CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich

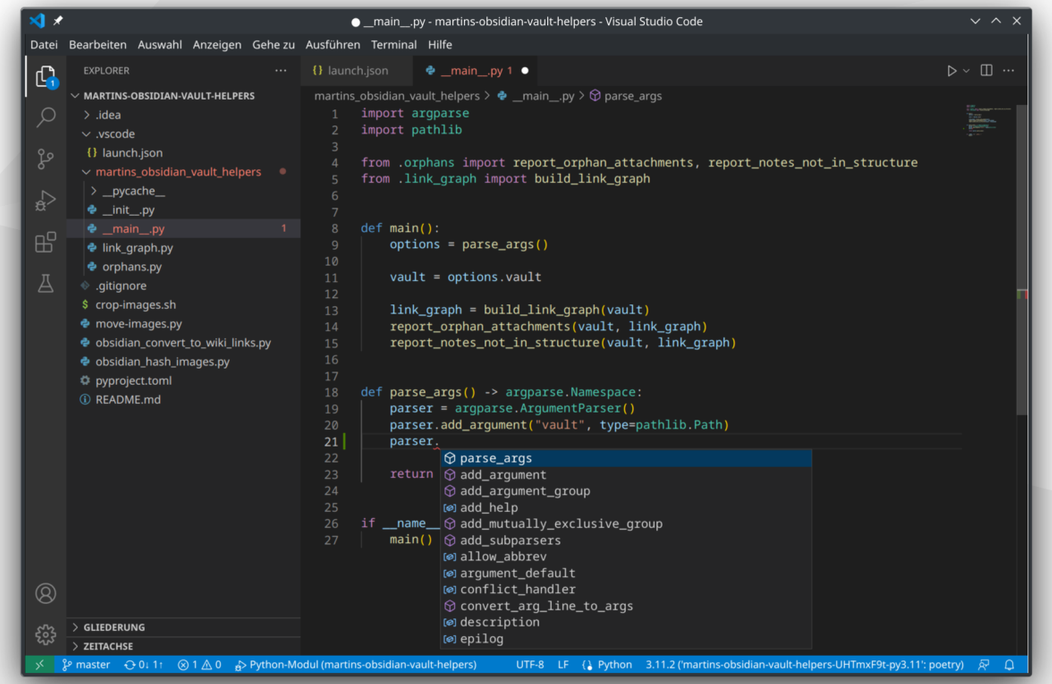
VS Code for Interactive Computing

Productivity and Development Tips

Prashanth Kanduri

Why use an IDE?

- Intuitive/familiar user experience
- Ease of managing files, visual previews
- Syntax highlighting
- Handy extensions for languages, frameworks and debugging
- Development aids like codebase awareness, code completion, etc
- Easy to setup!



Basic Setup

- Ensure that SSH config and 2FA works and SSH keys are refreshed
- `ssh <clustername>` on the local should get you to the shell of a cluster

