**Deploy app with Puppet on Microsoft Azure**

Puppet is a configuration management system that allows you to automate the provisioning and configuring of machines, by describing the state of your Infrastructure as Code.

**Tasks:**

**Task 1) Provision a Puppet Master and Node**

**Task 2) Install Puppet Agent on the node**

**Task 3) Configure the Puppet Production Environment**

**Task 4) Test the Production Environment**

**Task 5) Create a Puppet Program**

**Task 6) Run the Puppet Configuration on the Node**

A machine known as a Node will host the PU MRP app. The only task you will perform on the node is to install the Puppet Agent. The Puppet Agent can run on Linux or Windows. For this lab, we will configure the Node in a Linux Ubuntu Virtual Machine (VM).

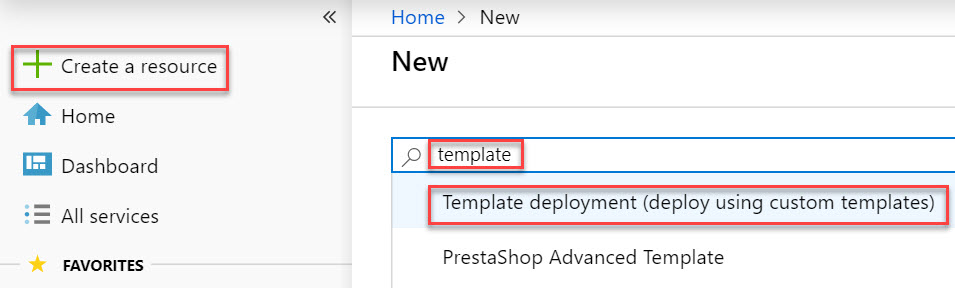
A Puppet Master machine. The rest of the configuration will be applied by instructing Puppet how to configure the Node through Puppet Programs, on the Puppet Master. The Puppet Master *must* be a Linux machine. For this lab, we will configure the Puppet Master in a Linux Ubuntu VM.

Step 1: Open Microsoft Azure Portal

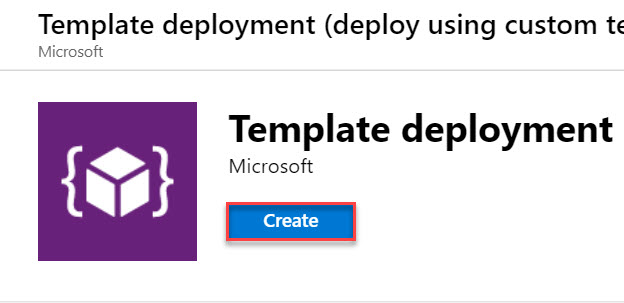
[https://portal.azure.com](https://portal.azure.com/)

**Task 1) Provision a Puppet Master and Node**

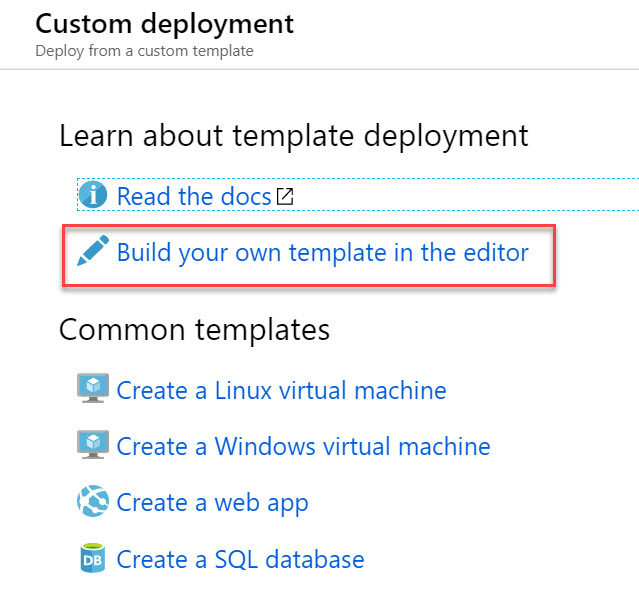
Step 2: Click on **+ Create a resource** -> Search for “**template deployment**”



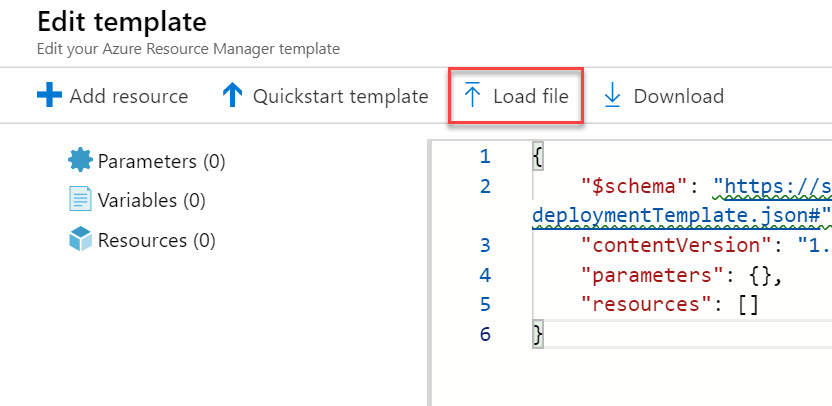
Click on Create button



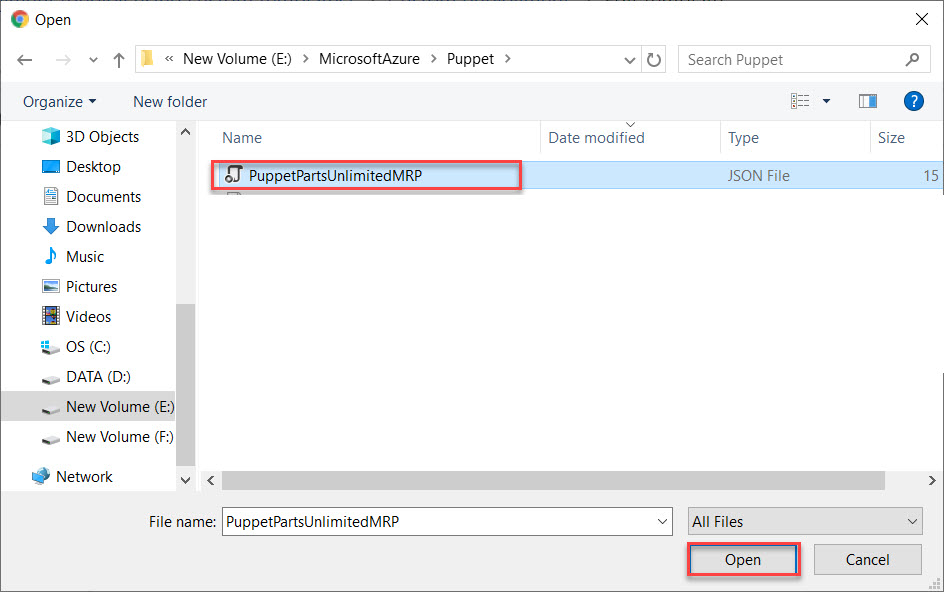
Click on **Build your own template in the editor**



