Name: Kazuki A. Ogata	**Date Performed:** September 29, 2023
Course/Section: CPE 232 - CPE31S5	**Date Submitted:** October 03, 2023
Instructor: Engr. Roman Richard	**Semester and SY:** 1st semester S.Y 2023-2024

Hands-on Prelim Exam

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
3. Clone your new repository in your CN.
4. In your CN, create an inventory file and ansible.cfg files.
5. Create an Ansible playbook that does the following with an input of a config.yaml file for both Manage Nodes
   - 5.1 Installs the latest python3 and pip3
   - 5.2 use pip3 as default pip
   - 5.3 use python3 as default python
   - 5.4 Install Java open-jdk
   - 5.5 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)"
   - 5.6 Create a user with a variable defined in config.yaml
5. PUSH and COMMIT your PrelimExam in your GitHub repo
6. Your document report should be submitted here.
7. For your prelim exam to be counted, please paste your repository link here.

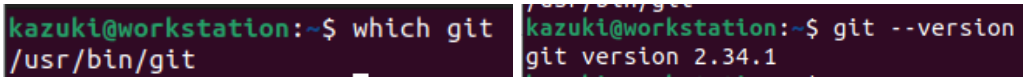

Figure 1 - Verifying Git in CN

I need to make sure I have Git installed in my Control Node before doing the activity, so I will not encounter errors and save me time.Also, this activity requires me to create a repository in GitHub and in order to interact with GitHub using Control Node I need to make sure that I have Git installed in my Control Node to push and pull changes from the repository.

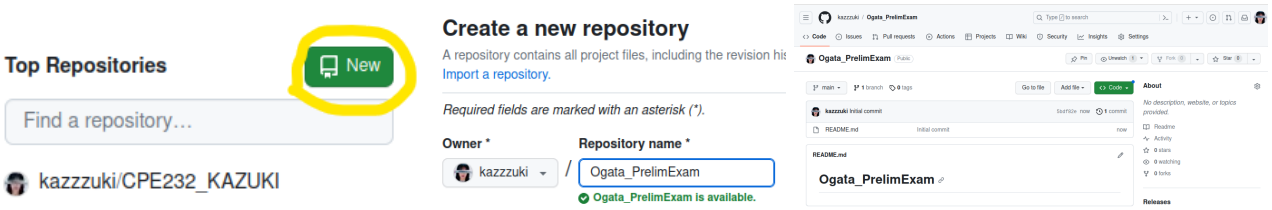

Figure 2 - Adding, Labeling, and Creating a New Repository on GitHub

I created another repository on my GitHub for this activity. With creating a repository, it is very easy since GitHub has a beginner friendly GUI. We create a repository to organize our works/files and it is very easy to manage.

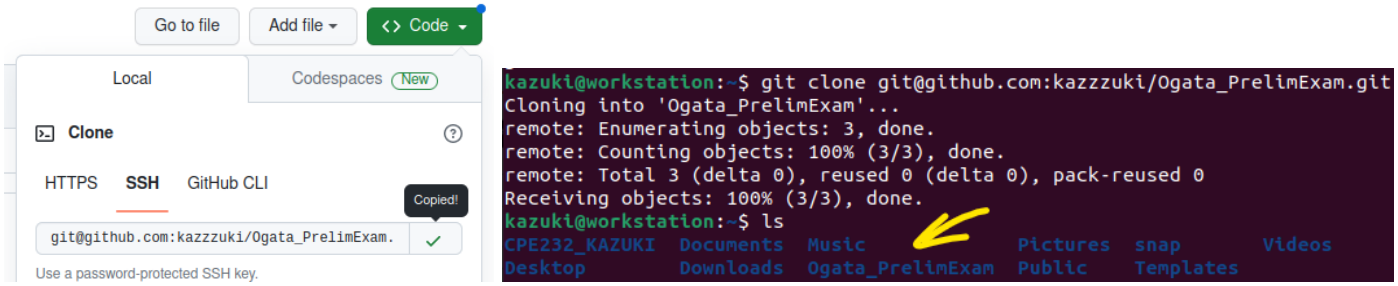

Figure 3 - Cloning the New Repository in CN

To clone the created repository, I click the “<> Code” button in my repository and click the SSH and copied the link there and then I type “git clone <repository link>” in terminal to clone my repository to my Control Node. With this,we can easily work with the repository's files, edit Ansible playbooks, and push and commit changes abc to the repository in GitHub.

