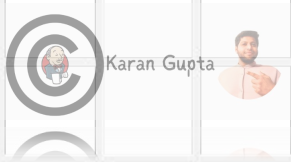
Karan Gupta

WORLD OF

Jenkins

About me – Karan



- I'm Karan Gupta
- I'm 5x AWS certified, 2x Azure, CKA certified
- Have worked with startups, Mid-Size and MNCs as a Devops Engineer.
- Published over 10+ research papers

- In free time, I love travel to new destination
- Workout enthusiast

Course Prerequisites

- **Level**
  - All (Freshers, Experienced, Professional)

- **Hardware/System Requirement**
  - Any OS will be fine
  - 256 MB of RAM, although more than 2 GB is recommended
  - 10 GB of drive space (for Jenkins and your Docker image)

- **Practice as much as possible.**
- **Study and learn at you pace.**

Contents

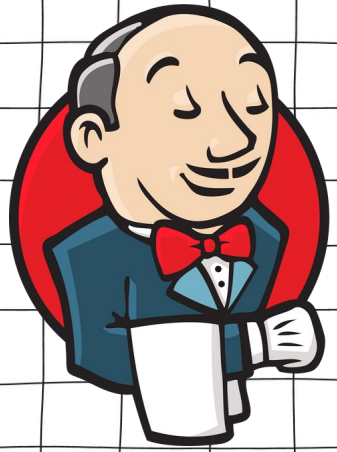
Bonus

01 Freestyle Project

02 Docker Deployment

03 Kubernetes Deployment

Real-Time



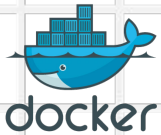
Jenkins

Industry based project

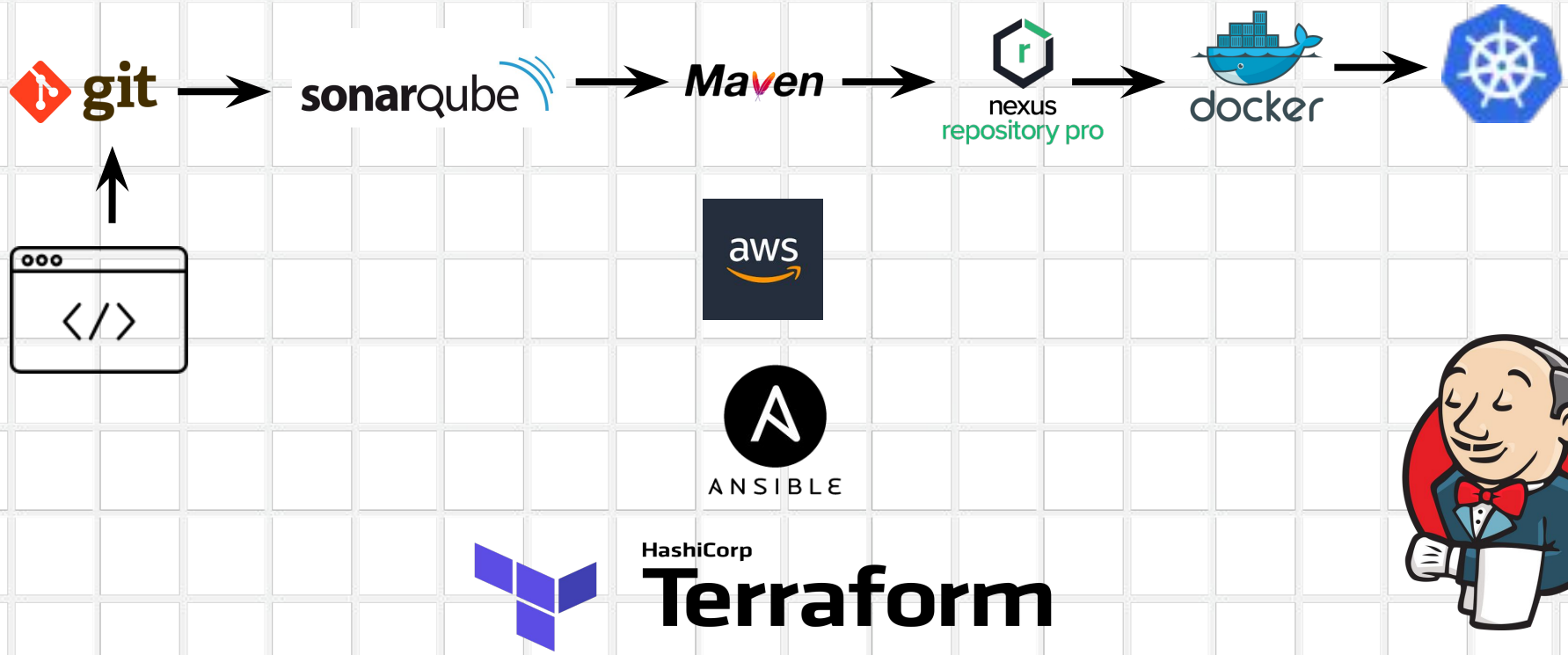
Major Project

Tools

- Git + GitHub
- AWS
- Jenkins
- Maven
- Docker
- Sonarqube
- Nexus
- Kubernetes (EKS)
- Terraform
- Ansible



Architecture

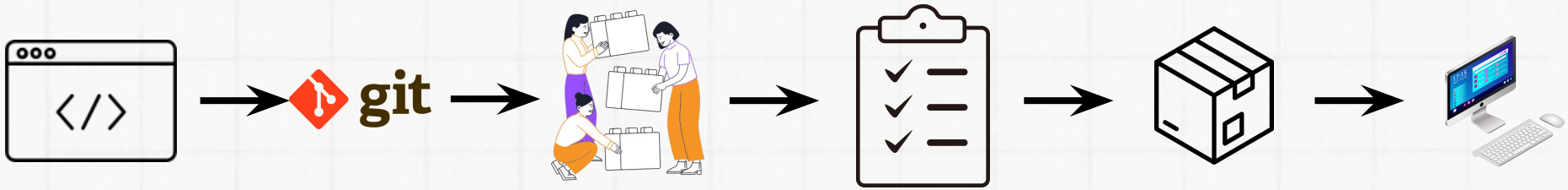


Introduction

- Jenkins is an open-source automation server that enables developers to build, test, and deploy code continuously.
- One of the most widely used automation tools for CI and CD.

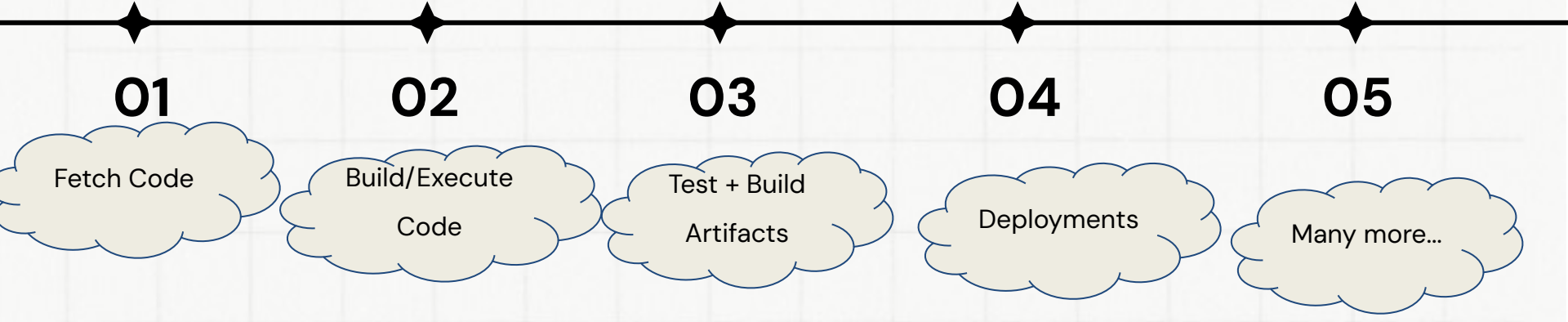
Automation

- Process of using tools and technologies to perform tasks automatically, without manual intervention



Jenkins process

- Jenkins plays a crucial role in DevOps by automating continuous integration and continuous delivery (CI/CD) processes



CI/CD

Continuous Integration

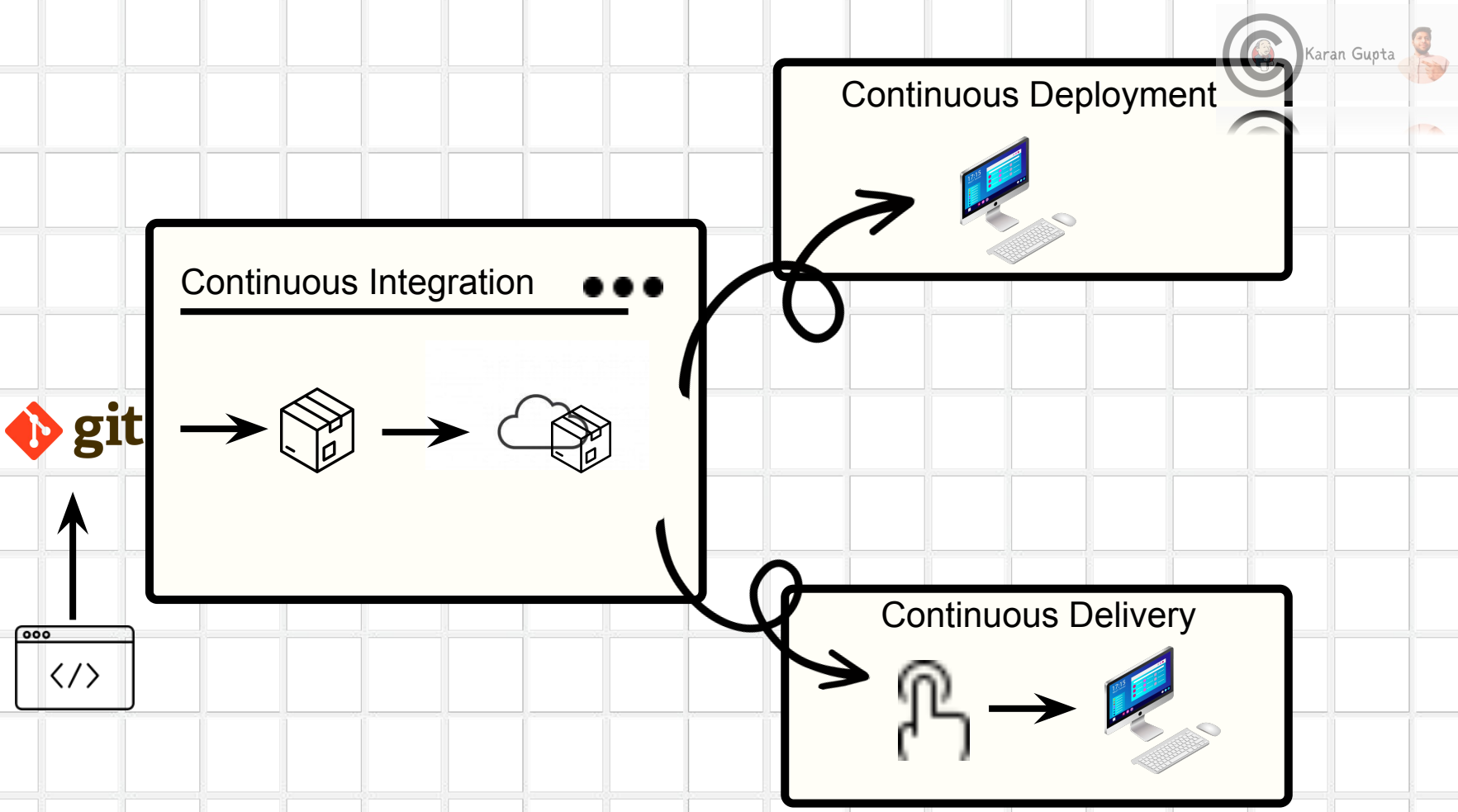
Process to get code and package it; then sent to CD for further processing

Continuous Delivery

Process in which package is deployed on system after MANUAL trigger.

Continuous Deployment

Process in which package is deployed on system AUTOMATICALLY.


Continuous Deployment
Continuous Integration
git
Continuous Delivery

Need

Challenges Jenkins
Solves / Benefits

Slower
and
Manual Builds

Automated
and
faster builds

Inefficient releases
and
Inconsistent environment

Efficient
and
Faster Releases

Non-scalable
And
Limited Visibility

Scalability
and
Extensibility

WHY JENKINS

BENEFITS

- Free / Enterprise Version
- Open-Source
- Plugins
- Easy and Portable

ALTERNATIVES

- Github Actions
- Gitlab CI/CD
- Codebuild
- TravisCI

GitHub Actions

GitLab

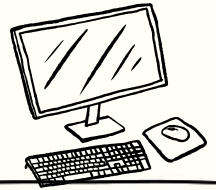
Installation

Dedicated Server

On any cloud AWS, AZURE, GCP or any other

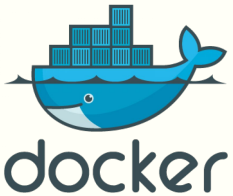
Locally

Can be installed on any OS locally.



