Deploying a Website

Ice AS

PROJECT 1 REPORT
Alternative 3

Syeda Taiyeba Haroon
(s307040)

Infrastructure Services and Operations
INF4017NSA

Kyrre Begnum

Network and System Administration

Oslo and Akershus University College
of Applied Science

Oslo, Norway
8th October 2016
Abstract

A Report on website deployment issued by Ice AS as a documentation for future uses.
Contents

1 Introduction 1

2 Overview 2

3 Workflow Design 2

3.1 Current Workflow 2

3.2 Technologies Used 3

3.2.1 Project Management Tools 3

3.2.2 Virtualization Resources 7

4 Workflow Test 9

5 Technical Documentation and Discussion 9

5.1 Scripts 9

5.2 Foreman 10

5.2.1 Puppet Classes 10

5.3 BackUp 12

5.4 Overview of Problems faced 13

6 Experimental Results 15

6.1 Test Instructions 15

6.1.1 Project 1 - Junior’s Test Environment 15

7 Conclusion 20
9 Appendix A

9.1 Scripts Used

9.1.1 Deployment script

9.1.2 Puppet Certificate Deleting Script

9.1.3 MLN Script

9.1.4 Adding Hosts Script

9.2 Puppet Classes

9.2.1 Admins

9.2.2 Loadbalancer

9.2.3 Web Cache

9.2.4 Database Server Mysql

9.2.5 Webserver Apache2

9.2.6 Webserver Nodejs

9.3 Miscellaneous

9.3.1 Server Status and Deployments on Foreman, Openstack and Puppet

9.3.2 Test Outputs
1 Introduction

A development team is about to deploy their first version of web solution that they have been working on. They have requested our company to set up a number of servers to build their site.

The requirements are as follows:

- One Loadbalancing server: running Pound
- Two database servers: running mysql server (bin-log should be enabled.)
- Two webservers: running aapche2, php5, php-mysql libraries
- One webserver: running node.js
- One webcache: running memcached

Users: tom, brady, janet

The above users should be available on all servers. They should be part of a new group named webadmins. They need sudo-rights.

This operation does not require us to configure the services or databases. The users will configure it as required for their system.

The architecture will look as shown in the figure below. No arrows as been included as the users will further decide what they want to place where.

Figure 1: Host Groups for Website
2 Overview

The project had a few other requirements, that needed to be met for future deployments of servers and to maintain some documentation and professionalism.

- A kanban board should be used for the project.
- The ALTO Cloud should be used for virtualization.
- All deployed servers should follow a namespace.
- All deployed servers should be manageable by a central configuration management system (namespace is very important for this to work correctly).
- All deployed servers should have ssh-key based login from the master.
- The report should include a Workflow Design explaining how the kanban board is used.
- Technical documentation of the work so that a junior system admin can follow.
- There should be a Backup scheme for the project.
- A Workflow test should be done by simulating the plan three times.
- A discussion on obstacles faced and how they were solved.
- An Experimental Result from two different users who tried to deploy the project.

3 Workflow Design

This section will discuss the Requirement Workflow by the deployment project, our analysis on the task and the technologies used in this project.

3.1 Current Workflow

There current work flow for deploying a website looks a lot like the following:
1. The deployment script is run.
2. The script deploys all the servers
3. All the required softwares are installed by configuration management
4. Certificates for the servers are automatically accepted by the master
5. The servers are deployed and ready to be used.
6. The servers can be viewed and managed on Foreman

The development team has asked that this task be made easier and efficient as they don’t have anyone with system administration skills and will be recruiting a junior system admin for handling the website in future. Therefore it will be the Administration teams job to train the juniors.

The aim of our system administration team is to automate this task as much as possible. The idea is to create a script that will automate the who process of deploying servers, installing software on servers and be further managed by a configuration management.

3.2 Technologies Used

Here is an overview of the technologies that are considered used in the project.
A short insight on each of the tools used are given below.

3.2.1 Project Management Tools

Trello

Trello in simple words is a collaboration tool for users. It lets users see everything about there project in a single glance on a screen. It is actually a replacement of large boards with colored tags for tasks.

