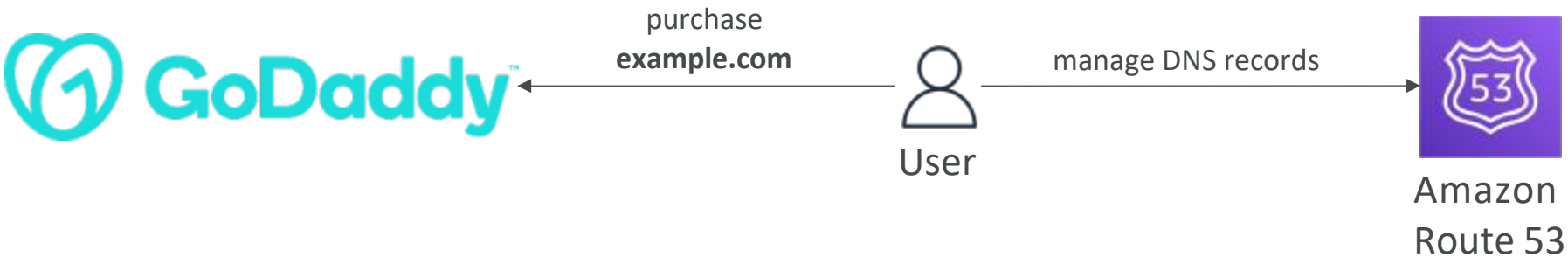
Routing Policies – Multi-Value

- Use when routing traffic to multiple resources
- Route 53 return multiple values/resources
- Can be associated with Health Checks (return only values for healthy resources)
- Up to 8 healthy records are returned for each Multi-Value query
- **Multi-Value is not a substitute for having an ELB**

Name	Type	Value	TTL	Set ID	Health Check
www.example.com	A Record	192.0.2.2	60	Web1	A
www.example.com	A Record	198.51.100.2	60	Web2	B
www.example.com	A Record	203.0.113.2	60	Web3	C

Domain Registar vs. DNS Service

- You buy or register your domain name with a Domain Registrar typically by paying annual charges (e.g., GoDaddy, Amazon Registrar Inc., ...)
- The Domain Registrar usually provides you with a DNS service to manage your DNS records
- But you can use another DNS service to manage your DNS records
- Example: purchase the domain from GoDaddy and use Route 53 to manage your DNS records



GoDaddy as Registrar & Route 53 as DNS Service

GoDaddy

Records

We can't display your DNS information because your nameservers aren't managed by us.

Nameservers

Using custom nameservers **Change**

Nameserver
ns-1083.awsdns-07.org
ns-932.awsdns-52.net
ns-1911.awsdns-46.co.uk
ns-481.awsdns-60.com

Amazon Route 53

Public Hosted Zone
stephanetheteacher.com

Hosted zone details — Edit hosted zone

Hosted zone ID	Type	Name servers
Z30IUJCCWPKZUV	Public hosted zone	ns-252.awsdns-31.com ns-1468.awsdns-55.org ns-633.awsdns-15.net ns-1800.awsdns-33.co.uk
Description HostedZone created by Route53 Registrar	Record count 22	
Query log		

3rd Party Registrar with Amazon Route 53

- If you buy your domain on a 3
  53 as the DNS Service provide

1. Create a Hosted Zone in Route 53
2. Update NS Records on 3rd party website to use Route 53 **Name Servers**

- **Domain Registrar != DNS Service**
- But every Domain Registrar usually comes with some DNS features

VPC Primer

Quick Overview of VPC

VPC – Crash Course

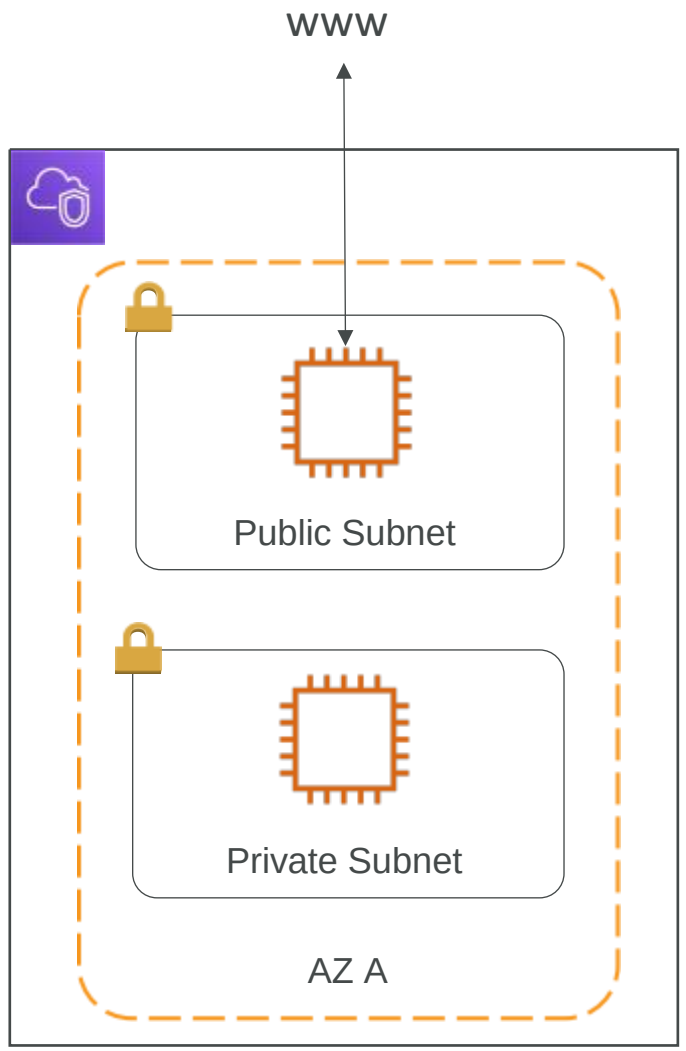
- VPC is something you should know in depth for the AWS Certified Solutions Architect Associate & AWS Certified SysOps Administrator

- **At the AWS Certified Developer Level, you should know about:**
  - VPC, Subnets, Internet Gateways & NAT Gateways
  - Security Groups, Network ACL (NACL), VPC Flow Logs
  - VPC Peering, VPC Endpoints
  - Site to Site VPN & Direct Connect

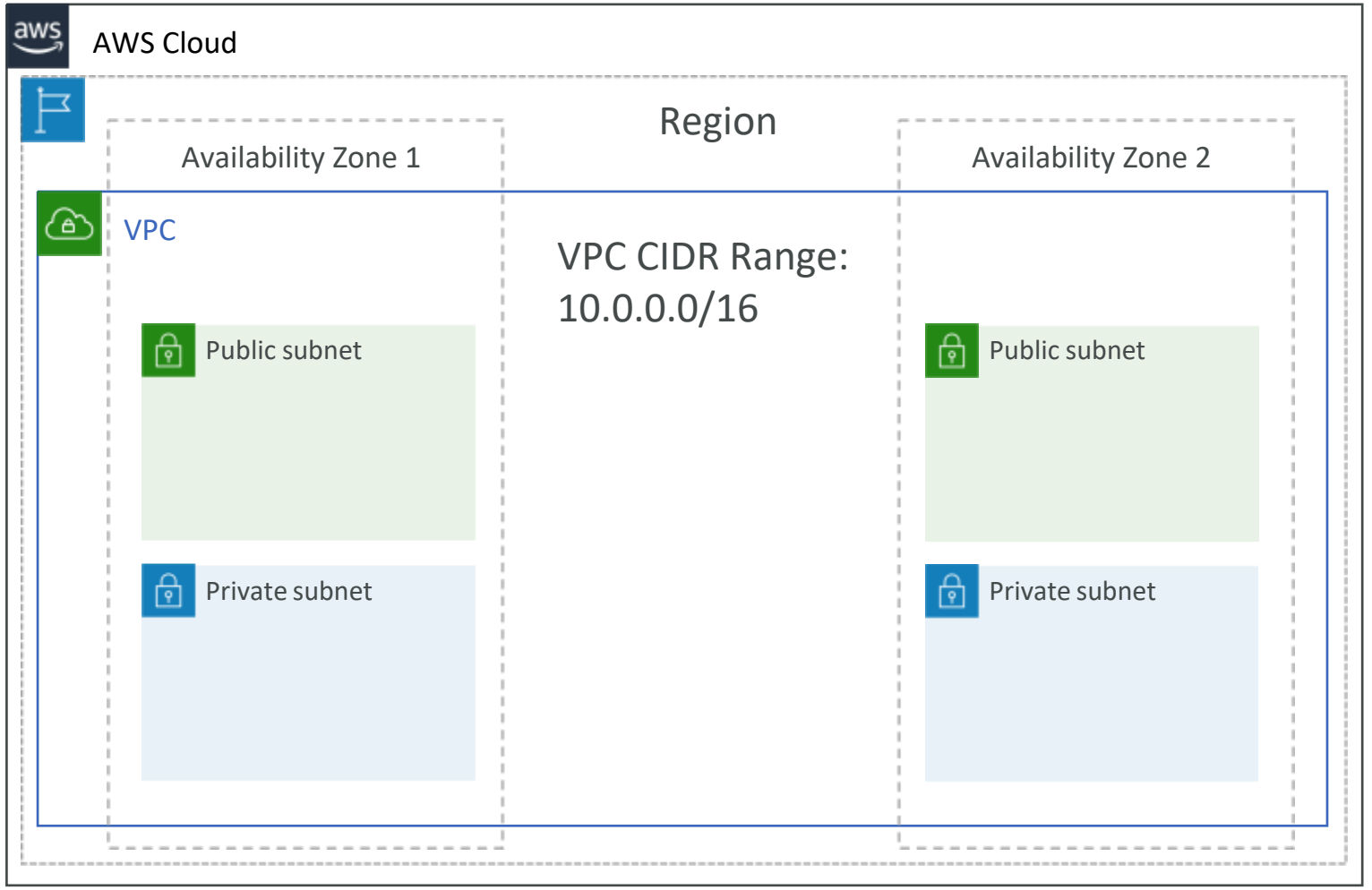
- I will just give you an overview, less than1 or 2 questions at your exam.
- Later in the course, I will be highlighting when VPC concepts are helpful

VPC & Subnets Primer

- **VPC**: private network to deploy your resources (regional resource)
- **Subnets** allow you to partition your network inside your VPC (Availability Zone resource)
- A **public subnet** is a subnet that is accessible from the internet
- A **private subnet** is a subnet that is not accessible from the internet
- To define access to the internet and between subnets, we use **Route Tables**.



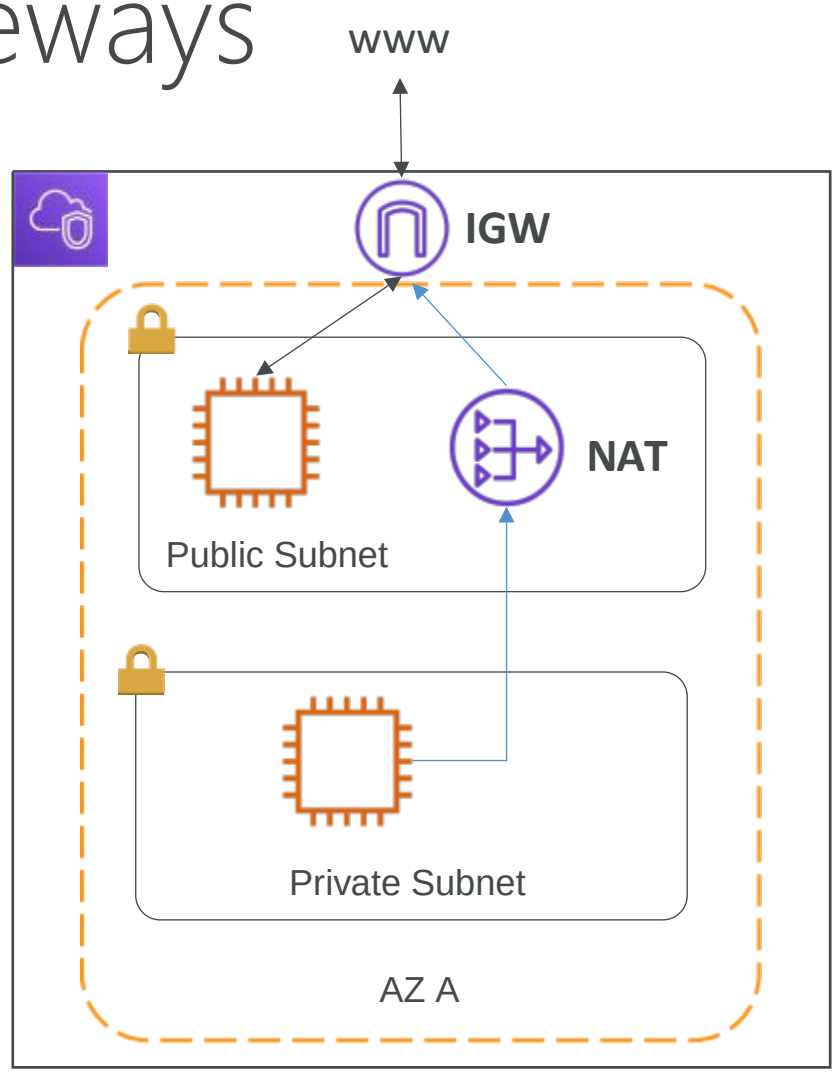
VPC Diagram


AWS Cloud
Region
Availability Zone 1
Availability Zone 2
VPC
VPC CIDR Range:
10.0.0.0/16
Public subnet
Private subnet
Public subnet
Private subnet

Internet Gateway & NAT Gateways

- **Internet Gateways** helps our VPC instances connect with the internet
- Public Subnets have a route to the internet gateway.

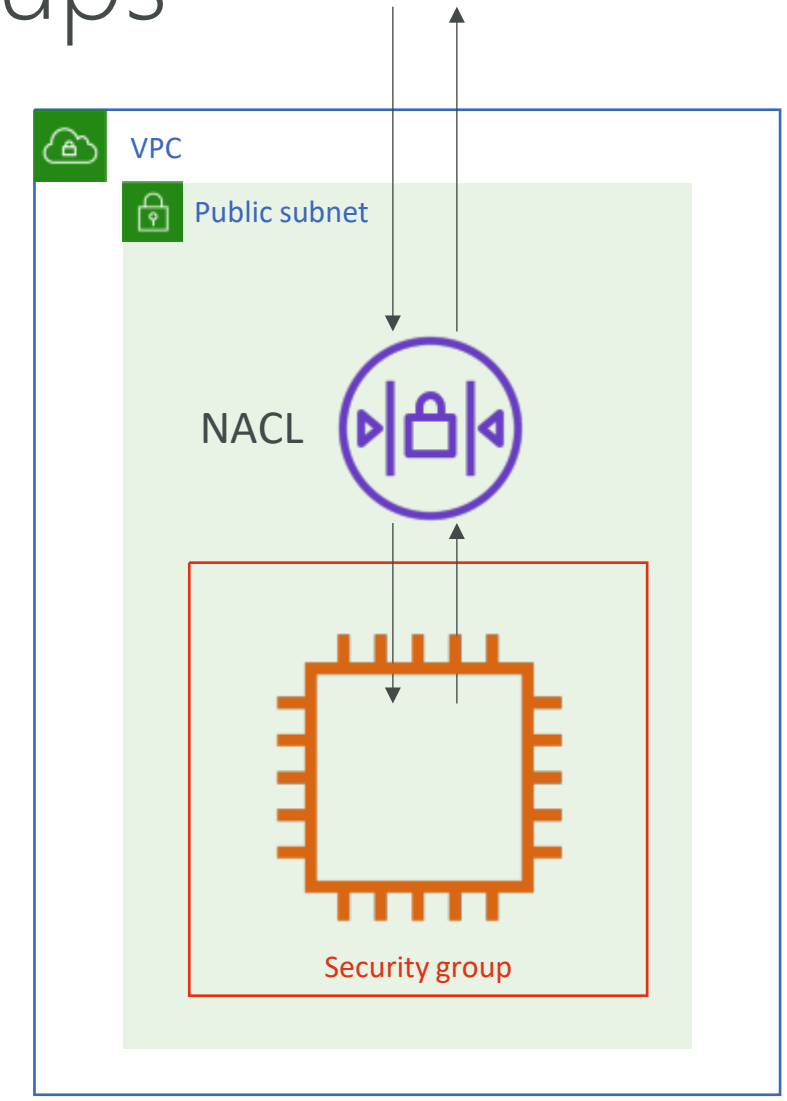
- **NAT Gateways** (AWS-managed) & **NAT Instances** (self-managed) allow your instances in your **Private Subnets** to access the internet while remaining private



Network ACL & Security Groups

- **NACL** (Network ACL)
  - A firewall which controls traffic from and to subnet
  - Can have ALLOW and DENY rules
  - Are attached at the **Subnet** level
  - Rules only include IP addresses

- **Security Groups**
  - A firewall that controls traffic to and from **an ENI / an EC2 Instance**
  - Can have only ALLOW rules
  - Rules include IP addresses and other security groups



Network ACLs vs Security Groups

Security Group	Network ACL
Operates at the instance level	Operates at the subnet level
Supports allow rules only	Supports allow rules and deny rules
Is stateful: Return traffic is automatically allowed, regardless of any rules	Is stateless: Return traffic must be explicitly allowed by rules
We evaluate all rules before deciding whether to allow traffic	We process rules in number order when deciding whether to allow traffic
Applies to an instance only if someone specifies the security group when launching the instance, or associates the security group with the instance later on	Automatically applies to all instances in the subnets it's associated with (therefore, you don't have to rely on users to specify the security group)

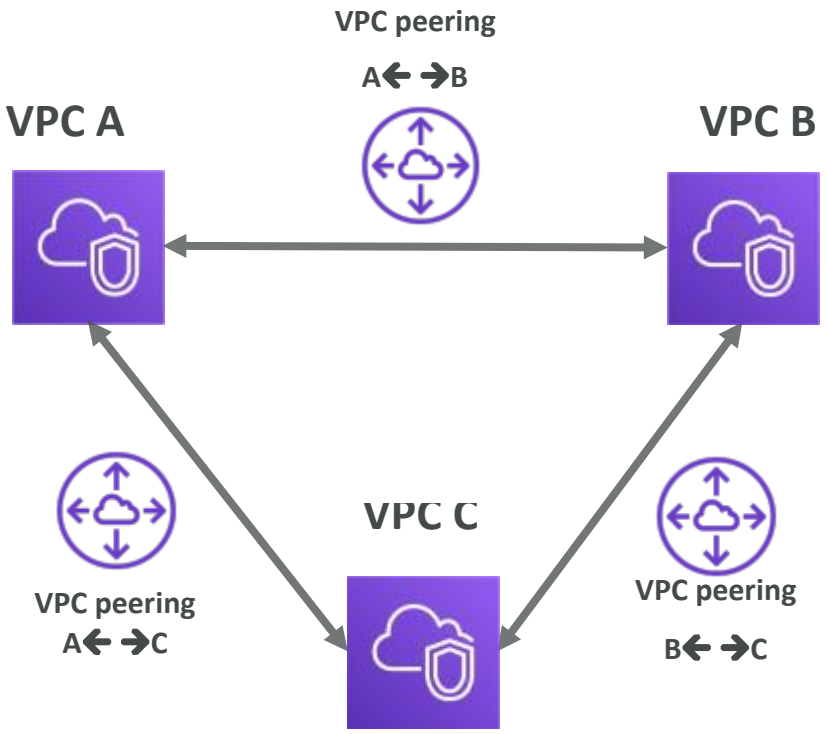
https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Security.html#VPC_Security_Comparison

VPC Flow Logs

- Capture information about IP traffic going into your interfaces:
  - **VPC** Flow Logs
  - **Subnet** Flow Logs
  - **Elastic Network Interface** Flow Logs
- Helps to monitor & troubleshoot connectivity issues. Example:
  - Subnets to internet
  - Subnets to subnets
  - Internet to subnets
- Captures network information from AWS managed interfaces too: Elastic Load Balancers, ElastiCache, RDS, Aurora, etc…
- VPC Flow logs data can go to S3 / CloudWatch Logs

VPC Peering

- Connect two VPC, privately using AWS' network
- Make them behave as if they were in the same network
- Must not have overlapping CIDR (IP address range)
- VPC Peering connection is **not transitive** (must be established for each VPC that need to communicate with one another)

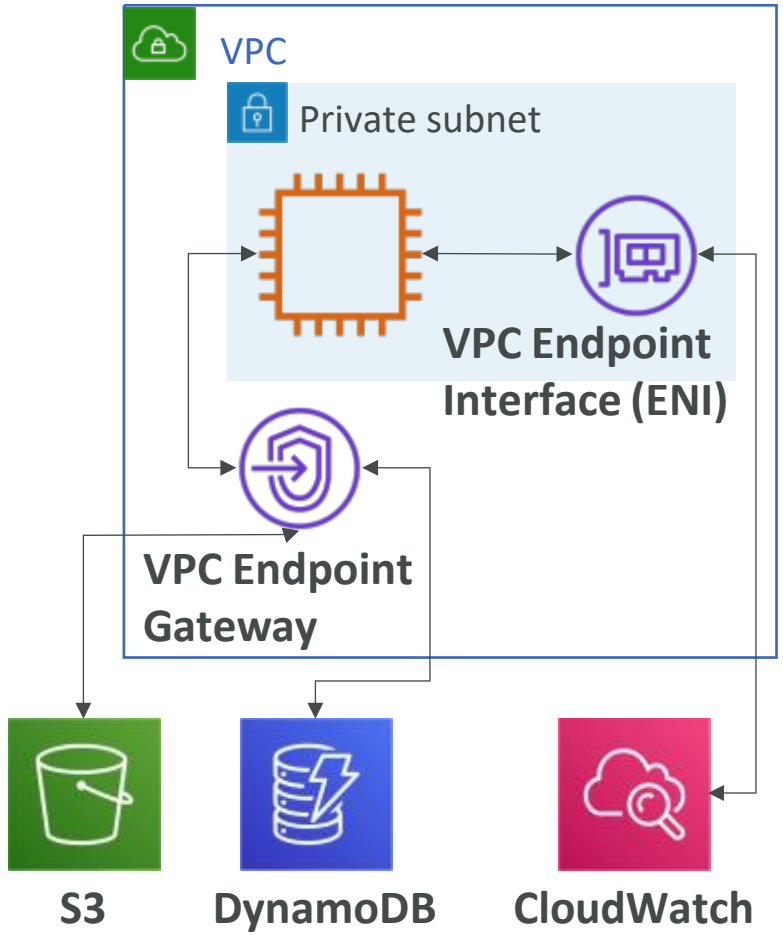


VPC Endpoints

- Endpoints allow you to connect to AWS Services **using a private network** instead of the public www network
- This gives you enhanced security and lower latency to access AWS services

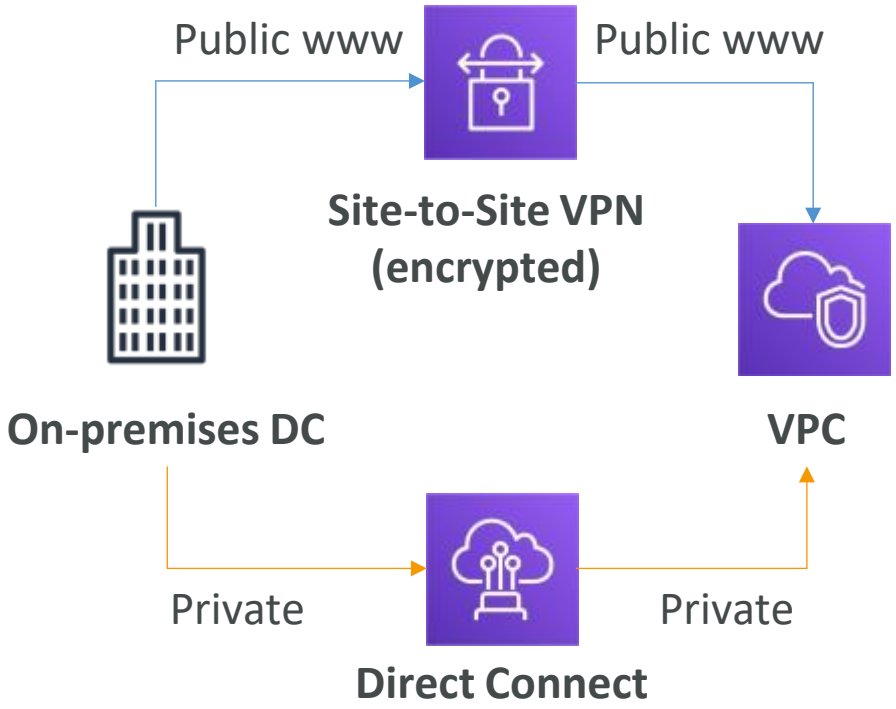
- VPC Endpoint Gateway: S3 & DynamoDB
- VPC Endpoint Interface: the rest

- Only used within your VPC



Site to Site VPN & Direct Connect

- **Site to Site VPN**
  - Connect an on-premises VPN to AWS
  - The connection is automatically encrypted
  - Goes over the public internet
- **Direct Connect (DX)**
  - Establish a physical connection between on-premises and AWS
  - The connection is private, secure and fast
  - Goes over a private network
  - Takes at least a month to establish
- **Note:** Site-to-site VPN and Direct Connect cannot access VPC endpoints



VPC Closing Comments

- **VPC:** Virtual Private Cloud
- **Subnets:** Tied to an AZ, network partition of the VPC
- **Internet Gateway:** at the VPC level, provide Internet Access
- **NAT Gateway / Instances:** give internet access to private subnets
- **NACL:** Stateless, subnet rules for inbound and outbound
- **Security Groups:** Stateful, operate at the EC2 instance level or ENI
- **VPC Peering:** Connect two VPC with non overlapping IP ranges, non transitive
- **VPC Endpoints:** Provide private access to AWS Services within VPC
- **VPC Flow Logs:** network traffic logs
- **Site to Site VPN:** VPN over public internet between on-premises DC and AWS
- **Direct Connect:** direct private connection to a AWS

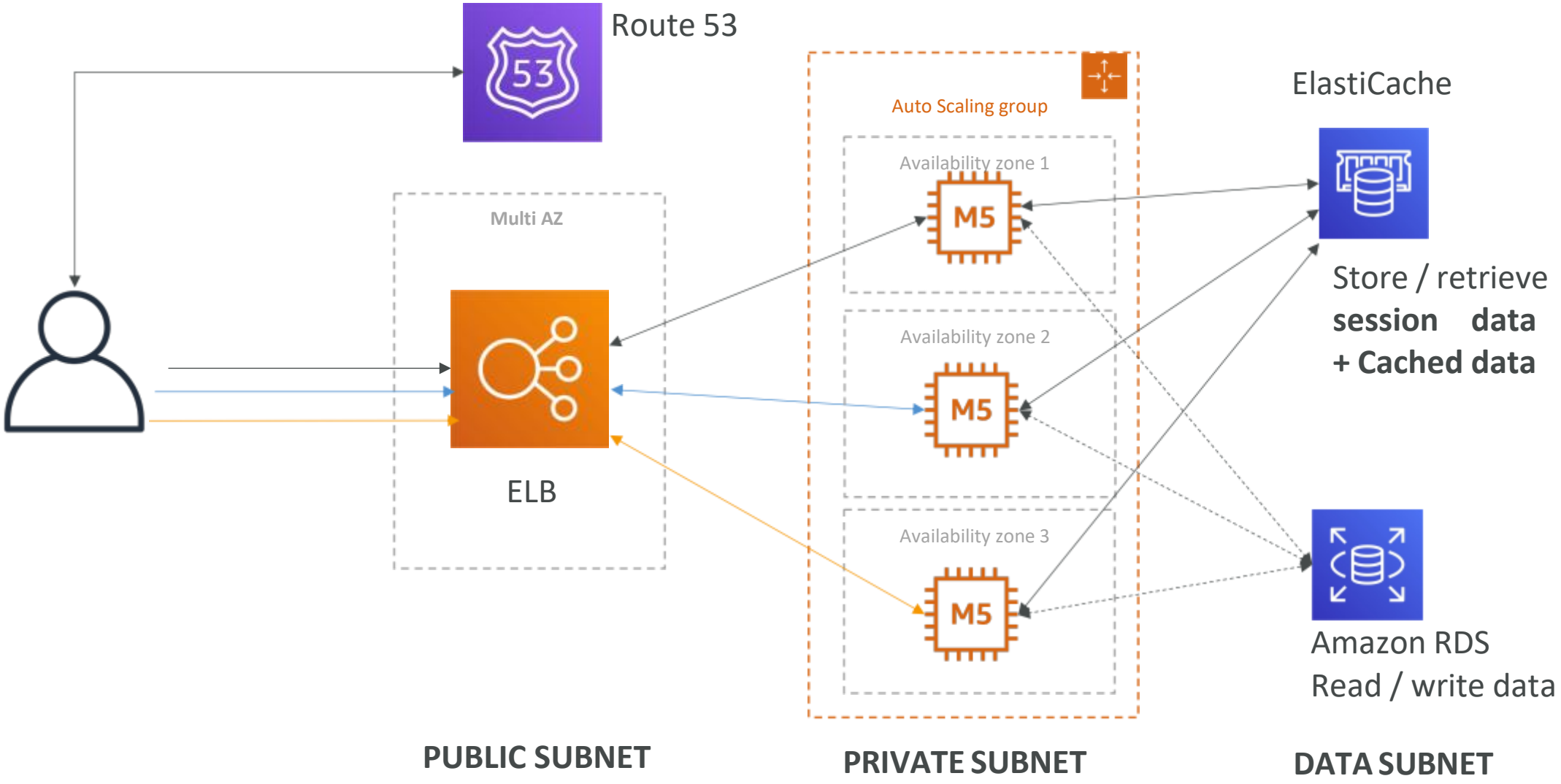
VPC note – AWS Certified Developer

- Don’t stress if you didn’t understand everything in that section
- I will be highlighting in the course the specific VPC features we need

- Feel free to revisit that section after you’re done in the course !

- Moving on ☺

Typical 3 tier solution architecture

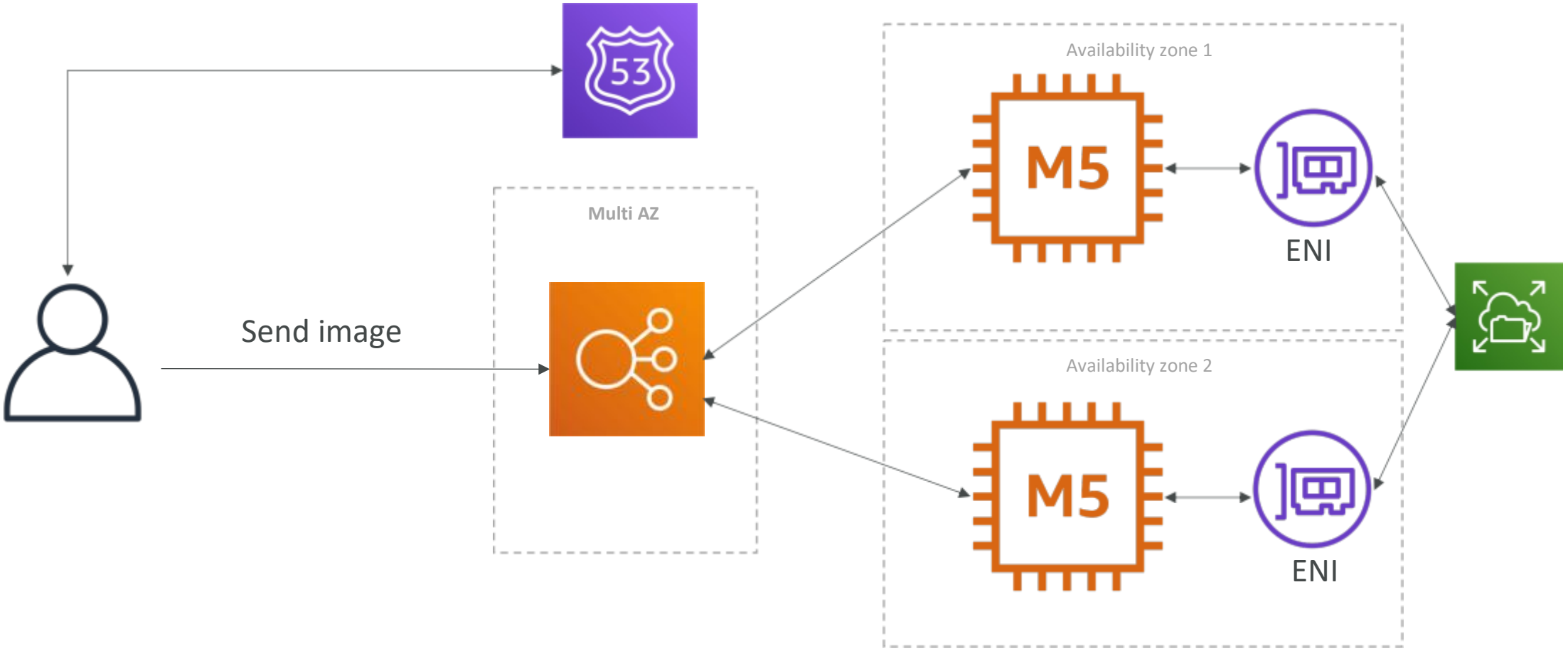


LAMP Stack on EC2

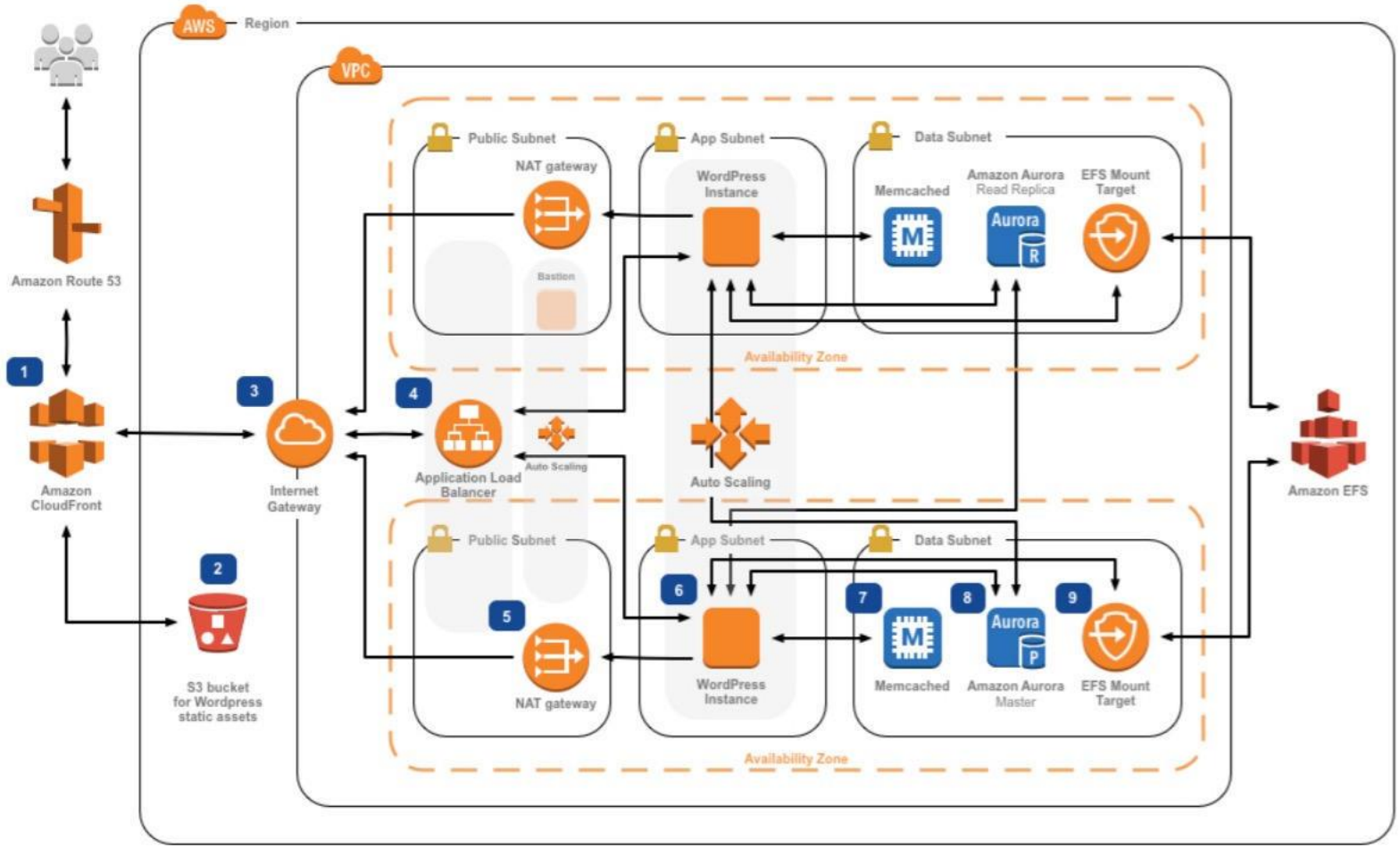
- **L**inux: OS for EC2 instances
- **A**pache: Web Server that run on Linux (EC2)
- **M**ySQL: database on RDS
- **P**HP: Application logic (running on EC2)

- Can add Redis / Memcached (**ElastiCache**) to include a caching tech
- To store local application data & software: **EBS** drive (root)

Wordpress on AWS



WordPress on AWS (more complicated)



https://aws.amazon.com/blogs/architecture/wordpress-best-practices-on-aws/

Amazon S3 Section

Section introduction

- Amazon S3 is one of the main building blocks of AWS
- It's advertised as "infinitely scaling" storage

- Many websites use Amazon S3 as a backbone
- Many AWS services use Amazon S3 as an integration as well

- We'll have a step-by-step approach to S3

Amazon S3 Use cases

- Backup and storage
- Disaster Recovery
- Archive
- Hybrid Cloud storage
- Application hosting
- Media hosting
- Data lakes & big data analytics
- Software delivery
- Static website

Nasdaq stores 7 years of data into S3 Glacier

Sysco runs analytics on its data and gain business insights

Amazon S3 - Buckets

- Amazon S3 allows people to store objects (files) in “buckets” (directories)
- Buckets must have a **globally unique name (across all regions all accounts)**
- Buckets are defined at the region level
- S3 looks like a global service but buckets are created in a region
- Naming convention
  - No uppercase, No underscore
  - 3-63 characters long
  - Not an IP
  - Must start with lowercase letter or number
  - Must NOT start with the prefix --
  - Must NOT end with the suffix -s3alias

S3 Bucket

Amazon S3 - Objects

- Objects (files) have a Key
- The **key** is the **FULL** path:
  - s3://my-bucket/**my_file.txt**
  - s3://my-bucket/**my_folder1/another_folder/my_file.txt**
- The key is composed of: prefix + object name
  - s3://my-bucket/**my_folder1/another_folder/my_file.txt**
- There's no concept of "directories" within buckets (although the UI will trick you to think otherwise)
- Just keys with very long names that contain slashes ("/")

Object

S3 Bucket with Objects

Amazon S3 – Objects (cont.)

