Project 1: A development environment

Project done by:

Christopher Jonas Nordal

&

Siham Sidali Feklani

Contents

1. Project selection 3
2. Workflow Design 3
2.1 Define the Workflow 3
3. Technical Documentation 4
3.1 Terraform Configuration 4
Storage Servers Setup: 5
Development Servers Setup: 5
Compile Servers Setup: 5
Docker Testing Server Setup: 6
Ansible Inventory Updater: Integrating Terraform Server IPs: 6
Central Configuration Management: ..... **Error! Bookmark not defined.**
3.2 Ansible Configuration 7
- Adding Users to sudo Group: 7
- Creating Developers Group: 7
- Updating apt: 8
4. Workflow Test 8
5. Experiment execution 11
5.1 Onboarding the Partner group: 11
5.2 Configuring access and permission: 11
5.3 Execution deployment tasks: 11
5.4 Experiment results 13

1. Project selection

A development project is about to start programming. They ask you to deploy a development environment for them. They have the following requirements:

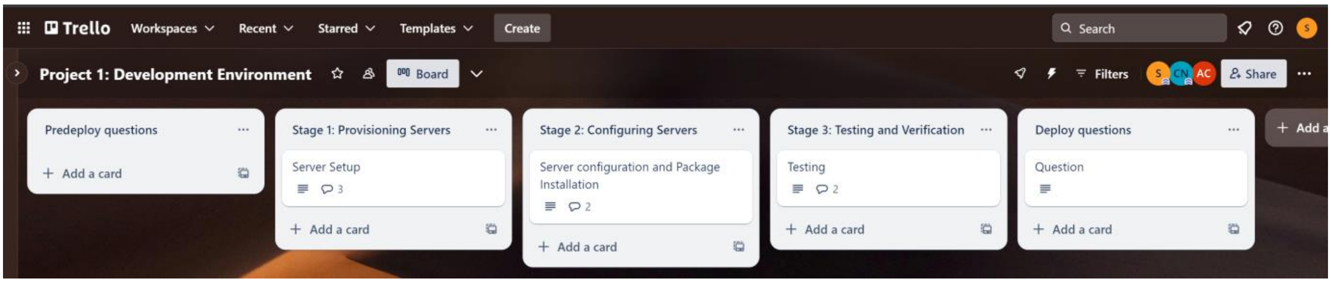
- Two storage servers with the GlusterFS server installed, each with a single CPU.
- Two servers the developers will use to write code on, each with a single CPU. They have to have emacs, jed and git installed
- All machines have to have four users with preferred usernames:

  - bob - janet - alice - tim
- All four users should get root access via sudo
- Tim and janet have to be members of the group «developers». That group has to be created.
- Two compile servers with gcc, make and binutils installed
- One server running Docker for testing the software developed
- You don't have to configure the GlusterFS and Docker services, the project members will do that themselves. Just install the packages.

2. Workflow Design

2.1 Define the Workflow

The deployment workflow for the development environment, as managed on Trello, entails provisioning servers using Terraform and configuring them using Ansible playbooks. The process is organized into stages represented by individual tasks on the Trello board to ensure a systematic approach.



The workflow is represented in a Trello board with distinct columns for each stage:

- **Stage 1: Provisioning Servers** - Tasks related to creating server instances using Terraform. The master server is already set up, and therefore only the servers which will be used as the development environment must be created. Terraform can efficiently deploy servers with a single script which can be easily changed based on the user's needs. It also allows for write commands to a file, which is used to add the created IP addresses to the Ansible inventory.

- **Stage 2: Configuring Servers** - Tasks for installing software packages and managing users using Ansible. Ansible is agentless and doesn't require installing it on any other server than the master server, only requiring a ssh connection to complete the given tasks. This speeds up the process of deploying the environment slightly and removes the risk of any hiccups that might arise from installing puppet on all the servers. This is already been done in the master server

- **Stage 3: Testing and Verification** - Tasks for verifying the functionality of the deployed environment. To successfully deploy the development environment, the user must execute the two scripts; provisioning with Terraform and server configuration with Ansible. Firstly, the Terraform script is executed, which will generate the servers. This can be confirmed either by looking at the output of the Terraform script, or by checking the OpenStack dashboard. Additionally, one can check the Ansible hosts-file containing the IP addresses in groups such as developing, storage, etc., to verify that they were written correctly by the script. It's worth noting that the user might need to wait a few seconds before executing the Ansible script, as the servers might not connect properly. Once the user executes the Ansible script, it will output the status of each task on all the servers in real-time, so that the user knows if the tasks failed/succeeded.

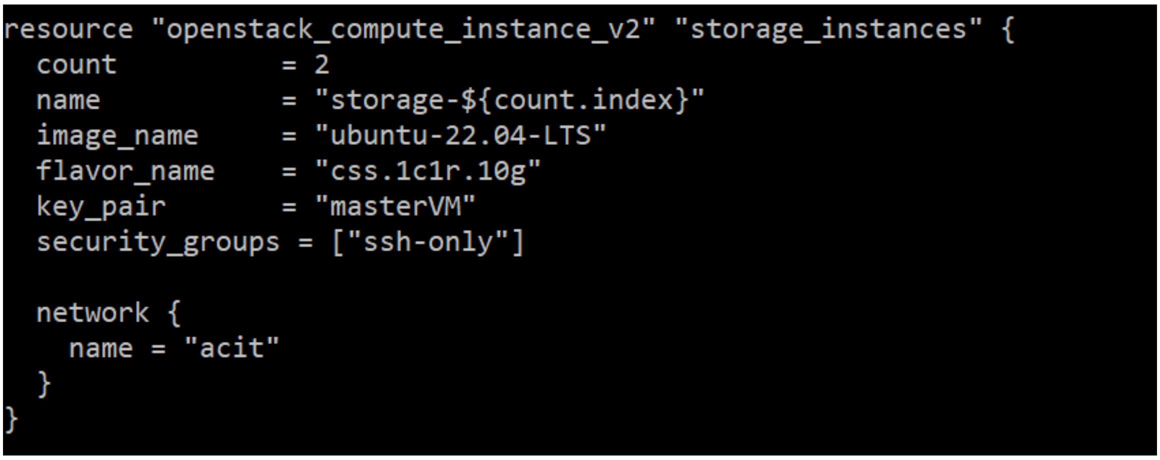
The workflow is designed to prioritize server provisioning before software configuration to ensure a stable infrastructure foundation. Tasks are sequenced logically to minimize dependencies and streamline the deployment process.

