JENKINS:

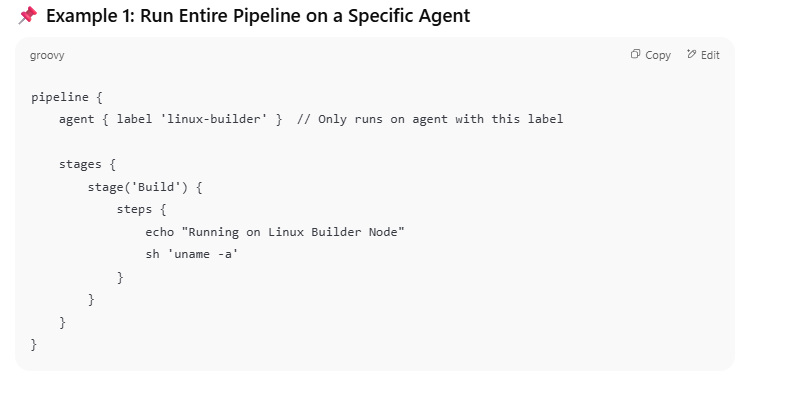
How do you pass parameters between in a jenkins declarative pipeline ?  
We can use script block to pass parameters in between stages.

|  |
| --- |
| def commitHash = ""  def dockerTag = ""    pipeline {  agent any    stages {  stage('Checkout Code') {  steps {  checkout scm  script {  // Get short git commit hash  commitHash = sh(script: 'git rev-parse --short HEAD', returnStdout: true).trim()  echo "Git Commit Hash: ${commitHash}"  }  }  }    stage('Build Docker Image') {  steps {  script {  dockerTag = "myapp:${commitHash}"  echo "Docker tag will be: ${dockerTag}"    // Simulate docker build  sh "echo docker build -t ${dockerTag} ."  }  }  }    stage('Deploy to Dev') {  steps {  script {  echo "Deploying Docker image: ${dockerTag} to Dev environment"  // Simulate deployment  sh "echo Deploying ${dockerTag} to dev cluster"  }  }  }  }  } |

* Stage 1 gets the **Git commit hash**.
* Stage 2 uses it to **tag a Docker image**.
* Stage 3 deploys that Docker image using the tag.

What is an agent in Jenkins? How do you configure a pipeline to on a specific agent?  
An **agent** (also called a **node**) is a machine (or container) where your pipeline’s steps run.  
 This could be:

* The **Jenkins controller** itself (not recommended for builds).
* A **dedicated build node** (slave) connected to Jenkins.

  
**How to Know Agent Labels?**

Go to **Jenkins Dashboard → Manage Jenkins → Manage Nodes**  
 Click on any node and see its **Label(s)**. Use those labels in agent { label 'your-label' }.

**------------------------------------------------------------------------------------------------------------------------**  
**Explain how do you use shared libraries in Jenkins ?**  
Using shared libraries we can reuse code across multiple pipelines. It allows our team to manages pipeline logic more centrally.  
  
**Directory Structure (inside Git repo)**

bash

CopyEdit

(root)  
└── vars/  
 └── myCustomStep.groovy # Scripted function callable as `myCustomStep()`  
└── src/  
 └── org/example/Helper.groovy # Helper class (for advanced usage)  
└── resources/  
 └── templates/template.txt # Files accessible via `libraryResource`  
└── README.md

## **🔧 Step 1: Configure Shared Library in Jenkins UI**

1. Go to **Jenkins Dashboard → Manage Jenkins → Configure System**.
2. Scroll to **Global Pipeline Libraries**.
3. Add a new library:
   1. **Name**: my-shared-lib (used in @Library('my-shared-lib'))
   2. **Default version**: main or master
   3. **Retrieval method**: Modern SCM (e.g., Git)
   4. Provide repository URL (e.g., GitHub, GitLab)

## **🛠️ Step 2: Use in Jenkinsfile**

@Library('my-shared-lib') \_ // Load the shared library by name  
  
pipeline {  
 agent any  
  
 stages {  
 stage('Use Shared Step') {  
 steps {  
 myCustomStep() // Defined in vars/myCustomStep.groovy  
 }  
 }  
 }  
}

Have you handled parallel execution in a pipeline? How and WHy ?

Yes, we used paralled execution to our Deploy and BVT job simultaneously by defining then in a parallel block   
  
this helped us to run the BVT test cases once the code is deployed to the K8s cluster and it will execute the BVT test cases on the pod and if the K8s cluster returns 200 OK status it will remove the older version and keep the newer version as the final version.  
  
  
**How do you secure your Jenkins?**   
A) Our jenkins server is only accessible on https.   
we secured our jenkins by using LAP for its authentication.  
Users once authenticated able to login to the jenkins but they will not have permissions   
  
**When it is deployed to specific environment how it takes that specific environment configurations?**

Basically, we have three different branches that are dev, uat and prod and we write different pipeline for each branch and we configure the credentials according to the respective branch this is how it takes configurations according to the specific deployment in specific environment.

