
FIN303

# DevOps Pipeline Security

## How to use AWS to secure your DevOps Pipeline like a bank


| | | |
|---|---|---|
| Alan Garver | AWS | Sr. Professional Services Consultant |
| Chuck Dudley | Stelligent | Director Financial Services Accounts |
| Jamie Greco | Citi | Sr. VP Technical Program Management |

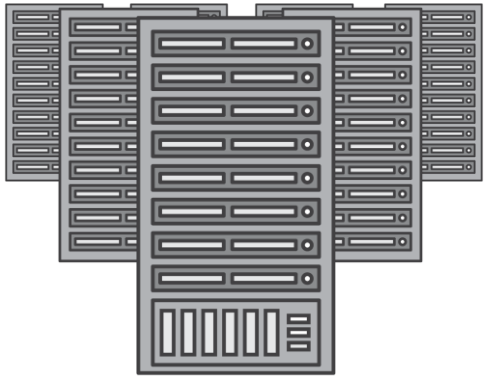



amazon web services

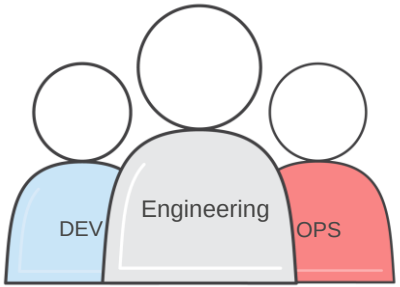
# What to Expect from the Session

- Simple Secure Build Artifact Repository with AWS
- Advanced DevOps Pipeline Concepts
- Static Code Analysis for Infrastructure as Code
- Use AWS Config Rules and AWS Lambda to Monitor Resource Compliance

# Technology Challenges in Financial Services



Monolithic Applications



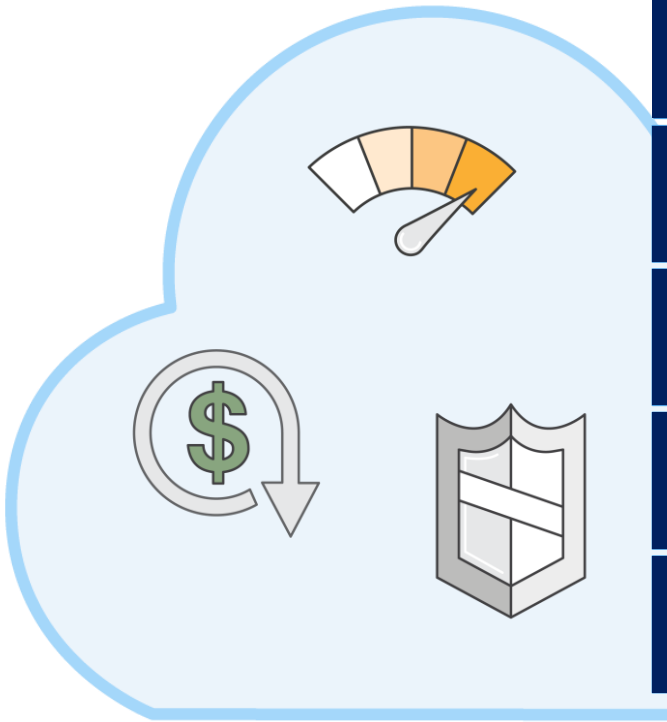

Organizational Boundaries

Regulatory Requirements


citi

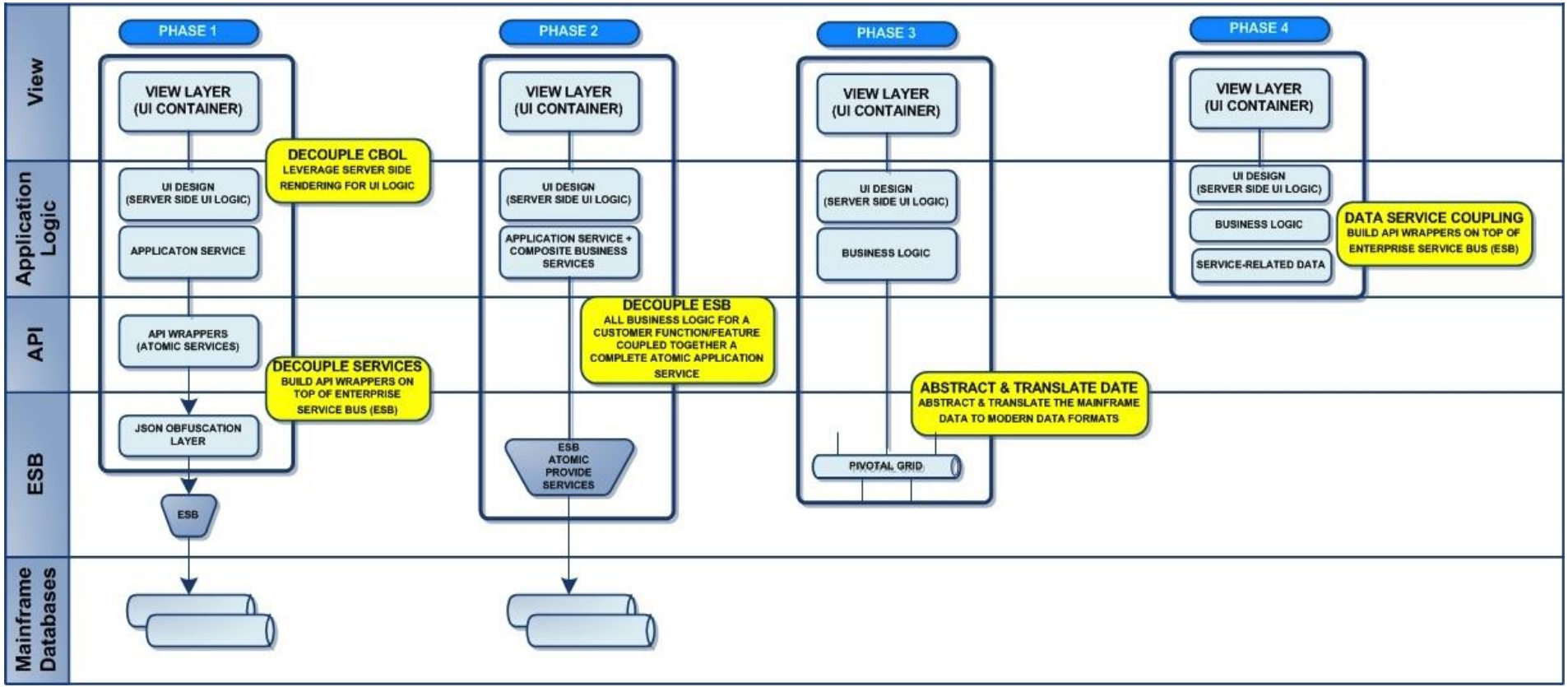
# Enable Continuous Delivery on the Cloud

Establish Cloud platform and enable developers to build and rapidly deploy



| | |
|---|---|
| Provisioning | • Deploy **provisioning tools**<br>• Practice to **provision & manage** all architecture and data components (e.g. operating system ) |
| Monitoring | • Implement **automated systems** to monitor infrastructure and applications to alert abnormal conditions. |
| CI / CD | • **Align disintegrated tools**, people, controls and processes<br>• Focus on **automated builds, orchestration & deployment** capabilities. |
| Orchestration | • **Manage overall orchestration** governing different actions and phases that make up the deployment pipeline (e.g. code check-in to go-live on cloud) |
| Tokenization & Encryption | • Consistent way to **protect information** |