3. Technical Documentation

3.1 Terraform Configuration

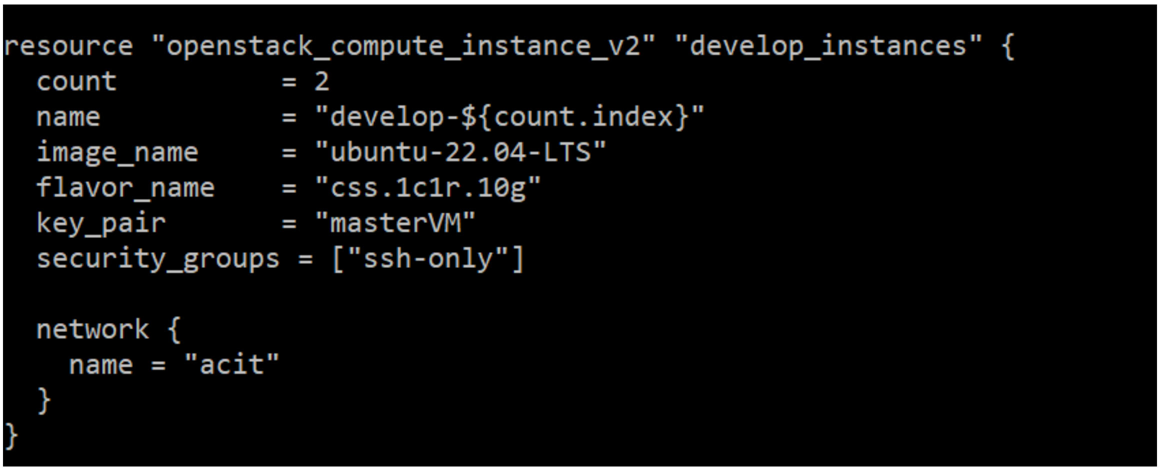
- ➢ In the Terraform script, we defined resource blocks for creating necessary servers.
- ➢ For those containing the count parameter, it was set to 2 to create two servers.
- ➢ Each server was configured with the necessary parameters such as name, image_name, flavor_name, key_pair, security_groups, and network.
  - The name parameter specifies the hostname of each server.
  - The image_name parameter specifies the operating system image to use (in this case, "ubuntu-22.04-LTS").
  - The flavor_name parameter defines the hardware specifications (e.g., CPU, memory) for each server.
  - The key_pair parameter specifies the SSH key pair to use for authentication.
  - The security_groups parameter defines the security group(s) to apply to each server, restricting access based on defined rules.
  - The network block specifies the network to which each server should be connected.
  - The code is similar for storage, developing, compiling servers, as we need 2 of each in groups, except the Docker server

Storage Servers Setup:



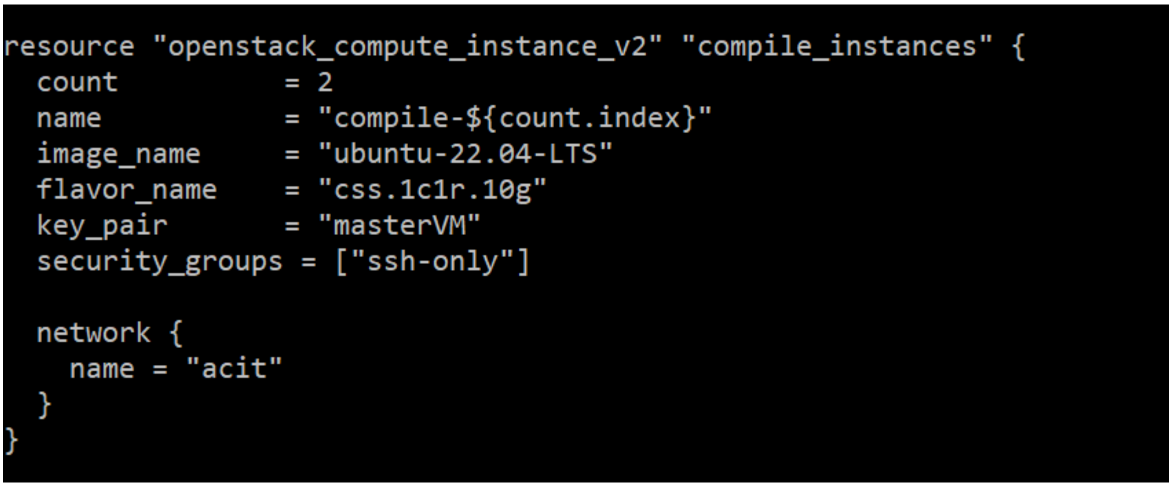
```
resource "openstack_compute_instance_v2" "storage_instances" {
  count           = 2
  name            = "storage-${count.index}"
  image_name      = "ubuntu-22.04-LTS"
  flavor_name     = "css.1c1r.10g"
  key_pair        = "masterVM"
  security_groups = ["ssh-only"]

  network {
    name = "acit"
  }
}
```

Development Servers Setup:



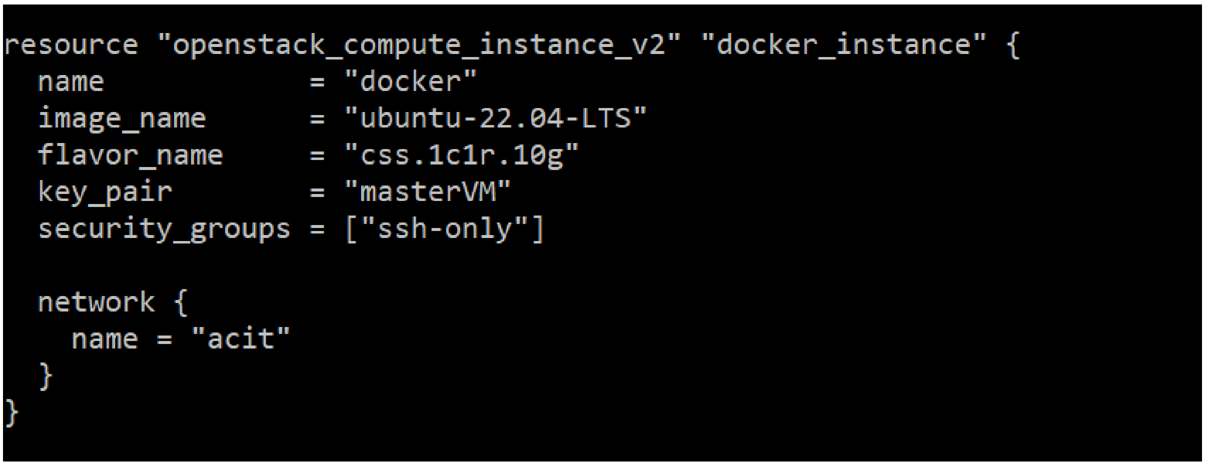
```
resource "openstack_compute_instance_v2" "develop_instances" {
  count           = 2
  name            = "develop-${count.index}"
  image_name      = "ubuntu-22.04-LTS"
  flavor_name     = "css.1c1r.10g"
  key_pair        = "masterVM"
  security_groups = ["ssh-only"]

  network {
    name = "acit"
  }
}
```

Compile Servers Setup:



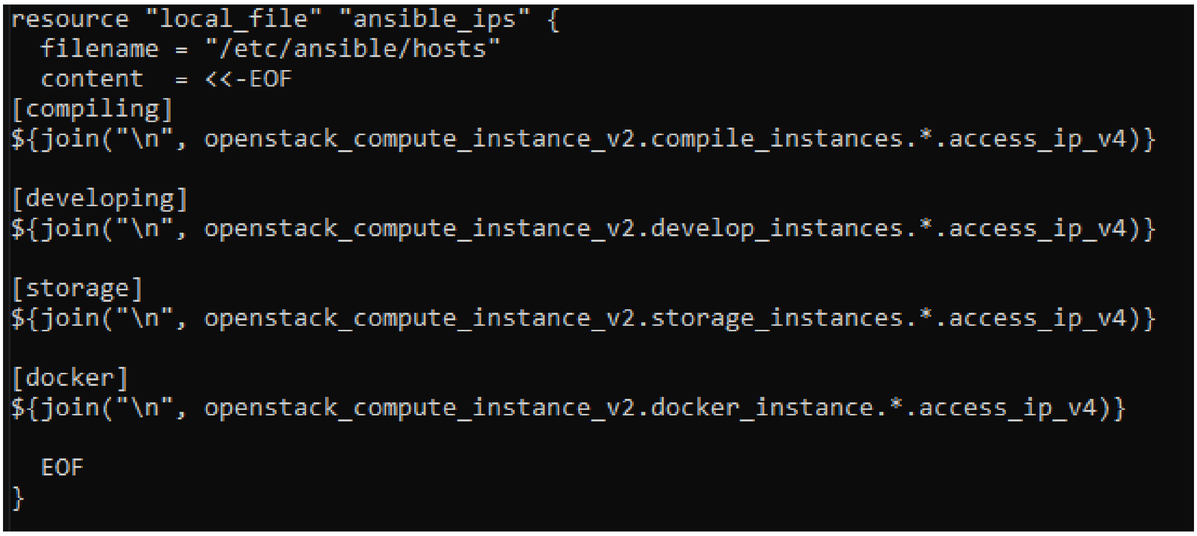
```
resource "openstack_compute_instance_v2" "compile_instances" {
  count           = 2
  name            = "compile-${count.index}"
  image_name      = "ubuntu-22.04-LTS"
  flavor_name     = "css.1c1r.10g"
  key_pair        = "masterVM"
  security_groups = ["ssh-only"]

  network {
    name = "acit"
  }
}
```

Docker Testing Server Setup:



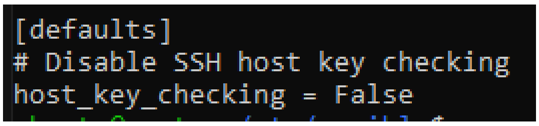
```
resource "openstack_compute_instance_v2" "docker_instance" {
  name            = "docker"
  image_name      = "ubuntu-22.04-LTS"
  flavor_name     = "css.1c1r.10g"
  key_pair        = "masterVM"
  security_groups = ["ssh-only"]

  network {
    name = "acit"
  }
}
```

Ansible Inventory Updater: Integrating Terraform Server IPs:



```
resource "local_file" "ansible_ips" {
  filename = "/etc/ansible/hosts"
  content  = <<-EOF
[compiling]
${join("\n", openstack_compute_instance_v2.compile_instances.*.access_ip_v4)}

[developing]
${join("\n", openstack_compute_instance_v2.develop_instances.*.access_ip_v4)}

[storage]
${join("\n", openstack_compute_instance_v2.storage_instances.*.access_ip_v4)}

[docker]
${join("\n", openstack_compute_instance_v2.docker_instance.*.access_ip_v4)}

  EOF
}
```

Separating them is useful since we can easily group the servers based on what instance they are when writing to the Ansible hosts-file at /etc/ansible/hosts. A resource is created, specifying that a local file should be created at the location mentioned. Four groups are created, which is necessary since we will be installing packages on the servers based on their purpose. The IPs need to be under the [group_name] categories, each IP on a new line.

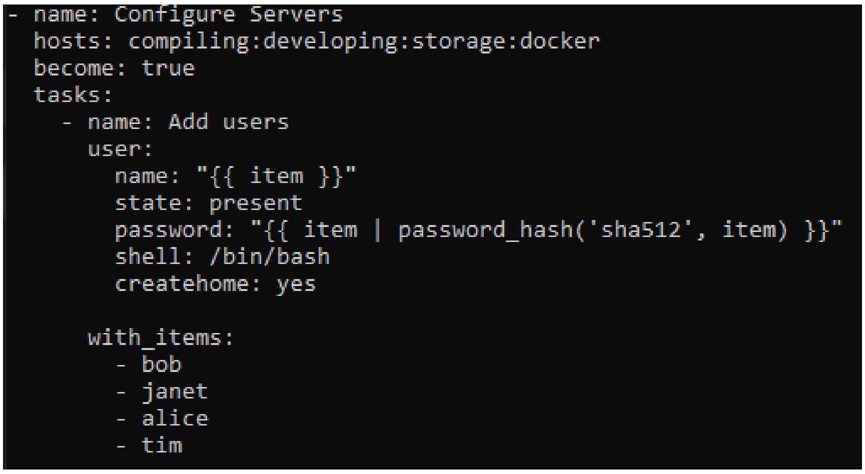


```
[defaults]
# Disable SSH host key checking
host_key_checking = False
```

Disabling host key checking in the Ansible configuration file at /etc/ansible/ansible.cfg removes a layer of security for the client, however, since the project focuses mainly on deploying an environment, it should be acceptable.

3.2 Ansible Configuration

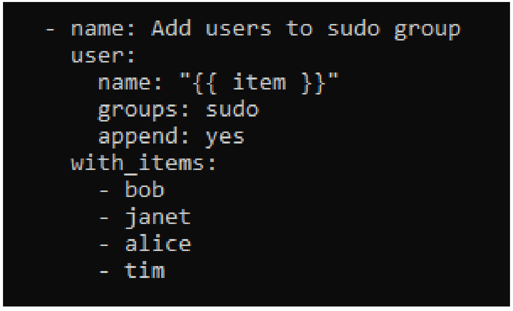
After deploying the storage servers using Terraform, we utilize Ansible to manage the configuration of these servers. We created an Ansible playbook specifically for configuring the storage servers (**project1.yml**). Within the playbook, we define tasks to perform various configuration steps on the storage servers.



```
- name: Configure Servers
  hosts: compiling:developing:storage:docker
  become: true
  tasks:
    - name: Add users
      user:
        name: "{{ item }}"
        state: present
        password: "{{ item | password_hash('sha512', item) }}"
        shell: /bin/bash
        createhome: yes

      with_items:
        - bob
        - janet
        - alice
        - tim
```

The hosts are specified, which are the groups created by the Terraform script, in the Ansible inventory. We ensure that the playbook has the necessary privilege to perform the tasks, since creating users and installing packages require elevated privileges. The name is given a placeholder name, and it will instead iterate over each item in the loop, containing the names of the users to be added. A password is created for each user, the password being the name of the user. We also set bash as the shell, since shell is default for Ansible.

