WSL/ssh/git commands

Donghun Jung

Here are some pros and cons of using an SSH server.

Advantage

1. **Secure**: SSH uses encryption to protect the data being transmitted between the client and server, making it a secure way to remotely access and manage a server.
2. **Flexible**: SSH can be used for a wide range of tasks, from running commands on a remote server to transferring files securely.
3. **Remote access**: SSH allows you to remotely access and manage a server from anywhere with an internet connection, making it convenient for administrators and developers who need to manage servers remotely.
4. **Cost-effective**: Using an SSH server can be more cost-effective than using a cloud-based server or dedicated hosting service, especially for small businesses or organizations.
5. **Scalable**: An SSH server can support multiple simultaneous connections, allowing multiple users to connect and use the server resources at the same time.
6. **Easy file sharing**: SSH can be used to securely transfer files(especially, code files) between devices, making it an ideal tool for remote teams or users who need to collaborate on a project or share files across different devices and locations.

Disadvantage

1. **Technical knowledge required**: Using an SSH server requires some technical knowledge, including how to install and configure the server and how to use the SSH client to connect to the server.
2. **Security risks**: If not configured correctly, an SSH server can be vulnerable to security threats such as brute-force attacks, unauthorized access, and malware.
3. **Resource-intensive**: Running an SSH server can require significant resources, such as memory and processing power, especially for high-traffic servers or servers that handle large amounts of data.
4. **Loss of foreground programs**: If running a program in the foreground, a lost SSH connection can result in the program being terminated, potentially leading to data loss or unsaved changes.
5. **Resource-intensive**: Running an SSH server can require significant resources, such as memory and processing power, especially for high-traffic servers or servers that handle large amounts of data. This can be further impacted when multiple users are connected simultaneously.

WSL

Essential commands in wsl

ls : This command is used to list the files and directories in the current working directory.

1. `ls -a`: This will list all files and directories in the current directory, including hidden files (files that start with a dot).
2. `ls -lh`: This will list the files and directories in the current directory in a long format with human-readable file sizes.
3. `ls -R`: This will list all the files and directories in the current directory and its subdirectories recursively.

cd : This command is used to change the current working directory.

1. `cd ~`: This will change the current working directory to the user's home directory.
2. `cd ..`: This will change the current working directory to the parent directory of the current directory.

rm : This command is used to remove files or directories.

1. `rm -f file.txt`: This remove the file named 'file.txt' without prompting for confirmation.
2. `rm -r -f directory`: This remove the directory named 'directory' and all its contents recursively without prompting for confirmation.
3. `rm -i file.txt`: This prompt for confirmation before removing the file named 'file.txt'.

cp : This command is used to copy files or directories.

1. `cp -r -v directory directory_copy`: This will copy the 'directory' and all its contents recursively to a new directory named 'directory_copy' while displaying the name of each file as it is copied.
2. `cp -p file.txt file_copy.txt`: This will make a copy of 'file.txt' named 'file_copy.txt' while preserving the original file's permissions and timestamp.
3. `cp -u file.txt directory`: This will copy 'file.txt' to 'directory' only if the file in the destination directory is older than the source file.

mkdir : This command is used to create a new directory.

1. `mkdir -p parent_directory/new_directory`: This will create a new directory named 'new_directory' inside 'parent_directory', even if 'parent_directory' does not exist.

1. `touch file.txt`: This will create an empty file named 'file.txt' if it does not already exist, or update the timestamp of the file if it does exist.

grep : This command is used to search for a pattern in a file or output.

1. `grep pattern file.txt`: This will search for the pattern in the file named 'file.txt'.
2. `command | grep pattern`: This will search for the pattern in the output of the command.

find : This command is used to search for files or directories that meet certain criteria.

1. `find directory -name '*.txt'`: This will search for all files in the 'directory' directory and its subdirectories that have a '.txt' extension.
2. `find / -type f -name 'file.txt' 2>/dev/null`: This will search for all files named 'file.txt' on the entire system and hide any error messages.

chmod : This command is used to change the permissions of a file or directory.

1. `chmod u+x file.txt`: This will add execute permission for the file owner on 'file.txt'.
2. `chmod 644 file.txt`: This will set the file permissions of 'file.txt' to read-write for the owner and read-only for everyone else.

sudo : This command is used to execute a command with elevated privileges. Note that it is not available for normal user.

1. `sudo apt-get update`: This will update the package list using the Advanced Packaging Tool (APT) with elevated privileges.
2. `sudo su`: This will switch the current user to the root user, giving you access to all files and directories on the system.

