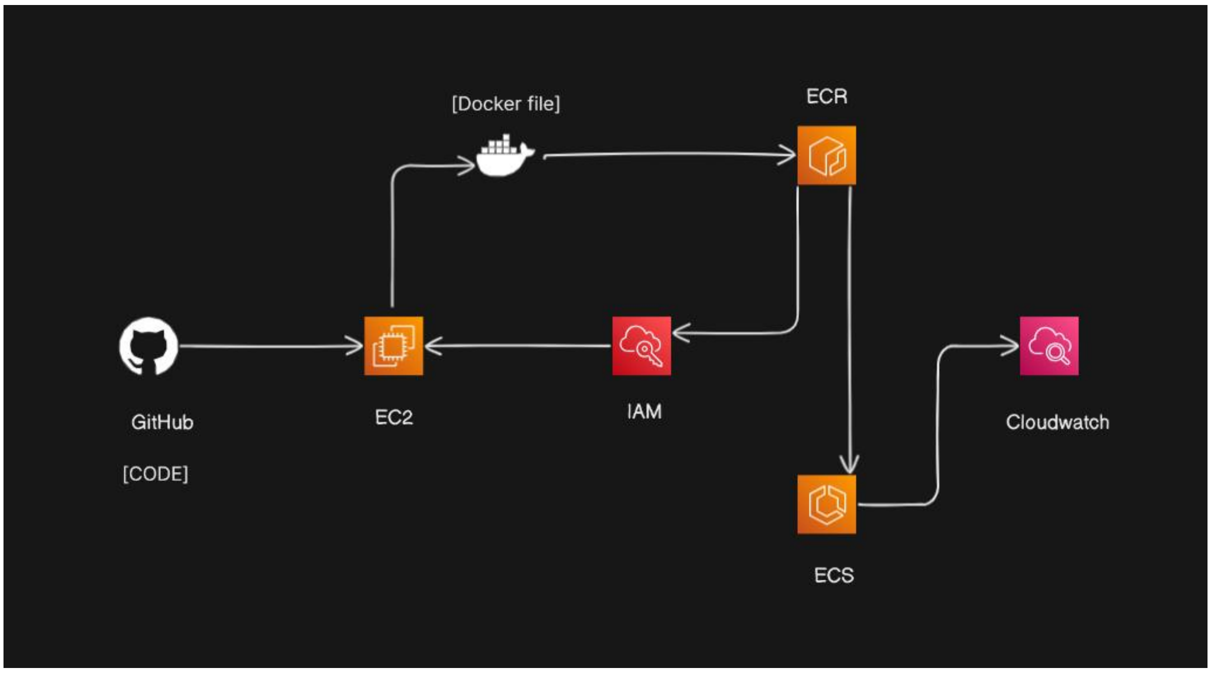
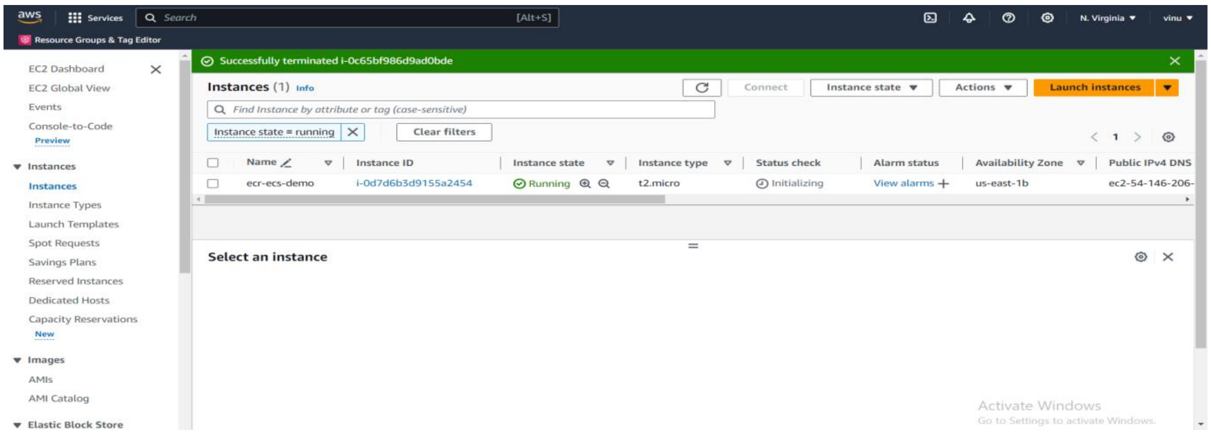
AWS PROJECT

Automated deployment of Node.js application on serverless AWS ECS Fargate with the image repository on ECR and the cloudwatch logging integrated with proper IAM roles and configuration.



