**Linux**

Download https://www.ubuntu.com/download/desktop

Install Ubuntu(64-bit) in VirtualBox VM

**Jenkins**

<https://jenkins.io/doc/pipeline/tour/getting-started/>

Download <http://mirrors.jenkins.io/war-stable/latest/jenkins.war>

-Install

-Java

Sudo apt install openjdk-8-jdk-headless

Java –version

Javac –version

-Git

Sudo apt install git

-initialAdminPassword

initialAdminPassword is required upon first login and can be found in

{home}/.jenkins/secrets/initialAdminPassword

-initial suggested plugins

-set admin account

…

-Credentials

-Github Plugin

Let Jenkins manage the GitHub Webhooks via:

-access token

Github->account settings->developer settings->personal access tokens

Generate token

-manage Jenkins->configure system->add github server

Credentials->secret text

Give Jenkins a public url to be reachable

-ngrok

ngrok exposes local servers behind NATs and firewalls to the public internet over secure tunnels.

Create account->Login->Download->follow commands:

-unzip

-authorize

-run: ./ngrok http 8080

Manage Jenkins->configure system->Jenkins URL

Set Jenkins URL to https://etc.. which ngrok gives

**Webhook**

Github->Repo->Settings-Webhooks

Jenkins will automatically add webhook here (via Github Server settings), make sure it is there

Or add manually: {JenkinsURL}/github-webhook/

**Pipeline**

New item->pipeline->set name

On General Tab->Github project->set github url

On Build trigger Tab->GitHub hook trigger for GITScm polling

On Pipeline Tab->Definition->Pipeline script from SCM

->SCM->Git

->Set Repo URL

->Credentials->Username with Password

**Nexus**

-Repository Connector Plugin

-Nexus Platform Plugin XXXXXXXXXXX

Sudo apt install maven

-Proxy Repo vs Hosted Repo

Proxy Repo (Cache for public library)

A proxy repository is linked to a remote repository, a public library.

A request to the Proxy Repo is checked against the local content.

If no local content is found, the request is forwarded to the remote repository. The component is then retrieved from the remote repo and stored locally to act as a cache.

If local content is found, the remote repository is NOT requested. Thus eliminating network bandwidth cost and time overhead of retrieving the component

Hosted Repo

A hosted repository is intended for internal releases/libraries and 3rd party components not available in external public repositories.

-Hosted: 3rd party

-Hosted: Release

Comparable to Production area

Released artifacts are considered to be solid, stable, and perpetual in order to guarantee that builds which depend upon them are repeatable over time.

-Hosted: Snapshots

Comparable to Development/Stage area

Snapshots capture a work in progress and are used during development.

When 1.0-SNAPSHOT version is deployed, SNAPSHOT is turned into timestamp

**Deploy to Nexus**

Pipeline:

Stages:

Git checkout

Mvn deploy

CleanWs()

Adjust pom.xml

distributionManagement -> Where to deploy

repositories -> Where to download artifacts from

1. **<distributionManagement>**
2. <!-- Publish the versioned releases here -->
3. **<repository>**
4. **<id>**vineetmanohar-nexus**</id>**
5. **<name>**vineetmanohar nexus**</name>**
6. **<url>**dav:http://nexus.vineetmanohar.com/nexus/content/repositories/releases**</url>**
7. **</repository>**
9. <!-- Publish the versioned releases here -->
10. **<snapshotRepository>**
11. **<id>**vineetmanohar-nexus**</id>**
12. **<name>**vineetmanohar nexus**</name>**
13. **<url>**dav:http://nexus.vineetmanohar.com/nexus/content/repositories/snapshots**</url>**
14. **</snapshotRepository>**
15. **</distributionManagement>**
16. <!-- download artifacts from this repo -->
17. **<repositories>**
18. **<repository>**
19. **<id>**vineetmanohar-nexus**</id>**
20. **<name>**vineetmanohar**</name>**
21. **<url>**http://nexus.vineetmanohar.com/nexus/content/groups/public**</url>**
22. **<releases>**
23. **<enabled>**true**</enabled>**
24. **</releases>**
26. **<snapshots>**
27. **<enabled>**true**</enabled>**
28. **</snapshots>**
29. **</repository>**
30. **</repositories>**

Adjust settings.xml of maven

Where to look for Nexus repo

1. **<settings>**
2. **<servers>**
3. **<server>**
4. <!-- this id should match the id of the repo server in pom.xml -->
5. **<id>**vineetmanohar-nexus**</id>**
6. **<username>**deployment**</username>**
7. **<password>**password\_goes\_here**</password>**
8. **</server>**
9. **</servers>**
10. **</settings>**

http://www.vineetmanohar.com/2010/06/getting-started-with-nexus-maven-repo-manager/

**Collect**

Mvn dependency:get –Dartifact=groupId:artifactId:version[:packaging][:classifier]

Mvn dependency:copy –Dartifact=groupId:artifactId:version[:packaging][:classifier]

-Doutputdirectory=

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Alternatively, you can use Maven to retreive a snapshot version. First download it to your local maven repository using the "dependency:get" goal":

mvn dependency:get -DremoteRepositories=http://localhost:8081/repository/maven-public -DgroupId=org.foo -DartifactId=project -Dversion=1.0.0-SNAPSHOT -Dtransitive=false

Then use the "dependency:copy" goal to move the snapshot from your local repository to wherever you like:

mvn dependency:copy -Dartifact=org.foo:project:1.0.0-SNAPSHOT -DoutputDirectory=/some/output/path

<https://support.sonatype.com/hc/en-us/articles/213465488-How-can-I-retrieve-a-snapshot-if-I-don-t-know-the-exact-filename->

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Curl [options] URL

-O --remote-name

-X –request

-u --user user:password, only user will prompt for password

--netrc-file <filepath>

Format <Machine host.domain.com login u password p>

URL={repoURL}/groupId/artifactId/version/artifactId-version.packaging

Curl –X GET –f –u user:passw

http://localhost:8081/repository/SAMPLE-REL/com/test/project/1.0/project-1.0.jar -O

Wget [options] URL

--user=user

--password=password

Wget --user=user --password=password

http://localhost:8081/repository/SAMPLE-REL/com/test/project/1.0/project-1.0.jar