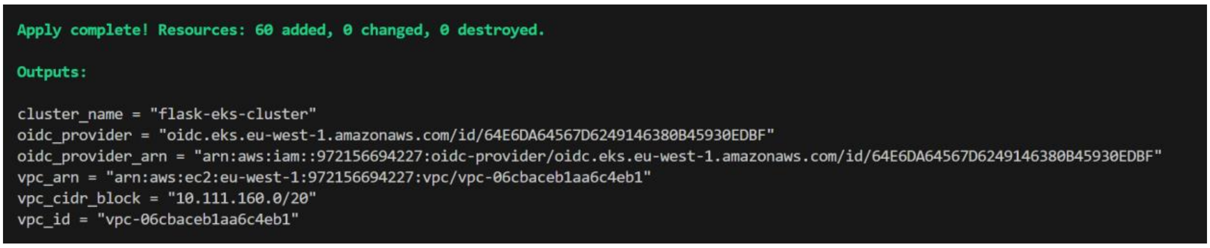
Tools used

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 image
4. Docker is used to build the flask image
5. AWS is the cloud provider
6. "aws-vault" tool (https://github.com/99designs/aws-vault) to access the EKS cluster via CLI
7. 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.

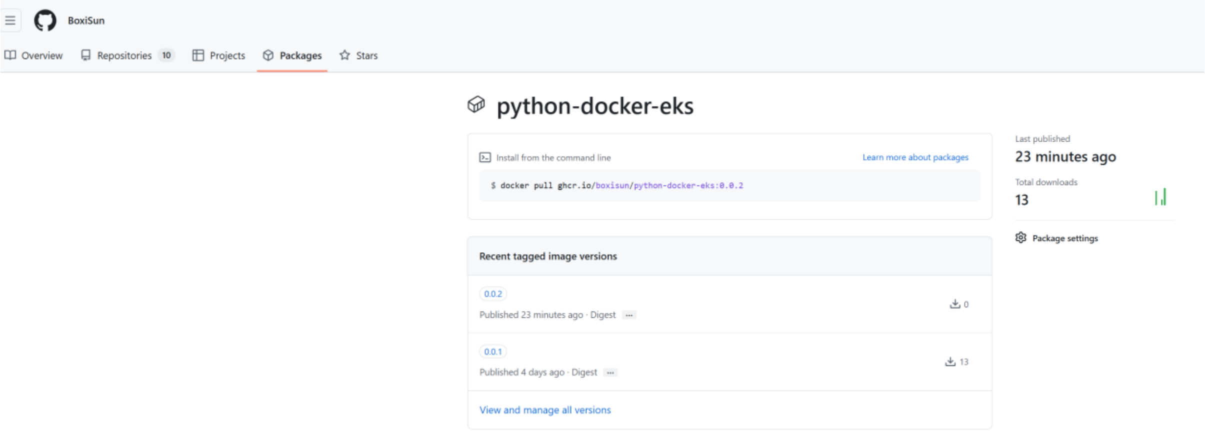
1. Due to the security concern, an IAM user with permission to assume an IAM role will be used to deploy the cluster.
2. VPC, subnets and other network resources were created via the "vpc" module
3. EKS cluster and load balancer controller roles were created via the "eks" module



```
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)



python-docker-eks

Install from the command line

Learn more about packages

```
$ docker pull ghcr.io/boxisun/python-docker-eks:0.0.2
```

Last published
23 minutes ago

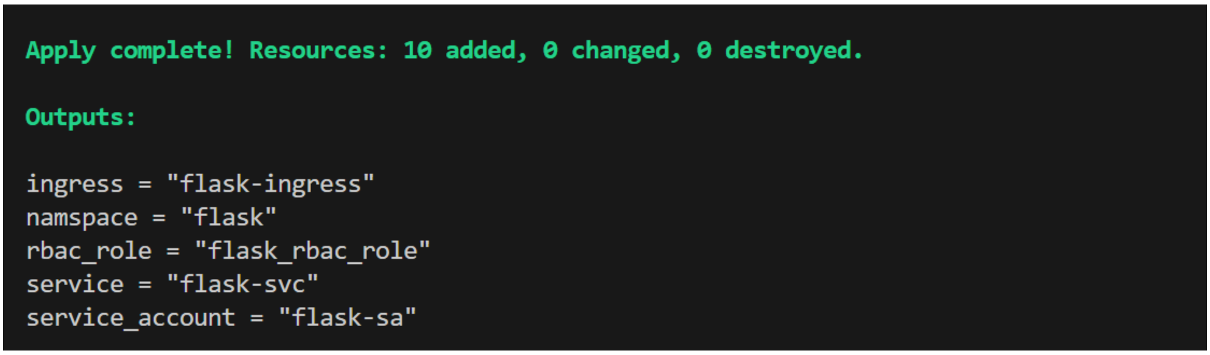
Total downloads
13

Recent tagged image versions

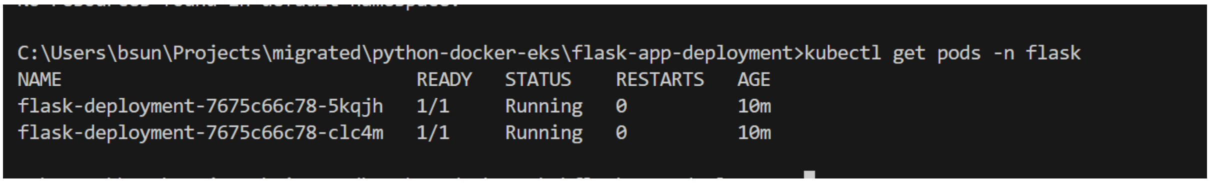
0.0.2
Published 23 minutes ago · Digest ··· 0

0.0.1
Published 4 days ago · Digest ··· 13

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



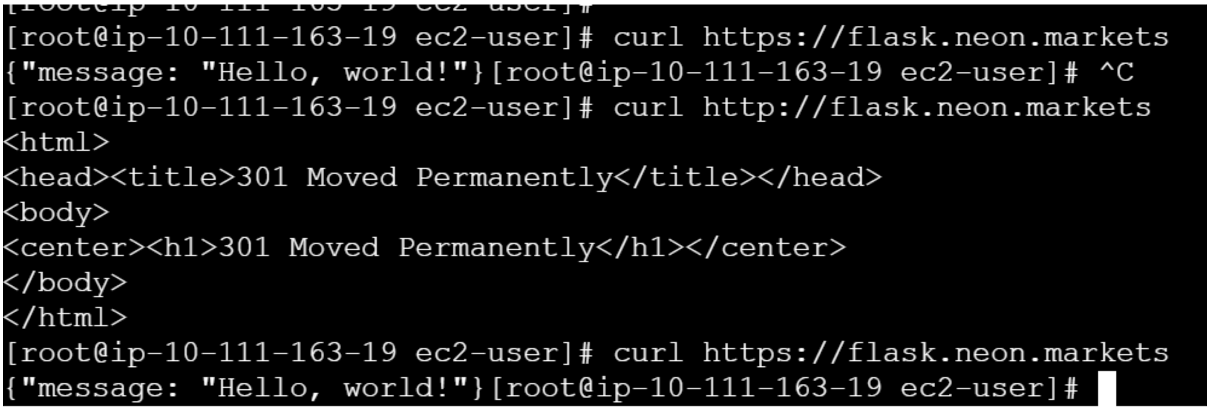
```
Apply complete! Resources: 10 added, 0 changed, 0 destroyed.

Outputs:

ingress = "flask-ingress"
namspace = "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.

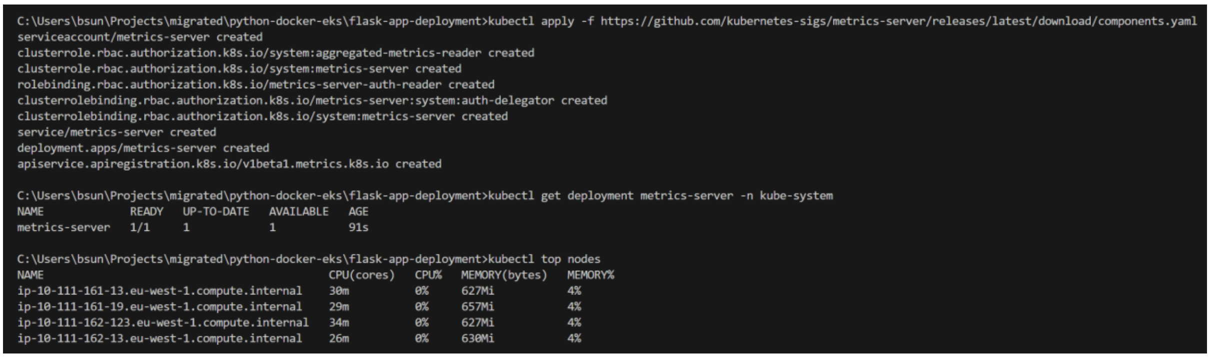
A DNS CNAME was created in the company hosted zone in another AWS account via Terraform



