UNIX questions:

- Can you describe the difference between hard and soft links?
    - soft link is an actual link or shortcut to the original file. Can link to directories or remote files. Deleting the original file makes the link useless. Deleting the link does nothing to original file.
    - hard link is a mirror copy of the original file i.e. creates another file with a link to the same underlying inode. Can't link to directories. Preserves the contents of the file. Even if you delete the original file, the hard link will still have the data of the original file. Deleting the link does nothing to original file.
- What is an inode?
    - The inode is a data structure in a Unix-style file system which describes a filesystem object such as a file or a directory. Each inode stores the attributes and disk block location(s) of the object's data including times of last change, access, modification, as well as owner and permission data.
- How would you get the log output for a given process?
    - You can access the output via the proc filesystem :- tail -f /proc/<pid>/fd/1 where 1 = stdout, 2 = stderr

Feature	`/proc/<pid>/fd/1` (stdout)	`/proc/<pid>/fd/2` (stderr)
Purpose	Regular output from the process.	Error or diagnostic messages.
Use Case	Viewing logs, results, or normal output.	Monitoring errors or issues in real-time.
Example Output	`Operation completed successfully.`	`Error: File not found.`

By monitoring these file descriptors, you can debug processes by separating their normal output (`stdout`) from error messages (`stderr`).

    - strace using process ID strace -p <pid>.
    - Examples of error redirection

        a. Redirect stdout to one file and stderr to another file:

            command > out 2>error

        b. Redirect stdout to a file (>out), and then redirect stderr to stdout (2>&1):

            command >out 2>&1

        c. Redirect both to a file (this isn't supported by all shells, bash and zsh support it, for example, but sh and ksh do not):

            command &> out

- You setup a webserver on a UNIX machine and connect to it with your browser. You get a 403 Forbidden error. What might be wrong? How would you fix it?

- is an HTTP status code which means that accessing the page or resource you were trying to reach is absolutely forbidden for some reason. Caused by :
    - An empty website directory - Fix could be to make sure that your website content has been uploaded to the correct directory on your server.
    - No index page - Fix home page for your website must be called index.html or index.php. To resolve this error, upload an index page to your httpdocs or public_html directory.
    - Permission / Ownership error - Fix chmod/chown Folders: 755, Static Content: 644, Dynamic Content: 700

- How would you find directories holding much of the disk space on a given machine?
    - du -s * | sort -nr | head -n10

Can you describe some commands you might use to get user information

Commands to Get User Information in Unix/Linux

Command	Purpose	Example
`id`	Displays UID, GID, and groups for a user	`id username`
`groups`	Lists groups a user belongs to	`groups username`
`finger`	Provides detailed user info (e.g., shell, home)	`finger username`
`lslogins`	Shows comprehensive user details	`lslogins`
`users`	Lists currently logged-in users	`users`
`who`	Shows logged-in users and their sessions	`who`
`w`	Displays logged-in users and their activities	`w`
`last`	Shows login history of users	`last username`
`lastlog`	Displays last login time for all users	`lastlog`
`cat /etc/passwd`	Shows user account details	`cat /etc/passwd`
`cat /etc/group`	Shows group account details	`cat /etc/group`
`ps`	Lists running processes for a specific user	`ps -u username`

- What is a PCAP file? How would you generate one and use it for analysis?
    - Packet Capture or PCAP (also known as libpcap) is an application programming interface (API) that captures live network packet data from OSI model Layers 2-7. Network analyzers like Wireshark create . pcap files to collect and record packet data from a network
    - You can create a .pcapfile by using a network analyzer or packet sniffing tool like Wireshark or tcpdump

- Please describe some commonly used network commands.
    - telnet: it is used for remote login as well as for communication with another hostname.
    - ping: it is defined as an echo request for checking network connectivity.
    - su: derived as a user switching command.
    - hostname: determines the IP address and domain name.
    - nslookup: performs DNS query.
    - xtraceroute: method to determine the number of hoops and response time required to reach the network host.
    - netstat: it provides a lot of information like ongoing network connection on the local system and ports, routing tables, interfaces statistics, etc.
    - tcpdump: captures traffic
- How would you check performance metrics on server?
    - top
    - vmstat
    - iostat
    - sar
    - free
    - lsof
    - nestat
    - tcpdump
    - htop
    - collectl

