**Lab 01**

**Laboratory Exercise**

**LAB EXERCISE**

**Time to Complete**

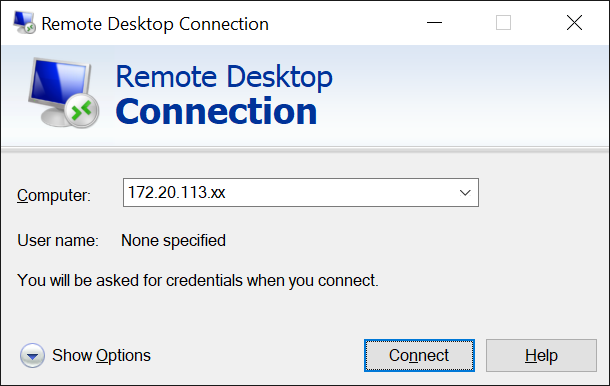
Approximately 45 Minutes

**What You Need**

* Be comfortable with basic Linux administration

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 | 172.20.113.136 | dockeradm | docker!2 |
| 2 | PDC2-Ubuntu-03 | 172.20.113.137 | dockeradm | docker!2 |
| 3 | PDC2-Ubuntu-04 | 172.20.113.138 | dockeradm | docker!2 |
| 4 | PDC2-Ubuntu-05 | 172.20.113.139 | dockeradm | docker!2 |
| 5 | PDC2-Ubuntu-06 | 172.20.113.140 | dockeradm | docker!2 |
| 6 | PDC2-Ubuntu-07 | 172.20.113.141 | dockeradm | docker!2 |
| 7 | PDC2-Ubuntu-08 | 172.20.113.142 | dockeradm | docker!2 |
| 8 | PDC2-Ubuntu-09 | 172.20.113.143 | dockeradm | docker!2 |
| 9 | PDC2-Ubuntu-10 | 172.20.113.144 | dockeradm | docker!2 |
| 10 | PDC2-Ubuntu-11 | 172.20.113.145 | dockeradm | docker!2 |
| 11 | PDC2-Ubuntu-12 | 172.20.113.146 | dockeradm | docker!2 |
| 12 | PDC2-Ubuntu-13 | 172.20.113.147 | dockeradm | docker!2 |
| 13 | PDC2-Ubuntu-14 | 172.20.113.148 | dockeradm | docker!2 |
| 14 | PDC2-Ubuntu-15 | 172.20.113.149 | dockeradm | docker!2 |
| 15 | PDC2-Ubuntu-16 | 172.20.113.150 | dockeradm | docker!2 |
| 16 | PDC2-Ubuntu-17 | 172.20.113.151 | dockeradm | docker!2 |
| 17 | PDC2-Ubuntu-18 | 172.20.113.152 | dockeradm | docker!2 |
| 18 | PDC2-Ubuntu-19 | 172.20.113.153 | dockeradm | docker!2 |
| 19 | PDC2-Ubuntu-20 | 172.20.113.154 | dockeradm | docker!2 |
| 20 | PDC2-Ubuntu-21 | 172.20.113.155 | dockeradm | docker!2 |
| 21 | PDC2-Ubuntu-22 | 172.20.113.156 | dockeradm | docker!2 |

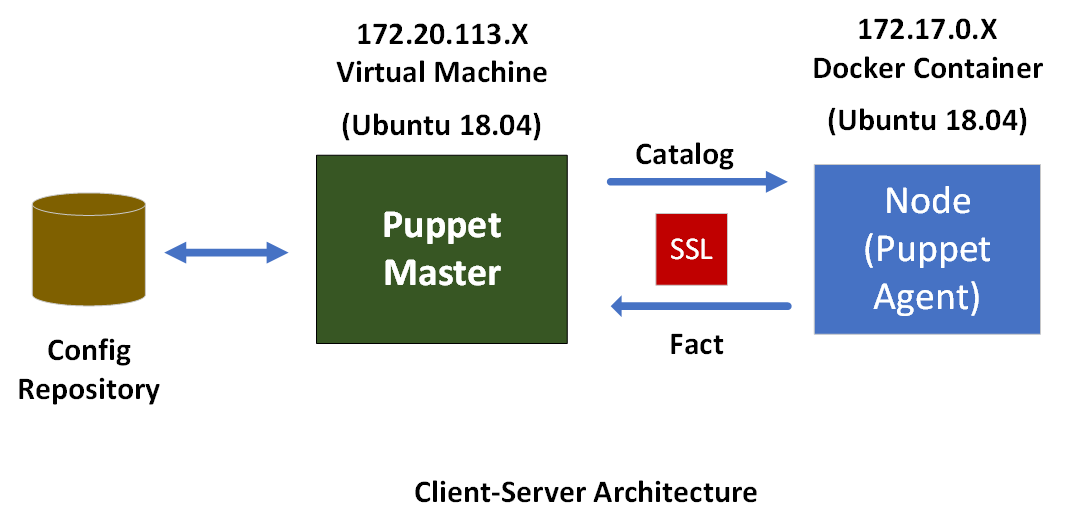


Replace xx with the IP address of the VM that you have been assigned.

**Part 1: Installation**

1. In this lab you install the Puppet Enterprise and Setup Puppet Agent (container) provided by the Puppet Collection

* Install the Puppet agent provided by the Puppet Collection
* Make Puppet Collection available by installing the release repository



1.1As **root,** edit **/etc/hosts** and **/etc/hostname**, put in **sddo-vm.localdomain**.

Note: A hosts file is a plain-text file used by the operating systems to map IP addresses to hostname on the local computer. The hosts file has priority over DNS.

Open **Terminal** and enter the following command to login as a superuser.

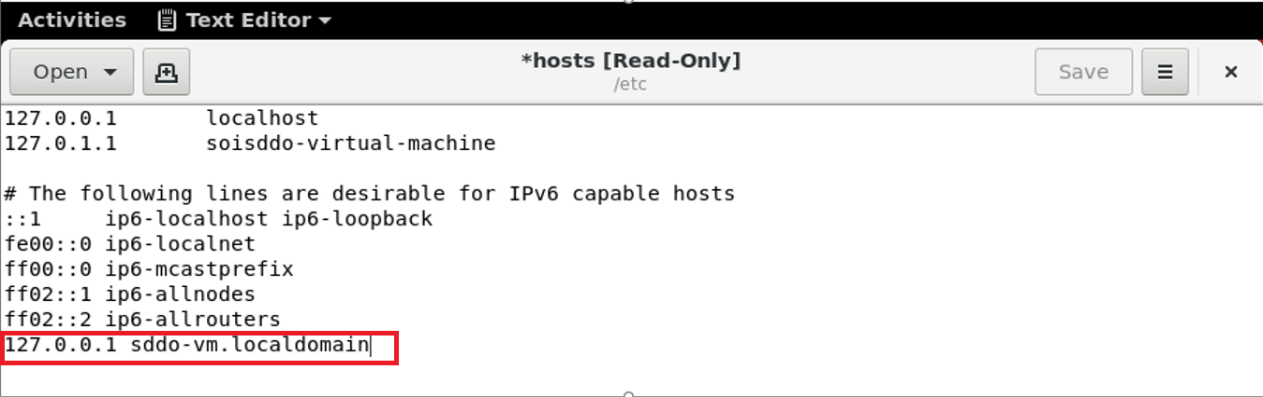
**# su**



\*password: **ubuntu**To add an entry or modify the hosts file, simply run the commands below to open the hosts file.

**# sudo nano /etc/hosts**

Make the change to the hosts file as per below.

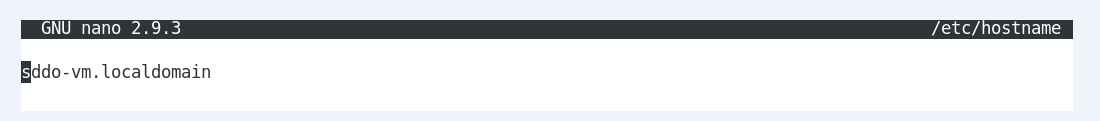


Save the file and exit. Changes you make to the hosts file take effect immediately.  
  
To add an entry or modify the hostname file, simply run the commands below to open the hostname file.

**# sudo nano /etc/hostname**



Make the change to the hostname file as per below.

  
Save the file and exit. Changes you make to the hostname file take effect immediately.

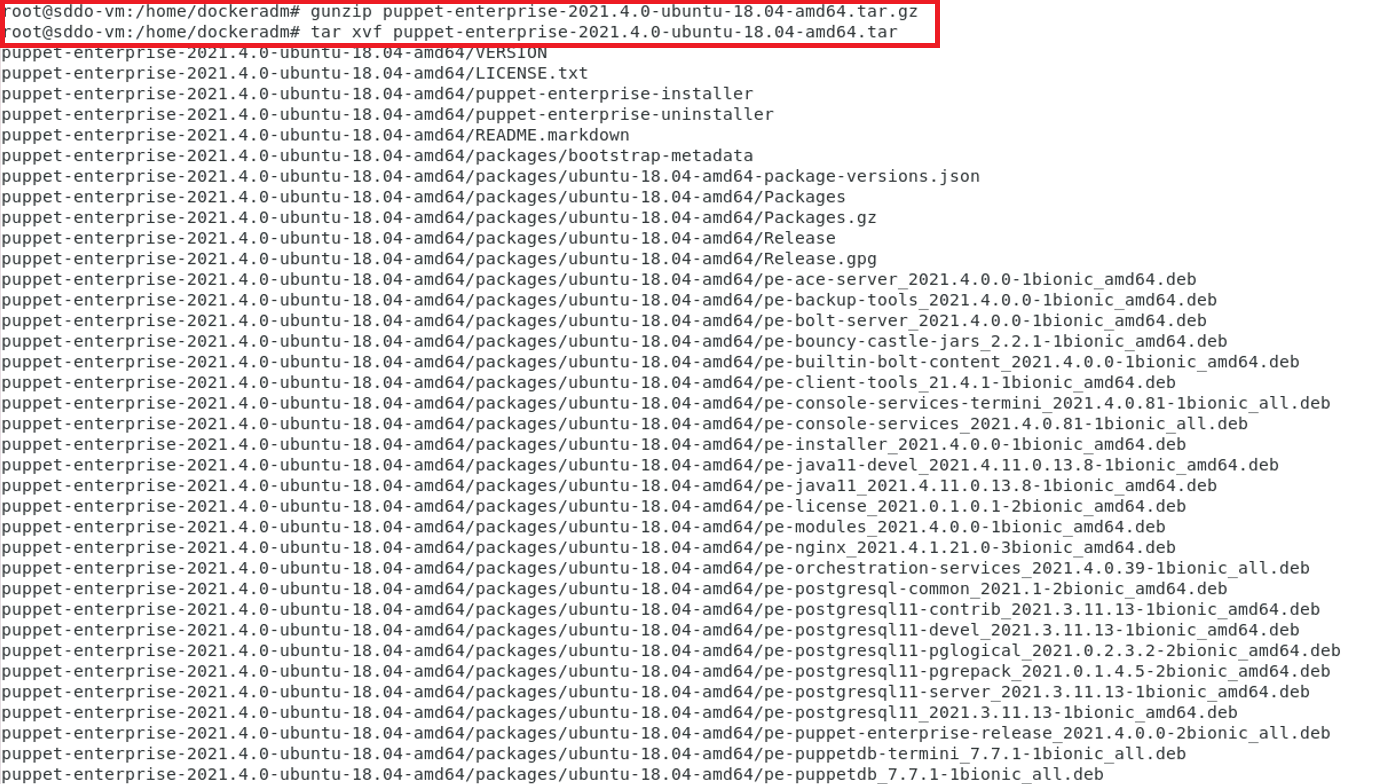
2. Setup Puppet Enterprise and Inspect what version is installed on your master

2.1 Download the tarball appropriate to your operating system and architecture.

# **wget --content-disposition 'https://pm.puppet.com/cgi-bin/download.cgi?dist=ubuntu&rel=18.04&arch=amd64&ver=latest'**  
  
  
  
2.2 Unpack the installation tarball:

# **gunzip <the download file>**

# **tar xvf <the download file>**



2.3 Customise pe.conf file   
  
A **pe.conf** file is a HOCON formatted file that declares parameters and values used to install, upgrade, or configure PE. A **default pe.conf** file is available in the **conf.d** directory in the installer tarball.

