Pre-requisites:-

Create AWS account

install docker

https://docs.docker.com/engine/installation/#installation

install awscli

sudo pip install aws-cli

AWS ECS:-

Use ecs-optimised ami for launching ec2 instance.

|  |  |  |
| --- | --- | --- |
| us-east-1 | **amzn-ami-2015.09.f-amazon-ecs-optimized** | ami-43043329 |

Launch container instance with proper IAM roles(**AmazonEC2ContainerServiceforEC2Role**).Else we cannot see the instance in the cluster (faced with this problem initially)

Reference to create IAM roles :- <http://docs.aws.amazon.com/AmazonECS/latest/developerguide/instance_IAM_role>.html

ECR:-(Ec2 Container Registry-Place where container images are stored)

Images can either be used form docker hub or can be uploaded to aws using ECR:-

to upload images to ECR:-

(make sure u have awscli installed)

For Docker basics:-

https://www.conetix.com.au/blog/docker-basics-practical-starters-guide

<https://www.digitalocean.com/community/tutorials/docker-explained-using-dockerfiles-to-automate-building-of-images>

Dockerfile is used to build the image.

docker build -t image\_name .(note the dot) (will build the image with dockerfile present in the current path and image name=image\_name)

docker images command can be used to check local images.

Create repository and then push ur image there.

(congifure aws first to connect to ur aws)

<http://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-started.html>

(u’ll need access key n secret access key for it)

It can be created from IAM role(service)->users .select the user and create access key.

Download n keep it safe.

Now create repo and follow steps to push image to ecr.

https://console.aws.amazon.com/ecs/home?region=us-east-1#/repositories/create

All steps are given in the above link.

Done with pushing image to repo in ECR? Cool Let’s move on to ECS now:-

(Ec2 Container Service-Provides complete container service )

We have a default cluster where we can see our ecs optimised instances if properly configured.

Then creating task definition by either json or aws console.

After successful creation running the task from Actions.. It shuld be in running state.

Many running tasks cannot be mapped to same port else we’ll get an error of RESOURCE:PORTS.

While writing the task defintion the main things to mention are port mappings,entrypoint or command as per given syntax(what is to be done when the container runs).

memory units and cpu units .. rest we can fill as required like the environment variables etc.

using aws cli to access ecs:-

aws ecs list-clusters

aws ecs list-task-definitions

aws ecs list-tasks --cluster default

aws ecs describe-tasks --cluster default --task task\_id(from prev command)

(gives task-definition)

aws ecs describe-container-instances --cluster default --container-instance conatiner\_id(frm prev command)

aws ecs run-task --task-definition task\_def --cluster default

References:-

<http://docs.aws.amazon.com/AmazonECS/latest/developerguide/launch_container_instance.html>

<http://docs.aws.amazon.com/AmazonECR/latest/userguide/ECR_GetStarted.html>

Running services :-

Same task definition will be used . Difference is that service will keep on running always.

Services can be run Without elb as well as with elb

To run with elb we need to create elb and ecsService role as well given as in below link.

<http://docs.aws.amazon.com/AmazonECS/latest/developerguide/service-load-balancing.html>

Service cannot be directly deleted. We need to update its task number to 0 and then delete it.

EC2 Container instances:-

<http://docs.aws.amazon.com/AmazonECS/latest/developerguide/ECS_instances.html>

Ecs service limits:

<http://docs.aws.amazon.com/AmazonECS/latest/developerguide/service_limits.html>

starting ec2 instances form cli

ec2-start-instances i-48d7b9cb -O access\_key -W secret\_key

Stopping instance

ec2-stop-instances i-48d7b9cb -O access\_key -W secret\_key

LAMBDA

<http://docs.aws.amazon.com/lambda/latest/dg/intro-core-components.html#intro-core-components-event-sources>

CloudWatch trigers lambda funcitons and it uses cron expressions to decide trigers timings.

<https://docs.aws.amazon.com/lambda/latest/dg/with-scheduled-events.html>

<https://docs.aws.amazon.com/lambda/latest/dg/tutorial-scheduled-events-schedule-expressions.html>

Creating lambda function

<https://docs.aws.amazon.com/lambda/latest/dg/getting-started-create-function.html>

Select lambda-canary blue-print for scheduled events in lamda console.

Here we use python to create lambda function

boto3 library of python is used to access aws

Since lambda needs to access ec2,ecs and dynamodb , permissions for those are given in execution role .We need to define the poilcy document as below:-

<https://docs.google.com/document/d/1uPKalZehZwksKXZOTOzIH4gpN47Tqb_dysYNNUNabxs/edit>

"iam:PassRole" is needed because we need to specify role for elb (ecsServiceRole) while creating service.

It’ll give error without this.

Here we use dynamodb to store info about instances,tasks,time and its ids like given below:-

|  |  |  |
| --- | --- | --- |
| **Column name** | **Type** | **Values** |
| time | String | 9am |
| action | String | Start/stop |
| noOfRequiredTasks | int | 1 |
| taskIDs | String Array | .. |
| instanceIDs | String Array | .. |

After giving this policy we can access ec2,ecs n dynamodb through our lamda function:-

Given here:-

<https://docs.google.com/document/d/13uHEWyZoEmT9rzll0V6BTRegw-lgiId8GRQl1vwBgkY/edit>

References for boto3 api used:-

EC2

<http://boto3.readthedocs.org/en/latest/reference/services/ec2.html#instance>

ECS

<http://boto3.readthedocs.org/en/latest/reference/services/ecs.html>

Dynamodb

<http://boto3.readthedocs.org/en/latest/guide/dynamodb.html>