AWS CREDENTIALS :

User name : anamika

Password : cpv}7j{YNnNM

Access key ID : AKIARTVSBBSDA4ZJNUX5

Secret access key : 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

Console login link : <https://110968769670.signin.aws.amazon.com/console>

MAJOR SERVICES :

* IAM
* S3
* EC2
* RDS
* CLOUDFRONT
* VPC
* SNS
* ELASTIC BEANSTALK
* LAMBDA
* AUTOSCALING
* ELASTICACHE
* CLOUDWATCH
* SQS
* SNS
* ROUTE53
* LIGHTSAIL
* ELB
* DYNAMODB

AWS stands for Amazon Web Services.provide different IT resources available on demand. It provides different services such as :

* infrastructure as a service (IaaS)
* platform as a service (PaaS)
* packaged software as a service (SaaS).

Features:

* + more time for core business tasks due to the instant availability of new features and services in AWS.
* no upfront investment, long-term commitment, and minimum expense when compared to traditional IT infrastructure
* autoscaling and elastic load balancing techniques are automatically scaled up or down, when demand increases or decreases respectively.
* end-to-end security and privacy to customers
* isolation of their operations.

Infrastructure :

* Global infrastructure is a region around the world in which AWS is based
* AWS is available in 24 regions, and 72 availability zones in December 2018
* components that make up the AWS infrastructure:

Availability Zones

Region

Edge locations

Regional Edge Caches

* zone :

facility that can be somewhere in a country or city. Inside this facility, i.e., Data Centre, we can have multiple servers, switches, load balancing, firewalls. The things which interact with the cloud sits inside the data centers. zone can be a several data centers, but if they are close together, they are counted as 1 availability zone.

* Region :

collection of data centers which are completely isolated from other regions.consists of more than two availability zones connected to each other through links.

* Edge location :

endpoints for AWS used for caching content. CloudFront, Amazon's Content Delivery Network (CDN).Edge location is not a region but a small location that AWS have.mainly located in most of the major cities to distribute the content to end users with reduced latency.

* Region Edge cache :

new type of edge location lies between CloudFront Origin servers and the edge locations.regional edge cache has a large cache than an individual edge location.Data is removed from the cache at the edge location while the data is retained at the Regional Edge Caches.

------------------------------------------------------------------------------------------------------------------------

Aws iam :Identity Access Management.

* manage users and their level of access to the aws console. set users, permissions and roles.grant access to the different parts of the aws platform.
* web service that enables Amazon Web Services (AWS) customers to manage users and user permissions in AWS.
* With IAM, Organizations can centrally manage users, security credentials such as access keys, and permissions that control which AWS resources users can access.
* create multiple users, each with its own security credentials, controlled and billed to a single aws account. IAM allows the user to do only what they need to do as a part of the user's job.
* control creation, rotation, and cancellation of each user's security credentials. what data in the aws system users can access and how they can access.
* Users can share the resources
* set a permission that user can use a particular service but not other services.
* multifactor authentication as we need to enter the username, password, and security check code to log in to the AWS Management Console.
* Users can be restricted to the AWS access based on their job duties,
* ensures that the users can access the AWS resources within the organization's corporate network.
* integrated with many different aws services.
* high availability by replicating the data across multiple servers within the Amazon's data center

IAM IDENTITIES :

provide authentication for people and processes in your aws account.

[TYPES: USERS , GROUPS , ROLES](https://www.javatpoint.com/aws-iam-users)

set of permissions that grant access to actions and resources in AWS.permissions are attached to the role, not to an IAM User or a group

associated with a single person; it can be used by anyone who needs it.

temporarily security credentials are created and provided to the user.

TERMINOLOGIES :

DELEGATION : granting the permissions (delegation) , access to the AWS resources that you control . trust between a trusted account (an account that owns the resource) and a trusting account (an account that contains the users that need to access the resources).

ACCOUNT TYPES :

* Same account
* Two different accounts under the same organization control
* Two different accounts owned by different organizations.

IT HAS 2 POLICIES :

PERMISSION POLICY : grants the user with a role the needed permissions . Document written in JSON format to define the actions and resources that the role can use.

