Version Control I

Joachim Vandekerckhove

Winter 2025

What are SSH keys?

In the physical world, we keep things secure by using locks that are opened by keys. In cyberspace, we secure things by *SSH keypairs* that work the same way. A keypair consists of two files:

1. **A public key** – This is like the lock on the box. You can share this lock with anyone because they can't open the box with just the lock.
2. **A private key** – This is your key that opens the lock. You keep it secret and safe.

What are SSH keys?

When working with GitHub (or any similar service), **SSH keys** work as follows:

- Your **public key** is uploaded to GitHub.
- Your **private key** stays on your computer or container.
- When you try to connect to GitHub using your account, it will present you with a challenge that is encrypted using the public key. Your computer will then decrypt this using the private key, and GitHub will check if the challenge matches before you're allowed in.

SSH keys are both secure and convenient – no need to enter a password every time you connect.

Why use SSH keys with Docker and GitHub?

To access GitHub from inside a **Docker container** you need to set up SSH keys so the container can communicate securely with GitHub.

By storing your keys in a **persistent volume**, you can avoid regenerating keys every time you start the container.

This builds on your understanding of connecting VSCode to containers (003-connect-with-vscode) and starting your class container (002-start-the-class-container).

Installing an SSH keypair in a Docker container with a persistent volume

1. Set up a persistent volume for keys

When creating your container, make sure to add a volume – a shared folder between the container and your computer. In our original docker setup, we called it `/workspace`. This directory can store SSH keys persistently.

Let's spin up our container and add a subdirectory:

```
mkdir /workspace/secrets
cd /workspace/secrets
```

2. Generate SSH keys inside the Docker container

Inside the container, generate a new SSH keypair: `bash ssh-keygen -t ed25519 -C "joachim@class-container.cogs106" -f /workspace/secrets/id_ed25519` - **Options explained:** - `-t ed25519` specifies the key type. - `-C "joachim@class-container.cogs106"` adds a comment for identification. You can write anything here. - `-f /workspace/secrets/id_ed25519` saves the keypair in the mounted folder for persistence.

2. Generate SSH keys inside the Docker container

The command creates two files in `/workspace/secrets`: - `id_ed25519` (private key) - `id_ed25519.pub` (public key)

Print the public key: `bash cat /workspace/secrets/id_ed25519.pub` Copy the output.

3. Add the public key to GitHub

- Go to GitHub SSH settings.
- Click **New SSH Key**.
- Paste the public key you copied and save it.
    - Make sure you don't copy line breaks.

4. Configure the Docker container to use the key

- Link the private key to the SSH directory:

```
mkdir -p ~/.ssh
cp /workspace/secrets/id_ed25519 ~/.ssh/id_ed25519
chmod 600 ~/.ssh/id_ed25519
```

- Add GitHub's public keys (for verification):

```
ssh-keyscan github.com >> ~/.ssh/known_hosts
```

5. Test the connection

Inside the container, test the SSH connection:

```
ssh -T git@github.com
```

If it works, you'll see a message like:

```
Hi username! You've successfully authenticated.
```

Summary

By generating SSH keys inside the Docker container and saving them in a persistent volume: - You've enabled secure access to GitHub without needing passwords. - The keys are saved outside the container, so they remain available even if the container is restarted or recreated. - You're now set up to clone, pull, and push repositories from inside the Docker container.