



UiO : Universitetet i Oslo



Infrastructure Services and Operations (INF4017NSA) Project 1 Report

Written and compiled by: Getinet Ayele Eshete(s326610)
Martin Veshovda Løland(s236323)

October 10, 2017

Table of Contents

1	Introduction.....	3
2	Problem Statement	3
3	Overview of technologies used	3
4	Workflow design and Implementation	4
4.1	The general thought.....	4
4.2	Configuring puppetmaster	4
4.3	Foreman and puppet classes	6
5	Workflow test.....	10
5.1	Self-deployment (x3)	10
5.2	Junior's test	11
6	Conclusion	11
7	Appendix	12

1 Introduction

Cloud environment and Virtual machines are becoming the trend technology around the world. HiOA provides us an OpenStack cloud environment called ALTO. We used this cloud environment to create and deploy policy on virtual machines using MLN, Puppet and Foreman tools.

In this assignment, we used central configuration management system to automatically deploy a web solutions which consists of seven virtual machines, four users with sudo permission and some specific website development environments by running a single command. Our program also give options to group hosts based on the web development environment and frameworks they are running.

The test results by group members (self-test) shows that the time required for deploying our infrastructure has been improved about 50% from test 1 to test 3. The deployment of our infrastructure is also done by junior system administrators with the assist of us and the guideline prepared.

2 Problem Statement

In this assignment, four problems were given. We selected the third problem which is about deploying web solution. The problem statement was given as follows:

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 binlogon 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»

3 Overview of technologies used

3.1.1 MLN

MLN is the basis of the project. MLN gives you the opportunity to create multiple hosts with different (or alike) basic settings (OS, hard drive size, network, and so on).

3.1.2 Puppet

Puppet is the means of configuration management. Puppet will apply all extra packages and user settings for all the different nodes with the use of puppet classes.

3.1.3 Foreman

Foreman is a web based GUI for puppet and can be used to automatically apply some puppet policies.

3.1.4 Additional

We used scripts to group up the use of different commands, you'll see this in later examples.

4 Workflow design and Implementation

At the beginning of this project, we listed out in Trello the main activities to be done in this project as follows:

1. We need to choose which project we want to complete.
2. Setting up a deployment environment
3. DOCUMENTING
4. Deploy our environment three times
5. Find someone to test our deployment based on our documentation
6. Write the report

26 Sep at 09:38 - [Reply](#) - [Delete](#)

But, during the whole project the members have had daily communication through Facebook about detail activities.

4.1 The general thought

The workflow design can be seen generally as:

- MLN will set up the machines
- The puppet master will accept all puppets and be a central command point
- Foreman/Puppetmaster will apply all node specific settings