Docker questions:

- Please describe some Dockerfile instructions (https://www.learnitguide.net/2018/06/dockerfile-explained-with-examples.html)
    - ADD
    - COPY
    - ENV
    - EXPOSE
    - FROM
    - LABEL
    - STOPSIGNAL

- USER
- VOLUME
- WORKDIR
- ONBUILD

Dockerfile Instructions

Instruction	Purpose	Example
ADD	Copies files/directories into the image and can also extract compressed files (e.g., tar).	ADD source_file.tar /destination_path
COPY	Copies files/directories from the host machine into the image.	COPY source_path /destination_path
ENV	Sets environment variables for the container.	ENV APP_ENV production
EXPOSE	Specifies the ports that the container will listen on at runtime.	EXPOSE 80 443
FROM	Defines the base image for the container. **This is the first instruction in a Dockerfile.**	FROM debian:stable
LABEL	Adds metadata to the image in key-value pairs. Useful for documentation and automation.	LABEL maintainer="your_email@example.com"
STOPSIGNAL	Specifies the system call signal to stop the container.	STOPSIGNAL SIGTERM
USER	Specifies the user (and optionally group) to use when running the image.	USER appuser
VOLUME	Creates a mount point for data volumes, which can store data persistently outside the container.	VOLUME ["/var/www", "/data"]
WORKDIR	Sets the working directory for instructions like RUN, CMD, ENTRYPOINT, etc.	WORKDIR /app
ONBUILD	Adds a trigger instruction to the image that runs when another image is built using this image as a base.	ONBUILD RUN apt-get update

```
FROM debian:stable

RUN apt-get update && apt-get install -y --force-yes apache2

EXPOSE 80 443

VOLUME ["/var/www", "/var/log/apache2", "/etc/apache2"]

ENTRYPOINT ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
```

```
# Set the base image

FROM debian:stable

# Install Apache HTTP server

RUN apt-get update && apt-get install -y --force-yes apache2

# Expose ports 80 and 443 for HTTP and HTTPS traffic

EXPOSE 80 443

# Create mount points for persistent storage

VOLUME ["/var/www", "/var/log/apache2", "/etc/apache2"]

# Set the entry point to keep Apache running in the foreground

ENTRYPOINT ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
```

Can you describe the difference between the CMD and ENTRYPOINT instructions?

- CMD is an instruction that is best to use if you need a default command which users can easily override. If a Dockerfile has multiple CMDs, it only applies the instructions from the last one.
- ENTRYPOINT is preferred when you want to define a container with a specific executable. You cannot override an ENTRYPOINT when starting a container unless you add the --entrypoint flag.
- Combine ENTRYPOINT with CMD if you need a container with a specified executable and a default parameter that can be modified easily. For example, when containerizing an application use ENTRYPOINT and CMD to set environment-specific variables.

Can you talk about some base Docker images you like to use and why?

Here are some base Docker images that are commonly used and why they are popular:

1. **alpine**: A minimal, lightweight base image, typically around 5MB. It's ideal for small, efficient containers and allows you to install only the packages you need, reducing the attack surface.
2. **debian**: A well-known, stable Linux distribution. It's larger than Alpine but provides a richer set of tools and libraries out-of-the-box, making it suitable for more complex applications where you need compatibility with Debian-based packages.
3. **ubuntu**: One of the most widely used base images, it provides a more comprehensive set of tools and libraries compared to Alpine. It's suitable for applications that require compatibility with Ubuntu's ecosystem, while still being a relatively lightweight option.
4. **node**: For Node.js applications, the official node image is optimized for running Node.js and includes necessary dependencies and tools for JavaScript-based projects.
5. **python**: The official python image is ideal for Python-based applications, as it comes pre-installed with Python, pip, and common libraries, saving time in setting up the environment.
6. **nginx**: If you need a reverse proxy or web server, the nginx image is a great choice. It's lightweight, fast, and highly configurable for serving static files or as a proxy for backend services.