SSH config

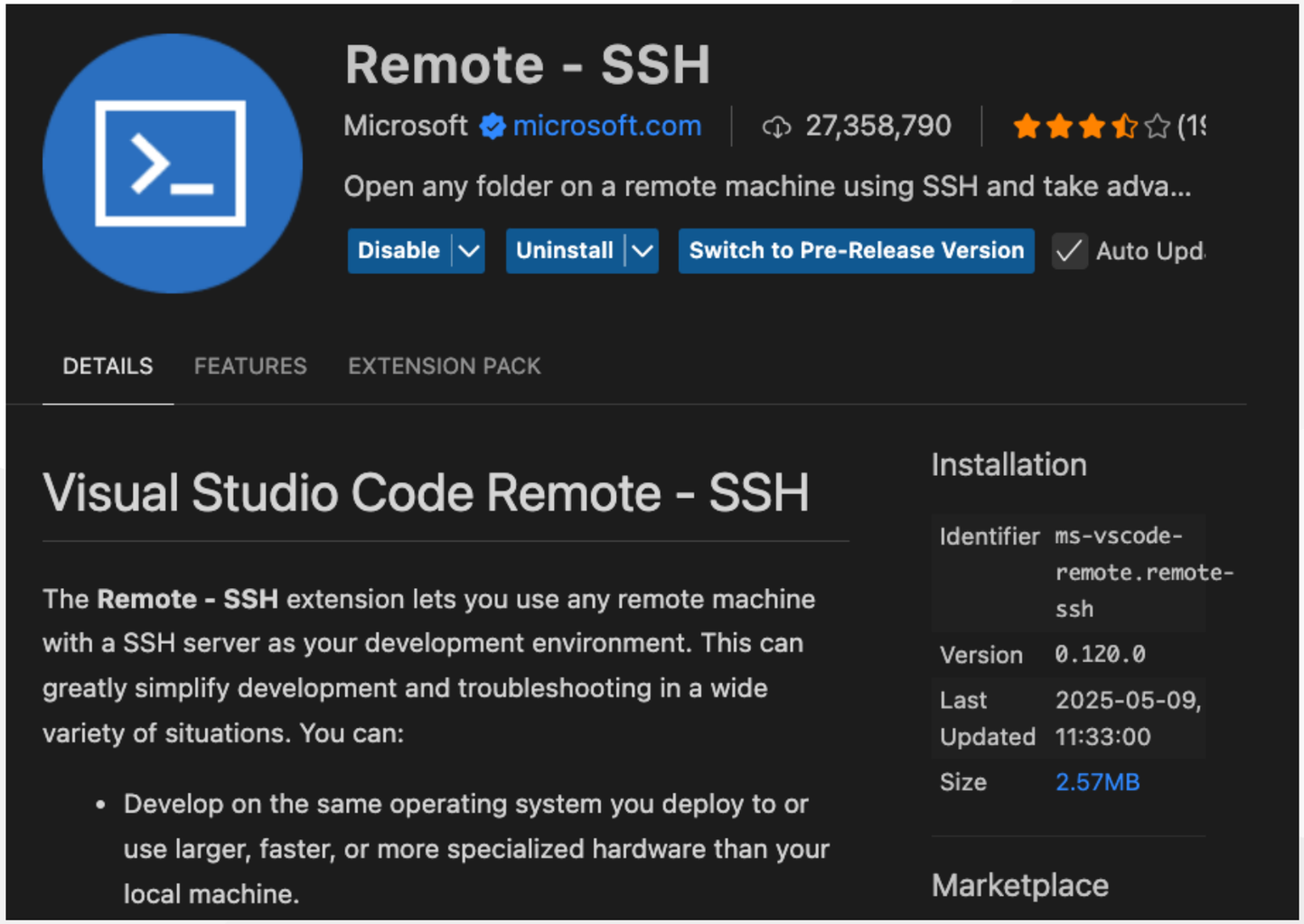
```
Host ela
    HostName ela.cscs.ch
    User <username>
    IdentitiesOnly yes
    IdentityFile ~/.ssh/cscs-key
    ForwardAgent yes
    ForwardX11 yes
    ForwardX11Trusted yes

Host daint
    HostName daint
    User <username>
    IdentitiesOnly yes
    IdentityFile ~/.ssh/cscs-key
    ProxyJump ela
    ForwardAgent yes
    ForwardX11 yes
    ForwardX11Trusted yes
```

Password-Less Login

```
➜  ssh daint

Last login: Sat Jul 19 02:20:42 2025
from ela5.cscs.ch
================================================
  IMPORTANT NOTICE FOR USERS of Alps (Daint)

  Documentation: Alps User Guide -
  https://confluence.cscs.ch/x/mQ_KMg

  Request support: Service Desk at
  https://support.cscs.ch

================================================
```

