**Lab 05**

**Laboratory Exercise**

**LAB EXERCISE**

**Time to Complete**

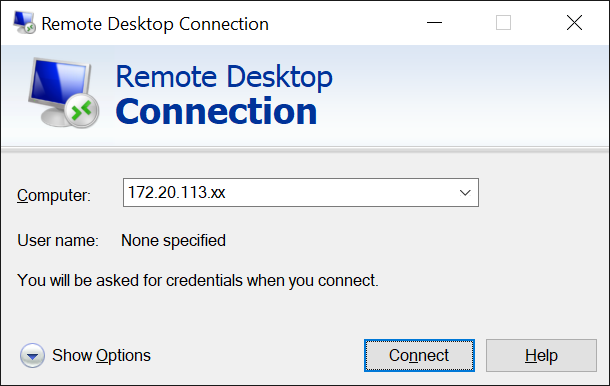
Approximately 60 Minutes

**What You Need in this lab**

* Create a control repository from the Puppet template
* Configure Code Manager
* Use Facter to get real-time information from system

From your machine logged-in to RP VPN, run Remote Desktop Connection to connect to the ubuntu Linux Virtual Machine (VM). Please login based on your assigned VM as shown below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S/N** | **VM** | **IP Address** | **User Name** | **Password** |
| 0 | PDC2-Ubuntu-01 | 172.20.113.182 | dockeradm | docker!2 |
| 1 | PDC2-Ubuntu-02 | 172.20.113.183 | dockeradm | docker!2 |
| 2 | PDC2-Ubuntu-03 | 172.20.113.184 | dockeradm | docker!2 |
| 3 | PDC2-Ubuntu-04 | 172.20.113.185 | dockeradm | docker!2 |
| 4 | PDC2-Ubuntu-05 | 172.20.113.186 | dockeradm | docker!2 |
| 5 | PDC2-Ubuntu-06 | 172.20.113.187 | dockeradm | docker!2 |
| 6 | PDC2-Ubuntu-07 | 172.20.113.188 | dockeradm | docker!2 |
| 7 | PDC2-Ubuntu-08 | 172.20.113.189 | dockeradm | docker!2 |
| 8 | PDC2-Ubuntu-09 | 172.20.113.190 | dockeradm | docker!2 |
| 9 | PDC2-Ubuntu-10 | 172.20.113.191 | dockeradm | docker!2 |
| 10 | PDC2-Ubuntu-11 | 172.20.113.192 | dockeradm | docker!2 |
| 11 | PDC2-Ubuntu-12 | 172.20.113.193 | dockeradm | docker!2 |
| 12 | PDC2-Ubuntu-13 | 172.20.113.194 | dockeradm | docker!2 |
| 13 | PDC2-Ubuntu-14 | 172.20.113.195 | dockeradm | docker!2 |
| 14 | PDC2-Ubuntu-15 | 172.20.113.196 | dockeradm | docker!2 |
| 15 | PDC2-Ubuntu-16 | 172.20.113.197 | dockeradm | docker!2 |
| 16 | PDC2-Ubuntu-17 | 172.20.113.198 | dockeradm | docker!2 |
| 17 | PDC2-Ubuntu-18 | 172.20.113.199 | dockeradm | docker!2 |
| 18 | PDC2-Ubuntu-19 | 172.20.113.200 | dockeradm | docker!2 |
| 19 | PDC2-Ubuntu-20 | 172.20.113.201 | dockeradm | docker!2 |
| 20 | PDC2-Ubuntu-21 | 172.20.113.202 | dockeradm | docker!2 |



Replace xx with the IP address of the VM that you have been assigned.

|  |
| --- |
| **Part 1: Create a control repository from the Puppet template** |

The main objective in this part of the lab is to connect puppet enterprise with your control repository.

Based on the Puppet control repo template, you can construct a control repository (repo) with a standard recommended structure, code examples, and configuration scripts.

**Steps/Hints**:

Set up a private SSH key on your primary server so that it can be recognized by your Git host.

To gain access to the control repository, create a private SSH key.

A password is not required for this SSH key.

