SSW590 Group 10

Version: 088dcf5

by

Jack Galligan, James Grant, Collin Smith

Stevens.edu

November 12, 2025

SSW590 Group 10
Version: 088dcf5

Jack Galligan, James Grant, Collin Smith
Stevens.edu

0.1 Document Update History

This document provides the requirements and design details of the PROJECT. The following table (Table 1) should be updated by authors whenever major changes are made to the architecture design or new components are added. Add updates to the top of the table. Most recent changes to the document should be seen first and the oldest last.

Table 1: Document Update History

Date	Updates
11/12/2025	JGr, JGa, CS: • Added Load Balancing (Chapter 12) to documentation
11/5/2025	JGr, JGa, CS: • Added Jenkins Chapter (Chapter 11) to documentation
10/29/2025	JGr, JGa, CS: • Added Grafana Chapter (Chapter 10) to documentation
10/22/2025	JGr, JGa, CS: • Added GitHub Actions (Chapter 9) to automatically update and compile LaTeX to the newest version.
10/15/2025	JGr, JGa, CS: • Added DNsSSLVersioning (Chapter 8) and synced Overleaf to Github. • Updated Bibliography and index
10/08/2025	JGr, JGa, CS: • Added Bugzilla (Chapter 6) • Added Overleaf (Chapter 7) • Added Hosts and Passwords (Table 2)
10/01/2025	JGr, JGa, CS: • Added AWS Deployment (Chapter 4) • Added LaTeX Docker (Chapter 5)

Table 1: Document Update History

Date	Updates
09/24/2025	JGr, JGa, CS: • Added Project Proposal (Chapter 2)
09/17/2025	JGr, JGa, CS: • Edited Linux Commands Chapter
09/15/2025	JGr, JGa, CS: • Created manual • Added Linux Commands

0.2 Hosts and Passwords

The following table (Table 2) includes the hosts and password hints to accounts that are used in the duration of the project.

Table 2: Hosts and Passwords

Hosts	Passwords
root	Hints: • 73 • Progress
jgallig1@stevens.edu	Hints: • Name of a close friend • Number of a parasite
jgrant4@stevens.edu	Hints: • Name of class • What you do in class

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

– Jack Galligan, James Grant, Collin Smith

I'm Jack Galligan, a software engineering major who has spent 8 years pursuing software development. My preference is Python although I've done a little bit of a lot of the major languages, and I have a dog named Luna.

My name is Collin Smith, and I'm a 4/4 computer science major. I am most experienced in Java and Python, but I have learned and used about a dozen different programming and scripting languages. I am the youngest of three brothers in my family, and we have a dog named Zoey.

My name is James Grant and I am a senior Software Engineering major. I've been programming for about 4 years now and my favorite coding languages are Python and Javascript. I've been a member of several organizations on campus including SUMAC and the Software Engineering Club, as well as a member of Chi Phi Fraternity for 3 years. In my free time, I like to watch movies, go to the gym, and play video games.

Chapter 2

Project Proposal

– JGa, JGr, CS

2.1 Project Description

Our project is our senior design project, GreekConnect, which is intended to be a web app and accompanying app which allows fraternities, sororities, and their members to schedule and view events. Our users include Stevens faculty as well as a significant student body, so for our app to be useful it needs to be easy to install and open on a computer as well as a phone by students who likely don't want to do very much in order to get access to the app. Some features which we will need to test include:

- Does the app work when multiple people try to make changes at once?
- Do transfers of roles and permissions work as intended?
- Do students and student leaders see messages intended for them?

We will also need a database, as our app plans on storing the information of many people's schedules. Considering the Stevens database only appears to be a convenient option on campus, we expect to need to look into alternatives like an Amazon Web Server. We will try out a few different options to determine the minimum specifications necessary for our needs.

2.2 Dev Ops Tool Specification

Source control management will be done using Git and Github, which will allow us to focus on testing and deployment collaboratively. Deployment will likely use Docker to containerize the app for various environments and Kubernetes to orchestrate and maintain the containers. Our project will likely use a synchronized database such as Firebase, which is a document-based NoSQL database, so that it's easier to program user data editing/collaboration. We will also explore SQL based options such as Supabase, PostgreSQL and Neon depending on our planned coding schedule. Testing will likely use Jest or other Javascript testing frameworks (React Testing Library, Cypress) to test the front and back end of the application.

Chapter 3

Linux Commands

– JG, CS

3.1 Terminal Bash Commands

collin@Collin:/mnt/c/WINDOWS/system32$
mkdir -p ~/lx-test && cd ~/lx-test

collin@Collin:~/lx-test$
printf ”alpha\nbeta\nGamma\ngamma\nbeta\n” > words.txt

collin@Collin:~/lx-test$
printf ”id,name,dept\n1,Ada,EE\n2,Linus,CS\n3,Grace,EE\n4,Dennis,CS\n” > people.csv

collin@Collin:~/lx-test$
printf ”INFO boot ok\nWARN disk low\nERROR fan fail\nINFO shutdown\n” > sys.log

collin@Collin:~/lx-test$ dd if=/dev/zero of=blob.bin bs=1K count=48 status=none

collin@Collin:~/lx-test$ mkdir -p src/lib tmp archive

collin@Collin:~/lx-test$ printf ”one two three four\n” > src/file1.txt

collin@Collin:~/lx-test$ printf ”two three four five\n” > src/file2.txt

collin@Collin:~/lx-test$ ln -s src/file1.txt link-to-file1

collin@Collin:~/lx-test$ touch -t 202401020304 old.txt

3.2 Linux Problem Set Answers

A) Navigation & File Ops

1. Show your present working directory path only.

```
collin@Collin:~/lx-test$ pwd
/home/collin/lx-test
```

2. List all entries in the current directory, one per line, including dotfiles.

```
collin@Collin:~/lx-test$ ls -1a
.
..
archive
blob.bin
link-to-file1
old.txt
people.csv
src
sys.log
tmp
words.txt
```

3. Copy src/file1.txt to tmp/ only if tmp exists; do it verbosely.

```
collin@Collin:~/lx-test$ cp -v src/file1.txt tmp/
'src/file1.txt' -> 'tmp/file1.txt'
```

4. Move old.txt into archive/ and keep its original timestamp.

```
collin@Collin:~/lx-test$ mv -v old.txt archive/
renamed 'old.txt' -> 'archive/old.txt'
collin@Collin:~/lx-test$ ls -l archive/
total 0
-rw-r--r-- 1 collin collin 0 Jan  2  2024 old.txt
```

5. Create a new empty file notes.md only if it doesn't already exist.

```
collin@Collin:~/lx-test$ ls notes.md
ls: cannot access 'notes.md': No such file or directory
collin@Collin:~/lx-test$ [ -e notes.md ] || touch notes.md
collin@Collin:~/lx-test$ ls notes.md
notes.md
```

6. Show disk usage (human-readable) for the src directory only (not total FS).

```
collin@Collin:~/lx-test$ du -h src
4.0K    src/lib
16K     src
```

B) Viewing & Searching

7. Print line numbers while displaying sys/log.

```
collin@Collin:~/lx-test$ cat -n sys.log
 1  INFO boot ok
 2  WARN disk low
 3  ERROR fan fail
 4  INFO shutdown
```

