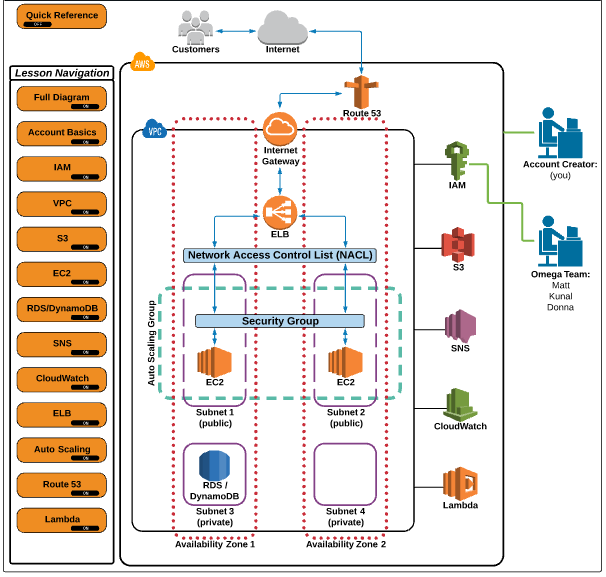
**Amazon Web Services**

Overview Design of AWS architecture



**Aws services:**

* EC2 – Elastic Cloud Compute
* EBS – Elastic Block Storage
* S3 – Simple Storage Service
* RDS – Relational Database Service
* DynamoDB – Nosql Database
* ELB – Elastic Load Balancing
* SNS – Simple Notification Service
* Lambda – Serverless Computing

**IAM:**

Identity Access management is where you manage your AWS users and their access to AWS accounts and Services.

The common use of IAM is to manage

* Users
* Groups
* Access Polices
* Roles

Note: The user created when you created the AWS account is called the root user by default is has full administrative access. Any users are created later has no access to any AWS services.

What is MFA?

Multi Factor Authentication it is an additional layer of security on root account that is provided by 3rd party.

How do get MFA code?

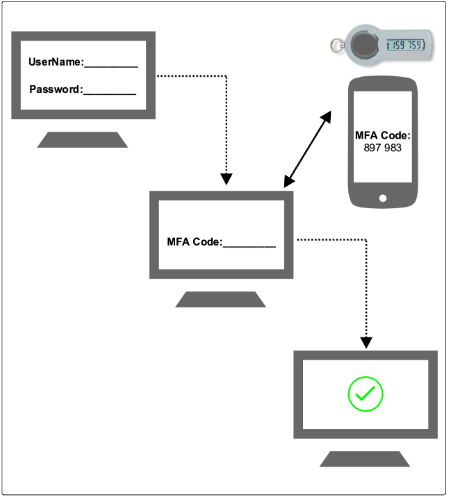
1. Virtual MFA Device

* Smart device
* Google Authenticator App

1. Hardware Key Fob

* Order it from AWS Account.

Example:



**VPC**

Virtual Private Cloud is a private sub section of AWS that you control in which you can place AWS resources such as ec2 instances and databases. You will have full control over who has access to the AWS resources that you place inside VPC.

Note: When you create an AWS account a default VPC is created with standard components that are need to functional and one account carries one VPC.

1. Internet gateway – IGW
2. Route Table ( with predefied routes to the default subnets)
3. Network Access Control List – NACL
4. Subnets to provision AWS resources like EC2 instances.
5. IGW is a horizontal scaled redundant and highly available VPC component that allows communication between instances in VPC and the internet.

* Default VPC already has an IGW.

1. Route Tables contains a set of rules called routes that are used to determine where network traffic is directed.

* Default VPC already has a main route table.

1. NCAL is a optional layer of security for VPC that acts as a firewall for controlling traffic in and out of one or more subnets.

* Default VPC already has an NACL.
* Default NACL allows both inbound and out bound traffics since the HTTP aloe rule has a lower rule. All other type of traffic will be denied via the catch all failsafe deny rule.
* New NACL when it created all traffic is denied by default.
* When it configured allow both inbound and out bound SSH traffic allow rule has a lower rule. Allow other type of traffic will be denied via the catch all failsafe deny rule.

1. What is Subnet?

When create a VPC it spans all of the availability zones in the region. After creating a VPC you can add one or more subnets in availability zone. Each subnet must reside entirely within one availability zone and cannot span zones.

