MYTHRI

Mythri Consulting

DevOps - Sessions

Lokesh Kamalay

DevOps Areas

Core areas

SCM	Cont. Integration	Cont. Deployments/Delivery	Cont. Monitoring	Configuration Mgmt	Cloud Technologies	Languages	Platform
Git, **GitHub**, GitLab, BitBucket, SVN, TFS	**Jenkins**, **CloudBees Jenkins**, Bamboo, GitLab, Travis	**Jenkins**, CloudBees Jenkins, UCD, Terraform, Spinnaker	**Nagios**, ELK, Splunk, New Relic, App Dynamics	Chef, Puppet, **Ansible**, CloudFormation	**AWS**, Azure, GCE, Bluemix	**Bash**, Python, R, Ruby,	Physical machines, **Virtual Machines**, **Containers**

DevOps Areas

Continuous Integration: The practice of frequently integrating one's new or changed code with the existing code repository – should occur frequently enough that no intervening window remains between commit and build, and such that no errors can arise without developers noticing them and correcting them immediately.

Continuous Delivery: Is a series of practices designed to ensure that code can be rapidly and safely deployed to production by delivering every change to a production-like environment and ensuring business applications and services function as expected through rigorous automated testing

Continuous Deployments: Is the next step of continuous delivery: Every change that passes the automated tests is deployed to production automatically

Continuous Monitoring: Is the process and technology used to detect compliance and risk issues associated with an organization's financial and operational environment. The financial and operational environment consists of people, processes, and systems working together to support efficient and effective operations.

DevOps Skills Set Example

Area	Tools & Technologies
Versioning Tools	Git, GitHub, Bitbucket, Stash, SVN, CVS, RTC
Build Tools	Maven, Ant, Gradle, NPM
Scripting	Perl, Bash, Ruby and Python
Cloud Technologies	AWS, Azure
Analytics and Monitoring	Geneos, Nagios, Newrelic, ELK
Management Tools	Jira, Confluence, Sharepoint.
Continuous Integration	Jenkins, CloudBees Jenkins, Bamboo
Continuous Deployment	Ansible, Puppet, Chef
Repositories	Nexus, Artifactory
App/Web Servers	Apache Tomcat, IBM Web Sphere, IIS
(R)DBMS	Oracle, SQL Server, MySQL, DynamoDB, Sybase, Mongo, Teradata
Network Protocols	DNS, Telnet, TCP/IP, HTTP, HTTPS, SSH, SFTP, SSL
Operating Systems	RHEL, CentOS, Ubuntu, Windows
Virtualization	Virtual Box, Vagrant, VMWare, Citrix XenApp
Containers (Orchestration)	Docker, Swarm, Kubernetes
Management Areas	Project Management, Service Lifecycle Mgmt, Incident & Problem Mgmt, Change & Release Mgmt.
Misc	Control M, Informatica

SCM Git

Version Control

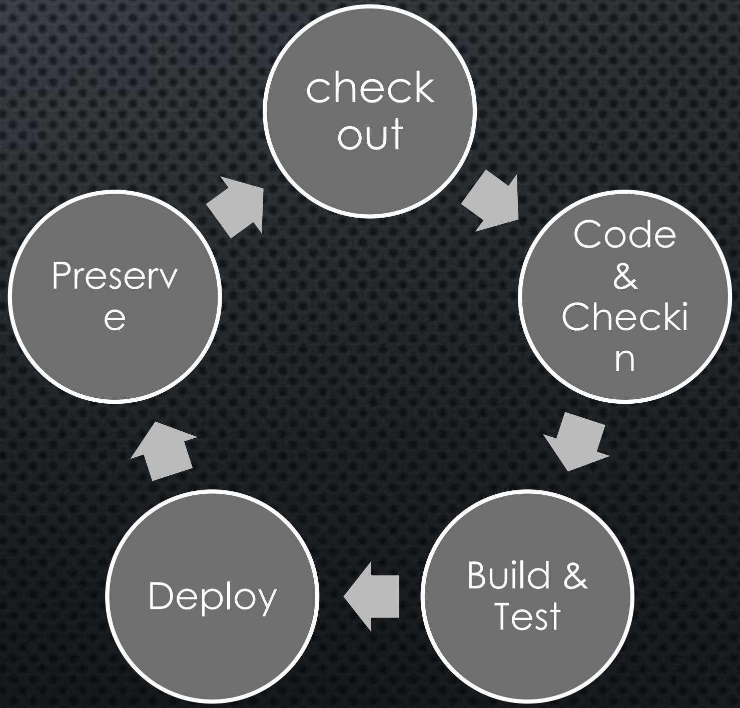
What – is a system that records changes to a file or set of files over time so that you can recall specific versions later.

Why – to control changes, to manage conflicts, covering up mistakes, view history, share the source code with other and to save the world.

Tools – RCS, CVS, SVN, VSS, TFS, IBM Clear Case, Perforce, Harvest, RTC, Bit keeper, GIT, Mercurial.

Key words –

Repository: A Secure store/collection of project artefacts
Commit: Save/Done/Final
Check-in: Update + Unlock + Sync with server
Check-out: Edit + Lock
Merge: Combine changes altogether
Branch: A Dummy workspace
Clone: First time
Pull: Getting additional changes from Server Repo
Push: Upload changes to Server Repo



SCM Git

What – Is a widely used source code management system for software development. It is a distributed revision control system with an emphasis on speed, data integrity, and support for distributed, non-linear workflows. Git was initially designed and developed in 2005 by Linux kernel developers (Linus Torvald & Co) for Linux kernel development.

