Requirements and its version:

• Ec2 Instance – Linux Server (Ubuntu Server 20.04 LTS)
• Docker version -25.0.2
• Aws-cli - 2.15.12

Step1: Connect to the server using the cmd or **mobaxterm** or **gitbash :**

• Ssh -i “fcra.pem” ubuntu@ip

Step2: Installation of Docker:

• apt-get install docker.io
• systemctl start docker
• systemctl enable docker
• systemctl status docker
•

Step2: Installation of Aws Cli :

• curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
• unzip awscliv2.zip
• apt install unzip -y
• unzip awscliv2.zip
• sudo ./aws/install (To check the path where it installed)
• AWS –version
• AWS configure (enter the id and password details of the iam role which uhave been created)

Step3:

Create a role in AWS with the access of Ec2 repository and admin access and create access key:

• for further requirements example below. * *
• Access key - XXXXXXXXXXXX
• Secret access key: XXXXXXXXXXXXX
• Region: ap-south-1

Step4:

Once you completed the above steps use the Commands to generate the self-signed certificate:

* openssl req -newkey rsa:4096 -x509 -sha256 -days 3650 -nodes -out protiviti.crt -keyout protiviti.key

* openssl pkcs12 -export -out protiviti.pfx -inkey protiviti.key -in protiviti.crt

Step5:

Then include in the certificate path in docker example below.

```
FROM mcr.microsoft.com/dotnet/aspnet:6.0-alpine AS base
WORKDIR /app
EXPOSE 80
EXPOSE 443

RUN apk apk update && apk upgrade &&
   \apk add --no-cache openssl \
   tzdata \
   icu-libs>=67 \
   krb5-libs>=1.18 \
   libgcc>=10.3 \
   libintl>=0.21 \
   libssl1.1>=1.1.1 \
   libstdc++>=10.3
   \zlib>=1.2.11

ENV TZ=Asia/Calcutta
ENV
ASPNETCORE_ENVIRONMENT=testing
ENV
DOTNET_RUNNING_IN_CONTAINER=tru
e
ENV DOTNET_SYSTEM_GLOBALIZATION_INVARIANT=false

FROM mcr.microsoft.com/dotnet/sdk:6.0-alpine AS build
WORKDIR /src
COPY ["FCRA.Web/FCRA.Web.csproj", "FCRA.Web/"]
COPY ["FCRA.Common/FCRA.Common.csproj", "FCRA.Common/"]
COPY ["FCRA.Repository/FCRA.Repository.csproj",
"FCRA.Repository/"]COPY ["FCRA.Models/FCRA.Models.csproj",
"FCRA.Models/"]
COPY ["FCRA.ViewModels/FCRA.ViewModels.csproj",
"FCRA.ViewModels/"]RUN dotnet restore "FCRA.Web/FCRA.Web.csproj"
COPY . .
WORKDIR "/src/FCRA.Web"
RUN dotnet build "FCRA.Web.csproj" -c Release -o

/app/buildFROM build AS publish
```

