Installation of VirtualBox

Download Ubuntu

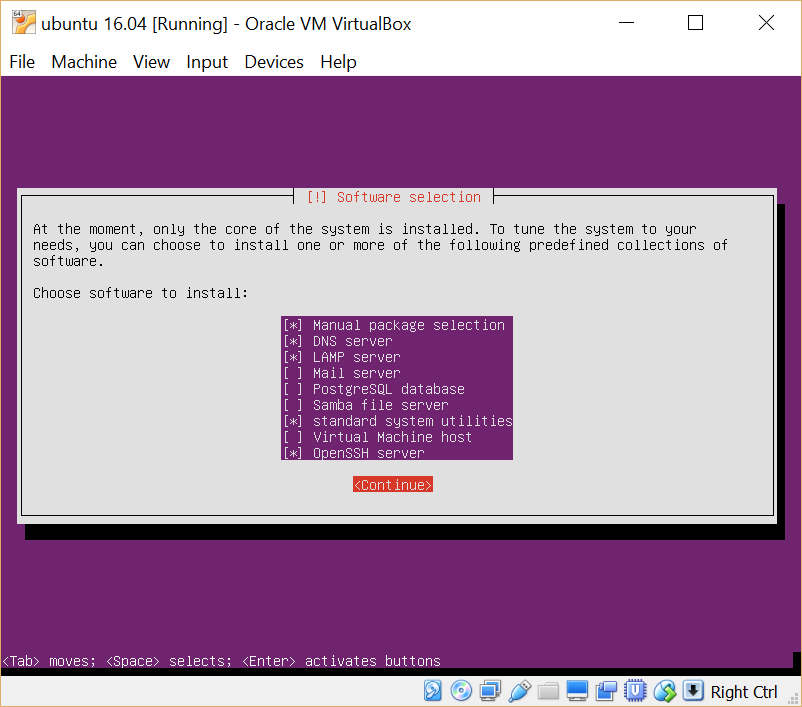
http://releases.ubuntu.com/16.04.3/ubuntu-16.04.3-server-amd64.iso?\_ga=2.1221873.114368501.1505156820-1577006882.1500922026

Installation of Ubuntu

To add the desktop Icon editor need to install

***apt-get install --no-install-recommends gnome-panel***

Package Selection



Post installation

To add a graphical desktop

sudo apt-get update

sudo apt-get install ubuntu-desktop

Need docker group added to the OS.

Add ons

groupadd docker

Create a metrolab user account

Add the following line to the /etc/sudoers file so you can execute root commands without having to enter a password.

Puppet Installation

Requirements

* Ubuntu 14.04 LTS “Trusty Tahr”
* Ubuntu 12.04 LTS “Precise Pangolin”
* Download the “puppetlabs-release” package for your OS version.
  + You can see a full list of these packages on the front page of <https://apt.puppetlabs.com/>. They are all named puppetlabs-release-<CODE NAME>.deb. (For Ubuntu releases, the code name is the adjective, not the animal.)
  + Architecture is handled automatically; there is only one package per OS version.
* Install the package by running dpkg -i <PACKAGE NAME>.
* Run apt-get update to get the new list of available packages.

For example, to enable the repository for Ubuntu 12.04 Precise Pangolin:

#wget <https://apt.puppetlabs.com/puppetlabs-release-precise.deb>

In our case it is

wget <https://apt.puppetlabs.com/puppet5-release-xenial.deb>

sudo dpkg -i puppet5-release-xenial.deb

sudo apt-get update

**Step 2: Install Puppet on the Puppet Master Server**

(Skip this step for a standalone deployment.)

On your Puppet master, run:

sudo apt-get install puppetserver

**Step 3: Install Puppet on Agent Nodes**

On your Puppet agents, run:

sudo apt-get install puppet

# Installing Puppet: Post-Install Tasks

In the [main] section of the master’s [puppet.conf](https://docs.puppet.com/puppet/3.8/config_file_main.html) file, set [the dns\_alt\_names setting](https://docs.puppet.com/puppet/3.8/configuration.html#dnsaltnames) to a comma-separated list of each hostname the master should be allowed to use:

dns\_alt\_names = puppet,puppet.example.com,puppetmaster01,puppetmaster01.example.com

For our purpose we will add puppet to the /etc/host file

*127.0.0.1 localhost*

*192.168.142.10 PupetMaster.corp.emc.com puppet PupetMaster*

### Get the Master’s Names and Certificates Set Up

When you create the puppet master’s certificate, you must include every DNS name at which agent nodes might try to contact the master.

