



UiO : Universitetet i Oslo



# Infrastructure Services and Operations (INF4017NSA) Project 2 Report

Compiled by: Martin Veshovda Løland (s236323)  
Getinet Ayele Eshete(s326610)

November 17, 2017

# Table of contents

Table of contents	1
1 Introduction	2
2 Problem Statement	2
3 Overview of technologies used	2
5 Workflow test	5
5.1 Self-deployment test x 3	5
5.2 Junior test	5
5.3 Comparing results	7
6 Conclusion	8
7 Appendix	9
7.1 Deployment files	9
7.2 Junior test document	10

# 1 Introduction

In the second project we went for alternative 1 (described in problem statement) as our deployment environment. In addition to the deployment, there was a requirement to implement a central logging station, monitoring and backup of the system. In this report we'll describe how we completed the different tasks, a short overview of technologies used, our use of trello for workflow organization, and documentation of deployment tests done by ourselves and tests done by juniors.

We would like to note that this report includes a lot of pictures and files connected to the appendix which makes the total page number somewhat high.

## 2 Problem Statement

The problem statement for alternative 1 regards setting up a development environment. It contains the following nodes: Two storage servers with GlusterFS server installed, two development servers with emacs, jed, subversion and git installed, two compile servers with gcc, make and binutils installed. All nodes should have four users with the names "Alice", "Bob", "Janet" and "Tim". They all need to have root access via sudo and Tim and Janet should be members of the group "developers".

In addition all nodes need to be monitored. Whether puppet executed successfully last run should be a part of what is monitored. All nodes need to send log messages to a central logging server. Lastly all of the nodes need to be connected to a backup system.

## 3 Overview of technologies used

### **MLN:**

MLN lets the user create multiple nodes at the same time, with specific settings for each node.

### **Puppet:**

Puppet is our means of configuration management. Puppet uses idempotent manifests to make sure a node is in a specific state. Puppet was used to apply extra packages and settings for each node, beyond what was specified in the MLN project.

### **Foreman:**

Foreman is a webGUI for puppet and was used to apply our puppet policies.

### **Git and Github:**

Git is a version control system that allows multiple users to edit files while making sure none of the changes are overwritten unintentionally. Git was mainly used together with github for off-site storing of files that could easily be downloaded where needed.

### **FoxyProxy:**

FoxyProxy is a proxy extension used in our webbrowser. This allows us to tunnel our internet connection through our site and access nodes with only local IP addresses.

## ELK-stack:

ELK-stack consists of ElasticSearch, LogStash and Kibana. Together they make a powerful logging instance where LogStash converts the received log information as JSON files while ElasticSearch is the logging database backend while Kibana serves as a front end webGUI.

## Sensu:

Sensu is a lightweight push based monitoring system. Sensu performs monitoring through checks. We created a sensu master through hiroakis' docker setup, while all the clients applied a modified version of Kyrre Begum's sensu client installation manifest.

## 4 Workflow design and Implementation

We created a list in trello dedicated to Project 2. From here we added all of our different parts of the workflow that we were working on, with a checklist to note which part of the element that was finished. Any problems were noted on the card associated with the problem. Here you can see a screenshot of our trello list.

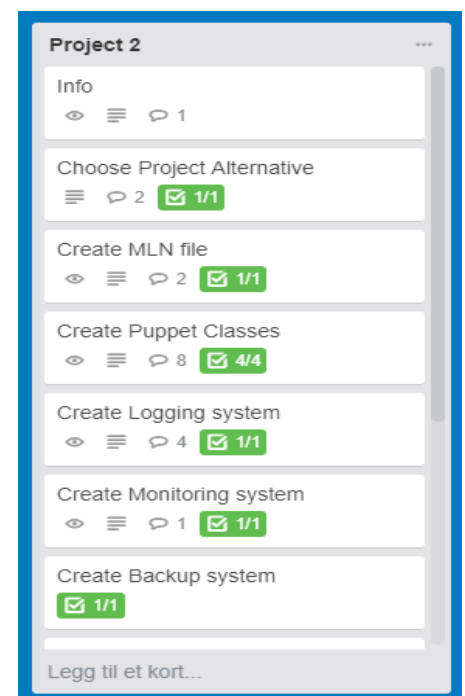
We wanted to make the workflow easy and completed in as few steps as possible. We divided the workflow into different parts:

- Creating the nodes with necessary packages (MLN)
- Apply necessary puppet manifests (Puppet & Foreman)
- Add nodes to logging system (rsyslog)
- Add nodes to monitoring system (Sensu-client)
- Add nodes to backup system

After completing the last project the first two parts were completed rather quickly. The MLN file uses a superclass with an extensive user\_data block to preinstall puppet and git for all new nodes. This means that all needed to get the nodes up and running is building the mln file and start the mln project.

