**DevOps Most Useful Real-Time Interview Questions**



**Introduction**

DevOps is a dynamic field that requires a combination of skills in development and operations. In a DevOps interview, candidates often face a variety of questions covering technical aspects, problem-solving, and real-world scenarios. Here, we delve into some of the most useful real-time interview questions for DevOps engineers, along with practical scenarios and solutions.

**2. What are the roles and responsibilities?**

**Answer:** In a DevOps role, responsibilities include collaborating with development and operations teams, automating processes, managing infrastructure as code, ensuring continuous integration/continuous deployment (CI/CD), monitoring systems, and resolving incidents promptly.

**3. What is an RCA report? How do you prepare it? What things do we need to consider to create an RCA report?**

**Answer:**RCA (Root Cause Analysis) Report: An RCA report identifies the core cause of an issue or incident. To prepare it, follow these steps:

* Collect Data: Gather information related to the incident.
* Identify Causes: Analyze data to determine the root cause.
* Documentation: Clearly document the identified cause and contributing factors.
* Recommendations: Suggest preventive measures to avoid recurrence.
* Feedback Loop: Implement feedback mechanisms for continuous improvement.

**4. You are the team leader managing teams. You received an escalation from a client for any service issues. How do you handle it?**

**Answer:**

1. Acknowledge: Acknowledge the escalation promptly.
2. Investigate: Gather information about the issue.
3. Communication: Keep the client informed about the investigation progress.
4. Resolution Plan: Develop a plan to resolve the issue.
5. Implement Solution: Execute the plan, ensuring minimal impact.
6. Post-Incident Review: Conduct a review to prevent future occurrences.
7. Client Communication: Update the client on the resolution and preventive measures.

**5. What major challenge have you faced in your role, and how did you handle it?**

**Answer:** Scenario: Scaling Infrastructure

* Challenge: Faced with a sudden spike in traffic, existing infrastructure struggled to handle the load.
* Solution:
* Implemented auto-scaling to dynamically adjust resources.
* Utilized CDN to distribute content globally.
* Conducted load testing regularly for capacity planning.

**6. Why are organizations implementing DevOps?**

**Answer:**Organizations adopt DevOps to:

* Accelerate delivery cycles.
* Enhance collaboration between development and operations.
* Improve deployment frequency and success rates.
* Automate manual processes for efficiency.
* Enhance system reliability and scalability.

**7. You received a task in Jira. What information do you need, and from where can you collect it before implementing the task?**

**Answer:**Before implementing a Jira task:

1. Task Details: Understand the task requirements and expected outcomes.
2. Dependencies: Identify any dependencies on other teams or services.
3. Testing Criteria: Define criteria for successful testing.
4. Documentation: Refer to project documentation or collaborate with stakeholders.
5. Environment Information: Confirm details about the deployment environment.

**8. Explain the Production release process.**

**Answer:** Production Release Process:

1. Planning: Schedule release window and coordinate with teams.
2. Code Freeze: Restrict code changes to stabilize the codebase.
3. Testing: Conduct thorough testing, including regression testing.
4. Deployment: Deploy the release to production servers.
5. Monitoring: Monitor systems for any anomalies or issues.
6. Rollback Plan: Have a rollback plan in case of issues.
7. Communication: Keep stakeholders informed throughout the process.

**9. If my changes work fine on dev and test environments but not on prod, what could be the issues, and how do you fix it?**

**Answer:** Issues and Solutions:

1. Environment Differences: Identify and replicate prod-like conditions in the lower environments.
2. Configuration Mismatch: Ensure configurations are consistent across environments.
3. Data Discrepancy: Check for data variations between environments.
4. Dependency Issues: Validate dependencies and versions in each environment.
5. Rollback or Hotfix: If critical, consider rolling back changes or applying a hotfix.

**10. What is the best way to design a 3-tier architecture? Which services are included, and how do you select services to design it?**

**Answer:** 3-Tier Architecture:

1. Presentation Tier: Front-end services, user interfaces.
2. Application Tier: Business logic, server-side processing.
3. Data Tier: Database storage, data management.

Service Selection:

* Choose services based on scalability, performance, and security requirements.
* Use load balancers for distribution and reliability.
* Implement caching mechanisms for performance optimization.
* Apply security groups and network ACLs to control traffic.

