DevOps Workshop Series

DevSecOps

DevSecOps

Goal

Integrate previous seven modules in brief fashion to cover security and compliance aspects through each element

- **Module 1 - DevOps**
- **Module 2 – CI/CD Pipelines**
- **Module 3 – Continuous Deployment**
- **Module 4 – Infrastructure as Code**
- **Module 5 – Continuous Testing**
- **Module 6 – Observability**
- **Module 7 – SRE and Incident Management**
- **Module 8 - DevSecOps**

Security considerations

Security exists in two phases

- *To GUARD your value*
- *To SIGNAL a threat*

Advancing DevOps perceptions fine-tunes the ability to understand dynamic value through improved experimentation

Teams fix problems with humans

Put the problem on the table

Problem – security teams used to separation?

- *Twenty years ago, they did ops, or dev, or networking*
- *Success resulted in promotion to compliance*
- *Let's bring folks back to the same team*

What are my integration parameters?

- *Standards*
- *Risk management*
- *Buying a solution*

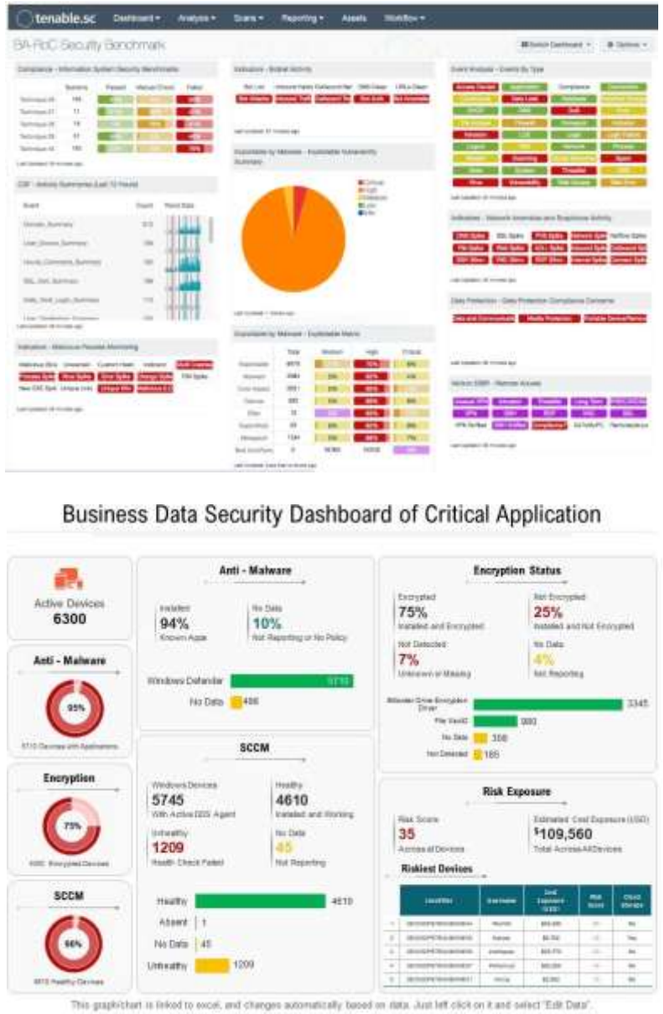
Re-valuing security

Where are my ‘crown jewels’ compared to the business?

- *What do customers want to secure? Data? Goods ? IP?*
- *What do managers want to secure? Revenues? Bonuses?*
- *What do employees want to secure? Employment? Bonuses?*

When you find a value stream, where is security?

- *An –ility such as scalability, adaptability, observability?*
- *Baked in and left to cool?*
- *A dynamic part of development and operations?*



Security gaps in DevOps

Different gaps create the whole hole

Compliance gaps

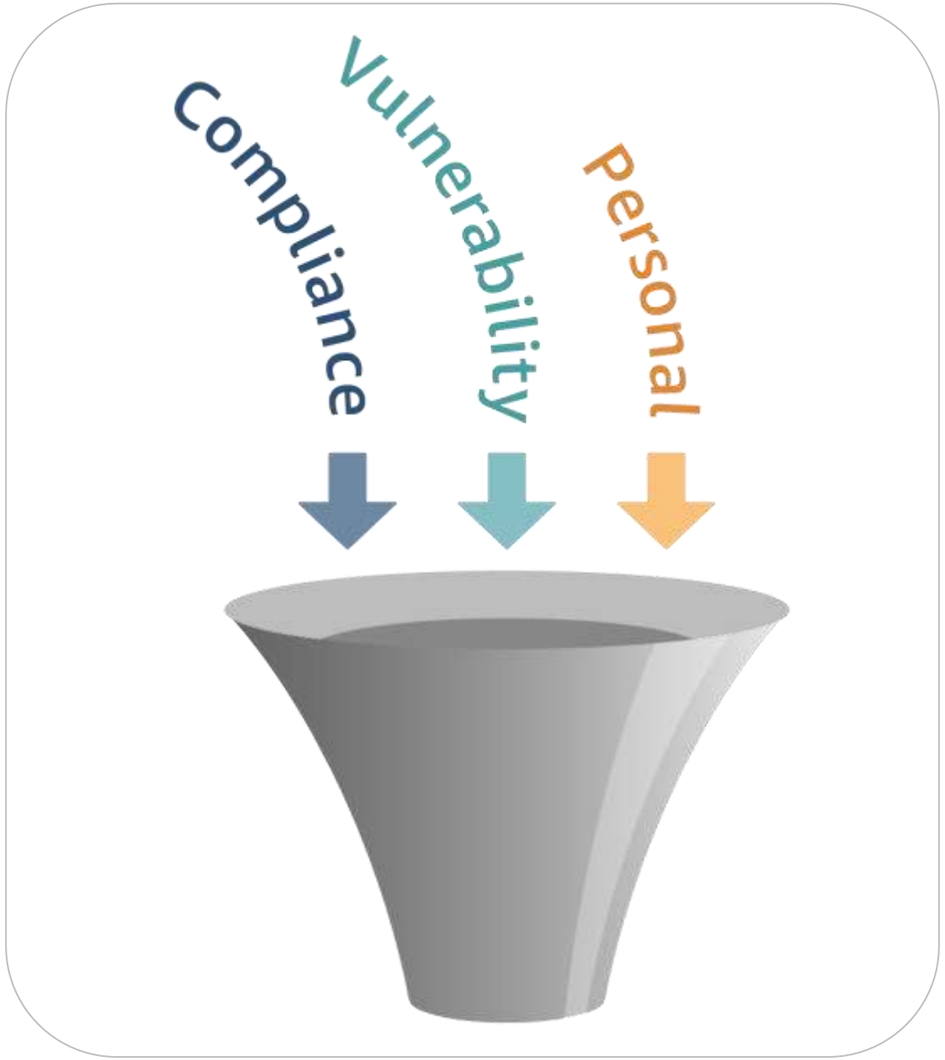
- *Failed to meet standard*
- *No policy created*

Vulnerability gaps

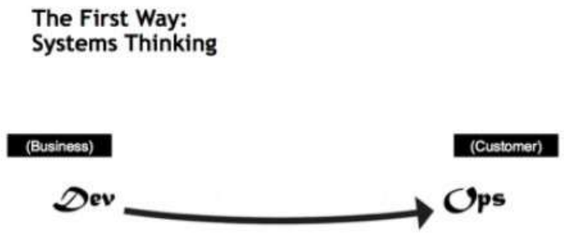
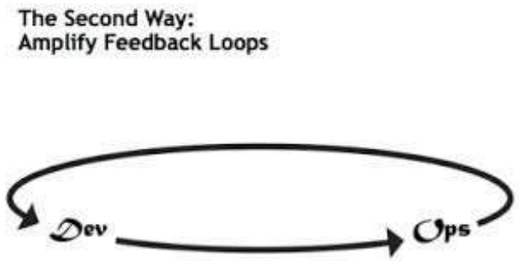
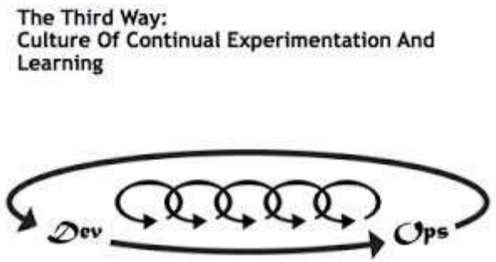
- *Scanned and failed*
- *New hole released*

Personal gaps

- *Failed to patch*
- *Introduced vulnerability*
- *Attacked by a rude person*



The Three Ways

		
FLOW	**FEEDBACK**	**CONTINUOUS EXPERIMENTATION AND LEARNING**
Ensure that security is not a constraint in the flow of work – shift security testing as far left as possible and automate. Use pre-approved security libraries, think like a value stream, and create mutual accountability.	*Ensure fast feedback by automating security testing, including security early in the process in product demos and creating a continuous peer-to-peer conversation. Use telemetry and observability.*	*Ensure that security team and software engineers are cross-skilling. Allocate time for them to sit and work together to learn from each other. Encourage the documenting and sharing of experiences – good and bad.*

Module 1: DevSecOps Advanced Basics

CALMS & DevSecOps

CULTURE	*All technology teams have accountability for security; security is everybody's job. All understand the end-to-end system and collaborate regularly to create trust.*
AUTOMATION	*Automation helps assure security by strategic use of codifying the orchestration and automation of tasks and processes that have security vulnerabilities when done manually and where automation can enhance security practices.*
LEAN	*Security is not a constraint in the value stream and teams aren't waiting for security activities to happen – flow is optimized. Work is visible through shared backlogs.*
MEASUREMENT	*Cost of breach is understood, business and attack metrics are shared, and a value stream centric approach is followed to optimize cycle time and ensure no delays caused by security.*
SHARING	*Security and software engineers cross-skill and collaborate to automate knowledge. Stories are shared through wikis, standups, and on a day-to-day basis.*

DevSecOps manifestos

"I believe the DevOps movement is a new fertile soil from which the build-security-in concept can be reborn, renamed, and remade."

Larry Maccherone

DEV SEC OPS

SOFTWARE SAFER SOONER

Goal: *Safely distributed security decisions at speed and scale*

VALUES

Build security in *more than bolt it on*

Rely on empowered development teams *more than security specialists*

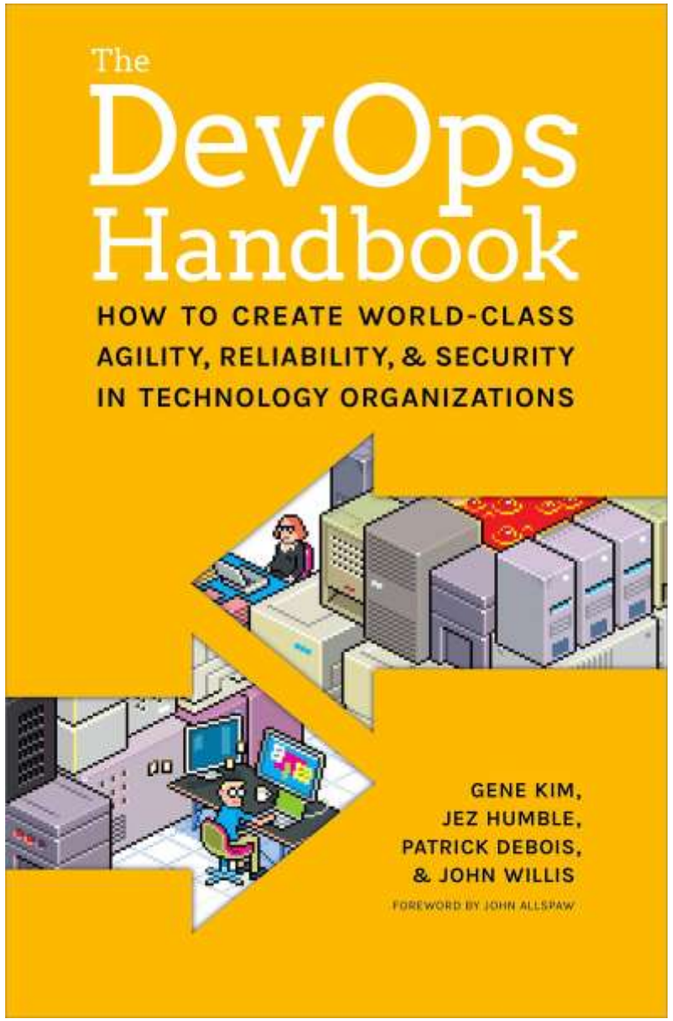
Implement features securely *more than security features*

Use tools as feedback for learning *more than end-of-phase stage gates*

Build on culture change *more than policy enforcement*

"Through Security as Code, we have and will learn that there is simply a better way for security practitioners, like us, to operate and contribute value with less friction. We know we must adapt our ways quickly and foster innovation to ensure data security and privacy issues are not left behind because we were too slow to change."

