W5 Question Bank

Cloud Computing Intro

Beginner-Level Questions (1–10)

1. **What is cloud computing, and how does it differ from traditional on-premises IT infrastructure?**
   Explain the basic idea of delivering computing resources over the internet.
2. **What is AWS Global Infrastructure, and why is it important for application reliability?**
   Describe the role of Regions and Availability Zones in simple terms.
3. **What is the difference between an AWS Region and an Availability Zone (AZ)?**
   Use an analogy (e.g., city vs. data center building) to clarify.
4. **What is the AWS Free Tier, and what are some common services included in it?**
   Explain its purpose for learning and testing.
5. **What is the AWS Management Console, and how do you access it?**
   Describe its role as a web-based interface for managing AWS resources.
6. **What is IAM in AWS, and what are its three main components?**
   Name Users, Groups, Roles, and Policies in simple terms.
7. **What is the AWS CLI, and why would a beginner use it instead of the console?**
   Explain command-line interaction with AWS services.
8. **What is Amazon EC2, and what is its primary use case?**
   Describe it as a virtual server in the cloud.
9. **What is an AMI (Amazon Machine Image), and why is it needed to launch an EC2 instance?**
   Compare it to a template or snapshot for creating servers.
10. **What is a Security Group in AWS, and how does it control traffic to an EC2 instance?**
    Explain inbound/outbound rules with a simple example (e.g., allowing HTTP).

Intermediate-Level Questions (11–25)

11. **How does AWS ensure high availability using Regions and Availability Zones? Give an example architecture.**
    Discuss deploying an application across multiple AZs in one Region.

12. **Explain the key differences between AWS Free Tier, pay-as-you-go, and Reserved Instances in terms of cost management.**
    Include when each pricing model is suitable.

13. **How do you navigate and manage resources using the AWS Management Console vs. AWS CLI?**
    Provide a real example: launching an EC2 instance using both methods.

14. **What are the core principles of IAM, and how do you apply the principle of least privilege?**
    Give an example of a policy that allows only S3 read access.

15. **Compare AWS EC2 instance types (e.g., t3, m5, c5) and explain when to use compute-optimized vs. memory-optimized instances.**
    Link instance families to real-world workloads.

16. **What is Autoscaling in AWS, and how does it work with EC2?**
    Explain target tracking, step scaling, and the role of CloudWatch.

17. **What is Amazon RDS, and how does it simplify database management compared to self-managed databases on EC2?**
    Highlight features like backups, patching, and Multi-AZ.

18. **Compare Amazon S3 and Amazon EBS in terms of use case, durability, and access patterns.**
    Explain object storage vs. block storage with examples.

19. **How do you choose between Multi-AZ and Read Replicas in RDS for high availability and performance?**
    Describe failover and read-heavy workload scenarios.

20. **What are the different S3 storage classes (e.g., Standard, Intelligent-Tiering, Glacier), and when would you use each?**
    Include cost and retrieval time trade-offs.

21. **How do you secure an EC2 instance using Security Groups and IAM Roles?**
    Explain why you'd attach a role instead of storing keys in the instance.

22. **What is the difference between stopping and terminating an EC2 instance? What happens to the data in each case?**
    Include the impact on EBS volumes and public IPs.

23. **Explain how to implement a highly available web application using EC2, Auto Scaling, and an Application Load Balancer across two AZs.**
    Sketch the architecture at a high level.

24. **What are some common database deployment strategies on AWS (e.g., single DB, read replicas, RDS Proxy)?**

Discuss scalability and connection management.

25. **How do you monitor costs in AWS, and what tools or features help prevent billing surprises?**
    Mention AWS Budgets, Cost Explorer, and Trusted Advisor.

AWS application Deployment and DB Integration

Beginner-Level Questions (1–10)

1. **What are the basic steps to launch an EC2 instance using the AWS Management Console?**
   Walk through instance type, AMI, key pair, and security group selection.

