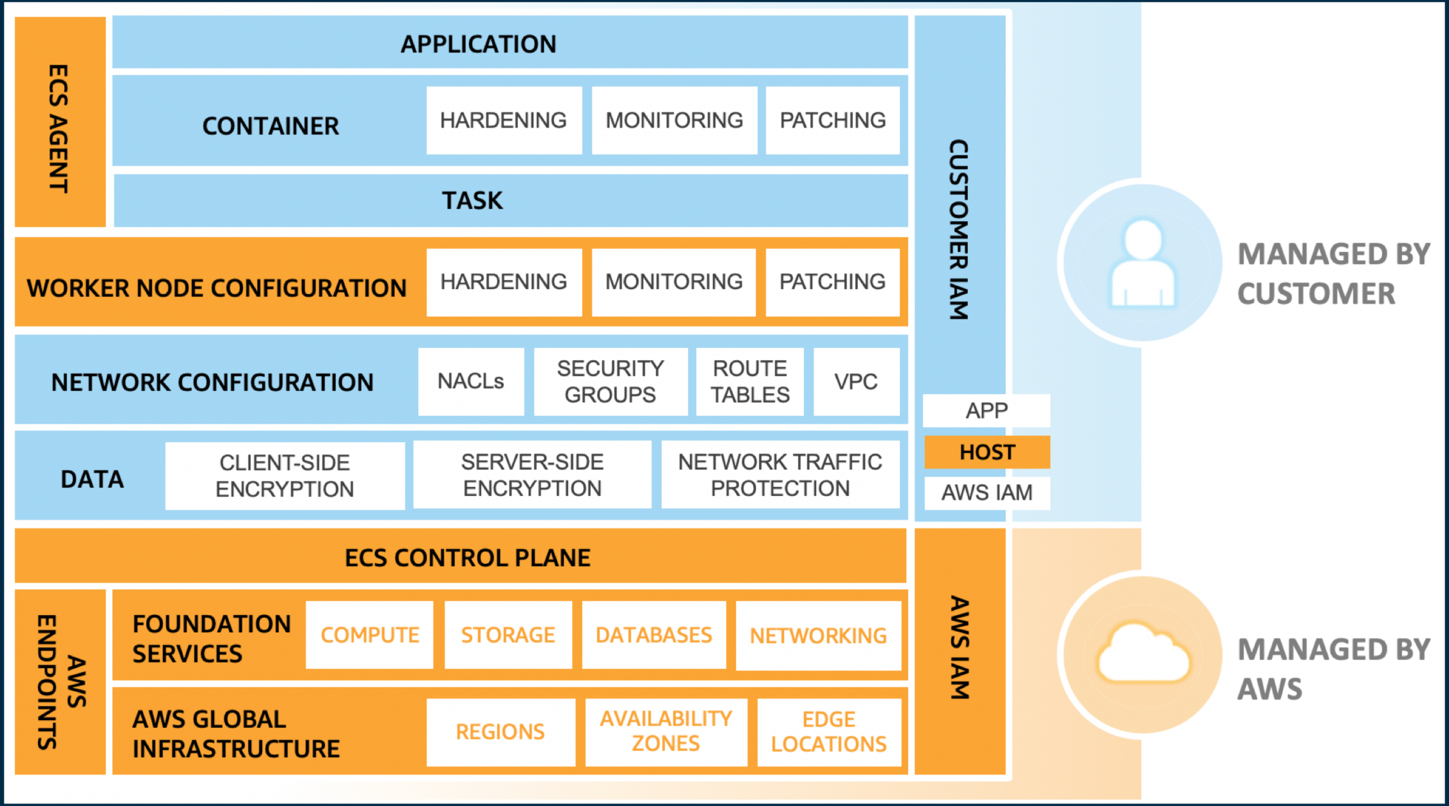
AWS shared responsibility model


CUSTOMER DATA
PLATFORM & APPLICATION MANAGEMENT
OPERATING SYSTEM, NETWORK, & FIREWALL CONFIGURATION
CLIENT-SIDE DATA ENCRYPTION & DATA INTEGRITY AUTHENTICATION
SERVER-SIDE ENCRYPTION File System and/or Data
NETWORK TRAFFIC PROTECTION Encryption / Integrity / Identity
OPTIONAL – OPAQUE DATA: 0S AND 1S (In transit/at rest)
CUSTOMER IAM
MANAGED BY AWS CUSTOMERS
AWS ENDPOINTS
FOUNDATION SERVICES
COMPUTE
STORAGE
DATABASES
NETWORKING
AWS GLOBAL INFRASTRUCTURE
REGIONS
AVAILABILITY ZONES
EDGE LOCATIONS
AWS IAM
MANAGED BY AMAZON WEB SERVICES

Amazon ECS: AWS shared responsibility model



AWS Fargate: AWS shared responsibility model



Automated pipelines: DevSecOps

Speaking of automation, you should automate everything, including

- Code and container builds
- Infrastructure via infrastructure as code patterns
- Deployments
- Process of making things self-healing
- Security!

Make it fast and easy for your team to do the right thing!

Container security threats

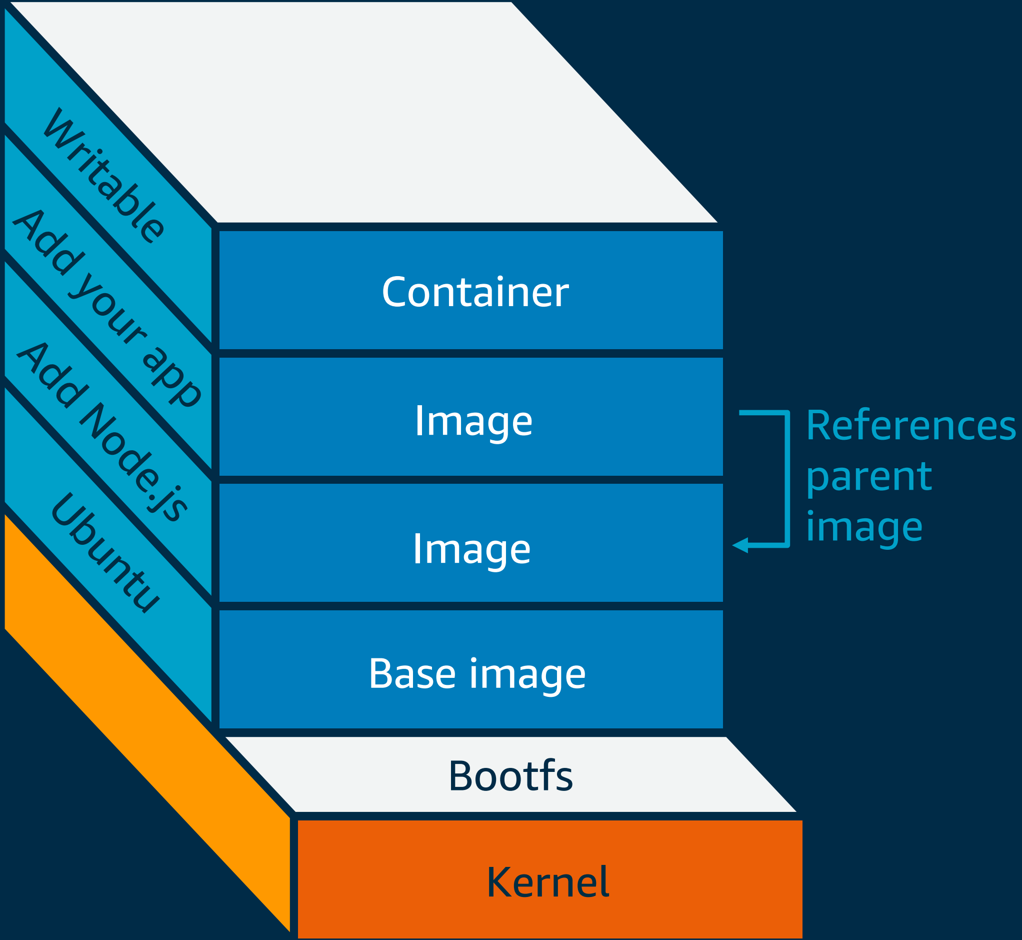
- Host security
- Image security
- Denial of service
- Credentials and secrets
- Container breakouts
- Runtime security

