Tools used

1. Terraform is used to deploy the EKS cluster and the flask application
2. Github is used as the code repository.
3. Github registry is used to store the docker iamge
4. Docker is used to build the flask image
5. AWS is the cloud provider
6. Lens(https://k8slens.dev/) and kubectl CLI are used to access the EKS cluster for troubleshooting

EKS cluster creation

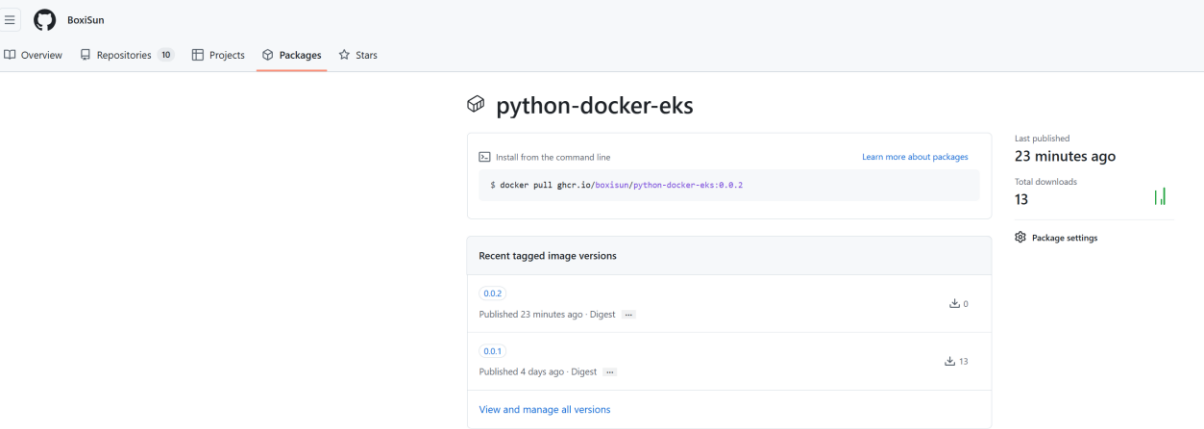
The EKS cluster was created using Terraform.

1. Due to the security concern, an IAM user with permission to assume an IAM role will be used to deploy the cluster.
2. VPC, subnets and other network resources were created via the “vpc” module
3. EKS cluster and load balancer controller roles were created via the “eks” module

```
Apply complete! Resources: 60 added, 0 changed, 0 destroyed.

Outputs:

cluster_name = "flask-eks-cluster"
oidc_provider = "oidc.eks.eu-west-1.amazonaws.com/id/64E6DA64567D6249146380B45930EDBF"
oidc_provider_arn = "arn:aws:iam::972156694227:oidc-provider/oidc.eks.eu-west-1.amazonaws.com/id/64E6DA64567D6249146380B45930EDBF"
vpc_arn = "arn:aws:ec2:eu-west-1:972156694227:vpc/vpc-06cbaceb1aa6c4eb1"
vpc_cidr_block = "10.111.160.0/20"
vpc_id = "vpc-06cbaceb1aa6c4eb1"
```

Python flask application build and deployment

The flask application was built and pushed to github as a package. A self-signed certificate was generated for the application server. (I roll back in my test with the AWS certificate as different certificates used by ALB and flask server will cause 502 Bad Gateway)



The flask application was deployed to the EKS cluster using Terraform.

```
Apply complete! Resources: 10 added, 0 changed, 0 destroyed.

Outputs:

ingress = "flask-ingress"
namespace = "flask"
rbac_role = "flask_rbac_role"
service = "flask-svc"
service_account = "flask-sa"
```

```
C:\Users\bsun\Projects\migrated\python-docker-eks\flask-app-deployment>kubectl get pods -n flask
NAME                                READY   STATUS    RESTARTS   AGE
flask-deployment-7675c66c78-5kqjh   1/1     Running   0          10m
flask-deployment-7675c66c78-clc4m   1/1     Running   0          10m
```

A TLS certificate was generated in AWS Certificate Manager to enable HTTPS connection.

A DNS CNAME was created in the company hosted zone in another AWS account via Terraform

```
[root@ip-10-111-163-19 ec2-user]# curl https://flask.neon.markets
{"message: "Hello, world!"}[root@ip-10-111-163-19 ec2-user]# ^C
[root@ip-10-111-163-19 ec2-user]# curl http://flask.neon.markets
<html>
<head><title>301 Moved Permanently</title></head>
<body>
<center><h1>301 Moved Permanently</h1></center>
</body>
</html>
[root@ip-10-111-163-19 ec2-user]# curl https://flask.neon.markets
{"message: "Hello, world!"}[root@ip-10-111-163-19 ec2-user]#
```

