DevOps Transformation

An Architect Perspective

Sajal Debnath (he/his/him)
Staff Multi-Cloud Solutions Architect

December 2022

Agenda

Why?

Why use DevOps?

What?

What is DevOps?

Capabilities

Governing capabilities of DevOps

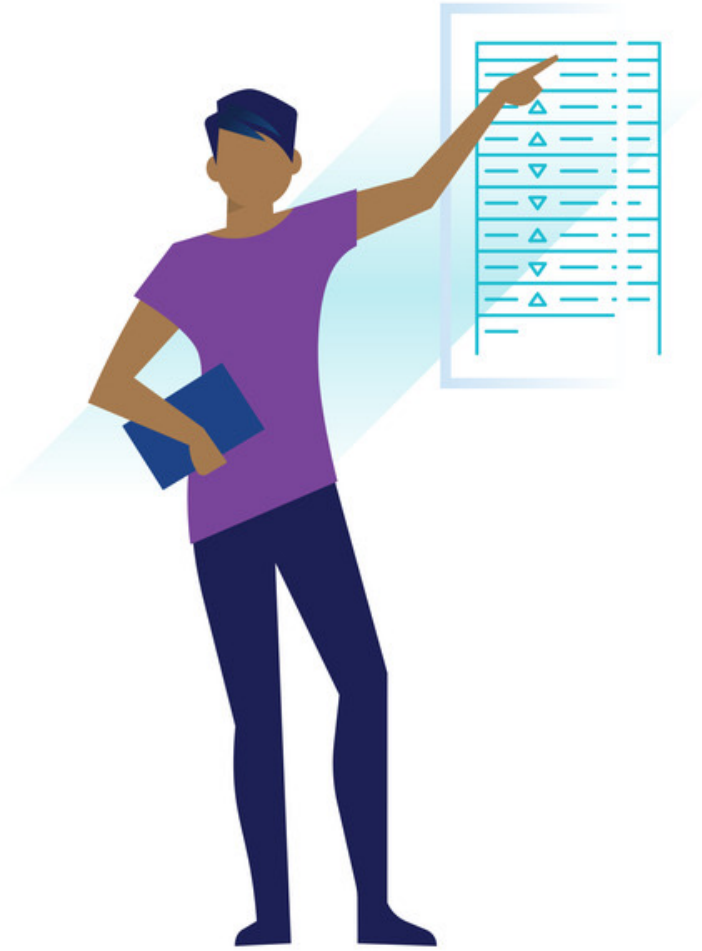
DevOps Architect

Who is a DevOps Architect?

The Transformation

How to lead an Organization in its DevOps journey?

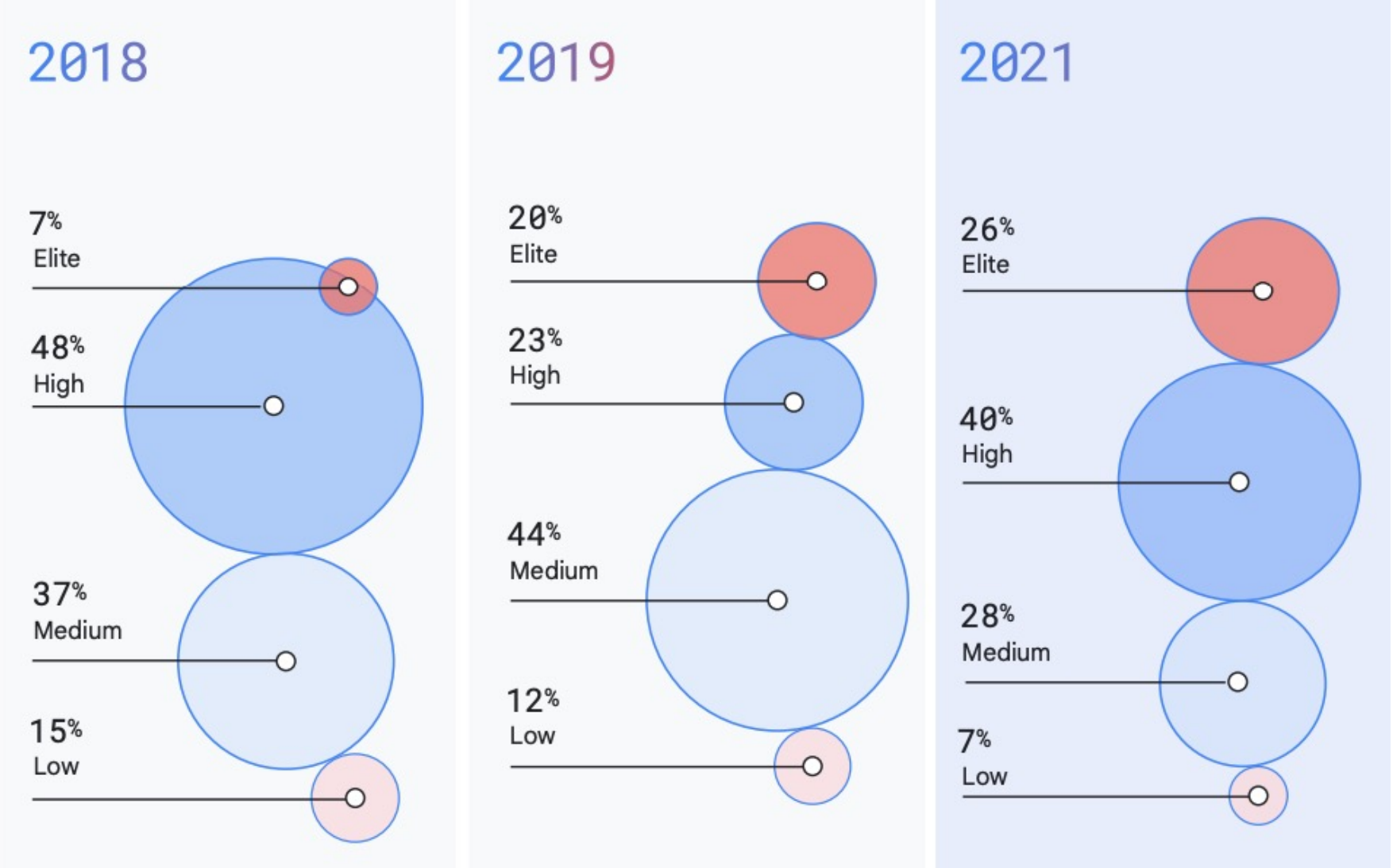
Reference Architecture & How to Implement



Why?

Why use DevOps?

DevOps Adoption Growth



Advantages

Elite Organization vs Low Organizations

973x

More frequent code deployment

6570x

Faster lead time from commit to deploy

This is not an error

3x

Lower change failure rate (changes are 1/3 less likely to fail)

6570x

Faster time to recover from incidents

Source: https://services.google.com/fh/files/misc/state-of-devops-2021.pdf

Overall DevOps Advantages

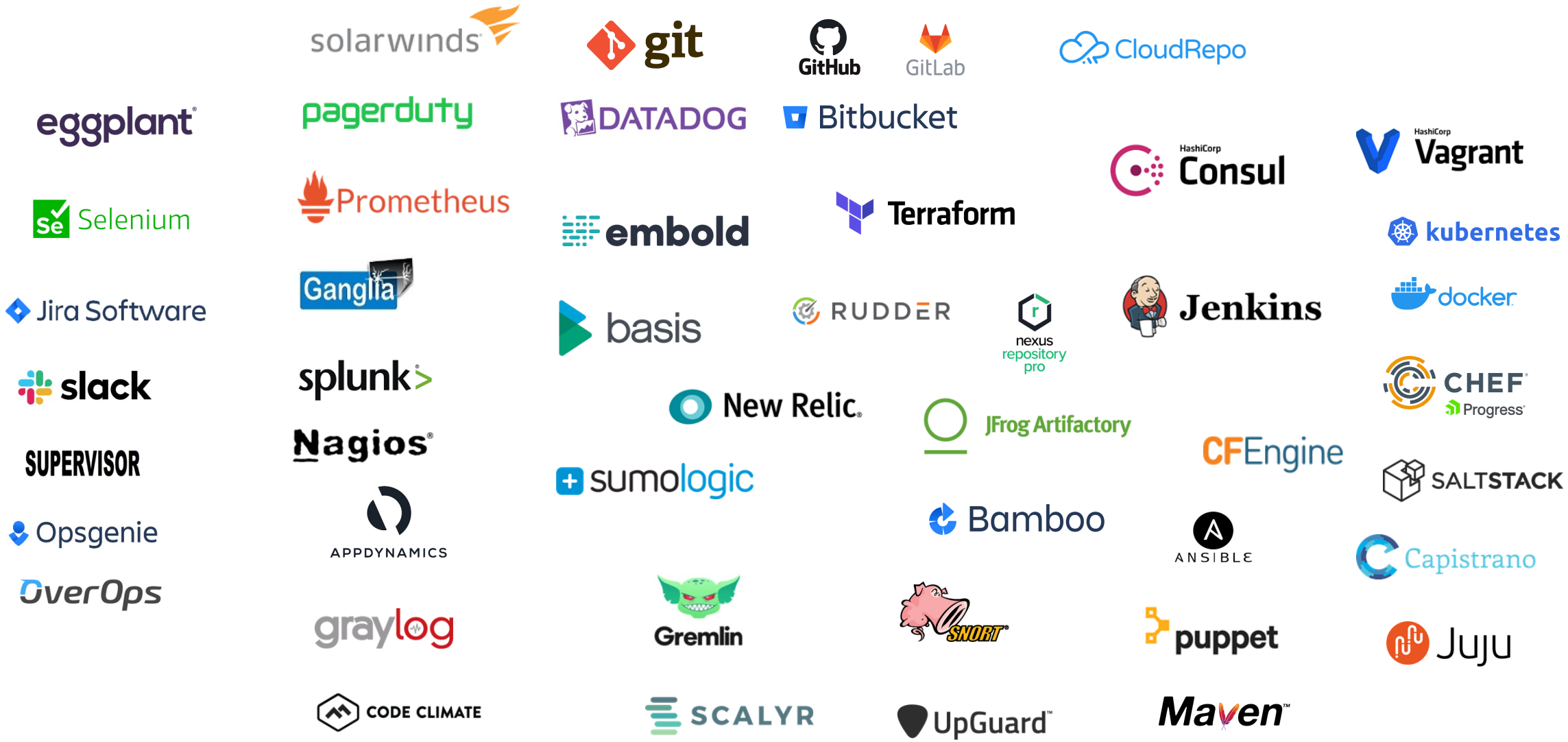
- Development Cycles
- Deployment Failures
- Communication
- Efficiencies
- Reduced Costs
- Faster Innovation
- Time to Recover
- Collaboration
- Rollbacks
- IT Headcount

