DevOps Commands Cheat Sheet

Basic Linux Commands -

Linux is the foundation of DevOps operations - it's like a Swiss Army knife for servers. These commands help you navigate systems, manage files, configure permissions, and automate tasks in terminal environments.

1. **pwd** - Print the current working directory.
2. **ls** - List files and directories.
3. **cd** - Change directory.
4. **touch** - Create an empty file.
5. **mkdir** - Create a new directory.
6. **rm** - Remove files or directories.
7. **rmdir** - Remove empty directories.
8. **cp** - Copy files or directories.
9. **mv** - Move or rename files and directories.
10. **cat** - Display the content of a file.
11. **echo** - Display a line of text.
12. **clear** - Clear the terminal screen.

Intermediate Linux Commands

13. **chmod** - Change file permissions.
14. **chown** - Change file ownership.
15. **find** - Search for files and directories.
16. **grep** - Search for text in a file.
17. **wc** - Count lines, words, and characters in a file.
18. **head** - Display the first few lines of a file.
19. **tail** - Display the last few lines of a file.
20. **sort** - Sort the contents of a file.
21. **uniq** - Remove duplicate lines from a file.
22. **diff** - Compare two files line by line.
23. **tar** - Archive files into a tarball.
24. **zip/unzip** - Compress and extract ZIP files.
25. **df** - Display disk space usage.
26. **du** - Display directory size.
27. **top** - Monitor system processes in real time.
28. **ps** - Display active processes.
29. **kill** - Terminate a process by its PID.
30. **ping** - Check network connectivity.
31. **wget** - Download files from the internet.
32. **curl** - Transfer data from or to a server.
33. **scp** - Securely copy files between systems.
34. **rsync** - Synchronize files and directories.

Learn 2025 DevOps/SRE tech stack www.heydevops.in
Our Placements https://heydevops.in/success-stories
More Project Videos https://www.youtube.com/praveensingampalli
Connect with me https://www.instagram.com/singam4devops/

ULTIMATE CHEAT SHEETS / ARCHITECTURE DESIGNS / KUBERNETES NOTES

Git Cheat Sheet

Setup

Set the name and email that will be attached to your commits and tags

```
$ git config --global user.name "Danny Adams"
$ git config --global user.email "my-email@gmail.com"
```

Start a Project

Create a local repo (omit <directory> to initialise the current directory as a git repo

```
$ git init <directory>
```

Download a remote repo

```
$ git clone <url>
```

Make a Change

Add a file to staging

```
$ git add <file>
```

Stage all files

```
$ git add .
```

Commit all staged files to git

```
$ git commit -m "commit message"
```

Add all changes made to tracked files & commit

```
$ git commit -am "commit message"
```

Basic Concepts

main: default development branch
origin: default upstream repo
HEAD: current branch
HEAD^: parent of HEAD
HEAD~4: great-great grandparent of HEAD

By @DoubleDanny

Branches

List all local branches. Add -r flag to show all remote branches, -a flag for all branches.

```
$ git branch
```

Create a new branch

```
$ git branch <new-branch>
```

Switch to a branch & update the working directory

```
$ git checkout <branch>
```

Create a new branch and switch to it

```
$ git checkout -b <new-branch>
```

Delete a merged branch

```
$ git branch -d <branch>
```

Delete a branch, whether merged or not

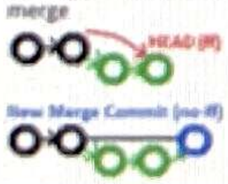
```
$ git branch -D <branch>
```

Add a tag to current commit (often used for new version releases)

```
$ git tag <tag-name>
```

Merging

Merge branch a into branch b. Add --no-ff option for no-fast-forward merge



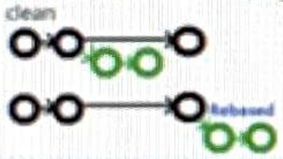

```
$ git checkout b
$ git merge a
```

Merge & squash all commits into one new commit

```
$ git merge --squash a
```

Rebasing

Rebase feature branch onto main (to incorporate new changes made to main). Prevents unnecessary merge commits into feature, keeping history clean




```
$ git checkout feature
$ git rebase main
```

Interatively clean up a branches commits before rebasing onto main

```
$ git rebase -i main
```

Interatively rebase the last 3 commits on current branch

```
$ git rebase -i Head~3
```

Undoing Things

Move (&/or rename) a file & stage move

```
$ git mv <existing_path> <new_path>
```

Remove a file from working directory & staging area, then stage the removal

```
$ git rm <file>
```

Remove from staging area only

```
$ git rm --cached <file>
```

View a previous commit (READ only)

```
$ git checkout <commit_ID>
```

Create a new commit, reverting the changes from a specified commit

```
$ git revert <commit_ID>
```

Go back to a previous commit & delete all commits ahead of it (revert is safer). Add --hard flag to also delete workspace changes (BE VERY CAREFUL)

```
$ git reset <commit_ID>
```

Review your Repo

List new or modified files not yet committed

```
$ git status
```

List commit history, with respective IDs

```
$ git log --oneline
```

Show changes to unstaged files. For changes to staged files, add --cached option

```
$ git diff
```