2. **What is SSH, and how do you use it to connect to a Linux EC2 instance after launch?**
   Explain the role of the `.pem` key file and the `ssh` command.

3. **What is a server in the context of web applications, and what basic configurations are needed post-EC2 launch?**
   Mention installing a web server (e.g., Apache/Nginx), opening port 80, and deploying a simple page.

4. **What is Amazon CloudWatch, and what are two basic metrics you can monitor for an EC2 instance?**
   Give examples like CPU utilization and network in/out.

5. **What is Amazon RDS, and how does it differ from installing MySQL directly on an EC2 instance?**
   Highlight managed backups, patching, and scaling.

6. **What are the steps to set up a basic RDS instance (e.g., MySQL) via the AWS Console?**
   Include DB engine, instance size, username/password, and VPC security.

7. **How does an application connect to an RDS database? What information is needed in the connection string?**
   Explain endpoint, port, database name, user, and password.

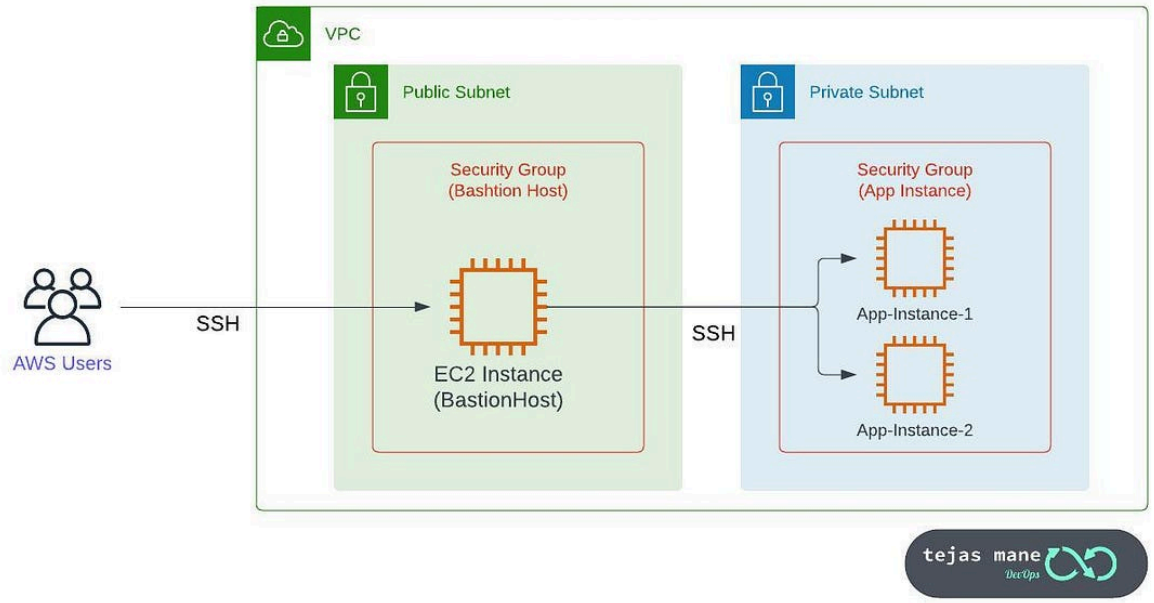
8. **What is a database backup in RDS, and what are the two types of backups AWS performs automatically?**
   Mention automated backups and snapshots.

9. **What is Multi-AZ deployment in RDS, and why would you enable it?**
   Use a simple analogy (e.g., standby server for failover).

10. **What is an S3 bucket, and what is its primary use case for static websites?**
    Explain storing HTML, CSS, JS files and enabling public access.

Intermediate-Level Questions (11–25)

11. **How do you securely connect to an EC2 instance without exposing SSH to the public internet?**
    Describe using a bastion host or AWS Systems Manager Session Manager. (Read about this if you have time..)



> https://www.reddit.com/r/aws/comments/szgbkq/regarding_the_purpose_of_a_bastion_host/

12. **Explain the difference between stopping and terminating an EC2 instance. What happens to root volume and public IP?**
    Include EBS behavior and Elastic IP usage.