What?

What is DevOps?

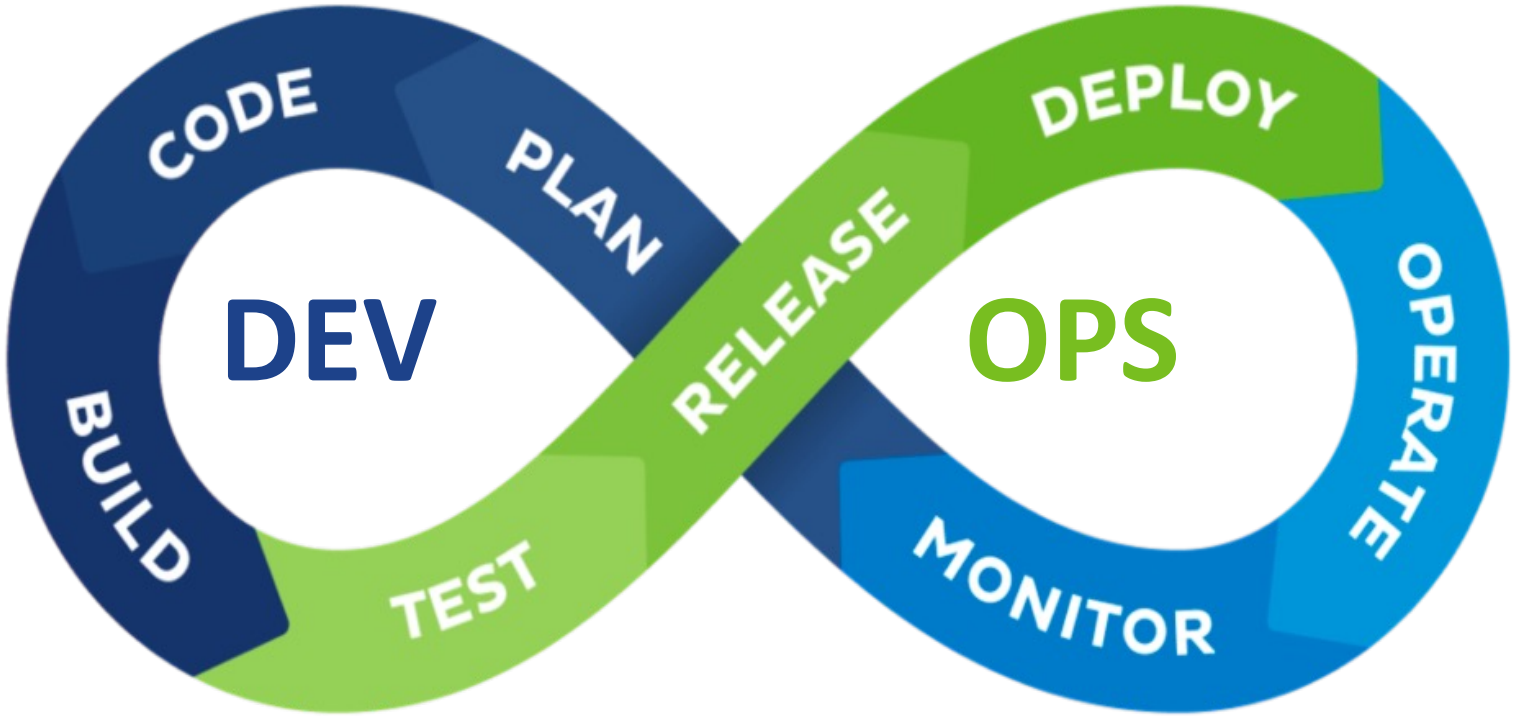
Tools

Is it a Collection of Tools?



A Set of Practices

No Formal Universal Definition

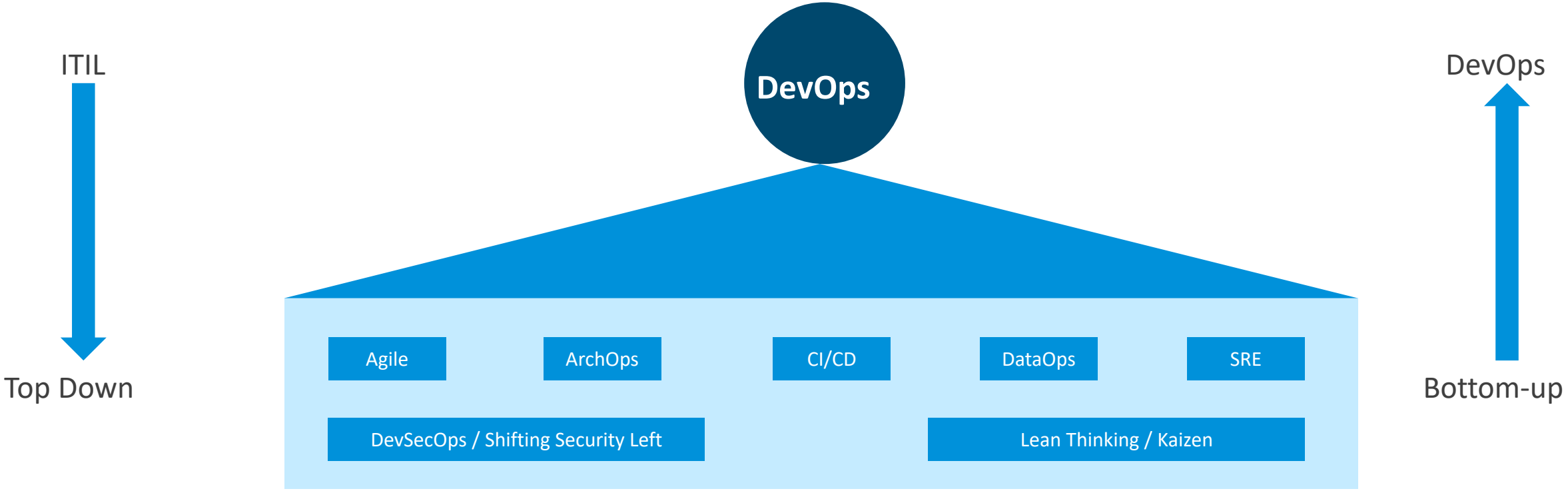


A set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality.

Source: https://en.wikipedia.org/wiki/DevOps

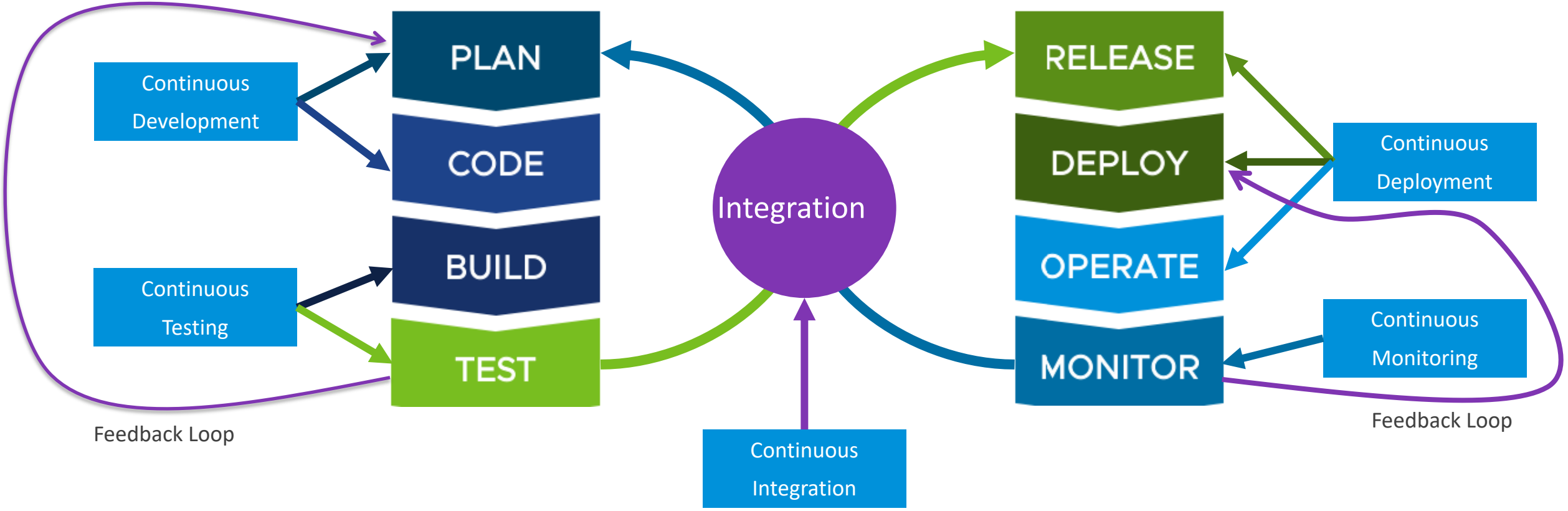
Relationship and Approach

Approach to DevOps and Relationship to Other Approaches



Flexible practice, created by software engineers, with software engineer needs in mind.