Show changes between two commits

```
$ git diff commit1_ID commit2_ID
```

Stashing

Store modified & staged changes. To include untracked files, add -u flag. For untracked & ignored files, add -a flag

```
$ git stash
```

As above, but add a comment

```
$ git stash save "comment"
```

Partial stash. Stash just a single file, a collection of files, or individual changes from within files

```
$ git stash -p
```

List all stashes

```
$ git stash list
```

Re-apply the stash without deleting it

```
$ git stash apply
```

Re-apply the stash at index 2, then delete it from the stash list. Omit stash@{n} to pop the most recent stash.

```
$ git stash pop stash@{2}
```

Show the diff summary of stash 1. Pass the -p flag to see the full diff.

```
$ git stash show stash@{1}
```

Delete stash at index 1. Omit stash@{n} to delete last stash made

```
$ git stash drop stash@{1}
```

Delete all stashes

```
$ git stash clear
```

Synchronizing

Add a remote repo

```
$ git remote add <alias> <url>
```

View all remote connections. Add -v flag to view urls.

```
$ git remote
```

Remove a connection

```
$ git remote remove <alias>
```

Rename a connection

```
$ git remote rename <old> <new>
```

Fetch all branches from remote repo (no merge)

```
$ git fetch <alias>
```

Fetch a specific branch

```
$ git fetch <alias> <branch>
```

Fetch the remote repo's copy of the current branch, then merge

```
$ git pull
```

Move (rebase) your local changes onto the top of new changes made to the remote repo (for clean, linear history)

```
$ git pull --rebase <alias>
```

Upload local content to remote repo

```
$ git push <alias>
```

Upload to a branch (can then pull request)

```
$ git push <alias> <branch>
```

Praveen Singampalli

9147890986 heydevopshelp@gmail.com linkedin.com/in/praveen-singampalli/

Professional summary

I am a dedicated DevOps/SRE Engineer with hands-on experience in automation, infrastructure management, and enhancing software development practices. With a strong background in CI/CD pipelines and cloud infrastructure, I have significantly improved operational efficiency and system reliability. My work in integrating monitoring solutions and securing DevOps practices has led to substantial reductions in incidents and enhanced performance across projects.

Technical Skills

DevOps Tools : Python, Java, Jenkins , Ansible, Kubernetes , Docker, Maven, Jfrog, Terraform,

Visualization Tools: Prometheus, Grafana , Istio, Kiali, Jaeger, New relic

Other Skills: Automations, Production Support, RCA, Oncall activities

Experience/Projects

Walmart **July 2021 – December 2023**

DevOps/SRE Engineer Bengaluru, India

- **Infrastructure Automation**: Designed and implemented **CI/CD pipelines** using **Jenkins**, **GitLab CI**, and **CircleCI** to automate deployment and provisioning of infrastructure in cloud environments, improving deployment speed by 40%.
- Managed cloud infrastructure on **Google Cloud Platform (GCP)**, automating provisioning with **Terraform** and **CloudFormation**. Reduced manual configuration errors and increased scalability by 30%.
- Developed automated infrastructure provisioning workflows using **Terraform** and **Ansible**, enabling seamless setup of virtual machines, databases in **GCP** and **AWS** with 70% automated efforts
- Led the integration of **monitoring** and **alerting** tools like **Prometheus**, **Grafana**, and **New Relic**, ensuring high availability and early detection of issues across cloud applications and infrastructure which helped in reduction of 30-50 support tickets.
- Collaborated with **development teams** to optimize the **software development lifecycle (SDLC)**, reducing bottlenecks by automating testing, deployments, and environment provisioning.
- Ensured **secure DevOps practices**, including **secrets management** using tools like **Vault**, **vulnerability scanning**, and implementing robust **access control** policies, resulting in improved security posture across cloud environments.
- Managed incident response and troubleshooting for production systems, ensuring a **high uptime** of 99% and fast resolution of issues in collaboration with engineering teams, minimizing service interruptions

Verizon **June 2019 – July 2021**

DevOps Engineer Kolkata, India

- Implemented **configuration management** solutions using **Ansible** and **Puppet** to automate the provisioning and configuration of infrastructure, reducing manual configuration tasks by 50%.
- Developed and managed **monitoring dashboards** using Grafana and Prometheus, improving visibility into system health and ensuring optimal performance of applications deployed on **AWS and GCP**.
- Worked closely with development teams to continuously improve CI/CD pipelines, facilitating smoother releases and addressing bottlenecks by automating infrastructure provisioning and testing.
- Spearheaded the use of cloud automation tools like terraform to ensure consistent infrastructure deployment, aligning with best practices for **infrastructure as Code**.
- Provided hands-on support for incident management, ensuring timely resolution of **production issues and collaborating** with teams to prevent recurring incidents, achieving a 20% improvement in system reliability.

