Server-side Programming

Zachary Krepelka

Wednesday, January 17th, 2024

Contents

List of Listings

List of Figures

Introduction

This semester I am taking a classed called 'Server-side Programming' at Saint Francis University. The purpose of this document is to journal my assignments as per the requirements of the class:

> Provide technical documentation for each of the assignments thus far detailing the installation/configuration process with text descriptions and screenshots. The document should be able to be taken by a semi-technical individual to complete each assignment without assistance. —Dr. Slonka

Organization

Each assignment is documented within its own section of this document. To provide context, the assignment is reproduced verbatim. Thereafter, the remaining content within each section is subdivided into subsections. They may include, but are not limited to, the following.

- *Background.* This subsection details preliminary research and elaborates on keywords within the assignment. *Context* is provided here.
- *Where I Stand.* Here I remark if I have had prior experience with any aspect of the assignment. This information can aid the reader; it helps to know where the author stands.
- *Steps.* These subsections detail the steps required to complete the assignment. They are numbered as necessary.
- *Resources.* Here I list the resources I accessed to research the assignment. This subsection entails a bulleted list of URLs.

Assumptions

We make the following assumptions about the reader.

- The reader is working on a Linux machine. We will be using a Ubuntu virtual machine with Oracle VM VirtualBox as our hypervisor. It is assumed that the operating system is already installed and set up. Instructions can be found online.
- The reader has rudimentary competency at the command line. The reader knows how to open a terminal and issue basic commands like `ls` or `cd`.

Typographical Conventions

Screenshots are reserved for graphical user interfaces. I don't see any good reason to provide screenshots of terminal commands since they are entirely text. Instead, I will prefer to provide shell commands in a textual format. They will appear as follows. To clearly distinguish input from output, the result of the command is indented whereas the command itself is left justified and prefixed with a dollar sign.

```
$ grep author doc.tex

        \author{Zachary Krepelka}
```

In other circumstances, however, the output is omitted entirely, like this:

```
$ sed 's/\(Zachary\) \(Krepelka\)/\1 Paul \2/' doc.tex
```

It is also possible for shell commands to appear inline, like this: `less doc.tex`. Listing 1 is included to provide an example of how code will appear throughout this document. Take notice the numbers on the side of Listing 1. These can be used to identify and discuss specific lines of code. Longer programs will be built up incrementally with commentary dispersed throughout, and a final listing will be provided thereafter.

```
 1  #!/usr/bin/env ruby
 2
 3  # FILENAME: totatives.rb
 4  # AUTHOR: Zachary Krepelka
 5  # DATE: Wednesday, November 29th, 2023
 6
 7  class Integer
 8      def totatives
 9          fail unless positive?
10          1.upto(self).select do |i|
11              gcd(i) == 1
12          end
13      end
14  end
```

Listing 1: a Ruby program to compute the set of totatives of a number.

0 Git

Our first assignment asked us to set up a GitHub repo for this class.

1. Create a GitHub account (if you don't already have one).
2. Create a directory on your Linux VM for your CPSC222 assignments.
3. Create a repo in that folder.
4. Create a README file in that folder that contains your name.
5. Push your changes to the repo.
6. Submit your repo URL so that the professor can clone your repo and see your files.

0.1 Background

Git is a version control system that tracks changes in computer files. It was originally authored by Linus Torvalds, the creator of the Linux kernel. GitHub, which is built on top of Git, is an online platform for software developers to store, manage, share, and collaborate on code.

0.2 Where I Stand

I already had a GitHub account prior to coming into this class, and I was already familiar with the basic Git terminal commands. Because I don't remember the specific steps involved in signing up, I will be creating a burner account on GitHub with a temporary email for the purpose of writing this documentation.

0.3 Steps

We work through the steps as specified.

0.3.1 Account Creation

Step #1 of this assignment asks us to create an account with GitHub.

1. Begin by pointing your web browser to `https://github.com/signup`. You will be met with a page as shown in Figure 1 that asks you to enter your new account credentials, viz., email, password, username, and email preferences. You will also be asked to verify that you are not a robot by preforming a small puzzle. To proceed, click on the text fields indicated by the pink arrows. After you have entered your email and new password, click 'Create Account'. Be sure to create a strong password. Write down your account credentials for save keeping.

2. Next you will be brought to a page as shown in Figure 2 that asks you to enter a verification code sent to your email. Access your email, retrieve the code, type it in, and continue.

3. The next pages asks you to personalize your account by supplying information about your demographics, e.g., student, teacher, N/A. For our purposes, we will be skipping the personalization by clicking the 'Skip personalization' button. However, if you are a student, you should fill this section out because GitHub will provide free educational access to certain tools and technologies. See Figure 3.

