A DevOps environment using Cloud Center: demo guide

Version: 0.3

Date: March, 26 - 2018

Document created by	Job Title
Alessandro Maretti	Associate Systems Engineer
Sylvain Larue	Associate Systems Engineer
Ololo Jaques Shembo	Associate Systems Engineer

Content provided by	Job Title
Luca Relandini	Technical Solution Architect, Solution Lead
Riccardo Tortorici	Software Systems Engineer
Fabio di Niro	Consulting Systems Engineer
Stefano Leonardo Gioia	Technical Solution Architect

Introduction: CloudCenter and DevOps

Cisco Cloud Center provides customers with a single, intuitive platform that helps them manage the entire application lifecycle across simple or complex hybrid IT environments. The Cisco Cloud Center platform provides a compelling solution for modern IT organizations whether they are moving their first applications to the cloud (either public or private), implementing self-service IT, or wanting to gain visibility and control across a vast portfolio of clouds, applications and users.

This **demo** intends to show how it is possible to deploy **a fully configured and functional DevOps environment** in a matter of minutes using the power of Cisco Cloud Center. These activities would usually require several days as well as interactions between different teams. The demo runs on the Cloud Center environment that has been set up in Rome Lab, thus in order to be able to use it, you must be connected to the Lab using a jump host.

DevOps is a software engineering culture and practice that aims at unifying software development (Dev) and software operation (Ops) by automating and monitoring all the steps of the software construction.

A common DevOps environment is composed by a number of building blocks (of course, methodology matters a lot):

- A machine, physical or virtual, dedicated for each developer populated and configured with all the necessary tools. For example, for a project of Java development: JDK, Eclipse and relative plugins, Maven, etc.
- A version control and management repository where all the source code of the project is stored
- An orchestrator that
    - retrieves the source code from the version control system whenever an event is generated (e.g. a new commit)
    - compiles the code
    - stores the compiled version of the software in an appropriate repository
    - deploys the new application in a specific context (e.g. development, test, operations)

- triggers tests (functional, performances, etc.) to qualify the deployment for next stage

Customer Scenarios

3 different user Scenarios have been developed. The Scenarios are written to be executed on sequence.

- Scenario 1: creation of the virtual environment for software development automation
- Scenario 2: demonstration of the newly created environment to deploy and modify an application
- Scenario 3: promote a deployment from one context to another

Scenario 1 – create a lab for software development

In Scenario 1 we will automatically deploy a DevOps environment as described above using Cloud Center. The overall objective is to reduce the time between the project is committed and when the Development and Operations teams start working on it.

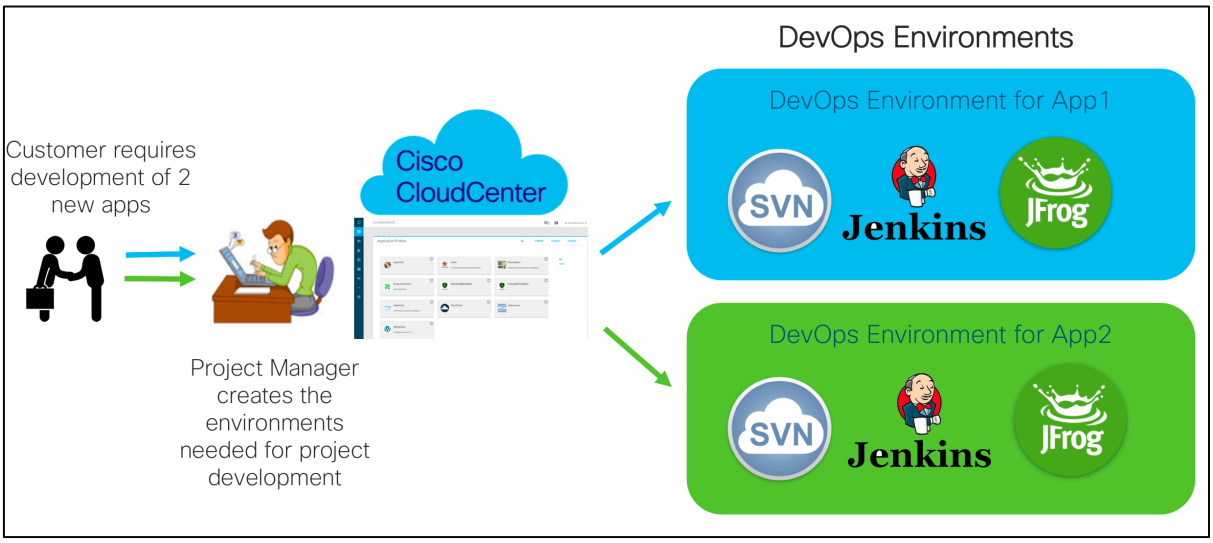


Figure 1: DevOps environment, from commitment to development in a matter of minutes

1 Deployed tools

In this demo, the following tools are deployed (and integrated end-to-end):

- 1 VM with Subversion (SVN) for code version control and management
- 1 VM with Jenkins for orchestration and automation
- 1 VM with JFrog Artifactory as the repository manager of the compiled code

2 Demonstration

- Access to the Homepage of Cisco Cloud Center (https://192.168.130.200 or https://ccm.rmlab.local). Login using *admin@gcloud.org/C1sco123* and *gcloud* as a tenant. Gcloud stands for Government Cloud and it's an example of a public company (Figure 2).

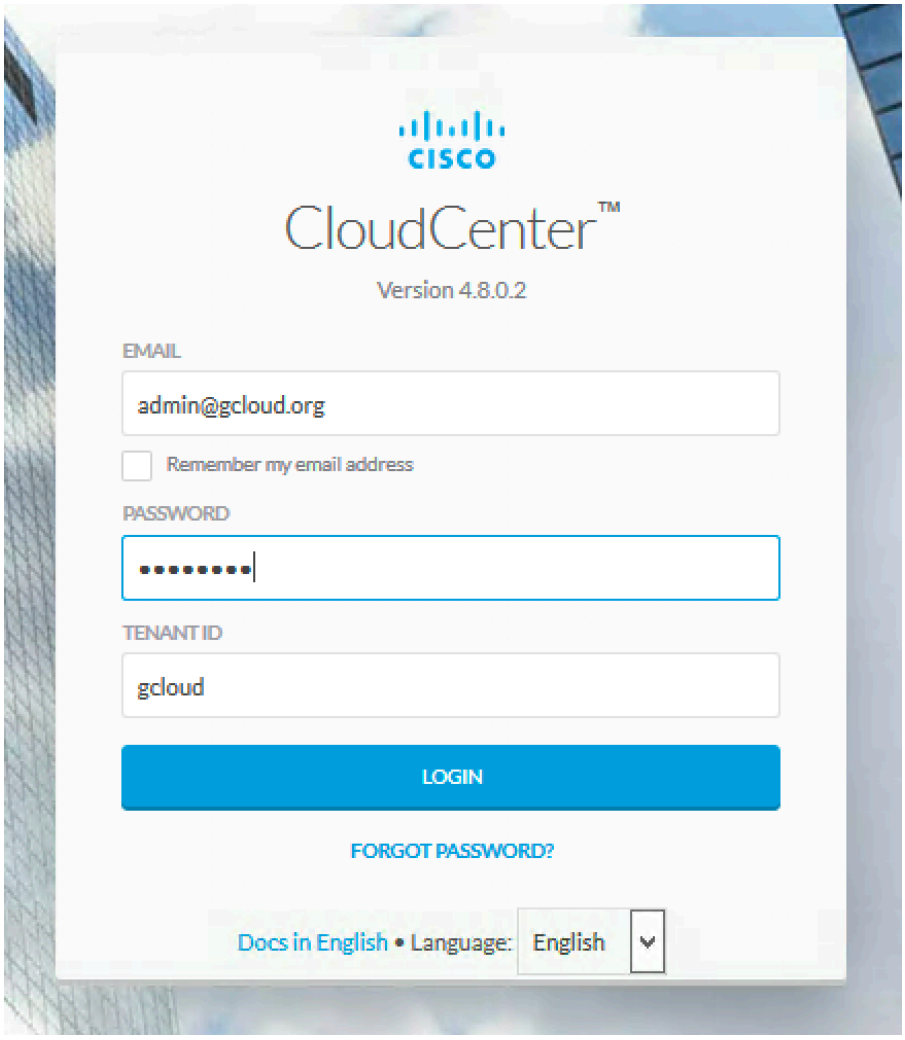


Figure 2: login to CloudCenter as administrator of gCloud, and example of government Cloud

- Under App Profiles (Figure 3), look for **DevOpsEnv** and click on it. The App Profile section lists all the application profiles that have created for that Tenant and for which the user has the rights.

