Lecture 9

AWS

Chapter 14

Introduction to Amazon Web Services

Introduction to the CloudStack Cloud

Amazon Web Services

Amazon Web Services is the cloud 900 lb gorilla

- According to the Synergy Research Group (2016):

  "For full-year 2015 AWS share of the worldwide market was 31%, followed by Microsoft (9%), IBM (7%), Google (4%) and Salesforce (4%)."

Amazon Web Services

- Synergy divides the cloud marketplace into two tiers.

They call their first tier “the big four”:

- Amazon
- Microsoft
- IBM
- Google

Amazon Web Services

- The Synergy second tier includes (among others):
  - Salesforce
  - Rackspace
  - Oracle
  - NTT
  - Fujitsu
  - Alibaba
  - HPE

Open Source Cloud Software

- OpenStack is the most widely used open source cloud software
- As of 2016, CloudStack is the second most widely used open source cloud
  - However, Citrix was a big backer of CloudStack
  - In January 2016 Citrix sold its commercial cloud products (based on CloudStack) to Accelerite (owned by Persistent Systems)

Amazon Web Services

AWS breaks its services into the following categories, each with sub-products:

- Compute
- Storage and Content Delivery
- Database
- Networking
- Analytics
- Enterprise Applications
- Internet of Things
- Mobile Services
- Developer Tools
- Management Tools
- Security and Identity
- Application Services
- Software (maintenance)

Some of these are not public offerings but rather support other areas of AWS

Amazon Web Services

Amazon Elastic Compute Cloud (EC2):

- IaaS offering
- Users manage own VMs and deployments

Amazon Elastic Load Balancing (ELB)

- IaaS offering
- automatically distributes incoming application traffic across multiple Amazon EC2 instances

Amazon Elastic Beanstalk

- PaaS offering
- Uses Amazon Elastic Load Balancing internally

Amazon Web Services

Amazon Virtual Private Cloud (VPC)

- allows a user to create a logically isolated section of the AWS cloud
- achieved through use of a VLAN

Amazon Elastic Block Store (EBS)

- allows a volume (block storage), typically containing a file system, to be attached to a virtual machine instance
- automatically backed up to provide fault tolerance

Amazon Simple Storage Service (S3)

- provides object storage (can be a file), with the objects accessible through a web API
- here is a RESTful API and a non-RESTful (SOAP-based) API,
  - however, the SOAP-based API has been deprecated, see Amazon Simple Storage Service (2016)

Amazon Web Services

- free-tier accounts exist
- Watch out for your credentials!
- Don’t embed in your software and upload to github

Amazon Web Services-Steps to Get Started

- Log in using your AWS account
- Select the EC2 Dashboard
- Note the Launch Instance button, we'll use it later
- Select region (see upper right hand corner)
- Scroll down left hand side menu until reach Security Group
- Click on Security Group
- Select Default Security group
  - Select Actions
  - Select Edit Inbound Rules
  - Click on Add Rule
  - Add SSh
  - Click on Add Rule
  - Add ALL ICMP

Amazon Web Services-Steps to Get Started (cont'd)

- Scroll down until get to Key Pairs on left hand side menu
- Select Key Pairs
  - Select Create Key Pair
  - Name your keypair "mine"
  - Download mine.pem file
    - This is your private key, AWS stores your public key

Next we will look at how to launch instances

Amazon Web Services-Steps to Get Started (cont'd)