To apply the necessary puppet manifests we created a puppet class for each package that were to be installed on the different nodes, and a common class for all items that were needed on all nodes (users, sensu-client and so on). These on the other hand does not apply themselves. We therefore used foreman to import the master puppet classes and add them to hostgroups. To make sure the hosts showed up in foreman they needed their certificates signed. This was simply done by issuing the command "puppet cert sign --all" on the master node. Lastly the nodes were put in the different hostgroups by assigning them manually in foreman (our default\_host plugin did not work).

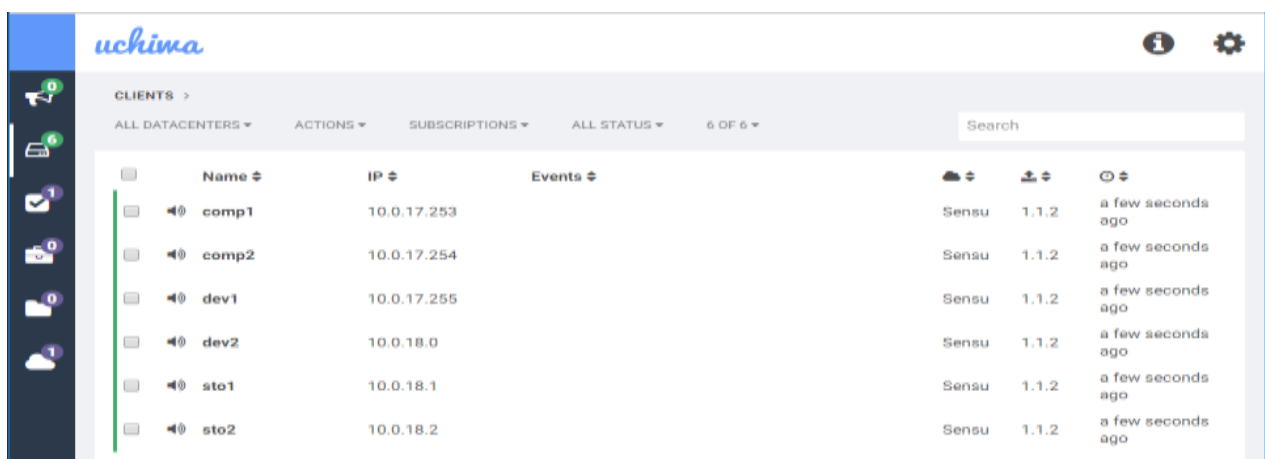
To add a node to the logging system we created a puppet manifest called "rsyslog" which contained a method to create the file 51-remotelogging.conf with the content "@<our\_ELK\_server\_ip>:514". This puppet class made the application of the minimal logging requirement automatically when the node was added to its corresponding hostgroup in foreman.



The monitoring requirement was solved the same way as our logging requirement, through puppet classes. The puppet class applied a modified version of Kyrre Begum's sensu-client installation manifest (can be seen here: <https://tinyurl.com/gm-sensu>), then added the sensu master certificate and key.

