GKE GCloud

The contribution of this document is the most cost effective way to provision a T4.

Colab T4 GPUs have performance issues with the FUSE gdrive mount long latency causing timeouts and networking limitations preventing http requests and open ports. COlab t4s also timeout when doing fine tuning despite paying for colab pro or extra hours. Gave up after buying 500-1k extra unused hours. They locked it down to prevent any AI/ML work on colab.

Command to get t4. It never succeeds. But GKE does.

```
WARNING: You have selected a disk size of under [200GB]. This may result in poor I/O performance. For more information, see:
https://developers.google.com/compute/docs/disks#performance.
ERROR: (gcloud.compute.instances.create) Could not fetch resource:
---
code: ZONE_RESOURCE_POOL_EXHAUSTED
errorDetails:
- help:
    links:
    - description: Troubleshooting documentation
      url: https://cloud.google.com/compute/docs/resource-error
- localizedMessage:
    locale: en-US
    message: A n1-standard-4 VM instance with 1 nvidia-tesla-t4 accelerator(s) is
      currently unavailable in the us-central1-f zone. Alternatively, you can try
      your request again with a different VM hardware configuration or at a later
      time. For more information, see the troubleshooting documentation.
- errorInfo:
    domain: compute.googleapis.com
    metadatas:
      attachment: nvidia-tesla-t4:1
      vmType: n1-standard-4
      zone: us-central1-f
      zonesAvailable: ''
    reason: resource_availability
message: The zone 'projects/seraphic-plexus-328620/zones/us-central1-f' does not have
  enough resources available to fulfill the request.  Try a different zone, or try
  again later.
```

There is no way to get a T4 in us-central1-f while requesting a T4 through GKE always results in available T4s. This isn't documented anywhere.

There are 2 GKE configurations, standard and auto. The online google dashboard pushes Auto without explanation.

This document goes through standard because auto introduces additional costs like deploying GKE control plane in HA configuration and using higher end CPU compute nodes before the GPUs are deployed.

Summary:

1) create GKE cloud controller. This needs a m2-small node for the dashboard; if the dashboard is enabled. Can delete it after GKE is created to minimize cost. Careful with the --region flag vs --zone flag. --region creates 3 replicas like zookeeper whiile --zone creates 1 copy only.
2) Create the gpu nodes. create in the same zone as GKE. central1-f.

3) there is a GKE dashboard which needs to be installed. it runs as a proxy. There is no direct URL; it hsa to run through the GKE node proxy.

```
gcloud compute instances create t4-vm \
  --zone=us-central1-f \
  --machine-type=n1-standard-4 \
  --accelerator=type=nvidia-tesla-t4,count=1 \
  --maintenance-policy=TERMINATE \
  --restart-on-failure \
  --image-family=ubuntu-2204-lts \
  --image-project=ubuntu-os-cloud \
  --boot-disk-size=100GB
```

A GKE or kubernetes cluster has 2 parts, a control plane and a set of nodes. The control plane is called gke, it functions like a zookeeper master where it coordinates with the nodes.

NOTES: GKE adds 1-3 CPU instances. If you don't specify it picks medium CPUs. If you specify region then it makes 3 copies of the GKE Controller like zookeeper. The cheapest dev environment is specify a zone and not a region then add a GPU from that same zone. Delete the m2-small after GKE control plane is created. Can't get t4 GPUs without using GKE.

A GKE control plane has 2 options:

1) HA mode where there are 3 copies allocated per region. us-central1 is region. A region has multiple zones. us-centra1 has "us-central1-a,us-central1-b,us-central1-c,us-central1-d,us-central1-e,us-central1-f". Note the removal of spaces in the string. Allocate this with a region option. A region option will create GKE CP in 3 separate zones. These cost a min of $.10/hr.

```
gcloud container clusters create gke_test \
--region=us-central-1 \
--machine-type=e2-small \
```

--num-nodes=1

2) A non HA mode where there is only 1 copy of the control plane. Do not specify a region, use a zone instead. These are free.

**gcloud container clusters create my-cluster **
**--zone=us-central1-f **
**--machine-type=e2-small **
--num-nodes=1

The cluster automatically adds a CPU node. There is no way to create the control plane without a compute node. The old docs suggest creating then removing but you are still charged. Create with the smallest instance then delete.