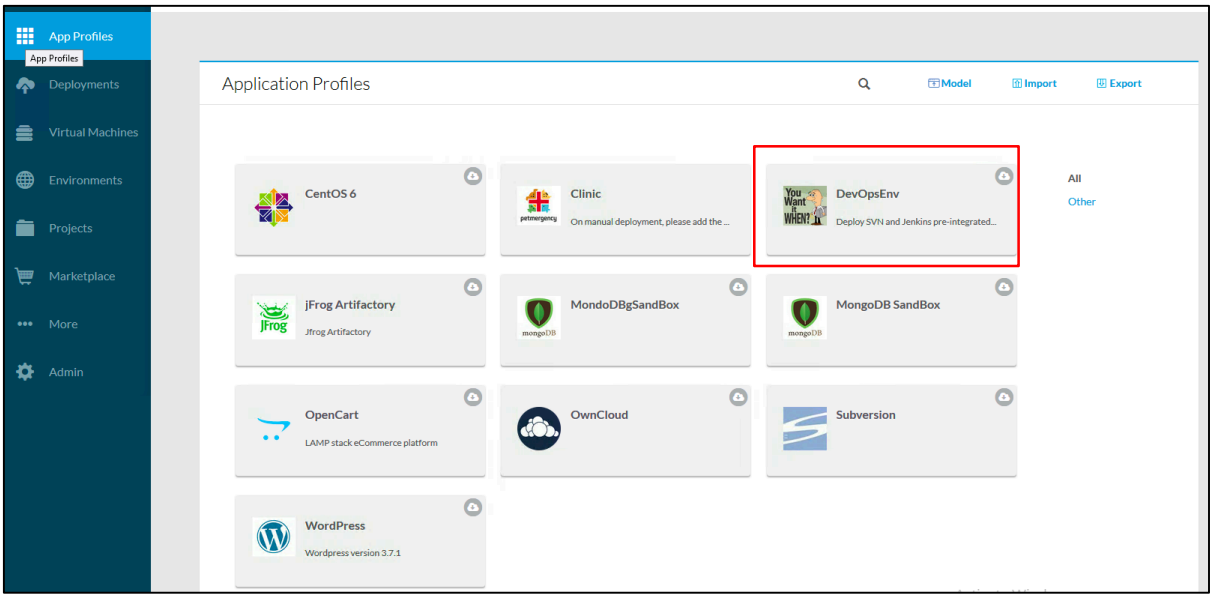


Figure 3: deploy a new Application Profile

- Under General Settings (Figure 4), look for Deployment Name and assign the name of the new project (e.g. “ProjectX”). Notice that there are other parameters that have already been defined in the configuration of the App Profile but can be modified as pleased (e.g. name of the repository, username and password of administrator). For this Demo, leave them at their default value. Click on **Next**.

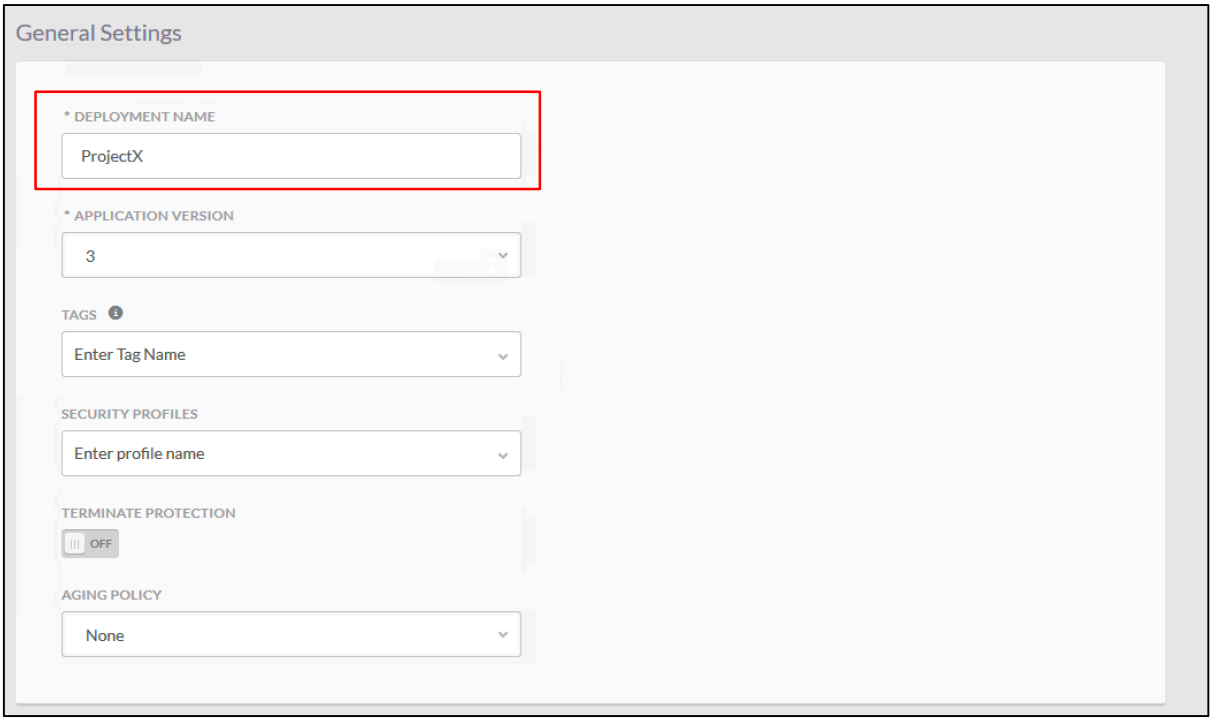


Figure 4: general settings

- The next screen (Figure 5) allows to select the VM size, the target DC (or target cloud region) and the cluster where I want my VM to be deployed to. The default setting is to deploy on the VMWare cluster of LabRome and to use Tiny machines. Leave the default values and click on **Deploy**.

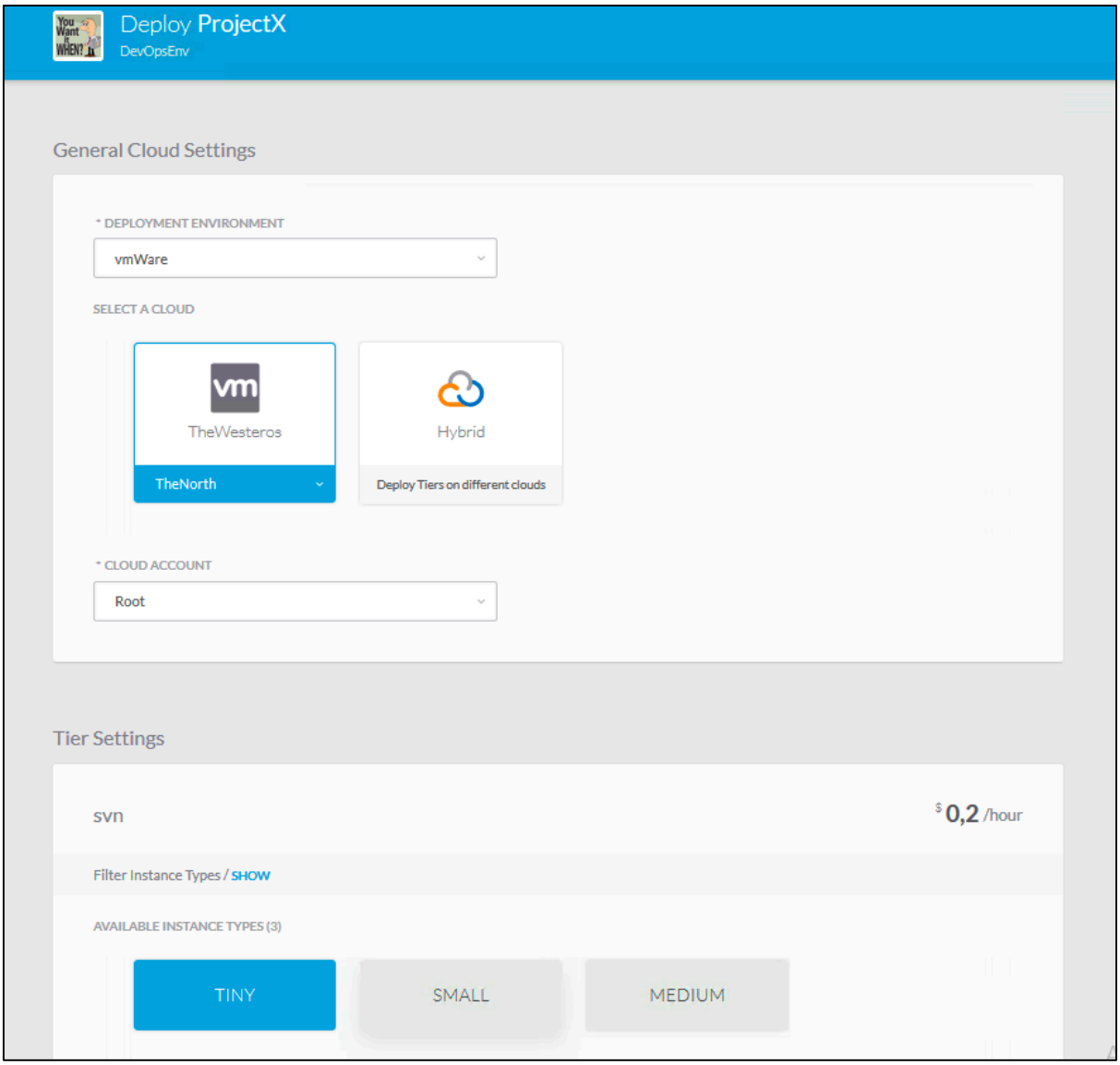


Figure 5: choose where to deploy the environment

- The following screen shows a summary of the ongoing deployment of the environment. At the moment of creation, the deployment state is **In Progress** (Figure 6), meaning that the virtual machines are starting up and or the configuration scripts are still running. Once the deployment is completed the status will be **Deployed** (Figure 7).

