DevSecOps Pipeline & Demo

JUNE 02, 2023

DevSecOps Innovations Team
Software Solutions Division

Carnegie Mellon University
Software Engineering Institute

DevSecOps Foundations

Carnegie Mellon University
Software Engineering Institute

DevOps is a set of principles and practices emphasizing collaboration and communication between software development teams and IT operations staff along with acquirers, suppliers, and other stakeholders in the lifecycle of a software system

DevSecOps is a model on integrating the software development and operational process considering security activities: requirements, design, coding, testing , delivery , deployment and incident response.

Mature DevOps practices are constantly testing, deploying and validating that software meets every requirement and allows for fast recovery in the event of a problem. As a result we can easily say,

"DevSecOps is DevOps done right"

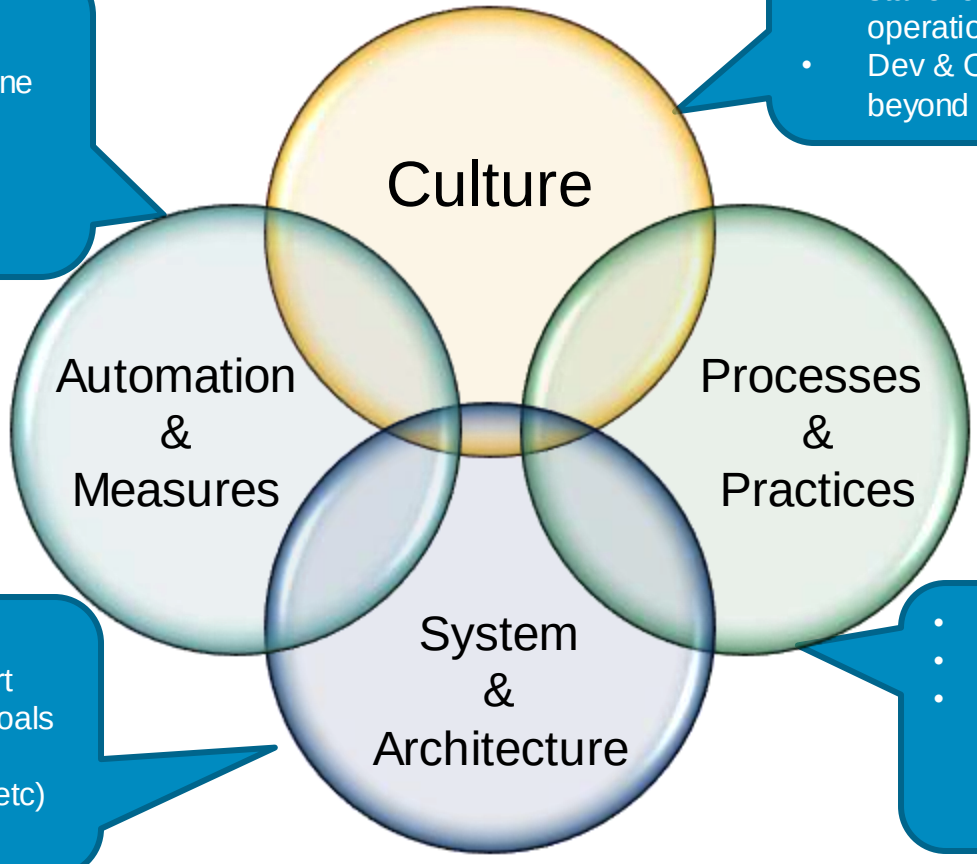
DevSecOps has four Fundamental Principles

- **Collaboration:** between project team roles
- **Infrastructure as Code:** all assets are versioned, scripted, and shared where possible
- **Automation:** deployment, testing, provisioning, any manual or human-error-prone process
- **Monitoring:** any metric in the development or operational spaces that can inform priorities, direction, and policy

It might seem simple, but it's NOT easy!

- What Some People Think Boundaries of DevSecOps is!
- Automate repetitive, error-prone tasks
- Static & Dynamic Systems Analysis
- Performance dashboards