DevOps phases



Capabilities

Governing capabilities of DevOps

Overall Requirements

Technical capabilities

Loosely coupled architecture

Continuous testing & Continuous integration

Trunk-based development instead of long-lived branches & delay merging

Use of open-source technologies

Monitoring and observability practices

Deployment automation

Detailed Capabilities

Detailed List of Technical Capabilities

- Version control
- Trunk-based development
- Continuous integration
- Deployment automation
- Continuous testing
- Continuous delivery
- Architecture
- Empowering teams to choose tools
- Test data management
- Shifting left on security
- Cloud infrastructure
- Code maintainability

Source: https://cloud.google.com/architecture/devops/capabilities

Other Capabilities

Detailed List of Process, Measurement and Cultural Capabilities

Process Capabilities

- Team experimentation
- Streamlining change approval
- Customer feedback
- Visibility of work in the value stream
- Working in small batches

Measurement Capabilities

- Monitoring systems to inform business decisions
- Monitoring and observability
- Proactive failure notification
- Work in process limits
- Visual management capabilities

Cultural Capabilities

- Westrum organizational culture
- Learning culture
- Job satisfaction
- Transformational leadership

DevOps Architect

Who is a DevOps Architect?

DevOps Architect vs DevOps Engineer

What defines an architect

DevOps Architect

- Understands the full software development lifecycle and knows how to integrate all aspects of it into a CI/CD pipeline. This is true of both application code and infrastructure-as-code.
- In-depth knowledge of deployment patterns and site-reliability engineering.
- Thought leader and visionary for the organization.

DevOps Engineer

- DevOps engineers are traditionally people involved with operations.
- Maintains the version and dependency management.
- Is involved in the building and testing code only.

Architect Skillset

Development	Operations	Coaching	Architecture
Advance knowledge of Agile workflow	Advance knowledge of CI tools (Jenkins/Gitlab/Bamboo)	Ability to clearly document and communicate key facts and information	Knowledge of CI/CD tooling integration in support of the SDLC
Experience with Automated testing	Advance knowledge of OS & MW technologies	Product-centric mindset	Experience with the cloud platform
Experience with build tools/frameworks (Maven, Gradle, NPM, ANT)	Strong operational experience in Linux/Unix environment and scripting languages (Shell, Python)	Cross-functional leadership skills	Experience designing highly available systems
Development experience with a object-oriented language (Java, Python, Ruby, JS)	Experience implementing application and system level monitoring	Ability to advocate for both operations and development staff	Advanced knowledge of N-tier application architecture
	Strong automation skills (tools agnostic) and the ability to drive better initiatives to automate process		Design & implement best practices as per enterprise standards
	Advanced knowledge of configuration management tools (Ansible/Chef/Puppet)		

Soft-skill | Technical Skill | Deep Technical Expertise

Source: https://betterprogramming.pub/what-exactly-is-a-devops-architect-a27e5a5c728b

Responsibilities

Overall responsibilities of a DevOps Architect

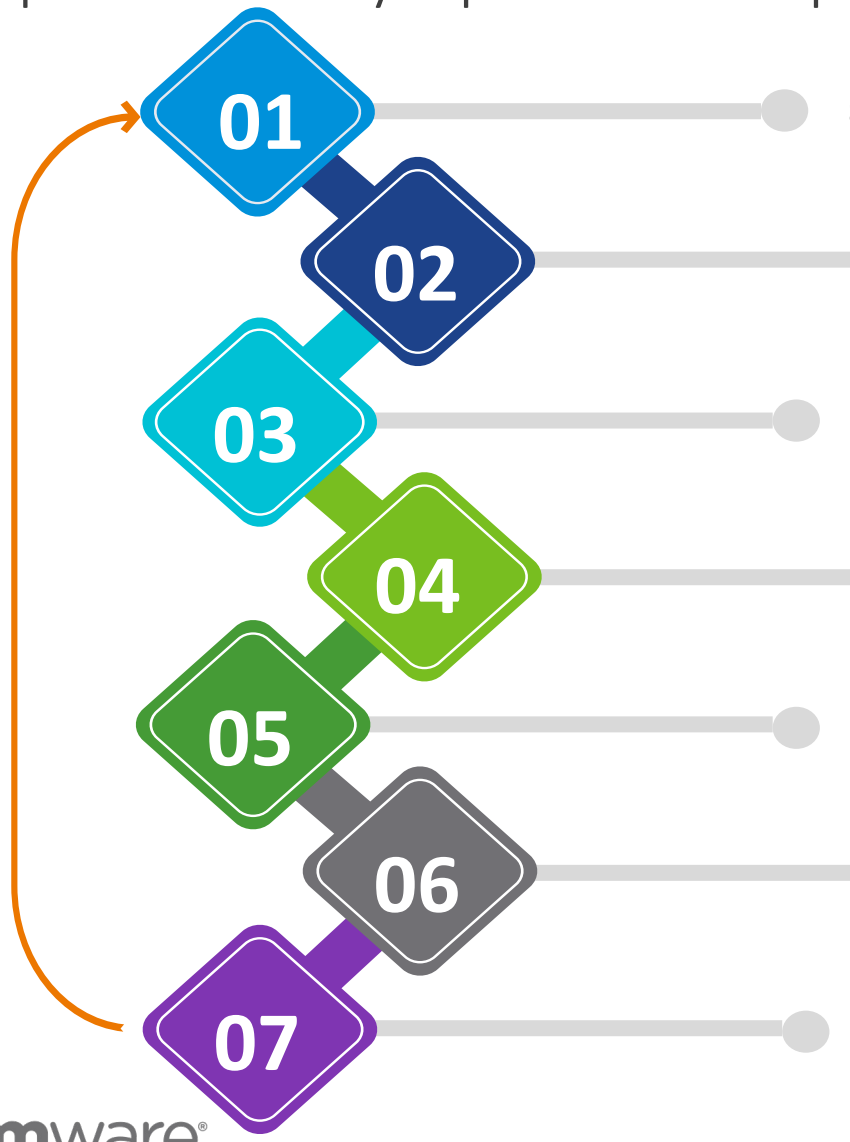
- Facilitating the development process and operations.
- Identifying setbacks and shortcomings.
- Creating suitable DevOps channels across the organization.
- Establishing continuous build environments to speed up software development.
- Designing efficient practices.
- Delivering comprehensive best practices.
- Managing and reviewing technical operations.
- Guiding the development team.

The Transformation

How to lead an Organization in its DevOps journey?

Transformation Process

Steps to successfully implement a DevOps process



STEP 01 – Planning & Preparation

Perform Audit and Evaluation (Application + Infra)

STEP 02 – DevOps CoE

Set up a DevOps Center of Excellence

STEP 03 – Cultural Change

Build a culture of learning and collaboration in the Organization

STEP 04 – KPI & Governance

Define and establish DevOps KPI process governance mechanism

STEP 05 – Project in-take & Scrum team workshops

Set up a clear project in-take process and conduct workshops for the scrum teams to ensure the right sizing of tools and processes.

