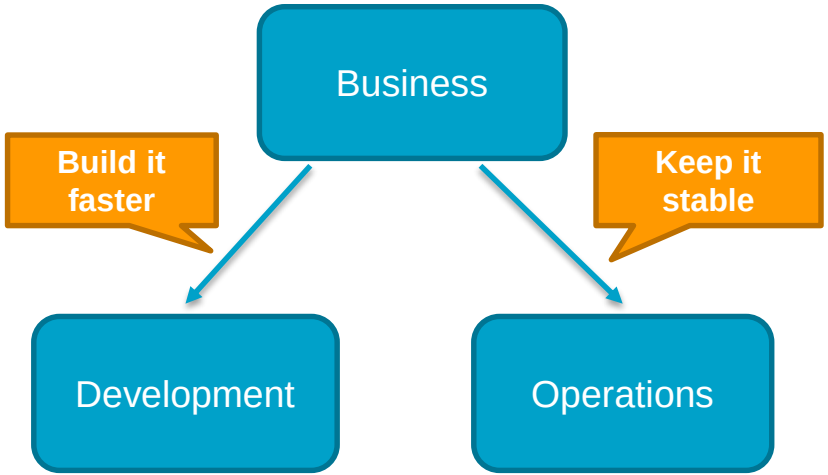
DevOps on AWS

Agenda

This section covers the following topics:

- What is DevOps?
- AWS Platform – CodeDeploy Services, Cloudformation etc.
- CI/CD for container on AWS – Architecture and demo

Why are we here today? Competing Forces



Change is the root cause of most outages

Two options:

(1) Make it a big deal

(2) Small iterative non-events

What is DevOps?

Cultural philosophies

Practices

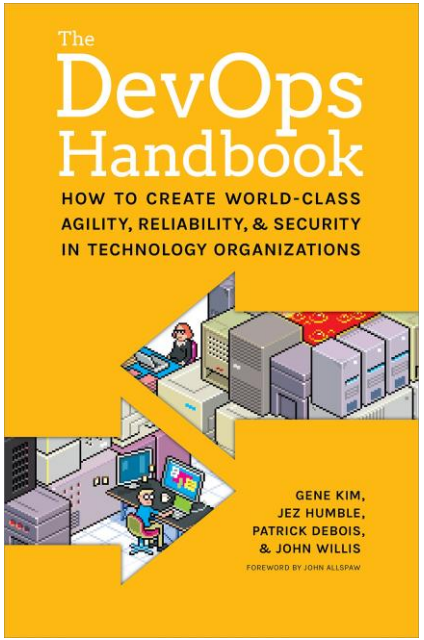
Tools

Shared responsibility

Ownership

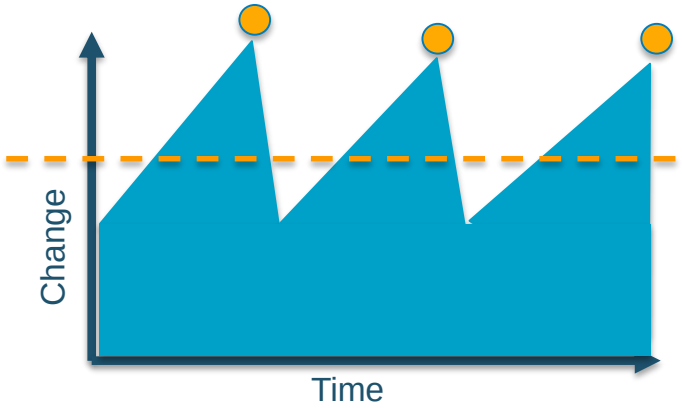
Visibility and communication

Faster release and deployment cycle!!!



Deploying more frequently lowers risk

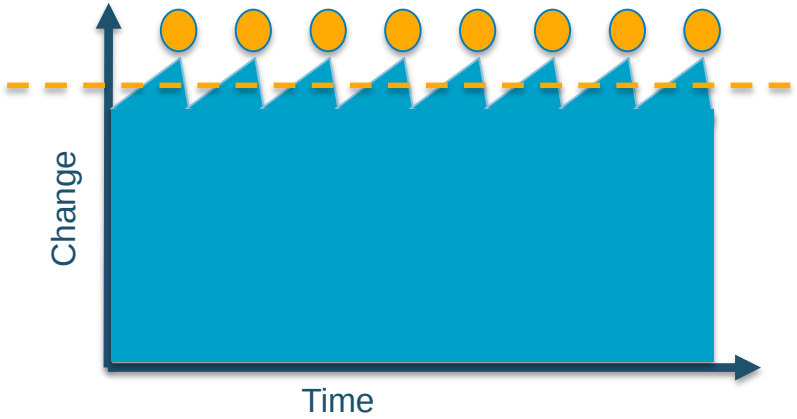
Rare Release Events: “Waterfall Methodology”



Larger Effort

“Increased Risk”

Frequent Release Events: “Agile Methodology”



Smaller Effort

“Minimized Risk”

Building a DevOps culture

Enable DevOps change across:

Culture

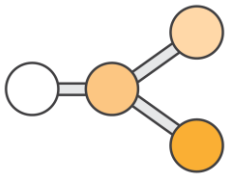
Technology ← Our Focus Today

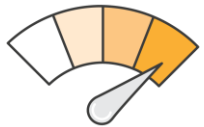
Management

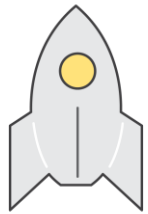
Implement across all tracks in parallel, if possible