- Object values are the content of the body:
  - Max. Object Size is 5TB (5000GB)
  - If uploading more than 5GB, must use “multi-part upload”

- Metadata (list of text key / value pairs – system or user metadata)
- Tags (Unicode key / value pair – up to 10) – useful for security / lifecycle
- Version ID (if versioning is enabled)

Amazon S3 – Security

- **User-Based**
  - **IAM Policies –** which API calls should be allowed for a specific user from IAM

- **Resource-Based**
  - Bucket Policies – bucket wide rules from the S3 console - allows cross account
  - Object Acces Control List (ACL) – finer grain (can be disabled)
  - Bucket Acces C less common (can be disabled)

- **Note:** an IAM principal can access an S3 object if
  - The user IAM permissions ALLOW it OR the resource policy ALLOWS it
  - AND there's no explicit DENY

- **Encryption:** encrypt objects in Amazon S3 using encryption keys

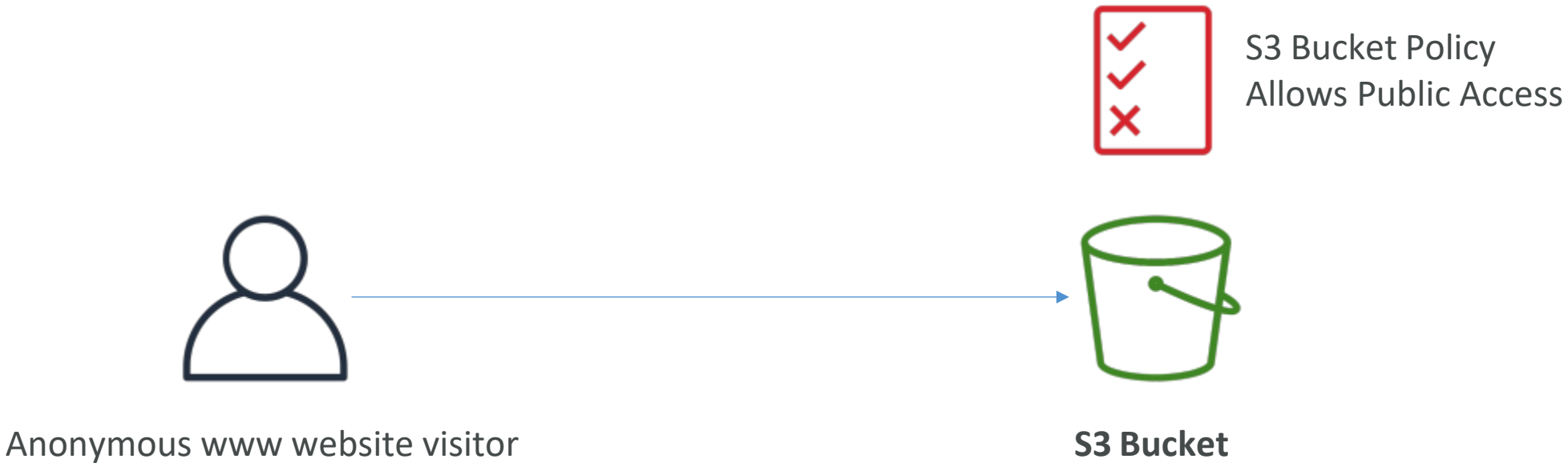
S3 Bucket Policies

- JSON based policies
  - Resources: buckets and objects
  - Effect: Allow / Deny
  - Actions: Set of API to Allow or Deny
  - Principal: The account or user to apply the policy to

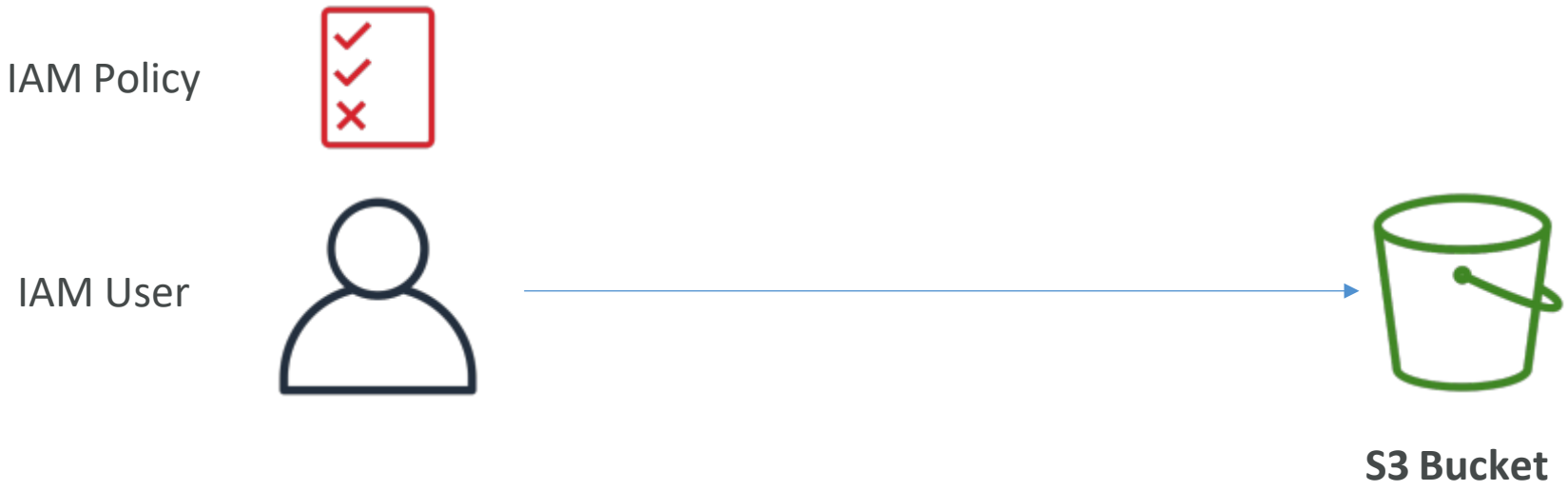
- Use S3 bucket for policy to:
  - Grant public access to the bucket
  - Force objects to be encrypted at upload
  - Grant access to another account (Cross Account)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicRead",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::examplebucket/*"
            ]
        }
    ]
}
```

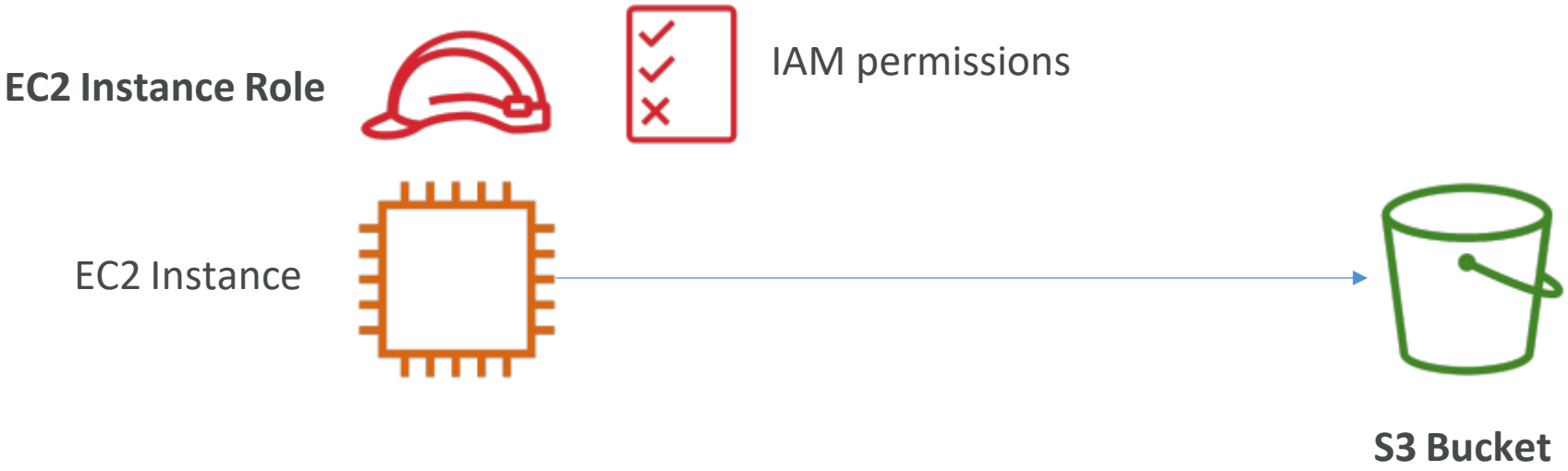
Example: Public Access - Use Bucket Policy



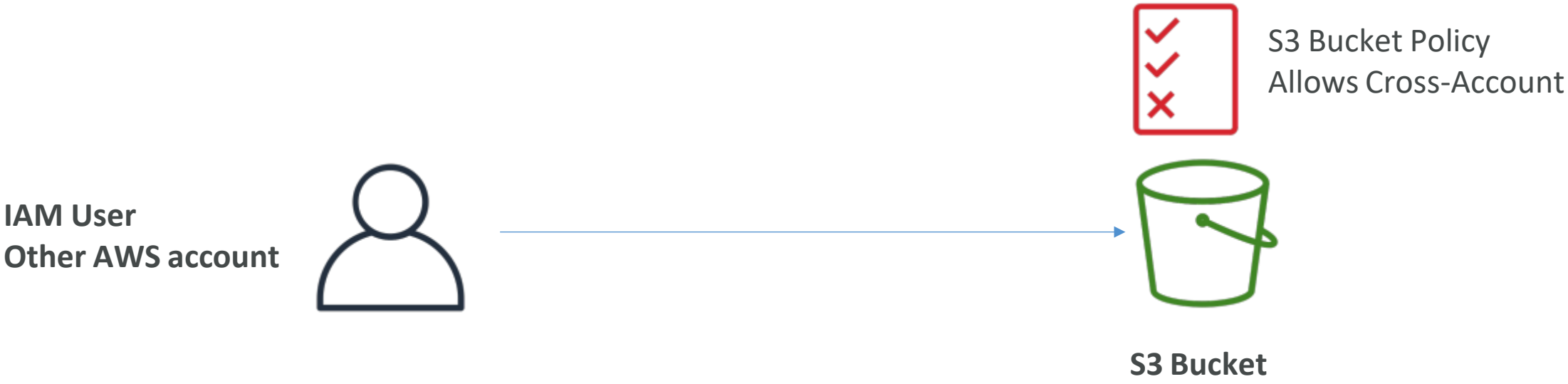
Example: User Access to S3 – IAM permissions



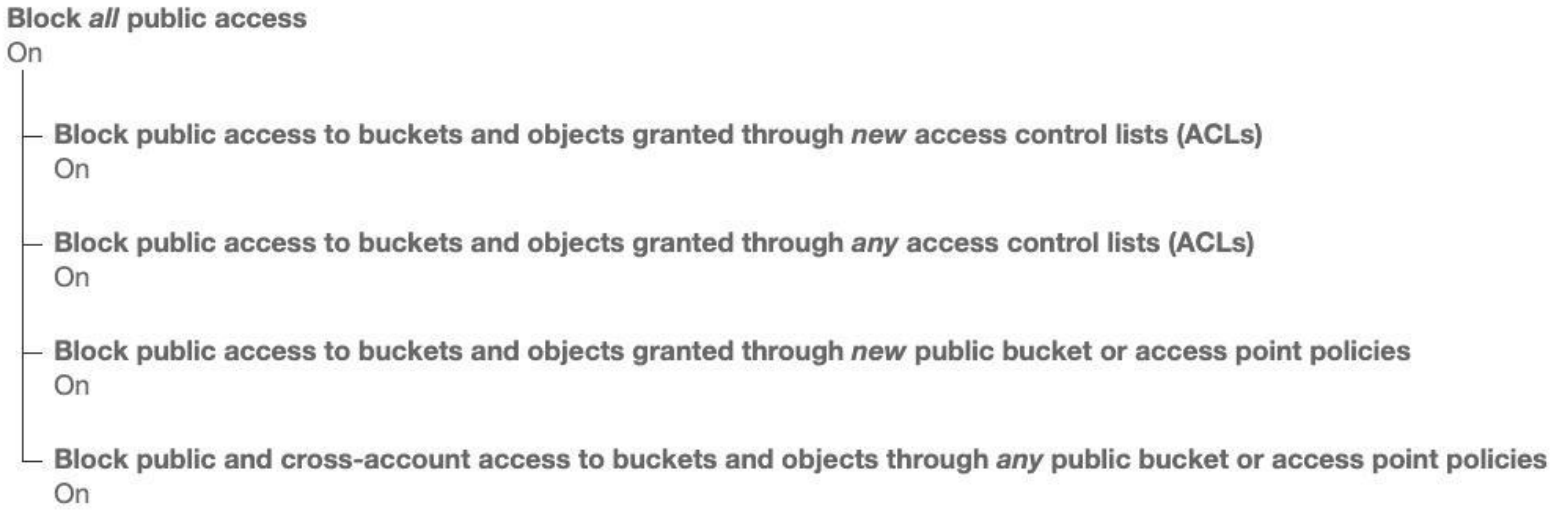
Example: EC2 instance access - Use IAM Roles



Advanced: Cross-Account Access – Use Bucket Policy



Bucket settings for Block Public Access



- **These settings were created to prevent company data leaks**
- If you know your bucket should never be public, leave these on
- Can be set at the account level

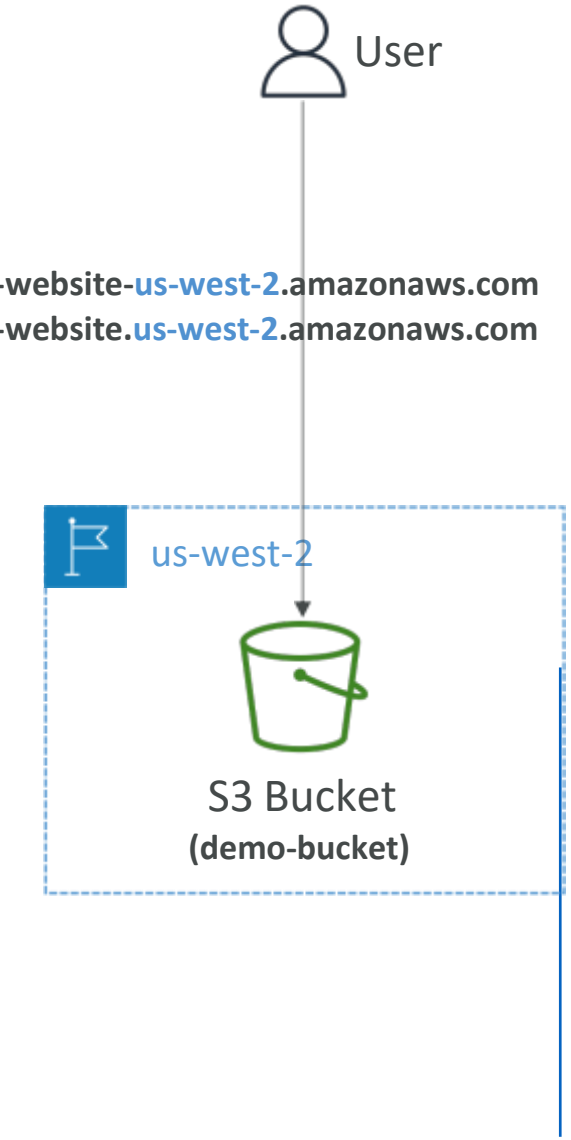
Amazon S3 – Static Website Hosting

- S3 can host static websites and have them accessible on the Internet

- The website URL will be (depending on the region)
  - **http://*bucket-name*.s3-website-*aws-region*.amazonaws.com**

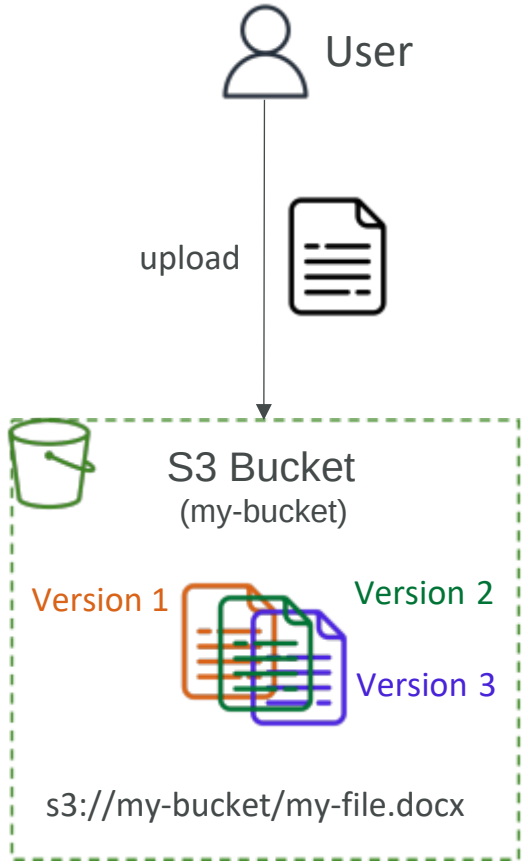
  OR
  - **http://*bucket-name*.s3-website.*aws-region*.amazonaws.com**

- If you get a **403 Forbidden** error, make sure the bucket policy allows public reads!



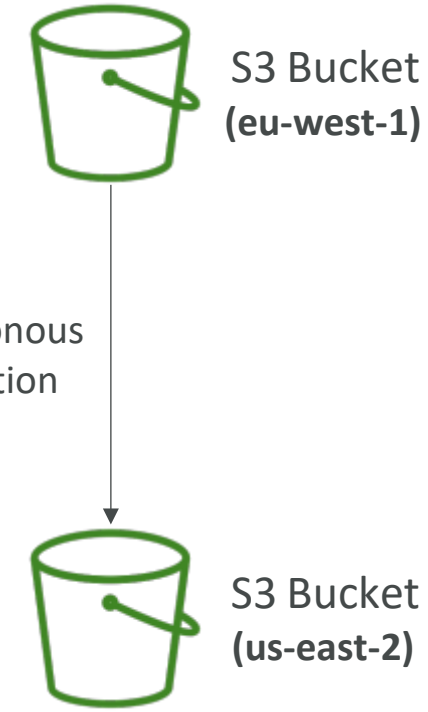
Amazon S3 -Versioning

- You can version your files in Amazon S3
- It is enabled at the **bucket level**
- Same key overwrite will change the "version": 1, 2, 3....
- It is best practice to version your buckets
  - Protect against unintended deletes (ability to restore a version)
  - Easy roll back to previous version
- Notes:
  - Any file that is not versioned prior to enabling versioning will have version "null"
  - Suspending versioning does not delete the previous versions



Amazon S3 – Replication (CRR & SRR)

- **Must enable Versioning** in source and destination buckets
- **Cross-Region Replication (CRR)**
- **Same-Region Replication (SRR)**
- Buckets can be in different AWS accounts
- Copying is asynchronous
- Must give proper IAM permissions to S3



- Use cases:
  - **CRR –** compliance, lower latency access, replication across accounts
  - **SRR –** log aggregation, live replication between production and test accounts

Amazon S3 – Replication (Notes)

- After you enable Replication, only new objects are replicated
- Optionally, you can replicate existing objects using **S3 Batch Replication**
  - Replicates existing objects and objects that failed replication

- For DELETE operations
  - **Can replicate delete markers** from source to target (optional setting)
  - Deletions with a version ID are not replicated (to avoid malicious deletes)

- **There is no "chaining" of replication**
  - If bucket 1 has replication into bucket 2, which has replication into bucket 3
  - Then objects created in bucket 1 are not replicated to bucket 3

S3 Storage Classes

- Amazon S3 Standard - General Purpose
- Amazon S3 Standard-Infrequent Access (IA)
- Amazon S3 One Zone-Infrequent Access
- Amazon S3 Glacier Instant Retrieval
- Amazon S3 Glacier Flexible Retrieval
- Amazon S3 Glacier Deep Archive
- Amazon S3 Intelligent Tiering

- Can move between classes manually or using S3 Lifecycle configurations

S3 Durability and Availability

- Durability:
  - High durability (99.999999999%, 11 9's) of objects across multiple AZ
  - If you store 10,000,000 objects with Amazon S3, you can on average expect to incur a loss of a single object once every 10,000 years
  - Same for all storage classes

- Availability:
  - Measures how readily available a service is
  - Varies depending on storage class
  - Example: S3 standard has 99.99% availability = not available 53 minutes a year

S3 Standard – General Purpose

- 99.99% Availability
- Used for frequently accessed data
- Low latency and high throughput
- Sustain 2 concurrent facility failures

- Use Cases: Big Data analytics, mobile & gaming applications, content distribution…

S3 Storage Classes – Infrequent Access

- For data that is less frequently accessed, but requires rapid access when needed
- Lower cost than S3 Standard

- **Amazon S3 Standard-Infrequent Access (S3 Standard-IA)**
  - 99.9% Availability
  - Use cases: Disaster Recovery, backups

- **Amazon S3 One Zone-Infrequent Access (S3 One Zone-IA)**
  - High durability (99.999999999%) in a single AZ; data lost when AZ is destroyed
  - 99.5% Availability
  - Use Cases: Storing secondary backup copies of on-premises data, or data you can recreate

Amazon S3 Glacier Storage Classes

- Low-cost object storage meant for archiving / backup
- Pricing: price for storage + object retrieval cost

- **Amazon S3 Glacier Instant Retrieval**
  - Millisecond retrieval, great for data accessed once a quarter
  - Minimum storage duration of 90 days
- **Amazon S3 Glacier Flexible Retrieval** (formerly Amazon S3 Glacier):
  - Expedited (1 to 5 minutes), Standard (3 to 5 hours), Bulk (5 to 12 hours) – free
  - Minimum storage duration of 90 days
- **Amazon S3 Glacier Deep Archive – for long term storage:**
  - Standard (12 hours), Bulk (48 hours)
  - Minimum storage duration of 180 days

S3 Intelligent-Tiering

- Small monthly monitoring and auto-tiering fee
- Moves objects automatically between Access Tiers based on usage
- There are no retrieval charges in S3 Intelligent-Tiering

- *Frequent Access tier (automatic):* default tier
- *Infrequent Access tier (automatic)*: objects not accessed for 30 days
- *Archive Instant Access tier (automatic):* objects not accessed for 90 days
- *Archive Access tier (optional):* configurable from 90 days to 700+ days
- *Deep Archive Access tier (optional):* config. from 180 days to 700+ days

S3 Storage Classes Comparison

	Standard	Intelligent-Tiering	Standard-IA	One Zone-IA	Glacier Instant Retrieval	Glacier Flexible Retrieval	Glacier Deep Archive
Durability	99.999999999% == (11 9's)						
Availability	99.99%	99.9%	99.9%	99.5%	99.9%	99.99%	99.99%
Availability SLA	99.9%	99%	99%	99%	99%	99.9%	99.9%
Availability Zones	>= 3	>= 3	>= 3	1	>= 3	>= 3	>= 3
Min. Storage Duration Charge	None	None	30 Days	30 Days	90 Days	90 Days	180 Days
Min. Billable Object Size	None	None	128 KB	128 KB	128 KB	40 KB	40 KB
Retrieval Fee	None	None	Per GB retrieved	Per GB retrieved	Per GB retrieved	Per GB retrieved	Per GB retrieved

https://aws.amazon.com/s3/storage-classes/

S3 Storage Classes – Price Comparison Example: us-east-1

	Standard	Intelligent-Tiering	Standard-IA	One Zone-IA	Glacier Instant Retrieval	Glacier Flexible Retrieval	Glacier Deep Archive
Storage Cost (per GB per month)	$0.023	$0.0025 - $0.023	%0.0125	$0.01	$0.004	$0.0036	$0.00099
Retrieval Cost (per 1000 request)	**GET:** $0.0004 **POST:** $0.005	**GET:** $0.0004 **POST:** $0.005	**GET:** $0.001 **POST:** $0.01	**GET:** $0.001 **POST:** $0.01	**GET:** $0.01 **POST:** $0.02	**GET:** $0.0004 **POST:** $0.03 **Expedited:** $10 **Standard:** $0.05 **Bulk:** free	**GET:** $0.0004 **POST:** $0.05 **Standard:** $0.10 **Bulk:** $0.025
Retrieval Time	Instantaneous					**Expedited** (1 – 5 mins) **Standard** (3 – 5 hours) Bulk (5 – 12 hours)	**Standard** (12 hours) **Bulk** (48 hours)
Monitoring Cost (pet 1000 objects)		$0.0025					

https://aws.amazon.com/s3/pricing/

Developing on AWS

CLI, SDK and IAM Policies

AWS CLI Dry Runs

- Sometimes, we'd just like to make sure we have the permissions…
- But not actually run the commands!

- Some AWS CLI commands (such as EC2) can become expensive if they succeed, say if we wanted to try to create an EC2 Instance
- Some AWS CLI commands (not all) contain a **--dry-run** option to simulate API calls

- Let's practice!

AWS CLI STS Decode Errors

- When you run API calls and they fail, you can get a long error message
- This error message can be decoded using the **STS** command line:
- sts **decode-authorization-message**

- Let's practice!

AWS EC2 Instance Metadata

- AWS EC2 Instance Metadata is powerful but one of the least known features to developers
- It allows AWS EC2 instances to "learn about themselves" **without using an IAM Role for that purpose.**
- The URL is http://169.254.169.254/latest/meta-data
- You can retrieve the IAM Role name from the metadata, but you CANNOT retrieve the IAM Policy.
- Metadata = Info about the EC2 instance
- Userdata = launch script of the EC2 instance

- Let's practice and see what we can do with it!

MFA with CLI

- To use MFA with the CLI, you must create a temporary session
- To do so, you must run the **STS GetSessionToken** API call

- **aws sts get-session-token** --serial-number arn-of-the-mfa-device --token-code code-from-token --duration-seconds 3600


```
{
    "Credentials": {
        "SecretAccessKey": "secret-access-key",
        "SessionToken": "temporary-session-token",
        "Expiration": "expiration-date-time",
        "AccessKeyId": "access-key-id"
    }
}
```

AWS SDK Overview

- What if you want to perform actions on AWS directly from your applications code ? (without using the CLI).
- You can use an SDK (software development kit) !
- Official SDKs are…
  - Java
  - .NET
  - Node.js
  - PHP
  - Python (named boto3 / botocore)
  - Go
  - Ruby
  - C++

AWS SDK Overview

- We have to use the AWS SDK when coding against AWS Services such as DynamoDB

- Fun fact… the AWS CLI uses the Python SDK (boto3)
- The exam expects you to know when you should use an SDK
- We'll practice the AWS SDK when we get to the Lambda functions

- Good to know: if you don't specify or configure a default region, then us-east-1 will be chosen by default

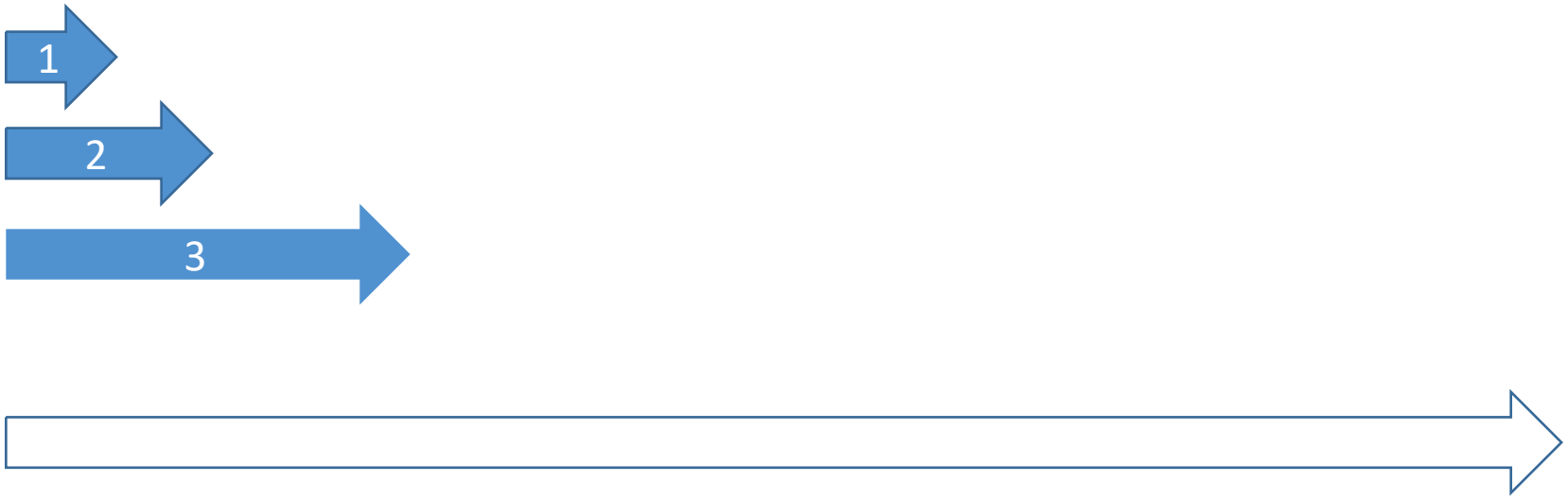
AWS Limits (Quotas)

- API Rate Limits
  - **DescribeInstances** API for EC2 has a limit of 100 calls per seconds
  - **GetObject** on S3 has a limit of 5500 GET per second per prefix
  - For Intermittent Errors: implement Exponential Backoff
  - For Consistent Errors: request an API throttling limit increase

- Service Quotas (Service Limits)
  - Running On-Demand Standard Instances: 1152 vCPU
  - You can request a service limit increase by **opening a ticket**
  - You can request a service quota increase by using the **Service Quotas API**

Exponential Backoff (any AWS service)

- If you get **ThrottlingException** intermittently, use exponential backoff
- Retry mechanism already included in AWS SDK API calls
- Must implement yourself if using the AWS API as-is or in specific cases
  - **Must only implement the retries on 5xx server errors and throttling**
  - Do not implement on the 4xx client errors



AWS CLI Credentials Provider Chain

- The CLI will look for credentials in this order

1. **Command line options** – --region, --output, and --profile
2. **Environment variables** – AWS_ACCESS_KEY_ID,AWS_SECRET_ACCESS_KEY, and AWS_SESSION_TOKEN
3. **CLI credentials file** –aws configure
   ~/.aws/credentials on Linux / Mac & C:\Users\user\.aws\credentials on Windows
4. **CLI configuration file** – aws configure
   ~/.aws/config on Linux / macOS & C:\Users\USERNAME\.aws\config on Windows
5. **Container credentials** – for ECS tasks
6. **Instance profile credentials** – for EC2 Instance Profiles

AWS SDK Default Credentials Provider Chain

- The Java SDK (example) will look for credentials in this order

1. **Java system properties** – aws.accessKeyId and aws.secretKey
2. **Environment variables** – AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY
3. **The default credential profiles file** – ex at: ~/.aws/credentials, shared by many SDK
4. **Amazon ECS container credentials** – for ECS containers
5. **Instance profile credentials**– used on EC2 instances

AWS Credentials Scenario

- An application deployed on an EC2 instance is using environment variables with credentials from an IAM user to call the Amazon S3 API.
- The IAM user has S3FullAccess permissions.
- The application only uses one S3 bucket, so according to best practices:
  - An IAM Role & EC2 Instance Profile was created for the EC2 instance
    The Role was assigned the minimum permissions to access that one S3 bucket

- **The IAM Instance Profile was assigned to the EC2 instance, but it still had access to all S3 buckets. Why?**
  the credentials chain is still giving priorities to the environment variables

AWS Credentials Best Practices

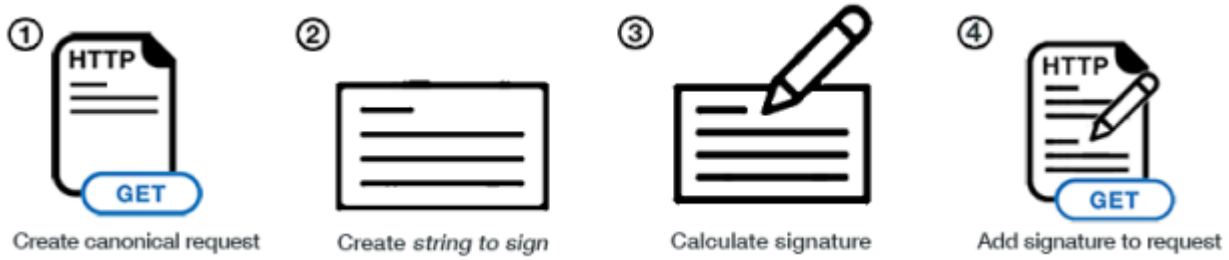
- **Overall, NEVER EVER STORE AWS CREDENTIALS IN YOUR CODE**
- Best practice is for credentials to be inherited from the credentials chain

- **If using working within AWS, use IAM Roles**
  - => EC2 Instances Roles for EC2 Instances
  - => ECS Roles for ECS tasks
  - => Lambda Roles for Lambda functions
- If working outside of AWS, use environment variables / named profiles

Signing AWS API requests

- When you call the AWS HTTP API, you sign the request so that AWS can identify you, using your AWS credentials (access key & secret key)
- Note: some requests to Amazon S3 don't need to be signed
- If you use the SDK or CLI, the HTTP requests are signed for you

- You should sign an AWS HTTP request using Signature v4 (SigV4)



SigV4 Request examples

- HTTP Header option

```
GET https://iam.amazonaws.com/?Action=ListUsers&Version=2010-05-08 HTTP/1.1
Authorization: AWS4-HMAC-SHA256 Credential=AKIDEXAMPLE/20150830/us-east-1/iam/aws4_request,
SignedHeaders=content-type;host;x-amz-date,
Signature=5d672d79c15b13162d9279b0855cfba6789a8edb4c82c400e06b5924a6f2b5d7
content-type: application/x-www-form-urlencoded; charset=utf-8
host: iam.amazonaws.com
x-amz-date: 20150830T123600Z
```

- Query String option (ex: S3 pre-signed URLs)