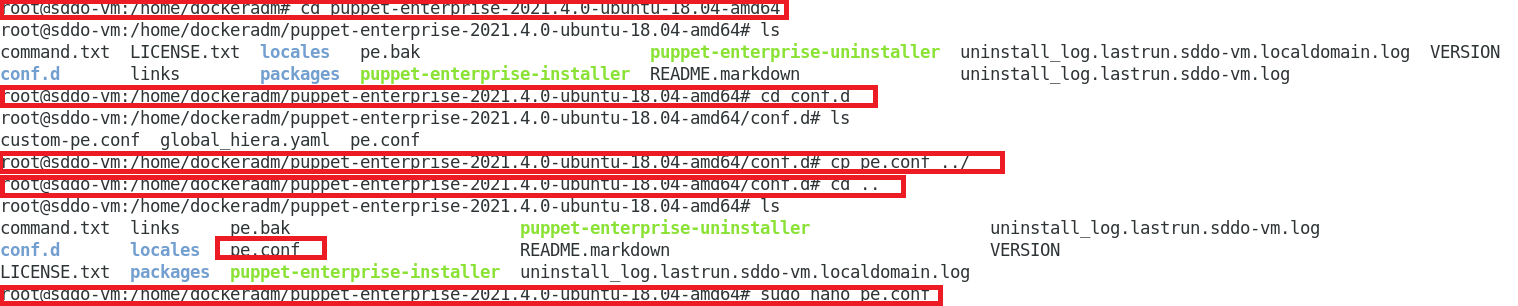
From the installer directory, create a copy of the pe.conf from the conf.d directory.

**# cd** **<installer directory>**

**# cd conf.d**

**# cp pe.conf ../**  (copy pe.conf file to the installer directory)

# **cd ..**  (return to installer directory)



Make the change of the pe.conf file to set the console administrator password:  
  
**# sudo nano pe.conf**

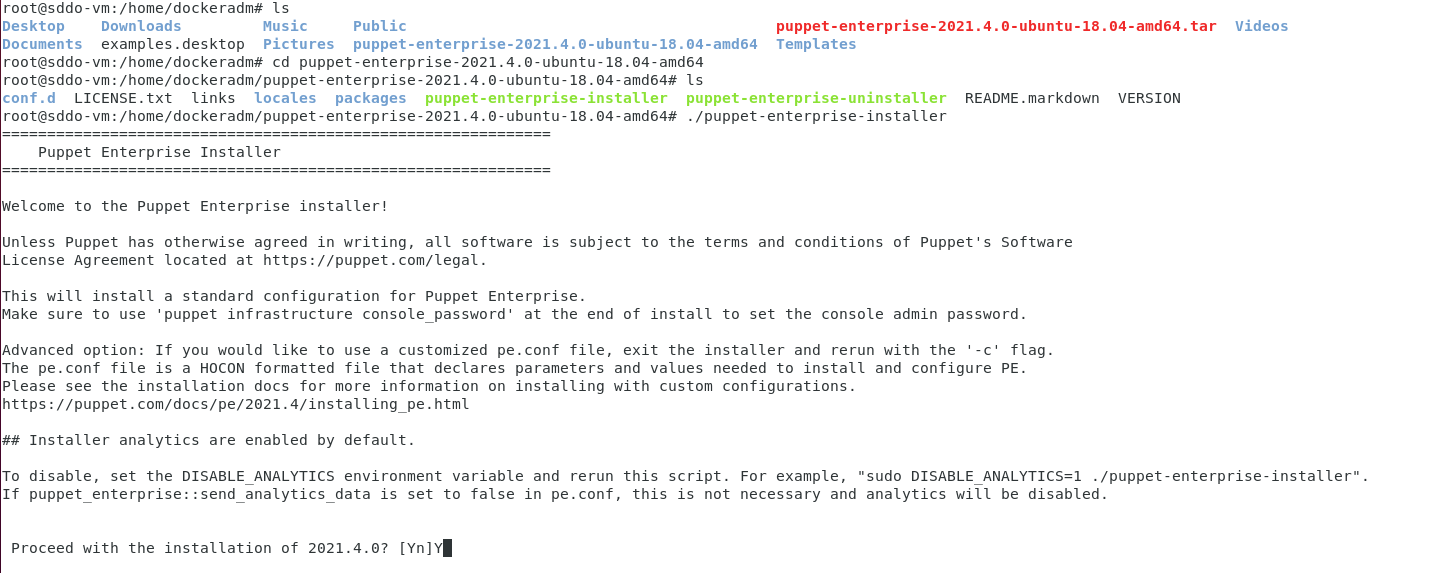


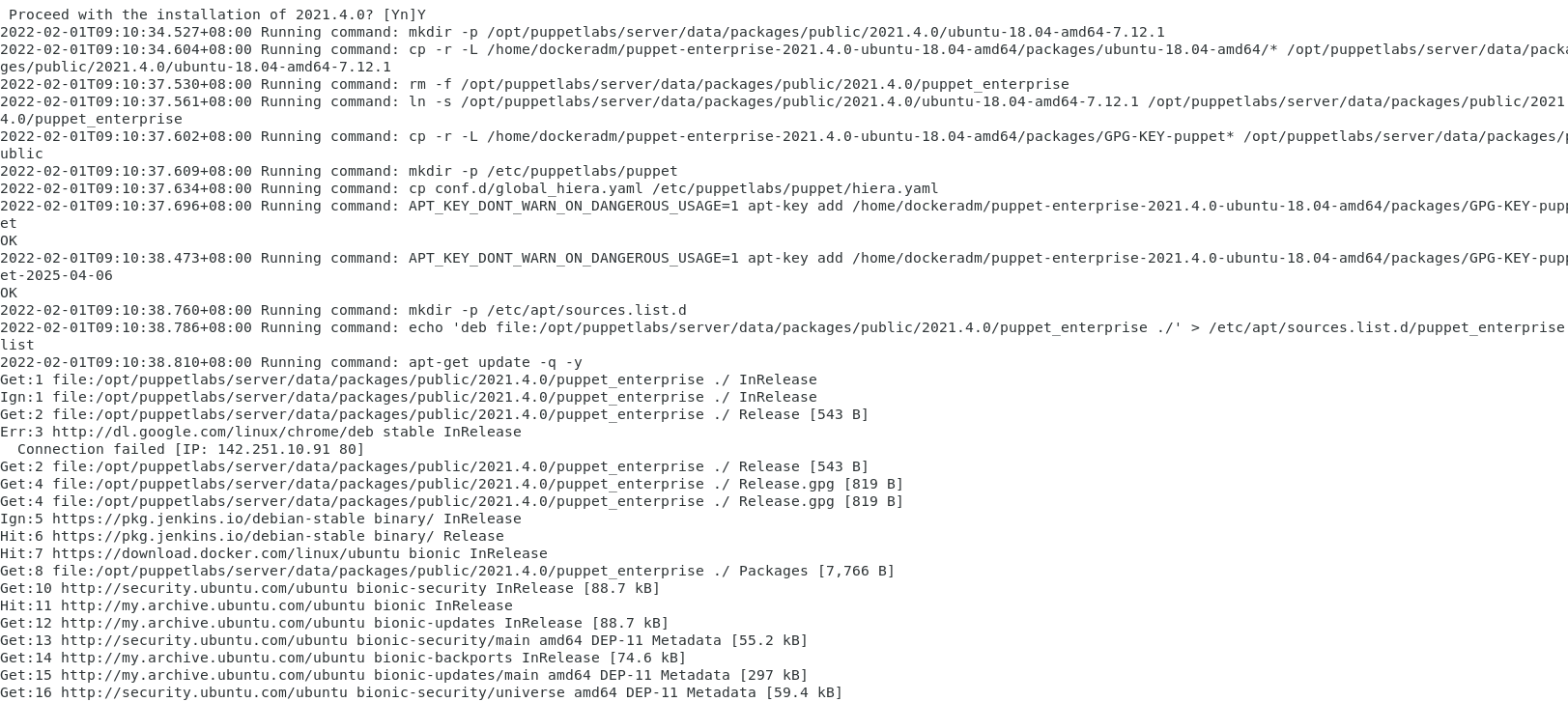
Save and exit the file.

From the installer directory:

# **puppet-enterprise-installer -c pe.conf**

|  |
| --- |
| Ref:[**https://puppet.com/docs/pe/2021.4/installing\_pe.html#installing\_pe**](https://puppet.com/docs/pe/2021.4/installing_pe.html#installing_pe) |