Container security threats

- Host security
- **Image security**
- Denial of service
- **Credentials and secrets**
- Container breakouts
- Runtime security

Security best practices for container images

- **Less is more (secure)**
- No secrets in them
- One service per container
- Minimize container footprint
- Include only what is **needed** at runtime



Security best practices for container images

- Use known and trusted base images
- Scan the image for CVEs
- Specify USER in Dockerfile (otherwise it's a root)
- Use unique and informative image tags
- Be able to tell which commit at a glance

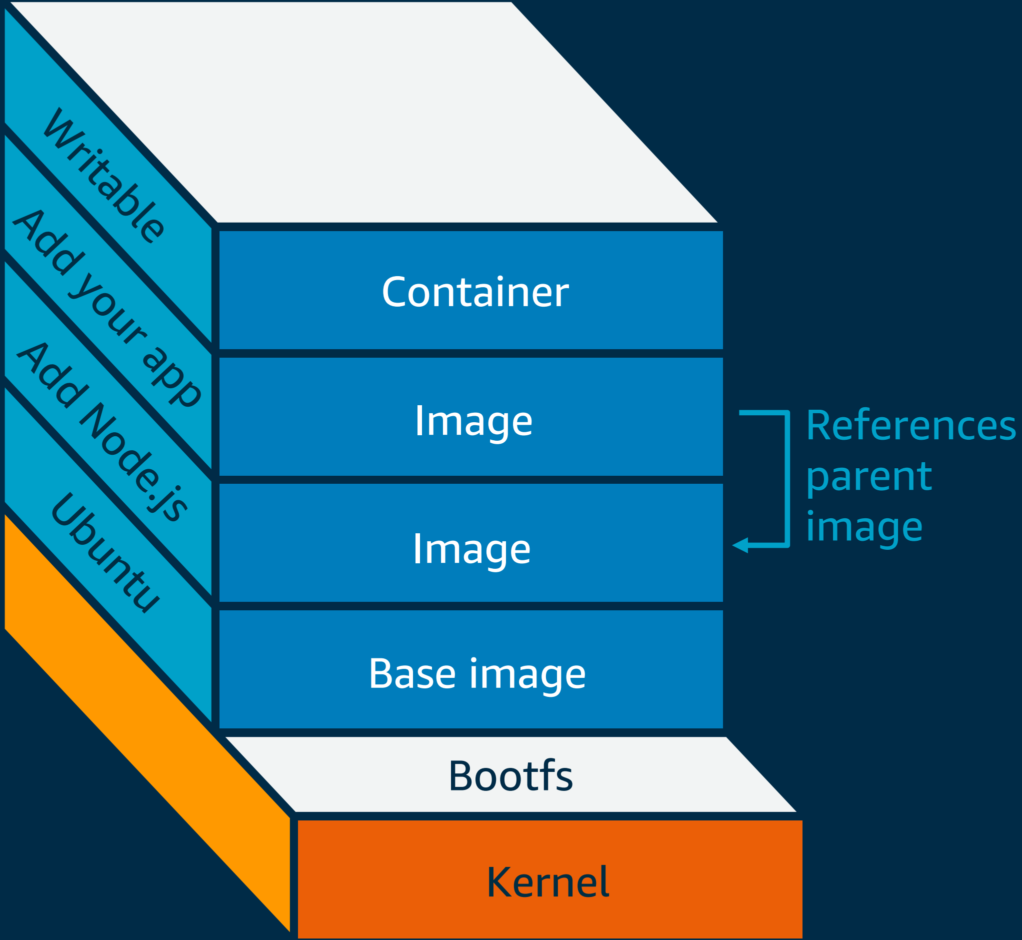
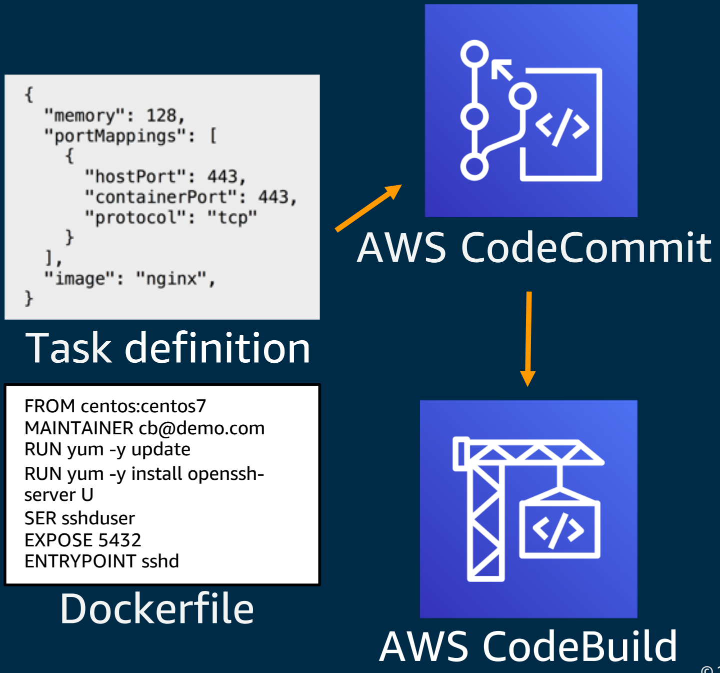


Image security

- Docker linting: Validation of Docker configuration (PCI DSS v3.2.1 Req 2.2)
  - hadolint
  - dockerfile_lint
- Secrets scanning in images (PCI DSS v3.2.1 Req 6.3.1)
  - truffleHog
  - git-secrets
- Vulnerability scanning of images in your build pipeline (PCI DSS v3.2.1 Req 6.1)
  - Anchore Open-Source Engine
  - CoreOS Clair

