Deploying Web Applications using Ansible

Description

Ansible is a configuration management tool that you can use to configure servers and deploy applications. Ansible is free, easy to set up with few dependencies, and allows you to model complex configuration and deployment workflows.

Learning how to deploy a web application end to end will make you more proficient in using Ansible.

In this lab, you will learn how to create and configure an Ansible role, and you will create and configure a role that deploys a web application.

Learning Objectives

Upon completion of this intermediate-level lab, you will be able to:

- Use the Vim text editor to edit YAML files
- Create an Ansible role to configure the LAMP stack
- Separate variables, templates, and handlers to make your role more reusable
- Deploy a Python web application with Ansible

Lab steps

- Logging In to the Amazon Web Services Console
- Connecting to the Ubuntu Virtual Machine using EC2 Instance Connect
- Creating an Ansible Role to Install a LAMP Stack
- Setting Default Variables and Using Notifications
- Deploying a Web Application

Connecting to the Ubuntu Virtual Machine using EC2 Instance Connect

Introduction

You will use an Amazon EC2 instance as the host for Ansible in this lab.

In this lab step you will connect to an Ubuntu instance using EC2 Instance Connect and access a shell.

Connect to your instance dialog box, ensure the **EC2 Instance Connect** tab is selected and the **User name** is **ubuntu**

Creating an Ansible Role to Install a LAMP Stack

0/1

0 out of 1 validations checks passed

Introduction

This lab builds on the Getting Started with Ansible, opens in a new tab lab. In that lab, everything was put into one playbook file. That approach works, but it makes it difficult to reuse logic.

In this lab step, you will create an Ansible role that is easy to reuse.

Instructions

1. To create a new directory named roles, enter the following commands:

```
cd ~
```

```
mkdir roles
```

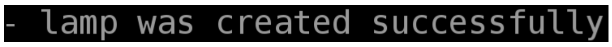
```
cd roles
```

These commands change to the home directory, create a new directory, and then change to it.

2. To create the skeleton of a new Ansible role, enter the following:

```
ansible-galaxy init lamp
```

You will see the following output:



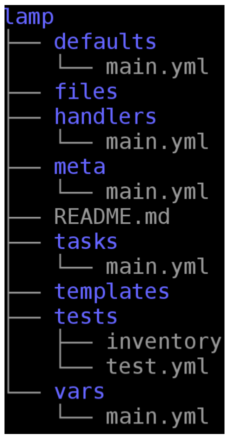
```
- lamp was created successfully
```

Ansible Galaxy is a tool for creating, installing, managing, and sharing Ansible roles. A full description of Ansible Galaxy is outside of the scope of this lab.

3. To see the directory structure of the role you created, enter the following command:

```
tree lamp
```

You will see the directory structure displayed:



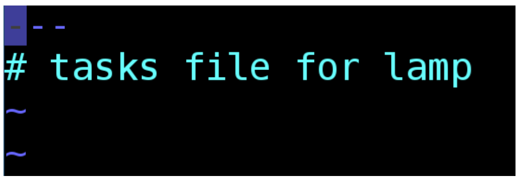
```
lamp
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── README.md
├── tasks
│   └── main.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml
```

The tree command is a tool for visually displaying directory structures in the terminal. This tool has been installed on the host for you.

4. To open a file in the Vim text editor, enter the following:

```
vim lamp/tasks/main.yml
```

You will see the following:



```
---
# tasks file for lamp
~
~
```

Vim is a modal text editor. This means it has different modes for different types of text editing functions. By default, you are in normal mode.

5. To remove the placeholder text, with your cursor at the top, press *dd* to delete each line until the file is empty.

You can use the arrow keys on your keyboard to move around in Vim.

6. Press *i* to enter Vim's insert mode.

At the bottom you will see the following to indicate that you are in insert mode

Note: If you don't see the mode change, press your *escape* key and then press *i* again.

