



CSCS

Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



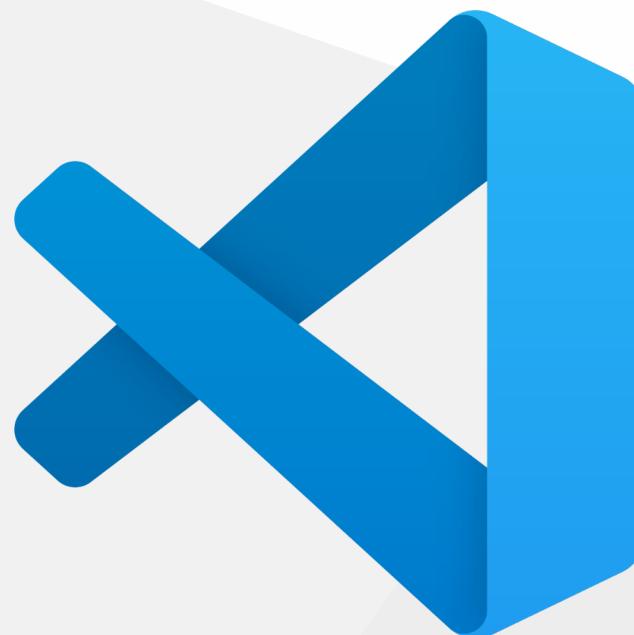
VS Code for Interactive Computing

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!



A screenshot of the Visual Studio Code interface. The left sidebar shows a file tree with a project named 'MARTINS-OBSIDIAN-VAULT-HELPERS'. The right pane displays a Python file named '_main_.py' with the following code:

```
import argparse
import pathlib
from .orphans import report_orphan_attachments, report_notes_not_in_structure
from .link_graph import build_link_graph

def main():
    options = parse_args()
    vault = options.vault

    link_graph = build_link_graph(vault)
    report_orphan_attachments(vault, link_graph)
    report_notes_not_in_structure(vault, link_graph)

def parse_args() -> argparse.Namespace:
    parser = argparse.ArgumentParser()
    parser.add_argument("vault", type=pathlib.Path)
    parser.parse_args()
    return parser.parse_args()

if __name__ == "__main__":
    main()
```

The code completion feature is active, with a tooltip showing various methods and attributes for the 'parser' variable, such as 'add_argument', 'add_argument_group', 'add_help', 'add_mutually_exclusive_group', 'add_subparsers', 'allow_abbrev', 'argument_default', 'conflict_handler', 'convert_arg_line_to_args', 'description', and 'epilog'.

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	Password-Less Login
<pre>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</pre>	<p>→ <code>ssh daint</code></p> <pre>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 =====</pre>

VS Code Setup

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

The screenshot shows the Microsoft Remote - SSH extension page in the Visual Studio Code Marketplace. The extension icon is a blue circle with a white terminal symbol. The title is "Remote - SSH" by Microsoft. It has 27,358,790 installs and a 4.5-star rating. The description says: "Open any folder on a remote machine using SSH and take adva...". Action buttons include "Disable" and "Uninstall" dropdowns, "Switch to Pre-Release Version", and an "Auto Upd" checkbox. Below the header are tabs for "DETAILS", "FEATURES", and "EXTENSION PACK". The main content area is titled "Visual Studio Code Remote - SSH" and describes the extension's purpose: "The **Remote - SSH** extension lets you use any remote machine with a SSH server as your development environment. This can greatly simplify development and troubleshooting in a wide variety of situations. You can:". A bulleted list follows: "• Develop on the same operating system you deploy to or use larger, faster, or more specialized hardware than your local machine." To the right is an "Installation" section with details: Identifier ms-vscode-remote.remote-ssh, Version 0.120.0, Last 2025-05-09, Updated 11:33:00, Size 2.57MB. At the bottom is a "Marketplace" link.

Remote - SSH

Microsoft [microsoft.com](#) | 27,358,790 | ★★★★☆ (1)

Open any folder on a remote machine using SSH and take adva...