4. You are now at your GitHub dashboard, as shown in Figure 4. From here, you'll want to click on the big, bright green button in the top left-hand corner of the screen that says 'Create repository.' You will be taken to a page as shown in Figure 5. You'll wnat to give your repository a name and description. Then choose whether you want a private or public repository. We will choose to make a public repository. You can ignore everything else. We will add a README file later. When everything has been filled out, click 'Create repository.'

5. After creating your repository, you will be brought to a quick setup page as shown in Figure 6. There will be a blue box containing your new GitHub repositorie's URl. Copy this to your clipboard and save it for later. You will see that there are instructions for creating a repositiory or pushing an existing repository from the command line. We will walk though this in the next subsection. At this point, you can savely close the window.

0.3.2 Directory Creation

Step #2 of this assignment asks us to create a directory to store our assignments for this class. Begin by opening a terminal on your Linux machine. This can be done by pressing `CTRL-ALT-T` on your keyboard, but it will also suffice to find the terminal by searching through the available software. This can be done by clicking the grid icon in the bottom left-hand corner of the screen of the Ubuntu desktop as shown in Figure 7. Once the terminal command is open, issue the following command.

```
$ cd Documents
```

This will change our current working directory to the documents folder. This is where we will be storing our code, but you can choose another location if you wish. The command should return silently, indicating success. Next you will want to make a new directory as instructed in the assignment. Do this by issuing the following command.

```
$ mkdir ssp; cd ssp
```

The `mkdir` command will create a new subdirectory in the user's current working directory (or elsewhere if specified). Its first positional argument is the name of the new directory, which we have chosen to be 'ssp' as an acronym for 'server-side programming.' The semicolon is a *command separator*.

0.3.3 Local Repo Creation

Step #3 of this assignment asks us to create a local Git repository in the directory we created in step #2. Recall that Git and GitHub are distinct entities as previously discussed. Our new Git repo will be local to our machine. First ensure that Git is installed on this system by typing `which git`. This command will point to the Git's executable binary if it exists. If the command returns nothing, you will need to install Git using your installation method of choice. Since we are using Ubuntu, we will use its default package manager by typing the following command.

```
$ sudo apt install git -y
```

The `-y` flag instructs the `apt` package manager to install the software without user intervention. Without it, the user would have to type 'y' for "yes, I want to install this software." After installation, the terminal will be cluttered with installation information, so you can type `clear` to clear your terminal. You will also need to set up git. Tell Git your name like this.

```
$ git config --global user.name "Joe Smith"
```

Tell Git your email like this.

```
$ git config --global user.email \
"162062323+throw-away-account-123@users.noreply.github.com"
```

Here the backslash is used for line continuation. You likely won't want to use your personal email when uploading changes to GitHub. For this reason, GitHub provides a special noreply email to hide your real email. To figure out yours, point your web browser to `github.com/settings/emails`, first ensuring that you are signed in. There you will find your new noreply email under the 'Keep my email addresses private' checkbox as shown in Figure **??**. You'll want to ensure that that check mark box is selected. To ensure the process worked, you can type `cat ~/.gitconfig` to preview your Git dotfile. It should look similar to mine.

```
[user]
        name = Joe Smith
        email = 162062323+throw-away-account-123@users.norepl...
```

Now we are ready to create the local Git repository. Once again confirm that you are in the directory you wish to initialize a repo in by typing `pwd`. Then type `git init` to create a Git repository in the current working directory. Typing `ls -a` will confirm that a hidden Git directory is present. Git will use this directory to keep track of changes in your project.

0.3.4 Creating a README

Step #4 of this assignment asks us to create a README file. Create a README file by typing `touch REAME.md`. Then edit the file using your text editor of choice. The nano text editor is a good choice for beginners, and you can use it by typing `nano README.md`[1]. Add the following lines.

```
# Server-side Programming
My name is Joe Smith.
```

The README file uses a markup language called Markdown to specify the appearance and content of the document. The README file will be rendered in the web browser when others visit your GitHub page, but Markdown is also designed to be readable from the terminal as plaintext. We will not go into detail here on how to write Markdown. The reader will find plenty of tutorials on the internet.

I will not explain the Git life cycle here in detail. For now, it will suffice to remark that Git works by *staging* changes in files. Thereafter, the changes can be finalized by *committing* them. By doing this, the user is able to track the change history of a project. The commits serve as way point markers in a project's history. Git does a lot more than this, but this is all we need to know for now.

To stage your new file for a commit, type `git add README.md`. Follow this by `git commit -m 'first commit'` to commit your changes. The `-m` flag specifies a message describing the commit. Be mindful when authoring commit messages. A Google search for 'commit message guidelines' is recommended.

Git's default branch (I will not explain branches here) used to be called 'master.' For political correctness, GitHub encourages you to change your default branch name to 'main.' This can be done by typing `git branch -M main`. You can call the main branch whatever you want.

