

# Learning Diary – Cloud Services

---

**Student:** Gleb Bulygin

**Group:** DIN24S

**Email:** [gbulygin@students.oamk.fi](mailto:gbulygin@students.oamk.fi)

## Week 1 Assignment

I have created a **cloud-services** folder for this project. My folder structure looks like this:

```
cloud-services/  
├── week01/  
│   ├── demo/  
│   │   └── ...  
│   └── img/  
├── week02/  
│   ├── app.js  
│   ├── report.pdf  
│   └── img/  
├── week03/  
│   ├── index.html  
│   ├── styles.css  
│   └── img/  
├── week-04/  
│   └── ...  
└── learning-diary.md
```

I chose the following task:

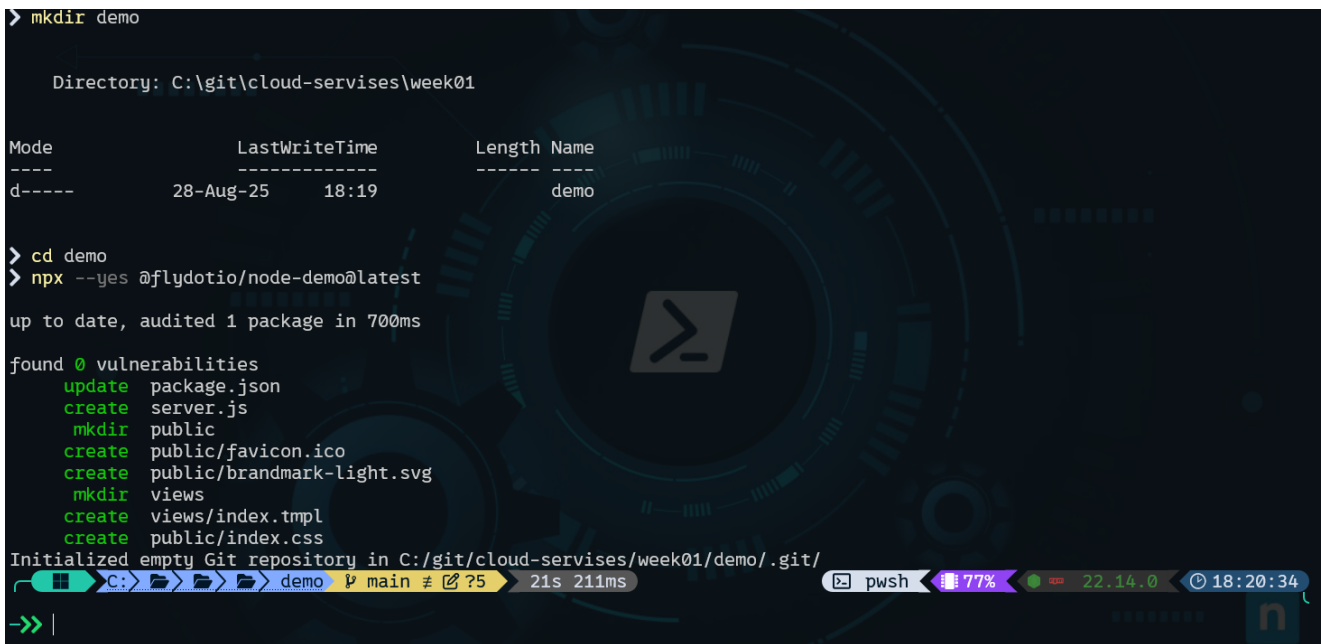
- Implement your own NodeJS app with [these](#) instructions

I created a **package.json** file with the following content:

```
{  
  "scripts": {  
    "start": "node server.js"  
  }  
}
```

Then run following commands:

```
# create a demo folder
mkdir demo
# go to the demo folder
cd demo
# Run the latest Fly.io Node.js demo app with npx (auto-confirm install)
npx --yes @flydotio/node-demo@latest
```