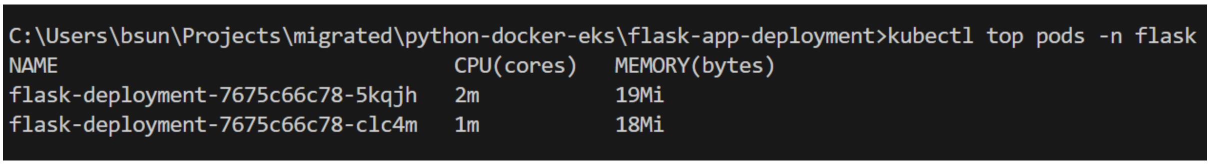
```
[root@ip-10-111-163-19 ec2-user]#
[root@ip-10-111-163-19 ec2-user]# curl https://flask.neon.markets
{"message: "Hello, world!"}[root@ip-10-111-163-19 ec2-user]# ^C
[root@ip-10-111-163-19 ec2-user]# curl http://flask.neon.markets
<html>
<head><title>301 Moved Permanently</title></head>
<body>
<center>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

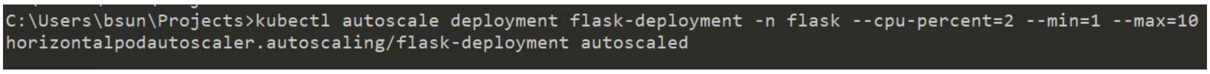


```
C:\Users\bsun\Projects\migrated\python-docker-eks\flask-app-deployment>kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
serviceaccount/metrics-server created
clusterrole.rbac.authorization.k8s.io/system:aggregated-metrics-reader created
clusterrole.rbac.authorization.k8s.io/system:metrics-server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



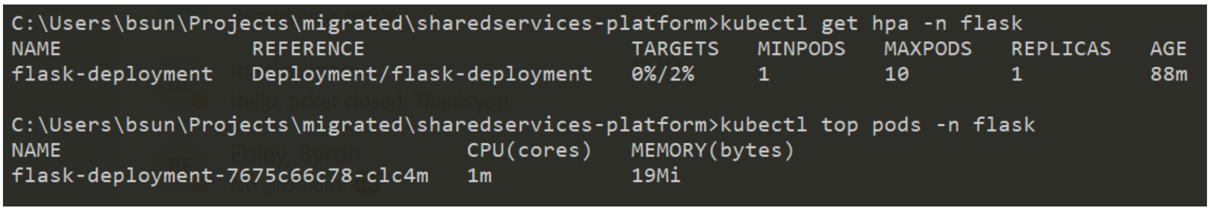
```
C:\Users\bsun\Projects\migrated\python-docker-eks\flask-app-deployment>kubectl top pods -n flask
NAME                                CPU(cores)   MEMORY(bytes)
flask-deployment-7675c66c78-5kqjh   2m           19Mi
flask-deployment-7675c66c78-clc4m   1m           18Mi
```

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



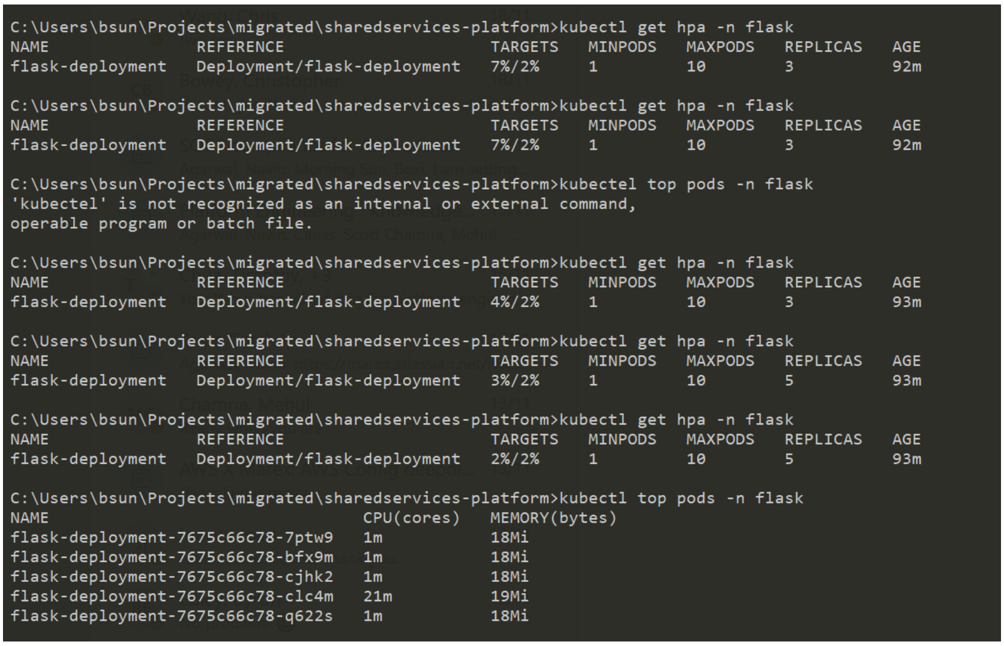
```
C:\Users\bsun\Projects>kubectl autoscale deployment flask-deployment -n flask --cpu-percent=2 --min=1 --max=10
horizontalpodautoscaler.autoscaling/flask-deployment autoscaled
```

Before scaling up:

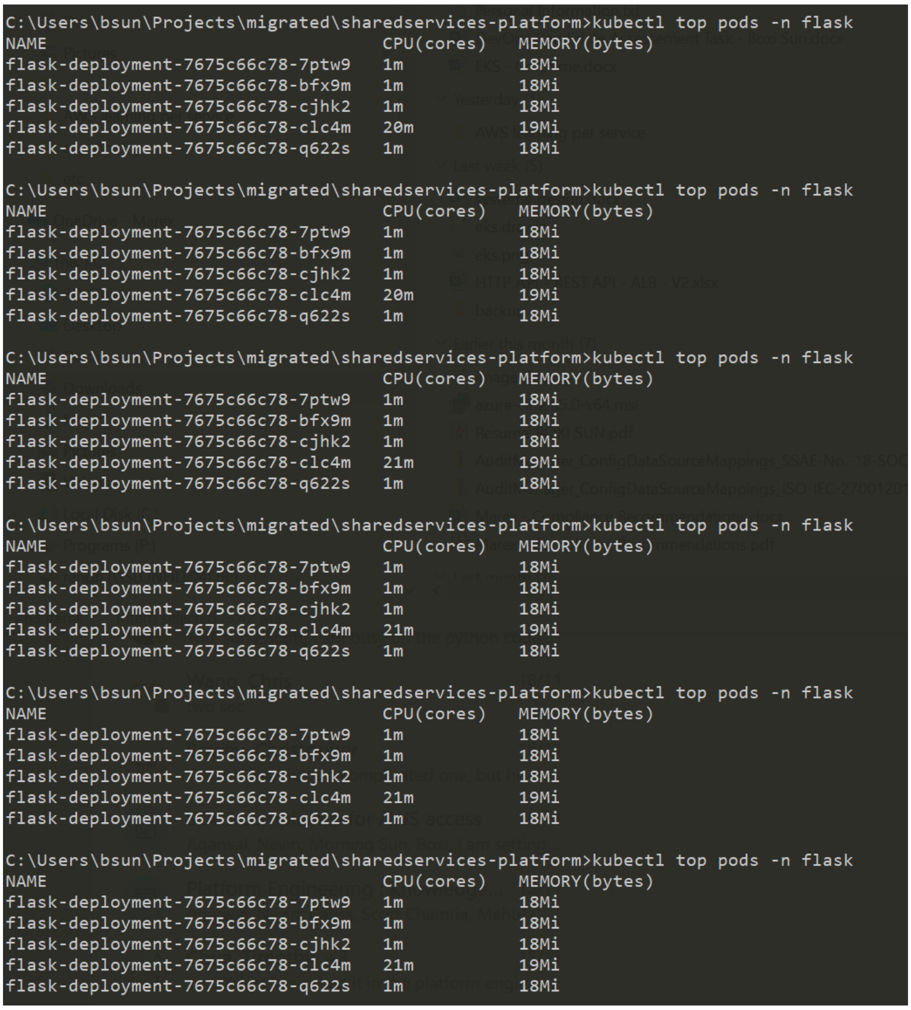


```
C:\Users\bsun\Projects\migrated\sharedservices-platform>kubectl get hpa -n flask
NAME               REFERENCE                     TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
flask-deployment   Deployment/flask-deployment   0%/2%     1         10        1          88m

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
```