DevSecOps in the DevOps Handbook



Chapter 22: Information Security as Everyone's Job, Every Day

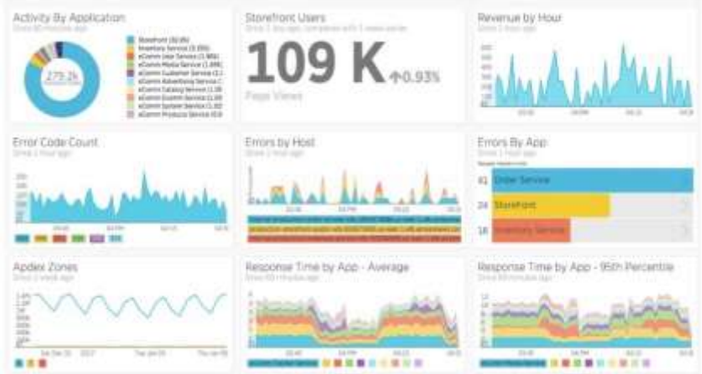
Chapter 23: Protecting the Deployment Pipeline, and Integrating into Change Management and Other Security and Compliance Controls

- *Integrate security into development iteration demonstrations*
- *Integrate security into defect tracking and post-mortems*
- *Integrate preventative security controls into shared source code repositories and shared services*
- *Integrate security into the deployment pipeline*
- *Ensure security of the application*
- *Ensure security of the software supply chain*
- *Ensure security of the environment*
- *Integrate information security into production telemetry*

Team visibility

Dashboards

- *Grouped answers*
- *Accelerates information gathering*



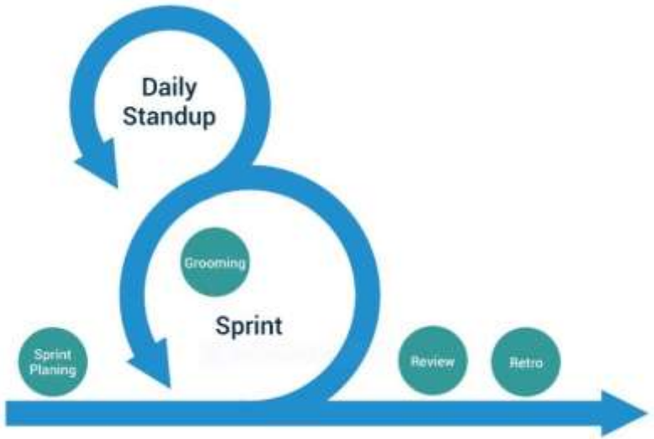
One key to DevOps is creating workflow visibility

Kanban

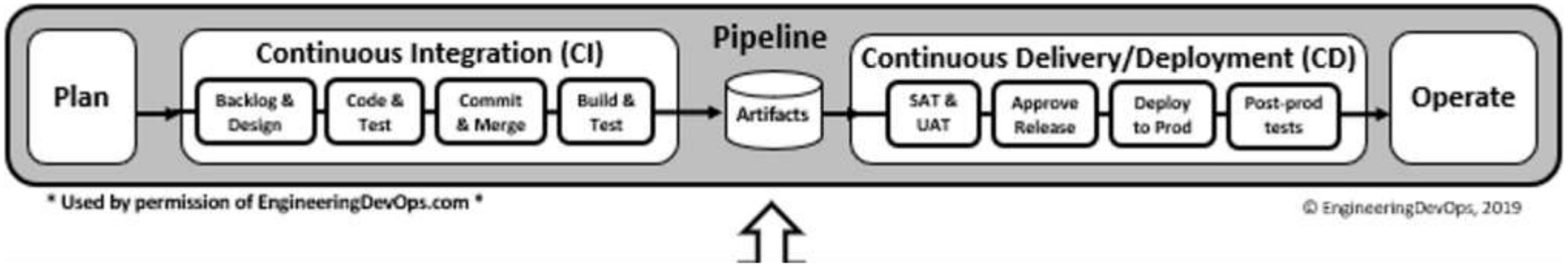
- *From Lean*
- *Visible workflow*
- *Columns*

Scrum

- *Daily standup*
- *Plan, Review, Retrospective*
- *Integration Events*



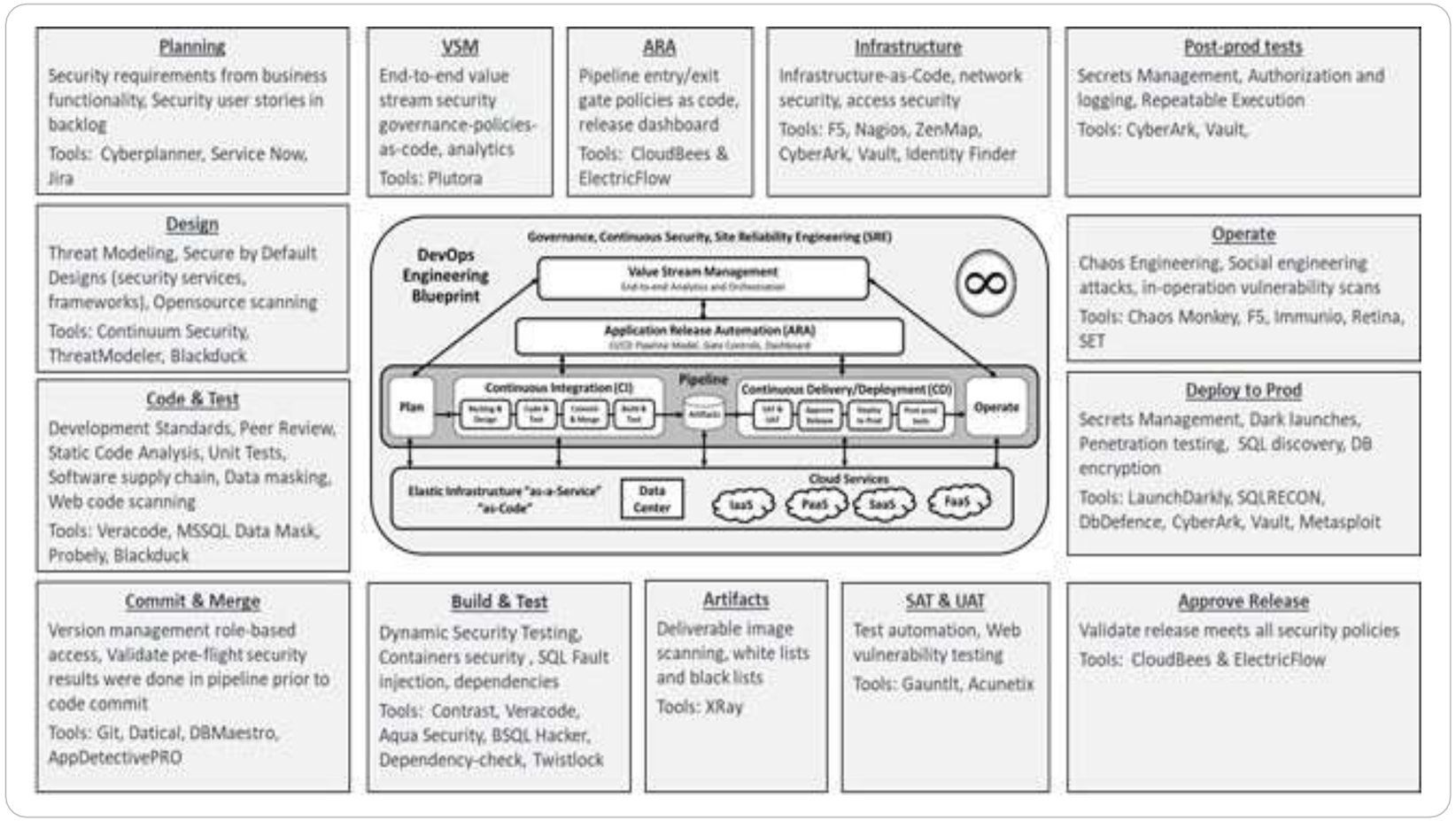
CI/CD Pipeline



- *DevOps built around pipelines*
- *Tools can manage the pipeline*
  - *Gitlab*
  - *Jenkins*
  - *Circle CI*
- *Pipelines integrate multiple tools to check stages including security tools*

- *Pipelines need to be*
  - *Scalable*
  - *Trainable*
  - *Offer support*
  - *Have an integrated user community*
- *Avoid vendor lock-in*

Managing continuous practices and risk



Continuous Monitoring

- *Automated metrics, set thresholds*

Continuous Security

- *Matching security to pipeline events*

Continuous Observability

- *Capable of transparency across activities*

Monitoring, Observability and Security build through DAST, SAST, and RAST tools

Finding the known safe

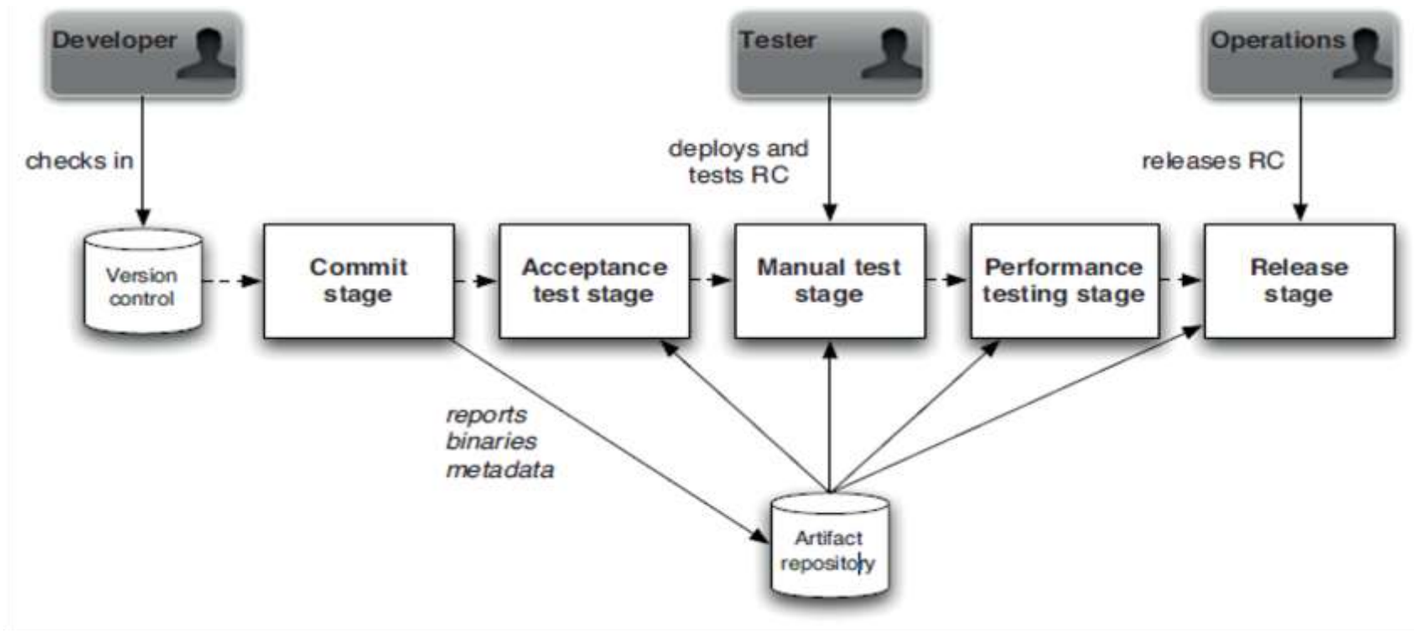
Shared Repos

- *Tested and proven code element*

Open source

- *Matched to known good*
- *Updated by teams*

Creates observability



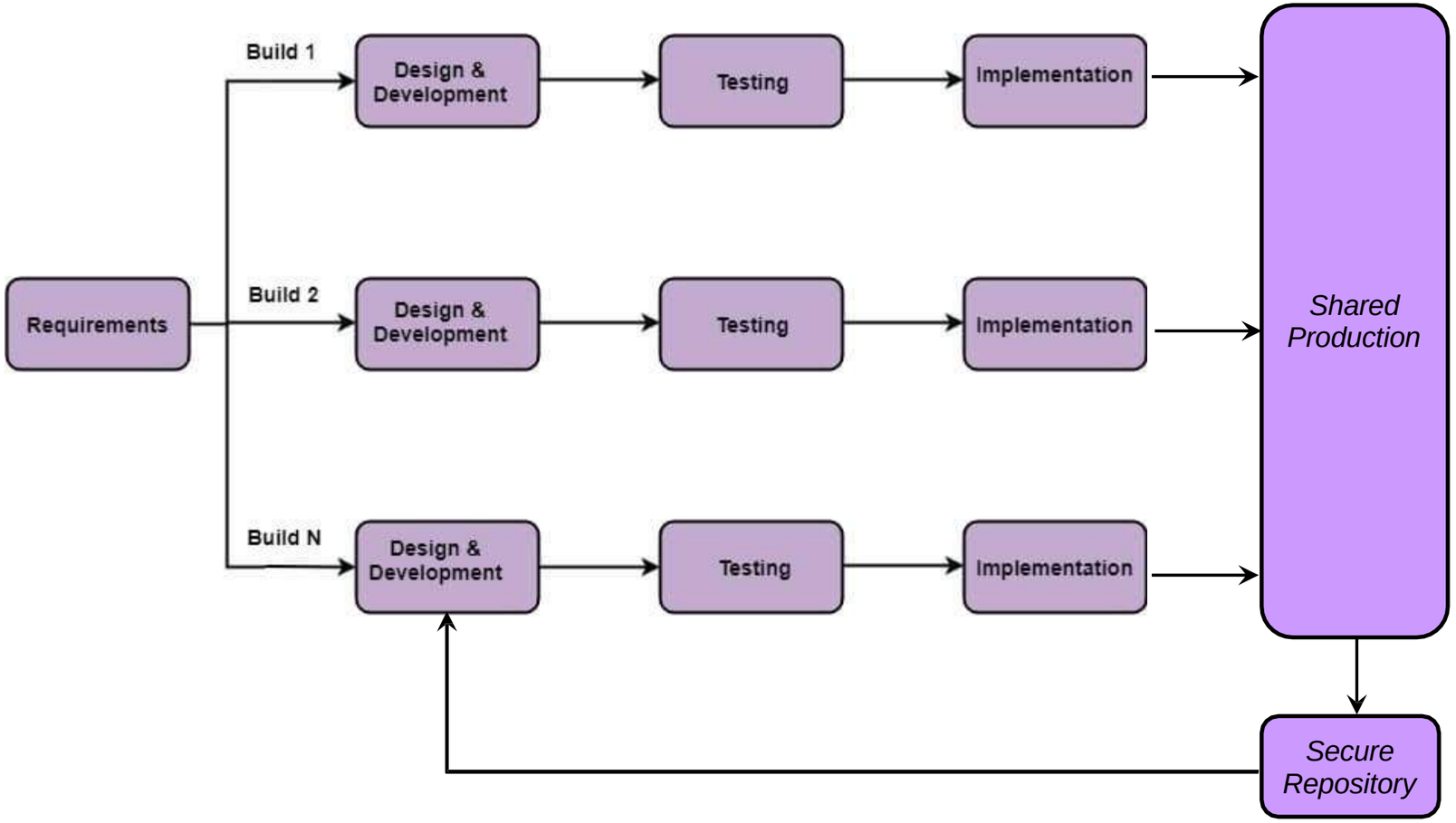
Use an artifact repository to store binaries, reports, and metadata for each of your release candidates (E.g., Archiva, Nexus, JFrog).

