**DevOps scenario: Azure Web App, Azure Container Services and VSTS with Java tools and technologies.**

**Summary**

Since 1996, Java Platform has been a strong enterprise-grade technology used by many organizations across the world. Currently, there’s still an important number of enterprise customers which are using JEE-based architectures and dev tools.

Starting from the traditional Petclinic sample application and evolving to an Cloud-Powered version, this project aims to use a set of Microsoft technologies -including Azure Services and Visual Studio Team Services- to implement a DevOps integrated scenario leveraging common Java tools and technologies like Tomcat, Eclipse, Maven, Jenkins, and others.

The Petclinic sample application is designed to show how the Spring application frameworks can be used to build simple, but powerful database-oriented enterprise applications.

During this experiment we were focused on testing the app in a typical on-premise scenario. We also we’ll experiment a DevOps scenario including Continuous Integration + Continuous Deployment using common tools in Devs and Ops areas that are using Java EE.

Several Java technologies are part of this sample app:

* Development : Oracle jdk 1.8.x + Git + Eclipse + Team Explorer Everywhere (VSTS Eclipse Plugin)
* Unit Testing: Junit + Visual Studio Team Services (for reporting purposes)
* Build: Apache Maven + Visual Studio Team Services + Slack integration
* Deploy: Azure App Services (Azure Websites w/ Apache Tomcat 7.x Container)

We built a Team Project in Visual Studio Team Services and enabled DevOps practices like Continuous Integration and Continuous deployment. With Maven we created a Build that checks code and publish directly to an Azure Web App.

We also experimented alternatives to build Docker containers directly from VSTS. With ACS managed by Mesosphere, we create the application enabling scalability and portability. So far we are tuning some parts of this deployment.

**Steps to work in an On-Premise scenario, installing application**

1. Download and install Java Dev Kit (jdk)

**Java SE Dev Kit 8u77**

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

* 1. Create directory c:\dev
  2. Install jdk in c:\dev\jdk1.8.0\_77
  3. Add c:\dev\jdk1.8.0\_77\bin to an environment variable "path" (must be in first position)
  4. Add environment variable named CLASSPATH. In value write a (.) dot.
  5. Add environment variable JAVA\_HOME. In value write c:\dev\jdk1.8.0\_77

1. **Eclipse IDE For Java Developers vMars2**

<http://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/mars2>

* 1. Copy folder "eclipse" to c:\dev resulting c:\dev\eclipse

1. **Apache Maven 3.3.9**

<http://www-eu.apache.org/dist/maven/maven-3/3.3.9/binaries/apache-maven-3.3.9-bin.zip>

* 1. Copy folder apache-maven-3.3.9 in c:\dev
  2. Add C:\dev\apache-maven-3.3.9\bin to the environment variable "path" (in second position)

1. **Apache Tomcat 7.0.69 -** <http://www-us.apache.org/dist/tomcat/tomcat-7/v7.0.69/bin/apache-tomcat-7.0.69.zip>
   1. Extract in C:\dev
2. **Spring PetClinic**

<https://github.com/spring-projects/spring-petclinic>

<https://github.com/spring-projects/spring-petclinic/archive/master.zip>

* 1. Create directory "workspace" in C:\dev and extract Petclinic there
  2. Read readme.rd file to more instructions

1. **Compile the project using Maven from Eclipse**
   1. In Eclipse; File -> Import -> Maven -> Existing Maven project
   2. Select workspace\spring-petclinic-master
   3. In the package explorer, position cursor over the root of project "spring-petclinic-master"
   4. Right click > run as...> 4 Maven Build > run
2. **Deploy app to Tomcat**
   1. Look for file petclinic.war in spring-petclinic-master\target and copy
   2. Paste it in C:\dev\apache-tomcat-7.0.69\webapps
3. Execute "startup.bat" from directory C:\dev\apache-tomcat-7.0.69\bin to start Tomcat Server

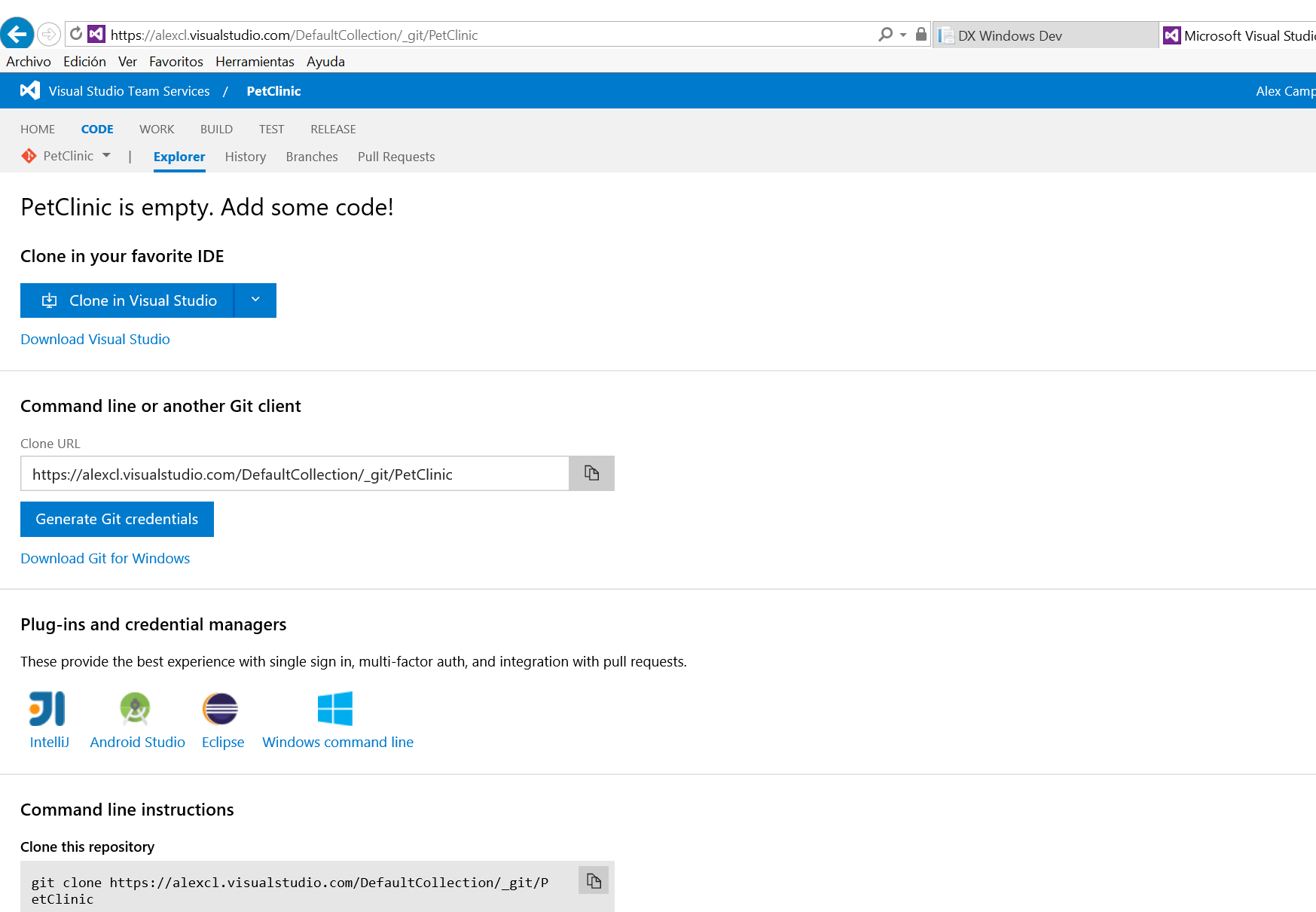
Note: to stop and shutdown the server, execute shutdown.bat

