Overview

Configuration management is an important methodology we use in IT to build and maintain systems in our organizations. It helps us manage change more effectively by leveraging key infrastructure as code principles. Instead of managing systems with a complex graphical program storing data hidden away in a proprietary database, we can use simple code scripts that are easy to read and place under version control. In this assignment you will work with Ansible, one of the leading configuration management tools in the marketplace.

Requirements

You need to have a personal AWS account and GitHub account for this assignment.

The assignment

Let's start managing configuration!

Launch three instances

You will need to launch three EC2 instances for this assignment. We will configure the first instance as a management server with the Ansible tool installed. The second instance will become a webserver, and the third instance will become a MySQL database server.

Fortunately you can use a CloudFormation template to rapidly create the infrastructure resources needed for this assignment. The template is available here: AnsibleSystems.json

Take a quick look at the template in a text editor. Notice how the UserData property for the `mgmt1` instance is configured. When CloudFormation launches this stack, it will automatically install and configure Ansible software on the management server. It's very common to use a small amount of scripting code to bootstrap configuration management software onto a new system. Once Ansible is installed it can be used to install and configure other software packages.

Launch a new stack using the `AnsibleSystems.json` template into the us-east-1 region. Remember to specify your server keypair name when launching the stack. You will also need to provide the current IP address of your workstation in CIDR format (x.x.x.x/32).

You can access the website http://checkip.amazonaws.com in a browser to quickly determine the IP address of your workstation. Remember that if you move your workstation from one network to another (like from home to work) you will need to update this workstation IP address in the CloudFormation stack by performing an update on the stack.

You can give the stack any name you like. When you create the stack, you will need to click a checkbox acknowledging the creation of IAM resources.

You will use the Ansible software installed on `mgmt1` to configure the `web1` and `database1` systems. Ansible communicates with remote hosts using the SSH protocol which means that the proper access credentials need to be in place on `mgmt1` in order to establish shell connections with the two other servers. Fortunately, the CloudFormation stack automates the setup of ssh credentials on all of the hosts.

Test Ansible

Log into the `mgmt1` instance and create a new directory in your home directory called `assignment9`. Initialize a new Git repository in this directory.

Ansible uses an inventory file to track managed hosts. The inventory file specifies how Ansible should connect to a remote host. It usually contains the IP address or fully-qualified domain name of a host, as well as the username and key required to access a host. The `hosts` file was dynamically created by an EC2 userdata script, so it already contains the IP addresses of the `web1` and `database1` instances. Take a look at the hosts file located at `/etc/ansible/hosts`.

You can verify that Ansible can successfully access the managed hosts by typing the following commands:

```
$ ansible all -m ping
```

You should see a response from Ansible that looks like this:

```
web1 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
database1 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

If Ansible returns a successful `pong` for each host then you are ready to move on!

Configure servers

You will create an Ansible playbook to configure the two remote hosts. Before beginning the playbook, take a look at the facts that Ansible can collect from a remote host:

```
$ ansible web1 -m setup
```

You will need to use a fact in your playbook, so take note of the fact variables that are returned by the setup module.

You can create a playbook file on the `mgmt1` server using a local text editor like nano or vi. Alternatively, you can use your favorite source code editor on your local workstation and copy

(scp) the file from your local workstation to the `assignment9`directory on `mgmt1`. Use whichever method is easiest for you.

Create a playbook called `playbook.yml` in the `/home/ec2-user/assignment9` directory to perform the following tasks on all of the hosts:

- Update all the current software packages on the system.
- Create a variable called `administrator_name` with a value of `corpadmin` in the play.
- Create a user with the value of the `administrator_name` variable on each system with a password of your choosing.

The playbook will perform the following tasks on the `web1` server:

- Install the Nginx server package.
- Copy the following `default.conf.j2` template to the system directory `/etc/nginx/conf.d/default.conf` substituting the text `example.com` with the fully-qualified domain name (fqdn) of the webserver (using a fact variable). Note, the fqdn of the server isn't `web1`. AWS automatically generates a domain address based on the private IP address of the instance.

```
server {
 listen       80;
        server_name  example.com;
        location / {
        root   /var/www/example.com/public_html/;
        index  index.html index.htm;
        }
  }
```

- Ensure the nginx service is running and will automatically start during the system boot.

The playbook will perform the following tasks on the `database1` server:

- Install the mysql server package.
- Configure the system to ensure mysql is running and will automatically start during the system boot.
- Use a loop to create 5 directories called:
    - /var/data/client1
    - /var/data/client2
    - /var/data/client3
    - /var/data/client4
    - /var/data/client5

Warning	Your playbook should only contain one software update task and one task to create a user called `corpadmin`. The play should apply each one of these tasks to multiple systems.

The `ec2-user` user doesn't have the necessary permissions to install new services on the instance. Ansible will need to become a super-user on these systems in order to install new packages.

Execute the playbook to configure the two hosts. If you encounter any error messages, review the playbook script and correct any mistakes.

This set of configuration tasks doesn't setup a real web application since there is no application code deployed. You can manually verify that the nginx and mysql services are running on each instance by using the `service` command. For example:

```
$ sudo service nginx status
```

You should confirm that all the configuration tasks have been applied to each of the servers before submitting your work.

Save your work

Create a new GitHub Classroom repository by clicking on this link:
https://classroom.github.com/a/jrxTH5d4

Commit your git repository (in the `assignment9` directory) to this repository.

Check your work

Here is what the contents of your git repository should look like before final submission:

```
├ default.conf.j2
└ playbook.yml
```

Terminate application environment

The last step in the assignment is to delete all the AWS services you created. Go to the CloudFormation dashboard, select your running stack, and choose the delete option. Watch as CloudFormation deletes all the resources previously created.

Submitting your assignment

I will review your published work on GitHub after the homework due date.