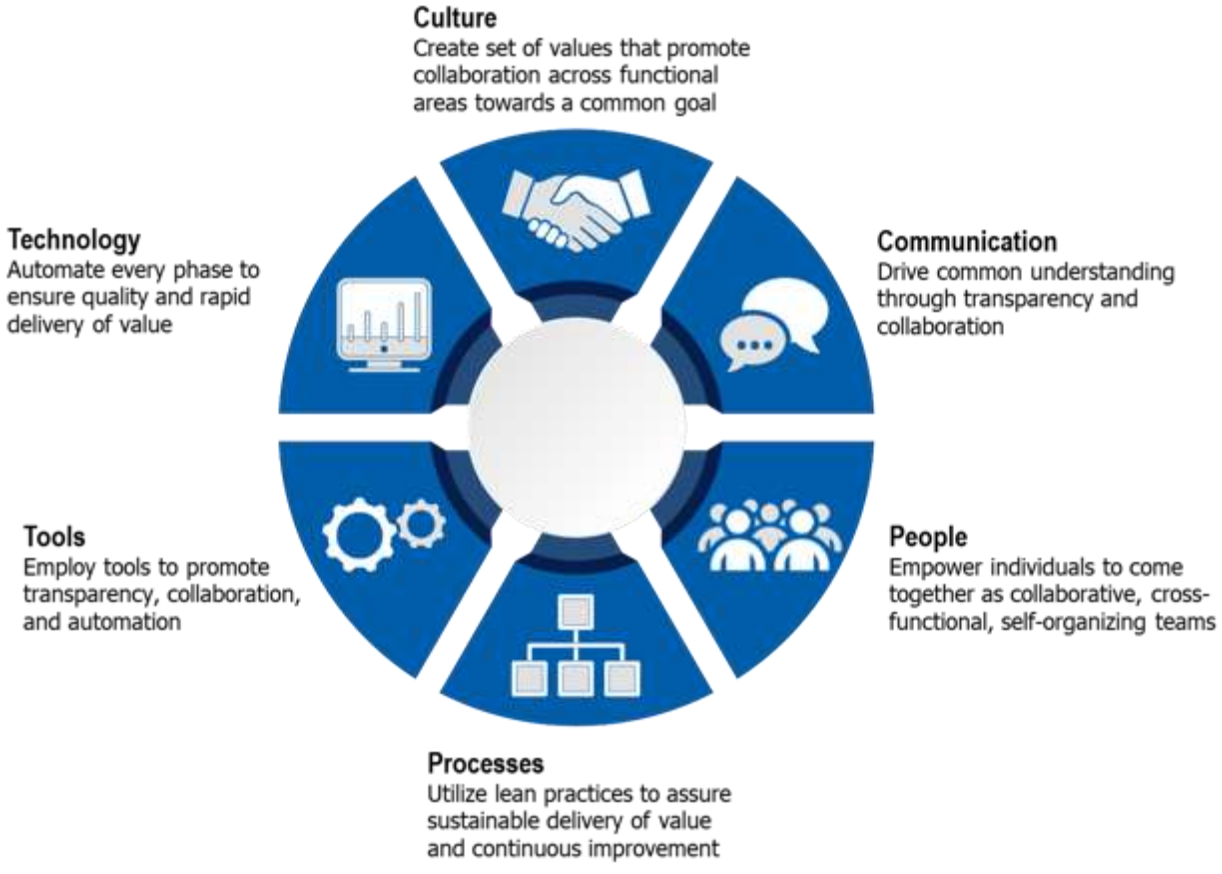
- All roles collaborate
- Dev, Ops, Sustainment have stakeholders that understand operational drivers
- Dev & Ops support products beyond delivery



- System architected to support integration and automation goals
- Represents important quality attributes (scalable, secure, etc)

- Value stream understanding
- Whole pipeline accounted for
- Continuous integration, automated test, virtualization, self-serve, scripting, automated deployment…

