**Lab 02**

**Laboratory Exercise**

**LAB EXERCISE**

**Time to Complete**

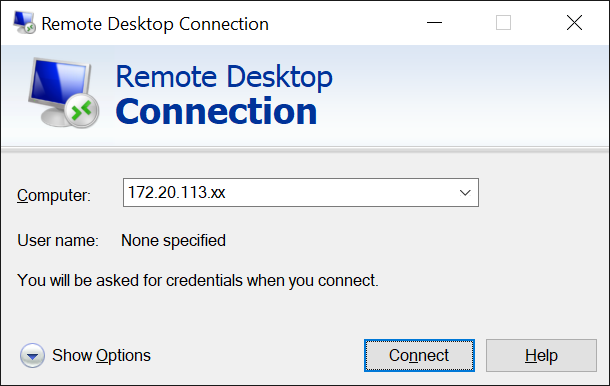
Approximately 60 Minutes

**What You Need**

* In this lab you will learn about Puppet Resources and Puppet Classes

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** |
| 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: Puppet Resources (File)** |

The main objective in this part of the lab is to create a file on the server using Puppet resource type **File**.

The basic unit for modeling system configurations is **resources**.

Each resource specifies the desired state of a system component, such as a specific **user, file,** **service** or **package**.

Puppet maintains every resource in the **catalog** as it applies it to the target system, ensuring that the actual state matches the planned state.

1. Recall from **lab 1** that we had learnt how to use puppet resource subcommand to display the **49** **built-in** puppet resources and used the **user** resource type to manage users.

**# puppet resource --types |more**

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

**Resource Declaration** adds a resource to the catalog and tells Puppet to manage the resource's state.

|  |
| --- |
| **resource\_type { 'name':**  **attribute => value,**  **}** |

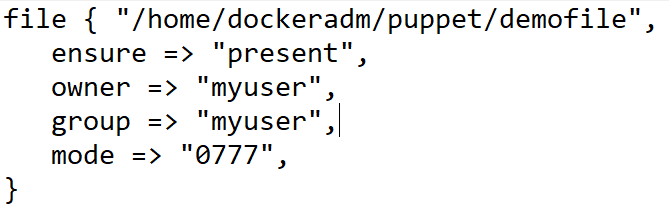
**Steps/Hints**:

1. Create a new **demofile.pp** file under the previously created directory log on using dockeradm account (**e.g. /home/dockeradm/puppet/manifest/**).

# **cd /home/dockeradm/puppet/manifest**

# **nano demofile.pp**

Add the following codes.



1. Validate whether the code has any syntax errors:

**# puppet parser validate demofile.pp**

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

1. Switch to the **root** to be able to complete the test without any error:

**# su root**

1. **Test** the code creation process. Execute to perform a smoke test:

**# puppet apply demofile.pp –noop**

1. **Run** the puppet in real mode and verify the output:

**# puppet apply demofile.pp**

**Question:** What happens after executing this command?

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

1. Ensure the file created is correct.

**# ls -lrt /home/dockeradm/puppet/demofile**

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

|  |
| --- |
| **Do it yourself** |

1. Add the following attribute and value to **demofile.pp**.

|  |  |
| --- | --- |
| **Attribute** | **Value** |
| content | “Welcome to this course DV1C04 Deployt & Monitoring in DevOps!\n” |

|  |
| --- |
| <Insert screen capture of the content of updated demofile.pp> |

1. Verify the content of the **demofile**.

# **cat** **/home/dockeradm/puppet/demofile**

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

|  |
| --- |
| **Part 2: Puppet Resources (Package)** |

The main objective in this part of the lab is to make sure the **Telnet** package is present on the server using Puppet resource type **Package**.

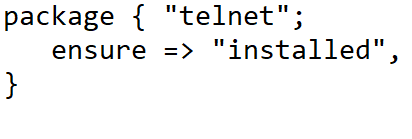
**Steps/Hints**:

1. Ensure that the telnet package is not already present on the server.  
   **# rpm -qa | grep -i telnet**

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

1. Write codes to install telnet package using Puppet DSL.

**# nano demopkg.pp**

****

1. Verify the written codes.

**# cat demopkg.pp**

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

1. Use puppet describe package to find the attributes to be used in the Puppet codes.

**# puppet describe package |more**

**Question:** What other values you can use with the attribute **ensure** Puppet resource type **Package**?  
  
**Solution:**

|  |
| --- |
|  |

1. Check, test and run to install the **Telnet** package.

**# puppet parser validate demopkg.pp**

**# puppet apply demopkg.pp**

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

1. verify the package is installed after the Puppet run.

**# rpm -qa | grep -i telnet**

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

**# rpm -qa telnet-x  
x**: follow what is shown in the earlier command.

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

|  |
| --- |
| **Part 3: Puppet Resources (Service)** |

The main objective in this part of the lab is to start **NFS** service on the server using Puppet Resource **Service**.