Metrics server and horizontal pod autoscaling

The metrics server was deployed via “kubectl”

```
C:\Users\bsun\Projects\migrated\python-docker-eks\flask-app-deployment>kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
serviceaccount/metrics-server created
clusterrole.rbac.authorization.k8s.io/system:aggregated-metrics-reader created
clusterrole.rbac.authorization.k8s.io/system:metrics-server created
rolebinding.rbac.authorization.k8s.io/metrics-server-auth-reader created
clusterrolebinding.rbac.authorization.k8s.io/metrics-server:system:auth-delegator created
clusterrolebinding.rbac.authorization.k8s.io/system:metrics-server created
service/metrics-server created
deployment.apps/metrics-server created
apiservice.apiregistration.k8s.io/v1beta1.metrics.k8s.io created

C:\Users\bsun\Projects\migrated\python-docker-eks\flask-app-deployment>kubectl get deployment metrics-server -n kube-system
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
metrics-server   1/1     1            1           91s

C:\Users\bsun\Projects\migrated\python-docker-eks\flask-app-deployment>kubectl top nodes
NAME                                       CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
ip-10-111-161-13.eu-west-1.compute.internal    30m          0%     627Mi           4%
ip-10-111-161-19.eu-west-1.compute.internal    29m          0%     657Mi           4%
ip-10-111-162-123.eu-west-1.compute.internal   34m          0%     627Mi           4%
ip-10-111-162-13.eu-west-1.compute.internal    26m          0%     630Mi           4%
```

```
C:\Users\bsun\Projects\migrated\python-docker-eks\flask-app-deployment>kubectl top pods -n flask
NAME                                   CPU(cores)   MEMORY(bytes)
flask-deployment-7675c66c78-5kqjh      2m           19Mi
flask-deployment-7675c66c78-clc4m      1m           18Mi
```

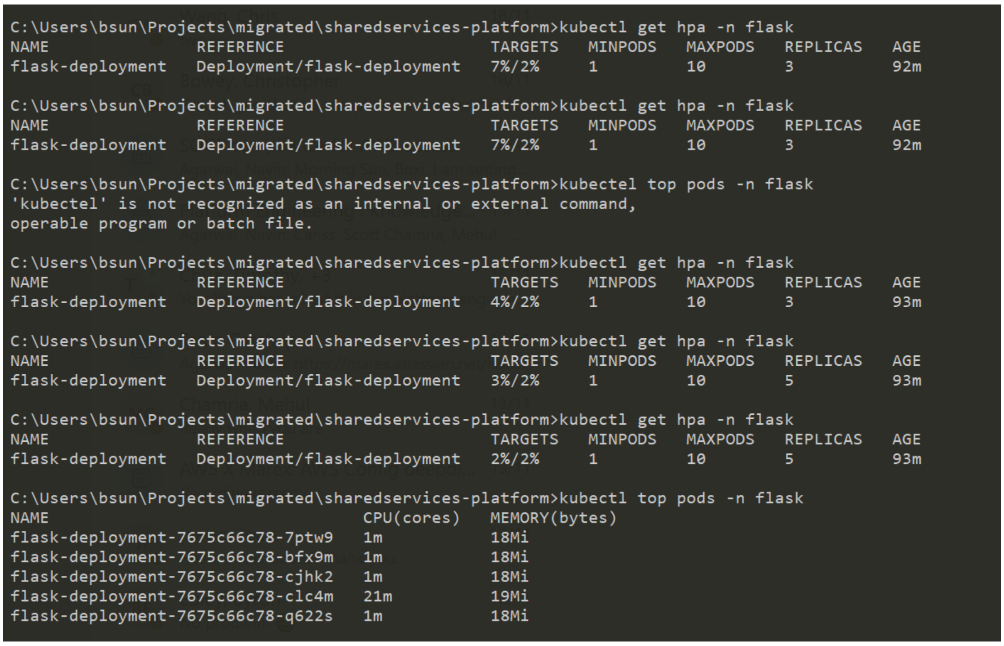
I run “while sleep 0.01; do wget -q -O- https://flask.neon.markets; done” in two sessions to trigger the horizontal autoscaling and monitor the resource usage

```
C:\Users\bsun\Projects>kubectl autoscale deployment flask-deployment -n flask --cpu-percent=2 --min=1 --max=10
horizontalpodautoscaler.autoscaling/flask-deployment autoscaled
```

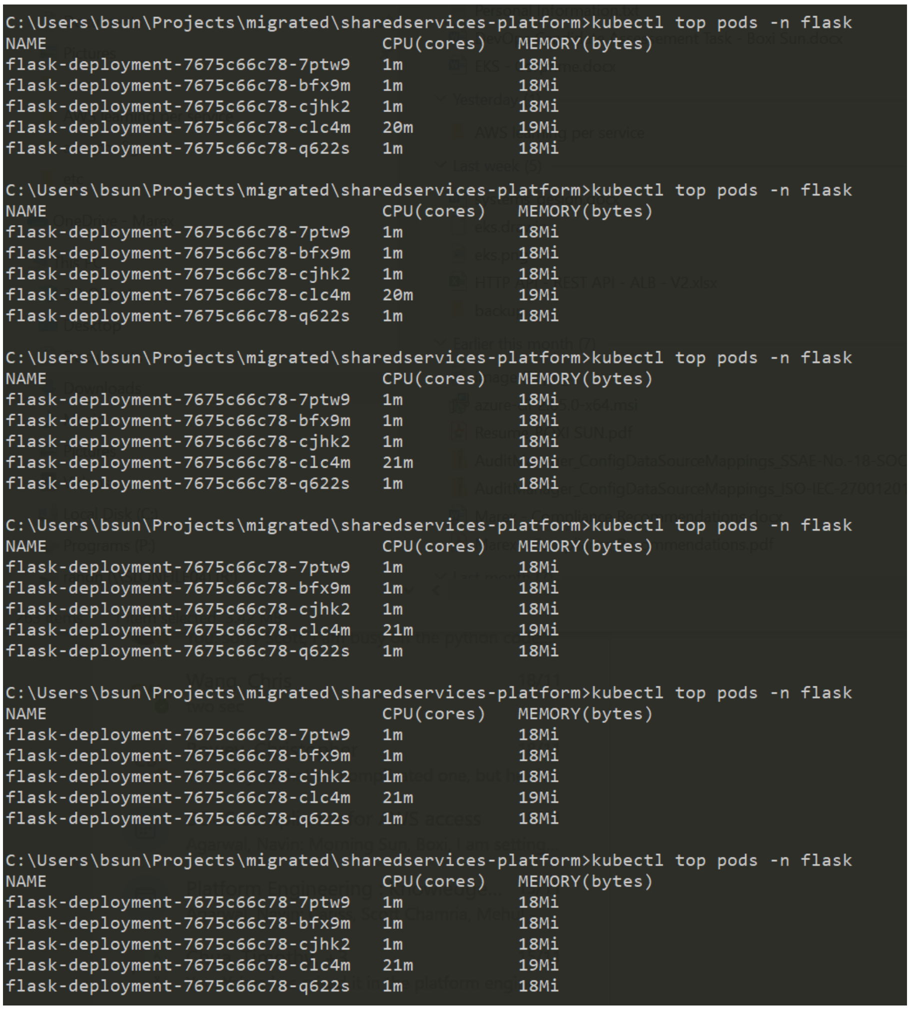
Before scaling up:

```
C:\Users\bsun\Projects\migrated\sharedservices-platform>kubectl get hpa -n flask
NAME               REFERENCE                     TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
flask-deployment   Deployment/flask-deployment   0%/2%     1         10        1          88m

C:\Users\bsun\Projects\migrated\sharedservices-platform>kubectl top pods -n flask
NAME                                   CPU(cores)   MEMORY(bytes)
flask-deployment-7675c66c78-clc4m      1m           19Mi
```