Personal Projects

Walmart: Chatbot Automation | Nodejs, DevOps / SRE **April 2022 - May 2022**

- Developed an interactive chatbot for supply chain, enhancing strategic decision-making with dynamic visualizations that **reduced data trend identification time by 40% and improved decision accuracy by 30%.**

Git Cheat Sheet

GIT BASICS

`git init <directory>`	Create empty Git repo in specified directory. Run with no arguments to initialize the current directory as a git repository.
`git clone <repo>`	Clone repo located at `<repo>` onto local machine. Original repo can be located on the local filesystem or on a remote machine via HTTP or SSH.
`git config user.name <name>`	Define author name to be used for all commits in current repo. Devs commonly use `--global` flag to set config options for current user.
`git add <directory>`	Stage all changes in `<directory>` for the next commit. Replace `<directory>` with a `<file>` to change a specific file.
`git commit -m "<message>"`	Commit the staged snapshot, but instead of launching a text editor, use `<message>` as the commit message.
`git status`	List which files are staged, unstaged, and untracked.
`git log`	Display the entire commit history using the default format. For customization see additional options.
`git diff`	Show unstaged changes between your index and working directory.

UNDOING CHANGES

`git revert <commit>`	Create new commit that undoes all of the changes made in `<commit>`, then apply it to the current branch.
`git reset <file>`	Remove `<file>` from the staging area, but leave the working directory unchanged. This unstages a file without overwriting any changes.
`git clean -n`	Shows which files would be removed from working directory. Use the `-f` flag in place of the `-n` flag to execute the clean.

REWRITING GIT HISTORY

`git commit --amend`	Replace the last commit with the staged changes and last commit combined. Use with nothing staged to edit the last commit's message.
`git rebase <base>`	Rebase the current branch onto `<base>`. `<base>` can be a commit ID, branch name, a tag, or a relative reference to HEAD.
`git reflog`	Show a log of changes to the local repository's HEAD. Add `--relative-date` flag to show date info or `--all` to show all refs.

GIT BRANCHES

`git branch`	List all of the branches in your repo. Add a `<branch>` argument to create a new branch with the name `<branch>`.
`git checkout -b <branch>`	Create and check out a new branch named `<branch>`. Drop the `-b` flag to checkout an existing branch.
`git merge <branch>`	Merge `<branch>` into the current branch.

REMOTE REPOSITORIES

`git remote add <name> <url>`	Create a new connection to a remote repo. After adding a remote, you can use `<name>` as a shortcut for `<url>` in other commands.
`git fetch <remote> <branch>`	Fetches a specific `<branch>`, from the repo. Leave off `<branch>` to fetch all remote refs.
`git pull <remote>`	Fetch the specified remote's copy of current branch and immediately merge it into the local copy.
`git push <remote> <branch>`	Push the branch to `<remote>`, along with necessary commits and objects. Creates named branch in the remote repo if it doesn't exist.

What is CI/CD? Continuous Integration & Continuous Delivery

What is Continuous Integration?

Continuous Integration is a software development method where team members integrate their work at least once a day. In this method, every integration is checked by an automated build to detect errors. This concept was first introduced over two decades ago to avoid "integration hell," which happens when integration is put off till the end of a project.

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

In this CI tutorial, you will learn:

Development without CI vs. Development with CI

Here are key differences between development using CI or without CI.

Development without CI	Development with CI
Lots of Bugs	Fewer bugs

DevOps Interview Question

1. What is Source Code Management?

It is a process through which we can store and manage any code. Developers write code, Testers write test cases and DevOps engineers write scripts. This code, we can store and manage in Source Code Management. Different teams can store code simultaneously. It saves all changes separately. We can retrieve this code at any point of time.

---

2. What are the Advantages of Source Code Management?

. Helps in Achieving teamwork
. Can work on different features simultaneously
. Acts like pipeline b/w offshore & onshore teams
. Track changes (Minute level)
. Different people from the same team, as well as different teams, can store code simultaneously (Save all changes separately)