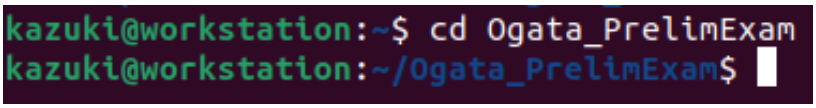

Figure 4.1 - Changing Directory to the Local Repository Directory

We need to navigate/change directory into the cloned repository directory, we can do this by using the “cd’ command. We do this to make sure that all of our works like playbook will be saved in the repository directory and avoid errors.

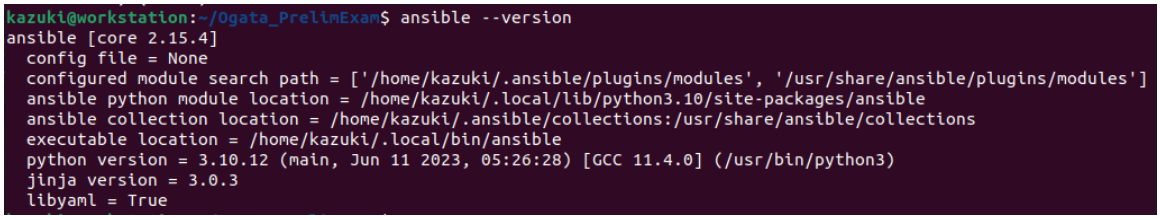

Figure 4.2 - Checking Installed Ansible

I used “ansible –version” to check if I have ansible installed in my Control Node. I did this to make sure that the playbook that I will be creating will not encounter an error when executed.

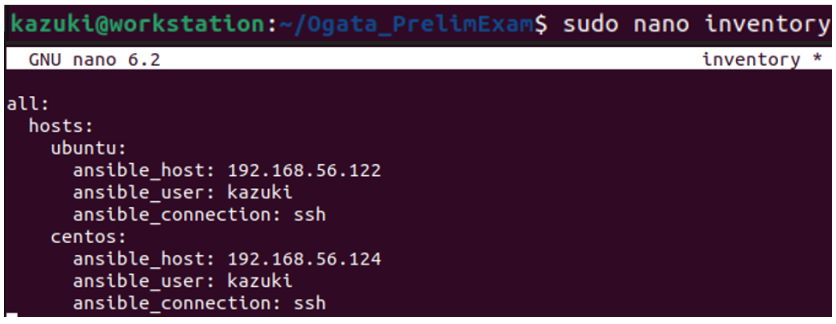

Figure 4.3.1 - Creating inventory File

I put all the hosts/managed nodes inside my inventory that I will be using in this activity. I included the hostnames, IP addresses, and the connection settings. This will help Ansible to connect to each managed node and make sure where it will execute the tasks.

Figure 4.3.2 - Verifying the hosts in the Created Inventory File

I used the command “ansible all –list-hosts” to verify the hosts/managed nodes I included in my inventory. This will display the hostnames ubuntu and centos. I used the command “ansible all -m ping” to verify the connection between my control node and managed nodes. This step is very important to make sure that there are connections and to avoid some errors.

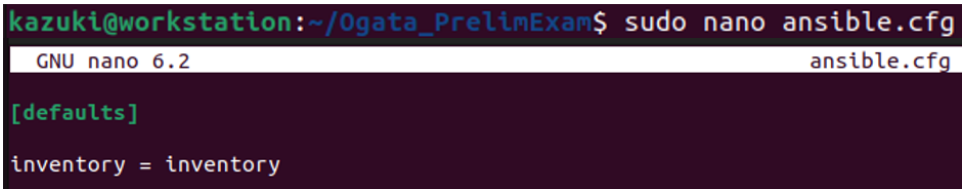

Figure 4.4 - Creating ansible.cfg File

I created an ansible.cfg file to specify Ansible configuration settings. I only put [defaults] and inventory=inventory to ensure that Ansible knows where my hosts/managed nodes are located and can easily connect to them.

