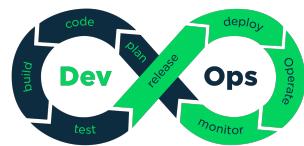


DEVOPS



GIT:-

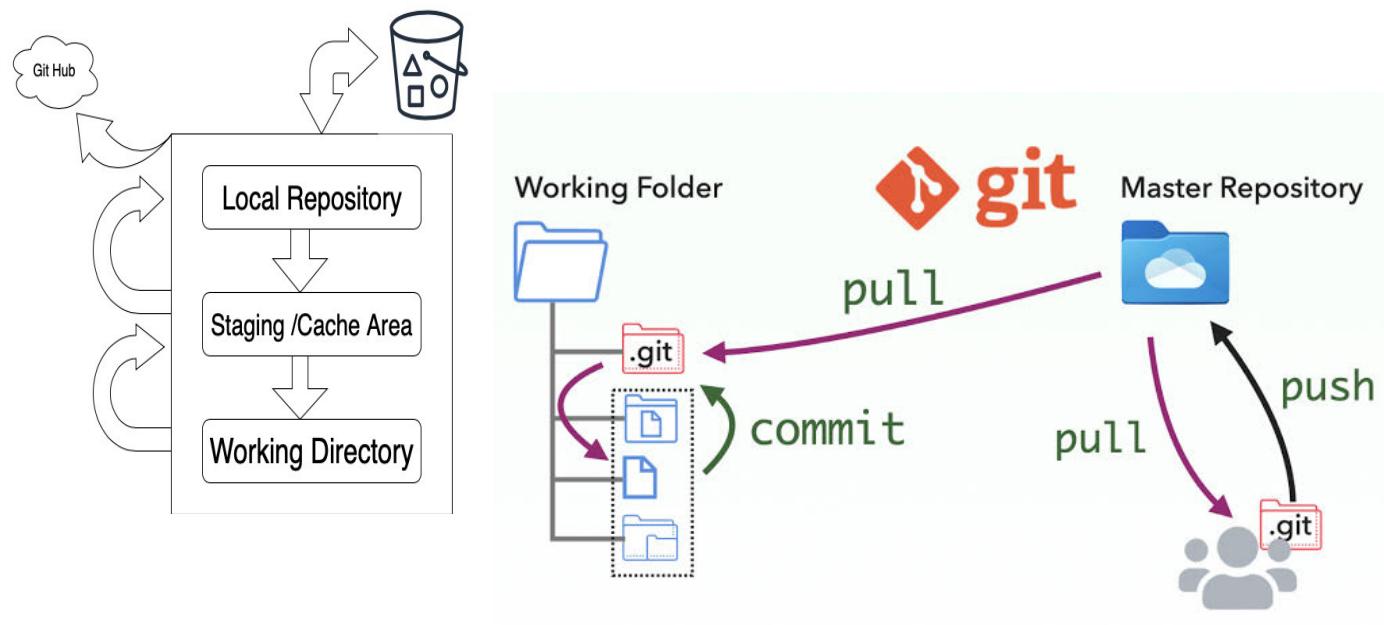
Git is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows.

Git was created by Linus Torvalds in 2005 for development of the Linux kernel, with other kernel developers contributing to its initial development. Its current maintainer since 2005 is Junio Hamano. As with most other distributed version-control systems, and unlike most client–server systems, every Git directory on every computer is a full-fledged repository with complete history and full version-tracking abilities, independent of network access or a central server.

Install git:-

- You should be running a server with any Ubuntu 16.04 LTS release.
- You will need to log in to SSH via the root user.

First, as always, we should start out by running general OS and package updates. On Ubuntu we'll do this by running:



```
>> apt-get update  
>> apt-get install git-core  
>> git --version
```

Installing GIT - apt-get install git

Telling the GIT to track this folder - git init

Colors – Red color = Files in working directory

Green color = Files in staging / cache Area

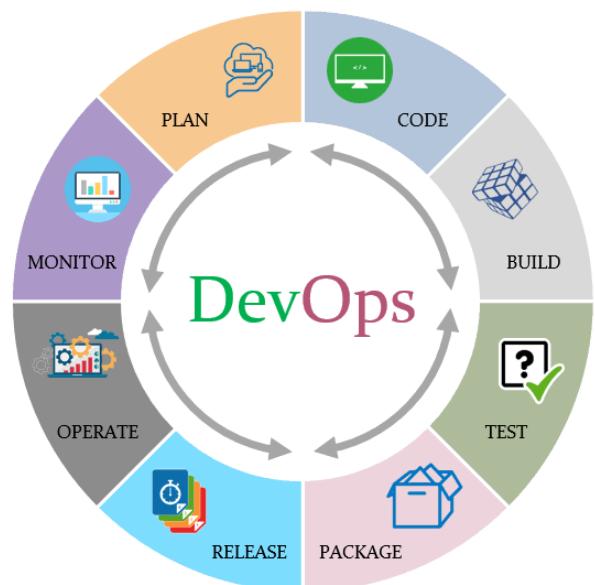
Status Check – git status (for checking the tracking of files)

Commit Id's – generally called as SHA1 number

Git init: -

To track the particular folder and git will only take care about the files but not folders For checking whether it is installed or not check the hidden files

```
>> ls -a (or) ls -al  
>> git config --global user.name "XXnameXX"  
>> git config --global user.email "XXemail IDXX"  
>> git add filename (or) .[for adding complete files]  
>> git commit -m "message for that task"  
>> git commit -am "message for the task"  
>> git log - --oneline  
>> git show commitid  
>> vi .gitignore  
  *.html  
  *.jpg
```



```
! filename.html  
>> "git add -f filename"  
>> "git checkout filename"
```

Git SERVER:-

Development of the GitHub platform began on October 19, 2007.[55][56] [57] The site was launched in April 2008 by Tom Preston-Werner, Chris Wanstrath, P. J. Hyett and Scott Chacon after it had been made available for a few months prior as a beta release.[58]

Projects on GitHub can be accessed and manipulated using the standard Git command-line interface and all of the standard Git commands work with it. GitHub also allows registered and unregistered users to browse public repositories on the site. Multiple desktop clients and Git plugins have also been created by GitHub and other third parties that integrate with the platform.

The site provides social networking-like functions such as feeds, followers, wikis (using wiki software called Gollum) and a social network graph to display how developers work on their versions ("forks") of a repository and what fork (and branch within that fork) is newest.

A user must create an account in order to contribute content to the site, but public repositories can be browsed and downloaded by anyone. With a registered user account, users are able to have discussions, manage repositories, submit contributions to others' repositories, and review changes to code. GitHub began offering unlimited private repositories at no cost in January 2019 (limited to three contributors per project). Previously, only public repositories were free.

Installation :-

- >> JAVA 8 version need to be installed
- >> Terminal should be updated
- >> Should have gitbucket .war should be downloaded
- >> IP Address should be Reserved and should fix manually
- >> Change to Root user - sudo su –root
- >> Install the Vim software – apt-get install vim

```
>> apt-get install software-properties-common  
>> apt-get update  
>> apt-get install default-jre  
>> apt-get install default-jdk  
>> add-apt-repository - rppa:webupdsteam/java  
>> apt-get update  
>> apt-get install oracle-java8-installer  
>> java --version  
>> Download Gitbucket.war  
>> Go to the path were the gutbucket.war file was situated  
>> Java -jar gutbucket.war  
>> java -jar gutbucket.war --port =8018  
>> apt - get install git
```

Using local Git bucket :-

```
>> mkdir myproject – Create a directory  
>> cd myproject – navigate to directory  
>> git init – initialize the git  
>> touch tarun – create a file in myproject  
>> git status  
>> git add tarun  
>> git commit –m ‘commit message’  
>> git log  
>> gibucket – sign In – root/root (username & password)  
>> Goto system Adminstration – New user – Create user with credentials – sign  
out - sign in with newly created user  
>> New repository - Create a repository  
>> git remote add origin URL  
>> git push –u origin master  
>> View the file called “.gitbucket” (hidden folder)  
>> Give the command “- tree .gitbucket “ to view the files in the repository
```

Git Branches:-

Branching, in version control and software configuration management, is the duplication of an object under version control (such as a source code file or

a directory tree) so that modifications can occur in parallel along multiple branches.

Branches are also known as trees, streams or codelines. The originating branch is sometimes called the parent branch, the upstream branch (or simply upstream, especially if the branches are maintained by different organizations or individuals), or the backing stream. Child branches are branches that have a parent; a branch without a parent is referred to as the trunk or the mainline.

```
>> git branch  
>> git branch newbranchname  
>> git checkout branchtochange  
>> git merge branchnametomerge  
>> git checkout master  
>> git branch -D branchname  
>> git push origin --delete branchname
```

Stash Area:-

```
>> git add .  
>> git stash save filename  
>> git stash list – To view the stashed files
```

Play with data in Stash Area

```
>> Copy + paste = Take a copy from stash area and use it in normally git stash apply stashID  
>> Cut + paste = Move a file from stash and use it normally git stash pop stashID  
>> Delete = Remove files from stash Area  
>> git stash drop stashID
```

Creating Version tags:-

```
>> git tag versionnumber = Creating a version tag  
>> git tag = Wrapping the files and pushing into version  
>> git push -u myproject versionnumber = Pushing into github  
>> git tag -d versionnumber = Remove versions locally
```

>> git push -u myproject --delete versionnumber = Delete the release in the git hub

Email Notification

>> Whatever happens in the github will be notified through email

>> Steps to activate email notification

>> Login into Git Hub - myproject – Settings – Notifications – Add - Email Address

Backup and Restore

>> For taking the backup of the files of the git bucket .It is a hidden folder.

