DICE
ANALYTICS

AWS

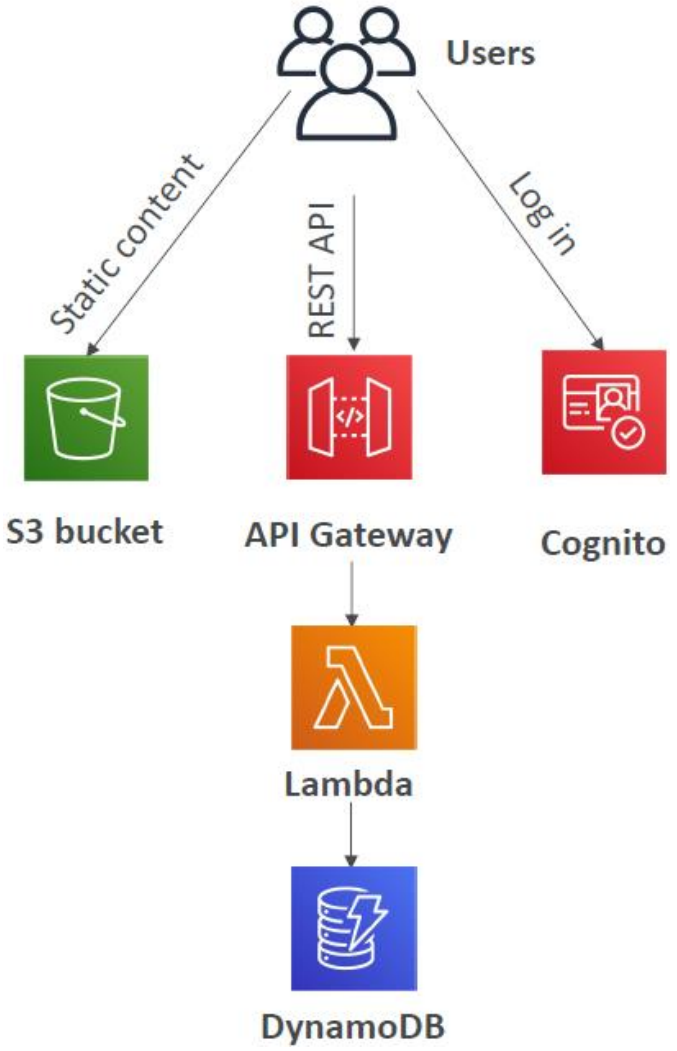
Serverless Overview

What’s serverless?

- Serverless is a new paradigm in which the developers don’t have to manage servers anymore…
- They just deploy code
- They just deploy… functions !
- Initially... Serverless == FaaS (Function as a Service)
- Serverless was pioneered by AWS Lambda but now also includes anything that’s managed: “databases, messaging, storage, etc.”
- Serverless does not mean there are no servers… it means you just don’t manage / provision / see them

Serverless in AWS

- AWS Lambda
- DynamoDB
- AWS Cognito
- AWS API Gateway
- Amazon S3
- AWS SNS & SQS
- AWS Kinesis Data Firehose
- Aurora Serverless
- Step Functions
- Fargate



Why AWS Lambda

Amazon EC2

- Virtual Servers in the Cloud
- Limited by RAM and CPU
- Continuously running
- Scaling means intervention to add / remove servers

Amazon Lambda

- Virtual **functions** – no servers to manage!
- Limited by time - **short executions**
- Run **on-demand**
- **Scaling is automated**!

AWS Lambda language support

- Node.js (JavaScript)
- Python
- Java (Java 8 compatible)
- C# (.NET Core)
- Golang
- C# / Powershell
- Ruby
- Custom Runtime API (community supported, example Rust)

- Lambda Container Image
  - The container image must implement the Lambda Runtime API
  - ECS / Fargate is preferred for running arbitrary Docker images

AWS Lambda Integrations
Main ones

API Gateway

Kinesis

DynamoDB

S3

CloudFront

CloudWatch Events
EventBridge

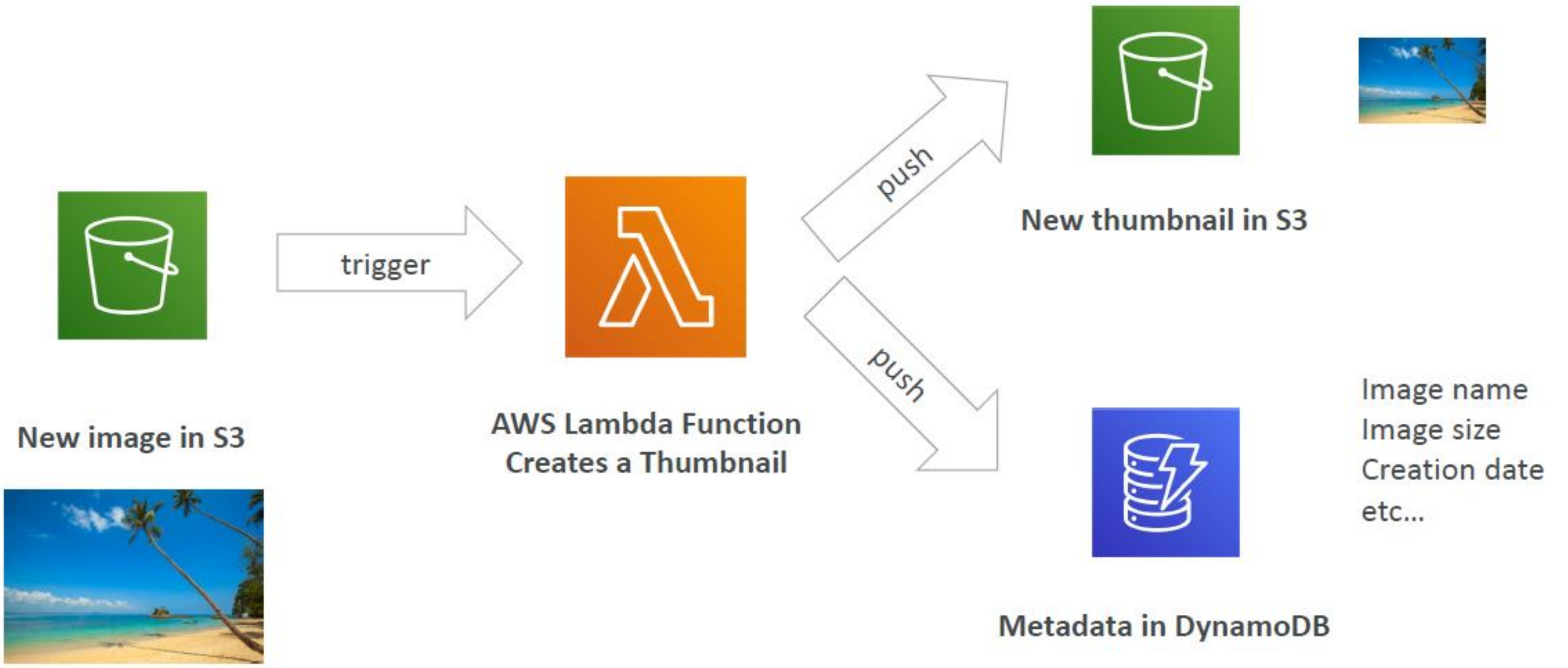
CloudWatch Logs

SNS

SQS

Cognito

Example: Serverless Thumbnail creation



New image in S3

trigger

AWS Lambda Function
Creates a Thumbnail

push

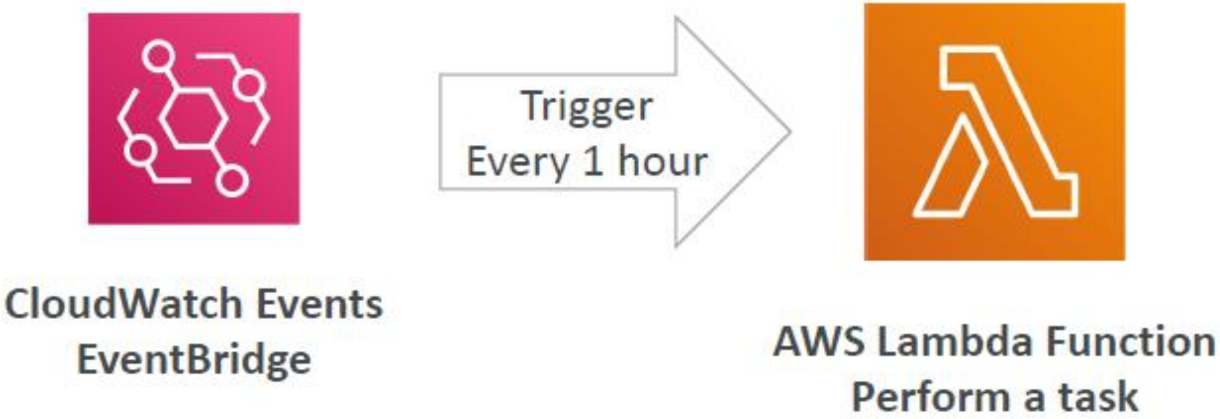
push

New thumbnail in S3

Metadata in DynamoDB

Image name
Image size
Creation date
etc…

Example: Serverless CRON Job



CloudWatch Events
EventBridge

Trigger
Every 1 hour

AWS Lambda Function
Perform a task

AWS Lambda Pricing: example

- You can find overall pricing information here: https://aws.amazon.com/lambda/pricing/
- Pay per **calls**:
  - First 1,000,000 requests are free
  - $0.20 per 1 million requests thereafter ($0.0000002 per request)
- Pay per **duration**: (in increment of 1 ms)
  - 400,000 GB-seconds of compute time per month for FREE
  - == 400,000 seconds if function is 1GB RAM
  - == 3,200,000 seconds if function is 128 MB RAM
  - After that $1.00 for 600,000 GB-seconds
- It is usually very cheap to run AWS Lambda so it's very popular

AWS Lambda Limits to Know - per region

- Execution:
  - Memory allocation: 128 MB – 10GB (1 MB increments)
  - Maximum execution time: 900 seconds (15 minutes)
  - Environment variables (4 KB)
  - Disk capacity in the "function container" (in /tmp): 512 MB
  - Concurrency executions: 1000 (can be increased)
- Deployment:
  - Lambda function deployment size (compressed .zip): 50 MB
  - Size of uncompressed deployment (code + dependencies): 250 MB
  - Can use the /tmp directory to load other files at startup
  - Size of environment variables: 4 KB

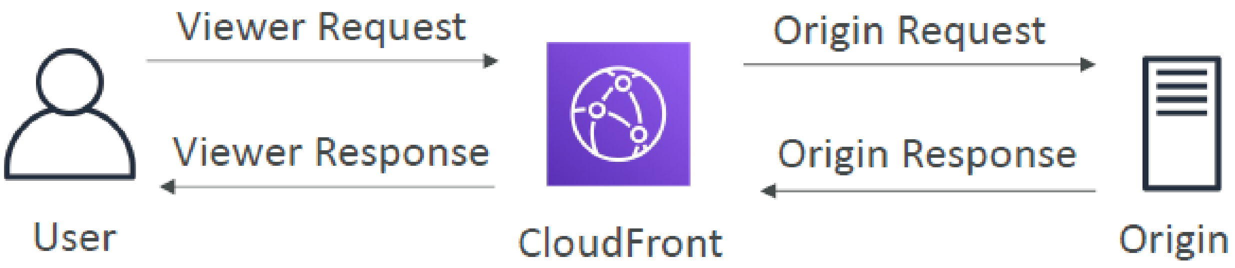
Lambda@Edge

- You have deployed a CDN using CloudFront
- What if you wanted to run a global AWS Lambda alongside?
- Or how to implement request filtering before reaching your application?