Release process have for major phases

Source	Build	Test	Deploy
• 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







Enabling common DevOps practices

Infrastructure as code

IT automation

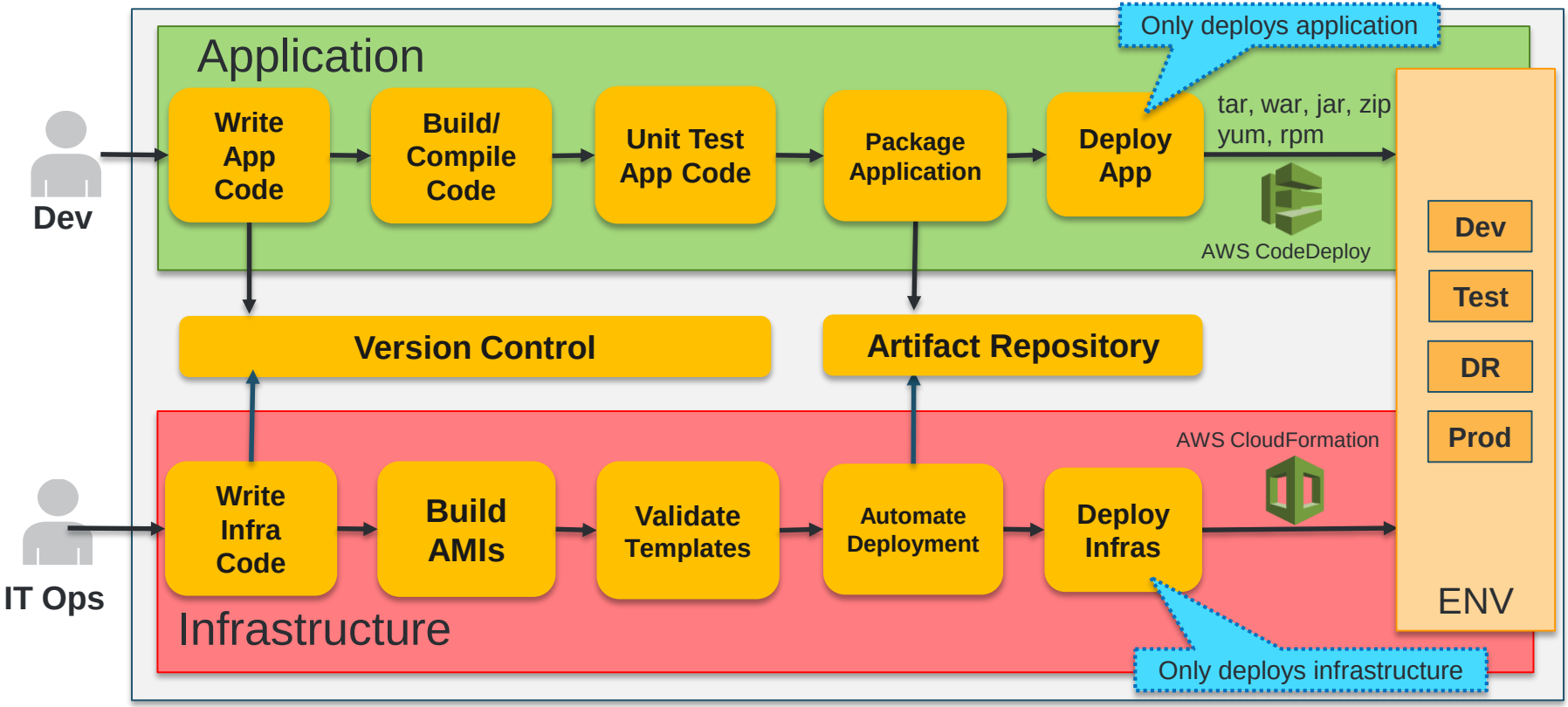
Continuous integration

Continuous deployment

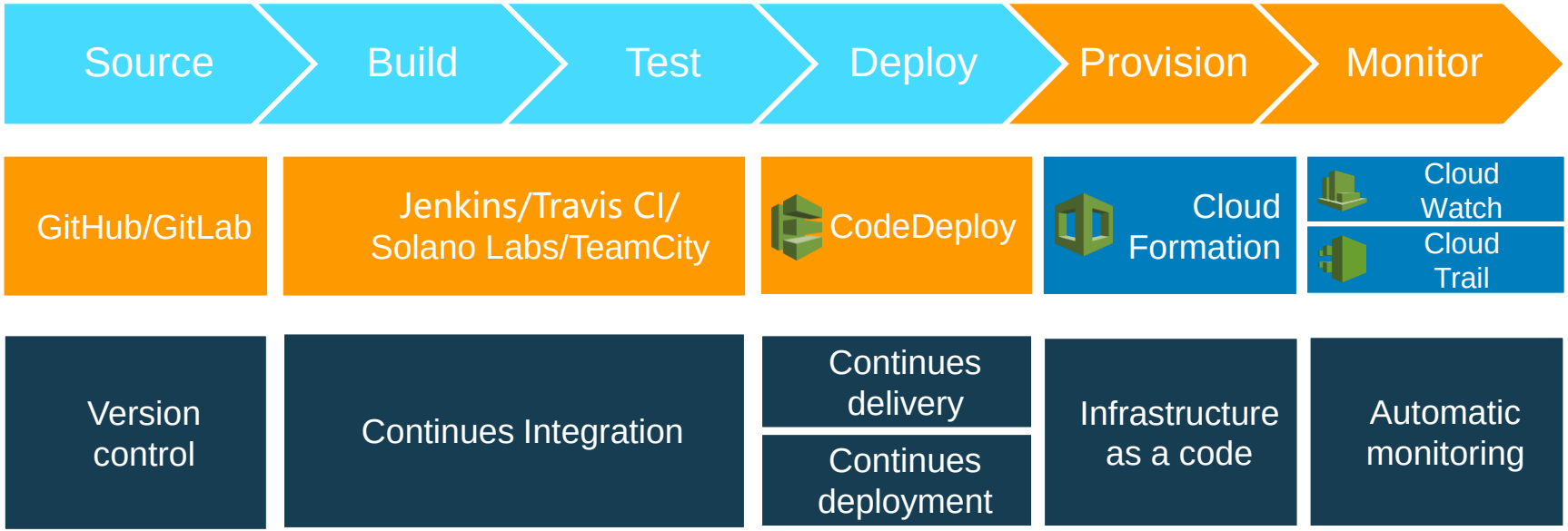
Integrated version control (application and infrastructure)

