

<https://codelabs.developers.google.com/codelabs/cloud-springboot-cloudshell/index.html?index=..%2F..%2Fspringone#0>

# 1. Overview

[Google Cloud Shell](#) is a browser-based command line tool to access Google Cloud Platform resources. Cloud Shell makes it really easy to manage your Cloud Platform Console projects and resources without having to install the Google Cloud SDK and other tools on your system. With Cloud Shell, the Cloud SDK `gcloud` command and tools you need to build a Java application, such as the JVM, Maven, and Gradle.

In this lab, you will learn about how to build and launch an Java web-application created with Spring Boot from Google Cloud Shell — without ever leaving the browser.

This tutorial uses the sample code from the [Spring Boot Getting Started guide](#).

## What you'll learn

- Google Cloud Shell
- How to create a simple Spring Boot Java application inside Google Cloud Shell
- How to edit the Java application using Eclipse Orion
- How to launch the Java application from Google Cloud Shell

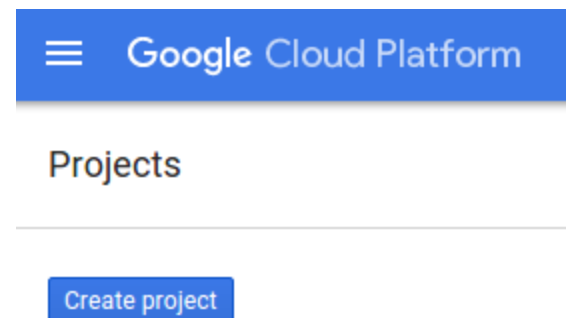
## What you'll need

- A Google Cloud Platform Project
- A Browser, such [Chrome](#) or [Firefox](#)
- Familiarity with standard Linux text editors such as Vim, EMACs or Nano

# 2. Setup and Requirements

## Self-paced environment setup

If you don't already have a Google Account (Gmail or Google Apps), you must [create one](#). Sign-in to Google Cloud Platform console ([console.cloud.google.com](https://console.cloud.google.com)) and create a new project:



## New Project

Project name ?

my-kubernetes-codelab

Your project ID will be my-kubernetes-codelab-1217 ? Edit

[Show advanced options...](#)

Create

Cancel

Remember the project ID, a unique name across all Google Cloud projects (the name above has already been taken and will not work for you, sorry!). It will be referred to later in this codelab as PROJECT\_ID.

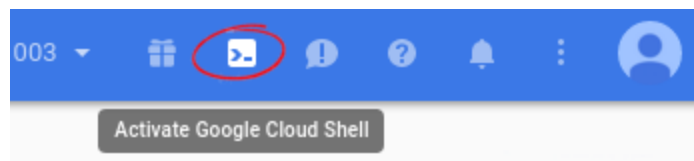
Next, you'll need to [enable billing](#) in the Developers Console in order to use Google Cloud resources. Running through this codelab shouldn't cost you more than a few dollars, but it could be more if you decide to use more resources or if you leave them running (see "cleanup" section at the end of this document).

New users of Google Cloud Platform are eligible for a [\\$300 free trial](#).

## Google Cloud Shell

While Google Cloud and Kubernetes can be operated remotely from your laptop, in this codelab we will be using [Google Cloud Shell](#), a command line environment running in the Cloud. This Debian-based virtual machine is loaded with all the development tools you'll need (docker, gcloud, kubectl and more), it offers a persistent 5GB home directory, and runs on the Google Cloud, greatly enhancing network performance and authentication. This means that all you will need for this codelab is a browser (yes, it works on a Chromebook).