**11. What are the strategies for infra cost optimization, and what actions will you take to reduce infra cost?**

**Answer:** Cost Optimization Strategies:

1. Reserved Instances: Utilize reserved instances for predictable workloads.
2. Spot Instances: Leverage spot instances for temporary and flexible workloads.
3. Rightsizing: Match instance types to actual resource needs.
4. Auto Scaling: Automatically adjust resources based on demand.
5. Tagging: Implement tagging for resource categorization and cost allocation.

**Actions:**

* Regularly analyze AWS Cost Explorer for cost breakdown.
* Implement automated shutdown policies for non-production environments.
* Optimize storage costs by selecting appropriate storage classes.

**12. How does auto-scaling work? Is it possible to change AMI in an auto-scaling group?**

**Answer:** Auto-Scaling Workflow:

1. Scaling Policies: Define policies based on metrics like CPU utilization.
2. Trigger Events: Events trigger scaling actions (e.g., launch or terminate instances).
3. Launch Configurations: Pre-defined configurations for launched instances.

Changing AMI in Auto-Scaling Group:

* Create a new launch configuration with the desired AMI.
* Update the auto-scaling group to use the new launch configuration.
* Instances launched thereafter will use the updated AMI.

**13. If someone created resources in AWS or deleted something, how do you get those details? Which AWS service can help find out these details?**

**Answer:** AWS CloudTrail:

* CloudTrail logs all API calls and actions within an AWS account.
* Enables tracking of resource creation, deletion, and modification.
* Provides detailed information, including the identity of the entity making the call and the time of the call.

**14. What is the difference between Latency-Based Routing and Geo DNS?**

**Answer:**

* Latency-Based Routing: Routes traffic based on the lowest network latency to improve response times.
* Geo DNS: Routes traffic based on the geographic location of the user, directing them to the nearest server for improved performance.

**15. What is the difference between a Domain and a Hosted Zone?**

**Answer:**

* Domain: Represents a website or application’s address (e.g., example.com).
* Hosted Zone: A container for DNS records, mapping domain names to IP addresses.

**16. When we create a VPC, what components are created by default?**

**Answer:** By default, creating a VPC in AWS includes:

* Main route table.
* Default security group.
* Default network access control list (ACL).
* Default subnet in each Availability Zone.

**17. How to recover a CloudFormation stack if it’s stuck in ‘create in progress’ or ‘failed’ status?**

**Answer:**

1. Check Dependencies: Ensure all dependencies are available.
2. Review Events: Analyze stack events for insights.
3. Rollback: If in ‘create in progress,’ consider rolling back and redeploying.
4. Troubleshoot Errors: Address issues causing failure.
5. Recreate Stack: In some cases, deleting and recreating the stack might be necessary.

**18. What is the use case of AWS Config service?**

**Answer:** AWS Config Service:

* Tracks changes to AWS resources.
* Provides a detailed inventory of configurations.
* Enables compliance checking and security analysis.

**19. How to track AWS cloud service changes?**

**Answer:**

* Use AWS Config to track configuration changes.
* Set up AWS Config rules for real-time alerts on specific changes.
* Utilize AWS CloudTrail for detailed API call logging.

**20. What is the use of DynamoDB?**

**Answer:**DynamoDB:

* Fully managed NoSQL database service by AWS.
* Scales seamlessly and provides low-latency access to data.
* Suitable for applications with variable and high read/write workloads.

**21. If you want to give someone temporary access for like 1 hour, how do you give it? How do you configure that?**

**Answer:**Temporary Access:

* Create an AWS Identity and Access Management (IAM) policy with the necessary permissions.
* Use AWS Security Token Service (STS) to generate temporary credentials.
* Set a time-limited session duration (e.g., 1 hour) when issuing the temporary credentials.

**22. What is the difference between AWS managed policy and customer-managed policy?**

**Answer:**

* AWS Managed Policy: Created and managed by AWS, provides pre-defined permissions.
* Customer-Managed Policy: Customized and managed by the customer, offering more flexibility and control over permissions.

**23. How to handle large traffic in Application Load Balancer?**

**Answer:**

* Auto Scaling: Dynamically adjust the number of instances based on traffic.
* Connection Draining: Gradually redirect traffic away from unhealthy instances during updates.
* Use of CDN: Distribute static content globally for reduced load on the ALB.