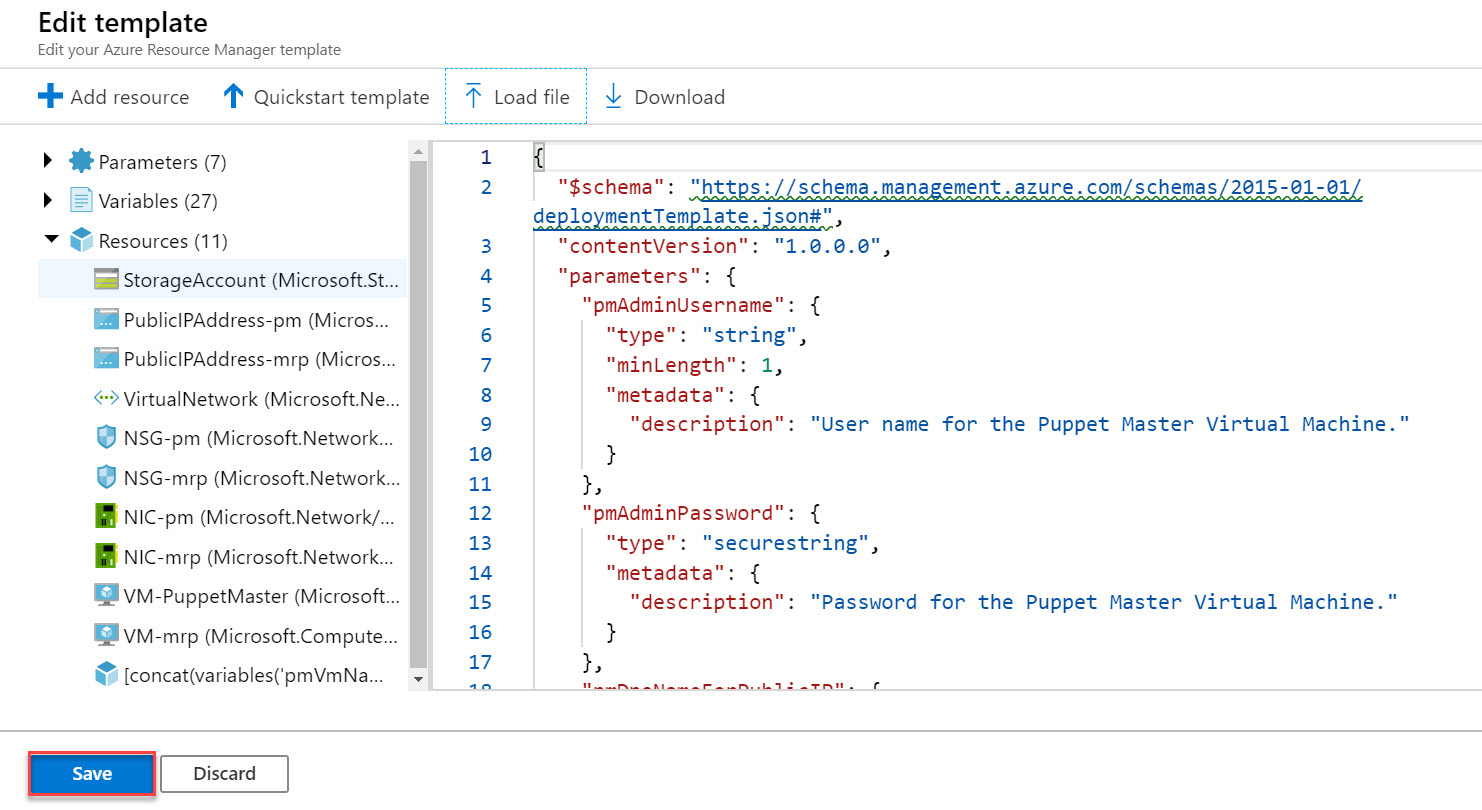
Click on **Load file** option



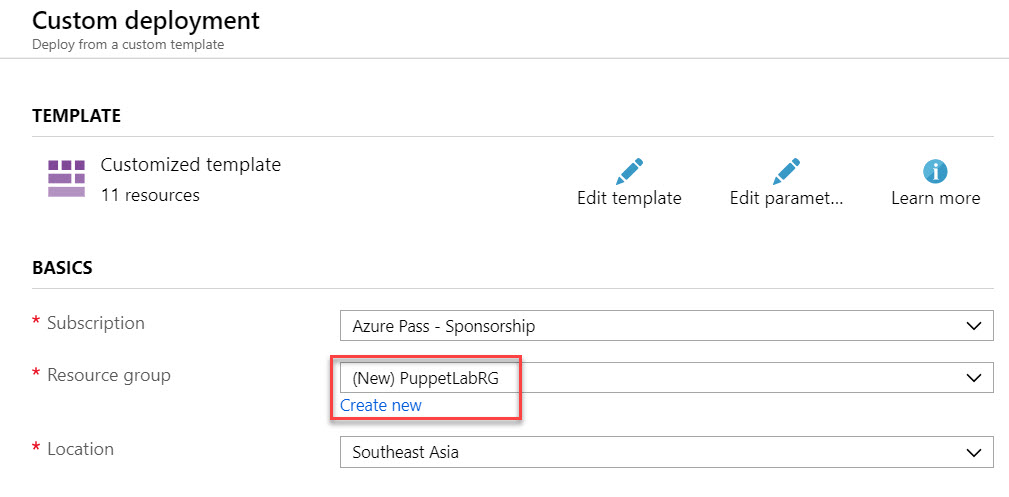
Browse **PuppetPartsUnlimitedMRP.json** file and click on Open button.

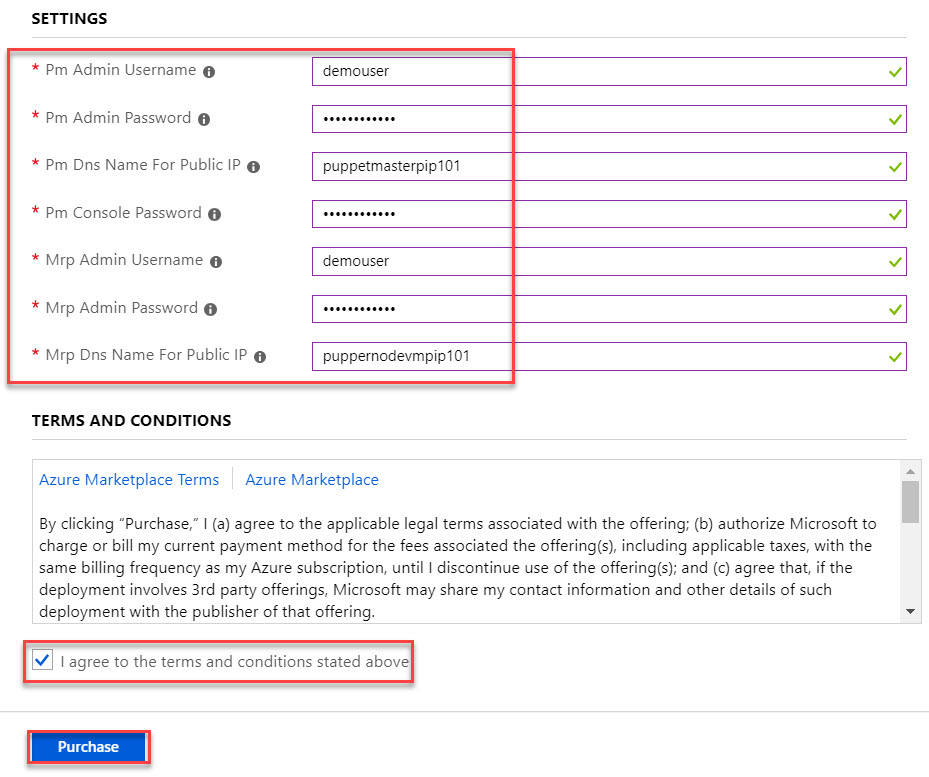


Verify Template and click on **Save** button.



Step 3: Enter Resource Group Name: **PuppetLabRG**





*Note: Here Pm is Puppet Master VM & Mrp is Puppet Node VM*

Pm Admin Username: **demouser**

Pm Admin Password: **Demo@pass123**

Pm Dns Name for Public IP: **Enter Unique name Ex. puppetmasterpip101**

Pm Console Password**: Demo@pass123**

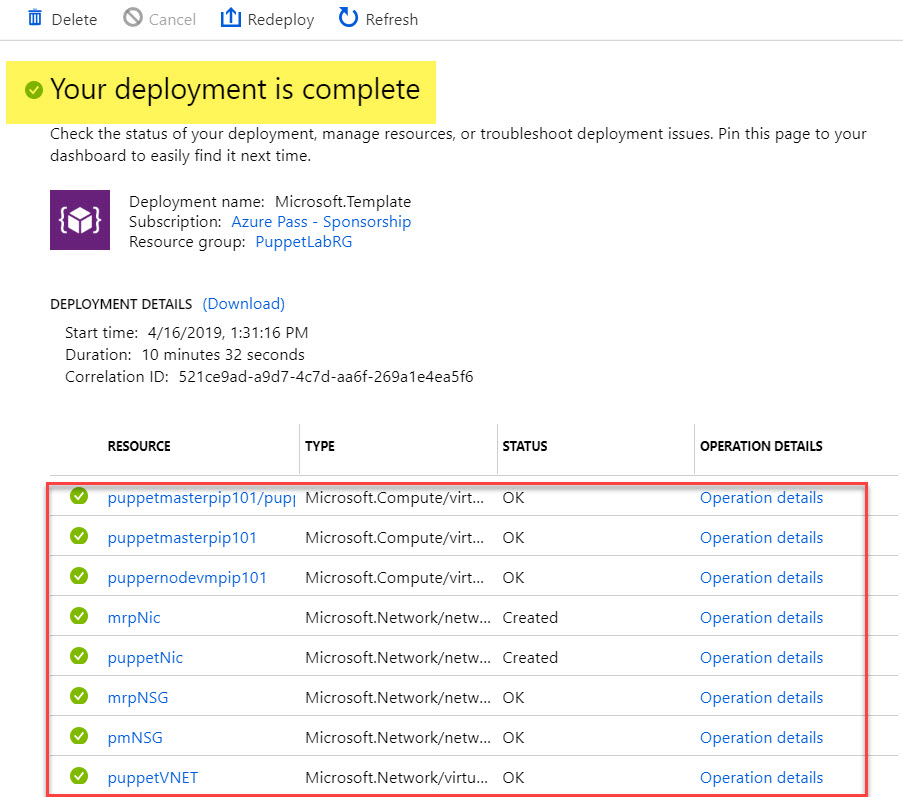
Mrp Admin Username : **demouser**

Mrp Admin Password : **Demo@pass123**

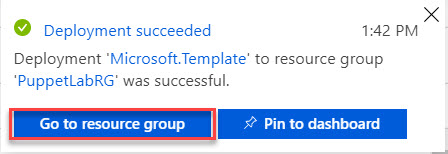
Mrp Dns Name for Public IP: **Enter Unique name Ex. puppetnodevmpip101**

Click on **Purchase** button.

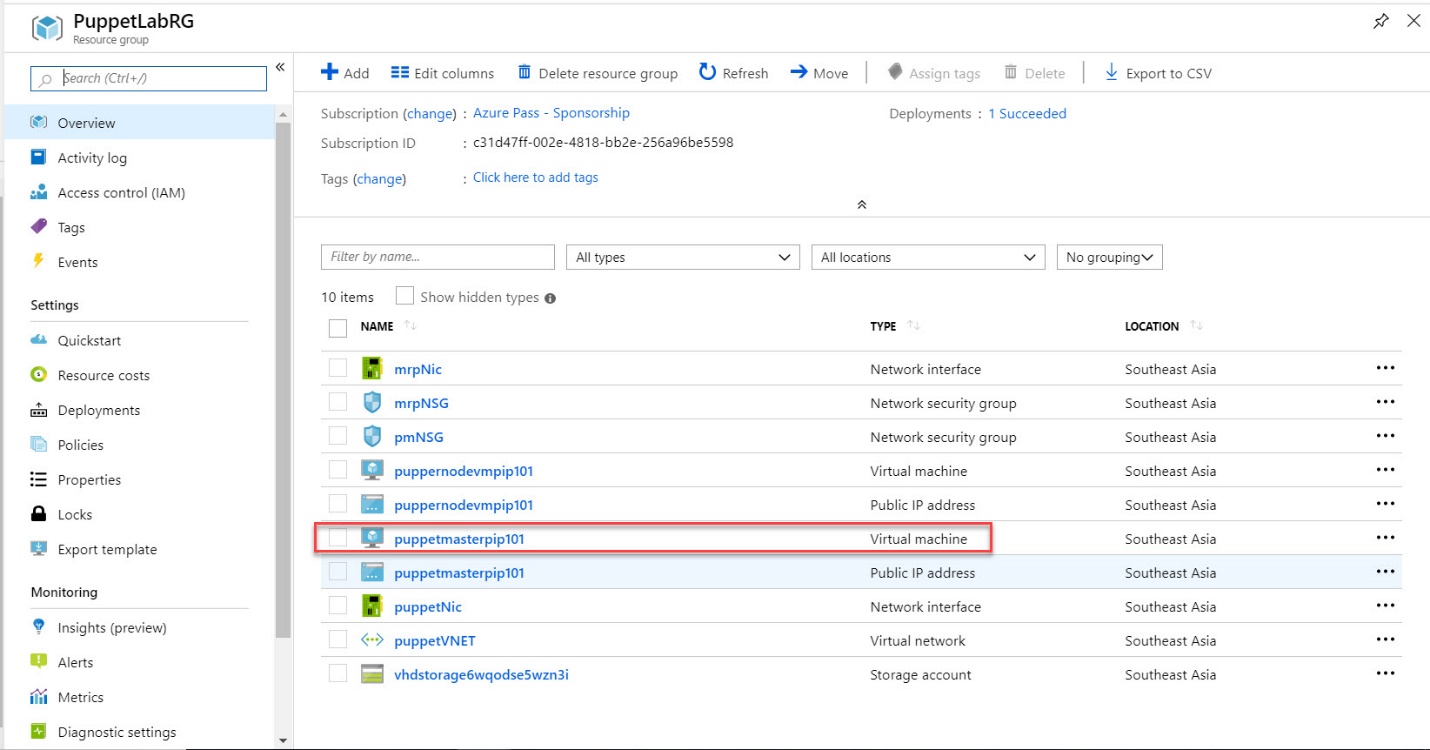
Wait for few minutes to deploy all resources.



