Cloud Run is a managed compute platform that enables you to run stateless containers that are invocable via HTTP requests. Cloud Run is serverless: it abstracts away all infrastructure management, so you can focus on what matters most — building great applications.

Cloud Run is built from Knative, letting you choose to run your containers either fully managed with Cloud Run, or in your Google Kubernetes Engine cluster with Cloud Run on GKE.

The goal of this lab is for you to build a simple containerized application image and deploy it to Cloud Run.
In this lab, you learn to:

- Enable the Cloud Run API.
- Create a simple Node.js application that can be deployed as a serverless, stateless container.
- Containerize your application and upload to Container Registry (now called "Artifact Registry.")
- Deploy a containerized application on Cloud Run.
- Delete unneeded images to avoid incurring extra storage charges.

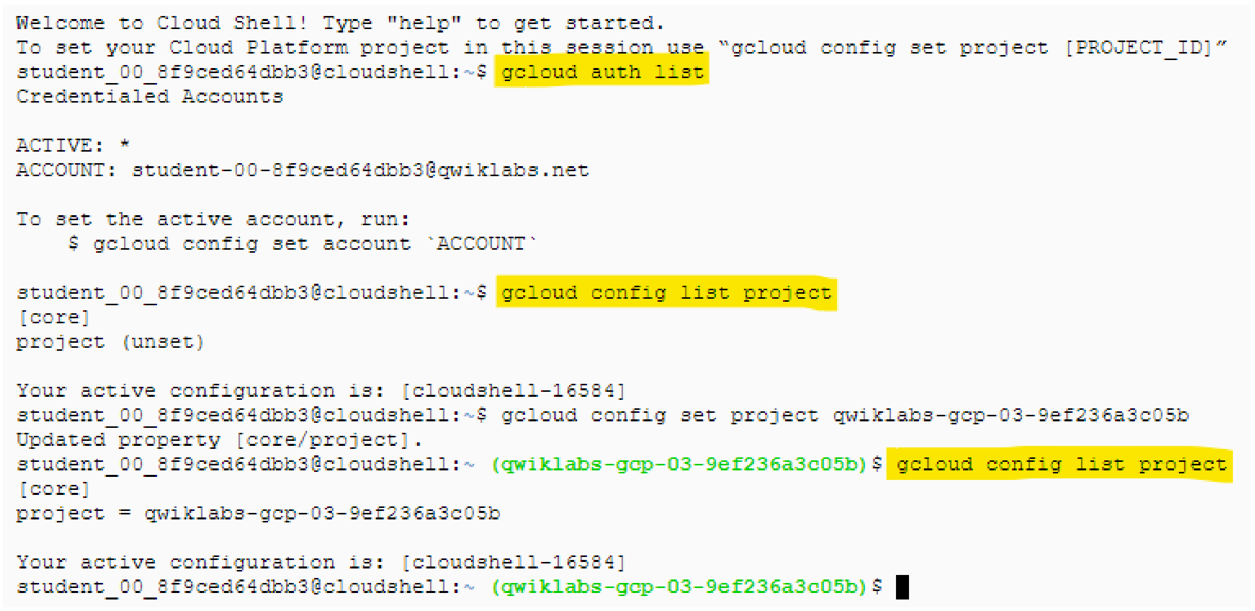
gcloud is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

- You can list the active account name with this command:

```
gcloud auth list
```

- You can list the project ID with this command:

```
gcloud config list project
```



```
Welcome to Cloud Shell! Type "help" to get started.
To set your Cloud Platform project in this session use "gcloud config set project [PROJECT_ID]"
student_00_8f9ced64dbb3@cloudshell:~$ gcloud auth list
Credentialed Accounts

ACTIVE: *
ACCOUNT: student-00-8f9ced64dbb3@qwiklabs.net

To set the active account, run:
    $ gcloud config set account `ACCOUNT`

student_00_8f9ced64dbb3@cloudshell:~$ gcloud config list project
[core]
project (unset)

Your active configuration is: [cloudshell-16584]
student_00_8f9ced64dbb3@cloudshell:~$ gcloud config set project qwiklabs-gcp-03-9ef236a3c05b
Updated property [core/project].
student_00_8f9ced64dbb3@cloudshell:~ (qwiklabs-gcp-03-9ef236a3c05b)$ gcloud config list project
[core]
project = qwiklabs-gcp-03-9ef236a3c05b

Your active configuration is: [cloudshell-16584]
student_00_8f9ced64dbb3@cloudshell:~ (qwiklabs-gcp-03-9ef236a3c05b)$
```