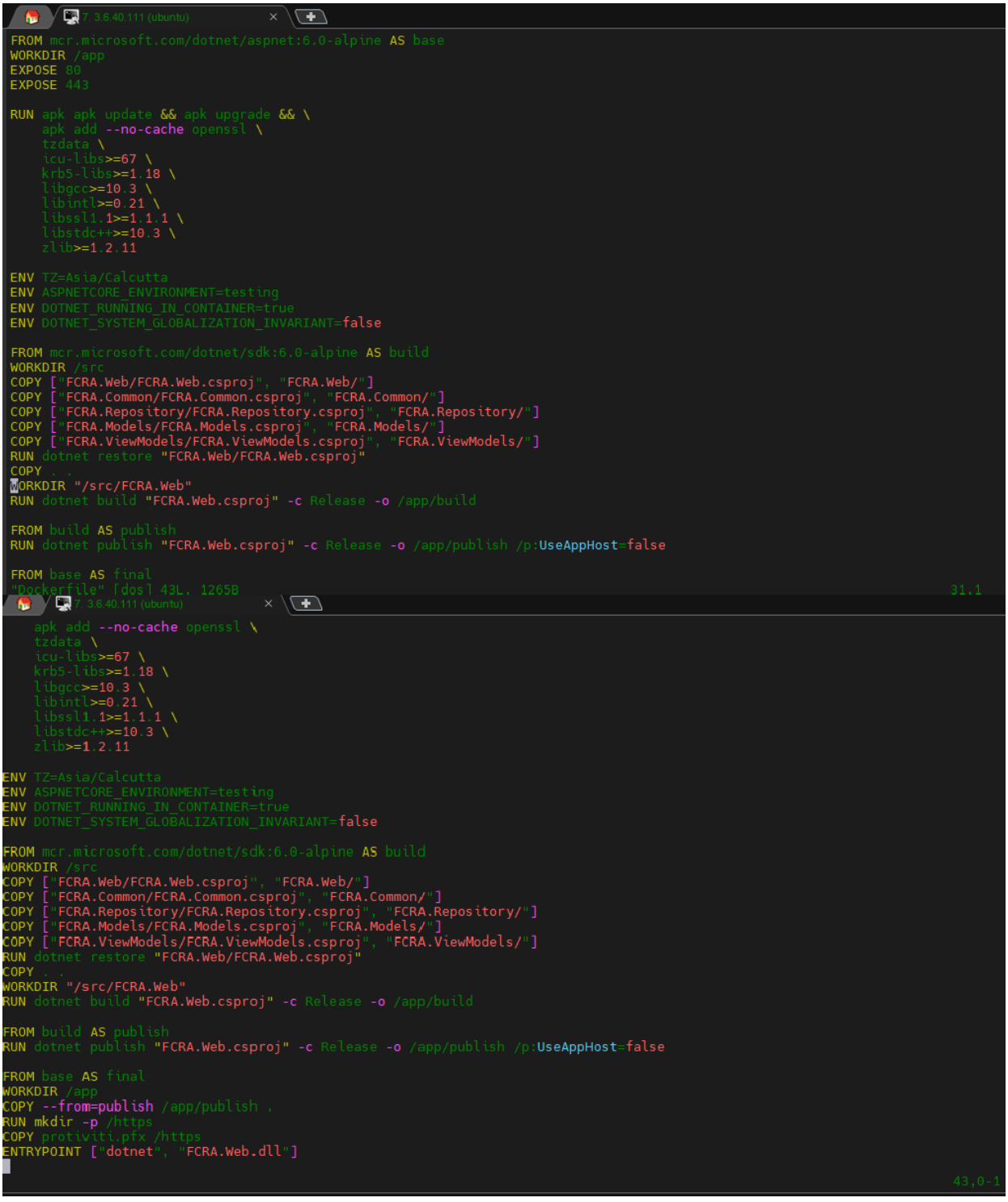
RUN dotnet publish "FCRA.Web.csproj" -c Release -o /app/publish /p:UseAppHost=false
//ARG CERT_PASSWORD
//RUN dotnet dev-certs https -ep /app/ewraprotiviti.pfx -p ewraprotiviti@123

FROM base AS
finalWORKDIR
/app
COPY --from=publish /app/publish .
//RUN mkdir -p /https
//COPY ewraprotiviti.pfx /app
COPY ewraprotiviti.pfx /https
ENTRYPOINT ["dotnet", "FCRA.Web.dll"]
* docker build -t <nameof_dockerfile> .
* docker build -t abcd .

Step6:

Then run the application with environment variables in docker using the command below.

```
docker run -d -p 80:80 -p 443:443 \
  -e ASPNETCORE_URLS="https://+;http://+" \
  -e ASPNETCORE_HTTPS_PORT=443 \
  -e ASPNETCORE_Kestrel__Certificates__Default__Password="protiviti@123" \
  -e ASPNETCORE_Kestrel__Certificates__Default__Path="/https/protiviti.pfx" \
  -e MetadataAddress="https://login.microsoftonline.com/ba04dd9d-19c9-423e-85c1-
63bc63f9ff4c/federationmetadata/2007-06/federationmetadata.xml?appid=d58bfe58-9cf7-4e27-84ec-
39fd728ab156" \
  -e username="sa" \
  -e password="fcra@123" \
  -e host="13.201.123.96" \
  -e port="" \
  -e bucketname="" \
  -e AWSRegion="ap-south-1" \
  -e dbInstanceIdentifier="RISKDBADCB" \
  -e engine="" \
  -e IsThroughSMTP="N" \
  -e From="" \
  -e SMTPUsername="" \
  -e SMTPPassword="" \
  -e SMTPHost="" \
  -e SMTPPort="587" \
  -e RealmUrl="https://ec2-3-6-40-111.ap-south-1.compute.amazonaws.com/" \
  -e certificatepath="" \
  -e certificatepassword="" \
  -e httpport="" \
  -e httpsport="" \
  -e IsSSOApplicable="Y" \
  -e IsEnvironmentVariableApplicable="Y"
  \Abcd
```



```
FROM mcr.microsoft.com/dotnet/aspnet:6.0-alpine AS base
WORKDIR /app
EXPOSE 80
EXPOSE 443

RUN apk apk update && apk upgrade && \
    apk add --no-cache openssl \
    tzdata \
    icu-libs>=67 \
    krb5-libs>=1.18 \
    libgcc>=10.3 \
    libintl>=0.21 \
    libssl1.1>=1.1.1 \
    libstdc++>=10.3 \
    zlib>=1.2.11

ENV TZ=Asia/Calcutta
ENV ASPNETCORE_ENVIRONMENT=testing
ENV DOTNET_RUNNING_IN_CONTAINER=true
ENV DOTNET_SYSTEM_GLOBALIZATION_INVARIANT=false

FROM mcr.microsoft.com/dotnet/sdk:6.0-alpine AS build
WORKDIR /src
COPY ["FCRA.Web/FCRA.Web.csproj", "FCRA.Web/"]
COPY ["FCRA.Common/FCRA.Common.csproj", "FCRA.Common/"]
COPY ["FCRA.Repository/FCRA.Repository.csproj", "FCRA.Repository/"]
COPY ["FCRA.Models/FCRA.Models.csproj", "FCRA.Models/"]
COPY ["FCRA.ViewModels/FCRA.ViewModels.csproj", "FCRA.ViewModels/"]
RUN dotnet restore "FCRA.Web/FCRA.Web.csproj"
COPY . .
WORKDIR "/src/FCRA.Web"
RUN dotnet build "FCRA.Web.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish "FCRA.Web.csproj" -c Release -o /app/publish /p:UseAppHost=false

FROM base AS final
"Dockerfile" [dos] 43L, 1265B                                                31,1
```

```
    apk add --no-cache openssl \
    tzdata \
    icu-libs>=67 \
    krb5-libs>=1.18 \
    libgcc>=10.3 \
    libintl>=0.21 \
    libssl1.1>=1.1.1 \
    libstdc++>=10.3 \
    zlib>=1.2.11

ENV TZ=Asia/Calcutta
ENV ASPNETCORE_ENVIRONMENT=testing
ENV DOTNET_RUNNING_IN_CONTAINER=true
ENV DOTNET_SYSTEM_GLOBALIZATION_INVARIANT=false

FROM mcr.microsoft.com/dotnet/sdk:6.0-alpine AS build
WORKDIR /src
COPY ["FCRA.Web/FCRA.Web.csproj", "FCRA.Web/"]
COPY ["FCRA.Common/FCRA.Common.csproj", "FCRA.Common/"]
COPY ["FCRA.Repository/FCRA.Repository.csproj", "FCRA.Repository/"]
COPY ["FCRA.Models/FCRA.Models.csproj", "FCRA.Models/"]
COPY ["FCRA.ViewModels/FCRA.ViewModels.csproj", "FCRA.ViewModels/"]
RUN dotnet restore "FCRA.Web/FCRA.Web.csproj"
COPY . .
WORKDIR "/src/FCRA.Web"
RUN dotnet build "FCRA.Web.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish "FCRA.Web.csproj" -c Release -o /app/publish /p:UseAppHost=false

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
RUN mkdir -p /https
COPY protiviti.pfx /https
ENTRYPOINT ["dotnet", "FCRA.Web.dll"]
                                                                             43,0-1
```

