**https://github.com/suvaatnbu/Orchestration-Scaling**

**Description**

**Notes:**

Jenkins Credentials:

URL: <http://3.111.188.91:8080/>

**Username:**herovired

Password: herovired

Project Link: <https://github.com/UnpredictablePrashant/SampleMERNwithMicroservices>

Fork this repository. For the update from the main repository, you can refer to this link:

<https://stackoverflow.com/questions/3903817/pull-new-updates-from-original-github-repository-into-forked-github-repository>

Project Steps:  
  
Step 1: Set Up the AWS Environment  
  
1. Set Up AWS CLI and Boto3:

   - Install AWS CLI and configure it with AWS credentials.

   - Install Boto3 for Python and configure it.

Step 2: Prepare the MERN Application

1. Containerize the MERN Application:

   - Ensure the MERN application is containerized using Docker. Create a Dockerfile for each component (frontend and backend).

2. Push Docker Images to Amazon ECR:

   - Build Docker images for the frontend and backend.

   - Create an Amazon ECR repository for each image.

   - Push the Docker images to their respective ECR repositories.

Step 3: Version Control

1. Use AWS CodeCommit:

   - Create a CodeCommit repository.

   - Push the MERN application source code to the CodeCommit repository.

Step 4: Continuous Integration with Jenkins

**1. Set Up Jenkins:**

   - Install Jenkins on an EC2 instance.

   - Configure Jenkins with necessary plugins.

2. Create Jenkins Jobs:

   - Create Jenkins jobs for building and pushing Docker images to ECR.

   - Trigger the Jenkins jobs whenever there's a new commit in the CodeCommit repository.

Step 5: Infrastructure as Code (IaC) with Boto3

1. Define Infrastructure with Boto3 (Python Script):

   - Use Boto3 to define the infrastructure (VPC, subnets, security groups).

   - Define an Auto Scaling Group (ASG) for the backend.

   - Create AWS Lambda functions if needed.

Step 6: Deploying Backend Services

1. Deploy Backend on EC2 with ASG:

   - Use Boto3 to deploy EC2 instances with the Dockerized backend application in the ASG.

Step 7: Set Up Networking

1. Create Load Balancer:

   - Set up an Elastic Load Balancer (ELB) for the backend ASG.

2. Configure DNS:

   - Set up DNS using Route 53 or any other DNS service.

Step 8: Deploying Frontend Services

1. Deploy Frontend on EC2:

   - Use Boto3 to deploy EC2 instances with the Dockerized frontend application.

Step 9: AWS Lambda Deployment

1. Create Lambda Functions:

- Use Boto3 to create AWS Lambda functions for specific tasks within the application.

- Backup of Db using Lambda Functions and store in S3 bucket - put time stamping on the backup

Step 10: Kubernetes (EKS) Deployment

1. Create EKS Cluster:

   - Use eksctl or other tools to create an Amazon EKS cluster.

2. Deploy Application with Helm:

   - Use Helm to package and deploy the MERN application on EKS.

Step 11: Monitoring and Logging

1. Set Up Monitoring:

   - Use CloudWatch for monitoring and setting up alarms.

2. Configure Logging:

   - Use CloudWatch Logs or another logging solution for collecting logs.

Step 12: Documentation

1. Document the Architecture:

 - Instruct learners to create documentation for the entire architecture and deployment process.

 - Put everything on the GitHub

Step 13: Final Checks

1. Validate the Deployment:

   - Ensure that the MERN application is accessible and functions correctly.

BONUS: ChatOps

Step 14: ChatOps Integration

Create SNS Topics:

1. Use Boto3 to create SNS topics for different events (e.g., deployment success, failure).
2. Create Lambda for ChatOps:
   1. Write a Lambda function that sends notifications to the appropriate SNS topics based on deployment events.
   2. Integrate ChatOps with Messaging Platform:
   3. Configure integrations with a messaging platform (e.g., Slack/MS Teams/ Telegram) to receive notifications from SNS.
   4. Configure SES

Submission Instructions:

To submit your assignment, please follow these guidelines:

- Ensure that your assignment is fully completed.

- Push your code to a GitHub repository.

- Share the repository link by including it in a text, Word, or PDF file format.

Submit the file/text containing the repository link via Vlearn.