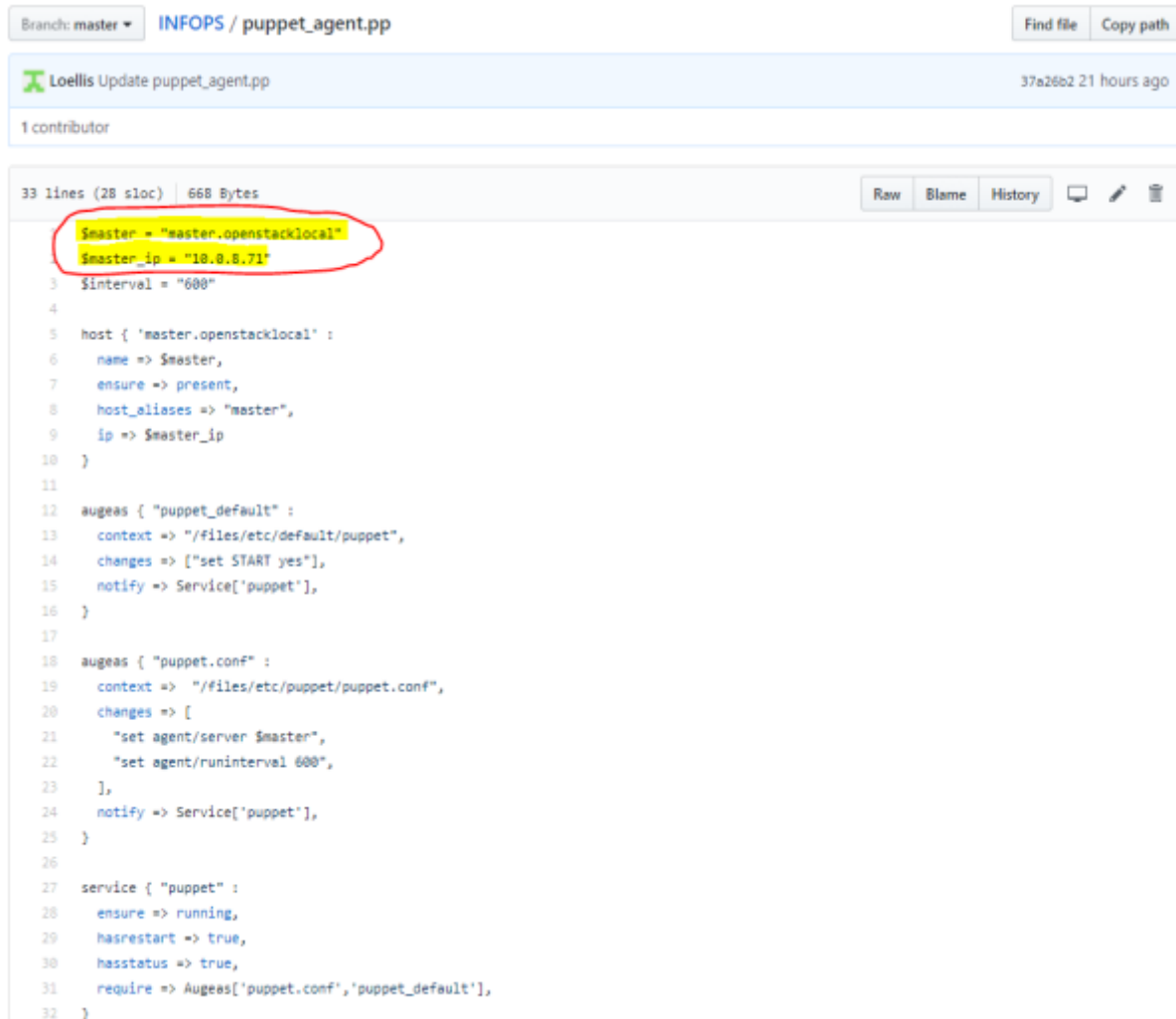
4.2 Configuring puppetmaster

The whole project started by setting up the puppetmaster. We'll assume the knowledge of how to install both MLN and Puppet is present and will therefore not be discussed.

After installing MLN and puppet we began installing Foreman. This was done by following the steps seen in Trello.

After installing MLN, Puppet, and Foreman we did a lot of testing with different uses of the MLN user_block. We lacked the knowledge of how the keypair keyword in mln worked. This caused one of us to not be able to reach the newly created nodes. After some help from Kyrre, we understood that the user who's credentials are stored in the .openstack file will be the only one with an automatic ssh-key to the newly created hosts. This problem was solved by creating a git-repository (<https://github.com/Loellis/INFOPS>) and storing a puppet manifest that applies the other user's ssh-key to the new host.

After successfully creating multiple hosts that both could ssh to, we had to make the hosts show up in the Foreman web GUI. We had some problems with this to begin with, but we realized we could add a puppet agent manifest in our git repository and fill in our puppet master's credentials:



```
Branch: master ▾ INFOPS / puppet_agent.pp Find file Copy path
Loellis Update puppet_agent.pp 37a26b2 21 hours ago
1 contributor

33 lines (28 sloc) | 668 Bytes Raw Blame History
1 $master = "master.openstacklocal"
2 $master_ip = "10.0.8.71"
3 $interval = "600"
4
5 host { 'master.openstacklocal' :
6   name => $master,
7   ensure => present,
8   host_aliases => "master",
9   ip => $master_ip
10 }
11
12 Augeas { "puppet_default" :
13   context => "/files/etc/default/puppet",
14   changes => ["set START yes"],
15   notify => Service['puppet'],
16 }
17
18 Augeas { "puppet.conf" :
19   context => "/files/etc/puppet/puppet.conf",
20   changes => [
21     "set agent/server $master",
22     "set agent/runinterval 600",
23   ],
24   notify => Service['puppet'],
25 }
26
27 service { "puppet" :
28   ensure => running,
29   hasrestart => true,
30   hasstatus => true,
31   require => Augeas['puppet.conf','puppet_default'],
32 }
```