7. To define some tasks, copy and paste the following into Vim:

```
- name: Install our packages
  apt:
    name: "{{ packages }}"
    state: present
    update_cache: true


- name: Confirm services are running
  service:
    name: "{{ item }}"
    state: started
  with_items: "{{ services }}"


- name: Enable Apache2 modssl

  shell: a2enmod ssl


- name: Enable Apache2 Default HTTPS site

  shell: a2ensite default-ssl


- name: Restart Apache

  service:

    name: apache2

    state: restarted
```

These tasks install packages, confirm that services are running, and configure the Apache web server to serve an HTTPS/SSL website.

8. To exit Vim's insert mode, press the *Escape* key on your keyboard.

Pressing escape returns you to Vim's normal mode.

9. To save the file and exit Vim, press *:* (colon) type *wq* and press *enter*.

You will be returned to the command-line prompt and see the following:

```
"lamp/tasks/main.yml" 22L, 448C written
```

You have defined a role that can install and configure a LAMP stack. However, you need a playbook to run your role.

10. To create a new file in your home directory, enter the following:

```
vim ~/app.yml
```

You will see Vim open in normal mode.

11. To define the code for your playbook, press *i* to enter insert mode, and copy and paste the YAML below:

Copy code

```yaml
---
- hosts: localhost
  gather_facts: false
  connection: local
  become: yes
  vars:
    packages:
      - apache2
      - mysql-server
      - mysql-common
      - mysql-client
      - libapache2-mod-wsgi
    services:
      - apache2
      - mysql
  roles:
    - lamp
```

Notice the roles property. It tells the playbook to include your lamp role.

Ansible knows to import the main.yml file from the tasks folder of your lamp role. This is conventional in Ansible.

12. To save the file and exit Vim, press *escape* press *:* (colon) type *wq* and press *enter*.

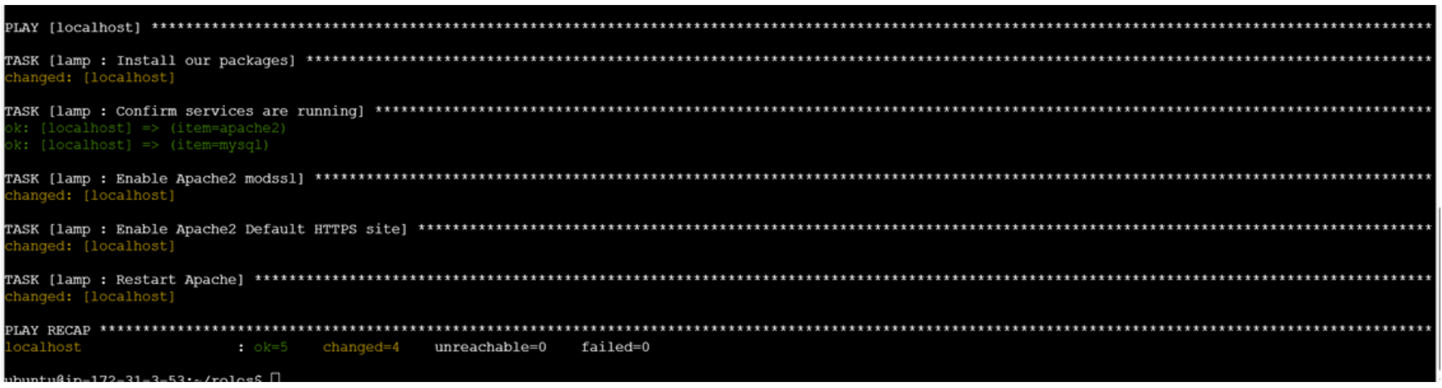
You will be returned to the command-line prompt.

13. To run your playbook, enter the following command:

```
ansible-playbook ~/app.yml
```

Note: This command may take a couple of minutes to complete.