TRUST POLICY : specifies which trusted account members can use the role.document was written in JSON format to define who is allowed to use the role.

PERMISSION BOUNDARY : limit the maximum permissions that the role can have and applied to IAM User or IAM role but cannot be applied to the service-linked role.

PRINCIPAL : AWS root account user, an IAM User, or a role.permissions that can be granted in one of the two ways:

* Attach a permission policy to a role.
* resource-based policies, allow you to attach the policy directly, ex: Amazon S3 buckets, Amazon SNS, Amazon SQS Queues.

APPLYING ROLES :

* Iam roles for service

select service -select/create policy – create role

through cli :

Create a role: aws iam create-role

Attach a permission policy to the role: aws iam put-role-policy

* iam role with instance such as Amazon EC2 instance

create an instance profile to store a role , thn create a role

Create an instance profile: aws iam create-instance-profile

Add a role to instance profile: aws iam add-role-to-instance-profile

* Iam roles for iam user

roles-create roles – anotheraws account -accnt id -next permissn – attach policy rolename – roledescrptn - create role

* Iam roles for a thrid party service provider

web identity federation : access to the AWS resources which have signed in with the login with facebook, Google, Amazon or another Open ID standard. first create and configure the identity provider thn reate the IAM Role

<https://console.aws.amazon.com/iam/> -- role – create role-- web identity -- client id - add condition – attach policy – create role

saml : access to the AWS resources in an organization that uses SAML

<https://console.aws.amazon.com/iam/> -- role – create role-- identity provider access – grant role type (sso/api)– saml provider -- api access – select attrbutes -- attach policy – create role

IAM COMMANDS :

**To add a client ID (audience) to an Open-ID Connect (OIDC) provider**

aws iam add-client-id-to-open-id-connect-provider --client-id my-application-ID --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/server.example.com

**To add a role to an instance profile**

aws iam add-role-to-instance-profile --role-name S3Access --instance-profile-name Webserver

**To add a user to an IAM group**

aws iam add-user-to-group --user-name Bob --group-name Admins

**To attach a managed policy to an IAM group**

aws iam attach-group-policy --policy-arn arn:aws:iam::aws:policy/ReadOnlyAccess --group-name Finance

**To attach a managed policy to an IAM role**

aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/ReadOnlyAccess --role-name ReadOnlyRole

**To attach a managed policy to an IAM user**

aws iam attach-user-policy --policy-arn arn:aws:iam:ACCOUNT-ID:aws:policy/AdministratorAccess --user-name Alice

**To change the password for your IAM user**

aws iam change-password --old-password <value> --new-password <value>

**To create an access key for an IAM user**

aws iam create-access-key --user-name Bob

**To create an account alias**

aws iam create-account-alias --account-alias examplecorp

**To create an IAM group**

aws iam create-group --group-name Admins

**To create an instance profile**

aws iam create-instance-profile --instance-profile-name Webserver

**To create a password for an IAM user**

aws iam create-login-profile --user-name <value> --password <value>

**To list the IAM groups for the current account**

aws iam list-groups

**Creates a new managed policy for your AWS account.**

aws iam create-policy --policy-name my-policy --policy-document file://policy

**To create a new version of a managed policy**

aws iam create-policy-version --policy-arn arn:aws:iam::123456789012:policy/MyPolicy --policy-document file://NewPolicyVersion.json --set-as-default

**To create an IAM role**

aws iam create-role --role-name Test-Role --assume-role-policy-document [file://Test-Role-Trust-Policy.json](smb://Test-Role-Trust-Policy.json/)

**create user :**

aws iam create-user --user-name Bob

**To create a virtual MFA device**

aws iam create-virtual-mfa-device --virtual-mfa-device-name BobsMFADevice --outfile C:/QRCode.png --bootstrap-method QRCodePNG

**To deactivate an MFA device**

aws iam deactivate-mfa-device --user-name Bob --serial-number arn:aws:iam::210987654321:mfa/BobsMFADevice

