Final Manifest

CSYE 7220 - Development and Operations

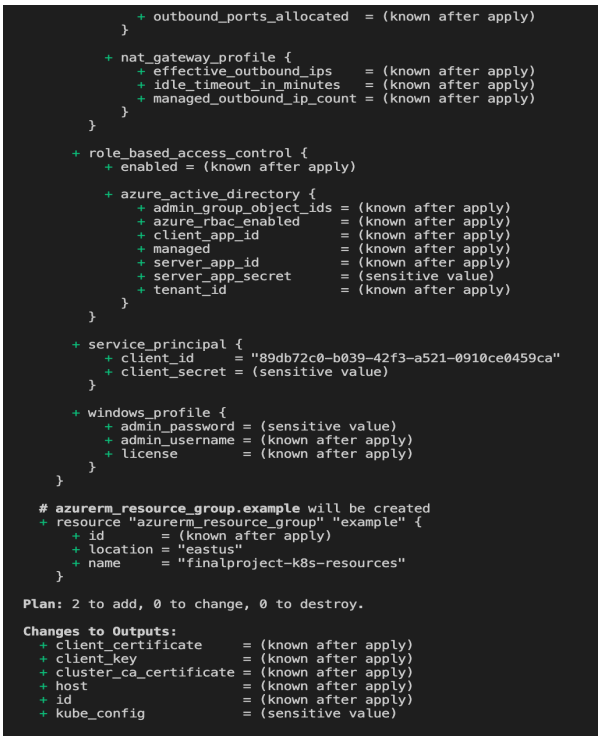
Deep Gamit - 001057437
Priyam Suthar - 001054934

Deploying the Rockstar App to AKS cluster with Prometheus for Monitoring

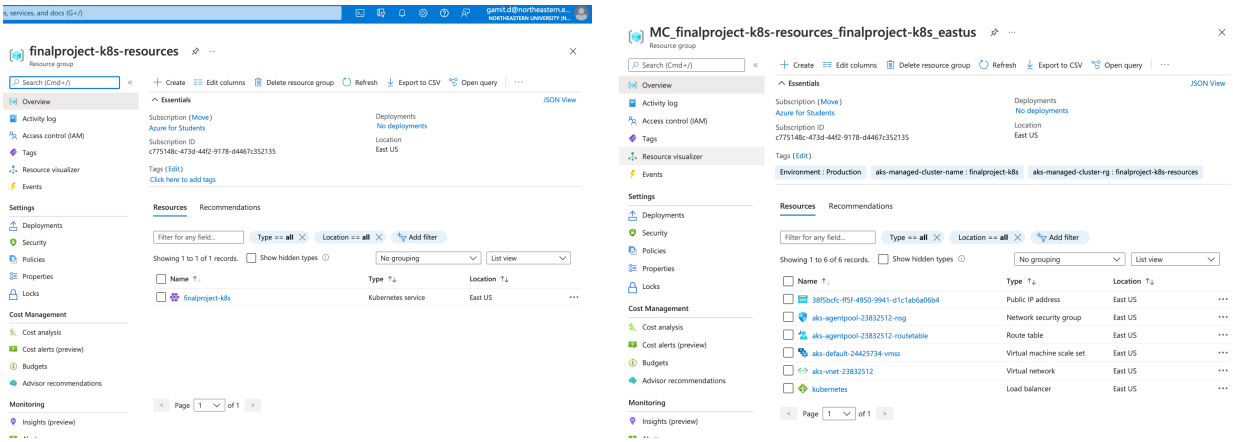
Step 1: We deploy the resources using terraform on Azure Kubernetes Service by applying the following commands:

```
terraform init
terraform plan
terraform apply
```

```
deepgamit@Deeps-MacBook-Pro AKS % terraform plan -var-file="azure.tfvars"
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.


------------------------------------------------------------------------

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # azurerm_kubernetes_cluster.example will be created
  + resource "azurerm_kubernetes_cluster" "example" {
      + dns_prefix                          = "finalproject-k8s"
      + fqdn                                = (known after apply)
      + id                                  = (known after apply)
      + kube_admin_config                   = (known after apply)
      + kube_admin_config_raw               = (sensitive value)
      + kube_config                         = (known after apply)
      + kube_config_raw                     = (sensitive value)
      + kubernetes_version                  = (known after apply)
      + location                            = "eastus"
      + name                                = "finalproject-k8s"
      + node_resource_group                 = (known after apply)
      + portal_fqdn                         = (known after apply)
      + private_cluster_enabled             = (known after apply)
      + private_cluster_public_fqdn_enabled = false
      + private_dns_zone_id                 = (known after apply)
      + private_fqdn                        = (known after apply)
      + private_link_enabled                = (known after apply)
      + resource_group_name                 = "finalproject-k8s-resources"
      + sku_tier                            = "Free"
      + tags                                = {
          + "Environment" = "Production"
        }

      + addon_profile {
          + aci_connector_linux {
              + enabled     = (known after apply)
              + subnet_name = (known after apply)
            }

          + azure_keyvault_secrets_provider {
              + enabled                  = (known after apply)
              + secret_identity          = (known after apply)
              + secret_rotation_enabled  = (known after apply)
              + secret_rotation_interval = (known after apply)
            }

          + azure_policy {
```



```
              + outbound_ports_allocated  = (known after apply)
            }

          + nat_gateway_profile {
              + effective_outbound_ips    = (known after apply)
              + idle_timeout_in_minutes   = (known after apply)
              + managed_outbound_ip_count = (known after apply)
            }
        }

      + role_based_access_control {
          + enabled = (known after apply)

          + azure_active_directory {
              + admin_group_object_ids = (known after apply)
              + azure_rbac_enabled     = (known after apply)
              + client_app_id          = (known after apply)
              + managed                = (known after apply)
              + server_app_id          = (known after apply)
              + server_app_secret      = (sensitive value)
              + tenant_id              = (known after apply)
            }
        }

      + service_principal {
          + client_id     = "89db72c0-b039-42f3-a521-0910ce0459ca"
          + client_secret = (sensitive value)
        }

      + windows_profile {
          + admin_password = (sensitive value)
          + admin_username = (known after apply)
          + license        = (known after apply)
        }
    }

  # azurerm_resource_group.example will be created
  + resource "azurerm_resource_group" "example" {
      + id       = (known after apply)
      + location = "eastus"
      + name     = "finalproject-k8s-resources"
    }

Plan: 2 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + client_certificate     = (known after apply)
  + client_key             = (known after apply)
  + cluster_ca_certificate = (known after apply)
  + host                   = (known after apply)
  + id                     = (known after apply)
  + kube_config            = (sensitive value)
```



Step 2: Configure kubectl locally in order to communicate with Azure Kubernetes Server. Running ***terraform output kube_config*** to generate a sample kubectl configuration to connect to our cluster