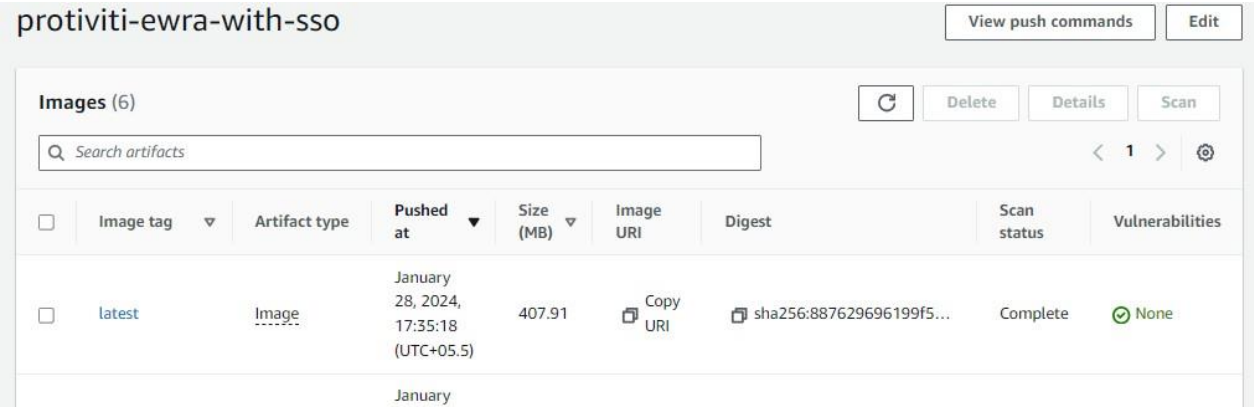
Step7:

Push the image into ECR with the commands below.

* aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 240887461522.dkr.ecr.ap-south-1.amazonaws.com

* docker tag abcd:latest 240887461522.dkr.ecr.ap-south-1.amazonaws.com/protiviti-ewra-with-sso:latest

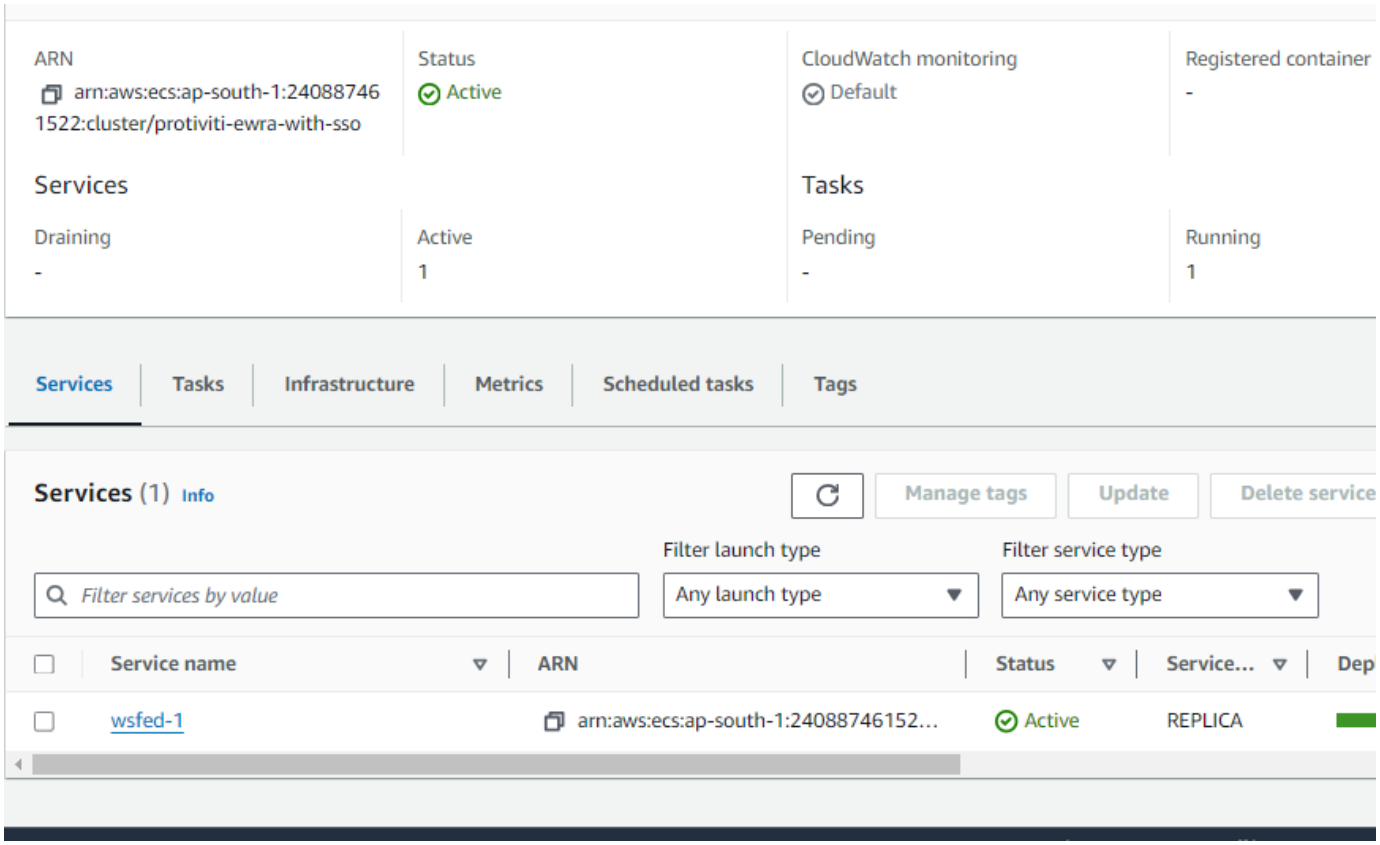
* docker push 240887461522.dkr.ecr.ap-south-1.amazonaws.com/protiviti-ewra-with-sso:latest