- For this, you can use **Lambda@Edge**:
  deploy Lambda functions alongside your CloudFront CDN
  - Build more responsive applications
  - You don't manage servers, Lambda is deployed globally
  - Customize the CDN content
  - Pay only for what you use

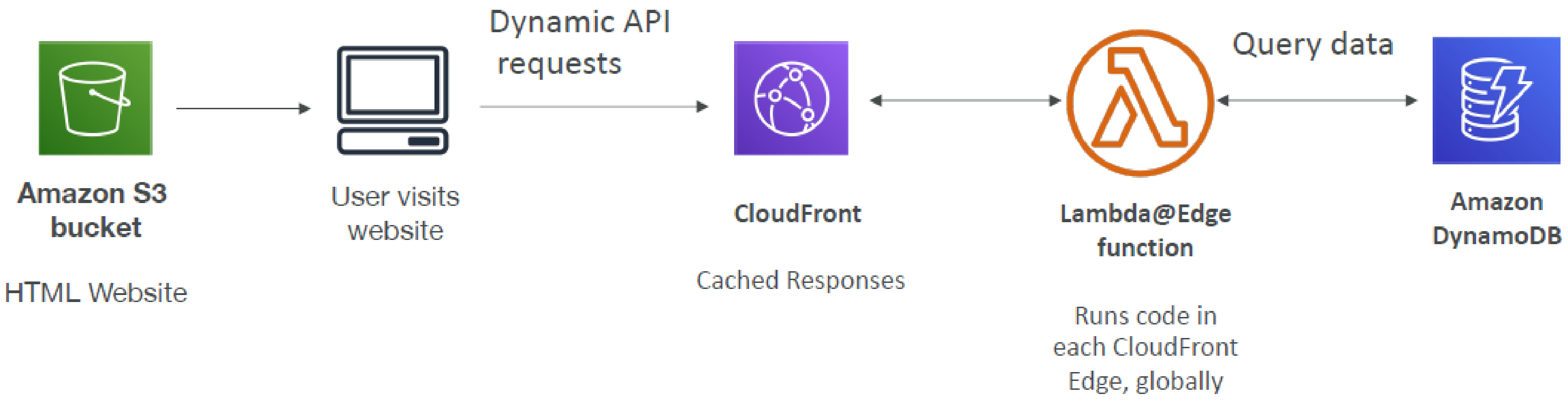
Lambda@Edge

- You can use Lambda to change CloudFront requests and responses:
  - After CloudFront receives a request from a viewer (viewer request)
  - Before CloudFront forwards the request to the origin (origin request)
  - After CloudFront receives the response from the origin (origin response)
  - Before CloudFront forwards the response to the viewer (viewer response)



- You can also generate responses to viewers without ever sending the request to the origin

Lambda@Edge: Global application



Lambda@Edge: Use Cases

- Website Security and Privacy
- Dynamic Web Application at the Edge
- Search Engine Optimization (SEO)
- Intelligently Route Across Origins and Data Centers
- Bot Mitigation at the Edge
- Real-time Image Transformation
- A/B Testing
- User Authentication and Authorization
- User Prioritization
- User Tracking and Analytics

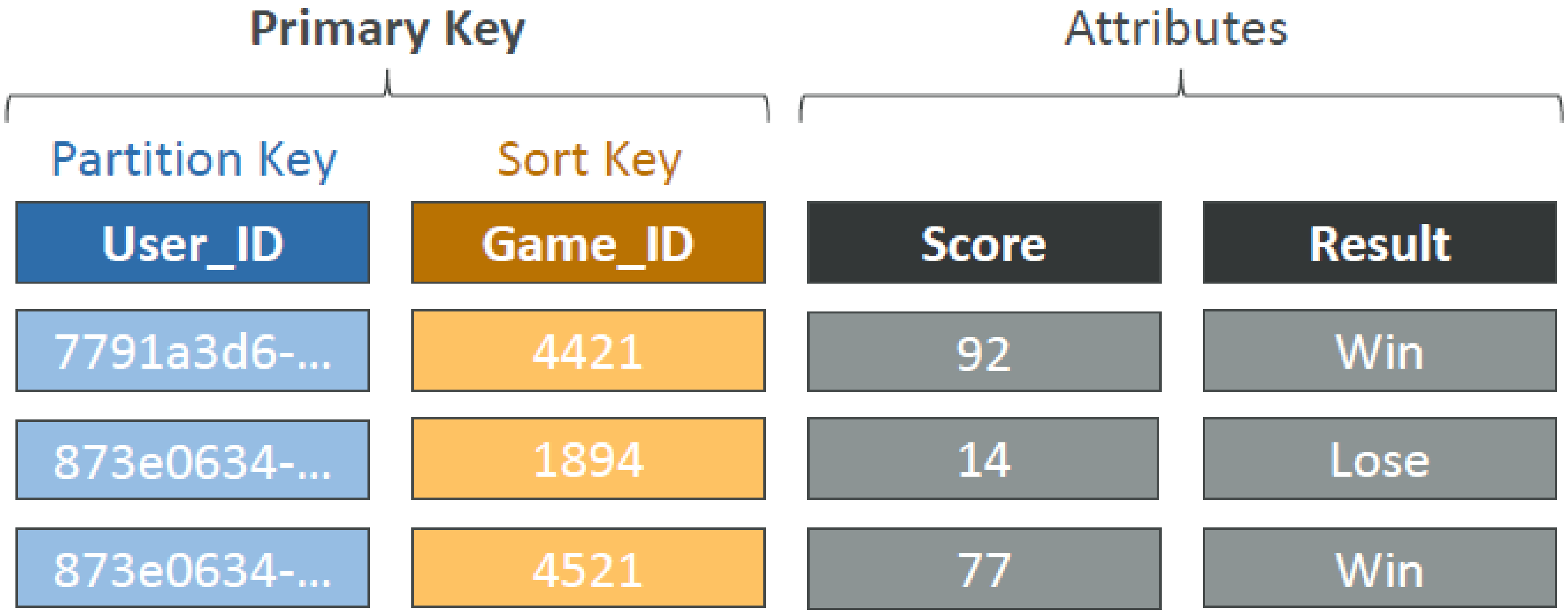
Amazon DynamoDB

- Fully managed, highly available with replication across multiple AZs
- NoSQL database - not a relational database
- Scales to massive workloads, distributed database
- Millions of requests per seconds, trillions of row, 100s of TB of storage
- Fast and consistent in performance (low latency on retrieval)
- Integrated with IAM for security, authorization and administration
- Enables event driven programming with DynamoDB Streams
- Low cost and auto-scaling capabilities
- Standard & Infrequent Access (IA) Table Class

DynamoDB - Basics

- DynamoDB is made of **Tables**
- Each table has a **Primary Key** (must be decided at creation time)
- Each table can have an infinite number of items (= rows)
- Each item has **attributes** (can be added over time – can be null)
- Maximum size of an item is **400KB**
- Data types supported are:
    - **Scalar Types** – String, Number, Binary, Boolean, Null
    - **Document Types** – List, Map
    - **Set Types** – String Set, Number Set, Binary Set

DynamoDB – Table example



Primary Key		Attributes	
Partition Key	Sort Key		
User_ID	Game_ID	Score	Result
7791a3d6-…	4421	92	Win
873e0634-…	1894	14	Lose
873e0634-…	4521	77	Win

DynamoDB – Read/Write Capacity Modes

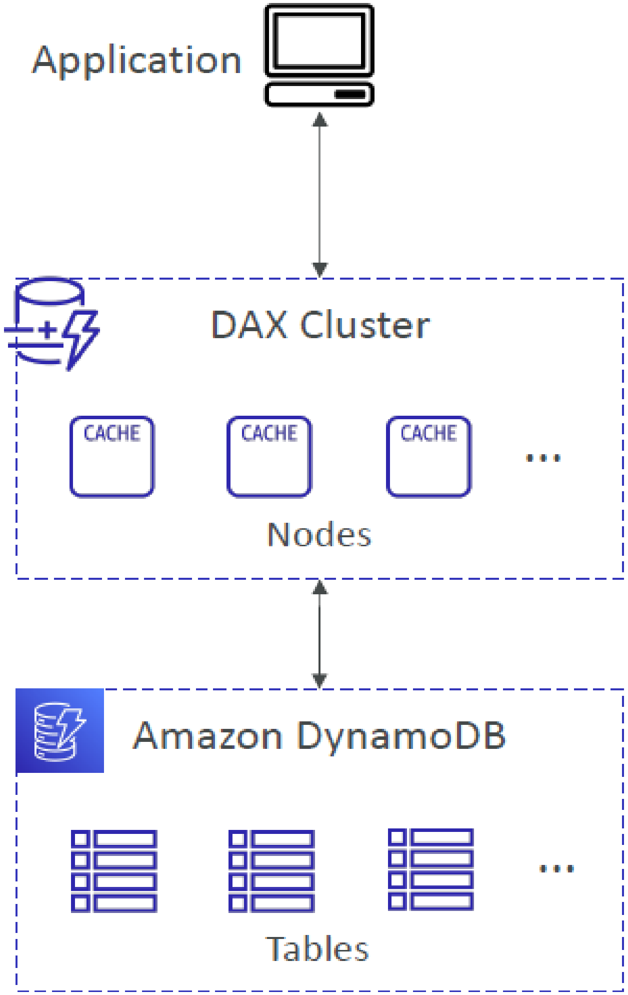
- Control how you manage your table's capacity (read/write throughput)

- Provisioned Mode (default)
  - You specify the number of reads/writes per second
  - You need to plan capacity beforehand
  - Pay for provisioned Read Capacity Units (RCU) & Write Capacity Units (WCU)
  - Possibility to add auto-scaling mode for RCU & WCU

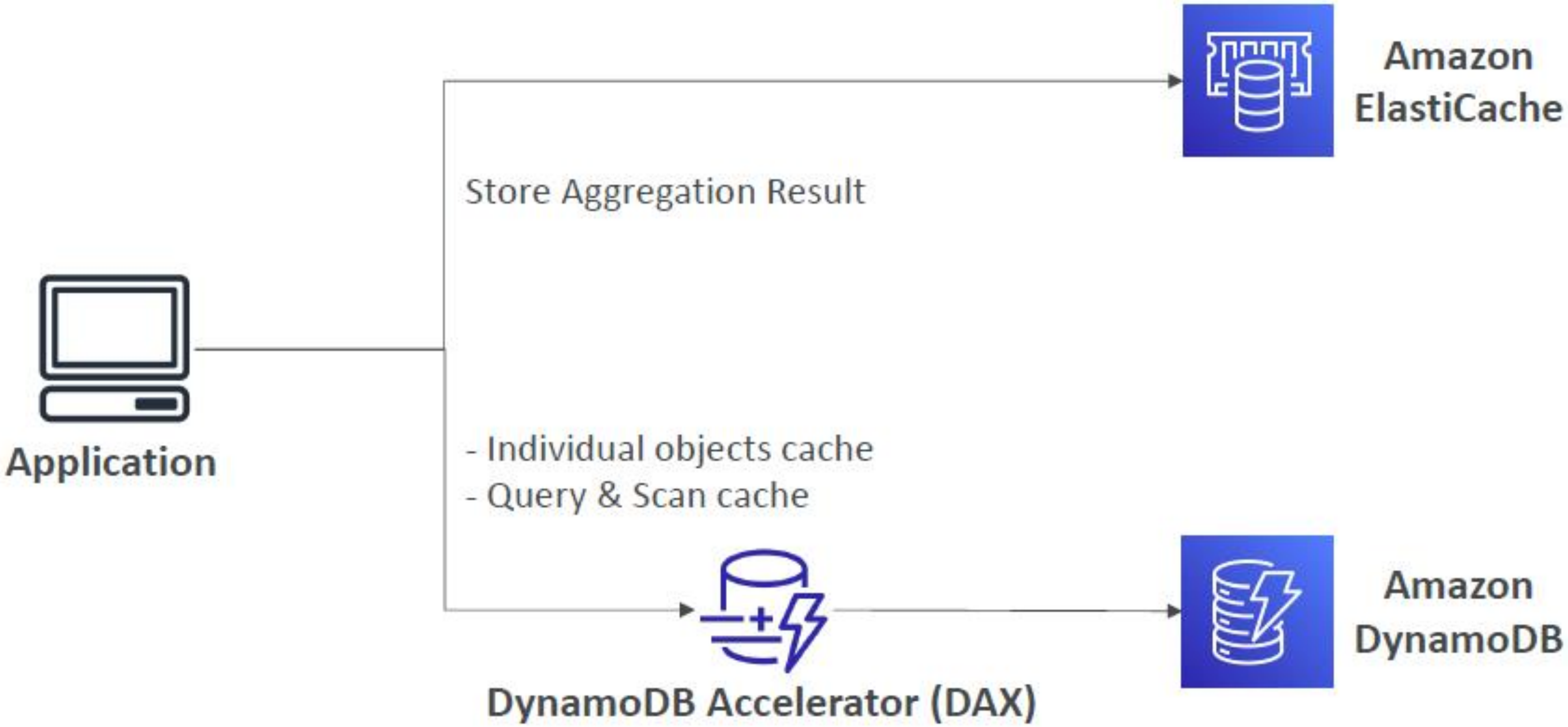
- On-Demand Mode
  - Read/writes automatically scale up/down with your workloads
  - No capacity planning needed
  - Pay for what you use, more expensive ($$$)
  - Great for unpredictable workloads