Running the following commands:

```
terraform output kube_config > ~/.kube/config-terraform-aks-demo
cp ~/.kube/config-terraform-aks-demo ~/.kube/config
```

```
deepgamit@Deeps-MacBook-Pro AKS % terraform output kube_config
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUU2VENDQXRHZ0F3SUJBZ0lSQU83NnNTd1NpNmpKWms1ZHFXM0RFR3d3RFFZS
ktvWklodmNOQVFFTEJRQXcKRFRFTE1Ba0dBMVVFQXhNQ1kyRXdJQmNOTWpFeE1qRTJNVE15TXpVNVdoZ1BNakExTVRFeU1UWXhNek16TlRsYQpNQTB4Q3pBSkJnTlZCQU1UQW
10aE1JSUNJakFOQmdrcWhraUc5dzBCQVFFRkFBT0NBZzhBTUlJQ0NnS0NBZ0VBCjNZRHVqdXBuTjVjNE9DM3YwaFppT25lOUpVNURGcnd4cUJPTXhvajN3a3pxL1MzS01NLzV
VS2FITzhhcWlvd0YKVTNORjMzRjZ5RFovYnQwakNLTFVlZ1Y0WDdWWEpUL2srWGtaSnM5aklRRHRoZzJqQW9ZVUpJMWpBMmFkQzc2WgpyNkx2cW5hRU5zV2FkQVI5a0o5R1JG
YlVoRE14eE1MODhLRXV2bFBoK3NOOW1xL09HTFZLUmR1ekZKQXFuMk9QCkxYV0g0RXlaT2JsMlFWY0hpVjdIWllpQVJMVHJ0RVVreUVZZjIxZU8rTy9VRDFDZ0wzL2JqL3RKR
FNmYXRuc2sKcm1kNGhlQkxjU3EyMzNhVWw0Vk84eGN6eWtlVEkwYVdLNVpIUExoaUR4TlFZVE16ZkRFcm5IRGkyZFRNZlFRMgpOWWxCZmxRa04xRzNYZE8wUWUvQU43VnZlb1
dxa0pod3o4emN3cEhCUTVRS0lzTkcxN3hIQjFyUit4aVJkMVVsCjFnUTlMcWlIMEFzZlhSQnlQVVJUYnQyTFJHc2pVa0pmd2J5RWpZQkpSUWEzdFZ0WGdrUlZXRU00QnRCeHR
xbmoKNzJIMitEREM0cEExb0NNWUsxRVJucnF3UGNNYUVZQXdtcTZRMHAxZk5OSXByYUs4Wll4VVkvcUVLUThrdTNnMAp3allQT05IZTdCYi9QUUFDS2tVZUF3aVQ4MDhrSnQx
L1oyU2RyeGFRTGYvdFVWMUtvUHZROXNXRTVDSlNZSDZ5Cnk3SG5qcTZpdXNQTFdySHV3a29Qa3JVRk0xWWFJdWs4YWVhUnJUTzRyS2t6dTVrWVVoMTh6eUhhQnZTV1k4QzEKR
VJpQURPZG9CUEhZb01KVUVYMjVZSnA0VGQ4YWltUzgwVkRkYW5sNjlHVUNBd0VBQWFOQ01FQXdEZ1lEVlIwUApBUUgvQkFREEFnS2tNQThHQTFVZEV3RUIvd1FGTUFNQkFmOH
dIUVlEVlIwT0JCWUVGS1BLVXBEOXEveWkrYUZiCmNHMURKN1BuVVFKVU1BMEdDU3FHU0liM0RRRUJDd1VBQTRJQ0FRQ2x6ekdmOTI2SW1FeTBjWlZsdU9XQThBZkcKQzJxR09
yNTBacUc5ZlpEaFlUbmE0by9nTlhkR2h6K0FnMTRWbDBaVGRUNU8yUm0zVjdMRTFwSG9heXozZ1RZOApCaG5vdzZESll2dnl6aFdDaERqT21qd296ckdLaDGRCYTdRbEFBaTM1
Qjc2TWptcFU3RzVNaTRCSzdycHRRcGdiCjBmQ0NGaUZSeXFTMUw0VDFDbjl6QU9JZUcrYUxCdDM4M3VoQzBDaXBNBW9xSjAyVUprcVpPZFh3bWRxYk9RNjUKdEh5N2JwVEFxM
2wwZ0JXaHltVU03KzJZbWxLVmlZZzRucmZWUW0wVk1sQ1htaFJOQkdMVTFLYnhmMzQwUy95Mwp3VjFaQi9XSkJEbVZWQ1VydHF3VHdxWDA2aUc1TzVidzErc3RaU1ZtV3dDSj
JyVVh1cnM5eHIxZUJSUkdRb1d1CjdVdDhYYWRYeGZOSkVDM0NUYzV1RnFHeFRSL01ZTFA2RFFtMkVzUndMOEtOSk1sYXVTUEpxd05hRnNqOUFrbEoKZlg2cExqUG9WNnVhUDl
VNVdTRDVBcVZTTnpQUG44RVZXYzVRWkF6cnQxejJPUVFHWUNBb0ZCQUZSYlpFcEd2KworUmR6WXFPUGNrVTg4NnVuT3dMY2NFcFdvMktIL1pBbFZ0WGMxaUxEaDlRZkFQV01T
UFVwUE41WlhjZFpMQmdhCk9iQkhPU1RGWE5YSWdFQUFCK2ZtZ05zN1FybnhLUFdNM1hkL0VWVmd0eURQQjhzRlB4S0lTUVBKYXJnVytDWGgKS2ZoU1dNYTlNTTd2MlVZVHRFd
lhkbThFK1RqVkZjS1dXY3prNWdUbEhwSzN1TnphRGRZaDUvRG11SDgvdEpITApaNkJXSEJ4ZjY4KzhEbFdqNWc9PQotLS0tLUVORCBDRVJUSUZJQ0FURS0tLS0tCg==
    server: https://finalproject-k8s-eda01e98.hcp.eastus.azmk8s.io:443
  name: finalproject-k8s
contexts:
- context:
    cluster: finalproject-k8s
    user: clusterUser_finalproject-k8s-resources_finalproject-k8s
  name: finalproject-k8s
current-context: finalproject-k8s
kind: Config
preferences: {}
users:
- name: clusterUser_finalproject-k8s-resources_finalproject-k8s
  user:
    client-certificate-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUZIakNDQXdhZ0F3SUJBZ0lSQU5zb1p0ZzV1bEkvZFV3RzZhVHBqWEF3RFFZSktv
WklodmNOQVFFTEJRQXcKRFRFTE1Ba0dBMVVFQXhNQ1kyRXdIaGNOTWpFeE1qRTJNVE15TXpVNVdoY05Nak14TWpFMk1UTXpNelU1V2pBdWpNUmN3RlFZRFZRUUtFdzV6ZVhOM
FpXMDZiV0Z6ZEdWeWN6RVZNQk1HQTFVRUF4TU1iV0ZzEdWeVkyeHBaVzUwCk1JSUNJakFOQmdrcWhraUc5dzBCQVFFRkFBT0NBZzhBTUlJQ0NnS0NBZ0VBVBM3RlZFZRUUtFdzV6ZVhOM
kvSit6Uzd0VisKckoxcmZ4b2ZaMjJPRWFQQm5zazlsOWY4d0lWaklEaFFjQlRuWUZiWlo5YTlqMTNrb29mWEJaSllVV0hxa1QzQgpRR1pObWlyL3M3TGhqUmFhUC9BMENHaUl
yTEhzamFnRkR3M29GNjlQVHgvUk0wWDJUcjE1RGMrbjBRaGMwa25aCjl6TUtVT09QMXkxRm9tTWtkcHlXanVDUHVkQk9oUmpLSVpnYW16cDdHd3dXWnBpcDJIdWJTK0pRdnkz
bkZYOE4KVmVkUDN5MjllS0F1ULRRVW1SZExFZXNJZ1Y1d29jemlETVJqeWE2QStjVHpML2ZOVm4ySTRiYTN2QmQ1S2EyaQpGSHczU3kxdkxESXdVaVdaTnVIZEV1K0I4bmU3S
25tQ2VzRnYvRzhpc1djV3FyemdGQnBRQm42RHg5a01sdGlXCmVQTzRhMFRsUm8yNWNxOE1ML3Q2SFpGUlg4REVlYWdoUSs5QzdMRlZTOGQ5VjU1eTVab1laN0o4UTc1SlhYZ2
oKTFhLM3FWTFgvVk9NcVZoc08rckFHa1VNYUNQRGs3U0thM3QyT0hHdjJ4V2FWaGc1RFJVTVRzSDZadUUyZXdYdgpVNVFWVW8rdUsrR0xRcTkzc3JFS1R5akI5YTcxRHpmS0w
0aTNnUHVkQjJuUWN1Rkh5OUJIVDlmZmE2ZU83YWFvCnZyZGFMZnZoOGkzeHZTZ2RrVDBJNUcyb3JGNEg2V1NrQ1VFamE4V0RLRm0zcHRuSVI4YUd5b1ZrWkNiU0xZSjkKTzdX
WUdkNjFDckFBOGY3V0lTNjF6cktzWHpRSlMrYUFXVE8rZzljanFIdnBwRFlyWFNoUHRhWGZjdzRMcm04Lwo5ZUFKaGgrZGVLaC9icjRITyt1S0dBMENBd0VBQWFOV01GUXdEZ
1lEVlIwUEFRSC9CQVFEQWdXZ01CTUdBMVVkCkpRUU1NQW9HQ0NzR0FRVUZCd01DTUF3R0ExVWRFd0VCL3dRQ01BQXdId1lEVlIwakJCZ3dGb0FVbzhwU2tQMnIKL0tMNW9WdH
diVU1ucytkUkFsUXdEUVlKS29aSWh2Y05BUUVMQlFBRGdnSUJBSmZ3QjV4a2VSTG52bXNWK2Q4VQpHclg5aVJTY2taMldZa1dNd1BCcTdQdnZxTFdUdUV3cG5JRHczcEFaVDM
ybCtBMmRQb2EzeVF2MWFNZGRna3EzCnRBRHFTZHgvbVIwRWRqdDB5cmtRdi9GZVFJSHh5cHE0RHRWM3VDNWJqQ3orYnZDdXpQV1dpdHZFMnVWOG13WjAKMzdWWlkyNVpJYWgz
WG9NK2tlaytKNE05Vnc0VHRvN3hkWkc0ZmV3OC9HOHh2VmRZL3BCMWMvWitwdTMxWTZVRAo5eTVOS1NTTUtUMHE1V3lMOTRyQ2cwUDJ4Z0ljU0VWUTNIZzV1MFR4QWN5SDhIa
HRNeTNhRHlSSmdieXBzUWxjCmV4bVhRRTRDTTlrVnkwYm1iQ2wrTVBXRnlJdkgrZEJTY09SY3hjNmVHTjdYa01Fc2ZDNWdYdUVHZklJVmNiWDAKV3A5b0dFZXBDUUpEbi9DYj
ZsdTUzZUhBM2hybExFKzNJTVhyNmxlMTZWUjNZbWUwczFnMGs0S0lQcWE4NGxYagpRYVgyV1f3VGdSNHp1aG5EWko2em90ORkdQcDhMZUZPS2NkTmcxaGNzNldaMjVHZ0Z6SjN
QMmxYK0ZXc2dvcUF0CitZYWJHOjRwaVNXWncxb3VkYmVOZkRJODM1Wkw5TFVhWTdubXI3UnMxOUIxb1ZuTTkybVRvbE44L3luT0pJcUUKZzJYMU8wREtZQ1IxeTNod0IxbFBh
TXlycUdseXBrOXluTWxwdk1UbFFGMjF5NkU5bTgveGJJOEhKSVc2SUtrYgpYa2RQWXE5OFoySjE4dWptZjVHVFhFSGRxU2hrWFlzdFB2VlpDbjdqNmw0Y1cyS1BlZUx2TjhVc
```

Step 3: Verify that the worker nodes are joining the cluster:

```
kubectl get nodes
```

```
deepgamit@Deeps-MacBook-Pro AKS % kubectl get nodes
NAME                              STATUS   ROLES   AGE   VERSION
aks-default-24425734-vmss000000   Ready    agent   14m   v1.20.9
deepgamit@Deeps-MacBook-Pro AKS %
```

Check Resource utilisation:

```
kubectl top nodes
```

```
deepgamit@Deeps-MacBook-Pro AKS % kubectl top nodes
NAME                              CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
aks-default-24425734-vmss000000   200m         10%    773Mi           16%
deepgamit@Deeps-MacBook-Pro AKS %
```

Get all deployed pods:

```
kubectl get deploy --all-namespaces
```

```
deepgamit@Deeps-MacBook-Pro AKS % kubectl get deploy --all-namespaces
NAMESPACE     NAME                READY   UP-TO-DATE   AVAILABLE   AGE
kube-system   coredns             2/2     2            2           19m
kube-system   coredns-autoscaler  1/1     1            1           19m
kube-system   metrics-server      1/1     1            1           19m
kube-system   tunnelfront         1/1     1            1           19m
deepgamit@Deeps-MacBook-Pro AKS %
```

Step 4: Now that we have our cluster up and running on AKS, we deploy our rockstar app by combining the deployment and service files and applying:

```
kubectl apply -f rockstar-react-fe-combo.yml
```

```
deepgamit@Deeps-MacBook-Pro AKS % kubectl apply -f rockstar-react-fe-combo.yaml
deployment.apps/rockstar-react-fe created
service/rockstar-react-fe-lb created
deepgamit@Deeps-MacBook-Pro AKS %
```

Step 5: Get the deployment & service details:

```
kubectl describe deploy rockstar-react-fe
```