Incremental pipeline build

Secure tools for Commit, Build, and Validate

Match to known Repos from commercial teams

Implement vendors to maximize minimal team skills



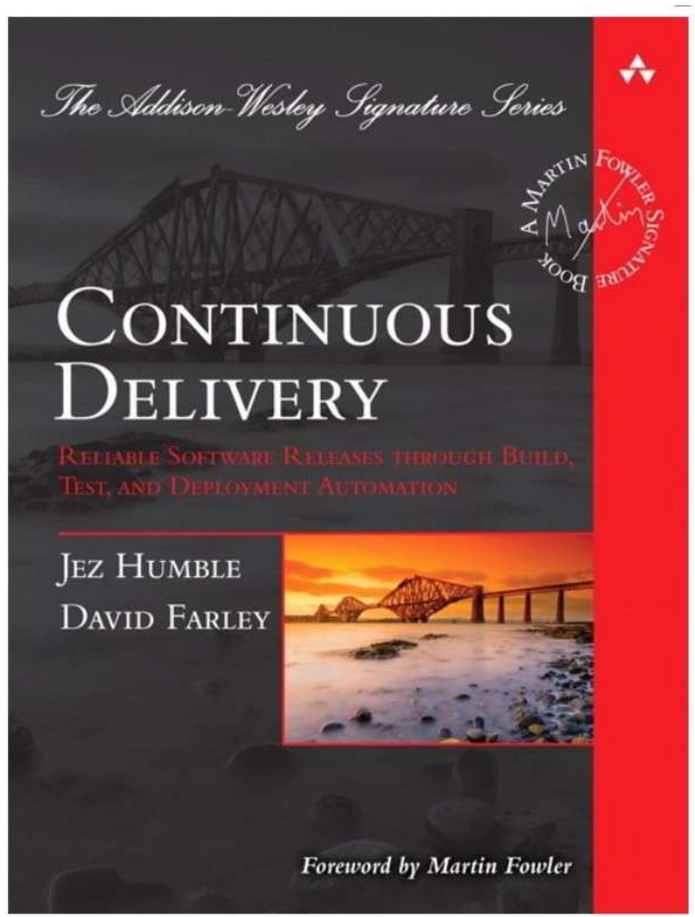
Continuous Delivery

Software in a constantly releasable state

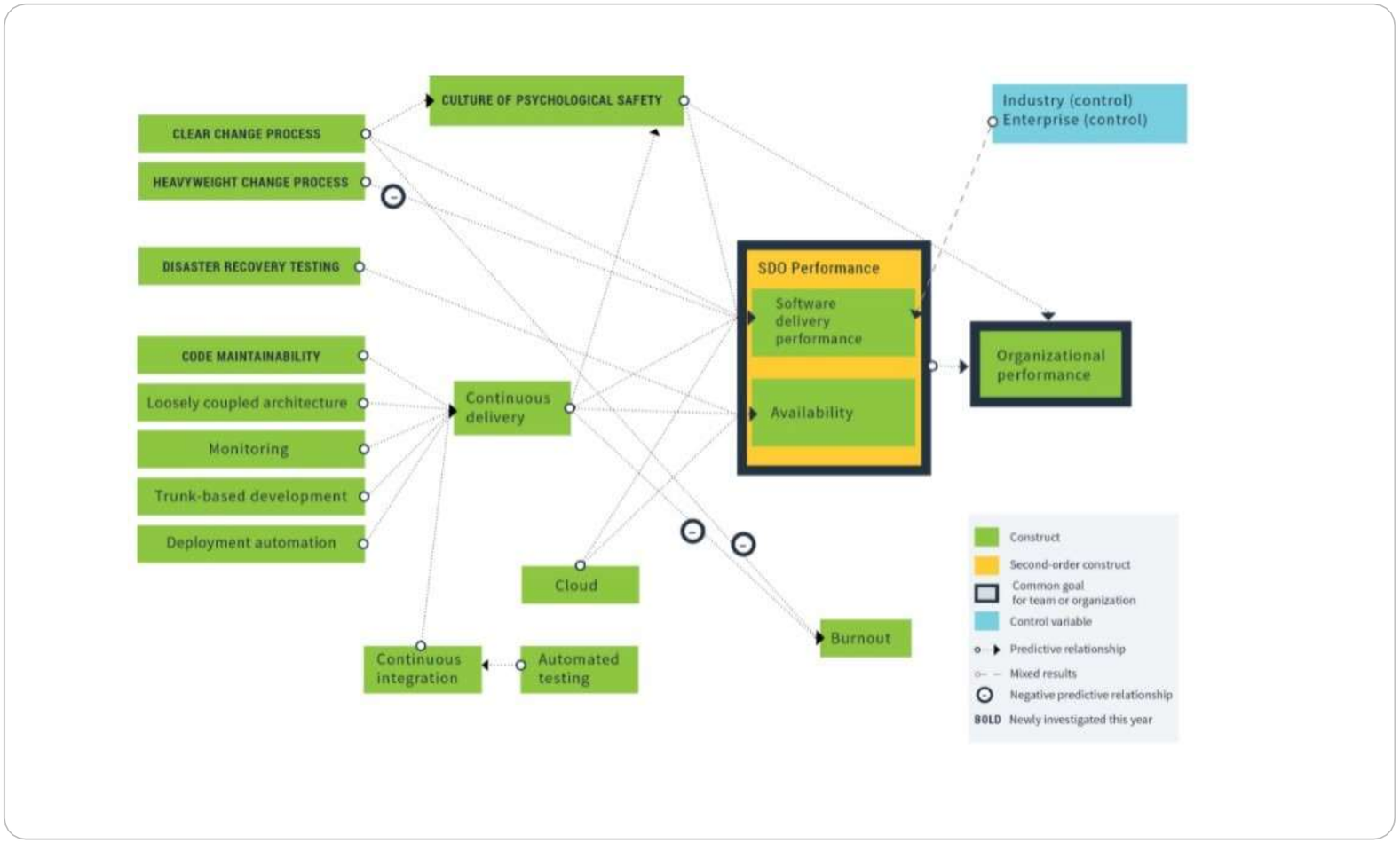
- *Accelerates continuous integration*
- *Provides fast, automated feedback on a system's production-readiness*

Relies on a deployment pipeline enabling deployment on demand

- *Prioritizes keeping software releasable/deployable over new features*
- *Reduces the cost, time, and risk change*



Continuous Deployment Processes



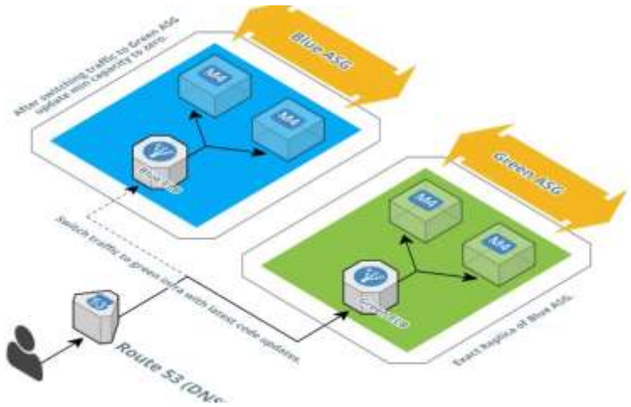
Leads to higher organizational performance

Linkages show dependencies

Continuous Deployment Patterns

Blue/Green

- *Duplicate environment*
- *After test, switch to new format*



A/B

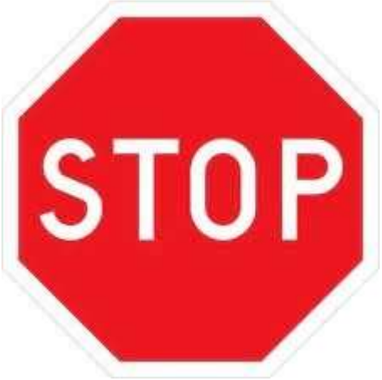
- *Authorized to user subset*
- *Progressively exposed to more users*

Canary

- *Receives traffic percentage*
- *After validation, traffic increased*

Feature Flags

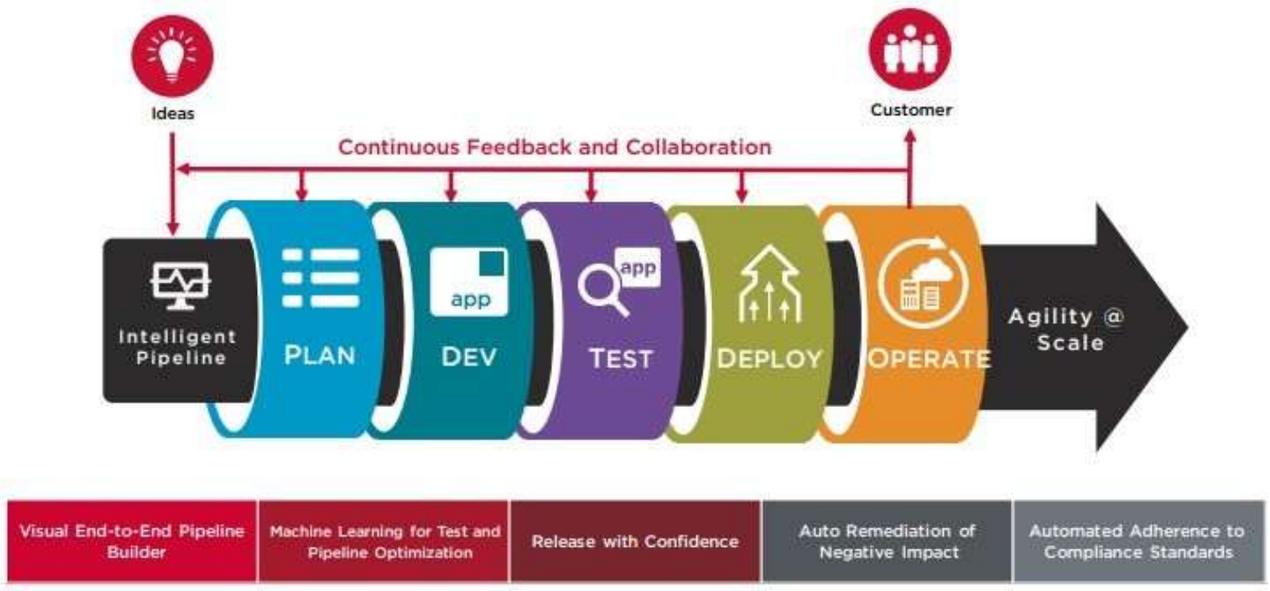
- *New features to production*
- *Used together with A/B testing, Canary, and other deployments*



Security is not a Deployment Pattern!

