Nokia uMEC project, Hardware team

Esko Aalto, Joakim Kurvinen

Introduction

This project was a collaboration between Metropolia university of applied sciences and Nokia. The objective was to create a simulated environment for microMEC devices by using Raspberry Pi's and Linux hardware servers. There were four teams and about ten people involved. The API team was responsible of creating an API to take pictures from a camera attached to one of the Raspberry Pi servers and then then run the result through a Tensorflow algorithm to recognize moose from the pictures. The continuous integration team was responsible of setting up a CI system for the project and the performance team was responsible of setting up tools to monitor the system.

Our team was responsible for setting up the hardware, operating systems, network and user accounts. On top of this we were responsible of system administration and general troubleshooting and support.

The project proceeded by meeting up with representatives from Nokia each week. In these "sprint meetings" our goals were set for the week. We would work with the given goal until we would get new goals the next week. The project lasted about two months.

Acquiring Hardware

The first thing our team did was acquiring the hardware. Nokia provided us with a server cabinet and 3 Air Frame servers. The challenge was to move the cabinet from the Nokia building to the Metropolia building. Because of its size it had to be moved by a pallet jack. Unfortunately moving the cabinet through the Nokia building was not an easy task since our escort didn't have access to some of the essential forms of transportation needed to move the cabinet, mainly elevators. We got around this problem by waiting inside the elevator with the cabinet while our escort climbed the stairs and called the elevator to the appropriate floor.

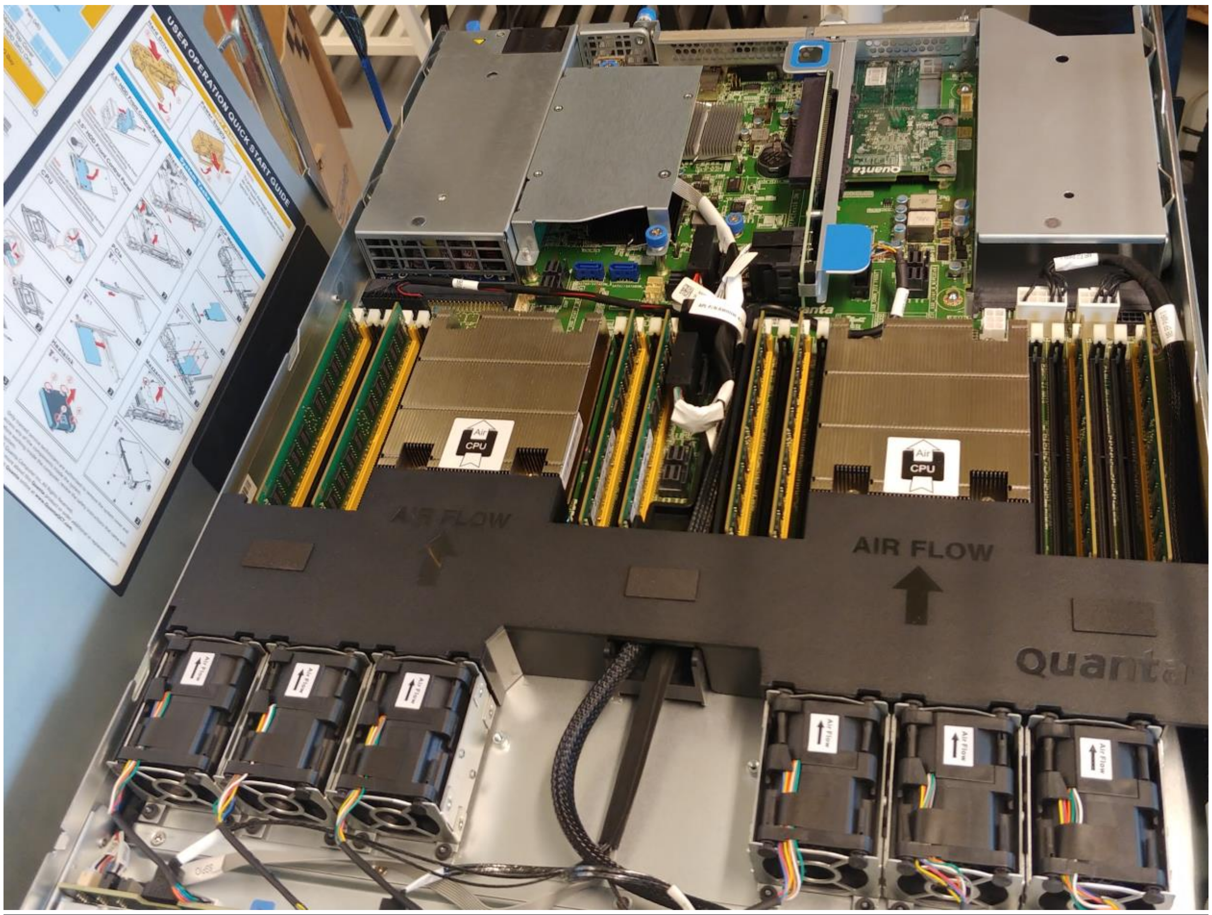
This gave us access to the overpass connecting the buildings. Unfortunately once we got to the other side we realized that the cabinet didn't fit through the door. We called a security guard to open the extension to the door but he didn’t have a key for it. Ultimately we were forced to go all the way back to the Nokia building, do the same maneuver in reverse with the elevator as before and drag the cabinet all the way to the other end of the building. From there we were able to go outside with the cabinet and drag it through a gravel road to a cargo elevator in the Metropolia building.