8. Show only the lines in sys.log that contain ERROR (case-sensitive)

```
collin@Collin:~/lx-test$ grep ERROR sys.log
ERROR fan fail
```

9. Count how many distinct words appear in words.txt (case-insensitive).

```
collin@Collin:~/lx-test$
tr '[:upper:]' '[:lower:]' < words.txt | sort | uniq | wc -l
3
```

10. From words.txt, show lines that start with g or G.

```
collin@Collin:~/lx-test$ grep '^[gG]' words.txt
Gamma
gamma
```

11. Display the first 2 lines of people.csv without using an editor.

```
collin@Collin:~/lx-test$ head -n 2 people.csv
id,name,dept
1,Ada,EE
```

12. Show the last 3 lines of sys.log and keep following if the file grows.

```
collin@Collin:~/lx-test$ tail -n 3 -f sys.log
WARN disk low
ERROR fan fail
INFO shutdown


^C
```

C) Text Processing

13. From people.csv, print only the name column (2nd), excluding the header.

```
collin@Collin:~/lx-test$ cut -d',' -f2 people.csv | tail -n +2
Ada
Linus
Grace
Dennis
```

14. Sort words.txt case-insensitively and remove duplicates.

```
collin@Collin:~/lx-test$ tr '[:upper:]' '[:lower:]' < words.txt | sort | uniq
alpha
beta
gamma
```

15. Replace every three with 3 in all files under src/ in-place, creating .bak backups.

```
collin@Collin:~/lx-test$ sed -i.bak 's/three/3/g' src/file1.txt src/file2.txt
collin@Collin:~/lx-test$ cat src/file1.txt
one two 3 four
collin@Collin:~/lx-test$ cat src/file2.txt
two 3 four five
collin@Collin:~/lx-test$ ls src/*.bak
src/file1.txt.bak  src/file2.txt.bak
```


16. Print the number of lines, words, and bytes for every *.txt file in src/.

```
collin@Collin:~/lx-test$ wc src/*.txt
 1  4 15 src/file1.txt
 1  4 16 src/file2.txt
 2  8 31 total
```

D) Permissions & Ownership

17. Make tmp/ readable, writable, and searchable only by the owner.

```
collin@Collin:~/lx-test$ chmod 700 tmp/
collin@Collin:~/lx-test$ ls -ld tmp/
drwx------ 2 collin collin 4096 Sep 15 16:09 tmp/
```

18. Give group execute permission to src/lib recursively without touching others/owner bits.

```
collin@Collin:~/lx-test$ chmod -R g+x src/lib
collin@Collin:~/lx-test$ ls -l src/lib
total 0
collin@Collin:~/lx-test$ ls src/lib
collin@Collin:~/lx-test$ ls -ld src/lib
drwxr-xr-x 2 collin collin 4096 Sep 15 15:28 src/lib
```

19. Show the numeric (octal) permissions of src/file2.txt.

```
collin@Collin:~/lx-test$ stat -c "%a" src/file2.txt
644
```

20. Make notes.md append-only for the owner via file attributes (if supported).

```
collin@Collin:~/lx-test$ sudo chattr +a notes.md
[sudo] password for collin:
collin@Collin:~/lx-test$ lsattr notes.md
-----a--------e------- notes.md
collin@Collin:~/lx-test$ echo "text" >> notes.md
collin@Collin:~/lx-test$ echo "overwrite" > notes.md
-bash: notes.md: Operation not permitted
```


E) Links & Find

21. Verify whether link-to-file1 is a symlink and show its target path.

```
collin@Collin:~/lx-test$ readlink link-to-file1
src/file1.txt
```

22. Find all regular files under the current tree larger than 40 KiB.

```
find . -type f -size +40k
./blob.bin
```

23. Find files modified in the last 10 minutes under tmp/ and print their sizes.

```
collin@Collin:~/lx-test$ find tmp/ -type f -mmin -10 -ls
```

F) Processes & Job Control

24. Show your processes in a tree view.

```
  jack@Jacktop:~$ ps -u $USER --forest
 PID TTY          TIME CMD
 908 pts/4    00:00:00 bash
 923 pts/4    00:00:00  \_ ps
 465 pts/1    00:00:00 bash
 367 pts/2    00:00:00 sh
 411 ?        00:00:00 systemd
 412 ?        00:00:00  \_ (sd-pam)
```

25. Start sleep 120 in the background and show its PID.

```
    jack@Jacktop:~$ sleep 120 & echo $1
  [1] 1047
```

26. Send a TERM signal to all sleep processes owned by you (don't use kill -9).

```
    jack@Jacktop:~$ pkill -TERM -u $USER sleep
  [1]+  Terminated              sleep 120
```

27. Show the top 5 processes by memory usage (one-shot, not interactive).

```
jack@Jacktop:~$ ps -eo pid,user,%mem,rss,cmd --sort=-%mem | head -n 6
  PID USER     %MEM   RSS CMD
  709 root      0.1 22952 /mnt/wsl/docker-desktop/docker-desktop-user-distro proxy --distro-name Ubu
  276 root      0.1 22400 /usr/bin/python3 /usr/share/unattended-upgrades/unattended-upgrade-shutc
   42 root      0.0 15200 /usr/lib/systemd/systemd-journald
    1 root      0.0 12316 /sbin/init
  207 root      0.0 11840 /usr/libexec/wsl-pro-service -vv
```

G) Archiving & Compression

28. Create a gzipped tar archive src.tgz from src/ with relative paths.

```
jack@Jacktop:~/lx-test$ tar -czf src.tgz -C src .
```

29. List the contents of src.tgz without extracting.

```
jack@Jacktop:~/lx-test$ tar -tzf src.tgz
./
./file2.txt
./file1.txt
./lib/
```

30. Extract only file2.txt from src.tgz into tmp/.

```
jack@Jacktop:~/lx-test$ tar -xzf src.tgz -C tmp ./file2.txt
```

H) Networking & System Info

31. Show all listening TCP sockets with associated PIDs (no root assumptions).

```
jack@Jacktop:~/lx-test$ ss -ltp
State    Recv-Q   Send-Q     Local Address:Port        Peer Address:Port   Process
LISTEN   0        4096       127.0.0.53%lo:domain           0.0.0.0:*
LISTEN   0        4096          127.0.0.54:domain           0.0.0.0:*
LISTEN   0        1000      10.255.255.254:domain           0.0.0.0:*
```

32. Print your default route (gateway) in a concise form.

```
jack@Jacktop:~/lx-test$ ip route show default
default via 172.20.240.1 dev eth0 proto kernel
```

33. Display kernel name, release, and machine architecture.

```
jack@Jacktop:~/lx-test$ uname -srm
Linux 6.6.87.2-microsoft-standard-WSL2 x86_64
```

34. Show the last 5 successful logins (or last sessions) on the system.

```
jack@Jacktop:~/lx-test$ last -n 5
reboot   system boot  6.6.87.2-microso Wed Sep 17 20:09   still running
reboot   system boot  6.6.87.2-microso Mon Sep 15 15:37   still running
reboot   system boot  6.6.87.2-microso Mon Sep 15 15:09   still running

wtmp begins Mon Sep 15 15:09:55 2025
```

I) Package & Services (Debian/Ubuntu)