The screenshot shows a Windows PowerShell terminal window. The first command is `mkdir demo`, which creates a new directory named 'demo' in the current path `C:\git\cloud-services\week01`. The second command is `cd demo`, which changes the current directory to `C:\git\cloud-services\week01\demo`. The third command is `npx --yes @flydotio/node-demo@latest`, which installs the latest version of the Node.js demo app. The output shows that the app is up to date, audited 1 package in 700ms, and found 0 vulnerabilities. It then lists the files to be created: `package.json`, `server.js`, `public`, `public/favicon.ico`, `public/brandmark-light.svg`, `views`, `views/index.tmpl`, and `public/index.css`. Finally, it initializes an empty Git repository in `C:\git\cloud-services\week01\demo\.git/`. The terminal status bar at the bottom shows the current directory as `C:\git\cloud-services\week01\demo`, the branch as `main`, and the commit hash as `75`. The battery level is at 77%, and the time is 18:20:34.

```
> mkdir demo

Directory: C:\git\cloud-services\week01

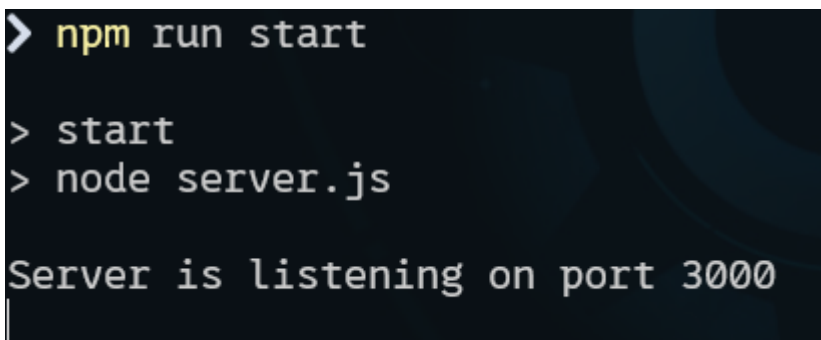
Mode                LastWriteTime         Length Name
----                -
d-----            28-Aug-25         18:19         demo

> cd demo
> npx --yes @flydotio/node-demo@latest

up to date, audited 1 package in 700ms
found 0 vulnerabilities
  update package.json
  create  server.js
  create  public
  create  public/favicon.ico
  create  public/brandmark-light.svg
  create  views
  create  views/index.tmpl
  create  public/index.css
Initialized empty Git repository in C:/git/cloud-services/week01/demo/.git/
C:\git\cloud-services\week01\demo > 21s 211ms 77% 22.14.0 18:20:34
```

**Figure 1.1:** Result of the commands above

Then the app can be started with the `npm run start` command from the terminal.



The screenshot shows a Windows PowerShell terminal window. The first command is `npm run start`, which starts the Node.js demo app. The output shows the command `start` being executed, which then runs `node server.js`. The final output is `Server is listening on port 3000`.

```
> npm run start

> start
> node server.js

Server is listening on port 3000
```

**Figure 1.2:** App started

The app is running locally on port 3000. To confirm, go to <http://localhost:3000/>.



**Figure 1.3:** App increments the counter after every page visit or refresh

---

**At this point, I realized that Fly.io would charge me for hosting the app. I did not want to deal with the billing system, so I switched to a different assignment:**

Or set up a Linux VPS with some cloud VPS provider such as CSC, DigitalOcean, Hetzner, Oracle, Upcloud, AWS, Azure, etc.

During the class, we went through the steps of setting up a virtual machine on [csc.fi](https://csc.fi). I will document the main steps here:

1. Set up a myCSC account. We used authentication with Haka:

## Select an authentication provider

⚠ Starting from 11.6.2025 using MyCSC portal will require multi-factor authentication (MFA). [Instructions for setting up MFA.](#) [↗](#)