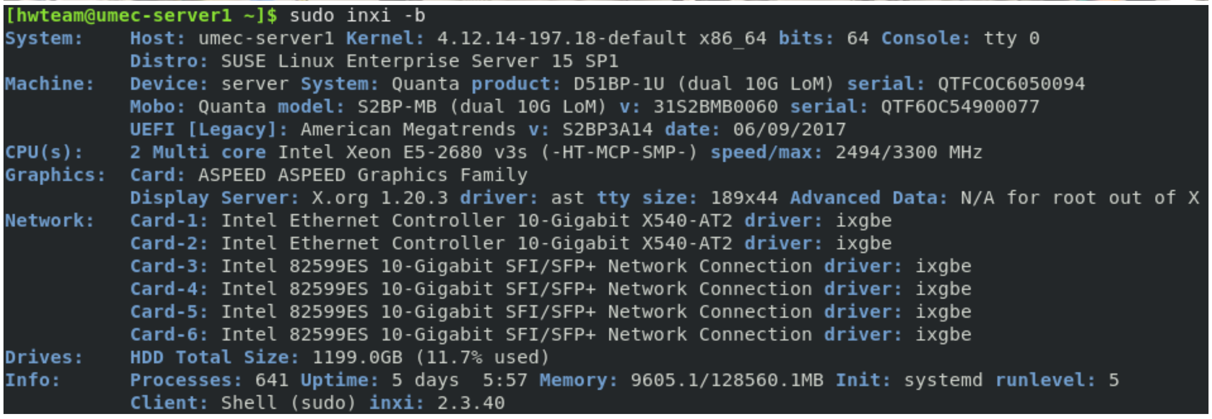
Obviously we didn’t have access to said elevator so we needed to ask the security guard to open up the elevator and give us access to the floor we were heading. Ultimately when we got to the room assigned to our server we found out that the server didn’t fit through the door. At this point we decided to just set up our server in a secluded area that was used as a storage for all kinds of office supplies.

After this our teacher gave us a ride to our old school building to get more essential equipment. We got a UPS, NAS, layer 3 switch, a PC, monitors, mouses, keyboards, cables and a heat sensor. After barely fitting all the hardware in the car we drove back and started the installation procedure.