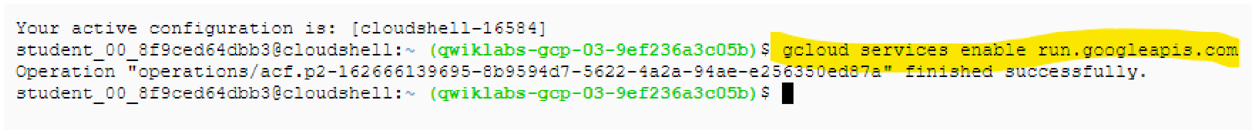
Basic Linux Commands

Below you will find a reference list of a few very basic Linux commands which may be included in the instructions or code blocks for this lab.

Command -->	Action	.	Command -->	Action
mkdir (*make directory*)	create a new folder	.	**cd** (*change directory*)	change location to another folder
ls (*list*)	list files and folders in the directory	.	**cat** (*concatenate*)	read contents of a file without using an editor
apt-get update	update package manager library	.	**ping**	signal to test reachability of a host
mv (*move*)	moves a file	.	**cp** (*copy*)	makes a file copy
pwd (*present working directory*)	returns your current location	.	**sudo** (*super user do*)	gives higher administration privileges

1. From Cloud Shell, enable the **Cloud Run API** :

```
gcloud services enable run.googleapis.com
```



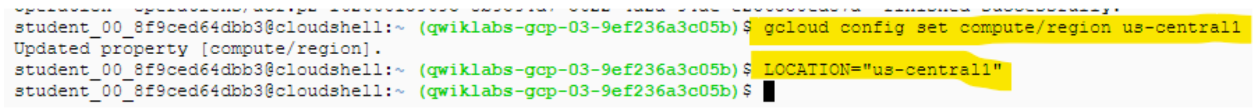
```
Your active configuration is: [cloudshell-16584]
student_00_8f9ced64dbb3@cloudshell:~ (qwiklabs-gcp-03-9ef236a3c05b)$ gcloud services enable run.googleapis.com
Operation "operations/acf.p2-162666139695-8b9594d7-5622-4a2a-94ae-e256350ed87a" finished successfully.
student_00_8f9ced64dbb3@cloudshell:~ (qwiklabs-gcp-03-9ef236a3c05b)$
```

3. Set the compute region:

```
gcloud config set compute/region us-central1
```

4. Create a LOCATION environment variable:

```
LOCATION="us-central1"
```



```
student_00_8f9ced64dbb3@cloudshell:~ (qwiklabs-gcp-03-9ef236a3c05b)$ gcloud config set compute/region us-central1
Updated property [compute/region].
student_00_8f9ced64dbb3@cloudshell:~ (qwiklabs-gcp-03-9ef236a3c05b)$ LOCATION="us-central1"
student_00_8f9ced64dbb3@cloudshell:~ (qwiklabs-gcp-03-9ef236a3c05b)$
```

In this task, you will build a simple express-based NodeJS application which responds to HTTP requests.