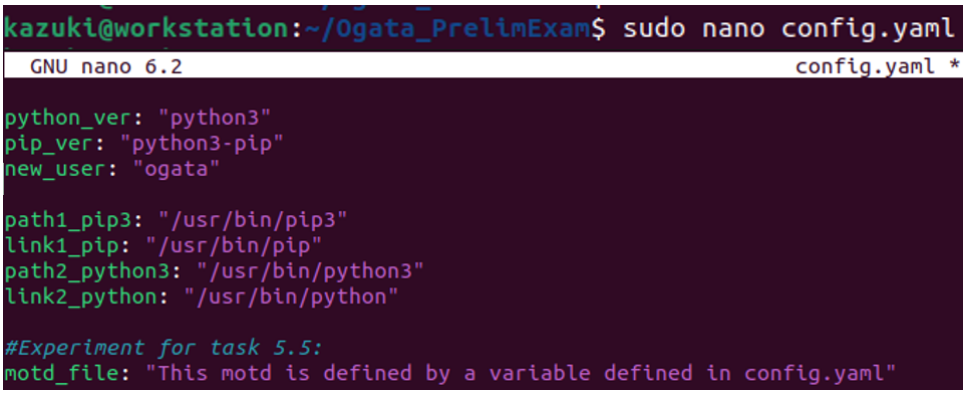

Figure 5.0.1 - Creating config.yaml File

I created a config.yaml file to store variables that I will be using for the configurations of my playbook. With this, it can make my playbook shorter and easier to debug/fix errors just in case. For example, the motd_file is too long If I include it in my playbook , when I can just use the variable “motd_file” and also the File paths and links.

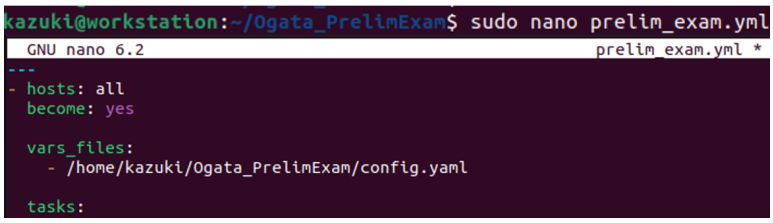

Figure 5.0.2 - Creating Playbook

The three dash (---) is just a document separator. The “- hosts: all” defines the hosts/managed nodes that the tasks in my playbook will be executed on. Since I added “all” in my inventory, I just put all in this part instead of “ubuntu, centos" or their ip addresses. The “become: yes” it just gives permission/privilege for our tasks to run successfully without any more error. The “vars_files:” includes the variable file (config.yaml). With this, we can import variable values that I will be using in my playbook tasks/plays. Lastly, the “tasks:” this is used to define the actual tasks/plays that I need to do in this activity.

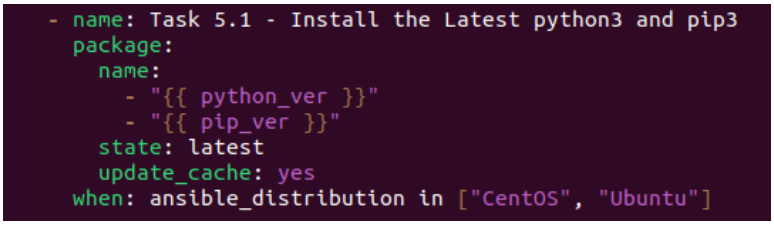

Figure 5.1.1 - Play 1: Installing the Latest python3 and pip3

The “- name:” just defines a name for this task. I used “package” since I will be executing this task in both Ubuntu and CentOS. I can use apt and dnf separately but it will make the code longer so I used this. The “name:” under the package specifies the packages to be managed , I put the variables I specified in the config.yaml file. The “state: latest” will make sure that the python3 and pip3 will install the latest version. I “update_cache: yes” just to make sure that the packages are installed and updated to the latest version and the next tasks will work just fine.

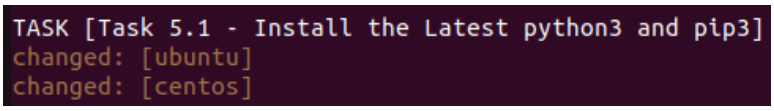

Figure 5.1.2 - Output of Play no.1

The “changed:” means that it successfully installed the latest versions of python3 and pip3 in both of my hosts/managed nodes Ubuntu and CentOS.