---

3. Available Source Code Management tools in the market?

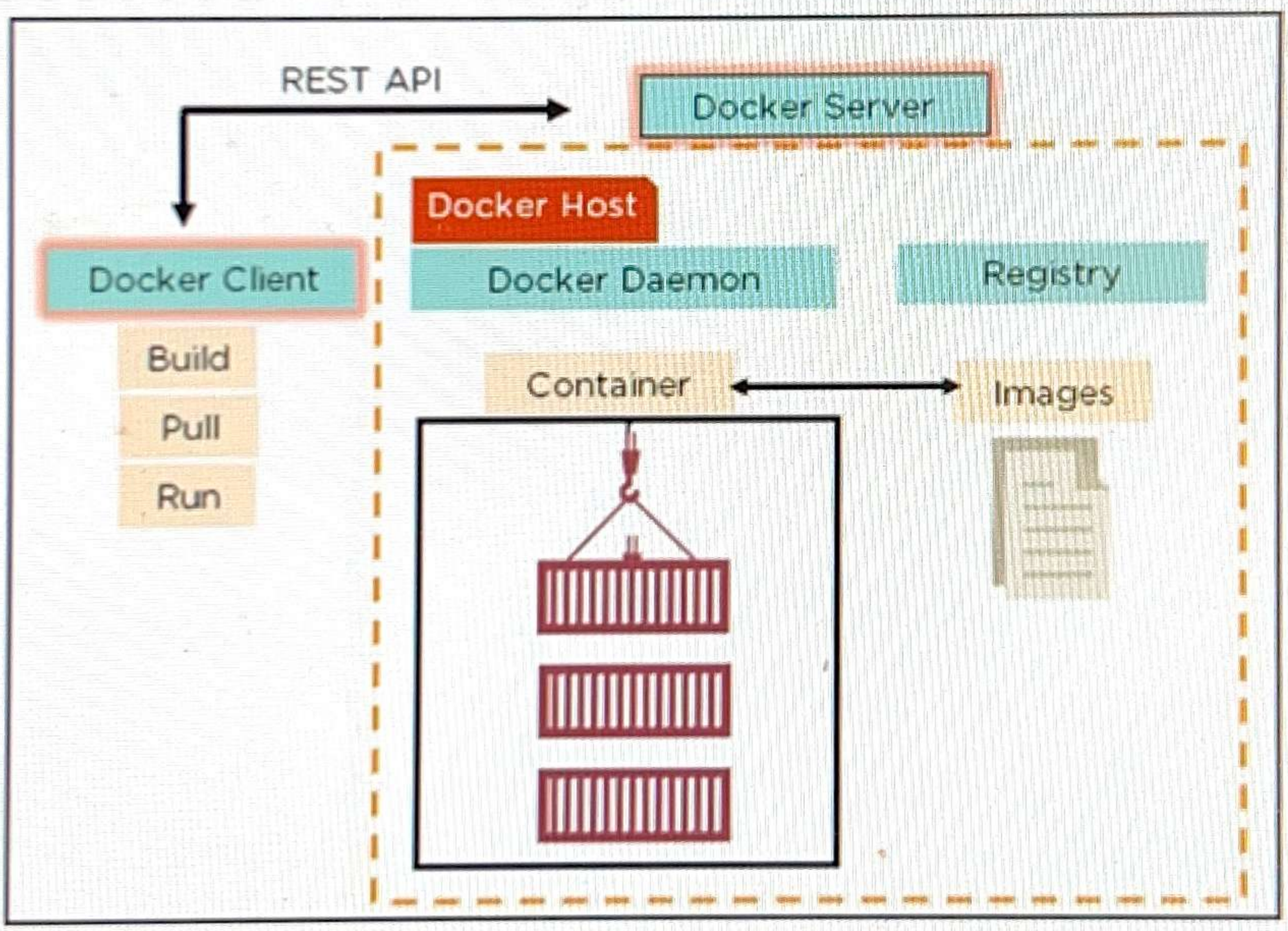
There are so many Source Code Management tools available in the market. Those are

. Git
. SVN
. Perforce
. Clear case

Out of all these tools, Git is the most advanced tool in the market where we are getting so many advantages compared to other Source Code Management tools.

DOCKER INTERVIEW QUESTIONS

1. **Explain the architecture of Docker**



- Docker uses a client-server architecture
- Docker Client is a service which runs a command. The command is translated using REST API and is sent to the Docker Daemon (server)
- Docker Daemon accepts the request and interacts with the operating system in order to build Docker Images and run Docker containers
- A Docker Image is a template of instruction which is used to create containers

Kubernetes For Everyone

Kubernetes introduction and features

How Kubernetes works?

In Kubernetes, there is a master node and multiple worker nodes, each worker node can handle multiple pods.
Pods are just a bunch of containers clustered together as a working unit. You can start designing your applications using pods.
Once your pods are ready, you can specify pod definitions to the master node, and how many you want to deploy. From this point, Kubernetes is in control.
It takes the pods and deploys them to the worker nods. If a worker node goes down, Kubernetes starts new pods on a functioning worker node.
This makes the process of managing the containers easy and simple.
It makes it easy to build and add more features and improving the application to attain higher customer satisfaction.

Finally, no matter what technology you're invested in, Kubernetes can help you.