You will see output that ends with:



```
PLAY [localhost] ***************************************************************************************************

TASK [lamp : Install our packages] *********************************************************************************
changed: [localhost]

TASK [lamp : Confirm services are running] *************************************************************************
ok: [localhost] => (item=apache2)
ok: [localhost] => (item=mysql)

TASK [lamp : Enable Apache2 modssl] ********************************************************************************
changed: [localhost]

TASK [lamp : Enable Apache2 Default HTTPS site] ********************************************************************
changed: [localhost]

TASK [lamp : Restart Apache] ***************************************************************************************
changed: [localhost]

PLAY RECAP *********************************************************************************************************
localhost                  : ok=5    changed=4    unreachable=0    failed=0

ubuntu@ip-172-31-3-53:~/roles$
```

You've created a role that will install and configure a LAMP stack on a host.

You could break this down further and have separate roles to install the web server and the database server. The exact structure you should use will depend upon the specific details of the task you want to accomplish. As a general rule, the more granular your roles are, the more reusable they are.

Summary

In this lab step, you created a role that installs the components needed for an application to run on the LAMP stack. You created a playbook that uses your role and you ran your playbook.

Setting Default Variables and Using Notifications

Introduction

In this lab step, you will make your role more reusable by separating different parts of it into separate files.

Instructions

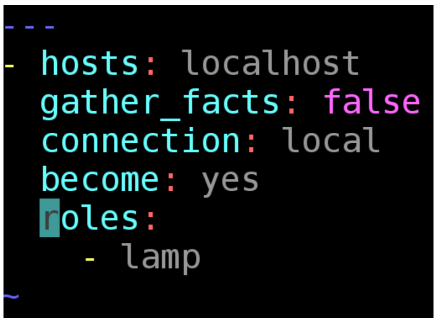
1. To open your playbook file, enter the following:

```
vim ~/app.yml
```

2. To delete the variables, move your cursor to the vars line, and press *dd* to delete each variable line.

Note: You can use the arrow keys on your keyboard to move the cursor in Vim.

Your app.yml file will look like this:



```
---
- hosts: localhost
  gather_facts: false
  connection: local
  become: yes
  roles:
    - lamp
~
```

3. To save the file and exit Vim, press your *escape* key, press *:* (colon) type *wq* and press *enter*.

4. To open the defaults file, enter the following:

```
vim ~/roles/lamp/defaults/main.yml
```

5. Remove the placeholder by pressing *dd* and replace it with the following YAML by pressing *i* and pasting the folowing:

```
packages:
  - apache2
  - mysql-server
  - mysql-common
  - mysql-client
services:
  - apache2
  - mysql
```

6. To save the file and exit Vim, press your *escape* key, press *:* (colon) type *wq* and press *enter*.

7. To re-run your playbook, enter the following:

```
ansible-playbook ~/app.yml
```

You will see the same output as before. This time the variables are defined in a separate file which makes reuse easier.

Next, you will make the Apache restart task into a handler.

8. To open the handlers file in Vim, enter the following:

```
vim ~/roles/lamp/handlers/main.yml
```

9. Press *dd* to delete each line of placeholder text and press *i* to enter insert mode.

10. Paste the following YAML into the handlers/main.yml file:

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

This is the same as the restart Apache task that you added to the tasks.yml file.

A handler is a task that only gets run when notified. In this case, Apache will only be restarted when something has changed.