Monitoring and logging

Continuous Integration / Deployment & Automation

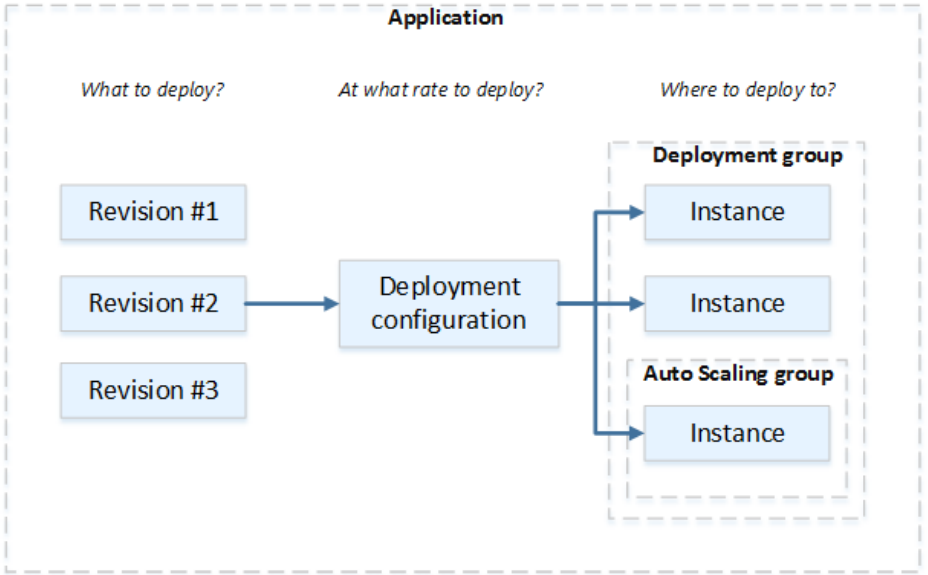


Implementing DevOps on AWS



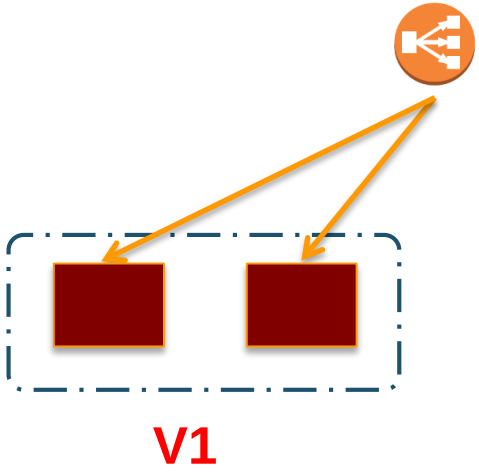
AWS CodeDeploy

- Automated application deployments to
  - EC2
  - On-premise VM
- Handle the complexity of updating your applications
- Supports rollback and redeployment
- Avoid downtime during application deployment
- Deployment type
  - In-place deployment
  - Blue-green deployment
- Integrates with third-party tools and AWS



Deployment Approaches – In-Place

- Deploy all at once (Service outage)

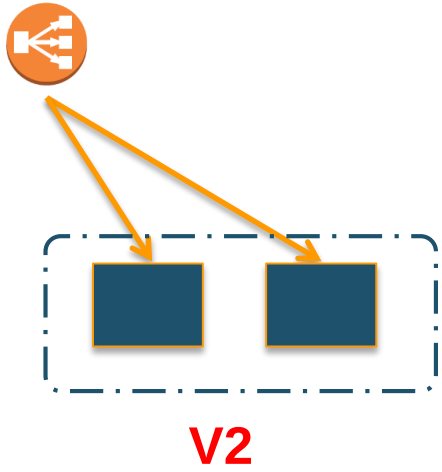


Deployment Approaches – In-Place

- Deploy all at once (Service outage)

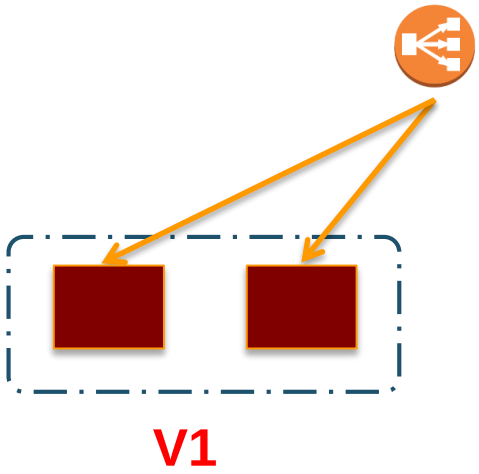
Deployment Approaches – In-Place

- Deploy all at once (Service outage)



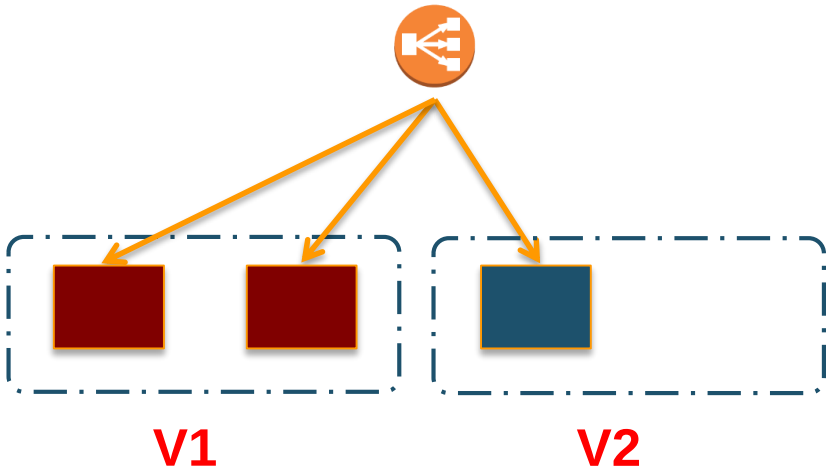
Deployment Approaches – Rolling Update

- Deploy one new version
- Run health checks
- If successful, terminate one old version



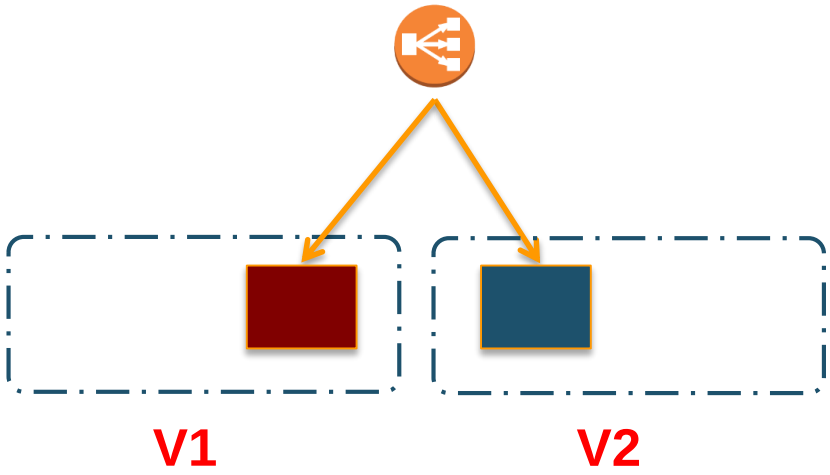
Deployment Approaches – Rolling Update

- Deploy one new version
- Run health checks
- If successful, terminate one old version