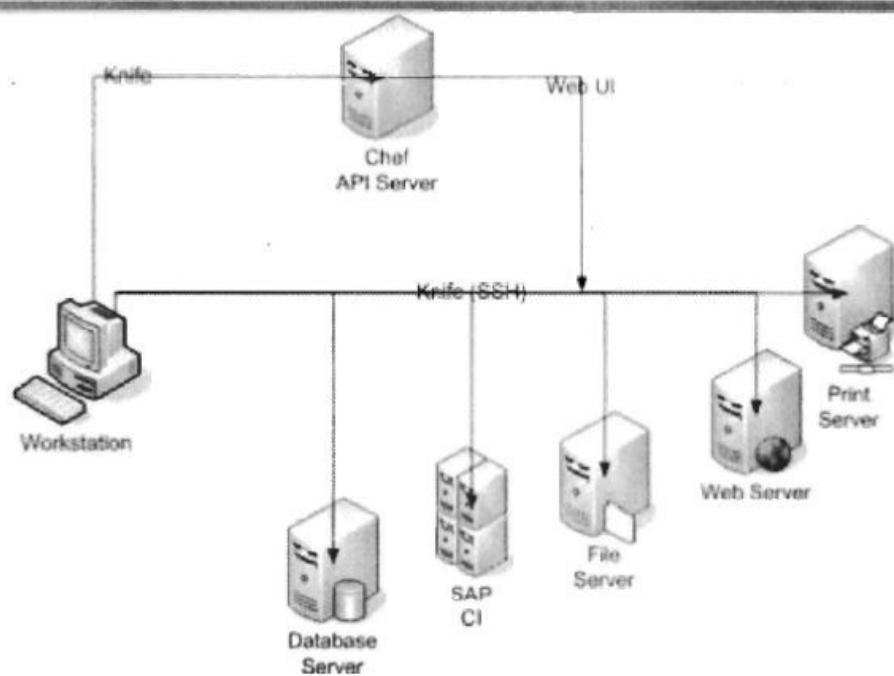
>> ls -a (View hidden files)

>> open the .gitbucket fil

>> There we can see the files which were pushed

CHEF:-

Chef Architecture



Chef is a company and the name of a configuration management tool written in Ruby and Erlang. It uses a pure-Ruby, domain-specific language (DSL) for writing system configuration "recipes". Chef is used to streamline the task of configuring and maintaining a company's servers.

The user writes "recipes" that describe how Chef manages server applications and utilities (such as Apache HTTP Server, MySQL, or Hadoop) and how they are to be configured. These recipes (which can be grouped together as a "cookbook" for easier management) describe a series of resources that should be in a particular state: packages that should be installed, services that should be running, or files that should be written. These various resources can be configured to specific versions of software to run and can ensure that software is installed in the correct order based on dependencies.

Chef can run in client/server mode, or in a standalone configuration named "chef-solo". In client/server mode, the Chef client sends various attributes about the node to the Chef server. The server uses Elasticsearch to index these attributes and provides an API for clients to query this information. Chef recipes

can query these attributes and use the resulting data to help configure the node.

Chef-server installation:-

```
>> hostname -f
>> cd ~
wget https://opscode-omnibus-
packages.s3.amazonaws.com/ubuntu/12.04/x86_64/chef-server_11.0.10-
1.ubuntu.12.04_amd64.deb
>> sudo dpkg -i chef-server*
>> sudo chef-server-ctl reconfigure
>> https://server_domain_or_IP
>> Default Username: admin
>> Default Password: p@ssw0rd1
>> mkdir -p ~/chef-repo/.chef
>> https://server_domain_or_IP
>> #chef-manage-ctl reconfigure
>> #chef-server-ctl user-create student student "student@pivotal.com"
"redhat" -f student.pem
>> #chef-server-ctl org-create myorg "pivotalsoft" -a student -f myorg-
validator.pem
>> #chef-server-ctl restart (for restart)
>> #chef-server-ctl start (for start)
>> #chef-server-ctl stop (for stop)
```

chef node installation:-

```
>> updat ip & hostadd
>> #dpkg -i chef-client.....
>> mkdir -p /etc/chef
copy both .pem files
>> cd /etc/chef
>> vi client.rb
log_level      :info
log_location    STDOUT
chef_server_url 'https://chefserver.pivotal.com/organizations/myorg'
```

```
validation_client_name 'myorg-validator'  
validation_key      '/etc/chef/myorg-validator.pem'  
client_key         '/etc/chef/student.pem'  
trusted_certs_dir  '/etc/chef/trusted_certs'
```

```
>> knife ssl fetch -s https://chefserver.pivotal.com  
>> knife ssl check -s https://chefserver.pivotal.com  
>> useradd rishi  
>> passwd rishi  
>> usermod -aG sudo rishi  
>> apt-get install ssh  
>> ssh-keygen
```

Chef workstation installation:-

```
>> update ip and host address  
>> dpkg -i chef-work.....  
>> cd /root/chef-repo/.chef  
copy both .pem files into .chef folder  
>> ls  
>> vi knife.rb  
log_level      :info  
log_location    STDOUT  
node_name       'student'  
client_key      '/root/chef-repo/.chef/student.pem'  
validation_client_name 'myorg-validator'  
validation_key    '/root/chef-repo/.chef/myorg-validator.pem'  
chef_server_url  'https://chefserver.pivotal.com/organizations/myorg'  
cookbook_path    ['/root/chef-repo/cookbooks']  
>> knife ssl fetch / knife ssl fetch -s https://chefserver.pivotal.com  
>> knife ssl check / knfie ssl check -s https://chefserver.pivotal.com  
>> knife bootstrap 192.168.0.221 --ssh-user rishi --sudo --identity-file  
~/.ssh/id_rsa --node-name chefnode.pivotal.com  
#knife node list
```

Chef cookbooks:-

Writing cookbooks/recipes

sample cookbooks:-

```
>> chef generate cookbook sample_file
>> vi /chef/cookbook/sample_file/recipes/default.rb
file "/tmp/test.txt" do
  owner "root"
  group "root"
  mode "0644"
  content "haiii this is test file"
  action :create
end
>> knife cookbook upload sample_file
>> knife node run_list add chefnode.pivotal.com sample_file
>> go to chefnode add type "chef-client"
```

Creates the sysadmin group and users:-

```
users_manage 'sysadmin' do
  group_id 2300
  action [:create]
end
```

Creates the testgroup group, and users

```
users_manage 'testgroup' do
  group_id 3000
  action [:create]
  data_bag 'test_home_dir'
end
```

Creates the nfsgroup group, and users

```
users_manage 'nfsgroup' do
  group_id 4000
  action [:create]
  data_bag 'test_home_dir'
  manage_nfs_home_dirs false
end
```

```
>> knife cookbook upload users  
>> knife node run_list add chefnode.pivotal.com sample_file  
>> go to chefnode add type "chef-client"
```

recipe for apache server:-

```
>> chef generate cookbook apache  
service['apache2'] is defined in the apache2_default_install resource but other  
resources are currently unable to reference it. To work around this issue,  
define the following helper in your cookbook:  
service 'apache2' do  
  extend Apache2::Cookbook::Helpers  
  service_name lazy { apache_platform_service_name }  
  supports restart: true, status: true, reload: true  
  action :nothing  
end  
  
apache2_install 'default_install'  
apache2_module 'headers'  
apache2_module 'ssl'  
  
apache2_default_site 'foo' do  
  default_site_name 'my_site'  
  template_cookbook 'my_cookbook'  
  port '443'  
  template_source 'my_site.conf.erb'  
  action :enable  
end  
>> knife cookbook upload sample_file  
>> knife node run_list add chefnode.pivotal.com apache  
>> go to chefnode add type "chef-client"
```

Chef roles:-

```
>> knife role bulk delete REGE  
>> knife role create ROLE_NAME (options)
```

```
>> knife role create role1
>> knife role edit ROLE_NAME

{
  "name": "role1",
  "default_attributes": {
  },
  "json_class": "Chef::Role",
  "run_list": ["recipe[cookbook_name::recipe_name]",
    "role[role_name]"
  ],
  "description": "",
  "chef_type": "role",
  "override_attributes": {
  }
}
>> knife role show ROLE_NAME
>> knife cookbook upload recipe
>> knife node run_list add chefnode.pivotal.com apache
```

To uninstall:-

```
>> chef-server-ctl uninstall
>> chef-manage-ctl cleanse
>> opscode-analytics-ctl uninstall
>> opscode-reporting-ctl uninstall
>> dpkg -P chefdk
>> rpm -qa *chef*
>> yum remove <package>
>> dpkg --list | grep chef # or dpkg --status chef
>> dpkg -P chef
>> sudo rm -rf /opt/chef
>> sudo rm -rf /etc/chef
```

Maven:-

Maven is a build automation tool used primarily for Java projects. Maven can also be used to build and manage projects written in C#, Ruby, Scala, and other languages. The Maven project is hosted by the Apache Software Foundation, where it was formerly part of the Jakarta Project.

Maven addresses two aspects of building software: how software is built, and its dependencies. Unlike earlier tools like Apache Ant, it uses conventions for the build procedure, and only exceptions need to be written down. An XML file describes the software project being built, its dependencies on other external modules and components, the build order, directories, and required plug-ins. It comes with pre-defined targets for performing certain well-defined tasks such as compilation of code and its packaging. Maven dynamically downloads Java libraries and Maven plug-ins from one or more repositories such as the Maven 2 Central Repository, and stores them in a local cache. This local cache of downloaded artifacts can also be updated with artifacts created by local projects. Public repositories can also be updated.

Maven is built using a plugin-based architecture that allows it to make use of any application controllable through standard input.

Maven installation:-

```
>> sudo apt-get update -y  
>> sudo apt-get upgrade -y  
>> add-apt-repository ppa:webupd8team/java  
>> apt-get update -y  
>> apt-get install oracle-java8-installer  
>> java -version  
>> wget http://www-eu.apache.org/dist/maven/maven-  
3/3.3.9/binaries/apache-maven-3.3.9-bin.tar.gz  
>> tar -xvzf apache-maven-3.3.9-bin.tar.gz  
>> mv apache-maven-3.3.9 maven  
>> nano /etc/profile.d/mavenenv.sh  
export M2_HOME=/opt/maven  
export PATH=${M2_HOME}/bin:${PATH}  
>> chmod +x /etc/profile.d/mavenenv.sh  
>> source /etc/profile.d/mavenenv.sh  
>> tar -xvf apache-maven -C /opt/
```

```
>> vi /etc/profile.d/apache-maven.sh
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
export M2_HOME=/opt/apache-maven
export MAVEN_HOME=/opt/apache-maven
export PATH=${M2_HOME}/bin:${PATH}
>> apt-get install maven
>> mvn --version
>> mvn archetype:generate
>> 1352
groupid:pivotal
architect:sample
Y
>> tree sample
>> mvn validate
>> mvn compile
>> mvn test
>> mvn package
>> tree sample
>> root@ubuntu:/home/student# mvn --help
Options:
-am,--also-make           If project list is specified, also
                           build projects required by the
                           list
-amd,--also-make-dependents   If project list is specified, also
                           build projects that depend on
                           projects on the list
-B,--batch-mode           Run in non-interactive (batch)
                           mode
-b,--builder <arg>         The id of the build strategy to
                           use.
-C,--strict-checksums      Fail the build if checksums don't
                           match
-c,--lax-checksums         Warn if checksums don't match
-cpu,--check-plugin-updates  Ineffective, only kept for
```

| | |
|--------------------------------------|---|
| | backward compatibility |
| -D,--define <arg> | Define a system property |
| -e,--errors | Produce execution error messages |
| -emp,--encrypt-master-password <arg> | Encrypt master security password |
| -ep,--encrypt-password <arg> | Encrypt server password |
| -f,--file <arg> | Force the use of an alternate POM file (or directory with pom.xml). |
| -fae,--fail-at-end | Only fail the build afterwards; allow all non-impacted builds to continue |
| -ff,--fail-fast | Stop at first failure in reactorized builds |
| -fn,--fail-never | NEVER fail the build, regardless of project result |
| -gs,--global-settings <arg> | Alternate path for the global settings file |
| -gt,--global-toolchains <arg> | Alternate path for the global toolchains file |
| -h,--help | Display help information |
| -l,--log-file <arg> | Log file where all build output will go. |
| -llr,--legacy-local-repository | Use Maven 2 Legacy Local Repository behaviour, ie no use of _remote.repositories. Can also be activated by using -Dmaven.legacyLocalRepo=true |
| -N,--non-recursive | Do not recurse into sub-projects |
| -npr,--no-plugin-registry | Ineffective, only kept for backward compatibility |
| -npu,--no-plugin-updates | Ineffective, only kept for backward compatibility |
| -nsu,--no-snapshot-updates | Suppress SNAPSHOT updates |
| -o,--offline | Work offline |
| -P,--activate-profiles <arg> | Comma-delimited list of profiles |