When we had all these pieces we decided to connect them automatically. We created a deployment script which would make all of the steps above connected to one single command. The script starts by stopping the mln project for our deployment and then removing it. This is to make sure that every time we test a new deployment, we don't have to do this manually and only focus on the setup part. We made a loop which iterates all the nodes in the command "nova list" which contains ".p2" (our mln project name) in their name, and then print the 6th column (awk 'printf "%s\n",\$6}') which contains the status of the node. If the status was ACTIVE the script would iterate the nodes again until they all were shut off. This was to make sure all nodes would be deleted correctly, since "mln remove -p <projectname>" will skip removing running nodes. After removing the mln project we created a loop to remove all signed certificates (except the master server certificate!) for puppet. When these were removed, we built the mln file and started the project after its build were completed. We then made a while true loop to look for puppet certificate requests. The loop was then broken when a counter reached the correct amount of certificates signed (in our case 6). When this was completed we added our backup method. Again we made a loop listing all lines from the "nova list" command. All nodes containing "p2" in their name would then have their 12th column added (this is their IP address) to a file called "bkup\_plan.conf". This file was then sent to our backup server through scp and used as the list containing servers to backup. Lastly our script checked for the nodes in foreman, and when all were found a test were made if they were in a hostgroup (they would be if we forgot to remove them from foreman before running the script). If they didn't have a hostgroup they would be automatically assigned to their correct hostgroup by the use of case.

With this script our workflow was reduced to pretty much running the script. To make sure everything is working as intended one might check the web interfaces of kibana and uchiwa as additional workflow steps. Below is a picture of all nodes non-critical in uchiwa.



Name	IP	Events
comp1	10.0.17.253	Sensu 1.1.2 a few seconds ago
comp2	10.0.17.254	Sensu 1.1.2 a few seconds ago
dev1	10.0.17.255	Sensu 1.1.2 a few seconds ago
dev2	10.0.18.0	Sensu 1.1.2 a few seconds ago
sto1	10.0.18.1	Sensu 1.1.2 a few seconds ago
sto2	10.0.18.2	Sensu 1.1.2 a few seconds ago

## 5 Workflow test

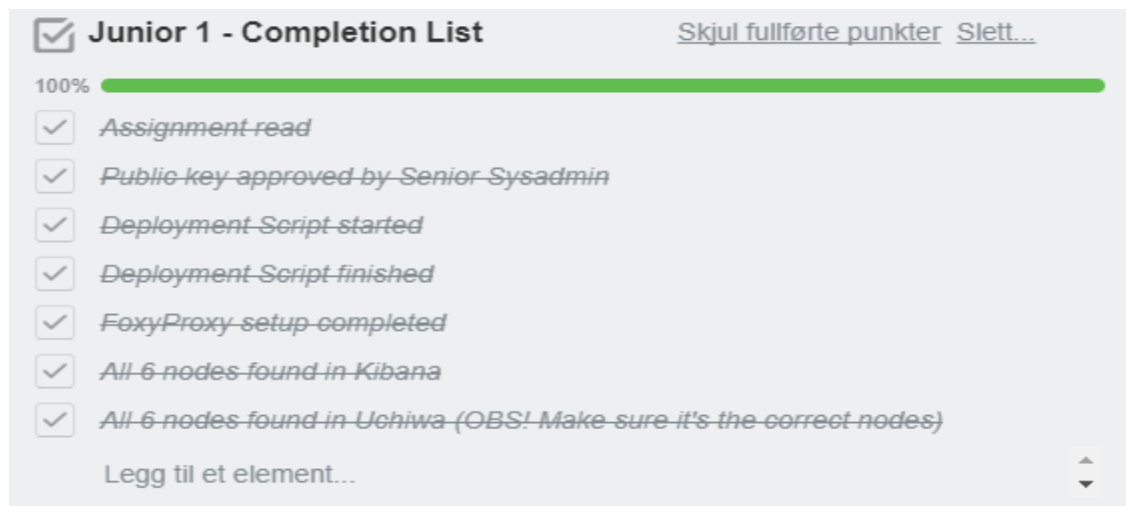
### 5.1 Self-deployment test x 3

This task was completed by timing the execution of the script, to all nodes showing up in uchiwa.