You have successfully created a local Git repository.

0.3.5 Uploading to the Cloud

Now we complete Step #5 of the assignment. To share your code online with other people, you can link your local Git repo with an online platform like GitHub. Issue this command.

[1]The touch command is not necessary. If supplied with a name on startup, the text editor will automatically create the file when it is saved for the first time.

```
git remote add origin [YOUR GITHUB REPO URL HERE FROM STEP 1]
```

Mine looks like this.

```
https://github.com/throw-away-account-123/throw-away-repo.git
```

This command only establishes a relationship. To actually store your code in the remote repository, you'll want to push the changes to the cloud, like this.

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

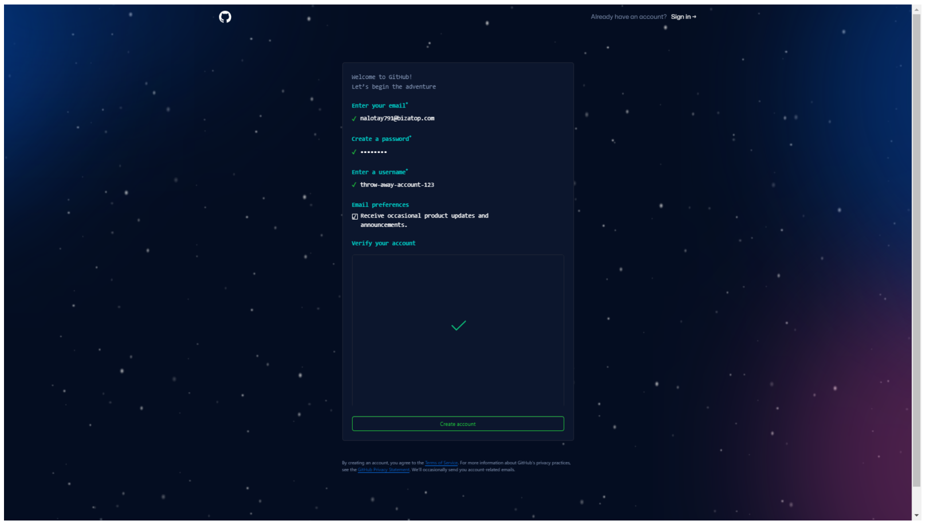
But we're not done yet. You have to supply your username and password for the remote repository. GitHub does not let you use your account password, so you have to create what is called an *access token*. Point your web browser to `https://github.com/settings/tokens/new` and enter your account password if required. You will be brought to a page as shown in Figure 8. Enter a description for your token under the note text field. Then check mark the appropriate boxes. It is often easier to just check mark them all if you are unsure. Once you have done this, click the bright green 'Generate token' button at the bottom of the page. The token will then be available for you to copy as shown in Figure 9. Write it down for save keeping. You will have to periodically regenerate your access token. Use this as your password for the `git push` command.

0.4 Resources

GitHub's quick start guide is a good place to start.

`https://docs.github.com/en/get-started/start-your-journey`

It will walk you through the basics of Git and GitHub. You will be asked to create a 'hello-world' repository.


Already have an account? Sign in →

Welcome to GitHub!
Let's begin the adventure

Enter your email*
✓ nalotay791@bizatop.com

Create a password*
✓ ••••••••

Enter a username*
✓ throw-away-account-123

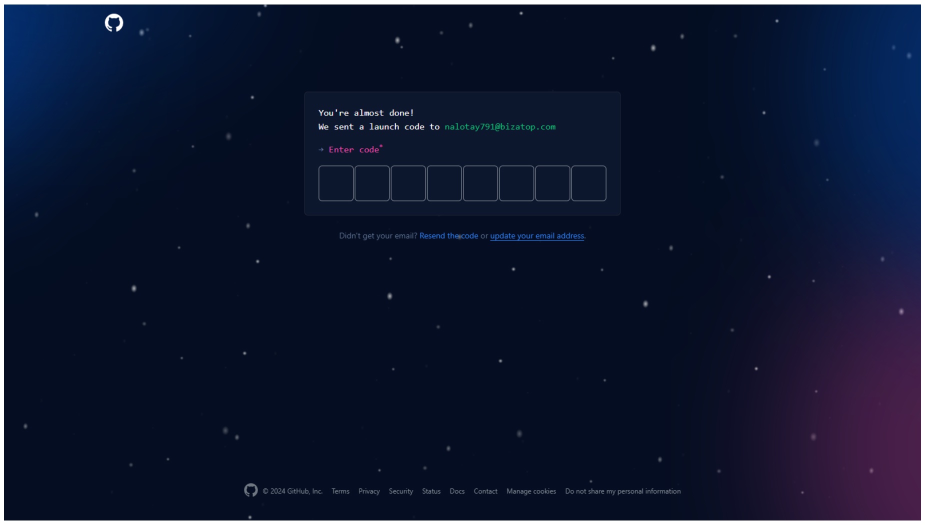
Email preferences
☑ Receive occasional product updates and announcements.

