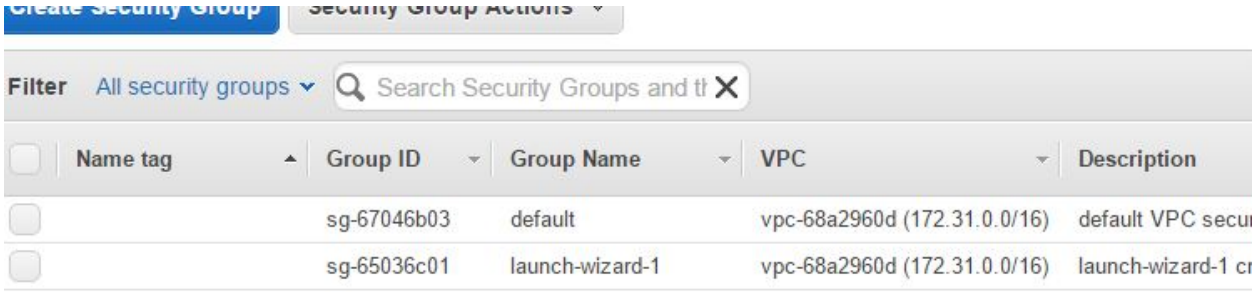
1. Create a security group that allows ssh access

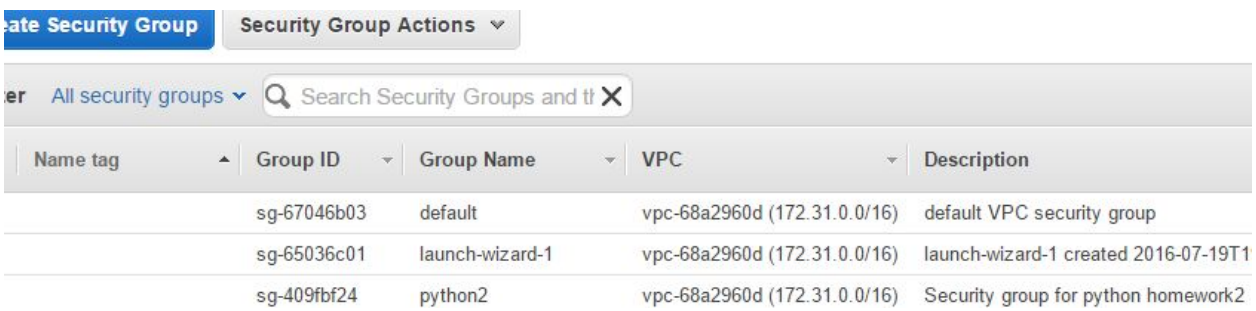
Before creation



Creating

```
>>> app = ec2.create_security_group(VPC_GROUP_NAME, VPC_GROUP_DESCRIPTION )
>>> sg = ec2.get_all_security_groups(filters={'group-name': [VPC_GROUP_NAME
>>> app.authorize('tcp','22', '22', "0.0.0.0/0")
True
>>>
>>> pprint (vars(sg[0]))
{'connection': EC2Connection:ec2.us-west-1.amazonaws.com,
 'description': u'Security group for python homework2',
 'id': u'sg-409fbf24',
 'item': u'\n        ',
 'name': u'python2',
 'owner_id': u'413365813430',
 'region': RegionInfo:us-west-1,
 'rules': [],
 'rules_egress': [IPPermissions:-1(None-None)],
 'tags': {},
 'vpc_id': u'vpc-68a2960d'}
>>> sgGroupId =  str(sg[0].id)
>>> sgVPCId = str(sg[0].vpc_id)
>>>
>>> vpccon = boto.vpc.connect_to_region(REGION)
```

After creation



2. Create a key-pair that is used for ssh access

Before

```
siming.meng@USABOS147305L ~/.aws
$ pwd
/home/siming.meng/.aws

siming.meng@USABOS147305L ~/.aws
$ ls -axp
./  ../  ami.txt  config  credentials  key-west1.pem

siming.meng@USABOS147305L ~/.aws
$ |
```

