Marc Vintró Alonso

DevOps Test Notes

I installed the provided helm package ping.tar.gz in GCP to test the functionality.
I created a cluster with the following parameters **[1]**.
Cluster was created **[2].**

I manually installed the helm chart using the cloud shell.

```
$tar -zxvf ping.tar.gz
```

I connected to the cluster to get the cluster credentials :

```
$gcloud container clusters get-credentials cluster-1 --zone us-central1-a --project instant-jetty-435813-p4
```

and installed the helm chart on it :

```
$helm install ping ./ --namespace default
```

I got the following error:

```
Error: INSTALLATION FAILED: parse error at (ping/templates/deployment.yaml:51): unexpected EOF
```

I added {{- end }} to deployment.yaml to be able to deploy it.

```
NAME: ping
LAST DEPLOYED: Sat Jan 18 10:32:33 2025
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
1. Get the application URL by running these commands:
  export POD_NAME=$(kubectl get pods --namespace default -l
"app.kubernetes.io/name=ping,app.kubernetes.io/instance=ping" -o jsonpath="{.items[0].metadata.name}")
  export CONTAINER_PORT=$(kubectl get pod --namespace default $POD_NAME -o
jsonpath="{.spec.containers[0].ports[0].containerPort}")
  echo "Visit http://127.0.0.1:8080 to use your application"
  kubectl --namespace default port-forward $POD_NAME 8080:$CONTAINER_PORT
```

Pod is up and running :

```
$kubectl get pods
NAME                    READY   STATUS    RESTARTS   AGE
ping-5989f667d5-624mc   1/1     Running   0          1m02s
```

After forwarding the port I was able to see the app running **[3]**.

-----------CHALLENGE 1------------

“Modify the Ping Helm Chart to deploy the application on the following restrictions:
Isolate specific node groups forbidding the pods scheduling in these node groups.

Ensure that a pod will not be scheduled on a node that already has a pod of the same type.Ensure that Pods are deployed across different availability zones. Ensure that another random service is up before applying the manifests.”

Isolate Specific Node Groups :

I will prevent Prevent pods from scheduling on specific node groups by using node affinity[4]. In templates/deployment.yaml, I updated the spec.template.spec section to include affinity. Here's the modified section:

```
spec:
  {{- with .Values.nodeSelector }}
  nodeSelector:
    {{- toYaml . | nindent 8 }}
  {{- end }}
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: "node-group"
            operator: NotIn
            values:
              {{- toYaml .Values.forbiddenNodeGroups | nindent 12 }}
```

I updated values.yaml, I added a new configuration field for forbiddenNodeGroups under values.yaml:

```
# List of node groups where pods are not allowed to be scheduled
forbiddenNodeGroups:
  - forbidden-group-1
  - forbidden-group-2
```

nodeAffinity: Specifies the scheduling rules for the pods.
requiredDuringSchedulingIgnoredDuringExecution: Ensures that pods cannot be scheduled on nodes with the specified labels (node-group in this case).
NotIn: Prevents scheduling on nodes with labels matching the specified values.

I upgraded the helm chart (first with a dry run) to test the changes:

```
$helm upgrade ping ./ --namespace default --dry-run –debug
```

No issues found I proceeded with the upgrade :

```
$helm upgrade ping ./ --namespace default
```

```
Release "ping" has been upgraded. Happy Helming!
NAME: ping
LAST DEPLOYED: Sat Jan 18 11:45:53 2025
NAMESPACE: default
STATUS: deployed
REVISION: 5
NOTES:
1. Get the application URL by running these commands:
  export POD_NAME=$(kubectl get pods --namespace default -l
"app.kubernetes.io/name=ping,app.kubernetes.io/instance=ping" -o jsonpath="{.items[0].metadata.name}")
  export CONTAINER_PORT=$(kubectl get pod --namespace default $POD_NAME -o
jsonpath="{.spec.containers[0].ports[0].containerPort}")
  echo "Visit http://127.0.0.1:8080 to use your application"
  kubectl --namespace default port-forward $POD_NAME 8080:$CONTAINER_PORT
```

```
$kubectl get pods -n default
NAME                   READY   STATUS    RESTARTS   AGE
ping-57d5b87fd6-2zfzb   1/1     Running   0          107s
```

```
$kubectl describe pods ping-57d5b87fd6-2zfzb -n default
See output in  [5]
```

```
$kubectl get pod ping-57d5b87fd6-2zfzb -n default -o yaml | grep -A 5 nodeAffinity
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: node-group
            operator: NotIn
```

The pod will not be scheduled on any node with `node-group` labels matching the forbidden values specified in values.yaml.