DynamoDB Accelerator (DAX)

- Fully-managed, highly available, seamless in-memory cache for DynamoDB
- **Help solve read congestion by caching**
- Microseconds latency for cached data
- Doesn't require application logic modification (compatible with existing DynamoDB APIs)
- 5 minutes TTL for cache (default)



DynamoDB Accelerator (DAX) vs. ElastiCache



Store Aggregation Result

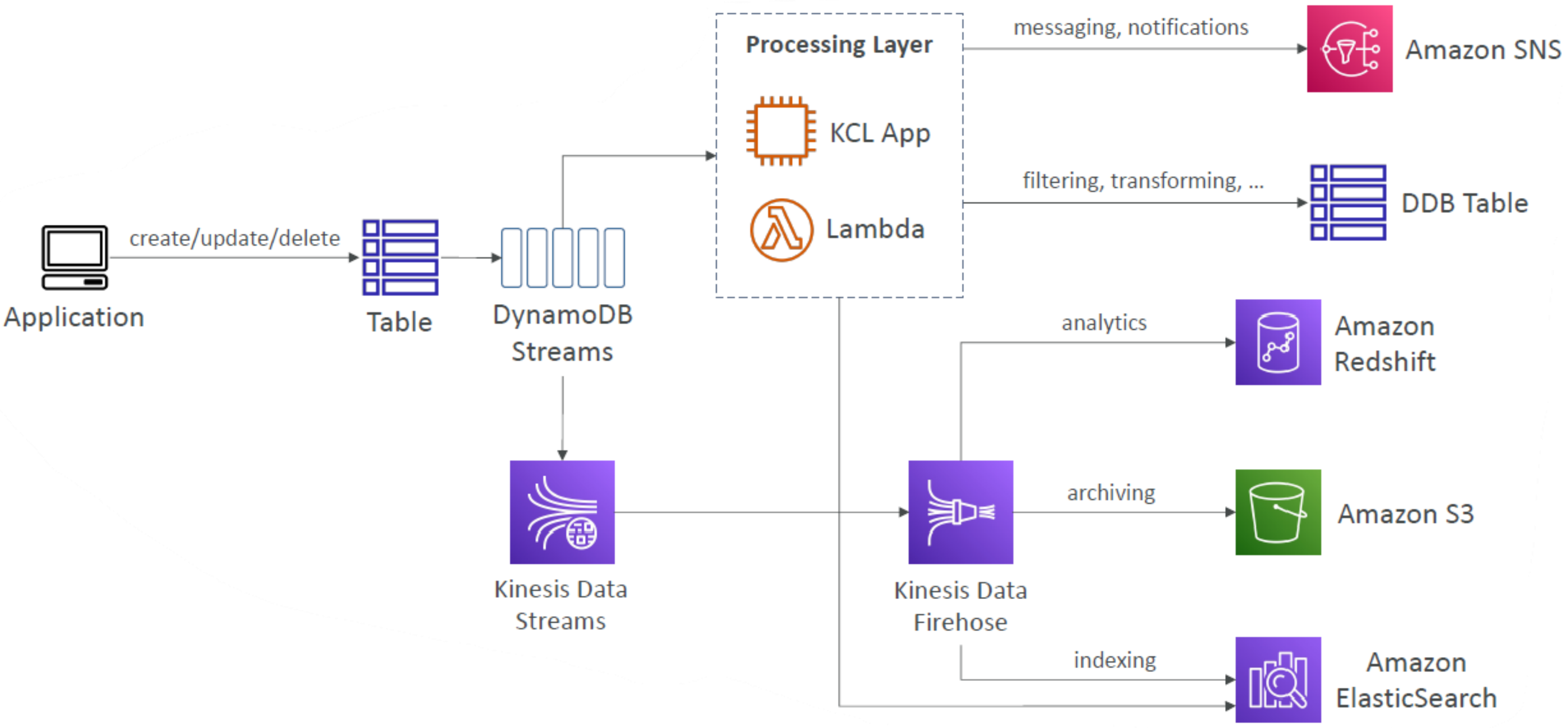
- Individual objects cache
- Query & Scan cache

DynamoDB Streams

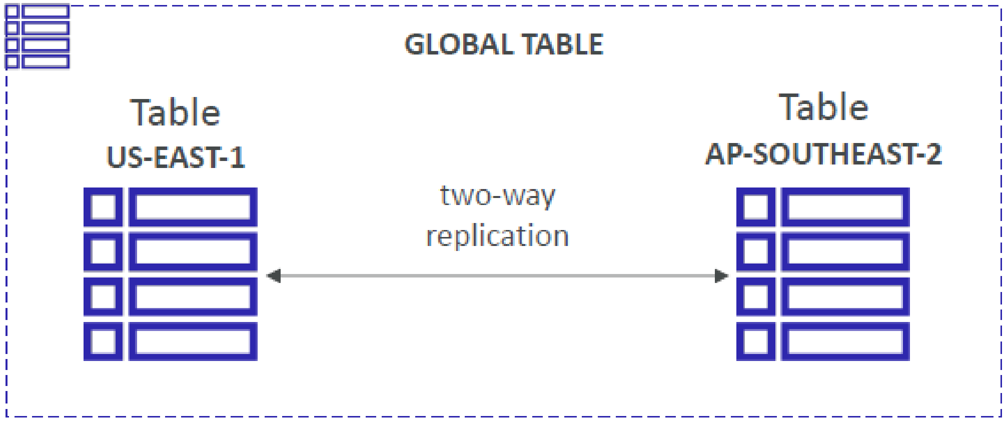
□□□□□

- Ordered stream of item-level modifications (create/update/delete) in a table
- Stream records can be:
  - Sent to **Kinesis Data Streams**
  - Read by **AWS Lambda**
  - Read by **Kinesis Client Library applications**
- Data Retention for up to 24 hours
- Use cases:
  - react to changes in real-time (welcome email to users)
  - Analytics
  - Insert into derivative tables
  - Insert into ElasticSearch
  - Implement cross-region replication

DynamoDB Streams



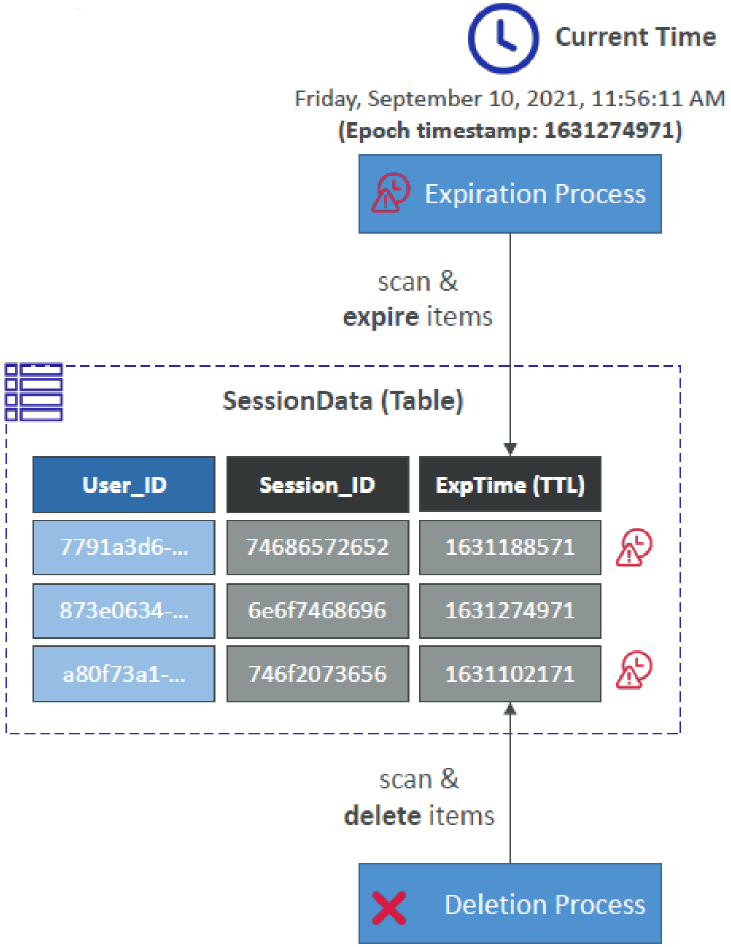
DynamoDB Global Tables



- Make a DynamoDB table accessible with **low latency** in multiple-regions
- Active-Active replication
- Applications can **READ** and **WRITE** to the table in any region
- Must enable DynamoDB Streams as a pre-requisite

DynamoDB – Time To Live (TTL)

- Automatically delete items after an expiry timestamp
- Use cases: reduce stored data by keeping only current items, adhere to regulatory obligations, …



Current Time

Friday, September 10, 2021, 11:56:11 AM
(Epoch timestamp: 1631274971)

User_ID	Session_ID	ExpTime (TTL)
7791a3d6-...	74686572652	1631188571
873e0634-...	6e6f7468696	1631274971
a80f73a1-...	746f2073656	1631102171

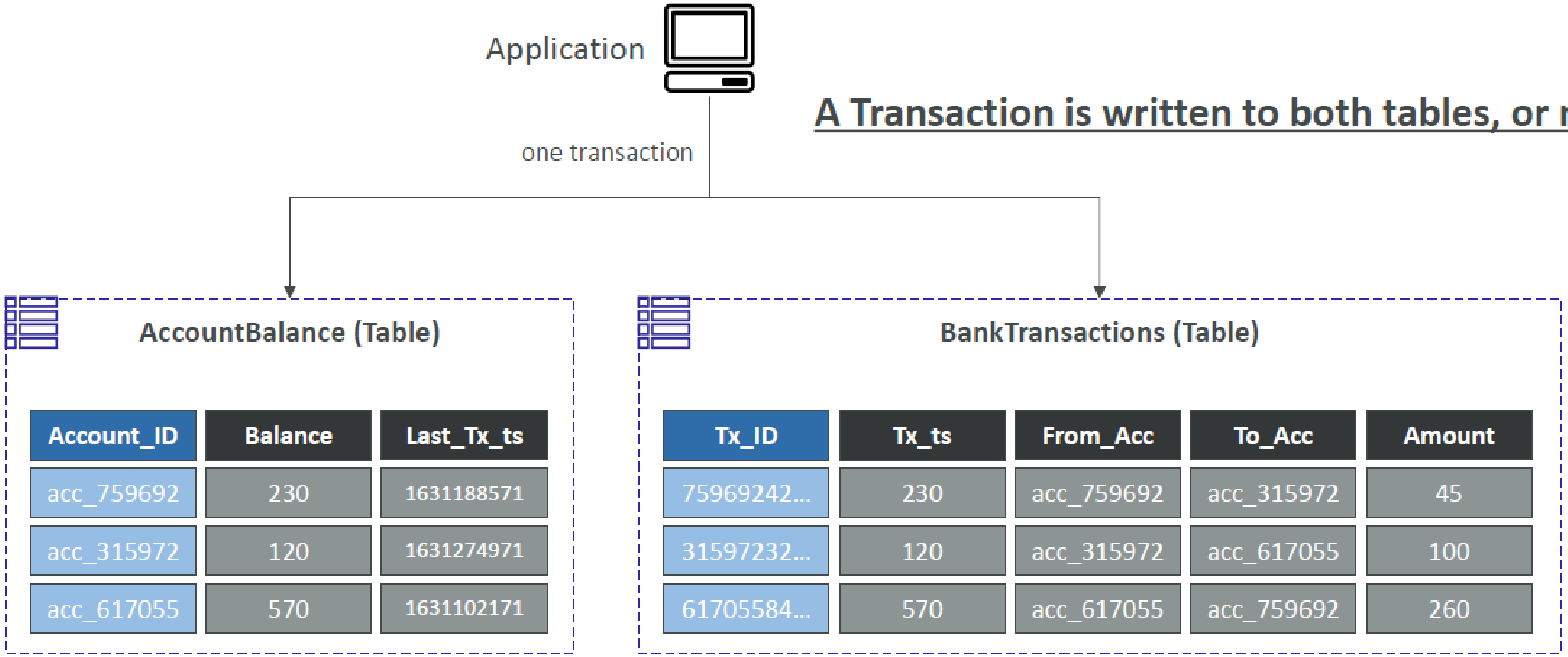
DynamoDB - Indexes

- Global Secondary Indexes (GSI) & Local Secondary Indexes (LSI)
- High level: allow to **query** on attributes other than the Primary Key

Primary Key		Attributes		
Partition Key	Sort Key			
User_ID	Game_ID	Game_TS	Score	Result
7791a3d6-…	4421	"2021-03-15T17:43:08"	92	Win
873e0634-…	4521	"2021-06-20T19:02:32"		Lose
a80f73a1-…	1894	"2021-02-11T04:11:31"	77	Win

- With Indexes, we can query by Game ID, Game_TS, Score, Result, etc…

DynamoDB - Transactions



A Transaction is written to both tables, or none!

