# SSH and Slack Basics

Rebecca Hsieh, Hailey Hullin, Zachary McGrew

May 2, 2019

# Contents

# Chapter 1

# SSH

## 1.1 Why SSH?

The **S**ecure **SH**ell (SSH) is a safe and convenient way of working remotely. In addition to being a shell where you can enter commands, it also provides utilities for transferring files and tunneling data.

## 1.2 SSH

### 1.2.1 Basic SSH Connection

Starting ssh session is easy! Simply run the following, replacing "username" with your username.

```
ssh -p 922 username@linux.cs.wwu.edu
```

This will connect to the Linux pool that is managed by CS Support. Let's break down what each part of this command and arguments do:

`ssh`: This is the actual command to run! Following it are the arguments.
`-p 922`: This means connect to port 922 on the remote host.
`username@`: This tells SSH to use your username when connecting to the the remote host.
`linux.cs.wwu.edu`: This tells SSH which remote host to connect to.

*Note:* `linux.cs.wwu.edu` is a round-robin DNS result that balances the load of incoming connections to various Linux servers on campus. You can also directly access a machine by providing it's full name. CS Support has provided us with `linux-01` through `linux-12`. An example of the full host name is `linux-01.cs.wwu.edu`.

Once you're connected to a remote machine you can enter commands as if you were sitting directly at the console. This means you can run Emacs or Vim and do your homework remotely!

## 1.3 SCP

SCP is the **S**ecure **C**o**P**y command. It allows you to both upload and download files from a remote computer securely over a network.

### 1.3.1 Local to Remote Copy (Uploading)

To upload a file from your local computer to the file system at school:

```
scp -P 922 ~/my_cool_file.txt username@linux.cs.wwu.edu:~/
```

Let's break this down into the command arguments and examine their meaning.

`scp`: This is the actual command to run! Following it are the arguments.
`-P 922`: This means connection port 922 on the remote host.
`~/my_cool_file.txt`: This is the path and file name of the local file to upload.
`username@linux.cs.wwu.edu`: This tells SSH which remote host to connect to.
`:/home/username/`: This is the path to put the file on the remote host.

Let's look a few more examples: What about copying a whole folder and all of it's files? For this we need the `-r` flag!

```
scp -P 922 -r ~/project_folder username@linux.cs.wwu.edu:~/
```

This copies the folder "my_cs_project_folder" and all the contents (including other folders) to the remote machine! Be careful, this can take quite a long time if you have a lot of big files to copy.

### 1.3.2 Remote to Local Copy (Downloading)

Often times you want to get a local copy of a remote file. This can be achieved by simply reversing the two paths used for uploading.

```
scp -P 922 username@linux.cs.wwu.edu:~/my_cool_file.txt ~/
```

This will copy a file by the name of "my_cool_file.txt" on the remote machine to your home folder on the local machine.

## 1.4 SFTP

SFTP or the **S**ecure **F**ile **T**ransfer **P**rotocol uses the same commands and interface as the FTP program. The only difference is that it allows for secure authentication and transfer of files. SFTP allows faster transmission of files compared to SCP, because it doesn't have the same overhead in re-encoding the files before transferring them.

### 1.4.1 Basic SFTP

To connect to an SFTP server at school we can use the following command:

```
sftp -P 922 username@linux.cs.wwu.edu
```

Once connected we can execute the same commands the FTP program uses. Some basic commands include:
`ls`: Get a listing of the files in the current folder.
`get`: Copy a file from the remote host to the local one.
`put`: Copy a file from the local host to the remote one.
`bye` or `quit`: Disconnect from the server.
Here's an example session using SFTP.

```
sftp> ls
```

```
dir1 dir2 file1 file2 my_cool_file.txt
```

```
sftp> get file1
```

```
Fetching /home/username/path/file1 to file1
```

```
sftp> put file3
```

```
Uploading file3 to /home/username/path/file3
```

```
sftp> ls
```

```
dir1 dir2 file1 file2 file3 my_cool_file.txt
```

In the above example we listed the folder contents, downloaded `file1`, uploaded `file3`, before finally checking the results again with `ls`.