Let's test it further :

I temporarily labeled all the nodes with a forbidden value.

```
kubectl label node gke-cluster-1-default-pool-1b47b03f-0k57 node-group=forbidden-group-1
kubectl label node gke-cluster-1-default-pool-1b47b03f-1k33 node-group=forbidden-group-1
kubectl label no gke-cluster-1-default-pool-1b47b03f-vn8v node-group=forbidden-group-2


node/gke-cluster-1-default-pool-1b47b03f-0k57 labeled
…
…
```

Checking the current pod:

```
kubectl get pod ping-57d5b87fd6-7nm4m -n default -o wide
NAME                   READY   STATUS    RESTARTS   AGE   IP           NODE                                        NOMINATED
NODE   READINESS GATES
ping-57d5b87fd6-7nm4m 1/1     Running   0          17m   10.100.1.5   gke-cluster-1-default-pool-1b47b03f-1k33   <none>
<none>
```

I forced rescheduling by deleting and recreating the pod:

```
kubectl delete pod ping-57d5b87fd6-7nm4m -n default
pod "ping-57d5b87fd6-7nm4m" deleted
```

Checking the current pod I can see :

```
kubectl get pod ping-57d5b87fd6-m7s7n -n default -o wide

NAME                   READY   STATUS    RESTARTS   AGE     IP       NODE     NOMINATED NODE   READINESS
GATES
ping-57d5b87fd6-m7s7n   0/1     Pending   0          2m16s   <none>   <none>   <none>           <none>
```

It's on pending status as expected, as pod didn't match the node affinity rules.

```
$kubectl describe pod ping-57d5b87fd6-m7s7n -n default

Warning  FailedScheduling   3m16s  default-scheduler   0/3 nodes are available: 3 node(s) didn't match Pod's node
affinity/selector. preemption: 0/3 nodes are available: 3 Preemption is not helpful for scheduling.
```

Full output here **[6].**

Test finished, affinity is working I will remove the node labels:

```
$kubectl label node gke-cluster-1-default-pool-1b47b03f-0k57 node-group-
node/gke-cluster-1-default-pool-1b47b03f-0k57 unlabeled
$kubectl label node gke-cluster-1-default-pool-1b47b03f-1k33 node-group-
$kubectl label no gke-cluster-1-default-pool-1b47b03f-vn8v node-group-
```

Challenge 1.2:

Ensure that a pod will not be scheduled on a node that already has a pod of the same type.

To ensure that a pod is not scheduled on a node that already has a pod of the same type, I can use a **pod anti-affinity** rule in the Helm chart.
I will add a podAntiAffinity rule under the affinity section of the deployment.yaml.
This ensures that pods with the same labels do not get scheduled on the same node.

Adding to deployment :

```
podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            - labelSelector:
                matchLabels:
                  app.kubernetes.io/name: {{ .Chart.Name }}
              topologyKey: "kubernetes.io/hostname"
```

- **podAntiAffinity:** Specifies rules to avoid scheduling pods of the same type on the same node.
- **requiredDuringSchedulingIgnoredDuringExecution:** Ensures strict enforcement during scheduling. Pods will not schedule on nodes that already have matching pods.
- **labelSelector:** Matches pods with specific labels. In this case, it ensures the rule applies to pods with the app.kubernetes.io/name label matching the chart's name.
- **topologyKey:** Defines the scope for the rule. "kubernetes.io/hostname" enforces this rule on a per-node basis.

Let’s render the yaml :

```
$helm template ./ --namespace default
$helm upgrade --install ping ./ --namespace default
```

Let’s increase the replica count in `values.yaml`

```
replicaCount: 4
helm upgrade --install ping ./ --namespace default
```

Checking the labels :

```
kubectl get pods -n default -o yaml | grep -A 10 "labels"
```

Label “ping” is correct.

