Git COMMANDS

```
git config --global user.name "username"
git config --global user.email "email@example.com"
git clone
git --help
git --version
git add filename
git add .
git commit -m "any message"
git push
git pull
git log
git status
git init
git remote add anyname url_of_empty_repository
git push anyname master
git branch branchname
git checkout branchname
git merge branchname1 branchname2
git rebase branchname1 branchname2
git branch -d branchname1 //delete the merged branch
git diff commitid1 commitid2
git revert commitid
git reset commitid
git reset --soft commitid
git reset --hard commitid
```

ANT COMMANDS

```
ant clean
ant init
ant compile
ant war
ant
```

MAVEN COMMANDS

```
mvn clean
mvn compile
mvn verify
mvn install
mvn clean install
```

```
mvn test
mvn sonar:sonar
mvn clean install sonar:sonar
```

GRADLE COMMANDS

```
gradle clean
gradle assemble
gradle build
gradle help
```

Jenkins pipeline

```
node  {

   stage ("scm")
   {
   git 'https://github.com/ghanigreen/maven_demo.git'
   }
   stage ("archive")
   {
   archiveArtifacts '**/*.war'
   }
   stage ("build")
   {
   bat 'mvn clean install'
   }
   stage("junit")
   {
   junit healthScaleFactor: 10.0, testResults: '**/gameoflife-web/target/surefire-reports/*.xml'
   }
   stage("sonarqube")
   {
   bat 'mvn sonar:sonar'
   }
   stage("deploy")
   {
   sh 'cp -R "E:\\workspace\\newpipeline\\gameoflife-web\\target\\gameoflife.war" "C:\\Program
Files\\Apache Software Foundation\\Tomcat 9.0\\webapps"'
   }

}
```

VAGRANTUP COMMANDS

vagrant box
vagrant init
vagrant init ubuntu/trusty64
vagrant up
vagrant halt
vagrant ssh
vagrant destroy
vagrant resume
vagrant suspend

LINUX BASIC COMMANDS

pwd
ls
ls -la
echo "content of file > name.txt
echo "additional content" >>name.txt
touch name1.txt
chmod 755 name1.txt
vi name1.txt

VI Editor commands

i - insert mode
ESC + :wq - to write and quit
Esc + :Q! - quit without save
Esc + :Q - quit the editor

PUPPET COMMANDS

```
        //vagrant file

        Vagrant.configure("2") do |config|
   config.vm.define "puppet" do |puppet|
   puppet.vm.box = "ubuntu/xenial64"
   puppet.vm.network "private_network", ip: "192.168.0.175"
   puppet.vm.hostname = "puppet"
        puppet.vm.provider "virtualbox" do |v|
   v.memory = 4096
```

```
  end
      end
  config.vm.define "node1" do |node1|
  node1.vm.box = "ubuntu/xenial64"
  node1.vm.network "private_network", ip: "192.168.0.176"
  node1.vm.hostname = "node1"
      end
      end
```

```
      //puppet server

      $sudo vi /etc/hosts
      192.168.0.176  node1
      sudo  curl -O https://apt.puppetlabs.com/puppetlabs-release-pc1-xenial.deb
      sudo dpkg -i puppetlabs-release-pc1-xenial.deb
      sudo apt-get update
      sudo apt-get install puppetserver
      sudo ufw allow 8140
      sudo vi /etc/default/puppetserver
      sudo systemctl start puppetserver
      sudo systemctl status puppetserver
```

```
      //node

      sudo vi /etc/hosts
      192.168.0.175 puppet
      sudo wget https://apt.puppetlabs.com/puppetlabs-release-pc1-xenial.deb
      sudo dpkg -i puppetlabs-release-pc1-xenial.deb
      sudo apt-get update
      sudo apt-get install puppet-agent
      sudo systemctl start puppet
      sudo systemctl enable puppet
```

```
      // puppet server

      sudo /opt/puppetlabs/bin/puppet cert list
      sudo /opt/puppetlabs/bin/puppet cert list --all //to view all certificate list
      sudo /opt/puppetlabs/bin/puppet cert sign node1.domain.name
      or
      sudo /opt/puppetlabs/bin/puppet cert sign –all
      sudo vi /etc/puppetlabs/code/environments/production/manifests/site.pp
```

```
//paste below commands in site.pp file

file {'/tmp/tempfile':                    # resource type file and filename
ensure  => present,                         # make sure it exists
mode    => '0644',                        # file permissions
content => "This is temporary file",
}

or


package {'screen':
ensure => present,
}
package {'git':
ensure => present,
}
package {'wget':
ensure => present,
}
package {'python':
ensure => present,
}
package {'apache2':
ensure => present,
}
package {'ant':
ensure => present,
}

//node
sudo /opt/puppetlabs/bin/puppet agent --test

//to verify all the software installed on node use below command
apt list --installed
```