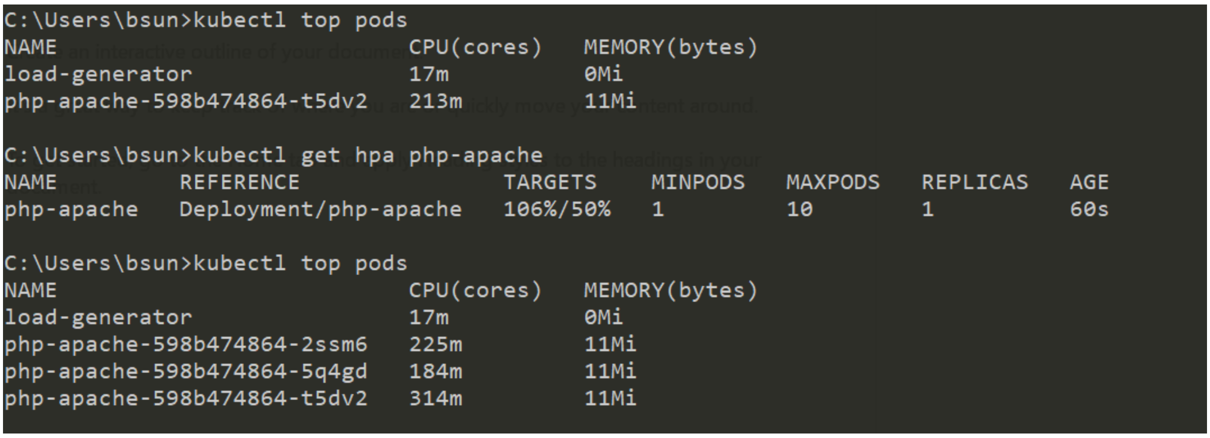


```
C:\Users\bsun\Projects\migrated\sharedservices-platform>kubectl top pods -n flask
NAME                                CPU(cores)   MEMORY(bytes)
flask-deployment-7675c66c78-7ptw9   1m           18Mi
flask-deployment-7675c66c78-bfx9m   1m           18Mi
flask-deployment-7675c66c78-cjhk2   1m     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

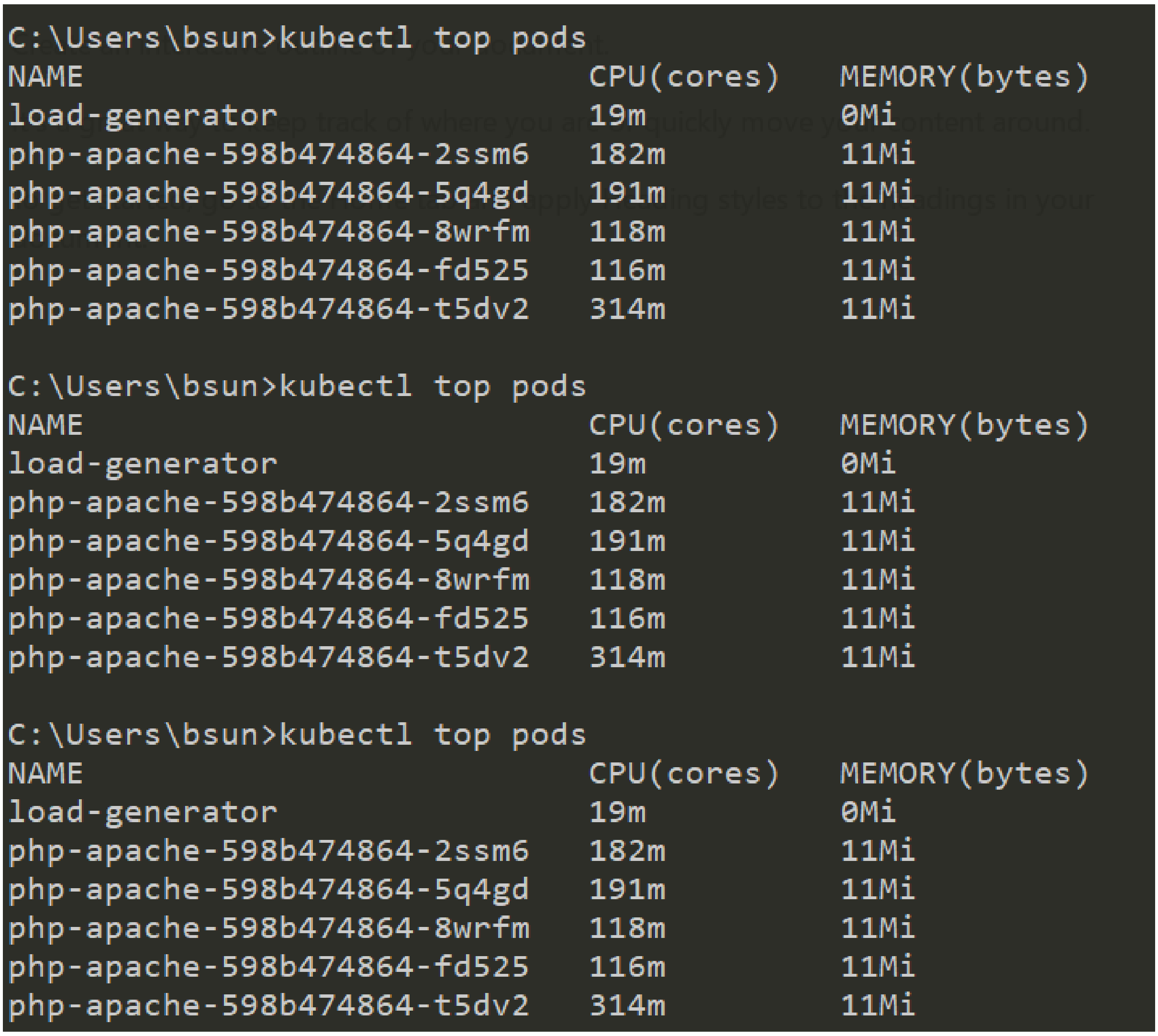
With an php-apache server however, I could see that the workload are evenly distributed

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
```

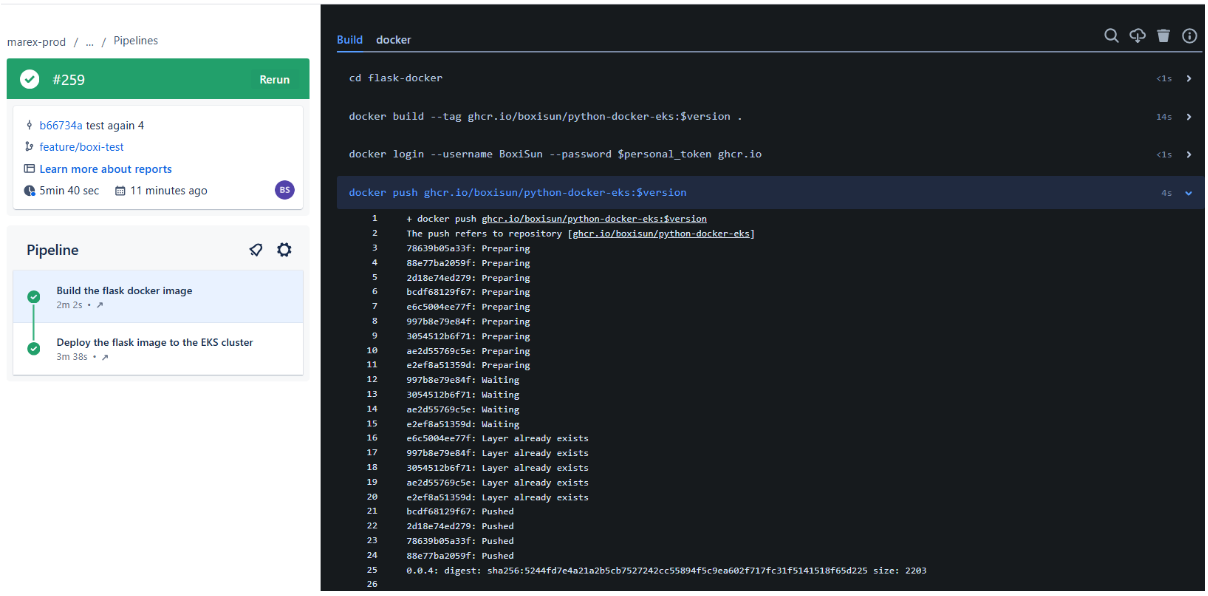
CICD with Bitbucket pipeline

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.



marex-prod / ... / Pipelines

#259 Rerun

b66734a test again 4
feature/boxi-test
Learn more about reports
5min 40 sec 11 minutes ago

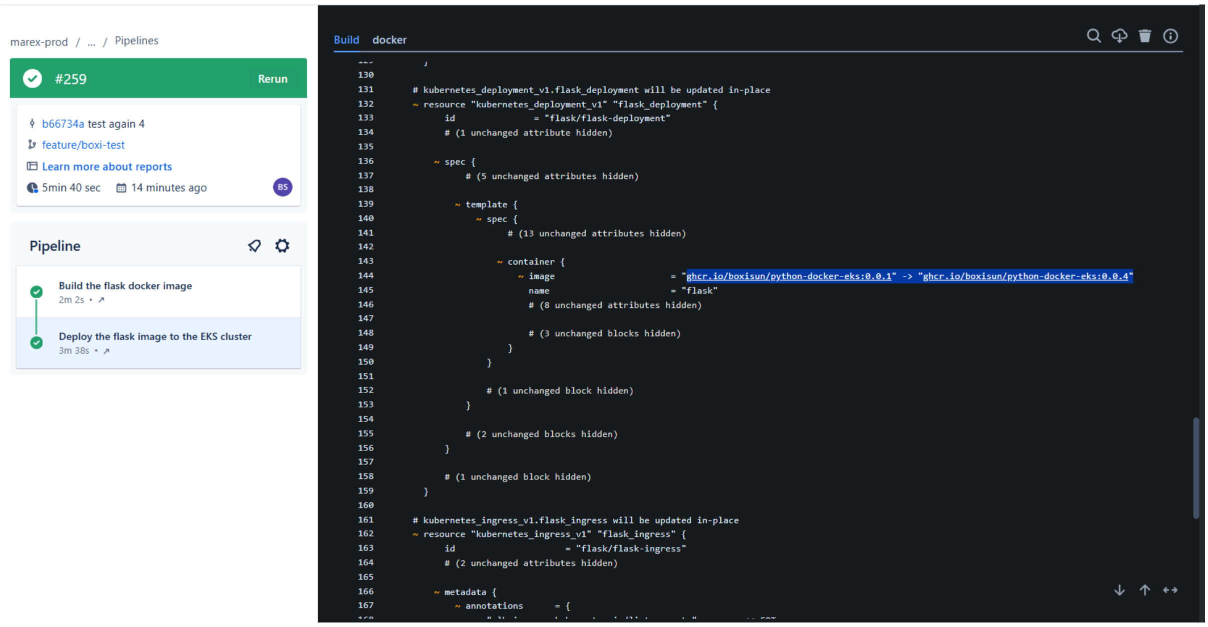
Pipeline

- Build the flask docker image — 2m 2s
- Deploy the flask image to the EKS cluster — 3m 38s

