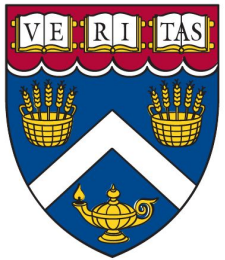


HARVARD
Extension School

Week 6

Amazon AWS SDK for Python

This week..

- **Week 06:** Amazon AWS SDK for Python

In this lecture, we will learn how to use Python libraries (mainly Boto3) to manage AWS resources. The same environment created in week 04 and week 05 will be created using AWS CLI and SDK.

- AWS CLI
- Python Boto3
- Building AWS resources using AWS SDK for Python

AWS CLI

- Control AWS services and resources through script
- Automation is easier with the CLI

AWS CLI - installation and configuration

1. Installation:

```
$> pip install awscli
```

https://docs.aws.amazon.com/cli/latest/userguide/installing.html

2. Configuration:

```
$> aws configure
```

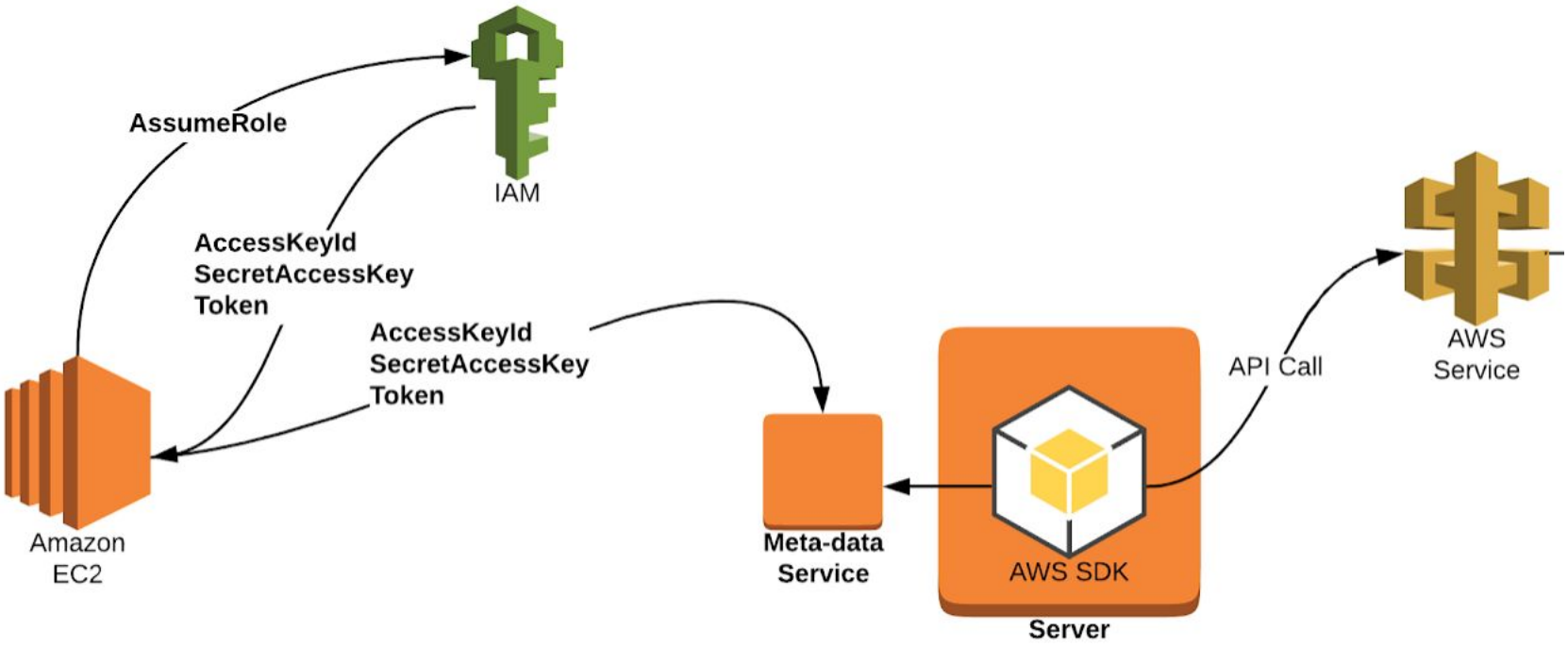
https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-started.html

AWS CLI - Authorization

- IAM User Credentials:
  - Access Key
  - Secret Key
  - To create Admin user : AWS Console -> AMI -> Users -> Add user -> User name “Administrators” -> check both Access Type boxes. -> Next -> Create User

