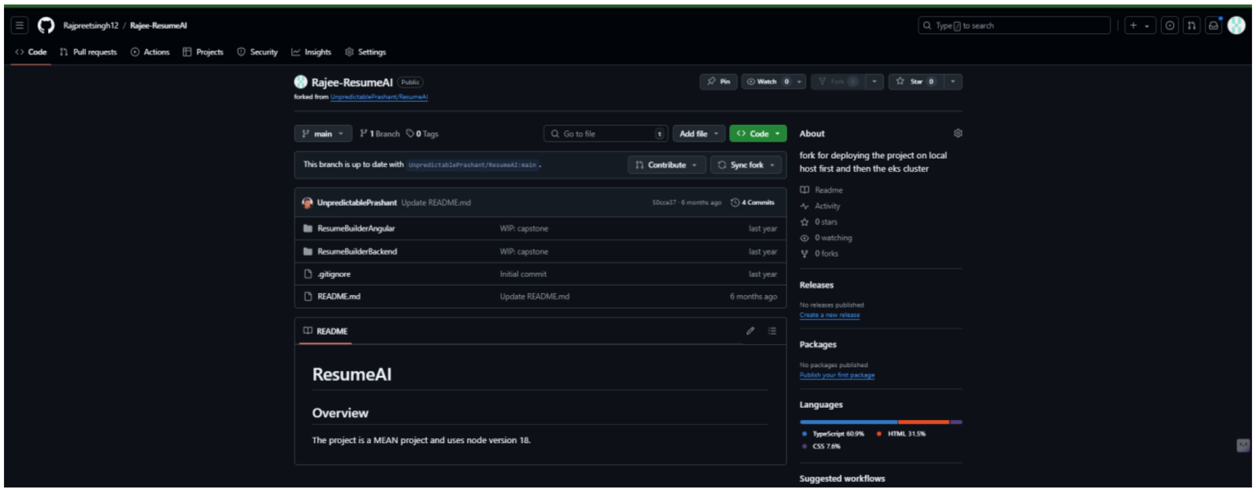
Practice Assignment on MEAN Project: -

1) Create a fork of the project with a different name: -



Rajee-ResumeAI Public
forked from UnpredictablePrashant/ResumeAI

This branch is up to date with UnpredictablePrashant/ResumeAI:main.

About
fork for deploying the project on local host first and then the eks cluster

UnpredictablePrashant Update README.md	50cce37 · 6 months ago	4 Commits
ResumeBuilderAngular	WIP: capstone	last year
ResumeBuilderBackend	WIP: capstone	last year
.gitignore	Initial commit	last year
README.md	Update README.md	6 months ago

ResumeAI

Overview

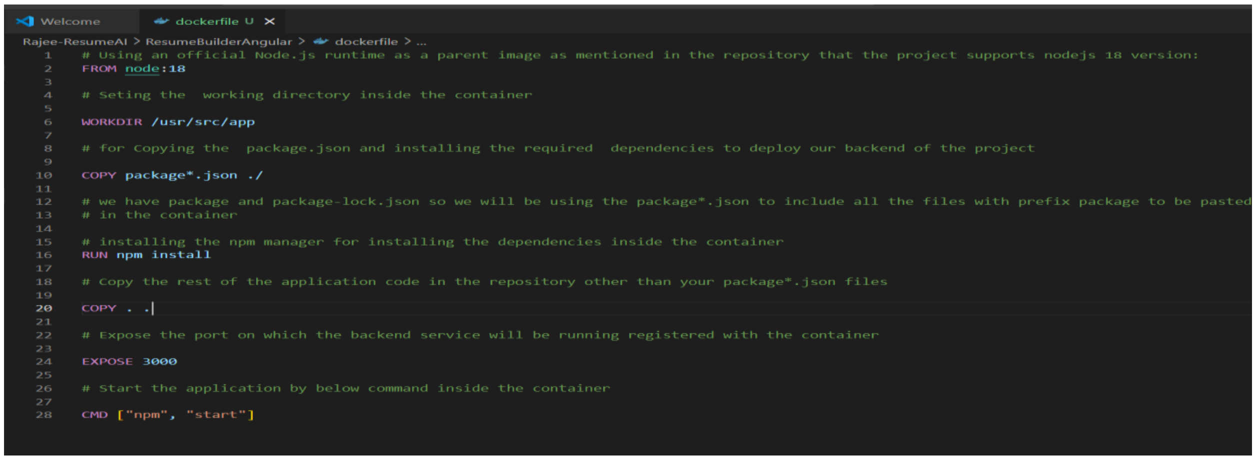
The project is a MEAN project and uses node version 18.

Languages: TypeScript 60.9% HTML 31.5% CSS 7.6%

2) Write a docker file for each component (e.g front-end and backend both):-

2.1 Backend Docker file: -

Create a `Dockerfile` for the backend service in the root of your backend folder:



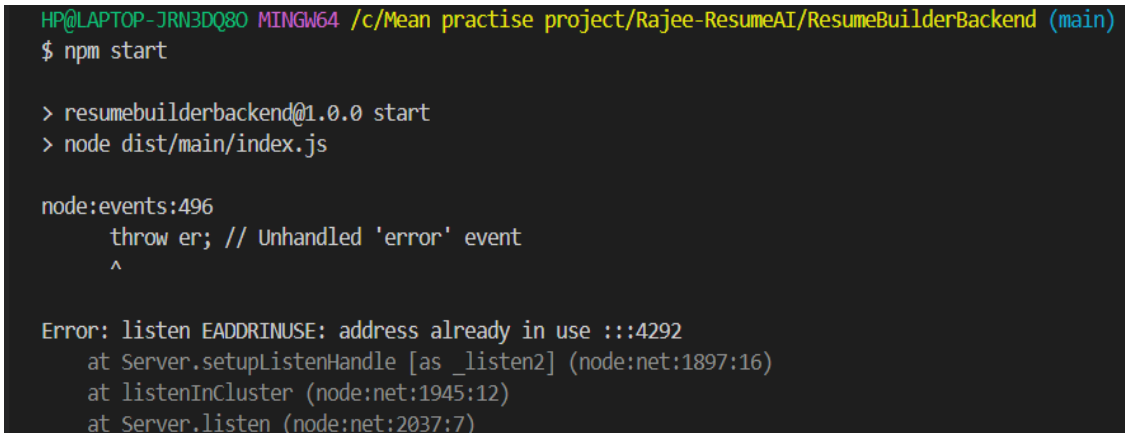
Rajee-ResumeAI > ResumeBuilderAngular > dockerfile > ...

```dockerfile
# Using an official Node.js runtime as a parent image as mentioned in the repository that the project supports nodejs 18 version:
FROM node:18

# Seting the  working directory inside the container

WORKDIR /usr/src/app

# for Copying the  package.json and installing the required  dependencies to deploy our backend of the project

COPY package*.json ./

# we have package and package-lock.json so we will be using the package*.json to include all the files with prefix package to be pasted
# in the container

# installing the npm manager for installing the dependencies inside the container
RUN npm install

# Copy the rest of the application code in the repository other than your package*.json files

COPY . .

# Expose the port on which the backend service will be running registered with the container

EXPOSE 3000

# Start the application by below command inside the container

CMD ["npm", "start"]
```

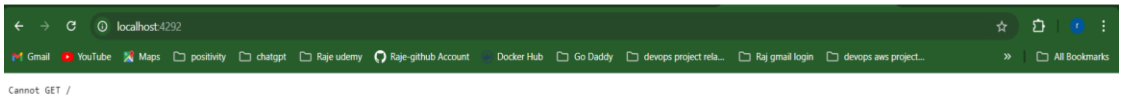
NOte:- first check if the application is running on local via npm install

1. by hitting npm install to install the necessary package managers for nodejs
2. for running the build process of the index.js file
3. now, you need to check the ls dist/main/index.js file if the build index.js exists.
4. Now you need to hit the npm start to start the nodejs for running your backend Application.