To get the best out of collaboration between the devops team a collaboration tool is very useful. The company has a board named "Ice AS" where they manage all their projects and activities. The Team put together a List named "Project 1 - Deploy Website" to plan and execute this project.
The board already is labelled with a number of List

- Inbox: New WIPs (our Todo list)
- To-do: WIPs that have been viewed
- In Progress: WIPs which are in progress
- QA: Quality checking of the WIP
- Done: Completed Tasks

They added two further lists and now it looks as follows:

- Inbox: New WIPs (our Todo list)
- Project 1-Deploy Website: All tasks reptalted to this project are listed here.
- To-do: WIPs that have been viewed
- In Progress: WIPs which are in progress
- QA: Quality checking of the WIP
- Backup and Documentation: Checking whether documentation has been written for that task
- Done: Completed Tasks

Other Color codes were used on the board to label tasks as follows:

- project1: cyan
- assigned tasks: purple
- needs attention: red
- testing: orange
- reading: yellow
- ready to deploy: green
Namespace

Namespaces are used in projects to create discipline and order. There are quite a few types of namespaces in use;

- Formulatic
- Thematic
- Functional
- Descriptive
- No Particular method
- Combo of Functional and Descriptive

The namespace adopted for this project is combination of functional and descriptive namespace.

The master is named prime which describes the master machine as being the primary instance of this project.

Figure 2: Master as seen on Openstack

The other machines are named as follows;

< machinedesignation > – < machineusage >

Webservers:

- webservers-apache2-w1
- webservers-apache2-w2
- webservers-nodejs-w1
Database Servers:

- db-server1
- db-server2

Web Cache:

- web-cache1

Loadbalancer:

- lb-server1

The host groups as seen on Foreman:

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>db-server</td>
</tr>
<tr>
<td>loadbalancer</td>
</tr>
<tr>
<td>web-cache</td>
</tr>
<tr>
<td>webserver-apache2</td>
</tr>
<tr>
<td>webserver-nodejs</td>
</tr>
</tbody>
</table>

Figure 3: Host Groups for Website
### 3.2.2 Virtualization Resources

**MLN**

MLN as the name suggests helps "Manage Large Networks". It is tool written in perl and supports various virtualization technologies such as Openstack, AmazonEC3, Xen etc.

In this project MLN is used to deploy the webservers on the ALTO Cloud as required.

The MLN tool maybe found at this link [http://mln.sourceforge.net/](http://mln.sourceforge.net/)

**Puppet**

Puppet is the best tool for automation and reliability in large infrastructures. Puppet is the most re-known configuration management tool in market now. Puppet can be used to ease every step of the software delivery process starting from virtual machines to packages, services, cron jobs, manage users, storage and network infrastructures. Puppet provides quite a few integrations and in this project puppet is integrated with [Openstack](http://openstack.org/).

When using puppet, very system is equipped with a puppet agent which in turn is controlled by a puppet master(central). Puppet also has a tool named *facter* which collects information about the system in a way usable by puppet.

Below are a few important terms which are important to understand puppet.

**Resources**

Puppet comes with a number of pre-defined resources which are the foundation of any infrastructure. The most commonly used resources are users, packages, files, services and cron.

**Manifests**

These resources as described above has to be in a special policy syntax to live. This is called the Manifests. manifests can be stand alone resource declarations or be defined in a group which can be applied by using class. These manifests files end with extension .pp Example of a resource manifest:

```
user { "susanne":
    ensure => present,
} 
```
**Modules**

Puppet Modules aka class are puppet codes that are pre-written to manage a particular task. These modules are a collection of files and directories that can contain manifests, as well as other objects such as files and templates, all packaged and organized in a way that Puppet can understand and use.

To use any module to manage a particular node first is to classify each node by assigning to it a class within the module.

![Diagram of Puppet modules]

Figure 4: Puppet pieced together

**Foreman**

Foreman is needed when repetitive tasks need to be automated. It is a complete lifecycle management tool for physical and virtual servers. If configured wisely, Foreman can quickly deploy applications, and proactively manage servers, locally or virtually in the cloud. In this project Foreman is a functional replacement of Puppet Master.

**GitLab**

GitLab is an application to code, test, and deploy code together. It provides Git repository management with fine-grained access controls, code reviews, issue tracking, activity feeds, wikis, and continuous integration.

