UNI: kb2896
Name: KUNAL BAWEJA
Subject: COMS E6998 Cloud Computing & Big Data

Screenshots for Creating AWS Instances Programatically

Mini-HW-1 (Part-2)

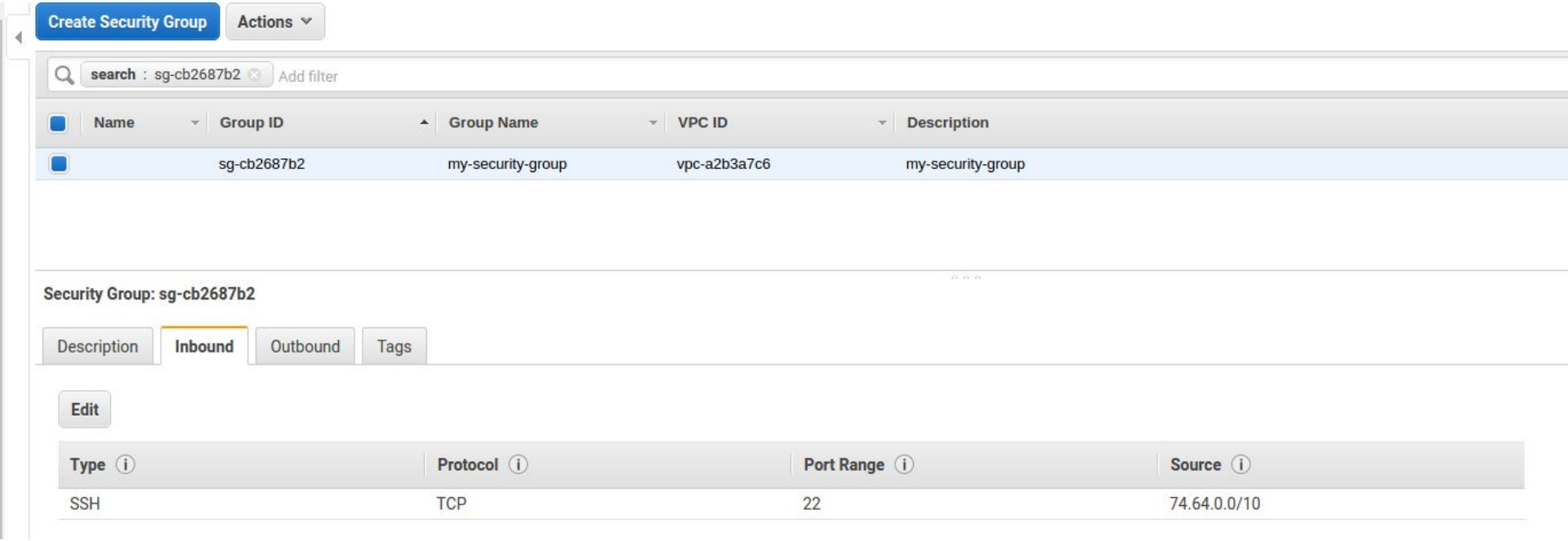
CreateEC2Sample.java | CreateSecurityGroupSample.java

```
            CreateSecurityGroupResult result = ec2
                    .createSecurityGroup(securityGroupRequest);
            System.out.println("Security group created: " + result.getGroupId());
        } catch (AmazonServiceException ase) {
            // Likely this means that the group is already created, so ignore.
            System.out.println(ase.getMessage());
        }

        // addr.getHostAddress() returns 127.0.1.1 but we need external ip address of our system
        // I consulted TA and he advised it is ok to hardcode the ip as below because getHostAddress
        // does not works as expected, if the system is behind a NAT. Following is an IP address
        // from Columbia University Network
        String ipAddr = "74.73.4.15/10";

//        String ipAddr = "0.0.0.0/0";
//
//         Get the IP of the current host, so that we can limit the Security Group
//         by default to the ip range associated with your subnet.
//        try {
//            InetAddress addr = InetAddress.getLocalHost();
//
//            // Get IP Address
//            ipAddr = addr.getHostAddress()+"/10";
//        } catch (UnknownHostException e) {
//        }

        // Create a range that you would like to populate.
        List<String> ipRanges = Collections.singletonList(ipAddr);

        // Open up port 22 for TCP traffic to the associated IP from above (e.g. ssh traffic).
        IpPermission ipPermission = new IpPermission()
                .withIpProtocol("tcp")
                .withFromPort(new Integer(22))
                .withToPort(new Integer(22))
                .withIpRanges(ipRanges);

        List<IpPermission> ipPermissions = Collections.singletonList(ipPermission);
```