STEP 06 – Pilot Program

Run the pilot program and compare results with defined KPI

STEP 07 – Feedback & Scale Out

Gather feedbacks and scale-out DevOps transformations process

Audit the Application

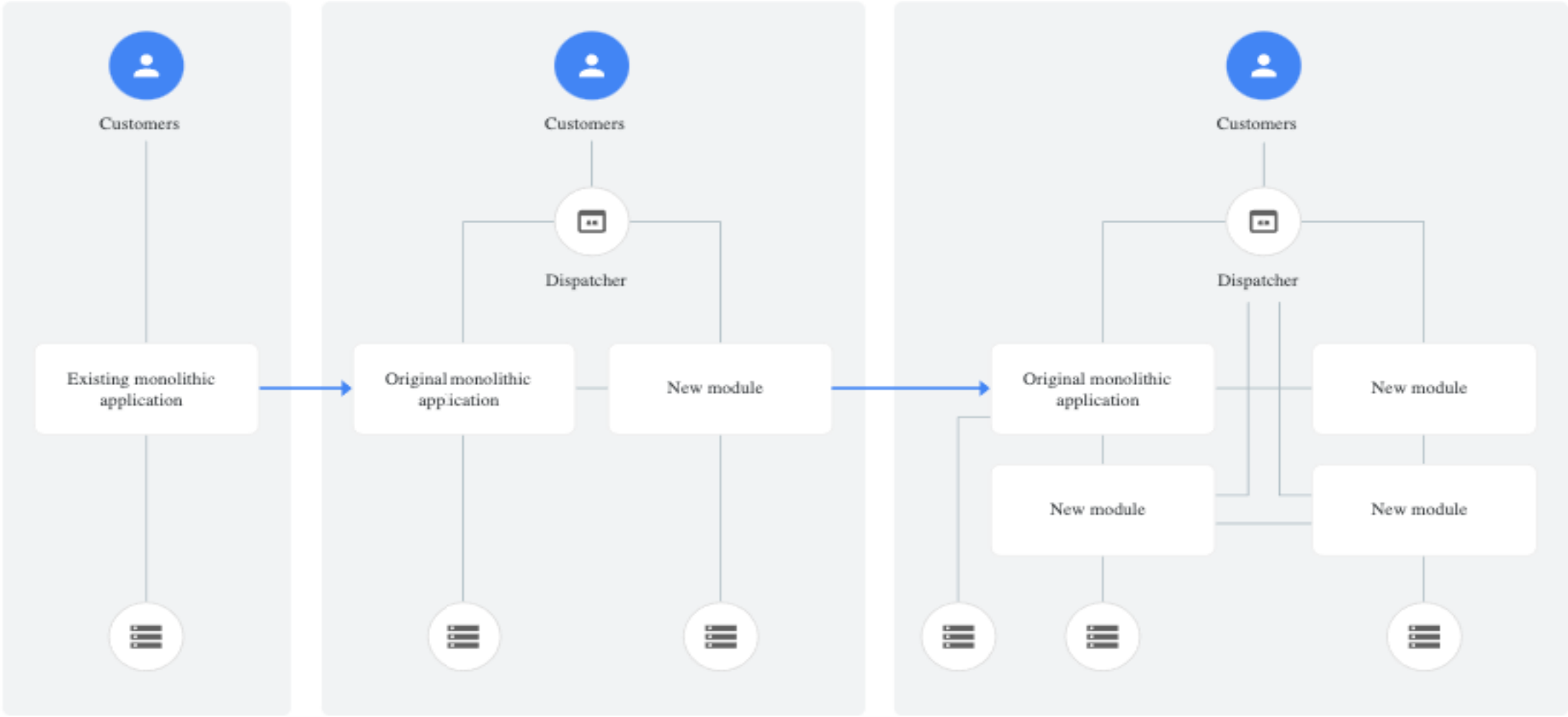
Identify the application type and stage

Archetype	Pros	Cons
Monolithic v1 (all functionality in one application)	• Simple at first • Low interprocess latencies • Single codebase, one deployment unit • Resource-efficient at small scales	• Coordination overhead increases as team grows • Poor enforcement of modularity • Poor scaling • All-or-nothing deploy (downtime failures) • Long build times
Monolithic v2 (set of monolithic tiers: frontend presentation, application server, database layer)	• Simple at first • Join queries are easy • Single schema deployment • Resource-efficient at small scales	• Tendency for increased coupling over time • Poor scaling and redundancy (all or nothing, vertical only) • Difficult to tune properly • All-or-nothing schema management
Microservice (modular, independent, graph relationship or tiers, isolated persistence)	• Each unit is simple • Independent scaling and performance • Independent testing and deployment • Can optimally tune performance (caching, replication, etc.)	• Many cooperating units • Many small repos • Requires more sophisticated tooling and dependency management • Network latencies

Source: https://cloud.google.com/architecture/devops/devops-tech-architecture

Application Transformation

Evolutionary Architecture – Strangler method



** Modernize an application by making it compatible with 12 factor App requirements - https://12factor.net/

Source: https://cloud.google.com/architecture/devops/devops-tech-architecture

Must Have Processes

Ensure the presence of the processes

Continuous Integration

Microservices

Continuous Delivery

Infrastructure as Code

Configuration Management

Policy as Code

Monitoring & Logging

Communication & Collaboration

Source: https://aws.amazon.com/devops/what-is-devops/

Standard Requirements

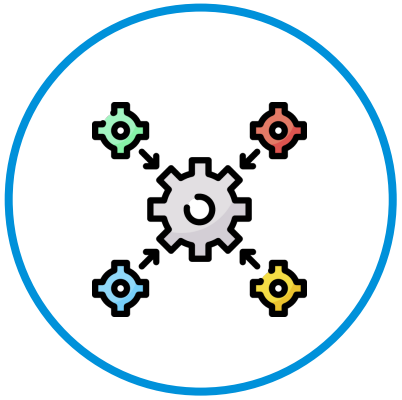
In general, what the Organization should have

- Application requirements
  - Cloud-native
  - Minimize vendor lock-in risk exposure as feasible
- Cloud-native infrastructure
  - Reduce maintenance costs and complexity
  - Cloud agnostic infrastructure
- Web Application Firewall (WAF)
  - Protect at application level
  - Audit logs
  - Distributing application traffic across multiple targets
- Key management
  - Manage cryptographic keys and control audit logs
- Monitoring
  - Monitoring should run separate from app
  - Automated monitoring of all stages
- Storage
- Development tools
  - Integrated and native development chain for CI/CD
- Security
  - Code quality analysis
  - Security checking of inhouse to conform to standards

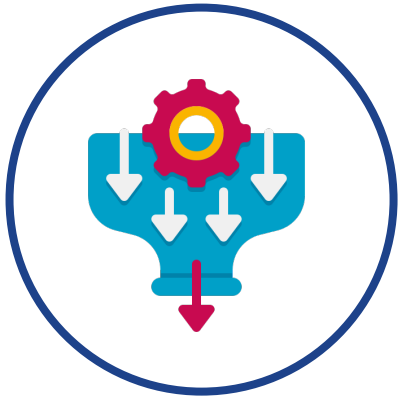
Common Pitfalls

Avoid these common mistakes

Simultaneously releasing many services



Integrating changes with the changes from hundreds, or even thousands, of other developers



Creating bottlenecks in the software delivery process

DevSecOps

Security – Shift Left and Integrate Throughout

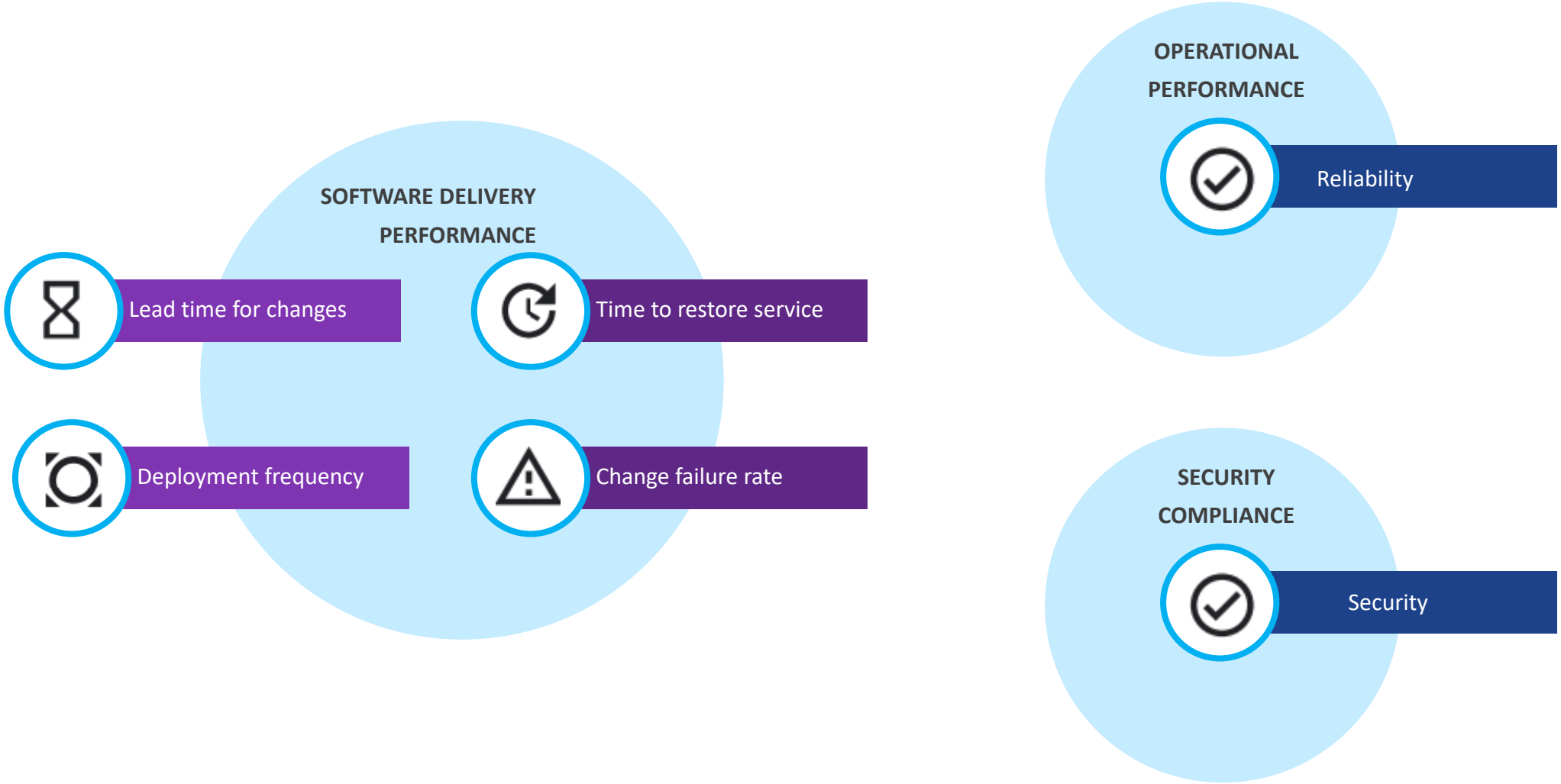
Test for security	Test security requirements as a part of the automated testing process, including areas where pre-approved code should be used.
Integrate review into every phase	Integrate InfoSec into the daily work of the entire software delivery lifecycle, including the design and architecture phases of the application, attend software demos, and provide feedback during demos.
Security reviews	Conduct a security review for all major features.
Build pre-approved code	Have the InfoSec team build pre-approved, easy-to-consume libraries, packages, toolchains, and processes for developers and IT operators to use in their work.
Invite InfoSec early and often	Include InfoSec during planning and all subsequent phases of application development, so that they can spot security-related weaknesses early, which gives the team ample time to fix them.