Let us continue to look at another important Puppet resource type **Service**.

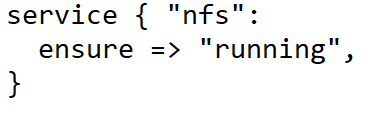
**Steps/Hints**:

1. Ensure that the nfs service is not already started on the server.  
   **# systemctl status nfs**

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

1. Write codes to install telnet package using Puppet DSL.

**# nano demoservice.pp**



1. Verify the written codes.  
   # **cat demoservice.pp**

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

**Question:** What other values you can use with the attribute **ensure** for Puppet resource type **Service**?  
  
**Solution:**

|  |
| --- |
|  |

1. Check, test and run to start the **NFS** service.

**# puppet parser validate demoservice.pp**

**# puppet apply demoservice.pp**

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

1. Ensure that the nfs service is now started on the server by the Puppet run.  
   **# systemctl status nfs**

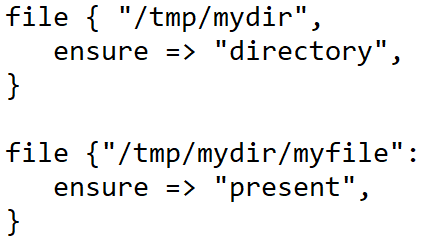
|  |
| --- |
| <Insert screen capture of results> |

|  |
| --- |
| **Part 4: Puppet Resources (Multi-resource)** |

The main objective in this part of the lab is to create a puppet run using multi-resource.

|  |
| --- |
| **Do it yourself** |

1. Create a new **demomulti.pp** file under the previously created directory log on using dockeradm account (**e.g. /home/dockeradm/puppet/manifest/**).



1. Check, test and run the **demomulti.pp**.

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

**Question:** What is the command that you can use to verify that the directory **testdir** is created successfully after the Puppet codes run?

**Solution:**

|  |
| --- |
|  |

**Question:** What is the command that you can use to verify that the file **testfile** is created successfully after the Puppet codes run?

**Solution:**

|  |
| --- |
|  |

|  |
| --- |
| **Part 5: Puppet Classes** |

The main objective in this part of the lab is to make sure the **NTP** package is present on the server using Puppet classes.

**Puppet classes** are the **collection of puppet resources** bundled together as a single unit.

Puppet introduced classes to make the structure re-usable and organized.

First, we need to define a class using class definition syntax; classes must be unique and can be declared only once with the same name:

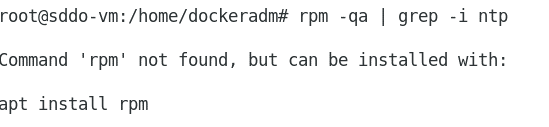
class <class-name> {

<Resource declarations>

}

**Steps/Hints**:

1. The following command will return nothing if the **ntp** is not present on the server:

**# rpm -qa | grep -i ntp**

1. Create a new **demontp.pp** file under the previously created directory log on using dockeradm account (**e.g. /home/dockeradm/puppet/manifest/**).

# **cd /home/dockeradm/puppet/manifest**

# **nano demontp.pp**

Add the following codes.



1. Validate whether the code has any syntax errors:

# **puppet parser validate demontp.pp**

1. Switch to the **root** to be able to complete the test without any error:

# **su root**

1. **Test** the code creation process. Execute to perform a smoke test:

**# puppet apply demontp.pp –noop**

1. **Run** the puppet in real mode and verify the output:

**# puppet apply demontp.pp**

**Question:** What happens after executing this command?

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

**Solution:**

Puppet didn’t perform anything because the demo class was just defined but not declared.  
So, until you declare the puppet class, the code will not get applied.

1. **Declare** the demo class inside the same code using **include class name** at the end of the code:

# **nano demontp.pp**



1. Validate whether the code has any syntax errors:

**# puppet parser validate demontp.pp**

1. Switch to the **root** to be able to complete the test without any error:

**# su root**

1. **Test** the code creation process. Execute to perform a smoke test:

**# puppet apply demontp.pp –noop**

1. **Run** the puppet in real mode and verify the output:

**# puppet apply demontp.pp**

**Question:** What happens after executing this command?

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

**Solution:**

This time the code gets applied because the class was defined and then declared.

1. Ensure that **ntp.conf** is now existing:

**# ls -lrt /etc/ntp.conf**

1. Verify the ntp service has been started by running the following command:

**# systemctl status ntp**

|  |
| --- |
| **Do it yourself** |

1. Create a new **webserver.pp** file under the previously created directory log on using dockeradm account (e.g. **/home/dockeradm/puppet/manifest/)** with a class named **webserver** to ensure that the latest version of the **apache2** package is installed, and checks that the service is running on the web servers managed by Puppet.

|  |
| --- |
| <Insert screen capture of the webserver.pp> |

**References**

* Puppet for Absolute Beginners - Hands-on by Mumshad Mannambeth, Yogesh Raheja

**--End of Lab Exercise --**