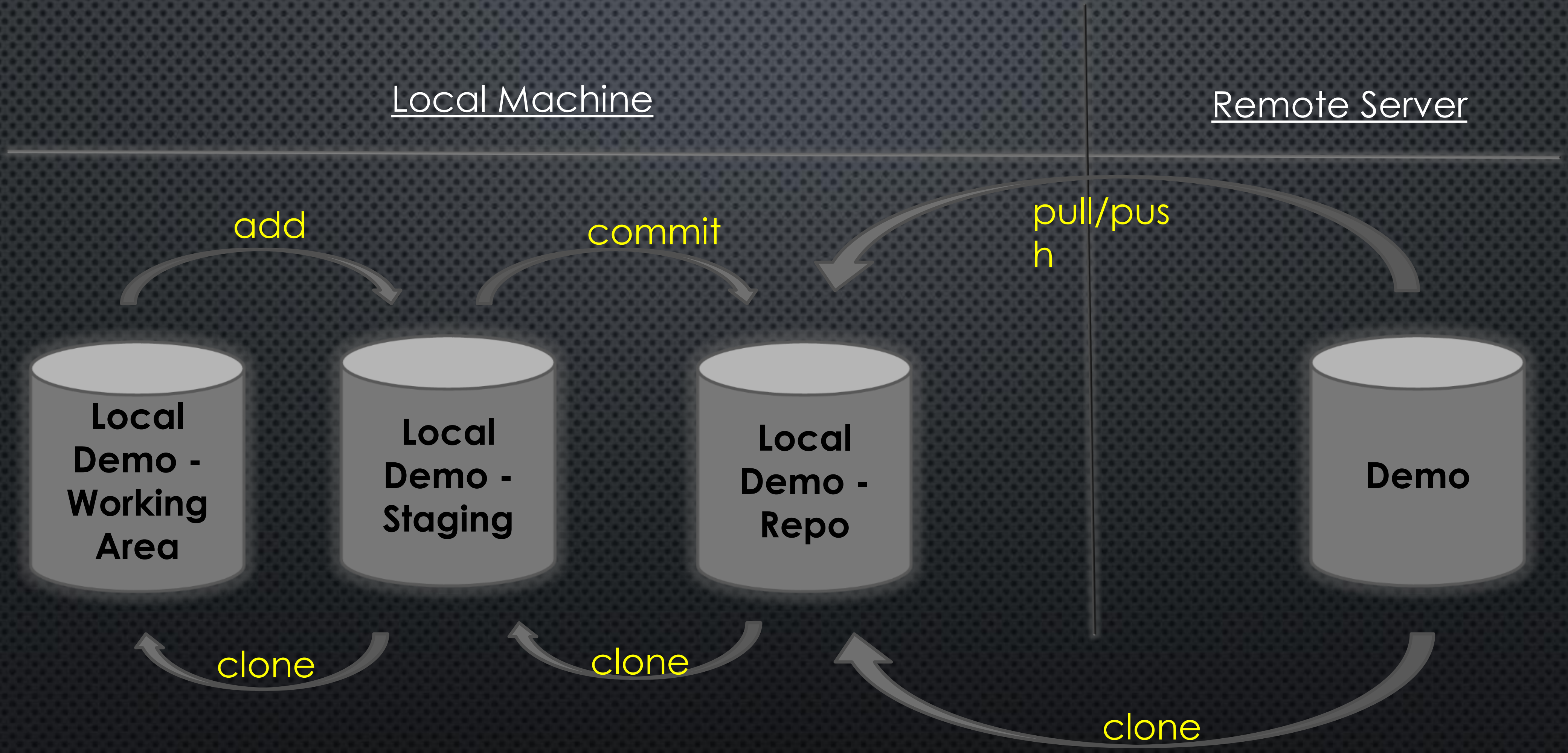
Why – Decentralized, Easy to handle (only if you are comfortable with CLI), Robust, Efficient in space management, support non-linear developments and more importantly open source.

Where – https://git-scm.com/download download based on your OS.

How – https://git-scm.com/book/en/v2/Getting-Started-Installing-Git follow the steps accordingly.

Hosted Services – https://github.com/ or https://bitbucket.org/

SCM Git


Local Machine
Remote Server
add
commit
pull/pus
h
Local Demo - Working Area
Local Demo - Staging
Local Demo - Repo
Demo
clone
clone
clone

Demo & Lab
Setup Git locally

SCM Git

Configs: Lets you configure your repositories and help generate unique commit IDs.

```
$ git config --global --edit
$ git config --global user.name <name>
$ git config --global user.email <email addr>
$ git config --system core.editor <vim/vi/nano>
$ git config --global alias.<alias name> [git command]
$ git config --global alias.unstage 'reset HEAD --'#unstage a file
$ git config – list
```

Initializing Repo:

```
$ git init <dir>  #--bare option is used to create remote repos preventing direct commits
$ git clone <repo> <dir>
```

.gitignore: list of files to be ignored from considering for commits.

```
# To add comments in ignore file.
*.log, *.txt, *.req files with these extensions will be skipped.
!abc.log this log will be exempted from skipping though all log files are added to ignore list
```