Navigate to: <http://localhost:8080>

<http://localhost/petclinic>

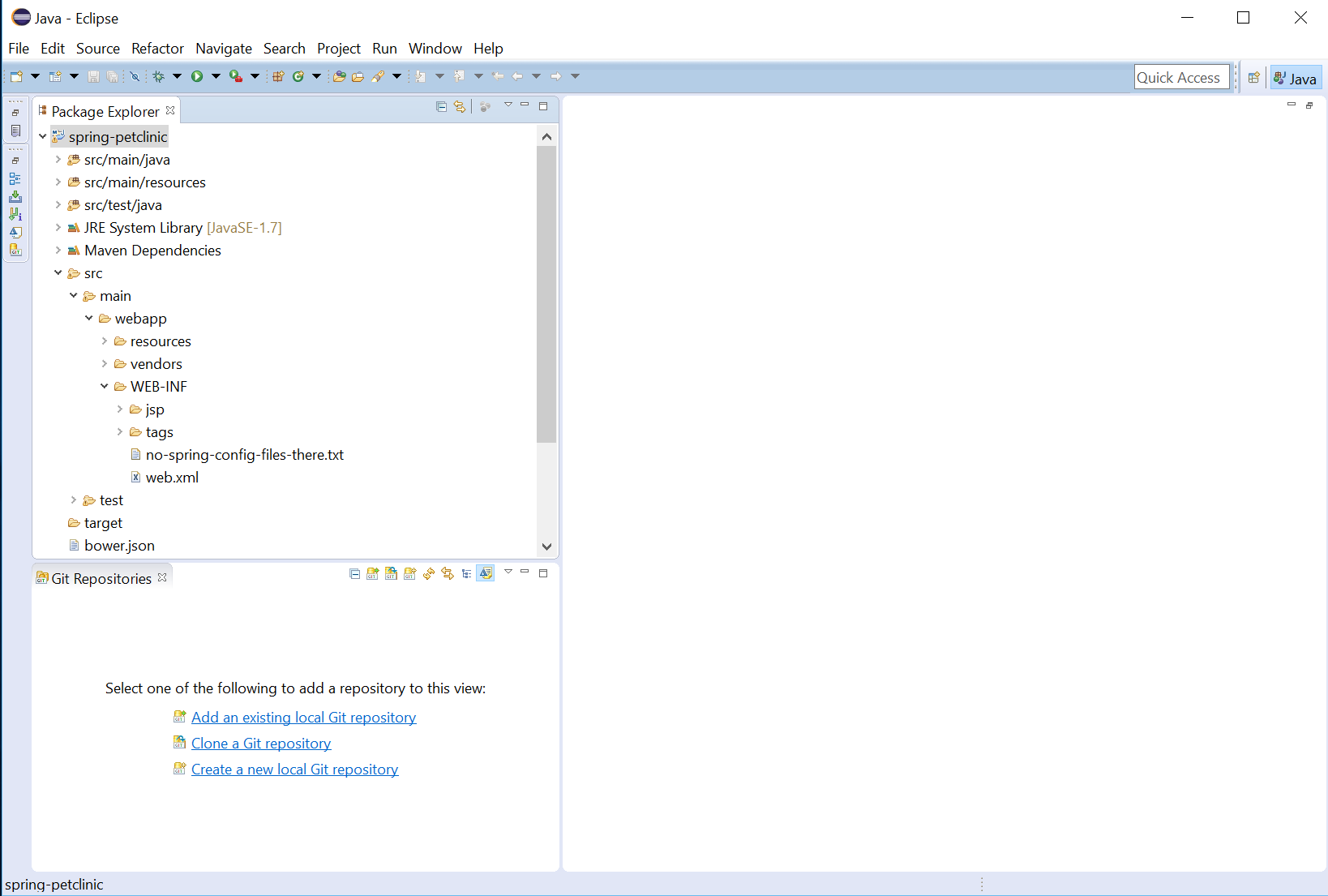
**Steps to share the code using GIT to Visual Studio Team Services scenario to collaborate with the team and enable DevOps Practices.**

1. In Visual Studio Team Services create a new team project, then select and click in Code



**Share your code in using Eclipse**

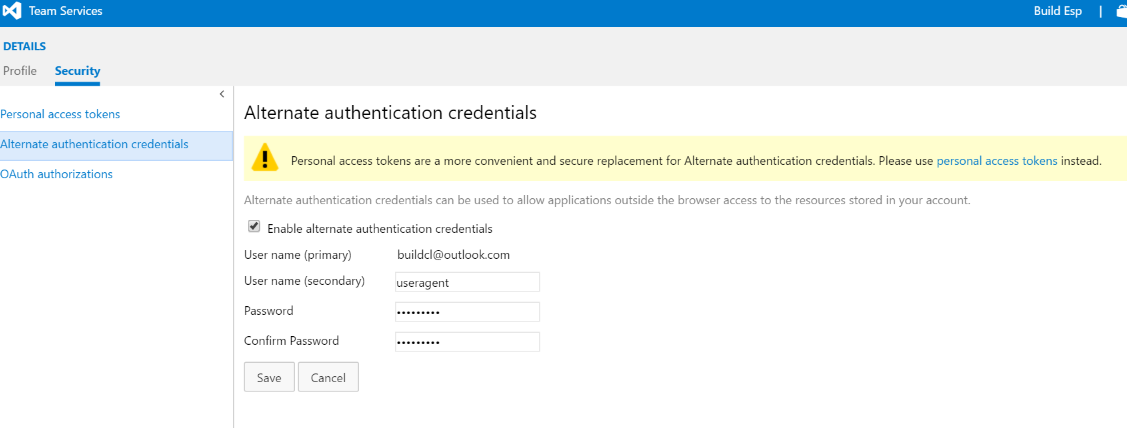
1. In Eclipse show the Git views; Windows -> Show View -> Other -> Git and then select Git Repositories & Git Staging. If you don’t have Git views, install from <http://www.eclipse.org/egit/>



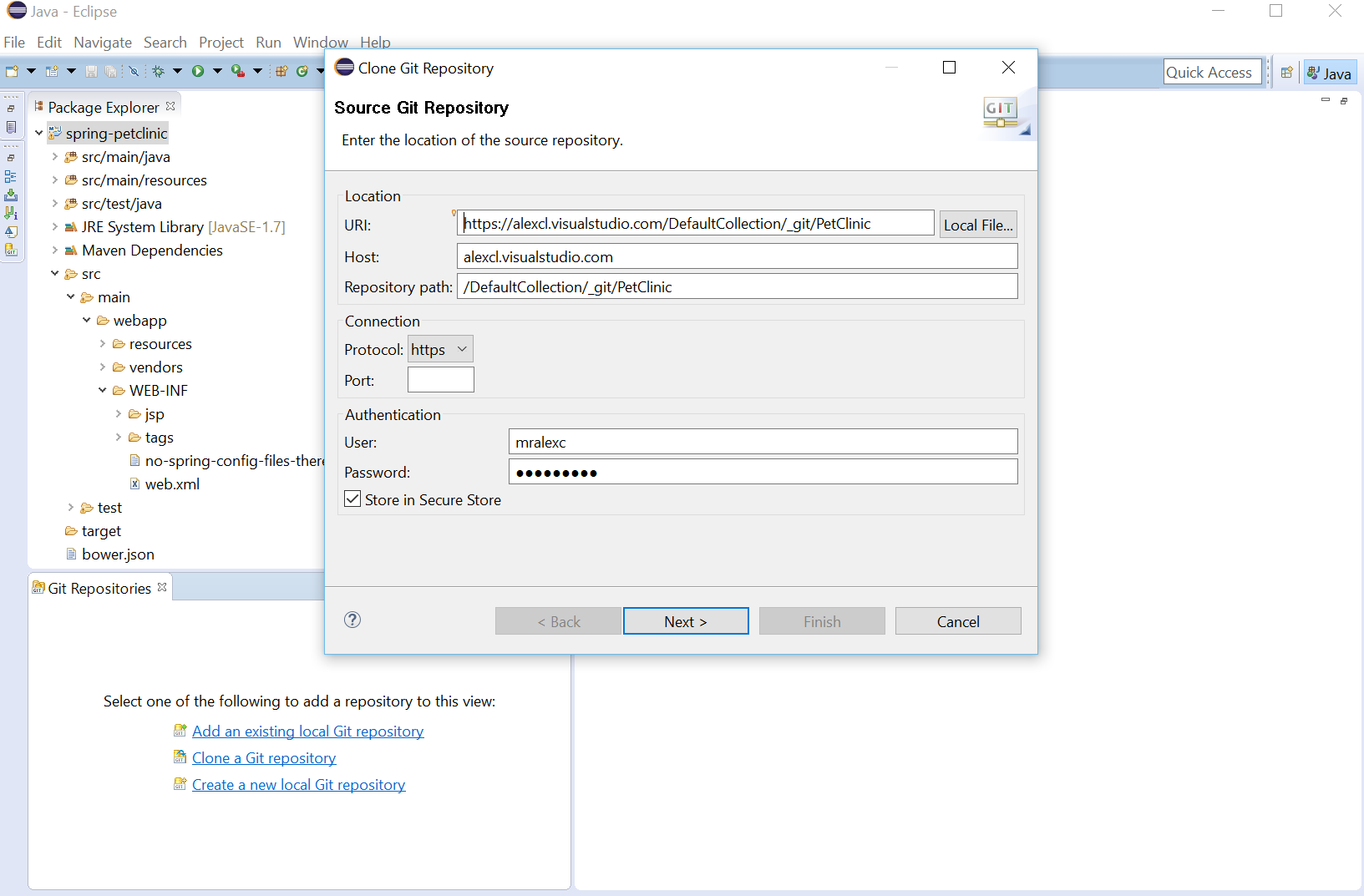
1. Before of Clone you need to create a personal access token, go to VSTS, click in the User and then click in My profile.