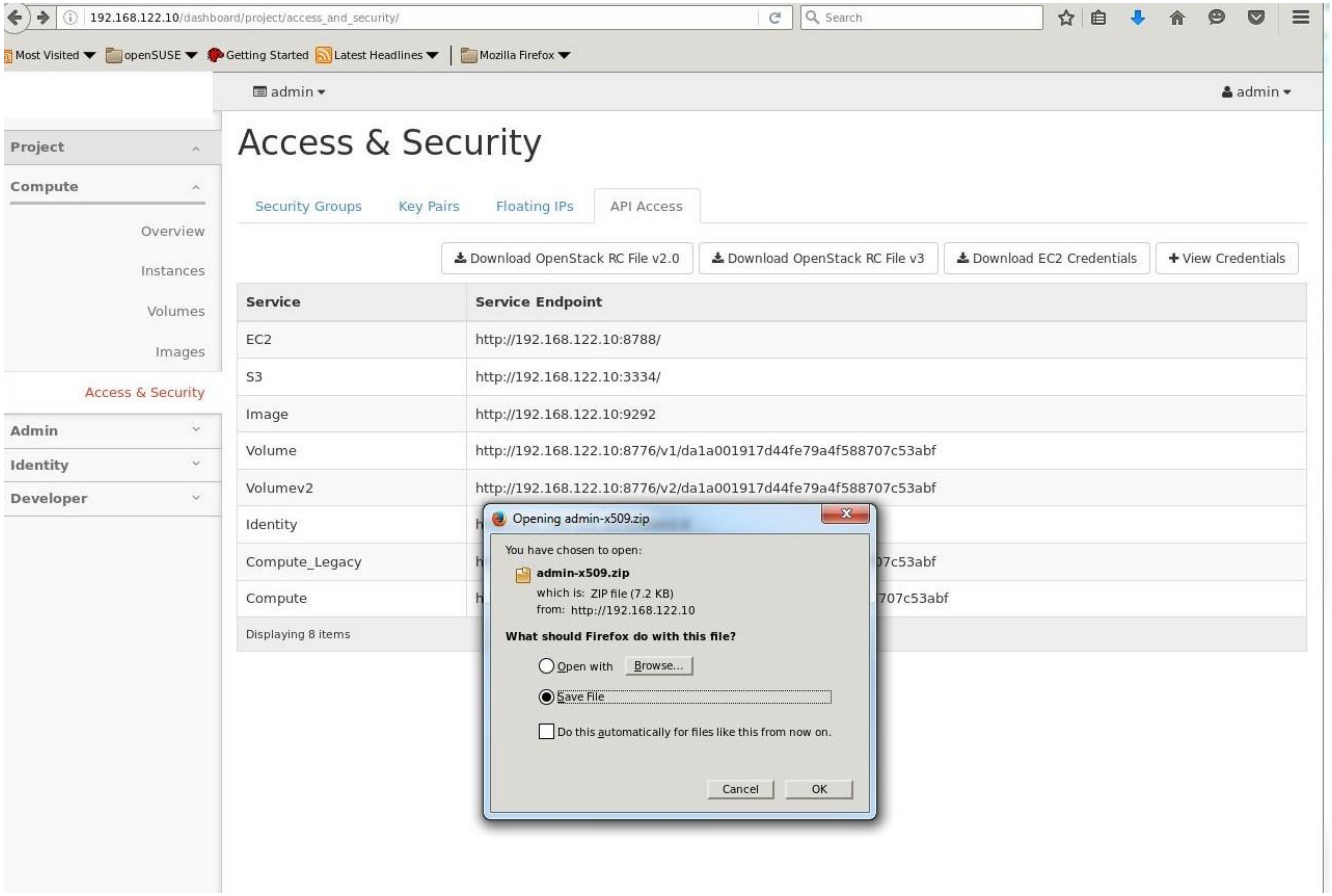
- Go back up to Launch Instance and click on it
  - See a list of Operating System images (some free-tier, some cost $)
  - Select the top free-tier image, something like:
    - Amazon Linux AMI 2016.03.1 (HVM), SSD Volume Type - ami-f5f41398
  - See screen to select instance type
    - Select t2.micro
  - Click Review and Launch (lower right hand corner of screen)
    - See a confirmation screen showing details about the instance
    - If you are happy with the settings click Launch
      - It will ask you to select a keypair to use
        - Select the "mine" keypair
    - Then click Launch Instances and you will have a running instance!

OpenStack EC2 Interface

- Assuming you’re using a devstack install
- Depending on which version of devstack you use, you may have to enable the nova EC2 API in the install by using the EC2 API plugin
- If you do this, your url for the EC2 API will be:
  - export EC2_URL=http://146.229.233.30:8788/services/Cloud
- whereas the url for the the original EC2 API that is part of nova is:
  - export EC2_URL=http://146.229.233.30:8773/services/Cloud

OpenStack EC2 Interface

- Go to Access and Security Tab on OpenStack Dashboard
- Click API Access
- Click Download EC2 Credentials



OpenStack EC2 Interface

- (assuming you’re using a devstack install here)
- source ec2rc.sh
  - This will set environment variables appropriately
  - Includes environment variables for your access key and secret key

OpenStack EC2 Interface—Python with boto authentication

```
def authfunc(boto):
    import os
    myendpoint='http://localhost:8788/services/Cloud'
    my_access_key=os.getenv("EC2_ACCESS_KEY")
    my_secret_key=os.getenv("EC2_SECRET_KEY")

    ec2 = boto.connect_ec2_endpoint(myendpoint,
                                    aws_access_key_id=my_access_key,
                                    aws_secret_access_key=my_secret_key)
    return ec2
```