```
GET https://iam.amazonaws.com?Action=ListUsers&Version=2010-05-08&
X-Amz-Algorithm=AWS4-HMAC-SHA256&
X-Amz-Credential=AKIDEXAMPLE%2F20150830%2Fus-east-1%2Fiam%2Faws4_request&
X-Amz-Date=20150830T123600Z&X-Amz-Expires=60&X-Amz-SignedHeaders=content-type%3Bhost&
X-Amz-Signature=37ac2f4fde00b0ac9bd9eadeb459b1bbee224158d66e7ae5fcadb70b2d181d02 HTTP/1.1
content-type: application/x-www-form-urlencoded; charset=utf-8
host: iam.amazonaws.com
```

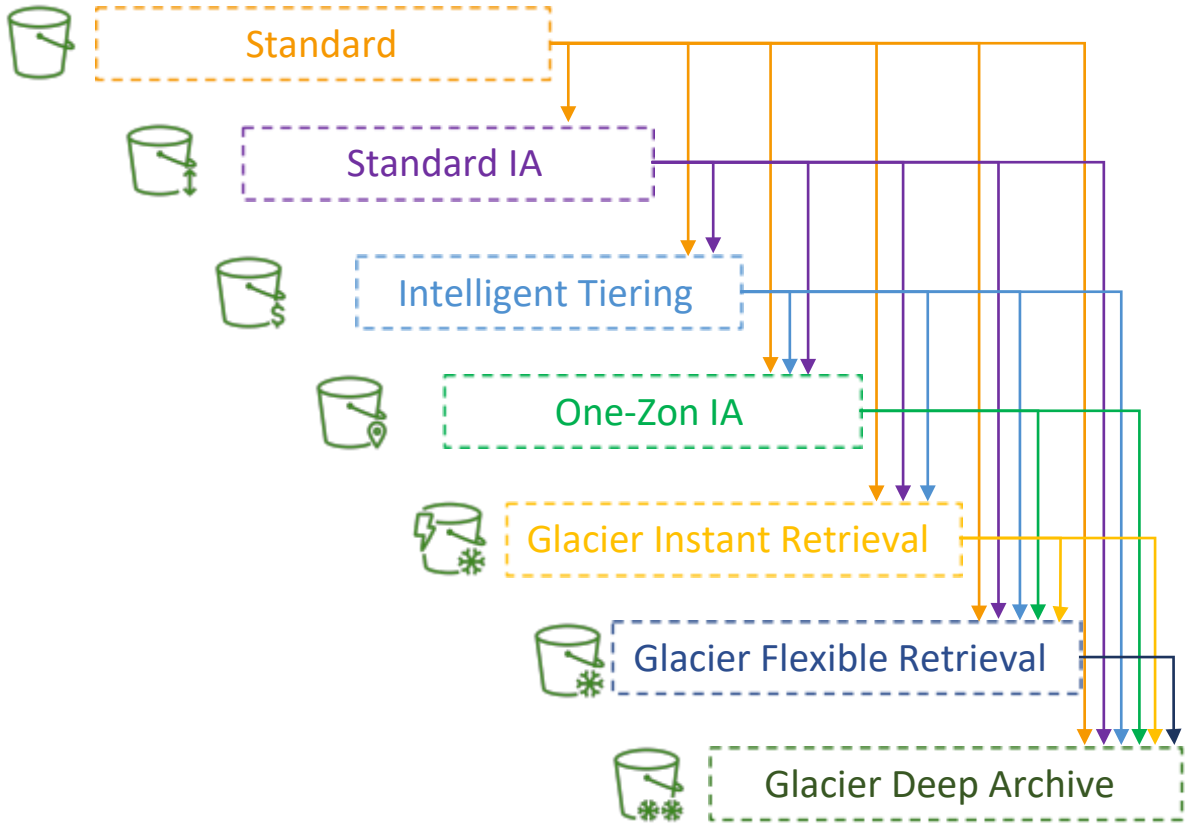
Advanced S3

Amazon S3 – Moving between Storage Classes

- You can transition objects between storage classes

- For infrequently accessed object, move them to **Standard IA**
- For archive objects that you don't need fast access to, move them to **Glacier or Glacier Deep Archive**

- Moving objects can be automated using a **Lifecycle Rules**



Amazon S3 – Lifecycle Rules

- **Transition Actions –** configure objects to transition to another storage class
  - Move objects to Standard IA class 60 days after creation
  - Move to Glacier for archiving after 6 months

- **Expiration actions –** configure objects to expire (delete) after some time
  - Access log files can be set to delete after a 365 days
  - **Can be used to delete old versions of files (if versioning is enabled)**
  - Can be used to delete incomplete Multi-Part uploads

- Rules can be created for a certain prefix (example: *s3://mybucket/mp3/*)*
- Rules can be created for certain objects Tags (example: *Department: Finance*)

Amazon S3 – Lifecycle Rules (Scenario 1)

- **Your application on EC2 creates images thumbnails after profile photos are uploaded to Amazon S3. These thumbnails can be easily recreated, and only need to be kept for 60 days. The source images should be able to be immediately retrieved for these 60 days, and afterwards, the user can wait up to 6 hours. How would you design this?**

- S3 source images can be on **Standard**, with a lifecycle configuration to transition them to **Glacier** after 60 days
- S3 thumbnails can be on **One-Zone IA**, with a lifecycle configuration to expire them (delete them) after 60 days

Amazon S3 – Lifecycle Rules (Scenario 2)

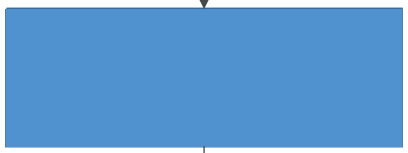
- **A rule in your company states that you should be able to recover your deleted S3 objects immediately for 30 days, although this may happen rarely. After this time, and for up to 365 days, deleted objects should be recoverable within 48 hours.**

- E**nable S3 Versioning** in order to have object versions, so that "deleted objects" are in fact hidden by a "delete marker" and can be recovered
- Transition the "noncurrent versions" of the object to **Standard IA**
- Transition afterwards the "noncurrent versions" to **Glacier Deep Archive**

Amazon S3 Analytics – Storage Class Analysis

- Help you decide when to transition objects to the right storage class
- Recommendations for **Standard** and **Standard IA**
  - Does NOT work for One-Zone IA or Glacier
- Report is updated daily
- 24 to 48 hours to start seeing data analysis

- Good first step to put together Lifecycle Rules (or improve them)!



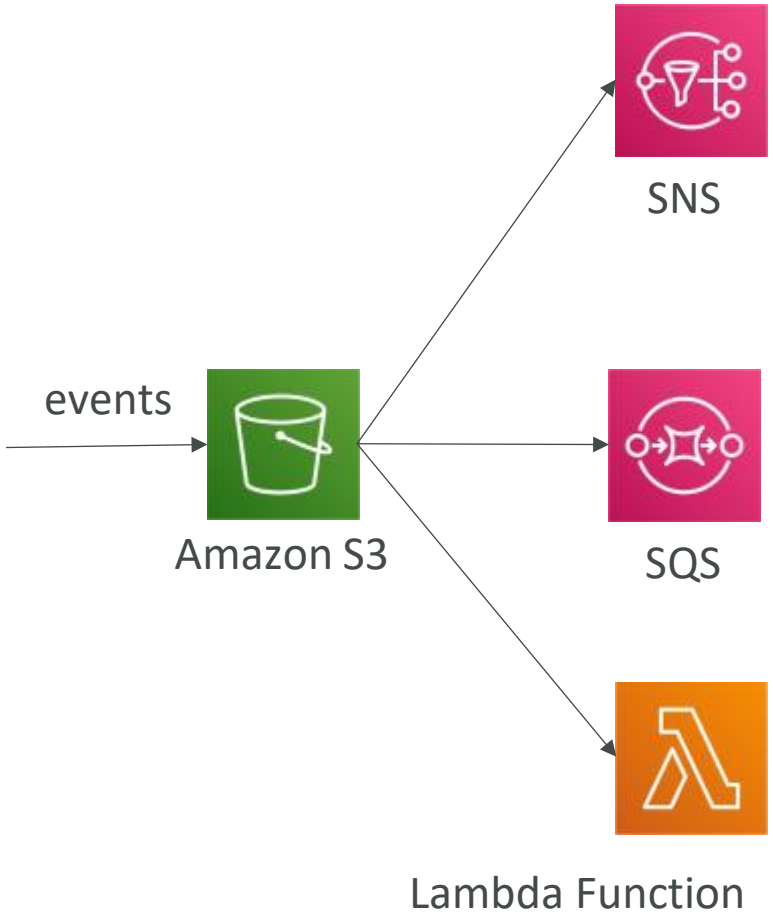
.csv report

Date	StorageClass	ObjectAge
8/22/2022	STANDARD	000-014
8/25/2022	STANDARD	030-044
9/6/2022	STANDARD	120-149

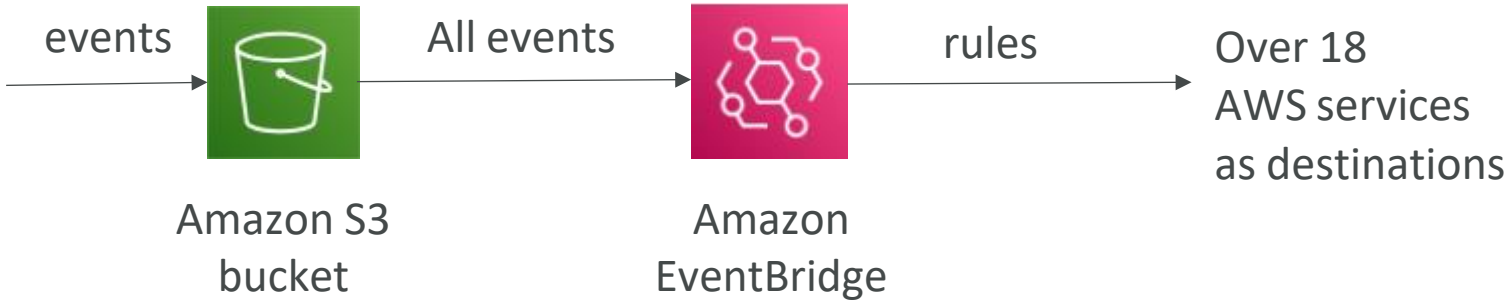
S3 Event Notifications

- S3:ObjectCreated, S3:ObjectRemoved, S3:ObjectRestore, S3:Replication…
- Object name filtering possible (*.jpg)
- Use case: generate thumbnails of images uploaded to S3
- **Can create as many "S3 events" as desired**

- S3 event notifications typically deliver events in seconds but can sometimes take a minute or longer



S3 Event Notifications with Amazon EventBridge



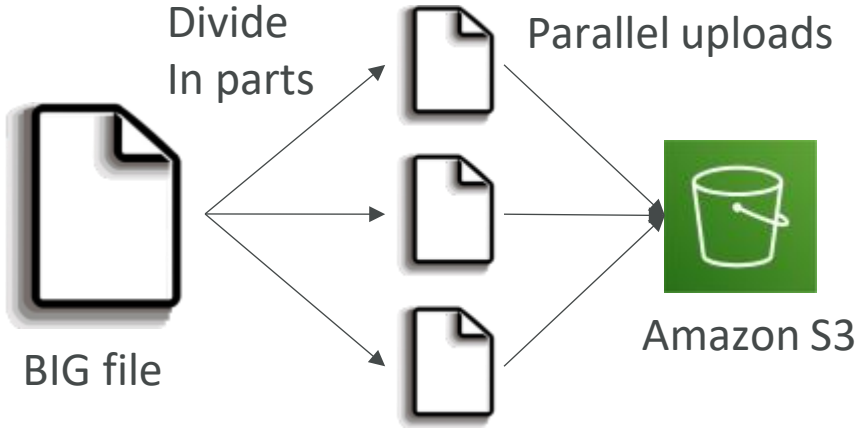
- **Advanced filtering** options with JSON rules (metadata, object size, name…)
- **Multiple Destinations** – ex Step Functions, Kinesis Streams / Firehose…
- **EventBridge Capabilities** – Archive, Replay Events, Reliable delivery

S3 – Baseline Performance

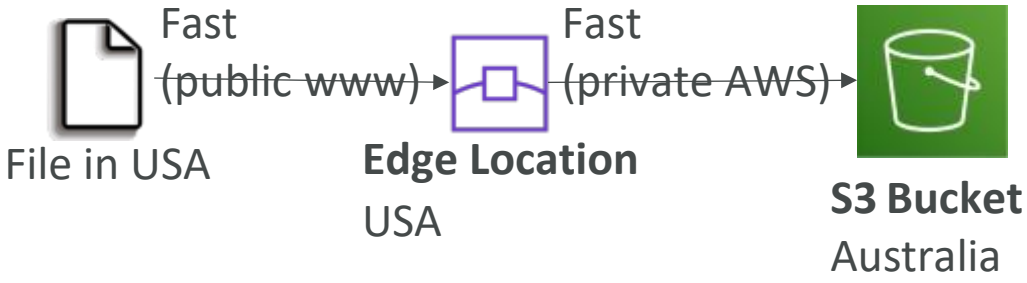
- Amazon S3 automatically scales to high request rates, latency 100-200 ms
- Your application can achieve at least **3,500 PUT/COPY/POST/DELETE and 5,500 GET/HEAD requests per second per prefix in a bucket.**
- There are no limits to the number of prefixes in a bucket.
- Example (object path => prefix):
  - bucket/folder1/sub1/file => /folder1/sub1/
  - bucket/folder1/sub2/file => /folder1/sub2/
  - bucket/1/file => /1/
  - bucket/2/file => /2/
- If you spread reads across all four prefixes evenly, you can achieve 22,000 requests per second for GET and HEAD

S3 Performance

- **Multi-Part upload:**
  - recommended for files > 100MB, must use for files > 5GB
  - Can help parallelize uploads (speed up transfers)



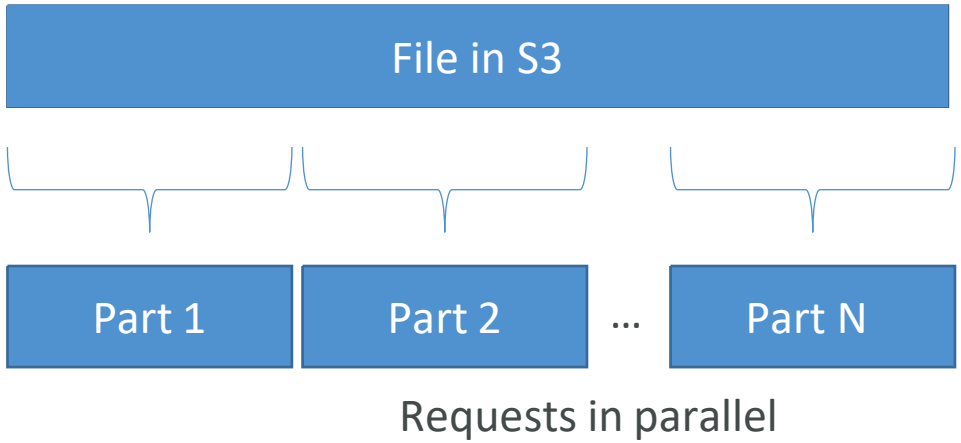
- **S3 Transfer Acceleration**
  - Increase transfer speed by transferring file to an AWS edge location which will forward the data to the S3 bucket in the target region
  - Compatible with multi-part upload



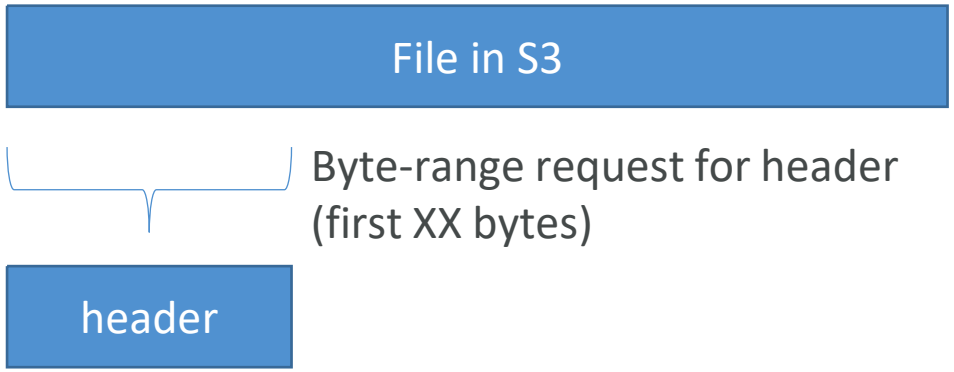
S3 Performance – S3 Byte-Range Fetches

- Parallelize GETs by requesting specific byte ranges
- Better resilience in case of failures

Can be used to speed up downloads

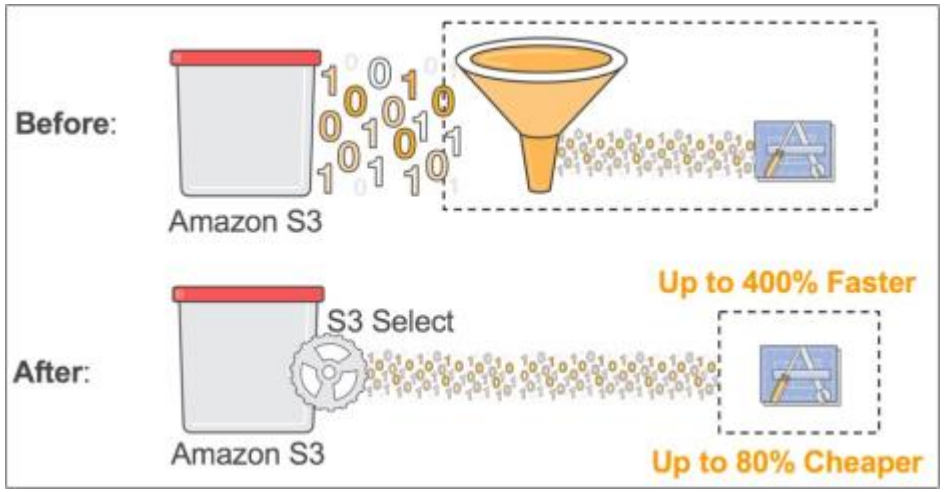


Can be used to retrieve only partial data (for example the head of a file)



S3 Select & Glacier Select

- Retrieve less data using SQL by performing **server-side filtering**
- Can filter by rows & columns (simple SQL statements)
- Less network transfer, less CPU cost client-side



https://aws.amazon.com/blogs/aws/s3-glacier-select/

Amazon S3 Security

Amazon S3 – Object Encryption

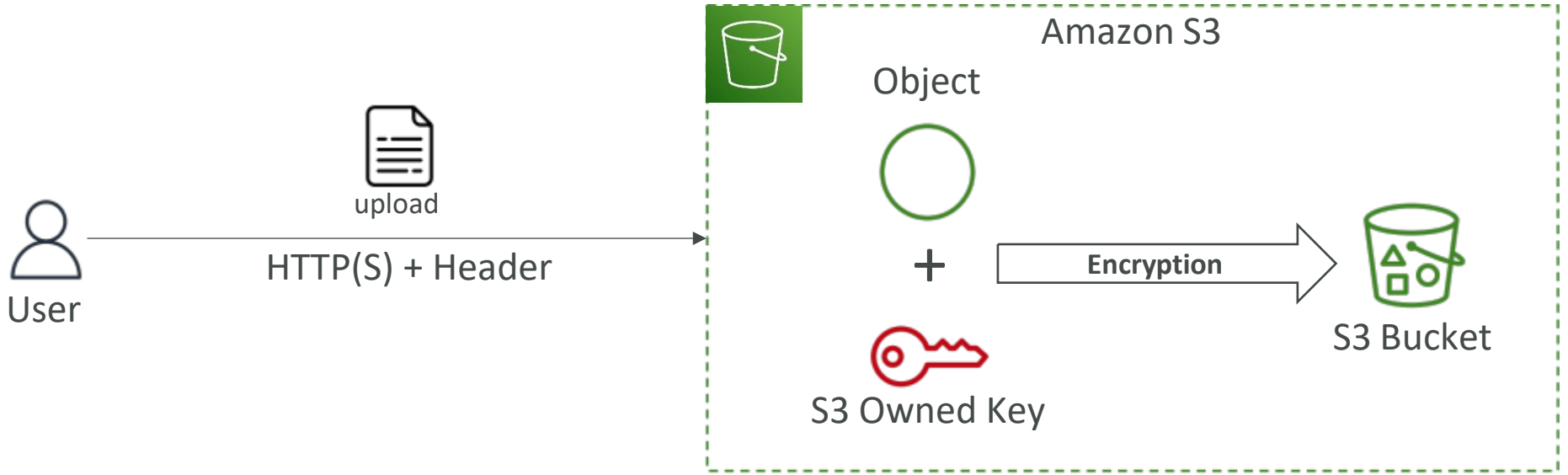
- You can encrypt objects in S3 buckets using one of 4 methods

- **Server-Side Encryption (SSE)**
  - **Server-Side Encryption with Amazon S3-Managed Keys (SSE-S3) – Enabled by Default**
    - Encrypts S3 objects using keys handled, managed, and owned by AWS
  - **Server-Side Encryption with KMS Keys stored in AWS KMS (SSE-KMS)**
    - Leverage AWS Key Management Service (AWS KMS) to manage encryption keys
  - **Server-Side Encryption with Customer-Provided Keys (SSE-C)**
    - When you want to manage your own encryption keys
- **Client-Side Encryption**

- It's important to understand which ones are for which situation for the exam

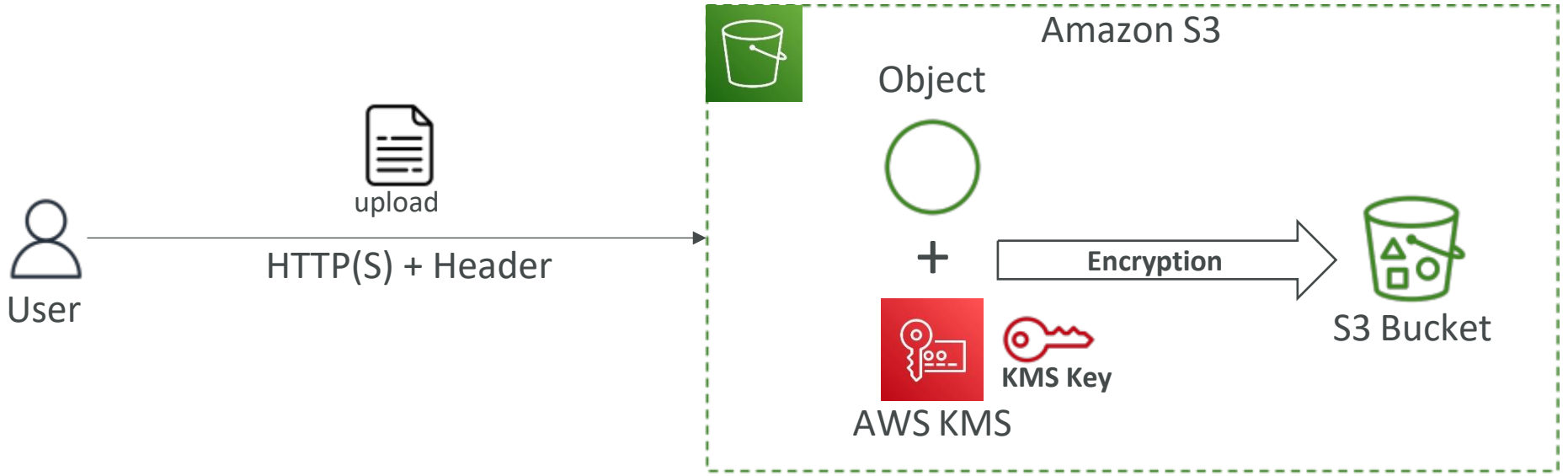
Amazon S3 Encryption – SSE-S3

- Encryption using keys handled, managed, and owned by AWS
- Object is encrypted server-side
- Encryption type is **AES-256**
- Must set header **"x-amz-server-side-encryption": "AES256"**
- **Enabled by default for new buckets & new objects**



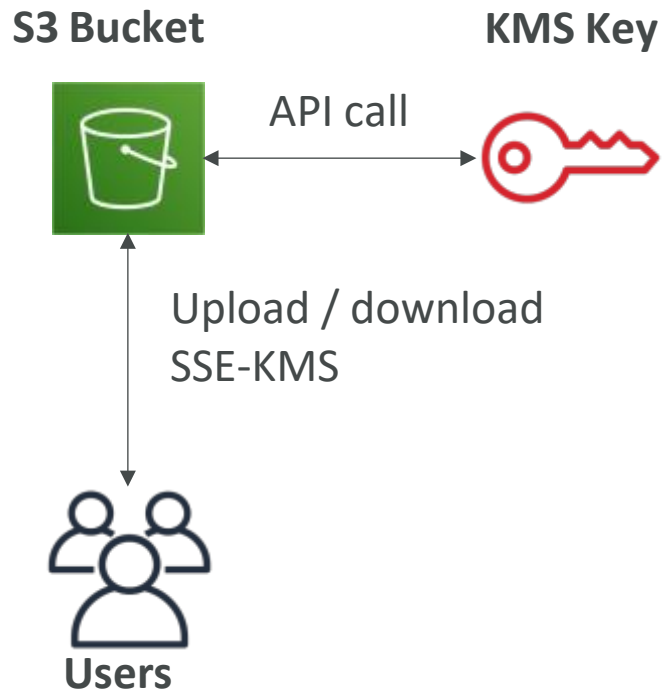
Amazon S3 Encryption – SSE-KMS

- Encryption using keys handled and managed by AWS KMS (Key Management Service)
- KMS advantages: user control + audit key usage using CloudTrail
- Object is encrypted server side
- Must set header **"x-amz-server-side-encryption": "aws:kms"**



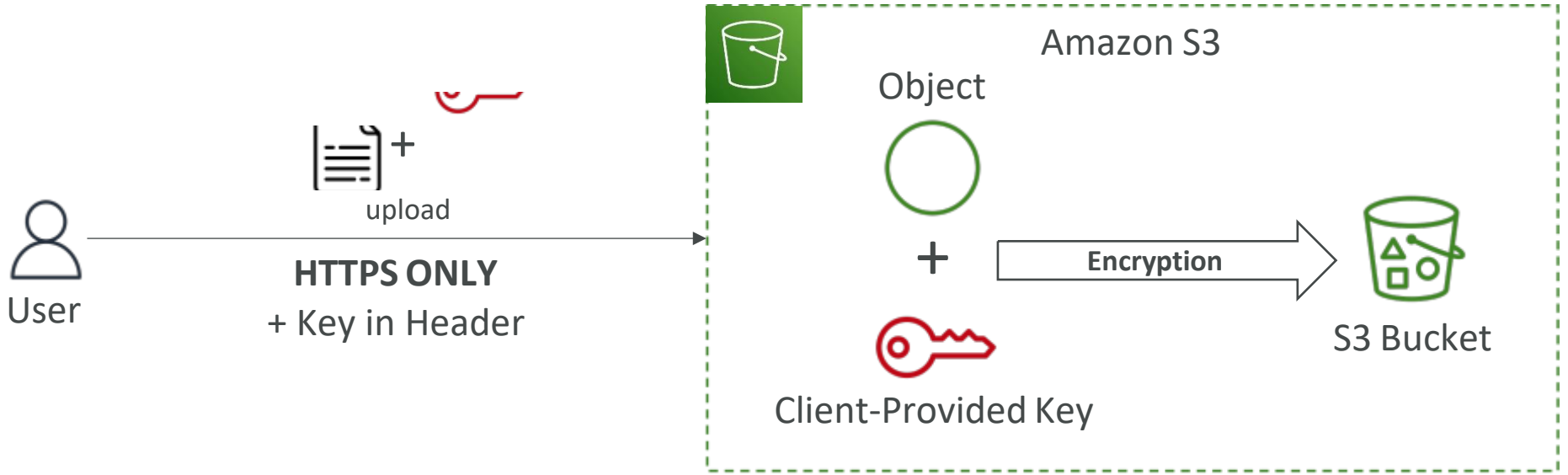
SSE-KMS Limitation

- If you use SSE-KMS, you may be impacted by the KMS limits
- When you upload, it calls the **GenerateDataKey** KMS API
- When you download, it calls the **Decrypt** KMS API
- Count towards the KMS quota per second (5500, 10000, 30000 req/s based on region)
- You can request a quota increase using the Service Quotas Console



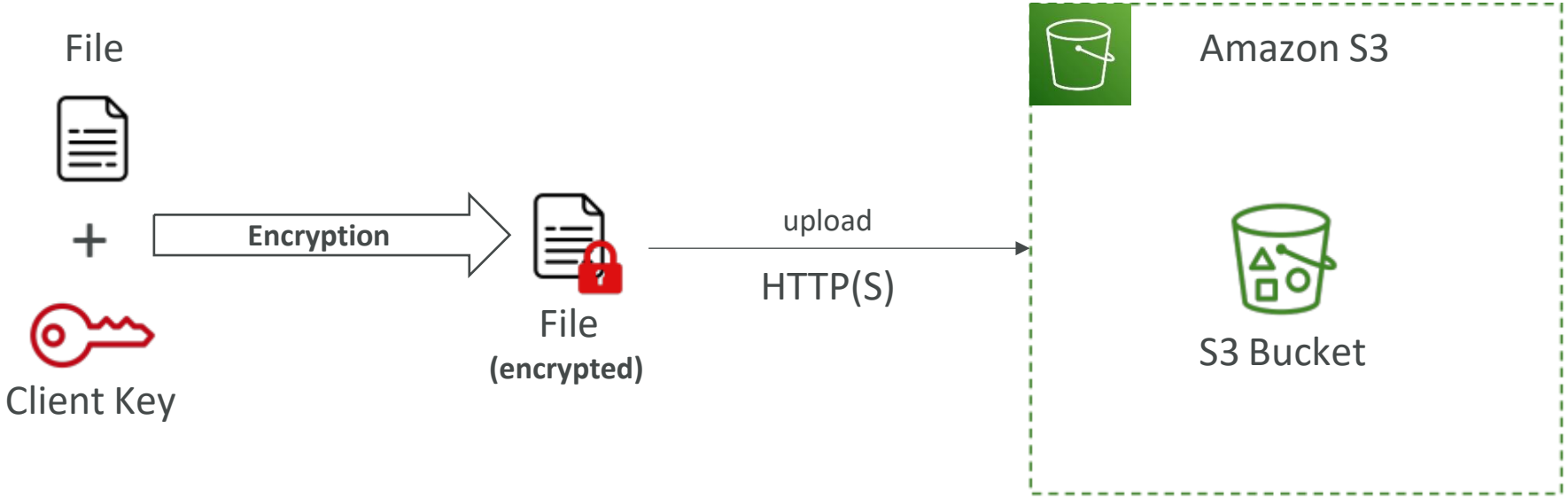
Amazon S3 Encryption – SSE-C

- Server-Side Encryption using keys fully managed by the customer outside of AWS
- Amazon S3 does **NOT** store the encryption key you provide
- **HTTPS must be used**
- Encryption key must provided in HTTP headers, for every HTTP request made



Amazon S3 Encryption – Client-Side Encryption

- Use client libraries such as **Amazon S3 Client-Side Encryption Library**
- Clients must encrypt data themselves before sending to Amazon S3
- Clients must decrypt data themselves when retrieving from Amazon S3
- Customer fully manages the keys and encryption cycle



Amazon S3 – Encryption in transit (SSL/TLS)

- Encryption in flight is also called SSL/TLS

- Amazon S3 exposes two endpoints:
  - **HTTP Endpoint** – non encrypted
  - **HTTPS Endpoint** – encryption in flight

- **HTTPS is recommended**
- **HTTPS is mandatory for SSE-C**
- Most clients would use the HTTPS endpoint by default

Amazon S3 – Default Encryption vs. Bucket Policies

- **SSE-S3 encryption is automatically applied to new objects stored in S3 bucket**
- Optionally, you can "force encryption" using a bucket policy and refuse any API call to PUT an S3 object without encryption headers (SSE-KMS or SSE-C)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "s3:PutObject",
            "Principal": "*",
            "Resource": "arn:aws:s3:::my-bucket/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-server-side-encryption": "aws:kms"
                }
            }
        }
    ]
}
```

