Deploy Node.js Express Application in App Engine

[favorite\_border](https://www.qwiklabs.com/favorite.json?item_id=3340&item_type=CatalogItem)

Add to favorites

help

—/100

Checkpoints

arrow\_forward

Deploying the Application into App Engine

Check my progress

/ 50

Updating the Application

Check my progress

/ 50

Start Lab

00:55:00

[GSP028](https://www.qwiklabs.com/focuses/3340?parent=catalog#step1)[Overview](https://www.qwiklabs.com/focuses/3340?parent=catalog#step2)[Setup and Requirements](https://www.qwiklabs.com/focuses/3340?parent=catalog#step3)[Enable the App Engine Admin API](https://www.qwiklabs.com/focuses/3340?parent=catalog#step4)[Get the Getting Started Example source code](https://www.qwiklabs.com/focuses/3340?parent=catalog#step5)[Run the Application Locally](https://www.qwiklabs.com/focuses/3340?parent=catalog#step6)[Deploying the Application into App Engine](https://www.qwiklabs.com/focuses/3340?parent=catalog#step7)[Updating the Application](https://www.qwiklabs.com/focuses/3340?parent=catalog#step8)[Congratulations!](https://www.qwiklabs.com/focuses/3340?parent=catalog#step9)

Deploy Node.js Express Application in App Engine

55 minutes5 Credits

Rate Lab

**GSP028**



**Overview**

Google App Engine applications are easy to create, easy to maintain, and easy to scale as your traffic and data storage needs change. With App Engine, there are no servers to maintain. You simply upload your application and it's ready to go.

App Engine applications automatically scale based on incoming traffic. load balancing, microservices, authorization, SQL and NoSQL databases, Memcache, traffic splitting, logging, search, versioning, roll out and roll backs, and security scanning are all supported natively and are highly customizable.

App Engine's environments, the [Standard Environment](https://cloud.google.com/appengine/docs/about-the-standard-environment) and the [Flexible environment](https://cloud.google.com/appengine/docs/flexible/) , support a host of programming languages, including Java, Python, PHP, Node.js, Go, etc.. The two environments give users maximum flexibility in how their application behaves since each environment has certain strengths. Read [The App Engine Environments](https://cloud.google.com/appengine/docs/the-appengine-environments) for more information.

This tutorial uses the sample code from the [Google Cloud Node.js Getting Started guide](https://github.com/GoogleCloudPlatform/nodejs-docs-samples)

* You will learn how to to connect to computing resources hosted on Google Cloud via the web. You will learn how to use Cloud Shell and the Cloud SDK gcloud command.

**What you'll learn**

* How to create a Node.js Express application on Google App Engine.
* How to update the code without taking the server down.

**Setup and Requirements**

**Qwiklabs setup**

**Before you click the Start Lab button**

Read these instructions. Labs are timed and you cannot pause them. The timer, which starts when you click **Start Lab**, shows how long Google Cloud resources will be made available to you.

This Qwiklabs hands-on lab lets you do the lab activities yourself in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials that you use to sign in and access Google Cloud for the duration of the lab.

**What you need**

To complete this lab, you need:

* Access to a standard internet browser (Chrome browser recommended).
* Time to complete the lab.

**Note:** If you already have your own personal Google Cloud account or project, do not use it for this lab.

**Note:** If you are using a Pixelbook, open an Incognito window to run this lab.

**How to start your lab and sign in to the Google Cloud Console**

1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is a panel populated with the temporary credentials that you must use for this lab.



1. Copy the username, and then click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Sign in** page.



***Tip:*** Open the tabs in separate windows, side-by-side.

If you see the **Choose an account** page, click **Use Another Account**. 

1. In the **Sign in** page, paste the username that you copied from the Connection Details panel. Then copy and paste the password.

***Important:*** You must use the credentials from the Connection Details panel. Do not use your Qwiklabs credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).

