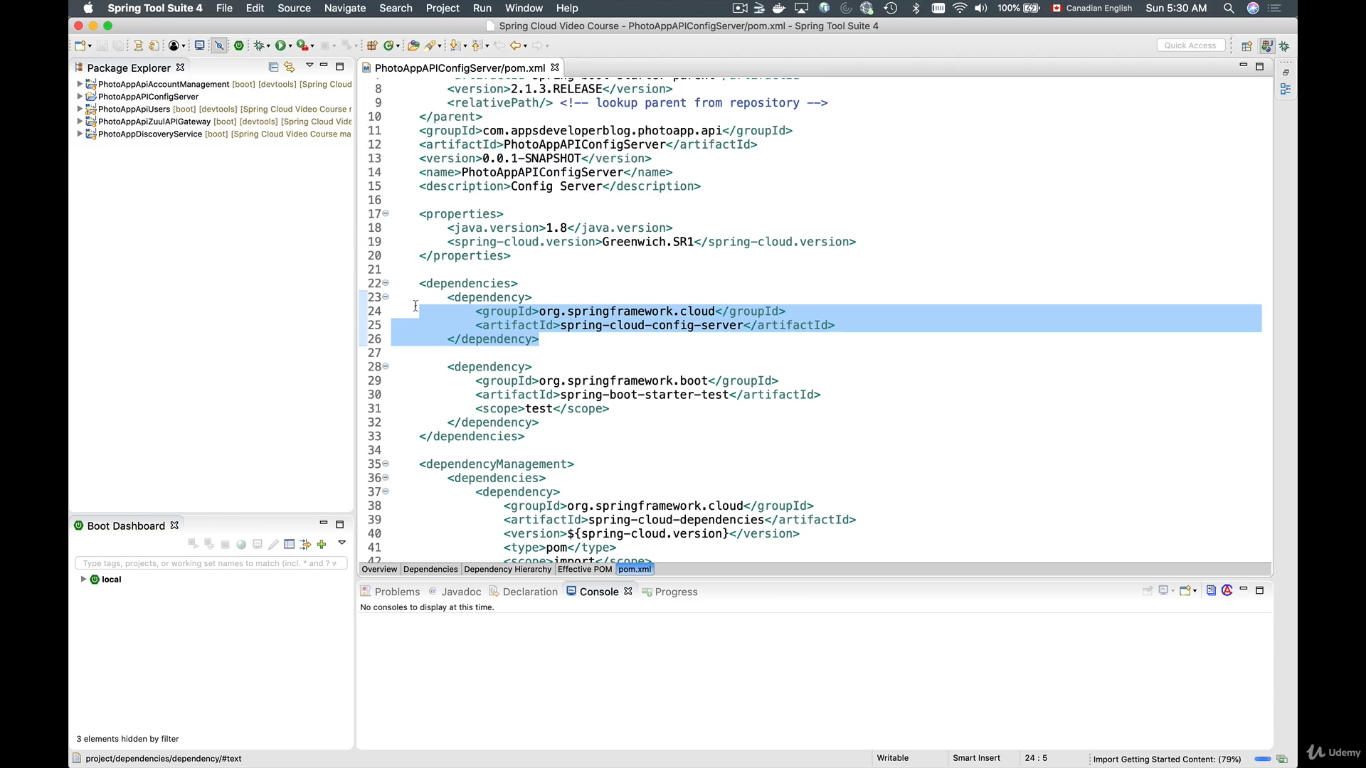
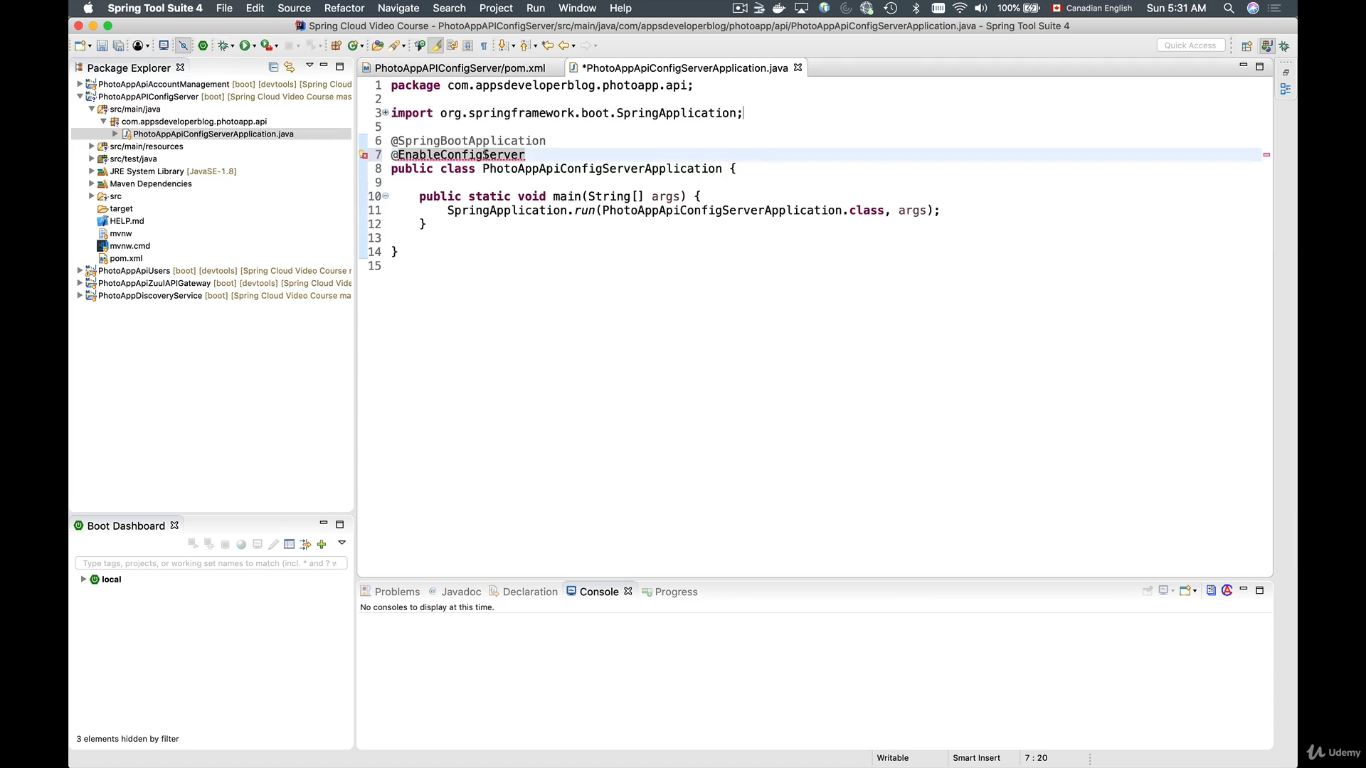