There are many more commands that can be covered in this quick overview. Take a look at the SFTP manual page to see a complete list of commands and all their options.

```
$ man 1 sftp
```

## 1.5 Configuring Options

Have you noticed the `"-P 922"`, `"-p 922"`, or constantly specifying the username and full hostname? These options and many more can be set inside the configuration file located in `~/.ssh/config`.

Open your `~/.ssh/config` in your favorite text editor. It's OK if it doesn't exist because you can just save there and create one.

### 1.5.1 Default Host Rules

We begin with setting the default host rules. These rules will be applied to all hosts when we connect to them, whether we specified the host or not in our configuration file. The rules we will apply to all hosts are pretty generic, but really helpful.

```
Host *
  ServerAliveInterval 60
  ControlMaster auto
  ControlPath ~/.ssh/master-%r@%h:%p
  ControlPersist 10m
```

Let's examine these rules we're using here:

ServerAliveInterval 60

This rule says that every 60 seconds we tell the server that we're still here. This helps keep the connection alive when we're not super active and in NATed environments that drop connections that go unused for too long.

ControlMaster auto

Enabling control master sockets lets us reuse one connection again and again. This options enables them and reuses them when available. To explain why this is beneficial think of opening a terminal and SSHing into the Linux pool. Then opening another terminal and doing the same thing. Instead of connecting and authenticating twice we only have to do it once!

ControlPath ~/.ssh/master-%r@%h:%p

In order for our sockets to be reused we need a place to save them. This rules saves them inside your .ssh folder; hidden away, but still easily accessible. We use the expansion variables to uniquely identify the connection.

ControlPersist 10m

If we close all the terminals running SSH this keeps our connection alive for ten minutes afterwards. Have you ever accidentally closed your ssh terminal and didn't mean to? Now you can easily reconnect!

### 1.5.2 Specific Hosts

Often times we want to set particular rules depending on the host. For example our WWU CS Linux pool uses the non-standard port of 922, and we have to keep specifying it each time we connect.

```
Host linux*
  HostName %h.cs.wwu.edu
  Port 922
  User username
  ForwardX11 yes
```

As we did previously with the default host rules, lets examine what each of these lines does for us.

Host linux*

This line allows us to use wildcard expansion on the hostnames that start with "linux". So we can use "linux" or "linux-03" or "linux-12". All will match this host rule.

HostName %h.cs.wwu.edu

Using the hostname matched from the line above, combine it with the suffix of our CS domain. Instead of typing "linux.cs.wwu.edu" everytime we just type "linux" now. Better yet, since the wildcard expansion was enabled above we can type "linux-04" to connect to "linux-04.cs.wwu.edu" —Pretty neat, huh?

Port 922

No longer do we need to tell the utility the port number! No more "`-p 922`" for ssh and no more "`-P 922`" for scp/sftp.

User username

You'll need to put your username here instead of "username," but once you've done that you no longer need to specify the "username@" argument to connect.

ForwardX11 yes

This allows us to run graphical programs across a secure tunnel. So we can run XWindows based programs remotely. For run try running a graphical text editor and see how it works!

### 1.5.3 Example Config File

The following an easy to use copy-and-paste version of the configuration described above. Feel free to use it as a starting point to making your ssh config. Don't forget to change "username" to you're real username!

```
Host *
  ServerAliveInterval 60
  ControlMaster auto
  ControlPath ~/.ssh/master-%r@%h:%p
  ControlPersist 10m

Host linux*
  HostName %h.cs.wwu.edu
  Port 922
  User username
  ForwardX11 yes
```

## 1.6 Keys Instead of Passwords

Passwords are great, but they can be a pain to type in every time you need to connect somewhere. This is especially true when you use SSH as a secure tunnel and not just a shell. Version control systems such as CVS and Git use SSH to authenticate to a remote machine and then pass data over this secure connection. The problem is each time you run a command with these tools, they have to reconnect, causing you to be prompted for your password each time.