OpenStack EC2 Interface—Python with boto start instance

```
python
import boto

import auth_proj
myconn=auth_proj.authfunc(boto)

myconn.get_all_images()
myimage=myconn.get_image('ami-00000001')
myimage.name

myconn.run_instances(key_name='mykey',image_id='ami-00000001',
instance_type='m1.nano')
```

OpenStack EC2 Interface—Python with boto3

- Boto3 has two main ways of looking at things
  - Resources
  - Clients

- A resource is a “higher level abstraction” than a client
- Clients provide a low-level interface whose methods map closely with service APIs

- Think of a service such as EC2 or S3 as a resource.
- You can do many higher level application type things by accessing the resource directly.
- Then if you need to you can create a client to do low level things

OpenStack EC2 Interface—Access OpenStack EC2 using python with boto3 resource

```
import boto3

ec2=boto3.resource('ec2',region_name='RegionOne',endpoint_url='http"//192.
168.122.10:8788')

instances=ec2.instances.all()

for w in instances:
   print w

images=ec2.images.all()
for w in images:
   print w
```

OpenStack EC2 Interface—Access OpenStack EC2 using python with boto3 resource (cont'd)

```
image=ec2.Image('ami-b7378f53')
image.name

ec2.create_instances(ImageId='ami-b7378f53', InstanceType='m1.nano',
Keyname='mykey', MinCount=1, MaxCount=1)

instances=ec2.instances.all()
for w in instances:
   print w
```

OpenStack EC2 Interface—Access OpenStack EC2 using python with boto3 client (cont'd)

```
import boto3
ec2=boto3.client('ec2',region_name='RegionOne',endpoint_url='http://192.168.122.10:8788')
ec2.describe_regions()
```

AWS EC2 API on AWS Cloud

- In main AWS home screen (log in screen), go to upper right hand corner
- Select your name, and you will get a drop down menu
  - Select Security Credentials
    - Click on Select New Access Key
    - Click Download Key file (it will be named rootkey.csv)
    - Rootkey.csv will have contents:
      - AWSAccessKeyId=*theaccesskeyitself*
      - AWSSecretKey=*thesecretkeyitself*
- copy/paste access key itself into a file named aa
- Copy/paste the secret key into a file named aa
- Then type:
  - export AWS_ACCESS_KEY_ID=`cat aa`
  - export AWS_SECRET_ACCESS_KEY=`cat bb`

AWS EC2 API on AWS Cloud

- Alternately, after you download your credentials, you can create a directory called “.aws”
- Make a text file called credentials, which contains:

  [default]

  aws_access_key_id=…put your very own access key here…

  aws_secret_access_key=…put your very own secret key here…

- NOTE: to make boto3 work, I had to do the credentials this way

AWS EC2 API on AWS Cloud—Access using python with boto

```
import boto
import boto.ec2

ec2=boto.ec2.connect_to_region('us-east-1')

the_image_id='ami-f5f41398'

my_image=ec2.get_image(the_image_id)

print "the image really is ",my_image_name

the_instance_type='t2.micro'

ec2.run_instances(image_id=the_image_id,instance_type=the_instance_type,
key_name='mine', security_groups=['launch-wizard-3'])
```

AWS EC2 API on AWS Cloud—Access using python with boto3 resource

```
import boto3

ec2=boto3.resource('ec2',region_name='us-east-1')

instances=ec2.instances.all()

for w in instances:
    print w

ec2.create_instances(ImageId='ami-f5f41398', instance_type='t2.micro',
KeyName='mine',
MinCount=1,MaxCount=1)
```

AWS EC2 API on AWS Cloud—Access using python with boto3 client

```
import boto3
ec2=boto3.client('ec2',region_name='us-east-1')
ec2.describe_regions()
```

AWS S3 API on AWS Cloud

- In S3, a “bucket” is used to store objects
- Inside a bucket, you have a key and then the object that is associated with that key
- You can choose a region where the bucket will be located

AWS S3 API on AWS Cloud—Access using python with boto3 resource

```
import boto3

s3=boto3.resource('s3',region_name='us-east-1')

s3.create_bucket(Bucket='longnastyname')

s3.Bucket('longnastyname').upload_file('testy.txt','thetestyfile')

bucket=s3.Bucket('longnastyname')

for theobject in bucket.objects.all():
   print theobject

s3.Bucket('longnastyname').download_file('thetestyfile','kk.txt')

for key in bucket.objects.all():
   key.delete()

bucket.delete()
```

AWS S3 API on AWS Cloud—Access using python with boto3 resource (cont'd)

- The bucket has a "longnastyname"
- It has to be a name that is unique across all buckets in AWS
- Amazon recommends prefixing your bucket name with the name of your company to avoid pre-existing bucket names
- As of now, however, "longnastyname" has been taken!