These images are popular because they are official, well-maintained, and come with an appropriate set of dependencies to get started quickly. They allow developers to focus on their application without spending too much time on setting up the base environment.

- What is a hypervisor? Please describe different types if you can.
    - is software that creates and runs virtual machines. Physical hardware, when used as a hypervisor, is called the host, while the many VMs that use its resources are guests.
    - Type 1 hypervisor, also referred to as a native or bare metal hypervisor, runs directly on the host's hardware to manage guest operating systems. It takes the place of a host operating system and VM resources are scheduled directly to the hardware by the hypervisor.
    - Type 2 hypervisor is better for individual users who want to run multiple operating systems on a personal computer. It works by abstracting guest operating systems from the host operating system. VM resources are scheduled against a host operating system, which is then executed against the hardware.

What is virtualisation?

- Virtualization is the process of running a virtual instance of a computer system in a layer abstracted from the actual hardware

- How does containerisation differ from virtualisation?
    - A container is a set of 1 or more processes that are isolated from the rest of the system. The container allows the process to access only the resource requests that have been specified. These resource limits ensure that the container is able to run on a node that has enough capacity.
    - VMs contain their own operating system (OS), allowing them to perform multiple resource-intensive functions at once. The increased resources available to VMs allow them to abstract, split, duplicate, and emulate entire servers, OSs, desktops, databases, and networks
    - A hypervisor also allows you to run multiple operating systems in VMs, but containers are only able to run a single type of operating system. A container running on a Linux server, for example, is only able to run a Linux operating system
    - Containers are sometimes thought of as a replacement for hypervisors, though this isn't exactly accurate since containers and virtualization meet different needs.

- How would you safely clear disk space used by Docker?
    - prune
- How would you enable remote access to the Docker API?
    - vi /lib/systemd/system/docker.service.
    - ExecStart=/usr/bin/dockerd -H fd:// -H tcp://0.0.0.0:<port>
    - restart docker
- How would you monitor Docker containers on a fleet of production hosts?
    - cAdvisor
- Please briefly explain the Docker container lifecycle.
    - Created: A container that has been created but not started
    - Running: A container running with all its processes
    - Paused: A container whose processes have been paused
    - Stopped: A container whose processes have been stopped
    - Deleted: A container in a dead state
- How would you share data between multiple containers?
    - create a data volume

Ansible questions:

- Please briefly describe Ansible and how it works
    - Ansible works by connecting to your nodes and pushing out small programs, called "Ansible modules" to them. These programs are written to be resource models of the desired state of the system. Ansible then executes these modules (over SSH by default), and removes them when finished.
    - Your library of modules can reside on any machine, and there are no servers, daemons, or databases required. Typically you'll work with your favorite terminal program, a text editor, and probably a version control system to keep track of changes to your content.
- Please state the requirements for an Ansible server
    - control node (the machine that runs Ansible), you can use any machine with Python 2 (version 2.7) or Python 3 (versions 3.5 and higher)
    - for managed nodes, Ansible makes a connection over SSH and transfers modules using SFTP
- What is idempotency?
    - an action which, when performed multiple times, has no further effect on its subject after the first time it is performed.
- How do you use Ansible to encrypt files or variables?
    - vault
- How would you setup an Ansible project to use multiple vault IDs?
- What are tags and how would you use them?
- If you have a large playbook, it may be useful to run only specific parts of it instead of running the entire playbook. You can do this with Ansible tags. Using tags to execute or skip selected tasks is a two-step process:

1. Add tags to your tasks, either individually or with tag inheritance from a block, play, role, or import.
2. Select or skip tags when you run your playbook.

