How to get started on Amazon EC2

Created June 2022 by Paul Bodily

Average time for assignment completion: 3-4 hours

The material in this tutorial I first encountered as an undergraduate research assistant in 2007. I wish I had had a tutorial like this at that time! **Knowing the basics of command line, SSH, Linux, Vi, and Git makes you extremely marketable and should be included on your CV/resume.** The tutorial is designed with sections be completed in order, but also as a quick reference should you need to come back later. I hope to have created here a resource you can draw back on throughout the semester and beyond!

Important note: All commands on Linux are case-sensitive. Be sure to pay close attention to letter casing (i.e., upper- and lower-case) in all of the below instructions, including the usernames, passwords, folder, and filenames you create. You will risk getting a 0 on the assignment or having to do it all over if your casing doesn't match what is in this tutorial!

If you experience trouble as you work through these steps, get help! Use Google, post on Discord, visit the TA during office hours, ask questions in class, or visit with Dr. Bodily during office hours!

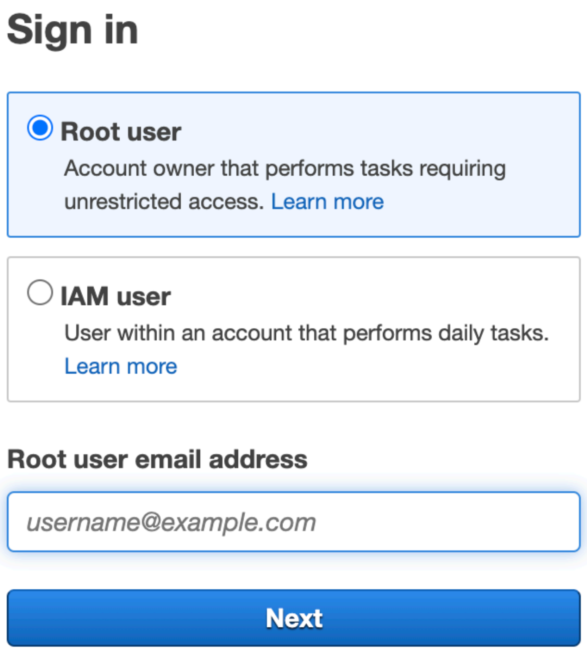
Important note: Attention to detail is critical on this assignment. To save you time and to avoid running into excessive problems, please **follow each step in order carefully**. If you have not already done so, I recommend making sure that your operating system (OS) is configured to display file extensions (if you need help, query Google for "displaying file extensions on [your OS]").

Table of Contents

Launching a server

Create an AWS account and sign up for a free EC2 web server

1. Go to https://portal.aws.amazon.com/billing/signup
2. Input an email address (e.g., your ISU email address), an AWS account name (e.g., first four letters of your last name followed by first four letters of your first name, all lowercase), and select "Verify email address"
3. Retrieve the verification code sent to the email you entered in the previous step and use it to verify your account
4. Select and confirm your "Root user password"
5. Indicate that you plan to use AWS for "Business - for your work, school, or organization" and fill out your contact information for the account. You can put "Idaho State University" for the "Organization name". Read and agree to the terms of agreement, and continue to the next step.
6. Enter credit or debit card information. **You will not be charged anything**. There will be an authorization charge of $1 sent to verify that the card is valid, but AWS doesn't proceed with the charge, and the charge should disappear within three to five business days. Having a card on file is in case you exceed the limits of the tier you are signing up for. AWS will make every effort to notify you before this should happen. The work you will do for this class will not require you to go over these limits. **A common issue is for your bank to reject this charge. In this case, try a different card and/or contact your bank.**
7. Confirm your identity on the next page and complete the security check.
8. Select the "Basic support - Free" plan and complete the sign up.
9. Select "Go to the AWS Management Console" (also accessed via https://console.aws.amazon.com/console/home). ***You should bookmark this site.***
10. Click on "Sign in using root user email". Log in as the root user using the account credentials for your AWS account (created above). On the page labeled "Keep your account secure" you can select "Skip for now" for now.


Sign in

[✓ Root user]
Account owner that performs tasks requiring unrestricted access. Learn more

IAM user
User within an account that performs daily tasks. Learn more

Root user email address

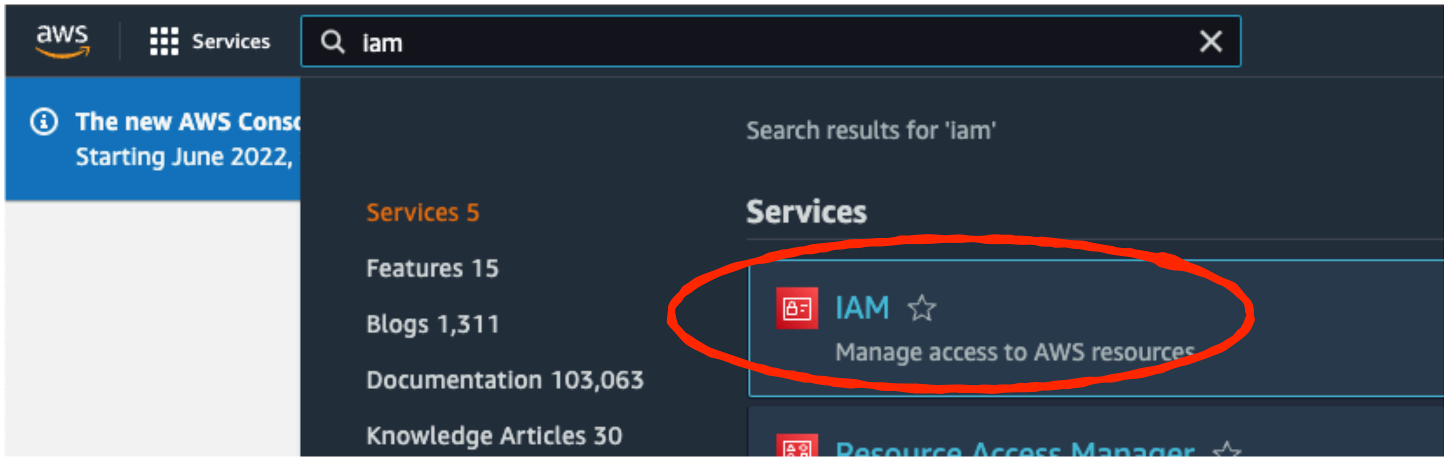
username@example.com

Next

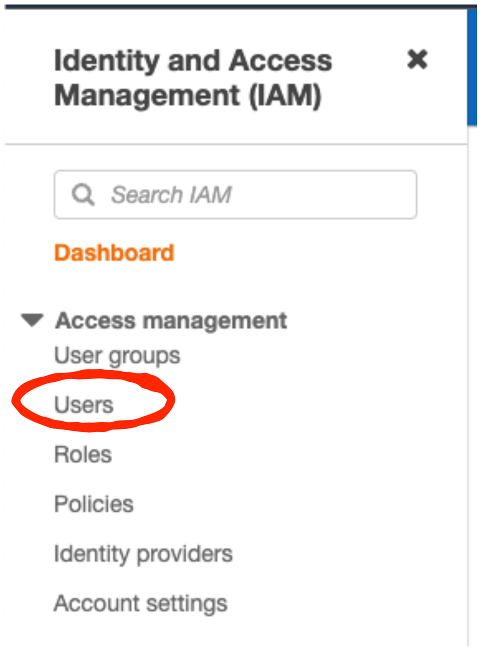
Adding a user in the AWS Management Console

Having access to the AWS Management Console will allow you to manage your server and to see details about the server that are needed to log in to the server. Because I or the TAs will need to log in to the server (e.g., to grade assignments, to help you resolve issues, etc.), it will be helpful for us to also have access to these same server management details. Here you will create an AWS Management Console account for me/the TAs. **Note that this account is just for *managing* the server, not for logging into or working on your server** (we'll create that account in a different step).