DevSecOps Overview

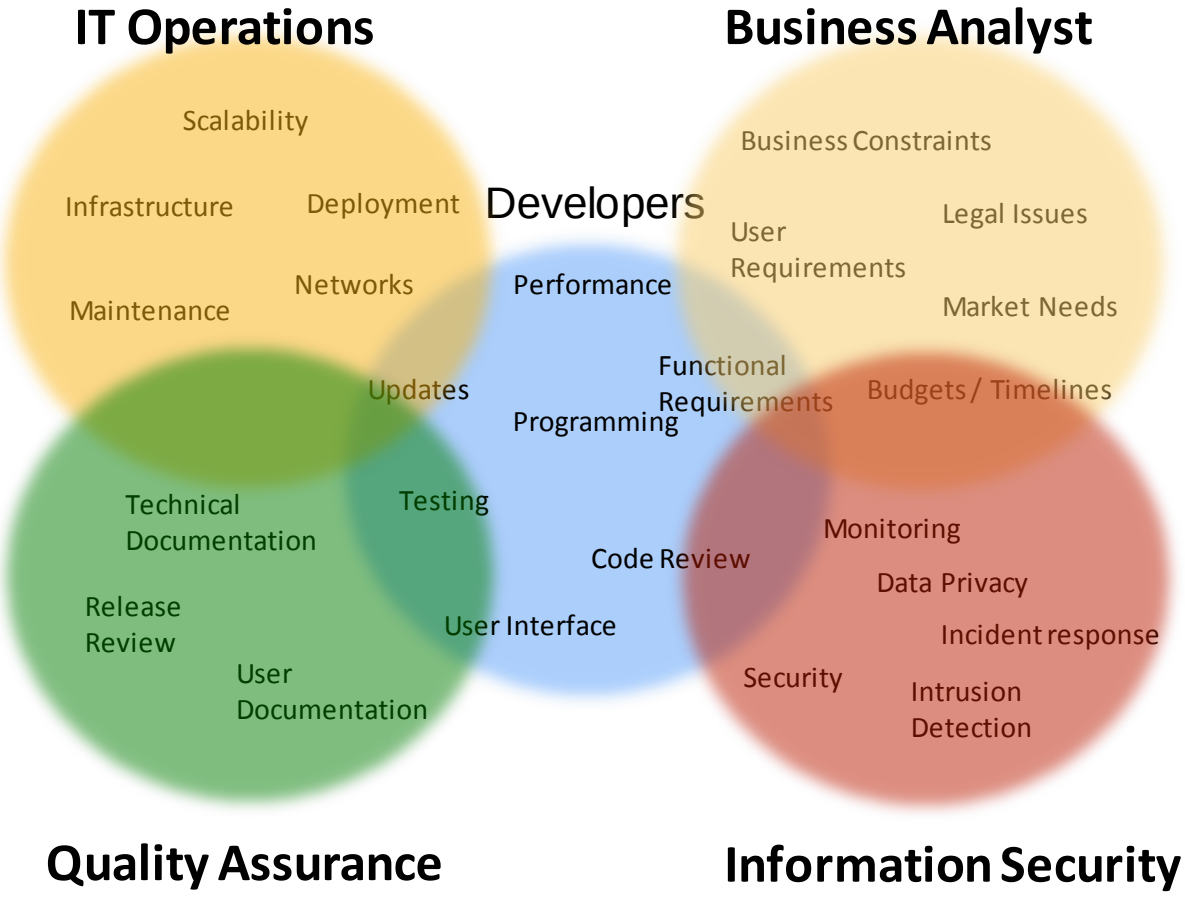


Agile practices+ *Architecture* + *Process*+ *Culture* = **Requirements** for an *Automated* DevSecOps Pipeline

The DevSecOps technology stack is *NOT* where we start

