GIT(2.23)

Git status: Check status of the current.

Git init : Initialing git repository

Git config --Global user.name "sanjay"

Git config --global user.email sanjay\_kaumarreddy@rediffmail.com

git add file name: To add file from work space to staging /index are.\

Git commit -m "first commit" : To commit from index area to local repository.

Git log : To display all commit. **WORK-SPACE---STAGING/index-LOCAL-REPO**

Git push: To send changes to remote repository.

Git show commitid: Ii will display all files with that commit id.

Git clone giturl : TO get entire repository from github.

git log --oneline: Display logs in single line.

Git reset --soft commitid : This will bring all file from local repository to index area with that commitid.

Git reset filename: This will bring file from staging area to work space area.

Git reset --mixed commit \_id : This will directly take local repository to work space area.

git branch : To Display all branches.

Note: If we cloned with git clone, No need to use git init commnd,To initiate. Just navigate to that folder.

Git pull : To get all updates from current branch.

Git --aboart : To aboart the conflicts.

Stash Memory: This is temporary area to store files .

git stash list : This will display all stashed files.

Git stash save "name of the stash" : This store all files in to stash memory.

Git stash show stash[0] : This display corresponding file in stash memory.

Git stash pop stash[0]:This will bring file from corresponding stash memory to staging/index area. And removed file from stash memory.

Pop \* Apply

Git stash apply stash[0]: This will bring all files from stash memory to index/staging area.

This will not delete file from stash memory.

Note: Here stash[0] is a memory location

Git stash drop stash[0]: TO drop the stash memory.

.ignorefile: This file contains all files names, That all files will not be added and committed. These files will ignored by git.

**Git tag : It will display all tags in current branch.**

**Git tag commitid tag\_name: This will create tag with respective commit id.**

**Git push orign tagname: To update the tage name in centralized location.**

**Git tag -d tagname: This will remove the tag from local repository,**

**Git push orign -d tagname: To remove tag in git hub.**

**Git branch -d branchname: To delete the branch in local repo.**

**Git ls-tree –r master --name-only: Gives all files in specified branch**

Note: If any changes in new branch it is not possible to delete new branch until unless these new changes will commit to main branch.

GitPull Request: While merging future branch in master branch,Pull request will required. This pull request will have approved by any one of collaborated access. This will happen from ui.

Note: Few facility like CODEONERS file will provide module level approval facility to users.

Git MergeToll: This tool will come by default from git. This tool will open ui and display conflicts automatically. We can directly move conflict code TO/From.

git config --mergetool: This command will open git ui with conflict code.

**GItTasks: Creating repositories, Creating Branches, Giving collaborating access to user, Code merges, Creating GitWebHucks , web-hook configuration, ssh-key adding,CODEONERS file editing.**

**Git clone can be happen in 3 ways :**

1. **Http : Directly clone using with : git clone url**
2. **SSH : copy the public key in git hub , Place the private key available to git command line tool. Then use clone option.**
3. **Token based: This token will created in ui of repository level and orginazation level.**

**1) In organization -repository will be there**

**2) deploy keys will apply on repository level.**

**git status : It will show current status of the file. It will display files**

**git diff : It will compared the modefied file with tracket file , and it will show difference**

Git Interview Question:

<https://dev.to/aershov24/11-painful-git-interview-questions-you-will-cry-on-1n2g>

Git Conflict will come based on same file in different branches while merging.

If different files are there in different branches no conflicts will come.

Bit bucket:

1. Atlasian tool , Datacenter version is there for production , paid plug-in are available for connecting to different tooll like sonar qube .example plugin code owner plug-in

MAVEN:3.6v

Maven Build Phases:

Deploy

Install

test

Build

compile

Validate

(check file structure) (.java to .class) (generating jar,war) (test asper junit) (copying .m2 folder) (uploading)

Maven life cycles:

Site

Clean

Default

Maven Goals: These are all triggers to maven build phases. Goals will trigger maven phases.

These goals are based or the pom.xml.

Example: mvn compile:compile ---------> This will trigger compile phase.

Note: In all phases only test phase can skip.

Maven always uses either one or two settings files. The global settings defined in (${M2\_HOME}/conf/settings.xml) is always required. The user settings file (defined in ${user.home}/.m2/settings.xml) is optional. Any settings defined in the user settings take precedence over the corresponding global settings

**GAVM Parameters: Group Id,Artifact Id,Version,Model**

Their types are local(.m2), Central(maven community) and remote(third party ex: nexus)

To run test classes in Maven, you need surefire plugin, check and configure your settings in setting.xml and pom.xml for a property named “test.”

Maven plugins are used to  
• Create a jar file  
• Create war file  
• Compile code files  
• Unit testing of code  
• Documenting projects  
• Reporting

**8) List out the dependency scope in Maven?**

The various dependency scope used in Maven are:

• Compile: It is the default scope, and it indicates what dependency is available in the classpath of the project  
• Provided: It indicates that the dependency is provided by JDK or web server or container at runtime  
• Runtime: This tells that the dependency is not needed for compilation but is required during execution  
• Test: It says dependency is available only for the test compilation and execution phases  
• System: It indicates you have to provide the system path  
• Import: This indicates that the identified or specified POM should be replaced with the dependencies in that POM’s section

**9) Mention how profiles are specified in Maven?**

Profiles are specified in Maven by using a subset of the elements existing in the POM itself.

**10) Explain how you can exclude dependency?**

By using the exclusion element, dependency can be excluded

**11) Mention the difference between Apache Ant and Maven?**

Apache Ant Maven  
• Ant is a toolbox – Maven is a framework  
• Ant does not have formal conventions like project directory structure – Maven has conventions  
• Ant is procedural; you have to tell to compile, copy and compress – Maven is declarative ( information on what to make & how to build)  
• Ant does not have lifecycle; you have to add sequence of tasks manually – Maven has a lifecycle  
• Ant scripts are not reusable – Maven plugins are reusable

**12) In Maven what are the two setting files called and what are their location?**

In Maven, the setting files are called settings.xml, and the two setting files are located at

• Maven installation directory: $M2\_Home/conf/settings.xml  
• User’s home directory: ${ user.home }/ .m2 / settings.xml

**14) List out the build, source and test source directory for POM in Maven?**

• Build = Target : Output of maven(maven generated files) will store here.  
• Source = src/main/java : developers code will be here.  
• Test = src/main/test : testing related information

Note:You will find the class files ${basedir}/target/classes/.

**16) Explain what would the “jar: jar” goal do?**

jar: jar will not recompile sources; it will simply just create a JAR from the target/classes directory considering that everything else has been done

**17) List out what are the Maven’s order of inheritance?**

The maven’s order of inheritance is

• Parent Pom  
• Project Pom  
• Settings  
• CLI parameters

**19) Explain how you can produce execution debug output or error messages?**

To produce execution debug output you could call Maven with X parameter or e parameter

Maven local repository: .m2

Maven remote repository: Maven community // this is based on company infracture

Maven centeral repository: Internet

SonarQube☹port 9090)

Sonar qube configuration filese. sonar.properties

Default Profile: Sonar Way

Quality Profile: This is collection of rules.

We can create own custme profile with default custme profile .We can set our profilce as default profile

We can take a Backup of sonarqube profile that profile is like a xml file.With that xml file we can create sonar profile (By importing this xml).

We can deactivate this rules based on the request.

To check the changes in the sonar profile we can check change log option by selecting to and from.

Note: Without login view permissions are available.

Default Gateway: Sonar Qube Way

Quality Gate: This is collection of thresh hold values.

Here rules/metrics are the basic elements.