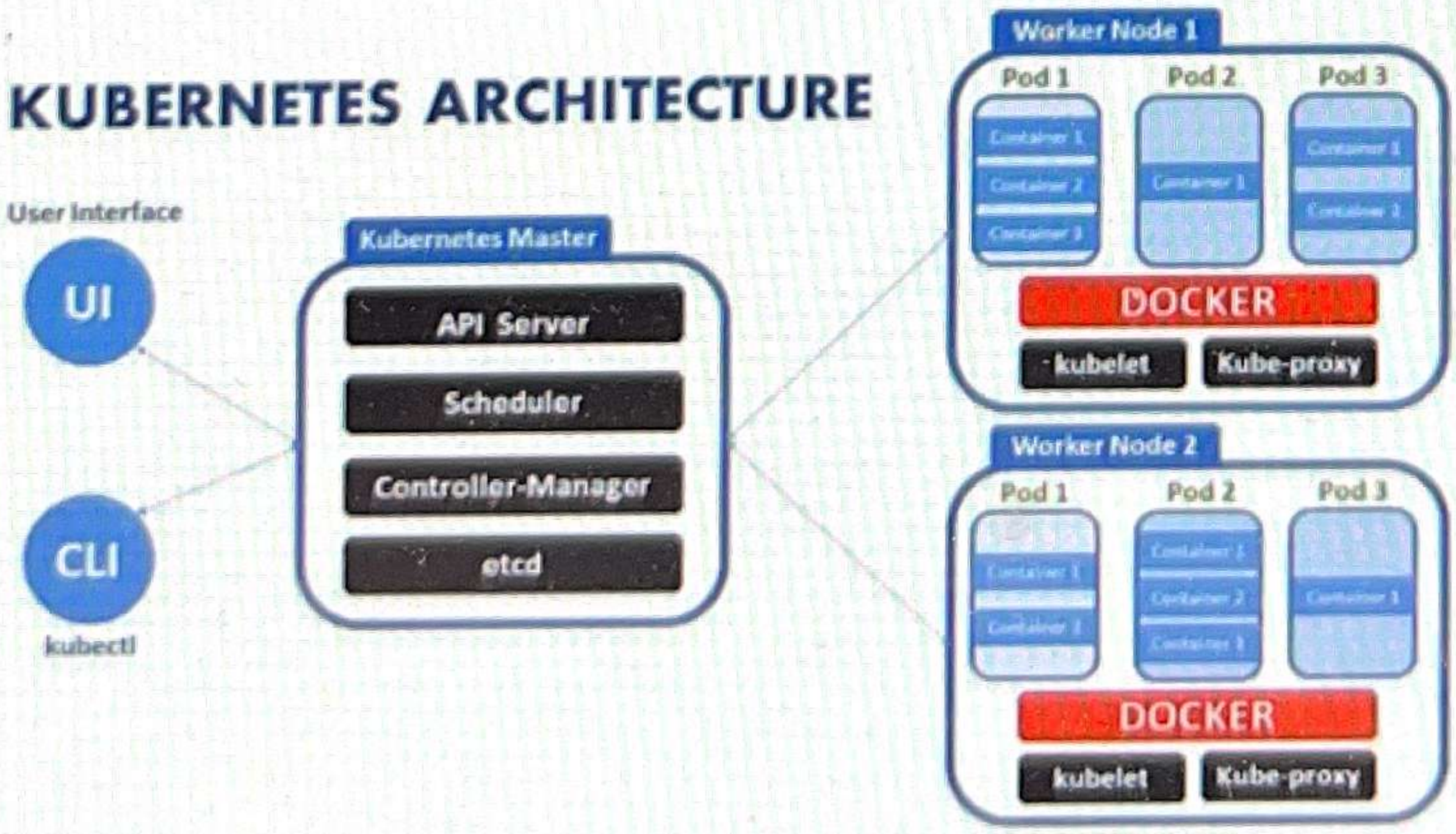


Image credits: Source: Knoldus Inc

1. What is Kubernetes?
   Kubernetes Is a tool used for managing containers. it manages clusters of container hosts and the containers which are enclosed in a pod that is managed bt Kubernetes.
2. What are the benefits of using Kubernetes?
   Monitoring-kubernetes monitor the availability of the container in every millisecond and whenever the OS/Container goes down it will launch the identical container and serve the requests coming.
   It's portable and 100% open source.
   Auto-Scaling-As load of the container increase the k8s launch more containers to handle the client request and as the load decrease the k8s shutdown the containers which causing less resource utilization
3. On which architecture does Kubernetes work?
   Kubernetes work on Master-Slaves Architecture. There is a node also called controller node on which k8s is running is called a master node and the container are running on the other node called slaves node.
4. What is a multinode cluster and single-node cluster in Kubernetes?
   When the controller program and the container node are running on the same machine or on the same operating system then it is called a single-node cluster MultiNode-Cluster-when the controller program of k8s in running on the one machine (master-node) and the container is running on the different container host machine (worker-node).
5. What do you mean by a pod in Kubernetes?
   pod is the smallest deployable unit of computing you can create and manage in k8s.
6. What is minikube?
   Minikube is a program or software that helps to setup kubernetes or K8s
7. How to install minikube and kubectl?
   For minikube, go to google and download the software and then double click the exe file to get minikube installed.
   Navigate to the Directory where Minikube is installed
   cmd> cd C:\Program Files\Kubernetes\Minikube
   and then run the command "minikube.exe start --driver=virtualbox --kubernetes-version=v1.20.0", so that it starts creating minikube vm in virtual box which in turn acts as a k8s server.
8. Whereas, for kubectl, you need to download the program file using curl -lo command and then it will get installed.
   curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.20.0/bin/windows/amd64/kubectl.exe

TOP 250+ Interviews Questions on AWS

Q1) What is AWS?

Answer:AWS stands for Amazon Web Services. AWS is a platform that provides on-demand resources for hosting web services, storage, networking, databases and other resources over the internet with a pay-as-you-go pricing.

Q2) What are the components of AWS?

Answer:EC2 – Elastic Compute Cloud, S3 – Simple Storage Service, Route53, EBS – Elastic Block Store, Cloudwatch, Key-Paris are few of the components of AWS.

Q3) What are key-pairs?

Answer:Key-pairs are secure login information for your instances/virtual machines. To connect to the instances we use key-pairs that contain a public-key and private-key.

Q4) What is S3?

Answer:S3 stands for Simple Storage Service. It is a storage service that provides an interface that you can use to store any amount of data, at any time, from anywhere in the world. With S3 you pay only for what you use and the payment model is pay-as-you-go.