1. Click in Security and then *Alternate authentication credentials.* Enable it and put the account secondary information and click Save. For more secure use personal access tokens instead.

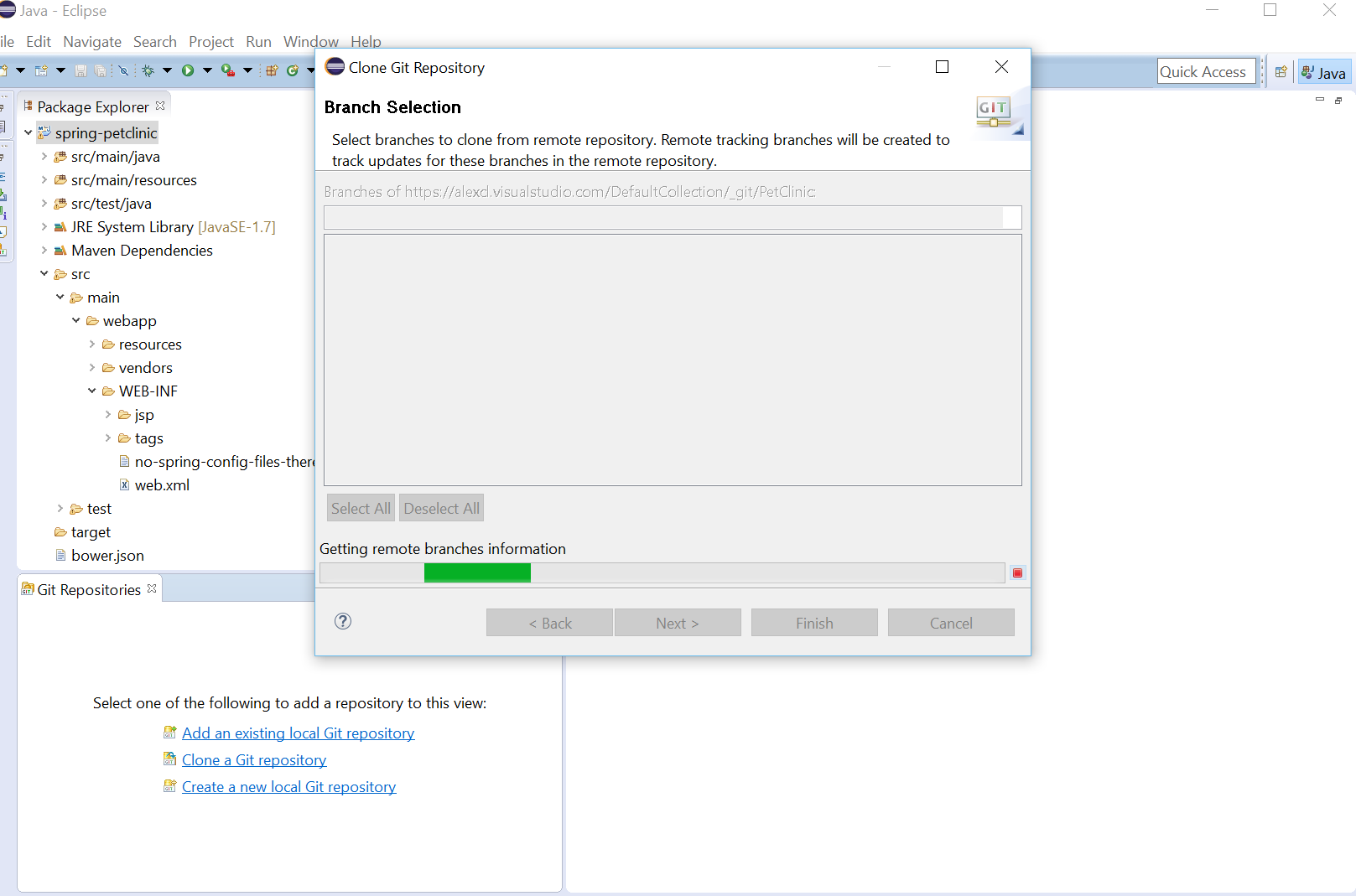


1. Click in *Clone a Git Repositary.* You will clone the Git repository to your team project.

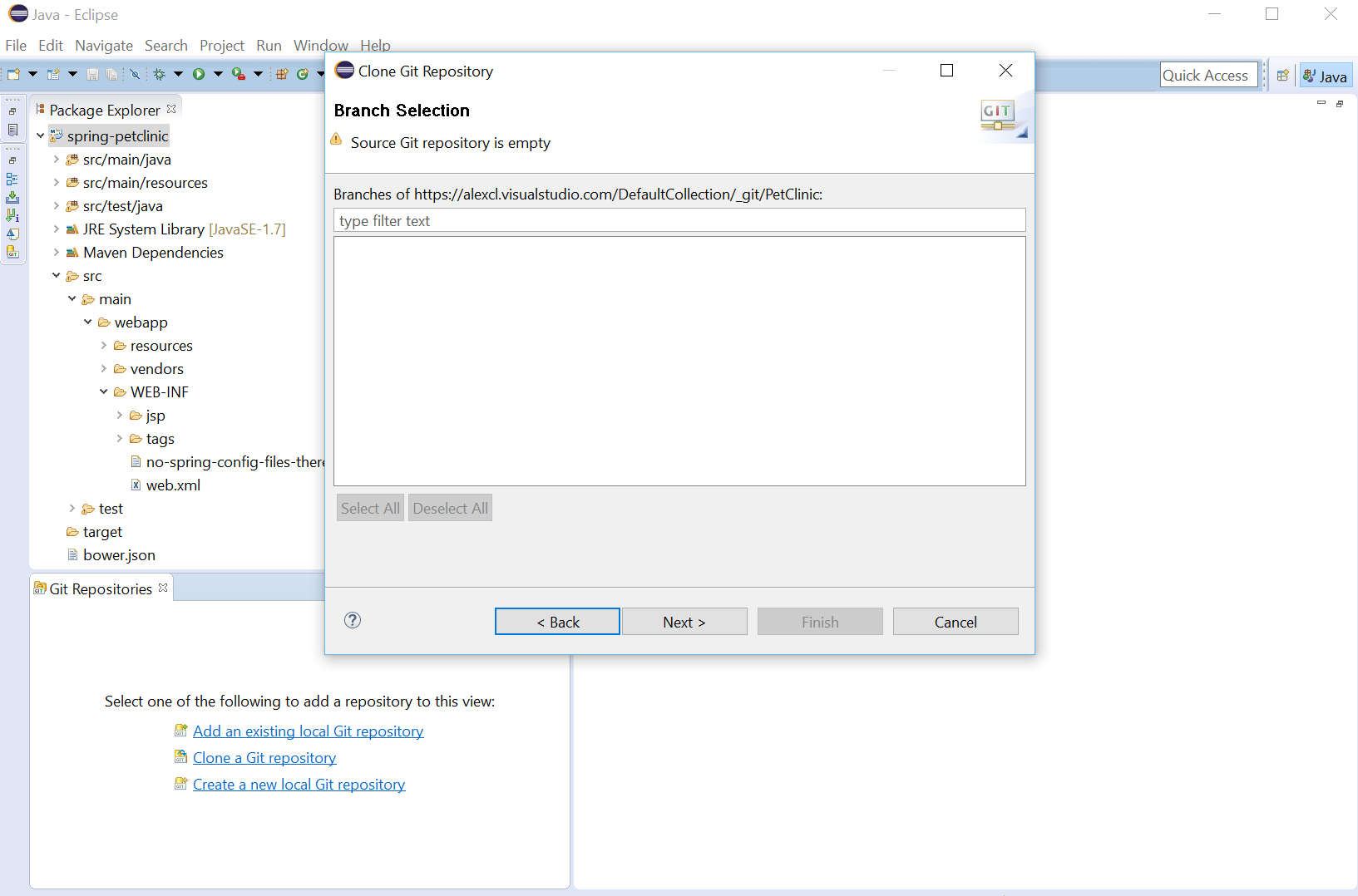


Specify the URL taken from your project repository in Visual Studio Team Services.