CHEF COMMANDS

First Step

Install Vagrant and virtual box

Create vagrant file

Second step

vagrant init ubuntu/trusty64
Replace with below commands in vagrant file

```
Vagrant.configure("2") do |config|
config.vm.define "developmentkit" do |developmentkit|
developmentkit.vm.box = "ubuntu/trusty64"
developmentkit.vm.network "private_network", ip: "192.168.0.252"
developmentkit.vm.hostname = "developmentkit.example.com"
end
config.vm.define "chefserver" do |chefserver|
chefserver.vm.box = "ubuntu/trusty64"
chefserver.vm.network "private_network", ip: "192.168.0.253"
chefserver.vm.hostname = "chefserver.example.com"
chefserver.vm.provider "virtualbox" do |v|
v.memory = 4096
v.cpus = 2
end
end
config.vm.define "node" do |node|
node.vm.box = "ubuntu/trusty64"
node.vm.network "private_network", ip: "192.168.0.3"
node.vm.hostname = "node.example.com"
end
end
```

//up the machine using below commands
vagrant up

Third Step
paste the chef developmentkit and chef server of ubuntu in vagrant folder

---

Fourth Step

```
vagrant ssh developmentkit
ls  /vagrant
sudo dpkg -i /vagrant/chefdk_0.9.0-1_amd64.deb
mkdir cookbooks
mkdir .chef
vi .chef/knife.rb
```

cookbook_path ['/home/vagrant/cookbooks'] //In knife.rb
cd cookbooks
chef generate cookbook my_cookbook
ls my_cookbook/
cd my_cookbook/recipes
vi default.rb

// In default.rb paste the below commands
file '/tmp/hello.txt' do
content 'hello world'
end

or

package 'nginx' do
action :install
end

service 'nginx' do
action [:enable, :start]
end

sudo chef-client -z --runlist 'recipe[my_cookbook]' //to verify my_cookbook is working or not
exit

Fifth step

// go to root user of developmentkit
sudo -s
ls /etc
vi hosts

//add these three lines in hosts file
192.168.0.253 chefserver.example.com chefserver
192.168.0.252 developmentkit.example.com developmentkit
192.168.0.3 node.example.com node

//save it by using escape :wq
exit

Sixth Step
//In chefserver machine
vagrant ssh chefserver
sudo -s
ls /etc
vi hosts

//add these three lines in hosts file

192.168.0.253 chefserver.example.com chefserver
192.168.0.252 developmentkit.example.com developmentkit
192.168.0.3 node.example.com node
//save it by using escape :wq
ping node // to verify node is connected to server - stop using ctrl+c
exit

Seventh Step //In node machine
vagrant ssh node
sudo -s
ls /etc
vi hosts

//add these three lines in hosts file
192.168.0.253 chefserver.example.com chefserver
192.168.0.252 developmentkit.example.com developmentkit
192.168.0.3 node.example.com node

//save it by using escape :wq
exit

Eighth Step //In chefserver machine

vagrant ssh chefserver
sudo -s
ls /vagrant/
dpkg -i /vagrant/chef-server-core_XXXXXXXX.deb //install chef server software in chef server
chef-server-ctl reconfigure
chef-server-ctl user-create admin admin admin admin@example.com LearnDevops -f admin.pem
chef-server-ctl org-create learndevops "Learn Devops Course" --association_user admin

cp admin.pem /vagrant
exit

Ninth Step // In Developmentkit machine

vagrant ssh developmentkit
cp /vagrant/*.pem .
ls
vi .chef/knife.rb

//Add below commands in knife.rb

```
current_dir = File.dirname(__FILE__)
log_level               :info
log_location            STDOUT
node_name                "admin"
client_key              "/home/vagrant/admin.pem"
chef_server_url          "https://chefserver.example.com/organizations/learndevops"
cookbook_path            ["/home/vagrant/cookbooks"]
```

cat .chef/knife.rb // to display all above commands - to verify
knife ssl fetch
knife client list // it will shows learndevops-validator
knife bootstrap node.example.com -N node -x vagrant --sudo // it will ask vagrant password - enter "vagrant"
knife client list //it will show node also

cd cookbooks/
ls
knife cookbook upload my_cookbook //uploading my_cookbook to chefserver
knife node run_list set node 'recipe[my_cookbook]' //it will show node run list and same uploaded to node run list
ssh node 'sudo chef-client' //give yes and enter vagrant node password in vagrant node - "vagrant"