**24. You received a notification from AWS about a potential security breach. What immediate actions will you take to secure the account?**

**Answer:**

1. Change Credentials: Rotate compromised access keys and passwords.
2. Review AWS Config: Check for unauthorized changes in AWS Config.
3. Disable Compromised Access: Temporarily disable compromised access points.
4. Investigate and Mitigate: Investigate the root cause and take actions to prevent further breaches.
5. Enable Multi-Factor Authentication (MFA): Enhance security by enabling MFA.

**25. If a database administrator mistakenly deleted chunks of data records from the database, how do you recover that?**

**Answer:**Data Recovery Steps:

1. Backup: Restore from the latest backup.
2. Point-in-Time Recovery: Use database features for point-in-time recovery.
3. Transaction Logs: Utilize transaction logs if available.
4. Database Replication: If applicable, consider replication from a healthy database.

**26. How to migrate large data from one S3 bucket to another S3 bucket?**

**Answer:**

1. AWS CLI: Use aws s3 sync command for efficient data transfer.
2. AWS DataSync: For large-scale, fast, and secure data transfer.
3. AWS Snowball: Physical device for large-scale data transfer.

**27. I need one EC2 instance only for 1 hour daily. Which instance types or options will you choose for daily use?**

**Answer:**

* Utilize AWS Spot Instances, which offer cost savings for short-term, intermittent workloads.

**28. How does a load balancer work? What are the algorithms?**

**Answer:**Load Balancer Workflow:

1. Distributes incoming traffic across multiple servers.
2. Ensures even distribution to optimize resource utilization.
3. Monitors server health and redirects traffic away from unhealthy instances.

Algorithms:

* Round Robin: Distributes traffic equally.
* Least Connections: Sends traffic to the server with the fewest active connections.
* IP Hash: Bases distribution on the client’s IP address.

**29. What is Control Tower and Landing Zone?**

**Answer:**

* AWS Control Tower: Service for setting up and governing a secure, multi-account AWS environment.
* AWS Landing Zone: An environment pre-configured using AWS Control Tower, establishing a secure and scalable multi-account structure.

**30. How to check server logs?**

**Answer:**

* Use tail or cat commands for real-time logs.
* Explore log directories like /var/log/ for specific logs.
* Check application-specific log locations.

**31. If server performance suddenly slows down, what steps or actions do we need to follow to resolve this issue?**

**Answer:**

1. Monitor Metrics: Identify performance bottlenecks.
2. Analyze Logs: Check system and application logs for errors.
3. Resource Scaling: Adjust resources based on demand.
4. Optimization: Optimize queries, code, or configurations.
5. Patch and Updates: Ensure systems are up to date.

**32. How to check which installed services are running on which port using a command?**

**Answer:** The netstat command can be used to check which services are running on which ports. For example:

netstat -tulpn

**33. If I’m unable to log in to my EC2 machine, how do I check the reason and fix it?**

Answer: To diagnose login issues:

* Check security group settings to ensure SSH access.
* Review key pair settings.
* Examine system logs using cloud-init-output.log or /var/log/auth.log for Linux instances.
* For Windows instances, check the Event Viewer.

**34. What is the best Git branching strategy?**

Answer: A popular branching strategy is Gitflow, which involves master, develop, feature, release, and hotfix branches. It ensures a structured approach to feature development, release management, and bug fixing.

**35. Explain Git Commands.**

Answer: Essential Git commands include:

* git clone: Clone a repository.
* git add: Stage changes for commit.
* git commit: Commit changes.
* git pull: Fetch changes from a remote repository.
* git push: Push changes to a remote repository.
* git branch: Create, list, or delete branches.

**36. How to resolve Git merge conflicts?**

Answer: Resolving Git merge conflicts involves:

* Identifying conflicted files using git status.
* Opening conflicted files and manually resolving conflicts.
* Marking conflicts as resolved using git add.
* Completing the merge with git commit.

**37. Explain Git troubleshooting.**

Answer: Git troubleshooting involves:

* Checking for network issues.
* Verifying repository permissions.
* Examining local configurations using git config --list.
* Debugging using git log and git reflog.

**38. Differences between Git rebase and Git merge?**

Answer: Git rebase incorporates changes from one branch into another by moving or combining commits. Git merge combines changes from different branches, creating a new merge commit. Rebase produces a linear history, while merge preserves branch history.