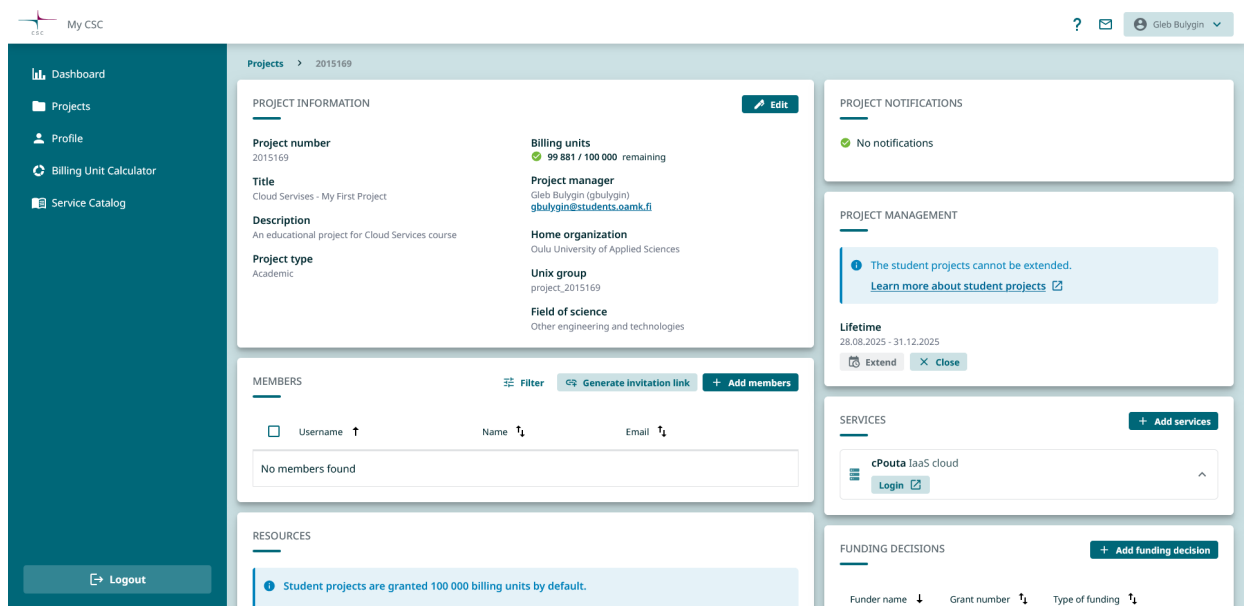
[Which authentication provider should I use?](#) [↗](#)



[Enable CSC Multi-Factor Authentication](#)

**Figure 1.4:** Login page

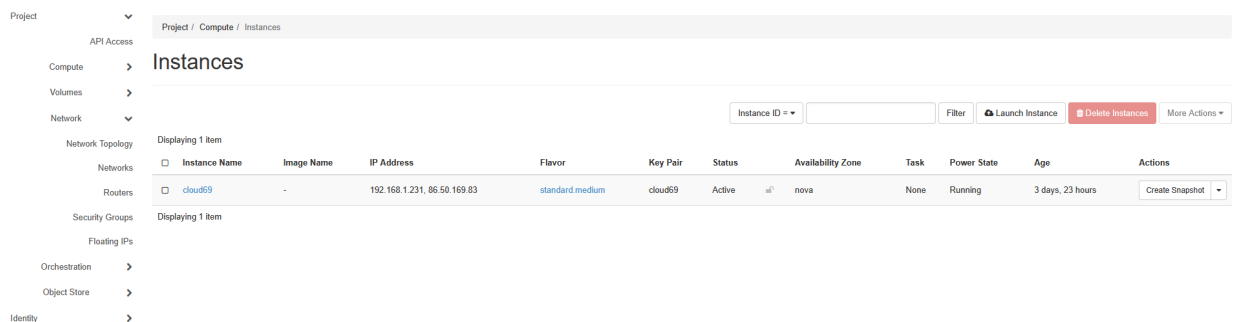
2. Create a new project.
3. Add a cPouta service to the project.