Verify your account

Create account

By creating an account, you agree to the Terms of Service. For more information about GitHub's privacy practices, see the GitHub Privacy Statement. We'll occasionally send you account-related emails.

Figure 1: GitHub's Signup Page.

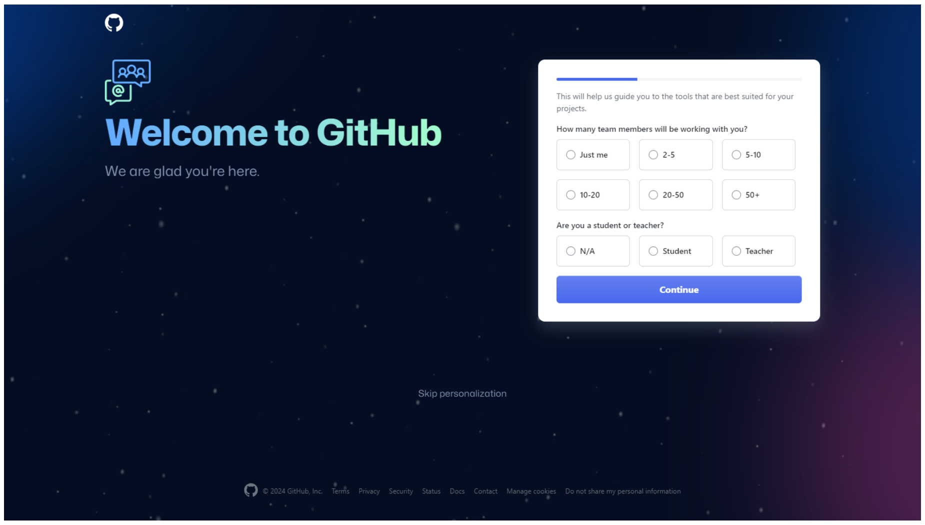

You're almost done!
We sent a launch code to nalotay791@bizatop.com

→ Enter code*

Didn't get your email? Resend the code or update your email address.

© 2024 GitHub, Inc. Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information

Figure 2: GitHub's Email Verification Page.



Welcome to GitHub

We are glad you're here.

This will help us guide you to the tools that are best suited for your projects.

How many team members will be working with you?

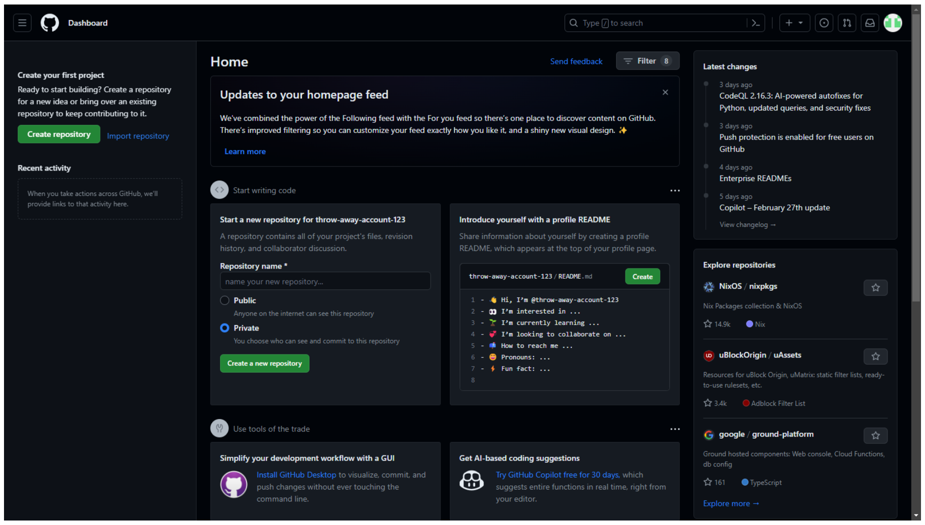
- Just me
- 2-5
- 5-10
- 10-20
- 20-50
- 50+

Are you a student or teacher?

- N/A
- Student
- Teacher

Continue
Skip personalization
© 2024 GitHub, Inc. Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information

Figure 3: GitHub's Personalization Page.



Create your first project

Ready to start building? Create a repository for a new idea or bring over an existing repository to keep contributing to it.

Recent activity

When you take actions across GitHub, we'll provide links to that activity here.