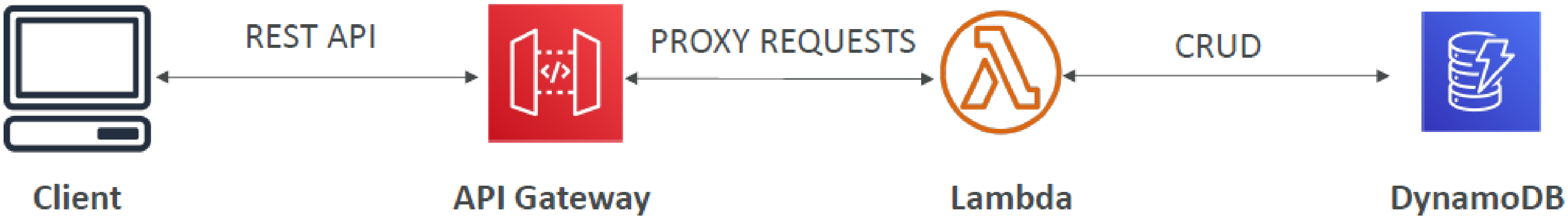
AccountBalance (Table)

Account_ID	Balance	Last_Tx_ts
acc_759692	230	1631188571
acc_315972	120	1631274971
acc_617055	570	1631102171

BankTransactions (Table)

Tx_ID	Tx_ts	From_Acc	To_Acc	Amount
75969242...	230	acc_759692	acc_315972	45
31597232...	120	acc_315972	acc_617055	100
61705584...	570	acc_617055	acc_759692	260

Example: Building a Serverless API



AWS API Gateway

- AWS Lambda + API Gateway: No infrastructure to manage
- Support for the WebSocket Protocol
- Handle API versioning (v1, v2…)
- Handle different environments (dev, test, prod…)
- Handle security (Authentication and Authorization)
- Create API keys, handle request throttling
- Swagger / Open API import to quickly define APIs
- Transform and validate requests and responses
- Generate SDK and API specifications
- Cache API responses

API Gateway – Integrations High Level

- **Lambda Function**
  - Invoke Lambda function
  - Easy way to expose REST API backed by AWS Lambda

- **HTTP**
  - Expose HTTP endpoints in the backend
  - Example: internal HTTP API on premise, Application Load Balancer…
  - Why? Add rate limiting, caching, user authentications, API keys, etc…

- **AWS Service**
  - Expose any AWS API through the API Gateway?
  - Example: start an AWS Step Function workflow, post a message to SQS
  - Why? Add authentication, deploy publicly, rate control…

API Gateway - Endpoint Types

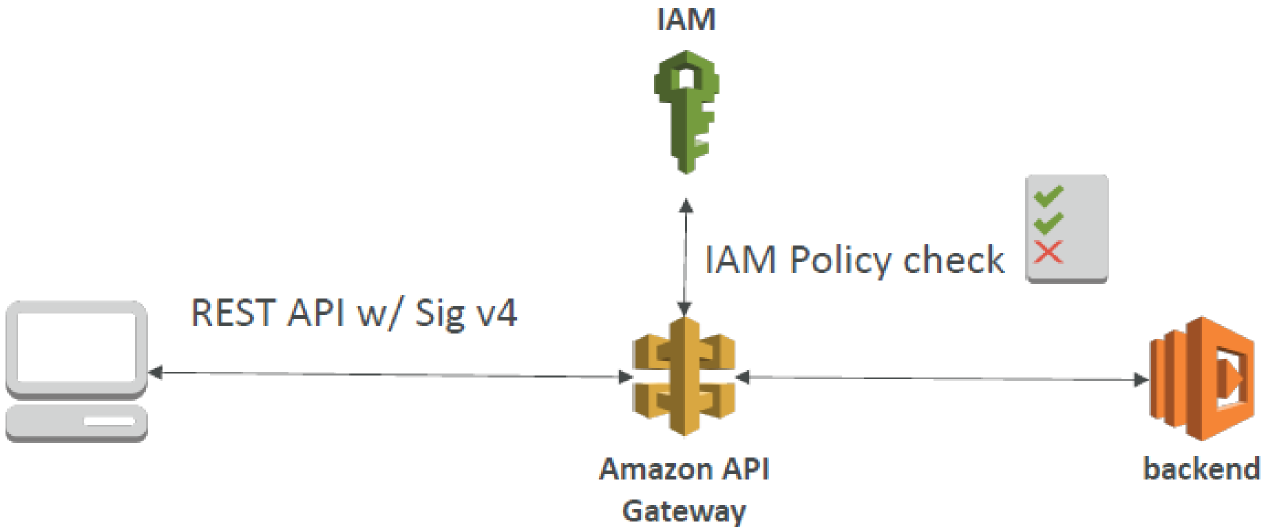
- **Edge-Optimized (default):** For global clients
  - Requests are routed through the CloudFront Edge locations (improves latency)
  - he API Gateway still lives in only one region

- **Regional:**
  - For clients within the same region
  - Could manually combine with CloudFront (more control over the caching strategies and the distribution)

- **Private:**
  - Can only be accessed from your VPC using an interface VPC endpoint (ENI)
  - Use a resource policy to define access

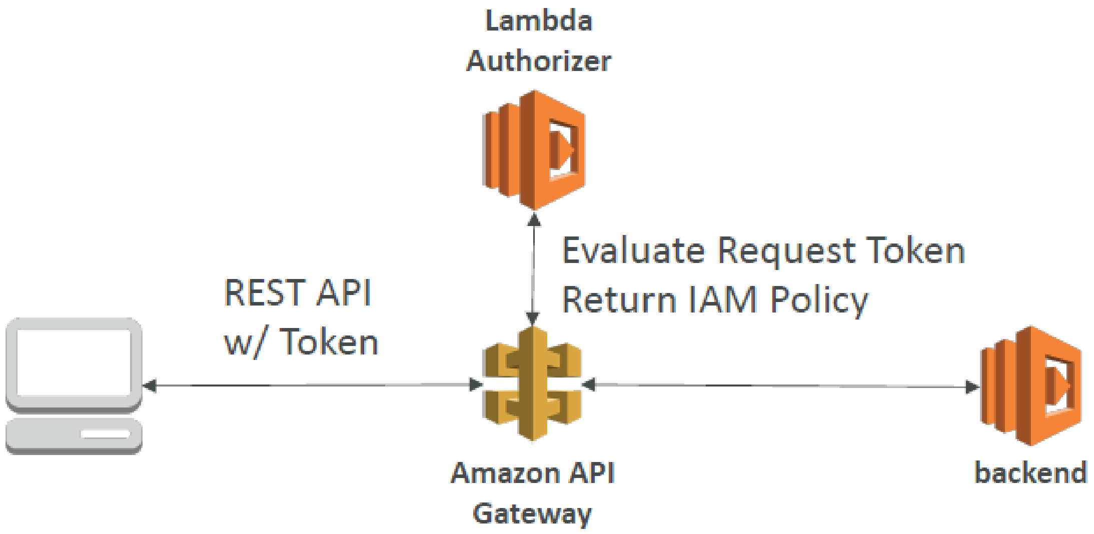
API Gateway – Security
IAM Permissions

- Create an IAM policy authorization and attach to User / Role
- API Gateway verifies IAM permissions passed by the calling application
- Good to provide access within your own infrastructure
- Leverages "Sig v4" capability where IAM credential are in headers



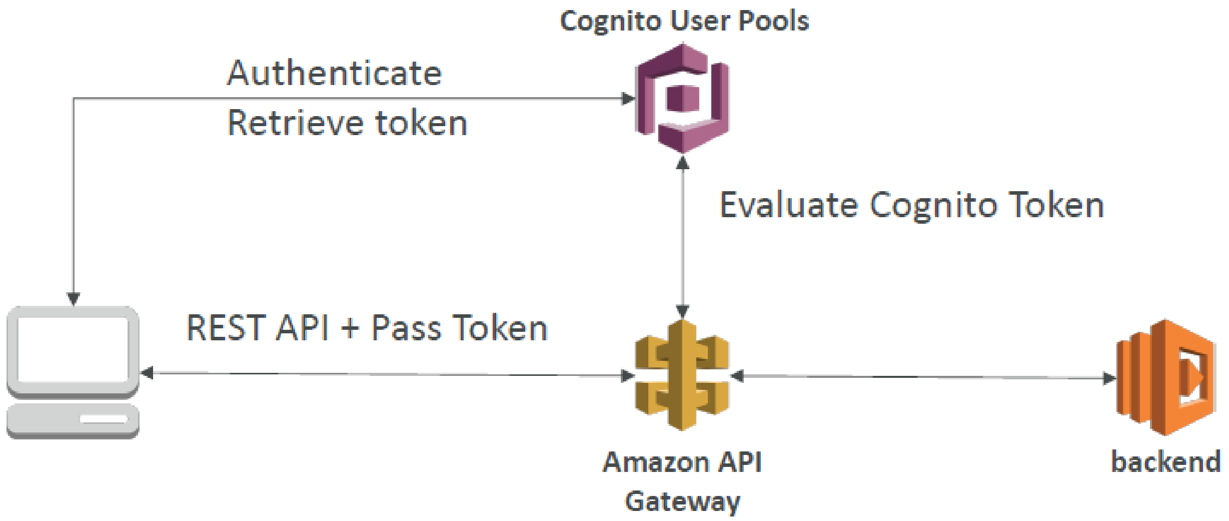
API Gateway – Security
Lambda Authorizer (formerly Custom Authorizers)

- Uses AWS Lambda to validate the token in header being passed
- Option to cache result of authentication
- Helps to use OAuth / SAML / 3rd party type of authentication
- Lambda must return an IAM policy for the user



API Gateway – Security
Cognito User Pools

- Cognito fully manages user lifecycle
- API gateway verifies identity automatically from AWS Cognito
- No custom implementation required
- Cognito only helps with authentication, not authorization



API Gateway – Security – Summary

- **IAM:**
  - Great for users / roles already within your AWS account
  - Handle authentication + authorization
  - Leverages Sig v4

- **Custom Authorizer:**
  - Great for 3rd party tokens
  - Very flexible in terms of what IAM policy is returned
  - Handle Authentication + Authorization
  - Pay per Lambda invocation

- **Cognito User Pool:**
  - You manage your own user pool (can be backed by Facebook, Google login etc…)
  - No need to write any custom code
  - Must implement authorization in the backend

AWS Cognito

- We want to give our users an identity so that they can interact with our application.