When this was completed we created a script to run our MLN build and project (<https://github.com/Loellis/INFOPS/blob/master/Deploy.sh>). This was to skip typing in every command every time you wanted to test a new build. The script consists of the following steps:

- Stop all currently running nodes
- Remove all current nodes and their certificates
- Build MLN file
- Start MLN project
- Listing and signings certificates of all nodes

When running the script the second time we met an unexpected error: There were no certificates to sign on the master server, and when SSH-ed into a sub-node they couldn't create a new certificate request. This stopped all our new nodes from showing up in

foreman. We suddenly realized that the nodes earlier created still had their certificates signed, but did not match the newly created nodes. This caused us to add another step in our deployment script: removing all old certificates. Now when we ran the script all we had to do, was to wait until all the certificate requests popped up when «puppet cert list» was typed or they all showed up in the directory «/var/lib/puppet/ssl/ca/requests/». We tried at first to add the command «puppet cert sign –all», but understood quickly the command was ran before the newly created nodes ever had a chance to send their requests.

After being juniors for Andrey and Emilien, they told us they had made their certificate signing process automated by doing a while loop until all certificates were found. We added this to our script (all credits to Emilien and Andrey for this automation) and now our deployment was completely automated to the point that all our nodes showed up in foreman.

4.3 Foreman and puppet classes

After seeing all our nodes in foreman, we wanted to automate all node specific actions by creating puppet classes, connect a puppet class to a given Hostgroup in foreman, and then have foreman apply the policy in the puppet class to all hosts in the Hostgroup.

We wanted 6 different puppet modules:

- **common:** To apply changes to all nodes (e.g. adding users)
- **database:** To make sure our database hosts run mysql-server
- **loadbalancer:** To make sure pound is installed to our loadbalancer host
- **memcache:** To make sure memcache is running on our memcache host
- **webnode:** To make sure node.js is installed on our webserver with node host
- **webserver:** To make sure our webserver both have Apache2, php-libraries, and php7.0*

*Since we used Ubuntu16.04 php5.0 is not compatible, and therefore used php7.0

To make sure all our puppet classes showed up in foreman we had to make all our class directories without capital letter:

```
root@master:/etc/puppet/environments/common# ls
common database loadbalancer memcache webnode webserver
```

Within each of the classes there were 3 subdirectories: manifests, files, and templates. In our case, we only needed to use the manifests directory. In the manifest directory we had to create a init.pp file. This is the file that foreman will apply. In our common class, we created subclasses of users and motd:

```
class common {  
    include common::users  
    include common::motd  
}
```

```
class common::users {  
    group { 'webadmins' :  
        ensure => present,  
    }  
  
    user { 'tom' :  
        ensure => 'present',  
        groups => ['sudo', 'webadmins'],  
        password => 'salSp1wOPp6fk',  
        require => Group['webadmins'],  
    }  
  
    user { 'brady' :  
        ensure => 'present',  
        groups => ['sudo', 'webadmins'],  
        password => 'salSp1wOPp6fk',  
        require => Group['webadmins'],  
    }  
  
    user { 'janet' :  
        ensure => 'present',  
        groups => ['sudo', 'webadmins'],  
        password => 'salSp1wOPp6fk',  
        require => Group['webadmins'],  
    }  
}
```

```
class common::motd {  
    file { 'motd' :  
        ensure => file,  
        path => '/etc/motd',  
        mode => 0644,  
        content => "ThinkQuick Welcomes you to the server: ${fqdn}."  
    }  
}
```

Since both subclasses were quite small, this wasn't really necessary, but it felt as good practice and good procedure. All other init.pp manifests were only one class.

After creating the puppet classes, we created all our host groups in foreman:

Host Groups

Filter ... × Q Search ▼			
Create Host Group Help			
Name	Hosts	Hosts including sub-groups	Actions
Common	0	0	Nest ▼
Common/Database	0	0	Nest ▼
Common/Loadbalancer	0	0	Nest ▼
Common/Memcache	0	0	Nest ▼
Common/Webserver	0	0	Nest ▼
Common/WebserverNode	0	0	Nest ▼

Displaying all 6 entries

From the picture above we can see that Common is a parent host group, while all the other host groups are children of this hostgroup. This is to make sure that the policies stored in Common will be applied to all the child host groups.