35. Show the installed version of package coreutils.

```
jack@Jacktop:~/lx-test$ dpkg -l coreutils | awk '/coreutils/ {print $3}'
9.4-3ubuntu6
```

36. Search available packages whose names contain ripgrep.

```
jack@Jacktop:~/lx-test$ apt-cache search ripgrep
elpa-consult - Useful commands based on completing-read for Emacs
elpa-dumb-jump - jump to definition for multiple languages without configuration
ripgrep - Recursively searches directories for a regex pattern
ugrep - faster grep with an interactive query UI
```

37. Check whether service cron is active and print its status line only.

```
jack@Jacktop:~/lx-test$ systemctl status cron | grep 'Active:'
Active: active (running) since Wed 2025-09-17 20:09:05 EDT; 51min ago
```

J) Bash & Scripting

38. Write a one-liner that loops over *.txt in src/ and prints: <filename>: <linecount>.

jack@Jacktop:~/lx-test$ for f in src/*.txt; do echo ”$f: $(wc -l < ”$f”)”; done
src/file1.txt: 1
src/file2.txt: 1

39. Write a command that exports CSV rows where dept == ”CS” to cs.txt (exclude header).

    jack@Jacktop:~/lx-test$ awk -F, '$2==”CS” {print}' people.csv > cs.txt

40. Create a variable X with value 42, print it, then remove it from the environment.

    jack@Jacktop:~/lx-test$ export X=42; echo $X; unset X
    42

Chapter 4

AWS Deployment

– JGa, JGr, CS

4.1 Steps taken to deploy website

The link to the website can be found here: https://mgn8mc4khh.us-east-2.awsapprunner.com/

After both installing AWS and creating accounts, here are the steps we took to deploy our website using AWS:

1. Configured environment variables:

```
export AWS_ACCOUNT_ID=<My Account ID>
export AWS_REGION=us-east-2
export ECR_REPO=myapp
export IMAGE_TAG=v1
export APP_NAME=my-apprunner-app
export CONTAINER_PORT=3000
```

2. Authenticated Docker with Amazon ECR:

```
aws ecr get-login-password --region $AWS_REGION --profile default \
| docker login --username AWS --password-stdin $AWS_ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com
```

3. Created an ECR repository:

```
aws ecr create-repository \
  --repository-name $ECR_REPO \
  --region $AWS_REGION \
  --profile default
```

4. Built our Docker image locally:

```
docker build -t $ECR_REPO:$IMAGE_TAG .
```

5. Tagged the image for our private ECR repository:

```
 docker tag $ECR_REPO:$IMAGE_TAG \
$AWS_ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com/$ECR_REPO:$IMAGE_TAG

```

6. Pushed the image to ECR:

```
 docker push $AWS_ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com/$ECR_REPO:$IMAGE_TAG

```

7. Deployed the image to App Runner:

```
 aws apprunner create-service \
   --service-name "$APP_NAME" \
   --region "$AWS_REGION" --profile default \
   --source-configuration "{
     \"ImageRepository\": {
       \"ImageIdentifier\": \"$AWS_ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com/$ECR_REPO:$IMAGE_TAG\",
       \"ImageRepositoryType\": \"ECR\",
       \"ImageConfiguration\": {\"Port\": \"$CONTAINER_PORT\"}
     },
     \"AuthenticationConfiguration\": {
       \"AccessRoleArn\": \"arn:aws:iam::$AWS_ACCOUNT_ID:role/AppRunnerECRAccessRole\"
     },
     \"AutoDeploymentsEnabled\": true
   }" \
   --instance-configuration "{\"Cpu\":\"1 vCPU\",\"Memory\":\"2 GB\"}"

```

8. Viewed the service status until it said RUNNING:

```
 aws apprunner describe-service \
   --service-arn arn:aws:apprunner:us-east-2:039612868337:service/my-apprunner-app/645d0eab8242460da212316afafce4ec \
   --region $AWS_REGION --profile default \
   --query 'Service.Status'

```

9. Once the status was RUNNING, we were able to access our live website using the ServiceUrl provided, e.g.:

   https://mgn8mc4khh.us-east-2.awsapprunner.com

4.2 Class Based Website

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Color Buttons App</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      text-align: center;
      margin-top: 50px;
      transition: background-color 0.3s ease;
    }
    button {
      padding: 12px 24px;
      font-size: 18px;
      margin: 10px;
      cursor: pointer;
    }
  </style>
</head>
<body>
  <h1>Click a Button to Change Background</h1>
  <button id="blueBtn">Blue</button>
  <button id="redBtn">Red</button>

  <script>
    // Define a class to handle color changes
    class ColorChanger {
      constructor() {
        this.body = document.body;
        this.blueBtn = document.getElementById("blueBtn");
        this.redBtn = document.getElementById("redBtn");
      }

      init() {
        this.blueBtn.addEventListener("click", () => this.changeColor("blue"))
    ;
        this.redBtn.addEventListener("click", () => this.changeColor("red"));
      }

      changeColor(color) {
        this.body.style.backgroundColor = color;
      }
    }

    // Create and initialize the object
    const colorChanger = new ColorChanger();
    colorChanger.init();
  </script>
</body>
</html>
```


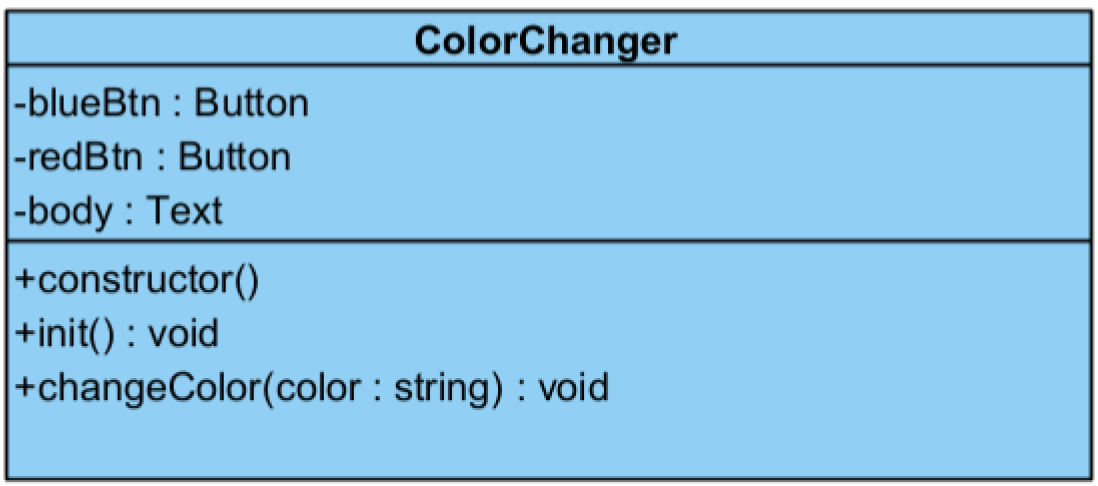

Figure 4.1: UML diagram of class based Color Changer

Chapter 5

LaTeX Docker

– JGa, JGr, CS

In this chapter, we use a Docker container to compile a simple LaTeX document using TeX Live.

To download TeX Live and create the Dockerfile, we use the following command in our Linux terminal:

```
cat > Dockerfile << 'EOF'
FROM ubuntu:22.04

RUN apt-get update && apt-get install -y texlive \
  && apt-get clean && rm -rf /var/lib/apt/lists/*

WORKDIR /data

COPY sample.tex .