Home

Updates to your homepage feed

We've combined the power of the Following feed with the For you feed so there's one place to discover content on GitHub. There's improved filtering so you can customize your feed exactly how you like it, and a shiny new visual design. ✨

Start a new repository for throw-away-account-123

A repository contains all of your project's files, revision history, and collaborator discussion.

Repository name *

name your new repository...

Public [✓ Private]

Public: Anyone on the internet can see this repository

Private: You choose who can see and commit to this repository

Introduce yourself with a profile README

Share information about yourself by creating a profile README, which appears at the top of your profile page.

throw-away-account-123 / README.md

```
- 👋 Hi, I’m @throw-away-account-123
- 👀 I’m interested in ...
- 🌱 I’m currently learning ...
- 💞️ I’m looking to collaborate on ...
- 📫 How to reach me ...
- 😄 Pronouns: ...
- ⚡ Fun fact: ...
```

Simplify your development workflow with a GUI

Install GitHub Desktop to visualize, commit, and push changes without ever touching the command line.

Get AI-based coding suggestions

Try GitHub Copilot free for 30 days, which suggests entire functions in real time, right from your editor.

Latest changes

- 3 days ago: CodeQL 2.16.3: AI-powered autofixes for Python, updated queries, and security fixes
- 3 days ago: Push protection is enabled for free users on GitHub
- 4 days ago: Enterprise READMEs
- 5 days ago: Copilot – February 27th update

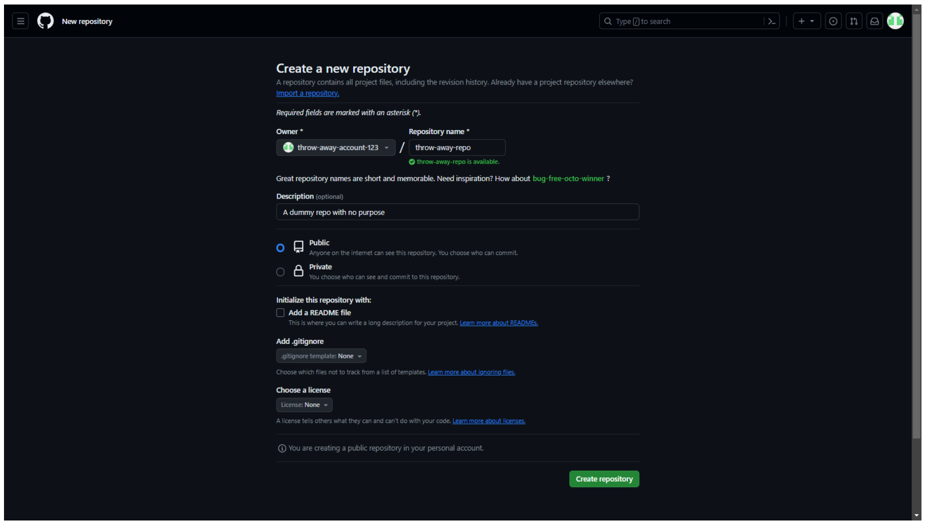
Explore repositories

NixOS / nixpkgs
Nix Packages collection & NixOS
☆ 14.9k ● Nix

uBlockOrigin / uAssets
Resources for uBlock Origin, uMatrix: static filter lists, ready-to-use rulesets, etc.
☆ 3.4k ● Adblock Filter List

google / ground-platform
Ground hosted components: Web console, Cloud Functions, db config
☆ 161 ● TypeScript

Figure 4: The GitHub Dashboard.



Create a new repository

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

Required fields are marked with an asterisk (*).

Owner *: throw-away-account-123 / **Repository name ***: throw-away-repo

throw-away-repo is available.

Great repository names are short and memorable. Need inspiration? How about bug-free-octo-winner ?

Description (optional): A dummy repo with no purpose

[✓ **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:

☐ **Add a README file**
This is where you can write a long description for your project. Learn more about READMEs.

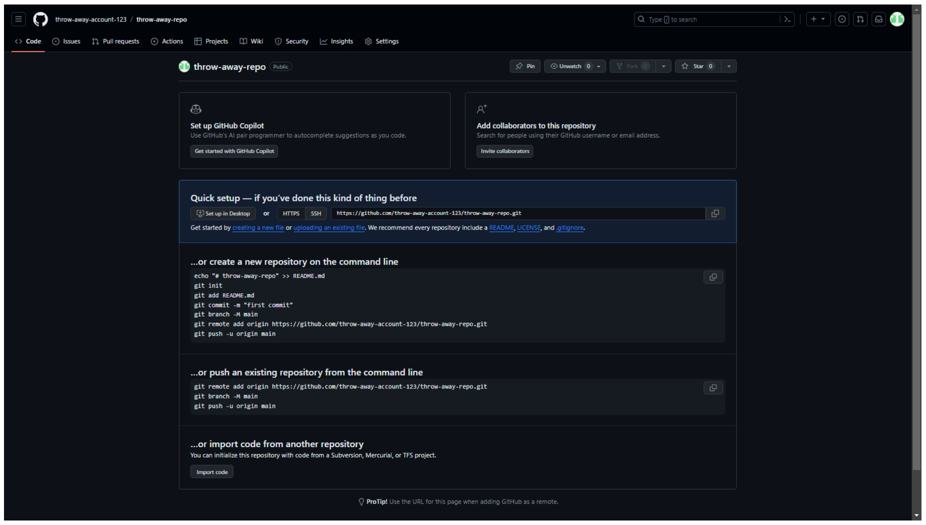
Add .gitignore
.gitignore template: **None**
Choose which files not to track from a list of templates. Learn more about ignoring files.

