UNIX - Section 1, Semester 3
Team Composition: Helene Rousseau, Rachel Herron, Sabrina Robinson

## Project Description/Goals

We would like to create a website and host it online with a GNU/Linux web server that has auto-deployment. To do so, we will set up a VPS and set up a domain name. This will allow us to handle basic system setup and security such as giving permissions to the various user types that will use the website. Users include visitors, normal users and admins. For normal users and admins will be required to create an account and authenticate it. The website will display user portfolios with static pages.

## Platform of Choice

The platform that we need to use for this project is a web server. We're considering DigitalOcean as our VPS of choice. The reason for this being that DigitalOcean is beginner friendly and has affordable pricing. Additionally, it provides plenty of documentation and tutorials to help guide us through the process of setting up the server for the first time. Lastly, if we were to continue expanding our project in the future, Digital Ocean's droplets provide the necessary features to facilitate this process such as a dedicated server, block storage, spaces, databases, etc.

## Demonstration Plan

We will connect one of our laptops via a HDMI wire to show our screen as we demonstrate the process in which a user may use the website. We will show the differences between a visitor, a user, and an admin. As well as how a user may upload their own static page to the website and how the server will handle this upload. The page should auto-deploy the new pages.

## Requirements

**Basic system setup and security:**
Accounts Creation:
Create a sudo user for deployment purposes that is not root. This user will be able to pull from the Git repository and manage web files.

Authentication Method:
Generate SSH key authentication for the user on the local machine, then copy the public key to the server.

File Permissions:
Only the user we created should be able to update the web directory and the server should be able to read the web directory.

Firewall Configuration:
Only allow ports that we will be using, for example port 22 for SSH access and then port 80 for the website.

**Process or service management/scheduling:**
Systemd Service for a Webhook Listener:
Run the script as a service

Cron Job for Scheduled Tasks:
Run periodic checks or backups

**Automated tasks using a script language:**
Automated backup:
Regularly back up the files on the web server or the entire web directory.

Webhook Listener Script:
Every time a push is made to the Git repository the webhook listener triggers the git pull command and optionally reloads Nginx to show the latest changes

## Major Technical Solutions Compared

| Platform | VPS Setup | Control | Web Server | Security config | HTTPS | Auto-Deployment | Database | Object storage | Auto-Scaling | Pricing |
|---|---|---|---|---|---|---|---|---|---|---|
| Digital Ocean | Yes | Root Access | Fully Configurable | Manual Setup | Manual | With Github actions | Yes with PostgreSQL | Yes with Spaces | Manual with load balancers | Starting at 5$/Month |
| AWS Ec2 | Yes | Root Access | Fully Configurable | Security Groups, IAM | Manual or Via ACM | With Github actions or other CI/CD | Yes with RDS(MySQL, Postgres) | Yes with S3 | Native auto-scaling available | Starting at 3$ / Month |
| Google Cloud | Yes | Root Access | Fully Configurable | With Firewall, IAM | Via Certbot or managed with SSL | With Github actions or other CI/CD | Yes with Cloud SQL | Yes with Cloud Storage | Native auto-scaling available | Pay-as-you-go (usually $$$) |
| Heroku | No | Limited | Built-in | Limited | Automatic SSL | Built-in | Add-ons | No native | Limited scaling with Dynos | Free |
| Netlify | No | None | Built-in static hosting | Limited | Automatic | Built-in | No database support | No native | Auto scaling via CDN | Free |

Digital Ocean is the best option for our project as we would like to have complete control over the infrastructure while ensuring that the costs for the project are tangible.

## Timeline

Week 1 : Create website to implement

- Create the HTML structure
- Transfer to a react website
- Test the website

Week 2 : Research for implementation

- Have the VPS working
- Search & choose a domain name
- Attach the domain to the website
- Prepare for the implementation

Week 3 : Implementation & presentation

- Implement the website
- Test if everything is working properly
- Prepare presentation
- Present

## Team Composition

- Rachel Herron
- Helene Rousseau
- Sabrina Robinson