- Can you describe how you would use asserts?
    - This module asserts that given expressions are true with an optional custom message.
- What does check_mode do?
    - Check mode is just a simulation. It will not generate output for tasks that use conditionals based on registered variables (results of prior tasks). Used for validating configuration management playbooks that run on one node at a time. Use --check flag
- What happens when gather_facts is set to true for a play?
    - gathers facts about remote host.
    - This module takes care of executing the configured facts modules, the default is to use the setup module.

    - This module is automatically called by playbooks to gather useful variables about remote hosts that can be used in playbooks.
    - It can also be executed directly by /usr/bin/ansible to check what variables are available to a host.
    - Ansible provides many *facts* about the system, automatically.
- For a project that provisions machines in the cloud, how would you dynamically populate inventory?
    - Dynamic inventory is an ansible plugin that makes an API call to AWS to get the instance information in the run time. It gives you the ec2 instance details dynamically to manage the AWS infrastructure.
    - Ansible integrates all of these options through a dynamic external inventory system. Ansible supports two ways to connect with external inventory: Inventory Plugins
- A piece of Ansible is running on a remote host. How would you access variables from other hosts?
    - gather facts then use hostvars which is a hash with inventory hostnames as keys. e.g. hostvars['test-1'], hostvars['test2-1'] etc
- How would you handle the use of different user accounts for different hosts?
    - use the become and become_user arguments
- How would you simply test the connection to all hosts in your inventory?
    - ansible all -m ping

AWS questions:

- What is the relation between the Availability Zone and Region?

    - Availability Zones are multiple, isolated locations within each Region.
    - Local Zones provide you the ability to place resources, such as compute and storage, in multiple locations closer to your end users.
    - AWS Outposts brings native AWS services, infrastructure, and operating models to virtually any data center, co-location space, or on-premises facility.
    - Wavelength Zones allow developers to build applications that deliver ultra-low latencies to 5G devices and end users. Wavelength deploys standard AWS compute and storage services to the edge of telecommunication carriers' 5G networks.
- Can you name some of the AWS services that are not region-specific?
    - Route53
    - iam
    - cloudfront
- Please describe the various types of EC2 instances.
    - General Purpose
    - Compute optimised
    - Memory optimised
    - Accelerated computing
    - Storage optimised
- Can you describe some differences between S3 and EBS and EFS storage?
    - Amazon EBS delivers high-availability block-level storage volumes for Amazon Elastic Compute Cloud (EC2) instances. It stores data on a file system which is retained after the EC2 instance is shut down. - Best performance and latency
    - Amazon EFS offers scalable file storage, also optimized for EC2. It can be used as a common data source for any application or workload that runs on numerous instances. Using an EFS file system, you may configure instances to mount the file system. The main differences between EBS and EFS is that EBS is only accessible from a single EC2 instance in your particular AWS region, while EFS allows you to mount the file system across multiple regions and instances. - Shared filesystem
    - Amazon S3 is an object store good at storing vast numbers of backups or user files. Unlike EBS or EFS, S3 is not limited to EC2. Files stored within an S3 bucket can be accessed programmatically or directly from services such as AWS CloudFront. This is why many websites use it to hold their content and media files, which may be served efficiently from AWS CloudFront.
- Can you describe some of the ways we might connect to internal services in our corporate network from services in AWS and vice versa?
    - AWS site-to-site vpn or Direct Connect
    - AWS PrivateLink