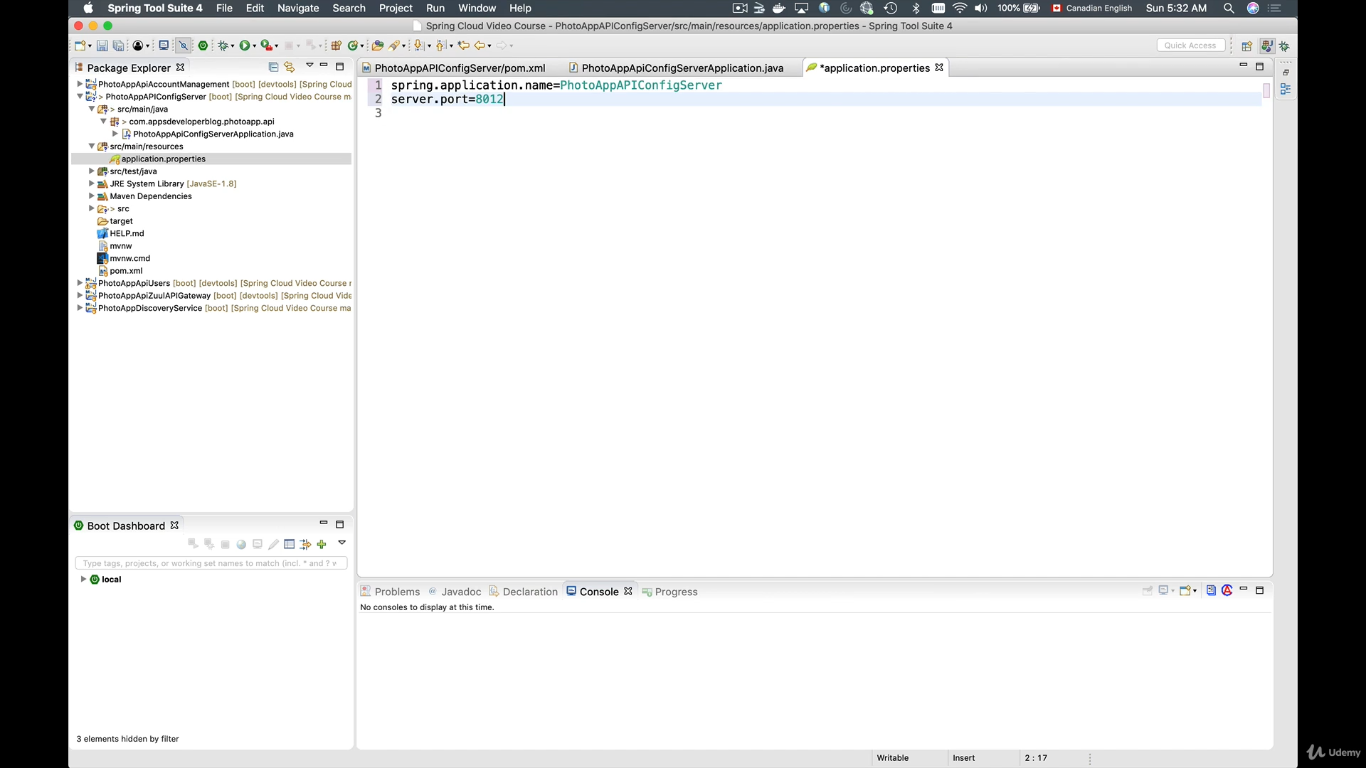