Choose a license
License: **None**
A license tells others what they can and can't do with your code. Learn more about licenses.

You are creating a public repository in your personal account.

Create repository

Figure 5: GitHub's Repository Creation Page.



Set up GitHub Copilot
Use GitHub's AI pair programmer to autocomplete suggestions as you code.
Get started with GitHub Copilot

Add collaborators to this repository
Search for people using their GitHub username or email address.
Invite collaborators

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH `https://github.com/throw-away-account-123/throw-away-repo.git`

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# throw-away-repo" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/throw-away-account-123/throw-away-repo.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/throw-away-account-123/throw-away-repo.git
git branch -M main
git push -u origin main
```

...or import code from another repository
You can initialize this repository with code from a Subversion, Mercurial, or TFS project.
Import code

ProTip! Use the URL for this page when adding GitHub as a remote.

Figure 6: GitHub's Quick Setup Page for a New Repository.

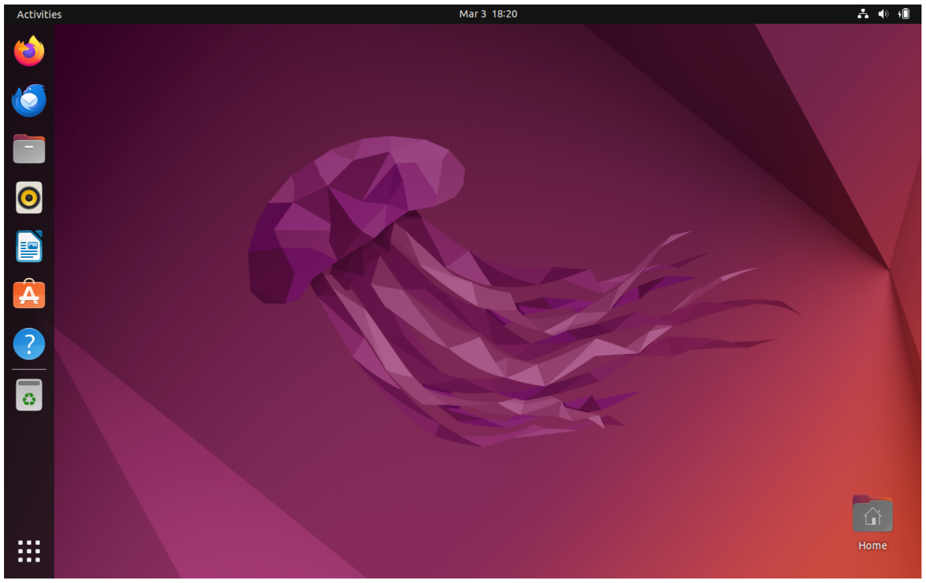
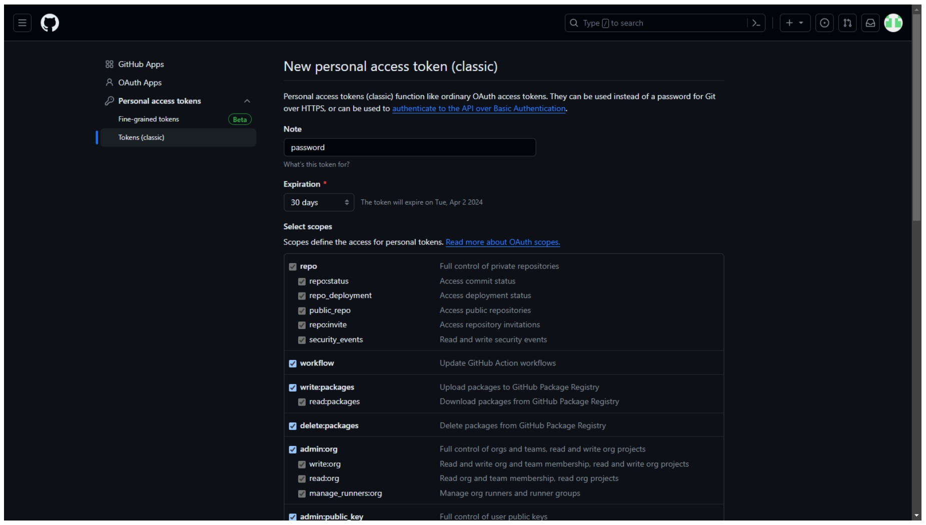


Figure 7: The Ubuntu Desktop.



New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Note

password

What's this token for?

Expiration *

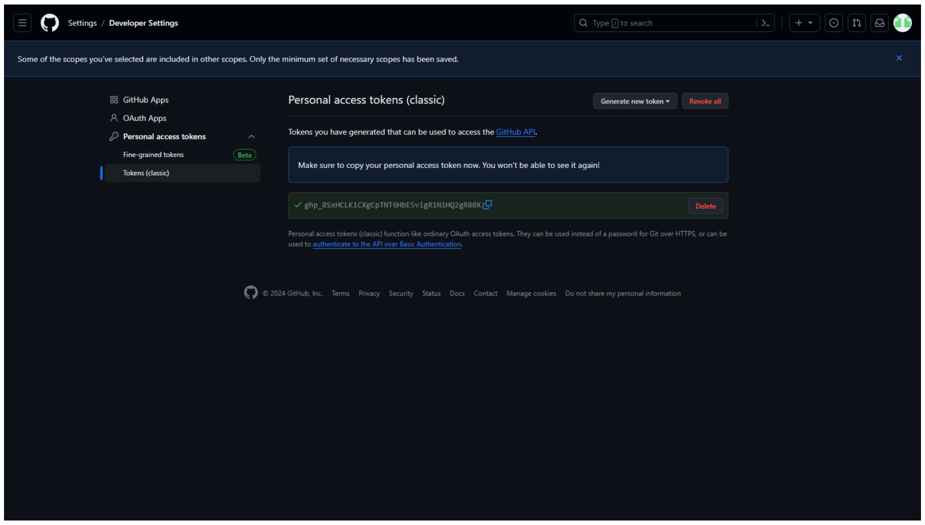
30 days — The token will expire on Tue, Apr 2 2024

Select scopes