Tasks | Console

```
<terminated> CreateSecurityGroupSample (1) [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Sep 26, 2016 11:44:29 PM)
Security group created: sg-cb2687b2
Ingress port authroized: [[{IpProtocol: tcp,FromPort: 22,ToPort: 22,UserIdGroupPairs: [],IpRanges: [74.73.4.15/10],PrefixListIds: []}]]
```

Create Security Group with Restriced IP Access Rule for SSH



Verify my-security-group

CreateEC2Sample.java | CreateSecurityGroupSample.java

```
                        .withKeyName("my-key-pair")
                        .withSecurityGroups("my-security-group");

        RunInstancesResult runInstancesResult = ec2.runInstances(runInstancesRequest);

        //TODO: Do something with the result
        String instanceResultString = new String(runInstancesResult.toString());
        List<String> instanceIdList = new ArrayList<String>();

        Matcher matcher = Pattern.compile("InstanceId:\\s*(\\S+),").matcher(instanceResultString);

        while(matcher.find())
            instanceIdList.add(matcher.group(1));

        //Sleep for 10 seconds, let ip allocation occur
        //Discussed with TA in Office Hours
        System.out.println("Sleep 10 seconds for public ip allocation and print details\n");
        try
        {
            TimeUnit.SECONDS.sleep(10);
        }
        catch (InterruptedException e)
        {
            e.printStackTrace();
        }

        //Get Instance Description
        DescribeInstancesRequest describeIntancesRequest = new DescribeInstancesRequest();
        describeIntancesRequest.withInstanceIds(instanceIdList);

        DescribeInstancesResult describeInstancesResult = ec2.describeInstances(describeIntancesRequest);
        List <Reservation> reservationList = describeInstancesResult.getReservations();
        List<Instance> instancesList;

        System.out.println("InstanceId\tPublicIp\tRegion");

        for(Reservation reservation: reservationList)
        {
            instancesList = reservation.getInstances();
            for (Instance instance: instancesList)
            {
                System.out.println(instance.getInstanceId() + "\t"
                        + instance.getPublicIpAddress() + "\t"
                        + instance.getPlacement().getAvailabilityZone());
            }
```

Tasks | Console

<terminated> CreateEC2Sample [Java Application] /usr/lib/jvm/java-8-openjdk-amd64/bin/java (Sep 26, 2016 11:47:45 PM)

```
gYB4J12WRZxXzwhpKiA2ekIem+gPbINj46j7MUWE6fh/mgdNV4AgfSZe+JcdfRC1R2UMAyR5eMiQ
OFs27qPwDEu1J0Zu8mY/IuHH1Cr2TXNfDTBZx+F3HBvkB4f/VM9NRm+/zcsBN4S9THZb1fKV8+IR
58ahmx5s3z6GfKKs/xTP3QKBgHhWIngLSElLkud6RgacESCV1Jof8sTZ3NbZ+6BApJj++i78MKWV
gIXqa2HBHr4PLclmxJjVQ/hYxfjwT0UlaL4ORB7vb6c9hqqMEj5SAUbV3t4WiGYwF4caOIBPx4l
DARdUUnwarPMgMflfR2YyS1SYE6qJ4LyScNt1PA+scnFAoGBAL+ozb71Kr9y+ARBKMe5CUbjOCmV
jGTfi/1Gn9XsO6xKcIfkJ/7Bk5GvyZdIaWOCL7M6CzRfdxw02kIJPAzFSDOd/D4GKSDK3jVGngJx
m1MwER8PQjx/n1hz+DaPIsX3nQMZgUUy46/Ya3h9JXoZJETSFjf5v7Ns5s3wSy8bX71n
-----END RSA PRIVATE KEY-----

Sleep 10 seconds for public ip allocation and print details

InstanceId      PublicIp        Region
i-0d1f13297e106f3cc     54.213.113.124  us-west-2b
```

Launch 1 EC2 Instance programatically and print it's details: instance id, public ip address and aws region(zone)

```
kunal@baweja:~/Documents/cloud-bigdata-6998/aws$ ssh -i "my-key-pair.pem" ec2-user@54.213.113.124
The authenticity of host '54.213.113.124 (54.213.113.124)' can't be established.
ECDSA key fingerprint is SHA256:NOEAnaviadVkMSy5bRwtySCbf0Fmsb4Jxaxgq1xYUIU.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '54.213.113.124' (ECDSA) to the list of known hosts.

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2016.03-release-notes/
13 package(s) needed for security, out of 26 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-29-174 ~]$
```

Verify launched instance by SSH