**To delete an access key for an IAM user**

aws iam delete-access-key --access-key AKIDPMS9RO4H3FEXAMPLE --user-name Bob

**To delete an account alias**

aws iam delete-account-alias --account-alias mycompany

**To delete the current account password policy**

aws iam delete-account-password-policy

**To delete an IAM group**

aws iam delete-group --group-name MyTestGroup

**To delete a policy from an IAM group**

aws iam delete-group-policy --group-name Admins --policy-name ExamplePolicy

**To delete an instance profile**

aws iam delete-instance-profile --instance-profile-name ExampleInstanceProfile

**To delete a password for an IAM user** /login-profile

aws iam delete-login-profile --user-name Bob

**To create an OpenID Connect (OIDC) provider**

aws iam create-open-id-connect-provider --generate-cli-skeleton > create-open-id-connect-provider.json

aws iam create-open-id-connect-provider --url <value>

[--client-id-list <value>] --thumbprint-list <value> [--cli-input-json <value>] [--generate-cli-skeleton <value>]

**To delete an IAM OpenID Connect identity provider**

aws iam delete-open-id-connect-provider --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com

**To delete an IAM policy**

aws iam delete-policy --policy-arn arn:aws:iam::123456789012:policy/MySamplePolicy

**To delete a version of a managed policy**

aws iam delete-policy-version --policy-arn arn:aws:iam::123456789012:policy/MyPolicy --version-id v2

**To delete an IAM role**

aws iam delete-role --role-name Test-Role

installing , configuring , using aws cli

install pip :

* sudo apt-get install python-pip
* sudo apt-get install python3-pip
* upgrade pip : pip install -U pip

install aws cli :

* pip3 install awscli --upgrade --user
* aws –version
* pip3 list -o
* pip3 install --upgrade --user awscli
* pip3 install awscli --upgrade --user

configuring aws cli :

* aws configure : aws credentials
* <https://console.aws.amazon.com/iam/>
* users-add user- anamika-add permssn -add user – copy permissn – create access key – show – download csv
* aws configure (gen access key n secret value key from console )
* aws configure --profile user2 (--region, --output, and --profile)
* aws s3 ls
* aws configure list
* describe instances : aws ec2 describe-instances --output table --region us-east-1
* --profile <profilename> : named profile
* --region <string>: AWS Region to send this command's AWS request to.
* --output <string> : output format to use for this command
* --endpoint-url <string> : URL to send the request to.
* --color <string> : color output. Valid values are on, off, and auto.
* --ca-bundle <string> : CA certificate bundle to use when verifying SSL certificates.
* --cli-connect-timeout <integer> : maximum socket connect time in seconds.

Aws cli with proxy :

* AWS CLI supports HTTP Basic authentication.
* export HTTPS\_PROXY=http://10.15.20.25:5678
* export HTTP\_PROXY=http://username:[password@proxy.example.com](mailto:password@proxy.example.com):1234
* proxy on ec2 : export NO\_PROXY=169.254.169.254

command completion :

**Tab** key to complete a partially typed command. two pieces of information: the name of the shell you're using and the location of the aws\_completer script.

* echo $SHELL
* ps
* which aws\_completer
* find / -name aws\_completer
* export PATH=/usr/bin/:$PATH
* source ~/.bash\_profile
* complete -C '/usr/bin/aws\_completer' aws
* aws s s

using aws cli

* aws ec2 describe-instances help
* aws ec2 help
* aws help
* command structure in aws cli : aws <command> <subcommand> [options and parameters]
* parameter pass in aws cli : aws ec2 create-key-pair --key-name 'my key pair'
* --instance-types m1.xlarge m1.medium
* --dry-run
* --filters file://filter.json
* aws ec2 run-instances --image-id ami-12345678 --block-device-mappings http://my-bucket.s3.amazonaws.com/filename.json
* ci token :
* aws ec2 run-instances --generate-cli-skeleton > ec2runinst.json
* aws ec2 stop-instances --instance-ids '["i-1486157a","i-1286157c","i-ec3a7e87"]'
* aws sns publish --topic-arn arn:aws:sns:us-east-1:546419318123:OperationsError --message "Script Failure"
* aws sqs receive-message --queue-url <https://queue.amazonaws.com/546419318123/Test>

------------------------------------------------------------------------------------------------------------------------

# S3 Bucket

S3 is object-based. <key:value pair >unlimited storage , one of the first services that has been produced by aws. , secure, durable, highly scalable object storage. universal namespace .