Scopes define the access for personal tokens. Read more about OAuth scopes.

✓ **repo**	Full control of private repositories
✓ repo:status	Access commit status
✓ repo_deployment	Access deployment status
✓ public_repo	Access public repositories
✓ repo:invite	Access repository invitations
✓ security_events	Read and write security events
✓ **workflow**	Update GitHub Action workflows
✓ **write:packages**	Upload packages to GitHub Package Registry
✓ read:packages	Download packages from GitHub Package Registry
✓ **delete:packages**	Delete packages from GitHub Package Registry
✓ **admin:org**	Full control of orgs and teams, read and write org projects
✓ write:org	Read and write org and team membership, read and write org projects
✓ read:org	Read org and team membership, read org projects
✓ manage_runners:org	Manage org runners and runner groups
✓ **admin:public_key**	Full control of user public keys

Figure 8: GitHub's Personal Access Token Creation Page.



Settings / Developer Settings
Type / to search
GitHub Apps
OAuth Apps
Personal access tokens
Fine-grained tokens
Beta
Tokens (classic)
Generate new token
Revoke all
Delete
© 2024 GitHub, Inc. Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information

Some of the scopes you've selected are included in other scopes. Only the minimum set of necessary scopes has been saved.

Personal access tokens (classic)

Tokens you have generated that can be used to access the GitHub API.

Make sure to copy your personal access token now. You won't be able to see it again!

