**Installation of Jenkins:**

* Jenkins required jdk before install jdk first then download Jenkins\_rpm and install
* systemctl start jenkin
* url to opne ip:8080 default port
* set proxy to install plugins

**Password of admin:**

/var/lib/jenkins/secrets/initialAdminPassword

**Proxy\_to\_install plugins: Nokia prox**

**2. Change of Jenkins user to root:**

vim /etc/sysconfig/jenkins

2. Find this $JENKINS\_USER and change to “root”:

$JENKINS\_USER="root"

3. Then change the ownership of Jenkins home, webroot and logs:

chown -R root:root /var/lib/jenkins

chown -R root:root /var/cache/jenkins

chown -R root:root /var/log/jenkins

**3. Seen how to pipeline the jobs, and different style viewing of jobs using plugins build monitor, build pipeline views**

**Poll mailbox trigger plugin -> will help to run job when you get mail with matched subject in Jenkins**

**4. Seen configure nodes in Jenkins and executing job on remote node:: mangage Jenkins -> manage nodes -> add remote machine details and while creating jobs select give label name**

**Select option ‘Restrict where this project can be run’**

**5. creating job with parameters: to create job with parameters select option “This project is parameterized” then select drop-down list as per yours**

**Jenkins pipelines:**

<https://www.edureka.co/blog/jenkins-pipeline-tutorial-continuous-delivery>

 These pipelines are a **collection of Jenkins jobs** which trigger each other in a specified sequence.

The key feature of this pipeline is to define the entire deployment flow through code.

**What is a Jenkinsfile?**

A Jenkinsfile is a text file that stores the entire workflow as code and it can be checked into a SCM on your local system. How is this advantageous? This enables the developers to **access, edit and check the code at all times**.

The Jenkinsfile is written using the Groovy DSL and it can be created through a text/groovy editor or through the configuration page on the Jenkins instance. It is written based on two syntaxes, namely:

### **Declarative pipeline syntax**

### **Scripted pipeline syntax**

1. Declarative pipeline is a relatively new feature that supports the pipeline as code concept. It makes the pipeline code easier to read and write. This code is written in a Jenkinsfile which can be checked into a source control management system such as Git.
2. Whereas, the scripted pipeline is a traditional way of writing the code. In this pipeline, the Jenkinsfile is**written on the Jenkins UI instance**. Though both these pipelines are based on the groovy DSL, the scripted pipeline uses stricter groovy based syntaxes because it was the first pipeline to be built on the groovy foundation. Since this Groovy script was not typically desirable to all the users, the declarative pipeline was introduced to offer a simpler and more optioned Groovy syntax.

**Simple groovy syntax:**

**pipeline{**

**agent** No agent type specified. Must be one of [any, docker, dockerfile, label, none]

**stages{**

**stage('Build'){**

**steps{**

**echo "welcome to jenkins pipeline,build software "**

**}**

**}**

**stage('Deploy'){**

**steps{**

**echo "Deploy the application on agent"**

**}**

**}**

**stage('Test'){**

**steps{**

**echo "Test application"**

**}**

**}**

**}**

**}**

**Parallel stages:**

**stage('Test'){**

**parallel {**

**stage('Unit Test') {**

**steps {**

**echo "Running the unit test..."**

**}**

**}**

**stage('jUNIT Test') {**

**steps {**

**echo "Running the unit test..."**

**}**

**}**

**}**

**}**

**Info about artifactory repos** <https://jenkins.io/doc/pipeline/examples/>