TP5 – Network Administration
- Ansible -

IFRIM Vasile-Alexandru
M2 SSI

The objective of this exercise is to create a deployment automation solution based on Ansible, using roles. As the automation manager, we are required to deploy a PHP web application using the CodeIgniter 4 PHP framework, on a remote server. In addition to CodeIgniter 4, this application requires a MySQL database for data storage.

We will create a modular, easy to reuse and maintain solution based on Ansible, using roles, that allows us to install and configure a web server (Apache or Nginx) with a CodeIgniter 4 site and a MySQL database.

For the environment we need to set up two machines, for which we will be using LXD containers. We do this through a script, `launch.sh`, where we first create a network for them. Because for now we only have these 2 servers, which need static IP addresses, we disable the DHCP service for them. To still grant them access the internet, we enable NAT for this network.

```
net_address="192.168.100.1"
net_netmask=24
net_name="net1"
lxc network create $net_name ipv4.nat=true ipv4.address=$net_address/$net_netmask ipv6.addres
s=none ipv4.dhcp=false ipv6.dhcp=false
containers="web db"
```

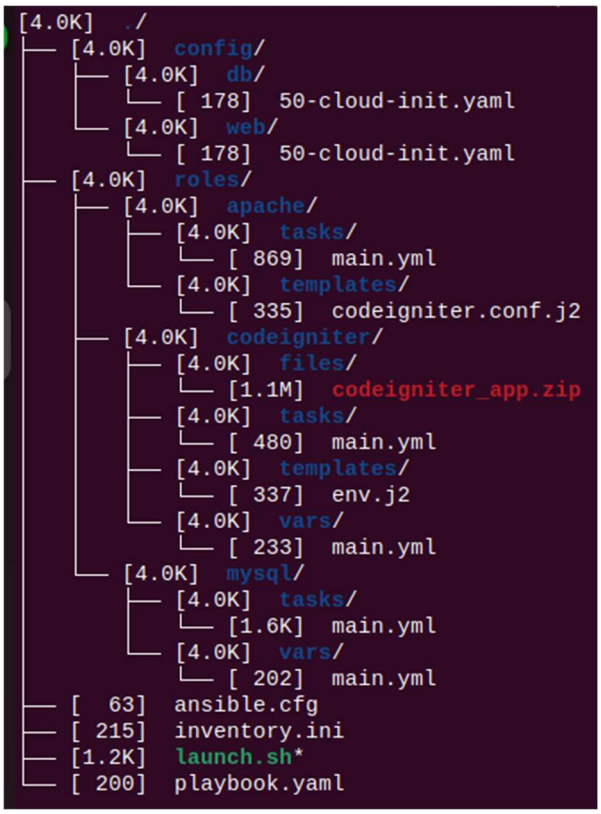
Then we create the containers and configure SSH access for the Ansible playbook. This involves accepting login as root and changing the root user's password so that Ansible can login through SSH, install the necessary dependencies, and any other required actions. These features, such as logging into SSH as root with password should be disabled, as they serious security concerns.

```
for cont in $containers;do
  echo "[$cont] Launch"
  lxc launch ubuntu:24.04 $cont

  echo "[$cont] Update"
  lxc exec $cont -- apt update
  lxc exec $cont -- apt install -y ssh

  echo "[$cont] Change root password"
  lxc exec $cont -- bash -c "echo 'root:passwd' | chpasswd"

  echo "[$cont] modify ssh root permissions"
  lxc exec $cont -- bash -c "sed -i 's/^#PermitRootLogin .*/PermitRootLogin yes/' /etc/ssh/ss
hd_config"
  lxc exec $cont -- bash -c "sed -i 's/^#PasswordAuthentication .*/PasswordAuthentication yes
/' /etc/ssh/sshd_config"
  lxc exec $cont -- bash -c "rm /etc/ssh/sshd_config.d/60-cloudimg-settings.conf"
  lxc exec $cont -- systemctl restart ssh
```

Finally, we reconfig their network from using the default lxdbr0 connected to each of them on eth0 (which we eliminate in the config), to using our custom network.

```
  echo "[$cont] config network"
  dir=$(pwd)
  lxc network attach $net_name $cont eth1
  lxc file push $dir/config/$cont/50-cloud-init.yaml $cont/etc/netplan/50-cloud-init.yaml --u
id 0 --gid 0 --mode 0644
  lxc exec $cont -- netplan apply
done
```

Below is the scheme of our deployment project. It is composed of a configuration file, `ansible.cfg`, a configuration folder for the two containers with YAML files for creating their network interfaces. The address of the web server will be 192.168.100.10 and that of the database will be 192.168.100.11.



The inventory file serves as a hosts file, through which Ansible is aware of its targets. The `playbook.yaml` file is also set to be automatically run at end of the `launch.sh` script. It is separated into three roles responsible for deploying Apache, CodeIgniter, and MySQL, respectively.

```yaml
---
- name: Deploy CodeIgniter4 webapp
  hosts: webservers
  become: true
  roles:
    - apache
    - codeigniter

- name: Config MySQL database
  hosts: dbservers
  become: true
  roles:
    - mysql
```

1 Apache

The role for deploying Apache starts with the task of installing the necessary packages for the app. PHP MySQL will be used to connect to the remote database. CodeIgniter will therefore use the `intl` and `mbstring` extensions, which need to be enabled in the `php.ini` file.

```yaml
- name: Install Apache, PHP, zip
  apt:
    name:
      - apache2
      - libapache2-mod-php
      - php
      - php-mysql
      - php-intl
      - php-mbstring
    state: present
    update_cache: true
```