DevSecOps container pipeline



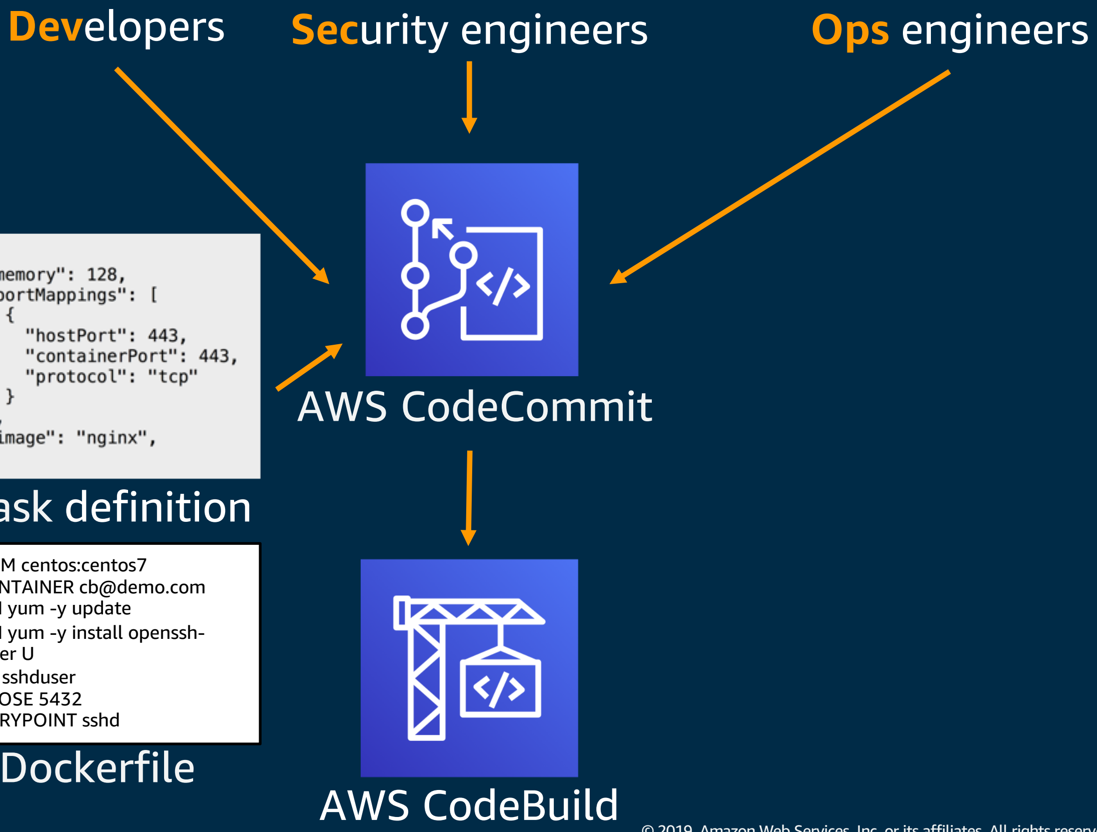
```
{
  "memory": 128,
  "portMappings": [
    {
      "hostPort": 443,
      "containerPort": 443,
      "protocol": "tcp"
    }
  ],
  "image": "nginx",
}
```

Task definition

```
FROM centos:centos7
MAINTAINER cb@demo.com
RUN yum -y update
RUN yum -y install openssh-
server U
SER sshduser
EXPOSE 5432
ENTRYPOINT sshd
```

Dockerfile

DevSecOps container pipeline



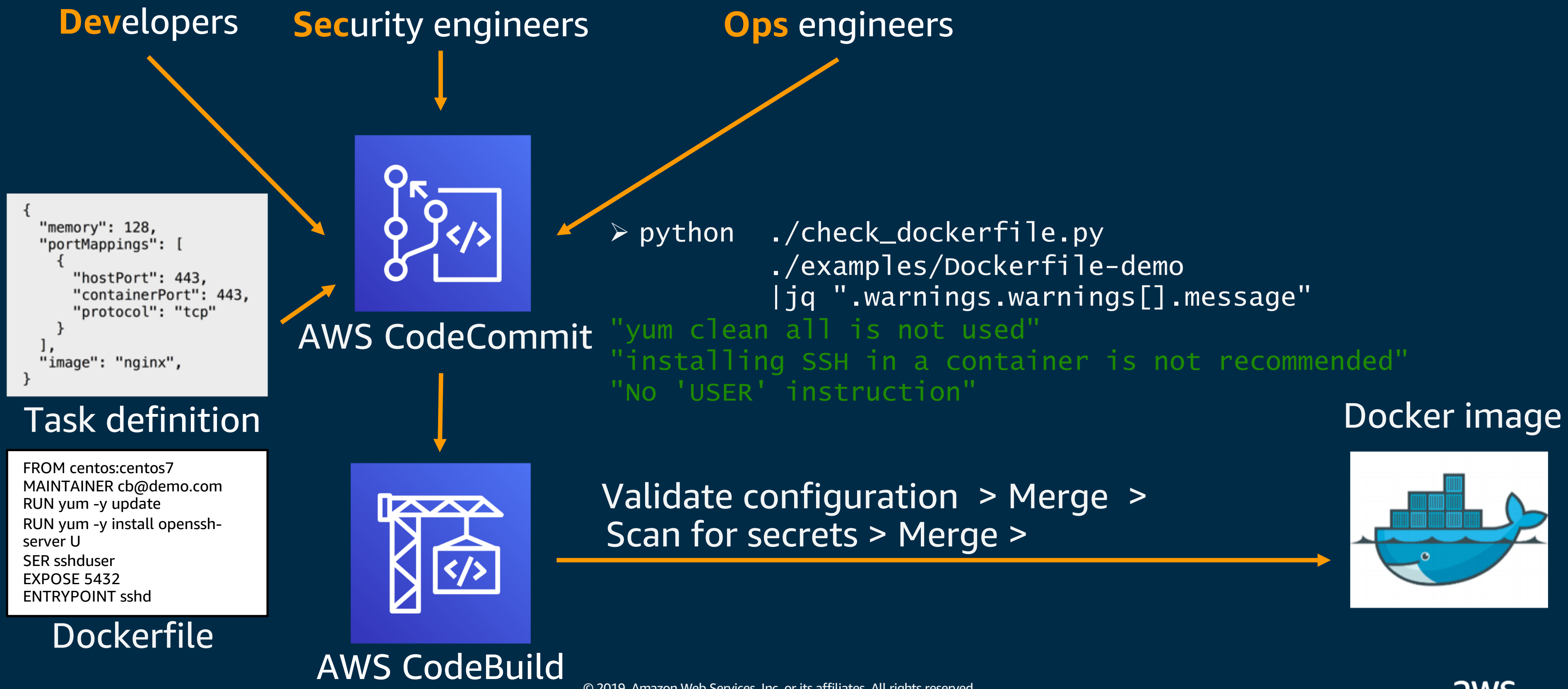
```
{
  "memory": 128,
  "portMappings": [
    {
      "hostPort": 443,
      "containerPort": 443,
      "protocol": "tcp"
    }
  ],
  "image": "nginx",
}
```

Task definition

```
FROM centos:centos7
MAINTAINER cb@demo.com
RUN yum -y update
RUN yum -y install openssh-
server U
SER sshduser
EXPOSE 5432
ENTRYPOINT sshd
```

Dockerfile

DevSecOps container pipeline



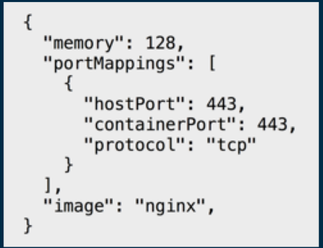
```
{
  "memory": 128,
  "portMappings": [
    {
      "hostPort": 443,
      "containerPort": 443,
      "protocol": "tcp"
    }
  ],
  "image": "nginx",
}
```

```
FROM centos:centos7
MAINTAINER cb@demo.com
RUN yum -y update
RUN yum -y install openssh-
server U
SER sshduser
EXPOSE 5432
ENTRYPOINT sshd
```

```
➢ python  ./check_dockerfile.py
          ./examples/Dockerfile-demo
          |jq ".warnings.warnings[].message"
"yum clean all is not used"
"installing SSH in a container is not recommended"
"No 'USER' instruction"
```