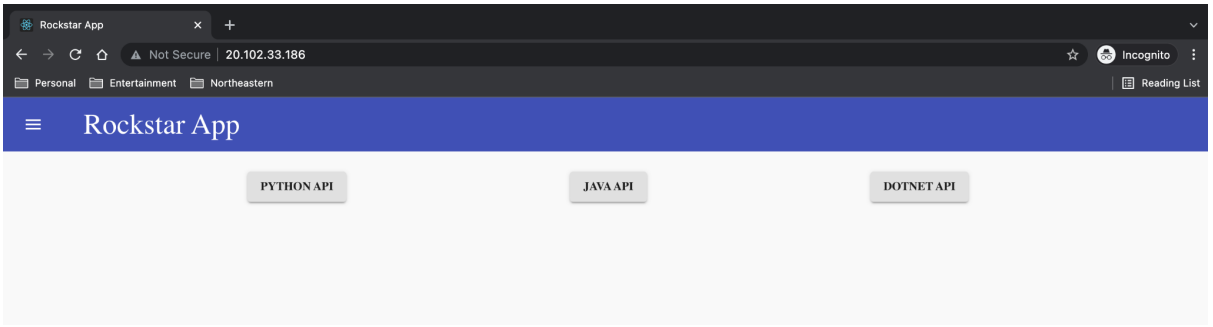
```
deepgamit@Deeps-MacBook-Pro AKS % kubectl describe deploy rockstar-react-fe


Name:                   rockstar-react-fe
Namespace:              default
CreationTimestamp:      Thu, 16 Dec 2021 08:58:37 -0500
Labels:                 <none>
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=rockstar-react-fe
Replicas:               1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        15
RollingUpdateStrategy:  1 max unavailable, 1 max surge
Pod Template:
  Labels:       app=rockstar-react-fe
  Annotations:  prometheus.io/path: /status/format/prometheus
                prometheus.io/port: 80
                prometheus.io/scrape: true
  Containers:
   rockstar-react-fe:
    Image:      deep2105/rockstar-react-fe:minikube
    Port:       80/TCP
    Host Port:  0/TCP
    Limits:
      cpu:  500m
    Requests:
      cpu:        200m
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   rockstar-react-fe-69ddcc89f7 (1/1 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  53s   deployment-controller  Scaled up replica set rockstar-react-fe-69ddcc89f7 to 1
```

```
kubectl get svc
kubectl describe svc rockstar-react-fe-lb
```

```
deepgamit@Deeps-MacBook-Pro AKS % kubectl get svc
NAME                TYPE           CLUSTER-IP    EXTERNAL-IP     PORT(S)        AGE
kubernetes          ClusterIP      10.0.0.1      <none>          443/TCP        30m
rockstar-react-fe-lb   LoadBalancer   10.0.10.103   20.102.33.186   80:32354/TCP   6m18s
deepgamit@Deeps-MacBook-Pro AKS %
```

```
deepgamit@Deeps-MacBook-Pro AKS % kubectl describe svc rockstar-react-fe-lb

Name:                     rockstar-react-fe-lb
Namespace:                default
Labels:                   <none>
Annotations:              <none>
Selector:                 app=rockstar-react-fe
Type:                     LoadBalancer
IP Families:              <none>
IP:                       10.0.10.103
IPs:                      10.0.10.103
LoadBalancer Ingress:     20.102.33.186
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  32354/TCP
Endpoints:                10.244.0.7:80
Session Affinity:         None
External Traffic Policy:  Cluster
Events:
  Type    Reason                Age    From                Message
  ----    ------                ----   ----                -------
  Normal  EnsuringLoadBalancer  9m44s  service-controller  Ensuring load balancer
  Normal  EnsuredLoadBalancer   9m27s  service-controller  Ensured load balancer
```

APPLICATION RUNNING ON IP: 20.102.33.186



Step 6: After deploying the application successfully, let's add horizontal autoscaling by creating a horizontal pod autoscaler file.

```
io.k8s.api.autoscaling.v1.HorizontalPodAutoscaler (v1@horizontalpodautoscaler.json)
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
 name: rockstar-react
spec:
 scaleTargetRef:
  apiVersion: apps/v1
  kind: Deployment
  name: rockstar-react-fe
 minReplicas: 1
 maxReplicas: 10
 targetCPUUtilizationPercentage: 5
```

Now create the auto scaler by:

```
kubectl create -f rockstar-app-scaler.yml
```

```
deepgamit@Deeps-MacBook-Pro AKS % kubectl create -f rockstar-app-scaler.yaml
horizontalpodautoscaler.autoscaling/rockstar-react created
deepgamit@Deeps-MacBook-Pro AKS %
```

Step 7: Now lets test if auto scaling works by running the following:

```
export SERVICE_IP=$(kubectl get svc rockstar-react-fe-lb
-o json | jq -r '.status.loadBalancer.ingress[].ip')
```

Here we export the service ip as an environment variable. Now we can use this variable for load testing with the following command:

```
ab -c 100 -n 100000 http://${SERVICE_IP}/
```

```
deepgamit@Deeps-MacBook-Pro AKS % ab -c 100 -n 100000 http://${SERVICE_IP}/


This is ApacheBench, Version 2.3 <$Revision: 1879490 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 20.102.33.186 (be patient)
Completed 10000 requests
Completed 20000 requests
```

Step 8: Check the stats with:

```
kubectl get hpa rockstar-react
```

```
deepgamit@Deeps-MacBook-Pro AKS % kubectl get hpa rockstar-react

NAME             REFERENCE                       TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
rockstar-react   Deployment/rockstar-react-fe   0%/5%     1         10        1          2m42s
deepgamit@Deeps-MacBook-Pro AKS %
```

```
deepgamit@Deeps-MacBook-Pro AKS % kubectl get hpa
NAME             REFERENCE                       TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
rockstar-react   Deployment/rockstar-react-fe   13%/5%    1         10        3          55m
deepgamit@Deeps-MacBook-Pro AKS %
```

```
deepgamit@Deeps-MacBook-Pro AKS % kubectl get hpa
NAME             REFERENCE                       TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
rockstar-react   Deployment/rockstar-react-fe   4%/5%     1         10        3          56m
deepgamit@Deeps-MacBook-Pro AKS % kubectl get hpa
NAME             REFERENCE                       TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
rockstar-react   Deployment/rockstar-react-fe   5%/5%     1         10        3          57m
deepgamit@Deeps-MacBook-Pro AKS % kubectl get hpa
NAME             REFERENCE                       TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
rockstar-react   Deployment/rockstar-react-fe   5%/5%     1         10        3          57m
deepgamit@Deeps-MacBook-Pro AKS % kubectl get hpa
NAME             REFERENCE                       TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
rockstar-react   Deployment/rockstar-react-fe   5%/5%     1         10        3          58m
deepgamit@Deeps-MacBook-Pro AKS %
```

Step 9: *Now we are ready to deploy prometheus for monitoring. Let's create a different namespace to separate monitoring pods from k8s app pods.*

```
kubectl create namespace monitoring
```

```
deepgamit@Deeps-MacBook-Pro AKS % kubectl create namespace monitoring

namespace/monitoring created
deepgamit@Deeps-MacBook-Pro AKS %
```

Step 10: For Prometheus to fetch metrics from Kubernetes' APIs, we will assign the cluster reader permission. For that, we create a cluster role:

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: prometheus
rules:
- apiGroups: [""]
  resources:
  - nodes
  - nodes/proxy
  - services
  - endpoints
  - pods
  verbs: ["get", "list", "watch"]
- apiGroups:
  - extensions
  resources:
  - ingresses
  verbs: ["get", "list", "watch"]
- nonResourceURLs: ["/metrics"]
  verbs: ["get"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: prometheus
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: prometheus
subjects:
- kind: ServiceAccount
  name: default
  namespace: monitoring
```

Create permissions with the following command:

```
kubectl create -f clusterRole.yaml
```

```
deepgamit@Deeps-MacBook-Pro AKS % kubectl create -f clusterRole.yaml
clusterrole.rbac.authorization.k8s.io/prometheus created
clusterrolebinding.rbac.authorization.k8s.io/prometheus created
deepgamit@Deeps-MacBook-Pro AKS %
```

Step 11: After giving proper permissions, we create a config map for our cluster with prometheus scrape config which will be mounter to Prometheus container in `/etc/prometheus` as files `prometheus.yml` and `prometheus.rules`

```
io.k8s.api.core.v1.ConfigMap (v1@configmap.json)
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-server-conf
  labels:
    name: prometheus-server-conf
  namespace: monitoring
data:
  prometheus.rules: |-
    groups:
    - name: devopscube demo alert
      rules:
      - alert: High Pod Memory
        expr: sum(container_memory_usage_bytes) > 1
        for: 1m
        labels:
          severity: slack
        annotations:
          summary: High Memory Usage
  prometheus.yml: |-
    global:
      scrape_interval: 5s
      evaluation_interval: 5s
    rule_files:
      - /etc/prometheus/prometheus.rules
    alerting:
      alertmanagers:
      - scheme: http
        static_configs:
        - targets:
          - "alertmanager.monitoring.svc:9093"

    scrape_configs:
      - job_name: 'kubernetes-apiservers'

        kubernetes_sd_configs:
        - role: endpoints
        scheme: https

        tls_config:
          ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
        bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
```

Create config map with the following command:

```
kubectl create -f config-map.yaml
```

```
deepgamit@Deeps-MacBook-Pro AKS % kubectl create -f config-map.yaml
configmap/prometheus-server-conf created
deepgamit@Deeps-MacBook-Pro AKS %
```

Step 12: Create Prometheus deployment after creating the config map

```
io.k8s.api.apps.v1.Deployment (v1@deployment.json)
apiVersion: apps/v1
kind: Deployment
metadata:
  name: prometheus-server
  namespace: monitoring
spec:
  selector:
    matchLabels:
      app: prometheus-server
  replicas: 1
  template:
    metadata:
      labels:
        app: prometheus-server
    spec:
      containers:
        - name: prometheus
          image: prom/prometheus:v2.12.0
          args:
            - "--config.file=/etc/prometheus/prometheus.yml"
            - "--storage.tsdb.path=/prometheus/"
          ports:
            - containerPort: 9090
          volumeMounts:
            - name: prometheus-config-volume
              mountPath: /etc/prometheus/
            - name: prometheus-storage-volume
              mountPath: /prometheus/
      volumes:
        - name: prometheus-config-volume
          configMap:
            defaultMode: 420
            name: prometheus-server-conf

        - name: prometheus-storage-volume
          emptyDir: {}
```

Deploy on monitoring namespace using prometheus-deployment file:

```
kubectl create -f prometheus-deployment.yaml
-namespace=monitoring
```

```
deepgamit@Deeps-MacBook-Pro AKS % kubectl create -f prometheus-deployment.yaml --namespace=monitoring
deployment.apps/prometheus-server created
deepgamit@Deeps-MacBook-Pro AKS %
```

Step 13: Checking all the deployments

```
kubectl get deployments --all-namespaces
```

```
deepgamit@Deeps-MacBook-Pro AKS % kubectl get deployments --all-namespaces
NAMESPACE     NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
default       rockstar-react-fe     1/1     1            1           38m
kube-system   coredns               2/2     2            2           61m
kube-system   coredns-autoscaler    1/1     1            1           61m
kube-system   metrics-server        1/1     1            1           61m
kube-system   tunnelfront           1/1     1            1           61m
monitoring    prometheus-server     1/1     1            1           91s
deepgamit@Deeps-MacBook-Pro AKS %
```

Step 14: Let's expose Prometheus as a service so that we can access the Prometheus dashboard over an IP

```
io.k8s.api.core.v1.Service (v1@service.json)
apiVersion: v1
kind: Service
metadata:
  name: prometheus-service
  namespace: monitoring
  annotations:
      prometheus.io/scrape: 'true'
      prometheus.io/port:   '9090'
spec:
  selector:
    app: prometheus-server
  type: LoadBalancer
  ports:
    - port: 80
      targetPort: 9090
```

Create service endpoint with the following command:

```
kubectl create -f prometheus-service.yaml
--namespace=monitoring
```

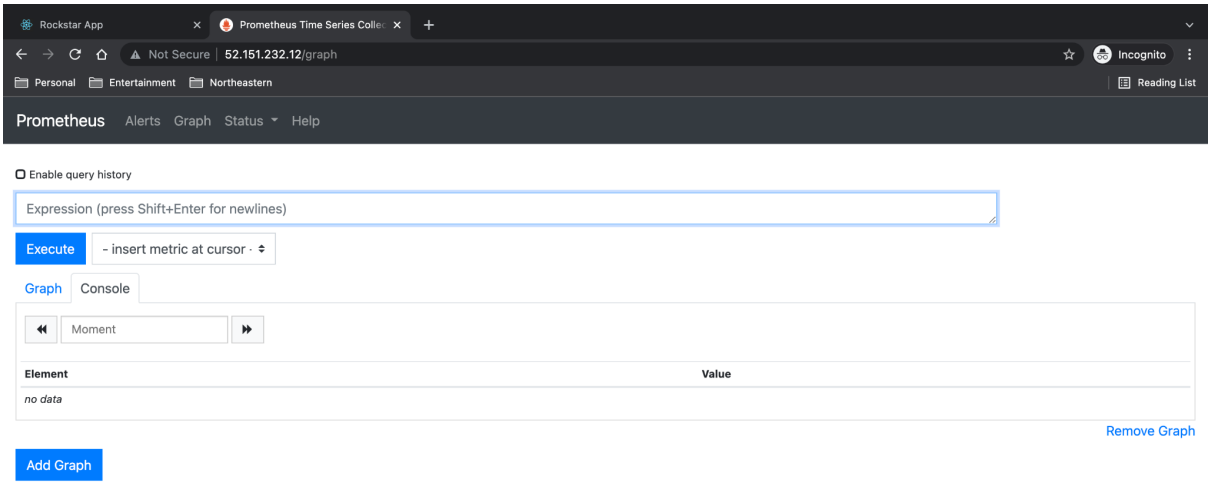
```
deepgamit@Deeps-MacBook-Pro AKS % kubectl create -f prometheus-service.yaml --namespace=monitoring
service/prometheus-service created
deepgamit@Deeps-MacBook-Pro AKS %
```

Step 15: Get the IP of Prometheus service with:

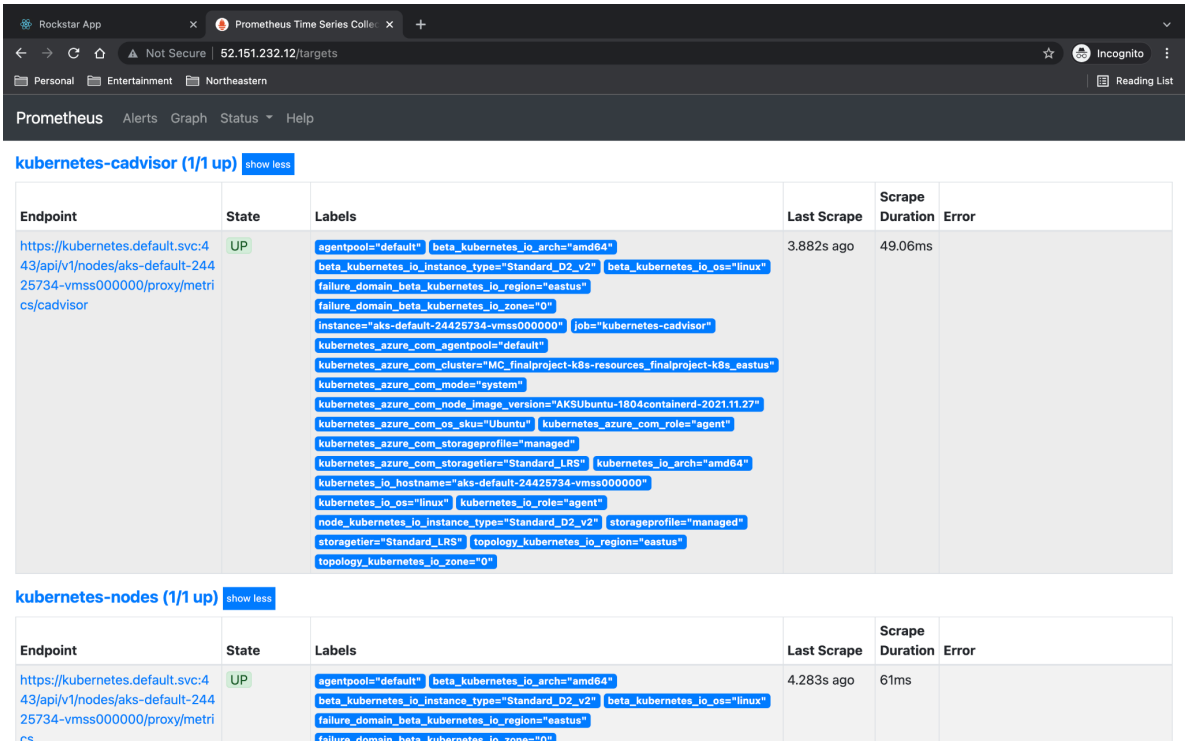
```
kubectl get svc --namespace=monitoring
```

```
deepgamit@Deeps-MacBook-Pro AKS % kubectl get svc --namespace=monitoring
NAME                 TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)        AGE
prometheus-service   LoadBalancer   10.0.158.166   52.151.232.12   80:31301/TCP   102s
deepgamit@Deeps-MacBook-Pro AKS %
```

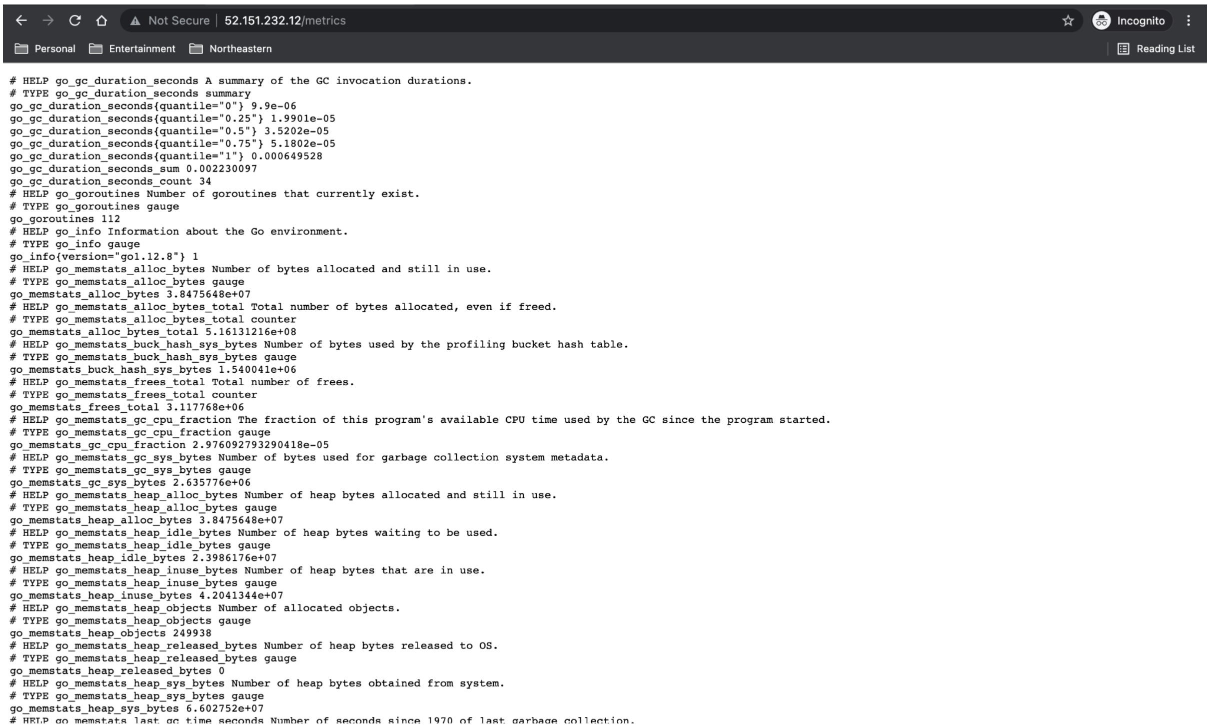
PROMETHEUS DASHBOARD IP: 52.151.232.12



TARGETS

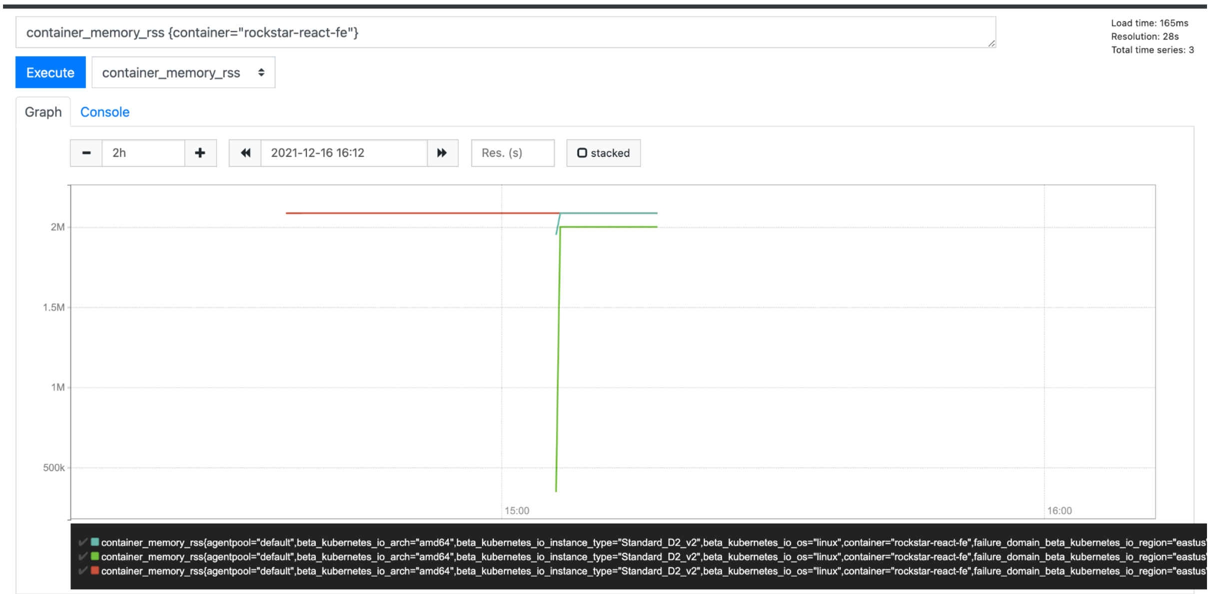


SEE ALL METRICS: 52.151.232.12/metrics



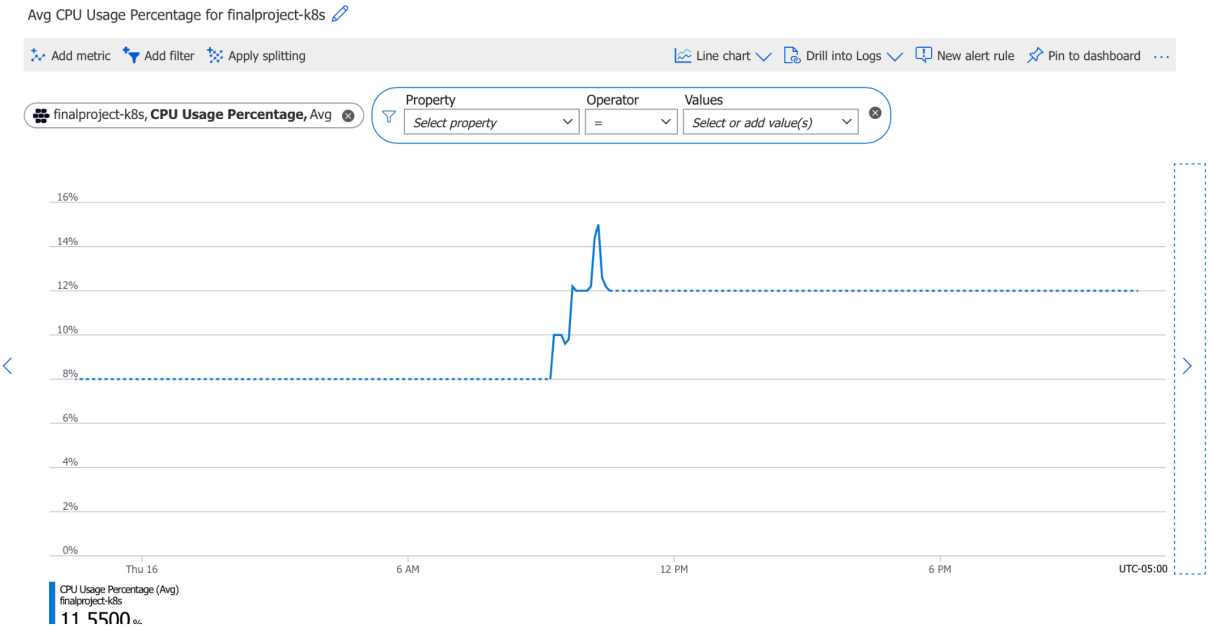

