[Docker install via Puppet](http://www.bogotobogo.com/DevOps/Docker/Docker_puppet.php)

Note

In this article, we'll see how Puppet can automate the provisioning of Docker containers. We'll use a module from Puppet forge to install Docker. It provides a puppet class for installing Docker and two defined types for managing images and containers.

Puppet vs Docker

1. **Puppet**

It is a system configuration tool.

It enforces states of resources defined in a puppet manifest. A resource can be anything that has a state such as files, daemons, databases and other services.

The exact implementation of how to manipulate a resource and what are the possible states of a resource are defined in modules. It runs in a master-agent (server-client) architecture, where agents periodically check in at the master and ask for the manifest.

1. **Docker**

It is a container technology. It is NOT virtualization, all containers share the same kernel.

Containers are based on images. Images are pre-configured states of a container. Consider them as a tar.gz of a Linux installation, including all libraries and packages. When a process is executed inside a container, it will be using libraries and configuration inside the container. Containers are single use: once the process terminates it can't be re-executed in the same container. (Not after docker 1.3: it introduced docker exec. But the original philosophy was that containers are immutable.)

Images are layered. Basically after executing something in the container, the end-state of the container can be saved to a new image. In order to save disk space, not the whole container will be saved, but only the difference between the original image and the end-state of the container. Docker uses AUFS (Another Union File System) to simulate a full filesystem based on these layers.

-From [Will Docker or Puppet work?](https://www.linkedin.com/pulse/docker-puppet-work-eric-l)

How to combine Docker & Puppet

The two technologies have one overlapping area: **provisioning**.

Docker has something called the **Dockerfile** which is a set of instructions on how-to build an image.

Puppet is a provisioning tool by itself, doing the same state: putting a system into a state.

But while Puppet excels at precise **orchestration**, it's also very **slow to execute**.

Docker has **limited configuration power** compared to Puppet, but it's extremely **fast** to spin up a new container based on an existing image.

The idea is to use Puppet to provision a complete service into a Docker image, then use this image on the agents to start the services.

There are a lot of benefits with this approach:

1. **Time saver**: The puppet manifest that provisions a service is only required to run once, during image creation. The same image can be used in test, staging, production and development, lowering the diversity of environments.
2. **Portability**: the image can be used in any virtualization, in the cloud or on bare metal.
3. **Rollbacks** are always possible and are always simple: we can always spin up a previous image.

-From [Will Docker or Puppet work?](https://www.linkedin.com/pulse/docker-puppet-work-eric-l)

Puppetmaster set up

In this article, we assume puppet master has been already installed: [Puppet install on Ubuntu 14.04 LTS](http://www.bogotobogo.com/DevOps/Puppet/puppet_install_on_ubuntu_14_04_trusty.php) and [Puppet master post install tasks](http://www.bogotobogo.com/DevOps/Puppet/puppet_master_post_install_tasks_on_ubuntu_14_04_trusty.php).

garethr/docker from Puppet forge

Let's install [garethr/docker](https://forge.puppetlabs.com/garethr/docker) from Puppet forge. It's a module for installing and managing docker.

$ sudo puppet module install garethr-docker

Notice: Preparing to install into /etc/puppet/modules ...

Notice: Downloading from https://forgeapi.puppetlabs.com ...

Notice: Installing -- do not interrupt ...

/etc/puppet/modules

|--- garethr-docker (v4.1.1)

|-- puppetlabs-apt (v2.1.1)

|-- puppetlabs-stdlib (v4.9.0)

|-- stahnma-epel (v1.1.1)

docker\_example.pp

The **/etc/puppet/manifests/docker\_example.pp** should look like as shown below:

include 'docker'

docker::image { 'ubuntu':

image\_tag => 'trusty',

}

The module includes a single class:

include 'docker'

By default, this sets up the docker hosted repository if necessary for our OS, and installs the docker package and on Ubuntu, any required Kernel extensions.

docker::image { 'ubuntu':

image\_tag => 'trusty'

}

The image tags is equivalent to running **docker pull -t="trusty"** ubuntu. Note that the image will only install if an image of that name does not already exist.

manifest apply

Let's apply the puppet manifest (**/etc/puppet/manifests/docker\_example.pp**) in order to get docker installed on our puppet master:

$ sudo puppet apply site.pp

...

Notice: Finished catalog run in 46.40 seconds

Check if docker is installed

Let's check if docker has been installed:

$ sudo docker version

Client version: 1.7.1

Client API version: 1.19

Go version (client): go1.4.2

Git commit (client): 786b29d

OS/Arch (client): linux/amd64

Server version: 1.7.1

Server API version: 1.19

Go version (server): go1.4.2

Git commit (server): 786b29d

OS/Arch (server): linux/amd64

Also, we can check there is no running docker:

$ sudo docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

Check the installation of the ubuntu docker image:

$ sudo docker images

REPOSITORY TAG IMAGE ID CREATED VIRTUAL SIZE

ubuntu trusty 91e54dfb1179 3 weeks ago 188.4 MB

It's looking good, and we have an image to work with.

manifest update

We can launch containers:

docker::run { 'helloworld':

image => 'ubuntu',

command => '/bin/sh -c "while true; do echo hello world; sleep 1; done"',

}

which is equivalent to running the following:

docker run -d base /bin/sh -c "while true; do echo hello world; sleep 1; done"

This will launch a Docker container managed by the local init system.

Launch docker container II

Let's run the updated puppet manifests:

$ sudo puppet apply docker\_example.pp

Notice: Compiled catalog for puppet in environment production in 0.57 seconds

Notice: /Stage[main]/Main/Docker::Run[helloworld]/File[/etc/init.d/docker-helloworld]/ensure: created

Notice: /Stage[main]/Main/Docker::Run[helloworld]/Service[docker-helloworld]/ensure: ensure changed 'stopped' to 'running'

Notice: Finished catalog run in 2.47 seconds

We can check if it ran successfully:

$ sudo docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

f7d43960cb62 ubuntu "/bin/sh -c 'while t About a minute ago Up About a minute helloworld

If we want, we can attach it:

$ sudo docker attach f7d43960cb62

hello world

hello world

hello world