Docker

Jenkins as Containers



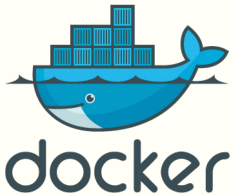
Cloud

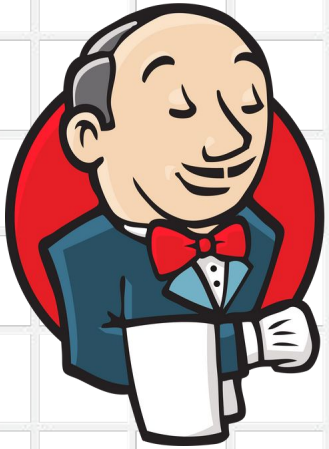
Taking AWS

Local

Taking Mac

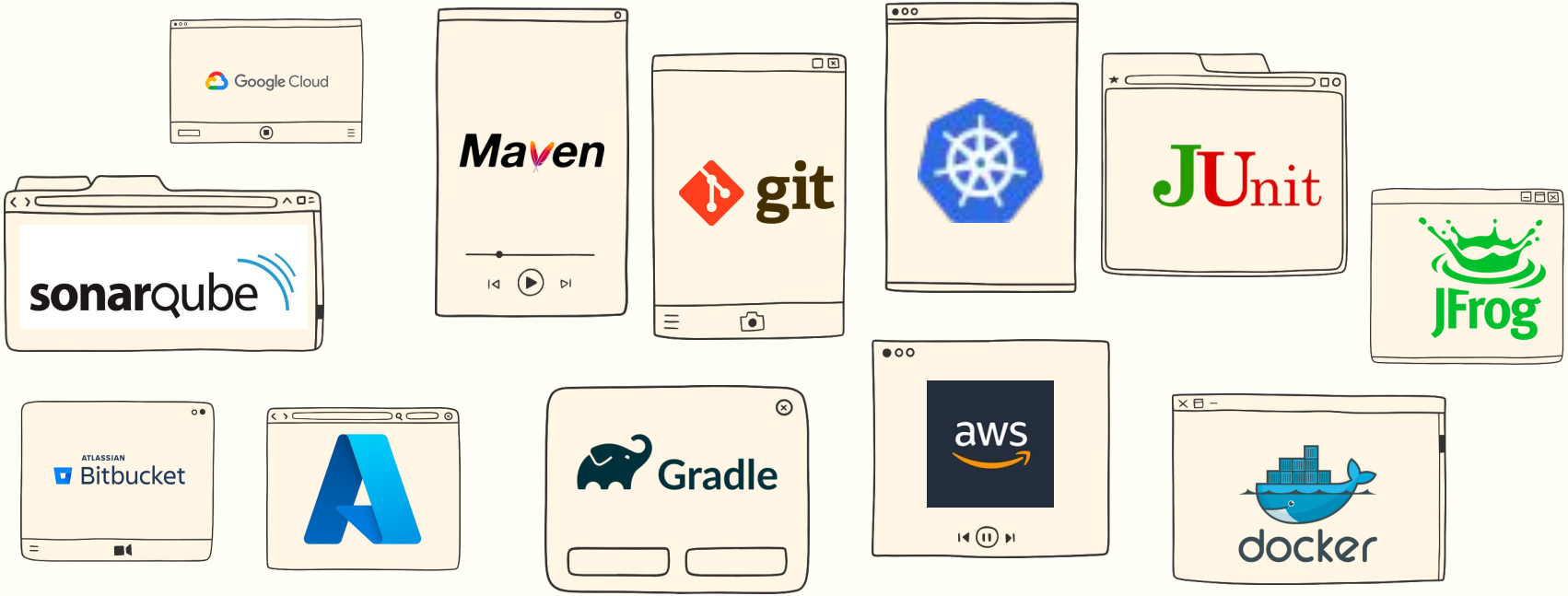
Docker

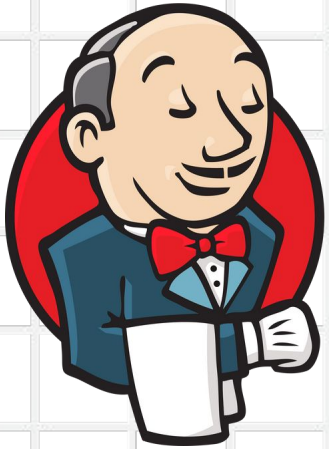




Overview of UI

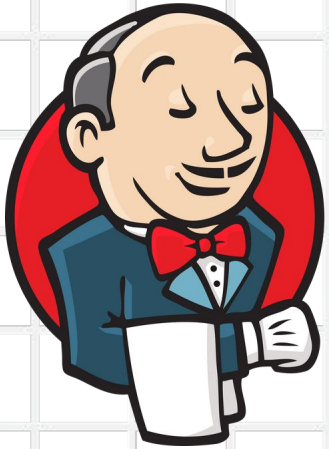
Plugins





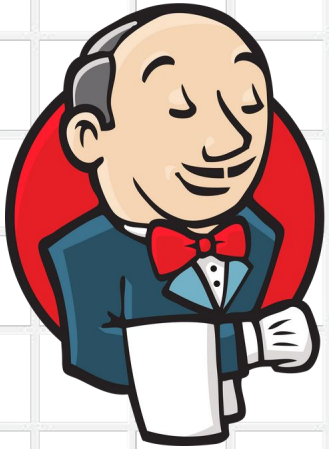
Plugins

Installation and Updation



Plugins

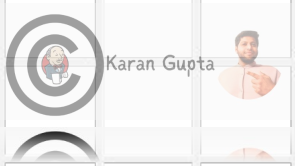
Installation Directly on Server



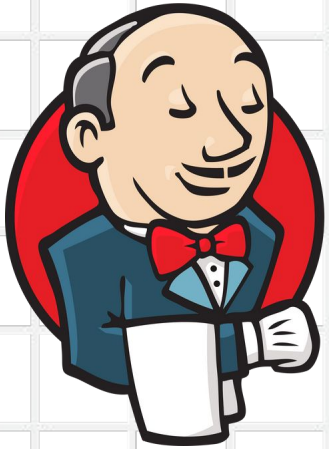
HOME_DIRECTORY

/var/lib/jenkins

Folder Structure



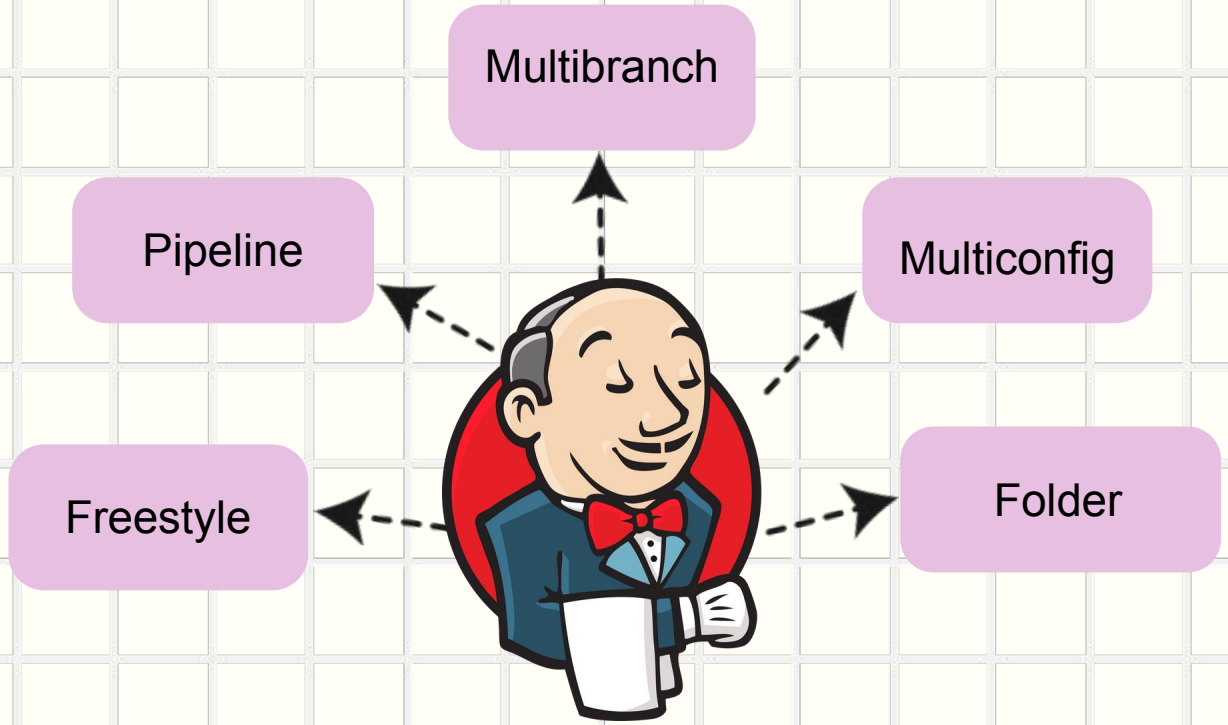
- **config.xml:** global configurations
- **credentials:** Holds sensitive information
- **plugins:** Downloaded plugin files
- **jobs:** Contains subfolders for each defined job:
  - **config.xml:** Specific configuration of the job
  - **builds:** Stores the build history
  - **workspace**
- **nodes:** Configuration files for nodes

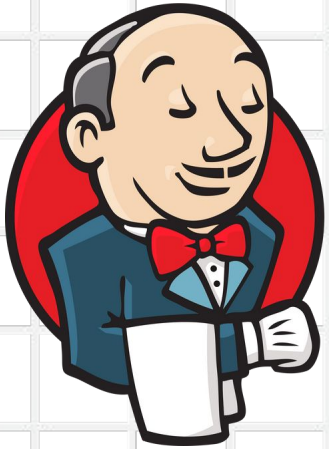


Jenkins

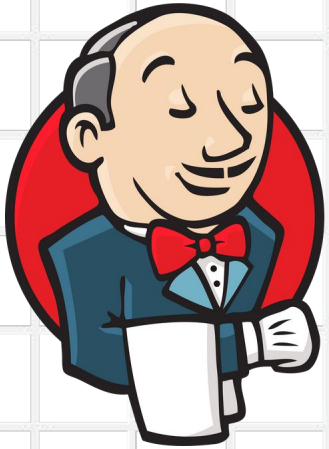
Credentials

JOBS





Freestyle Job



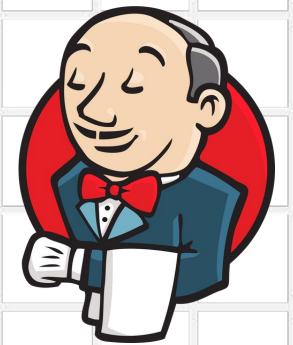
GIT Refresher

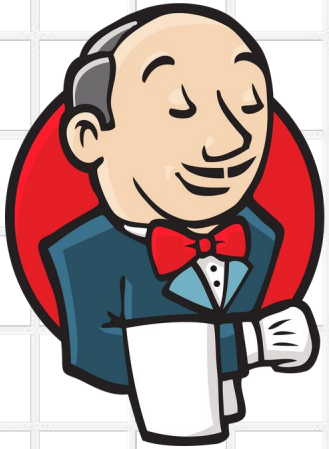
GIT

- Git is a **distributed version control system (DVCS)** used primarily for **tracking changes in code** and coordinating collaboration among developers.

Commands

- git init
- git clone
- git add
- git commit -m “message”
- git push





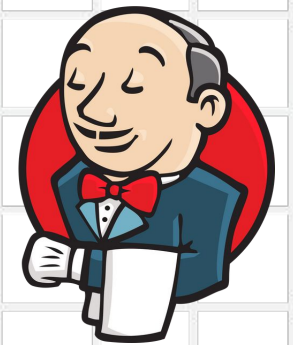
Maven Refresher

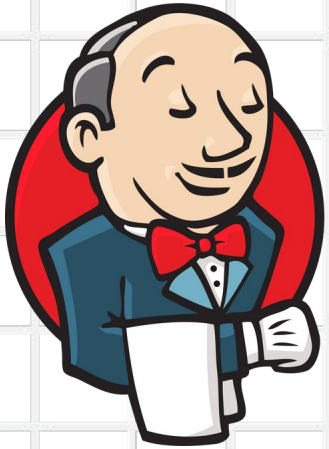
Maven Tool

- **Maven** is a free and open-source project management tool specifically designed for **Java-based projects**.