After scaling up:



```
C:\Users\bsun\Projects\migrated\sharedservices-platform>kubectl get hpa -n flask
NAME               REFERENCE                     TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
flask-deployment   Deployment/flask-deployment   7%/2%     1         10        3          92m

C:\Users\bsun\Projects\migrated\sharedservices-platform>kubectl get hpa -n flask
NAME               REFERENCE                     TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
flask-deployment   Deployment/flask-deployment   7%/2%     1         10        3          92m

C:\Users\bsun\Projects\migrated\sharedservices-platform>kubectel top pods -n flask
'kubectel' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\bsun\Projects\migrated\sharedservices-platform>kubectl get hpa -n flask
NAME               REFERENCE                     TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
flask-deployment   Deployment/flask-deployment   4%/2%     1         10        3          93m

C:\Users\bsun\Projects\migrated\sharedservices-platform>kubectl get hpa -n flask
NAME               REFERENCE                     TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
flask-deployment   Deployment/flask-deployment   3%/2%     1         10        5          93m

C:\Users\bsun\Projects\migrated\sharedservices-platform>kubectl get hpa -n flask
NAME               REFERENCE                     TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
flask-deployment   Deployment/flask-deployment   2%/2%     1         10        5          93m

C:\Users\bsun\Projects\migrated\sharedservices-platform>kubectl top pods -n flask
NAME                                CPU(cores)   MEMORY(bytes)
flask-deployment-7675c66c78-7ptw9   1m           18Mi
flask-deployment-7675c66c78-bfx9m   1m           18Mi
flask-deployment-7675c66c78-cjhk2   1m           18Mi
flask-deployment-7675c66c78-clc4m   21m          19Mi
flask-deployment-7675c66c78-q622s   1m           18Mi
```



```
C:\Users\bsun\Projects\migrated\sharedservices-platform>kubectl top pods -n flask
NAME                                CPU(cores)   MEMORY(bytes)
flask-deployment-7675c66c78-7ptw9   1m           18Mi
flask-deployment-7675c66c78-bfx9m   1m           18Mi
flask-deployment-7675c66c78-cjhk2   1m           18Mi
flask-deployment-7675c66c78-clc4m   20m          19Mi
flask-deployment-7675c66c78-q622s   1m           18Mi

C:\Users\bsun\Projects\migrated\sharedservices-platform>kubectl top pods -n flask
NAME                                CPU(cores)   MEMORY(bytes)
flask-deployment-7675c66c78-7ptw9   1m           18Mi
flask-deployment-7675c66c78-bfx9m   1m           18Mi
flask-deployment-7675c66c78-cjhk2   1m           18Mi
flask-deployment-7675c66c78-clc4m   20m          19Mi
flask-deployment-7675c66c78-q622s   1m           18Mi

C:\Users\bsun\Projects\migrated\sharedservices-platform>kubectl top pods -n flask
NAME                                CPU(cores)   MEMORY(bytes)
flask-deployment-7675c66c78-7ptw9   1m           18Mi
flask-deployment-7675c66c78-bfx9m   1m           18Mi
flask-deployment-7675c66c78-cjhk2   1m           18Mi
flask-deployment-7675c66c78-clc4m   21m          19Mi
flask-deployment-7675c66c78-q622s   1m           18Mi

C:\Users\bsun\Projects\migrated\sharedservices-platform>kubectl top pods -n flask
NAME                                CPU(cores)   MEMORY(bytes)
flask-deployment-7675c66c78-7ptw9   1m           18Mi
flask-deployment-7675c66c78-bfx9m   1m           18Mi
flask-deployment-7675c66c78-cjhk2   1m           18Mi
flask-deployment-7675c66c78-clc4m   21m          19Mi
flask-deployment-7675c66c78-q622s   1m           18Mi

C:\Users\bsun\Projects\migrated\sharedservices-platform>kubectl top pods -n flask
NAME                                CPU(cores)   MEMORY(bytes)
flask-deployment-7675c66c78-7ptw9   1m           18Mi
flask-deployment-7675c66c78-bfx9m   1m           18Mi
flask-deployment-7675c66c78-cjhk2   1m           18Mi
flask-deployment-7675c66c78-clc4m   21m          19Mi
flask-deployment-7675c66c78-q622s   1m           18Mi

C:\Users\bsun\Projects\migrated\sharedservices-platform>kubectl top pods -n flask
NAME                                CPU(cores)   MEMORY(bytes)
flask-deployment-7675c66c78-7ptw9   1m           18Mi
flask-deployment-7675c66c78-bfx9m   1m           18Mi
flask-deployment-7675c66c78-cjhk2   1m           18Mi
flask-deployment-7675c66c78-clc4m   21m          19Mi
flask-deployment-7675c66c78-q622s   1m           18Mi
```