1. In Cloud Shell create a new directory named `helloworld`, then move your view into that directory:

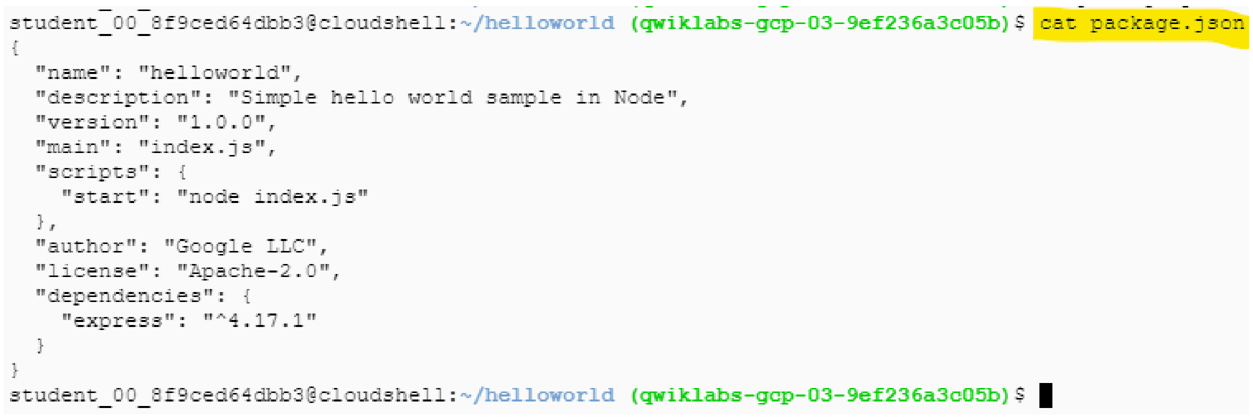
```
mkdir helloworld && cd helloworld
```

2. Create a `package.json` file, then add the following content to it:

```
nano package.json
```

```
{
  "name": "helloworld",
  "description": "Simple hello world sample in Node",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "start": "node index.js"
  },
  "author": "Google LLC",
  "license": "Apache-2.0",
  "dependencies": {
    "express": "^4.17.1"
  }
}
```

Most importantly, the file above contains a start script command and a dependency on the Express web application framework.

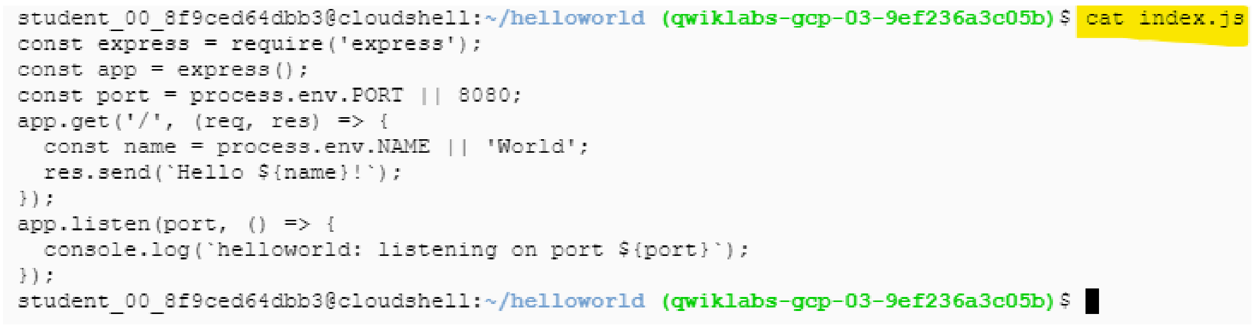


```
student_00_8f9ced64dbb3@cloudshell:~/helloworld (qwiklabs-gcp-03-9ef236a3c05b)$ cat package.json
{
  "name": "helloworld",
  "description": "Simple hello world sample in Node",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "start": "node index.js"
  },
  "author": "Google LLC",
  "license": "Apache-2.0",
  "dependencies": {
    "express": "^4.17.1"
  }
}
student_00_8f9ced64dbb3@cloudshell:~/helloworld (qwiklabs-gcp-03-9ef236a3c05b)$
```

4. Next, in the same directory, create a `index.js` file, and copy the following lines into it:

```
nano index.js
```

```
const express = require('express');
const app = express();
const port = process.env.PORT || 8080;
app.get('/', (req, res) => {
  const name = process.env.NAME || 'World';
  res.send(`Hello ${name}!`);
});
app.listen(port, () => {
  console.log(`helloworld: listening on port ${port}`);
});
```



