Cloud Provisioning and Configuration Management with Puppet

Introduction 3

Puppet Terminology 3

Installation 4

Debian & Ubuntu: 4

RedHat & Centos: 4

Configuration 4

Puppet Master: 4

Puppet Agent Node: 5

Generate Certificates 5

Resources 6

Resource Dependency 7

Provisioning DigitalOcean droplets with Puppet 8

# Introduction

Puppet is a configuration management tool. Configuration management tools are to keep the servers in a desired state. These tools are used to bring your infrastructure administration into a codebase, describing all processes necessary for deploying a server in a set of provisioning scripts that can be versioned and easily reused. These tools are used to check if the server is in a desired state or not, if not then configuration management tool brings that server in the desired state i.e. correct packages are installed, configuration files contain the expected values, files are having expected, right services are running, and so on.

Puppet is ruby based tool. Puppet is used to automate common infrastructure tasks and express solutions to common infrastructure problems. Along with the Puppet, there are other famous configuration management tools as well in this category Ansible, Chef, and Salt. Puppet is being used in many small and large organizations. Puppet can integrate with cloud-based platforms such as Digital Ocean, AWS to automatically provision and configure new machines.

# Puppet Terminology

Below are the main terms which are used when you are working with puppet.

**Puppet Master:** This is the master server which controls configuration on the nodes (agents)

**Puppet Agent Node:** A node controlled by a Puppet Master

**Manifest:** A file that contains a set of instructions to be executed

**Resource:** A portion of code that declares an element of the system and how its state should be changed. For instance, to install a package we need to define a package resource and ensure its state is set to "installed". Some of the resources are package, service, user, files

**Module:** a collection of manifests and other related files organized in a pre-defined way to facilitate sharing and reusing parts of a provisioning

**Class:** just like with regular programming languages, classes are used in Puppet to better organize the provisioning and make it easier to reuse portions of the code

**Facts:** global variables containing information about the system, like network interfaces and operating system

**Services:** used to trigger service status changes, like restarting or stopping a service

# Installation

You need to run below commands to install puppet master or agent on puppet server or node respectively.

## **Debian & Ubuntu:**

On debian and Ubuntu OS, puppet installation repos are already present and you will have to run below commands for installation

|  |
| --- |
| apt-get install puppet # On clients (nodes)  apt-get install puppetmaster # On server (master) |

## **RedHat & Centos:**

You will have to add rpm repo for as it not added be default. You can get the repo URL from <http://yum.puppetlabs.com/> . Go to this link and select the version you are using. For example: We need to run below command for CentOS6

|  |
| --- |
| #For centos 6  rpm -ivh <http://yum.puppetlabs.com/puppetlabs-release-el-6.noarch.rpm>  yum install puppet # On clients (nodes)  yum install puppet-server # On server (master) |

# Configuration

There is some configuration required before you start using puppet.

## **Puppet Master:**

You should also do below configuration on puppet master:

|  |
| --- |
| #get ip address of master and agent servers  ifconfig    #edit hosts  vi /etc/hosts    #Make entry like below  <ip\_address> puppet <some domain name i.e. puppet.myorg.co>    #Edit puppet conf file  vi /etc/puppet/puppet.conf    #edit in main section  dns\_alt\_names = puppet,puppet.myorg.co  certname=puppet    #Start puppet master  service puppetmaster start |

## **Puppet Agent Node:**

You should also do below configuration on puppet agent nodes:

|  |
| --- |
| #get ip address of master and agent servers  ifconfig    #edit hosts  vi /etc/hosts    #Make entry like below. Use puppet master ip and domain name used in puppet master config  <ip\_address> puppetagent  <puppet master ip\_address> puppet <some domain name i.e. puppet.myorg.co>    #Edit puppet conf file  vi /etc/puppet/puppet.conf    #edit in agent section  server = puppet.myorg.co  #Start puppet agent  service puppet start |