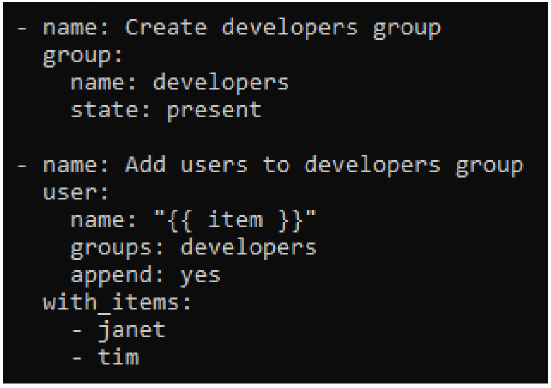
- Adding Users to sudo Group:



```
  - name: Add users to sudo group
    user:
      name: "{{ item }}"
      groups: sudo
      append: yes
    with_items:
      - bob
      - janet
      - alice
      - tim
```

The users are added to the sudo group so that they get root access. Again, a loop is created which iterates over all the users. The users are appended to the group, in case they were part of other groups as well.

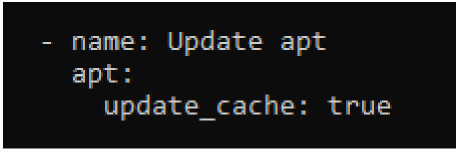
- Creating Developers Group:



```
- name: Create developers group
  group:
    name: developers
    state: present

- name: Add users to developers group
  user:
    name: "{{ item }}"
    groups: developers
    append: yes
  with_items:
    - janet
    - tim
```

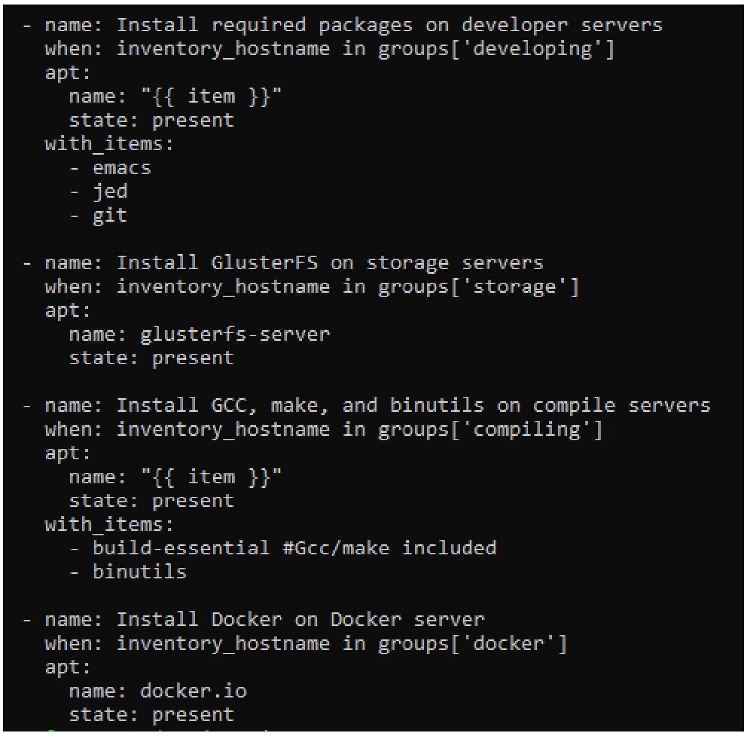
A developer's group is created for Janet and Tim. They are also appended to this group, since they are also in the sudo group.

- Updating apt:



```
- name: Update apt
  apt:
    update_cache: true
```

Updating the apt cache ensures that the latest packages are available.



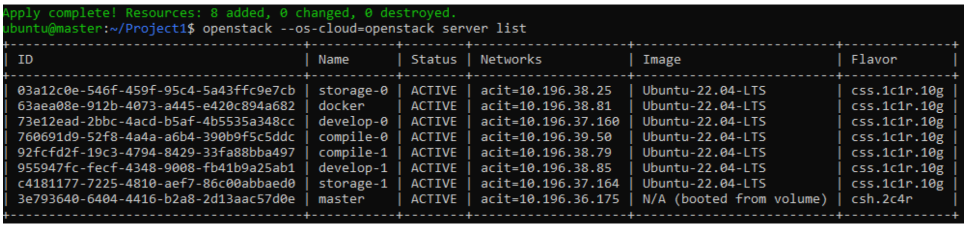
```
- name: Install required packages on developer servers
  when: inventory_hostname in groups['developing']
  apt:
    name: "{{ item }}"
    state: present
  with_items:
    - emacs
    - jed
    - git

- name: Install GlusterFS on storage servers
  when: inventory_hostname in groups['storage']
  apt:
    name: glusterfs-server
    state: present

- name: Install GCC, make, and binutils on compile servers
  when: inventory_hostname in groups['compiling']
  apt:
    name: "{{ item }}"
    state: present
  with_items:
    - build-essential #Gcc/make included
    - binutils

- name: Install Docker on Docker server
  when: inventory_hostname in groups['docker']
  apt:
    name: docker.io
    state: present
```

Packages are installed for developing, storage and compile servers. Again, we can use loops to efficiently write the code to install packages if there are multiple.

4. Workflow Test

Before running the Terraform script, the user must initialize the directory with ‘terraform init’. ‘terraform plan’ can be used to view the changes that will happen if the script executes. ‘sudo terraform apply’ will execute the script if the user enters ‘yes’. ‘terraform destroy’ will destroy the servers created from the Terraform script in the current directory. You can verify that they have been created by looking at the OpenStack dashboard or commands from the terminal. The IPs and their groups can be checked at /etc/ansible/hosts as well. After the servers have been created, it is recommended to wait a few seconds after the Terraform outputs it as completed, to avoid any connection issues when executing the Ansible script. The Ansible script is executed with ‘ansible-playbook ansible_files/project1.yml’. The Ansible output should be enough to know if the tasks were executed correctly, but manual tests can be done on the servers, such as checking installed packages, all users having been created, root access, etc.