Deleting the ec2 node since it isnt being used and a compute node is allocated for the GPU.

Adding GPU nodes:
A T4 GPU node is available in us-central1-f. There is a higher priority given to GKE clusters.

CIDR address inside zone less general than address in region. Gives warning on number IP addresses.

Creating GKE cluster with single e2-small.

```
gcloud container clusters create my-cluster \
        --zone=us-central1-f \
        --machine-type=e2-small \
        --num-nodes=1
Note: Your Pod address range (`--cluster-ipv4-cidr`) can accommodate at most 1008 node(s).
Creating cluster my-cluster in us-central1-f... Cluster is being configured...⠼
```

gcloud container clusters describe my-cluster --zone us-central1-f

```
addonsConfig:
  gcePersistentDiskCsiDriverConfig:
    enabled: true
  kubernetesDashboard:
    disabled: true
  networkPolicyConfig:
    disabled: true
anonymousAuthenticationConfig:
```

```
  mode: ENABLED
autopilot: {}
autoscaling:
  autoprovisioningNodePoolDefaults:
    imageType: COS_CONTAINERD
    management:
      autoRepair: true
      autoUpgrade: true
    oauthScopes:
    - https://www.googleapis.com/auth/devstorage.read_only
    - https://www.googleapis.com/auth/logging.write
    - https://www.googleapis.com/auth/monitoring
    - https://www.googleapis.com/auth/service.management.readonly
    - https://www.googleapis.com/auth/servicecontrol
    - https://www.googleapis.com/auth/trace.append
    serviceAccount: default
  autoscalingProfile: BALANCED
clusterIpv4Cidr: 10.96.0.0/14
controlPlaneEndpointsConfig:
  dnsEndpointConfig:
    allowExternalTraffic: false
    enableK8sCertsViaDns: false
    enableK8sTokensViaDns: false
    endpoint: gke-45e417124f6d4454ad617cc65cf38b7a9f48-428120510823.us-central1-
f.gke.goog
  ipEndpointsConfig:
    authorizedNetworksConfig: {}
    enablePublicEndpoint: true
    enabled: true
    privateEndpoint: 10.128.0.16
    publicEndpoint: 104.197.232.0
createTime: '2026-01-04T19:38:38+00:00'
currentMasterVersion: 1.33.5-gke.1308000
currentNodeCount: 1
currentNodeVersion: 1.33.5-gke.1308000
databaseEncryption:
  currentState: CURRENT_STATE_DECRYPTED
  state: DECRYPTED
defaultMaxPodsConstraint:
  maxPodsPerNode: '110'
endpoint: 104.197.232.0
enterpriseConfig:
  clusterTier: STANDARD
etag: acbdfbfc-e9bd-409a-bc83-ac63ef48e747
id: 45e417124f6d4454ad617cc65cf38b7a9f4819d367534dba89acdb1837121185
initialClusterVersion: 1.33.5-gke.1308000
instanceGroupUrls:
- https://www.googleapis.com/compute/v1/projects/seraphic-plexus-328620/zones/us-
central1-f/instanceGroupManagers/gke-my-cluster-default-pool-5221495b-grp
ipAllocationPolicy:
  clusterIpv4Cidr: 10.96.0.0/14
  clusterIpv4CidrBlock: 10.96.0.0/14
  clusterSecondaryRangeName: gke-my-cluster-pods-45e41712
  defaultPodIpv4RangeUtilization: 0.001
```

```
  networkTierConfig:
    networkTier: NETWORK_TIER_DEFAULT
  podCidrOverprovisionConfig: {}
  servicesIpv4Cidr: 34.118.224.0/20
  servicesIpv4CidrBlock: 34.118.224.0/20