Build docker

```
cd flask-docker                                                  <1s
docker build --tag ghcr.io/boxisun/python-docker-eks:$version .  14s
docker login --username BoxiSun --password $personal_token ghcr.io  <1s
docker push ghcr.io/boxisun/python-docker-eks:$version           4s

+ docker push ghcr.io/boxisun/python-docker-eks:$version
The push refers to repository [ghcr.io/boxisun/python-docker-eks]
78639b05a33f: Preparing
88e77ba2059f: Preparing
2d18e74ed279: Preparing
bcdf68129f67: Preparing
e6c5004ee77f: Preparing
997b8e79e84f: Preparing
3054512b6f71: Preparing
ae2d55769c5e: Preparing
e2ef8a51359d: Preparing
997b8e79e84f: Waiting
3054512b6f71: Waiting
ae2d55769c5e: Waiting
e2ef8a51359d: Waiting
e6c5004ee77f: Layer already exists
997b8e79e84f: Layer already exists
3054512b6f71: Layer already exists
ae2d55769c5e: Layer already exists
e2ef8a51359d: Layer already exists
bcdf68129f67: Pushed
2d18e74ed279: Pushed
78639b05a33f: Pushed
88e77ba2059f: Pushed
0.0.4: digest: sha256:5244fd7e4a21a2b5cb7527242cc55894f5c9ea602f717fc31f5141518f65d225 size: 2203
```

The deploy step:



marex-prod / ... / Pipelines

#259 Rerun

b66734a test again 4
feature/boxi-test
Learn more about reports
5min 40 sec 14 minutes ago

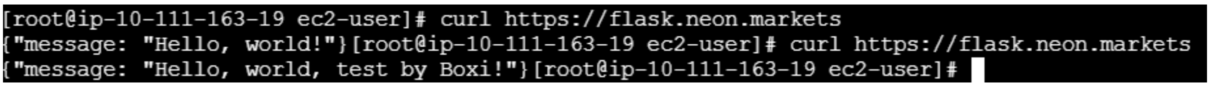
Pipeline

- Build the flask docker image — 2m 2s
- Deploy the flask image to the EKS cluster — 3m 38s

Build docker

```
  # kubernetes_deployment_v1.flask_deployment will be updated in-place
  ~ resource "kubernetes_deployment_v1" "flask_deployment" {
        id               = "flask/flask-deployment"
        # (1 unchanged attribute hidden)

      ~ spec {
            # (5 unchanged attributes hidden)

          ~ template {
              ~ spec {
                    # (13 unchanged attributes hidden)

                  ~ container {
                      ~ image                      = "ghcr.io/boxisun/python-docker-eks:0.0.1" -> "ghcr.io/boxisun/python-docker-eks:0.0.4"
                        name                       = "flask"
                        # (8 unchanged attributes hidden)

                        # (3 unchanged blocks hidden)
                    }
                }

                # (1 unchanged block hidden)
            }

            # (2 unchanged blocks hidden)
        }

        # (1 unchanged block hidden)
    }

  # kubernetes_ingress_v1.flask_ingress will be updated in-place
  ~ resource "kubernetes_ingress_v1" "flask_ingress" {
        id                    = "flask/flask-ingress"
        # (2 unchanged attributes hidden)

      ~ metadata {
          ~ annotations      = {
```

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]#
```

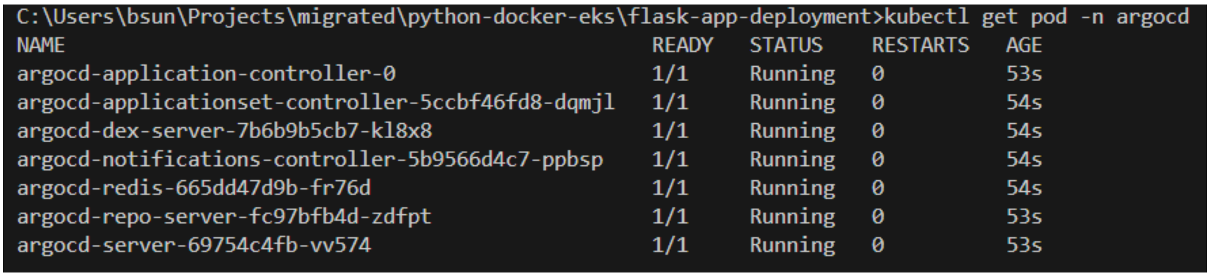
Argo CD

Argo CD is deployed to the EKS cluster by following the steps below:

1. Argo CD only works with yaml manifest files. I need to translate the terraform deployment code in the main.tf. The easiest way to do that is to run below command to export the flask deployment:
   kubectl get deploy -n flask -o yaml > flask-app-deployment/app_manifests/flask-deployment.yaml

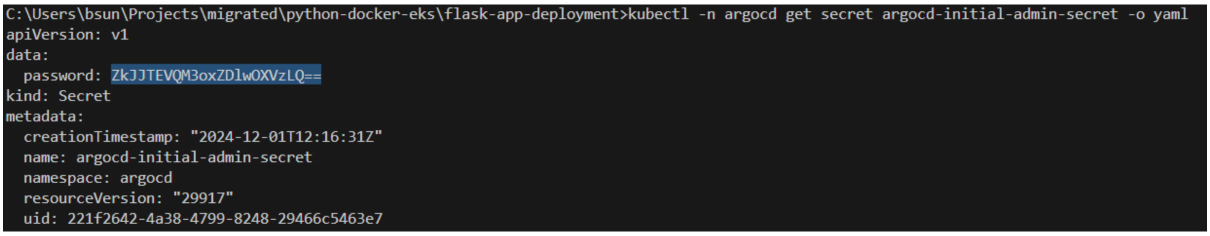
   Note: The file exported contains some additional status information, I need to do some cleanup so that the file can be deployed. Try running "**kubectl apply -f flask-deployment.yaml**" until you get "deployment.apps/flask-deployment configured" without error.

2. Install argocd to the EKS cluster:
   kubectl create namespace argocd
   kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml



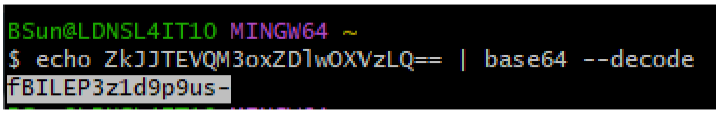
```
C:\Users\bsun\Projects\migrated\python-docker-eks\flask-app-deployment>kubectl get pod -n argocd
NAME                                                READY   STATUS    RESTARTS   AGE
argocd-application-controller-0                     1/1     Running   0          53s
argocd-applicationset-controller-5ccbf46fd8-dqmjl   1/1     Running   0          54s
argocd-dex-server-7b6b9b5cb7-kl8x8                  1/1     Running   0          54s
argocd-notifications-controller-5b9566d4c7-ppbsp    1/1     Running   0          54s
argocd-redis-665dd47d9b-fr76d                       1/1     Running   0          54s
argocd-repo-server-fc97bfb4d-zdfpt                  1/1     Running   0          53s
argocd-server-69754c4fb-vv574                       1/1     Running   0          53s
```

3. Start Argo CD on your local computer and get the passport of "admin" to login the console:
   kubectl port-forward -n argocd svc/argocd-server 8080:443
   kubectl -n argocd get secret argocd-initial-admin-secret -o yaml



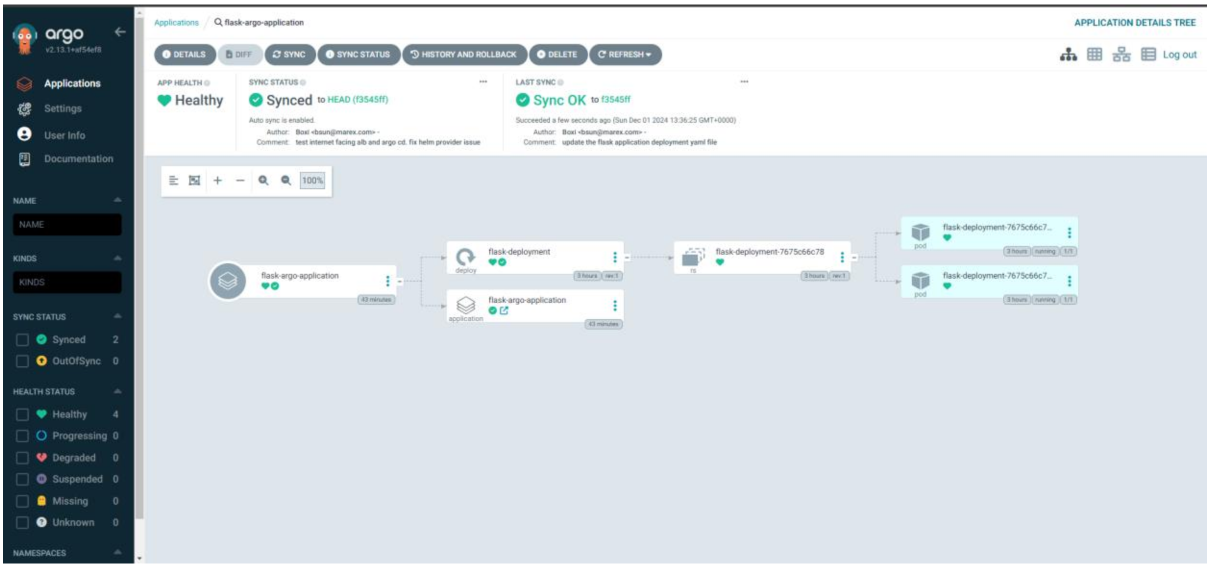
```
C:\Users\bsun\Projects\migrated\python-docker-eks\flask-app-deployment>kubectl -n argocd get secret argocd-initial-admin-secret -o yaml
apiVersion: v1
data:
  password: ZkJJTEVQM3oxZDlwOXVzLQ==
kind: Secret
metadata:
  creationTimestamp: "2024-12-01T12:16:31Z"
  name: argocd-initial-admin-secret
  namespace: argocd
  resourceVersion: "29917"
  uid: 221f2642-4a38-4799-8248-29466c5463e7
```

   echo ZkJJTEVQM3oxZDlwOXVzLQ== | base64 –decode