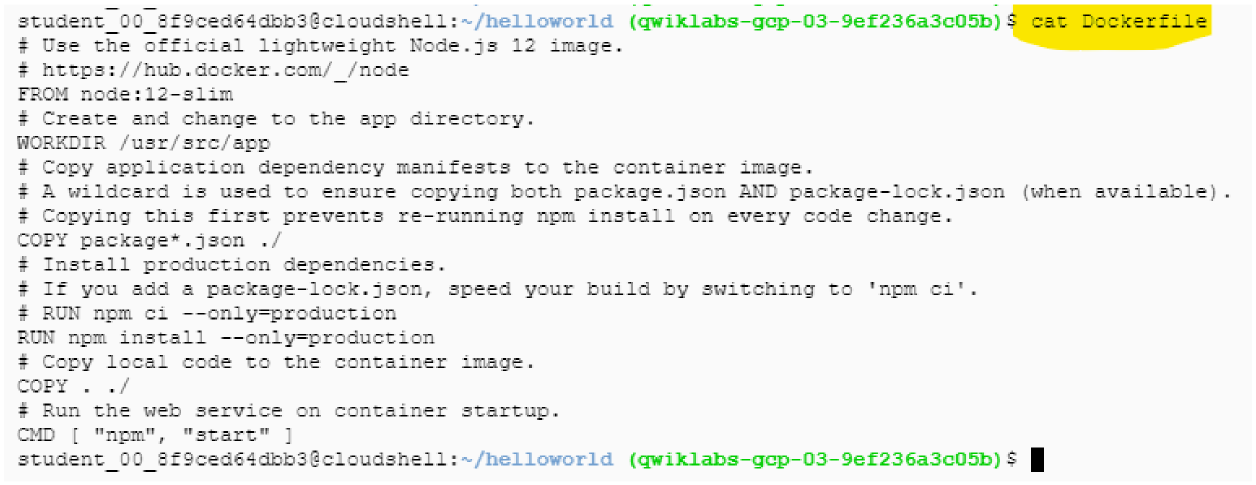
```
student_00_8f9ced64dbb3@cloudshell:~/helloworld (qwiklabs-gcp-03-9ef236a3c05b)$ cat index.js
const express = require('express');
const app = express();
const port = process.env.PORT || 8080;
app.get('/', (req, res) => {
  const name = process.env.NAME || 'World';
  res.send(`Hello ${name}!`);
});
app.listen(port, () => {
  console.log(`helloworld: listening on port ${port}`);
});
student_00_8f9ced64dbb3@cloudshell:~/helloworld (qwiklabs-gcp-03-9ef236a3c05b)$
```

This code creates a basic web server that listens on the port defined by the PORT environment variable. Your app is now finished and ready to be containerized and uploaded to Container Registry

1. To containerize the sample app, create a new file named `Dockerfile` in the same directory as the source files, and add the following content:

```
nano Dockerfile
```

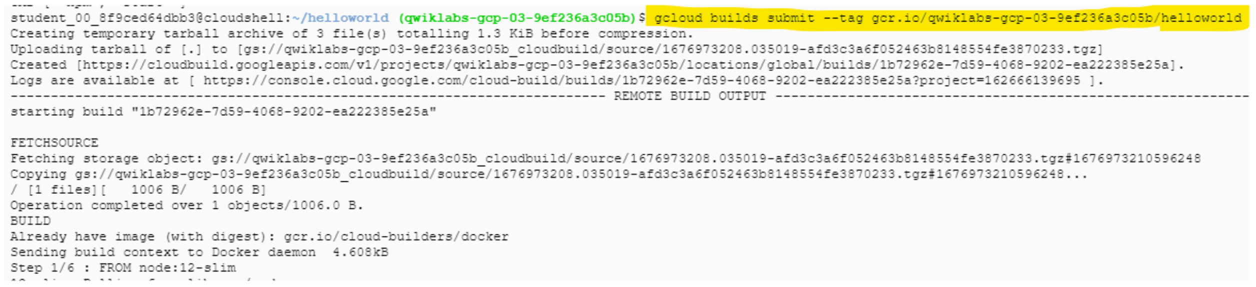
```
# Use the official lightweight Node.js 12 image.
# https://hub.docker.com/_/node
FROM node:12-slim
# Create and change to the app directory.
WORKDIR /usr/src/app
# Copy application dependency manifests to the container image.
# A wildcard is used to ensure copying both package.json AND package-lock.json (when available).
# Copying this first prevents re-running npm install on every code change.
COPY package*.json ./
# Install production dependencies.
# If you add a package-lock.json, speed your build by switching to 'npm ci'.
# RUN npm ci --only=production
RUN npm install --only=production
# Copy local code to the container image.
COPY . ./
# Run the web service on container startup.
CMD [ "npm", "start" ]
```



```
student_00_8f9ced64dbb3@cloudshell:~/helloworld (qwiklabs-gcp-03-9ef236a3c05b)$ cat Dockerfile
# Use the official lightweight Node.js 12 image.
# https://hub.docker.com/_/node
FROM node:12-slim
# Create and change to the app directory.
WORKDIR /usr/src/app
# Copy application dependency manifests to the container image.
# A wildcard is used to ensure copying both package.json AND package-lock.json (when available).
# Copying this first prevents re-running npm install on every code change.
COPY package*.json ./
# Install production dependencies.
# If you add a package-lock.json, speed your build by switching to 'npm ci'.
# RUN npm ci --only=production
RUN npm install --only=production
# Copy local code to the container image.
COPY . ./
# Run the web service on container startup.
CMD [ "npm", "start" ]
student_00_8f9ced64dbb3@cloudshell:~/helloworld (qwiklabs-gcp-03-9ef236a3c05b)$
```