Disable | Uninstall | Switch to Pre-Release Version Auto Upd

DETAILS FEATURES EXTENSION PACK

Visual Studio Code Remote - SSH

The **Remote - SSH** extension lets you use any remote machine with a SSH server as your development environment. This can greatly simplify development and troubleshooting in a wide variety of situations. You can:

- Develop on the same operating system you deploy to or use larger, faster, or more specialized hardware than your local machine.

Installation

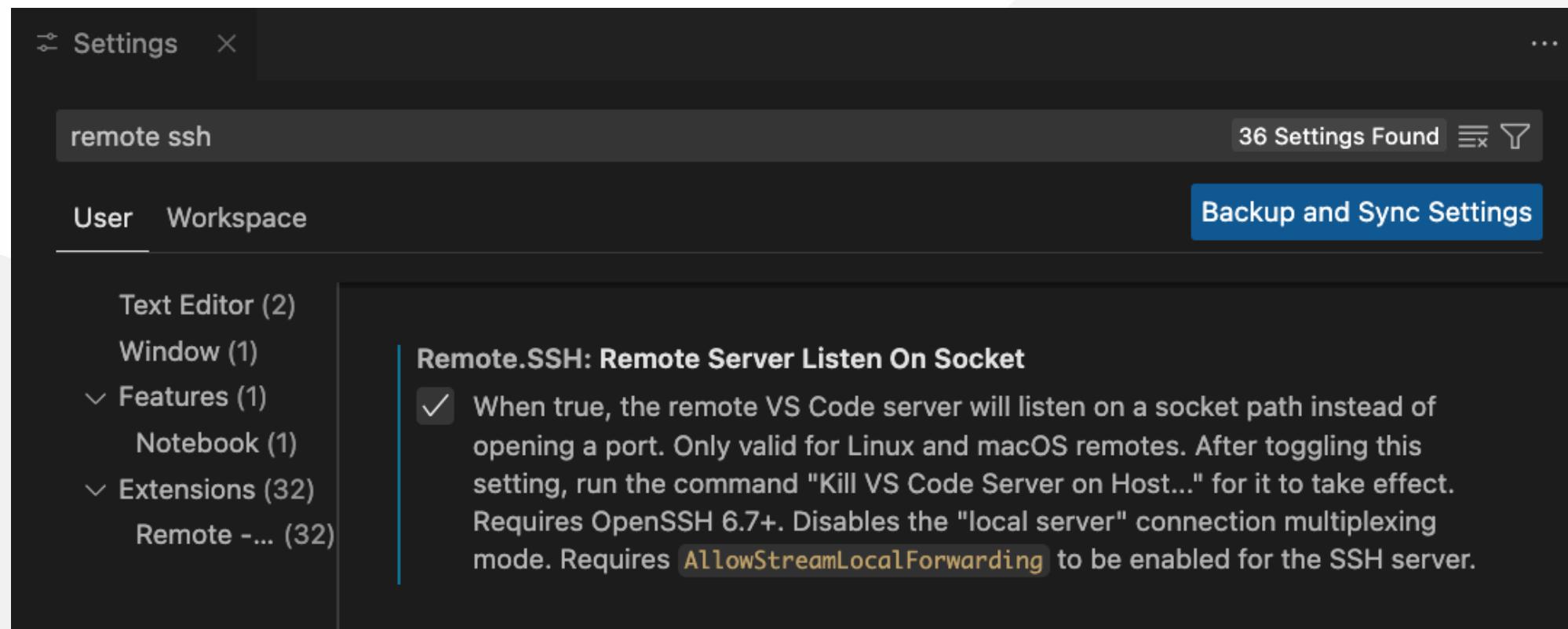
Identifier ms-vscode-remote.remote-ssh
Version 0.120.0
Last 2025-05-09, Updated 11:33:00
Size 2.57MB

[Marketplace](#)

VS Code Setup cont...

After installation, in the settings, scroll down to the extensions and find the section concerning the one above.

