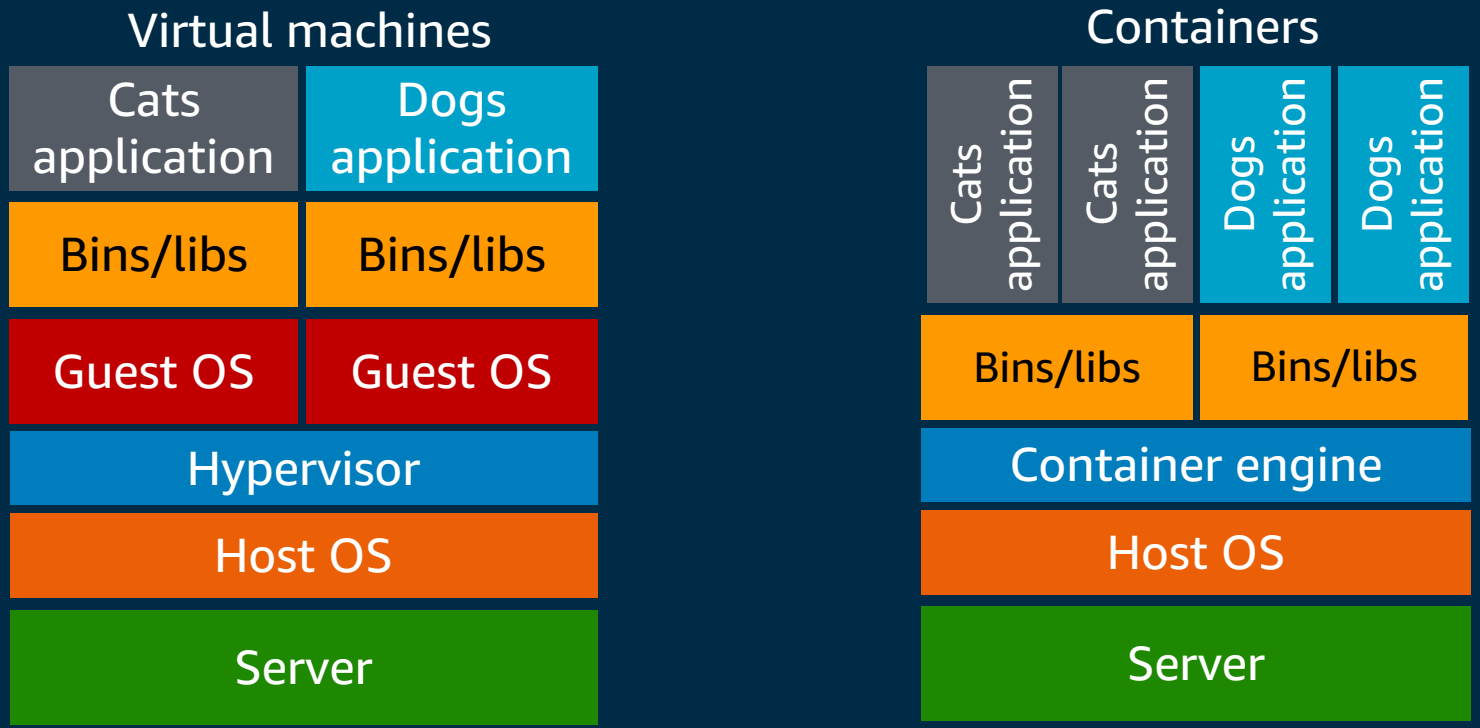
RE:INFORCE

Goals

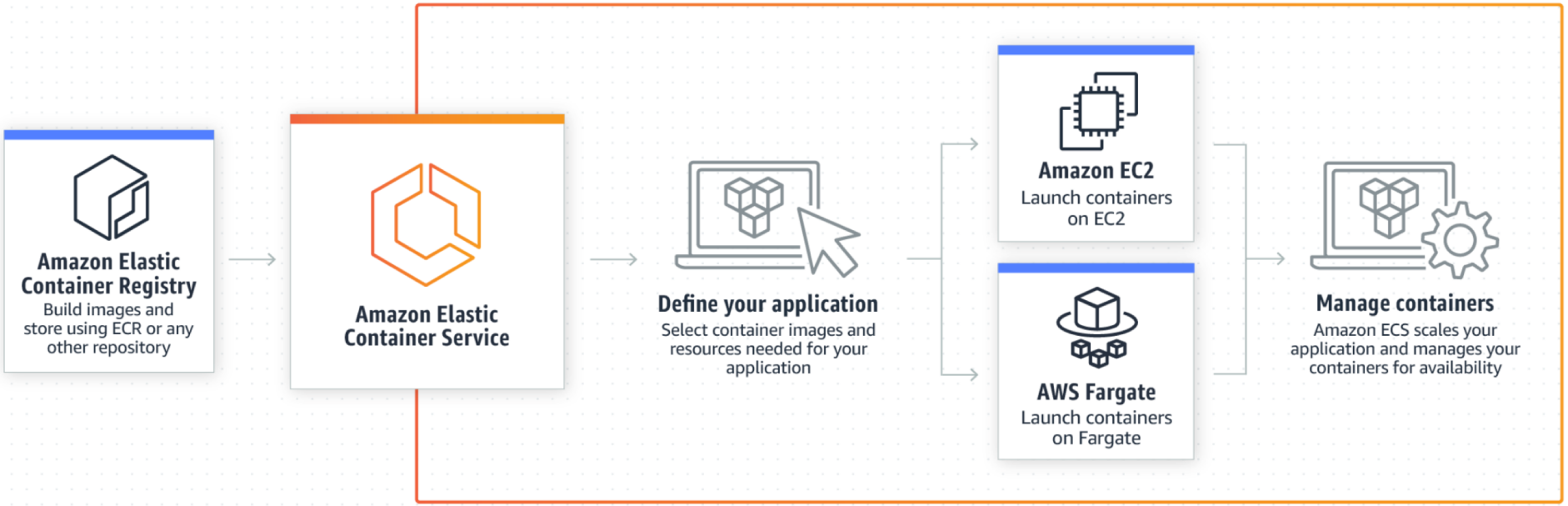
- Learn about container security using DevSecOps
- Learn about open-source container security tools and standards
- Learn about AWS development tools and DevOps services
- *Have fun while you're at it!*

Why is container security different?