**39. If a file is suddenly deleted in Git, how do you recover it?**

Answer: If a file is deleted in Git, you can recover it from a previous commit using the following command:

git checkout <commit-hash> -- path/to/deleted/file

**40. Differences between**git pull**and**git fetch**?**

Answer: git pull fetches changes from a remote repository and merges them into the current branch. git fetch only downloads changes from the remote repository, allowing you to review and merge them separately.

**41. What is the difference between Git and GitHub?**

Answer: Git is a distributed version control system, while GitHub is a web-based platform that hosts Git repositories and provides collaboration features such as pull requests, issues, and project boards.

**42. What are some key benefits of using Git for version control?**

Answer: Key benefits of using Git include:

* Distributed version control.
* Efficient branching and merging.
* Offline capabilities.
* Strong support for non-linear development.
* Speed and performance.

**43. What is a webhook?**

Answer: A webhook is an automated way to receive notifications and trigger actions when specific events occur. In the context of Git and GitHub, webhooks are used to inform external systems about repository events like pushes or pull requests.

**44. What is the difference between a freestyle project and a declarative pipeline in Jenkins?**

Answer: In Jenkins, a freestyle project is a traditional project type with a graphical user interface for configuring builds. A declarative pipeline, on the other hand, is defined in a Jenkinsfile using a simplified, structured syntax that allows version control and code review.

**45. What is multi-stage deployment?**

Answer: Multi-stage deployment involves deploying an application through multiple environments, such as development, testing, and production. Each stage validates specific aspects of the application before progressing to the next stage.

**46. How to check deployment logs?**

Answer: Deployment logs can be checked in Jenkins by navigating to the specific build job and selecting the “Console Output” option. This provides detailed information about each step in the deployment process.

**47. If my deployment fails, how do I check logs and fix it?**

Answer: To troubleshoot a failed deployment:

* Review the Jenkins console output for error messages.
* Check application logs on deployed servers.
* Validate configuration settings.
* If needed, roll back to the previous stable version.
* Collaborate with the development and operations teams to identify and resolve issues.

**48. If my Jenkins server crashes, how do I recover it?**

Answer: Jenkins server recovery involves restoring the Jenkins home directory from backups. Regularly backing up the Jenkins home directory, including configuration files and job configurations, ensures a smooth recovery process.

**49. What is Jenkins master-slave architecture?**

Answer: In a Jenkins master-slave architecture, the master server manages the configuration, scheduling, and monitoring of builds, while slave servers execute build jobs. This distributed approach allows for parallel execution of builds across multiple machines.

**50. What is the use of Jenkins slave server?**

Answer: Jenkins slave servers perform the actual build jobs delegated by the master server. They help distribute the workload, enabling parallel and efficient execution of builds on different machines.

**51. How to maintain Jenkins failover or high availability?**

Answer: Jenkins failover or high availability can be achieved by setting up a Jenkins master in a highly available architecture, including multiple instances behind a load balancer. Additionally, configuring Jenkins master and slave nodes ensures continuous build availability.

**52. How to secure credentials in Jenkins pipelines?**

Answer: Secure credentials in Jenkins pipelines by using the Jenkins Credential Binding plugin. This allows sensitive information such as usernames and passwords to be securely stored and accessed during pipeline execution.

**53. How to rollback deployment in Jenkins?**

Answer: Rollback deployment in Jenkins by selecting a previous stable build version and triggering a new deployment. Ensure that the rollback process includes reversion of configuration settings to match the selected build version.

**54. Steps to create freestyle project and declarative pipeline in Jenkins.**

Answer: The steps to create a freestyle project and declarative pipeline in Jenkins involve navigating to the Jenkins dashboard, selecting “New Item,” choosing the project type, configuring build settings, and saving the configuration.

**55. How to integrate SonarQube in Jenkins?**

Answer: Integrating SonarQube in Jenkins involves installing the SonarQube Scanner plugin, configuring SonarQube server details in Jenkins global settings, and adding the SonarQube scanner build step to the Jenkins job.

**56. What is poll scm?**

Answer: “Poll SCM” in Jenkins refers to the periodic checking of the version control system for changes. If changes are detected, Jenkins triggers a build. It is a way to automate builds based on changes in the source code repository.

**57. How do you create a backup of Jenkins?**

Answer: Creating a backup of Jenkins involves regularly copying the entire Jenkins home directory, including configurations, jobs, and plugins. This ensures that all Jenkins settings are backed up and can be restored in case of failure.