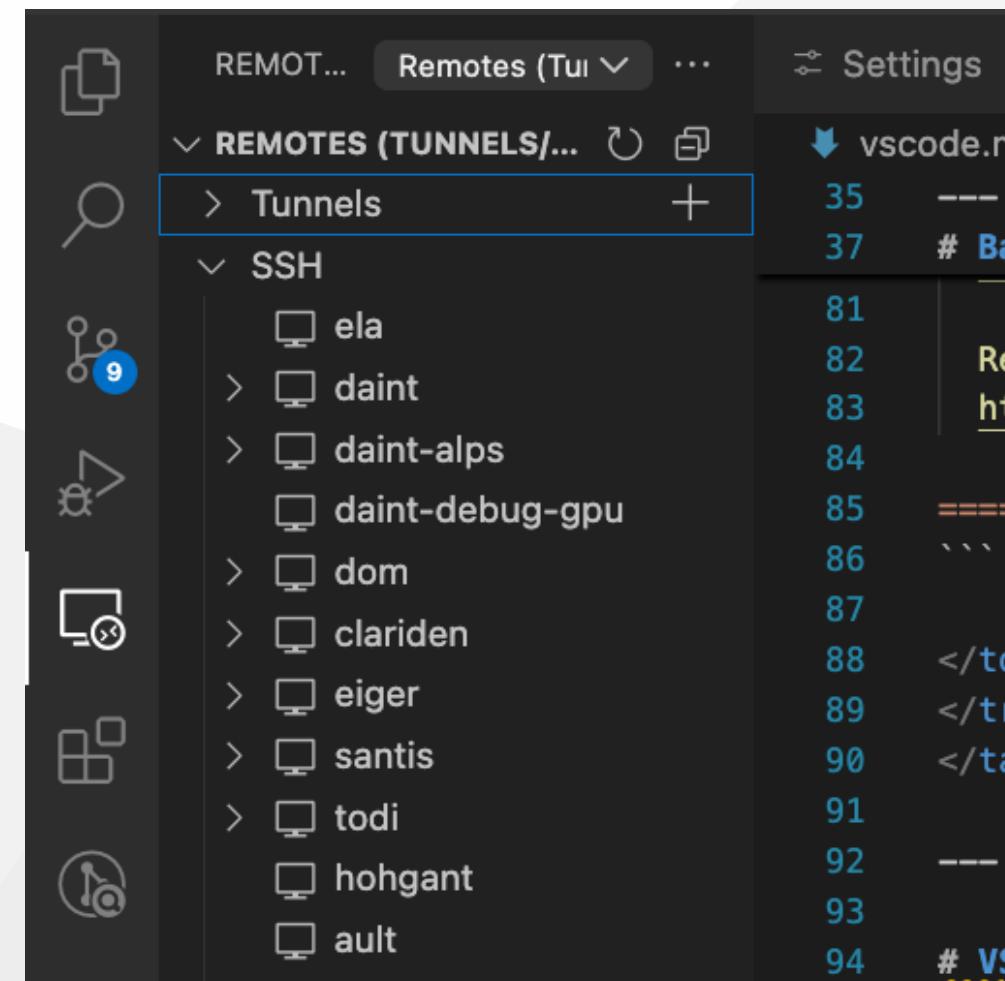
There activate the **Remote Server Listen On Socket** setting.



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.

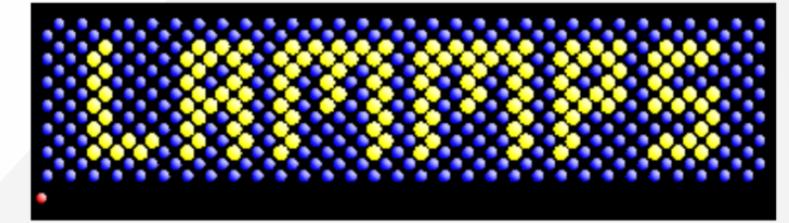
Select the cluster and the window will reload to login. One can select a folder to open there.



