Download Install Paks - 下載所有安裝資源及設定

==================================================

1. 安裝包下載位置: https://drive.google.com/file/d/1mwwOeyx6X7hmajlijBkegWvSorD-pFdy/view?usp=sharing

安裝包內容如下: Tubro監控平台/監控節點安裝Script

cd /

tar zxvf ibm-turbo-install-paks-20220719.tar.gz

解壓縮於根目錄/ibm資料夾結構如下說明

[root@localhost ibm]# ls -l -a

總計 0

drwxr-xr-x. 6 root root 80 7月 17 02:14 .

dr-xr-xr-x. 19 root root 251 7月 16 08:02 ..

drwxr-xr-x. 2 root root 6 7月 16 18:40 kubeturbo-script //監控節點安裝Script

drwxr-xr-x. 2 root root 196 7月 18 18:36 turbo-script //監控平台安裝Script

1. 防火牆開通連接:

外部: [Turbo 80, 443], 內部: [Kubeturbo 10250, 10255]

GCP/SaaS (port 443)

34.150.190.118

34.86.215.201

35.236.228.144

34.86.115.171

34.86.133.48

34.86.49.244

34.85.157.11

35.221.7.118

34.150.248.70

34.86.59.56

1. Trubo平台/資源下載白名單: 確保網路正常通訊位置

Trubo SaaS URL位置: https://tsmc.turbonomic.io  
https://auth.docker.io

https://registry-1.docker.io

https://production.cloudflare.docker.com

1. 離線安裝:  
   離線安裝資料下載及說明參考:  
   https://ibm.box.com/s/vijxnphbhcr54p8tttpa2d0bi6yiw4ll

參考資料:  
 https://github.com/turbonomic/t8c-install/wiki/Working-with-a-Private-Repo-&-Image-Pull-Secrets

1. 建置軟硬體需求:

https://github.com/turbonomic/t8c-install/wiki/3.-Sizing-your-Deployment

1. ~~Ingress/Route設定:~~ 不需設定

https://github.com/turbonomic/t8c-install/wiki/4.-Turbonomic-Multinode-Deployment-Steps#option-1-nginx-as-proxy--bring-your-own-ingressroute

設定ingress on GKE

apiVersion: networking.k8s.io/v1beta1

kind: Ingress

metadata:

name: ingress-turbonomic

annotations:

# use the shared ingress-nginx

kubernetes.io/ingress.class: "nginx"

spec:

rules:

http:

paths:

- path: "/setting-url-path"

backend:

serviceName: "nginx"

servicePort: 80

1. GCP帳號權限設定

4種角色權限需執行於Google Cloud可再參考

Project Access Role

Organization Access Role

Project Action Role

Custom Billing

refer.

http://docs.turbonomic.com/docApp/doc/index.html?config=UG\_Pnt#!/Latest/\_Target\_Configuration\_Topics/TargetsAppendix\_GCP\_Onboard\_OpsTarget.xml

http://docs.turbonomic.com/docApp/doc/index.html?config=UG\_Pnt#!/Latest/\_Target\_Configuration\_Topics/TargetsAppendix\_GCP\_Onboard\_BillingTarget.xml

1. 部署架構:



參考資料:

https://github.com/turbonomic/t8c-install/wiki

https://github.com/turbonomic/kubeturbo/wiki

系統需求及儲存空間:

<https://github.com/turbonomic/t8c-install/wiki/3.-Sizing-your-Deployment>

<https://github.com/turbonomic/t8c-install/wiki/Storage-Class-Requirements#persistent-volumes>

[以下安裝步驟適用於 k8s 環境，並已於 GKE 環境上測試驗證]

Multi-node Deployment Steps - 安裝流程說明 (監控平台端)

==================================================

於安裝包解壓縮路徑內執行:/ibm/turbo-script/

1. 建立Namespace:

kubectl create namespace turbonomic

1. 安裝資源內容: create the Custom Resource

kubectl create -f <https://raw.githubusercontent.com/turbonomic/t8c-install/master/operator/config/crd/bases/charts.helm.k8s.io_xls.yaml>

Kubernetes version 1.22 and higher:

kubectl create -f charts.helm.k8s.io\_xls.yaml

Kubernetes version 1.11 up to 1.21

kubectl create -f charts\_v1alpha1\_xl\_crd.yaml

1. 部署維運中心:

建立服務帳號: Operator service account

kubectl create -f <https://raw.githubusercontent.com/turbonomic/t8c-install/master/operator/deploy/service_account.yaml> -n turbonomic

kubectl create -f service\_account.yaml -n turbonomic

建立角色: Create the role

kubectl create -f <https://raw.githubusercontent.com/turbonomic/t8c-install/master/operator/deploy/role.yaml>-n turbonomic

kubectl create -f role.yaml -n turbonomic

綁定角色: Create the role binding

kubectl create -f <https://raw.githubusercontent.com/turbonomic/t8c-install/master/operator/deploy/role_binding.yaml> -n turbonomic

kubectl create -f role\_binding.yaml -n turbonomic

啟動服務: Init operator pod

kubectl create -f <https://raw.githubusercontent.com/turbonomic/t8c-install/master/operator/deploy/operator.yaml> -n turbonomic

kubectl create -f operator.yaml -n turbonomic

檢查狀態:

kubectl get pods -n turbonomic -w

[root@localhost turbo-script]# kubectl get pods -n turbonomic -w

NAME READY STATUS RESTARTS AGE

t8c-operator-7488ff7cd-dk2lk 1/1 Running 0 11m

1. 建立Custom Resource

Download/Modify Default Custom Resource YAML:

curl https://raw.githubusercontent.com/turbonomic/t8c-install/master/operator/deploy/crds/charts\_v1alpha1\_xl\_cr.yaml -o turbo\_default\_cr.yaml

vim turbo\_default\_cr.yaml (設定參數需要調整)

a.Modify tag to be latest version, ex tag: 8.5.7

b.Uncomment any mediation targets you want to enable (enable GCP/vCenter/Dynatrace)

c.Add Ingress Internal and nginxIsPrimaryIngress setting

spec:

global:

ingress:

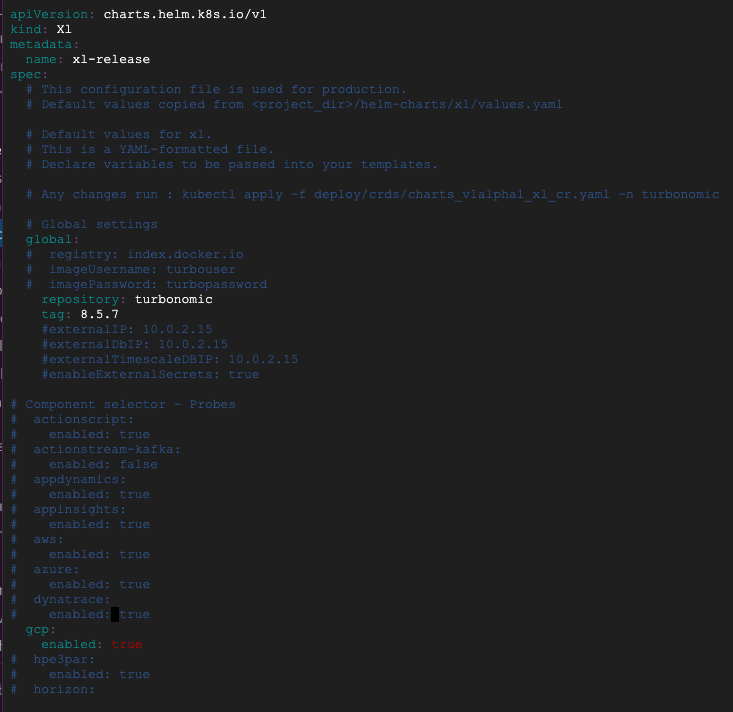
annotations:

cloud.google.com/load-balancer-type: "Internal"

nginx:

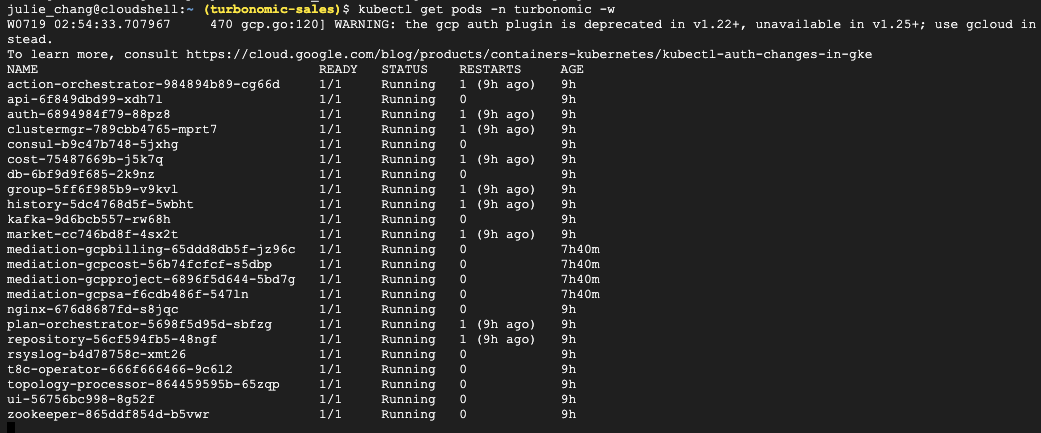
nginxIsPrimaryIngress: true

externalTrafficPolicy: Cluster



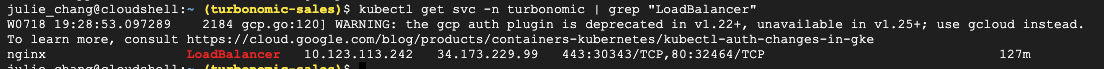
1. Create Custom Resource: kubectl apply -f turbo\_default\_cr.yaml -n turbonomic

Confirm pods start: kubectl get pods -n turbonomic -w

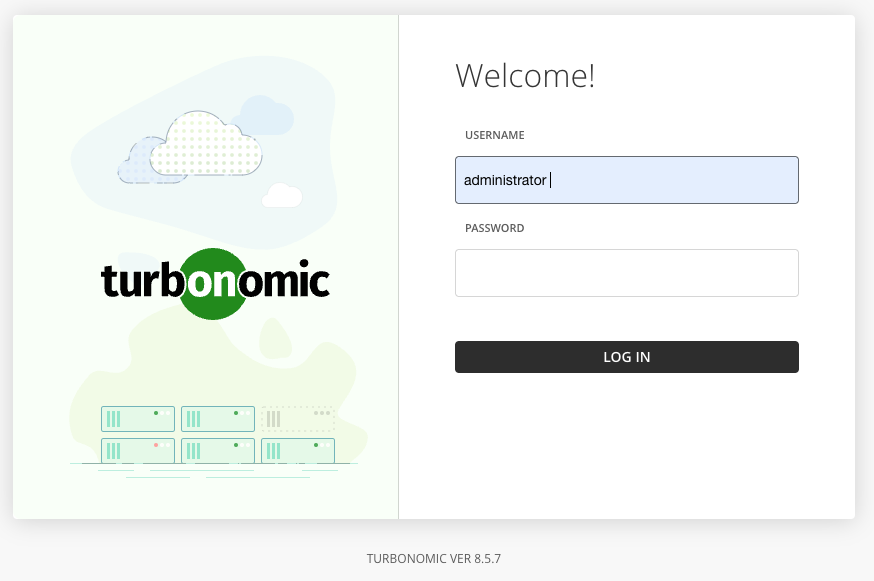


1. Once all pods are Running, then get external IP: kubectl get svc -n turbonomic | grep "LoadBalancer"

For example: 34.173.229.99 (is my external IP)



1. Use External IP in web browser to access GUI. (第一次登入時需設定密碼)



參考資料:

客製化設定可參考章節: Configure the Turbonomic Instance: The Custom Resource

<https://github.com/turbonomic/t8c-install/wiki/4.-Turbonomic-Multinode-Deployment-Steps>

Kubeturbo Install Steps - 安裝流程說明 (監控端)

