Provision and Configure EC2 Instances with Ansible

# Configure the AWS CLI on your Ubuntu Shell

* Ansible doesn’t play well with Windows, so we need to work in the WSL on our VMs
* Open a Ubuntu shell
* Login as the ansible user:

root@EC2AMAZ-8RI616P:~# login ansible

Password: <password is c0nygre – doesn’t display when you type>

* Now configure the AWS CLI as before using your access key or the one supplied by your instructor:

ansible@EC2AMAZ-8RI616P:~$ aws configure

AWS Access Key ID [None]: <AWS ACCESS KEY>

AWS Secret Access Key [None]: <AWS SECRET ACCESS KEY>

Default region name [None]: eu-west-2

Default output format [None]: json

# Create a Key Pair for SSH Sign In

* Now create a variable for your key pair, with your own unique name

$export KEY\_NAME=<put-your-key-name-here>

* Then create a key pair from the aws command line. This exercise uses the aws region eu-west-2, so you will need to create your key in that same region:

$ aws ec2 create-key-pair --region "eu-west-2" --key-name $KEY\_NAME | jq '.KeyMaterial' --raw-output >$KEY\_NAME.pem

* AWS requires that the key is not publicly viewable, run the following command

$chmod 400 $KEY\_NAME.pem

# Launch an EC2 Instance

* Ansible can not only configure resource, it can provision them too! There are Ansible modules available for most of the resources available on AWS.
* In this case we will use Ansible to create a security group and provision a single EC2 linux instance that is accessible using SSH.
* The Ansible playbook file is available using this command from your Ubuntu command line:

curl -LJO https://raw.githubusercontent.com/dascog/terraform-3day/main/labs/student/2.1-ansible-configure-ec2/launch\_ec2.yml

* If you run $cat launch\_ec2.yml the text should be as follows:

---

- name: Create AWS EC2 Resources

  hosts: localhost

  connection: local

  gather\_facts: false

  tasks:

    - name: Create security group

      ec2\_group:

        name: ansible\_ssh\_sg

        description: "Security group for SSH connection to EC2 instances"

        region: eu-west-2

        rules:

          - proto: tcp

            rule\_desc: "ssh"

            from\_port: 22

            to\_port: 22

            cidr\_ip: 0.0.0.0/0

        rules\_egress:

          - proto: all

            rule\_desc: "all"

            cidr\_ip: 0.0.0.0/0

      register: basic\_firewall

    - name: Create ec2 instance

      ec2:

        instance\_type: t2.micro

        image: ami-060c4f2d72966500a

        count: 1

        group\_id: "{{basic\_firewall.group\_id}}"

        key\_name: "{{lookup('env', 'KEY\_NAME')}}"

        region: eu-west-2

        wait: yes

* When all this is in place, we can run the ansible playbook and create the necessary resources:

$ansible-playbook -v launch\_ec2.yml

Using /etc/ansible/ansible.cfg as config file

PLAY [Create AWS EC2 Resources] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [Create security group] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [localhost] => {"ansible\_facts": {"discovered\_interpreter\_python": "/usr/bin/python3"}, "changed": false, "description": "Security group for SSH connection to EC2 instances", "group\_id": "sg-09e6cfb3c46e3a18e", "group\_name": "ansible\_ssh\_sg", "ip\_permissions": [{"from\_port": 22, "ip\_protocol": "tcp", "ip\_ranges": [{"cidr\_ip": "0.0.0.0/0", "description": "ssh"}], "ipv6\_ranges": [], "prefix\_list\_ids": [], "to\_port": 22, "user\_id\_group\_pairs": []}], "ip\_permissions\_egress": [{"ip\_protocol": "-1", "ip\_ranges": [{"cidr\_ip": "0.0.0.0/0", "description": "all"}], "ipv6\_ranges": [], "prefix\_list\_ids": [], "user\_id\_group\_pairs": []}], "owner\_id": "369574464406", "tags": {}, "vpc\_id": "vpc-4bd1e123"}

TASK [Create ec2 instance] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [localhost] => {"changed": true, "instance\_ids": ["i-08ce975c49395a9cb"], "instances": [{"ami\_launch\_index": "0", "architecture": "x86\_64", "block\_device\_mapping": {"/dev/xvda": {"delete\_on\_termination": true, "status": "attached", "volume\_id": "vol-04358cf537947694d"}}, "dns\_name": "ec2-13-40-180-209.eu-west-2.compute.amazonaws.com", "ebs\_optimized": false, "groups": {"sg-09e6cfb3c46e3a18e": "ansible\_ssh\_sg"}, "hypervisor": "xen", "id": "i-08ce975c49395a9cb", "image\_id": "ami-060c4f2d72966500a", "instance\_type": "t2.micro", "kernel": null, "key\_name": "LinuxAnsible", "launch\_time": "2022-03-08T13:55:32.000Z", "placement": "eu-west-2b", "private\_dns\_name": "ip-172-31-37-20.eu-west-2.compute.internal", "private\_ip": "172.31.37.20", "public\_dns\_name": "ec2-13-40-180-209.eu-west-2.compute.amazonaws.com", "public\_ip": "13.40.180.209", "ramdisk": null, "region": "eu-west-2", "root\_device\_name": "/dev/xvda", "root\_device\_type": "ebs", "state": "running", "state\_code": 16, "tags": {}, "tenancy": "default", "virtualization\_type": "hvm"}], "tagged\_instances": []}

