Getting started with Google App Engine and Spring Boot in 5 steps

Google Cloud Platform (GCP) provides good integration tools for developer to build and deploy application to Google App Engine. Here is a tutorial to get you setup and deploy your Spring Boot application on App Engine in just 5 steps using Maven.

Leave me any comments or questions you have and I will try to answer them. Happy Coding!!!

Spring Boot running on Google App Engine

Base knowledge & tools (what you need to have)

Google Cloud Platform account, and basic familiarity with GCP console.

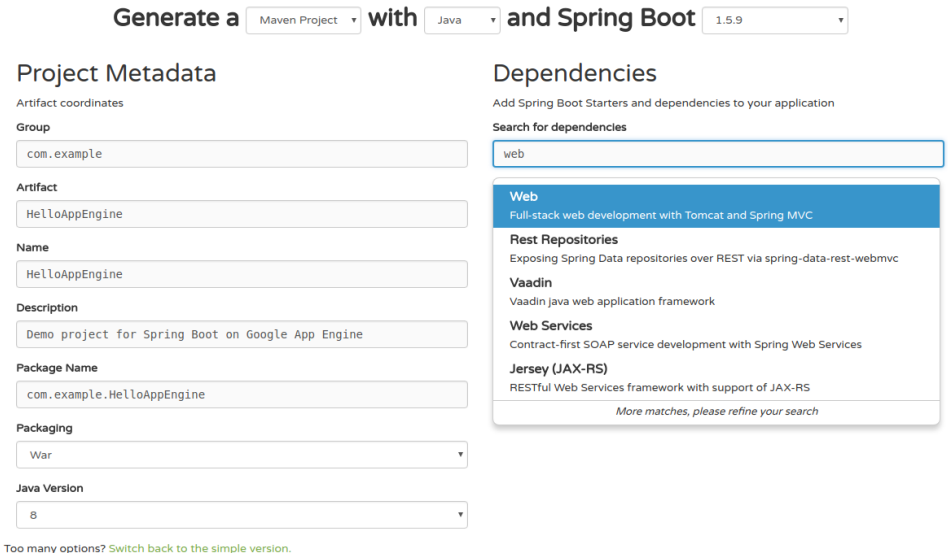
Java, Spring Boot, Maven installed on your machine.

GCloud SDK installed on your local computer with initialization.

2. Build your basic Spring Boot app

Go to https://start.spring.io/ configure your Spring Boot project and download Maven project template.

Make sure you select Java Version = 8, Packaging = war, and dependencies = Web. This tutorial uses Group = com.example, and Artifact = Name = HelloAppEngine.



You will download a zip file to your local computer, then extract it. In this tutorial you will have HelloAppEngine folder after extraction.

Change folder into HelloAppEngine, and then edit HelloAppEngineApplication.java to add a new controller to return “Hello Spring Boot!” message.

```
$vi
src/main/java/com/example/HelloAppEngine/HelloAppEngineAppl
```

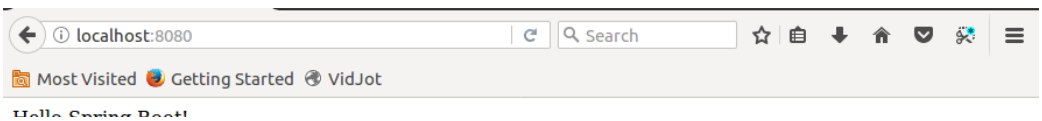
Replace HelloAppEngineApplication.java with following content.

```
package com.example.HelloAppEngine;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplicatio
import org.springframework.web.bind.annotation.*;

@SpringBootApplication
@RestController
public class HelloAppEngineApplication {

public static void main(String[] args) {

SpringApplication.run(HelloAppEngineApplication.class,
args);
        }

@GetMapping("/")
        public String hello() {
                return "Hello Spring Boot!";
        }
}
```

Run HelloAppEngine Spring Boot application. Then point your local browser to http://localhost:8080

```
cd HelloAppEngine
mvn spring-boot:run
```

You should see your first Hello Spring Boot message. You now have your Spring Boot application running locally. Quit application and move to next step.



Hello Spring Boot!

3. Make your application compatible with App Engine

This step will involve modifying Maven plugin and editing some part of pom.xml so the project is compatible with Google App Engine environment.