Commands

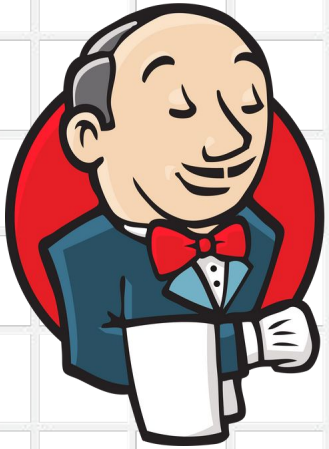
- mvn clean
- mvn compile
- mvn test
- mvn package





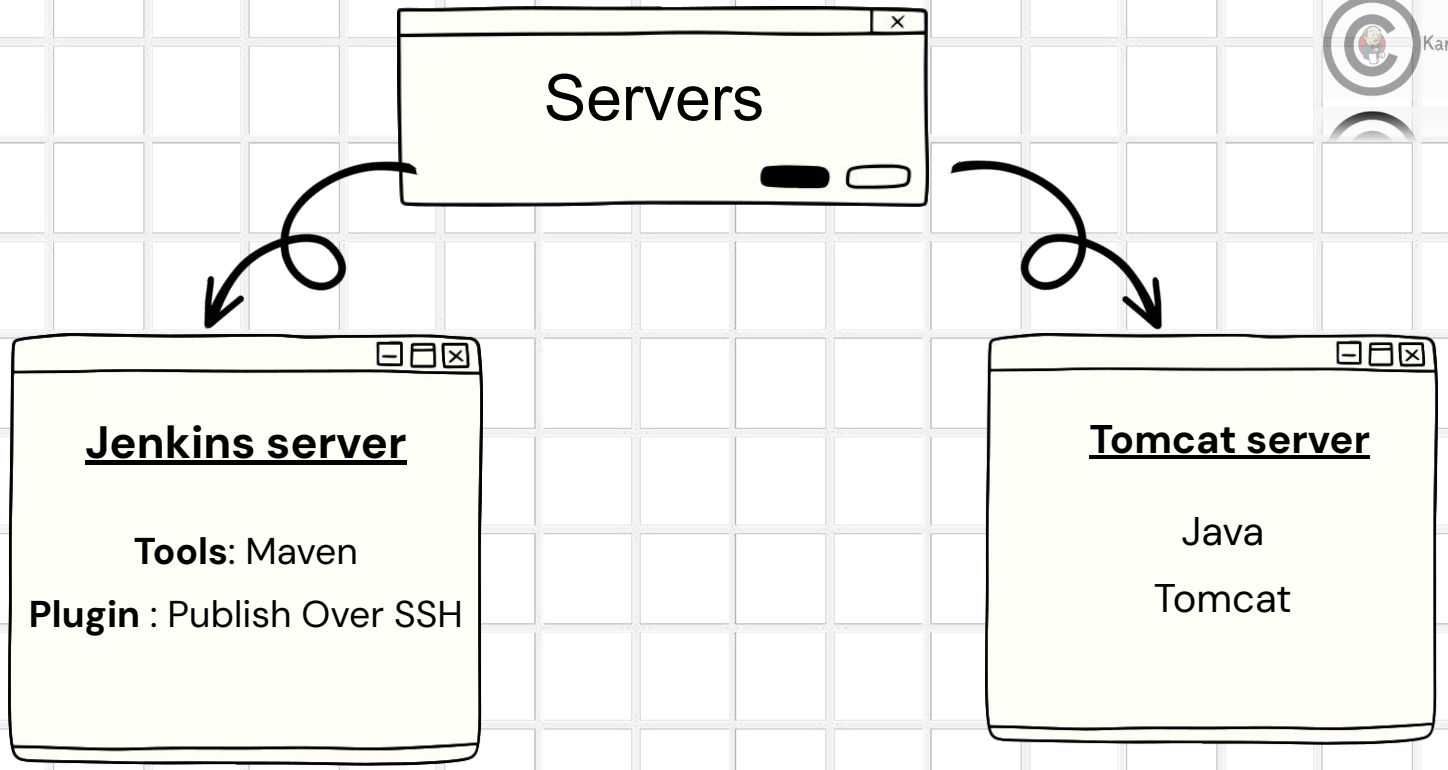
Freestyle

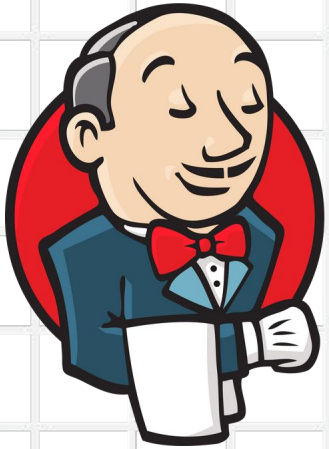
Using Maven



Freestyle

Project


Servers
Jenkins server
Tools: Maven
Plugin : Publish Over SSH
Tomcat server
Java
Tomcat



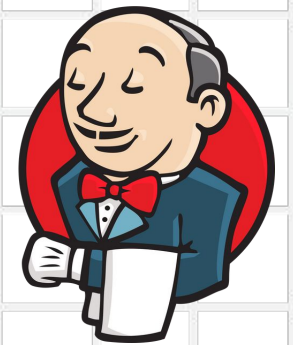
Github Webhooks

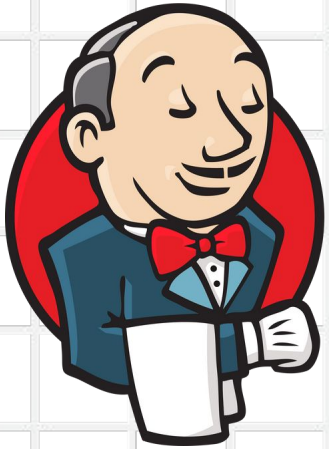
GitHub Webhook

- Event-driven notification system from GitHub that automatically triggers actions in your Jenkins server when specific events occur in your repository.

Benefits

- Reduced Latency
- Automated Triggering
- Traceability and Visibility
- Secured





Pipeline

Job

Jenkinsfile


```
node {
    stage('Checkout') {
        echo 'this is checkout stage'
    }
    stage('Build') {
        echo 'this is build stage'
    }
}
```

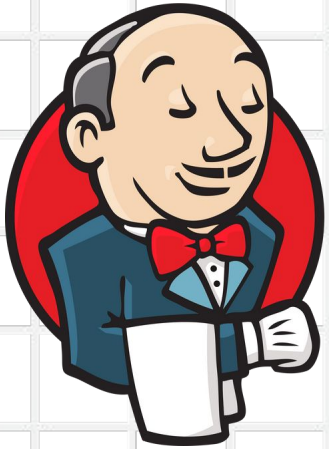
Imperative

```
pipeline{
    agent any
    stages{
        stage('build stage'){
            steps{
                echo "this is build stage"
            }
        }
        stage('deploy stage'){
            steps{
                echo "this is deploy stage"
            }
        }
    }
}
```

Declarative

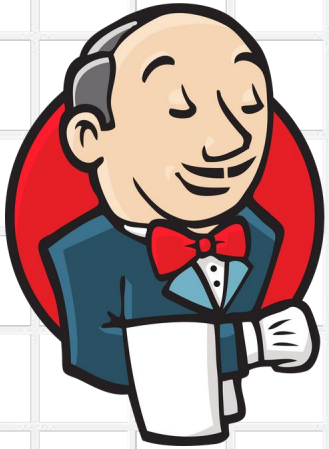
Declarative Jenkinsfile


```
pipeline{
    agent any
    stages{
        stage('build stage'){
            steps{
                echo "this is build stage"
            }
        }
        stage('deploy stage'){
            steps{
                echo "this is deploy stage"
            }
        }
    }
}
```