Checking the pods :

```
kubectl get pods -o wide -n default
NAME                READY   STATUS    RESTARTS   AGE   IP           NODE                                       NOMINATED
NODE   READINESS GATES
ping-58b97c5bb4-2jtsg   0/1     Pending   0          76s   <none>       <none>                                     <none>
<none>
ping-58b97c5bb4-8t4d4   1/1     Running   0          76s   10.100.1.12   gke-cluster-1-default-pool-1b47b03f-1k33
<none>           <none>
ping-58b97c5bb4-x6xhp   1/1     Running   0          76s   10.100.0.8    gke-cluster-1-default-pool-1b47b03f-vn8v   <none>
<none>
ping-58b97c5bb4-xt6dx   1/1     Running   0          76s   10.100.2.18   gke-cluster-1-default-pool-1b47b03f-0k57   <none>
<none>
```

```
$kubectl describe pod ping-58b97c5bb4-2jtsg   -n default
```

```
Events:
  Type     Reason             Age    From                Message
  ----     ------             ----   ----                -------
  Warning  FailedScheduling   2m28s  default-scheduler   0/3 nodes are available: 3 node(s) didn't match pod anti-
affinity rules. preemption: 0/3 nodes are available: 3 No preemption victims found for incoming pod.
  Warning  FailedScheduling   2m27s  default-scheduler   0/3 nodes are available: 3 node(s) didn't match pod anti-
affinity rules. preemption: 0/3 nodes are available: 3 No preemption victims found for incoming pod.
  Normal   NotTriggerScaleUp  2m29s  cluster-autoscaler  pod didn't trigger scale-up:
```

Challenge 1.2 completed. Pod anti-affinity rules are working.

Challenge 1.3

Ensure that the Pods are deployed across different availability zones.

To ensure pods are deployed across different availability zones, I will modify the deployment.yaml to include a **podAntiAffinity** rule based on the zone topology. Kubernetes uses the topology.kubernetes.io/zone label to represent the availability zone of each node.

Changes to `deployment.yaml` to include a `podAntiAffinity` rule targeting zones :

```
#Ensure pods are spread across nodes (anti-affinity based on hostname)
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            - labelSelector:
                matchLabels:
                  app.kubernetes.io/name: {{ .Chart.Name }}
              topologyKey: "kubernetes.io/hostname" # Avoid scheduling multiple pods of the same type on the same node
            #Avoid scheduling in the same zone
            - label selector:
                matchLabels:
                   app.kubernetes.io/name: {{ .Chart.Name }}
              topologyKey: "topology.kubernetes.io/zone"
```

I currently have a zonal cluster, I will create a new cluster to be able to test it.

In a zonal cluster it works as expected of 4 replicas only one is scheduled:

```
NAME                  READY   STATUS    RESTARTS   AGE
ping-6cb9b69695-6tdvb   0/1     Pending   0          83m
ping-6cb9b69695-7bw8f   1/1     Running   0          83m
ping-6cb9b69695-7p9gc   0/1     Pending   0          83m
ping-6cb9b69695-h88lv   0/1     Pending   0          83m
```

Trying it in a regional cluster **[7]**:

```
$helm install ping ./ --namespace default
```

```
NAME                  READY   STATUS    RESTARTS   AGE
ping-6cb9b69695-7nwm4   1/1     Running   0          6s
ping-6cb9b69695-7x6j8   0/1     Pending   0          6s
ping-6cb9b69695-w274f   1/1     Running   0          6s
ping-6cb9b69695-w6dxm   1/1     Running   0          6s
```

Expected result, one pod in each zone, the solution is working.

Challenge 1.4

Ensure that another random service is up before applying the manifests.

I will create a pre-install hook to check for the random service. `templates/hooks/check-random-service.yaml` that will ensure the random service is up before proceeding [8].

The helm chart structure will be like :

```
ping/
├── Chart.yaml
├── values.yaml
├── templates/
│   ├ deployment.yaml
│   ├ service.yaml
│   ├── hooks/
│   │   ├ check-random-service.yaml
```

I initially thought in creating the random service as a dependency of the ping helm chart and loading it in the Chart.yaml. However, after testing it as a dependency, the hook was created before the random-service creation. I ended up with a simple workaround, I created 2 separate Helm Releases, because I wanted to be completely sure that random service was created before the hook and the ping deployment.