1. **Code coverage.( So many drop down will be there)**
2. **Blocker Issues. (So many drop down will be there)**
3. **Critical Issues. (So many drop down will be there)**
4. **Comments lines.( So many drop down will be there)**

**Above all drop down contains thresh hold values.**

**Issues:**

1. **Blocker: Problem will come in producation, jdbc ,input/output streams problems.**

**This should fix before deploying to the production.**

1. **Critical: sql injection possibilities.**
2. **Major: lot of duplicated code, variable declared but not used.**
3. **Minor: Big methods,Un nessary curly brackets,varaible declearation in not standard format, Namng convenction.**
4. **Info: just for information.(nether a bug nor a quality flaw jus a findings)**

**Note: In technical review first we can deside it is isseue or not , If it is isseue asign that isseue to developer. After developer fix that isseue then we will chage that a fixed. After this , If again scan is done . If isseue is not fixed properly by the developer again sonar will open this isseue.**

**We can change all this with bulk change.We can send the push notification when any thing is happening.**

**We can deactivate the rule next time that will display as removed state.**

**Please find below table for better undeerstanding.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **severity** | **status** | **Attachedto** | **Planning** | **comments** | **Time to fix** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Blocker** | **Open** | **Not Assigned** | **Notplaned** | **comment** | **time** |
| **Critical** | **Conform** |  | **Planed** | **“based or decession”** | **Isseue time to take decession.** |
| **Major** | **Resolved as fixed.** | **Admin** | **Not planed.** | **“based or decession”** | **Isseue time to take decession.** |
| **Minor** | **Resolved as false positive** | **Developer-1** |  | **“based or decession”** | **Isseue time to take decession.** |
| **Info** | **Resolved as wont fixed.** | **Developer -2** |  |  | **Isseue time to take decession** |

<https://www.javainuse.com/misc/sonarqube-interview-questions>

Jenkins(2.x) Production level installation with different db:

/etc/init.d/Jenkins : To make Jenkins process as demon process

1. /var/lib/jenkins/jobs ---------------------> It will store all jobs
2. /var/lib/Jenkins/workspace/jobname/------🡪 It contains all file.
3. **.hpi**: This is the file extension to upload custom plugin.

2) /var/lib/jenkins/config.xml---------------------> Basic configuration file

3) ./jenkins.sh-------------------------------------> To start jenkins

4) Major Plugins:

1) Git plugin -------Useful to connect git(scm)

2) SSh copy plugin -------------It will copy files to destinnation

3) Blueu Ocean ------------------It will change view of the jenkins

4) Git Parameter Plug-in------------It will provide git credentials as parameters.

5) Extended Mail Plug-in------------It will send mail to respected person when specific event occurse

6) RoleBack plugin------------------

7) Maven Plugin

9) Sonar Scanner

10) Backupplugin --------------To take a backup

11) AnsiblePlugin --------------TO contact ansible server

12) DockerPlugin -------------To connect docker container

13) PublishOverSSH -------------Sending and running a commands in target machine

14) EmbededBuildStatusPlugin-----It will display the job status.

15) BigBucket-------------------To integrate with bigbucket

16) GreenBalls -----------------To change color of succes blue to grean

17) Junit-----------------------To publish junit testcases.

18) Rebuild---------------------With out entering parameters again when build rerun this plugin will enter paremeters again

19) JobGeneratorPlugin--------This plugin gibves flexiblity to define templat4es,

Developers will create own jobs

20) Role Based authentication --------To give authrization to user

21) Configuration Slicing Plugging--------TO change bulk changes in multiple jobs

22) Delivery pipeline plugin: To display delivery pipeline as proper UI

23) Build pipeline plugin:

23) NOfification plugin: This will send email notification

24) POll mail trigger plugin.

25)Cloud be adviser plugin.

CONFIGURE: This will describe all installed softwares/

GLOBAL TOOL CONFIGURATION: Tool of all with direct installation.

Build Triggers types:

1) Manual 2)Pollscm3) Buildperiodically 4)Build with token in url 5)Git web hook

\* poll SCM will build if any changes happen in build in git location, this will run based or corn jobs.

\* Build periodically build in one specific time . Even changes are happening or not it will build code.

var/lib/Jenkins.

Jenkins Tasks: CreatingFreeStyleProjects, CreatingPipelineScripts, CreatingParameterisedBilds,Providing Build Tokens,To the onsite people.Adding Jenkins slaves.

How to add Jenkins slave to master:

1. By using ssh(22)
2. By running a jar file in jenkins slave machine.(jnlp)[for this port enabling required from Jenkins side]
3. By running a command on Jenkins

Jenkins libraries: Reusable code in groovy .  
  
library stricture:   
  
  
   
vars. Contain grovy  
src: customized grovy .

Resource: other files consumed by

sample grovy file :

mailUser.groovy.

Def mailUser()  
{  
echo “send amail to customer”  
  
}

usage in Jenkins file :  
  
@Librareis(‘LibraryName@branch’)  
  
mailUser()  
  
Note : first based on LibraryName is searched in

**Mangejenkins>configurationSystem>GlobalPipelineLibraries**

Sample Groovy Snippet

node {

stage('Preparation') {

sh “git 'https://github.com/javahometech/my-app.git”

}

stage('Build') {

def mvnHome = tool name: 'M2', type: 'maven'

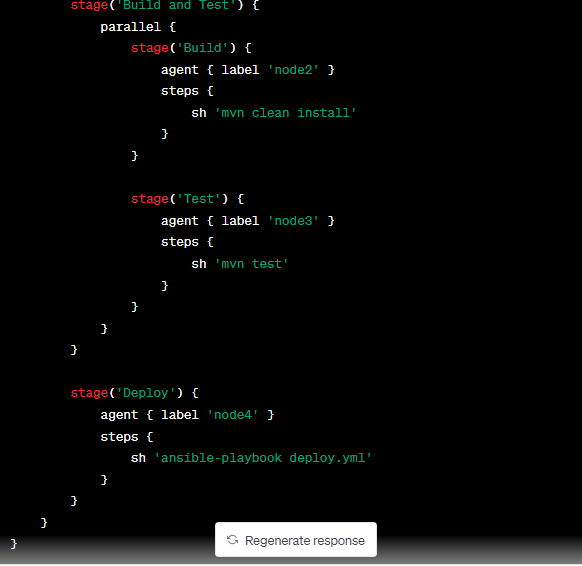
sh "${mvnHome}/bin/mvn package "

}

}

|  |  |
| --- | --- |
| Declarative | Scripted |
| Try Catch block is not support but always handled the errors | Try Catch block is supported |
| Asy to understand | Difficult |
| Less customization | More customization |
| Yaml based syntax | Ues grovy bases scripted language |

**Jenkins distributed Build** Multiple agent machines concurrently run on multiple machines.  
  
----------------------------------------------------------------------------------------------------------------------------  
  
In Jenkins file we can write multiple stages with different nodes .  
  

while running with parallel()

block inside stages are running parallelly .  
For each stage node can change with agent { lable ‘ node’}

But in starting of Jenkins file we agent should nun we need to write agent none

We can use try catch and finally block in jenkns file .

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Ansible(2.5)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Ansible configuration file : /etc/ansible/ansible.cfg

Default inventory/hostfiles : /etc/ansible/ansible.hosts

host\_key\_checking = true/false : This will use to set host keys as known hosts.

Gather\_facts:no/yes: This variable is used to disable setup module. by default this module will enabled in ansible yaml file.This variable is in variable section.

Serial: This is a key word to set machines as a batch.

This is as a batch of machines. Takes are applied on batch.

To clubbing the group:

[newgroupname:children]

Children 1

Children 2

General Variables for All connections:

|  |
| --- |
| ansible\_host,ansible\_ssh\_key,ansible\_ssh\_pass,ansible\_ssh\_private\_keyfile,ansible\_port |

We can specify all these values in inventory file. for password we can us ansible valtue.