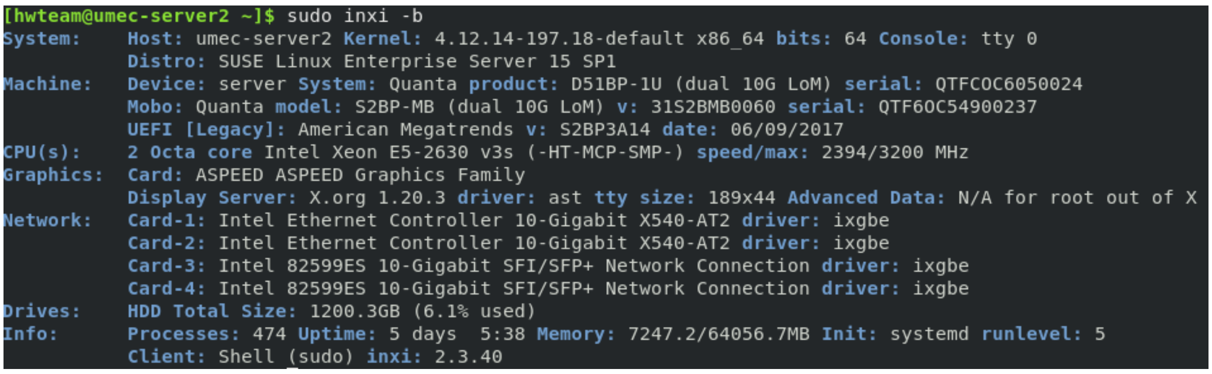
The Hardware

Nokia gave us 3 Air Frame servers

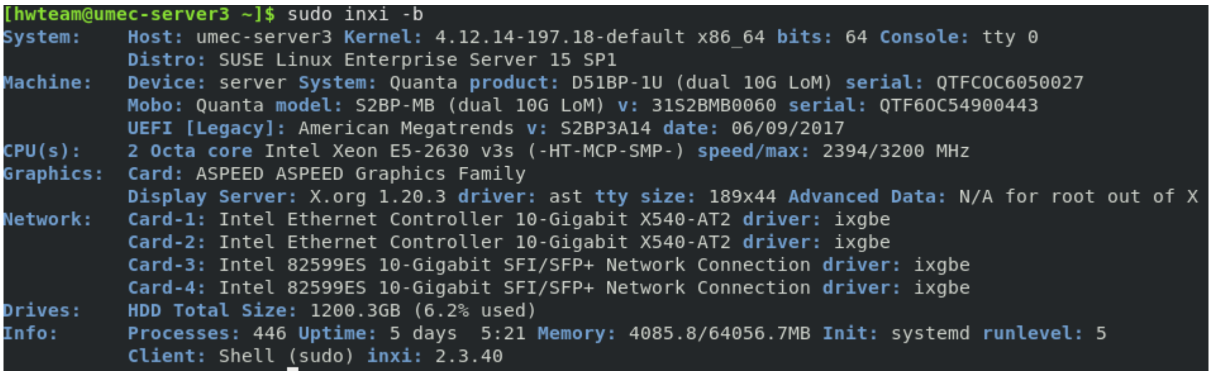




```
[hwteam@umec-server1 ~]$ sudo inxi -b
System:    Host: umec-server1 Kernel: 4.12.14-197.18-default x86_64 bits: 64 Console: tty 0
           Distro: SUSE Linux Enterprise Server 15 SP1
Machine:   Device: server System: Quanta product: D51BP-1U (dual 10G LoM) serial: QTFCOC6050094
           Mobo: Quanta model: S2BP-MB (dual 10G LoM) v: 31S2BMB0060 serial: QTF60C54900077
           UEFI [Legacy]: American Megatrends v: S2BP3A14 date: 06/09/2017
CPU(s):    2 Multi core Intel Xeon E5-2680 v3s (-HT-MCP-SMP-) speed/max: 2494/3300 MHz
Graphics:  Card: ASPEED ASPEED Graphics Family
           Display Server: X.org 1.20.3 driver: ast tty size: 189x44 Advanced Data: N/A for root out of X
Network:   Card-1: Intel Ethernet Controller 10-Gigabit X540-AT2 driver: ixgbe
           Card-2: Intel Ethernet Controller 10-Gigabit X540-AT2 driver: ixgbe
           Card-3: Intel 82599ES 10-Gigabit SFI/SFP+ Network Connection driver: ixgbe
           Card-4: Intel 82599ES 10-Gigabit SFI/SFP+ Network Connection driver: ixgbe
           Card-5: Intel 82599ES 10-Gigabit SFI/SFP+ Network Connection driver: ixgbe
           Card-6: Intel 82599ES 10-Gigabit SFI/SFP+ Network Connection driver: ixgbe
Drives:    HDD Total Size: 1199.0GB (11.7% used)
Info:      Processes: 641 Uptime: 5 days  5:57 Memory: 9605.1/128560.1MB Init: systemd runlevel: 5
           Client: Shell (sudo) inxi: 2.3.40
```