Decide on a main name for Puppet services at your site, and make sure your DNS resolves it to the puppet master (or its load balancer). Unconfigured agents will try to find a master at puppet, so if you use this name it can reduce setup time.

In the [main] section of the master’s [puppet.conf](https://docs.puppet.com/puppet/3.8/config_file_main.html) file, set [the dns\_alt\_names setting](https://docs.puppet.com/puppet/3.8/configuration.html#dnsaltnames) to a comma-separated list of each hostname the master should be allowed to use:

dns\_alt\_names = puppet,puppet.emc.co,PuppetMaster.corp.emc.com,PuppetMaster

**For CA Masters**

If this is the only puppet master in your deployment, or if it will be acting as the CA server for a multi-master site, you should now run:

$ sudo puppet master --verbose --no-daemonize

This will create the CA certificate and the puppet master certificate, with the appropriate DNS names included. Once it says Notice: Starting Puppet master version <VERSION>, type ctrl-C to kill the process.

Puppet Site Config file –

/etc/puppet/manifests

Entry for installing Docker

exec {

'install\_Docker':

#To add by download

command => 'groupadd docker; apt-get update;curl https://download.docker.com/linux/ubuntu/dists/xenial/pool/stable/amd64/docker-ce\_17.03.2~ce-0~ubuntu-xenial\_amd64.deb > /tmp/docker-$(lsb\_release -cs).deb; dpkg -i /tmp/docker-$(lsb\_release -cs).deb; systemctl start docker.service; docker run hello-world > /tmp/hello.out',

#To add by http repository

#groupadd docker; apt-get -y install apt-transport-https ca-certificates curl software-properties-common; curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add - ;add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu xenial stable";apt-get update; apt-get -y install docker-ce; dockerd & sleep 5; docker run hello-world',

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

More About Puppet Configuration

<https://askubuntu.com/questions/61080/how-to-copy-file-to-all-clients-using-puppet>

Used to get files over to client

<https://www.digitalocean.com/community/tutorials/getting-started-with-puppet-code-manifests-and-modules>

https://www.digitalocean.com/community/tutorials/configuration-management-101-writing-puppet-manifests

**Class Name Restrictions**

https://puppet.com/docs/pdk/1.0/pdk\_reference.html

|  |  |  |
| --- | --- | --- |
| <class\_name> | **Required**. The name of the class to generate. | A class name beginning with a lowercase letter and including only lowercase letters, digits, and underscores. |

Puppet Cookbook - <https://www.puppetcookbook.com/posts/run-a-basic-docker-container.html>

When using variables in the exec command, need to use double quotes “ and not single ‘

exec { 'apache\_build':

#command =>'docker build -t my\_apache $apache\_root'

command => "/bin/echo ${apache\_root} > /tmp/apache\_root.out",

}

Puppet configurations ***/etc/puppet/puppet.conf***

[main]

logdir=/var/log/puppet

vardir=/var/lib/puppet

ssldir=/var/lib/puppet/ssl

rundir=/run/puppet

factpath=$vardir/lib/facter

prerun\_command=/etc/puppet/etckeeper-commit-pre

postrun\_command=/etc/puppet/etckeeper-commit-post

#Added these options

**runinterval = 60m**

**daemonize = false**

However, by running the

puppet agent –t

will manually applied the puppet configuration

# Puppet control Commands use to control the Puppet services

<https://askubuntu.com/questions/19320/how-to-enable-or-disable-services>

**SYSTEMD**

Starting with Ubuntu 15.04, Upstart will be deprecated in favor of Systemd. With Systemd to manage the services we can do the following:

systemctl start SERVICE - Use it to start a service. Does not persist after reboot

systemctl stop SERVICE - Use it to stop a service. Does not persist after reboot

systemctl restart SERVICE - Use it to restart a service

systemctl reload SERVICE - If the service supports it, it will reload the config files related to it without interrupting any process that is using the service.

systemctl status SERVICE - Shows the status of a service. Tells whether a service is currently running.