11. To save the file and exit Vim, press your *escape* key, press *:* (colon) type *wq* and press *enter*.

12. To open the main tasks file, enter the following:

```
vim ~/roles/lamp/tasks/main.yml
```

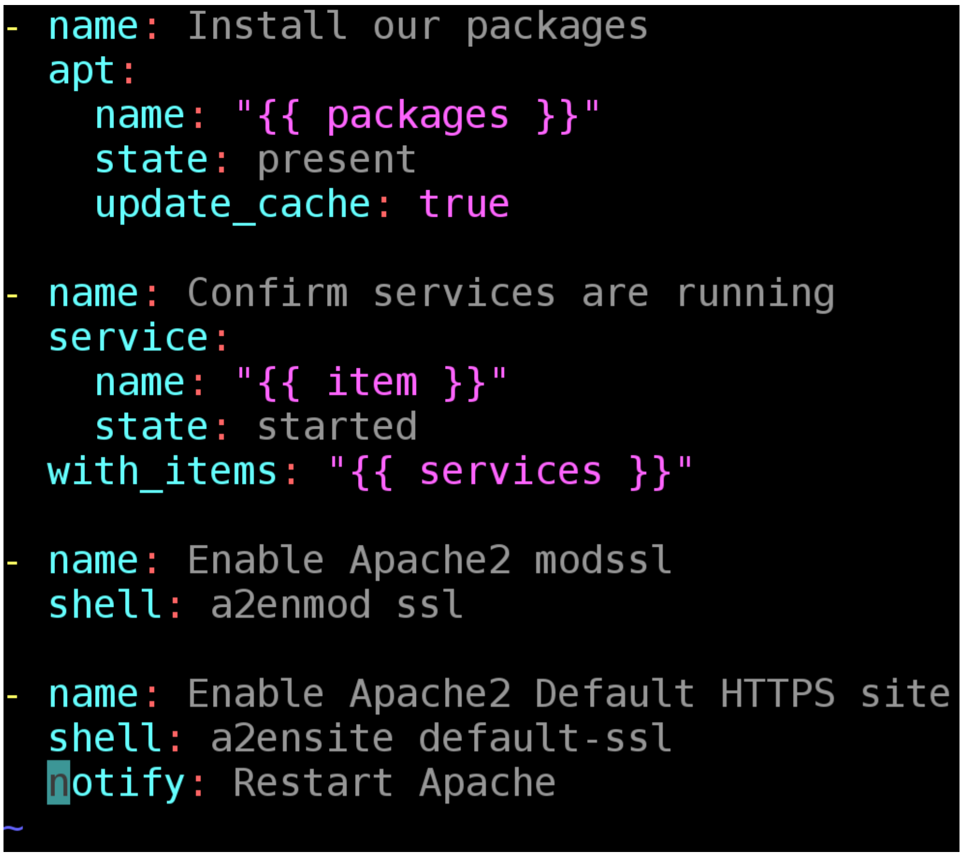
13. At the bottom of the file, remove the last Restart Apache task by moving your cursor and entering *dd* for each line.

14. To notify the handler, add the following line to the Enable Apache2 Default HTTPS site task:

```
notify: Restart Apache
```

Note: Remember to press *i* to enter insert mode when adding the line.

Your tasks/main.yml file should look like this:

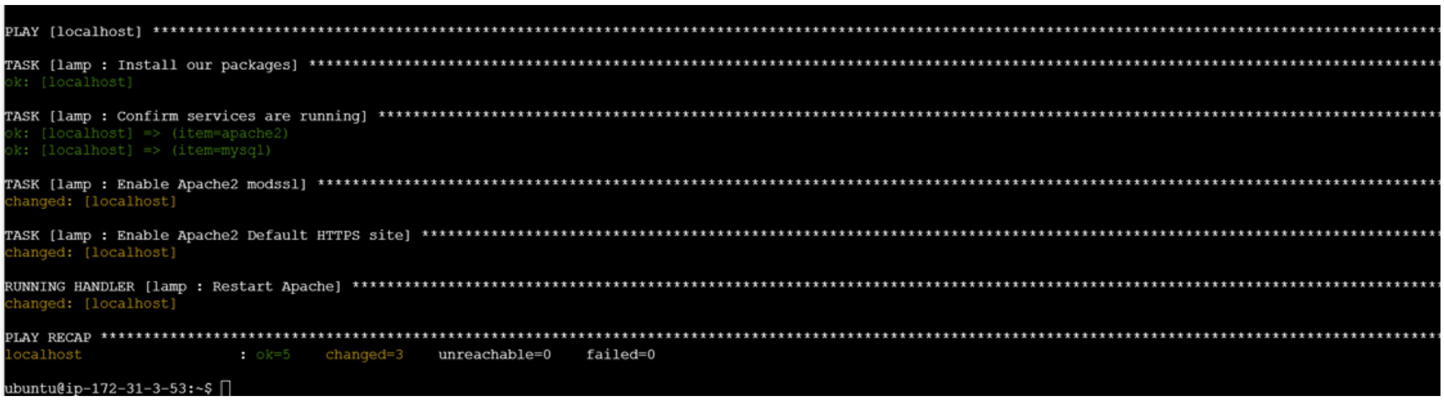


```
- name: Install our packages
  apt:
    name: "{{ packages }}"
    state: present
    update_cache: true

- name: Confirm services are running
  service:
    name: "{{ item }}"
    state: started
  with_items: "{{ services }}"

- name: Enable Apache2 modssl
  shell: a2enmod ssl

- name: Enable Apache2 Default HTTPS site
  shell: a2ensite default-ssl
  notify: Restart Apache
~
```

