Dev[Sec]Ops – A Whirlwind Tour

Tools, Practices for building secure Applications

A quick overview of Dev[Sec]Ops and tools, practices and samples for building secure applications.

Hands on lab sessions for getting started with tools and overview of incorporating secure practices in your pipelines

Presented by – Seshagiri Sriram

What We will be covering

- Session 1 - Introduction to CI/CD and advanced CI/CD pipelines
  - Multi-stage pipelines
  - Parallel and sequential jobs
  - Integration with Jenkins and GitLab
- Session 2 - Infrastructure as Code
  - Concepts and benefits
  - Terraform
  - Ansible
  - CloudFormation
  - Code Modularity and Re-Use
  - States and Secret Management
  - Provisioning Infrastructure using IAC

What We will be covering

- Session 3 – Containerization and Orchestration
  - Docker and Docker Compose
  - Multi-stage Builds

- Session 4 – Getting Started with Kubernetes
  - Architecture and Components
  - Deployment and Managing Applications

What We will be covering

- Session 5 – Monitoring and Logging
  - Introduction to Grafana, Prometheus, and ELK stack
  - Centralized Logging
  - Metrics and Metrics Scrapping
  - Monitoring Kubernetes Pods with ELK

- Session 6 – Review !!

What We will be covering

- Session 7 – Security in DevOps
  - Coding Guidelines & Best Practices
  - Microservices and REST API
  - Input and Output Validation
  - Authentication and Password Management
  - Authorization
  - Session Management
  - Errors and Exception Handling
  - Logging
  - API Security

What We will be covering

- Session 8 – Configuration Management
  - Chef, Puppet and Ansible
  - Multi-environment Configuration
  - Playbooks
  - Managing Secrets and Sensitive Information
- Session 9 – GitOps and Continuous Deployment
  - Introduction and Benefits
  - ArgoCD Introduction
  - Implementing GitOps

What We will be covering

- Session 10 – Advanced Scripting
  - Integrating Scripting with CI/CD pipelines
  - Shell Scripting
  - GitHub Actions

- Session 11 – Advanced Configuration Management
  - Deeper drive into Ansible
  - State and Secret Management

About me

25+ Years Exp in IT as Architect, Project Manager, Tech. Change Management and IT Training and Education.

Java/DotNet/RDBMS/NoSQL/Messaging/Enterprise Integration/APIs

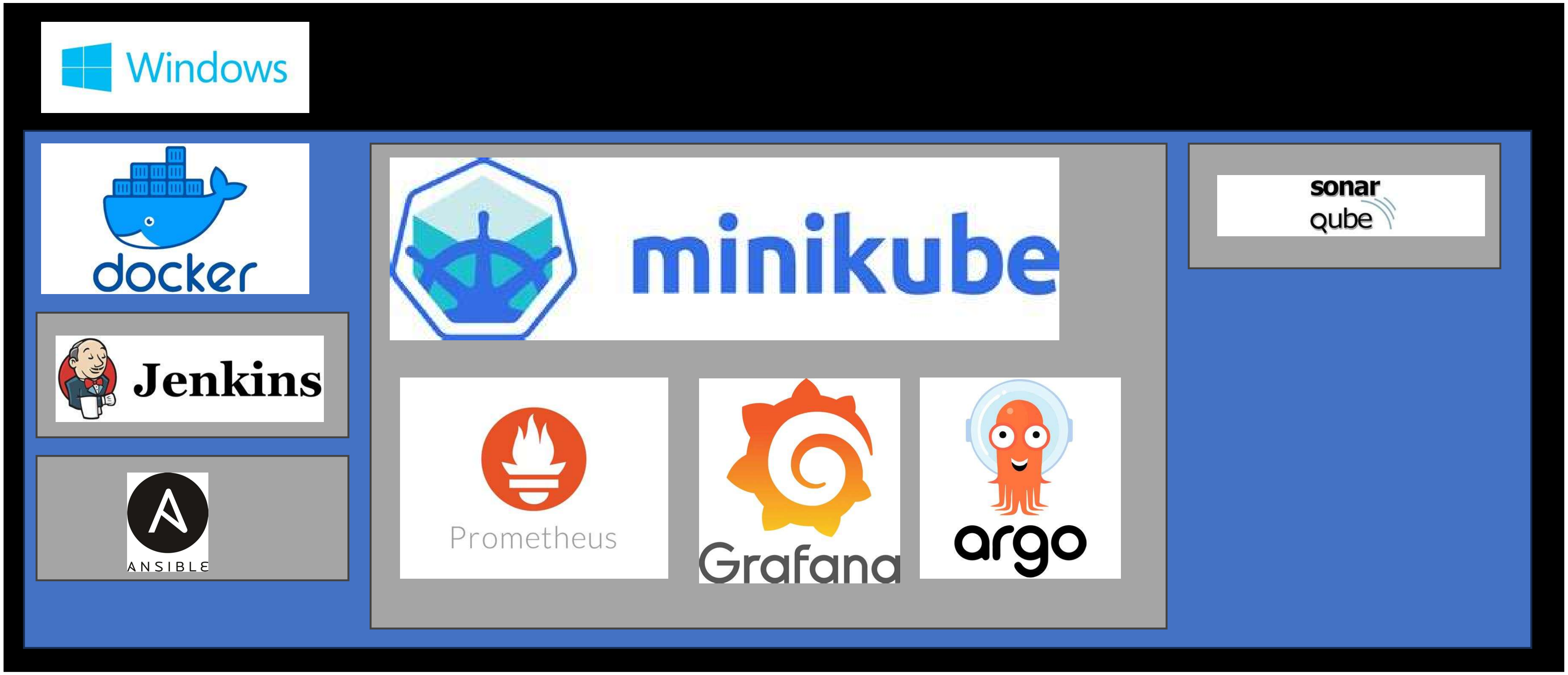
E·FUNDS℠
Make every transaction more valuable℠

R SYSTEMS

The Lab Environment

What tools will we use?