With an php-apache server however, I could see that the workload are evenly distributed

kubectl apply -f https://k8s.io/examples/application/php-apache.yaml

kubectl run -i --tty load-generator --rm --image=busybox --restart=Never -- /bin/sh -c "while sleep 0.01; do wget -q -O- http://php-apache; done" (this is to trigger the scaling)

```
C:\Users\bsun>kubectl top pods
NAME                          CPU(cores)   MEMORY(bytes)
load-generator                17m          0Mi
php-apache-598b474864-t5dv2   213m         11Mi

C:\Users\bsun>kubectl get hpa php-apache
NAME         REFERENCE                TARGETS    MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache    106%/50%   1         10        1          60s

C:\Users\bsun>kubectl top pods
NAME                          CPU(cores)   MEMORY(bytes)
load-generator                17m          0Mi
php-apache-598b474864-2ssm6   225m         11Mi
php-apache-598b474864-5q4gd   184m         11Mi
php-apache-598b474864-t5dv2   314m         11Mi
```

```
C:\Users\bsun>kubectl top pods
NAME                          CPU(cores)   MEMORY(bytes)
load-generator                19m          0Mi
php-apache-598b474864-2ssm6   182m         11Mi
php-apache-598b474864-5q4gd   191m         11Mi
php-apache-598b474864-8wrfm   118m         11Mi
php-apache-598b474864-fd525   116m         11Mi
php-apache-598b474864-t5dv2   314m         11Mi

C:\Users\bsun>kubectl top pods
NAME                          CPU(cores)   MEMORY(bytes)
load-generator                19m          0Mi
php-apache-598b474864-2ssm6   182m         11Mi
php-apache-598b474864-5q4gd   191m         11Mi
php-apache-598b474864-8wrfm   118m         11Mi
php-apache-598b474864-fd525   116m         11Mi
php-apache-598b474864-t5dv2   314m         11Mi

C:\Users\bsun>kubectl top pods
NAME                          CPU(cores)   MEMORY(bytes)
load-generator                19m          0Mi
php-apache-598b474864-2ssm6   182m         11Mi
php-apache-598b474864-5q4gd   191m         11Mi
php-apache-598b474864-8wrfm   118m         11Mi
php-apache-598b474864-fd525   116m         11Mi
php-apache-598b474864-t5dv2   314m         11Mi
```