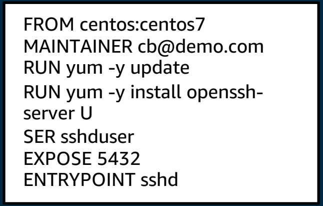
DevSecOps container pipeline

Developers **Security engineers** **Ops engineers**



```
{
  "memory": 128,
  "portMappings": [
    {
      "hostPort": 443,
      "containerPort": 443,
      "protocol": "tcp"
    }
  ],
  "image": "nginx",
}
```

Task definition

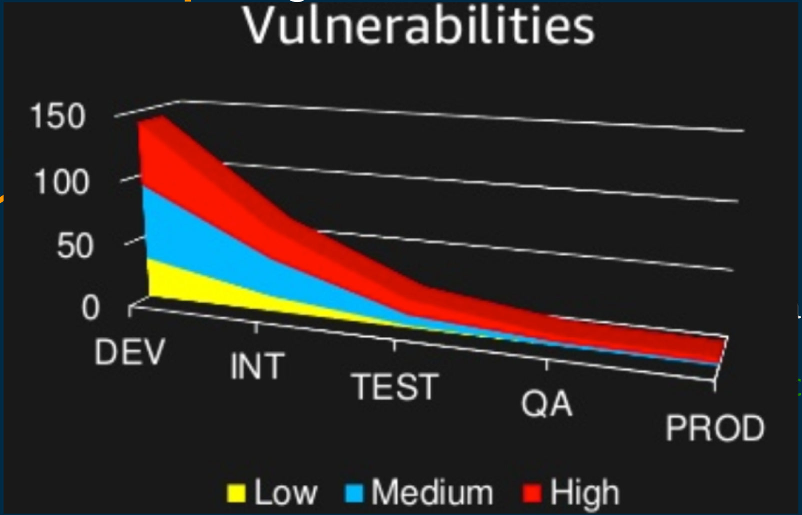


```
FROM centos:centos7
MAINTAINER cb@demo.com
RUN yum -y update
RUN yum -y install openssh-
server U
SER sshduser
EXPOSE 5432
ENTRYPOINT sshd
```

Dockerfile

AWS CodeCommit

AWS CodeBuild



ge"

commended"

Amazon EC2 container registry

Docker image

Validate configuration > Merge >
Scan for secrets > Merge >

Scan Docker image > Publish >

Credentials and secrets

AWS has Parameter Store and AWS Secrets Manager to store your secrets

They are integrated into Amazon ECS, but you need to call them within the pod on Kubernetes via AWS CLI or SDK

Assigning an IAM role to an instance, task, or function means that the right AWS access key and secret to call the AWS CLI or SDK are transparently obtained and rotated

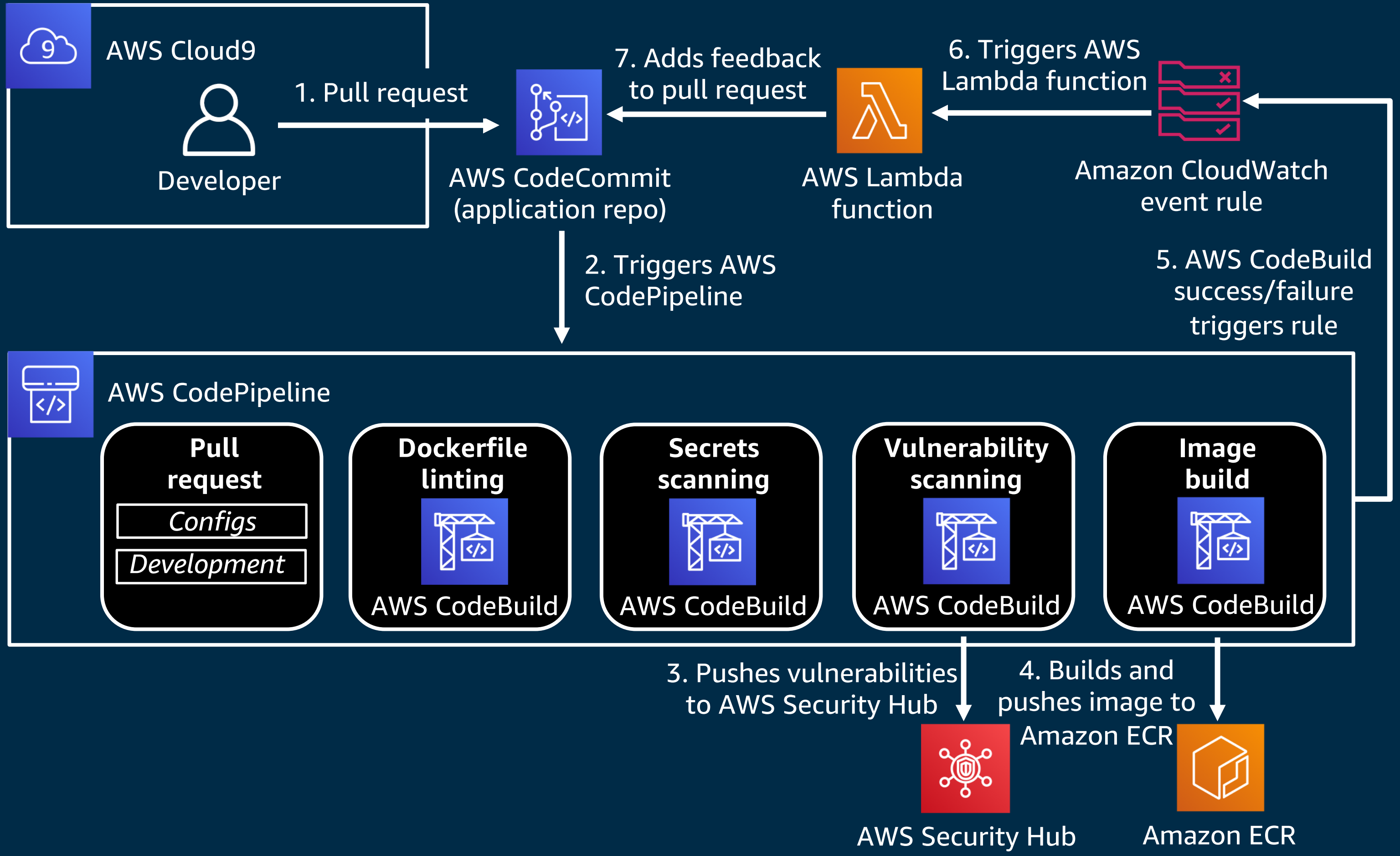
The image part with relationship ID rId32 was not found in the file.

The image part with relationship ID rId30 was not found in the file.

The image part with relationship ID rId9 was not found in the file.