An Overview


Windows
docker
Jenkins
ANSIBLE
minikube
Prometheus
Grafana
argo
sonar qube

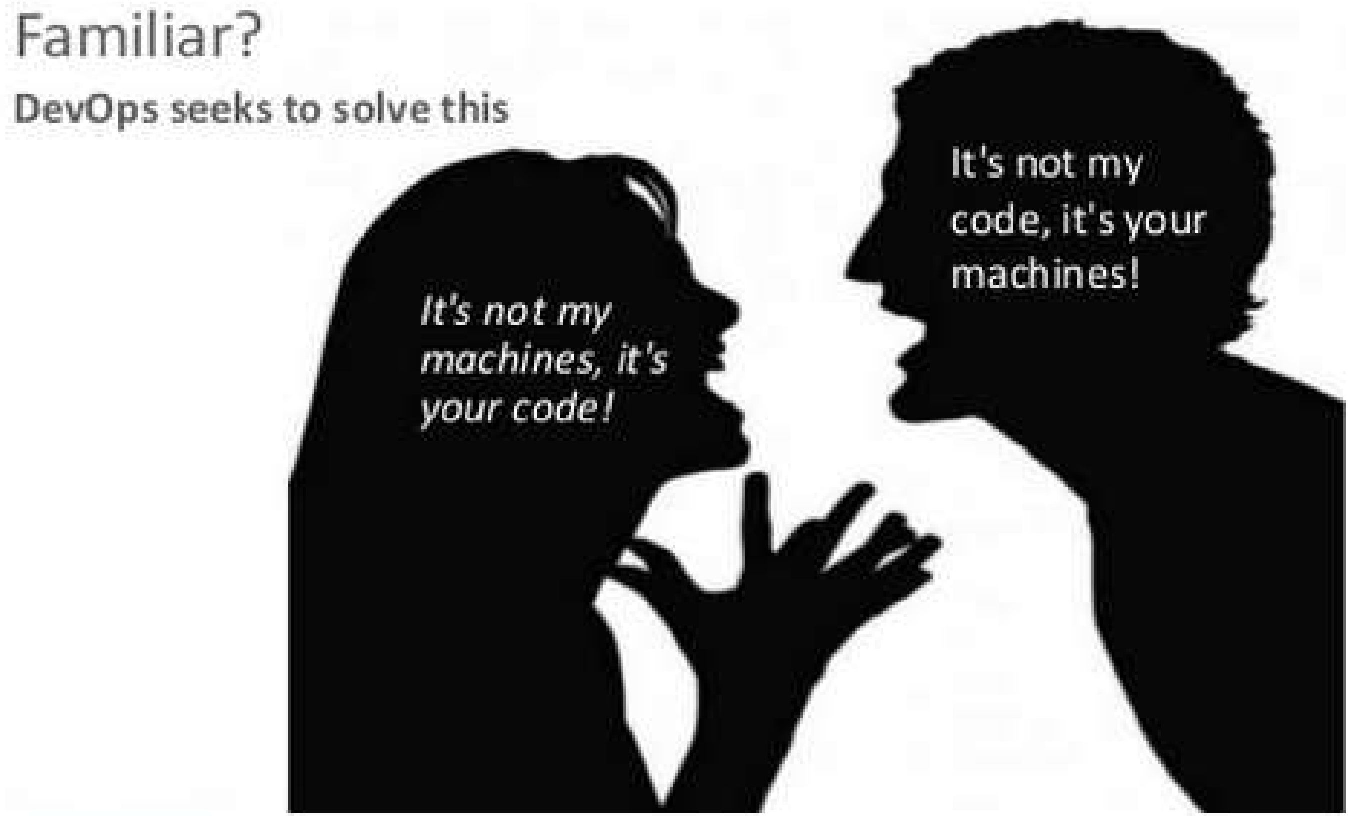
An Overview

- Not shown but will be used
  - GitHub
  - GitLab
  - Helm
  - ‘kubectl / kubeadm
- Optional and Nice to Have
  - Oracle Virtual Box / Vagrant (to build k8s cluster with 1 control pane and 2 nodes)
- Other pre-requisites
  - Credentials for GitHub / GitLab
  - Docker Hub Credentials

Devops and CI/CD

A quick introduction to Devops, CI/CD and Pipelines

Devops

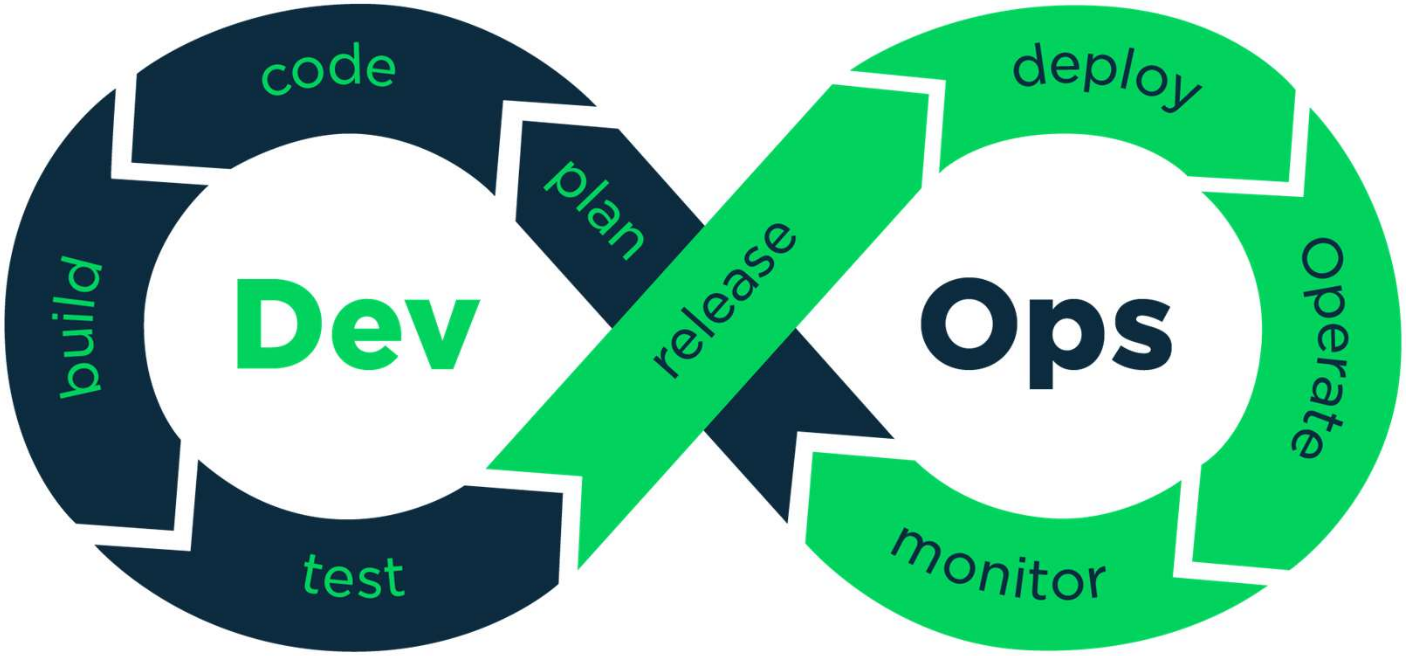


Familiar?

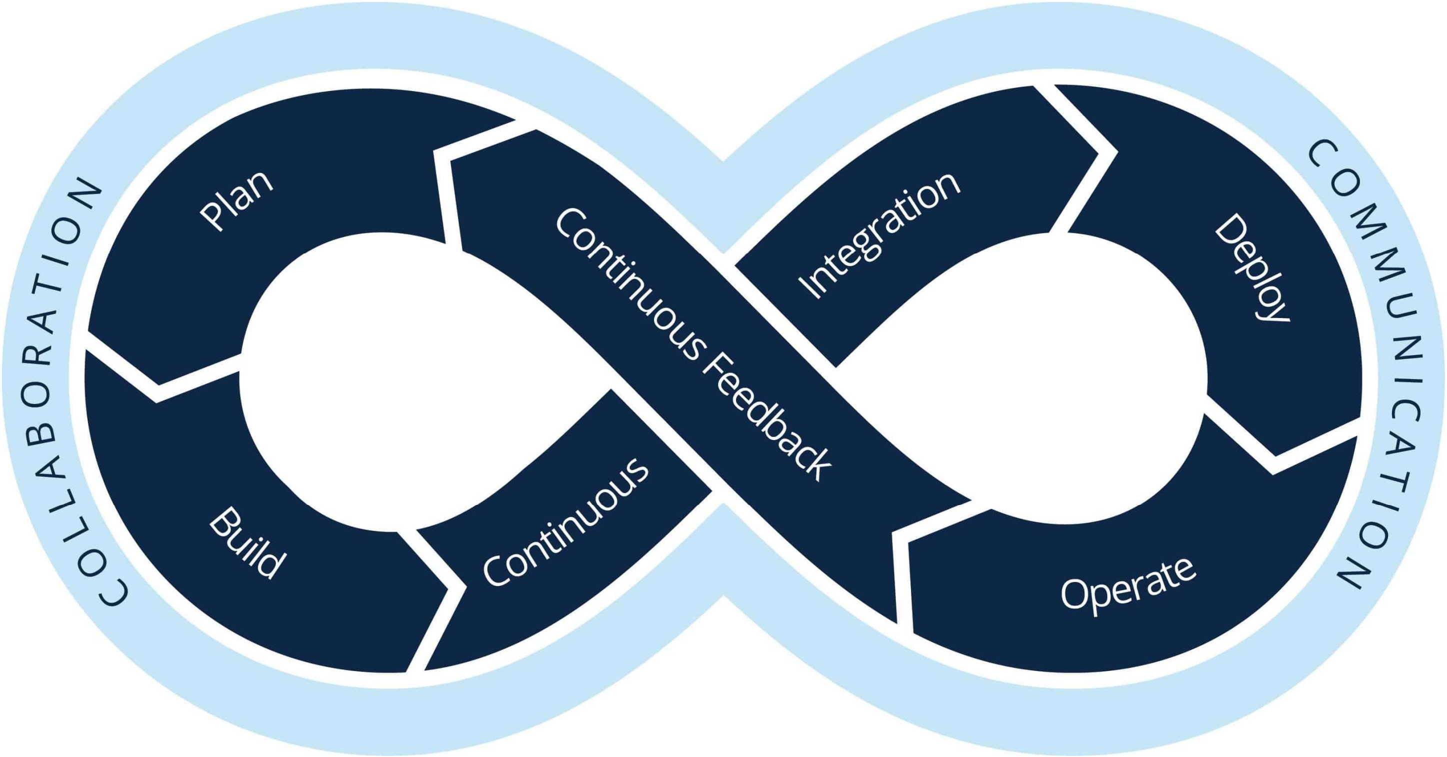
DevOps seeks to solve this

It's not my machines, it's your code!
It's not my code, it's your machines!