11. In the search box, type "IAM" and select the "IAM" service



12. In the lefthand menu, select "Users".

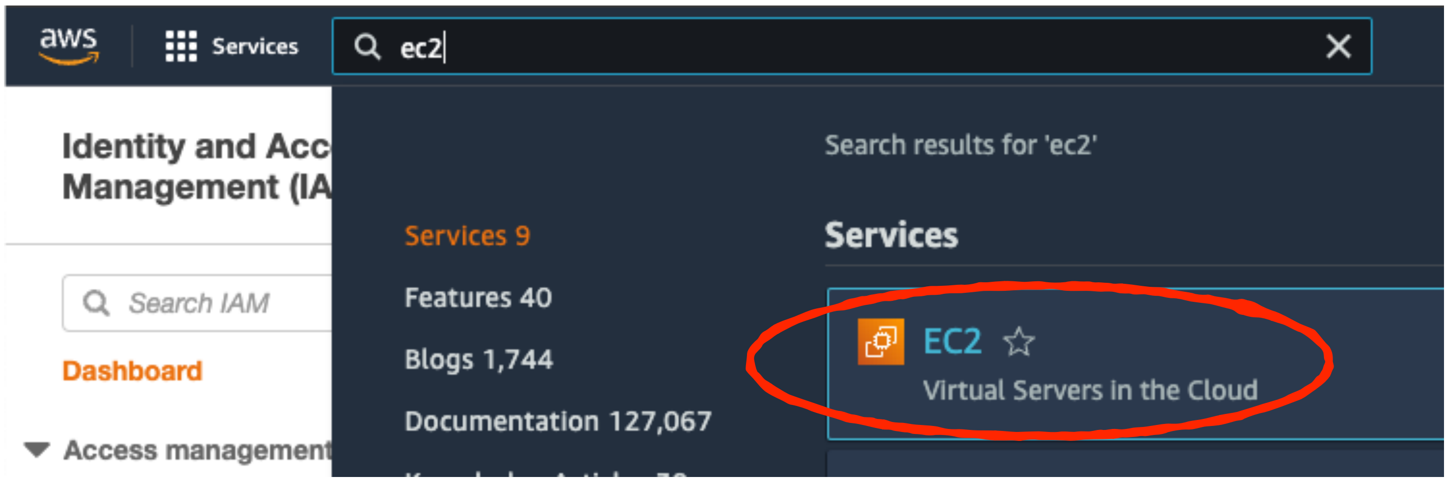


13. Using the "Create user" button, add a user. The User name should be "bodipaul".
14. Check the box to "Provide user access to the AWS Management Console".
15. Select "I want to create an IAM user".
16. The "Console password" can be left as "Autogenerated password". Ensure the box "Users must create a new password at next sign-in" is checked. Hit "Next" to go to "Set Permissions".
17. In "Set permissions", select "Add user to group" and create a group. This group should be given an appropriate name (e.g., "admins") and be assigned the policy "AdministratorAccess". After creating the group, check the box next to the newly created group to add the user to the group. Hit "Next" to go to "Review and create". Assuming all fields have been set appropriately, click "Create user".
18. On the next page, click "Show" to show the password. **Copy and save this password**. You will need to submit it as part of your submission for this assignment. You should also click the "Email sign-in instructions" button and email the instructions to isu.cs.1337@gmail.com. (This button is designed to use the "mailto:" feature of your browser. If you haven't set up your browser's default email client for mailto links, here are instructions on doing so: https://support.procore.com/faq/how-do-i-change-my-web-browsers-default-email-client-for-mailto-links.)
19. If at any point you need to create, change, or delete an IAM user password, follow the corresponding instructions on this site: https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_passwords_admin-change-user.html.

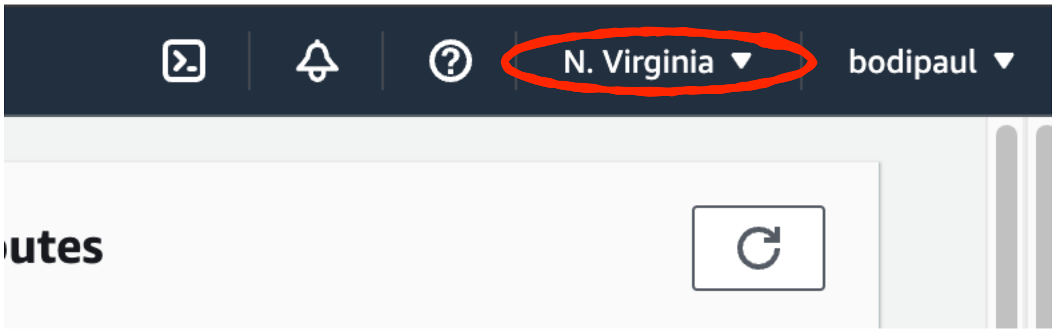
Launching a server instance

Amazon calls its web servers *instances*, because you can have many of them running in parallel. For now, you only need one instance. After launching a server instance, that instance will be "running". **The time that an instance is running counts against your 750 hours per month. Note that 31 days is only 744 hours, so as long as you are only ever launch/are running a single instance, you will never exceed your 750 hours per month that are allotted for the free tier.** Here are instructions on how to launch your server instance.

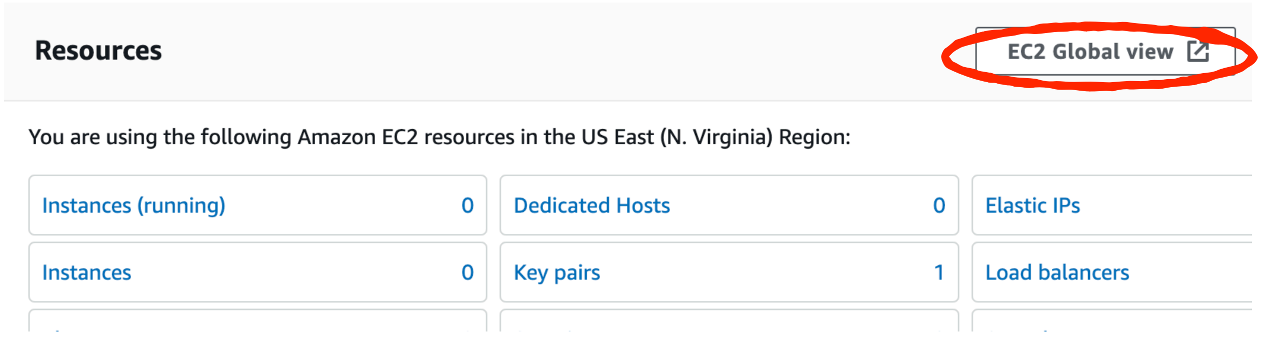
20. From the search box, type "EC2" and select the "EC2" service.



21. **WARNING:** Part of launching a server instance is being aware of where that server instance is ***physically*** located. You will be asked to report this information in your submission for this assignment. The physical location of the server can impact latency times. For this reason, some users will launch multiple instances in multiple geographic regions so that users in those regions have decreased latency. **You should take careful note of the region in which you are operating so that you avoid accidentally launching multiple server instances in different regions. The region is indicated in the top right of the page next to your username:**



Noting the region in which you launch your server instance is important because by default the EC2 Management Console view is region-specific. That means if you launched your instance in "N. Virginia", but the region you are currently looking at in the Console is "Ohio", you will not see the N. Virginia instance listed. This can inadvertently lead some users to erroneously assume they have no instances running and to subsequently launch a second instance. Launching a second instance will eventually result in you exceeding the limits of the free-tier and being charged for the extra usage. Be sure the first thing you do each time you log in is to verify the region which you are viewing. To see all instances across all regions, you can select the "EC2 Global view".



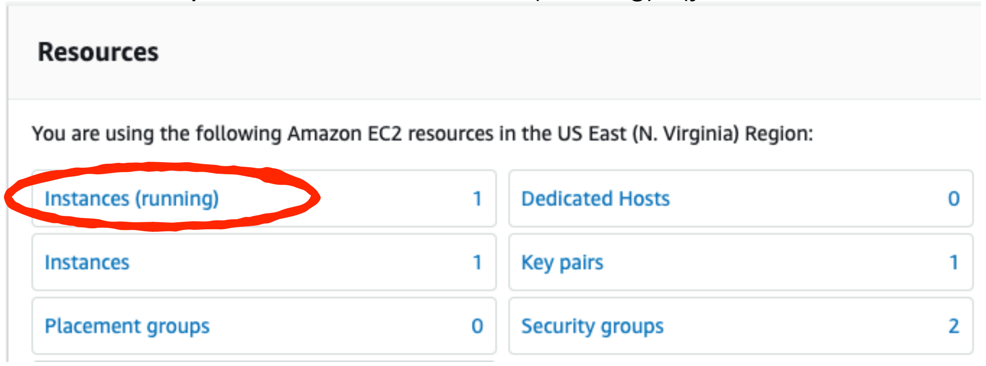
Resources

EC2 Global view

You are using the following Amazon EC2 resources in the US East (N. Virginia) Region:

Instances (running)	0	Dedicated Hosts	0	Elastic IPs
Instances	0	Key pairs	1	Load balancers