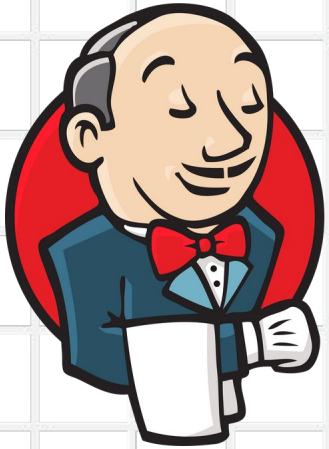
Jenkinsfile

1 - Basic



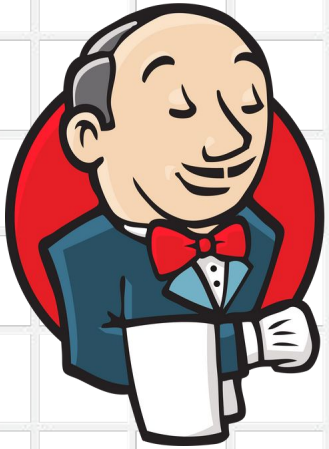
Jenkinsfile

2 - Using Options



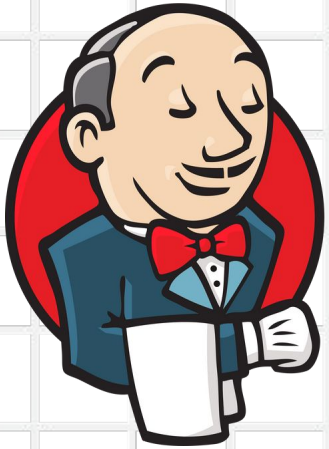
Jenkinsfile

3- Checkout Stage



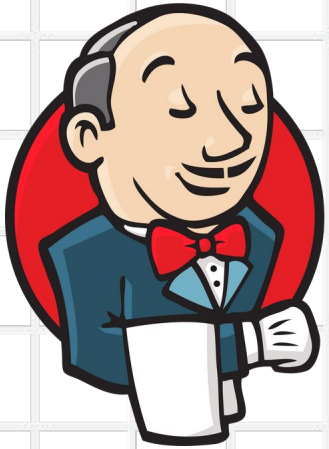
Jenkinsfile

4 - Using Tools



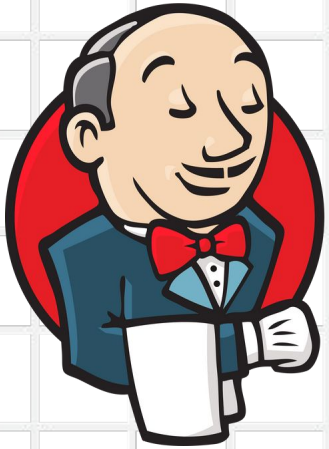
Jenkinsfile

5 - Env. Variables



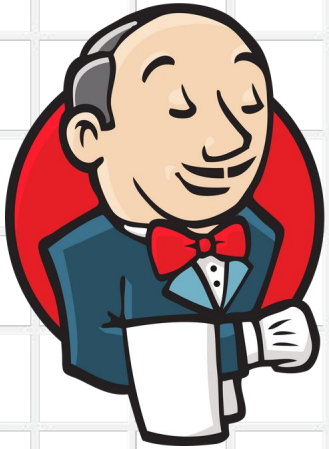
Jenkinsfile

6 - Parameterized



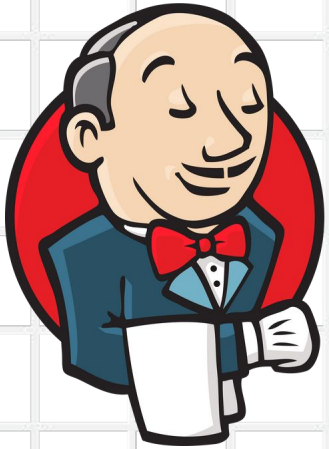
Jenkinsfile

7 - Upstream & Downstream Jobs



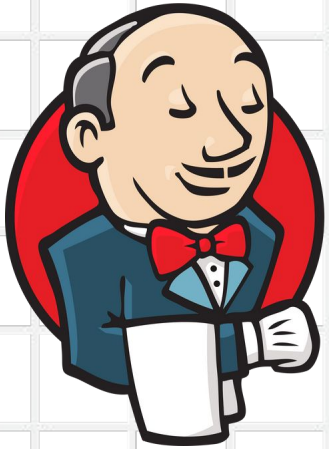
Jenkinsfile

8 - Post Actions



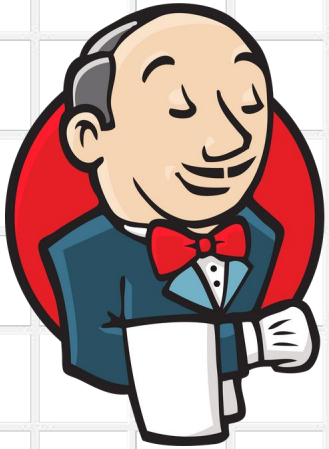
Jenkinsfile

9 - Built-In Variables



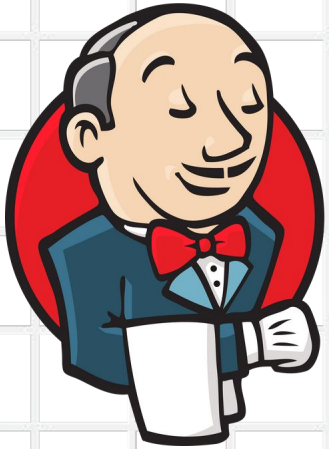
Jenkinsfile

10 - Using Script



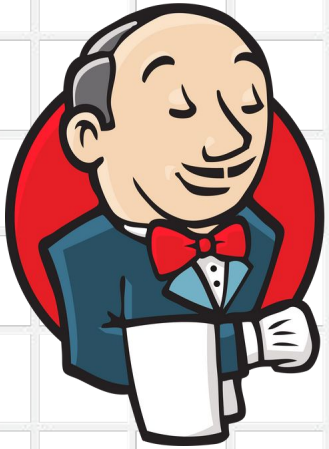
Jenkinsfile

11 - Script from file



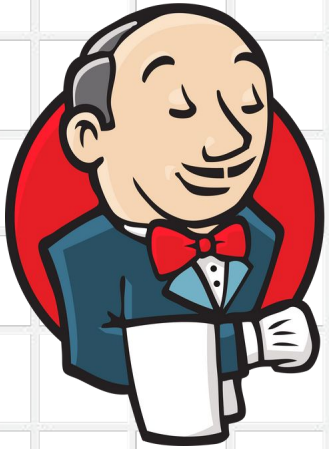
Jenkinsfile

12 - Applying Loops



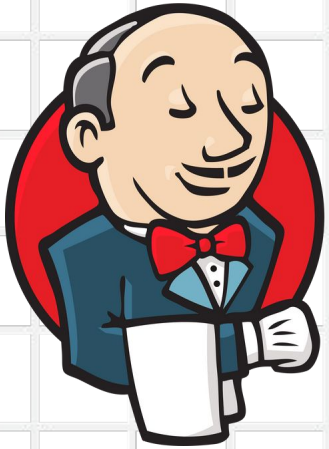
Jenkinsfile

13 - Loops + Script



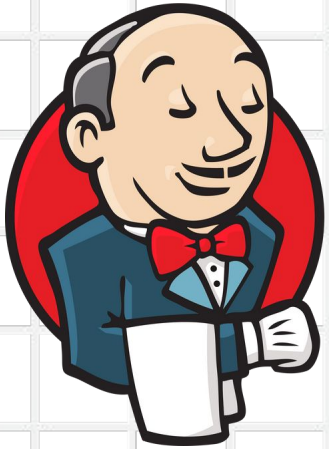
Jenkinsfile

14 - Conditions



Jenkinsfile

15- Functions



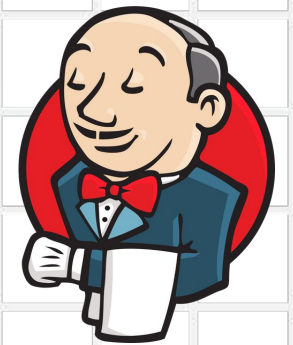
Docker Refresher

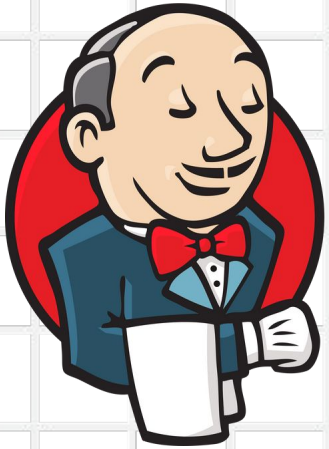
Docker

- Docker is an open-source platform for **developing, shipping, and running applications** in a standardized way using **containers**.

Commands

- docker ps
- docker run [image] [command]
- docker stop [container id/name]
- docker rm [container id /name]
- docker images
- docker rmi [image id/name]





Docker

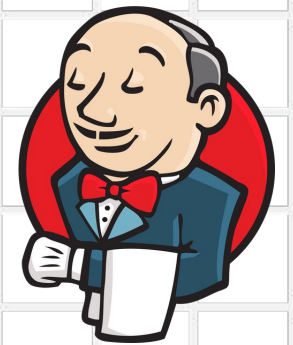
Server Setup

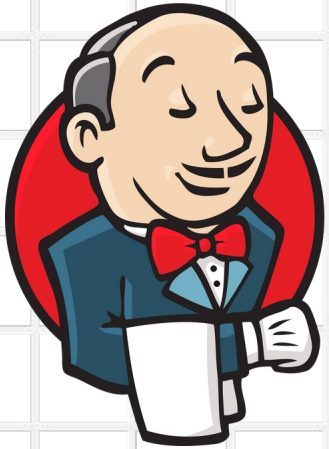
Docker Hub

- **Docker Hub** is a **cloud-based registry service** owned by Docker, Inc. Docker Hub serves as a **central repository** for storing and sharing **Docker images**.

Commands

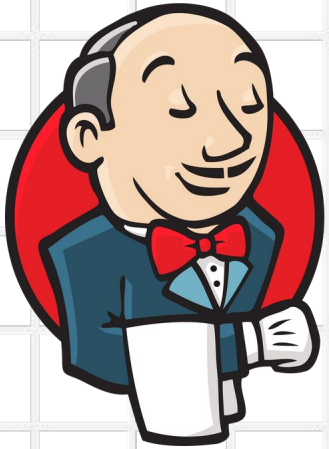
- docker login
- docker pull [image]
- docker push [image]
- docker search [term]





Jenkinsfile

16 - Docker



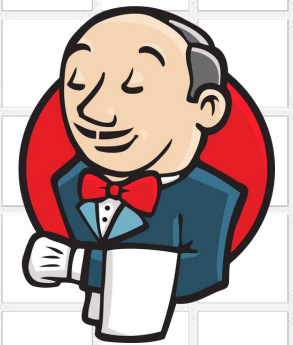
Jenkinsfile

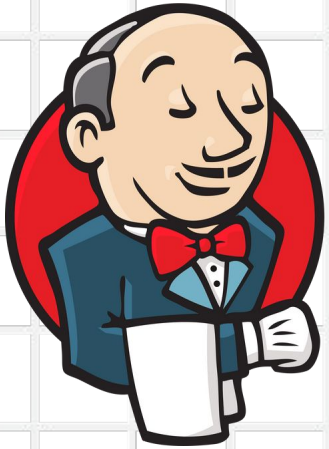
17 - Docker + Dockerhub

PROJECT

Docker Deployment

Project
Docker Deployment



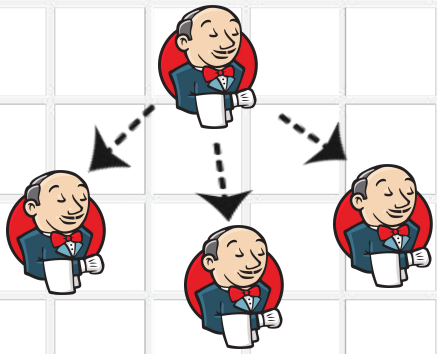


Agents

Agents Benefits

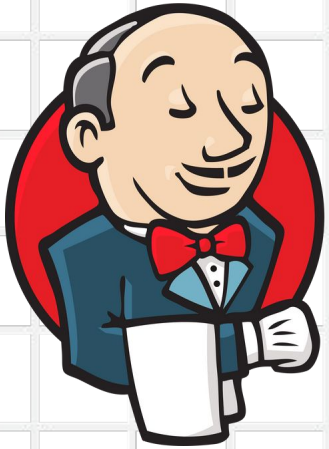
Scalability

Flexibility

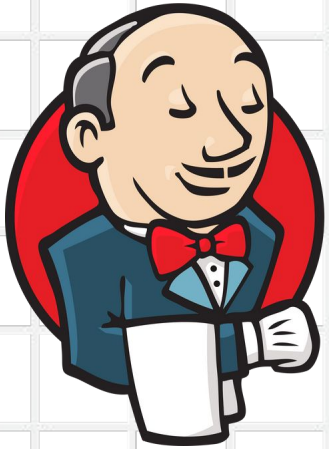


Maintenance

Security

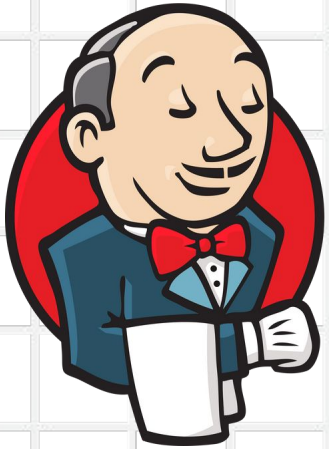


Adding Agent



Jenkinsfile

18 - Agent Usage



Post Action

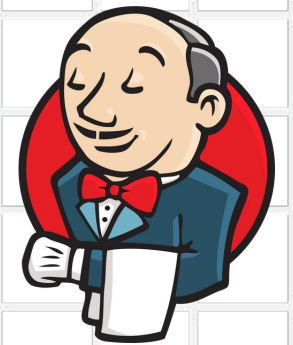
Email Notification

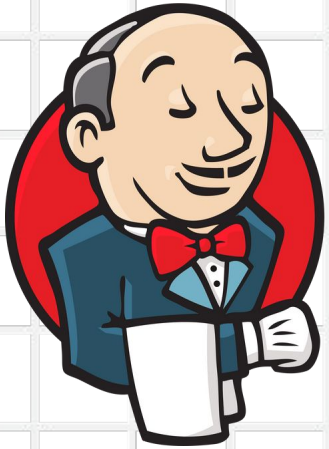
Email Notification

- E-mail notification allows you to receive automatic emails about various events and activities within your builds and pipelines.

Gmail Sample

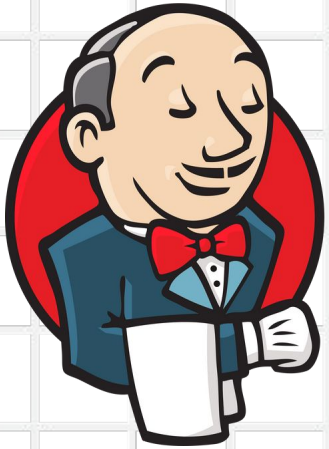
- **Smtp server** - smtp.gmail.com
- **Suffix** - @gmail.com
- **Port**: 465
- Username and App Passwords
- SSL selected
- Recipient address





Jenkinsfile

19 - Email



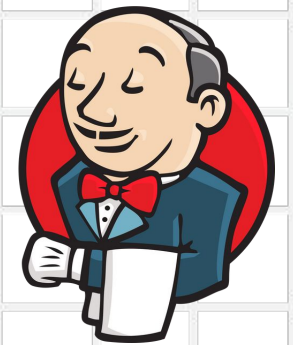
Multi-Branch Job

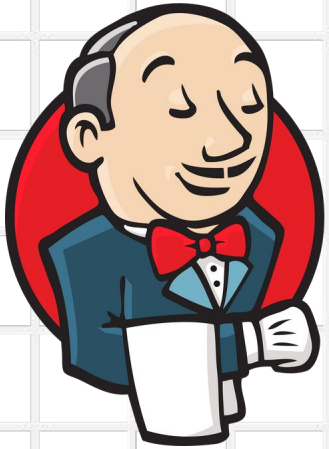
Multi Branch Job

- A multi-branch job in Jenkins is a powerful feature that allows you to manage and automate builds for multiple branches or repositories from a single configuration.