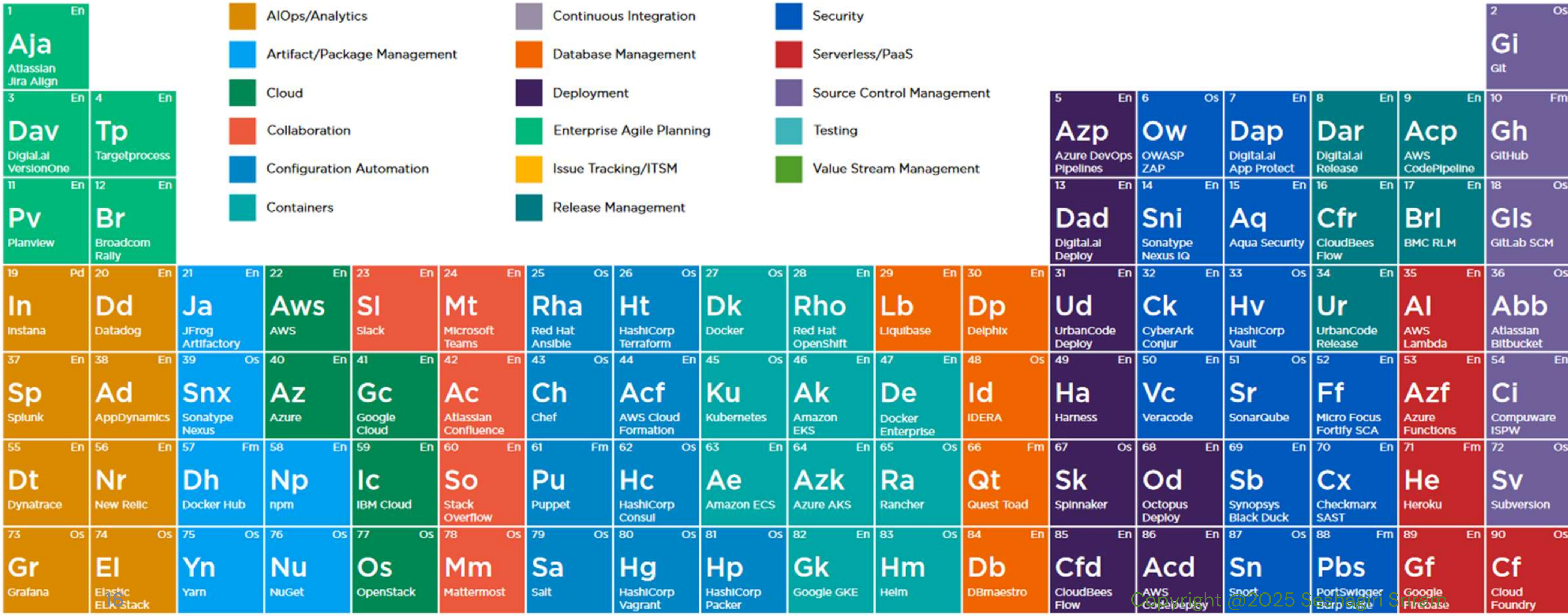
The Devops Cycle



The Devops Cycle



Devops Tools

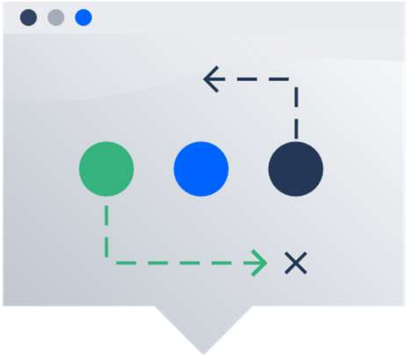


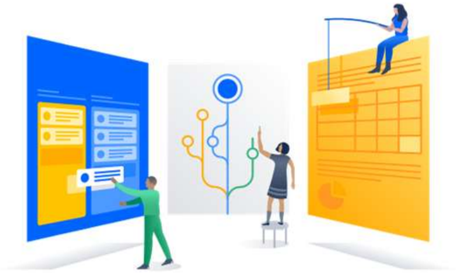
The Periodic Table of DevOps Tools (V4)

- 🟧 AIOps/Analytics
- 🟦 Artifact/Package Management
- 🟩 Cloud
- 🟥 Collaboration
- 🟦 Configuration Automation
- 🟦 Containers
- ⬜ Continuous Integration
- 🟧 Database Management
- 🟪 Deployment
- 🟩 Enterprise Agile Planning
- 🟨 Issue Tracking/ITSM
- 🟦 Release Management
- 🟦 Security
- 🟥 Serverless/PaaS
- 🟪 Source Control Management
- 🟦 Testing
- 🟩 Value Stream Management

1 En Aja Atlassian Jira Align																	2 Os Gi Git
3 En Dav Digital.ai VersionOne	4 En Tp Targetprocess											5 En Azp Azure DevOps Pipelines	6 Os Ow OWASP ZAP	7 En Dap Digital.ai App Protect	8 En Dar Digital.ai Release	9 En Acp AWS CodePipeline	10 Fm Gh GitHub
11 En Pv Planview	12 En Br Broadcom Rally											13 En Dad Digital.ai Deploy	14 En Sni Sonatype Nexus IQ	15 En Aq Aqua Security	16 En Cfr CloudBees Flow	17 En Brl BMC RLM	18 Os Gls GitLab SCM
19 Pd In Instana	20 En Dd Datadog	21 En Ja JFrog Artifactory	22 En Aws AWS	23 En Sl Slack	24 En Mt Microsoft Teams	25 Os Rha Red Hat Ansible	26 Os Ht HashiCorp Terraform	27 Os Dk Docker	28 En Rho Red Hat OpenShift	29 En Lb Liquibase	30 En Dp Delphix	31 En Ud UrbanCode Deploy	32 En Ck CyberArk Conjur	33 Os Hv HashiCorp Vault	34 En Ur UrbanCode Release	35 En Al AWS Lambda	36 Os Abb Atlassian Bitbucket
37 En Sp Splunk	38 En Ad AppDynamics	39 Os Snx Sonatype Nexus	40 En Az Azure	41 En Gc Google Cloud	42 En Ac Atlassian Confluence	43 Os Ch Chef	44 En Acf AWS Cloud Formation	45 Os Ku Kubernetes	46 En Ak Amazon EKS	47 En De Docker Enterprise	48 Os Id IDERA	49 En Ha Harness	50 En Vc Veracode	51 Os Sr SonarQube	52 En Ff Micro Focus Fortify SCA	53 En Azf Azure Functions	54 En Ci Compuware ISPW
55 En Dt Dynatrace	56 En Nr New Relic	57 Fm Dh Docker Hub	58 En Np npm	59 En Ic IBM Cloud	60 En So Stack Overflow	61 Fm Pu Puppet	62 Os Hc HashiCorp Consul	63 En Ae Amazon ECS	64 En Azk Azure AKS	65 Os Ra Rancher	66 Fm Qt Quest Toad	67 Os Sk Spinnaker	68 En Od Octopus Deploy	69 En Sb Synopsys Black Duck	70 En Cx Checkmarx SAST	71 Fm He Heroku	72 Os Sv Subversion
73 Os Gr Grafana	74 Os El Elastic ELK Stack	75 Os Yn Yarn	76 Os Nu NuGet	77 Os Os OpenStack	78 Os Mm Mattermost	79 Os Sa Salt	80 Os Hg HashiCorp Vagrant	81 Os Hp HashiCorp Packer	82 En Gk Google GKE	83 Os Hm Helm	84 En Db DBmaestro	85 En Cfd CloudBees Flow	86 En Acd AWS CodeDeploy	87 Os Sn Snort	88 Fm Pbs PortSwigger Burp Suite	89 En Gf Google Firebase	90 Os Cf Cloud Foundry

