Hands-on Prelim Exam	
Name: Shaniah Rose Hope M. Sumaoang	**Date Performed: September 30, 2023**
Course/Section: CPE 232-CPE31S5	**Date Submitted: October 3, 2023**
Instructor: Engr. Roman Richard	**Semester and SY: 1st Sem 2023-2024**

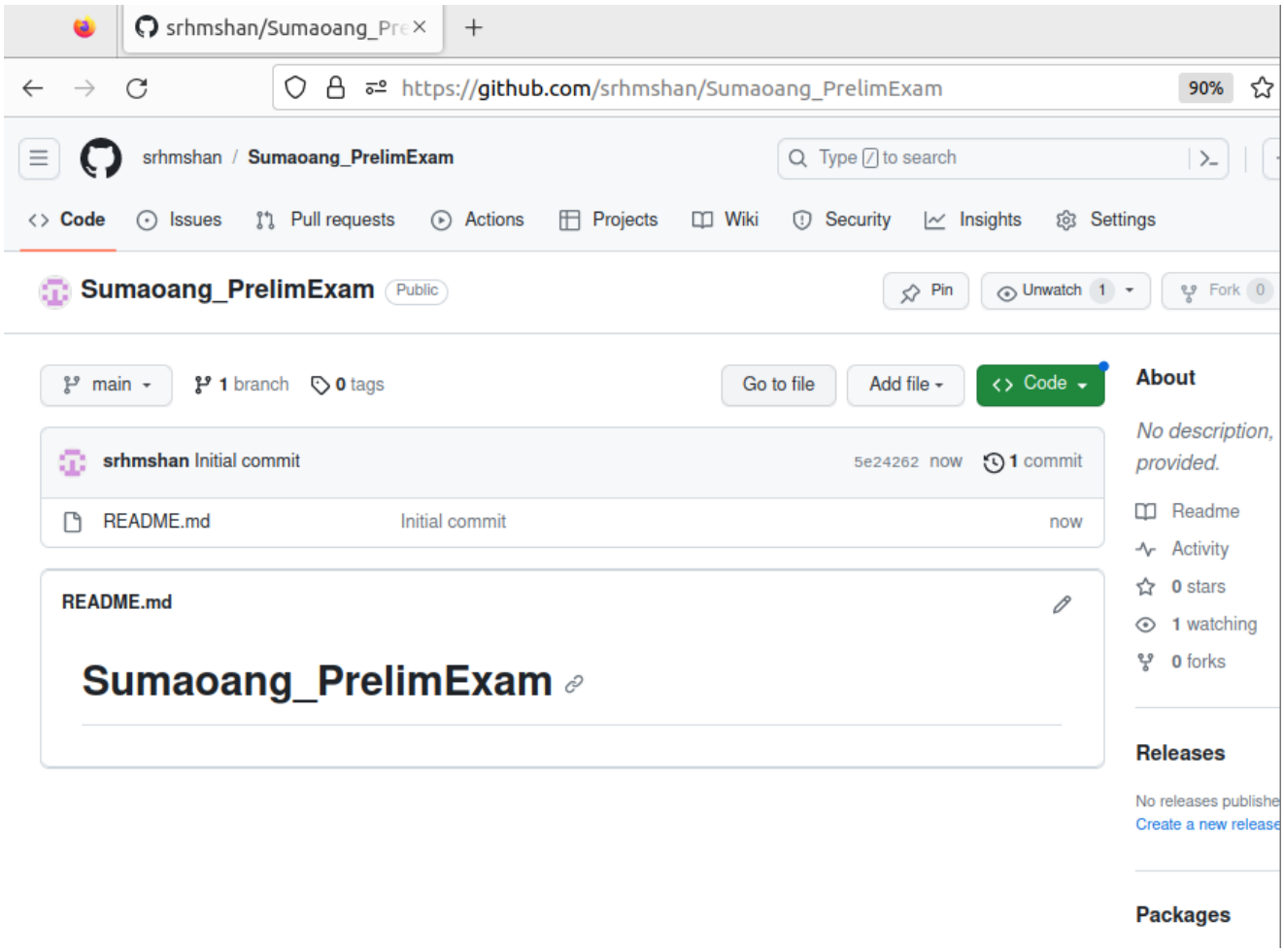
Tools Needed:

1. Control Node (CN) - 1
2. Manage Node (MN) - 1 Ubuntu
3. Manage Node (MN) - 1 CentOS

Procedure:

1. Note: You are required to create a document report of the steps you will do for this exam. All screenshots should be labeled and explained properly.