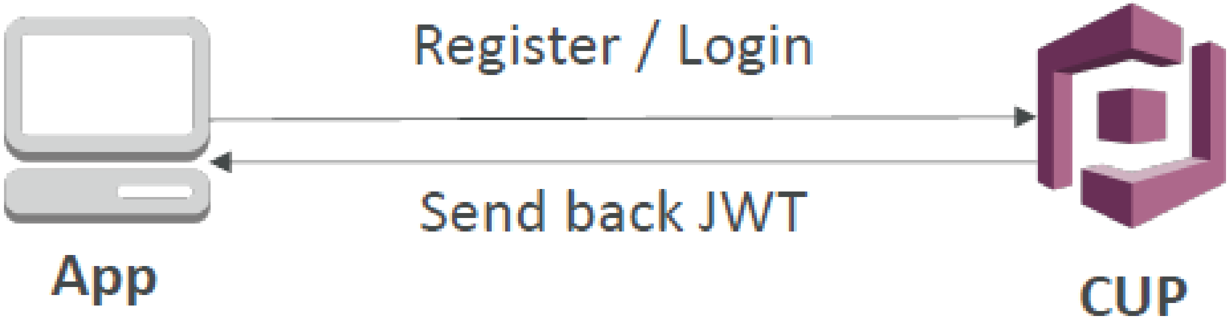
- **Cognito User Pools:**
  - Sign in functionality for app users
  - Integrate with API Gateway

- **Cognito Identity Pools (Federated Identity):**
  - Provide AWS credentials to users so they can access AWS resources directly
  - Integrate with Cognito User Pools as an identity provider

- **Cognito Sync:**
  - Synchronize data from device to Cognito.
  - May be deprecated and replaced by AppSync

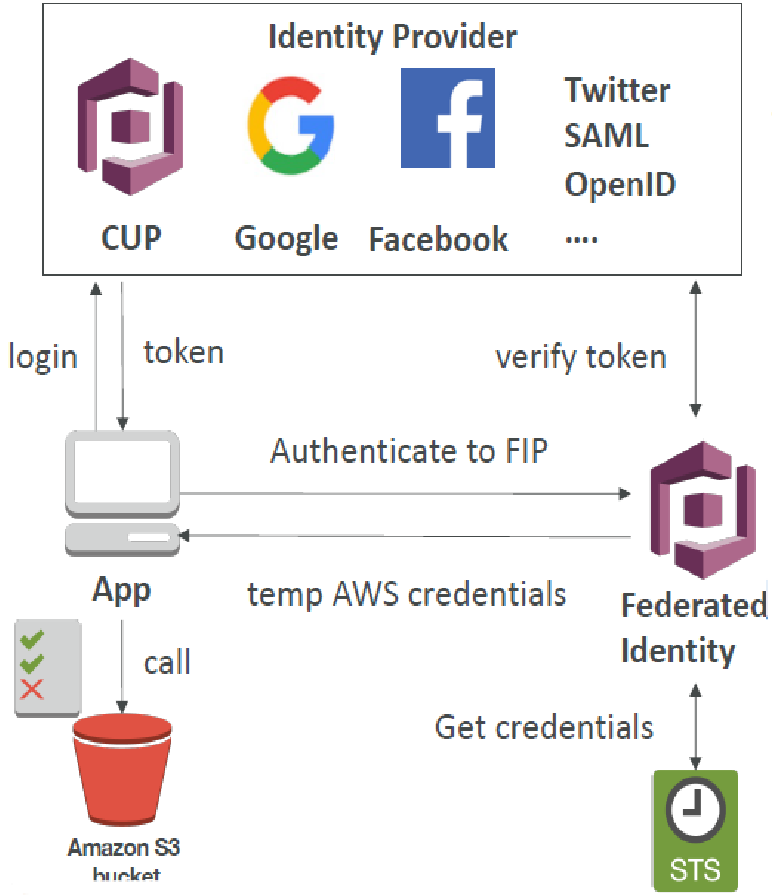
AWS Cognito User Pools (CUP)

- Create a serverless database of user for your mobile apps
- Simple login: Username (or email) / password combination
- Possibility to verify emails / phone numbers and **add MFA**
- Can enable Federated Identities (Facebook, Google, SAML…)
- Sends back a JSON Web Tokens (JWT)
- **Can be integrated with API Gateway for authentication**



AWS Cognito – Federated Identity Pools

- **Goal:**
  - Provide direct access to AWS Resources from the Client Side
- **How:**
  - Log in to federated identity provider – or remain anonymous
  - Get temporary AWS credentials back from
  - the Federated Identity Pool
  - These credentials come with a pre-defined
  - IAM policy stating their permissions
- **Example:**
  - provide (temporary) access to write to S3
  - bucket using Facebook Login



DICE ANALYTICS

AWS Cognito Sync

- **Deprecated – use AWS AppSync now**

- Store preferences, configuration, state of app
- Cross device synchronization (any platform – iOS, Android, etc…)
- Offline capability (synchronization when back online)
- **Requires Federated Identity Pool in Cognito (not User Pool)**
- Store data in datasets (up to 1MB)
- Up to 20 datasets to synchronise

AWS SAM - Serverless Application Model

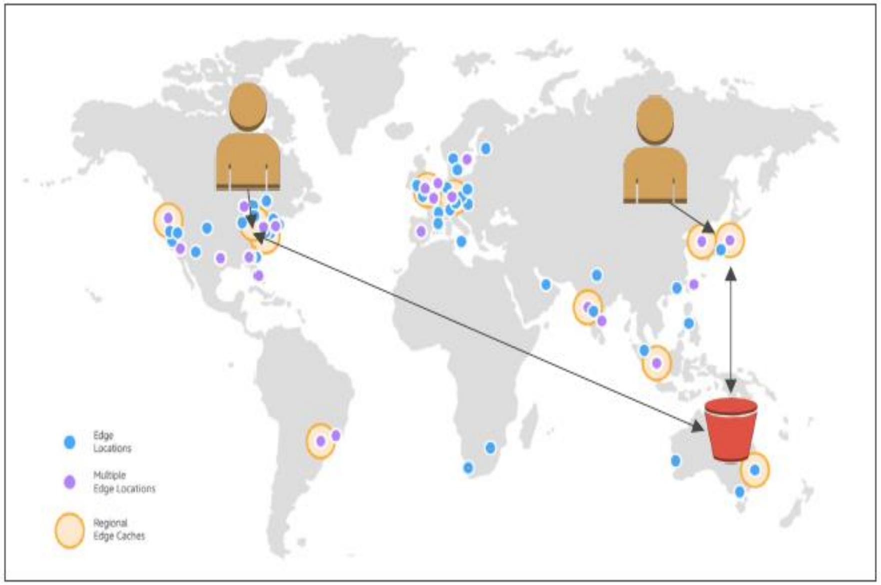
- **SAM = Serverless Application Model**
- Framework for developing and deploying serverless applications
- All the configuration is YAML code
    - Lambda Functions
    - DynamoDB tables
    - API Gateway
    - Cognito User Pools

- SAM can help you to run Lambda, API Gateway, DynamoDB locally
- SAM can use CodeDeploy to deploy Lambda functions

AWS CloudFront

AWS CloudFront

- Content Delivery Network (CDN)
- Improves read performance, content is cached at the edge
- 216 Point of Presence globally (edge locations)
- DDoS protection, integration with Shield, AWS Web Application Firewall
- Can expose external HTTPS and can talk to internal HTTPS backends

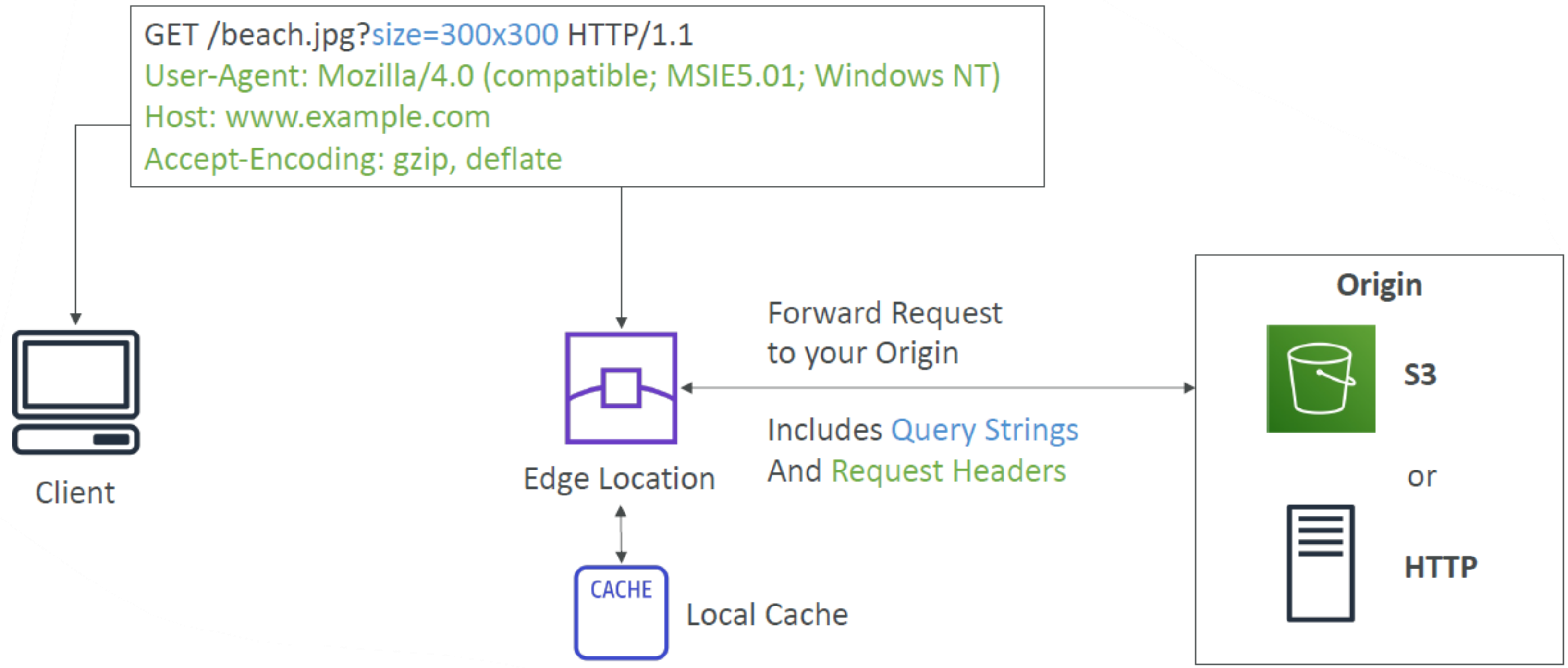


Source: https://aws.amazon.com/cloudfront/features/?nc=sn&loc=2

CloudFront – Origins

- **S3 bucket**
  - For distributing files and caching them at the edge
  - Enhanced security with CloudFront Origin Access Identity (OAI)
  - CloudFront can be used as an ingress (to upload files to S3)
- **Custom Origin (HTTP)**
  - Application Load Balancer
  - EC2 instance
  - S3 website (must first enable the bucket as a static S3 website)
  - Any HTTP backend you want