Remotes:

```
$ git remote add origin <https/ssh url>
$ git remote set-url origin <https/ssh url> #Amend existing URL
$ git remote rm origin #Deletes the link
$ git remote -v #Lists repos
```

SCM Git

Staging:

```
$ git add <file/folder name>
$ git add *.java
$ git add .
$ git reset HEAD <file name>  #Un-stage a file
```

Committing:

```
$ git commit
$ git commit -m
$ git commit --amend
$ git commit -a -m  #Commits without a staging
$ git blame <filename> #Who, when and what
```

Status:

```
$ git status
        M modified       File has been modified
        C copy-edit      File has been copied and modified
        R rename-edit    File has been renamed and modified
        A added          File has been added
        D deleted        File has been deleted
        U unmerged       File has conflicts after a merge
$ git status -s
```

SCM Git

Log:

```
$ git log # --online provides only heads
$ git log -p -2 #Shows last two commits
$ git log --stat  #Prints abbreviated stats
$ git log --pretty=oneline --max-count=2
$ git log --pretty=oneline --since=‘7 days ago'
$ git log --pretty=oneline --until='5 minutes ago'
$ git log --pretty=oneline --author=<your name>
$ git log --all --pretty=format:"%h %cd %s (%an)" --since='7 days ago‘
        %h is the abbreviated hash of the commit
        %d commit decorations (e.g. branch heads or tags)
        %ad is the commit date
        %s is the comment
        %an is the name of the author
        --graph tells git to display the commit tree in the form of an ASCII graph layout
        --date=short keeps the date format short and nice
$ git config --global alias.hist 'log --pretty=format:"%h %ad | %s%d [%an]" --graph --     date=short’
```

Diff:

```
$ git diff   #Shows what you changed, but haven't staged
$ git diff --cached  #Shows what has been staged, but not committed
```

SCM Git

Tag: Helps to mark files at important points in history

```
$ git tag  #Shows all tags, -l option lists the related tags
$ git tag -a <version/ref. no> -m  <comments> #keep file, but remove from staging area
$ git tag -a <version/ref. no> <HEAD> #Creates tag on commit
$ git tag  <version/ref. no>  #Creates lightweight tag
$ git push origin <tag name> #Push tag to origin repo
$ git push origin --tags #Push all tags to origin repo
$ git checkout -b <branch name> <tag name> #Switch to a new branch
```

Branch: A new work area with same code

```
$ git branch #Show all branches, -v option shows commits as well
$ git branch <name> #creates branch
$ git branch <option> <name> #-d delete branch, -D force delete, -m renames,
$ git branch --merged/--no-merged #shows merged/Non-merged branches.
$ git branch –d  #deletes the branch
```

Checkout:

```
$ git checkout <branch name>
$ git checkout -b <new branch name> <origin/branch> #-b New branch and checkout, take from repo
$ git checkout <lable> #For detached head checkouts.
```

Merge:

```
$ git merge <branch name>
$ git merge squash #get all commits but not history
```

SCM Git

Rebase: FF Merge. Moves a branch to a new commit.

```
$ git rebase master
```

Revert: undo changes made in that commit and makes a new commit

```
$ git revert <HEAD>
```

Reset: Eliminates previous commits and we never get them back.

```
$ git reset <filename> #removes from staging but working dir unchanged
$ git reset #revoke staging area to most recent commit, and working dir is intact
$ git reset --hard #resets both staging and working dir to match most recent commit
$ git reset <HEAD>  #rollback to that given commit
$ git reset --hard <HEAD> #resets both staging and working dir to the given commit state
```

(Re)move:

```
$ git rm <file>  #you must add the -f option in order to remove the file if committed
$ git rm --cached <file> #keep file, but remove from staging area
$ git mv <file1> <file2> #Renames a file
```

Clean: Removes untracked files from repo.

```
$ git clean -n #lists the files to be removed
$ git clean -f  #cleans untracked files, -d for directories
```

Lab & Demo
Setup Account in GitHub

SCM Git

Push: Push the changes to remote repo

```
$ git push origin
```

Pull: Fetches the changes from remote repo and merges it with local.

```
$ git pull origin
```

Clone: Downloads the repository to local.

```
$ git clone <repo>
```

Remote: Adding a remote repo URL

```
$ git remote –v  #Gives you the remote repos URL
$ git add remote <name> <URL>  #This will add remote repo besides origin
```

SCM Git/GitHub

Keywords:

Organization: Keeping group of repositories working to achieve a common goal, or, all repositories pertains one department.

Fork: Creating a similar repo in ones' Org. or personal account. He/she can freely make changing without impacting the main repo.

Pull Requests: Requesting to review and merge changes to a secured branch.

SSH Integration: Access repo without password through SSH protocol.

Issues: Create issues in case the code is having bugs.

Collaborators: Contributors to projects

Personal Access Tokens: Token to use in applications to connect to repo.

Hooks & Services: To add the integrations, like Jenkins.