The 3 Ways





Systems Thinking

Recognize that Software Applicatons are complex systens

Amplifying Feedback Loops

Improve Bi-Directional Communication between team members

Cultural Change

Culture that encourages continuos experimentation and learning

Key Devops Principles

Collaboration

- Full Stack Development
- Fully Functional Team

Automation

- Key in CI/CD Pipelines
- IAC
- Notifications

Key Devops Principles

Continuous Improvement

- Focus on
  - Experimentation
  - Minimize Waste
  - Optimize speed and Cost

Key Devops Principles

Customer Centric Action

- Short Feedback loops
- Handle User Feedbacks
  - In Real Time ☺

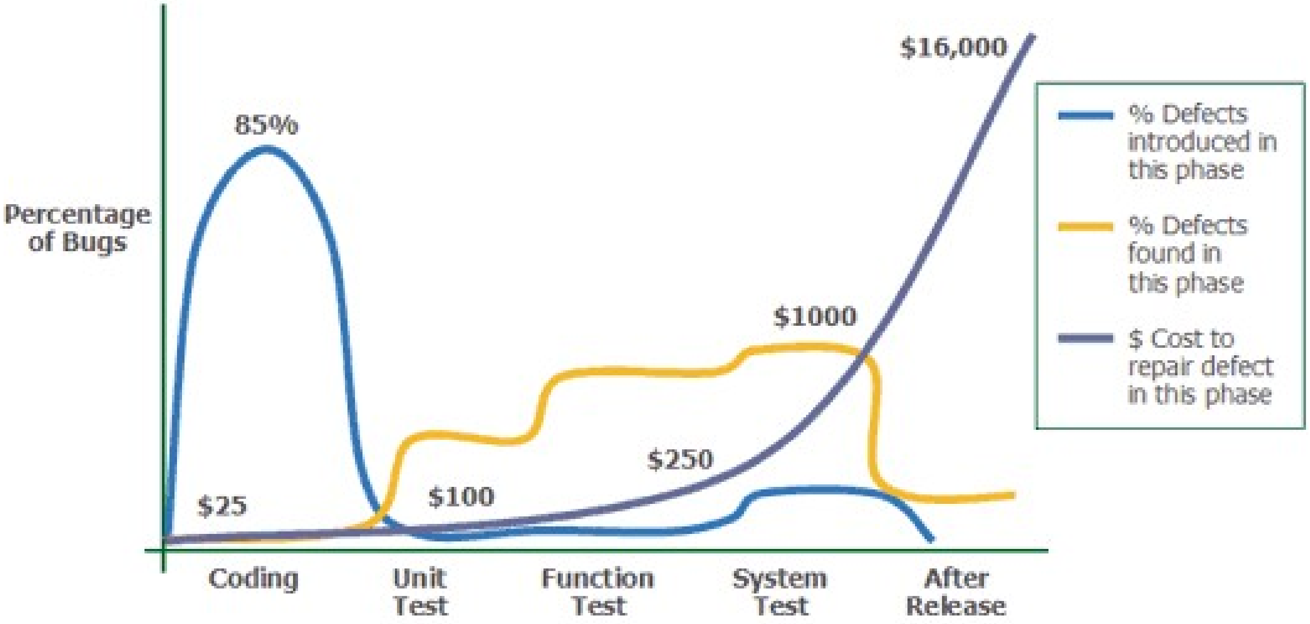
Keep the final goal in mind

- Review Assumptions
- Listen to the end user or customer
- Your work is **not** done when a module is complete

What is CI/CD

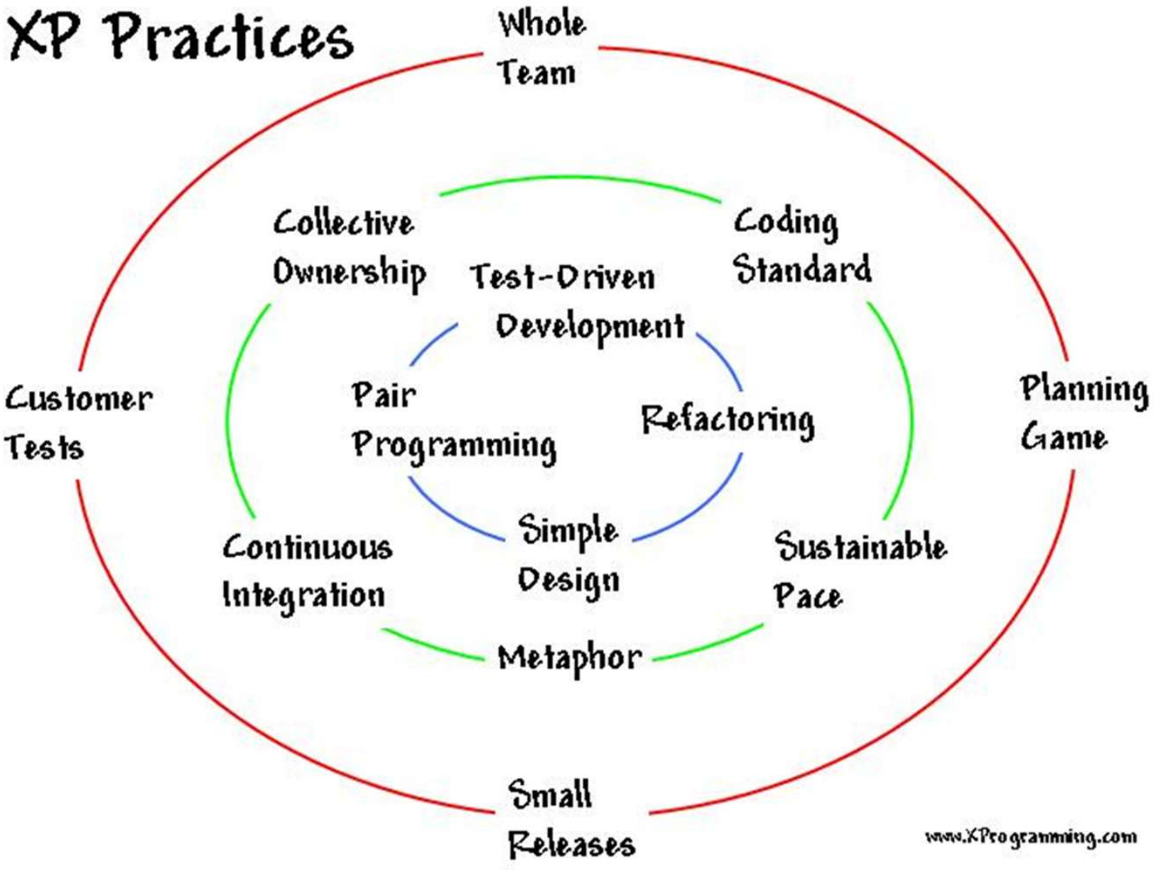
- Martin Fowler defines Continuous Integration (CI) as
- *A **Software Development Process** where*
- ***Every team member** merge their changes along*
- *With changes from **other team members***
- ***At least once daily***
- ***Each** of these integrations are verified by **automated builds***
- ***(including Tests)***
- *To **detect errors early** on in the process*

Do we have data?