Config server will have the configuration common to all the microservices. We will set up a git repository where we will keep our common configuration file. Spring cloud is nothing but an another spring boot application.

Create a new project with spring cloud dependency.

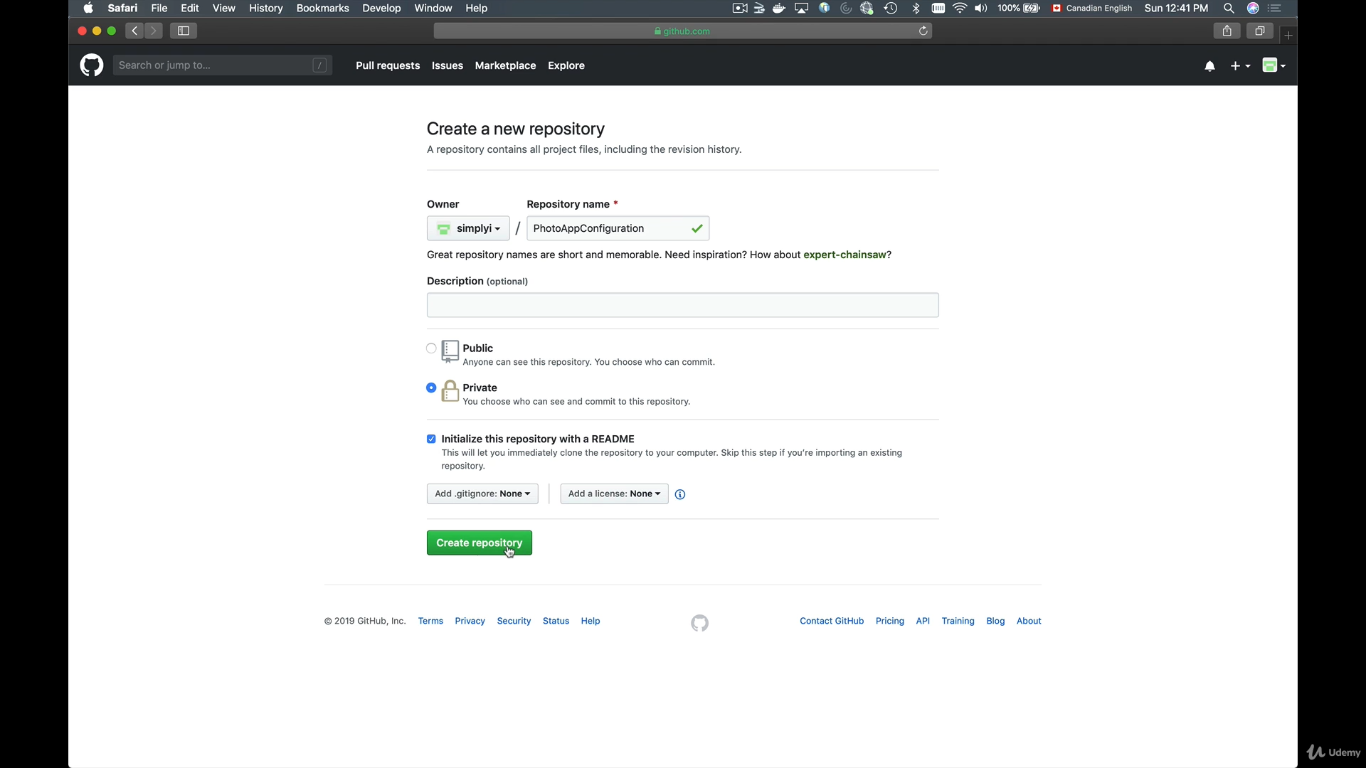


@enableconfigserver

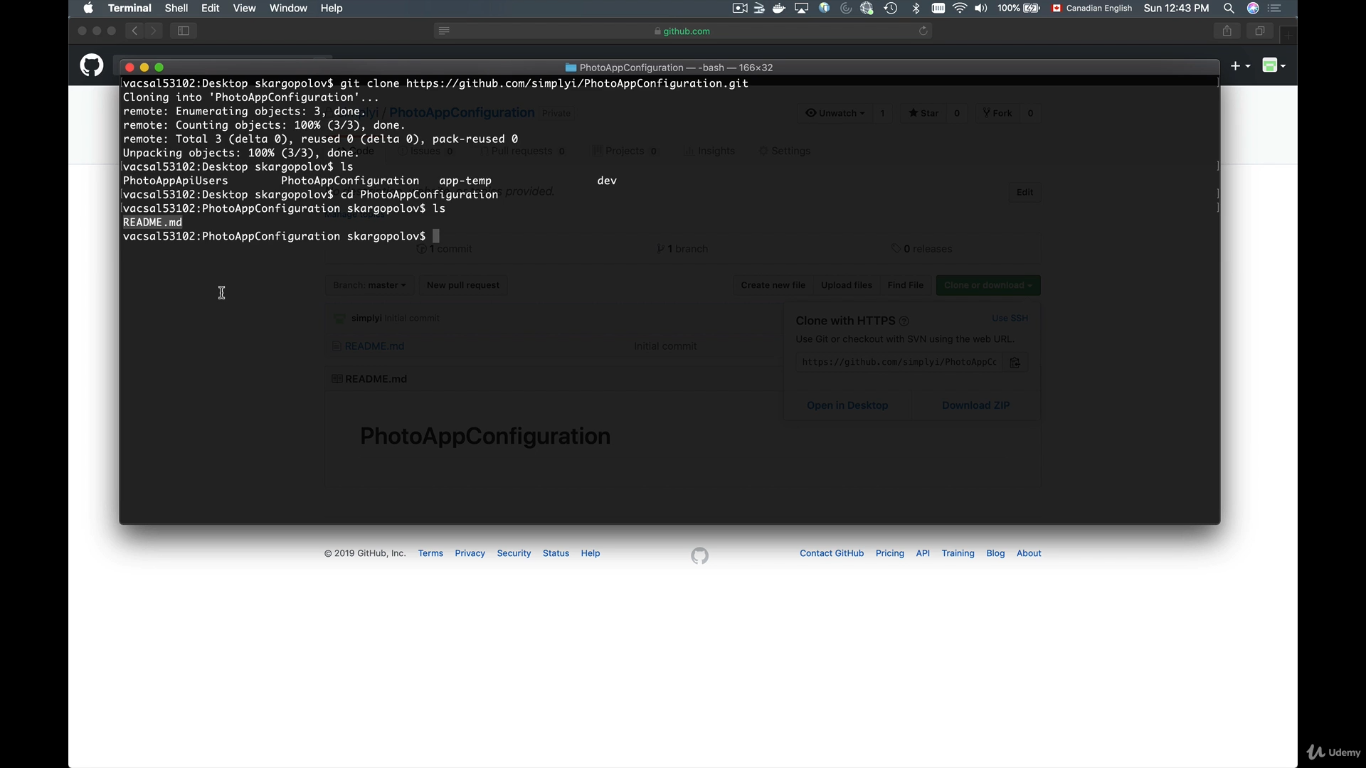


Basic spring cloud configuration.