```
BSun@LDNSL4IT10 MINGW64 ~
$ echo ZkJJTEVQM3oxZDlwOXVzLQ== | base64 --decode
fBILEP3z1d9p9us-
```

4. Link the flask GitHub repository to argocd:
   kubectl apply -f flask-app-deployment/argocd/flask-argo-application.yaml

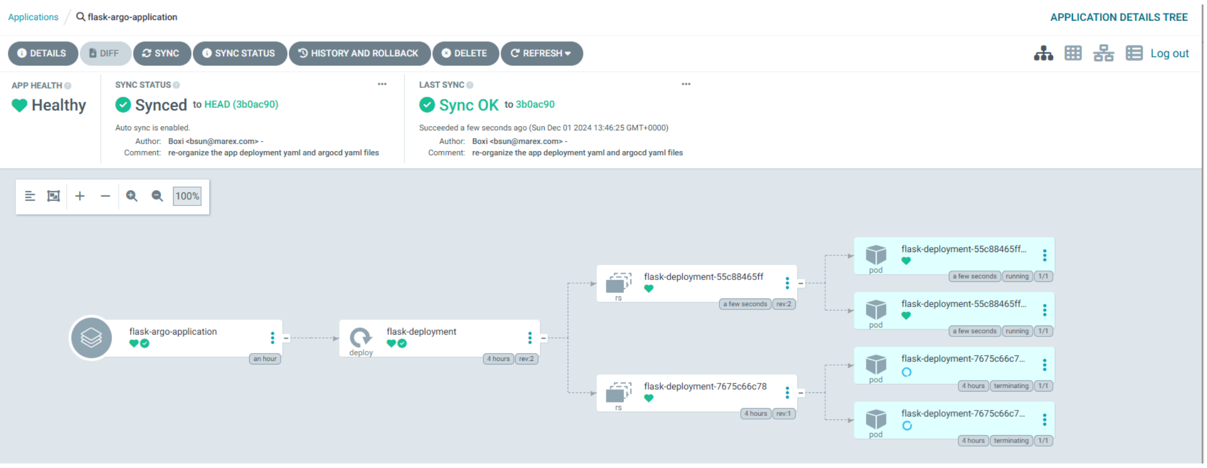


APP HEALTH
Healthy

SYNC STATUS
Synced to HEAD (f3545ff)
Auto sync is enabled.
Author: Boxi <bsun@marex.com> -
Comment: test internet facing alb and argo cd. fix helm provider issue

LAST SYNC
Sync OK to f3545ff
Succeeded a few seconds ago (Sun Dec 01 2024 13:36:25 GMT+0000)
Author: Boxi <bsun@marex.com> -
Comment: update the flask application deployment yaml file

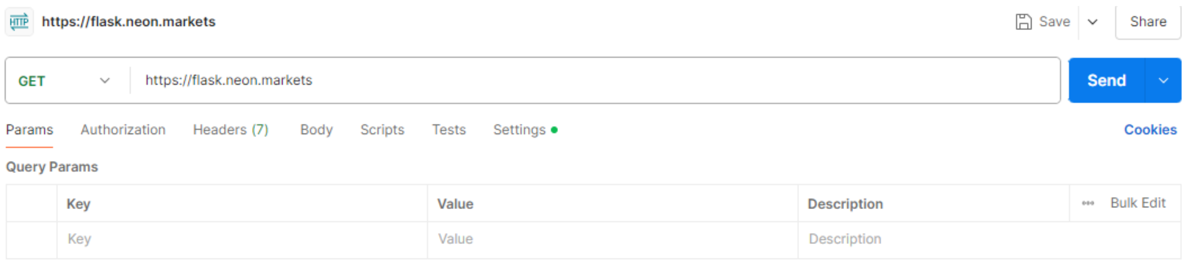
5. Update the image version in the **flask-deployment.yaml** file and verify the change (tested twice with version 0.0.4 and 0.0.3):



APP HEALTH
Healthy

SYNC STATUS
Synced to HEAD (3b0ac90)
Auto sync is enabled.
Author: Boxi <bsun@marex.com> -
Comment: re-organize the app deployment yaml and argocd yaml files

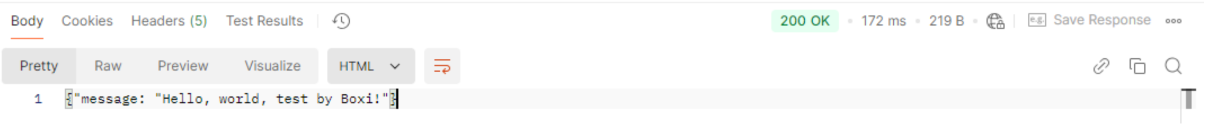
LAST SYNC
Sync OK to 3b0ac90
Succeeded a few seconds ago (Sun Dec 01 2024 13:46:25 GMT+0000)
Author: Boxi <bsun@marex.com> -
Comment: re-organize the app deployment yaml and argocd yaml files



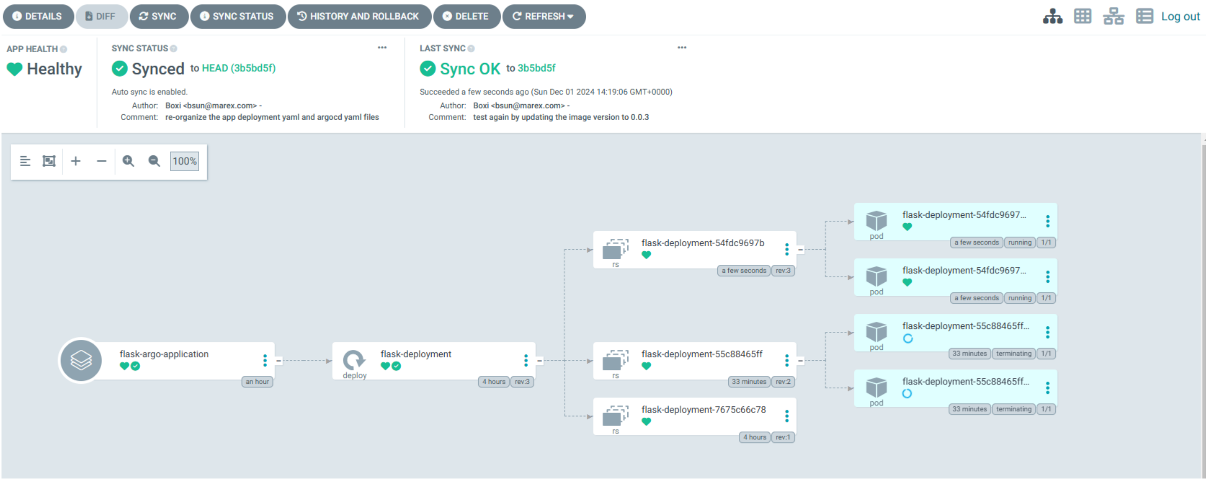
https://flask.neon.markets

Query Params

Key	Value	Description
Key	Value	Description



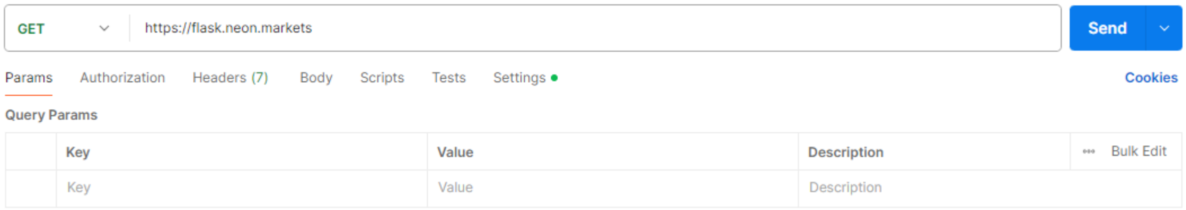
```
{"message: "Hello, world, test by Boxi!"}
```



APP HEALTH
Healthy

SYNC STATUS
Synced to HEAD (3b5bd5f)
Auto sync is enabled.
Author: Boxi <bsun@marex.com> -
Comment: re-organize the app deployment yaml and argocd yaml files

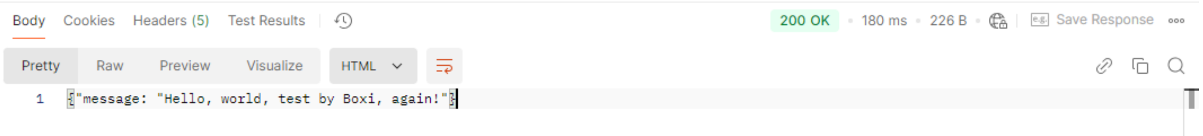
LAST SYNC
Sync OK to 3b5bd5f
Succeeded a few seconds ago (Sun Dec 01 2024 14:19:06 GMT+0000)
Author: Boxi <bsun@marex.com> -
Comment: test again by updating the image version to 0.0.3



https://flask.neon.markets

Query Params

Key	Value	Description
Key	Value	Description