  stackType: IPV4
  useIpAliases: true
labelFingerprint: a9dc16a7
legacyAbac: {}
location: us-central1-f
locations:
- us-central1-f
loggingConfig:
  componentConfig:
    enableComponents:
    - SYSTEM_COMPONENTS
    - WORKLOADS
loggingService: logging.googleapis.com/kubernetes
maintenancePolicy:
  resourceVersion: e3b0c442
masterAuth:
  clientCertificateConfig: {}
  clusterCaCertificate:REMOVED for security
masterAuthorizedNetworksConfig: {}
monitoringConfig:
  advancedDatapathObservabilityConfig: {}
  componentConfig:
    enableComponents:
    - SYSTEM_COMPONENTS
    - STORAGE
    - HPA
    - POD
    - DAEMONSET
    - DEPLOYMENT
    - STATEFULSET
    - JOBSET
    - CADVISOR
    - KUBELET
    - DCGM
  managedPrometheusConfig:
    enabled: true
monitoringService: monitoring.googleapis.com/kubernetes
name: my-cluster
network: default
networkConfig:
  network: projects/seraphic-plexus-328620/global/networks/default
  serviceExternalIpsConfig: {}
  subnetwork: projects/seraphic-plexus-328620/regions/us-central1/subnetworks/default
nodeConfig:
  bootDisk:
    diskType: pd-balanced
    sizeGb: '100'
  diskSizeGb: 100
  diskType: pd-balanced
```

```
  effectiveCgroupMode: EFFECTIVE_CGROUP_MODE_V2
  imageType: COS_CONTAINERD
  kubeletConfig:
    insecureKubeletReadonlyPortEnabled: false
    maxParallelImagePulls: 2
  machineType: e2-small
  metadata:
    disable-legacy-endpoints: 'true'
  oauthScopes:
  - https://www.googleapis.com/auth/devstorage.read_only
  - https://www.googleapis.com/auth/logging.write
  - https://www.googleapis.com/auth/monitoring
  - https://www.googleapis.com/auth/service.management.readonly
  - https://www.googleapis.com/auth/servicecontrol
  - https://www.googleapis.com/auth/trace.append
  resourceLabels:
    goog-gke-node-pool-provisioning-model: on-demand
  serviceAccount: default
  shieldedInstanceConfig:
    enableIntegrityMonitoring: true
  windowsNodeConfig: {}
nodePoolAutoConfig:
  nodeKubeletConfig:
    insecureKubeletReadonlyPortEnabled: false
nodePoolDefaults:
  nodeConfigDefaults:
    loggingConfig:
      variantConfig:
        variant: DEFAULT
    nodeKubeletConfig:
      insecureKubeletReadonlyPortEnabled: false
nodePools:
- autoscaling: {}
  config:
    bootDisk:
      diskType: pd-balanced
      sizeGb: '100'
    diskSizeGb: 100
    diskType: pd-balanced
    effectiveCgroupMode: EFFECTIVE_CGROUP_MODE_V2
    imageType: COS_CONTAINERD
    kubeletConfig:
      insecureKubeletReadonlyPortEnabled: false
      maxParallelImagePulls: 2
    machineType: e2-small
    metadata:
      disable-legacy-endpoints: 'true'
    oauthScopes:
    - https://www.googleapis.com/auth/devstorage.read_only
    - https://www.googleapis.com/auth/logging.write
    - https://www.googleapis.com/auth/monitoring
    - https://www.googleapis.com/auth/service.management.readonly
    - https://www.googleapis.com/auth/servicecontrol
    - https://www.googleapis.com/auth/trace.append
```