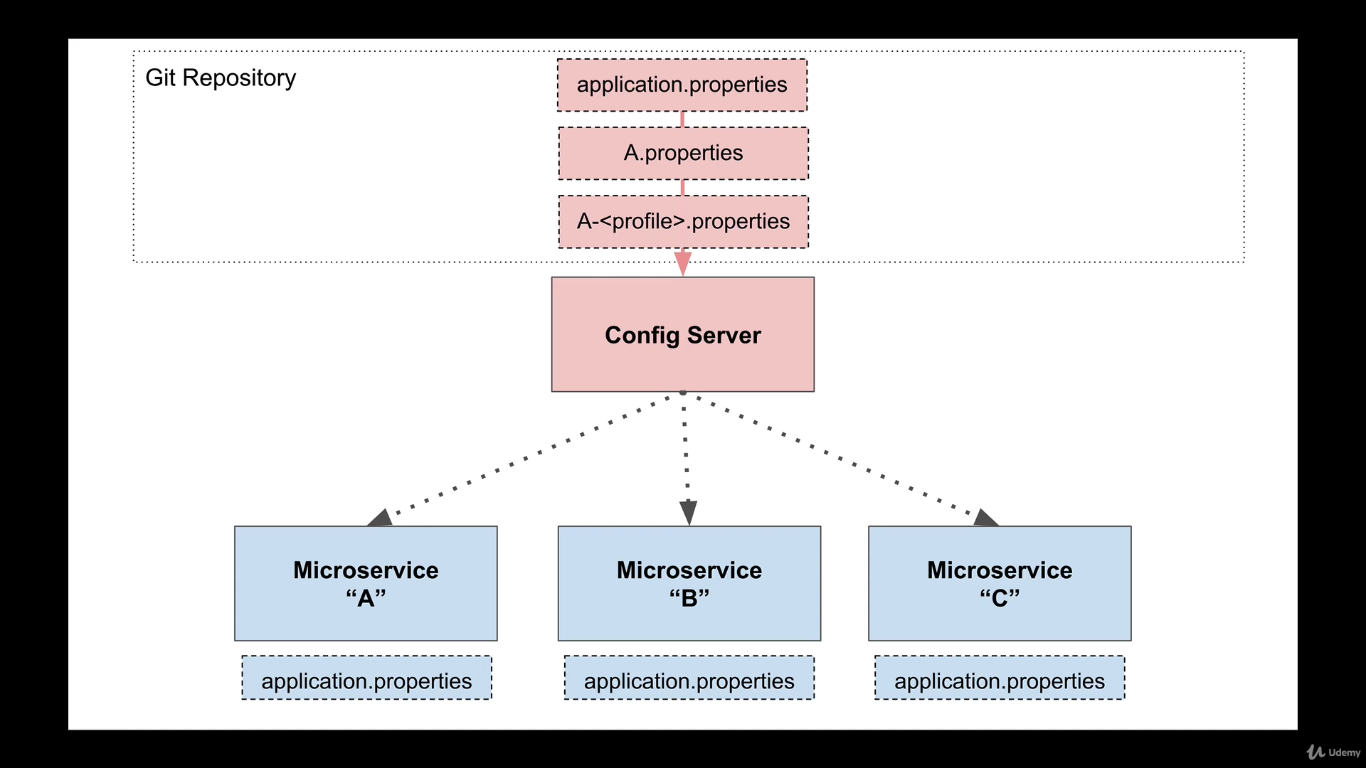
Create a private repository



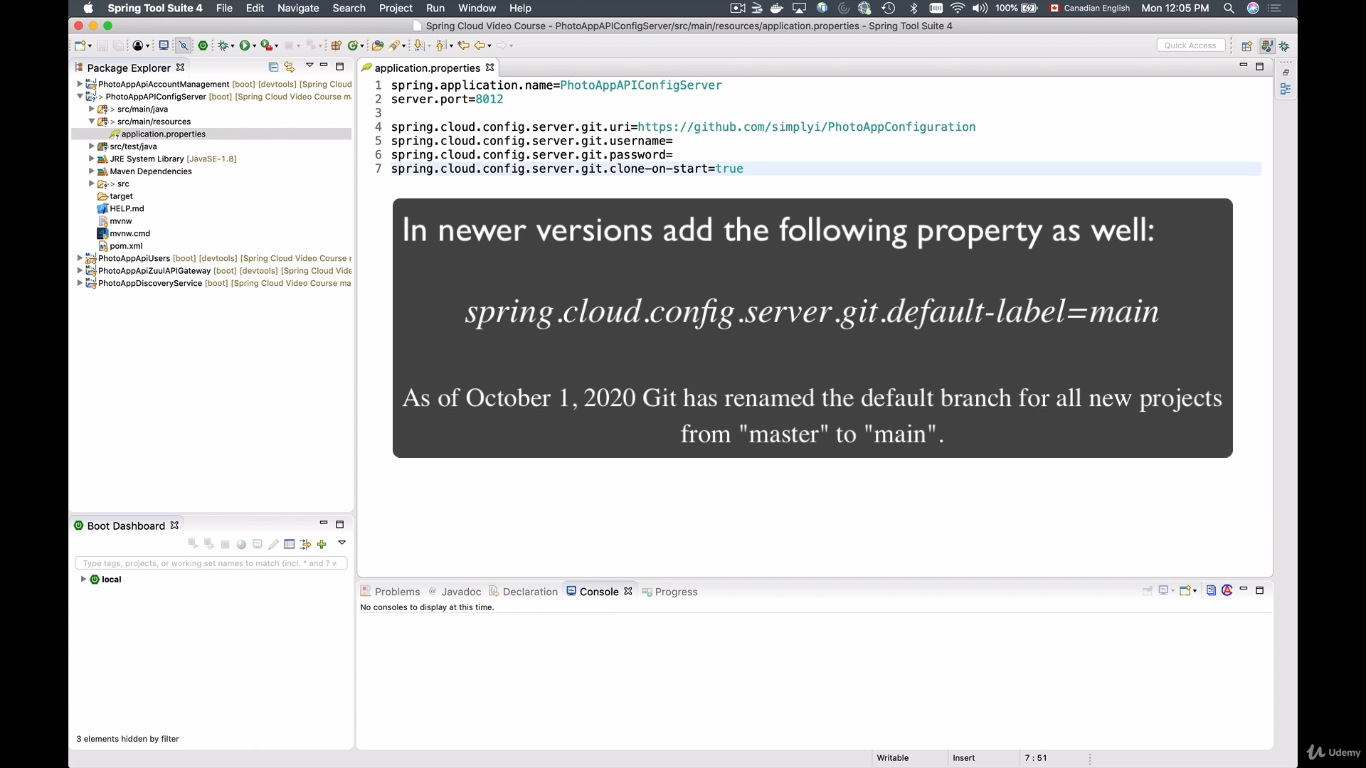