Ansible-valtue encrypt /decrypt secretfile\_name : To lock file& unlock file.

Ansible-voltue view secretfilename : To view file

Ansible –voltue edit secretfilrname To edit file

Ansible-voltue rekey secretfilenmae To change file.

Modules:

ansible webservers -m service -a " name=httpd state=started" : To start the service

ansible webservers -m service -a " name=httpd state=stopped" : To Stop the service

ansible webservers -m yum -a " name=httpd state=present" : To Install the service

ansible webservers -m yum -a " name=httpd state=abesnt" :To uninstall the service

ansible webservers -m yum -a " name=httpd state=latest" : To update the service

ansible webservers -m copy -a " src=/etc/file.txt dest=/var/folder" : To Copy the file from one location to another location.

ansible webservers -m file -a " dest=/var/folder mode=read oner=sanjay group =devops " : To change the file permission and owner ship of the group.

ansible all -m setup --tree /tmp/facts : Setup module will give all information about target machines in the form of jsons.

ansible all -m setup " filter=facter\_\*" : This will displays specific detail of all facts .

anisble 192.168.13.34 -m user -a " user=sanjay password=redhat" : This user module will create user with specified password in that machine.

|  |
| --- |
| Play Book stricture:  -----  Host Section: This section describes about host details.  Variables Sections: This section describes all variables using before. We can mention variable files here.  Users:  Tasks: Invidual task names and tasks can write here  Templates: To write common file with different values with .j2 templates  Handlers: If any certain task is carried out successful and, That task is did changes in target machine that is called handlers. After handler’s task is completed again control goes to normal location. |

1) Become =yes : This will describe that play book is running with sudo user.

2) {{ varable name}} : To use the variable values in the middle of the play book we use this pattern. Here varable name is replaced by variable values.

3) -include=playbookname : This will include another play book .

4): This shell module will use to execute shell command.

shell:/user/bin/whoami

register:username

This will register the output of the above command to further use.

output of above command will store in username variable.

5)Debug module will use for print messages and variable values while play book execution.

-debug: msg=" inventery hostnames are {{inventory\_hostnames}}"

Prompting for inputs:

While executing plabooks some values required like db password.

**vars\_prompt:**

**-name:"dbpasswd" //ur value will store in dbpassword**

**prompt: " Enter password for data base"**

**Handlers:**

We use handlers using “notify” syntax. If task is successful (did any changes) completed this notify will work otherwise it will not work.

From notify control will go to handler section.

Example for handlers:

tasks:

-name: copy the website config file

copy: src=/tmp/httpd.conf dest=/etc/httpd/httpd/httpd.conf

notify:

-Apache Restart

handlers:

-name: Apache Restart

service : name=httpd status=restart

Conditional execution:

Sometime we want to run tasks in a play book in certain condition.

tasks:

-name: Install apache in cent os

yum : service =httpd state=present

when: ansible\_os\_family=="redhat"

**Note: This task will execute when variable values is red-hat only**

**Templates:**

**While template modules get executed it will read the template file and change all the variables to its value and copy the file in target place.**

**tasks:**

**-name: Coping file using template**

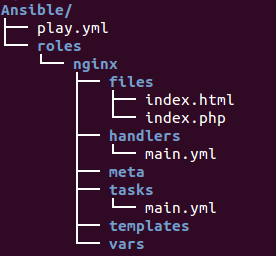
Template: src=templates/index.j2 dest={{doc\_root}}/index.html mode=0644

This value come from play book.

To create role:

ansible-galaxy init role\_name

Ansible Role Streacture:



To call a ansible role in a play book:

roles:

-{name:rolename ,http\_port:8080}

we can override the values when calling the role.

Loops:

For repeating tasks, we use looping concept.

-name: add several users

user:

name: {{item}}

state: Andhra

group: Civil

with\_items:

-rest\_user\_1

-rest\_user\_2

------------------------------------------------------------------------------------------------------------------------------------------Dictionary:

network\_interfaces:

-name:enp0s3

configure:true

This red color block is Dictionary

method:dhcp

parameters:

-param:pre-up sleep

val:2

List of Disctionaryes

-name:enp0s8

configure:true

This blue color Block is Dictionary

method:static

address:192.168.250.10

netmask:255.255.255.0

----------------------------------------------------------------------------------------------------------------------------------------

List:

**---**

*# A list of tasty fruits*

**-**Apple

**-**Orange

**-**Strawberry

**-**Mango

**...**

Ansible exception:

Block : -- Problematic code

Rescue: Handling the problem

Always: This always exiguity

Gathering Facts

# plays will gather facts by default, which contain information about

# the remote system.

#

# smart - gather by default, but don't regather if already gathered

# implicit - gather by default, turn off with gather facts: False

# explicit - do not gather by default, must say gather facts: True

gathering = implicit

# retry files

# When a playbook fails by default a .retry file will be created in ~/(where the ply book placed)

# You can disable this feature by setting retry\_files\_enabled to False(in ansible.cfg)

# and you can change the location of the files by setting retry\_files\_save\_path(in ansible.cfg)

#retry\_files\_enabled = False (in ansible.cfg)

#retry\_files\_save\_path = ~/.ansible-retry (in ansible.cfg)

Note: This retry files will useful when roles/play book partially executed. We can re execute these play book by passing parameters like ***--retry***

Other files:

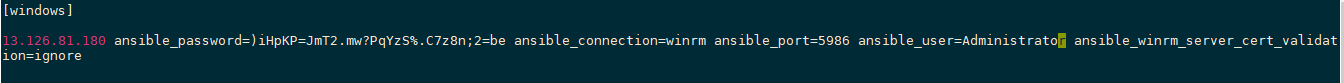
[DEPRECATION WARNING]: The sudo command line option has been deprecated in favor of the "become" command line arguments. This feature will be removed in version 2.6.

Deprecation warnings can be disabled by setting deprecation\_warnings=False in ansible.cfg.

Controlling Windows machine:

In ansible Machine:

1. **pyWinrm** should installed in ansible control machine(pip install pywinrm)
2. below changes required in ansible host file.



In windows machine:

Powershell and .net framw work installed

To install required softwares use below powershell script

<https://raw.githubusercontent.com/ansible/ansible/devel/examples/scripts/ConfigureRemotingForAnsible.ps1>

above script will run in powershell only.

Example commd to ping windows host from ansible:

ansible windows -m win\_shell -a date

ansible tower interview questions:  
  
How do you manage permissions and roles within AWX?   
  
Answer: AWX uses a role-based access control system to manage permissions. This allows administrators to assign specific roles and permissions to users and groups based on their level of access. To manage permissions and roles, an admin can navigate to the Users or Teams section within AWX and create or modify users and teams as need.  
1 )

Types of Variable in AWX tower.

1. Inventory variables: These variables are specific to a particular host or group in the inventory. They are defined in the inventory file or in the host or group variables in AWX Tower.
2. Playbook variables: These variables are defined within the playbook itself using the "vars" section. They can be overridden by inventory variables or extra variables passed at runtime.
3. Extra variables: These variables are passed to a playbook at runtime using the "extra\_vars" parameter. They can be used to override inventory variables and playbook variables.

The precedence of variables in AWX Tower is as follows:

1. Extra variables passed at runtime
2. Inventory variables
3. Playbook variables

In Ansible, error handling can be done using a variety of methods. Here are some examples:

1. Ignore errors: You can ignore errors by adding the "**ignore\_errors**: yes" option to a task. This will allow the playbook to continue even if the task fails.
2. Fail a task: You can fail a task by adding the "**failed\_when**" option to a task. This option allows you to define a condition that, when true, will cause the task to fail. For example:
3. Retry a task: You can retry a task a certain number of times using the "retries" and "delay" options. This is useful for tasks that may fail temporarily due to network issues or other factors.
4. Notify a handler: Handlers are tasks that are only run when notified by another task. You can use handlers to handle errors by notifying them when a task fails.