Tips when using WSL

1. **Use the Tab key for auto-completion**: When typing a file or directory name, you can press the Tab key to automatically complete the name for you. If there are multiple files or directories that match the name you are typing, pressing Tab multiple times will cycle through them.
2. **Use Ctrl+Shift+C and Ctrl+Shift+V for copy and paste**: You can use these keyboard shortcuts to copy and paste text between WSL and other applications in Windows.
3. **Use the WSL command to open a new command prompt**: You can open a new command prompt in the current directory by typing 'wsl' and pressing Enter.
4. **Use the 'cd' command to change directories**: You can use the 'cd' command to change directories in WSL. For example, 'cd /mnt/c/Users' will change the current directory to the 'Users' directory on the C drive.
5. **Use 'ls' to list files and directories**: Use the 'ls' command to list the files and directories in the current directory.
6. **Use 'cp' to copy files and directories**: Use the 'cp' command to copy files and directories. For example, 'cp file.txt directory/' will copy the file 'file.txt' to the 'directory' directory.
7. **Use 'mv' to move or rename files and directories**: Use the 'mv' command to move or rename files and directories. For example, 'mv file.txt directory/file2.txt' will move the file 'file.txt' to the 'directory' directory and rename it to 'file2.txt'.
8. **Use 'rm' to remove files and directories**: Use the 'rm' command to remove files and directories. For example, 'rm file.txt' will remove the file 'file.txt'.
9. **Use 'sudo' for elevated privileges**: Use the 'sudo' command to execute a command with elevated privileges. For example, 'sudo apt-get update' will update the package list using the Advanced Packaging Tool (APT) with elevated privileges.
10. **Use 'alias' to create shortcuts for commands**: Use the 'alias' command to create shortcuts for commands. For example, "alias ll='ls -lh'" will create a shortcut 'll' that will list the files and directories in the current directory in a long format with human-readable file sizes when typed.

SSH

Here are some general steps you can follow:

1. **Install WSL**: If you're using Windows, you'll need to install WSL first. You can do this by following the instructions on the official Microsoft documentation:

   `https://learn.microsoft.com/en-us/windows/wsl/install`

2. **Install an SSH client**: Once you've installed WSL, you can use any SSH client that is compatible with Linux. For example, you can install OpenSSH by running the following command in your WSL terminal: sudo apt-get install openssh-client. If you install wsl properly, it must be available.

3. **Create account**: Ask your administrator to create an account.

4. **Open your SSH client**: Open your SSH client in the WSL terminal and enter the server's IP address or hostname in the appropriate field as following:

```
        ssh {your_id}@192.168.0.39 -p 2012

```

   And then, enter your password.

   Warning! : Your Wi-fi must be `cqioffice2`.

   Causion! : When entering your password in the WSL terminal, you will not see any characters or feedback on the screen. This means that if you mistype your password, you will not be able to tell if you made a mistake. To avoid errors, type your password slowly and carefully, and double-check that it is correct before pressing Enter.

If you have difficulty using the server, please contact your administrator.

Shortcut using VScode

Many people use VS Code as their primary code editor. Here are the steps to connect to an SSH server using the SSH extension in VS Code:

1. **Install the SSH extension**: Open VS Code and go to the Extensions view by clicking on the Extensions icon in the left sidebar or by pressing Ctrl+Shift+X (Windows/Linux) or Cmd+Shift+X (macOS). Search for "Remote Development" and install the "Remote Development" extension pack, which includes the SSH extension.

2. **Open the Command Palette**: Open the Command Palette by pressing Ctrl+Shift+P (Windows/Linux) or Cmd+Shift+P (macOS).

3. **Select "Connect to Host"**: In the Command Palette, type "ssh" and select "Remote-SSH: Connect to Host".

4. **Open your SSH client**: Open your SSH client in the WSL terminal and enter the server's IP address or hostname in the appropriate field as following:

```
        ssh {your_id}@192.168.0.39 -p 2012

```

   Warning! : Your Wi-fi must be `cqioffice2`.

Git

Git is a distributed version control system, which means that each user has a complete copy of the codebase, along with its entire history of changes. This allows developers to work offline and make changes to the codebase without needing to be connected to a central server. Notice that git is available in the server. Here are pros and cons.

Advantage

1. **Distributed architecture**: Git's distributed architecture allows each developer to have their own local copy of the repository. This allows for more flexibility in workflow and enables developers to work offline without requiring a constant connection to a central server.
2. **Branching and merging**: Git's branching and merging capabilities allow developers to work on multiple features or bug fixes simultaneously and merge them back into the main codebase. This enables more efficient collaboration and reduces the likelihood of conflicts.
3. **Version control**: Git's version control system allows developers to track changes to the codebase over time. This provides a historical record of the project and makes it easier to revert to previous versions of the code.
4. **Open-source and widely adopted**: Git is open-source software and is widely adopted in the software development industry. This means that there is a large community of developers contributing to its development and providing support.
5. **Integration with other tools**: Git integrates with a wide range of other software development tools, such as IDEs, continuous integration systems, and issue trackers. This allows for a streamlined development workflow and makes it easier to manage software projects.
6. **Branching and merging**: Git allows developers to create multiple branches of the codebase, which can be used to work on different features or bug fixes in isolation. This means that developers can work on different parts of the codebase simultaneously without interfering with each other. When the changes are ready, they can be merged back into the main codebase.
7. **Code reviews**: Git makes it easier to perform code reviews, which are a critical part of the development process. Code reviews allow developers to review each other's code for bugs, potential security issues, and best practices. Git provides tools for creating pull requests, which allow developers to propose changes to the codebase and have them reviewed by other developers before they are merged.
8. **Collaborative workflows**: Git enables different types of collaborative workflows, such as centralized workflows, where one central repository is used as the main source of truth, and distributed workflows, where each developer has their own copy of the repository. This flexibility allows teams to choose the workflow that best fits their needs.