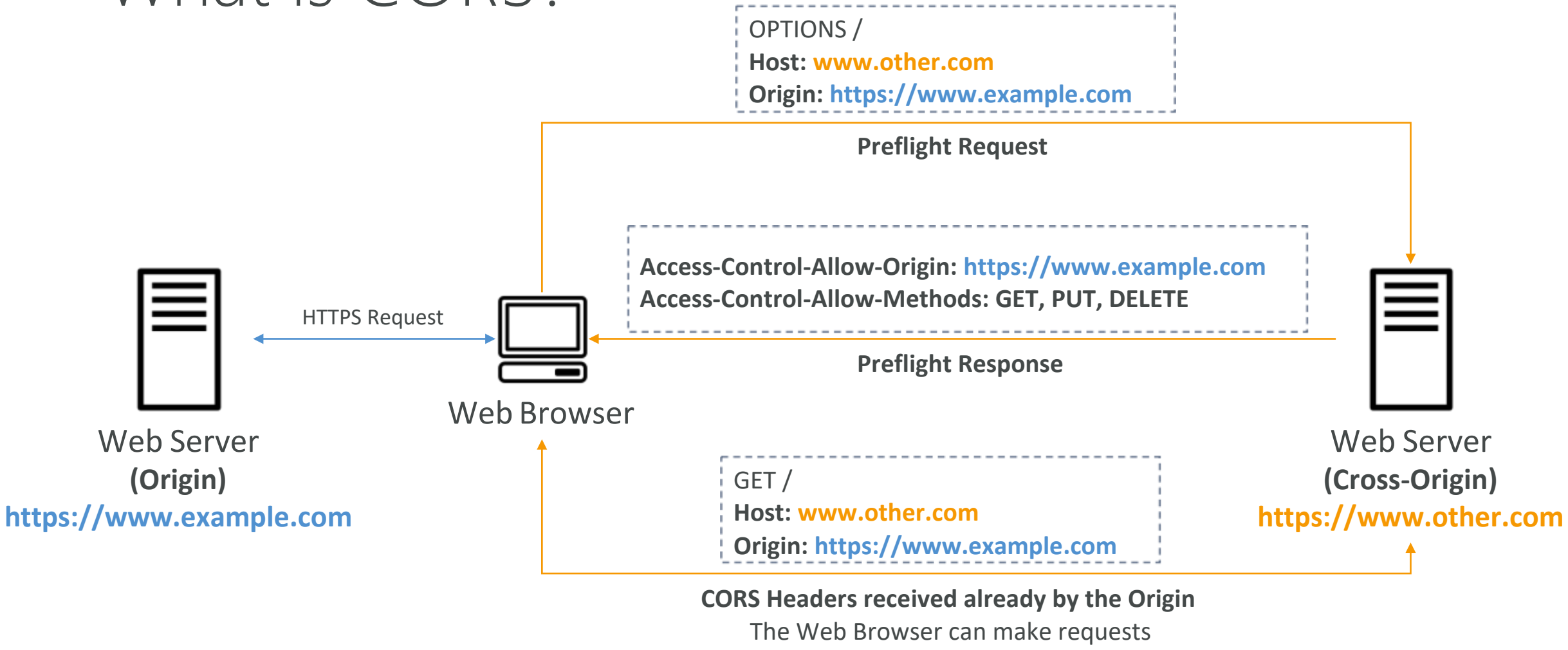
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "s3:PutObject",
            "Principal": "*",
            "Resource": "arn:aws:s3:::my-bucket/*",
            "Condition": {
                "Null": {
                    "s3:x-amz-server-side-encryption-customer-algorithm": "true"
                }
            }
        }
    ]
}
```

- **<u>Note:</u> Bucket Policies are evaluated before "Default Encryption"**

What is CORS?

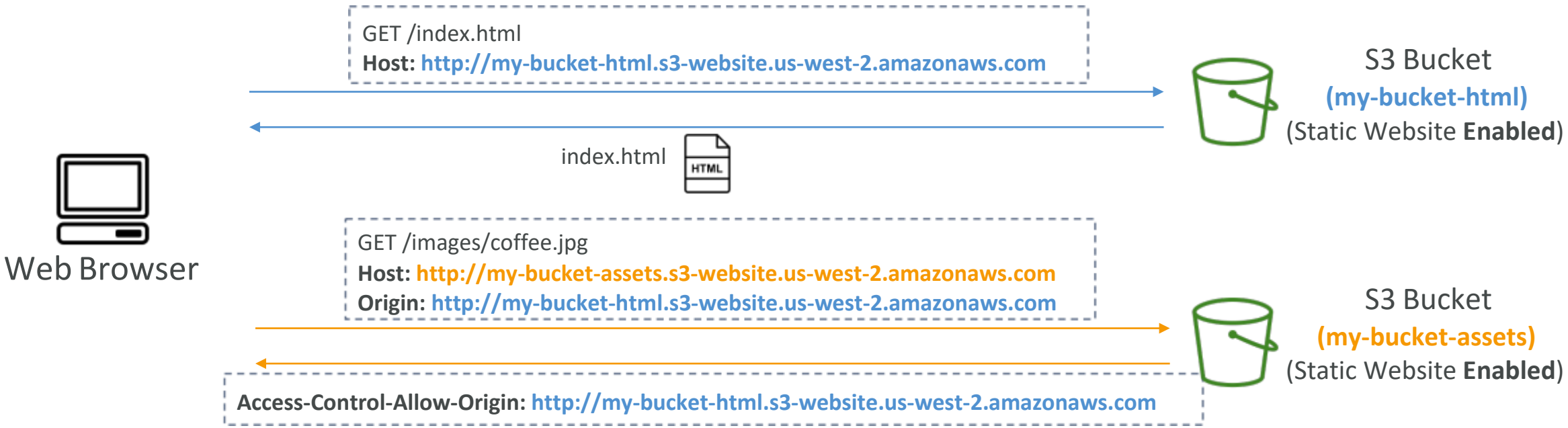
- **Cross-Origin Resource Sharing (CORS)**
- **Origin = scheme (protocol) + host (domain) + port**
  - example: https://www.example.com (implied port is 443 for HTTPS, 80 for HTTP)
- **Web Browser** based mechanism to allow requests to other origins while visiting the main origin
- Same origin: **http://example.com**/app1 & **http://example.com**/app2
- Different origins: **http://www.example.com** & **http://other.example.com**
- The requests won't be fulfilled unless the other origin allows for the requests, using **CORS Headers** (example: **Access-Control-Allow-Origin**)

What is CORS?


OPTIONS /
Host: www.other.com
Origin: https://www.example.com
Preflight Request
Access-Control-Allow-Origin: https://www.example.com
Access-Control-Allow-Methods: GET, PUT, DELETE
HTTPS Request
Preflight Response
Web Server
(Origin)
https://www.example.com
Web Browser
Web Server
(Cross-Origin)
https://www.other.com
GET /
Host: www.other.com
Origin: https://www.example.com
CORS Headers received already by the Origin
The Web Browser can make requests

Amazon S3 – CORS

- If a client makes a cross-origin request on our S3 bucket, we need to enable the correct CORS headers
- It's a popular exam question
- You can allow for a specific origin or for * (all origins)



Amazon S3 – MFA Delete

- **MFA (Multi-Factor Authentication) –** force users to generate a code on a device (usually a mobile phone or hardware) before doing important operations on S3
- MFA will be required to:
  - Permanently delete an object version
  - Suspend Versioning on the bucket
- MFA won't be required to:
  - Enable Versioning
  - List deleted versions
- To use MFA Delete, **Versioning must be enabled** on the bucket
- **Only the bucket owner (root account) can enable/disable MFA Delete**

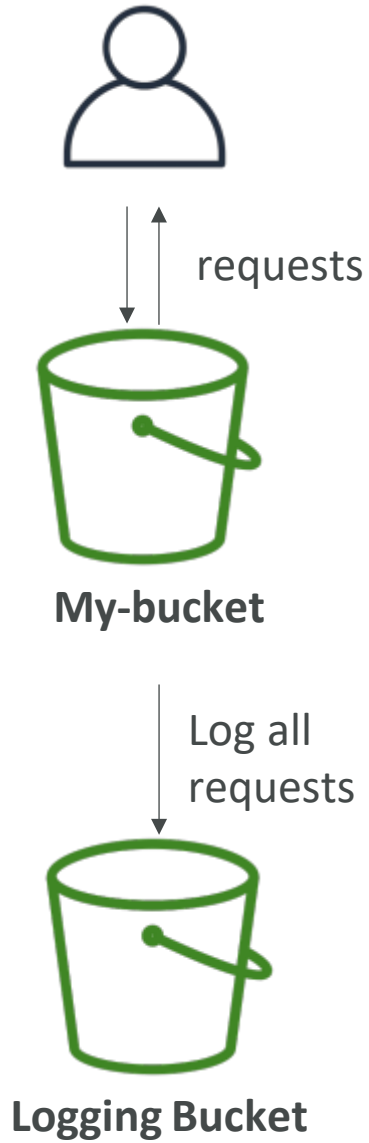
Google Authenticator

MFA Hardware Device

S3 Access Logs

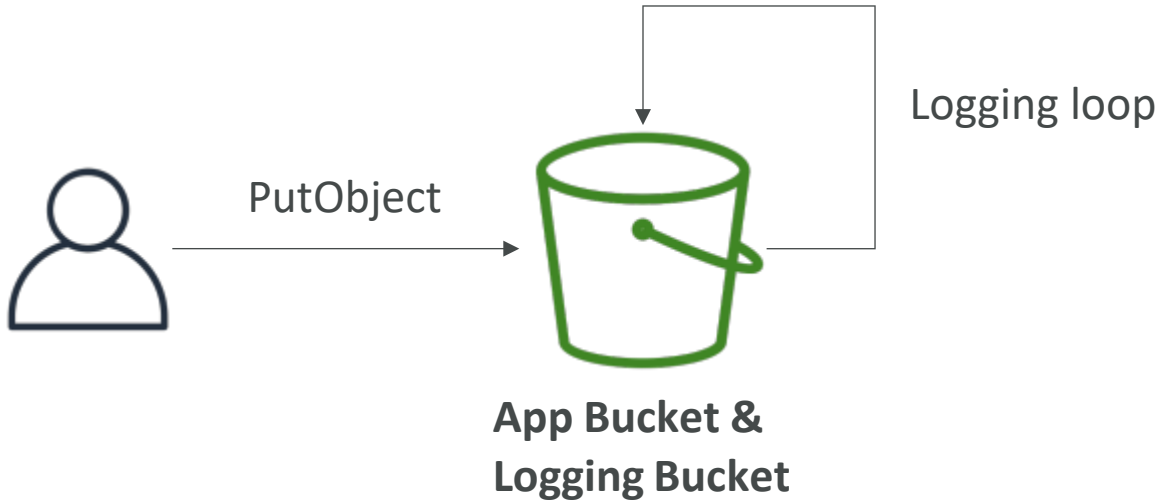
- For audit purpose, you may want to log all access to S3 buckets
- Any request made to S3, from any account, authorized or denied, will be logged into another S3 bucket
- That data can be analyzed using data analysis tools…
- The target logging bucket must be in the same AWS region

- The log format is at: https://docs.aws.amazon.com/AmazonS3/latest/dev/LogFormat.html



S3 Acess Logs: Warning

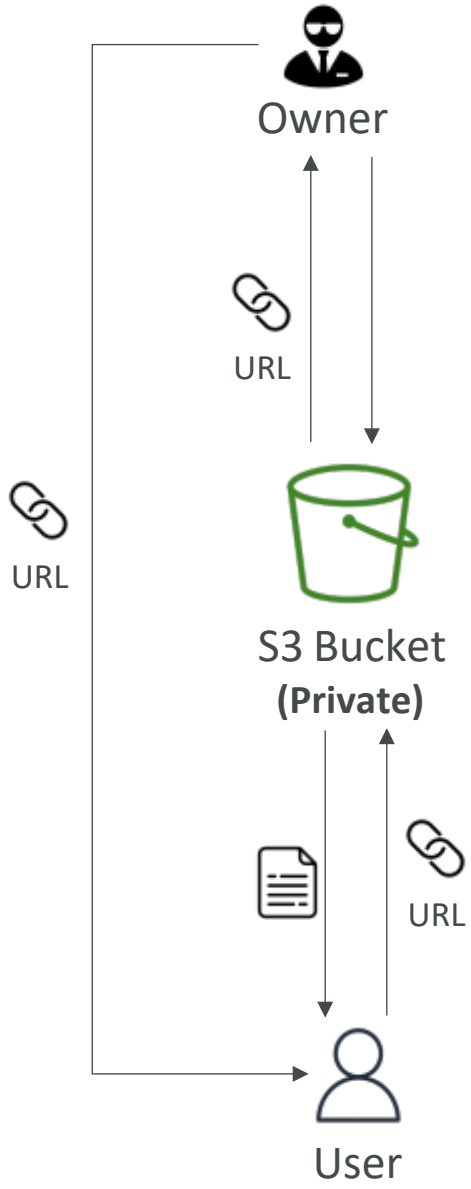
- Do not set your logging bucket to be the monitored bucket
- It will create a logging loop, and **your bucket will grow exponentially**



Amazon S3 – Pre-Signed URLs

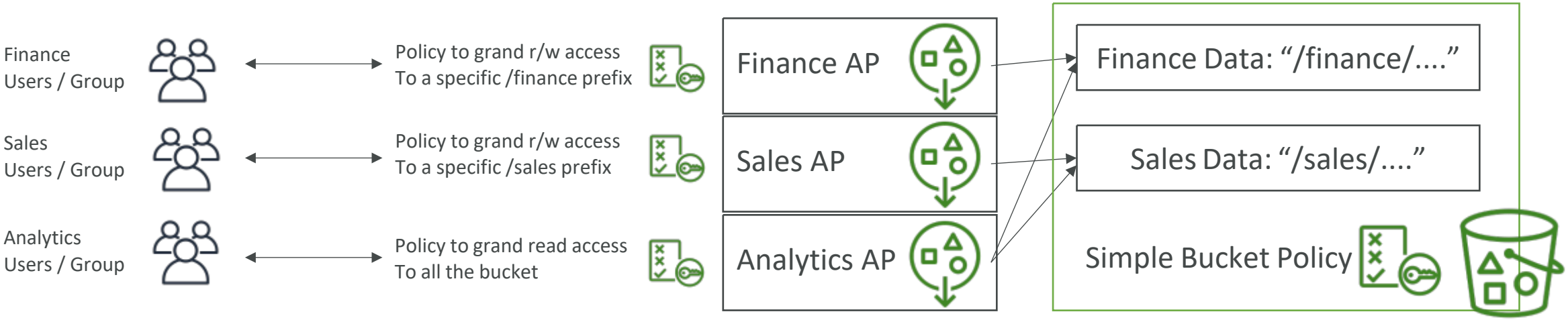
- Generate pre-signed URLs using the **S3 Console, AWS CLI or SDK**
- **URL Expiration**
  - S3 Console – 1 min up to 720 mins (12 hours)
  - AWS CLI – configure expiration with *--expires-in* parameter in seconds (default 3600 secs, max. 604800 secs ~ 168 hours)
- Users given a pre-signed URL inherit the permissions of the user that generated the URL for GET / PUT

- Examples:
  - Allow only logged-in users to download a premium video from your S3 bucket
  - Allow an ever-changing list of users to download files by generating URLs dynamically
  - Allow temporarily a user to upload a file to a precise location in your S3 bucket



S3 – Access Points

- Each Access Point gets its own DNS and policy to limit who can access it
  - A specific IAM user / group
  - One policy per Access Point => **Easier to manage than complex bucket policies**



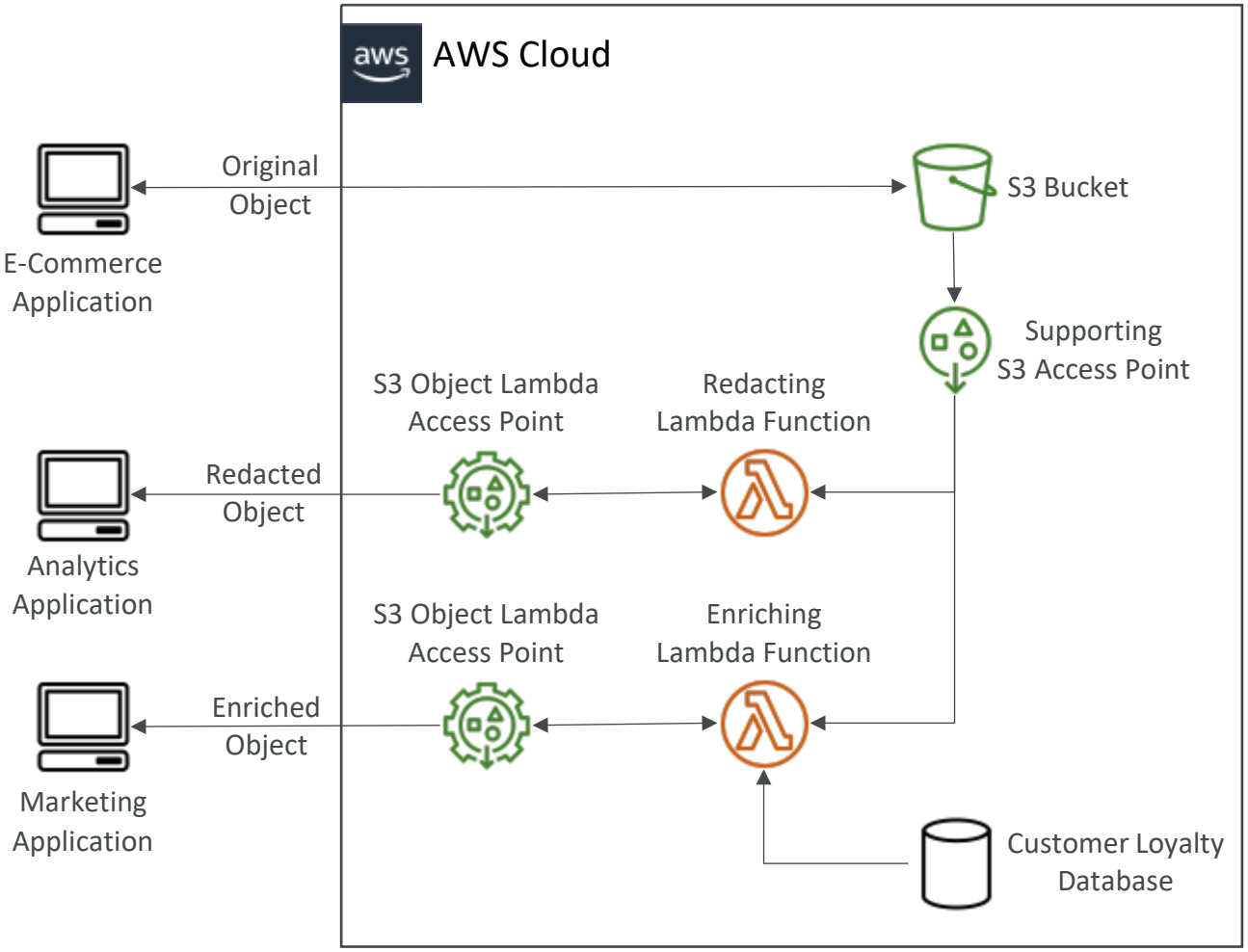
S3 Object Lambda

- Use AWS Lambda Functions to change the object before it is retrieved by the caller application

Only one S3 bucket is needed, on top of which we create **S3 Access Point** and **S3 Object Lambda Access Points.**

Use Cases:

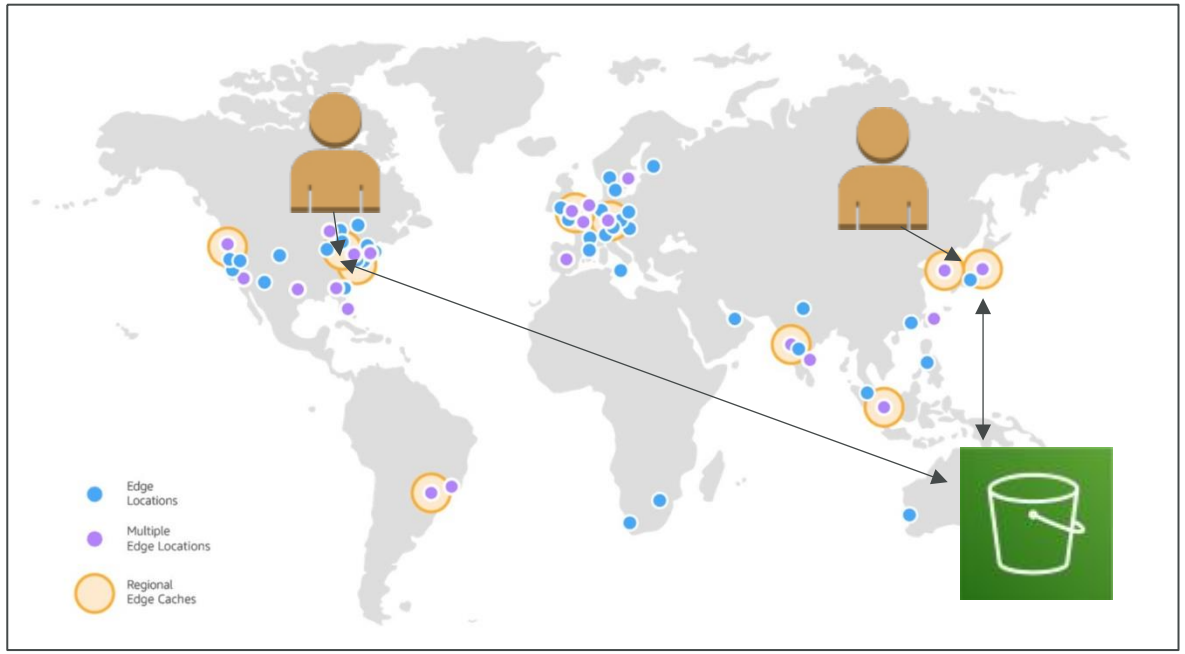
- Redacting personally identifiable information for analytics or non-production environments.
- Converting across data formats, such as converting XML to JSON.
- Resizing and watermarking images on the fly using caller-specific details, such as the user who requested the object.



CloudFront Section

Amazon CloudFront

- Content Delivery Network (CDN)
- **Improves read performance, content is cached at the edge**
- Improves users experience
- 216 Point of Presence globally (edge locations)
- **DDoS protection (because worldwide), integration with Shield, AWS Web Application Firewall**



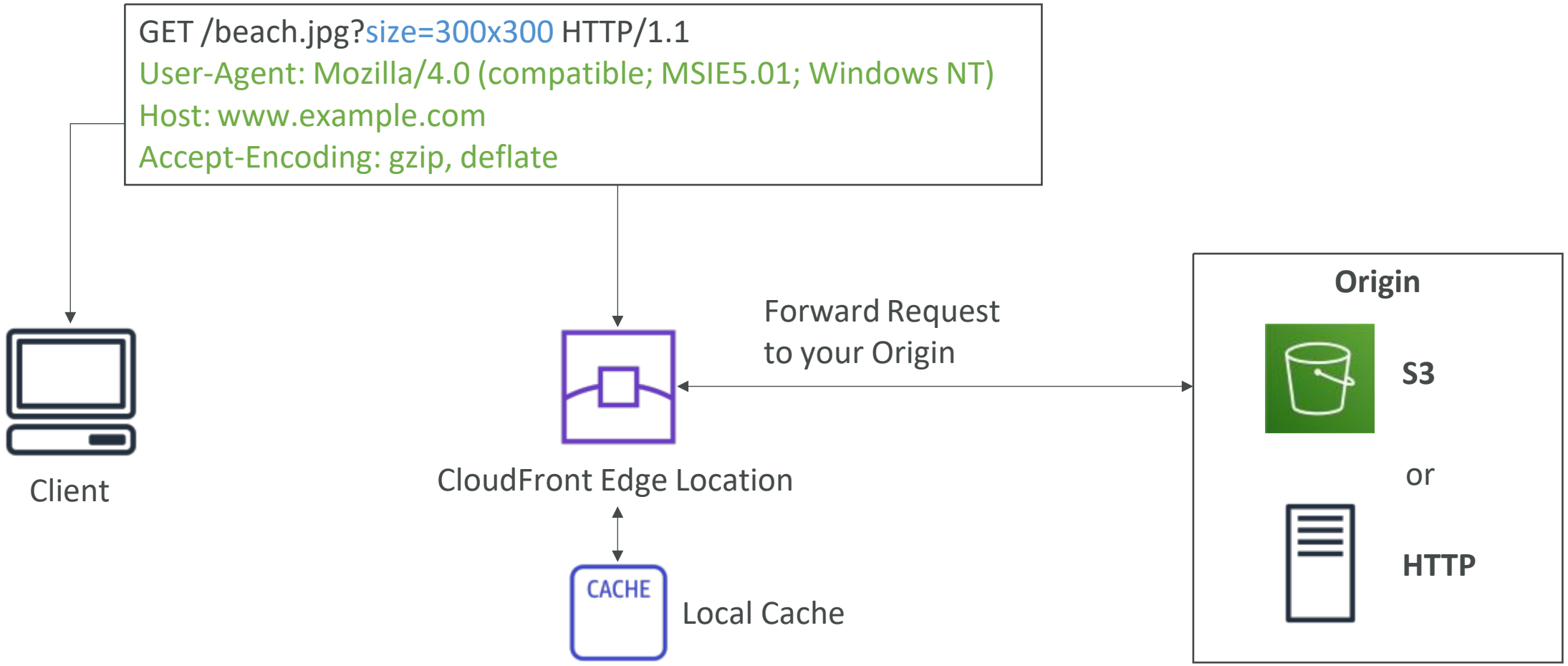
Source: https://aws.amazon.com/cloudfront/features/?nc=sn&loc=2

CloudFront – Origins

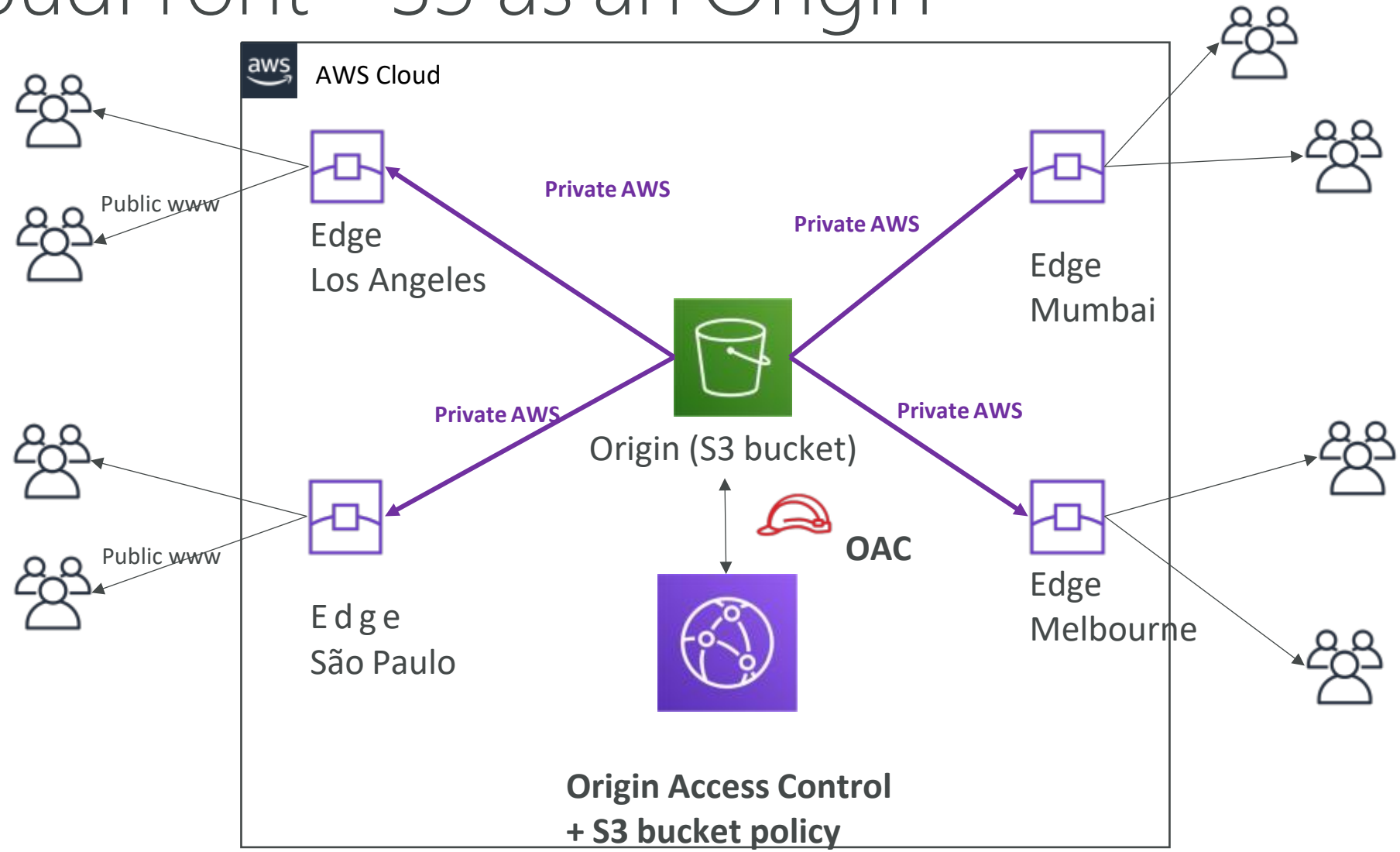
- **S3 bucket**
  - For distributing files and caching them at the edge
  - Enhanced security with CloudFront **Origin Access Control (OAC)**
  - OAC is replacing Origin Access Identity (OAI)
  - CloudFront can be used as an ingress (to upload files to S3)

- **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



CloudFront – S3 as an Origin



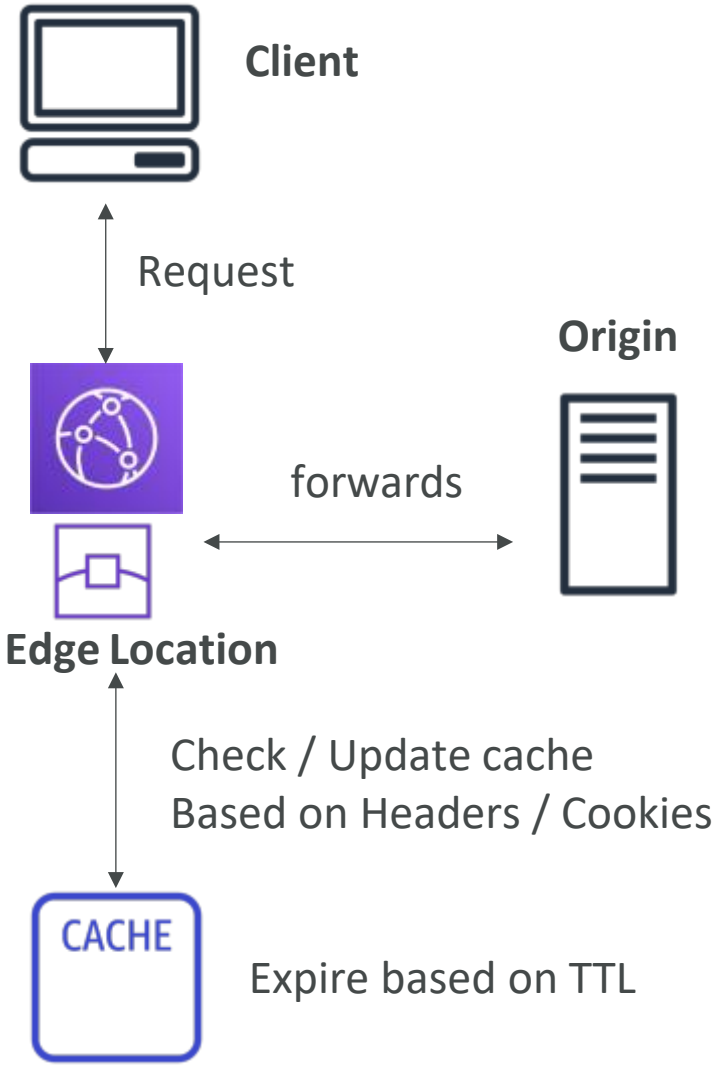
CloudFront vs S3 Cross Region Replication

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

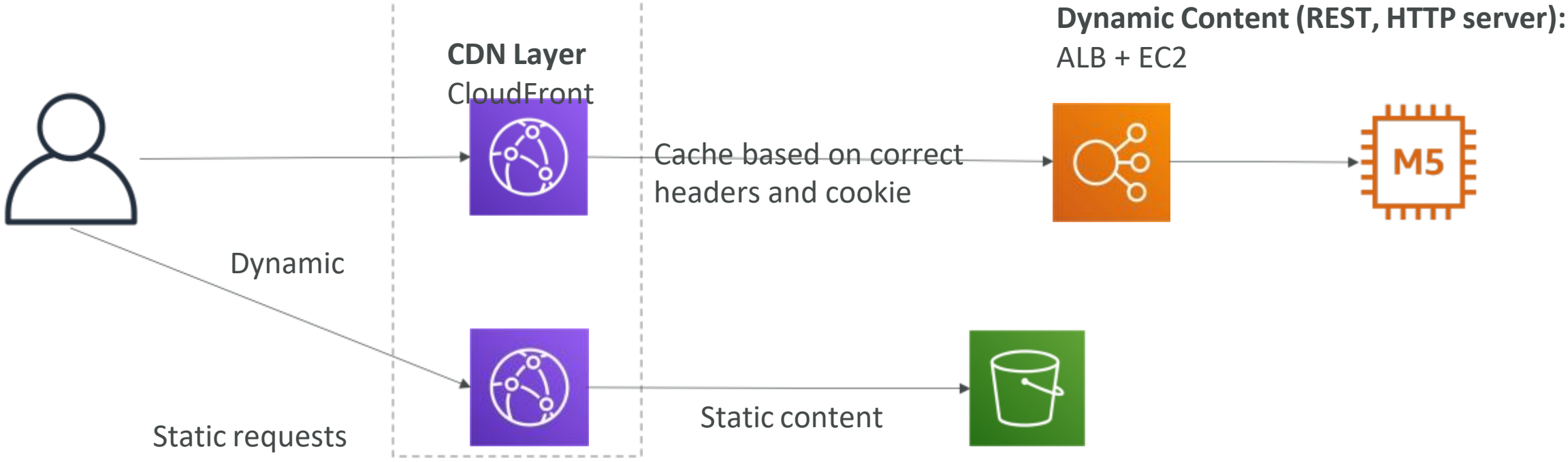
- S3 Cross Region Replication:
  - Must be setup for each region you want replication to happen
  - Files are updated in near real-time
  - Read only
  - Great f **or dynamic content that needs to be available at low-latency in few** regions

CloudFront Caching

- Cache based on
  - Headers
  - Session Cookies
  - Query String Parameters
- The cache lives at each CloudFront **Edge Location**
- You want to maximize the cache hit rate to minimize requests on the origin
- Control the TTL (0 seconds to 1 year), can be set by the origin using the Cache-Control header, Expires header…
- You can invalidate part of the cache using the **CreateInvalidation** API