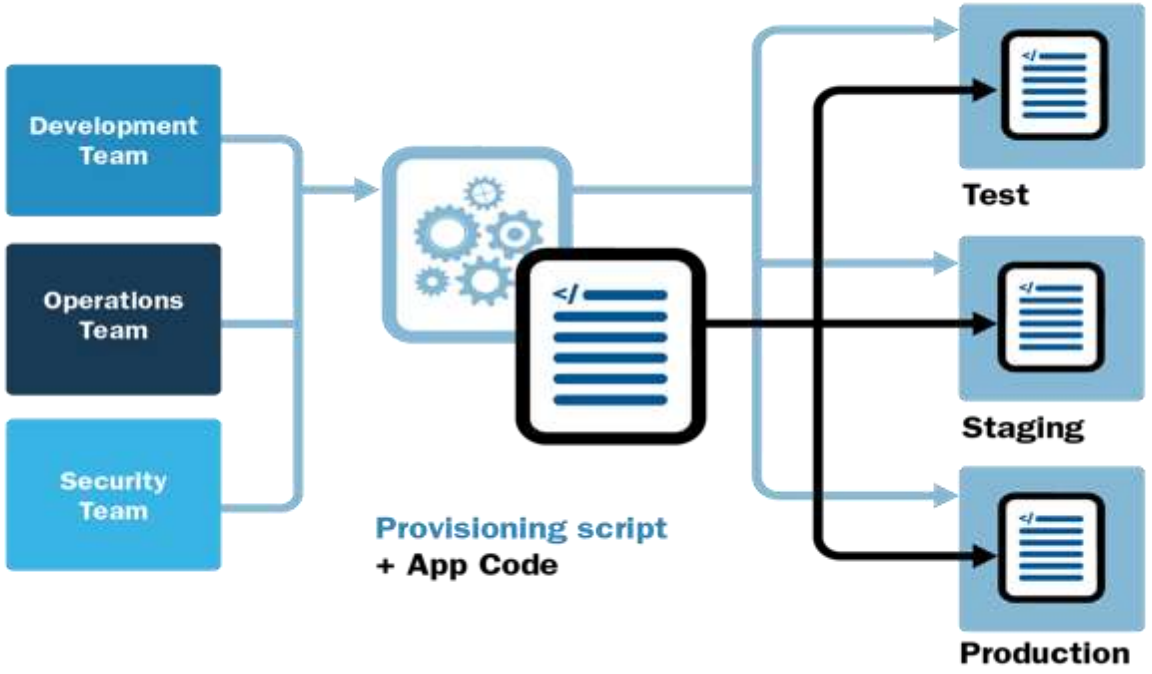
- Automating processes that don't add to or lead to value is more waste

Collaboration: Many stakeholders



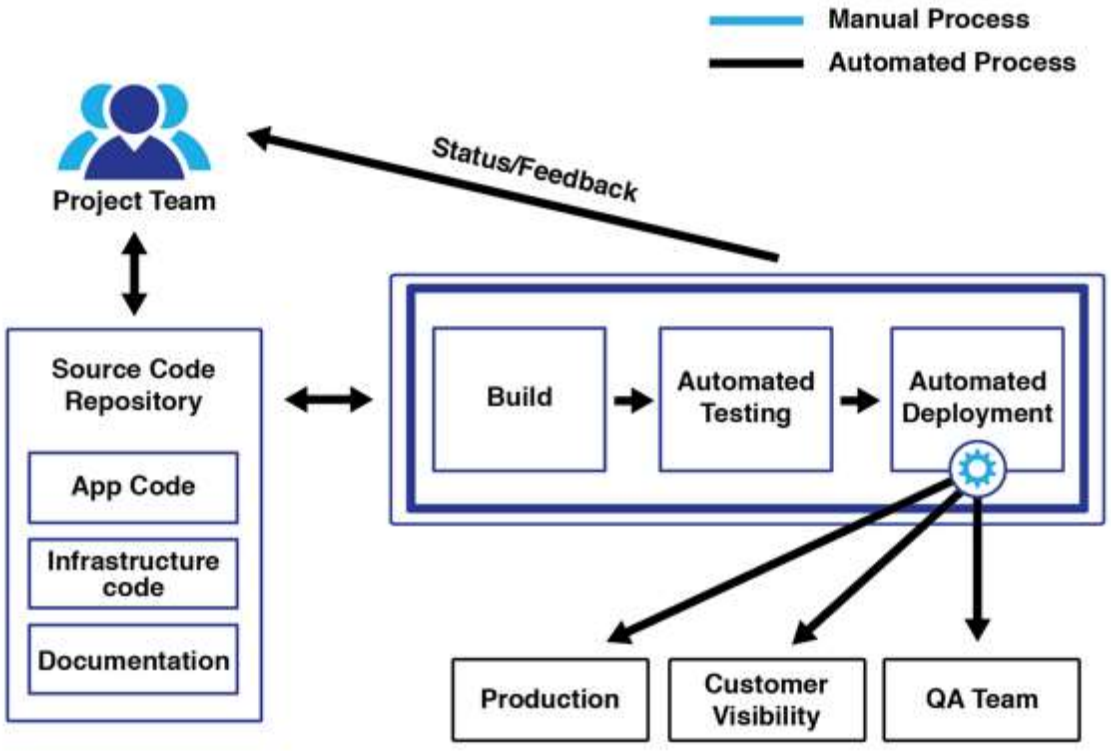
Infrastructure as Code (IaC)

