Homework File

by

Justin Baumann

Gianna Cerbone

Thomas Ung

Spurthi Setty

Stevens.edu

October 16, 2025

Homework File

Justin Baumann
Gianna Cerbone
Thomas Ung
Spurthi Setty
Stevens.edu

This document provides the requirements and design details of the assignments from the Fall 2025 section of SSW590.

Table 1: Document Update History

Date	**Updates**
10/15/2025	Updates (TU&JB): • Added chapter on obtaining a domain.(Chapter **??**). • Added section on how domain was integrated with GitHub and appropriate steps taken. (Chapter **??**).
10/15/2025	Updates (SS): • Added chapter on How to compile Latex with all the packages, documenting all the steps and solutions to troubleshooting issues encountered.(Chapter **??**).
10/14/2025	Updates (GC): • Added chapter on SSL research for overleaf and how to add SSL certificates using docker. (Chapter **??**).
10/08/2025	Updates (JB&GC&TU): • Updated passwords (Chapter 2) with new information to allow user access to Bugzilla and course-related services • Updated hosts (Chapter 4) with new information to allow user access to Bugzilla and course-related services

Table 1: Document Update History

Date	Updates
10/06/2025	Overleaf on Digital Ocean (SS): • Added Overleaf Chapter (Chapter **??**) and detailed steps I took to get Overleaf community edition running on a port • Detailed my troubleshooting efforts to fix issues I ran into
10/04/2025	Bugzilla on Digital Ocean (SS): • Added Bugzilla Chapter (Chapter **??**) and detailed steps I took to get Bugzilla running on a port
09/29/2025	LaTeX Docker (GC): • Added LaTeX Docker (Chapter **??**) with description of writing done and Docker file code changed.
09/29/2025	Website Refactor (GC&JB&TU&SS): • Added section on Website refactoring (Chapter 7) with updated JavaScript code and UML Class Diagram for ColorController class.
09/29/2025	AWS Deployment (SS): • Followed instructions to deploy Two button app on AWS, troubleshooting issues and documented steps in detail in (Chapter 7) • Included link for successfully deployed website
09/17/2025	Project Proposal (GC&JB&TU&SS): • Added Project Proposal (Chapter 6) with description (section 6.1)
09/10/2025	Linux Commands (GC&JB&TU&SS): • Added LinuxCommands (Chapter 5) demonstrating bash command output and solution to Linux ProblemSet
09/09/2025	Introduction and Setup (GC&JB&TU&SS): • Updated the course introduction (Chapter 1) and Glossary
09/03/2025	Hosts (GC&JB&TU&SS): • Created a Hosts chapter and add a long table with names of hosts you will be configuring for your development environment (Chapter 4).
09/03/2025	Kanban Setup (GC&JB&TU&SS): • Added Kanban Setup chapter. (Chapter 3).

Table 1: Document Update History

Date	**Updates**
09/03/2025	Passwords (GC&JB&TU&SS): • Added Passwords chapter (Chapter 2). • Added a table with user/password/server rules.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

– Justin Baumann, Gianna Cerbone, Thomas Ung, Spurthi Setty

SSW-590: DevOps Principles and Practices (Fall 2025) teaches the culture, principles, and tools behind modern DevOps. It covers software lifecycles, configuration management, automated testing, code infrastructure, monitoring, and containerization. It allows students to apply these in AWS with Docker through a hands-on service implementation, tying DevOps practices back to core software engineering life-cycle concepts.

Chapter 2

Passwords

– Justin Baumann, Gianna Cerbone, Thomas Ung, Spurthi Setty

Created a rule that is our own and included hints to what the passwords are while not listing the passwords.

User	root
Password	REDACTED
Server	167.99.54.162
Hint	SSH/root account on DigitalOcean. Password follows Anchor+SiteCode+Policy — long, mixed-case, includes digits and a special character.

User	admin
Password	REDACTED
Server	167.99.54.162:8080
Hint	Service admin user for Bugzilla. Password begins with “admin” and ends with a short numeric pattern.

User	ssetty2@stevens.edu
Password	REDACTED
Server	167.99.54.162:8090
Hint	Overleaf web app login. Hint: friendly English word (capitalized) + 4-digit number + one special character.

Chapter 3

Kanban Setup

– Justin Baumann, Gianna Cerbone, Thomas Ung, Spurthi Setty

3.1 Kanban Setup

This is the order of operations executed to set up a Kanban board in Atlassian JIRA.

1. Go to Atlassian and select Kanban.
2. Name your first project. We named ours SSW590.
3. Select types of work needed. We selected task and story.
4. Track work using status states including: To Do, In Progress, In Review, Done.
5. Select Finish.
6. Share with team members.

Chapter 4

Hosts

– Justin Baumann, Gianna Cerbone, Thomas Ung

Host Name	DigitalOcean Ubuntu Server
IP Address	167.99.54.162
Operating System	Ubuntu 22.04 (64-bit)
Specifications	2 GPUs, 8 GB RAM
Purpose	Main server hosting Docker and Overleaf instance. Used for testing deployments and connecting with GitHub for project credit setup.
Access Method	SSH via `ssh root@167.99.54.162`
Security Notes	Root password stored separately; SSH keys recommended. Docker containers isolated and managed manually.

Chapter 5

Linux Commands

– Justin Baumann, Gianna Cerbone, Thomas Ung, Spurthi Setty

5.1 Linux Bash Commands

Below is a screenshot of the Linux commands that were run.

```
gcerb@Gianna:~$ mkdir -p ~/lx-test && cd ~/lx-test
gcerb@Gianna:~/lx-test$ printf "alpha\nbeta\nGamma\ngamma\nbeta\n" > words.txt
gcerb@Gianna:~/lx-test$ printf "id,name,dept\n1,Ada,EE\n2,Linus,CS\n3,Grace,EE\n4,Denni
s,CS\n" > people.csv
gcerb@Gianna:~/lx-test$ printf "INFO boot ok\nWARN disk low\nERROR fan fail\nINFO shutd
own\n" > sys.log
gcerb@Gianna:~/lx-test$ dd if=/dev/zero of=blob.bin bs=1K count=48 status=none
gcerb@Gianna:~/lx-test$ mkdir -p src/lib tmp archive
gcerb@Gianna:~/lx-test$ printf "one two three four\n" > src/file1.txt
gcerb@Gianna:~/lx-test$ printf "two three four five\n" > src/file2.txt
gcerb@Gianna:~/lx-test$ ln -s src/file1.txt link-to-file1
ln: failed to create symbolic link 'link-to-file1': File exists
gcerb@Gianna:~/lx-test$ touch -t 202401020304 old.txt
```

Figure 5.1: Screenshot of Linux Bash commands executed.