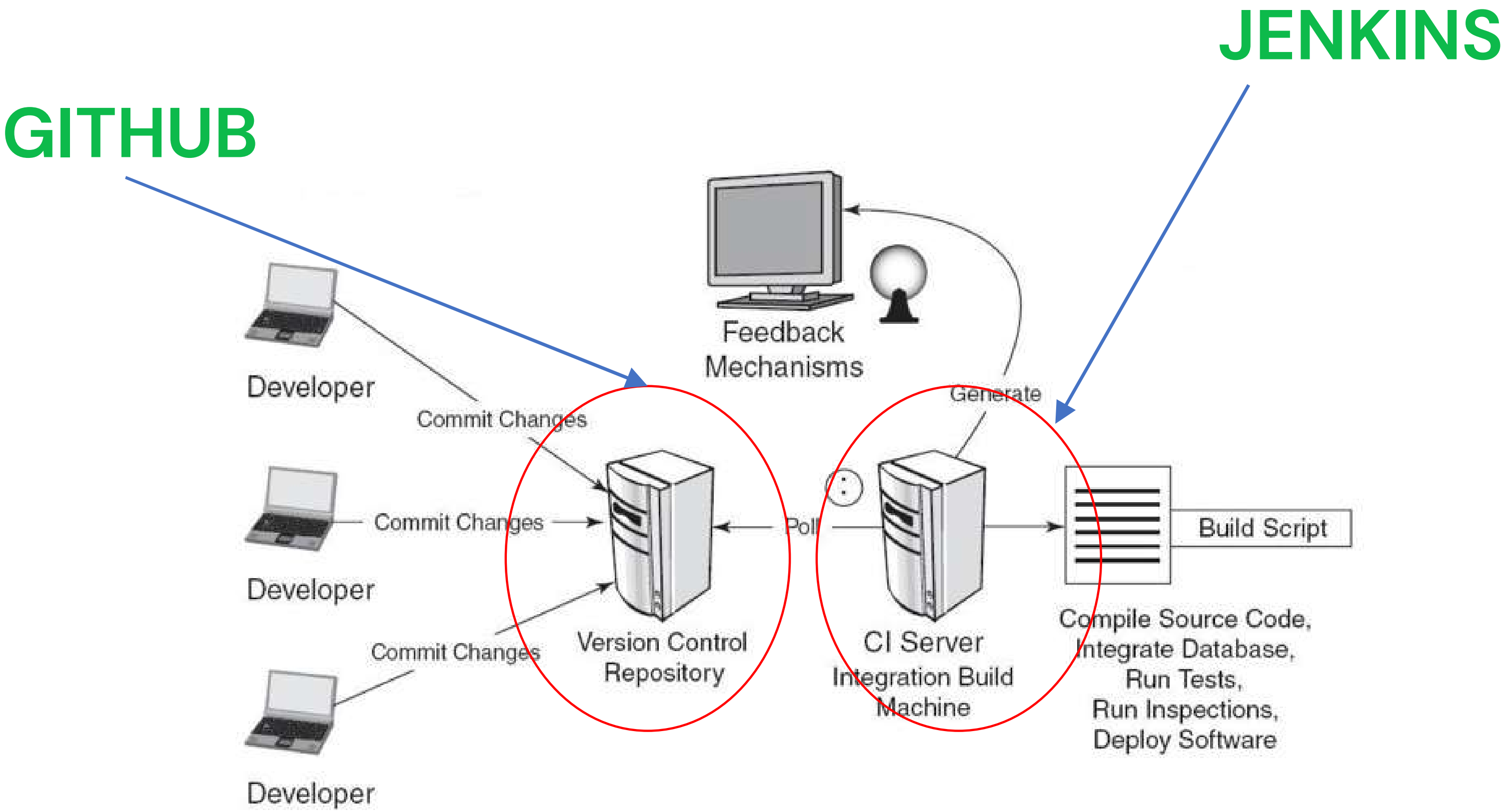


Source: http://www.agitar.com/solutions/why_unit_testing.html

CI – The origins



A basic flow for CI

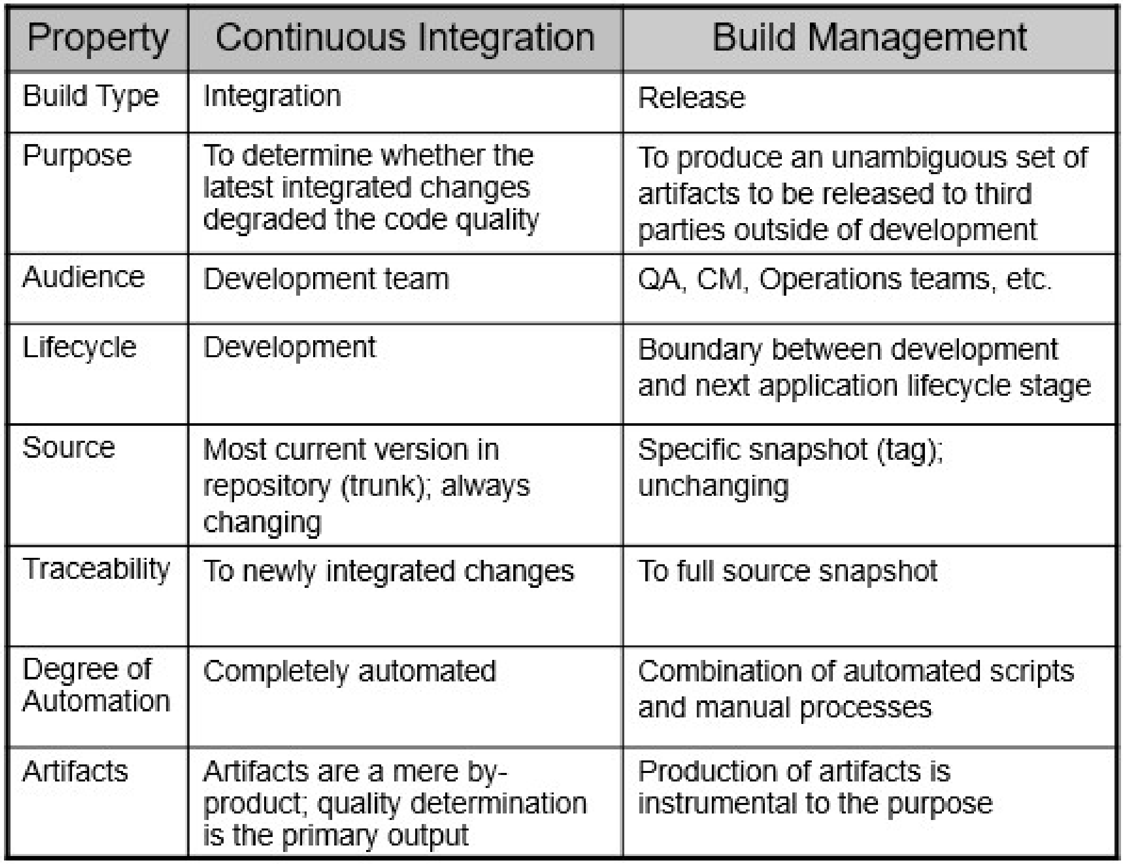


GITHUB

JENKINS

Feedback Mechanisms
Developer
Commit Changes
Developer
Commit Changes
Commit Changes
Developer
Version Control Repository
Poll
Generate
CI Server Integration Build Machine
Build Script
Compile Source Code, Integrate Database, Run Tests, Run Inspections, Deploy Software

CI vs Build Management

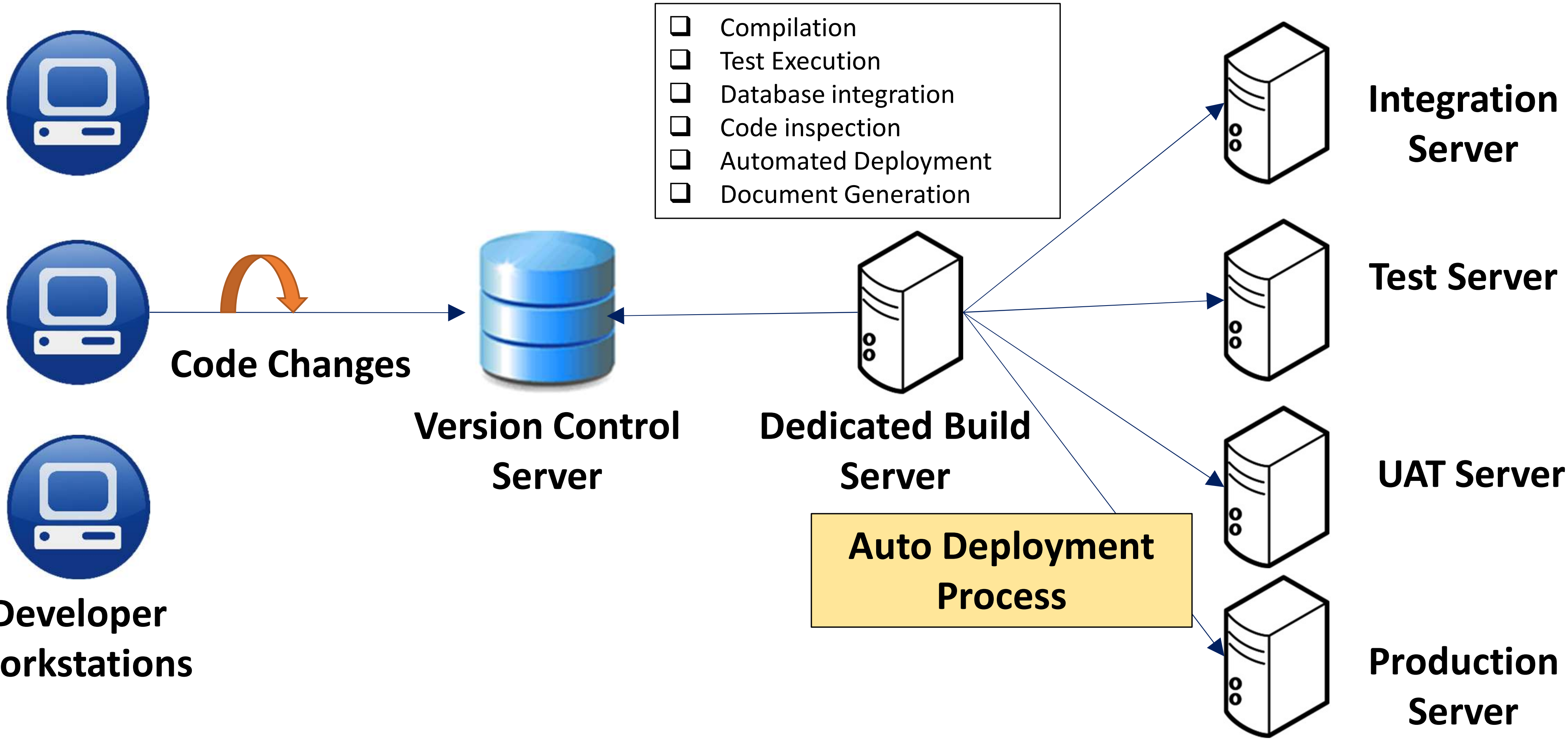


Property	Continuous Integration	Build Management
Build Type	Integration	Release
Purpose	To determine whether the latest integrated changes degraded the code quality	To produce an unambiguous set of artifacts to be released to third parties outside of development
Audience	Development team	QA, CM, Operations teams, etc.
Lifecycle	Development	Boundary between development and next application lifecycle stage
Source	Most current version in repository (trunk); always changing	Specific snapshot (tag); unchanging
Traceability	To newly integrated changes	To full source snapshot
Degree of Automation	Completely automated	Combination of automated scripts and manual processes
Artifacts	Artifacts are a mere by-product; quality determination is the primary output	Production of artifacts is instrumental to the purpose

CI is not just Compiling Code!

