**TOOLS FOR INTERVIEWS**

**AWS SERVICES :**

**S3 :** [**https://www.youtube.com/watch?v=oaZ3R4NCRu8**](https://www.youtube.com/watch?v=oaZ3R4NCRu8)

**ECS :** [**https://www.youtube.com/watch?v=46mFdtpy3NQ&t=609s**](https://www.youtube.com/watch?v=46mFdtpy3NQ&t=609s)

**ECR :** [**https://www.youtube.com/watch?v=Yy9AGt4m0\_I**](https://www.youtube.com/watch?v=Yy9AGt4m0_I)

**IAM :** [**https://www.youtube.com/watch?v=o0p04B7-NFY**](https://www.youtube.com/watch?v=o0p04B7-NFY)

**Sts token & role -** [**https://www.youtube.com/watch?v=dqF4VJCska4**](https://www.youtube.com/watch?v=dqF4VJCska4)

**ASG :** [**https://www.youtube.com/watch?v=4EOaAkY4pNE**](https://www.youtube.com/watch?v=4EOaAkY4pNE)

**Application load balancer -** <https://www.youtube.com/watch?v=JhH7epqrvF0>

**S3**

Amazon S3 is an object storage service that stores data as objects within buckets. An *object* is a file and any metadata that describes the file. A *bucket* is a container for objects. S3 is globally accessible.

A bucket is a container for objects stored in Amazon S3.

Objects are the fundamental entities stored in Amazon S3.

An *object key* (or *key name*) is the unique identifier for an object within a bucket. Every object in a bucket has exactly one key. The combination of a bucket, object key, and optionally, version ID (if S3 Versioning is enabled for the bucket) uniquely identify each object.

aws s3api put-object --bucket cyclones-terraform-state-2 --key env:/fit4/forgerock\_access\_manager\_backup --body /Users/m1055938/Downloads/forgerock\_access\_manager.json

A bucket policy is a resource-based AWS Identity and Access Management (IAM) policy that you can use to grant access permissions to your bucket and the objects in it.

You can use ACLs to grant read and write permissions to authorized users for individual buckets and objects.

**Storage class:**

Amazon s3 standard for frequent data access

Amazon s3 standard for infrequent data access

Amazon Glacier

One zone-IA storage class

Amazon s3 standard reduced redundancy storage

**FEATURES:**

**Life cycle rules:**

We can create life cycle rules to move buckets from one storage class to another based on conditions.

**Bucket Policy:**

Allow or deny access to the s3 resources

**Versioning:**

Goto properties and we have bucket versioning section there click on enable

**ECS**

Amazon Elastic Container Service (Amazon ECS) is a highly scalable, fast container management service that makes it easy to run, stop, and manage containers on a cluster. Your containers are defined in a task definition that you use to run individual tasks or tasks within a service.

for more control over your infrastructure, you can run your tasks and services on a cluster of Amazon EC2 instances that you manage.

Cluster

Task definition

Service

task

**IAM**

Aws identity and access management safeguards access to aws resources and services and to create and manage aws users and groups.

Components:

**IAM users:**

Person or app that interacts with aws

It has a name and credentials.

We assign iam policy to the user to provide access to resources

Root user and admin access are not same – eg root user has option to change the email account, can change the support contract, view billing info and all

**IAM Groups:**

Collection of iam users

Assign policies to groups to give access to many users for specific resources at a time

**IAM ROLES:**

It is like an iam user but this is allowed for a time period. There are no longterm credentials attached with role.

We do not assign role to a user or service. User or service assumes the role

they can get permission at run time by assuming a role

**IAM Policies:**

Policies are created and attached to users,groups,roles.

This is a json file where we mention access permissions for resources or aws services

Whether the action is approved or rejected depends on these policies

How to force a user to do MFA to access any resource?

Create a policy and assign it to the user

{

"Sid": "BlockMostAccessUnlessSignedInWithMFA",

"Effect": "Deny",

"NotAction": [

"iam:CreateVirtualMFADevice",

"iam:DeleteVirtualMFADevice",

"iam:ListVirtualMFADevices",

"iam:EnableMFADevice",

"iam:ResyncMFADevice",

"iam:ListAccountAliases",

"iam:ListUsers",

"iam:ListSSHPublicKeys",

"iam:ListAccessKeys",

"iam:ListServiceSpecificCredentials",

"iam:ListMFADevices",

"iam:GetAccountSummary",

"sts:GetSessionToken"

],

"Resource": "\*",

"Condition": {

"Bool": {

"aws:MultiFactorAuthPresent": "false",

"aws:ViaAWSService": "false"

}

}

}

Assume role:

aws configure

AWS Access Key ID [None]: ExampleAccessKeyID1

AWS Secret Access Key [None]: ExampleSecretKey1

Default region name [None]: eu-west-1

Default output format [None]: json

Then,

aws sts assume-role --role-arn "arn:aws:iam::123456789012:role/example-role" --role-session-name AWSCLI-Session

with this u will get session token, access key , secret key temporarily for specified amount of time

**LOADBALANCERS:**

Types:

Classic loadbalancer : basic load balancer for ec2 it basically routes based on ip addresses and tcp port.

Application loadbalancer : advances loadbalancing routes http,https traffic based on application paths.

Network loadbalancer : layer4 traffic, routes tcp traffic