Disadvantage

1. **Steep learning curve**: Git has a steep learning curve and can be difficult to understand for new users. This can make it challenging to get started with Git, particularly for developers who are not familiar with version control systems.
2. **Complexity**: Git is a complex system with many commands and options, which can make it overwhelming for some users. This complexity can lead to mistakes and errors if not used correctly.
3. **Command-line interface**: Git's command-line interface may be intimidating for some developers who are used to graphical user interfaces. While there are GUI tools available for Git, many advanced features require the use of the command line.
4. **File size limitations**: Git can struggle with large files, particularly binary files such as images or videos. This can make it difficult to use Git for projects that require the management of large files.

Overall, Git is a powerful tool that offers many advantages for software development projects. However, it does have a steep learning curve and can be complex to use, particularly for new users. Git, there are typically five stages that a file can go through as it moves from being untracked to being committed to the repository. These stages are:

1. `Untracked`: When a file is first added to a repository, it is untracked by Git. This means that Git is not aware of the file's existence and it will not be included in any commits.

2. `Unmodified`: After a file has been added to the repository and committed for the first time, it enters the unmodified stage. This means that the file has not been changed since the last commit.

3. `Modified`: When a file that has previously been committed is changed, it enters the modified stage. This means that the contents of the file have been altered in some way.

4. `Staged`: After making changes to a modified file, the changes must be staged before they can be committed. The staged stage is where Git has been told to include the changes in the next commit.

5. `Committed`: Once changes have been staged, they can be committed to the repository. The committed stage represents a snapshot of the file at a specific point in time.

Essential Git Commands

1. `git add`:This command is used to add changes made to modified files to the staging area. For example, to add all changes made to modified files, you can use git add . which adds all files in the current directory and its subdirectories.

2. `git commit`: This command is used to commit staged changes to the local repository. For example, to commit all staged changes with a commit message, you can use git commit -m "Commit message".

3. `git rm`: This command is used to remove files from the Git repository. For example, to remove a file named example.txt from the repository, you can use git rm example.txt. Note that this command also removes the file from the local file system.

4. `git restore`: This command is used to restore files to a previous state. For example, to restore a file named example.txt to the state it was in the last commit, you can use git restore example.txt. This command restores the file to its previous state in the working directory, but does not affect the staged or committed changes.

5. `git restore --staged`: This command is used to unstage changes made to a file. For example, if you have staged changes made to a file named example.txt using git add, but you decide that you don't want to include those changes in the next commit, you can use git restore –staged example.txt. This command removes the file from the staging area and restores it to the state it was in the last commit.

Tips when using Git

1. **Commit early and often**: It's important to commit changes frequently, rather than waiting until you've completed a large amount of work. This makes it easier to track changes and revert to a previous version if necessary.

2. **Write clear commit messages**: When committing changes, write clear and descriptive commit messages that explain what changes were made and why. This makes it easier for other developers to understand your changes and review your code.

3. **Use branching**: Branching is one of the most powerful features of Git, and it's important to use it effectively. When working on a new feature or bug fix, create a new branch rather than working directly on the main branch. This keeps your changes isolated and reduces the risk of conflicts with other developers.

4. **Resolve merge conflicts**: Merge conflicts can occur when two developers make changes to the same file in different ways. When this happens, Git will prompt you to resolve the conflict manually. It's important to carefully review the changes made by both developers and make a decision about how to merge the changes together. Tools like GitHub's pull request review system can make this process easier by providing a visual diff of the changes.

5. **Use GitHub**: If you're working with a team, GitHub can be a powerful tool for collaboration. It provides tools for pull requests, code reviews, and issue tracking, as well as features like project boards and wikis. GitHub also provides a central repository for storing and sharing code.

6. **.gitignore**: The .gitignore file is used to specify files or directories that should be ignored by Git. This is useful for preventing files like logs, compiled binaries, and other generated files from being accidentally committed to the repository. You can create a .gitignore file in the root directory of your project and list the files or directories that you want to ignore. Git will then automatically exclude those files from the repository.

7. **.editorconfig**: The .editorconfig file is used to define coding styles and preferences for a project. It specifies things like indentation style, line endings, and encoding. By using an .editorconfig file, you can ensure that all developers on the project are using consistent coding styles, regardless of the editor or IDE they are using.

8. **IntelliCode**: IntelliCode is a machine learning tool developed by Microsoft that integrates with Visual Studio Code and other IDEs. It provides intelligent code completions, code suggestions, and other helpful features based on the code you're working with. IntelliCode uses machine learning algorithms to analyze millions of open-source projects and generate intelligent suggestions based on the context of your code. It can help speed up development and reduce errors by suggesting common patterns and best practices. IntelliCode is particularly useful for teams working on large codebases with complex code structures.

What needs to be updated

1. Install and activate 'CUDA'

2. Use a terminal multiplexer such as 'tmux' or Use a job scheduler such as 'Slurm' or 'PBS'

3. Calculation-specific servers