- What AWS features would you use to secure your services in AWS?
    - iam
    - security groups
    - Network acls
- What would you use the various different types of AWS ELBs for?
    - Application Load Balancer - balancing of HTTP and HTTPS traffic and provides advanced request routing targeted at the delivery of modern application architectures, including microservices and containers. Application Load Balancer routes traffic to targets within Amazon VPC based on the content of the request
    - Network Load Balancer - balancing of Transmission Control Protocol (TCP), User Datagram Protocol (UDP), and Transport Layer Security (TLS) traffic where extreme performance is required. Network Load Balancer routes traffic to targets within Amazon VPC and is capable of handling millions of requests per second while maintaining ultra-low latencies.
    - Gateway Load Balancer - makes it easy to deploy, scale, and run third-party virtual networking appliances. Providing load balancing and auto scaling for fleets of third-party appliances, Gateway Load Balancer is transparent to the source and destination of traffic. This capability makes it well suited for working with third-party appliances for security, network analytics, and other use cases.
- How would you allow a user access to specific folder in an S3 bucket?
    - Amazon S3 offers access policy options broadly categorized as resource-based policies and user policies.
    - Access policies that you attach to your resources (buckets and objects) are referred to as *resource-based policies*. For example, bucket policies and access control lists (ACLs) are resource-based policies.
    - You can also attach access policies to users in your account. These are called *user policies*. You can choose to use resource-based policies, user policies, or some combination of these to manage permissions to your Amazon S3 resources.
- How would you vertically scale an EC2 instance?
    - instance has to be stopped, then the instance size changed, then restarted. This is due to the fact that Amazon has to often move the VM to a different piece of hardware with the available resources for the size change.
- Can you establish a peering connection to a VPC in a different REGION?
    - Not possible. Peering Connection are available only between VPC in the same region.
- Can you connect your VPC with a VPC owned by another AWS account?
    - Yes, Possible. Provided the owner of other VPCs accepts your connection.
- How can you monitor network traffic in your VPC?
    - It is possible using Amazon VPC Flow-Logs feature.

- What is the difference between Security Groups and ACLs in a VPC?
    - A Security Group defines which traffic is allowed TO or FROM EC2 instance. Whereas ACL, controls at the SUBNET level, scrutinise the traffic TO or FROM a Subnet.
- Key Pair and its uses?
    - You use Key Pair to login to your Instance in a secured way. You can create a key pair using EC2 console. When your instances are spread across regions you need to create key pair in each region.
- Can you edit a Route Table in VPC?
    - Yes. You can always modify route rules to specify which subnets are routed to the Internet gateway, the virtual private gateway, or other instances.
- What is autoscaling & mentions some of its benefits?
    - Autoscaling is a service that automatically scales EC2 instance capacity out and in based on the criteria's that we are going to set. Autoscaling benefits its use for dynamic workloads like web spikes, retail shop flash sales, ticket booking system on the vacations etc.,
- If you peer vpc A to vpc B and I peer vpc B to vpc C, does that mean VPC's A and C peer?
    - No, Transitive peering relationships are not supported.
- Can you assign 2 IPs for a single EC2 instance?
    - Yes, primary and secondary IP is possible. Only when it is private IP.
- How can you convert a public subnet to private subnet?
    - Remove IGW & add NAT Gateway, Associate subnet in Private route table

Other questions:

- What are SLAs, SLOs, and SLIs?
    - An SLA (service level agreement) is an agreement between provider and client about measurable metrics like uptime, responsiveness, and responsibilities.
    - An SLO (service level objective) is an agreement within an SLA about a specific metric like uptime or response time.
    - An SLI (service level indicator) measures compliance with an SLO (service level objective).

Scenario 1:

- You've been brought in to help architect a solution for a scalable web application. A delivery team has produced an application can that be scaled horizontally, and have their code for it in Github which is tagged and versioned appropriately. Also in this GitHub is a Dockerfile that packages the app as a Docker image, and requires environment variables to configure how it runs which are well documented.
- The application hosts a RESTful API on a HTTP endpoint, and also uses Couchbase as a backend database. A Dockerfile is also provided in GitHub for the Couchbase instances, and

also require some environment variables to be set when run. Couchbase runs as a cluster and there are no rules as to which node a given instance of the app has to connect to.

- Please architect a solution for hosting the app and and the database clusters and making the app available on the internet. Ideally, the app and Couchbase clusters should be scalable with minimal effort, and recovery should be automatic in the case of failure.
- Please make use of the following:
    - Docker registry
    - Jenkins or other pipeline solution
    - EC2
    - EBS
    - Route53
    - ELBs
    - Terraform or Ansible or both
    - Prometheus or other live metric monitoring solution

Terraform questions:

- Please describe some of the features of Terraform.