**Figure 1.5:** *myCSC project*

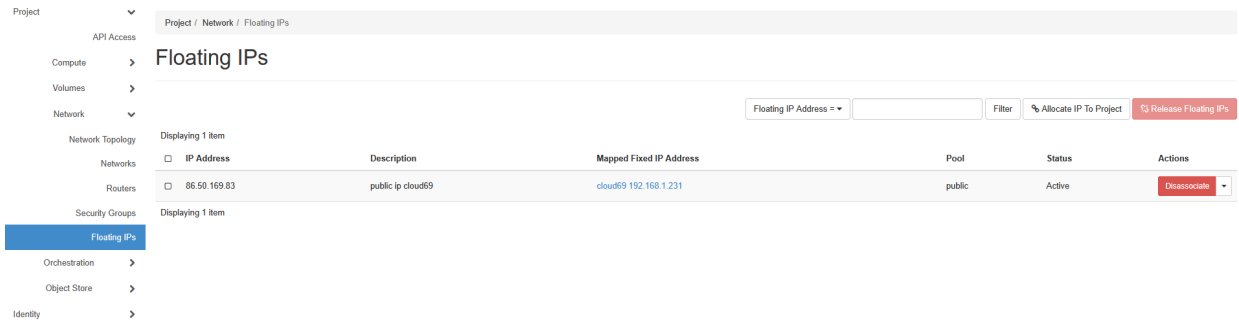
4. Log in to [cPouta service](#).
5. Create a new instance. I chose Ubuntu 24.04, [standard.medium](#), and created a new SSH key pair for it.

**Important!** An SSH key can only be added to the instance at the moment of creation. I tried to do it afterward and failed.



**Figure 1.6:** *myCSC project after all settings are applied*

6. On the **Network/Floating IPs** tab, create a floating IP and assign it to the instance. This IP address will be used to access the instance.

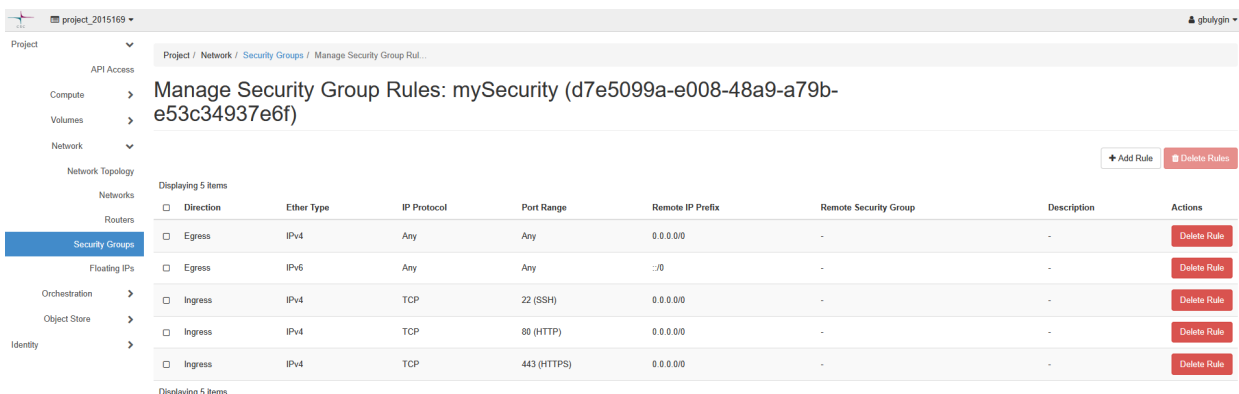


**Figure 1.7: Floating IP**

All changes to the instance are applied from the **Actions** button on the right-hand side of the table. For example, to associate a Floating IP, click the dropdown list and select **Associate Floating IP**.

7. On the **Network/Security groups** tab, create a new security group and add the following rules:

- SSH (default parameters, I left the IP range set to 0.0.0.0/0 – no restrictions for now).
- Custom TCP rule: port 80 (HTTP).
- Custom TCP rule: port 443 (HTTPS).



**Figure 1.8: Security group settings**

8. Move the private part of the generated earlier SSH key to the **~/.ssh** folder.

Instructions suggested using PuTTY to connect to the virtual machine, but I preferred PowerShell. (I tried PuTTY as well, and it worked.)