A program that creates infrastructure,



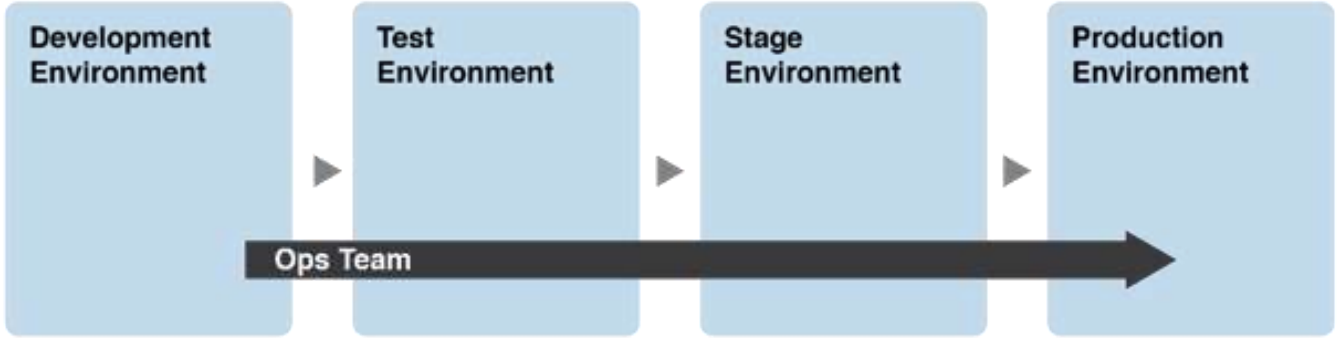
A concretely defined description of the environment is good material for conversation between team members.

Automation : *Continuous Integration (CI)*



Continuous integration is a process that continually merges a system's artifacts, including source code updates and configuration items from all stakeholders on a team, into a shared mainline to build and test the developed system.

Automation : *Continuous Delivery / Deployment (CD)*

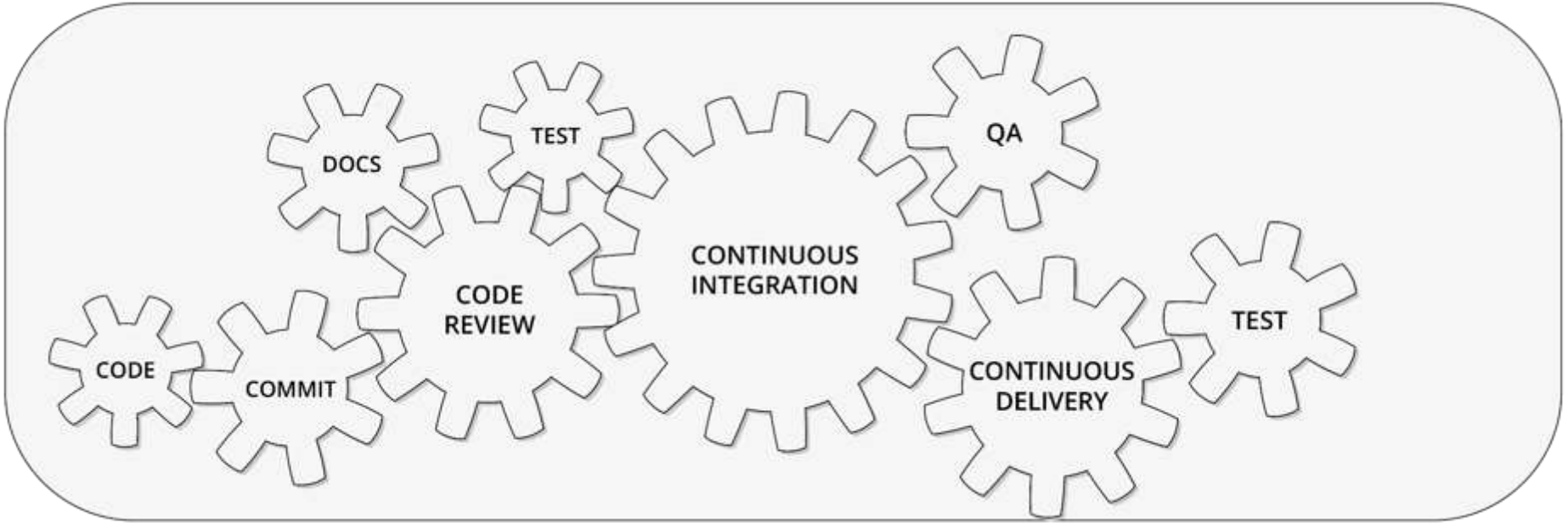


Shift Left Operational Concerns Enforced by Continuous Delivery with parity across various environment

