Jenkins - Table of Contents

Jenkins Installation

- Launch an EC2 instance with Amazon Linux 2 with below `userdata`

```
#!/bin/bash
sudo yum update -y
sudo wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-
ci.org/redhat/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat/jenkins.io.key
sudo yum install java-1.8.0 -y
sudo amazon-linux-extras install epel -y
sudo yum install jenkins -y
sudo yum install tree -y
sudo yum install git -y
sudo service jenkins start
sudo systemctl enable jenkins
```

This package installation will:

- Setup Jenkins as a daemon launched on start. See `/etc/init.d/jenkins` for more details.
- Create a `jenkins` user to run this service.
- Direct console log output to the file `/var/log/jenkins/jenkins.log`.
- Check this file if you are troubleshooting Jenkins.
- Populate `/etc/default/jenkins` with configuration parameters for the launch, e.g JENKINS_HOME
- Set Jenkins to listen on port 8080. Access this port with your browser to start configuration.
- Login to EC2 Jenkins Server using ssh.

```
netstat -nltp
sudo service jenkins status
sudo service jenkins stop
sudo service jenkins restart
```

- Check Jenkins Port Information

```
sudo cat /etc/sysconfig/jenkins | grep -i port
ps -elf | grep 8080
```

- Lets look at the jenkins log file

```
sudo cat /var/log/jenkins/jenkins.log
```

- Access the Jenkins UI

```
http://public-ip:8080
```

- Admin Password is written to a file

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

- Select `Install Suggested Plugins` only
- It will ask for creating for first admin user, enter details as required.
- A linux user with name `jenkins` is created while Jenkins Installation, add `jenkins` user in linux to `sudoers` group.
- All Jobs in jenkins are executed by `jenkins` linux user

```
cat /etc/passwd | grep -i 'jenkins'
sudo usermod -a -G wheel jenkins
id jenkins
```

- Current home directory for jenkins is **/var/lib/jenkins**
- To view the Jenkins Systems Information, navigate to `Manage Jenkins` > `System Information`

Setup JDK for Jenkins

- Install OpenJDK 8 JDK

- To install OpenJDK 8 JDK using yum, run this command:

```
sudo yum install java-1.8.0-openjdk-devel
```

- Use below find command to search for files with name "jdk"

```
sudo find / -name "*jdk*"
```

Provide the path under provide the path under : Go to `Jenkins Dashboard -> Manage Jenkins -> Global Tool Configuration > JDK` > Give a Name > Give appropriate path to JDK e.g `/usr/lib/jvm/java-1.8.0-openjdk`

Installing the Git plugin

- We need to install the Git client on to our Jenkins server

```
sudo yum install git -y
git --version
```

- Check if Git Plugin is installed under: `Manage Jenkins > Manage Plugins > Installed Tab > Filter "Git Plugin"` , if not install it from `available` tab.
- This will prompt Jenkins to download the plugin, install it, and restart Jenkins to make it available for use.

Jenkins Freestyle Project

- Click on **New Item** then enter an item name, select **Freestyle project**.
- Under **Add Build Step** , enter some build commands i.e bash commands `printenv` to be executed in the freestyle project.

Build with Parameters

- Sometimes, it is useful/necessary to have your builds take several **parameters**.
- This can to run a Pipeline Job as per SDLC Environment or any other value to be passed on Job Runtime.
- Under a specific jenkins project, select `Configure` option, select the checkbox `This project is parameterized` and `Add Parameter`
- For testing the value of the runtime parameter, keep `Source Code Management` as `None`
- The parameters are available as `environment variables`. So a shell `$PARAM_VAR`,can be used to access these values.

Integrate Jenkins with Github Repo

- Select the GitHub project checkbox and set the Project URL to point to your GitHub Repository. `https://github.com/YourUserName/`
- Under Source Code Management Section: Provide the Github Repository URL where Source Code is present, keep the branch as master.
- Go to Jenkins Project -> Configure -> Under Build Environment Build Step > Select `Execute Shell Script` from dropdown > `write shell commands`

> Execute a shell script stored in Github repo by providing path.

- Click on **Build Now** to Build this Project

Jenkins jobs and workpace information

The Jenkins home directory structure Colons can be used to align columns.

Directory	Description
jobs	Path `/var/lib/jenkins/jobs`. It contains configuration details about the build jobs that Jenkins manages, as well as the artifacts and data resulting from these builds.
workspace	Path `/var/lib/jenkins/workspace`. It is where Jenkins builds your project: it contains the `source code` Jenkins checks out, plus any files generated by the build itself.This workspace is reused for each successive build, there is only ever one workspace directory per project, and the disk space it requires tends to be relatively stable.