5.2 Linux Problem Set

A) Navigation & File Ops

1. pwd
2. ls -A1
3. [-d tmp] && cp -v src/file1.txt tmp/
4. mv -v --preserve=timestamps old.txt archive/
5. touch notes.md (only if not exists: test -e notes.md ||
(continued) touch notes.md)
6. du -sh src

B) Viewing & Searching

```
7.  nl sys.log
8.  grep 'ERROR' sys.log
9.  tr '[:upper:]' '[:lower:]' < words.txt | tr -c '[:alnum:]' '[\n*]' |
(continued) sort -u | wc -l
10. grep -i '^g' words.txt
11. head -n 2 people.csv
12. tail -n 3 -f sys.log
```

C) Text Processing

```
13. cut -d',' -f2 people.csv | tail -n +2
14. sort -f words.txt | uniq
15. sed -i.bak 's/three/3/g' src/*
16. wc src/*.txt
```

D) Permissions & Ownership

```
17. chmod 700 tmp/
18. chmod -R g+x src/lib
19. stat -c "%a" src/file2.txt
20. chattr +a notes.md
```

E) Links & Find

```
21. test -L link-to-file1 && readlink -f link-to-file1
22. find . -type f -size +40k
23. find tmp/ -type f -mmin -10 -exec ls -lh {} +
```

F) Processes & Job Control

```
24. pstree -p
25. sleep 120 & echo $!
26. pkill -TERM -u "$USER" sleep
27. ps -eo pid,comm,%mem --sort=-%mem | head -n 6
```

G) Archiving & Compression

```
28. tar -czf src.tgz src/
29. tar -tzf src.tgz
30. tar -xzf src.tgz -C tmp src/file2.txt
```

H) Networking & System Info

```
31. ss -ltnp
32. ip route show default
```

```
33. uname -srm
34. last -n 5
```

I) Package & Services (Debian/Ubuntu)

```
35. dpkg -s coreutils | grep Version
36. apt-cache search ripgrep
37. systemctl is-active cron
```

J) Bash & Scripting

```
38. for f in src/*.txt; do echo "$f: $(cat "$f")"; done
39. awk -F',' '$3=="CS" && NR>1 {print > "cs.txt"}' people.csv
40. export X=42; echo $X; unset X
```

Chapter 6

Project Proposal

– Justin Baumann, Gianna Cerbone, Thomas Ung, Spurthi Setty

6.1 Project Proposal

6.1.1 Project Title: QuackOps User Interface

The QuackOps Senior Design project has the following mission statement: To develop an autonomous drone delivery system that uses AI for navigation and visual target recognition, and to provide fast, contactless, and efficient on-campus delivery of goods and food.

To aid in the QuackOps Senior Design project, our team will help in developing the graphic user interface with computer vision, AI, and real time updating components from the data gathered by the QuackOps drone. Our contribution will be the web based dashboard that allows users to define delivery parameters, track the drone's location in real time, manage geofences, and monitor fleet health and compliance logs. Our main focus will be the software and user centered interface using DevOps techniques and skills learned in class.

We will be using Jira KANBAN for task tracking and distribution. Github for source control. CI/CD Actions within GitHub is what we are considering for testing as it is implemented directly into GitHub. Our team needs to make sure that the interface is modular and can be integrated with whatever we are using to get the drone data and can also be able to communicate with the drone in some way. For this part - we will need to think about the drone project extensively and keep both projects intertwined.

Chapter 7

AWS Deployment

– Justin Baumann, Gianna Cerbone, Thomas Ung, Spurthi Setty

7.1 Overview

These instructions are for a Windows machine, assuming Docker and AWS CLI are already installed and an AWS account has been created. These are step-by-step instructions to:

1. Configure AWS CLI with IAM Identity Center (SSO).
2. Push a Docker container image to Amazon Elastic Container Registry (ECR).
3. Deploy the container image to AWS App Runner.

7.2 Set up Environment

1. Ensure that Docker Desktop is installed and open the application
2. Open command prompt and navigate to the directory which contains the Dockerfile for the color button app.
3. Verify installation of AWS CLI by running the following in your command prompt

   ```
   aws --version
   ```

   You should get an output something like

   ```
   aws-cli/2.15.54 Python/3.11.8 Windows/10 exe/AMD64
   ```

7.3 Set up IAM Identity Center

1. Login into you AWS Console
2. Navigate to IAM Identity Center, and click enable
3. On the left hand menu, click on Users and then the Add User button on the top right

4. Enter the specified username, email and Name. I created a user with the following information
    - username: spurthi
    - email: spurthi.setty@gmail.com
    - Display name: Spurthi Setty
5. Click on next - no need to fill out additional information or add the user to a group
6. Select your preferred method of creating a password, I used a one time code, and set up 2FA with my Microsoft authenticator app.
7. Follow instructions to verify your email for your user
8. In your user, click on the tab for AWS Accounts, and then the button called Assign Accounts
9. Click on Create permissions set → predefined → Administrator Access
10. Assign the access for your user and wait for the confirmation message
11. Go back to AWS → IAM Identity Center → Settings. Here you should see a AWS access portal URL. This is your SSO Start URL for the next section. For me it was