5.if you got the output as below which means the backend is allotted with the port 4292 you can refer below ss for the more clarity:-



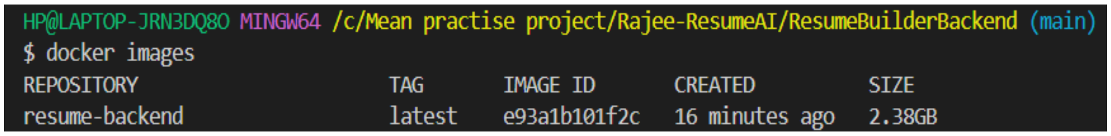
```
HP@LAPTOP-JRN3DQ80 MINGW64 /c/Mean practise project/Rajee-ResumeAI/ResumeBuilderBackend (main)
$ npm start

> resumebuilderbackend@1.0.0 start
> node dist/main/index.js

node:events:496
      throw er; // Unhandled 'error' event
      ^

Error: listen EADDRINUSE: address already in use :::4292
    at Server.setupListenHandle [as _listen2] (node:net:1897:16)
    at listenInCluster (node:net:1945:12)
    at Server.listen (node:net:2037:7)
```



```
Cannot GET /
```

6.Now you need to dockerize the bakend image by building the docker file present in the backend root folder: -

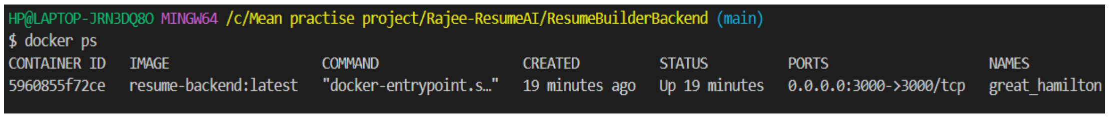
6.1 Hit the command “docker build –t resume-backend:latest “.



```
HP@LAPTOP-JRN3DQ80 MINGW64 /c/Mean practise project/Rajee-ResumeAI/ResumeBuilderBackend (main)
$ docker images
REPOSITORY                     TAG       IMAGE ID       CREATED          SIZE
resume-backend                 latest    e93a1b101f2c   16 minutes ago   2.38GB
```

6.2 Now you need to containerize your docker image by hitting below command:

“Docker run –p 3000:3000 resume-backend:latest”



```
HP@LAPTOP-JRN3DQ80 MINGW64 /c/Mean practise project/Rajee-ResumeAI/ResumeBuilderBackend (main)
$ docker ps
CONTAINER ID   IMAGE                   COMMAND                  CREATED          STATUS          PORTS                    NAMES
5960855f72ce   resume-backend:latest   "docker-entrypoint.s…"   19 minutes ago   Up 19 minutes   0.0.0.0:3000->3000/tcp   great_hamilton
```

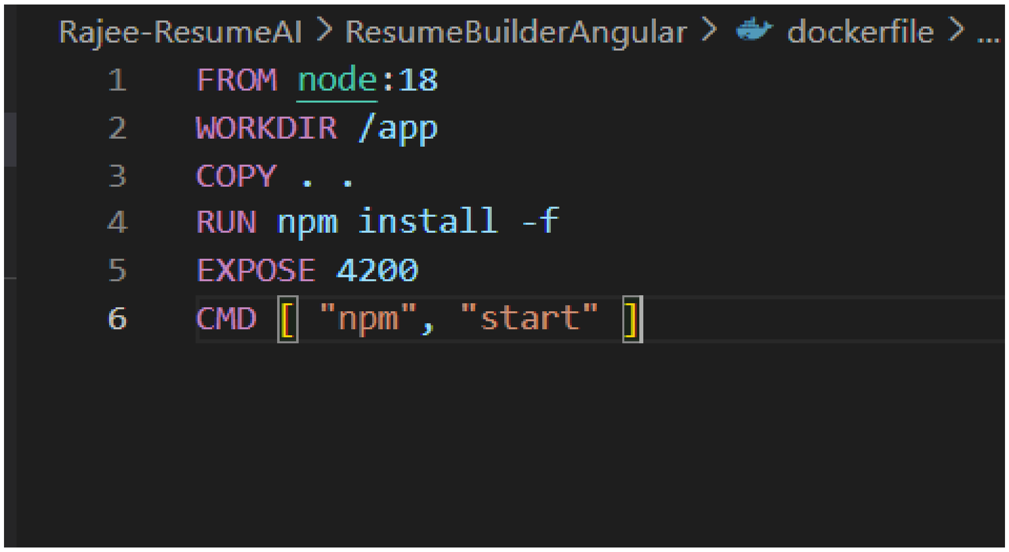
As you can you can see the above ss show that the container is running on the port 3000 or 4292 you can check the same in your browser by hitting the below: -

e.g:- localthost:4292

3)Frontend Dockerfile:-

Create a `Docker file` for the frontend service:

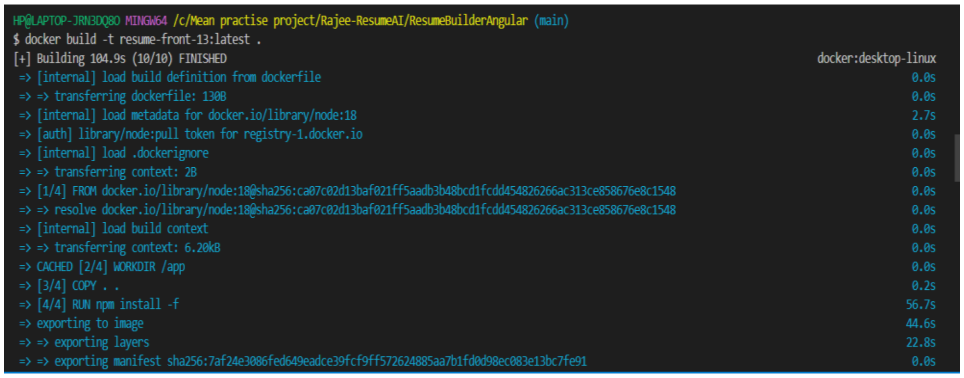
Now you need to navigate to your frontend directory for wiriting the below dockerfile with detailed layers: -



Rajee-ResumeAI > ResumeBuilderAngular > dockerfile > ...

```dockerfile
FROM node:18
WORKDIR /app
COPY . .
RUN npm install -f
EXPOSE 4200
CMD [ "npm", "start" ]
```

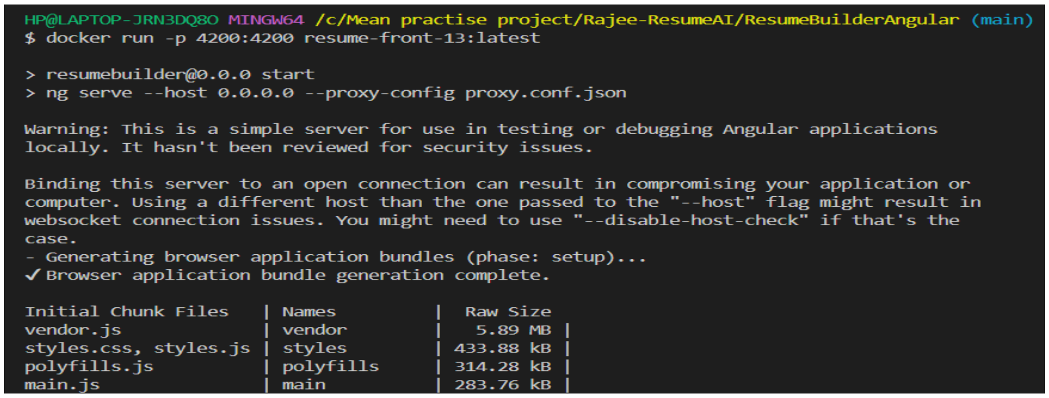
For building the above docker image you need to hit the below command “Docker build –t resumeai-frontend:latest .”



```
HP@LAPTOP-JRN3DQ8O MINGW64 /c/Mean practise project/Rajee-ResumeAI/ResumeBuilderAngular (main)
$ docker build -t resume-front-13:latest .
[+] Building 104.9s (10/10) FINISHED                                                              docker:desktop-linux
 => [internal] load build definition from dockerfile                                                              0.0s
 => => transferring dockerfile: 130B                                                                              0.0s
 => [internal] load metadata for docker.io/library/node:18                                                        2.7s
 => [auth] library/node:pull token for registry-1.docker.io                                                       0.0s
 => [internal] load .dockerignore                                                                                 0.0s
 => => transferring context: 2B                                                                                   0.0s
 => [1/4] FROM docker.io/library/node:18@sha256:ca07c02d13baf021ff5aadb3b48bcd1fcdd454826266ac313ce858676e8c1548  0.0s
 => => resolve docker.io/library/node:18@sha256:ca07c02d13baf021ff5aadb3b48bcd1fcdd454826266ac313ce858676e8c1548  0.0s
 => [internal] load build context                                                                                 0.0s
 => => transferring context: 6.20kB                                                                               0.0s
 => CACHED [2/4] WORKDIR /app                                                                                     0.0s
 => [3/4] COPY . .                                                                                                0.2s
 => [4/4] RUN npm install -f                                                                                     56.7s
 => exporting to image                                                                                           44.6s
 => => exporting layers                                                                                          22.8s
 => => exporting manifest sha256:7af24e3086fed649eadce39fcf9ff572624885aa7b1fd0d98ec083e13bc7fe91                0.0s
```