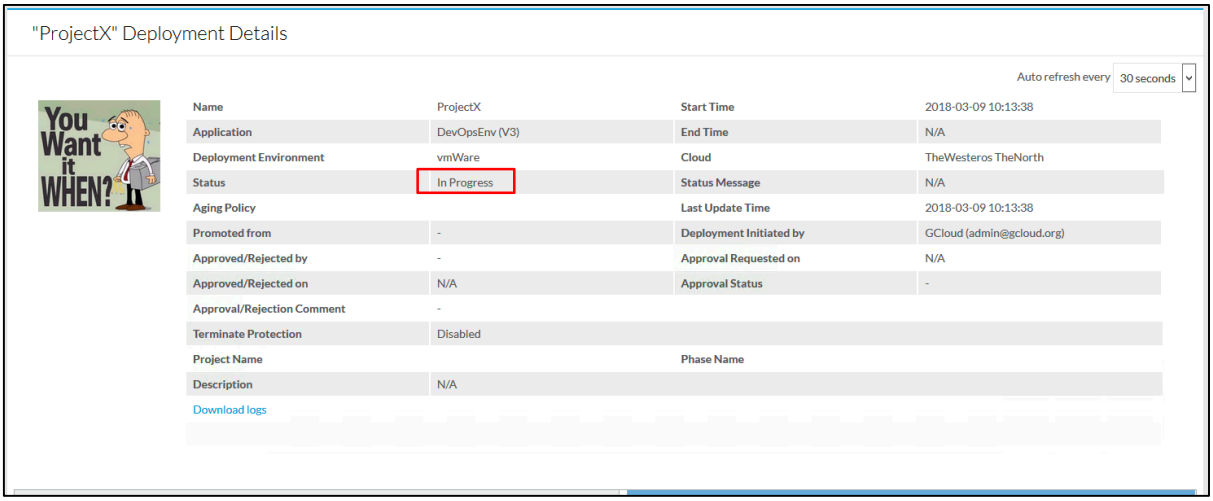


Figure 6: deployment state is "In Progress"

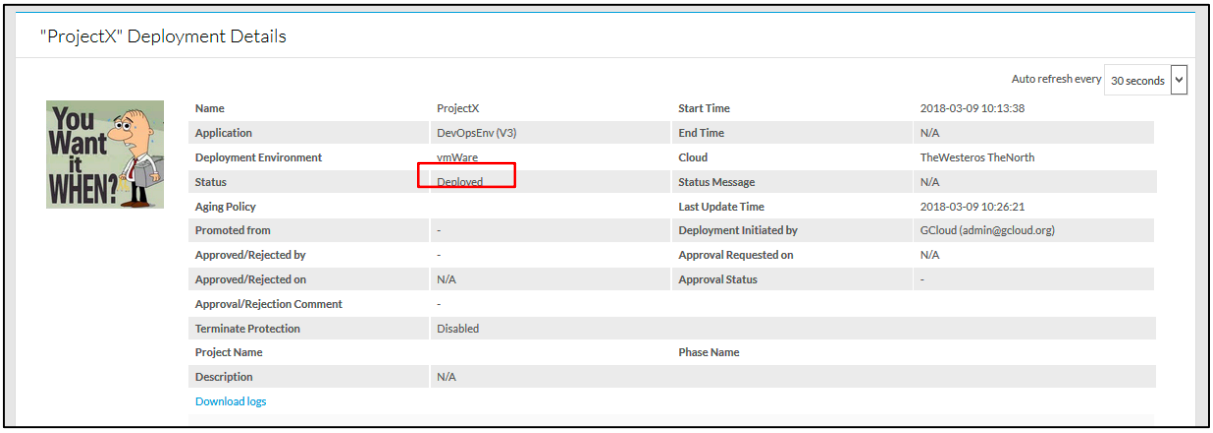


Figure 7: deployment state is "Deployed"

- Once the deployment is completed you can access to the details of the virtual machines just created by clicking on the VM on the **left** (in this case Jenkins – Figure 8) and expanding the running nodes on the **right**. Notice
    - The **IP address** of the VM
    - The possibility to access to the VM through **SSH** directly through browser (that is very important, because connectivity and security are proxied by CloudCenter and you don't need access to the network where the VM is deployed).

In this specific example, the following information can be derived, please notice that in your specific demo these IP addresses will differ and you will have to note these manually (Figure 9):

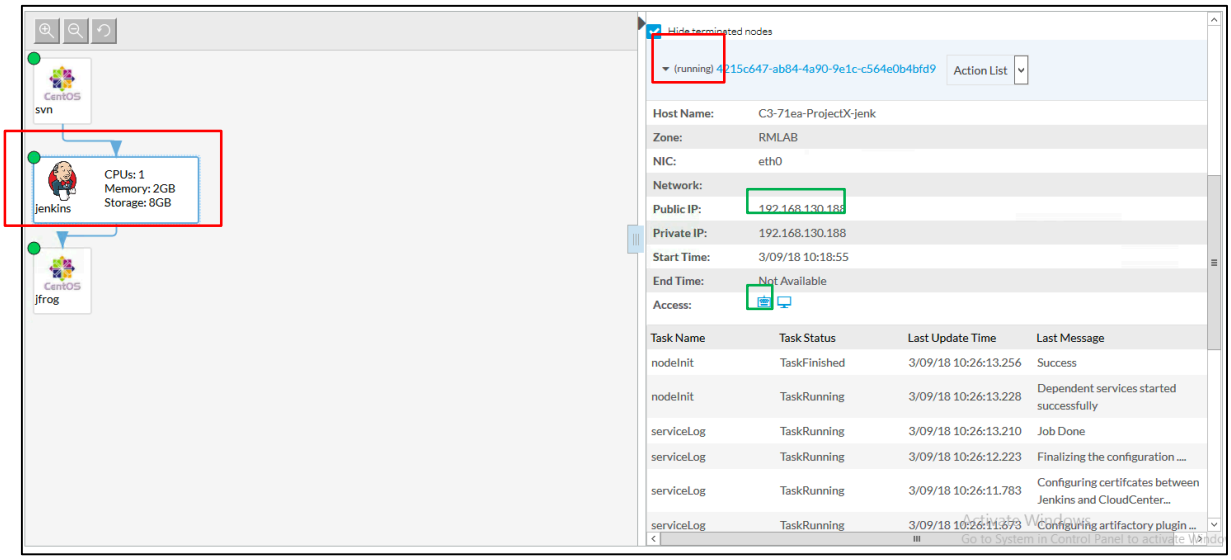


Figure 8: accessing to the details of a VM

Server	IP address	URL access
SVN	192.168.130.132	http://192.168.130.132/svn/repo1/
Jenkins	192.168.130.188	http://192.168.130.188:8080
jfrog	192.168.130.156	http://192.168.130.156:8081

Figure 9: IP and HTTP addresses for the VM

- Under the Deployments section (Figure 10). it is possible to see all the deployed App Profiles with options, for example, to suspend, terminate and upgrade.

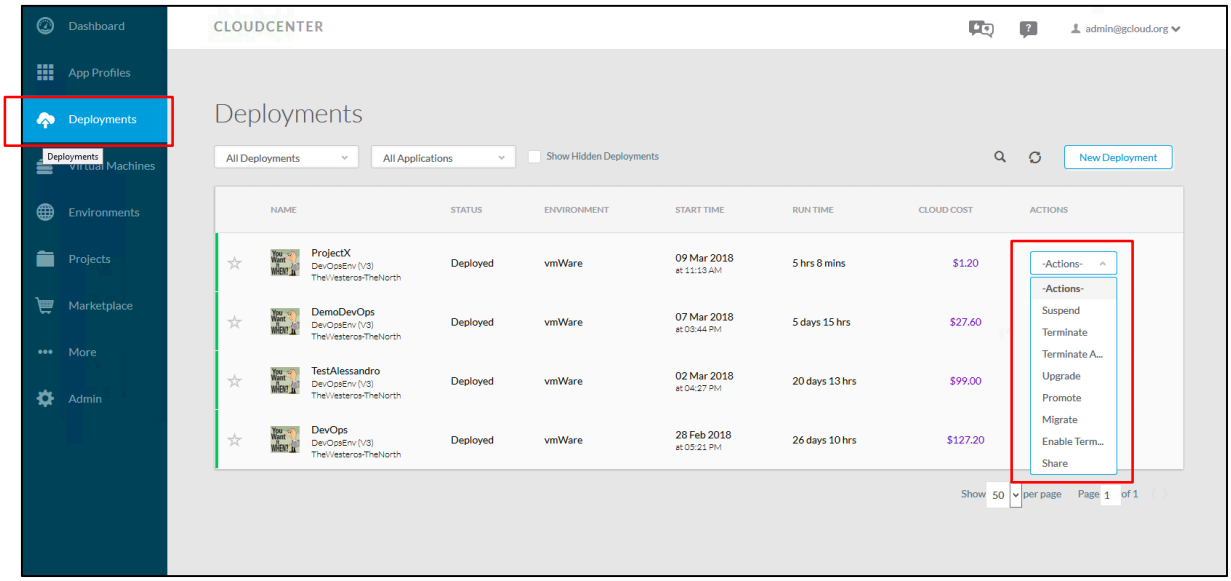


Figure 10: list the currently deployed App Profiles

- At this time, the SVN repository just created (repo1) is empty, it will be used in Scenario 2 (Figure 11)

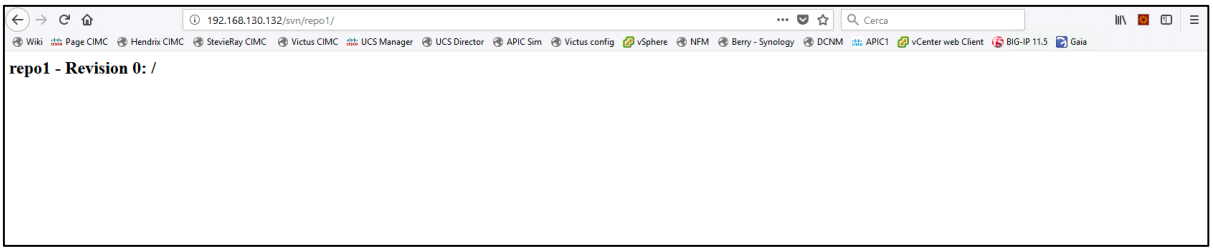


Figure 11: accessing to repo1

- The Jenkins orchestrator (Figure 12) is already configured with a connection to the repository (repo1) and with Cloud Center (deploy). The only manual configuration we need to do is to set the login and password of the repository.