Q5) What are the pricing models for EC2instances?

Answer:The different pricing model for EC2 instances are as below,

- On-demand
- Reserved
- Spot
- Scheduled
- Dedicated

Q6) What are the types of volumes for EC2 instances?

Terraform Walkthrough

What is IaC?

Infrastructure as code (IaC) means to manage your IT infrastructure using configuration files.

Why IaC?

Historically, managing IT infrastructure was a manual process. People would physically put servers in place and configure them. Only after the machines were configured to the correct settings required by the OS and dependencies would those people deploy the application.

Businesses are making a transition where traditionally-managed infrastructure can no longer meet the demands of today's businesses. IT organizations are quickly adopting the public cloud, which is predominantly API-driven.

To meet customer demands and save costs, application teams are architecting their applications to support a much higher level of elasticity, supporting technology like containers and public cloud resources. These resources may only live for a matter of hours, therefore the traditional method of raising a ticket to request resources is no longer a viable option.

Benefits of IaC

Speed

IaC benefits a company's IT architecture and workflow as it uses automation to substantially increase the provisioning speed of the infrastructure' s development, testing, and production.

Consistency

Since it is code, it generates the same result every time. It provisioned the same environment every time, enabling improved infrastructure consistency at all times.

Cost

One of the main benefits of IaC is, without a doubt, lowering the costs of infrastructure management. With everything automated and organized, engineers save up on time and cost which can be wisely invested in performing other manual tasks and higher-value jobs.

Minimum Risk

IaC allows server configuration that can be documented, logged, and tracked later for reference. Configuration files will be reviewed by a person or policy as a code (sentinel) for security leakages.

Everything Codified

The main benefit of IaC is explicit coding to configure files in use. You can share codes with the team, test them to ensure accuracy, maintain uniformity and update your infrastructure into the same flow of IaC.

Version Controlled, Integrated

Since the infrastructure configurations are codified, we can check-in into version control like GitHub and start versioning it.

IaC allows you to track and give insight on what, who, when, and why anything changed in the process of deployment. This has more transparency which we lack in traditional infrastructure management.

Now, we know what is Infrastructure as Code means, now let's deep dive into Terraform...

Terraform

Terraform is a tool for building, changing, and versioning infrastructure safely and efficiently. Terraform can manage existing and popular service providers as well as custom in-house solutions.

Terraform Walkthrough

What is IaC?

Infrastructure as code (IaC) means to manage your IT infrastructure using configuration files.

Why IaC?

Historically, managing IT infrastructure was a manual process. People would physically put servers in place and configure them. Only after the machines were configured to the correct settings required by the OS and dependencies would those people deploy the application.

Businesses are making a transition where traditionally-managed infrastructure can no longer meet the demands of today's businesses. IT organizations are quickly adopting the public cloud, which is predominantly API-driven.

To meet customer demands and save costs, application teams are architecting their applications to support a much higher level of elasticity, supporting technology like containers and public cloud resources. These resources may only live for a matter of hours; therefore the traditional method of raising a ticket to request resources is no longer a viable option.

Benefits of IaC

Speed

IaC benefits a company's IT architecture and workflow as it uses automation to substantially increase the provisioning speed of the infrastructure's development, testing, and production.

Consistency

Since it is code, it generates the same result every time. It provisioned the same environment every time, enabling improved infrastructure consistency at all times.

Cost

One of the main benefits of IaC is, without a doubt, lowering the costs of infrastructure management. With everything automated and organized, engineers save up on time and cost which can be wisely invested in performing other manual tasks and higher-value jobs.

Minimum Risk

IaC allows server configuration that can be documented, logged, and tracked later for reference. Configuration files will be reviewed by a person or policy as a code (sentinel) for security leakages.

Everything Codified

The main benefit of IaC is explicit coding to configure files in use. You can share codes with the team, test them to ensure accuracy, maintain uniformity and update your infrastructure into the same flow of IaC.

Version Controlled, Integrated

Since the infrastructure configurations are codified, we can check-in into version control like GitHub and start versioning it.

IaC allows you to track and give insight on what, who, when, and why anything changed in the process of deployment. This has more transparency which we lack in traditional infrastructure management.

Terraform Walkthrough

What is IaC?

Infrastructure as code (IaC) means to manage your IT infrastructure using configuration files.

Why IaC?

Historically, managing IT infrastructure was a manual process. People would physically put servers in place and configure them. Only after the machines were configured to the correct settings required by the OS and dependencies would those people deploy the application.

Businesses are making a transition where traditionally-managed infrastructure can no longer meet the demands of today's businesses. IT organizations are quickly adopting the public cloud, which is predominantly API-driven.

To meet customer demands and save costs, application teams are architecting their applications to support a much higher level of elasticity, supporting technology like containers and public cloud resources. These resources may only live for a matter of hours, therefore the traditional method of raising a ticket to request resources is no longer a viable option.

Benefits of IaC

Speed

IaC benefits a company's IT architecture and workflow as it uses automation to substantially increase the provisioning speed of the infrastructure' s development, testing, and production.

Consistency

Since it is code, it generates the same result every time. It provisioned the same environment every time, enabling improved infrastructure consistency at all times.