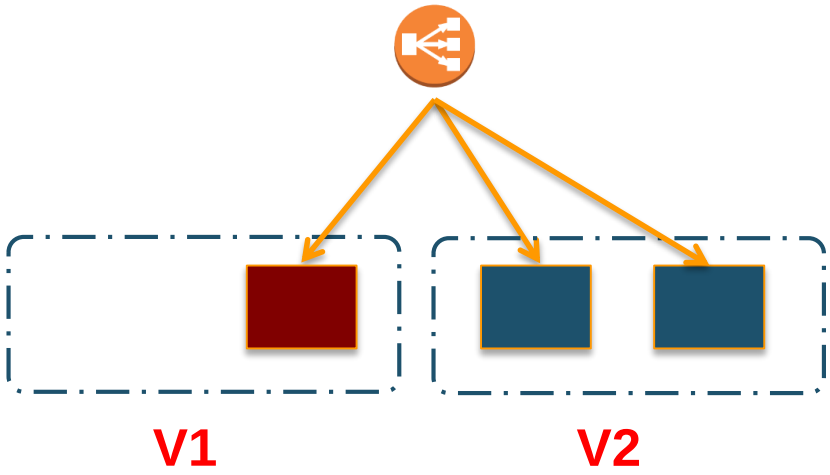
Deployment Approaches – Rolling Update

- Deploy one new version
- Run health checks
- If successful, terminate one old version



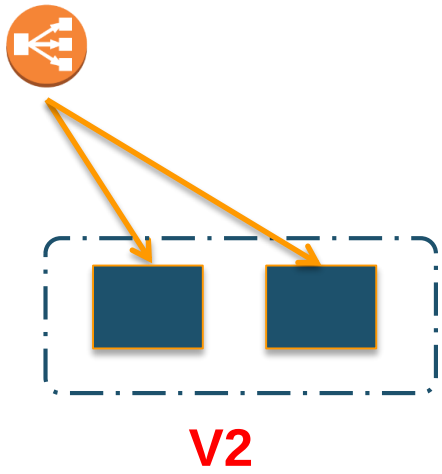
Deployment Approaches – Rolling Update

- Deploy one new version
- Run health checks
- If successful, terminate one old version



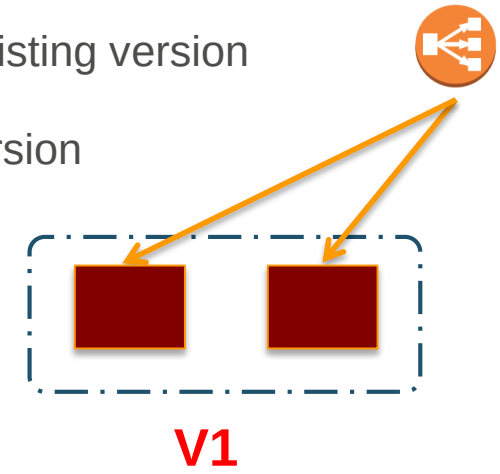
Deployment Approaches – Rolling Update

- Deploy one new version
- Run health checks
- If successful, terminate one old version



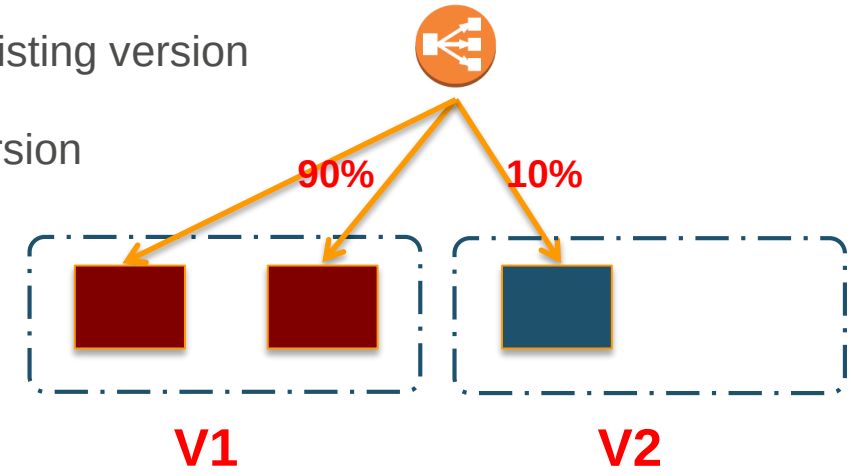
Deployment Approaches - Canary

- Deploy a new version while retaining existing version
- Redirect percentage of traffic to new version
- Once satisfied, roll out new version



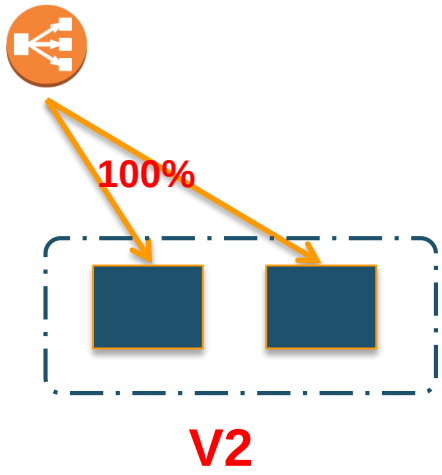
Deployment Approaches - Canary

- Deploy a new version while retaining existing version
- Redirect percentage of traffic to new version
- Once satisfied, roll out new version



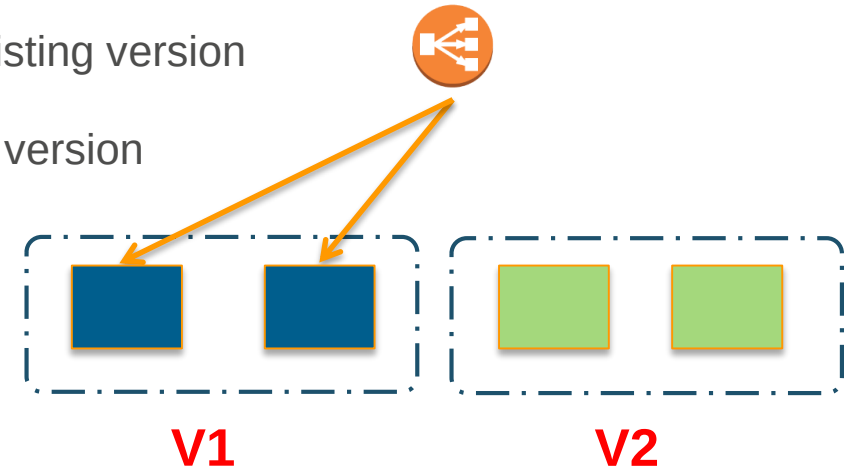
Deployment Approaches - Canary

- Deploy a new version while retaining existing version
- Redirect percentage of traffic to new version
- Once satisfied, roll out new version