1. **docker images - To see all images in current machine.**
2. **docker pull– To download images.**

**docker run –it imagename:version /bin/bash – To run the container from image**

1. **ps –elf – To check what process is running in docker container.**
2. **Docker ps – To check running containers. We can use filter here with ---filter “status=exited”**
3. **Docker ps –a - To check all containers.**
4. **docker stop/start container name or container id – To stop/start the container.**
5. **Docker rm<container name or container id> – To remove stopped container.**
6. **docker inspect <container name or container id> – To get container information in json format.**
7. **Docker rmi image name or image id – To remove image.**
8. **docker run –d –-name container\_name -p 8080:80 imagename:version**

**note: p(caps)----🡪 will automatically assign the port number from host.Small (p) manual**

1. **docker run –p 8080:8080 –v /host machine path: /container pathimagename:version – To map thevolumes with host machine.**
2. **docker push username/repositoryname:version**

**doker login --name <name of the user in docker> -- mail id mail id of the account – To login into docker.**

1. **docker tag image\_name username/repository:tag – to tag the image**
2. **docker push username/repository:tag – to upload images to docker hub.**
3. **Docker exec -it containername /bin/bash: To login to runing container:**
4. **Docker cp source destination : To copy the files from host machine to container.(This will work on host not in container)**
5. **Docker run -d --name=db redis:latest**

**Docker file instruction:**

1. **FROM – to download the image from Docker hub. (This will use at the time of docker file starting)**
2. **ADD – Too download the context from url.**
3. **CMD – it will run at the time of creating container it is uniqe (if we add more cmd command last command will work)**
4. **ENTRY-POINT—it will run at runtime (If we add more enty points first one will work ,because it will not override)**
5. **ENV – this instruction can be used to set environment veriable.**
6. **EXPOSE – to open the ports when running the container.**
7. **MAINTAINER – author of the Docker file.(This is not mandatory)**
8. **RUN – on the top of the existing layer this command will work.**
9. **USER–to set the user id or user name to use when running the image.**
10. **VOLUME – enable access to a location on the host system from container.**
11. **WORKDIRECTORY – to set current working directory. If directory not exist it will create in the container.**
12. **ON BUILD – when one image is used as the base for some other image.(like down streamjenkins job).**
13. **Docker network ps – to display all networks.**
14. **docker network create --driver = brdge java-hari : To create docker customer bridge network.**
15. **docker inspect network name – gives all ips of that network.**
16. **docker logs—all logs.**
17. **Docker volume create volume name – to create a volume name.**
18. **Docker volume ps – to display all docker volumes.**

**VOLUMES**

|  |  |
| --- | --- |
| **NORMAL VOLUMES**  **This volumes are residing in the host machine**  **Managed by the Docker**  **Non Docker process cannot modify this volumes** | **BINDED VOLUMES**  **This volumes are residing in the host machine**  **This can be managed by user.**  **Non Docker process will modify this volumes easily.** |

1. **Docker import: Creates an image from a tarball.**
2. **Docker load: load an image from tarfile. (image is inside tar file.)**
3. **Dockerbuild: To create a image from docker file.**
4. **Docker commit – creating a image from a container, pausing. it is temporary if it is running.**
5. **Docker container prune: To remove stopped containers**
6. **Docker images prune : To remove untagged and unused images used by containers.**
7. **Docker images prune -a: To remove all unused Docker images**
8. **docker cp hostpat/filename containerid:containerpath: To copy the files from host to container**
9. **docker cp containerid:containerpath hostpat/filename : To copy files from container to hostmachine**
10. **Naming convensions of docker :**

**DockerFile.azure.prod,DockerFile.azure.qa,azure.prod.DockerFile,azure.qa.DockerFile.**

**Docker build -f DockerFile.azure.prod -t dokcerimagename: This wiss specify exact docker file.**

1. docker save -o <path for generated tar file><image name>: To save the images to tarfile . (This is useful to copy images one machine to other machine)
2. docker load -i <path to image tar file>: To load the image from tar file.
3. docker-machine scp ./docker-images.tar remote-machine:/home/ubuntu

**d**

docker save myimage:latest | gzip > myimage\_latest.tar.gz

**docker log –format{{.LogPath}} containerid : This will falter the specific string value from log out put.**

**docker service logs**

**docker service create**

**docker images are stored in /var/lib/dockerimages/devisemapper**

**version: ‘3’**

**services:**

**database:**

**image: redis**

**web:**

**image :nginx**

**ports:**

**-56392: 80**

**version: ‘3’ --------------- version of the dockerfile.**

**services: ---------------services**

**database: ---------------service name**

**image: redis --------------- image name**

**web:**

**image :nginx ------------- image name**

**ports : ----------------To expose the ports**

**-56392: 80**

**Docker-compose will used to up more services at a time.**

**docker- compose up/down -d --scale database=4**

**docker-compose ps --status of the container in current stack.**

**docker-compose kill –to stop the container started with docker-compose up.**

**Docker ps –size : Display all sizes of the containesrs.**

**DockerOwnRegistry:**

**how-to-host-you-own-docker-registry/**

**Docker Machine: This will perform docker Docker Machine commands. This docker machine will create docker nodes (based on the envirnoment) and it will install docker Engine on all hosts.**

**Example: Docker Machine ls: This will disply all docker nodes in current envirnomnet.**

**Docker Engine : This will run docker commands.**

**Example: Docker ps**

**Docker Networking :**

**When docker is installed on machine it will create one interface this is called docker 0 bridge interface. This will connect to regular network interface.**

**Docker core networks are :**

1. **Bridge:**

**This is default network in docker . This network is connected to host nic card as a bridge.**

**Every request is coming to this network will redirected to host nic card. To expose container services in this network port forwarding is required.**

**While creating the container port forwarding is required(-p,P)**

1. **Host:**

**This network will expose the container to outside world without any port forward.**

**All ports of host will used by container. This leads to security breach .**

**Example : If any service is running in the container automatically that service will take host port.**

1. **Null: This network is useful to isolated the container.**

**Note: Bridge,Host,Null all network scope is local**

**Note: I can attach multiple interfaces to the container , Those multiple interface belongs to multiple networks.**

**Docker network connect cid: Connecting the network with a container.**

**Docker network disconnect cid: To disconnect**

**Note: Docker demon is the service is running in the background .**

**Docker run –itd –name test-cid –hostName test.locl.san –-network networkname image:version**

**Question-1:**

**Q: While restating the host container is in stopping stage how to start container automatically.**

**A: While creating the container we should specify the restart flag to always. No alternative way.**

**Docker run –itd –name testServer –-hostname test –ip 17.168.0.11 –restart always**

**Question-2:**

**Q: Is it possible to use port forwarding after container started.**

**A: Not possible.**

**Open points:**

1. **Docker volumes are saved in this location by default .**

**/var/lib/dockervolumes**

1. **) File system in Docker file:[ 1)Overlay 2) overlay2 3)aufs 4)xfs 5)devise-mapper 6)btvfs]**

**3)tmpfs mounts are stored in ram so after restart data will removed.**

**4) volume can mount n number of containers .**

**Docker run –itd –mount source=volume1 destination = pathInsideContainer read/write**

**Default is write**

1. **NameSpace: name space is the isolated work space of collection of container,images**

**Ip netns add namespace1**

**Ip netns exec namespace1 bash : Login to that name space.**

**Dockers Security:**

**Linux.**

**mtime — updated when the file contents change. This is the "default" file time in most cases.**

**ctime — updated when the file or its metadata (owner, permissions) change**

**atime — updated when the file is read.**

Cat > filename ----------------------------------------create

Cat >> filename------------------------------------------edit file

Touch file1 file2 file3 -------------------------------------to create multiplefiles in a single shot

Vi filename ------------------------------------------------------------------to edit file

Nano filename -----------------------------------------------------------to create file

Tomake a multiple directory

Mkdir –p kernalTech/{linux/{advlinux,linuxclstr},aix/{hacmp,lpr},storage/{san,netapp}}

Renaming the directory

Mv oldname new name

Softlink: Is a shorkut file,if original file is gone link is broken,can be create in different partition ,can created cross partition so inode numbers are different

ln -s filename linkname.slink

HardLisk: is a back up file ,one file is delaeted another file data is available,cannot be created cross partition so inode numbers are same.