Securing the Continuous Deployment

- *Continuous deployment requires constant security*
  - *Map threats and secure connections*
  - *Tighten access control*
  - *Separate duties and enforce permissions*
  - *Keep secrets safe*
  - *Lock up your code repository*
  - *Diligently monitor and clean up*
  - *Stay informed and have a plan*
- *Every pipeline step needs variable approach*



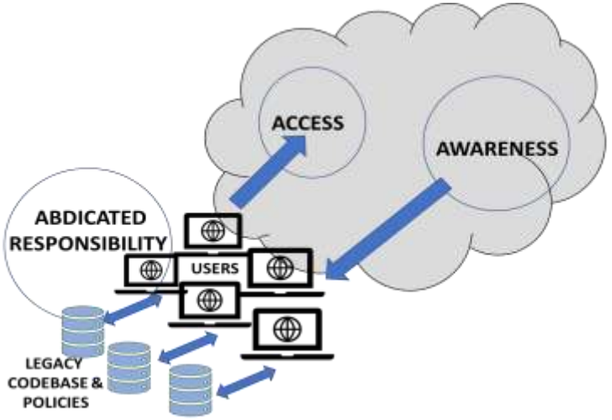
Infrastructure as Code

All underlying process run as code

- *Set up pipeline in multiple places*
- *Easy duplication of test environment*

Allows the 3 A's of Cloud

- *Access (globally)*
- *Awareness (immediately)*
- *Abdicated Responsibility (for security)*



Related to Infrastructure as Service

- *Based on not owning physical architecture*
- *Provided processing, storage, and networks*
- *Software up to customer*

Building from the bottom

Evaluate current needs

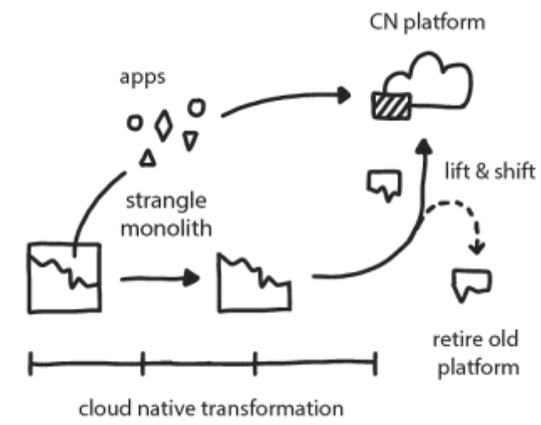
- *On-premise*
- *Cloud-using*
- *Cloud-native*
- *Hybrid*

Migrating architecture

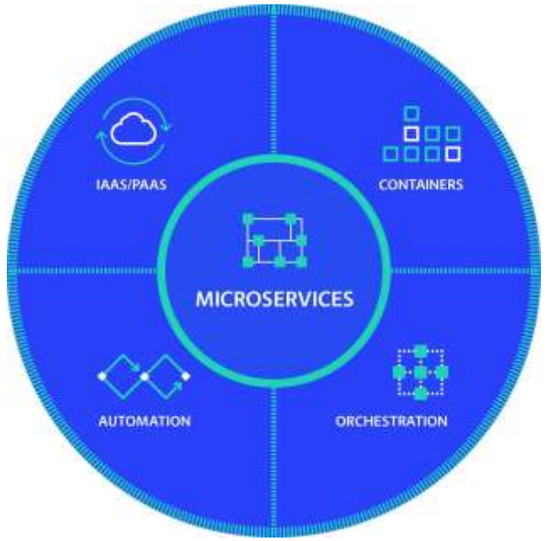
- *On-premise to cloud native*
- *Monolith to microservices*
- *Cloud-native to cloud-based*

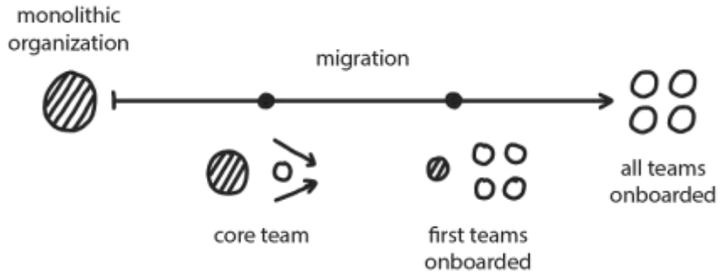
Strangler Pattern

- *Apps*
- *Orgs*



For design options in pattern: cnpatterns.org





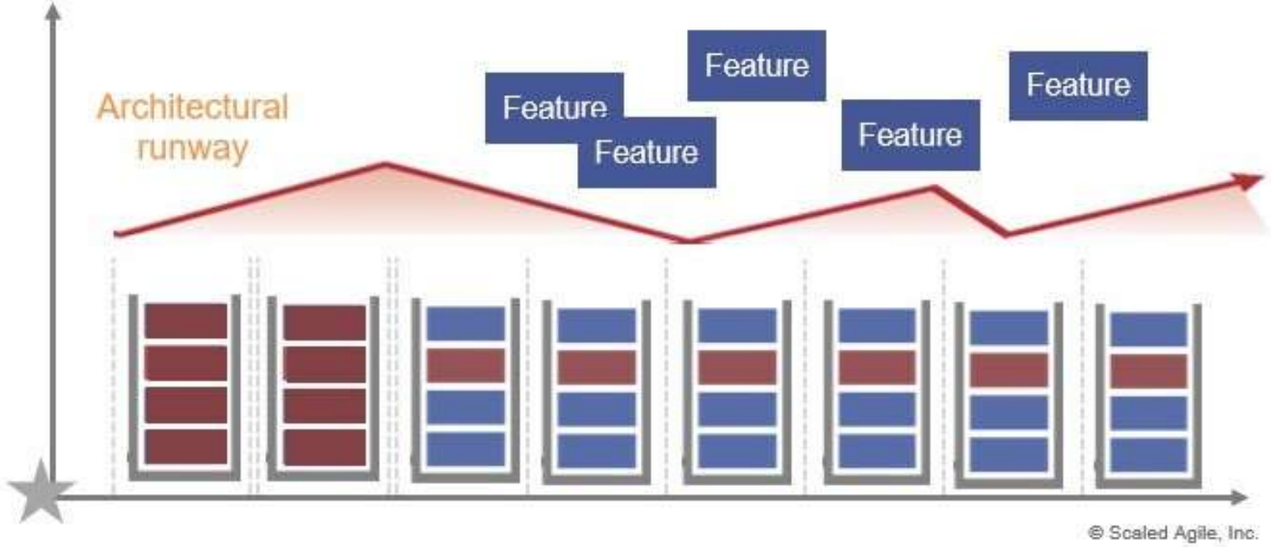
Architectural runway

Runways are for landing!

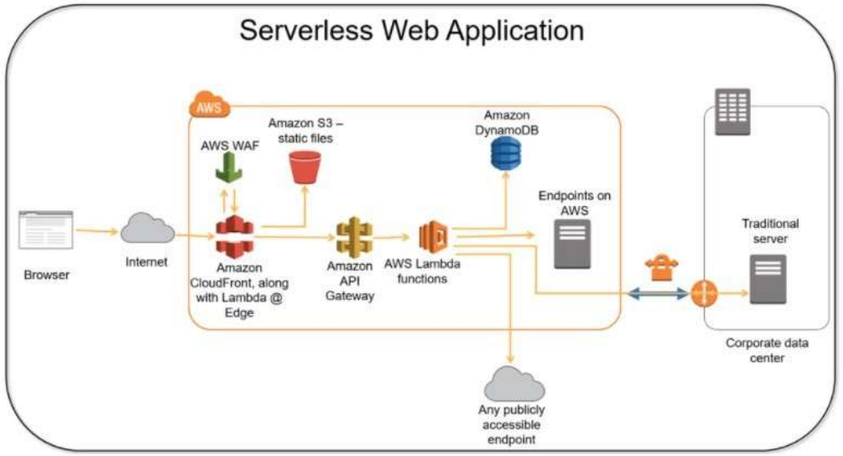
- *Articulates dev timeframes*
- *Allows visual link to features*
- *Do you have security features*

Designing a runway

- *Identify iterations*
- *Highlight key features*
- *Track dependencies*
- *Group consensus*



Server-less architecture



- *Cloud computing model where resources are allocated on demand*
  - *Servers are centrally modified*
  - *Not owned by primary customary*
  - *Can accelerate code for DevOps*
  - *Backend between event-driven architectures*
- *Examples*
  - *AWS Lambda*
- *Pros – Cost, Elasticity vs. Scalability, Productivity*
- *Cons – Performance, resource limits, monitoring and debugging, security, privacy, vendor lock-in*

Stages for testing

DEVELOPMENT	INTEGRATION	DELIVERY	DEPLOYMENT
Guarantee code quality *Catch issues early*	*Validate service*	*Final gate before production*	*Testing in Production*
Unit testing	*Data driven testing*	*Functional Tests*	*Runtime security tests*
Code linting	*End-to-End*	*Dynamic Security Tests*	*Metrics, traces and log analysis*
Static analysis	*Regression*	*Blue/Red Teaming*	*Reversion Tests*
Code reviews	*Performance*		

Testing and feedback

Feedback requires monitoring

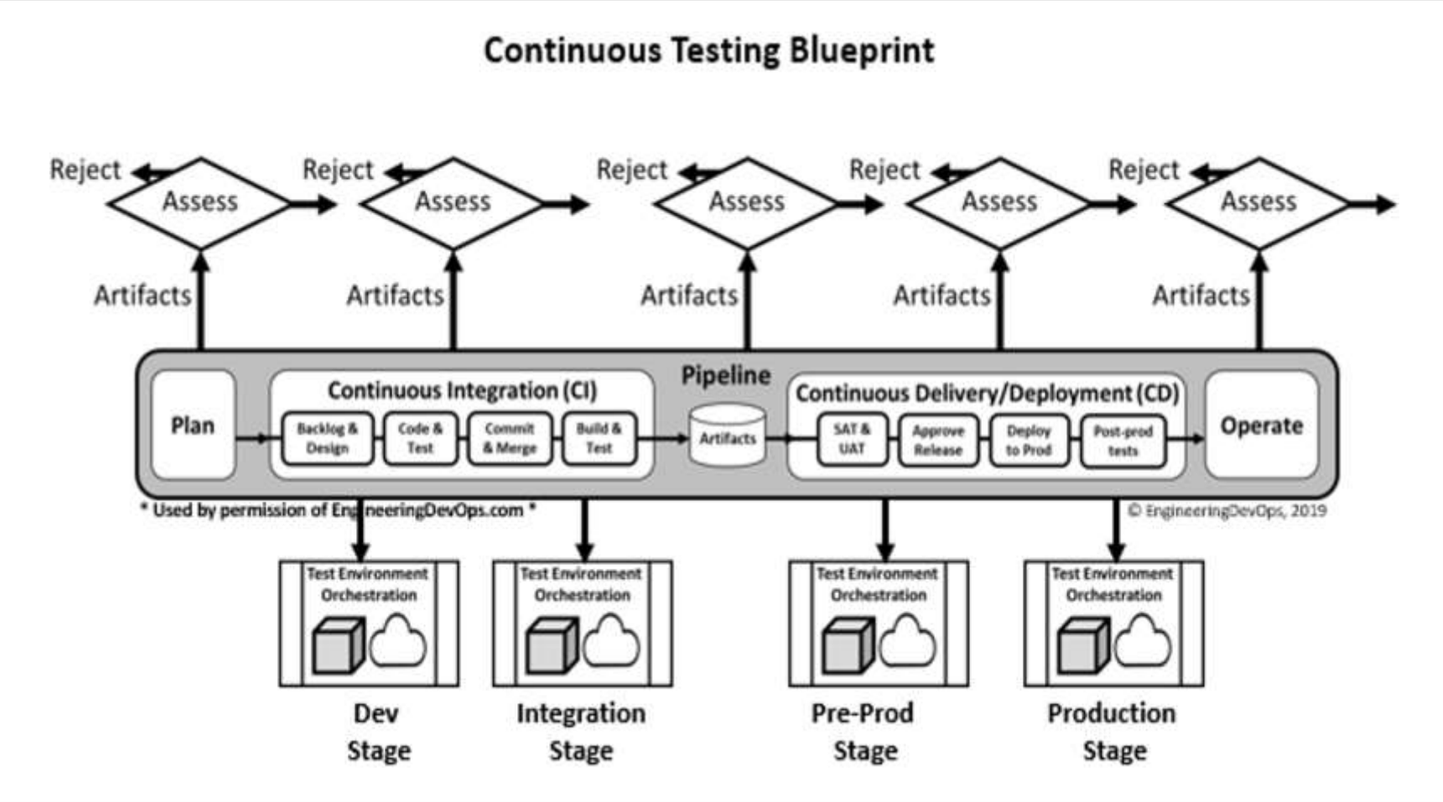
- *Monitoring implies testing*
- *Testing for feedback creates artifacts*

Testing looks for set barriers

- *Telemetry captures status*
- *Metrics, logs, traces might not be testing*

How do you report test results?

- *Automated security gates*
- *Shared dashboards*
- *Weekly staff meetings*



Used with permission from EngineeringDevOps.com

Design the test case

What do you want to test?

What is the expected answer?

What happens if test fails?

Checking previous tests

Find security in building test cases

- *Vulnerability Tests*
- *Compliance Tests*
- *Personal Tests*

1. *Designing a test case*
2. *Analyze requirements*
3. *Set up a test environment*
4. *Analyze software/hardware needs*
5. *List how systems should respond*
6. *List testing methods*
7. *Design test cases*
8. *Run tests, study, save results*

Test Case Type	Description	Test Step	Expected Result	Status
Functionality	*Area should accommodate up to 20 characters*	*Input up to 20 characters*	*All 20 characters in the request should be appropriate*	*Pass or Fail*
Security	*Verify password rules are working*	*Create a new password in accordance with rules*	*The user's password will be accepted if it adheres to the rules*	*Pass or Fail*
Usability	*Ensure all links are working properly*	*Have users click on various links on the page*	*Links will take users to another web page according to the on-page URL*	*Pass or Fail*

Common errors and solutions

Check with integration with tools

Look for more fidelity in results

Don't debug immediately – gather results and fix the whole

Investigate root causes

Resolve the problem and test again

- *Quick resolutions might not fix all the issues*

Capture the data, and get help

		Truth about the population	
		H_0 true	H_a true
Decision based on sample	Reject H_0	Type I error	Correct decision
	Accept H_0	Correct decision	Type II error

Observable, observing, observability

Observable: *functions designed to produce data in manipulatable formats*

Observing: *the identification of persons or tools who interact with data at specified points*

Observability: *the state of observing outputs from tools and functions*

Relating things to other things

≠

≠

Smooth Skin	Rough Skin	Rough Skin
Round	Round	Round at parts
Navel with stem	Navel with stem	Navel with stem
Juicy on inside	Juicy on inside	Juicy on inside