    `https://d-906629391d.awsapps.com/start`

7.4 Configure AWS CLI with SSO

1. Run the following command

    `aws sso login`

2. You will be prompted to enter a series of inputs, here are the values to provide
    - SSO session name: my-sso (or any name)
    - SSO start URL: https://d-906629391d.awsapps.com/start (or whatever you AWS Access portal URL is)
    - SSO region: us-east-1
    - SSO registration scopes: (blank)
3. The browser will open the url, and the command prompt will aslo provide the URL to an SSO authorization page.
4. Enter the username and password on the page for the user you created in the previous section.
5. You will then be asked to input a 6 digit code on your browser from you command prompt, or asked to confirm the code.
6. Approve any permissions and you should get a confirmation message that your request has been approved.
7. You can now close this tab from your browser

8. Confirm you have sucessfull logged in by running the following command
9. Ensure that you are logged in as the user you created with admistrator access. If you are not, then try the sso command again. An example of the expected output for the previous step is

7.5 Set Environment Variables (Windows CMD)

Run the following commands, adjusting names and values are needed. the AWS_ACCOUNT_ID should correspond to the value for account in the previous step. The CONTAINER_PORT should correspond to whatever port is specified in you Dockerfile.

7.6 Create an ECR repository

1. Run the following command to describe and create an ECR repository
2. The output should look something like this
3. Confirm that the ECR was created by checking it on your AWS console under Elastic Container Registry

7.7 Build, Tag, and Push Docker Image

1. Login to docker by running the following command You should get a message that Login succeeded
2. Build you Docker container by running the following command If successfull, you terminal should look something like
3. Tag it to your ECR by running the following command
4. Push your docker container by running the following command If sucessfull, you should see these Images populate in your AWS console within this ECR
5. Very that the image is pushed by running the following command

   The output should look like this, with whatever you specified the image tag as:

7.8 Create App Runner ECR Access Role

1. Create the role:
2. Attach the policy:
3. Save the role ARN in an environment variable:

7.9 Deploy with App Runner

1. Create the App Runner service:

7.10 Verify Service and Get URL

1. Check the service status and retrieve the URL:
2. You can also get the URL by going to AWS App Runner in your console, and clicking the url in the default domain. Here is the App Runner service URL for when I set it up:

   You can also access it here: https://8fjyjsrizv.us-east-1.awsapprunner.com/

7.11 Website Refactor

To improve maintainability and follow object-oriented design principles, we refactored the JavaScript for the two-buttons application into a class-based design. The `ColorController` class encapsulates all button logic.

7.11.1 Updated JavaScript Code

7.11.2 UML Class Diagram

The UML diagram in Figure **??** illustrates the structure of the `ColorController` class.

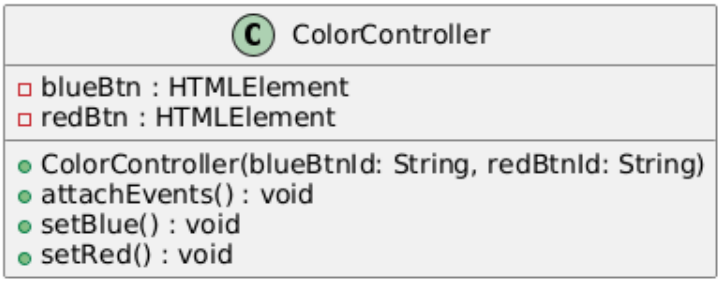

Figure 7.1: UML Class Diagram for the `ColorController` class

Chapter 8

LaTeX Docker

– Justin Baumann, Gianna Cerbone, Thomas Ung, Spurthi Setty

8.1 Overview

This chapter demonstrates building a Docker container that compiles LaTeX documents with TeX Live, similar to how Overleaf runs builds.

8.2 Dockerfile

8.3 Build and Run

Chapter 9

Bugzilla

– Spurthi Setty

9.1 Setting Up Bugzilla in Docker on DigitalOcean

This section documents the detailed procedure for setting up Bugzilla using Docker on a DigitalOcean Ubuntu 22.04 Droplet. The setup uses Docker Compose with two services: a MariaDB database and an Apache-based Bugzilla application container.

9.1.1 Environment Setup

1. **Create the Droplet:** - OS: Ubuntu 22.04 LTS - Size: 2 vCPUs, 4 GB RAM (recommended minimum) - Note the public IP address (e.g., `167.99.54.162`).
2. **Update and Install Docker:**
3. **Verify Docker Installation:**

9.1.2 Deploying Bugzilla via Docker Compose

1. **Create Directory Structure:**
2. **Create the Docker Compose File:**
3. **Start the Containers:**

   The Bugzilla service listens on port `8080` on the host, mapped to port `80` inside the container.

9.1.3 Configuring Bugzilla Inside the Container

1. **Access the Running Container:**
2. **Create and Configure the Database:**
3. **Set Up Bugzilla Configuration:**
4. **Initialize Bugzilla:**

   The script validates Perl modules, initializes the database schema, and generates the `params.json` configuration file.

9.1.4 Resolving Apache Configuration and Permissions

1. **Fix Permissions:**
2. **Replace the Default Apache Configuration:**
3. **Restart Apache (no systemd available):**

9.1.5 Testing and Verification

1. **Verify Local Connectivity:**

   Expect either `HTTP/1.1 200 OK` or `302 Found (index.cgi)`.
2. **Verify Host Mapping:**
3. **Access the Web Interface:**

   Log in with:

   ```
   admin@example.com / admin123
   ```

9.1.6 Troubleshooting Notes

- **403 Forbidden Error:** Fixed by defining explicit `Require all granted` and enabling CGI execution in Apache config.
- **500 Internal Server Error:** Caused by missing `params.json`; running `perl checksetup.pl` regenerates it.
- **Apache Startup Errors:** If logs are missing, recreate them:
- **Testing Without Browser:** Use:

9.1.7 Final Result

Bugzilla was successfully deployed and is accessible at:

http://167.99.54.162:8080/

This configuration persists through container restarts, with all data stored in Docker volumes defined in the Compose file.