1. Deployed random-service first:

```
$helm install random-service ./charts/random-service --namespace default –debug
```

This release contains the random service.

2. After the random service I deployed “ping”:

   This release contains the ping workload with a prehook that will check if the random workload is running (pre-install) and continue the ping deployment if it is running.

```
helm install ping ./ --namespace default --debug
```

This ensures random-service is fully deployed before the ping chart.

The random-service helm code is :
Deployment -> ping/charts/random-service/templates/deployment.yaml [9]
Service -> ping/charts/random-service/templates/service.yaml [10]

The hook code is :
ping/templates/hooks/check-random-service.yaml [8]

I tested it in the already created GKE cluster, the logic between the hook works and only if random-service is up , the ping workload is deployed.

Challenge 2

To automate the described process of copying Helm charts from the reference GCP Artifact Registry (reference.gcr.io) to another GCP Artifact Registry (instance.gcr.io), I'll provide an automation solution using **Terraform** module.

```
terraform-project/
└── main.tf                 # Main Terraform configuration file
└── variables.tf
└── terraform.tfvars
└── modules/
|   └── helm_chart_copy/
|       └── main.tf          # Module implementation
|       └── variables.tf      # Variables specific to the module
|       └── outputs.tf        # Outputs specific to the module
```

Root main.tf (Main Terraform configuration file where the module is invoked.)

```
terraform {
  required_providers {
    google = {
      source  = "hashicorp/google"
      version = "~> 4.0"
    }
  }

  required_version = ">= 1.3.0"
}

provider "google" {
  credentials = file(var.service_account_key)
  project     = var.project_id
  region      = var.region
}

module "copy_helm_charts" {
  source            = "./modules/helm_chart_copy"
  source_registry   = var.source_registry
  target_registry   = var.target_registry
  helm_chart_list   = var.helm_chart_list
  sa_email          = var.sa_email
}
```

Root Variables.tf (all configurable inputs in `main.tf` are parameterized, promoting reusability and flexibility)

```
variable "project_id" {
  description = "The ID of the Google Cloud project"
  type        = string
}

variable "region" {
  description = "The region for the Google Cloud resources"
  type        = string
}

variable "service_account_key" {
  description = "Path to the service account key JSON file"
  type        = string
}
```

```
variable "helm_chart_list" {
  description = "List of Helm charts to copy"
  type        = list(string)
}

variable "source_registry" {
  description = "The source registry for Helm charts"
  type        = string
  default     = "https://reference.gcr.io/helm"
}

variable "target_registry" {
  description = "The target registry for Helm charts"
  type        = string
}

variable "sa_email" {
  description = "The email of the service account to use for authentication"
  type        = string
}
```

Root Terraform.tfvars example file

```
project_id          = "your-project-id"
region              = "us-central1"
service_account_key = "/path/to/your/service-account-key.json"
helm_chart_list     = ["chart1", "chart2"]
source_registry     = "https://reference.gcr.io/helm"
target_registry     = "instance.gcr.io"
sa_email            = "your-service-account-email"
```

Module: modules/helm_chart_copy/main.tf

#The logic for copying Helm charts resides here.

```
variable "source_registry" {}
variable "target_registry" {}
variable "helm_chart_list" {
  type = list(string)
}
variable "sa_email" {}

resource "null_resource" "copy_helm_charts" {
  provisioner "local-exec" {
    command = <<EOT
    gcloud auth activate-service-account ${var.sa_email} --key-file=/path/to/key.json

    # Add source Helm repository
    helm repo add source ${var.source_registry}
    helm repo update

    for chart in ${join(" ", var.helm_chart_list)}; do
      # Pull the Helm chart from the source repository
      helm pull source/$chart --untar

      # Push the chart to the target Artifact Registry
      helm push $chart.tar.gz oci://${var.target_registry}
    done
    EOT
  }
}
```