S3 replicates the data to multiple servers to achieve high availability.

Bucket :

If you create a bucket, URL look like: HTTP 200 code means that the uploading of a file is successful.



* create bucket - name bucket – STORE FILES , like folder ,
* Buckets are the containers in S3 - stores data.
* unique name to generate a unique DNS address ,
* unlimited upload n dwnload wd permissn ,
* standard interfaces REST and SOAP interfaces ,
* security by protecting unauthorized users from accessing your data,
* ownership of a bucket is not transferrable .

OBJECTS AND KEY :

* <name of the object: data inside the file> = <KEY:VALUE > PAIR
* object : stored in an S3 , consists of object data and metadata , default metadata such as date last modified, and standard HTTP metadata, such as Content type. Custom metadata can also be specified at the time of storing an object. METADATA data about data that you are storing.
* Version ID uniquely identifies the object
* key : unique identifier for an object , object is associated with key , combination of bucket name, key, and version ID signifies the object
* http://jtp.s3.amazonaws.com/2019-01-31/Amazons3.wsdl where "jtp" is the bucket name, and key is "2019-01-31/Amazons3.wsdl"

REGION :

* geographical region in which you want to store the buckets , Objects will not leave the region unless you explicitly transfer the objects to another region.

DATA CONSISTENCY MODEL :

overwrite PUTS and DELETES : PUTS and DELETES to objects, the changes are reflected eventually, and they are not available immediately.

PUTS of new objects. : PUTS does not take time for propagation, the changes are reflected immediately.

# Creating and configuring a s3 bucket :

* Login to aws – s3 services - create bucket – bucket name – unique – lowecase – create
* add a object in bucket :

upload – upload files – add files – upload – click to get object url – hit url – no access

* view an object

set permission – uncheck all – save – actions – make public – hit url – able to access

* move/delete an obj :

bucket name – name – selct obj – del /move object

* Encryption is of two types, i.e., Client Side Encryption and Server Side Encryption
* Access to the buckets can be controlled by using either ACL (Access Control List) or bucket policies.
* Viewing buckt proprties :

console – bucket – bucket name slct – properties

* enable/disable versioning : keep multiple versions of an object in one bucket.

console – bucket – bucket name slct – properties -versioning – suspend/select

* enable default encryption :default encryption behavior for an S3 bucket. objects are encrypted using server-side encryption with either Amazon S3-managed keys (SSE-S3) or AWS KMS-managed keys (SSE-KMS). server-side encryption, Amazon S3 encrypts an object before saving it to disk in its data centers and decrypts it when you download the objects.

Console- bucket name – properties – default encryption - choose **AES-256** or **AWS-KMS**.

256: managed by Amazon S3 for default encryption

kms: managed by AWS KMS for default encryption,

choose a master key fromAWS KMS master keys - Amazon Resource Name (ARN) of the AWS KMS key - ARN for your AWS KMS key in the IAM (**Encryption keys**.)

* enable server access logging for s3 bucket

Console- bucket name – properties –server access logging – bucket name – region name – save

* enable object level logging

Console- bucket name – properties – obj level logging – exixting cloudtrail select (or creae new) – selcet event – read /wrte – create

read: to log Amazon S3 read API : GET OBJECT

write :to log Amazon S3 write APIs : PUT OBJECT

COMMANDS :

CP:

aws s3 cp file.txt s3://mybucket/ --grants read=uri=http://acs.amazonaws.com/groups/global/AllUsers [full=emailaddress=user@example.com](mailto:full%3Demailaddress%3Duser@example.com) --expires 2014-10-01T20:30:00Z

LS:

aws s3 ls s3://mybucket

MB:

aws s3 mb s3://mybucket --region us-west-1

MV:

aws s3 mv test.txt s3://mybucket/test2.txt --recursive --exclude "\*.jpg"

--acl public-read-write

PRESIGN : Generate a pre-signed URL for an Amazon S3 object.

