

Starting with GKE

1- Setting a default compute zone

Your [compute zone](#) is an approximate regional location in which your clusters and their resources live. For example, `us-central1-a` is a zone in the `us-central1` region. Start a new session in Cloud Shell and run the following command to set your default compute zone to `us-central1-a`:

```
gcloud config set compute/zone us-central1-a
```

You receive the following output:

```
ahmedhosni_contact@cloudshell:~ (widigital-ci)$ gcloud config set compute/zone us-central1-a
Updated property [compute/zone].
```

2- Creating a Kubernetes Engine cluster

A [cluster](#) consists of at least one *cluster master* machine and multiple worker machines called *nodes*. Nodes are [Compute Engine virtual machine \(VM\) instances](#) that run the Kubernetes processes necessary to make them part of the cluster.

To create a cluster, run the following command, replacing `[CLUSTER-NAME]` with the name you choose for the cluster (for example `my-cluster`). Cluster names must start with a letter, end with an alphanumeric, and cannot be longer than 40 characters.

```
gcloud container clusters create [CLUSTER-NAME]
```

You can ignore any warnings in the output. It might take several minutes to finish creating the cluster. Soon after you should receive a similar output:

```

ahmedhosni_contact@cloudshell:~ (widual-c1) $ gcloud container clusters create ahmed-cluster
WARNING: Currently VPC-native is not the default mode during cluster creation. In the future, this will become the default mode and can be disabled using '--no-enable-ip-alias' flag. Use '--[n
o-enable-ip-alias]' flag to suppress this warning.
WARNING: Newly created clusters and node-pools will have node auto-upgrade enabled by default. This can be disabled using the '--no-enable-autoupgrade' flag.
WARNING: Starting with version 1.18, clusters will have shielded GKE nodes by default.
WARNING: Your Pod address range ('--cluster-ipv4-cidr') can accommodate at most 1008 node(s).
This will enable the autorepair feature for nodes. Please see https://cloud.google.com/kubernetes-engine/docs/node-auto-repair for more information on node autorepairs.
Creating cluster 'ahmed-cluster' in us-central1-a... Cluster is being health-checked (master is healthy)...done.
Created [https://container.googleapis.com/v1/projects/widual-c1/zones/us-central1-a/clusters/ahmed-cluster].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload/_gcloud/us-central1-a/ahmed-cluster?project=widual-c1
kubeconfig entry generated for ahmed-cluster.
NAME          LOCATION  MASTER_VERSION  MASTER_IP      MACHINE_TYPE  NODE_VERSION  NUM_NODES  STATUS
ahmed-cluster  us-central1-a  1.15.12-gke.2   35.232.16.90   n1-standard-1  1.15.12-gke.2  3          RUNNING
ahmedhosni_contact@cloudshell:~ (widual-c1) $

```

3- Get authentication credentials for the cluster

After creating your cluster, you need to get authentication credentials to interact with the cluster.

To authenticate the cluster run the following command, replacing [CLUSTER-NAME] with the name of your cluster:

```
gcloud container clusters get-credentials [CLUSTER-NAME]
```

You should receive a similar output:

```

ahmedhosni_contact@cloudshell:~ (widual-c1) $ gcloud container clusters get-credentials ahmed-cluster
Fetching cluster endpoint and auth data.
kubeconfig entry generated for ahmed-cluster.

```

4- Deploying an application to the cluster

Now that you have created a cluster, you can deploy a [containerized application](#) to it.

For this lab you'll run hello-app in your cluster.

Kubernetes Engine uses Kubernetes objects to create and manage your cluster's resources. Kubernetes provides the [Deployment](#) object for deploying stateless applications like web servers. [Service](#) objects define rules and load balancing for accessing your application from the Internet.

Run the following [kubectl create](#) command in Cloud Shell to create a new Deployment hello-server from the hello-app container image:

```
kubectl create deployment hello-server --image=gcr.io/google-samples/hello-app:1.0
```

You should receive the following output:

```
ahmedhosni_contact@cloudshell:~ (widiigital-ci) $ kubectl create deployment hello-server --image=gcr.io/google-samples/hello-app:1.0
deployment.apps/hello-server created
ahmedhosni_contact@cloudshell:~ (widiigital-ci) $
```

This Kubernetes command creates a Deployment object that represents hello-server. In this case, `--image` specifies a container image to deploy. The command pulls the example image from a [Google Container Registry](#) bucket. `gcr.io/google-samples/hello-app:1.0` indicates the specific image version to pull. If a version is not specified, the latest version is used.

Now create a Kubernetes Service, which is a Kubernetes resource that lets you expose your application to external traffic, by running the following `kubectl expose` command:

```
kubectl expose deployment hello-server --type=LoadBalancer --port 8080
```

In this command:

- `--port` specifies the port that the container exposes.
- `type="LoadBalancer"` creates a Compute Engine load balancer for your container.

You should receive the following output:

```
ahmedhosni_contact@cloudshell:~ (widiigital-ci) $ kubectl expose deployment hello-server --type=LoadBalancer --port 8080
service/hello-server exposed
ahmedhosni_contact@cloudshell:~ (widiigital-ci) $
```

Inspect the hello-server Service by running `kubectl get`:

```
kubectl get service
```

You should receive a similar output:

```
ahmedhosni_contact@cloudshell:~ (widiigital-ci) $ kubectl get service
NAME                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
hello-server        LoadBalancer  10.47.243.142  <pending>      8080:32357/TCP   37s
kubernetes          ClusterIP     10.47.240.1   <none>         443/TCP          4m41s
```

Note: It might take a minute for an external IP address to be generated. Run the above command again if the EXTERNAL-IP column is in "pending" status.

From this command's output, copy the Service's external IP address from the EXTERNAL IP column. View the application from your web browser using the external IP address with the exposed port:

`http://[EXTERNAL-IP]:8080`

Your page should resemble the following:



5- Clean Up

Run the following to delete the cluster:

```
gcloud container clusters delete [CLUSTER-NAME]
```

When prompted, type **Y** to confirm. Deleting the cluster can take a few minutes. For more information on deleted Google Kubernetes Engine clusters, view the [documentation](#).

```
ahmedhosni_contact@cloudshell:~ (widigital-ci)$ gcloud container clusters delete ahmed-cluster
The following clusters will be deleted.
- [ahmed-cluster] in [us-centrall1-a]

Do you want to continue (Y/n)? Y

Deleting cluster ahmed-cluster...!
```