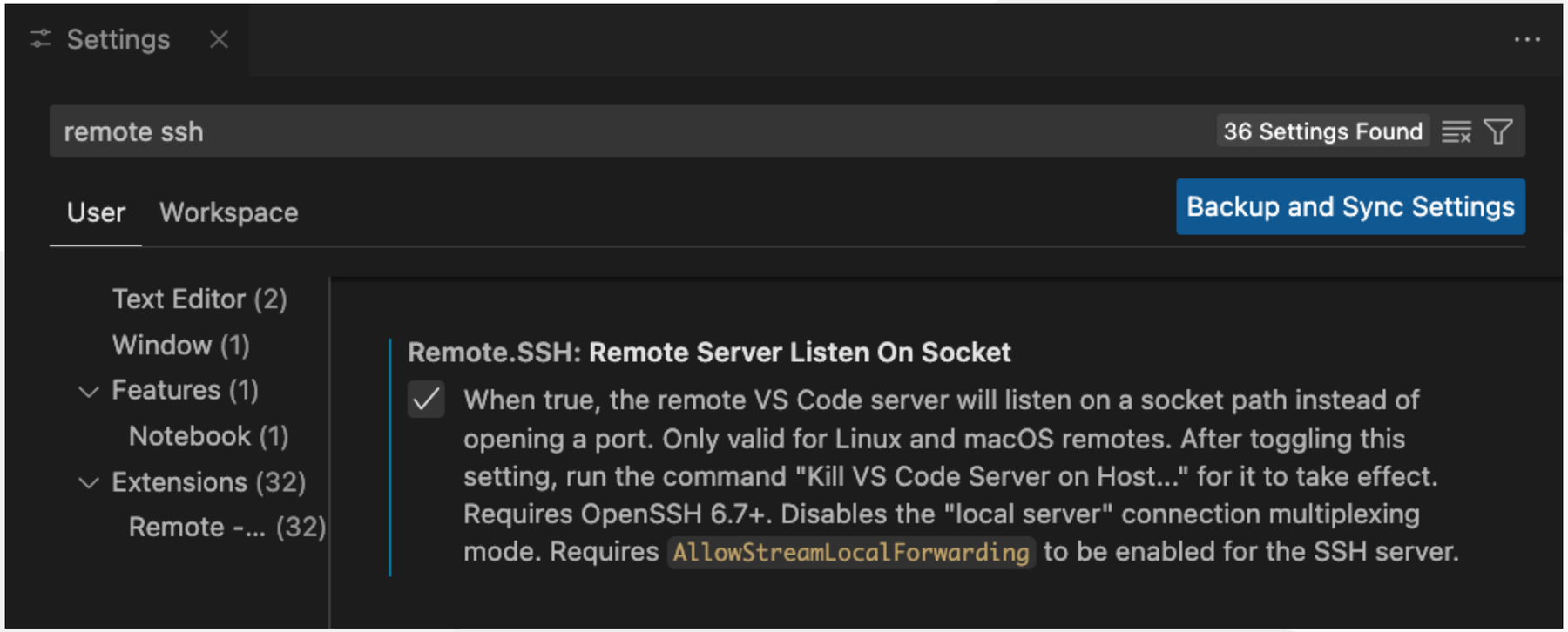
VS Code Setup

Install the Remote-SSH extension by Microsoft from the marketplace.



VS Code Setup cont...

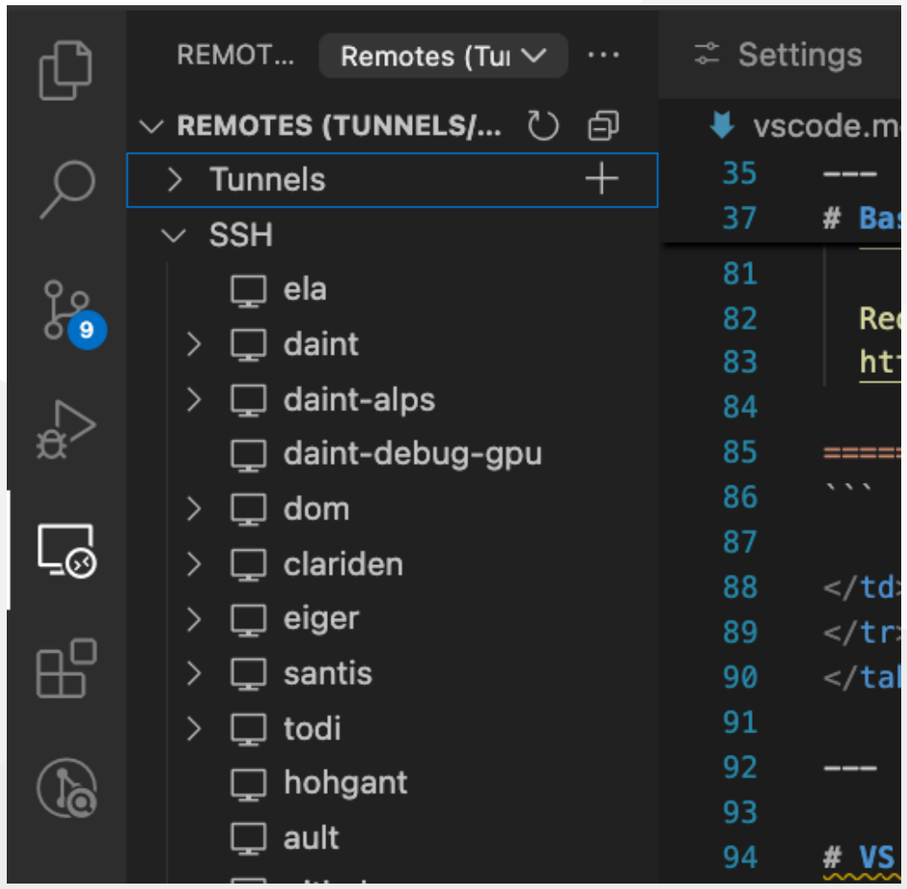
After installation, in the settings, scroll down to the extensions and find the section concerning the one above.
There activate the **Remote Server Listen On Socket** setting.