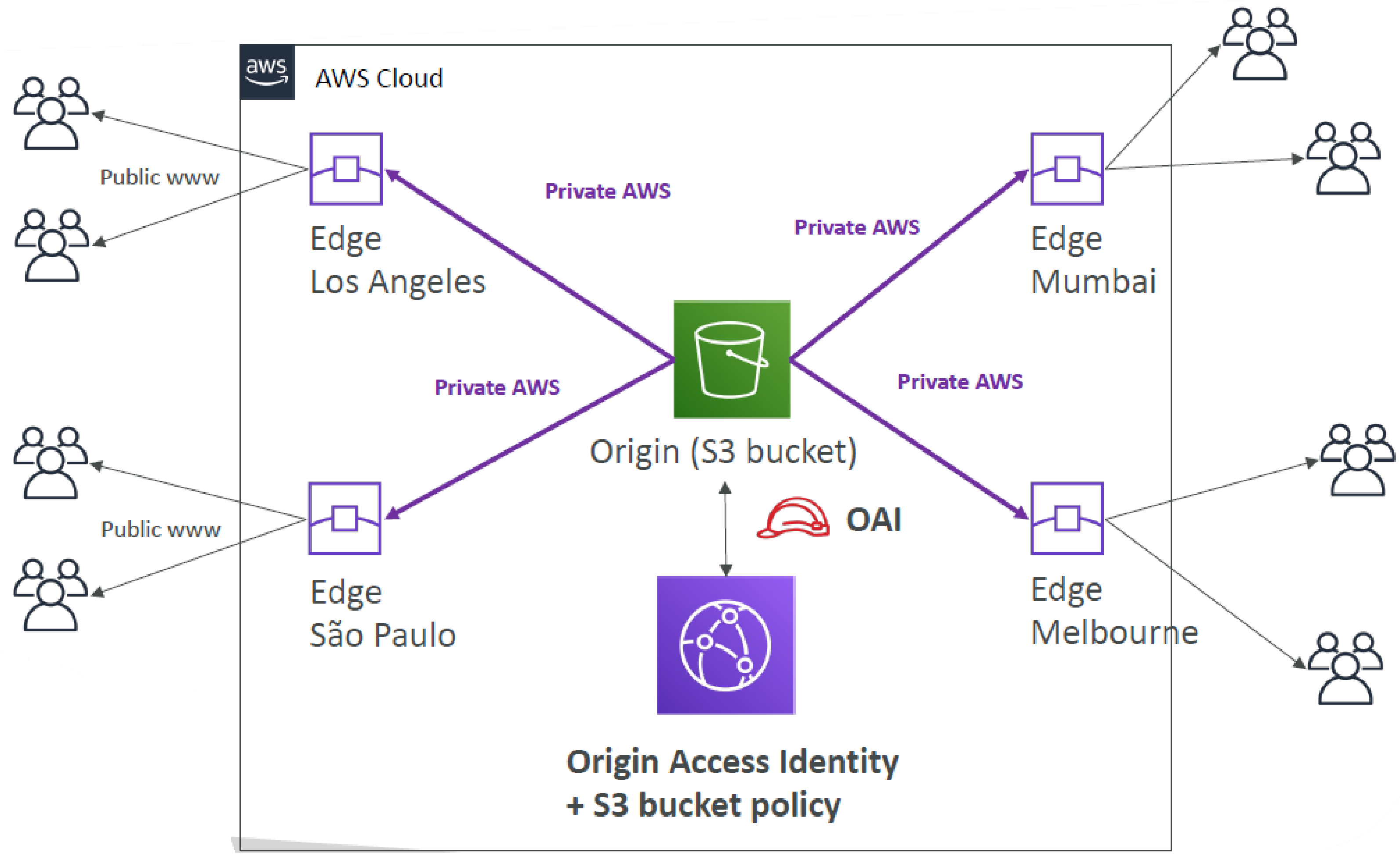
CloudFront at a high level



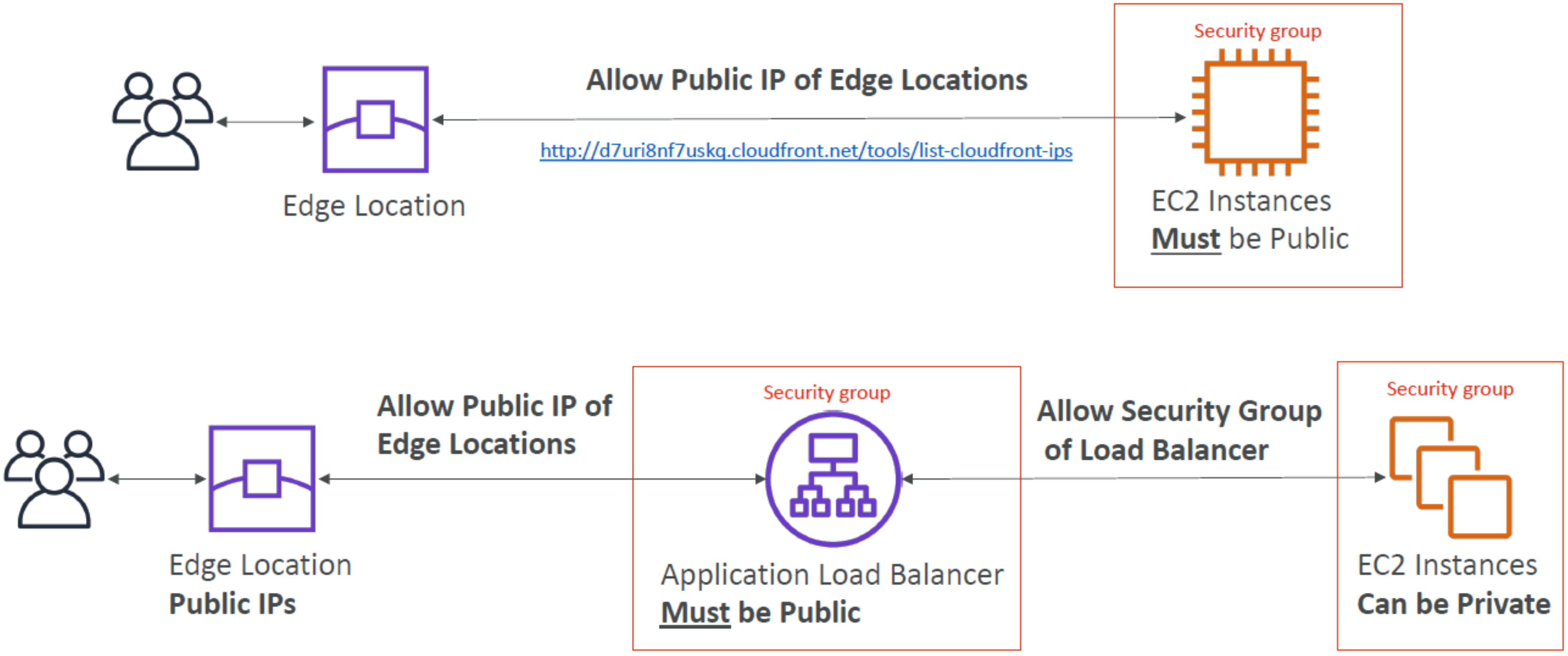
GET /beach.jpg?size=300x300 HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.example.com
Accept-Encoding: gzip, deflate

Client
Edge Location
Local Cache
CACHE
Forward Request to your Origin
Includes Query Strings And Request Headers
Origin
S3
or
HTTP

CloudFront – S3 as an Origin



CloudFront – ALB or EC2 as an origin



CloudFront Geo Restriction

- You can restrict who can access your distribution
  - **Whitelist:** Allow your users to access your content only if they're in one of the countries on a list of approved countries.
  - **Blacklist:** Prevent your users from accessing your content if they're in one of the countries on a blacklist of banned countries.
- The “country” is determined using a 3rd party Geo-IP database
- Use case: Copyright Laws to control access to content

CloudFront vs S3 Cross Region Replication

- CloudFront:
  - Global Edge network
  - Files are cached for a TTL (maybe a day)
  - **Great for static content that must be available everywhere**

- S3 Cross Region Replication:
  - Must be setup for each region you want replication to happen
  - Files are updated in near real-time
  - Read only
  - **Great for dynamic content that needs to be available at low-latency in few regions**

AWS CloudFront Hands On

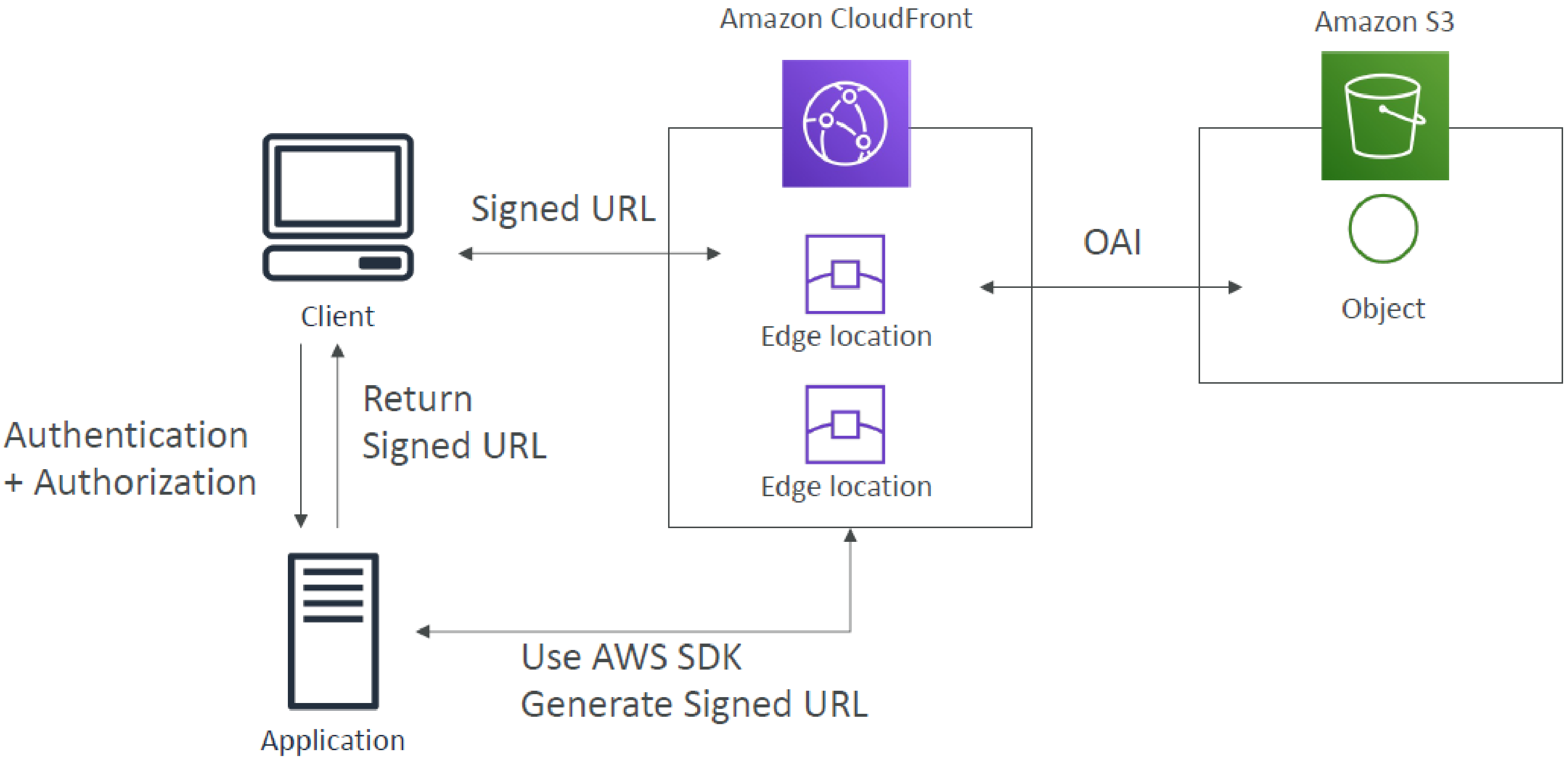
- We’ll create an S3 bucket
- We’ll create a CloudFront distribution
- We’ll create an Origin Access Identity
- We’ll limit the S3 bucket to be accessed only using this identity

CloudFront Signed URL / Signed Cookies

- You want to distribute paid shared content to premium users over the world
- We can use CloudFront Signed URL / Cookie. We attach a policy with:
  - Includes URL expiration
  - Includes IP ranges to access the data from
  - Trusted signers (which AWS accounts can create signed URLs)
- How long should the URL be valid for?
  - Shared content (movie, music): make it short (a few minutes)
  - Private content (private to the user): you can make it last for years

- Signed URL = access to individual files (one signed URL per file)
- Signed Cookies = access to multiple files (one signed cookie for many files)

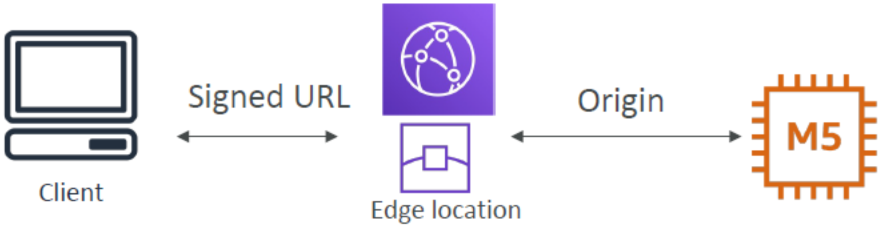
CloudFront Signed URL Diagram

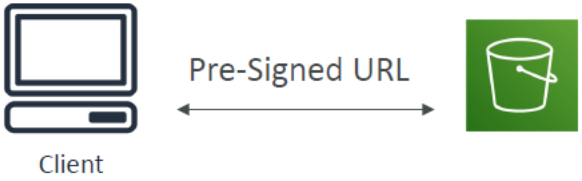


CloudFront Signed URL vs S3 Pre-Signed URL

- CloudFront Signed URL:
  - Allow access to a path, no matter the origin
  - Account wide key-pair, only the root can manage it
  - Can filter by IP, path, date, expiration
  - Can leverage caching features