Loading Development Environments



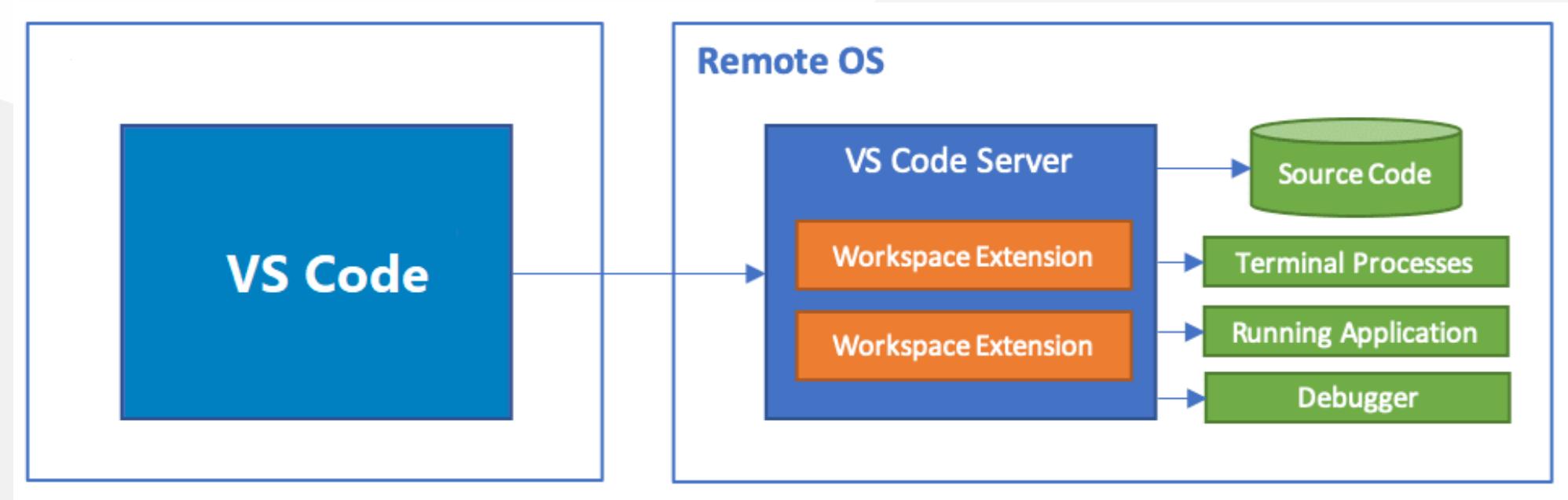
NVIDIA.
CUDA®



Working with Remote Tunnels

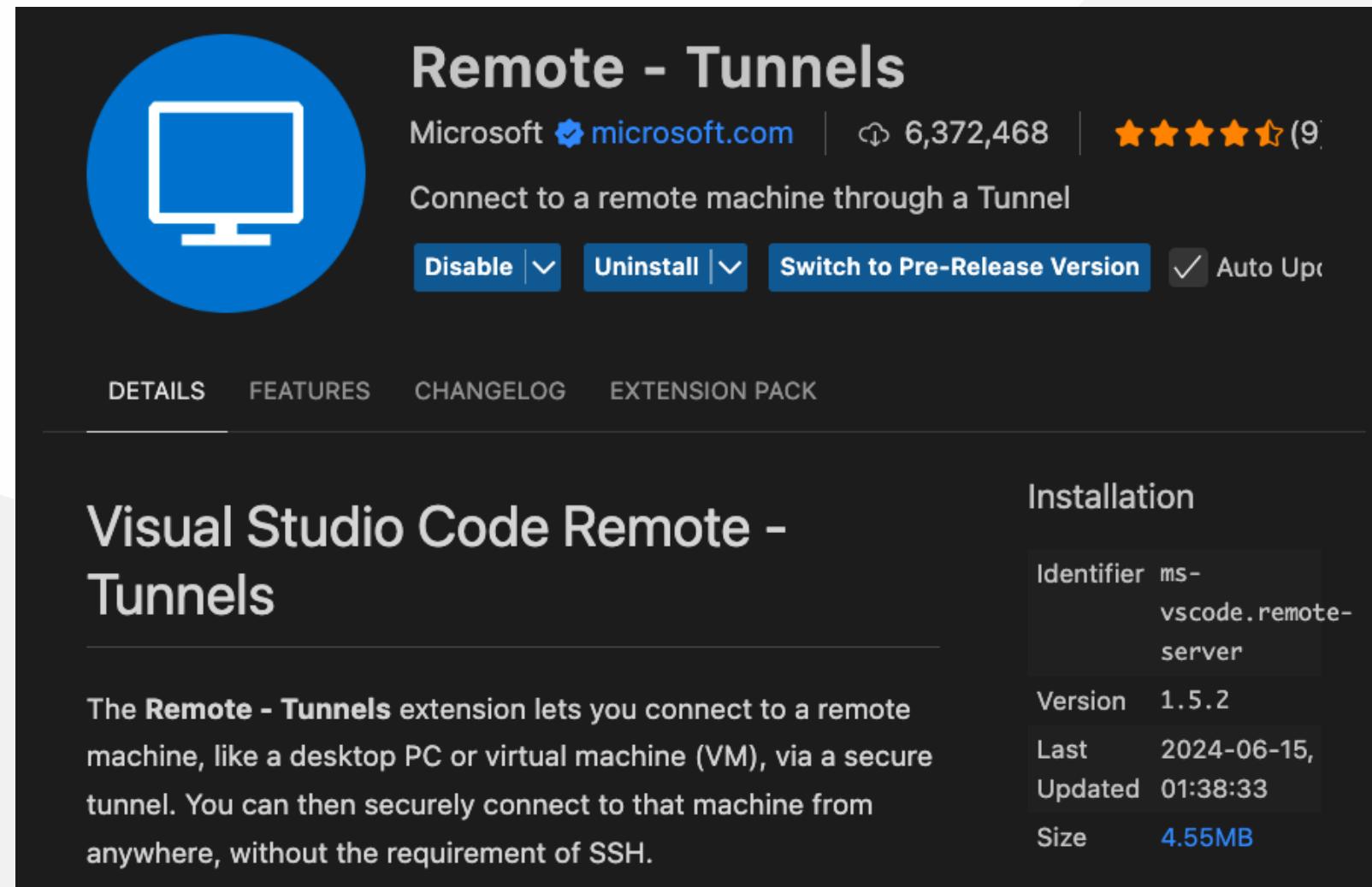
“ The Visual Studio Code **Remote - Tunnels** extension lets you connect to a remote machine, like a desktop PC or virtual machine (VM), via a secure tunnel. You can connect to that machine from a VS Code client anywhere, without the requirement of SSH.

”



Setting up Locally

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



The screenshot shows the Microsoft Marketplace page for the "Remote - Tunnels" extension. The extension icon is a blue circle containing a white computer monitor. The title "Remote - Tunnels" is displayed in large white font. Below it, the developer "Microsoft" and the URL "microsoft.com" are shown, along with the download count "6,372,468" and a 5-star rating. A descriptive text states "Connect to a remote machine through a Tunnel". Action buttons include "Disable", "Uninstall", "Switch to Pre-Release Version", and "Auto Update". Below the main header, there are navigation links for "DETAILS", "FEATURES", "CHANGELOG", and "EXTENSION PACK". The "DETAILS" tab is selected. The main content area is titled "Visual Studio Code Remote - Tunnels" and describes the extension's purpose: connecting to remote machines via a secure tunnel without SSH. To the right, under the heading "Installation", are the following details:

Identifier	ms-vscode.remote-server
Version	1.5.2
Last	2024-06-15,
Updated	01:38:33
Size	4.55MB

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.