**rm**

systemctl disable SERVICE - Turns the service off on the next reboot or on the next stop event. It persists after reboot.

systemctl is-enabled SERVICE - Check if a service is currently configured to start or not on the next reboot.

systemctl is-active SERVICE - Check if a service is currently active.

systemctl show SERVICE - Show all the information about the service.

sudo systemctl mask SERVICE - Completely disable a service by linking it to /dev/null; you cannot start the service manually or enable the service.

sudo systemctl unmask SERVICE - Removes the link to /dev/null and restores the ability to enable and or manually start the service.

## GIT

Git is already installed on with Ubuntu

For the purpose of this these exercise, we are going to use the Client VM as the developer workstation. This is where all of the files will be created and pushed to the source code repository (GitHub). The Master VM will pull the files Puppet configuration files and place them in the appropriate Puppet directories.

For the application thread of the exercise, Jenkins will pull the code for the application portion of the exercise.

Create a directory for the GIT repository

$ cd /opt

$ mkdir MetroLab

$ cd MetoLab

$ git init # Initialize the git repository

$

# Building Web Server Container

# (https://writing.pupius.co.uk/apache-and-php-on-docker-44faef716150)

#### Setting up the project

Set up a project with the following structure:

(I have on client in /var/src/docker\_src/apache2)

./Dockerfile  
./apache-config.conf  
./www/index.php

Dockerfile:

*FROM ubuntu:latest*

*2 MAINTAINER Dan Pupius <dan@pupi.us>*

*3*

*4 # Install apache, PHP, and supplimentary programs. openssh-server, curl, and lynx-cur are for debugging the container.*

*5 RUN apt-get update && apt-get -y upgrade && DEBIAN\_FRONTEND=noninteractive apt-get -y install \*

*6 apache2 php7.0 php7.0-mysql libapache2-mod-php7.0 curl lynx-cur*

*7*

*8 # Enable apache mods.*

*9 RUN a2enmod php7.0*

*10 RUN a2enmod rewrite*

*11*

*12 # Update the PHP.ini file, enable <? ?> tags and quieten logging.*

*13 RUN sed -i "s/short\_open\_tag = Off/short\_open\_tag = On/" /etc/php/7.0/apache2/php.ini*

*14 RUN sed -i "s/error\_reporting = .\*$/error\_reporting = E\_ERROR | E\_WARNING | E\_PARSE/" /etc/php/7.0/apache2/php.ini*

*15*

*16 # Manually set up the apache environment variables*

*17 ENV APACHE\_RUN\_USER www-data*

*18 ENV APACHE\_RUN\_GROUP www-data*

*19 ENV APACHE\_LOG\_DIR /var/log/apache2*

*20 ENV APACHE\_LOCK\_DIR /var/lock/apache2*

*21 ENV APACHE\_PID\_FILE /var/run/apache2.pid*

*22*

*23 # Expose apache.*

*24 EXPOSE 80*

*25*

*26 # Copy this repo into place.*

*27 ADD www /var/www/site*

*28*

*29 # Update the default apache site with the config we created.*

*30 ADD apache-config.conf /etc/apache2/sites-enabled/000-default.conf*

*31*

*32 # By default start up apache in the foreground, override with /bin/bash for interative.*

*33 CMD /usr/sbin/apache2ctl -D FOREGROUND*

apache-config.conf:

*<VirtualHost \*:80>*

*2 ServerAdmin me@mydomain.com*

*3 DocumentRoot /var/www/site*

*4*

*5 <Directory /var/www/site/>*

*6 Options Indexes FollowSymLinks MultiViews*

*7 AllowOverride All*

*8 Order deny,allow*

*9 Allow from all*

*10 </Directory>*

*11*

*12 ErrorLog ${APACHE\_LOG\_DIR}/error.log*

*13 CustomLog ${APACHE\_LOG\_DIR}/access.log combined*

*14*

15 </VirtualHost>

www/index.php:

<? echo "<p>Hello?</p>"; ?>

#### Building the container

Once you have setup the project run:

*docker build -t mysite ./*

#### Getting it running

To run apache in a background process, simply start the container using the following command:

docker run -p 8080:80 -d mysite

-p 8080:80 publishes port 80 in the container to 8080 on the host machine.  
-d detaches from the process, use docker ps and docker stop to … stop.