CMD ["pdflatex", "sample.tex"]
EOF
```

We build and run the container as follows:

```
docker build -t latex-docker .
docker run --rm -v $(pwd):/data latex-docker
```

We can finally view the sample.pdf file created from a sample LaTeX document:

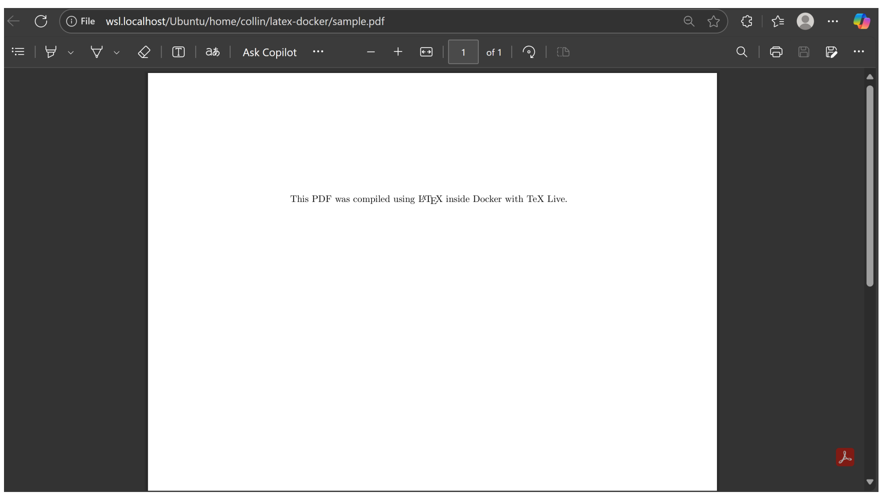

Figure 5.1: Sample LaTeX PDF generated through Docker and TeX Live.

Chapter 6

Bugzilla

– JGa, JGr, CS

In this chapter, we discuss the steps taken for the configuration of the Bugzilla Docker container on Digital Ocean.

6.1 Preface

Before anything else, make sure you have a Digital Ocean Droplet running.

Then, enter the following into the terminal:

```
ssh root@164.90.141.14
```

6.2 Installing Docker

Run the following lines in the terminal to install Docker:

```
# update
apt update && apt upgrade -y

# install docker & compose plugin
apt install -y ca-certificates curl gnupg lsb-release
mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | gpg --dearmor -o /etc/apt/keyrings/docker.gpg
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | tee /etc/apt/sources.list.d/docker.list > /dev/null
apt update && apt install -y docker-ce docker-ce-cli containerd.io docker-compose-plugin

# enable docker
systemctl enable --now docker
```

6.3 Clone Bugzilla's Repo

The following commands can be run to clone the Bugzilla repository:

```
apt install -y git
cd /opt
git clone https://github.com/bugzilla/bugzilla.git
cd bugzilla
# the repo contains a docker-compose.yml (demo)
docker compose up -d
# watch logs until it finishes
docker compose logs -f
```

6.4 Public IP:Port

The Bugzilla site can be accessed at: http://164.90.141.14:8080/

Chapter 7

Overleaf

– JGa, JGr, CS

In this chapter, we discuss the details on the configuration of the Overleaf Docker container on Digital Ocean.

7.0.1 Creating the Digital Ocean Droplet

First, we created a new droplet through the Digital Ocean dashboard by selecting Ubuntu 22.04 LTS as the operating system. We chose a plan wth 4GB RAM and 2 vCPUs, selected the New York datacenter region, and configured SSH key authentication. After creating the droplet, we connected via SSH using the assigned IP address.

7.0.2 Installing Docker and Docker Compose

We first updated the system packages and installed the required dependencies:

```
apt update && apt upgrade -y
apt install -y apt-transport-https ca-certificates curl \
    software-properties-common
```

Then it was necessary to add Docker's official GPG key and repository, ensuring to specify the correct architecture (amd64) for the Intel/AMD droplet:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | \
    gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg ] \
    https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | \
    tee /etc/apt/sources.list.d/docker.list > /dev/null

apt update
apt install -y docker-ce docker-ce-cli containerd.io
```

Finally, we installed Docker Compose and verified both installations:

```
curl -L "https://github.com/docker/compose/releases/latest/download/\
    docker-compose-$(uname -s)-$(uname -m)" \
```

```
    -o /usr/local/bin/docker-compose
chmod +x /usr/local/bin/docker-compose

docker --version
docker-compose --version
```

7.0.3 Configuring the Environment

First, we created a dedicated directory for the Overleaf configuration and navigated into it:

```
mkdir ~/overleaf
cd ~/overleaf
```

We created a .env file to store environment variables, which proved cleaner than inline configuration:

```
OVERLEAF_MONGO_URL=mongodb://mongo/overleaf?directConnection=true
OVERLEAF_REDIS_HOST=redis
OVERLEAF_SITE_URL=http://138.197.20.201
OVERLEAF_APP_NAME=MyOverleaf
EMAIL_CONFIRMATION_DISABLED=true
```

7.0.4 Docker Compose Configuration

We then added the docker-compose.yml file with several important modifications from standard configurations. The final configuration was:

```yaml
version: '2.2'
services:
  sharelatex:
    restart: always
    image: sharelatex/sharelatex:latest
    container_name: sharelatex
    depends_on:
      mongo:
        condition: service_healthy
      redis:
        condition: service_started
    ports:
      - 80:80
    volumes:
      - ~/sharelatex_data:/var/lib/overleaf
    env_file:
      - .env

  mongo:
    restart: always
    image: mongo:6.0
```

```
    container_name: mongo
    command: [”--replSet”, ”rs0”]
    expose:
      - 27017
    volumes:
      - ~/mongo_data:/data/db
    healthcheck:
      test: echo 'db.runCommand(”ping”).ok' |
        mongosh localhost:27017/test --quiet
      interval: 10s
      timeout: 10s
      retries: 5
      start_period: 40s

  redis:
    restart: always
    image: redis:6.2
    container_name: redis
    expose:
      - 6379
    volumes:
      - ~/redis_data:/data
    healthcheck:
      test: [”CMD”, ”redis-cli”, ”ping”]
      interval: 5s
      timeout: 3s
      retries: 5
```

7.0.5 Deploying and Initializing

We composed the containers using Docker-Compose:

```
docker compose up -d
```

After waiting approximately 30 seconds for MongoDB to fully start, the MongoDB replica set was initialized, which was essential for preventing transaction-related errors:

```
docker exec mongo mongosh --eval \
   "rs.initiate({_id: 'rs0', members: [{_id: 0, host: 'mongo:27017'}]})"
```

We then restarted the Overleaf container to establish the connection with the initialized replica set:

```
docker compose restart sharelatex
```

7.0.6 Verification

Once we saw the runit daemon start message, we verified all containers were running properly:

```
docker ps
```

The Overleaf web interface is successfully hosted at http://138.197.20.201, confirming the deployment was complete and functional.