In this project Github and/or Gitlab has been used to keep update of the Project as it progressed.
4 Workflow Test

We have designed the workflow to look like as follows:

- Deploy Servers
- Wait for the Servers to show up in Foreman
- Check that all the requirements are fulfilled
- Check if the Servers correct statuses are showing

5 Technical Documentation and Discussion

5.1 Scripts

This project is created with the use of 4 scripts.

1. `website.sh`
2. `del_cert.sh`
3. `deploywebsite.sh`
4. `add_hosts.sh`

`website.sh`

This is the script that the system admin has to run to deploy the website. It is where all the other scripts are called to take place in the correct order.

`del_cert.sh`

This script looks for certificates with the same name as the machines about to be deployed and deletes the old certificates.

`deploywebsite.sh`

This script contains the mln script for deploying the servers. It deploys the 7 servers as the requirement; one loadbalancer, 2 apache2 webservers, one nodejs webserver, one memcached server and 2 database servers.
Along with deploying the website the script also installs puppet and git on all the webservers and configures puppet agent on each of them.

`add_hosts.sh`

This script adds the new servers ip address to the `/etc/hosts` file of the master so that foreman can fetch the ip addresses.

### 5.2 Foreman

The job of foreman in this project is to act as the puppet master and apply all the manifests to the correct host servers and depending on the manifest it reports whether the manifest has been applied properly.

Foreman Reports page shows reports for each host server and what manifest has been applied on it.

#### 5.2.1 Puppet Classes

All puppet classes are stored in:

```
/etc/puppet/environments/common/
```

When puppet classes and environments are fetched in foreman, they are each fetched based on the host groups created on foreman. These host group names are the same as the puppet class names as shown below.

Each of these classes contain one `init.pp` file which contains the manifest to be applied.

There is only one manifest file because the requirements for the project is quite simple.

The `init.pp` file contains reference to the package to be installed and ensures that it is installed.

Some manifests also include writing some configurations to files and some also ensures that the package is running.
Webserver apache2 Class

/etc/puppet/environments/production/modules/webserver–apache2/manifests/

Webserver nodejs Class

/etc/puppet/environments/production/modules/webserver–nodejs/manifests/

Loadbalancer Class

/etc/puppet/environments/production/modules/loadbalancer/manifests/

Webcache Class

/etc/puppet/environments/production/modules/web–cache/manifests/

Database Class

/etc/puppet/environments/production/modules/db–server/manifests/

Enable Diffs

To see configuration file changes from within Foreman’s Reports view, edit puppet.conf:

    sudo nano /etc/puppet/puppet.conf

    show_diff = true
5.3 BackUp

Last a little discussion on back up is very important as, there can be various circumstances that can lead to lose of work.

Initially backup of the system was only kept on Gitlab, but as it seems there was a major Server breakdown on the 2nd of October and the server was only back up the next day. During that time deployment wasn’t possible because we had kept the puppet_agent config file on gitlab. Therefore a backup of this mechanism would not always be reliable.

It was decided that if there was enough time before the completion of this project a Back Up script would be set up as follows.

A decision was made to use the script written by Kyrre Begnum, found in the link here.

It can be downloaded on the server with wget and extracted for use.

```
#To download script
wget http://www.cs.hioa.no/~kyrre/pull_backup.tgz

#To extract Script
tar xzf pull_backup.tgz
```

This backup script uses the system file `/etc/backup_plan.conf` to find out what IP addresses to SSH into and take backup from.

Therefore it is required that a script be written that finds all the servers running on nova client and fetch them for the script to be able to take backup. At the end of the project, due to a lot of problems with configuring the master a public project was created on github so that no ssh keys wold need to be configured each time.

Another solution for the puppet_agent file would be to have a webserver running just for these files, but this was not voted for as it was the only file that needed to be copied and it would be a waste of resource to create e webserver just for that.
5.4 Overview of Problems faced

Over all writing the scripts and manifests were the easy tasks.

Setting up the master environment is where I faced obstacles, particularly with Foreman.

The first setup of master had everything working smoothly. Foreman was also installed correctly but issued an error when trying to import puppet classes. Below is two different kinds of errors that I got.

![Error type 1: Related to Proxy](image1.png)

**Figure 5: Error type 1: Related to Proxy**

![Error type 2: Related to environment](image2.png)

**Figure 6: Error type 2: Related to environment**

As tried, Foreman is quite difficult to remove and clean from a system therefore a new master deployment was the best suggestion.