**Subnet Rules:**

* Subnets must be associated with a route table.
* A public subnet has a route to the internet.
* A private subnet does not have a route to the internet.
* A subnet is located in one specific availability zone.
* Default VPC already has a main route table.

**Simple Storage Service (S3)**

S3 has a simple web services interface that you can use to store and retrieve any amount of data at any time, from anywhere on the web. It gives any user access to the same highly scalable, reliable, fast, inexpensive data storage infrastructure that amazon uses to run it owns global network of websites.

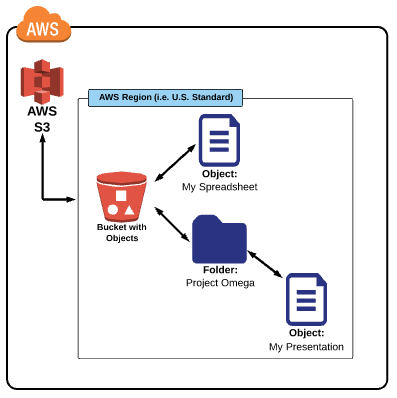
Components and structure:

1. Basics

* S3 – simple storage service
* AWS primary storage
* Can store any type of file

1. Buckets

* Root level folder created in S3 are referred to as buckets.
* Any sub folder you create in a bucket is referred to as a folder
* Bucket names can only contain lower case letters, numbers & hyphens.
* Bucket names must not be formatted as an ip address.



1. Uploading an object to a bucket:

* Navigate into a bucket
* Under “Actions” select “upload”
* Select a file to upload
* Click “start Upload”

1. Creating a folder in a bucket:

* Navigate in to a bucket
* Click on “create Folder”
* Give the folder a name

1. Bucket level Properties:

* General info
* Permissions
* Static Web Hosting
* Logging
* Events
* Versioning
* Life cycle
* Cross Region Replication
* Tags
* Requester Pays
* Transfer Acceleration

1. Folder Level Properties

* General Info
* Details

1. Object level Properties

* General Info
* Details
* Permissions
* Meta Data

1. Objects

* Files stored in a bucket are referred to as objects.

1. What is Storage Classes?

A storage class represents the classification assigned to each object in S3.

Available storage classes include:

* **Standard:** It’s designed for general all-purpose storage & default storage option. It is a eleven nines (99.99999999999%) object durability & 99.00 object availability. Is the most expensive storage class.
* **Reduced Redundancy Storage (RRS):** It’s designed for non-critical reproducible object. It is a 99.99% object durability & 99.99% object availability. Is less expensive than the standard storage class.
* **Infrequent Access (S3-IA):** It’s designed for objects that you do not access frequently but must be immediately available when accessed. It is a 99.99999999999% object durability & 99.90% object availability. Is less expensive than the standard RRS storage classes.
* **Glacier:**  It’s designed for long term archival storage. May take several hours for objects stored in glacier to be retrieved. It is 99.99999999999% object durability. Is the cheapest S3 storage class (very low cost)

Each storage class has varying attributes that dictate things like:

* Storage Cost
* Object Availability
* Object Durability
* Frequency of Access

Each object must be assigned to a storage class- standard is the default class

We can change the storage class of an object at any time.

1. Regions

* When you create a bucket you must select region for it to exist. This means that any data you upload to the S3 bucket will be physically located in a data center in that region.

**Setting /Changing Storage Class:**

1. By default all new objects uploaded to S3 are set to the standard storage class.
2. If requires new objects to have a different storage class, then need to set the proper settings prior to or during the upload process.
3. We can manually switch the objects storage class amongst then at any time by changing the storage class in the objects properties.
   * + Standard
     + RRS
     + S3 –IA
4. To move an object to the glacier storage class –We need to use object lifecycles. The change to glacier may take 1-2 days to take effect.

What are S3 permissions?

S3 permissions are what allow you to have granular control over who can view access and use specific buckets and objects.