```
# HELP go_gc_duration_seconds A summary of the GC invocation durations.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 9.9e-06
go_gc_duration_seconds{quantile="0.25"} 1.9901e-05
go_gc_duration_seconds{quantile="0.5"} 3.5202e-05
go_gc_duration_seconds{quantile="0.75"} 5.1802e-05
go_gc_duration_seconds{quantile="1"} 0.000649528
go_gc_duration_seconds_sum 0.002230097
go_gc_duration_seconds_count 34
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 112
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.12.8"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 3.8475648e+07
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 5.16131216e+08
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 1.540041e+06
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 3.117768e+06
# HELP go_memstats_gc_cpu_fraction The fraction of this program's available CPU time used by the GC since the program started.
# TYPE go_memstats_gc_cpu_fraction gauge
go_memstats_gc_cpu_fraction 2.976092793290418e-05
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata.
# TYPE go_memstats_gc_sys_bytes gauge
go_memstats_gc_sys_bytes 2.635776e+06
# HELP go_memstats_heap_alloc_bytes Number of heap bytes allocated and still in use.
# TYPE go_memstats_heap_alloc_bytes gauge
go_memstats_heap_alloc_bytes 3.8475648e+07
# HELP go_memstats_heap_idle_bytes Number of heap bytes waiting to be used.
# TYPE go_memstats_heap_idle_bytes gauge
go_memstats_heap_idle_bytes 2.3986176e+07
# HELP go_memstats_heap_inuse_bytes Number of heap bytes that are in use.
# TYPE go_memstats_heap_inuse_bytes gauge
go_memstats_heap_inuse_bytes 4.2041344e+07
# HELP go_memstats_heap_objects Number of allocated objects.
# TYPE go_memstats_heap_objects gauge
go_memstats_heap_objects 249938
# HELP go_memstats_heap_released_bytes Number of heap bytes released to OS.
# TYPE go_memstats_heap_released_bytes gauge
go_memstats_heap_released_bytes 0
# HELP go_memstats_heap_sys_bytes Number of heap bytes obtained from system.
# TYPE go_memstats_heap_sys_bytes gauge
go_memstats_heap_sys_bytes 6.602752e+07
# HELP go_memstats_last_gc_time_seconds Number of seconds since 1970 of last garbage collection.
```

METRICS OF INTEREST ON PROMETHEUS

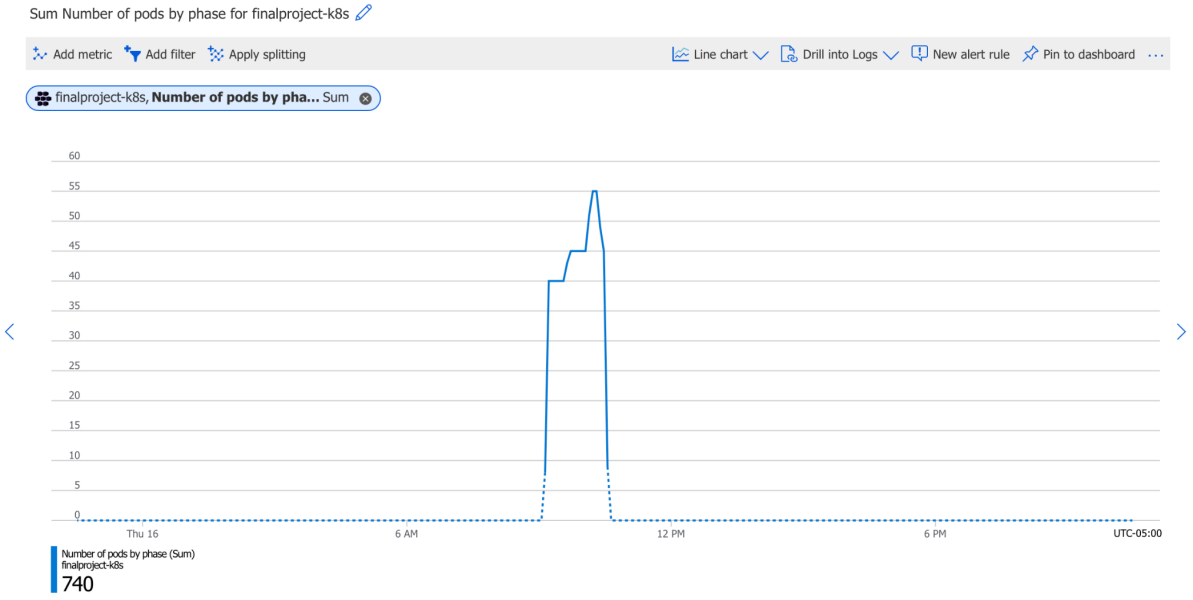


METRICS ON AZURE

Average CPU Usage Percentage



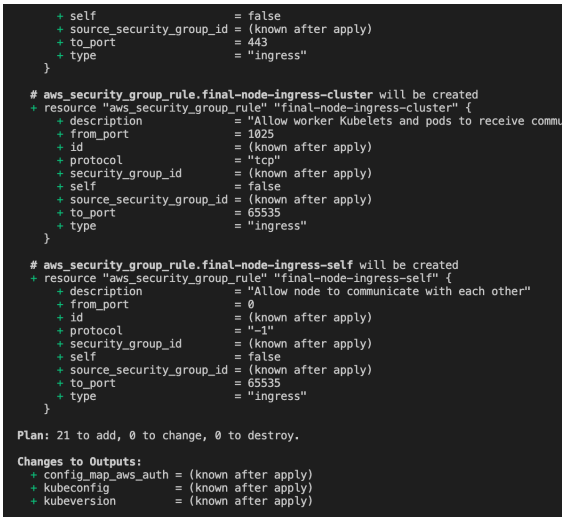
Sum Of Number of Pods in Phase

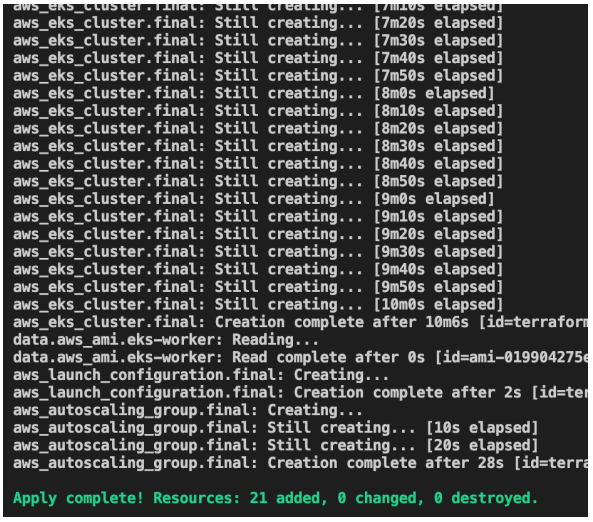


Deploying the Rockstar App to EKS cluster with CloudWatch for Monitoring

Step 1: We deploy the resources using terraform on AWS Elastic Kubernetes Service by applying the following commands:

```
terraform init
terraform plan
terraform apply
```





Step 2: Configure kubectl locally in order to communicate with AWS Elastic Kubernetes Server. Running ***terraform output kubeconfig*** to generate a sample kubectl configuration to connect to our cluster

Running the following commands:

```
terraform output kubeconfig > ~/.kube/config-terraform-eks-demo
cp ~/.kube/config-terraform-eks-demo ~/.kube/config
```

```
Outputs:

config_map_aws_auth =

apiVersion: v1
kind: ConfigMap
metadata:
  name: aws-auth
  namespace: kube-system
data:
  mapRoles: |
    - rolearn: arn:aws:iam::658346791705:role/terraform-eks-final-node
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes

kubeconfig =

apiVersion: v1
clusters:
- cluster:
    server: https://5FCD16FF626CF20BF7E6E8C1B91F289B.gr7.us-east-1.eks.amazonaws.com
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUM1ekNDQWMrZ0F3SUJBZ0lCQURBTkJna3Foa2lHOXcwQkFRc0Z
BREFWTVJNd0VRWURWUVFERXdwcmRXSmwKY201bGRHVnpNQjRYRFRJeE1USXhOakUzTURjME5Gb1hEVE14TVRJeE5ERTNNRGMwTkZvd0ZURVRNQkVHQTFVRQpBeE
1LYTNWaVpYSnVaWFJsY3pDQ0FTSXdEUVlKS29aSWh2Y05BUUVCQlFBRGdnRVBBRENDQVFvQ2dnRUJBTFVBCm5nUnlQdHNSSFpqUDVRSTQzNmFYOElvb2REUUQ4Q
lRRZXNBU0dSTk5pUGh0MGQ4MXdzY1ZnTnMvSk1lTTRyUTQKdjdkNko1Q25XV1NBMjVWSFhJYVZYMWsvdmpMZFhHcnNFTWNrUjNlcndOVytDZXlzM3hSY2ROU21J
djg4aEJ4bQpuVHdBTXkvcGcyRlNaUGFYekQrM3gzWFpSQ0FyWFpCQ3dZTnp4bkZsbTNrTlpJWnBiRWw0MEJCRWFiN2RabFhjCmFhWGM2QjQ3dGtoRGd3Sm1VSWR
YclJQTHEwV3BjUTA1aTNhYjM4Ui9QemZoa3ZZNEcxV1MrcmZ5WWhodmhGVlIKMTFxbUVxU21rSlRVSnN6cXhOYWQ3aHhYQlpKdVVXSURJUkJtYXNVWGprVjdMWW
RYMWVkYUIwUG92Q1kwc0RDYwpqMkNOY29vTGphQ1RSbk1WeXpFQ0F3RUFBYU5DTUVBd0RnWURWUjBQQVFIL0JBUURBZ0trTUE4R0ExVWRFd0VCCi93UUZNQU1CQ
WY4d0hRWURWUjBPQkJZRUZMYVMvbmxMNTVpVnB6REMzQm5xdzBuWlNhWkhNQTBHQ1NxR1NJYjMKRFFFQkN3VUFBNElCQVFBWnZFaFlzTlBjMGhMVnR5Y3ZRNGFw
aGdJRDNVVEhuWFhqUU9yZ0RQWVlGeVFCdmc5NgptSWo1Q2phaVZ5OTYzRGVVaDBua21RNFF1d25NMW83T0JLOTNXNVJCd2JlOFBhWDZKbWhQdG02emFDTXlDTyt
xCmhNem9ya201Rzg3UUpmNlF5VHZNN0JLclNYUnJrUHlQQmN5L29NdE1rL3pUOVJKd2FqRXFMQzJrZUdlRkF0MEYKNUVuTlZTbDBzOUhwSTR0L2hqdmJ1VDBJQ0
45Z2g1bDNSNTZQK2NreVZOd21YU0k5Zm16cExUbGRXYUpUTnE2awo3WnZMSVJjcDRmMm5SdU0ycnhJeWN4S3dKMVFhU2FxRkNVMzRJS09kK003RFV5bm1wT1JRT
0paNFJTYUpUdmM3CjdjUnlZZU5VWVNsUDVUelVScUhiR0tnRzNEVHFMWTRlVDJvLwotLS0tLUVORCBDRVJUSUZJQ0FURS0tLS0tCg==
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: aws
  name: aws
current-context: aws
kind: Config
preferences: {}
users:
- name: aws
  user:
    exec:
      apiVersion: client.authentication.k8s.io/v1alpha1
      command: aws-iam-authenticator
      args:
        - "token"
        - "-i"
```

Step 3: Verify that the worker nodes are joining the cluster:

```
kubectl get nodes
```

```
deepgamit@Deeps-MacBook-Pro EKS % kubectl get nodes
NAME                           STATUS   ROLES    AGE   VERSION
ip-10-0-0-216.ec2.internal   Ready    <none>   66s   v1.21.5-eks-bc4871b
ip-10-0-1-176.ec2.internal   Ready    <none>   65s   v1.21.5-eks-bc4871b
deepgamit@Deeps-MacBook-Pro EKS %
```

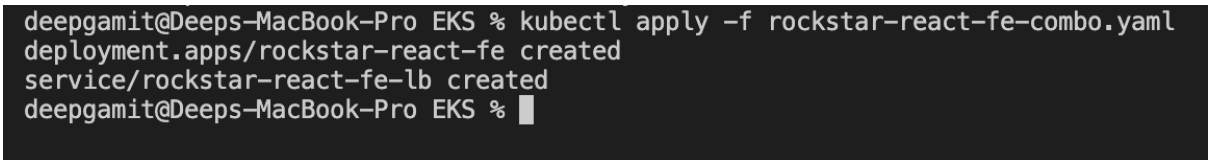
Get all deployed pods:

```
kubectl get deploy --all-namespaces
```

```
deepgamit@Deeps-MacBook-Pro EKS % kubectl get deploy --all-namespaces
NAMESPACE     NAME      READY   UP-TO-DATE   AVAILABLE   AGE
kube-system   coredns   2/2     2            2           9m42s
deepgamit@Deeps-MacBook-Pro EKS %
```

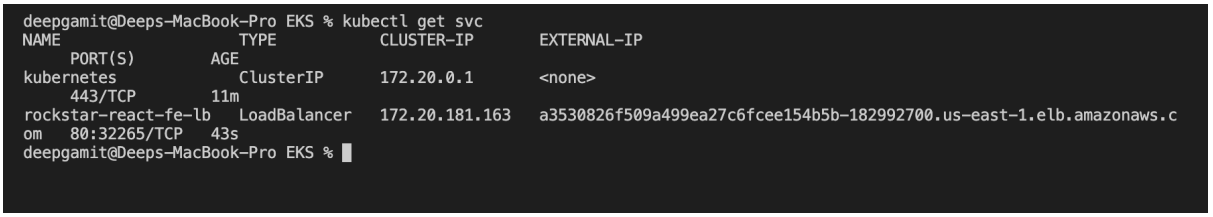
Step 4: Now that we have our cluster up and running on EKS, we deploy our rockstar app by combining the deployment and service files and applying:

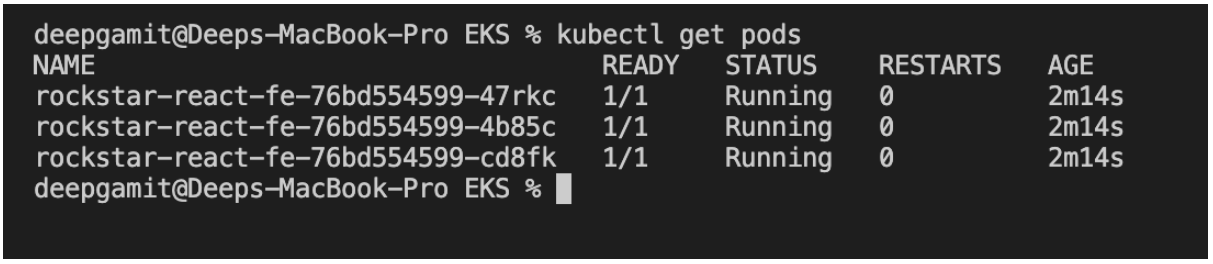
```
kubectl apply -f rockstar-react-fe-combo.yaml
```



Step 5: Get service details:

```
kubectl get svc
kubectl get pods
```





APPLICATION RUNNING ON IP: a3530826f509a499ea27c6fcee154b5b-182992700.us-east-1.elb.amazonaws.com

Setup CloudWatch Agent to collect metrics

Step 6: Create a namespace for CloudWatch:

```
kubectl apply -f cwagent-namespace.yaml
kubectl get namespaces
```

```
deepgamit@Deeps-MacBook-Pro EKS % kubectl apply -f cwagent-namespace.yaml
namespace/amazon-cloudwatch created
deepgamit@Deeps-MacBook-Pro EKS % kubectl get namespaces
NAME                STATUS   AGE
amazon-cloudwatch   Active   9s
default             Active   16m
kube-node-lease     Active   16m
kube-public         Active   16m
kube-system         Active   16m
deepgamit@Deeps-MacBook-Pro EKS %
```

Step 7: Create a service account in the cluster with ClusterRole & ClusterRoleBinding

```
kubectl apply -f cwagent-serviceaccount.yaml
```

```
deepgamit@Deeps-MacBook-Pro EKS % kubectl apply -f cwagent-serviceaccount.yaml
serviceaccount/cloudwatch-agent created
clusterrole.rbac.authorization.k8s.io/cloudwatch-agent-role created
clusterrolebinding.rbac.authorization.k8s.io/cloudwatch-agent-role-binding created
deepgamit@Deeps-MacBook-Pro EKS %
```

```yaml
# create cwagent service account and role binding
apiVersion: v1
kind: ServiceAccount
metadata:
  name: cloudwatch-agent
  namespace: amazon-cloudwatch

---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: cloudwatch-agent-role
rules:
  - apiGroups: [""]
    resources: ["pods", "nodes", "endpoints"]
    verbs: ["list", "watch"]
  - apiGroups: ["apps"]
    resources: ["replicasets"]
    verbs: ["list", "watch"]
  - apiGroups: ["batch"]
    resources: ["jobs"]
    verbs: ["list", "watch"]
  - apiGroups: [""]
    resources: ["nodes/proxy"]
    verbs: ["get"]
  - apiGroups: [""]
    resources: ["nodes/stats", "configmaps", "events"]
    verbs: ["create"]
  - apiGroups: [""]
    resources: ["configmaps"]
    resourceNames: ["cwagent-clusterleader"]
    verbs: ["get","update"]
```

```yaml
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: cloudwatch-agent-role
rules:
  - apiGroups: [""]
    resources: ["pods", "nodes", "endpoints"]
    verbs: ["list", "watch"]
  - apiGroups: ["apps"]
    resources: ["replicasets"]
    verbs: ["list", "watch"]
  - apiGroups: ["batch"]
    resources: ["jobs"]
    verbs: ["list", "watch"]
  - apiGroups: [""]
    resources: ["nodes/proxy"]
    verbs: ["get"]
  - apiGroups: [""]
    resources: ["nodes/stats", "configmaps", "events"]
    verbs: ["create"]
  - apiGroups: [""]
    resources: ["configmaps"]
    resourceNames: ["cwagent-clusterleader"]
    verbs: ["get","update"]

---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: cloudwatch-agent-role-binding
subjects:
  - kind: ServiceAccount
    name: cloudwatch-agent
    namespace: amazon-cloudwatch
roleRef:
  kind: ClusterRole
  name: cloudwatch-agent-role
  apiGroup: rbac.authorization.k8s.io
```

Step 8: Create a ConfigMap for the CloudWatch agent

```
kubectl apply -f cwagent-configmap.yaml
```

```
deepgamit@Deeps-MacBook-Pro EKS % kubectl apply -f cwagent-configmap.yaml
configmap/cwagentconfig created
deepgamit@Deeps-MacBook-Pro EKS %
```

```yaml
# create configmap for cwagent config
apiVersion: v1
data:
  # Configuration is in Json format. No matter what configure change you make,
  # please keep the Json blob valid.
  cwagentconfig.json: |
    {
      "logs": {
        "metrics_collected": {
          "kubernetes": {
            "cluster_name": "terraform-eks-final",
            "metrics_collection_interval": 60
          }
        },
        "force_flush_interval": 5
      }
    }
kind: ConfigMap
metadata:
  name: cwagentconfig
  namespace: amazon-cloudwatch
```

Step 9: Deploy the CloudWatch agent as a DaemonSet

```
kubectl apply -f cwagent-daemonset.yaml
```

```
deepgamit@Deeps-MacBook-Pro EKS % kubectl apply -f cwagent-daemonset.yaml
daemonset.apps/cloudwatch-agent created
deepgamit@Deeps-MacBook-Pro EKS %
```

```yaml
io.k8s.api.apps.v1.DaemonSet (v1@daemonset.json)
# deploy cwagent as daemonset
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: cloudwatch-agent
  namespace: amazon-cloudwatch
spec:
  selector:
    matchLabels:
      name: cloudwatch-agent
  template:
    metadata:
      labels:
        name: cloudwatch-agent
    spec:
      containers:
        - name: cloudwatch-agent
          image: amazon/cloudwatch-agent:1.247348.0b251302
          #ports:
          #  - containerPort: 8125
          #    hostPort: 8125
          #    protocol: UDP
          resources:
            limits:
              cpu:  200m
              memory: 200Mi
            requests:
              cpu: 200m
              memory: 200Mi
          # Please don't change below envs
          env:
            - name: HOST_IP
              valueFrom:
                fieldRef:
                  fieldPath: status.hostIP
            - name: HOST_NAME
              valueFrom:
```