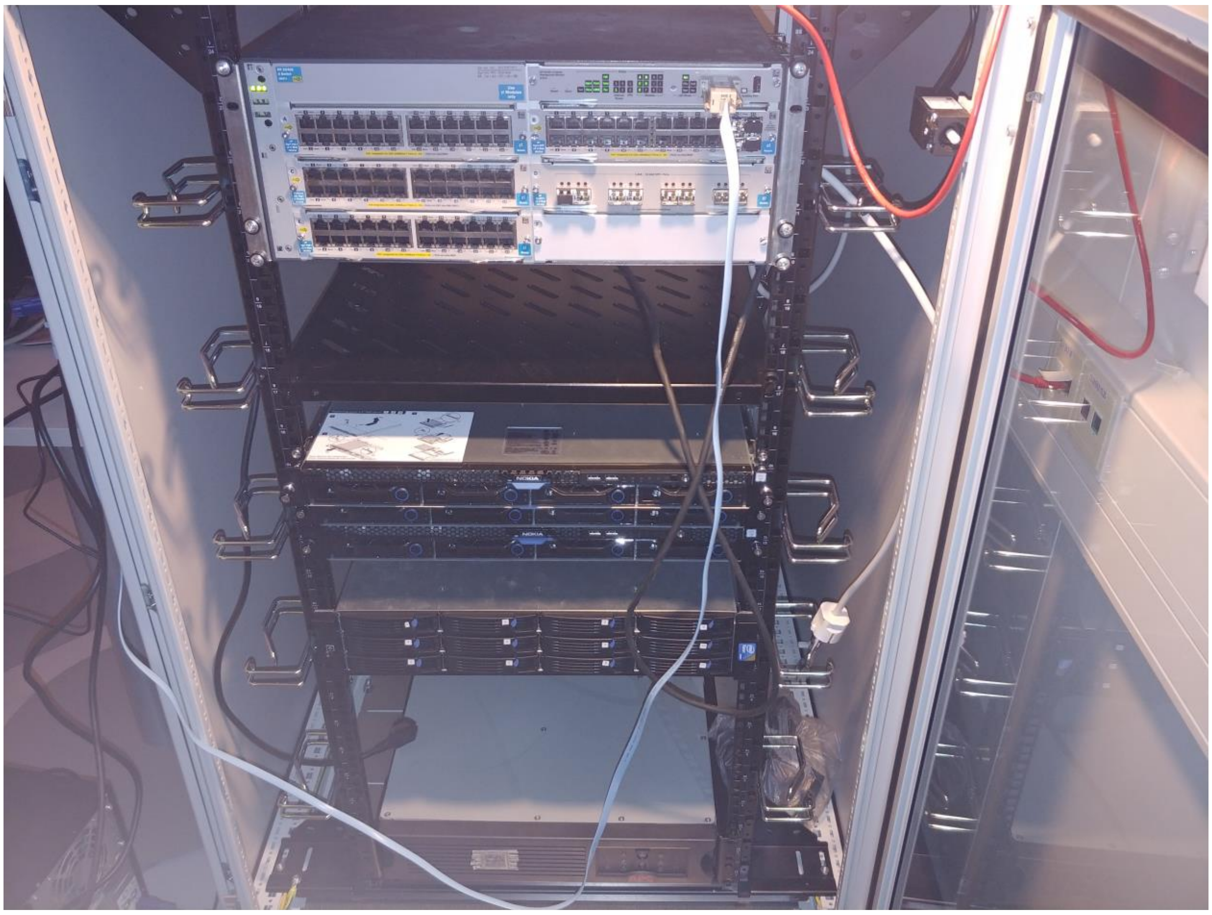


```
[hwteam@umec-server2 ~]$ sudo inxi -b
System:    Host: umec-server2 Kernel: 4.12.14-197.18-default x86_64 bits: 64 Console: tty 0
           Distro: SUSE Linux Enterprise Server 15 SP1
Machine:   Device: server System: Quanta product: D51BP-1U (dual 10G LoM) serial: QTFCOC6050024
           Mobo: Quanta model: S2BP-MB (dual 10G LoM) v: 31S2BMB0060 serial: QTF60C54900237
           UEFI [Legacy]: American Megatrends v: S2BP3A14 date: 06/09/2017
CPU(s):    2 Octa core Intel Xeon E5-2630 v3s (-HT-MCP-SMP-) speed/max: 2394/3200 MHz
Graphics:  Card: ASPEED ASPEED Graphics Family
           Display Server: X.org 1.20.3 driver: ast tty size: 189x44 Advanced Data: N/A for root out of X
Network:   Card-1: Intel Ethernet Controller 10-Gigabit X540-AT2 driver: ixgbe
           Card-2: Intel Ethernet Controller 10-Gigabit X540-AT2 driver: ixgbe
           Card-3: Intel 82599ES 10-Gigabit SFI/SFP+ Network Connection driver: ixgbe
           Card-4: Intel 82599ES 10-Gigabit SFI/SFP+ Network Connection driver: ixgbe
Drives:    HDD Total Size: 1200.3GB (6.1% used)
Info:      Processes: 474 Uptime: 5 days  5:38 Memory: 7247.2/64056.7MB Init: systemd runlevel: 5
           Client: Shell (sudo) inxi: 2.3.40
```