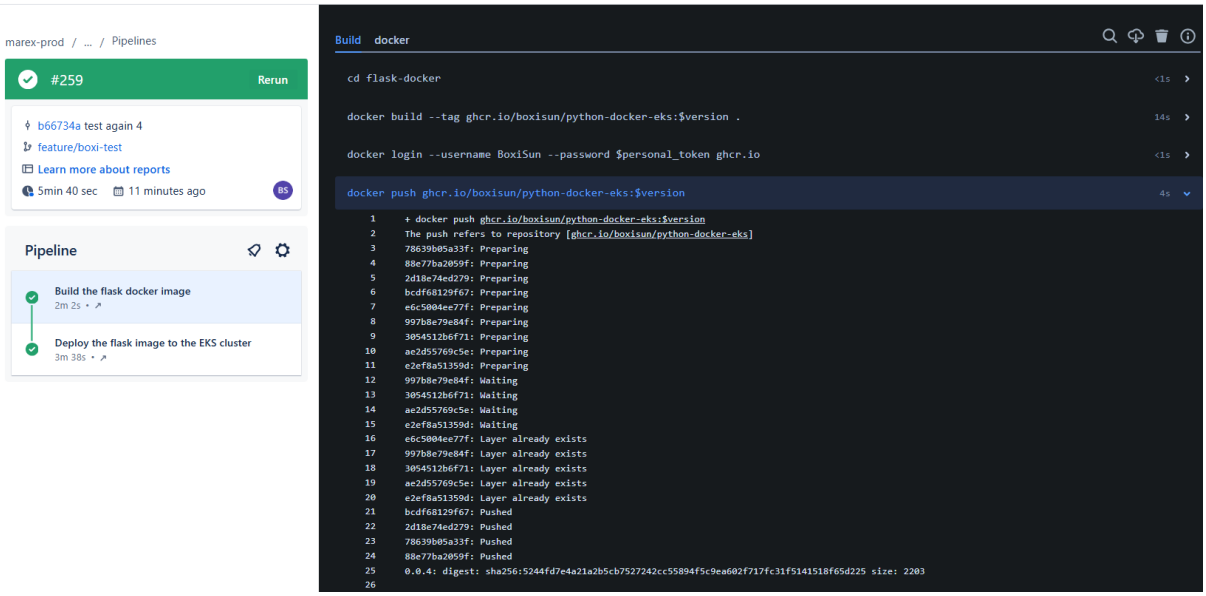
CICD with Bitbucket pipeline

Note: The Bitbucket pipeline requires the Bitbucket environment

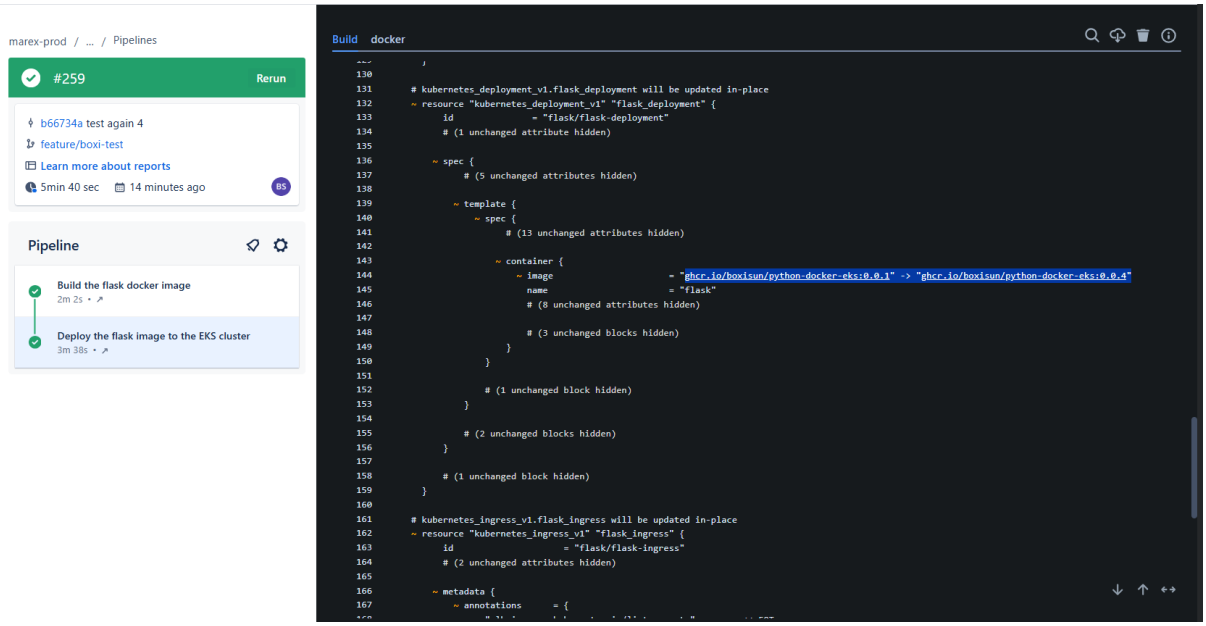
The build step is triggered when there is a file change in the folder below:

```
filter:
  includes:
    - "flask-app-deployment/*"
```

Bug: Pulling the image of the same version will not overwrite the existing one.



The deploy step:



After deployment:

```
[root@ip-10-111-163-19 ec2-user]# curl https://flask.neon.markets
{"message: "Hello, world!"}[root@ip-10-111-163-19 ec2-user]# curl https://flask.neon.markets
{"message: "Hello, world, test by Boxi!"}[root@ip-10-111-163-19 ec2-user]#
```

Issue faced and solutions applied

1. Can't connect to the API group after creating the EKS cluster

Error:

```
C:\Users\bsun\Projects>kubectl get svc
E1124 17:59:53.250966   25816 memcache.go:265] couldn't get current server API group list: Get "https://90F2F70114A39F575C0197231A604818.gr7.eu-west-1.eks.amazonaws.com/api?timeout=32s": dial tcp 10.111.162.76:443: i/o timeout
E1124 18:00:23.260130   25816 memcache.go:265] couldn't get current server API group list: Get "https://90F2F70114A39F575C0197231A604818.gr7.eu-west-1.eks.amazonaws.com/api?timeout=32s": dial tcp 10.111.162.76:443: i/o timeout
```

Troubleshooting:

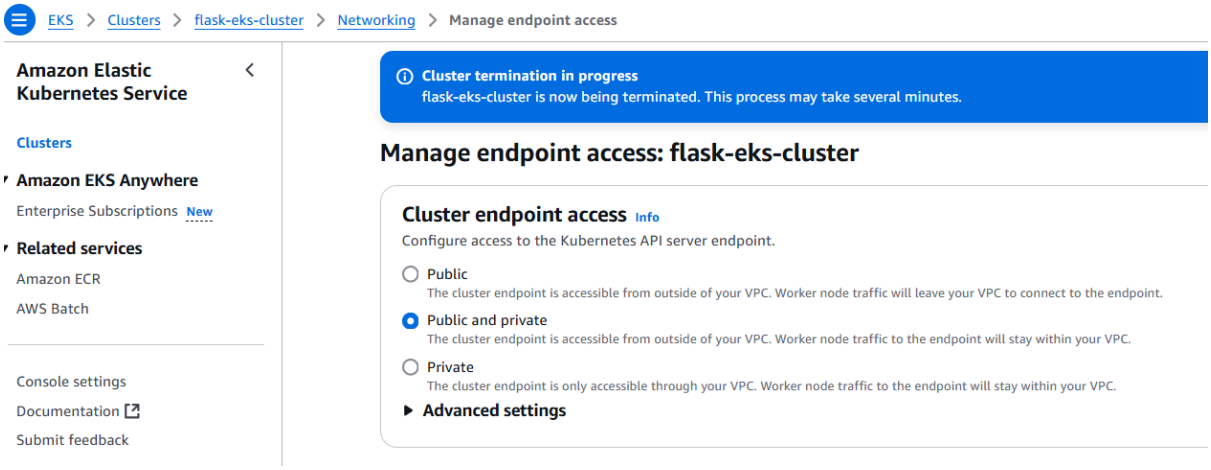
- Check the firewall (AWS security groups) and make sure they are not blocking my IP.