1. Permissions functionality can be found on the bucket and object level.
2. On the bucket level you can control for each bucket individually
   * + List
     + Upload/ Delete
     + View Permissions
     + Edit Permissions
3. On object level you can control for each object individually
   * + Open/ Download
     + View Permissions
     + Edit Permissions

**Elastic Compute Cloud (EC2)**

Ec2 provides scalable computing capacity in the AWS cloud. Using EC2 eliminates your need to invites in hardware up front, so you can develop & deploy applications faster. We use Ec2 to launch as many or as few virtual servers as we need, configure security and networking and manage storage. Ec2 enables to scale up or down to handle changes in requirements or spikes in popularity reducing your need to forecast traffic.

**EC2 Basics:**

1. **On Demand:** Allows to choose any instance type we like and provision / terminate it at any time on demand.

It is the most expensive purchasing option.

Is the most Flexible purchasing option

We are only charged when the instance is running and billed by the hour.

1. **Reserved**: Allows to purchase an instance for a set time period of one / three years

It allows for a significant price discount over using on demand.

We can select to pay upfront, partial upfront, no upfront.

Once we buy a reserved instance, you own it for the selected time period and are responsible for the entire price- regardless of how often use it.

1. **Spot**: Is a way for us to bid on an instance type and only pay for and use that instance when the spot price is equal to or below our bid price.

This option allows amazon to sell the use of unused instances for short amounts of time at a substantial discount.

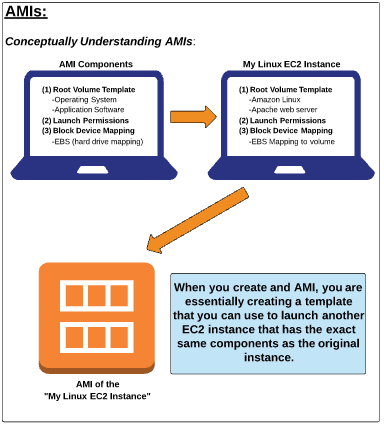
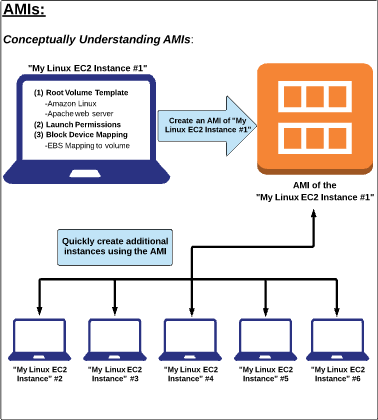
Spot prices fluctuate based on supply and demand in the spot market place.

We are charged by hour

When we have active bid an instance is provisioned for you when the spot price is equal or less than our bid price.

**Amazon Machine Image (AMI):**

AMI provides the information required to launch an instance which is a virtual server in the cloud. We specify an AMI when we launch instances and we can launch as many instances from AMI as we need.

**Selecting an AMI:**

AMI comes in 3 main categories:

1. Community AMI’s :

* Free to use
* Generally with these AMIS we are just selecting the OS we want

1. AWS market Place AMI:

* Pay to use
* Generally comes packaged with additional, licensed software’s.

1. My AMI:

* AMIs that we create our self.

What is an Instance Type?

When we launch an instance type that we specify determines the hardware of the host computer used to our instance. Each instance type offers different compute, memory and storage capabilities and are grouped in instance families based on these capabilities select an instance type on the requirements of the application.

Instance Types Components:

1. **Family**: Away of categorizing instance types based on what they are required to do.
2. **Type**: Subcategory for each family type.
3. **VCPUs**: The number of virtual CPUs the instance uses.
4. **Memory**: The amount of RAM the instance type uses
5. **Instance Storage**: The local instance storage volume hard rive
6. **EBS-Optimized Available**: Indicates in EBS –Optimized is an option for the instance type.
7. **Network Performance**: Rating based on its data transfer rate.

Creating Ec2 instance / auto scaling group configuration.

#/bin/bash

Yum update –y

Yum install –y httpd

Services httpd start

**TYPES OF INSTANCES**

**Two types of virtualization types**

1. HVM ; hard virtual machine are latest Amazon uses it

HVM still uses PV for network & storage drivers which gives it better performance

Where we have capable to do ext. on HVM by using PV it’s recommended.