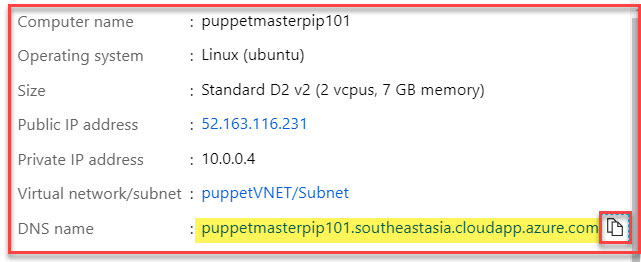
Click on **Go to Resource Group** option



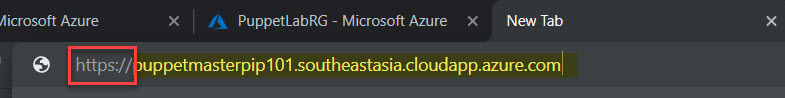
Step 4: Open **Resource Group** and click on **Puppet Master VM**



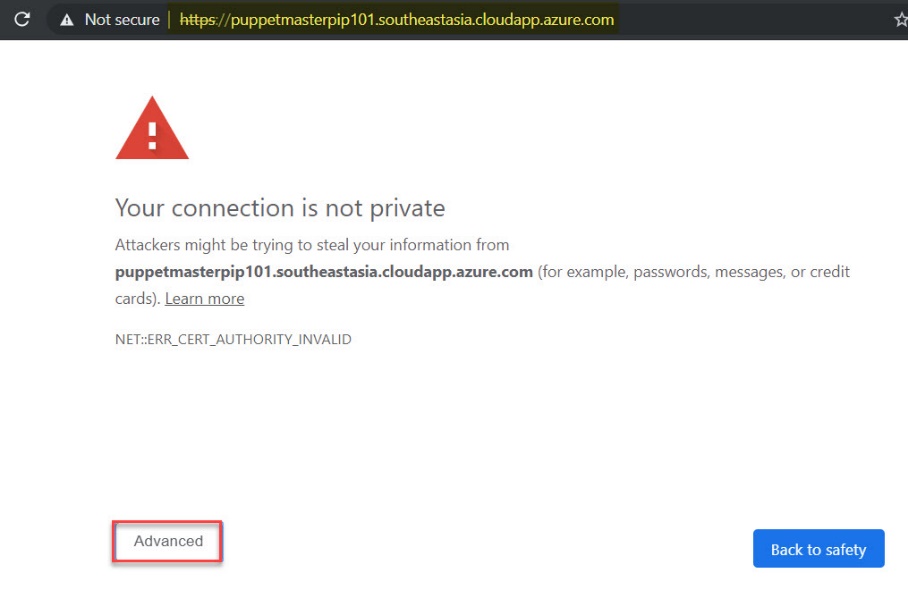
Copy **DNS name**



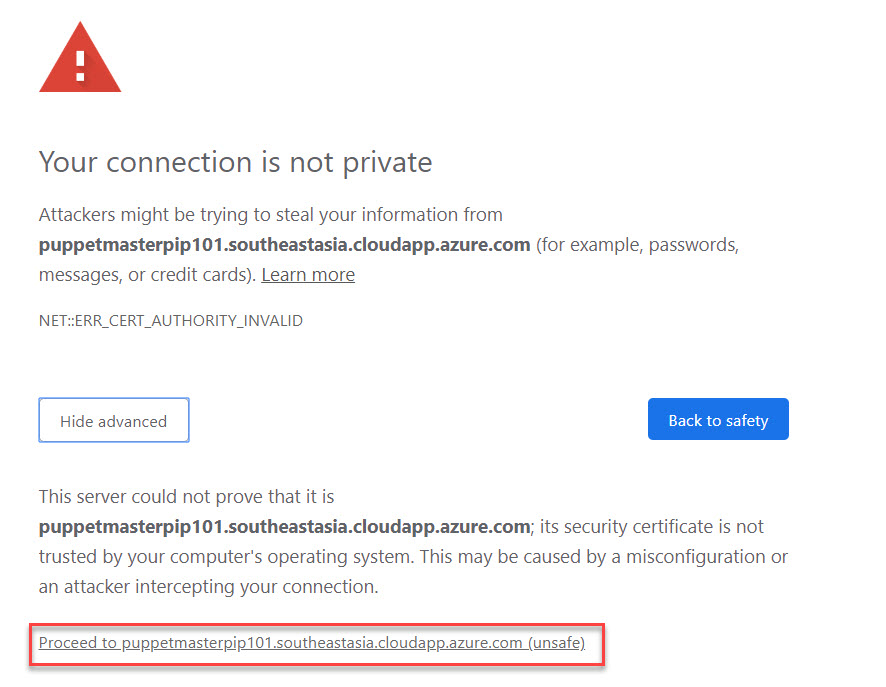
Step 5: Open another tab in browser and type **https://**dnsname



Click on **Advanced** button.



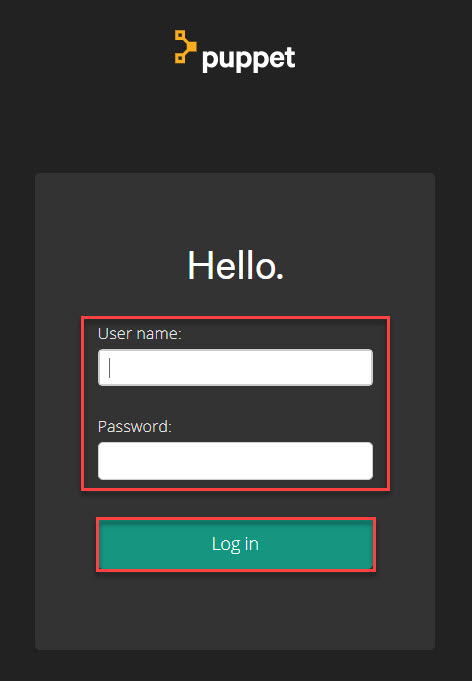
Click on **Proceed** link



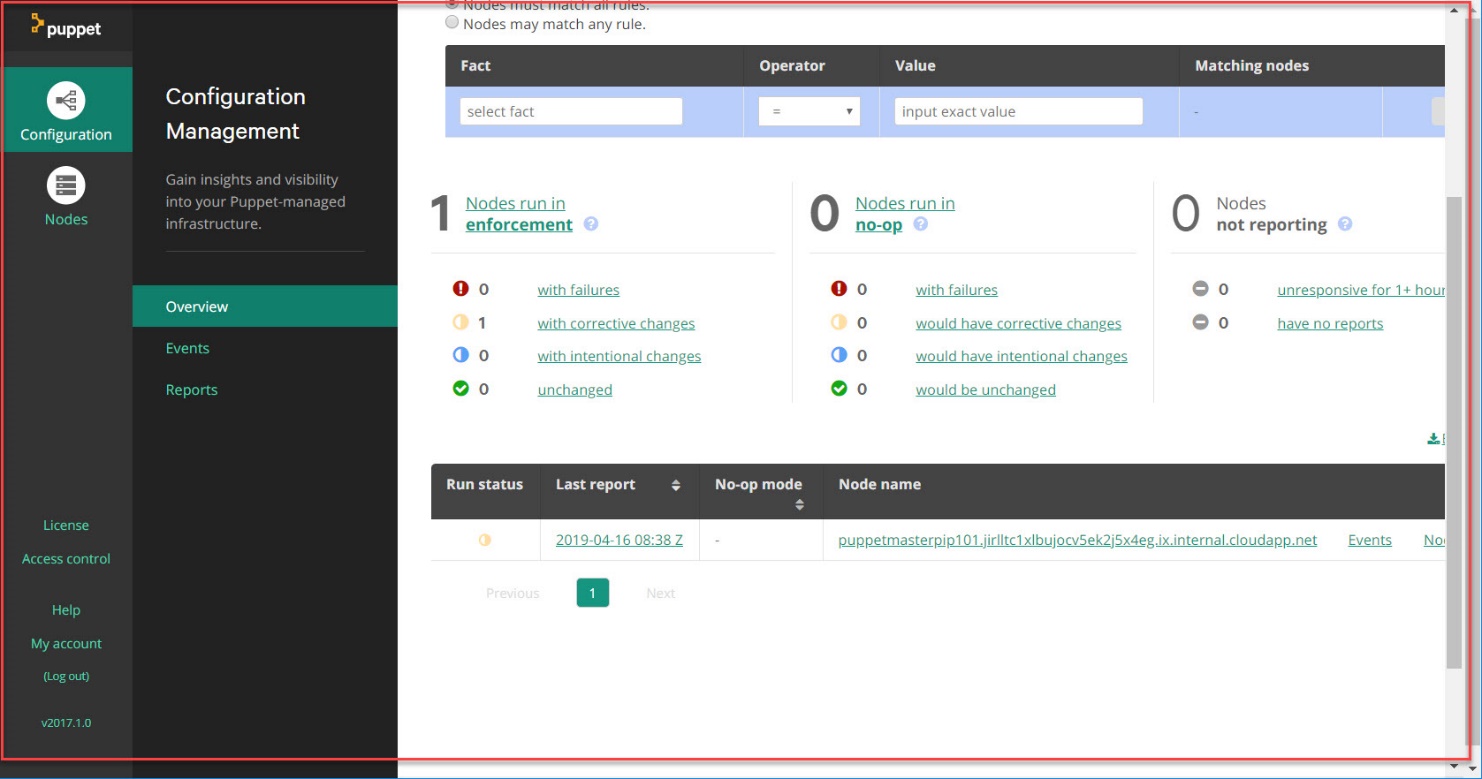
Step 6: **Puppet Console** will open so enter username & password