### 1.6.1 Basic Idea

An SSH key is a way to authenticate without a password. In addition to being really cool, they're also more secure! The quick overview is that you need to create a key pair: a private key and a public key. We'll put the public key on the remote host in our `~/.ssh/authorized_hosts` file, and keep the private key locally. When we connect to the server next time it will attempt to use the key for authentication instead of our password! Of course, there's still a password needed to unlock the key, but we can unlock our key and give it to the SSH-Agent for safe keeping until we want to use it. Let's examine the process of making and using keys.

### 1.6.2 Generating Keys

To generate a new SSH key you only need to run the following command:

```
ssh-keygen -b 4096 -t rsa
```

Here's an example of what ssh-keygen will ask you and display.

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/username/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/username/.ssh/id_rsa.
Your public key has been saved in /home/username/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:Sswg+uiETiycokuCXnD5z7qV63lab/PLzgzMrFutC/M username@cf405-13
The key's randomart image is:
+---[RSA 4096]----+
|                 |
|                 |
|  . .            |
| . ..+           |
|.. o  + S        |
|=o+ .. .. + .    |
|BB.. ..o + * .   |
|% .   + +.Bo*    |
|o=   o+BoooE=B.  |
+----[SHA256]-----+
```

Don't worry too much about the image. It's a feature that is supposed to help you visualize your keys and to verify when a key has been altered. I don't really know anybody that uses it - seriously.

*Note:* It's important to set a password for your key! If someone can access your private key and it's not password protected they can log in as you!

### 1.6.3 SSH-Copy-Id

Now that we have a key generated, we need to install the public key in the `~/.ssh/authorized_keys` so that we can authenticate with it. Thankfully the OpenSSH developers have created `ssh-copy-id` to help with this process.

```
ssh-copy-id -i ~/.ssh/id_rsa.pub linux
```

This command will login to `linux.cs.wwu.edu` and add the contents of `~/.ssh/id_rsa.pub` to the remote `~/.ssh/authorized_keys`. *Note:* If you didn't setup your config file from the previous section, you will need to run this command instead:
`ssh-copy-id -p 922 -i ~/.ssh/id_rsa.pub username@linux.cs.wwu.edu`

### 1.6.4 SSH-Agent

In order to effectively use your key and not need to type your password over and over to unlock the key, we use SSH-Agent. This agent will run in the background and respond to requests from the various ssh tools. First we need to load the agent and configure it's environment. Luckily this is easily done with the following:

```
eval $(ssh-agent)
```

```
Agent pid 2381
```

This shows that the agent is running and ready to accept connections. Next we need to add our keys to the agent so that the rest of the tools can use them. By using the `ssh-add` command we can load the private keys into the agent.

```
ssh-add
```

```
Enter passphrase for /home/username/.ssh/id_rsa:
Identity added:  /home/username/.ssh/id_rsa
(/home/username/.ssh/id_rsa)
```

`ssh-add` will prompt for the password to the key, unlock it, and give it to the agent. If you want to use a different key besides `/home/username/.ssh/id_rsa`, then you can specify it as an argument on the command line to `ssh-add`.

Once the ssh key is loaded, we can connect to services that have our public key listed in their `~/.ssh/authorized_keys`. For example if you have followed up to this point, we can connect to the Linux pool by simply typing:

```
ssh linux
```

No password will be asked. You will simply get a prompt and can start using the shell. If you run SCP it will just start copying, again, no prompt for a password.

One important thing to be aware of is that the SSH-Agent will still be around after you log out, which is not what you want. Make sure you kill off your SSH-Agent before logging out! An easy way to do this is by running the following, replacing "username" with your username.