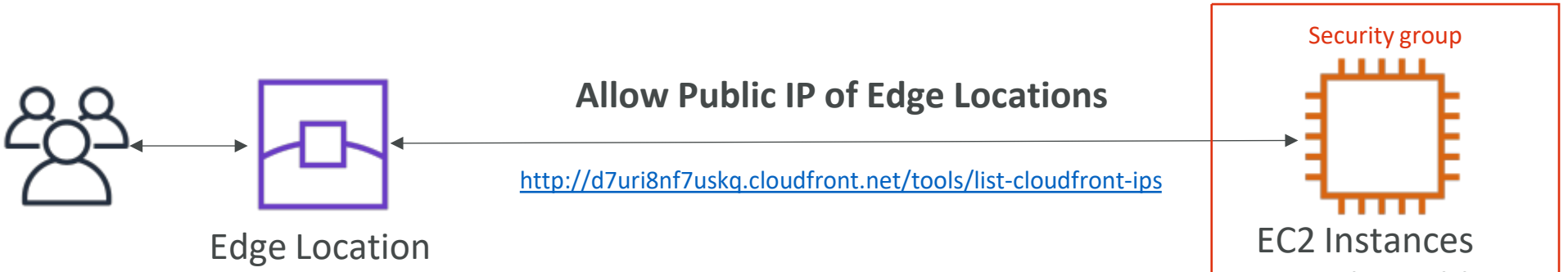


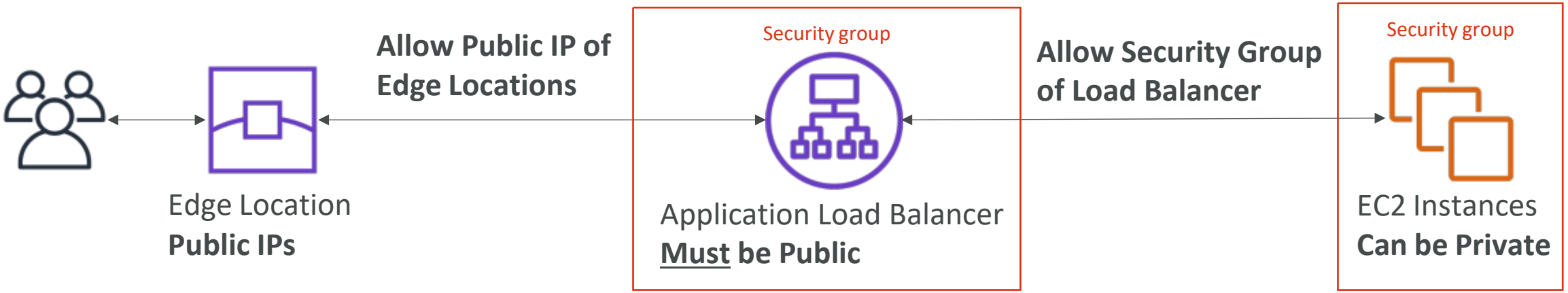
CloudFront – Maximize cache hits by separating static and dynamic distributions



No headers / session caching rules
Required for maximizing cache hits

CloudFront – ALB or EC2 as an origin





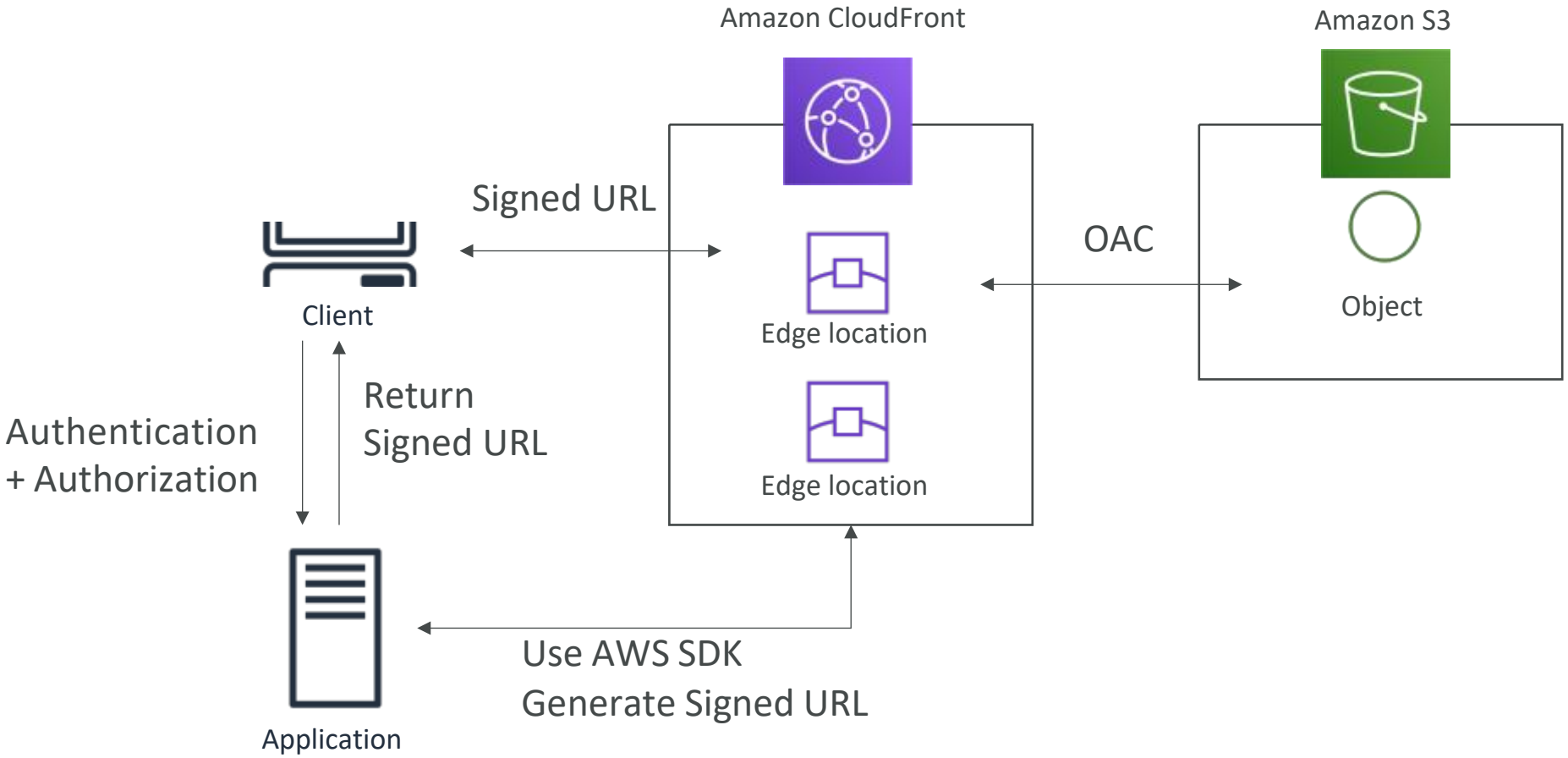
CloudFront Geo Restriction

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

CloudFront Signed URL / Signed Cookies

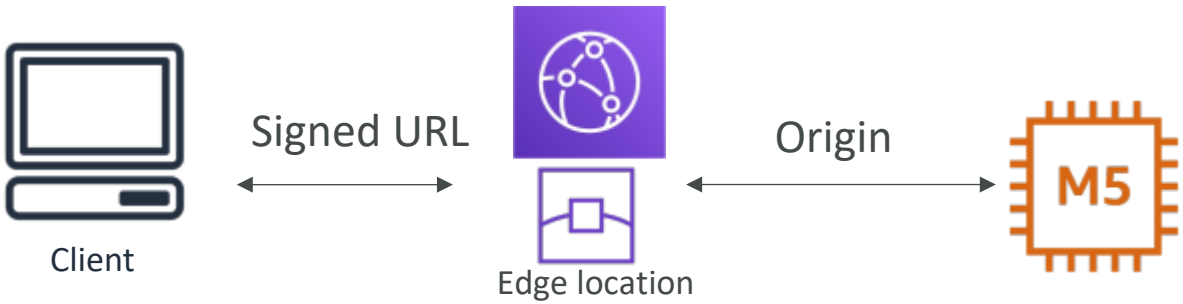
- You want to distribute paid shared content to premium users over the world
- We can use CloudFront Signed URL / Cookie. We attach a policy with:
  - Includes URL expiration
  - Includes IP ranges to access the data from
  - Trusted signers (which AWS accounts can create signed URLs)
- How long should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

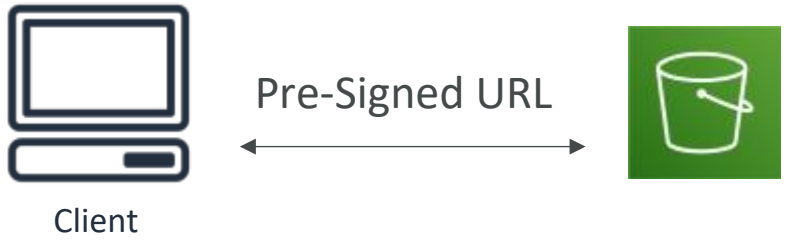
CloudFront Signed URL Diagram



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 Signed URL Process

- Two types of signers:
  - Either a trusted key group (recommended)
    - Can leverage APIs to create and rotate keys (and IAM for API security)
  - An AWS Account that contains a CloudFront Key Pair
    - Need to manage keys using **the root account and the AWS console**
    - Not recommended because you shouldn't use the root account for this
- In your CloudFront distribution, create one or more **trusted key groups**
- You generate your own public / private key
  - The private key is used by your applications (e.g. EC2) to sign URLs
  - The public key (uploaded) is used by CloudFront to verify URLs

CloudFront - Pricing

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

Regional Data Transfer Out to Internet (per GB)

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

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

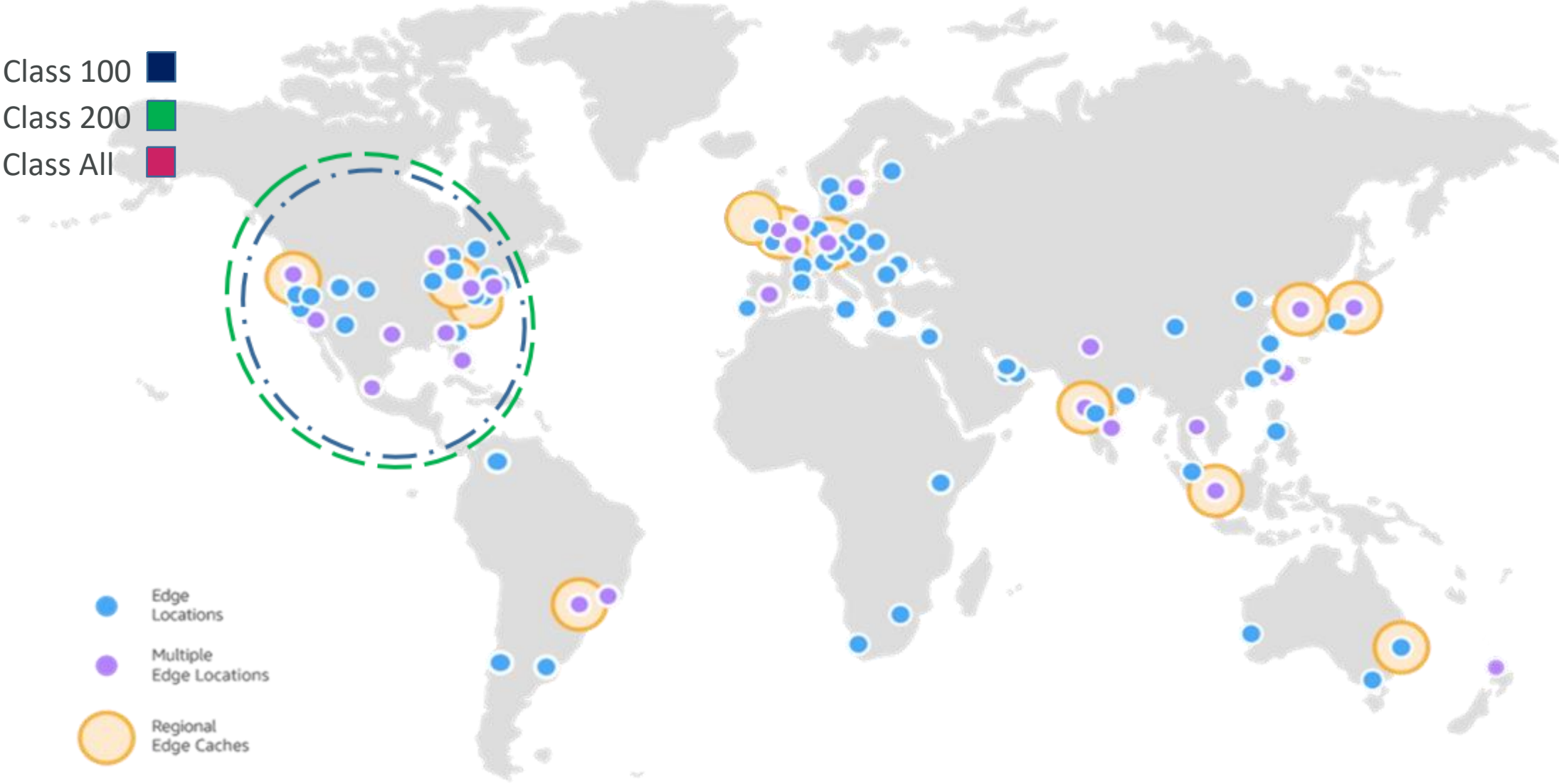
Edge Locations Included Within	**United States, Mexico, & Canada**	**Europe & Israel**	**South Africa, Kenya, & Middle East**	**South America**	**Japan**	**Australia & New Zealand**	**Hong Kong, Philippines, Singapore, South Korea, Taiwan, & Thailand**	**India**
Price Class All	Yes	Yes	Yes	Yes	Yes	Yes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

Prices Class 100

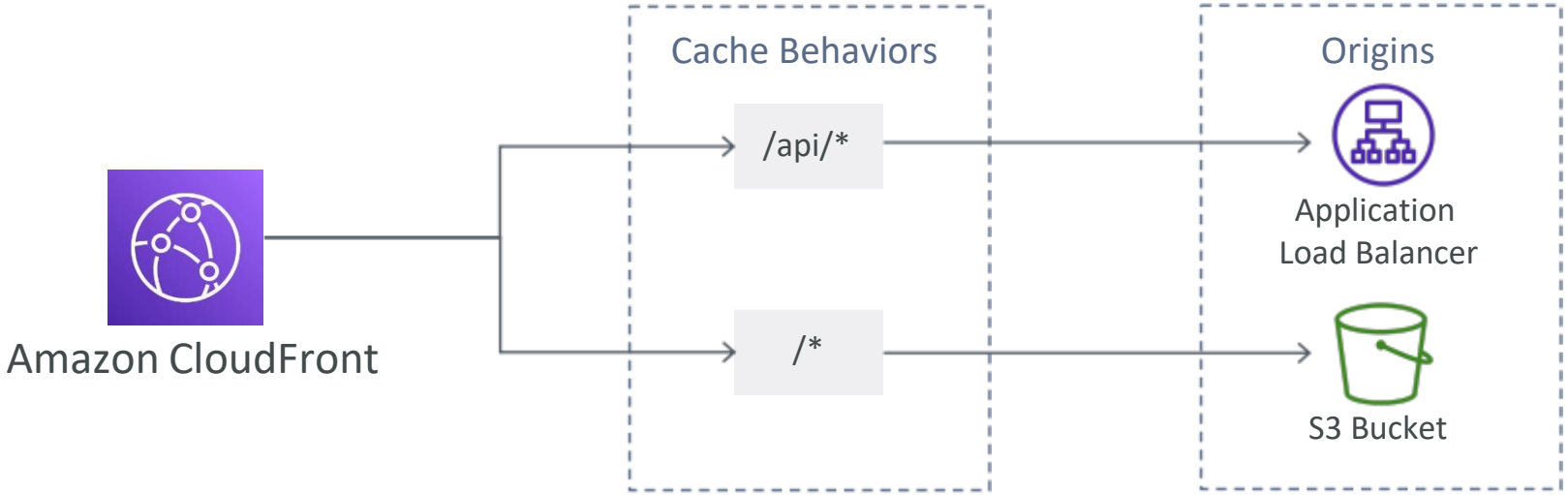
Prices Class 200

Prices Class All



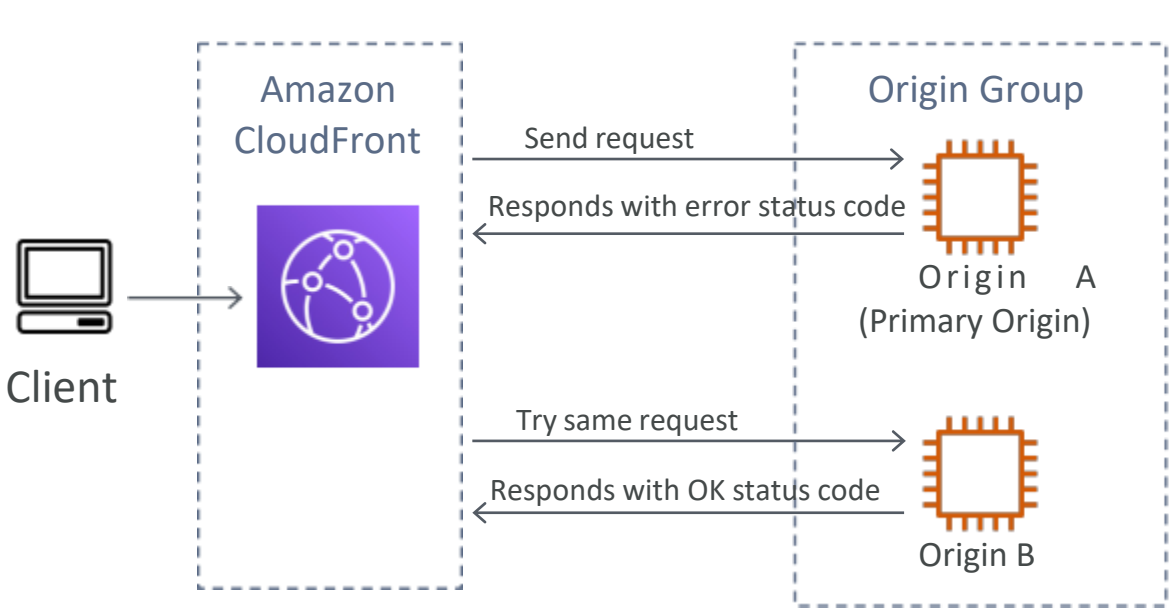
CloudFront – Multiple Origin

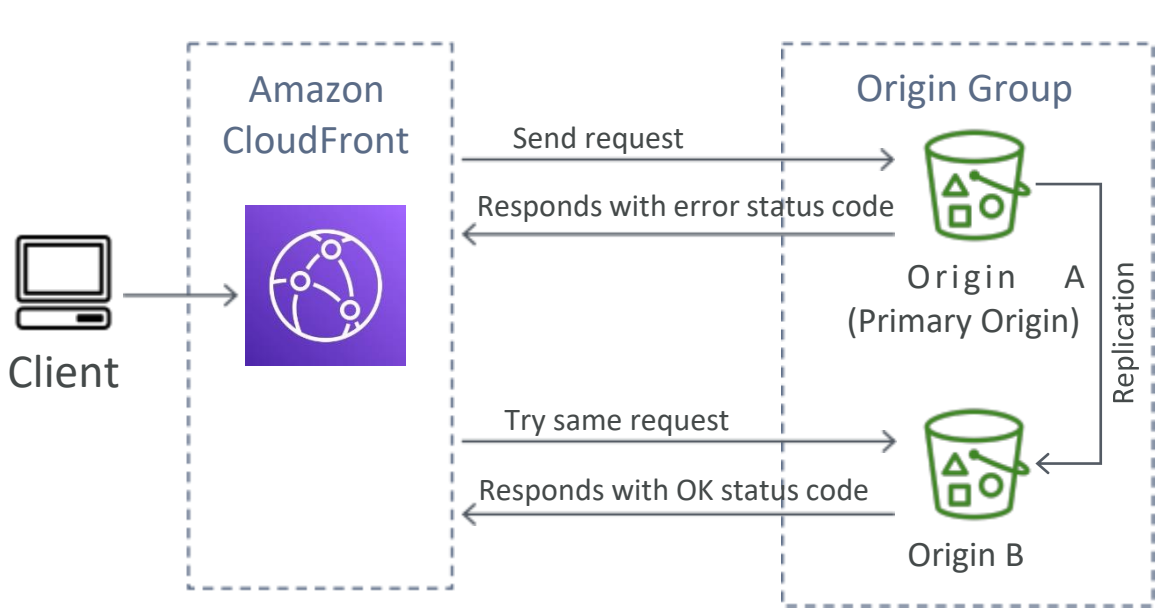
- To route to different kind of origins based on the content type
- Based on path pattern:
  - /images/*
  - /api/*
  - /*



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

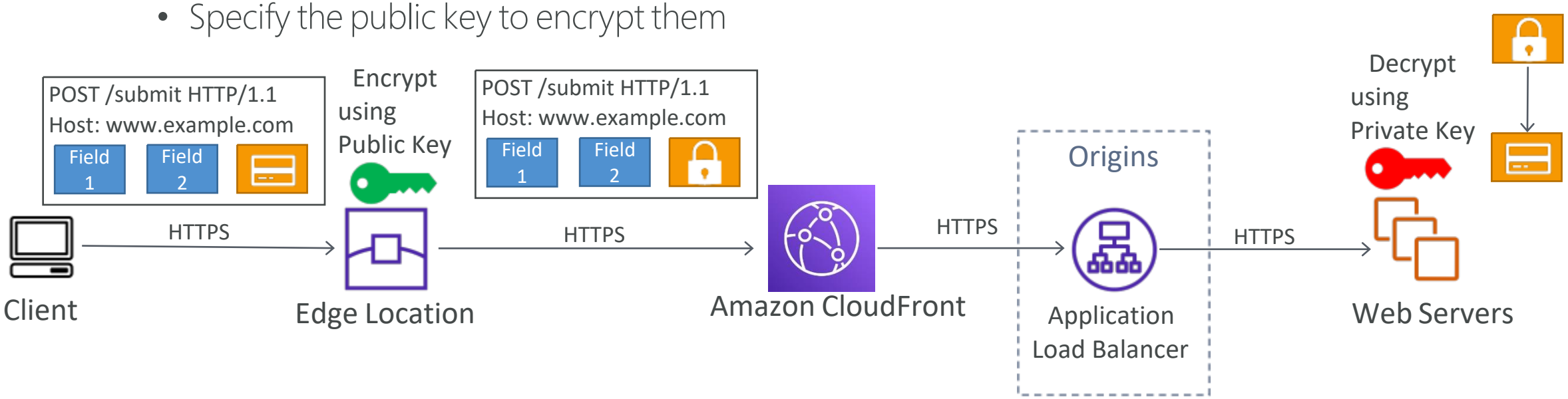




3 + CloudFront – Region-level High Availability

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

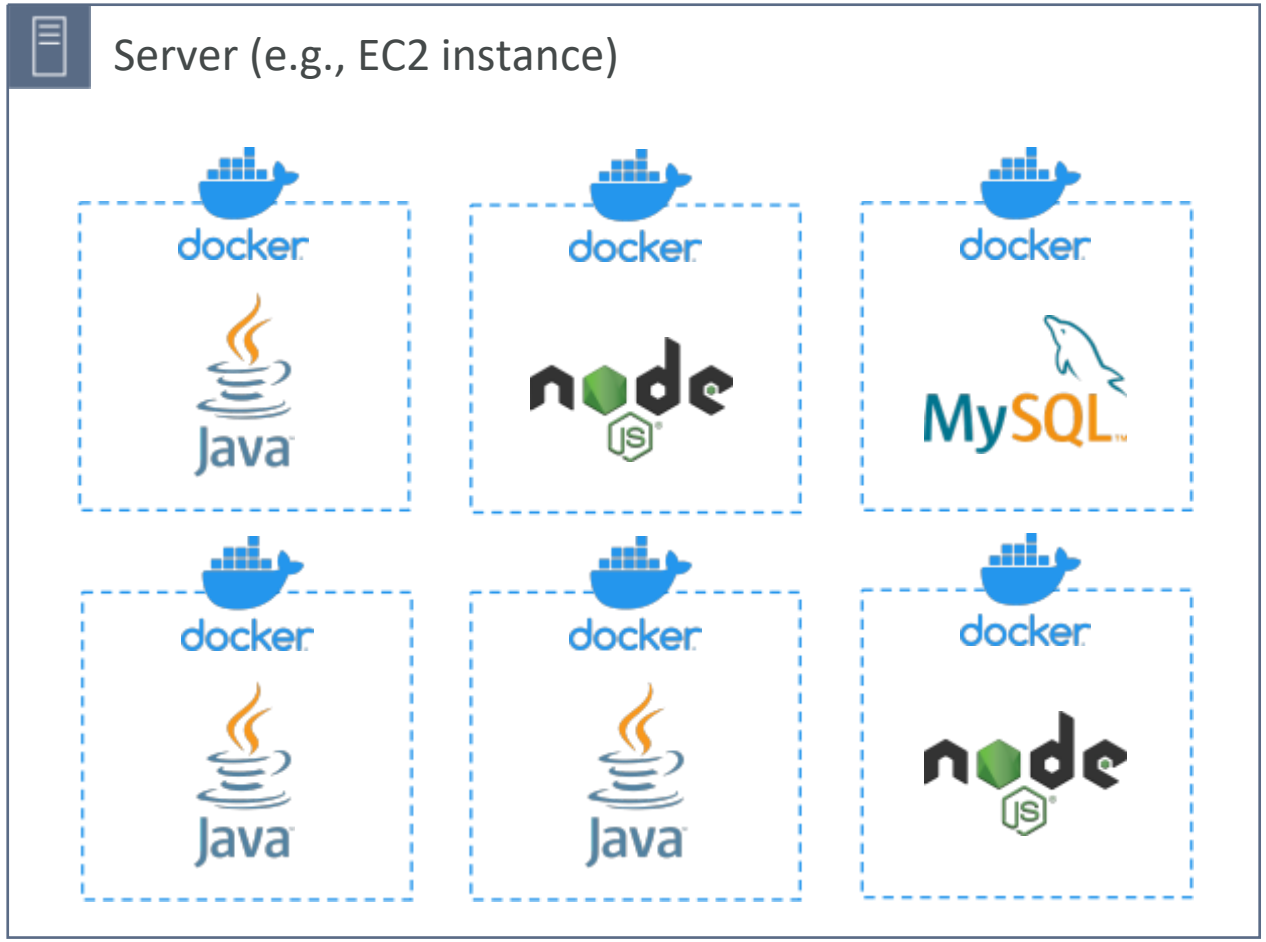


Container Section

What is Docker?

- Docker is a software development platform to deploy apps
- Apps are packaged in **containers** that can be run on any OS
- Apps run the same, regardless of where they're run
  - Any machine
  - No compatibility issues
  - Predictable behavior
  - Less work
  - Easier to maintain and deploy
  - Works with any language, any OS, any technology
- Use cases: microservices architecture, lift-and-shift apps from on-premises to the AWS cloud, …

Docker on an OS



Where are Docker images stored?

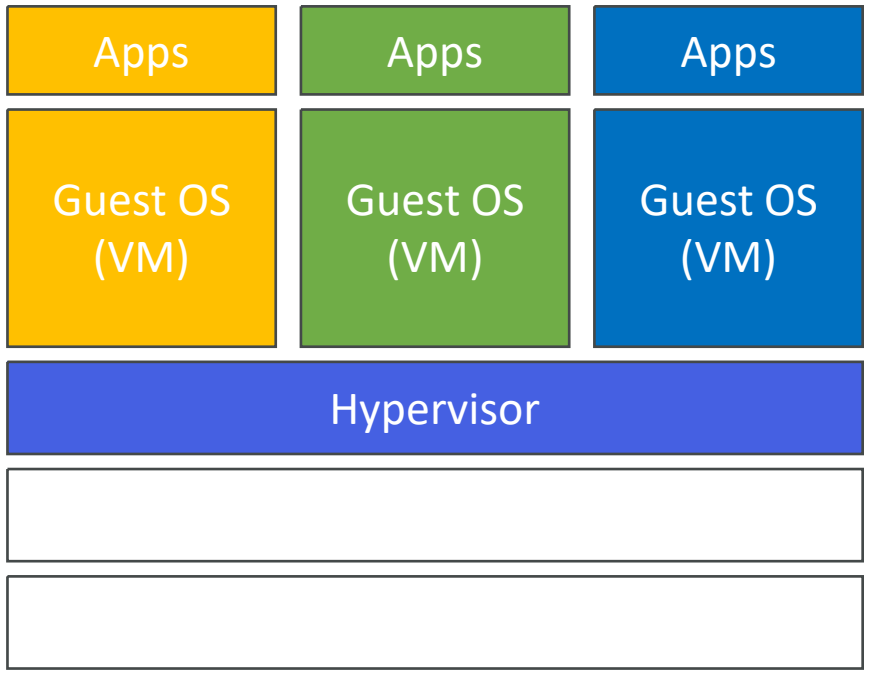
- Docker images are stored in Docker Repositories

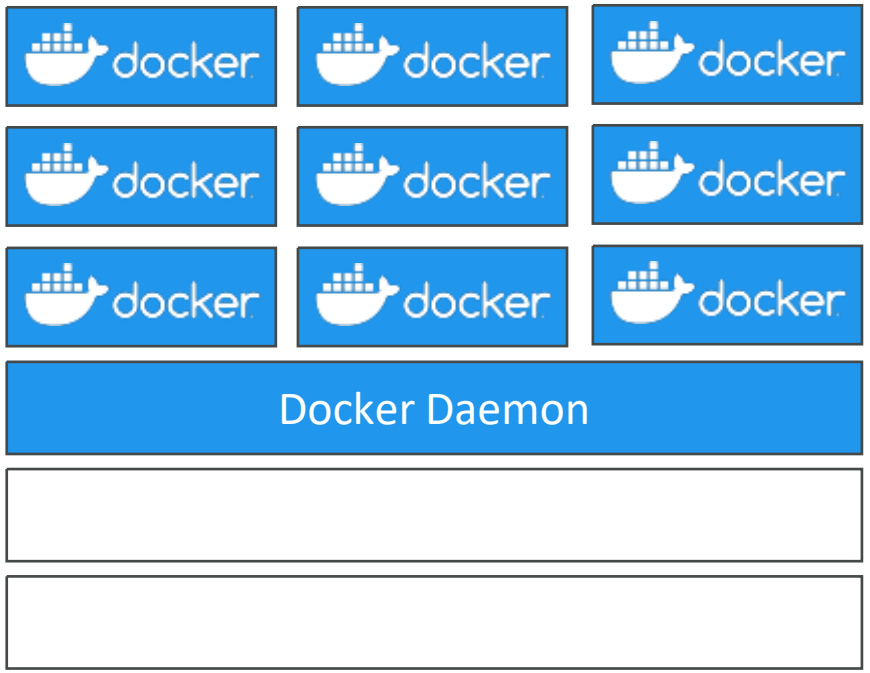
- **Docker Hub (**https://hub.docker.com**)**
  - **Public** repository
  - Find base images for many technologies or OS (e.g., Ubuntu, MySQL, …)

- **Amazon ECR (Amazon Elastic Container Registry)**
  - **Private** repository
  - **Public** repository (**Amazon ECR Public Gallery** https://gallery.ecr.aws)

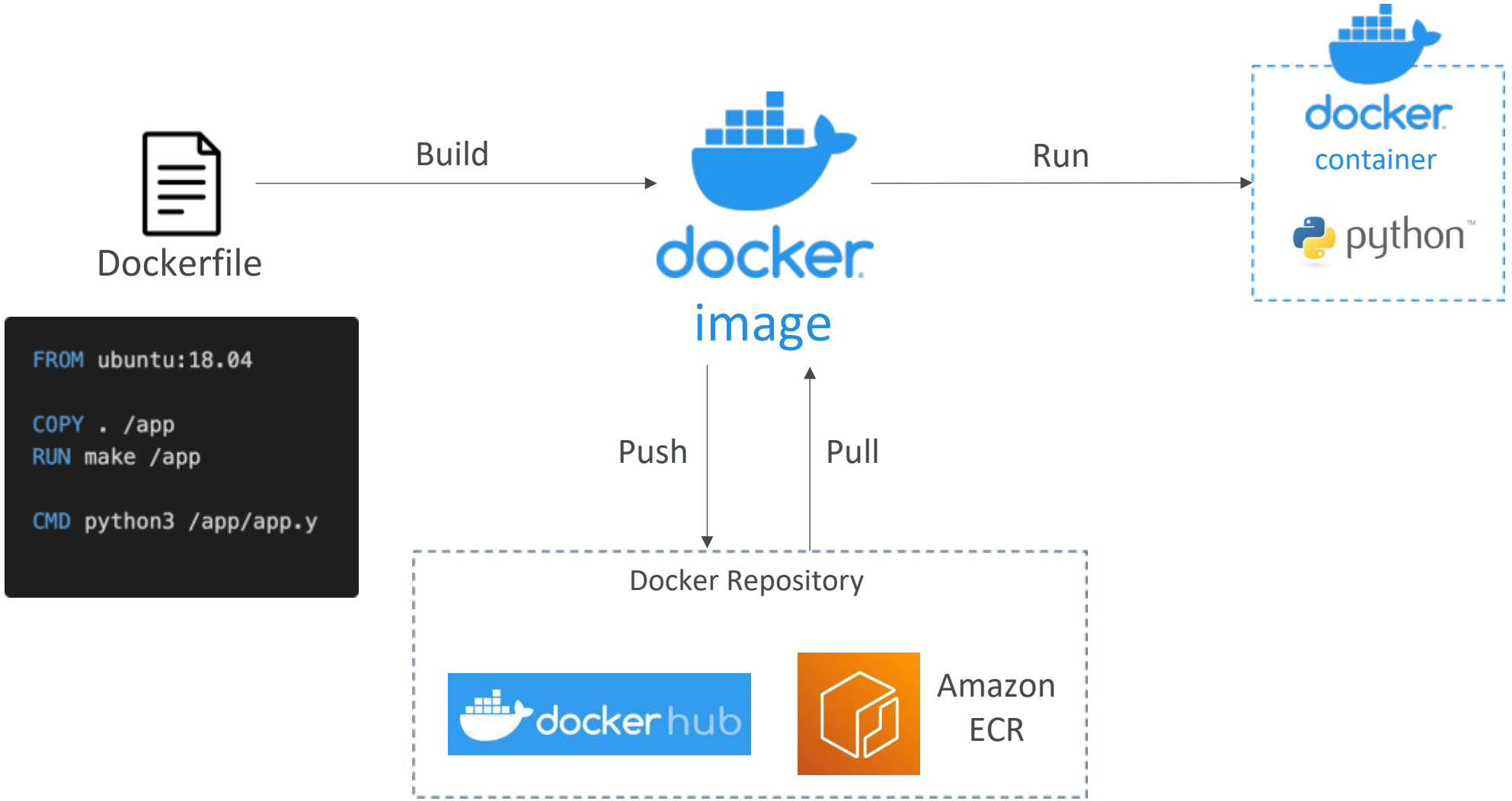
Docker vs. Virtual Machines

- Docker is "sort of" a virtualization technology, but not exactly
- Resources are shared with the host => many containers on one server





Getting Started with Docker



Docker Containers Management on AWS

- **Amazon Elastic Container Service (Amazon ECS)**
  - Amazon's own container platform

Amazon ECS

- **Amazon Elastic Kubernetes Service (Amazon EKS)**
  - Amazon's managed Kubernetes (open source)

Amazon EKS

- **AWS Fargate**
  - Amazon's own Serverless container platform
  - Works with ECS and with EKS

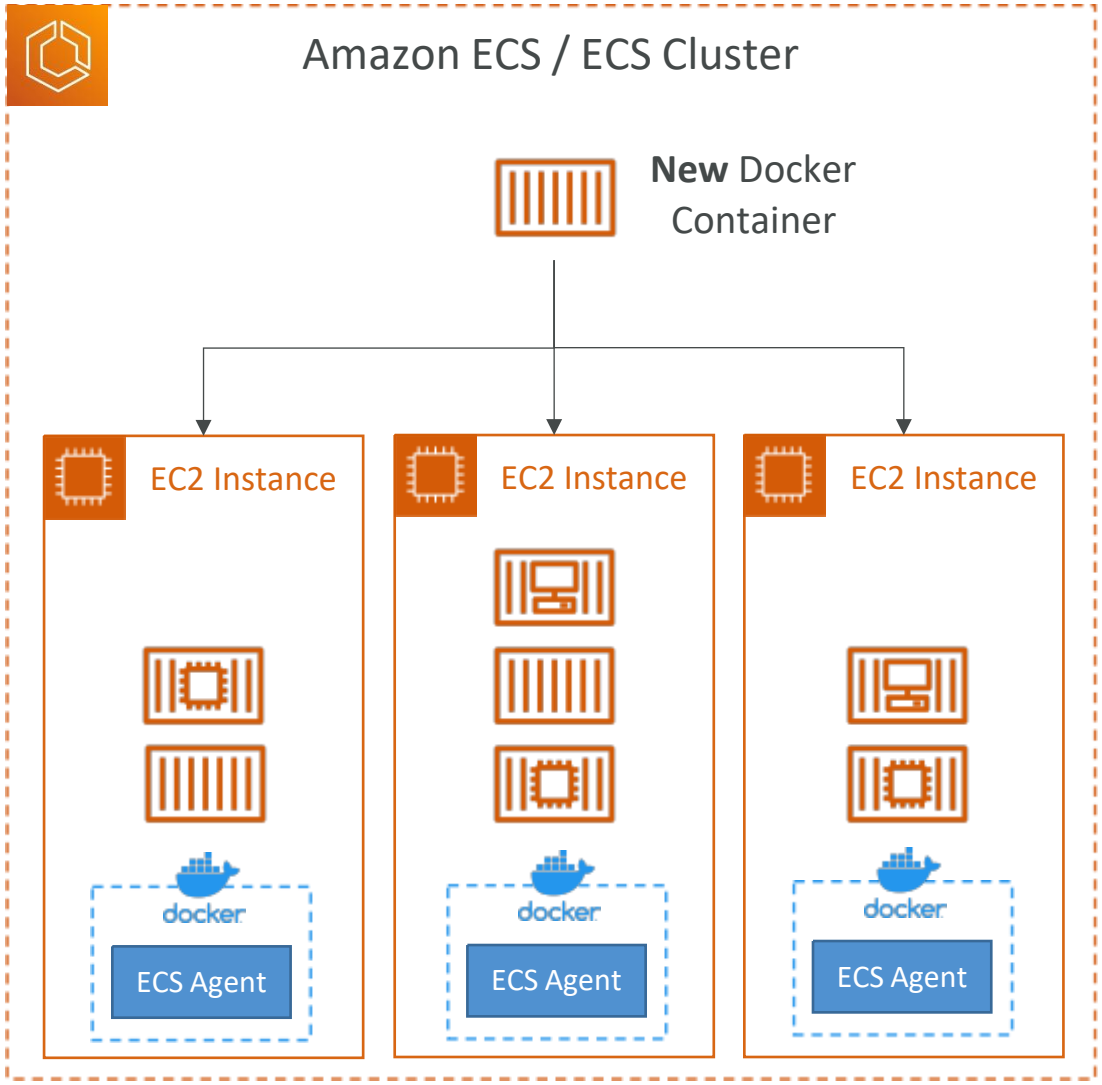
AWS Fargate

- **Amazon ECR:**
  - Store container images

Amazon ECR

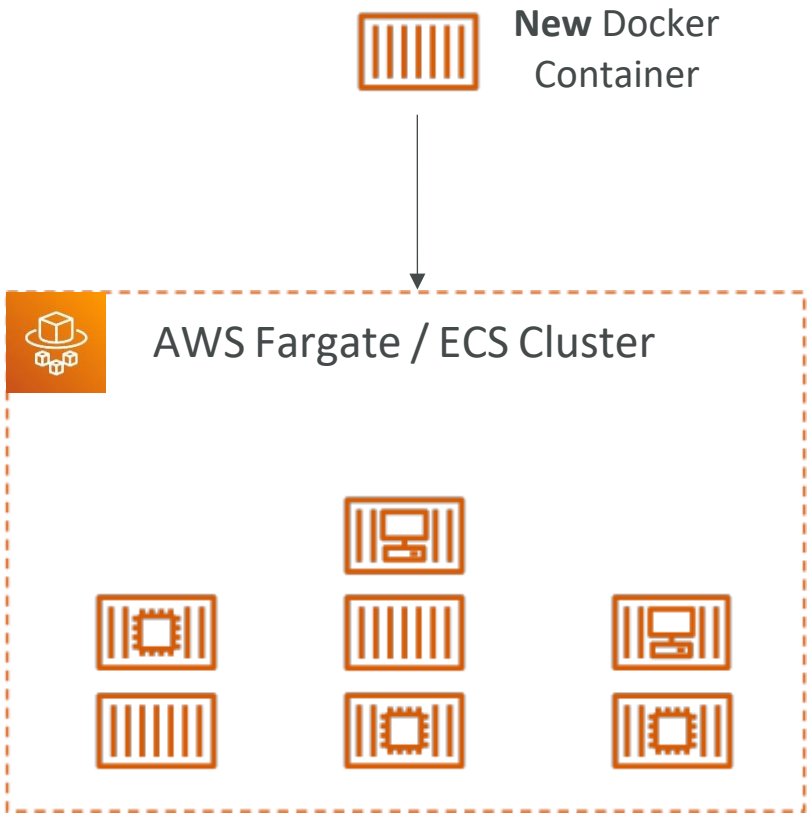
Amazon ECS - EC2 Launch Type

- ECS = Elastic Container Service
- Launch Docker containers on AWS = Launch **ECS Tasks** on ECS Clusters
- **EC2 Launch Type you must provision & maintain the infrastructure (the EC2 instances)**
- Each EC2 Instance must run the ECS Agent to register in the ECS Cluster
- AWS takes care of starting / stopping containers



Amazon ECS – Fargate Launch Type

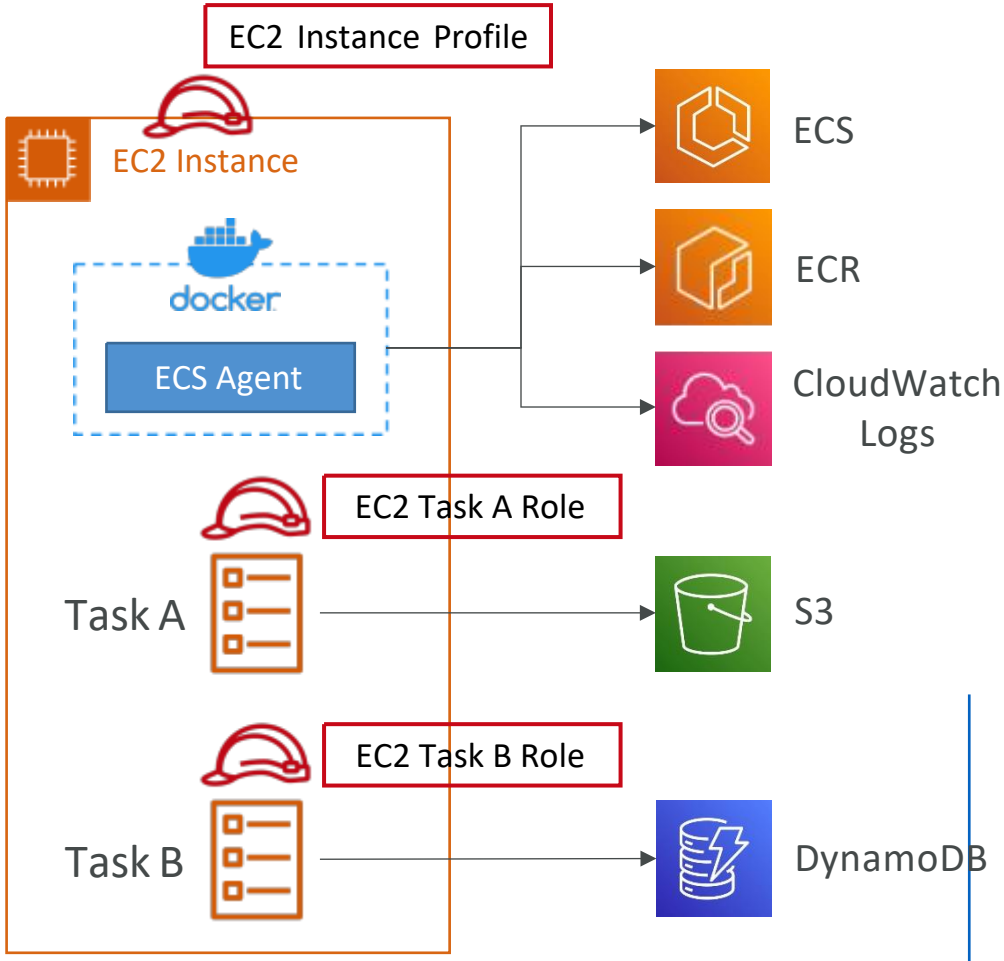
- Launch Docker containers on AWS
- **You do not provision the infrastructure (no EC2 instances to manage)**
- **It's all Serverless!**
- You just create task definitions
- AWS just runs ECS Tasks for you based on the CPU / RAM you need
- To scale, just increase the number of tasks. Simple - no more EC2 instances



Amazon ECS – IAM Roles for ECS

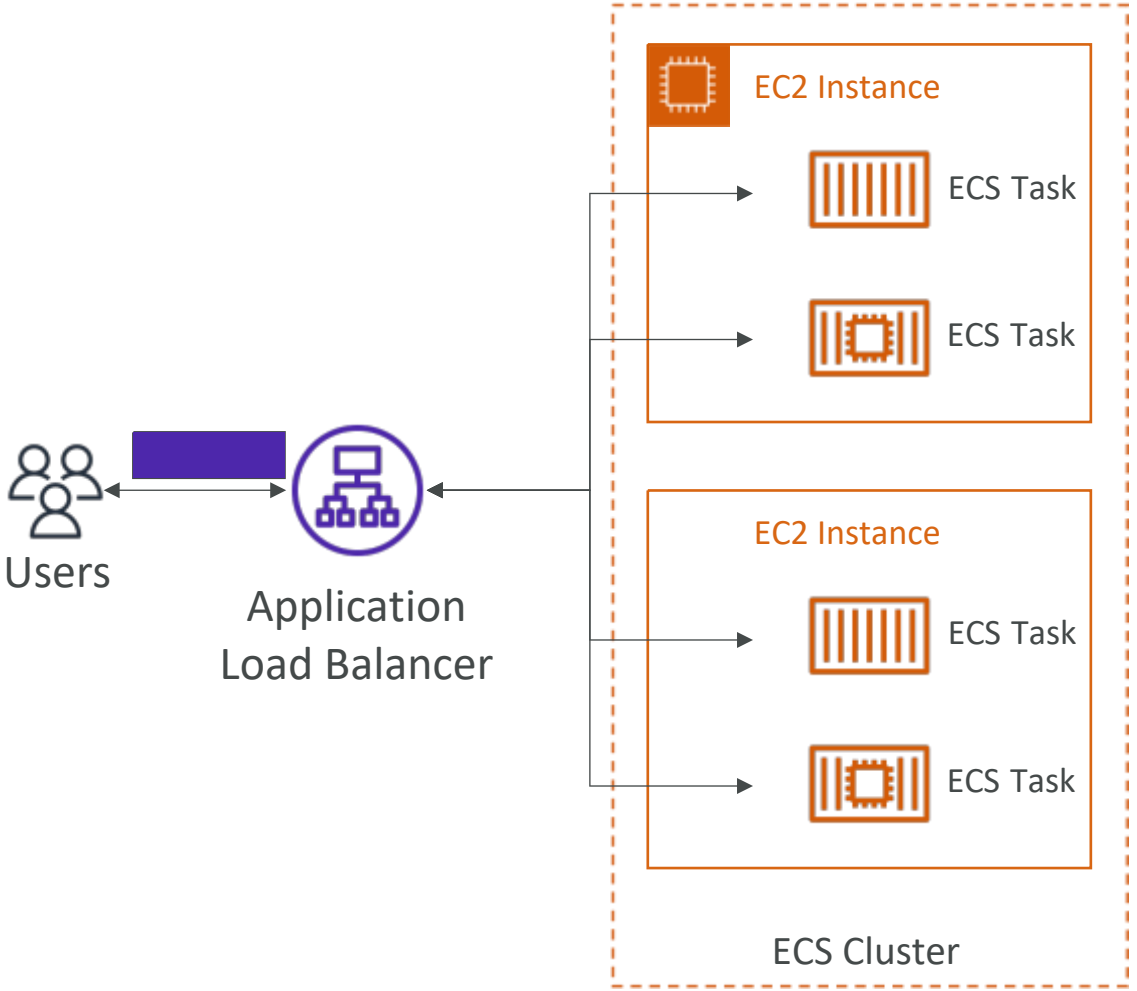
- **EC2 Instance Profile (EC2 Launch Type only):**
  - Used by the ECS agent
  - Makes API calls to ECS service
  - Send container logs to CloudWatch Logs
  - Pull Docker image from ECR
  - Reference sensitive data in Secrets Manager or SSM Parameter Store

- **ECS Task Role:**
  - Allows each task to have a specific role
  - Use different roles for the different ECS Services you run
  - Task Role is defined in the **task definition**



Amazon ECS – Load Balancer Integrations

- **Application Load Balancer** supported and works for most use cases
- **Network Load Balancer** recommended only for high throughput / high performance use cases, or to pair it with AWS Private Link
- **Elastic Load Balancer** supported but not recommended (no advanced features – no Fargate)