Tenth Step
Verify Now hello.txt is copied in node /tmp folder

ANSIBLE COMMANDS

Installation

Vagrant File -Installation

```
Vagrant.configure("2") do |config|

config.vm.define "ansible" do |ansible|
ansible.vm.box = "ubuntu/trusty64"
ansible.vm.network "private_network", ip: "192.168.0.51"
ansible.vm.hostname = "ansible"
end
config.vm.define "node1" do |node1|
node1.vm.box = "ubuntu/trusty64"
node1.vm.network "private_network", ip: "192.168.0.52"
node1.vm.hostname = "node1"
end

end
```

```
vagrant up  //to up the two machines
vagrant ssh ansible //login ansible machine
```

```
Ansibe Installation for ubuntu // In ansible machine

sudo apt-get update
sudo apt-get install software-properties-common
sudo apt-add-repository ppa:ansible/ansible
sudo apt-get update
sudo apt-get install ansible
```

```
//In ansible machine

ssh-keygen
pwd  // it will be in a directory /home/vagrant
ls -la //to list hidden files and folders - it will show .ssh folder
cd .ssh  // changing the directory to .ssh - it will show generated private and public key
cat id_rsa.pub //copy the public key
(or)
vi id_rsa.pub //copy the public key

//In node machine
sudo -s //to obtain a admin rights or login through root user
cd /root  //change the directory to root folder
cd .ssh   //change the directory to hidden .ssh folder
ls        //to view the authorized_keys file
vi authorized_keys  //paste the key
```

(or)
sudo -s
echo "ssh-rsa paste_the_key">/root/.ssh/authorized_keys

// go back to ansible machine

vi hosts => add ip address => 192.168.0.52

ssh-agent bash
ssh-add .ssh/id_rsa // add the private key

ansible -i hosts -u root -m ping all

//In ansible machine

ls -lha /etc/ansible //to view the ansible core files
cp -R /etc/ansible myplatform // copy the ansible core files into myplatform directory

cd myplatform // change the directory to myplatform
ls -lha // confirm all the core files or copies to myplatform

---

//In Ansible machine - Creating configuration in myplatform directory

vi ansible.cfg
inventory = hosts //uncomment inventory and change the hosts path

vi hosts // open the hosts file in editor
192.168.0.52 // insert ip address of node1

ansible -u root -m ping all
ansible -u root -m shell -a 'hostname' all
ansible -u root -m shell -a 'df-h' all
ansible -u root -m shell -a 'whoami' all

Creating Main.YML file
//change the directory to myplatform

```
cd roles              //change the directory to roles
mkdir basic               // create a directory called basic
cd basic                  //change directory to basic
mkdir tasks               // create a directory called tasks
cd tasks             //change the directory to tasks
vi main.yml           // create a main.yml file using vi editor

// In Main.Yml file - paste the below commands

- name: "Installing Vim"
apt: pkg=vim state=installed


(or)


// Multipe installaiton configuration -- // vi roles/basic/tasks/main.yml

- name: "Installing Vim"
  apt: pkg=vim state=installed

- name: "Installing DNS Utils"
  apt: pkg=dnsutils state=installed

- name: "Installing Vim"
  apt: pkg=git state=installed

 Run => ansible playbook -K playbook.yml

  (or)

- name: "Installing additional software"
  apt: pkg={{item}} state=installed
with_items:
        - dnsutils
        - git
        - vim
        - ntp
        - at
        - lvm2
```

// Go to myplatform directory or home directory - Create a playbook.yml
// it need to be created in myplatform directory

vi playbook.yml // to create a new playbook.yml file using vi editor

//IN playbook.yml file -paste below commands

```
---
- hosts: all
  become: true
  roles:
   - basic
```

// Run playbook.yml using following command - verify you are in myplatform directory

ansible-playbook -u root -s playbook.yml

Docker Commands

docker info
docker pull
docker images
docker run
docker ps
docker ps -a
docker stop
docker rm

docker run <image>
docker run --name=<customname> <image>
docker run --rm <image>
docker run --d <image>
docker run --d -it <image>
docker run --d -p 4000:4000

Examples:

```
docker pull alpine
docker run alpine sh
docker ps -a
docker run -d  -it alpine sh
docker run -it alpine sh

ps

docker run -it --name=MyLinux alpine sh
docker ps -a
docker stop MyLinux
docker rm MyLinux


docker pull microsoft/nanoserver

docker run -it microsoft/nanoserver cmd

docker run -it microsoft/nanoserver cmd /c echo "some message"


docker network ls
docker inspect nat

docker inspect

docker pull microsoft/iis:nanoserver

docker run -d microsoft/iis

docker run -d -p 8000:8000 microsoft/iis:nanoserver

docker ps -a

docker inspect
```