**1. AWS Services**

* **AWS Free Tier**: [AWS Free Tier](https://aws.amazon.com/free/)
* **AWS Documentation**: [AWS Docs](https://docs.aws.amazon.com/)
* **AWS Services Overview**: [AWS Services Overview](https://aws.amazon.com/products/)
* **AWS Hands-on Labs**: [AWS Skill Builder](https://aws.amazon.com/training/)
* **EC2 Deep Dive**: [EC2 Documentation](https://docs.aws.amazon.com/ec2/)
* **IAM & Security Best Practices**: [IAM Best Practices](https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html)
* **AWS Well-Architected Framework**: [AWS WA Framework](https://aws.amazon.com/architecture/well-architected/)
* **Practice Questions**: AWS Quiz

**2. React.js & Node.js (Express.js)**

**React.js**

* **React Docs**: [React.js Official](https://react.dev/)
* **Advanced React Patterns**: [React Patterns](https://reactpatterns.com/)
* **React Performance Optimization**: React Performance
* **State Management (Redux, Zustand, Recoil)**: Redux Toolkit

**Node.js (Express.js)**

* **Node.js Docs**: Node.js Official
* **Express.js Guide**: [Express.js Docs](https://expressjs.com/)
* **Middleware & Authentication**: Auth in Express
* **Building REST APIs**: [RESTful API Design](https://restfulapi.net/)

**3. MongoDB (Backup, Replication, Performance)**

* **MongoDB Docs**: [MongoDB Official Docs](https://www.mongodb.com/docs/)
* **Backup & Restore**: [Backup Strategies](https://www.mongodb.com/docs/manual/core/backups/)
* **Replication & Sharding**: [MongoDB Replication](https://www.mongodb.com/docs/manual/replication/)
* **Performance Optimization**:
  + **Indexes**: [MongoDB Indexing](https://www.mongodb.com/docs/manual/indexes/)
  + **Aggregation Framework**: [MongoDB Aggregation](https://www.mongodb.com/docs/manual/aggregation/)
  + **Schema Design Best Practices**: [Schema Design](https://www.mongodb.com/docs/manual/core/data-models/)
* **Practice Problems**: [MongoDB University](https://university.mongodb.com/)

**4. CI/CD Pipelines (GitHub Actions)**

* **GitHub Actions Docs**: [GitHub Actions](https://docs.github.com/en/actions)
* **Setting Up CI/CD for Node.js**: GitHub Actions for Node
* **Automating Tests & Deployments**: Testing with GitHub Actions
* **AWS Deployment via GitHub Actions**: [Deploy to AWS](https://github.com/marketplace/actions/deploy-to-aws)

**5. Infrastructure-as-Code (CloudFormation & Terraform)**

**Terraform**

* **Terraform Docs**: Terraform Official
* **Hands-on Terraform**: Terraform Up & Running
* **AWS with Terraform**: Terraform AWS Guide

**CloudFormation**

* **CloudFormation Docs**: [AWS CloudFormation](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/)
* **Templates & Best Practices**: [CloudFormation Templates](https://aws.amazon.com/cloudformation/resources/templates/)
* **Hands-on Labs**: [AWS Workshops](https://workshops.aws/)

**Mock Interviews & Practice Questions**

* **System Design & DSA**: [LeetCode](https://leetcode.com/)
* **Backend Interview Prep**: [Backend Interview Handbook](https://backendinterview.com/)
* **AWS Scenario-Based Questions**: [AWS FAQs](https://aws.amazon.com/faqs/)
* **Full-Stack Interview Guide**: [Full-Stack Interviews](https://www.fullstack.cafe/)

**Interview Preparation Guide**

**1. AWS Interview Questions**

**EC2 & S3**

* What are the different types of EC2 instances, and how do you choose the right one?
* How do you set up auto-scaling for an EC2 instance?
* What are the different storage classes in S3, and when would you use each?
* How do you secure an S3 bucket from unauthorized access?

**VPC & Networking**

* What is the difference between a public and private subnet in AWS?
* How do you set up a VPC with a NAT gateway?
* How does security group vs. NACL work in AWS networking?

**IAM & Security**

* How do you enforce least privilege in AWS IAM?
* What’s the difference between IAM roles and IAM policies?
* How do you implement multi-account IAM security using AWS Organizations?

**CloudWatch & Monitoring**

* How do you set up custom CloudWatch metrics for an application?
* What is AWS X-Ray, and how does it help in monitoring microservices?

**2. React.js & Node.js Interview Questions**

**React.js**

* How does React's reconciliation process work?
* What are React hooks, and how do they work internally?
* How do you optimize React applications for performance?
* Explain how React.memo() and useMemo() work.

**Node.js & Express.js**

* How does the event loop work in Node.js?
* What are middleware functions in Express.js?
* How do you handle real-time communication in a Node.js app?
* What’s the difference between process.nextTick() and setImmediate()?

**3. MongoDB Interview Questions**

* How does MongoDB handle high availability and failover?
* What’s the difference between replica sets and sharding?
* How do you improve query performance in MongoDB?
* How do you handle data migrations in MongoDB?

**4. CI/CD (GitHub Actions) Interview Questions**

* How does GitHub Actions work, and how do you define a workflow?
* How would you set up a CI/CD pipeline for a Node.js + React app?
* How do you manage secrets in GitHub Actions?
* What are self-hosted runners, and when would you use them?

**5. Infrastructure-as-Code (Terraform & CloudFormation) Questions**

* What is the difference between Terraform and CloudFormation?
* How does Terraform state management work?
* How do you handle multi-environment deployments with Terraform?
* How do you write reusable Terraform modules?

**Project Ideas for Hands-on Learning**

**1. AWS-Powered URL Shortener**

**Tech Stack**: React.js, Node.js (Express), DynamoDB/MongoDB, AWS Lambda, API Gateway  
**Features**:

* Shorten URLs and store them in DynamoDB
* AWS Lambda + API Gateway for serverless execution
* CloudWatch logs for monitoring

**2. Serverless Chat App**

**Tech Stack**: React.js, Node.js, AWS WebSockets API, DynamoDB  
**Features**:

* Real-time messaging with WebSockets
* User authentication with AWS Cognito
* DynamoDB for chat history storage

**3. CI/CD Pipeline for React + Express App**

**Tech Stack**: GitHub Actions, AWS S3, EC2, Terraform  
**Features**:

* GitHub Actions workflow to deploy the app
* Infrastructure-as-Code (Terraform) to provision resources

**4. AWS-Powered E-Commerce API**

**Tech Stack**: Express.js, MongoDB, AWS S3, Terraform  
**Features**:

* User authentication with JWT
* Image storage on AWS S3
* Deploy API using AWS ECS with Terraform

**5. Full-Stack Monitoring Dashboard**

**Tech Stack**: React.js, Node.js, CloudWatch, Prometheus  
**Features**:

* Track API response times and errors
* Custom CloudWatch dashboards
* Alerts with SNS

**Additional Resources and Links**

* [AWS Documentation](https://docs.aws.amazon.com/)
* [React Official Docs](https://reactjs.org/docs/getting-started.html)
* [Node.js Official Docs](https://nodejs.org/en/docs/)
* [MongoDB Documentation](https://www.mongodb.com/docs/)
* [GitHub Actions Documentation](https://docs.github.com/en/actions)
* [Terraform Documentation](https://developer.hashicorp.com/terraform/docs)
* [AWS CloudFormation Docs](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/)

Would you like further customization or additional resources?