SCM Git/GitHub

Industry Implementations:
- GitHub Enterprise
- Branching Strategy
- Developers Vs DevOps Engineers
- DR, HA, Failover
- Integrations with Jenkins

Interview Questions:
- Which version control are you familiar with
- What are organizations
- What branching strategy is followed in your previous Org.
- Do you have an experience migrating SVN/CVS to Git.
- How do you migrate a SVN repo to Git.
- What is Git Squash
- What re personal access tokens
- Can we clone a repo and push the changes to another repo? If yes, how do you do
- What is forking
- What is a pull request
- When merge conflict occurs and how do you resolve it

SCM Git/GitHub

Useful Links

- Interview Questions
  - https://www.edureka.co/blog/interview-questions/git-interview-questions/
  - https://career.guru99.com/top-40-interview-questions-on-git/

- Videos
  - https://www.youtube.com/watch?v=cEGIFZDyszA&list=PL6gx4Cwl9DGAKWClAD_iKpNC0bGHxGhcx
  - https://www.youtube.com/watch?v=0fKg7e37bQE
  - https://www.youtube.com/watch?v=rFQbYSvz7ms
  - https://www.youtube.com/watch?v=inzXL7IWwhM
  - https://www.youtube.com/watch?v=i5T-DB8tb4A
- Material
  - https://www.git-tower.com/blog/git-cheat-sheet
  - https://www.atlassian.com/git/tutorials/migrating-overview

MYTHRI

redhat Linux

Linux

ubuntu

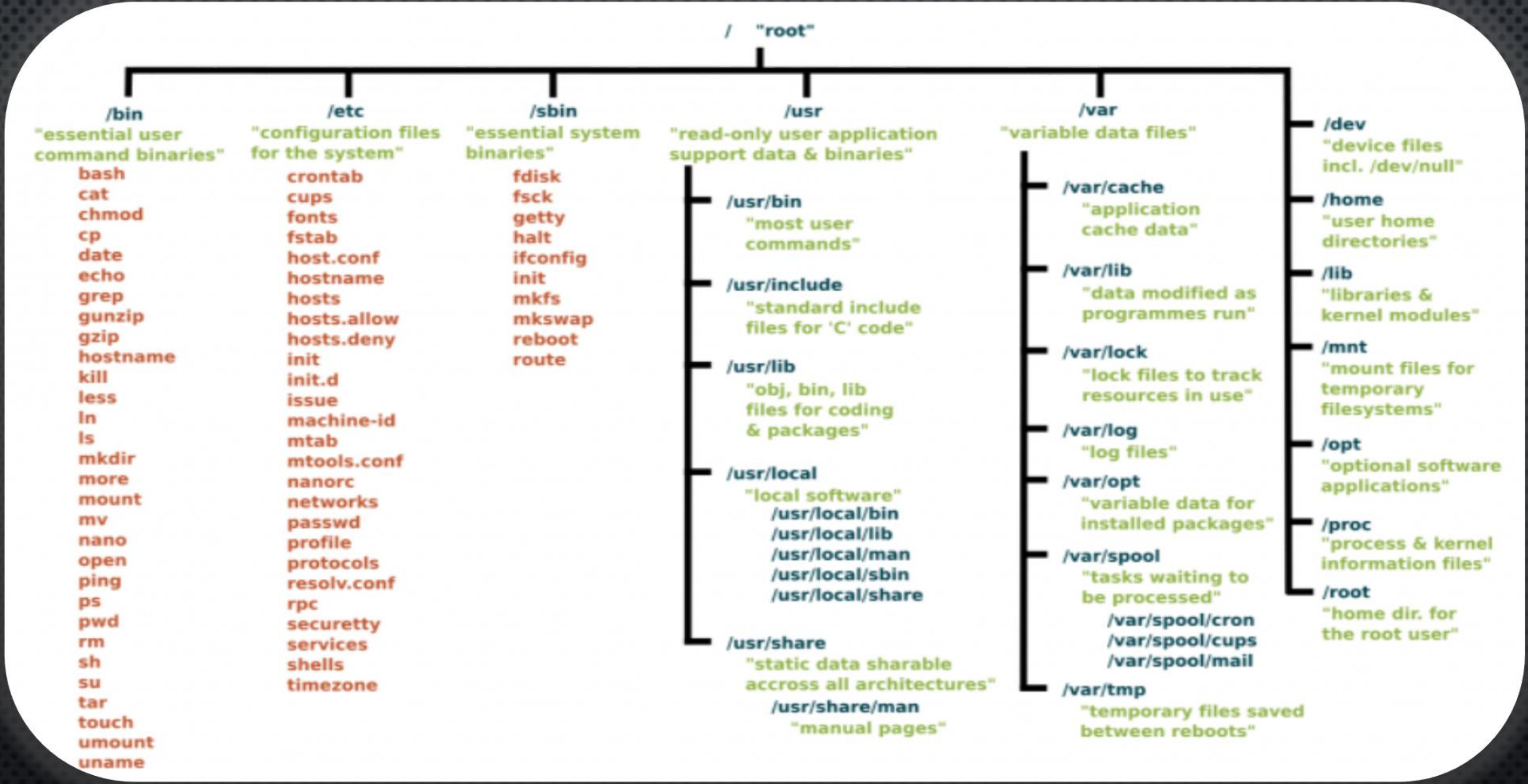
debian GNU/Linux

alpine Linux

Open AWS Account

OS – Linux Filesystem

Is an Operating System. Flavors: RHEL, CentOS, Debian, Ubuntu, Alpine etc



OS - Linux Filesystem

$ **ls –l** indicates all these types

- -rw-r--r-- ordinary file
- brw-rw---- block device file
- crw-rw-rw- character device file
- drwxr-xr-x directory file
- lrwxrwxrwx symbolic file
- srw-rw-rw- socket file

  prw-rw-rw- named pipe file

Directories: These are files that contains other files and directories, and provide pointers to them. (blue)

Symbolic links: These special files link to another file, in a different location. (Cyan)

