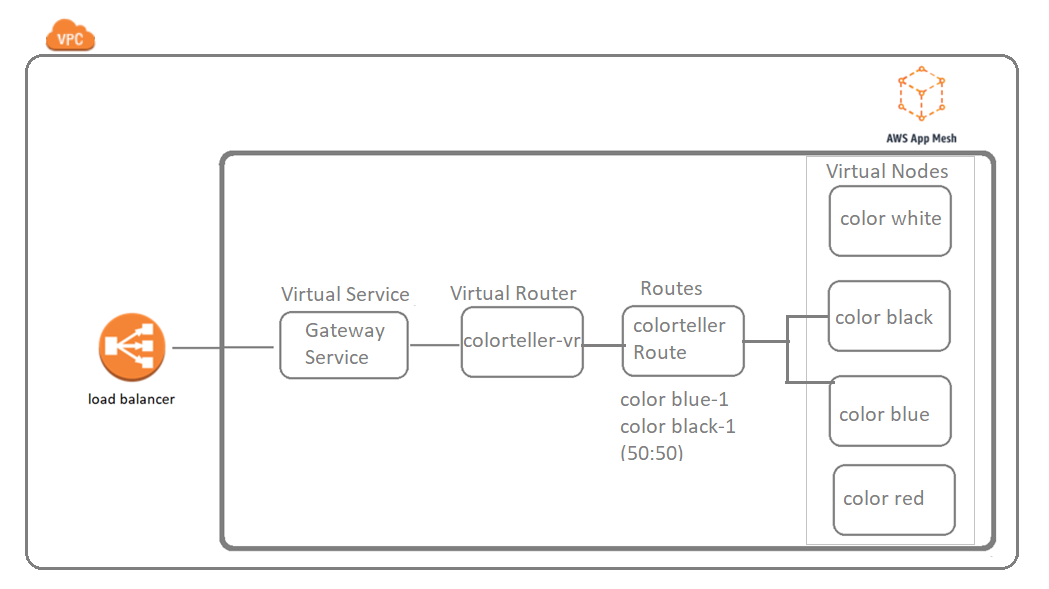
**APP-MESH on FARGATE cluster Automated**

AWS App Mesh is a service mesh that provides application-level networking to make it easy for your services to communicate with each other across multiple types of compute infrastructure. App Mesh standardizes how your services communicate, giving you end-to-end visibility and ensuring high-availability for your applications.

In this document we are going to clone color teller application from GIT repository and use that code to understand all resources created step by step so that we can understand how APPMESH works.

**High level diagram:**



**Pre Requisite** before you use your GIT repo:

1. Create an EC2 instance: ssh into the instance (ssh -i test.pem [ubuntu@18.141.240.103](mailto:ubuntu@18.141.240.103))

Run following command to download nodejs, typescript, AWS CLI, AWS CDK.

> sudo apt-get update

> sudo apt install curl

> curl -sL https://deb.nodesource.com/setup\_10.x | sudo -E bash -

> sudo apt install nodejs

> node --version

> npm --version

> sudo apt-get update

> sudo apt-get install awscli

> aws configure/ region and keys

> npm -g install typescript

> npm install -g aws-cdk

1. Before using CDK for fargate creation install docker from this link in ec2 instance <https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-18-04>

Or command given below:

sudo apt update

sudo apt install apt-transport-https ca-certificates curl software-properties-common

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add –

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable“

sudo apt update

apt-cache policy docker-ce

sudo apt install docker-ce

sudo systemctl status docker

**Creating APPMESH over FARGATE cluster using color teller application (Automated)**:

1. Clone GIT URL: clone any one of the GIT repository. Trimmed code is the modified code that will create only necessary resources to understand the app mesh.

git clone <https://github.com/chitranshuz/appmesh.git> (Trimmed code) or

git clone <https://github.com/enghwa/cdkcolorteller.git>

1. Go under the cloned project run following command for trimmed code

npm install

npm run build

cdk deploy

Follow this document for the other code:

<https://github.com/enghwa/cdkcolorteller>

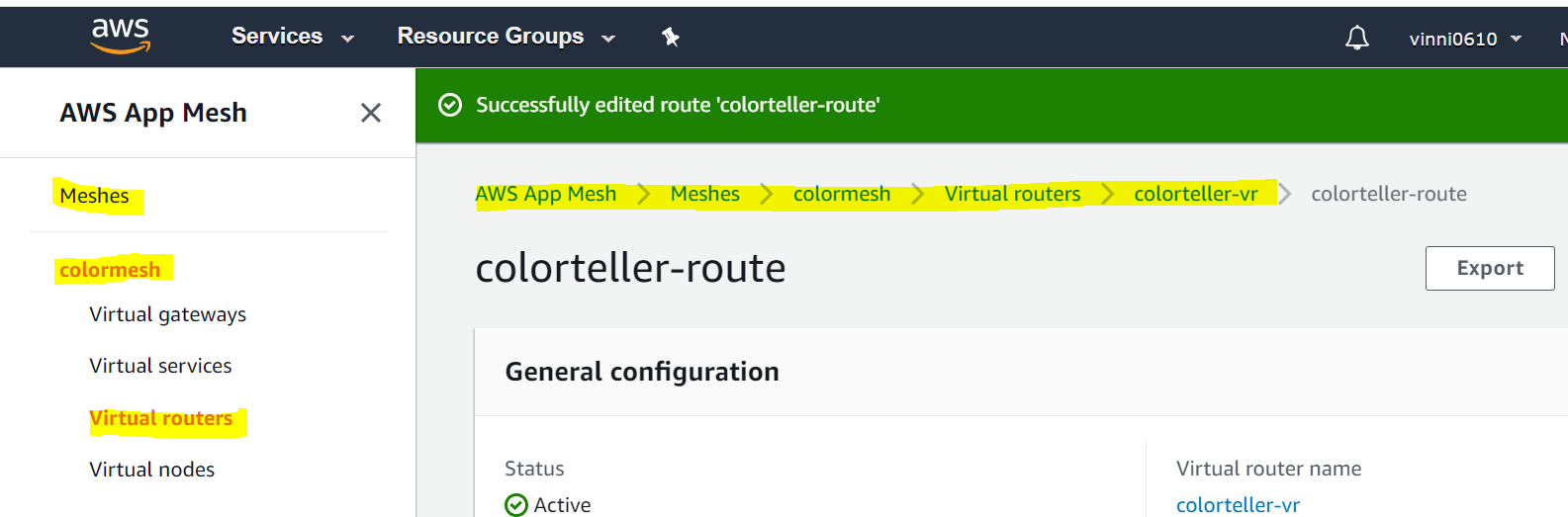
Once completed you will receive a load balancer in the output

1. Open chrome enter:

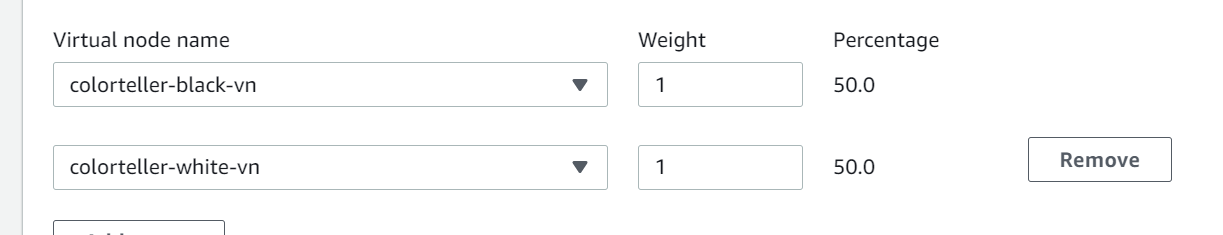
http://<ALB-DNS>/color

**Test:**

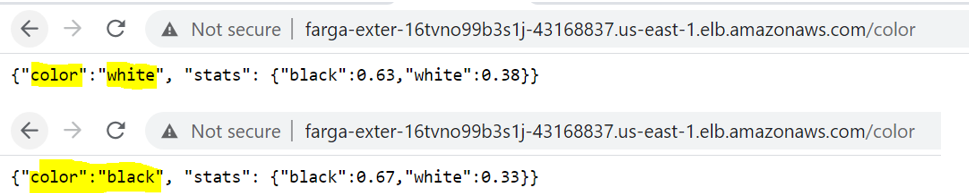
In AWS account go under appmesh -> virtual route open route scroll down open route.



Check if routing is 50:50 for apps you want to route traffic too.



1. Open url:

http://<ALB-DNS>/color

Paste the URL and refresh multiple times you will see the traffic is routing towards two different apps with same URL

**Destroy everything: destroy using this command instead of manually destroying the resources one by one.**

sudo cdk destroy –f

**Code explanation with resources created in AWS:**

Using code we are going to:

1. Created a VPC with 2 subnets.
2. Public facing load balancer pointing towards gateway service In ECS cluster.
3. Task definition for gateway and color teller application with container and envoy using images already pushed to Docker.
4. Launching Fargate service with task definitions and service name are registered in colordemo.local and also use service discovery methods to connect it with virtual nodes.
5. Created appmesh and its components.

**Table for resources created and linked with:**

1. Fargate cluster details:

|  |  |
| --- | --- |
| Service | Task definitions |
| [colorgatewayserviceServiceDD5B4D03-17KS2K0J26RG6](https://console.aws.amazon.com/ecs/home?region=us-east-1#/clusters/FargateAppmeshCdkStack-fgappMeshClusterA46305BE-ohTE0sTwrgEV/services/FargateAppmeshCdkStack-colorgatewayserviceServiceDD5B4D03-17KS2K0J26RG6/details) | [FargateAppmeshCdkStackcolorgatewaytaskdefinition39C34763](https://console.aws.amazon.com/ecs/home?region=us-east-1#/taskDefinitions/FargateAppmeshCdkStackcolorgatewaytaskdefinition39C34763/status/ACTIVE) |
| [colortellerblackserviceServiceC9571C54-1Q1OW9TSRSTQG](https://console.aws.amazon.com/ecs/home?region=us-east-1#/clusters/FargateAppmeshCdkStack-fgappMeshClusterA46305BE-ohTE0sTwrgEV/services/FargateAppmeshCdkStack-colortellerblackserviceServiceC9571C54-1Q1OW9TSRSTQG/details) | [FargateAppmeshCdkStackcolortellerblacktaskdefinitionEFDFEC5F](https://console.aws.amazon.com/ecs/home?region=us-east-1#/taskDefinitions/FargateAppmeshCdkStackcolortellerblacktaskdefinitionEFDFEC5F/status/ACTIVE) |
| [colortellerblueserviceService251F265D-943SQ5DOGF1I](https://console.aws.amazon.com/ecs/home?region=us-east-1#/clusters/FargateAppmeshCdkStack-fgappMeshClusterA46305BE-ohTE0sTwrgEV/services/FargateAppmeshCdkStack-colortellerblueserviceService251F265D-943SQ5DOGF1I/details) | [FargateAppmeshCdkStackcolortellerbluetaskdefinitionAB61A556](https://console.aws.amazon.com/ecs/home?region=us-east-1#/taskDefinitions/FargateAppmeshCdkStackcolortellerbluetaskdefinitionAB61A556/status/ACTIVE) |
| [colortellerredserviceServiceCAF33827-146POJ6CDS9SC](https://console.aws.amazon.com/ecs/home?region=us-east-1#/clusters/FargateAppmeshCdkStack-fgappMeshClusterA46305BE-ohTE0sTwrgEV/services/FargateAppmeshCdkStack-colortellerredserviceServiceCAF33827-146POJ6CDS9SC/details) | [FargateAppmeshCdkStackcolortellerredtaskdefinition6A0127EB](https://console.aws.amazon.com/ecs/home?region=us-east-1#/taskDefinitions/FargateAppmeshCdkStackcolortellerredtaskdefinition6A0127EB/status/ACTIVE) |
| [colortellerwhiteserviceService0E925EC1-1HJ69JZTMA8DD](https://console.aws.amazon.com/ecs/home?region=us-east-1#/clusters/FargateAppmeshCdkStack-fgappMeshClusterA46305BE-ohTE0sTwrgEV/services/FargateAppmeshCdkStack-colortellerwhiteserviceService0E925EC1-1HJ69JZTMA8DD/details) | [FargateAppmeshCdkStackcolortellerwhitetaskdefinitionD2CDB525](https://console.aws.amazon.com/ecs/home?region=us-east-1#/taskDefinitions/FargateAppmeshCdkStackcolortellerwhitetaskdefinitionD2CDB525/status/ACTIVE) |

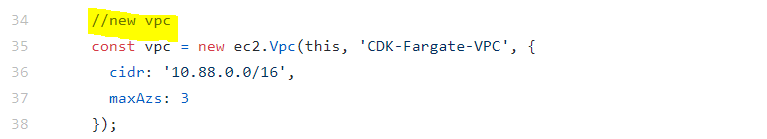
1. Task definitions- containers and images used

