**Deploy aws instance using Ansible**

**Objective**

This tutorial will walk you through the process of using Ansible, an agent-less automation tool, to create instances on AWS. The Ansible playbook we will use is relatively simple; you can use it as a base to experiment with more advanced features.

Ansible is written in Python and is installed as a Python module on the control host. The only requirement for the hosts managed by Ansible is the ability to login with SSH. There is no requirement to install any software on the host managed by Ansible.

**Prerequisites**

 You must have an existing AWS account.

 You must have access to your AWS Access and Secret keys.

 You are responsible for all AWS costs incurred.

**Scope**

This tutorial was tested using the following environment and components:

 Amazon Web Services

 Anaconda 21.48.22.134-1 (Python 2.7.5)

 Ansible 2.6.3

## Steps

## You need to create a directory for your Ansible playbook. I prefer to create my project directories in ~/Development.

1. mkdir ~/Development/ansible-aws
2. cd ~/Development/ansible-aws

### **Install Ansible module**

If you use the Anaconda version of Python, you already have access to Ansible. If you are not using Anaconda, then you can usually install Ansible using the following command:

pip install ansible

### **Overview of our Ansible playbook**

Our playbook is relatively simple. It consists of a single inventory file, single group\_vars file and a single playbook file. Here is the layout of the file and directory structure:

1. +- ansible-aws/
2. |
3. +- group\_vars/
4. | +- all
5. |
6. +- inventory/
7. | +- hosts
8. |
9. +- playbooks/
10. | +- ansible-aws.yml

#### group\_vars/all

You can use variables in your playbooks using the {{variable name}} syntax. These variables are populated based on values stored in your variable files. You can explicitly load variable files in your playbooks.

However, all playbooks will automatically load the variables in the group\_vars/all variable file. The all variable file is loaded for all hosts regardless of the groups the host may be in. In our playbook, we are placing our AWS configuration values in the all file.

Edit the group\_vars/all file. Copy and paste the following text into the file:

1. aws\_access\_key: <enter AWS access key>
2. aws\_secret\_key: <enter AWS secret key>
3. key\_name: <enter private key file alias name>
4. aws\_region: <enter AWS region>
5. vpc\_id: <enter VPC ID>
6. ami\_id: ami-6d1c2007
7. instance\_type: m4.2xlarge
8. my\_local\_cidr\_ip: <enter cidr\_ip>
9.  aws\_access\_key: You need to enter your AWS Access key
10.  aws\_secret\_key: You need to enter your AWS Secret key
11.  key\_name: The alias name you gave to the AWS private key which you will use to SSH into the instances. In my case I created a key called ansible.
12.  aws\_region: The AWS region where you want to deploy your instances. In my case I am using us-east-1.
13.  vpc\_id: The specific VPC in which you want to place your instances.
14.  ami\_id: The specific AMI you want to deploy for your instances. The ami-6d1c2007 AMI is a CentOS 7 image.
15.  instance\_type: The type of AWS instance. For deploying Hadoop, I recommend at least m4.2xlarge. A faster alternative is c4.4xlarge.
16.  my\_local\_cidr\_ip: Your local computer's CIDR IP address. This is used for creating the security rules that allow your local computer to access the instances. An example CIDR format is 192.168.1.1/32. Make sure this set to your computer's public IP address.
17. After you have entered your appropriate settings, save the file.

#### inventory/hosts

Ansible requires a list of known hosts against which playbooks and tasks are run. We will tell Ansible to use a specific host file with the -i inventory/hosts parameter.

Edit the inventory/hosts file. Copy and paste the following text into the file:

1. [local]
2. localhost ansible\_python\_interpreter=/Users/myoung/anaconda/bin/python
3.  local]: Defines the group the host belongs to. You have the option for a playbook to run against all hosts, a specific group of hosts, or an individual host. This AWS playbook only runs on your local computer. That is because it uses the AWS APIs to communicate with AWS.
4.  localhost: This is the hostname. You can list multiple hosts, 1 per line under each group heading. A host can belong to multiple groups.
5.  ansible\_python\_interpreter: Optional entry that tells Ansible which specific version of Python to run. Because I am using Anaconda Python, I've included that setting here.

### **playbooks/ansible-aws.yml**

The playbook is where we define the list of tasks we want to perform. Our playbook will consist of 2 tasks. The first task is to create a specific AWS Security Group. The second tasks is to create a specific configuration of 6 instances on AWS.

Edit the file playbooks/ansible-aws.yml. Copy and paste the following text into the file

---

# Basic provisioning example

- name: Create AWS resources

hosts: localhost

connection: local

gather\_facts: False

tasks:

- name: Create a security group

ec2\_group:

name: ansible

description: "Ansible Security Group"

region: "{{aws\_region}}"

vpc\_id: "{{vpc\_id}}"

aws\_access\_key: "{{aws\_access\_key}}"

aws\_secret\_key: "{{aws\_secret\_key}}"

rules:

- proto: all

cidr\_ip: "{{my\_local\_cidr\_ip}}"

- proto: all

group\_name: ansible

rules\_egress:

- proto: all

cidr\_ip: 10.1.0.0/16

register: firewall

- name: Create an EC2 instance

ec2:

aws\_access\_key: "{{aws\_access\_key}}"

aws\_secret\_key: "{{aws\_secret\_key}}"

key\_name: "{{key\_name}}"

region: "{{aws\_region}}"

group\_id: "{{firewall.group\_id}}"

instance\_type: "{{instance\_type}}"

image: "{{ami\_id}}"

wait: yes

volumes:

- device\_name: /dev/sda1

volume\_type: gp2

volume\_size: 2

delete\_on\_termination: true

exact\_count: 3

count\_tag:

Name: aws-demo

instance\_tags:

Name: aws-demo

register: ec2

This playbook uses the Ansible ec2 and ec2\_group modules. You can read more about the options available to those modules here:

 [ec2](http://docs.ansible.com/ansible/ec2_module.html)

 [ec2\_group](http://docs.ansible.com/ansible/ec2_group_module.html)

The task to create the EC2 security group creates a group named ansible. It defines 2 ingress rules and 1 egress rule for that security group. The first ingress rule is to allow all inbound traffic from any host in the security group ansible. The second ingress rule is to allow all inbound traffic from your local computer IP address. The egress rule allows all traffic out from all of the hosts.

The task to create the EC2 instances creates 6 hosts because of the exact\_count setting. It creates a tag called hadoop-demoon each of the instances and uses that tag to determine how many hosts exists. You can chose to use smaller number of hosts.

You can specify volumes to mount on each of the instances. The default volume size is 8 GB and is too small for deploying Hadoop later. I recommend setting the size to at least 100 GB as above. I also recommend you set delete\_on\_termination to true. This will tell AWS to delete the storage after you have deleted the instances. If you do not do this, then storage will be kept and you will be charged for it.

### Running the Ansible playbook

Now that our 3 files have been created and saved with the appropriate settings, we can run the playbook. To run the playbook, you use the ansible-playbook -i inventory/hosts playbooks/ansible-aws.yml command. You should see something similar to the following:

1. $ ansible-playbook -i inventory/hosts playbooks/ansible-aws.yml
2. PLAY [Create AWS resources] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
3. TASK [Create a security group] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
4. changed: [localhost]
5. TASK [Create an EC2 instance] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
6. changed: [localhost]
7. PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
8. localhost : ok=2 changed=2 unreachable=0 failed=0

### Check AWS console.