**Best practice to provide credentials in Jenkins pipeline?**   
**How do you manage secrets in jenkins**

We need credential plugin to provide credentials and credential binding plugin to give credentials as environmental variables in our build job next thing in Jenkins GUI-> manage Jenkins -> manage credentials -> global credentials -> add credentials here we can add credentials as username and password, secret's file, secret text etc.

**difference between free style job and multibranch pipeline ?**  
Free style job :  
it is a simple job with all the configuration done manually on the jenkins UI   
One job per project.  
Simple build and test per branch   
  
multibranch pipeline :  
Automatically creates pipeline for each branch in the repository   
Jenkins will scan the git repo for branches   
We get full cicd per branch   
  
  
**How can you secure Jenkins?**

Securing Jenkins is a little lengthy process, and there are two aspects of securing Jenkins: (i) Access Control which includes authenticating users and giving them an appropriate set of permissions, which can be done in 2 ways.

* **Security Realm** determines a user or a group of users with their passwords.
* **Authorization Strategy** defines what should be accessible to which user. In this case, there might be different types of security based on the permissions granted to the user such as Quick and simple security with easy setup, Standard security setup, Apache front-end security, etc.

(ii) Protecting Jenkins users from outside threats

**JENKINS LDAP Configuration:**

We need an LDAP Server configured and create users and groups under the LDAP server.

JENKINS --> Manage Jenkins --> Configure Global Security --> Security Realm --> LDAP Server: SERVER:Port

root DN: dc=automations,dc=com

Manager DN: cn=manager, dc=linuxautomations,dc=com Manager Password: …......

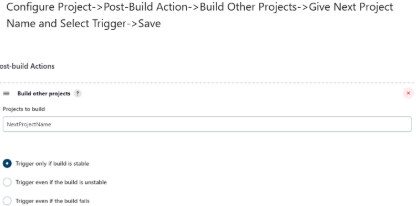
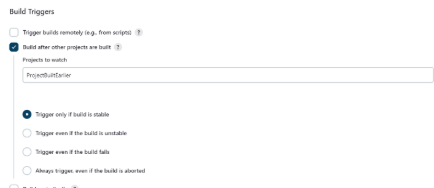
Test LDAP settings ----> user name and password

Therefore, ldap users can login to Jenkins

How to do Jenkins Backup?  
- We enabled backed on jenkins by installing a thin Backup plugin   
**Manage Jenkins > Manage Plugins**, switch to the **Available** tab, and install **Thin Backup**.  
- Once installed, go to **Manage Jenkins > Configure System** and scroll to the **ThinBackup** section.  
Configure Backup Settings

|  |  |
| --- | --- |
| Setting | Description |
| Backup Directory | **/var/lib/jenkins/JENKINS\_BACKUP** (created automatically if missing) |
| Schedule | Cron expressions for periodic backups |
| Differential Backup | Enable to back up only changed data |
| Max Backup Sets | Retain the latest N backups |
| Exclude Patterns | Regex for files/folders to skip |
| Archive as ZIP | Compress backup folders |
| Include Build Archives | Optionally include full build artifacts |

Perform manual backup as well  
Manage Jenkins --> ThinBackuo --> Click on Backup Now  
  
Aslo on the server side backup like we can have shell script to backup jenkins home directory or   
Specific high priority directories like config.xml (Jenkins conf file) , jobs , plugins. Secrets folder  
  
As our Jenkins instance is running in AWS EC2 we enabled EBS Snapshots

What are upstream and downstream Jobs ?  
  
Upstream jobs trigger another job after build   
  
Downstream jobs trigger itself when a particular build job is completed.  


### **How to create & use a Shared Library in Jenkins?**

Basic requirements for a Jenkins shared library to be used in a Pipeline Code are -

* A Repository with pipeline shared library code in SCM.
* An appropriate SCM Plugin configuration for the Jenkins instance.
* Global Shared Library should be configured in Jenkins Global configuration.
* Include the Shared Library in the Pipeline Code and use the methods defined in the Jenkins Shared Library.

**E.g.**

#!/urs/bin/env groovy  
@Library('fs\_jenkins\_shared\_library@v2.0.7')\_

**How do you secure your jenkins?**  
- We enabled SSO login for our Jenkins UI. Oly our org members will be able to do login   
- Once Logged in none if the users will get access jobs i=or any kind of configuration   
- We will ask their Trigram ID which they use to access jenkins   
- We will add then to required jobs to provide then will all the necessary permissions like read, build, view.  
  
How do you implement approval gated in a jenkins pipeline [Ex: manual approval before production]   
Using the input Step (Declarative Pipeline)