```
kill $(pgrep -u username ssh-agent)
```

## 1.7 Windows

### 1.7.1 PuTTY

PuTTY is a free and open-source terminal emulator, serial console and network file transfer application that was written by Simon Tatham. It has been speculated that PuTTY stands for `Pu(Public Encryption)+TTY(Teletype)`, but there is no concrete meaning for it. It is a popular tool that is used by Microsoft Windows users to SSH into Linux of Unix servers that run an SSH server. PuTTY supports several network protocols such as SCP, SSH, Telnet, connect to a serial port, etc.

To download PuTTY, go to: https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html. Underneath "Package Files", you can either download the 32-bit or 64-bit version of MSI ('Windows Installer'). Once you have installed the correct executable that works with your computer, when you open up the

program, it should look similar to Figure 1.1. We have also added in the correct inputs for PuTTY to SSH into the school network in the image as well.

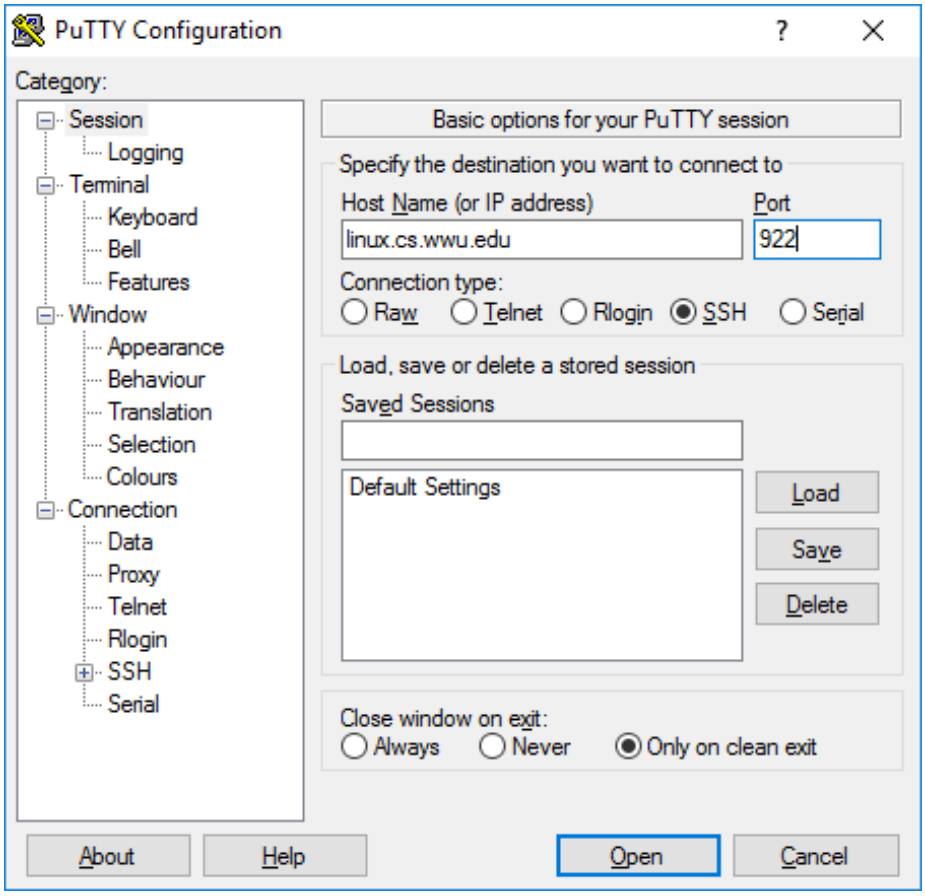


Figure 1.1: PuTTY Login Window

Do note that you can also save your sessions when you SSH into the network. By clicking save (button to the right, underneath load), you are able to save the session to whatever you'd like to name it. You are able to easily log back into that session afterwards by clicking on that session name, and clicking the load button. If you look at Figure 1.2, you are able to see that we have named and saved our current session as "linux".

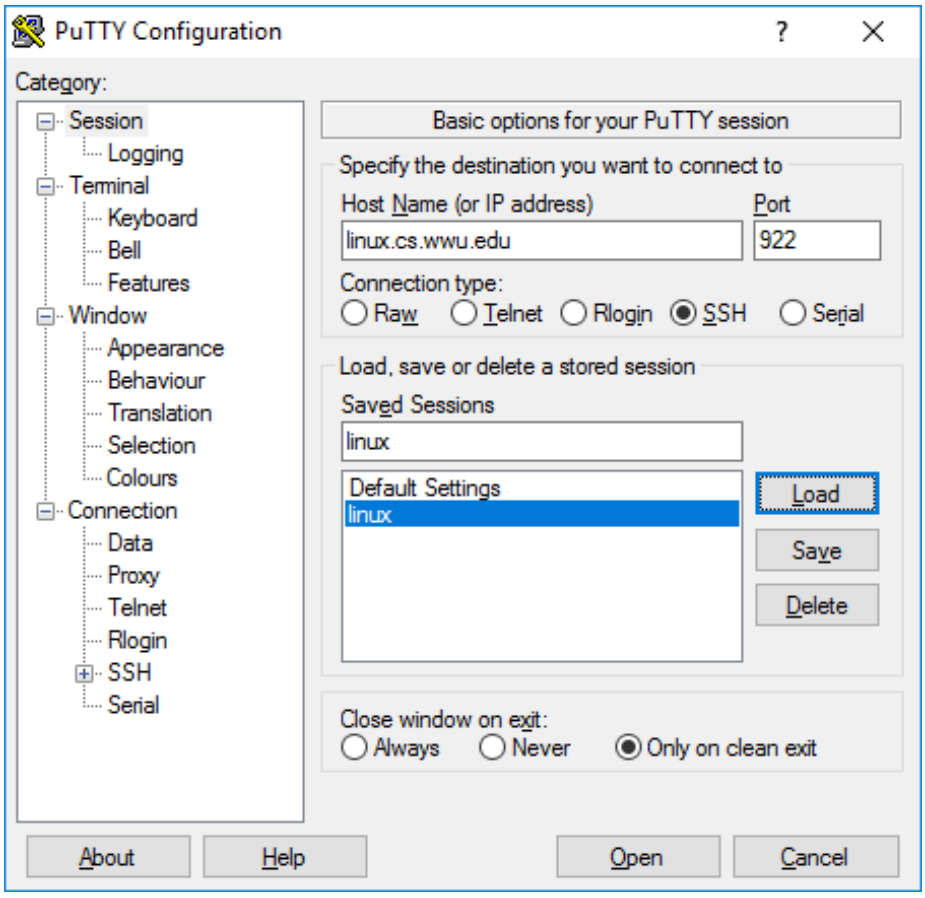


Figure 1.2: PuTTY Login Window with saved settings

### 1.7.2 WinSCP

While PuTTY provides a usable terminal for working remotely, many Windows users also want a graphical way to copy files. WinSCP provides a simple graphical interface that supports both SFTP and SCP transfers. WinSCP can be downloaded here: https://winscp.net/eng/download.php

Upon running WinSCP you will be greeted with a connection setup window like the following.

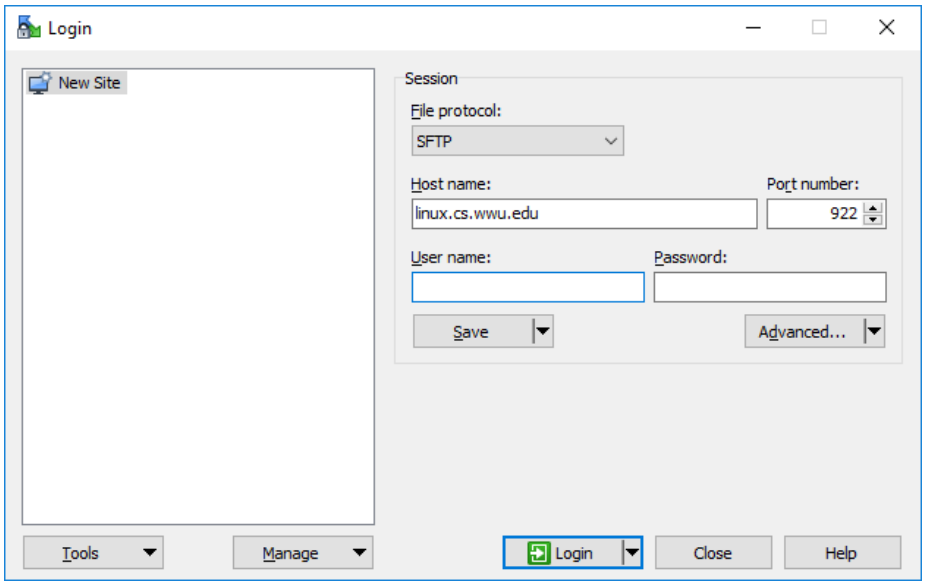


Figure 1.3: WinSCP Login Screen

After filling out your login information, click "Login" and you will be prompted to verify the host keys. Once approved, you will be presented with a two pane window. On the left hand side you have your local files, and on the right you have the remote files. Drag and drop works both ways.

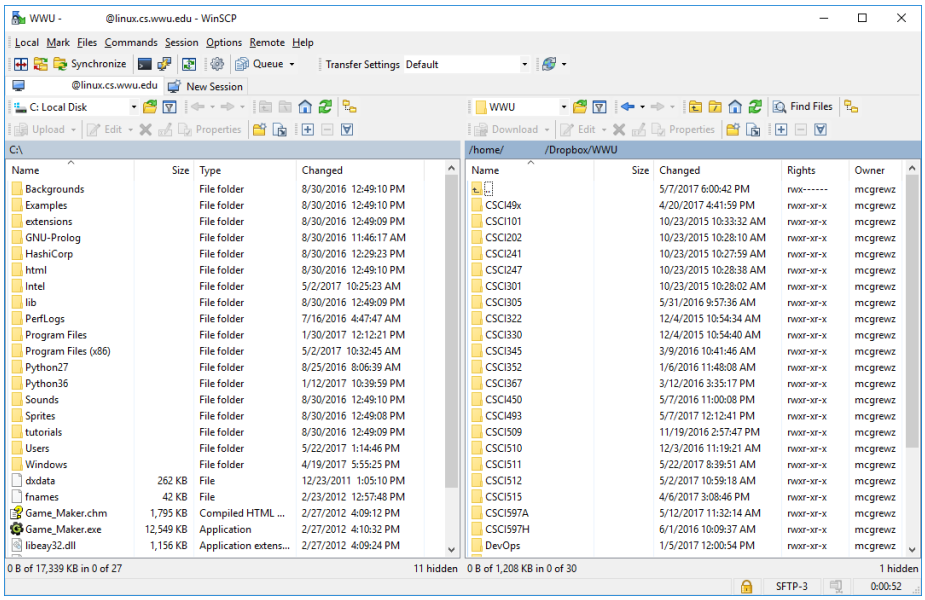


Figure 1.4: WinSCP File Browsing