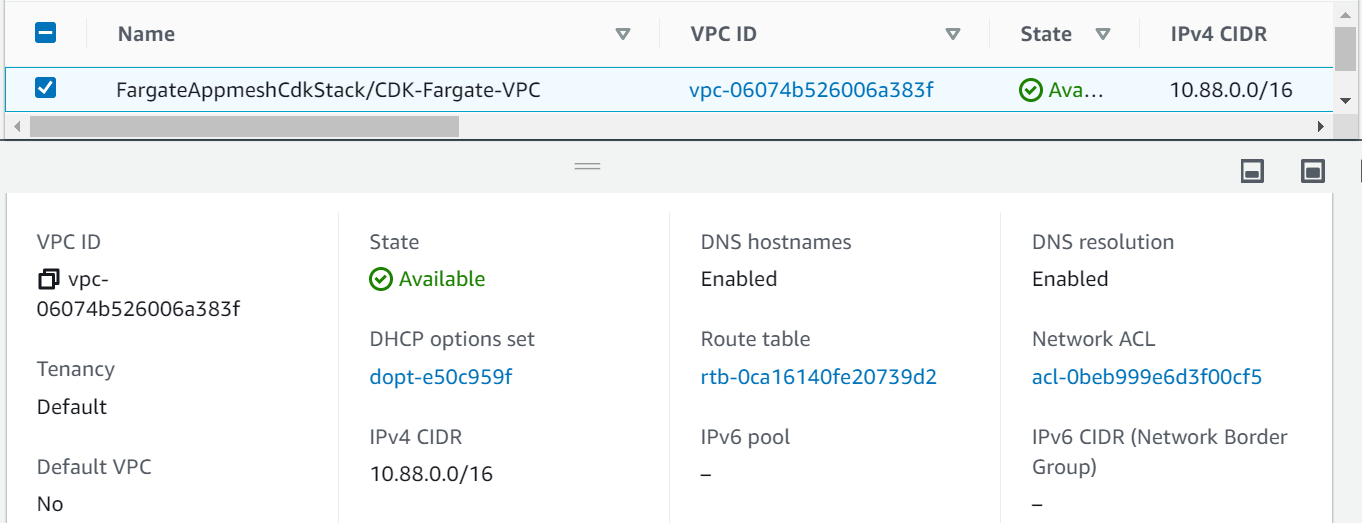
|  |  |  |
| --- | --- | --- |
| Task definition for service | Containers type | Docker images |
| [FargateAppmeshCdkStackcolorgatewaytaskdefinition39C34763](https://console.aws.amazon.com/ecs/home?region=us-east-1#/taskDefinitions/FargateAppmeshCdkStackcolorgatewaytaskdefinition39C34763/status/ACTIVE) | colorgateway | kopi/colorgateway:latest |
| envoy | kopi/appmesh:latest |
| [FargateAppmeshCdkStackcolortellerblacktaskdefinitionEFDFEC5F](https://console.aws.amazon.com/ecs/home?region=us-east-1#/taskDefinitions/FargateAppmeshCdkStackcolortellerblacktaskdefinitionEFDFEC5F/status/ACTIVE) | colortellerApp | kopi/colorteller |
| envoy | kopi/appmesh:latest |
| [FargateAppmeshCdkStackcolortellerbluetaskdefinitionAB61A556](https://console.aws.amazon.com/ecs/home?region=us-east-1#/taskDefinitions/FargateAppmeshCdkStackcolortellerbluetaskdefinitionAB61A556/status/ACTIVE) | colortellerApp | kopi/colorteller |
| envoy | kopi/appmesh:latest |
| [FargateAppmeshCdkStackcolortellerredtaskdefinition6A0127EB](https://console.aws.amazon.com/ecs/home?region=us-east-1#/taskDefinitions/FargateAppmeshCdkStackcolortellerredtaskdefinition6A0127EB/status/ACTIVE) | colortellerApp | kopi/colorteller |
| envoy | kopi/appmesh:latest |
| [FargateAppmeshCdkStackcolortellerwhitetaskdefinitionD2CDB525](https://console.aws.amazon.com/ecs/home?region=us-east-1#/taskDefinitions/FargateAppmeshCdkStackcolortellerwhitetaskdefinitionD2CDB525/status/ACTIVE) | colortellerApp | kopi/colorteller |
| envoy | kopi/appmesh:latest |

1. Services linked with AppMesh virtual nodes

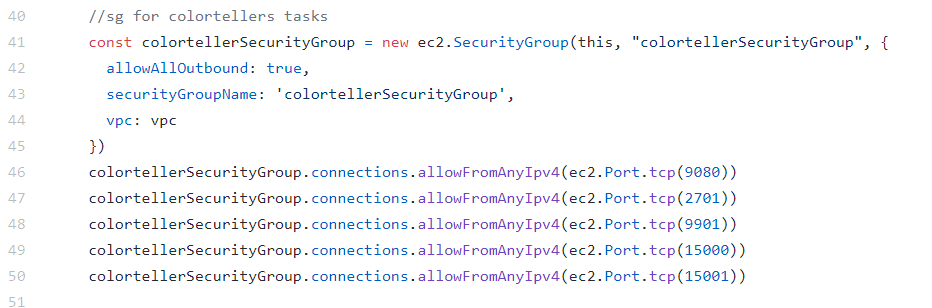
|  |  |  |
| --- | --- | --- |
| Service | Virtual nodes in Appmesh | DNS NAME used in Fargate service discovery |
| [colorgatewayserviceServiceDD5B4D03-17KS2K0J26RG6](https://console.aws.amazon.com/ecs/home?region=us-east-1#/clusters/FargateAppmeshCdkStack-fgappMeshClusterA46305BE-ohTE0sTwrgEV/services/FargateAppmeshCdkStack-colorgatewayserviceServiceDD5B4D03-17KS2K0J26RG6/details) | [colorgateway-vn](https://console.aws.amazon.com/appmesh/148981515206/meshes/colormesh/virtual-nodes/colorgateway-vn?region=us-east-1) | colorgateway.colordemo.local |
| [colortellerblackserviceServiceC9571C54-1Q1OW9TSRSTQG](https://console.aws.amazon.com/ecs/home?region=us-east-1#/clusters/FargateAppmeshCdkStack-fgappMeshClusterA46305BE-ohTE0sTwrgEV/services/FargateAppmeshCdkStack-colortellerblackserviceServiceC9571C54-1Q1OW9TSRSTQG/details) | [colorteller-black-vn](https://console.aws.amazon.com/appmesh/148981515206/meshes/colormesh/virtual-nodes/colorteller-black-vn?region=us-east-1) | colorteller-black.colordemo.local |
| [colortellerblueserviceService251F265D-943SQ5DOGF1I](https://console.aws.amazon.com/ecs/home?region=us-east-1#/clusters/FargateAppmeshCdkStack-fgappMeshClusterA46305BE-ohTE0sTwrgEV/services/FargateAppmeshCdkStack-colortellerblueserviceService251F265D-943SQ5DOGF1I/details) | [colorteller-blue-vn](https://console.aws.amazon.com/appmesh/148981515206/meshes/colormesh/virtual-nodes/colorteller-blue-vn?region=us-east-1) | colorteller-blue.colordemo.local |
| [colortellerredserviceServiceCAF33827-146POJ6CDS9SC](https://console.aws.amazon.com/ecs/home?region=us-east-1#/clusters/FargateAppmeshCdkStack-fgappMeshClusterA46305BE-ohTE0sTwrgEV/services/FargateAppmeshCdkStack-colortellerredserviceServiceCAF33827-146POJ6CDS9SC/details) | [colorteller-red-vn](https://console.aws.amazon.com/appmesh/148981515206/meshes/colormesh/virtual-nodes/colorteller-red-vn?region=us-east-1) | colorteller-red.colordemo.local |
| [colortellerwhiteserviceService0E925EC1-1HJ69JZTMA8DD](https://console.aws.amazon.com/ecs/home?region=us-east-1#/clusters/FargateAppmeshCdkStack-fgappMeshClusterA46305BE-ohTE0sTwrgEV/services/FargateAppmeshCdkStack-colortellerwhiteserviceService0E925EC1-1HJ69JZTMA8DD/details) | [colorteller-white-vn](https://console.aws.amazon.com/appmesh/148981515206/meshes/colormesh/virtual-nodes/colorteller-white-vn?region=us-east-1) | colorteller.colordemo.local |

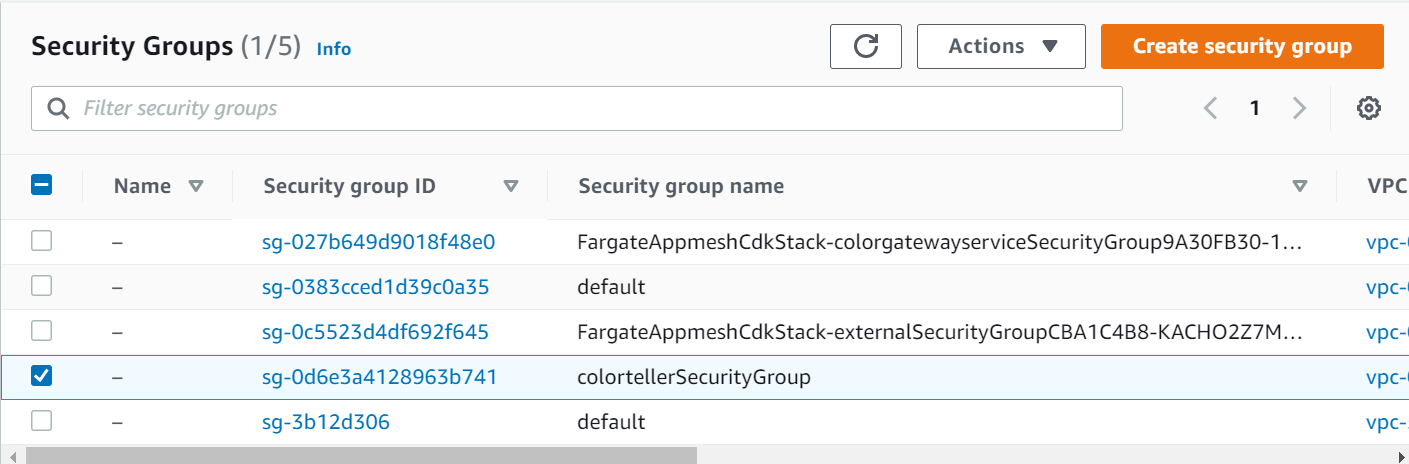
Open file “lib/fargate-appmesh-cdk-stack.ts” in you EC2 instance. Follow the comment and look into you console to understand which part of code created which all resources. Which is also given below:

1. First we will create VPC with cidr group 10.88.0.0/16.

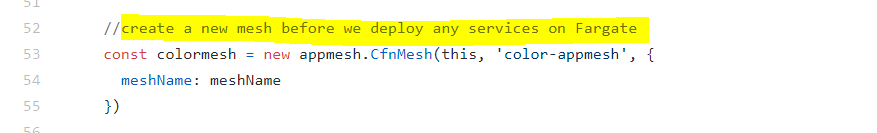
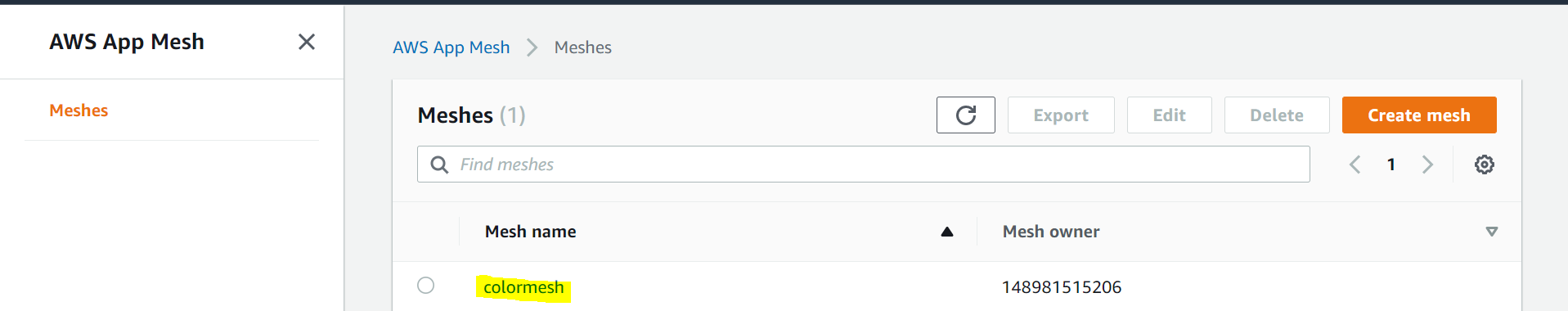


1. Security group.

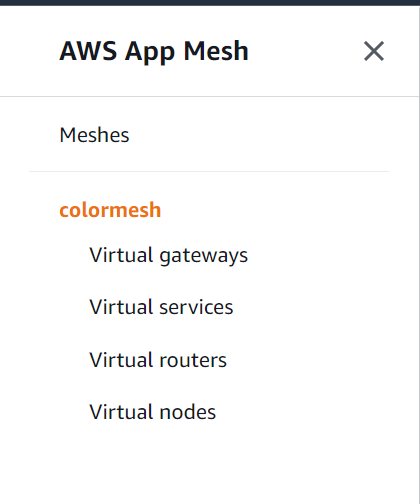




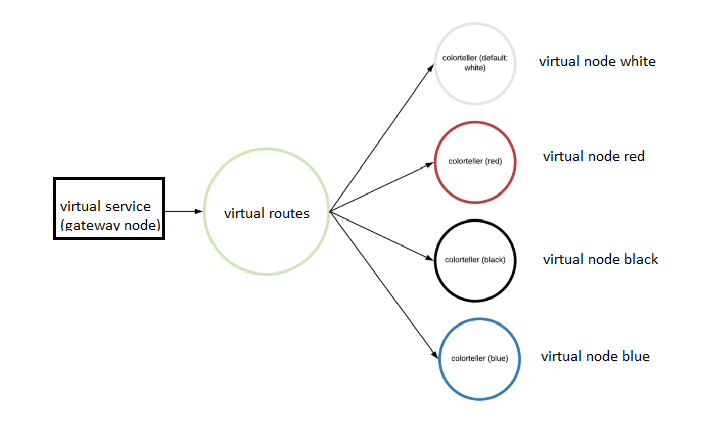
1. Mesh creation.