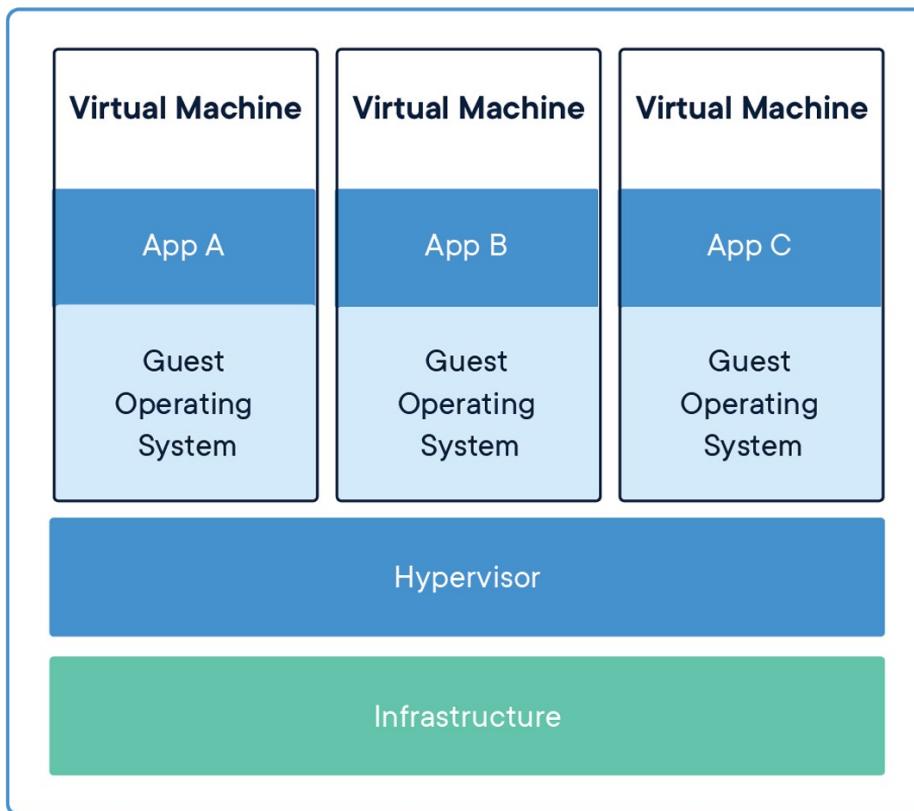
| | |
|-------------------------|--|
| | to activate |
| -pl,--projects <arg> | Comma-delimited list of specified reactor projects to build instead of all projects. A project can be specified by [groupId]:artifactId or by its relative path. |
| -q,--quiet | Quiet output - only show errors |
| -rf,--resume-from <arg> | Resume reactor from specified project |
| -s,--settings <arg> | Alternate path for the user settings file |
| -t,--toolchains <arg> | Alternate path for the user toolchains file |
| -T,--threads <arg> | Thread count, for instance 2.0C where C is core multiplied |
| -U,--update-snapshots | Forces a check for missing releases and updated snapshots on remote repositories |
| -up,--update-plugins | Ineffective, only kept for backward compatibility |
| -v,--version | Display version information |
| -V,--show-version | Display version information WITHOUT stopping build |
| -X,--debug | Produce execution debug output |

Docker :-

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an

application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package. By doing so, thanks to the container, the developer can rest assured that the application will run on any other Linux machine regardless of any customized settings that machine might have that could differ from the machine used for writing and testing the code. In a way, Docker is a bit like a virtual machine. But unlike a virtual machine, rather than creating a whole virtual operating system, Docker allows applications to use the same Linux kernel as the system that they're running on and only requires applications be shipped with things not already running on the host computer. This gives a significant performance boost and reduces the size of the application.

And importantly, Docker is open source. This means that anyone can contribute to Docker and extend it to meet their own needs if they need additional features that aren't available out of the box.



```
>> sudo apt-get update -y  
>> sudo apt-get upgrade -y
```

```
>> add-apt-repository ppa:webupd8team/java
>> apt-get update -y
>> apt-get install oracle-java8-installer
>> java -version
>> sudo apt update
>> sudo apt-key adv --keyserver hkp://ha.pool.skskeyservers.
net:80 --recv-keys
58118E89F3A912897C070ADBF76221572C52609D
>> sudo apt-add-repository "deb
https://apt.dockerproject.org/repo ubuntu-xenial main"
>> sudo apt update
>> sudo apt install docker-engine
>> sudo systemctl start docker
>> docker images
>> docker pull ubuntu
>> root@ubuntu:/home/student# docker --help
```

Options:

```
--config string    Location of client config files (default
                  "/root/.docker")
-D, --debug        Enable debug mode
--help             Print usage
-H, --host list    Daemon socket(s) to connect to
-l, --log-level string Set the logging level
                  ("debug"|"info"|"warn"|"error"|"fatal")
                  (default "info")
--tls              Use TLS; implied by --tlsverify
--tlscacert string Trust certs signed only by this CA (default
                  "/root/.docker/ca.pem")
--tlscert string   Path to TLS certificate file (default
                  "/root/.docker/cert.pem")
--tlskey string    Path to TLS key file (default
                  "/root/.docker/key.pem")
--tlsverify        Use TLS and verify the remote
-v, --version       Print version information and quit
```

Management Commands:

container Manage containers
image Manage images
network Manage networks
node Manage Swarm nodes
plugin Manage plugins
secret Manage Docker secrets
service Manage services
stack Manage Docker stacks
swarm Manage Swarm
system Manage Docker
volume Manage volumes

Commands:

attach Attach local standard input, output, and error streams to a running container
build Build an image from a Dockerfile
commit Create a new image from a container's changes
cp Copy files/folders between a container and the local filesystem
create Create a new container
diff Inspect changes to files or directories on a container's filesystem
events Get real time events from the server
exec Run a command in a running container
export Export a container's filesystem as a tar archive
history Show the history of an image
images List images
import Import the contents from a tarball to create a filesystem image
info Display system-wide information
inspect Return low-level information on Docker objects
kill Kill one or more running containers
load Load an image from a tar archive or STDIN
login Log in to a Docker registry
logout Log out from a Docker registry

logs Fetch the logs of a container
pause Pause all processes within one or more containers
port List port mappings or a specific mapping for the container
ps List containers
pull Pull an image or a repository from a registry
push Push an image or a repository to a registry
rename Rename a container
restart Restart one or more containers
rm Remove one or more containers
rmi Remove one or more images
run Run a command in a new container
save Save one or more images to a tar archive (streamed to STDOUT by default)
search Search the Docker Hub for images
start Start one or more stopped containers
stats Display a live stream of container(s) resource usage statistics
stop Stop one or more running containers
tag Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
top Display the running processes of a container
unpause Unpause all processes within one or more containers
update Update configuration of one or more containers
version Show the Docker version information
wait Block until one or more containers stop, then print their exit codes

To run Images:-

```
>> docker images
>> docker run -ti --rm ubuntu /bin/bash\
>> docker ps
>> docker ps -a
>> docker run -ti ubuntu /bin/bash
>> docker ps
>> docker ps -a
>> docker exec -ti <container id> /bin/bash
>> docker run -ti --name "ubuntu18" --hostname "pivotal"
ubuntu /bin/bash
```

```
>> docker start <container id>
>> docker stop <container id>
>> docker rm <container id>
>> docker image rm <image id>
```

Gitbucket Configuration on Docker:-

Need to maintain gitbucket.war file and Dockerfile in /root Dir.

```
>> vi Dockerfile
From java:latest
MAITAINER student@pivotal.com
LABEL evn=production
ENV apparea /data/app
Run mkdir -p $apparea
ADD ./gitbucket.war $apparea
WORKDIR $apparea
CMD ["java","-jar","gitbucket.war"]
:wq!
>> docker build -t pivotal/git . (to build Dockerfile)
>> docker images
>> docker run -d -p 80:8080 pivotal/git (to port forwarding)
>> ifconfig
Open Firefox and give 192.168.0.151:80 to launch gitbucket server
```

Jenkins Configuration on Docker:-

Need to maintain gitbucket.war file and Dockerfile in /root Dir.

