### **GIT**

**Git installation and repository configuration**

#sudo **dnf** install git ;; #**git** –version to install and chech the version

#**git** config --global user.name "murali" ;; #**git** config --global user.email "murali@gmail.com" to set email & user name

#**git** config --list ;; #**git** config --global --unset user.email to unset your email address

#**mkdir** ~/path/git && cd /path/git && git clone <git-remoterepo-url> To clone existing remote repo to local

#**cd** /path/git/testapp && git init && git status if no repo is present then to create a new local repo & check status

**#git** remote -v to see list of all remote repos

|  |  |
| --- | --- |
| **Git Local & Remote repository management** | |
| **Managing Local repo**  #**mkdir** <repo-name> && **cd** <repo-name> create a dirctory for each remote repo, helps to maintain seperate local repos  #**git** clone <git-remoterepo-url> **\*** Only first timeto clone the remote repo to local #**git** remote add origin < remoterepo-url> To add remote repo to local repo  #**git** pull origin main to pull the latest updated file, ifany.  #**vim** test.txt (Add: content as needed, :wq) #**git** status to check the add and commit status of file #**git** add . to add untrack file to staging area  #**git** commit -m "tag message" to commit the changes  #**git** log [-2][--oneline --graph] to see commit logs  #**git** show [commit-id] to dee changes in commit #**git** push-u origin main (enter credentials) #**git** pull origin main to pull the latest updated file, ifany.  #**touch** .gitignore ; git add .gitignore ; git commit -m "git ignore file" ; git status to add a file to enter ignore files  **Tag** #**git** tag to show the list of tag. #**git** tag -a <new-tag> -m "comment" <commit-id> To give tag to commit  #**git** show <tag-n> to see particular tag's commit create #**git** tag -d <tag-n> to delete a tag  **stash** #git stash to stash working-space file to stash area #git stash list to see stashed item list o/p: (0)(1)(2)(3)(4) [recent <-- oldest] #**git** stash apply stash@(1) to bring back stash to workspace | **Git Branch Management** #**git** branch [-a] to list local branches (-a local & remote repos) #**git** checkout <b-n> to change the branch  #**git** branch [-d/D] <b-n> to delete a branch (-d/D to delete/force) #**git** push origin --delete <b-n> to delete branch in remote repo  #**git** fetch --all To fetch new branches from remote-repo, were others pushed new branches #**git** merge <b-n> && git log --oneline to merge branches In case of conflict with file name while merging the branches, use vim and reslove conflict, save & exit, then #**git** add . && **git** commit -m "merge conflict resolved" && **git** psh origin <b-n> to push the conflict resoved file to reore repo #**git** mv file1 test1 to rename file #**git** branch -m <oldb-n> <newb-n> ; **git** push origin <b-n> to rename branch & push changes  **Reset** #**git** reset <file-n> (or) git reset . to reset staging area. #**git** reset --hard to reset changes from staging/working area.  **Revert** #**git** revert <commit-id> go back to pervious commit / changes  #**git** clean [-n dryrun][-f force][-f <file-n> for particular file] to remove untracked file  Check:  Gitlab-ci-linter |
|  |  |

### **Chef**

In webpage "manage.chef.io", create a chef-server account & download starter-kit from manage organization.

Copy the starter kit to localhost and extract the zip file.

**Inside Chef-repo directory**

#**knife** ssl check run from chef-repo directory, to verify ssl from api's

#**chef** generate cookbook cb-name If cookbook directory not present

#**chef** generate recipe rcp-name run from inside cookbook

#**chef** exec ruby -c cb-name/recipes/rcp-name.rb To check syntax error

#**chef**-client-client -zr "recipe[cb-name::recipe-name],recipe[cb2-name::recipe-name]"

#**knife** cookbook upload cb-name to upload cb to chef server to chef server

#**knife** bootstrap <node-pvt-ip> -U myuser -P mypassword Need to do for every node to bootstart from wrokstation

#**knife** node run-list set node1/2/.. "recipe[cb-name::rcp-name]" to attach recipe to individual nodes

#**knife** node show node1/2/..