Module: modules/helm_chart_copy/variables.tf

The modules/helm_chart_copy/variables.tf file defines variables specifically for the **module**, making the module independent and reusable.

```
variable "source_registry" {
  description = "Source Helm repository URL"
  type        = string
}
variable "target_registry" {
  description = "Target GCP Artifact Registry URL"
  type        = string
}
variable "helm_chart_list" {
  description = "List of Helm charts to copy"
  type        = list(string)
}
variable "sa_email" {
  description = "Service account email for permissions"
  type        = string
}
```

Module: modules/helm_chart_copy/outputs.tf (Optional)

Useful information about the process using an outputs file.

```
output "copied_helm_charts" {
  description = "List of Helm charts copied"
  value       = var.helm_chart_list
}
```

Module Code Explanation:

helm pull: This downloads the chart directly from the source Helm repository. It avoids the need for using fetch for OCI repositories.
Helm Repository Management: Adds and updates the Helm repository before pulling charts.
OCI Push: Assumes the target registry is OCI-compliant and pushes the charts accordingly.

How to use the module:

Replace placeholders with the correct values (source_registry, target_registry, etc.).
Ensure the service account (sa) has necessary roles in GCP:

> roles/artifactregistry.reader for the source repository.
> roles/artifactregistry.writer for the target repository.

Save this as a module and call it from the Terraform configuration.

Challenge 3

Create a Github workflow to allow installing helm chart from Challenge #1 using module from Challenge #2, into a GKE cluster (considering a preexisting resource group and cluster name).

I create the file structure:

```
.github/
└─ workflows/
   └─ install-helm-chart.yaml
```

.github/workflows/install-helm-chart.yaml.

The install-helm-chart.yaml workflow yaml content:

```
name: Deploy Helm Chart to GKE

on:
  workflow_dispatch: # Allows manual triggering of the workflow
  push:
    branches:
      - main  # Trigger on push to the main branch

jobs:
  deploy-helm:
    runs-on: ubuntu-latest

    env:
      GOOGLE_CREDENTIALS: ${{ secrets.GOOGLE_CREDENTIALS }} # GCP Service Account Key
      PROJECT_ID: ${{ secrets.GCP_PROJECT_ID }}            # GCP Project ID
      REGION: ${{ secrets.GCP_REGION }}                    # GCP Region
      CLUSTER_NAME: ${{ secrets.GKE_CLUSTER_NAME }}        # GKE Cluster Name

    steps:
    - name: Checkout Code
      uses: actions/checkout@v3

    - name: Set up Google Cloud SDK
      uses: google-github-actions/setup-gcloud@v1
      with:
        service_account_key: ${{ secrets.GOOGLE_CREDENTIALS }}
        project_id: ${{ secrets.GCP_PROJECT_ID }}

    - name: Authenticate with GKE
      run: |
        set -e
        gcloud container clusters get-credentials $CLUSTER_NAME --region $REGION --project $PROJECT_ID

    - name: Set up Terraform
      uses: hashicorp/setup-terraform@v2
      with:
        terraform_wrapper: true

    - name: Initialize Terraform
      run: terraform init

    - name: Plan Terraform Changes
run: |
        set -e
        terraform plan \
          -var="project_id=${{ secrets.GCP_PROJECT_ID }}" \
          -var="region=${{ secrets.GCP_REGION }}" \
          -var="cluster_name=${{ secrets.GKE_CLUSTER_NAME }}"
  - name: Apply Terraform Changes
run: |
        set -e
        terraform apply -auto-approve \
          -var="project_id=${{ secrets.GCP_PROJECT_ID }}" \
          -var="region=${{ secrets.GCP_REGION }}" \
          -var="cluster_name=${{ secrets.GKE_CLUSTER_NAME }}"


  - name: Verify Deployment
    run: |
      kubectl get pods -n default
```

Secrets required in the GitHub repository:

1. **GOOGLE_CREDENTIALS:**
   JSON key file for a GCP service account with the following permissions:
   Kubernetes Engine Admin
   Artifact Registry Reader
   Service Account User
2. **GCP_PROJECT_ID:**
   The GCP project ID where the cluster resides.
3. **GCP_REGION:**
   The region of the GKE cluster.
4. **GKE_CLUSTER_NAME:**
   The name of the existing GKE cluster.