- Since the EKS cluster is deployed in private subnets, the endpoint access is set to "private" meaning that it could only be accessible from within the VPC.

Solutions:

Create an EC2 jumpbox in the same private subnet of the EKS cluster and confirmed that I could access the cluster

Per the document below, change the endpoint access from "private" to "Public and private".



2. Load balancer is not created and terraform creating ingress timeout

Error:

```
Error: Load Balancer is not ready yet

  with kubernetes_ingress_v1.sample_application_ingress,
  on sample_app.tf line 175, in resource "kubernetes_ingress_v1" "sample_application_ingress":
 175: resource "kubernetes_ingress_v1" "sample_application_ingress" {
```

Troubleshooting:

kubectl describe ingress sample-application-ingress -n sample-application

```
Message
-------
Failed build model due to couldn't auto-discover subnets: unable to resolve at least one subnet (0 match VPC and tags: [kubernetes.io/role/elb])
```

Solutions:

The error indicates that the alb couldn't find any subnet match. Per the document below, I choose to specify subnets as annotations

https://docs.aws.amazon.com/eks/latest/userguide/alb-ingress.html

- Your public and private subnets must meet the following requirements. This is unless you explicitly specify subnet IDs as an annotation on a service or ingress object. Assume that you provision load balancers by explicitly specifying subnet IDs as an annotation on a service or ingress object. In this situation, Kubernetes and the AWS load balancer controller use those subnets directly to create the load balancer and the following tags aren't required.
  - **Private subnets** – Must be tagged in the following format. This is so that Kubernetes and the AWS load balancer controller know that the subnets can be used for internal load balancers. If you use `eksctl` or an Amazon EKS AWS CloudFormation template to create your VPC after March 26, 2020, the subnets are tagged appropriately when created. For more information about the Amazon EKS AWS CloudFormation VPC templates, see Create an Amazon VPC for your Amazon EKS cluster.
    - **Key** – `kubernetes.io/role/internal-elb`
    - **Value** – `1`
  - **Public subnets** – Must be tagged in the following format. This is so that Kubernetes knows to use only the subnets that were specified for external load balancers. This way, Kubernetes doesn't choose a public subnet in each Availability Zone (lexicographically based on their subnet ID). If you use `eksctl` or an Amazon EKS AWS CloudFormation template to create your VPC after March 26, 2020, the subnets are tagged appropriately when created. For more information about the Amazon EKS AWS CloudFormation VPC templates, see Create an Amazon VPC for your Amazon EKS cluster.
    - **Key** – `kubernetes.io/role/elb`
    - **Value** – `1`

  https://kubernetes-sigs.github.io/aws-load-balancer-controller/v2.2/guide/ingress/annotations/

- `alb.ingress.kubernetes.io/subnets` specifies the Availability Zone that ALB will route traffic to. See Load Balancer subnets for more details.

> You must specify at least two subnets in different AZ. both subnetID or subnetName(Name tag on subnets) can be used.

> **Tip**
>
> You can enable subnet auto discovery to avoid specify this annotation on every Ingress. See Subnet Discovery for instructions.

> **Example**

```
alb.ingress.kubernetes.io/subnets: subnet-xxxx, mySubnet
```

3. EKS failed to pull the docker image from github

Error:

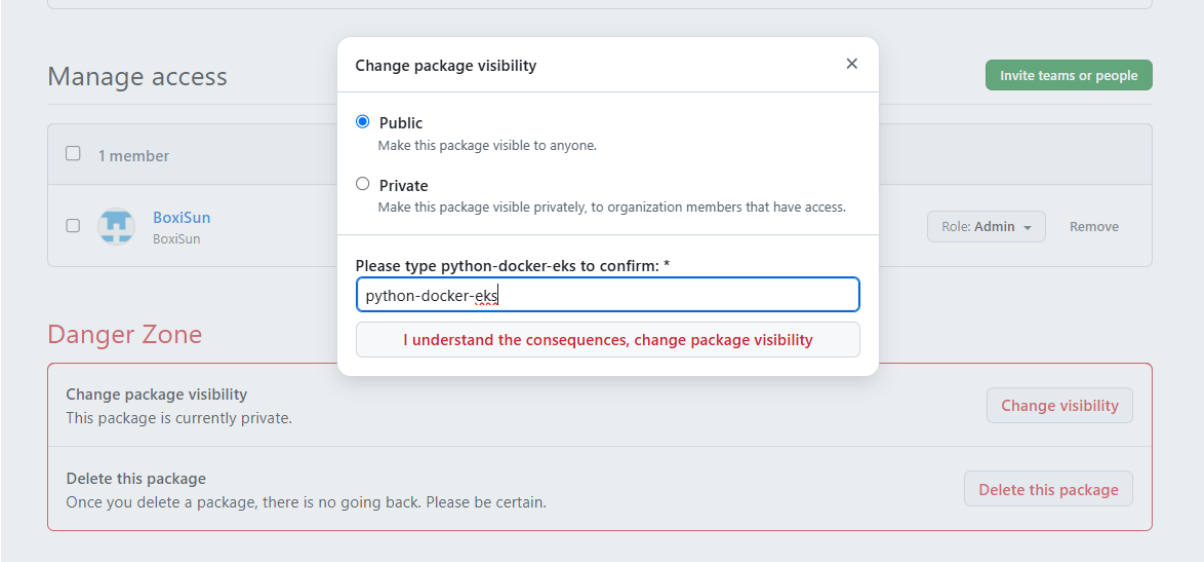
```
Pulling image "ghcr.io/boxisun/python-docker-eks:0.0.1"

Source          kubelet ip-10-215-144-9.eu-west-1.compute.internal
Count           4
Sub-object      spec.containers{python-flask}
Last seen       2024-11-22T16:44:36Z

Failed to pull image "ghcr.io/boxisun/python-docker-eks:0.0.1": failed to pull and unpack image "ghcr.io/boxisun/python-docker-eks:0.0.1": failed to resolve reference "ghcr.io/boxisun/python-docker-eks:0.0.1": failed to authorize: failed to fetch anonymous token: unexpected status from GET request to https://ghcr.io/token?scope=repository%3Aboxisun%2Fpython-docker-eks%3Apull&service=ghcr.io: 401 Unauthorized

Source          kubelet ip-10-215-144-9.eu-west-1.compute.internal
```