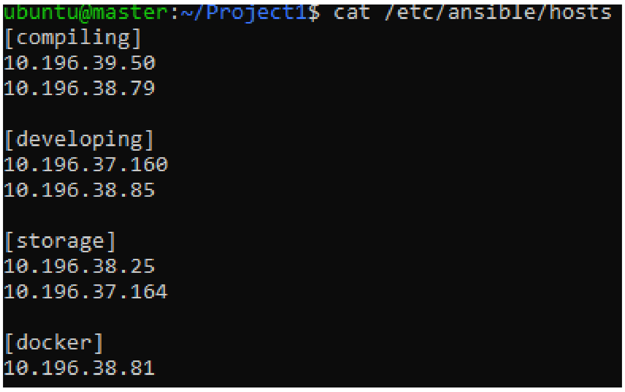
- **<u>First test:</u>**



```
Apply complete! Resources: 8 added, 0 changed, 0 destroyed.
ubuntu@master:~/Project1$ openstack --os-cloud=openstack server list
+--------------------------------------+-----------+--------+--------------------+--------------------------+--------------+
| ID                                   | Name      | Status | Networks           | Image                    | Flavor       |
+--------------------------------------+-----------+--------+--------------------+--------------------------+--------------+
| 03a12c0e-546f-459f-95c4-5a43ffc9e7cb | storage-0 | ACTIVE | acit=10.196.38.25  | Ubuntu-22.04-LTS         | css.1c1r.10g |
| 63aea08e-912b-4073-a445-e420c894a682 | docker    | ACTIVE | acit=10.196.38.81  | Ubuntu-22.04-LTS         | css.1c1r.10g |
| 73e12ead-2bbc-4acd-b5af-4b5535a348cc | develop-0 | ACTIVE | acit=10.196.37.160 | Ubuntu-22.04-LTS         | css.1c1r.10g |
| 760691d9-52f8-4a4a-a6b4-390b9f5c5ddc | compile-0 | ACTIVE | acit=10.196.39.50  | Ubuntu-22.04-LTS         | css.1c1r.10g |
| 92fcfd2f-19c3-4794-8429-33fa88bba497 | compile-1 | ACTIVE | acit=10.196.38.79  | Ubuntu-22.04-LTS         | css.1c1r.10g |
| 955947fc-fecf-4348-9008-fb41b9a25ab1 | develop-1 | ACTIVE | acit=10.196.38.85  | Ubuntu-22.04-LTS         | css.1c1r.10g |
| c4181177-7225-4810-aef7-86c00abbaed0 | storage-1 | ACTIVE | acit=10.196.37.164 | Ubuntu-22.04-LTS         | css.1c1r.10g |
| 3e793640-6404-4416-b2a8-2d13aac57d0e | master    | ACTIVE | acit=10.196.36.175 | N/A (booted from volume) | csh.2c4r     |
+--------------------------------------+-----------+--------+--------------------+--------------------------+--------------+
```

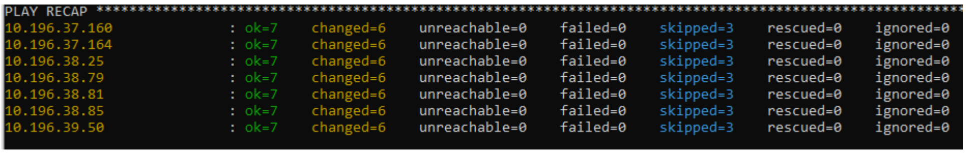
Figure: servers are successfully created

From the figure, 8 resources are added, while we only create 7 servers. This is because of the additional resource which is responsible for writing the server IP addresses to the Ansible host-file.



```
ubuntu@master:~/Project1$ cat /etc/ansible/hosts
[compiling]
10.196.39.50
10.196.38.79

[developing]
10.196.37.160
10.196.38.85

[storage]
10.196.38.25
10.196.37.164

[docker]
10.196.38.81
```

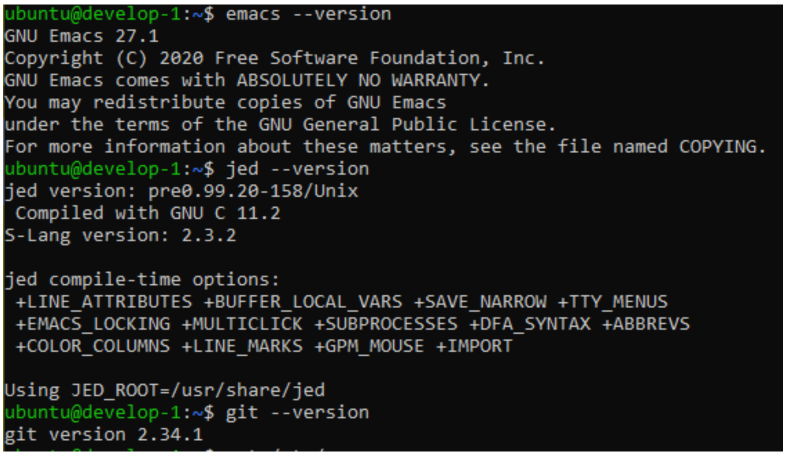
Figure: grouping the servers based on their purpose



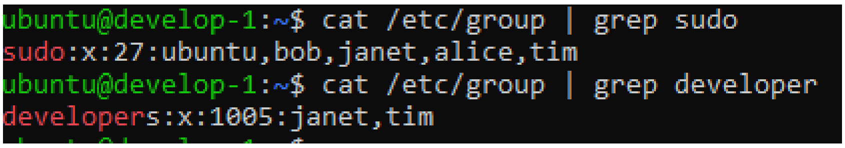
```
PLAY RECAP *************************************************************************************
10.196.37.160              : ok=7    changed=6    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
10.196.37.164              : ok=7    changed=6    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
10.196.38.25               : ok=7    changed=6    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
10.196.38.79               : ok=7    changed=6    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
10.196.38.81               : ok=7    changed=6    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
10.196.38.85               : ok=7    changed=6    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
10.196.39.50               : ok=7    changed=6    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
```

Figure: Final output after running Ansible playbook

With the output from the figure above, the tasks were successfully completed. We can also do manual checks on a random server, for example the develop-1 server, which should have emacs, jed, git installed.

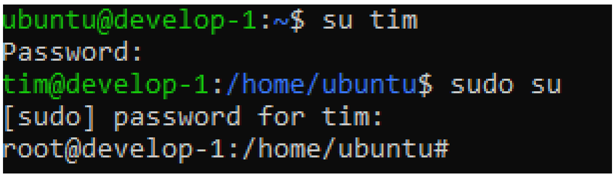


```
ubuntu@develop-1:~$ emacs --version
GNU Emacs 27.1
Copyright (C) 2020 Free Software Foundation, Inc.
GNU Emacs comes with ABSOLUTELY NO WARRANTY.
You may redistribute copies of GNU Emacs
under the terms of the GNU General Public License.
For more information about these matters, see the file named COPYING.
ubuntu@develop-1:~$ jed --version
jed version: pre0.99.20-158/Unix
 Compiled with GNU C 11.2
S-Lang version: 2.3.2

jed compile-time options:
 +LINE_ATTRIBUTES +BUFFER_LOCAL_VARS +SAVE_NARROW +TTY_MENUS
 +EMACS_LOCKING +MULTICLICK +SUBPROCESSES +DFA_SYNTAX +ABBREVS
 +COLOR_COLUMNS +LINE_MARKS +GPM_MOUSE +IMPORT

Using JED_ROOT=/usr/share/jed
ubuntu@develop-1:~$ git --version
git version 2.34.1
```



```
ubuntu@develop-1:~$ cat /etc/group | grep sudo
sudo:x:27:ubuntu,bob,janet,alice,tim
ubuntu@develop-1:~$ cat /etc/group | grep developer
developers:x:1005:janet,tim
```

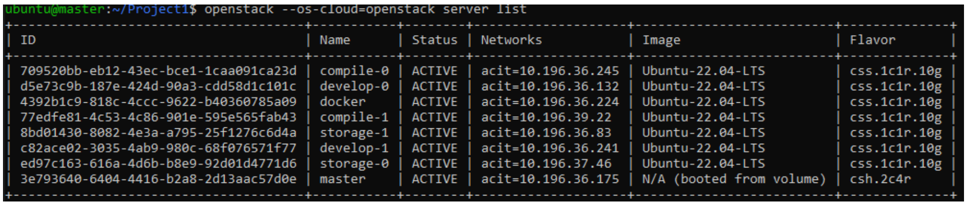
The figures above confirm that the required packages have been installed and the users are in the specified groups.