22. From the "Resources" pane, select "Instances (running)" (yours will have a "0" next to it)

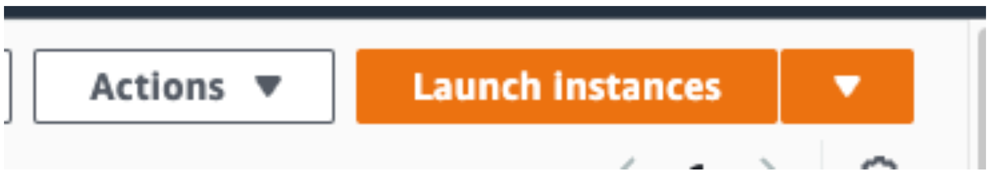


Resources

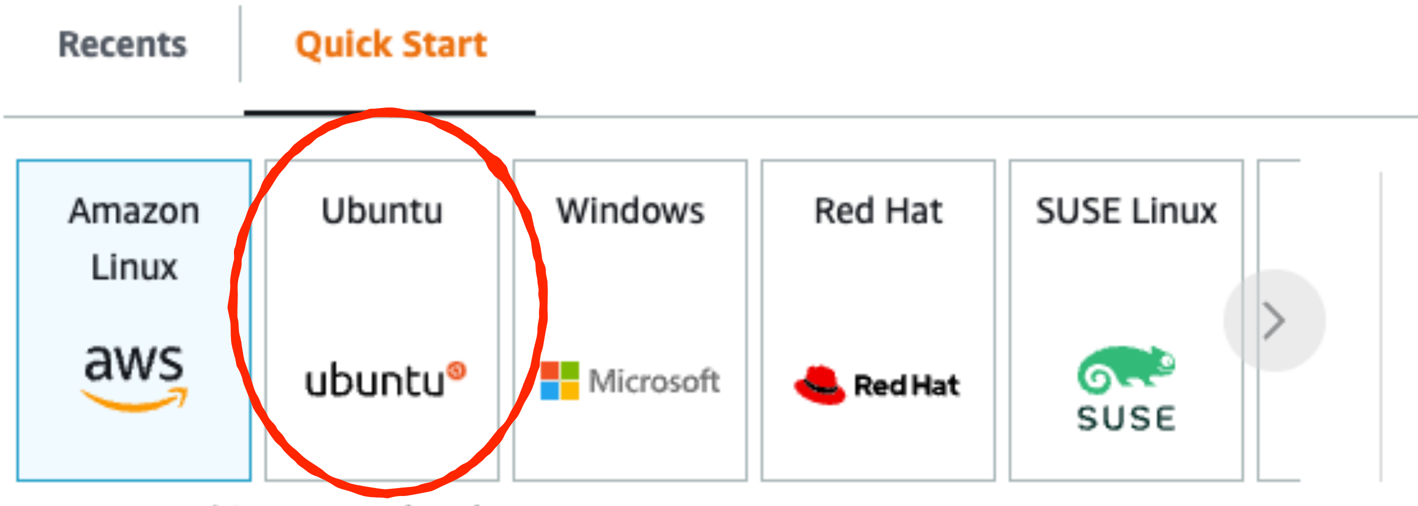
You are using the following Amazon EC2 resources in the US East (N. Virginia) Region:

Instances (running)	1	Dedicated Hosts	0
Instances	1	Key pairs	1
Placement groups	0	Security groups	2

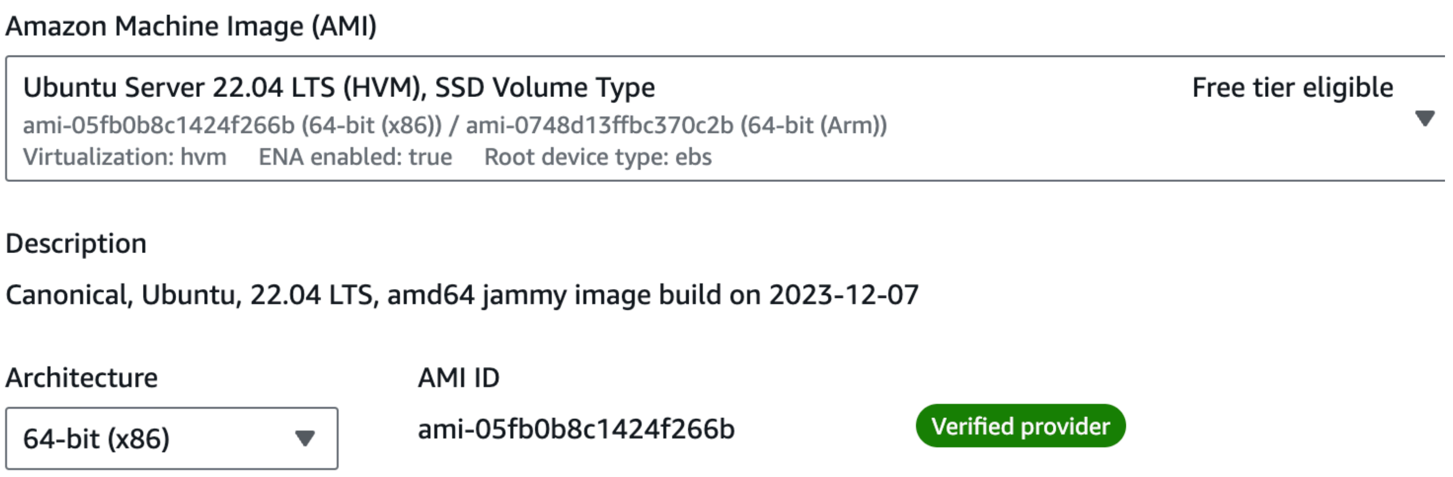
23. In the upper right corner, select "Launch instances"



24. Give the server a meaningful name (e.g., "Paul Bodily's Web Server")
25. From the "Quick Start" menu select the "**Ubuntu**" Amazon Machine Image. Note that this **Amazon Machine Image (AMI)** is a commonly used acronym that (in our case) will refer to this Ubuntu installation. (An **image** is a serialized copy of the entire state of a computer system.)



26. From the dropdown menu select the Ubuntu Server with SSD Volume type that is free tier eligible (likely the default will be fine). Be sure that the "64-bit (x86)" Architecture is selected.



Amazon Machine Image (AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type — Free tier eligible
ami-05fb0b8c1424f266b (64-bit (x86)) / ami-0748d13ffbc370c2b (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Description

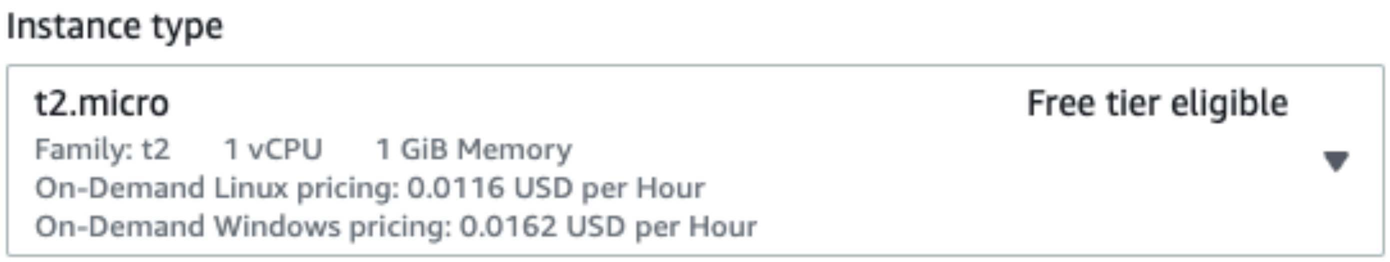
Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2023-12-07

Architecture: 64-bit (x86)

AMI ID: ami-05fb0b8c1424f266b

Verified provider

27. For the "Instance type", select "t2.micro", which is "free tier eligible".


Instance type

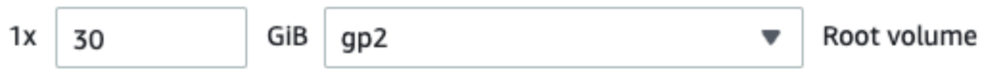
t2.micro — Free tier eligible

Family: t2 1 vCPU 1 GiB Memory

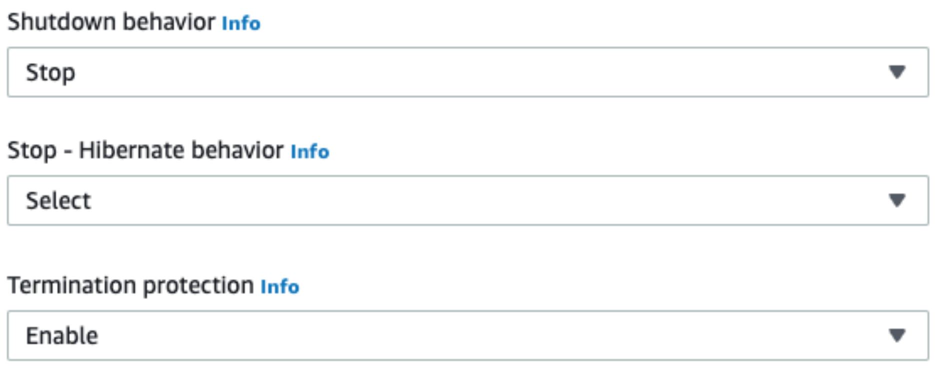
On-Demand Linux pricing: 0.0116 USD per Hour

On-Demand Windows pricing: 0.0162 USD per Hour

28. For the "Key pair (login)", select "Create new key pair". The Key pair name should be something descriptive (e.g., "paul_macbook")[1]. It should be RSA type. **If the local machine you are using runs Linux or MacOS**, the "Private key file format" will be ".pem"; **if you are running Windows**, then select ".ppk". Selecting "Create key pair" will download a file containing your key pair (which, should you be interested, can be visualized in a basic text editor). Selecting "Create key pair" will result initiate the downloading of a key-pair file. **You will need this key-pair file later so don't lose it**.
29. Leave Network settings alone. Under "Configure storage", set it to "1x 30 GiB gp2". You do **not** need to click "Add new volume".