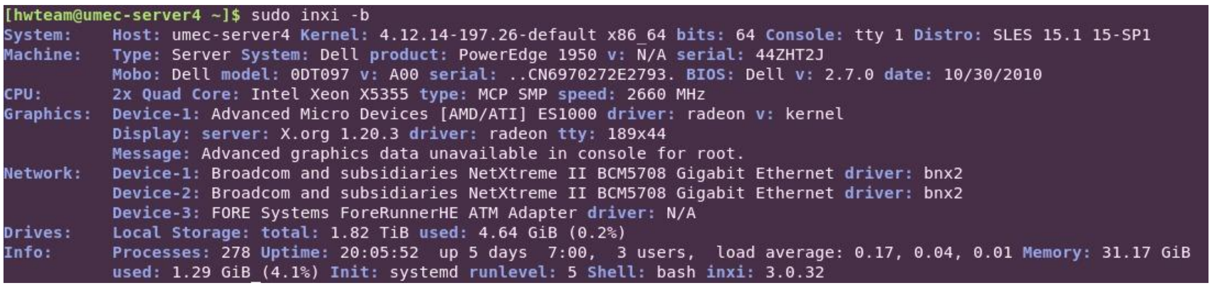


```
[hwteam@umec-server3 ~]$ sudo inxi -b
System:    Host: umec-server3 Kernel: 4.12.14-197.18-default x86_64 bits: 64 Console: tty 0
           Distro: SUSE Linux Enterprise Server 15 SP1
Machine:   Device: server System: Quanta product: D51BP-1U (dual 10G LoM) serial: QTFCOC6050027
           Mobo: Quanta model: S2BP-MB (dual 10G LoM) v: 31S2BMB0060 serial: QTF6OC54900443
           UEFI [Legacy]: American Megatrends v: S2BP3A14 date: 06/09/2017
CPU(s):    2 Octa core Intel Xeon E5-2630 v3s (-HT-MCP-SMP-) speed/max: 2394/3200 MHz
Graphics:  Card: ASPEED ASPEED Graphics Family
           Display Server: X.org 1.20.3 driver: ast tty size: 189x44 Advanced Data: N/A for root out of X
Network:   Card-1: Intel Ethernet Controller 10-Gigabit X540-AT2 driver: ixgbe
           Card-2: Intel Ethernet Controller 10-Gigabit X540-AT2 driver: ixgbe
           Card-3: Intel 82599ES 10-Gigabit SFI/SFP+ Network Connection driver: ixgbe
           Card-4: Intel 82599ES 10-Gigabit SFI/SFP+ Network Connection driver: ixgbe
Drives:    HDD Total Size: 1200.3GB (6.2% used)
Info:      Processes: 446 Uptime: 5 days  5:21 Memory: 4085.8/64056.7MB Init: systemd runlevel: 5
           Client: Shell (sudo) inxi: 2.3.40
```

All of the servers have 10-Gigabit connectivity and 2 Octa-core Intel Xeon processors. The first server has 128GB of memory and the second and third 64GB, all DDR4



The fourth server installed later has 2 quad-core Xeon processors with DDR3 memory



```
[hwteam@umec-server4 ~]$ sudo inxi -b
System:    Host: umec-server4 Kernel: 4.12.14-197.26-default x86_64 bits: 64 Console: tty 1 Distro: SLES 15.1 15-SP1
Machine:   Type: Server System: Dell product: PowerEdge 1950 v: N/A serial: 44ZHT2J
           Mobo: Dell model: 0DT097 v: A00 serial: ..CN6970272E2793. BIOS: Dell v: 2.7.0 date: 10/30/2010
CPU:       2x Quad Core: Intel Xeon X5355 type: MCP SMP speed: 2660 MHz
Graphics:  Device-1: Advanced Micro Devices [AMD/ATI] ES1000 driver: radeon v: kernel
           Display: server: X.org 1.20.3 driver: radeon tty: 189x44
           Message: Advanced graphics data unavailable in console for root.
Network:   Device-1: Broadcom and subsidiaries NetXtreme II BCM5708 Gigabit Ethernet driver: bnx2
           Device-2: Broadcom and subsidiaries NetXtreme II BCM5708 Gigabit Ethernet driver: bnx2
           Device-3: FORE Systems ForeRunnerHE ATM Adapter driver: N/A
Drives:    Local Storage: total: 1.82 TiB used: 4.64 GiB (0.2%)
Info:      Processes: 278 Uptime: 20:05:52  up 5 days  7:00,  3 users,  load average: 0.17, 0.04, 0.01 Memory: 31.17 GiB
           used: 1.29 GiB (4.1%) Init: systemd runlevel: 5 Shell: bash inxi: 3.0.32
```

Since there was a SUSE representative involved in the project, he suggested that we install SUSE Linux Enterprise Server on the servers.

