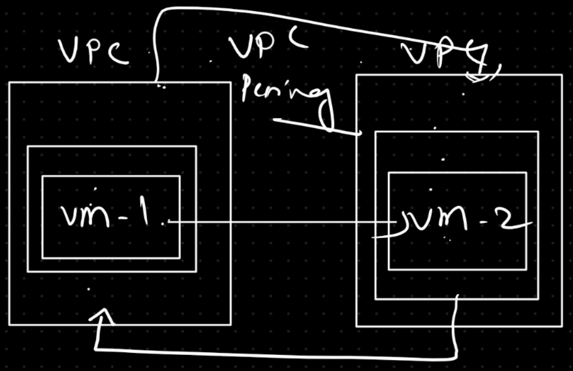
⇒ VPC -> VPC Peering
SG
NACL }

EFS
Cloudformation
Terraform }

Terraform
Ansible }

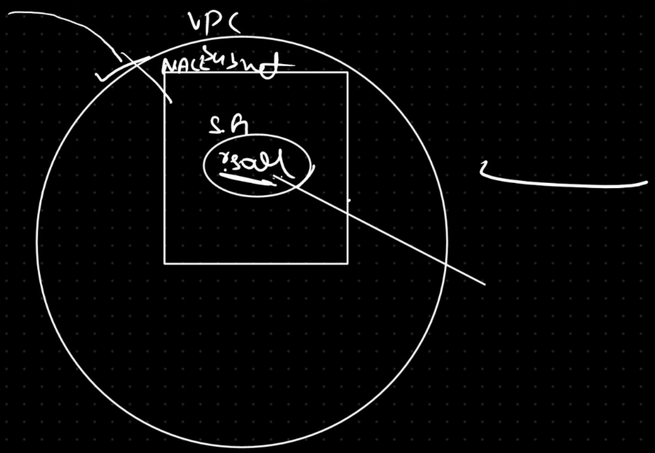


Security Groups
============
--> Acts like a Firewall to secure our resources
--> SG contains InBound Rules(Incoming Traffic) and Outbound Rules(Outgoing traffic)

__> In One SG we can ad 50 Rules

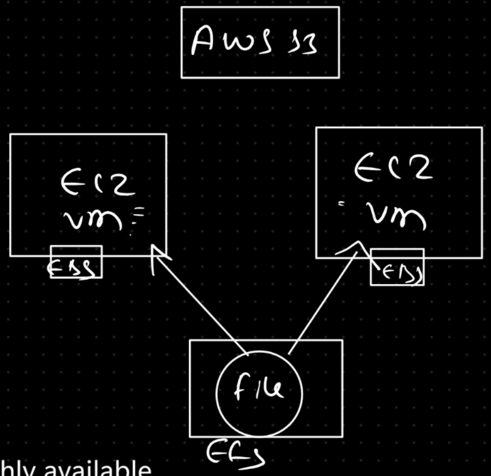
-->SG Only allow rules (by default all rules are denied)
--> We cannot configure deny rule in security group
--> Security Group are applicable at resource level and manually we have to add security group to a resource
--> Security Group are statefull

NACL(network access control list)
==================
NACL acts a firewall for our subnets in VPC
Its applicable at the subnet level
NACL Rules are applicable for all the resources which are part of a subnet
NACL is stateless
In NACL we can configure both Allow and deny rules
One Subnet can have only one NACL However One NACL can be added to multiplr subnets
NACL acts as first level of defense for incoming traffic and SG acta as first level of defense for outgoing traffic



Elastic File System (EFS)

→ EBS → Elastic Block store
→ S3 – object storage
→ EFS – File system storage



Managed File Storage for EC2

Advantage: Create scalable file storage to be used on EC2
Fully managed by AWS, Low Cost, pay for what you use and highly available and scalable performance.