To test the GitHub action:

```
$git add .github/workflows/install-helm-chart.yaml
$git commit -m "Add Helm chart deployment workflow"
$git push origin main
```

Select the workflow and click **Run Workflow**.

Final Notes

Changed the ping chart version to 0.1.1.
I created the helm package:
$ helm package.
ping-0.1.1.tgz

Included also the helm release for the random-service as random-service-1.0.0.tgz.
(you can find it also inside ping release in charts folder)

```
[1] $gcloud beta container --project "instant-jetty-435813-p4" clusters create "cluster-1" --
zone "us-central1-a" --tier "standard" --no-enable-basic-auth --cluster-version "1.30.8-
gke.1051000" --release-channel "regular" --machine-type "e2-medium" --image-type
"COS_CONTAINERD" --disk-type "pd-balanced" --disk-size "100" --metadata disable-legacy-
endpoints=true --scopes
"https://www.googleapis.com/auth/devstorage.read_only","https://www.googleapis.com/aut
h/logging.write","https://www.googleapis.com/auth/monitoring","https://www.googleapis.co
m/auth/servicecontrol","https://www.googleapis.com/auth/service.management.readonly","
https://www.googleapis.com/auth/trace.append" --num-nodes "3" --
logging=SYSTEM,WORKLOAD --
monitoring=SYSTEM,STORAGE,POD,DEPLOYMENT,STATEFULSET,DAEMONSET,HPA,CADVISOR,
KUBELET --enable-ip-alias --network "projects/instant-jetty-435813-
p4/global/networks/default" --subnetwork "projects/instant-jetty-435813-p4/regions/us-
central1/subnetworks/default" --no-enable-intra-node-visibility --default-max-pods-per-node
"110" --enable-ip-access --security-posture=standard --workload-vulnerability-
scanning=disabled --no-enable-master-authorized-networks --no-enable-google-cloud-access --
addons HorizontalPodAutoscaling,HttpLoadBalancing,GcePersistentDiskCsiDriver --enable-
autoupgrade --enable-autorepair --max-surge-upgrade 1 --max-unavailable-upgrade 0 --
binauthz-evaluation-mode=DISABLED --enable-managed-prometheus --enable-shielded-nodes
--node-locations "us-central1-a"
[2] kubectl cluster-info

Kubernetes control plane is running at https://104.154.163.47

GLBCDefaultBackend is running at https://104.154.163.47/api/v1/namespaces/kube-
system/services/default-http-backend:http/proxy

KubeDNS is running at https://104.154.163.47/api/v1/namespaces/kube-
system/services/kube-dns:dns/proxy

Metrics-server is running at https://104.154.163.47/api/v1/namespaces/kube-
system/services/https:metrics-server:/proxy
```

```
[3] {
  "host": {
    "hostname": "127.0.0.1",
    "ip": "::ffff:127.0.0.1",
    "ips": []
  },
  "http": {
    "method": "GET",
    "baseUrl": "",
    "originalUrl": "/?authuser=0",
    "protocol": "http"
  },
  "request": {
    "params": {
      "0": "/"
    },
    "query": {
      "authuser": "0"
    },
    "cookies": {

    },
    "body": {

    },
    "headers": {
      "host": "127.0.0.1:8080",
      "user-agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36",
      "accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7",
      "accept-encoding": "gzip, deflate, br, zstd",
      "accept-language": "es-ES,es;q=0.9,ca;q=0.8",
      "referer": "https://shell.cloud.google.com/",
      "sec-ch-ua": "\"Google Chrome\";v=\"131\", \"Chromium\";v=\"131\", \"Not_A Brand\";v=\"24\"",
      "sec-ch-ua-mobile": "?0",
      "sec-ch-ua-platform": "\"Windows\"",
      "sec-fetch-dest": "document",
      "sec-fetch-mode": "navigate",
      "sec-fetch-site": "cross-site",
      "sec-user-ip": "10.0.0.32",
      "upgrade-insecure-requests": "1",
      "x-forwarded-host": "8080-cs-848137978281-default.cs-europe-west1-onse.cloudshell.dev"
    }
  },
  "environment": {
    "PATH": "/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",
    "HOSTNAME": "ping-5989f667d5-624mc",
    "NODE_VERSION": "16.16.0",
    "YARN_VERSION": "1.22.19",
    "KUBERNETES_PORT_443_TCP": "tcp://34.118.224.1:443",
    "PING_SERVICE_PORT_HTTP": "80",
    "PING_PORT": "tcp://34.118.229.251:80",
    "PING_PORT_80_TCP_ADDR": "34.118.229.251",
    "KUBERNETES_PORT": "tcp://34.118.224.1:443",
    "KUBERNETES_PORT_443_TCP_ADDR": "34.118.224.1",
    "PING_PORT_80_TCP_PROTO": "tcp",
    "PING_PORT_80_TCP_PORT": "80",
    "KUBERNETES_SERVICE_PORT": "443",
    "KUBERNETES_SERVICE_PORT_HTTPS": "443",
    "PING_SERVICE_HOST": "34.118.229.251",
    "PING_SERVICE_PORT": "80",
    "KUBERNETES_SERVICE_HOST": "34.118.224.1",
    "KUBERNETES_PORT_443_TCP_PROTO": "tcp",
    "KUBERNETES_PORT_443_TCP_PORT": "443",
    "PING_PORT_80_TCP": "tcp://34.118.229.251:80",
    "HOME": "/root"
  }
}
```