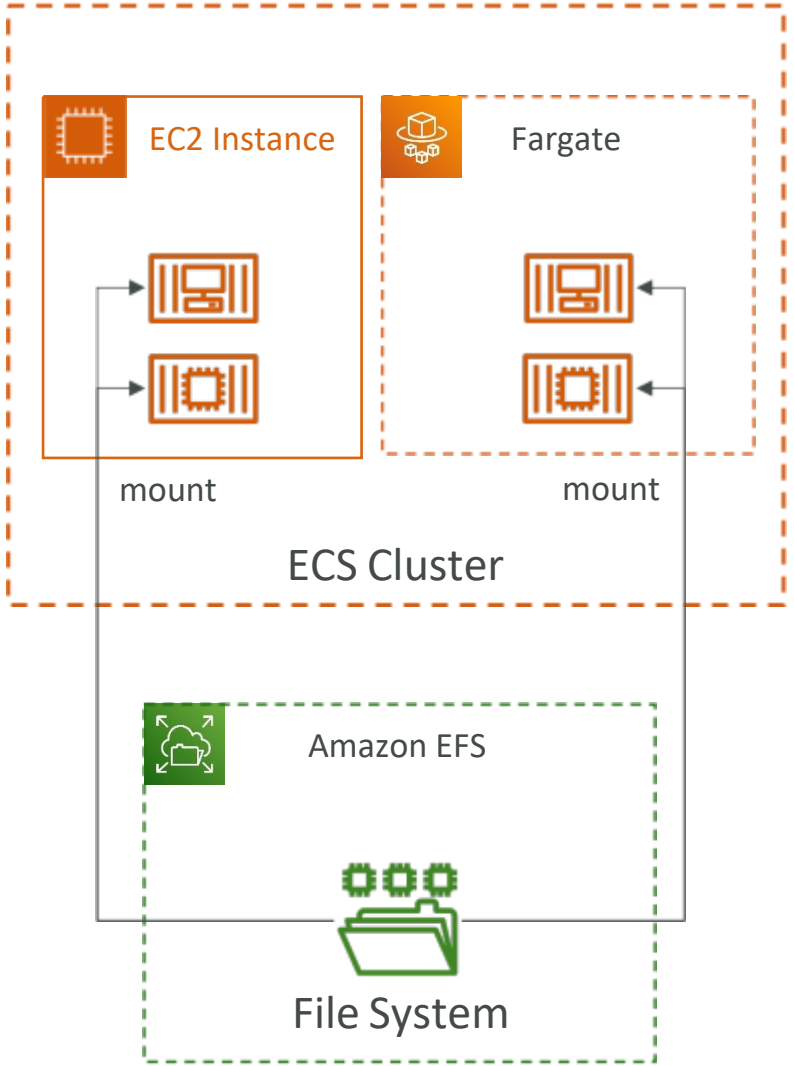


Amazon ECS – Data Volumes (EFS)

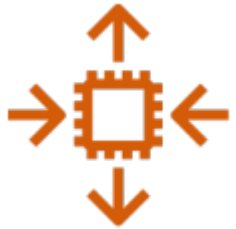
- Mount EFS file systems onto ECS tasks
- Works for both **EC2** and **Fargate** launch types
- Tasks running in any AZ will share the same data in the EFS file system
- **Fargate + EFS = Serverless**

- Use cases: persistent multi-AZ shared storage for your containers

- Note:
  - Amazon S3 cannot be mounted as a file system



ECS Service Auto Scaling



- Automatically increase/decrease the desired number of ECS tasks

- Amazon ECS Auto Scaling uses **AWS Application Auto Scaling**
  - ECS Service Average CPU Utilization
    ECS Service Average Memory Utilization - Scale on RAM
    ALB Request Count Per Target – metric coming from the ALB

- **Target Tracking –** scale based on target value for a specific CloudWatch metric
- **Step Scaling –** scale based on a specified CloudWatch Alarm
- **Scheduled Scaling –** scale based on a specified date/time (predictable changes)

- ECS Service Auto Scaling (task level) ≠ EC2 Auto Scaling (EC2 instance level)
- Fargate Auto Scaling is much easier to setup (because **Serverless**)

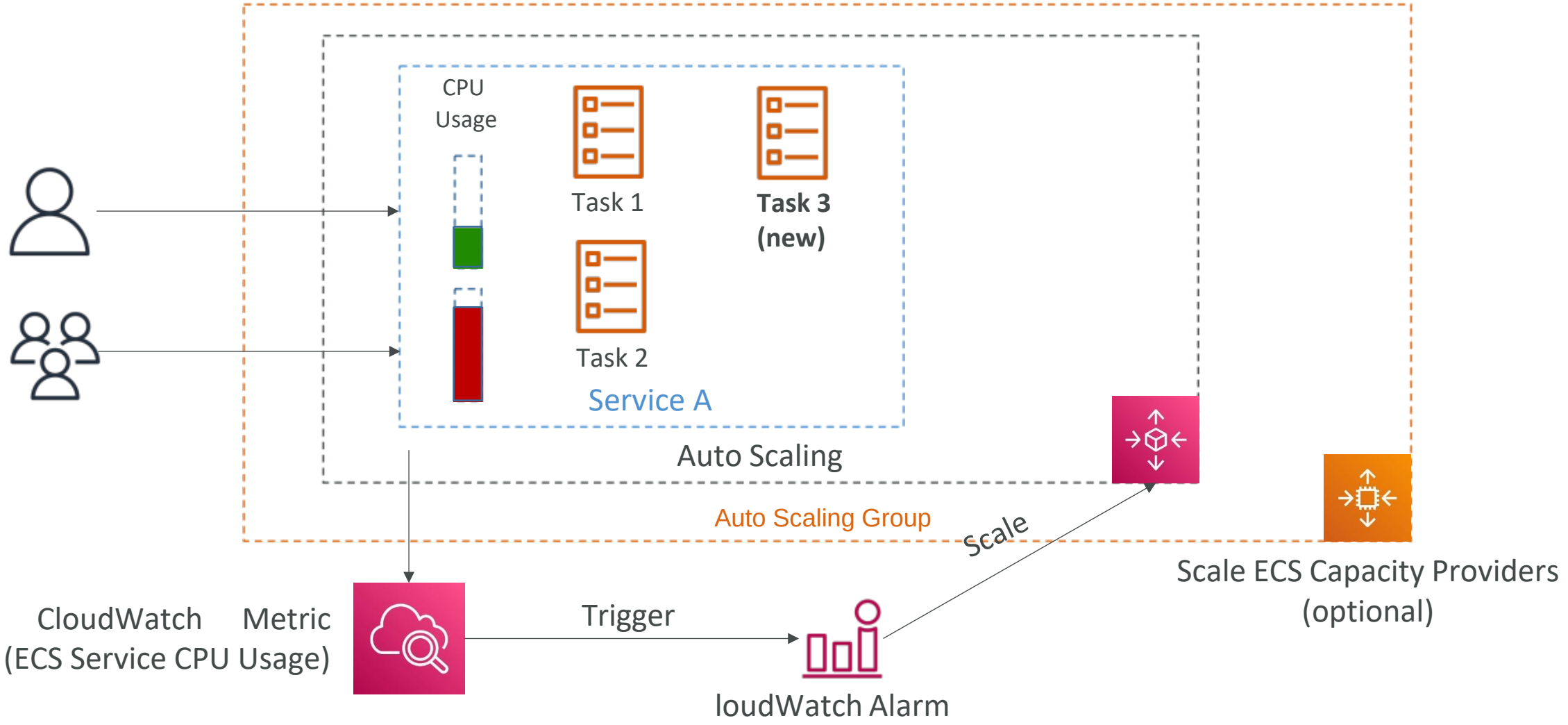
EC2 Launch Type – Auto Scaling EC2 Instances

- Accommodate ECS Service Scaling by adding underlying EC2 Instances

- **Auto Scaling Group Scaling**
  - Scale your ASG based on CPU Utilization
  - Add EC2 instances over time

- **ECS Cluster Capacity Provider**
  - Used to automatically provision and scale the infrastructure for your ECS Tasks
  - Capacity Provider paired with an Auto Scaling Group
  - Add EC2 Instances when you're missing capacity (CPU, RAM…)

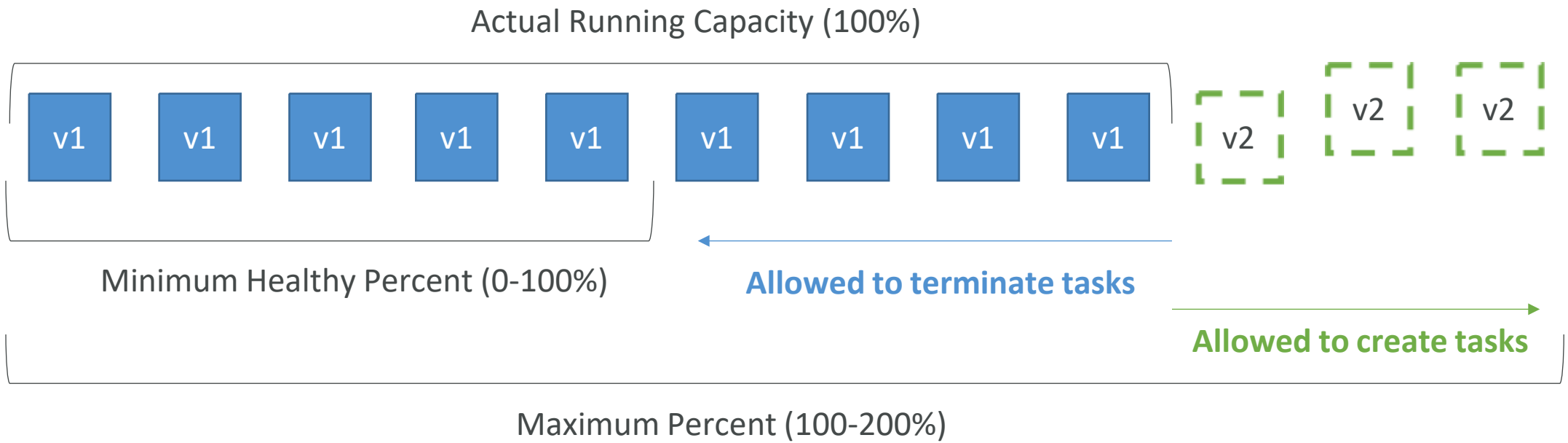
ECS Scaling – Service CPU Usage Example



ECS Rolling Updates

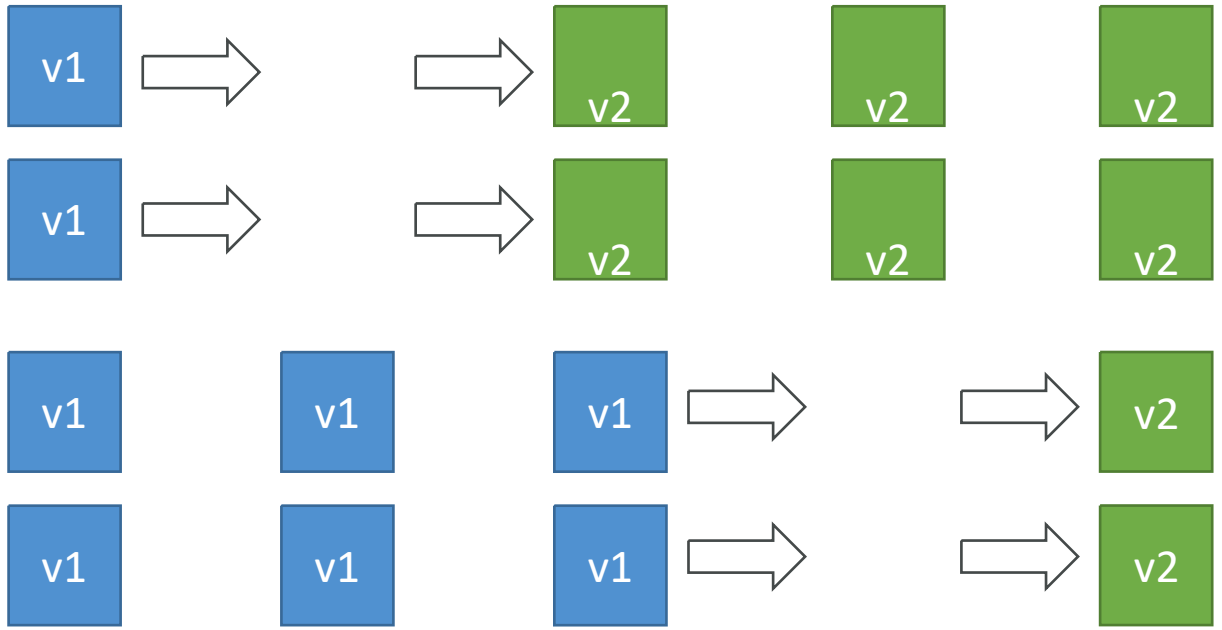
- When updating from v1 to v2, we can control how many tasks can be started and stopped, and in which order

ECS Service update screen



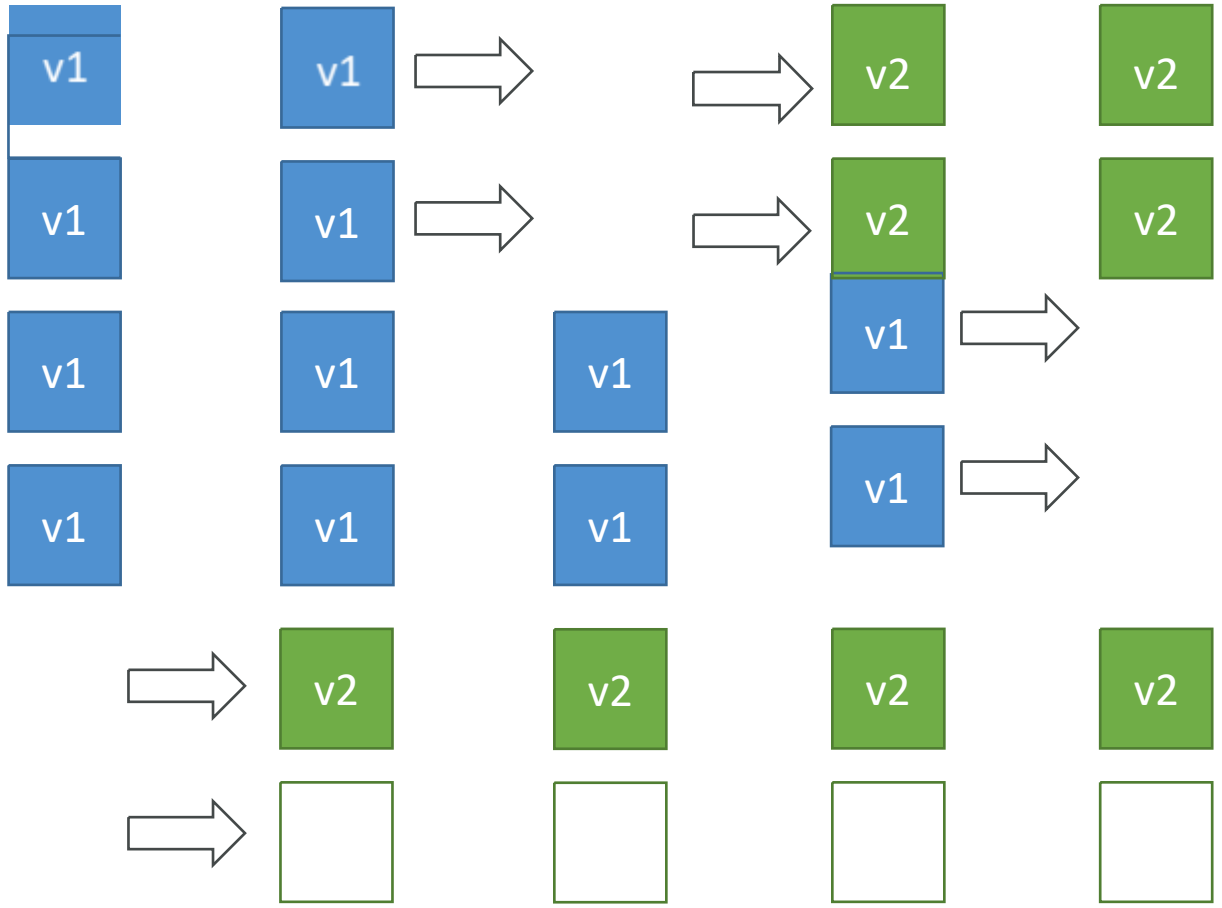
ECS Rolling Update – **Min 50%**, Max 100%

- Starting number of tasks: 4

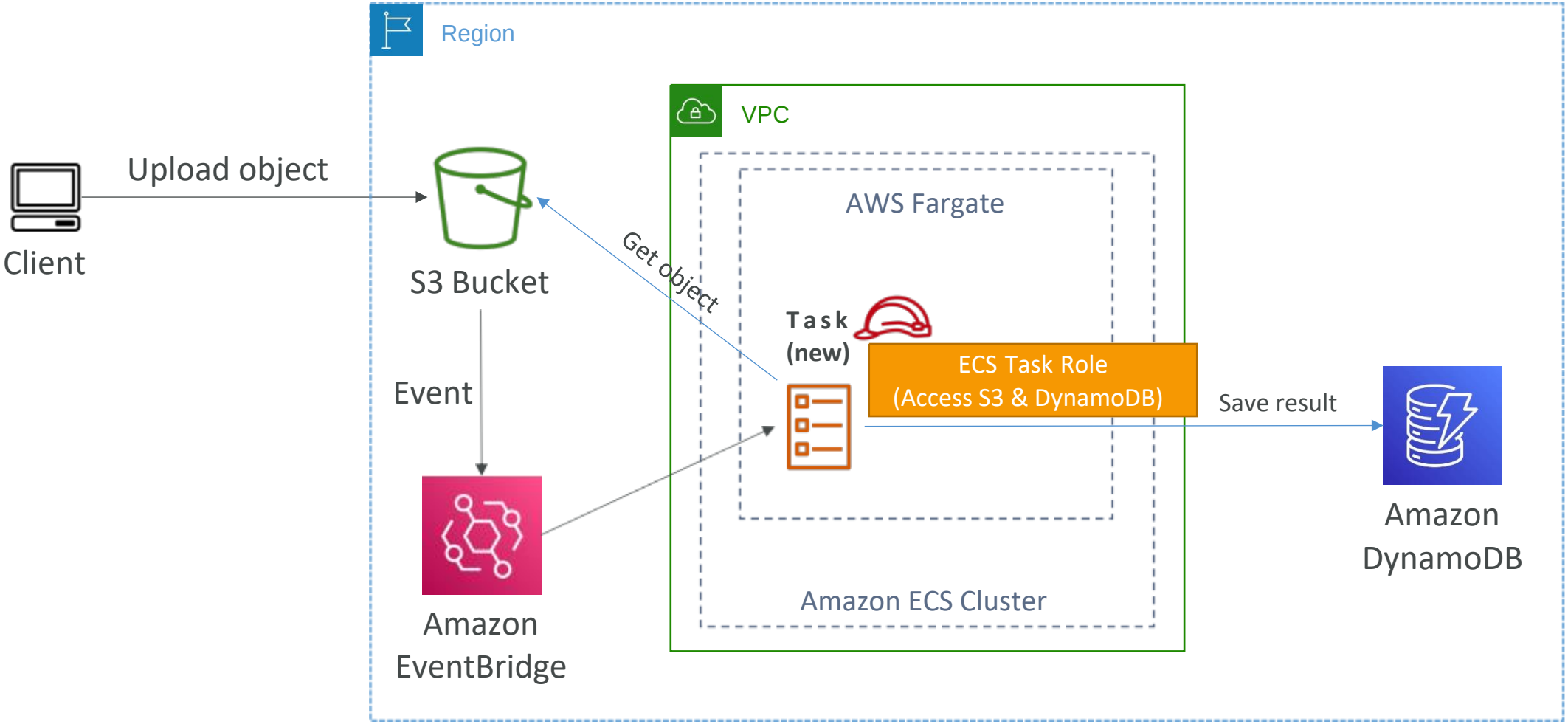


ECS Rolling Update – Min 100%, **Max 150%**

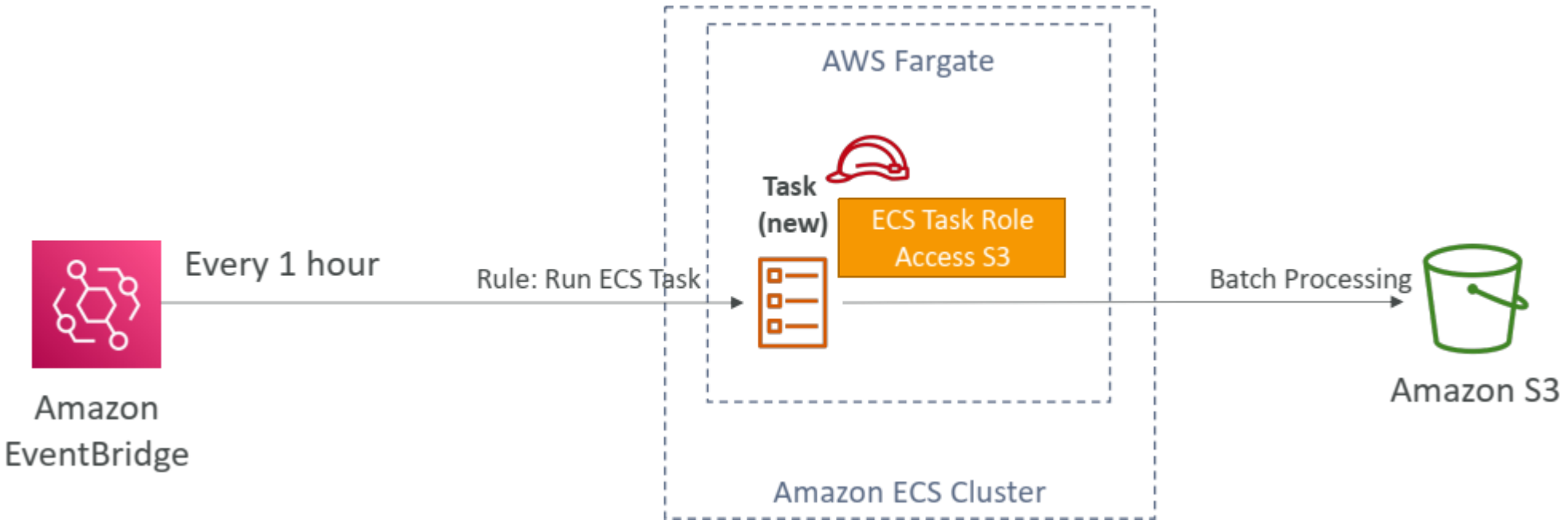
- Starting number of tasks: 4



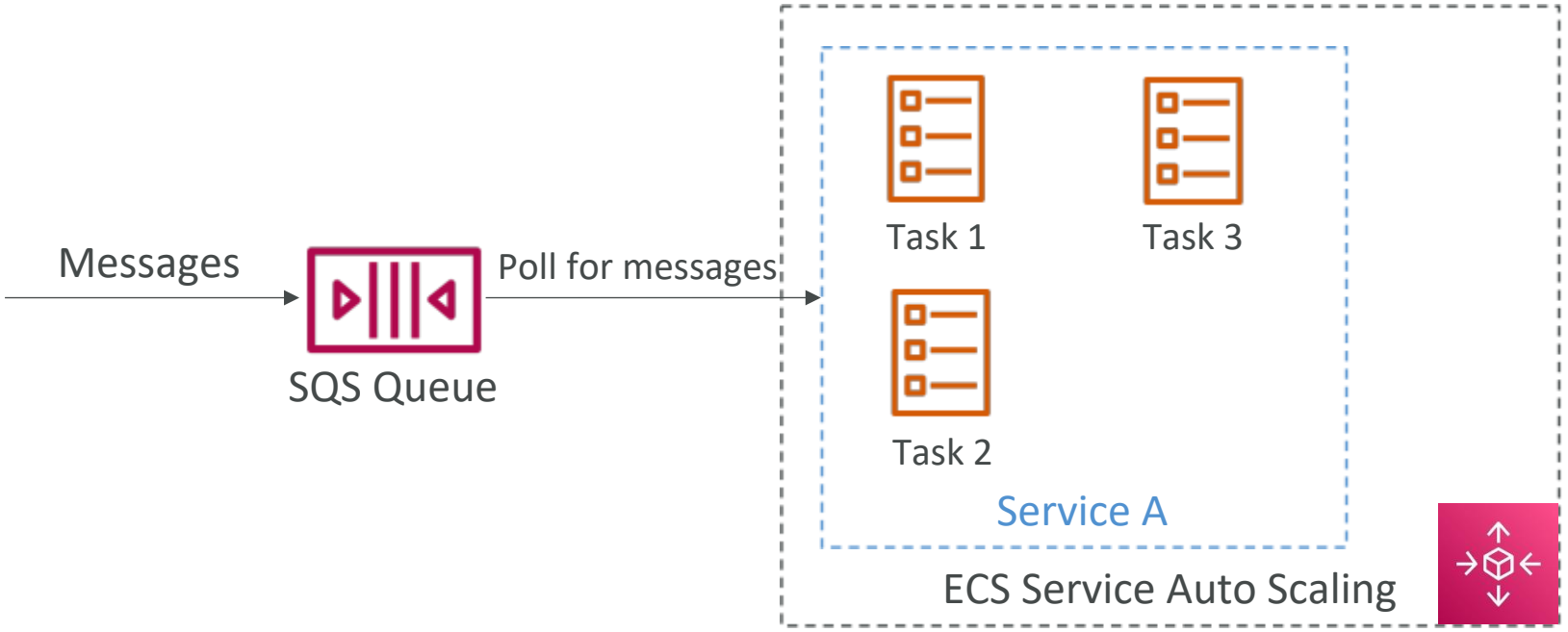
ECS tasks invoked by Event Bridge



ECS tasks invoked by Event Bridge Schedule

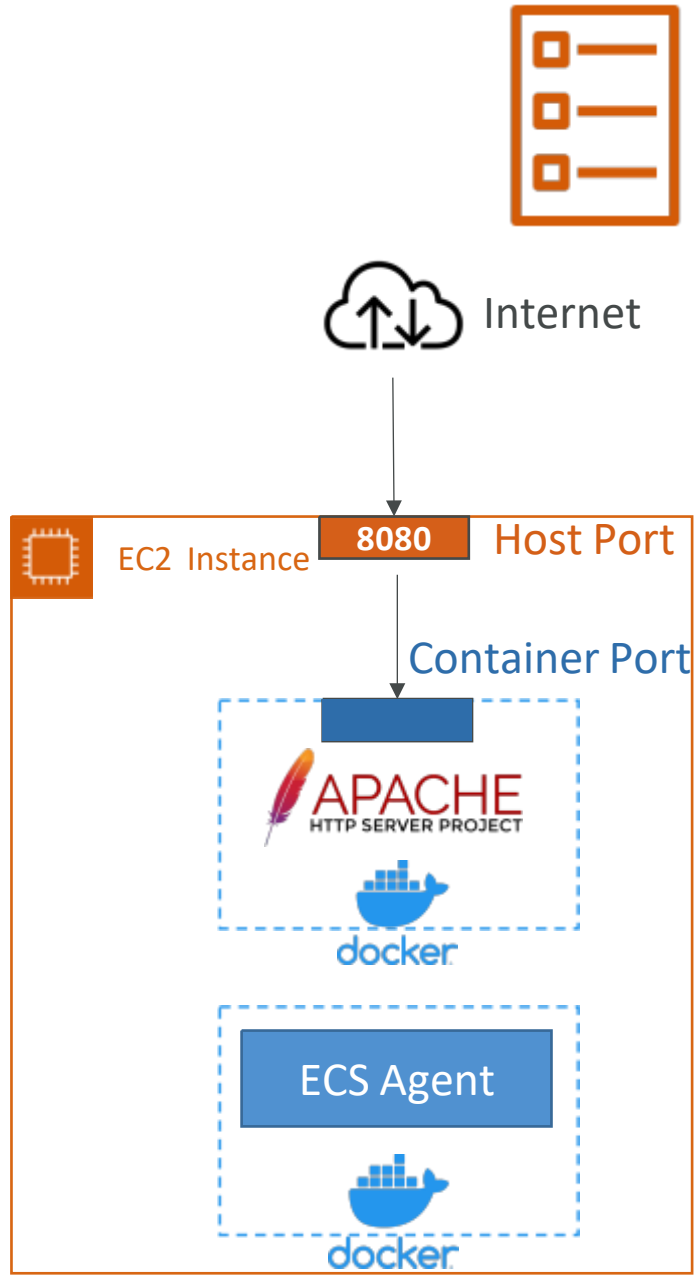


ECS – SQS Queue Example



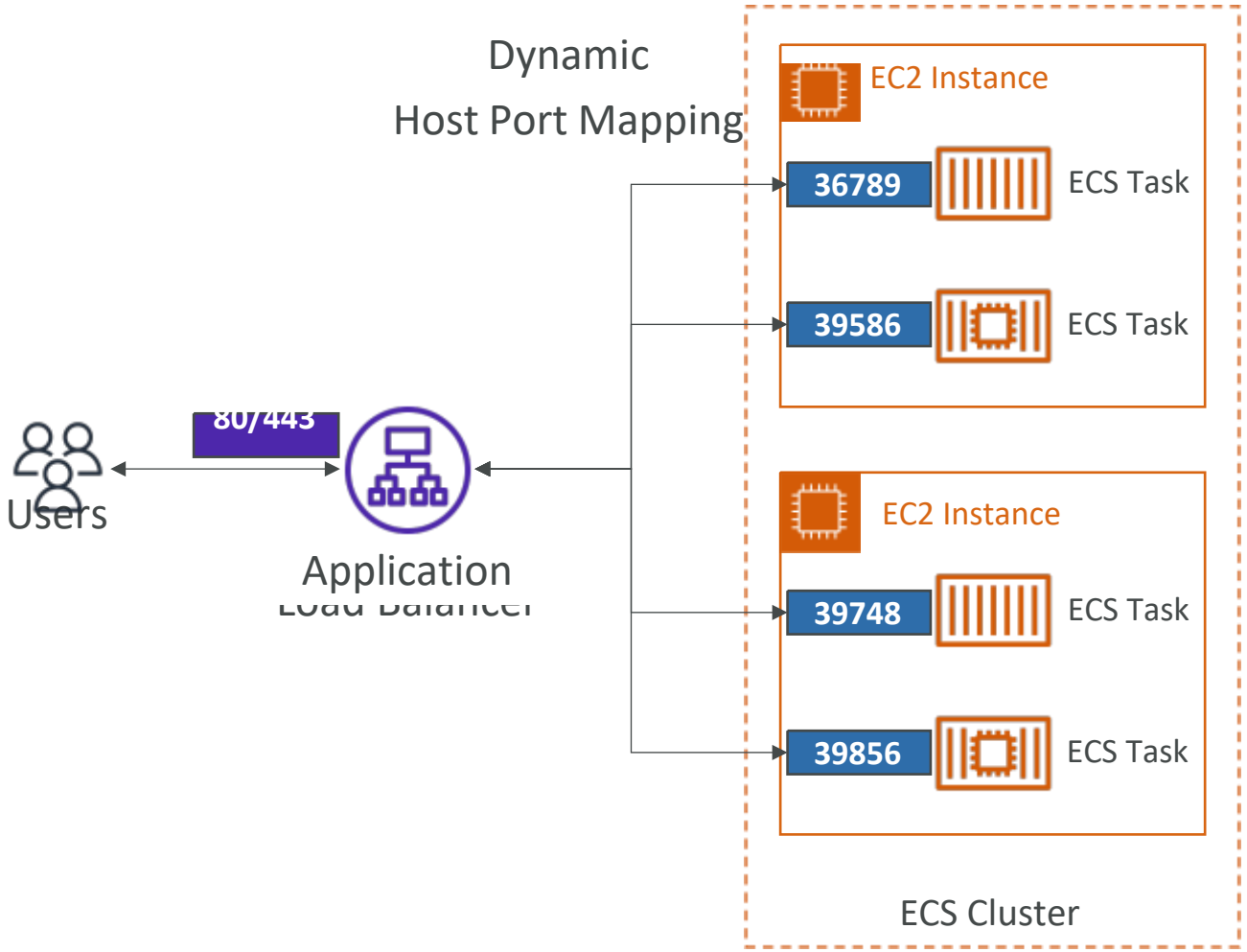
Amazon ECS –Task Definitions

- Task definitions are metadata in **JSON form** to tell ECS how to run a Docker container
- It contains crucial information, such as:
  - Image Name
  - Port Binding for Container and Host
  - Memory and CPU required
  - Environment variables
  - Networking information
  - IAM Role
  - Logging configuration (ex CloudWatch)
- Can define up to 10 containers in a Task Definition



Amazon ECS – Load Balancing (EC2 Launch Type)

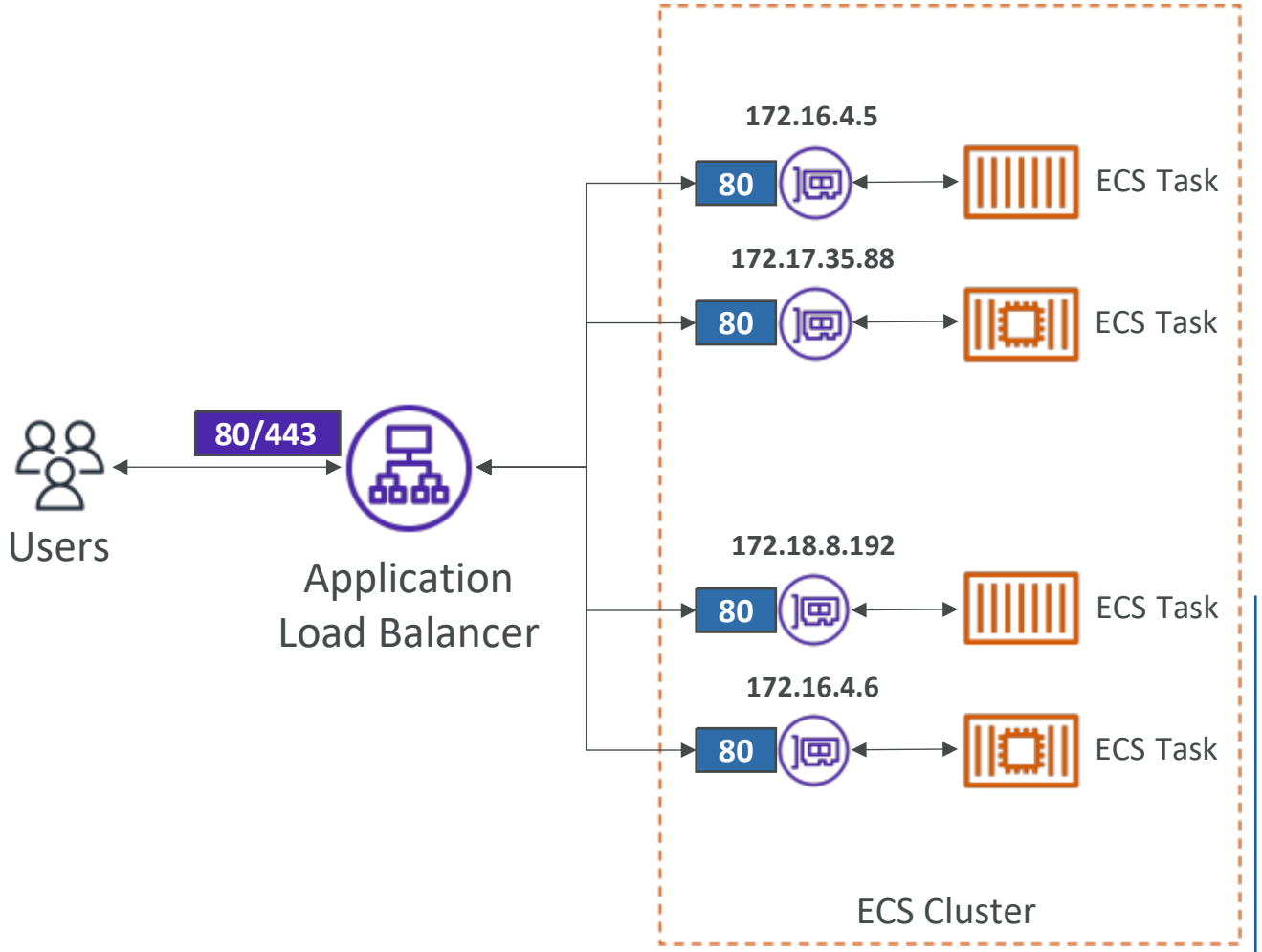
- We get a **Dynamic Host Port Mapping if you define only the container**
- The ALB finds the right port on your EC2 Instances
- **You must allow on the EC2 instance's Security Group any port from the ALB's Security Group**



Amazon ECS – Load Balancing (Fargate)

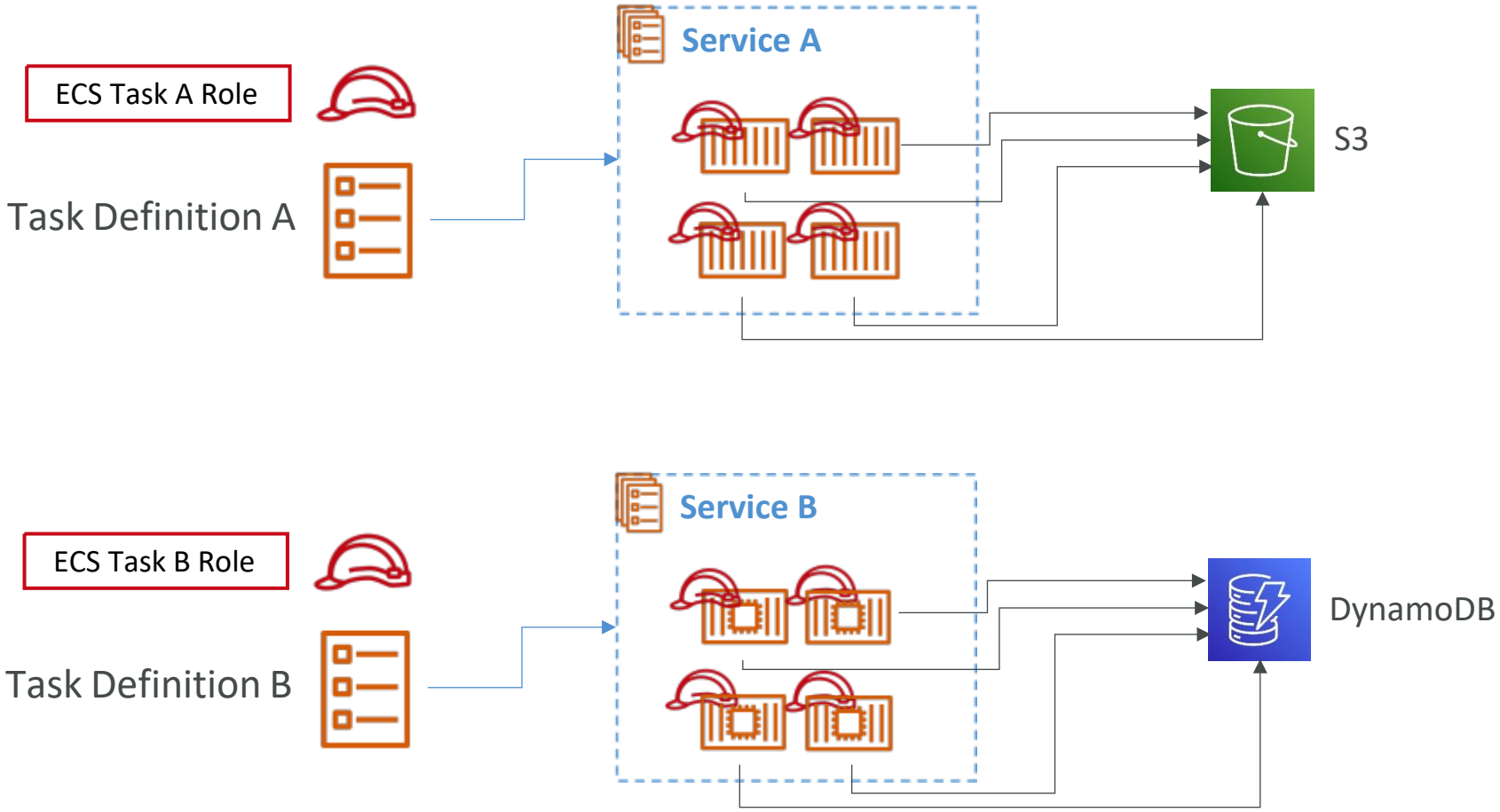
- Each task has a **unique private IP**
- **Only define the container port** (host port is not applicable)

- Example
  - **ECS ENI Security Group**
    - Allow port 80 from the ALB
  - **ALB Security Group**
    - Allow port 80/443 from web



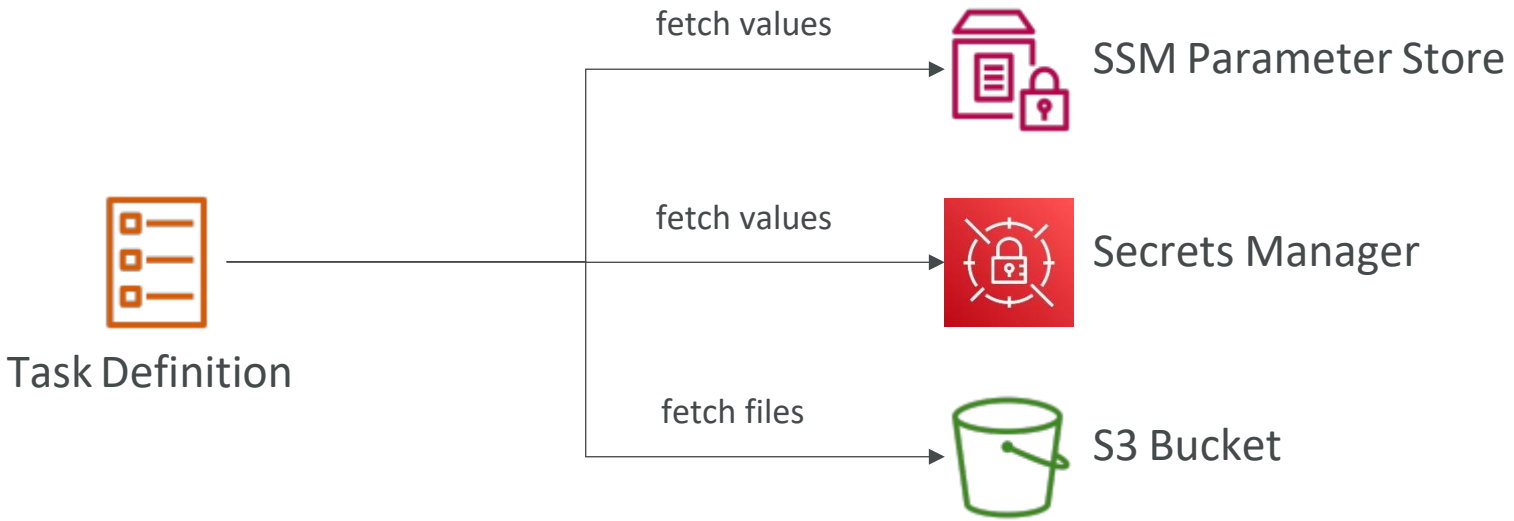
Amazon ECS

One IAM Role per Task Definition



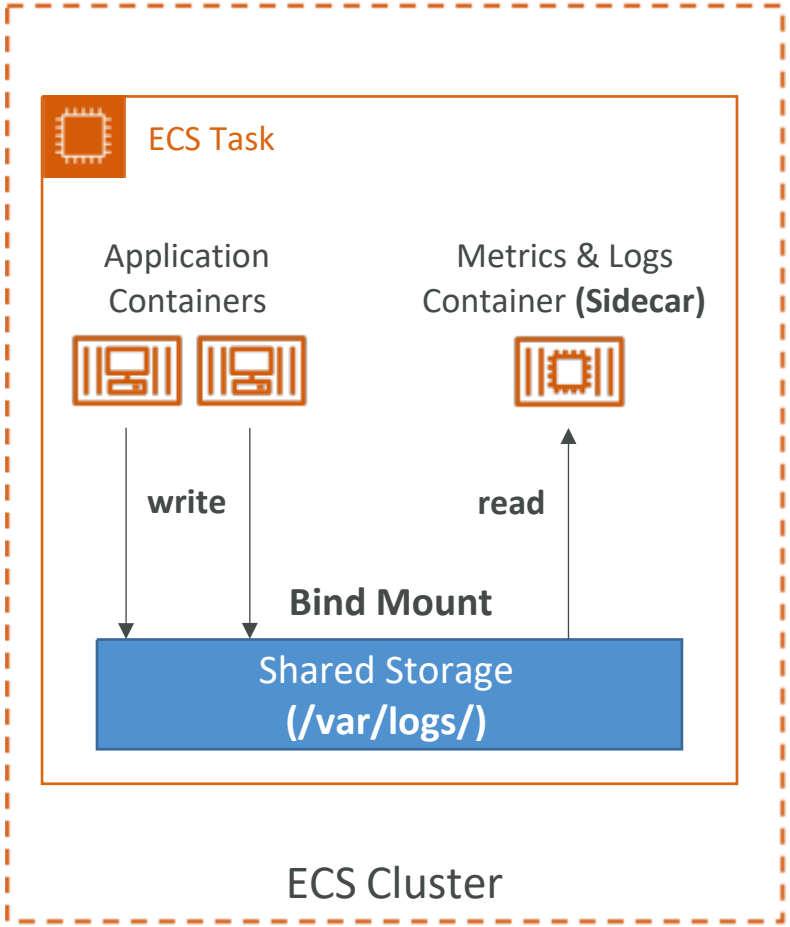
Amazon ECS – Environment Variables

- Environment Variable
  - **Hardcoded** – e.g., URLs
  - **SSM Parameter Store** – sensitive variables (e.g., API keys, shared configs)
  - **Secrets Manager** – sensitive variables (e.g., DB passwords)
- Environment Files (bulk) – Amazon S3



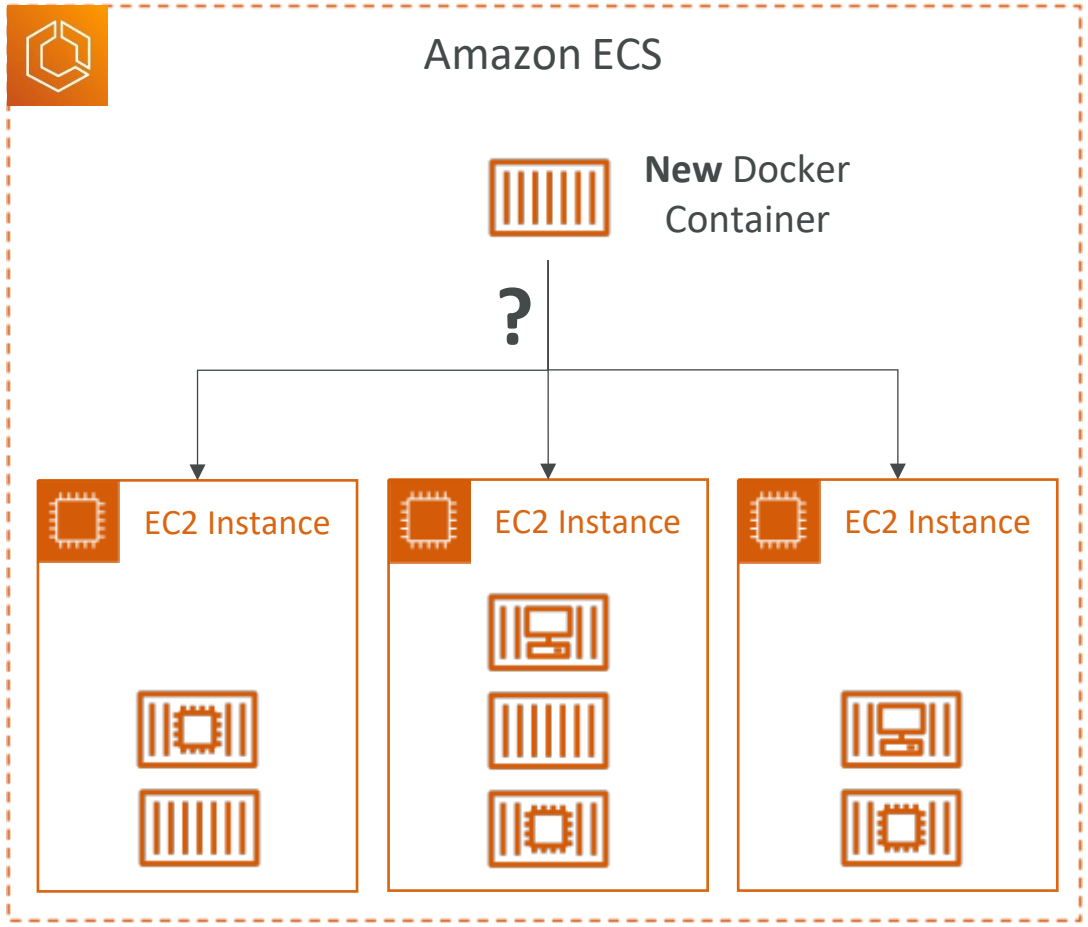
Amazon ECS – Data Volumes (Bind Mounts)

- Share data between multiple containers in the same Task Definition
- Works for both **EC2** and **Fargate** tasks
- **EC2 Tasks** – using EC2 instance storage
  - Data are tied to the lifecycle of the EC2 instance
- **Fargate Tasks** – using ephemeral storage
  - Data are tied to the container(s) using them
  - 20 GiB – 200 GiB (default 20 GiB)
- Use cases:
  - Share ephemeral data between multiple containers
  - "Sidecar" container pattern, where the "sidecar" container used to send metrics/logs to other destinations (separation of conerns)



Amazon ECS –Task Placement

- When an ECS task is started with EC2 Launch Type, ECS must determine where to place it, with the constraints of **CPU** and **memory (RAM)**
- Similarly, when a service scales in, ECS needs to determine which task to terminate
- You can define:
  - **Task Placement Strategy**
  - **Task Placement Constraints**
- **Note:** only for ECS Tasks with EC2 Launch Type **(Fargate not supported)**



Amazon ECS –Task Placement Process

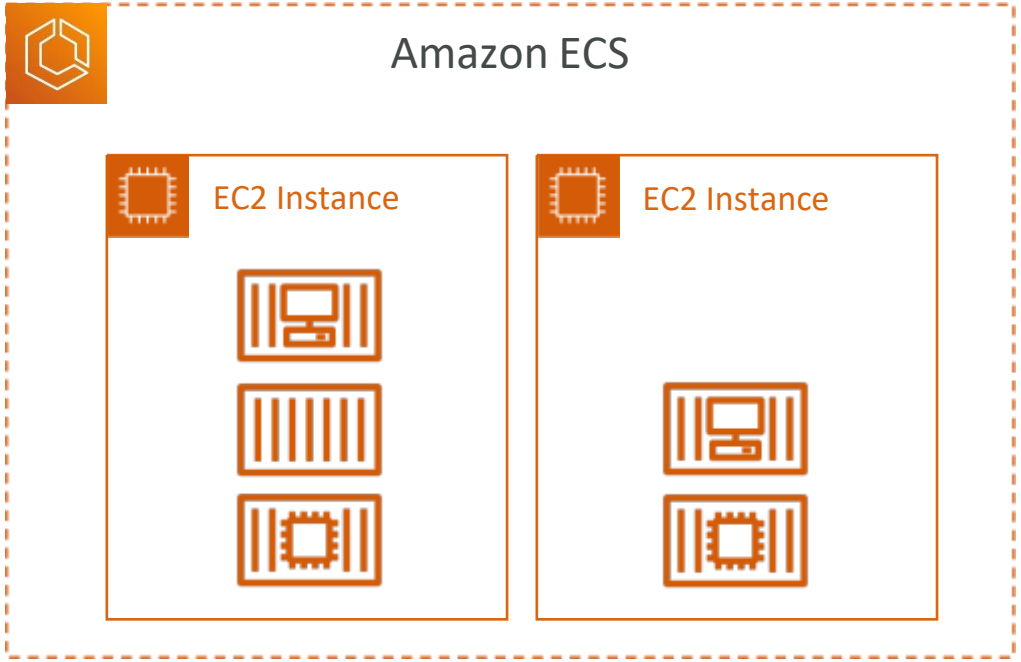
- Task Placement Strategies are a **best effort**

- When Amazon ECS places a task, it uses the following process to select the appropriate EC2 Container instance:
  1. Identify which instances that satisfy the **CPU, memory, and port** requirements
  2. Identify which instances that satisfy the **Task Placement Constraints**
  3. Identify which instances that satisfy the **Task Placement Strategies**
  4. Select the instances

Amazon ECS –Task Placement Strategies

- **Binpack**
  - Tasks are placed on the least available amount of CPU and Memory
  - Minimizes the number of EC2 instances in use (cost savings)

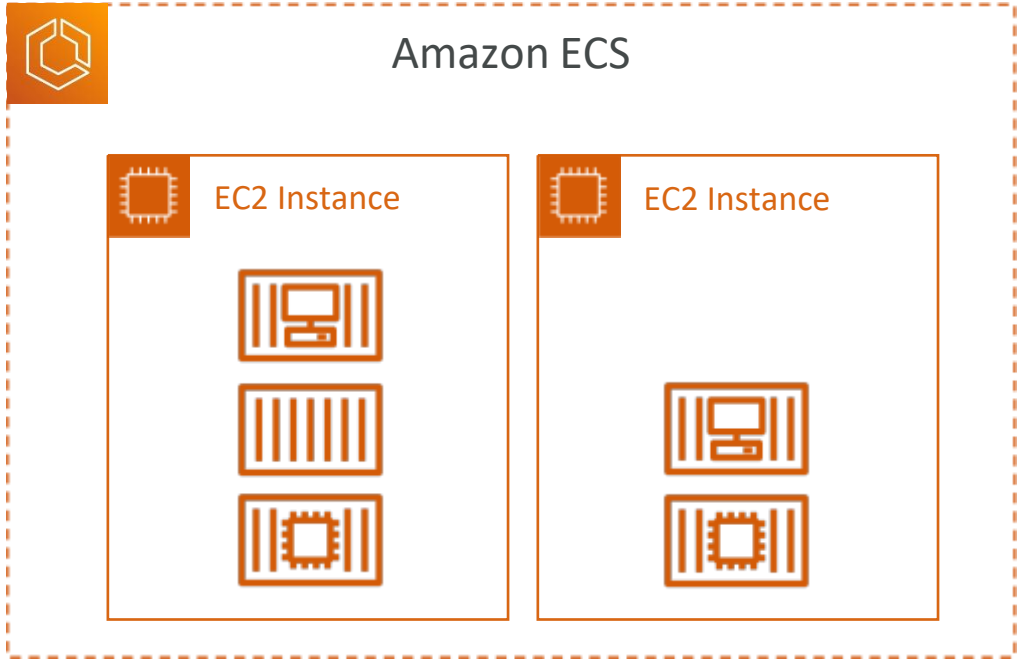
```
"placementStrategy": [
    {
        "type": "binpack",
        "field": "memory"
    }
]
```



Amazon ECS –Task Placement Strategies

- **Random**
  - Tasks are placed randomly

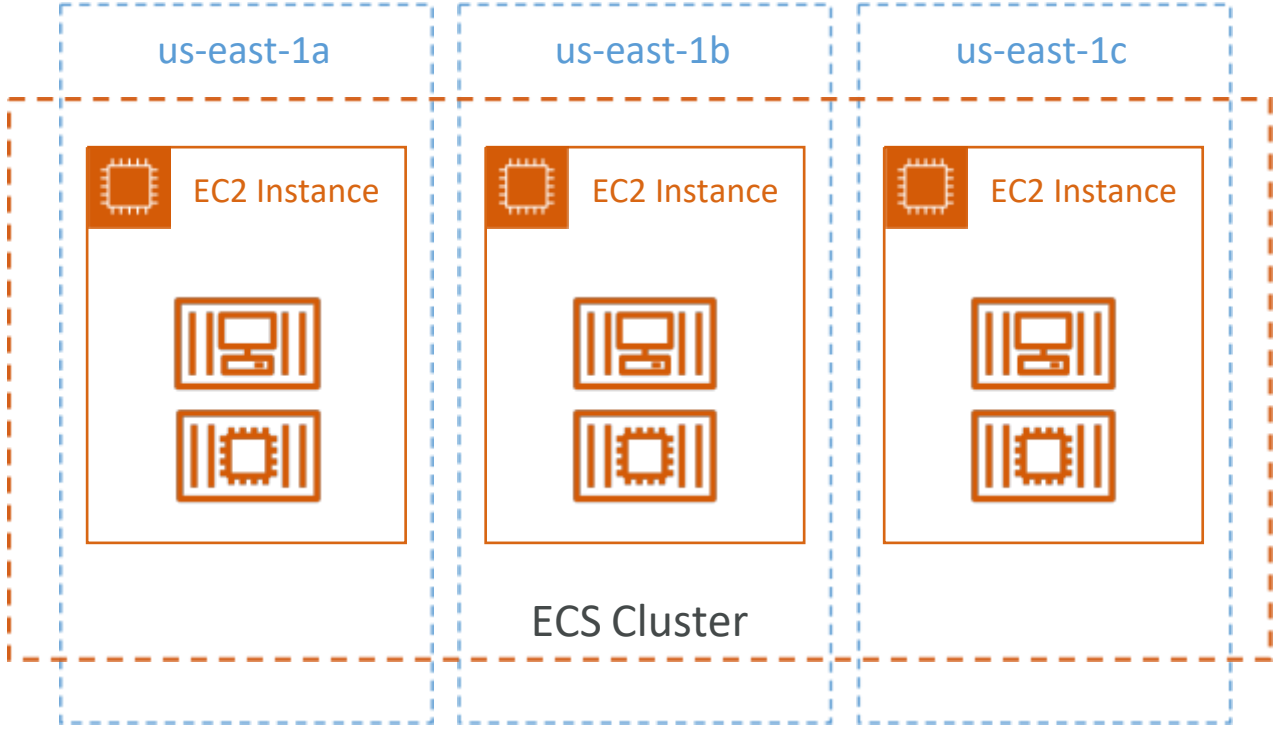
```
"placementStrategy": [
    {
        "type": "random"
    }
]
```



Amazon ECS –Task Placement Strategies

- **Spread**
  - Tasks are placed evenly based on the specified value
  - Example: **instanceId**, **attribute:ecs.availability-zone**, **...**