Block and character device files: All physical devices in Linux are represented by device files. e.g. /dev/sda (yellow)

Socket file: Provides protected inter-process networking. (Purple)

Named Pipe file: Like socket files but doesn’t use network socket semantics. (Red)

Linux Commands - Lab

OS - Linux User Commands

File Processing:

- **ls** – directory listing
- **ls** -al – formatted listing with hidden files
- **cd** dir - change directory to dir
- **cd** – change to home
- **pwd** – show current directory
- **mkdir** dir – create a directory dir
- **rm** file – delete file
- **rm** -r dir – delete directory dir
- **rm** -f file – force remove file
- **rm** -rf dir – force remove directory dir *
- **cp** file1 file2 – copy file1 to file2
- **cp** -r dir1 dir2 – copy dir1 to dir2;
- **mv** file1 file2 – rename or move file1 to file2
- **ln** -s file link – create symbolic link link to file
- **touch** file – create or update file
- **cat** > file – places standard input into file
- **more** file – output the contents of file
- **head** file – output the first 10 lines of file
- **tail** file – output the last 10 lines of file
- **tail** -f file – output the contents of file as it grows
- **chmod** – changing the permissions of a file
- **chown** – Changing the ownership of a file
- **useradd** – Adding users
- **groupadd** – Adding groups

System Commands:

- **ps** – display your currently active processes
- **top** – display all running processes
- **kill** pid – kill process id pid
- **ssh** user@host – connect to host as user
- **ssh** -p port –i key user@host – connect to host on port
- **ssh-copy-id** user@host – add your key to host for user
- **grep** –i <pattern> pattern match
- **date** – show the current date and time
- **cal** – show this month's calendar uptime – show current
- **uptime** w – display who is online
- **uname** -a – show kernel information
- **cat** /proc/cpuinfo – cpu information
- **cat** /proc/meminfo – memory information
- **man** command – show the manual for command
- **df** – show disk usage
- **du** – show directory space usage
- **free** – show memory and swap usage
- **tar** –zcvf – create a tar file (-x extract)
- **Ping** – pinging the server
- **telnet** – check the port
- **Wget** & **curl** – download the files from net/url
- **Systemctl** – manage services
- **Yum** – package manager
- **Find** – find the files

CI Server - Jenkins

Jenkins

Continuous Integration, Testing and Deployment tool.

CloudBees vs Open Sources

Jobs – Free style vs Scripted Pipelines vs Declarative Pipelines

Dashboard – Classic vs Blue Ocean.

Master Vs Agents

Executors

Plugins

Type of Builds:

- Maven, npm, Gradle, Ms Build.

RBAC vs Matrix

Tools Configuration

Multibranch Pipelines / GitHub Projects

MYTHRI

Jenkins Jobs - Lab

Jenkins – Manage Jenkins Options

Configure: All Plugins and System related settings are configured here.

Configure Global Security: Authentication (Local or AD or LDAP), Authorization (Matrix or RBAC), jnlp and sshd ports are configured here.

Credentials: To configure credentials like username-password, secret text, secret file, certificates etc.

Tool Configuration: All installed software's (in master and agents) path or source configured here.

Reload Configuration From Disk: This will restart your Jenkins and includes latest changes that are made to config files.

Manage Plugins: One place to install/remove/upgrade plugins and to configure proxy.

System Info: Jenkins system information

System Log: All event's and activities' log is captured here. Most handy during troubleshooting of plugins or agents.

Load Statistics: Metrics of Jenkins

Jenkins CLI: Information on accessing your Jenkins through CLI.

Script Console: Execute Groovy scripts that affects Jenkins server. Used during troubleshooting or to get some info like which labels assigned to which nodes.

Jenkins – Manage Jenkins Options

Manage Nodes: Configuring Agents. It can be SSH or JNLP.

About Jenkins: Jenkins Configuration details.

Manage Old Data: In case Jenkins got upgraded and degraded, plugins got upgraded or degraded, sometimes the data stored on the disk is not compatible with the newer version, so the old format lies there for indefinite time, so it is recommended to manage this.

Manage Users: User Management.

In-Process Script Approvals: Few methods (classes) are not recommended by Jenkins due to security or they are deprecated or they might prone risk. When those such classes are used inside a pipeline, Jenkins put a requests here for explicit approval from Jenkins Admin. Until this request is approved, the class/method won't be executed.

Prepare for Shutdown: This option prevents newly triggered jobs from running, and allow current running jobs to process as it is. Used when you want to restart the Jenkins.

Jenkins - Tips

Industry Implementations:
- CloudBees Jenkins Platform (Master and Operation Center)
- CloudBees Jenkins Enterprise (Mesos or Kubernetes)
- Housekeeping
- DR, HA, Failover

Interview Questions:
- Which type of builds are you familiar with?
- Have you written Pipelines?
- How many jobs run every day and how do you manage failures?
- Explain typical pipeline that you are aware of or you developed in recent times?
- What is the most challenging task you have faced?
- Have you used shared libraries?
- Are you running Jenkins inside a container? Are you using ephemeral docker agents?
- How many executors you have?
- Explain your day to day activities?
- Have you created any build containers?
- What is a junit report, code coverage report and how do you genereate them?
- Tell me your approach on fixing a failed build?
- Provide me few plugins on which you worked recently?
- What is stage in pipeline?
- What are the Maven lifecycles and explain?
- What is Jenkinsfile, multibranch piprline, GitHub Org?
- What are the build triggers available in Jenkins?
- Have you integrated Jenkins with Git?