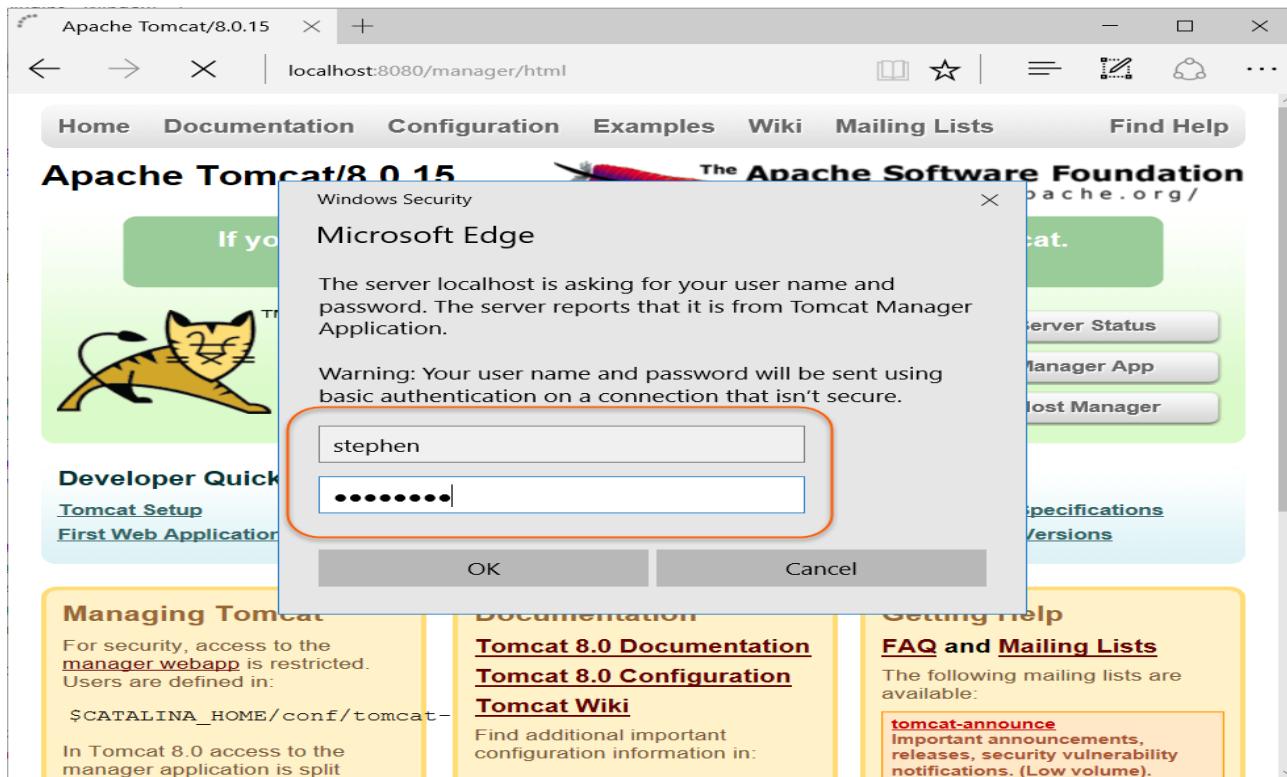
```
>> vi Dockerfile
From java:latest
MAITAINER student@pivotal.com
LABEL evn=production
ENV apparea /data/app
Run mkdir -p $apparea
ADD ./jenkins.war $apparea
```

```
WORKDIR $apparea
CMD ["java","-jar","jenkins.war"]
:wq!
>> docker build -t pivotal/git . (to build Dockerfile)
>> docker images
>> docker run -d -p 80:8080 pivotal/jenkins (to port forwarding)
>> ifconfig
Open Firefox and give 192.168.0.151:80 to launch gitbucket
server
```

Apache tomcat server:-

Download apache-tomcat app from internet

```
#tar -xvf apache-tomcat -C /opt/
#cd /opt/apache-tomcat/bin
#./startup.sh
#firefox &
http://192.168.149.159:8080
set user Path-----
#vi /opt/apache-tomcat/conf/tomcat-users.xml
<role rolename="manager-gui"/>
<user username="student" password="redhat" roles="manager-gui"/>
</tomcat-users>
:wq!
http://192.168.149.159:8080
```



open manager app and deploy .war files
ex: <http://192.168.159.149:8080/sampleweb/>

Install Apache Tomcat 8:-

```
>> apt-get update
>> apt-get install default-jdk
>> groupadd tomcat
>> useradd -s /bin/false -g tomcat -d /opt/tomcat tomcat
>> cd /tmp
>> curl -O http://apache.mirrors.ionfish.org/tomcat/tomcat-
8/v8.5.5/bin/apache-tomcat-8.5.5.tar.gz
>> mkdir /opt/tomcat
>> tar xzvf apache-tomcat-8*tar.gz -C /opt/tomcat --strip-components=1
>> /opt/tomcat
>> chgrp -R tomcat /opt/tomcat
>> chmod -R g+r conf
>> chmod g+x conf
```

```
>> chown -R tomcat webapps/ work/ temp/ logs/
>> update-java-alternatives -l
>> /usr/lib/jvm/java-1.8.0-openjdk-amd64/jre
>> nano /etc/systemd/system/tomcat.service
[Unit]
Description=Apache Tomcat Web Application Container
After=network.target

[Service]
Type=forking

Environment=JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64/jre
Environment=CATALINA_PID=/opt/tomcat/temp/tomcat.pid
Environment=CATALINA_HOME=/opt/tomcat
Environment=CATALINA_BASE=/opt/tomcat
Environment='CATALINA_OPTS=-Xms512M -Xmx1024M -server -XX:
+UseParallelGC'
Environment='JAVA_OPTS=-Djava.awt.headless=true
-Djava.security.egd=file:/dev/./urandom'

ExecStart=/opt/tomcat/bin/startup.sh
ExecStop=/opt/tomcat/bin/shutdown.sh

User=tomcat
Group=tomcat
UMask=0007
RestartSec=10
Restart=always

[Install]
WantedBy=multi-user.target

>> systemctl daemon-reload
>> systemctl start tomcat
```

```

>> systemctl status tomcat
>> ufw allow 8080
>> http://server_domain_or_IP:8080
>> systemctl enable tomcat
>> nano /opt/tomcat/conf/tomcat-users.xml
<tomcat-users . . .>
  <user username="admin" password="password" roles="manager-gui,admin-
gui"/>
</tomcat-users>
>> nano /opt/tomcat/webapps/manager/META-INF/context.xml
>> nano /opt/tomcat/webapps/host-manager/META-INF/context.xml
>> systemctl restart tomcat
>> http://server_domain_or_IP:8080

```

[Home](#) [Documentation](#) [Configuration](#) [Examples](#) [Wiki](#) [Mailing Lists](#)

[Find Help](#)

Apache Tomcat/8.0.33



If you're seeing this, you've successfully installed Tomcat. Congratulations!



Recommended Reading:

[Security Considerations HOW-TO](#)

[Manager Application HOW-TO](#)

[Clustering/Session Replication HOW-TO](#)

[Server Status](#)

[Manager App](#)

[Host Manager](#)

Developer Quick Start

[Tomcat Setup](#)

[First Web Application](#)

[Realms & AAA](#)

[JDBC DataSources](#)

[Examples](#)

[Servlet Specifications](#)

[Tomcat Versions](#)

http://server_domain_or_IP:8080/manager/html

Tomcat Web Application Manager

| Message: OK | | | | | |
|---|-----------------------------------|---------------------------------|------------------------------|----------|--|
| Manager | | | | | |
| List Applications | HTML Manager Help | | Manager Help | | Server Status |
| Applications | | | | | |
| Path | Version | Display Name | Running | Sessions | Commands |
| / | <i>None specified</i> | Welcome to Tomcat | true | 0 | Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes |
| /docs | <i>None specified</i> | Tomcat Documentation | true | 0 | Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes |
| /examples | <i>None specified</i> | Servlet and JSP Examples | true | 0 | Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes |
| /host-manager | <i>None specified</i> | Tomcat Host Manager Application | true | 0 | Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes |
| /manager | <i>None specified</i> | Tomcat Manager Application | true | 1 | Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes |

| | | | | | |
|--|--|--|--|--|--|
| Deploy | | | | | |
| Deploy directory or WAR file located on server | | | | | |
| Context Path (required): <input type="text"/> XML Configuration file URL: <input type="text"/> WAR or Directory URL: <input type="text"/> <input type="button" value="Deploy"/> | | | | | |
| WAR file to deploy | | | | | |
| Select WAR file to upload <input type="button" value="Choose File"/> No file chosen <input type="button" value="Deploy"/> | | | | | |

Uploading Gitbucket and Jenkins:-

>> go to Tomcat manager
 >> click on deploy option
 >> context path /gitbucket
 >> war or Directory URL /opt/gitbucket.war
 >> deploy
 open Gitbucket from Applications

Jenkins :-

>> go to Tomcat manager
 >> click on deploy option
 >> context path /jenkins
 >> war or Directory URL /opt/jenkins.war

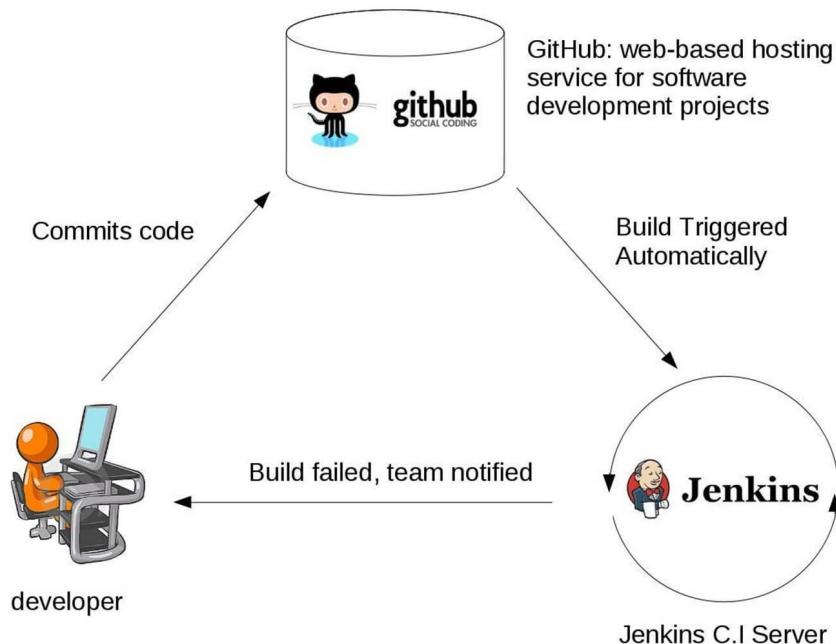
>> deploy
open Gitbucket from Applications

Jenkins:-

Jenkins is a self-contained, open source automation server which can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software.

Jenkins can be installed through native system packages, Docker, or even run standalone by any machine with a Java Runtime Environment (JRE) installed.

In Continuous Integration after a code commit, the software is built and tested immediately. In a large project with many developers, commits are made many times during a day. With each commit code is built and tested. If the test is passed, build is tested for deployment. If deployment is a success, the code is pushed to production. This commit, build, test, and deploy is a continuous process and hence the name continuous integration/deployment.



Jenkins Plugins:-

By default, Jenkins comes with a limited set of features. If you want to integrate your Jenkins installation with version control tools like Git, then you need to

install plugins related to Git. In fact, for integration with tools like Maven you need to install respective plugins in your Jenkins.