However with identical machine and puppet configuration, foreman kept throwing errors during installation. After about six master server configurations (foreman was not installing without errors), the seventh version of the master VM named prime had a success with foreman installation.

After that there was no more errors with confuring the master.

Below is an example of the error during installation:

Further down the road I had some issues with puppet classes where I had
Figure 7: Error during Installation

asked the manifest to ensure that the service was running even though I had not configured it because the project requirement was to just install the services.

So these minor errors were fixed after the test run and there was no more errors reported at the time of writing this report.
6 Experimental Results

6.1 Test Instructions

A Trello List was created to list all the tasks assigned to the Juniors. Cards assigned to juniors were marked with **black color**.

Two juniors were picked, one of them less experienced in the field and the other with more experience. Labelled as A and B.

6.1.1 Project 1 - Junior’s Test Environment

**Junior is given an Introduction to the task**

1. Explain the project: The development project is about to deploy their first version of a web solution they have been working on. They now request a set of servers to build the site on:

   - One server who is to be the loadbalancer. This one should run pound.
   - Two database servers running the mysql server
   - Two webservers, with apache2, php5 and php-mysql libraries
   - One webserver with node.js installed
   - One machine running the memcached service
   - They can configure their database themselves, but require you to enable bin-log on both.
   - The users tom, brady and janet should be on all servers.
   - They need sudo rights and to be member of the new group webadmins.

2. Explain the steps involved

   - ssh into machine
   - cd into project1
   - run bash website.sh
   - follow the instructions
Good luck!

**Open project1**

- If you have the configuration needed to run this project:
  
  `git clone https://www.github.com/graphiteflake/project1.git`

  OR

  As we assume in this case that you will be provided access to the machine that has the environment setup and the project downloaded.

- Next, inspect the files in project 1 directory

- Use the trello board as you progress.

**Run website.sh**

$ bash *website.sh*

Note the time you start running the script.

**Pre-deploy Questions?**

If you have any questions, add them here. Mark with Red label

**Junior A:**

Personally assisted Junior A during the deployment. She has questions related to how many servers will be deployed and how she should check if the software’s and users were correctly installed.

It seemed the description of the project was not included in the trello card therefore it was presented to her during the deployment.

**Junior B:**

She was given ssh access to the master machine and she didn’t have any questions as the description was provided to her beforehand.
Record Time of Deployment

The script will tell you when the deployment starts. Record that in comments here.

Junior A:
Deployment Started @ 11:38

Junior B:
Deployment Started: 3:45

These times were recorded when the script said "Deployment Started"

Record Deployment End Time

When the deployment is complete, The nova list will be displayed. Note the time down in comments calculate and write down the time taken for the deployment.

Junior A:
Deployment Ended @ 11:40

Junior B:
Deployment Finished: 3:46

So as recorded the deployment took approximately 2 minutes including the pauses that were included in the script to avoid any errors during deployment.

Recheck status of Servers

You can check that with :

$ mln status -p deploywebsite

Or check on openstack (if you have permission)

Part of the script they ran showed them the output of

$ mln status -p deploywebsite

Therefore they did not have to run this command or visit openstack to check.
Check Foreman

- Check foreman to see All hosts.
- Check Puppet Classes
- Check Puppet Environments

ps: all servers are deployed in production mode :) 

The servers took some time to show up in foreman. None of the testers recorded the time as they focused on watching which machines deployed faster and which ones reported errors midway.

Junior A, ssh’d into each of the deployed servers and ensured that the users were created and that the services were installed.

Junior B, checked the reports on foreman to ensure user creation and service installations.

After the tests we checked for the errors that occurred and it was because of the errors that were fixed in the puppet classes earlier but were not imported before deployment.

As soon as the puppet classes were imported the server reporting error became active.

One last deployment test was done by me to ensure that every error was taken care off before completing the report.
Add a Short comment on the overview of the project and report

![Comment from one of the Junior A on Trello](image)

**Figure 8**: Comment from one of the Junior A on Trello

![Comment from one of the Junior B on Skype](image)

**Figure 9**: Comment from one of the Junior B on Skype

From the comments seen above, it seemed the project was a success from a user perspective.
7 Conclusion

After having spend some time working with mln the task seemed quite easy. Puppet configuration was also quite easier then I anticipated. However there was quite a hassle with foreman but in the end all the project requirements were fulfilled. At the end of the report the appendix contains infographs of different test outcomes and scripts used in the project.