- CI is **not** the same as a build
- The latter term is used for compilation only
- CI includes
  - Compilation
  - Running Unit tests
  - Generating Documents and Reports
  - Integrating Databases
  - Running System/Integration/Regression/Security Test
  - Integrate with **C**ontinuous **D**elivery (**CD**) for deployment

Continuous Integration Basics



❑ Compilation
❑ Test Execution
❑ Database integration
❑ Code inspection
❑ Automated Deployment
❑ Document Generation

Developer Workstations

Code Changes

Version Control Server

Dedicated Build Server

Auto Deployment Process

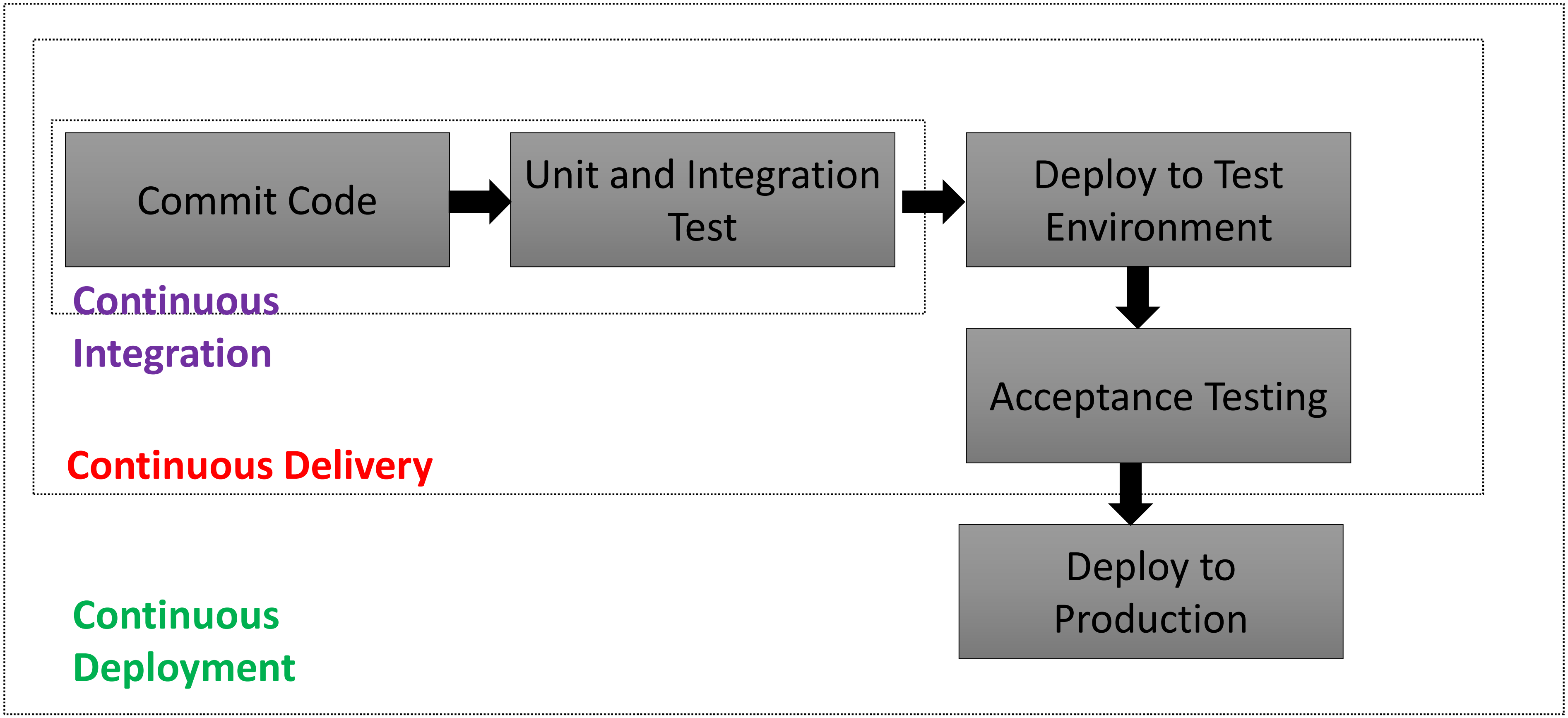
Integration Server

Test Server

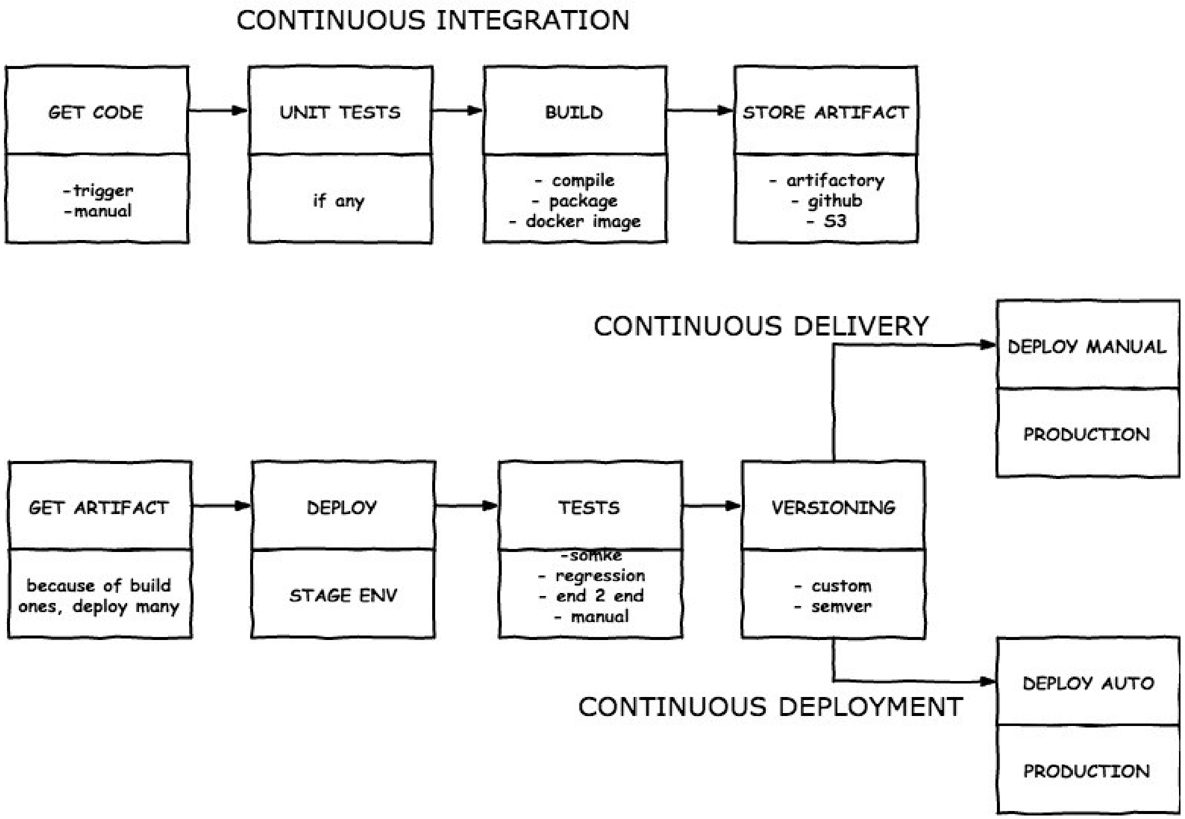
UAT Server

Production Server

Simplified CI/CD Pipeline



CI and CD



CONTINUOUS INTEGRATION

GET CODE
- trigger
- manual

UNIT TESTS
if any

BUILD
- compile
- package
- docker image

STORE ARTIFACT
- artifactory
- github
- S3

CONTINUOUS DELIVERY

DEPLOY MANUAL
PRODUCTION

GET ARTIFACT
because of build ones, deploy many

DEPLOY
STAGE ENV

TESTS
-somke
- regression
- end 2 end
- manual

VERSIONING
- custom
- semver

CONTINUOUS DEPLOYMENT

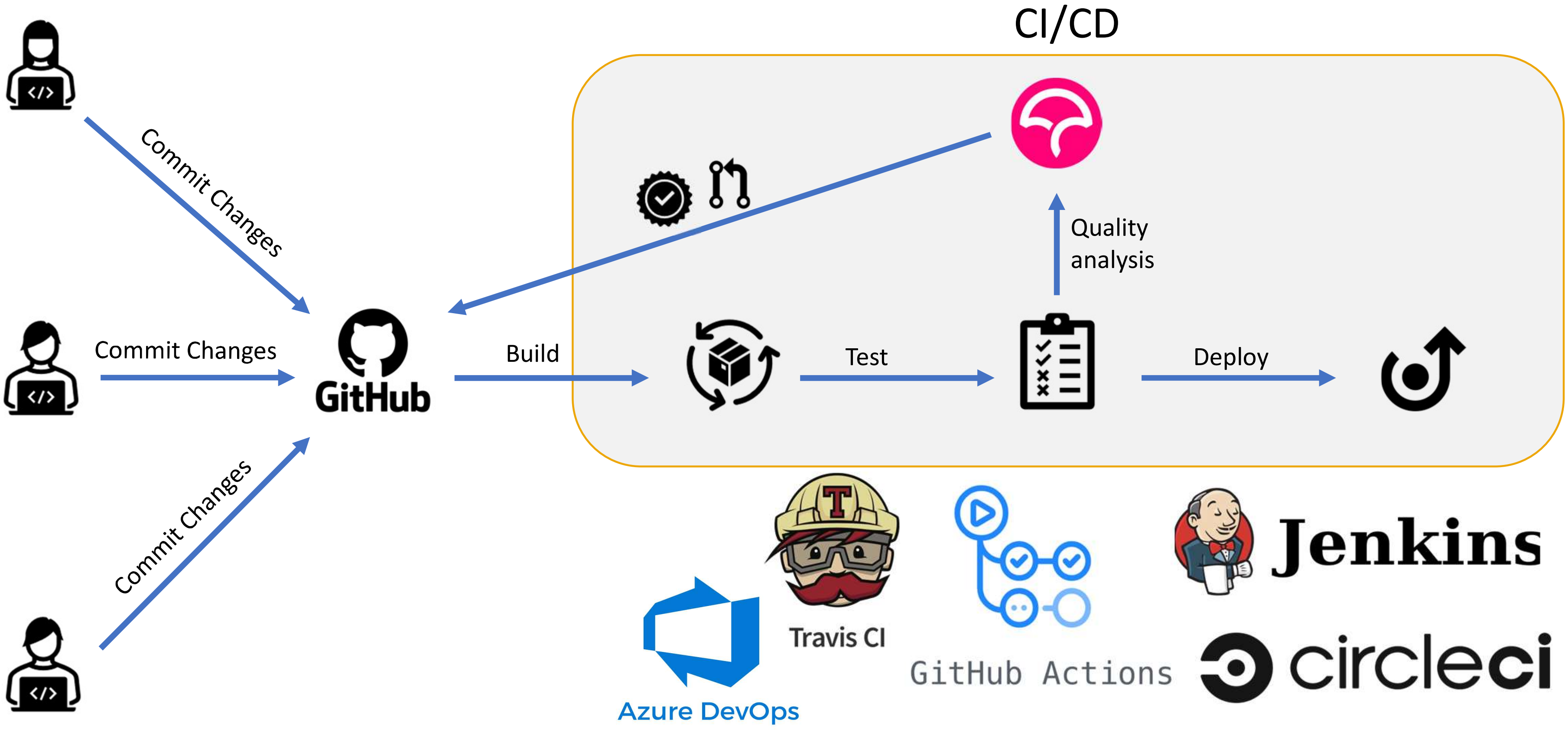
DEPLOY AUTO
PRODUCTION

Bamboo

...