Step8:

Then Create a task definition in AWS ECS and attach the environment variables with the values.

- Create a task definition with the below configurations.



SMTPHost	Value	Add value	Remove
SMTPPassword	Value	Add value	Remove
SMTPPort	Value	587	Remove
SMTPUsername	Value	Add value	Remove
bucketname	Value	Add value	Remove
certificatepassword	Value	Add value	Remove
certificatepath	Value	Add value	Remove
dbInstanceIdentifier	Value	RISKDBADCB	Remove
engine	Value	Add value	Remove
host	Value	13.201.123.96	Remove
httpport	Value	Add value	Remove

Add individually

Add a key-value pair to specify an environment variable.

Key	Value type	Value	
ASPNETCORE_HTTPS_PC	Value	443	Remove
ASPNETCORE_Kestrel__C	Value	protiviti@123	Remove
ASPNETCORE_Kestrel__C	Value	/https/ewraprotiviti.pfx	Remove
ASPNETCORE_URLS	Value	https://+;http://+	Remove
AWSRegion	Value	ap-south-1	Remove
From	Value	Add value	Remove
IsEnvironmentVariableAp	Value	Y	Remove
IsSSOApplicable	Value	Y	Remove
IsThroughSMTP	Value	N	Remove
MetadataAddress	Value	https://login.microsoftor	Remove

Container details

Specify a name, container image, and whether the container should be marked as essential. Each task definition must have at least one essential container.

Name	Image URI	Essential container
ewrat	240887461522.dkr.ecr.ap-south-1.amazonaws.com/fcraadcb-uat@sha256:4d	Yes

Private registry | Info

Store credentials in Secrets Manager, and then use the credentials to reference images in private registries.

Private registry authentication

Port mappings | Info

Add port mappings to allow the container to access ports on the host to send or receive traffic. For port name, a default will be assigned if left blank.

Container port	Protocol	Port name	App protocol	
80	TCP	ewrat-80-tcp	HTTP	Remove
443	TCP	ewrat-443-tcp	App protocol	Remove

Add port mapping

Read only root file system | Info

When this parameter is turned on, the container is given read-only access to its root file system.

Read only

Resource allocation limits - *conditional* | Info

Container-level CPU, GPU, and memory limits are different from task-level values. They define how much resources are allocated for the container. If container attempts to exceed the memory specified in hard limit. the container is terminated.

Task definition configuration

Task definition family | Info

Specify a unique task definition family name.

ewrat-adcb-fcra

Up to 255 letters (uppercase and lowercase), numbers, hyphens, and underscores are allowed.

Revision

Source revision

20

Infrastructure requirements

Specify the infrastructure requirements for the task definition.

Launch type | Info

Selection of the launch type will change task definition parameters.

AWS Fargate

Serverless compute for containers.

Amazon EC2 instances

Self-managed infrastructure using Amazon EC2 instances.

OS, Architecture, Network mode

Network mode is used for tasks and is dependent on the compute type selected.