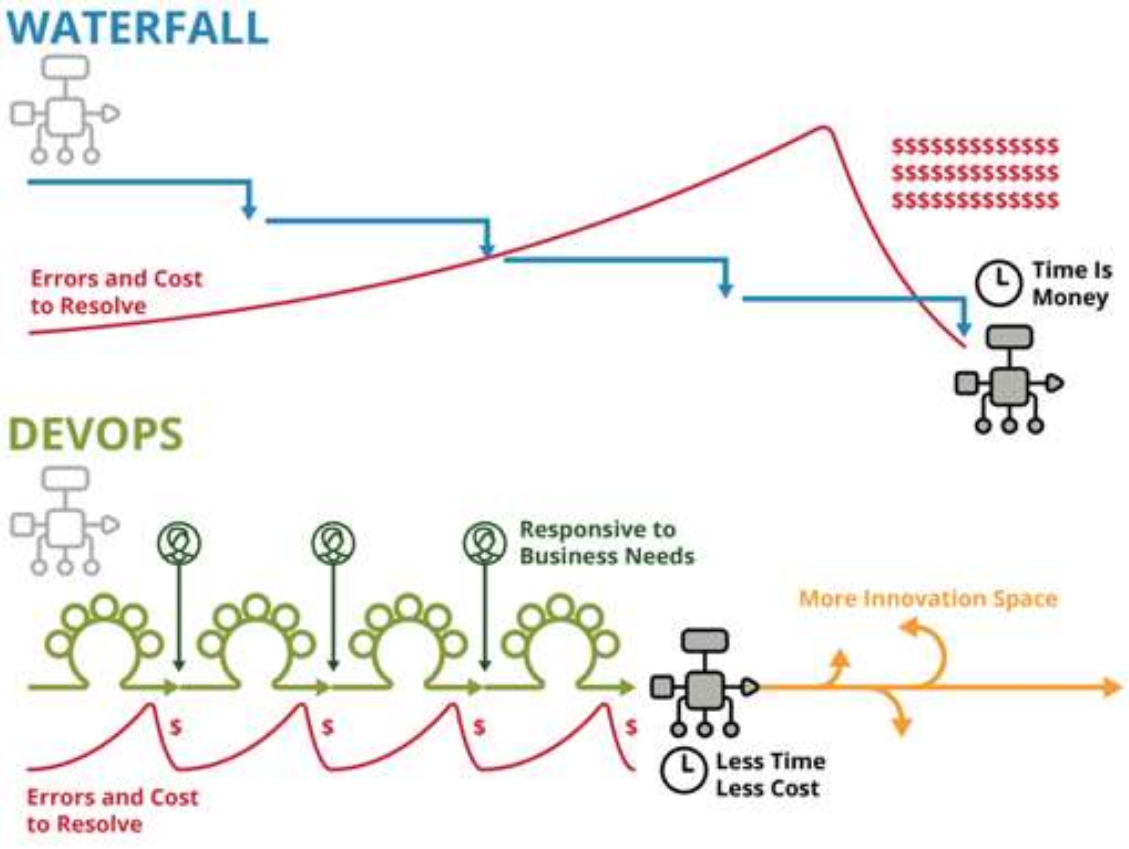
Continuous delivery is a software engineering practice that allows for frequent releases of new software to staging or various test environments through the use of automated testing.

Continuous deployment is the automated process of deploying changes to production by verifying intended features and validations to minimize risk.

Automation with IaC, CI, CD



Key Benefits of DevSecOps

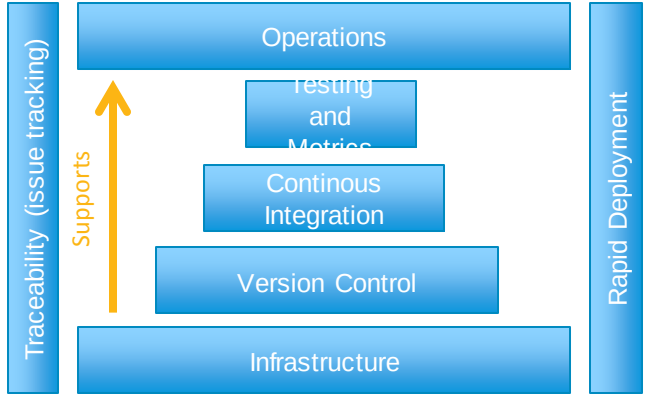


- Reduced errors during deployment
- Reduced time to deploy and resolve discovered errors
- **Repeatable** steps
- **Continuous availability** of pipeline and application
- Increased innovation time
- **Responsiveness** to business needs
- **Traceability** throughout the application lifecycle
- Increased stability and quality
- **Continuous feedback**
- **Continuous Security & cATO**

DevSecOps Pipeline

Pipeline Tool Landscape Complexity (~250 tools)