Username: **admin**

Password: **Demo@pass123**

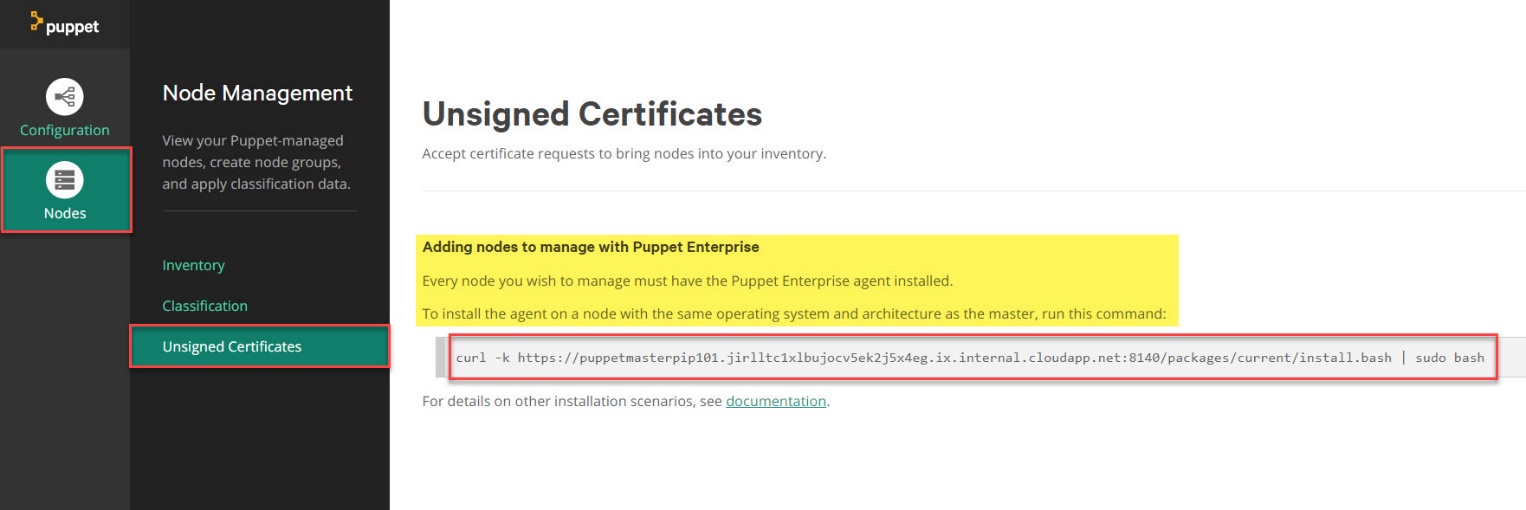


Puppet Console will look like as below:



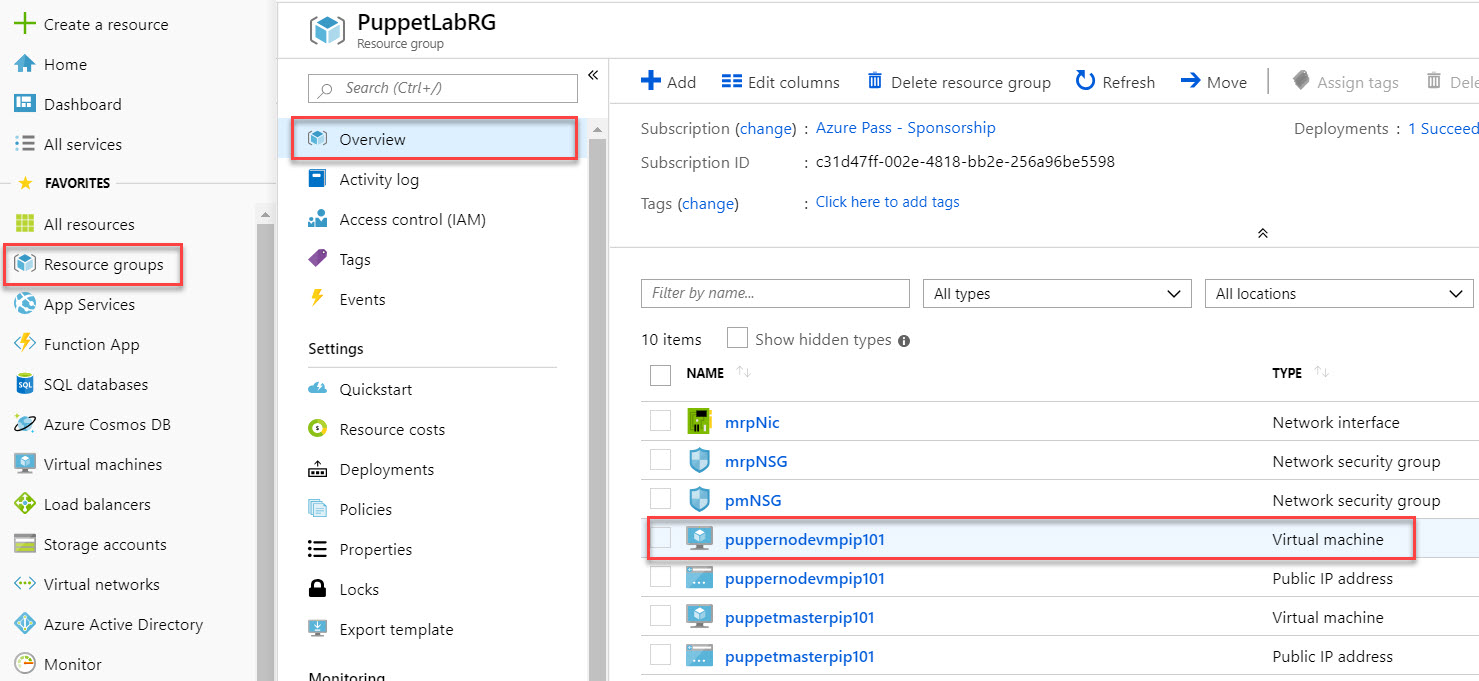
Step 7: Click on **Nodes -> Unsigned Certificates**

To Add node run below command

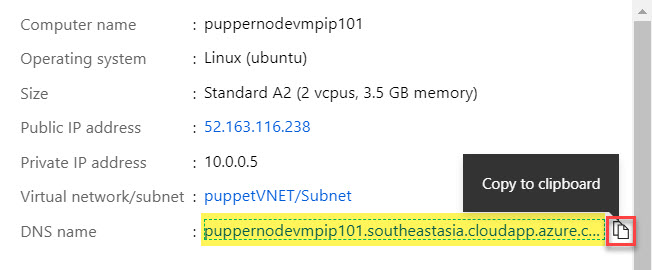


Step 8: Navigate to Azure Portal

**Resource Group -> Overview -> Select Node Virtual Machine**

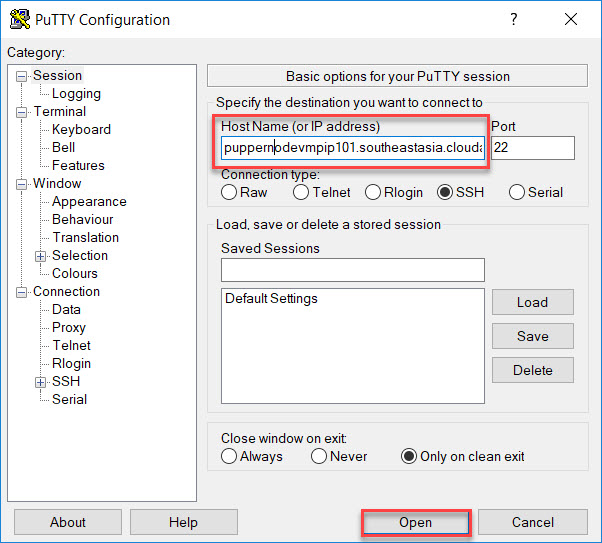


Click on **Copy** icon of DNS Name

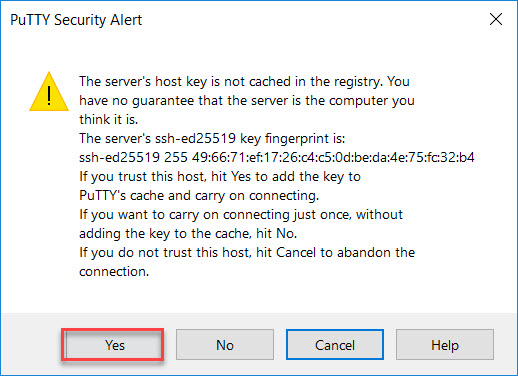


Step 9: Run PuTTy. If you don’t have download from here: <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>

Paste DNS Name into Hostname & Click on Open button.



Click on **Yes** button.



Step 10: Enter below credentials:

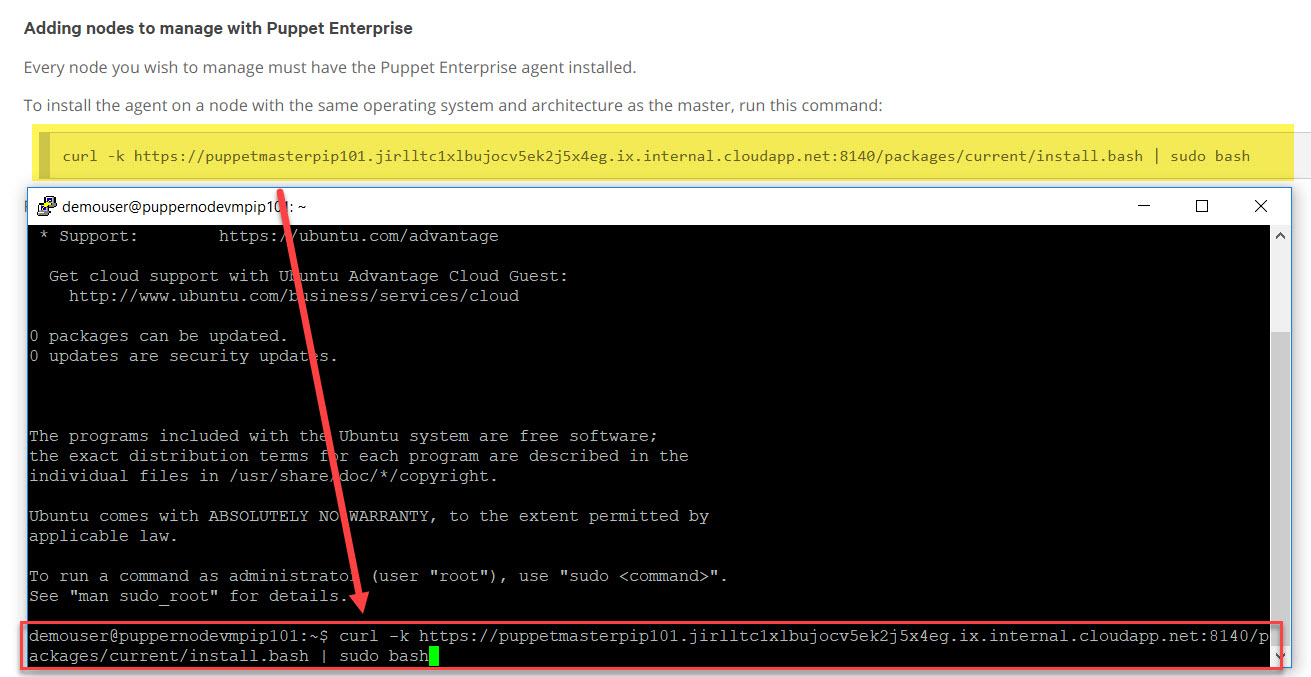
Username: **demouser**

Password: **Demo@pass123**



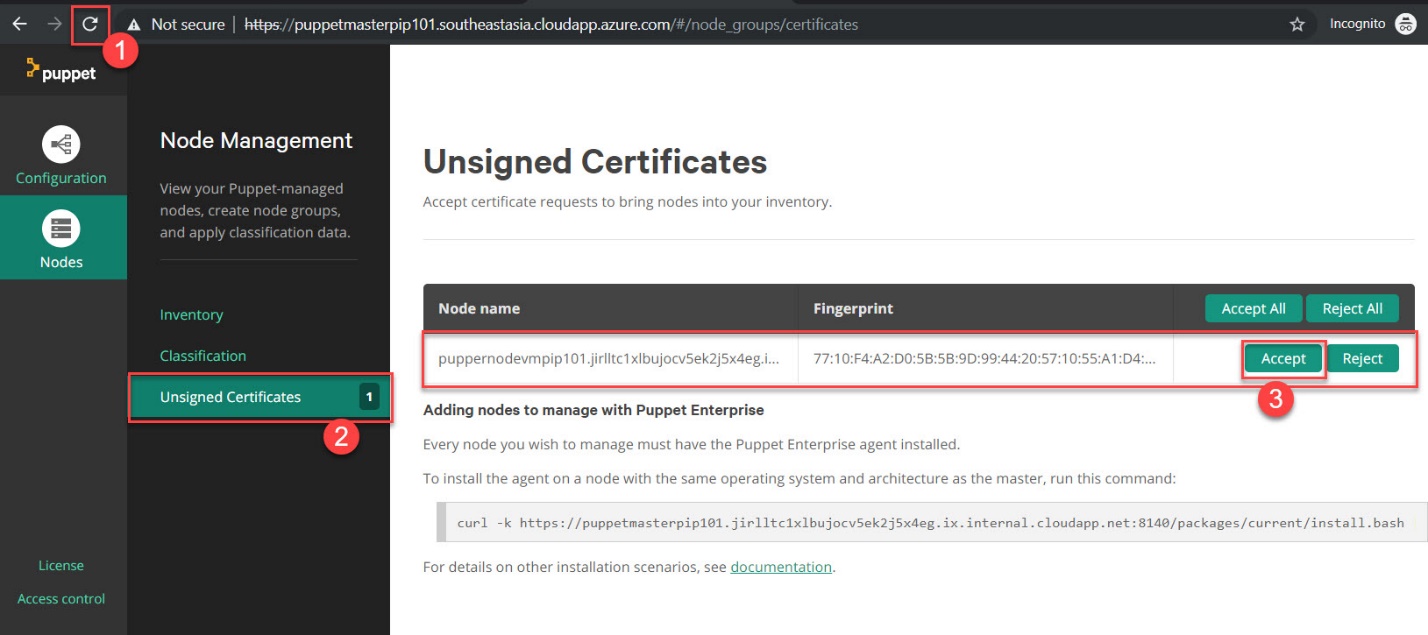
**Task 2) Install Puppet Agent on the node**

Copy curl command from Puppet Console and paste into **Node VM console**



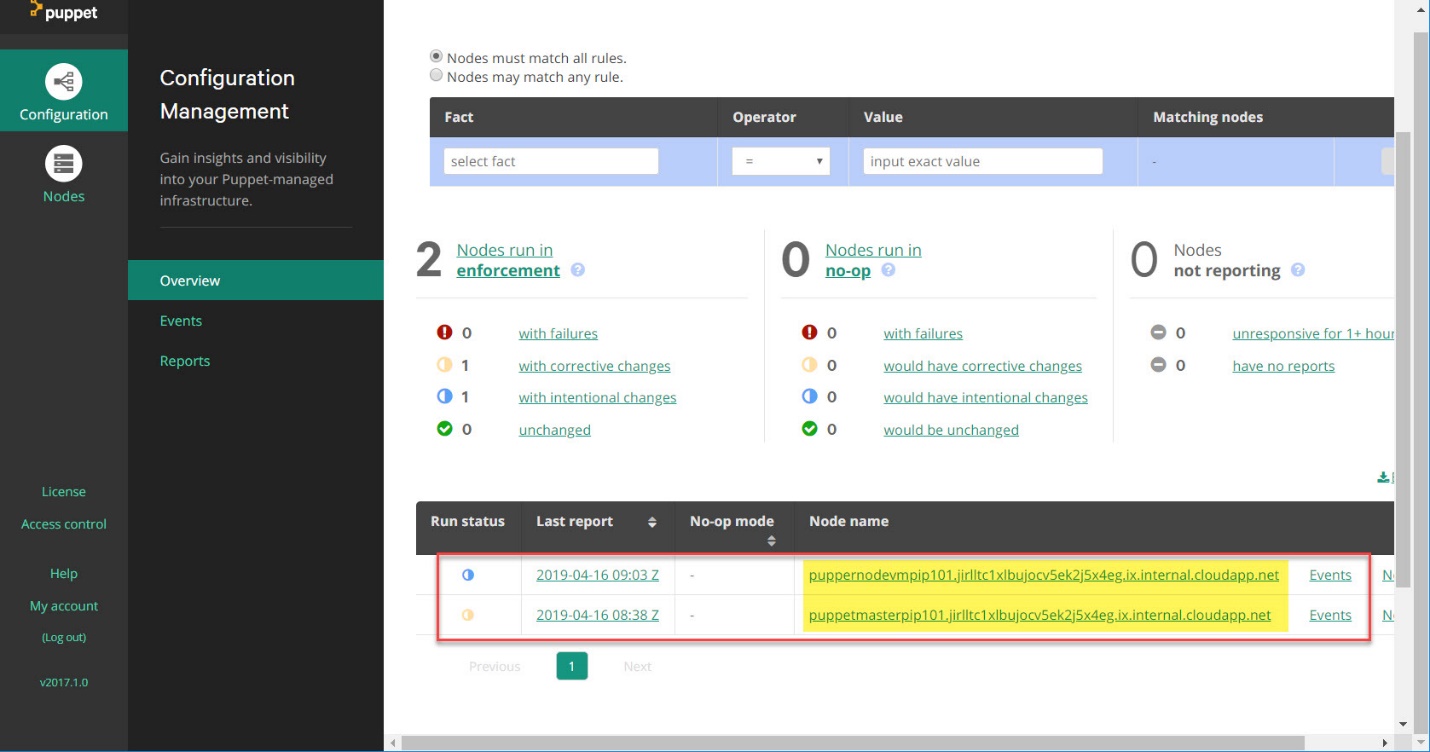
Step 11: After executing command refresh Puppet Console and check Unsigned Certificates – 1 available.

Click on **Accept** button.

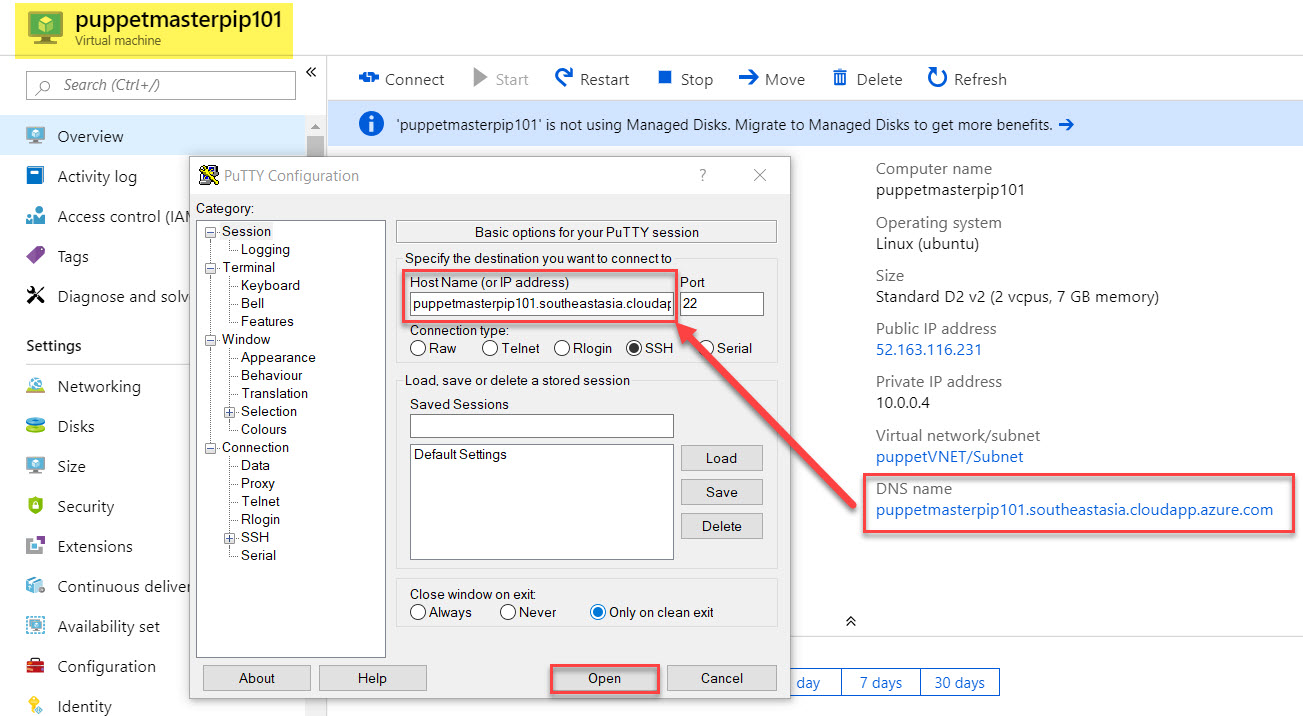


Step 12: Wait for few minutes and click on **Configuration -> Overview -> 2 Nodes running**