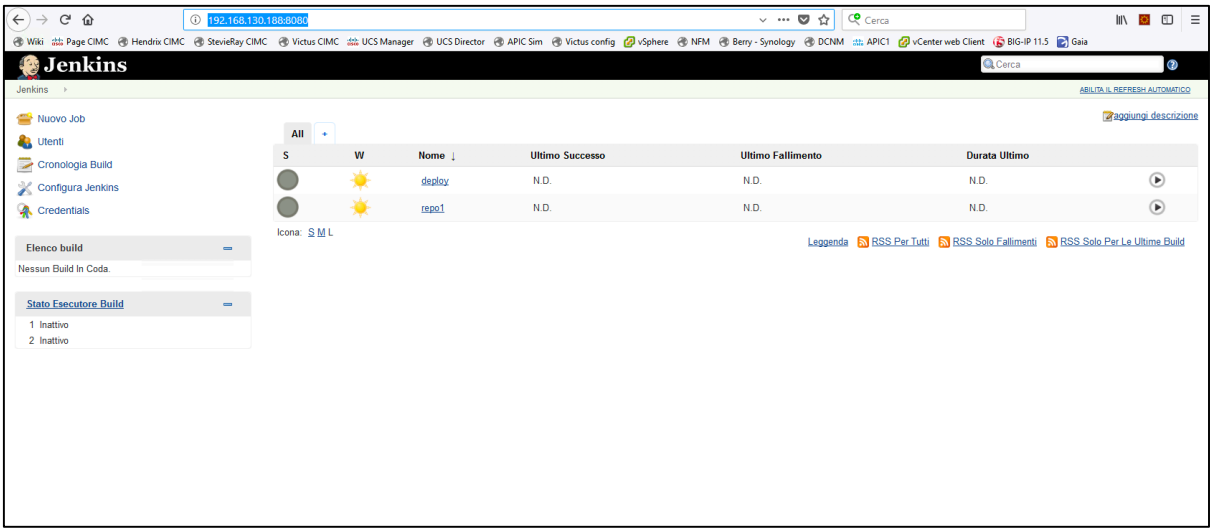


Figure 12: accessing to Jenkins orchestrator

- Click on repo1 (Figure 13), then configure, then enter credentials through username/password authentication (user001/C1sco123). At this point, Jenkins is fully configured.

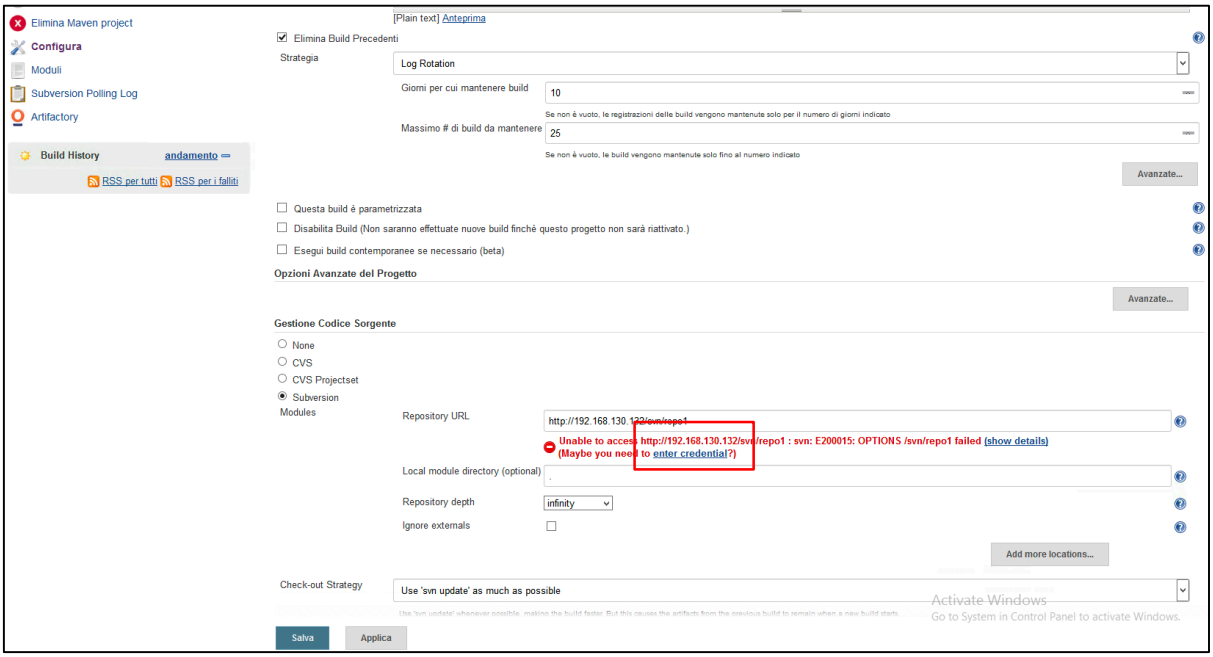


Figure 13: setting up the repository

- The jFrog Artifactory is already configured to received binaries from Jenkins (Figure 14)

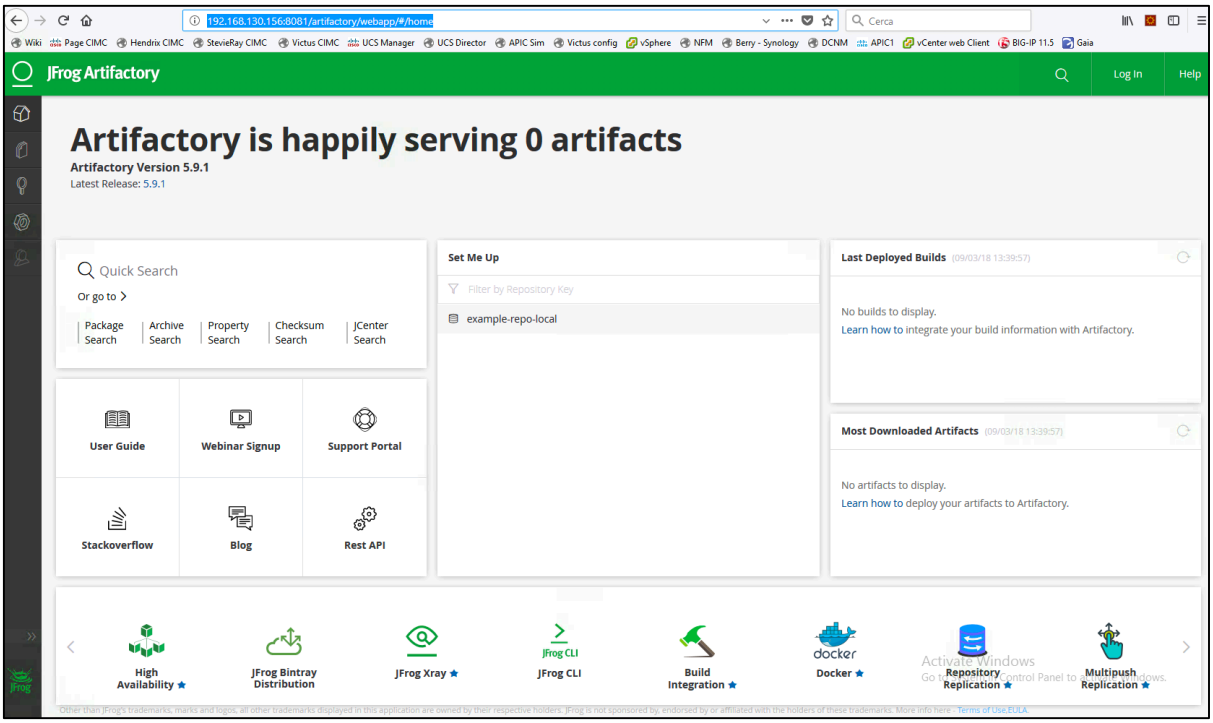


Figure 14: accessing to Artifactory

- The last configuration to do is to set up the Repository of Cloud Center to point to our Artifactory repository, so that new builds of your application can be deployed automatically as they are released.
- To do that we go back to Cloud Center, on the left click on More and then **Repositories** on the left, on the repository gCloudArtifactory, select edit as action on the right (Figure 15)

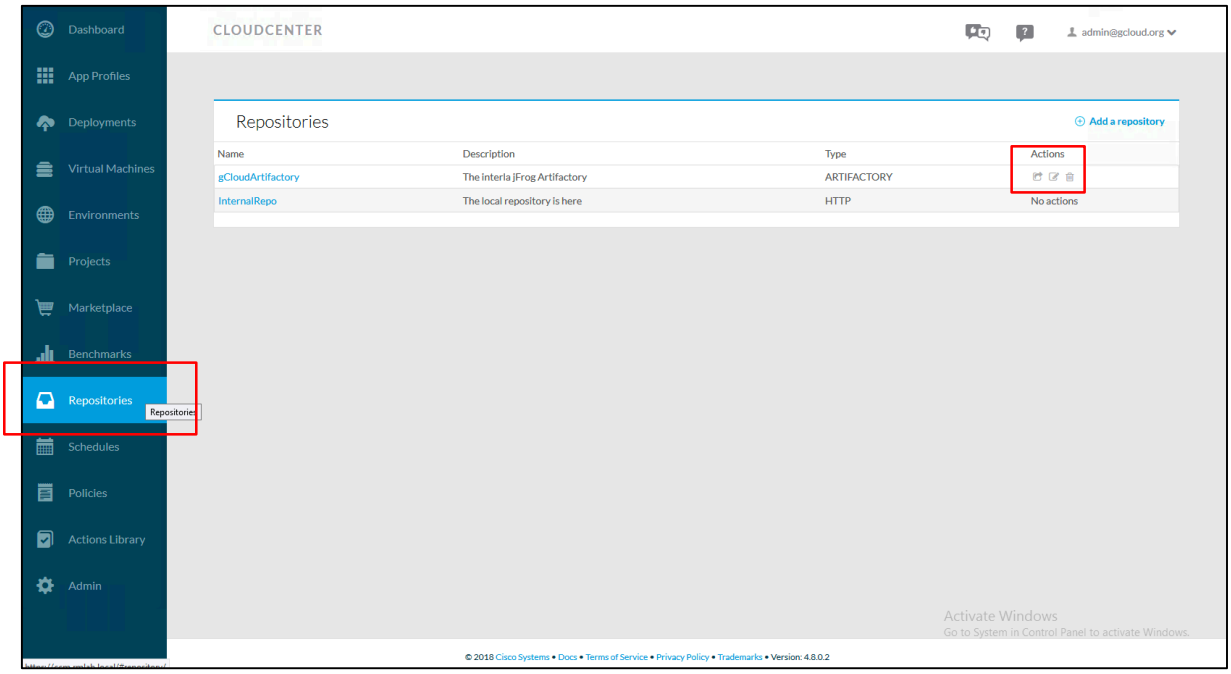


Figure 15: setting up the repository on Cloud Center (1)

- Change the hostname with the IP address of the Artifactory server (Figure 16), in our case 192.168.130.156 and Save.