- Roles:
  - EC2 role
  - Lambda role
  - To create Admin Role : AWS Console -> AMI -> Roles -> Create Role -> EC2 -> AdministratorAccess -> Role Name “AdminFullAccess”

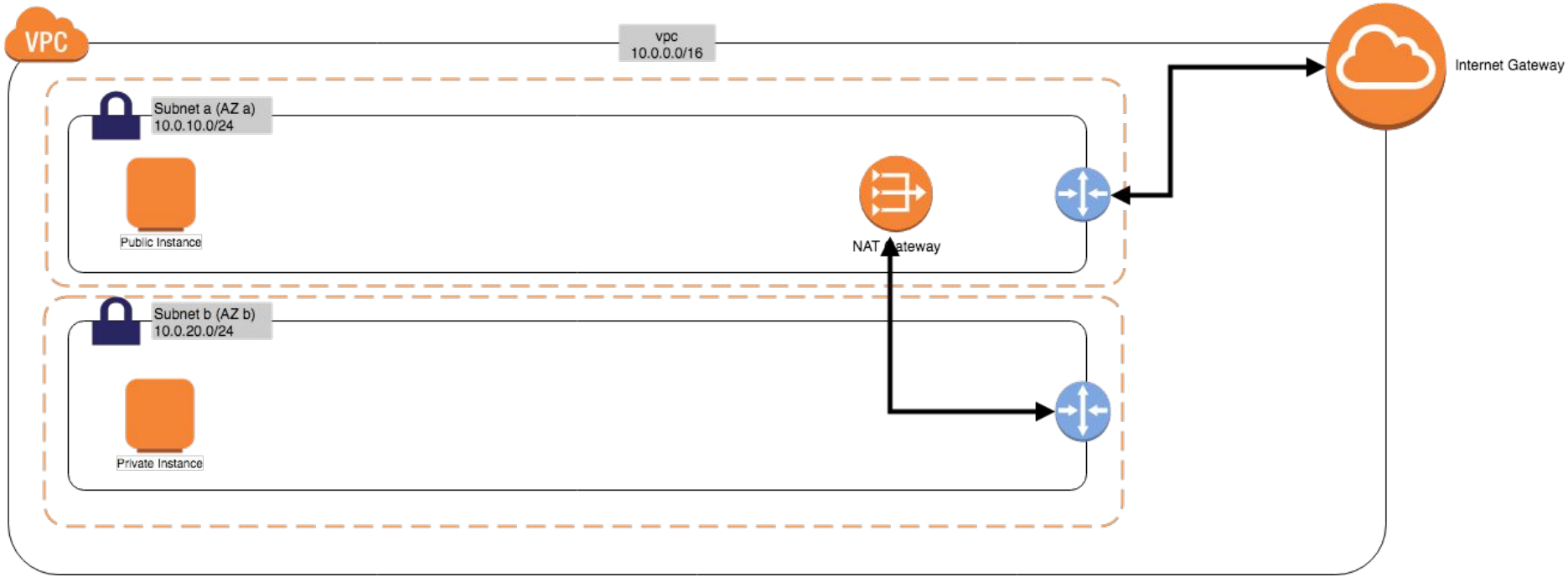
Role and Security Token Service



Review of mostly used AWS CLI commands

```
apt -get update -y
apt-get install python3-pip
pip3 install awscli
aws configure
aws help
aws ec2 help
aws ec2 describe-instances
```

Environment we have so far..