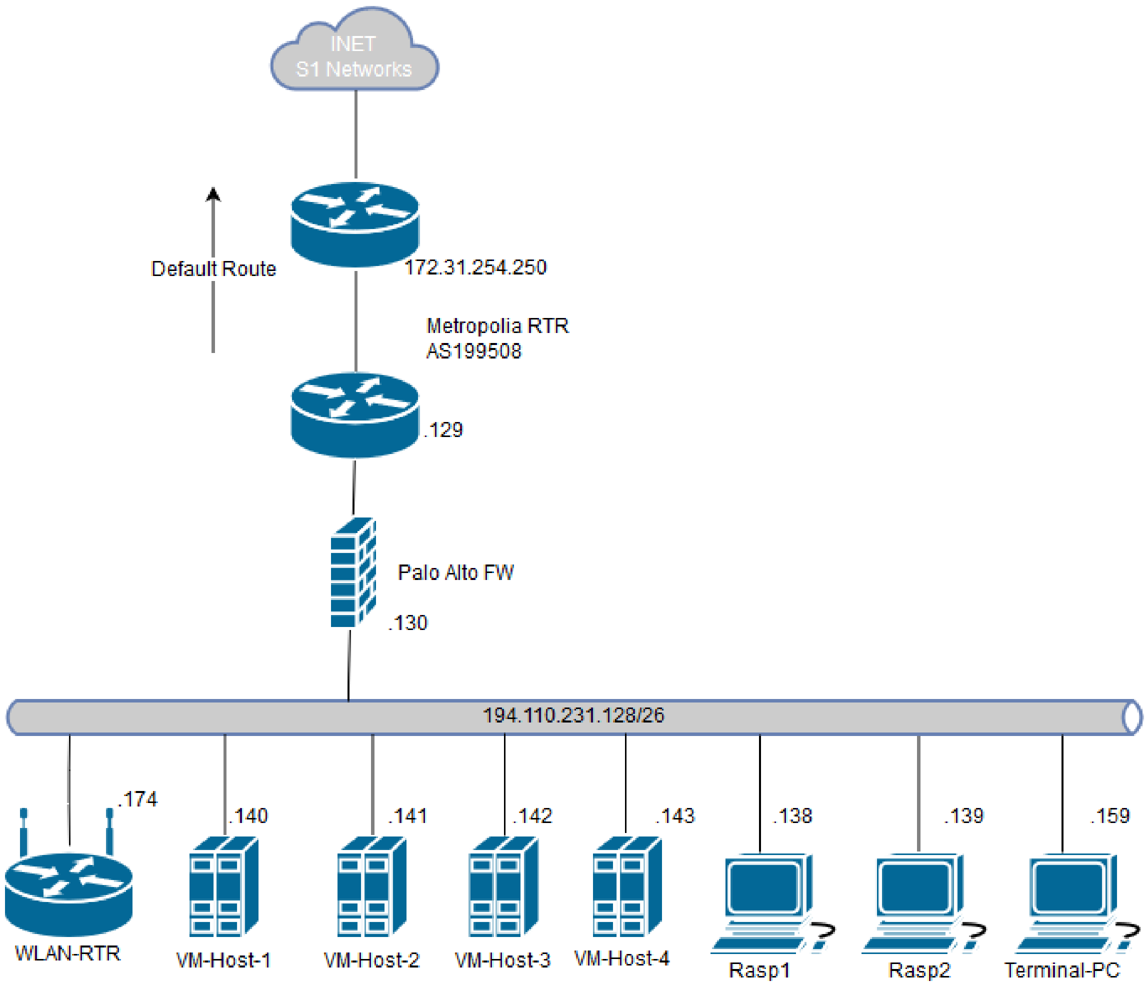
The switch is a HP layer 3 E5406 ZL series switch with both gigabit and 10 gigabit ports, as well as SFP+ ports.

The network

We were given the address range of 194.110.231.192/26. This gave us 62 public host IP addresses in the range of 194.220.231.129-191 to use with the first reserved for the default gateway. The addresses from 160 onward are distributed by DHCP.

We decided to use static IP addresses for our servers beginning with the address 194.110.231.140. The second server is 141, third 142 and fourth 143. The raspberry Pi servers start from 194.110.231.139 and decrement by one with each of the servers.

Our console PC was left with dynamic IP configuration since a static IP was not needed for the machine in question. This also left plenty of room for other hosts to possibly join the network if needed. Shown in the figure is the logical topology of the network.



We also got a generic wlan router with a couple ports towards the end of the project for use, as some of the other teams' members took to working in the server space, but had issues with connectivity to the eduroam wireless network. As such, we installed the wlan router, connecting it with a standard RJ-45 cable to the HP switch, reset it, and set up our own wireless network with capabilities to also link to it with ethernet. The other teams' members reported successful connectivity.