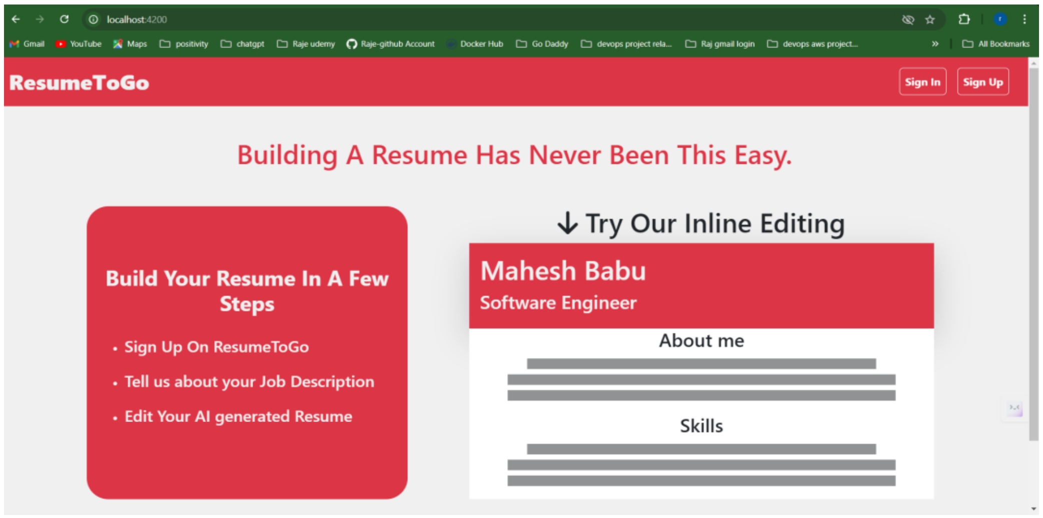
Now you need to containerize the above docker image by hitting the below

Command “docker run –p 4200:4200 resumeai-frontend:latest”



```
HP@LAPTOP-JRN3DQ8O MINGW64 /c/Mean practise project/Rajee-ResumeAI/ResumeBuilderAngular (main)
$ docker run -p 4200:4200 resume-front-13:latest

> resumebuilder@0.0.0 start
> ng serve --host 0.0.0.0 --proxy-config proxy.conf.json

Warning: This is a simple server for use in testing or debugging Angular applications
locally. It hasn't been reviewed for security issues.

Binding this server to an open connection can result in compromising your application or
computer. Using a different host than the one passed to the "--host" flag might result in
websocket connection issues. You might need to use "--disable-host-check" if that's the
case.
- Generating browser application bundles (phase: setup)...
✓ Browser application bundle generation complete.

Initial Chunk Files     | Names          |  Raw Size
vendor.js               | vendor         |   5.89 MB |
styles.css, styles.js   | styles         | 433.88 kB |
polyfills.js            | polyfills      | 314.28 kB |
main.js                 | main           | 283.76 kB |
```

Note: - for accessing your frontend over the local host you need to hit below command “localhost:42000”



Building A Resume Has Never Been This Easy.

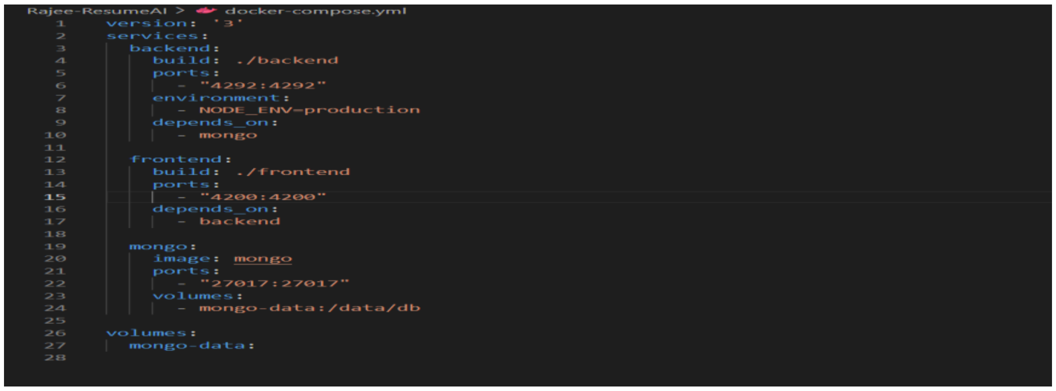
Build Your Resume In A Few Steps

- Sign Up On ResumeToGo
- Tell us about your Job Description
- Edit Your AI generated Resume

↓ Try Our Inline Editing

Mahesh Babu
Software Engineer
About me
Skills

3. **Write a `docker-compose.yml` File**

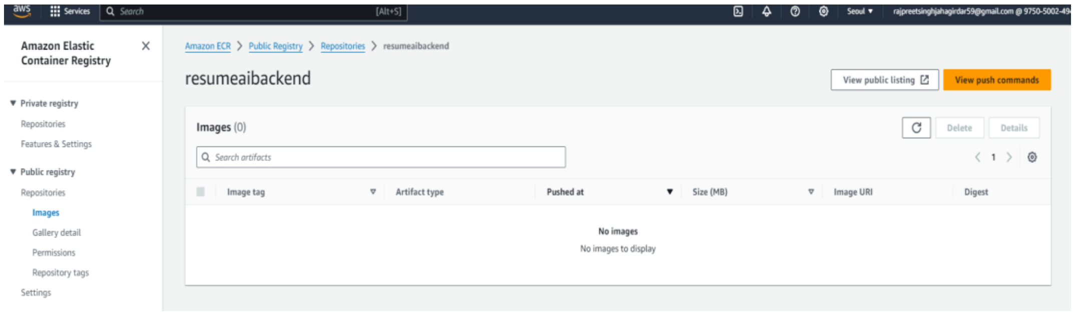


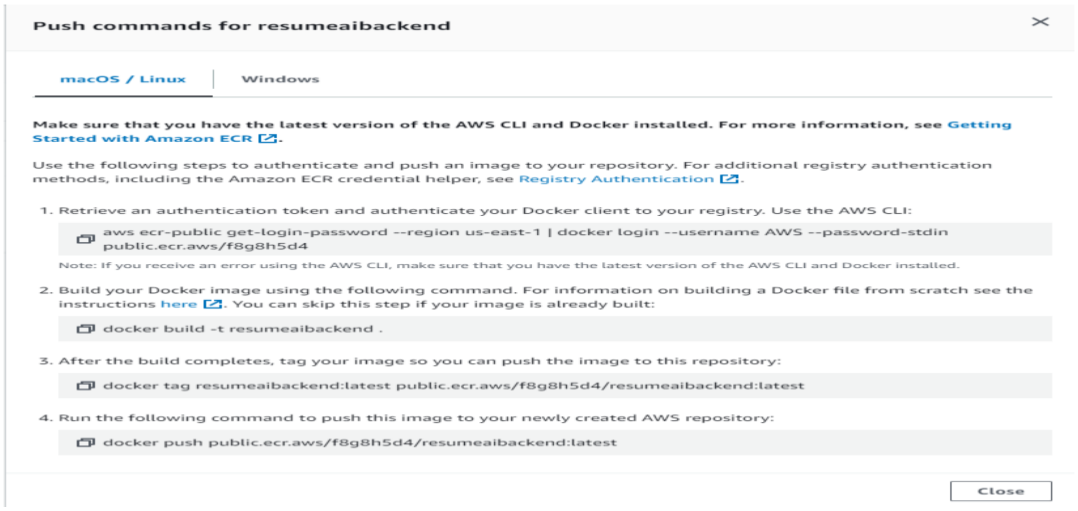
```
Rajee-ResumeAI > docker-compose.yml
version: '3'
services:
  backend:
    build: ./backend
    ports:
      - "4292:4292"
    environment:
      - NODE_ENV-production
    depends_on:
      - mongo

  frontend:
    build: ./frontend
    ports:
      - "4200:4200"
    depends_on:
      - backend

  mongo:
    image: mongo
    ports:
      - "27017:27017"
    volumes:
      - mongo-data:/data/db

volumes:
  mongo-data:
```

4. **Push the Docker Images into ECR: -**

4.1 pushing the frontend docker image into its repo on ecr:-

Note:-for pushing the docker images build on local to your ecr repo on aws you need to follow the steps by clicking the view push command just in the top right corner of the each ecr repo: -





Push commands for resumeaibackend

macOS / Linux | Windows

Make sure that you have the latest version of the AWS CLI and Docker installed. For more information, see Getting Started with Amazon ECR.

Use the following steps to authenticate and push an image to your repository. For additional registry authentication methods, including the Amazon ECR credential helper, see Registry Authentication.

