CS 7001-03: Report for AWS Lab 2 - AWS Resource Discovery and Instance Setup

Chanmann Lim
cl9p8@mail.mail.missouri.edu

March 03, 2015

Install `awscli` tool via easy_install `pip` on Mac OS.

1. Create an AWS key pair using `aws ec2 create-key-pair` command with `--key-name` option set to 'cloud-key':

```
# aws ec2 create-key-pair --key-name cloud-key
```

Delete a key pair using `aws ec2 delete-key-pair` command with `--key-name` option set to 'cloud-key':

```
# aws ec2 delete-key-pair --key-name cloud-key
```

2. Create a security group in AWS.

Use `aws ec2 create-security-group` command with options

`--group-name` **:** set security group name.

`--description` **:** set security group description.

and adding inbound traffic rule to security group via `aws ec2 authorize-security-group-ingress` command with options

`--group-name` **:** security group name.

`--protocol` **:** IP protocol eg. tcp, udp or icmp.

`--port` **:** tcp or tcp port range.

`-cidr` **:** IP range.

```
# aws ec2 create-security-group  \
      --group-name cloud-group   \
      --description "Open ports"
# aws ec2 authorize-security-group-ingress \
      --group-name cloud-group            \
      --protocol tcp                      \
      --port 22                           \
      --cidr 0.0.0.0/0
# aws ec2 authorize-security-group-ingress \
      --group-name cloud-group            \
      --protocol tcp                      \
      --port 80                           \
      --cidr 0.0.0.0/0
# aws ec2 authorize-security-group-ingress \
      --group-name cloud-group            \
      --protocol tcp                      \
      --port 443                           \
      --cidr 0.0.0.0/0
```

Delete security group via `aws ec2 delete-security-group` command with `--group-name` option set to 'cloud-group'.

```
# aws ec2 delete-security-group --group-name cloud-group
```

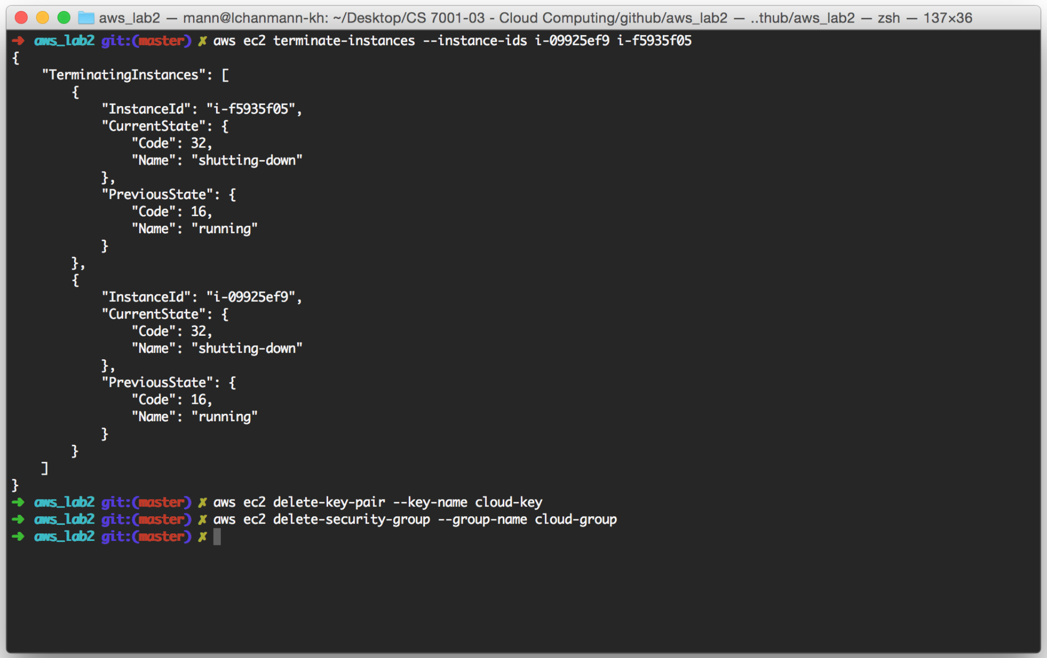
3. Execute command:

```
# aws ec2 run-instances --image-id ami-caf9a6a2 \
--instance-type t1.micro \
--count 2 \
--key-name cloud-key \
--security-groups cloud-group \
--region us-east-1
```

will result in launching two instances (servers) in North Virginia(us-east-1) Amazon cloud using private Amazon Machine Image (AMI) – "ami-caf9a6a2" as the template, configuring security group to "cloud-group" (SSH, HTTP and HTTPS are opened) and embedding "cloud-key" key pair for ssh login to both instances without password.