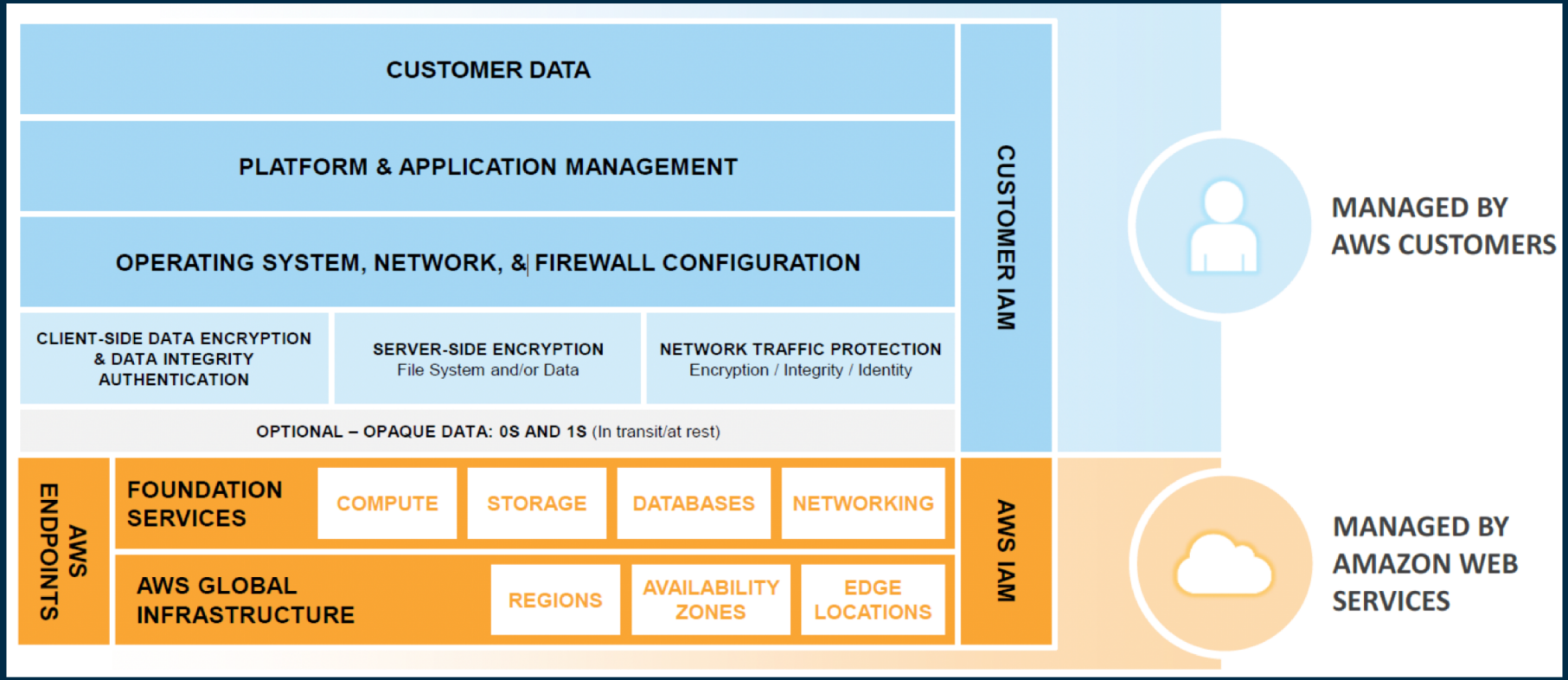


aws

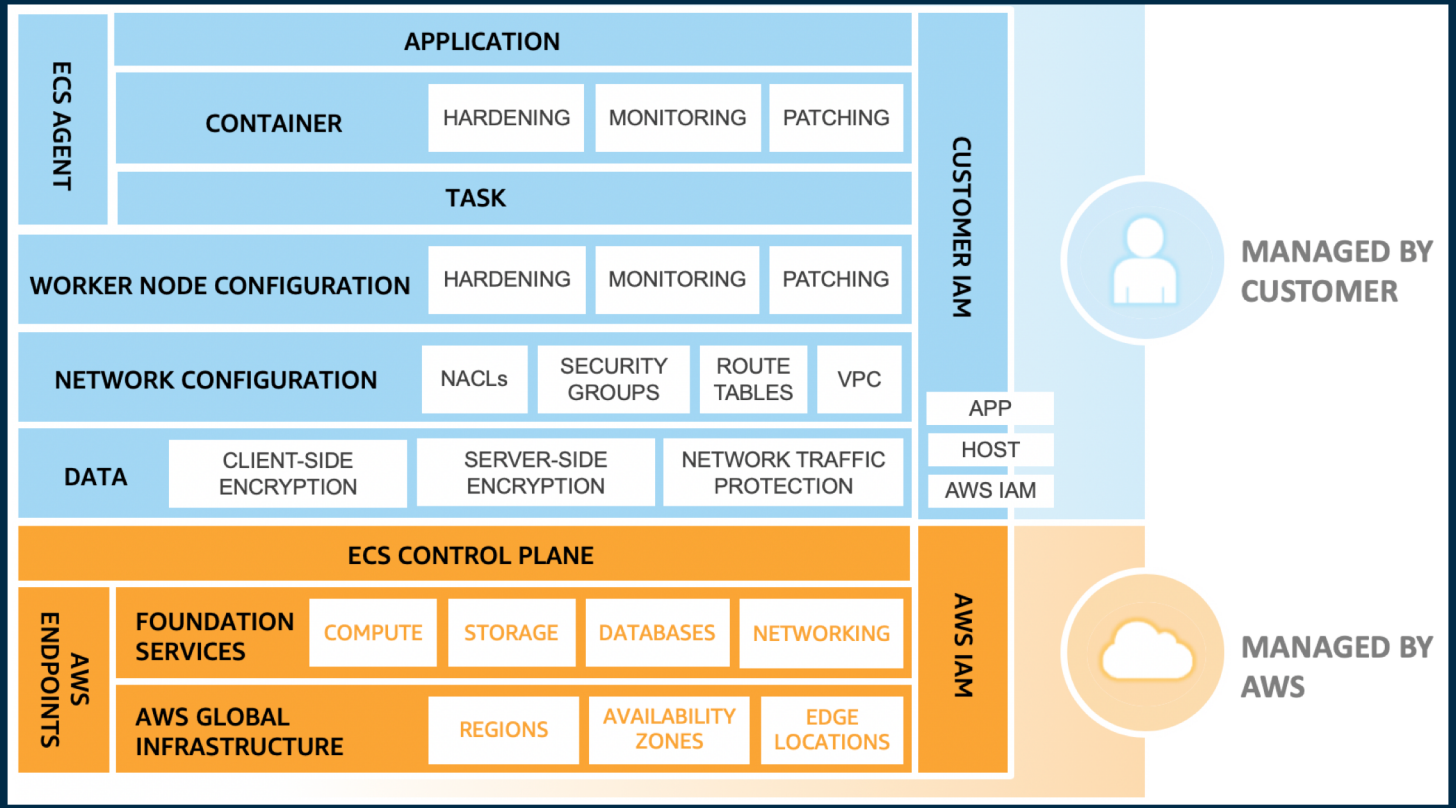
Containers on AWS



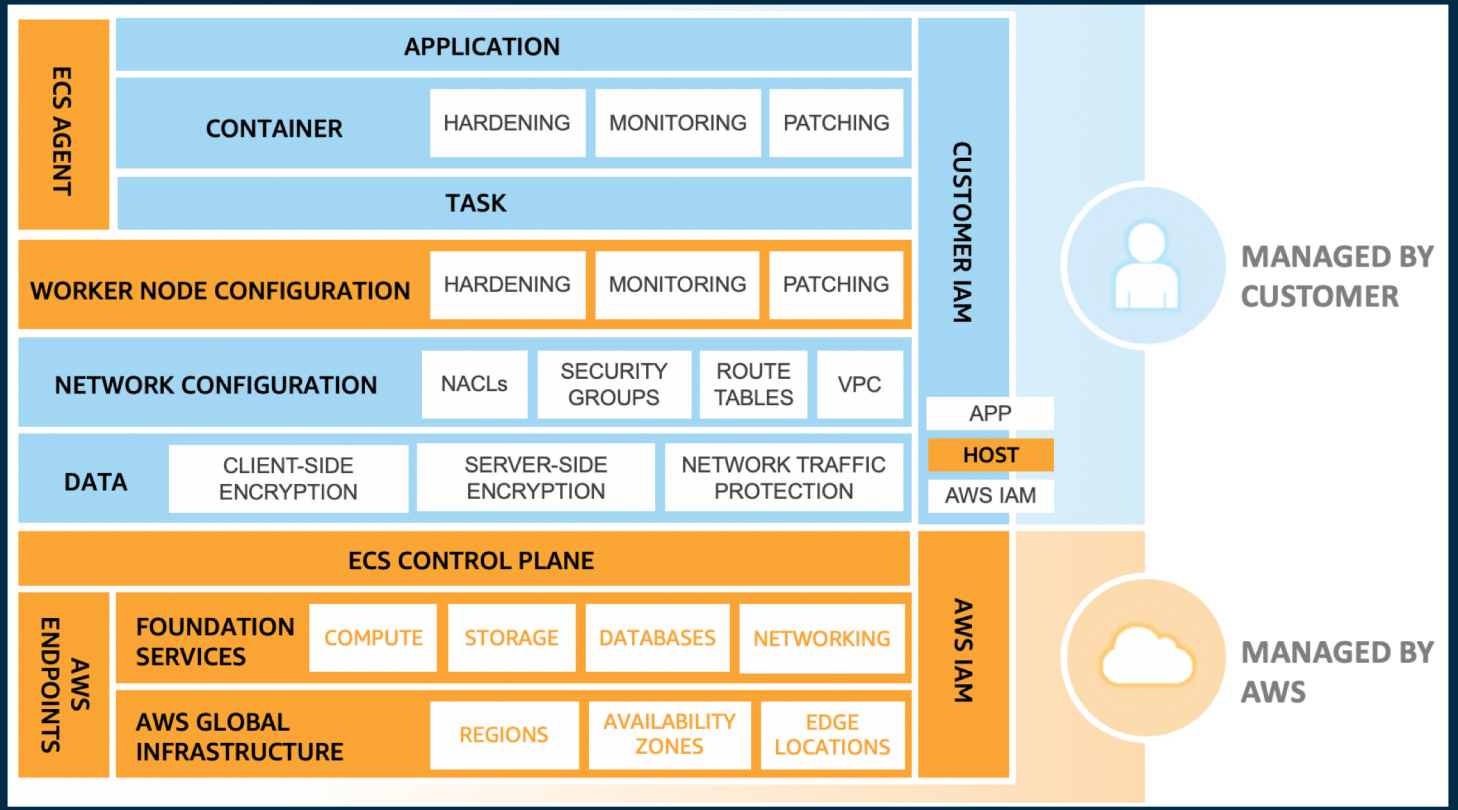
AWS shared responsibility model



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!