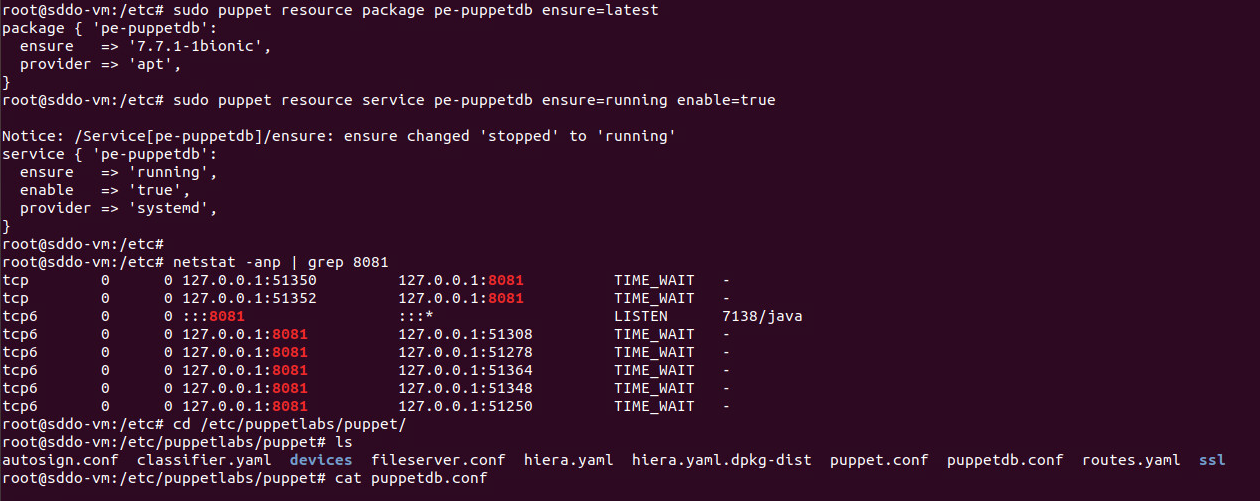


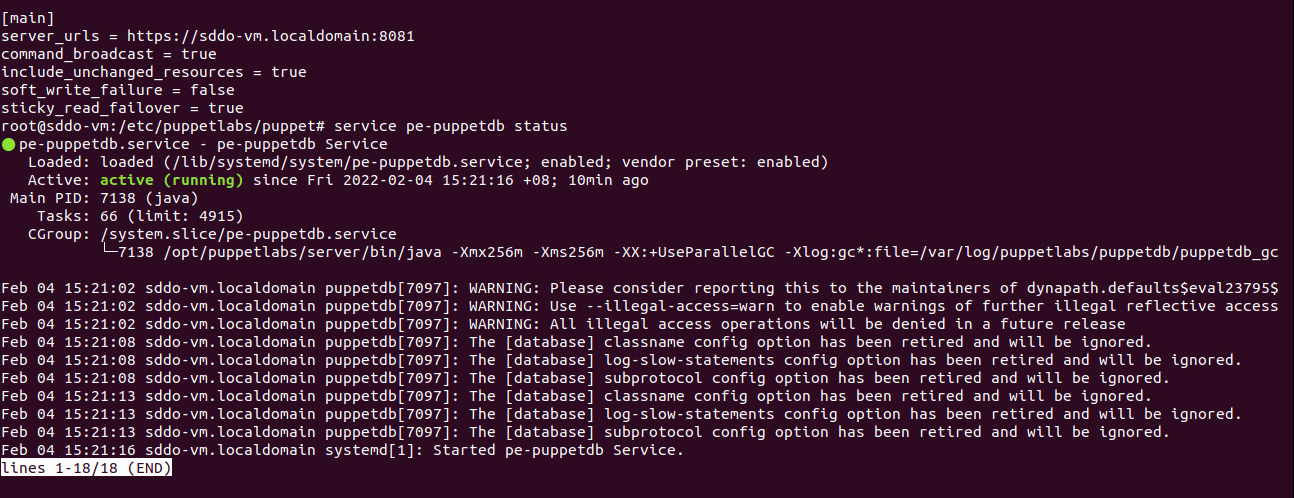
2.4 Install PuppetDB

|  |
| --- |
| Ref: <https://docs.huihoo.com/puppet/puppetdb/1/install.html> |

**# sudo puppet resource package pe-puppetdb ensure=latest**

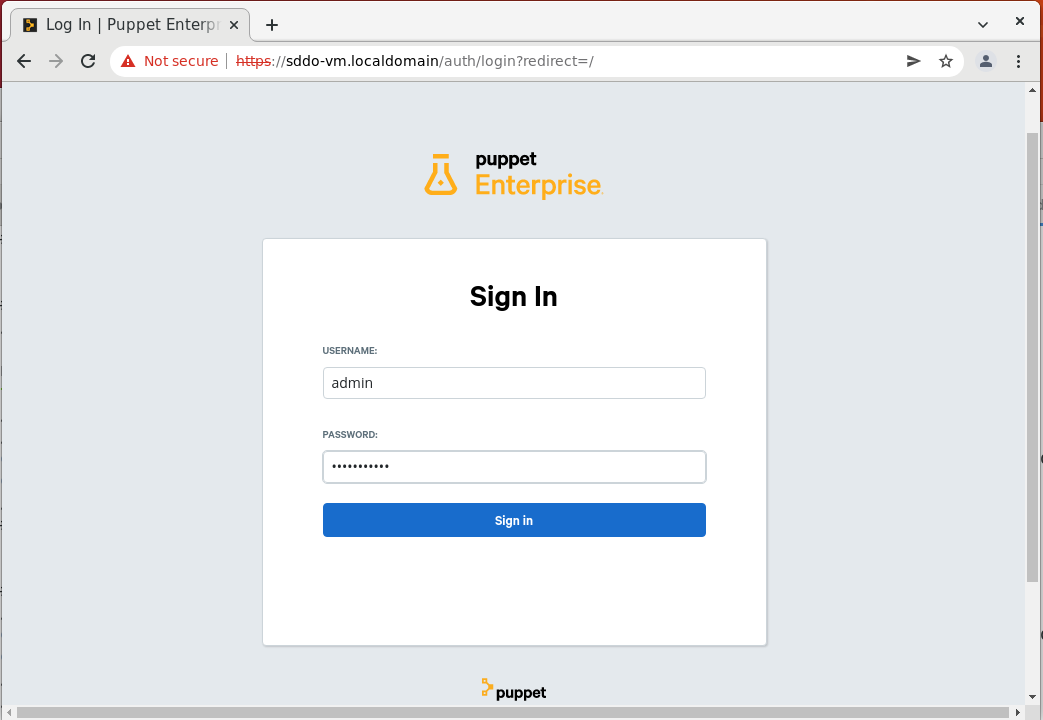
**# sudo puppet resource service pe-puppetdb ensure=running enable=true**

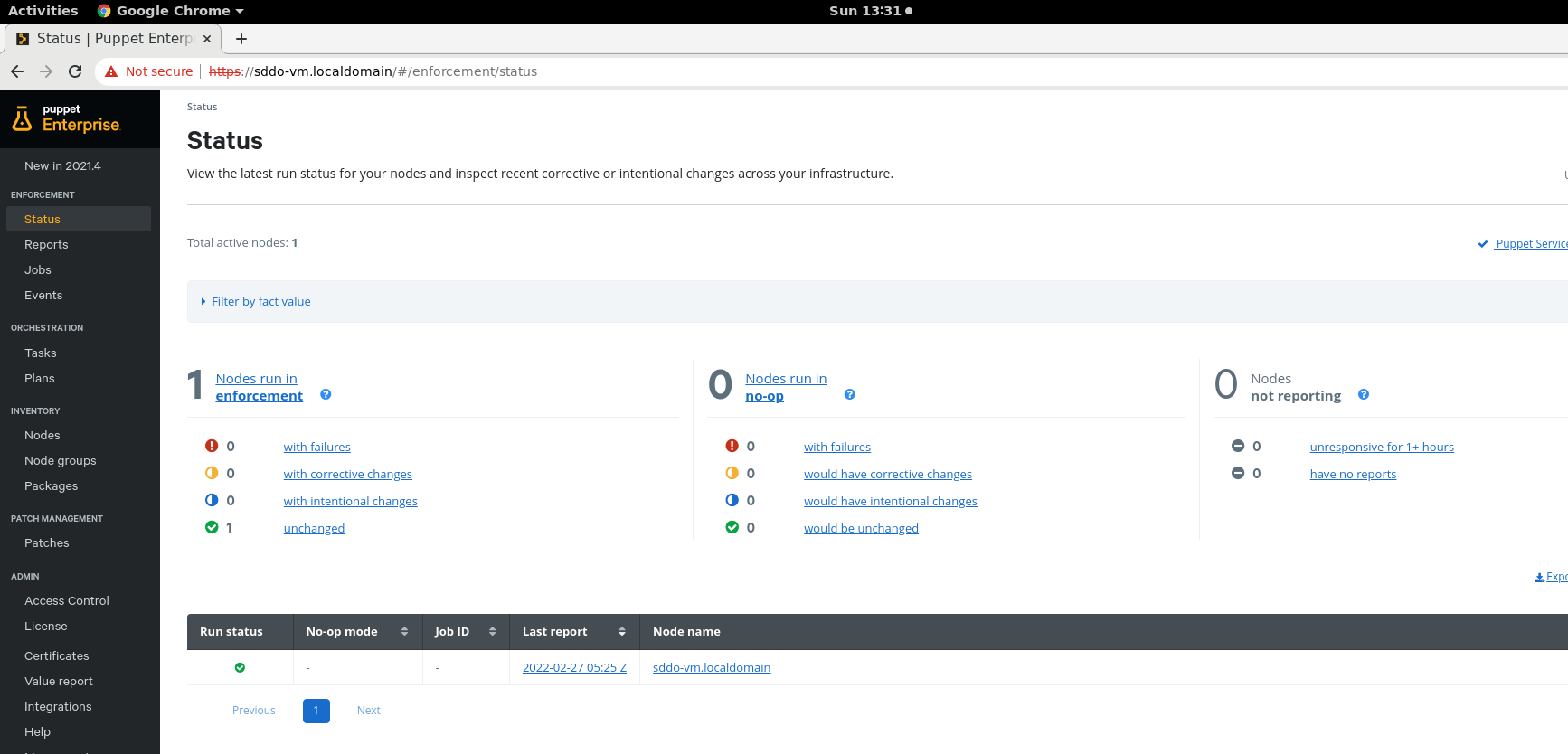




2.5 Accessing the console  
  
The console is the web interface for Puppet Enterprise.

Open a web browser (e.g. Google Chrome), enter **http://localhost**. Puppet Enterprise console will be shown.   
  
Enter username: **admin**, password: **password1234** to sign in.



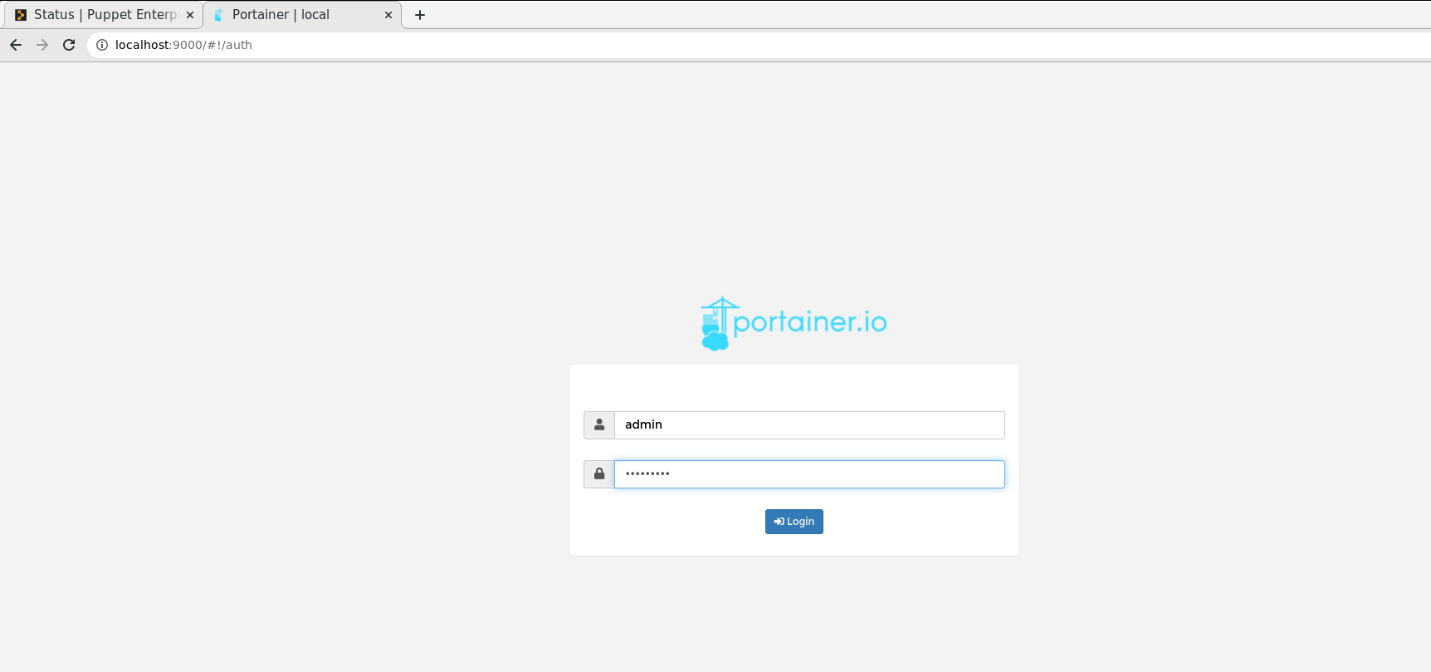


|  |
| --- |
| Optional: Reset the console administrator password |
| If you're unable to log in to the console as admin, you can change the password from the command line of the node running console services.  Log in as root to the node running console services (usually your primary server) and reset the console admin password: |