### 1.7.3 Cygwin

Installing Cygwin is out of the scope of this workshop, but there are numerous tutorials online to help if you find it confusing. The important thing to remember is **DON'T CHECK EVERY BOX**. The default package selection doesn't include openssh, so make sure you check it. If you find yourself needing a package you didn't install earlier, simply run the installer again and select that package. The package we will use here is the "openssh" package.

Once Cygwin is installed you can open the Cygwin terminal from the desktop shortcut or Start menu and run all the SSH commands presented above.

# Chapter 2

# Slack

## 2.1 What is Slack?

Slack is a communication tool that can be used to organize teams, send direct messages between users, and set up channels for different topics. Slack can also be used for private channels for more sensitive information, and can be used to make calls like Skype, Google Hangouts, or Facebook Messenger. You can access the WWU CS Slack team by visiting: https://wwucs.slack.com
In order to register an account you must use your wwu.edu email address.

## 2.2 Why use Slack?

Slack has a lot of cool, useful features to help make communication easier and more organized. These features include Slack-bots, commands like /remind and /status, various text formatting, and document sharing. Not only is Slack a great tool, but Western's wwucs Slack team is a great place to go when you need help with an assignment (within the bounds of academic honesty), have a general question, or just want to talk to other nerds/classmates.

## 2.3 Things to Remember

wwucs.slack.com is monitored by various admins and professors, so conduct yourselves accordingly, and be respectful of each other. All posts and material shared in Slack should be safe for work material only, yes, even in the "random" channel. This ensures that everyone feels comfortable accessing the resources available to them. If someone is acting inappropriately, you can report them to an admin by going to the team directory, looking up the admins, and messaging one of them.

## 2.4 Channels

Slack channels are places where you can designate the topic or purpose of the conversation. You can join and create channels.