1x 30 GiB gp2 Root volume

30. Under "Advanced details", set "Shutdown behavior" to "Stop", and set "Termination protection" to "Enable".


Shutdown behavior: Stop

Stop - Hibernate behavior: Select

Termination protection: Enable

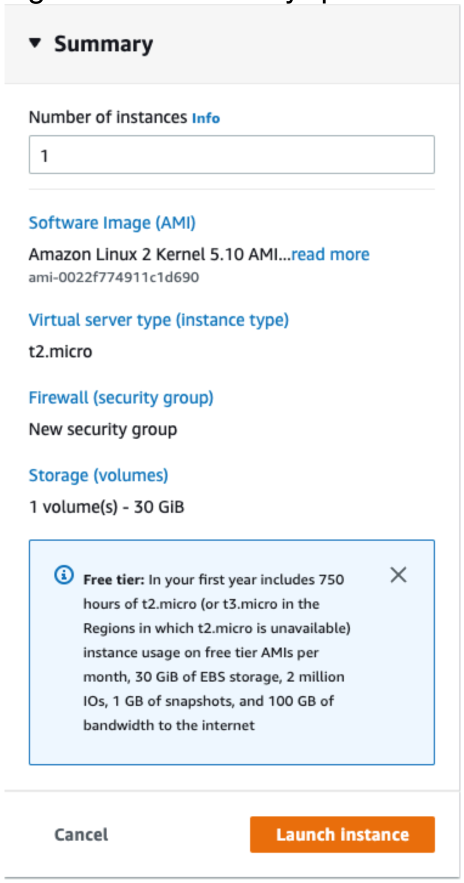
<u>WARNING</u>: The purpose of these settings is to help you avoid accidentally wiping all of the data from your server. A "stopped" server instance allows the data and configuration of the instance to be preserved even though the server is not running (i.e., accessible for public users). **<u>Note that a "stopped" instance still counts against your 750-hour quota.</u>**

When the time comes that you want eliminate the server and everything on the server, you will need to "Terminate" the server. Because you have enabled "Termination protection," you will

[1] For this and all other filenames, I **strongly** recommend that you avoid filenames that include spaces or punctuation marks other than the underscore or dash characters. Because command lines use spaces to delimit arguments, it can get messy trying to pass filenames that contain spaces as arguments. If you *do* encounter filenames with spaces or weird punctuation, tab-completion is your friend and will automatically fill in the appropriate escape character sequences to match these files.

see that the option to "Terminate" is disabled in the standard menus. To terminate, simply right click on the server and adjust settings to disable "Termination protection" prior to then Terminating the instance. **When you "terminate" an instance, it ceases to count against your usage quota. Termination will wipe everything from your server instance.**

31. If at any point now or in the future you receive notification that underlying hardware is degraded (rare), follow the instructions here: https://aws.amazon.com/premiumsupport/knowledge-center/ec2-linux-degraded-hardware/
32. Hit "Launch Instance" in the righthand "Summary" panel



▼ Summary
Number of instances Info
1
Software Image (AMI)
Amazon Linux 2 Kernel 5.10 AMI...read more
ami-0022f774911c1d690
Virtual server type (instance type)
t2.micro
Firewall (security group)
New security group
Storage (volumes)
1 volume(s) - 30 GiB
Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet
Cancel
Launch instance

33. From the search box, type "EC2" and select the "EC2" service again (you may need to do a hard refresh of the page). Ensure that you have the correct region selected in the upper right (the same region in which you launched your instance). You should now see that you have one instance running.

Checking server usage

Periodically you can (and should) check that your forecasted usage is below what is allotted for your free tier. The usage and forecasted usage are for the current month.

34. From the search box, type "Billing" and select the "Billing and Cost Management" service.
35. In the lefthand menu, select "Free Tier".
36. Check that the forecasted usage is below 100% of allotment. (Close to 100% is fine, as long as it is below.) **Note** that if you only barely launched your instance then you will likely see that "There is no Free Tier services data available". Come back in a little bit when AWS has had the chance to prepare cost and usage data.

Rebooting your server

If at any point you have trouble accessing your server, or you get an email or message indicating that you need to reboot your server, here are instructions on how to reboot your server.

37. From the search box, type "EC2" and select the "EC2" service.
38. Ensuring that you've selected the correct region in the upper right corner, select "Instances (running)" in the Resources menu.
39. Right-click on the instance you want to reboot and select "Reboot instance".

Configuring a server

Accessing the terminal

Now that you have launched a server instance, your server is live! This means a dedicated physical machine is running somewhere remotely and is accessible via the internet. You will access and interact with your server through the terminal. The ***terminal***, also known as the ***command line interface (CLI)*** or ***console***, is a powerful textual interface that allows the user to accomplish and automate tasks on a computer (and/or on a remote computer) without the use of a graphical user interface. Every computer has one, but they are called and accessed differently depending on which operating system (OS) you are using. You will use the terminal for various tasks related to your server.

40. **In Windows:** Click Start and search for "Command Prompt." Alternatively, you can also access the command prompt by pressing Ctrl + r on your keyboard, type "cmd" and then click OK.
41. **In MacOS:** Open Launchpad and search for "terminal". Alternatively, you can access the terminal by pressing ⌘ + space on your keyboard and searching for "terminal."
42. **In Linux:** Depending on which interface you use (e.g. GNOME, KDE, Xfce), the terminal will be accessed differently. We recommend you check Ubuntu's **Using the Terminal** page for the various ways to access the terminal.
43. Your local computer will likely have an **SSH client** installed by default. You can verify this by typing `ssh` at the command line and hitting enter. If your computer doesn't recognize the command, you need to install an SSH client (see step 47 below).
44. If you're using MacOS or Linux, take this moment to set permissions of your private key file for later use. You can do this by running the command

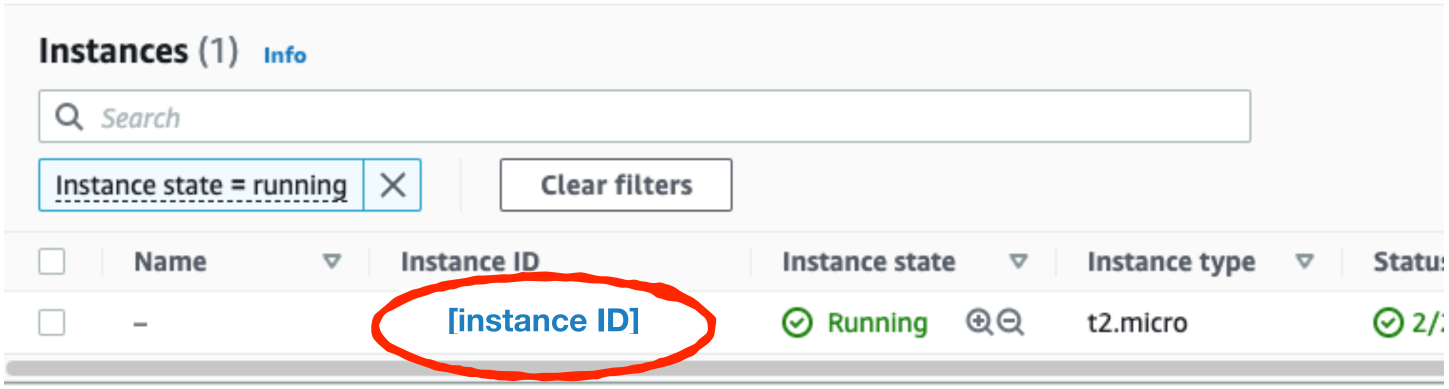
    ```
    chmod 400 ~/Downloads/my-key-pair.pem
    ```

    where `my-key-pair.pem` should be the name of your key-pair file. Note that this assumes that your key-pair file is still in your Downloads folder. Press enter to run the command.