|  |
| --- |
| Ref: <https://puppet.com/docs/pe/2021.1/console_accessing.html> |

3. Install and Setup puppet-agent Container

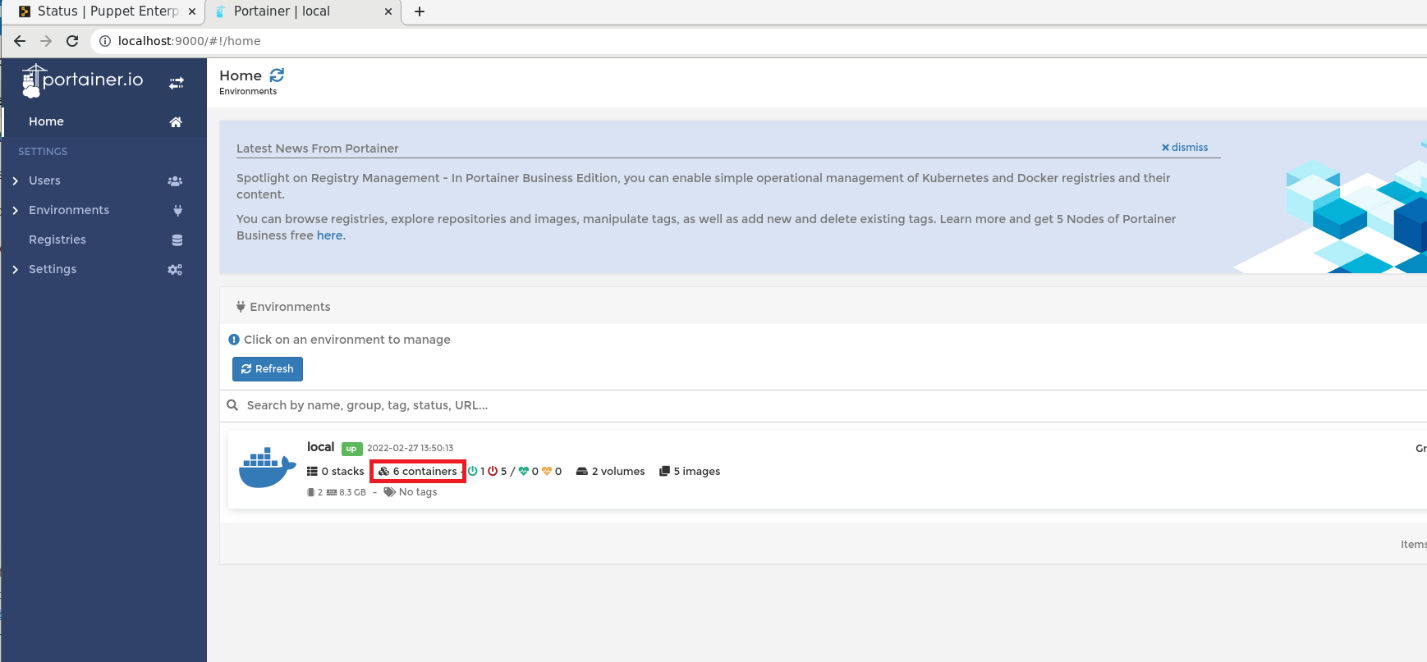
Open a web browser (e.g. Google Chrome), enter **http://localhost:9000**.



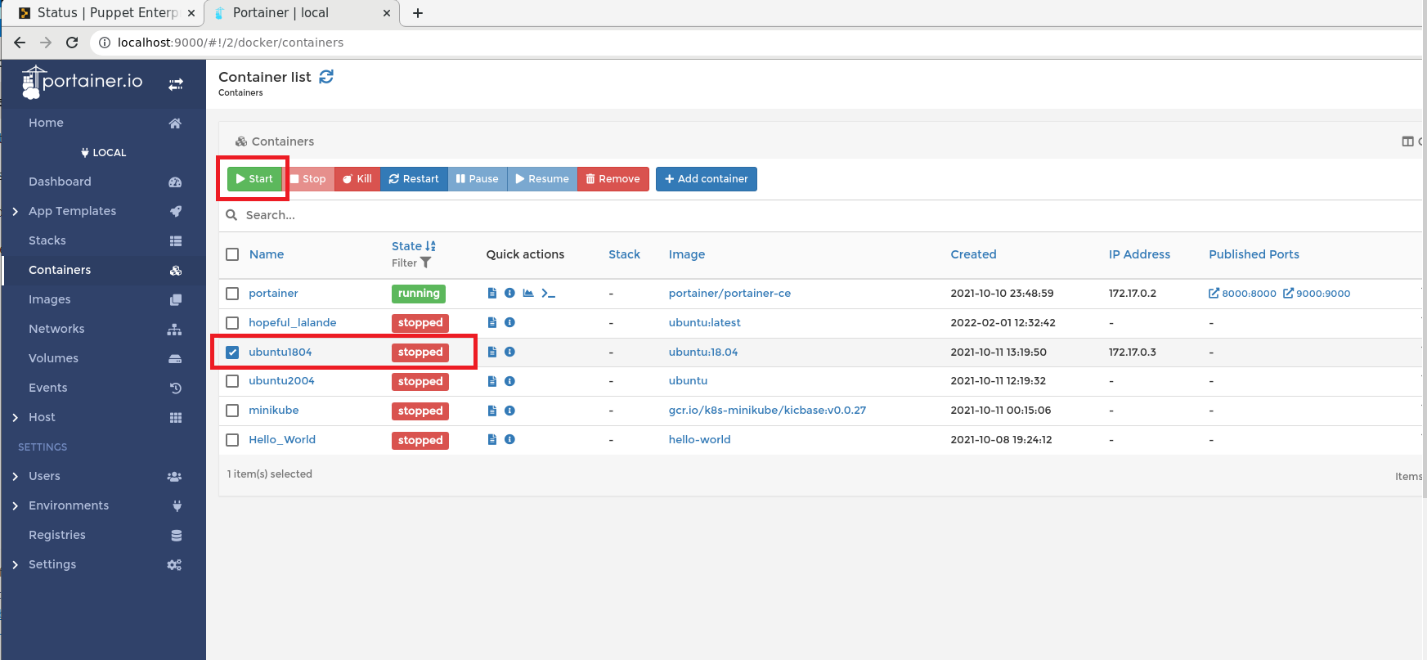
Enter username: **admin**, password: **admin!234** to sign in.

Note: Portainer is a lightweight management UI which allows you to easily manage your Docker host.

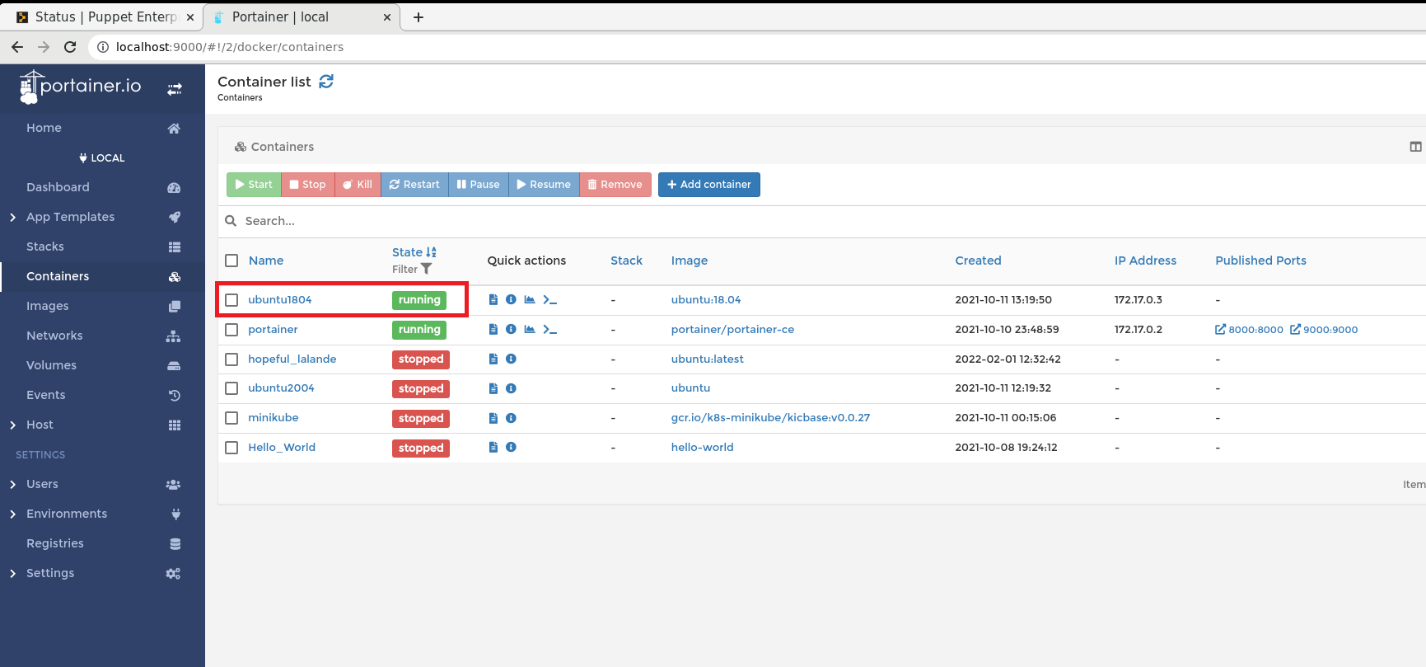
Click **containers**



Select the existing **ubuntu1804** container and click **Start**.