```
{"message: "Hello, world, test by Boxi, again!"}
```

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:

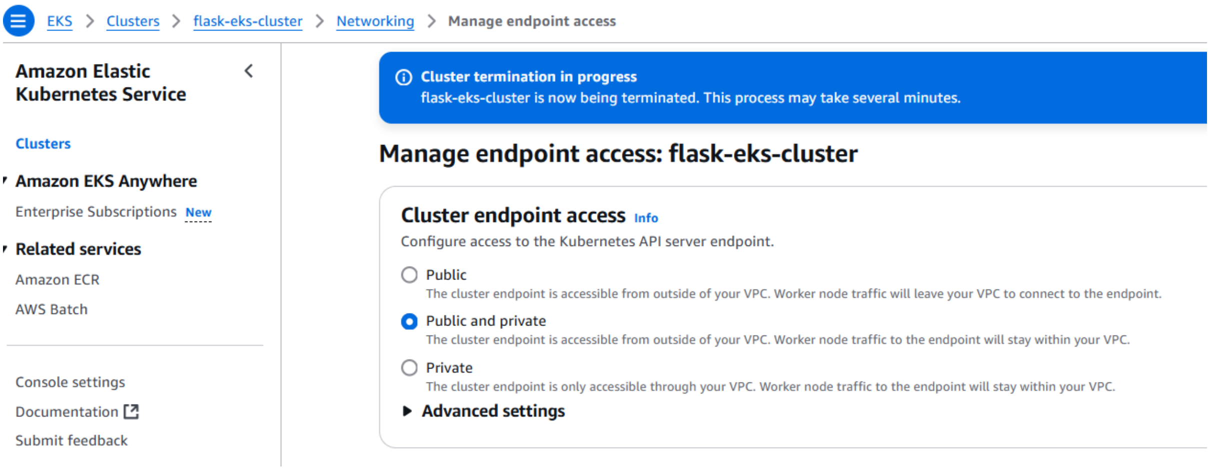
- Check the firewall (AWS security groups) and make sure they are not blocking my IP.

- 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”.


EKS > Clusters > flask-eks-cluster > Networking > Manage endpoint access
Amazon Elastic Kubernetes Service
Clusters
Amazon EKS Anywhere
Enterprise Subscriptions New
Related services
Amazon ECR
AWS Batch
Console settings
Documentation
Submit feedback

Cluster termination in progress
flask-eks-cluster is now being terminated. This process may take several minutes.

Manage endpoint access: flask-eks-cluster

Cluster endpoint access Info
Configure access to the Kubernetes API server endpoint.

○ Public
The cluster endpoint is accessible from outside of your VPC. Worker node traffic will leave your VPC to connect to the endpoint.

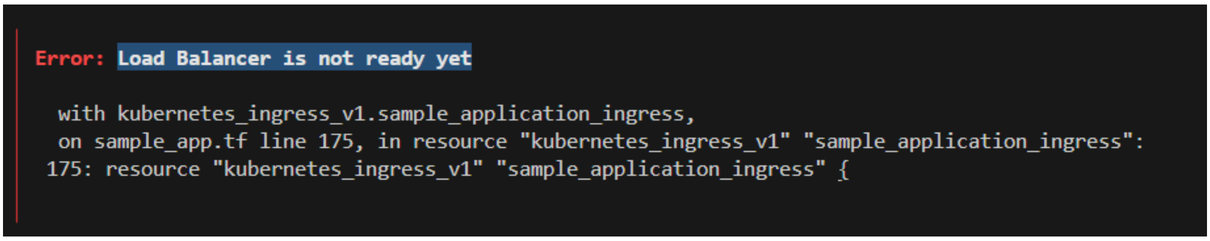
● Public and private
The cluster endpoint is accessible from outside of your VPC. Worker node traffic to the endpoint will stay within your VPC.

○ Private
The cluster endpoint is only accessible through your VPC. Worker node traffic to the endpoint will stay within your VPC.

▶ Advanced settings

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

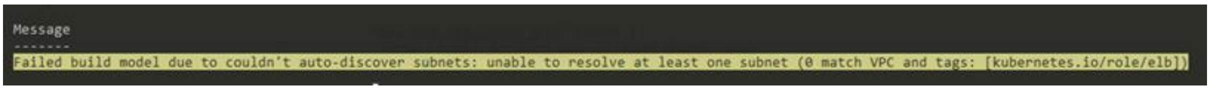
Error:



```
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

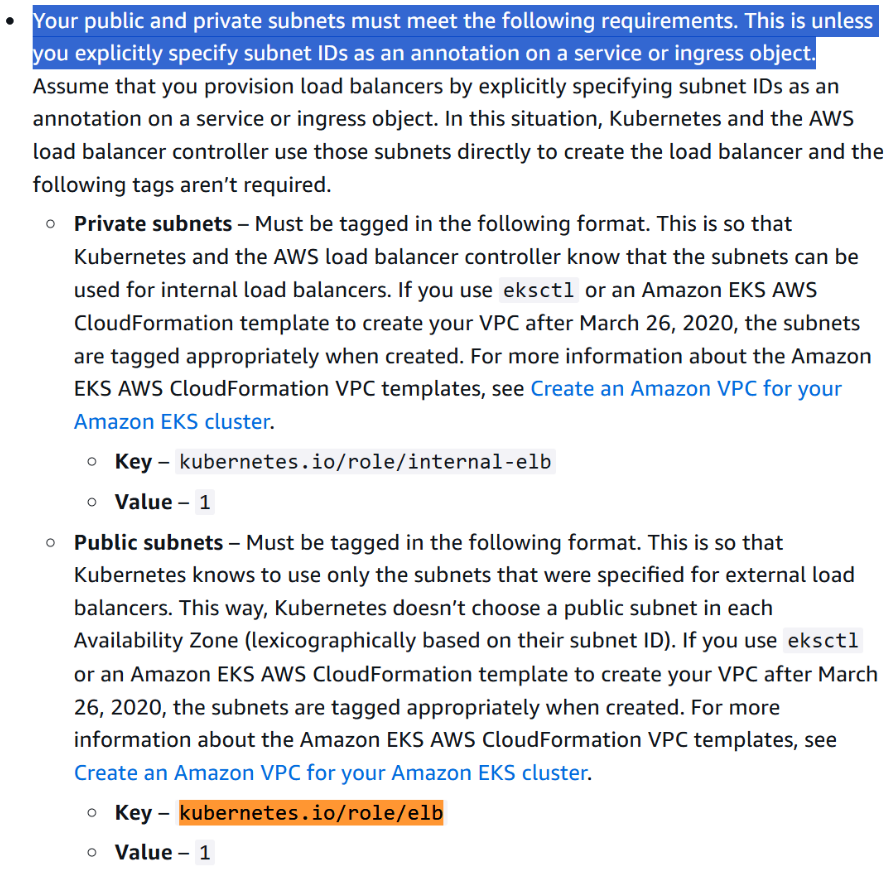


```
Message
-------
Failed build model due to couldn't auto-discover subnets: unable to resolve at least one subnet (0 match VPC and tags: [kubernetes.io/role/elb])
```

Solutions:

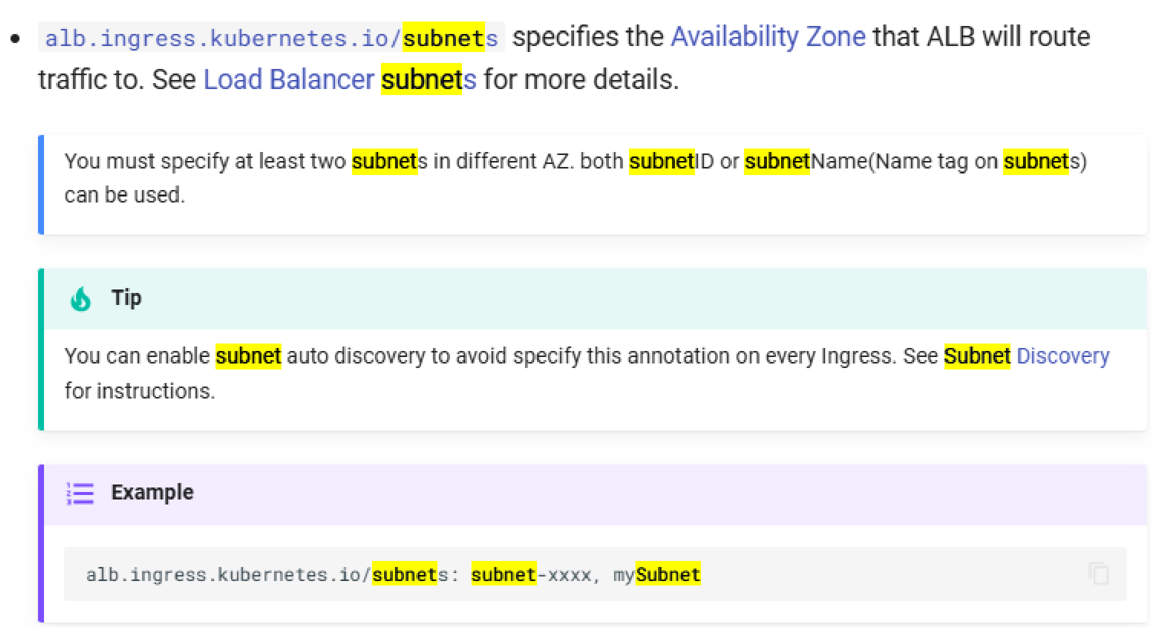
The error indicates that the alb couldn't find any subnet match. Per the document below, I choose to specify subnets as annotations

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.