9. Under **~/.ssh**, create a file named **config**.

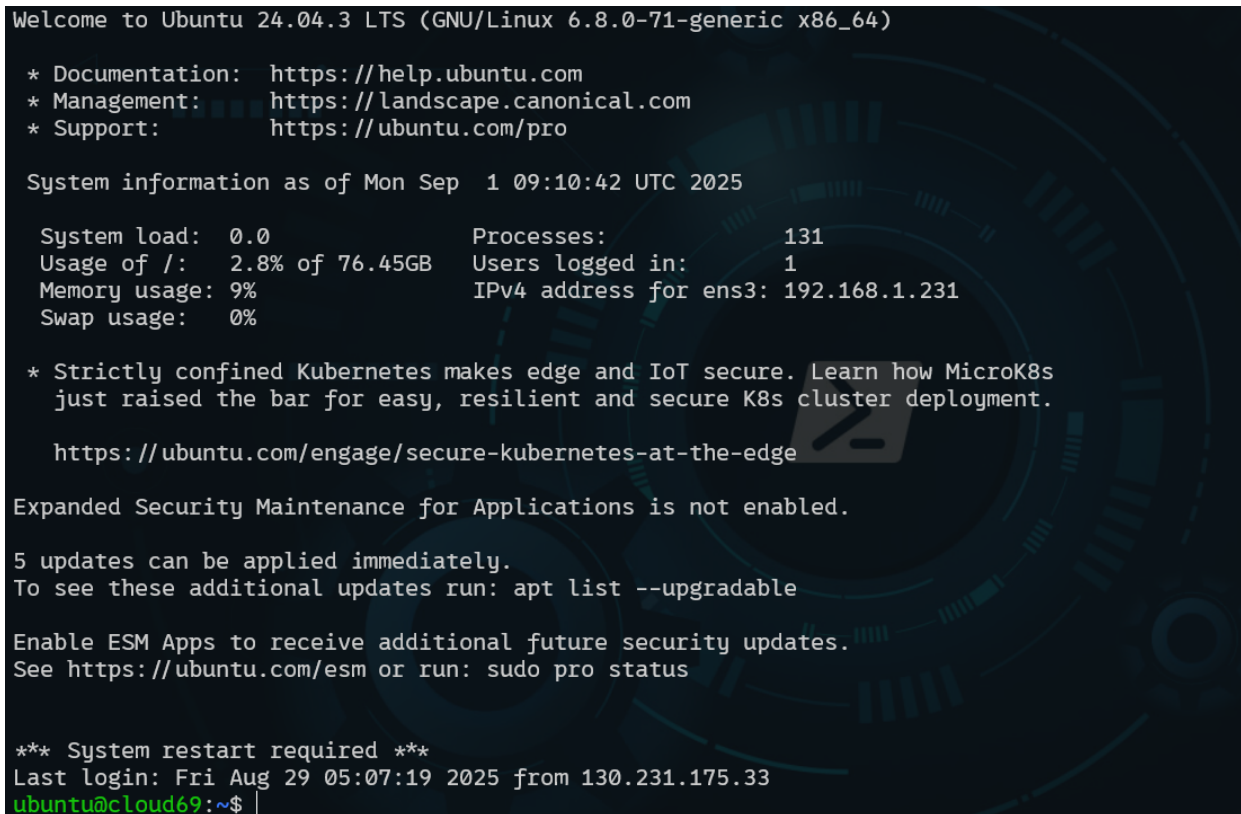
10. Paste the following content into it:

```
Host cloud69
  HostName 86.50.169.83
  User ubuntu
  IdentityFile ~/.ssh/cloud69
```

- **Host** – local name for the SSH connection
- **HostName** – floating IP of the virtual machine
- **User** – default user (**ubuntu**)
- **IdentityFile** – private part of the SSH key pair generated during VM creation

11. Connect to the virtual machine:

```
ssh cloud69
```



```

Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.8.0-71-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Mon Sep  1 09:10:42 UTC 2025

System load:  0.0               Processes:            131
Usage of /:   2.8% of 76.45GB   Users logged in:     1
Memory usage: 9%               IPv4 address for ens3: 192.168.1.231
Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

5 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

*** System restart required ***
Last login: Fri Aug 29 05:07:19 2025 from 130.231.175.33
ubuntu@cloud69:~$

```

**Figure 1.9:** Successful connection to the virtual machine

12. Run the following commands to install and start the Apache2 service:

```

sudo apt update
sudo apt install apache2

sudo systemctl start apache2
sudo systemctl enable apache2
sudo systemctl status apache2