Prerequisites:

- **AWS Account**: Ensure you have an AWS account with the necessary permissions to create ECS clusters, ECR repositories, and set up Cloud Watch Logs.
- Install the AWS CLI on created EC2 Instance.



Successfully terminated i-0c65bf986d9ad0bde

Instances (1) Info

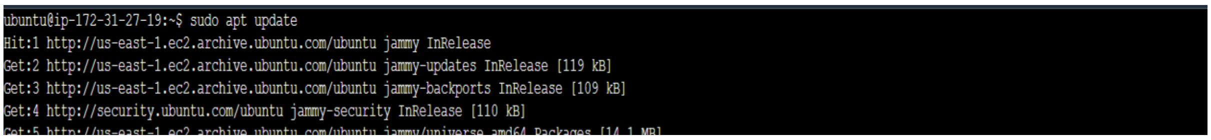
Find Instance by attribute or tag (case-sensitive)

Instance state = running

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
ecr-ecs-demo	i-0d7d6b3d9155a2454	Running	t2.micro	Initializing	View alarms	us-east-1b	ec2-54-146-206-

Select an instance

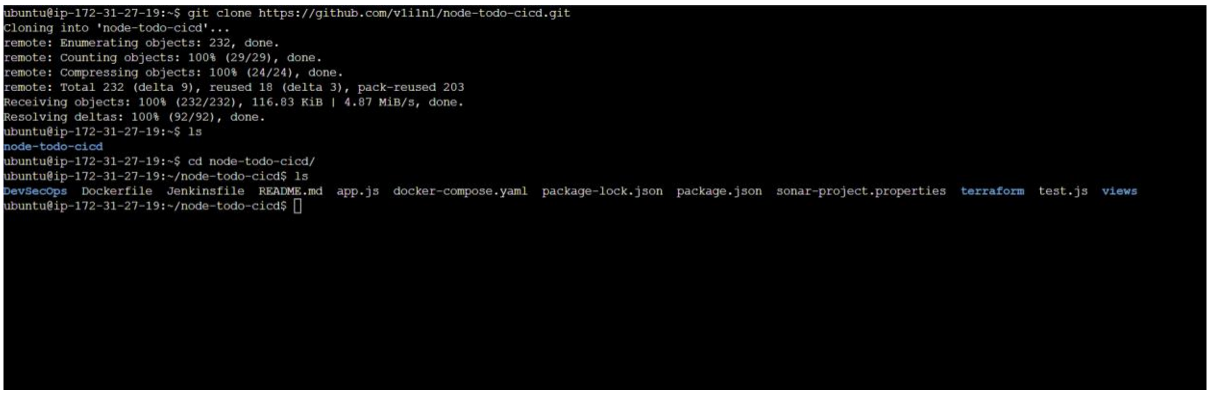
NOTE: After creating instance connect and do update it.



```
ubuntu@ip-172-31-27-19:~$ sudo apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [109 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
```

Containerize Your Node.js App:

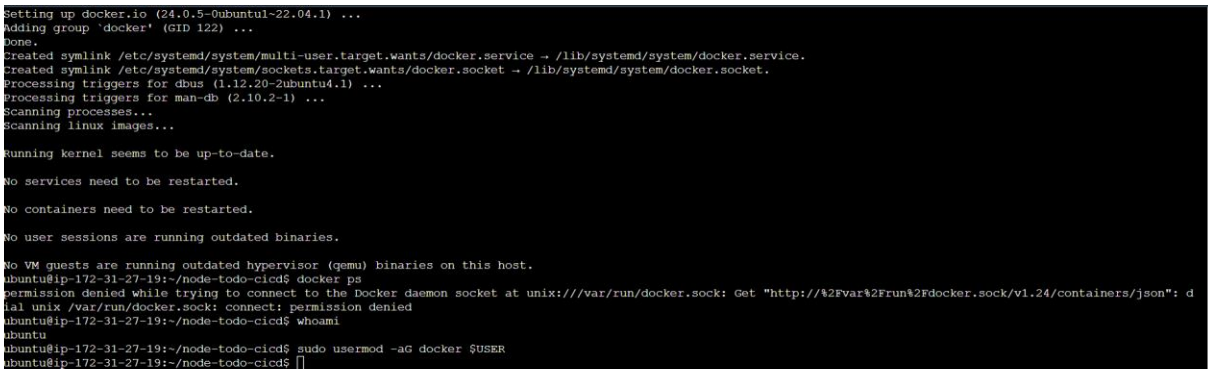
- Dockerize your Node.js application by creating a Dockerfile in the root of your project.



```
ubuntu@ip-172-31-27-19:~$ git clone https://github.com/v1i1n1/node-todo-cicd.git
Cloning into 'node-todo-cicd'...
remote: Enumerating objects: 232, done.
remote: Counting objects: 100% (29/29), done.
remote: Compressing objects: 100% (24/24), done.
remote: Total 232 (delta 9), reused 18 (delta 3), pack-reused 203
Receiving objects: 100% (232/232), 116.83 KiB | 4.87 MiB/s, done.
Resolving deltas: 100% (92/92), done.
ubuntu@ip-172-31-27-19:~$ ls
node-todo-cicd
ubuntu@ip-172-31-27-19:~$ cd node-todo-cicd/
ubuntu@ip-172-31-27-19:~/node-todo-cicd$ ls
DevSecOps  Dockerfile  Jenkinsfile  README.md  app.js  docker-compose.yaml  package-lock.json  package.json  sonar-project.properties  terraform  test.js  views
ubuntu@ip-172-31-27-19:~/node-todo-cicd$ 
```

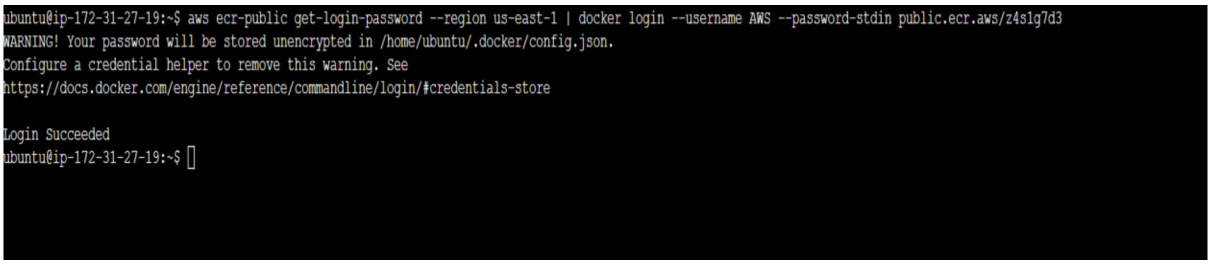
Build and Push Docker Image to ECR:

- **# Login to AWS ECR**
  $(aws ecr get-login --no-include-email --region your-region)
- **# Create ECR repository**
  aws ecr create-repository --repository-name your-repo-name --region your-region
- **# Build Docker image**
  docker build -t your-repo-name .
- **# Tag Docker image**
  docker tag your-repo-name:latest your-ecr-repo-url/your-repo-name:latest
- **# Push Docker image to ECR**
  docker push your-ecr-repo-url/your-repo-name:latest