Jenkins

Useful Links

- Interview Questions
  - https://mindmajix.com/jenkins-interview-questions-answers
  - https://codingcompiler.com/jenkins-interview-questions-answers/
- Videos
  - https://www.youtube.com/watch?v=89yWXXIOisk&list=PLhW3qG5bs-L_ZCOA4zNPSoGbnVQ-rp_dG
  - https://www.youtube.com/watch?v=r6RtHI8Oj4Y
  - https://go.cloudbees.com/doc/index.html
- Material
  - https://jenkins.io/doc/book/pipeline/
  - https://jenkins.io/doc/
  - https://plugins.jenkins.io/

MYTHRI

amazon web services

Google Cloud

Cloud Technologies - AWS

IBM Cloud

Azure

AWS

Regions

Region Name	Region	Endpoint	Protocol
US East (Ohio)	us-east-2	rds.us-east-2.amazonaws.com	HTTPS
US East (N. Virginia)	us-east-1	rds.us-east-1.amazonaws.com	HTTPS
US West (N. California)	us-west-1	rds.us-west-1.amazonaws.com	HTTPS
US West (Oregon)	us-west-2	rds.us-west-2.amazonaws.com	HTTPS
Asia Pacific (Tokyo)	ap-northeast-1	rds.ap-northeast-1.amazonaws.com	HTTPS
Asia Pacific (Seoul)	ap-northeast-2	rds.ap-northeast-2.amazonaws.com	HTTPS
Asia Pacific (Osaka-Local)	ap-northeast-3	rds.ap-northeast-3.amazonaws.com	HTTPS
Asia Pacific (Mumbai)	ap-south-1	rds.ap-south-1.amazonaws.com	HTTPS
Asia Pacific (Singapore)	ap-southeast-1	rds.ap-southeast-1.amazonaws.com	HTTPS
Asia Pacific (Sydney)	ap-southeast-2	rds.ap-southeast-2.amazonaws.com	HTTPS
Canada (Central)	ca-central-1	rds.ca-central-1.amazonaws.com	HTTPS
China (Beijing)	cn-north-1	rds.cn-north-1.amazonaws.com.cn	HTTPS
China (Ningxia)	cn-northwest-1	rds.cn-northwest-1.amazonaws.com.cn	HTTPS
EU (Frankfurt)	eu-central-1	rds.eu-central-1.amazonaws.com	HTTPS
EU (Ireland)	eu-west-1	rds.eu-west-1.amazonaws.com	HTTPS
EU (London)	eu-west-2	rds.eu-west-2.amazonaws.com	HTTPS
EU (Paris)	eu-west-3	rds.eu-west-3.amazonaws.com	HTTPS
South America (São Paulo)	sa-east-1	rds.sa-east-1.amazonaws.com	HTTPS

AWS - Topics

Availability Zones
Cloud Services

- SaaS
- PaaS
- IaaS
- Private Cloud
- Hybrid Cloud

Direct Connect

Data Centers

AWS Console

AWS CLI

AWS Responsibilities

AWS Support model

AWS Free Tier and Usage limits

AWS Scaling Options

AWS Cost Model

AWS – Services (With Lab)

VPC (Virtual Private Cloud)

- CIDR
- Subnets (Private, Public)
- Routers
- Internet Gateway
- NAT Gateway
- Elastic Ips
- Peering
- NACLs
- Security Groups
- VPN

IAM (Identity Access Mgmt)

- Root Account
- Groups
- Users
  - Keys
- Roles
- Policies
- Password Management

EC2 (Elastic Compute Cloud)

- DashBoard
- Limits
- Instances
- Templates
- Requests (Spot, Reserved, Dedicated, Advance)
- AMIs
- Volumes
- Snapshots
- Placement Groups
- Managed Instances and Services
- Auto Scaling
  - Scale-in
  - Scale-out
  - Launch Config
  - Launch Group
- ELB (Elastic Load Balancer)

Cloud Watch

- Metrics
- Resource Monitoring
- Memory Monitoring
- Alerts
- Topics

AWS – Services (With Lab)

S3 (Simple Storage Service)

- Bucket Creation
- Versioning
- Lifecycle Mgmt
- Hosting Static Website
- Permissions
- Misc

Glacier

- Vaults
- Life Cycle Mgmt

EFS & EBS (Elastic File <Block> Storage)

- NAS
- SAN
- NFS Protocol

Storage GW

- Volumes
- Files
- Tape
- Caching

RDS (Relational Database Service)

- Primary/Secondary
- Backups
- Read replicas
- Events

Snow ball

- Transition

CloudFront

- Geo Location
- Misc

Route53

- DNS Servers
- DNS Mgmt
- Traffic Mgmt
- Availability Mgmt

DC (Direct Connect)

- Data Centers
- VPN Tunnels

AWS – Services (Theoretical)

ECS (Elastic Container Service)

EBStalk (Elastic BeanStalk)

EKS (Elastic Kubernetes Service)

Elastic Cache

Elastic Transcoder

Server Migration

Database Migration Service

Code Commit

Code Deploy

Code Pipeline

Config

Opswork

CloudFormation Templates

CloudTrail

Media Store

SES (Simple Email Service)

SQS (Simple Queue Service)

SNS (Simple Notification Service)