Clone the repository to your local



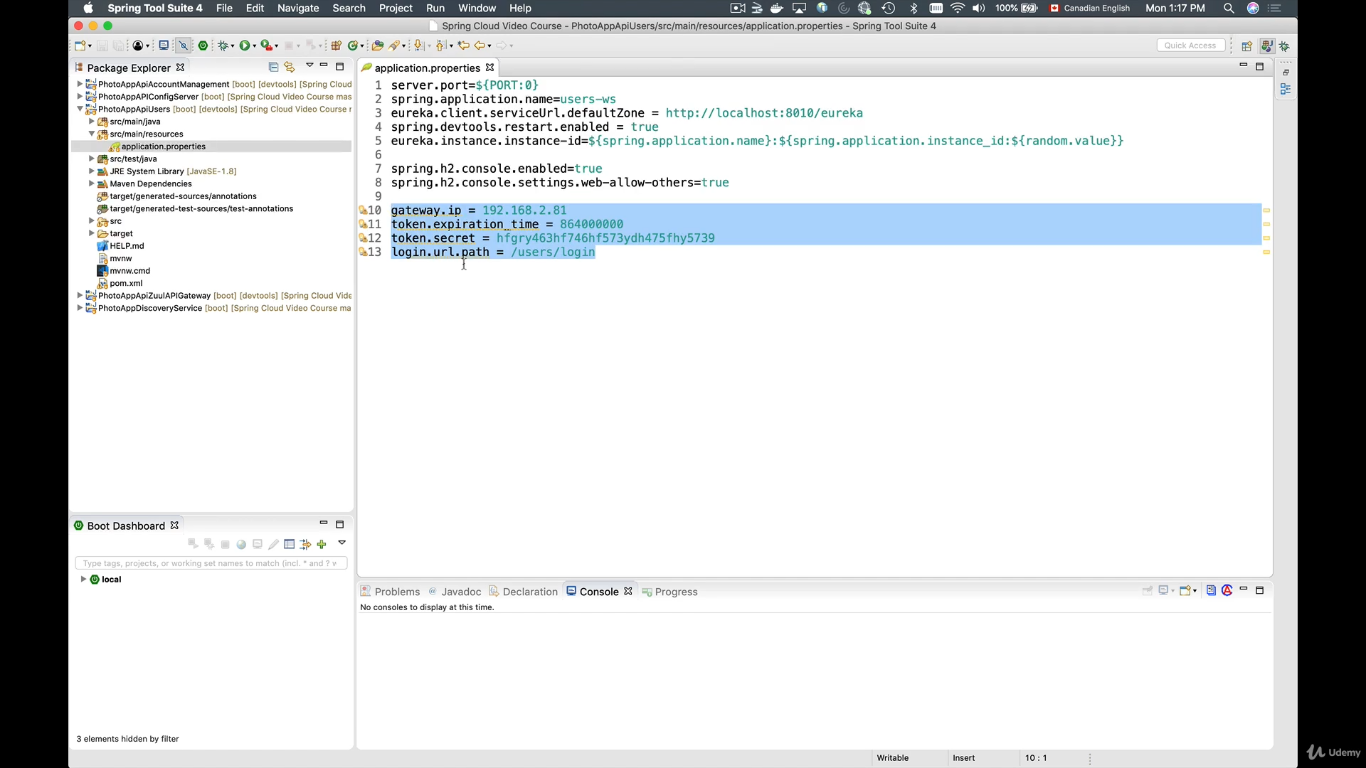
Creating a property file and pushing it to the git repository.