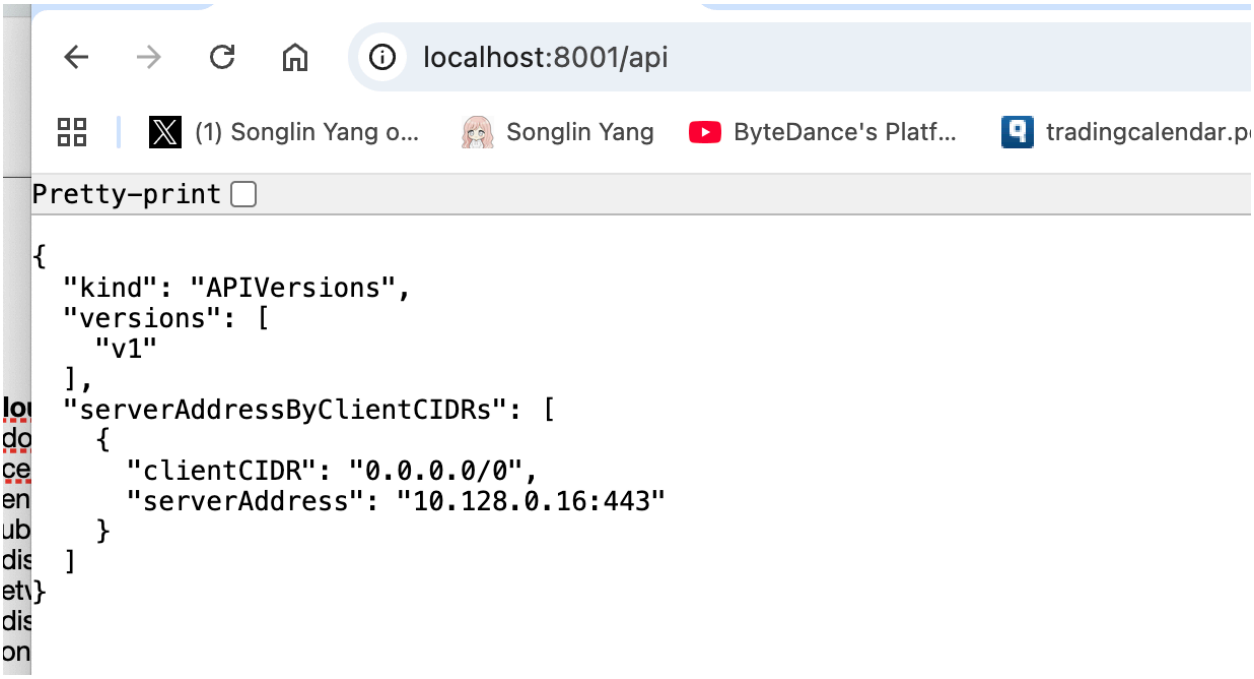
```
    resourceLabels:
      goog-gke-node-pool-provisioning-model: on-demand
    serviceAccount: default
    shieldedInstanceConfig:
      enableIntegrityMonitoring: true
    windowsNodeConfig: {}
  etag: 80a376f4-1a8d-41d6-a412-638b780d4348
  initialNodeCount: 1
  instanceGroupUrls:
  - https://www.googleapis.com/compute/v1/projects/seraphic-plexus-328620/zones/us-
central1-f/instanceGroupManagers/gke-my-cluster-default-pool-5221495b-grp
  locations:
  - us-central1-f
  management:
    autoRepair: true
    autoUpgrade: true
  maxPodsConstraint:
    maxPodsPerNode: '110'
  name: default-pool
  networkConfig:
    networkTierConfig:
      networkTier: NETWORK_TIER_DEFAULT
    podIpv4CidrBlock: 10.96.0.0/14
    podIpv4RangeUtilization: 0.001
    podRange: gke-my-cluster-pods-45e41712
    subnetwork: projects/seraphic-plexus-328620/regions/us-central1/subnetworks/default
  podIpv4CidrSize: 24
  selfLink: https://container.googleapis.com/v1/projects/seraphic-plexus-328620/zones/us-
central1-f/clusters/my-cluster/nodePools/default-pool
  status: RUNNING
  upgradeSettings:
    maxSurge: 1
    strategy: SURGE
  version: 1.33.5-gke.1308000
notificationConfig:
  pubsub: {}
podAutoscaling:
  hpaProfile: PERFORMANCE
privateClusterConfig:
  privateEndpoint: 10.128.0.16
  publicEndpoint: 104.197.232.0
rbacBindingConfig:
  enableInsecureBindingSystemAuthenticated: true
  enableInsecureBindingSystemUnauthenticated: true
releaseChannel:
  channel: REGULAR
securityPostureConfig:
  mode: BASIC
  vulnerabilityMode: VULNERABILITY_MODE_UNSPECIFIED
selfLink: https://container.googleapis.com/v1/projects/seraphic-plexus-328620/zones/us-
central1-f/clusters/my-cluster
servicesIpv4Cidr: 34.118.224.0/20
shieldedNodes:
  enabled: true
```