aws

Container security threats

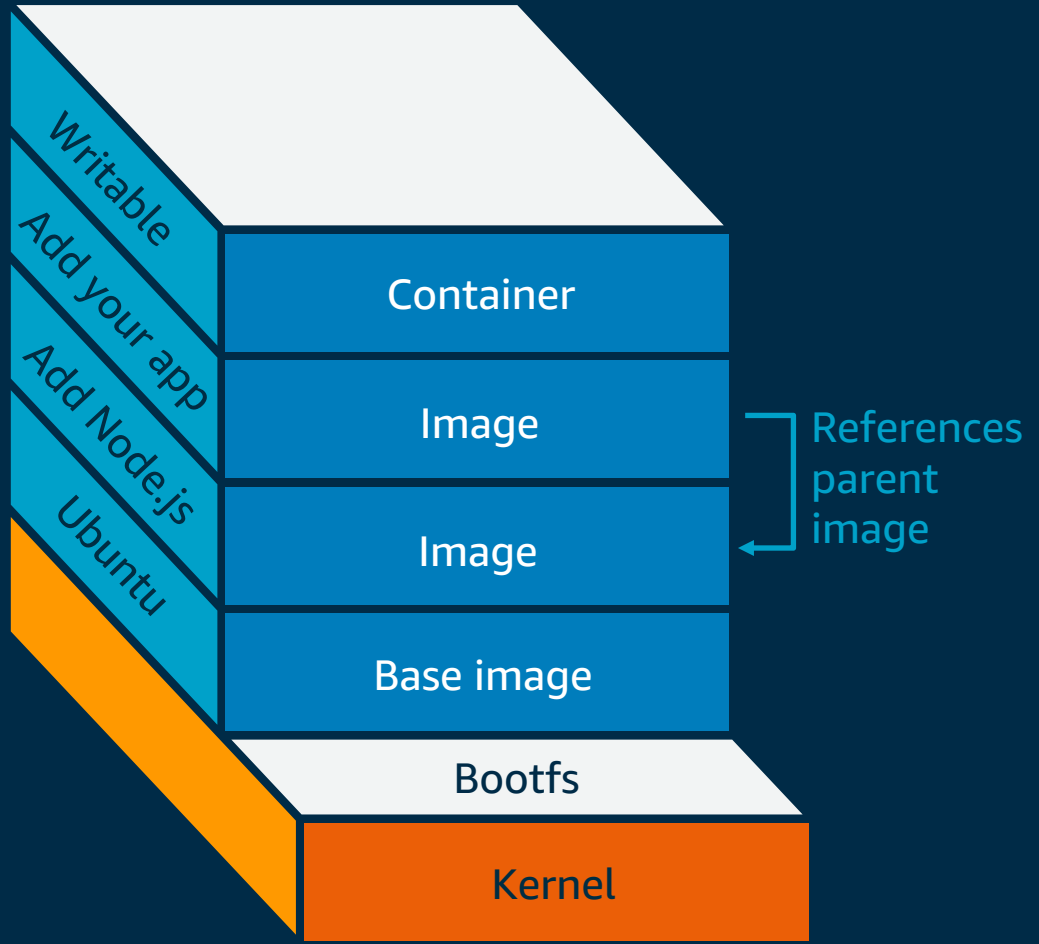
- Host security
- Image security
- Denial of service
- Credentials and secrets
- Container breakouts
- Runtime security