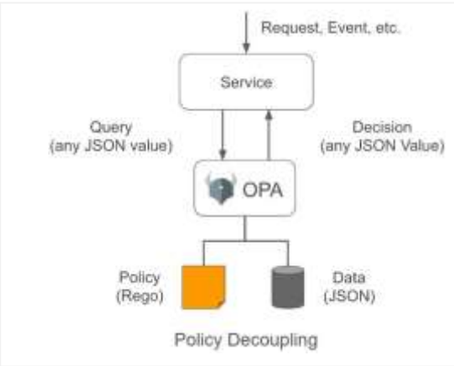
Open policy agent

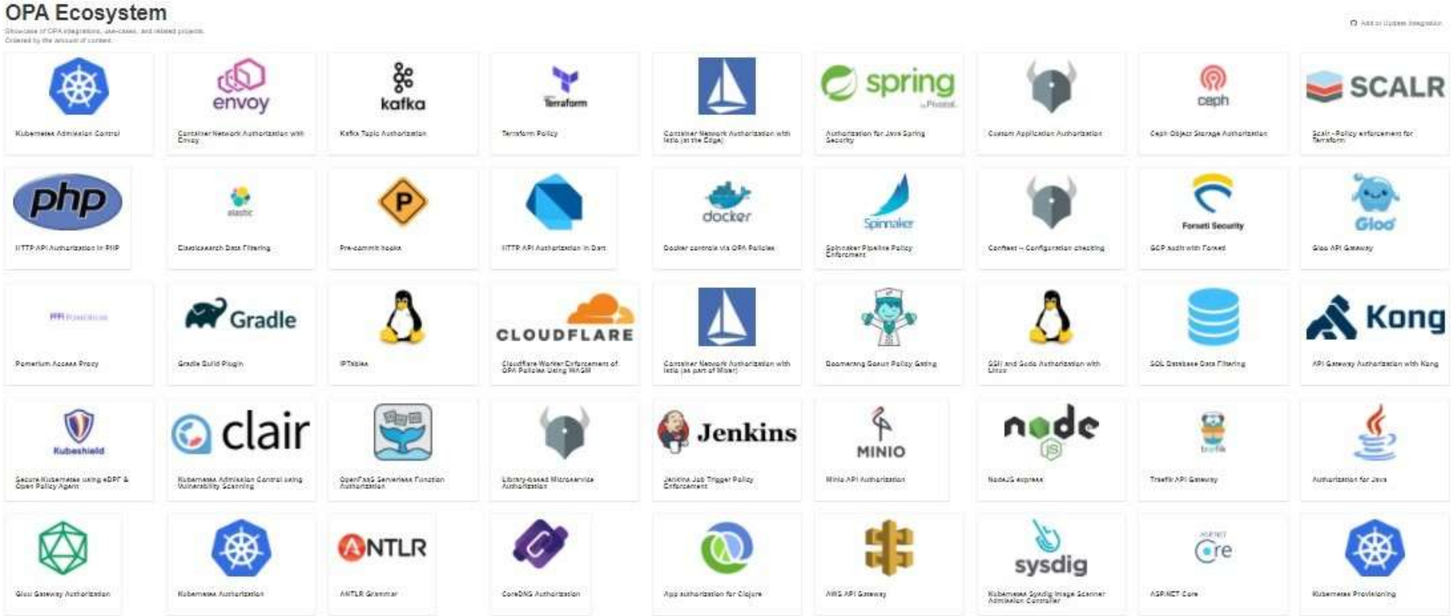
Open source, policy engine with high-level declarative language (Rego)

Sidecar service to verify policies and configuration in:

- *Microservices*
- *CI/CD Pipelines*
- *API gateways*

Decouples decision-making from enforcement





Observing security

Security requires observation similar to other pipeline aspects

Question teams about security

Decouple decision-making from enforcement

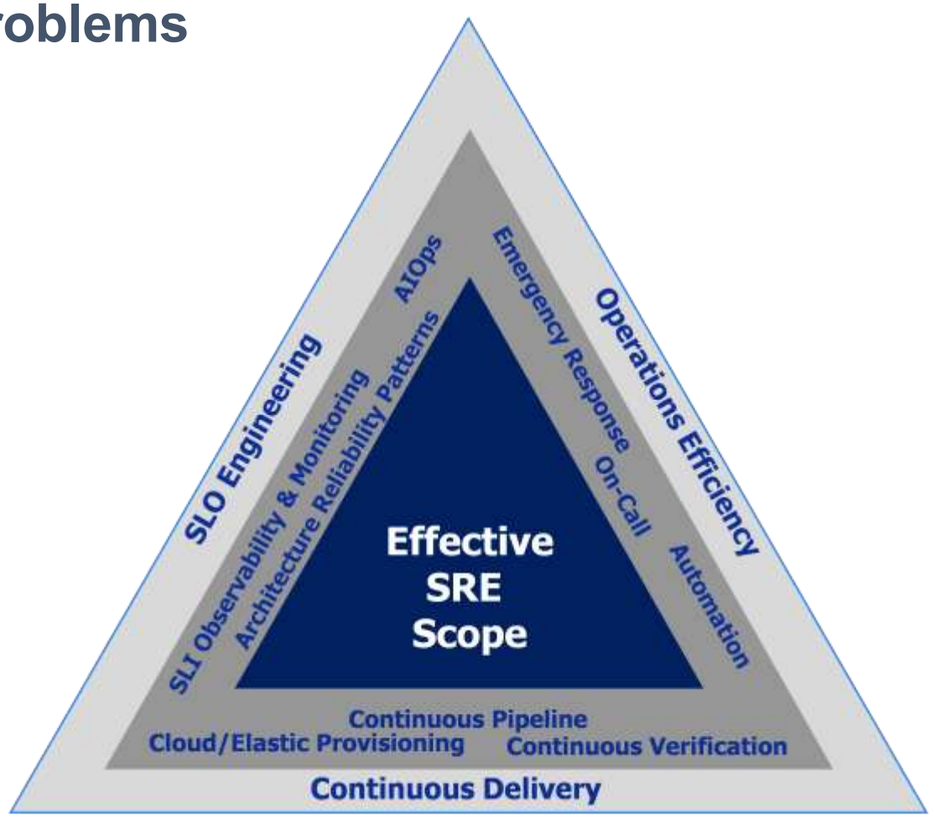
Security questions for teams

- *What are my vulnerabilities?*
- *What are the risks of these?*
- *How do I fix it?*
- *How long will that take?*
- *How can you help me fix it?*
- *How do we make sure the same problem isn't anywhere else?*
- *Why should I care?*
- *Why should we tell?*
- *Where can we record this?*
- *Can we automate this?*
- *How do we stop this happening again?*

Establishing a secure SRE

The System Reliability Engineer works end to end problems

- *Break glass for the DevOps team*
- *Based on three factors*
  - *Service Level Agreement (SLA)*
  - *Service Level Objective (SLO)*
  - *Service Level Indicator (SLI)*
- *Not always trained on security issues*
- *Ensure factors include security*



SRE values vs. security

SRE

Keep the site running

- *Isolate failure domains*
- *Redundant systems*
- *Load balancing*

Empower dev teams with distributed decisions

Approach ops as engineering problem

Achieve business success through measured promises

Security

Security doesn't stop ops

- *Find problems*
- *Are redundancies secure?*
- *Same as above*

Take part in deployment decisions

Approach pipeline as security solution

Achieve business success through measured promises

SRE enabling functions vs security

SRE

Monitoring, metrics, KPIs

Incident management and emergency response

Capacity planning and demand forecasts

Performance analysis and optimization

Provisioning, change management, and velocity

Security

Transparency in metrics

Ensure crown jewels safely stored

- *Are alternate sites secure?*
- *What version is backed up?*

Can current security scale?

Do security practices affect performance?

- *Do you know?*
- *Do you know the margins*

Prepare for change

SRE antipatterns and security takeaways

SRE

Humans staring at screens

Mob incident response

Magic solutions

Hiring a dog-walker for pets

Speed-bumps and production postponement

Ungainly governance

Security

Observability demands automation

When a flaw exists, what happens next?

Who can fix security? Team efforts

Minimize management software – don' just add security layers

DevSecOps, like DevOps, based on acceleration. Find the pain points

If you can't explain the security guidelines, no one else can either

Dashboard shortcut security discussions

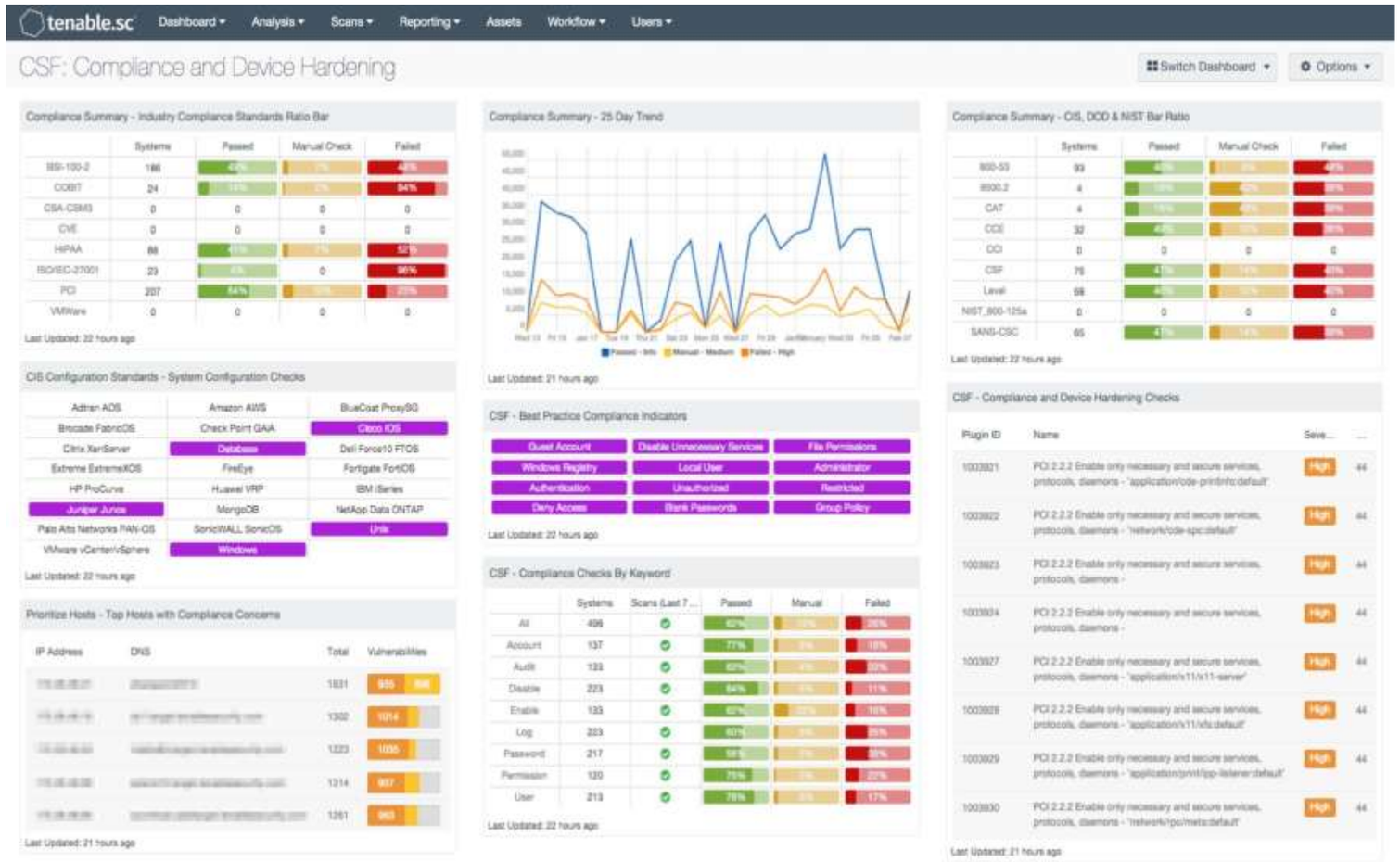
Enhanced visibility improves flow

Observability creates awareness

- *Metrics*
- *Logs*
- *Traces*

Value question

- *What do you learn from dashboard?*
- *How does it create action?*



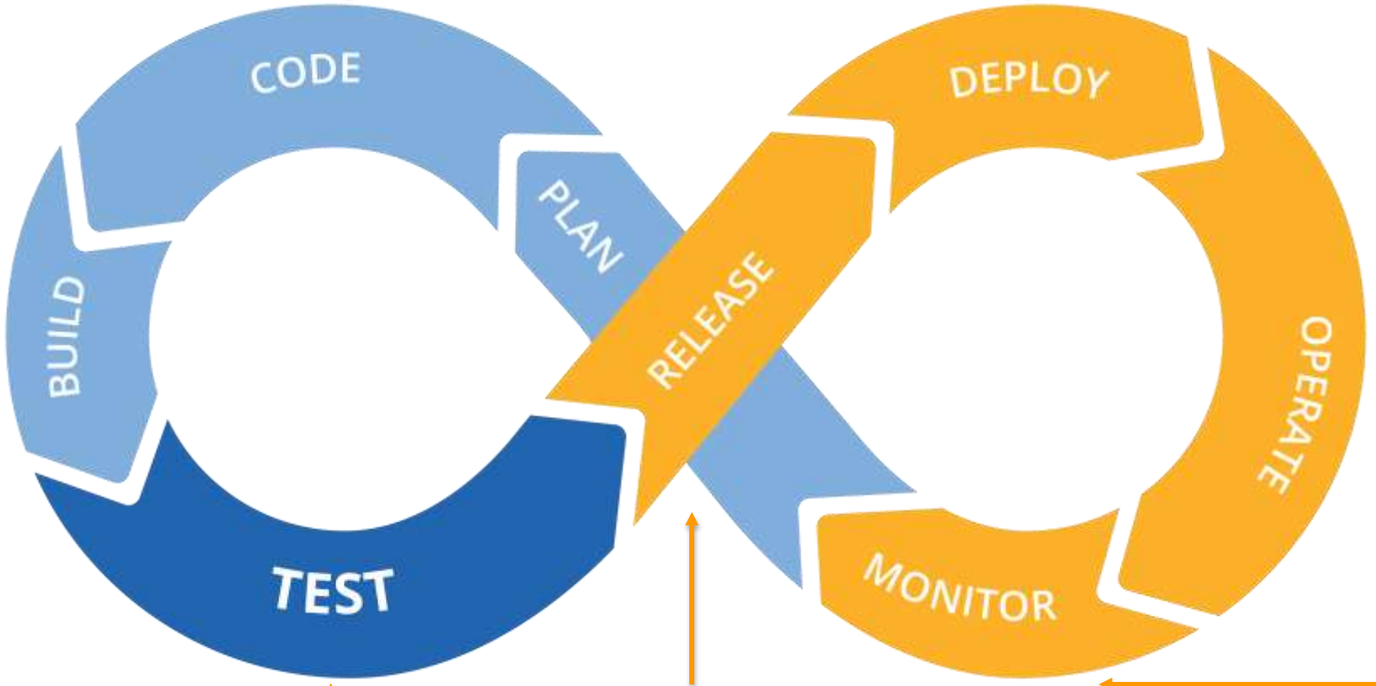
Where do I go from here?