Chapter 8

Domain Names, SSL, and Versioning

– JGa, JGr, CS

8.1 Domain Name Configuration

8.1.1 Acquiring a Domain Name

We obtained a free domain name through the GitHub Student Developer Pack, which provides a one-year free domain registration through Namecheap. The domain overleafssw590group10.me was registered for hosting our Overleaf instance.

Steps to Obtain Domain

1. Sign up for the GitHub Student Developer Pack at https://education.github.com/pack
2. Verify student status with a valid .edu email address
3. Access the Namecheap benefit and register a .me domain

8.1.2 DNS Configuration

After acquiring the domain, we configured DNS records in Namecheap to point to our Digital Ocean Droplet at IP address 138.197.20.201:

- A Record: Host: @, Value: 138.197.20.201
- A Record: Host: www, Value: 138.197.20.201

We verified DNS was working with:

curl -I http://overleafssw590group10.me

8.2 SSL Certificate Configuration with Let's Encrypt

8.2.1 Overview

We implemented SSL certificates using Let's Encrypt with Nginx as a reverse proxy to handle HTTPS traffic. The setup includes automatic certificate renewal every 90 days.

8.2.2 Implementation

Step 1: Update Docker Compose Configuration

We added Nginx and Certbot services to docker-compose.yml:

```
version: '3.9'
services:
  mongo:
    image: mongo:6.0
    container_name: overleaf-mongo
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: rootpassword
    volumes:
      - ./mongo-data:/data/db
      - ./mongodb-keyfile:/data/mongodb-keyfile:ro
    restart: always
    command: ["--replSet", "overleaf", "--keyFile",
              "/data/mongodb-keyfile"]

  redis:
    image: redis:7
    container_name: overleaf-redis
    restart: always
    volumes:
      - ./redis-data:/data

  overleaf-app:
    image: sharelatex/sharelatex:latest
    container_name: overleaf-app
    depends_on:
      - mongo
      - redis
    expose:
      - "80"
    environment:
      OVERLEAF_MONGO_URL: mongodb://root:rootpassword@mongo/
```

```
      sharelatex?authSource=admin&replicaSet=overleaf
     OVERLEAF_REDIS_HOST: redis
     OVERLEAF_SITE_URL: https://overleafssw590group10.me
     OVERLEAF_BEHIND_PROXY: 'true'
     OVERLEAF_ADMIN_EMAIL: jamesgrant2225@gmail.com
     OVERLEAF_APP_NAME: MyOverleaf
   volumes:
     - ./data:/var/lib/overleaf
   restart: always

 nginx:
   image: nginx:alpine
   container_name: overleaf-nginx
   ports:
     - "80:80"
     - "443:443"
   volumes:
     - ./nginx/nginx.conf:/etc/nginx/nginx.conf:ro
     - ./nginx/certbot/conf:/etc/letsencrypt:ro
     - ./nginx/certbot/www:/var/www/certbot:ro
   depends_on:
     - overleaf-app
   restart: always

 certbot:
   image: certbot/certbot
   container_name: overleaf-certbot
   volumes:
     - ./nginx/certbot/conf:/etc/letsencrypt
     - ./nginx/certbot/www:/var/www/certbot
   entrypoint: "/bin/sh -c 'trap exit TERM; while :; do
     certbot renew; sleep 12h & wait $${!}; done;'"
```

Key changes: changed OVERLEAF_SITE_URL to https://, set OVERLEAF_BEHIND_PROXY to true, and exposed Overleaf only internally while Nginx handles external ports 80 and 443.

Step 2: Create Nginx Configuration

Created nginx/nginx.conf:

```
events {
   worker_connections 1024;
}

http {
```

```
    server {
        listen 80;
        server_name overleafssw590group10.me;

        location /.well-known/acme-challenge/ {
            root /var/www/certbot;
        }

        location / {
            return 301 https://$host$request_uri;
        }
    }

    server {
        listen 443 ssl;
        server_name overleafssw590group10.me;

        ssl_certificate /etc/letsencrypt/live/
          overleafssw590group10.me/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/
          overleafssw590group10.me/privkey.pem;

        client_max_body_size 50M;

        location / {
            proxy_pass http://overleaf-app:80;
            proxy_set_header X-Forwarded-For
              $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header Host $host;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "upgrade";
        }
    }
}
```

Step 3: Initialize MongoDB Replica Set

Overleaf requires MongoDB to run as a replica set. We created a keyfile and initialized it:

```
# Create MongoDB keyfile
openssl rand -base64 756 > ./mongodb-keyfile
chmod 400 ./mongodb-keyfile
```

```
sudo chown 999:999 ./mongodb-keyfile

# Start MongoDB
docker-compose up -d mongo

# Initialize replica set
docker exec -it overleaf-mongo mongosh -u root -p rootpassword \
  --authenticationDatabase admin \
  --eval "rs.initiate({ _id: 'overleaf', members:
    [ { _id: 0, host: 'mongo:27017' } ] })"
```

Step 4: Obtain SSL Certificate

We obtained the certificate from Let's Encrypt using Certbot:

```
# Stop Nginx to free port 80
docker-compose stop nginx

# Request certificate
docker run --rm -it \
  -v "$PWD/nginx/certbot/conf:/etc/letsencrypt" \
  -v "$PWD/nginx/certbot/www:/var/www/certbot" \
  -p 80:80 \
  certbot/certbot \
  certonly --standalone \
  -d overleafssw590group10.me \
  --email jamesgrant2225@gmail.com \
  --agree-tos \
  --no-eff-email

# Start all services
docker-compose up -d
```

Step 5: Verify HTTPS

We verified the SSL certificate was working:

```
curl -I https://overleafssw590group10.me
```

The site was now accessible at https://overleafssw590group10.me with a valid SSL certificate. The Certbot container automatically renews the certificate every 90 days.

8.3 Overleaf Container LaTeX Configuration

To ensure that all LaTeX packages were available for project compilation, we installed the full TeX Live distribution inside our Overleaf container and verified installation of the TeX Live 2025 version.

```
sudo docker exec -it overleaf-app bash
apt update
apt install -y texlive-full
pdflatex --version
```

Chapter 9

Github Actions

– JGa, JGr, CS

9.1 GitHub Actions Workflow Configuration

9.1.1 Overview

We implemented a GitHub Actions workflow to automatically compile our LaTeX document whenever changes are pushed to the repository. This workflow accomplishes three key objectives: adding a version number based on the Git commit count, compiling the LaTeX document using a GitHub runner, and saving the resulting PDF back to the repository.

9.1.2 Workflow File Structure

GitHub Actions workflows are defined in YAML files stored in the .github/workflows/ directory of the repository. We created our workflow file at .github/workflows/latex.yml.