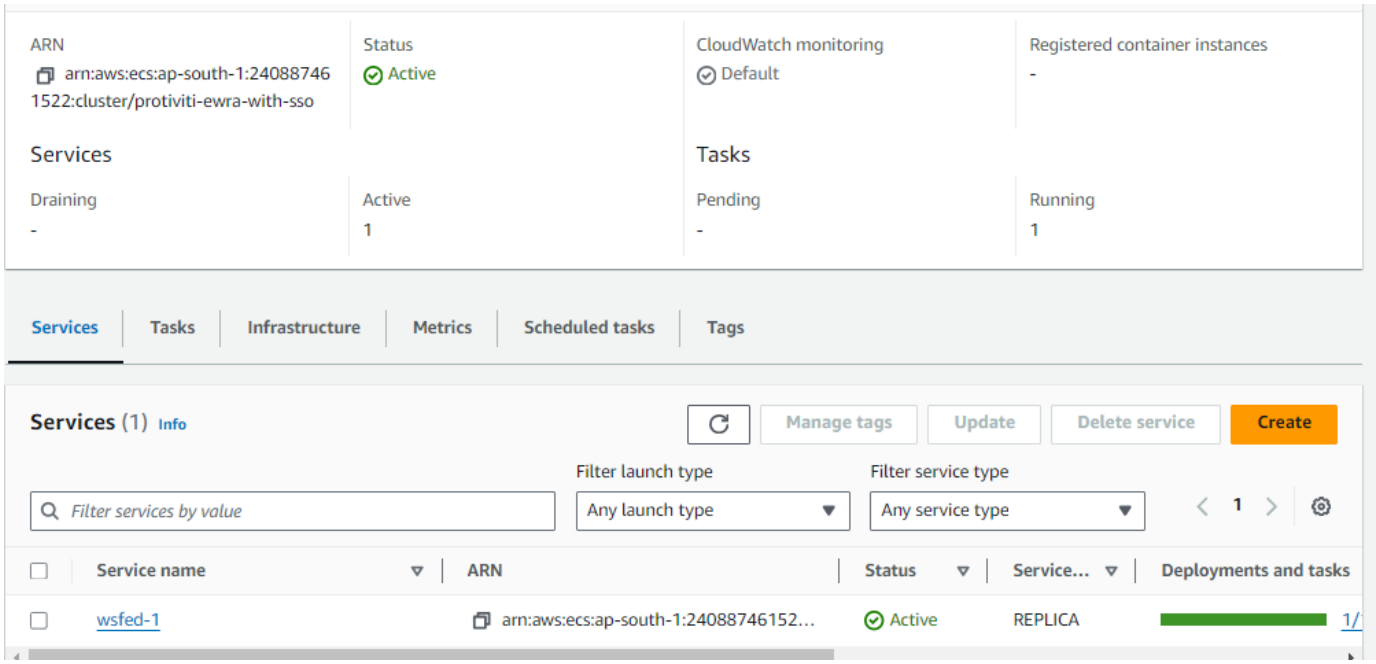
Operating system/Architecture | Info

Linux/X86_64

Network mode | Info

Step9:

Then attach the created task for the Cluster shown in the below image.



```
                    "name": "AWSRegion",
                    "value": "ap-south-1"
                },
                {
                    "name": "SMTPPort",
                    "value": "587"
                },
                {
                    "name": "password",
                    "value": "fcra@123"
                },
                {
                    "name": "SMTPPassword'',
                    "value": ""
                },
                {
                    "name": "engine",
                    "value": ""
                },
                {
                    "name": "bucketname",
                    "value": ""
                },
                {
                    "name": "httpport",
                    "value": ""
                },
                {
                    "name": "host"
```

JSON

```
{
    "taskDefinitionArn": "arn:aws:ecs:ap-south-1:240887461522:task-definition/protiviti-ewra-with-sso:12",
    "containerDefinitions": [
        {
            "name": "protiviti-ewra-with-sso",
            "image": "240887461522.dkr.ecr.ap-south-1.amazonaws.com/protiviti-ewra-with-sso:latest",
            "cpu": 0,
            "portMappings": [
                {
                    "name": "protiviti-ewra-with-sso-80-tcp",
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp",
                    "appProtocol": "http"
                },
                {
                    "name": "protiviti-ewra-with-sso-443-tcp",
                    "containerPort": 443,
                    "hostPort": 443,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "environment": [
                {
                    "name": "IsSSOApplicable",
                    "value": "Y"
```

```
        {
            "name": "ecs.capability.task-eni"
        },
        {
            "name": "com.amazonaws.ecs.capability.docker-remote-api.1.29"
        }
    ],
    "placementConstraints": [],
    "compatibilities": [
        "EC2",
        "FARGATE"
    ],
    "requiresCompatibilities": [
        "FARGATE"
    ],
    "cpu": "1024",
    "memory": "3072",
    "runtimePlatform": {
        "cpuArchitecture": "X86_64",
        "operatingSystemFamily": "LINUX"
    },
    "registeredAt": "2024-01-28T12:08:07.696Z",
    "registeredBy": "arn:aws:iam::240887461522:user/sachchidanandjha",
    "tags": []
}
```