Troubleshooting:

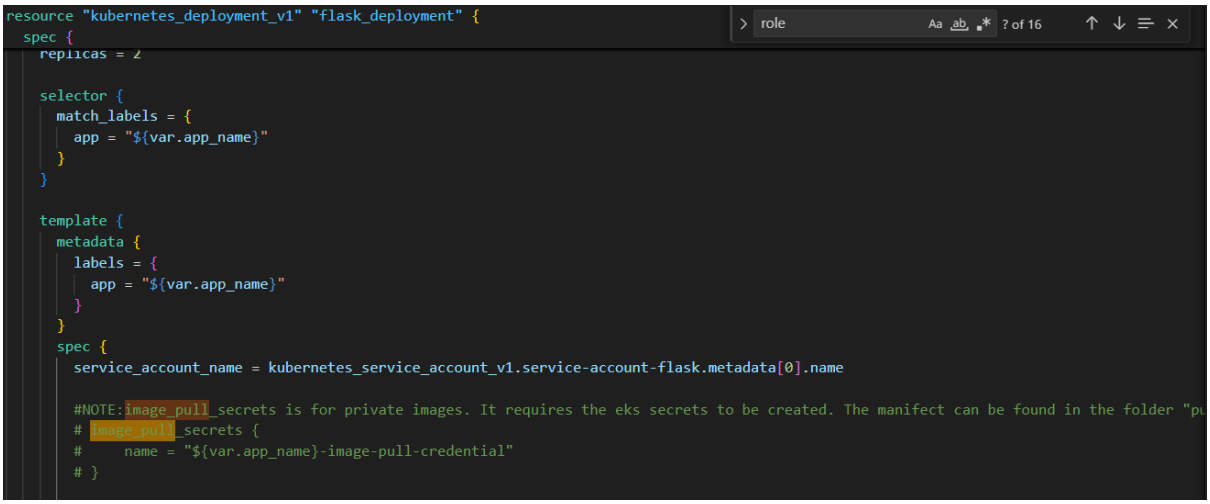
The error indicates that EKS failed to fetch the token and therefore not authorized to pull the image

Solution:

1. Change the flask image to be public:



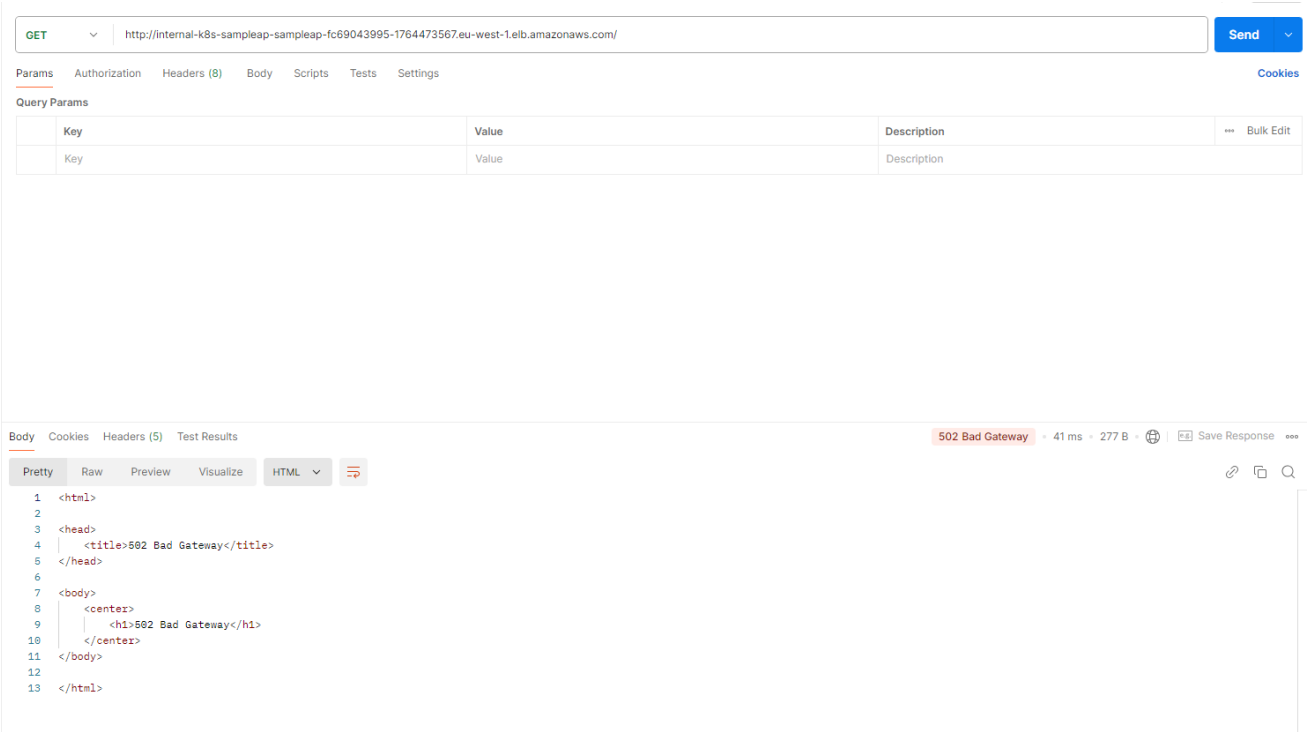
2. Create a personal access token and follow the steps in the document below to generate a dockerconfigjson secret. Apply the secret and add the "image_pull_secret" to the deployment terraform code:



```
resource "kubernetes_deployment_v1" "flask_deployment" {
  spec {
    replicas = 2

    selector {
      match_labels = {
        app = "${var.app_name}"
      }
    }

    template {
      metadata {
        labels = {
          app = "${var.app_name}"
        }
      }
      spec {
        service_account_name = kubernetes_service_account_v1.service-account-flask.metadata[0].name

        #NOTE:image_pull_secrets is for private images. It requires the eks secrets to be created. The manifect can be found in the folder "pu
        # image_pull_secrets {
        #     name = "${var.app_name}-image-pull-credential"
        # }
```