Benefits

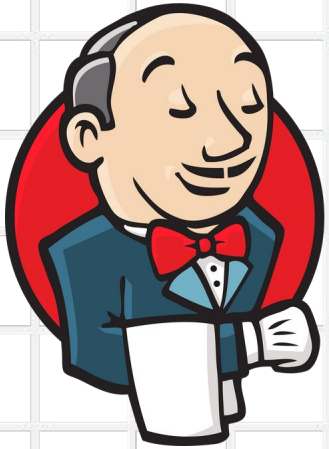
- Reduce duplication
- Centralized management
- Branch-specific configurations
- Individual build histories





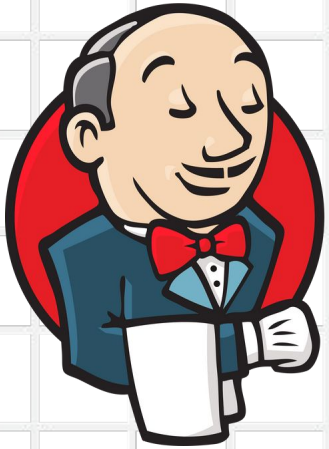
Jenkinsfile

20 - Multibranch



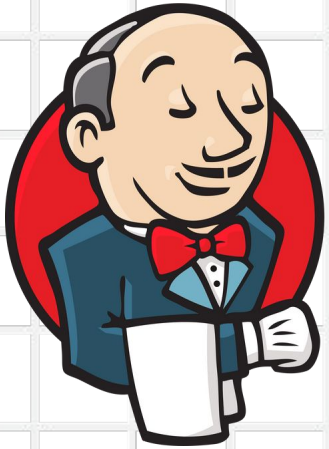
Jenkinsfile

21 - AWS



Jenkinsfile

21 (a) - AWS



Jenkins

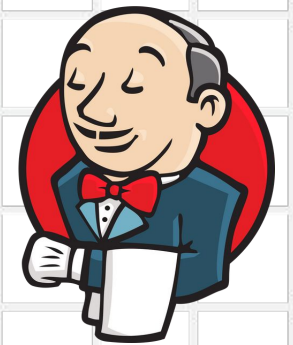
AWS

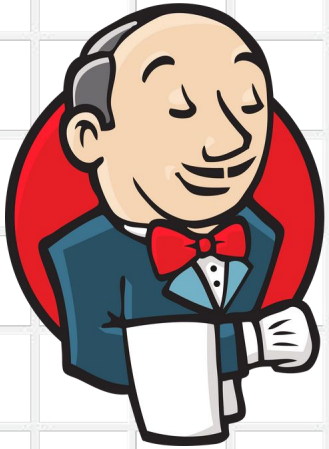
AWS with Jenkins

- Integrating **Jenkins** with **AWS** offers several advantages for building and deploying applications on the AWS Cloud.

Few Industry tasks

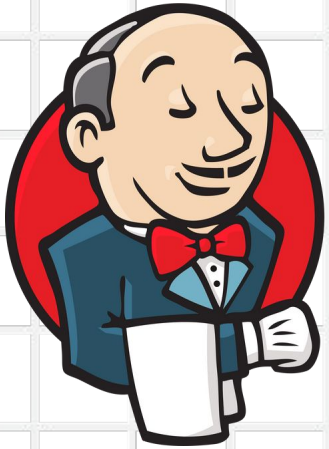
- Artifact management
- Creating custom AMI
- Taking snapshots
- Codepipeline
- Creating Infrastructure





Jenkinsfile

21 (b) - AWS



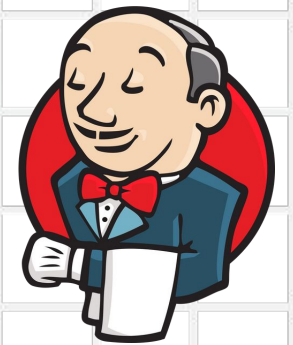
Kubernetes Refresher

Kubernetes

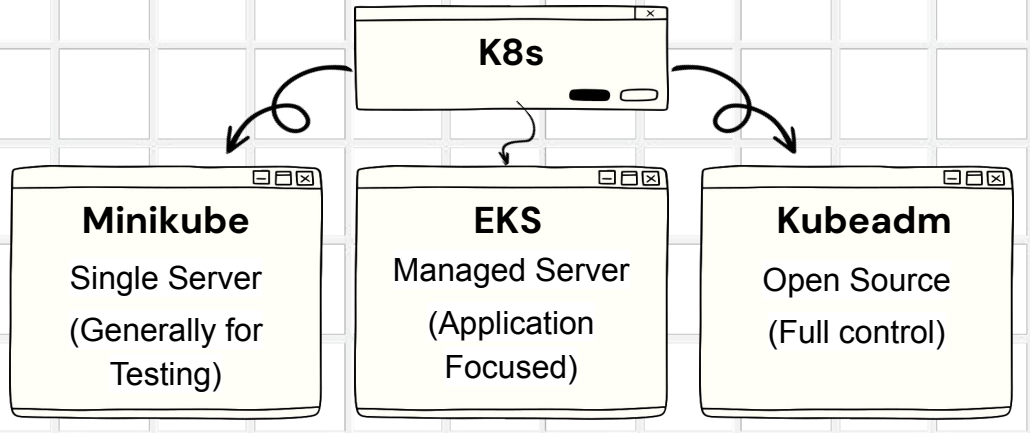
- Kubernetes (often shortened to K8s) is an open-source platform designed for automating the deployment, scaling, and management of containerized applications.

Benefits

- Automated Container Orchestration
- Self-healing Capabilities
- High Availability
- Faster Time to Market
- Portability

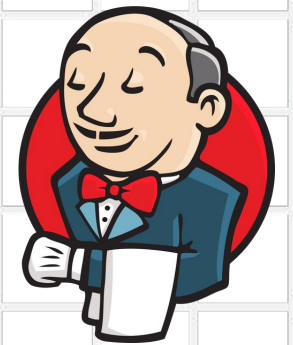


Different Flavors

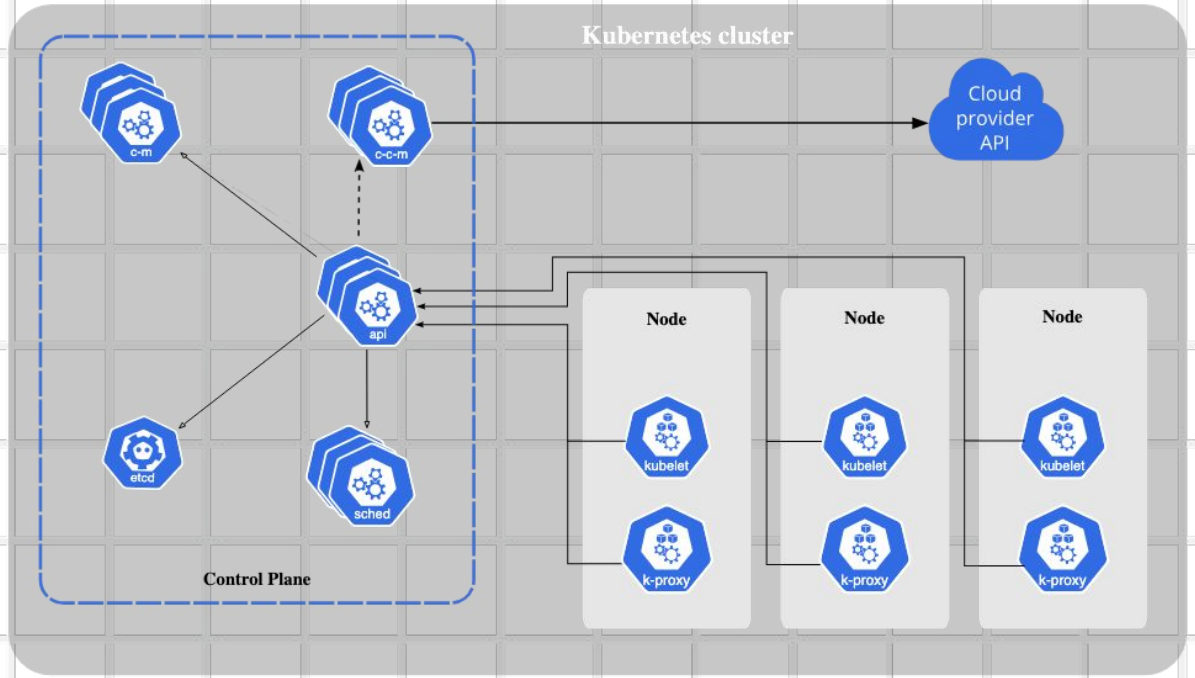


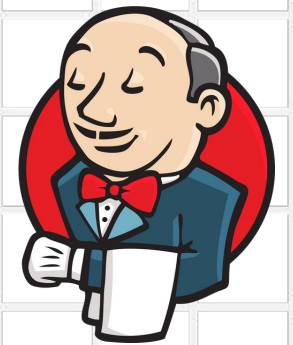
Commands

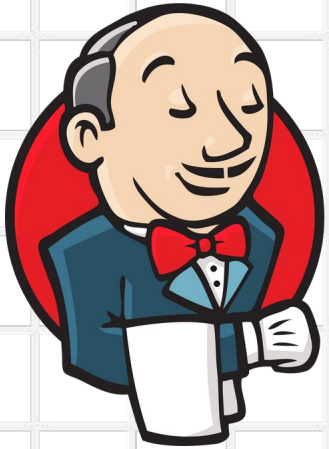
- kubectl apply -f <filename>
- kubectl get <resource> <resource name>
- kubectl describe <resource> <resource name>
- Kubectl delete <resource> <resource name>



Architecture

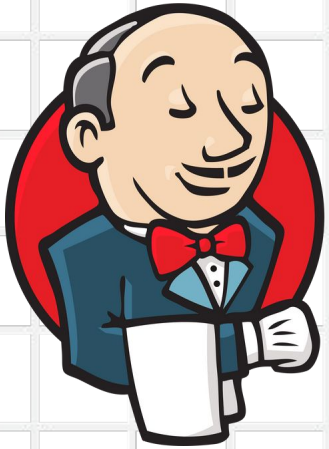






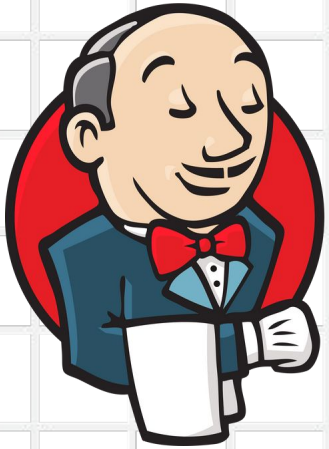
Docker

Server Setup



Jenkinsfile

22 - Kubernetes



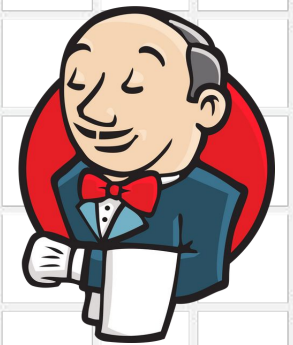
Shared Library

Shared Library

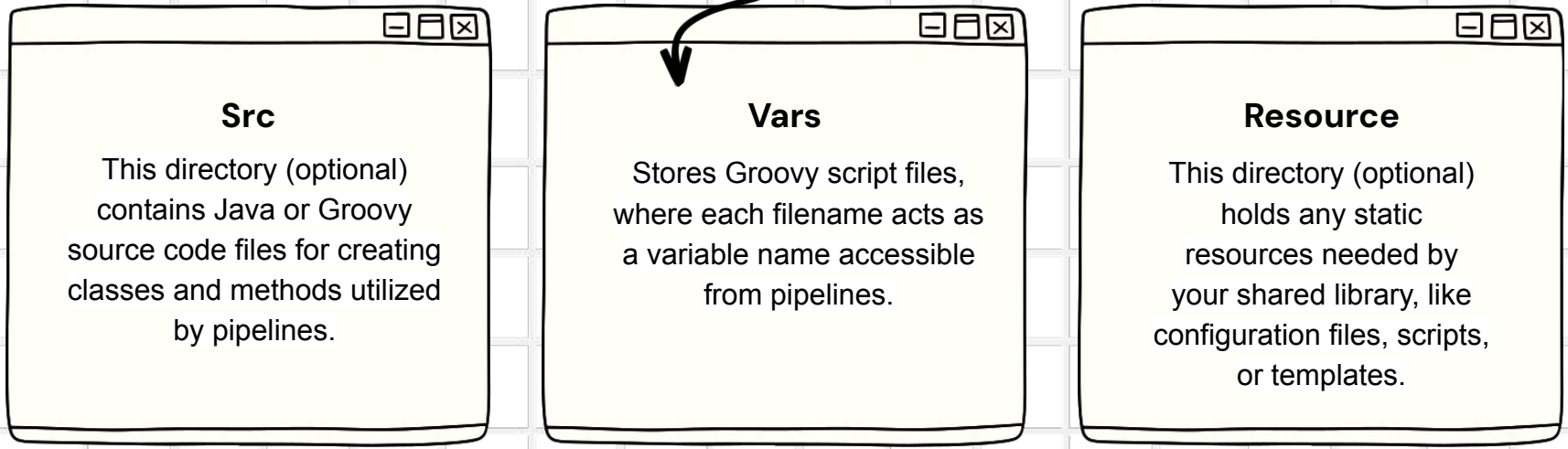
- A shared library in Jenkins is a reusable collection of Groovy scripts that can be used by multiple Jenkins jobs and pipelines. It allows you to modularize your code, improve consistency, and simplify pipeline development.