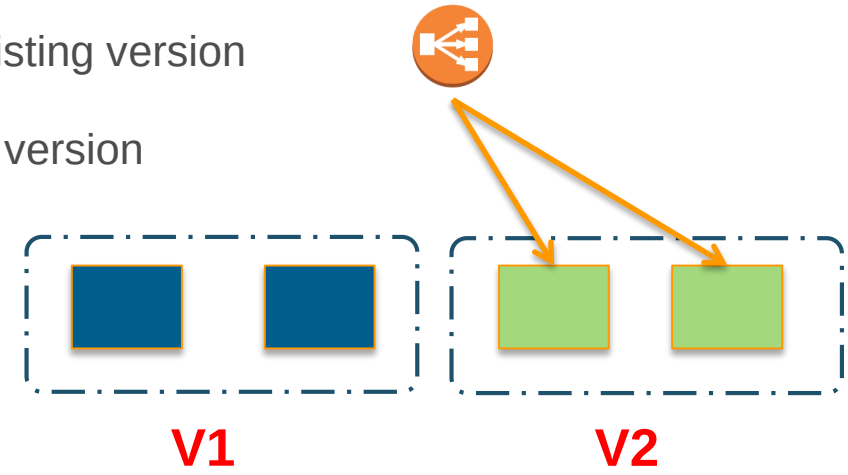
Deployment Approaches – Blue-Green

- Deploy a new version while retaining existing version
- Switch all traffic from old version to new version
- Rollback to old version if necessary

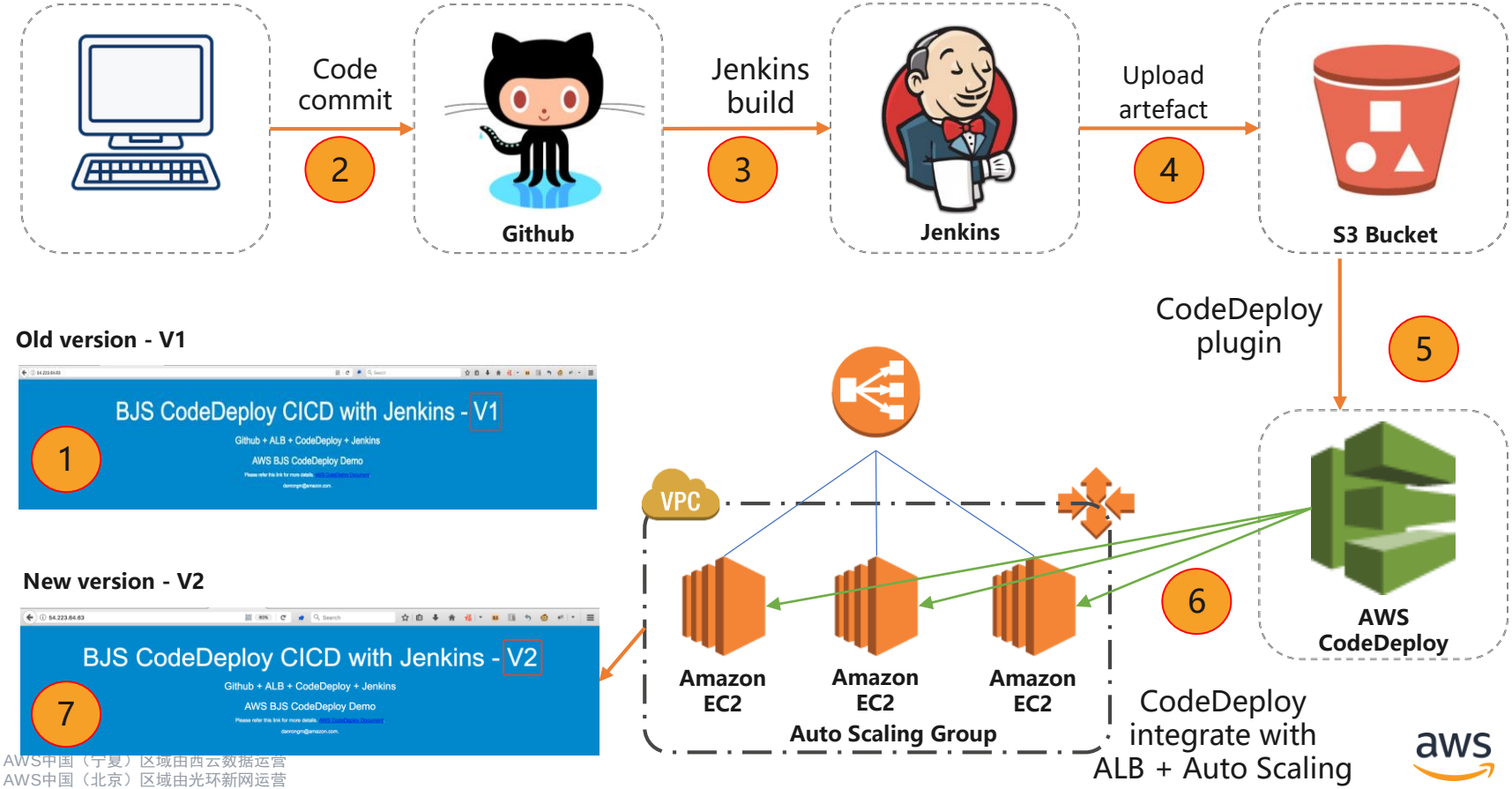


Deployment Approaches – Blue-Green

- Deploy a new version while retaining existing version
- Switch all traffic from old version to new version
- Rollback to old version if necessary



