Project 1 Report
Alia Mohamed
March 19, 2023

**Task 1**: Launch a Kubernetes Cluster with one master and at least two workers.
For this task I chose to work with AWS EKS and the command line tool KubeCTL.
The following commands and output were used to setup and test KubeCTL:

```
 kubectl version --short --client
zsh: command not found: kubectl
alia@Alias-MBP ~ % curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.6/2023-01-30/bin/darwin/amd64/kubectl
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 47.9M  100 47.9M    0     0  7729k      0  0:00:06  0:00:06 --:--:-- 8935k
alia@Alias-MBP ~ % curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.6/2023-01-30/bin/darwin/amd64/kubectl.sha256
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    73  100    73    0     0    203      0 --:--:-- --:--:-- --:--:--   207
alia@Alias-MBP ~ % openssl sha1 -sha256 kubectl
SHA256(kubectl)= 9102988689ff79b44334c406d799ff2d1ec4afbe11ac5612c60b51f2e21be72f
alia@Alias-MBP ~ % chmod +x ./kubectl
alia@Alias-MBP ~ % mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
alia@Alias-MBP ~ % echo 'export PATH=$PATH:$HOME/bin' >> ~/.bash_profile
alia@Alias-MBP ~ % kubectl version --short --client
Flag --short has been deprecated, and will be removed in the future. The --short output will become the default.
Client Version: v1.25.6-eks-48e63af
Kustomize Version: v4.5.7
```

To launch the EKS cluster I used the command: `eksctl create cluster --name my-cluster --region us-east-1 --nodegroup-name my-worker-group --node-type t2.micro --nodes 2`
Output for the command:

```
2023-03-19 18:06:24 [i]  eksctl version 0.134.0
2023-03-19 18:06:24 [i]  using region us-east-1
2023-03-19 18:06:25 [i]  setting availability zones to [us-east-1b us-east-1a]
2023-03-19 18:06:25 [i]  subnets for us-east-1b - public:192.168.0.0/19 private:192.168.64.0/19
2023-03-19 18:06:25 [i]  subnets for us-east-1a - public:192.168.32.0/19 private:192.168.96.0/19
2023-03-19 18:06:25 [i]  nodegroup "my-worker-group" will use "" [AmazonLinux2/1.25]
2023-03-19 18:06:25 [i]  using Kubernetes version 1.25
2023-03-19 18:06:25 [i]  creating EKS cluster "my-cluster" in "us-east-1" region with managed nodes
2023-03-19 18:06:25 [i]  will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
2023-03-19 18:06:25 [i]  if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=us-east-1 --cluster=my-cluster'
2023-03-19 18:06:25 [i]  Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "my-cluster" in "us-east-1"
2023-03-19 18:06:25 [i]  CloudWatch logging will not be enabled for cluster "my-cluster" in "us-east-1"
2023-03-19 18:06:25 [i]  you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)} --region=us-east-1 --cluster=my-cluster'
2023-03-19 18:06:25 [i]
2 sequential tasks: { create cluster control plane "my-cluster",
    2 sequential sub-tasks: {
        wait for control plane to become ready,
        create managed nodegroup "my-worker-group",
    }
}
2023-03-19 18:06:25 [i]  building cluster stack "eksctl-my-cluster-cluster"
2023-03-19 18:06:26 [i]  deploying stack "eksctl-my-cluster-cluster"
2023-03-19 18:06:56 [i]  waiting for CloudFormation stack "eksctl-my-cluster-cluster"
2023-03-19 18:07:26 [i]  waiting for CloudFormation stack "eksctl-my-cluster-cluster"
```

```
2023-03-19 18:08:26 [i]  waiting for CloudFormation stack "eksctl-my-cluster-cluster"
2023-03-19 18:09:26 [i]  waiting for CloudFormation stack "eksctl-my-cluster-cluster"
2023-03-19 18:10:26 [i]  waiting for CloudFormation stack "eksctl-my-cluster-cluster"
2023-03-19 18:11:27 [i]  waiting for CloudFormation stack "eksctl-my-cluster-cluster"
2023-03-19 18:12:27 [i]  waiting for CloudFormation stack "eksctl-my-cluster-cluster"
2023-03-19 18:13:27 [i]  waiting for CloudFormation stack "eksctl-my-cluster-cluster"
2023-03-19 18:14:27 [i]  waiting for CloudFormation stack "eksctl-my-cluster-cluster"
2023-03-19 18:15:27 [i]  waiting for CloudFormation stack "eksctl-my-cluster-cluster"
2023-03-19 18:16:27 [i]  waiting for CloudFormation stack "eksctl-my-cluster-cluster"
2023-03-19 18:17:28 [i]  waiting for CloudFormation stack "eksctl-my-cluster-cluster"
2023-03-19 18:18:28 [i]  waiting for CloudFormation stack "eksctl-my-cluster-cluster"
2023-03-19 18:20:30 [i]  building managed nodegroup stack "eksctl-my-cluster-nodegroup-my-worker-group"
2023-03-19 18:20:30 [i]  deploying stack "eksctl-my-cluster-nodegroup-my-worker-group"
2023-03-19 18:20:30 [i]  waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-worker-group"
2023-03-19 18:21:01 [i]  waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-worker-group"
2023-03-19 18:21:31 [i]  waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-worker-group"
2023-03-19 18:22:53 [i]  waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-worker-group"
2023-03-19 18:24:08 [i]  waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-worker-group"
2023-03-19 18:24:08 [i]  waiting for the control plane to become ready
2023-03-19 18:24:08 [✓]  saved kubeconfig as "/Users/alia/.kube/config"
2023-03-19 18:24:08 [i]  no tasks
2023-03-19 18:24:08 [✓]  all EKS cluster resources for "my-cluster" have been created
2023-03-19 18:24:08 [i]  nodegroup "my-worker-group" has 2 node(s)
2023-03-19 18:24:08 [i]  node "ip-192-168-12-77.ec2.internal" is ready
2023-03-19 18:24:08 [i]  node "ip-192-168-40-75.ec2.internal" is ready
2023-03-19 18:24:08 [i]  waiting for at least 2 node(s) to become ready in "my-worker-group"
2023-03-19 18:24:08 [i]  nodegroup "my-worker-group" has 2 node(s)
2023-03-19 18:24:08 [i]  node "ip-192-168-12-77.ec2.internal" is ready
2023-03-19 18:24:08 [i]  node "ip-192-168-40-75.ec2.internal" is ready
2023-03-19 18:24:10 [i]  kubectl command should work with "/Users/alia/.kube/config", try 'kubectl get nodes'
2023-03-19 18:24:10 [✓]  EKS cluster "my-cluster" in "us-east-1" region is ready
```

