**Using Puppet to launch a Docker Swarm**

[Docker Swarm](https://docs.docker.com/swarm/) is part of the official Docker orchestration effort, and allows for managing containers across a fleet of hosts rather than just on a single host.

The [Puppet Docker module](https://forge.puppetlabs.com/garethr/docker) supports installing and managing Docker, and running individual docker containers. Given Swarm is packaged as containers, that means we can install a Swarm cluster using Puppet.

Swarm supports a number of [discovery backends](http://docs.docker.com/swarm/discovery/). For this example I'll be using [Consul](https://www.consul.io/), again all managed by Puppet.

**Usage**

vagrant up --provider virtualbox

This will launch 2 virtual machines, install Consul and register a cluster, install Docker and Swarm and then establish the swarm.

You can access the swwarm using a docker client, either from you local machine or from one of the virtual machines. For instance:

docker -H tcp://10.20.3.11:3000 info

If you don't have docker installed locally you can run the above command from one of the virtual machines using:

vagrant ssh swarm-1 -c "docker -H tcp://localhost:3000 info"

This should print something like:

Containers: 4

Nodes: 2

swarm-1: 10.20.3.11:2375

└ Containers: 3

└ Reserved CPUs: 0 / 1

└ Reserved Memory: 0 B / 490 MiB

swarm-2: 10.20.3.12:2375

└ Containers: 1

└ Reserved CPUs: 0 / 1

└ Reserved Memory: 0 B / 490 MiB

**Growing the cluster**

We can also automatically scale the cluster by launching additional virtual machines.

INSTANCES=4 vagrant up --provider virtualbox

This will give us a total of 4 virtual machines, 2 new ones and the 2 existing machines we already launched. Once the machines have launched you should be able to run the above commands again, this time you'll get something like:

Containers: 6

Nodes: 4

swarm-1: 10.20.3.11:2375

└ Containers: 3

└ Reserved CPUs: 0 / 1

└ Reserved Memory: 0 B / 490 MiB

swarm-2: 10.20.3.12:2375

└ Containers: 1

└ Reserved CPUs: 0 / 1

└ Reserved Memory: 0 B / 490 MiB

swarm-3: 10.20.3.13:2375

└ Containers: 1

└ Reserved CPUs: 0 / 1

└ Reserved Memory: 0 B / 490 MiB

swarm-4: 10.20.3.14:2375

└ Containers: 1

└ Reserved CPUs: 0 / 1

└ Reserved Memory: 0 B / 490 MiB

**Implementation details**

The example uses the Docker module to launch the swarm containers. First we run the main swarm container on all hosts.

::docker::run { 'swarm':

image => 'swarm',

command => "join --addr=${::ipaddress\_eth1}:2375 consul://${::ipaddress\_eth1}:8500/swarm\_nodes"

}

Then on one host we run the swarm manager:

::docker::run { 'swarm-manager':

image => 'swarm',

ports => '3000:2375',

command => "manage consul://${::ipaddress\_eth1}:8500/swarm\_nodes",

require => Docker::Run['swarm'],

}

Consul is managed by the excellent [Consul module](https://github.com/solarkennedy) from [Kyle Anderson](https://github.com/solarkennedy). Much of the Consul configuration is in the hiera data, for example:

consul::config\_hash:

data\_dir: '/opt/consul'

client\_addr: '0.0.0.0'

bind\_addr: "%{::ipaddress\_eth1}"