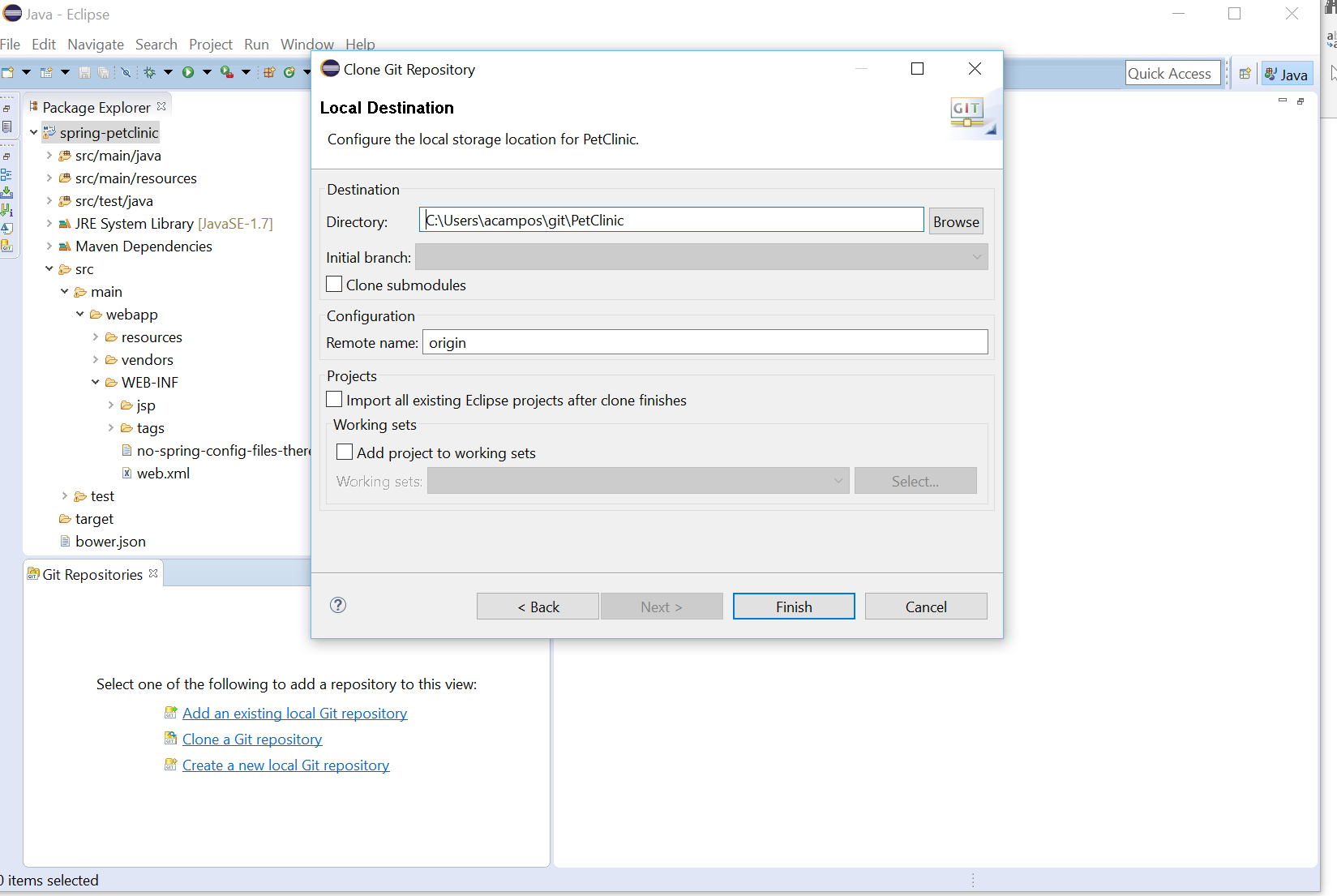
1. If you have branches they will appear,



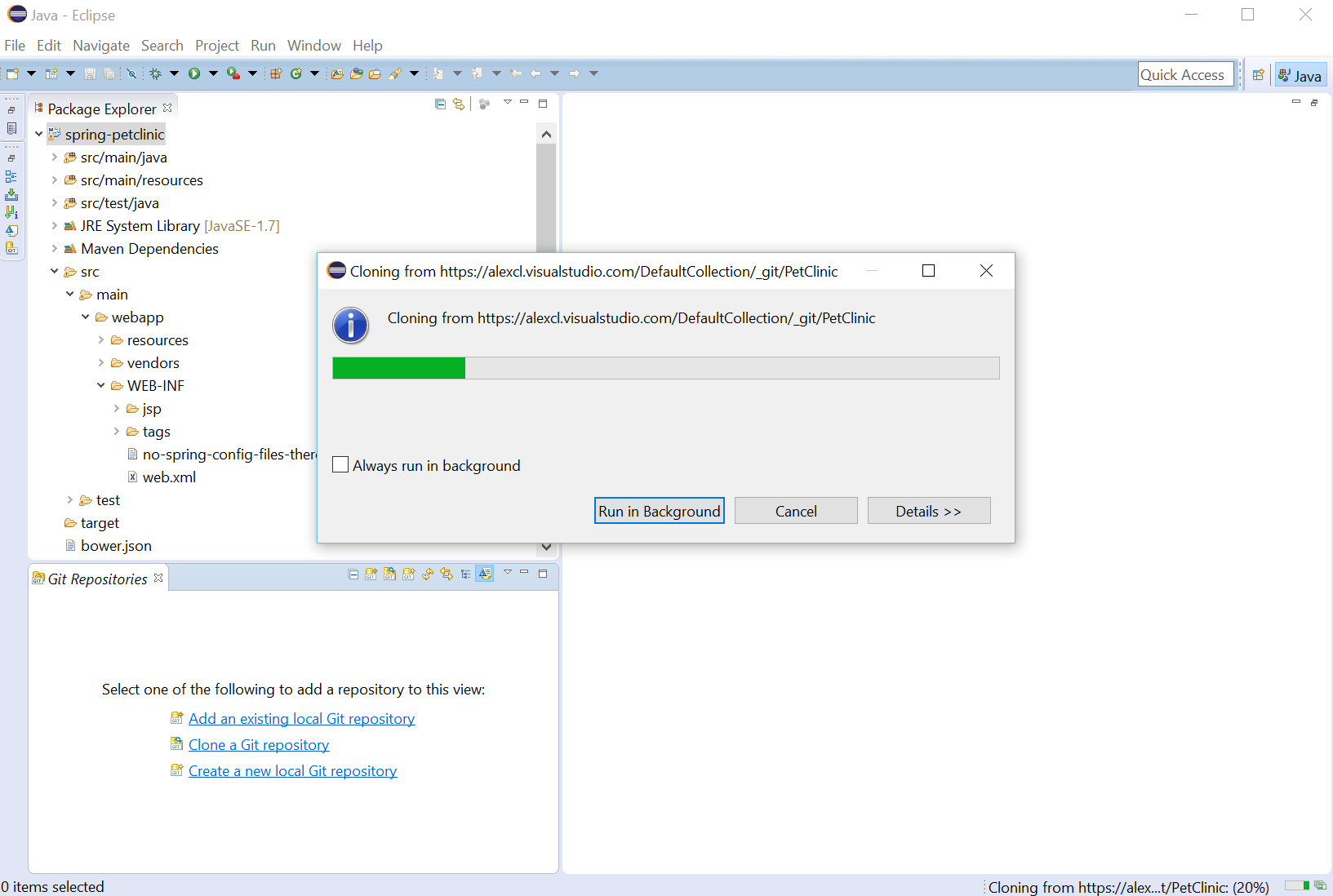
1. if not then click next.