status: RUNNING
subnetwork: default
userManagedKeysConfig: {}
zone: us-central1-f

Proxy
Runs on local dev machine and forwards requests to Kube Server. NO certificate/ssh required.

>kubectl proxy

Verify can access localhost:8001/api



Kube Cluster Dashboard

1) get credentials
2) kubectl get-nodes
3) install dashboard kubectl apply -f \

https://raw.githubusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yaml
4) check running kubectl get pods -n kubernetes-dashboard

```
gcloud container clusters get-credentials my-cluster \
  --zone us-central1-f
Fetching cluster endpoint and auth data.
kubeconfig entry generated for my-cluster.
kubectl get nodes
NAME                                          STATUS   ROLES    AGE   VERSION
gke-my-cluster-default-pool-5221495b-xcc0   Ready    <none>   12m   v1.33.5-gke.1308000
kubectl apply -f \
https://raw.githubusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yaml
namespace/kubernetes-dashboard created
serviceaccount/kubernetes-dashboard created
service/kubernetes-dashboard created
secret/kubernetes-dashboard-certs created
secret/kubernetes-dashboard-csrf created
secret/kubernetes-dashboard-key-holder created
configmap/kubernetes-dashboard-settings created
role.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrole.rbac.authorization.k8s.io/kubernetes-dashboard created
rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
clusterrolebinding.rbac.authorization.k8s.io/kubernetes-dashboard created
deployment.apps/kubernetes-dashboard created
service/dashboard-metrics-scraper created
deployment.apps/dashboard-metrics-scraper created
```

Context:
kubectl writes context pairs,(cluster, user, namespace), into .kube/context. kube command kubectl get pods uses the context file to find the context pairs to send the request to the right cluster.

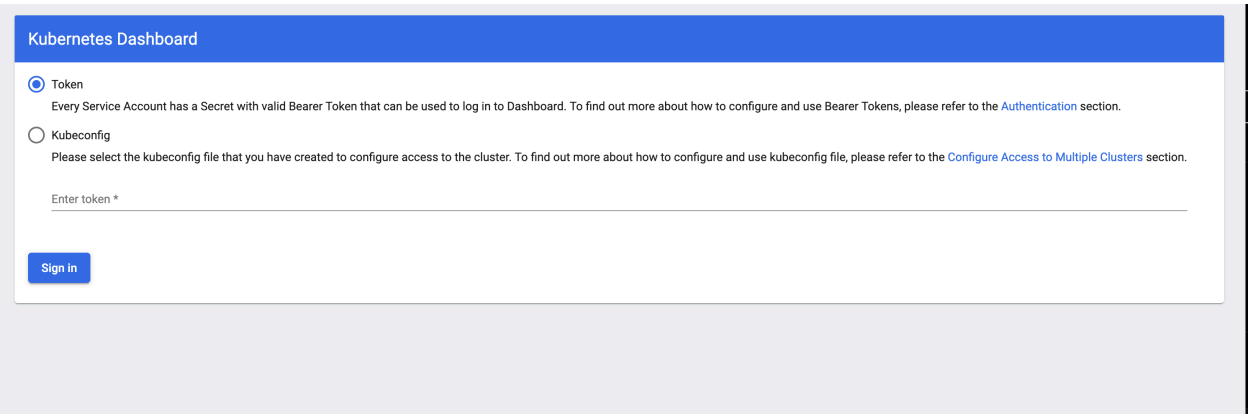
Problem is this file isn't pruned automaticaly and as clusters are deleted the context pairs are no lunger used.

Create RBAC login, service account after dashboard created
kubectl create serviceaccount dashboard-viewer -n kubernetes-dashboard

```
kubectl create clusterrolebinding dashboard-viewer \
  --clusterrole=view \
  --serviceaccount=kubernetes-dashboard:dashboard-viewer
```

Get the token:
kubectl -n kubernetes-dashboard create token dashboard-viewer

Paste in the token to the dashboard



Where it says Enter token in light gray.