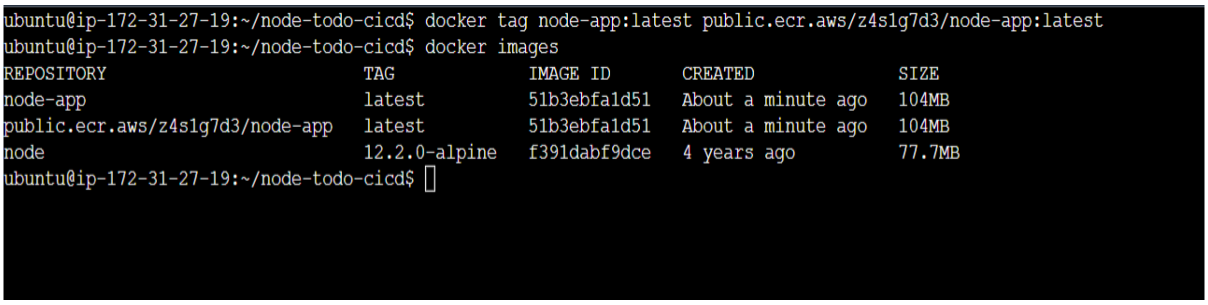


```
Setting up docker.io (24.0.5-0ubuntu1~22.04.1) ...
Adding group `docker' (GID 122) ...
Done.
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /lib/systemd/system/docker.socket.
Processing triggers for dbus (1.12.20-2ubuntu4.1) ...
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-27-19:~/node-todo-cicd$ docker ps
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/containers/json": d
ial unix /var/run/docker.sock: connect: permission denied
ubuntu@ip-172-31-27-19:~/node-todo-cicd$ whoami
ubuntu
ubuntu@ip-172-31-27-19:~/node-todo-cicd$ sudo usermod -aG docker $USER
ubuntu@ip-172-31-27-19:~/node-todo-cicd$ 
```

NOTE: Please install docker in created instance and add the current user to docker list.



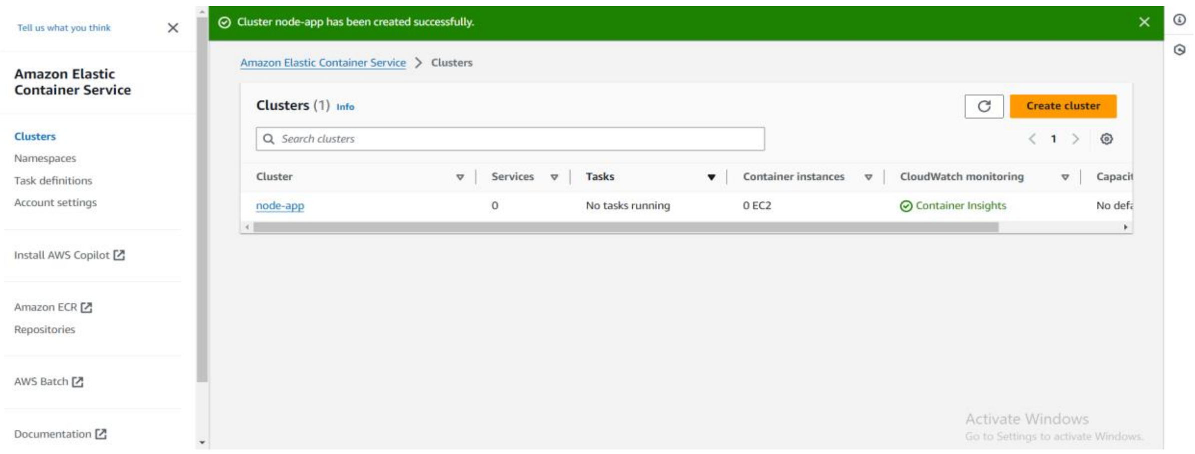
```
ubuntu@ip-172-31-27-19:~$ aws ecr-public get-login-password --region us-east-1 | docker login --username AWS --password-stdin public.ecr.aws/z4s1g7d3
WARNING! Your password will be stored unencrypted in /home/ubuntu/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
ubuntu@ip-172-31-27-19:~$
```



```
ubuntu@ip-172-31-27-19:~/node-todo-cicd$ docker tag node-app:latest public.ecr.aws/z4s1g7d3/node-app:latest
ubuntu@ip-172-31-27-19:~/node-todo-cicd$ docker images
REPOSITORY                         TAG            IMAGE ID       CREATED              SIZE
node-app                           latest         51b3ebfa1d51   About a minute ago   104MB
public.ecr.aws/z4s1g7d3/node-app   latest         51b3ebfa1d51   About a minute ago   104MB
node                               12.2.0-alpine  f391dabf9dce   4 years ago          77.7MB
ubuntu@ip-172-31-27-19:~/node-todo-cicd$
```

Create ECS Cluster:

- # **Create ECS cluster**
  aws ecs create-cluster --cluster-name your-cluster-name --region your-region



Cluster node-app has been created successfully.