Jenkins Environment Variables:

- To view all the environment variables simply append `env-vars.html` to your Jenkins Server's URL. For e.g `http://<JENKINS_IP>:8080/env-vars.html`
- Create a simple free style job to display the value of the environment variables that are set for a Jenkins Job:
- Under Build Section > Add build step > Execute shell , add below commands:

```
echo "BUILD_NUMBER" :: $BUILD_NUMBER
echo "BUILD_ID" :: $BUILD_ID
echo "BUILD_DISPLAY_NAME" :: $BUILD_DISPLAY_NAME
echo "JOB_NAME" :: $JOB_NAME
echo "EXECUTOR_NUMBER" :: $EXECUTOR_NUMBER
echo "NODE_NAME" :: $NODE_NAME
echo "NODE_LABELS" :: $NODE_LABELS
echo "WORKSPACE" :: $WORKSPACE
echo "JENKINS_HOME" :: $JENKINS_HOME
echo "JENKINS_URL" :: $JENKINS_URL
echo "BUILD_URL" ::$BUILD_URL
echo "JOB_URL" :: $JOB_URL
echo "Below output is all the environment variable in Jenkins"
printenv
```

- The `printenv` command prints all the Jenkins Environment Variables set for that specific Build.

Jenkins Github SSH Integration

Github SSH Keys Configuration

- Generate ssh keys and add to `Github Account` **OR** `Specific Github Repo`

```
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

- This creates a new ssh key pair, using the provided email as a label.
- SSH Keys can be configured as per below:

Github Account SSH Keys	Github Repository SSH Keys
`Github UI` > `Settings > SSH and GPG Keys > New SSH key > Add SSH Key` > `Confirm password`	`Github Repository` > `Settings` > `Deploy keys` > `Add Deploy Key` > `Enter Name and Public ssh key Allow Write Access` > `Add Key`

Jenkins Credentials

- Jenkins configuration to access private repo:
  - Navigate to `Jenkins dashboard -> Managed Jenkins > Manager > Jenkins > Credentials -> System -> Global credentials -> Add credentials.`
  - From dropdown select **SSH Username with Private Key** and specify the **ID , Username** and configure the **SSH Private Key** which is stored in **.ssh** folder under the file name **id_rsa**.
- While setting up a Job in Jenkins, add the credentials created above to the credentials section in `Source Code Management` under the **Repository URL**.
- Execute the Job and test whether Jenkins Job is able to check out Github Specified Branch

Jenkins Build with Jenkinsfile

- Navigate to Provide a name for your new item and select `Pipeline`
- Enter below Pipeline code into the Script text area
- **Jenkinsfile**

```
pipeline {
    agent any
    parameters {
        string(name: 'myParameter', defaultValue: 'myVal', description: 'Enter
Parameter value?')
    }
    stages {
        stage('Build') {
            steps {
                echo 'Building..'
                echo "Running ${env.BUILD_ID} on ${env.JENKINS_URL}"
            }
```

```
        }
        stage('Test') {
            steps {
                echo 'Testing..'
                echo "${params.myParameter} is value retrieved!"
            }
        }
        stage('Deploy') {
            steps {
                echo 'Deploying....'
            }
        }
    }
}
```

Jenkins pipeline-syntax

- Jenkins has a built-in **Snippet Generator** utility that is helpful for creating bits of code for individual steps, discovering new steps provided by plugins, or experimenting with different parameters for a particular step.
- The Snippet Generator is dynamically populated with a list of the steps available to the Jenkins instance. The number of steps available is dependent on the plugins installed which explicitly expose steps for use in Pipeline.
- To generate a **step snippet** with the **Snippet Generator**:
    - Navigate to the Pipeline Syntax link from a configured Pipeline, or at `${YOUR_JENKINS_URL}/pipeline-syntax`.
    - Select the desired step in the Sample Step dropdown menu
    - Use the dynamically populated area below the Sample Step dropdown to configure the selected step.
    - Click Generate Pipeline Script to create a snippet of Pipeline which can be copied and pasted into a Pipeline.
- The code for a Jenkinsfile should be available in Github Repo
- Click the Add Source button, select git choose the type of repository you want to use and fill in the details.
- Click the Save button and watch your first Pipeline run!

Configuring Credentials in Jenkinsfile

- Navigate to `Jenkins Home page > Credentials > System > Add Credentials`.
- Select Scope as 'Global' **Global** - When credentials are to be added for a Pipeline project/item. **System** - When credentials are to be added for a Jenkins itself to interact with system administration functions., such as email authentication, agent connection, etc. This option applies the scope of the credential to a single object only.

- Types of credentials:
    - **Secret text** - a token such as an API token (e.g. a GitHub personal access token)
    - **Username and password** - which could be a colon separated string in the format username:password

Reference

- https://issues.jenkins.io/browse/JENKINS-66361