1. Retrieve an authentication token and authenticate your Docker client to your registry. Use the AWS CLI:

```
aws ecr-public get-login-password --region us-east-1 | docker login --username AWS --password-stdin
public.ecr.aws/f8g8h5d4
```

Note: If you receive an error using the AWS CLI, make sure that you have the latest version of the AWS CLI and Docker installed.

2. Build your Docker image using the following command. For information on building a Docker file from scratch see the instructions here. You can skip this step if your image is already built:

```
docker build -t resumeaibackend .
```

3. After the build completes, tag your image so you can push the image to this repository:

```
docker tag resumeaibackend:latest public.ecr.aws/f8g8h5d4/resumeaibackend:latest
```

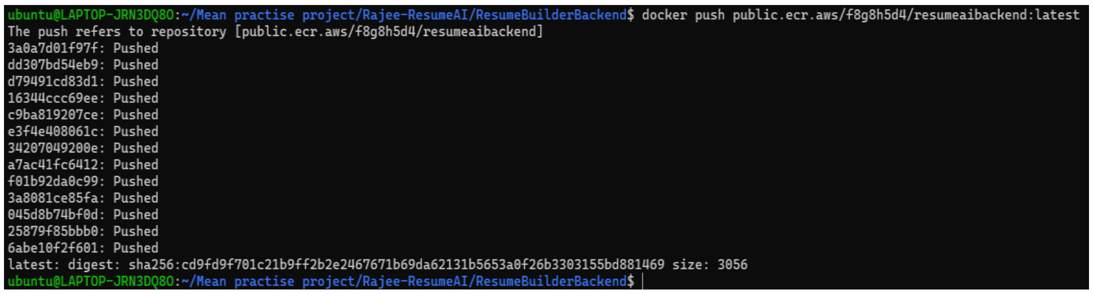
4. Run the following command to push this image to your newly created AWS repository:

```
docker push public.ecr.aws/f8g8h5d4/resumeaibackend:latest
```

Close

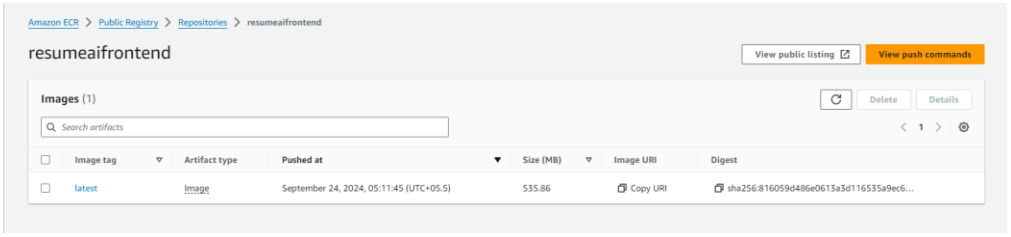
Same you need to push the backend docker image by following the above steps

In the ecr repo for backend: -



```
ubuntu@LAPTOP-JRN3DQ80:~/Mean practise project/Rajee-ResumeAI/ResumeBuilderBackend$ docker push public.ecr.aws/f8g8h5d4/resumeaibackend:latest
The push refers to repository [public.ecr.aws/f8g8h5d4/resumeaibackend]
3a0a7d01f97f: Pushed
dd307bd54eb9: Pushed
d79491cd83d1: Pushed
16344ccc69ee: Pushed
c9ba819207ce: Pushed
e3f4e408061c: Pushed
34207049200e: Pushed
a7ac41fc6412: Pushed
f01b92da0c99: Pushed
3a8081ce85fa: Pushed
045d8b74bf0d: Pushed
25879f85bbb0: Pushed
6abe10f2f601: Pushed
latest: digest: sha256:cd9fd9f701c21b9ff2b2e2467671b69da62131b5653a0f26b3303155bd881469 size: 3056
ubuntu@LAPTOP-JRN3DQ80:~/Mean practise project/Rajee-ResumeAI/ResumeBuilderBackend$
```

To ensure the images you pushed are there in the respective ecr repos or not you can check by entering the repo: -



Amazon ECR > Public Registry > Repositories > resumeaifrontend

resumeaifrontend

Images (1)

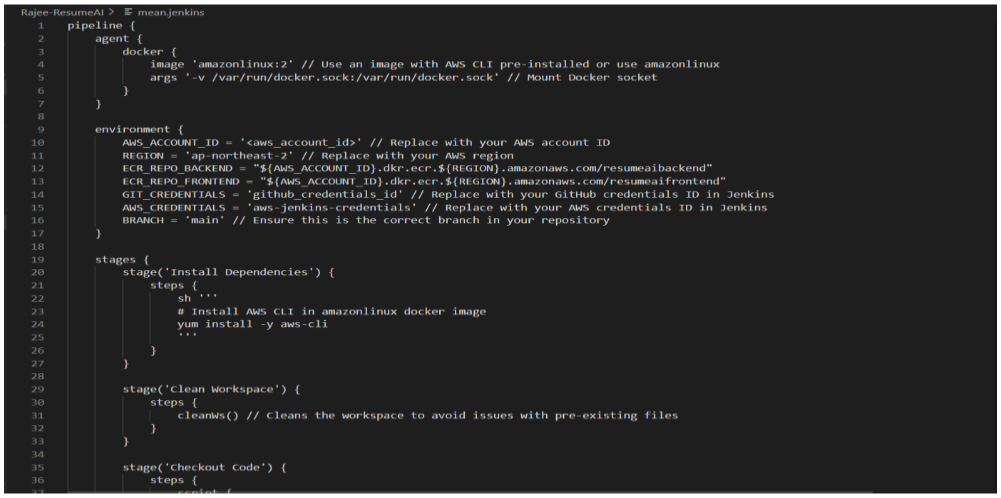
	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest
☐	latest	Image	September 24, 2024, 05:11:45 (UTC+05.5)	535.86	Copy URI	sha256:816059d486e0613a3d116535a9ec6...

5. **Create a CI/CD Pipeline using Jenkins: -**

5.1) create the jenkins job a freesytle pipeline (job):-

For automating the cloning of the project code from git-hub repo.

Building the images for the frontend and backend and containerizing the docker images and pushing them in the respected ecr repo on aws.



Rajee-ResumeAI > mean.jenkins

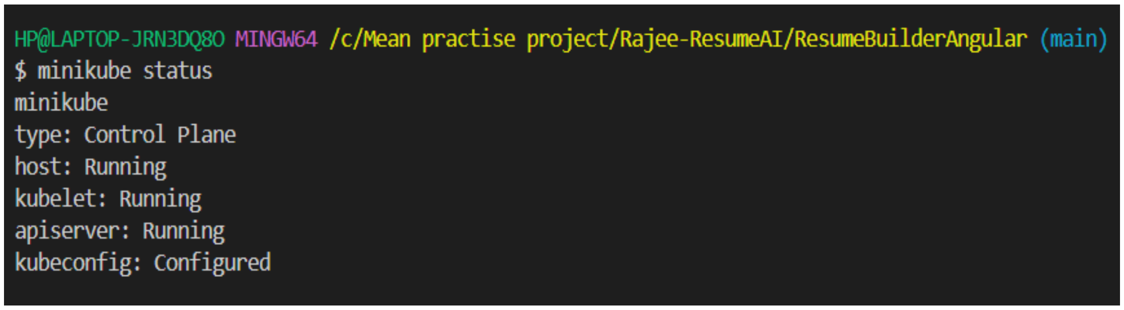
```
pipeline {
    agent {
        docker {
            image 'amazonlinux:2' // Use an image with AWS CLI pre-installed or use amazonlinux
            args '-v /var/run/docker.sock:/var/run/docker.sock' // Mount Docker socket
        }
    }

    environment {
        AWS_ACCOUNT_ID = '<aws_account_id>' // Replace with your AWS account ID
        REGION = 'ap-northeast-2' // Replace with your AWS region
        ECR_REPO_BACKEND = "${AWS_ACCOUNT_ID}.dkr.ecr.${REGION}.amazonaws.com/resumeaibackend"
        ECR_REPO_FRONTEND = "${AWS_ACCOUNT_ID}.dkr.ecr.${REGION}.amazonaws.com/resumeaifrontend"
        GIT_CREDENTIALS = 'github_credentials_id' // Replace with your GitHub credentials ID in Jenkins
        AWS_CREDENTIALS = 'aws-jenkins-credentials' // Replace with your AWS credentials ID in Jenkins
        BRANCH = 'main' // Ensure this is the correct branch in your repository
    }

    stages {
        stage('Install Dependencies') {
            steps {
                sh '''
                # Install AWS CLI in amazonlinux docker image
                yum install -y aws-cli
                '''
            }
        }

        stage('Clean Workspace') {
            steps {
                cleanWs() // Cleans the workspace to avoid issues with pre-existing files
            }
        }

        stage('Checkout Code') {
            steps {
```