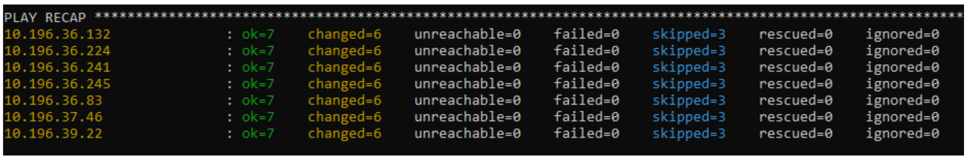
```
ubuntu@develop-1:~$ su tim
Password:
tim@develop-1:/home/ubuntu$ sudo su
[sudo] password for tim:
root@develop-1:/home/ubuntu#
```

Figure: switching to user and root access from user

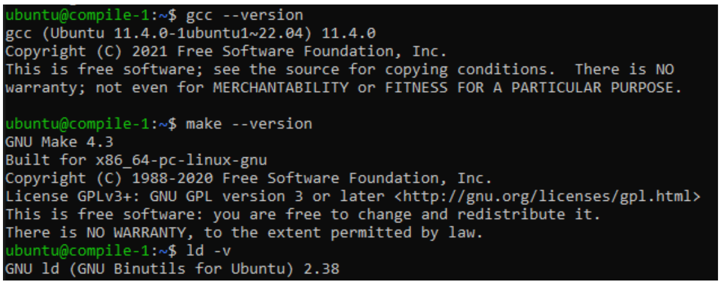
- **Second test:**



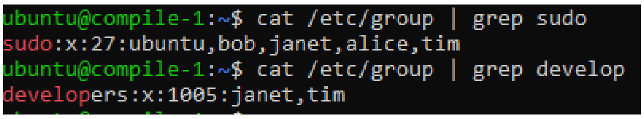
```
ubuntu@master:~/Project1$ openstack --os-cloud=openstack server list
+--------------------------------------+-----------+--------+--------------------+--------------------------+--------------+
| ID                                   | Name      | Status | Networks           | Image                    | Flavor       |
+--------------------------------------+-----------+--------+--------------------+--------------------------+--------------+
| 709520bb-eb12-43ec-bce1-1caa091ca23d | compile-0 | ACTIVE | acit=10.196.36.245 | Ubuntu-22.04-LTS         | css.1c1r.10g |
| d5e73c9b-187e-424d-90a3-cdd58d1c101c | develop-0 | ACTIVE | acit=10.196.36.132 | Ubuntu-22.04-LTS         | css.1c1r.10g |
| 4392b1c9-818c-4ccc-9622-b40360785a09 | docker    | ACTIVE | acit=10.196.36.224 | Ubuntu-22.04-LTS         | css.1c1r.10g |
| 77edfe81-4c53-4c86-901e-595e565fab43 | compile-1 | ACTIVE | acit=10.196.39.22  | Ubuntu-22.04-LTS         | css.1c1r.10g |
| 8bd01430-8082-4e3a-a795-25f1276c6d4a | storage-1 | ACTIVE | acit=10.196.36.83  | Ubuntu-22.04-LTS         | css.1c1r.10g |
| c82ace02-3035-4ab9-980c-68f076571f77 | develop-1 | ACTIVE | acit=10.196.36.241 | Ubuntu-22.04-LTS         | css.1c1r.10g |
| ed97c163-616a-4d6b-b8e9-92d01d4771d6 | storage-0 | ACTIVE | acit=10.196.37.46  | Ubuntu-22.04-LTS         | css.1c1r.10g |
| 3e793640-6404-4416-b2a8-2d13aac57d0e | master    | ACTIVE | acit=10.196.36.175 | N/A (booted from volume) | csh.2c4r     |
+--------------------------------------+-----------+--------+--------------------+--------------------------+--------------+
```



```
PLAY RECAP ***************************************************************************************************
10.196.36.132              : ok=7    changed=6    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
10.196.36.224              : ok=7    changed=6    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
10.196.36.241              : ok=7    changed=6    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
10.196.36.245              : ok=7    changed=6    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
10.196.36.83               : ok=7    changed=6    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
10.196.37.46               : ok=7    changed=6    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
10.196.39.22               : ok=7    changed=6    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
```



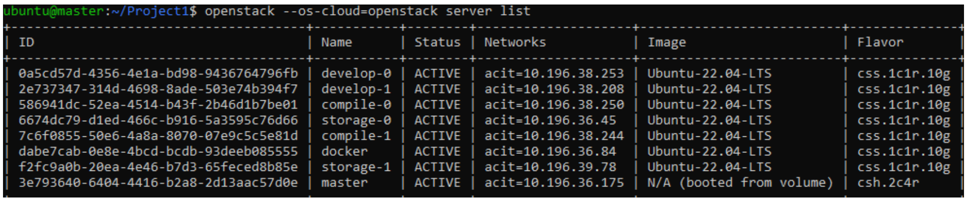
```
ubuntu@compile-1:~$ gcc --version
gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

ubuntu@compile-1:~$ make --version
GNU Make 4.3
Built for x86_64-pc-linux-gnu
Copyright (C) 1988-2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
ubuntu@compile-1:~$ ld -v
GNU ld (GNU Binutils for Ubuntu) 2.38
```



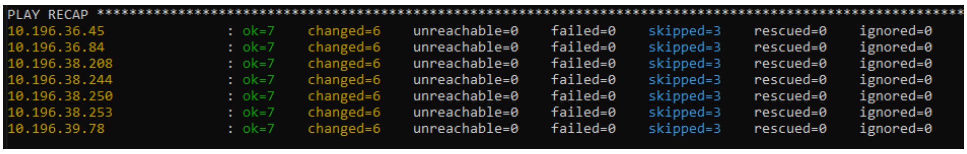
```
ubuntu@compile-1:~$ cat /etc/group | grep sudo
sudo:x:27:ubuntu,bob,janet,alice,tim
ubuntu@compile-1:~$ cat /etc/group | grep develop
developers:x:1005:janet,tim
```

The Ansible output shows the same output as the previous test, which indicates that all tasks were completed successfully. Again, we can check a random server, such as the compile-1, where GCC, make, binutils were installed.