```
"placementStrategy": [
    {
        "type": "spread",
        "field": "attribute:ecs.availability-zone"
    }
]
```



Amazon ECS –Task Placement Strategies

- You can mix them together

```
"placementStrategy": [
    {
        "type": "spread",
        "field": "attribute:ecs.availability-zone"
    },
    {
        "type": "spread",
        "field": "instanceId"
    }
]
```

```
"placementStrategy": [
    {
        "type": "spread",
        "field": "attribute:ecs.availability-zone"
    },
    {
        "type": "binpack",
        "field": "memory"
    }
]
```

Amazon ECS –Task Placement Constraints

- **distinctInstance**
  - Tasks are placed on a different EC2 instance

```
"placementConstraints": [
    {
        "type": "distinctInstance"
    }
]
```

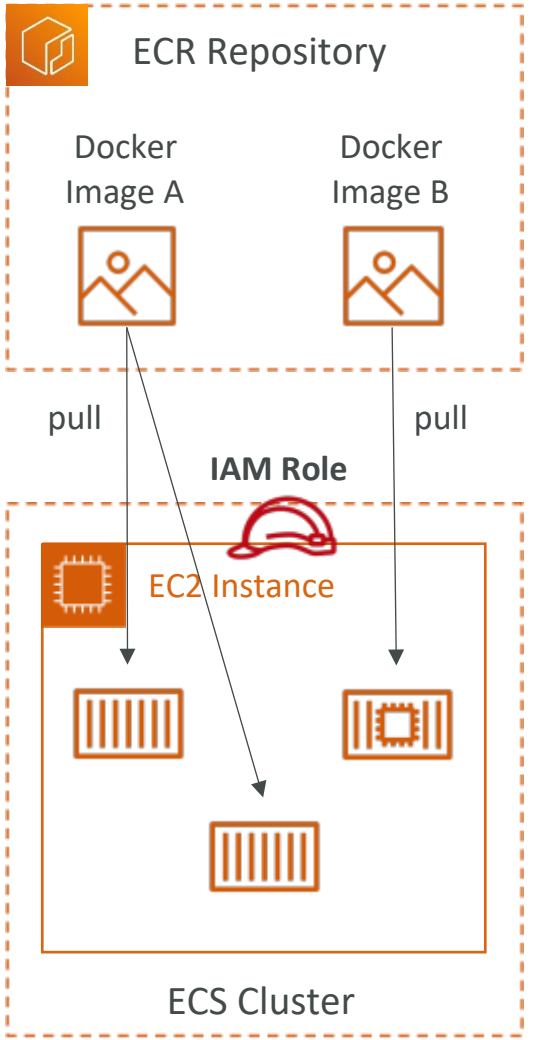
- **memberOf**
  - Tasks are placed on EC2 instances that satisfy a specified expression
  - Uses the **Cluster Query Language** (advanced)

```
"placementConstraints": [
    {
        "type": "memberOf",
        "expression": "attribute:ecs.instance-type =~ t2.*"
    }
]
```

```
"placementConstraints": [
    {
        "type": "memberOf",
        "expression": "attribute:ecs.availability-zone in
 [eu-west-2a, eu-west-2b]"
    }
]
```

Amazon ECR

- ECR = Elastic Container Registry
- Store and manage Docker images on AWS
- **Private** and **Public** repository (**Amazon ECR Public Gallery** https://gallery.ecr.aws)
- Fully integrated with ECS, backed by Amazon S3
- Access is controlled through IAM (permission errors => policy)
- Supports image vulnerability scanning, versioning, image tags, image lifecycle, ...



Amazon ECR – Using AWS CLI

- **Login Command**
  - AWS CLI v2

```
aws ecr get-login-password --region region | docker login --username AWS
      --password-stdin   aws_account_id.dkr.ecr.region.amazonaws.com
```

- **Docker Commands**
  - Push