Benefits

- Reduce duplication
- Code Reuse
- Consistency and Maintainability
- Collaborations



Folder Structure



Src

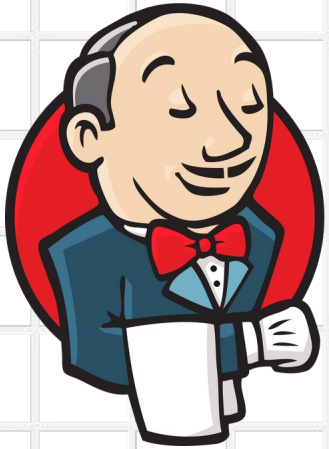
This directory (optional) contains Java or Groovy source code files for creating classes and methods utilized by pipelines.

Vars

Stores Groovy script files, where each filename acts as a variable name accessible from pipelines.

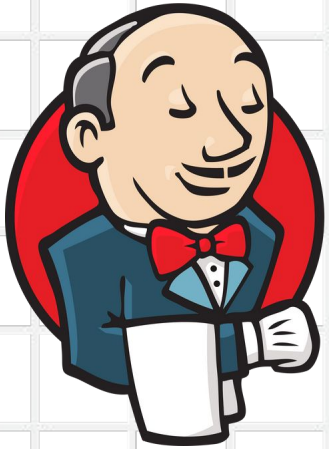
Resource

This directory (optional) holds any static resources needed by your shared library, like configuration files, scripts, or templates.



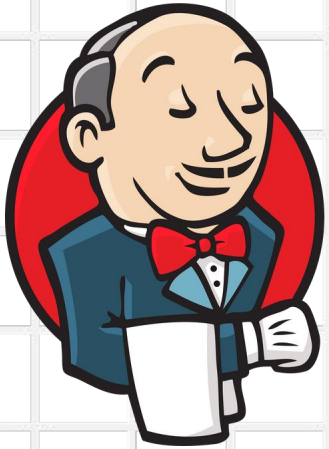
Jenkinsfile

23 - Shared Library



Jenkinsfile

23 - Shared Library (a)



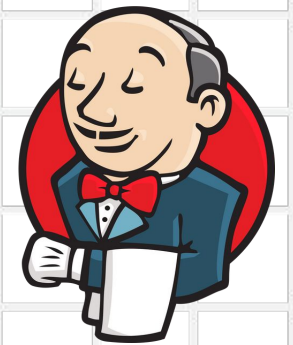
SonarQube

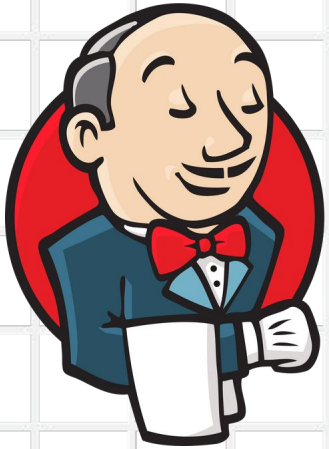
SonarQube

- SonarQube is an open-source platform that provides continuous code quality and security analysis for various programming languages.

Benefits

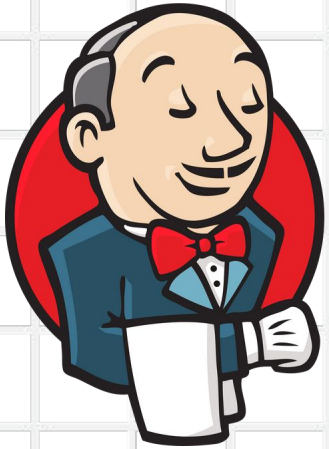
- Improved Code Quality
- Enhanced Security
- Early Defect Detection
- Continuous Monitoring





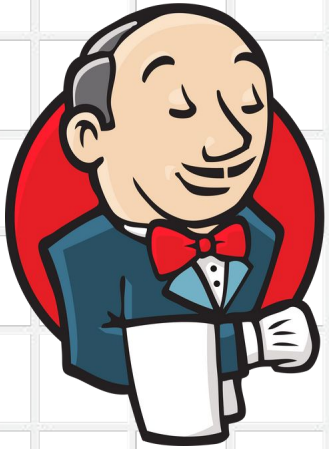
SonarQube

Server Setup



Jenkinsfile

24 - Sonarqube



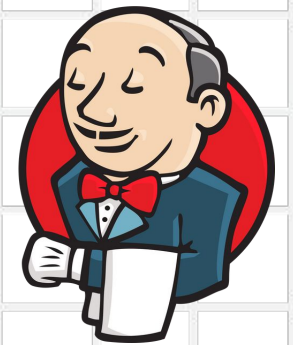
Nexus Repository

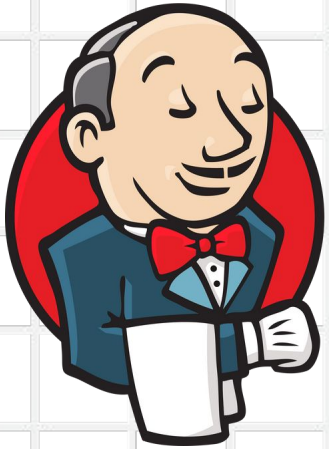
Nexus Repository

- Nexus acts as a central hub for storing, managing, and distributing various software artifacts like build outputs, libraries, packages, and container images.

Benefits

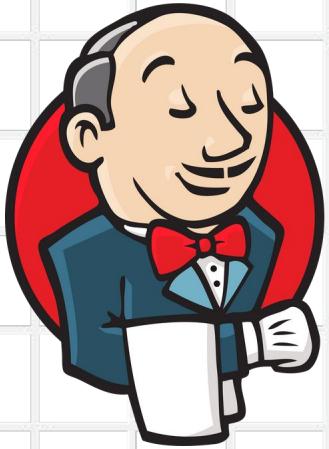
- Artifact Management
- Improved Build Efficiency
- Security and Access Control
- Collaborations





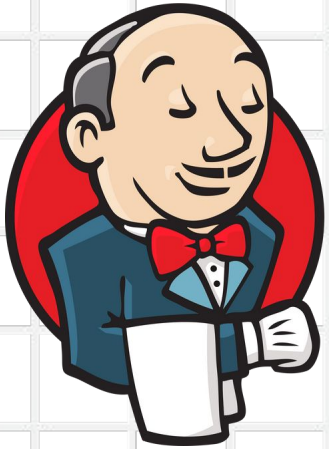
Nexus

Server SetUp



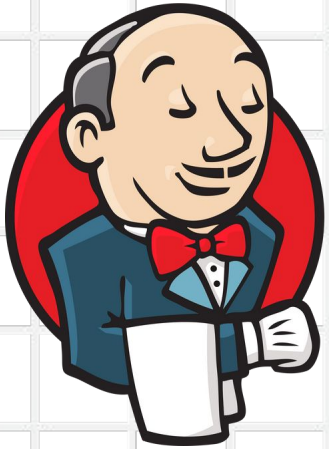
Jenkins

Port Change



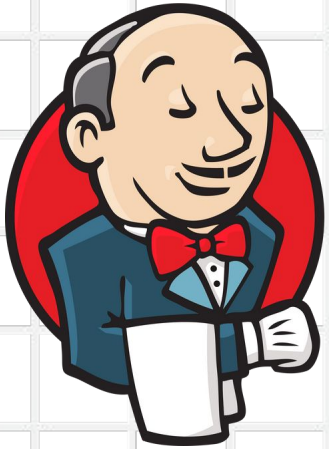
Nexus

Freestyle



Jenkinsfile

25 - Nexus Repo



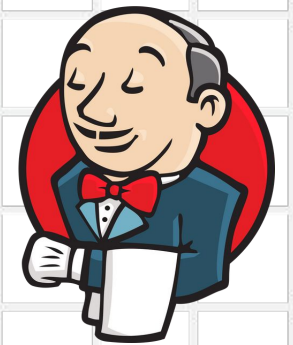
Terraform Refresher

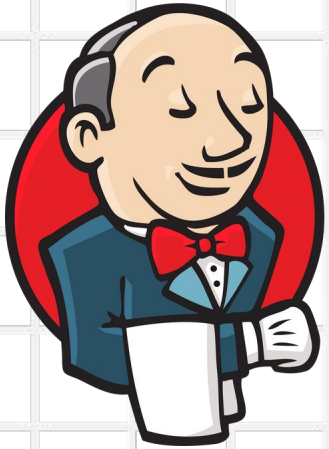
Terraform

- **Terraform** is an open-source tool developed by HashiCorp that enables you to manage infrastructure as code (IaC). It allows you to define and provision infrastructure resources like servers, networks, databases, and other components **using human-readable configuration files**.

Commands

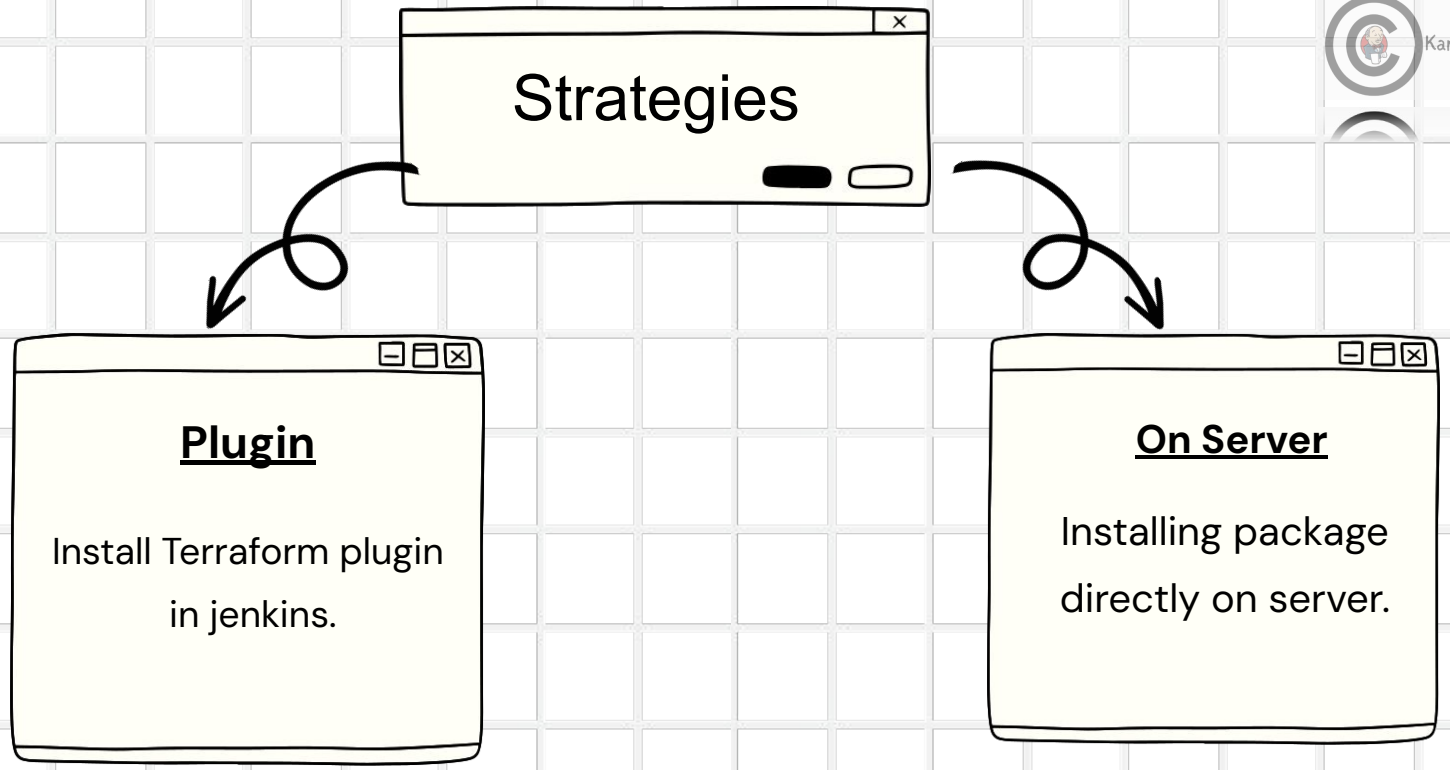
- terraform init
- terraform plan
- terraform apply -auto-approve
- terraform destroy -auto-approve

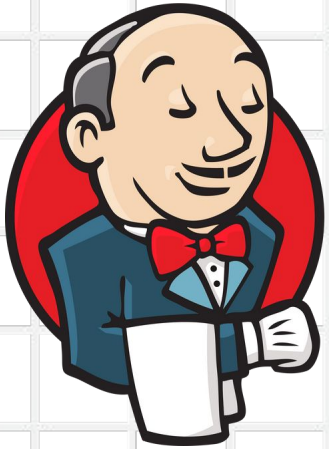




Terraform

SetUp


Strategies
Plugin
Install Terraform plugin in jenkins.
On Server
Installing package directly on server.



