Lab Overview: Getting Started with Container Engine

Overview

In this lab, you use the Google Developer Console to create a container cluster - a group of Compute Engine instances that provide the foundation for Google Container Engine-managed deployments. You also use the Kubernetes command-line tool (kubectl) to deploy a pod with two containers running the Guestbook application.

Duration

The timing of this lab is as follows:

Component	Timing
Introduction	5 minutes
Lab	10 minutes
Total	**15 minutes**

What you need

To complete this lab, you need:

- The Google Cloud SDK installed and configured on your labs instance
- A Google Cloud project and project ID

What you will learn

In this lab, you will:

- Use the Google Developer Console to create a container cluster
- Use the kubectl command-line tool to deploy a pod with two containers running the Guestbook application

Lab: Getting Started with Container Engine

Overview In this lab you:

- Use the Google Developer Console to create a container cluster
- Use the kubectl command-line tool to deploy a pod with two containers running the Guestbook application

---

Create a cluster To create a container cluster:

Step	Action
1	Access the Google Developers Console by typing the following URL in your browser: https://console.developers.google.com
2	If you have more than one project, click the **cp100** project. Otherwise proceed to the next step.
3	In the navigator pane, click the **Gallery** icon (to the left of Google Developers Console at the top of the page).
4	Click **Container Engine > Container clusters**.
5	Click **Create a container cluster** in the dialog box.
6	On the **Create a new container cluster** page, in the **Name** field, type: **cp100**.
7	For **ZONE**, choose the same zone you used when configuring the Cloud SDK (for example: **us-central1-b** or **europe-west1-c**).
8	For **Machine type**, accept the default value: **n1-standard-1**.
9	For **Cluster size**, change the value to **1** (this will create one worker node in addition to the master node). Note that the number of Total cores and Total memory change when you change the number of nodes.

10	Accept the remaining default values and click **Create**.
11	Clicking Create opens the **Activities** pop-up window. This window shows the status of the cluster you created. The creation process may take several minutes.
12	Leave the Google Developers Console open.

Clone the project

When you created your Compute Engine labs instance, you installed Git on the machine. In this section of the lab, you use Git to clone the repository containing your application.

To clone the project:

Step	Action
1	In the navigator pane, click the **Gallery** icon (to the left of Google Developers Console at the top of the page).
2	Click **Compute Engine > VM instances**.
3	To the right of the **cp100-labs** instance, in the **Connect** column, click **SSH**.
4	Type the following command to clone the code repository. **`git clone \`** **`https://github.com/GoogleCloudPlatformTraining/`** **`cp100-container-engine-python.git`**
5	Leave the SSH connection open.

Deploy a pod

To deploy the Guestbook application pod:

Step	Action
1	Type the following command to view the current configuration of the Cloud SDK.

	gcloud config list
2	Verify that the account is set to your email address and that the project ID matches the ID of the project (cp100) you created earlier.
3	Type the following command to configure the Container Engine cluster. **gcloud config set container/cluster cp100**
4	If your cluster is created using the Developers Console instead of the SDK, or if you create it with gcloud from a different machine, you must run the get-credentials command to make your credentials available to kubectl. Type the following command to retrieve your credentials. Replace <zone> with the zone you chose when you created the Container Engine cluster. **gcloud container clusters get-credentials --zone <zone> cp100** The output should be similar to: **kubeconfig entry generated for cp100**
5	Navigate to the **cp100-container-engine-python** folder by typing the following command. This is the Git repository you cloned earlier. **cd cp100-container-engine-python**
6	Type the following command to examine the project's resource file. **cat guestbook.yaml** Notice it includes two containers, one to host a frontend Python Flask application and another to host the Redis backend.
7	Type the following command to create a Replication Controller with a single Pod containing 2 Docker containers. **kubectl create -f guestbook.yaml**

8	Type the following kubectl command to check the status of your pod. **`kubectl get pods`** You should see your Guestbook pod listed and the status should be 'Running.' It may take a moment for the status to change from 'Pending' to 'Running'. Repeat the 'get pods' command until the status is 'Running'. **Note**: The Guestbook pod will have a name similar to: guestbook-9xx74. To retrieve the status of the Guestbook pod, you may also type: **`kubectl get pods guestbook-9xx74`**
9	By default, the pod is only accessible within the cluster using its internal IP address. In order to make the Guestbook container accessible from outside the kubernetes virtual network, you have to expose the pod as a kubernetes service. Type the following command to expose the Replication Controller to external traffic by creating an external network load balancer. The --create-external-load-balancer=true flag creates an external IP on which the pod accepts traffic. **`kubectl expose rc guestbook --port=80 --create-external-load-balancer=true`** The kubernetes master creates the load balancer and related Compute Engine forwarding rules and target pools. The output of the command should indicate that the load-balanced service was created.
10	After a few moments, the external IP of the load balancer is listed in the IP(s) column of the service. Type the following command to retrieve the IP addresses. **`kubectl get services guestbook`** Notice there are 2 IP(s) listed, both serving port 80. One

	is the internal IP that other pods in the cluster can use to talk to your service (likely beginning with 10.); the other is the external load-balanced IP. Note the external IP address. You use it later. **Note**: It may take a moment for both IP addresses to appear. Issue the 'kubectl get services guestbook' command until both addresses appear.
11	Switch to the Google Developers Console.
12	If necessary, click **Compute Engine > VM Instances**.
13	You should see the node instance listed. In the **Name** column, click the instance name link.
14	Click **Edit**, and in the **Firewalls** section, check **Allow HTTP traffic**.
15	Click **Save**. When the action is complete, the Activities window will update.
16	Open a new tab in your browser.
17	In the address bar, type the external IP address you recorded previously.
18	(Optional) When the Guestbook application loads, create a test entry. When you are finished, close the tab.
19	Switch to your SSH window and type **`exit`** to close it.

Clean up To clean up the resources used in the lab:

Step	Action
1	Switch to the Google Developer Console window.
2	In the navigator pane, click the **Gallery** icon (to the left of Google Developers Console at the top of the page).
3	Click **Container engine > Container clusters**.

4	Click the check box to the left of the cluster name. This will activate the Delete button at the top of the page.
5	Click **Delete** to remove the Container Cluster and the Compute Engine instances.
6	Click the **Gallery** icon and then click **Networking > Firewall rules**.
7	If you have more than one network, in the **Networks** section, in the **Name** column, click the **default** link. Otherwise, skip to the next step.
8	In the **Firewall rules** section, check the http firewall rule that begins with k8s and then click **Delete**. The target tags for this firewall rule will include the name of your cluster node (for example, gke-cp100-c65xxxx0-node).
9	Click **Network load balancing**.
10	Check the forwarding rule for port 80 and then click **Delete**.