1. PV: paravirtual: these are faster than HVM but no longer the case.

**Instance Types – General Purpose**

**T2 instances**: New amazon account members will get free T2 instances for limited time to usage

* Intended for workloads that do not the full CPU often or Consistently
* Provide **Burstable** Performance gives base line performance. ex: 20% cpu core uses for web servers with less traffic.
* EBS only storage

**M3 Instances**

* Provide a balance of compute , memory, and network resources
* SSD storage (Instance store): once u terminate the instance we will lose the data too immediately.

**M4 Instances**: Currently in usage for better & fast performance

* Provide a balance of compute, memory, and network resources
* Support enhanced networking
* EBS –optimized by default for better performance
* EBS only no SSD
* Lower CPU utilization
* PPS packets per second for boost performance

**Instance Types – Compute Optimized**

* Lowest price/ compute performance in EC2 also recommended for high traffic, video & host of other use case.

**C3 Instance**

* SSD backed instance storage
* Support for Enhanced networking clustering

**C4 Instance**

* Latest generation of compute-optimized instances
* Highest performing processors (optimized specifically for EC2)
* Support for enhanced networking & clustering
* EBS-Optimized

**Instance Types – Memory Optimized**

* Lowest price per amount (GB of RAM) and memory performance

**R3 Instances**

* **S**SD backed instance storage
* High memory capacity
* Support for enhanced networking

Ex: lot of performances for databases, memory & big data in Linux, spark use cases.

**Instance Types – GPU**

* Graphics and general purpose GPU Compute

**G2 Instances**

* High performance processors
* High performance NVIDIA GPU’s
* Onboard hardware video encoder
* Low-latency frame capture and encoding, enabling interactive streaming
* Useful for GPU compute workloads, machine learnings, video encoding, 3D application streaming etc.

**Instance Types – GPU**

* Very fast SSD backed instance storage optimized for high random I/O performance and high IOPS 9input / output operations per second)

**I2 Instances**

* High I/O performance
* High performance processors
* SSD
* Supports Trim
* Supports enhanced networking

Instances Status Checks

**System Status Check**

We need to wait until AWS fix it or we can do it by our self with below steps:

**Issues:**

* Loss of Network Connectivity & System Power
* Software & Hardware issues on the physical host

**Sol:**

* Stop & Start Instances
* Terminate and re-launch the instances
* Contact AWS

Instance Status Check

**Issues:**

* Failed system status check
* Incorrect networking or startup configuration
* Exhausted memory
* Corrupted file system
* Incompatible kernel

**Sol:**

* Start & stop instances
* Terminate and re launch instances with more memory a different kernel, or different networking configuration.

**Cloud Watch**

We can setup thresholds to alter before something happen. We have diff services on AWS with WC.

**Metrics:**

Lot of default metrics available based on service. Which are not available we can write scripts for example MEMORY status / utilization etc..

* Billing Metrics
* EC2 Metrics > instance id, Instance name, Metric name
* S3 Metrics > Bucket name, Storage name, Metric name
* EBS metrics
* Events metrics

**Alarm States:**

**Ok** State means the metrics that we set up with in the defined threshold

**Alarm** State means outside of threshold state

**Insufficient** State means alarm just started or the metrics is not available, possible not insufficient data

**Terraform and Ansible for AWS**

**Initial set up**

* Create a WordPress environment an AWS using Terraform and Ansible.
* Components are used is VPC, S3, RDS, Load config/ Auto Scaling, ELB, Route53.

1. **Idempotency**: No matter how many times runs it won’t cause duplicates or other resources.

Ex: if Ec2 / S3 deleted or get terminated accidentally by running this scripts it will create without effecting other running resources. It may requires some application config’s.

1. **Immutability**: We no need to fix any infrastructure features non-working / broken

Ex: dev server ec2 has AWS issues we can terminate and recreate the ec2 without outage in normal time as usually.

1. **Scalability**: our environments can scalable as many times.

**Logon to AWS console**

* Create User
* Assign User to existing / create group with full admin permissions policy.

