Linux Shell Scripting

- Linux Shell Scripting
  - Security Group Modification using Shell Script
    - Shell Script Enhancements
  - Shell Script Assignments
  - Reference for AWS CLI JMESPATH Commands

--

Security Group Modification using Shell Script

- Find a Security Group by a Name and add ingress/inbound rules to this security group from a csv file.

1. AWS CLI Command to add one security group ingress rule.
2. Find a particular Security Group ID when Name is given.
3. Read a csv file per line and split the entries in csv file as per delimiter
4. Add the inbound rule in the security group.

--

- The above use case can be done using `AWS CLI` or using Python(boto3 library) as well. We will see it as below using aws cli.
- Find security group options in the aws cli command

```
aws ec2 help | grep security
```

- Generic command to add a inbound security group will look like.

```
aws ec2 authorize-security-group-ingress --protocol tcp --port 22 --cidr
192.168.1.1/32 --group-id sg-03aa0c3fe2a061293

aws ec2 authorize-security-group-ingress --group-name jenkins-sg --protocol tcp --
port 22 --cidr 203.0.113.0/24

aws ec2 authorize-security-group-ingress --protocol tcp --port $port --cidr ${ip}
--group-id $group_id
```

```
aws ec2 describe-security-groups --region ap-south-1
```

- Replace the Group Name in the below command as per your AWS Account.

```
```

```
aws ec2 describe-security-groups --query "SecurityGroups[?
GroupName=='LBSecurityGroup']" --region ap-south-1
```

--

- Below is the sample Output for above command


```
[
    {
        "Description": "default created 2018-12-03T21:11:54.345+05:30",
        "GroupName": "default",
        "IpPermissions": [
            {
                "IpProtocol": "-1",
                "IpRanges": [
                    {
                        "CidrIp": "49.35.206.245/32"
                    }
                ],
                "Ipv6Ranges": [],
                "PrefixListIds": [],
                "UserIdGroupPairs": []
            },
            {
                "FromPort": 22,
                "IpProtocol": "tcp",
                "IpRanges": [
                    {
                        "CidrIp": "49.35.206.245/32"
                    }
                ],
                "Ipv6Ranges": [],
                "PrefixListIds": [],
                "ToPort": 22,
                "UserIdGroupPairs": []
            }
        ],
        "OwnerId": "0829123458139",
        "GroupId": "sg-0a97252c1661bf218",
        "IpPermissionsEgress": [
            {
                "IpProtocol": "-1",
                "IpRanges": [
                    {
                        "CidrIp": "0.0.0.0/0"
                    }
                ],
                "Ipv6Ranges": [],
                "PrefixListIds": [],
                "UserIdGroupPairs": []
            }
```

```
        ],
        "Tags": [
            {
                "Key": "Name",
                "Value": "Test-SG"
            }
        ],
        "VpcId": "vpc-7ad93503"
    }
]
```

--

- Shell Script individual commands

```
aws ec2 describe-security-groups --query "SecurityGroups[?
GroupName=='LBSecurityGroup'].[GroupId]" --region ap-south-1

## OUTPUT
[
    [
        "sg-0a97252c1661bf218"
    ]
]

aws ec2 describe-security-groups --query "SecurityGroups[?
GroupName=='LBSecurityGroup'].[GroupId]" --output text --region ap-south-1

sg-0a97252c1661bf218
```

```
aws ec2 describe-security-groups --query "SecurityGroups[?
GroupName=='LBSecurityGroup']"
```

--

- Add a single inbound rule using aws cli command:

```
aws ec2 authorize-security-group-ingress --protocol tcp --port 80 --cidr
10.0.0.0/24 --group-id sg-0a97252c1661bf218
```

- `echo` the content of the csv file using shell for loop command

```
for i in $(cat inbound_rules.csv); do echo "This is i: $i"; done
```

- Now we need to get individual value from each line that is separated using `,` and execute the `aws cli` command to add the ip in the security group.
- Get the individual value using `awk` utility.

```
cat ./inbound_rules.csv | awk -F, '{ print $1 }'
cat ./inbound_rules.csv | awk -F, '{ print $2 }'
```

```
#!/bin/bash
for i in $(cat inbound_rules.csv);
do
    echo "Value of i from file is $i"
    INBOUND_IP=$(echo $i | awk -F, '{ print $1}')
    INBOUND_PORT=$(echo $i | awk -F, '{ print $2}')
    echo "Inbound ip is $INBOUND_IP"
    echo "Inbound port is $INBOUND_PORT"
done
```

```
for i in $(cat inbound_rules.csv); do echo $i | awk -F, '{ print $1 $2 }'; done
```