### 2.4.1 Joining

On the left menu bar inside Slack, there is a header labeled "CHANNELS". The number next to the CHANNELS header is the number of channels you can join in the wwucs Slack. You can join a channel by clicking on CHANNELS, which will take you to a different page with a list of all the channels you can join. When you click on one of the channels there, it will show you a preview of what the channel looks like, who is in it, and what is being discussed. If you want to join the channel, there is a JOIN button by the text entry field you can click, or you can hit the return/enter button on the keyboard.

#### #partyparrot

Join the partyparrot channel because it rocks. No pressure.

### 2.4.2 Creating

On the left menu bar, by the above-mentioned label CHANNELS is a + icon. If you hit the + icon, Slack takes you to a new page where you can specify if the channel is public or private, what the channel's name and purpose is, and who to send initial invites to.

#### Public vs Private

When creating a new channel, you are given the option of creating either a public or private channel. A public channel means that anyone on the wwucs team could see the channel or choose to join it. A private channel means that only the people you invite to the channel can see or join it. If you have a school project that is a pair programming assignment, or a senior project group, it may be beneficial to make a private channel for that. Otherwise, other students could see any code, conversation, or documents you post in that channel, which could lead to academic dishonesty.

## 2.5 Direct messages

Under the channels list on the side menu bar, there is a "DIRECT MESSAGES" tab, where you can add people the same way as you can add channels. You can direct message anyone on the team, but this provides an easy shortcut to doing so.

### 2.5.1 New messages

There are a few ways you can direct message someone. One is by going under the DIRECT MESSAGES tab on the left menu bar and clicking on the person's username, which takes you to a new sort of "channel" where it is just you and that person. Two other ways are using the commands /dm or /msg, followed by an @ and the username of the person you're messaging, then the message to send. This will create a shortcut to that person on the left menu bar under the DIRECT MESSAGES tab if there was not a shortcut for direct messaging them before.

### 2.5.2 Replying

Replying in Slack is simple, Slack will show you if there are messages for you in direct messaging by putting a number by the name of the person messaging you, indicating the number of unseen messages. When this happens on channels, it indicates there are unread messages by making the name of the channel bolded. To reply, switch to that direct message or channel and type your reply.