#**knife** cookbook/node/client/role] delete (cb/n/c/r)name to check the uploaded cb's/nodes/clinets/roles list

**Inside roles directory**

#**vim** devops to create a roles file use any name

name "devops"

description "comment as needed"

runlist "recipe[cb-name::rcp-name]"

:wq

from chef-repo directory

#knife

**Automating the process in node's**

#vim /etc/crontab in every node add thi file

\* \* \* \* \* root chef-client-client

:wq This will make node to check the clinet server as scheduled, using root

|  |  |
| --- | --- |
| **Chef Recipe Resources -** | |
| #**vim** cookbook-name/recipes/recipes-name.rb  Add: below contents into recipe file as needed  ## Recipe to install Nginx on an Ubuntu system package 'nginx' do  action :install end  ## Enable and start the Nginx service service 'nginx' do  action [:enable, :start] end  ## Sample Chef Recipe: Directory, File, and Ownership directory '/opt/my\_app' do  owner 'myuser'  group 'mygroup'  mode '0755'  recursive true  action :create end  ## Create a file within the directory file '/opt/my\_app/index.html' do  content '<html><body>Welcome to my app!</body></html>'  owner 'myuser'  group 'mygroup'  mode '0644'  action :create end  ## Copy the file to another location (e.g., /var/www/html) cookbook\_file '/var/www/html/index.html' do  source 'index.html'  owner 'root'  group 'root'  mode '0644'  action :create end | ## Recipe to capture node attributes and save them to a file file ‘/path/’ do  content <<~ATTRIBUTES  Hostname: #{hostname}  IP Address: #{ip\_address}  Platform: #{platform}  Total Memory: #{memory\_total}  ATTRIBUTES  owner 'root'  group 'root'  mode '0644'  action :create end  ## Execute linux commands using recipe. execute 'create\_backup\_directory' do  command <<-EOH  mkdir /opt/my\_app/backup  EOH  not\_if { ::File.exist?('/opt/my\_app/backup') } end  execute 'copy\_file\_to\_backup' do  command <<-EOH  cp /opt/my\_app/index.html /opt/my\_app/backup/  EOH  not\_if { ::File.exist?('/opt/my\_app/backup/index.html') } end  ## execute multiline Linux commands within a Chef recipe execute 'Execute shell script' do  command <<-EOH  # Your multiline shell commands go here  mkdir /opt/my\_app/backup  cp /opt/my\_app/index.html /opt/my\_app/backup/  # Add more commands as needed  EOH end  ## Creating user and adding user to group. user "murali" do action :create end  group "avengers" do action :create members 'murali' append true end |

### **Doker Containeraization Engine**

#**dnf** install docker ;; #**systemctl** [status/start/stop/enable] docker.service To manage the docker.service damone

#**docker** images List images from localhost

#**docker** [search/pull] image-name[:tag] To search and pull images from dockerhub

#**docker** login To login into docker hub remote-repo

#**docker** push your-repo/your-image-name:tag Push an image from locaalhost to remote repository

#**docker** rmi [img-n] [img-id] or #**docker** image remove img\_n To remove the images from localhost

#**docker** ps [-a] (-a all containers) List running containers

#**docker** run [-it/-d] img-name:tag /bin/bash To Run(build) a container in interactive/detached mode

Additional arguments: [--name=container\_n] [-v docker-vol:/container/path/]

[-v --mount source=/path/,target=/container/path/] [--network my-network] [-p 80:80]

#**docker** [stop] [rm] container\_n To stop and remove container

#**docker** run -td --name=nginx-container -p 80:80 nginx To port expose for servieses needed port

#**docker** port nginx-container To check the port assigned to container

#**docker** [volume/network] [--help/create] [volume-name/network-name] To manage docker volumes

#**docker** container [--help/start/attach/stop] container-name To manage docker containers

#**docker** exec -it container-name Execute a command inside a running container

#**ctrl**+p+q to detach from container without exiting ;; #**exit** to stop & exit from container

#**docker** diff container-name To check the modifications happened in container after build

