[Installing LAMP via puppet on Docker](http://www.bogotobogo.com/DevOps/Docker/Installing-LAMP-with-puppet-on-Docker.php)

Note

In this article, we'll install LAMP on Ubuntu 14.04 within Docker container. This is not much different from a simple install of LAMP via puppet ([puppet install on ubuntu 14.04](http://www.bogotobogo.com/DevOps/Puppet/puppet_install_on_ubuntu_14_04_trusty.php)) except our work is done within Docker. Only difference is just one command at the beginning:

$ docker run -it ubuntu:14.04

...

root@374a0b304efe:/#

Puppet with Amazon EC2 : Install LAMP with a Single Manifest

We'll develop a manifest on a Puppet agent node, and execute it via **puppet apply**.

In the process, we learn how to write a manifest using the following types of resource declarations:

1. **exec**: to execute commands, such as apt-get
2. **package**: to install packages via apt
3. **service**: to ensure that a service is running
4. **file**: to ensure that certain files exist

Installing puppet agent

Install puppet agent:

root@374a0b304efe:/# apt-get update

root@374a0b304efe:/# apt-get install puppet

Creating a Manifest

Puppet has been installed. Now, let's create a new manifest:

$ sudo vi /etc/puppet/manifests/lamp.pp

Add the following lines to declare the resources that we just determined we wanted. The inline comments detail each resource declaration:

# execute 'apt-get update'

exec { 'apt-update': # exec resource named 'apt-update'

command => '/usr/bin/apt-get update' # command this resource will run

}

# install apache2 package

package { 'apache2':

require => Exec['apt-update'], # require 'apt-update' before installing

ensure => installed,

}

# ensure apache2 service is running

service { 'apache2':

ensure => running,

}

# install mysql-server package

package { 'mysql-server':

require => Exec['apt-update'], # require 'apt-update' before installing

ensure => installed,

}

# ensure mysql service is running

service { 'mysql':

ensure => running,

}

# install php5 package

package { 'php5':

require => Exec['apt-update'], # require 'apt-update' before installing

ensure => installed,

}

# ensure info.php file exists

file { '/var/www/html/info.php':

ensure => file,

content => '<?php phpinfo(); ?>', # phpinfo code

require => Package['apache2'], # require 'apache2' package before creating

}

Applying the Manifest - "puppet apply"

Now we will use the **puppet apply** command to execute the manifest. On ip-172-31-25-70 node, run this:

root@374a0b304efe:/# puppet apply /etc/puppet/manifests/lamp.pp

Notice: Compiled catalog for 374a0b304efe.attlocal.net in environment production in 0.78 seconds

Notice: /Stage[main]/Main/Exec[apt-update]/returns: executed successfully

Notice: /Stage[main]/Main/Service[mysql]/ensure: ensure changed 'stopped' to 'running'

Notice: Finished catalog run in 13.69 seconds

The apache2 seems to be running fine:

root@374a0b304efe:/# ps -ef|grep apache2

root 5622 1 0 02:45 ? 00:00:00 /usr/sbin/apache2 -k start

www-data 5625 5622 0 02:45 ? 00:00:00 /usr/sbin/apache2 -k start

Creating a LAMP image

We successfully installed the LAMP, however, to use port 80, and see the web page, we need to set port forwarding. Let's close our Docker, and resume it with port option. First, we need to created the image we worked on using the image id, 374a0b304efe.

$ docker commit -m "LAMP added to Ubuntu14" -a "bogo" 374a0b304efe bogo-lamp

14c9538ef4e43f7fe82184e443cec0f4fa8fd528438d9dbc2ac354b95c39e0b4

Here we used the **docker commit** command. We specified two flags: **-m** and **-a**. The **-m** flag allows us to specify a commit message, much like we would with a commit on a version control system. The **-a** flag allows us to specify an author for our update.

We also specified the container we want to create this new image from, 374a0b304efe (the ID we recorded earlier) and we specified a target for the image:

bogo-lamp

Let's see what images we have:

$ docker images

REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE

bogo-lamp latest 14c9538ef4e4 About a minute ago 401 MB

ubuntu 14.04 91e54dfb1179 4 weeks ago 188.4 MB

Docker run with port forwarding

Then, resume our container with the image we created in previous section:

$ docker run -it -p 9999:80 bogo-lamp

What we did is forwarding port from host port 9999 to docker's port 80.

On docker, let's run apache2 server:

root@0a873afa3bde:/# service apache2 restart

\* Restarting web server apache2