13. **How do you configure CloudWatch to send an alarm when CPU utilization exceeds 80% for 5 minutes?**
    Walk through metric, threshold, period, and SNS notification.

14. **What are the key parameters to configure when launching an RDS instance for a production application?**
    Cover Multi-AZ, storage type (GP2 vs. IO1), backup retention, and parameter groups.

15. **How do you handle database credentials securely in an application running on EC2?**
    Compare environment variables, AWS Secrets Manager, and IAM DB authentication.

16. **What is the difference between automated backups and manual snapshots in RDS? When would you use each?**
    Include retention, cross-region copy, and restore process.

17. **Explain how Multi-AZ works under the hood in RDS. What happens during a failover?**
    Describe synchronous replication and automatic failover (include DNS update).

18. **What S3 bucket settings are required to host a static website? Name at least four.**
    Include static website hosting enablement, index/error documents, public read policy, and block public access override.

19. **How does CloudFront improve performance and security for an S3-hosted static website?**
    Explain edge locations, caching, HTTPS, and integration with S3.

20. **What is a CDN, and how does CloudFront distribute content globally?**
    Describe origin (S3), edge locations, and cache behavior.

21. **Compare S3 file upload methods: Console, CLI, and SDK. When would you use multipart upload?**
    Include use case for large files (>100 MB).

22. **What is the difference between monolithic and microservices deployment? How does AWS support both?**
    Mention EC2 for monolith vs. ECS/Fargate + ALB for microservices.

23. **Explain the role of a load balancer in a web application. What are the two types in AWS?**
    Compare Application Load Balancer (ALB) vs. Classic Load Balancer (ELB).

24. **How does Auto Scaling work with EC2 and a load balancer? Describe the components involved.**
    Include launch configuration/template, Auto Scaling group, scaling policies, and health checks.

25. **Design a scalable web architecture using EC2, Auto Scaling, ALB, RDS (Multi-AZ), and S3 + CloudFront. Label each component's role.**
    Sketch: ALB → ASG (EC2) → RDS, S3 → CloudFront for static assets.

> https://youtu.be/bWbsMUOm9w8

CICD Intro

Beginner-Level Questions (1–5)

1. **What is DevOps, and how does it differ from traditional software development and operations (siloed approach)?**
   Explain the goal of collaboration, automation, and faster delivery.

2. **What are the key cultural principles of DevOps? Name at least three practices that support it.**
   Include collaboration, shared responsibility, and continuous feedback.

3. **Explain the terms CI, CD, and IaC in simple language. Give one example of each.**

- CI: Build and test on every code change
- CD: Automatically deploy to production
- IaC: Manage servers via code (e.g., Terraform)

4. **What is a Git branching strategy, and why do teams use one in DevOps?**
   Mention consistency, collaboration, and release management.

5. **What is the purpose of a code review process in a DevOps team?**
   Explain quality, knowledge sharing, and catching bugs early.

Intermediate-Level Questions (6–12)

6. **Describe a typical Git workflow for a DevOps team using feature branches, pull requests, and protected main branch.**
   Walk through: `feature/login` → PR → review → merge → CI trigger.

7. **Compare GitFlow and GitHub Flow branching strategies. When would you use each?**
   Include complexity, release frequency, and team size.

8. **How does Git integrate with CI/CD pipelines? Explain the role of webhooks and automated triggers.**
   Example: Push to `main` → GitHub Actions runs tests and deploy.

9. **What are the main components of Jenkins architecture, and how does it execute a CI/CD pipeline?**
   Cover Master-Agent model, Jenkinsfile, stages (build, test, deploy).

10. **Explain the concept of a CI/CD pipeline. What are the typical stages, and what benefits does automation bring?**
    Stages: Checkout → Build → Test → Deploy → Notify.