#**docker** commit container-name your-repo/new-imagename:tag To tag directly while creating image

#**docker** tag <image-name> <your-repo/ image-name:tag> To tag an existing image

#**docker** stop/rm $(docker ps [-a] -q) (-a only active containers) To Stop/remove all containers

#**docker** [image/container] prune Remove all images / stopped containers

A **Dockerfile** is a text file that contains instructions for building a Docker image

A **Docker Compose** file is a YAML configuration file that defines a multi-container Docker application.

|  |  |
| --- | --- |
| Dockerfile | Docker-compose |
| #**docker** build -t newimg-name:tag To build a image from Dockerfile  #**vim** Dockerfile # Use an official CentOS base image FROM centos:8  # Set the maintainer label LABEL maintainer="your-email@example.com"  # Install necessary packages (Apache web server) RUN yum update && \  yum install -y httpd && \  rm -rf /var/cache/yum/\*  # Copy your application files into the container COPY /local/path/ /var/www/html/  # Expose port 80 for HTTP traffic EXPOSE 80  # Set environment variables ENV MY\_VAR=my\_value  # Specify the working directory WORKDIR /app  # Add a new group RUN groupadd mygroup  # Add an existing user to the group (append, not replace) RUN usermod -a -G mygroup myuser  # Define a volume mount for data persistence VOLUME /data  # Add a healthcheck (optional) HEALTHCHECK --interval=30s --timeout=3s CMD curl -f http://localhost/ || exit 1  # Add custom labels LABEL version="1.0" description="My custom image"  # Define build-time variables ARG MY\_ARG=default\_value  # Multi-stage build: Build stage FROM node:14 AS build WORKDIR /app COPY . . RUN npm install && npm run build  # Multi-stage build: Final stage FROM centos:8 COPY --from=build /app/dist /usr/share/nginx/html  # Cleanup unnecessary files RUN yum clean all && rm -rf /var/cache/yum/\*  # Start Apache in the foreground CMD ["/usr/sbin/httpd", "-D", "FOREGROUND"] | #**docker**-compose [up] (-d detach mode)] / [down] #**docker**-compose -f docker-compose.yml -f docker-compose.prod.yml up (-f combines multiple yaml files)  #**vim** docker-compose services:  web:  image: nginx:alpine  ports:  - "8080:80"  volumes:  - ./html:/usr/share/nginx/html   redis:  image: redis:alpine  ports:  - "6379:6379"   db:  image: postgres:alpine  environment:  POSTGRES\_USER: myuser  POSTGRES\_PASSWORD: mypassword  POSTGRES\_DB: mydb  healthcheck:  test: ["CMD-SHELL", "pg\_isready -U myuser -d mydb"]  interval: 10s  timeout: 5s  retries: 3   app:  image: myapp  depends\_on:  db:  condition: service\_healthy  networks:  - mynetwork  networks:  mynetwork: |

### **Ansible**

**Configuration Of Ansible server & all nodes**

**On Ansible Server& all nodes**

#**vim** /etc/sshd/sshd\_config

(Uncomment: PermitRootLogin yes (In Authentication) & PasswordAuthentication Yes ;; comment#PasswordAuthentication No)

Create “ansible” user & passwd and also assign sudo permissions "ansible ALL+(ALL) NOPASSWD: ALL"

NOTE: everything need to be performed from ansible user

**On only Ansible Server**

#**ssh-keygen** -t rsa -b 4096 (to create a ssh pub&pvt keys)

#**vim** /etc/ansible/ansible.cfg (Uncomment: inventory=/etc/ansible/hosts & become\_user=root)

#**vim** /etc/ansible/hosts to add nodes in ansible server hosts file

Add:

[devops] ---> [grp-name]

10.0.1.2 |

10.0.1.5 |--> Add node's pvt ip-addr

[management] ---> [grp-name]

10.0.1.2 |

10.0.4.5 |--> Add node's pvt ip-addr

:wq

#**ssh-copy-id** ansible@node-pvt-ip (prompt nods ansible passwd for first time)