- Release configuration and release software (e.g., Puppet, Chef)
- Scripts and code used to release software (e.g., Python scripts)
- Servers, network or other infrastructure that support release tools
- Software and tools to support developer self-service operations
- External test frameworks (e.g., Jersey Test Framework)
- External operational monitoring and log mining tools (e.g., Splunk)
- Source code repositories (e.g., Gitlab)
- Issue tracking systems (e.g., JIRA)
- Container driven tools (e.g., Docker)
- Rqmts mgmt. (Doors, Blueprint)
- Infrastructure and cloud providers
- IDEs integrated DevOps process



Engineering the Deployment Pipeline is a *challenge*

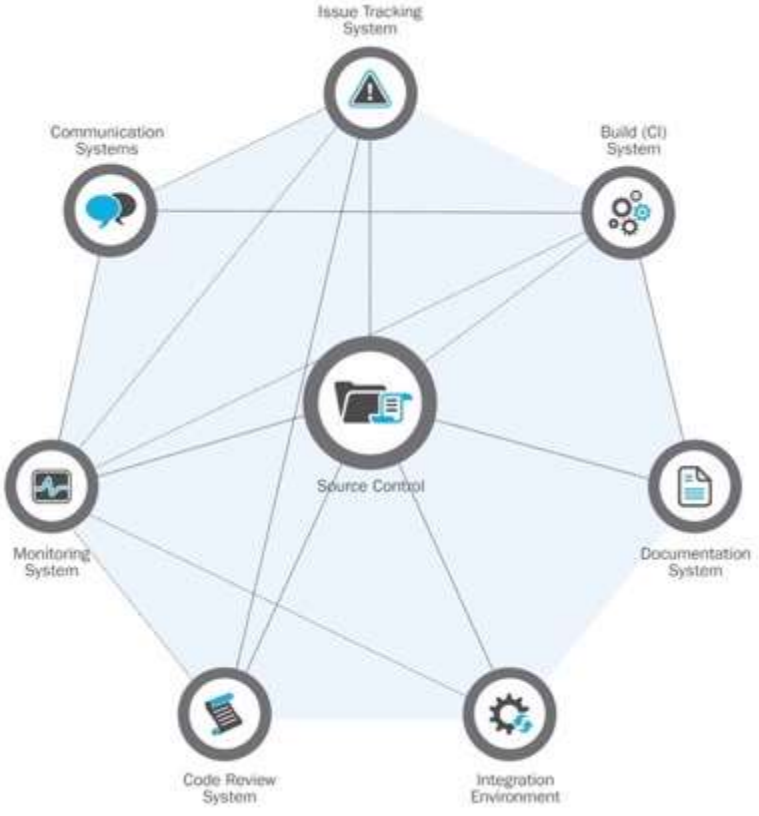
- If the pipeline is not engineered, it may require extensive effort to integrate tools and share data across the pipeline.
- Key questions related to designing the integrated pipeline include:
  - Who owns the integrated deployment pipeline?
  - How/what to measure/monitor to assess pipeline health?
  - What are the key qualities attributes teams should look for as they select tools for pipeline integration?
- Whether designing or buying, it is important to understand the end-to-end requirements (e.g., workflow visibility).

Integrated Pipeline Key Quality Attributes

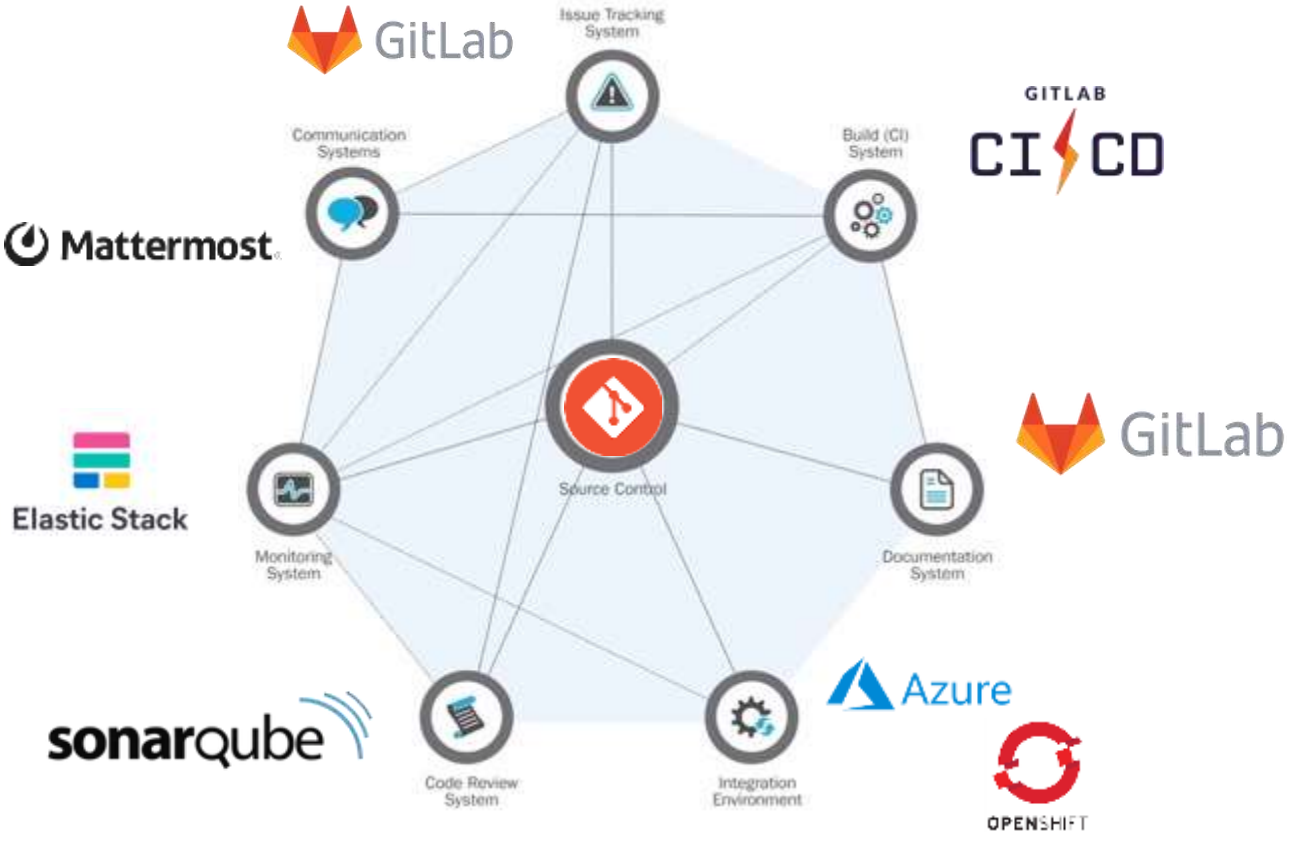
- Integrate-ability
- Interoperability
- Usability
- Portability
- Resilience
- Security/Permissions
- Availability (Error handling)
- Scalability