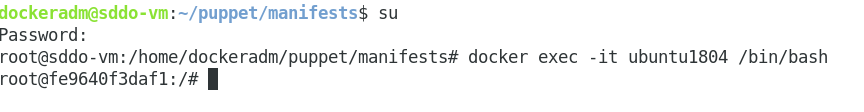


**ubuntu1804** container is now running.

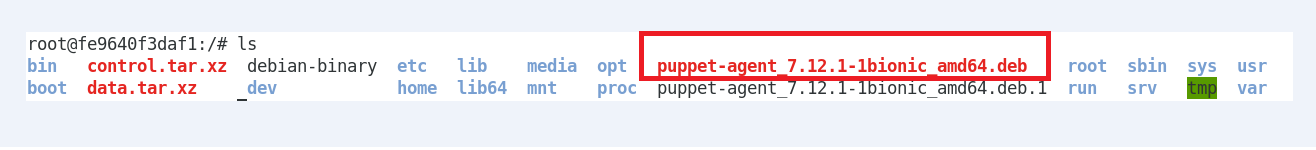


At **Terminal**,  
**# su**

# **docker exec -it ubuntu1804 /bin/bash**

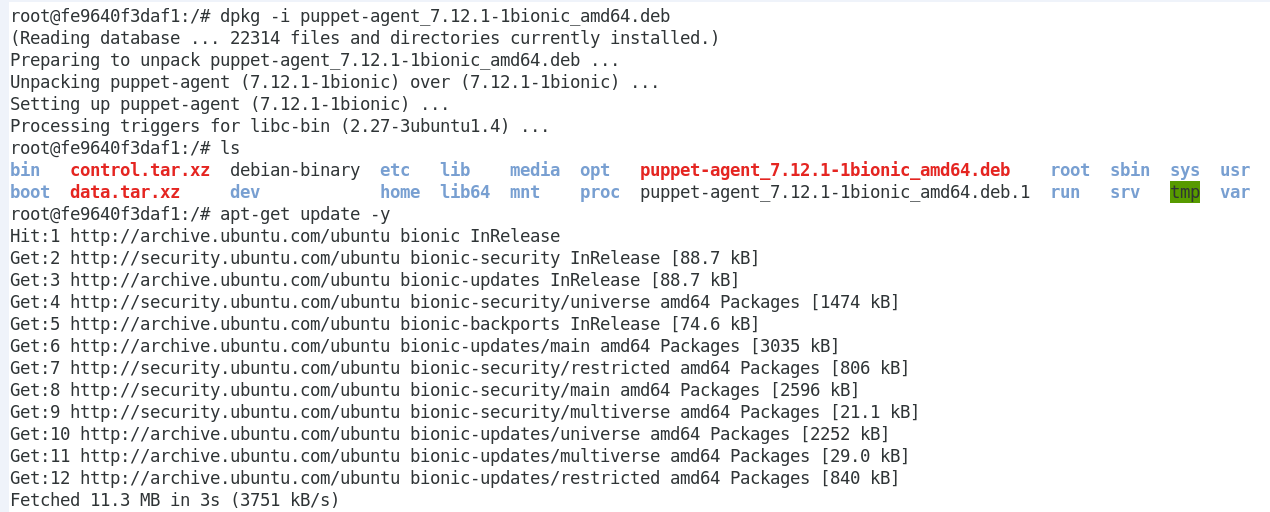
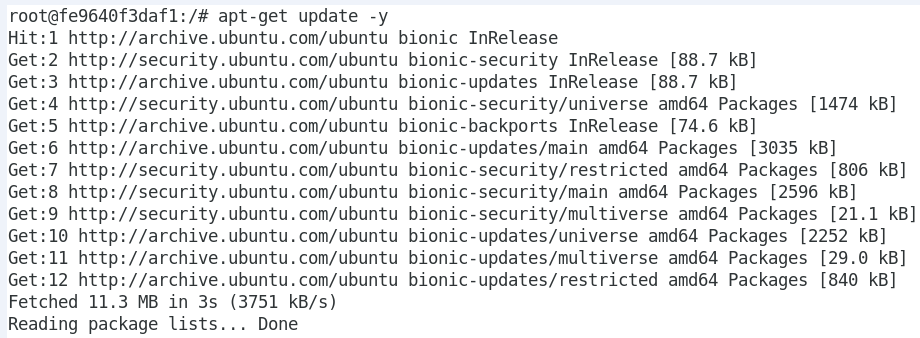
  
**<container name >** **# wget --content-disposition 'https://pm.puppetlabs.com/puppet-agent/2021.4.0/7.12.1/repos/deb/bionic/puppet7/puppet-agent\_7.12.1-1bionic\_amd64.deb'**

  
  
Verify the file is downloaded.  
**# ls**

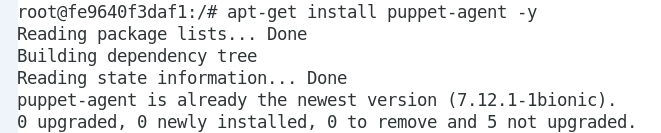


Install the release package.

**<container name > # dpkg -I <file with .deb>**

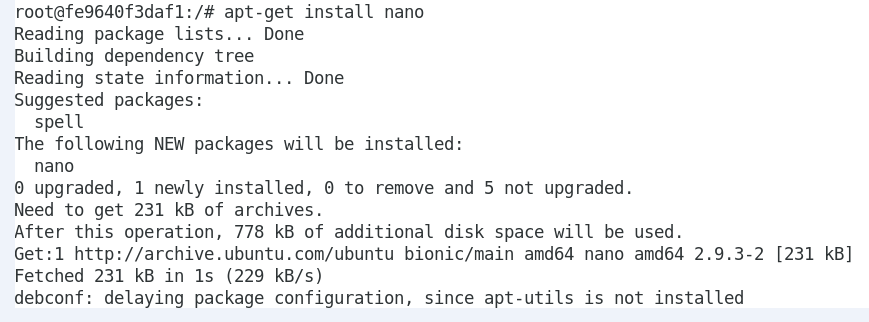
  
  
Update the apt package lists  
**# sudo apt-get update**

Installing the agent  
**# sudo apt-get install puppet-agent**



Installing Nano (Simple Text Editor)

**# sudo apt-get install nano**



Configuring Puppet Agent  
  
Install iproute

**# apt-get install iproute2**

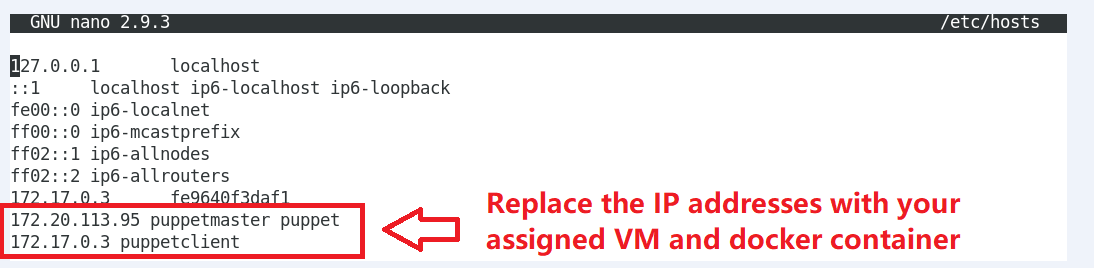
**# ip addr | grep global**

Write down the ip shown in the above output below.

|  |  |
| --- | --- |
| IP Address |  |

**# nano /etc/hosts**

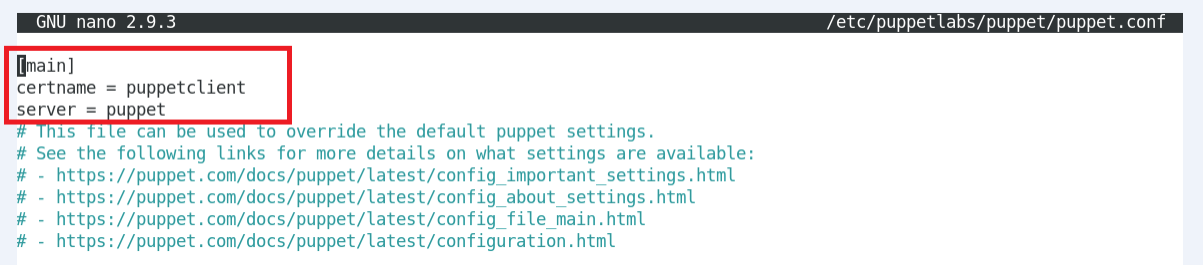
Make the change to the file as per below.



Save and exit the file.

**# nano /etc/puppetlabs/puppet/puppet.conf**

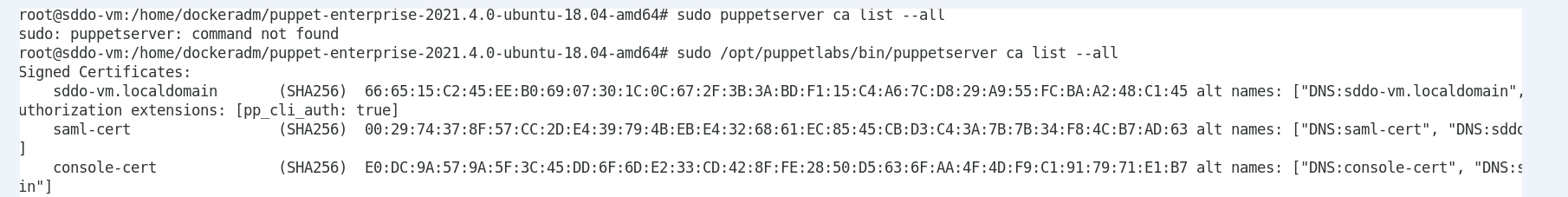
  
  
Make the change to the file as per below.



Save and exit the file.

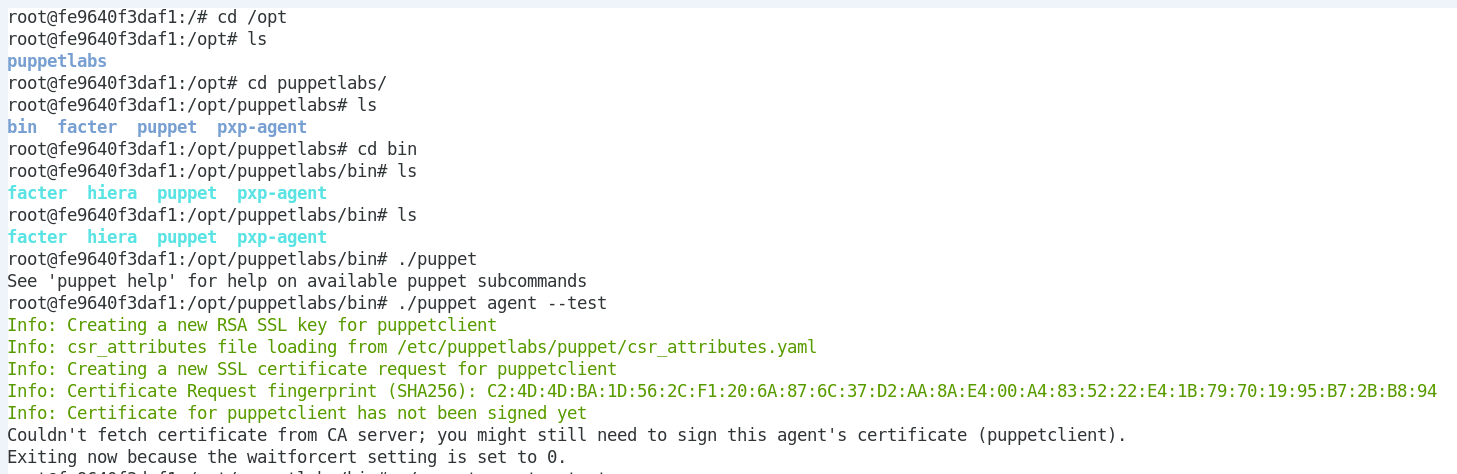
|  |
| --- |
| Ref: <https://www.cloudbees.com/blog/install-and-configure-puppet-agent> |

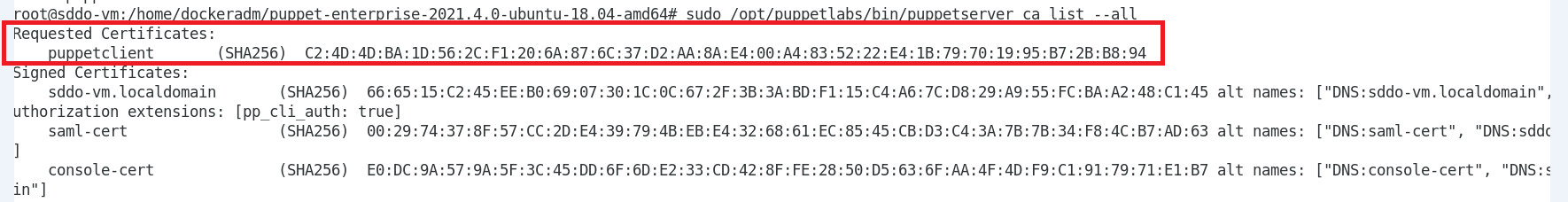
Before you can test your agent, you need to sign its certificate. With the help of certificates, Puppet can check the identity of nodes. It emits these certificates when an agent sends its **first request** to the server; in the next iterations, the agent then uses the certificate to identify itself.  
  
On the **server node (Puppet Master)**, list all the available certificates. Notice that there is no cert from the **client node (Puppet Agent)** yet.  
**# sudo /opt/puppetlabs/bin/puppetserver ca list --all**



On the **client node (Puppet Agent)**, sends its **first request**.

**# /opt/puppetlabs/bin/puppet agent --test**

  
  
On the **server node (Puppet Master)**, list all the available certificates. Notice that the cert from the **client node (Puppet Agent)** is now shown.

  
  
Sign the certificates.

