Setting up SSH

A quick guide to setup remote access to your server

Step 1: Installing OpenSSH

This step won't be guiding you on how to actually install OpenSSH since I don't know which distribution you are running. But I can still link some documentation:

- Debian
- Alpine
- NixOS

Step 2: Generating a SSH key

SSH keys are used to safely connect to remote machines, without a password, among other things.

If you already have a SSH key, you can skip to the next step.

We are going to generate a SSH key. (no clue how that works on windows!)

Linux

On Linux, simply type `ssh-keygen -t ed25519`.
Leave default location (`~/.ssh/id_ed25519`).
For the password, you can leave it empty if you can guarantee that no one else will ever have access to that file, else set one.

Step 3: Copy SSH key to server

Now that we have a key, we can transfer it to our server.
This can be done with `ssh-copy-id`. Run: `ssh-copy-id <user>@<hostname>` and it should just work(TM).

If this didn't work, you can manually add the key yourself.
Print the key in your terminal and copy it: `cat ~/.ssh/id_ed25519.pub`.
Connect to your server: `ssh <user>@<hostname`.
Edit the authorized-keys file: `nano ~/.ssh/authorized_keys`.
Paste what you copied earlier.
Save and exit.

Logout of the server and try logging in again. It should not ask for a password.

Step 4: Securing OpenSSH

⚠ Before continuing: This will make login via password impossible. If you lose your SSH key, you will lose access to your server and it will require a reinstallation.

The OpenSSH server configuration should be located at `/etc/ssh/sshd_config` but we aren't going to edit this file. We are going to create a new file at `/etc/ssh/sshd_config.d/hardening.conf`. (add video)

```
X11Forwarding no

AllowAgentForwarding no

PermitEmptyPasswords no

MaxAuthTries 3

# NOTE: this will disable password authentication entirely! Setup SSH keys before applying this config!
PubkeyAuthentication yes
PasswordAuthentication no
PermitRootLogin no

Protocol 2
```

Now to apply these settings, we need to restart the ssh server with: `sudo systemctl restart sshd`, and voilà!

Step 5: Setting up fail2ban

We are now going to setup fail2ban, basically something than bans IPs from accessing your server if they fail to authenticate multiple times.

Installing

Installation varies from distro to distro, here you can check how.

Configuration

We are going to setup fail2ban to check ssh connections. You can configure it for many many things, even custom ones.

You can add this to `/etc/fail2ban/jail.d/jail.local`:

```
[sshd]
enabled = true
```

Restart fail2ban with `sudo systemctl restart fail2ban`, and it should work! You can check that the jail is set up properly by running `sudo fail2ban-client status`.

End word

The setup is now complete! You can somewhat safely make your server publicly accessible now, if it wasn't already.

Note that this does not make your server perfectly safe. SSH can now be considered safe, as login is only possible via SSH key. But every app is also an attack vector, and you should implement things to mitigate those vectors.