Jenkins

Manage Jenkins

- Configure System** Configure global settings and paths.
- Reload Configuration from Disk** Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly.
- Manage Plugins** Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- System Information** Displays various environmental information to assist trouble-shooting.
- System Log** System log captures output from `java.util.logging` output related to Jenkins.
- Load Statistics** Check your resource utilization and see if you need more computers for your builds.
- Jenkins CLI** Access/manage Jenkins from your shell, or from your script.
- Script Console** Executes arbitrary script for administration/trouble-shooting/diagnostics.
- Manage Nodes** Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- Install as Windows Service** Installs Jenkins as a Windows service to this system, so that Jenkins starts automatically when the machine boots.
- Prepare for Shutdown** Stops executing new builds, so that the system can be eventually shut down safely.

Jenkins installation and configuration:-

Configure tomcat server and Maven

- >> Download Jenkins.war and gitbucket.war files
- >> Deploy Jenkins.war and gitbucket.war to Tomcat server
- >> Open jenkins console and gitbucket console through firefox

Jenkins Plug in management:-

- >> Manage Jenkins Manage plugins Available
- >> type your required package name
- >> install without restart.

Compile Maven code:-

- >> Go to Jenkins Dashboard
- >> New item item name
- >> Select Maven project ok

New Item [Jenkins] - Mozilla Firefox
/manager New Item [Jenkins] +
192.168.159.154:8080/jenkins/view/all/newJob
Jenkins > All >
Enter an item name
Maven_compile
» A job already exists with the name 'Maven_compile'
Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.
If you want to create a new item from other existing, you can use this option:
OK Type to autocomplete

Description GitBucket Url Source code management

- >> Gitbucket url
- >> Delete workspace before build starts
- >> Build Pom.xml location goal command<compile>
- >> save.

General GitBucket Source Code Management Build Triggers Build Environment Pre Steps Build Post Steps
Build Settings Post-build Actions
Office Impress Description data compilation
[Plain text] Preview
 Discard old builds
GitBucket
URL http://192.168.159.154:8080/gitbucket/git/student/Mvnproject1.git
 Enable hyperlink to the issue

Source Code Management

None
 Git

Repositories

| | | |
|----------------|---|--------------------------------|
| Repository URL | http://192.168.159.154:8080/gitbucket/git/student/Mvnproject1.git | ? |
| Credentials | - none - | Add |
| | | Advanced... |
| | | Add Repository |

Branches to build

| | | | |
|------------------------------------|----------|----------------------------|-------------------|
| Branch Specifier (blank for 'any') | */master | X | ? |
| | | Add Branch | |

Delete workspace before build starts
[Advanced...](#)

Execute shell script on remote host using ssh
[?](#)

SSH Agent

Pre Steps

[Add pre-build step ▾](#)

Build

Root POM /Mwebapp/sampleweb/pom.xml
Goals and options compile
[Advanced...](#)

Doct States

Test Maven code (CB):-

- >> Go to Jenkins Dashboard
- >> New item
- >> item name

>> Select Maven project >> ok

General GitBucket Source Code Management Build Triggers Build Environment Pre Steps Build Post Steps

Build Settings Post-build Actions

Description testing

[Plain text] [Preview](#)

Discard old builds

GitBucket

URL

Enable hyperlink to the issue

This project is parameterized

Throttle builds

Disable this project

Execute concurrent builds if necessary

[Advanced...](#)

Source Code Management

None

Git

Repositories

Repository URL

Credentials [Add](#)

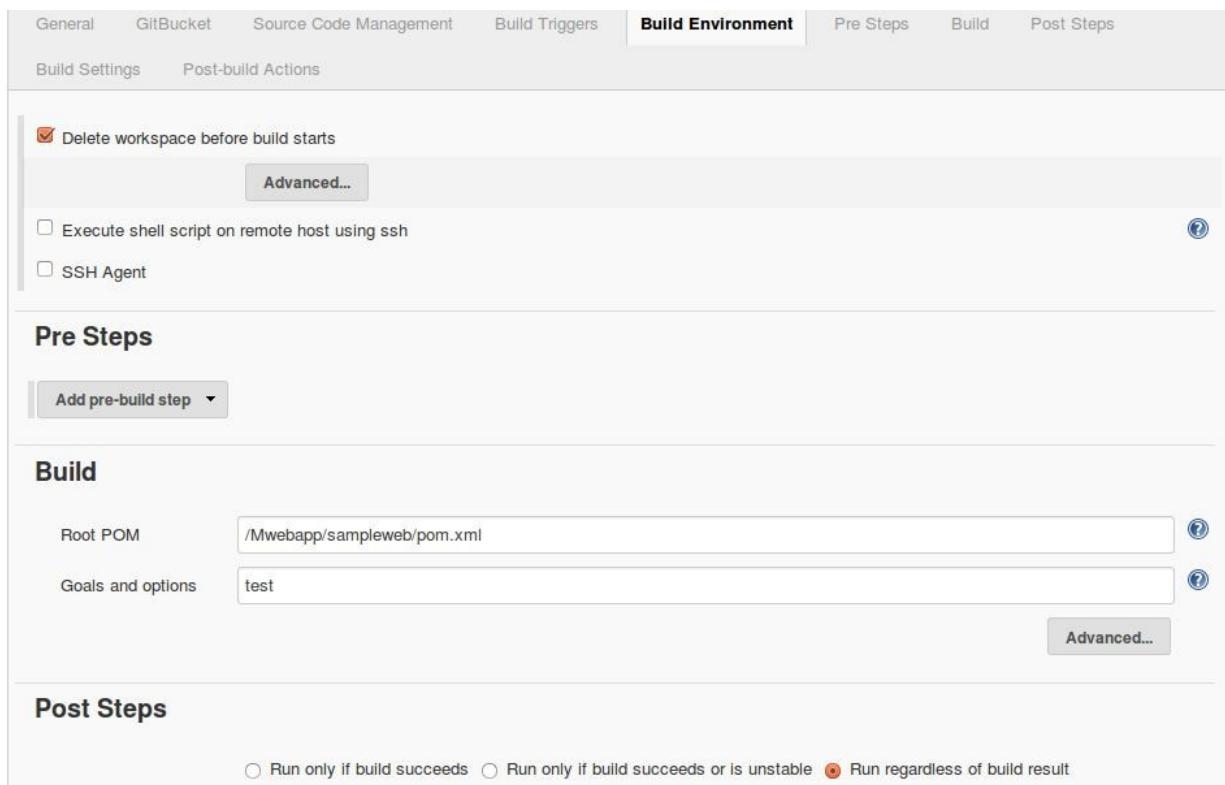
[Advanced...](#)

[Add Repository](#)

Branches to build

Branch Specifier (blank for 'any')

[Add Branch](#)



The screenshot shows the Jenkins 'Build Environment' configuration page for a job. The top navigation bar includes tabs for General, GitBucket, Source Code Management, Build Triggers, Build Environment (which is selected), Pre Steps, Build, and Post Steps. Under 'Build Environment', there are sections for 'Build Settings' and 'Post-build Actions'. In 'Build Settings', the 'Delete workspace before build starts' checkbox is checked. Below it are 'Advanced...' and 'Execute shell script on remote host using ssh' (unchecked) buttons. There is also an 'SSH Agent' section. The 'Pre Steps' section contains a 'Add pre-build step' dropdown. The 'Build' section includes 'Root POM' set to '/Mwebapp/sampleweb/pom.xml' and 'Goals and options' set to 'test'. An 'Advanced...' button is located next to the goals field. The 'Post Steps' section has three radio buttons: 'Run only if build succeeds', 'Run only if build succeeds or is unstable', and 'Run regardless of build result', with the third option selected.

Integrate Maven code in Jenkins:-

- >> Go to Jenkins Dashboard
- >> New item item name
- >> Select Maven project ok
- >> Comment
- >> git url
- >> Build whenever a SNAPSHOT dependency is built
- >> Delete workspace before build starts
- >> Set root pom path
- >> set branches path
- >> apply ok
- >> click on build icon

General GitBucket Source Code Management Build Triggers Build Environment Pre Steps Build Post Steps

Build Settings Post-build Actions

Description Integration

[Plain text] [Preview](#)

Discard old builds [?](#)

GitBucket

URL <http://192.168.159.154:8080/gitbucket/git/student/Mvnproject1.git>

Enable hyperlink to the issue

This project is parameterized [?](#)

Throttle builds [?](#)

Disable this project [?](#)

Execute concurrent builds if necessary [?](#)

[Advanced...](#)

Source Code Management

None

Git

Repositories

Repository URL <http://192.168.159.154:8080/gitbucket/git/student/Mvnproject1.git> [?](#)

Credentials student/***** [Add](#) [Advanced...](#)

Add Repository

Branches to build

Branch Specifier (blank for 'any') */branch1 [X](#) [?](#)

Add Branch

Repository browser (Auto) [?](#)

Additional Behaviours [Add](#) [?](#)

Subversion

Source Code Management

None

Git

Repositories

Repository URL: ?

Credentials: Add ?

Advanced... Add Repository

Branches to build

Branch Specifier (blank for 'any'): X ?

Add Branch

Repository browser

(Auto) ?

Additional Behaviours

Add ▼

Subversion ?

General GitBucket Source Code Management **Build Triggers** Build Environment Pre Steps Build Post Steps

Build Settings Post-build Actions

Subversion ?

Build Triggers

Build whenever a SNAPSHOT dependency is built ?

Schedule build when some upstream has no successful builds ?

Trigger builds remotely (e.g., from scripts) ?

Build after other projects are built ?

Build periodically ?

Build when a change is pushed to GitBucket ?

Poll SCM ?

Build Environment

Delete workspace before build starts

Advanced...

Execute shell script on remote host using ssh ?

SSH Agent ?

With Ant ?

Build Settings Post-build Actions

Add pre-build step ▾

Build

Root POM /Mwebapp/sampleweb/pom.xml ?

Goals and options ?

on Advanced...

Post Steps

Run only if build succeeds Run only if build succeeds or is unstable Run regardless of build result
Should the post-build steps run only for successful builds, etc.

Add post-build step ▾

Build Settings

E-mail Notification

Post-build Actions

General GitBucket Source Code Management Build Triggers Build Environment Pre Steps Build Post Steps

Build Settings Post-build Actions X

E-mail Notification

Post-build Actions

Git Publisher

Push Only If Build Succeeds ?

Merge Results ?

If pre-build merging is configured, push the result back to the origin

Force Push ?

Add force option to git push

Tags Add Tag ?

Tags to push to remote repositories

Branches X ?

Branch to push master ?

Target remote name origin ?