Terminate the instances, delete "cloud-key" key pair and "cloud-group" security group with following commands:

```
# aws ec2 terminate-instances --instance-ids i-09925ef9 i-f5935f05
# aws ec2 delete-key-pair --key-name cloud-key
# aws ec2 delete-security-group --group-name cloud-group
```

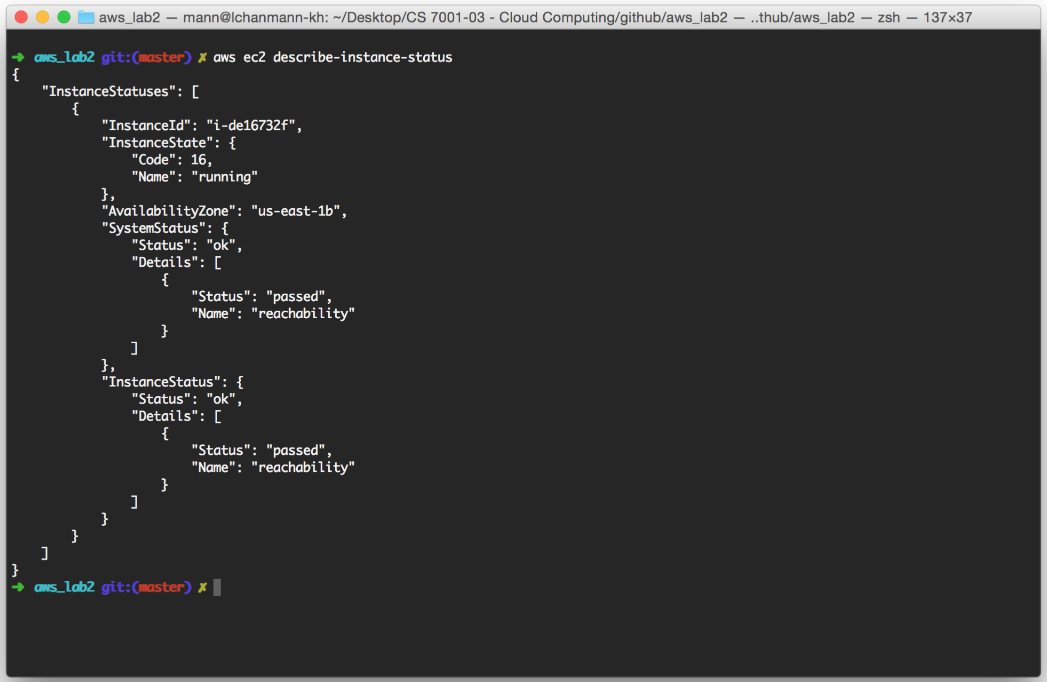


```
aws_lab2 — mann@lchanmann-kh: ~/Desktop/CS 7001-03 - Cloud Computing/github/aws_lab2 — ..thub/aws_lab2 — zsh — 137×36
➜ aws_lab2 git:(master) ✗ aws ec2 terminate-instances --instance-ids i-09925ef9 i-f5935f05
{
    "TerminatingInstances": [
        {
            "InstanceId": "i-f5935f05",
            "CurrentState": {
                "Code": 32,
                "Name": "shutting-down"
            },
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        },
        {
            "InstanceId": "i-09925ef9",
            "CurrentState": {
                "Code": 32,
                "Name": "shutting-down"
            },
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}
➜ aws_lab2 git:(master) ✗ aws ec2 delete-key-pair --key-name cloud-key
➜ aws_lab2 git:(master) ✗ aws ec2 delete-security-group --group-name cloud-group
➜ aws_lab2 git:(master) ✗
```

Figure 1: Terminate instances, delete key pair and security group.

