Management Tools (https://cloud.google.com/products/management/)
Cloud Shell (https://cloud.google.com/shell/) Documentation

Preview and deploy a Kubernetes Engine application

This page walks through how to quickly deploy a Kubernetes Engine app using Cloud Shell.

Take the following steps to enable the Kubernetes Engine API:

1. Visit the Kubernetes Engine page (https://console.cloud.google.com/projectselector/kubernetes) in the Google Cloud Console.
2. Create or select a project.
3. Wait for the API and related services to be enabled. This can take several minutes.
4. Make sure that billing is enabled for your Google Cloud project. Learn how to confirm billing is enabled for your project (https://cloud.google.com/billing/docs/how-to/modify-project).

Deploying a Kubernetes application

Cloud Shell comes pre-installed with `gcloud` and `kubectl` command-line tools. You can use `gcloud` and `kubectl` without any additional setup!

1. **Click the Activate Cloud Shell button at the top of the Console window:**

   This launches a Cloud Shell session in a frame at the bottom of the Console (http://console.cloud.google.com).

2. **Set your defaults:**

   To set your default compute zone (https://cloud.google.com/compute/docs/zones#available) with a desired geographical compute zone, run:

   ```
   gcloud config set compute/zone [COMPUTE_ZONE]
   ```

   If you've set your project on the Console, it'll automatically be set in Cloud Shell. If you haven't, set your project ID (https://support.google.com/cloud/answer/6158840), by running:

   ```
   gcloud config set project [CLOUD_SHELL_PROJECT_ID]
   ```

3. **Clone the sample app:**

   To clone the sample application and go to its directory, run:

   ```
   git clone https://github.com/GoogleCloudPlatform/kubernetes-engine-samples
   cd kubernetes-engine-samples/hello-app
   ```

   `hello-app` is a sample Go web server application and comes with a Dockerfile needed to build the Docker image of the application.

4. **Build the container image:**

   To build the container image of `hello-app` and tag it with the name `gcr.io/${CLOUD_SHELL_PROJECT_ID}/hello-app:v1` for uploading, run:

   ```
   docker build -t gcr.io/${CLOUD_SHELL_PROJECT_ID}/hello-app:v1 .
   ```

5. **Preview your application:**

   To preview your application in your browser, run:

   ```
   docker run --rm -p 8080:8080 gcr.io/${CLOUD_SHELL_PROJECT_ID}/hello-app:v1
   ```

   On Cloud Shell, click the Web Preview button  and choose 'Preview on port 8080'.

   Cloud Shell opens the preview URL on its proxy service in a new browser window.

6. **Edit your application:**

   Make a small change to `main.go` such as replacing "Hello, world!" with "Teapot!".

   Type **Ctrl + C** in your Cloud Shell session to stop the development server.

   Build and upload your image again. Now, when you preview your application by running it on Docker engine and clicking Web Preview, you'll see your changes reflected in your browser.

7. **Upload the image to Container Registry:**

   To upload the image to the Container Registry using the Docker command-line tool (which comes installed, authenticated, and configured with Cloud Shell), run:

   ```
   docker push gcr.io/${CLOUD_SHELL_PROJECT_ID}/hello-app:v1
   ```

8. **Create a cluster with a unique name:**

To create a cluster named `hello-cluster`, run:

```
gcloud container clusters create hello-cluster
```

This step might take several minutes to complete.

9. **Deploy an your application:**

   To create a new Deployment called `hello-server`, run:

   ```
   kubectl run hello-server --image gcr.io/google-samples/hello-app:1.0 --port 80
   ```

   The Deployment's pod runs the `hello-app` image in its container.

10. **Create a Service:**

    In order to expose your application to external traffic to allow users to use it, you'll need to create a Service.

    ```
    kubectl expose deployment hello-server --type LoadBalancer \
    --port 80 --target-port 8080
    ```

★ **Note:** Load balancers are billed per Compute Engine's load balancer pricing (https://cloud.google.com/compute/pricing#lb).

11. **Inspect the created service:**

    To get the external IP address of your service, run:

    ```
    kubectl get service hello-server
    ```

    From the command's output, copy the Service's external IP address from the `EXTERNAL-IP` column.

★ **Note:** It might take several minutes for the Service's external IP address to be populated. If the application's external IP is <pending>, run **`kubectl get`** again.

12. **Your containerized web application should be live!**

    View the application from your web browser using the external IP address from the previous step.

```
http://[EXTERNAL_IP]/
```

13. **Remember to clean up after to avoid unnecessary billing:**

    To delete the application's Service (and the Compute Engine load balancer you created), run:

    ```
    kubectl delete service hello-server
    ```

    Next, to delete your cluster, run:

    ```
    gcloud container clusters delete hello-cluster
    ```


Last updated December 3, 2019.