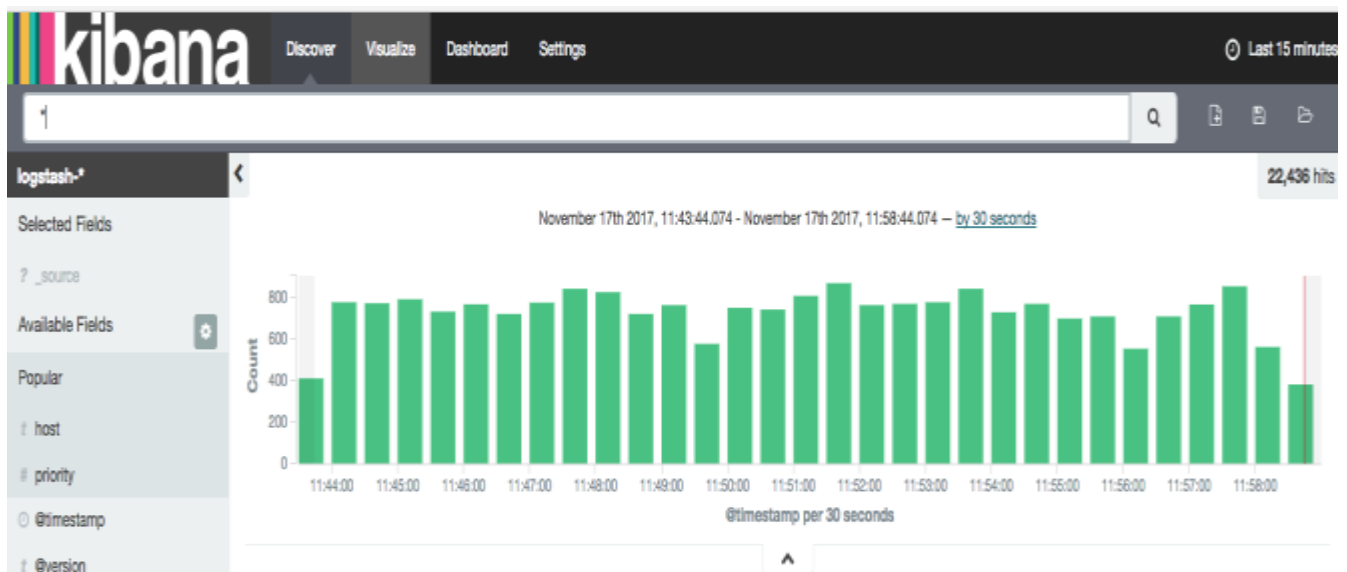
Test #	Script runtime	Wait time before hosts showup in uchiwa	Total
Run 1	3:03	4:18	7:21
Run 2	2:56	4:13	7:09
Run 3	3:02	4:18	7:20
Average	3:00	4:16	7:17

### 5.2 Junior test

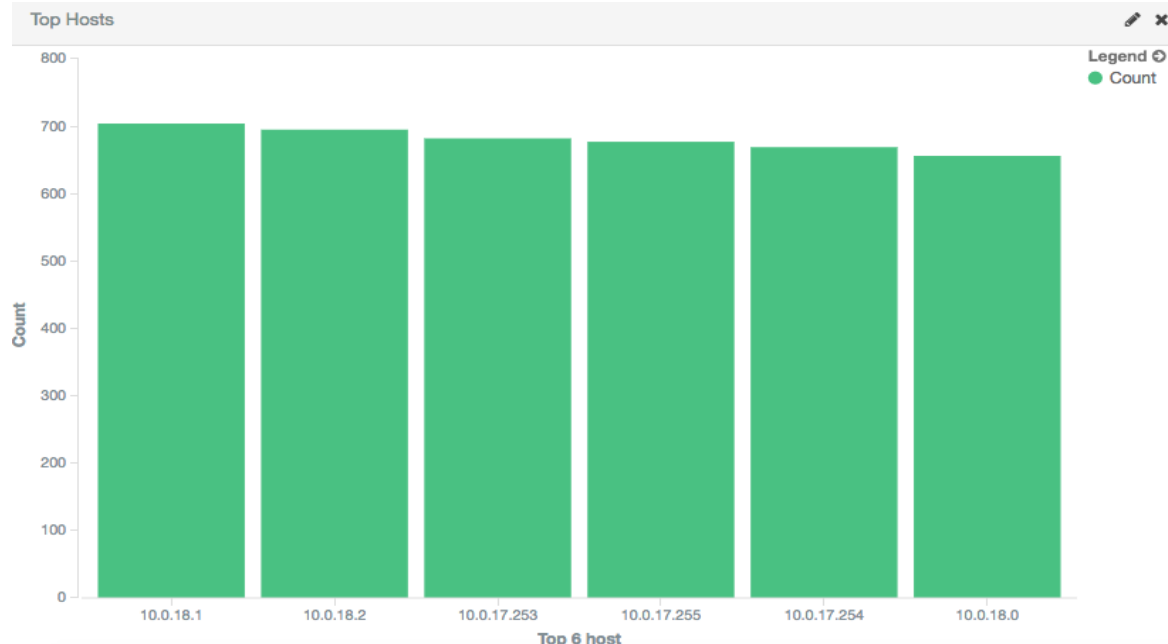
Our juniors were given access to our trello board, with a WIP card for each of them. In this card they got information on where to begin and were then opted to check off tasks as they were completed. In addition they got a descriptive document showing how to complete the task, from logging into the master node, to checking the web interfaces for errors (this can be seen in the appendix). Below is a picture of the list used to check off their progress.