Cost

One of the main benefits of IaC is, without a doubt, lowering the costs of infrastructure management. With everything automated and organized, engineers save up on time and cost which can be wisely invested in performing other manual tasks and higher-value jobs.

Minimum Risk

IaC allows server configuration that can be documented, logged, and tracked later for reference. Configuration files will be reviewed by a person or policy as a code (sentinel) for security leakages.

Everything Codified

The main benefit of IaC is explicit coding to configure files in use. You can share codes with the team, test them to ensure accuracy, maintain uniformity and update your infrastructure into the same flow of IaC.

Version Controlled, Integrated

Since the infrastructure configurations are codified, we can check-in into version control like GitHub and start versioning it.

IaC allows you to track and give insight on what, who, when, and why anything changed in the process of deployment. This has more transparency which we lack in traditional infrastructure management.

Now, we know what is Infrastructure as Code means, now let's deep dive into Terraform...

Terraform

Terraform is a tool for building, changing, and versioning infrastructure safely and efficiently. Terraform can manage existing and popular service providers as well as custom in-house solutions.

Company Wise Python Interview Questions

Capegemini:

1. Is python interpreted language?

Ans: Yes, Python is an interpreted language. This means that Python code is not compiled into machine code before it is run. Instead, the Python interpreter reads and executes the code directly from the source code files. When you run a Python program, the interpreter parses the code and executes it one line at a time. This makes it easier to write and debug Python code, as you can see the results of your code immediately. However, interpreted languages are generally slower than compiled languages, as the interpreter has to translate the code into machine code at runtime.

2. Difference between List and Tuple.

Ans:

Mutability:
Lists are mutable, which means you can add, remove, or modify elements in a list after it has been created. Tuples, on the other hand, are immutable, which means that once a tuple is created, its contents cannot be changed.
Syntax:
Lists are created using square brackets [], while tuples are created using parentheses ().
Performance:
Tuples are generally faster than lists because they are immutable, so they can be optimized by the interpreter more efficiently. Lists require more memory allocation and deallocation, which makes them slower.
Usage:
Lists are typically used when you need to store a collection of items that may change over time, such as a list of to-do items. Tuples, on the other hand, are typically used when you need to store a collection of items that won't change, such as the coordinates of a point in space.

3. Int to string conversion.

Ans: In Python, you can convert an integer to a string using the str() function.
Example:

```
x = 10
print(str(x))
O/P: "10"
```

4. Difference between is and == in python.

Ans: The == operator is used for value equality. It compares the values of two objects to determine whether they are equal or not.
Example:

AWS/DEVOPS INTERVIEW QUESTIONS

Happiest mindes

1. Write a program to count no of times each word present in the string and save it in the dictionary?
2. Git Merge and rebase?
3. What is docker file how it works?
4. What are the ansible tasks?
5. How to deploy to 100 servers at a time?
6. What is docker volume?
7. What have you done in these years what are tools involved?
8. Have you worked on ansible modules?

WIPRO:

1. What is DevOps?
2. Why do we need a DevOps?
3. How do you configure the job in Jenkins?
4. What are the roles you played in your laptop?
5. How do you configure ansible in Jenkins?
6. Difference between ant and maven?
7. Git workflow?
8. Maven lifecycle?
9. Where do you find errors in Jenkins?
10. How do you integrate sonar Qube in Jenkins?

Good Work Labs

1. How do you configures3 bucket?
2. Differences between git rebase and git merge?
3. What is git init?
4. What is git clone?
5. If there is suddenly the file is deleted in git and how do you get it back?
6. What is CI/CD?
7. What is the purpose of Docker?
8. In Jenkins how can you find log files?
9. By using Ansible how to deploy in Jenkins?
10. What is the use of ansible?
11. What is configuration management?

Capgemini

1. Roles and Responsibilities?
2. Daily activities what you have done in the current project?
3. Jenkins workflow and write a script for this workflow?

Kubernetes Important interview Questions.

Kubernetes Q&A

1. **What is Kubernetes and why it is important?**
   Kubernetes is an open-source platform used to automate the deployment, scaling, and management of containerized applications. It is like a traffic controller for containerized applications. It ensures that these applications are running efficiently and reliably, by managing their deployment, scaling, and updating processes.
   Kubernetes is important because it makes much easier to deploy and manage complex applications across different environments and infrastructures. By providing a consistent platform for containerized applications, Kubernetes allows developers to focus on building and improving their applications, rather than worrying about the underlying infrastructure. Additionally, Kubernetes helps organizations to achieve greater efficiency, scalability, and flexibility, which can result in significant cost savings and faster time-to-market.

2. **What is difference between docker swarm and Kubernetes?**

Kubernetes	Docker Swarm
1. Kubernetes Setup is complicated but once installed the cluster is very strong.	1. Docker Swarm setup is very simple but the cluster is not very strong.
2. GUI is the Kubernetes Dashboard.	2. There is no GUI.
3. Kubernetes can do auto-scaling.	3. Docker Swarm can't do auto-scaling.
4. Kubernetes can deploy rolling updates, & does automatic rollbacks.	4. Docker Swarm can deploy rolling updates, but not automatic rollbacks.
5. Can share Storage volumes only with other containers in the same pod.	5. Can share Storage volumes with any other containers.
6. It has built-in tools for logging and monitoring.	6. It uses 3rd party tools like ELK stack for logging and monitoring.
7. Manual intervention needed for load balancing traffic between different containers and pods.	7. It does auto load balancing of traffics between containers in the cluster.