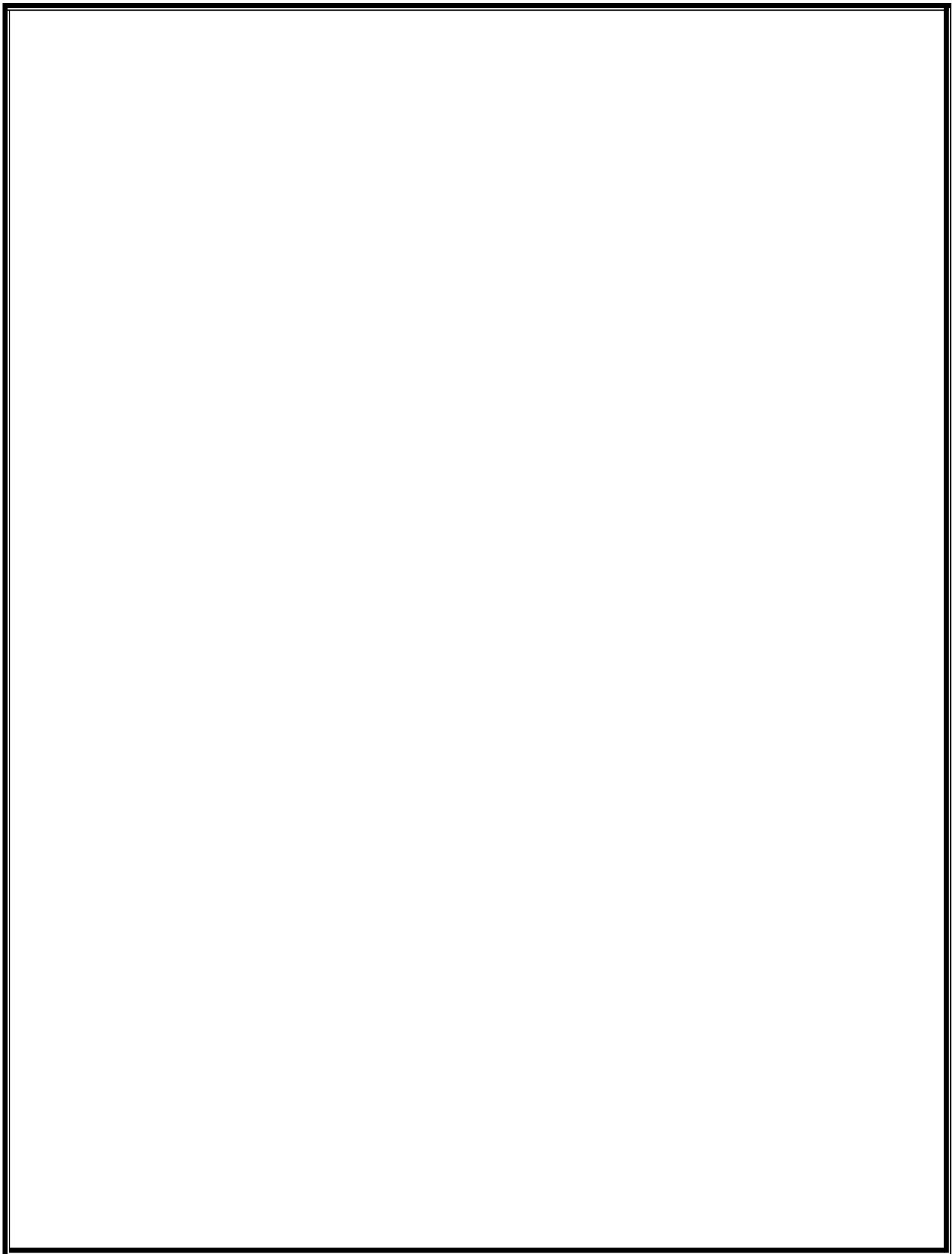
Add Branch

Branches to push to remote repositories

Package Maven code (CB):-

>> Go to Jenkins Dashboard
>> New item >> item name
>> Select Maven project >> ok
>> Description
>> git url
>> Build whenever a SNAPSHOT dependency is built
>> Delete workspace before build starts
>> Set root pom path
>> Goal package
>> set branches >> path >> apply >> save >> build now.

The screenshot shows the Jenkins 'General' configuration page for a new item. The 'Description' field contains the word 'package'. The 'GitBucket' section is expanded, showing the 'Source Code Management' tab selected. Under 'Source Code Management', the 'Git' option is selected. A repository is configured with the URL `http://192.168.159.154:8080/gitbucket/git/student/Mvnproject1.git` and credentials 'student/*****'. The 'Branches to build' section shows a branch specifier `*/branch1`. The 'Repository browser' is set to '(Auto)'. At the bottom, there are 'Additional Behaviours' and a 'Subversion' checkbox.



The screenshot shows the Jenkins job configuration interface. It includes sections for Pre Steps, Build, Post Steps, and Build Settings. In the Pre Steps section, there is a link to 'Add pre-build step'. The Build section contains fields for Root POM (set to '/root/.jenkins/workspace/Maven_package/pom.xml') and Goals and options (set to 'package'). There is also an 'Advanced...' button. The Post Steps section includes radio buttons for 'Run only if build succeeds', 'Run only if build succeeds or is unstable', and 'Run regardless of build result', with the third option selected. A note below says 'Should the post-build steps run only for successful builds, etc.' The Build Settings section has a checkbox for 'E-mail Notification'.

Automation with Pipeline View:-

1st step

>> Go to Maven_compile configuration
>> Build triggers
>> Build after other projects are built
>> Maven_integration
>> Apply >> save.

2nd step

>> Go to Maven_test configuration
>> Build triggers
>> Build after other projects are built
>> Maven_compile
>> Apply >> save.

3rd step

>> Go to Maven_Package configuration
>> Build triggers
>> Build after other projects are built
>> Maven_test

>> Apply >> save

The screenshot shows a configuration interface for a build job. At the top, there is a header with tabs like 'General', 'Source Code Management', 'Build Triggers', 'Build Environment', 'Test', 'Deployment', and 'Script'. Below the header, the 'Build Triggers' section is active.

Build Triggers:

- Build whenever a SNAPSHOT dependency is built
 - Schedule build when some upstream has no successful builds
- Trigger builds remotely (e.g., from scripts)
- Build after other projects are built
 - Trigger only if build is stable
 - Trigger even if the build is unstable
 - Trigger even if the build fails
- Build periodically
- Build when a change is pushed to GitBucket
- Poll SCM

Build Environment:

- Delete workspace before build starts
 - Execute shell script on remote host using ssh

At the bottom, there are two buttons: **Save** (dark blue) and **Apply** (light blue).

Additional Behaviours | Add ▾

Subversion

Build Triggers

Build whenever a SNAPSHOT dependency is built

Schedule build when some upstream has no successful builds

Trigger builds remotely (e.g., from scripts)

Build after other projects are built

Projects to watch: Maven_compile

Trigger only if build is stable

Trigger even if the build is unstable

Trigger even if the build fails

Build periodically

Build when a change is pushed to GitBucket

Poll SCM

Build Environment

Delete workspace before build starts

Execute shell script on remote host using ssh

SSH Agent

Save **Apply**

Build Triggers

Build whenever a SNAPSHOT dependency is built

Schedule build when some upstream has no successful builds

Trigger builds remotely (e.g., from scripts)

Build after other projects are built

Projects to watch: Maven_test

Trigger only if build is stable

Trigger even if the build is unstable

Trigger even if the build fails

Build periodically

Build when a change is pushed to GitBucket

Poll SCM

Build Environment

Delete workspace before build starts

Execute shell script on remote host using ssh

SSH Agent

With Ant

Pre Steps

Save **Apply**

Pipeline Installation:-

1st Step

>> Manage Jenkins
>> Manage plugins
>> Available <type build pipeline package name>
>> install without restart.

2nd step

>> Jenkins Dashboard
>> New view
>> Name
>> build pipeline view
>> apply >> save.

3rd Step

>> View configure Upstream / downstream config
>> Select >> Maven_integration >> Apply >> ok.

The screenshot shows the Jenkins 'New View' configuration dialog. At the top, there is a 'View name' input field containing 'newproject'. Below it, there are three radio button options: 'Build Pipeline View' (selected), 'List View', and 'My View'. The 'Build Pipeline View' option is described as showing jobs in a build pipeline view. The 'List View' option is described as showing items in a simple list format. The 'My View' option is described as automatically displaying all jobs the user has access to. At the bottom right of the dialog is an 'OK' button. The main Jenkins interface is visible in the background, showing sections for 'Build Queue' (empty) and 'Build Executor Status' (1 Idle, 2 Idle).

Jenkins > Project1 >

Build History

Edit View

Delete View

Project Relationship

Check File Fingerprint

Manage Jenkins

My Views

Credentials

New View

Build Queue
No builds in the queue.

Build Executor Status
1 Idle
2 Idle

Pipeline Flow

Filter build queue [Plain text] [Preview](#)

Filter build executors

Build Pipeline View Title

Layout

Based on upstream/downstream relationship

This layout mode derives the pipeline structure based on the upstream/downstream trigger relationship between jobs. This is the only out-of-the-box supported layout mode, but is open for extension.

Upstream / downstream config

Select Initial Job [?](#)

Trigger Options

Build Cards [?](#)

Use the default build cards

Restrict triggers to most recent successful builds Yes No [?](#)

Always allow manual trigger on pipeline steps Yes No [?](#)

Jenkins 3 search student | log out

Jenkins > Project1 >

ENABLE AUTO REFRESH

Build Pipeline

Run History Configure Add Step Delete Manage

Pipeline #10 Maven_Integration
Ubuntu Software Center, 2019 10:09:23 AM
student 20 sec

#8 Maven_compile
13 Apr, 2019 10:09:48 AM 15 sec

#8 Maven_test
13 Apr, 2019 10:10:13 AM 9.4 sec

#13 Maven_package
13 Apr, 2019 10:10:28 AM 13 sec

Fully Automation in Jenkins:-

1st Step

>> Go to Jenkins user configure

>> Add new Token and copy

>> Apply Save.

2nd step

>> Go to Gitbucket

>> Account Settings

>> Service hooks Add payroll url(Jenkins url)

>> Past the Token >> tick on Push >> save.

3rd step

>> Go to Jenkins

>> Maven_integaration

>> Configure

>> Build Triggers

>> Build when a change is pushed to Gitbucket >> apply >> save.

4th step

>> Go to Terminal Push new code to Gitbucket server.

The screenshot shows the Jenkins dashboard with the following details:

- Left Sidebar:** Includes links for New Item, LibreOffice Calc (selected), Build History, Project Relationship, Check File Fingerprint, Manage Jenkins, My Views, Credentials, and New View.
- Central Dashboard:** A table listing four Jenkins projects:

| S | W | Name | Last Success | Last Failure | Last Dur |
|---|----|-------------------|-------------------|------------------|----------|
| ● | ☀️ | Maven_compile | 2 hr 36 min - #10 | N/A | 12 sec |
| ● | ☀️ | Maven_integration | 2 hr 36 min - #11 | 3 days 3 hr - #3 | 21 sec |
| ● | ☀️ | Maven_package | 2 hr 35 min - #16 | 20 hr - #10 | 14 sec |
| ● | ☀️ | Maven_test | 2 hr 35 min - #10 | N/A | 12 sec |
- Bottom Navigation:** Includes links for Legend, RSS for all, RSS for failures, and RSS for just latest builds.