6. **Deploy the Application using Minikube (Localhost)**

Installed Minikube:

Installed the minikube by using the steps mentioned in the document: -

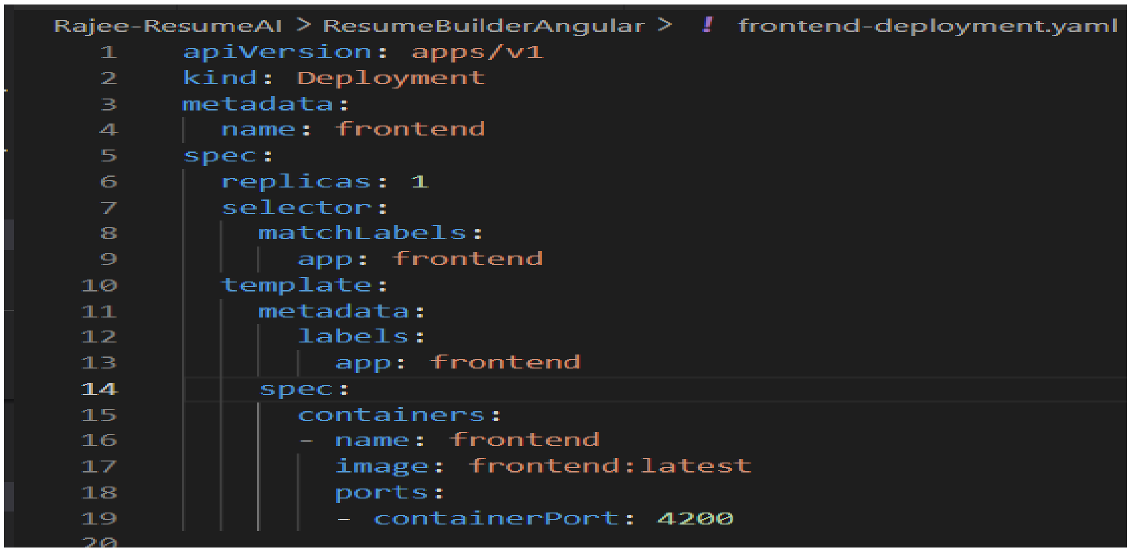
https://minikube.sigs.k8s.io/docs/start/?arch=%2Fwindows%2Fx86-64%2Fstable%2F.exe+download .



```
HP@LAPTOP-JRN3DQ8O MINGW64 /c/Mean practise project/Rajee-ResumeAI/ResumeBuilderAngular (main)
$ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured
```

6.1 Created Kubernetes Deployment Files:

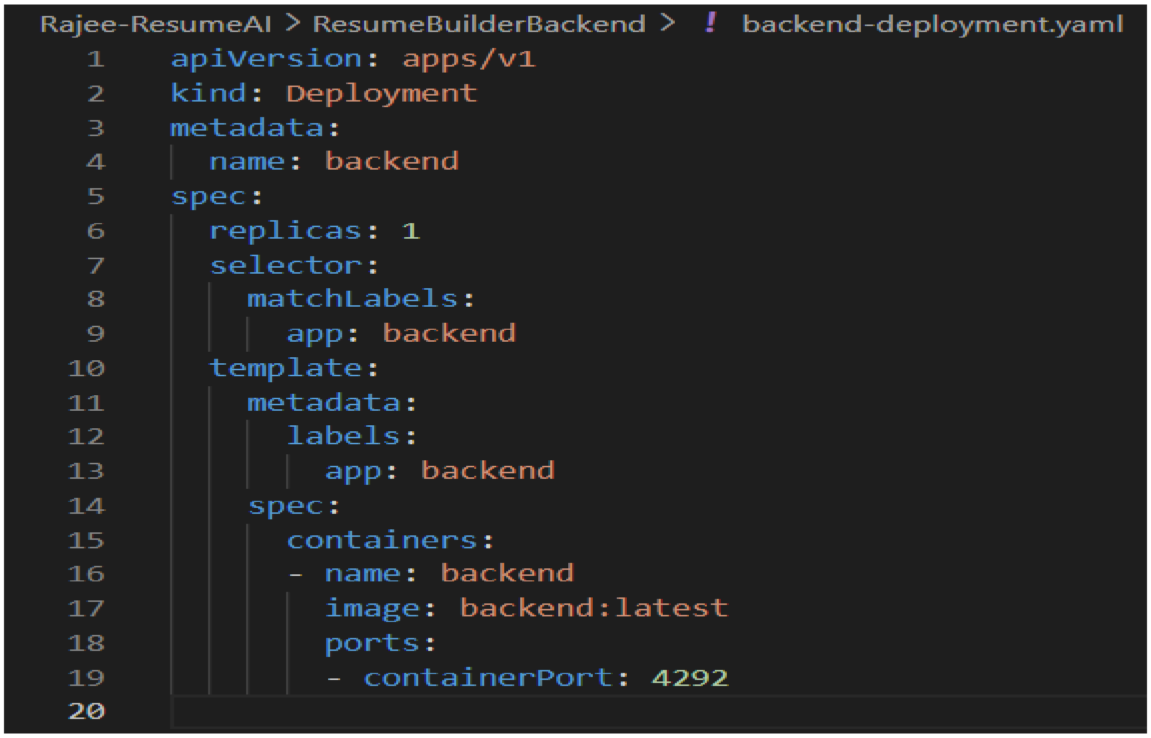
Deployment .yaml file for frontend: -



Rajee-ResumeAI > ResumeBuilderAngular > ! frontend-deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
spec:
  replicas: 1
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
      - name: frontend
        image: frontend:latest
        ports:
        - containerPort: 4200
```

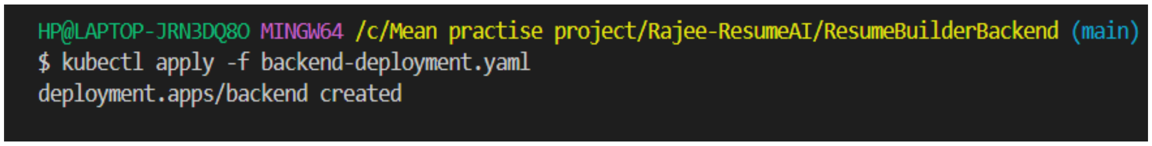
Deployment .yaml file for backend: -



```
Rajee-ResumeAI > ResumeBuilderBackend > ! backend-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend
spec:
  replicas: 1
  selector:
    matchLabels:
      app: backend
  template:
    metadata:
      labels:
        app: backend
    spec:
      containers:
      - name: backend
        image: backend:latest
        ports:
        - containerPort: 4292
```

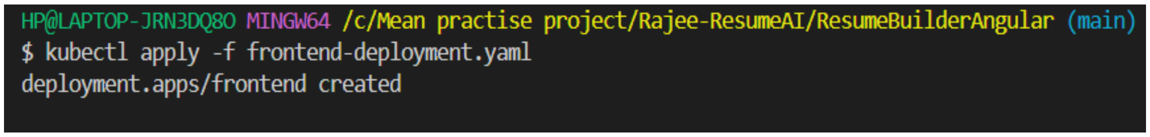
Deploy the application: -

kubectl apply -f backend-deployment.yaml (deploying the backend on localhost)