--

- In the above shell script we have individual variable value for IP and Port, we can include the command to add security group inbound rule to run for each rule to be added to specific group.

```
#!/bin/bash
SECURITY_GROUP_NAME="Demo-ALB-SG"
SECURITY_GROUP_ID=$(aws ec2 describe-security-groups --query "SecurityGroups[?
GroupName=='$SECURITY_GROUP_NAME'].[GroupId]" --output text --region us-east-1)
echo "SECURITY_GROUP_ID value is $SECURITY_GROUP_ID"
for i in $(cat inbound_rules.csv);
do
    INBOUND_IP=$(echo $i | awk -F, '{ print $1}')
    INBOUND_PORT=$(echo $i | awk -F, '{ print $2}')
    echo "Inbound ip is $INBOUND_IP"
    echo "Inbound port is $INBOUND_PORT"
    aws ec2 authorize-security-group-ingress --protocol tcp --port $INBOUND_PORT -
-cidr $INBOUND_IP --group-id $SECURITY_GROUP_ID --region us-east-1
done
```

- Execute the above shell script and validate the entries in the SG

--

Shell Script Enhancements

- Pass the **SECURITY_GROUP_NAME** value as the command line parameter.
- Take inbound csv file path as Command Line Argument
- Add exception handling for `SECURITY_GROUP_ID` variable
- Add path for input csv file, also check if the path is correct
- Shell script should run in any AWS Region.

```
#!/bin/bash
SECURITY_GROUP_NAME=$1
INPUT_FILE_NAME=$2
REGION_NAME=$3
SECURITY_GROUP_ID=$(aws ec2 describe-security-groups --region $REGION_NAME --query
"SecurityGroups[?GroupName=='$SECURITY_GROUP_NAME'].[GroupId]" --output text)
echo "SECURITY_GROUP_ID value is $SECURITY_GROUP_ID --region $REGION_NAME"
if [ $SECURITY_GROUP_ID != "" ]; then
    if [ -f $INPUT_FILE_NAME ]; then
        for i in $(cat $INPUT_FILE_NAME);
        do
            INBOUND_IP=$(echo $i | awk -F, '{ print $1}')
            INBOUND_PORT=$(echo $i | awk -F, '{ print $2}')
            echo "Inbound ip is $INBOUND_IP"
            echo "Inbound port is $INBOUND_PORT"
            aws ec2 authorize-security-group-ingress --region $REGION_NAME --
protocol tcp --port $INBOUND_PORT --cidr $INBOUND_IP --group-id $SECURITY_GROUP_ID
        done
    else
        echo "File $INPUT_FILE_NAME does not exists"
    fi
else
    echo "$SECURITY_GROUP_ID is blank, cannot execute"
fi

# Execute shell script with below cli parameters
# bash add_inbound.sh default inbound_rules.csv us-west-2
# bash add_inbound.sh ElasticMapReduce-slave inbound_rules.csv us-east-2
```

--

Shell Script Assignments

1. Write a Shell Script to find all Elastic IP Address in all AWS Regions, send an email with list of all Available Elastic IP Address specifying the region in which it is Available.
2. Write a Shell Script to find all EBS Volumes in all AWS Regions, send an email with list of all Available EBS Volumes specifying the region in which it is Available.
3. Write a shell script to find all IAM Roles with AdministratorAccess Policy and write the Role Name into a file and send an email with list of these IAM role to AWS Team.
4. Above security group example to be added as Python Boto3 Script.

--

Reference for AWS CLI JMESPATH Commands

```
# Using dictionary notation, if we want to get the value of the nested json list
to get the InstanceId that is present within the Attachments Nested Json List.
aws ec2 describe-volumes --query 'Volumes[*].
{ID:VolumeId,InstanceId:Attachments[0].InstanceId,AZ:AvailabilityZone,Size:Size}'
aws ec2 describe-vpc-endpoint-services --query 'ServiceDetails[?
ServiceName==`com.amazonaws.ap-south-1.s3`].AvailabilityZones'
# lists the five most recent Amazon Machine Images (AMIs) that amazon has created,
sorted from most recent to oldest.
aws ec2 describe-images --owners amazon --query
'reverse(sort_by(Images,&CreationDate))[:5].
{id:ImageId,date:CreationDate,Name:Name, Public:Public}'
# describe subnet to get SubnetId and AZ
aws ec2 describe-subnets --query 'Subnets[*].
{AZ:AvailabilityZone,SubnetId:SubnetId}'
aws ec2 describe-subnets --query 'Subnets[*].
{AZ:AvailabilityZone,SubnetId:SubnetId}' --output text
# Get a list of policies attached to all Roles in IAM
aws iam list-roles --query Roles[0]
aws iam list-roles --query Roles[0].RoleName
aws iam list-attached-role-policies --role-name AWS_CICD_ROLE
for role_name in $(aws iam list-roles --query Roles[*].RoleName --output text);
do
echo "This is role_name : $role_name";
aws iam list-attached-role-policies --role-name $role_name --query
AttachedPolicies[*].[PolicyName]
done
for role_name in $(aws iam list-roles --query Roles[*].RoleName --output text); do
echo ${role_name} && aws iam list-attached-role-policies --role-name ${role_name}
--output text; done
for role_name in $(aws iam list-roles --query Roles[*].RoleName --output text); do
echo ${role_name} && aws iam list-attached-role-policies --role-name ${role_name}
--query AttachedPolicies[*] --output text; done
for i in {1..5}; do echo $i; done
```