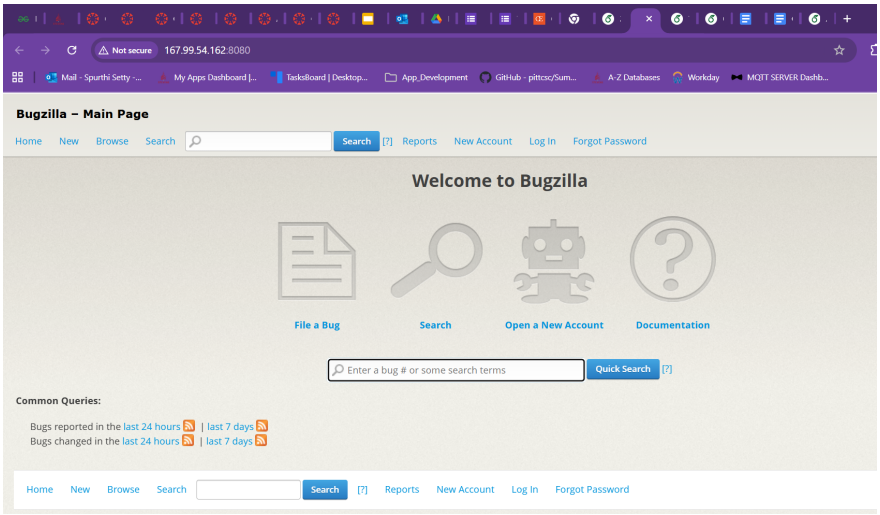

Figure 9.1: Screenshot of bugzilla working at http://167.99.54.162:8080/

Chapter 10

Overleaf

– Spurthi Setty

10.1 How to compile this chapter

Add to your preamble and compile with shell-escape to enable `minted`:

10.2 Context

We deployed **Overleaf Community Edition** on an existing DigitalOcean droplet that already served another site on `http://167.99.54.162:8080`. To avoid port conflicts, Overleaf was mapped to port `8090`.

10.3 Prerequisites (Ubuntu 22.04/24.04)

10.4 Baseline deployment

10.4.1 Create working directory

10.4.2 Open firewall

10.4.3 Initial `docker-compose.yml`

We used `sharelatex/sharelatex` (the Overleaf CE image) with MongoDB and Redis. Note the **Overleaf-branded** env vars and the **new data path** `/var/lib/overleaf`.

10.4.4 Start the stack

10.5 Issues encountered and exact fixes

Below are the exact errors we hit and the precise commands that resolved them on this host.

10.5.1 (A) Wrong image / registry hiccup

Symptom: Pull errors for `ghcr.io/overleaf/overleaf` (access denied/registry auth).
Fix: Switch to Docker Hub image `sharelatex/sharelatex`.

10.5.2 (B) Legacy bind mount path: `/var/lib/sharelatex`

Symptom: Container logs show rebranding guard refusing to start due to old path.
Fix: Use the new path `/var/lib/overleaf` in the bind mount.

10.5.3 (C) Legacy env var names: `SHARELATEX_*`

Symptom: `000_check_for_old_env_vars_5.sh` refuses startup, listing `SHARELATEX_MONGO_URL`, `SHARELATEX_REDIS_HOS` `SHARELATEX_SITE_URL`.
Fix: Rename to the Overleaf-branded variants.

10.5.4 (D) Connection reset / app crash loop due to Mongo transactions

Symptom: Logs show: `Transaction numbers are only allowed on a replica set member or mongos`.
Cause: Overleaf 5+ expects MongoDB with transactions support (i.e., a replica set), even for a single node.
Fix: Run Mongo as a single-node replica set and update the connection string.

Step D1: Add override to enable replica set and connection string

Step D2: Recreate the stack

Step D3: Initialize the replica set (one-time)

Step D4: Verify replica set is healthy

10.5.5 (E) Optional: tune kernel warning from Redis

10.6 Verification commands we used

10.7 Accessing the site

First-time admin setup (*Launchpad*): http://167.99.54.162:8090/launchpad After creating the admin, use the main URL: http://167.99.54.162:8090

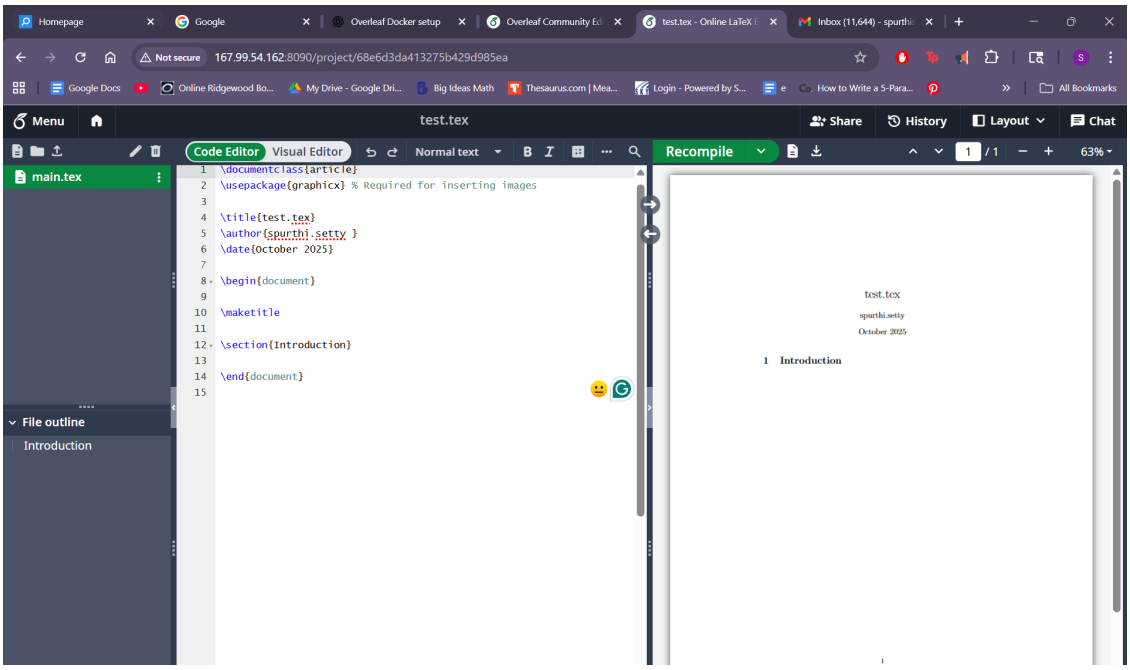

Figure 10.1: Screenshot of Compiled File at http://167.99.54.162:8090

10.8 Maintenance cheatsheet

10.9 Final status

After applying fixes (B) path update, (C) env var rename to `OVERLEAF_*`, and (D) enabling a single-node Mongo replica set, Overleaf CE started cleanly and became reachable at http://167.99.54.162:8090. The `/launchpad` route was used to create the initial admin account.