This operation took 18 minutes to complete resulting the creation of the cluster with two worker nodes and active EC2 instances

The following commands and output were used to obtain information about the cluster and nodes:

Command: kubectl get nodes -o wide

Output:

```
NAME                STATUS   ROLES    AGE    VERSION               INTERNAL-IP     EXTERNAL-IP     OS-
IMAGE         KERNEL-VERSION                CONTAINER-RUNTIME
ip-192-168-12-77.ec2.internal   Ready    <none>   7m36s   v1.25.6-eks-48e63af   192.168.12.77   18.234.214.145   Amazon Linux
2   5.10.167-147.601.amzn2.x86_64   containerd://1.6.6
ip-192-168-40-75.ec2.internal   Ready    <none>   7m39s   v1.25.6-eks-48e63af   192.168.40.75   3.88.115.10      Amazon Linux
2   5.10.167-147.601.amzn2.x86_64   containerd://1.6.6
```

Command: kubectl get pods -A -o wide

Output:

```
NAMESPACE     NAME                  READY   STATUS    RESTARTS   AGE     IP              NODE                        NOMINATED
NODE   READINESS GATES
kube-system   aws-node-2rt27        1/1     Running   0          8m12s   192.168.40.75   ip-192-168-40-
75.ec2.internal   <none>           <none>
kube-system   aws-node-7q7fk        1/1     Running   0          8m9s    192.168.12.77   ip-192-168-12-
77.ec2.internal   <none>           <none>
kube-system   coredns-7975d6fb9b-8dvrb   1/1     Running   0          16m     192.168.59.169   ip-192-168-40-
75.ec2.internal   <none>           <none>
kube-system   coredns-7975d6fb9b-shv2b   1/1     Running   0          16m     192.168.60.116   ip-192-168-40-
75.ec2.internal   <none>           <none>
kube-system   kube-proxy-cvg8j      1/1     Running   0          8m12s   192.168.40.75   ip-192-168-40-
75.ec2.internal   <none>           <none>
```

```
kube-system   kube-proxy-dgssd         1/1     Running   0          8m9s    192.168.12.77   ip-192-168-12-
77.ec2.internal   <none>           <none>
```