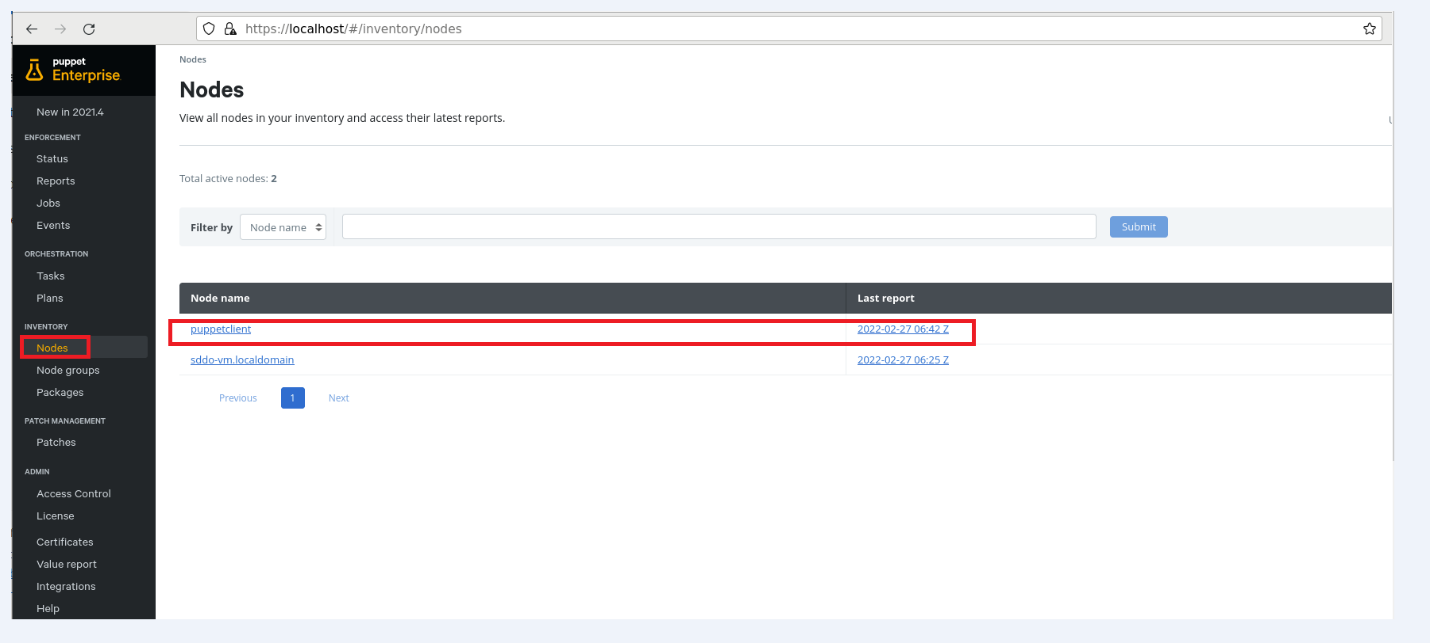
**# sudo /opt/puppetlabs/bin/puppetserver ca sign –all**

On the **client node (Puppet Agent)**, use the following command to test the communication between the server and client node.

**# /opt/puppetlabs/bin/puppet agent –test**



On the **server node (Puppet Master)**, access the console to see the new **client node (Puppet Agent)** is now added.

****

4. Extend your path variable to include the puppet command

Edit ~/.bashrc file to make permanent modifications. Use a text editor like nano to open the file

**nano ~/.bashrc**

append the file with the following

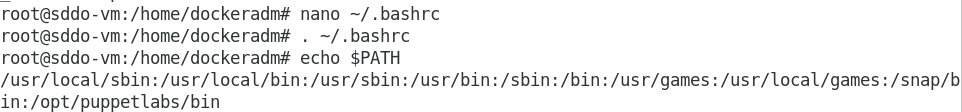
**export PATH=$PATH:/opt/puppetlabs/bin**

Force changes in current terminal session

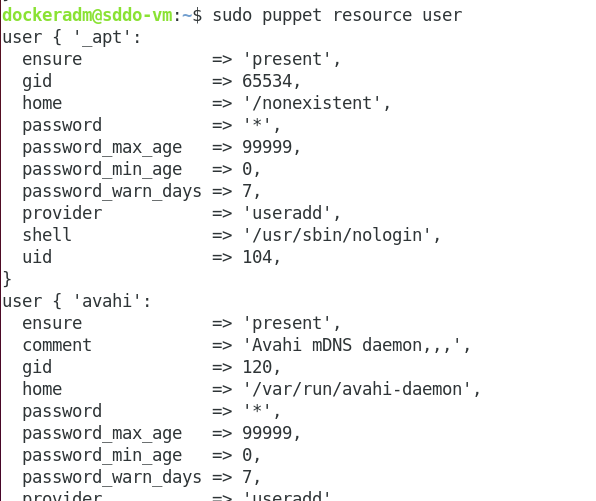
**. ~/.bashrc**

Open a new terminal session and issue the following to verify

**echo $PATH**

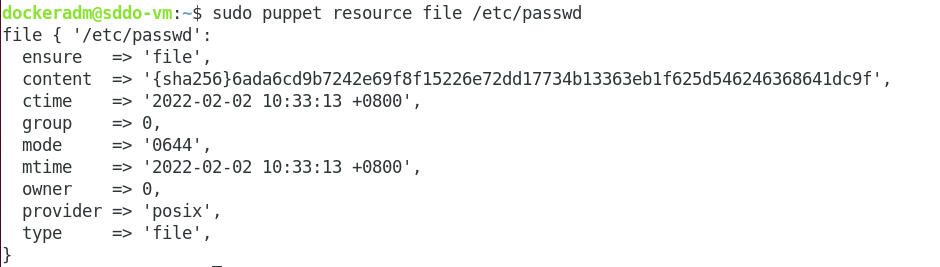


5. Use Puppet Resource Command to query and change resources (Query all users)  
**sudo puppet resource user**



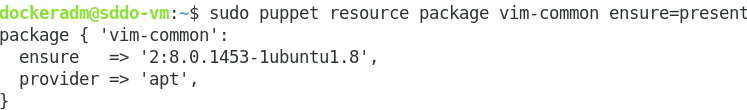
6. Query state of the file "**/etc/passwd**"

**sudo puppet resource file /etc/passwd**



7. Ensure package "**vim-common**" is installed

**sudo puppet resource package vim-common ensure=present**



**Part 2: Create yourself a user**

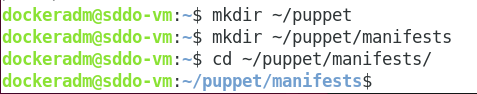
8. Create a manifest "user.pp" to manage your user

* Create a new directory "manifests" under your home directory and create a manifest file "user.pp"

**mkdir ~/puppet/**

**mkdir ~/puppet/manifests/**

**cd ~/puppet/manifests/**



\* ~ is a "shortcut" to the home directory

\* Puppet language files are called manifests, and are named with the .pp file extension.

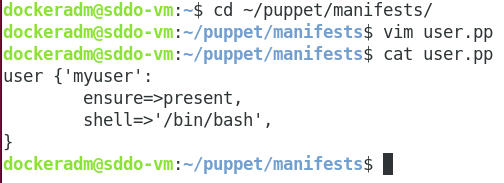
**vim user.pp**

9. Declare your user including its shell as "/bin/bash"

* Declare your user with your typically username, ensure that he is present and set its shell to "/bin/bash"

**user { 'myuser':   
 ensure => present,   
 shell => '/bin/bash',**

**}**



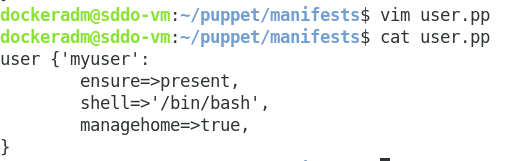
10. Set it to also manage your home directory

* Enforcing the user now would create your home directory depending on the default of your operating system, to enforce its creation set the attribute "**managehome**" to true.

**user { 'myuser':**

**ensure => present,   
 shell => '/bin/bash',   
 managehome => true,**

**}**

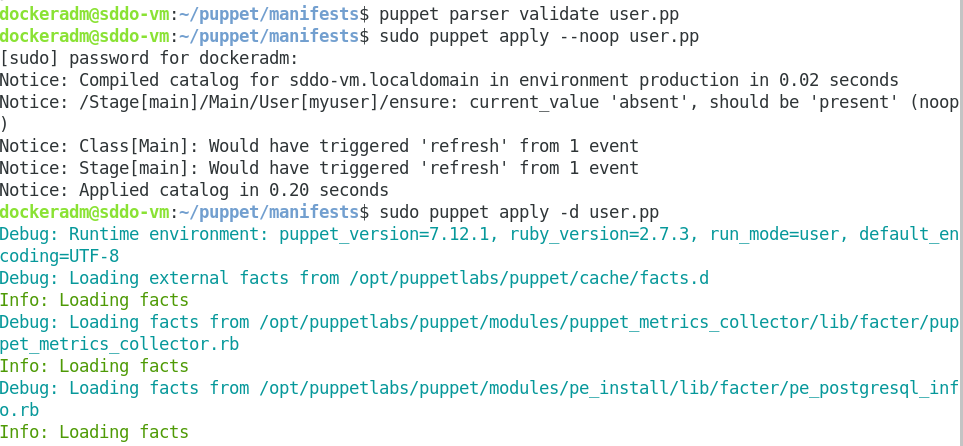


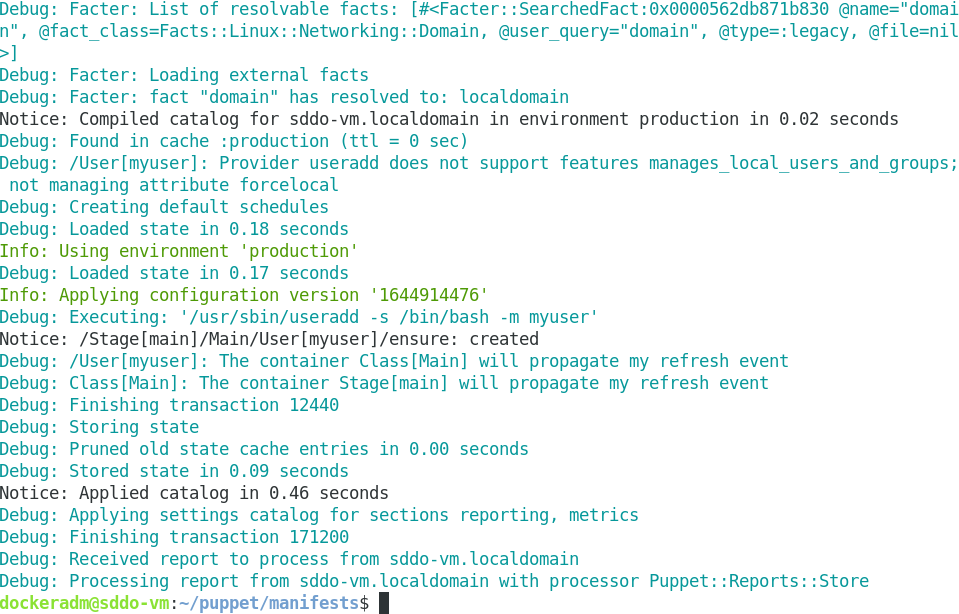
11. Apply the manifest with "**puppet apply user.pp**"

* Save the manifest and then use "puppet apply" to enforce it. Add "**--noop**" if you want to simulate first, add "**-d**" if you want to see the commands executed. If you are not sure about the syntax run the syntax validation before.