==================================================

於安裝包解壓縮路徑內執行: /ibm/kubeturbo-script

**1. Create a Namespace**

kubectl create namespace kubeturbo

**2. Create a Service Account**

kubectl create sa turbo-user -n kubeturbo

* servicaccount/turbo-user created

vim serviceAccount.yaml

| apiVersion: v1  kind: ServiceAccount  metadata:  # use this yaml to create a service account to run kubeturbo.  # Provide a value for name:  # Update the namespace value if required  name: turbo-user  namespace: kubeturbo |
| --- |

kubectl -n kubeturbo apply -f serviceAccount.yaml

* servicaccount/turbo-user configured

**3. Create Role Binding**

vim serviceAccountRoleBinging.yaml

| kind: ClusterRoleBinding  # For OpenShift 3.4-3.7 use apiVersion: v1  # For kubernetes 1.9 use rbac.authorization.k8s.io/v1  # For kubernetes 1.8 use rbac.authorization.k8s.io/v1beta1  apiVersion: rbac.authorization.k8s.io/v1  metadata:  # use this yaml to create a binding that will assign cluster-admin to your turbo ServiceAccount  # Provide a value for the binding name: and update namespace if needed  name: turbo-all-binding  subjects:  - kind: ServiceAccount  # Provide the correct value for service account name: and namespace if needed  name: turbo-user  namespace: kubeturbo  roleRef:  # User creating this resource must have permissions to add this policy to the SA  kind: ClusterRole  # accepted values cluster-reader disc and monitoring. federation?  name: cluster-admin  # For OpenShift v3.4 remove apiGroup line  apiGroup: rbac.authorization.k8s.io |
| --- |

kubectl -n kubeturbo apply -fserviceAccountRoleBinging.yaml

* clusterrolebinding.rbac.authorization.k8s.io/turbo-all-binding created

**4.** **Create a configMap**

vim turboConfig.yaml

填 "turboServer": "https://tsmc.turbonomic.io"

若自建環境可寫入IP位置

填 "turboServer": "https://<GKE IP>"

targetName: 需使用不同名稱

"targetConfig": { "targetName": "<每個 cluster 都要用不同命名作為標籤>"

| apiVersion: v1  kind: ConfigMap  metadata:  # use this yaml to create a config that kubeturbo will use to connect to the Turbo Server  # requires Turbo Server and kubeturbo pod 6.4.3 and higher  # Provide a value for the config name: and update namespace if needed  name: turbo-config  namespace: kubeturbo  data:  # Update the values for version, turboServer, opsManagerUserName, opsManagerPassword  # For version, use Turbo Server Version, even when running CWOM  # The opsManagerUserName requires Turbo administrator role  #  # For targetConfig, targetName provides better group naming to identify k8s clusters in UI  # - If no targetConfig is specified, a default targetName will be created from the apiserver URL in  # the kubeconfig.  # - Specify a targetName only will register a probe with type Kubernetes-<targetName>, as well as  # adding your cluster as a target with the name Kubernetes-<targetName>.  # - Specify a targetType only will register a probe without adding your cluster as a target.  # The probe will appear as a Cloud Native probe in the UI with a type Kubernetes-<targetType>.  #  # Define node groups by node role, and automatically enable placement policies to limit to 1 per host  # DaemonSets are identified by default. Use daemonPodDetectors to identify by name patterns using regex or by namespace.  #  # The annotationWhitelist provides a mechanism for discovering annotations for kubernetes objects.  # By default, no annotations are collected. In order to collect annotations, provide a regular  # expression for each entity type for which the annotations are desired.  turbo.config: |-  {  "communicationConfig": {  "serverMeta": {  "version": "8.0",  "turboServer": "https://<Turbo\_server\_URL>"  },  "restAPIConfig": {  "opsManagerUserName": "<Turbo\_username>",  "opsManagerPassword": "<Turbo\_password>"  }  },  "targetConfig": {  "targetName": "GKE POV"  },  "HANodeConfig": {  "nodeRoles": [ "master" ]  },  "annotationWhitelist": {  "containerSpec": "<regex>",  "namespace": "<regex>",  "workloadController": "<regex>"  }  } |
| --- |