To activate Google Cloud Shell, from the developer console simply click the button on the top right-hand side (it should only take a few moments to provision and connect to the environment):



## Google Cloud Shell

Free, pre-installed with the tools you need for the Google Cloud Platform. [Learn More](#)

```
Starting update of app: test-project, version: 1
10:35 PM Cloning 1 static file.
10:35 PM Cloning 5 application files.
10:35 PM Compilation starting.
10:35 PM Compilation completed.
10:35 PM Starting deployment.
10:35 PM Checking if deployment succeeded.
10:35 PM Deployment successful.
10:35 PM Checking if updated app version is
10:35 PM Completed update of app: test-project, version: 1
devstar1069@cloudshell:~/appengine-example$
```

### Real Linux environment

- Linux Debian-based OS
- 5GB persisted home directory
- Add, edit and save files

### Configured for Google Cloud

- Google Cloud SDK
- Google App Engine SDK
- Docker
- Git
- Kubernetes
- Build tools
- View more [↗](#)

### Popular language support

- Python
- Java
- Go
- Node.js

Start Cloud Shell

Cancel

codelab-test003 x +

```
Welcome to Cloud Shell! For help, visit https://cloud.google.com/cloud-shell/help.
testuser003@codelab-test003:~$ gcloud auth list
```

Once connected to the cloud shell, you should see that you are already authenticated and that the project is already set to your PROJECT\_ID :

```
$ gcloud auth list
```

```
Credentialed accounts:
```

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

Note: gcloud is the powerful and unified command-line tool for Google Cloud Platform. Full documentation is available from <https://cloud.google.com/sdk/gcloud>. It comes pre-installed on CloudShell and you will surely enjoy its support for tab-completion.

```
$ gcloud config list project
```

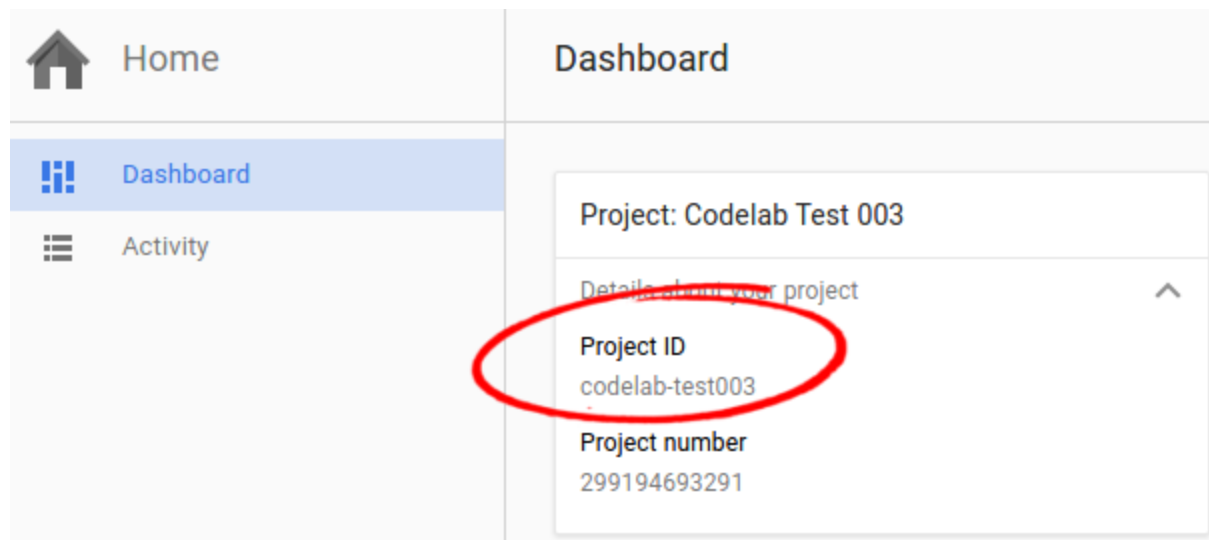
```
[core]
```

```
project = <PROJECT_ID>
```

If for some reason the project is not set, simply issue the following command :

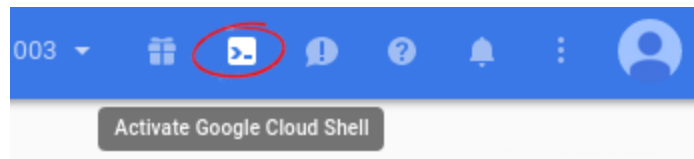
```
$ gcloud config set project <PROJECT_ID>
```