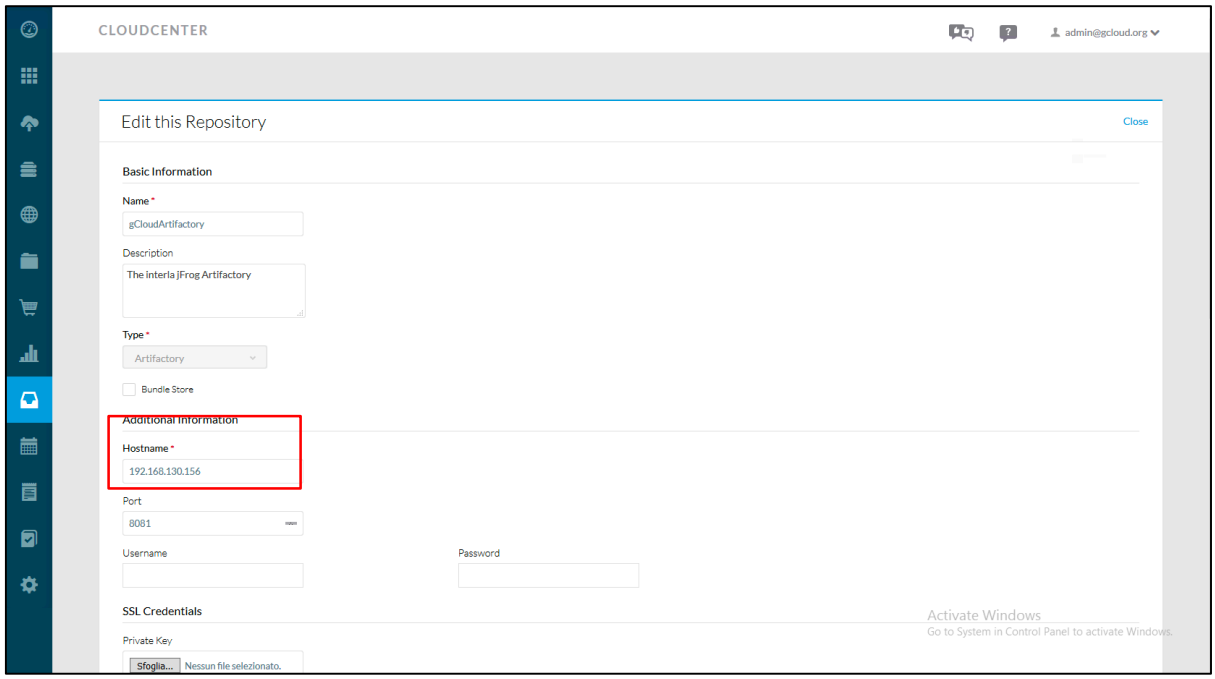


Figure 16: setting up the repository on Cloud Center (2)

Scenario 2 – lifecycle of an application release

In Scenario 2 we will demonstrate how a developer can use the just created environment to start writing an application that is compiled, built and deployed in a test environment. Figure 17 shows the list of operations automatized by the environment.

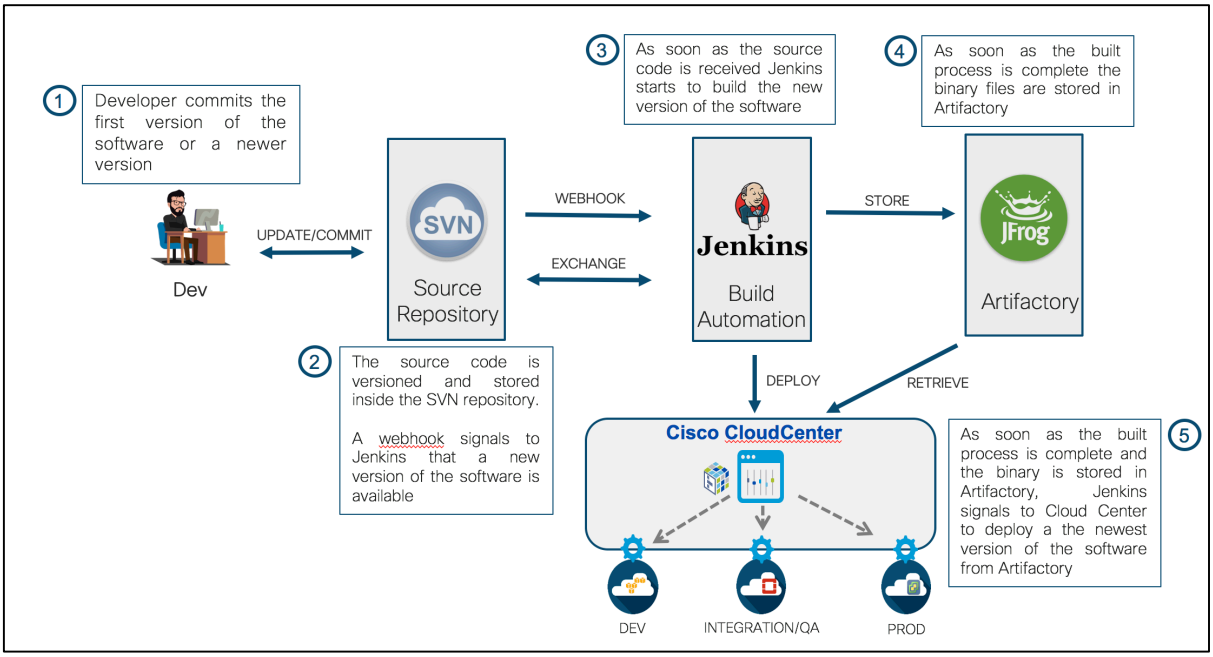


Figure 17: sequence of operations in Scenario 2

For the scope of this demo we will first commit an already developed Java application called PetClinic, follow its automatic build and deployment, and then modify and commit the source code (representing a new application release) to see the procedure happening all over.

3 Preparing the Local Environment

In this section, we prepare the developer environment and we commit a new version of the PetClinic app.

- Download the PetClinic application from this link and extract the folder on the desktop
- If you don't have a client available, download an SVN Client such as Tortoise SVN
- Right click on **SVN Checkout** (Figure 18)
- In "URL of repository" insert the URL of the SVN repository that was deployed on Scenario 1. In our case is http://192.168.130.132/svn/repo1. You will be asked for username and password of the repository (default user001/C1sco123) (Figure 19)

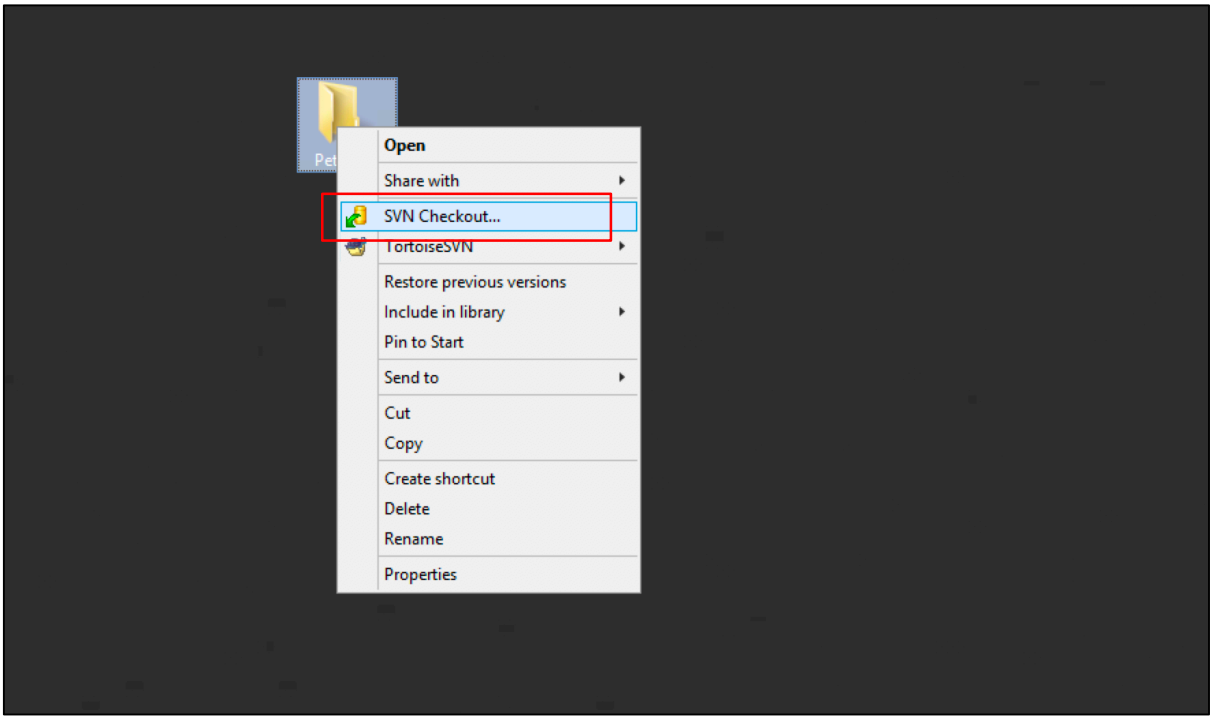


Figure 18: setting up the SVN client (1)