**Install terraform**

* Open Linux server : check for python ( python –version) / install it (yum –y install python)
* Download pip and install (curl -0 <https://bootstrap.pypa.io/get-pip.py>)
* Python get-pip.py
* Pip installawscli
* awsconfigure

(Aws Access Key ID)

(AWS secret Access KEY ID)

* Region name

**Install Ansible**

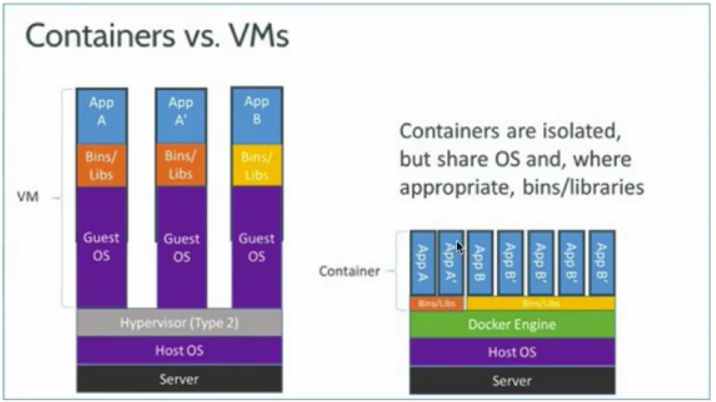
* Open Linux server
* Yum –y install ansible
* Aws route53 create-reusable-delegation-set --caller-reference 12345 (copy the output & save it).
* Curl canhazip.com (to get ip address).

**Docker**

It is a tool that packages up an application and all its dependencies in a virtual container so that it can be run any Linux system or distribution.

**Container**: it is an entirely isolated set of packages, libraries and/or applications that are completely independent from its surroundings.

**Architecture**



**Hasn’t this already been done?**

1. Containers are not a new concept in technology it just appears that Docker has captured the buzz (right place, right time). Lot of companies and projects have been working on the concept of application virtualization for some time

-Example:

* + FreeBSD –Jails
  + Sun (and now oracle) Solaris –Zones
  + Google – Imctfy(let me contain for you)
  + OpenVZ

**Use cases of Docker:**

1. **Configuration simplification**: Configuring pre role docker instances. Can easily distribute small “tar” file that contains the complex config for application in large organization and can deploy the file time after time without any complications.
2. **Enhance Developer Productivity**: It gives a developer to spin off and deploy the changes inside of a container and see its behavior and destroy the unwanted.
3. **Application Isolation:** If an application goes down it does not necessary to take down the entire environment it’s just docker instances takes down no effect for environment.
4. **Rapid Deployment**
5. **Build Management:**

**Docker Hub?**

The Docker Hub is a public registry/ repository that is maintained by Docker Inc. containing a large number of images that we can download and use to build containers

* <http://hub.docker.com>

**Installation & Configuration: docker --version**

* Docker requires 64bit OS
* Distribution & type doesn’t matter until its run on 3.10 kernel & above.
* Uname –r ( to check kernel version)
* Create a Docker repository
* cd /etc/yum.repos.d
* vim.docker.repo

[dockerrepo]

name=Docker Repository

baseurl=https://yum.dockerproject.org/repo/main/centos/7/

enabled=1

gpgcheck=1

gpgkey=https://yum.dockerproject.org/gpg

:wq!

* Yum update
* Yum install –y docker–engine (to available docker engine)

sudo systemctl enable docker

sudo systemctl start docker

sudo systemctl satus docker

* Docker creates some default images at /var/run

To run docker need root/docker user

Sudo usermod “user id” –G docker ( to add our self as user)

* docker run hello-world (to check installation is working correctly)
* path for container: /var/lib/docker/container
* docker ps (to see running container)
* docker ps –a (to see past running container)

**Creating Image**

* Path: /var/lib/docker/images/devicemapper/content/sha256 ( command to see : docker images)
* Logon to docker hub
* Select Ubuntu/ any app to pull image on to server
* Example : docker pull ubuntu (get latest version)

docker pull ubuntu:xenial (for specific version)

* docker run -i -t ubuntu:xenial /bin/bash (it will start / run & attach to container as root)
* docker run –I –t –d /-itd ubuntu:xenial /bin/bash (to run containers on back ground)
* docker images (copy the image name “musing\_bose”)
* docker restart musing\_bose
* docker ps
* docker inspect musing\_bose (to see details of container)
* docker inspect musing\_bose | grep IP (to see all ip info)
* docker stop ubuntu:xenial

**Docker images search with command line without web browser**

* docker image ubuntu
* docker image ruby
* docker search training/sinatra

**Docker Package a Customized Container**

* Go to container
* apt-get
* apt-get install telnet openssh-server
* adduser test
* exit
* docker commit -m "already Installed ssh and created test user" -a "Spartan" oldconatiner name spar/new container name

clear

* mkdir build
* vi dockerfile
* FROM ubuntu:xaniel
* MAINTAINER Spartans
* RUN apt-get update
* RUN apt-get install telnet openssh-server

:wq!