**58. How to integrate AWS with Jenkins?**

Answer: Integrating AWS with Jenkins is achieved by configuring AWS credentials in Jenkins global settings, installing necessary plugins (such as AWS SDK and EC2), and creating Jenkins jobs that interact with AWS services.

**59. Explain Dockerfile and how it works.**

Answer: A Dockerfile is a text document containing instructions for building a Docker image. It specifies a base image, adds dependencies, and defines commands to run when the container starts. Docker reads the Dockerfile to build an image that can be executed as a container.

**60. Explain any 5 Docker commands.**

Answer: Common Docker commands include:

1. docker build: Builds a Docker image from a Dockerfile.
2. docker run: Creates and starts a Docker container.
3. docker ps: Lists running containers.
4. docker exec: Runs a command in a running container.
5. docker images: Lists available Docker images.

**61. Explain the end-to-end process to build an image and push it into a registry.**

Answer: The process involves:

1. Writing a Dockerfile with necessary instructions.
2. Building the image using docker build.
3. Tagging the image using docker tag.
4. Logging into the Docker registry.
5. Pushing the image to the registry using docker push.

**62. Explain COPY and ADD commands in Dockerfile.**

Answer: Both COPY and ADD commands in a Dockerfile are used to copy files into the image. The key difference is that ADD allows for additional features like URL support and automatic extraction of compressed files.

**63. Explain Dockerfile structure.**

Answer: A Dockerfile typically includes:

* A FROM instruction specifying the base image.
* RUN instructions for executing commands.
* COPY or ADD instructions to copy files into the image.
* CMD or ENTRYPOINT instruction defining the default command when the container starts.

**64. What is the Dockerfile format, and how should it be saved?**

Answer: The Dockerfile format is a plain text document. It should be saved with the filename “Dockerfile” (no file extension) and should be located in the root of the build context.

**65. Explain Kubernetes (k8s) architecture.**

Answer: Kubernetes architecture includes:

* Master Node: Controls the cluster and schedules tasks.
* Worker Nodes: Host containers and execute tasks.
* etcd: Stores configuration data.
* API Server: Exposes the Kubernetes API.
* Controller Manager: Enforces desired state.
* Scheduler: Assigns tasks to nodes.

**66. Why is Kubernetes called k8s?**

Answer: “K8s” is a shorthand notation for Kubernetes, derived by replacing the eight characters “ubernete” with the number 8. It simplifies the pronunciation and typing of the word.

**67. Explain the end-to-end process to create an EKS cluster in AWS EKS.**

Answer: Creating an EKS cluster involves:

1. Configuring the AWS CLI and EKS CLI.
2. Creating an Amazon EKS cluster using the eksctl command.
3. Configuring kubectl to communicate with the EKS cluster.

**68. What is a pod? How many containers can run in a single pod, and are there any limitations?**

Answer: A pod is the smallest deployable unit in Kubernetes, representing one or more containers. Multiple containers can run in a single pod, sharing the same network namespace. However, it is recommended to colocate containers that need to communicate closely. There are no strict limitations on the number of containers but consider resource constraints.

**69. Explain the complete end-to-end process of deploying pods in an EKS cluster.**

Answer: Deploying pods in an EKS cluster involves:

1. Creating a Kubernetes deployment manifest.
2. Applying the manifest using kubectl apply.
3. Monitoring pod status using kubectl get pods.
4. Troubleshooting any issues with logs and events.

**70. What is Helm?**

Answer: Helm is a package manager for Kubernetes that simplifies the deployment and management of applications. Helm uses charts, which are packages of pre-configured Kubernetes resources, making it easier to share and version application configurations.

**71. Any major troubleshooting you have done in Kubernetes?**

Answer: One common troubleshooting task in Kubernetes involves identifying and resolving issues with pod scheduling, resource constraints, or networking. Tools like kubectl describe and examining pod logs are often used in troubleshooting scenarios.

**72. After deployment, what actions do we need to perform in Kubernetes?**

Answer: Post-deployment actions may include:

* Monitoring pod health and resource usage.
* Scaling pods based on demand.
* Implementing rolling updates for application changes.
* Periodic reviews of configuration and security settings.