Note: on OSX use `docker-machine ip default` to get the right IP to use (assuming default is your machine name).

Sometimes you’ll want to debug issues with the container; maybe there are PHP configuration issues or you want to view error logs. To do that you can start the container in interactive mode and then start apache manually:

docker run -i -t -p 8080:80 mysite /bin/bash  
apachectl start

#### Making changes

If you’re actively developing you want to be able to change files in your usual editor and have them reflected within the container without having to rebuild it. The -v flag allows us to mount a directory from the host into the container:

docker run -p 8080:80 –name=apache -d-v `pwd`/www:/var/www/site mysite

To Stop the container

docker stop apache

**Installing Jenkins**

wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | sudo apt-key add -

sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'

sudo apt-get update

sudo apt-get install Jenkins

**Access to Jenkins server**

http://127.0.0.1:8080/

ID: admin

PWD: M3tr0Lab

**Configure Jenkins with GitHub**

Created a new GitHub Account

ID: DemoMetroLab

PWD : S3r…e

Repository is MetroLab1

On Jenkins Server

* If Jenkins does run as different user, set this up on your server
* Set git user.name and user.email global config options.
* 'git config --global user.email JENKINS\_USERNAME@WHATEVER\_HOSTNAME'
* 'git config --global user.name JENKINS\_USERNAME'
* This should match the GitHub utility account username and email address