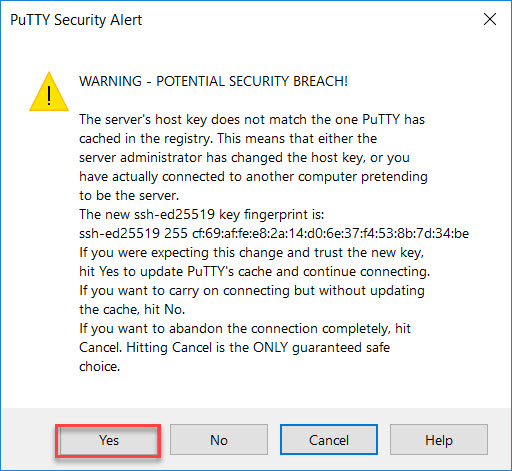
Master & Node



Step 13: Start another **PuTTy** and paste **Puppet Master DNS name**



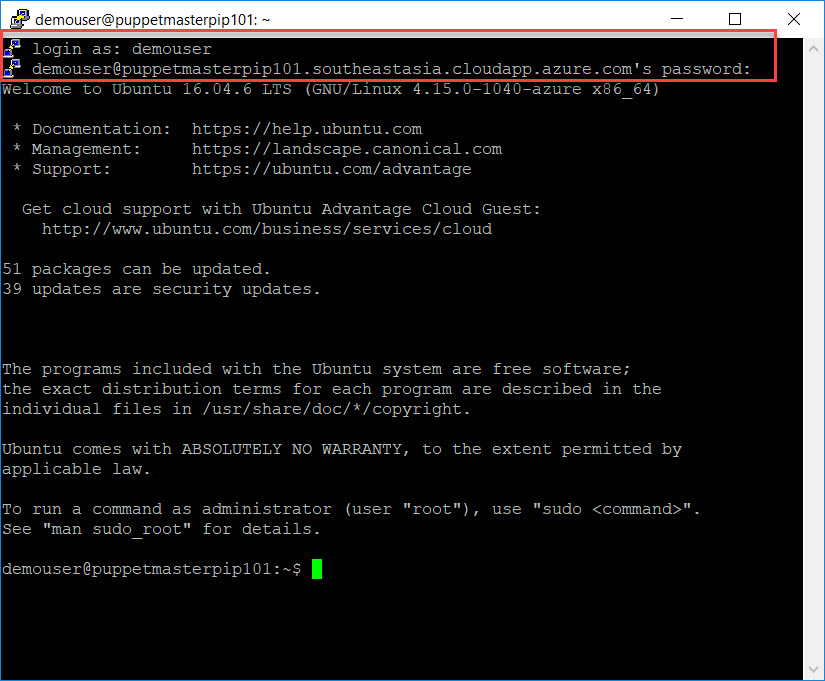
Click on **Yes**



Step 14: Enter Credentials

Username: **demouser**

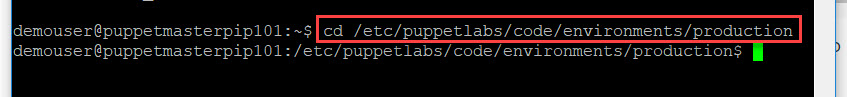
Password: **Demo@pass123**



**Task 3) Configure the Puppet Production Environment**

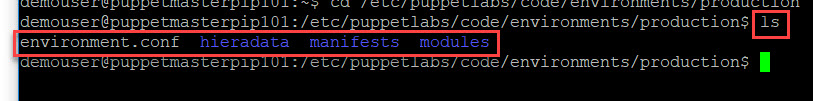
Step 15: Inspect the Production modules

cd /etc/puppetlabs/code/environments/production



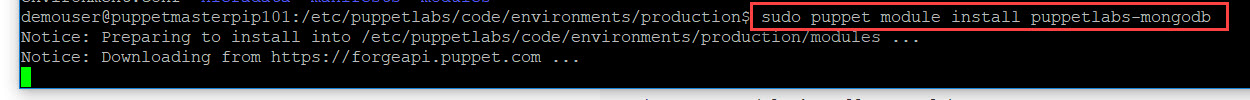
Step 16: List the content

ls



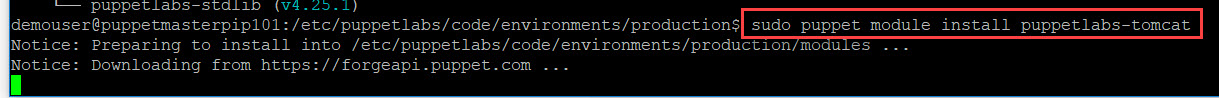
Step 17: Install **MongoDB** module

sudo puppet module install puppetlabs-mongodb



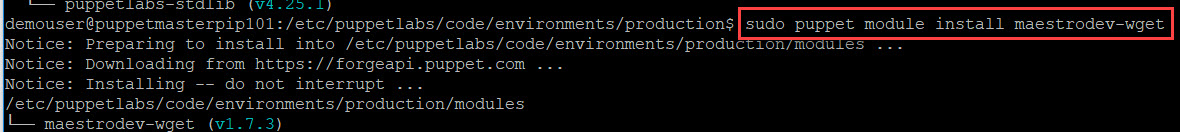
Step 18: Install **Tomcat** module

sudo puppet module install puppetlabs-tomcat



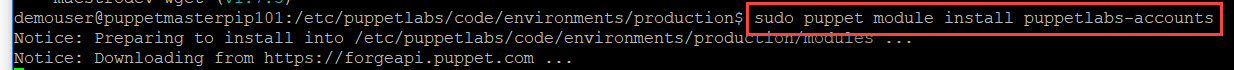
Step 19: Install **wget** module

sudo puppet module install maestrodev-wget



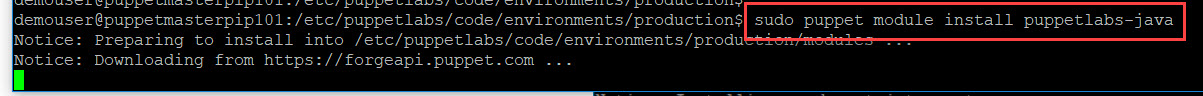
Step 20: Install **account** module

sudo puppet module install puppetlabs-accounts



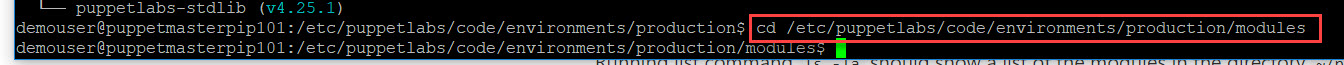
Step 21: Install **java** module

sudo puppet module install puppetlabs-java



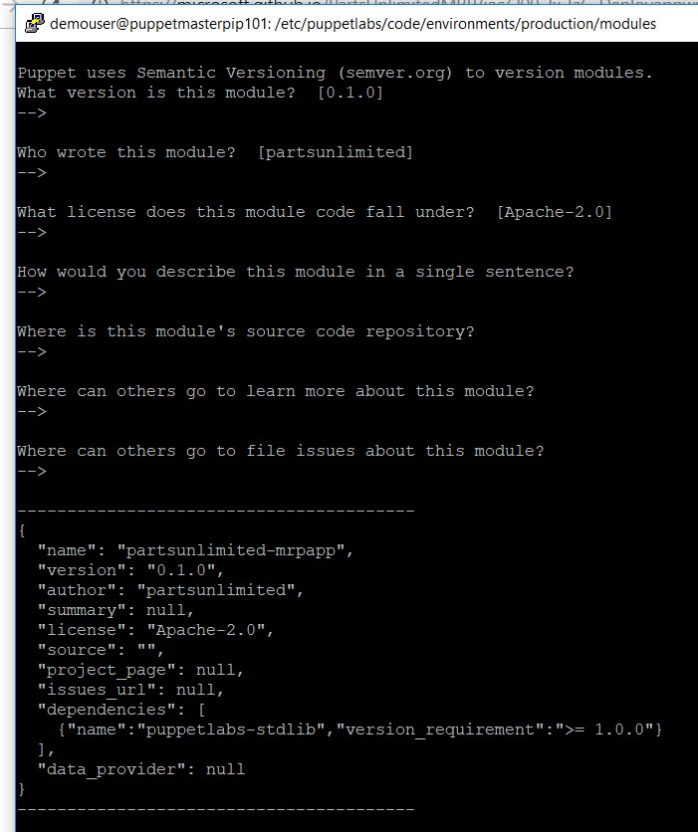
Step 22: Create Custom Module in the production

cd /etc/puppetlabs/code/environments/production/modules



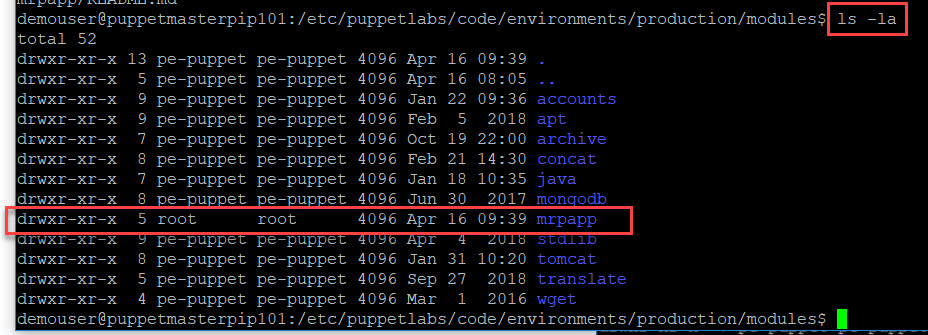
Step 23: create mrpapp run module generate command

sudo puppet module generate partsunlimited-mrpapp



Step 24: To list long list format & show hidden files

ls -la

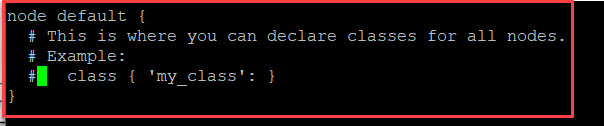


Step 25:

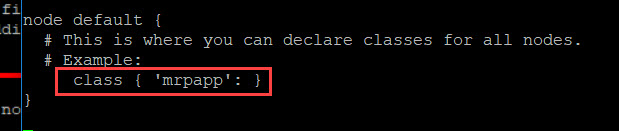
sudo nano /etc/puppetlabs/code/environments/production/manifests/site.pp



Step 26: Scroll down to node default



Remove # in class line and change ‘my\_class’ to ‘**mrpapp’**



**Task 4) Test the Production Environment Configuration**

Step 27:

sudo nano /etc/puppetlabs/code/environments/production/modules/mrpapp/manifests/init.pp



Step 28:

class mrpapp {

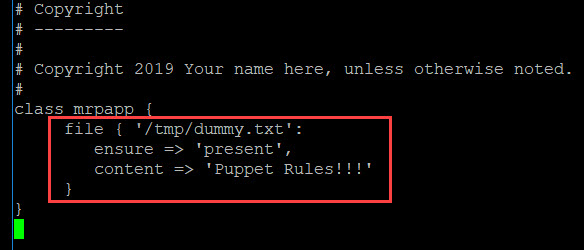
file { '/tmp/dummy.txt':

ensure => 'present',

content => 'Puppet rules!'