**Junior 1** completed task 1 Assignment read at 10.23 (this task is mainly to time the start of the workflow). It then took 7 minutes to add the public key for ssh (this step took some time longer than expected because the seniors weren't quite prepared). The script was started at 10.30 and was finished 10.31. FoxyProxy setup was then completed a minute later (10.32). All nodes were found in both kibana and uchiwa at 10.36. Junior 1 asked two questions: if he were to create a public key for the test and how to identify the hosts in kibana. Since some of the elements of the junior test were written as if it was a real company assignment, it might be a bit confusing. The second question could have been avoided by modifying our kibana. We had minimal logging and filtering applied which made the page somewhat confusing of what to look for. Below is a picture of our kibana. This is not very self-explanatory.



To actually see which nodes are sending the log messages one would have to click on "Available Fields" and from there choose hosts. Below is a picture of our kibana after clicking on hosts field.



**Junior 1** completed the assignment in 13 minutes. 7 of these minutes were used to give the junior access to the master server while the actual deployment only took 6 minutes.

**Junior 2** completed task 1 at 10.43. It then took 3 minutes to add the public key for ssh (10.46). The script was started at 10.47 and finished 10.51. FoxyProxy setup was completed at 10.51. And nodes were found in both kibana and uchiwa at 10.56. Junior 2 asked no questions.

**Junior 2** completed the assignment in 13 minutes. Only 3 of these minutes were used to gain access to the master server, while the actual deployment took 10 minutes.

### 5.3 Comparing results

As seen from our self deployed test and our junior tests we see a variation in the time of the deployment. The difference from our test and junior 1 is 1 minute in favor of the junior and the difference between our test and junior 2 is 4 minutes in favor of our test. It is somewhat hard to say 100% sure if the deployment time of junior 1 and junior 2 are correct considering the junior might have forgotten to check the checklist at the exact time of completion. On average of all the tests the time of the complete deployment was 7 minutes and 46 seconds.

Apart from our minor configuration of Kibana, we feel our execution of the workflow were quite easy to complete and understand.



## 6 Conclusion

After finishing the project we are very happy with how our workflow is executed, but not quite happy with the finishing touch. We used too much time on some tasks which could have been used to improve our monitoring, logging and backup systems. In the start of the project we tried to create our whole deployment script as a puppet manifest, but this task became a bit too complicated considering our time limitation. Secondly we struggled to make sure our nodes showed up in uchiwa, which took up a healthy amount of our time.

Our finished product completes all the minimal requirements of the project and our script uses a lot of color and feedback to the user for a better experience when deploying the environment. It's easy to complete and our scripts are well explained with comments if the user were to read the script. For future work there are a few improvements:

- The backup-pull script does not have root access and can therefore not copy any directories where root is needed.
- At the moment there is no automatic cleaning of the web interfaces like removing the hosts from uchiwa or removing log messages from old nodes in kibana.
- There is a bug in the script that checks if puppet was executed properly.
- Add more monitoring checks.
- Add more log messages.
- Add a dashboard with easy to understand graphs in kibana.

## **7 Appendix**

### **7.1 Deployment files**

#### **MLN File**

<https://tinyurl.com/gm-mln>

#### **Deployment script**

<https://tinyurl.com/gm-depl>

#### **ELK setup file**

<https://tinyurl.com/gm-elk-setup>

#### **Sensu Client setup file**

<https://tinyurl.com/gm-sensu>

#### **Sensu puppet check script**

<https://tinyurl.com/gm-puppet-check>

## 7.2 Junior test document

### Junior Task

#### Prerequisite:

To be able to complete this task you must already have sent your public key to the senior sysadmin, and have gotten his approval to continue. You will also need a functioning proxy. We suggest using FoxyProxy.