SSHing into your EC2 instance as the default user

SSH, also known as Secure Shell or Secure Socket Shell, is a network protocol that gives users, particularly system administrators, a secure way to access a computer over an unsecured network. Initially your server has only one server account (**note the server account is distinct from the AWS server *management* account created above**). This default server account has the default user name **ubuntu** since that is the AMI that was used to launch your instance. In your career as a programmer, you will rarely interact with a server as the default user, so the only thing you will do as the default user is to set up other server accounts (one for you and one for me). Setting up these other accounts requires first logging in (i.e., "SSHing" in) with the default account. **This is frequently the step where students run into the most issues. If you run into errors, try Google, try Discord, try the TAs or the professor, you can even try reaching out to Amazon. Share your successful strategies on the course discord. Sometimes simply rebooting the instance (see section above on rebooting) can work, or terminating the instance and launching a new one.**

45. From the "Instances (running)" menu (accessible from the EC2 menu), select the instance that you just launched.



	Name	Instance ID	Instance state	Instance type	Statu
	–	[instance ID]	Running	t2.micro	2/

46. The instance summary will provide you with some important data. A full rundown of what is shown is described here: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/connection-prereqs.html. For now, you will need the IP address listed under **Public IPv4 DNS**. Stopping or rebooting your instance can cause your IPv4 address sometimes to change, so if you at any point find yourself unable to log in, double check your IPv4 address. **You will need to report this Public IPv4 DNS address in your Moodle submission for this assignment, so keep it handy**.
47. **If running MacOS or Linux,** in your terminal, log into your server instance as the default user 'ubuntu' by running the following command:

```
ssh -i ~/Downloads/my-key-pair.pem ubuntu@12.34.56.78
```

where `my-key-pair.pem` should be replaced with the path to your key-pair file, and `12.34.56.78` should be replaced with your **Public IPv4 DNS** address from the previous step. If you did not set permissions for your key-pair file, go back to step 44 that explains how to do this.

If running Windows, you need to download and install PuTTY (https://www.chiark.greenend.org.uk/~sgtatham/putty/). Launch PuTTY, and enter your instance's **Public IPv4 DNS** as the IP address. In the lefthand menu, navigate to Connection/SSH/Auth/Credentials. Click **Browse** next to private key access and select the .ppk file containing your key. Finally, click **Open**. When prompted for a username, log in as the default user '**ubuntu**'. More information about connecting to your Linux instance from Windows can be found here: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/putty.html.
48. You might next see a response like the following (PuTTY users may not):

```
The authenticity of host 'ec2-198-51-100-1.compute-1.amazonaws.com
(198-51-100-1)' can't be established.
ECDSA key fingerprint is l4UB/neBad9tvkgJf1QZWxheQmR59WgrgzEimCG6kZY.
Are you sure you want to continue connecting (yes/no)?
```

Enter `yes`. You will see a response like:

```
Warning: Permanently added 'ec2-198-51-100-1.compute-1.amazonaws.com'
(ECDSA) to the list of known hosts
```

You will then see a series of login messages and a command prompt. Congratulations, you are logged in to your remote server instance as the default user.

To reset your server login password

49. If you ever need to reset a password for an account, you can always log back in as the default user following the steps in the previous section.

Adding user server accounts (with sudo privileges)

In your career as a programmer, you will rarely interact with a server as the default user, so the only thing you will do as the default user for now is to set up two other server accounts (one for you and one for me), each with sudo privileges. Don't worry about doing updates with the default user; we'll do that in a minute. The username for me should be `bodipaul`. Choose your own username for the server account that you will use for the semester.

50. To add a user `bodipaul` enter the following command:

    ```
    sudo adduser bodipaul
    ```

    When prompted for the password, go ahead and pick a good password for me (I'll change it later anyway). **You will need to report this password in your Moodle submission for this assignment, so keep it handy**. It won't show the password as you type it, but it is recording it. So make sure you type it in correctly and then just hit enter. If you make a mistake as you're typing, you should be able to just hold down delete for a few seconds and start again. For the user information, just press ENTER for the default for all fields. Type 'Y' to indicate all information is correct.
51. The keyword `sudo` is short for "super user do". It is typed before commands that may require super user privileges to perform (like adding users). You will want and need sudo privileges on the accounts you create for this course. As you saw in the previous step, the default user has sudo privileges. To add sudo privileges to the `bodipaul` user, enter the command:

    ```
    sudo usermod -aG sudo bodipaul
    ```

52. Repeat the previous two steps to add an account for yourself with sudo privileges. Be sure to keep track of your password. **You will need to provide your case-sensitive username in your submission for this assignment**.
53. You need to modify a configuration file to allow users to login via SSH. This file is protected by sudo privileges and thus requires the keyword `sudo` when editing it. We will use the a command line text editor called **Vi** to edit the file. The full command is

    ```
    sudo vi /etc/ssh/sshd_config
    ```

    Basic instructions for using Vi to edit a file are in the next section below. Following those instructions carefully, edit the file so that `PasswordAuthentication` (line 57, scroll down with the arrow keys) is set to `yes`. Then set `KbdInteractiveAuthentication` (line 62) to `yes`. **Ensure there is no leading '#' character on any of these lines (which would comment them out).** Save and quit Vim.

54. Restart the SSH service by running the following command from the commandline:

    ```
    service ssh restart
    ```

    When prompted to choose an identity to authenticate, choose the server account that you created for yourself (in step 52) and input the password you created for that account.
55. **Prior to logging out, use the instructions in the section below to update your server.**
56. This completes all you need to do as the default user. To **log out**, enter the terminal command:

    ```
    exit
    ```

Editing files in the terminal using Vi

Vi is the universal text editor of Linux. You can use it to edit text files on Linux systems. **Vim** is simply an improved version of Vi, but unlike Vi, Vim is not universal. An excellent resource for learning vim is simply to enter the `vimtutor` command in Linux and follow the tutorial.

57. Once in Vi, you can use the arrow keys to navigate to where you want to make changes.
58. Type '`i`' (for insert) to enter **insert mode**. Once in insert mode you can add and delete characters as normal.
59. Once edits are done, press `esc` to exit insert mode and to return to **command mode**.
60. In command mode, '`u`' is undo.
61. In command mode, type '`:wq`' (':' is preparatory to issuing commands, `w` means "write" (i.e., save), and `q` means "quit").
62. You might consider finding a good Vi or Vim shortcut sheet that you like.

Keeping your server up to date

When you first log in each time you will see several log in messages. Sometimes those messages will include a notice that updates are available for your server. You'll likely see a message telling you that "System restart required". You can ignore this message as it should not affect anything you will need to do with your servers this semester.

63. To update your server enter the following commands in order, entering your password and the letter Y at the appropriate prompts to continue:

    ```
    sudo apt-get update

    sudo apt-get upgrade
    ```

    Simply enter "Y", "yes", or "Ok" as prompted until update and upgrade are complete.

SSHing into your EC2 instance with your server account

You used a private key (as stored in the .pem or .ppk file) to log in as the default user. In most real-world scenarios you will not interact with servers as the default user. Hence, you will use the server account you just created for the rest of the class. You will use a password to log in with this account.

64. **If running MacOS or Linux**, if my username were `bodipaul` and my Public IPv4 DNS address were `12.34.56.78`, I would log in by running the following command:

    ```
    ssh bodipaul@12.34.56.78
    ```

    When prompted, enter your password. You're in!

    If running Windows, if my username were `bodipaul` and my Public IPv4 DNS address were `12.34.56.78`, I would open PuTTY, and in the Host Name field I would type:

    ```
    bodipaul@12.34.56.78
    ```

    Before hitting 'Open', give a name to save the session (e.g., '1337 AWS SSH') and hit 'Save' (this way don't have to reenter the Host Name each time). *Common error: make sure you put **both** the username and the IP address in the Host Name field **as shown above***.

    For all users, if you get the message "Permission denied (publickey).", log back in as the default user and go back to step 53 and ensure you have the server configured to allow SSH. If you are unable to login in as either the default use or the username you created, you will need to **terminate** the instance you created (first disabling termination protection), and launch a new instance.
65. (Recommended) Note that there are ways to be able to login to your server account *without having to enter your password each time*. **For MacOS and Linux users**, details are here: https://www.thegeekstuff.com/2008/11/3-steps-to-perform-ssh-login-without-password-using-ssh-keygen-ssh-copy-id/. Complete all three steps. Note that where it refers to "remote-host" you should put in your your username@IPv4_address (e.g., bodipaul@12.34.56.78).

    For Windows users, there are methods to being able to not have to enter your password each time you login, but I haven't been able to find a simple one. I suggest just entering your password each time you log in. (If you find a good one, share it on the class discord!)
66. (Recommended) **For MacOS and Linux users**: You can also *simplify server login* by storing the login details (i.e., hostname, username) in your user-specific SSH configuration file on your machine (you cannot store your password this way, however, as storing passwords anywhere in plain text is not secure, but if you did the previous step, you shouldn't need the password anyway). Details are here: https://linuxize.com/post/using-the-ssh-config-file/.

    For Windows users: if you saved your PuTTY session above, you can just double click on the saved session in PuTTY each time you login.