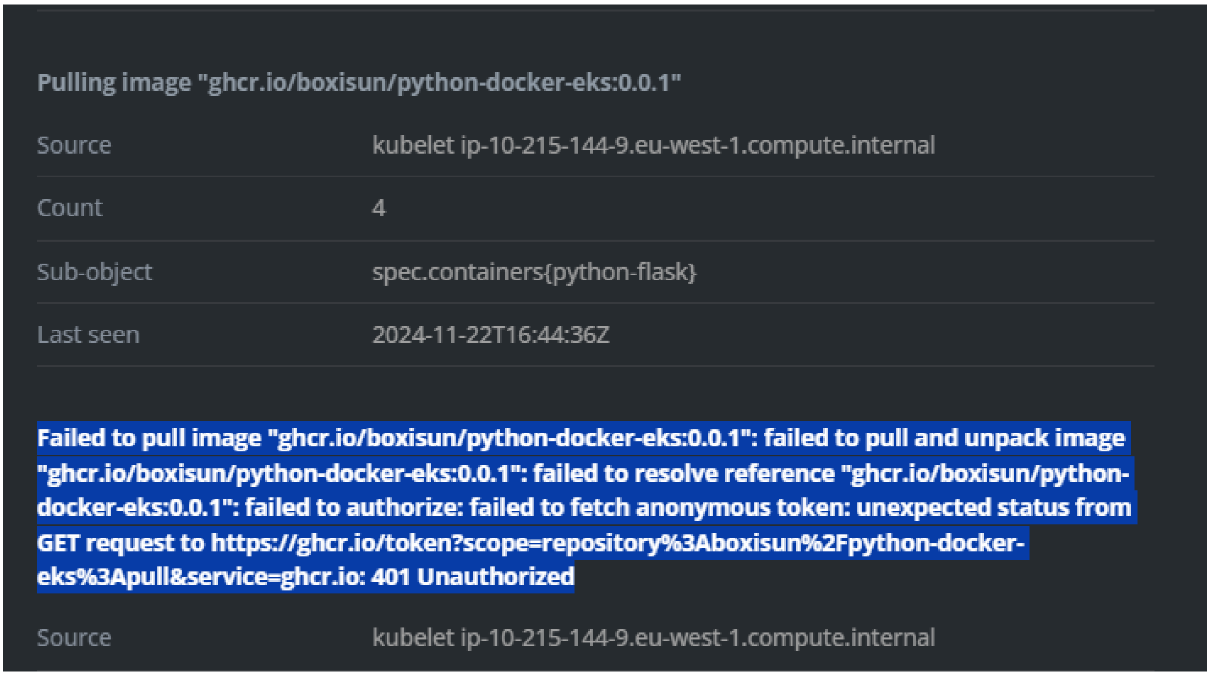
> You must specify at least two subnets in different AZ. both subnetID or subnetName(Name tag on subnets) can be used.

> **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:



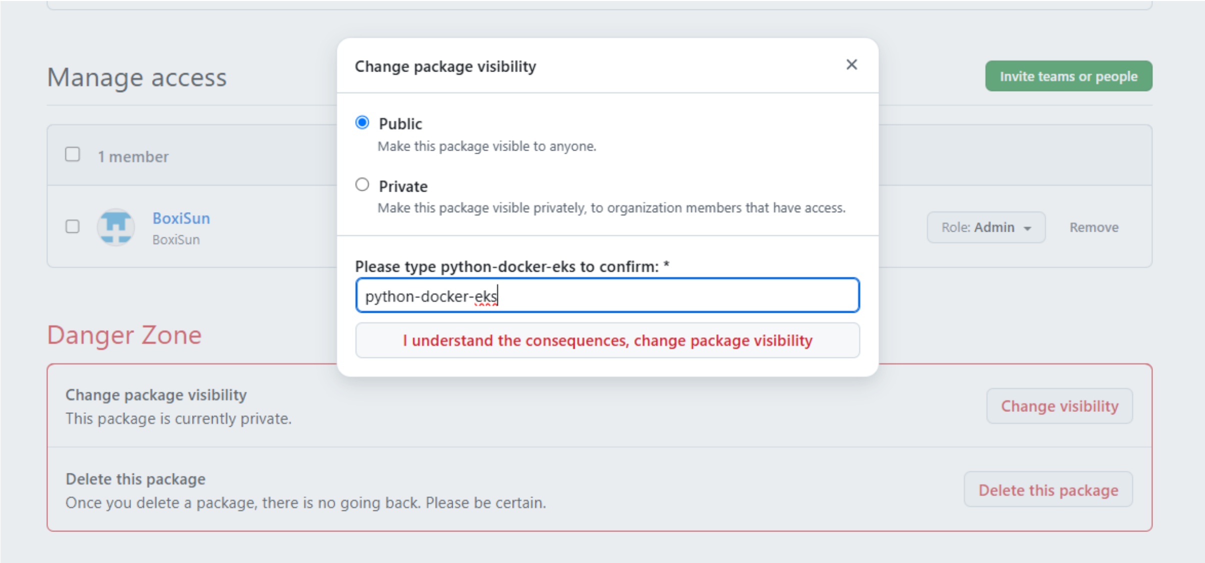
```
Pulling image "ghcr.io/boxisun/python-docker-eks:0.0.1"
Source          kubelet ip-10-215-144-9.eu-west-1.compute.internal
Count           4
Sub-object      spec.containers{python-flask}
Last seen       2024-11-22T16:44:36Z

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:



Change package visibility

● Public
Make this package visible to anyone.

○ Private
Make this package visible privately, to organization members that have access.

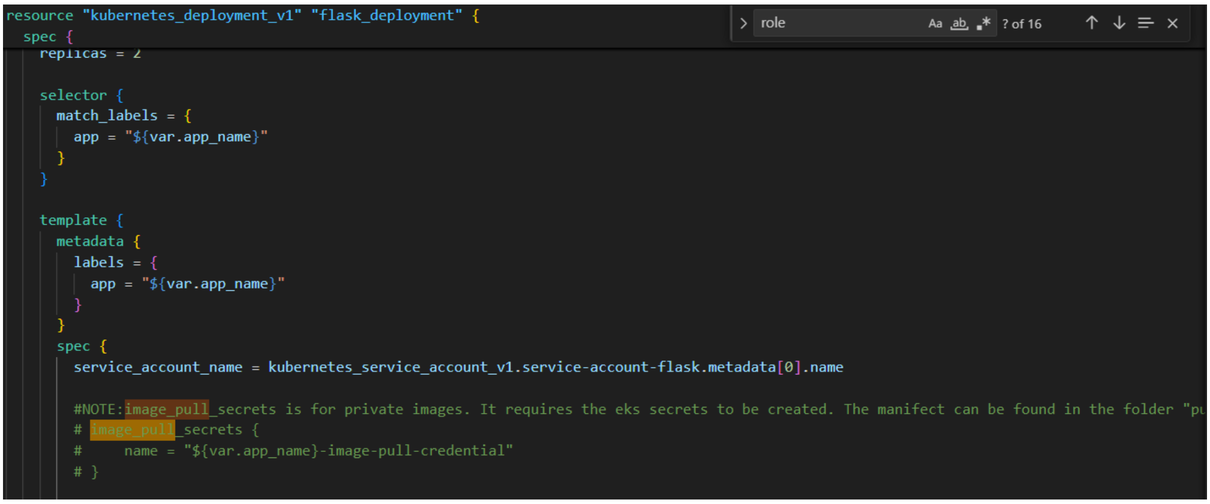
Please type python-docker-eks to confirm: *

python-docker-eks

Change package visibility
This package is currently private.

Delete this package
Once you delete a package, there is no going back. Please be certain.

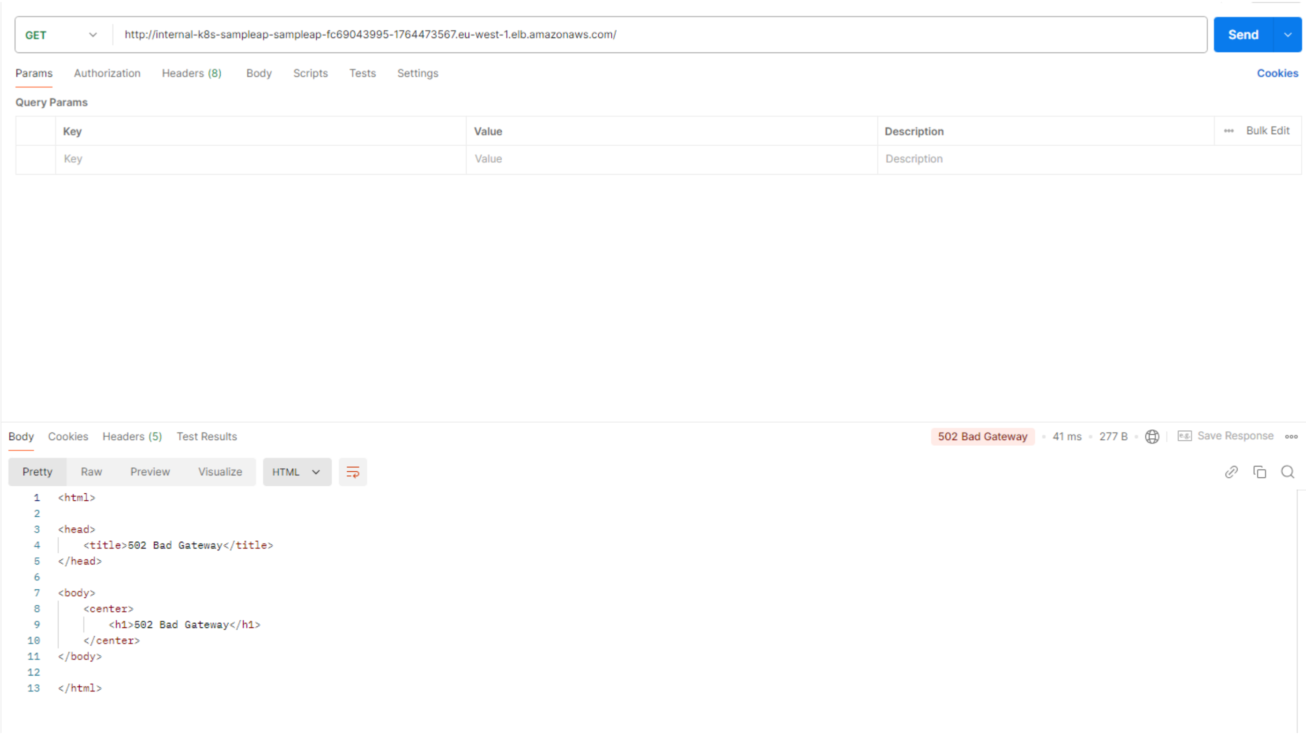
2. Create a personal access token and follow the steps in the document below to generate a dockerconfigjson secret. Apply the secret and add the "image_pull_secret" to the deployment terraform code:



```
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:



GET http://internal-k8s-sampleap-sampleap-fc69043995-1764473567.eu-west-1.elb.amazonaws.com/

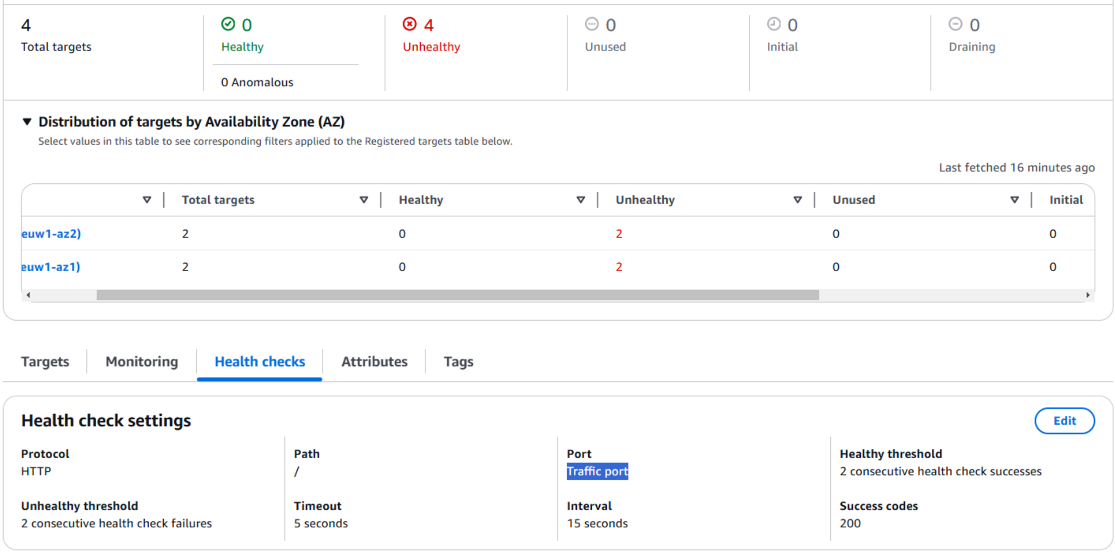
Key	Value	Description
Key	Value	Description

502 Bad Gateway • 41 ms • 277 B

```
<html>

<head>
    <title>502 Bad Gateway</title>
</head>

<body>
    <center>
        <h1>502 Bad Gateway</h1>
    </center>
</body>

</html>
```

Troubleshooting:

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



4	0	4	0	0	0
Total targets	Healthy 0 Anomalous	Unhealthy	Unused	Initial	Draining

Distribution of targets by Availability Zone (AZ)

Select values in this table to see corresponding filters applied to the Registered targets table below.

Last fetched 16 minutes ago

	Total targets	Healthy	Unhealthy	Unused	Initial
euw1-az2)	2	0	2	0	0
euw1-az1)	2	0	2	0	0

Health check settings

Protocol HTTP	Path /	Port Traffic port	Healthy threshold 2 consecutive health check successes
Unhealthy threshold 2 consecutive health check failures	Timeout 5 seconds	Interval 15 seconds	Success codes 200

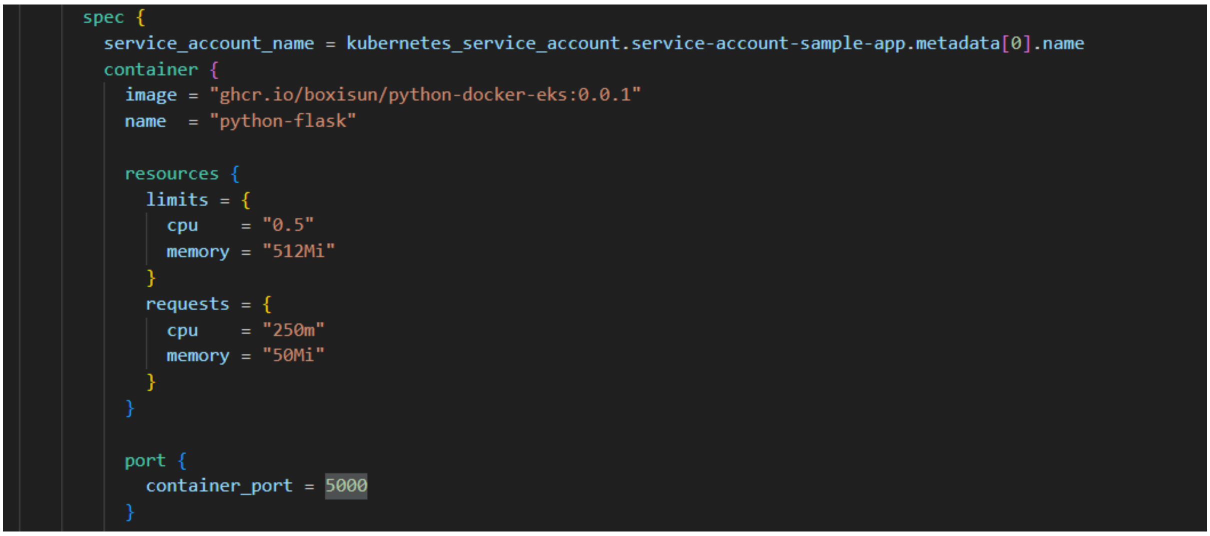
In this case, the “Traffic port” of flask application is 5000. The deployment log confirms that:

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

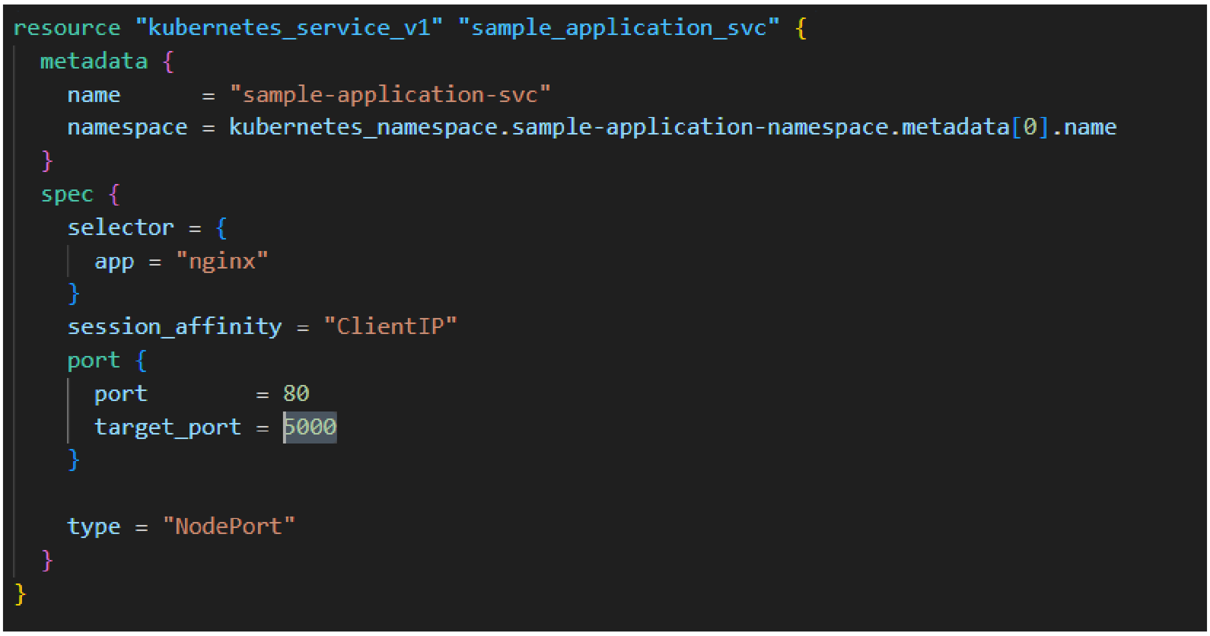
```
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://100.64.0.241:5000
Press CTRL+C to quit
```

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



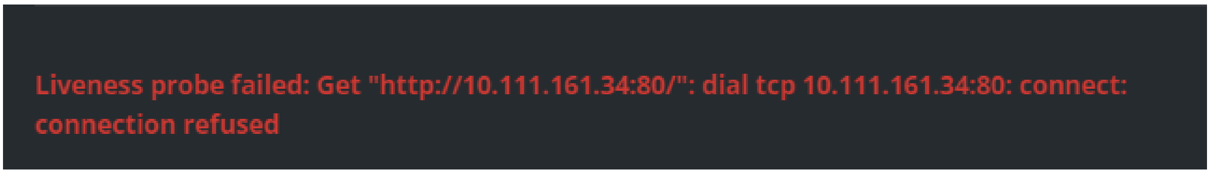
```
    spec {
      service_account_name = kubernetes_service_account.service-account-sample-app.metadata[0].name
      container {
        image = "ghcr.io/boxisun/python-docker-eks:0.0.1"
        name  = "python-flask"

        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)
```