1	*Decision creates error*	*non-decision invites disaster*
2	*2nd Law, Thermodynamics*	*System entropy always increases*
3	*Know the base rate for decision*	*When growth stops, model the best tools to accelerate value*

DevSecOps (security hooks in the pipeline)

Secret detection
Leak prevention

Architectural Guardrails
Build Compliance
Vulnerability Assessment



Static Code Analysis
Compliance of Libraries used

Secrets management
Enabling security monitoring
Enabling compliance checks

Dynamic Code Analysis
Vulnerability Testing
Penetration Testing

Compliance Checks
Configuration settings
Secrets and Keys
Secure network design
Well architected

Vulnerability Assessment
Penetration Tests
Compliance Checks
Behaviour Anomalies

Why Operating models are critical for cloud adoption?

Traditional operating model—simplified

Business Software

Custom developed or commercial off-the-shelf

Infrastructure

Compute	Runtime
Network	Data
Storage	Operations
Middleware	Security

	Engineering	Operations
Applications	Applications Engineering	Applications Operations
Infrastructure	Infrastructure Engineering	Infrastructure Operations

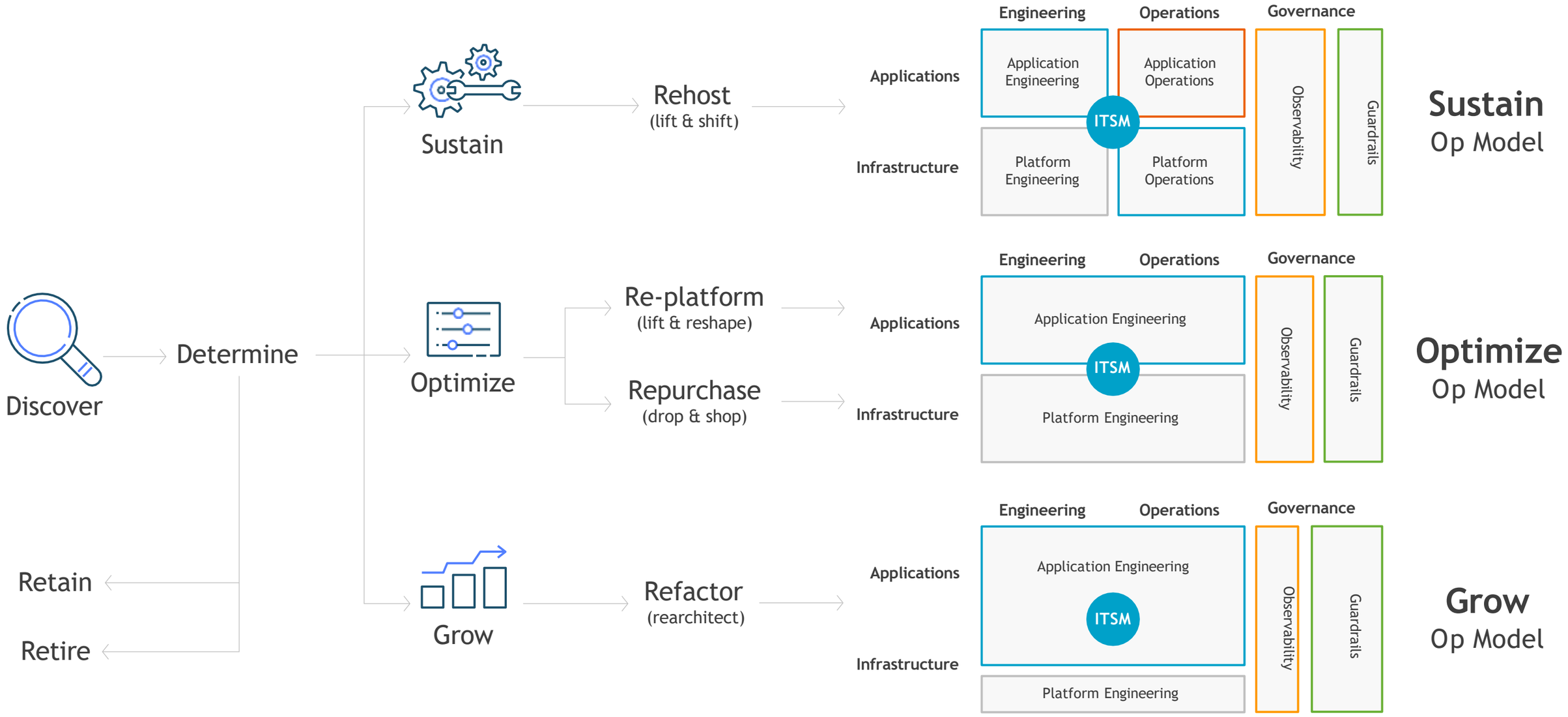
Develop, Build, and Test

All activities needed to define and validate Platform Infrastructure or Business Applications

Deploy, Operate, and Manage

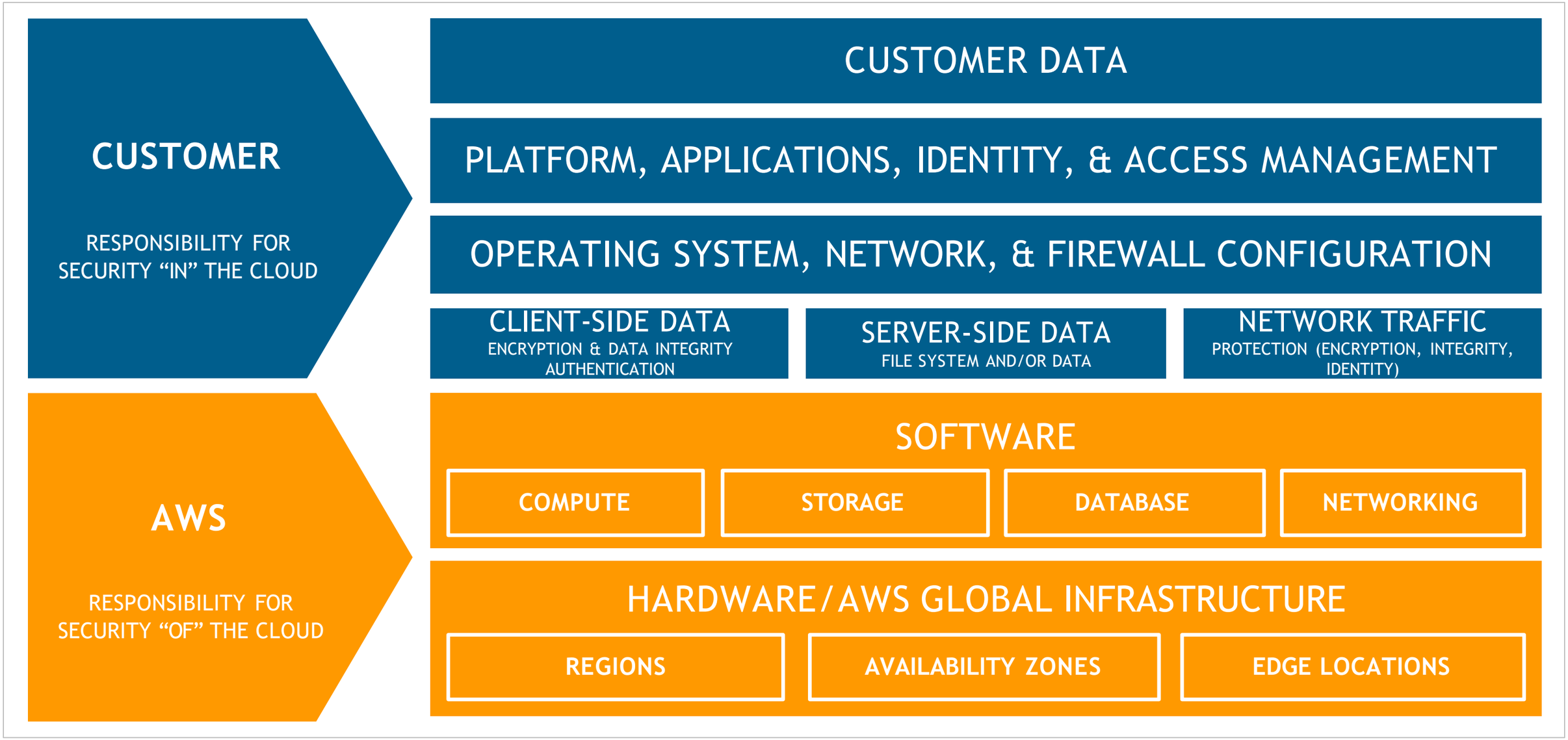
All activities needed to deploy and support platform infrastructure and applications in production

How does this impact cloud adoption?

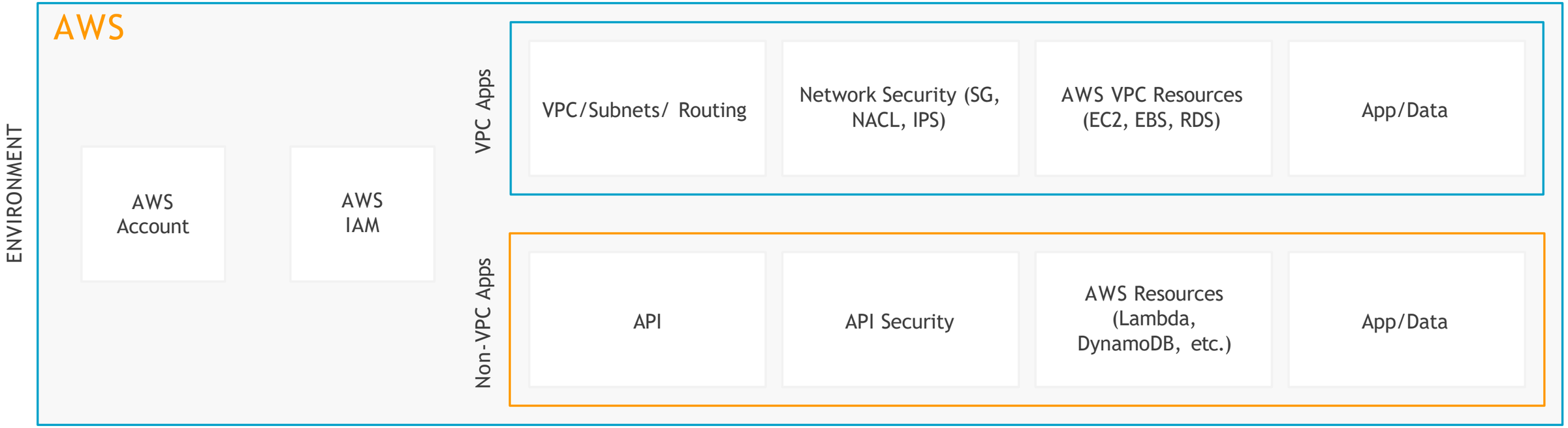


Adopting to cloud responsibilities

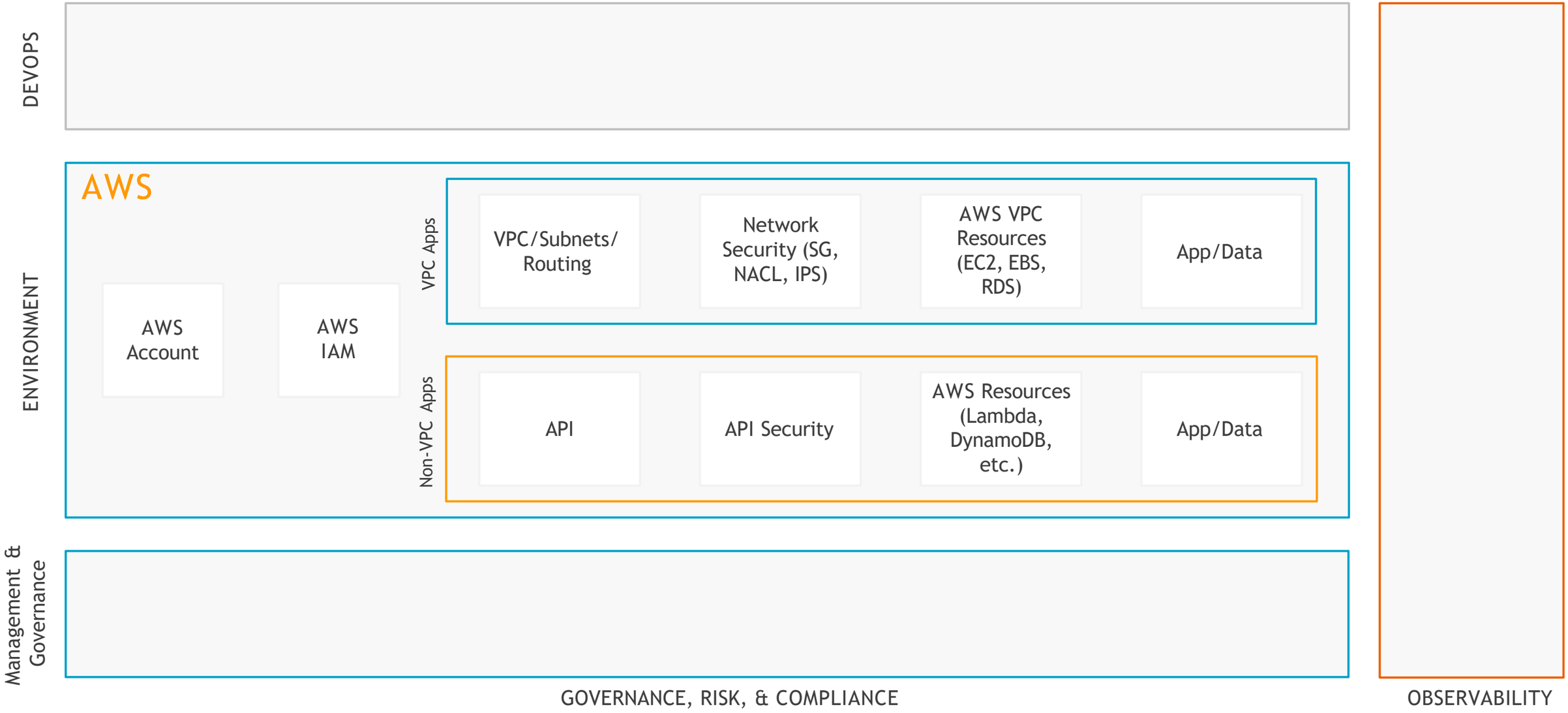
Understanding customer responsibility is key