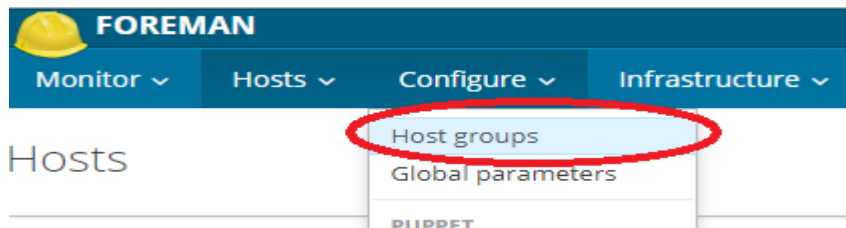
We then imported all puppet classes from our puppet master under Configure --> Classes and then press the Import environments button. This gave the following classes:

Puppet classes

Filter ... × Q Search ▼		Import environments from master.openstacklocal Documentation				
Class name	Environments	Host groups	Hosts	Parameters	Variables	Actions
common	common, development, and production	Common	0	0	0	Delete
common:motd	common, development, and production	Common	0	0	0	Delete
common:users	common, development, and production	Common	0	0	0	Delete
database	common, development, and production	Common/Database	0	0	0	Delete
loadbalancer	common, development, and production	Common/Loadbalancer	0	0	0	Delete
memcache	common, development, and production	Common/Memcache	0	0	0	Delete
webnode	common, development, and production	Common/WebserverNode	0	0	0	Delete
webserver	common, development, and production	Common/Webserver	0	0	0	Delete

Displaying all 8 entries

Since this picture is taken after the project was finished you can see that the classes have already been assigned to the different host groups. This was done by going to Configure --> Host groups, clicking on a specific Host group, choose the Puppet Classes tab, and from there add the corresponding puppet classes. Below you can see an example of every step for the puppet class common:



Host Groups

Filter ...

Name
Common
Common/Database
Common/Loadbalancer
Common/Memcache

Edit Common

Host Group Puppet Classes Network

Parent

Name *

Common

Environment

Included Classes

Available Classes

Filter classes

common

- common
- common::motd
- common::users
- + database
- + loadbalancer

Click to add common

- + memcache
- + webnode
- + webserver

Cancel Submit

After creating the host groups and adding our puppet classes we wanted to make the assignment of hosts into groups automated. To do this we start by installing the package «ruby-foreman-default-host» on our master server. When the package was installed we had to configure the plugin to fit our needs. This was done by editing the file: «/etc/foreman/plugins/default_hostgroup.yaml». The content of the file can be seen below:

```
:default_hostgroup:
  :facts_map:
    "Common/Loadbalancer":
      "hostname": "lb"
    "Common/Database":
      "hostname": "db*"
    "Common/WebserverNode":
      "hostname": "webnode"
    "Common/Webserver":
      "hostname": "web*"
    "Common/Memcache":
      "hostname": "mem"
```

Here you can see that all the hostgroups (including their parent) is listed as well as a regex to note which hosts to put in the group. Since the policy is first match found will be applied, it's important that webnode comes above web* in our list, if not our webserver that was supposed to have node.js installed would've ended up equal to the other webservers.

After a lot of tests, we couldn't add automated host group assignment. Some noted that you had to install the default host package through the foreman-installer instead of the apt-get method. When we did this, our foreman webpage didn't allow us access. After having almost no time left, we decided to skip this part and have our host groups to be added manually to every host.

5 Workflow test

5.1 Self-deployment (x3)

After setting up all our nodes in MLN, finishing our puppet master, and adding host groups with puppet classes in foreman we started testing our deployment with our deployment script.

The script ran as root and took an average 13 minutes and 05 seconds to complete the set up. The exact times of all the different parts of the test can be seen in the table below:

Test #	Script runtime	Time of hosts showing up in foreman	Time of application of policies	Total
Run 1	03m 11s	01m 53s	10m 27s	15m 31s
Run 2	03m 03s	02m 02s	10m 50s	15m 55s
Run 3	03m 07s	02m 05s	02m 37s	07m 50s
Average	03m 07s	2m 00s	07m 50s	13m 05s

Note: m-stands for minutes and s-stands for second

One can clearly see a decrease in time from run 2 to run 3, this was of the fact that we changed the puppet run interval on our hosts from 10 minutes to 2 minutes.