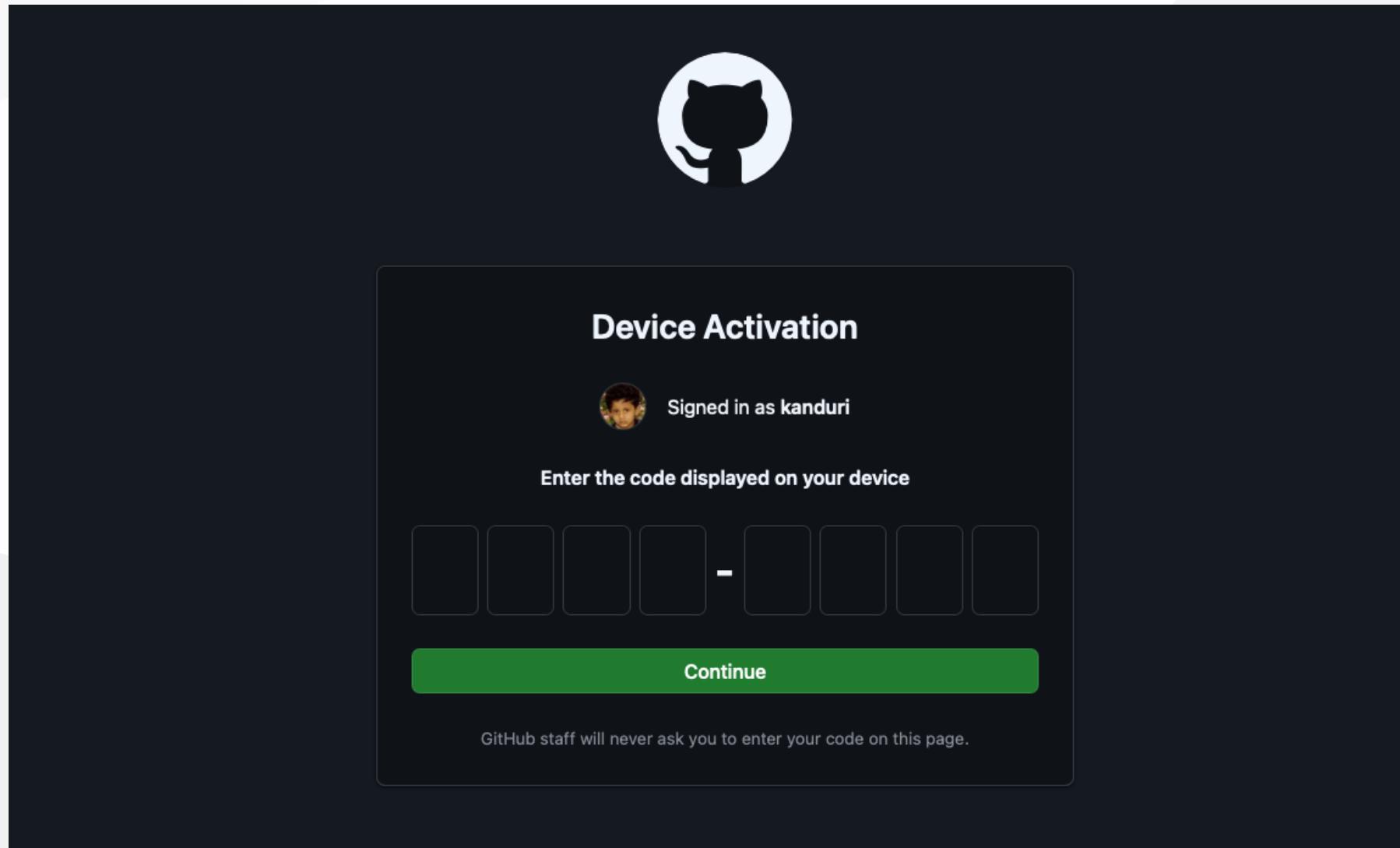
```
# start a uenv session on the login node
uenv start --view=default prgenv-gnu/24.11:v1
# then start the tunnel
code tunnel --name=$CLUSTER_NAME-tunnel
```