kubectl -n kubernetes-dashboard create token dashboard-viewer

eyJhbGciOiJSUzI1NiIsImtpZCI6IkRBbkI3LWxkdlhmdnhtbmwtTVdfbE5zcjEtZko4bVB6OUZJa1
UtbjBfYlEifQ.eyJhdWQiOlsiaHR0cHM6Ly9jb250YWluZXIuZ29vZ2xlYXBpcy5jb20vdjEvcHJvam
VjdHMvc2VyYXBoaWMtcGxleHVzLTMyODYyMC9sb2NhdGlvbnMvdXMtY2VudHJhbDEtZi9jbH
VzdGVycy9teS1jbHVzdGVyIl0sImV4cCI6MTc2NzU2MjI3MiwiaWF0IjoxNzY3NTU4NjcyLCJpc3M
iOiJodHRwczovL2NvbnRhaW5lci5nb29nbGVhcGlzLmNvbS92MS9wcm9qZWN0cy9zZXJhcGh
pYy1wbGV4dXMtMzI4NjIwL2xvY2F0aW9ucy91cy1jZW50cmFsMS1mL2NsdXN0ZXJzL215LWN
sdXN0ZXIiLCJqdGkiOiIyMWQxN2ExYi1mZDY0LTQyY2MtYTBhZC1jN2U0ZjQ5N2M5NmMiLCJ
rdWJlcm5ldGVzLmlvIjp7Im5hbWVzcGFjZSI6Imt1YmVybmV0ZXMtZGFzaGJvYXJkIiwic2Vydmlj
ZWFjY291bnQiOnsibmFtZSI6ImRhc2hib2FyZC12aWV3ZXIiLCJ1aWQiOiIxMDA4M2NjZi00N2Q
wLTQ5OWQtYTJiZi00NTQzZDA2NDZmMWYifX0sIm5iZiI6MTc2NzU1ODY3Miwic3ViIjoic3lzdGV
tOnNlcnZpY2VhY2NvdW50Omt1YmVybmV0ZXMtZGFzaGJvYXJkOmRhc2hib2FyZC12aWV3Z
XlifQ.cl3nznL1ubOVWg824BgAWaTAKEhgnQTSDKRH_a-MgX6HOtMq5OTKLywSn0pIN-
YCwXxYTM7cY2gwtgqudgy_6P-
yyUWyAshkKzh048vULH_FSbNPloPy93sXQLlDxSSOncAvYzvPp7O9HyWyFt4Tb8EkJpUQxx3
d1fphLtorYKEJiJwgKNxuo2H-cJ8zWVhj7-
i3ZR0L_uIPjfnQANHs4vYa1eq3vLgovnhHYxnO0NWTv7TwKp8SxXCjBEg3D5grsFF1Nfovnh2qH
6xozm9BQXojF8pQ8C3i7x3LlNUrtOvyJ5t1SOrpjqQ5Ycjt2wcmYo2JOZR-bfmw9E_P9I-iog

Creating GPU nodes:

for 1 gpu do not use zones and specify 1 region. If add csv with multiple locations it creates 1 GPU per zone.

```
PROJECT="seraphic-plexus-328620"
```

```
ZONE="us-central1-f"
CLUSTER="my-cluster"
MACHINE_TYPE=n1-standard-4
POOL="t4-pool"
MACHINE_TYPE="n1-standard-8"
GPU_TYPE="nvidia-tesla-t4"
GPU_COUNT=1
NUM_NODES=1
```

```
gcloud container node-pools create "t4-pool" \
    --cluster "my-cluster" \
    --zone "us-central1-f" \
    --num-nodes=1 \
    --machine-type="n1-standard-8" \
    --accelerator "type=nvidia-tesla-t4,count=1" \
    --image-type=UBUNTU_CONTAINERD \
    --enable-autorepair \
    --disk-type pd-standard \
    --disk-size 100 \
    --enable-autoupgrade \
    --scopes=https://www.googleapis.com/auth/cloud-platform
```

remove --spot

```
gcloud container node-pools create "t4-pool" \
   --cluster "my-cluster" \
   --zone "us-central1-f" \
   --num-nodes=1 \
   --machine-type="n1-standard-8" \
   --accelerator "type=nvidia-tesla-t4,count=1" \
   --image-type=UBUNTU_CONTAINERD \
   --enable-autorepair \
   --disk-type pd-standard \
   --disk-size 100 \
   --enable-autoupgrade \
   --scopes=https://www.googleapis.com/auth/cloud-platform
```

Note: Modifications on the boot disks of node VMs do not persist across node recreations. Nodes are recreated during manual-upgrade, auto-upgrade, auto-repair, and auto-scaling. To preserve modifications across node recreation, use a DaemonSet.
Note: Machines with GPUs have certain limitations which may affect your workflow. Learn more at https://cloud.google.com/kubernetes-engine/docs/how-to/gpus
Note: Starting in GKE 1.30.1-gke.115600, if you don't specify a driver version, GKE installs the default GPU driver for your node's GKE version.
Creating node pool t4-pool...⠧

kubectl exec -it <pod-name> -- bash

Nodes:
**kubectl get nodes **
** -l cloud.google.com/gke-nodepool=t4-pool **
** -o name**
node/gke-my-cluster-t4-pool-49f74196-s944