**Task 2**: Deploy a containerized application which run multiple instances of the same container
For this task there were multiple approaches. The first approach was following the instruction on the EKS documentation as follows:

Creating a namespace to allow the creation of the application and group resources in the cluster:
Command: kubectl create namespace my-kubernetes-namespace
Output:
namespace/my-kubernetes-namespace created

creating deployment manifest using the code: this is the code for the deployment file for the application.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: eks-sample-linux-deployment
  namespace: my-kubernets-namespace
  labels:
    app: eks-sample-linux-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: eks-sample-linux-app
  template:
    metadata:
      labels:
        app: eks-sample-linux-app
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: kubernetes.io/arch
                operator: In
                values:
                - amd64
                - arm64
      containers:
      - name: nginx
        image: public.ecr.aws/nginx/nginx:1.21
        ports:
        - name: http
```

```
        containerPort: 80
      imagePullPolicy: IfNotPresent
    nodeSelector:
      kubernetes.io/os: linux
```

Applying the deployment manifest to the cluster:
Command: kubectl apply -f eks-sample-deployment.yaml
Output:
deployment.apps/eks-sample-linux-deployment created

Creating a service with the code: this is the code for the service provided by the application.

```
apiVersion: v1
kind: Service
metadata:
  name: eks-sample-linux-service
  namespace: my-kubernets-namespace
  labels:
    app: eks-sample-linux-app
spec:
  selector:
    app: eks-sample-linux-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

applying the server manifest to the cluster:
Command: kubectl apply -f eks-sample-service.yaml
Output:
service/eks-sample-linux-service created

Viewing resources of the namespace:
Command: kubectl get all -n my-kubernetes-namespace
Output:

```
NAME                                         READY   STATUS              RESTARTS   AGE
pod/eks-sample-linux-deployment-7f646d456c-84jfx   1/1     Running             0          33s
pod/eks-sample-linux-deployment-7f646d456c-ktg2p   0/1     Pending             0          33s
pod/eks-sample-linux-deployment-7f646d456c-nbkw7   0/1     ContainerCreating   0          33s
NAME                             TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
service/eks-sample-linux-service   ClusterIP   10.100.241.255   <none>        80/TCP    14s
NAME                                    READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/eks-sample-linux-deployment   1/3     3            1           33s
NAME                                               DESIRED   CURRENT   READY   AGE
replicaset.apps/eks-sample-linux-deployment-7f646d456c   3         3         1       33s
```

Details of the service:
Command: kubectl -n my-kubernetes-namespace describe service eks-sample-linux-service

## Output:

```
Name:              eks-sample-linux-service
Namespace:         my-kubernetes-namespace
Labels:            app=eks-sample-linux-app
Annotations:       <none>
Selector:          app=eks-sample-linux-app
Type:              ClusterIP
IP Family Policy:  SingleStack
IP Families:       IPv4
IP:                10.100.241.255
IPs:               10.100.241.255
Port:              <unset>  80/TCP
TargetPort:        80/TCP
Endpoints:         192.168.23.117:80,192.168.4.191:80
Session Affinity:  None
Events:            <none>
```

## Details for one of the pods:

## Command: kubectl -n my-kubernetes-namespace describe pod eks-sample-linux-deployment-7f646d456c-84jfx

## Output:

```
Name:             eks-sample-linux-deployment-7f646d456c-84jfx
Namespace:        my-kubernetes-namespace
Priority:         0
Service Account:  default
Node:             ip-192-168-12-77.ec2.internal/192.168.12.77
Start Time:       Sun, 19 Mar 2023 18:31:40 -0400
Labels:           app=eks-sample-linux-app
                  pod-template-hash=7f646d456c
Annotations:      <none>
Status:           Running
IP:               192.168.4.191
IPs:
  IP:           192.168.4.191
Controlled By:  ReplicaSet/eks-sample-linux-deployment-7f646d456c
Containers:
  nginx:
    Container ID:   containerd://e6f861822c3854de7b768987875a7cfdfa646e5b902b3f99e73590a8812df6eb
    Image:          public.ecr.aws/nginx/nginx:1.21
    Image ID:       public.ecr.aws/nginx/nginx@sha256:3aac7c736093ce043a17d6e83ef5addb8be321b5b6b93879141e51474448ca65
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Sun, 19 Mar 2023 18:31:46 -0400
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-29zkz (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  kube-api-access-29zkz:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              kubernetes.io/os=linux
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
```

```
                 node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age    From               Message
  ----    ------     ----   ----               -------
  Normal  Scheduled  2m19s  default-scheduler  Successfully assigned my-kubernetes-namespace/eks-sample-linux-deployment-
7f646d456c-84jfx to ip-192-168-12-77.ec2.internal
  Normal  Pulling    2m18s  kubelet            Pulling image "public.ecr.aws/nginx/nginx:1.21"
  Normal  Pulled     2m14s  kubelet            Successfully pulled image "public.ecr.aws/nginx/nginx:1.21" in 4.265129131s (4.265166386s
including waiting)
  Normal  Created    2m14s  kubelet            Created container nginx
  Normal  Started    2m13s  kubelet            Started container nginx
```