1. Any mesh containes the following.

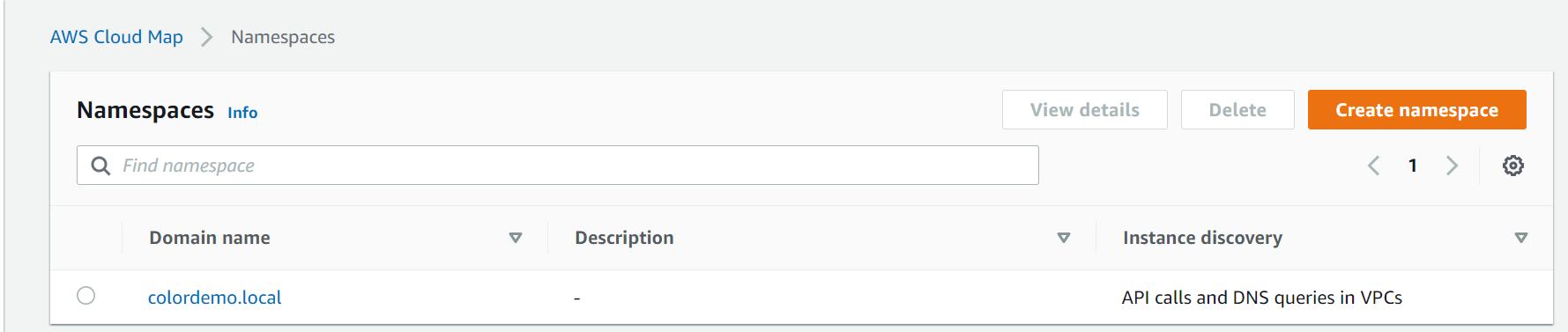


Flow diagram: how virtual service route traffic to nodes.

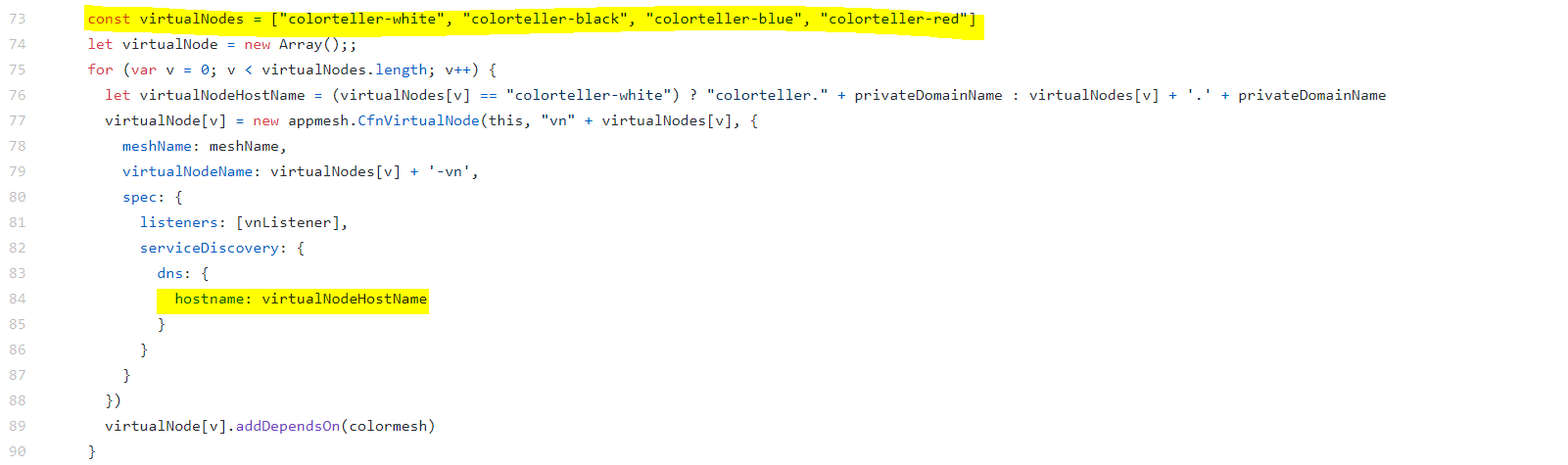


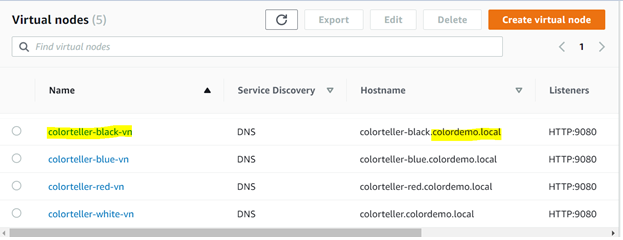
**Virtual nodes**:

We will create a virtual node under private domain name defined in cloud map.



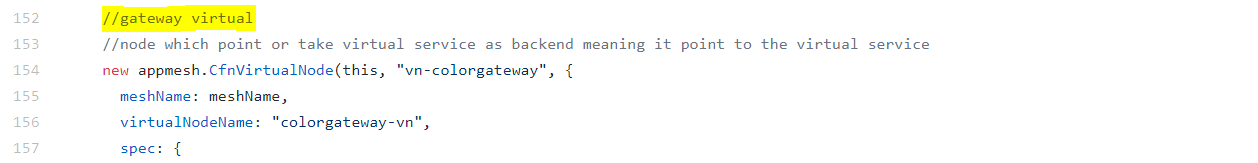
This part of code creates nodes under DNS name defined.

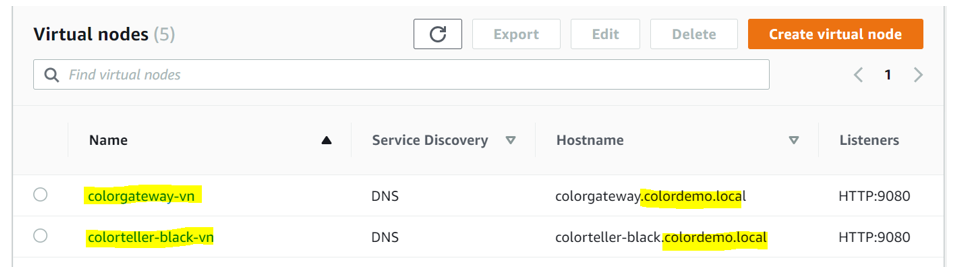


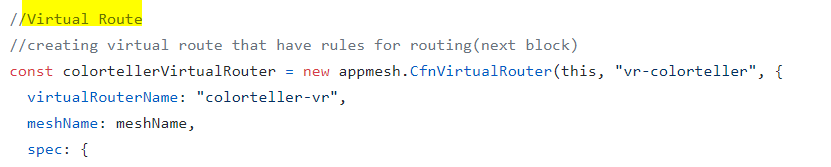
Once done our node will look like this with their common DNS name colordemo.local 

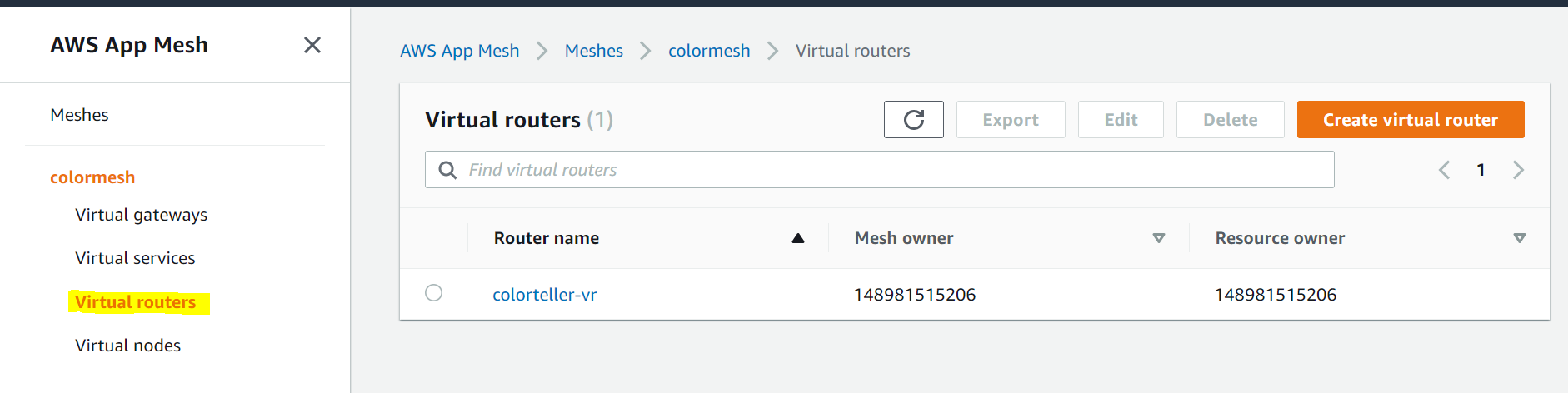
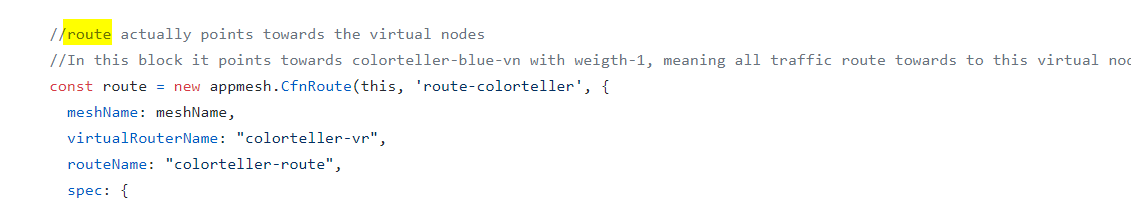
Every node will be linked with the fargate service we will create in coming steps.

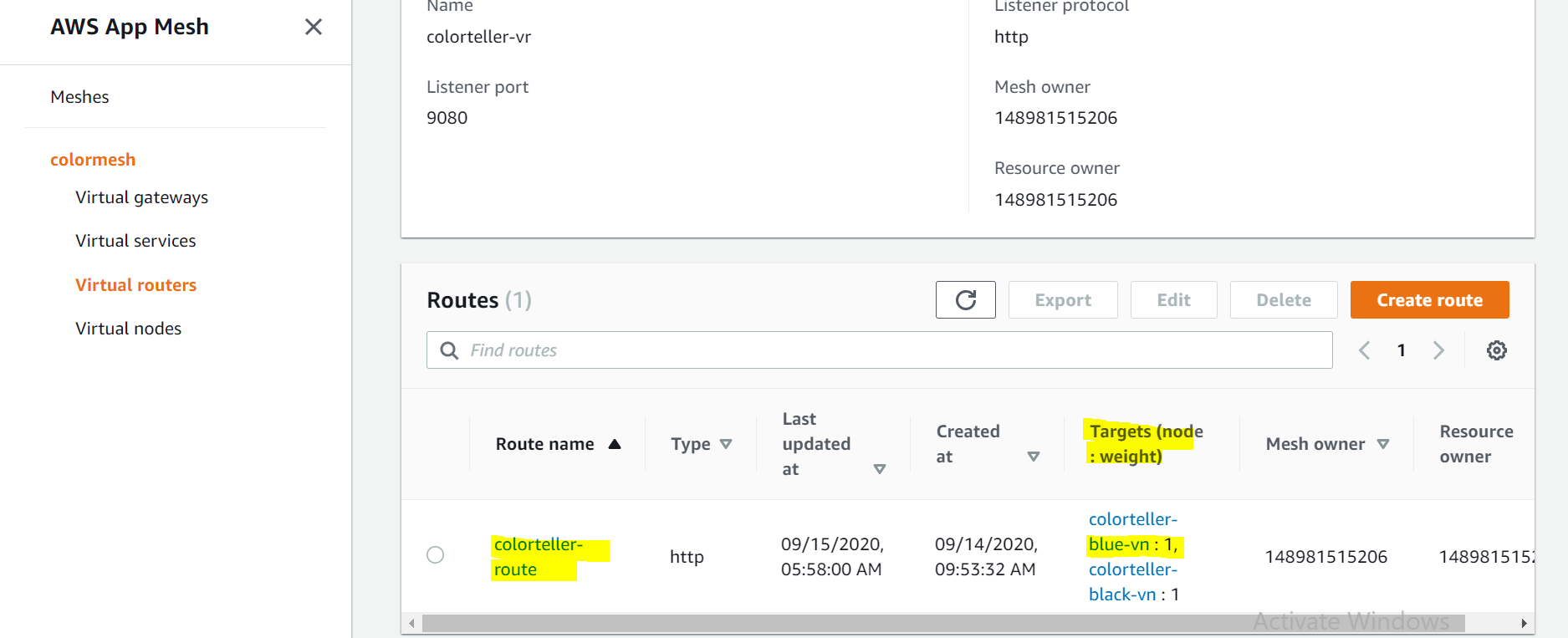
**Gateway node**: this node points to our actual vertual service.



