**AWS Job Oriented**

Regions : This is a geographical area where cluster of data centers are located. Each Region comprises of one or more Availability Zones.

Availability Zones : They are isolated data-centers located in a region, usually there are multiple AZ's in a region are connected with low latency bandwidth and do not share any physical equipment with each other, thus making them resilient to any infrastructure failures.

Selecting a region for a specific requirement:

1. Check the latency (<https://cloudping.info/>)
2. Check for services available in that region
3. Pricing varies from region to region([https://calculator.aws/#/](https://calculator.aws/" \l "/))
4. SLA (Service Level Agreement)
5. Compliance – weather you are complaint with the regulations of that region

**Identity and Access Management:**

This Service is a **Global Service** that provides centralized control over the authentication and authorization managment to AWS Account.

Free service

Users

Attach to a group

copy permissions

Attach policies directly

tags

User Groups

Roles

AWS Service

Allows AWS service to communicate with another service in aws.

Attach a policy to the role to define the level of authorization

AWS Account

Allows aws cross account service interaction or AWS to third part account interaction.

Policies

Defines the authorization roles that are attached to a user or role. By default all the permissions are denied.

AWS Managed Policy

Customer Managed Policy

AWS Managed Job-Function – Curated regularly performed jobs.

Policy format – Json

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "Statement1",

"Effect": "Allow",

"Action": [],

"Resource": []

}

]

}

It better to use aws default polcies as if there are any change to the service made then custom managed polices have to be updated manually.

MFA(Multi-Factor Authentication):

Enable MFA for all the users

mode of authentication:

Authenticator App, Security key(Yubi Key), hardware totp token (time based one-time token)

A user can have multiple MFA authentications assigned

Create a group with admin privilege(No priviledge to view the billing information) and add a user to the group  
  
There are 3 ways to access aws

UI

Sdk

CLI

Accessing AWS from CLI

Create Access Key for user and download the details then install aws cli

Access from terminal using the aws configure and enter the relevant details

Access the aws account from cli using   
 aws service action-verb or aws iam help

Access details are stored in local in .aws folder, Its a good practice to delete the .aws folder

**EC2, Load Balancer,AutoScaling Route 53:**

Integration between EC2 – Security Groups - VPC – Subnets– ALB – Target Groups – Launch Templates - ASG – Route 53 – Cloud Watch

EC2:

This is a compute instance which gives the user control over operating system, Processor, Shape, Storage and any user data that should be pre-loaded during instance creation. This is charged based on the instance shape, size and storage.

Security Groups:

This allows to place restrictions on inbound and outbound network traffic.

VPC:

Virtual Private Cloud creates a isolated network space inside AWS which hosts all the regional AWS services where firewall rules can be defined on what kind of traffic can enter and exits from the VPC

Subnets:

This created a seggregation of IP Address range based VPC, This allows to create subnets which

SelectASG – Free Of Charge

EC2 – Charged based on the shape, storage

Route 53 – Charged – Global Service

Elastic Bean Stalk

Allows a user to create a environment in an automated way based on the values defined for Code configuration, EC2, LB, ASG, VPC, Subnets, Target Groups, Security Groups, IAM Roles to communicate with AWS services, Storage, Databases, Cloud Watch Alarms

**Networking and VPC**

VPC:

Virtual Private Cloud creates a isolated network space inside AWS which hosts all the regional AWS services where firewall rules can be configured on what kind of traffic can enter and exits from the VPC

When a VPC is created, by default a route table, NACL is created.

AWS recommends that you specify a CIDR block (of /16 or smaller) from the private IPv4 address ranges as specified in [RFC 1918](http://www.faqs.org/rfcs/rfc1918.html). Here are the address blocks you can get started with:

* 10.0.0.0–10.255.255.255(10/8 prefix)
* 172.16.0.0–172.31.255.255(172.16/12 prefix)
* 192.168.0.0–192.168.255.255

Subnet:

This is to create smaller networks based on the IP range of the VPC. When subnetting the network we always have 5 reserved IP address

* + - 1. 10.0.0.0 - Network Address
      2. 10.0.0.1 - Reserved for AWS VPC Router
      3. 10.0.0.2 - Reserved for DNS Server
      4. 10.0.0.3 - Reserved for AWS for future purpose
      5. 10.0.0.255 - Reserved for Broadcast Address

In order to auto-assign public IP to the instances created in a subnet, we have to enable auto-assign public IP in the subnet. This allows the instance to access and be accessed from the internet.