```
HP@LAPTOP-JRN3DQ8O MINGW64 /c/Mean practise project/Rajee-ResumeAI/ResumeBuilderBackend (main)
$ kubectl apply -f backend-deployment.yaml
deployment.apps/backend created
```

kubectl apply -f frontend-deployment.yaml (deploying the frontend on localhost)



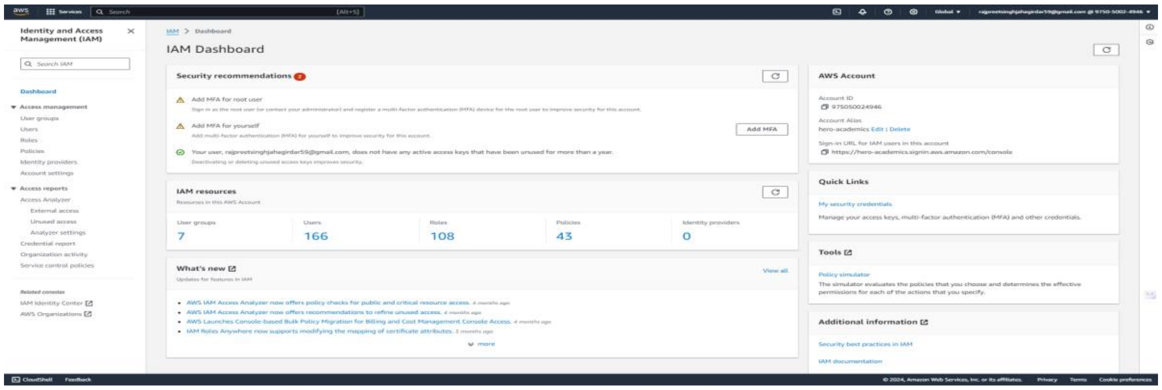
```
HP@LAPTOP-JRN3DQ8O MINGW64 /c/Mean practise project/Rajee-ResumeAI/ResumeBuilderAngular (main)
$ kubectl apply -f frontend-deployment.yaml
deployment.apps/frontend created
```

7. **Create a Kubernetes Cluster and Deploy the Application using EKS: -**

7.1 Create an EKS cluster using the AWS Console or CLI: -

7.1.1 Create IAM Roles for EKS: -

7.1.2 Go to the **IAM** console: -



IAM > Dashboard

IAM Dashboard

Security recommendations

Add MFA for root user

Sign in as the root user (or contact your administrator) and register a multi-factor authentication (MFA) device for the root user to improve security for this account.

Add MFA for yourself

Add multi-factor authentication (MFA) for yourself to improve security for this account.

Your user, rajpreetsinghjahagirdar59@gmail.com, does not have any active access keys that have been unused for more than a year.

Deactivating or deleting unused access keys improves security.

IAM resources

Resources in this AWS Account

User groups	Users	Roles	Policies	Identity providers
7	166	108	43	0

What's new

Updates for features in IAM

- AWS IAM Access Analyzer now offers policy checks for public and critical resource access. 4 months ago
- AWS IAM Access Analyzer now offers recommendations to refine unused access. 4 months ago
- AWS Launches Console-based Bulk Policy Migration for Billing and Cost Management Console Access. 4 months ago
- IAM Roles Anywhere now supports modifying the mapping of certificate attributes. 5 months ago

AWS Account

Account ID: 975050024946

Account Alias: hero-academics Edit | Delete

Sign-in URL for IAM users in this account: https://hero-academics.signin.aws.amazon.com/console

Quick Links

My security credentials

Manage your access keys, multi-factor authentication (MFA) and other credentials.

Tools

Policy simulator

The simulator evaluates the policies that you choose and determines the effective permissions for each of the actions that you specify.

Additional information

Security best practices in IAM

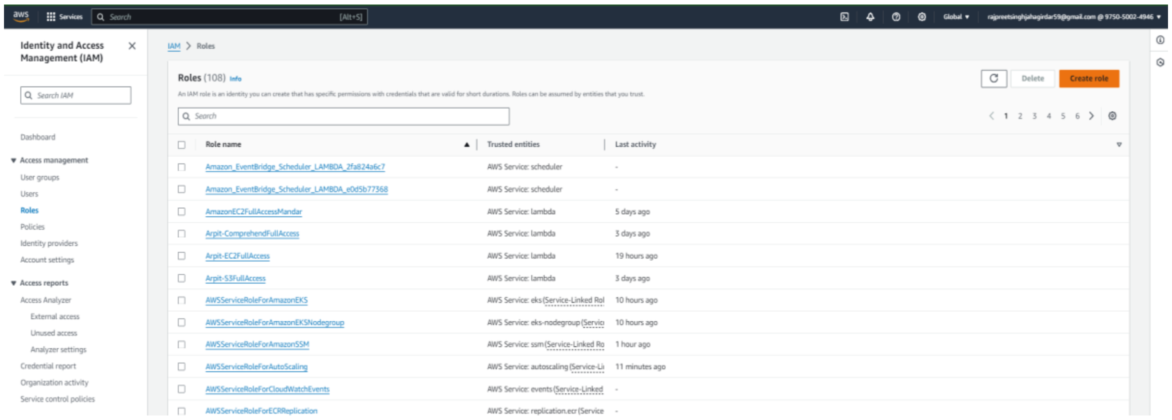
IAM documentation

© 2024, Amazon Web Services, Inc. or its affiliates.

7.1.3 Create two new IAM roles: -

EKS Cluster Role: -

7.1.4 Go to **Roles** → **Create Role**.



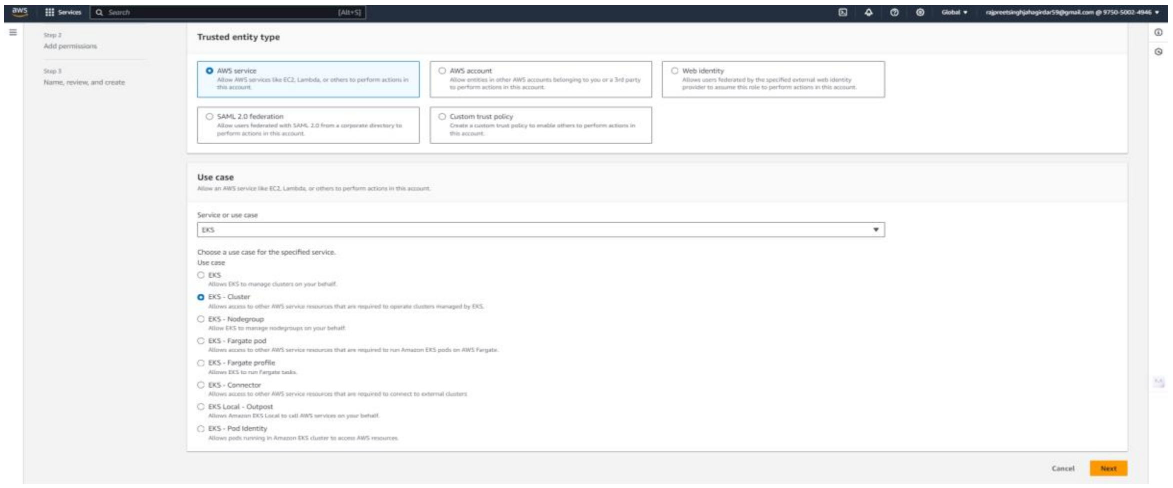
IAM > Roles

Roles (108)

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Role name	Trusted entities	Last activity
Amazon_EventBridge_Scheduler_LAMBDA_2fa824a6c7	AWS Service: scheduler	-
Amazon_EventBridge_Scheduler_LAMBDA_e0d5b77368	AWS Service: scheduler	-
AmazonEC2FullAccessMandar	AWS Service: lambda	5 days ago
Arpit-ComprehendFullAccess	AWS Service: lambda	3 days ago
Arpit-EC2FullAccess	AWS Service: lambda	19 hours ago
Arpit-S3FullAccess	AWS Service: lambda	3 days ago
AWSServiceRoleForAmazonEKS	AWS Service: eks (Service-Linked Rol	10 hours ago
AWSServiceRoleForAmazonEKSNodegroup	AWS Service: eks-nodegroup (Servic	10 hours ago
AWSServiceRoleForAmazonSSM	AWS Service: ssm (Service-Linked Ro	1 hour ago
AWSServiceRoleForAutoScaling	AWS Service: autoscaling (Service-Li	11 minutes ago
AWSServiceRoleForCloudWatchEvents	AWS Service: events (Service-Linked	-
AWSServiceRoleForECRReplication	AWS Service: replication.ecr (Service	-

7.1.5 Choose **EKS** service and select **EKS - Cluster**.



Step 2
Add permissions

Step 3
Name, review, and create

Trusted entity type

- [✓ AWS service] Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account: Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity: Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- SAML 2.0 federation: Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- Custom trust policy: Create a custom trust policy to enable others to perform actions in this account.

Use case

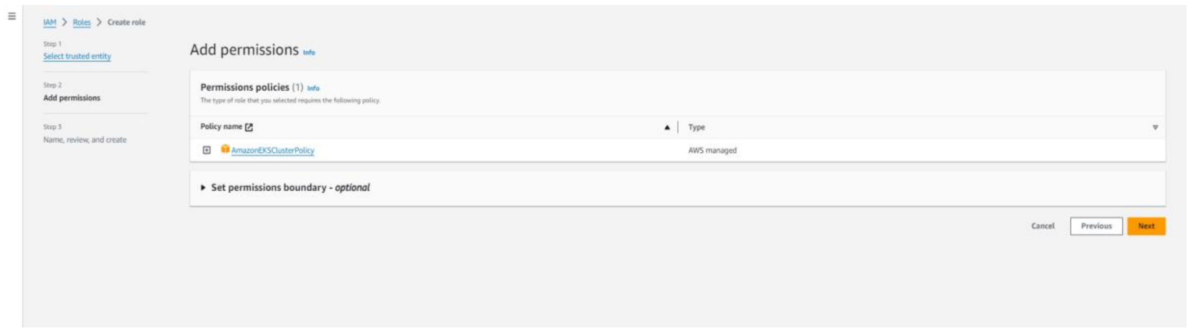
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case: EKS

Choose a use case for the specified service.
Use case

- EKS: Allows EKS to manage clusters on your behalf.
- [✓ EKS - Cluster] Allows access to other AWS service resources that are required to operate clusters managed by EKS.
- EKS - Nodegroup: Allow EKS to manage nodegroups on your behalf.
- EKS - Fargate pod: Allows access to other AWS service resources that are required to run Amazon EKS pods on AWS Fargate.
- EKS - Fargate profile: Allows EKS to run Fargate tasks.
- EKS - Connector: Allows access to other AWS service resources that are required to connect to external clusters
- EKS Local - Outpost: Allows Amazon EKS Local to call AWS services on your behalf.
- EKS - Pod Identity: Allows pods running in Amazon EKS cluster to access AWS resources.

7.1.6 Attach the policy `AmazonEKSClusterPolicy`.



Add permissions Info

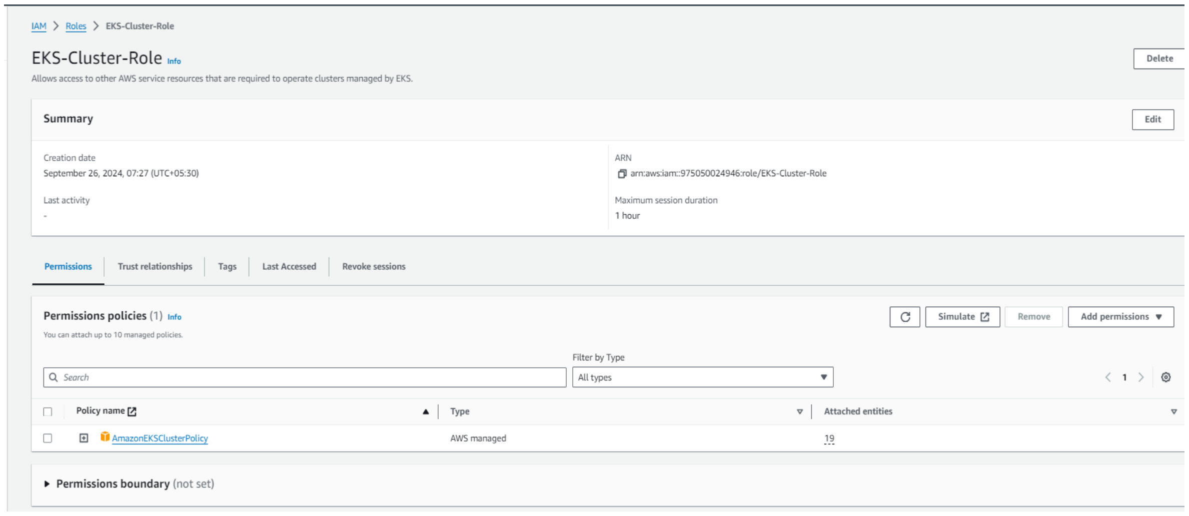
Permissions policies (1) Info

The type of role that you selected requires the following policy.

Policy name	Type
AmazonEKSClusterPolicy	AWS managed

▸ Set permissions boundary - *optional*

7.1.7 Name it something like `EKS-Cluster-Role:-`



EKS-Cluster-Role Info

Allows access to other AWS service resources that are required to operate clusters managed by EKS.

Summary

Creation date: September 26, 2024, 07:27 (UTC+05:30)	ARN: arn:aws:iam::975050024946:role/EKS-Cluster-Role
Last activity: -	Maximum session duration: 1 hour

Permissions policies (1) Info

You can attach up to 10 managed policies.

Policy name	Type	Attached entities
AmazonEKSClusterPolicy	AWS managed	19

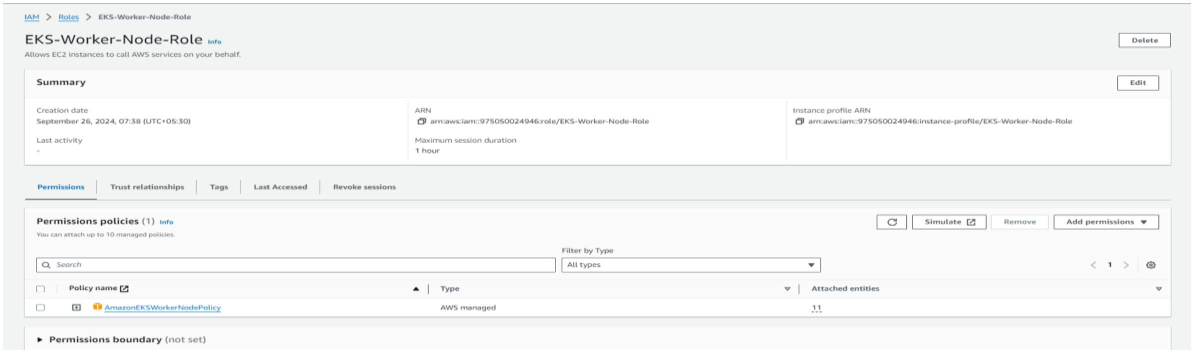
▸ Permissions boundary (not set)

7.1.8 EKS Worker Node Role: -

Go to **Roles** → **Create Role**.

Choose **EC2** service and attach the policy `AmazonEKSWorkerNodePolicy`, `AmazonEC2ContainerRegistryReadOnly`, and `AmazonEKS_CNI_Policy`.

Name it something like `EKS-Worker-Node-Role`



IAM > Roles > EKS-Worker-Node-Role

EKS-Worker-Node-Role Info

Allows EC2 instances to call AWS services on your behalf.

Summary

Creation date	ARN	Instance profile ARN
September 26, 2024, 07:38 (UTC+05:30)	arn:aws:iam::975050024946:role/EKS-Worker-Node-Role	arn:aws:iam::975050024946:instance-profile/EKS-Worker-Node-Role
Last activity	Maximum session duration	
-	1 hour	

Permissions | Trust relationships | Tags | Last Accessed | Revoke sessions

Permissions policies (1) Info

You can attach up to 10 managed policies.

Filter by Type: All types

Policy name	Type	Attached entities
AmazonEKSWorkerNodePolicy	AWS managed	11

Permissions boundary (not set)

8. Create a VPC for EKS: -

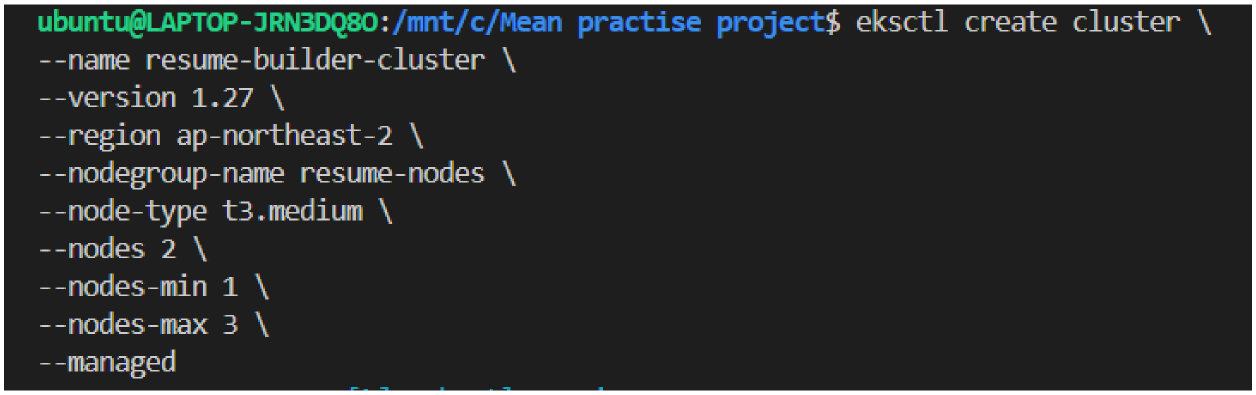
If you don't have a VPC that meets the EKS requirements, you can create a VPC using the **Amazon VPC Quickstart**. It sets up subnets, route tables, and Internet gateways needed for EKS.

You can follow the official guide for **Creating a VPC for EKS**

8.1 Create the EKS Cluster: -

Prerequisites: - you need to install aws cli and configure your local with your aws account and also you need to install eksctl on your local and minikube cluster needs to be there in the local.

Use `eksctl` to create the EKS cluster. This command will create the control plane, networking, and worker nodes automatically.