Overall I can see a few other things that could have been automated too and some more features could be added. However the time spend on re-installing the master so many times, I feel satisfied with the project.

Finally, having kept documentation of everything from the beginning really helped reconfigure and fix things faster when obstacles were faced.

8 References

Website Construction Image
About Gitlab
The Project scripts
The Project Puppet Classes
Forked and Re-based Nova Script
9 Appendix A

9.1 Scripts Used

9.1.1 Deployment script

name: website.sh

#!/bin/bash
echo "*****DEPLOYING WEBSITE*****"
echo "*****Checking for irrelevant certificates*****"
sudo bash cert_del.sh
echo ".....old certificates taken care off....."
sleep 5
echo "*****Constructing Website*****"
sudo mln build -f deploywebsite.mln
echo "***Construction Complete!***"
echo "Please Wait as we Deploy..."
sleep 20
sudo mln start -p deploywebsite
echo "wait ... check openstack for visual..."
sleep 30
echo "*****Website Deployed*****"
echo "Auto Signing agent certificates"
sleep 3
echo "................."
sleep 10
echo "*****Showing you list of servers:*****"
sudo mln status -p deploywebsite
sudo bash addhosts.sh
echo "*****You may now check on Foreman if the servers are up*****"
echo "This may take up to a minute or two..."

echo "Deployment Complete!!"

9.1.2 Puppet Certificate Deleting Script

name: cert_del.sh

#!/bin/bash

sudo puppet cert clean webserver-apache2-w1.openstacklocal
sudo puppet cert clean webserver-apache2-w2.openstacklocal
sudo puppet cert clean web-cache-1.openstacklocal
sudo puppet cert clean db-server-1.openstacklocal
sudo puppet cert clean db-server-2.openstacklocal
sudo puppet cert clean loadbalancer-1.openstacklocal
sudo puppet cert clean webserver-nodejs-w1.openstacklocal

9.1.3 MLN Script

name: deploywebsite.mln

global {
    project deploywebsite
}

superclass common {
    openstack {
        image Ubuntu14.04
        flavor m1.medium
        keypair prime-key
        user_data {
            echo "127.0.1.1 $(hostname).openstacklocal $(hostname)"
            \ >> /etc/hosts
        }
    }
}
sudo apt-get update
sudo apt-get -y install puppet git
git clone https://github.com/graphiteflake/project1.git
cd project1
sudo puppet apply puppet_agent.pp
sudo service puppet restart
sudo puppet agent --enable

network eth0 {
    net nsa_master_net
    address dhcp
}

superclass loadbalancer {
    superclass common
    openstack {
        user_data {
            sudo apt-get update
        }
    }
}

superclass db-server {
    superclass common
    openstack {
        user_data {
            sudo apt-get update
        }
    }
}
superclass web-cache {
    superclass common
    openstack {
        user_data {
            sudo apt-get update
        }
    }
}

superclass webserver-apache2 {
    superclass common
    openstack {
        user_data {
            sudo apt-get update
        }
    }
}

superclass webserver-nodejs {
    superclass common
    openstack {
        user_data {
            sudo apt-get update
        }
    }
}

host loadbalancer-1 {
superclass loadbalancer
}

host db-server-1 {
  superclass db-server
}

host db-server-2 {
  superclass db-server
}

host webserver-nodejs-w1 {
  superclass webserver-nodejs
}

host webserver-apache2-w1 {
  superclass webserver-apache2
}

host webserver-apache2-w2 {
  superclass webserver-apache2
}

host web-cache-1 {
  superclass web-cache
}
9.1.4 Adding Hosts Script

name: addhosts.sh

#!/bin/bash

echo "Adding agents to host list"
sudo chmod 666 /etc/hosts
IFS="
"

for i in 'nova list --all-tenants | grep -v '^-' | grep -v '^| ID' | \
cut -d "|" -f 2,3,5 | sed -e "s/ */ */g" -e "s/* */g" ; do
ID='echo $i | cut -d, -f 1'
NAME='echo $i | cut -d, -f 2'
STATUS='echo $i | cut -d, -f 3'
SHOW='nova show ${ID}''
HV='echo "$SHOW" | grep OS-EXT-SRV-ATTR:host
\ | awk '{print $4;}''
IP='echo "$SHOW" | grep " network"
\ sed -e "s/.*network */ */ | awk '{print $4}''
if [ "$NAME" != "ice-core" ]; then
    echo -e "$IP\t$NAME" >> /etc/hosts
fi
done
9.2 Puppet Classes