Output of shell command to a variable

```
$ cat > filesfolders.sh << EOF
#!/bin/bash
mkdir test
mkdir test/t1
mkdir test/t2
echo "this is t1 dir" > test/t1/README.t1
echo "this is t2 dir" > test/t2/README.t2
echo "This is main folder dir" > test/MAIN.txt
EOF

$ chmod +x filesfolders.sh
$ ./filesfolders.sh
$ # OR source filesfolders.sh

$ cd test
$ ls *
$ lsv=`ls *`
$ echo $lsv

$ lsv=$(ls *)`
$ echo $lsv
```

```
$ dva=$(date)
$ msg=msg="today is ..  "
$ echo "$msg $dva"

$ free
              total        used        free      shared  buff/cache   available
Mem:         982496       56816      748464         332      177216      777176
Swap:             0           0           0

$ free -m
              total        used        free      shared  buff/cache   available
Mem:            959          55         730           0         173         759
Swap:             0           0           0

$ free -m | grep Mem
Mem:            959          55         730           0         173         758

$ free -m | grep Mem | awk '{print $4}'
730

memM=$(free -m | grep Mem | awk '{print $4}')
echo "I have $memM megabytes of unused  memory "
I have 730 megabytes of unused  memory
```

```
$ apt-get update -y

$ apt-get install python3-pip

$ pip3 install awscli

$ aws configure  # What user ? root? Or any user in the system ?

$ aws ec2 describe-instances

$ aws ec2 describe-volumes

$ aws ec2 describe-vpcs  --query 'Vpcs[*].{ID:VpcId}' --output text

$ aws help

$ aws <service > help
```

```
$ aws ec2 create-vpc --cidr-block 10.0.0.0/16

$ aws ec2 create-subnet --cidr-block 10.0.0.0/17 --vpc-id vpc-03938376907823078 --availability-zone
us-east-1a

$ aws ec2 create-subnet --cidr-block 10.0.128.0/17 --vpc-id vpc-03938376907823078 --availability-zone
us-east-1c

$ aws ec2 create-internet-gateway

$ aws ec2 attach-internet-gateway --vpc-id vpc-03938376907823078 --internet-gateway-id igw-047ae6de9733b792d

$ aws ec2  create-route-table --vpc-id vpc-03938376907823078

$ aws ec2 create-route --route-table-id rtb-0747899edc9f679c9 --destination-cidr-block 0.0.0.0/0 --gateway-id
igw-047ae6de9733b792d

$ aws ec2 describe-subnets | grep us-east-1a

$ aws ec2 describe-route-tables | grep 10.0.0

$ aws ec2 associate-route-table --route-table-id rtb-0747899edc9f679c9 --subnet-id subnet-0d5ff97b62f73ae31
```