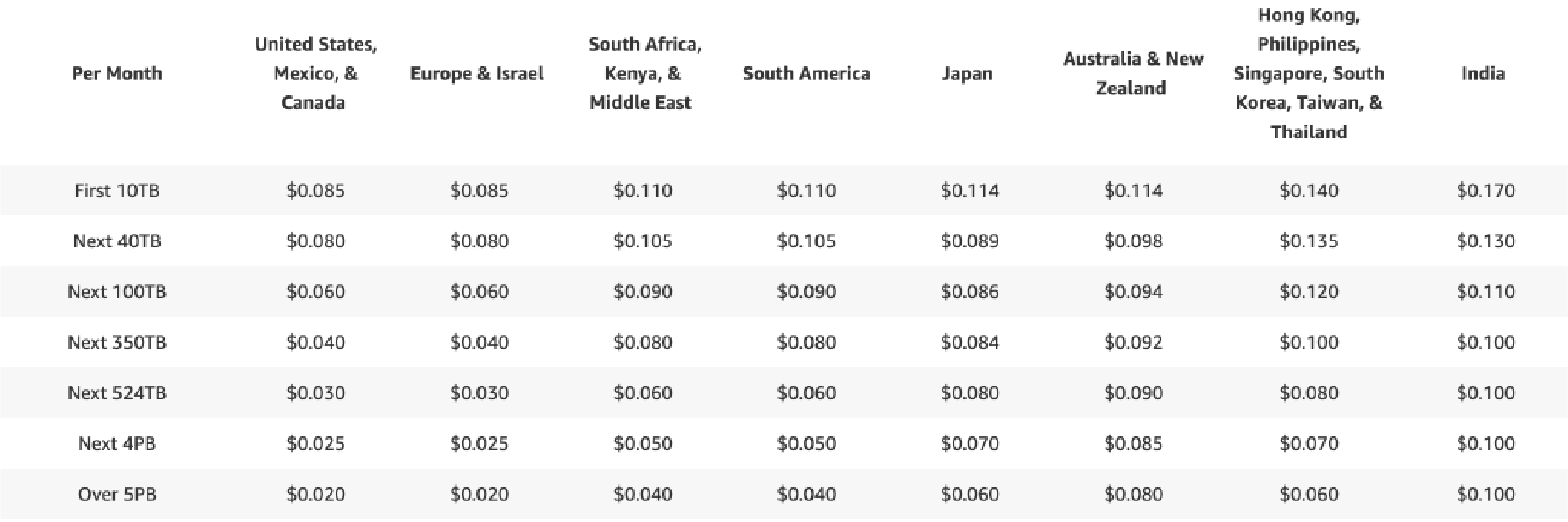
- S3 Pre-Signed URL:
  - Issue a request as the person who pre-signed the URL
  - Uses the IAM key of the signing IAM principal
  - Limited lifetime





CloudFront - Pricing

- CloudFront Edge locations are all around the world
- The cost of data out per edge location varies

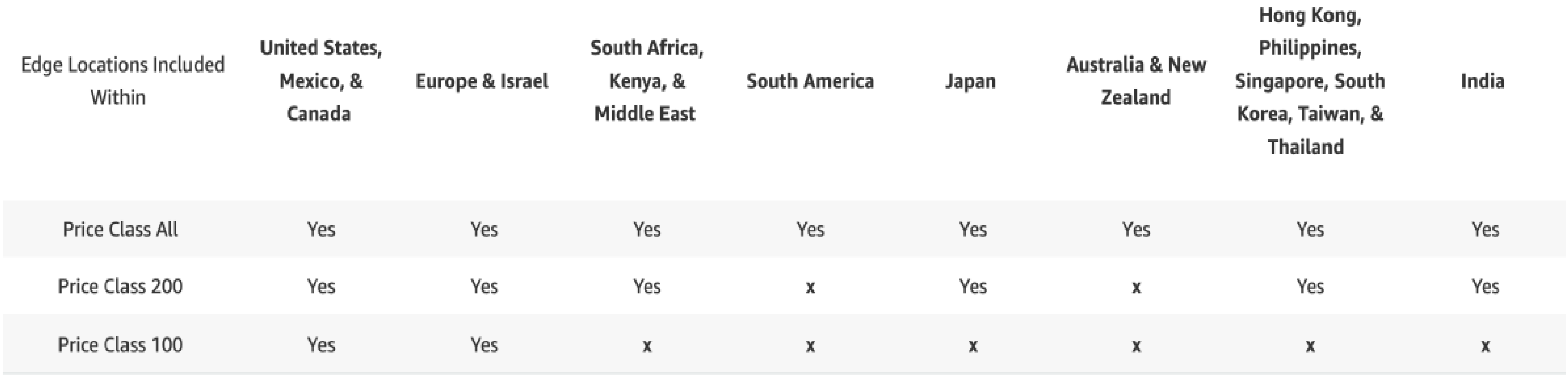


Per Month	United States, Mexico, & Canada	Europe & Israel	South Africa, Kenya, & Middle East	South America	Japan	Australia & New Zealand	Hong Kong, Philippines, Singapore, South Korea, Taiwan, & Thailand	India
First 10TB	$0.085	$0.085	$0.110	$0.110	$0.114	$0.114	$0.140	$0.170
Next 40TB	$0.080	$0.080	$0.105	$0.105	$0.089	$0.098	$0.135	$0.130
Next 100TB	$0.060	$0.060	$0.090	$0.090	$0.086	$0.094	$0.120	$0.110
Next 350TB	$0.040	$0.040	$0.080	$0.080	$0.084	$0.092	$0.100	$0.100
Next 524TB	$0.030	$0.030	$0.060	$0.060	$0.080	$0.090	$0.080	$0.100
Next 4PB	$0.025	$0.025	$0.050	$0.050	$0.070	$0.085	$0.070	$0.100
Over 5PB	$0.020	$0.020	$0.040	$0.040	$0.060	$0.080	$0.060	$0.100

lower → higher

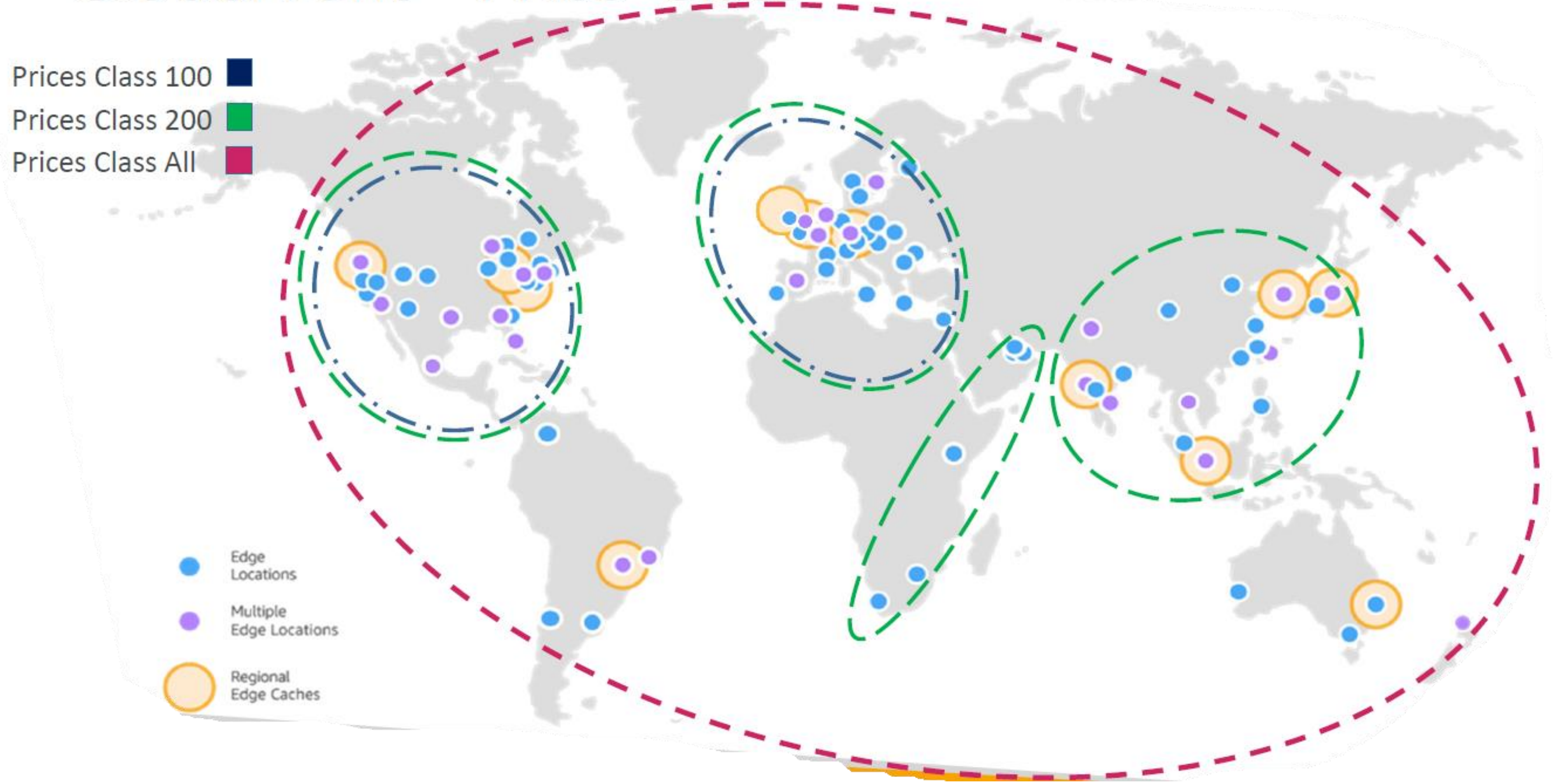
CloudFront – Price Classes

- You can reduce the number of edge locations for cost reduction
- Three price classes:
  1. Price Class All: all regions – best performance
  2. Price Class 200: most regions, but excludes the most expensive regions
  3. Price Class 100: only the least expensive regions



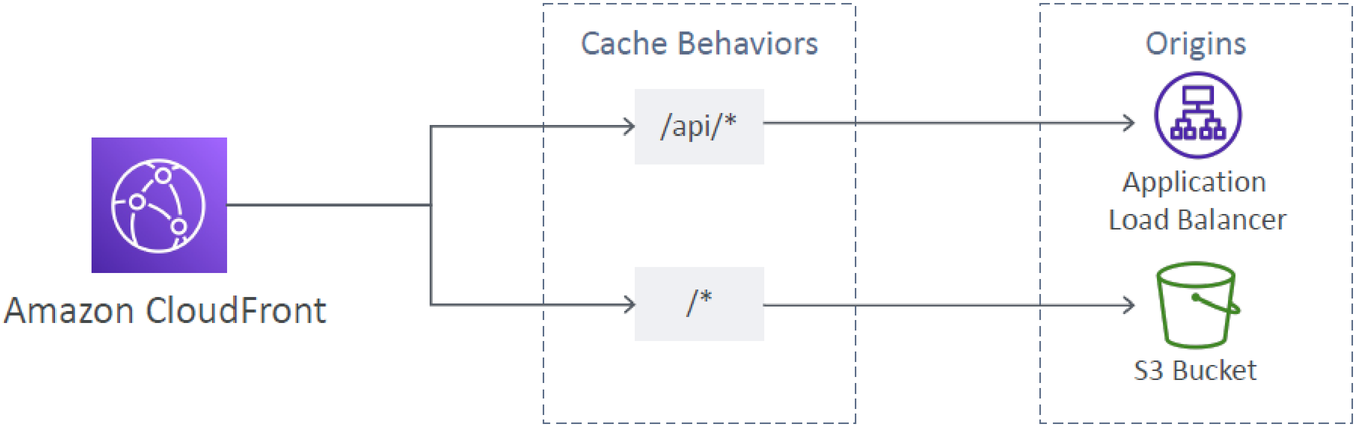
Edge Locations Included Within	United States, Mexico, & Canada	Europe & Israel	South Africa, Kenya, & Middle East	South America	Japan	Australia & New Zealand	Hong Kong, Philippines, Singapore, South Korea, Taiwan, & Thailand	India
Price Class All	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Price Class 200	Yes	Yes	Yes	x	Yes	x	Yes	Yes
Price Class 100	Yes	Yes	x	x	x	x	x	x