Getting started on a server

Some basic Linux/Unix terminal commands

When you first login to your server, all commands you enter are by default executed from your home directory. Some terminal commands are a single word (executed by hitting enter). Other commands take additional arguments delimited and separated from the command name using white space. (There are some fun easter eggs hidden in terminals, too.) These commands are ones you will use over and over again, so try them each now and get use to them:

67. `pwd` - this command (which stands for "print working directory") echoes (i.e., prints) back to you the path to the directory you are currently in (referred to as the working directory), which on a freshly opened terminal is the home directory.
68. `ls` - this command (which stands for "listing") echoes back to you a list of the files and folders in your working directory. If nothing is printed, it is because the working directory is empty.
69. `mkdir temp` - the command `mkdir` creates a new directory named `temp`. If you run `ls` again you will see the directory you just created.
70. `cd temp` - the command `cd` (which stands for "change directory") changes your working directory to the directory named `temp`. You can check this by running `pwd`. If you run `ls` again you will see listed the contents of this directory (which, since it is newly created, is nothing).
71. `ls ..` - (note that is a lowercase LS, not a `1`) for any directory, "`..`" is a reference to the directory's **parent directory**. In this case, the `ls` command lists the contents of the parent directory (which if you've been following along will be the contents of the home directory again).
72. `cd ..` - What does this do? You guessed it: changes the current working directory to the current working directory's parent directory.
73. `rm temp` - the command `rm` (which stands for "remove") **is an extremely dangerous command in the terminal—*so be careful with it!*** In this case you will get an error warning because terminals do not let you remove directories without an additional flag. But in general, the `rm` command removes whatever files are listed after the command—and they never come back. There is no "trash" you can pull them back out of—no "undo" feature. **So be very careful.**
74. This time type `rm -r tem` (this is not a typo—leave the 'p' off), but before you hit enter, hit the `tab` key instead. You should see the line automatically completed for you—terminals have tab completion. As long as what you have typed so far has a unique match to a file or folder in your working directory, it will tab-complete (if there are multiple matches, it will list all the matches it finds so far and wait for you to add more letters). Tab-complete will save you a lot of time and headaches!
75. `rm -r temp` - the `-r` is what we call a **flag**. It's a boolean switch that, in this case for this command, tells the `rm` command to execute *recursively*, removing not only the directory, but also recursively removing all of the contents of the directory and any subdirectories.
76. `man rm` - this command (which stands for "manual") shows a manual entry for the command provided as an argument (in this case `rm`). Using the arrow keys, you can scroll up and down and see what types of flags and arguments can be given to this command and what they do.

`man` is your friend in learning to use terminal commands effectively (you can also always use Google). When you're done, use the `esc` key to get back to the command line.

77. Another common command is the `mv` command. See if you can use `man` to learn what `mv` does and how to use it. *Hint: it is used both for renaming files/directories and for moving them.*

Git and GitHub

Installing Git on your server

Git is software for backing up and tracking changes in files. Chances are very high that you will use Git regularly in your career to coordinate work among programmers collaboratively developing source code during software development. Git is so common that it has become the norm for prospective employers to ask to see your Git repository when you apply for a job. We use Git in this course for 3 primary reasons: 1) to teach you the basics of how to use Git on the commandline; 2) to allow you to back up your work in case the server crashes (unlikely); and 3) so that you can build up a portfolio of work you have done to show to future employers.

78. Login to your server. (You can install Git from any directory.)
79. Type `sudo apt install git`. Because the command is sudo it will require you to enter your password (for the user account on which you are logged into the server). Simply hit enter for any prompts that arise during the installation process.

    Important: note that this command requires sudo. Installations often do. Some students get in the habit of always typing sudo before every command. **This is a bad habit!** Naturally we'll use sudo a lot early on when getting the server set up, but **generally speaking you should not use sudo**. The fact that certain commands require sudo is deliberately to protect you from screwing things up too badly. If a command requires sudo and you forget to type sudo, it will fail with a message telling you that it requires sudo, and then you can think carefully about whether or not you really need to execute that command.
80. Check that git is installed and working by entering the command `git --version` (most terminal programs have a `-v` or `--version` flag that simply prints back the version of the program).

Setting up and adding to your Git repository

Git is a tool that facilitates version control. Using Git you create **repositories** (or 'repos' for short) in which you will put your files and code. When connected subsequently to GitHub it serves as a backup of all of the work you've done.

81. Login to your server.
82. Make sure you are in your home directory (i.e., when you type `pwd`, you should be in `/home/your_user_name`). If you aren't there, then typing `cd` with no arguments should take you there.
83. Create a directory in your home directory called `CS_1337`
84. Change directories to be in the directory `CS_1337` that you just created.
85. To initialize a git repository in this directory, run the

    ```
    git init
    ```

command. You have now initialized a git repo in the directory `CS_1337`. This is currently just a local repository. Soon we'll connect it to the cloud.

86. The first file we're going to add to the repo is a README file. Open up a file in Vi called README.md using the following command (note that 'md' is short for 'markdown' which allows for us later to put more than just plain text in our README file):

    ```
    vi README.md
    ```

87. Following the instructions on Editing files in the terminal using vi, enter insert mode and type in the following (or something similar—you can always change it later):

    ```
    # Your_name's CS 1337 GitHub Repo

    This repo contains all of the work and assignments completed for CS
    1337, Intro to Computing Systems, taught by Dr. Paul Bodily at Idaho
    State University.
    ```

    Save the file and quit Vi.

88. Once you've added or modified files or folders in a folder containing a git repo, git will notice that the file exists inside the repo. But, git won't track the file unless you explicitly tell it to. Git only saves/manages changes to files that it tracks, so we'll need to send a command to confirm that yes, we want git to track our new file. Use the `git status` command to see which files git knows exist. You will see your `README.md` file listed under "Untracked files". What this basically says is, "Hey, we noticed you created a new file called `README.md`, but unless you use the 'git add' command we aren't going to do anything with it."

89. Add `README.md` to your git repo with the command

    ```
        git add README.md
    ```

    This command adds the file to a "staging environment"; it keeps track of all of the files whose changes *will* be **committed** when next you command git to commit changes (Surprise! Git records changes only when explicitly told to). Use the `git status` command again and you'll see that `README.md` is staged to be committed. **Note:** providing the add command with a directory name instead of a filename will recursively add all files in that directory. Be careful with this—you generally may not want to add everything to your repo!

90. When you commit, git stores a username and email with the commit so that it is known who is responsible for the changes. To set this username and email for all future commits, execute the two following commands with the strings in quotation marks replaced with your information:

    ```
        git config --global user.email "you@example.com"
        git config --global user.name "Your Name"
    ```

91. Time for your first commit! Run the command `git commit -m "Added README file"`. Every commit requires a message be added. **This message is more important than you realize**. The message serves as a reminder later on about what precisely differentiates this version of your repo from the last version. Without the message, you literally have to sift through the changes to try and remember. Nightmare! Practice now at leaving good commit messages!

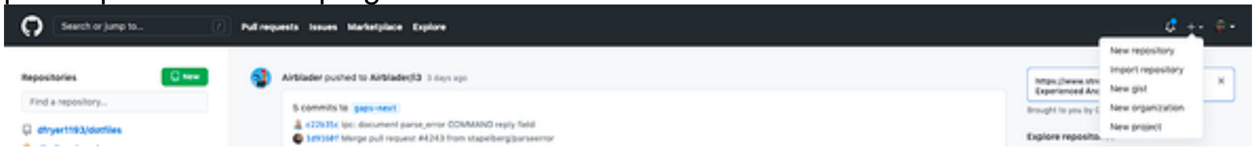
92. Use the `git status` command again and you'll see that there is now "nothing to commit".

93. **IMPORTANT FYI**: One should really use branching when using Git. By default, all of your changes will be made on the **main** or master branch. In industrial-scale applications, the main branch is the production version of the application. Say you want to make a new feature but are worried about making changes to the main project while developing the feature. In this case, Git has functionality to allow you to create a new *branch* from one of the versions in the main branch. After you're confident that the branched version works as desired, this can then be **merged** into the main branch. You will invariably learn this in future CS classes. For the sake of keeping things simple, because you will be working solo in this class, and because we're not concerned about any production version code—we will not do branching. But you should be aware that it exists and why it exists.