9.2.1 Admins

class: admins

class admins {
    group { 'webadmins':
        ensure => present,
        gid => $uid,
    }

    group { 'sudo':
        ensure => present,
        gid => '27',
    }

    user { 'tom' :
        ensure => present,
        uid => $uid,
        gid => '27',
        groups => [ 'webadmins' ],
        password => 'tommy',
    }

    user { 'brady' :
        ensure => present,
        uid => $uid,
        gid => '27',
        groups => [ 'webadmins' ],
        password => 'brandy',
    }
}
user { 'janet' :
    ensure => present,
    uid => $uid,
    gid => '27',
    groups => [ 'webadmins' ],
    password => 'janet',
}

file {'/etc/motd':
    ensure => file,
    mode => '0644',
    content => "This VM is set up and managed by Ice AS.
    This VMs IP address is ${ipaddress}, Hostname ${fqdn}.
    Running ${operatingsystem} ${operatingsystemrelease} and Puppet ${puppetversion}.",
}

9.2.2 Loadbalancer

class: loadbalancer

class loadbalancer {
    exec { 'apt-update':
        command => '/usr/bin/apt-get update'
    }
    package { 'pound':

require => Exec['apt-update'],
ensure => installed,
}

service { 'pound':
    ensure => running,
}

file { '/etc/default/pound':
    ensure => file,
    content => 'startup=1',
    require => Package['pound'],
}

9.2.3 Web Cache

class web-cache

class web-cache {
    exec { 'apt-update':
        command => '/usr/bin/apt-get update'
    }
    package { 'memcached':
        require => Exec['apt-update'],
        ensure => installed,
    }
    service { 'memcached':
        ensure => running,
    }
}
9.2.4 Database Server Mysql

class: db-server

class db-server {

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

    package { 'mysql-server':
        require => Exec['apt-update'],
        ensure => installed,
    }

    file { '/etc/my.cnf':
        ensure => file,
        mode => '0644',
        content => '
            [mysqld]
            log-bin=mysql-bin
            expire-log-days=14
            max-binlog-size=500M',
    }
}

9.2.5 Webserver Apache2

class: webserver-apache2

class webserver-apache2 {

    exec { 'apt-update':
        command => '/usr/bin/apt-get update'
    }
}
package { 'apache2':
    require => Exec['apt-update'],
    ensure => installed,
}

service { 'apache2':
    ensure => running,
}

package { 'php5':
    require => Exec['apt-update'],
    ensure => installed,
}

file { '/var/www/html/info.php':
    ensure => file,
    content => '<?php phpinfo(); ?>',
    require => Package['apache2'],
}

package { 'php5-mysql':
    ensure => installed,
}

}
9.2.6  Webserver Nodejs

class: webserver-nodejs

class webserver-nodejs {
    exec { 'apt-update':
        command => '/usr/bin/apt-get update'
    }

    package { 'nodejs': require => Exec['apt-update'],
        ensure => installed,
    }
}

9.3 Misellaneous

This section contains screenshots of tests done through out the project compilation.

9.3.1 Server Status and Deployments on Foreman, Openstack and Puppet

Figure 10: Hostgroups on Master
Figure 11: Hosts and Hostgroups configured

Figure 12: Deployed servers on Openstack
### Figure 13: Deployed servers on Foreman