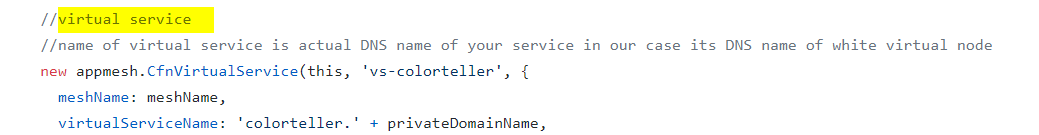


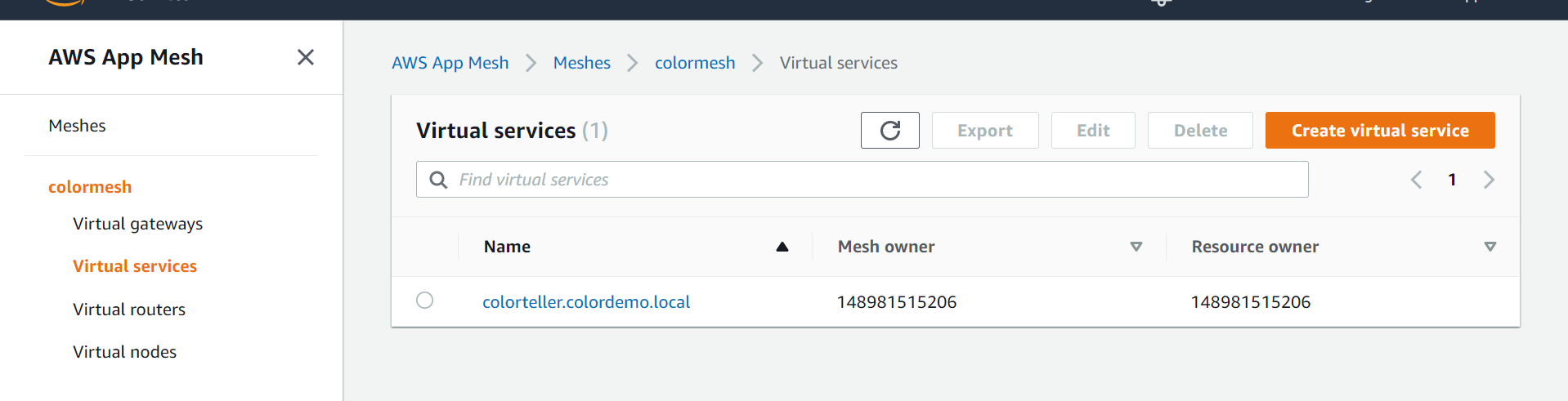
**Routes**: used to transfer traffic. 

Routing complete traffic to the blue color teller application in fargate which we will integrate with virtual node in app mesh in coming steps. 

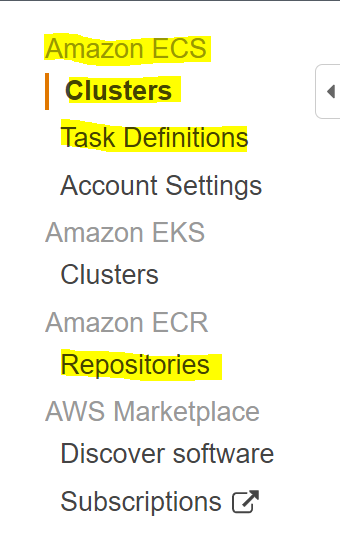


**Virtual service:**





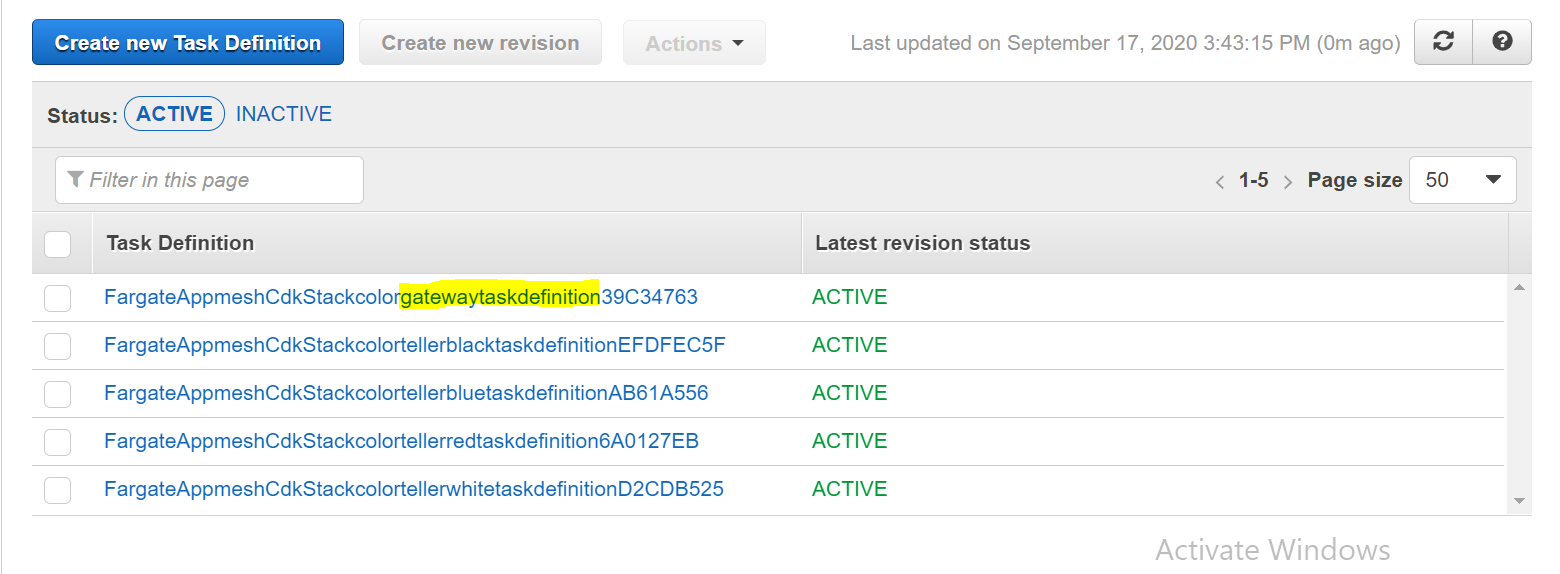
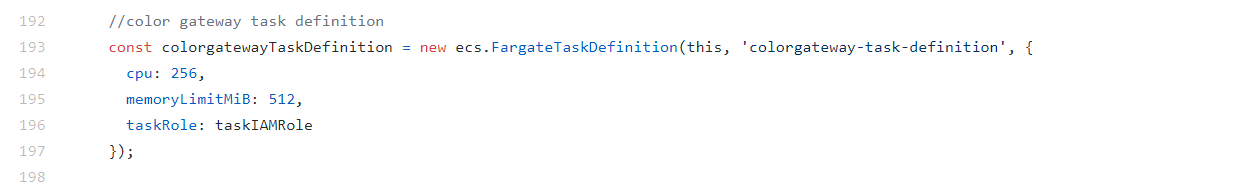
1. FARGATE cluster:

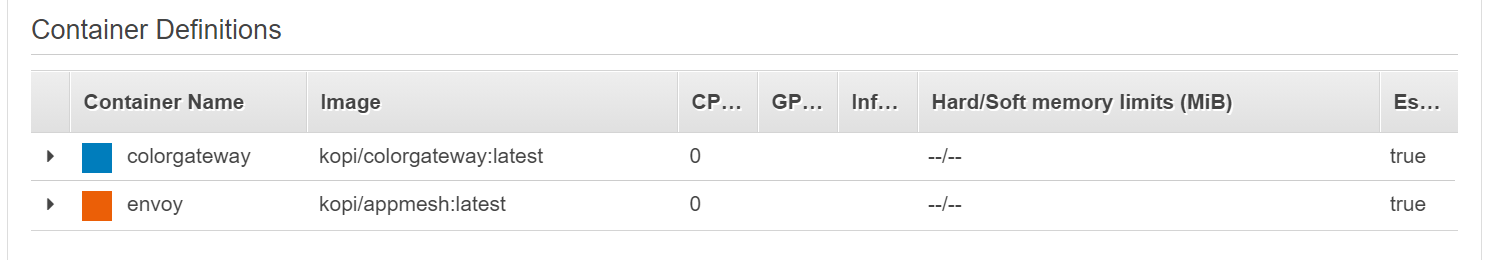
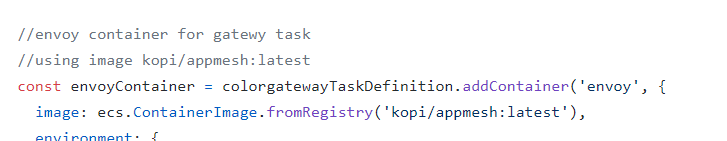
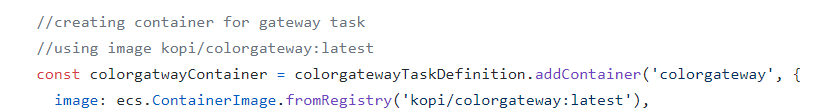


**Repositories**: contains the Docker images in our case we are using images from Docker hub which are already pushed.

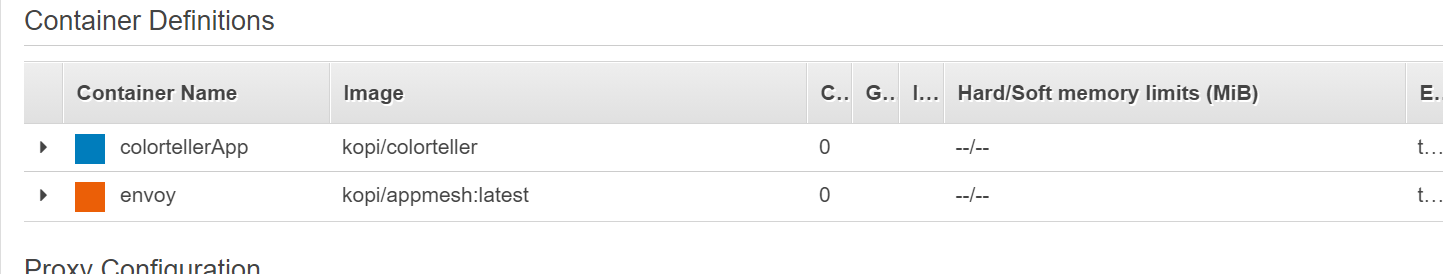
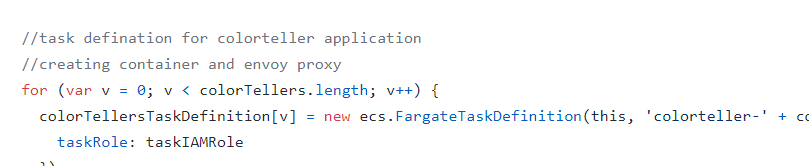
**Task definition**:

Gateway task definition contains fargate service details. Like container, envoy and Docker images it should use.