Application.properties file present inside the git repository will always have higher priority than the local application.properties file. If there is a properties file with specific microservice name, it will have even more priority.

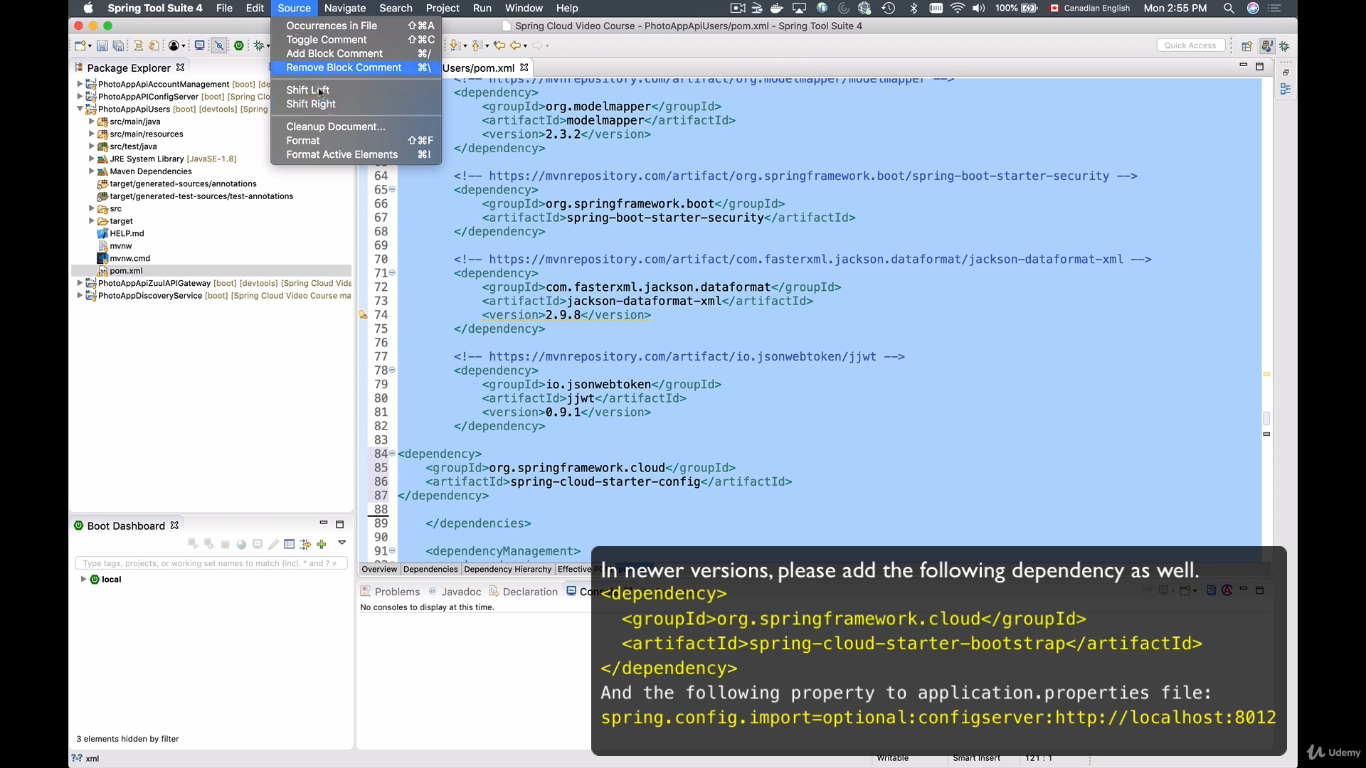


Above configurations will clone the repository on startup not on demand.