Internet Gateway (IGW):

By default, all network traffic is denied in a VPC, in order to access the internet, we need to create a IGW and attach to the VPC.

Only one IGW can be attached to a single VPC

Route table:

Route Table is set of rules where network traffic is determined. It is associated with gateways and the destination Ip Address and the subnets associations

NAT Gateway:

This is a charged based on the gateways, This allows only egress traffic from a VPC to internet. This can be configured on a private subnet as it allows the services under this private subnet to access the internet but not the other way around.

Security Group:

Security group is configured at instance level which allows inbound and outbound traffic from the defined protocol and IP address range.

NACL(Network Access Control List):

NACL acts like a virtual firewall at VPC level that is associated with the subnets. It allows and also denies traffic to/from IP range with the defined protocol.

VPC Peering:

This service allows to connect VPC’s in different regions or accounts.

Conditions for VPC Peering:

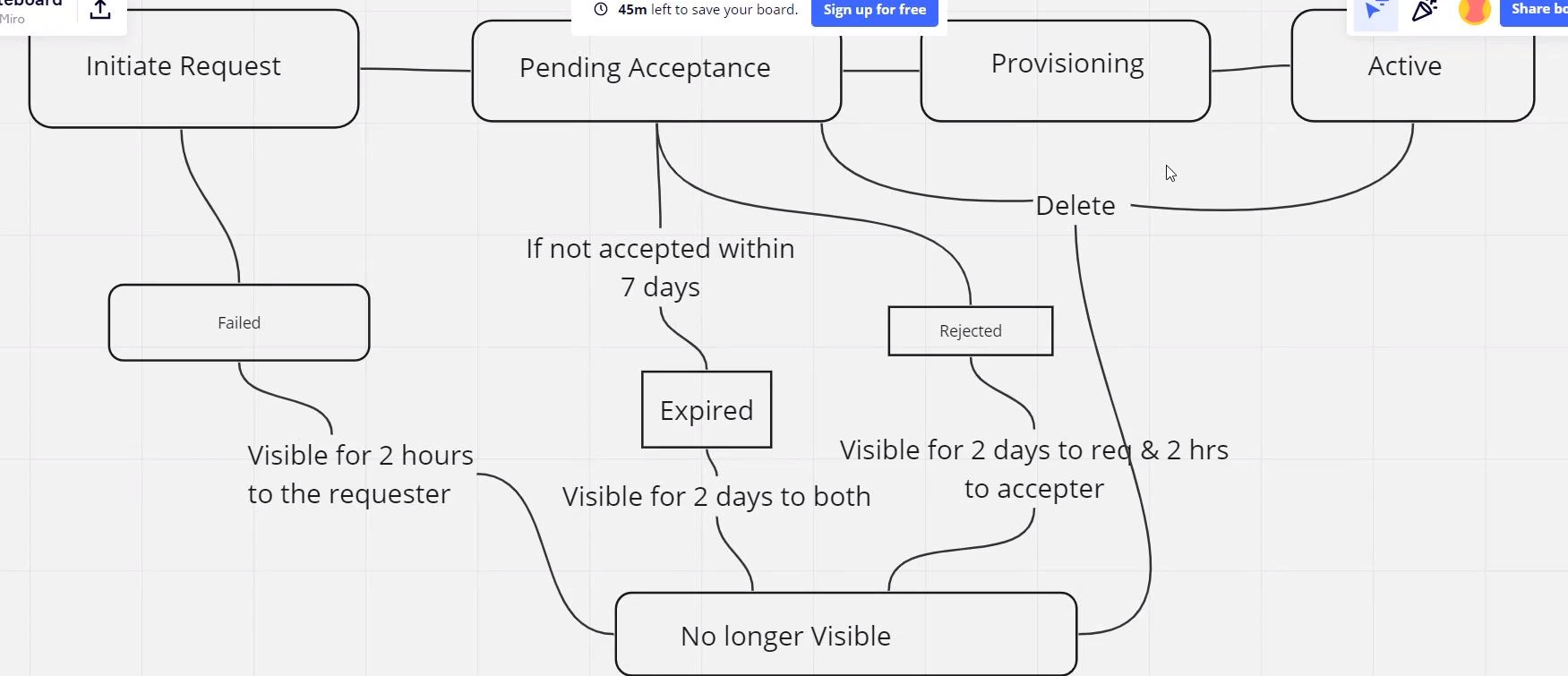
1. The IP Address range should not conflict