The second approach was to use another service application hamister.yaml

Deploying the application:

Command: kubectl apply -f examples/hamster.yaml

Output:

```
verticalpodautoscaler.autoscaling.k8s.io/hamster-vpa created
deployment.apps/hamster created
```

viewing pods for the application

Command: kubectl get pods -l app=hamster

Output:

```
NAME                   READY   STATUS    RESTARTS   AGE
hamster-59cc68d575-hhpsj   0/1     Pending   0          27s
hamster-59cc68d575-xd9wf   0/1     Pending   0          27s
```

description for one of the pods:

Command: kubectl describe pod hamster-59cc68d575-hhpsj

Output:

```
Name:             hamster-59cc68d575-hhpsj
Namespace:        default
Priority:         0
Service Account:  default
Node:             <none>
Labels:           app=hamster
                  pod-template-hash=59cc68d575
Annotations:      <none>
Status:           Pending
IP:
IPs:              <none>
Controlled By:    ReplicaSet/hamster-59cc68d575
Containers:
  hamster:
    Image:      registry.k8s.io/ubuntu-slim:0.1
    Port:       <none>
    Host Port:  <none>
    Command:
      /bin/sh
    Args:
      -c
      while true; do timeout 0.5s yes >/dev/null; sleep 0.5s; done
    Requests:
      cpu:        100m
      memory:     50Mi
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-66r2g (ro)
Conditions:
  Type           Status
  PodScheduled   False
```

```
Volumes:
  kube-api-access-66r2g:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   Burstable
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason            Age   From               Message
  ----     ------            ----  ----               -------
  Warning  FailedScheduling  67s   default-scheduler  0/2 nodes are available: 2 Too many pods. preemption: 0/2 nodes are available: 2
No preemption victims found for incoming pod.
```

All operation during this task did not take a very long time to execute. All operation almost executed instantly.

**Task 3**: Scale the pod to more container instances.
Looking through the documentation provided for EKS, the options that I found for scaling the cluster were the Vertical and Horizontal Autoscalers which I used for this part of the project.