Connect to a Cluster

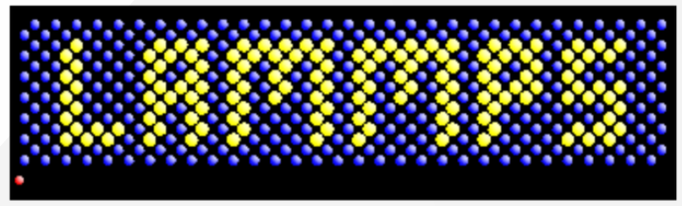
With these settings, restart VS Code and on the left-hand side, find the **Remote Explorer** with the hostnames setup in the `~/.ssh/config` file.

Select the cluster and the window will reload to login. One can select a folder to open there.



Loading Development Environments

PyTorch



NVIDIA CUDA

GCC

FAST. FLEXIBLE. FREE. GROMACS

python™

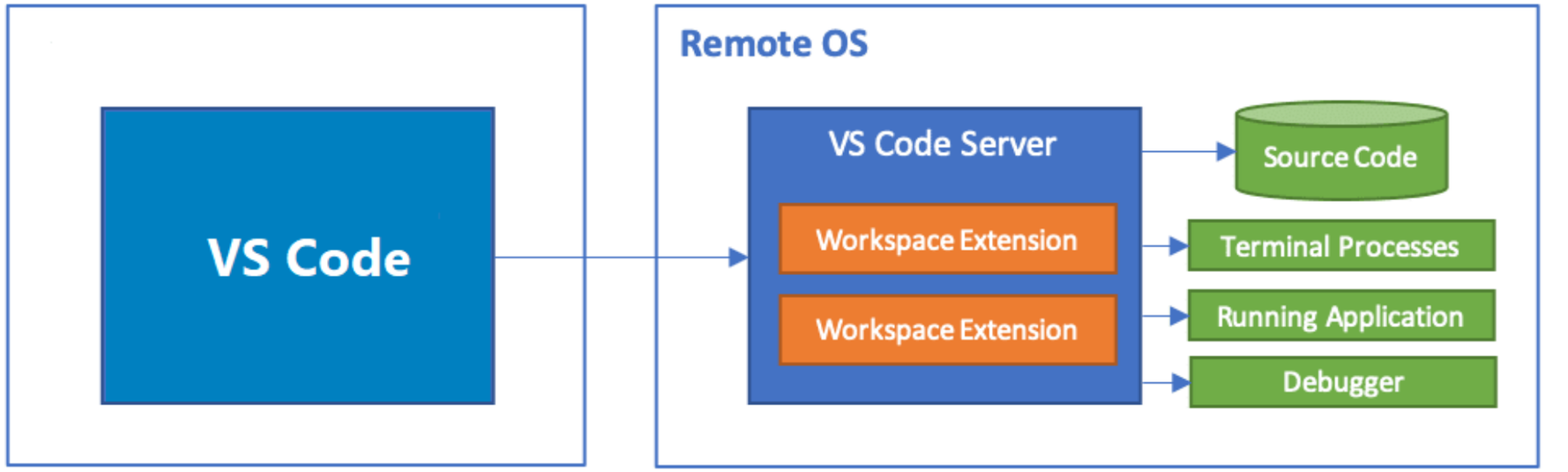
ICON

kokkos

GridTools 4python

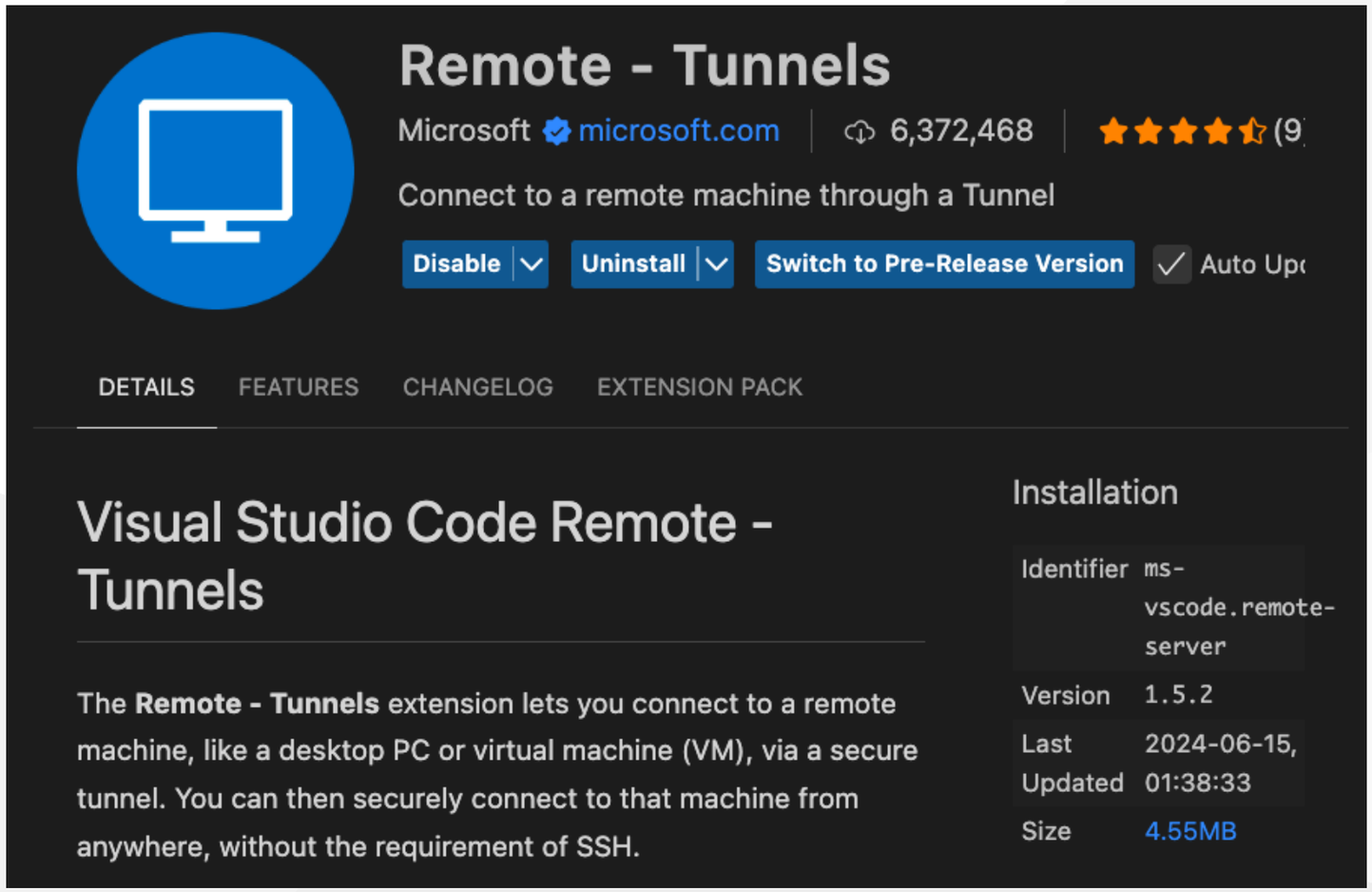
Working with Remote Tunnels

> The Visual Studio Code **Remote - Tunnels** extension lets you connect to a remote machine, like a desktop PC or virtual machine (VM), via a secure tunnel. You can connect to that machine from a VS Code client anywhere, without the requirement of SSH.



Setting up Locally

Install the **Remote Tunnels** extension from Microsoft on VS Code.



Setting up on Alps

Download the VS Code CLI tool `code`, which CSCS provides for easy download. There are two executables, one for using on systems with x86 or ARM CPUs respectively.

`ARM64` → `daint`, `clariden`, `santis`
`x86-64` → `eiger`, `bristen`

Download with the following commands (for ARM64) on the cluster:

```
wget https://jfrog.svc.cscs.ch/artifactory/uenv-sources/vscode/vscode_cli_alpine_arm64_cli.tar.gz
tar -xf vscode_cli_alpine_arm64_cli.tar.gz
```

We will now have an executable named `code` which we shall use. We can make it accessible from anywhere by adding its absolute path to the `PATH` environment variable in our `.bashrc` file.

```
export PATH=/location/of/code/executable:$PATH
```

Tunnel with UENV

On the shell of a cluster, start a uenv session and then a tunnel. For a `uenv` already loaded, one can use the following command.