**73. How to rollback to the last stable version if a prod deployment fails?**

Answer: Rolling back to the last stable version in Kubernetes involves:

1. Identifying the previous stable release.
2. Using kubectl rollout undo to revert the deployment to the previous version.

**74. What is a DaemonSet?**

Answer: A DaemonSet in Kubernetes ensures that all (or some) nodes run a copy of a pod. It is useful for deploying system daemons or services, ensuring they run on every node in the cluster.

**75. Difference between ReplicaSet and Deployment.**

Answer: A ReplicaSet ensures a specified number of replicas of a pod are running, while a Deployment provides a higher-level abstraction, managing ReplicaSets and enabling rolling updates and rollbacks.

**76. If pods are not coming online post deployment, how to fix that?**

Answer: To fix pods not coming online:

1. Review pod logs for error messages.
2. Inspect the pod status using kubectl describe pod.
3. Verify network policies and service configurations.
4. Ensure resource constraints are not causing scheduling issues.

**77. How to export pod logs to CloudWatch or Grafana?**

Answer: Pod logs can be exported to CloudWatch or Grafana using tools like Fluentd or Fluent Bit. Set up a logging agent on each node to collect and forward logs to the desired monitoring platform.

**78. How to monitor pods in monitoring tools like Grafana or Prometheus?**

Answer: Monitoring pods in Grafana or Prometheus involves setting up appropriate dashboards and configuring data sources. Collect pod metrics, such as CPU and memory usage, and visualize them in the monitoring tools.

**79. Explain the end-to-end deployment cycle.**

Answer: The end-to-end deployment cycle includes:

1. Writing application code and configuring it for deployment.
2. Building a container image using tools like Docker.
3. Pushing the image to a container registry.
4. Deploying the image to a Kubernetes cluster using manifests or Helm charts.
5. Monitoring and managing the deployed application.

**80. What is CI and CD?**

Answer: CI (Continuous Integration) is the practice of automating code integration and testing. CD (Continuous Delivery/Continuous Deployment) extends CI by automating the entire software release process, from integration and testing to deployment and monitoring.

**81. In Jenkins, if we want to deploy particular jobs from a slave server, where can we make those changes?**

Answer: In Jenkins, to deploy specific jobs from a slave server, you can make changes in the job configuration by selecting the option to restrict where the project can be run. Specify the label of the slave node where the job should execute.

**82. If you receive a requirement to migrate a data center server to AWS, how would you plan for it?**

Answer: Planning a data center server migration to AWS involves:

1. Assessing current workloads and dependencies.
2. Identifying suitable AWS services.
3. Creating an incremental migration plan.
4. Testing the migration in a controlled environment.
5. Executing the migration with minimal downtime.
6. Validating post-migration performance and reliability.

**83. How to migrate an on-prem data center database to AWS RDS?**

Answer: Migrating an on-prem data center database to AWS RDS involves:

1. Assessing database compatibility with RDS.
2. Creating an RDS instance.
3. Extracting and transforming data.
4. Loading data into RDS.
5. Redirecting applications to use the RDS instance.

**84. What actions will you take if migration is completed for a database but post-migration you are facing data mismatch issues?**

Answer: If facing data mismatch issues post-migration:

1. Validate data consistency between source and destination.
2. Check for data transformation errors during migration.
3. Compare database schema and data types.
4. Consider rolling back to the previous state and reevaluating the migration process.

**85. Explain types of migration.**

Answer: Types of migration include:

* Rehosting (lift and shift)
* Refactoring or rearchitecting
* Replatforming or re-platforming
* Retire or retain

**86. How to migrate very large data, like a petabyte, from on-prem to AWS?**

Answer: Migrating very large data involves strategies such as parallel data transfers, using AWS Snowball or Snowmobile for offline transfers, and optimizing data transfer mechanisms. AWS DataSync and AWS Direct Connect can also be considered for efficient data migration.

**87. How to avoid major downtime during migration from on-prem to AWS?**

Answer: To avoid major downtime during migration:

1. Implement incremental migration strategies.
2. Conduct thorough testing in a staging environment.
3. Leverage AWS services that support live migrations.
4. Use DNS switchover techniques for minimal service interruption.

**88. How to handle application traffic worldwide to improve application performance?**

Answer: Handling application traffic worldwide involves:

1. Distributing application instances globally using Content Delivery Networks (CDNs).
2. Implementing Anycast or Geo DNS for routing traffic to the nearest server.
3. Leveraging AWS Global Accelerator or AWS CloudFront for edge locations.