<table>
<thead>
<tr>
<th>Name</th>
<th>Operating system</th>
<th>Environment</th>
<th>Model</th>
<th>Host group</th>
<th>Last report</th>
</tr>
</thead>
<tbody>
<tr>
<td>db-server-1.operatlocal</td>
<td>Ubuntu 14.x</td>
<td>production</td>
<td>OpenStack Nova</td>
<td>db-server</td>
<td>10 minutes ago</td>
</tr>
<tr>
<td>db-server-2.operatlocal</td>
<td>Ubuntu 14.x</td>
<td>production</td>
<td>OpenStack Nova</td>
<td>db-server</td>
<td>10 minutes ago</td>
</tr>
<tr>
<td>loadbalancer-1.operatlocal</td>
<td>Ubuntu 14.x</td>
<td>production</td>
<td>OpenStack Nova</td>
<td>loadbalancer</td>
<td>10 minutes ago</td>
</tr>
<tr>
<td>prime-operatlocal</td>
<td>Ubuntu 14.x</td>
<td>production</td>
<td>OpenStack Nova</td>
<td></td>
<td>20 minutes ago</td>
</tr>
<tr>
<td>web-cache-1.operatlocal</td>
<td>Ubuntu 14.x</td>
<td>production</td>
<td>OpenStack Nova</td>
<td>web-cache</td>
<td>10 minutes ago</td>
</tr>
<tr>
<td>webserver-apache2-operatlocal</td>
<td>Ubuntu 14.x</td>
<td>production</td>
<td>OpenStack Nova</td>
<td>webserver-apache2</td>
<td>10 minutes ago</td>
</tr>
<tr>
<td>webserver-apache2-operatlocal</td>
<td>Ubuntu 14.x</td>
<td>production</td>
<td>OpenStack Nova</td>
<td>webserver-apache2</td>
<td>10 minutes ago</td>
</tr>
<tr>
<td>webserver-nodes-1.operatlocal</td>
<td>Ubuntu 14.x</td>
<td>production</td>
<td>OpenStack Nova</td>
<td>webserver-nodes</td>
<td>10 minutes ago</td>
</tr>
</tbody>
</table>

### Figure 14: Changed Environments

#### Changed environments

Select the changes you want to realize in Foreman

<table>
<thead>
<tr>
<th>Environment</th>
<th>Operation</th>
<th>Puppet Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>common</td>
<td>Add</td>
<td>admin, db-server, loadbalancer, web-cache, webserver-apache2, and webserver-nodes</td>
</tr>
<tr>
<td>development</td>
<td>Add</td>
<td>admin, db-server, loadbalancer, web-cache, webserver-apache2, and webserver-nodes</td>
</tr>
<tr>
<td>example_dev</td>
<td>Add</td>
<td>admin, db-server, loadbalancer, web-cache, webserver-apache2, and webserver-nodes</td>
</tr>
<tr>
<td>production</td>
<td>Add</td>
<td>admin, db-server, loadbalancer, web-cache, webserver-apache2, and webserver-nodes</td>
</tr>
</tbody>
</table>
Figure 15: Including classes for Hostgroups

Figure 16: The first master on Openstack
9.3.2 Test Outputs

This section has some screenshots from when tests were run.

Figure 17: Users tested for sudo permission

Figure 18: Users present in /etc/passwd

Figure 19: The Opening message on all Servers

Figure 20: Loadbalancer

Figure 21: Web-cache Server
**Figure 22: Webserver-Node.js**

```
ubuntu@webserver-nodejs-w1:~$ nodejs --version
v0.10.25
```

**Figure 23: Webserver Apache2**

```
ubuntu@webserver-apache2-w1:~$ php -m | grep mysql
mysql
pdo_mysql
ubuntu@webserver-apache2-w1:~$ php -v
PHP 5.5.9-ubuntu4.20 (cli) (built: Oct 3 2016 13:00:37)
Copyright (c) 1997-2014 The PHP Group
Zend Engine v2.5.0, Copyright (c) 1998-2014 Zend Technologies
with Zend OPcache v7.0.3, Copyright (c) 1999-2014, by Zend Technologies
ubuntu@webserver-apache2-w1:~$ sudo service apache2 status
* apache2 is running
ubuntu@webserver-apache2-w1:~$
```

**Figure 24: Database Server**

```
ubuntu@db-server-2:~$ sudo service mysql status
mysql start/running, process 4990
ubuntu@db-server-2:~$
```

**Figure 25: Database Server**

```
ubuntu@db-server-2:~$ sudo mysql
Welcome to the MySQL monitor. Commands end with ; or \
Your MySQL connection id is 43
Server version: 5.5.52-0ubuntu5.14.04.1-log (ubuntu)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> SHOW VARIABLES LIKE 'log_bin';
<table>
<thead>
<tr>
<th>Variable_name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>log_bin</td>
<td>ON</td>
</tr>
</tbody>
</table>
1 row in set (0.00 sec)
```

38
Figure 26: Sql configuration Error reported

Figure 27: Database Server Healthy