AWS

Industry Implementations:
- AWS Accounts
- AWS Organizations
- Roles and Policies
- Billings
- DR, HA, Failover

Interview Questions:
- What is VPC/Subnets/Security Groups/EC2/ECS/EBS/S3/ELB/Auto Scaling/etc?
- What is CIDR Block and how do you design a VPC?
- How do you create private subnet and public subnets?
- Why we create private subnets and what makes it different from Public?
- How do you configure Route tables?
- What is Internet Gateway?
- What is NAT Gateways?
- How do you provide Internet to machines in Private subnets?
- Have you used AWS CLI?
- Have you configured weightage load in Route53?
- What procedures do you follow to reduce the cost of Infrastructure?
- What is snowball and have you ever happened to use it?
- Have you migrated an on-premises instance to cloud?
- Did you happen to configure a cluster with EKS/ECS?
- What is Vertical and Horizontal scaling?
- What is the difference between Cloud Watch and Cloud Trail?
- What are read replicas?

AWS

Useful Links

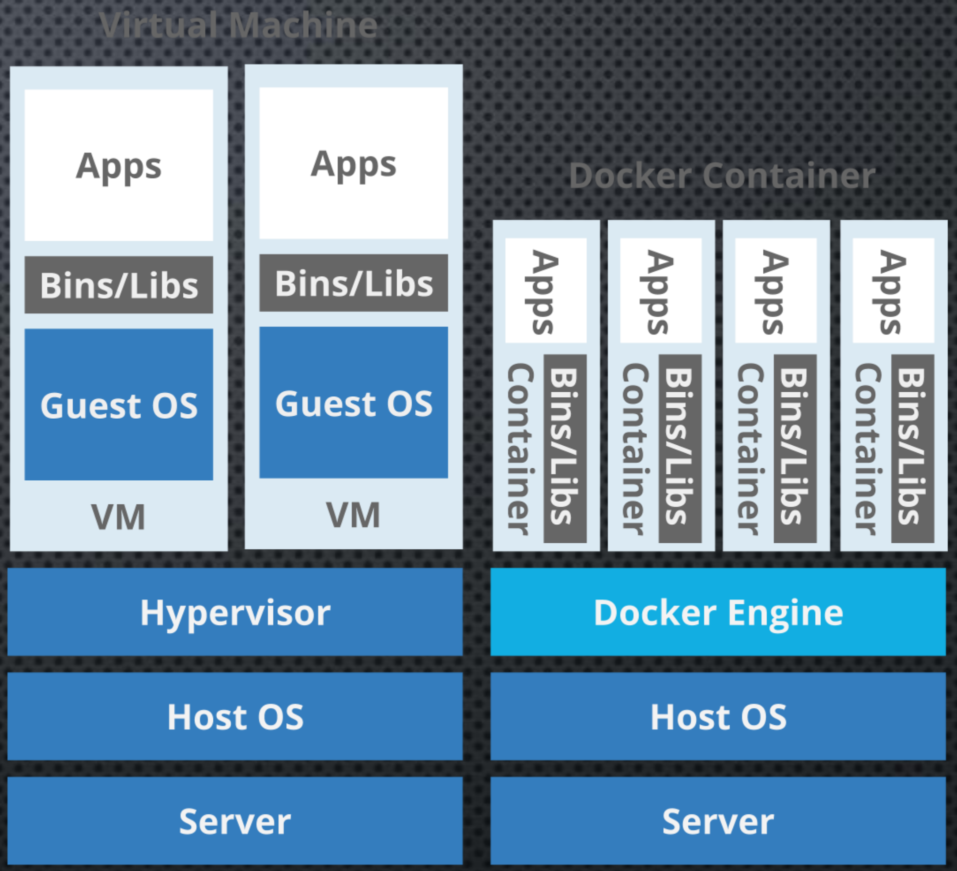
- Interview Questions
  - http://www.interviewquestionspdf.com/2016/02/top-30-aws-interview-questions-answers.html
  - https://www.wisdomjobs.com/e-university/amazon-web-services-aws-interview-questions-answers.html
- Videos
  - https://www.youtube.com/watch?v=ye7hgGZwSsY&list=PLJlqXVV4K5LXT8gkTJjHAgzZgW1K3se8U
- Material
  - https://aws.amazon.com/documentation/

Containers/Private Cloud-Docker

Docker

Is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package.

- History
- Docker Hub



Docker

Concepts:

Image: The basis of a Docker container. The required packages in one place.

Container: The image when running (serving its purpose)

Engine: Core part of Docker system. Provides networking, volumes, security, and executes the commands like run, build etc for containers.

Control Plane: When Engines are in cluster, this is one place to control your resources.

Registry: Place to store Docker images.

Dockerfile: Contains instructions to prepare an image.

Layers: Intermediate images that get created during image build for each step.

Volumes: Persistence Volume which can be accessed out side of the container.

Networking: bridge (default), host(uses host network), overlay(when multiple daemons are running, eg cluster), macvlan (to define a dedicated MAC address to each container)

Security: Containers run in isolation.