> Open pom.xml for editing. Remove spring-boot-starter-tomcat, and add javax.servlet-api dependency as provided. This is because Google App Engine uses Jetty while default Spring Boot uses Tomcat.

```
<dependencies>

   <!-- Remove spring-boot-starter-tomcat dependency
  <dependency>
       <groupId>org.springframework.boot</groupId>
       <artifactId>spring-boot-starter-
tomcat</artifactId>
       <scope>provided</scope>
  </dependency>
  -->

   <!-- add following dependency under dependencies
section -->
   <dependency>
       <groupId>javax.servlet</groupId>
       <artifactId>javax.servlet-api</artifactId>
       <version>3.1.0</version>
       <scope>provided</scope>
   </dependency>

   <dependency>
       <groupId>org.springframework.boot</groupId>
       <artifactId>spring-boot-starter-test</artifactId>
       <scope>test</scope>
  </dependency>

</dependencies>
```

Add Maven App Engine plugin

```
<plugins>
    <!-- add appengine-maven-plugin -->
            <plugin>

<groupId>com.google.cloud.tools</groupId>
                <artifactId>appengine-maven-
plugin</artifactId>
                <version>1.3.1</version>
            </plugin>
</plugins>
```

Add appengine_web.xml for deployment descriptor.

```
$ mkdir -p src/main/webapp/WEB-INF/
$ touch src/main/webapp/WEB-INF/appengine-web.xml
```

Add following context to appengine-web.xml.

```
<appengine-web-app
xmlns="http://appengine.google.com/ns/1.0">
  <version>1</version>
  <threadsafe>true</threadsafe>
  <runtime>java8</runtime>
</appengine-web-app>
```

Run your application with App Engine plugin. Point your browser to http://localhost:8080, then you should see default Spring Boot page just like prior step with Spring Boot plugin, but this time it is running on App Engine local server.

```
$ mvn appengine:run
```

4. Deploy to Google Cloud

This section will walk you through creating app engine on GCP and deploy your application to it. All commands are using GCloud SDK installed on your local machine and already initialized to your GCP account.

Create your project on GCP via console.cloud.google.com. This tutorial is has project id of "**spring-boot-187904**". You should refer to your project id when following this tutorial.

Set the project parameter and use GCloud sdk to create app engine app. Make sure you replace "**spring-boot-187904**" with your project id.

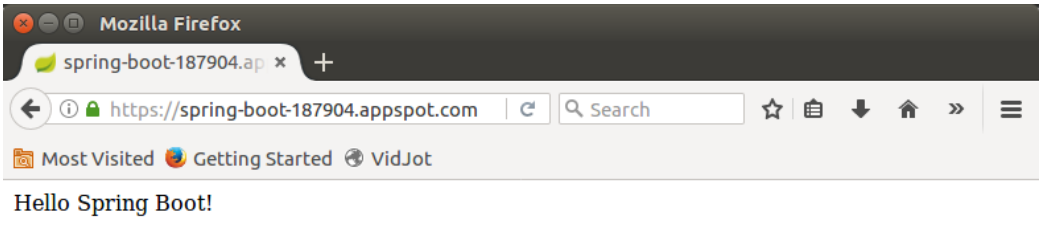
```
$ gcloud config set project spring-boot-187904

Updated property [core/project].

$ gcloud app create --region us-central
You are creating an app for project [spring-boot-
187904].
WARNING: Creating an App Engine application for a
project is irreversible and the region
cannot be changed. More information about regions is at
<https://cloud.google.com/appengine/docs/locations>.

Creating App Engine application in project [spring-boot-
187904] and region [us-central]....done.
Success! The app is now created. Please use `gcloud app
deploy` to deploy your first app.
```

Use Maven and App Engine plugin to deploy your application to Google App Engine (GAE)

```
$ mvn appengine:deploy
```

```
...
#=========================================================
[INFO] GCLOUD: File upload done.
[INFO] GCLOUD: Updating service [default]...
[INFO] GCLOUD: .............done.
[INFO] GCLOUD: Updating service [default]...
[INFO] GCLOUD: Waiting for operation [apps/spring-boot-
187904/operations/64bc4dfd-5fff-4279-bb7f-b6fe2fbb7206]
to complete...
[INFO] GCLOUD: .............done.
[INFO] GCLOUD: done.
[INFO] GCLOUD: Deployed service [default] to
[https://spring-boot-187904.appspot.com]
[INFO] GCLOUD:
[INFO] GCLOUD: You can stream logs from the command line
by running:
[INFO] GCLOUD:   $ gcloud app logs tail -s default
[INFO] GCLOUD:
[INFO] GCLOUD: To view your application in the web
browser run:
[INFO] GCLOUD:   $ gcloud app browse
[INFO] ------------------------------------------------
------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------
------------
[INFO] Total time: 44.212 s
[INFO] Finished at: 2017-12-03T21:20:31-08:00
[INFO] Final Memory: 33M/246M
[INFO] ------------------------------------------------
------------
```

Use browser to go to [YOUR_PROJECT_ID].appspot.com to access your Spring Boot application. You should see the default page just like on your local machine. Congratulations, you now have your app running on GAE.