Workshop architecture: From 10,000 feet

RE:INFORCE


AWS Cloud9
Developer
1. Pull request
AWS CodeCommit (application repo)
7. Adds feedback to pull request
AWS Lambda function
6. Triggers AWS Lambda function
Amazon CloudWatch event rule
2. Triggers AWS CodePipeline
5. AWS CodeBuild success/failure triggers rule
AWS CodePipeline
Pull request
Configs
Development
Dockerfile linting
AWS CodeBuild
Secrets scanning
AWS CodeBuild
Vulnerability scanning
AWS CodeBuild
Image build
AWS CodeBuild
3. Pushes vulnerabilities to AWS Security Hub
4. Builds and pushes image to Amazon ECR
AWS Security Hub
Amazon ECR

Let's build and have fun!

Integrating security testing into your container build pipeline: Workshop prerequisites

- Start with https://container-devsecops.awssecworkshops.com
- Module 0: Environment Setup (15 min.)
    - Use *AWS Event Engine* Option (first option)
    - Use your *Hash* to login to your AWS account

Use “AWS Event Engine”

Use “us-east-2”

Integrating security testing into your container build pipeline: Module 1

- Start with https://container-devsecops.awssecworkshops.com
- Module 1: Dockerfile linting (15 mins)
  - Create buildspec file
  - Add hadolint configuration
- Module 2: Secrets scanning
- Module 3: Vulnerability scanning
- Module 4: Pipeline testing

Integrating security testing into your container build pipeline: Module 2

- Start with https://container-devsecops.awssecworkshops.com
- ~~Module 1: Dockerfile linting~~
- Module 2: Secrets scanning (15 mins)
  - Create buildspec file
  - Add truffleHog RegEx configuration
- Module 3: Vulnerability scanning
- Module 4: Pipeline testing

Integrating security testing into your container build pipeline: Module 3

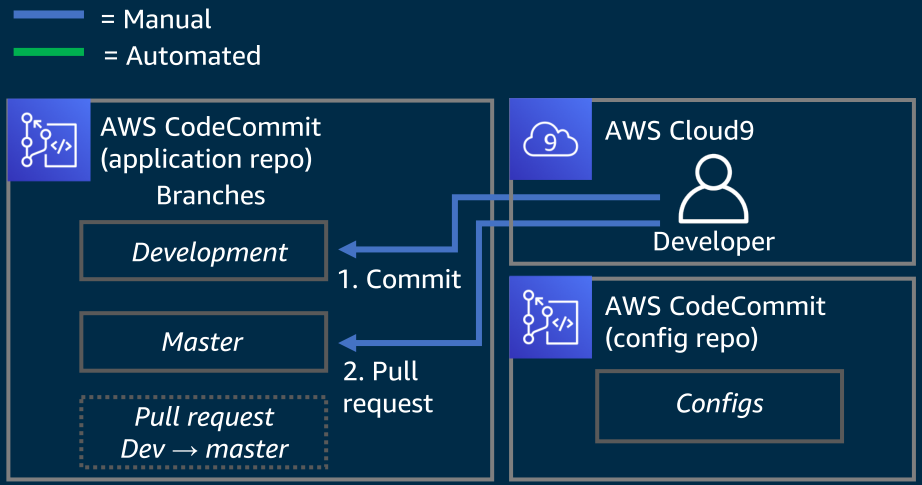
- Start with https://container-devsecops.awssecworkshops.com
- ~~Module 1: Dockerfile linting~~
- ~~Module 2: Secrets scanning~~
- Module 3: Vulnerability scanning (15 mins)
  - Create buildspec file
  - Add command to run Anchore
- Module 4: Pipeline testing

Integrating security testing into your container build pipeline: Module 4

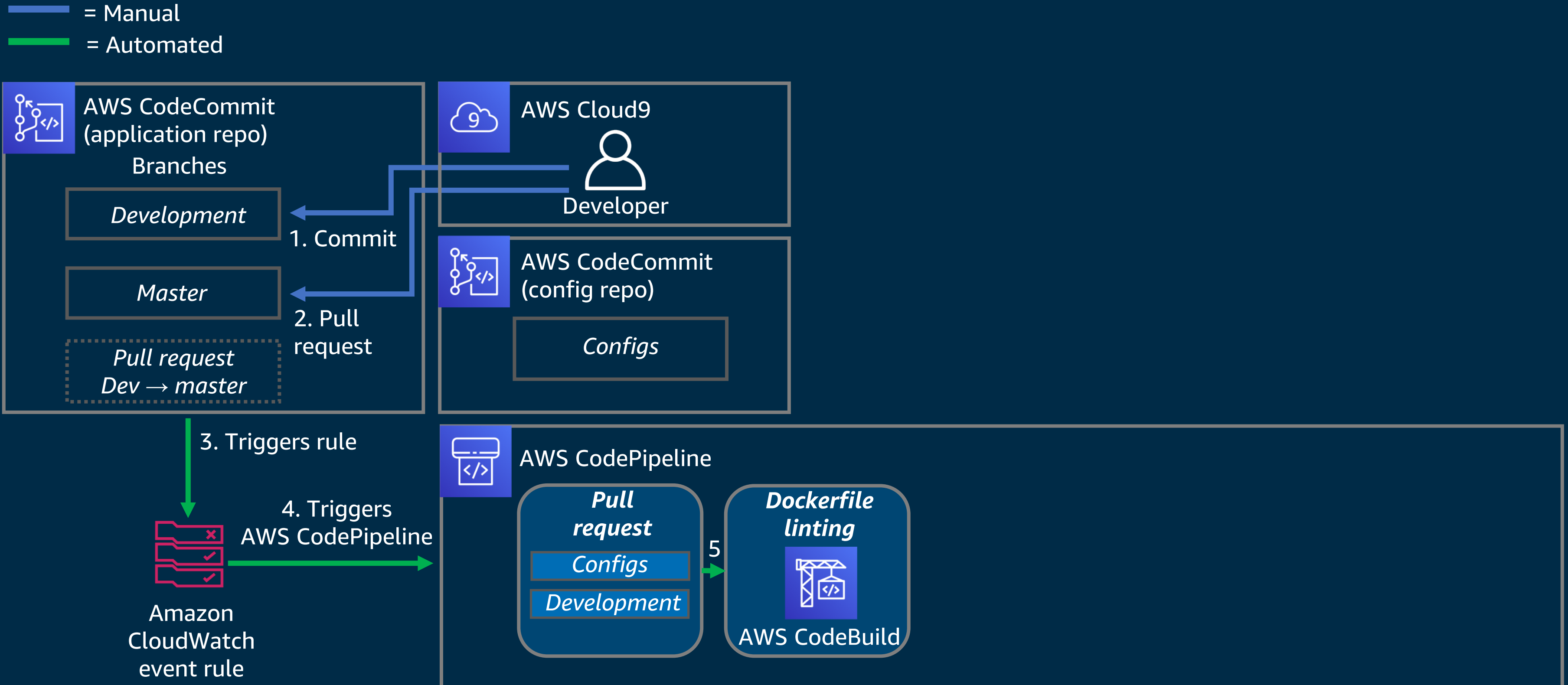
- Start with https://container-devsecops.awssecworkshops.com
- ~~Module 1: Dockerfile linting~~
- ~~Module 2: Secrets scanning~~
- ~~Module 3: Vulnerability scanning~~
- Module 4: Pipeline testing (15 mins)
  - Make a commit
  - View feedback loop

