SSH from within Visual Studio Code

The Visual Studio Code integrated development environment (IDE) makes it possible to edit program code locally, and compile and execute it remotely, through a SSH connection.

This is a very useful and powerful feature, but not everything is possible, or easy.

Simple alternative: run VScode on the same system as your program

Perhaps strange to suggest on a page about the use of ssh in vscode, to first suggest not to use ssh. But if possible, a local setup is much easier, as you may realize after reading (and trying) the other setups.

All our desktops, including the vdesk cluster, have a recent version of vscode installed. You can find it in the menu, or execute the command `code` from the terminal. Starting from the terminal has the big advantage that you can setup an environment first, and then everyting you compile or run inside vscode will be using that environment. Example:

```
module load AMUSE/2023.5.1
code
```

this will run the vscode program with the AMUSE environment loaded, so you can directly execute your AMUSE scripts from vscode.

The ssh plugin

Instructions for installing the vscode ssh plugin can be found here.

Some common pitfalls:

- Setup ssh to go directly to your target host. Sterrewacht desktops are directly reachable over the internet, no setup needed. But if you want to run on a Sterrewacht compute node, the ALICE cluster or a Institute Lorentz machine, you will need a proxy setup to go through the appropriate gateway. See SSH tips and tricks, especially example 3 at the bottom.
- In the context of vscode, setting up a Proxy is NOT identical to logging in on the gateway, then loggin in to your target; if you configure vscode to go to the gateway only, vscode's file browser will be running on the gateway, and vscodes internal server code will be running there too. And since most ssh gateways are just gateways, and not powerful compute nodes, this will be very limiting. So do setup that proxy config!!
- vscode will automatically install some server code on the target to receive and handle your connections. This is conveniently done without any user interaction, but inconveniently, this code ends up in `$HOME/.vscode-server` and space in teh home disk is limited. If this fails, check your quota, move things around, get rid of the incomplete vscode directory and try again.
- What usually works (NOT FULLY TESTED YET): first log in to your target server, create a

directory .vscode-server on a local disk of that system, and make a symbolic link to that location in your home directory, e.g.

```
mkdir /data1/username/.vscode-server
ln -s /data1/username/.vscode-server $HOME

* Creating the .vscode-server directory on a local disk, also avoids the pitfall, that in our institutes, we have computers running different Linux versions (eg desktops running Fedora, older compute nodes running RHEL 7 or 8 and newer ones running RHEL 9 or Rocky 9). And software installed for one of these, might not be compatible with any of the others. And if .vscode-server is in the shared $HOME directory, all operating systems will be using the same instance of this code, and might fail in unpredictable ways.
```