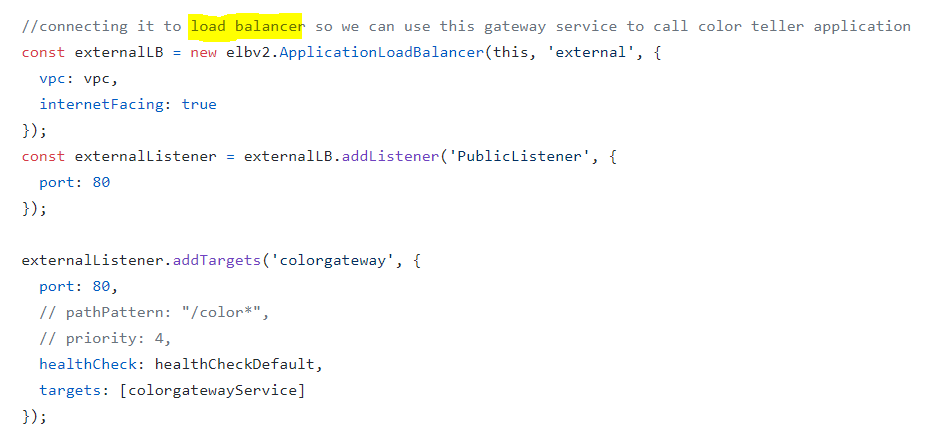
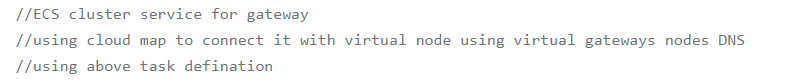
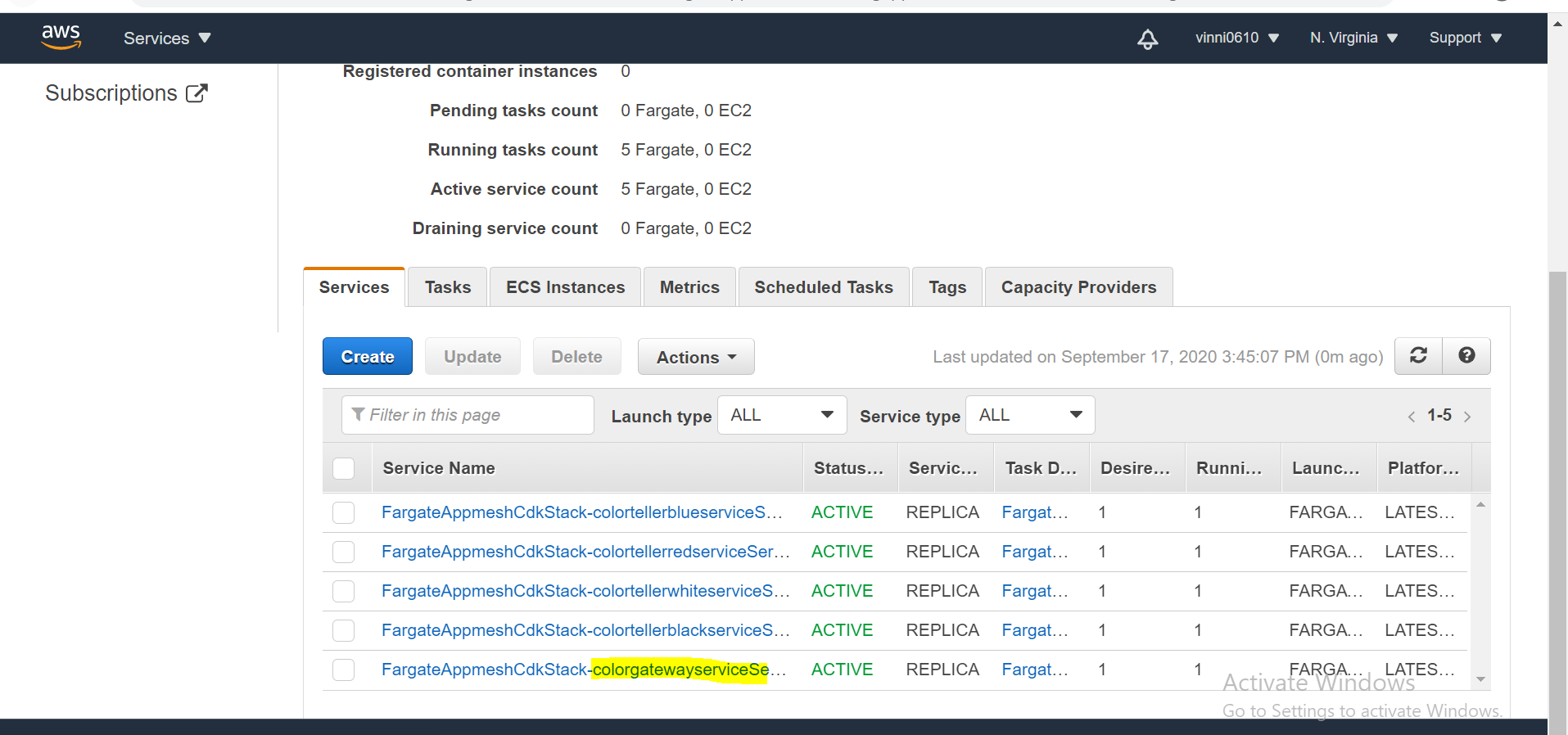
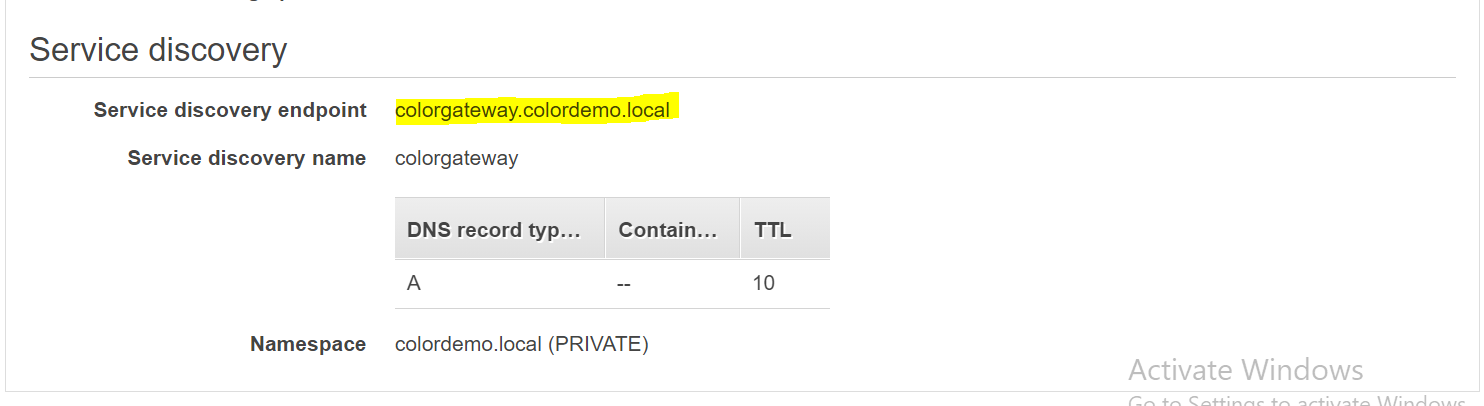
Creating container and envoy for gateway service

Other task defination are defined in this block and all are same.



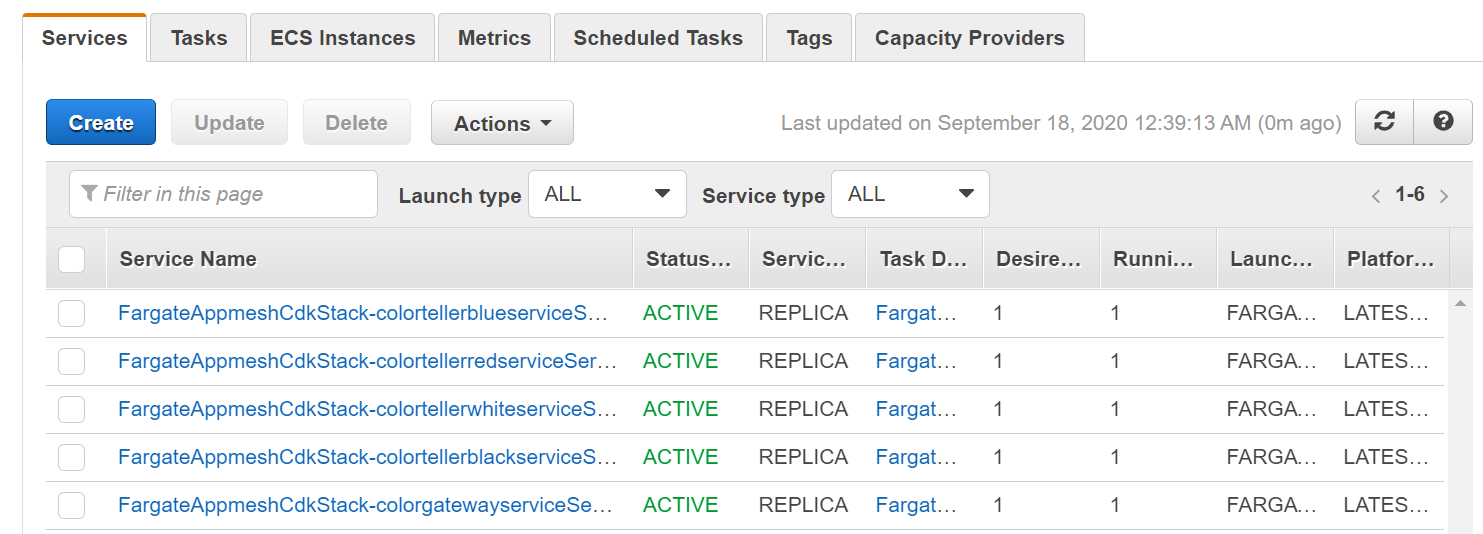
**Clusters -> Services:**

Creating Gateway service using gateway task definition attached to the ALB and also connecting it with appmesh gateway node using cloud map.

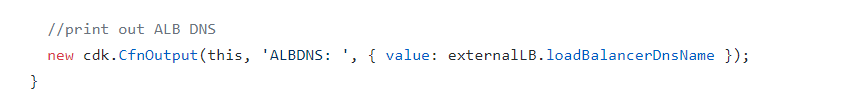
  

Pointing towards the gateway virtual node with DNS name colorgateway.colordemo.local

Similarly all color task and service is defined inside one block see comment 



1. Outputs load balancer after resources are created after cdk deploy command.



Summary:

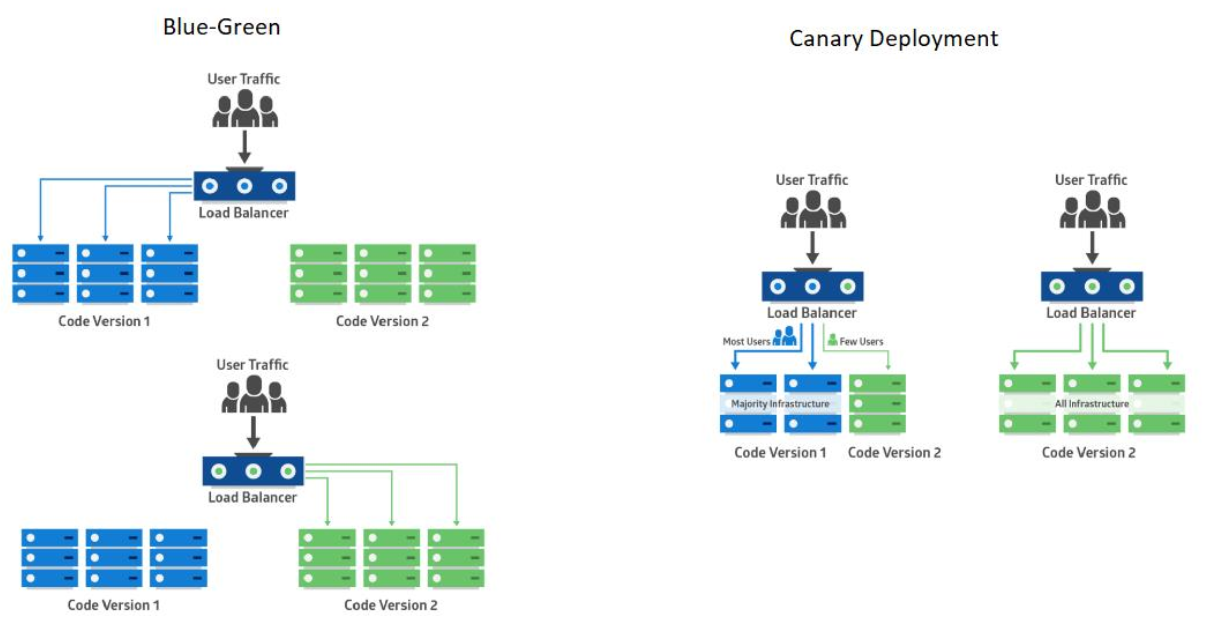
1. We created a VPC with 2 subnets.
2. Public facing load balancer pointing towards gateway service In ECS cluster.
3. Task definition for gateway and color teller application with container and envoy using images already pushed to Docker.
4. Fargate service are launched with task definitions and service name are registered in colordemo.local and also used service discovery methods to connect it with virtual nodes.
5. Created appmesh and its components.