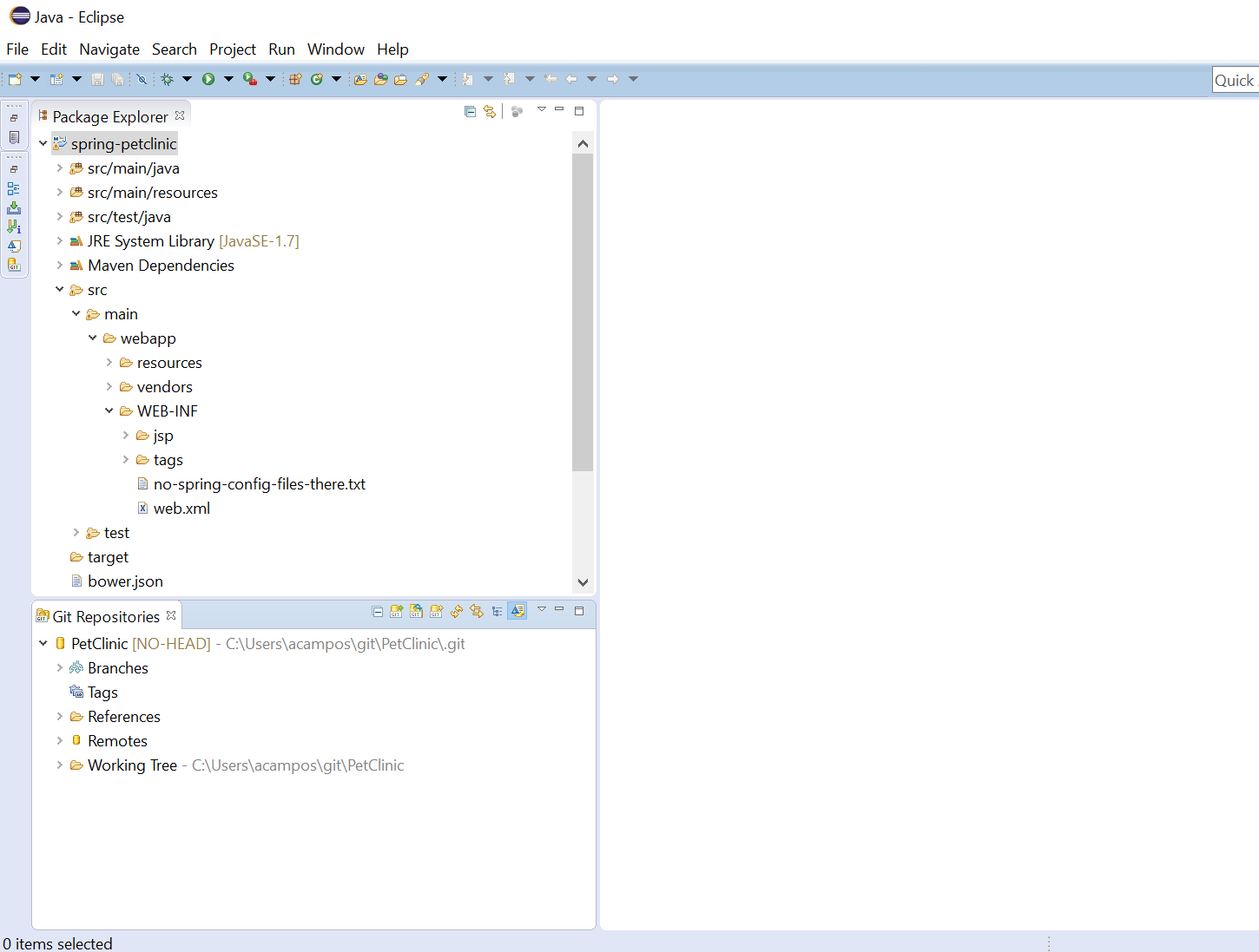
1. Define the local destination



1. Click Finnish

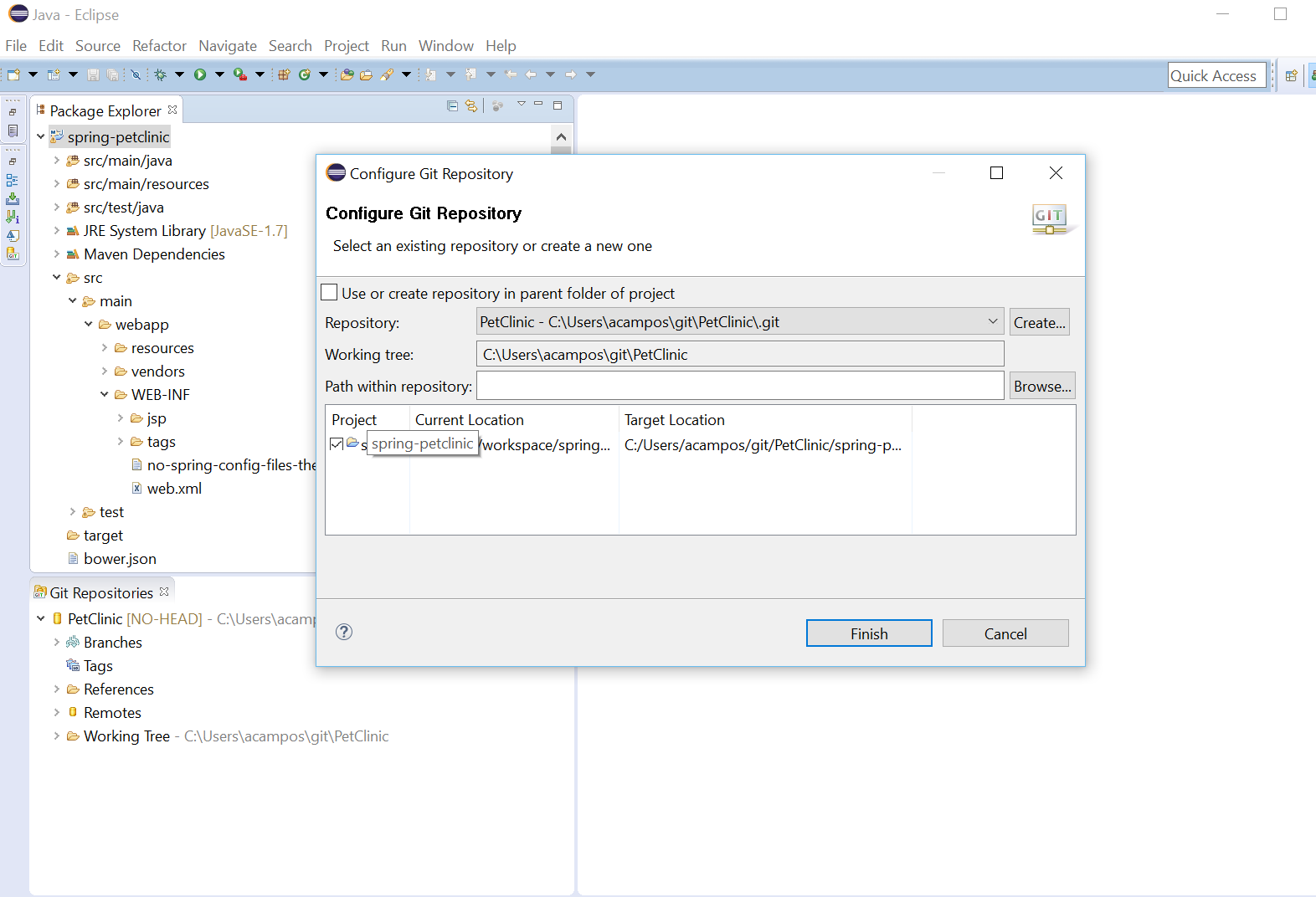


1. The repository of your team is shows up in the Git repositories view

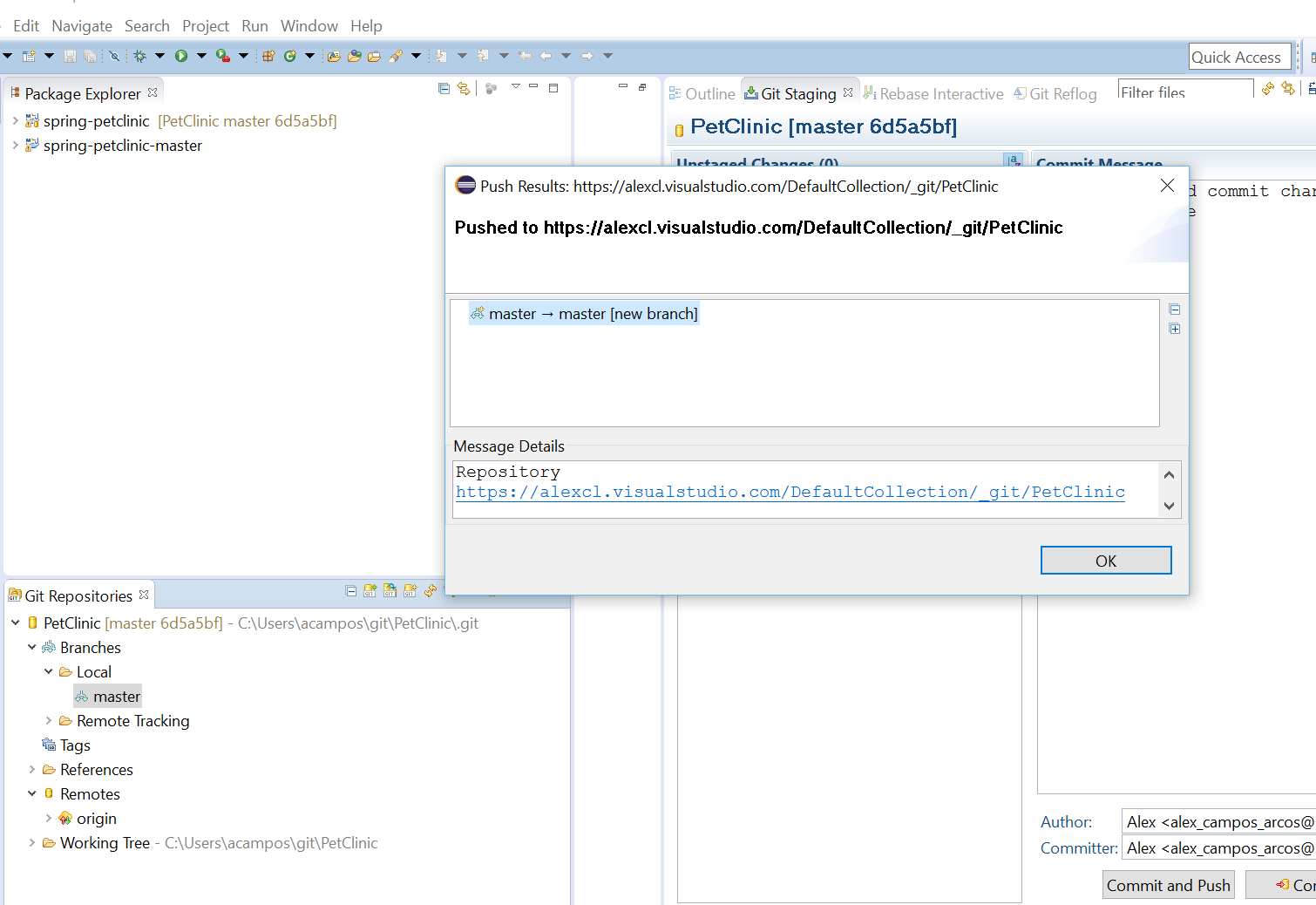