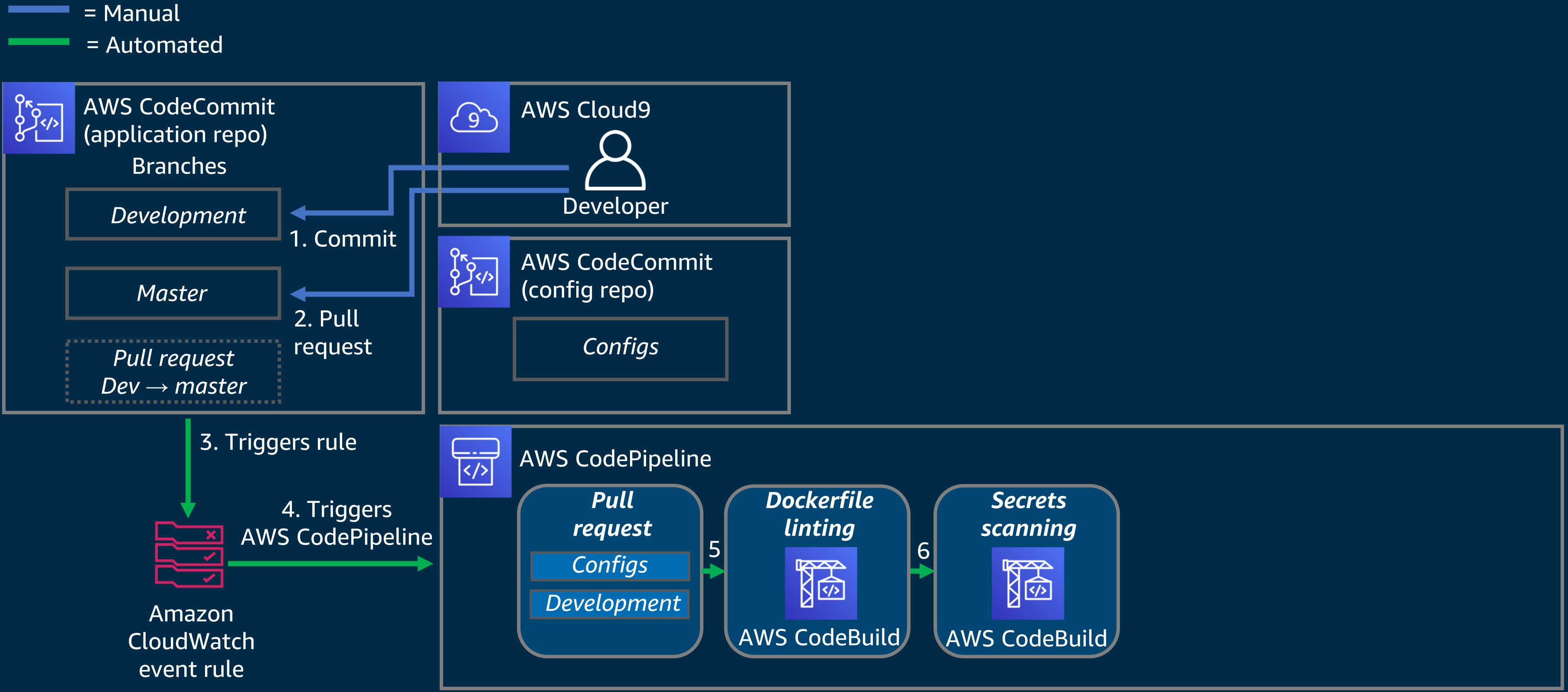
Let’s wrap up

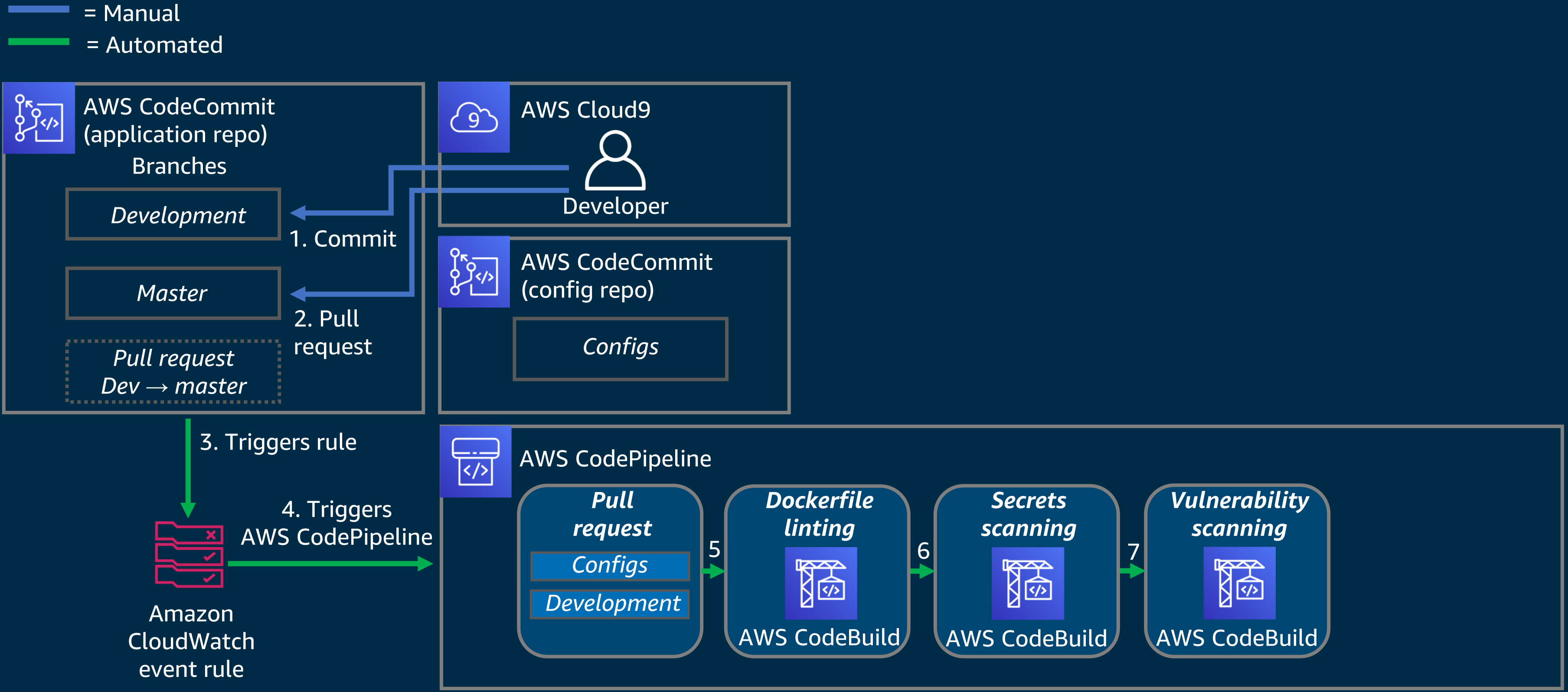
AWS RE:INFORCE

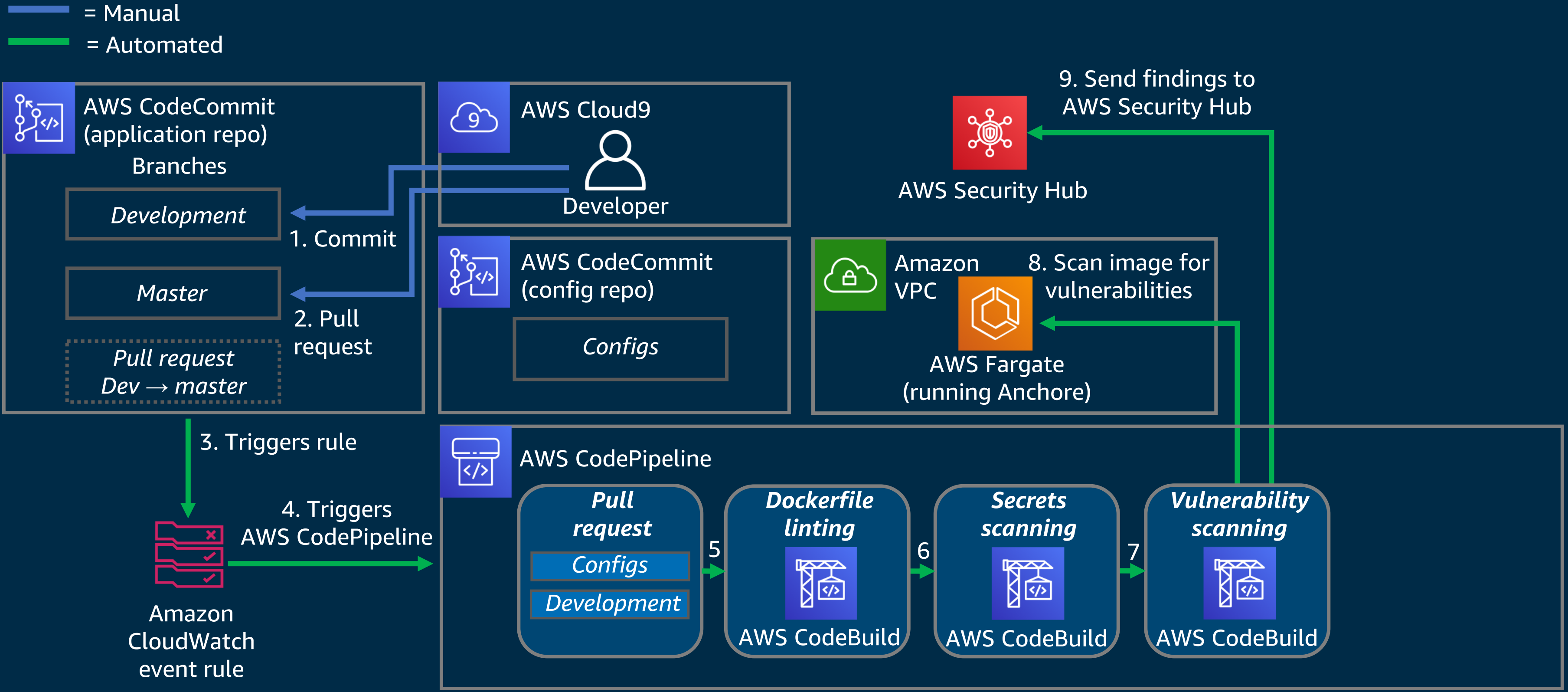


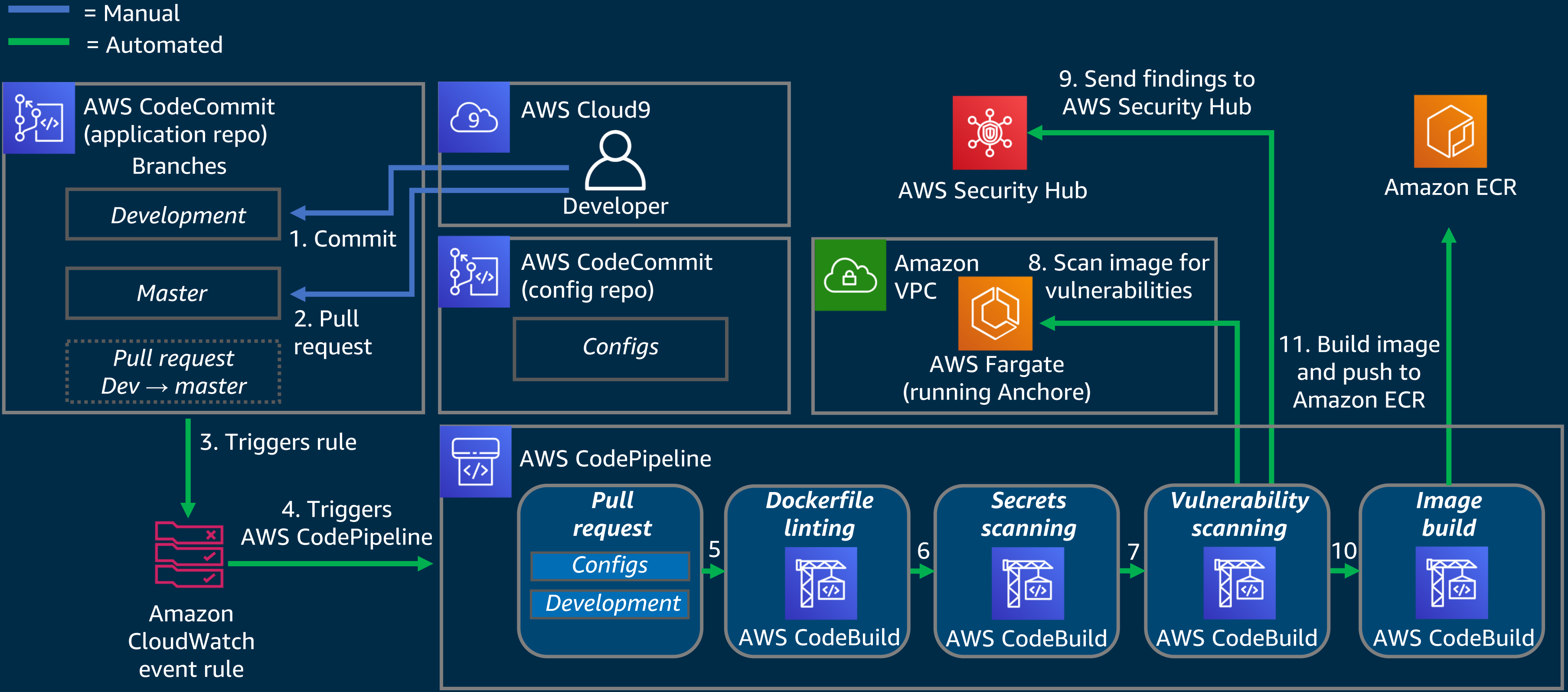
= Manual
= Automated
AWS CodeCommit
(application repo)
Branches
Development
Master
Pull request
Dev → master
1. Commit
2. Pull
request
AWS Cloud9
Developer
AWS CodeCommit
(config repo)
Configs


= Manual
= Automated
AWS CodeCommit (application repo)
Branches
Development
Master
Pull request Dev → master
1. Commit
2. Pull request
AWS Cloud9
Developer
AWS CodeCommit (config repo)
Configs
3. Triggers rule
4. Triggers AWS CodePipeline
Amazon CloudWatch event rule
AWS CodePipeline
Pull request
Configs
Development
5
Dockerfile linting
AWS CodeBuild