Between

```
>>> vpc = vpccon.get_all_vpcs(vpc_ids=[sgVPCId])[0];
>>>
>>> #get security group id, from "Security Groups" section of EC2 management con
... group = ec2.get_all_security_groups(group_ids=[sgGroupId])[0]
>>> sn=vpccon.get_all_subnets(filters={'vpcId':[sgVPCId]})
>>> sn1=sn[0]
>>> key = ec2.get_all_key_pairs(keynames=[KEY_NAME])[0]
>>> newKeyname = 'mynewkey2'
>>> newKey = ec2.create_key_pair(newKeyname)
>>> newKey.save("./")
True
>>> |
```

After

```
siming.meng@USABOS147305L ~/.aws
$ ls
ami.txt  config  credentials  key-west1.pem  mynewkey1.pem  mynewkey2.pem

siming.meng@USABOS147305L ~/.aws
```

3. Create two AMI Micro instances in us-west-1

Before creation

Launch Instance | Connect | Actions

Filter by tags and attributes or search by keyword

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Al
West1	i-06b03430774725f56	t2.micro	us-west-1a	terminated		No

Between

```
>>> instance1 = ec2.run_instances('ami-31490d51', instance_type='t2.mi
n1.id, key_name=newKeyname)
>>> pprint(vars(instance1))
{'RunInstancesResponse': u'\n',
 'connection': EC2Connection:ec2.us-west-1.amazonaws.com,
 'groups': [],
 'id': u'r-0af94613a6485fb54',
 'instances': [Instance:i-062bf0dea3f84f47d],
 'owner_id': u'413365813430',
 'region': RegionInfo:us-west-1,
 'requestId': u'bdb4ab5b-96f6-4656-8f98-7ec312cd3c12'}
>>> id1 = str(instance1.instances[0].id)
>>> time.sleep(1)
>>> print 'int1 [', id1 , '] created.'
int1 [ i-062bf0dea3f84f47d ] created.
>>>
>>> # wait for initialization to complete
    while True:
```

```
>>> instance2 = ec2.run_instances('ami-31490d51', instance_type=
n1.id, key_name=newKeyname)
>>> pprint(vars(instance2))
{'RunInstancesResponse': u'\n',
 'connection': EC2Connection:ec2.us-west-1.amazonaws.com,
 'groups': [],
 'id': u'r-014a72538f43e5d43',
 'instances': [Instance:i-0252143fe7414a70f],
 'owner_id': u'413365813430',
 'region': RegionInfo:us-west-1,
 'requestId': u'1a4f9369-985d-440b-8106-812ae4fb86aa'}
>>> id2 = str(instance2.instances[0].id)
>>> time.sleep(1)
>>>
```

After creation

nch Instance | Connect | Actions

Filter by tags and attributes or search by keyword

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status C
	i-0252143fe7414a70f	t2.micro	us-west-1b	running	2/2 c
	i-062bf0dea3f84f47d	t2.micro	us-west-1b	running	2/2 c
West1	i-06b03430774725f56	t2.micro	us-west-1a	terminated	

4. Print out information about these two instances

```
>>> print 'int1 [', id1 , '] created.'
int1 [ i-062bf0dea3f84f47d ] created.
>>> print 'int2 [', id2 , '] created.'
int2 [ i-0252143fe7414a70f ] created.
>>> |
```

5. Programmatically terminate both instances

```
>>> term1= ec2.terminate_instances(instance_ids=[str(id1)])
>>> term2= ec2.terminate_instances(instance_ids=[str(id2)])
>>> |
```

ıme	Instance ID	Instance Type	Availability Zone	Instance State	S
	i-0252143fe7414a70f	t2.micro	us-west-1b	terminated	
	i-062bf0dea3f84f47d	t2.micro	us-west-1b	terminated	
est1	i-06b03430774725f56	t2.micro	us-west-1a	terminated	