Looking for your PROJECT\_ID? Check out what ID you used in the setup steps or look it up in the console dashboard :



## Start Cloud Shell

Navigate to the Google Cloud Console from another browser tab/window, to <https://console.cloud.google.com>. Use the login credential given to you by the lab proctor. You will do all of the work from the [Google Cloud Shell, a command line environment running in the Cloud](#). This Debian-based virtual machine is loaded with all the development tools you'll need (gcloud, git and others) and offers a persistent 5GB home directory. Open the Google Cloud Shell by clicking on the icon on the top right of the screen:



## 3. Use OpenJDK 8

Google Cloud Shell has both Java 7 and Java 8 installed. It uses Java 7 by default. Let's switch to use Java 8 instead. In the Cloud Shell, use update-alternative command to change the default Java version (make sure you select the java-8-openjdk option by typing "2"):

```
$ sudo update-alternatives --config javac
```

There are 2 choices for the alternative javac (providing /usr/bin/javac).

Selection	Path	Priority	Status
-----			
* 0	/usr/lib/jvm/java-7-openjdk-amd64/bin/javac	...	
1	/usr/lib/jvm/java-7-openjdk-amd64/bin/javac	...	

```
2 /usr/lib/jvm/java-8-openjdk-amd64/bin/javac ...
Press enter to keep the current choice[*], or type selection number: 2
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/javac to provide /usr/bin/javac
(javac) in manual mode
```

```
$ sudo update-alternatives --config java
There are 2 choices for the alternative java (providing /usr/bin/java).
Selection Path Priority Status
-----
* 0 /usr/lib/jvm/java-7-openjdk-amd64/jre/bin/java ...
1 /usr/lib/jvm/java-7-openjdk-amd64/jre/bin/java ...
2 /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java ...
Press enter to keep the current choice[*], or type selection number: 2
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java to provide /usr/bin/java
(java) in manual mode
```

## 4. Install SDK Manager & Spring Boot Command Line Utility

For the lab, we'll build something from scratch. First, install the basic command line utilities that you'll need to initialize a Spring Boot Java application.

First, install SDK manager - this can help you install Spring Boot CLI with ease:

```
$ curl -s "https://get.sdkman.io" | bash
```

Then, run the command to initialize the environmental variables:

```
$ source "$HOME/.sdkman/bin/sdkman-init.sh"
```

Finally, install the Spring Boot CLI utility:

```
$ sdk install springboot
```

## 5. Initialize a new Spring Boot Java Application

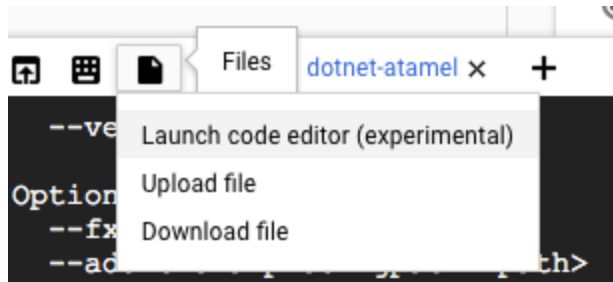
Once the Spring Boot CLI is installed, you can initialize and bootstrap a new Helloworld web application:

```
$ spring init --dependencies=web helloworld
```