Ln filename linkname.link

Less/more/head/taile

To disk play customer lines:

Head –n /etc/passwd (n can be any number)

Tail -n /etc/passwd (n can be any number)

Sed : to search a word in a file

Sed ‘s/searchfor/modefywith/g’ filename

It will not replace with new word but if will display with combination

Mounting:

/etc/mtab : file system temprory mounting

/etc/fstab :pernment mounting

Partition of disk:  
updating the partition with out restarting the system:

Partprobe /dev/sda

Partx –a /dev/sda

Kpartx /dev/sda

Formatting the partion with ext4:

Mkfs.ext4 /dev/sda7

Mkfs.ext3 /dev/sda7

Mounting colums:  
/etc/fstab

Devicename /directoryname /typeoffilesys / mountingoptions /dumpoptions /checkopt

To check who is using the partion:  
fuser –cu /dev/sda7

Lsof /dev/sda7 to check which file is open

Fuser -ck /kernel/hello to kill athe open file connection

df -h usage information of mounted file system.

Du -h size of the file or directory.

Assigned a lable to a partition:

#e2lable /dev/sda7 ktdisk

/etc/fstab

|  |
| --- |
| LABEL=’--------------‘ =/sdb1 /kernel ext4 defaults 0 0 |
| UUID=’222222222’ /sdb1 /kernel ext4 defaults 0 0 |

Swap formula:

If ram < 2gb swap will be 2\*2gb=4gb

If ram >2gb swap wil be 2gb+ramsize

MandatoryPartions:

/boot , /, /swap

Creating s swapping:

To change the partition as a swap change the hexa code

Fdisk /dev/sda6

Select t change the hexacode as 82

Mkswap /dev/sda6 #for format

Swapon /dev/sda6 #for on the swap

Make a entry in /etc/fstab

Removing swapping:

Swap off /dev/sda6

Remove the entry from /etc/fstab

Delete that portion with fdisk /dev/sda6

Encrypting Partition :

Cryptsetup luksFormat /dev/sda5

Cryptsetup luksOpen /dev/sda5 filename

Enter password

Then the partition name is changed to dev/mapper/name

Make entry in etc/fstab

Make entry in etc/crypttab

Add data to that folder

Unmount that partion

Umount /dev/mapper/name

Cryptsetup luksclose /dev/mapper/name

Volume group create:

Vgcreate ktvg /dev/sda7

Vg extend ktvg /dev/sda8

Logical volume create:

Lvcreate -L 300 M -n ktlv ktvg

Logical volume extend:

Lvextend -L +200 M /dev/ktvg/ktlv

Resize2fs /dev/ktvg/ktlv

Check with df-h

Logical Volume reduce:

Umount

Request for downtime

Check with df-h

e2fsck -f /dev/ktvg/ktlv

resize2fs /dev/ktvg/ktlv 300M

lvreduce -L -200M /dev/ktvg/ktlv

mount

Moving or migrating the Lv(data) from one pv to another

Failed pv < good pv

Check that mounted directory contain data or not

Cd /ktdir

Umount /ktdir(this is mounted directory of corrupted r failed pv)

Vgextend /ktvg /dev/sda7

Pvmove /dev/sda6(corrupted pv) /dev/sda7

Again mount

Removing corrupted pv:  
vgreduce /ktvg /dev/sda6(corrupted pv)

Deleting /removing a lv:

Lvremove /dev/ktvg/ktlv

Removing the volume group

Vgremove ktvg # pvrmove /dev/sda6

User and group administration

Home/username created

Var/spool/mail created

Uuid & guid created

Etc/password ,etc/-password(backupfile)

Etc/shadow .etc/-password(backupfile)

Password file content:

Name:linktopasswordfile:uid:gid:comment:homedirectory:shell

Etc/shadow file fields:

Username: encryptedpassword:dayspassword lastchnaged:passwordmustbe changedin:userwatningtime:after password expire when user will lock

Resweved

User created:

Useradd sanjay -u 505 -g 506 -d /home/kernel -c hero

Usermod -L(lock) name

Usemod -U(unlock) name

Userdel -r name

To change password parameters:

Change username

groupadd groupname

Modefing properties of the group:

groupmod -g/-n groupname # group id,name

Add the user to a group:

Usermod -G groupname username

Adding multiple users:

gpassword -M user1,user2 ktgroup

Adding ,Removing single user with gpassword:

gpassword -a usersingle groupname

gpassword -d usersingle groupname

we can use gui tool

system-config-users

ACL LIST

To mount acl list:

Mount -o /dev/sda7 /ktdir

/etc/fstab

/dev/sda7 /ktdir ext4 default ,acl 0 0

To check which permissions’ are there in directory/file

getfacl /ktdir

setfacl -m u:ktdir : rx /ktdir ---------to give permissions to existing user .

setfacl -m g:ktgroup :rx /ktdir -----------to give permissions to existing group.

Setfacl -x u:ktdir :rx /ktdir ------------to remove permissions’ to existing user

Setfacl -x g:ktgroup :rw /ktdir-----------------to remove permission to existing group

Setfacl -R u:ktuser :r /ktdir--------------------to give permission to inner content.

Process State:

Ps --------------------------current running process in terminal

Ps –a ---------------------to display all permissions’

Ps -u sanjay ---------to check which process are running by specific user

Total signals: 64

Kill -1 relode kill -15 terminate

Kill -9 kill kill -20 stop

Nice values range:

-20 to +19

Ex: renice -10 pid

Top command:

Firstline: uptime ,no of users,load average first minute, 5th minute,15th minute

Secondline: NO of process and current status

Thirdline: cpu utilization

Fourthline: Ram,Swap Memory cache memory

Network

/etc/sysconfig/network-scripts /ifconfig-eth0----to change the ip address

/etc/sysconfig/network-scripts -------all network devices information

/etc/sysconfig/network--------------------------------this file shows hostname

/etc/hosts-----------------------------------------------it acts as local dns

/etc/resolv.conf-----------------------------------------Address of dns

Service network manager restart

Service network restart

Selinux Configuration

/etc/selinux/config ----------------------------------

disable(complet disable)/enforcing(fullsecure)/permissive(full secure but policy applicable)

ls -Z to display context of a file

ls -ldZ to display context of directory

Context means additional information.

Getsebool -a | grep <servicename>

Setsebool

Kernel parameters : 0-shutdown,1-singleusermod,2-multiuserwith out nfs,3-multiuserwithoutgui 4-unused,5-multiuserwith gui

Run level programs are automatically started at system boot up

They are exheguted from the following directory.