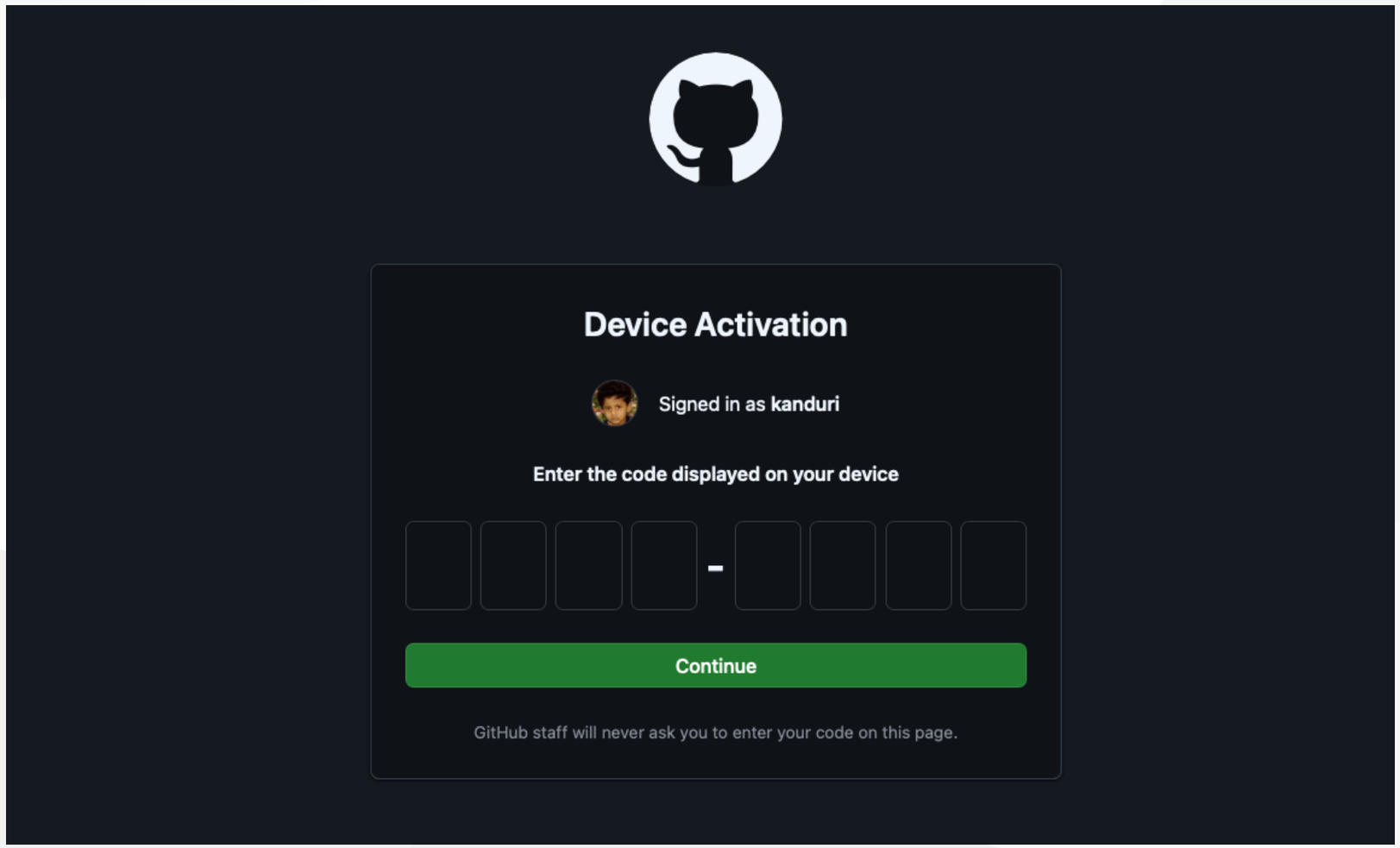
```
# start a uenv session on the login node
uenv start --view=default prgenv-gnu/24.11:v1
# then start the tunnel
code tunnel --name=$CLUSTER_NAME-tunnel
```

It will ask the user to sign in using Microsoft or GitHub account. We have reliably tested it with GitHub.