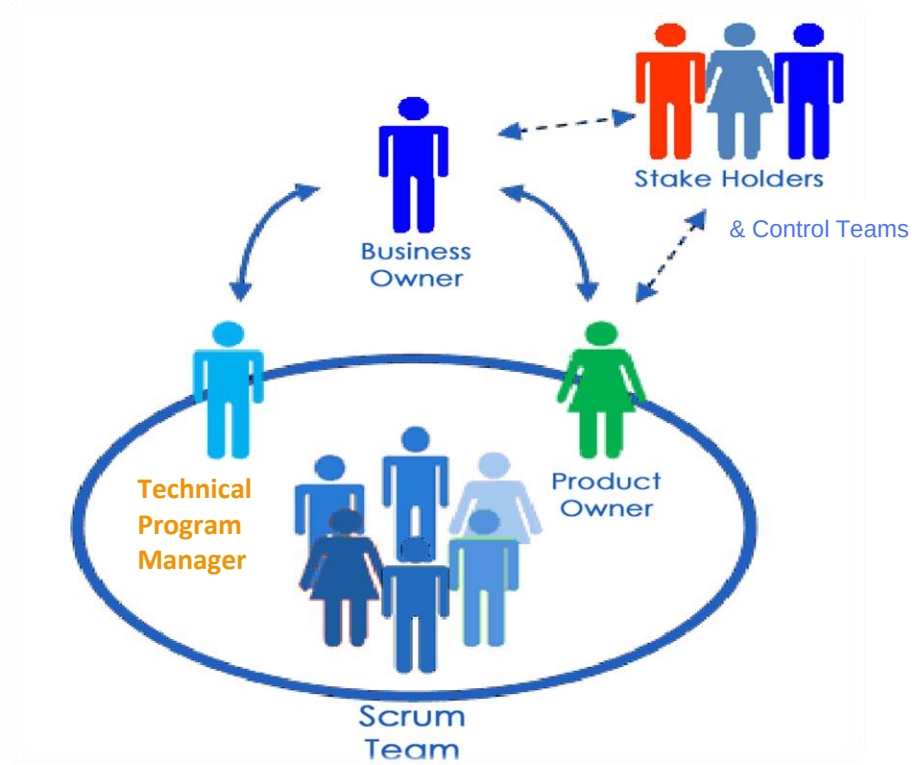
# Journey to Decouple the Mainframe and ESB



# Empowering Teams

Empower teams to accelerate decision making and delivery

| | |
|---|---|
| DEDICATED TEAMS | ▪ Organize in **2-pizza teams**<br>▪ Map capabilities to **service owners** with dedicated teams |
| OWNERSHIP | ▪ **Autonomous teams** that can build, test and deploy independently<br>▪ **Decision making authority** for service at team level |
| TRANSPARENCY | ▪ **Inspection and transparency** of the team performance, service capability and roadmap<br>▪ Services are tracked, mapped and managed via the **Service Catalog** |



# Accelerating Innovation and Product Delivery

## 1 BUILD GLOBAL CLOUD FOUNDATION

- Deploy cloud infrastructure
- Establish scale and availability
- Enable continuous integration/continuous delivery
- Protect Citi information

## 2 BUILD MICROSERVICES

- Create operating framework
- Establish design patterns for microservices
- Build, re-use and extend services
- Test driven development

## 3 EMPOWER TEAMS

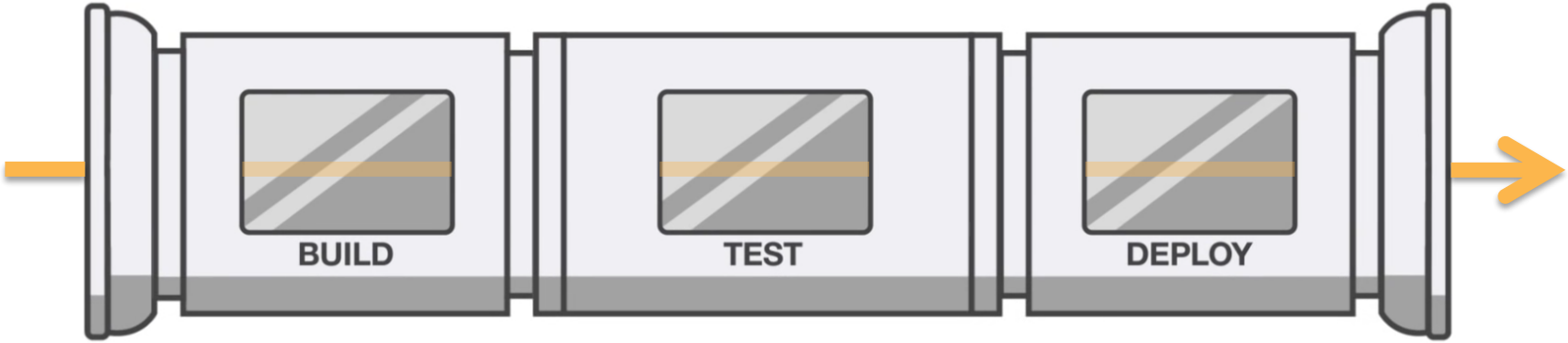
- Build full stack, autonomous agile, scrum teams
- Single ownership structure
- Empowered development with decentralized functions
- Continuous integration / deployment

IMPROVING

**SPEED, COST & QUALITY**

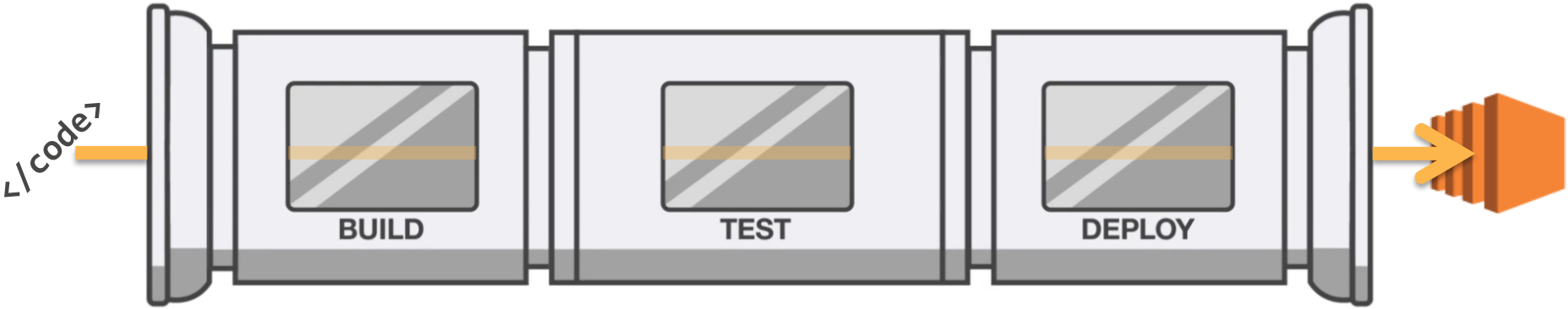
# The DevOps Pipeline

# Continuous Delivery Pipeline



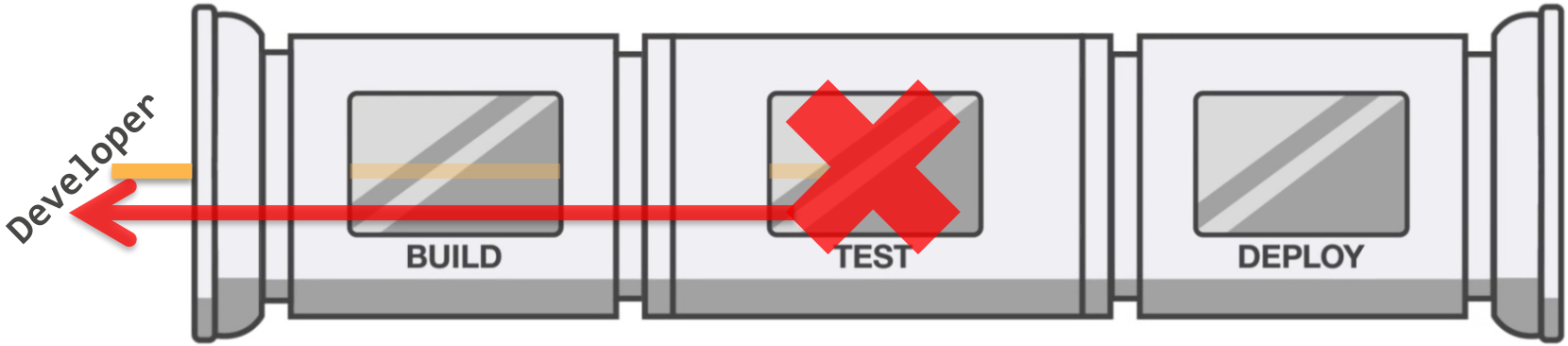

- A secure automated transport mechanism
- Moves a resources from point A to point B

# Continuous Delivery Pipeline




- Transports code from development to production
- Tests ensure integrity and validity of the resource
- Resources morph from source, to executable, to operational