Copyright @2025 Seshagiri Sriram

Collaborative software development



CI/CD

Commit Changes

Commit Changes

Commit Changes

GitHub

Build

Test

Deploy

Quality analysis

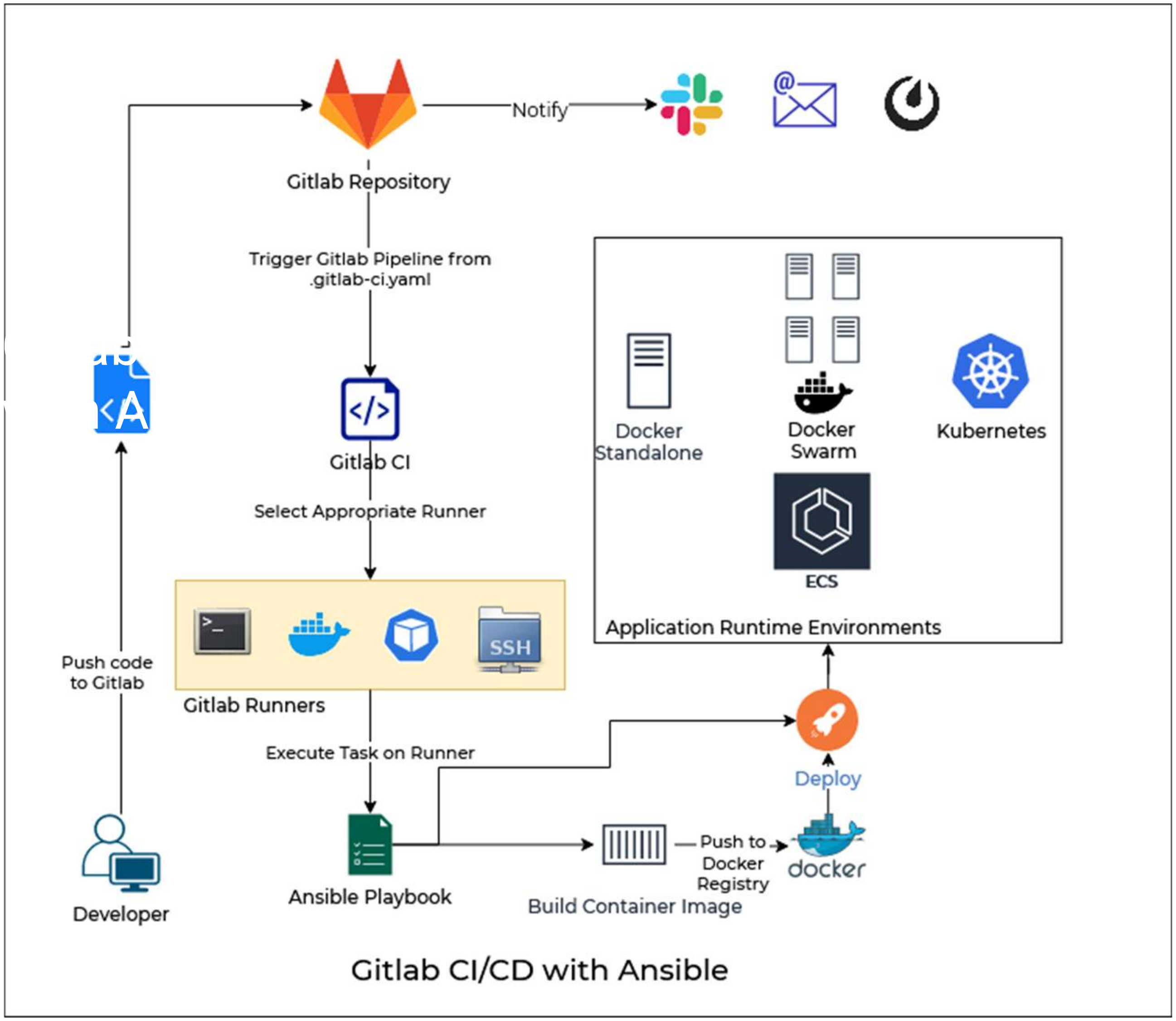
Travis CI

GitHub Actions

Jenkins

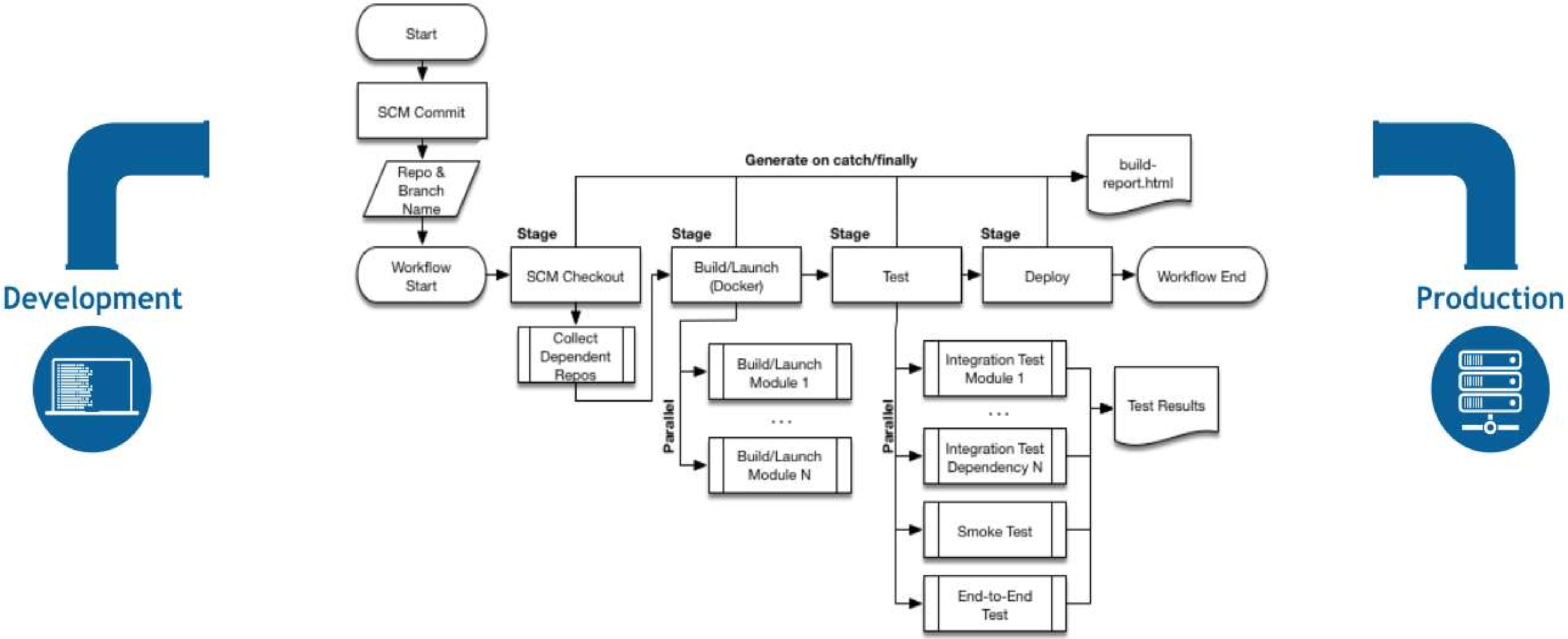
circleci

Azure DevOps



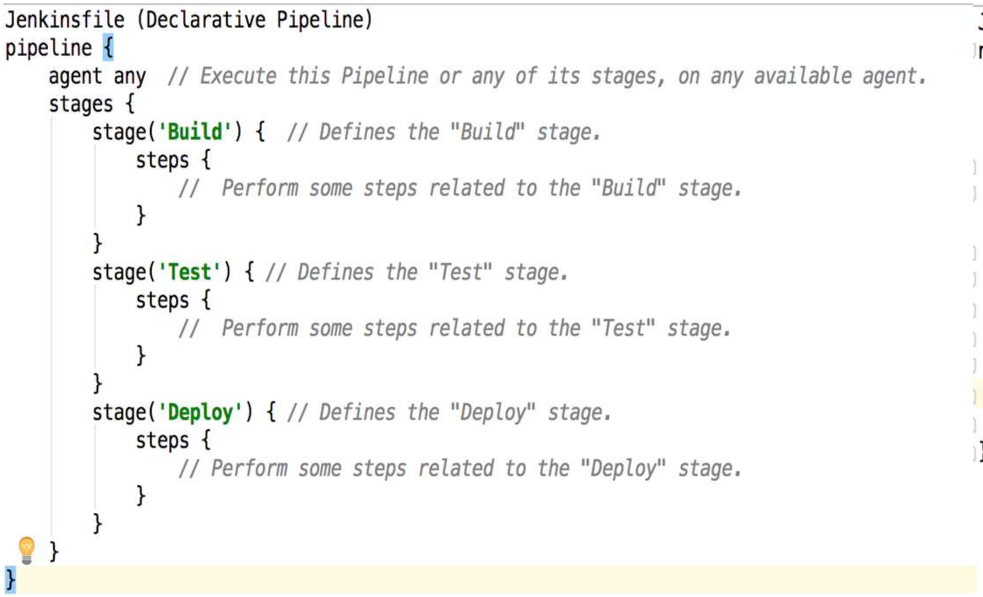
Gitlab CI/CD with Ansible

Jenkins: Pipelines

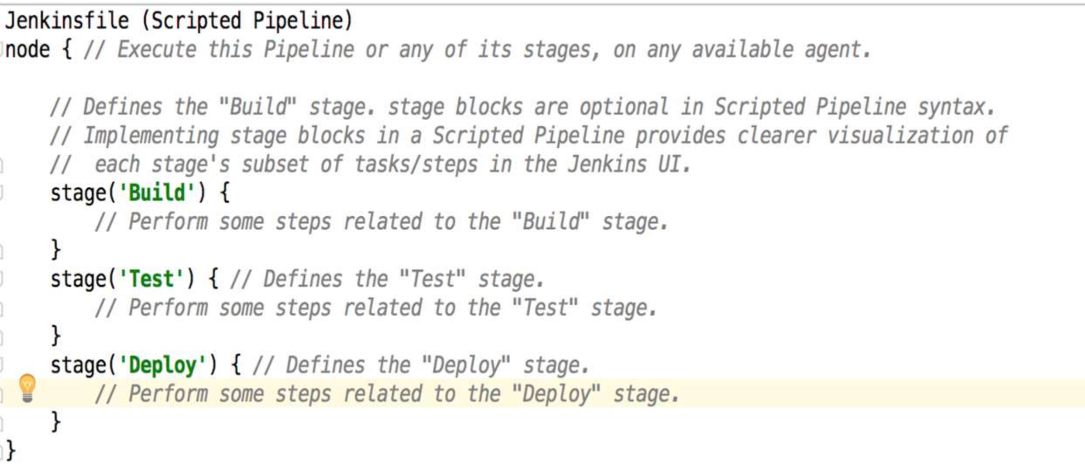


Jenkins: Pipelines. Concepts

Pipeline – service syntax (optional, declarative implementation)
Node – where to run
Stage – Group of action/tasks
Step – what to do



```
Jenkinsfile (Declarative Pipeline)
pipeline {
    agent any  // Execute this Pipeline or any of its stages, on any available agent.
    stages {
        stage('Build') {  // Defines the "Build" stage.
            steps {
                //  Perform some steps related to the "Build" stage.
            }
        }
        stage('Test') { // Defines the "Test" stage.
            steps {
                //  Perform some steps related to the "Test" stage.
            }
        }
        stage('Deploy') { // Defines the "Deploy" stage.
            steps {
                // Perform some steps related to the "Deploy" stage.
            }
        }
    }
}
```



```
Jenkinsfile (Scripted Pipeline)
node { // Execute this Pipeline or any of its stages, on any available agent.