15. To save the file and exit Vim, press your *escape* key, press *:* (colon) type *wq* and press *enter*.
16. To re-run your playbook, enter the following:

```
ansible-playbook ~/app.yml
```

Observe the output and look for the **RUNNING HANDLER** line:



```
PLAY [localhost] ****************************************************************************************************

TASK [lamp : Install our packages] **********************************************************************************
ok: [localhost]

TASK [lamp : Confirm services are running] **************************************************************************
ok: [localhost] => (item=apache2)
ok: [localhost] => (item=mysql)

TASK [lamp : Enable Apache2 modssl] *********************************************************************************
changed: [localhost]

TASK [lamp : Enable Apache2 Default HTTPS site] *********************************************************************
changed: [localhost]

RUNNING HANDLER [lamp : Restart Apache] *****************************************************************************
changed: [localhost]

PLAY RECAP **********************************************************************************************************
localhost                  : ok=5    changed=3    unreachable=0    failed=0

ubuntu@ip-172-31-3-53:~$ 
```

This shows that the handler you defined ran successfully.

Summary

In this lab step, you separated the variables into a separate file so that the default variables can be easily overridden. You created a handler to allow for the Apache web server service to be restarted if a task notifies the handler.

Deploying a Web Application

Introduction

In this lab step, you will create an Ansible role to install a web application on the LAMP host.

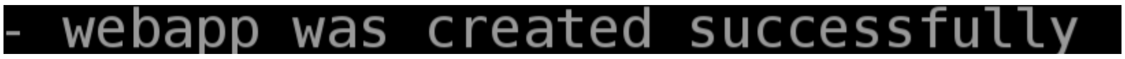
Instructions

1. To create a new role, enter the following commands:

```
cd ~/roles
```

```
ansible-galaxy init webapp
```

You will see the output:



```
- webapp was created successfully
```

You have created a new Ansible role for deploying a web application.

2. To open the defaults file in Vim, enter the following

```
vim ~/roles/webapp/defaults/main.yml
```

3. Delete the placeholder content (press *dd* for each line), and add the following YAML (press *i* to enter insert mode):

```
---

app_download_dest: /tmp/webapp

app_dest: /var/www/webapp

app_repo: https://github.com/cloudacademy/ansible_demo.git
```

These variables define directories to store files temporarily and to store the web application. The app_repo variable points to a GitHub repository that holds the source code for a Python web application.