1. Perform a backup of your files and subdirectories in the **/etc/puppetlabs/code/environments**.

**# mkdir /home/dockeradm/puppet/backup**

**# cp -R /etc/puppetlabs/code/environments /home/dockeradm/puppet/backup**

**# cd /home/dockeradm/puppet/backup**

**# ls**

|  |
| --- |
| <Insert screen capture of results> |

1. Generate the key pair:

**# su**

**# ssh-keygen -t ed25519 -P '' -f /etc/puppetlabs/puppetserver/ssh/id-control\_repo.ed25519**

|  |
| --- |
| <Insert screen capture of results> |

1. Set appropriate permissions so that the pe-puppet user can access the key:

**# puppet infrastructure configure**

|  |
| --- |
| <Insert screen capture of results> |

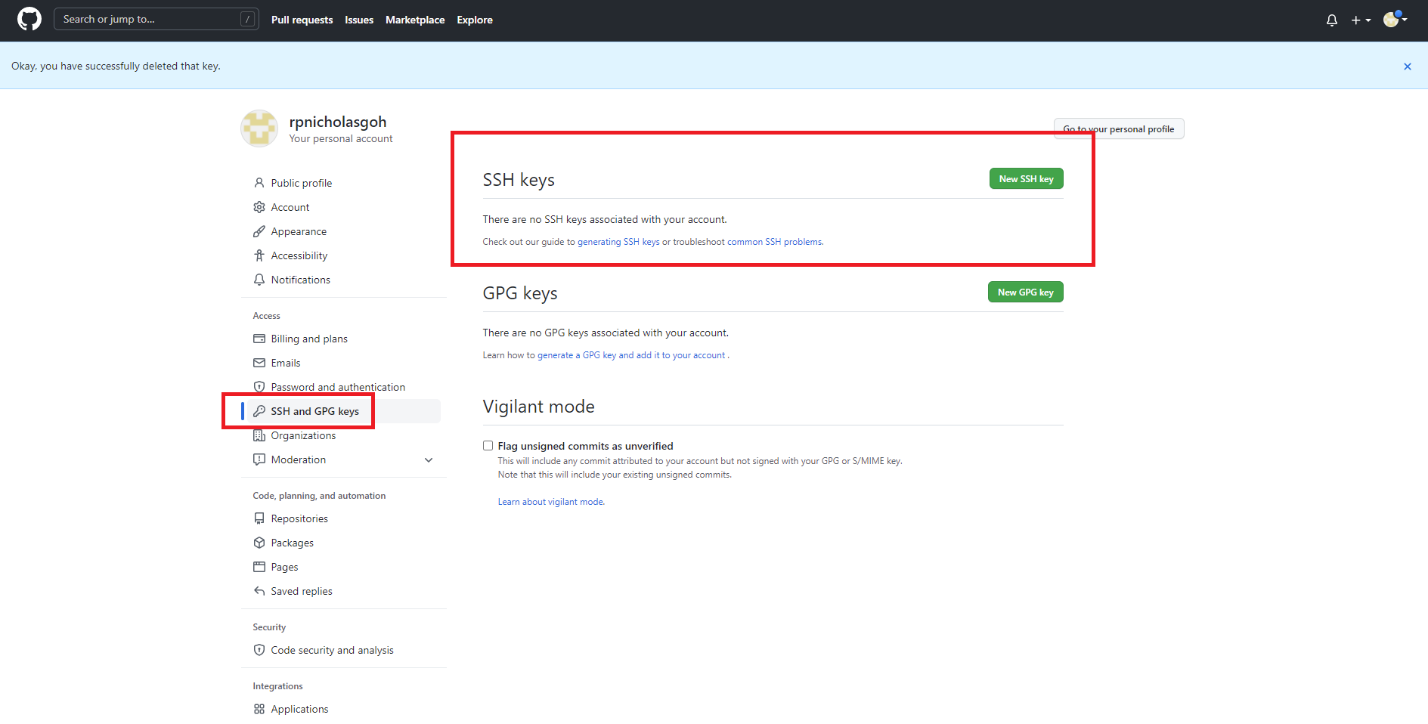
Your keys are now located in **/etc/puppetlabs/puppetserver/ssh**:

Private key: **/etc/puppetlabs/puppetserver/ssh/id-control\_repo.ed25519**

Public key: **/etc/puppetlabs/puppetserver/ssh/id-control\_repo.ed25519.pub**

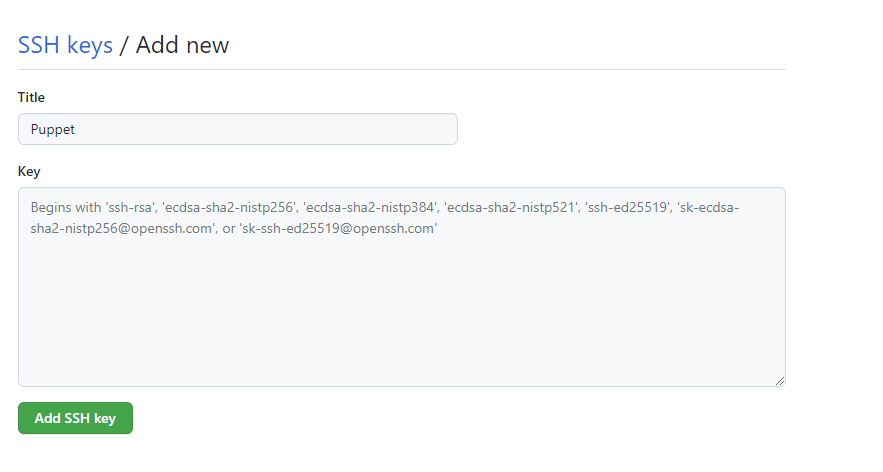
1. Configure your Git host to use the generated SSH public key. Every Git host has its own procedure for doing so. The SSH public key is usually assigned to a user or service account after it is created.

Read access to your control repository, as well as any module repositories referenced in the Puppetfile, is required for code management.

1. Navigate to gitlab.com on your browser.
2. Click on the icon at the upper right corner of the screen and select **Settings**.
3. Click **SSH and GPG Keys** from the menu on the left.  
     
   
4. Click on **New SSH key**.
5. On the Terminal, show the public key.  
   **# cat /etc/puppetlabs/puppetserver/ssh/id-control\_repo.ed25519.pub**

|  |
| --- |
| <Insert screen capture of results> |

1. Copy the printed public key in the previous section and paste it to the Key textarea and click **Add SSH key** button.

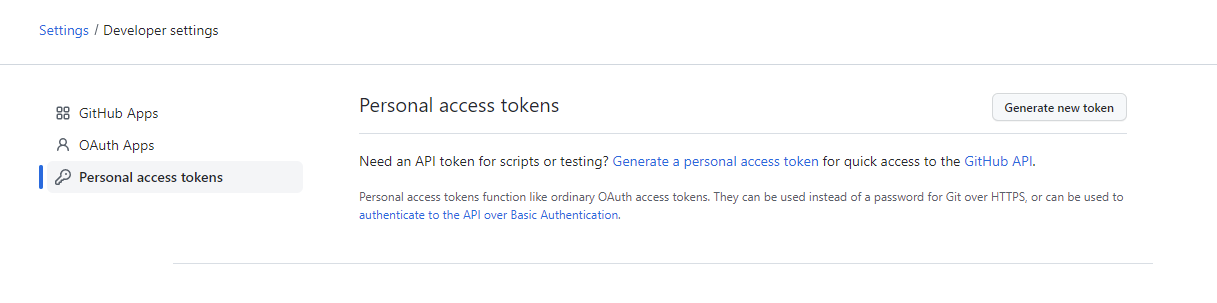


|  |
| --- |
| <Insert screen capture of results> |

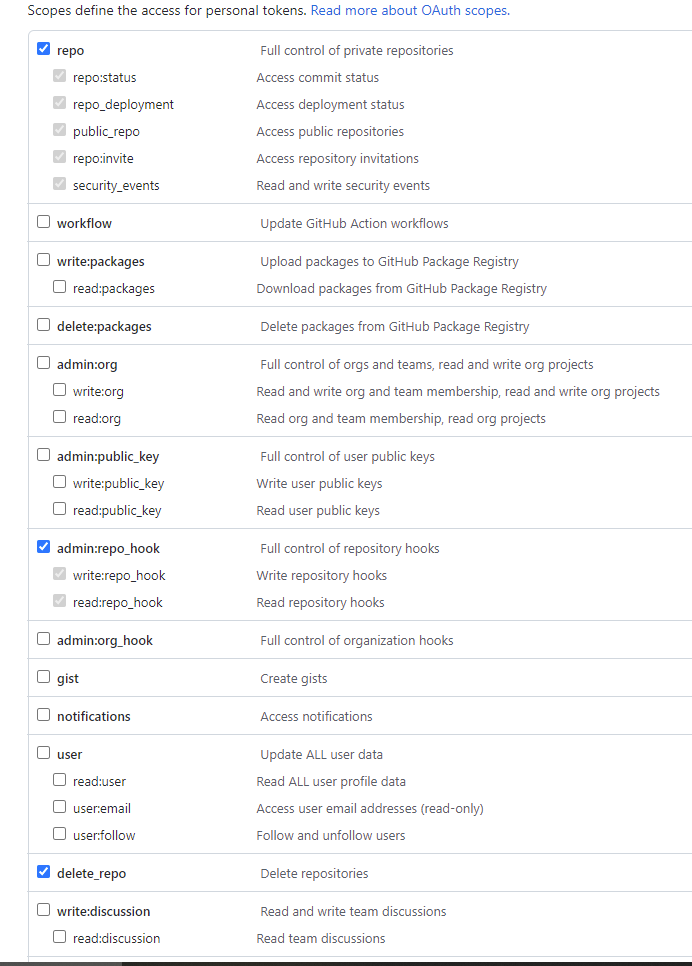
1. Now you are ready to access GitLab.
2. In the upper-right corner of any page, click your profile photo, then click Settings.
3. In the left sidebar, click **Developer settings**.