**MANUAL deployment of black application version 2(Dark black)**

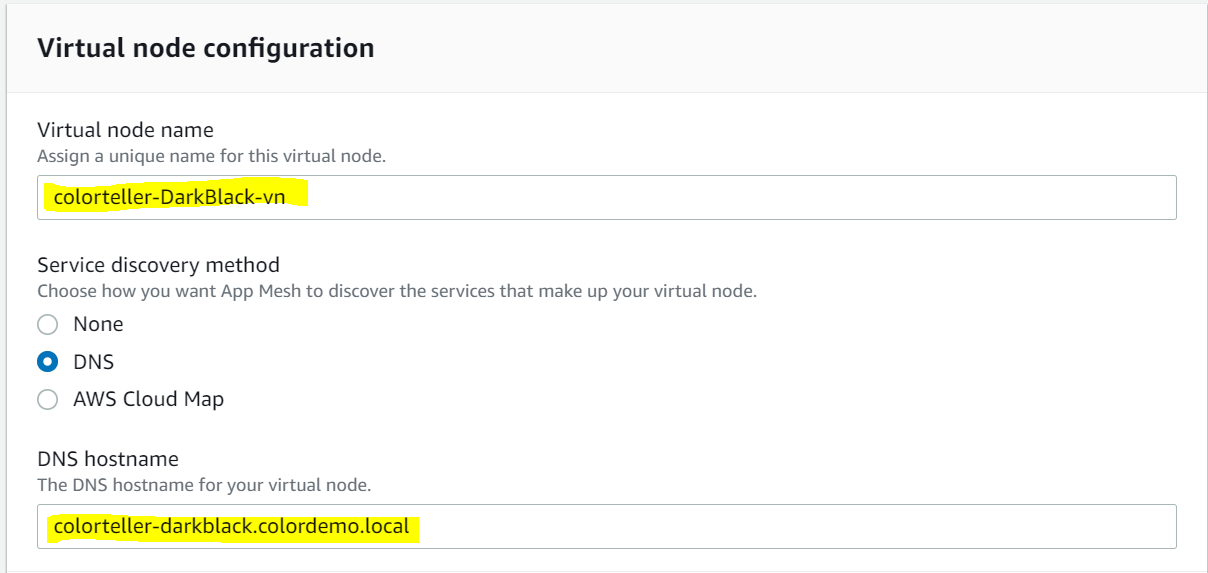
In this section we are going to use console to deploy version 2 of black app which outputs Dark Black.

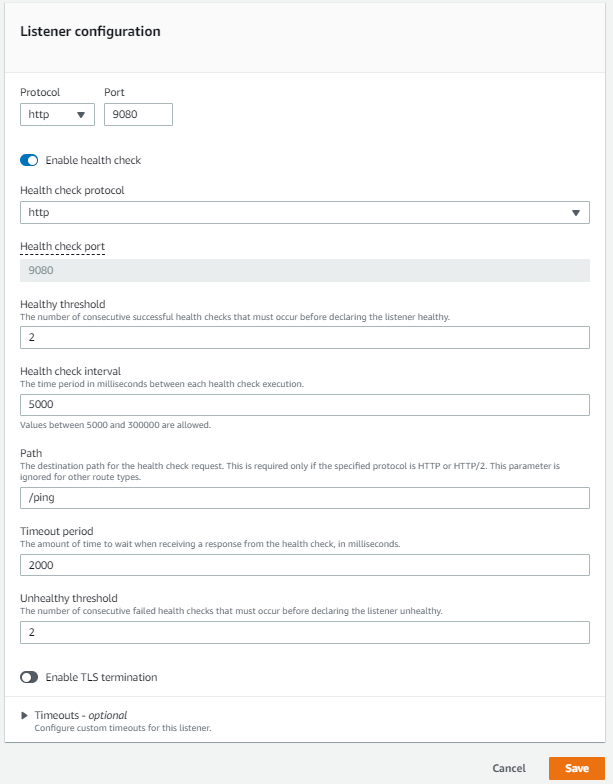
We will use canary deployment: meaning that we will deploy dark black application and route only 50% traffic towards it if it works fine we will transfer 100% to our new version that is dark black. Canary deployment is same as the blue green deployment the only difference is in canary we transfer traffic slowly instead of switching it in one step.

Diagrammatic representation of difference:

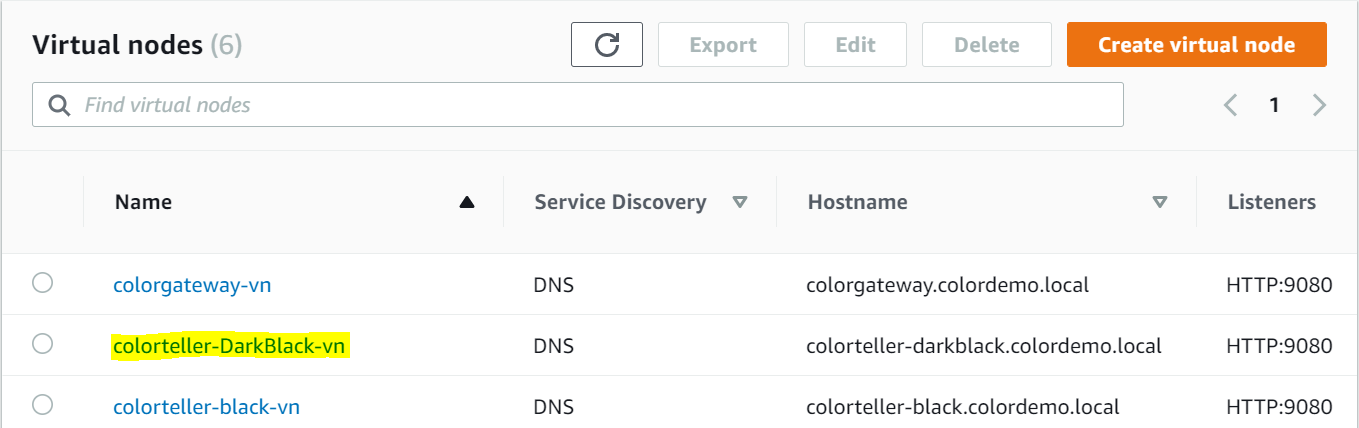


1. Go under virtual nodes create new node with following configurations.

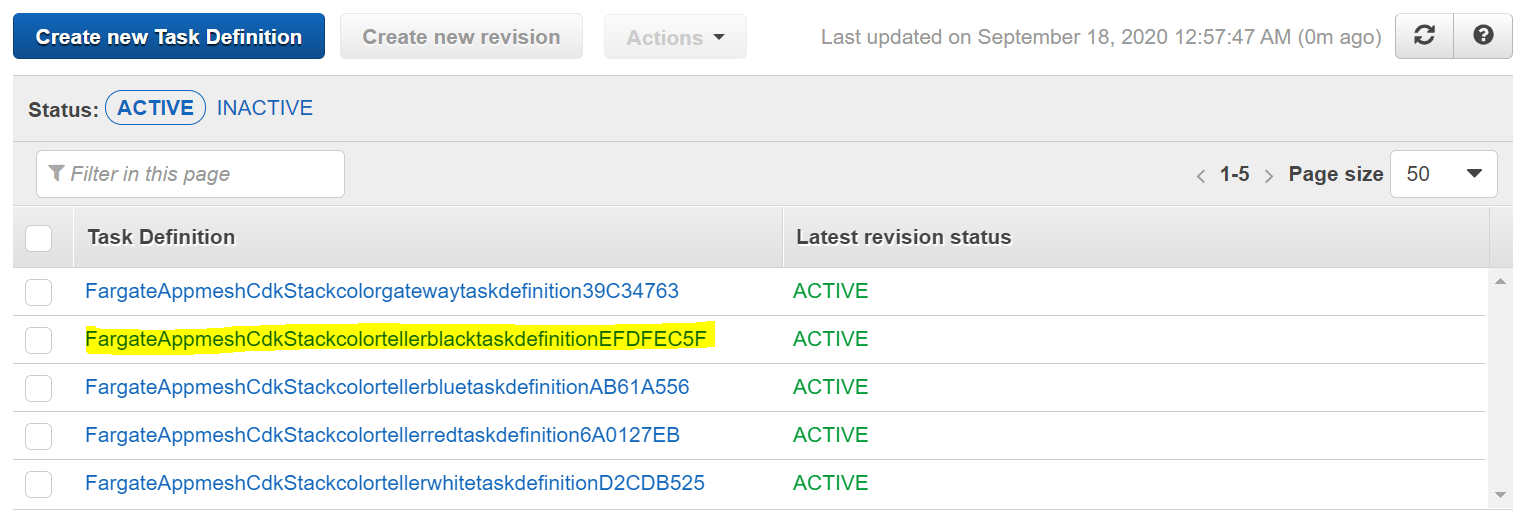




Click save.



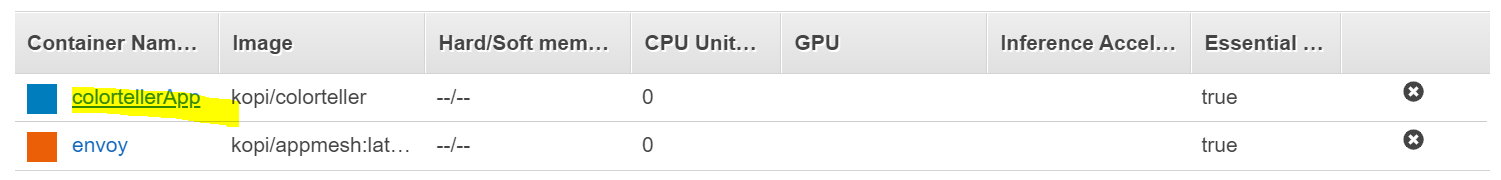
1. We will make change in task definition for black service. Open ECS -> task definations