0 –etc/rc.d/rc0.d 1-etc/rc.d/rc1.d 2-etc/rc.d/rc2.d

3-etc/rc.d/rc3.d 4-etc/rc.d/rc4.d 5-etc/rc.d/rc5.d 6-etc/rc.d/rc6.d

To check which run level is currently running

Who – r

To change run level init <runlevel number 1 to 6>

Grub

To configure the grub /boot/grub/grub.config (it shows kernel version)

Uname –a -----------------to check which vesion we are using

Arch – to check architecture of a machine

All kernel modules are residing in ------------------etc/lib/modules

modinfo cdrom --------------to check the module information

modprob vfat---------------------to install the module

modprob -r vfat---------------to remove the module

/etc/modprobe.d/blacklist.conf-----------------------add this module in this configuration file to prevent loading when insert the device

To block the device:  
ex: block list usb\_storage

Job auto mation

/etc/crontab

/etc/cron.deny /etc/at.deny

/etc/cron.allow /etc/at.allow

Job scheduling:

minutes(0-59) hours(0-23)day of month(1-31),month of year(1-12),day of weak(0-6)

tocheck which process are running by whic user

crontab -u username

login to ssh(secure socket host)

ssh 192.168.1.1

ssh –keygen----------------------- to generate keys

root/.ssh------------private and public keys are stored in this with this files id\_rsa,id\_rsa.pub

to copy the public key in remote system

ssh-copy-id -i /root/.ssh/id\_rsa.pub 192.168.1.1

to taransfer file :  
scp /ktfile 192.168.10.95:/root/

rsync -rv ssh /ktdir 192.168.1.1:/root/

it will search ktdir directrory in root folder if it is found it apply changes

rsync -avz ssh /ktdir/ktfile 192.168.1.1:/root/ktdir

it will search for ktfile in ktdir and update the change.

**Packages**

**rpm -uvh <packagename> to update the package**

**all repository files are here:**

**/etc/yum.repos.d/**

|  |
| --- |
| **[ktrepo] #repository file name can any name**  **Name = device.id.repo #repositoryname can any name**  **Baseurl =** [**file:///var/ftp/pub/rhel6**](file:///\\var\ftp\pub\rhel6) **(or) ftp://192.168.10.1/pub #where dump is**  **Enabled = 1 #to enable r disable repository**  **Gpgcheck = 0 #signature verified or not** |

**Yum install <packagename> -y**

**Compressing:**

**Tar –cvf /opt/etc.tar(name to b) /etc(to be compressed)**

**Gzip /opt/etc.tar -----------etc.tar.gz(become like this)**

**Gunzip /opt/etc.tar.gz -----------etc.tar (become like this)**

**Tar -xvf /opt/etc.tar -----------etc (become like this)**

**Gzip ------------file compress single file**

**Gzip -------------replace the original file**

**Services settings**

**Chkconfig --list**

**Chkconfig <service> on --------service available after restart**

**Chkconfig <service> off ---------service unavailable after restart**

**Zcat filename to see all files in zipped directory**

**My Sql DB**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Mysqladmin -u root password “new password”;**

**Mysql -u root -p to give super user**

**Export PATH = $PATH:/user/bin:/user/sbin**

**To start mysql or booting time /etc/rc.local**

**To check my sql is running or not**

**Ps –ef| grep mysqld**

**To start my sql server**

**Cd /usr/bin/safe\_mysqld&**

**To shut down**

**Cd /usr/bin/mysqladmin -u root -p shutdown**

**To create user in database the below content is ised**

**Insert into user (host,user,password,select\_priv,,insert\_prev,update\_priv) values (“192.168.1.1”,”guest”,PASSWORD(“guest 123”),’Y’,’Y’,’Y’);**

**Example Privileges:**

**Select\_prev,insert\_prev,update\_prev,delete\_prev,drop\_prev**

**Configuring docker registry to push docker images**

**In docker.json file menction below lines,**

**{**

**“insecured-registeries”: [“192.168.13.45:9000”],**

**“disable-insecureregistry” : true**

**}**

**DockerOwnRegistrySetUp**

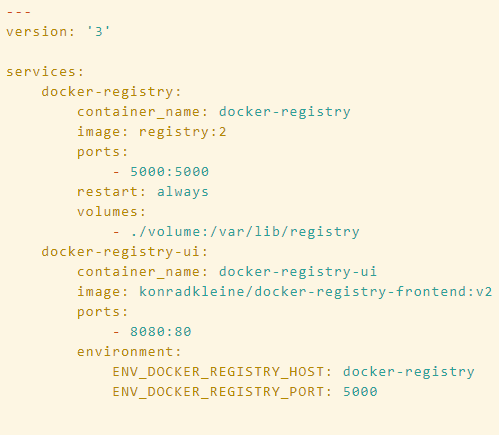
**Understanding the setup**

* **we are having two machines Node 1 and Node 2**
* **The IP Address for Node 1 is 10.42.0.10**
* **The IP address for Node2 is 10.42.0.1**
* **Node 2 will host the docker registry and Node 1 is the machine where images are getting created and then pushed to the docker registry or pulled from the docker registry.**

**Prerequsitive**

* **Node 1 should have docker installed**
* **Node 2 should have docker and docker-compose installed**

**Creating docker-compose file on Node 2**

****

**Update the docker daemon.json file with following values in node1.**

**{**

**"insecure-registries":["selftuts.local.com:5000"] //hostnameofnode2(registry)**

**}  
 service docker start.**

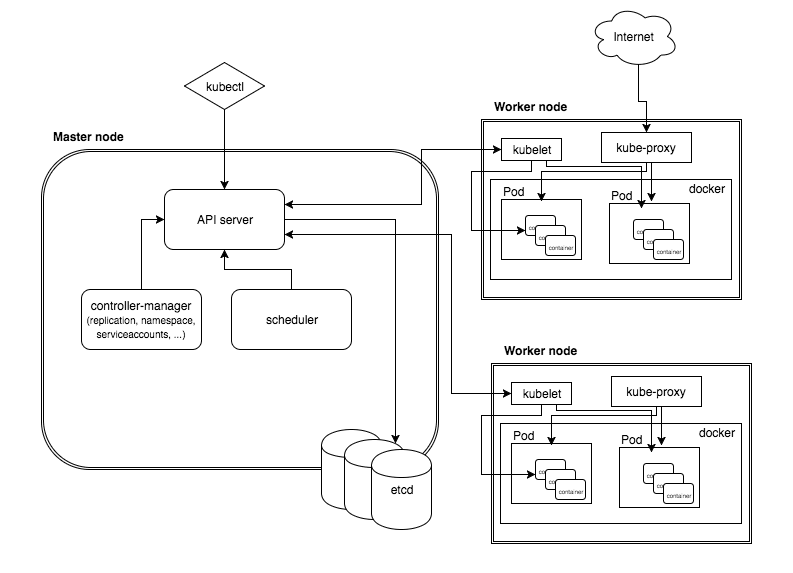
**To push docker image to docker own registry.**

**docker push hostname:port(node2)/username/<image-name>:tag**

[**http://IP-ADDRESS-OF-NODE2:8080**](http://IP-ADDRESS-OF-NODE2:8080) **//to access ui of docker registry.**

**DockerOwenRegistryWith ssl.**

**Kubernetes:**

****

**Basic commands:**

**Kubectl get pods : List of all pods.**

**Kubectl get nodes : List of all nodes.**

**Kubectl get rc : List of all replica controllers.**

**Kubectl get rs: List of all replication set.**

**Kubectl get deployments: List all deployments**

**Kubectl get svc List all services**

**Kubectl get pv List of all PV**

**Kubectl get pvc List of all pvc**

**Pod:**

**Controllers:**

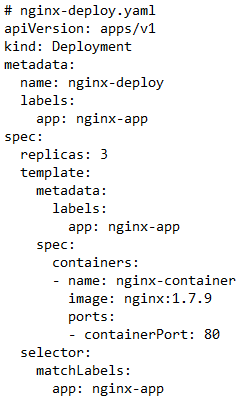
**---1) jobs.**

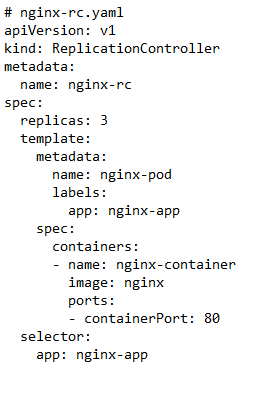
**---2) Replication Controller.**

**---3) Replication Set.**

**---4) Deployment.**

**---5) DaumenSets.( to create a single pod in specific nodes we can use this services)**

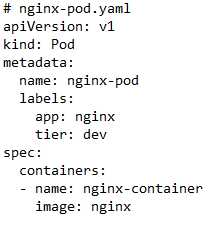






labels and selectors.