- Please describe the various terraform commands.
- Can you describe how Terraform differs to Ansible?
- How do plugins work in Terraform? How do you use them?
- How would I use an older version of a plugin?
- Can I use Terraform for on-premise infrastructure?
- I've worked on some terraform to spin up some infrastructure in my dev environment. How might I adapt the code so I can use it for other environments?
- What is a terraform state file, how is it used, and where is it stored?
- How would I destroy a single resource with terraform?

Features of Terraform

- **Infrastructure as Code (IaC)**: Declaratively define infrastructure using configuration files.
- **State Management**: Maintains the state of your infrastructure to manage changes over time.
- **Resource Graph**: Automatically determines dependencies between resources and creates them in the correct order.
- **Multi-Provider Support**: Works with various providers like AWS, Azure, GCP, and on-premise systems.
- **Plan and Apply**: Preview changes with terraform plan and apply them with terraform apply.
- **Modules**: Enables reuse of configurations to manage complex environments efficiently.

Terraform Commands

1. **terraform init**: Initializes a working directory with the necessary provider plugins.
2. **terraform plan**: Creates a plan showing changes Terraform will make to match the desired state.
3. **terraform apply**: Applies the changes defined in the configuration files.
4. **terraform destroy**: Deletes all resources managed by the configuration.
5. **terraform validate**: Validates the syntax and configuration of Terraform files.
6. **terraform fmt**: Formats configuration files to a standard style.
7. **terraform state**: Manages and inspects the state file (e.g., listing resources or moving them).
8. **terraform output**: Displays outputs from the state file.
9. **terraform taint**: Marks a resource for destruction and recreation during the next apply.
10. **terraform import**: Imports existing resources into Terraform's state file.

11. **terraform workspace**: Manages multiple workspaces/environments.

Terraform vs. Ansible

Feature	Terraform	Ansible
Purpose	Focuses on infrastructure provisioning.	Focuses on configuration management.
Declarative/Procedural	Declarative. Users define the end state.	Procedural. Users define step-by-step tasks.
State Management	Maintains a state file to track resources.	Stateless; it does not track resource state.
Resource Graph	Builds a dependency graph automatically.	Requires manual handling of dependencies.

Terraform Plugins

- **How Plugins Work**: Plugins are used to extend Terraform's capabilities, allowing it to interact with specific providers or systems (e.g., AWS, Azure, Kubernetes).
- **How to Use Plugins**: When you run terraform init, Terraform downloads and installs the required provider plugins based on the configurations in your .tf files.
- **Using Older Versions of Plugins**:
    - Specify the plugin version in the configuration file:

```
terraform {
 required_providers {
  aws = {
   source  = "hashicorp/aws"
   version = "3.50.0"
  }
 }
}
```

Using Terraform for On-Premise Infrastructure

Yes, Terraform can be used for on-premise infrastructure by leveraging providers like:

- VMware vSphere
- OpenStack

- Bare Metal (via custom providers)

Adapting Code for Multiple Environments

- **Use Variables**: Define variables for environment-specific values (e.g., region, instance type).
- **Workspaces**: Use terraform workspace to manage multiple environments.
- **Modules**: Extract common code into reusable modules, then parameterize them for different environments.

Example directory structure:

environments/

dev/

main.tf

staging/

main.tf

production/

main.tf

Terraform State File

- **What it is**: A file (terraform.tfstate) that records the current state of your infrastructure.
- **Usage**: Helps Terraform understand the resources it manages and plans changes accordingly.
- **Storage**: Stored locally by default or remotely (e.g., S3, GCS, Azure Blob) for team collaboration.

Destroying a Single Resource

- Use the -target flag with terraform destroy:

terraform destroy -target=aws_instance.my_instance

This removes only the specified resource without affecting others.