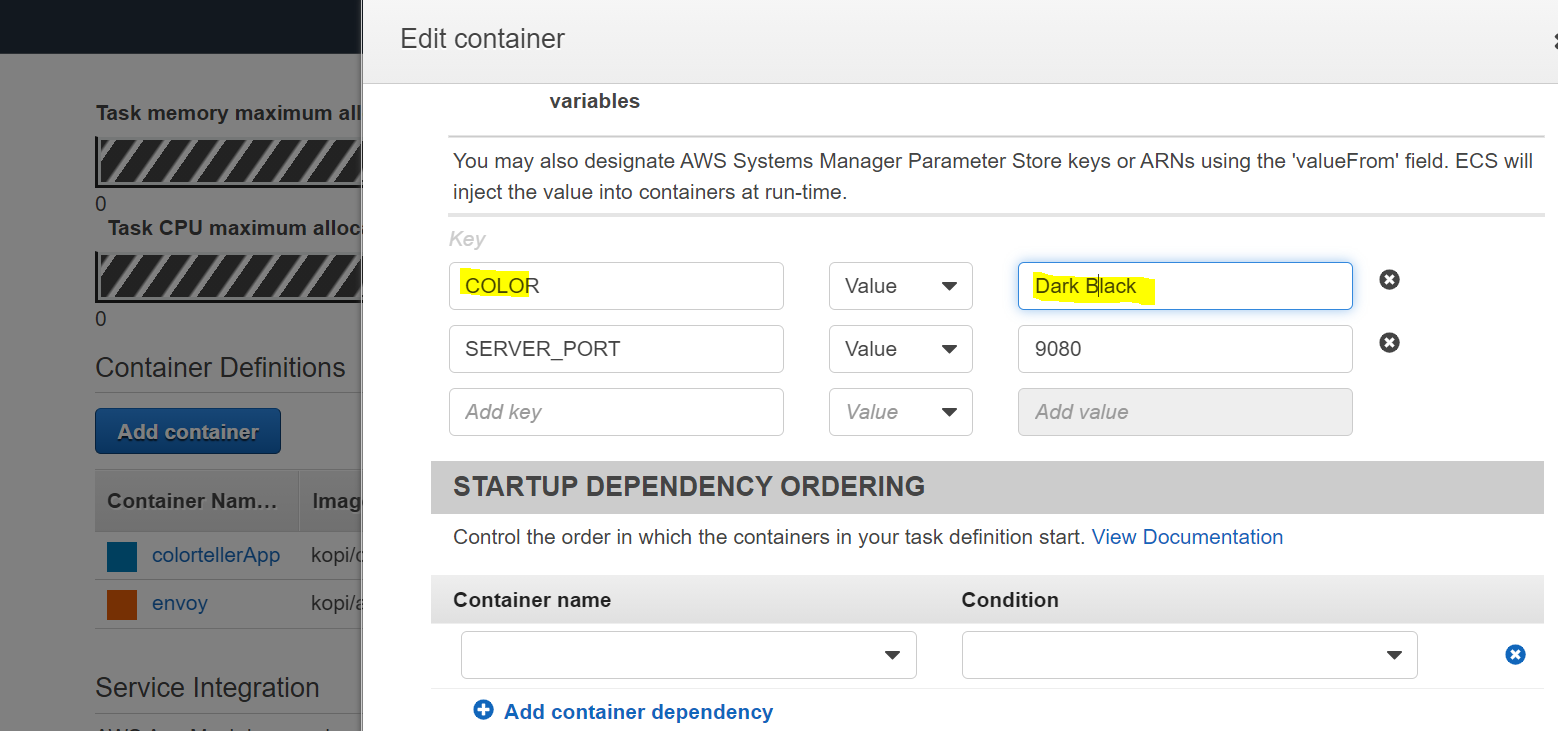
Open black task defination

Select already existing task click create new task definition

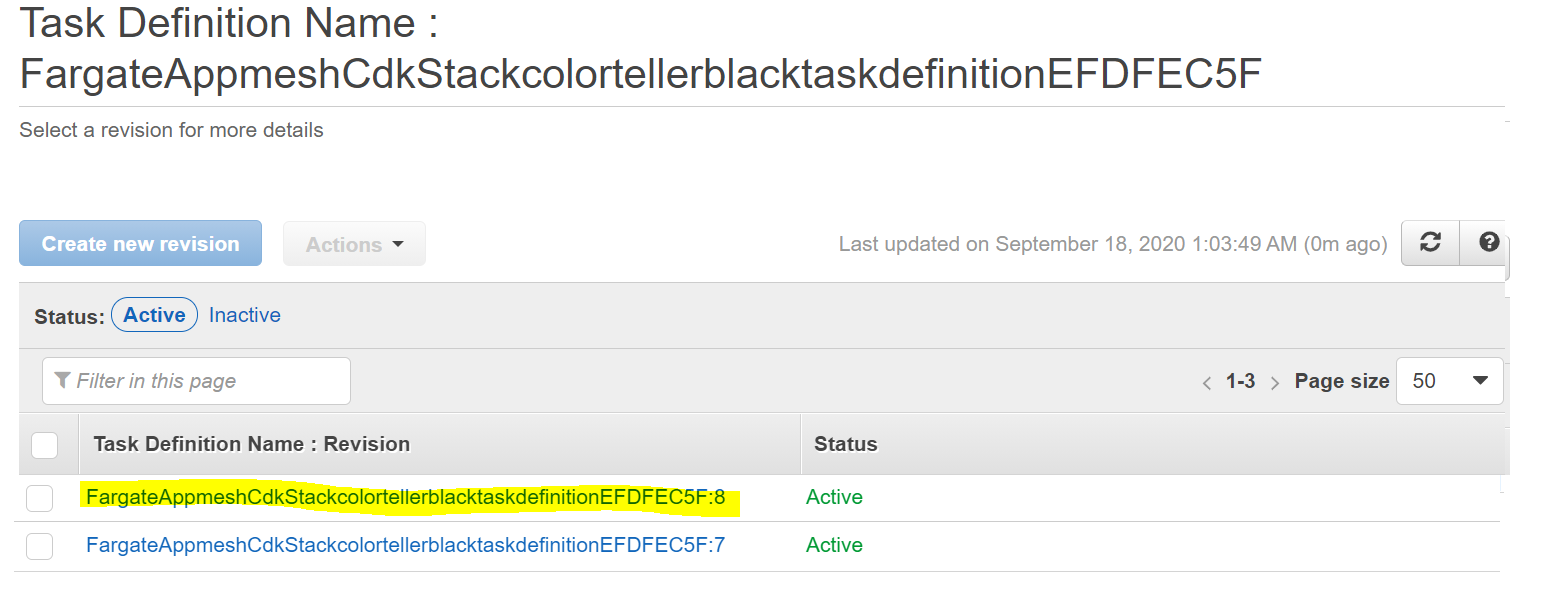
Scroll down open and change colortellerApp container

Change environment variable from here color teller application pick up the color.



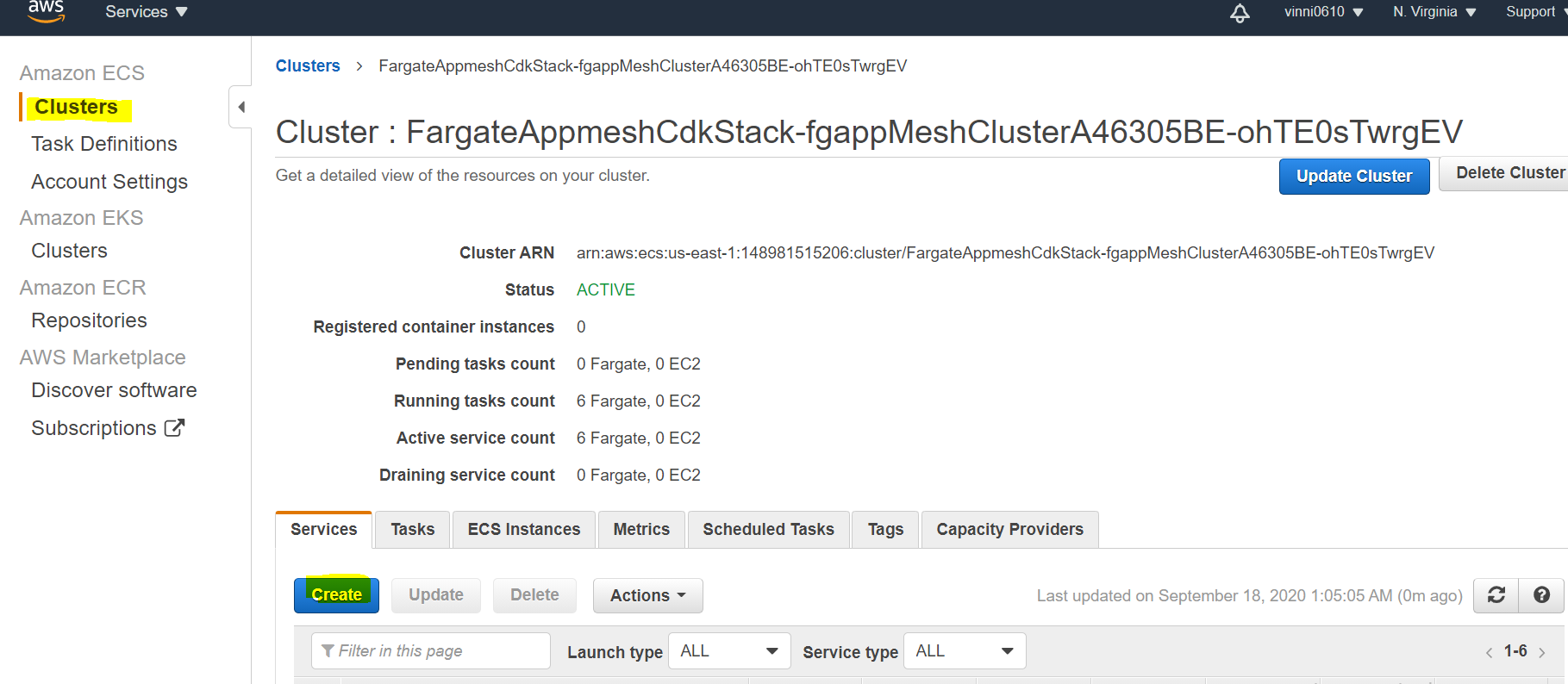


Save the changes of container and scroll down create new task

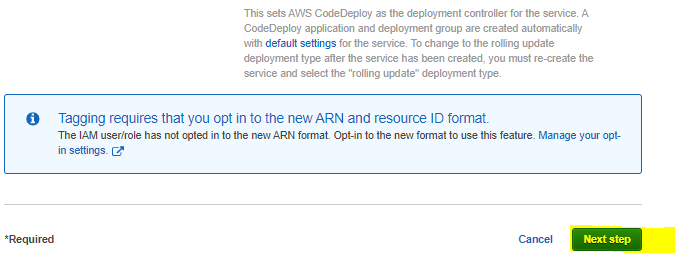
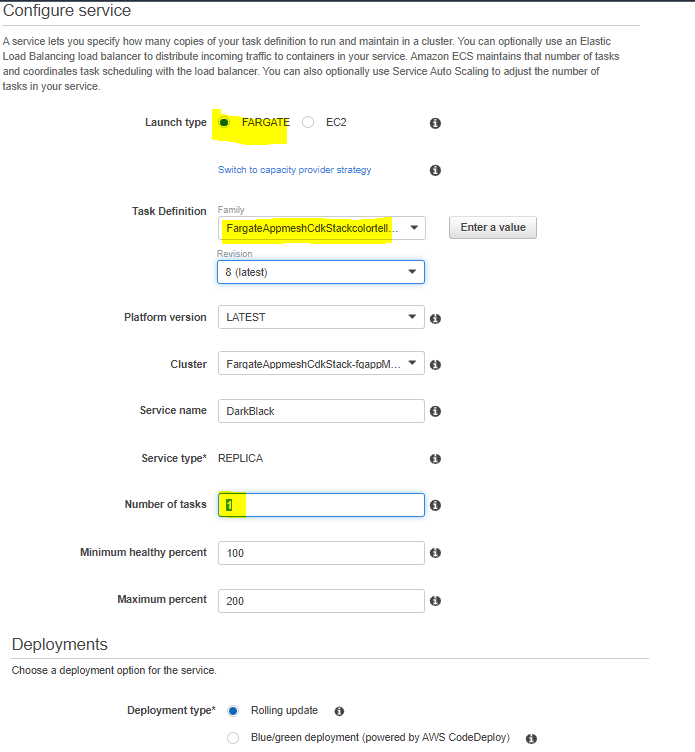


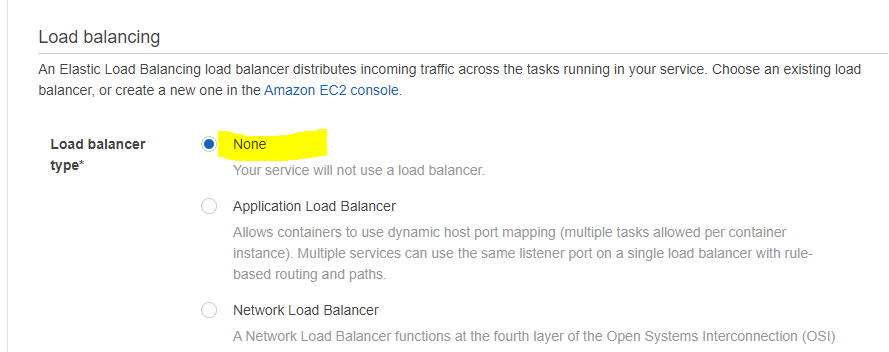
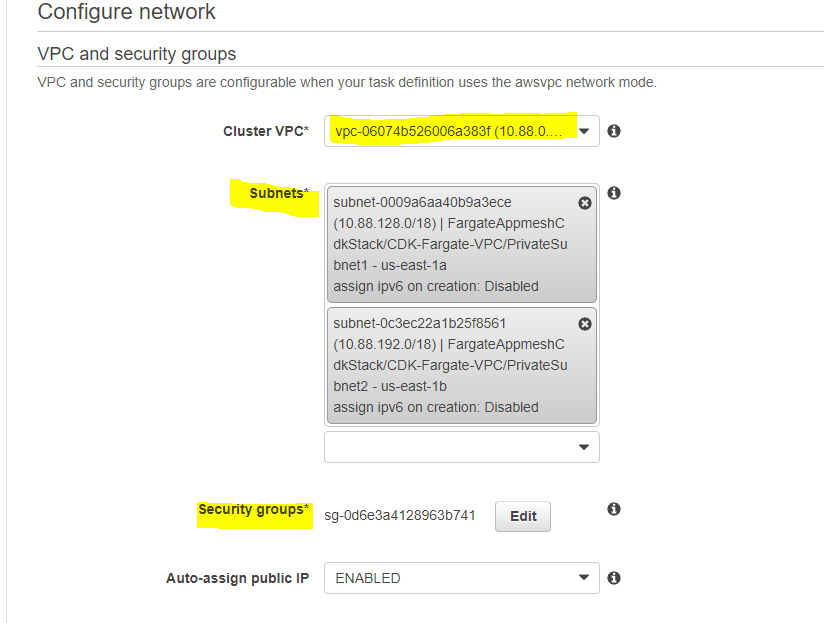
1. Create a service

Under cluster click create.

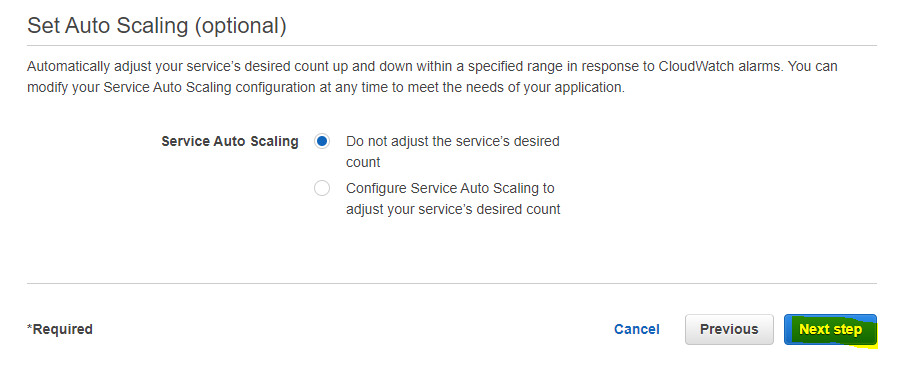
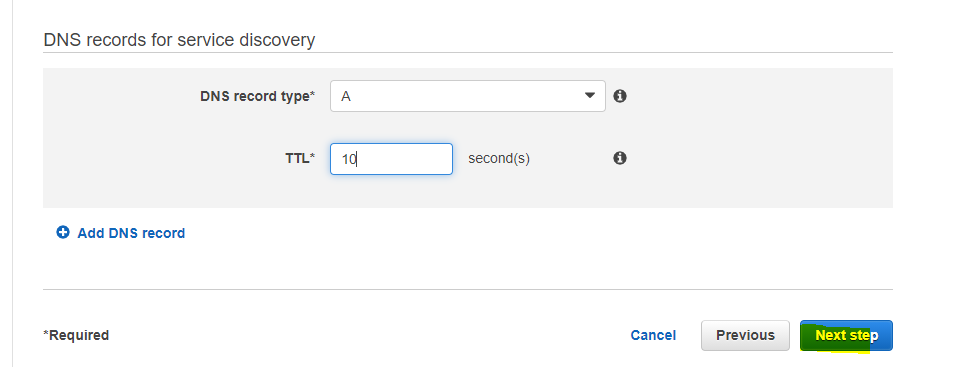
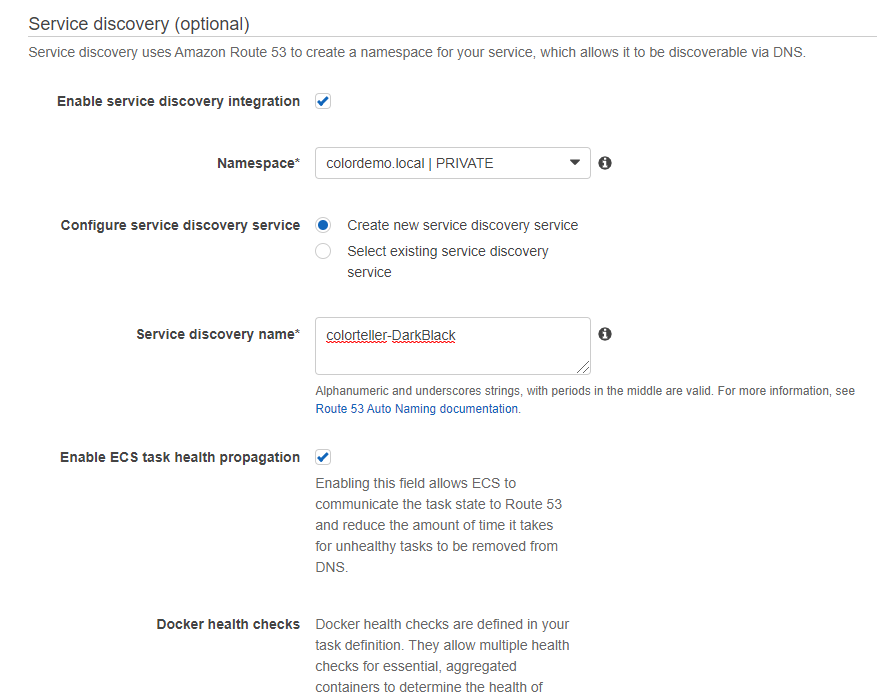