1. Click through the subsequent pages:
   * Accept the terms and conditions.
   * Do not add recovery options or two-factor authentication (because this is a temporary account).
   * Do not sign up for free trials.

After a few moments, the Cloud Console opens in this tab.

**Note:** You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-left. 

Activate Cloud Shell

Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources.

In the Cloud Console, in the top right toolbar, click the **Activate Cloud Shell** button.



Click **Continue**.



It takes a few moments to provision and connect to the environment. When you are connected, you are already authenticated, and the project is set to your *PROJECT\_ID*. For example:



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

(Output)

Credentialed accounts:

- <myaccount>@<mydomain>.com (active)

(Example output)

Credentialed accounts:

- google1623327\_student@qwiklabs.net

You can list the project ID with this command:

gcloud config list project

(Output)

[core]

project = <project\_ID>

(Example output)

[core]

project = qwiklabs-gcp-44776a13dea667a6

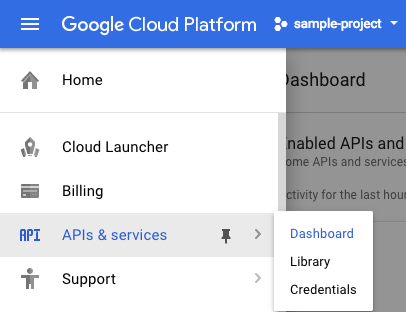
For full documentation of gcloud see the [gcloud command-line tool overview](https://cloud.google.com/sdk/gcloud" \t "_blank).

**Enable the App Engine Admin API**

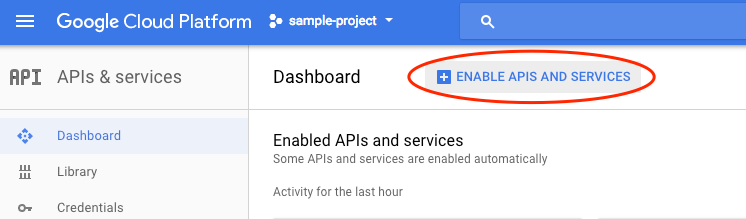
Click on the **Navigation menu** icon in the top left of the screen:

burger_menu.png

Select **APIs & services** from the drop down and click on **Dashboard**:



Click on **Enable APIs and services**:



Then, search for App Engine in the search box.

Click on **App Engine Admin API**, then click **Enable** if the API isn't enabled already.

**Get the Getting Started Example source code**

After Cloud Shell launches, you can use the command line to clone the example source code in the home directory:

git clone https://github.com/GoogleCloudPlatform/nodejs-docs-samples.git && cd nodejs-docs-samples/appengine/hello-world/flexible

Now, install the dependencies with npm:

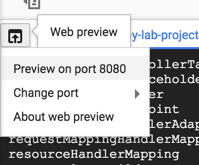
npm install

**Run the Application Locally**

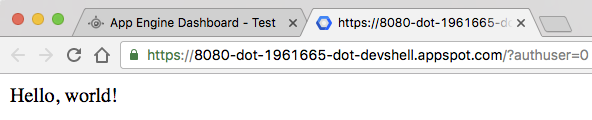
You can start the Node.js application normally with npm:

npm start

Once the application started, click on the Web Preview icon web_preview_icon.pngin the Cloud Shell toolbar and choose **preview on port 8080**.



A tab in your browser opens and connects to the server you just started.



In the cloud shell command line, press **Ctrl**+**C** to quit.

**Deploying the Application into App Engine**

App Engine uses a file called app.yaml to describe an application's deployment configuration. If this file is not present, App Engine will try to guess the deployment configuration. However, it is a good idea to provide this file.