#### STEP 1 – ACCESS THE MASTER NODE

Enter the master server via SSH with a port tunnel. The tunnel will later be used with FoxyProxy, so if you already have a working proxy we suggest you use the same port as in your proxy settings.

```
ssh -D <Your-Port-Choice> junior@128.39.121.75
```

Login as root with the password: test

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

#### STEP 2 – RUN THE DEPLOYMENT SCRIPT:

Run the script by typing:

```
./DepEnv.sh
```

You can now check the point «Deployment Script started» in Trello.

The script takes some time to complete so be patient.  
When the script prompts:

```
DEPLOYMENT COMPLETED
```

You can now check the point «Deployment Script finished» in Trello.

Continue to the next step.

#### STEP 3 – FOXYPROXY:

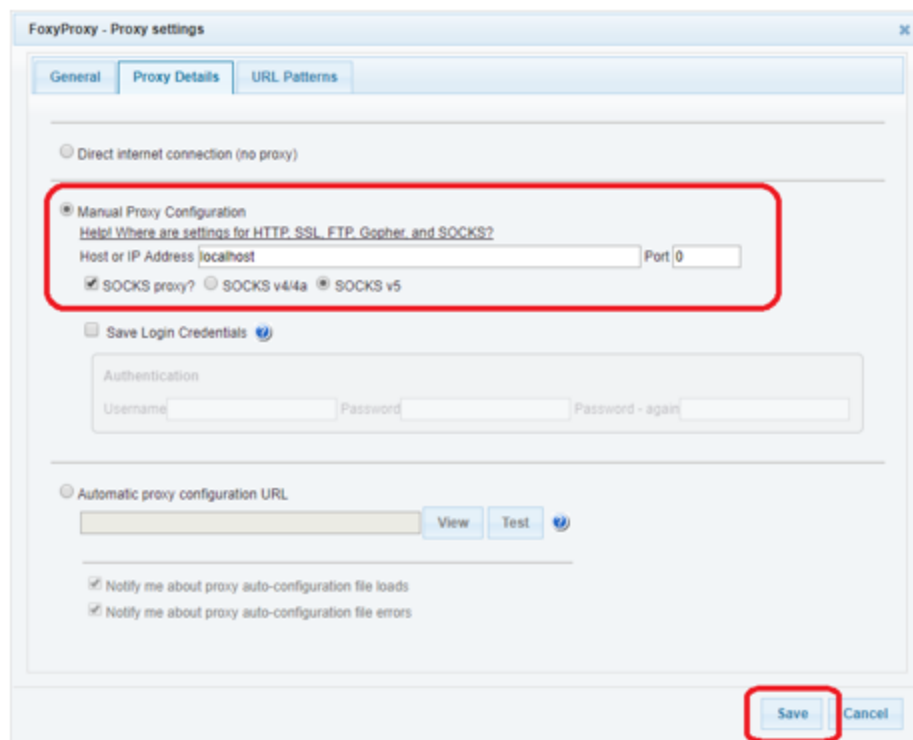
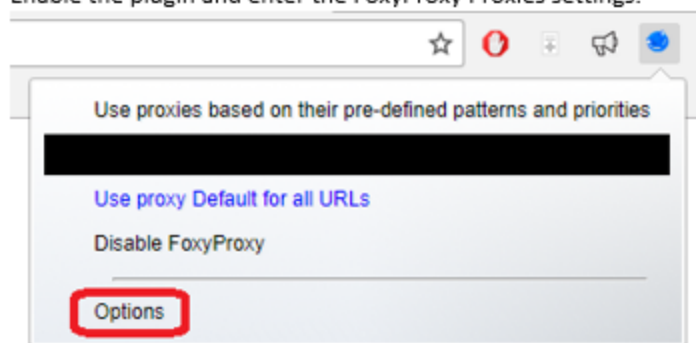
If you have a working FoxyProxy setting, you can skip this step.

If your FoxyProxy is working, check «FoxyProxy setup completed» in Trello.

If you don't have FoxyProxy installed, download the plugin for your preferred browser.