**puppet parser validate user.pp**

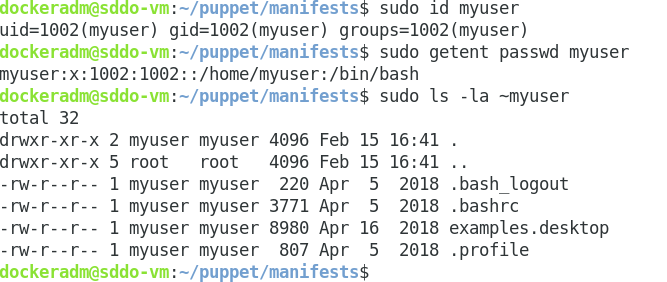
**sudo puppet apply --noop user.pp  
sudo puppet apply -d user.pp**





12. To verify the user's existence you can use "id" or "getent", also verify the creation of the home directory with "ls -la".

**sudo id myuser  
sudo getent passwd myuser  
sudo ls -la ~myuser**



**Part 3: Manage the group membership of your us**

13. Add a group definition for your user's private group to your manifest (a group with the same name)

* Open your manifest user.pp and add a group resource.

**group { 'myuser':   
 ensure => present,**

**}**



14. Add a group definition for your user's private group to your manifest (a group with the same name)

* Add a group definition for a administrative group "admins" to your manifest (Add another group resource)

**group { 'admins':   
 ensure => present,**

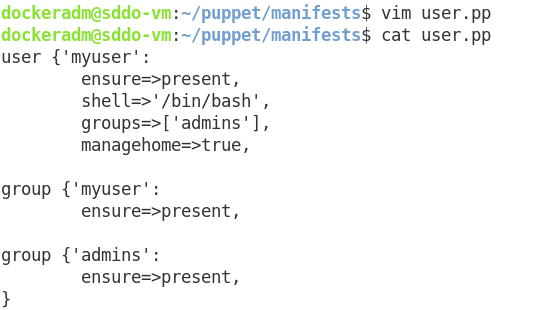
**}**



15. Change your user definition to set its private group as its primary group and the administrative group as secondary

* Add the groups to your user definition

**user { 'myuser':   
 ensure => present,   
 shell => '/bin/bash',   
 gid => 'myuser',   
 groups => [ 'admins' ],   
 managehome => true,   
}**

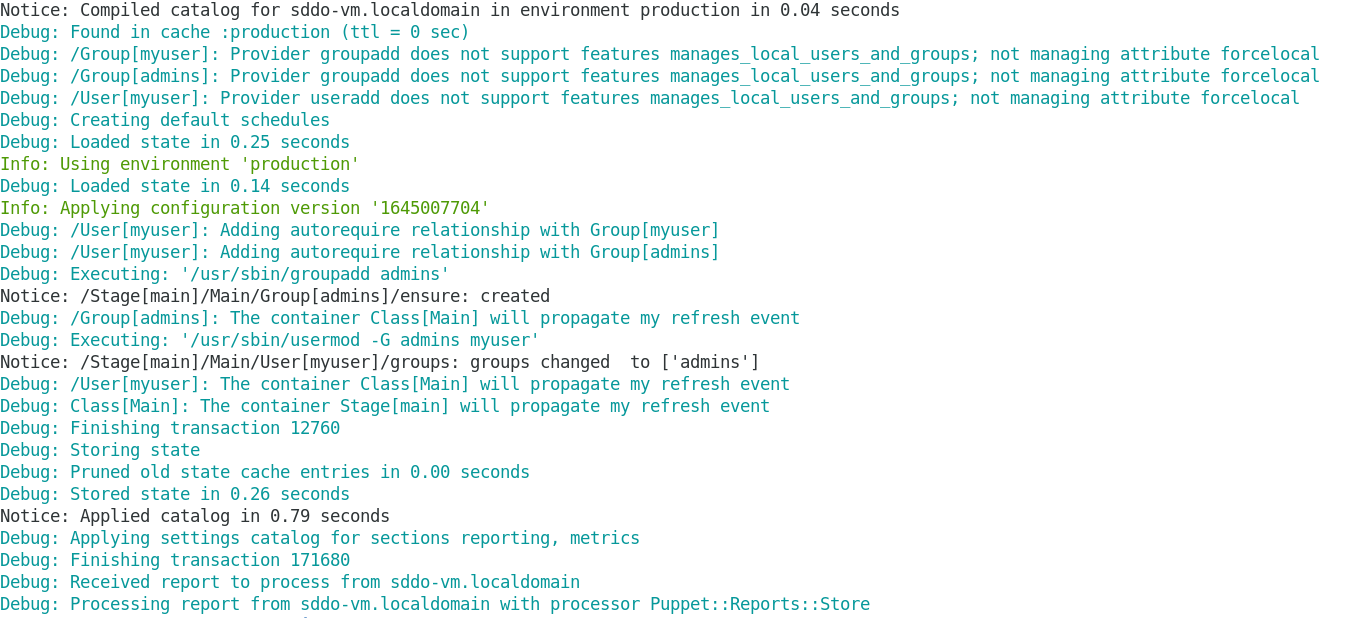


16. Apply the manifest with "puppet apply user.pp"

* Save the manifest and then use "puppet apply" to enforce it. Add "--noop" if you want to simulate first, add "-d" if you want to see the commands executed.

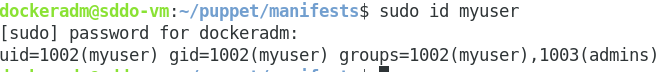
**sudo puppet apply --noop user.pp   
sudo puppet apply -d user.pp**





* To verify the user's group membership you can use "id".

sudo id myuser

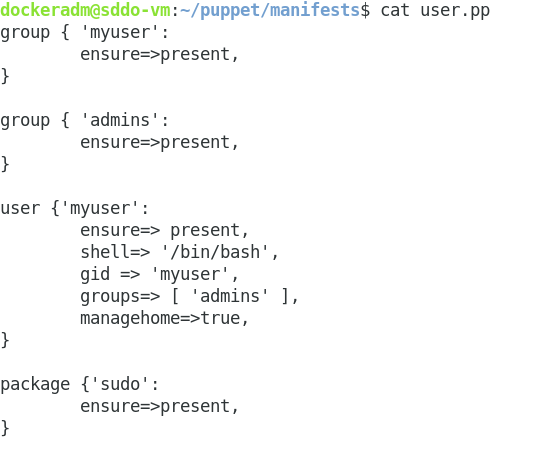


**Part 4: Grant your user administrative privileges using sudo**

17. Ensure package "sudo" is installed

* Edit your manifest to include a definition ensuring the package "sudo" is installed.

package { 'sudo':   
 ensure => present,   
}



18. Manage a file in "/etc/sudoers.d" for your user allow to execute all commands as root

* Add a file resource "/etc/sudoers.d/myuser" with owner "root", group "root", mode "0400" and content "myuser ALL=(ALL) NOPASSWD: ALL".

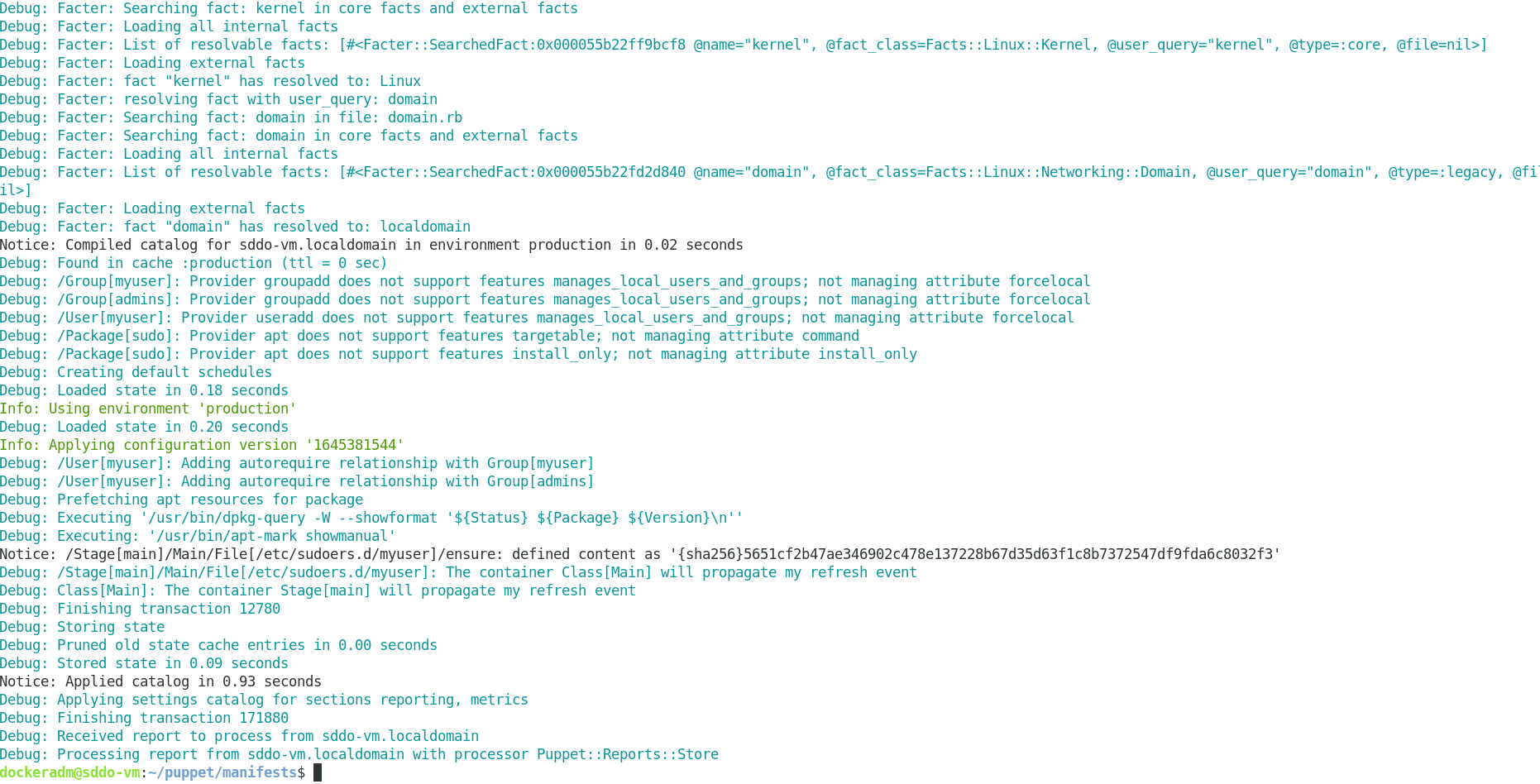
file { '/etc/sudoers.d/myuser':   
 ensure => file,   
 owner => 'root',   
 group => 'root',   
 mode => '0400',   
 content => "myuser ALL=(ALL) NOPASSWD: ALL\n",   
}



19. Apply the manifest with "puppet apply user.pp"

* Save the manifest and then use "puppet apply" to enforce it. Add "--noop" if you want to simulate first, add "-d" if you want to see the commands executed.

sudo puppet apply --noop user.pp  
sudo puppet apply -d user.pp



* You can verify the sudo permissions by displaying them with sudo or testing it as the user.