**89. How to secure an S3 bucket?**

Answer: Securing an S3 bucket involves:

1. Configuring bucket policies to control access.
2. Enforcing fine-grained permissions using IAM roles.
3. Implementing bucket-level access controls.
4. Using AWS Identity and Access Management (IAM) roles and policies.

**90. How to reduce cost for AWS infrastructure?**

Answer: Reducing costs for AWS infrastructure involves:

1. Right-sizing instances and resources.
2. Utilizing reserved instances or savings plans.
3. Implementing auto-scaling for dynamic workloads.
4. Monitoring and optimizing resource usage.
5. Leveraging cost management tools like AWS Cost Explorer.

**91. What is the difference between a reserved instance and a savings plan? Which one is better?**

Answer: Reserved Instances (RIs) and Savings Plans (SPs) are cost-saving models for AWS. RIs require a commitment to a specific instance type in a specific region, while SPs offer more flexibility across instance types and regions. The choice depends on workload predictability and flexibility needs.

**92. How to save costs for Lambda functions? What is power tuning?**

Answer: Saving costs for Lambda functions involves optimizing memory allocation, execution time, and concurrency settings. Power tuning is the practice of finding the optimal balance between memory and execution time to minimize costs while maintaining performance.

**93. How to save costs for lower environments like dev and test infrastructure?**

Answer: To save costs for lower environments:

1. Use on-demand or spot instances based on workload characteristics.
2. Implement auto-scaling to adapt to varying resource requirements.
3. Leverage infrastructure as code to manage and provision resources efficiently.

**94. What are the AWS 6 pillars, and why should every architecture follow them?**

Answer: The AWS Well-Architected Framework defines six pillars:

1. Operational Excellence
2. Security
3. Reliability
4. Performance Efficiency
5. Cost Optimization
6. Sustainability Following these pillars ensures that architectures are designed for success, meeting key criteria for operational excellence, security, and efficiency.

**95. What is Terraform, and why should we use it?**

Answer: Terraform is an infrastructure as code (IaC) tool used for provisioning and managing infrastructure on various cloud platforms. It allows for declarative configuration, version control, and automation of infrastructure, promoting consistency and repeatability.

**96. Any limitations in Terraform?**

Answer: Some limitations of Terraform include:

1. Lack of built-in dependency handling between resources.
2. Limited support for dynamic values and expressions.
3. State file management complexities.
4. Limited native support for looping and conditionals.

**97. Difference between Ansible & Terraform?**

Answer: Ansible and Terraform serve different purposes. Ansible is a configuration management tool focusing on automating tasks on existing servers, while Terraform is an infrastructure provisioning tool designed for creating and managing infrastructure resources.

**98. Let’s say I have created resources like EC2, S3 bucket, and Lambda function using Terraform in AWS. When I am going to create other services like RDS or CloudFront, one of my Lambda functions got deleted. What could be the reason, and how to fix that?**

Answer: The deletion of a Lambda function while creating other resources may be due to changes in the Terraform configuration or dependencies between resources. To fix this, review the Terraform configuration, ensure dependencies are correctly defined, and use Terraform state to manage resource relationships.

**99. How to secure resources while creating them using Terraform?**

Answer: Securing resources in Terraform involves:

1. Using variables and expressions to parameterize sensitive information.
2. Storing sensitive data in secure locations.
3. Implementing role-based access controls for Terraform operations.
4. Encrypting Terraform state files using remote backends.

**100. How to give temporary access to someone for specific AWS resources, and how to configure that?**

Answer: Temporary access can be granted by creating an IAM (Identity and Access Management) user or role with an associated policy. The policy includes permissions for the required resources, and the user or role is configured with an expiration time, granting access for the specified duration. This ensures temporary and controlled access.

**101. You have a Linux machine with 20 GB storage, but it’s showing full even after deleting log files. How do you fix that?**

**Answer:**

1. Check Deleted Files: Verify if log files were successfully deleted.
2. Processes Holding Files: Identify any processes holding onto deleted files.
3. Filesystem Check: Run df -h to check the filesystem status.
4. Inodes Check: Ensure there are available inodes using df -i.
5. Logrotate Configuration: Review and optimize logrotate configurations.