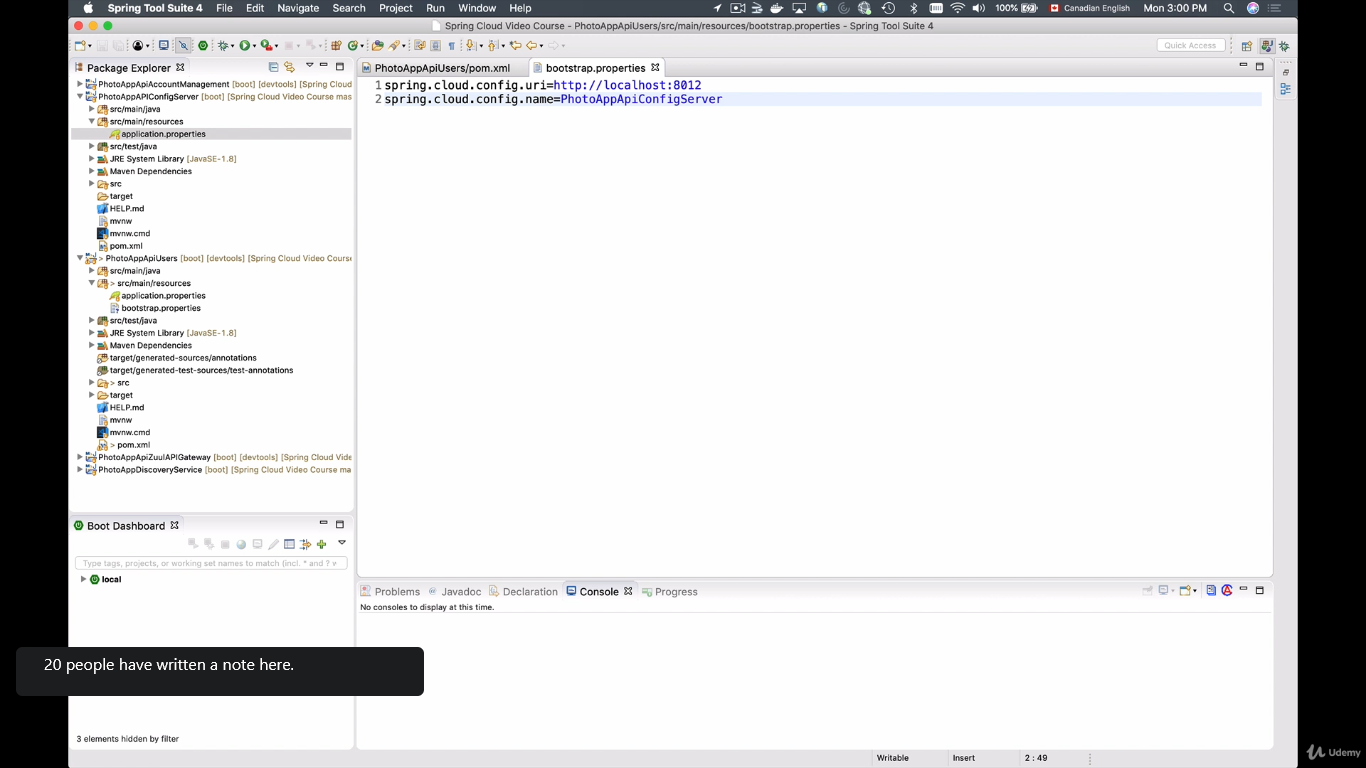


Copy the above configurations from “Users” microservice and paste into a text file and name it as application.properties. Then “add” to stages, “commit” and “push” to the centralized repository.

In order to make the “users” microservice as client of “spring config server” do the following.



1. Add the two dependencies, 1) spring cloud starter bootstrap 2) spring cloud starter config.
2. Create a new properties file called “bootstrap.properties” this file will load even before the beans are created.
3. Add the following configuration in properties file.



Now run the app and check

As same as above make the spring cloud API gateway also as a client of config server.