The second task will push the necessary Jinja2 configuration for CodeIgniter, specifying that the root folder will be the public folder, for optimal security management, with a predefined `.htaccess` file. Next we activate the extensions needed, `intl` and `mbstring`, the site itself and the `rewrite` module, finishing with a service restart to apply these changes.

```yaml
- name: Copy Apache config for CodeIgniter
  template:
    src: codeigniter.conf.j2
    dest: /etc/apache2/sites-available/000-default.conf

- name: Activate extensions intl, mbstring
  lineinfile:
    path: /etc/php/{{ php_version }}/apache2/php.ini
    regexp: '^;?(extension=intl|extension=mbstring)$'
    line: 'extension=\1'
  vars:
    php_version: "8.3"
```

```yaml
- name: Activate the website and the rewrite module
  command: "{{ item }}"
  with_items:
    - a2ensite 000-default
    - a2enmod rewrite

- name: Restart Apache
  service:
    name: apache2
    state: restarted
```

2 CodeIgniter

This role has the responsability of, first, installing the zip package needed by the Ansible unarchive module to handle the app archive. We unzip the archive itself to `/var/www/html`.

```yaml
- name: Install zip
  apt:
    name:
      - zip
    state: present
    update_cache: true
```

```yaml
- name: Copy the CodeIgniter app archive
  unarchive:
    src: codeigniter_app.zip
    dest: /var/www/html/
    remote_src: no
```

Next task is to set the proper permissions and ownership on the application directory to the `www-data` user. Since Ansible connects using the root account via SSH, the application will be owned by it.

```
- name: Fix permissions
  file:
    path: /var/www/html/codeigniter
    owner: www-data
    group: www-data
    recurse: yes
```

For the last part, we've created an `env.j2` file, where we set the proper configuration, environment variables, and information needed when accessing the remote database. Just before being pushed to the web server, this file will automatically retrieve the values specified by us, separately, in `roles/codeigniter/vars/main.yaml`.

```
- name: Push the .env
  template:
    src: env.j2
    dest: /var/www/html/codeigniter/.env
    owner: www-data
    group: www-data
    mode: '0640'

- name: Restart Apache
  service:
    name: apache2
    state: restarted
```

```
CI_ENVIRONMENT={{ ci_environment }}
app.baseURL={{ app_base_url }}
database.default.hostname={{ db_hostname }}
database.default.database={{ codeigniter_db }}
database.default.username={{ codeigniter_user }}
database.default.password={{ codeigniter_password
database.default.DBDriver={{ db_driver }}
database.default.port={{ db_port }}
```

```
CI_ENVIRONMENT=production
app.baseURL=http://192.168.100.10/
database.default.hostname=192.168.100.11
database.default.database=codeigniter_db
database.default.username=codeigniter_user
database.default.password=passwd
database.default.DBDriver=MySQLi
database.default.port=3306
```

3 MySQL

The first steps are to install the PyMySQL python3 module, needed to connect to the MySQL database. Then, we configure the database's root user password.

```
---
- name: Install PyMySQL
  apt:
    name: python3-pymysql
    state: present

- name: Configure root password for MySQL
  debconf:
    name: mysql-server
    question: "mysql-server/root_password"
    value: "{{ mysql_root_password }}"
    vtype: "password"

- name: Confirm root password for MySQL
  debconf:
    name: mysql-server
    question: "mysql-server/root_password_again"
    value: "{{ mysql_root_password }}"
    vtype: "password"
```

Next we install the database itself, start& enable it to automatically start after the container boots. In the following task, we modify the listening interface of the MySQL service to the ‘any’ interface (0.0.0.0) so the web server can access it.

```
- name: Start and enable MySQL
  service:
    name: mysql
    state: started
    enabled: true

- name: Modify MySQL to accept external connections
  lineinfile:
    path: /etc/mysql/mysql.conf.d/mysqld.cnf
    regexp: '^bind-address'
    line: "bind-address = {{ mysql_bind_address }}"
    backup: yes
```

We continue by creating a database and a database user for CodeIgniter to use and connect with, from the web server. In MySQL, the "%" is a wildcard host specifier, informing the mysql_user module to allow the current user to connect from any host, without restricting to a specific hostname or IP.

```
- name: Create a db for CodeIgniter
  mysql_db:
    name: "{{ codeigniter_db }}"
    state: present
    login_user: root
    login_password: "{{ mysql_root_password }}"
    login_host: "localhost"
    login_unix_socket: "{{ mysql_socket_path | default('/var/run/mysqld/mysqld.sock') }}"

- name: Create a MySQL user for CodeIgniter
  mysql_user:
    name: "{{ codeigniter_user }}"
    host: '%'
    password: "{{ codeigniter_password }}"
    priv: "{{ mysql_privileges }}"
    state: present
    login_user: root
    login_password: "{{ mysql_root_password }}"
    login_host: "%"
    login_unix_socket: "{{ mysql_socket_path | default('/var/run/mysqld/mysqld.sock') }}"
```

```
mysql_root_password: 'passwd'
codeigniter_db: 'codeigniter_db'
codeigniter_user: 'codeigniter_user'
codeigniter_password: 'passwd'
mysql_privileges: 'codeigniter_db.*:ALL'
mysql_bind_address: '0.0.0.0'
```

roles/mysql/vars/main.yaml

4 Demo

After deploying the project, we can access the website from the host machine, or the database:


```
root Jan 21 22:36 ~ > mysql -h 192.168.100.11 -u codeigniter_user -p codeigniter_db
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.40-0ubuntu0.24.04.1 (Ubuntu)

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```