An Example of Github+Jenkins+CodeDeploy



CloudFormation – Components & Technology

Infrastructure as Code

JSON/YAML formatted file

Parameter definition

Resource creation

Configuration actions

Framework

Stack creation

Stack updates

Error detection and rollback

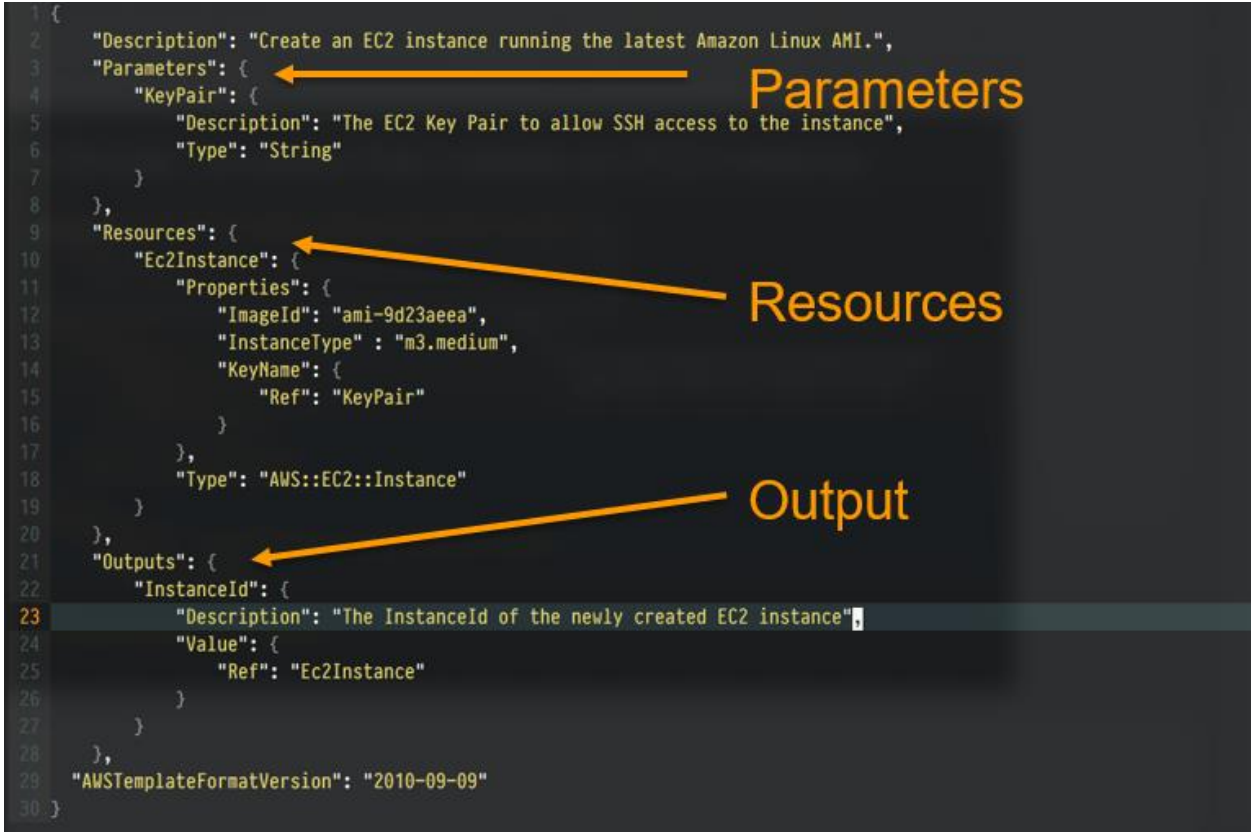
Configured AWS resources

Comprehensive service support

Service event aware

Customizable

CloudFormation – Example



CloudFormation Change Sets

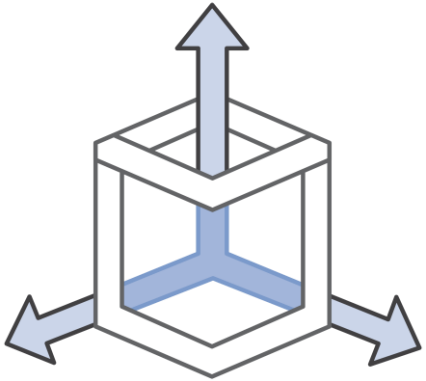
Diff before execute

Stage changes for analysis or execution by differently privileged users

Get more detail on changes before update

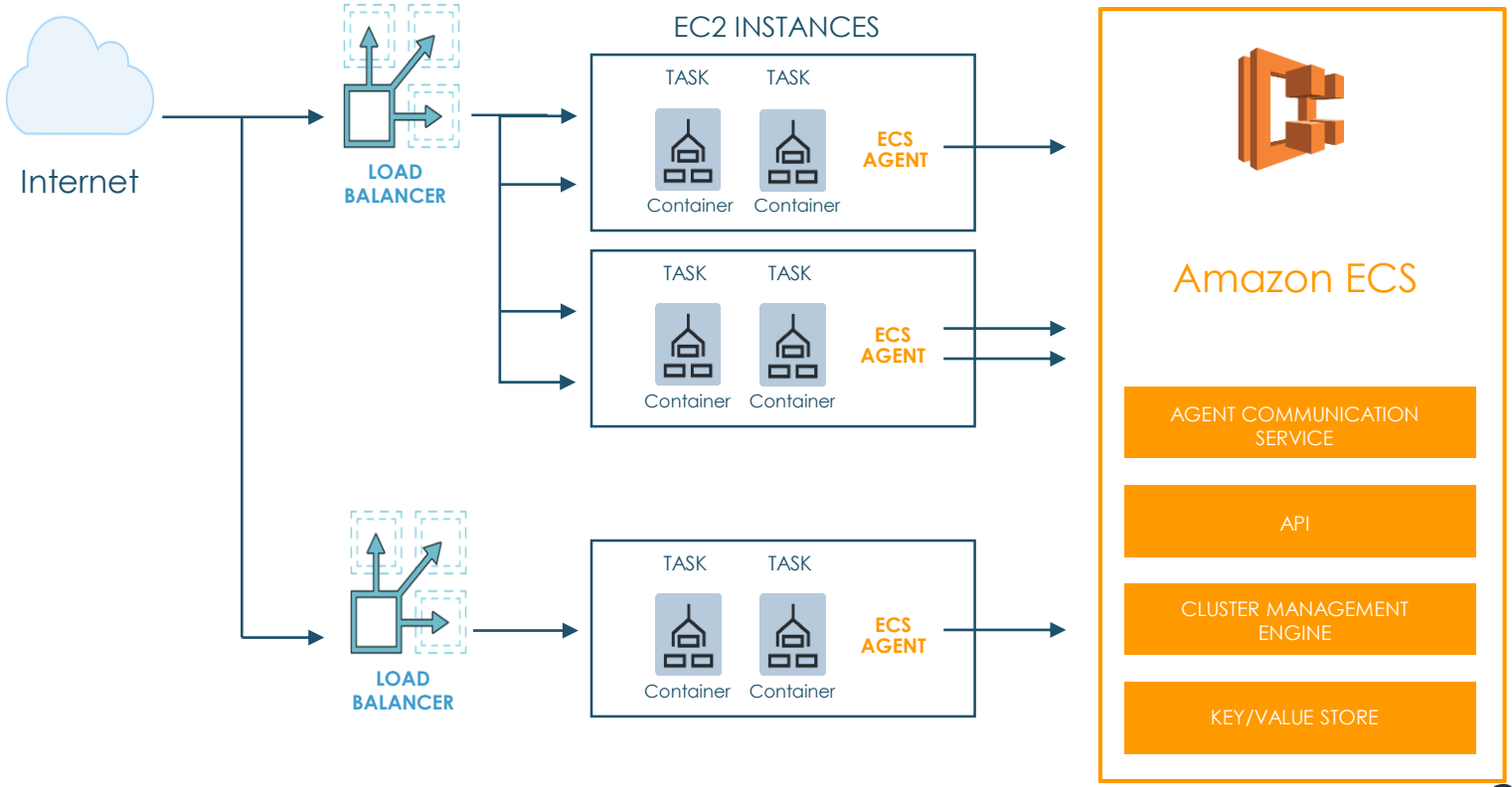
Iteration on infrastructure environment

Amazon Elastic Container Service



- Fully managed elastic service – You don’t need to run anything, and the service scales as your microservices architecture grows