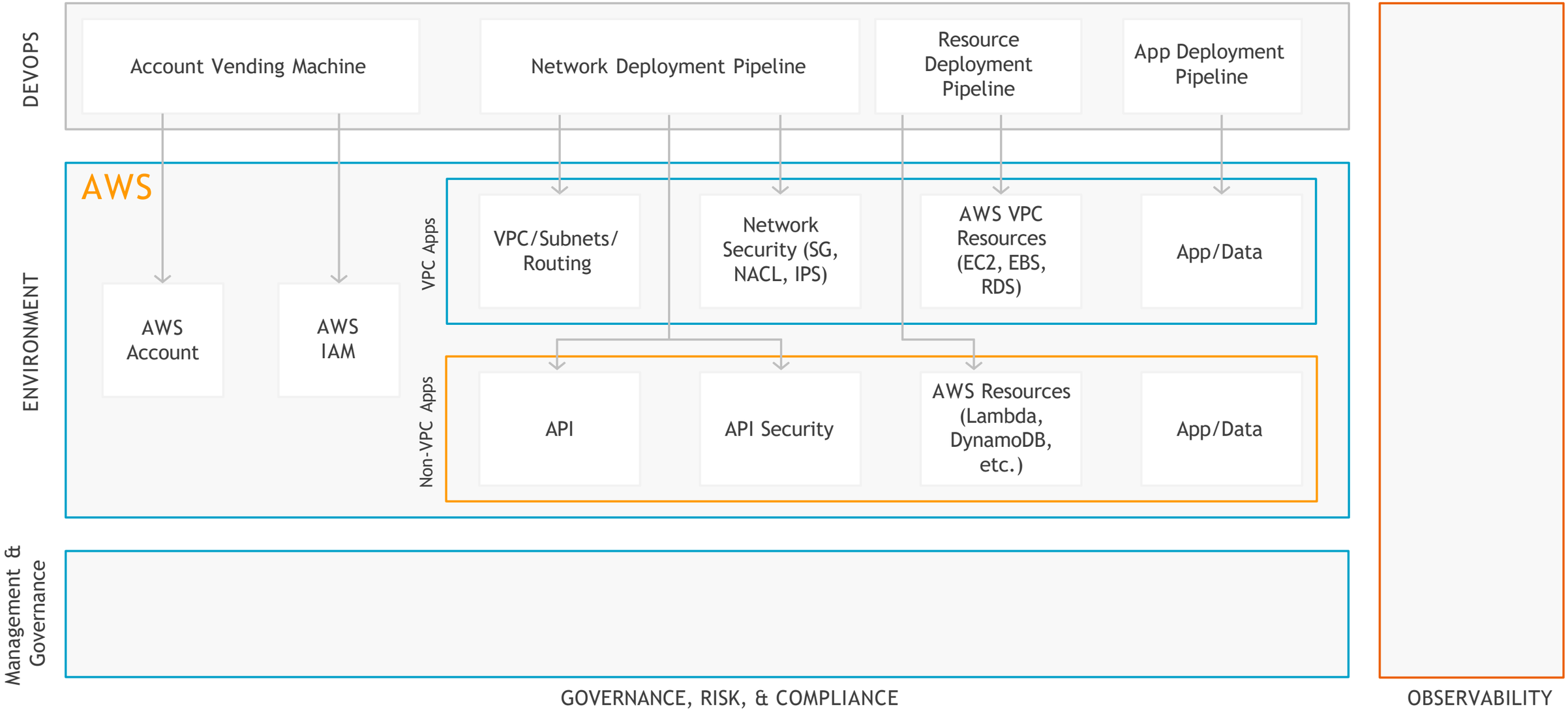
DevOps, security, and observability for AWS



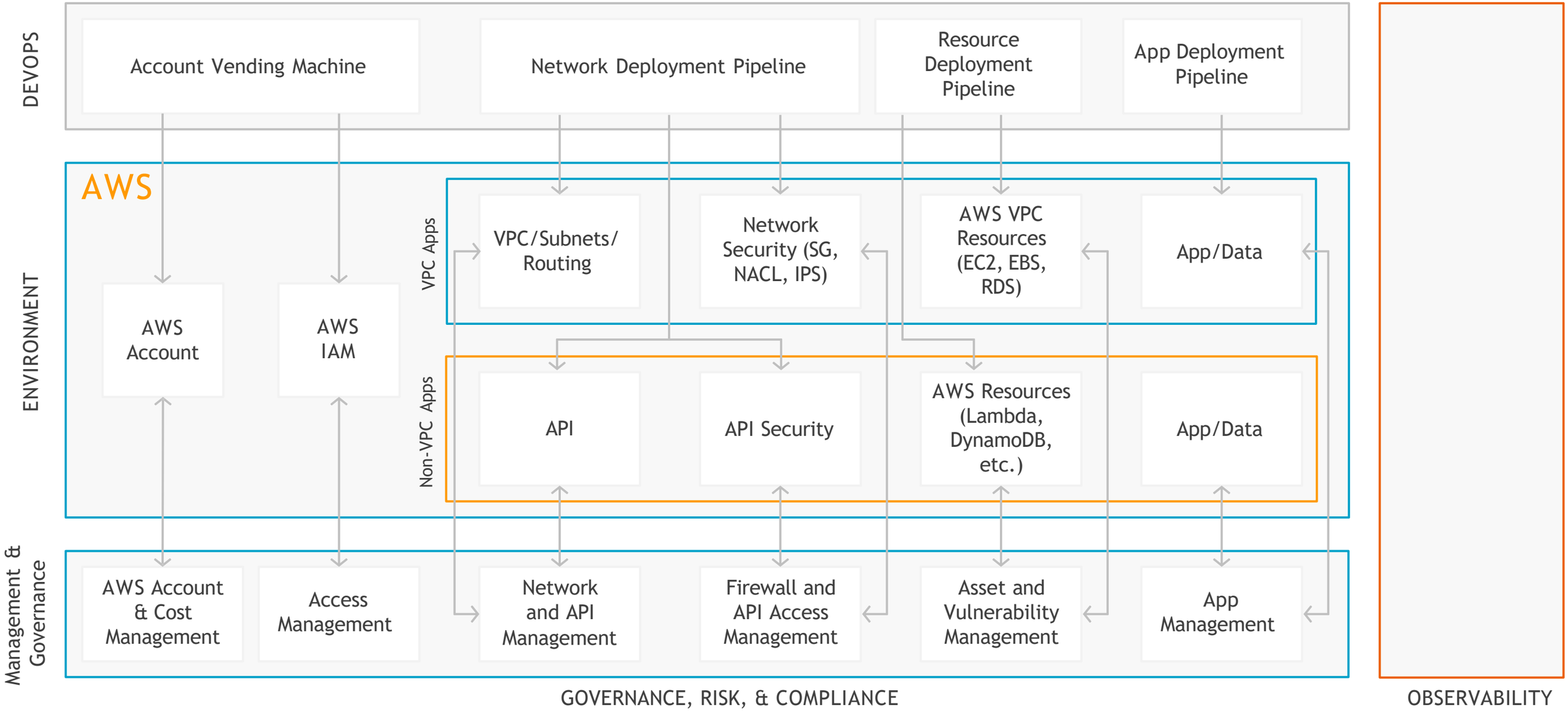
DevOps, security, and observability for AWS



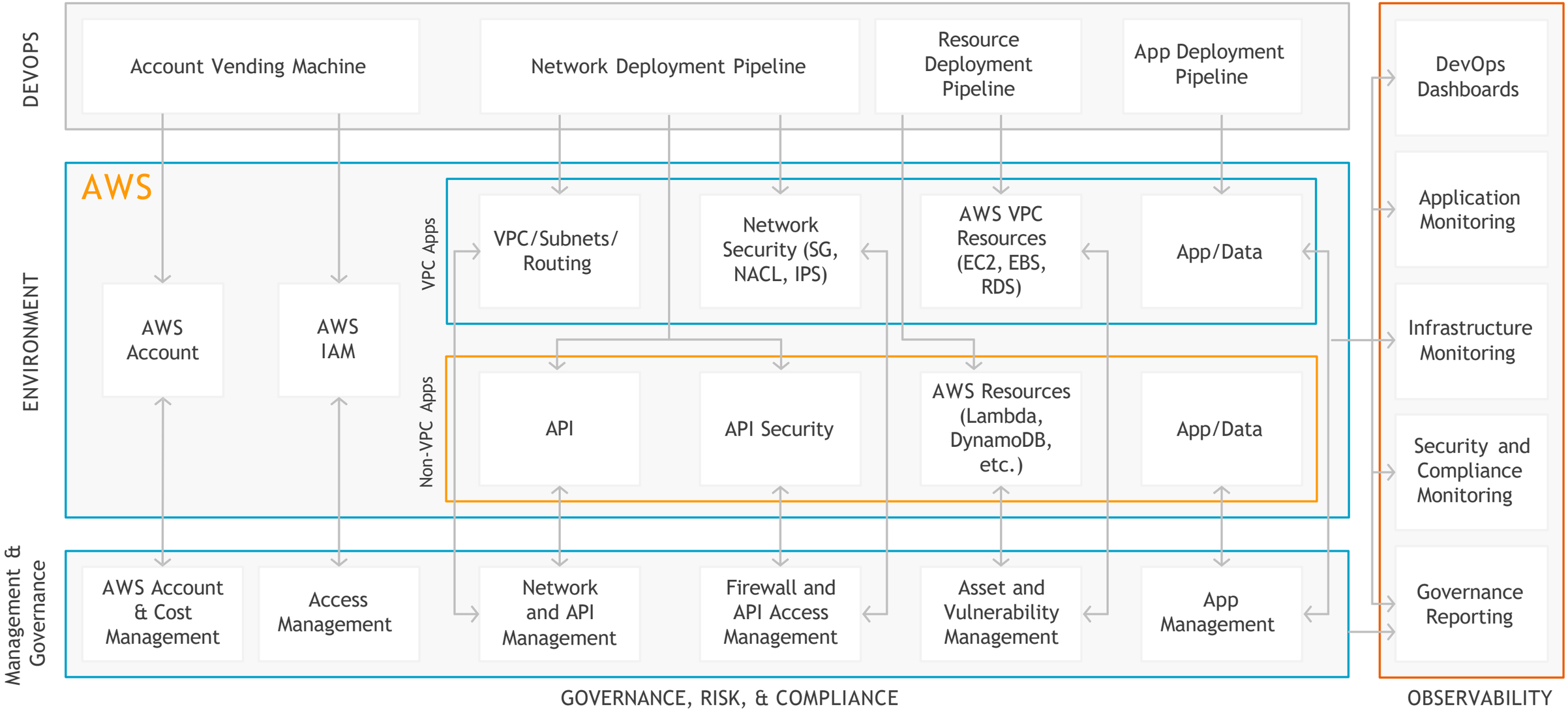
DevOps, security, and observability for AWS