Container security threats

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

Security best practices for container images

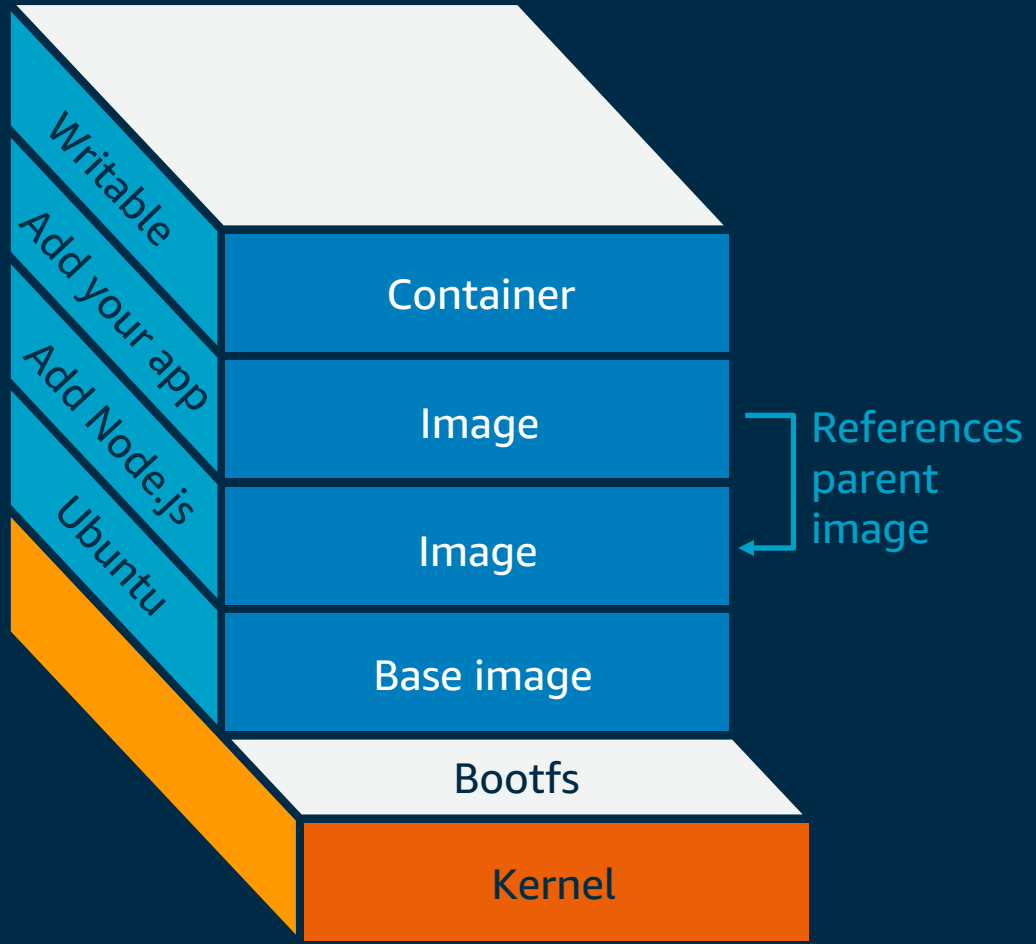
- **Less is more (secure)**
- No secrets in them
- One service per container
- Minimize container footprint
- Include only what is **needed** at runtime



aws

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



aws

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

AWS CodeCommit

```
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 CodeBuild

DevSecOps container pipeline

Developers **Sec**urity engineers **Ops** engineers

AWS CodeCommit

```
{
  "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 CodeBuild

aws

DevSecOps container pipeline

Developers **Sec**urity engineers **Ops** engineers

AWS CodeCommit

```
➢ 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"
```

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

Task definition

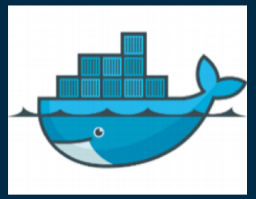
```
FROM centos:centos7
MAINTAINER cb@demo.com
RUN yum -y update
RUN yum -y install openssh-
server U
SER sshduser
EXPOSE 5432
ENTRYPOINT sshd
```

Dockerfile

AWS CodeBuild

Validate configuration > Merge > Scan for secrets > Merge >

Docker image



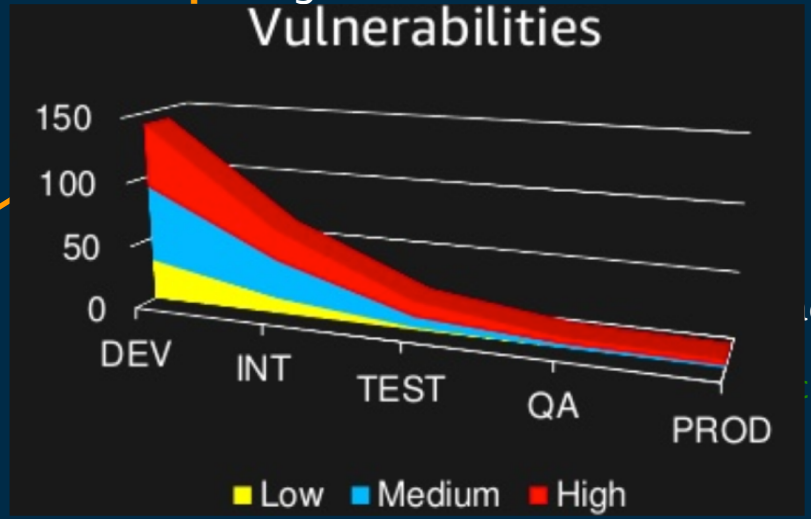
aws

DevSecOps container pipeline

Developers

Security engineers

Ops engineers



Amazon EC2 container registry

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

Task definition

AWS CodeCommit

Docker image

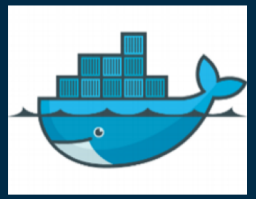
```
FROM centos:centos7
MAINTAINER cb@demo.com
RUN yum -y update
RUN yum -y install openssh-
server U
SER sshduser
EXPOSE 5432
ENTRYPOINT sshd
```