Measuring Success

Metrics for success measurement

Important Metrics

Operational Metrics and Security Compliance Added Check



Categorization of Organizations

Constant evaluation of Organization

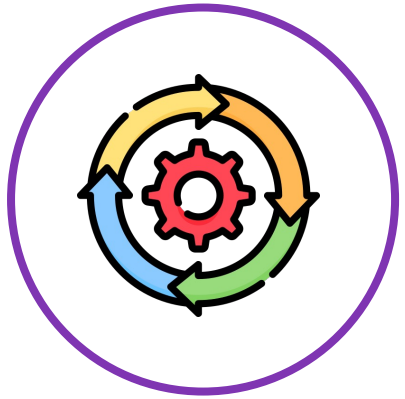
Software delivery performance metric	Elite	High	Medium	Low
Deployment frequency For the primary application or service you work on, how often does your organization deploy code to production or release it to end users?	On-demand (multiple deploys per day)	Between once per week and once per month	Between once per month and once every 6 months	Fewer than once per six months
Lead time for changes For the primary application or service you work on, what is your lead time for changes (i.e., how long does it take to go from code committed to code successfully running in production)?	Less than one hour	Between one day and one week	Between one month and six months	More than six months
Time to restore service For the primary application or service you work on, how long does it generally take to restore service when a service incident or a defect that impacts users occurs (e.g., unplanned outage or service impairment)?	Less than one hour	Less than one day	Between one day and one week	More than six months
Change failure rate For the primary application or service you work on, what percentage of changes to production or released to users result in degraded service (e.g., lead to service impairment or service outage) and subsequently require remediation (e.g., require a hotfix, rollback, fix forward, patch)?	0%-15%	16%-30%	16%-30%	16%-30%

Source: https://services.google.com/fh/files/misc/state-of-devops-2021.pdf

The Result

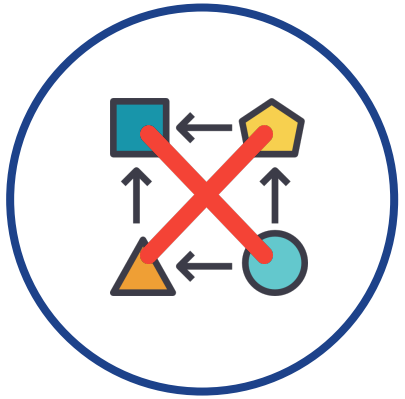
What can be achieved!!

Correct Architecture Results



Teams can make large-scale changes to the design of their systems without the permission of somebody outside the team or depending on other teams.

Teams can complete work without needing fine-grained communication and coordination with people outside the team.



Teams deploy and release their product or service on demand, independently of the services it depends on or of other services that depend on it.

Teams do most of their testing on demand, without requiring an integrated test environment.

Teams can deploy during normal business hours with negligible downtime.

Source: https://cloud.google.com/architecture/devops/devops-tech-architecture

Samples of End Result

Software Deploy Frequencies of Different Companies

Company	Deploy Frequency	Deploy Lead Time	Reliability	Customer Responsiveness
Amazon	23,000 / day	minutes	high	high
Google	5,500 / day	minutes	high	high
Netflix	500 / day	minutes	high	high
Facebook	1 / day	hours	high	high
Twitter[2]	3 / week	hours	high	high
typical enterprise	once every 9 months	months or quarters	low/medium	low/medium

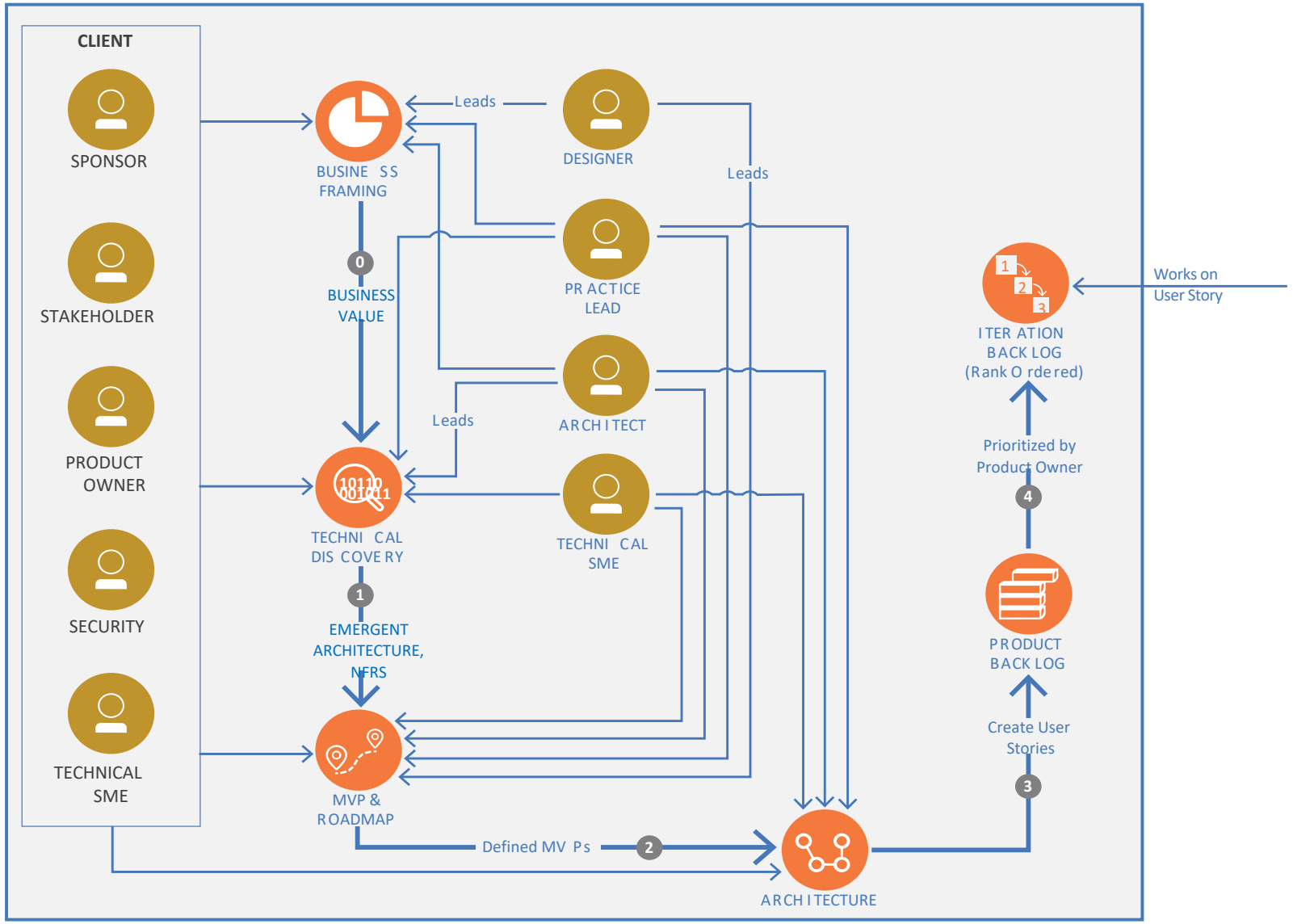
Source: https://www.oreilly.com/library/view/the-phoenix-project/9781457191350/42-resourceWhy.xhtml