[4]: https://kubernetes.io/docs/concepts/scheduling-eviction/taint-and-toleration/

```
[5] Name:         ping-57d5b87fd6-2zfzb
Namespace:      default
Priority:       0
Service Account:  ping
Node:           gke-cluster-1-default-pool-1b47b03f-1k33/10.128.0.34
Start Time:     Sat, 18 Jan 2025 11:45:55 +0000
Labels:         app.kubernetes.io/instance=ping
             app.kubernetes.io/name=ping
             pod-template-hash=57d5b87fd6
Annotations:    <none>
Status:         Running
IP:           10.100.1.5
IPs:
  IP:         10.100.1.5
Controlled By:  ReplicaSet/ping-57d5b87fd6
Containers:
  ping:
    Container ID:   containerd://9e1d41df62e9183e353b77cd88eb1010b339ccb8c45c9aae8049640a6e29fc55
    Image:          ealen/echo-server:0.7.0
    Image ID:       docker.io/ealen/echo-server@sha256:086cba978dd74f4bbdbe91230c929d64b77201767fcec07d12697ff28abffbf9
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Sat, 18 Jan 2025 11:45:56 +0000
    Ready:          True
    Restart Count:  0
    Liveness:       http-get http://:http/ delay=0s timeout=1s period=10s #success=1 #failure=3
    Readiness:      http-get http://:http/ delay=0s timeout=1s period=10s #success=1 #failure=3
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-q64tr (ro)
Conditions:
  Type                    Status
  PodReadyToStartContainers   True
  Initialized               True
  Ready                     True
  ContainersReady           True
  PodScheduled              True
Volumes:
  kube-api-access-q64tr:
    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:           <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
  ----    ------     ----  ----              -------
  Normal  Scheduled  3m    default-scheduler  Successfully assigned default/ping-57d5b87fd6-2zfzb to gke-cluster-1-default-pool-
1b47b03f-1k33
  Normal  Pulled     3m    kubelet           Container image "ealen/echo-server:0.7.0" already present on machine
  Normal  Created    3m    kubelet           Created container ping
  Normal  Started    3m    kubelet           Started container ping
```

```
[6]: Events:
Name:         ping-57d5b87fd6-m7s7n
Namespace:      default
Priority:       0
Service Account:  ping
Node:           <none>
Labels:         app.kubernetes.io/instance=ping
             app.kubernetes.io/name=ping
             pod-template-hash=57d5b87fd6
Annotations:    cloud.google.com/cluster_autoscaler_unhelpable_since: 2025-01-18T12:09:42+0000
             cloud.google.com/cluster_autoscaler_unhelpable_until: Inf
Status:         Pending
IP:
IPs:          <none>
Controlled By:   ReplicaSet/ping-57d5b87fd6
Containers:
  ping:
    Image:       ealen/echo-server:0.7.0
    Port:        80/TCP
    Host Port:   0/TCP
    Liveness:    http-get http://:http/ delay=0s timeout=1s period=10s #success=1 #failure=3
    Readiness:   http-get http://:http/ delay=0s timeout=1s period=10s #success=1 #failure=3
    Environment: <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-4fp42 (ro)
Conditions:
  Type           Status
  PodScheduled   False
Volumes:
  kube-api-access-4fp42:
    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:              <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   3m16s  default-scheduler   0/3 nodes are available: 3 node(s) didn't match Pod's node affinity/selector. preemption: 0/3 nodes are available: 3 Preemption is not helpful for scheduling.
  Warning  FailedScheduling   3m15s  default-scheduler   0/3 nodes are available: 3 node(s) didn't match Pod's node affinity/selector. preemption: 0/3 nodes are available: 3 Preemption is not helpful for scheduling.
  Normal   NotTriggerScaleUp  3m17s  cluster-autoscaler  pod didn't trigger scale-up:
```