Deploying the Vertical autoscaler: this step will allow the cluster to grow vertically creating more space and resources for the application
Command: vertical-pod-autoscaler % ./hack/vpa-up.sh
Output:

```
customresourcedefinition.apiextensions.k8s.io/verticalpodautoscalercheckpoints.autoscaling.k8s.io created
customresourcedefinition.apiextensions.k8s.io/verticalpodautoscalers.autoscaling.k8s.io created
clusterrole.rbac.authorization.k8s.io/system:metrics-reader created
clusterrole.rbac.authorization.k8s.io/system:vpa-actor created
clusterrole.rbac.authorization.k8s.io/system:vpa-checkpoint-actor created
clusterrole.rbac.authorization.k8s.io/system:evictioner created
clusterrolebinding.rbac.authorization.k8s.io/system:metrics-reader created
clusterrolebinding.rbac.authorization.k8s.io/system:vpa-actor created
clusterrolebinding.rbac.authorization.k8s.io/system:vpa-checkpoint-actor created
clusterrole.rbac.authorization.k8s.io/system:vpa-target-reader created
clusterrolebinding.rbac.authorization.k8s.io/system:vpa-target-reader-binding created
clusterrolebinding.rbac.authorization.k8s.io/system:vpa-evictioner-binding created
serviceaccount/vpa-admission-controller created
serviceaccount/vpa-recommender created
serviceaccount/vpa-updater created
clusterrole.rbac.authorization.k8s.io/system:vpa-admission-controller created
clusterrolebinding.rbac.authorization.k8s.io/system:vpa-admission-controller created
clusterrole.rbac.authorization.k8s.io/system:vpa-status-reader created
clusterrolebinding.rbac.authorization.k8s.io/system:vpa-status-reader-binding created
deployment.apps/vpa-updater created
deployment.apps/vpa-recommender created
Generating certs for the VPA Admission Controller in /tmp/vpa-certs.
Generating RSA private key, 2048 bit long modulus
.....................................................................+++
.........................................+++
e is 65537 (0x10001)
unknown option -addext
req [options] <infile >outfile
where options  are
 -inform arg    input format - DER or PEM
 -outform arg   output format - DER or PEM
 -in arg        input file
 -out arg       output file
```

```
-text          text form of request
-pubkey        output public key
-noout         do not output REQ
-verify        verify signature on REQ
-modulus       RSA modulus
-nodes         don't encrypt the output key
-subject       output the request's subject
-passin        private key password source
-key file      use the private key contained in file
-keyform arg   key file format
-keyout arg    file to send the key to
-newkey rsa:bits generate a new RSA key of 'bits' in size
-newkey dsa:file generate a new DSA key, parameters taken from CA in 'file'
-newkey ec:file generate a new EC key, parameters taken from CA in 'file'
-[digest]      Digest to sign with (md5, sha1, md4)
-config file   request template file.
-subj arg      set or modify request subject
-multivalue-rdn enable support for multivalued RDNs
-new           new request.
-batch         do not ask anything during request generation
-x509          output a x509 structure instead of a cert. req.
-days          number of days a certificate generated by -x509 is valid for.
-set_serial    serial number to use for a certificate generated by -x509.
-newhdr        output "NEW" in the header lines
-asn1-kludge   Output the 'request' in a format that is wrong but some CA's
               have been reported as requiring
-extensions .. specify certificate extension section (override value in config file)
-reqexts ..    specify request extension section (override value in config file)
-utf8          input characters are UTF8 (default ASCII)
-nameopt arg    - various certificate name options
-reqopt arg    - various request text options
ERROR: Failed to create CA certificate for self-signing. If the error is "unknown option -addext", update your openssl version or deploy
VPA from the vpa-release-0.8 branch.
deployment.apps/vpa-admission-controller created
service/vpa-webhook created
```

## Verifying that the autoscaler works:

Command: kubectl get pods -n kube-system

Output:

```
NAME                                READY   STATUS    RESTARTS   AGE
aws-node-2rt27                      1/1     Running   0          120m
aws-node-7q7fk                      1/1     Running   0          120m
coredns-7975d6fb9b-8dvrb            1/1     Running   0          128m
coredns-7975d6fb9b-shv2b            1/1     Running   0          128m
kube-proxy-cvg8j                    1/1     Running   0          120m
kube-proxy-dgssd                    1/1     Running   0          120m
metrics-server-8ff8f88c6-4zg99      1/1     Running   0          97m
vpa-admission-controller-5cc669f7f8-6v28m   0/1     Pending   0          2m19s
vpa-recommender-5bcd4fd7bc-gzgpx    0/1     Pending   0          2m20s
vpa-updater-68c46997f8-278vx
```

## Describing the VPA for the scaler:

Command: kubectl describe vpa/hamster-vpa

Output:

```
Name:         hamster-vpa
Namespace:    default
Labels:       <none>
Annotations:  <none>
API Version:  autoscaling.k8s.io/v1
Kind:         VerticalPodAutoscaler
Metadata:
  Creation Timestamp:  2023-03-20T00:23:29Z
  Generation:          1
```

```
  Managed Fields:
    API Version:  autoscaling.k8s.io/v1
    Fields Type:  FieldsV1
    fieldsV1:
      f:metadata:
        f:annotations:
          .:
          f:kubectl.kubernetes.io/last-applied-configuration:
      f:spec:
        .:
        f:resourcePolicy:
          .:
          f:containerPolicies:
        f:targetRef:
    Manager:         kubectl-client-side-apply
    Operation:       Update
    Time:            2023-03-20T00:23:29Z
  Resource Version:  28782
  UID:               98ff8a63-d6cd-402c-b34d-56bac2ff6260
Spec:
  Resource Policy:
    Container Policies:
      Container Name:  *
      Controlled Resources:
        cpu
        memory
      Max Allowed:
        Cpu:     1
        Memory:  500Mi
      Min Allowed:
        Cpu:     100m
        Memory:  50Mi
  Target Ref:
    API Version:  apps/v1
    Kind:         Deployment
    Name:         hamster
Events:           <none>
```