DevOps, security, and observability for AWS

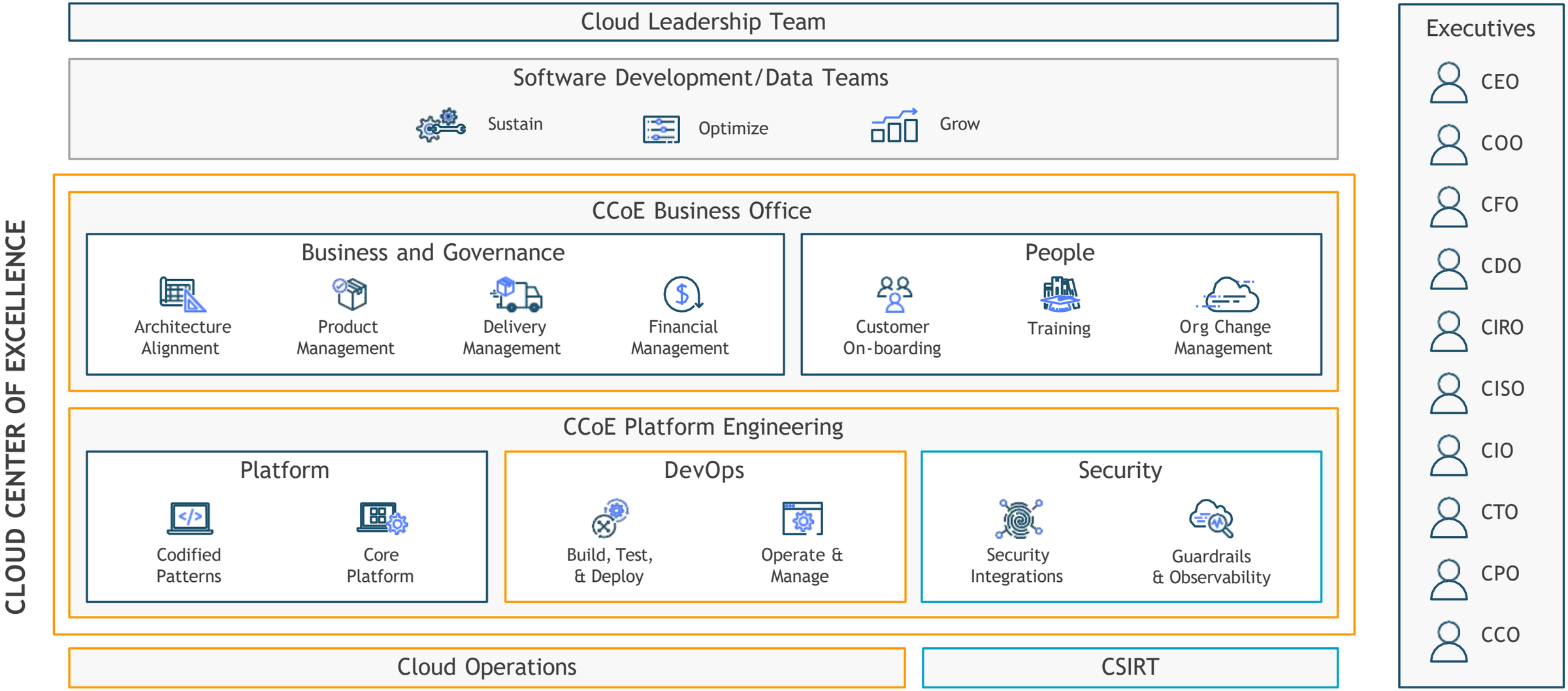


DevOps, security, and observability for AWS



Executing DevSecOps with Cloud Center of Excellence (CCoE)

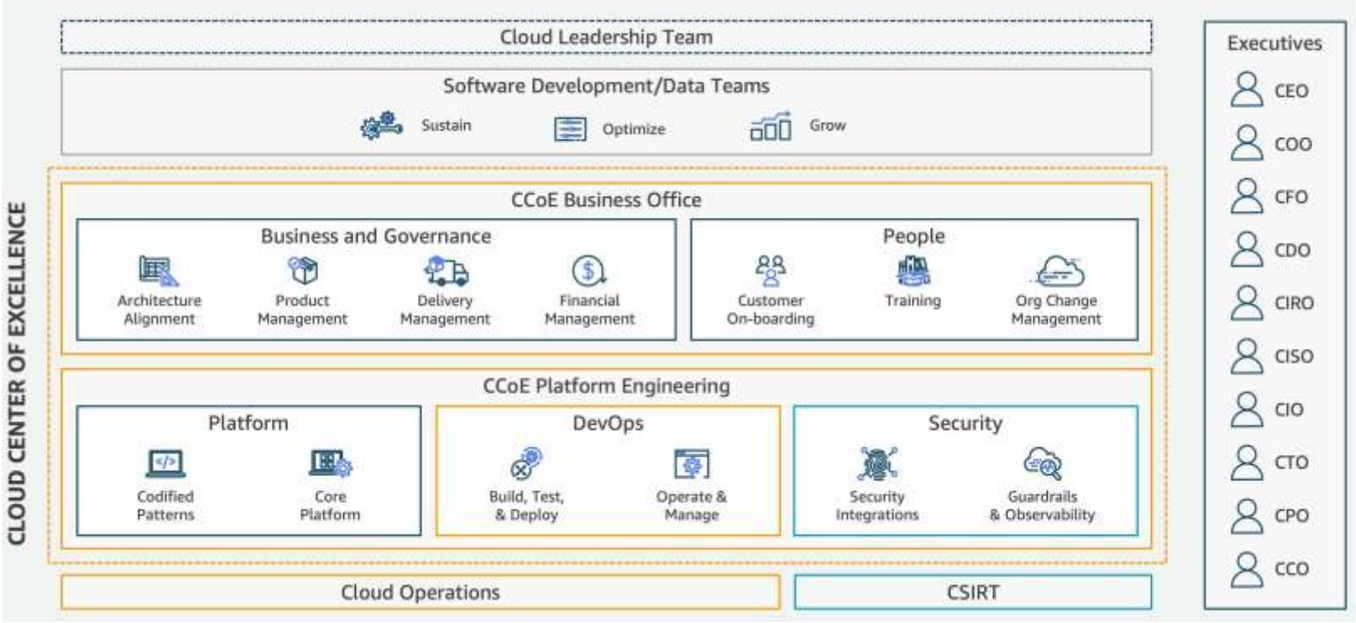
Organizing the Cloud Center of Excellence (CCoE)



Enabling cross-functional teams

DEVOPS

- Account Vending Machine
- Network Deployment Pipeline
- Resource Deployment Pipeline
- App Deployment Pipeline



GRC

- AWS Account & Cost Management
- Access Management
- Network and API Management
- Firewall and API Access Management
- Asset and Vulnerability Management
- App Management

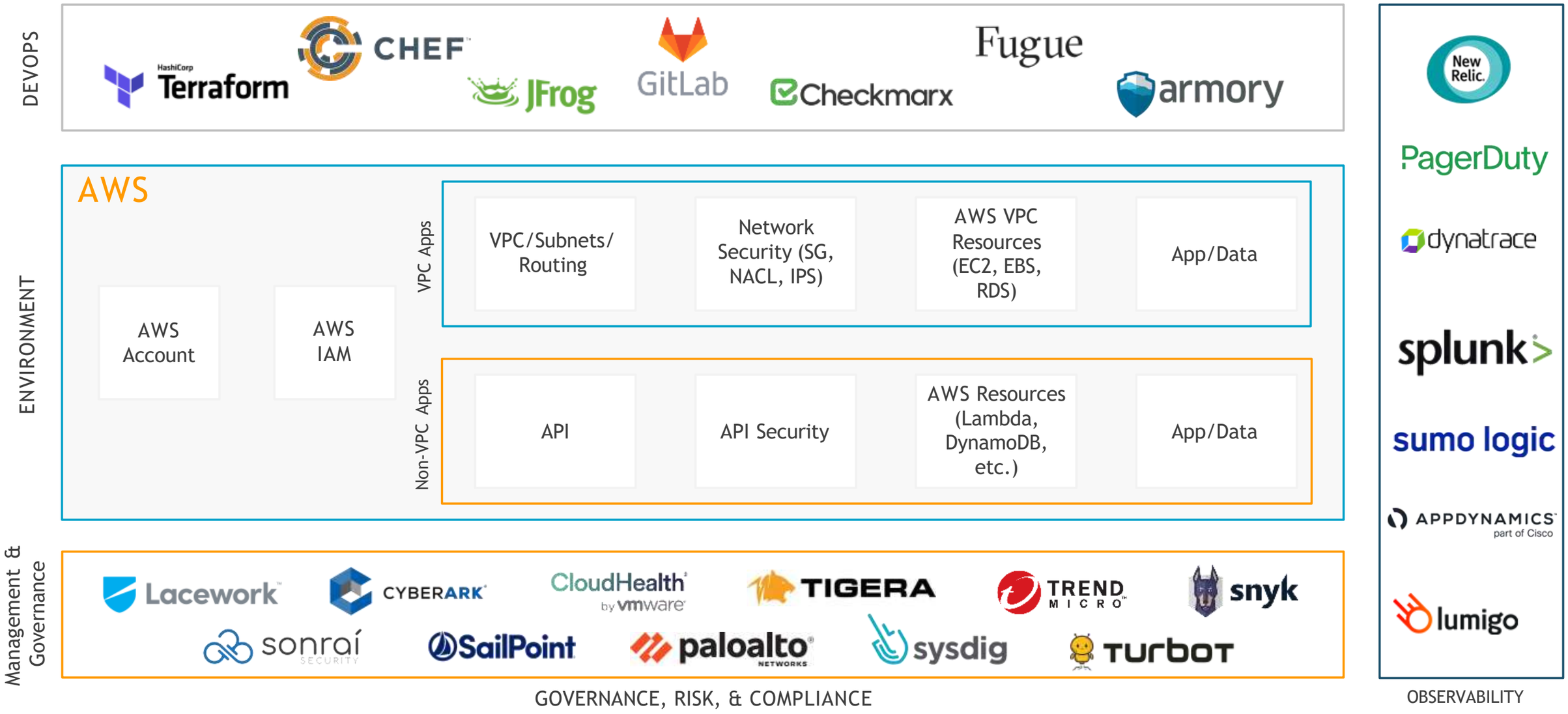
GOVERNANCE, RISK, & COMPLIANCE

- DevOps Dashboards
- Application Monitoring
- Infrastructure Monitoring
- Security and Compliance Monitoring
- Governance Reporting

OBSERVABILITY

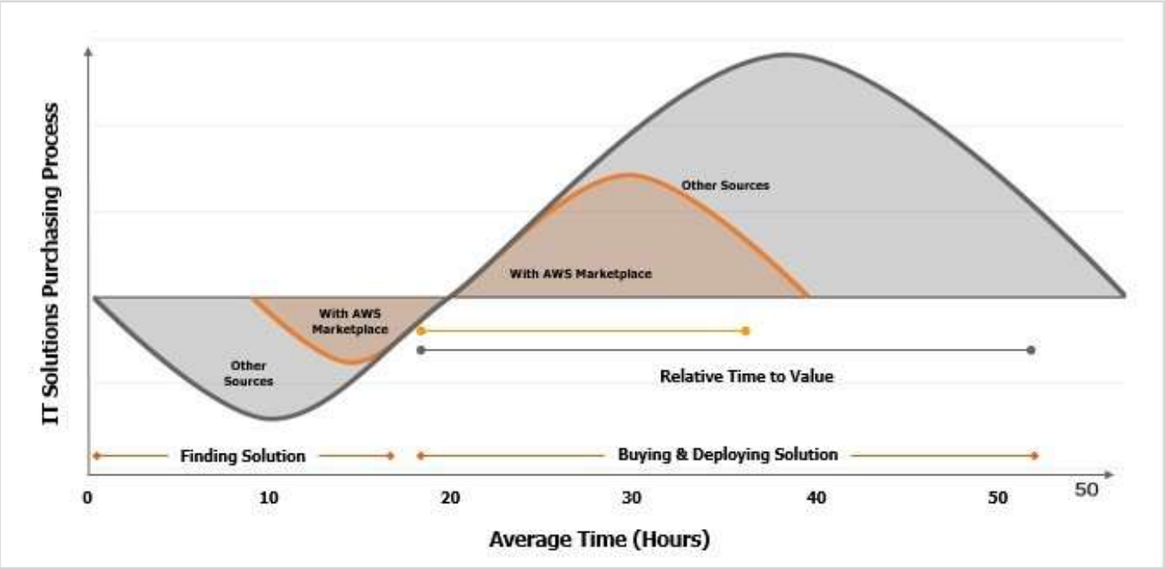
How do we enable DevSecOps via AWS Marketplace?

Enabling DevSecOps with AWS Marketplace



Why AWS Marketplace?

Find, buy, and deploy solutions quicker



IT decision-makers (ITDMS) cut their time in half using AWS Marketplace compared to other sources.

Make more satisfying purchases

ITDMS feel 2.4x better about purchasing using AWS Marketplace compared to other sources.

*Amazon Web Services (AWS) Marketplace surveyed 500 IT decision-makers (ITDMs) and influencers across the U.S. to understand software usage, purchasing, consumption models, and compared savings.

How can you get started?

Find

A breadth of DevOps solutions including:

TREND MICRO
snyk
sysdig
JFrog
turbot
TIGERA
Lacework

And more:
https://aws.amazon.com/marketplace/solutions/devops

Buy

Through flexible purchasing options:

Free trial

Pay-as-you-go

Budget alignment

Bring Your Own License (BYOL)

Private Offers

Billing consolidation

Enterprise Discount Program

Private Marketplace

Deploy

With multiple deployment options:

SaaS

Amazon Machine Image (AMI)

CloudFormation Template

Containers

Amazon EKS/Amazon ECS

AI/ML models

AWS Data Exchange

Workshop summary

DEVOPS

HashiCorp Terraform, CHEF, JFrog, GitLab, Checkmarx, Fugue, armory

Enabling Governance products

- Identifying and establishing Governance, Risk, and Compliance framework with the right partner solutions

Enabling DevOps products

- Everything as code (infrastructure to application)
- Templatized deployment for consistency
- Automated tests and deployment via CICD Pipeline

Enabling Observability products

- Strong Integration with tools for real-time monitoring and reporting for end-to-end observability

New Relic, PagerDuty, dynatrace, splunk>, sumo logic, APPDYNAMICS part of Cisco, lumigo

OBSERVABILITY

Management & Governance

Lacework, CYBERARK, CloudHealth by vmware, TIGERA, TREND MICRO, snyk, sonraí SECURITY, SailPoint, paloalto NETWORKS, sysdig, TURBOT

GOVERNANCE, RISK, & COMPLIANCE

Thank You