Sure Nani, here's a **simple AWS tutorial guide** made especially for **Java Full Stack Developers** like you. I've included plain text explanations, real-world usage, and common interview questions. The focus is on concepts you will likely use and get asked about in interviews.

---

**AWS TUTORIAL FOR JAVA FULL STACK DEVELOPERS**

**INTRODUCTION TO AWS**

Amazon Web Services (AWS) is a cloud platform that provides various services to build, deploy, and manage applications. Instead of buying physical servers, you can rent resources from AWS and pay only for what you use.

EXAMPLE: You are developing a Spring Boot app. You can deploy it on a virtual server (EC2), store data in a managed database (RDS), and host the frontend React app on S3.

---

1. EC2 – Elastic Compute Cloud

---

**What it is:** A virtual server in the cloud. You can install Java, run your Spring Boot backend, and deploy APIs here.

**Example:** Create a Linux EC2 instance, SSH into it, install Java, and run your JAR file using java -jar.

**Why it's used:**

- Host backend services (like REST APIs)
- Run custom software
- Use like your own server without owning hardware

**Interview Questions:**

- What is EC2 and how do you connect to it?
- How do you deploy a Spring Boot app on EC2?
- Difference between stop and terminate?

---

2. S3 – Simple Storage Service

---

**What it is:** A place to store files (images, PDFs, HTML, videos, etc.).

**Example:** Store frontend React build files in S3 and make it public to access the website.

**Why it's used:**

- Host static websites
- Store backups and media
- Store logs or reports from your application

**Interview Questions:**

- What is S3 and how do you upload files?
- Can you host a website on S3?
- What are buckets in S3?

---

3. RDS – Relational Database Service

---

**What it is:** A managed database like MySQL, PostgreSQL, or Oracle without needing to install or patch anything manually.

**Example:** Use RDS to store your application's data instead of running your own MySQL on EC2.

**Why it's used:**

- Fully managed database
- Automatic backup and recovery
- High availability with multi-AZ

**Interview Questions:**

- What is RDS and why is it better than installing DB on EC2?
- Can you connect to RDS from your local machine?
- What is Multi-AZ?

---

4. IAM – Identity and Access Management

---

**What it is:** A service to manage users, roles, and permissions in AWS.

**Example:** Give permission to a user to only access S3 and not EC2.

**Why it's used:**

- Secure access control
- Role-based permissions
- Share access without sharing passwords

**Interview Questions:**

- What is IAM?
- What is a role vs user in IAM?
- How do you secure your AWS resources?

---

5. VPC – Virtual Private Cloud

---

**What it is:** A private network for your AWS resources.

**Example:** You can create a VPC to isolate your backend EC2 instance and make it only accessible via a load balancer.

**Why it's used:**

- Secure networking
- Control traffic using security groups and network ACLs
- Deploy public/private subnets

**Interview Questions:**

- What is VPC and why is it used?
- What is the difference between public and private subnet?
- What are security groups?

---

6. Elastic Beanstalk

---

**What it is:** A service to quickly deploy applications like Java, Node.js without manually setting up EC2, Load Balancer, etc.

**Example:** Deploy a Spring Boot WAR or JAR using Elastic Beanstalk and it handles everything for you.

**Why it's used:**

- Fast deployment
- Auto-scaling and monitoring
- Easy to manage apps without knowing full infra

**Interview Questions:**

- What is Elastic Beanstalk and how does it work?
- How is it different from EC2?

---

7. CloudWatch

---

**What it is:** A monitoring tool to track logs, metrics, and alerts from AWS services.

**Example:** View logs from your Spring Boot app deployed in EC2 using CloudWatch.

**Why it's used:**

- Troubleshooting application issues
- Monitoring performance
- Setting up alerts on high CPU, low disk, etc.

**Interview Questions:**

- What is CloudWatch?
- How do you monitor logs of your EC2 app?

---

8. Route 53

---

**What it is:** AWS's DNS service to manage domain names and routing traffic to your app.

**Example:** Map your domain www.example.com to point to your app running on EC2.

**Why it's used:**

- Manage custom domains
- Redirect traffic
- Health checks for endpoints

**Interview Questions:**

- What is Route 53?
- How do you point a domain to an EC2 instance?