Jenkinsfile

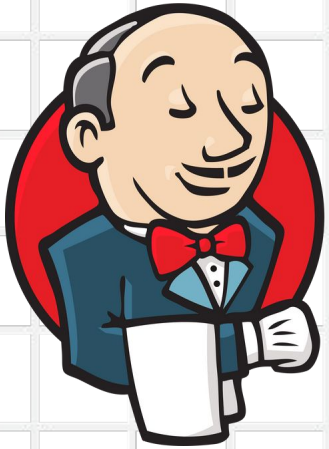
26 - Terraform

Jenkinsfile

26 - Terraform (a)

Jenkinsfile

26 - Terraform (b)



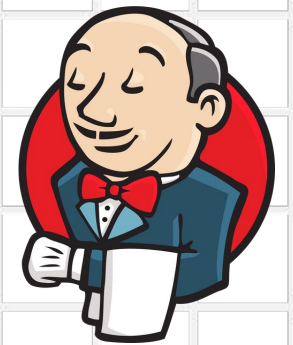
Ansible Refresher

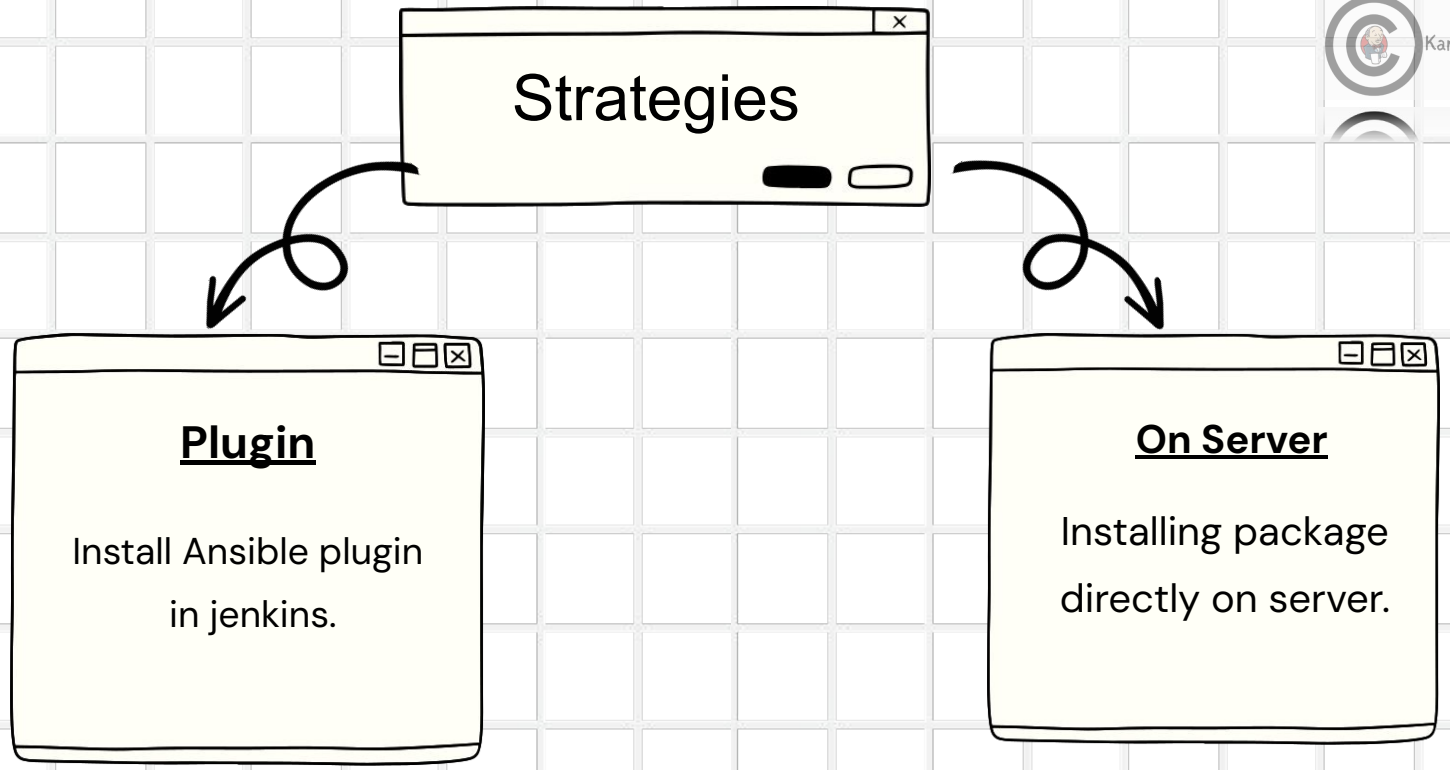
Ansible

- Ansible is an open-source automation tool that automates IT tasks such as configuration management, application deployment, cloud provisioning, and orchestration.

Benefits

- Automated Deployment
- Integration
- Consistency
- Scalability



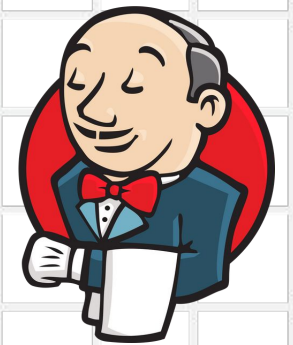

Strategies
Plugin
Install Ansible plugin in jenkins.
On Server
Installing package directly on server.

Ad-Hoc Commands

- **Ping**
  - ansible all -i <inventory_file> -m ping
- **Disk Space Check**
  - ansible all -i <inventory_file> -m shell -a "df -h"

Playbook Working

- Write Playbook
- Create Inventory File
- ansible-playbook -i <inventory_file> <playbook_file.yml>
- Review Results



Jenkinsfile

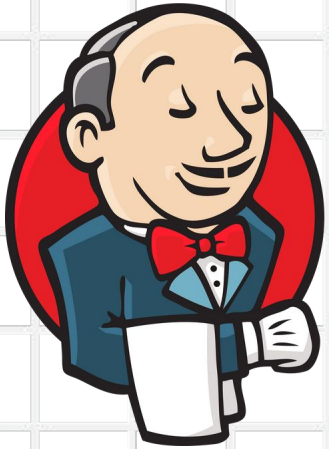
27 - Ansible

Jenkinsfile

27 - Ansible (a)

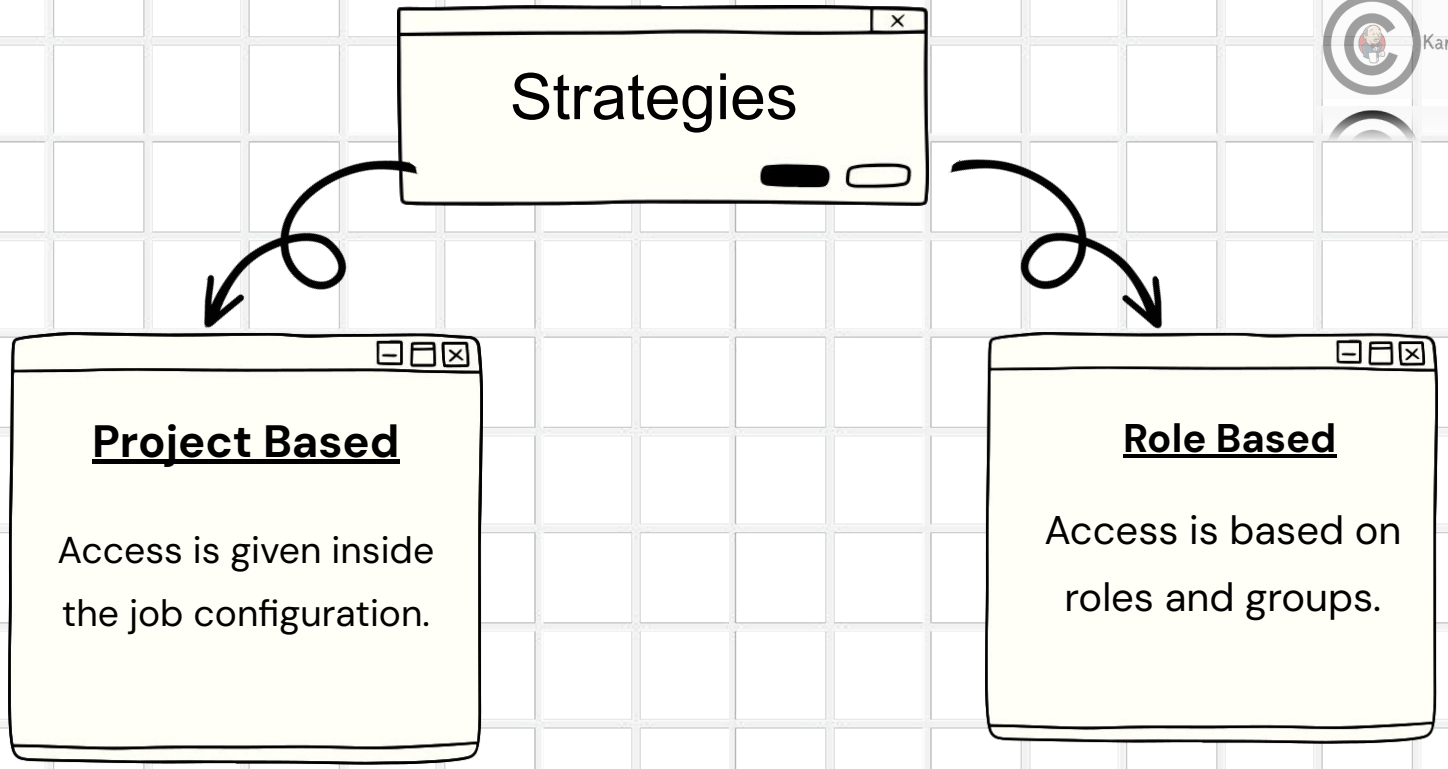
Jenkinsfile

27 - Ansible (b)