Pods:
Need the node name to get the pod names

**kubectl get pods -A -o wide **
** --field-selector spec.nodeName=gke-my-cluster-t4-pool-49f74196-s944**
NAMESPACE NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES
gke-managed-system dcgm-exporter-t5zmx 1/1 Running 0 72m 10.96.1.4 gke-my-cluster-t4-pool-49f74196-s944 <none> <none>
gmp-system collector-qzdr7 2/2 Running 0 72m 10.96.1.3 gke-my-cluster-t4-pool-49f74196-s944 <none> <none>
kube-system fluentbit-gke-fq847 3/3 Running 0 72m 10.128.0.18 gke-my-cluster-t4-pool-49f74196-s944 <none> <none>
kube-system gke-metrics-agent-7mxz2 3/3 Running 0 72m 10.128.0.18 gke-my-cluster-t4-pool-49f74196-s944 <none> <none>
kube-system kube-proxy-gke-my-cluster-t4-pool-49f74196-s944 1/1 Running 0 72m 10.128.0.18 gke-my-cluster-t4-pool-49f74196-s944 <none> <none>
kube-system maintenance-handler-5lj5h 1/1 Running 0 72m 10.96.1.2 gke-my-cluster-t4-pool-49f74196-s944 <none> <none>
kube-system nvidia-gpu-device-plugin-small-ubuntu-fdmc9 3/3 Running 0 72m 10.128.0.18 gke-my-cluster-t4-pool-49f74196-s944 <none> <none>
kube-system pdcsi-node-dwqsv 2/2 Running 0 72m 10.128.0.18 gke-my-cluster-t4-pool-49f74196-s944 <none> <none>

Create a persistent volume: notebooks-pvc.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: notebooks-pvc
spec:
  accessModes: ["ReadWriteOnce"]
  resources:
    requests:
      storage: 50Gi
  storageClassName: standard-rwo
```

kubectl apply -f notebooks-pvc.yaml

Create jupyter-t4.yaml:

```
apiVersion: v1
kind: Pod
metadata:
  name: jupyter-t4
spec:
  restartPolicy: Never

  nodeSelector:
    cloud.google.com/gke-nodepool: t4-pool

  tolerations:
    - key: "nvidia.com/gpu"
      operator: "Equal"
      value: "present"
      effect: "NoSchedule"

  containers:
    - name: jupyter
      image: pytorch/pytorch:2.2.2-cuda12.1-cudnn8-runtime
      resources:
        limits:
          nvidia.com/gpu: 1
          cpu: "4"
          memory: "16Gi"

      ports:
        - containerPort: 8888

      volumeMounts:
        - name: notebooks
          mountPath: /workspace
```

```
    command: ["bash", "-lc"]
    args:
      - |
        pip install --no-cache-dir jupyterlab matplotlib && \
        jupyter lab \
          --ip=0.0.0.0 --port=8888 --no-browser \
          --ServerApp.root_dir=/workspace \
          --ServerApp.token='' --ServerApp.password=''
  volumes:
    - name: notebooks
      persistentVolumeClaim:
        claimName: notebooks-pvc
```

Appy the jupyter notebook yaml
kubectl apply -f jupyter-t4.yaml

Wait until running:
kubectl get pod jupyter-t4 -w

```
dc@dcs-MacBook-Pro kube_vllm % kubectl delete pod jupyter-t4
kubectl apply -f jupyter-t4.yaml
kubectl get pod jupyter-t4 -w
pod "jupyter-t4" deleted
pod/jupyter-t4 created
NAME        READY   STATUS              RESTARTS   AGE
jupyter-t4   0/1     ContainerCreating   0          0s
jupyter-t4   1/1     Running             0          9s
```

```
dc@dcs-MacBook-Pro kube_vllm % kubectl apply -f jupyter-t4.yaml

pod/jupyter-t4 created
dc@dcs-MacBook-Pro kube_vllm % kubectl get pod jupyter-t4 -w

NAME        READY   STATUS              RESTARTS   AGE
jupyter-t4   0/1     ContainerCreating   0          9s
```

Port forward:
kubectl port-forward pod/jupyter-t4 8888:8888

Go to localhost:8888

Open Jupyter cell and verify cells work

import torch
torch.cuda.is_available(), torch.cuda.get_device_name(0)