10.10 Creating a New User Account

To create a new account manually from the administrator dashboard, follow these steps:

1. **Login as an administrator.** Open your Overleaf instance and sign in with your admin credentials.
2. **Open the Manage Users panel.** Click on your profile icon at the top right corner and select `Manage Users`.
3. **Register the new user.** In the user management form, enter the email address of the new user and click `Register`.
4. **Access the Set Password page.** Copy the `Set Password` URL generated for that user and paste it into your browser.
5. **Set a new password.** The page will prompt you to create a new password for the account. Enter the desired password and confirm it.

6. **Activate and log in.** After setting the password, click `Activate`. The user can now log in using the new credentials.

Chapter 11

SSL Research

– Gianna Cerbone

11.1 Overview

This chapter summarizes the research conducted on implementing SSL (Secure Sockets Layer) for Overleaf containers and images. The focus is on how to add SSL certificates, manage them using free services such as Let's Encrypt, and automate their renewal process. Let's Encrypt provides free certificates that expire every 90 days, making automated renewal a critical part of the configuration.

11.2 Architecture Considerations

In most deployments, Overleaf is hosted using Docker containers or the Overleaf Toolkit. The recommended approach is to run Overleaf behind a reverse proxy (commonly Nginx) that handles TLS termination. This means HTTPS traffic is decrypted at the proxy, which then forwards plain HTTP requests to the Overleaf application container.

Embedding SSL certificate management directly inside the Overleaf container is possible but discouraged. It adds unnecessary complexity and makes updates harder. Managing certificates at the proxy layer is simpler, more secure, and allows reuse of certificates for multiple services.

11.3 Using the Overleaf Toolkit with TLS Proxy Mode

For those using the Overleaf Toolkit, TLS support is already integrated. You can enable it by initializing the toolkit with:

```
bin/init --tls
```

This generates an Nginx configuration and placeholder certificates located in:

```
config/nginx/certs/overleaf_certificate.pem
config/nginx/certs/overleaf_key.pem
```

Replace these placeholder files with your actual SSL certificate and key. The following settings can be configured in `config/overleaf.rc`:

```
NGINX_ENABLED=true
NGINX_CONFIG_PATH=config/nginx/nginx.conf
NGINX_HTTP_PORT=80
TLS_CERTIFICATE_PATH=config/nginx/certs/overleaf_certificate.pem
TLS_PRIVATE_KEY_PATH=config/nginx/certs/overleaf_key.pem
TLS_PORT=443
```

After modifying these settings, re-run:

```
bin/up
```

to recreate and restart the containers.

If an external proxy handles SSL termination, add the proxy IP address to:

```
OVERLEAF_TRUSTED_PROXY_IPS
```

so that Overleaf correctly interprets forwarded HTTPS requests.

11.4 Using an External Reverse Proxy

Another common approach is to use an external reverse proxy, such as Nginx or Traefik, to manage SSL and forward traffic to the Overleaf container. This approach simplifies certificate management and keeps the Overleaf image lightweight.

11.4.1 Advantages

- TLS configuration is isolated from the Overleaf container.
- Certificates can be easily renewed and reused for other services.
- Simplifies upgrades to Overleaf since SSL is handled separately.

11.4.2 Example Nginx Configuration

```
server {
    listen 80;
    server_name overleaf.example.com;

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

    location / {
        proxy_pass http://127.0.0.1:8000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
```

```
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Real-IP $remote_addr;
    }
}

server {
    listen 443 ssl;
    server_name overleaf.example.com;

    ssl_certificate /etc/letsencrypt/live/overleaf.example.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/overleaf.example.com/privkey.pem;

    location / {
        proxy_pass http://127.0.0.1:8000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

11.5 Using Let's Encrypt for Free SSL Certificates

Let's Encrypt offers free SSL certificates that are valid for 90 days. To generate and install one using `certbot`, run the following commands:

```
sudo apt install certbot python3-certbot-nginx
sudo certbot certonly --webroot -w /var/www/acme-challenges \
    -d overleaf.example.com
```

After the certificate is installed, configure automatic renewal:

```
sudo certbot renew --post-hook "systemctl reload nginx"
```

The renewal command can be added to a cron job or systemd timer to run periodically.

11.6 Renewal and Automation

Because Let's Encrypt certificates expire every three months, it is essential to automate the renewal process. Recommended best practices include:

- Schedule automatic renewals using `certbot renew`.
- Reload or restart the Nginx proxy after renewal to apply the new certificate.
- Test renewal scripts regularly using the `--dry-run` flag.
- Ensure that HTTP port 80 is available for ACME challenge responses.

11.7 Installing Certificates Inside the Container (Not Recommended)

While possible, installing and managing SSL certificates inside the Overleaf container is not recommended. This approach introduces additional maintenance complexity, such as updating trust stores, managing permissions, and triggering restarts on renewal.

If necessary, certificates can be mounted as Docker volumes and renewed via a sidecar container running `certbot`. The containerized Overleaf application would then reload the updated certificates periodically.

11.8 Summary and Recommendations

- Use a reverse proxy such as Nginx to handle SSL termination.
- Use Let's Encrypt for free, automated SSL certificates that renew every 90 days.
- In the Overleaf Toolkit, enable TLS with `bin/init --tls` and replace the default certificates.
- If using an external proxy, ensure websockets and headers (`Upgrade`, `X-Forwarded-Proto`) are properly forwarded.
- Avoid embedding certificate management logic inside the Overleaf application container.

By following these practices, Overleaf can be securely deployed with HTTPS, using automated and renewable SSL certificates, ensuring both encrypted communication and reduced administrative overhead.

Chapter 12

Configuring Overleaf with Full LaTeX Package Support

– Spurthi Setty

12.1 Introduction

In order to compile LaTeX documents with advanced packages such as `tikz`, `pgfplots`, and `minted`, it is necessary to configure Overleaf's self-hosted Docker environment with the full TEX Live distribution. This chapter outlines the complete setup process, including Docker configuration, LaTeX package installation, troubleshooting missing packages, and environment adjustments to ensure Overleaf compiles documents without errors.

12.2 Setting Up Overleaf with Docker

We used the official `sharelatex/sharelatex` Docker image as a base and extended it with the full TEX Live distribution to ensure comprehensive package support.

12.2.1 Directory Structure

All files were placed under:

Key files include:

- `Dockerfile` – to install `texlive-full`
- `docker-compose.yml` – to define Overleaf, MongoDB, and Redis services

12.2.2 Dockerfile

The Dockerfile extends the base image and installs TEX Live:

12.2.3 docker-compose.yml

The Docker Compose configuration links Overleaf with MongoDB and Redis:

12.2.4 Enabling MongoDB Replica Set

Overleaf requires MongoDB to support transactions, which are only available in replica sets. After container startup, we enabled the replica set manually:

12.2.5 Building and Starting Containers

After creating the Dockerfile and docker-compose.yml, we built the custom image:

Then started all services:

12.2.6 Verifying Initial Package Availability

After building and starting the containers, we confirmed that LaTeX packages such as `tikz`, `pgfplots`, and `psfrag` were available by executing:

If no path was returned, we updated the environment variables inside the container to point to the correct TEX Live installation path and rebuilt the file name database with:

12.3 Troubleshooting Missing LaTeX Packages

After successfully deploying Overleaf with the full TEX Live distribution, we encountered several missing package errors when attempting to compile a Cornell University thesis template. This section documents the systematic troubleshooting process used to identify and install missing LaTeX packages.

12.3.1 Initial Environment Configuration

Our Overleaf deployment consisted of three Docker containers running on an Ubuntu virtual machine hosted on Digital Ocean:

- `sharelatex` – Overleaf application server (image: `overleaf-fulltex`)
- `overleaf-mongo-1` – MongoDB 6.0 database
- `overleaf-redis-1` – Redis 7 cache server

The containers were accessible via port 8090 on the host machine at `http://167.99.54.162:8090`, running TEX Live 2025.

12.3.2 Verifying Container Status

Before beginning troubleshooting, we verified that all containers were running properly:

Expected output:

12.3.3 Sequential Package Installation

Missing Package: `psfrag`

The first compilation error indicated a missing `psfrag.sty` file:

We verified whether the package was already installed:

The output confirmed the package was installed, but the TEX filename database needed updating:

We verified the package file location:

Result: `/usr/local/texlive/2025/texmf-dist/tex/latex/psfrag/psfrag.sty`

Finally, we restarted the container:

Missing Package: `fancyvrb`

After resolving the `psfrag` issue, compilation failed with:

We installed the missing package directly:

Missing Package: `algorithmic`

The next compilation attempt failed at line 40:

The `algorithmic` package is part of the `algorithms` collection:

Missing Package: `txfonts`

Further compilation revealed a missing font package:

We attempted to install the font package:

However, after installation, a font error occurred:

The `txfonts` package caused persistent font mapping issues. We resolved this by commenting out the package in the document preamble:

For documents requiring Times-style fonts, a modern alternative can be used:

To use this alternative, install the package first:

12.3.4 Installing Complete Package Collections

Installing packages individually became inefficient as each compilation attempt revealed a new missing package. To avoid this iterative process, we installed comprehensive package collections.

We installed the `collection-latexextra` collection, which includes hundreds of commonly-used LaTeX packages:

This collection includes packages such as:

- `fancyvrb` – Sophisticated verbatim text handling
- `algorithmic` – Algorithm typesetting
- `algorithm2e` – Alternative algorithm package
- `listings` – Source code formatting
- `tcolorbox` – Colored and framed text boxes
- `enumitem` – Customizable list environments

- And many others

After installation, we updated the filename database:

12.3.5 Compiler Configuration

Initial compilation used the `latex` compiler, which generates DVI output. This caused hyperref warnings:
We changed the compiler in the Overleaf web interface:

1. Navigate to **Menu** (top left corner)
2. Under **Settings**, locate **Compiler**
3. Change from `LaTeX` to **pdfLaTeX**
4. Click **Recompile**

This resolved the hyperref driver warnings and enabled direct PDF generation.

12.3.6 Document-Level Fixes

Header Height Adjustment

The `fancyhdr` package warned that the header height was too small:
We added the following line after `\usepackage{fancyhdr}` in the preamble:

Hyperref Deprecated Options

We removed deprecated options from the `hyperref` package declaration. Original code:
Updated code:

PDF Bookmark Unicode Warning

Line breaks (\\) in chapter or section titles cannot be included in PDF bookmarks. We used an optional argument for the PDF bookmark:

12.3.7 Persistence of Installed Packages

Docker containers are ephemeral by design. Without additional steps, all installed packages would be lost when the container is restarted. After all packages were successfully installed, we committed the container to a new Docker image:
We then modified the `docker-compose.yml` file to reference the new image:
After updating the configuration, we restarted the containers:

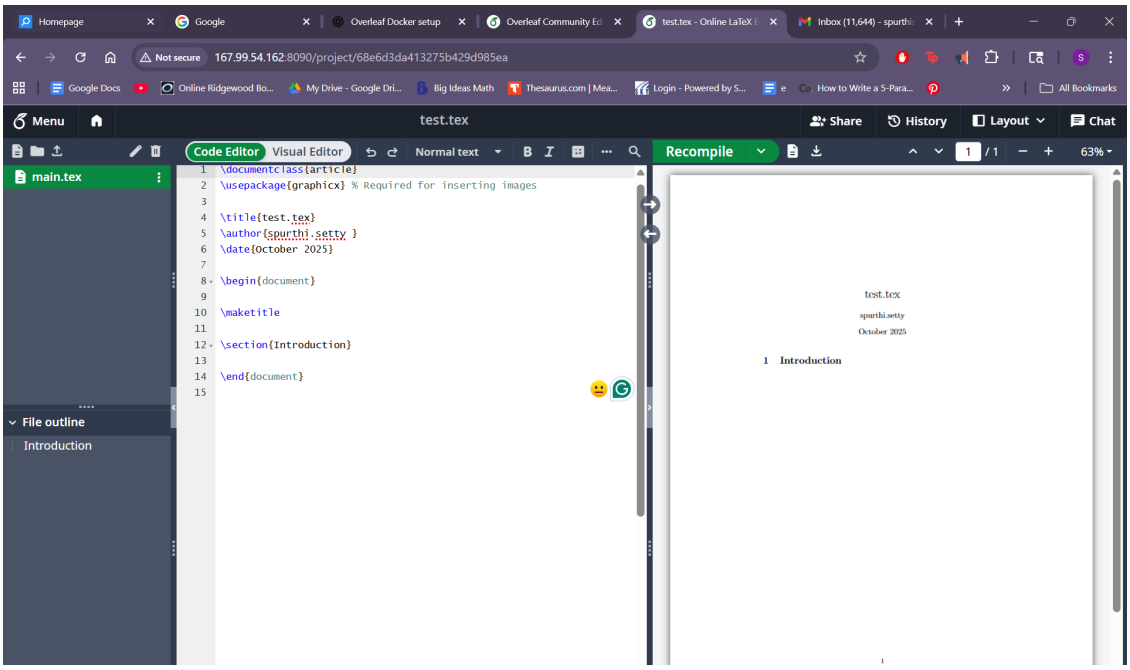

Figure 12.1: Successfully compiled template in Overleaf showing the PDF output, URL:http://167.99.54.162:8090/project/68eed5f5ba6fdc86c648466b

12.4 Final Working Configuration

After completing all troubleshooting steps and package installations, the Cornell thesis template compiled successfully without errors. Figure **??** shows the final working Overleaf instance with the successfully compiled document.

The deployment is accessible at:

http://167.99.54.162:8090/project/68eed5f5ba6fdc86c648466b

Key indicators of successful configuration include:

- All packages load without errors
- PDF compiles successfully with pdfLaTeX
- No missing package warnings
- Headers and footers render correctly
- Hyperlinks function properly in the PDF
- All bibliography and index features work as expected

Chapter 13

Domain Hosting

– Thomas Ung, Justin Baumann

13.1 Introduction

QuackOps.me was registered through Namecheap using the GitHub Student Developer Pack, which offers students a free domain for one year. Namecheap provides an easy domain management interface and integrates smoothly with GitHub Pages for hosting. By linking the QuackOps domain to GitHub, we were able to deploy and manage a live website directly from a repository—no paid hosting needed. This setup gives our group full control over custom DNS records, SSL, and subdomains (like http://overleaf.quackops.me:8090/project), making it an ideal foundation for hosting and connecting projects like our self-hosted Overleaf instance.

13.2 Connecting Overleaf to a Custom Domain

13.2.1 Overview

This section documents the configuration process used to connect our self-hosted Overleaf Community Edition (CE) instance to our custom domain, `quackops.me`, registered through Namecheap. The goal was to make Overleaf accessible via `overleaf.quackops.me` instead of using the server IP and port number.

13.2.2 Environment Setup

- **Server IP:** `167.99.54.162`
- **Overleaf instance URL:** `http://167.99.54.162:8090/project`
- **Domain registrar:** Namecheap
- **Registered domain:** `quackops.me`
- **Subdomain for Overleaf:** `overleaf.quackops.me`

13.2.3 Step 1: Configure DNS Records in Namecheap

We configured the following DNS records under the **Advanced DNS** tab for the domain `quackops.me`. This setup matches the configuration in Figure **??**.

Type	Host	Value / Target	TTL
A Record	overleaf	167.99.54.162	30 min
CNAME Record	www	quackops.me.	30 min
URL Redirect Record	@	http://overleaf.quackops.me:8090/project (Unmasked)	30 min
URL Redirect Record	www	http://overleaf.quackops.me:8090/project (Unmasked)	30 min

Table 13.1: Final DNS configuration for Overleaf domain setup

Explanation:

- The **A Record** maps the subdomain `overleaf.quackops.me` directly to the server IP address.
- The **CNAME Record** ensures that requests to `www.quackops.me` resolve to the same base domain.
- The two **URL Redirect Records** make both `quackops.me` and `www.quackops.me` automatically forward users to Overleaf's project dashboard.
- Both redirect records are **Unmasked**, which allows the browser to display the true Overleaf address rather than embedding it in a Namecheap frame.

13.2.4 Step 2: Verify DNS Propagation

After saving the records, DNS propagation can take up to one hour. To confirm that the records are active, run:

```
nslookup overleaf.quackops.me
dig overleaf.quackops.me +short
```

If the DNS has propagated successfully, these commands should return:

```
167.99.54.162
```

13.2.5 Step 3: Configure Overleaf Docker Environment

The Overleaf instance must be aware of its new domain. We set the environment variables when launching the Docker container:

```
docker run -d \
  --name sharelatex \
  -p 8090:80 \
  -e SHARELATEX_MONGO_URL=mongodb://overleaf-mongo-1:27017/sharelatex \
  -e SHARELATEX_REDIS_HOST=overleaf-redis-1 \
  -e SHARELATEX_SITE_URL=http://overleaf.quackops.me \
  -e SHARELATEX_BEHIND_PROXY=true \
  --network overleaf-net \
  unibaktr/overleaf:latest
```

13.2.6 Step 4: Test the Connection

Once DNS propagation is complete, open a browser and navigate to:

`http://overleaf.quackops.me:8090/project`

The Overleaf dashboard should now load successfully, showing all projects.

13.2.7 Step 5: Notes

- The setup uses `http` for simplicity. For production, a reverse proxy (e.g., Nginx or Caddy) with HTTPS should be configured.
- Using `Unmasked` redirects prevents issues where Namecheap frames obscure the destination URL.
- The `@` record ensures that navigating to the root domain (`quackops.me`) automatically forwards users to the Overleaf interface.

13.3 Integrating Domain With GitHub

13.3.1 Overview

This document outlines the process of connecting a self-hosted Overleaf Community Edition (CE) instance to GitHub and exporting all LaTeX source files from the Overleaf Docker container to a public repository. The process includes setting up SSH authentication, locating the Overleaf compile directory, copying project files, and pushing them to GitHub for backup and version control.

13.3.2 Environment Setup

- **Host:** Ubuntu server at `167.99.54.162`
- **Overleaf container name:** `sharelatex`
- **GitHub repository:** `git@github.com:JustBaumann/JustBaumann.github.io.git`

13.3.3 Step 1: Connect to the Server

```
ssh root@167.99.54.162
```

13.3.4 Step 2: Generate SSH Key on Ubuntu Host

Create a new SSH key pair and register it with GitHub for secure push access:

```
ssh-keygen -t rsa -b 4096 -C "server@quackops.me"
cat /root/.ssh/id_rsa.pub
```

Copy the key and add it in GitHub under: `Settings → SSH and GPG Keys → New SSH Key`
Verify connection:

```
ssh -T git@github.com
# Expected output:
# Hi JustBaumann! You've successfully authenticated, but GitHub does not provide shell access.
```

13.3.5 Step 3: Clone the GitHub Repository

```
cd /root
git clone git@github.com:JustBaumann/JustBaumann.github.io.git
git config --global user.name "JustBaumann"
git config --global user.email "overleaf@quackops.me"
```

13.3.6 Step 4: Locate LaTeX Source Files in Overleaf Container

Overleaf CE stores compiled project data under the `/var/lib/overleaf/data/compiles/` directory. To confirm their presence:

```
docker exec -it sharelatex bash -lc \
'find /var/lib/overleaf/data/compiles -type f -name "*.tex" | head'
```

13.3.7 Step 5: Copy Files from the Container to the Host

```
docker cp sharelatex:/var/lib/overleaf/data/compiles /root/overleaf_tex_backup
```

13.3.8 Step 6: Copy Only Source Files to the Repository

Create a destination folder and extract only LaTeX sources:

```
mkdir -p /root/JustBaumann.github.io/overleaf_sources
find /root/overleaf_tex_backup -type f \
  \( -name "*.tex" -o -name "*.bib" -o -name "*.cls" -o -name "*.sty" -o -name "*.bst" \) \
  -exec cp {} /root/JustBaumann.github.io/overleaf_sources/ \;
```

13.3.9 Step 7: Commit and Push to GitHub

```
cd /root/JustBaumann.github.io
git add .
git commit -m "Added LaTeX source files recovered from Overleaf compiles/"
git push origin main
```

13.3.10 Step 8: Verification

After pushing, visit:

`https://github.com/JustBaumann/JustBaumann.github.io`

All LaTeX source files should now be visible under the `overleaf_sources/` directory.

13.3.11 Notes

- The Overleaf CE instance used MongoDB for project metadata, but in this case, raw LaTeX sources were found under the compiled directory.
- Only relevant source files (`.tex`, `.bib`, `.cls`, `.sty`, `.bst`) were extracted; compiled outputs (`.pdf`, `.log`, `.aux`, etc.) were intentionally omitted.
- The SSH key created for the Ubuntu host allows seamless pushing to GitHub without re-entering credentials.

Chapter 14

Compiling Overleaf

– Spurthi Setty

This chapter outlines the steps required to compile an Overleaf-based LaTeX project directly from the command line, outside of the Overleaf web interface.

14.1 Steps Taken

Run the following commands to ensure that git is installed

Navigate to the github repository that we cloned in the previous step

Compile a tex file here with the following command

It should sucessfully compile as evidenced by the following screenshot

The output saved to main.pdf, we can commit this to github and check it out in the repository with the following commands

You can then see the compiled pdf, as evidenced here

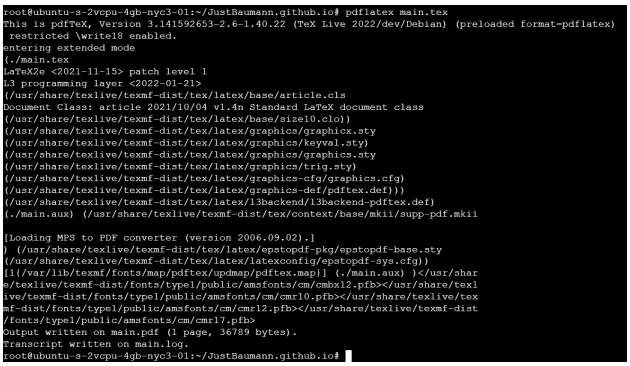

Figure 14.1: Cmd Output of compiled overleaf

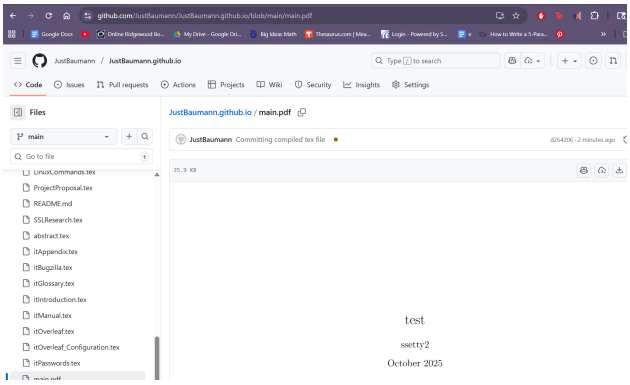

Figure 14.2: PDF Output of compiled overleaf

Appendix A

Appendix

– Justin Baumann, Gianna Cerbone, Thomas Ung, Spurthi Setty