9.1.3 Creating the Workflow Directory

First, we created the necessary directory structure in our repository:

```
cd /var/lib/overleaf/data/user_files/Overleaf590
mkdir -p .github/workflows
```

9.1.4 Workflow Configuration

We created the workflow file latex.yml with the following configuration:

```
name: Build Overleaf PDF
on:
  push:
    branches:
      - master
    paths-ignore:
```

```
      - '**.pdf'
permissions:
  contents: write
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout repository
        uses: actions/checkout@v4

      - name: Set changelist number
        id: version
        run: |
          echo "CHANGE_ID=$(git rev-list --count HEAD)" >> $GITHUB_ENV
          echo "Version: $(git rev-list --count HEAD)"

      - name: Update version in LaTeX file
        env:
          CHANGE_ID: ${{ env.CHANGE_ID }}
        run: |
          sed -i "s/Version:.*/Version: ${CHANGE_ID}/" dsnManual.tex || true

      - name: Insert git version into LaTeX
        run: |
          GIT_HASH=$(git rev-parse --short HEAD)
          echo "Inserting git hash $GIT_HASH into dsnManual.tex"
          sed -i "s/PLACEHOLDER/$GIT_HASH/g" dsnManual.tex

      - name: Compile LaTeX document
        uses: xu-cheng/latex-action@v3
        continue-on-error: true
        with:
          root_file: dsnManual.tex
          latexmk_use_xelatex: true
          latexmk_shell_escape: true
          args: -interaction=nonstopmode -file-line-error -f

      - name: Upload PDF to artifacts
        uses: actions/upload-artifact@v4
        with:
          name: dsnManual-pdf
          path: dsnManual.pdf

      - name: Commit PDF back to repo
        run: |
```

```
    git config user.name "github-actions"
    git config user.email "actions@github.com"
    git add dsnManual.pdf
    git diff --cached --quiet || git commit -m "Add compiled PDF version $CHANGE_ID"
    git push
```

9.1.5 Workflow Components Explained

Trigger Configuration

The workflow triggers automatically on every push to the master branch, but ignores changes to PDF files to prevent infinite loops:

```
on:
  push:
    branches:
      - master
    paths-ignore:
      - '**.pdf'
```

Version Number Generation

The changelist number is generated using Git's commit count, which provides a unique, incrementing version number:

```
echo "CHANGE_ID=$(git rev-list --count HEAD)" >> $GITHUB_ENV
```

This command counts all commits in the repository history and stores the value in an environment variable accessible to subsequent steps.

Version Placeholder Replacement

We defined a placeholder in our LaTeX master file (dsnManual.tex):

```
\newcommand{\gitversion}{PLACEHOLDER}
```

The workflow replaces this placeholder with the actual Git commit hash:

```
GIT_HASH=$(git rev-parse --short HEAD)
sed -i "s/PLACEHOLDER/$GIT_HASH/g" dsnManual.tex
```

LaTeX Compilation

The workflow uses the xu-cheng/latex-action@v3 action to compile the LaTeX document with XeLaTeX:

```yaml
- name: Compile LaTeX document
  uses: xu-cheng/latex-action@v3
  continue-on-error: true
  with:
    root_file: dsnManual.tex
    latexmk_use_xelatex: true
    latexmk_shell_escape: true
    args: -interaction=nonstopmode -file-line-error -f
```

Key parameters:

- latexmk_use_xelatex: true – Uses XeLaTeX engine for compilation
- latexmk_shell_escape: true – Allows shell commands (required for minted package)
- -f flag – Forces compilation to continue despite warnings
- continue-on-error: true – Prevents workflow failure on LaTeX warnings

9.1.6 Committing and Deploying the Workflow

After creating the workflow file, we committed it to the repository:

```bash
# Stage and commit the workflow
git add .github/workflows/compile-latex.yml
git commit -m "Add GitHub Actions workflow for LaTeX compilation"

# Push to GitHub (requires Personal Access Token with workflow scope)
git push origin master
```

9.1.7 Verification

After pushing the workflow, we verified its execution:

1. Navigate to the repository on GitHub: https://github.com/Jgalligan1/Overleaf590
2. Click the Actions tab
3. View the workflow run and check for successful completion
4. Download the compiled PDF from the Artifacts section or view it directly in the repository

Chapter 10

Grafana

– JGa, JGr, CS

In this chapter, we discuss the setup of Prometheus, Node Explorer, and Grafana. Prometheus is a tool that monitors and collects metrics from various targets. Node Exporter exposes detailed operating system metrics on the host including CPU, memory, disk, and network usage. Prometheus scrapes these metrics from Node Exporter and stores them for analysis and visualization in Grafana.

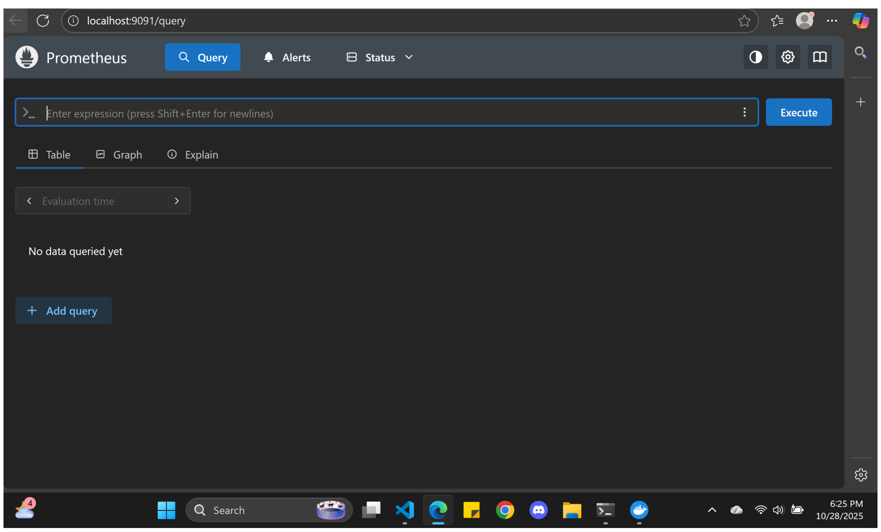

Figure 10.1: Prometheus running through Docker on http://localhost:9091.

Grafana provides dashboards that sync to the Prometheus data source, which can be refreshed at specified intervals, allowing for easy viewing of the host's real-time system performance metrics. The Grafana Dashboard 1860, the "Node Exporter Full" dashboard, visualizes key indicators of system health, including CPU load, memory utilization, RAM usage, and disk I/O activity. Together, these tools create a stack that enables easy monitoring and diagnosis of performance issues on the host system, as seen in Figure 10.2.

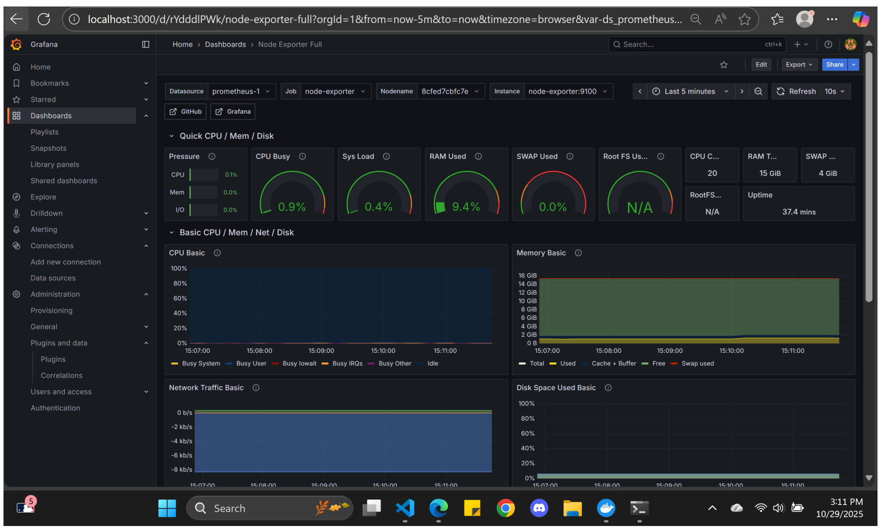

Figure 10.2: Live Grafana dashboard 1860 with local host performance metrics displayed.