(do invididualy to every node, to add ssh pub-key to nodes)

**Addng new nodes to Ansible server**

When node to be added to server, just need to add node pvt-ipaddr in groups in

"#**vim** /etc/ansible/hosts" & run #**ssh-copy-id** ansible@node-pvt-ip"

**Host Pattern (on Ansible server)**

#**ansible**-inventory [--list] [--list --yaml] [--graph] to see all groups from inventoey

#**ansible** all/grp-n/grp[0]/[-1]/[0:5] --list-hosts to list ranage of node from a grp

#**ansible** grp1[0:5]:grp2[10:15] --list-hosts to list ranage of node from multiple grp

**Ansible Adhoc, Modules & playbook**

* Ansible Adhoc commands

#**ansible** all/grp-n/grp[0]/[-1]/[0:5] -a "ls -lSrt" (-a argument) to list content from range of node

#**ansible** all/grp-n/grp[0]/[-1]/[0:5] -ba "sudo yum install httpd -y" (-b become-user(sudo) to list content from range of node

* Ansible Modules (To execute single module)

#**ansible** all/grp-n -b -m yum -a "pkg=httpd state=present"

#**ansible** all/grp-n -b -m service -a "name=httpd state=started"

package state: present/absent/latest // Services state: restarted/stopped/started/enabled/reloaded/enforcing/permissive/disabled

#**ansible** grp-n -b -m user -a "name=murali' to create a user in node groups

#**ansible** grp-n -b -m copy -a "src=/path/ dst=/path/ to copy files from server to nodes

#**ansible** grp-n -m setup [-a "filter=\*ipv4\*] full info abt all nodes from grp

**Ansible Vaults**

#**ansible-vault** [create/decrypt/edit/view/rekey] murali-pb.yml To encrypt/decrypt playbook

**Ansible Playbook**

Target Section: Highlighted in Green, playbookwill be appplied on the target hosts

Tasks Section: Highlighted in blue, mentioned task will be executed on the target hosts & also added condition as example

Handler Section: Highlighted in purple, mentioned services status can be handeled.

#**ansible**-playbook --syntax-check murali-pb.yml To check syntax errors in playbook.tml

#**ansible**-playbook murali-pb.yml --check To dryrun the playbook to check for errors/any blocks

#**ansible-playbook** murali-pb.yml To execute the playbook

|  |  |
| --- | --- |
| **# Ansible YAML Playbook Draft** | |
| #vim murali-pb.yml  Add:  ---  - name: Install and start Apache  hosts: devops / [0:5]  user: ansible  become: true  connection: ssh   tasks:  - name: Install Apache #Install Apache package  yum:  name: httpd  state: present  (or)  tasks:  - name: Install package using yum (for Red Hat-based systems)  yum:  name: httpd  state: present  when: ansible\_os\_family == 'RedHat'  - name: Install package using apt (for Debian-based systems)  apt:  name: apache2+  state: present  when: ansible\_os\_family == 'Debian'  - name: Start Apache service #Start Apache service  service:  name: httpd  state: started   - name: Copy custom config file #Deploy custom config  copy:  src: /path/to/myapp.conf  dest: /etc/httpd/conf.d/myapp.conf  owner: ansible  group: ansible  mode: '0644' | - name: Delete a file #Remove old config file  file:  path: /etc/httpd/conf.d/oldconfig.conf  state: absent  - name: Run the shell script and append output  shell: sh /path/to/file.sh >> /path/to/output.log    - name: Copy custom config file #Deploy custom config  copy:  src: /path/to/myapp.conf  dest: /etc/httpd/conf.d/myapp.conf  owner: ansible  group: ansible  mode: '0644'   - name: Delete a file #Remove old config file  file:  path: /etc/httpd/conf.d/oldconfig.conf  state: absent   - name: Run the shell script and append output  shell: sh /path/to/file.sh >> /path/to/output.log   - name: Add a cron job #Add a crontab schedule for script  cron:  name: "Run the shell script every day at midnight"  minute: 45  hour: 20  day: 31  month: "\*"  weekday: "\*"  user: ansible  job: "sh /path/to/file.sh >> /path/to/output.log"  … |