4. To save your file and exit Vim, press *escape* press *:* type *wq* and press *enter*.

5. To create a new task file to the web application's database, enter the following:

```
vim ~/roles/webapp/tasks/database.yml
```

6. Add the following to define tasks for setting up a database:

```
- apt: name=python-mysqldb state=present
- mysql_user: name=appuser password=94nfsUl7 priv=*.*:ALL state=present
```

```
- mysql_db: name=appdata state=present
```

7. To save your file and exit Vim, press *escape* press *:* type *wq* and press *enter*.

8. To create a task file for the application, enter the following

Copy code

```
vim ~/roles/webapp/tasks/app.yml
```

9. Press *i* to enter insert mode and add the following:

```
- apt: name=libmysqlclient-dev state=present

- apt: name=python-pip state=present

- git: repo={{app_repo}} dest="{{app_download_dest}}"

- pip: requirements={{app_download_dest}}/app/requirements.txt

- copy: src={{app_download_dest}}/app/ dest={{app_dest}}
```

These tasks use variables that you defined in the defaults file. Notice that they are enclosed in double braces (also known as curly brackets).

Next, you will create an Apache config file template.

10. To save your file and exit Vim, press *escape* press *:* type *wq* and press *enter*.

11. To open a new template file, enter the following:

```
vim ~/roles/webapp/templates/apache.conf
```

12. Press *i* to enter insert mode and paste the following:

```
<VirtualHost *>

  ServerName {{inventory_hostname}}



  WSGIDaemonProcess webapp user=ubuntu group=ubuntu threads=5

  WSGIScriptAlias / {{app_dest}}/wsgi.py



  <Directory {{app_dest}}>
```

```
        WSGIProcessGroup webapp

        WSGIApplicationGroup %{GLOBAL}

        Require all granted

    </Directory>

</VirtualHost>
```

Notice the use of variables again here in the template. You may recall that you set the defaults for this role and it didn't include the "**inventory_hostname**." And that's because that is an Ansible provided variable. In this case, it will say "localhost" in the rendered template. While facts are out of scope for this lab, if you were to enable fact gathering you'd have access to even more variables.

13. To save your file and exit Vim, press *escape* press *:* type *wq* and press *enter*.

14. To open a new task file for the Apache site, enter the following:

```
vim ~/roles/webapp/tasks/site.yml
```

15. Add the following to the file:

```
- apt: name=libapache2-mod-wsgi state=present

- name: Copy the apache configuration file

  template:

    src: apache.conf

    dest: /etc/apache2/sites-available/000-default.conf

  notify: Restart Apache
```

Next, you will edit the main.yml file so it includes these three task files.

16. Open the following file and add replace the contents with the following lines of YAML;

```
vim ~/roles/webapp/tasks/main.yml
```

YAML:

Copy code

```
- include: database.yml

- include: app.yml
```

`- include: site.yml`

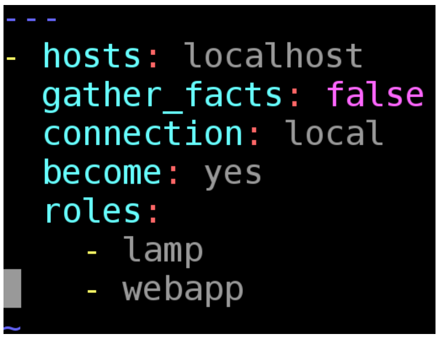
You have completed defining the webapp role. However, your playbook needs to be updated to run the webapp role. You will do this next.

17. To open your playbook, enter the following:

`vim ~/app.yml`

18. Move the cursor to the roles, press *i* to enter insert mode, and add a second role called webapp.

You playbook file should look like this:



```
---
- hosts: localhost
  gather_facts: false
  connection: local
  become: yes
  roles:
    - lamp
    - webapp
~
```