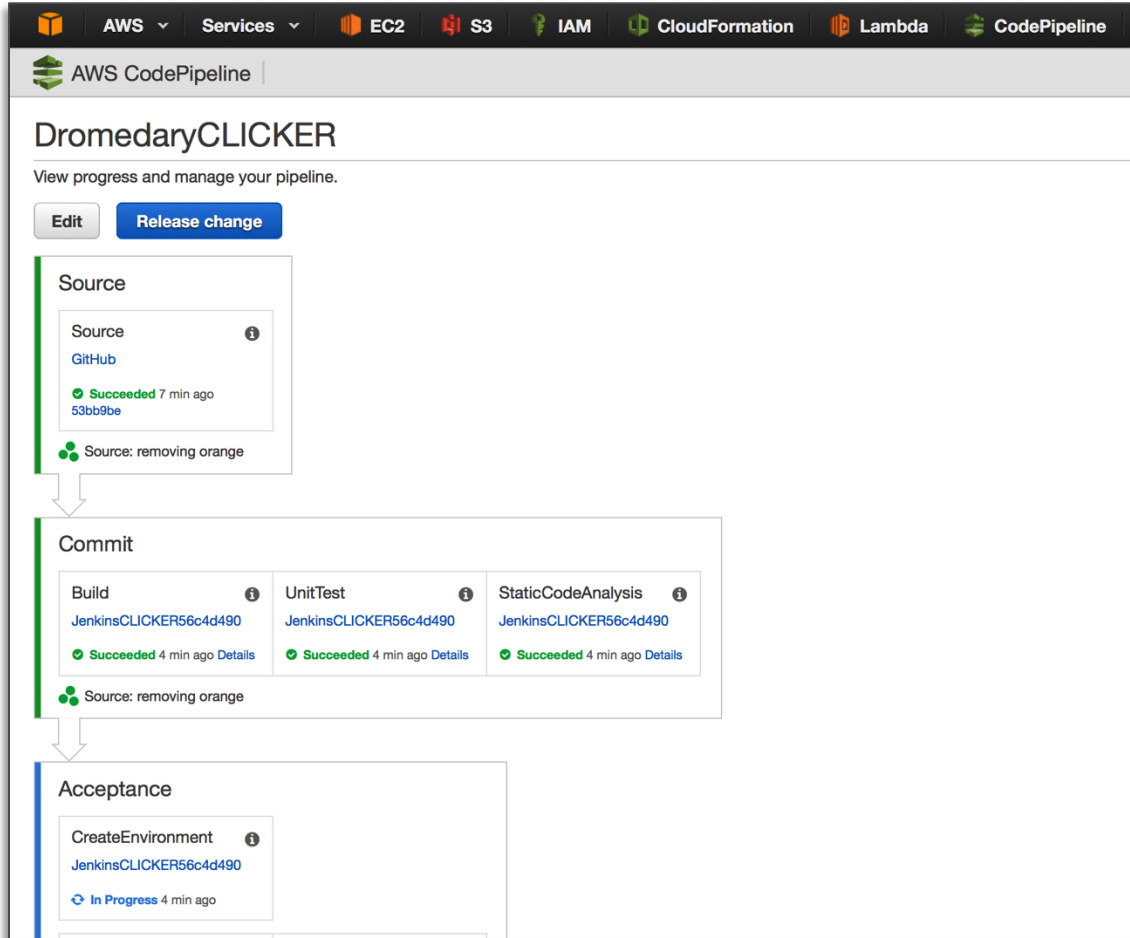
# Continuous Delivery Pipeline



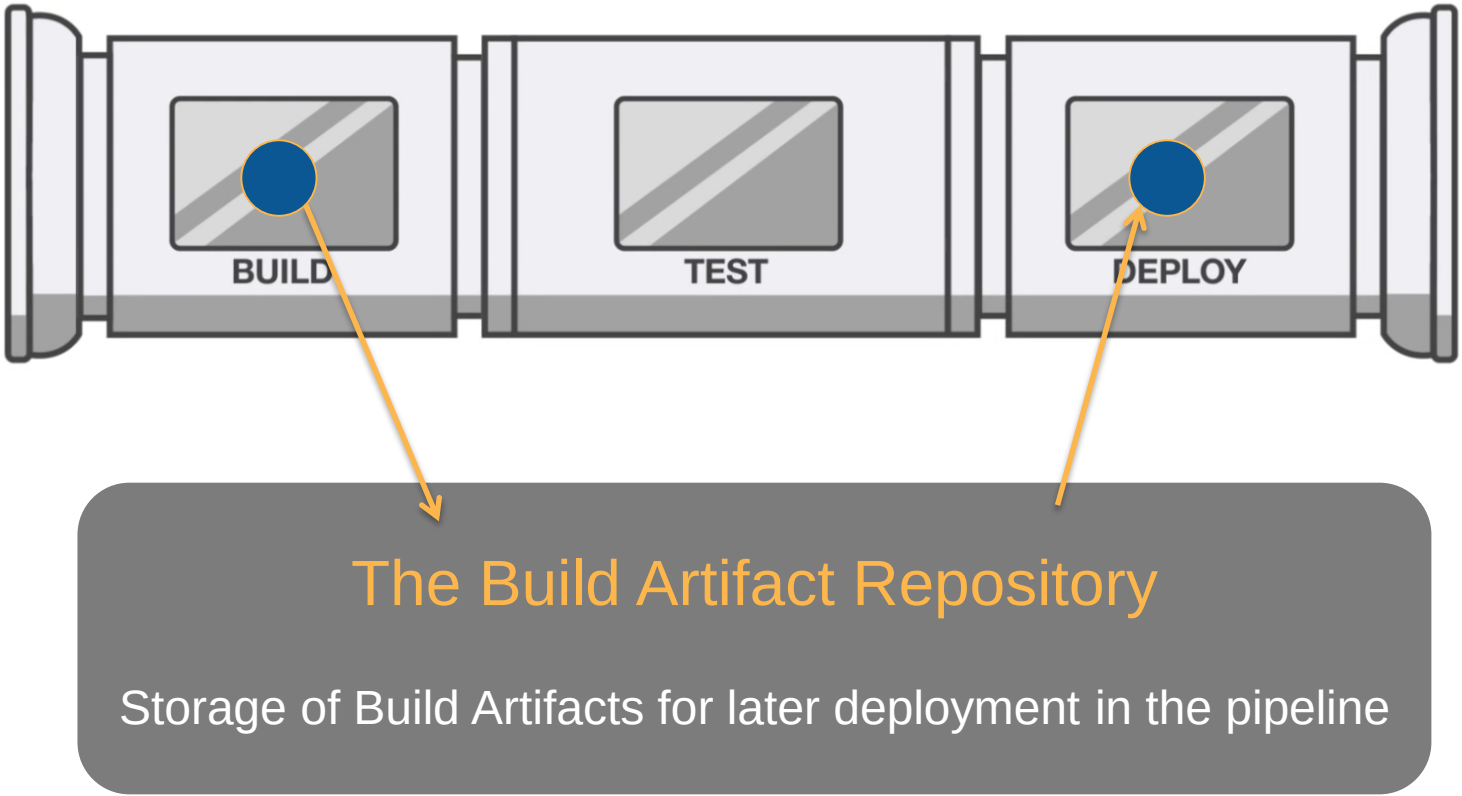

- Failures stop the line, and prevent breakages to production
- Fast feedback provided to the developer
- Customized to your software development lifecycle

# AWS CodePipeline

- Quickly model and configure release stages
- View progress at-a-glance
- Use your favorite tools
- Integrates with other AWS services



# The Build Artifact Repository

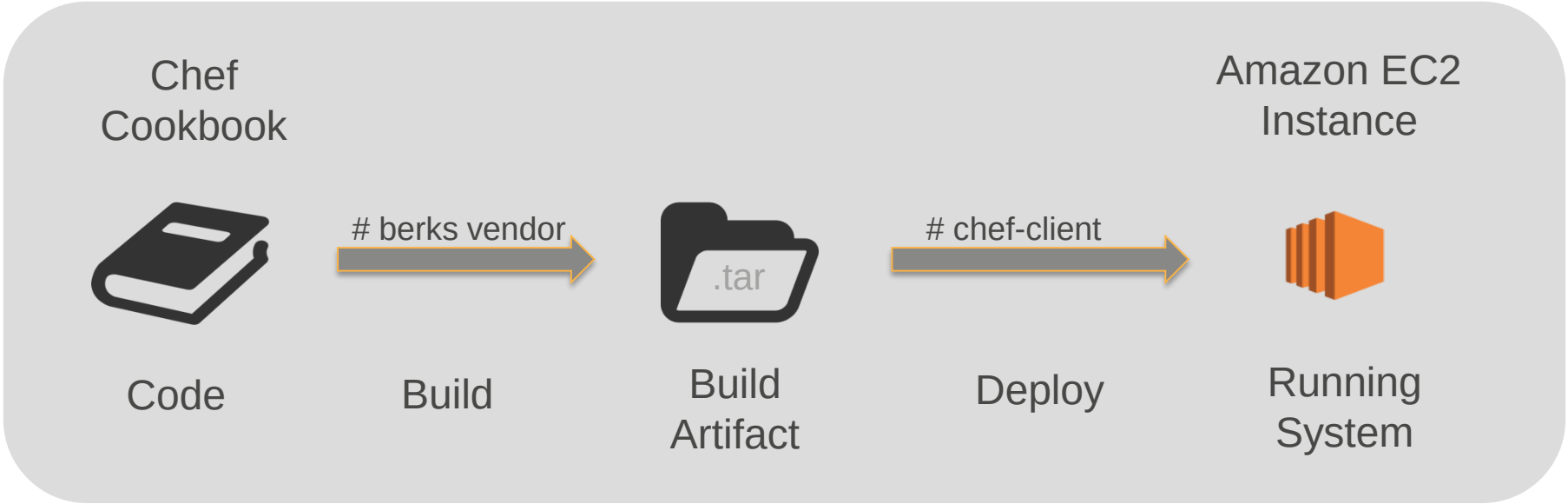


# Why Build Artifact Repository

- Build once, deploy many times
- Version control
- Artifacts available for later deploy events (Scale Up)
- Build Server and Deployed Services don’t need to talk to each other