2. **Create a repository in your GitHub account and label it as Surname_PrelimExam**



- After I logged in on GitHub, in my Control Node (Ubuntu Desktop), I created a new repository named “Sumaoang_PrelimExam” in GitHub, ticked the checkbox to add a “README.md” and created the new repository.

```
srhmshan@localmachine:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQCnx/bUolNrk0LomYoKPIzHPlK0Us+5d0aPfS26AvEYO0usAgTjqh9rEIwtryft+
YY/o+30XNtw92wyRtp91BgnHZD4wSPgNQ1PgpjLxr01J2JcVtEKndVQtmUyoQgH8WF/Zm4QXEhyxuS1vfxHg9kly2/af3bGAHOUUB
dzU2Y+q89mhTOQooQheZ0ZMqNjOI3iPE/Xln3RihMB1LQU8AqhnUpzZeet78bAeMmM/bArrW0M2mXes2MfJ8JJj+7ZCVqJFssK/eJ
TNSs7e8m2LQMubPXpzmPxYvA5CDajolpkiKTVesAujlxxC6zJ14tf2vZFA9j4NOvk06JDkNrnGYnZg1GZgTgeuHioDR9bN8wB1/88
509nkmWbhmqS843nxNvng84nV+mY9A80tz8XFUBWdrIwV2wOzZ+CEZ8csvULOy6h//9hYIpZqoqurscyAeQnFyByyGbJq7KSMDnGB
o/FNbS7e/EFDaOZ/Ig0injkXFd8DEG8PmWU3bxyTSufT67PBH7hshZGVCfzhvgkWlbxbON/4b+u0vnf493aVXoepBh2LDvcsteMg/
+jhmlPl/i+mA54UI2GDfn8dcmYNdU/LSSHpD99BQb6xjeKas0dlGNayhuEB+pVhardMrQpSzD0RRrooH/9jd5MlPjF8VH7c+N2ios
37Q4OULXAip8rtw== srhmshan@localmachine
```

SSH keys

New SSH key

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

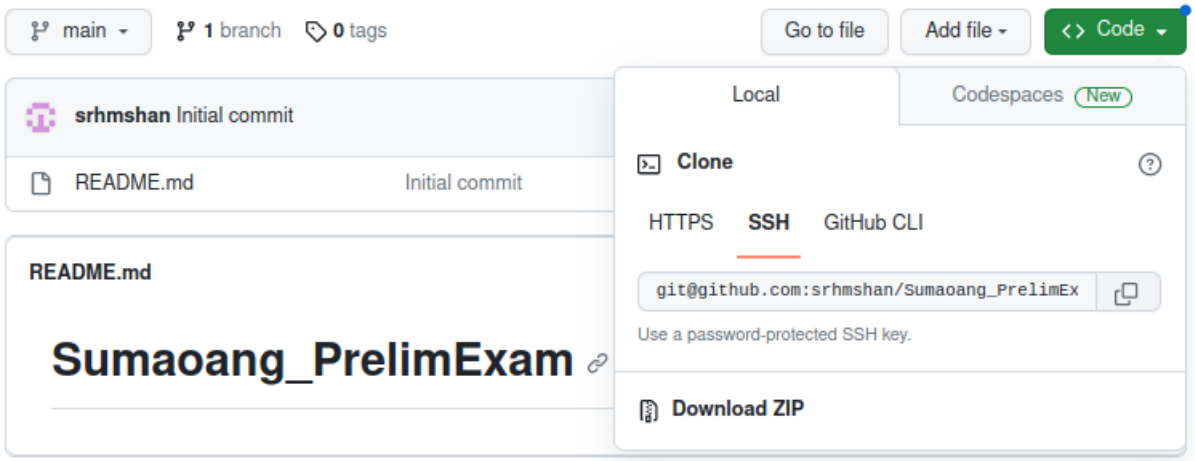
Authentication Keys

CPE232
SHA256:dqdF1vXctpHVk1/JyJsYgCJNmTtxLYfHz7te6SqFa5Q
Added on Aug 24, 2023
Last used within the last week — Read/write
SSH
Delete

Check out our guide to generating SSH keys or troubleshoot common SSH problems.