5. Add more features to your app and update

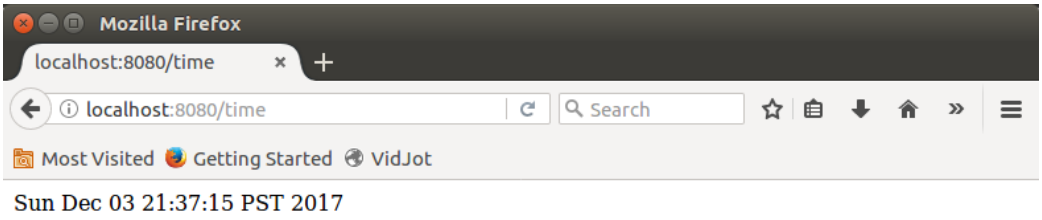
This section, we will add some features into your application and redeploy it to GAE.

Add more controller and endpoints to application. Let's modify default message and add a time service. Replace following content to HelloAppEngineApplication.java file.

```
package com.example.HelloAppEngine;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplicatio
import org.springframework.web.bind.annotation.*;

import java.util.Calendar;
import java.util.Date;

@SpringBootApplication
@RestController
public class HelloAppEngineApplication {

public static void main(String[] args) {
                SpringApplication.run(

HelloAppEngineApplication.class, args);
        }

@GetMapping("/")
        public String hello() {
                return "Hello Spring Boot, I'm on App
Engine!!!";
        }

@GetMapping("/time")
        public String time() {
                return
Calendar.getInstance().getTime().toString();
        }
}
```

Run it local App Engine plugin, and test with browser. You should see time service.

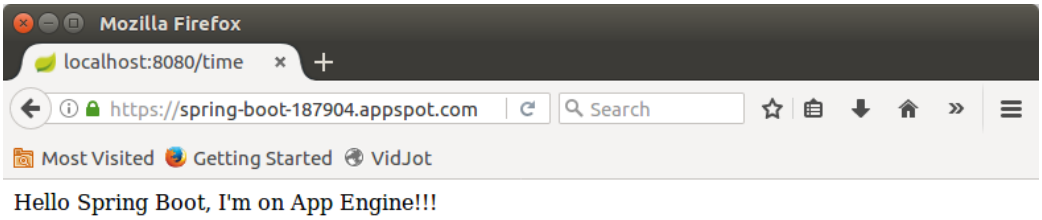
```
$ mvn appengine:run
```

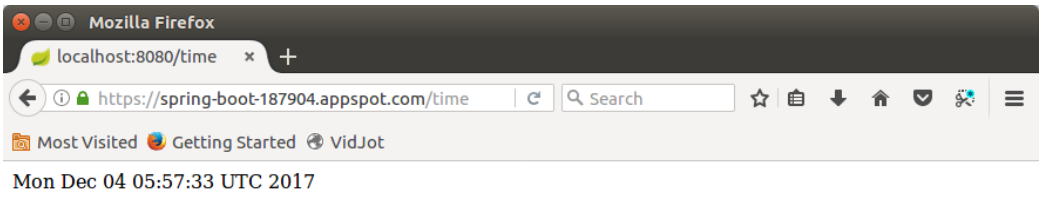


Deploy new version to App Engine, using Maven appengine plugin same as prior step. Use browser and go to [YOUR_PROJECT_ID].appspot.com then you should see a new message. Go to [YOUR_PROJECT_ID].appspot.com/time should give you current time message.

```
$ mvn appengine:deploy

...
#============================================================
[INFO] GCLOUD: File upload done.
[INFO] GCLOUD: Updating service [default]...
[INFO] GCLOUD: ........done.
[INFO] GCLOUD: Updating service [default]...
[INFO] GCLOUD: Waiting for operation [apps/spring-boot-
187904/operations/87ebf5c8-6b33-4806-8342-afa79cf5caef]
to complete...
[INFO] GCLOUD: ............done.
[INFO] GCLOUD: .done.
[INFO] GCLOUD: Deployed service [default] to
[https://spring-boot-187904.appspot.com]
[INFO] GCLOUD:
[INFO] GCLOUD: You can stream logs from the command line
by running:
[INFO] GCLOUD:   $ gcloud app logs tail -s default
[INFO] GCLOUD:
[INFO] GCLOUD: To view your application in the web
browser run:
[INFO] GCLOUD:   $ gcloud app browse
[INFO] ------------------------------------------------
```

```
------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
------------
[INFO] Total time: 01:02 min
[INFO] Finished at: 2017-12-03T21:53:41-08:00
[INFO] Final Memory: 30M/214M
[INFO] ------------------------------------------------------------------------
------------
```





Congratulations! You now have build your application, deploy, and update to Google App Engine.