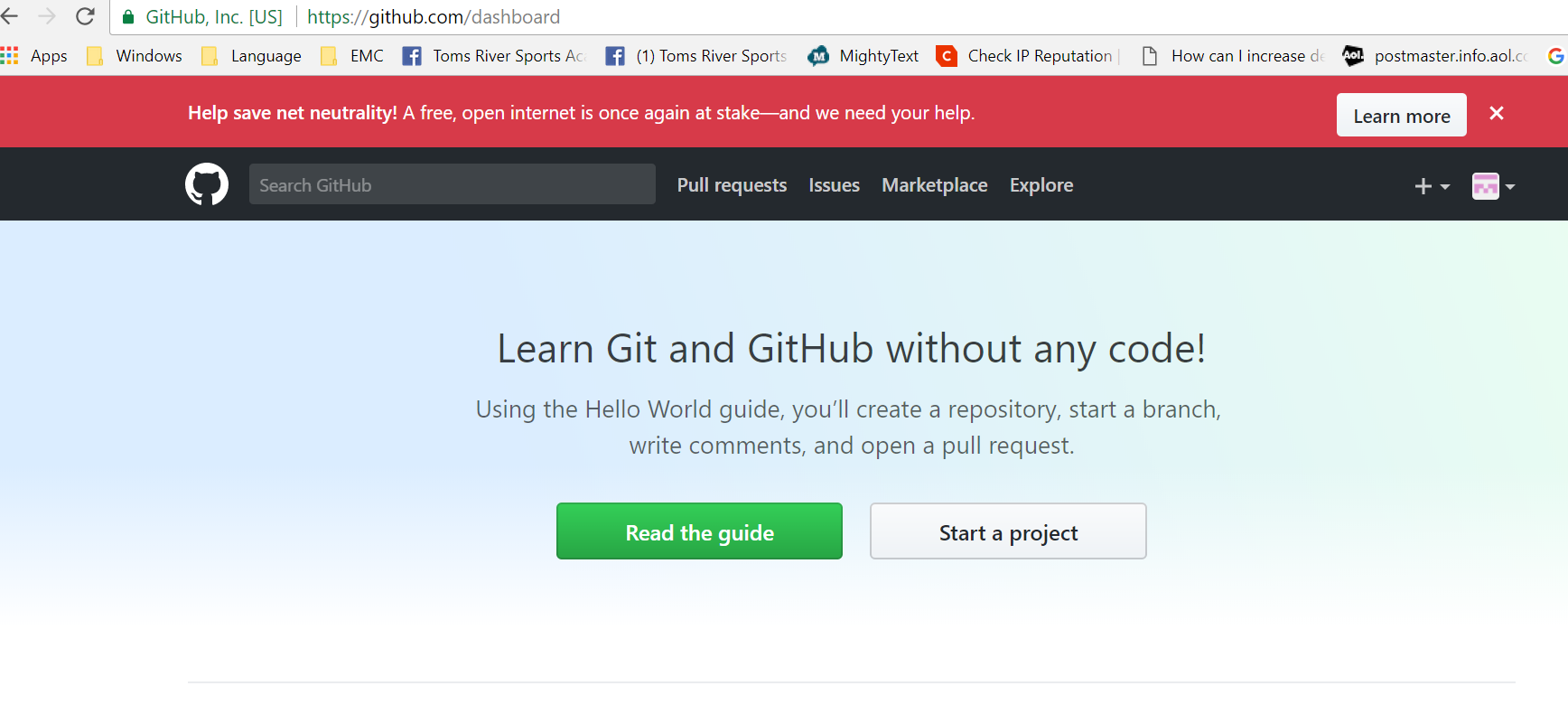
## GIT

Git is already installed on with Ubuntu

For the purpose of this these exercise, we are going to use the Client VM as the developer workstation. This is where all of the files will be created and pushed to the source code repository (GitHub). The Master VM will pull the files Puppet configuration files and place them in the appropriate Puppet directories.

For the application thread of the exercise, Jenkins will pull the code for the application portion of the exercise.

Signup for GitHub



Create a directory for the GIT repository

$ cd /opt

$ mkdir MetroLab

$ cd MetoLab

$ git init # Initialize the git repository

Copy the puppet site configuration file to your git project location.c

$git status

*metrolab@PuppetMaster:~/MetroLab1$ git status*

*On branch master*

*Your branch is up-to-date with 'origin/master'.*

*Untracked files:*

*(use "git add <file>..." to include in what will be committed)*

*site.pp*

*nothing added to commit but untracked files present (use "git add" to track)*

$git remote add origin https://github.com/username/myproject.git

The first part is familiar; we’ve used git add already with files. We’ve tacked the word origin onto it to indicate a new place from which files will originate. remote is a descriptor of origin, to indicate the origin is not on the computer, but somewhere online.

Git now knows there’s a remote repository and it’s where you want your local repository changes to go. To confirm, type this to check:

**Make a remote repository from the command line**

curl -u **'*USER\_NAME*'** https://api.github.com/user/repos -d "{\"name\":\"***NAME\_OF\_REPO***\"}"

Add the directory to the remote repository

git remote add origin "https://github.com/your\_github\_username/$repo\_name.git"

$git remote –v

*metrolab@PuppetMaster:~/MetroLab1$ git remote -v*

*origin https://github.com/DemoMetroLab/MetroLab1 (fetch)*

*origin https*://github.com/DemoMetroLab/MetroLab1 (push)

This command gives you a list of all the remote origins your local repository knows about. Assuming you’ve been with me so far, there should only be one, the DemoMetroLab one we just added. It’s listed twice, which means it is available to push information to, and to fetch information from

$ git add *{FileName}*

$ git commit *{FileName}*

$ git push

**Git user name and GitHub Identity**

root@PuppetMaster:/opt/MetroLab\_Docker# git commit -m "adding deploy script"

[master 9c17e2c] adding deploy script

Committer: root <root@PuppetMaster.corp.emc.com>

Your name and email address were configured automatically based

on your username and hostname. Please check that they are accurate.

You can suppress this message by setting them explicitly. Run the

following command and follow the instructions in your editor to edit

your configuration file:

git config --global --edit

After doing this, you may fix the identity used for this commit with:

git commit --amend --reset-author

Client ID metro PWD: password

Root PWD: password

Puppet

<https://www.digitalocean.com/community/tutorials/configuration-management-101-writing-puppet-manifests>

ScaleIO Installation

Downloaded ScaleIO from

Add the following to the /etc/sysctl.conf file

kernel.shmmax=209715200

Install these:

Packages required for all components:

**—numactl**

**—libaio**

Packages required for MDM components:

**—mutt (for Call-Home)**

**—bash-completion (for scli completion)**

**—Latest version of Python 2.X**