2. Now, build your container image using Cloud Build by running the following command from the directory containing the `Dockerfile.` (Note the $GOOGLE_CLOUD_PROJECT environmental variable in the command, which contains your lab's Project ID):

gcloud builds submit --tag gcr.io/$GOOGLE_CLOUD_PROJECT/helloworld



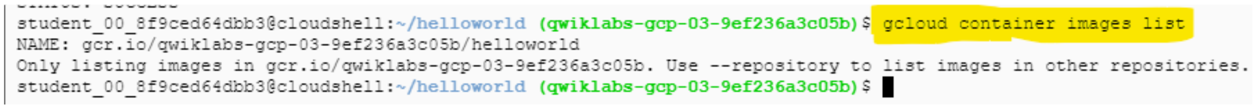
```
student_00_8f9ced64dbb3@cloudshell:~/helloworld (qwiklabs-gcp-03-9ef236a3c05b)$ gcloud builds submit --tag gcr.io/qwiklabs-gcp-03-9ef236a3c05b/helloworld
Creating temporary tarball archive of 3 file(s) totalling 1.3 KiB before compression.
Uploading tarball of [.] to [gs://qwiklabs-gcp-03-9ef236a3c05b_cloudbuild/source/1676973208.035019-afd3c3a6f052463b8148554fe3870233.tgz]
Created [https://cloudbuild.googleapis.com/v1/projects/qwiklabs-gcp-03-9ef236a3c05b/locations/global/builds/1b72962e-7d59-4068-9202-ea222385e25a].
Logs are available at [ https://console.cloud.google.com/cloud-build/builds/1b72962e-7d59-4068-9202-ea222385e25a?project=162666139695 ].
------------------------------------------------------------------------ REMOTE BUILD OUTPUT ---------------------------------------------------------
starting build "1b72962e-7d59-4068-9202-ea222385e25a"

FETCHSOURCE
Fetching storage object: gs://qwiklabs-gcp-03-9ef236a3c05b_cloudbuild/source/1676973208.035019-afd3c3a6f052463b8148554fe3870233.tgz#1676973210596248
Copying gs://qwiklabs-gcp-03-9ef236a3c05b_cloudbuild/source/1676973208.035019-afd3c3a6f052463b8148554fe3870233.tgz#1676973210596248...
/ [1 files][   1006 B/   1006 B]
Operation completed over 1 objects/1006.0 B.
BUILD
Already have image (with digest): gcr.io/cloud-builders/docker
Sending build context to Docker daemon  4.608kB
Step 1/6 : FROM node:12-slim
```

Cloud Build is a service that executes your builds on GCP. It executes a series of build steps, where each build step is run in a Docker container to produce your application container (or other artifacts) and push it to Cloud Registry, all in one command.
Once pushed to the registry, you will see a SUCCESS message containing the image name (`gcr.io/[PROJECT-ID]/helloworld`). The image is stored in Artifact Registry and can be re-used if desired.