Creating a horizontal aoutoscaler: this step will allow the cluster to grow horizontally providing more resources for the services.
Command: kubectl autoscale deployment hamster --cpu-percent=50 --min=1 --max=10
Output:

```
horizontalpodautoscaler.autoscaling/hamster autoscaled
```

Describing the autoscaler:
Command: kubectl get hpa
Output:

```
NAME      REFERENCE           TARGETS         MINPODS   MAXPODS   REPLICAS   AGE
hamster   Deployment/hamster   <unknown>/50%   1         10        2          61s
```

for this task I noticed that the autoscores work to expand the network as needed when a service that need more CPU and Memory is deployed by creating more replicas and instances.

**Task 4**: Update the application with a new software version
For this task I chose to update the hamster.yaml application which was already up to date when it was first installed.

Command: kubectl patch deployment hamster -p '{"space": {"minReadySeconds": 20}}'
Output:
**Warning:** unknown field "space"
deployment.apps/hamster patched (no change)

**Task 5**: delete the application and stop the cluster

Deleting the application: deleting the application filer that were created.
Command: kubectl delete -f examples/hamster.yaml
Output:
verticalpodautoscaler.autoscaling.k8s.io "hamster-vpa" deleted
deployment.apps "hamster" deleted

Stopping the cluster: deleting the cluster and all the resources and file that were created.
Command: delete cluster --name my-cluster
Output:
2023-03-19 21:01:56 [**i**] deleting EKS cluster "my-cluster"
2023-03-19 21:01:56 [**i**] will drain 0 unmanaged nodegroup(s) in cluster "my-cluster"
2023-03-19 21:01:56 [**i**] starting parallel draining, max in-flight of 1
2023-03-19 21:01:57 [**i**] deleting Fargate profile "my-fargate-profile"
2023-03-19 21:06:14 [**i**] deleted Fargate profile "my-fargate-profile"
2023-03-19 21:06:14 [**i**] deleted 1 Fargate profile(s)
2023-03-19 21:06:14 [**i**] will delete stack "eksctl-my-cluster-fargate"
2023-03-19 21:06:14 [✓] kubeconfig has been updated
2023-03-19 21:06:14 [**i**] cleaning up AWS load balancers created by Kubernetes objects of Kind Service or Ingress
2023-03-19 21:06:16 [**i**]
2 sequential tasks: { delete nodegroup "my-worker-group", delete cluster control plane "my-cluster" [async]
}
2023-03-19 21:06:16 [**i**] will delete stack "eksctl-my-cluster-nodegroup-my-worker-group"
2023-03-19 21:06:16 [**i**] waiting for stack "eksctl-my-cluster-nodegroup-my-worker-group" to get deleted
2023-03-19 21:06:16 [**i**] waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-worker-group"
2023-03-19 21:06:46 [**i**] waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-worker-group"
2023-03-19 21:07:30 [**i**] waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-worker-group"
2023-03-19 21:08:52 [**i**] waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-worker-group"
2023-03-19 21:09:53 [**i**] waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-worker-group"
2023-03-19 21:11:07 [**i**] waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-worker-group"
2023-03-19 21:12:26 [**i**] waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-worker-group"
2023-03-19 21:14:05 [**i**] waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-worker-group"
2023-03-19 21:15:53 [**i**] waiting for CloudFormation stack "eksctl-my-cluster-nodegroup-my-worker-group"
2023-03-19 21:15:53 [**i**] will delete stack "eksctl-my-cluster-cluster"
2023-03-19 21:15:54 [✓] all cluster resources were deleted

# References

Installing or updating kubectl from https://docs.aws.amazon.com/eks/latest/userguide/install-kubectl.html

Getting started with Amazon EKS – eksctl from https://docs.aws.amazon.com/eks/latest/userguide/getting-started-eksctl.html

AWS Fargate Profile from https://docs.aws.amazon.com/eks/latest/userguide/fargate-profile.html

Deploy a sample application from https://docs.aws.amazon.com/eks/latest/userguide/sample-deployment.html

Installing the Kubernetes Metrics Server from https://docs.aws.amazon.com/eks/latest/userguide/metrics-server.html

Horizontal Pod Autoscaler from https://docs.aws.amazon.com/eks/latest/userguide/horizontal-pod-autoscaler.html

Vertical Pod Autoscaler from https://docs.aws.amazon.com/eks/latest/userguide/vertical-pod-autoscaler.html