Amazon Elastic Container Service > Clusters

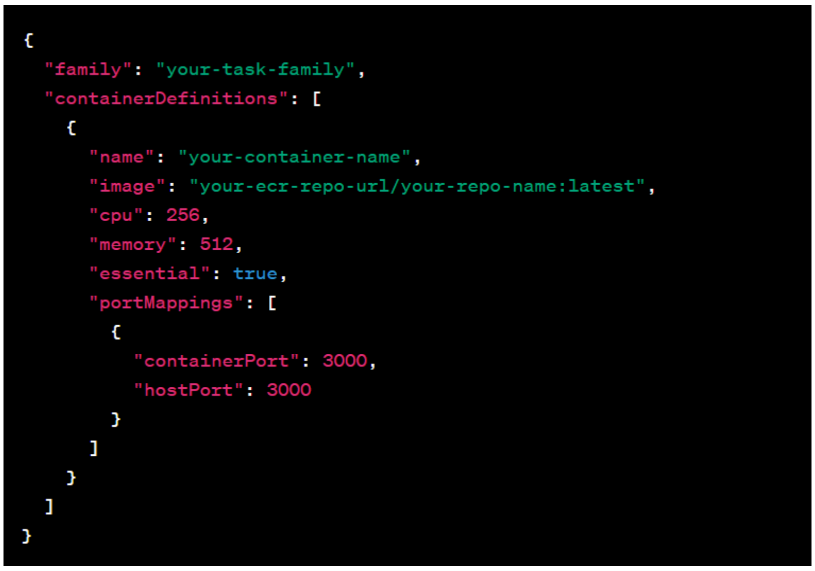
Clusters (1)

Cluster	Services	Tasks	Container instances	CloudWatch monitoring	Capaci
node-app	0	No tasks running	0 EC2	Container Insights	No defa

Create ECS Task Definition:

- Create a file named ecs-task-definition.json with your configuration. Replace placeholders with your values OR we can choose another option in AWS to configure in GUI.

Example for json format.



```
{
  "family": "your-task-family",
  "containerDefinitions": [
    {
      "name": "your-container-name",
      "image": "your-ecr-repo-url/your-repo-name:latest",
      "cpu": 256,
      "memory": 512,
      "essential": true,
      "portMappings": [
        {
          "containerPort": 3000,
          "hostPort": 3000
        }
      ]
    }
  ]
}
```

- **# Register ECS Task Definition**
  aws ecs register-task-definition --cli-input-json file://ecs-task-definition.json --region your-region

Create ECS Service:

- **# Create ECS Service**
  aws ecs create-service --cluster your-cluster-name --service-name your-service-name --task-definition your-task-family --desired-count 1 --region your-region

Set Up CloudWatch Logs:

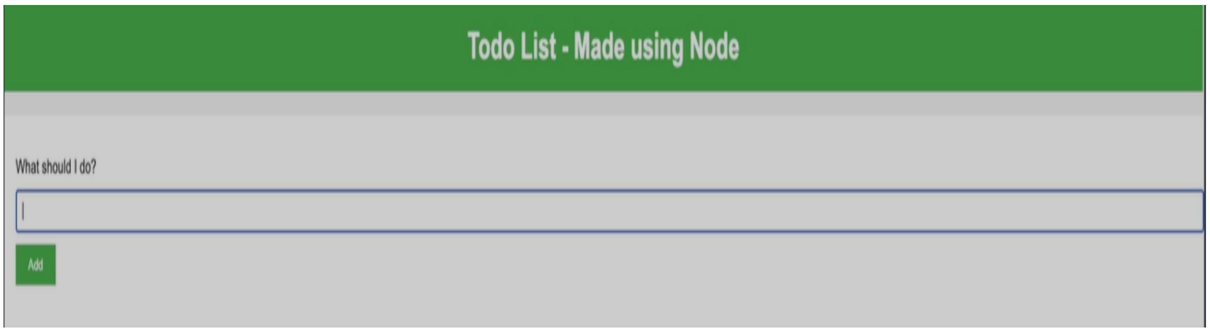
- Go to the AWS Management Console.
- Navigate to CloudWatch.
- Create a new log group and set up log streams in ECS service

Update ECS Service to Enable CloudWatch Logs:

- Update your ECS service to enable CloudWatch logs by adding the logConfiguration section to your task definition.

Verify Deployment:

- Check the ECS console to ensure your service is running. Access your Node.js app using the public IP and port defined in your task definition.


Todo List - Made using Node
What should I do?
Add