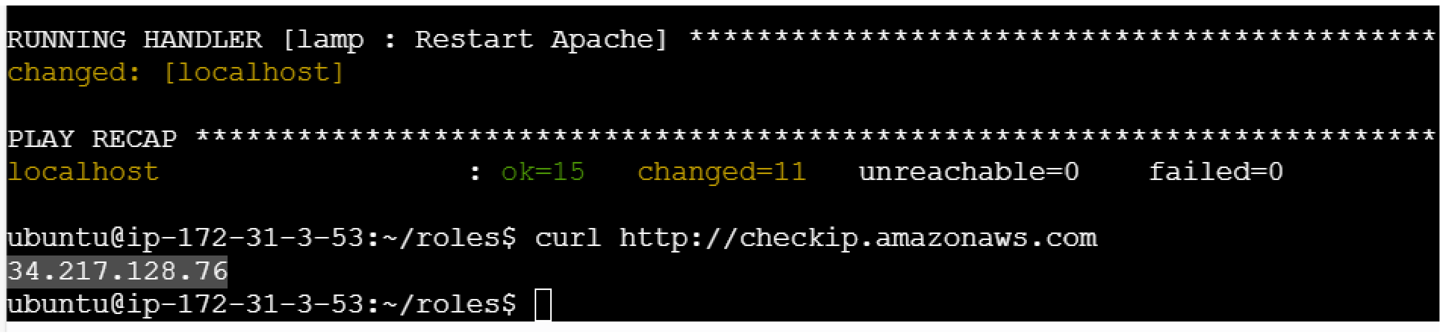
19. To run your playbook, enter the following:

`ansible-playbook ~/app.yml`

You will see output ending with:

20. To see the IP address of the server, enter the following:

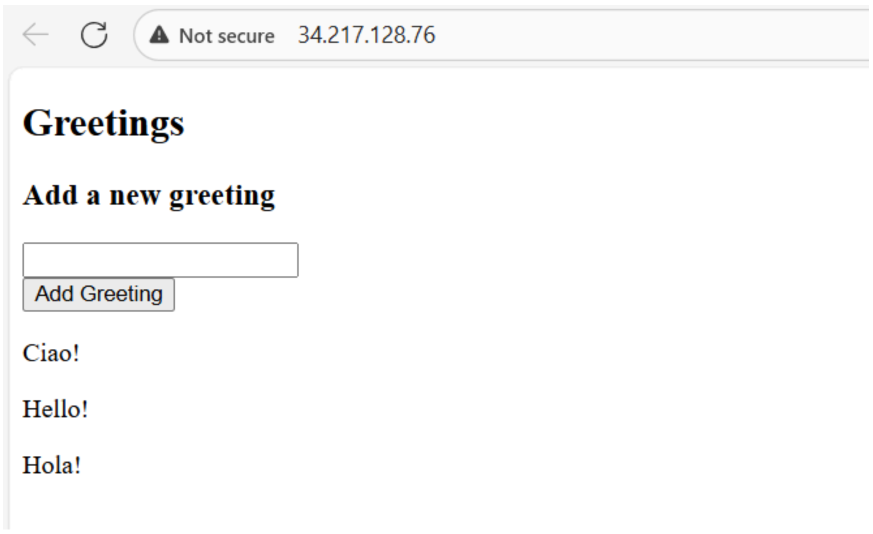
`curl http://checkip.amazonaws.com`



```
RUNNING HANDLER [lamp : Restart Apache] *********************************************
changed: [localhost]

PLAY RECAP **************************************************************************
localhost                  : ok=15    changed=11    unreachable=0    failed=0

ubuntu@ip-172-31-3-53:~/roles$ curl http://checkip.amazonaws.com
34.217.128.76
ubuntu@ip-172-31-3-53:~/roles$ 
```

21. In a new browser tab, navigate to the IP address of the server.

You will see the web application displayed:


Not secure 34.217.128.76

Greetings

Add a new greeting

Add Greeting

Ciao!

Hello!

Hola!

Note: If you see the default Apache page instead of the Greetings web application, check that you added the webapp role to your app.yml file.

Summary

In this lab, you learned how to create a reusable Ansible role that configures a server to use the LAMP stack. You learned how to separate variables and handlers in an Ansible role. And you created a second role that installs a Python web application, including configuring a new Apache site and setting up a Mysql database.