Dockerfile

AWS CodeBuild

Validate configuration > Merge > Scan for secrets > Merge >

Scan Docker image > Publish >



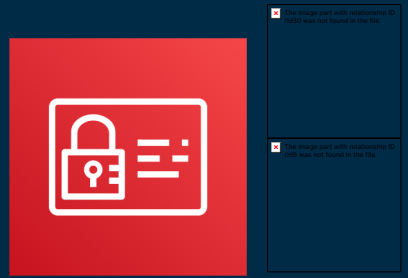
aws

Credentials and secrets

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

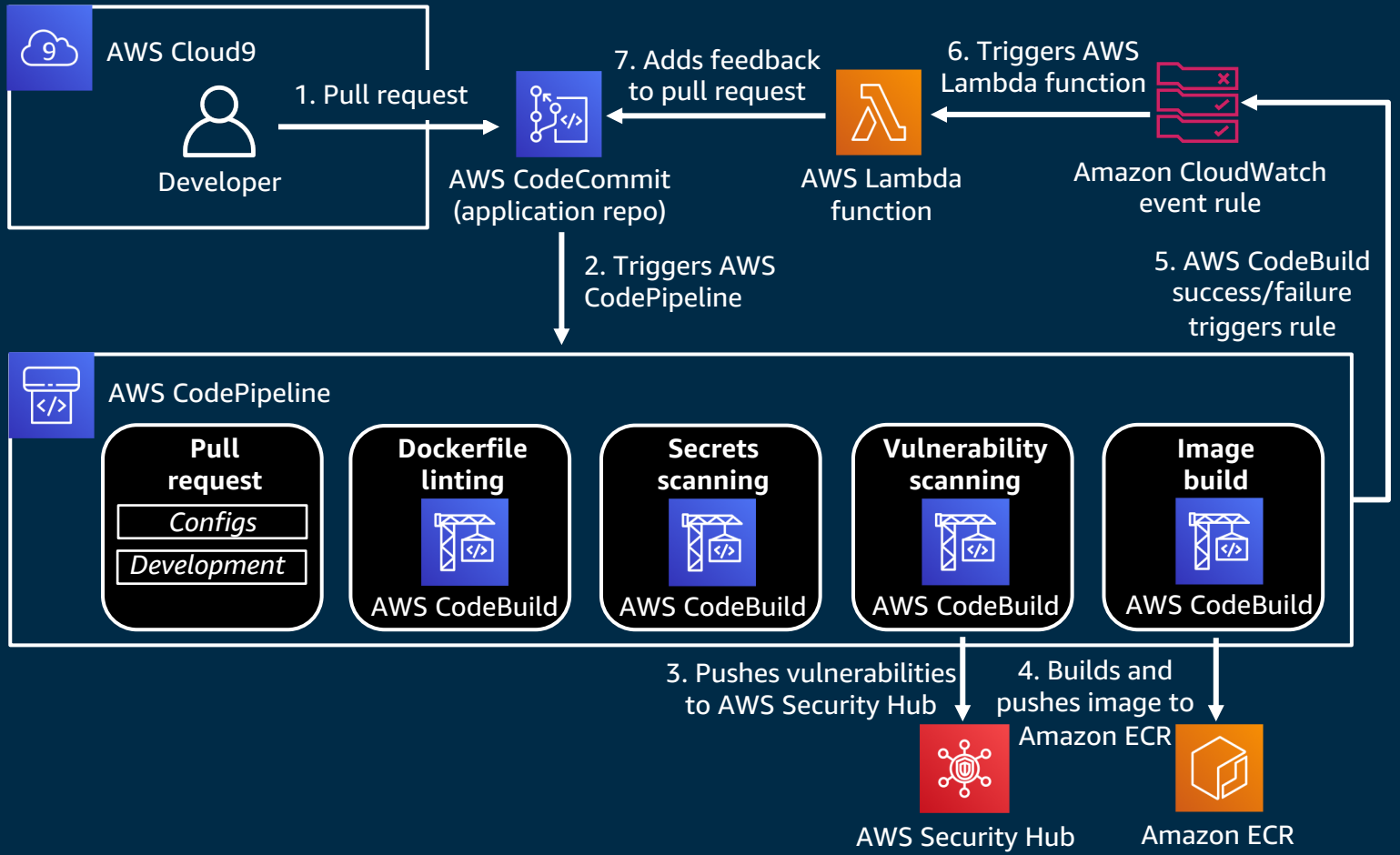
They are integrated into Amazon ECS, but you need to call them within the pod on Kubernetes via AWS CLI or SDK

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



Workshop architecture: From 10,000 feet

aws RE:INFORCE



Let’s build and have fun!

aws RE:INFORCE

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

aws

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

aws

Integrating security testing into your container build pipeline: Module 3

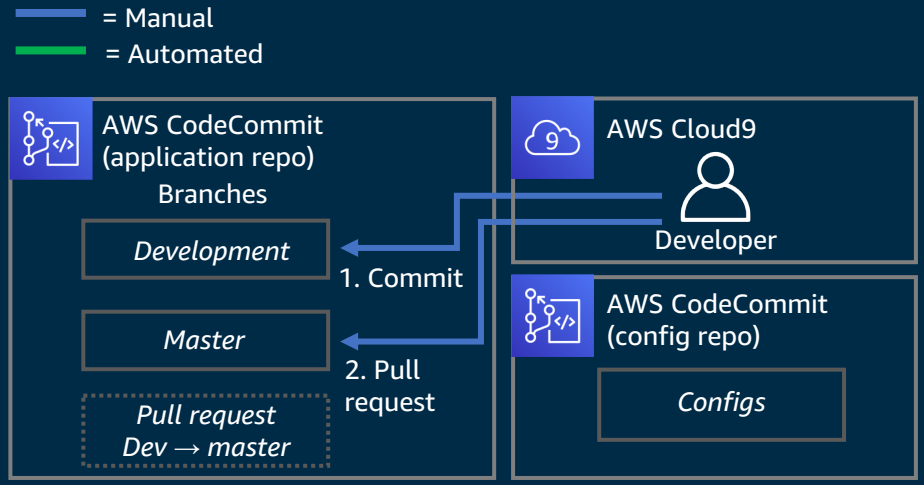
- Start with https://container-devsecops.awssecworkshops.com
- ~~Module 1: Dockerfile linting~~
- ~~Module 2: Secrets scanning~~
- Module 3: Vulnerability scanning (15 mins)
  - Create buildspec file
  - Add command to run Anchore
- Module 4: Pipeline testing

