Jenkins Group Level Project

Problem1:

- What is a Jenkins plugin?

*A Jenkins plugin is a piece of code that enables Jenkins to extend its functionality beyond its initial configuration.*

- Name 5 Jenkins plugins and their use cases

1. *Git plugin: used to connect Jenkins jobs to remote repositories and run git operations against them.*
2. *GitHub Integration: used to integrate Jenkins directly with Github, pull files from Github to Jenkins, scheduled builds and process pull requests.*
3. *Maven plugin: used to This plugin provides an advanced integration for Maven 2/3 projects.*
4. *Kubenetes plugin: used for automating build agents on a Kubernetes cluster.*
5. *Docker Pipeline: used to build, test, and use Docker images from Jenkins Pipeline projects.*
6. *Docker Plugin: used to spin Docker containers and run builds on them automatically. With this plugin, DevOps teams can use a Docker host to dynamically provision a docker container as a Jenkins agent node that runs a single build.*
7. *SonarQube Scanner: used to integrate Jenkins with SonarQube for Continuous Inspection of code quality.*
8. *Slack Integration: used to integrate Jenkins into Slack, a chat client that also integrates with other tools. This can be used to send notifications about the build status into a specified slack channel.*
9. *Jacoco-plugin (JaCoCo): used to capture and visualize code coverage testing results for projects using the JaCoCo for code-coverage analysis.*
10. *Dashboard View plugin: used to create a customized view of Jenkins dashboard.*
11. *Jira: is an open source plugin used to integrate Jira Software* *to provide more visibility into the development pipeline.*

- What is the difference between scripted pipeline and declarative pipeline?

* *A Scripted pipeline is the older Jenkins code syntax that includes only node and stage blocks while,*
* *A Declarative pipeline is modern Jenkins mode that breaks down stages into multiple steps, thus making writing and reading Pipeline codes easier. It also allows addition of directives which can be easily created with a code generator.*

- Use generic syntax to demo scripted and declarative pipeline as code

scripted pipeline:

syntax:

node {

stages{

stage('This is a scripted pipeline'){

echo "Scripted pipelines have 2 basic structures"

}

stage("Structural components"){

echo "They consist of Nodes and Stages"

}

stage("Build"){

echo "Build scripted pipeline"

}

}

}

declarative pipeline:

syntax:

pipeline {

agent any

stages{

stage('This is a declarative pipeline'){

steps{

echo "Modern syntax model for Jenkins"

}

}

stage("Structurally different from scripted pipelines"){

steps{

echo "Scripted pipelines include steps and directives"

}

}

stage("Build"){

steps{

echo "Build declarative pipeline"

}

}

}

}

- What is a stage-level variable and how is it different from top-level variables?

* *Stage level variables are defined and called within a stage while top level variables are defined before the stages and can be called within any stage.*

- What information does Jenkins need to integrate with GitHub platform?

* *Github password*
* *Payload URL (Jenkins url)*
* *Secret*

- How do inject script into your pipeline?

- What are the similarities between a dependency and a plugin?

* A dependency is … while a plugin is …

- How do you configure service account credentials for Jenkins?

Problem2:

- Build an N-STAGE Jenkins pipeline where N=number of group members such that each stage should represent members name

- Use the git flow best practice and add a ReadMe file on your steps to contribute to a project

*ReadME*

*Steps to contribute to a project:*

* + *Create a folder (repository) for the project on the local pc.*
  + *cd into the local repository on VS code using wsl for windows-based system.*
  + *Clone the main repository for the project from GitHub into the local repository $git clone url*
  + *Confirm the branch with $git branch.*
  + *Create a new branch with $git branch branchname*
  + *Switch into the new branch with $git checkout branchname*
  + *Make inputs in the file.*
  + *Check new additions in working area with $git status*
  + *Move to staging area with $git add*
  + *Commit changes to local repository with $git commit -m*
  + *Use $git push to create and upstream branch of the committed file.*
  + *Switch to GitHub, navigate to the project repo, create a pull request and assign to the approving engineer.*
  + *After approval, merge the new branch with existing (main) branch.*

- Each member should write a simple two steps of (ps -ef and sudo systemctl status Jenkins)

Red: Pending

Yellow: for inputs

Green: done