4. Get status information of all instances using aws-cli commands.

```
# aws ec2 describe-instance-status
```


aws_lab2 — mann@lchanmann-kh: ~/Desktop/CS 7001-03 - Cloud Computing/github/aws_lab2 — ..thub/aws_lab2 — zsh — 137×37

```
➜ aws_lab2 git:(master) ✗ aws ec2 describe-instance-status
{
    "InstanceStatuses": [
        {
            "InstanceId": "i-de16732f",
            "InstanceState": {
                "Code": 16,
                "Name": "running"
            },
            "AvailabilityZone": "us-east-1b",
            "SystemStatus": {
                "Status": "ok",
                "Details": [
                    {
                        "Status": "passed",
                        "Name": "reachability"
                    }
                ]
            },
            "InstanceStatus": {
                "Status": "ok",
                "Details": [
                    {
                        "Status": "passed",
                        "Name": "reachability"
                    }
                ]
            }
        }
    ]
}
➜ aws_lab2 git:(master) ✗
```

Figure 2: AWS instances status

5. Create snapshot command: `aws ec2 create-snapshot` with options

`--volume-id` **:** set EBS volume to be snapshot.

`--description` **:** set snapshot description.

```
# aws ec2 create-snapshot      \
      --volume-id vol-54c4644f \
      --description "Backup"
```

Delete snapshot command: `aws ec2 delete-snapshot` with `--snapshot-id` option.

```
# aws ec2 delete-snapshot --snapshot-id snap-51cf8cd0
```

6. Add a new EBS volume with `aws ec2 create-volume` with options:

`--size` **:** set volume size (in GB).

`--availability-zone` **:** set availability zone of the volume.

```
#aws ec2 create-volume --size 3 --availability-zone us-east-1b
```

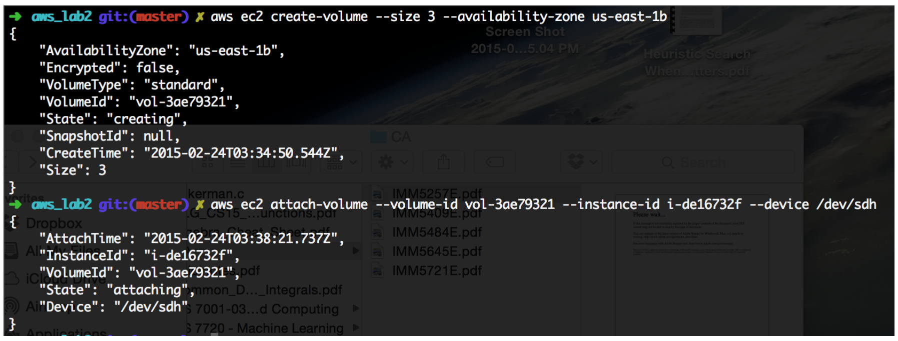
Attach the volume to the running instance (i-de16732f) via `aws ec2 attach-volume` with options:

`--volume-id` **:** set volume id to attach.

`--instance-id` **:** set instance id to be attached to.

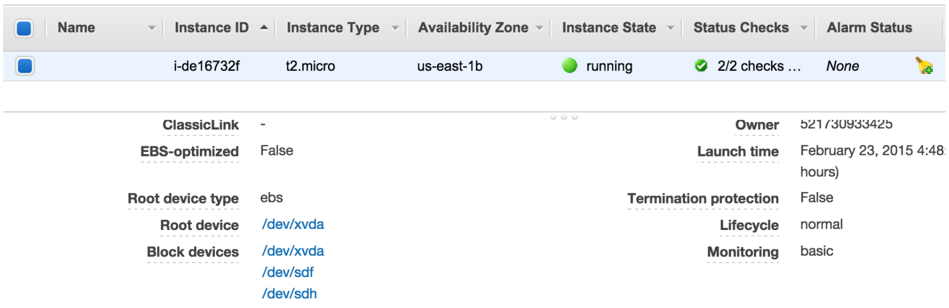
`--device` **:** set device name with which the instance will use to interact.

```
#aws ec2 attach-volume          \
     --volume-id vol-3ae79321 \
     --instance-id i-de16732f \
     --device /dev/sdh
```



```
➜ aws_lab2 git:(master) ✗ aws ec2 create-volume --size 3 --availability-zone us-east-1b
{
    "AvailabilityZone": "us-east-1b",
    "Encrypted": false,
    "VolumeType": "standard",
    "VolumeId": "vol-3ae79321",
    "State": "creating",
    "SnapshotId": null,
    "CreateTime": "2015-02-24T03:34:50.544Z",
    "Size": 3
}
➜ aws_lab2 git:(master) ✗ aws ec2 attach-volume --volume-id vol-3ae79321 --instance-id i-de16732f --device /dev/sdh
{
    "AttachTime": "2015-02-24T03:38:21.737Z",
    "InstanceId": "i-de16732f",
    "VolumeId": "vol-3ae79321",
    "State": "attaching",
    "Device": "/dev/sdh"
}
```