```

```

ubuntu@cloud69:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled; preset: enabled)
   Active: active (running) since Thu 2025-08-28 10:06:22 UTC; 3 days ago
     Docs: https://httpd.apache.org/docs/2.4/
   Process: 39806 ExecReload=/usr/sbin/apachectl graceful (code=exited, status=0/SUCCESS)
  Main PID: 2501 (apache2)
    Tasks: 55 (limit: 4540)
   Memory: 8.4M (peak: 14.2M)
      CPU: 21.689s
   CGroup: /system.slice/apache2.service
           └─ 2501 /usr/sbin/apache2 -k start
              39812 /usr/sbin/apache2 -k start
              39813 /usr/sbin/apache2 -k start

Aug 28 10:06:22 cloud69 systemd[1]: Started apache2.service - The Apache HTTP Server.
Aug 30 00:00:01 cloud69 systemd[1]: Reloading apache2.service - The Apache HTTP Server...
Aug 30 00:00:01 cloud69 apachectl[16136]: AH00558: apache2: Could not reliably determine the server's fully qualified d
Aug 30 00:00:01 cloud69 systemd[1]: Reloaded apache2.service - The Apache HTTP Server.
Aug 31 00:00:01 cloud69 systemd[1]: Reloading apache2.service - The Apache HTTP Server...
Aug 31 00:00:01 cloud69 apachectl[32244]: AH00558: apache2: Could not reliably determine the server's fully qualified d
Aug 31 00:00:01 cloud69 systemd[1]: Reloaded apache2.service - The Apache HTTP Server.
Sep 01 00:00:05 cloud69 systemd[1]: Reloading apache2.service - The Apache HTTP Server...
Sep 01 00:00:05 cloud69 apachectl[39809]: AH00558: apache2: Could not reliably determine the server's fully qualified d
Sep 01 00:00:05 cloud69 systemd[1]: Reloaded apache2.service - The Apache HTTP Server.
lines 1-24/24 (END)

```

Figure 1.10: Apache2 status

13. Open the page in a browser: <http://86.50.169.83/>



Figure 1.11: Apache2 default page showing in the browser



14. With a little help of chat GPT I hve created a dashboard that shows some system Information of my Virtual machine.

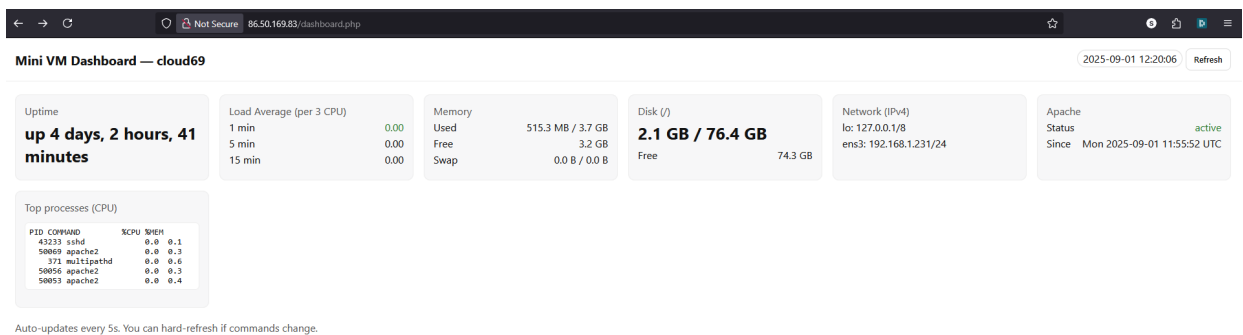
a. Run following commands on virtual machine terminal

```
sudo apt update
sudo apt install -y php libapache2-mod-php
sudo systemctl restart apache2
```

b. Create the dashboard

Create and file `/var/www/html/dashboard.php`. The code is too long to include it here. It is posted on my [private repo](#) for this course.

c. Visit <http://86.50.169.83/dashboard.php> to check it out.



**Figure 1.12:** *Dashboard panel*