```yaml
            - name: dockersock
              mountPath: /var/run/docker.sock
              readOnly: true
            - name: varlibdocker
              mountPath: /var/lib/docker
              readOnly: true
            - name: containerdsock
              mountPath: /run/containerd/containerd.sock
              readOnly: true
            - name: sys
              mountPath: /sys
              readOnly: true
            - name: devdisk
              mountPath: /dev/disk
              readOnly: true
      volumes:
        - name: cwagentconfig
          configMap:
            name: cwagentconfig
        - name: rootfs
          hostPath:
            path: /
        - name: dockersock
          hostPath:
            path: /var/run/docker.sock
        - name: varlibdocker
          hostPath:
            path: /var/lib/docker
        - name: containerdsock
          hostPath:
            path: /run/containerd/containerd.sock
        - name: sys
          hostPath:
            path: /sys
        - name: devdisk
          hostPath:
            path: /dev/disk/
      terminationGracePeriodSeconds: 60
      serviceAccountName: cloudwatch-agent
```

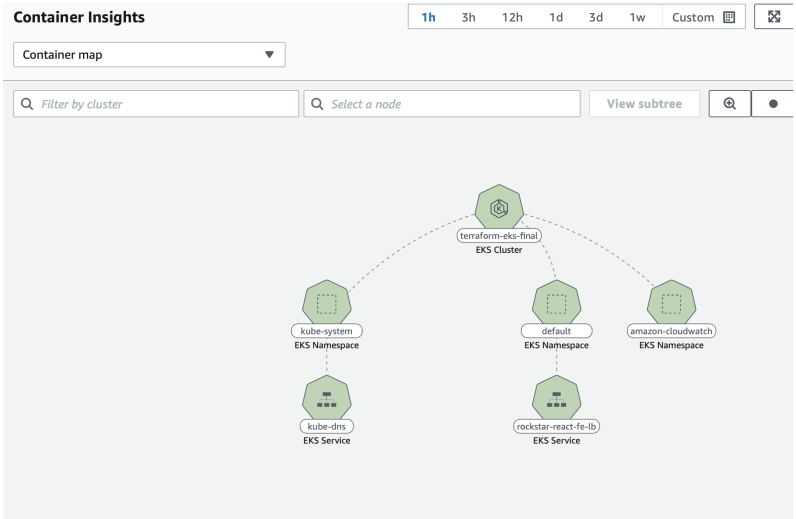
Step 10: Validate that the agent is deployed by running the following command.

```
kubectl get pods -n amazon-cloudwatch
```

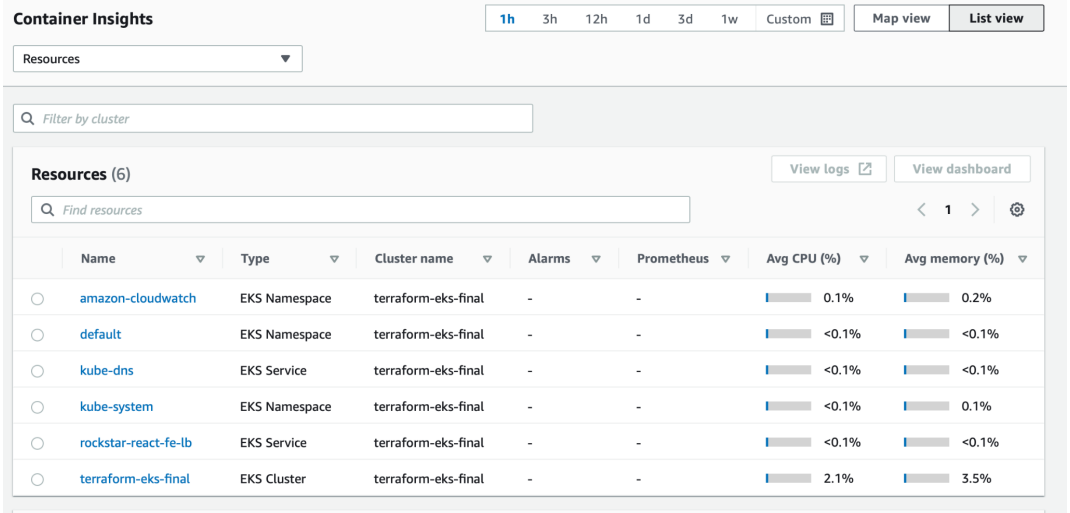
```
deepgamit@Deeps-MacBook-Pro EKS % kubectl get pods -n amazon-cloudwatch
NAME                     READY   STATUS    RESTARTS   AGE
cloudwatch-agent-9sglg   1/1     Running   0          24s
cloudwatch-agent-mvnmt   1/1     Running   0          24s
deepgamit@Deeps-MacBook-Pro EKS %
```

Viewing Container Insights Metrics

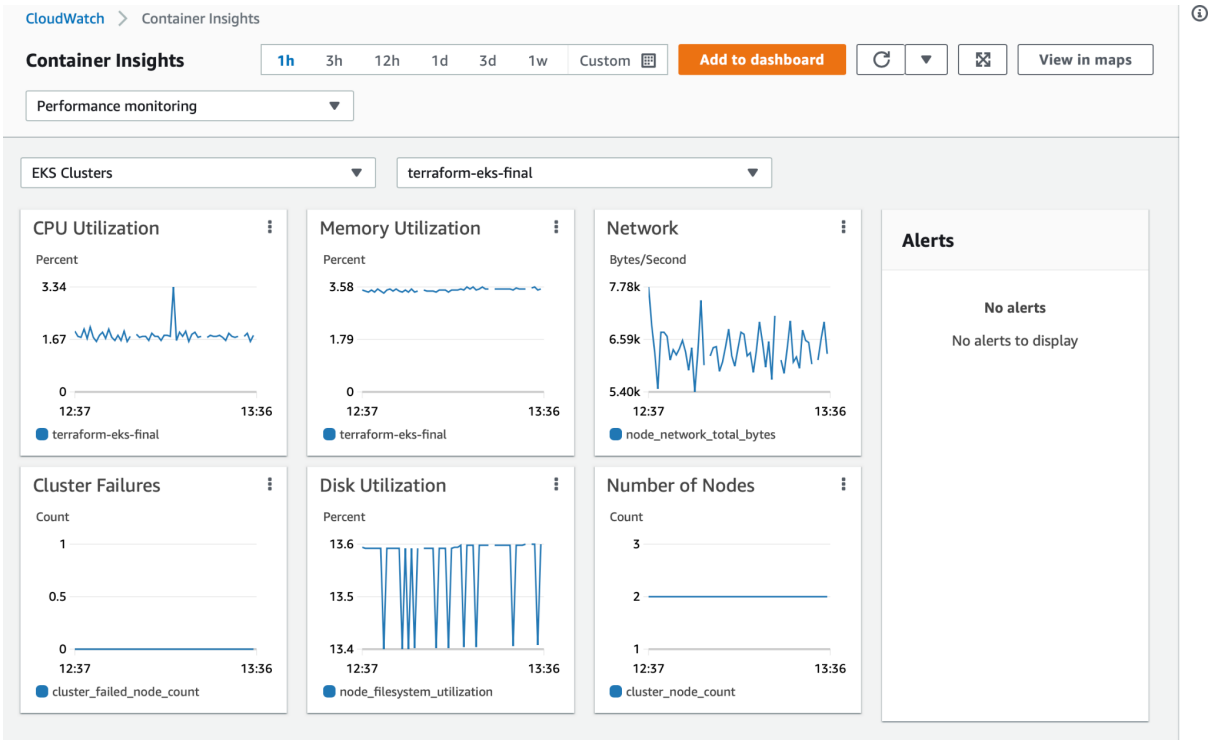
Container Map



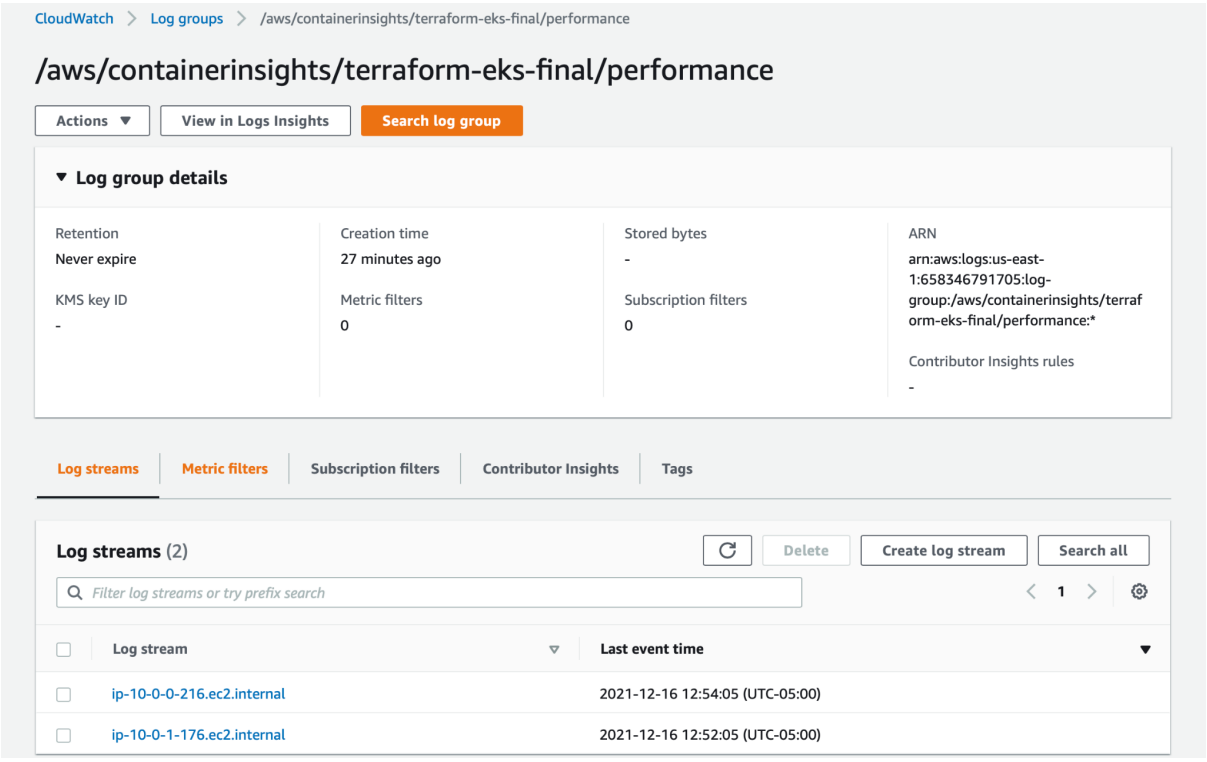
Resources



Performance Monitoring



Log Groups for Container Insights



Setup SNS Topic & CloudWatch Alarms

Step 1: Create a topic with name ***final-eks-sns***

```
deepgamit@Deeps-MacBook-Pro EKS % aws sns create-topic --name final-eks-sns
{
    "TopicArn": "arn:aws:sns:us-east-1:658346791705:final-eks-sns"
}
deepgamit@Deeps-MacBook-Pro EKS %
```

Step 2: Subscribe the topic to email address