kubectl -n kubeturbo apply -f turboConfig.yaml

* configmap/turbo-config created

5. **Create a deployment for kubeturbo**

vim kubeTurboDeploy.yaml

| apiVersion: apps/v1  kind: Deployment  metadata:  # use this yaml to deploy the kubeturbo pod  # Provide a value for the deploy/pod name: and update namespace if needed  name: kubeturbo  namespace: kubeturbo  spec:  replicas: 1  selector:  matchLabels:  app.kubernetes.io/name: kubeturbo  template:  metadata:  annotations:  kubeturbo.io/monitored: "false"  labels:  app.kubernetes.io/name: kubeturbo  spec:  # Update serviceAccount if needed  serviceAccountName: turbo-user  containers:  - name: kubeturbo  # Replace the image with desired version:6.4.4 or latest  image: turbonomic/kubeturbo:8.5.7  env:  - name: KUBETURBO\_NAMESPACE  valueFrom:  fieldRef:  fieldPath: metadata.namespace  args:  - --turboconfig=/etc/kubeturbo/turbo.config  - --v=2  # Comment out the following two args if running in k8s 1.10 or older, or  # change to https=false and port=10255 if unsecure kubelet read only is configured  - --kubelet-https=true  - --kubelet-port=10250  # Uncomment for pod moves in OpenShift  #- --scc-support=\*  # Uncomment for pod moves with pvs  #- --fail-volume-pod-moves=false  # Uncomment to override default, and specify your own location  #- --busybox-image=docker.io/busybox  # or uncomment below to pull from RHCC  #- --busybox-image=registry.access.redhat.com/ubi8/ubi-minimal  # Uncomment to specify the secret name which holds the credentials to busybox image  #- --busybox-image-pull-secret=<secret-name>  # Specify nodes to exclude from cpu frequency getter job.  # Note kubernetes.io/os=windows and/or beta.kubernetes.io/os=windows labels will be automatically excluded by default.  # If specified all the labels will be used to select the node ignoring the default.  #- --cpufreq-job-exclude-node-labels=kubernetes.io/key=value  # Uncomment to stitch using IP, or if using Openstack, Hyper-V/VMM  #- --stitch-uuid=false  # Uncomment to customize readiness retry threshold. Kubeturbo will try readiness-retry-threshold times before giving up. Default is 60. The retry interval is 10s.  #- --readiness-retry-threshold=60  volumeMounts:  # volume will be created, any name will work and must match below  - name: turbo-volume  mountPath: /etc/kubeturbo  readOnly: true  - name: turbonomic-credentials-volume  mountPath: /etc/turbonomic-credentials  readOnly: true  - name: varlog  mountPath: /var/log  volumes:  - name: turbo-volume  configMap:  # Update configMap name if needed  name: turbo-config  - name: turbonomic-credentials-volume  secret:  defaultMode: 420  optional: true  # Update secret name if needed  secretName: turbonomic-credentials  - name: varlog  emptyDir: {}  restartPolicy: Always |
| --- |

kubectl -n kubeturbo apply -f kubeTurboDeploy.yaml

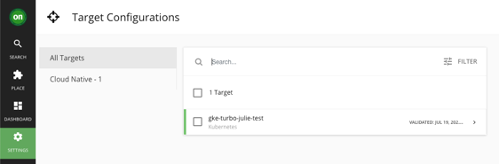
-> deployment.apps/kubeturbo created

**6. Check kubeturbo is running**

kubectl -n kubeturbo get pods

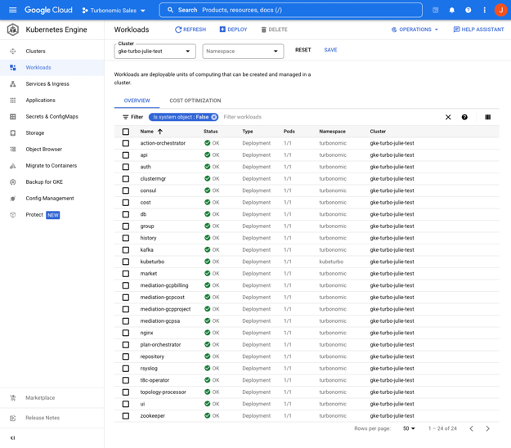


**7.Validate data in the Turbo UI**

****

****

**8. Verified workloads from GKE UI**

****

參考資料:

Kubeturbo監控節點:

https://hub.docker.com/r/turbonomic/kubeturbo/tags

https://github.com/turbonomic/kubeturbo/tree/master/deploy/kubeturbo\_yamls

Kubeturbo安裝步驟:  
https://github.com/turbonomic/kubeturbo/wiki/Yaml-Deployment-Details#deploy-kubeturbo  
https://github.com/turbonomic/kubeturbo/wiki/Operator-Details

