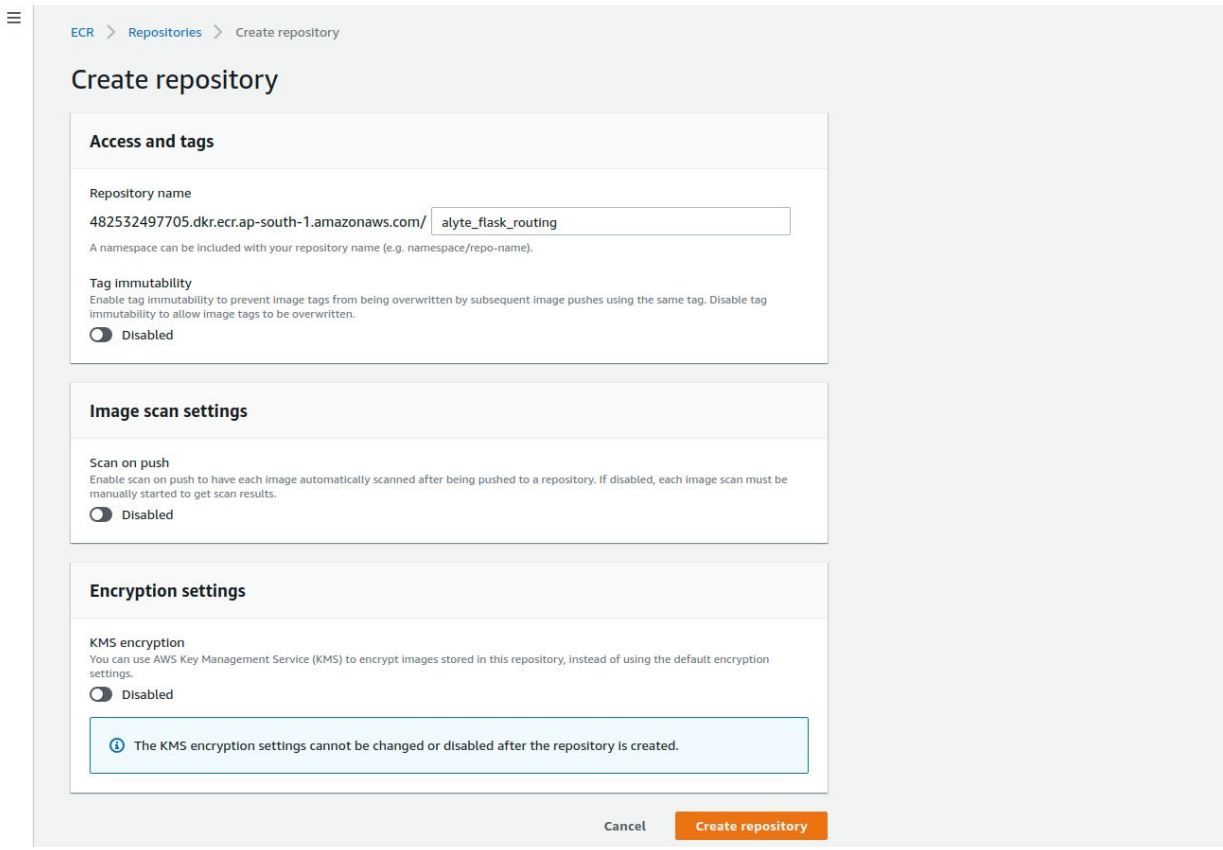
Alyte python routing service for Sandbox ENV

Note: Before creation please Create VPC | Alyte-Prod [Click here for VPC Creation If not create]

Step 1 : Create ECR repository : alyte_flask_routing

1. Login aws portal
2. Go to ECR service and create it
3. 

Step 2 : Create Task definition : alyte_flask_rrouting

Note: copy sandbox definition

Account Settings
Amazon ECR
Repositories
AWS Marketplace
Discover software
Subscriptions

Task Definition Name* alyte-flask-rounting-api

Task Role ecsTaskExecutionRole

Optional IAM role that tasks can use to make API requests to authorized AWS services. Create an Amazon Elastic Container Service Task Role in the IAM Console

Network Mode awsvpc

If you choose <default>, ECS will start your container using Docker's default networking mode, which is Bridge on Linux and NAT on Windows. <default> is the only supported mode on Windows.

Requires compatibilities
- [ ] EC2
- [x] FARGATE

Task execution IAM role

This role is required by tasks to pull container images and publish container logs to Amazon CloudWatch on your behalf. If you do not have the ecsTaskExecutionRole already, we can create one for you.

Task execution role ecsTaskExecutionRole

Task size

The task size allows you to specify a fixed size for your task. Task size is required for tasks using the Fargate launch type and is optional for the EC2 launch type. Container level memory settings are optional when task size is set. Task size is not supported for Windows containers.

Task memory (GB) 1GB

The amount of memory (in MiB) used by the task. It can be expressed as an integer using MiB, for example 1024, or as a string using GB, for example '1GB' or '1 gb'.

Task CPU (vCPU) 0.5 vCPU

The number of CPU units used by the task. It can be expressed as an integer using CPU units, for example 1024, or as a string using vCPUs, for example '1 vCPU' or '1 vcpu'.