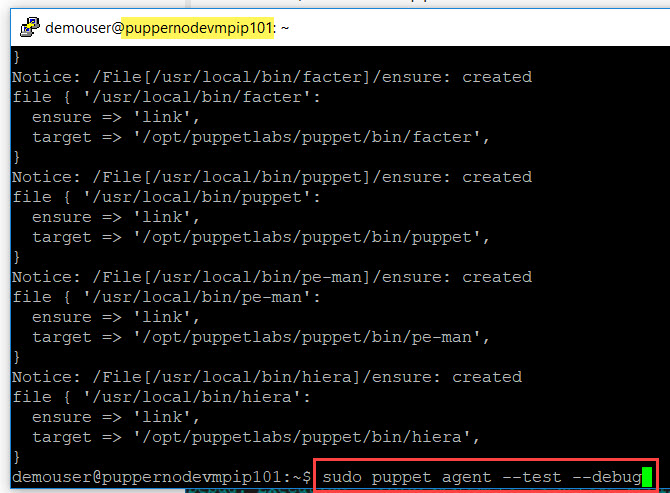
}

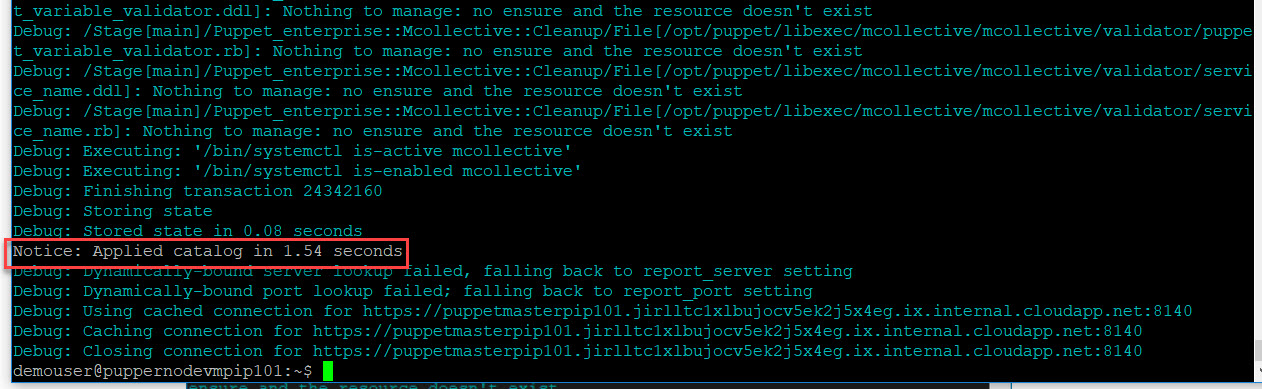
}



Step 29: Navigate to **Puppet Node VM**

sudo puppet agent --test --debug





Step 30:

cat /tmp/dummy.txt



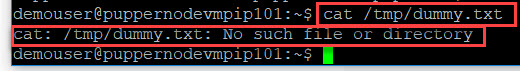
Step 31: Delete file

sudo rm /tmp/dummy.txt



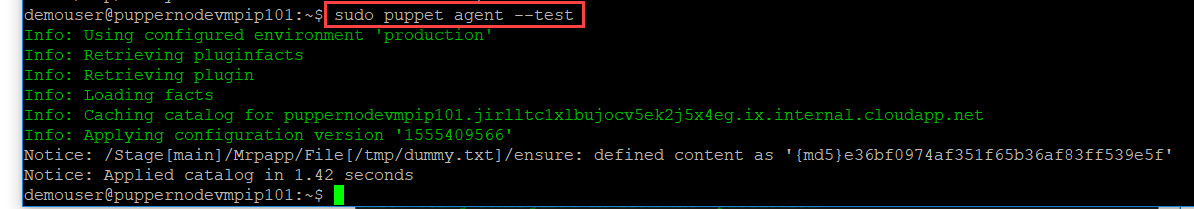
Step 32: type once again

cat /tmp/dummy.txt



Step 33:

sudo puppet agent --test



Step 34:

cat /tmp/dummy.txt



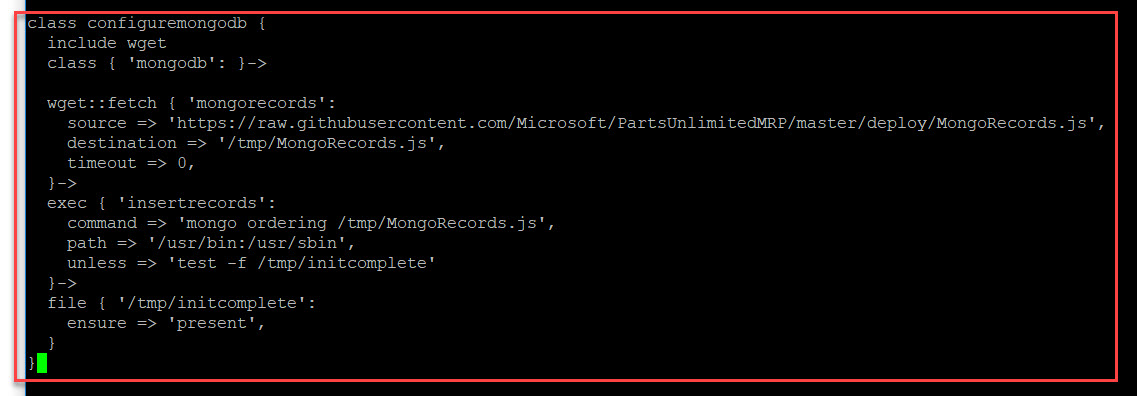
Step 35:

sudo nano /etc/puppetlabs/code/environments/production/modules/mrpapp/manifests/init.pp



**Task 5) Create a Puppet Program**

Step 36: Once MongoDB is configured, we want Puppet to download a Mongo script that contains data for our PU MRP application’s database. We will include this as part of our MongoDB setup. To implement these requirements, add a Class to configure MongoDB.



class configuremongodb {

include wget

class { 'mongodb': }->

wget::fetch { 'mongorecords':

source => 'https://raw.githubusercontent.com/Microsoft/PartsUnlimitedMRP/master/deploy/MongoRecords.js',

destination => '/tmp/MongoRecords.js',

timeout => 0,

}->

exec { 'insertrecords':

command => 'mongo ordering /tmp/MongoRecords.js',

path => '/usr/bin:/usr/sbin',

unless => 'test -f /tmp/initcomplete'

}->

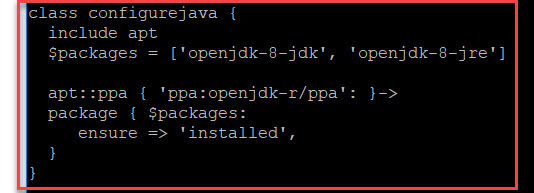
file { '/tmp/initcomplete':

ensure => 'present',

}

}

Step 37: Configure Java



class configurejava {

include apt

$packages = ['openjdk-8-jdk', 'openjdk-8-jre']

apt::ppa { 'ppa:openjdk-r/ppa': }->

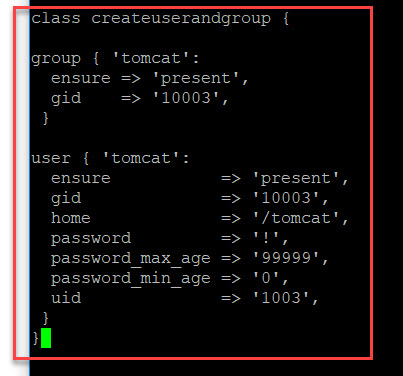
package { $packages:

ensure => 'installed',

}

}

Step 38: Create User & Group



class createuserandgroup {

group { 'tomcat':

ensure => 'present',

gid => '10003',

}

user { 'tomcat':

ensure => 'present',

gid => '10003',

home => '/tomcat',

password => '!',

password\_max\_age => '99999',

password\_min\_age => '0',

uid => '1003',

}

}

Step 39: Configure Tomcat



class configuretomcat {

class { 'tomcat': }

require createuserandgroup

tomcat::instance { 'default':

catalina\_home => '/var/lib/tomcat7',

install\_from\_source => false,

package\_name => ['tomcat7','tomcat7-admin'],

}->

tomcat::config::server::tomcat\_users {

'tomcat':

catalina\_base => '/var/lib/tomcat7',

element => 'user',

password => 'password',

roles => ['manager-gui','manager-jmx','manager-script','manager-status'];

'tomcat7':

catalina\_base => '/var/lib/tomcat7',

element => 'user',

password => 'password',

roles => ['manager-gui','manager-jmx','manager-script','manager-status'];

}->

tomcat::config::server::connector { 'tomcat7-http':

catalina\_base => '/var/lib/tomcat7',

port => '9080',

protocol => 'HTTP/1.1',

connector\_ensure => 'present',

server\_config => '/etc/tomcat7/server.xml',

}->

tomcat::service { 'default':

use\_jsvc => false,

use\_init => true,

service\_name => 'tomcat7',

}

}

Step 40: Deploy WAR file



class deploywar {

require configuretomcat

tomcat::war { 'mrp.war':

catalina\_base => '/var/lib/tomcat7',

war\_source => 'https://raw.githubusercontent.com/Microsoft/PartsUnlimitedMRP/master/builds/mrp.war',