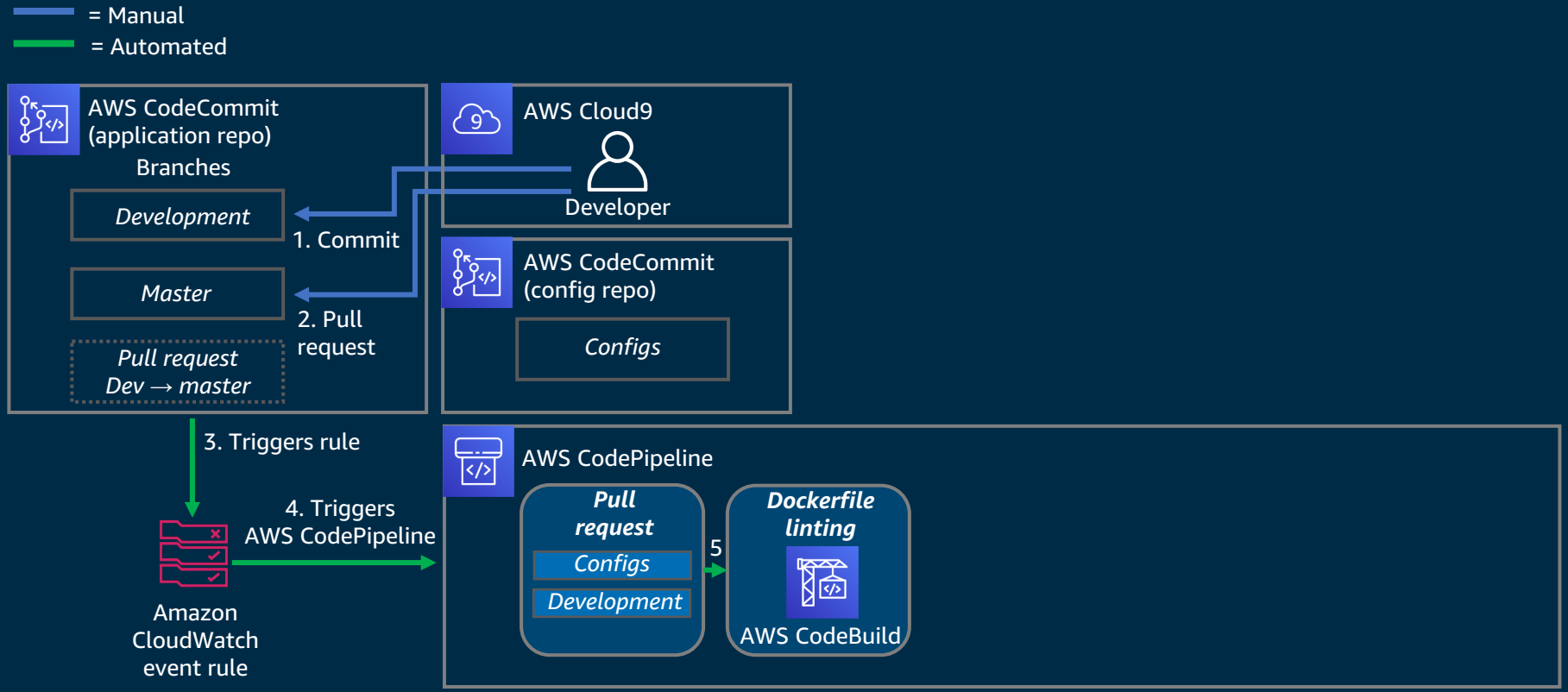
Integrating security testing into your container build pipeline: Module 4

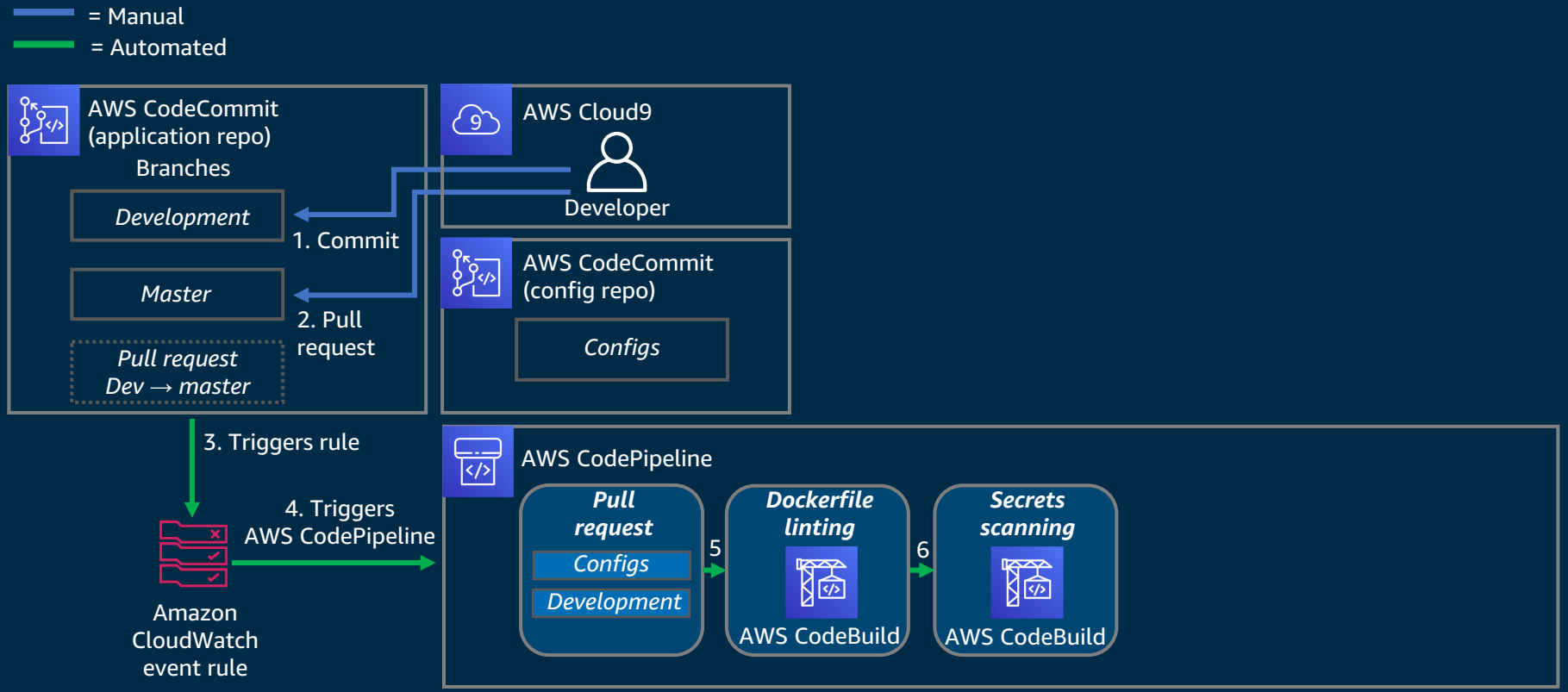
- Start with https://container-devsecops.awssecworkshops.com
- ~~Module 1: Dockerfile linting~~
- ~~Module 2: Secrets scanning~~
- ~~Module 3: Vulnerability scanning~~
- Module 4: Pipeline testing (15 mins)
  - Make a commit
  - View feedback loop