**Push the code to the remote repository**

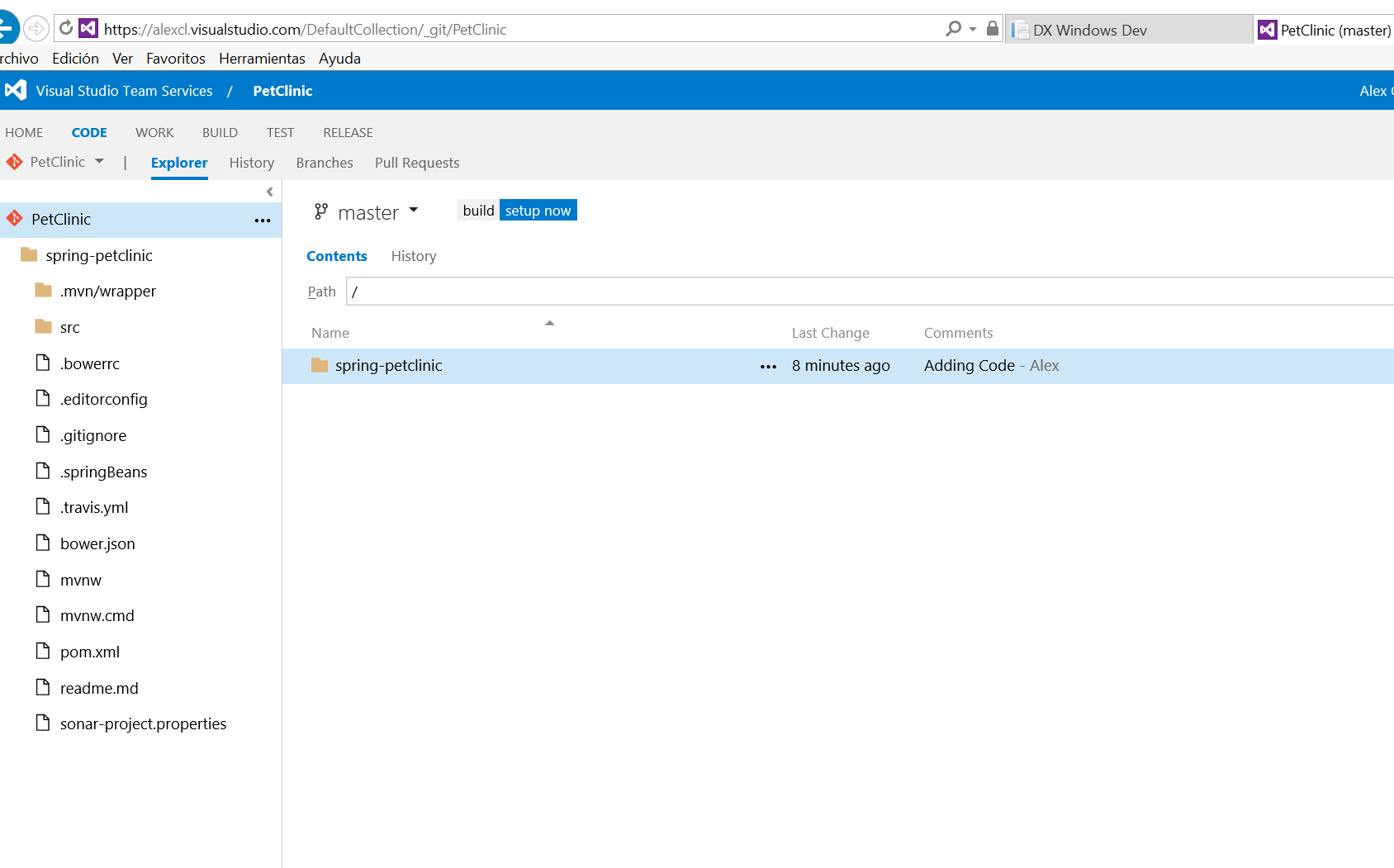
1. In Package Explorer, right-click the project and choose Team, Share Project and Map the remote repository to your working directory