Jenkins

student

4 search student | log out

People

LibreOffice Calc

Builds

Configure

My Views

Credentials

Full Name: student

Description:

API Token

Current token(s): Token created on 2019-04-13T10:13:55.463+05:30
Created 0 day(s) ago ⚠ Never used

Add new Token

Credentials

Credentials are only available to the user they belong to

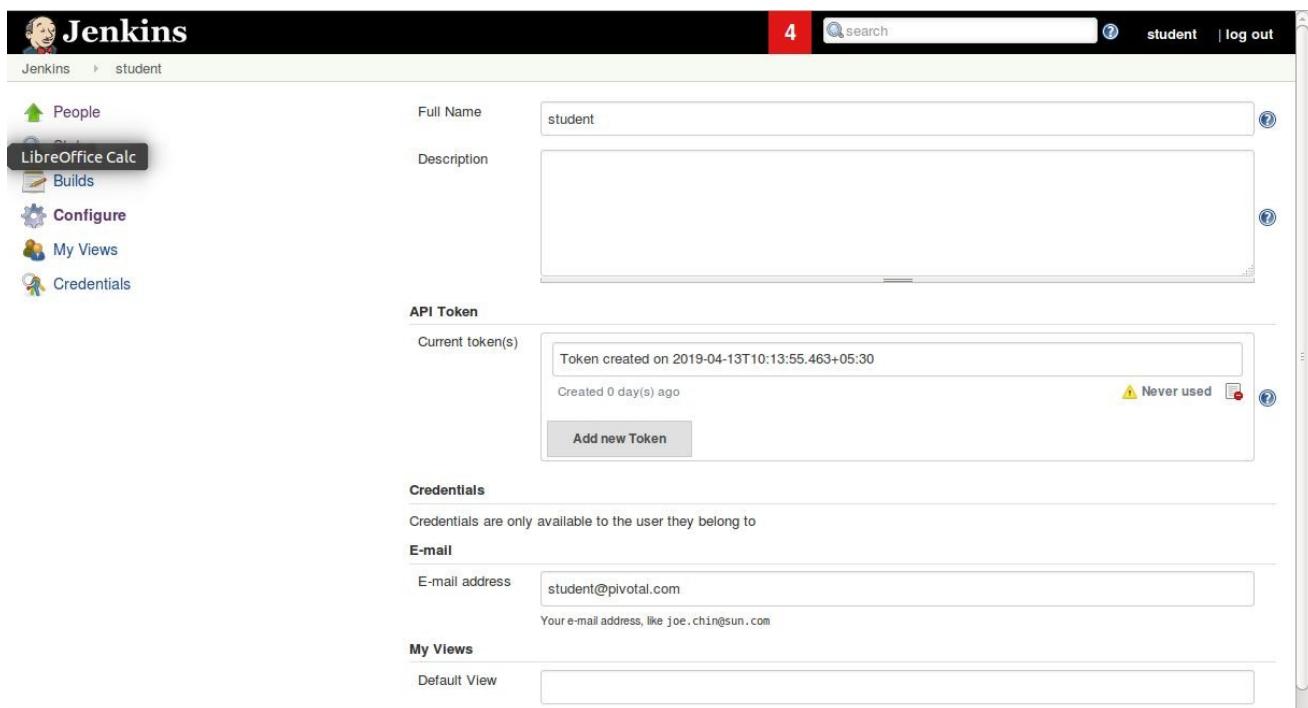
E-mail

E-mail address: student@pivotal.com

Your e-mail address, like joe.chin@sun.com

My Views

Default View:



Profile

Applications

Service Hooks

Notifications

Webhook / Manage webhook

Payload URL: http://192.168.159.154:8080/jenkins/

Test Hook

Content type: application/x-www-form-urlencoded

Security Token: Token created on 2019-04-13T10:13:55.463+05:30

Which events would you like to trigger this webhook?

Create
Branch, or tag created.

Gollum
Wiki page updated.

Issue comment
Issue commented on.

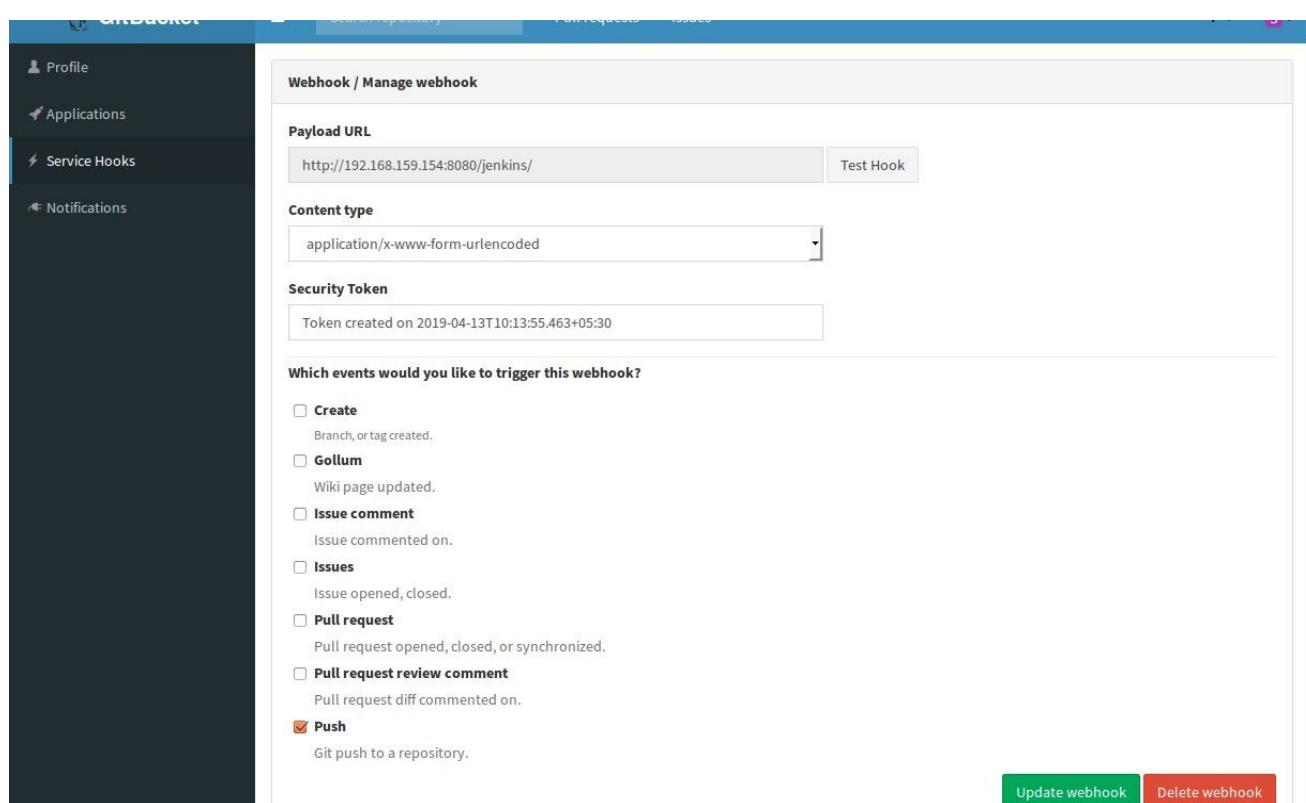
Issues
Issue opened, closed.

Pull request
Pull request opened, closed, or synchronized.

Pull request review comment
Pull request diff commented on.

Push
Git push to a repository.

Update webhook Delete webhook



The screenshot shows the Jenkins project configuration for 'Maven_integration'. The 'Build Triggers' tab is selected. Under 'Additional Behaviors', 'Subversion' is listed. The 'Build Triggers' section contains several options: 'Build whenever a SNAPSHOT dependency is built' (checked), 'Schedule build when some upstream has no successful builds' (unchecked), 'Trigger builds remotely (e.g., from scripts)' (unchecked), 'Build after other projects are built' (unchecked), 'Build periodically' (unchecked), 'Build when a change is pushed to GitBucket' (checked), 'Pass-through Git commit' (checked), and 'Poll SCM' (unchecked). The 'Build Environment' section includes the option 'Delete workspace before build starts' (checked). An 'Advanced...' button is visible at the bottom.

Continues Deploy:-

1st step

- >> Go to Jenkins Dashboard
- >> Manage Jenkins
- >> Manage Plugins >> Available >> Deploy to container
- >> install without restart >> ok.

2nd step

- >> Go to Jenkins Dashboard
- >> Maven_Package Configure
- >> Post-build Actions
- >> Deploy war/ear to container >> war/ear files=**/*.war
- >> Context path=/sampleweb >> credentials
- >> Tomcat Url Apply Save.

3rd Step

- >> Go to Terminal push some new code to Gitbucket than see the changes in firefox.

Jenkins Backup and Restore:-

The screenshot shows the Jenkins Plugin Manager interface. It lists several installed plugins with their versions and uninstallation options:

- Command Agent Launcher Plugin**: Version 1.3, Uninstall
- Conditional BuildStep**: Version 1.3.6, Uninstall
- Credentials Plugin**: Version 2.1.18, Uninstall
- Deploy to container Plugin**: Version 1.13, Uninstall
- Display URL API**: Version 2.3.1, Uninstall
- Durable Task Plugin**: Version 1.29, Uninstall
- External Monitor Job Type Plugin**: Version 1.7, Uninstall
- Folders Plugin**: Version 6.8, Uninstall
- Git client**: Version 2.7.6, Downgrade to 2.7.6, Uninstall

1st step

>> Go to Jenkins Dashboard

The screenshot shows the Jenkins job configuration interface under the "Post-build Actions" section. A "Deploy war/ear to a container" action is selected, with the following settings:

- WAR/EAR files: `**/*.war`
- Context path: `/sampleweb`
- Containers: **Tomcat 7.x**
 - Credentials: `deploy/*****`
 - Tomcat URL: `http://192.168.159.154:8080/`

Other options shown include "Add Container" and "Deploy on failure". At the bottom, there are "Save" and "Apply" buttons.