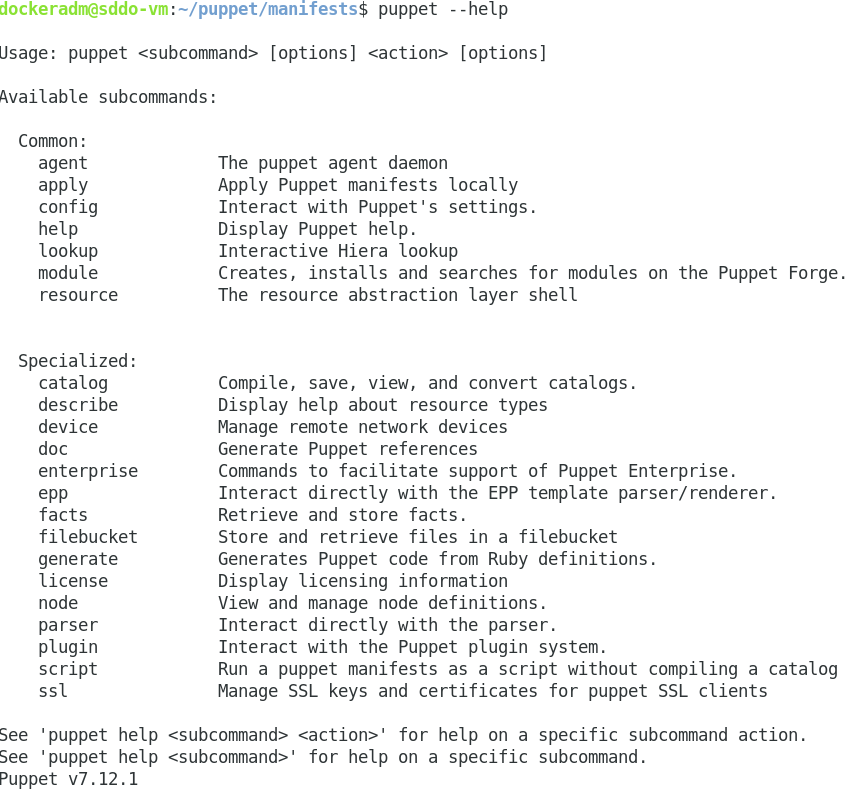
sudo -l -U myuser



20. Explore about puppet resource types

* In the terminal, type the following command to display a list of Puppet relevant subcommands

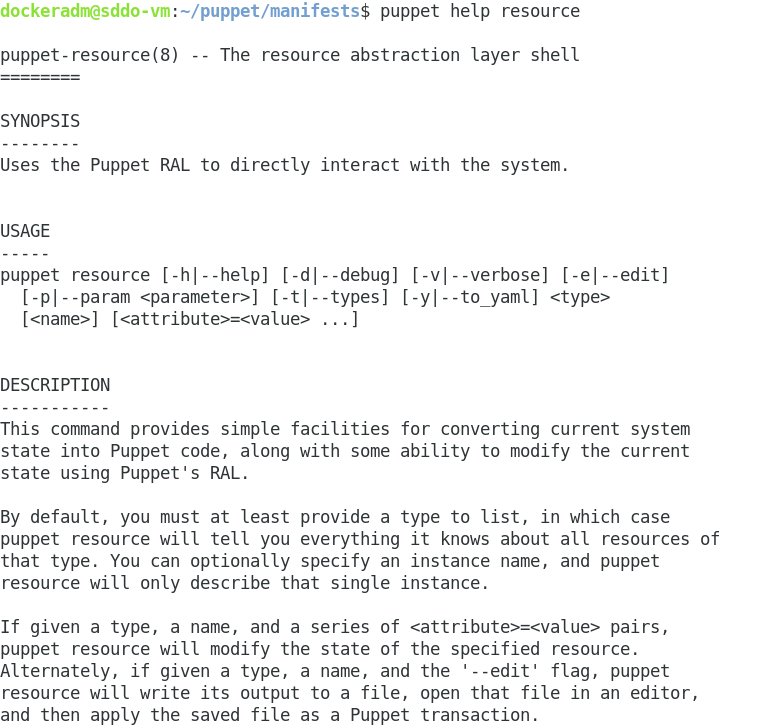
puppet –help

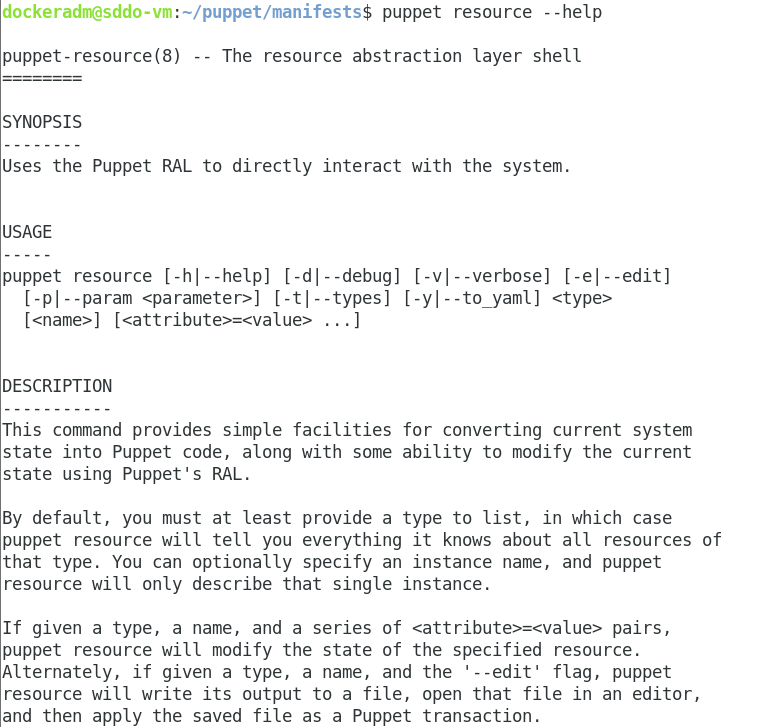


In our case, we are interested in the subcommand “resource” which we will use to find the information about inbuilt puppet resource types.

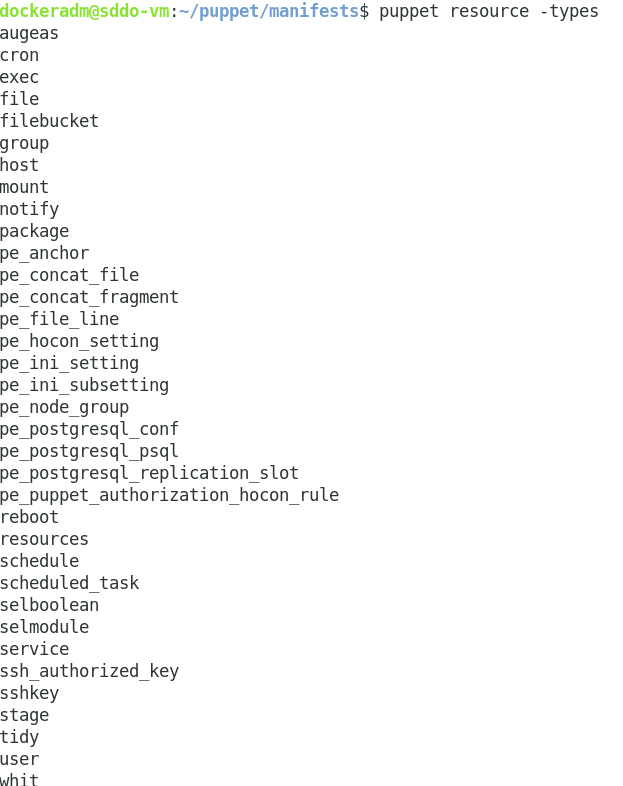
21. In the terminal, type any of the following commands to display a list of actions associated with the puppet subcommand “resource“:  
  
puppet help resource

puppet resource --help





22. In this case, we have the resource as subcommand and –types as action. In this case, we have the resource as subcommand and –types as action.  
  
puppet resource –types



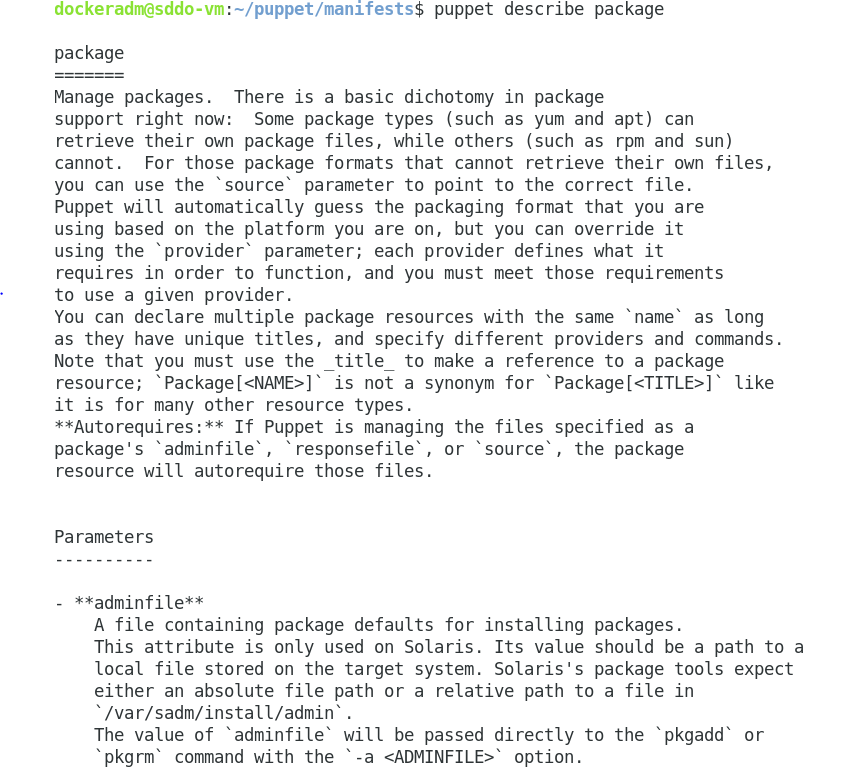
Puppet has **39** inbuilt core resource types.

Each type supports a list of attributes. These attributes provide a detailed description that Puppet uses to manage the resource.

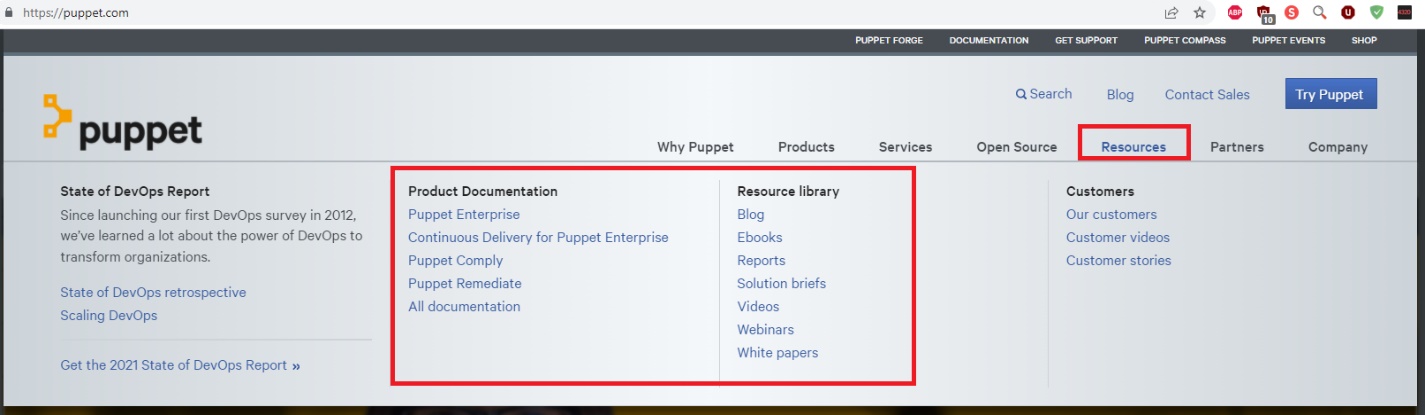
23. To find out all the attributes associated with the puppet resource type, use the following command: puppet describe <resource type name>

Parameters will list all the available attributes for that resource type.

24. To check for the appropriate attribute to declare for the package resource type we use the puppet describe command: puppet describe package



25. You can find additional resources at <https://puppet.com>.



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