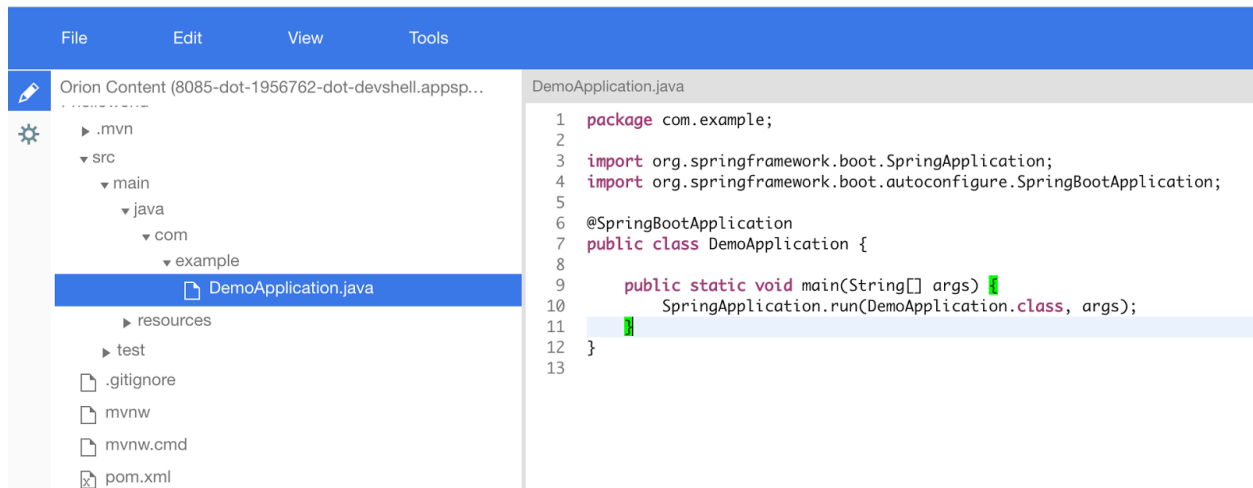
This will create a new directory with a new Maven project, along with Maven's pom.xml, a Maven wrapper, as well as an application entrypoint.

## 6. Create a new RESTful service with Eclipse Orion

Open Eclipse Orion by clicking Files and then Launch code editor from the Cloud Shell menu.



Once the code editor is open, find the file `helloworld/src/main/java/com/example/DemoApplication.java`.



Once the code is open, create a new RESTful controller to respond "Hello". In the `DemoApplication.java` file, add a new `HelloWorld` class definition in addition to the current one:

```
package com.example;
```

```
import ...;
```

```
...
```

```
import org.springframework.web.bind.annotation.*;
```

```
@SpringBootApplication
```

```
public class DemoApplication {
```

```
...
```

```
}
```

```

@RestController
class Helloworld {
    @GetMapping("/")
    public String greet() {
        return "Hello!";
    }
}

```

Don't forget to save the file!

DemoApplication.java

```

1  package com.example;
2
3  import org.springframework.boot.SpringApplication;
4  import org.springframework.boot.autoconfigure.SpringBootApplication;
5  import org.springframework.web.bind.annotation.*;
6
7  @SpringBootApplication
8  public class DemoApplication {
9
10     public static void main(String[] args) {
11         SpringApplication.run(DemoApplication.class, args);
12     }
13 }
14
15 @RestController
16 class Helloworld {
17     @GetMapping("/")
18     public String greet() {
19         return "Hello!";
20     }
21 }
22

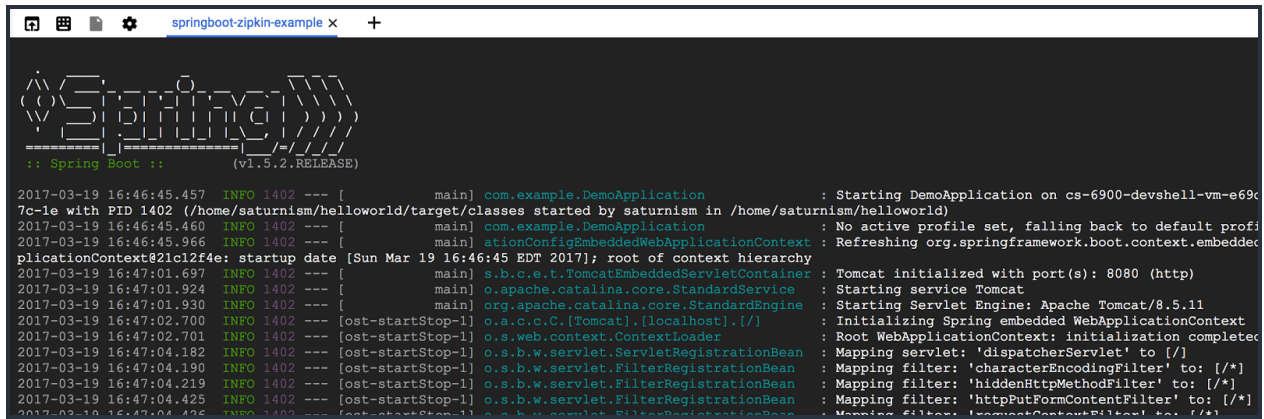
```