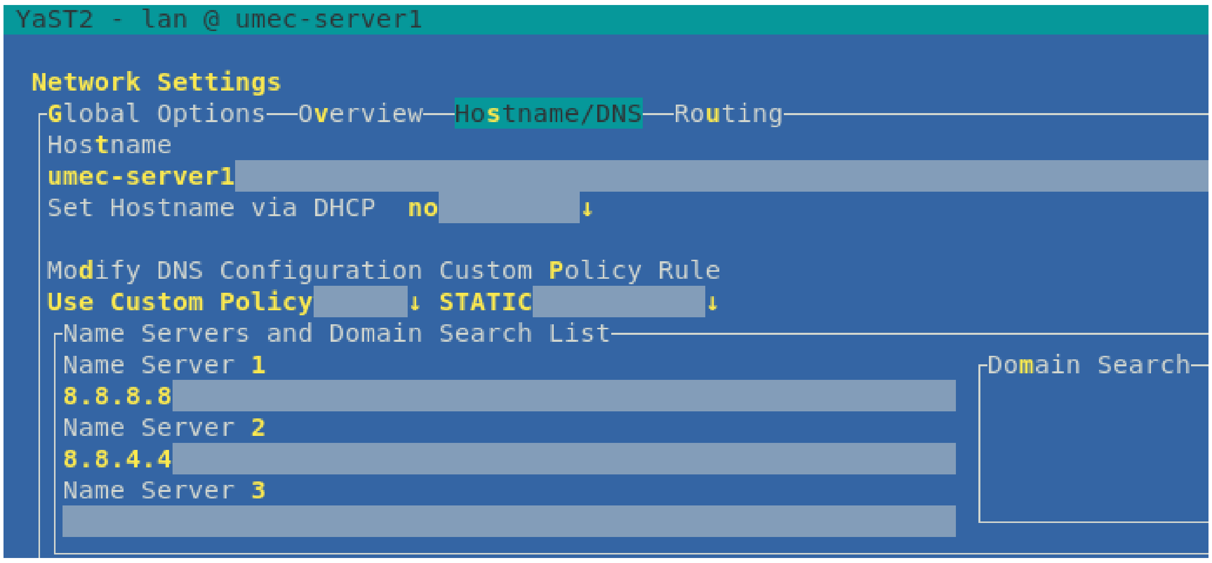
Backups

Backups of the servers' hard drives were taken by using an external hard drive at the servers themselves. Clonezilla-amd64 software was loaded onto a USB flash drive using Rufus. The servers were then rebooted into bios, the startup order of devices changed to the flash drive, and Clonezilla ran automatically. The backups were taken using the beginner options where the user only has to specify the target partition on a list of device partitions, and the hard drives the user wishes to backup from. The backups were in the format of Clonezilla disk images.

Software setup

In the servers we installed SUSE Linux Enterprise Server. At first we used a 60 day trial license but later on got a real license from school. The servers were pre-installed with RedHat Enterprise Linux that needed to be erased with the new the installation. The installation was pretty straightforward. We used Rufus to apply the installation image to a USB stick. We also had an extra keyboard, mouse and display to attach to the servers during installation. Since the installation media required network connectivity we configured the interfaces to DHCP clients during the process. After deleting the old data and downloading the required packets, the installation proceeded to completion.

First thing to configure was the network and SSH connections. SLES comes with a configuration utility called YAST2 that we used for this purpose. In YAST, the interface connected to the switch needed to be configured as external. After this the network information including the static ip address, default gateway, hostname, and DNS servers needed to be configured. We used Google's DNS servers for this project (8.8.8.8 and 8.8.4.4).



```
YaST2 - lan @ umec-server1

 Network Settings
 Global Options──Overview──Hostname/DNS──Routing──
  Hostname
  umec-server1
  Set Hostname via DHCP   no          ↓

  Modify DNS Configuration Custom Policy Rule
  Use Custom Policy      ↓ STATIC           ↓
   Name Servers and Domain Search List
   Name Server 1                                           Domain Search
   8.8.8.8
   Name Server 2
   8.8.4.4
   Name Server 3
```

example of the Yast2 configuration utility.

When the SSH connections were tested to be working it was time to create user accounts. These were created one for each team: CI team, performance team, and API team.

```
useradd -m [USERNAME]
```

```
passwd [USERNAME]
```

The users needed to be added to the wheel group to give them sudo privileges

```
usermod -aG wheel [USERNAME]
```

Then it was needed to change the sudoers file to allow sudo privileges to the wheel group. This was done with the visudo command. There is already a commented line for giving sudo privileges to wheel so it was only needed to uncomment this line.

```
# %wheel ALL=(ALL) ALL
```

We also changed the sudo password request from the target password to the user password by commenting the lines

```
Defaults targetpw   # ask for the password of the target user i.e. root
```

```
ALL   ALL=(ALL) ALL   # WARNING! Only use this together with 'Defaults targetpw'
```

At first the servers worked with password authentication. This meant that any outside entity could try to brute force the passwords since we didn’t have a utility like Fail2ban installed. Fortunately no security incidents were experienced at this point, most likely thanks to the strong passwords set.

Later we were told to disable password authentication and use only public key authentication. We also disabled root logins at this time for extra security. This was done by manipulating the /etc/ssh/sshd_config file. First we uncommented and changed the lines

```
# PasswordAuthentication yes
```

```
# ChallengeResponseAuthentication yes
```

to “no”. Also as an extra security measure we disabled root logins by changing the line

```
PermitRootLogin yes
```

to “no”. Because we wanted to display a pre-login banner we changed the line

```
#Banner none
```

to

```
Banner /etc/banner
```

Then it was time to create the /etc/banner file and write a pre-login message. We also changed the post-login message by changing the /etc/motd file.