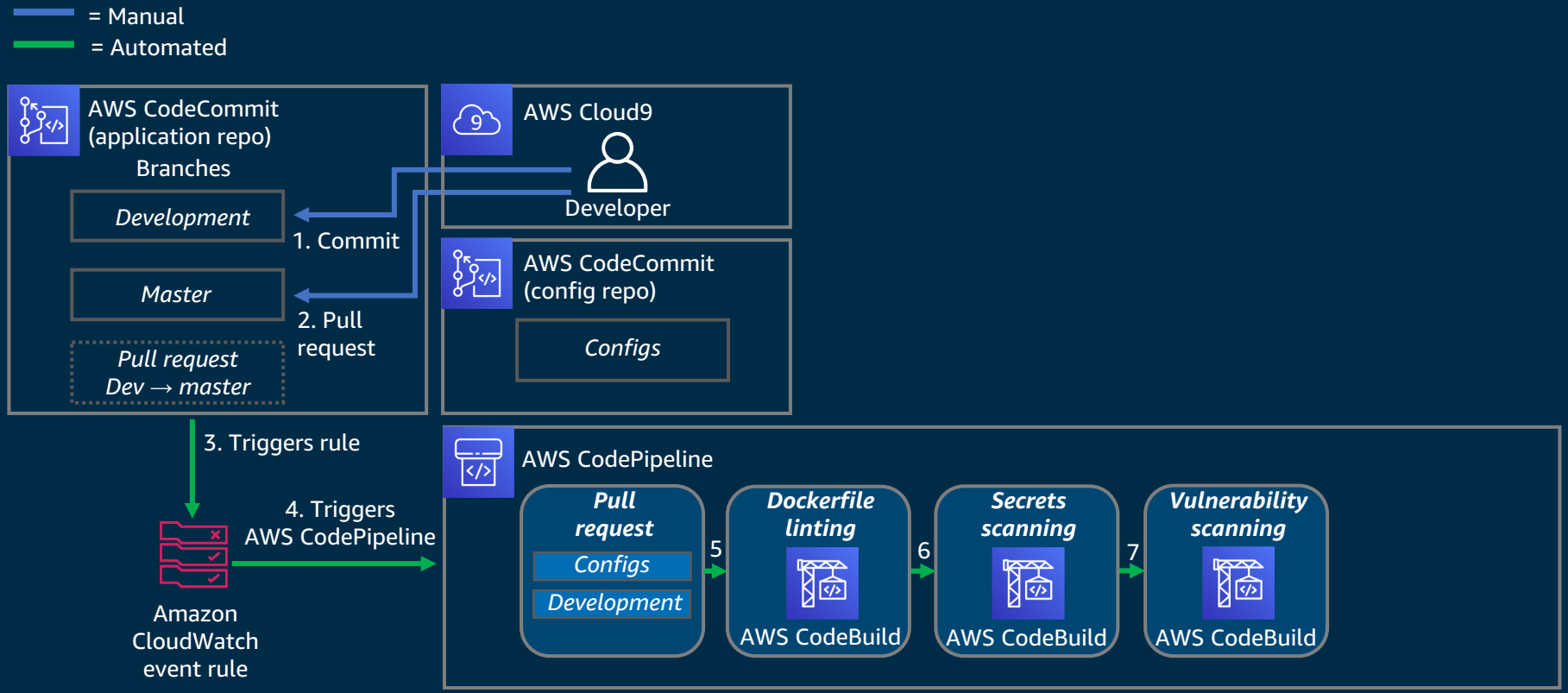
Let’s wrap up

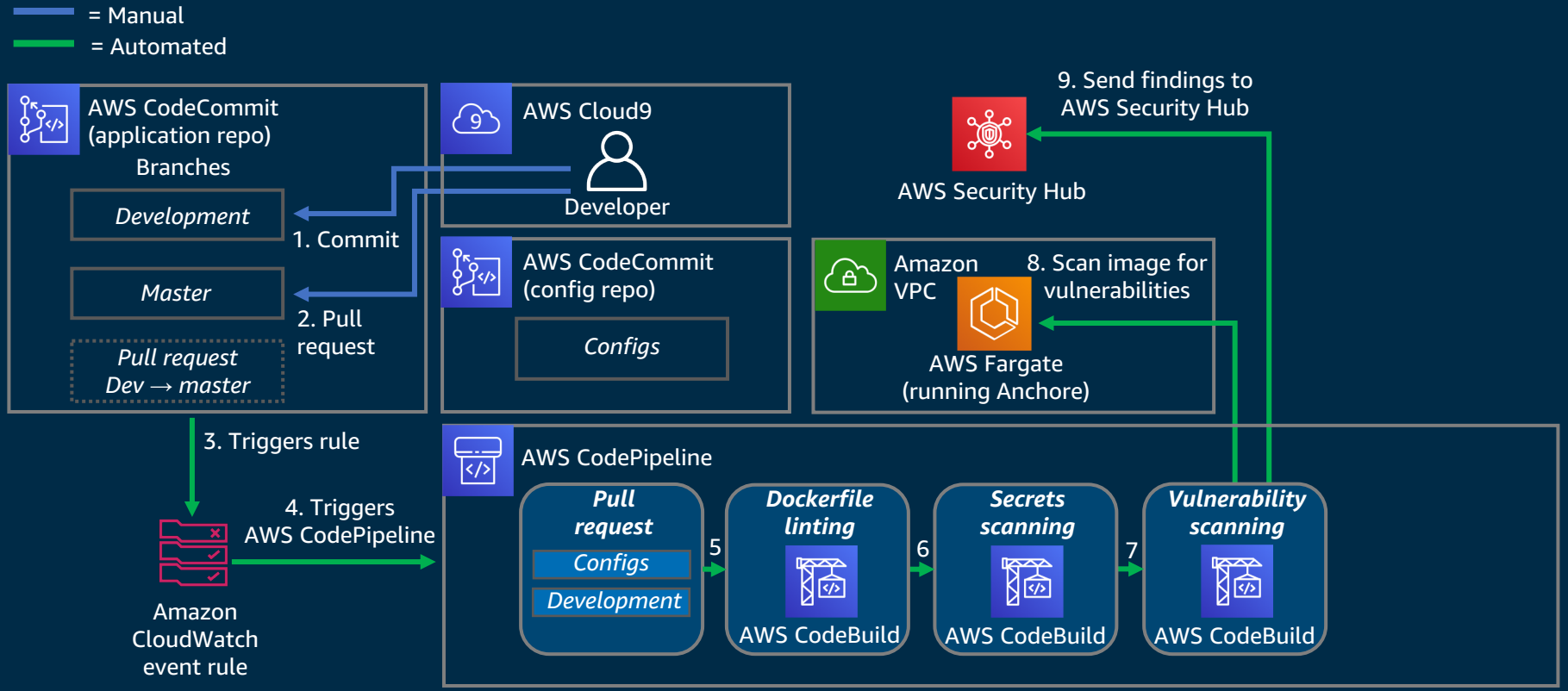
aws RE:INFORCE