Reference Architecture

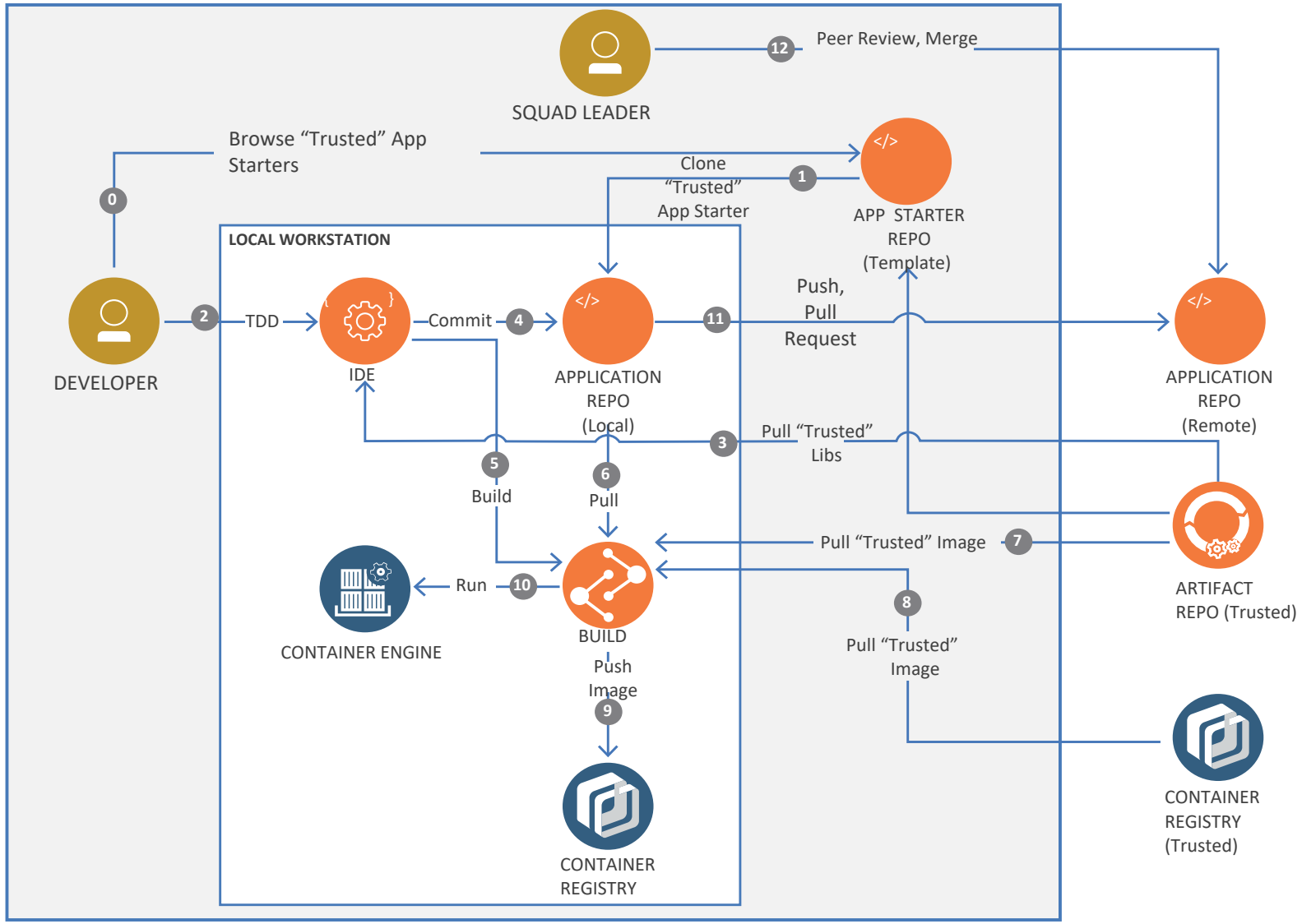
Example stages

Source: https://www.ibm.com/cloud/architecture/architectures/devOpsArchitecture/reference-architecture/