- I did not need to add an SSH key and just used the existing one instead. If I were to add a new SSH key, I will copy the ID from the new SSH keygen that I will create, delete the existing Authentication Key and paste the new one.

3. Clone your new repository in your CN.



- To clone my new repository in my control node, I copied the SSH and put it in the control node using the following command:

```
srhmshan@localmachine:~$ git clone git@github.com:srhmshan/Sumaoang_PrelimExam.git
Cloning into 'Sumaoang_PrelimExam'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
srhmshan@localmachine:~$ ls
ansible          Desktop    Downloads  Pictures  snap                Templates
CPE232_Sumaoang  Documents  Music      Public    Sumaoang_PrelimExam Videos
```

- I issued the command “ls” to check if the new repository has been successfully cloned in my control node.

4. In your CN, create an inventory file and ansible.cfg files.

- Before creating these files in my control node, I changed my directory to the new repository I created and used “sudo nano” to create and edit the content of the files I created.

```
srhmshan@localmachine:~/Sumaoang_PrelimExam$ sudo nano ansible.cfg
srhmshan@localmachine:~/Sumaoang_PrelimExam$ sudo nano inventory
```

```
  GNU nano 6.2                                    ansible.cfg
[defaults]
inventory = inventory
private_key_file = ~/.ssh/id_rsa
```

- The “ansible.cfg” serves as the main configuration file of our playbook. The “inventory”, on the other hand, is where the hosts’ IP addresses are. It is important to make sure that these manage nodes are connected through SSH. Otherwise, there will be no connection established between the nodes.

  I also changed my network settings as follows: NAT for Control Node and Bridged Adapter for the Manage Nodes. I then ran the command “ifconfig” in the manage nodes to show their IP addresses that I placed inside the inventory file.

```
srhmshan@server1:~$ ifconfig
enp0s3: flags=4163<UP,BROADCA
        inet 192.168.5.46  ne
        inet6 fe80::a00:27ff
```

- Ubuntu Server

```
[srhmshan@CentOS ~]$ ifconfig
enp0s3: flags=4163<UP,BROADCAS
        inet 192.168.5.230  ne
```

- CentOS Server

```
  GNU nano 6.2                                    inventory
[Ubuntu]
192.168.5.46

[CentOS]
192.168.5.230
```

5. **Create an Ansible playbook that does the following with an input of a config.yaml file for both Manage Nodes**

- Installs the latest python3 and pip3

```
srhmshan@localmachine:~/Sumaoang_PrelimExam$ sudo nano config.yaml
```

```
  GNU nano 6.2                                    config.yaml
---
- name: Configure Manage Nodes
  hosts: all
  become: true

  vars:
    user_name: srhmshan

  tasks:
  - name: Install python3 and pip3 on Ubuntu
    apt:
      name:
        - python3
        - python3-pip
      state: latest
      update_cache: yes
    when: ansible_distribution == "Ubuntu"

  - name: Install python3 and pip3 on CentOS
    dnf:
      name:
        - python3
        - python3-pip
      state: latest
      use_backend: dnf4
      update_cache: yes
    when: ansible_distribution == "CentOS"
```

- To install the latest python3 and pip3 on Ubuntu and CentOS, I used the “apt” module for Ubuntu and specified that the ansible_distribution should be Ubuntu for my manage node with an Ubuntu-based system. On the other hand, although the command is almost the same, CentOS uses “dnf” and I added a backend for error handling. The lines where it’s stated to get the latest should ensure that the listed packages are at their latest available versions and “update-cache” should update the package cache before its installation.

```
srhmshan@localmachine:~/Sumaoang_PrelimExam$ ansible-playbook --ask-become-pass config.yaml
BECOME password:

PLAY [Configure Manage Nodes] *****************************************************************

TASK [Gathering Facts] ************************************************************************
ok: [192.168.5.46]
ok: [192.168.5.228]

TASK [Install python3 and pip3 on Ubuntu] *****************************************************
skipping: [192.168.5.228]
ok: [192.168.5.46]

TASK [Install python3 and pip3 on CentOS] *****************************************************
skipping: [192.168.5.46]
ok: [192.168.5.228]

PLAY RECAP ************************************************************************************
192.168.5.228              : ok=2    changed=0    unreachable=0    failed=0    skipped=1    rescued=0
    ignored=0
192.168.5.46               : ok=2    changed=0    unreachable=0    failed=0    skipped=1    rescued=0
    ignored=0
```

- The first task went successful. I would like to give notice that I purposefully used “apt” and “dnf”, instead of using “package” to show heterogeneity. This is shown when each node skipped the task when the “ansible_distribution” condition didn’t specify that the task was for them correspondingly.