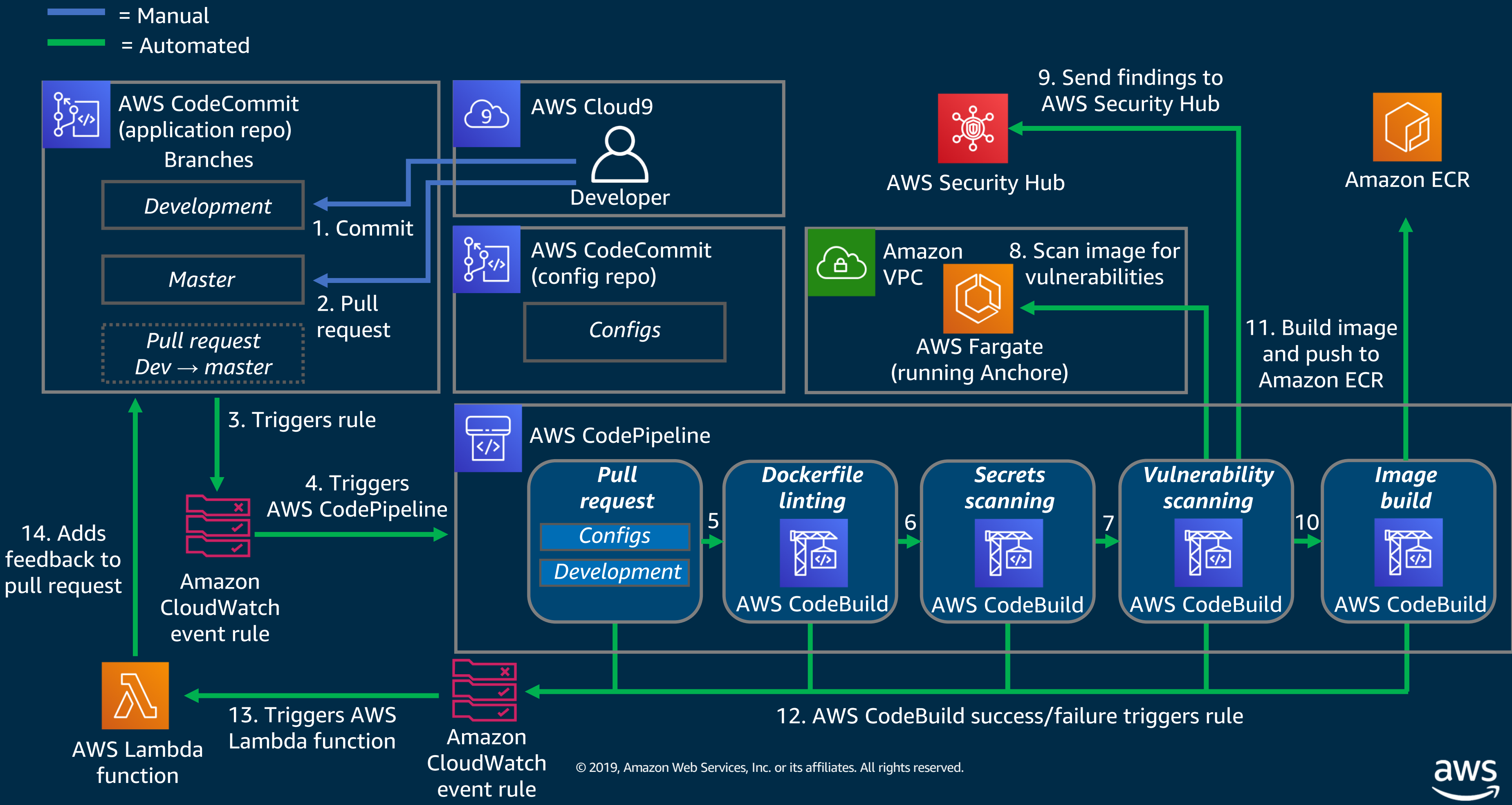

= Manual
= Automated
AWS CodeCommit
(application repo)
Branches
Development
Master
Pull request
Dev → master
1. Commit
2. Pull
request
AWS Cloud9
Developer
AWS CodeCommit
(config repo)
Configs
3. Triggers rule
4. Triggers
AWS CodePipeline
Amazon
CloudWatch
event rule
AWS CodePipeline
Pull
request
Configs
Development
5
Dockerfile
linting
AWS CodeBuild
6
Secrets
scanning
AWS CodeBuild


= Manual
= Automated
AWS CodeCommit (application repo)
Branches
Development
Master
Pull request Dev → master
1. Commit
2. Pull request
AWS Cloud9
Developer
AWS CodeCommit (config repo)
Configs
3. Triggers rule
Amazon CloudWatch event rule
4. Triggers AWS CodePipeline
AWS CodePipeline
Pull request
Configs
Development
5
Dockerfile linting
AWS CodeBuild
6
Secrets scanning
AWS CodeBuild
7
Vulnerability scanning
AWS CodeBuild


= Manual
= Automated
AWS CodeCommit (application repo)
Branches
Development
Master
Pull request Dev → master
1. Commit
2. Pull request
AWS Cloud9
Developer
AWS CodeCommit (config repo)
Configs
9. Send findings to AWS Security Hub
AWS Security Hub
Amazon VPC
8. Scan image for vulnerabilities
AWS Fargate (running Anchore)
3. Triggers rule
4. Triggers AWS CodePipeline
Amazon CloudWatch event rule
AWS CodePipeline
Pull request
Configs
Development
5
Dockerfile linting
AWS CodeBuild
6
Secrets scanning
AWS CodeBuild
7
Vulnerability scanning
AWS CodeBuild



= Manual
= Automated
AWS CodeCommit (application repo)
Branches
Development
Master
Pull request Dev → master
1. Commit
2. Pull request
AWS Cloud9
Developer
AWS CodeCommit (config repo)
Configs
Amazon VPC
8. Scan image for vulnerabilities
AWS Fargate (running Anchore)
9. Send findings to AWS Security Hub
AWS Security Hub
Amazon ECR
11. Build image and push to Amazon ECR
3. Triggers rule
Amazon CloudWatch event rule
4. Triggers AWS CodePipeline
AWS CodePipeline
Pull request
Configs
Development
5
Dockerfile linting
AWS CodeBuild
6
Secrets scanning
AWS CodeBuild
7
Vulnerability scanning
AWS CodeBuild
10
Image build
AWS CodeBuild


= Manual
= Automated
AWS CodeCommit (application repo)
Branches
Development
Master
Pull request Dev → master
AWS Cloud9
Developer
1. Commit
2. Pull request
AWS CodeCommit (config repo)
Configs
Amazon VPC
8. Scan image for vulnerabilities
AWS Fargate (running Anchore)
9. Send findings to AWS Security Hub
AWS Security Hub
Amazon ECR
11. Build image and push to Amazon ECR
3. Triggers rule
4. Triggers AWS CodePipeline
Amazon CloudWatch event rule
AWS CodePipeline
Pull request
Configs
Development
5
Dockerfile linting
AWS CodeBuild
6
Secrets scanning
AWS CodeBuild
7
Vulnerability scanning
AWS CodeBuild
10
Image build
AWS CodeBuild
14. Adds feedback to pull request
AWS Lambda function
13. Triggers AWS Lambda function
Amazon CloudWatch event rule
12. AWS CodeBuild success/failure triggers rule

Thank you!

aws RE:INFORCE

!

Please complete the session survey in the mobile app.

aws RE:INFORCE