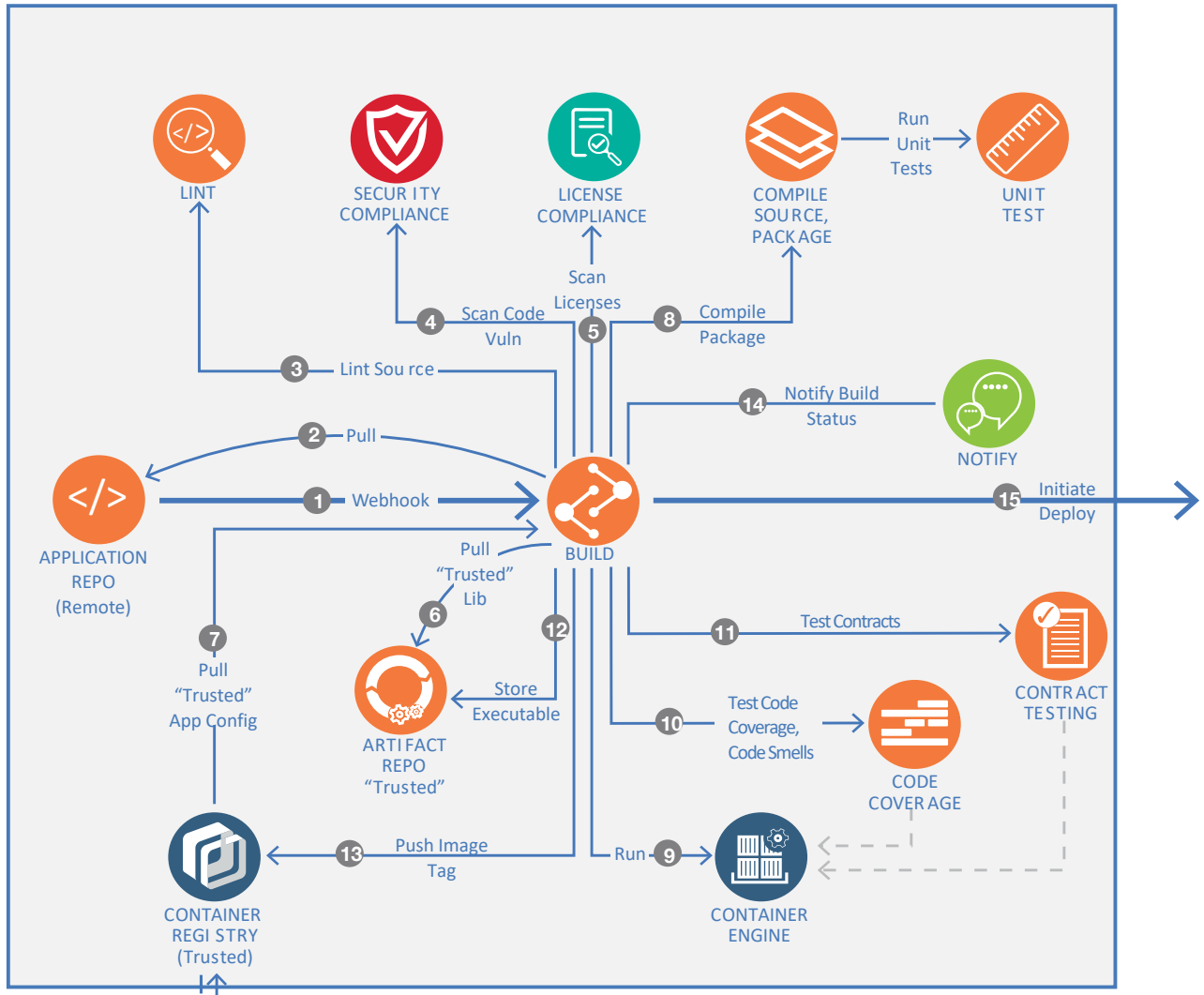
Continuous Planning



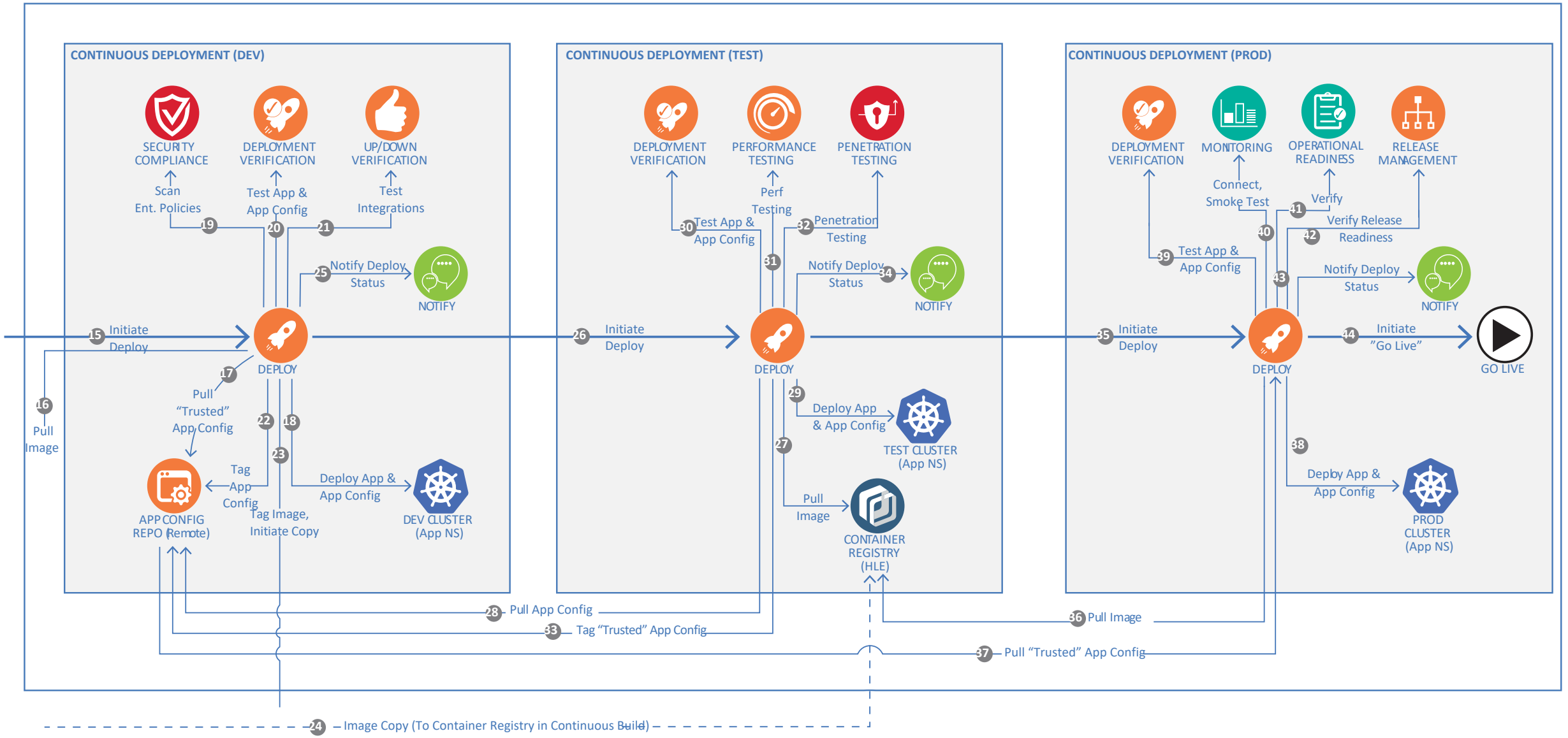
Continuous Development



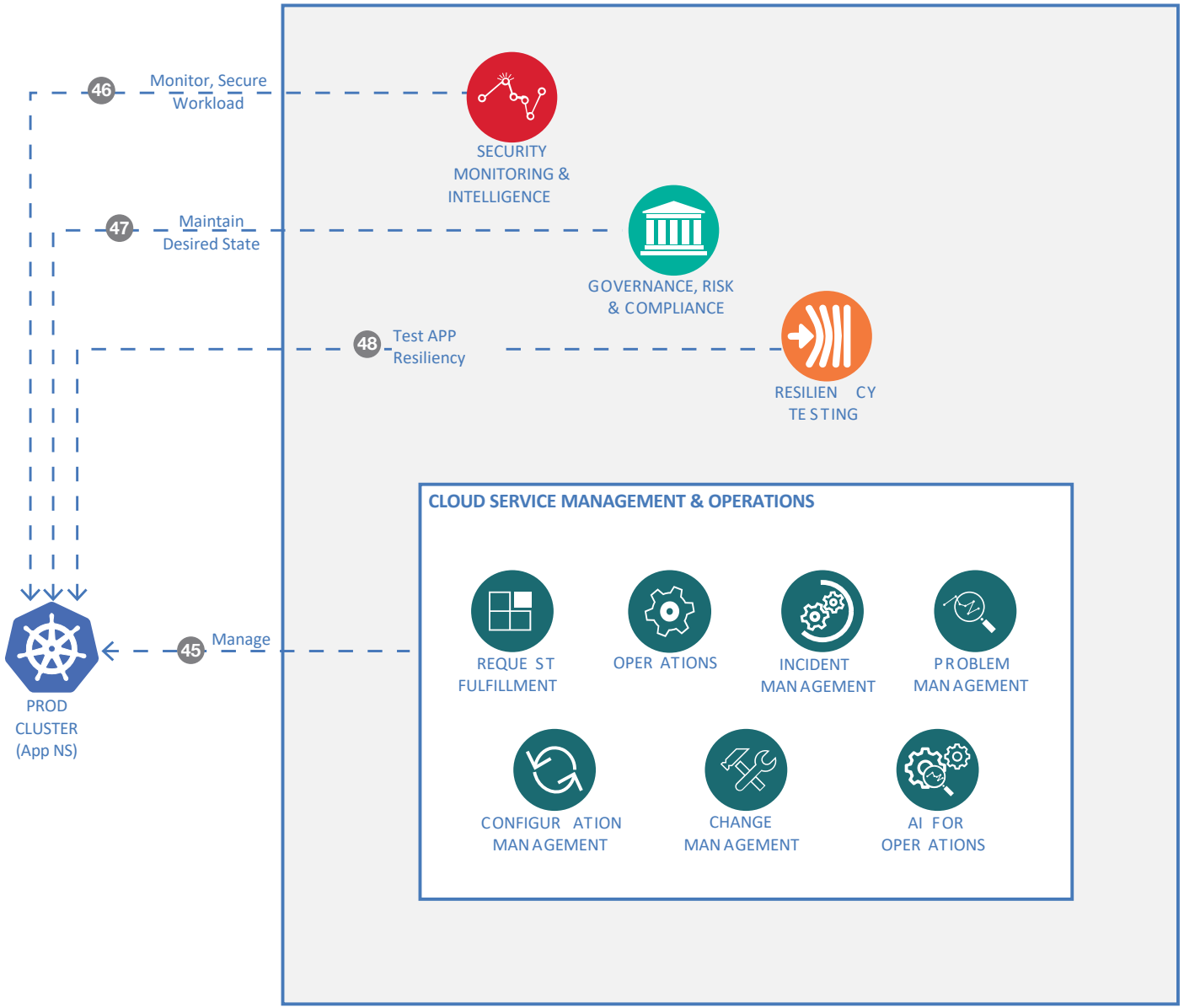
Continuous Integration



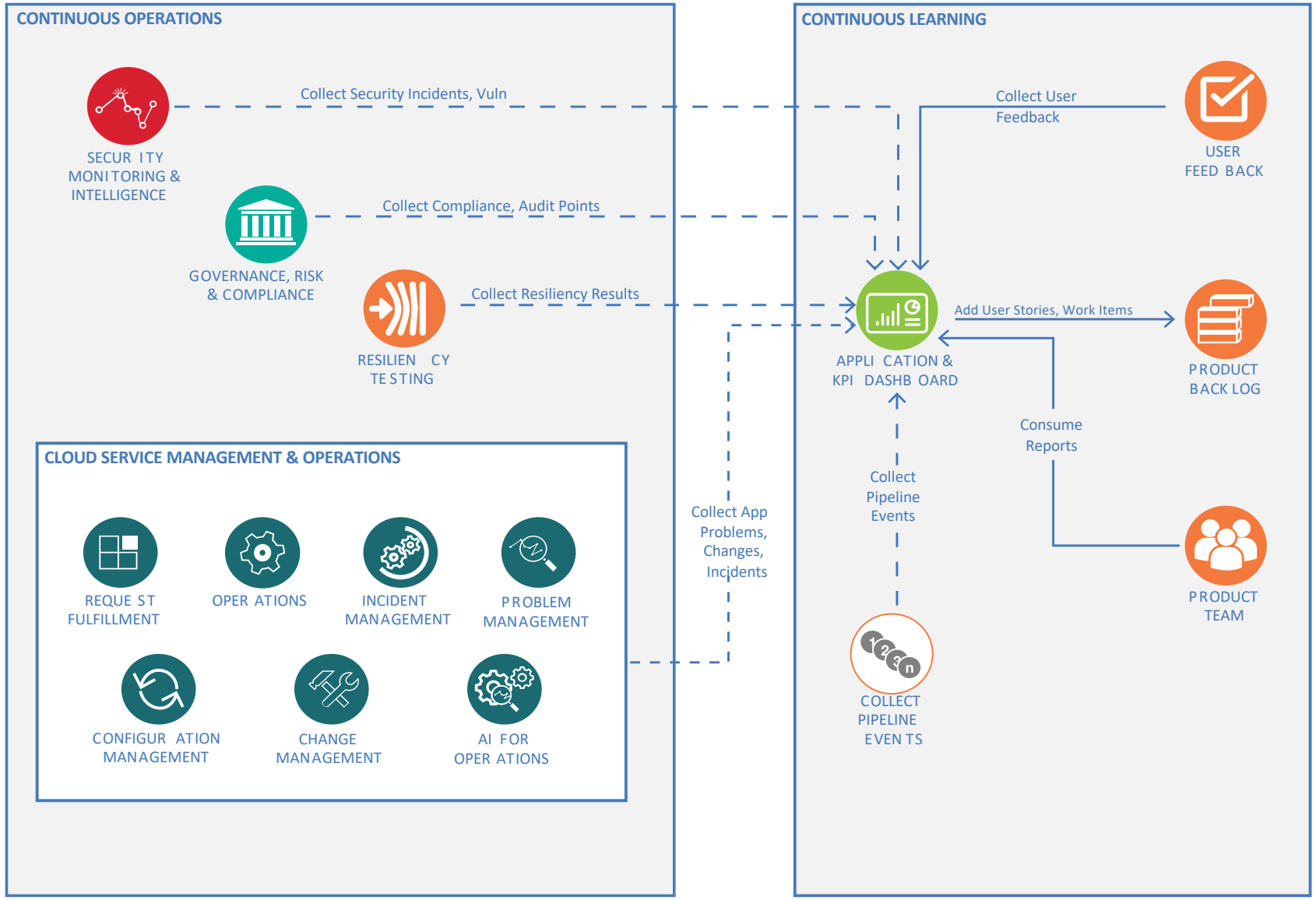
Continuous Deployment



Continuous Operations



Continuous Feedbacks



How to Implement?