- use pip3 as default pip
- use python3 as default python

```
GNU nano 6.2                                    config.yaml
---
- name: Configure Manage Nodes
  hosts: all
  become: true

  vars:
    user_name: srhmshan

  tasks:
    - name: Use pip3 as default pip on Ubuntu
      command: update-alternatives --install /usr/bin/pip pip /usr/bin/pip3 1
      when: ansible_distribution == "Ubuntu"

    - name: Use python3 as default python on Ubuntu
      command: update-alternatives --install /usr/bin/python python /usr/bin/python3 1
      when: ansible_distribution == "Ubuntu"

    - name: Use pip3 as default pip on CentOS
      command: alternatives --set pip /usr/bin/pip3
      when: ansible_distribution == "CentOS"

    - name: Use python3 as default python on CentOS
      command: alternatives --set python /usr/bin/python3
      when: ansible_distribution == "CentOS"
```

- This ensures that pip3 becomes the default pip package manager and python3 the default Python interpreter on Ubuntu and CentOS. I used the update-alternatives and alternatives commands, and their execution is conditionally controlled based on the host's Linux distribution to prevent compatibility issues. To show another successful task:

```
srhmshan@localmachine:~/Sumaoang_PrelimExam$ ansible-playbook --ask-become-pass config.yaml
BECOME password:

PLAY [Configure Manage Nodes] ***************************************************************

TASK [Gathering Facts] ***********************************************************************
ok: [192.168.5.46]
ok: [192.168.5.230]

TASK [Use pip3 as default pip on Ubuntu] *****************************************************
skipping: [192.168.5.230]
changed: [192.168.5.46]

TASK [Use python3 as default python on Ubuntu] ***********************************************
skipping: [192.168.5.230]
changed: [192.168.5.46]

TASK [Use pip3 as default pip on CentOS] *****************************************************
skipping: [192.168.5.46]
changed: [192.168.5.230]

TASK [Use python3 as default python on CentOS] ***********************************************
skipping: [192.168.5.46]
changed: [192.168.5.230]

PLAY RECAP ***********************************************************************************
192.168.5.230              : ok=3    changed=2    unreachable=0    failed=0    skipped=2    rescued=0
    ignored=0
192.168.5.46               : ok=3    changed=2    unreachable=0    failed=0    skipped=2    rescued=0
    ignored=0
```

- Install Java open-jdk

```
  GNU nano 6.2                                    config.yaml
---
- name: Configure Manage Nodes
  hosts: all
  become: true

  vars:
    user_name: srhmshan

  tasks:
    - name: Install Java on Ubuntu
      apt:
        name:
          - openjdk-11-jre
        state: latest
        update_cache: yes
      when: ansible_distribution == "Ubuntu"

    - name: Install Java on CentOS
      dnf:
        name:
          - java-11-openjdk
        state: latest
        update_cache: yes
      when: ansible_distribution == "CentOS"
```

- Similar to the first task, this installs the latest version of Java (OpenJDK) on Ubuntu and CentOS depending on the type of Linux system. For Ubuntu, it uses the "apt" method to install the "openjdk-11-jre" package, and for CentOS, it uses "dnf" to install the "java-11-openjdk" package. I used “when” for when the “ansible_distribution” specifies that the task was for Ubuntu or CentOS correspondingly.

```
srhmshan@localmachine:~/Sumaoang_PrelimExam$ ansible-playbook --ask-become-pass config.yaml
BECOME password:

PLAY [Configure Manage Nodes] *************************************************************

TASK [Gathering Facts] ********************************************************************
ok: [192.168.5.214]
ok: [192.168.5.46]

TASK [Install Java on Ubuntu] *************************************************************
skipping: [192.168.5.214]
changed: [192.168.5.46]

TASK [Install Java on CentOS] *************************************************************
skipping: [192.168.5.46]
changed: [192.168.5.214]

PLAY RECAP ********************************************************************************
192.168.5.214              : ok=2    changed=1    unreachable=0    failed=0    skipped=1    rescued=0
    ignored=0
192.168.5.46               : ok=2    changed=1    unreachable=0    failed=0    skipped=1    rescued=0
    ignored=0
```