- Shared state optimistic scheduling
- Integration with CloudWatch service for monitoring and logging
- Integration with other services for continuous integration and delivery (CI/CD)

Amazon Elastic Container Service



Amazon ECR

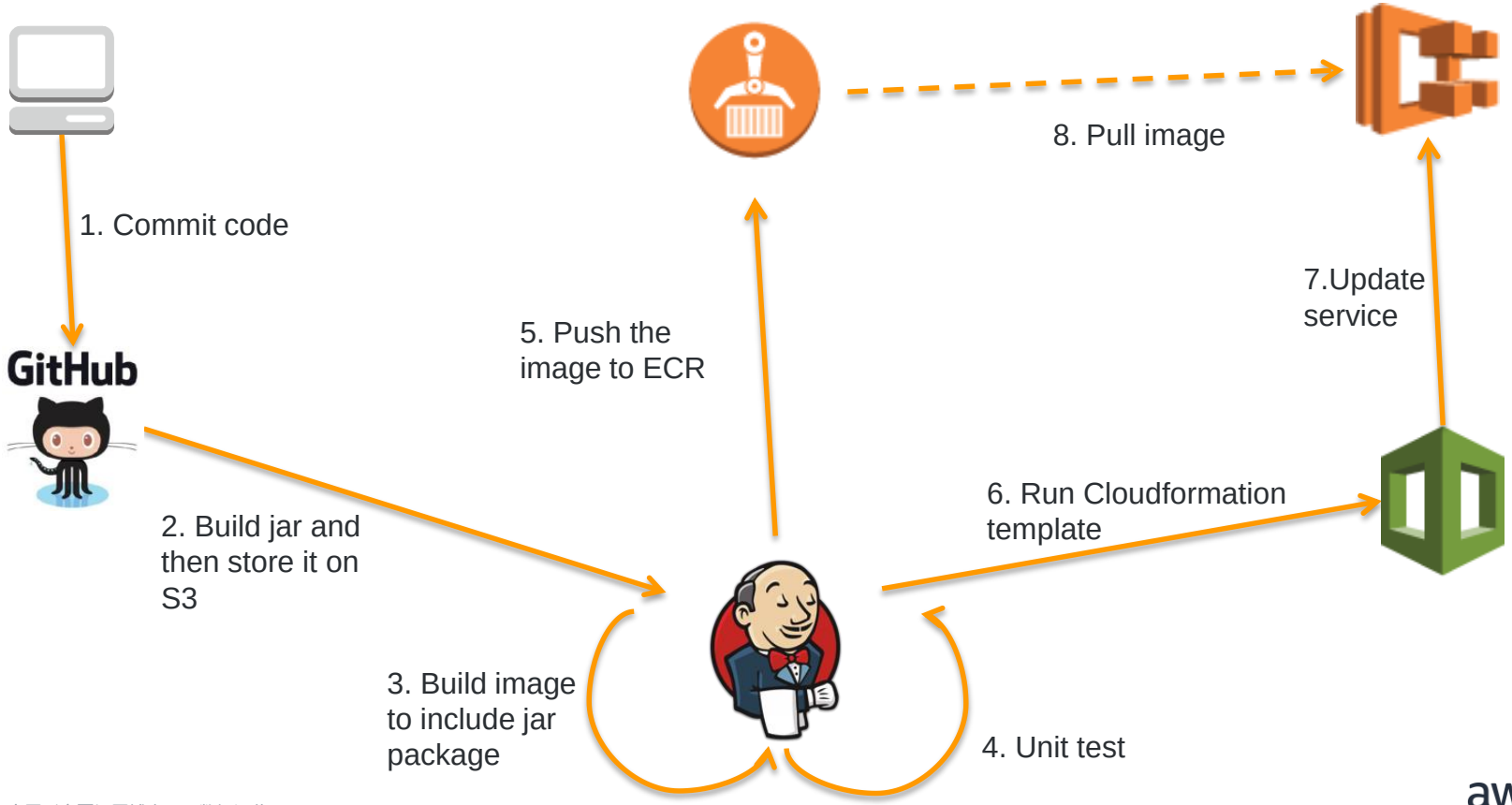
- AWS Managed container registry
- Support Docker CLI，AWS CLI and AWS SDK
- Secure access control, 12h session expiration
- IAM granular access control
- Encrypted data-in-transit and data-at-rest

```
# Authenticate Docker to your Amazon ECR registry
> aws ecr get-login
> docker login -u AWS -p <password> -e none https://<aws_account_id>.dkr.ecr.cn-north-1.amazonaws.com.cn

# List images within a repository called ecr-demo
> aws ecr describe-images --repository-name --repository-name ecr-demo

# Pull an image
> docker pull <aws_account_id>.dkr.ecr.cn-north-1.amazonaws.com.cn/ecr-demo:v1
```