>> Manage Jenkins
>> Manage Plugins >> Available
>> Backup Plugin >> install without restart >> ok.
2nd step
>> Go to Jenkins Dashboard
>> Manage Jenkins
>> Backup manager
>> Setup >> Backup directory >> Format >> save.
3rd step
>> Go to Jenkins Dashboard
>> Manage Jenkins
>> Backup manager
>> Backup Hudson configuration >> Ok.
4th step
>> Go to Jenkins Dashboard
>> Manage Jenkins
>> Backup manager
>> Restore Hudson configuration >> Launch Restore.

Ansible:-

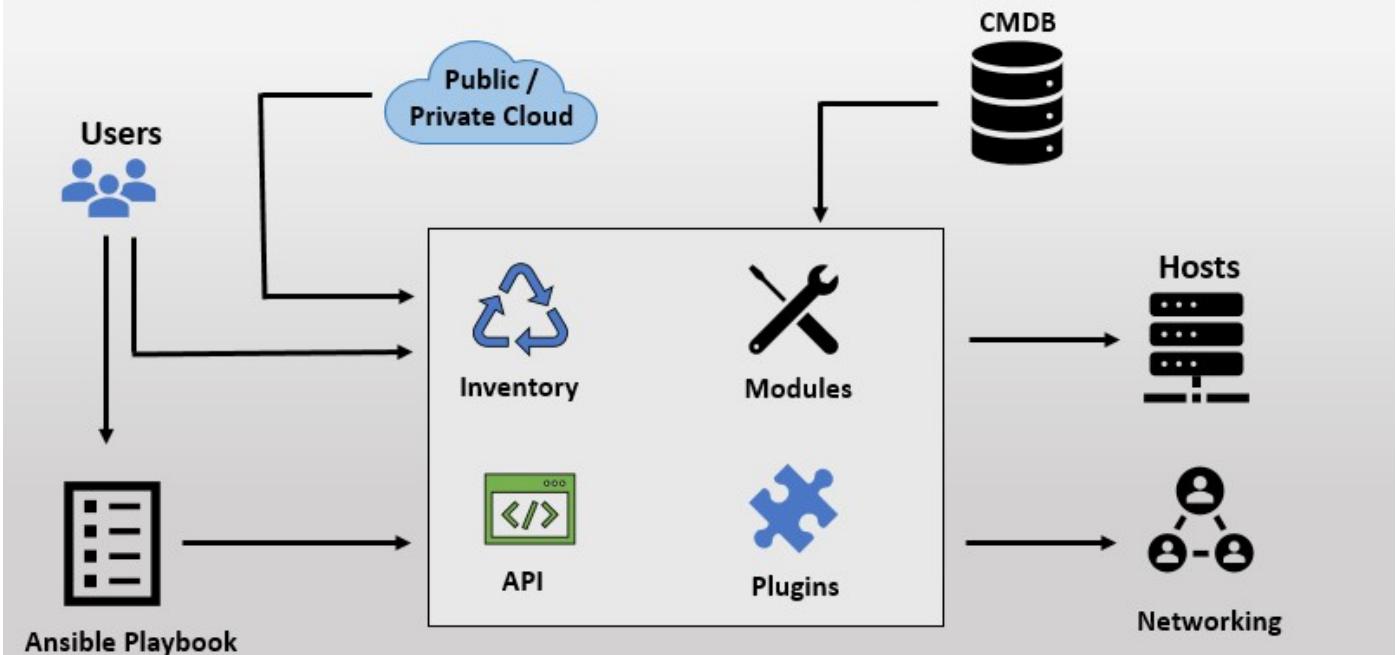
Ansible is a radically simple IT automation system. It handles configuration

The screenshot shows the Jenkins Backup manager configuration page. On the left, there's a sidebar with various Jenkins management links like New Item, People, Build History, etc. The main area is titled "Backup config files". It has two main sections: "Backup configuration" and "Backup content". In "Backup configuration", the Hudson root directory is set to "/root/.jenkins" and the backup directory is set to "/home/student/Desktop". The format is set to "zip", and the file name template is "backup_@date@. @extension@". There are also options for "Verbose mode", "Configuration files (.xml) only", and "No shutdown". In "Backup content", there are checkboxes for "Backup job workspace" (unchecked), "Backup builds history" (checked), "Backup maven artifacts archives" (checked), and "Backup fingerprints" (unchecked). A "Save" button is at the bottom.

The screenshot shows the Jenkins Backup manager restore page. The sidebar is identical to the configuration page. The main area is titled "Backup manager" and shows "Available backup in /home/student/Desktop :". It lists several backup files: VMwareTools-10.0.0-2977863.tar.gz (selected), vmware-tools-distrib, backup_20190413_1441.zip, backup_20190412_1828.zip, Maven_compile1.png, and jenkins.war. A "Launch restore" button is at the bottom. Below the sidebar, there are sections for "Build Queue" (empty) and "Build Executor Status" (2 Idle).

management, application deployment, cloud provisioning, ad-hoc task execution, network automation, and multi-node orchestration. Ansible makes complex changes like zero-downtime rolling updates with load balancers easy.

Ansible Architecture



- Minimal in nature. Management systems should not impose additional dependencies on the environment.[16]
- Consistent. With Ansible one should be able to create consistent environments.
- Secure. Ansible does not deploy agents to nodes.
Only OpenSSH and Python are required on the managed nodes.[16][12]
- Highly reliable. When carefully written, an Ansible playbook can be idempotent, to prevent unexpected side-effects on the managed systems. [18] It is entirely possible to have a poorly written playbook that is not idempotent.
- Minimal learning required. Playbooks use an easy and descriptive language based on YAML and Jinja templates.
- Control machines have to be a Linux/Unix host (for example, Red Hat Enterprise Linux, Debian, CentOS, macOS, BSD, Ubuntu[11]), and Python 2.7 or 3.5 is required.[3]

Ansible Installation:-

Configure in System 1, System2 and System3 :-

```
>> set ip address and hostaddress  
>> install ssh  
>> install epel-release packages  
>> install yum packages  
>> sudo yum localinstall --nogpgcheck  
https://download1.rpmfusion.org/free/el/rpmfusion-free-release-7.noarch.rpm  
>> sudo yum localinstall --nogpgcheck  
https://download1.rpmfusion.org/nonfree/el/rpmfusion-nonfree-release-7.noarch.rpm  
>> sudo yum localinstall --nogpgcheck  
http://dl.fedoraproject.org/pub/epel/7/x86\_64/Packages/e/epel-release-7-11.noarch.rpm  
>> sudo yum localinstall --nogpgcheck  
http://rpms.famillecollet.com/enterprise/remi-release-7.rpm  
>> sudo rpm --import https://www.elrepo.org/RPM-GPG-KEY-elrepo.org  
sudo rpm -Uvh http://www.elrepo.org/elrepo-release-7.0-3.el7.elrepo.noarch.rpm  
>> sudo yum localinstall --nogpgcheck  
http://repo.websatic.com/yum/el7/websatic-release.rpm  
>> yum update  
>> yum clean all  
>> yum install ansible  
ssh-key has to setup on both the nodes
```

Ansible server talks to managed nodes using ssh

Default location of inventory: /etc/ansible/hosts

add hosts to /etc/ansible/hosts

and configure password less authentication

Generate ssh keys and setup password less authentication between server and clients

perform jobs either using ansible command line or playbooks.

Ansible command line:-

```
>> ansible all -m ping  
>> ansible all -a "touch /tmp/hello"  
>> ansible webservers -m ping
```

Ansible playbooks:-

playbook for file copying

```
---
```

```
- hosts: all  
  become_user: root  
  
  tasks:  
    - name: Copy file with owner and permissions  
      copy:  
        src: /root/playfile  
        dest: /tmp  
        owner: root  
        group: root  
        mode: '0644'
```

```
>> ansible-playbook apache.yml --check  
>> ansible-playbook filename
```

Web playbook:-

```
- hosts: all  
  become_user: root
```

```
  tasks:
```

```
- name: 1. Install Latest Version of HTTP/Apache
  yum: name=httpd state=present

- name: 2. start httpd service
  service: name=httpd state=started enabled=yes

- name: 3. copy the standard index.html file
  copy: src=/tmp/index.html dest=/var/www/html/index.html mode=0664

- name: 4. Add apache iptable rule
  command: /sbin/iptables -I INPUT 1 -p tcp --dport http -j ACCEPT -m comment
  --commnet "Apache on port 80"

- name: 5. Save iptable rule
  command: iptables-save

>> ansible-playbook apache.yml --check
>> ansible-playbook filename
```

users playbook:-

```
---
```

```
- hosts: all
  become_user: root
```

```
tasks:
```

```
# this task creates groups
- name: add a group
  group:
    name={{ item }}
```

```
state=present
with_items:
- demogrp
- demogrp1
tags: add_new_grp
# this task creates users
- name: add a user
  user:
    name={{ item }}
    state=present
    password="redhat"
    shell=/bin/bash
  with_items:
    - demouser1
    - demouser2
    - demouser3
  tags: add_new_user
# this tasks is to delete the users
- name: delete several users
  user:
    name={{ item }}
    state=absent
  with_items:
    - demouser1
  tags: remove_user
# this task is to delete the groups
- name: delete groups
  group:
    name={{ item }}
    state=absent
  with_items:
    - demogrp
    - demogrp1
  tags: remove_group
```

```
>> ansible-playbook apache.yml --check
>> ansible-playbook user.yml --list-tags
>> ansible-playbook user.yml --tags add_net_user

- name: Patch Windows systems against Meltdown and Spectre
  hosts: "{{ target_hosts | default('all') }}"

vars:
  reboot_after_update: no
  registry_keys:
    - path: HKLM:\SYSTEM\CurrentControlSet\Control\Session
      Manager\Memory Management
      name: FeatureSettingsOverride
      data: 0
      type: dword

    - path: HKLM:\SYSTEM\CurrentControlSet\Control\Session
      Manager\Memory Management
      name: FeatureSettingsOverrideMask
      data: 3
      type: dword

  # https://support.microsoft.com/en-us/help/4072699
  - path:
      HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\QualityCompat
      name: cadca5fe-87d3-4b96-b7fb-a231484277cc
      type: dword
      data: '0x00000000'

tasks:
  - name: Install security updates
    win_updates:
      category_names:
```

```
- SecurityUpdates  
notify: reboot windows system
```

```
- name: Enable kernel protections  
win_regedit:  
  path: "{{ item.path }}"  
  name: "{{ item.name }}"  
  data: "{{ item.data }}"  
  type: "{{ item.type }}"  
  with_items: "{{ registry_keys }}"
```

handlers:

```
- name: reboot windows system  
  win_reboot:  
    shutdown_timeout: 3600  
    reboot_timeout: 3600  
    when: reboot_after_update  
>> ansible-playbook apache.yml --check  
>> ansible-playbook user.yml --list-tags  
>> ansible-playbook user.yml --tags add_net_user
```