- Another successful installation on the manage nodes. Also, note that the IP changed in the CentOS so I had to edit the inventory to put the correct IP address.

- Create Motd containing the text defined by a variable defined in config.yaml file and if there is no variable input the default motd is "Ansible Managed node by (your user name)"

  - I created a .j2 (Jinja2) file and edited the contents first:

```
  GNU nano 6.2                                        motd.j2
{{ motd_content | default("Ansible Managed node by {{ user_name }}") }}
```

```
  GNU nano 6.2                                      config.yaml
---
- name: Configure Manage Nodes
  hosts: all
  become: true

  vars:
    user_name: srhmshan

  tasks:
    - name: Create Motd
      template:
        src: motd.j2
        dest: /etc/motd
```

  - This is used in the src parameter under “template”. In the vars section, I defined the username. It will take the contents from motd.j2, creating a motd_content with the default message that includes the username that I defined.

```
srhmshan@localmachine:~/Sumaoang_PrelimExam$ ansible-playbook --ask-become-pass config.yaml
BECOME password:

PLAY [Configure Manage Nodes] ***************************************************************

TASK [Gathering Facts] **********************************************************************
ok: [192.168.5.214]
ok: [192.168.5.46]

TASK [Create Motd] **************************************************************************
changed: [192.168.5.46]
changed: [192.168.5.214]

PLAY RECAP **********************************************************************************
192.168.5.214              : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0
    ignored=0
192.168.5.46               : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0
    ignored=0
```

- Create a user with a variable defined in config.yaml

```
  GNU nano 6.2                                    config.yaml
---
- name: Configure Manage Nodes
  hosts: all
  become: true

  vars_prompt:
    - name: user_name
      prompt: Enter username

  tasks:
    - name: Create user
      user:
        name: "{{ user_name }}"
        state: present
        createhome: yes
      become: yes
```

- This allows me to customize the username with a defined variable. After I receive the username input, I use it to create new user account. This user account includes a home directory, which is a space where the user can store their files and settings. The task to create the user is carried out with sudo privileges.

```
srhmshan@localmachine:~/Sumaoang_PrelimExam$ ansible-playbook --ask-become-pass config.yaml
BECOME password:
Enter username:

PLAY [Configure Manage Nodes] ***************************************************************

TASK [Gathering Facts] **********************************************************************
ok: [192.168.5.46]
ok: [192.168.5.214]

TASK [Create user] **************************************************************************
changed: [192.168.5.46]
changed: [192.168.5.214]

PLAY RECAP **********************************************************************************
192.168.5.214              : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0
    ignored=0
192.168.5.46               : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0
    ignored=0
```

6. PUSH and COMMIT your PrelimExam in your GitHub repo

```
srhmshan@localmachine:~/Sumaoang_PrelimExam$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        ansible.cfg
        config.yaml
        inventory
        motd.j2

nothing added to commit but untracked files present (use "git add" to track)
```

- I issued this command to see the status of my git. It is on the main branch and these untracked files are what I am supposed to add.

```
srhmshan@localmachine:~/Sumaoang_PrelimExam$ git add .
```