## 2.6 Messages

Messages are useful in keeping conversations organized. You can follow up regarding an older conversation, share the context of a discussion with members in other channels, and also keep other team members in the conversation in the loop.

### 2.6.1 Messages aren't permanent

You can delete or edit your messages. You can do this by hovering over it, and on the right side of the screen options should pop up, click on the ... button to show message actions. This will show options to edit and delete, among many others.

### 2.6.2 Pin Messages

In Slack you can also pin messages, files, etc. to provide easier access to look them up.

#### How to Pin Messages

You can pin a message by going to the ... button again to show the message actions. In this menu there is an option to "pin to this conversation...". You can also un-pin a message using this same button once it is pinned.

### 2.6.3 How to Access Pinned Messages

You can access pinned messages by going to the top left corner of the channel/message view, close to the left menu bar, and click on the push-pin-like button. This button will only be there if there are pinned items in the conversation you are in, and it will disappear if there are not.

### 2.6.4 Threaded Messages

If there was a message far back in the stack of messages in a conversation, and you want to reply to that message without worrying it is now out of context, you can hover over that message and click on the "start a thread" button that pops up on the right side by the ... button. This allows you to reply to a message.

### 2.6.5 Reactions

You can also add a reaction to a message by clicking on the "Add reaction..." button that appears when hovering over a message. (partyparrot!!!)

## 2.7 /Commands

As mentioned briefly before, there are commands you can type in a chat that can be pretty useful.

### 2.7.1 List of Commands

These commands include but are not limited to:

- /archive - Archive the current channel
- /away - Toggle your “away” status
- /collapse - Collapse all inline images and video in the current channel (opposite of /expand)
- /expand - Expand all inline images and video in the current channel (opposite of /collapse)
- /feed help [or subscribe, list, remove. . . ] - Manage RSS subscriptions
- /feedback [your feedback] - Send feedback to Slack
- /invite @user [channel] - Invite another member to a channel
- /join [channel] - Open a channel
- /keys - Open the keyboard shortcuts dialog