---

9. Docker on AWS (ECS, EKS)

---

**What it is:** You can run Docker containers on AWS using ECS (Elastic Container Service) or EKS (Kubernetes).

**Example:** Package your Spring Boot app in a Docker container and deploy it to ECS.

**Why it's used:**

- Run containers in scalable way
- Use microservices architecture
- Reduce deployment issues

**Interview Questions:**

- How do you deploy Docker containers in AWS?
- What is ECS and how is it different from EC2?

---

10. CodePipeline + GitHub + EC2

---

**What it is:** Automate deployment whenever you push code to GitHub.

**Example:** When you push to GitHub, it builds the app and deploys to EC2 automatically.

**Why it's used:**

- CI/CD (Continuous Integration/Deployment)
- Automation
- Faster release cycles

**Interview Questions:**

- What is CodePipeline?
- How do you implement CI/CD in AWS?

---

**ECS – Elastic Container Service**

What it is: ECS allows you to run Docker containers on AWS. You can package your Java Spring Boot application into a Docker image and deploy it easily.

Example:

1. Create a Docker image of your Spring Boot app.
2. Push it to Amazon ECR (Elastic Container Registry).
3. Use ECS to deploy and run that container.

Why it's used:

- No need to manage EC2 directly
- Easily run containers with auto-scaling
- Use with load balancer and networking

Key Terms:

- Task: A running container

- Service: A group of tasks
- Cluster: Group of resources to run services

Interview Questions:

- What is ECS?
- How do you deploy Docker apps using ECS?
- What is the difference between ECS and EC2?

---

**EKS – Elastic Kubernetes Service**

What it is: EKS lets you run Kubernetes clusters on AWS. Kubernetes is used to manage containers (like Docker) in large-scale systems.

Why it's used:

- Run microservices architecture
- Manage many containers together
- Auto-scaling, self-healing

When to use EKS instead of ECS:

- You already know or use Kubernetes
- You need more control and custom features
- You're using cloud-native tools like Helm charts

Interview Questions:

- What is the difference between ECS and EKS?
- When would you choose EKS over ECS?
- What is a pod in Kubernetes?

---

**API Gateway + Lambda (Serverless Architecture)**

API Gateway:
This lets you create and expose REST APIs. It works as an entry point for client requests.

Lambda:
This runs code without a server. You write a function and AWS runs it when triggered. You only pay when it runs.

Example:

1. Create an API endpoint with API Gateway.
2. Connect it to a Lambda function written in Java, Python, or Node.js.
3. The function fetches data from DynamoDB or sends a response.

Use Case:
You want to write simple APIs without managing any backend server (no EC2).

Why it's used:

- Serverless = no need to manage infrastructure
- Cost effective for low traffic apps
- Auto-scaling out of the box

Interview Questions:

- What is API Gateway?
- What is AWS Lambda and how is it different from EC2?
- What is cold start in Lambda?

---

**CloudFront – Content Delivery Network (CDN)**

What it is: CloudFront caches your content (images, static HTML, React build files) at edge locations close to the user.

Why it's used:

- Faster access for global users
- Reduced load on origin server
- Works well with S3 and EC2

Example:
Host your frontend React app in S3. Use CloudFront to serve it faster to users all over the world.

Key Terms:

- Origin: Where the data comes from (like S3 or EC2)
- Edge Location: Global caching servers
- Distribution: Configuration that tells CloudFront what and how to deliver

Interview Questions:

- What is CloudFront and when do you use it?
- What is the difference between CloudFront and S3?
- What is an edge location?

---

REAL-TIME SCENARIO

Let's say you build an e-commerce site using Java Spring Boot and React.

- Use EC2 to host your Spring Boot API
- Use RDS for the product/order database
- Use S3 to host the React frontend
- Use CloudFront to cache and deliver the frontend fast
- Use API Gateway + Lambda for email/SMS OTP APIs
- Use ECS to deploy microservices as Docker containers
- Use CodePipeline for CI/CD with GitHub
- Monitor logs with CloudWatch
- Secure access using IAM
- Use Route 53 to map domain name to CloudFront or EC2

---

**SUMMARY**

As a Java full stack developer, you will mostly use:

- EC2 (to host backend)
- S3 (to host frontend)
- RDS (for database)
- IAM (for secure access)
- CloudWatch (to see logs)
- CodePipeline (for CI/CD)
- ECS/EKS (for Docker containers)
- API Gateway + Lambda (for serverless)
- CloudFront (for caching/static content)