3. **How does Kubernetes handle network communication between containers?**
   Kubernetes defines a network model called the container network interface (CNI), but the actual implementation relies on network plugins. The network plugin is responsible for allocating internet protocol (IP) addresses to pods and enabling pods to communicate with each other within the Kubernetes cluster.
   When a pod is created in Kubernetes, the CNI plugin is used to create a virtual network interface for the pod. Each container in the pod is then assigned its own unique IP address within the pod's network namespace. This enables containers within the pod to communicate with each other via localhost, as if they were running on the same host.
   To enable communication between pods, Kubernetes sets up a virtual network overlay using the selected network plugin. Each node in the cluster runs a network agent that communicates with other agents on other nodes to establish the overlay network. This enables communication between containers running in different pods, even if they are running on different nodes.

4. **How does Kubernetes handle scaling of applications?**
   Kubernetes provides built-in mechanisms for scaling applications horizontally and vertically, allowing you to meet changing demands for your application.
   Horizontal scaling, also known as scaling out, involves adding more instances of an application to handle increased traffic. Kubernetes can manage this automatically through the use of ReplicaSet,

SHELL Scripting Interview Questions and Answers

1. What is a shell script?

Question: What is a shell script?

Answer: A shell script is a text file that contains a sequence of commands for a UNIX-based operating system's shell to execute. Shell scripts are used to automate repetitive tasks, manage system operations, and simplify complex commands. They can contain standard UNIX commands, conditional statements, loops, and functions.

2. How do you create and execute a shell script?

Question: How do you create and execute a shell script?

Answer:

1. **Create a Shell Script:**
   - Open a text editor and write your script.
   - Save the file with a `.sh` extension (e.g., `myscript.sh`).

   Example script:

   ```
   #!/bin/bash
   echo "Hello, World!"
   ```

2. **Make the Script Executable:**

   ```
   chmod +x myscript.sh
   ```

3. **Execute the Script:**

   ```
   ./myscript.sh
   ```

3. What is the significance of `#!/bin/bash` at the beginning of a script?

Question: What is the significance of `#!/bin/bash` at the beginning of a script?

Answer: The `#!/bin/bash` line at the beginning of a shell script is called a shebang or hashbang. It specifies the path to the interpreter that should be used to execute the script. In this case, it indicates that the script should be run using the Bash shell. It ensures that the script runs with the correct interpreter, regardless of the user's default shell.

4. How do you define and use variables in a shell script?

Question: How do you define and use variables in a shell script?

Answer:

1. What and all Jenkins jobs have you written.
2. Have you come across Jenkins files
3. Create a pipeline job (Might be asked to write the syntax or a basic piplejob depending on the input the interviewer gives)
4. Give me real time scenario which you worked upon Jenkins pipeline job
5. Have you written any multi branch pipeline job? What is the advantage of hanging multi branches over single branches?
6. You install a plugin, do you reboot Master?
7. Multiple test cases to run at the same time parallel, how to write in declarative.
8. How nexus credentials are passed to Jenkins file to upload arti-factory.
9. How you are managing your secrets to the Jenkins file.
10. Sonarqube setup and integration with Jenkins pipeline.
11. How you ensured wrong codes did not get into your pipeline.
12. Types of jobs in Jenkins
13. How do you configure maven in Jenkins?
14. Have you configured sonarqube in Jenkins?
15. Steps in Jenkins job, with syntax
16. When you have Jenkins job with declarative pipeline. Is it possible to merge the code with the master branch, using Jenkins?
17. Explain CICD complete process.
18. What application have you built for the CICD java/java script applications?
19. How do you enable continuous integration in Jenkins?
20. How to run a shell command inside a stage?
21. After generating code quality reports, how do you get the report back to developers automatically from Sonarqube in the Jenkins pipeline?
22. How to get email notification whenever a stage fails?
23. There are 10 stages in the pipeline, 4 stages are successfully completed, build fails in 5th stage, now you need to start the build from where the build process failed, How do you do that?

JENKINS PRODUCT BASED COMPANY QUESTIONS

1) What do mean by Build?

Build is nothing but compiling of sources into distributed artifacts. So developers will be writing the source code...and it needs to be compiled and packaged so then it can be distributed to the customers or users for using it..... So this build might be happening after development....it can happen during development or it can happen on a periodic basis like every week or every month...

2) What is CI?

CI is nothing but a software development practice where members of each team integrate their work daily....by integrating their work daily, they can test it every day and it reduces the problems and it can identify the problems earlier and they can fix it earlier...then we can make the product more stable... That is the main use of CI ....If we start the building at the end of the project we will identify so many issues, those are integration issues because code will be developed by the multiple developers.....if we start integrating them daily so you identify issues earlier and we can fix them earlier....

So that our code quality will get increased and will not see any integration issues during release time.....