    // Defines the "Build" stage. stage blocks are optional in Scripted Pipeline syntax.
    // Implementing stage blocks in a Scripted Pipeline provides clearer visualization of
    //  each stage's subset of tasks/steps in the Jenkins UI.
    stage('Build') {
        // Perform some steps related to the "Build" stage.
    }
    stage('Test') { // Defines the "Test" stage.
        // Perform some steps related to the "Test" stage.
    }
    stage('Deploy') { // Defines the "Deploy" stage.
        // Perform some steps related to the "Deploy" stage.
    }
}
```

Which one?

- GITLAB/GITHUB started as Pure Version Control Systems
- Now they provide CI/CD support in form of YAML files
- The syntax is different – but in essence, these are declarative
- For more details, refer to the Documentation provided by GitLab and GitHub
- **For the demo purpose**, we will show how to do this with Jenkins and GitLab.

Notes on Demo

- We will be showing both multi-stage and Parallel jobs
- Demo on both Jenkins as well as on GITLAB (for ||)
- Difference will be pointed out during demo.
- Sample Code can be obtained here:
  - git@github.com:SeshagiriSriram/DemoRepos.git
  - https://gitlab.com/seshagirisriram1/mycooldotnetapplication.git

A Simple Multi-stage Pipeline Demo

A quick introduction to Devops, CI/CD and Pipelines

Q &A

We're done!

Thank you for your time and participation.