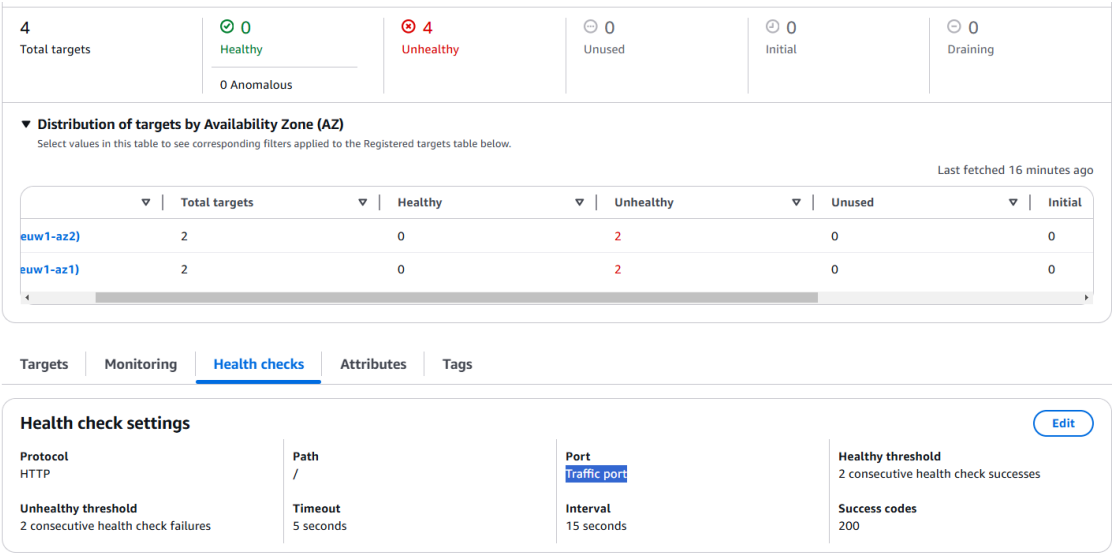
4. Failed with “502 Bad Gateway” when testing the endpoint:

Error:



Troubleshooting:

Check the status of the load balancer target group and find out that all instances are “unhealthy”. This means the health check failed.



In this case, the “Traffic port” of flask application is 5000. The deployment log confirms that:

```
kubectl logs -n sample-application sample-application-deployment-79c75f45b8-9rqv5

 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://100.64.0.241:5000
Press CTRL+C to quit
```

Solutions:

I found from the terraform code that the container port and target_port were both 80. I updated them to 5000 and the issue was fixed

```
spec {
  service_account_name = kubernetes_service_account.service-account-sample-app.metadata[0].name
  container {
    image = "ghcr.io/boxisun/python-docker-eks:0.0.1"
    name  = "python-flask"

    resources {
      limits = {
        cpu    = "0.5"
        memory = "512Mi"
      }
      requests = {
        cpu    = "250m"
        memory = "50Mi"
      }
    }

    port {
      container_port = 5000
    }
```

```
resource "kubernetes_service_v1" "sample_application_svc" {
  metadata {
    name      = "sample-application-svc"
    namespace = kubernetes_namespace.sample-application-namespace.metadata[0].name
  }
  spec {
    selector = {
      app = "nginx"
    }
    session_affinity = "ClientIP"
    port {
      port        = 80
      target_port = 5000
    }

    type = "NodePort"
  }
}
```

5. Pods reboot over and over again

Error:

```
Liveness probe failed: Get "http://10.111.161.34:80/": dial tcp 10.111.161.34:80: connect:
connection refused
```

Troubleshooting:

Similar to Issue 4, the port 80 in the error got my attention. Plus, the “initial_delay_seconds” and “period_seconds” setting are way too short.

Solutions:

I updated the terraform code and apply the change

```
~ liveness_probe {
  ~ initial_delay_seconds = 3 -> 10
  ~ period_seconds       = 3 -> 10
    # (3 unchanged attributes hidden)

  ~ http_get {
    ~ port   = "80" -> "5000"
      # (2 unchanged attributes hidden)
```