Build: To build the images

RUN: To run the images (start containers)

PUSH: To store images into registry

Docker-compose: To run multi-container application

Docker - Dockerfile

Command	Description
ADD	Copies a file from the host system onto the container
CMD	The command that runs when the container starts
ENTRYPOINT	Specifies the default app that you want to run (This is the way to configure a container that will run as an executable.)
ENV	Sets an environment variable in the new container
EXPOSE	Opens a port for linked containers
FROM	The base image to use in the build. This is mandatory and must be the first command in the file.
MAINTAINER	An optional value for the maintainer of the script
ONBUILD	A command that is triggered when the image in the Dcokerfile is used as a base for another image
RUN	Executes a command and save the result as a new layer
USER	Sets the default user within the container
VOLUME	Creates a shared volume that can be shared among containers or by the host machine
WORKDIR	Set the default working directory for the container

Docker - Lab

Docker

Commands:

docker build -t <imagename>:<tag> -f Dockerfile . # to build an image

docker run -d –p <hostport>:<containerport> <imagename>:<tag> #to run an image

docker exec –it <containerID> bash # to login to a running container

docker login -u <username> -p <password> <Registry>   #to login to registry

docker push <imagename>:<tag>   #to push the image to a registry

docker kill <containerID> #to kill the running container

docker ps   #to display the running processes, -a option shows all processes

docker inspect <containerID> #to get all the details like IP, resources, name, start time etc.

Misc:

Docker-compose (up, run and start)

Docker Swarm

Kubernetes

Microservices Architecture

Wordpress Setup: https://docs.docker.com/compose/wordpress/#define-the-project

Walk through: https://kubernetes.io/docs/tutorials/stateful-application/mysql-wordpress-persistent-volume/

Docker

Industry Implementations:

- Docker Daemon
- Docker Enterprise vs Community
- Docker Registries
- Orchestration Tools
- Jenkins Integration (Ephemeral Agents, CICD)
- Microservices Architecture

Interview Questions:

- Difference between VM and Container?
- What is Docker compose?
- How do you architect an application suiting Microservices Framework?
- What is a registry and how does it work?
- Walk me through a sample Dockerfile?
- What is the difference between ADD and COPY, CMD and ENTRYPOINT?
- How do you deploy applications into Kubernetes?
- How do you access your Kubernetes resources?
- What is a Pod?
- How do you externalize a service?
- What is an Ingress?
- What is Persistence Volumes?
- What are secrets?

Docker

Useful Links

- Interview Questions
  - https://tekslate.com/docker-interview-questions/
  - https://mindmajix.com/docker-interview-questions
  - https://www.javacodegeeks.com/2015/01/key-concepts-of-kubernetes.html

- Videos
  - https://www.youtube.com/watch?v=pGYAg7TMmp0&list=PLoYCgNOIyGAAzevEST2qm2Xbe3aeLFvLc&index=1
  - https://www.youtube.com/watch?v=DgoKjlDteEA&index=1&list=PLifxs-ivl_0Sq9Q7_tNPsYcratDh5sq34
  - https://www.youtube.com/watch?v=R-3dfURb2hA&list=PLbG4OyfwIxjFE5Ban_n2JdGad4EDWmisR
  - https://www.youtube.com/watch?v=R-3dfURb2hA&list=PLbG4OyfwIxjFE5Ban_n2JdGad4EDWmisR&index=1

- Material
  - https://docs.docker.com/
  - https://kubernetes.io/docs/home/?path=users&persona=app-developer&level=foundational

MYTHRI

Configuration Mgmt Tool - Ansible

puppet labs

CHEF

Ansible

Configuration Mgmt tools help maintain consistency across all machines' attributes and properties.

Current State: Current status of the configurations

Desired State: How it is expected to be.

Known Tools:

- Chef (Client, Server and Workstation)
- Puppet (Master and Agent)
- Ansible (Agentless)

Installation on AWS:

- sudo yum-config-manager --enable epel
- sudo yum install ansible

Ansible

Topics

- SSH Connectivity
- Host file
- Config file
- Ansible playbook
- Ansible Community
- Modules
- Galaxy roles
- Vault
- Role Structure
- Ansible Tower

Role Structure:

```
[ec2-user@ip-172-31-18-17 test]$ tree
.
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── README.md
├── tasks
│   └── main.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml
```

Ansible - Lab

Ansible

Commands:

```
ansible --list-hosts web* #to list the hosts matching web
ansible all -a "ls -latr" #to run a command on hosts
ansible-playbook <filename.yml> #to execute the tasks
ansible-playbook <filename.yml> --limit <groupname> #to run tasks on selected machines.
ansible-playbook <filename.yml> -i <hostfile>
ansible-galaxy init <rolename> #to initialize role
ansible-vault create/edit <filename>
```

Ansible

Industry Implementations:

- Control Machine & Access
- Agents and Integration with Jenkins
- Ansible Tower
- Roles Creation and Maintenance
- Security the playbooks and roles
- Requirements.yml

Interview Questions:

- What is config.Mgmt and why we use it?
- What are Galaxy Roles?
- How do you create a secret variable in Ansible?
- What is a host file?
- Have you written any custom modules?

Ansible

Useful Links

- Interview Questions
    - https://tekslate.com/ansible-interview-questions
    - https://www.edureka.co/blog/interview-questions/chef-interview-questions/
    - https://www.edureka.co/blog/interview-questions/puppet-interview-questions/

- Videos
    - https://www.youtube.com/watch?v=tJVkERqw8SI&list=PLLsor6GJ_BEEC9jUSc760iqaOx6u5lqRA
    - https://www.youtube.com/watch?v=w216Ynm9HxY

- Material
    - https://docs.ansible.com/
    - https://galaxy.ansible.com/home
    - https://serversforhackers.com/c/an-ansible-tutorial