🔰 “ Create High Availability Architecture with AWS CLI ” 🔰

Amazon CloudFront is a web service that speeds up distribution of your static and dynamic web content, such as .html, .css, .js, and image files, to your users. CloudFront delivers your content through a worldwide network of data centers called edge locations. When a user requests content that you're serving with CloudFront, the request is routed to the edge location that provides the lowest latency (time delay), so that content is delivered with the best possible performance.

Use cases:

- If the content is already in the edge location with the lowest latency, CloudFront delivers it immediately.

- If the content is not in that edge location, CloudFront retrieves it from an origin that you've defined—such as an Amazon S3 bucket, a MediaPackage channel, or an HTTP server (for example, a web server) that you have identified as the source for the definitive version of your content.

🔆The architecture includes-

- Webserver configured on EC2 Instance

- Document Root(/var/www/html) made persistent by mounting on EBS Block Device.

- Static objects used in code such as pictures stored in S3

- Setting up Content Delivery Network using CloudFront and using the origin domain as S3 bucket.

- Finally place the Cloud Front URL on the webapp code for security and low latency.

STEP 1 : Logging in to our aws using our command line interface.

STEP 2: Launching our instance using our created key-pair and Security group through aws cli.

abhijeet — less ◂ aws ec2 run-instances --image-id ami-0eeb03e72075b9bcc --instance-type t2.micro --key-name BabykeyArth --security-group-ids sg-0e3925fd0299f54b4 --count 1 — 163×48

```
(base) abhijeet@Abhijeets-MacBook-Pro ~ % aws ec2 create-security-group --description "allow all" --group-name mybsg
{
    "GroupId": "sg-0e3925fd0299f54b4"
}
(base) abhijeet@Abhijeets-MacBook-Pro ~ % aws ec2 authorize-security-group-ingress --group-id sg-0e3925fd0299f54b4 --protocol tcp --port 80 --cidr 0.0.0.0/0
(base) abhijeet@Abhijeets-MacBook-Pro ~ % aws ec2 run-instances --image-id ami-0eeb03e72075b9bcc --instance-type t2.micro --key-name BabykeyArth --security-group-i
ds sg-0e3925fd0299f54b4 --count 1
{
    "Groups": [],
    "Instances": [
        {
            "AmiLaunchIndex": 0,
            "ImageId": "ami-0eeb03e72075b9bcc",
            "InstanceId": "i-0ed93e0787d5d1890",
            "InstanceType": "t2.micro",
            "KeyName": "BabykeyArth",
            "LaunchTime": "2021-03-21T06:03:21+00:00",
            "Monitoring": {
                "State": "disabled"
            },
            "Placement": {
                "AvailabilityZone": "ap-south-1a",
                "GroupName": "",
                "Tenancy": "default"
            },
            "PrivateDnsName": "ip-172-31-37-64.ap-south-1.compute.internal",
            "PrivateIpAddress": "172.31.37.64",
            "ProductCodes": [],
            "PublicDnsName": "",
            "State": {
                "Code": 0,
                "Name": "pending"
            },
            "StateTransitionReason": "",
            "SubnetId": "subnet-2edccd46",
            "VpcId": "vpc-b7678fdc",
            "Architecture": "x86_64",
            "BlockDeviceMappings": [],
            "ClientToken": "1bda306d-84d4-479d-a602-d53244a4fce5",
            "EbsOptimized": false,
            "EnaSupport": true,
            "Hypervisor": "xen",
            "NetworkInterfaces": [
                {
:
```

STEP 3: Installing our HTTPD server through yum install httpd command.

STEP 4: Connected EBS volume to our EC-2 instance

STEP 5: Formatting our EBS volume so that we can mount.

STEP 6: Mounting our formatted drive to the /var/www/html storage and making it persistent.

STEP 7: Creating our S3 bucket through cli.

STEP 8: Adding image to our S3 storage.

STEP 8: Checking the S3 bucket through amazons graphical interface.

STEP 8: Creating CloudFront distribution.

abhijeet — less ◂ aws cloudfront create-distribution --origin-domain-name babybucket-arth.s3.amazonaws.com — 163×48

```
(base) abhijeet@Abhijeets-MacBook-Pro ~ % aws cloudfront create-distribution --origin-domain-name babybucket-arth.s3.amazonaws.com
{
    "Location": "https://cloudfront.amazonaws.com/2019-03-26/distribution/E1L20UIHK6FZ3",
    "ETag": "E17XMXWVGPULD7",
    "Distribution": {
        "Id": "E1L20UIHK6FZ3",
        "ARN": "arn:aws:cloudfront::294702585446:distribution/E1L20UIHK6FZ3",
        "Status": "InProgress",
        "LastModifiedTime": "2021-03-21T06:39:53.168000+00:00",
        "InProgressInvalidationBatches": 0,
        "DomainName": "d7m6dxk6i6qlt.cloudfront.net",
        "ActiveTrustedSigners": {
            "Enabled": false,
            "Quantity": 0
        },
        "DistributionConfig": {
            "CallerReference": "cli-1616308791-764",
            "Aliases": {
                "Quantity": 0
            },
            "DefaultRootObject": "",
            "Origins": {
                "Quantity": 1,
                "Items": [
                    {
                        "Id": "babybucket-arth.s3.amazonaws.com-1616308791-648849",
                        "DomainName": "babybucket-arth.s3.amazonaws.com",
                        "OriginPath": "",
                        "CustomHeaders": {
                            "Quantity": 0
                        },
                        "S3OriginConfig": {
                            "OriginAccessIdentity": ""
                        },
                        "ConnectionAttempts": 3,
                        "ConnectionTimeout": 10
                    }
                ]
            },
            "OriginGroups": {
                "Quantity": 0
            },
            "DefaultCacheBehavior": {
                "TargetOriginId": "babybucket-arth.s3.amazonaws.com-1616308791-648849",
                "ForwardedValues": {
                    "QueryString": false,
                    "Cookies": {
                        "Forward": "none"
```

STEP 9: Starting our HTTPD server.

STEP 10: Showing that the EBS is mounted to /var/www/html folder.

STEP 11: Finally adding a very basic html code to our webpage folder so that when client opens the webpage it will display image through our object storage(S3).