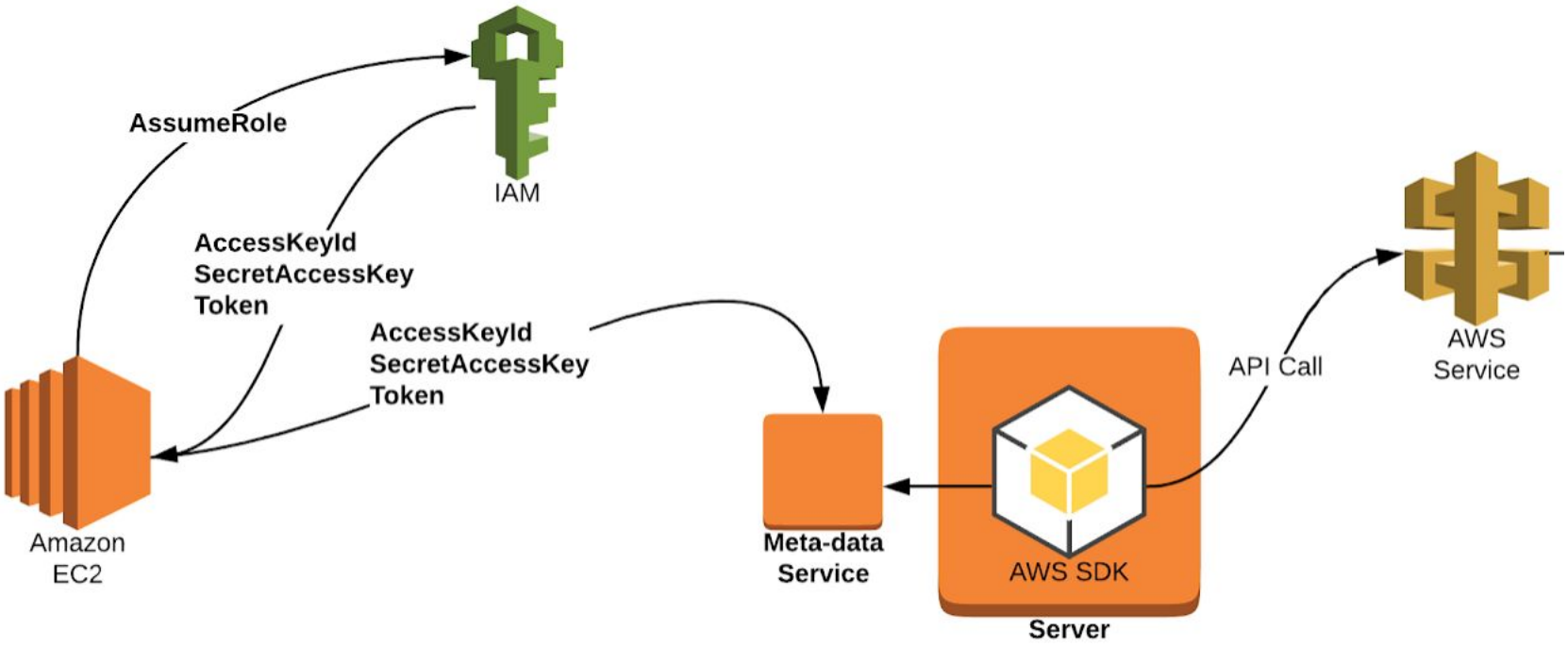
```
$ aws ec2 create-vpc --cidr-block 10.0.0.0/16
$ VPCID=vpc-06a876984aadcc3fa #<-- VPC id from the above command

$ aws ec2 create-subnet --cidr-block 10.0.0.0/17 --vpc-id $VPCID --availability-zone us-east-1a
$ PubSub=subnet-04c60b22077c116e4

$ aws ec2 create-subnet --cidr-block 10.0.128.0/17 --vpc-id $VPCID --availability-zone us-east-1c
$ PriSub=subnet-0f41447db258895ff

$ aws ec2 create-internet-gateway
$ IGW=igw-0969ad6e052b84741 #<-- Internet Gateway id generated from create-internet-gateway command

$ aws ec2 attach-internet-gateway --vpc-id $VPCID --internet-gateway-id  $IGW

$ aws ec2  create-route-table --vpc-id $VPCID
$ PubRte=rtb-0a88a2e5a2944e81f

$ aws ec2 create-route --route-table-id $PubRte --destination-cidr-block 0.0.0.0/0 --gateway-id $IGW

$ aws ec2 associate-route-table --route-table-id $PubRte --subnet-id $PubSub
```

```
#!/bin/bash

VPCID=$(aws ec2 create-vpc --cidr-block 10.0.0.0/16 --output text | grep VPC | awk '{print $7}')

PubSub=$(aws ec2 create-subnet --cidr-block 10.0.0.0/17   --vpc-id $VPCID --availability-zone us-east-1a --output
text | awk '{print $9}')

PriSub=$(aws ec2 create-subnet --cidr-block 10.0.128.0/17 --vpc-id $VPCID --availability-zone us-east-1c --output
text | awk '{print $9}')

IGW=$(aws ec2 create-internet-gateway | awk '{print $2}')

aws ec2 attach-internet-gateway --vpc-id $VPCID  --internet-gateway-id  $IGW

PubRte=$(aws ec2  create-route-table --vpc-id $VPCID | grep rtb | awk '{print $2}')

aws ec2 create-route --route-table-id $PubRte --destination-cidr-block 0.0.0.0/0 --gateway-id $IGW

aws ec2 associate-route-table --route-table-id $PubRte --subnet-id $PubSub
```

```
SGID=$(aws ec2 create-security-group --description 'open ports 22 and 80 to the world' --group-name
open-ssh-and-web --vpc-id $VPCID)

aws ec2 authorize-security-group-ingress --group-id $SGID --protocol tcp --port 80 --cidr 0.0.0.0/0

aws ec2 authorize-security-group-ingress --group-id $SGID --protocol tcp --port 22 --cidr 0.0.0.0/0

sudo yum install jq -y

AMI=$(aws ec2 describe-images --owners amazon --filters 'Name=name,Values=amzn2-ami-hvm-2.0.????????-x86_64-gp2'
'Name=state,Values=available' --output json | jq -r '.Images | sort_by(.CreationDate) | last(.[]).ImageId')

EC2ID=$(aws ec2 run-instances --image-id $AMI --count 1 --instance-type t2.micro --key-name e91_key
--security-group-ids $SGID --subnet-id $PubSub --associate-public-ip-address | grep INSTANCES | awk '{print $7}')

aws ec2 describe-instances --instance-ids $EC2ID
```

```
PUBIP=$(aws ec2 describe-instances --instance-ids $EC2ID --output json --query
'Reservations[0].Instances[0].PublicIpAddress' --output text)

ssh $PUBIP 'sudo ls /tmp/'

ssh $PUBIP 'sudo yum install -y httpd'

ssh $PUBIP 'sudo systemctl start httpd'

ssh $PUBIP 'sudo systemctl enable httpd'

ssh $PUBIP 'echo CSCI-E91 > /tmp/index.html'

ssh $PUBIP 'sudo cp /tmp/index.html /var/www/html/'

curl $PUBIP
ESCI-E91


#aws ec2 terminate-instances --instance-ids $EC2ID
```