1. In the left sidebar, click **Personal access tokens**.



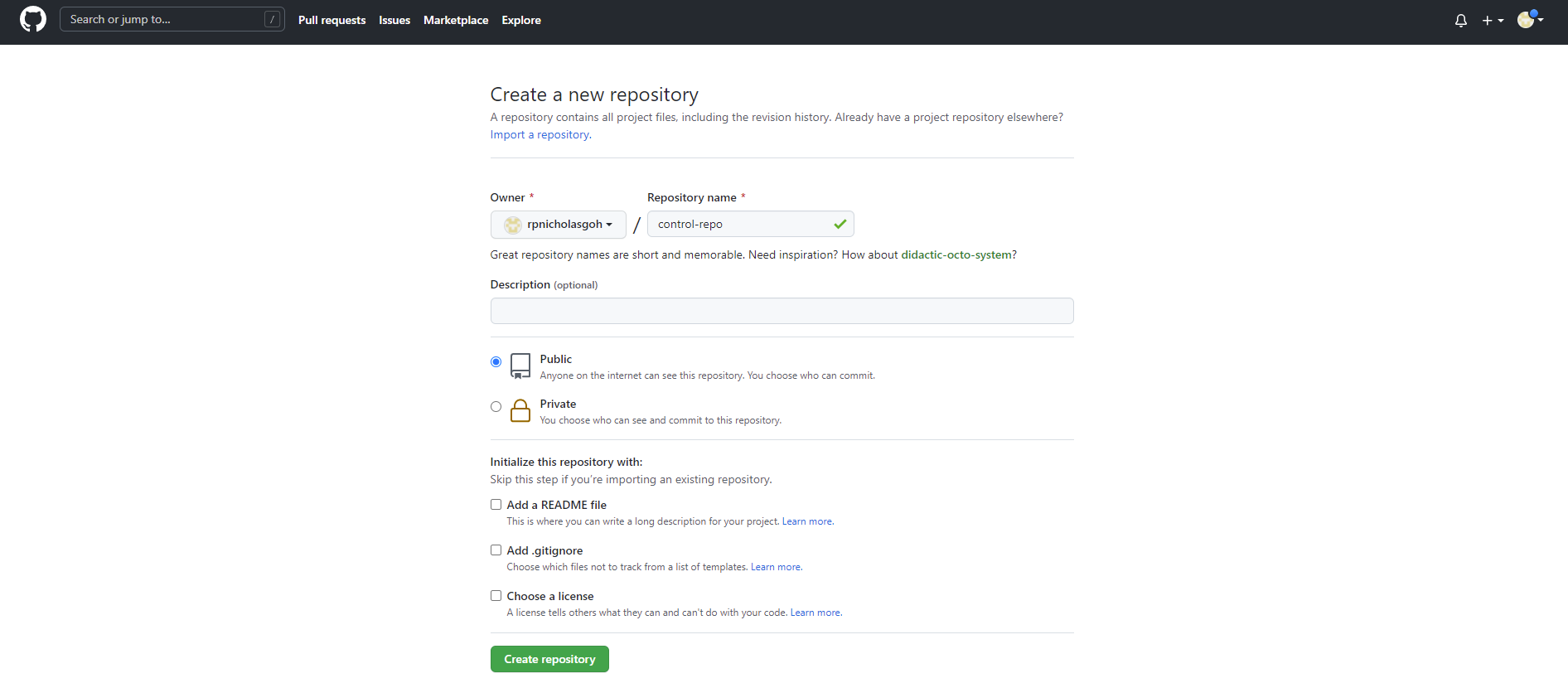
1. Click **Generate new token**.
2. Give your token a descriptive name such as **Puppet**.
3. To give your token an expiration, select the **Expiration** drop-down menu, then click a **default** such as **30 days**.
4. Select the scopes, or permissions, you'd like to grant this token. To use your token to access repositories from the command line, select **repo**.