}

file { '/var/lib/tomcat7/webapps/':

path => '/var/lib/tomcat7/webapps/',

ensure => 'directory',

recurse => 'true',

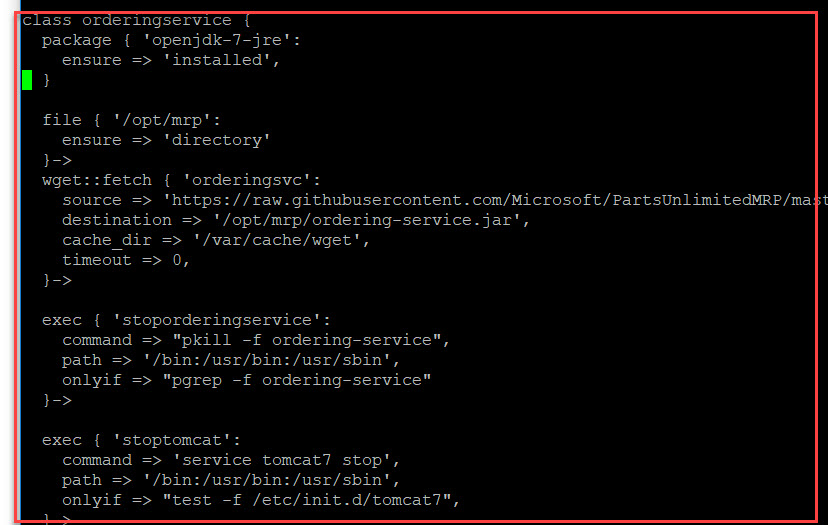
mode => '777',

}

}

Step 41: Start the Ordering Service

It is a REST API that manages orders in the Mongo DB. The Ordering Service is compiled to a .jar file. We need to copy the .jar file file to our Node. The Node then runs the Ordering Service, in the background, so that it can listen for REST API requests.



class orderingservice {

package { 'openjdk-7-jre':

ensure => 'installed',

}

file { '/opt/mrp':

ensure => 'directory'

}->

wget::fetch { 'orderingsvc':

source => 'https://raw.githubusercontent.com/Microsoft/PartsUnlimitedMRP/master/builds/ordering-service-0.1.0.jar',

destination => '/opt/mrp/ordering-service.jar',

cache\_dir => '/var/cache/wget',

timeout => 0,

}->

exec { 'stoporderingservice':

command => "pkill -f ordering-service",

path => '/bin:/usr/bin:/usr/sbin',

onlyif => "pgrep -f ordering-service"

}->

exec { 'stoptomcat':

command => 'service tomcat7 stop',

path => '/bin:/usr/bin:/usr/sbin',

onlyif => "test -f /etc/init.d/tomcat7",

}->

exec { 'orderservice':

command => 'java -jar /opt/mrp/ordering-service.jar &',

path => '/usr/bin:/usr/sbin:/usr/lib/jvm/java-8-openjdk-amd64/bin',

}->

exec { 'wait':

command => 'sleep 20',

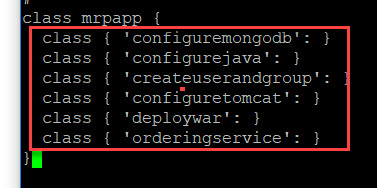
path => '/bin',

notify => Tomcat::Service['default']

}

}

Step 42: Go to **mrpapp** class and **delete dummy.txt**



class mrpapp {

class { 'configuremongodb': }

class { 'configurejava': }

class { 'createuserandgroup': }

class { 'configuretomcat': }

class { 'deploywar': }

class { 'orderingservice': }

}

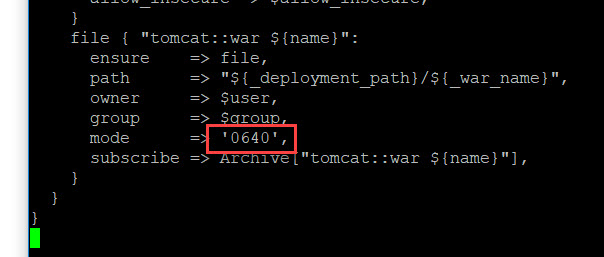
Step 43: Configure .war file extraction permissions

Tomcat automatically extracts .war files from the webapps directory. However, in our lab we will set full read, write and access permissions in the file war.pp, to ensure that the file extracts sucessfully.

sudo nano /etc/puppetlabs/code/environments/production/modules/tomcat/manifests/war.pp



Step 44: Change **mode => ‘0640’** to **mode => ‘0777’**



file { "tomcat::war ${name}":

ensure => file,

path => "${\_deployment\_path}/${\_war\_name}",

owner => $user,

group => $group,

mode => '0777',

subscribe => Archive["tomcat::war ${name}"],

}

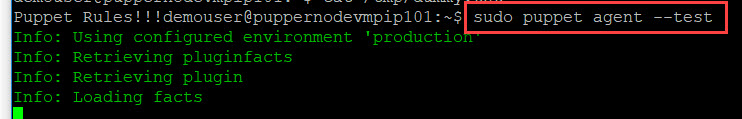
}

}

**Task 6) Run the Puppet Configuration on the Node**

Step 45: Run once again

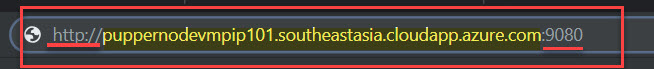
sudo puppet agent --test



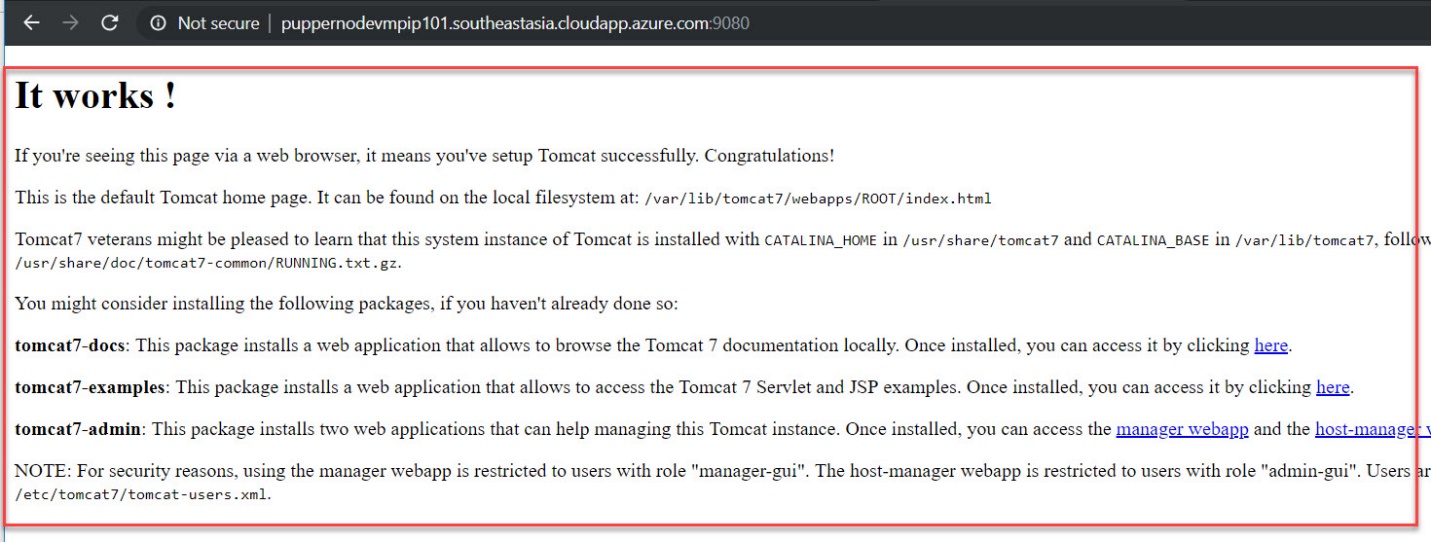
Wait for complete all process



Step 46: type <http://nodevm>-address:9080

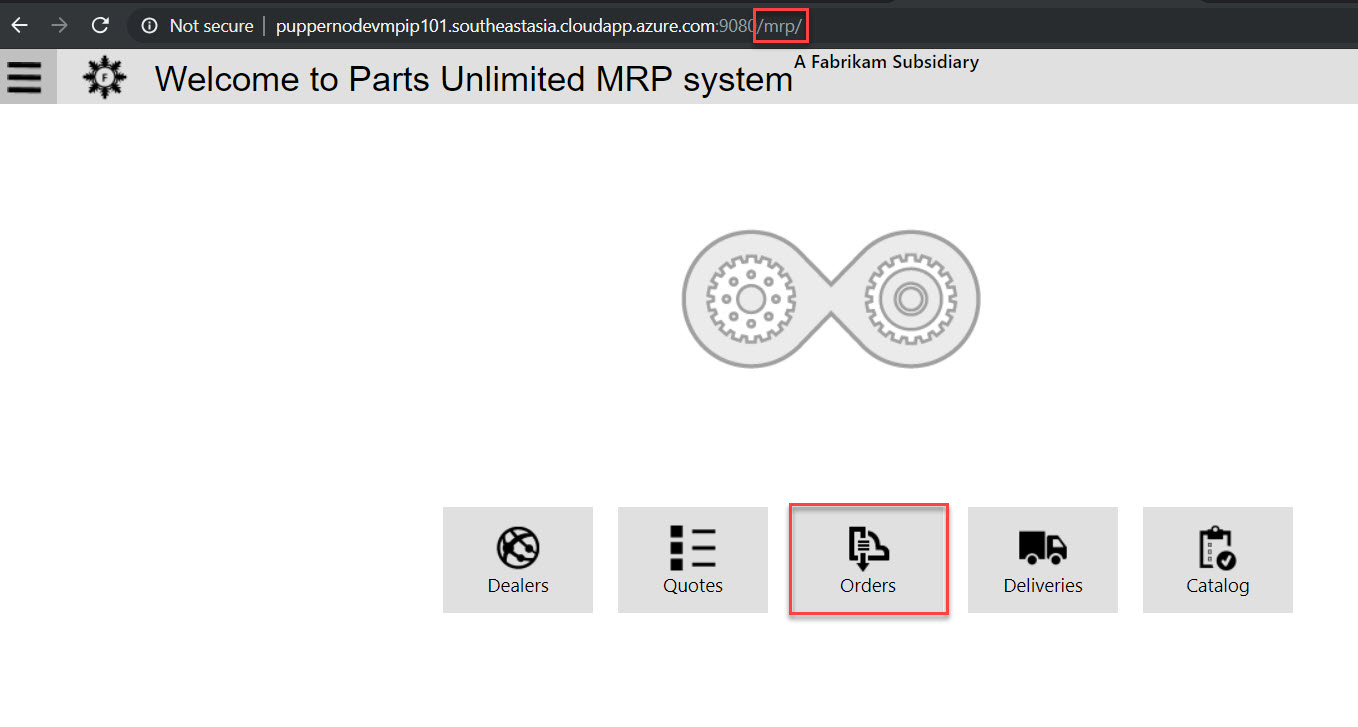


Tomcat Sample page will execute



Step 47: add /mrp/ after 9080 port

Click on **Orders** option



Order details available