Chapter 11

Jenkins Testing

– JG, CS

A Continuous Integration (CI) pipeline was created using Jenkins to automatically build and test a Python project hosted on GitHub. The process began by setting up a local Jenkins container through Docker and connecting it to a public GitHub repository. A declarative Jenkinsfile was written to define the pipeline stages, including dependency installation, test execution, and report generation.

The pipeline automatically cloned the repository, installed dependencies from requirements.txt, and executed unit tests using pytest. Jenkins then displayed the results in the console output, indicating that two tests passed and one failed intentionally to confirm that the CI process detects code errors correctly.

This setup demonstrates how Jenkins integrates with GitHub to automate testing and ensure code reliability after every change.

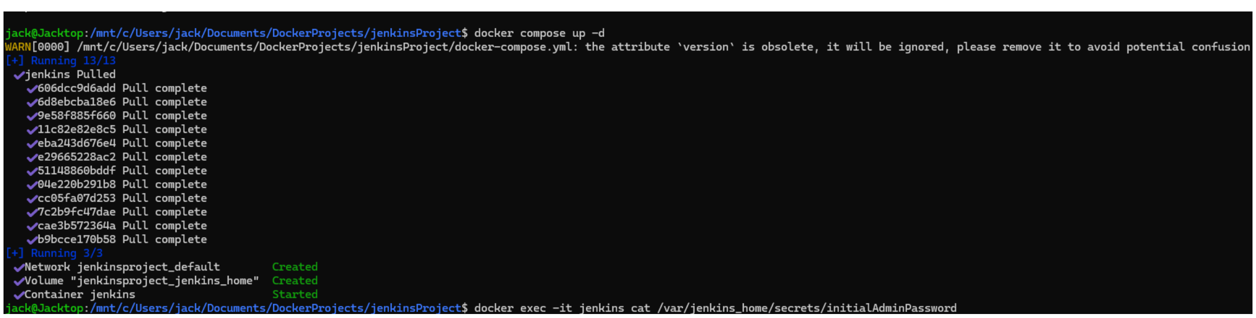

Figure 11.1: Jenkins environment creation

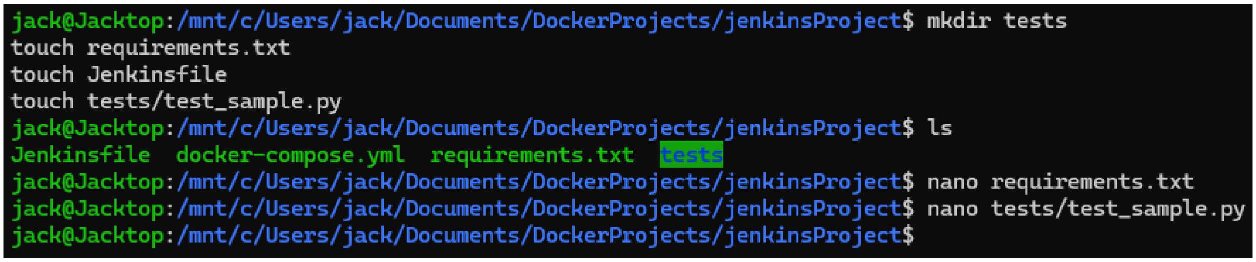

Figure 11.2: Jenkins file creation

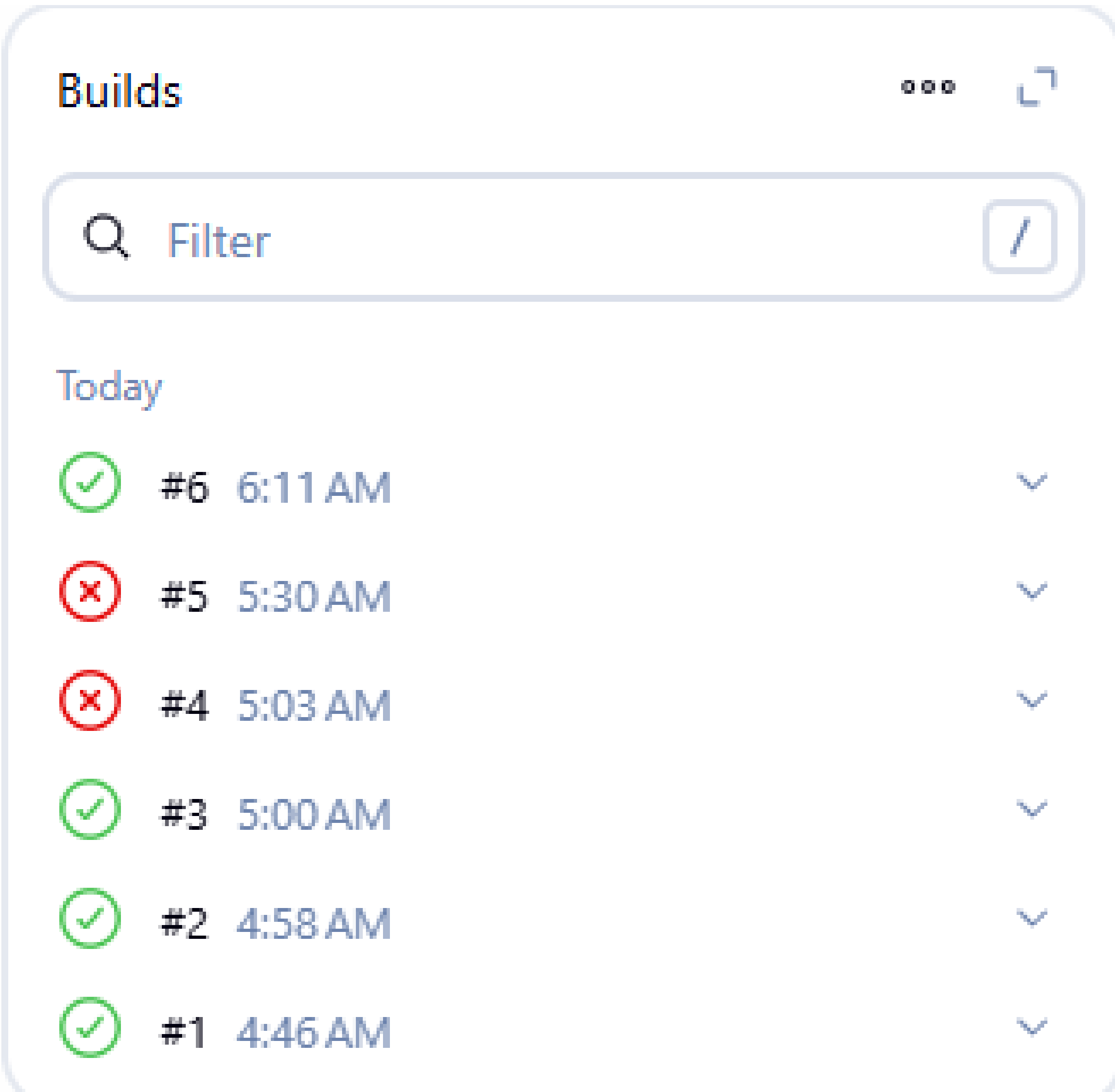

Figure 11.3: Jenkins overall testing

```
tests/test_sample.py ..F                                                  [100%]

=================================== FAILURES ===================================
_____________________________ test_failure_example _____________________________

    def test_failure_example():
>       assert 2 * 2 == 5  # This will fail intentionally
        ^^^^^^^^^^^^^^^^^
E       assert (2 * 2) == 5

tests/test_sample.py:8: AssertionError
- generated xml file: /var/jenkins_home/workspace/Python-Pytest-Demo/report.xml -
=========================== short test summary info ============================
FAILED tests/test_sample.py::test_failure_example - assert (2 * 2) == 5
========================= 1 failed, 2 passed in 0.02s ==========================
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Publish Report)
Stage "Publish Report" skipped due to earlier failure(s)
[Pipeline] getContext
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
ERROR: script returned exit code 1
Finished: FAILURE
```

Figure 11.4: Jenkins failure testing

```
+ ./venv/bin/pytest --junitxml=report.xml
============================= test session starts ==============================
platform linux -- Python 3.13.5, pytest-8.4.2, pluggy-1.6.0
rootdir: /var/jenkins_home/workspace/Python-Pytest-Demo
collected 3 items

tests/test_sample.py ...                                                 [100%]

- generated xml file: /var/jenkins_home/workspace/Python-Pytest-Demo/report.xml -
============================== 3 passed in 0.01s ===============================
```

Figure 11.5: Jenkins success testing

Chapter 12

Docker Load Balancing with Nginx

– JG, CS