[7]

```
gcloud beta container --project "instant-jetty-435813-p4" clusters create "cluster-2" --region "us-central1" --tier "standard" --no-enable-basic-auth --cluster-version "1.30.8-gke.1051000" --release-channel "regular" --machine-type "e2-medium" --image-type "COS_CONTAINERD" --disk-type "pd-balanced" --disk-size "100" --metadata disable-legacy-endpoints=true --scopes "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/servicecontrol","https://www.googleapis.com/auth/service.management.readonly","https://www.googleapis.com/auth/trace.append" --num-nodes "1" --logging=SYSTEM,WORKLOAD --monitoring=SYSTEM,STORAGE,POD,DEPLOYMENT,STATEFULSET,DAEMONSET,HPA,CADVISOR,KUBELET --enable-ip-alias --network "projects/instant-jetty-435813-p4/global/networks/default" --subnetwork "projects/instant-jetty-435813-p4/regions/us-central1/subnetworks/default" --no-enable-intra-node-visibility --default-max-pods-per-node "110" --enable-ip-access --security-posture=standard --workload-vulnerability-scanning=disabled --no-enable-master-authorized-networks --no-enable-google-cloud-access --addons HorizontalPodAutoscaling,HttpLoadBalancing,GcePersistentDiskCsiDriver --enable-autoupgrade --enable-autorepair --max-surge-upgrade 1 --max-unavailable-upgrade 0 --binauthz-evaluation-mode=DISABLED --enable-managed-prometheus --enable-shielded-nodes
```

`[8]: check-random-service.yaml`

```
apiVersion: batch/v1
kind: Job
metadata:
  name: {{ .Release.Name }}-check-random-service
  annotations:
    "helm.sh/hook": pre-install,pre-upgrade
    "helm.sh/hook-delete-policy": hook-succeeded
spec:
  template:
    spec:
      containers:
        - name: check-random-service
          image: busybox
          command:
            - /bin/sh
            - -c
            - |
              for i in {1..60}; do
              if wget -qO- random-service-random1.default.svc.cluster.local:80; then
              echo "Random service is up!";
                  exit 0;
                fi;
                echo "Waiting for random service...";
                sleep 5;
              done;
              echo "Random service is not ready.";
              exit 1;
      restartPolicy: Never
  backoffLimit: 1
```

[9] random-service deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ .Release.Name }}-random1
  labels:
    app.kubernetes.io/name: {{ .Chart.Name }}
    app.kubernetes.io/instance: {{ .Release.Name }}
spec:
  replicas: {{ .Values.replicaCount }}
  selector:
    matchLabels:
      app.kubernetes.io/name: {{ .Chart.Name }}
  template:
    metadata:
      labels:
        app.kubernetes.io/name: {{ .Chart.Name }}
    spec:
      containers:
        - name: random-service
          image: "us-docker.pkg.dev/google-samples/containers/gke/hello-app:2.0"
```

10: Random-service service.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: {{ .Release.Name }}-random1
  labels:
    app.kubernetes.io/name: {{ .Chart.Name }}
    app.kubernetes.io/instance: {{ .Release.Name }}
spec:
  type: ClusterIP
  ports:
    - port: 80
      targetPort: 8080
      protocol: TCP
  selector:
    app.kubernetes.io/name: {{ .Chart.Name }}
```