Class 17

1. Types of EBS volumes

EC2 Dashboard -> Volumes -> Create Volume -> Volume Type

IOPS: input output per second (Drive performance calculated by IOPS)

SSD Drives: solid state drive has no mechanical components, have high performance (like pen drive, memory card in mobiles)

**General Purpose SSD (gp2)**

**Provisioned IOPS SSD (io1)**

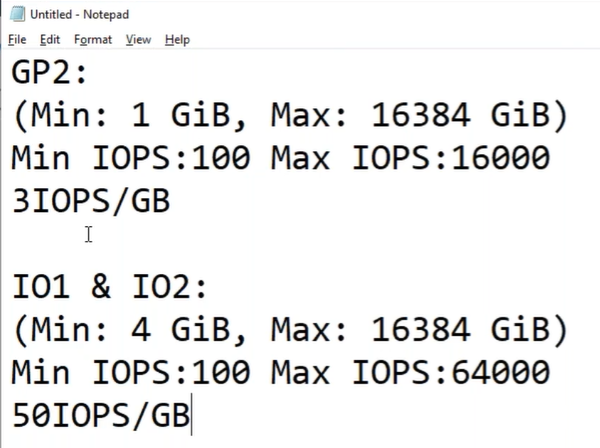
**Provisioned IOPS SSD (io2)**

HDD Drives: Hard disk drive has mechanical components, have low performance

**Cold HDD (sc1)**

**Throughput Optimized HDD (st1)**

**Magnetic (standard)**



**Now where to use these drives:**

**For example:**

**Types of Drives to use Microsoft Sql DB Servers:**

1. **OS: GP2**
2. **Page files/virtual memory: IS/GP2/ST (IS-internal store)**
3. **Software (drive to keep downloaded Sql software): GP2/ST1**
4. **In DB there are two files:**

**Data files: GP2/IO1/IO2**

**Log files: GP2/IO1/IO2 (Transaction log files)**

1. **Temp DB: IS/GP2/ST1 (no data in this file but we need Temp DB to work DB)**
2. **Backups: SC1/ST1/STD (to keep sql data backup)**

**\*Interview Question:**

**Could you tell me about your environment?**

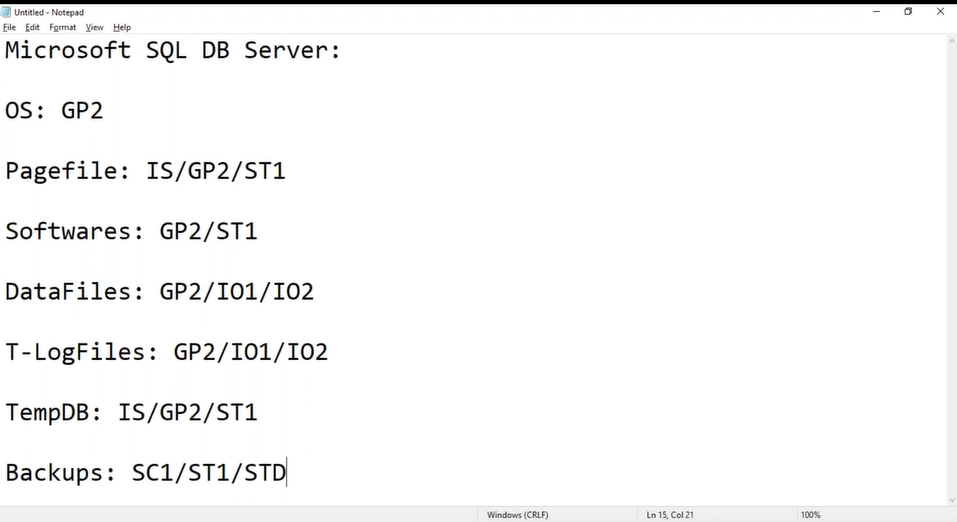
**A: we are AWS users and**

**we deployed all our infrastructure using cloud formation templates or Terraform**

**The Major workloads are Database servers as well as File Servers**

**In Database servers, we use multi EBS volumes like GP2/IO1/IO2 and SC1/ST1/STD**

**We are running OS in GP2……. (tell all Types of Drives to use Microsoft Sql DB Servers:)**

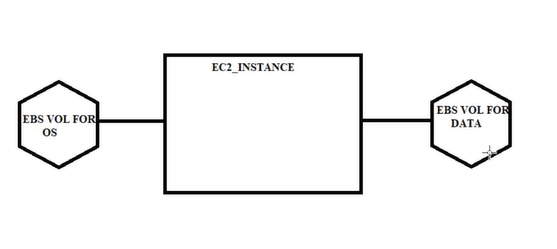


1. Use cases of EBS volume types: discussed in last class
2. AMI Vs Snapshots

AMI: Amazon Machine Image, we are using image while creating instance, we are not installing OS here.

* For every E2 instance there will be two EBS volumes, one is root volume for OS & other is for data.