* Command to build: docker build -t="spar/tel" .

**Running Container Commands with Docker & Port Redirects**

* docker exec container ame /bin/cat /etc/profile - we can run command wen container running
* docker run -d container name /bin/bash -s "while true;do echo Hello;sleep 1;done"
* docker logs container name | wc -l
* docker stop cointainer namer -commands do run
* –I : interactive
* –p port
* –t tty
* Docker –d –p 8080:80 ubuntu:latest – redirect from local port to container port

**Docker File Directives : User and Run**

* Create user to use Docker Image
* Mkdir build
* Vi dockerfile
* # this is docker user with non-privileges
* FROM ubuntu:Latest
* MAINTAINER [spartansk05@gmail.com](mailto:spartansk05@gmail.com)
* RUN usesradd –ms /bin/bash phani
* USER Phani

:wq!

Command to run file: docker build –t Ubuntu/latest1:v1 .

Docker run –it Ubuntu/latest1:v1 /bin/bash

docker exce –u 0 –it aliasname /bin/bash – to connect as root user for container

Ps aux – some containers running

**Naming Containers**

docker run -itd --name cox1 spar/tel /bin/bash – it create new container with new name

docker rename oldcontainername newcontainerneme (coxtest1) – while running container

We cannot change container ID’s any time

We can change container names any number of times

Docker Events

-When we start /stop containers certain events are created

* docker events
* docker events --since '1h' :shows the last 1hour events that happened with date, name details
* docker attach
* docker kill idname

**Filters:**

* docker events --filter event=attach / image/ label/ type/ network
* docker events --filter event=attach – can create alert when anyone attach to specific container
* docker events --filter event=attach –filter event=attach –filter event=die --filter event=stop

**Pushing to Docker Hub**

* Create repository on Docker Hub
* Visibility: private
* On server - docker login
* Logon details of docker hub
* Docker tag spar/tel:v1 hub username
* Docker login –username= xxxx --email=xxxxxx
* Password: xxxxx
* Docker push username/spar/tel
* Docker logout

**Using Java & Tomcat**

1. Download java & tomcat & sample war file
2. Docker pull Ubuntu
3. docker run –itd docker.io/Ubuntu /bin/bash
4. cd downloads

* cp all downloaded file here
* docker run -it -–name jdk –v / docker/downloads:/root/downloads ubuntu:latest /bin/bash
* yum update
* yum install wget sudo which
* tar –xvf jre
* cp / mv jre1 > /opt/java/

**Git & Git Lab**

**Installation:**

* Yum install git
* git –version
* cd / vi gitconfig
* [user]
* name = Root User
* email = [spartansk05@gmail.com](mailto:spartansk05@gmail.com)
* 3 types of git configurations
* **Local** is for particular repository
* **System** is for system level
* **Global** is for our self
* Git config - -system syatem.name “git repo server 1” (or) vi git config
* [system]
* Name = git repo server 1
* Git config - -global core.editor vim (or) vi git config
* [core]
* Editor = vim
* Git config - -global core.pager ‘more’

**Empty Repositories**

* Cd /opt/ mkdir git
* Cd git
* Git init – (.git)it creates empty repository with sub directories (branches, config, description,onjects, books, refs, read, info.
* Echo “this is my test file” > test.txt
* Git add test.txt
* Git commit test.txt
* Git status

**Our Repositories**

These local initialized repositories

* Cd /opt/ mkdir unit
* Cd unit
* Cp all kind of files
* Git init
* Git add \*
* Git commit -m “these are test file”
* Git rm file name – it deletes file from git copies
* Git commit -m “test delete”

**Git Ignore**

* **Vi .gitignore : the file formats which you don’t want to add to git repositories**