Steps to work with EFS practicals:
Login to our AWS Console --> Services --> EFS (Under storage) --> Clicked on Create file system

AFter creating we will get File id: fs-04734653(example)

Create Ec2 instances (2 instances)

Login to EC2 instance(get connected) and Install NFS client
```
$ sudo yum install -y amazon-efs-utils
```

Create a folder/directory
```
$ sudo mkdir efsdir
```

Mouting --> Mount filesyatem
```
$ sudo mount -t efs -o tls fs-74384:/ efsdir   --> fs-74384 in place of this add your file id
```

```
$ cd efsdir
```

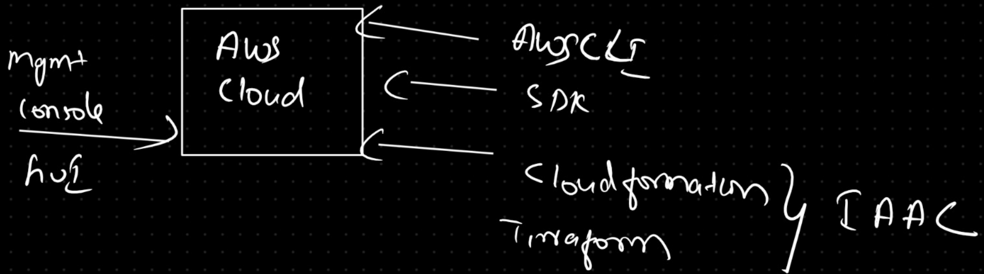
Creta file or do some operation

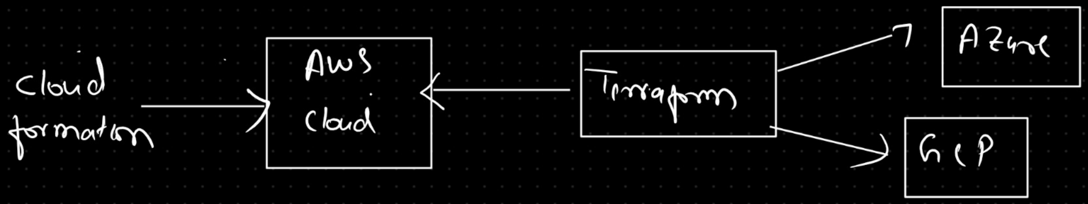
and then connect to instance number 2 and repeat same steps and check the behaviour of file shared system

---

Cloud formation :-

↳ IAAC → Infrastructure as a code





AWS CloudFormation provides a common language to describe and provision all the infrastructure resources in your environment in a safe, repeatable way.
CloudFormation--> create stack--> exiting template--> uploaded file file (below scrip save with .yml formatand upload)

```
AWSTemplateFormatVersion: '2010-09-09'
Description: Telusko - Build Linux Web Server

Parameters:
  LatestAmiId:
    Description: AMI for Amazon Linux 2 EC2 instance
    Type: 'AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>'
    Default: '/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2'

Resources:
  webserver1:
    Type: 'AWS::EC2::Instance'
    Properties:
      InstanceType: 't2.micro'
      ImageId: !Ref LatestAmiId
      SecurityGroupIds:
        - !Ref WebserverSecurityGroup
      Tags:
        - Key: 'Name'
          Value: 'teluskoserver1'
      UserData:
        Fn::Base64: !Sub |
          #!/bin/bash -xe
          yum update -y
          yum install httpd -y
          service httpd start
          chkconfig httpd on
          cd /var/www/html
          echo "<br>" >> index.html
          echo "<h2><b>Telusko Linux Demo</b></h2>" >> index.html

  WebserverSecurityGroup:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
      GroupDescription: 'Enable Port 80 for HTTP access'
      Tags:
        - Key: 'Name'
          Value: 'webserver-sg'
      SecurityGroupIngress:
        - IpProtocol: 'tcp'
          FromPort: '80'
          ToPort: '80'
          CidrIp: '0.0.0.0/0'
```
Verify EC2 dashboard and we can see server getting created.

EC2 + LBR + ASG
EBS
S3
RDS
IAM
VPC
Elastic Bean Stack
Cloud watch
SNS
CLI
EFS
Cloud formation
AWS Lamdas →

{ Terraform
Ansible } 4-5 { Mavin
Git }

Tomcat →
Docker
K8s
Jinkins
Nexus
Sonarqube }