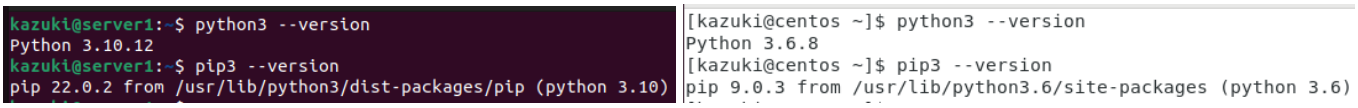

Figure 5.1.3 - Verification of Play no.1 in Ubuntu and CentOS

I added this just as proof that the task/play on my playbook successfully works in both of my hosts/managed nodes Ubuntu and CentOS. Btw, the version of python3 in CentOS are not the same with the version of my Ubuntu since the version of CentOS that I am not using is not the latest version.

```
- name: Task 5.2 - Use pip3 as default pip
  alternatives:
    name: pip
    path: "{{ path1_pip3 }}"
    link: "{{ link1_pip }}"
  when: ansible_distribution in ["CentOS", "Ubuntu"]
```

Figure 5.2.1 - Play 2: Using pip3 as Default pip

I used “alternatives” for cross distribution compatibility since I will be changing the default pip. In path and link I used the variables I defined in my config.yaml file. This task will ensure that the default pip is using pip3.

```
TASK [Task 5.2 - Use pip3 as default pip]
ok: [ubuntu]
ok: [centos]
```

Figure 5.2.2 - Output of Play no.2

This output shows that my task/play was successfully executed in both Ubuntu and CentOS. It is labeled as “ok” meaning that it successfully used pip3 as default pip.

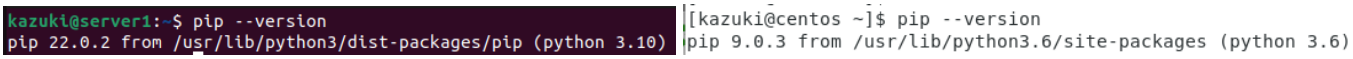

Figure 5.2.3 - Verification of Play no.2 in Ubuntu and CentOS

This is the proof that my Ubuntu and CentOS are now using pip3 as default pip.

```
- name: Task 5.3 - Use python3 as default python
  alternatives:
    name: python
    path: "{{ path2_python3 }}"
    link: "{{ link2_python }}"
  when: ansible_distribution in ["CentOS", "Ubuntu"]
```

Figure 5.3.1 - Play 3: Using python3 as Default python

I just did the same thing here, I used alternatives and the variables defined in config.yaml for path and link.

```
TASK [Task 5.3 - Use python3 as default python]
ok: [ubuntu]
ok: [centos]
```

Figure 5.3.2 - Output of Play no.3

This output shows that my task/play was successfully executed in both Ubuntu and CentOS. It is labeled as “ok” meaning that It successfully used python3 as default python.

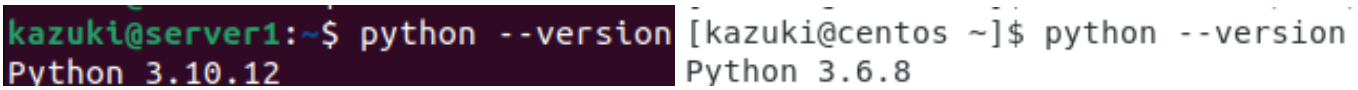

Figure 5.3.3 - Verification of Play no.3 in Ubuntu and CentOS

This is the proof that my hosts/managed nodes Ubuntu and CentOS are now using python3 as default python.

```
- name: Task 5.4 - Install Java open-jdk
  package:
    name: "{{ 'java-1.8.0-openjdk-devel' if ansible_distribution == 'CentOS' else 'openjdk-11-jdk' }}"
    update_cache: yes
  when: ansible_distribution in ["CentOS", "Ubuntu"]
```

Figure 5.4.1 - Play 4: Installing Java open-jdk

In installing Java, I researched first. Based on my research, the version of my CentOS does not have the same java packages supported by my Ubuntu. So I defined them separately. I used the if-else statement here to install the “java-1.8.0-openjdk-devel when it is CentOS” since they are both defined in “when” both of that versions will be installed for both of them.

```
TASK [Task 5.4 - Install Java open-jdk]
changed: [centos]
changed: [ubuntu]
```

Figure 5.4.2 - Output of Play no.4

This output shows that it successfully installs Java open-jdk in Ubuntu and CentOS.