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

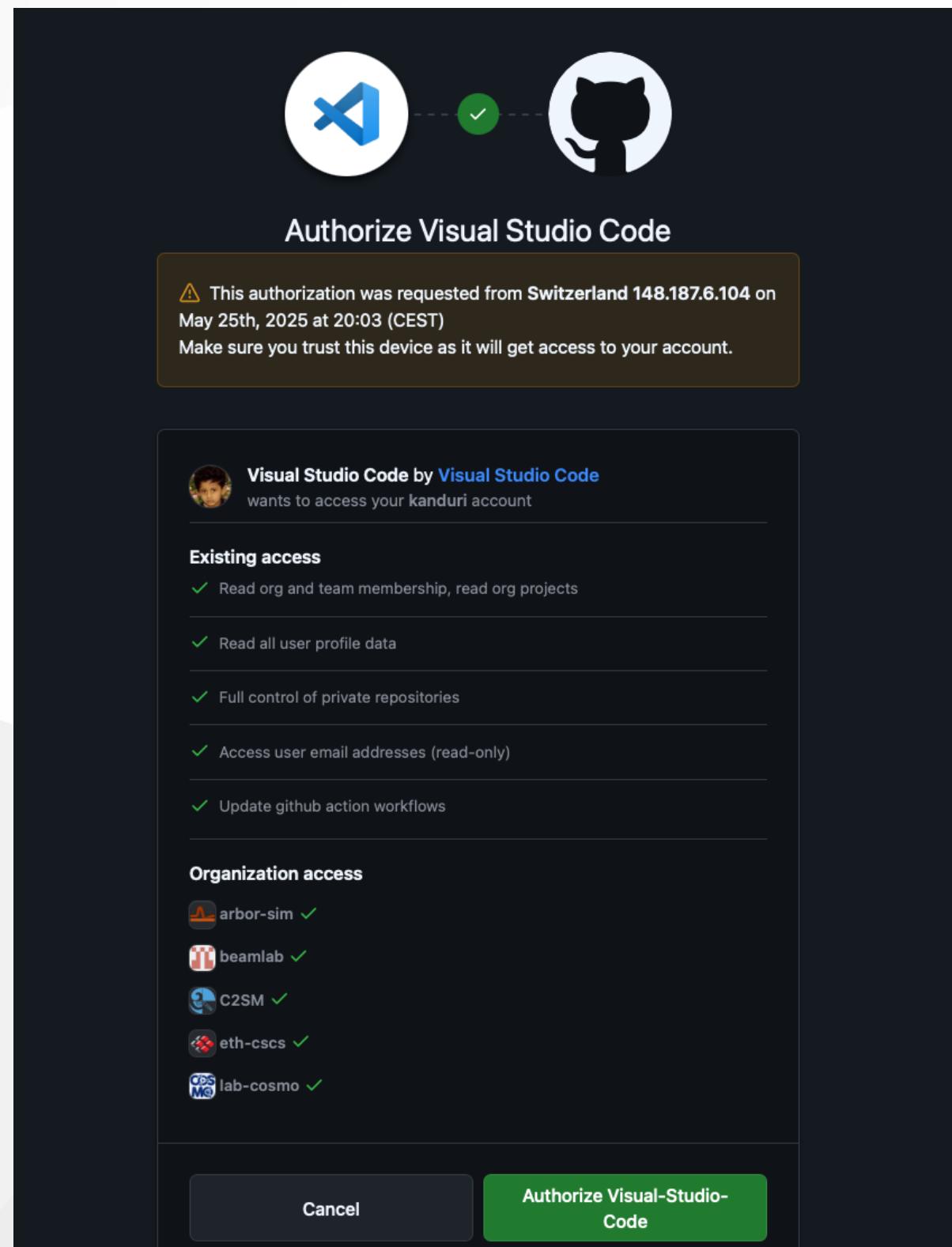
```
> code tunnel --name=$CLUSTER_NAME-tunnel
...
? How would you like to log in to Visual Studio Code? ›
  Microsoft Account
> GitHub Account
```