11. **What is automated testing in a CI/CD pipeline? Compare unit, integration, and end-to-end tests with examples.**
    Include when each runs and tools (e.g., Jest, Postman, Cypress).

12. **How does Infrastructure as Code (IaC) enable deployment automation? Give an example using Terraform to deploy an EC2 instance.**
    Show `main.tf` with provider, resource, and `terraform apply`.
    (we will do this in the class.)

Docker

https://dockerlabs.collabnix.com/docker/cheatsheet/

Beginner-Level Questions (Focus on Fundamentals)

1. **What is a container in the context of software development?**
   Explain the basic purpose of containers and how they differ from traditional application deployment methods.

2. **How do containers compare to virtual machines in terms of resource usage and isolation?**
   Describe the key architectural differences and why one might be preferred over the other for lightweight applications.

3. **What are some primary benefits of using containers in application development?**
   List at least three benefits and provide a simple use case for each, such as development consistency.

4. **What is the container ecosystem, and what are its main components?**
   Give an overview of how tools like Docker fit into the broader ecosystem for building and running applications.

5. **What is Docker, and why is it a popular tool for containerization?**
   Explain Docker's role as a platform and its high-level architecture, including the daemon and client.

6. **How do you use Docker to create a simple container from an existing image?**
   Walk through the basic steps using the `docker run` command.

7. **What is the difference between a Docker image and a Docker container?**
   Use an analogy (e.g., blueprint vs. house) to clarify the relationship between the two.

8. **What stages make up the typical lifecycle of a container?**
   Describe the process from creation to cleanup, including key Docker commands for each stage.

9. **What is a Dockerfile, and what is its basic purpose?**
   Explain how it serves as a script for automating image builds.

10. **What does the `FROM` instruction do in a Dockerfile?**
    Provide an example and explain why it's usually the first instruction in a file.

Intermediate-Level Questions (Focus on Practical Concepts)

11. **How does Docker's client-server architecture work, and what is the role of the Docker daemon?**
    Describe the interaction between the Docker CLI, daemon, and runtime for managing containers.

12. **Explain how to manage multiple containers in a development environment using Docker Compose.**
    Outline the basics of a `docker-compose.yml` file for defining services, networks, and volumes.

13. **What are container registries, and how do Docker Hub and AWS ECR differ in usage?**
    Compare public vs. private registries and when to use each for image storage and sharing.

14. **How do you build a custom Docker image from a Dockerfile?**
    Step through the `docker build` process, including tagging and handling build contexts.

15. **What is image versioning in Docker, and why is it important for security and reliability?**
    Discuss best practices like semantic versioning and scanning for vulnerabilities.

16. **How do you handle persistent data in containers using volumes?**
    Explain Docker volumes vs. bind mounts, with an example of mounting a host directory.

17. **What challenges arise in multi-container applications, and how does orchestration help?**
    Describe scenarios like service dependencies and introduce basic orchestration concepts.

18. **How would you optimize a Dockerfile for faster builds and smaller image sizes?**
    Cover techniques like multi-stage builds and minimizing layers.

19. **What security best practices should be followed when working with Docker images?**
    Include running containers as non-root users and avoiding unnecessary privileges.

20. **Explain the concept of container networking and how Docker handles it by default.**
    Describe bridge networks and how to expose ports for inter-container communication.

21. **How do you debug a failing container, such as one that won't start?**
    Outline steps using `docker logs`, `docker inspect`, and interactive shells.

22. **What is the role of environment variables in Dockerfiles and containers?**
    Provide an example of using `ENV` in a Dockerfile and overriding it at runtime.

23. **Compare the use of `docker run` vs. `docker-compose up` for starting applications.**
    Discuss when each is appropriate for single vs. multi-service setups.

24. **How does image layering work in Docker, and why does it matter for efficiency?**
    Explain how changes create new layers and the impact on storage and rebuilds.

25. **What considerations are needed for deploying containers to a cloud registry like AWS ECR?**
    Cover authentication, pushing images, and integrating with CI/CD pipelines.