In order to establish connection, we have two components Accepter VPC and Requester VPC.

1. Requester VPC sends a request to Accepter VPC.

2. Accepter VPC should approve the request within 7days

3. Then Requester VPC should update the Route Table and NSG in case any specific IP from the IP range of the Accepter VPC is blocked



VPC EndPoint:

Transit Gateway:

Direct Connect Gateway:

AWS Job Oriented Programs Tasks:

1. Creating user and attaching it to user groups.

2. Creating a Policy and attaching it to user groups

3. Creating a Role and attaching a Policy to it

4. Creating Custom Policies

5. Configuring AWS CLI using access Keys

6. Creating MFA for users

7. Creating EC2 Instances, AMI, Launch Templates

8. Creating AutoScaling Groups

9. Create Load Balancer and Target Groups

10. Create Route 53 domain and assign records

11. Integrating Route53, Load Balancer, Target Groups, ASG, Security Groups, Launch Templates, Instances

12. Creating VPC, Subnets(Public and Private), Route Tables, IGW, NACL

13. Understanding CIDR Ranges.

14. Creating a BASTION host to access private instances

15. Configuring NAT Gateway inorder to allow traffic through Private Instance

16. Configure VPC Peering

17. Configure VPC Endpoint

18. Configure a Transit Gateway for VPC

19. UnderStand AWS Direct Connect process

20. Know the VPC Quotas

**DOCKER & KUBERNETES in AWS**

**Understanding docker containers:**

Containers --> created user Images

Images --> created using docker file

docker file --> docker file created using docker commands

Follows Layered architecture

Each Instruction on a docker file forms a layer, this layered architecture allows docker to share the image layer when another docker file which contains similar command is executed.

Images created locally can be pushed onto a docker registry which can be shared with others, Similarly images can be pulled into local from the docker registries

Some basic commands:

pull image : docker pull <image>

create container : docker run --name=<containername> <image>:<tag>

running in detach mode : docker run -d --name=<containername> <image>:<tag>

expose port from container to be accessed publically using host machine IP and port

docker run -p <hostport>:<docker port>--name=<containername> <image>:<tag>

check running containers : docker ps

check all containers(running and exited) : docker ps -a

persist changes using volumes :

docker run –mount source=<vol1>,target=<cntpath>--name=<containername> <image>:<tag>

persist changes using bind mount :

docker run –mount type=bind,source=<srcpath>,target=<cntpath>--name=<cntname> <image>:<tag>

check docker images : docker images

delete images : docker rmi <image id>

delete containers : docker stop <cnt id>, docker rm <cnt id>

Build an image from docker file:

docker build . -f <docker file> -t <imagename>:<tag>

Pushing to docker registry:

docker tag <imagename>:<tag> <user/image>:<tag>

docker push <user/image>:<tag>

**ECS (Elastic Container Service):**

Amazon ECS makes it easy to deploy, manage, and scale Docker containers running applications, services, and batch processes.

This orchestration gives us high availability, Scalability, Self Healing, AutoScaling, Maintaining the desired state, Load Balancing, Monitoring, Security

**ECR:**

**Kubernetes**

Kubernetes - This is container orchestration tool that manages multiple containers. It allows users to create clusters that contains master and worker nodes.

Ideally in a cluster setup, we have a single master node and multiple worker nodes. In some cases in order to provision high availability we can also have multiple master nodes.