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

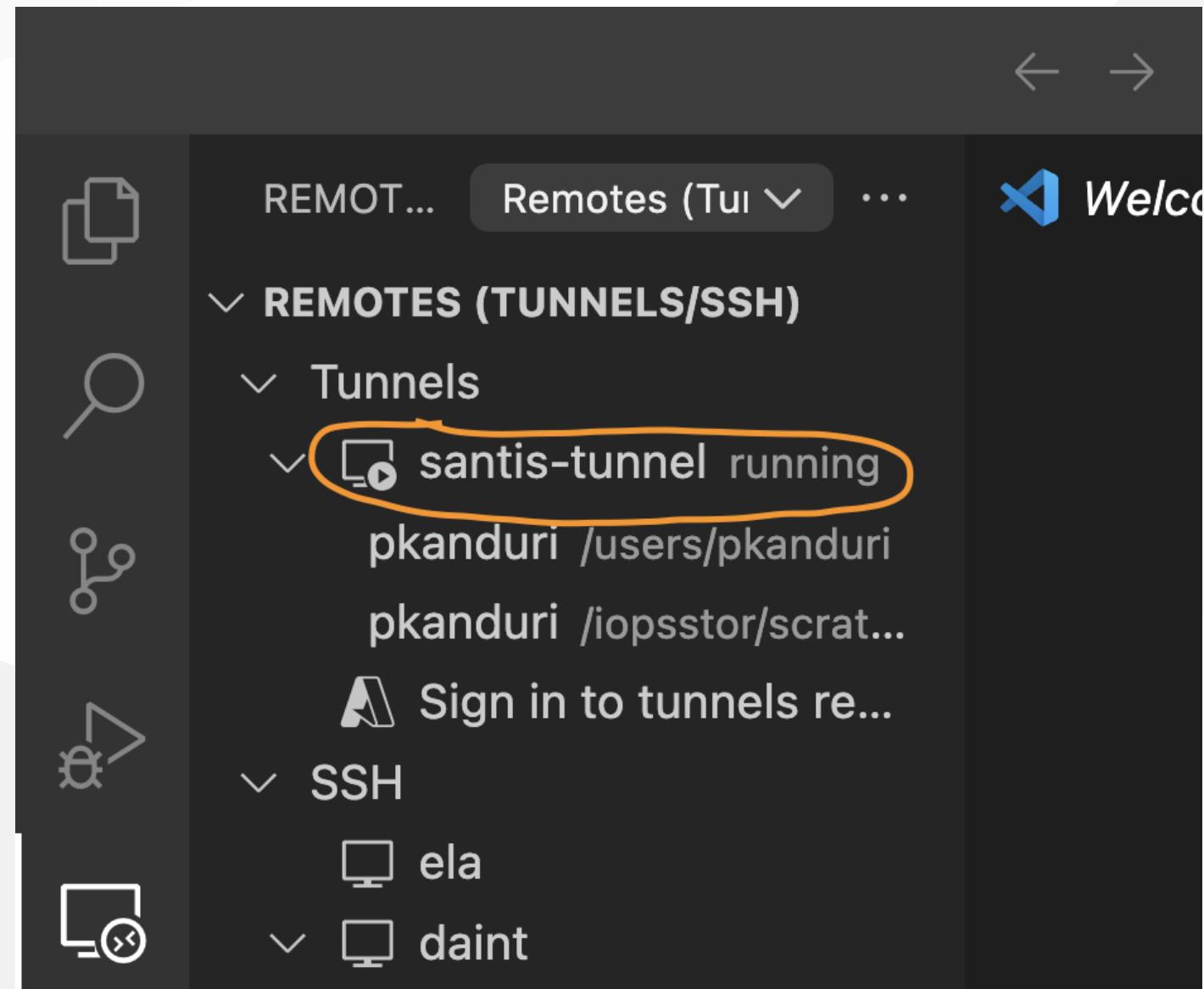
First-Time Setup



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
```