This project demonstrates a containerized load balancing setup using Docker and Nginx. The goal was to distribute incoming requests across multiple backend web servers to ensure reliability and scalability.

Environment Setup

The environment consists of three Nginx containers: two backend servers (web1 and web2) and one load balancer. Each backend container serves a unique static page to verify traffic distribution.

The containers were defined in a docker-compose.yml file, specifying the services, their networks, and volume mounts for the HTML content:

```
services:
  web1:
    image: nginx
    volumes:
      - ./web1:/usr/share/nginx/html
    networks:
      - lbnet

  web2:
    image: nginx
    volumes:
      - ./web2:/usr/share/nginx/html
    networks:
      - lbnet

  loadbalancer:
    image: nginx
    ports:
      - "8080:80"
    volumes:
      - ./nginx/nginx.conf:/etc/nginx/nginx.conf:ro
    depends_on:
```

```
      - web1
      - web2
    networks:
      - lbnet

networks:
  lbnet:
```

Listing 12.1: docker-compose.yml file

Load Balancer Configuration

The load balancer uses Nginx's upstream directive to define the backend servers and distributes requests in a round-robin fashion. The configuration also forwards the original request headers to preserve client information:

```
events {}

http {
    upstream backend {
        server web1:80;
        server web2:80;
    }

    server {
        listen 80;

        location / {
            proxy_pass http://backend;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
        }
    }
}
```

Listing 12.2: nginx.conf for the load balancer

Testing and Verification

After bringing up the containers using docker compose up –build -d, the backend servers responded with distinct pages. web1 displayed:
while web2 displayed:

Repeatedly accessing the load balancer endpoint confirmed that requests were alternated between the two backend servers, demonstrating successful round-robin load balancing. The overall setup can be visualized as follows:

The link to the load-balancer is here, at http://157.245.10.33:8080/

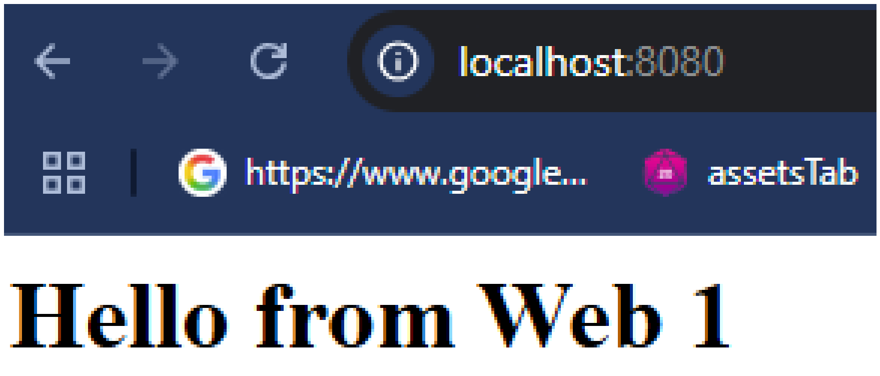

Figure 12.1: Response from web1 container

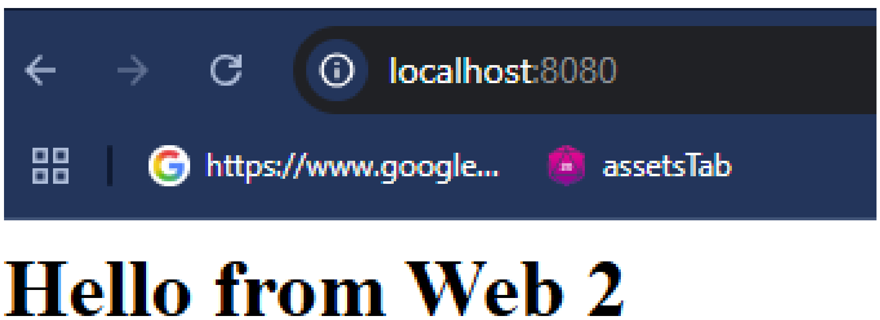

Figure 12.2: Response from web2 container

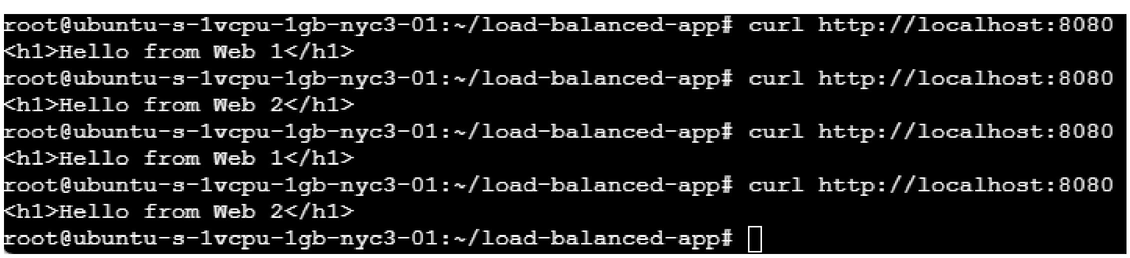

Figure 12.3: Docker network showing load balancer distributing requests to backend servers

Conclusion

This project shows how Docker and Nginx can be combined to implement a simple load-balanced web service. The configuration ensures that multiple backend containers can serve traffic reliably and that scaling up is straightforward. By examining the responses from each container and observing alternating content at the load balancer endpoint, we verified that traffic distribution is working as expected.

Bibliography

Index