# Pipeline Build Artifacts

Objects assembled during a build process from code used for testing and convergence down stream in a pipeline



# Examples of Build Artifacts







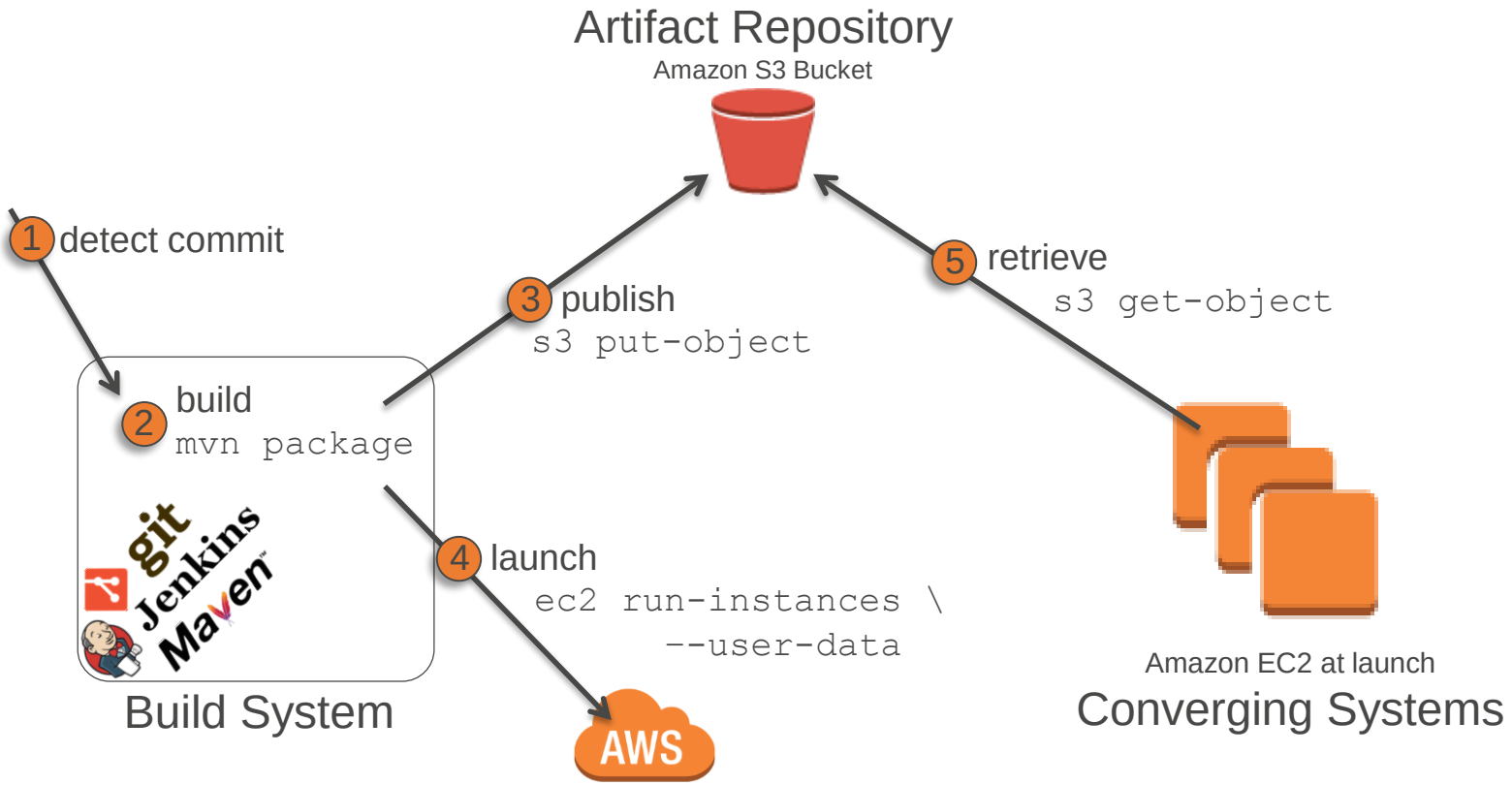









# Simple Artifact Repository with AWS



# Pipeline Build Artifacts Like a Bank

Data Protection

AWS KMS

Entitlement

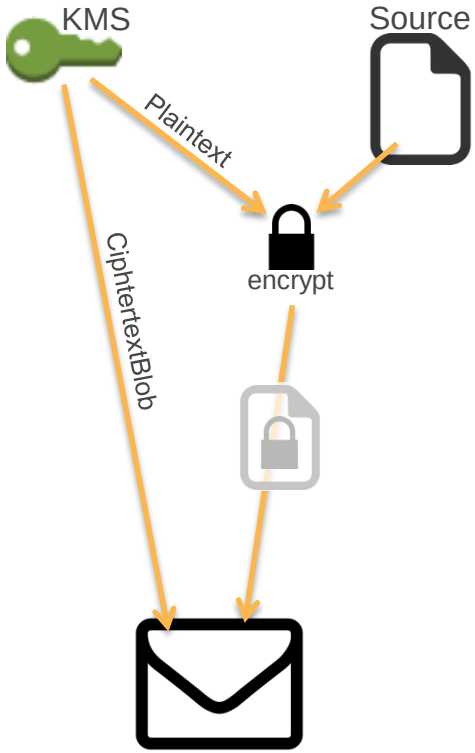
AWS IAM

Integrity

sha256sum

- Generate Data Keys for client side encryption
- Use Server Side Encryption integration with Amazon S3
  - Use IAM Roles to grant access to resources
  - Implement strict resource policies for S3 Buckets and KMS Keys
    - Validate integrity with sha-sum
    - Implement sha integrity database

