Lesson 11 Demo 6: Provisioning EC2

This section will guide you to:

* Implement modules through command line

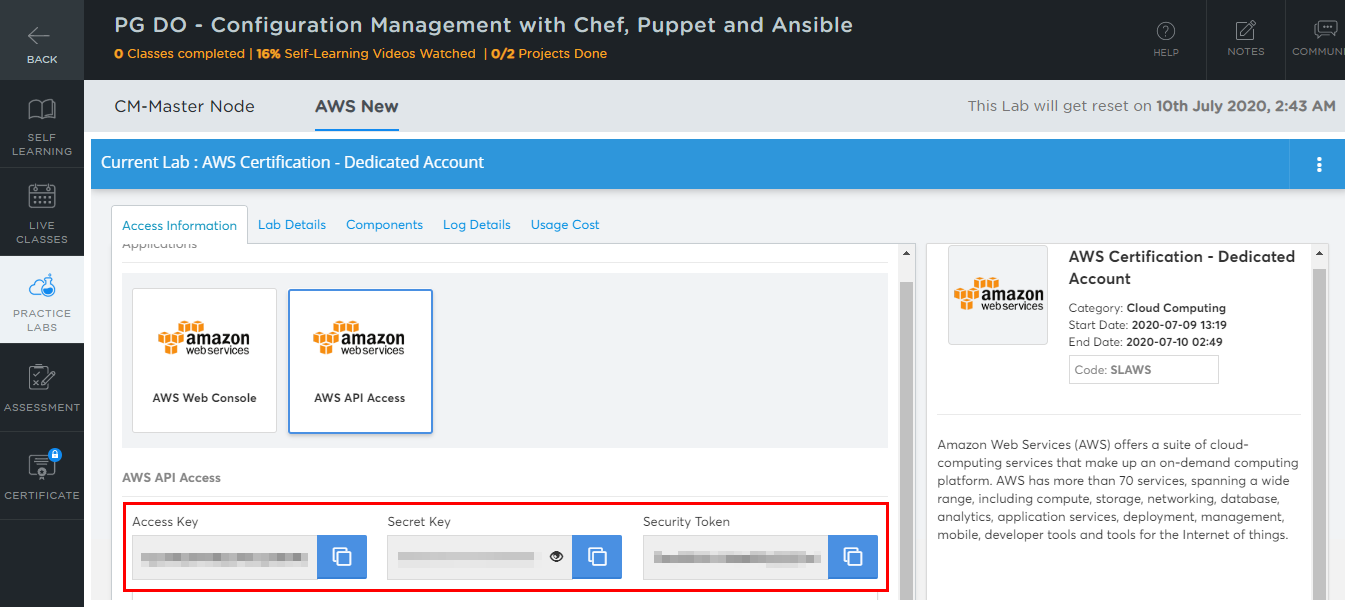
This lab has three subsections, namely:

1. Preparing the machine
2. Preparing Ansible playbook
3. Running the playbook

***Note:*** *Ansible 2.2.3 is already installed in your lab.*

**Prerequisite:**

* Establish an SSH connection between master and slave node. Your master node is your VM Lab and slave node is an EC2 instance created in AWS Lab.
* Use the AWS access credentials provided in the AWS API Access tab in your LMS as shown in the screenshot below:



**Note:** AWS access credentials will change when the AWS Lab session expires after every four hours.

### **Step 1:** Preparing the machine

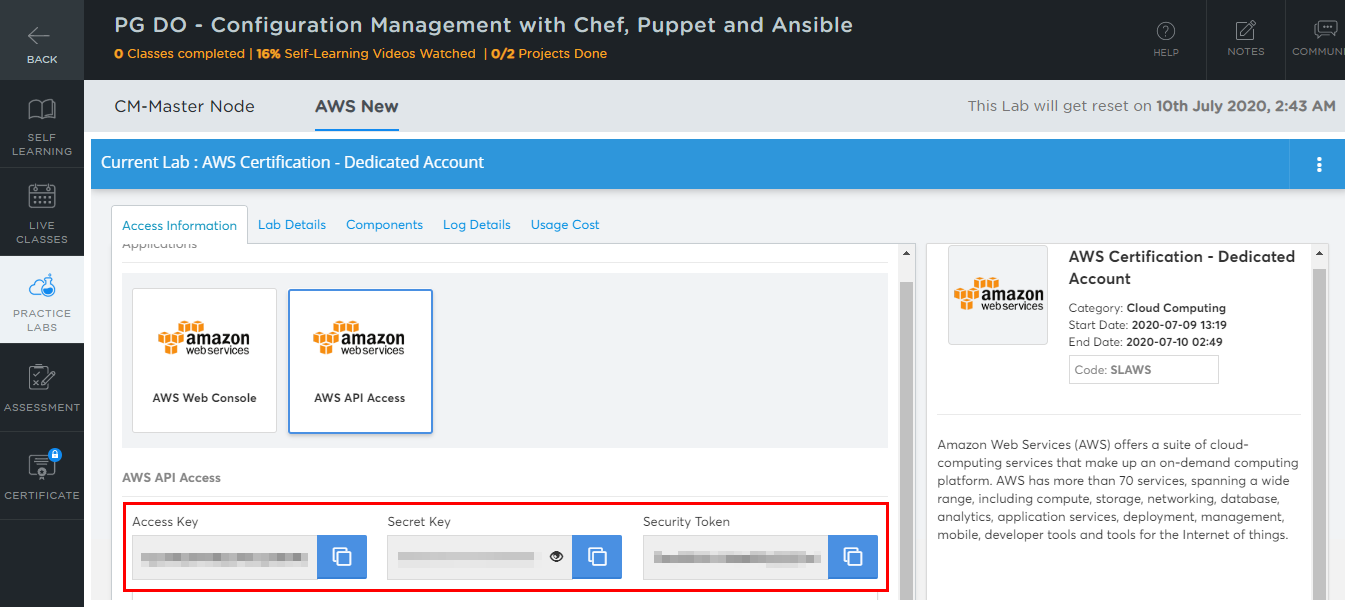
* Execute the following commands to set up your AWS credentials as environment variables:

**export AWS\_ACCESS\_KEY\_ID=<enter your access key>**

**export AWS\_SECRET\_ACCESS\_KEY= <enter your secret key>**

**export SECURITY\_TOKEN = <enter your security token>**

* Use the AWS access credentials provided in the AWS API Access tab in your LMS as shown in the screenshot below:



**Note:** AWS access credentials will change when the AWS Lab session expires after every four hours.

* Install Ansible based on the OS of the machine from which you plan to execute the script

### **Step 2:** Prepare Ansible playbook

* Ansible uses the folder structure shown below:

├── ansible.cfg

├── inventory

│ ├── base

│ ├── ec2.ini

│ ├── ec2.py

│ ├── static\_hosts

├── variables.yml

├── ec2\_prov\_playbook.yml

├── ec2\_term\_playbook.yml

* Enter the code given below in **ansible.cfg:**

**[defaults]**

**# update, as needed, for your scenario**

**host\_key\_checking=false**

**inventory = inventory/**

**# replace with your private key file or other auth method**

**private\_key\_file=${AWS\_EC2\_PEM\_KEYPATH}**

**control\_path=%(directory)s**

* Enter the code given below in **variables.yml:**

**STATE\_RES\_NAME: "${STATE\_RES\_NAME}"**

**ec2\_region: "${ec2\_region}"**

**ec2\_instance\_type: "${ec2\_tag\_Type}"**

**ec2\_image: "${ec2\_image}"**

**ec2\_keypair: "${ec2\_keypair}"**

**ec2\_volume\_size: ${ec2\_volume\_size}**

**ec2\_count: ${ec2\_count}**

**ec2\_security\_group\_id: "${security\_group\_id}"**

**ec2\_subnet\_id: "${public\_subnet\_id}"**

**ec2\_tag\_Type: "${ec2\_tag\_Type}"**

**ec2\_tag\_Role: "${ec2\_tag\_Role}"**

* Enter the code given below in **ec2\_prov\_playbook.yml:**

**---**

**### provision AWS EC2 instance**

**- hosts: localhost**

**connection: local**

**gather\_facts: false**

**user: root**

**pre\_tasks:**

**- include\_vars: variables.yml**

**tasks:**

**- name: Provision {{ ec2\_count }} instances with tag {{ ec2\_tag\_Role }}**

**local\_action:**

**module: ec2**

**key\_name: "{{ ec2\_keypair }}"**

**group\_id: "{{ ec2\_security\_group\_id }}"**

**instance\_type: "{{ ec2\_instance\_type }}"**

**image: "{{ ec2\_image }}"**

**vpc\_subnet\_id: "{{ ec2\_subnet\_id }}"**

**region: "{{ ec2\_region }}"**

**instance\_tags: '{"Type":"{{ec2\_instance\_type}}", "Role":"{{ec2\_tag\_Role}}"}'**

**assign\_public\_ip: yes**

**wait: true**

**exact\_count: "{{ ec2\_count }}"**

**count\_tag:**

**Role: "{{ ec2\_tag\_Role }}"**

**volumes:**

**- device\_name: /dev/xvda**

**volume\_type: gp2**

**volume\_size: "{{ ec2\_volume\_size }}"**

**delete\_on\_termination: true**

**register: ec2**

**- add\_host:**

**name: "{{ item.public\_ip }}"**

**groups: tag\_Type\_{{ec2\_tag\_Type}}**

**ec2\_region: "{{ ec2\_region }}"**

**ec2\_tag\_Type: "{{ ec2\_tag\_Type}}"**

**ec2\_tag\_Role: "{{ ec2\_tag\_Role }}"**

**ec2\_ip\_address: "{{ item.public\_ip }}"**

**with\_items: "{{ ec2.instances }}"**

**- name: Wait for the instances to boot by checking the ssh port**

**wait\_for: host={{item.public\_ip}} port=22 delay=15 timeout=300 state=started**

**with\_items: "{{ ec2.instances }}"**

**# update shippable resource state**

**- name: run cmd**

**shell: |**

**shipctl put\_resource\_state "{{ STATE\_RES\_NAME }}" "INST\_{{ item.ami\_launch\_index }}\_PUBLIC\_IP" "{{ item.public\_ip }}"**

**shipctl put\_resource\_state "{{ STATE\_RES\_NAME }}" "INST\_{{ item.ami\_launch\_index }}\_ID" "{{ item.id }}"**

**shipctl put\_resource\_state\_multi "{{ STATE\_RES\_NAME }}" "ec2\_tag\_Type={{ ec2\_tag\_Type }}" "ec2\_tag\_Role={{ ec2\_tag\_Role }}" "ec2\_region={{ ec2\_region }}"**

**with\_items: "{{ ec2.instances }}"**

**post\_tasks:**

**- name: refresh hosts inventory list**

**meta: refresh\_inventory**

* Enter the code given below in **ec2\_term\_playbook.yml:**

**---**

**### terminate AWS EC/2 instances**

**- hosts: localhost**

**connection: local**

**gather\_facts: false**

**user: root**

**pre\_tasks:**

**- include\_vars: variables.yml**

**tasks:**

**- name: Get EC2 instance IDs**

**run\_once: true**

**ec2\_remote\_facts:**

**filters:**

**"tag:Type": "{{ ec2\_tag\_Type }}"**

**"tag:Role": "{{ ec2\_tag\_Role }}"**

**region: "{{ ec2\_region }}"**

**register: instances**

**- name: display instances**

**run\_once: true**

**debug:**

**var: instances**

**- name: Remove registered instances**

**run\_once: true**

**ec2:**

**state: absent**

**wait: true**

**instance\_ids: "{{instances|json\_query('instances[\*].id')}}"**

**region: "{{ ec2\_region }}"**

**when: instances**

### **Step 3:** Running the playbook

* Execute the following command to run the Ansible playbook from the directory that contains the playbook:

**ansible-playbook –v ec2\_prov\_plaubook.yml**

* Verify on AWS if the EC2 machine was provisioned
* You can terminate the instance by running the command given below:

**ansible-playbook –v ec2\_term\_playbook.yml**