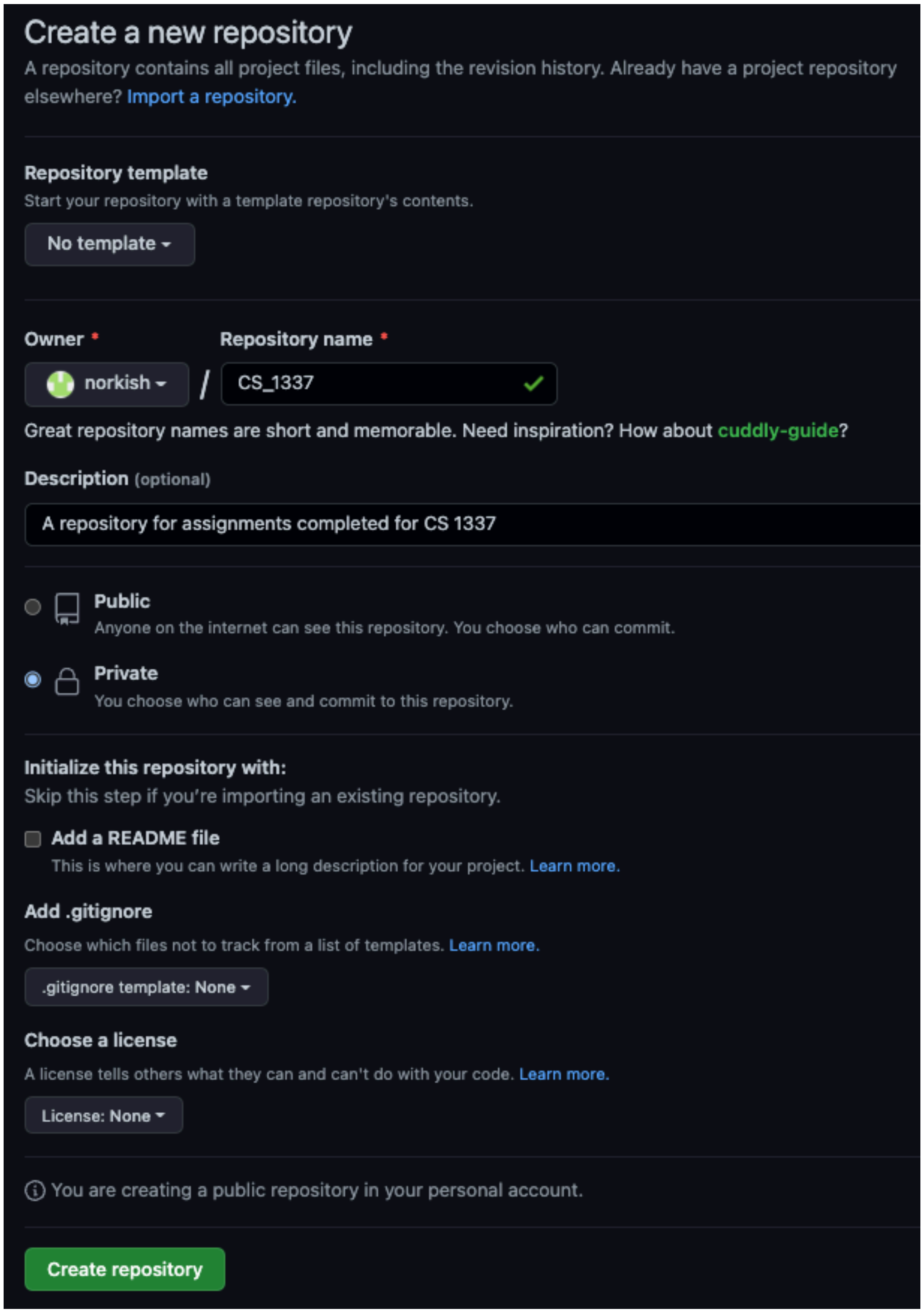
Connecting your Git repo to GitHub

By itself, Git allows you to keep track locally of changes to files in the repository. But the real power of git is in the ability to back up your repository to the cloud. This is useful for more than just keeping a copy of your data; it comes in handy when you have multiple programmers working on the same project and needing to merge their changes. For this class you'll be working solo on all of your assignments, but all of the commands you'll learn will carry over when you start working on group projects in other classes. **GitHub** is an extremely popular online platform for storing and interfacing with Git repositories. You are welcome to use an existing GitHub account or create a new one for this class.

94. Create and/or login to your GitHub account.
95. On the GitHub home page, find the "New repository" option under the "+" sign next to your profile picture in the top right corner of the navbar:



96. After clicking the button, GitHub will ask you to name your repo (e.g., "CS_1337") and provide a brief description. **You are required for this class to select a 'Private' repo so that your work stays your work.** You don't need to change anything else from the default repo settings.


Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Repository template

Start your repository with a template repository's contents.

No template

Owner * norkish / Repository name * CS_1337

Great repository names are short and memorable. Need inspiration? How about cuddly-guide?

Description (optional)

A repository for assignments completed for CS 1337

( ) Public — Anyone on the internet can see this repository. You choose who can commit.

(●) Private — You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

[ ] Add a README file — This is where you can write a long description for your project. Learn more.

Add .gitignore

Choose which files not to track from a list of templates. Learn more.

.gitignore template: None

Choose a license

A license tells others what they can and can't do with your code. Learn more.

License: None

You are creating a public repository in your personal account.

97. When you're done filling out the information, press the 'Create repository' button to make your new repo. Follow the '....or push an existing repository from the command line' section, which contains three commands you will execute on your server. We'll walk you through them here.
98. **First**, on your AWS server, push your existing repository to GitHub with the following command[2]. Be sure to replace the URL with the URL for the GitHub repo you just created. **For record-keeping purposes, we will ask you for your github repo URL** (of the form https://github.com/user_name/repo_name) **in the submission for this assignment**.

[2] Note that git commands can be executed from the directory in which you initialized the git repo or any subdirectory of that directory.

```
git remote add origin https://github.com/url/to/repo.git
```

Common bug: It is not uncommon for people to put the wrong URL and then when they try to push to their repo they run into an error. You can test that the URL is correct because it should work when opened in a browser. If your origin URL is wrong, you can change it using the command

```
git remote set-url origin https://github.com/url/to/repo.git
```

99. **Second**, by default the main branch in Git is named **master** (to see this execute `git branch`). On Oct 1, 2020, GitHub (not Git) announced that in an effort to abandon non-inclusive language, that GitHub "will use **main** as the default branch, instead of **master**." It is up to you what language you wish to use, but this tutorial will use **main** as the primary branch. To name the primary branch **main**, execute the command:

    ```
    git branch -M main
    ```

100. To be able to push your local repo to GitHub, **GitHub requires authentication**. As of August 2021, this requires the use of **personal access tokens** (**PAT**s) rather than passwords. Follow GitHub's instructions for generating a token (classic, NOT fine-grained) as provided here: https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token#creating-a-personal-access-token-classic.

    I would suggest for the Note something like "CS 1337 AWS Server". Every push to GitHub will require this token, so I would suggest setting an **expiration date sometime after the end of the semester** unless you want to be regularly generating new tokens. **Under "Select Scopes", select "repo".** Click "Generate token".

    Warning: Treat your tokens like passwords and keep them secret. **You will need this PAT every time you push to your server, so save it somewhere.** If you lose your PAT or you are having trouble, you can always delete the token and create a new one.
101. With your AWS Git repo now configured with a remote GitHub repository, you can push the repo to GitHub with the following command. **When prompted, use your PAT in place of your GitHub password**.

    ```
    git push -u origin main
    ```

    It will not show the password as you type it, but it is registering your keystrokes. Note: It can be a little tricky to paste in the Windows Putty terminal, which you'll need to do for this step when entering your PAT as the password In my experience, tapping the trackpad with two fingers works. I suggest practicing pasting into the terminal before being prompted for the password so you can actually see what is being pasted.
102. **You can now view your GitHub repo online at https://github.com/user_name/repo_name.** Since it is private, only those you share the repo with will be able to see it.
103. We need access to your GitHub repo to be able to check that you are successfully backing up your work to your repo. To provide us with access, from your GitHub repo page (i.e., the url referred to in the previous step), select "Settings" along the top menu bar, then under "Access"

in the lefthand menu, select "Collaborators." Click the "Add people" button and add the user "cs1337ta" to your repository.

Everyday Git commands

A common issue with using Git is that the version of the repo on your local system gets out of sync with the version of the repo that is in the cloud or that others are using. For this reason it is important to **always follow a specific order of instructions** in your everyday usage of Git (note that with the `add` and `commit` commands you will need to add additional parameters):

104. Stash your local changes (in case those local changes conflict with the remote version):
```
    git stash
```
105. Update the branch to the latest code (don't forget to use the PAT as the password)
```
    git pull
```
106. Merge your local changes into the latest code:
```
    git stash apply
```
107. Add, commit and push your changes
```
    git add [subdirs or files to add or with changes to commit]
    git status
    git commit -m "Helpful message describing updates"
    git push -u origin main
```

Note that 1) adding a directory automatically adds everything in that directory; 2) adding is required for any file with changes since the previous commit; 3) issue these commands from the directory containing subdirectory or files with changes to commit; 4) I assume that `main` is the name of your primary branch, and you may need to change that last command if yours is something different.

In this class, since you're working solo, you should never have to merge local changes into the latest code (because local changes should always **be** the latest code). But **it is a good habit to get into to always stash and pull before you add, commit, and push** so that when you begin working with others that you don't try to push your changes to a version of the code that is more up-to-date than the one you were working from. This can lead to overwriting changes that others have made, and although it can be fixed, it can take a lot of time and is no fun!! ***You should always start a coding session by pulling the latest code and you should always end a coding session with the above sequence of commands.***

Getting started coding on the server

The main purpose of this tutorial has been to set up a common coding environment for all students on a remote server. (A secondary purpose is to teach you SSH and Linux commandline.) Within this coding environment we will be exploring the C language and how it gets compiled into assembly and machine languages.