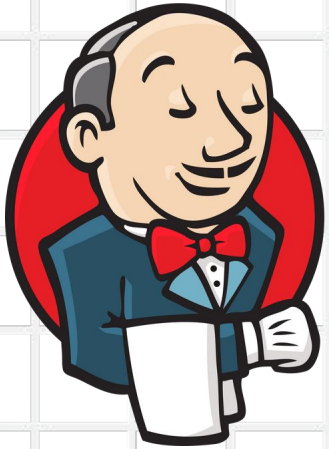
Add Users

Permissions



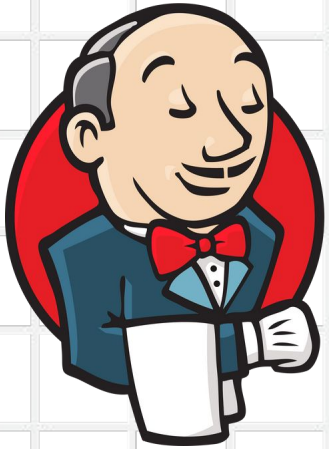
Logged-In can do Anything

Anyone can do Anything



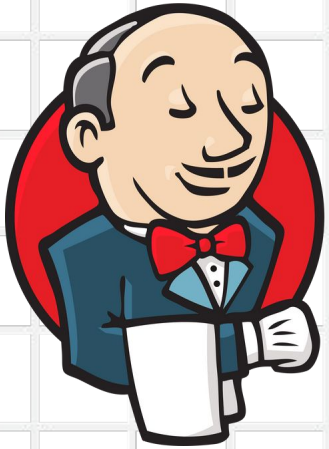
Strategy -1

Project Based

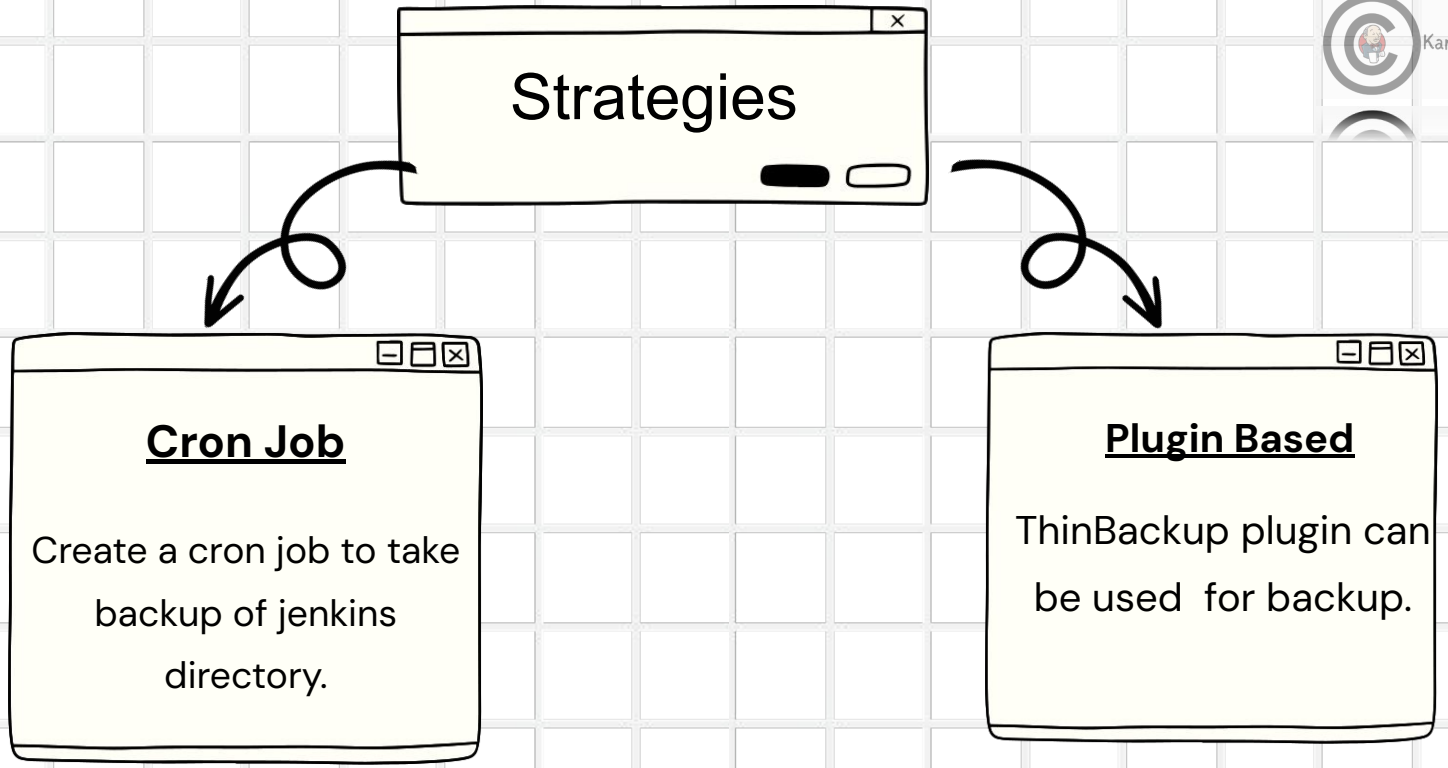


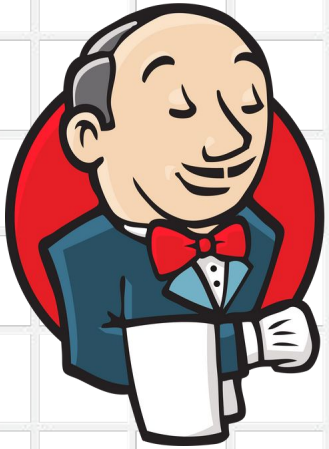
Strategy -2

Role Based



Jenkins Backup


Strategies
Cron Job
Create a cron job to take backup of jenkins directory.
Plugin Based
ThinBackup plugin can be used for backup.



Jenkins Upgradation

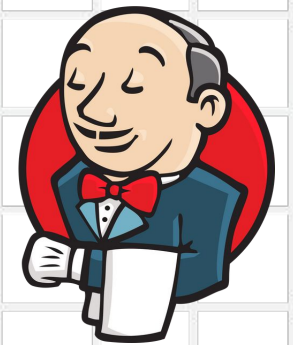
Jenkins Update

- A Jenkins upgrade refers to the process of updating your Jenkins installation to a newer version.

  The preferred way is the step case manner for the updation.

Process

- Backup
- Review Release Notes
- Check Plugin Compatibility:
- Prepare for Downtime
- Rollback Plan (For emergency)
- Test Environment and Monitor



Jenkins

Script Console

Jenkins

Troubleshooting Password Issue

Jenkins

Timestamps

Jenkins

CLI

Jenkins

Disable, Enable and Delete Job

Jenkins

Restart from a Stage

Jenkins

Logs

Jenkins

Blue Ocean

Jenkins

Folder

Jenkins

Timeout

Jenkins

Taking Input during Run-Time

Jenkins

Schedule Jobs

Scheduling a Job

In Jenkins, job scheduling refers to the functionality that allows you to automate the execution of build, test, and deployment workflows for your applications at specific times

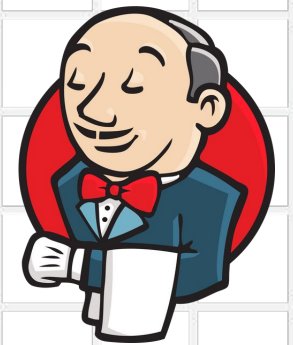
Cron Table

*	*	*	*	*
Minute(0-59)	Hour(0-23)	Date(1-31)	Month(1-12)	Day(0-6)

PROJECT

K8s Deployment

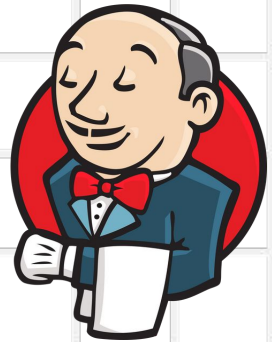
Project
Kubernetes Deployment



Industry PROJECT

Project Process:

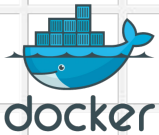
- **Step 1: Customer Business Requirement**
  - The first step is acquiring requirement from business
- **Step 2: Short listing of the services to be utilised**
  - Listing all possible options with alternatives
  - Shortlisting them with the business alignment
- **Step 3: Creating a pseudo model or process**
  - Try to build the process of the working with the team
- **Step 4: Build the application**
  - Firstly try to build the application in dev
- **Step 5: Deploy it to the production**
  - After testing and approval finally deploy to production env.



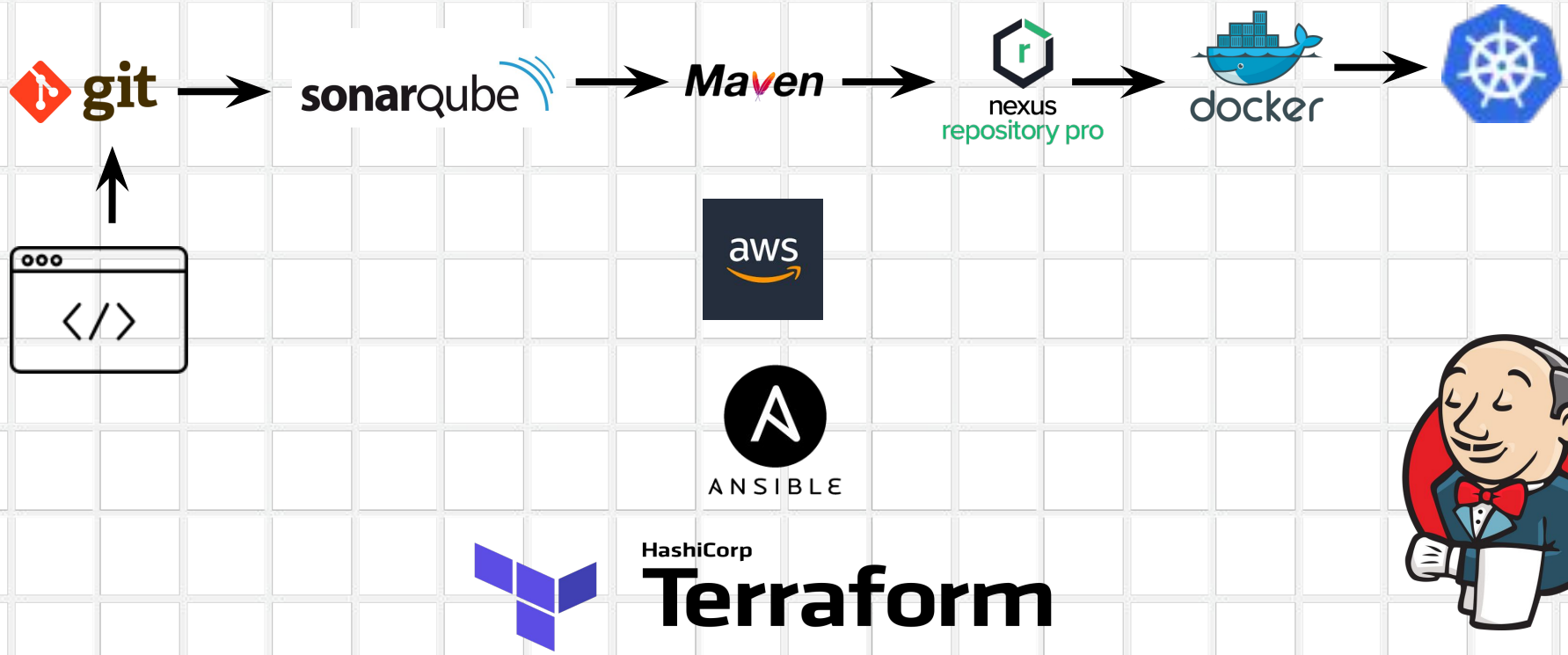
Major Project

Tools

- Git + GitHub
- AWS
- Jenkins
- Maven
- Docker
- Sonarqube
- Nexus
- Kubernetes (EKS)
- Terraform
- Ansible

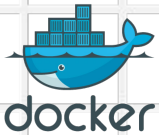


Architecture



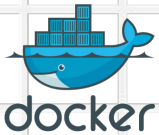
Tools Usage

- **Linux**
  - For script and commands
- **Git + GitHub**
  - As a SCM
- **AWS**
  - Cloud service provider
- **Sonarqube**
  - For the code analysis
- **Maven**
  - For building project
- **Nexus**
  - For storing artifacts



Tools Usage

- **Docker**
  - For creating images
- **Docker Hub**
  - For storing Images
- **Kubernetes (EKS)**
  - For Deployment purpose
- **Terraform**
  - For creating servers infra
- **Ansible**
  - For installing softwares



Phase-1

Infrastructure setup.

In this phase we try to lay foundation of the project.

Phase-2

Dependency setup.

In this phase, we add dependencies of the project like credentials, tools.

Phase-3

Deployment phase.

Final build of project with all the jenkins stages.

Phase-1

Infrastructure

Phase-1 - Infrastructure Setup

1. Setting Jenkins Server

A. Create script to install jenkins on a server.

B. Either using plugins or installing package directly on server install terraform, jenkins, docker K8s and git on same server.

C. After setting up the jenkins server, add maven tool.

D. Configure AWS user too

Phase-1

2. Git Setup - Public/Private

A. Create a repository for placing all codes in the repository.

B. For public repo:

   a. No credentials required

C. For private repo

   a. Credentials required

Phase-1

3. Kubernetes Setup - K8s

A. Using EKS or Kubeadm setup the K8s cluster.

   a. Setup worker nodes

   b. Have details of users/roles

   c. Information about kubeconfig file

Phase-1

4. Terraform Scripts

A. Create terraform modules for having 3 servers for the working using jenkins job.

   a. Sonar Server

   b. Nexus Server

   c. Test Server

B. Place the terraform script in the github for the working.

Phase-1

5. Nexus and Sonar Server Setup

A. Download and Install the Nexus server

   a. Verify its working at 8081

B. Download and Install the Sonar server

   a. Verify its working at 9000

Phase-1

6. Test Server Setup

A. Using ansible playbook script from jenkins job install the following:

   a. Docker

   b. Git

   c. Maven

Phase-1

7. Plugins and Credentials

A. Plugins
   a. Install Sonarqube Plugin
   b. Install AWS credentials plugin

B. Credentials
   a. AWS keys
   b. Private Key for Ansible
   c. Nexus Username and Password
   d. Docker Hub Creds (Optional in case of ECR)

Phase-2

Dependency

Phase-2 - Repository, Project Setup

1. Docker Repository

A. Create Repository for docker images

   a. ECR

   b. Docker Hub

Phase-2

2. Sonarqube Project Setup

A. Create a sonarqube Project

B. Then API token from security

C. Update setting of sonarqube server in system and scanner in tools

D. Add token in credentials

Phase-3

Deployment

Phase-3 - Jenkins Job

1. Options

A. Build Discarder

B. Timeout

C. Clean Workspace

2. Tools

A. Maven

Phase-3 - Jenkins Job

3. Environment

A. Custom Variables

B. Credentials

4. Agents

A. Based on requirement

Phase-3 - Jenkins Job

5. Stages

A. Checkout

B. Sonar test

C. Maven Build

D. Nexus Artifacts

E. Docker Build

F. Docker Hub Push/ ECR push

G. K8s Deployment

H. Post actions

Project Outcome

Things achieved:

A. Understanding of Continuous Integration and Continuous Deployment (CI/CD)

B. Proficiency in Jenkins Installation and Configuration

C. Creating and Managing Jenkins Jobs

D. Working with Jenkins Pipelines

E. Integration with Version Control Systems

F. Artifact Management and Distribution:

G. Testing and Quality Assurance

H. Best Practices and Troubleshooting