Open app.yaml to see what it contains. You can use vim, nano,or emacs to view the file:

**app.yaml**

runtime: nodejs

env: flex

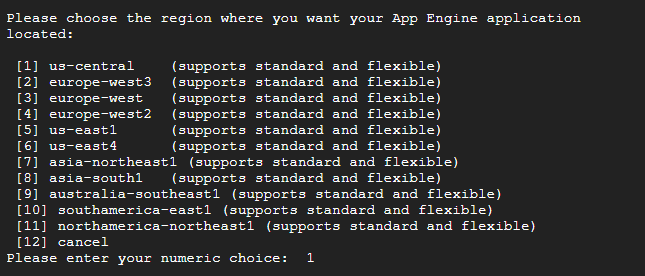
This is the basic configuration needed to deploy a Node.js application. You can learn more about configuring App Engine [here](https://cloud.google.com/appengine/docs/flexible/nodejs/configuring-your-app-with-app-yaml) .

Close the file without making any modifications.

To deploy your application into App Engine environment, run:

gcloud app deploy

Because this is the first time you are deploying App Engine, the tool will prompt you to select a location to deploy the app. For this lab choose us-central.



When prompted, enter y to confirm deployment.

First time deployment may take several minutes. This is because App Engine Flexible environment automatically provisions a Compute Engine virtual machine for you behind the scenes, and then installs and starts the application.

After the application deploys, you can visit it by opening the URL http://<project-id>.appspot.com in your web browser. The project-id is the Project ID in the CONNECTION DETAILS section of the lab.

Alternatively, you enter the following command for the URL link:gcloud app browse

In this step, you set up a simple Node.js application and ran and deployed your application on App Engine.

**Test Completed Task**

Click **Check my progress** to verify your performed task. If you have successfully deployed the application into App Engine, you will see an assessment score.

Deploying the Application into App Engine

Check my progress

**Updating the Application**

Update the application to generate a [UUID](https://en.wikipedia.org/wiki/Universally_unique_identifier) every time someone visits the page.

Install the uuid package with npm:

npm install uuid@^3.1 --save

Now modify app.js to use uuid. You can use vim, nano,or emacs to edit the file:

Enter the following in the the Cloud Shell Command Line to update the app.js file:

vi app.js

Start the editor:

i

Import the uuid module in app.js file:

const uuid = require('uuid/v4');

Update the following in app.js file:

.send(`Hello, ${uuid()}!`)

Save the app.js file by pressing: <Esc> then entering:

:wq

**app.js**

'use strict';

// [START gae\_flex\_quickstart]

const express = require('express');

const uuid = require('uuid/v4');

const app = express();

app.get('/', (req, res) => {

res

.status(200)

.send(`Hello, ${uuid()}!`)

.end();

});

// Start the server

const PORT = process.env.PORT || 8080;

app.listen(PORT, () => {

console.log(`App listening on port ${PORT}`);

console.log('Press Ctrl+C to quit.');

});

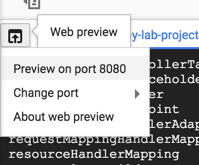
// [END gae\_flex\_quickstart]

Now, you can test the application locally.

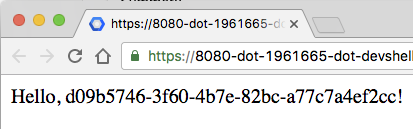
Start the Node.js application with npm:

npm start

Once the application started, click on the Web Preview icon web_preview_icon.pngin the Cloud Shell toolbar and choose **preview on port 8080**.



A tab in your browser opens and connects to the server you just started. You should see the new app that displays a UUID!



In the cloud shell command line, press **Ctrl**+**C** to quit.

To deploy your updated application, run:

gcloud app deploy

In a few minutes, the new version will be deployed and you can visit it by opening the URL http://<project-id>.appspot.com in your web browser.

Alternatively, you enter the following command for the URL link: gcloud app browse

In this step, you updated your Node.js application without any downtime.

**Test Completed Task**

Click **Check my progress** to verify your performed task. If you have successfully deployed the updated application into App Engine, you will see an assessment score.

Updating the Application

Check my progress

**Congratulations!**

You learned how to write and deploy your first App Engine web application!