# Generate Certificates

You need to generate certificates to establish trusted connection between puppet master and puppet agent nodes. Please follow below steps:

1. Run below commands on puppet master

|  |
| --- |
| #stop puppet master  service puppetmaster stop    #generate master certificate  sudo -u puppet puppet master --no-daemonize --verbose    #press ctrl c once you see "Starting puppet master version x.y.z"    #Run puppet master  puppet resource service puppetmaster ensure=running |

1. Now, run below command on puppet agent node:

|  |
| --- |
| #stop puppet agent  service puppet stop    #generate certificate signing request for puppet master  puppet agent -t |

1. Go to puppet master and check if there is any cert signing request pending

|  |
| --- |
| #Check if any cert signing request is pending  puppet cert list    #sign cert  puppet cert sign <certname i.e. puppetagent>  #start puppet agent  puppet resource service puppet ensure=running |

# **Resources**

We will talk about resources here as puppet code uses resource block to make the changes in node state. Tasks or steps are defined by declaring resources. Resources can be packages, files, services, users, and commands. Resource might have a desired state declared. Puppet will bring the resource in that state on requested node. For example, desired state of a package resource is set to installed in your manifest. It will trigger a package installation on the system if the package is not installed.

This is what a package resource looks like:

|  |
| --- |
| package { 'nginx':  ensure => 'installed'  } |

Here, it will ensure that package nginx is installed on server.

You can execute any arbitrary command by declaring an exec resource, like the following:

|  |
| --- |
| exec { 'apt-get update':  command => '/usr/bin/apt-get update'  } |

The apt-get update portion in the first line is not the actual command. It is an identifier for this unique resource. Often we need to reference other resources from within a resource, and we use their identifier for that. In this case, the identifier is apt-get update, but it could be any other string.

# **Resource Dependency**

While writing the manifest file in puppet, you should remember that Puppet doesn't evaluate the resources in the same order they are defined. This is a common source of confusion for those who are getting started with Puppet. So, you should define the dependency if you want resources to be executed in specific order.

For example, you want execute a command, but you need to make sure a dependency is installed first:

|  |
| --- |
| package { 'python-software-properties':  ensure => 'installed'  }  exec { 'add-repository':  command => '/usr/bin/add-apt-repository ppa:ondrej/php5 -y'  require => Package['python-software-properties']  } |

The require option is used for that. Here, it is saying that resource 'python-software-properties' package should be evaluated first.

Now let's say you need to make sure a task is executed before another. For a case like this, we can use the before option instead:

|  |
| --- |
| package { 'curl':  ensure => 'installed'  before => Exec['install script']  }  exec { 'install script':  command => '/usr/bin/curl http://example.com/some-script.sh'  } |

# **Provisioning DigitalOcean droplets with Puppet**

You can provision digitalocean droplets using puppet. For this, you will need garethr-digitalocean module. This module can be found at below location

<https://forge.puppet.com/garethr/digitalocean>

To use this module, add this declaration to your Puppetfile:

|  |
| --- |
| mod 'garethr-digitalocean', '0.4.0' |

To manually install this module with puppet module tool:

|  |
| --- |
| puppet module install garethr-digitalocean --version 0.4.0 |

Here, we will see example that how we can launch 2 droplets droplet-dev-1 and deoplet-dev-2. Create a manifest file named launch-droplets.pp with below contents:

|  |
| --- |
| droplet { [' droplet-dev-1', droplet-dev-2']:  ensure => present,  region => 'lon1',  size => '512mb',  image => 14169855,  } |

You can run it with below command:

|  |
| --- |
| puppet apply --test launch-droplets.pp |

It will check if these two droplets exist in that region, and have that size. If they don’t exist then it will launch droplets using the specified image id. Hence, same command can be run multiple time. It droplets are already there then it will not do anything, if not then it will launch those droplets. Once it is launched then you can use droplet as puppet agent node and provision like other agent nodes.