CloudFront - Price Class



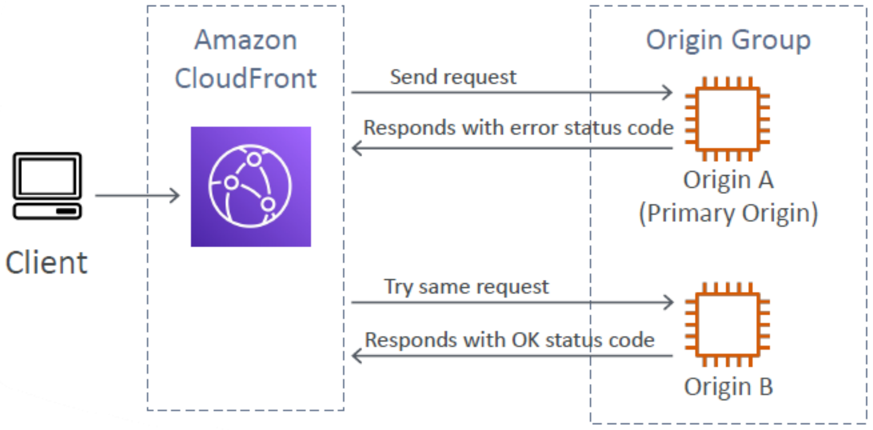
CloudFront – Multiple Origin

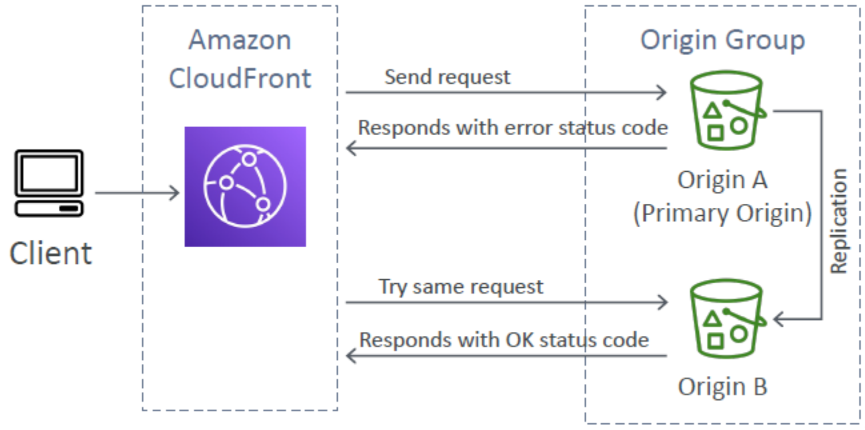
- To route to different kind of origins based on the content type
- Based on path pattern:
  - /images/*
  - /api/*
  - /*



CloudFront – Origin Groups

- To increase high-availability and do failover
- Origin Group: one primary and one secondary origin
- If the primary origin fails, the second one is used

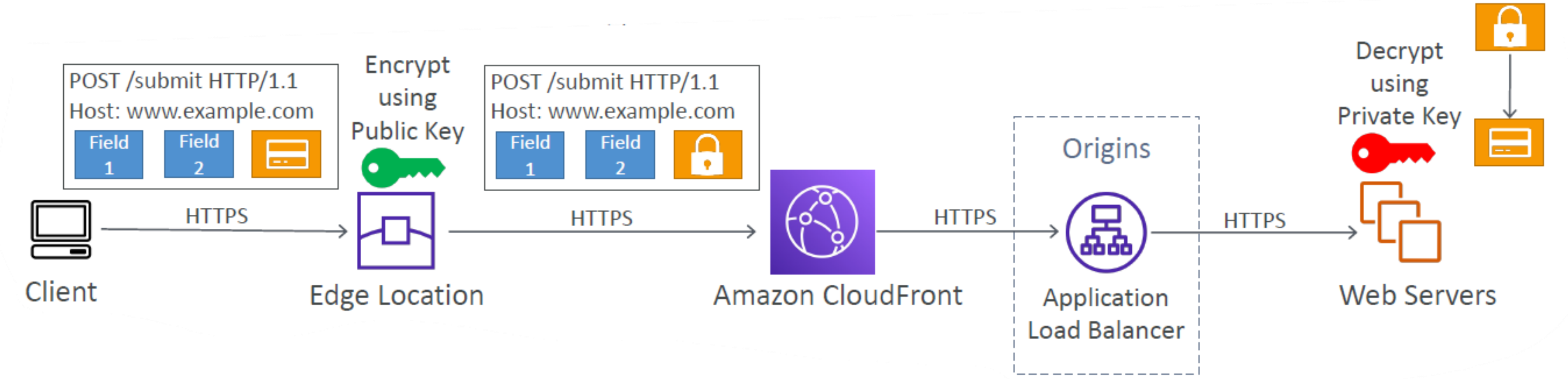




S3 + CloudFront – Region-level High Availability

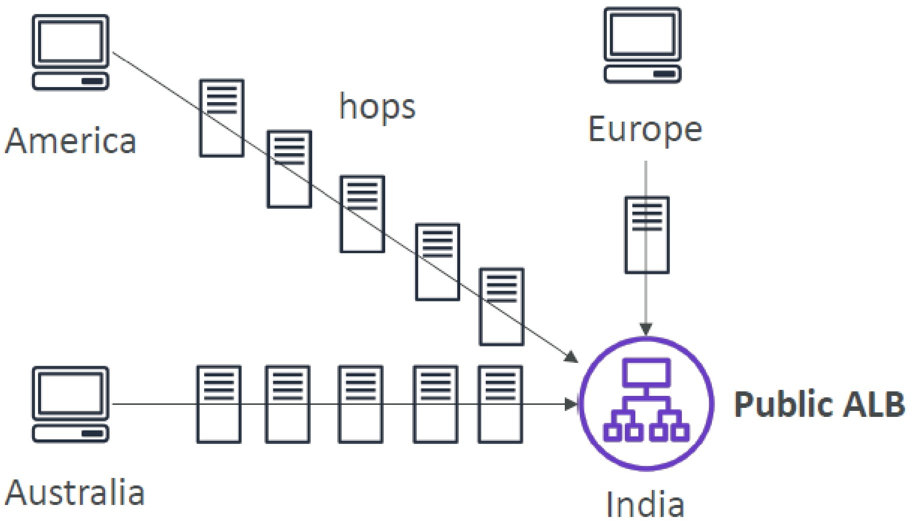
CloudFront – Field Level Encryption

- Protect user sensitive information through application stack
- Adds an additional layer of security along with HTTPS
- Sensitive information encrypted at the edge close to user
- Uses asymmetric encryption
- Usage:
    - Specify set of fields in POST requests that you want to be encrypted (up to 10 fields)
    - Specify the public key to encrypt them



Global users for our application

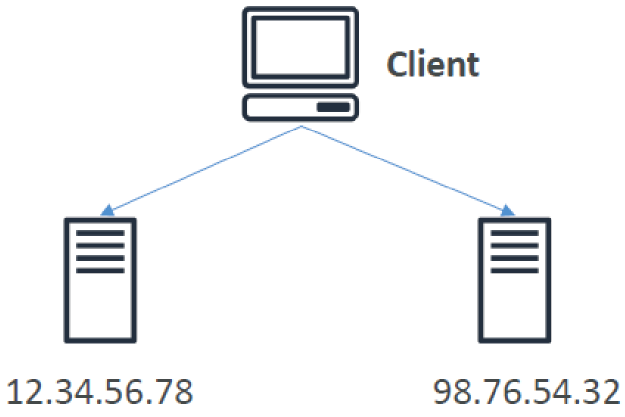
- You have deployed an application and have global users who want to access it directly.
- They go over the public internet, which can add a lot of latency due to many hops
- We wish to go as fast as possible through AWS network to minimize latency

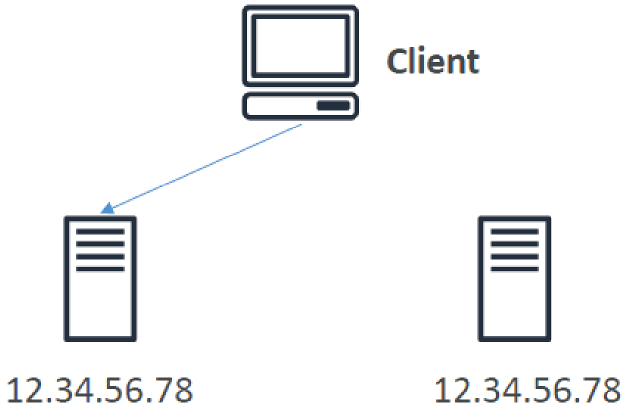


Unicast IP vs Anycast IP

- **Unicast IP:** one server holds one IP address

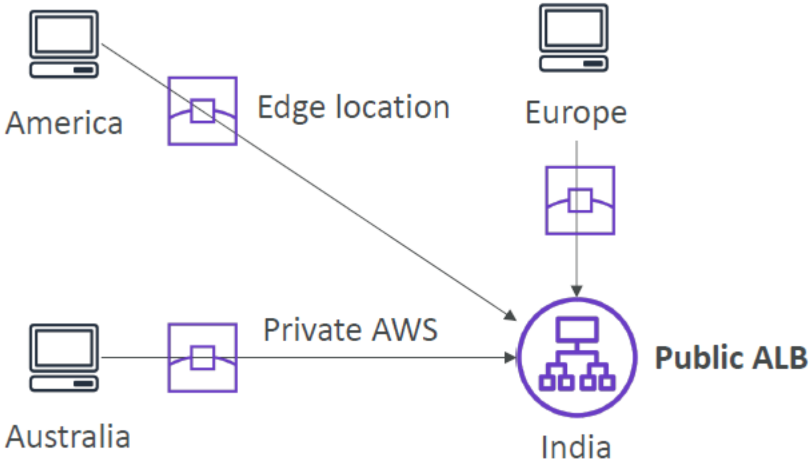
- **Anycast IP:** all servers hold the same IP address and the client is routed to the nearest one





AWS Global Accelerator

- Leverage the AWS internal network to route to your application
- **2 Anycast IP** are created for your application
- The Anycast IP send traffic directly to Edge Locations
- The Edge locations send the traffic to your application



AWS Global Accelerator

- Works with **Elastic IP, EC2 instances, ALB, NLB, public or private**
- Consistent Performance
  - Intelligent routing to lowest latency and fast regional failover
  - No issue with client cache (because the IP doesn’t change)
  - Internal AWS network

- Health Checks
  - Global Accelerator performs a health check of your applications
  - Helps make your application global (failover less than 1 minute for unhealthy)
  - Great for disaster recovery (thanks to the health checks)

- Security
  - only 2 external IP need to be whitelisted
  - DDoS protection thanks to AWS Shield

AWS Global Accelerator vs CloudFront

- They both use the AWS global network and its edge locations around the world
- Both services integrate with AWS Shield for DDoS protection.

- **CloudFront**
  - Improves performance for both cacheable content (such as images and videos)
  - Dynamic content (such as API acceleration and dynamic site delivery)
  - Content is served at the edge

- **Global Accelerator**
  - Improves performance for a wide range of applications over TCP or UDP
  - Proxying packets at the edge to applications running in one or more AWS Regions.
  - Good fit for non-HTTP use cases, such as gaming (UDP), IoT (MQTT), or Voice over IP
  - Good for HTTP use cases that require static IP addresses
  - Good for HTTP use cases that required deterministic, fast regional failover