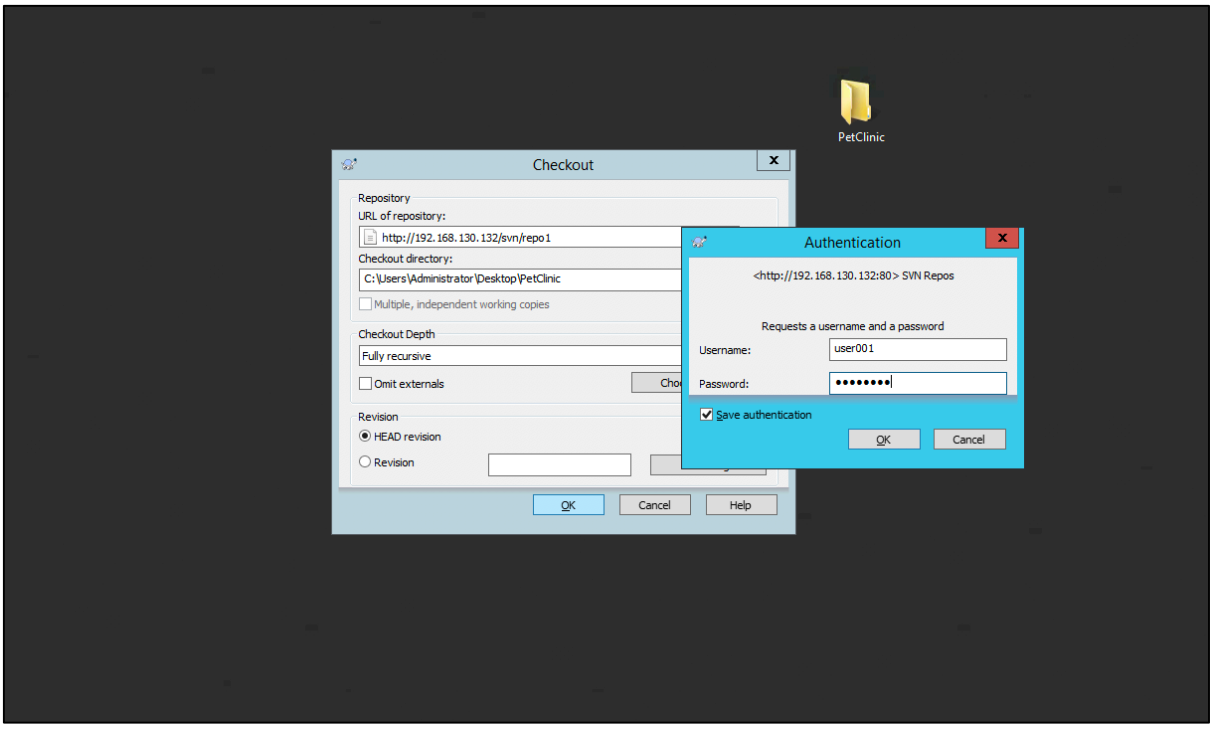


Figure 19: setting up the SVN client (2)

4 First Deployment

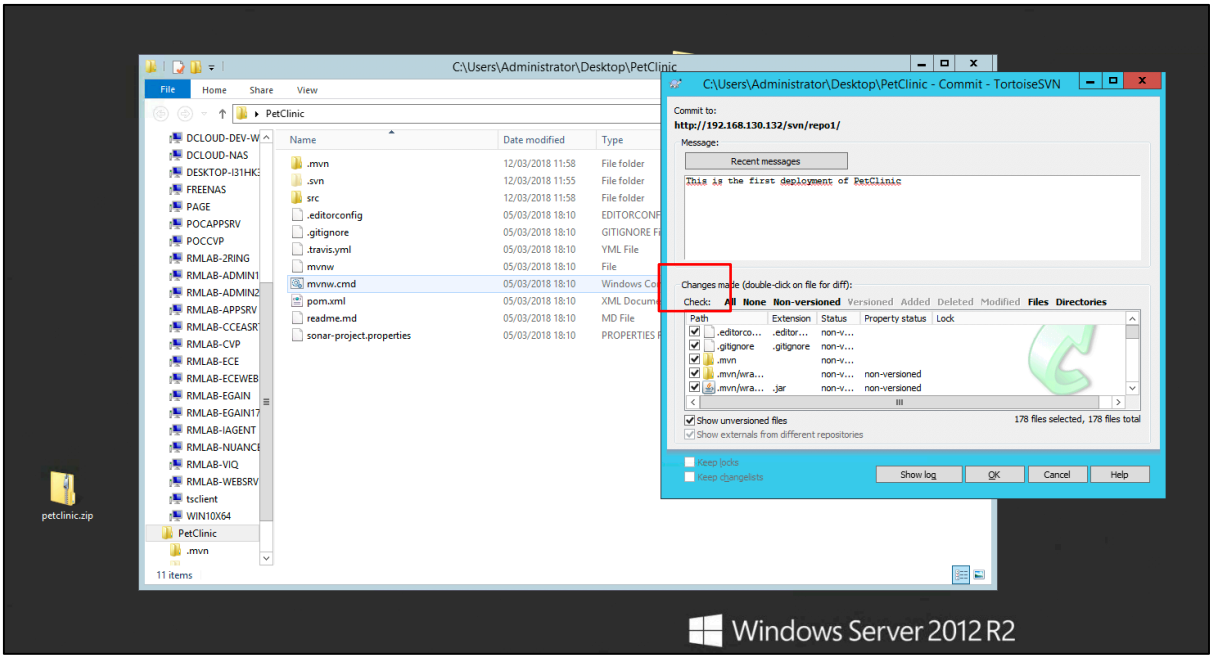


Figure 20: commit to SVN

- Right click inside the folder Petclinic and select SVN Commit, you will be prompted with the possibility to choose which files to commit (Figure 20). Click on **All** and then **Ok**

This is the only manual action that the developer needs to do, since the other actions (2 to 5 on Figure 17) are automatically executed.

- If we access to Jenkins at http://jenkinsIP:8080 (Figure 21) through a web browser and we select the project repo1 we can see that Jenkins (through Maven) is currently creating a new build. As soon as the building process is terminated the binaries are copied into Artifactory and the deploy process starts.

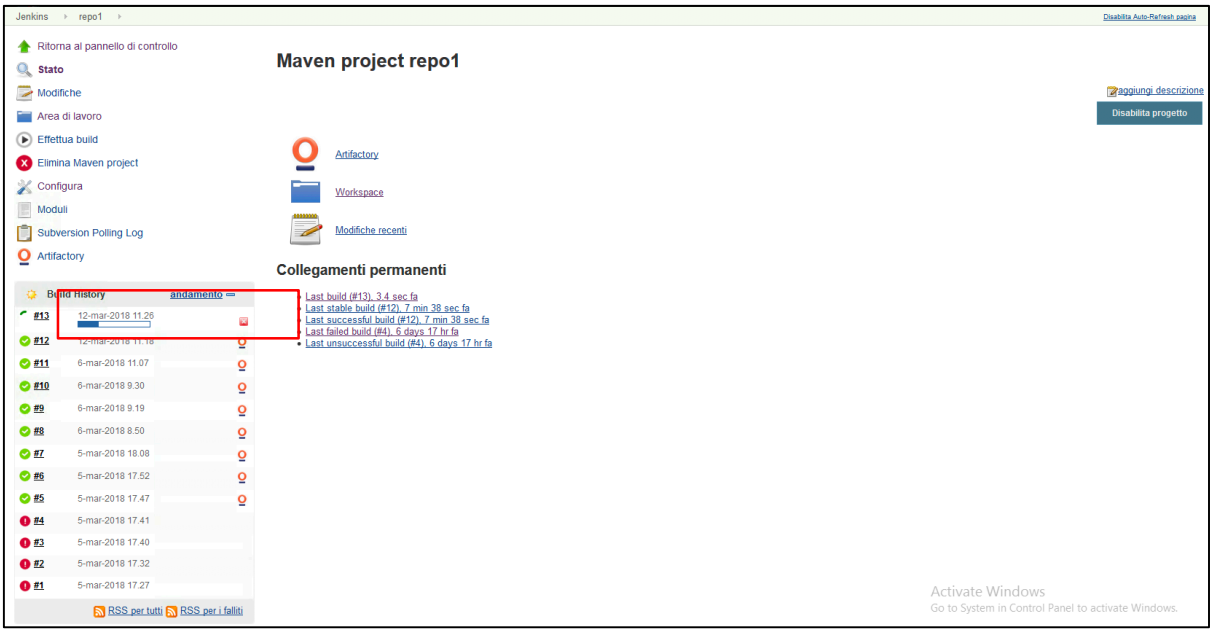


Figure 21: follow the build process through Jenkins

- If we access to the "deploy" job of Jenkins (Figure 22) we can see that a new build of PetClinics has been sent to Cloud Center to deploy. This is possible thanks to the plugin that connects Jenkins with Cloud Center.

Figure 22: deployment of PetClinics

- If anything in the process of the deployment would fail, a red flag would appear and it would be possible to through the logs to see where and why the process failed.