```
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
```

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Figure 9: An Example of a Personal Access Token.

1 SNMP sudo usage

Our first non-trivial assignment follows.

1. Write a Perl script that produces the number of successful sudo sessions on your host. The output shall only be a number without a trailing newline.
2. Make this script's results available via SNMP.

Add your script and snmp config file to your git repo and submit your repo URL and the command to prove that you can read the result via SNMP.

1.1 Background

Three words stand out: SNMP, Perl, and sudo. Let's discuss each.

1.1.1 SNMP

It stands for simple network management protocol. In general, a communication protocol is a set of rules used to coordinate computers so they know how to interact with each other. An important point is that a protocol is not a piece of software. Rather, a protocol is a specification published in a formal document, put together by a committee of people. Be that as it may, a communication protocol needs to be implemented in software to be useful. For example, one should distinguish between the Secure Shell protocol, which is a set of rules, and the software implementing that protocol, like `openssh-client` and `openssh-server` on Linux.

In this assignment, we will be using the simple network management protocol, which is a networking protocol for monitoring and managing devices on a network[2]. In this explanation, I will focus on the monitoring aspect. A standard use case is to access information about a remote computer, e.g., the speed of a fan, the temperature of a CPU, the host name of the device, etc. The SNMP protocol structures this information in a tree database called a *management information base* (MIB). It can be likened to a file tree. Because the information that SNMP monitors and manages often pertains to proprietary hardware, SNMP can be expanded with third-party MIB files. For copyright reasons, these are usually not included by default on Linux systems having SNMP software pre-installed. The individual nodes in a MIB tree are identified and accessed with Object Identifiers, or OIDs for short.

1.1.2 Perl

The SNMP protocol can be extended to expose the results of a script. In this assignment, we will be writing a Perl script to use in conjunction with SNMP. Perl is a scripting language with close ties to the Unix family of tools, e.g., sed, AWK, sh, and C. It's well known for its text processing capabilities and powerful regular expression engine. It was created in 1987 by Larry Wall.

[2] `https://askubuntu.com/a/252994`

1.1.3 Sudo Session

Linux permissions work in part by granting the user temporary access to elevated privileges only when they are necessary[3]. This is done by prefixing a command with the command `sudo`, for 'super user do.' The duration of time that the user has elevated privileges is called the sudo session. That way, commands that require sudo privileges can be issued in succession without adding sudo to every one—imagine reentering your password every time! A quick Google search tells us that a standard sudo session lasts for 5 minutes.

1.2 Where I Stand

I have some limited prior experience with Perl, I didn't know anything about SNMP before this assignment.

1.3 Steps

We address the two components of the assignment separately.

1.3.1 Writing the Perl Script

To write the script, we will first need to know where to find information about sudo usage. A few Google searches will tell us that we can find the information we need in `/var/log/auth.log`[4]. This is a log file containing information about *authorization* attempts on the system. A careful study of this file will reveal that sudo sessions are recorded on lines containing the strings `sudo:session` and `opened`. Hence, our script must count the occurrences of lines in `/var/log/auth.log` containing both `sudo:session` and `opened`. As a preliminary exercise, it is not hard to cook up a standard shell script for this purpose. In fact, it can be done quite succinctly as shown in Listing 2. On line seven, we first search for the string 'sudo:session' in our `auth.log` file using grep, a command-line program whose name is an acronym for 'global regular expression print.' Because grep deals with regular expressions, the user must be careful when searching for literal strings. If the string should happen to contain a metacharacter, the search will not be interpreted literally. In more complicated cases of literal string matching, prefer to use the `-F` flag to *fix* the string as a literal. For us, the strings are simple enough that this does not matter. After searching for the 'sudo:session' string, we search for the 'opened' string in lines already containing the 'sudo:session' string. This is accomplished by *piping* the first search command into the second. Piping is the process of redirecting the output of one command as input into another command; it is signified by the vertical bar. Hence, the second grep command searches within the output of the first grep command. In effect, this captures lines in the original file containing both strings. The output of these two commands are then piped into the word

[3]`https://en.wikipedia.org/wiki/Principle_of_least_privilege`
[4]https://unix.stackexchange.com/q/167935

count command `wc` with the flag `-l` to determine the number of lines containing both strings. Finally, that number result is piped into the translate command `tr` with the `-d` command to *delete* the trailing newline.

```sh
#!/usr/bin/env sh

# FILENAME: count-sudo-sessions.sh
# AUTHOR: Zachary Krepelka
# DATE: Monday, January 15th, 2024

grep 'sudo:session' /var/log/auth.log |
grep opened | wc -l | tr -d '\n'
```

Listing 2: a shell script to count the number of sudo sessions.

We were asked to write the script in Perl. I initially wrote the shell script at the command line just to process my thoughts. Now that we understand what we need to do, writing the Perl script is just a matter of translation. Listing 3 shows our desired Perl script. Let's walk through it line-by-line.

```
#!/usr/bin/env perl

# FILENAME: count-sudo-sessions.pl
# AUTHOR: Zachary Krepelka
# DATE: Monday, January 15th, 2024

open my $fh, '<', '/var/log/auth.log' or die $1;

my $log = do { local $/; <$fh> };

close $fh;

my $sudo_session_count = () = $log =~ m/sudo:session.*opened/g;

        # We put the regex match into
        # list context and then into
        # scalar context to get the
        # number of matches.

print $sudo_session_count;
```

Listing 3: a Perl script to count the number of sudo sessions.

2 Python API

The assignment follows.

Write an unprivileged Python script that exposes an HTTP API

- Your script should respond to HTTP POST requests
- Your script should only respond with data given the proper username/password (test/abcABC123)
- Your script should have two methods, users and groups, that provide a list of your system's users and groups
- Methods are to be accessed via `http://YOUR-IP/api/METHOD`

Add your script (and any other necessary files) to your repo and submit your repo URL