The guide below shows the steps done in google chrome. There might be minor deviations if you use another browser.

Enable the plugin and enter the FoxyProxy Proxies settings:



Enter «localhost» in the «Host or IP Address» field and in the «Port» field choose the same port that you used as a tunnel when ssh'ing into the master node. Check «SOCKS proxy», choose version 5 and then click save. You should now see your Proxy in the list in FoxyProxy.

You can now check the point «FoxyProxy setup completed» in Trello, unless you already did earlier.

## STEP 4 – KIBANA:

Go to the following website in your browser and make sure you have activated your Proxy:

<http://10.0.16.15:5601>

From here select the dashboard tab. You will see a «Top 20» bar chart.

At the same time on the master node run in the following script:

```
./listHosts.sh
```

This script will list the newly deployed nodes together with their IP.

If the IPs are consistent with the ones in kibana, the newly deployed nodes are successfully logging.

**NOTE:** You might see some old nodes in the kibana dashboard, just ignore these and focus on the nodes matching the ones from the script.

If none of the IP addresses listed on the master node is in Kibana, contact your senior sysadmin for further instructions.

You can now check the point «All 6 nodes found in Kibana» in Trello.

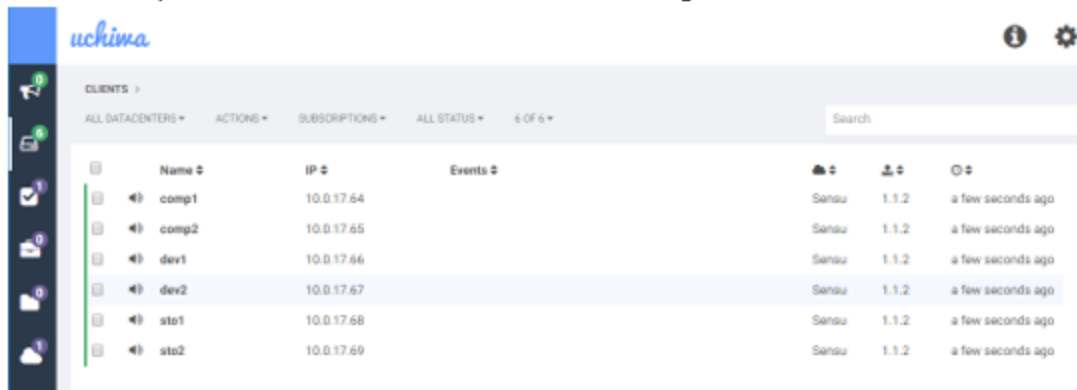
You can now continue to the next step.

## STEP 5 – UCHIWA:

Go to the following website in your browser and make sure you have activated your Proxy:

<http://10.0.16.16:3000>

Under clients you should see all nodes. It should look something like this:



The screenshot shows the Uchiwa web interface. The header includes the 'uchiwa' logo and navigation icons. Below the header, there's a 'CLIENTS' section with tabs for 'ALL DATACENTERS', 'ACTIONS', 'SUBSCRIPTIONS', 'ALL STATUS', and '6 OF 6'. A search bar is present. The main content is a table with columns: Name, IP, Events, and three columns for status (represented by icons). The table lists six clients: comp1, comp2, dev1, dev2, sto1, and sto2, all with IP addresses in the 10.0.17.x range and 'Senou' as the status.

Name	IP	Events	Status 1	Status 2	Status 3
comp1	10.0.17.64		Senou	1.1.2	a few seconds ago
comp2	10.0.17.65		Senou	1.1.2	a few seconds ago
dev1	10.0.17.66		Senou	1.1.2	a few seconds ago
dev2	10.0.17.67		Senou	1.1.2	a few seconds ago
sto1	10.0.17.68		Senou	1.1.2	a few seconds ago
sto2	10.0.17.69		Senou	1.1.2	a few seconds ago

If this is not the case, contact your senior sysadmin.

**NOTE:** It takes some time for the nodes to show up in Uchiwa. If our senior sysadmin did a sloppy job you will see some old nodes listed with a keepalive warning. As mentioned, these are OLD nodes, be patient and the new ones will show up eventually.

You can now check the point «All 6 nodes found in Uchiwa» in Trello.

**Congratulations you have completed the deployment!**