## 7. Run the Application Locally

You can start the Spring Boot application normally with the Spring Boot plugin:


```
$ cd $HOME/helloworld
```

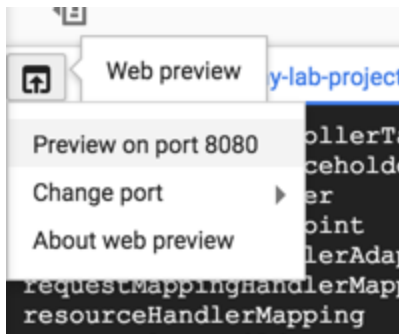
```
$ ./mvnw -DskipTests spring-boot:run
```



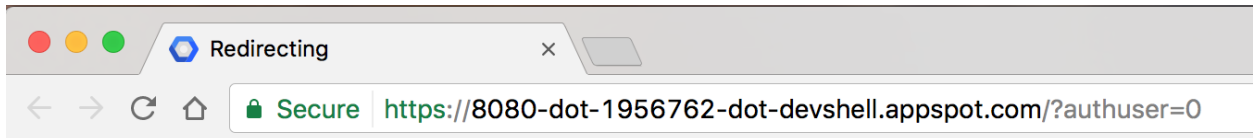
```
Spring Boot (v1.5.2.RELEASE)

2017-03-19 16:46:45.457 INFO 1402 --- [main] com.example.DemoApplication : Starting DemoApplication on cs-6900-devshell-vm-e69c
7c-1e with PID 1402 (/home/saturnism/helloworld/target/classes started by saturnism in /home/saturnism/helloworld)
2017-03-19 16:46:45.460 INFO 1402 --- [main] com.example.DemoApplication : No active profile set, falling back to default profi
2017-03-19 16:46:45.966 INFO 1402 --- [main] ationConfigEmbeddedWebApplicationContext : Refreshing org.springframework.boot.context.embedde
plicationContext821c12f4e: startup date [Sun Mar 19 16:46:45 EDT 2017]; root of context hierarchy
2017-03-19 16:47:01.697 INFO 1402 --- [main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat initialized with port(s): 8080 (http)
2017-03-19 16:47:01.924 INFO 1402 --- [main] o.apache.catalina.core.StandardService : Starting service Tomcat
2017-03-19 16:47:01.930 INFO 1402 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet Engine: Apache Tomcat/8.5.11
2017-03-19 16:47:02.700 INFO 1402 --- [ost-startStop-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2017-03-19 16:47:02.701 INFO 1402 --- [ost-startStop-1] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization complete
2017-03-19 16:47:04.182 INFO 1402 --- [ost-startStop-1] o.s.b.w.servlet.ServletRegistrationBean : Mapping servlet: 'dispatcherServlet' to [/]
2017-03-19 16:47:04.190 INFO 1402 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'characterEncodingFilter' to: [/]
2017-03-19 16:47:04.219 INFO 1402 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'hiddenHttpMethodFilter' to: [/]
2017-03-19 16:47:04.425 INFO 1402 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'httpPutFormContentFilter' to: [/]
```

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



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



Hello!