### **Apache - Tomcat Webapplication server –** Dynamic

Download JDK-V21 rpm.tar.gz package, extract and install

#**wget** <jdk-21-urlrpm.tar.gz> ;; #**dnf** install jdk-21.rpm ;; #**java** --version

Download apache-tomcat9 rpm package, extrac and install

#**wget** <apache-tomcat9.rpm.tar.gz> ;; #**tar** -xzvf <apache-tomcat9.rpm.tar.gz>

#**mv** <apache-tomcat9.rpm.tar.gz> tomcat9 ;; #**cp** -r tomcat9 /usr/share/

#**groupadd** -r tomcat ;; **useradd** -d /usr/share/tomcat9 -r -s /bin/false -g tomcat tomcat;;

**chown** -R tomcat:tomcat /usr/share/tomcat9

#**echo** "export CATALINA\_HOME="/usr/share/tomcat9'' >> /home/murali/.bashrc

**source** /home/murali/.bashrc

**tail** -n 1 /home/murali/.bashrc

#**vim** /etc/systemd/system/tomcat.service

ADD:

[Unit]

Description=Apache Tomcat Web Application Container

After=syslog.target network.target

[Service]

Type=forking

Environment=JAVA\_HOME=/usr/lib/jvm/jdk-21-oracle-x64 (To find #alternatives --config java)

Environment=CATALINA\_PID=/usr/share/apache-tomcat-9.0.87/temp/tomcat.pid

Environment=CATALINA\_HOME=/usr/share/apache-tomcat-9.0.87

Environment=CATALINA\_BASE=/usr/share/apache-tomcat-9.0.87

Environment='CATALINA\_OPTS=-Xms512M -Xmx1024M -server -XX:+UseParallelGC'

Environment='JAVA\_OPTS=-Djava.awt.headless=true -Djava.security.egd=file:/dev/./urandom'

ExecStart=/usr/share/apache-tomcat-9.0.87/bin/startup.sh

ExecStop=/usr/share/apache-tomcat-9.0.87/bin/shutdown.sh

User=tomcat

Group=tomcat

UMask=0007

RestartSec=10

Restart=always

[Install]

WantedBy=multi-user.target

#**systemcl** daemon-reload ;; **systemctl** start/enable tomcat.service

#**firewall**-cmd --add-port=8080/tcp --permanent ;; #**firewall**-cmd --reload

#**vim** /usr/share/apache-tomcat-9.0.87/conf/tomcat-users.xml

Will Already be there check once:

<tomcat-users xmlns="http://tomcat.apache.org/xml"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://tomcat.apache.org/xml tomcat-users.xsd"

version="1.0">

ADD: just below to above details

<!-- Define roles and users for Tomcat authentication -->

<role rolename="manager-gui"/>

<user username="tomcat" password="STRONGPASSWORD" roles="manager-gui"/>

<!-- Additional roles -->

<role rolename="manager-script"/>

<role rolename="manager-jmx"/>

<role rolename="manager-status"/>

<!-- Additional users with different roles -->

<user username="admin" password="STRONGPASSWORD" roles="manager-gui, manager-script, manager-jmx, manager-status"/>

<user username="deployer" password="STRONGPASSWORD" roles="manager-script"/>

save & exit

**Incase of SELINUX isue use below**

#**setsebool** -P httpd\_can\_network\_connect 1 ;; #**setsebool** -P httpd\_can\_network\_relay 1

#**setsebool** -P httpd\_graceful\_shutdown 1 ;; #**setsebool** -P nis\_enabled 1

even after config is setup, only tomcat homepage access will be avaialbe to get the access of server status & Manager App. Need to make chanages in /usr/share/tomcat9/webapps

#**vim** /usr/share/apache-tomcat-9.0.87/webapps/host-manager/META-INF/conext.xml

MODIFY THIS LINE:

<Valve className="org.apache.catalina.valves.RemoteAddrValve"

allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1 | 192.162.1.\* (or) .\*" /> 192.162.1.\* for whole subnet ip-addtess