5.2 Junior's test

We prepared a manual, attached in the appendix, for our two juniors to deploy our infrastructure. Based on the manual, the first junior deployed the complete infrastructure in 19m 36s and the second junior deployed the infrastructure in 16m 03s. We can classify the total time into two stages. i.e. Stage 1 and 2. Stage 1 is the time from the junior first seeing the task in Trello to login onto the master server. Our first junior completed this stage in 6m 58s. The second junior finished just in 3m 58s. Stage 2 includes from running the deployment script to applying policies. our juniors completed in 12m 38s and 12 m 05s respectively.

It should be noted that junior one used a putty generated ssh-key which were new to us, so it took a little extra time for us to know how to add it correctly, giving junior one access to the master server. In addition to this, the puppet run interval was set to 5 minutes during the junior tests.

6 Conclusion

In cloud environment, it is common to create multiple virtual machines for providing services to customers. We used MLN to create our virtual machines. We wrote a script that created seven hosts through mln, all with the same basic settings. The main aim of these hosts was to provide the web services the customer wanted, after installing web based programming tools and infrastructures to the given hosts.

After creating the hosts, we created policies using puppet classes. The policy creation includes writing a puppet manifests which specify the type of web development tools to be installed at the different hosts. Based on the problem statement, we created the policies such as installing MySQL, PHP and all others on specifications to each corresponding servers. We also used foreman to visualize the host management and the policies applied to them.

We used both self-testing and Junior system administrators to test our deployment infrastructure. Self-testing was done by recording the time required to deploy the complete infrastructure by changing the puppet run time interval. The best time recorded for deploying our infrastructure was 07m 50s. The Juniors also deployed our infrastructure services.

In general, we have mainly learned the following from this assignment:

- Troubleshooting MLN, and Puppet errors
- Troubleshooting foreman errors
- Writing scripts that can simplify and minimize configuration steps
- Writing scripts that can troubleshoot foreman, and puppet failures
- A taste of working in a real cloud environment

7 Appendix

Junior Task

Prerequisite:

For you to be able to complete this task you must already have sent your public key to the senior system administrator.

Step 1 – Access the masternode:

Enter the puppet master via SSH:

```
ssh junior@128.39.121.75
```

Login as root with the password test, and go to root's home directory:

```
sudo su
```

```
[sudo] password for junior:
```

```
cd ~
```

Step 2 – Run the deployment script:

Run the script by typing:

```
./Deploy.sh
```

The script takes some time to complete so be patient.

If under any circumstance the script freezes between a Starting Task prompt and Finished task prompt (in other words if a key is pressed and there is no response), exit the node, repeat step 1, and re-run the script.

When the script prompts:

```
DEPLOYMENT FINISHED
```

Continue to the next step.

Step 3 – Access Foreman:

Login to the foreman webGUI with the address:

<https://128.39.121.75>

You might be prompted that the address isn't safe, if this is the case, click Advanced Settings and continue to the webpage.

Your login credentials are:

Username: junior

Password: jn_proj1_H17

Step 4 – Add the hosts to their respective HostGroups:

After successfully login into foreman, click on Hosts -> All Hosts.

If the only host there is the puppet master, give the system some time to synchronize.

After seeing all hosts (db1, db2, lb, mem, web1, web2, webnode) in the list, it is time to add them to their respective hostgroup.

To do this mark the host(s) you want to add the hostgroup to, and choose «Change group» from the dropdown menu, above the hostlist.

The hostnames corresponds to the different hostgroups as seen in the table below:

Hostname	Hostgroup
db1, db2	Common/Database
lb	Common/Loadbalancer
mem	Common/Memcache
web1, web2	Common/Webserver
webnode	Common/WebserverNode

Now go to Monitor -> Dashboard and wait for the policies to be applied.

Waiting for the hostgroups' policies to be applied may take a while (up to 5 minutes).

Feel free to take a break and look back regularly.

When «Latest events» shows a list of all the hosts with a blue square in the «applied» category, report back to your senior sysadmin. Your job is then complete.

Example picture of all seven nodes with their applied policies.

Thank you for your patience!