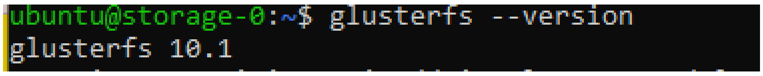
- **Third test:**



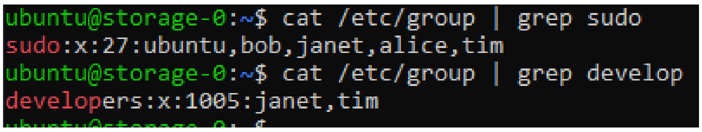
```
ubuntu@master:~/Project1$ openstack --os-cloud=openstack server list
+--------------------------------------+-----------+--------+--------------------+--------------------------+--------------+
| ID                                   | Name      | Status | Networks           | Image                    | Flavor       |
+--------------------------------------+-----------+--------+--------------------+--------------------------+--------------+
| 0a5cd57d-4356-4e1a-bd98-9436764796fb | develop-0 | ACTIVE | acit=10.196.38.253 | Ubuntu-22.04-LTS         | css.1c1r.10g |
| 2e737347-314d-4698-8ade-503e74b394f7 | develop-1 | ACTIVE | acit=10.196.38.208 | Ubuntu-22.04-LTS         | css.1c1r.10g |
| 586941dc-52ea-4514-b43f-2b46d1b7be01 | compile-0 | ACTIVE | acit=10.196.38.250 | Ubuntu-22.04-LTS         | css.1c1r.10g |
| 6674dc79-d1ed-466c-b916-5a3595c76d66 | storage-0 | ACTIVE | acit=10.196.36.45  | Ubuntu-22.04-LTS         | css.1c1r.10g |
| 7c6f0855-50e6-4a8a-8070-07e9c5c5e81d | compile-1 | ACTIVE | acit=10.196.38.244 | Ubuntu-22.04-LTS         | css.1c1r.10g |
| dabe7cab-0e8e-4bcd-bcdb-93deeb085555 | docker    | ACTIVE | acit=10.196.36.84  | Ubuntu-22.04-LTS         | css.1c1r.10g |
| f2fc9a0b-20ea-4e46-b7d3-65feced8b85e | storage-1 | ACTIVE | acit=10.196.39.78  | Ubuntu-22.04-LTS         | css.1c1r.10g |
| 3e793640-6404-4416-b2a8-2d13aac57d0e | master    | ACTIVE | acit=10.196.36.175 | N/A (booted from volume) | csh.2c4r     |
```



```
PLAY RECAP ***************************************************************************************************
10.196.36.45               : ok=7    changed=6    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
10.196.36.84               : ok=7    changed=6    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
10.196.38.208              : ok=7    changed=6    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
10.196.38.244              : ok=7    changed=6    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
10.196.38.250              : ok=7    changed=6    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
10.196.38.253              : ok=7    changed=6    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
10.196.39.78               : ok=7    changed=6    unreachable=0    failed=0    skipped=3    rescued=0    ignored=0
```



```
ubuntu@storage-0:~$ glusterfs --version
glusterfs 10.1
```



```
ubuntu@storage-0:~$ cat /etc/group | grep sudo
sudo:x:27:ubuntu,bob,janet,alice,tim
ubuntu@storage-0:~$ cat /etc/group | grep develop
developers:x:1005:janet,tim
```

➢ **Reflection**

Overall, our implementation successfully completed the project requirements. We provision the servers with Terraform and install packages, create users with root access, as well as a group for the developers with Ansible. The scripts can be easily expanded based on the user's needs, such as provisioning more servers and installing more packages, creating users, etc. More complex solutions could be explored as well, such as only having to run one script that would execute both scripts. The main script would need to handle any errors that might arise, such as the Ansible script being executed too early, before the servers are available for ssh connection. Such a solution could improve the overall deployment time.

5. Experiment execution

5.1 Onboarding the Partner group:

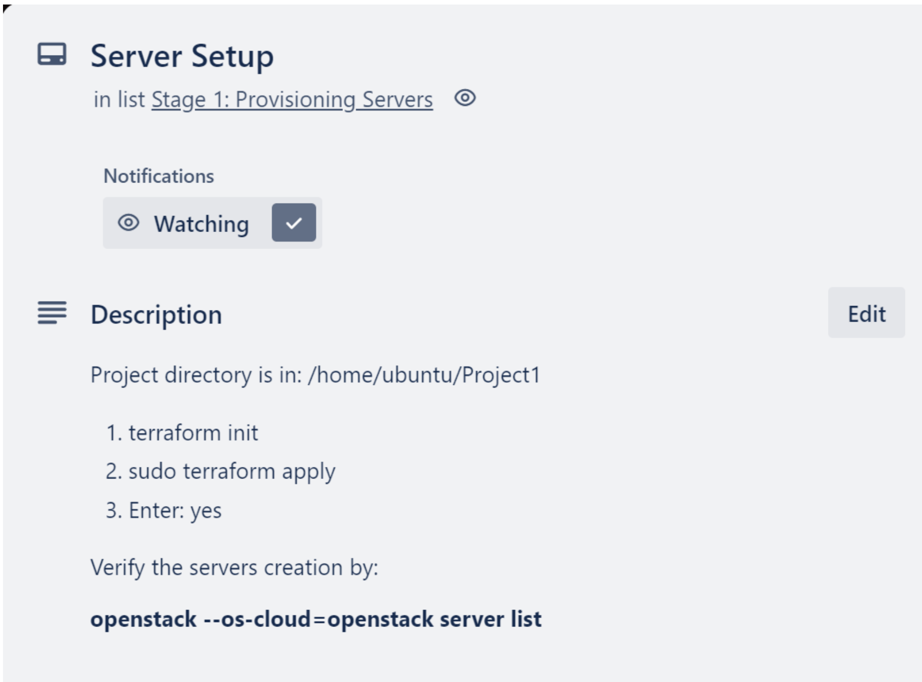
- We've successfully onboarded a partner group and added them to the Trello board.

5.2 Configuring access and permission:

- To facilitate their participation in the deployment process, we've obtained the junior member's public SSH key and added it to the authorized hosts on our master server
- We've shared the IP address of the master server with the partner group, enabling them to establish secure SSH connections for executing tasks.