Note : Simple difference between the above deployment, Replica set, ReplicaController is match labeles only.



**Config Map:**

**This is for configuring the pods with configuration files.**

Creating Config Maps from

• Directories

• Files

• Literal Values**.**

Containers will take configuration from

• Configuration files.

• Command line arguments.

• Environment variables.

**CreatingConfigMaps**:

kubectl create configmap game-config --from-file=configure-podcontainer/configmap/kubectl/:

Above line is for creating config map from folder.In folder multiple configuration file can be present.

kubectl create configmap example-redis-config --from-file=redis-config:

Above line is for creating config map from single file.

kubectl create configmap special-config --from-literal special.how=very

Above line is for configuring config map from literal values.

**Acceesing ConfigMaps:**

Stored In files: By creating volumes we can use this config Maps. Here key is the Config file name and path is the file path.

Stored In Environment Variable: By using special key called “configMapKeyRef:”we can refer it.

**Secretes:**

Secrets are stored in ETCD in master.

Max- size -1mb only.

Secrets Can be careated in 2 ways.

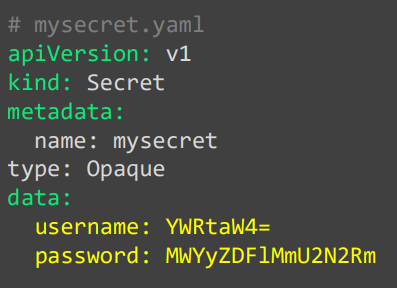
1): Using kubectl :

kubectl create secret generic db-user-pass --from-file=./username.txt -- from-file=./password.txt

In above command secret texts are in username,password files .

Note: This type of secrets can be used by Creating volumes with that secrets and consume it by mounting.

2): Throw manifest (manually):



Note: This type of secrets can consume secrets from environment variable.

**DeploymentStrategies:**

**Recreate: Deleting all old versions then create new versions. Down time will come.**

**Rolling Update(incremental)Default: New version will create and tested once this instance successfully accepting the requests this will added in the pool. One by one will add to the pool.**

**Canary Deployment: First two instances is checked for expected response, If response is expected result. Remaining all machines will ready then, Old version will deleted**

**Blue/Grean: Equal new versions are created after successful creation of new versions. Traffic will redirect to new versions. This will have done in load balancer level.**

**Deployment.**

kubectl create -f nginx-deploy.yaml : To create a deployments

kubectl set image deploy nginx-deployment nginx-container=nginx:1.9.1 : To update the image to latest one.

kubectl edit deploy nginx-deployment : To edit the deployment in run time.

kubectl rollout status deployment/nginx-deployment : To check the roll out status of the deployment.

**RollBack(history,undu,status)**

kubectl set image deploy nginx-deployment nginx-container=nginx:1.91 –record : This will record all activities on this deployment.

Note: If activity is recorded we can easily role back those changes.

kubect rollout history deployment/nginx-deployment : To display the role out history.

kubectl rollout undo deployment/nginx-deployment : This will roled out all activity on this deployment.

kubectl scale deployment nginx-deployment --replicas=5 : This will increase the replicas to 5.

**DaemonSet:**

**To deploy a single pod in each node in a cluster we can use this Daemon Set.**

A Daemon Set ensures that all (or some) Nodes run a copy of a Pod.

As nodes are added to the cluster, Pods are added.

As nodes are removed from the cluster, those Pods are garbage collected

**Use cases:** Node monitoring daemons: Ex: collectd,

Log collection daemons: Ex: fluentd.

Storage daemons: Ex: ceph

**Jobs:**

Job is a Controller in k8s, which supervises Pods for carrying out certain tasks.

Types: Run to complete(jobs), Cron jobs.

In Kubernetes, there are two types of jobs:

1. **Single-Task Jobs:** Single-task jobs are used to run a single pod to completion. Once the pod completes successfully, the job is considered complete. Single-task jobs are useful for running batch jobs or one-off tasks that do not need to be run on a regular schedule.
2. **Multi-Task Jobs:** Multi-task jobs are used to run multiple pods to completion. Once all the pods complete successfully, the job is considered complete. Multi-task jobs are useful for running periodic tasks or batch jobs that need to be run on a regular schedule.

Use cases:

One-time initialization of resources such as Databases

**Services:**

**--Node Port Service:**

**This service is used to expose the service to outside world.**

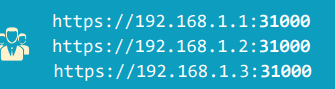
**Same Port activate on all machines with single service.**

**Port range is 30000 to 32767.**

Downsides:

You can only have once service per port.

If your Node/VM IP address change, you need to deal with that.

****

**Note: To give service to outside world we can give any one url from above.**

Imagine that, using nodePort service type you exposed your web app to outside world on the internet s which node IP and nodePort will you provide to end users? This is a problem with node port service.

**--Cluster Ip Service:**

This service is used to communicate with in the cluster. Like db service , out side service will not communicate with db. In this scenario cluster ip service we can use .

Default service in k8 is cluster ip service .

All the pods will communicate with cluster ip service only.

**--Load Balancer Service:**

**This is based on the where the k8 will deployed. To provide single url to the end user we will use this service. This service is depends on cloud which provider we are using . For on premise we need to follow different process. In this example we are using gcloud.**

**After creating this load balancer service external ip will automatically create.**

**Storage Volumes:**

**--Empty Dir:**

This will create at the time of pod creation. This volume will mount in pod file. All the data with this volume will go after pod is dead.

Use cases:

• Temporary space

**--Host Path:**

Mounts a file or directory from the host node’s filesystem into your Pod

Remains even after the pod is terminated

Similar to docker volume

Use cautiously when required

Host issues might cause problem to host Path

Note: There is no guarantee that same pod will create in same node so problem may come.

**--Persistent Disk(GCE):**

Google Compute Engine (GCE) this disk belongs to google cloud. In AWS(Elastic Block Store) , AzureDisk s are there. This is completely specific to where the k8 is deployed.

First raw disk will create in Google cloud, then it will be formatted, then it will be mounted to directory.

We will create the volume with this disk and mount this volume to mount paths. This will have done in manifest files.

This data will be still available even after pod is deleted.