# Envelope Encryption with AWS KMS

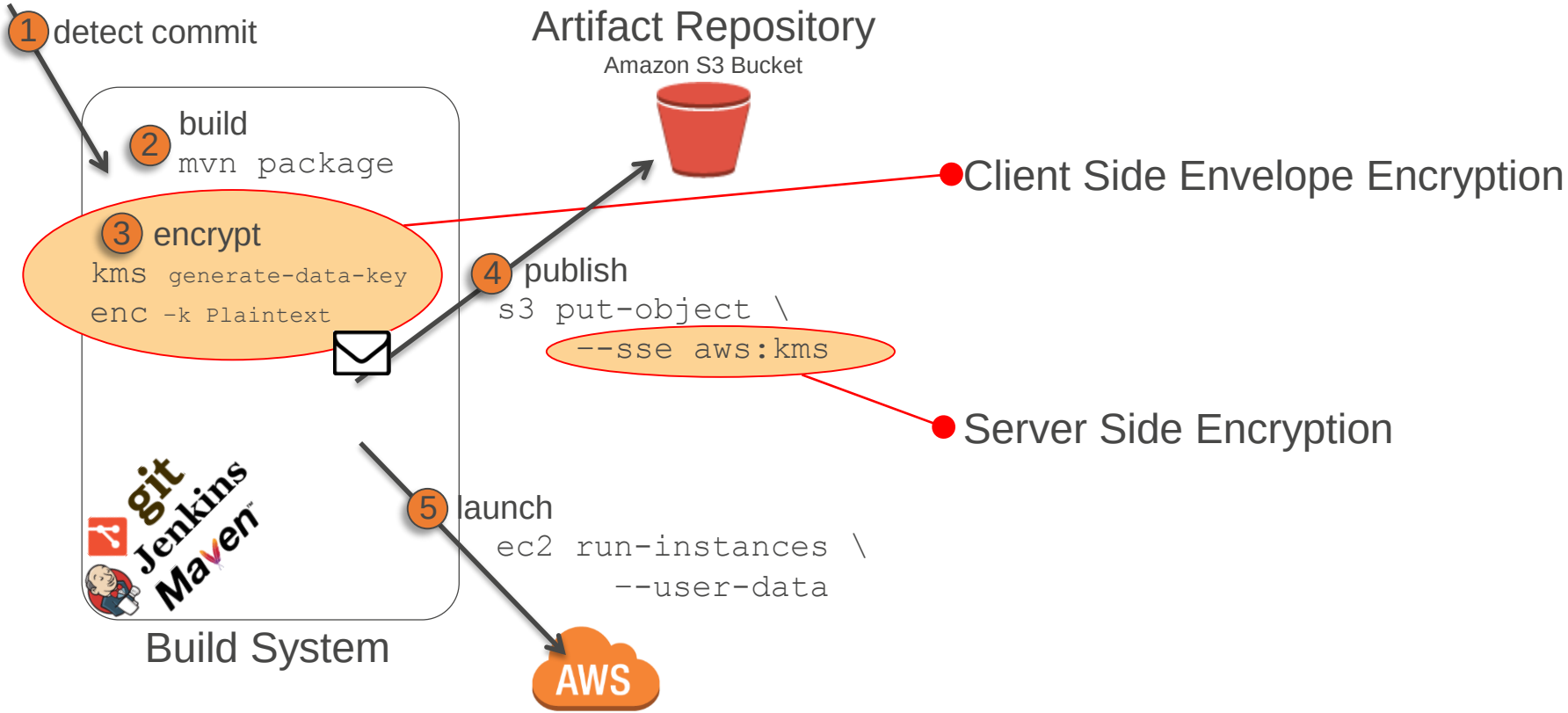



```
$> aws kms generate-data-key  \
      --key-id alias/artifact-demo  \
      --key-spec AES_256 --output text  \
      --query [Plaintext,CiphertextBlob]

$> openssl enc -aes-256-cbc -salt  \
      -in source.tar  \
      -out encrypted.out  \
      -k ${Plaintext}

$> tar -czvf artifact.tgz  \
      encrypted.out  \
      CiphertextBlob.out
```

# Artifact Repository on AWS with encryption



# Entitle Access with Resource Policies

## Artifact Repository

Amazon S3 Bucket

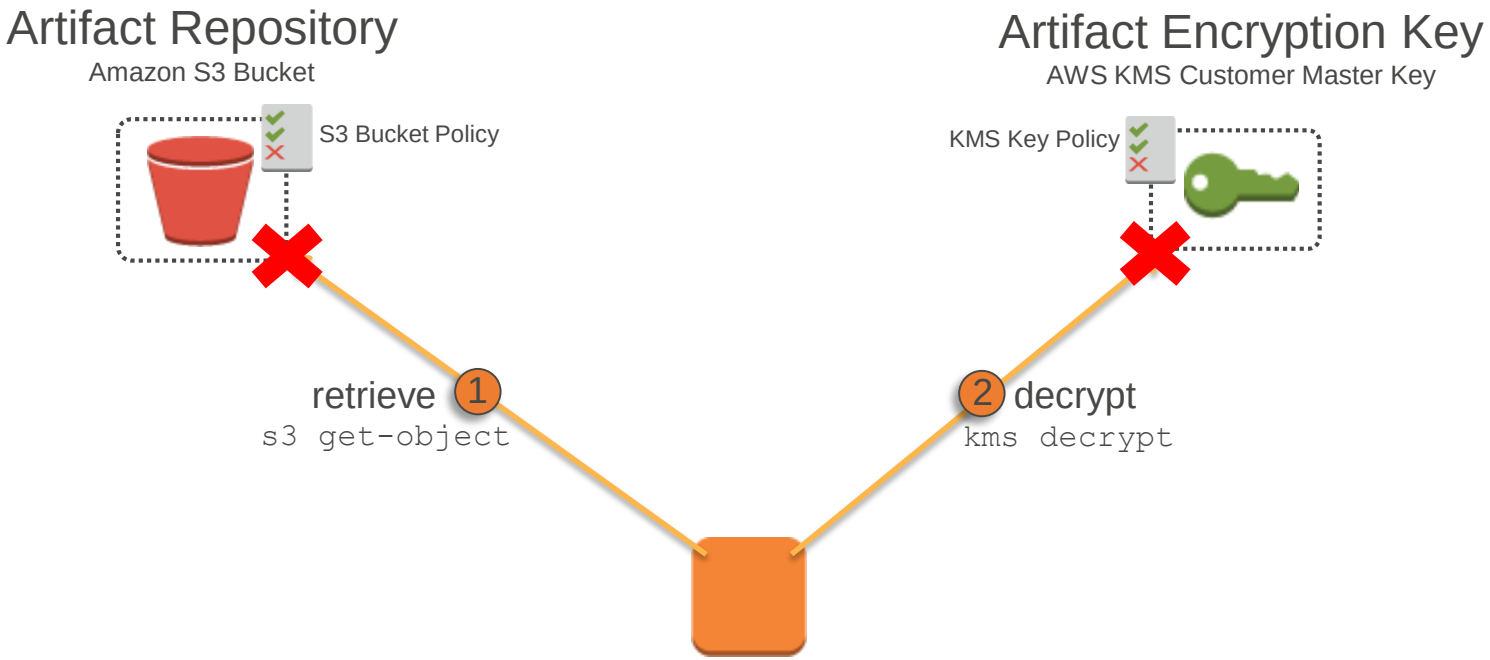

```
ArtifactBucketPolicy:
  Type: "AWS::S3::BucketPolicy"
  Properties:
    Bucket: !Ref ArtifactS3Bucket
    PolicyDocument:
      Version: "2012-10-17"
      Id: "ArtifactRepositoryBucketPolicy"
      Statement:
      - Sid: "Fetch"
        Action:
        - "s3:GetObject"
        - "s3:GetObjectAcl"
        Effect: "Allow"
        Resource: !Join [ '', ['arn:aws:s3:::', !Ref ArtifactS3Bucket, "/*" ] ]
        Principal:
          AWS: !GetAtt ArtifactClientRole.Arn
      - Sid: "ListBucket"
        Action: "s3:ListBucket"
        Effect: "Allow"
        Resource: !Join [ '', ['arn:aws:s3:::', !Ref ArtifactS3Bucket ] ]
        Principal:
          AWS: !GetAtt ArtifactClientRole.Arn
```

## Artifact Encryption Key

AWS KMS Customer Master Key


```
  KeyPolicy:
    Version: "2012-10-17"
    Id: "key-default-1"
    Statement:
      -
        Sid: "ArtifactClients"
        Effect: "Allow"
        Resource: "*"
        Principal:
          AWS: !GetAtt ArtifactClientRole.Arn
        Action:
          - "kms:Decrypt"
          - "kms:DescribeKey"
```

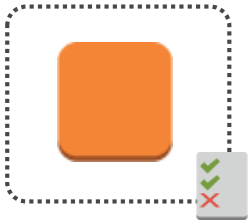
# Entitle Access with Resource Policies




Converging Systems

# Entitle Access with Resource Policies

```
ArtifactClientRole:
  Type: "AWS::IAM::Role"
  Properties:
    Path: "/"
    AssumeRolePolicyDocument:
      Version: "2012-10-17"
      Statement:
        Effect: "Allow"
        Action: "sts:AssumeRole"
        Principal:
          Service: "ec2.amazonaws.com"
    Policies:
    - PolicyName: "ArtifactConsumer"
      PolicyDocument:
        Version: "2012-10-17"
        Statement:
          Effect: "Allow"
          Action:
          - "s3:GetBucketLocation"
          - "s3:ListAllMyBuckets"
          Resource: "arn:aws:s3:::*"
```