- /kick @user or /remove @user - Removes user from the current channel. This action may be restricted to owners or admins.
- /leave or /close or /part - Close a channel or direct message
- /me [your text] - Display action text, e.g. "/me does a dance" will display "does a dance"
- /msg user [your message] or /dm user [your message] - Send a private message to another user
- /mute - Mute a channel, or unmute a channel that is muted
- /open [channel] - Open a channel
- /prefs - Open the preferences dialog
- /remind me in [time] to [message] or /remind me to [message] at [time] - Set a Slackbot reminder that will send you a direct message at the time you specify. To schedule a reminder for a specific date, use the format MM/DD/YYYY or DD.MM.YYYY.
- /remind help - Learn more about how to set reminders.
- /remind list - Get a list of the reminders you've set.
- /rename [new name] - Rename a channel (admin only)
- /search [your text] - Search Slack messages and files
- /shortcuts - Open the keyboard shortcuts dialog
- /shrug [your message] - Appends a shrugging emoji to your message
- /topic [text] - Set the channel topic
- /who - List users in the current channel

### 2.7.2 Useful commands

Demo on use of:

- /shrug
- /remind
- /me
- /status
- /dnd

## 2.8 Formatting Messages

### 2.8.1 Markdown-ish

Slack borrows some of the ideas from Markdown formatting. The important takeaway here is that it borrows them, and doesn't actually completely comply with real Markdown. The markdowns listed below can be used in messages. They can be used to put emphasis on any point that you are trying to get across.

- *bold* - **Bolded** text
- _italics_ - *Italicized* text
- ~strikethrough~ - ~~Strikes through text~~
- > Single-line Quote - Blockquote a single line
- >>> Multi-line quote - Blockquote multiple lines

Additional help with formatting can be found here: https://get.slack.help/hc/en-us/articles/202288908-Format-your-messages

### 2.8.2 Lists

Pressing [Shift] + [Enter] or [Control] + [Enter] will create new lines in your message. Simply start each line with a number or bullet point.

### 2.8.3 Code Snippets

Code snippets are useful when trying to emulate or explain code to someone over Slack. Please note the ` are back ticks (above the tab key on the left of the keyboard) and not quotation marks.

- `code` - Unformatted code style which can go inline with text
- ```code``` - A code block, preformatted code style that can span multiple lines