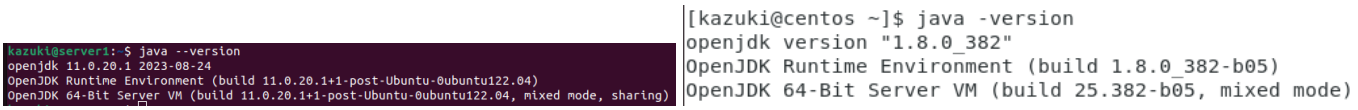

Figure 5.4.3 - Verification of Play no.4 in Ubuntu and CentOS

This is the proof that my Ubuntu and CentOS have Java open-jdk installed.

```
- name: Task 5.5 - Create Motd File
  copy:
    content: "{{ motd_file | default('Ansible Managed node by ' ~ansible_user) }}"
    dest: /etc/motd
  when: ansible_distribution in ["CentOS", "Ubuntu"]
```

Figure 5.5.1.1 - Play 5: Creating Message Of The Day File

I use the copy command to copy the content/file to the managed nodes. In content, I put the variable that I created in my config.yaml. If I remove the content of motd_file in my config.yaml or it is not defined it will print/add to the file the default message “Ansible Managed node by - ansible_user” I input the destination of the file (etc/motd).

```
#Experiment for task 5.5:
motd_file: "This motd is defined by a variable defined in config.yaml"
```

Figure 5.5.1.2 - Variable is Defined in config.yaml File

I am experimenting in this part of my playbook, first it is defined.

```
TASK [Task 5.5 - Create Motd File]
changed: [centos]
changed: [ubuntu]
```

Figure 5.5.1.2 - Output of Play no.5 (Variable Defined)

This output shows that It successfully executed the task in both Managed Nodes.

```
kazuki@server1:~$ cat /etc/motd
This motd is defined by a variable defined in config.yaml
```

```
[kazuki@centos ~]$ cat /etc/motd
This motd is defined by a variable defined in config.yaml
```

Figure 5.5.1.3 - Verification of Play no.5 in Ubuntu and CentOS (Variable Defined)

This is the proof that when a variable is defined in creating a motd file, it will print the message that I put in the variable.

```
#Experiment for task 5.5:
#motd_file: "This motd is defined by a variable defined in config.yaml"
```

Figure 5.5.2.1 - Variable is not Defined in config.yaml File

I removed/commented the defined variable in config.yaml for experimentation.

```
TASK [Task 5.5 - Create Motd File]
changed: [centos]
changed: [ubuntu]
```

Figure 5.5.2.2 - Output of Play no.5 (Default)

This output shows that it successfully executed the task in Ubuntu and CentOS.

```
kazuki@server1:~$ cat /etc/motd
Ansible Managed node by kazukikazuki@server1:~$
```

```
[kazuki@centos ~]$ cat /etc/motd
Ansible Managed node by kazuki[kazuki@centos ~]$
```

Figure 5.5.2.3 - Verification of Play no.5 in Ubuntu and CentOS (Default)

This is the proof that it will run the default message when the variable is not defined.

```
- name: Task 5.6 - Create new User
  user:
    name: "{{ new_user }}"
    state: present
  when: ansible_distribution in ["CentOS", "Ubuntu"]
```

Figure 5.6.1 - Play 6: Creating New User

In creating a new user, I use the “user” command and put the variable I defined in config.yaml in the name command. I included the “state: present” just to make sure that the “name” command exists on the managed nodes.

```
TASK [Task 5.6 - Create new User]
changed: [ubuntu]
changed: [centos]
```

Figure 5.6.2 - Output of Play no.6

This output shows that the task was successful in creating a new user in Ubuntu and CentOS.

```
kazuki@server1:~$ getent passwd ogata:x:1001:1001::/home/ogata:/bin/sh
```

Figure 5.6.4 - Verification of the Created New User (ogata) in Ubuntu

I used the “getent passwd” command to check the created new user and it successfully shows the new user name “ogata” in Ubuntu.

```
[kazuki@centos ~]$ getent passwd ogata:x:1001:1001::/home/ogata:/bin/bash
```

Figure 5.6.5 - Verification of the Created New User (ogata) in CentOS

I used the “getent passwd” command to check the created new user in CentOS and it also shows the new username “ogata”.

```
- name: Ansible Distribution Checker
  debug:
    var: ansible_distribution
```

Figure 5.7.2 - Additional Part of the Playbook (Optional)

I added this just to check if my playbook successfully works in both Ubuntu and CentOS Managed nodes. Because, in some cases, the “ok:” output of the Playbook does not actually execute your playbook in the specified managed nodes.