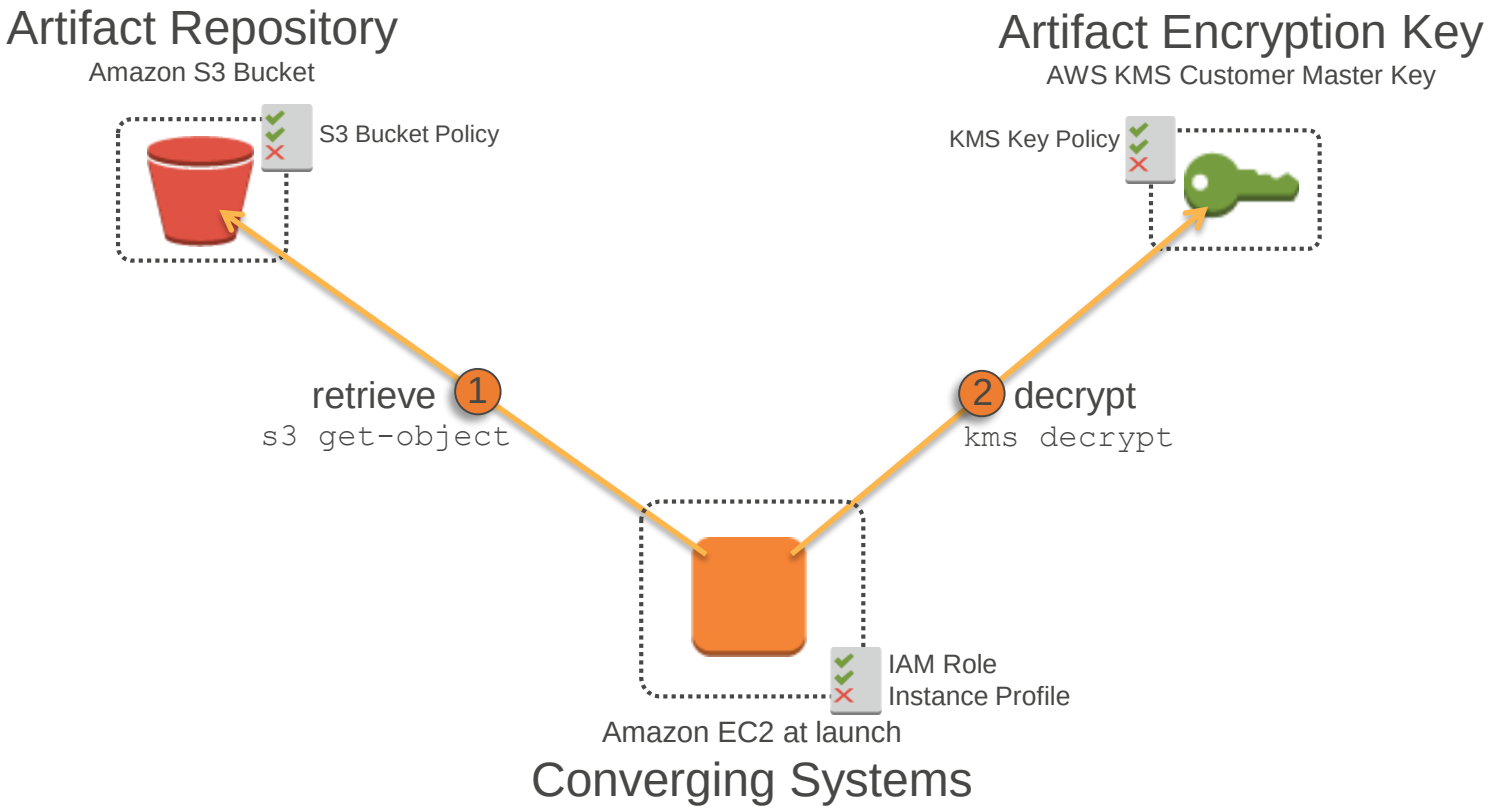


IAM Role
Instance Profile

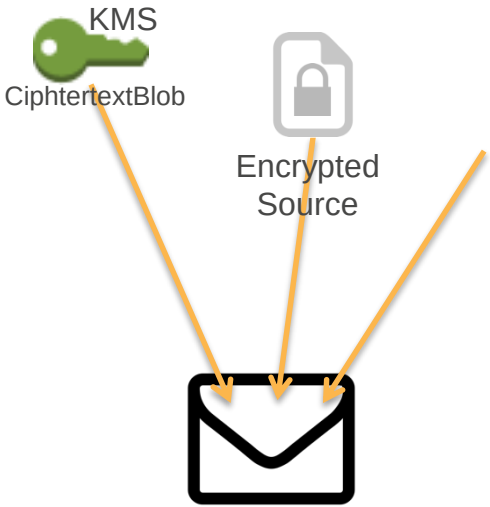
Amazon EC2 at launch

Converging Systems

# Entitle Access with Resource Policies



# Validate Artifact Integrity

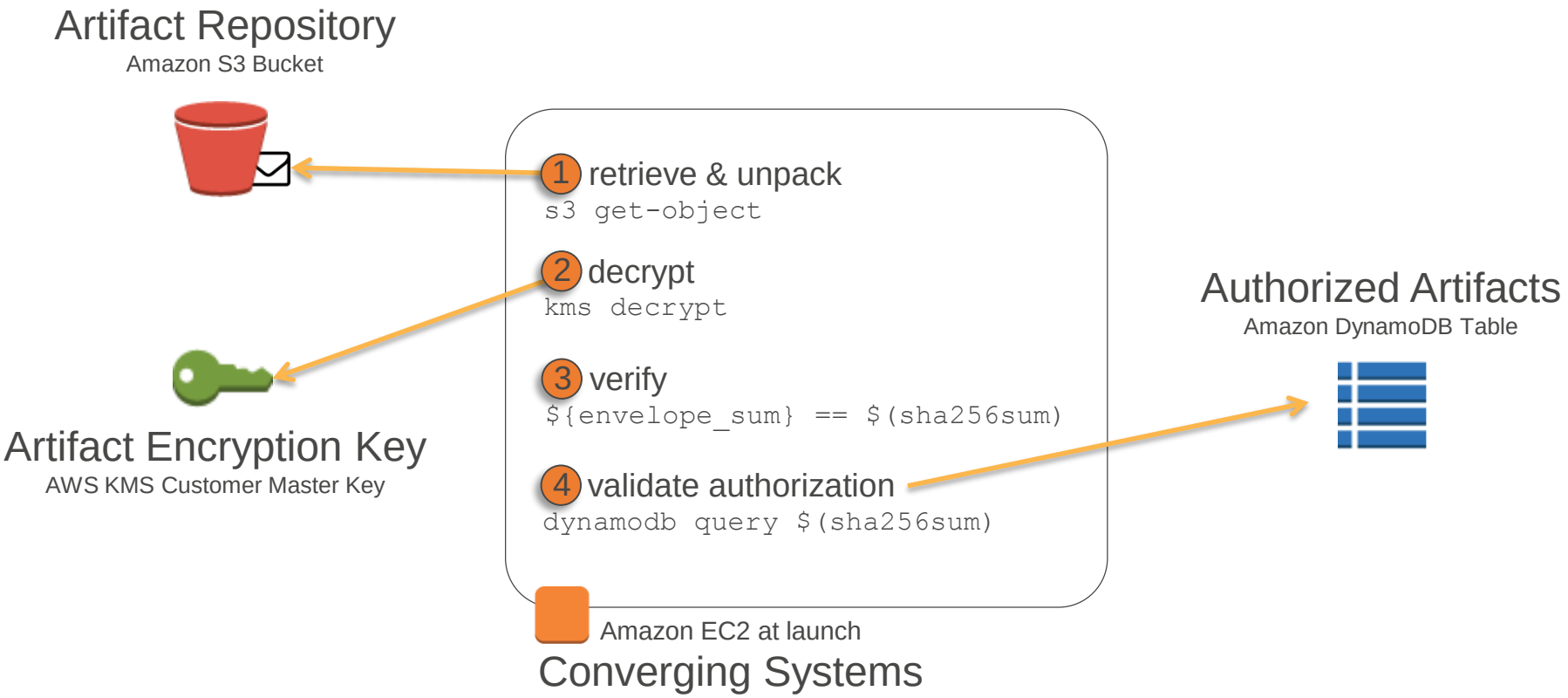



```
$> sha256sum mysource

b2f3fb7e84761eac78eb34aaaae2793efb41f23141a31f2c  mysource
```

```
$> tar -czvf artifact.tgz  \
      encrypted.out  \
      sha256sum.out \
      CiphertextBlob.out
```