1. Click **Generate token**.

|  |
| --- |
| <Insert screen capture of results> |

1. Create a repository with the name you desire for your control repo in your Git account.



1. From the command line, clone the Puppet control-repo template.

**# git clone** [**https://github.com/puppetlabs/control-repo.git**](https://github.com/puppetlabs/control-repo.git)

Key in the username and password (personal token)

|  |
| --- |
| <Insert screen capture of results> |

1. Change directory into your contro repo.

**# cd <NAME OF YOUR CONTROL REPO>**

|  |
| --- |
| <Insert screen capture of results> |

1. Remove the template repository as your default source.

**# git remote remove origin**

|  |
| --- |
| <Insert screen capture of results> |

1. Add the control repository you created as the default source.

**# git remote add origin <URL OF YOUR GIT REPOSITORY>**

|  |
| --- |
| <Insert screen capture of results> |

1. Push the contents of the cloned control repo to your remote copy of the control repo:

**# git push origin production**

|  |
| --- |
| <Insert screen capture of results> |

1. Verify that this is set up correctly by using **git remote -v** to list the URL of the remote repo.

**# git remote -v**

|  |
| --- |
| <Insert screen capture of results> |

1. Show the content of your control repo.

**# ls**

|  |
| --- |
| <Insert screen capture of results> |

|  |
| --- |
| **Part 2: Configure Code Manager** |

The main objective in this part of the lab is to configure Code Manager.

When you make changes to your code, Code Manager stages, commits, and synchronizes it, automatically managing your environments and modules.

**Steps/Hints**:

1. Login to the console (web interface) of Puppet Enterprise on primary server using web browser.

**http://localhost**

1. In the console, click **Node groups**, set the following parameters in the **puppet\_enterprise::profile::master class** in the PE Master node group.

|  |  |
| --- | --- |
| code\_manager\_auto\_configure | **true** |
| r10k\_remote | (Location of your control repository) For example, **https://github.com/XXXXXXX/control-repo.git** |
| r10k\_private\_key | (The path to the SSH private key that permits the pe-puppet user to access your Git repositories)  **/etc/puppetlabs/puppetserver/ssh/id-control\_repo.ed25519** |

1. Click **Commit**.

|  |
| --- |
| <Insert screen capture of results> |

1. Test the configuration on primary server.

**# puppet agent -tv**

|  |
| --- |
| <Insert screen capture of results> |

|  |
| --- |
| *Note: r10k is a code management tool that allows you to manage your environment configurations — production, testing, and development — in a source control repository.* |

1. Request an authentication token for deployments from the command line of the primary server.

**# puppet-access login --lifetime 180d**

|  |
| --- |
| <Insert screen capture of results> |

|  |
| --- |
| *Note: Authentication tokens have a one-hour lifespan by default. With the Override default expiration permission enabled, you can adjust the token's lifetime to something more suitable for a long-running, automated procedure.* |

1. Run the following command to test whether Code Manager can connect to the control-repo under a single environment (production).

**# puppet-code deploy production --dry-run --wait**

|  |
| --- |
| <Insert screen capture of results> |

|  |
| --- |
| *Note: You must deploy your code before Code Manager can detect or manage the content when you make changes to your Puppetfile, such as adding a new module or creating a repo.*  ***puppet-code deploy --all --wait*** *deploy code to every possible environment.*  *You can also use a webhook or custom scripts to deliver code.* |

|  |
| --- |
| **Part 3: Facter** |

The main objective in this part of the lab is to use facter to get real-time information from system via the command line.

Facter is a different program that Puppet relies on. It's Puppet's technique for gathering information about the target system (node); facter refers to the nuggets of data as facts. You may get real-time information from the system by running facter.

**Steps/Hints**:

1. Use facter to print all the facts known about the system.

**# facter**

|  |
| --- |
| <Insert screen capture of results> |

1. Use facter to determine the system's current uptime using the uptime fact.

**# facter uptime 0:12 hours**

|  |
| --- |
| <Insert screen capture of results> |

1. Compare this to the result of the uptime command on Linux.

**# uptime**

|  |
| --- |
| <Insert screen capture of results> |

1. Try out the other fact definitions.

**# facter ipaddress**

|  |
| --- |
| <Insert screen capture of results> |

**# facter osfamily**

|  |
| --- |
| <Insert screen capture of results> |

**# facter os**

|  |
| --- |
| <Insert screen capture of results> |

**# facter os.release.full**

|  |
| --- |
| <Insert screen capture of results> |

**References**<https://hayato-iriumi.net/2019/07/08/how-to-integrate-puppet-master-with-git/>

Puppet 5 Cookbook - Fourth Edition by Thomas Uphill

**End of Lab Exercise --**