Deployment Steps

Monday, August 25, 2025 12:19 AM

Step 0: Pre-requisites

- AWS account
- AWS IAM user (e.g. github-cicd)
- GitHub repo with Docker + FastAPI app
- .github/workflows/ci.yaml + task_definition.json ready

Step 1: IAM User Setup (GitHub ke liye)

Goal: GitHub ko allow karna to build/push Docker image, deploy ECS service

1. Create an IAM **User** → github-cicd
2. Attach these **5 AWS managed policies**:
   1. AmazonEC2ContainerRegistryFullAccess
   2. AmazonECS_FullAccess
   3. AmazonS3FullAccess ← optional
   4. CloudWatchLogsFullAccess
   5. SecretsManagerReadWrite
3. Get credentials:
   - AWS_ACCESS_KEY_ID
   - AWS_SECRET_ACCESS_KEY
4. **Save them into GitHub secrets**:
   AWS_ACCESS_KEY_ID
   AWS_SECRET_ACCESS_KEY

Step 2: IAM Role for ECS Tasks (ecsTaskExecutionRole)

Goal: Allow ECS task to pull from ECR, read Secrets, and send logs

1. Go to IAM → Roles → **Create Role**
2. Select:
   - **Trusted Entity**: AWS service
   - **Use Case**: Elastic Container Service → **Elastic Container Service Task**
3. Attach policies:
   - AmazonECSTaskExecutionRolePolicy
   - CloudWatchLogsFullAccess (optional but recommended)
   - Custom inline policy for **SecretsManager access**:

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
        {
          "Sid": "AllowSecretsManagerRead",
          "Effect": "Allow",
          "Action": [
            "secretsmanager:GetSecretValue"
          ],
          "Resource": "your_secrate_manager_arn_details"
        }
      ]
    }
```

4. Name it: ecsTaskExecutionRole

Step 3: AWS Secrets Manager

Goal: Store API keys (Groq/Google) securely for ECS

1. Go to Secrets Manager → Create Secret
2. Choose **Other type of secret**
3. Add keys:

```
{
  "GROQ_API_KEY": "sk-xxx",
  "GOOGLE_API_KEY": "AIza-xxx"
}
```

4. Name: API_KEYS-XOVLi3
5. Copy the **full ARN** and use it in your ECS task definition under:

```
"secrets": [
  {
    "name": "API_KEYS",
    "valueFrom": "arn:aws:secretsmanager:ap-south-1:059948105653:secret:API_KEYS-XOVLi3"
  }
]
```

Step 4: Amazon ECR Setup

Goal: Host Docker image

1. Go to **ECR → Create Repository**
   - Name: documentportal
2. Update your GitHub Actions Docker build step to push to:

123456789012.dkr.ecr.ap-south-1.amazonaws.com/documentportal

Step 5: ECS Setup (Fargate)

Goal: Define service to run app container

1. Go to ECS → Create **Cluster**
   - Type: **Networking only (Fargate)**
2. Define:
   - Task Definition (use your task_definition.json)
   - Service
   - Execution Role: ecsTaskExecutionRole

Step 6: GitHub Actions Workflow

Goal: Automate build + deploy using GitHub Actions

1. .github/workflows/deploy.yaml should have:
   - Docker login
   - Build and push image to ECR
   - render-task-definition
   - deploy-to-ecs
2. Confirm task_definition.json includes:
   - Correct containerDefinitions[].name
   - "image": <your ECR image uri>
   - "executionRoleArn": "arn:aws:iam::059948105653:role/ecsTaskExecutionRole"
3. Run workflow → check status

Step 7: Open App in Browser

Goal: Access deployed FastAPI app

1. Go to ECS → Cluster → Service → Task → Public IP
2. Visit: http://<your_public_ip>:8080
3. If no public IP:
   - Go to ENI (Elastic Network Interface)
   - Find IPv4 public address

Step 8: View Logs

Goal: Debug or monitor app

1. Go to **CloudWatch Logs**
2. Look for log group /ecs/<task-name>
3. View logs from the latest task

Final Checklist Recap

Area	Done?
IAM user + policies added for GitHub	☑
ECS task role with secrets access	☑
ECR repo created and Docker image pushed	☑
Secrets Manager configured	☑
ECS cluster, service, and task setup	☑
GitHub workflow renders and deploys	☑
App accessible via public IP	☑
Logs visible in CloudWatch	☑