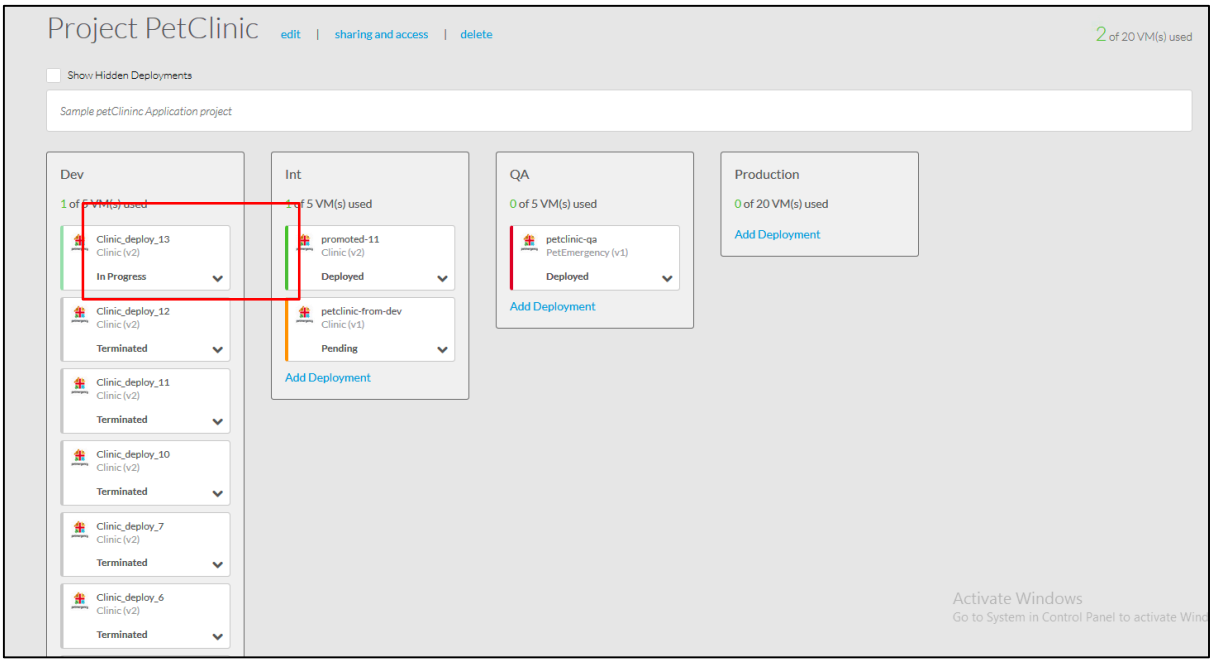


Figure 23: check the deployment in Cloud Center

- I we access to Cloud Center -> Projects -> Project PetClinic (Figure 23) we can see that a new Build of Pet Clinic (Clinic_deploy_13) is being deployed while the previous version (Clinic_deploy_12) has been terminated. Once the deployment is terminated we can click on it and access to a view of the VM that is hosting our application.

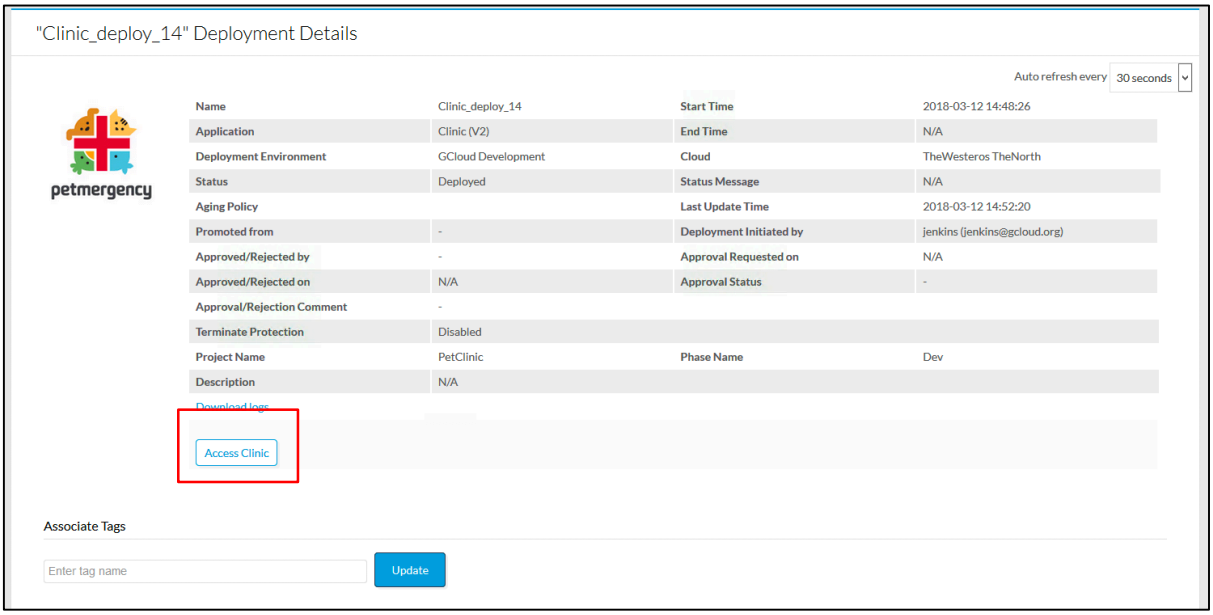


Figure 24: view the deployment details in Cloud Center

- From this view, we can have a detailed view of the VM that is hosting our application as well as accessing to it by clicking on **Access Clinic**.

- PetClinic is a sample java web based application based on the Spring framework (Figure 25)

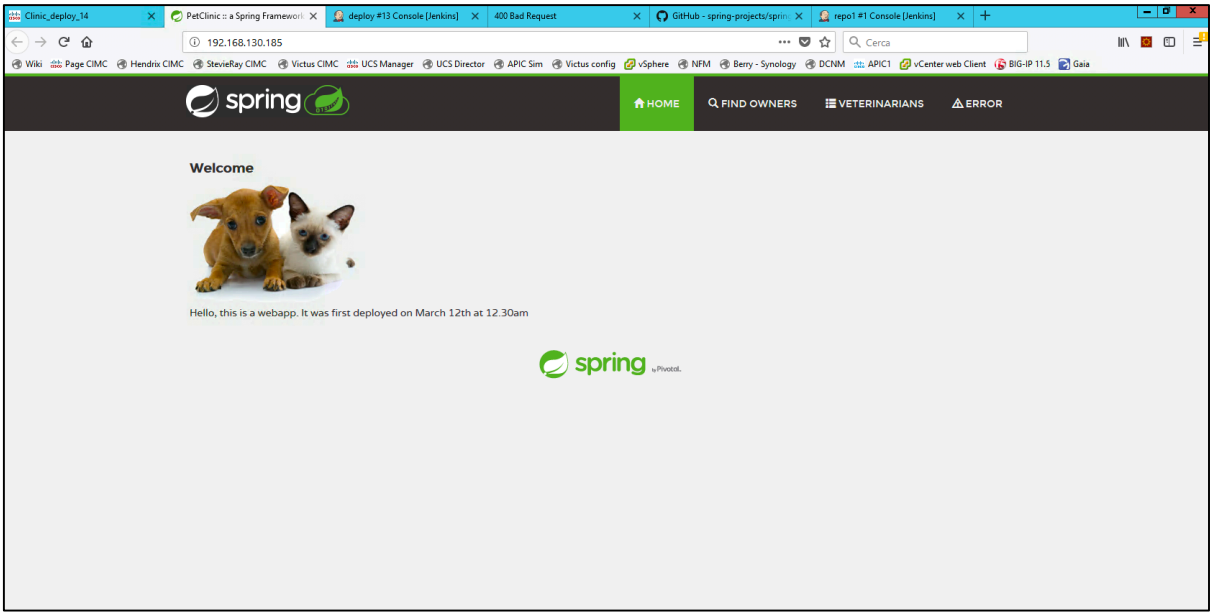


Figure 25: view of the deployed application through a web browser

5 Commit of a new version

Imagine that another developer is working in improving the front end of the application and is ready to commit a major chunk of code. For the sake of the demonstration we will only change the picture and the text on the homepage but we will show that upon committing the modifications to SVN, the new applications is automatically deployed in the Development environment of Cloud Center.