```
ubuntu@LAPTOP-JRN3DQ8O:/mnt/c/Mean practise project$ eksctl create cluster \
--name resume-builder-cluster \
--version 1.27 \
--region ap-northeast-2 \
--nodegroup-name resume-nodes \
--node-type t3.medium \
--nodes 2 \
--nodes-min 1 \
--nodes-max 3 \
--managed
```

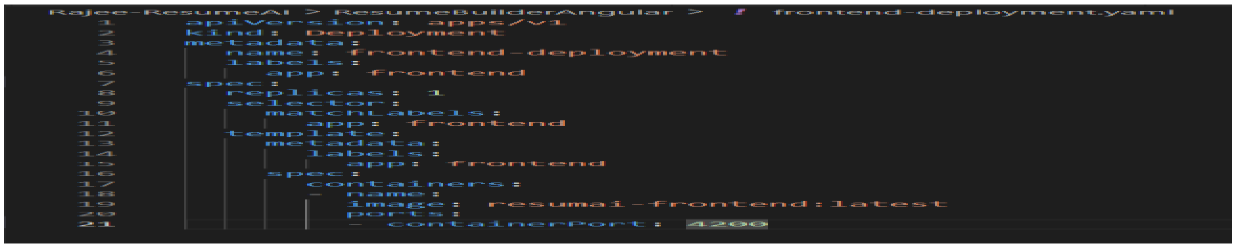
9) Deploy Application on Minikube(local host): -

To build your Docker image inside Minikube:

Hit the below command “eval $(minikube docker-env) docker build -t <your-image-name> .”

9.1. **Deploy the Application: -**

Frontend Deployment and Service: -