- Performance
- Modifiability
- Configurability
- “Automate-ability” (of manual tasks)
- “Approvability” (allows for manual approval)
- Measurability
- Others?

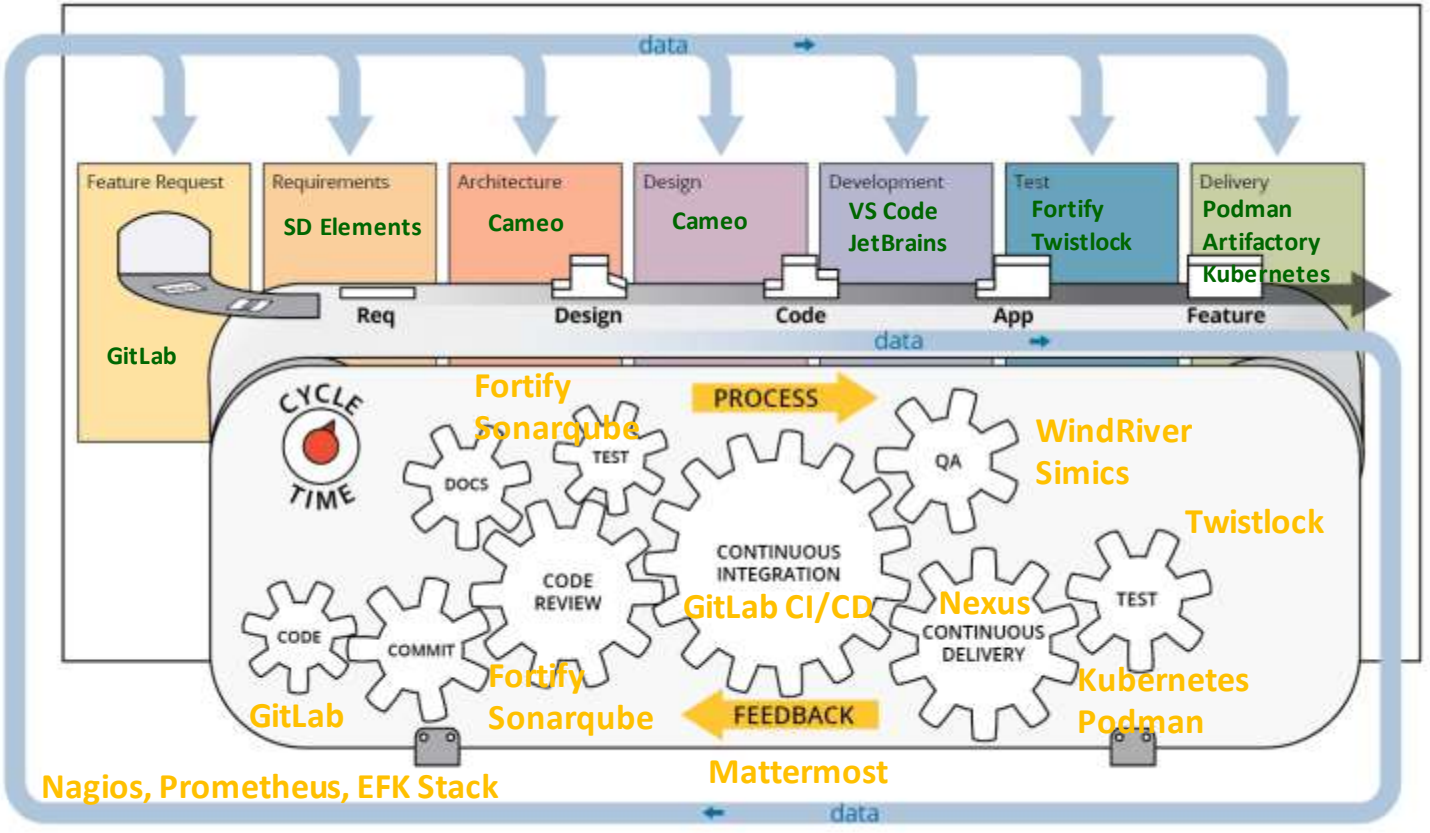
Integrated Pipeline - General



Integrated Pipeline - With Tooling



Exemplary DevOps Stack



App Lifecyle

CI/CD Automation

RedHat OpenShift/CoreOS

SEI Sandbox Environment

Carnegie Mellon University Software Engineering Institute

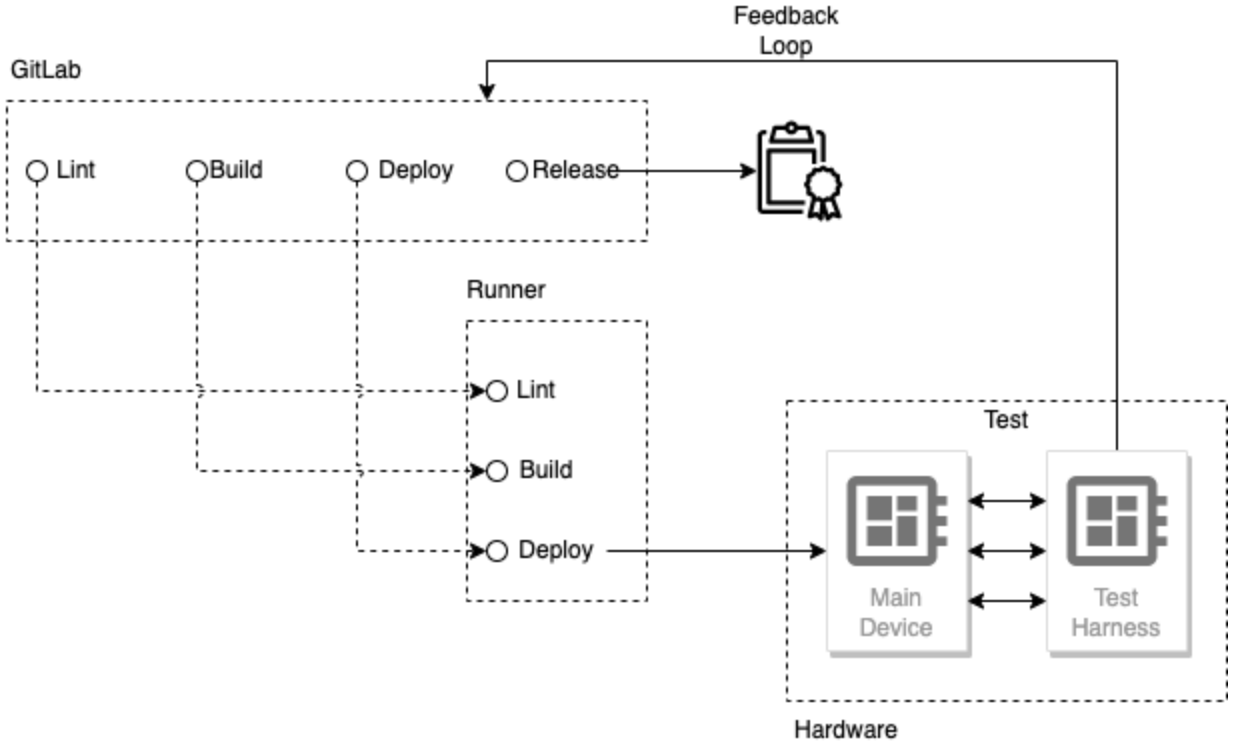
- An accessible environment for research and development
- Ongoing integration of tools, Infrastructure as Code and SOPs
- Provide parity across environments
- ”Lift and Shift” – provide a baseline for deployment within organization
- Used for comparison during Independent Verification and Validation
- Flexible environments
  - AWS
  - Azure
  - OpenShift (Air-gapped)

Demos

Pipeline Demo

- Infrastructure layer
- Threat modeling
- Build and scan code
- Generate SBOM
- Track SBOM
- Package, scan and push artifacts
- Deploy containers

Generic Pipeline with Hardware-in-the-Loop (HWIL)



Challenges in DSO with HWIL

- Many programs have unforeseen difficulties in coordinating SW development with HW deployment and testing.
  - Most scenarios covered by COTS tools do not include HW
  - Can simulate HW, but cannot deploy and test
  - HW Feedback-loop is laborious

- Unavailability of professionals with expertise that encompasses whole breadth of development.

- Complexity of HW increases the coordination effort greatly

Demo

Thank you

Hasan Yasar
Technical Director
Continuous Deployment of Capability
hyasar@cmu.edu

Jeffrey Hamed
DevOps Software Engineer
Continuous Deployment of Capability
jhamed@sei.cmu.edu

Patrick Earl
Associate DevOps Engineer
Continuous Deployment of Capability
pearl@sei.cmu.edu

David Shepard
MTS - Senior Engineer
Continuous Deployment of Capability
djshepard@sei.cmu.edu

Luiz Antunes
DevOps Software Engineer
Continuous Deployment of Capability
lantunes@sei.cmu.edu

MONTH 00, 2023

Backup Slides

Carnegie Mellon University
Software Engineering Institute