Note: This will work only if disk is in the same Project/zone

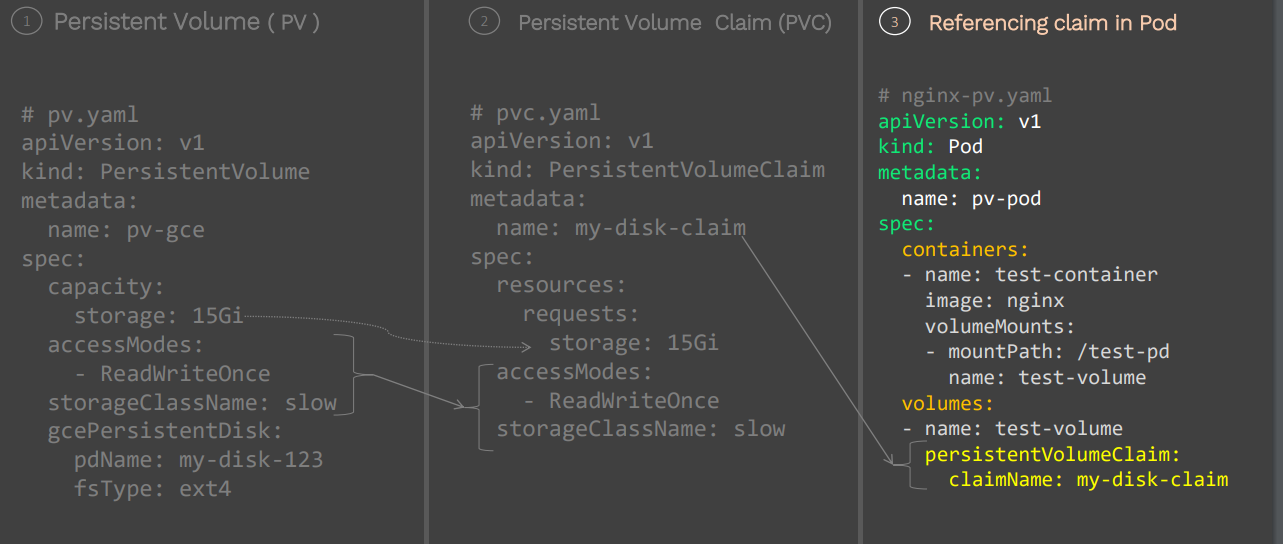
**--Persistence Volume.**

This is a volume created in k8 to use in manifest files.

This volume can be used with **Persistence Volume** Claim only. While creating PV it is formatted with file system.

**--Persistence Volume Claim: This is a request to get volumes from pool. In this PVC manifest file, we will point out the PV using with storage class tag.**

**Below diagram illustrate flow between PV,PVC**

****

Apache Tomcat

Configuration Files: server.xml,user.xml,catilina.sh,start.sh

Jconsole:

http,https(certificates self signed)

JVM monitoring with jconsole.

/webapp: This contains

Getting access log:

LogLoevel In tomcat :

1. Info 2.Debug 3.Warn 4.Notice 5.Crit 6.Alarm 8 Error

**Status code 403:**It refers to a forbidden error like if a file misses some security context.

**Status code 404:**It refers to an error message that it is an http response and the client was not able to communicate with the given server.

# Tomcat installation on EC2 instance

### Pre-requisites

1. EC2 instance with Java v1.8.x

### Install Apache Tomcat

1. Download tomcat packages from https://tomcat.apache.org/download-80.cgi onto /opt on EC2 instance

```sh

# Create tomcat directory

cd /opt

wget http://mirrors.fibergrid.in/apache/tomcat/tomcat-8/v8.5.35/bin/apache-tomcat-8.5.35.tar.gz

tar -xvzf /opt/apache-tomcat-8.5.35.tar.gz

```

1. give executing permissions to startup.sh and shutdown.sh which are under bin.

```sh

chmod +x /opt/apache-tomcat-8.5.35/bin/startup.sh

shutdown.sh

```

1. create link files for tomcat startup.sh and shutdown.sh

```sh

ln -s /opt/apache-tomcat-8.5.35/bin/startup.sh /usr/local/bin/tomcatup

ln -s /opt/apache-tomcat-8.5.35/bin/shutdown.sh /usr/local/bin/tomcatdown

tomcatup

```

#### Check point :

access tomcat application from browser on prot 8080

- http://<Public\_IP>:8080

Using unique ports for each application is a best practice in an environment. But tomcat and Jenkins runs on ports number 8080. Hence lets change tomcat port number to 8090. Change port number in conf/server.xml file under tomcat home

```sh

cd /opt/apache-tomcat-8.5.35/conf

# update port number in the "connecter port" field in server.xml

# restart tomcat after configuration update

tomcatdown

tomcatup

```

#### Check point :

Access tomcat application from browser on prot 8090

- http://<Public\_IP>:8090

1. now application is accessible on port 8090. but tomcat application doesnt allow to login from browser. changing a default parameter in context.xml does address this issue

```sh

#search for context.xml

find / -name context.xml

```

1. above command gives 3 context.xml files. comment (<!-- & -->) `Value ClassName` field on files which are under webapp directory.

After that restart tomcat services to effect these changes

```sh

tomcatdown

tomcatup

```

1. Update users information in the tomcat-users.xml file

goto tomcat home directory and Add below users to conf/tomcat-user.xml file

```sh

<role rolename="manager-gui"/>

<role rolename="manager-script"/>

<role rolename="manager-jmx"/>

<role rolename="manager-status"/>

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

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

<user username="tomcat" password="s3cret" roles="manager-gui"/>

```

1. Restart serivce and try to login to tomcat application from the browser. This time it should be Successful

NEXUS INSTALLATION

<https://devopscube.com/how-to-install-latest-sonatype-nexus-3-on-linux/>

yum update-y

yum install java-1.8.0-openjdk.x86\_64

mkdir/app&&cd/app

wget https://sonatype-download.global.ssl.fastly.net/nexus/3/nexus-3.0.2-02-unix.tar.gz

tar-xvf nexus-3.0.2-02-unix.tar.gz

mv nexus-3.0.2-02nexus

adduser nexus

chown-Rnexus:nexus/app/nexus

Open /app/nexus/bin/nexus.rc file, uncomment run\_as\_user parameter and set it as following.

run\_as\_user="nexus"

vi/app/nexus/bin/nexus.vmoptions(MEMORY OPTIONS)

-Xms1200M

-Xmx1200M

-XX:+UnlockDiagnosticVMOptions

-XX:+UnsyncloadClass

-Djava.net.preferIPv4Stack=truer

-Dkaraf.home=.

-Dkaraf.base=.

-Dkaraf.etc=etc

-Djava.util.logging.config.file=etc/java.util.logging.properties

-Dkaraf.data=/nexus/nexus-data

-Djava.io.tmpdir=data/tmp

-Dkaraf.startLocalConsole=false

Creating soft link:

ln-s/app/nexus/bin/nexus/etc/init.d/nexus

Execute the following commands to add nexus service to boot.

chkconfig--add nexus

chkconfig--levels345nexus on

service nexus start

---------------------------------------------------------------------

Nexus credentials we will configure in ui(Jenkins settings).

The repository name is mentioned in the <distributed-management tag>

The snapshot repository mentioned in <snapshot-repository-tag>

In nexus:

Types of nexus repositories:

---proxy central: Act as cache between local and central repository

---Hosted repository(maven/docker: Own repository generated by the project.

NGNX

Installation

**DEVOPS ENGINNER TO DO LIST:**

|  |  |
| --- | --- |
|  |  |

Checking Email Checking alerts Checking JIRA / Any ticketing tool for pending/scheduled tasks. Clear Notifications of alerting system.

Ensure if any new server is created and monitoring has been set up on that.

Verify if all the service running on that server are covered under the monitoring system.

Check and automate if any server is running out of disk.

Taking a backup of instance and restoring if required.

Taking a backup of Prod DB and providing that DB to developers on Staging / Testing Environment for testing of any issue.

Automation setup for daily tasks like (DB/Instance/Logs/Config-Files) backup.

In case of new project setting up new Jenkins Job. ( FreeStyle / Pipeline).

Making config changes on servers using (Ansible

/chef/ puppet).

Writing playbooks for automating daily tasks.

Deploying code on Development and Production servers.

Ensuring that post-deployment sanity of code is done and proper sign-offs are given.

Seeking for approvals from (Business/QA/UAT and Security Sign-offs).

Providing assistance during Audits.

Ensuring that access on servers are given to required users only that’s too after proper approvals.