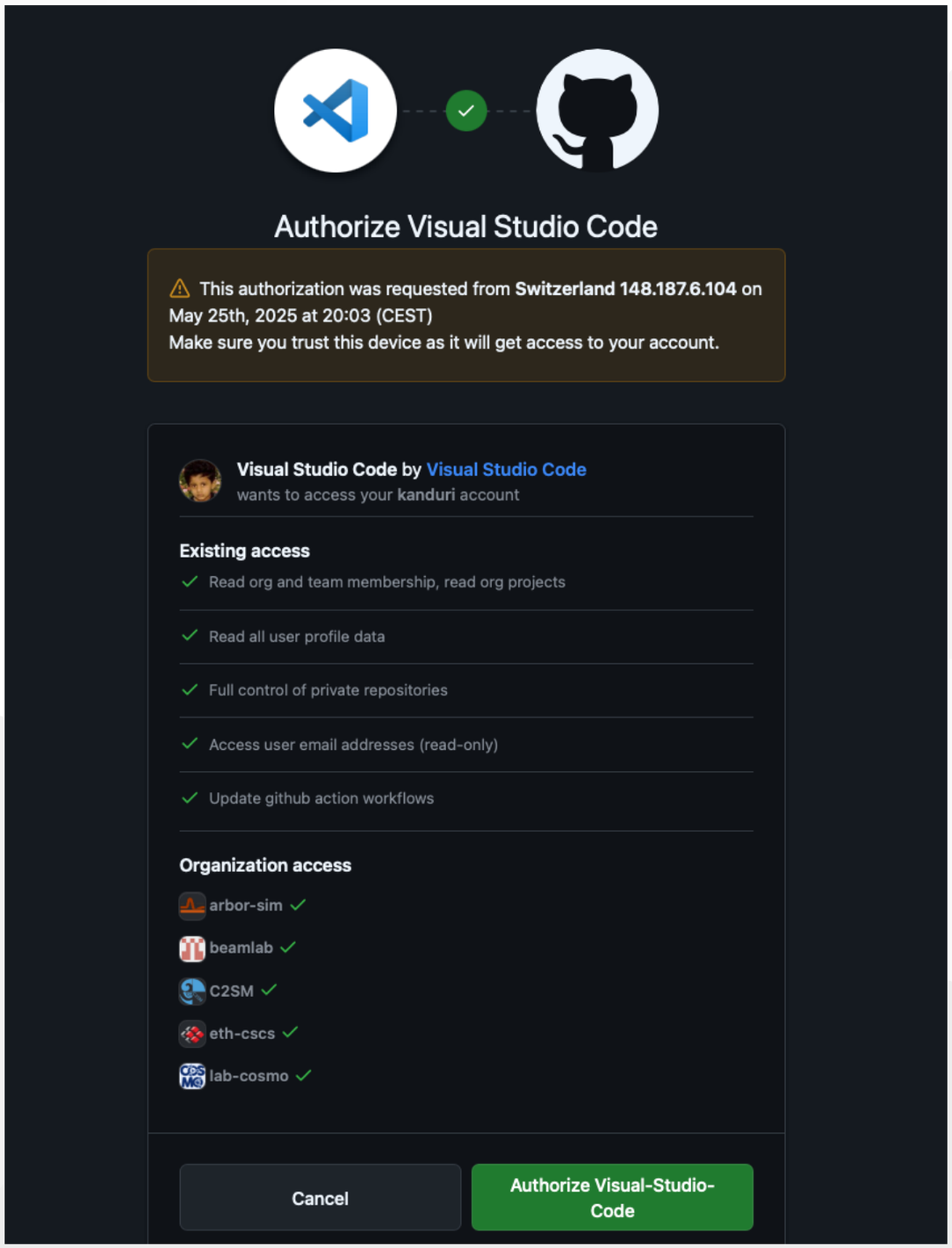
```
> code tunnel --name=$CLUSTER_NAME-tunnel
...
? How would you like to log in to Visual Studio Code? ›
  Microsoft Account
› GitHub Account
```

This will begin the first time setup process. It will ask to open a URL on the browser.