High-level, low-level, interpreted, and compiled languages

In computer science we talk about **high-level languages** and **low-level languages**. You have previously learned **Python** (a high-level language). In this course you will learn **C** (another high-level language). As the purpose of this course is to crack open the proverbial hood and look inside, we will also look at low-level languages like machine and assembly languages with the express purpose of helping you understand how they work.

Low-level languages are sometimes referred to as **machine languages** or **assembly languages**. **Machine language** is the encoding of instructions in binary (0's and 1's) so that they can be directly executed by the computer. **Assembly language** uses a slightly more readable format to refer to the low level instructions. Loosely speaking, computers can only execute programs written in low-level languages. To be exact, computers can actually only execute programs written in machine language. Thus, programs written in a high-level language (and even those in assembly language) have to be processed before they can run. This extra processing takes some time, which is a small disadvantage of high-level languages. However, the advantages to high-level languages are enormous.

Advantages of high-level languages (this may be on the midterm)

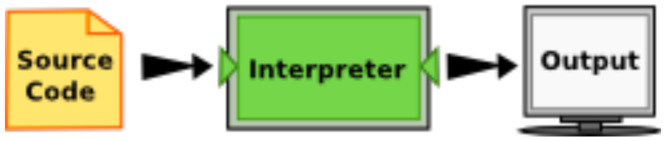
1. **It is much easier to program in a high-level language**. Programs written in a high-level language take less time to write, they are shorter and easier to read, and they are more likely to be correct.
2. **High-level languages are portable**, meaning that they can run on different kinds of computers with few or no modifications. Low-level programs can run on only one kind of computer and have to be rewritten to run on another.

Due to these advantages, almost all programs are written in high-level languages. Low-level languages are used only for specialized applications (e.g., they're used a lot in some cybersecurity applications), but understanding them will make you a more efficient and effective programmer in general.

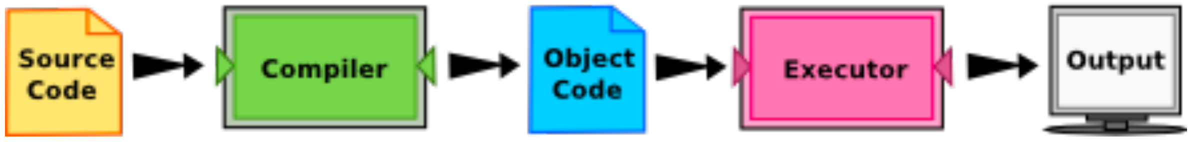
Two ways to process high-level programs (this may also be on the midterm)

Two kinds of programs process high-level languages into low-level languages: interpreters and compilers (shown in green below). High-level languages are usually categorized as either **interpreted languages or compiled languages** depending on which process is most commonly used.

1. An **interpreter** reads a high-level program and executes it. It processes the program a little at a time, alternately reading lines and performing computations.



2. A **compiler** reads the program and translates it completely before the program starts executing. In this case, the high-level program is called the source code, and the translated program is called the object code or the executable. Once a program is compiled, you can execute it repeatedly without further translation.



Many modern languages use both processes. They are first compiled into a lower level language, called byte code, and then interpreted by a program called a virtual machine. Python uses both processes, but because of the way most programmers interact with it, it is usually considered an **interpreted** language. C is only a **compiled** language.

In this course, we'll essentially start with C and work our way backward to assembly and then to machine code. I hope this excites you. *By the end of this course you will have uncovered the magic and mystery of how a computer works!*

Installing GCC on your server

GCC stands for GNU Compiler Collections which is the compiler (i.e., "green box") used to compile programs in the C and C++ languages. Since we're learning C this semester, you'll need to have GCC installed on your server.

108. Login to your server. (You can install the C compiler from any directory.)
109. Type `sudo apt install gcc`. Because the command is sudo it will require you to enter your password (for the user account on which you are logged into the server). Simply hit enter for any prompts that arise during the installation process.
110. Check that gcc is installed and working by entering the command `gcc -v`.

Installing GDB on your server

GDB is the GNU Debugger that runs on Unix-like systems and works for many programming languages. We will use it extensively later on in the semester when we learn assembly, but I encourage you to learn and use it in Labs 2 and 3 to help you debug your C programs.

111. Login to your server. (You can install gdb from any directory.)

112. Type `sudo apt install gdb`. Because the command is sudo it will require you to enter your password (for the user account on which you are logged into the server). Simply hit enter for any prompts that arise during the installation process.
113. Check that gdb is installed and working by entering the command `gdb -v`.

Your first assignment: hello_world.c

114. Login to your server.
115. In your `CS_1337` directory, create a directory called `assignments`
116. Change directories to be in the directory `/home/your_user_name/CS_1337/assignments`
117. Create a directory called `lab_1`
118. Change directories to be in the directory `/home/your_user_name/CS_1337/assignments/lab_1`
119. Each programming language has conventions, i.e., norms that are not mandatory for programs to run, but styles which programmers who use that language generally expect will be followed. Often times those conventions are merely historical, but sometimes they have good practical motivations. **You should generally follow the conventions for the language you are using as regards filenames, commenting, variable and function names, etc.** In the C language, it is convention that **filenames are to be all lowercase with underscores used to separate words**. The suffix for a C source file is ".c". Following this pattern create a file in Vi called hello_world.c using the following command:

    ```
    vi hello_world.c
    ```

120. Following the instructions on Editing files in the terminal using Vi, enter insert mode and type in the Hello World program as found here: https://www.tutorialspoint.com/cprogramming/index.htm. Save the file and quit Vi.
121. Type `ls` to get a directory listing. Do you see your source file?
122. Compile your source code with the command `gcc hello_world.c`
123. Type `ls` to get a directory listing. Do you see your binary object file?
124. Just as "`..`" is a shortcut for a directory's parent directory, "`.`" is a shortcut for a directory itself. Why would we need this? Some commands can be executed from anywhere in the file directory structure (e.g., we just ran `gcc` which is not in the current directory). When we need to distinguish that we're referring to a command or file in the current directory, we use this. Execute your binary object file with the command

    ```
    ./a.out
    ```

    You'll learn later how to specify the name for your object file.
125. Don't forget to add, commit, and push your changes! Navigate back to your CS_1337 directory on your server and let's add your `assignments` directory (and everything in it) to your repository:

    ```
    git add assignments
    ```

    Let's check what got added.

```
git status
```

You should see that it has recursively added your lab_1 directory and its contents. Now we need to commit these changes.

```
git commit -m "Added Lab 1 Hello World in C"
```

Now push to to GitHub!

```
git push -u origin main
```

Again, this last command assumes your primary branch is called `main`. You'll need again to type in your GitHub username and your PAT. You should now be able to see your Lab 1 code in your GitHub repo in a web browser.

A quick GDB tutorial

126. Before we finish, let's see how we use gdb to debug programs in C. You'll first need to navigate back to the lab_1 directory on your server.
    1. To run C programs in gdb requires compiling with the -g flag. Recompile your `hello_world.c` with the -g flag with the command

       ```
       gcc -g hello_world.c
       ```
    2. Next, launch gdb on your binary object file with the command

       ```
       gdb a.out
       ```
    3. In gdb, you'll see the program has its own command line. Set a breakpoint at line 5 (in your `hello_world.c` file) with the command

       ```
       break 5
       ```
    4. Now run the program with the command `run`.
    5. Our `hello_world.c` program doesn't have any variables, so there's not much to look at, but take a minute to visit https://u.osu.edu/cstutorials/2018/09/28/how-to-debug-c-program-using-gdb-in-6-simple-steps/ to see how, in more complex programs, you would print variables. Use this tutorial to figure out how to `step` your way (or `continue`) through the rest of your `hello_world.c` program.
127. For now that's all! You don't need to do anything else. The TA will login to your server using the credentials you sent me and check that you have done the assignment correctly. **Nice job!**

A Note About IDEs

128. You are welcome to use an interactive development environment (IDE) on your local machine for editing files for this class if you prefer. Visual Studio Code has a Remote - SSH package that will allow you to connect over SSH. You can then install the C/C++ package on both the host and remote server to have syntax highlighting and error messages. More information on this can be found here and here. You can also complete assignments in a local IDE (e.g.,

CLion or VSCode) to get help with debugging your C code and then copy the completed code to your server (be sure that it runs on the server before final submission!). All of that said, knowing how to edit files using a remote file editor is an important skill for anyone who spends time working on remote machines. It may not always be practical or possible to configure a local IDE every time you need to modify files on a remote machine.

When the Free Tier period expires

When the 12-month Free Tier period with AWS is about to expire you should receive an email notifying you. Assuming you have **terminated all active resources**, you will not be billed anything. You can optionally close your account. A succinct summary of options and instructions can be found here: https://aws.amazon.com/premiumsupport/knowledge-center/free-tier-expiring/.

When the time comes that you want eliminate the server and everything on the server, you will need to "Terminate" the server. Because you have enabled "Termination protection," you will see that the option to "Terminate" is disabled in the standard menus. To terminate, simply right click on the server and adjust settings to **disable "Termination protection" prior to then Terminating the instance.**