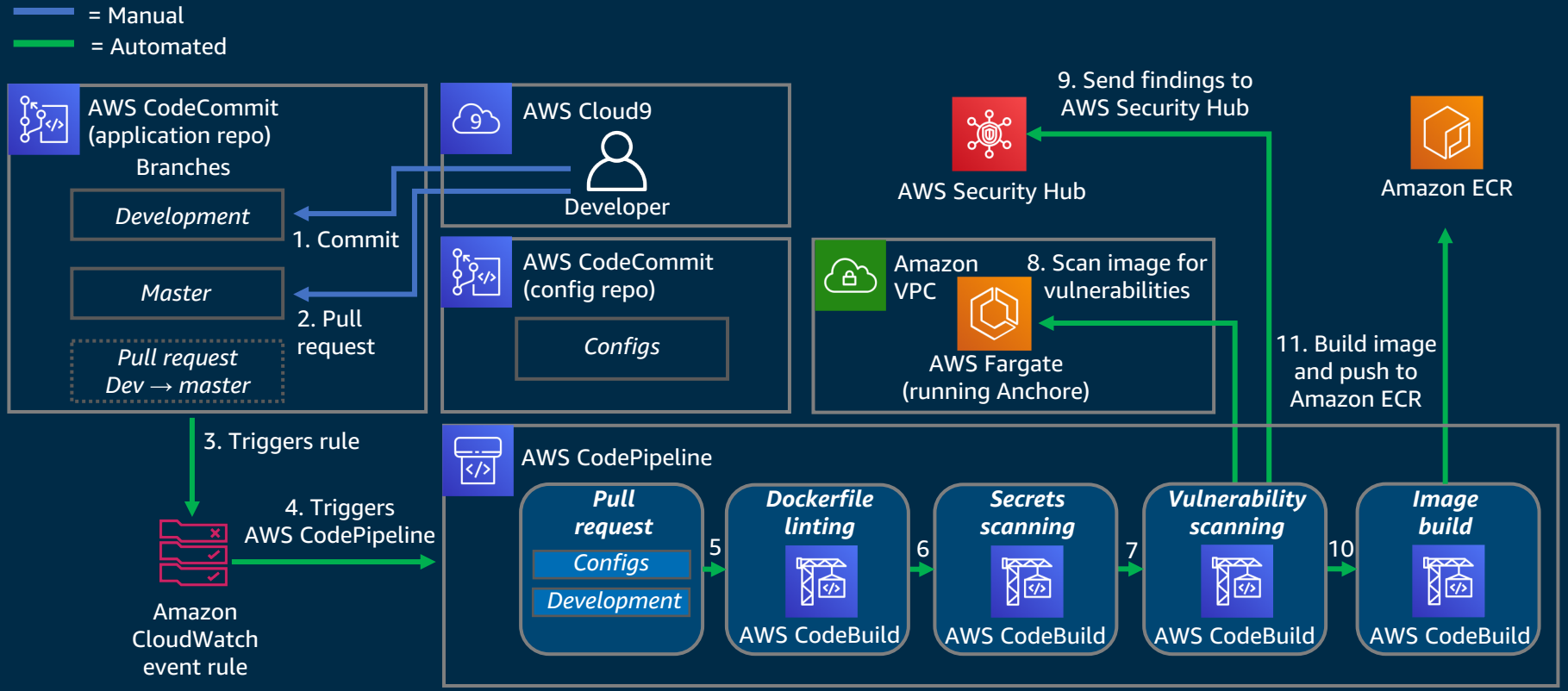


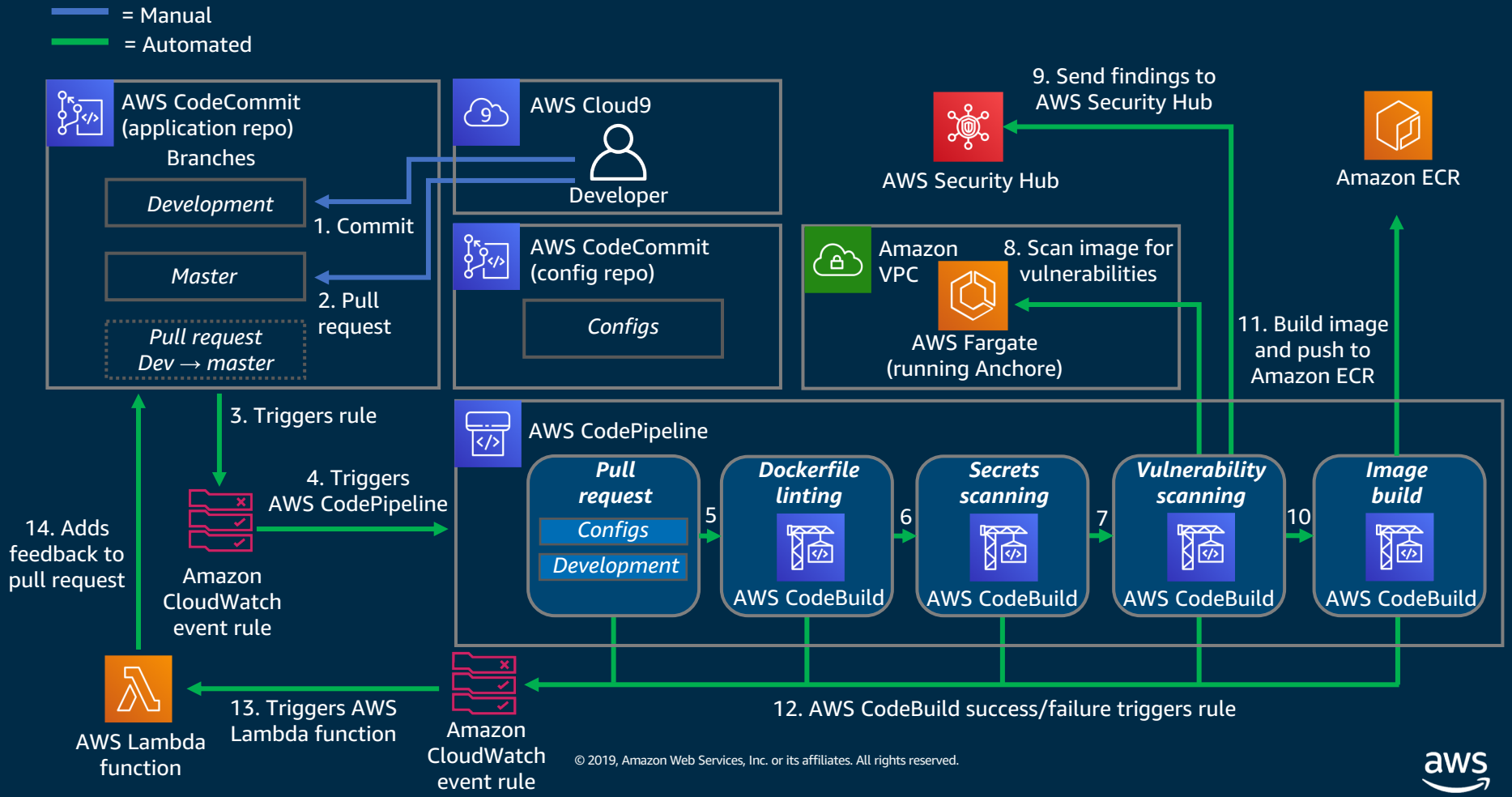












Thank you!

aws RE:INFORCE