1. Add your file to the Git Index and then Commit and Push



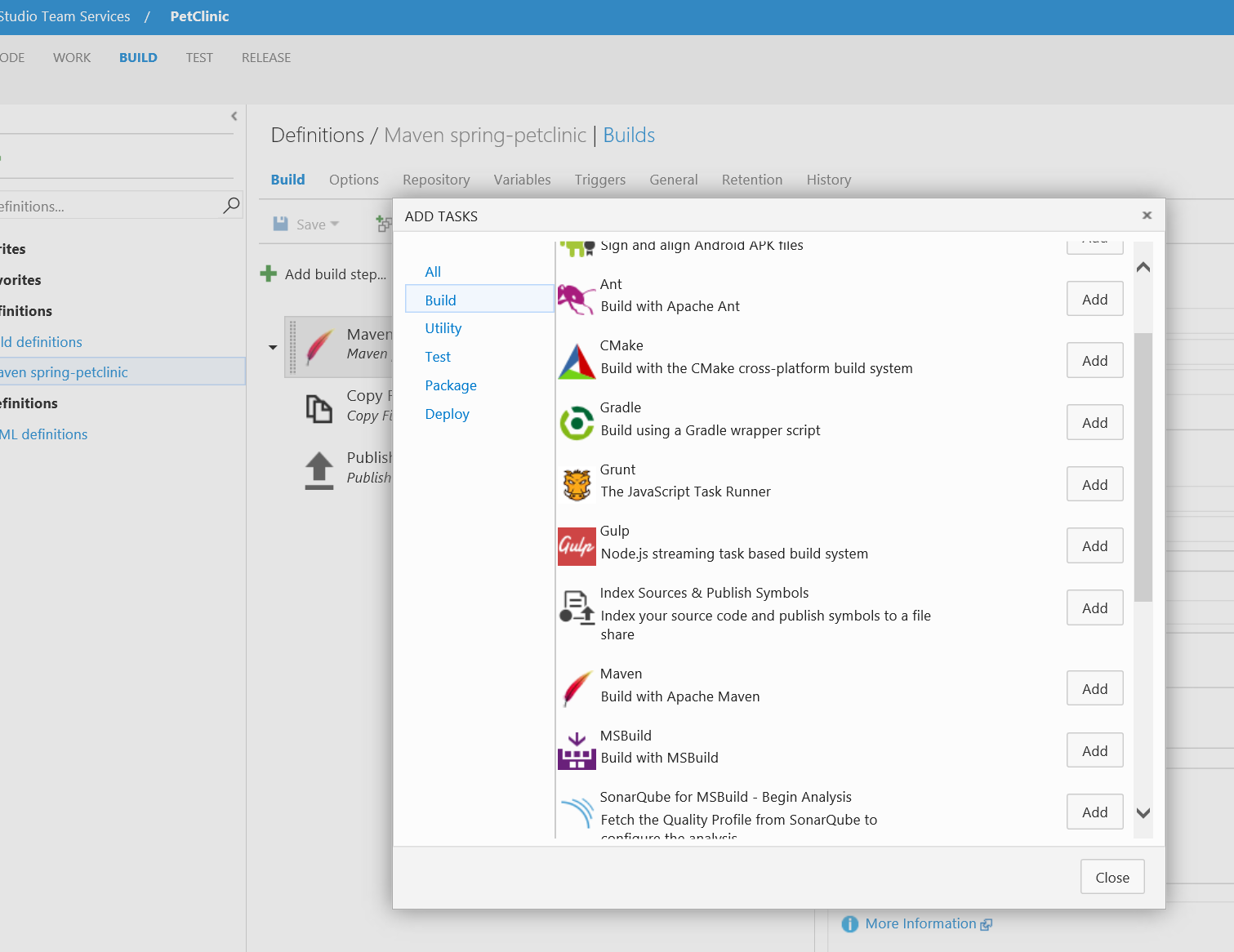
1. After a seconds you will see the code cloned in the Team project at VSTS.



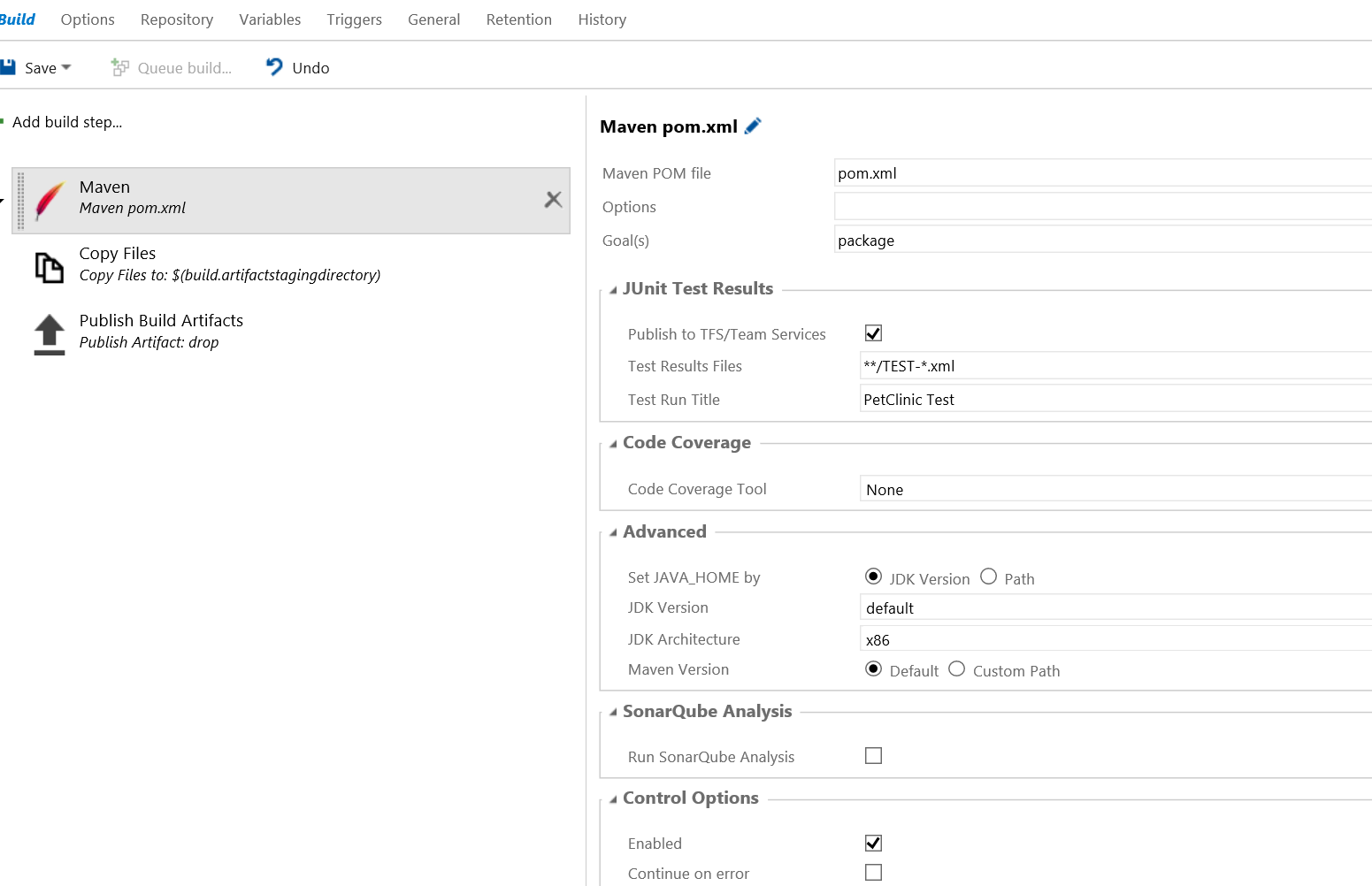
**Creating Build to enable Continuous Integration**

In Azure Portal, create a Web App and copy the FTP hostname and username.

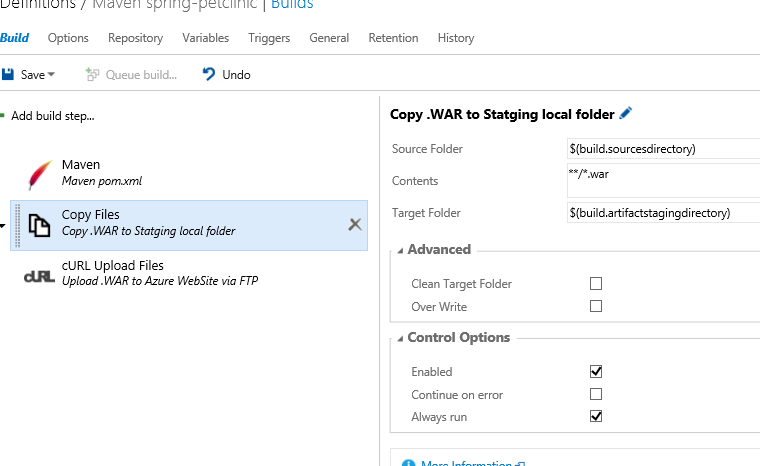
In VSTS go to the team project and click in Build, then click in plus sign and create a new Build definition. Add build step and select Maven Tasks.