This is the Playbook with all the tasks:

```
---
- hosts: all
  become: yes

  vars_files:
    - /home/kazuki/Ogata_PrelimExam/config.yaml

  tasks:

    - name: Task 5.1 - Install the Latest python3 and pip3
      package:
        name:
          - "{{ python_ver }}"
          - "{{ pip_ver }}"
        state: latest
        update_cache: yes
      when: ansible_distribution in ["CentOS", "Ubuntu"]

    - name: Task 5.2 - Use pip3 as default pip
      alternatives:
        name: pip
        path: "{{ path1_pip3 }}"
        link: "{{ link1_pip }}"
      when: ansible_distribution in ["CentOS", "Ubuntu"]

    - name: Task 5.3 - Use python3 as default python
      alternatives:
        name: python
        path: "{{ path2_python3 }}"
        link: "{{ link2_python }}"
      when: ansible_distribution in ["CentOS", "Ubuntu"]
    - name: Task 5.4 - Install Java open-jdk
      package:
        name: "{{ 'java-1.8.0-openjdk-devel' if ansible_distribution == 'CentOS' else 'openjdk-11-jdk' }}"
        update_cache: yes
      when: ansible_distribution in ["CentOS", "Ubuntu"]

    - name: Task 5.5 - Create Motd File
      copy:
        content: "{{ motd_file | default('Ansible Managed node by ' ~ansible_user) }}"
        dest: /etc/motd
      when: ansible_distribution in ["CentOS", "Ubuntu"]

    - name: Task 5.6 - Create new User
      user:
        name: "{{ new_user }}"
        state: present
      when: ansible_distribution in ["CentOS", "Ubuntu"]

    - name: Ansible Distribution Checker
      debug:
        var: ansible_distribution
```

This is the output when the Playbook is executed:

```
kazuki@workstation:~/Ogata_PrelimExam$ ansible-playbook --ask-become-pass prelim_exam.yml
BECOME password:

PLAY [all] *********************************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************
ok: [ubuntu]
ok: [centos]

TASK [Task 5.1 - Install the Latest python3 and pip3] **************************************************************************************
ok: [centos]
ok: [ubuntu]

TASK [Task 5.2 - Use pip3 as default pip] **************************************************************************************************
ok: [ubuntu]
ok: [centos]

TASK [Task 5.3 - Use python3 as default python] ********************************************************************************************
ok: [ubuntu]
ok: [centos]

TASK [Task 5.4 - Install Java open-jdk] ****************************************************************************************************
ok: [centos]
ok: [ubuntu]

TASK [Task 5.5 - Create Motd File] *********************************************************************************************************
ok: [centos]
ok: [ubuntu]

TASK [Task 5.6 - Create new User] **********************************************************************************************************
ok: [centos]
ok: [ubuntu]

TASK [Ansible Distribution Checker] ********************************************************************************************************
ok: [ubuntu] => {
    "ansible_distribution": "Ubuntu"
}
ok: [centos] => {
    "ansible_distribution": "CentOS"
}
PLAY RECAP *********************************************************************************************************************************
centos                     : ok=8    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
ubuntu                     : ok=8    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

```
kazuki@workstation:~/Ogata_PrelimExam$ git commit -m "Prelim Skills Exam"
[main e0868b6] Prelim Skills Exam
 4 files changed, 81 insertions(+)
 create mode 100644 ansible.cfg
 create mode 100644 config.yaml
 create mode 100644 inventory
 create mode 100644 prelim_exam.yml
kazuki@workstation:~/Ogata_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), 1.20 KiB | 408.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:kazzzuki/Ogata_PrelimExam.git
   65ad290..e0868b6  main -> main
```

Figure 6.1 - Pushing and Committing my PrelimExam in GitHub

It used git commit and added a message “Prelim Skills Exam” and then git push it so it will be saved to my GitHub.

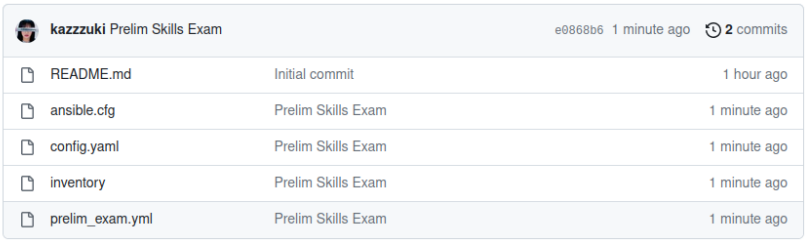

Figure 6.2 - Verification of Pushed and Commited Files in GitHub

This is the proof that I successfully pushed and committed the files that I used/created in this activity on my GitHub Repository.

Link of my GitHub Repository: https://github.com/kazzzuki/Ogata_PrelimExam