Task memory maximum allocation for container memory reservation

0 — 24 shared of 1024 MiB

Task CPU maximum allocation for containers

0 — 512 shared of 512 CPU units

Container Definitions

Add container

Container Name	Image	Hard/Soft memory limit...	CPU Units ...	GPU	Essential ...	
flask-routing-api	482532497705.dkr.ecr.a...	–/1000	0		true	

Service Integration

AWS App Mesh is a service mesh based on the Envoy proxy that makes it easy to monitor and control microservices. App Mesh standardizes how your microservices communicate, giving you end-to-end visibility and helping to ensure high-availability for your applications. To enable App Mesh integration, complete the following fields and then choose **Apply** which will auto-configure the proxy configuration. Learn more

Enable App Mesh integration [ ]

Proxy Configuration

The configuration details for App Mesh proxy. These fields are auto-configured for you after applying the App Mesh integration options above, otherwise must be configured manually. Learn More

Enable proxy configuration [ ]

Log Router Integration

FireLens for Amazon ECS helps you route logs to an AWS service or AWS Partner Network (APN) destination for log storage and analysis. FireLens works with Fluentd and Fluent Bit. To auto-configure a log router container, complete the following fields and then choose **Apply**. Learn more

Enable FireLens integration [ ]

Volumes

Use a volume configuration to add volumes for use by the containers within a task. To add a volume, choose **Add volume**, complete the fields, and then choose **Add**. Learn more

Add volume

Configure via JSON

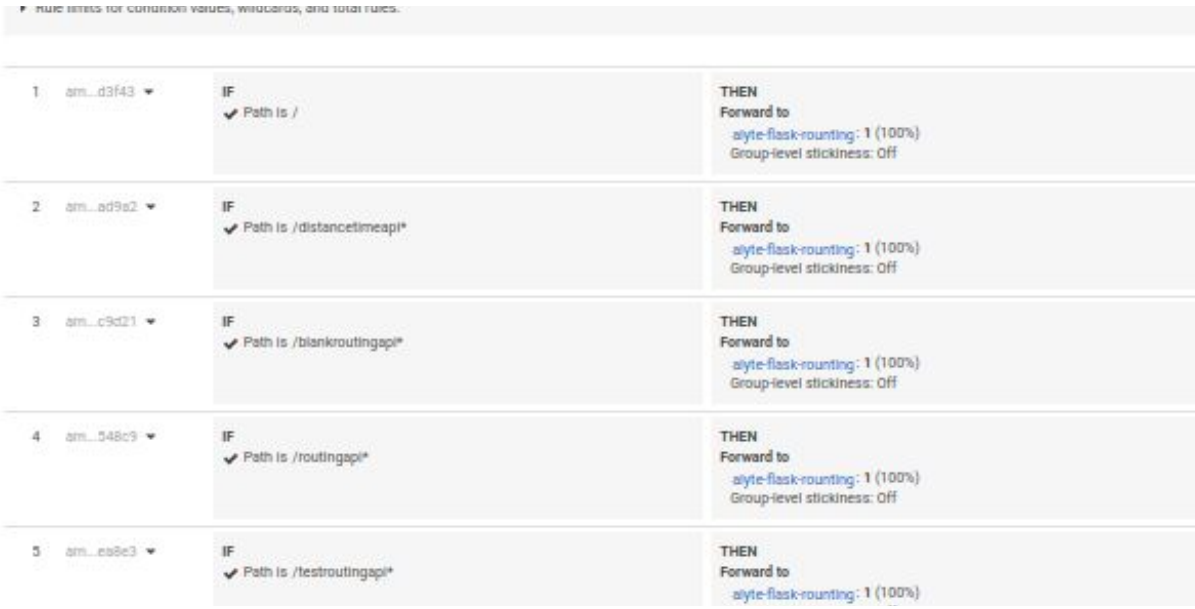
Step 3. application Load balancer : node js backend load balancer

Ec2 << load balancer << Alb << update nodejs backend balancer

Note: Load balancer under VPC | alyte_prod

Note: update nodejs load balancer target group

1. Click load balancer << listener<< view and edit rules << Edit



1	am...d3f43	IF Path is /	THEN Forward to alyte-flask-rounting: 1 (100%) Group-level stickiness: Off
2	am...ad9a2	IF Path is /distancetimeapi*	THEN Forward to alyte-flask-rounting: 1 (100%) Group-level stickiness: Off
3	am...c9d21	IF Path is /blankroutingapi*	THEN Forward to alyte-flask-rounting: 1 (100%) Group-level stickiness: Off
4	am...548c9	IF Path is /routingapi*	THEN Forward to alyte-flask-rounting: 1 (100%) Group-level stickiness: Off
5	am...ea8e3	IF Path is /testroutingapi*	THEN Forward to alyte-flask-rounting: 1 (100%)

Step 4 : Create ECS Cluster AS Fargate type : Click here for custer creation

Step 5 : Create Service under cluster AS Fargate type Service : Create ECS SERVICE

Step 6 : Create ECS service with Codepipeline CICD : Please follow step for CICD

NOTE : Refer sandbox build spec file for deployment

```
version: 0.2

phases:
  install:
    runtime-versions:
      docker: 18
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - aws --version
      - $(aws ecr get-login --region ap-south-1 --no-include-email)
      -
```

```
REPOSITORY_URI=482532497705.dkr.ecr.ap-south-1.amazonaws.com/flask_api
      - COMMIT_HASH=$(echo $CODEBUILD_RESOLVED_SOURCE_VERSION | cut -c 1-7)
      - IMAGE_TAG=${COMMIT_HASH:=latest}
  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t flask_api .
      - docker tag flask_api:latest $REPOSITORY_URI:$IMAGE_TAG
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker images...
      - docker push $REPOSITORY_URI:$IMAGE_TAG
      - echo Writing image definitions file...
      - printf '[{"name":"flask-routing-api","imageUri":"%s"}]'
$REPOSITORY_URI:$IMAGE_TAG > imagedefinitions.json
artifacts:
    files: imagedefinitions.json
```