Each worker node comprises of below components, This is also called a data plane node

Container engine : Creates containers based on the image provided

Kube Proxy : Allows the user to interact with the applications deployed on pods

Cadvisor : Monitors the stats of the containers

Pod : This is a kubernetes object that encapsulates the container

Kubelet : It runs on port 10250, It is a agent that runs on worker node that communicates with API servers on the requests and stats of the worker node components

Master Node : As the name indicates this is the control center for a kubernetes cluster, This is also known as control plane and contains the key components which maintain the desired state of the cluster.

The key components are

Api Server :

This runs on port 6443 and is the communication components to interact with the kubernetes resources on worker nodes.

Customer Managed Controller/Cloud Managed Controller:

This runs on port 10257 and manages the desired state of the cluster, if all the kubernetes objects in the master and worker nodes are upto the configuration defined.

Cloud Controller – This allows the kubernetes components to interact with the cloud native resources.

Scheduler:

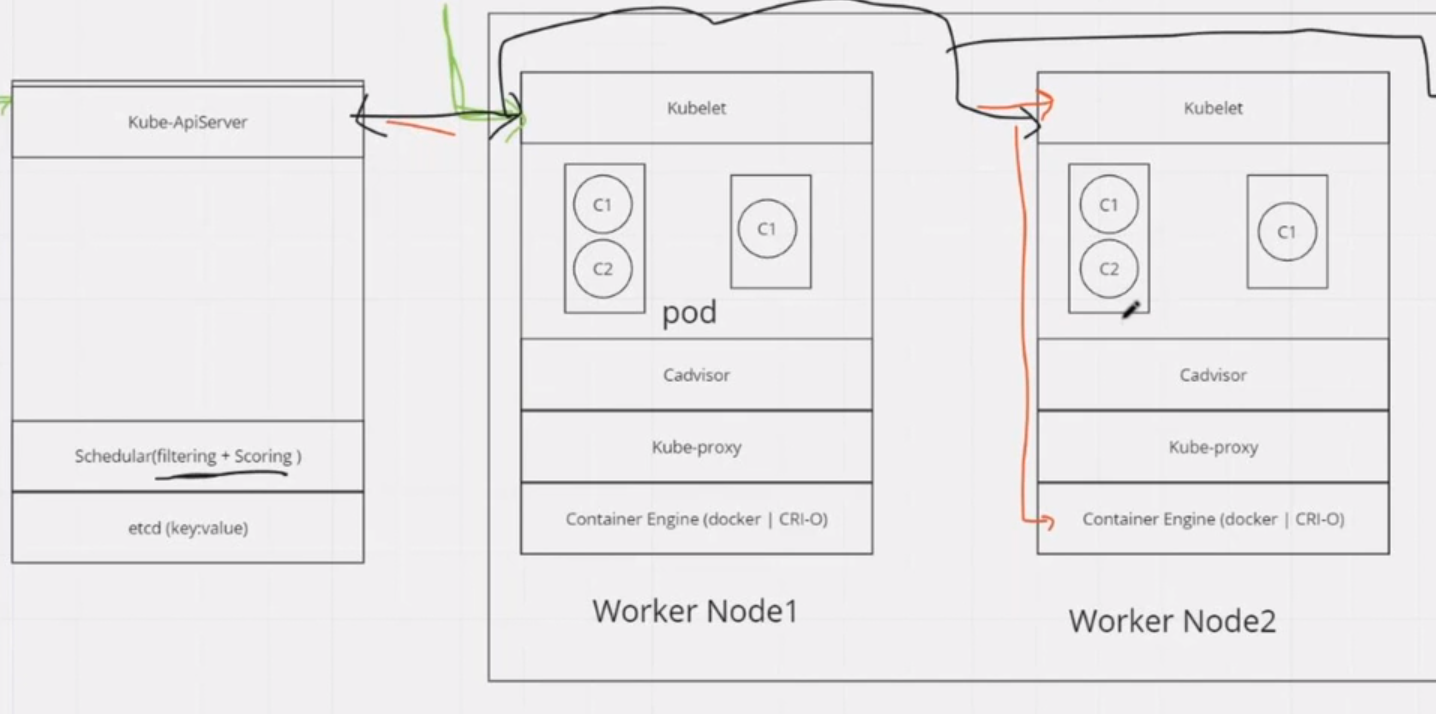
This runs on port 10259. It manages the location where the pods have to deployed on the worker nodes. This node selection is done through filtering, scoring

Etcd :

This runs on 2379 and if there are multiple master nodes it also utilizes port 2380. It is key/value pair that contains the configuration of the cluster.