Ingress Controller with private IP

==================================

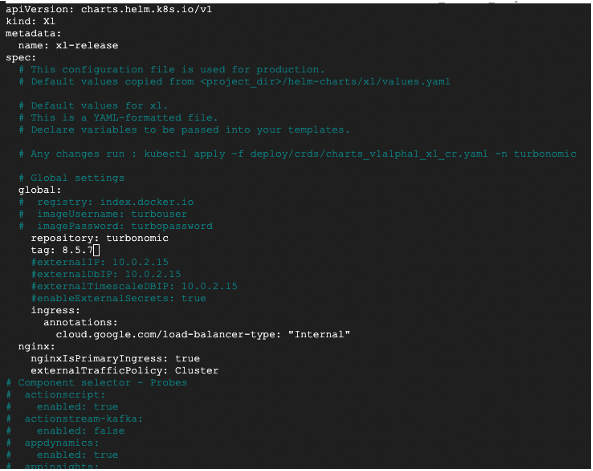
* 不需要使用 Ingress file 可移除

kubectl delete svc ingress-turbonomic -n turbonomic

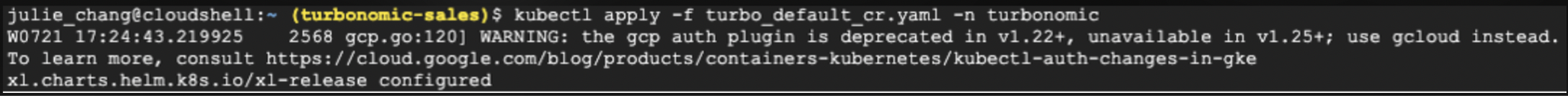
* 需要在 CR file 增加以下資訊

vim turbo\_default\_cr.yaml

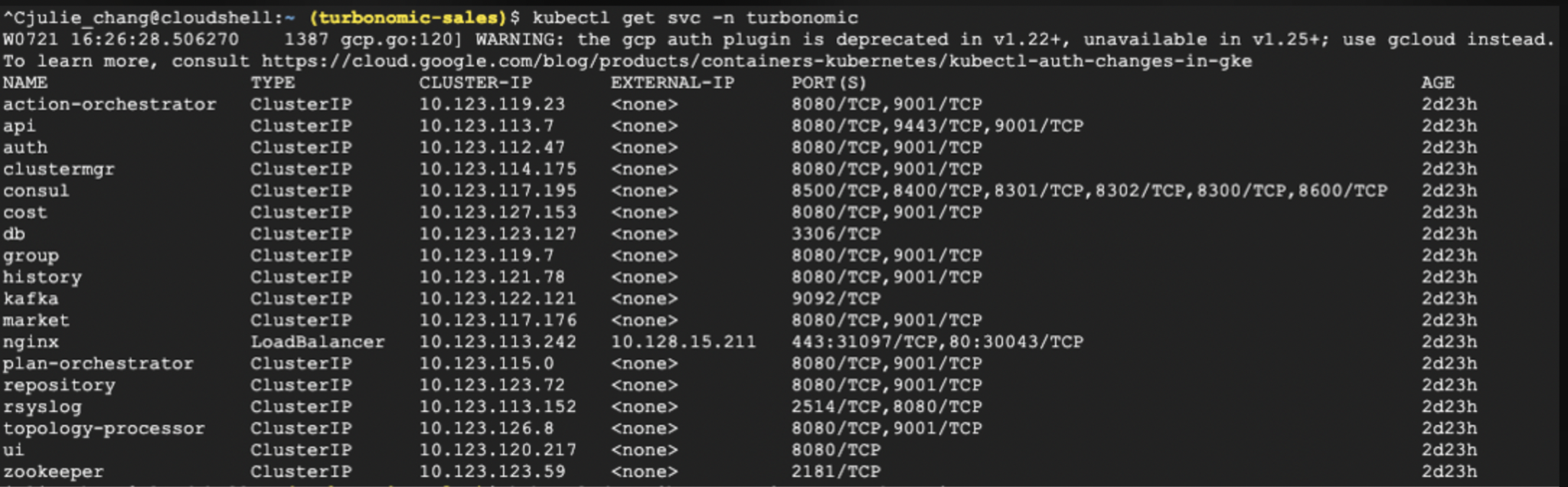
| spec:  global:  ingress:  annotations:  cloud.google.com/load-balancer-type: "Internal"  nginx:  nginxIsPrimaryIngress: true  externalTrafficPolicy: Cluster |
| --- |