Role and Security Token Service



```
ssh EC2WithRoleIP 'curl http://169.254.169.254/latest/meta-data/iam/security-credentials/EC2FullAccess'
{
  "Code" : "Success",
  "LastUpdated" : "2018-10-02T19:08:01Z",
  "Type" : "AWS-HMAC",
  "AccessKeyId" : "ASIARGRHK43OKHVSUYP2",
  "SecretAccessKey" : "j5vJ1dcXMNNtWuKv/RSkmpKDdOH9b9ufOixhz4Mq",
  "Token" : "FQoGZXIvYXdzEPX ..=”,
  "Expiration" : "2018-10-03T01:28:14Z"
}

$ export AWS_ACCESS_KEY_ID=ASIARGRHK43OKHVSUYP2
$ export AWS_SECRET_ACCESS_KEY=j5vJ1dcXMNNtWuKv/RSkmpKDdOH9b9ufOixhz4Mq
$ export AWS_SESSION_TOKEN=FQoGZXIvYXdzEPX...<remainder of security token>
$ aws ec2 describe-instances
```

```
sudo yum install  python3
sudo yum install  python3-pip

sudo pip3 install boto3
sudo pip3 install ipython


$ python
```

```
import boto3

# http://boto3.readthedocs.io/en/latest/reference/services/ec2.html#service-resource

#ec2 = boto3.resource('ec2', aws_access_key_id='AWS_ACCESS_KEY_ID',
aws_secret_access_key='AWS_SECRET_ACCESS_KEY', region_name='us-east-1')

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

# create VPC
vpc = ec2.create_vpc(CidrBlock='10.0.0.0/16')

# Wait for the VPC to be created
vpc.wait_until_available()

# Create internet gateway - IGW
igw = ec2.create_internet_gateway()

# Attach IGW to the VPC
vpc.attach_internet_gateway(InternetGatewayId=igw.id)
```

```
# create a public subnet
public_subnet = ec2.create_subnet(CidrBlock='10.0.0.0/17', AvailabilityZone='us-east-1a', VpcId=vpc.id)

# create a private subnet
private_subnet = ec2.create_subnet(CidrBlock='10.0.128.0/17', VpcId=vpc.id)

# Create a route table for the public subnet
route_table = vpc.create_route_table()
print(route_table.id)

# Add a public route to the public routing table
route = route_table.create_route (DestinationCidrBlock='0.0.0.0/0',GatewayId=igw.id)


# associate the route table with the public_subnet
route_table.associate_with_subnet(SubnetId=public_subnet.id)
```

```
# Create security group
sg = ec2.create_security_group(GroupName='open-ssh-and-web', Description='open ports 22 and 80 to the world',
VpcId=vpc.id)

sg.authorize_ingress(CidrIp='0.0.0.0/0', IpProtocol='tcp', FromPort=22, ToPort=22)
sg.authorize_ingress(CidrIp='0.0.0.0/0', IpProtocol='tcp', FromPort=80, ToPort=80)


# Create instance using AMI image ami-0e6d2e8684d4ccb3e

instances = ec2.create_instances(
    ImageId='ami-0e6d2e8684d4ccb3e', InstanceType='t3.small', MaxCount=1, MinCount=1,
    NetworkInterfaces=[{'SubnetId': public_subnet.id, 'DeviceIndex': 0, 'AssociatePublicIpAddress': True,
'Groups': [sg.group_id]}])

instances[0].wait_until_running()

print( "Instance with ID " + instances[0].id +  " Is ready ")
```

Load Balancer

- Layer 7 HTTP/HTTPS Application Layer Load Balancer
  - Target groups and Rules to route paths
- Layer 4 Network Load Balancer
  - Extreme performance is required.
  - Low latency while handling millions of requests per second
- Classic Load Balancer: Layer 4 and can use layer 7
  - X-Forwarding
  - sticky session

Live Demo Load Balancer

Auto scaling

- ➔ Launch Configuration :
  - ◆ EC2 launch configuration with userdata
- ➔ Auto scaling :
  - ◆ Auto scaling configuration using the Launch configuration and Auto Scaling Groups

Live Demo Auto Scaling

404 - Not Found

How to update the Script in launch configuration ?

1) Copy the launch configuration and choose new name
    a) Update the user data
    b) Make sure all config is ok
2) Replace the Autoscaling with the new Launch Configuration
3) Remove instances: for each instance in the LB
    a) terminate the instance
    b) wait for the new instance to come up.
    c) Test the new instance functionality
4) Remove the old launch configuration

Resources

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/iam-roles-for-amazon-ec2.html

https://docs.aws.amazon.com/autoscaling/ec2/userguide/what-is-amazon-ec2-auto-scaling.html

https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-welcome.html

https://docs.aws.amazon.com/cli/latest/userguide/cli-roles.html

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-metadata.html