Give following configurations



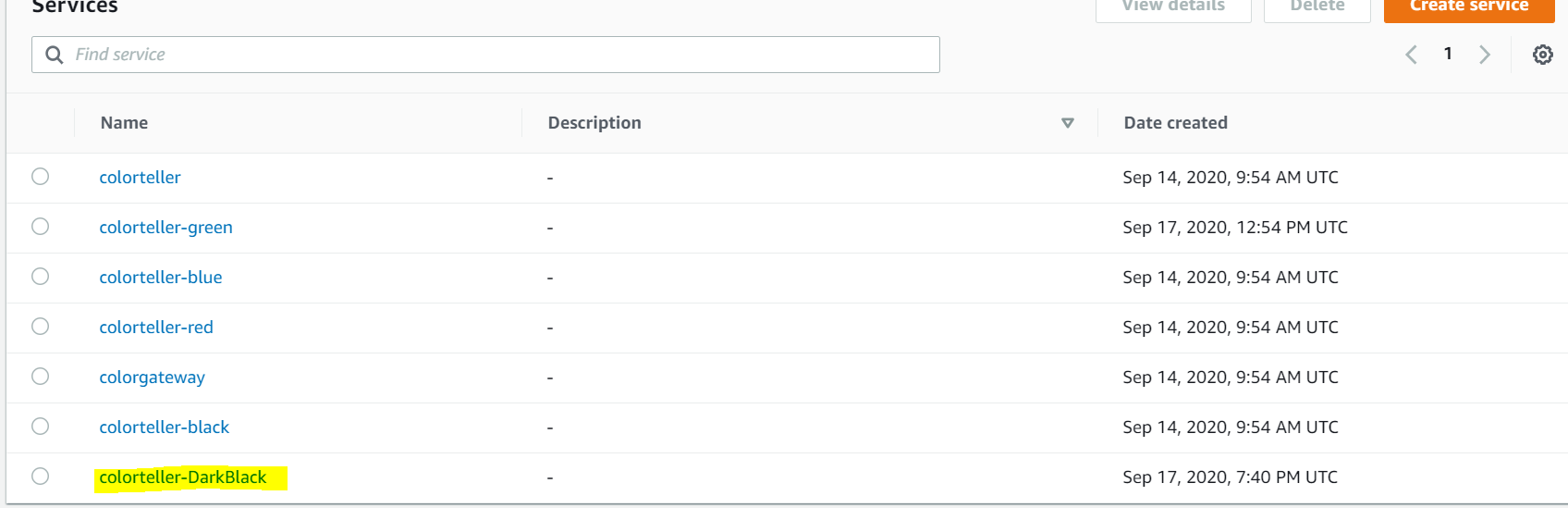


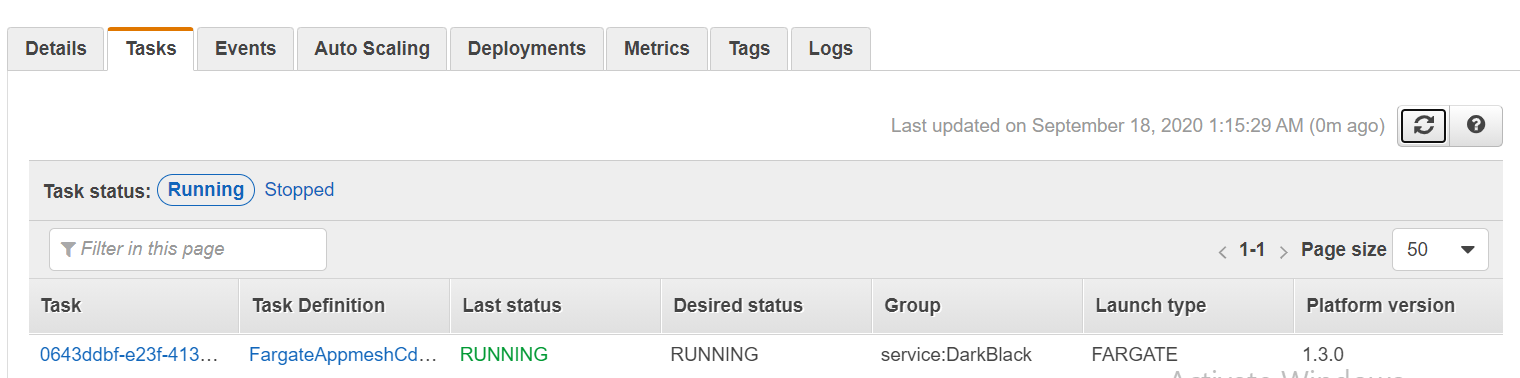
In this part you will connect you service to the DNS name of you virtual node using cloud map as service discovery method.



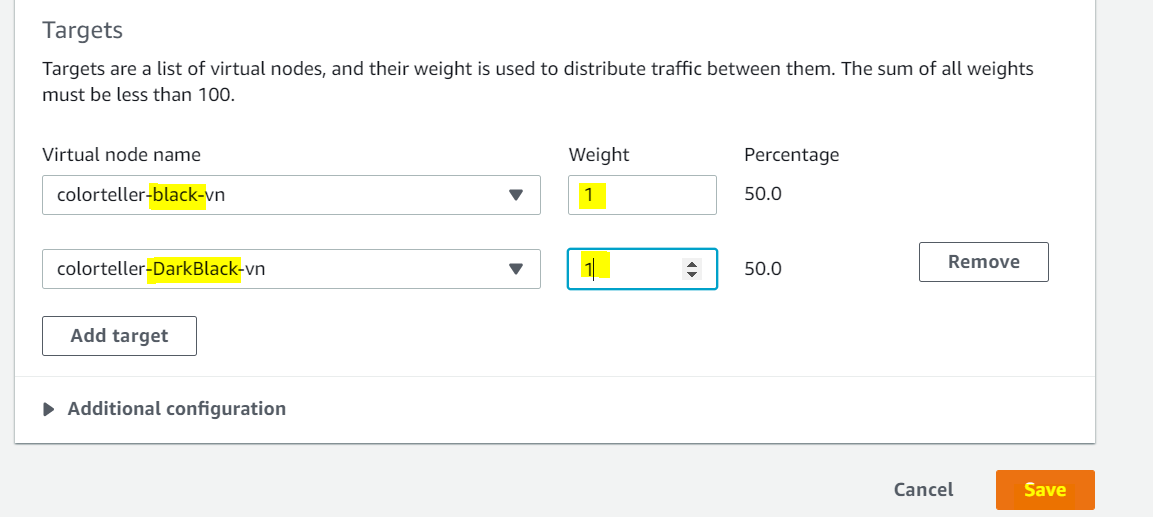
Review and create

Go to cloud map and check if service is created or not



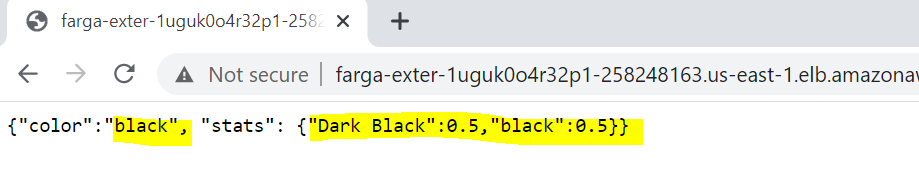
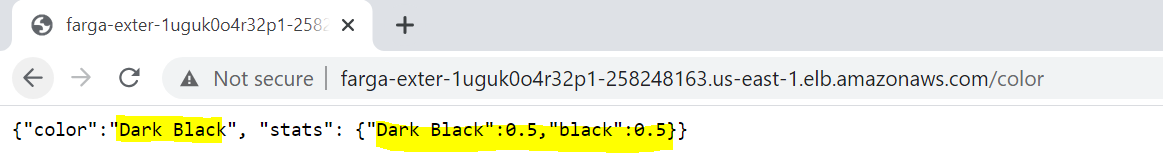
Once task is running 

Go inside Appmesh change the routing in virtual route

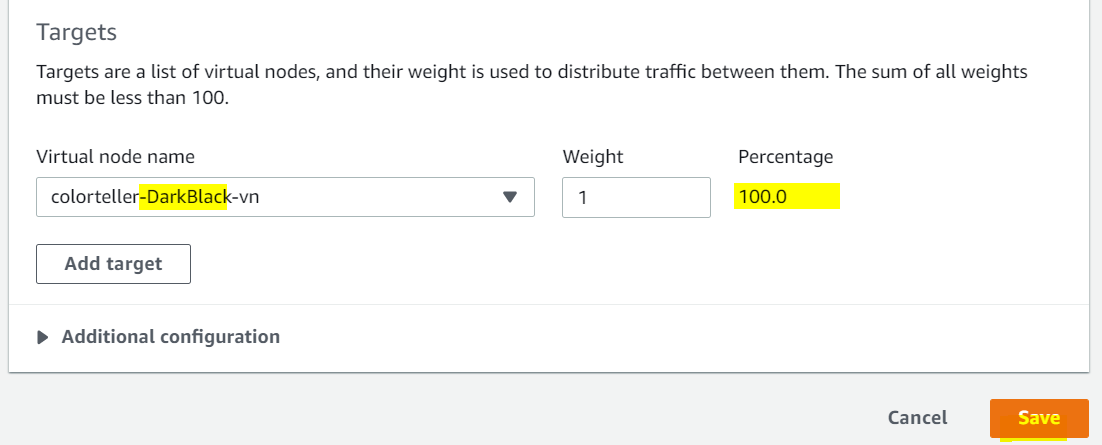


Hit the load balancer and refresh it several times.

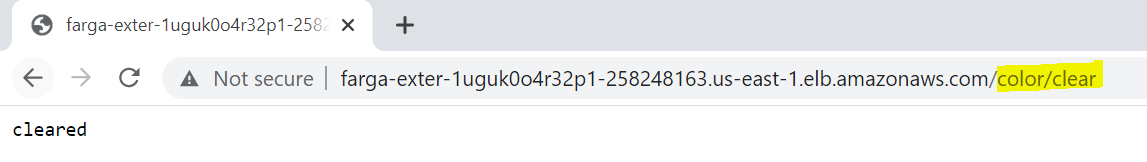
<http://ALB/color>



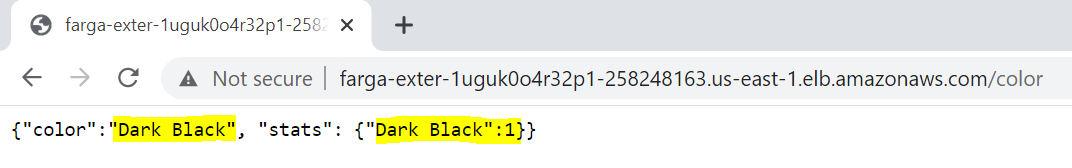
If your application is working fine route 100% traffic to dark black (version 2) and remove the previous/old version



Give

<http://ALB/color/clear> to clear old stats

Hit the /color again



Refresh multiple time and check that now only dark black/ version 2 while be available.

Now you can go ahead and delete the node from app mesh and service from ECS or you can keep it in case of rollback to old application.

Summary:

1. Manually created a virtual node in APPMESH.
2. Created new version of task definition.
3. Used task definition and created service in ECS fargate.