kubectl apply -f turbo\_default\_cr.yaml -n turbonomic

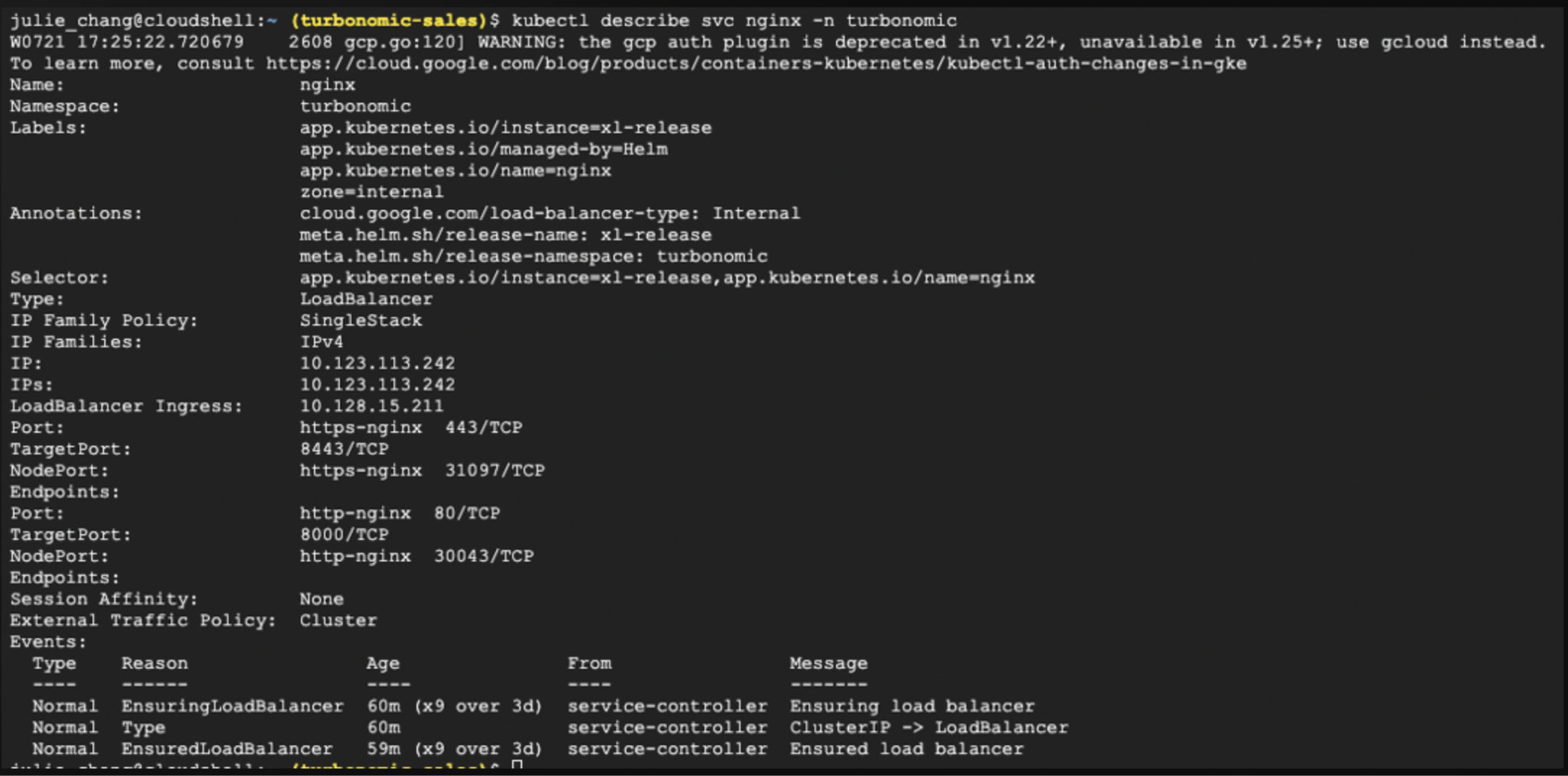


kubectl get svc -n turbonomic



kubectl describe svc nginx -n turbonomic

(my LB Ingress is 10.128.15.211)



Test internal IP connection in my GCP VM (a node)

curl -kX 'GET' '<https://10.128.15.211/api/v3/admin/versions?check_for_updates=false>' -H 'accept: application/json'