aws s3 presign s3://awsexamplebucket/test2.txt --expires-in 604800

RB : Deletes an empty S3 bucket

aws s3 rb s3://mybucket --force

RM : Deletes an S3 object.

aws s3 rm s3://mybucket/test2.txt –recursive --exclude "\*.jpg"

SYNC:

aws s3 sync . s3://mybucket --exclude "\*.jpg" --source-region us-west-2 --region us-east-1

aws s3 sync s3://mybucket . (DOWNLOAD)

* copy a object into s3 bucket location with permission

aws s3 cp install-sshpass.sh s3://srev-emr-test/sbx/stk1/repository/bootstrap-actions/ --acl bucket-owner-full-control --acl authenticated-read

* Copy MyFile.txt in current directory to s3://my-bucket/path

aws s3 cp install-sshpass.sh s3://srev-emr-test/sbx/stk1/repository/bootstrap-actions/

* grant permission to the file :

aws s3 cp install-sshpass.sh s3://srev-emr-test/sbx/stk1/repository/bootstrap-actions/ --acl bucket-owner-full-control --acl authenticated-read

* Create s3 bucket

aws s3 mb s3://bucket-name//

* Move all .jpg files in s3://my-bucket/path to ./MyDirectory

aws s3 mv s3://my-bucket/path ./MyDirectory --exclude "\*" --include "\*.jpg" --recursive

* List the contents of my-bucket

aws s3 ls s3://my-bucket

* List the contents of path in my-bucket

aws s3 ls s3://my-bucket/path/

* Delete s3://my-bucket/path/MyFile.txt

aws s3 rm s3://my-bucket/path/MyFile.txt

* Delete s3://my-bucket/path and all of its contents

aws s3 rm s3://my-bucket/path --recursive

* create a local cluster with command /cluster launch from local

bin/orca deploy dataservices-emr-cluster V2Dev DataServices-5.14.0 SBX STK1 Ssh --dry-run

* install aws sdk

sudo gem install aws-sdk

EC2 INSTANCE

* Amazon Elastic Compute Cloud  provides scalable computing capacity in the Amazon Web Services
* Virtual computing environments, known as instances
* templates for your instances, known as Amazon Machine Images
* Various configurations of CPU, memory, storage, and networking capacity for your instances, known as instance types
* Secure login information for your instances using key pairs
* Storage volumes for temporary data that's deleted when you stop or terminate your instance, known as instance store volumes
* Persistent storage volumes for your data using Amazon Elastic Block Store
* Multiple physical locations for your resources, such as instances and Amazon EBS volumes, known as Regions and Availability Zones
* A firewall that enables you to specify the protocols, ports, and source IP ranges that can reach your instances using security groups
* Static IPv4 addresses for dynamic cloud computing, known as Elastic IP addresses
* Metadata, known as tags, that you can create and assign to your Amazon EC2 resources
* Virtual networks you can create that are logically isolated from the rest of the AWS cloud, and that you can optionally connect to your own network, known asvirtual private clouds (VPCs)

launch a t2.micro instance in the specified subnet of a VPC

aws ec2 run-instances --image-id *ami-xxxxxxxx* --count 1 --instance-type t2.micro --key-name *MyKeyPair* --security-group-ids *sg-903004f8* --subnet-id *subnet-6e7f829e*

launch a t1.microinstance in EC2-Classic

aws ec2 run-instances --image-id *ami-173d747e* --count 1 --instance-type t1.micro --key-name *MyKeyPair* --security-groups *my-sg*

## **Adding a Block Device to Your Instance**

--block-device-mappings "[{\"DeviceName\":\"*/dev/sdf*\",\"Ebs\":{\"VolumeSize\":*20*,\"DeleteOnTermination\":false}}]"

adding a tag to instance

aws ec2 create-tags --resources *i-5203422c* --tags Key=*Name*,Value=*MyInstance*

listing instance :

aws ec2 describe-instances --filters "Name=tag:*Name*,Values=*MyInstance*"

aws ec2 describe-instances --filters "Name=image-id,Values=*ami-x0123456*,*ami-y0123456*,*ami-z0123456*"