```
deepgamit@Deeps-MacBook-Pro EKS % aws sns subscribe --topic-arn arn:aws:sns:us-east-1:658346791705:final-eks-sns --protocol
 email --notification-endpoint deepkgamit@gmail.com
{
    "SubscriptionArn": "pending confirmation"
}
deepgamit@Deeps-MacBook-Pro EKS %
```

Step 3: Confirm subscription

---

amazon web services — Simple Notification Service

Subscription confirmed!

You have successfully subscribed.

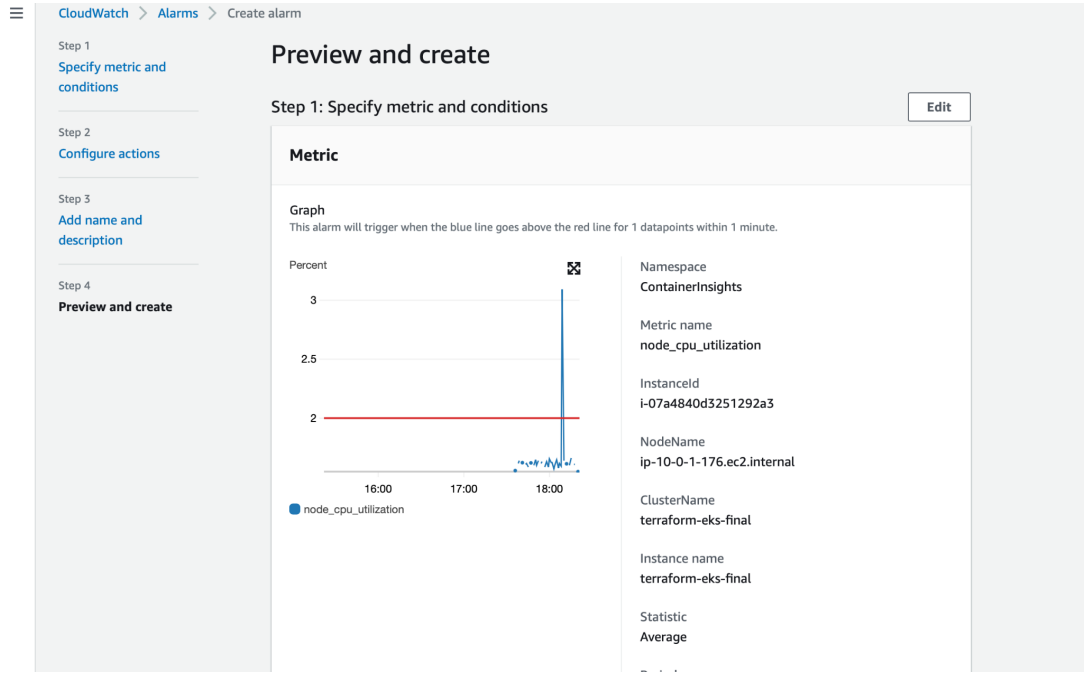
Your subscription's id is:
arn:aws:sns:us-east-1:658346791705:final-eks-sns:7a5fb202-04d8-4749-a9c5-68dc95b2e86b

If it was not your intention to subscribe, click here to unsubscribe.

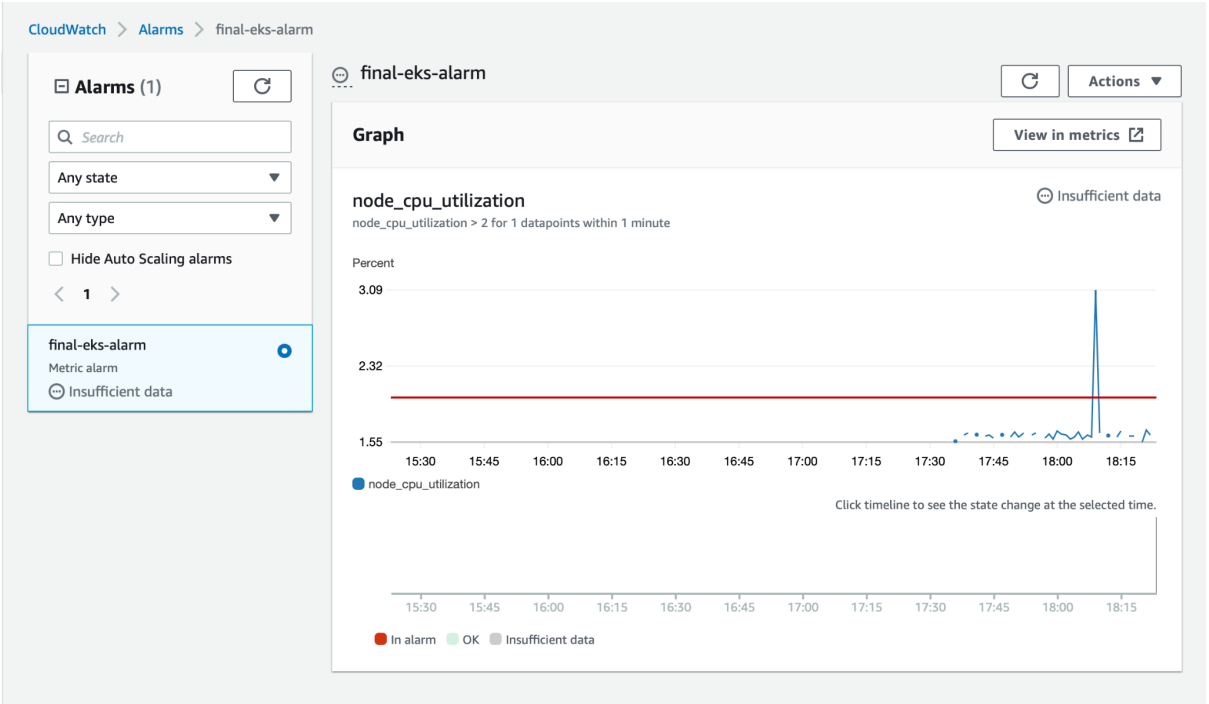
Step 4: Check subscription

```
deepgamit@Deeps-MacBook-Pro EKS % aws sns list-subscriptions-by-topic --topic-arn arn:aws:sns:us-east-1:658346791705:final-
eks-sns
{
    "Subscriptions": [
        {
            "SubscriptionArn": "arn:aws:sns:us-east-1:658346791705:final-eks-sns:7a5fb202-04d8-4749-a9c5-68dc95b2e86b",
            "Owner": "658346791705",
            "Protocol": "email",
            "Endpoint": "deepkgamit@gmail.com",
            "TopicArn": "arn:aws:sns:us-east-1:658346791705:final-eks-sns"
        }
    ]
}
deepgamit@Deeps-MacBook-Pro EKS %
```

Step 5: Setup CloudWatch Alarm to get notified when CPU Utilisation for cluster goes beyond a specific threshold.



Step 6: View CloudWatch Alarm (***final-eks-alarm***) on AWS Console

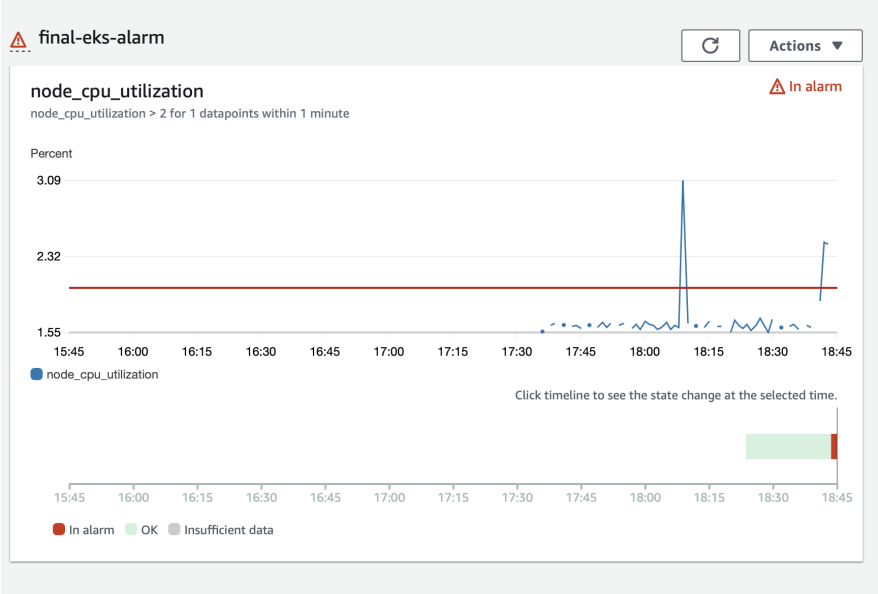


Testing SNS & CloudWatch Alarm

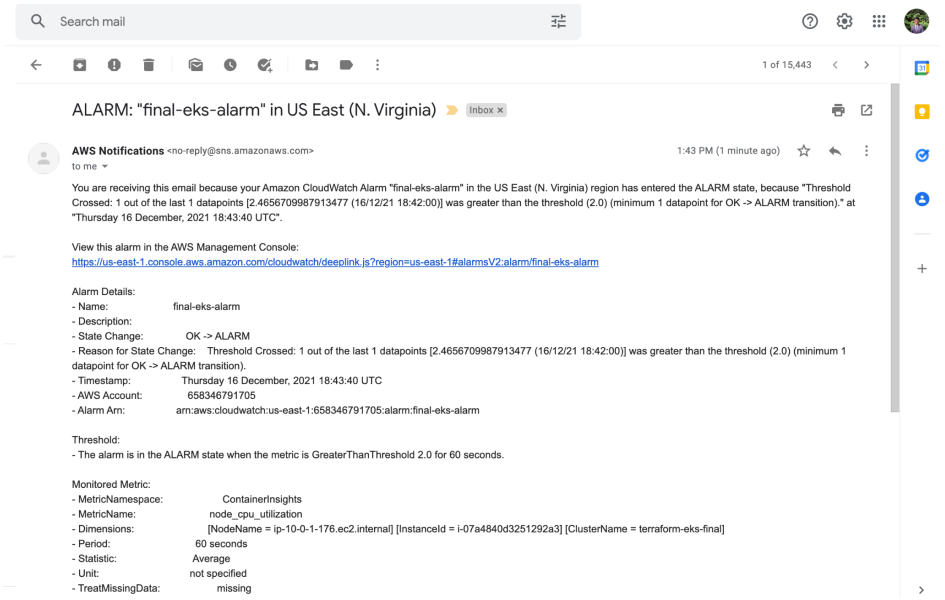
Step 1: Increased load on the app using ***AB*** tool

```
deepgamit@Deeps-MacBook-Pro EKS %  ab -c 50 -n 50000 http://a3530826f509a499ea27c6fcee154b5b-182992700.us-east-1.elb.amazonaws.com/
This is ApacheBench, Version 2.3 <$Revision: 1879490 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking a3530826f509a499ea27c6fcee154b5b-182992700.us-east-1.elb.amazonaws.com (be patient)
```

Step 2: Cloud Watch Alarm Console



Step 3: Check your mailbox for Email



CloudWatch Container Insights