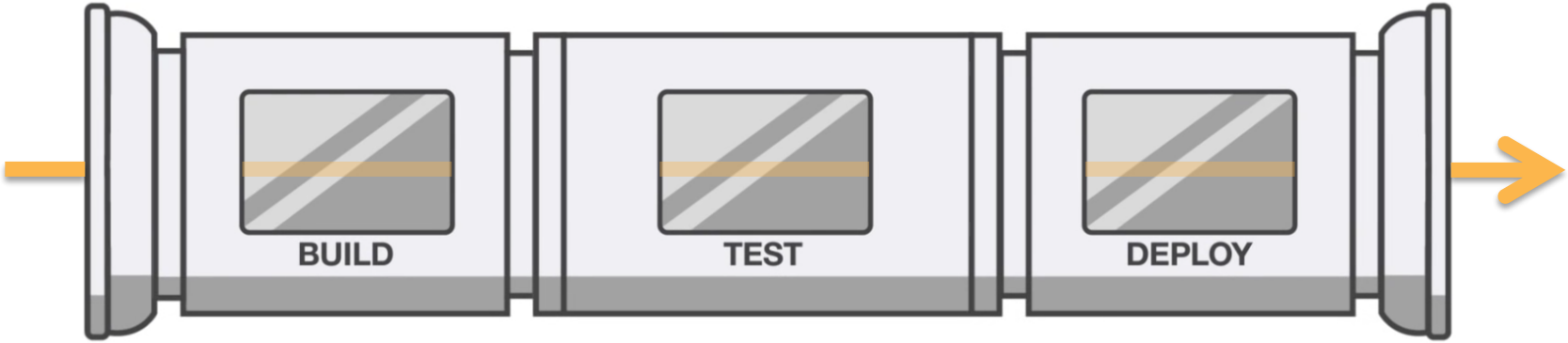
# Validate Artifact Integrity



stelligent™

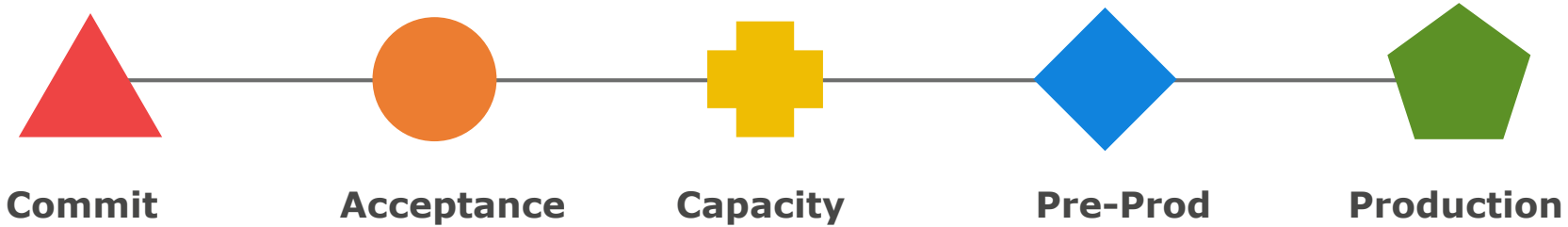

amazon
web services
Partner
Network
ADVANCED CONSULTING PARTNER
DEVOPS COMPETENCY

# Continuous Delivery Pipeline

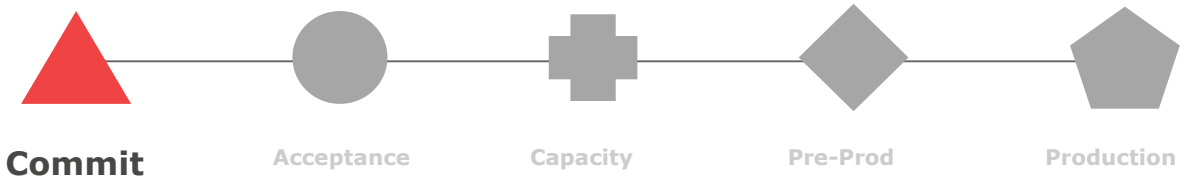



- A secure automated transport mechanism
- Moves a resources from point A to point B

# The Stelligent Pipeline



# The Commit Stage




**GOAL:**

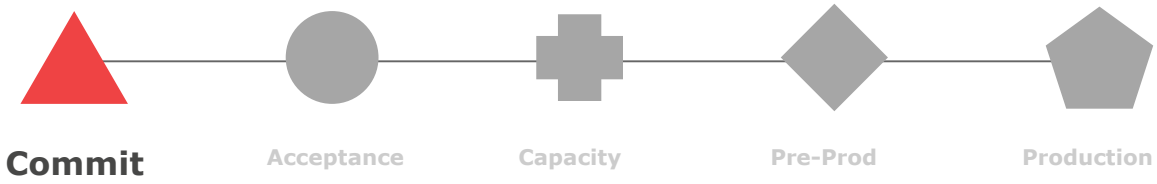
Fast feedback for developers

---

**PIPELINE ACTIONS:**

1. Unit Tests
2. Static Code Analysis

# The Commit Stage




**GOAL:**

Fast feedback for developers

---

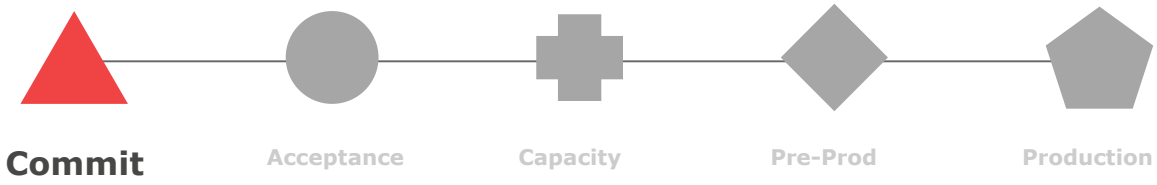
**PIPELINE ACTIONS:**

1. Unit Tests
2. Static Code Analysis

**SECURITY TESTS:**

1. Security static analysis of application code

# The Commit Stage




**GOAL:**

Fast feedback for developers

---

**PIPELINE ACTIONS:**

1. Unit Tests
2. Static Code Analysis

**SECURITY TESTS:**

1. Security static analysis of application code
2. Security static analysis of ***infrastructure code***

# Security Static Analysis of CloudFormation

- Security static analysis builds a model of templates in order to verify compliance with best practices and organizational standards.
- This can be a powerful tool to stop bad things ***before*** they happen.
- A security organization can define their policy in code and have all development efforts unambiguously verify against that standard without manual intervention.

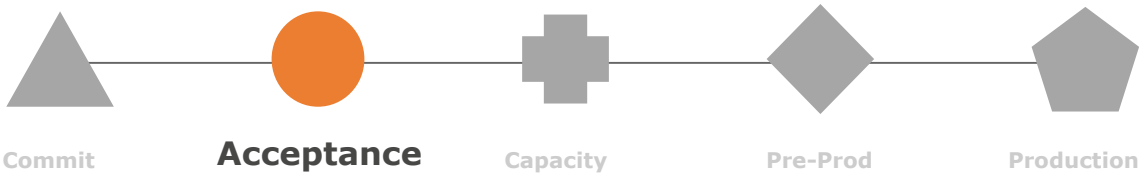
# Static Analysis of CloudFormation with cfn-nag

The cfn-nag tool inspects the JSON of a CloudFormation template ***before*** convergence to find patterns that may indicate:

- Overly permissive IAM policies
- Overly permissive security groups
- Disabled access logs
- Disabled server-side encryption

# *Demo*

# The Acceptance Stage




**GOAL:**

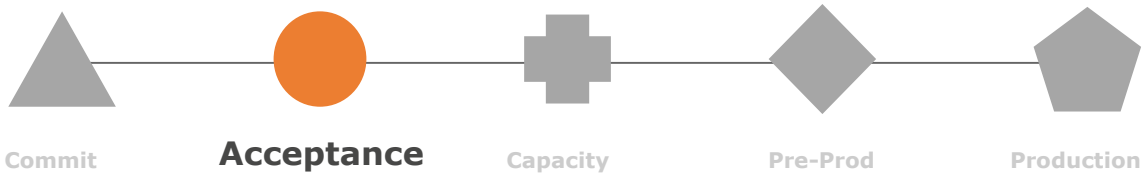
Comprehensive testing of the application and its infrastructure

---

**PIPELINE ACTIONS:**

1. Integration Tests
2. Acceptance Tests

# The Acceptance Stage




**GOAL:**

Comprehensive testing of the application and its infrastructure

---

**PIPELINE ACTIONS:**

1. Integration Tests
2. Acceptance Tests

**SECURITY TESTS:**

1. Infrastructure Analysis

# Testing Infrastructure Changes

**Problems to solve:**

- Prevent infrastructure changes that violate company security policies.
- Need the ability to codify security rules and get notifications when violations occur.
- Ability to execute on-demand compliance testing.

# Testing Infrastructure Changes

**AWS Config solves these problems, but…**

- Pipeline enablement can be challenging.
- Console-centric.