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

localhost : ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

* The output should look like something like the above.
* In your output look for the public\_dns\_name entry (highlighted in red above). This can be used to SSH into your new EC2 instance (you can also find the DNS name and connection instructions in the EC2 Management Console)
* In your Ubuntu command line type the following to access your new EC2 instance:

$ ssh -i "$KEY\_NAME.pem" ec2-user@<YOUR PUBLIC\_DNS\_NAME>

The authenticity of host 'ec2-13-40-180-209.eu-west-2.compute.amazonaws.com (13.40.180.209)' can't be established.

ECDSA key fingerprint is SHA256:ROV1cALFF+aSeROlYe7mhPnabQ8Zvp3nva9hf1hbS54.

Are you sure you want to continue connecting (yes/no/[fingerprint])? yes

Warning: Permanently added 'ec2-13-40-180-209.eu-west-2.compute.amazonaws.com,13.40.180.209' (ECDSA) to the list of known hosts.

\_\_| \_\_|\_ )

\_| ( / Amazon Linux 2 AMI

\_\_\_|\\_\_\_|\_\_\_|

https://aws.amazon.com/amazon-linux-2/

No packages needed for security; 1 packages available

Run "sudo yum update" to apply all updates.

* Congratulations! You have provisioned an EC2 instance using Ansible! Leave this terminal open, we will need it later.

# Configuring a Server with Ansible

* The server we have provisioned is quite limited and only has SSH (command line access), however we can imagine there being a long line of fully fledged webservers that we now need to configure – this is where Ansible comes into its own!
* On a new Ubuntu terminal, login as ansible, then create a file called ansible\_hosts (using nano or vim) with the following content (note that you have to put your own key name – the same one as in your KEY\_NAME environment variable – in place of <YOUR-KEY-NAME>):

[aws]

<YOUR EC2 INSTANCE PUBLIC\_DNS\_NAME: e.g. ec2-13-40-180-209.eu-west-2.compute.amazonaws.com>

[aws:vars]

ansible\_user=ec2-user

ansible\_ssh\_private\_key\_file=/home/ansible/<YOUR-KEY-NAME>.pem

* Make sure you copy your own public DNS name in the top section, and verify that the private key path is correct on your host machine.
* Next create a simple playbook to test the connection is working. Let’s call is test-aws.yml

---

  - hosts: aws

    tasks:

      - debug: msg="Ansible is working!"

* This playbook can be run with the following command, and you should get output like the output listed:

$ ansible-playbook -i ansible\_hosts test-aws.yml

PLAY [aws] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [Gathering Facts] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[WARNING]: Platform linux on host ec2-13-40-180-209.eu-

west-2.compute.amazonaws.com is using the discovered Python interpreter at

/usr/bin/python, but future installation of another Python interpreter could

change this. See https://docs.ansible.com/ansible/2.9/reference\_appendices/interpreter\_discovery.html for more information.

ok: [ec2-13-40-180-209.eu-west-2.compute.amazonaws.com]

TASK [debug] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [ec2-13-40-180-209.eu-west-2.compute.amazonaws.com] => {

"msg": "Ansible is working!"

}

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ec2-13-40-180-209.eu-west-2.compute.amazonaws.com : ok=2 changed=0 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

* The ok=2 at the end tells us that all is well, and we can see the debug message successfully printing in the TASK section – we are good to go!
* Now we’ll add a second task to the test-aws.yml file to install some software. Add the following task to test-aws.yml to install the critical linux function cowsay:

- yum: name=cowsay state=present

* However to run the yum command you need to have elevated privileges. To do this in an ansible playbook you need to use the command become: yes. This tells ansible to elevate your user privileges if possible. After you have done this your playbook should look like:

---

  - hosts: aws

    become: yes

    tasks:

      - debug: msg="Ansible is working!"

      - yum: name=cowsay state=present

* Now run the playbook as you did above.
* In your EC2 terminal, run the command cowsay mooo!

\_\_\_\_\_\_

< mooo! >

------

\ ^\_\_^

\ (oo)\\_\_\_\_\_\_\_

(\_\_)\ )\/\

||----w |

|| ||

* Now edit your playbook file in the host so the state is absent rather than present and run the playbook again.
* In your EC2 instance try to run the cowsay mooo command again. The software has been uninstalled.