Kubernetes Diagram:

Master(Control plane) and worker node(Data plane)



**Features**:

1. High Availability

2. Self Healing : This feature restores the desired state of the cluster.

Eg: If a pod goes down or is in the process of going down in a node, cadvisor detects this and informs the kubelet which inturn communicates with the API server of master node. API server conveys this information to the controller which checks for the configuration/manifest file, based on it, it reciprocates in the same way to the create a new pod.

3. Auto Scaling

4. Storage Allocation

**EKS:**

EKS is Elastic Kubernetes Service provided by aws which manages the Control plane and Data Plane.

Considerations before creating kubernetes cluster

1. How many virtual machine are required

a. Number of applications to be deployed on the cluster

b. Storage (Capacity/iops)

c. Computer (CPU/RAM)

d. Network Bandwidth

e. No. Of users accessing the application

f. Number of pods to be deployed

2. Load Balancers

3. Security

4. Service Mesh

5. Data Migration Plan from on-prem to cloud

6. Etcd back plans

7. Multi region deployment of worker nodes

8. exit plan – If you want to switch between cloud environments

9. Monitoring

Ideally we need to use terraform/Cloud formation tool to deploy a cluster on EKS, This way we would be able to replicate the cluster configuration in multiple aws accounts using the template

**Cluster Creation EKS with worker nodes:**

1. Create the EKS cluster from AWS Console or terraform

Role : EKSClusterRole

2. Install Kubectl in AWS CLI from you

3. Connect to the EKS Cluster from command line

aws eks –region <region> update-kubeconfig –name <cluster name>

4. Create node groups

Configure Node Group:

Role Creation : EC2 to EKS

Policy : AmazonEC2ContainerRegistryReadOnly, Amazon EKS Cluster Policy,Amazon EKS Service Policy,Amazon EKS CNI Policy, AmazonEKSWorkerNodePolicy,

EC2 Launch Template – Create EC2 Instances using a blueprint

Add a label – For Filtering the nodes

Taints – Adds restrictions on the nodes which can tolerate it with taint effect

Compute Details:

Compute Details on the node

Number of nodes (desired, maximum, minimum)

Networking:

Subnets

EC2 Keypair(Optional)

Security Groups(Optional)

**EKS with fargate:**

Fargate is a serverless container platform where the server maintenance is taken care by AWS and user only has to create the containers to deploy applications.

**Creation:**

1. Create Role – EKSFargatePod (AWS Documentation)

2. Create Fargate Profile

Name

Subnets : Private Subnets

Namespace

**ECR:**

Elastic Container Registry is AWS private registry for storing images.

**HPA:**

Horizontal Pod Autoscaler, Based on the metrics defined HPA in the manifest file, Number of pods are increased or decreased, These metrics are retrieved from the metrics server which can be deployed using a YAML file from github

**Trouble Shooting:**

1. Pod not getting created

ImagePullBackOff - Check the pod using kubectl describe pod <podname>

1. The pod image used might be wrong

2. The registry is not authenticated

2. Unable to see the nodes after creation on node groups or fargate profile

Check the security group inbound rules

3. Unable to create pods in Fargate

ImagePullBackOff -

**AWS docker&kubernetes Job Oriented Programs Tasks:**

1. Create EC2 instance with docker installation

2. Pull Docker image httpd, create httpd container and expose port,

3. Write a Docker file

4. Create EKS Cluster

5. Install Kubectl on aws cli on your terminal

6. Create Node Group

7. Create namespaces

8. Set cluster in kube config

9. Create Deployment, replica set, Pods

10. try kubernetes commands to scale, navigate, taints

11. Create Deployment in namespace

12. Create pod using YAML (resources, namespace, ports)

13. Create Multi container pod (httpd, ubuntu, mysql)

14. Create Deployment using YAML

15. Create a Deployment and expose it using a service node port to be accessed publically

16. Create a Deployment and expose it using a service Load Balancer to be accessed publically

17 Create EKS with Fargate

18. Create ECR –

Push docker image to ECR Public Repository

Push docker image to ECR Private Repository

19. Create HPA with CPU Metrics pod and scale the pods – Map the pod with resource limits

20. Create EKS Cluster using command line

21. Create a Storage Class Mapping to aws gp2, create a PVC and map it to a pod, Create IAM policy to EBS CSI Driver and map it to node group EC2 role