# config-rule-status

ConfigRuleStatus is an open source tool that enables continuous monitoring and on-demand testing of security compliance for infrastructure through the AWS Config service.

## How does it solve the problem?

- Sets up AWS Config for resource monitoring.
- Creates Config Rules and Lambda functions to evaluate security compliance.
- Creates a Tester Lambda function that returns aggregated compliance status.

# config-rule-status

## How should it be used?

- The bundled CLI provides commands for deploying the tool.
- The Tester Lambda function can be invoked with the bundled CLI or the AWS CLI.
- Invoke it from a CD pipeline to catch policy violations before they get to production.

# Core Technology

**Config -** for monitoring AWS resources and defining security rules

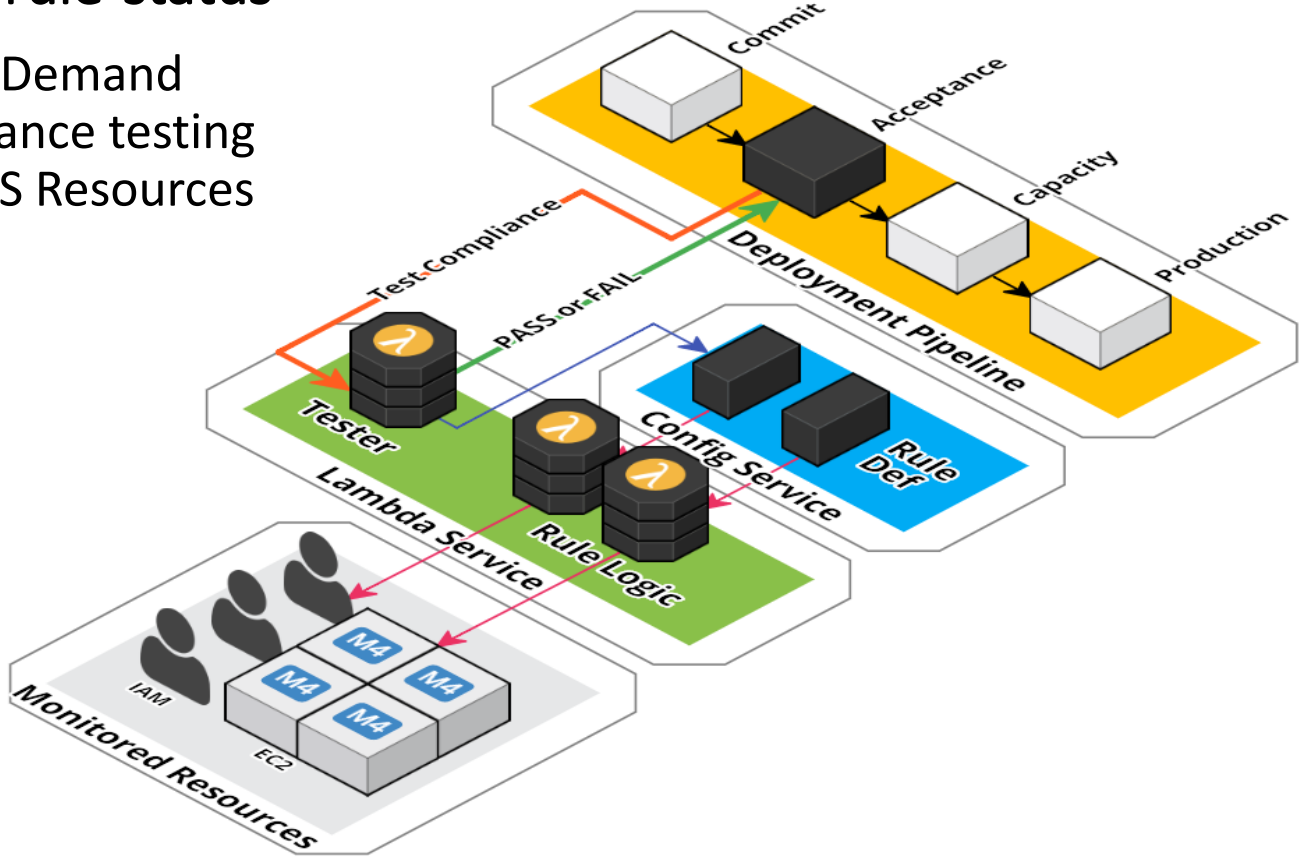
**Lambda** - used as the platform for Config Rule logic implementation

**CloudFormation** - for programatic provisioning of all supporting resources

**Serverless Framework** - for orchestrating deployment of Lambda functions and their supporting CloudFormation stacks.

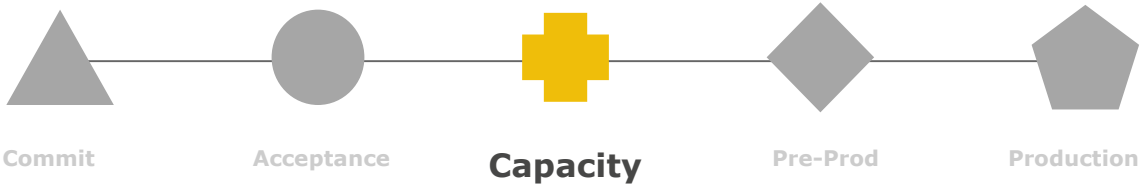
# config-rule-status

On-Demand
compliance testing
for AWS Resources



# *Demo*

# The Capacity Stage




**GOAL:**

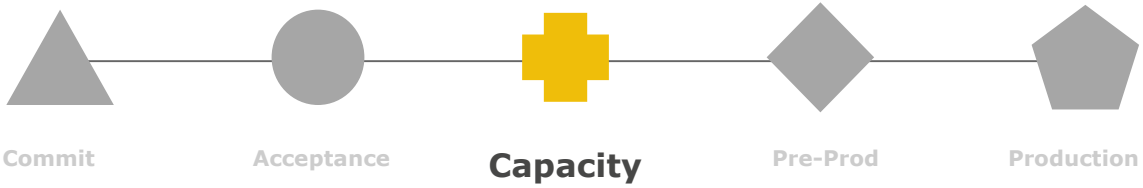
Test the system under real world conditions

---

**PIPELINE ACTIONS:**

1. Performance Tests
2. Load Tests

# The Capacity Stage




**GOAL:**

Test the system under real world conditions

---

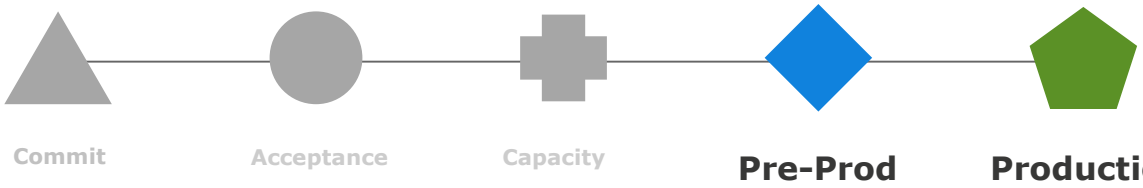
**PIPELINE ACTIONS:**

1. Performance Tests
2. Load Tests

**SECURITY TESTS:**

1. OWASP ZAP Pen Test
2. OpenSCAP Image Testing

# The Production Stage




**GOAL:**

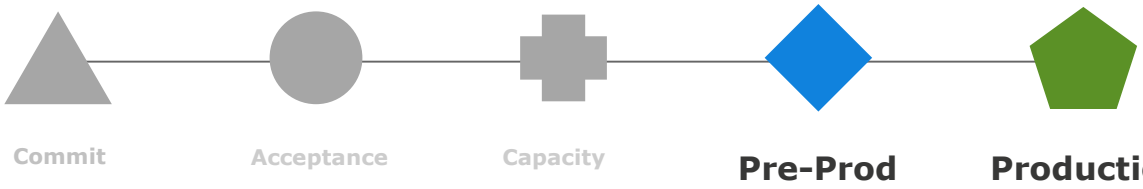
Go / no-go decision for blue/green deployment

---

**PIPELINE ACTIONS:**

1. Build Pre-Prod Stack
2. Data Migration
3. Blue/green Deployment

# The Production Stage




**GOAL:**

Go / no-go decision for blue/green deployment

---

**PIPELINE ACTIONS:**

1. Build Pre-Prod Stack
2. Data Migration
3. Blue/green Deployment

**SECURITY ACTIONS:**

1. Prevent out-of-band changes
2. Security metrics for feedback loops

# Resources

stelligent.com/fin303


# Thank you!

# Remember to complete your evaluations!