4. List all the container images associated with your current project using this command:

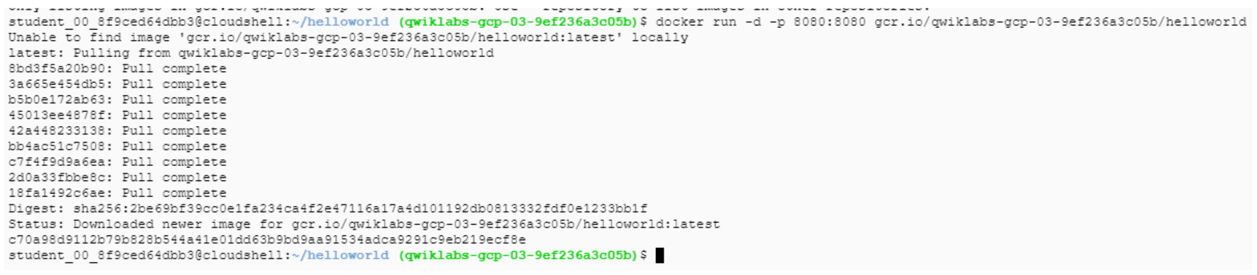
`gcloud container images list`



```
student_00_8f9ced64dbb3@cloudshell:~/helloworld (qwiklabs-gcp-03-9ef236a3c05b)$ gcloud container images list
NAME: gcr.io/qwiklabs-gcp-03-9ef236a3c05b/helloworld
Only listing images in gcr.io/qwiklabs-gcp-03-9ef236a3c05b. Use --repository to list images in other repositories.
student_00_8f9ced64dbb3@cloudshell:~/helloworld (qwiklabs-gcp-03-9ef236a3c05b)$
```

5. To run and test the application locally from Cloud Shell, start it using this standard `docker` command:

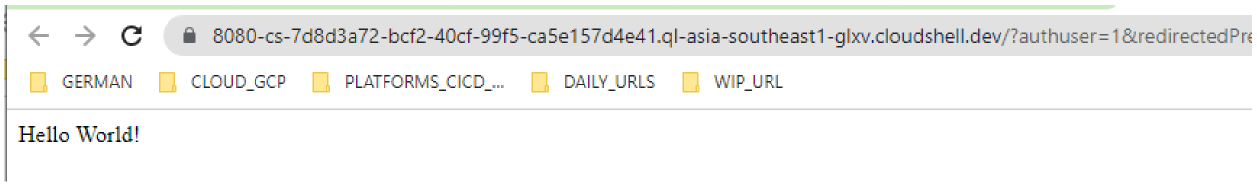
```
docker run -d -p 8080:8080 gcr.io/$GOOGLE_CLOUD_PROJECT/helloworld
```



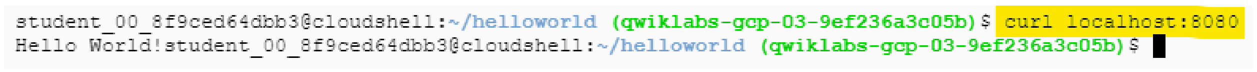
```
student_00_8f9ced64dbb3@cloudshell:~/helloworld (qwiklabs-gcp-03-9ef236a3c05b)$ docker run -d -p 8080:8080 gcr.io/qwiklabs-gcp-03-9ef236a3c05b/helloworld
Unable to find image 'gcr.io/qwiklabs-gcp-03-9ef236a3c05b/helloworld:latest' locally
latest: Pulling from qwiklabs-gcp-03-9ef236a3c05b/helloworld
8bd3f5a20b90: Pull complete
3a665e454db5: Pull complete
b5b0e172ab63: Pull complete
45013ee4878f: Pull complete
42a448233138: Pull complete
bb4ac51c7508: Pull complete
c7f4f9d9a6ea: Pull complete
2d0a33fbbe8c: Pull complete
18fa1492c6ae: Pull complete
Digest: sha256:2be69bf39cc0e1fa234ca4f2e47116a17a4d101192db0813332fdf0e1233bb1f
Status: Downloaded newer image for gcr.io/qwiklabs-gcp-03-9ef236a3c05b/helloworld:latest
c70a98d9112b79b828b544a41e01dd63b9bd9aa91534adca9291c9eb219ecf8e
student_00_8f9ced64dbb3@cloudshell:~/helloworld (qwiklabs-gcp-03-9ef236a3c05b)$
```

6. In the Cloud Shell window, click on **Web preview** and select **Preview on port 8080**.

This should open a browser window showing the "Hello World!" message. You could also simply use `curl localhost:8080`.