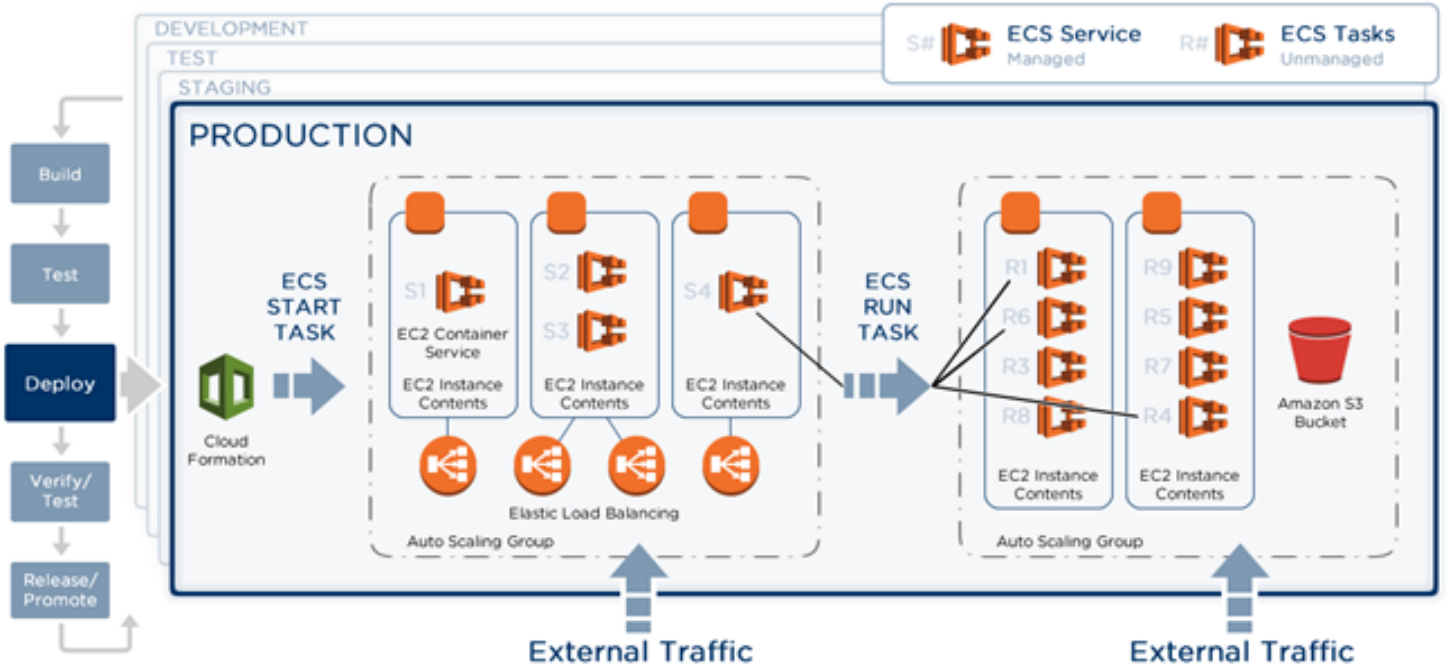
Deploying Containers on ECS – Choose a Scheduler

Batch Jobs	Long-Running Apps
ECS task scheduler	ECS service scheduler
Run tasks once	Health management
Batch jobs	Scale-up and scale-down
RunTask (random)	AZ aware
StartTask (placed)	Grouped containers

Deploying to ECS with Jenkins



Linden Lab case study

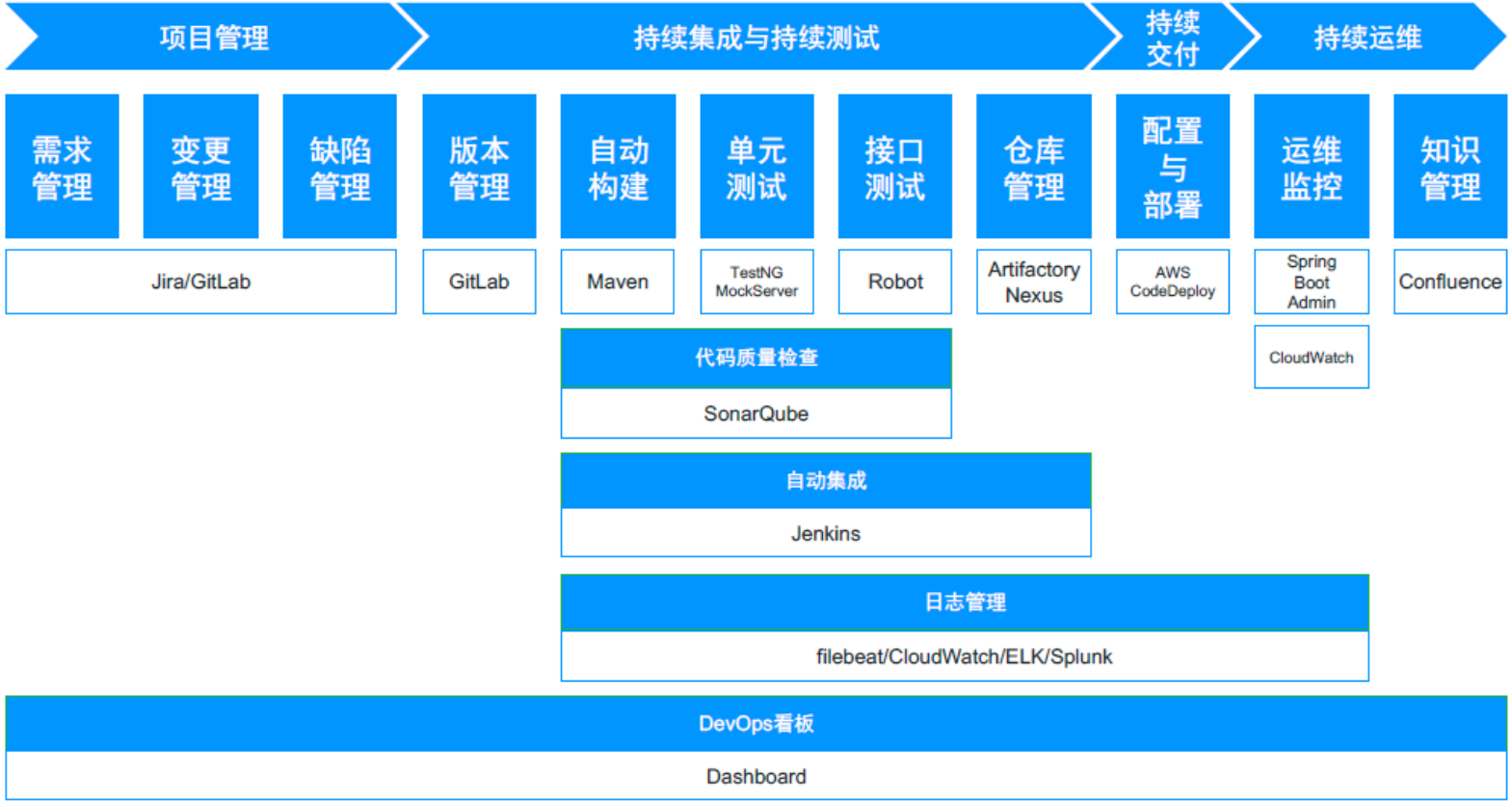


Linden Lab

PROJECT SANSAR

aws

A media company case study



Any Questions?

https://aws.amazon.com/devops/