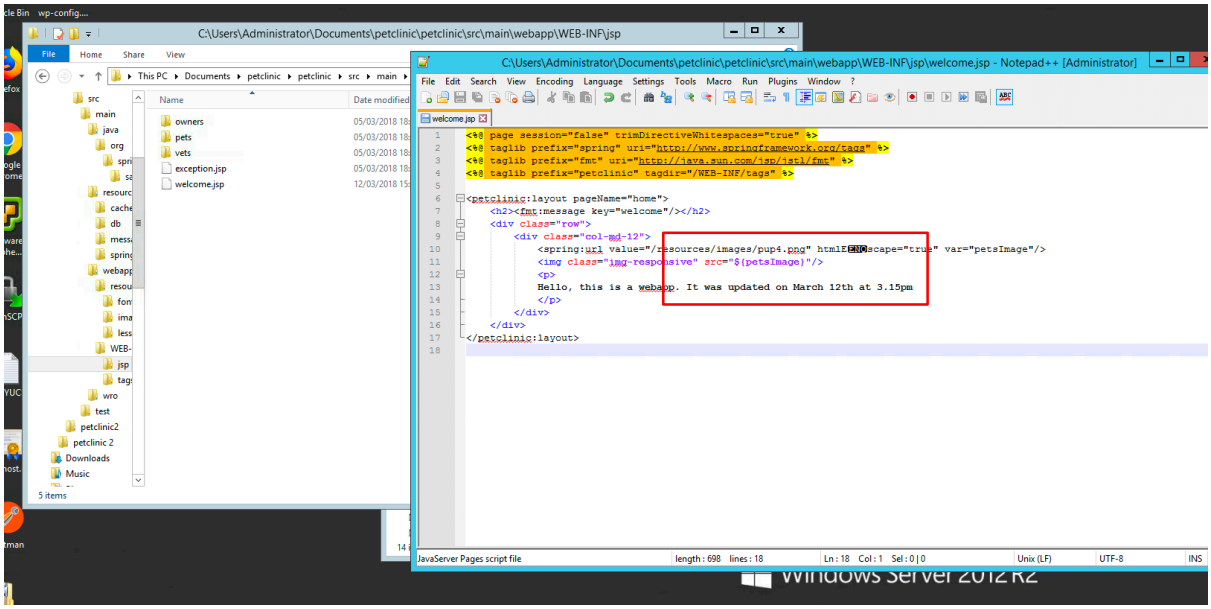
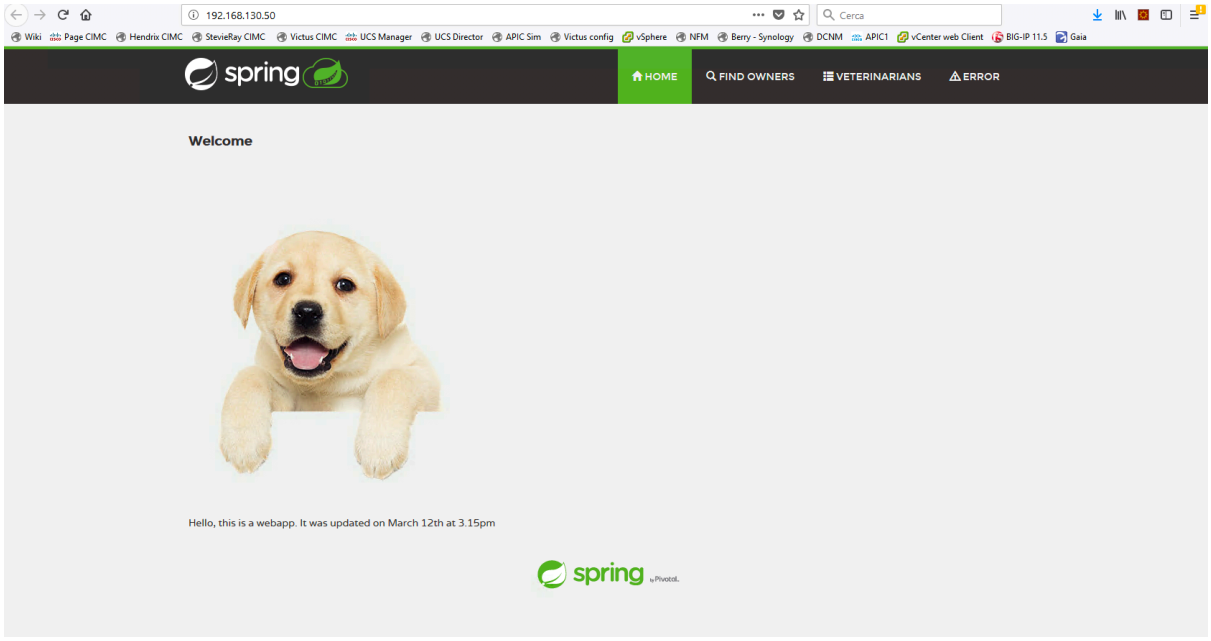


Figure 26: introducing a new version of the code of PetClinic

- For this demonstration, we modify the file petclinic/src/main/webapp/WEB-INF/jsp/welcome.jsp with a text and we change the text and reference picture (Figure 26). Once we are done we save and commit the new version of the file (right click, SVN Commit).



- If we wait a few minutes for the whole chain of operation to finish we will find out a new deployment of the application that we we'll be able to open from Cloud Center -> Projects -> Project PetClinic.

Scenario 3 – promoting a deployment to next stage

In scenario 3 we show how to move the PetClinic Deployment that we have created in the last scenario from the Development phase up to the Integration phase using Cloud Center.

Suppose the integration engineer, after the developers have committed their daily work, want to run some integration tests on the release. To do so, it promotes the last Deployment (Clinic_Deploy_17). On Cloud Center access to Projects -> Project PetClinic (Figure 27), click on the latest deployed version and then **Promote**.

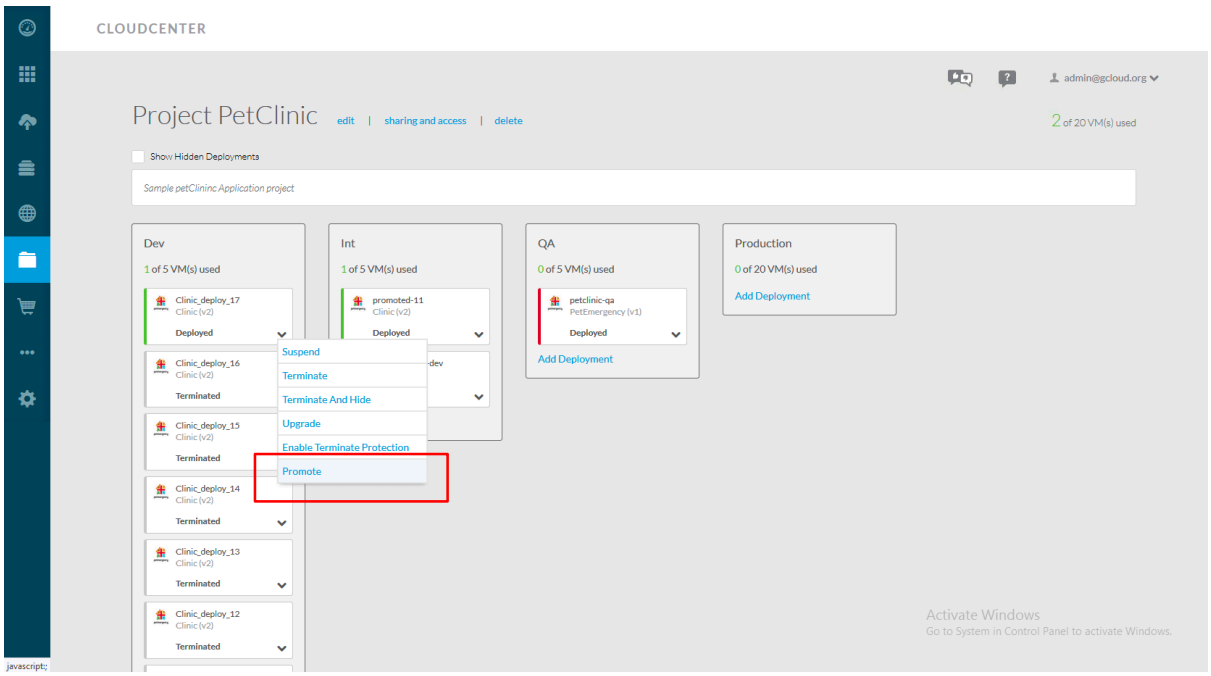


Figure 27: promoting a deployment

- In the next Step we need to select a name for the promoted Deployment as well as indicate the build number that we intend to deploy. In our example, the **build number** is 17 (it will be used by CloudCenter to pull the right release from the Artifactory repository).

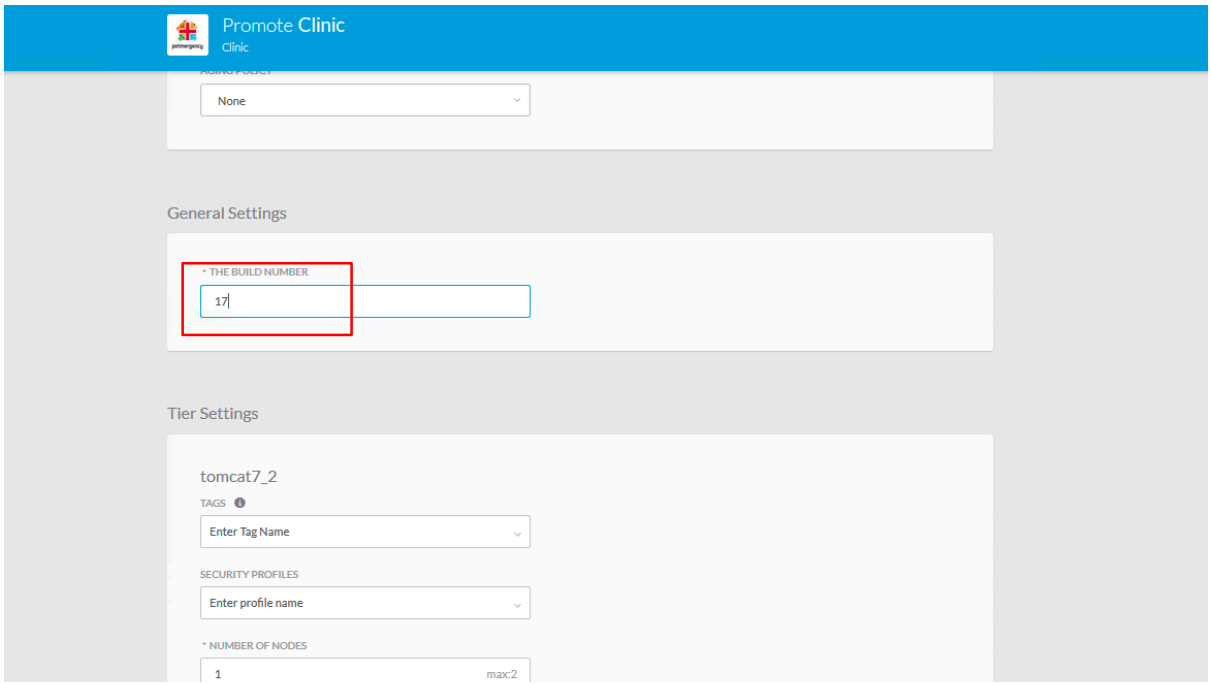


Figure 28: Configuring the deployment for the promotion

- Cloud Center will create a completely new VM, retrieve the build from Artifactory and deploy the application. Since this is a different test

environment compared to the previous one (e.g. stress test instead of functional test), the characteristics of the VM and the environment (e.g. network policies, deployment cluster, etc) are designed to fit this particular need and the new deployment will inherit all those setting from the target environment.

- As a result of the promotion, a new deployment will exist and the old one can either be terminated or remain active.