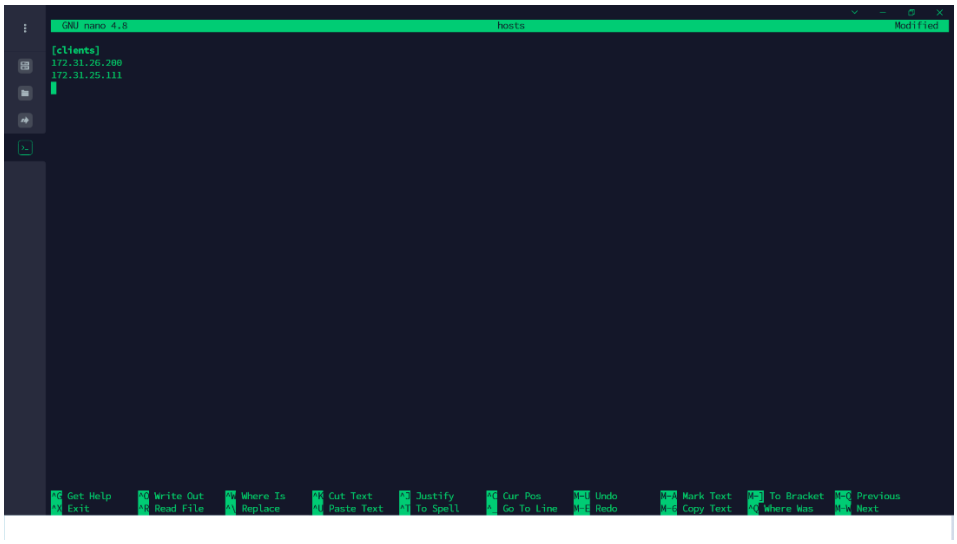
Ansible

Ansible is an open-source automation tool, or platform, used for IT tasks such as configuration management, application deployment, interservice orchestration, and provisioning. Automation is crucial these days, with IT environments that are too complex and often need to scale too quickly for system administrators and developers to keep up if they had to do everything manually. Automation simplifies complex tasks, not just making developers jobs more manageable but allowing them to focus attention on other tasks that add value to an organization. In other words, it frees up time and increases efficiency.

Ansible works by connecting to your nodes and pushing out small programs using, called "Ansible modules" to them. To do this it's necessary to have a SSH connection between the Ansible server machine and the ansible client machine.

It's necessary too, to create a file named hosts with the determinate group and the hosts IP's. In the picture below I show an example of hosts file with the group **clients**, specifying two different hosts IP's inside the group.



Example of Ansible module.

```
ansible -i hosts clients -m service -a "name=apache2 state=started"
```

This module initializes the apache2 service for the IP's corresponding to the group clients.

Ansible-Playbooks

Ansible-playbooks can contain a play or multiple plays and help you automate system configurations in a reusable set of tasks which will step through the implementation of your system into its desired state. Ansible playbook uses YAML sintax.

To define the plays, we use the word **tasks**.

```
tasks:
- name: ensure apache is installed and up to date
  apt: name=apache2 state=latest
```

The "**-name**" provides a clear description of the task we are performing. "**apt**" will install the service that we put in name and we define the version on the state.

Using **copy** module:

```
- name: write the apache config file
  copy: src=000-default.conf dest=/etc/apache2/sites-available/000-default.conf
  notify:
  - restart apache2
```

The value listed in notify specifies the name entry of a **handler** module we

will set up:

```
handlers:
- name: restart apache2
  service: name=apache2 state=restarted
```

This handler module respond to the notify "restart apache2" causing the apache2 service to restart.

After creating the playbook.yml we can run it with the **ansible-playbook** command:

```
Ansible-playbook -i hosts playbook.yml
```

Ansible Sintax in Finer detail

hosts – define the group of Ip's or you can define all;

remote_user - is the account that logs into the machine;

become – Ansible allows you to 'become' another user, different from the user that logged into the machine (remote user);

become_user - is to run a particular task as a specific user in general. To use become_user you should also set the become to yes.

Adding a variable in a playbook:

```
vars:
  mysql_root_password: password
```

This is an example of a variable called "mysql_root_password" and we stored the value "password" inside the variable.

Create a loop in a playbook:

```
tasks:
- name: install mysql and python-myslqdb
  apt: name={{ item }} update_cache=yes
cache_valid_time=3600 state=present
  with_items:
  - python3-mysqldb
  - mysql-server
```

We specify **{{ item }}** inside the name of we want to install and will get the packages in "**with-items**:" that is python3-mysqldb and mysql-server.