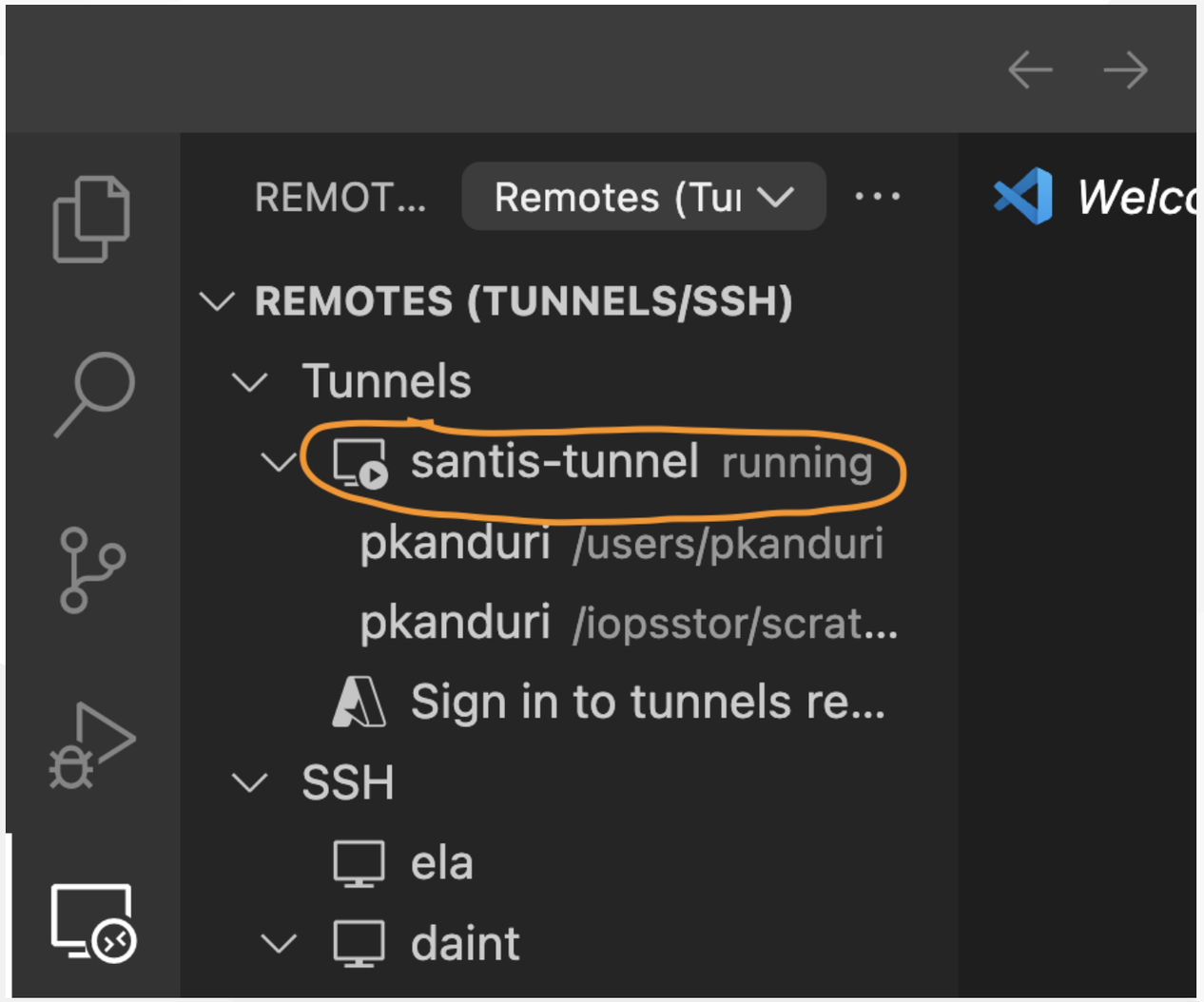
First-Time Setup



First-Time Setup cont...



Access from VS Code



Interactively Connect to a Compute Node

One can log into a compute node and use the same `code tunnel` command.

```
# log into daint
ssh daint

# start an interactive shell session
srun -t 120 -N 1 --pty bash

# set up the environment before starting the tunnel
uenv start prgenv-gnu/24.11:v1 --view=default
code tunnel --name=$CLUSTER_NAME-tunnel
```

- `-t 120` requests a 2 hour (120 minute) reservation
- `-N 1` requests a single rank - only one rank/process is required for VSCode
- `--pty` allows forwarding of terminal I/O, for `bash` to work interactively

Connect VS Code with a Container

This will use CSCS's custom **Container Engine** which can easily pull a container from a registry like DockerHub. Same setup process as earlier with GitHub.

TOML File with Image and Mount Paths

```
image = "nvcr.io#nvidia/pytorch:24.01-py3"
writable = true
mounts = ["/iopsstor/scratch/cscs/username",
      "/users/username/.local/aarch64/bin",
          "/users/username/.bash_history"]
workdir = "/iopsstor/scratch/cscs/username"
```

Launch Container & Tunnel

```
# launch container on compute node
srun -N 1 --environment=/path/to/vscode-pytorch.toml --pty bash
# start tunnel
cd /path/to/code/executable
./code tunnel --name=$CLUSTER_NAME-tunnel
```

Using `code` on Different Architectures

Given the `code` application is architecture specific, and the Home Directory is shared across clusters, we can use a trick to streamline our workflow.

On both `x86` and `ARM` clusters, we can download the respective clusters and store them in specific folders using the `uname -m` command.

```
mkdir -p $HOME/.local/$(uname -m)/bin
cp ./code $HOME/.local/$(uname -m)/bin
```

Then in the `.bashrc` file, we can add the following line which will ensure that the correct executable is available for the architecture.

```
export PATH=$HOME/.local/$(uname -m)/bin:$PATH
```