.\* from any ip-address

MODIFY THIS LINE: adding self ip

#**vim** /usr/share/apache-tomcat-9.0.87/webapps/manager/META-INF/context.xml

<Valve className="org.apache.catalina.valves.RemoteAddrValve"

allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1 | 192.162.1.\* (or) .\*" />

#**systemctl** daemon-reload ;; **systemctl** start/enable tomcat.service

We can configure the application into tomcat by adding the ".war" file to webapps folder

#cd /usr/share/apache-tomcat-9.0.87/webapps

**To chage the tomcat port#**

Config file:

#**vim** /usr/share/tomcat9/conf/server.xml

Modify port# from “<connector section”

**To Add “jenkins.war” file into the tomcat webapp container.**

Download “jenkins.war” file from ‘jenkins.io portal’ and copy to tomcat webapps directory

#**wget** <jenkins.war url> ;; #**cp** jenkins.war/usr/share/tomcat9/webapps

**#export** JENKINS\_HOME=/usr/share/jenkins

#**vim** /etc/systemd/system/tomcat.service

ADD: below to tomcat Environment variables

Environment=JENKINS\_HOME=/usr/share/jenkins

#**chown** -R tomcat:tomcat /usr/share/jenkins

#**chmod** -R 755 /usr/share/jenkins

#**systemctl** restart tomcat.service

Now refresh the browser and access

URL: <http://192.168.1.8:8081/jenkins>

Should be able to access the jenkins login page.

### **MYSQL**

#systemctl status mysql

#**mysql** -u murali To login into mysql

**Database**

#SHOW DATABASES;

#CREATE DATABASE murali;

#USE murali;

#SELECT DATABASE();

#DROP DATABASE murali;

**Table**

#CREATE TABLE members (Acct INT PRIMARY KEY/NOT NULL AUTO\_INCREMENT, Dept VARCHAR(255) DEFAULT 'Student' NOT NULL, Fname VARCHAR(255) NOT NULL);

#SHOW TABLES;

#DESC members;

#INSERT INTO members (Acct, Dept, Fname) VALUES(123, 'admin', 'himanish');

#INSERT INTO members VALUES (234, 'admin', 'vedansh'),(345, 'linux', 'rajesh');

#INSERT INTO members (Dept) VALUES('admin'); If NOT NULL specified at table creation, wont accept the data.

#SELECT \* FROM members; // SELECT Fname,Dept FROM members;

#SELECT \* FROM members WHERE Acct=345; // SELECT admin FROM members WHERE Acct=345;

#ALTER TABLE members CHANGE COLUMN Fname Firstname VARCHAR(255) NOT NULL;

#ALTER TABLE members ADD COLUMN Address VARCHAR(255) NOT NULL;

#ALTER TABLE members DROP COLUMN Address; // SELECT Fname AS 'Firstname' FROM members; (only on screen)

#UPDATE members SET Fname='newrajesh' WHERE Acct=345;

#DELETE FROM members WHERE Acct=123; Need to select the Pimarykey filter.

#DROP TABLE members;

**CONCAT**

#SELECT Acct, CONCAT(Firstname, ' ', Dept) AS 'Full' FROM members;

#SELECT Acct, CONCAT\_WS(' : ', Firstname, Dept) AS 'Full' FROM members;

**Replicate the Databases (One time Backup)**

#mysqldump -u username -p murali > murali\_backup.sql

#CREATE DATABASE muralibackup;

#mysql -u username -p muralibackup < murali\_backup.sql

|  |  |
| --- | --- |
| Relocate table from SAME Database (Murali) | Relocating table DIFFERENT DB (murali to himanish) |
| Existing Table: members **|** New Table: staff  CREATE TABLE murali.staff LIKE murali.members; INSERT INTO murali.staff SELECT \* FROM murali.members; | Existing DB: murali **|** New DB: himanish Existing Table: members **|** New Table: staff  CREATE TABLE himanish.staff LIKE murali.members; INSERT INTO himanish.staff SELECT \* FROM murali.members; |