Figure 3: Create and attach volume using aws-cli



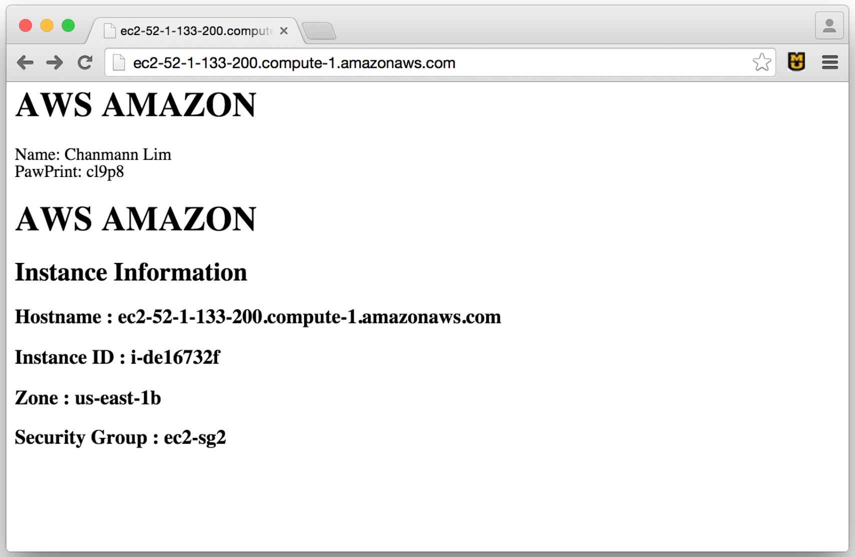
	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status
		i-de16732f	t2.micro	us-east-1b	● running	✓ 2/2 checks ...	None

ClassicLink	-	Owner	521730933425
EBS-optimized	False	Launch time	February 23, 2015 4:48 hours)
Root device type	ebs	Termination protection	False
Root device	/dev/xvda	Lifecycle	normal
Block devices	/dev/xvda /dev/sdf /dev/sdh	Monitoring	basic

Figure 4: The new volume is attached to /dev/sdh

7. Provide a screenshot taken in Step 3.4.2

`http://ec2-52-1-133-200.compute-1.amazonaws.com/`



ec2-52-1-133-200.compute-1.amazonaws.com

AWS AMAZON

Name: Chanmann Lim
PawPrint: cl9p8

AWS AMAZON

Instance Information

Hostname : ec2-52-1-133-200.compute-1.amazonaws.com

Instance ID : i-de16732f

Zone : us-east-1b

Security Group : ec2-sg2

Figure 5: AWS web server

8. The six best practices described by Amazon AWS:

Design for failure and nothing will fail :
By assuming everything can fail (hardware failure, software failure, outage and failure due to natural disaster), application architects should embrace automated recovery for data, OS configurations, and in-progress jobs processing.

Decouple your components :
The cloud encourages Service-Oriented-Architecture for decoupling the components in the system to support concurrency, high availability.

Implement elasticity :
Automated scaling by automating deployment process with script to help reduce error and facilitate an efficient and scalable update process.

Think parallel :
The cloud architects should parallelize their applications to make them thread-safe, multi-threading and request parallelization to achieve maximum performance and throughput.

Keep dynamic data closer to the compute and static data closer to the end-user :
To reduce the processing and internet latency by moving the data and data-consuming applications into the cloud and moving static content such as images, videos, audios, pdfs, etc., into content delivery network which get cached at edge locations closer to the users to lower the access latency.

Security Best Practices:
Developers should implement security best practices and standard methods such as 1) configuring SSL or setting up Virtual Private Cloud for sensitive information exchanges, 2) using file encryption tools to secure the storage, 3) using AWS access key and access key rotation when the key get compromised or X.509 certificates for authentication and multi-factor authentication with Amazon account information, 3) using AWS Identity and Access Management (IAM) to manage user permission on accessing AWS services, 4) configuring security group for incoming network traffic, using firewall softwares (*netfilter* and *iptables*), regularly updating patches and run security checks.