5.3 Execution deployment tasks:

- Given that our partner group has selected the same project and is already familiar with it, there were no pre-deployment questions necessary.
- The junior followed the defined workflow steps, including provisioning servers, configuring server settings, and testing the deployed environment.
- We noted the start of this each task when the junior initiated the deployment tasks and noted the end time upon initiation completion.
    - Stage 1- Provisioning Servers: started at 10:13 and finished at 10:17 -> take 4 min



Server Setup

in list Stage 1: Provisioning Servers

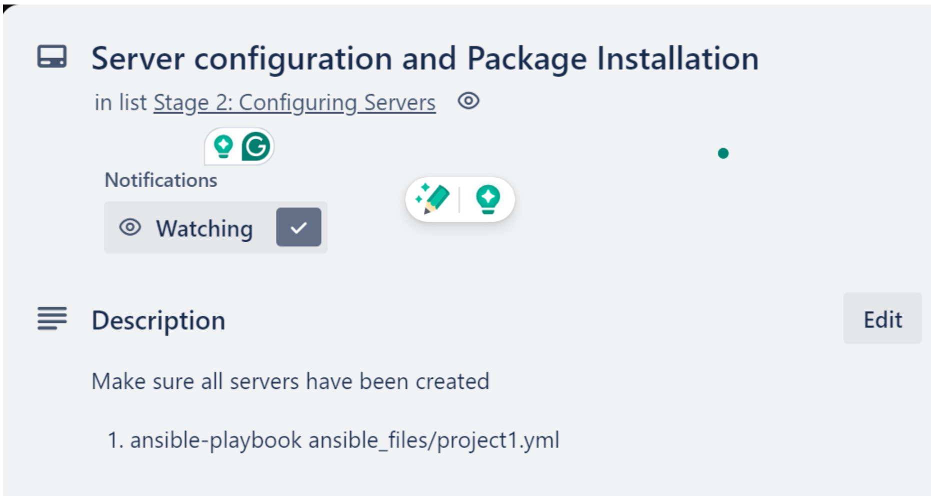
Project directory is in: /home/ubuntu/Project1

1. terraform init
2. sudo terraform apply
3. Enter: yes

Verify the servers creation by:

```
openstack --os-cloud=openstack server list
```

- Stage 2- Configuring Servers: started at 10:17 and finished at 10:21 -> take 4 min



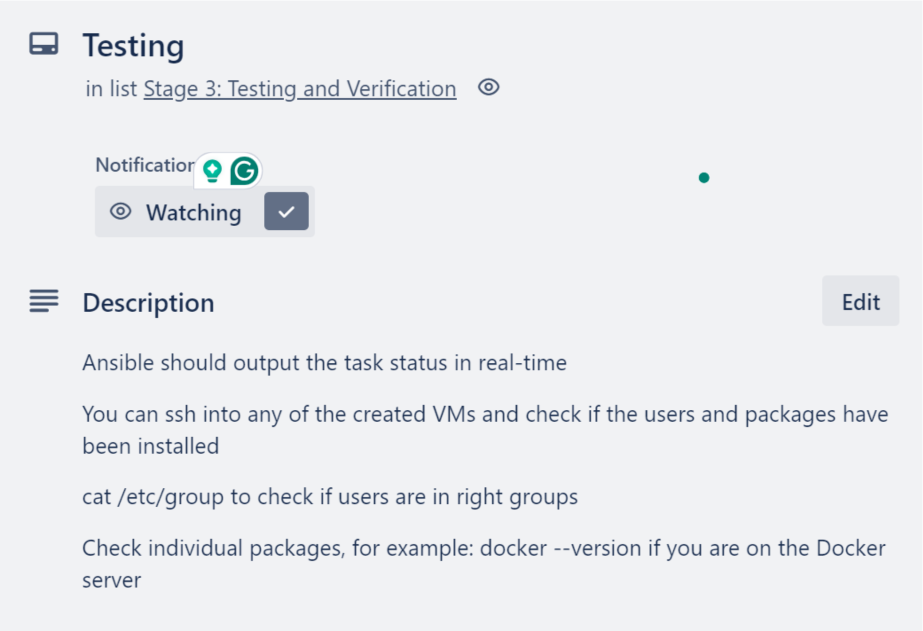
Server configuration and Package Installation

in list Stage 2: Configuring Servers

Make sure all servers have been created

1. ansible-playbook ansible_files/project1.yml

- Stage 3 - Testing and Verification: started at 10:22 and finished at 10:30 -> take 8 min



Testing

in list Stage 3: Testing and Verification

Description

Ansible should output the task status in real-time

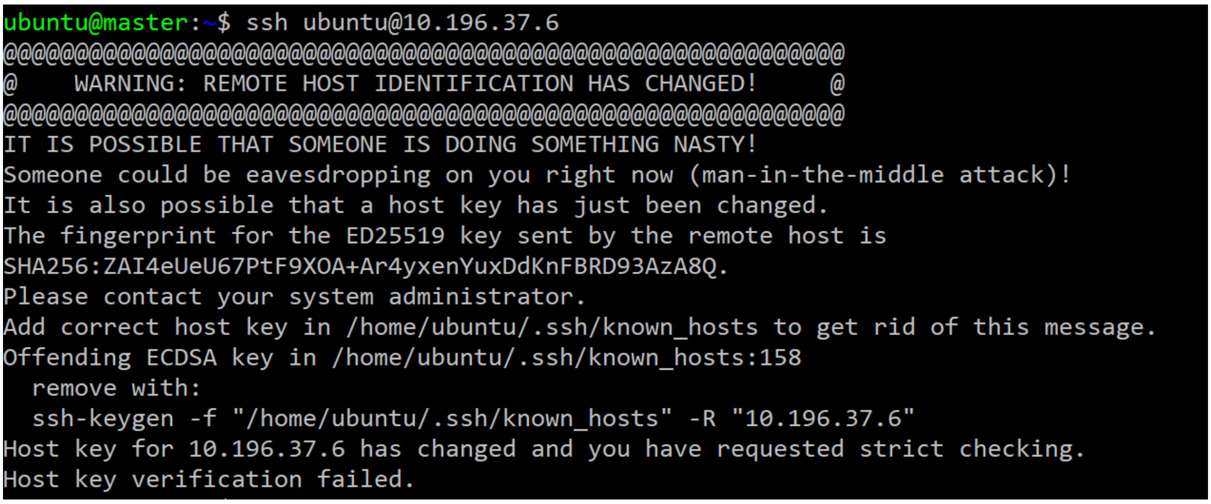
You can ssh into any of the created VMs and check if the users and packages have been installed

cat /etc/group to check if users are in right groups

Check individual packages, for example: docker --version if you are on the Docker server

5.4 Experiment results

- Stage 1- Provisioning Servers: all the servers were successfully created
- Stage 2- Configuring Servers: all the tasks were successfully completed
- Stage 3 - Testing and Verification: error when accessing one of the created servers (IP : 10.196.37.6). This warning typically occurs when the SSH key or fingerprint associated with the server has changed since the last time. We believe that happened due to “Server Rebuild”, the server was destroyed and rebuilt, it would have a new SSH key.



```
ubuntu@master:~$ ssh ubuntu@10.196.37.6
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ED25519 key sent by the remote host is
SHA256:ZAI4eUeU67PtF9XOA+Ar4yxenYuxDdKnFBRD93AzA8Q.
Please contact your system administrator.
Add correct host key in /home/ubuntu/.ssh/known_hosts to get rid of this message.
Offending ECDSA key in /home/ubuntu/.ssh/known_hosts:158
  remove with:
  ssh-keygen -f "/home/ubuntu/.ssh/known_hosts" -R "10.196.37.6"
Host key for 10.196.37.6 has changed and you have requested strict checking.
Host key verification failed.
```

This was resolved by running the command:

ssh-keygen -f “/home/ubuntu/.ssh/known_hosts” -R “10.196.37.6”