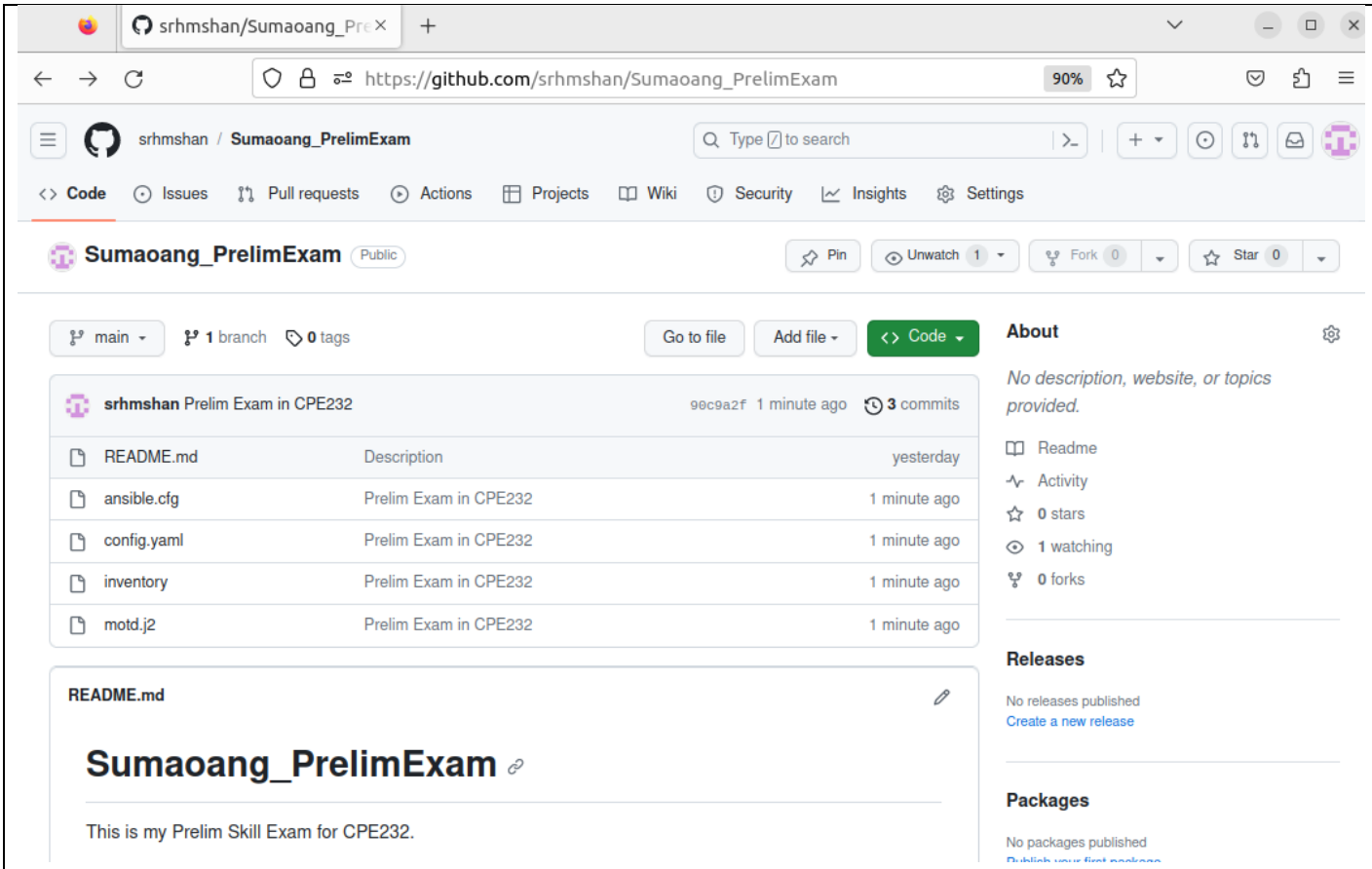
- Using "." will add everything that I've done so far, without having to type each filename.

```
srhmshan@localmachine:~/Sumaoang_PrelimExam$ git commit -m "Prelim Exam in CPE232"
[main 90c9a2f] Prelim Exam in CPE232
 4 files changed, 85 insertions(+)
 create mode 100644 ansible.cfg
 create mode 100644 config.yaml
 create mode 100644 inventory
 create mode 100644 motd.j2
```

- After adding, I issued the command "commit" with a short description that will be shown beside these files.

```
srhmshan@localmachine:~/Sumaoang_PrelimExam$ git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 2 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 1.02 KiB | 1.02 MiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:srhmshan/Sumaoang_PrelimExam.git
   4e8fa4f..90c9a2f  main -> main
```

- I successfully pushed the contents to GitHub.



- This is the result of pushing and committing my PrelimExam to my GitHub repository. https://github.com/srhmshan/Sumaoang_PrelimExam