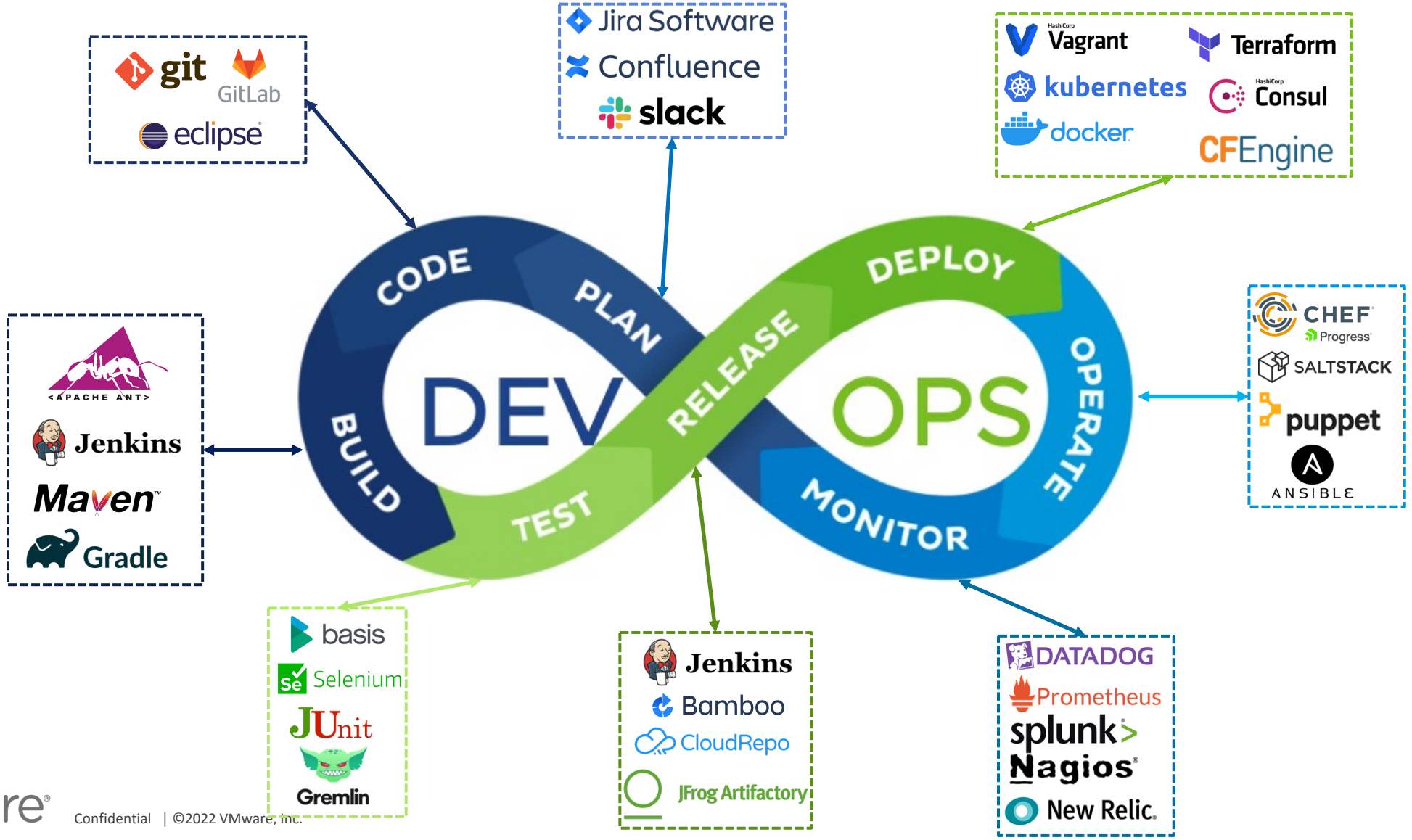
Tools to use and ways to implement the solution.

Tools aligned with stages

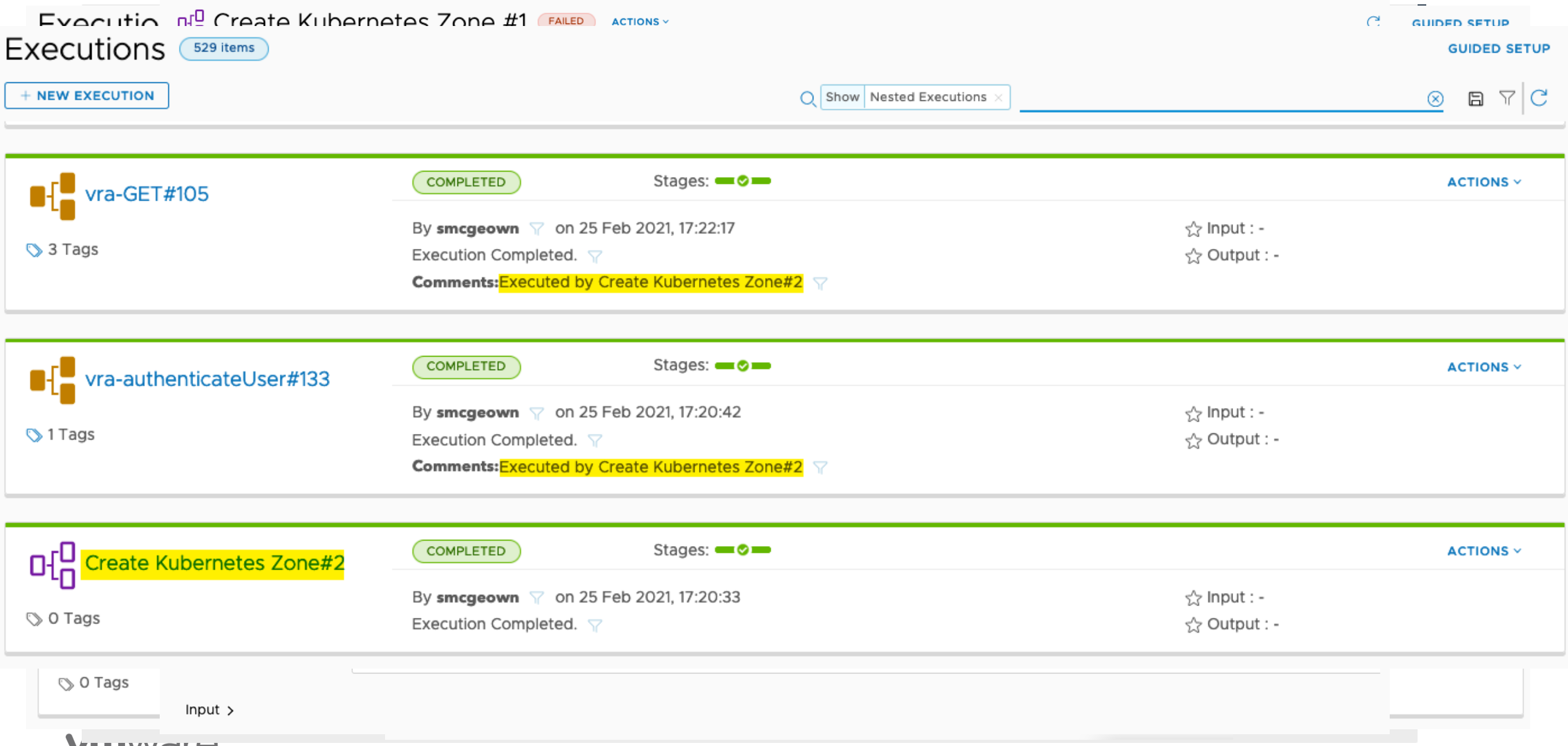
Defining Toolchains

Plan	Code-Build	Test	Release	Deploy/ Configuration	Operations/ Monitoring	Feedback
Jira Software Confluence slack	eclipse <APACHE ANT> Jenkins Maven Gradle	embold basis eggplant Selenium Jira Software Gremlin	Jenkins Bamboo JFrog Artifactory CloudRepo nexus repository pro	kubernetes docker CHEF Progress ANSIBLE puppet Terraform SALTSTACK CFEngine Capistrano Juju RUDDER	DATADOG solarwinds pagerduty Prometheus Ganglia splunk> Nagios sumologic OverOps SUPERVISOR CODE CLIMATE New Relic. SCALYR graylog APPDYNAMICS	GetFeedback slack Jira Service Management pendo
		Security SNORT UpGuard				

Example set of tools



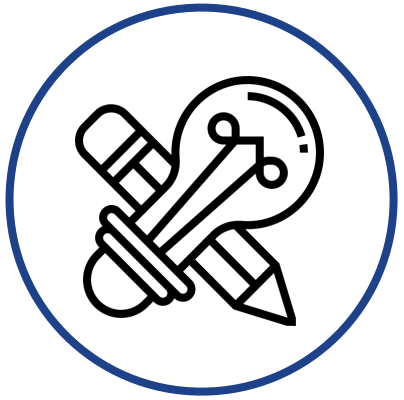
VMware Code Stream



Top Three Takeaways

In-depth understanding of DevOps

Ability to analyze and evaluate and environment



Successfully build a DevOps solution

- Listen to the presentation again
- Read! Check out https://learncodestream.github.io/
- Engage, design, implement

Thank You

sdebnath@vmware.com

https://sajaldebnath.com