8080-cs-7d8d3a72-bcf2-40cf-99f5-ca5e157d4e41.ql-asia-southeast1-glxv.cloudshell.dev/?authuser=1&redirectedPre
GERMAN
CLOUD_GCP
PLATFORMS_CICD_...
DAILY_URLS
WIP_URL

Hello World!

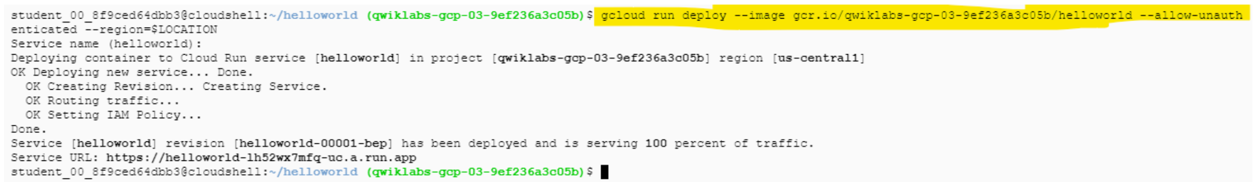


```
student_00_8f9ced64dbb3@cloudshell:~/helloworld (qwiklabs-gcp-03-9ef236a3c05b)$ curl localhost:8080
Hello World!student_00_8f9ced64dbb3@cloudshell:~/helloworld (qwiklabs-gcp-03-9ef236a3c05b)$
```

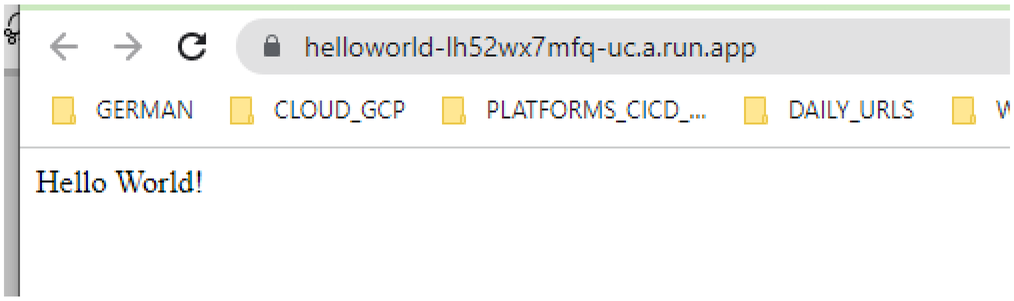
Note: If the `docker` command cannot pull the remote container image then try running this: `gcloud auth configure-docker`

1. Deploying your containerized application to Cloud Run is done using the following command adding your Project-ID:

```
gcloud run deploy --image gcr.io/$GOOGLE_CLOUD_PROJECT/helloworld
--allow-unauthenticated --region=$LOCATION
```



```
student_00_8f9ced64dbb3@cloudshell:~/helloworld (qwiklabs-gcp-03-9ef236a3c05b)$ gcloud run deploy --image gcr.io/qwiklabs-gcp-03-9ef236a3c05b/helloworld --allow-unauth
enticated --region=$LOCATION
Service name (helloworld):
Deploying container to Cloud Run service [helloworld] in project [qwiklabs-gcp-03-9ef236a3c05b] region [us-central1]
OK Deploying new service... Done.
  OK Creating Revision... Creating Service.
  OK Routing traffic...
  OK Setting IAM Policy...
Done.
Service [helloworld] revision [helloworld-00001-bep] has been deployed and is serving 100 percent of traffic.
Service URL: https://helloworld-lh52wx7mfq-uc.a.run.app
student_00_8f9ced64dbb3@cloudshell:~/helloworld (qwiklabs-gcp-03-9ef236a3c05b)$
```



Hello World!

The allow-unauthenticated flag in the command above makes your service publicly accessible.

You can now visit your deployed container by opening the service URL in any browser window.