```
Rajee-ResumeAI > ResumeBuilderAngular > frontend-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend-deployment
  labels:
    app: frontend
spec:
  replicas: 1
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
      - name:
        image: resumai-frontend:latest
        ports:
        - containerPort: 4200
```

9.1.2 frontend-service.yaml:-

Prerequisites for deploying the project on Eks: -

1.AWS CLI installed and configured -from google or aws guide

2.kubectl installed (Installation Guide).- https://blog.knoldus.com/how-to-install-eksctl-the-official-cli-for-amazon-eks/#:~:text=download%20this%20cli.-,Downloading%20Eksctl%20on%20Ubuntu%2FDebian,eksctl%20with%20the%20following%20command.&text=Test%20that%20your%20installation%20was%20successful%20with%20the%20following%20command.&text=Now%20after%20finishing%20the%20setup,with%20this%20command%2Dline%20utility.

3.eksctl installed for managing EKS clusters-

4.Docker images pushed to Docker Hub (or another registry accessible to EKS).

10. Steps for EKS Deployment: -

10.1 Create an EKS Cluster: -

Use eksctl to create a cluster in your AWS account:-

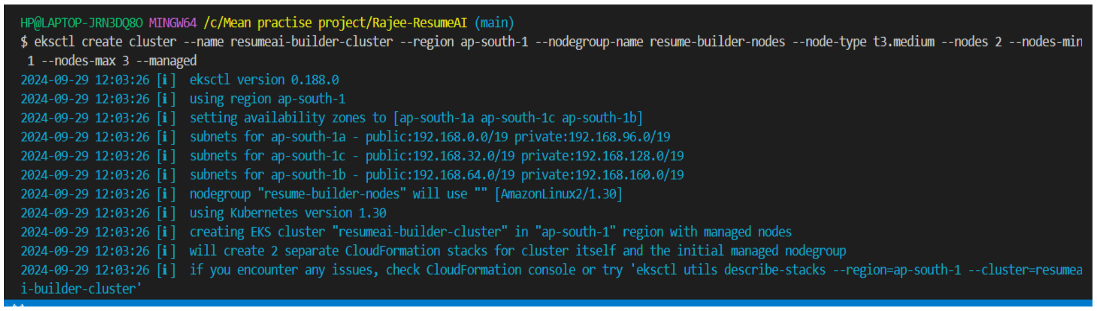
```
eksctl create cluster \
--name resume-builder-cluster \
--region <your-aws-region> \
--nodegroup-name resume-builder-nodes \
--node-type t3.medium \
--nodes 2 \
--nodes-min 1 \
--nodes-max 3 \
--managed
```

11. Configure kubectl:-

Once the cluster is created, configure kubectl to interact with your EKS cluster.

```
aws eks --region <your-aws-region> update-kubeconfig --name resume-builder-cluster
```

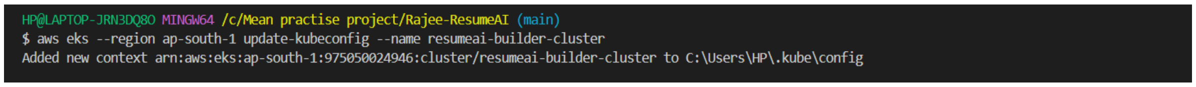
- Replace <your-aws-region> with the AWS region where you want the cluster (e.g., us-east-1).
- This command creates a managed EKS cluster with 2 nodes, with auto-scaling enabled.



```
HP@LAPTOP-JRN3DQ8O MINGW64 /c/Mean practise project/Rajee-ResumeAI (main)
$ eksctl create cluster --name resumeai-builder-cluster --region ap-south-1 --nodegroup-name resume-builder-nodes --node-type t3.medium --nodes 2 --nodes-min
 1 --nodes-max 3 --managed
2024-09-29 12:03:26 [ℹ]  eksctl version 0.188.0
2024-09-29 12:03:26 [ℹ]  using region ap-south-1
2024-09-29 12:03:26 [ℹ]  setting availability zones to [ap-south-1a ap-south-1c ap-south-1b]
2024-09-29 12:03:26 [ℹ]  subnets for ap-south-1a - public:192.168.0.0/19 private:192.168.96.0/19
2024-09-29 12:03:26 [ℹ]  subnets for ap-south-1c - public:192.168.32.0/19 private:192.168.128.0/19
2024-09-29 12:03:26 [ℹ]  subnets for ap-south-1b - public:192.168.64.0/19 private:192.168.160.0/19
2024-09-29 12:03:26 [ℹ]  nodegroup "resume-builder-nodes" will use "" [AmazonLinux2/1.30]
2024-09-29 12:03:26 [ℹ]  using Kubernetes version 1.30
2024-09-29 12:03:26 [ℹ]  creating EKS cluster "resumeai-builder-cluster" in "ap-south-1" region with managed nodes
2024-09-29 12:03:26 [ℹ]  will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
2024-09-29 12:03:26 [ℹ]  if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=ap-south-1 --cluster=resumea
i-builder-cluster'
```

12. Configure kubectl:-

Once the cluster is created, configure kubectl to interact with your EKS cluster



```
HP@LAPTOP-JRN3DQ8O MINGW64 /c/Mean practise project/Rajee-ResumeAI (main)
$ aws eks --region ap-south-1 update-kubeconfig --name resumeai-builder-cluster
Added new context arn:aws:eks:ap-south-1:975050024946:cluster/resumeai-builder-cluster to C:\Users\HP\.kube\config
```

13.Apply Kubernetes manifests for your backend,frontend and database services:-

Deployment: -

Create a deployment.yml files backend,frontend,mongo-db and also you need to create the namespaces for all the frontend,backend & mongo-db and also you need to create theservice.yml for the frontend and backend as well

And then you need to apply the deployment in the below manner: -

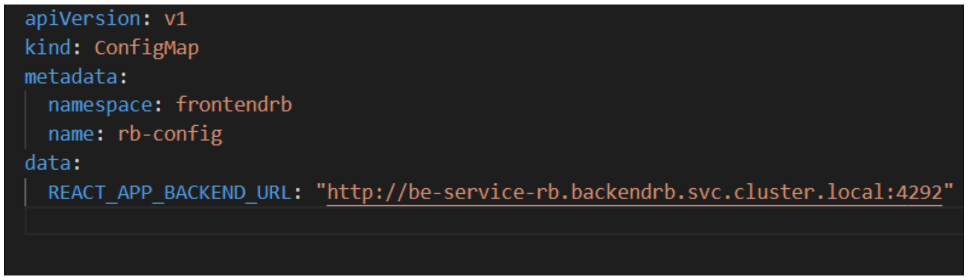
-kubectl apply -f mongo-deployment.yml

And then you need to apply your services using the below command: -

-kubectl apply -f backend-service.yaml

14. Create a ConfigMap for Environment Variables:-

Create a configmap.yaml file to store environment variables, such as the backend URL:-



```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  namespace: frontendrb
  name: rb-config
data:
  REACT_APP_BACKEND_URL: "http://be-service-rb.backendrb.svc.cluster.local:4292"
```

15. Monitor the Application:-

You can monitor the status of your deployments and services with these commands:-

kubectl get pods -n frontendrb (it will display all the running pods inside your eks cluster)

kubectl get svc -n frontendrb (it will deisplay the load balancer in the namespac)

kubectl logs <pod-name> -n frontendrb (it will show you all the logs of your kubernetes pods in the eks cluster)