Microsoft Visual Studio Code (vscode) and Remote SSH to Discovery

17 more properties

Introduction

Microsoft Visual Studio Code (aka VS Code aka vscode) is currently the most popular code editor. Many Discovery users use its Remote SSH feature to transparently edit code that resides on the cluster. This has resulted in many users running resource-intensive vscode server processes on the login nodes `login-00` and `login-01`. Many of these processes are caught by CPUBoundBot and the users notified accordingly. Ideally these vscode server processes would execute on compute nodes so as to not overwhelm the login nodes. This document aims to provide a potential solution to this problem and was heavily inspired by this GitHub issue: **Issue #1722**.

Approach

The vscode **Remote SSH** extension requires that OpenSSH be installed and uses its configuration directly (by default `~/.ssh/config`). Additional `Host` specifications can be added to the configuration file which allow for starting a SLURM job on a single compute node and transparently proxying the vscode SSH connection through the login node to the compute node. The SSH client configuration option `ProxyCommand` creates a normal SSH connection to `login.discovery.neu.edu` but instead of executing the user's default shell, the SLURM command `/usr/bin/salloc` is executed instead. The command passed to `salloc(1)` is `/bin/bash`. The command passed to `/bin/bash` 1) places the value of environment variable `SLURM_JOBID` into a file (for later consumption by the command specified in SSH client configuration option `RemoteCommand`), and 2) executes netcat to connect to the allocated compute node (determined via environment variable `SLURM_NODELIST`) on SSH default port 22. The SSH client configuration option `RemoteCommand` specifies SLURM command `srun(1)` with option `--jobid` (whose value is obtained from the file generated in `ProxyCommand`) and command `/bin/bash`. The `--jobid` option is critical here as it ensures that the resulting shell and vscode server processes are executed using the SLURM job resource allocation.

Prerequisites

- You have a recent version of vscode installed along with the **Remote Development** and **Remote SSH** extensions (these were already installed out-of-the-box for the author).
- You can SSH from your local system to `login.discovery.neu.edu` non-interactively via SSH public key authentication.
- You can SSH from `login.discovery.neu.edu` to any compute node non-interactively via SSH public key authentication.

Edit SSH client configuration file

Edit file `~/.ssh/config` and add the following to it (note that `User` must be replaced with your own Discovery cluster username):

```
Host login UserKnownHostsFile /dev/null StrictHostKeyChecking no LogLevel
ERROR HostName login.discovery.neu.edu # substitute your username here User
d.hummel Host vscode UserKnownHostsFile /dev/null StrictHostKeyChecking no
LogLevel ERROR # substitute your username here User d.hummel RequestTTY yes #
value of --immediate should match that of connection timeout in # remote ssh
extension settings ProxyCommand ssh -q login "/usr/bin/salloc --immediate=180
--job-name=vscode \ --quiet /bin/bash -c 'echo \$SLURM_JOBID > ~/vscode-job-
id; \ nc \$SLURM_NODELIST 22'" RemoteCommand srun --jobid $(cat ~/vscode-job-
id) --pty /bin/bash
```

(NOTE: This configuration is perhaps generic enough to be generally useful and doesn't contain anything vscode-specific except for the `Host` being named `vscode`.)

Configure Remote SSH extension

The following settings should be changed:

- **Config File**: `~/.ssh/config`
- **Connect Timeout (seconds)**: 180 . Because the SSH configuration starts an interactive SLURM job via `salloc(1)`, it may take some time for the resource allocation to be created depending on how busy the cluster is. Setting this timeout to 3 minutes should generally result in a successful connection even when the cluster is busy. The value specified here should match the value passed to the `--immediate` option of `salloc(1)` in the SSH client configuration file.
- **Enable Remote Command**: (checked) . This is an experimental feature that should be present in recent vscode versions and seems to work reliably.

Test the SSH client configuration

Execute the following commands and you should end up with an interactive shell on a login node and a compute node:

```
ssh login ssh vscode
```

Use the Remote SSH custom configuration

- Click on the **Remote Explorer** icon in the far left column. You should be presented with a list of hosts that correspond to the `Host` directives in the SSH client configuration file.
- Right-click on host `vscode` and select either **Connect to Host in Current Window** or **Connect to Host in New Window**.
- You should see a connection progress pop-up in the lower-right corner. Click the `details` link to view progress and connection details.
- Check your SLURM job status with the usual commands.
- Browse files and use the editor as normal.

Caveats

- The options passed to `salloc(1)` should be revisited to specify resource requirements appropriately. For example, in order to access a GPU node on the cluster, one need to pass `–partition=gpu –gres=gpu:1` to `salloc(1)` . For example:

```
ProxyCommand ssh -q login "/usr/bin/salloc --partition=gpu --nodes=1 --
ntasks=1 --gres=gpu:1 --mem=64Gb --immediate=360 --job-name=vscode-gpu --
quiet /bin/bash -c 'echo \$SLURM_JOBID > ~/vscode-gpu-job-id; nc
\$SLURM_NODELIST 22'”
```

Screenshot 2023-01-23 at 12.26.25 PM.png 144.0KB

- The handling of the file containing the SLURM job ID could probably be better.
- This requires the user to actually implement the configuration changes and stick to using them. It would be better if we could do this transparently on the user's behalf.