Congratulations! You have just deployed an application packaged in a container image to Cloud Run. Cloud Run automatically and horizontally scales your container image to handle the received requests, then scales down when demand decreases. In your own environment, you only pay for the CPU, memory, and networking consumed during request handling.

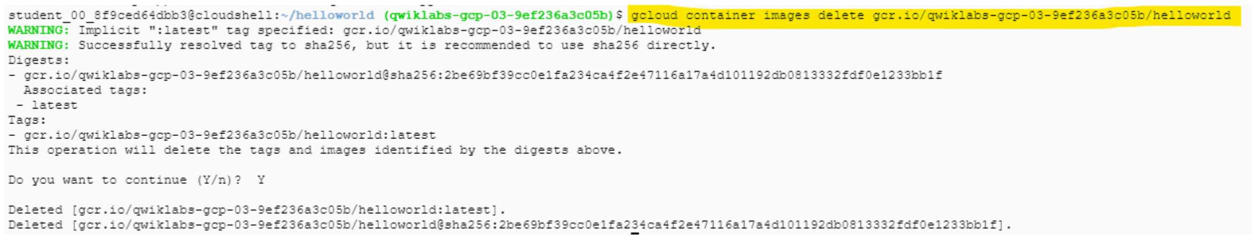
For this lab you used the `gcloud` command-line. Cloud Run is also available via Cloud Console.

- From the **Navigation menu**, in the Serverless section, click **Cloud Run** and you should see your `helloworld` service listed:

While Cloud Run does not charge when the service is not in use, you might still be charged for storing the built container image.

1. You can either decide to delete your GCP project to avoid incurring charges, which will stop billing for all the resources used within that project, or simply delete your `helloworld` image using this command :

gcloud container images delete gcr.io/$GOOGLE_CLOUD_PROJECT/helloworld



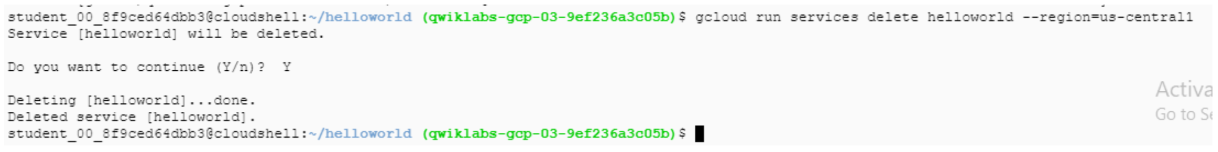
```
student_00_8f9ced64dbb3@cloudshell:~/helloworld (qwiklabs-gcp-03-9ef236a3c05b)$ gcloud container images delete gcr.io/qwiklabs-gcp-03-9ef236a3c05b/helloworld
WARNING: Implicit ":latest" tag specified: gcr.io/qwiklabs-gcp-03-9ef236a3c05b/helloworld
WARNING: Successfully resolved tag to sha256, but it is recommended to use sha256 directly.
Digests:
- gcr.io/qwiklabs-gcp-03-9ef236a3c05b/helloworld@sha256:2be69bf39cc0e1fa234ca4f2e47116a17a4d101192db0813332fdf0e1233bb1f
  Associated tags:
 - latest
Tags:
- gcr.io/qwiklabs-gcp-03-9ef236a3c05b/helloworld:latest
This operation will delete the tags and images identified by the digests above.

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

Deleted [gcr.io/qwiklabs-gcp-03-9ef236a3c05b/helloworld:latest].
Deleted [gcr.io/qwiklabs-gcp-03-9ef236a3c05b/helloworld@sha256:2be69bf39cc0e1fa234ca4f2e47116a17a4d101192db0813332fdf0e1233bb1f].
```

2. To delete the Cloud Run service, use this command :

gcloud run services delete helloworld --region=us-central1



```
student_00_8f9ced64dbb3@cloudshell:~/helloworld (qwiklabs-gcp-03-9ef236a3c05b)$ gcloud run services delete helloworld --region=us-central1
Service [helloworld] will be deleted.

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

Deleting [helloworld]...done.
Deleted service [helloworld].
student_00_8f9ced64dbb3@cloudshell:~/helloworld (qwiklabs-gcp-03-9ef236a3c05b)$
```