Last thing to do was to restart the ssh service.

```
systemctl restart sshd
```

We also modified user bash prompt colors and formats by editing the .bashrc files found in their home directories. Specifically this was done by adding the following line to the file:

```
export PS1="\e[1;32m[\u@\h \W]\$ \e[m"
```

This changes the prompt color to light green and shows only the working directory. It also adds brackets to the prompt. The same thing was done to the root account except with bright red color.

```
export PS1='\e[1;31m[\u@\h \W]\$ \e[m'
```

To help with system administration we wrote a few shell scripts. These are available in the scripts folder with their descriptions. These scripts were used for user account creation, deletion, key management and security.

We also requested licenses for modules such as the High Availability and Container as a Service. In the end the High availability extension was not utilized, but the CaaS module worked with Kubernetes and OpenFAAS to complete the CI/CD system.

The Raspberry Pi’s ran with the standard raspbian installation. At first they were connectable through using key-based SSH authentication, and later were set to be available only through the three servers through SSH. They were later connectable easily through a shell script by typing r1 or r2 in the terminals of the servers, standing for Raspberry 1 or 2.

Server 3 incident

On 7th October the server 3 became unresponsive from outside connection attempts. Upon further inspection the server was not reachable even from inside the network. When arriving on site, it was determined that the server had crashed. The reason for this was apparently the swapping of the SSD drive that happened a few days prior.

Unfortunately the server was unrecoverable and needed a full OS reinstall. Fortunately at this point nothing of importance was running on server 3 so the damage was minimal. After the reinstallation no further problems were reported.

Security breach incident

The fourth server was brought in hastily, and set up at evening in the server tower. The installation was a clean SLES 15. The SSH port 22 was opened once every configuration task was done. The server was set up with the hardware teams account with access to root privileges, and had a strong password. However, the person who brought the server who was in charge of installing the OpenFAAS had set a very weak user password. As the installation was done in the evening, and he was in a hurry, we had no time to set key-based authentication. Later, this became a critical issue as his user account had been given access to root privileges, as he needed to install the software needed.

Problems in the network were first discovered on November 11th, as none of the network devices were reachable. We didn’t suspect anything at the time, and went to troubleshoot the situation in the afternoon. We reset the firewall, and everything seemed to be working fine. Nevertheless, the problems continued intermittently. The firewall was taken offline. Finally on November 20th, suspecting the issues were with the fourth server as everything slowed down after its installation, we troubleshooted it as the network was having issues again. We found a process with massive resource usage that was eventually identified to be the Xor DDoS botnet. After isolating the fourth server, we removed the infection working together with the SUSE representative. After letting the server run for a day, we noticed no more abnormalities with it, but decided to wipe it and do a clean installation of SLES 15 again.

We believe the Xor DDoS botnet infection was due to the weak password set on the account made by the person who brought it in and installed OpenFAAS. Had the server been set initially with key-based authentication, or a stronger password for the other user, we do not believe it would have been infected. The issue was compounded by the user account having access to root privileges. None of the other network devices were infected, and we believe this was thanks to the adequate security measures in place on them, such as key-based authentication and minimal services open to the internet. We believe no permanent damage was incurred as a result of the incident.

Conclusion

We were tasked with setting up and maintaining the hardware, operating system and network for a microMEC project. The project spanned over the period of around two months. There were about ten students involved in the project, two representatives from Nokia and one from SUSE. We were tasked each week in a sprint meeting with a new objective.

The challenge in this project was that our team had no prior system administration experience with a linux system. On top of this neither of us had worked with real server hardware before.

Ultimately our team succeeded in creating a Continuous Integration/Continuous Delivery (CI/CD) environment with the servers attached to Raspberry Pi’s emulating the micro Mobile Edge Computing (uMEC) units, where users are able to run containers. The containers can represent applications such as visual recognition of animals or objects through cameras, air quality indicators and noise level indicators among others. The system in its entirety represents a local cluster of uMEC devices and their backend, the building block which the smart city project is built using multiple clusters of.

During the course of this project we learned a lot about linux operating system, system administration, networking, troubleshooting, shell scripting, teamwork, user account management, security and server hardware, as well as the physical installation of networking devices and servers in a device closet.

The project had multiple possible parts to improve on, such as addition of extra switches for more robust availability and faster local networking between devices, connecting the servers and other devices with the uninterrupted power supply, as well as connecting and setting up the network attached storage for extra space. Among other things, as the installation was our first time handling servers and switches, we were not familiar with how to properly do cable management in the device closet, which could be improved upon.