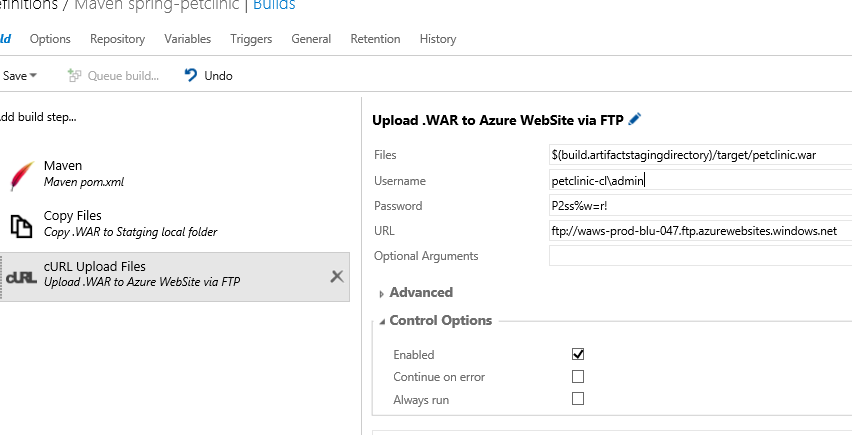
Complete with the following values to Maven task



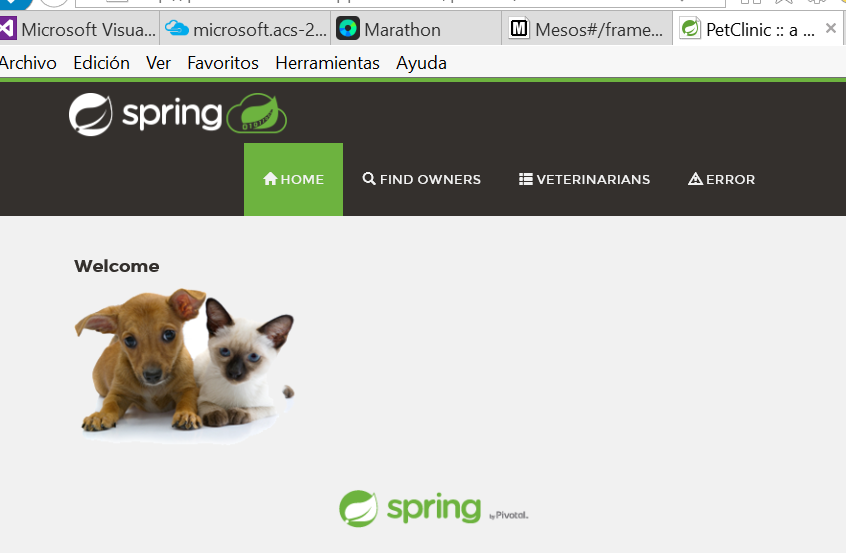
Change in Copy Files the content to upload the .war file



Add a cURL task to upload .war file to Azure WebApp via FTP and then click Save.

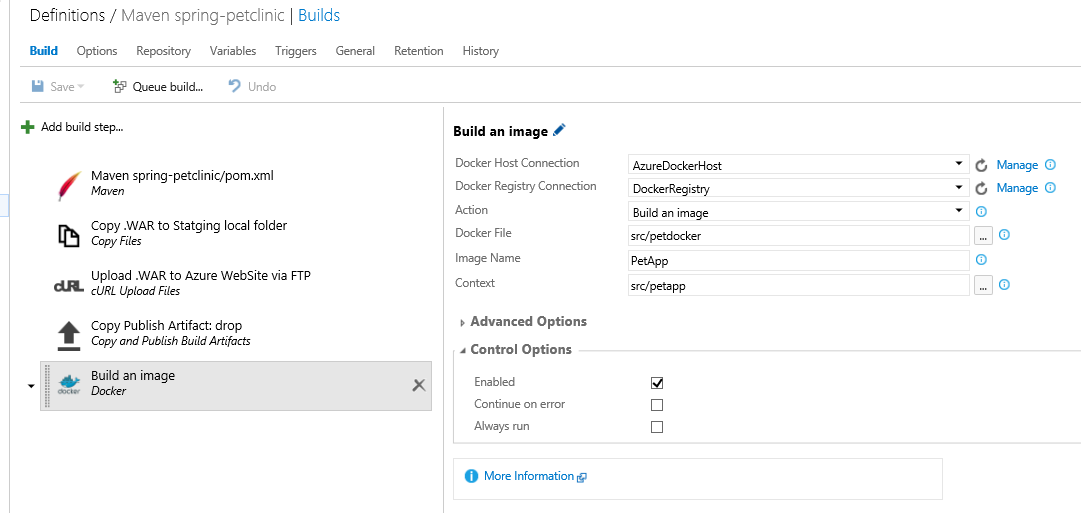


Before the test passed we can navigate to the application published in a WebApp in Azure.



**Azure Container Services**

Also you can choose a step at the definition to take some actions from Docker.

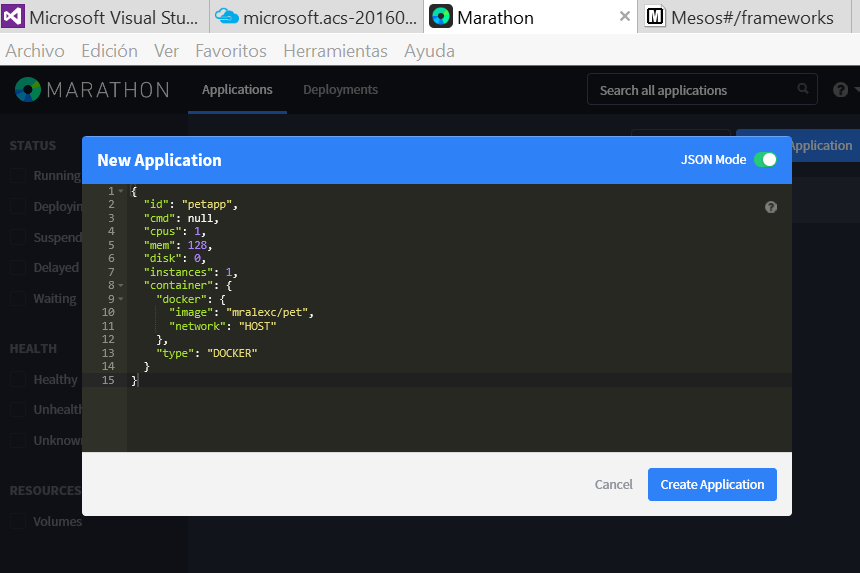


In our case, we are building an image that will contain Ubuntu, Tomcat and the Petclinic application. We are using the simple Dockerfile:

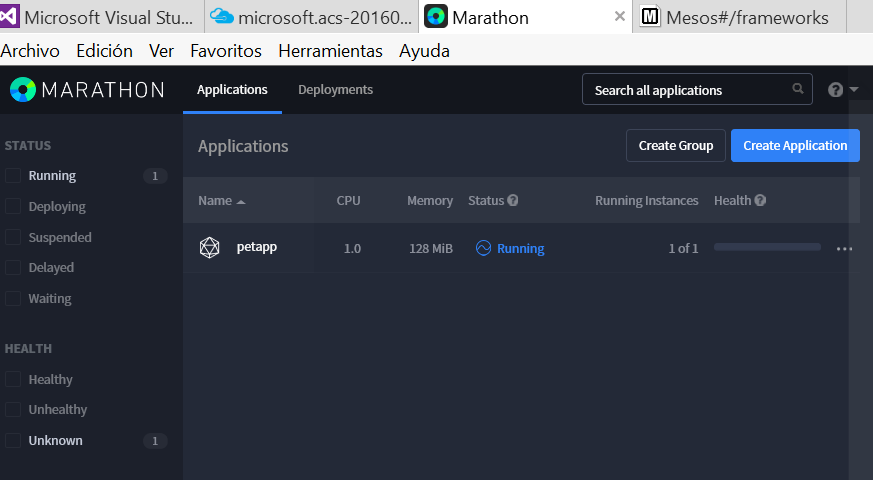
FROM mralexc/pet  
MAINTAINER mralexc  
#Update the repositary source list  
RUN apt-get update

This image will pushed to dockerhub allowing to create containers to deploy our application in Azure Container Services managed by Mesosphere.

In Mesosphere in Json mode we can create the application



Running



Then you can scale the application with Mesos in charge of manage the infrastructure required.