22. Create Ingress Controller and link with the AWS Load Balancers

**AWS Interview Questions:**

1. What are types of virtualization do we have on AWS platform?

There are two types of virtualizations

1. paravirtualization

2. Hardware virtual machine

2. What are the type of root devices?

There are two types are root devices

1. EBS – Elastic Block Storage

2. Instance Store

3. What is the difference between t2.micro and t3.micro?

T2 and T3 are the various family types that aws offers which vary in the size of the machine such as CPU and Memory

4. What are the types of EBS volumes and the use cases?

There 4 types of EBS volumes:

1. GP2/GP3 (General Purpose)

2. Provisioned IOPS (IO1/IO2)

3. SC1 (Cold HDD) – This is a low cost HDD volume which is designed for frequently accessed workloads

4. ST1 (Through put optimized HDD) – This is the cheapest HDD volume available

5. Why do we attach IAM role with EC2 machine while creating it?

IAM role basically allows AWS service to access another AWS service without any intervention of the user, This way if the user want to upload files to S3 using EC2, we can assign a IAM role to EC2 to access S3 directly without the need of configuring the access keys of the user in EC2

6. What is the advantage of using IAM role with EC2 machine?

Ideally when we configure the access key of a certain user in EC2 machine it is stored in a plain text format which can be easily accessed by another other user who can access the EC2 machine thus voilating privacy. In order to mitigate this issue we can assign an IAM role to EC2 to any aws service

7. What are the usages of TAG’s with EC2/AWS resources?

Tags help to identify a the purpose the resource was created, it acts as a metadata which describes the resource, We can any number of tags to a resource.

8. Why do we need a security group?

Security group helps in configuring inbound and outbound connections from selected type and ports for a EC2 instance

9. What are the types of hypervisors in AWS?

There are two types of hypervisors

1. XEN – These hypervisors are applied on bare metal

2. Nitro – There are light weight hypervisors that can handle high I/O speeds

10. How can we recover lost ec2 ssh key?

There is no way to to recover a lost ssh key, we can move the root volume to another instance and recover of there

11. How to check shared AMI’s?

We can check the AMI’s section and search for images which are public, this shows the list of images that are shared by others

12. You have been asked to design a VPC architecture for a 2-tier application. The application needs to be highly available and scalable. How would you design the VPC architecture?

We design a VPC with two subnets, one as public subnet which has the ALB and other as private subnet which has application deployed on it, In order to make it highly available we distribute the subnet across multiple AZ’s and in order to make it scalable we apply Autoscaling groups on the EC2 instances

13. Your organization has a VPC with multiple subnets. You want to restrict outbound internet access for resources in one subnet, but allow outbound internet access for resources in another subnet. How would you achieve this ?

This can done using the route tables, since we have two subnets for one subnet we can remove the outbound route to anywhere through IGW and restrict the access to internet and in the other subnet we can the IGW route to anywhere

14. You have a VPC with a public subnet and a private subnet. Instances in the private subnet need to access the internet for software updates. How would you allow internet access for instances in the private subnets?

We can achieve only outbound internet access in a private subnet by attaching it to a NAT gatway or NAT instance in the route table.

15. You have launched EC2 instances in your VPC and you want them to communicate with each other using private IP addresses. What steps would you take to enable this communication?

Ideally if both the EC2 instances are in the same VPC and under the same subnet they can communicate with each other without any configuration as they come under the same CIDR range, if they are under different subnets in other VPC’s then we have establish a VPC peering connection.

Also we need to ensure that the NSG groups associated with EC2 instance are allowing inbound and outbound connections from the respective subnets