**-.mf**

**-.xml**

**.jpeg**

**Cloning**

* 3 ways to copies of repositories from & to
* cp
* Git clone unit/ repounit
* Remote repository
* Local host ip address
* Ssh-keygen
* Ssh-copy-id user localhost name /ip address
* Ssh-localhost : to check exchanged keysn wont ask password
* Git clone user@ipaddress:/opt/unit new filename

**Tagging**

* Is created to point to specific commit doesn’t change rest of the branch moves on
* Two types of tags
* Branches & tags are similar but there is specific difference between them.
* ~~Go to cd /opt/unit~~
* ~~Git tag tag1~~
* ~~Echo “tag1.Post” > postag.txt~~
* ~~Git add postag.txt~~
* ~~Git commit -m~~

**Branches**

* we use for multiple project development in single product
* when we do git status it shows/ connects to branch master default
* to check any branches existing : git branch -a
* to create branch development/ qa : git checkout -b development
* TO switch back to master branch : git checkout master
* Changes made on master won’t show on branch same as branch to master

**Merging**

* Git merge development - -no-ff
* It will merge dev / any braches to master

### **Logging and Repository Auditing**

* Git log
* Git log -p -2/4/6 =(p shown the diff between commit; 2= shows last 2)
* Git log - -pretty=online = shows reference for each commit
* Git log - -preety=format:”#h: #an, #ae, #cn, #cd, #s”

**Git -Hub**

1. Free public resource account
2. Co-operative features
3. Public & Private Repositories
4. Git hub enterprise – it’s an image, we can host on VM ware - more expensive

## **Secure Communication**

* Adding new repo
* Name
* Description
* Public/ private account
* Go to setting
* Https/SSH -To set up password list exchange with git hub
* Go to server
* Ssh -l user ip address
* Cd /home/user/ mkdir test
* Mv test repo
* Cd repo
* Ssh-keygen – all setting will default
* **Cat ../.ssh/id\_rsa.pub** – copy the key and paste on github.com>settings>ssh key

**Working With repo with git hub**

* Go to github.com > user
* Right side of page click on ssh and copy the text & go to linux server : git clone :paste” – now you can use

**JENKINS**

Installation

**AZURE**

**TEST**

1. You manage a social media website on EC2 instances in an Auto Scaling group. You have configured your Auto Scaling group to deploy one new EC2 instance when CPU utilization is greater than 90% for 3 consecutive periods of 10 minutes. You notice that between 6:00 pm and 10:00 pm every night, you see a gradual increase in traffic to your website. Although Auto Scaling launches several new instances every night, some users complain they are seeing timeouts when trying to load the index page during those hours. What is the least cost-effective way to resolve this problem?

**Incorrect**

**Correct answer**

Increase the minimum number of instances in the AutoScaling group

**Explanation**

Increasing the minimum number of instances in the AutoScaling group will keep more instances running around the clock, thus making it a very inefficient way to manage cost. The other options all increase the AutoScaling group's sensitivity to an increase in load and enable it to respond quicker to increased load by spinning up instances as soon as they become necessary.

**2)** You have an Amazon VPC that has a private subnet and a public subnet in which you have a NAT instance server. You have created a group of EC2 instances that configure themselves at startup by downloading a bootstrapping script from S3 that deploys an application via GIT. Which one of the following setups would give us the highest level of security?

**Incorrect**

**Correct answer**

EC2 instances in our private subnet, no EIPs, route outgoing traffic via the NAT

**Explanation**

EC2 instances in this example do not need to be in the public subnet, because the private subnet has access to resources in the public subnet, and therefore can access the NAT instance. That way, we can make sure those EC2 instances are hidden from public access.

**3)** True or False: Multi-AZ RDS replications use asynchronous data replication.

**Incorrect**

**Correct answer**

False

**Explanation**

Data replication is synchronous

**4)** Your company is setting up an application that is used to share files. Because these files are important to the sales team, the application must be highly available. Which AWS-specific storage option would you set up for low cost, reliability, and security?

**Correct**

**Correct answer**

Use Amazon S3, which can be accessed by end users with signed URLs.

**5)** You maintain an application on AWS to provide development and test platforms for your developers. Currently, both environments consist of an m1.small EC2 instance. Your developers notice performance degradation as they increase network load in the test environment. How would you mitigate these performance issues in the test environment?

**Incorrect**

**Correct answer**

Upgrade the m1.small to a larger instance type