When Launch an instance, t2.micro in “Add Volumes” we can see root volume



Now we launch instance and make changes in nginx server, like putting GIF file and create an image from that instance

Ec2 instance ->t2.micro -> install ->putty

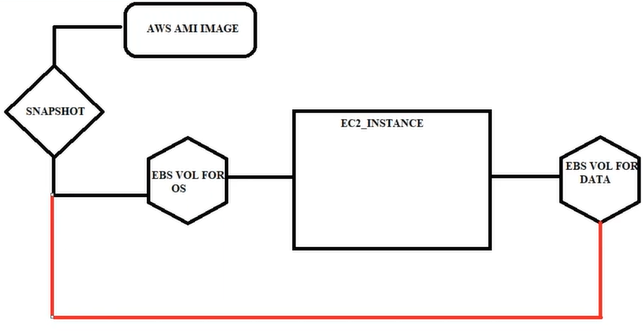
nano /usr/share/nginx/html/index.html (keep one GIF file)

chkconfig nginx on

Go to instance- >RC -> image -> create image

* When we created AMI image it doesn’t come directly, first one Snapshot creates and then creates image. That image be in -> images -> AMI’s

We generally don’t take Data volume backup



Now with this image we create **Instance**

Instance -> my AMI -> create

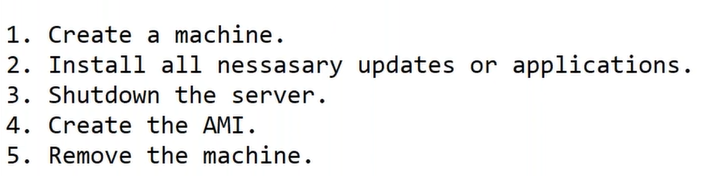
Go to instance and check test with Public DNS, you find same details

We can copy this AMI image in different regions AMI -> RC -> copy AMI ->

If we want to remove AMI -> RC -> Deregister

Snapshot -> Delete

Here we have done all the following steps manually



1. AMI creation and automated way using packer

Now here we must do all the following steps using automation

We must download Devopps packer tool from <https://www.packer.io/>

CMD run as administrator -> sysdm.cpl

System properties -> Advanced -> Environment Variables… -> User Variables -> path(packer path)

-> packer

Download visual studio code to edit our script <https://code.visualstudio.com/>

File -> Open Folder -> New File(myAMI.json)

<https://github.com/mavrick202> -> terraformsingleinstance -> packer.json -> Raw -> copy -> past in myAMI.json

now we need to fill this and Save code, this code contains all 5 above steps

"aws\_access\_key": "AKIAZ6OFFUXWZN6E6EHK",

      "aws\_secret\_key": "rspfecH0FekgYvgyv20C9B479dkhVeY2TRdlTU7c",

      "region": "us-east-1",

      "source\_ami":"ami-032930428bf1abbff",

      "instance\_type":"t2.micro",

      "vpc\_id": "vpc-0187b51aedcbd91e2",

      "subnet\_id": "subnet-057b872e980ff2a3e"

To get aws access key & aws secret key:

IAM -> users -> user name -> Attach existing policies directly -> AdministratorAccess -> copy and past access key & secret key

Goto CMD admin -> cd c:\Users\dheer\Desktop\packer\_1.6.4\_windows\_amd64 (myAMI.json path)

-> packer validate myAMI.json

-> packer build myAMI.json

Now all codes are executed, and AMI created

Launch an instance using My AMI to test

Total code for reference

{

    "\_comment" : "Create a AWS AMI ith AMZ Linux 2018 with Java and Tomcat",

    "variables": {

      "aws\_access\_key": "AKIAZ6OFFUXWZN6E6EHK",

      "aws\_secret\_key": "rspfecH0FekgYvgyv20C9B479dkhVeY2TRdlTU7c",

      "region": "us-east-1",

      "source\_ami":"ami-032930428bf1abbff",

      "instance\_type":"t2.micro",

      "vpc\_id": "vpc-0187b51aedcbd91e2",

      "subnet\_id": "subnet-057b872e980ff2a3e"

    },

    "\_comment1" : "packer build -var \"aws\_secret\_key=foo\" template.json",

    "\_comment2" : "packer build -var-file creds.json template.json",

    "builders": [{

      "access\_key": "{{user `aws\_access\_key`}}",

      "secret\_key": "{{user `aws\_secret\_key`}}",

      "type": "amazon-ebs",

      "region": "{{user `region`}}",

      "source\_ami": "{{user `source\_ami`}}",

      "instance\_type": "{{user `instance\_type`}}",

      "ssh\_username": "ec2-user",

      "ami\_name": "DevOpsClass-Build-{{isotime | clean\_resource\_name}}",

      "vpc\_id": "{{user `vpc\_id`}}",

      "subnet\_id": "{{user `subnet\_id`}}",

      "tags": {

        "Name": "DevOpsClass-Build-{{isotime | clean\_resource\_name}}"

      }

    }],

    "provisioners": [{

      "type": "shell",

      "inline": [

        "sleep 30",

        "sudo yum update -y",

        "sudo yum install nginx -y",

        "sudo yum install git -y",

        "sudo git clone https://github.com/mavrick202/webhooktesting.git",

        "sudo rm -rf /usr/share/nginx/html/index.html",

        "sudo cp webhooktesting/index.html /usr/share/nginx/html/index.html",

        "sudo cp webhooktesting/style.css /usr/share/nginx/html/style.css",

        "sudo cp webhooktesting/scorekeeper.js /usr/share/nginx/html/scorekeeper.js",

        "sudo service nginx start",

        "sudo chkconfig nginx on"

        ]

    }]

  }