```
docker  push  aws_account_id.dkr.ecr.region.amazonaws.com/demo:latest
```

  - Pull

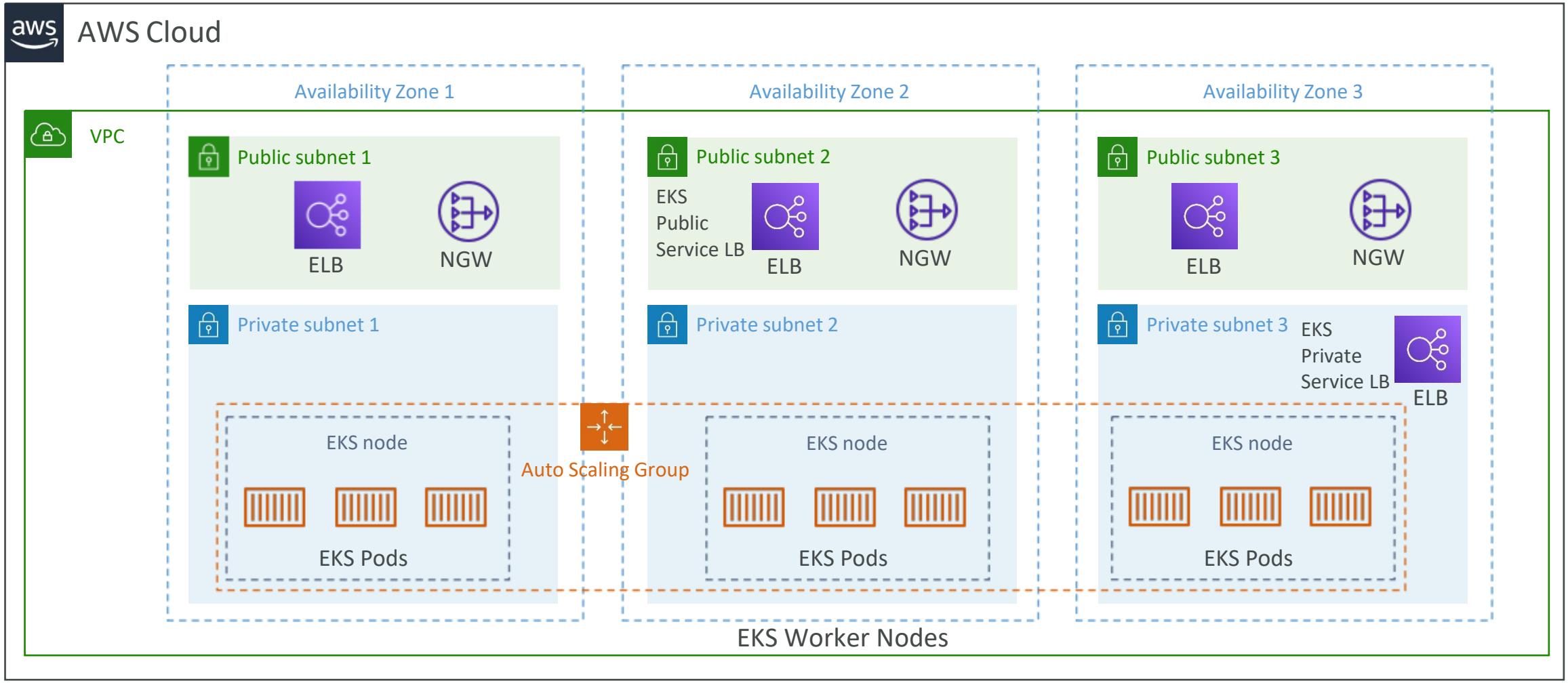
```
docker  pull  aws_account_id.dkr.ecr.region.amazonaws.com/demo:latest
```

  - **In case an EC2 instance (or you) can't pull a Docker image, check IAM permissions**

Amazon EKS Overview

- Amazon EKS = Amazon Elastic **Kubernetes** Service
- It is a way to launch **managed Kubernetes clusters on AWS**
- Kubernetes is an **open-source system** for automatic deployment, scaling and management of containerized (usually Docker) application
- It's an alternative to ECS, similar goal but different API
- EKS supports **EC2** if you want to deploy worker nodes or **Fargate** to deploy serverless containers
- **Use case:** if your company is already using Kubernetes on-premises or in another cloud, and wants to migrate to AWS using Kubernetes
- **Kubernetes is cloud-agnostic** (can be used in any cloud – Azure, GCP…)
- For multiple regions, deploy one EKS cluster per region
- Collect logs and metrics using **CloudWatch Container Insights**

Amazon EKS - Diagram



Amazon EKS – Node Types

- **Managed Node Groups**
  - Creates and manages Nodes (EC2 instances) for you
  - Nodes are part of an ASG managed by EKS
  - Supports On-Demand or Spot Instances

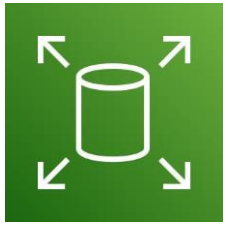
- **Self-Managed Nodes**
  - Nodes created by you and registered to the EKS cluster and managed by an ASG
  - You can use prebuilt AMI - Amazon EKS Optimized AMI
  - Supports On-Demand or Spot Instances

- **AWS Fargate**
  - No maintenance required; no nodes managed

Amazon EKS – Data Volumes

- Need to specify **StorageClass** manifest on your EKS cluster
- Leverages a **Container Storage Interface (CSI)** compliant driver

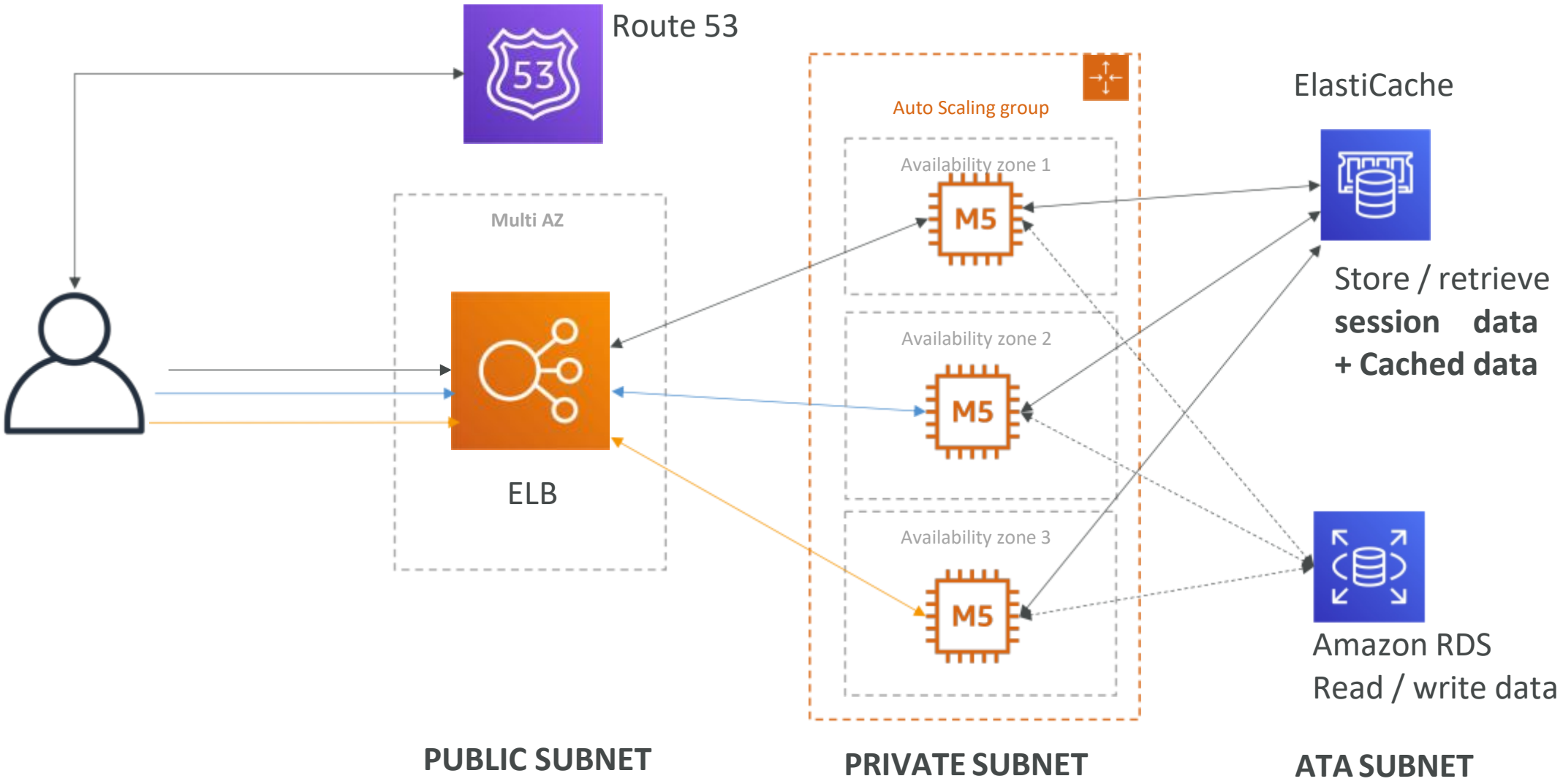
- Support for…
- Amazon EBS
- Amazon EFS (works with Fargate)
- Amazon FSx for Lustre
- Amazon FSx for NetApp ONTAP



AWS Elastic Beanstalk

Deploying applications in AWS safely and predictably

Typical architecture: Web App 3-tier



Developer problems on AWS

- Managing infrastructure
- Deploying Code
- Configuring all the databases, load balancers, etc
- Scaling concerns

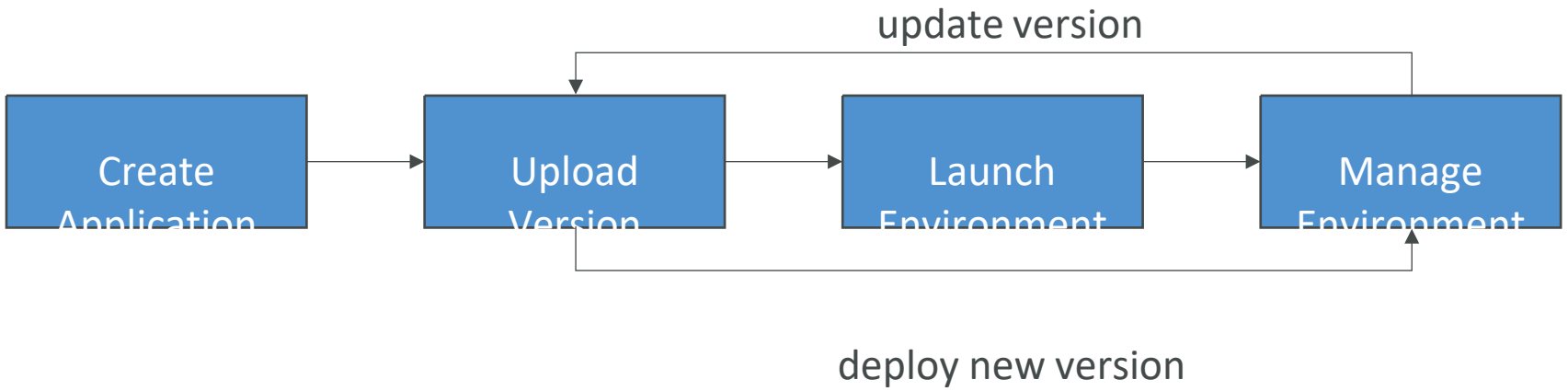
- Most web apps have the same architecture (ALB + ASG)
- All the developers want is for their code to run!
- Possibly, consistently across different applications and environments

Elastic Beanstalk – Overview

- Elastic Beanstalk is a developer centric view of deploying an application on AWS
- It uses all the component's we've seen before: EC2, ASG, ELB, RDS, …
- Managed service
  - Automatically handles capacity provisioning, load balancing, scaling, application health monitoring, instance configuration, …
  - Just the application code is the responsibility of the developer
- We still have full control over the configuration
- Beanstalk is free but you pay for the underlying instances

Elastic Beanstalk – Components

- **Application:** collection of Elastic Beanstalk components (environments, versions, configurations, …)
- **Application Version:** an iteration of your application code
- **Environment**
  - Collection of AWS resources running an application version (only one application version at a time)
  - **Tiers:** Web Server Environment Tier & Worker Environment Tier
  - You can create multiple environments (dev, test, prod, …)

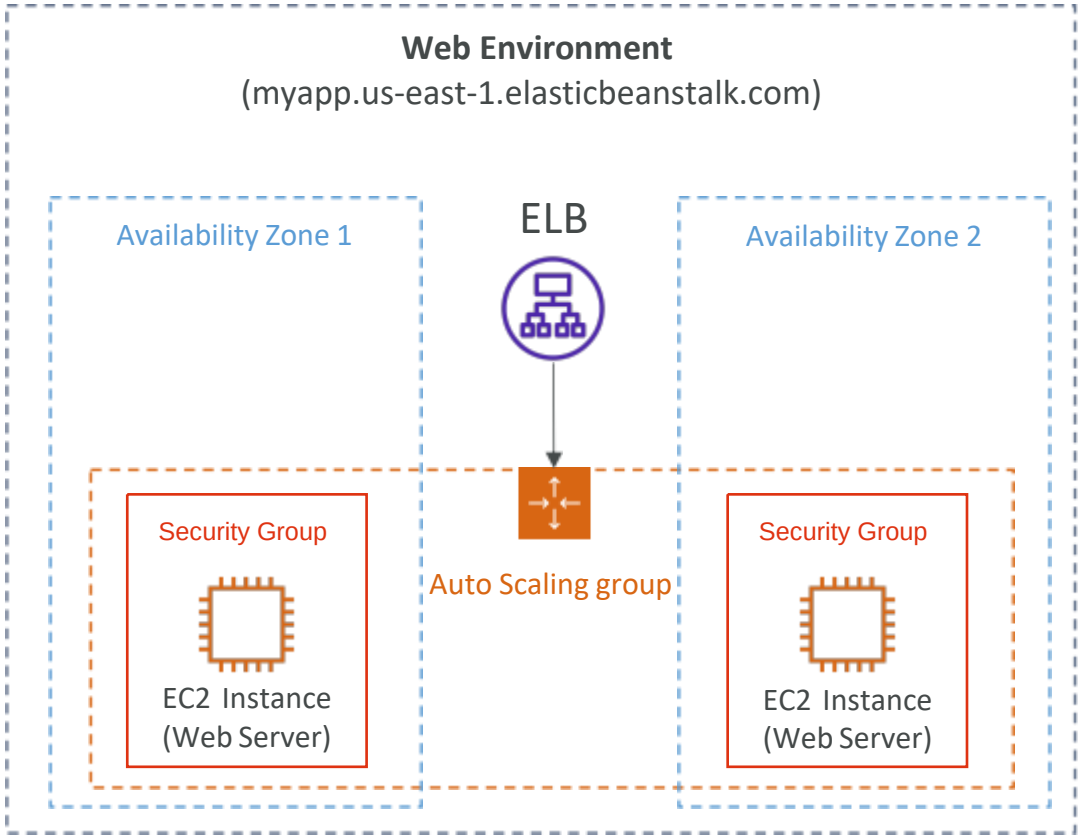


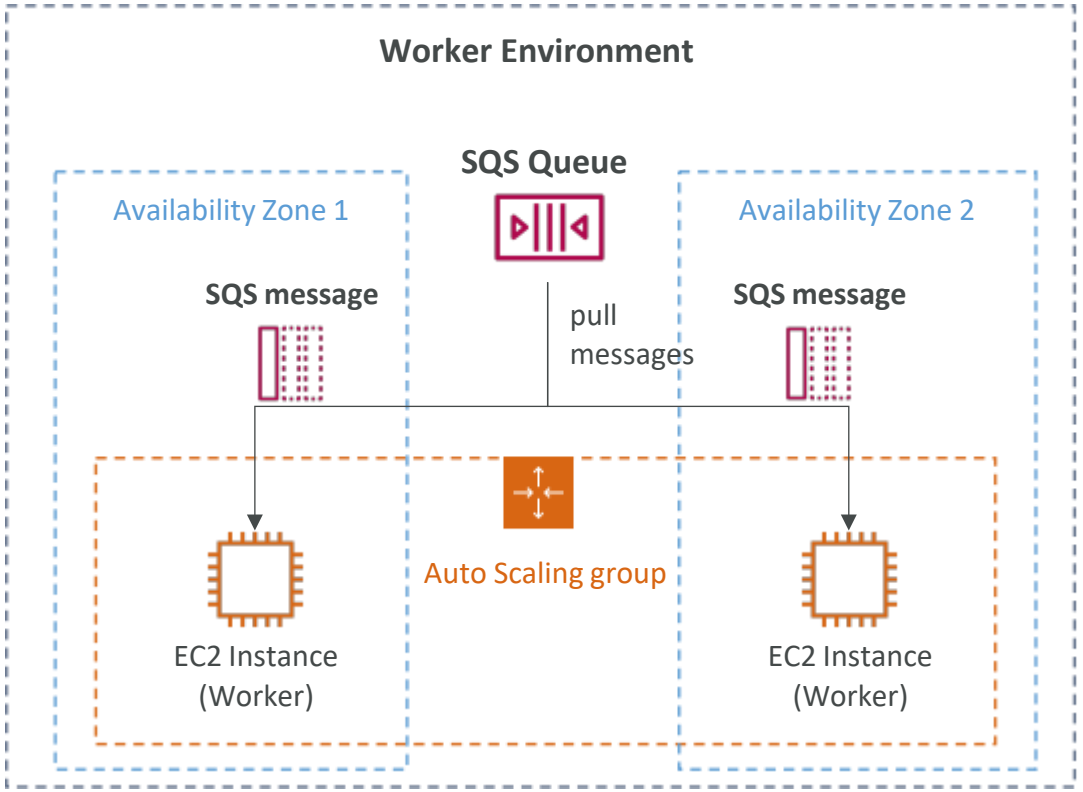
Elastic Beanstalk – Supported Platforms

- Go
- Java SE
- Java with Tomcat
- .NET Core on Linux
- .NET on Windows Server
- Node.js
- PHP
- Python
- Ruby
- Packer Builder
- Single Container Docker
- Multi-container Docker
- Preconfigured Docker

- If not supported, you can write your custom platform (advanced)

Web Server Tier vs. Worker Tier

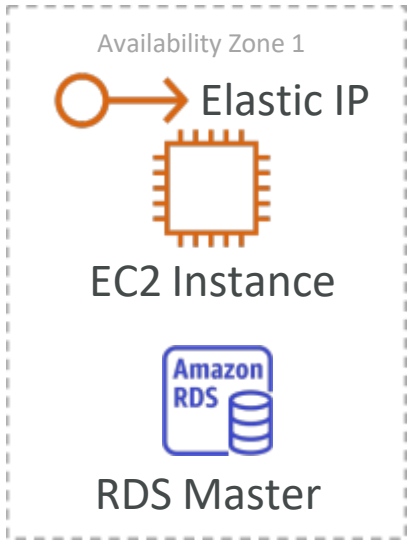




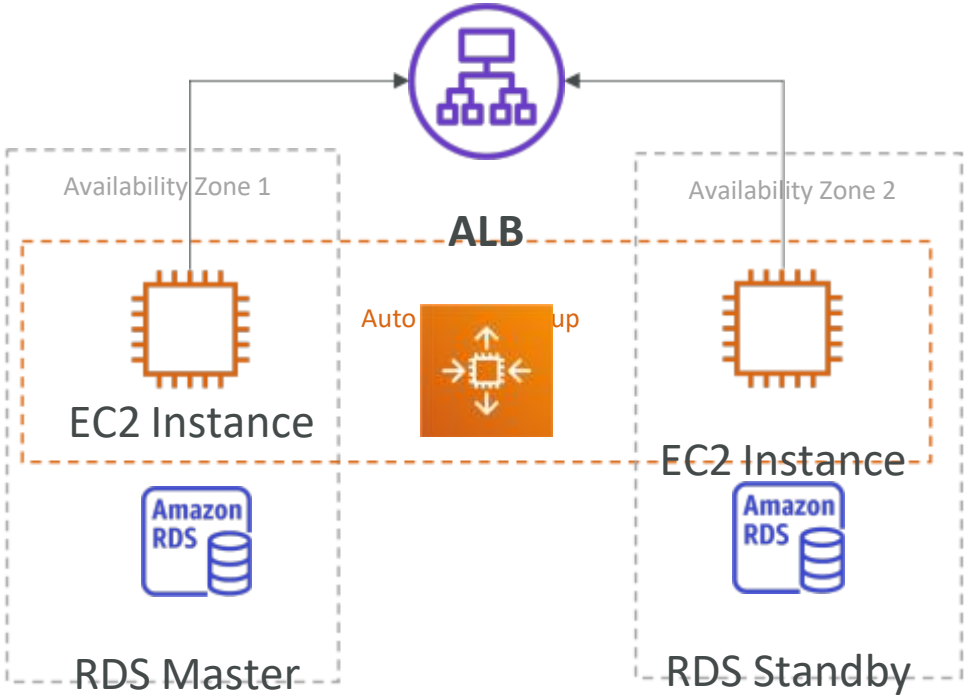
- Scale based on the number of SQS messages
- Can push messages to SQS queue from another Web Server Tier

Elastic Beanstalk Deployment Modes

Single Instance
Great for dev



High Availability with Load Balancer
Great for prod



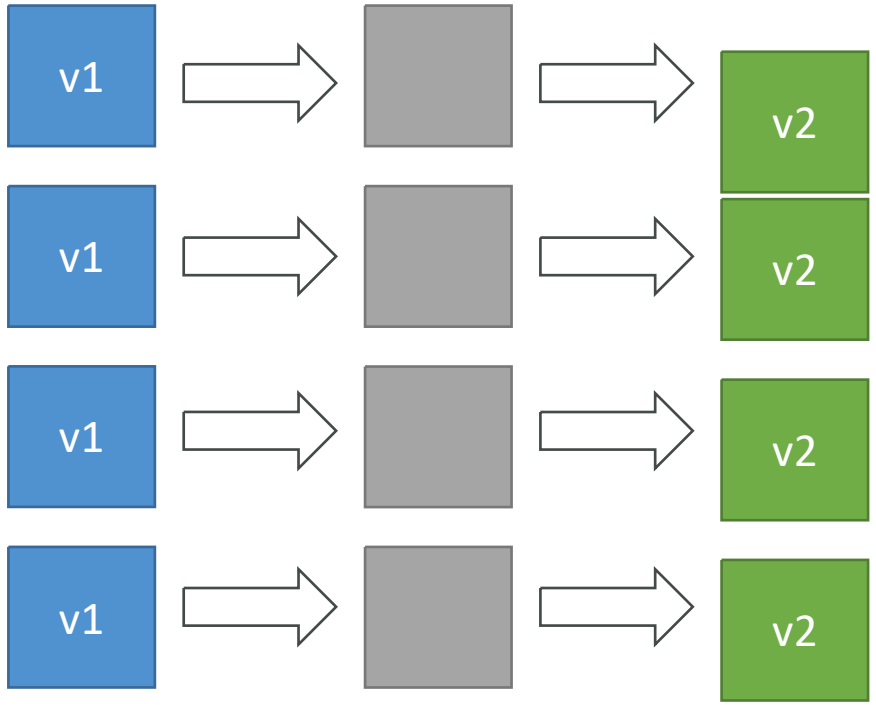
Beanstalk Deployment Options for Updates

- **All at once (deploy all in one go)** – fastest, but instances aren't available to serve traffic for a bit (downtime)
- **Rolling:** update a few instances at a time (bucket), and then move onto the next bucket once the first bucket is healthy
- **Rolling with additional batches:** like rolling, but spins up new instances to move the batch (so that the old application is still available)
- **Immutable:** spins up new instances in a new ASG, deploys version to these instances, and then swaps all the instances when everything is healthy

Elastic Beanstalk Deployment

All at once

- Fastest deployment
- Application has downtime
- Great for quick iterations in development environment
- No additional cost



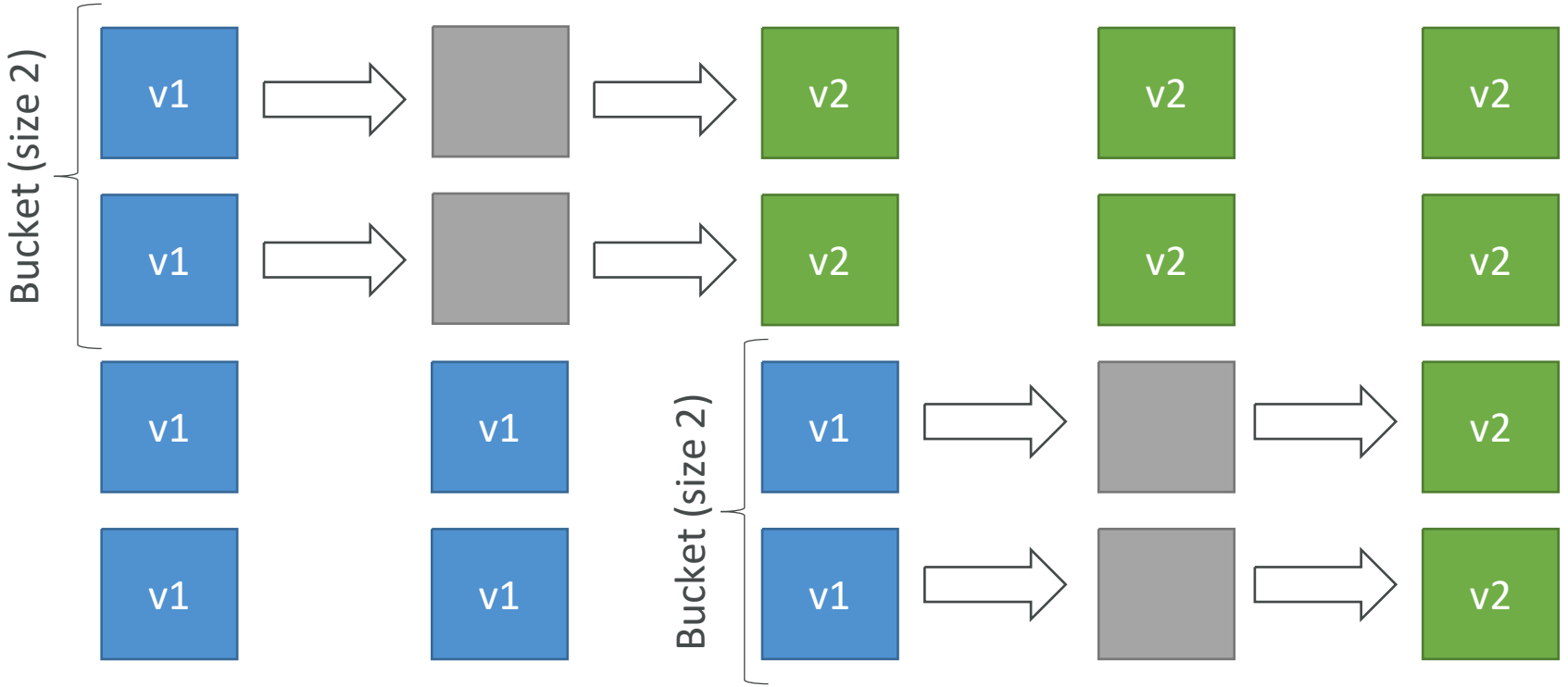
Elastic Beanstalk Deployment

Rolling

- Application is running below capacity
- Can set the bucket size

Application is running both versions simultaneously

- No additional cost
- Long



deployment

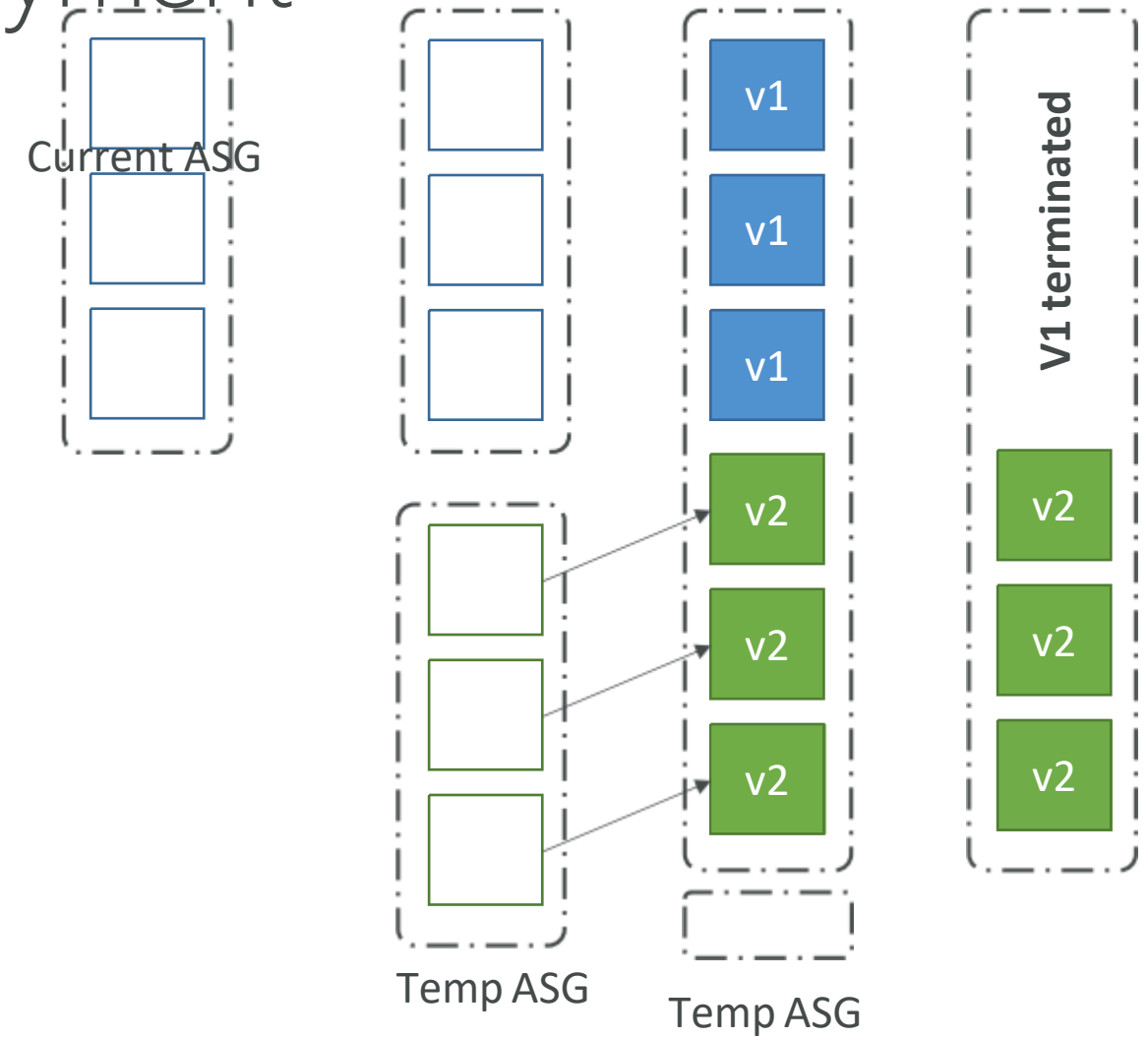
Elastic Beanstalk Deployment

Rolling with additional batches

- Application is running at capacity
- Application is running both versions simultaneously
- Small additional cost
- Additional batch is removed at the end of the deployment
- Longer deployment
- Good for prod

Immutable Elastic Beanstalk Deployment

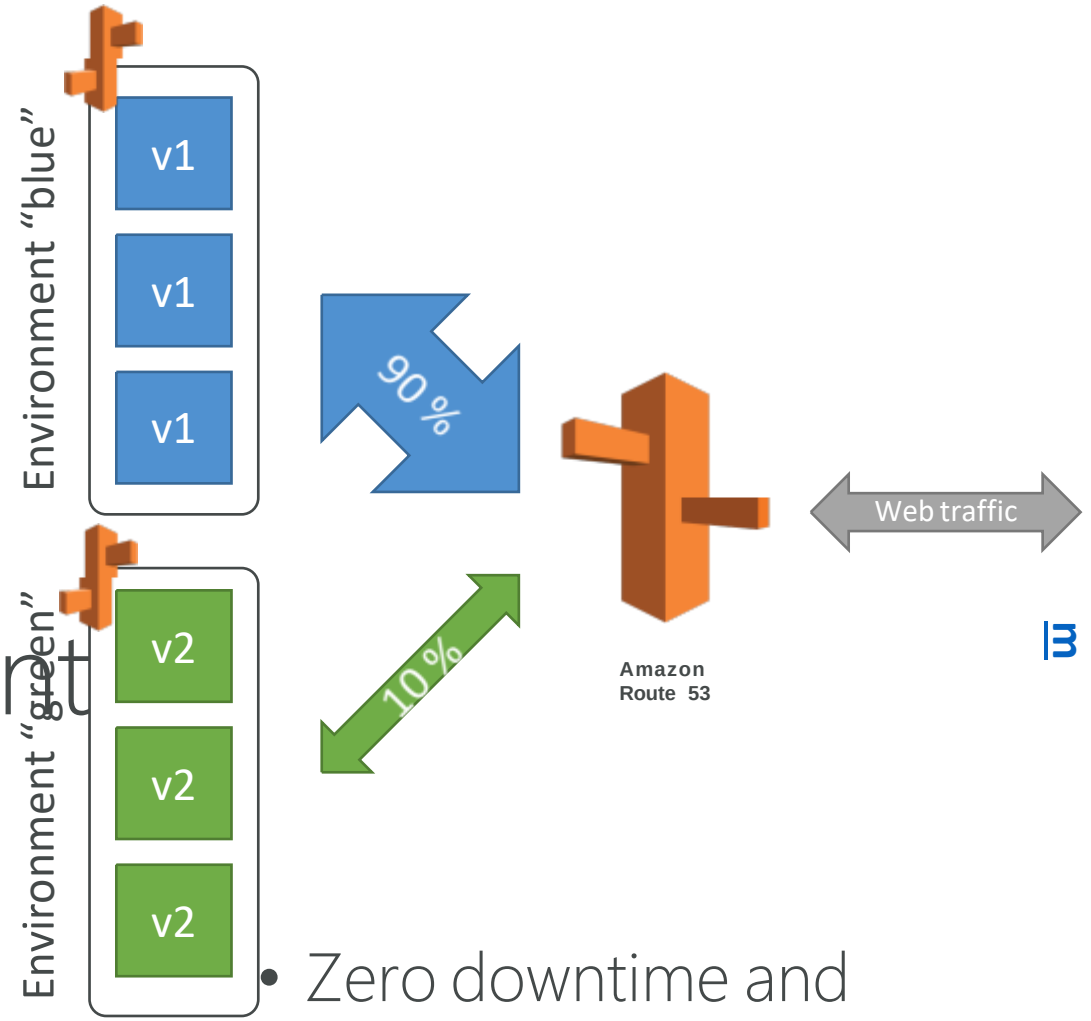
- Zero downtime
- New Code is deployed to new instances on a temporary ASG
- High cost, double capacity
- Longest deployment
- Quick rollback in case of failures (just terminate new ASG)
- Great for prod



Current ASG Current ASG Current ASG

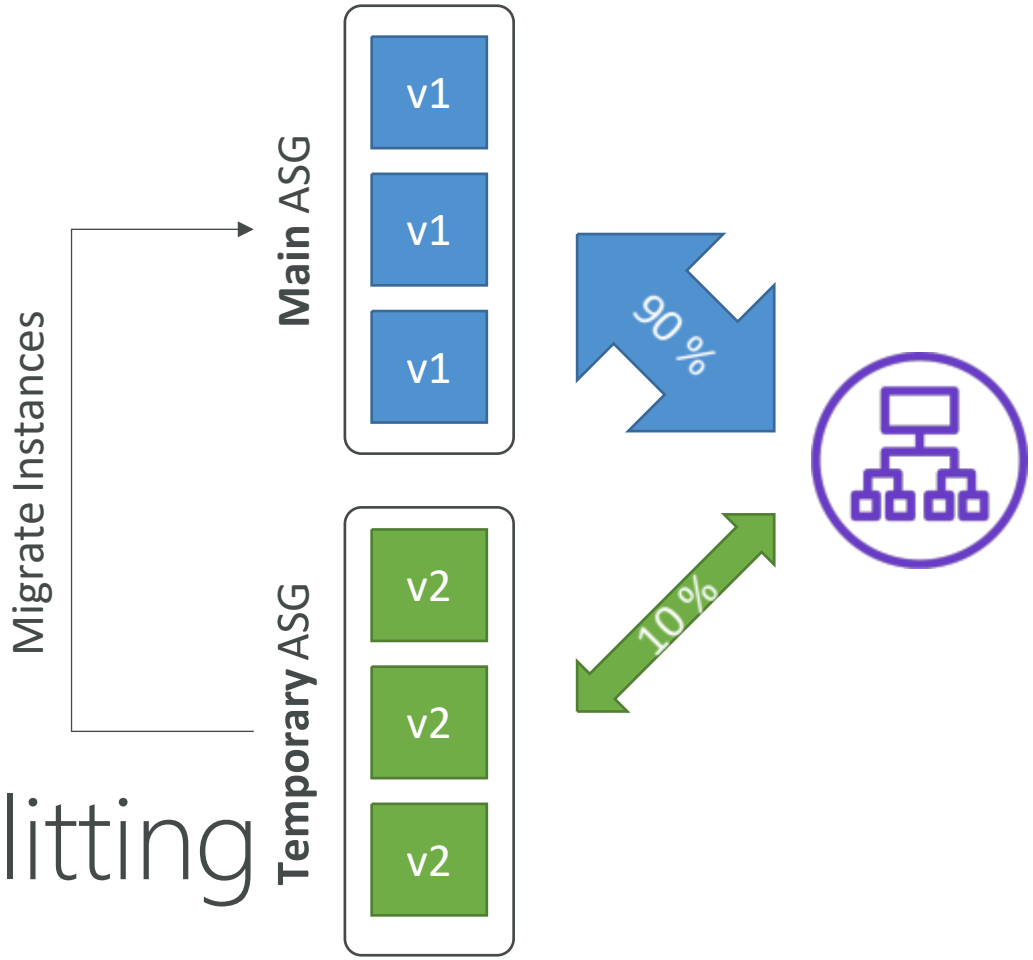
Elastic Beanstalk Deployment Blue / Green

- ot a “direct feature” of Elastic Beanstalk



- Zero downtime and release facility

- Create a new "stage" environment and deploy v2 there
- The new environment (green) can be validated independently and roll back if issues
- Route 53 can be setup using weighted policies to redirect a little bit of traffic to the stage environment
- A small % of traffic is sent to the temporary ASG for a configurable amount of time
- Using Beanstalk, "swap URLs" when done with the environment test



Elastic Beanstalk -Traffic Splitting

- **Canary Testing**
  - New application

version is deployed to a temporary ASG with the same capacity

- Deployment health is monitored
- If there's a deployment failure, this triggers an **automated rollback (very quick)**
- No application downtime
- New instances are migrated from the temporary to the original ASG
- Old application version is then terminated

Elastic Beanstalk Deployment Summary

from AWS Doc

- https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.deploy-existing-version.html

Deployment methods

Method	Impact of failed deployment	Deploy time	Zero downtime	No DNS change	Rollback process	Code deployed to
All at once	Downtime	🕒	X	✓	Manual redeploy	Existing instances
Rolling	Single batch out of service; any successful batches before failure running new application version	🕒🕒†	✓	✓	Manual redeploy	Existing instances
Rolling with an additional batch	Minimal if first batch fails; otherwise, similar to **Rolling**	🕒🕒🕒†	✓	✓	Manual redeploy	New and existing instances
Immutable	Minimal	🕒🕒🕒🕒	✓	✓	Terminate new instances	New instances
Traffic splitting	Percentage of client traffic routed to new version temporarily impacted	🕒🕒🕒🕒††	✓	✓	Reroute traffic and terminate new instances	New instances
Blue/green	Minimal	🕒🕒🕒🕒	✓	X	Swap URL	New instances

Elastic Beanstalk CLI

- We can install an additional CLI called the “EB cli” which makes working with Beanstalk from the CLI easier
- Basic commands are:
  - eb create
  - eb status
  - eb health
  - eb events
  - eb logs
  - eb open
  - eb deploy
  - eb config
  - eb terminate
- It’s helpful for your automated deployment pipelines!

Elastic Beanstalk Deployment Process

- Describe dependencies
  (requirements.txt for Python, package.json for Node.js)
- Package code as zip, and describe dependencies
  - Python: requirements.txt
  - Node.js: package.json
- **Console:** upload zip file (creates new app version), and then deploy
- **CLI:** create new app version using CLI (uploads zip), and then deploy

- Elastic Beanstalk will deploy the zip on each EC2 instance, resolve dependencies and start the application

Beanstalk Lifecycle Policy

- Elastic Beanstalk can store at most 1000 application versions
- If you don’t remove old versions, you won’t be able to deploy anymore
- To phase out old application versions, use a **lifecycle policy**
  - Based on time (old versions are removed)
  - Based on space (when you have too many versions)
- Versions that are currently used won’t be deleted
- Option not to delete the source bundle in S3 to prevent data loss

Elastic Beanstalk Extensions

- A zip file containing our code must be deployed to Elastic Beanstalk
- All the parameters set in the UI can be configured with code using files
- Requirements:
  - in the .ebextensions/ directory in the root of source code
  - YAML / JSON format
  - **.config** extensions (example: logging.config)
  - Able to modify some default settings using: option_settings
  - Ability to add resources such as RDS, ElastiCache, DynamoDB, etc…

- Resources managed by .ebextensions get deleted if the environment goes away

Elastic Beanstalk Under the Hood

- Under the hood, Elastic Beanstalk relies on CloudFormation
- CloudFormation is used to provision other AWS services (we'll see later)

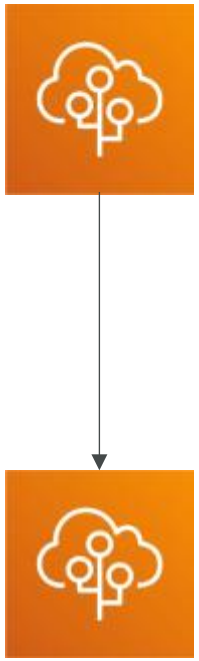
Elastic Beanstalk

CloudFormation

- Use case: you can define CloudFormation resources in your **.ebextensions** to provision ElastiCache, an S3 bucket, anything you want!
- Let's have a sneak peak into it!

Elastic Beanstalk Cloning

- Clone an environment with the exact same configuration
- Useful for deploying a “test” version of your application

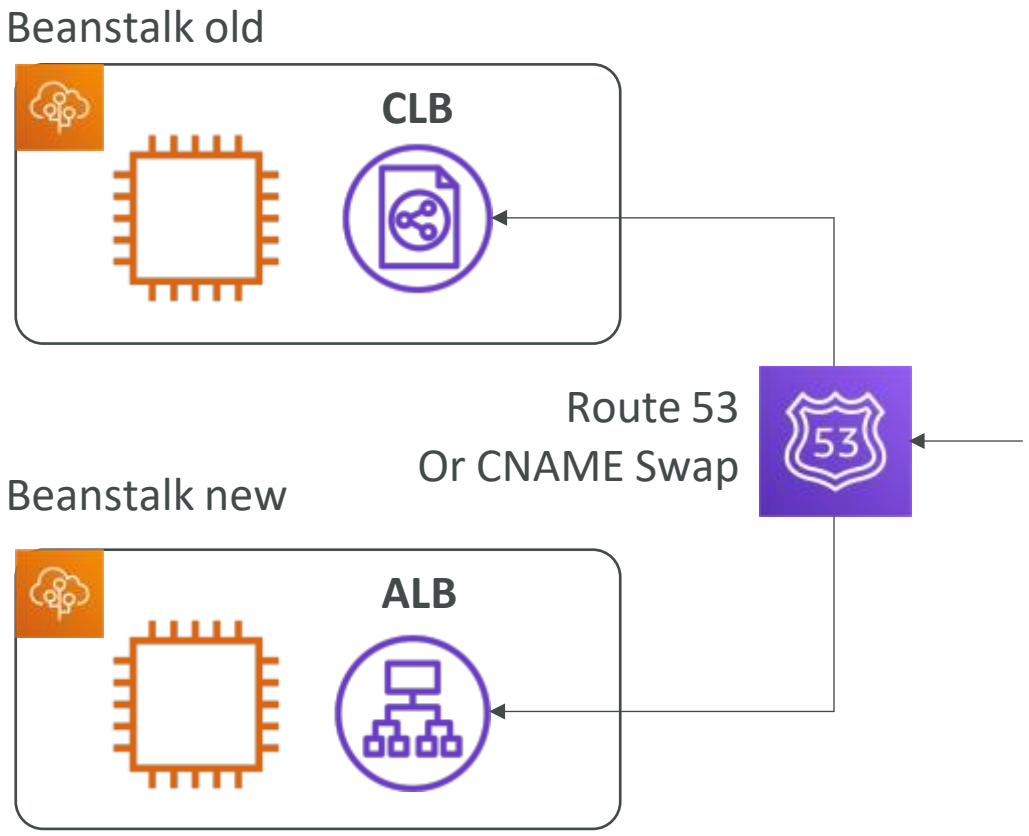


- All resources and configuration are preserved:
  - Load Balancer type and configuration
  - RDS database type (but the data is not preserved)
  - Environment variables

- After cloning an environment, you can change settings

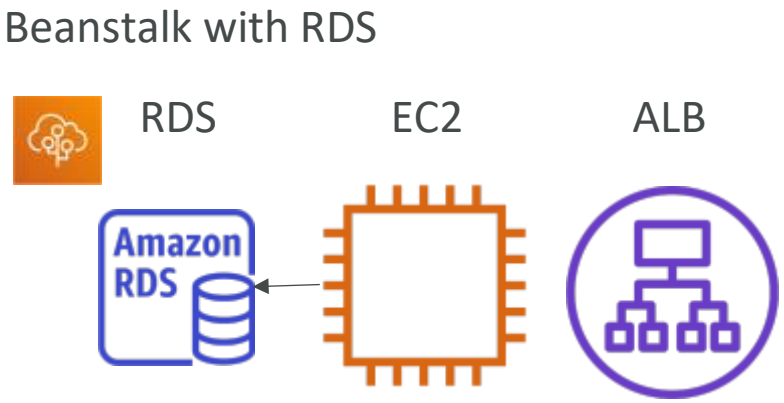
Elastic Beanstalk Migration: Load Balancer

- After creating an Elastic Beanstalk environment, **you cannot change the Elastic Load Balancer type** (only the configuration)
- To migrate:
  1. create a new environment with the same configuration except LB (can't clone)
  2. deploy your application onto the new environment
  3. perform a CNAME swap or Route 53 update



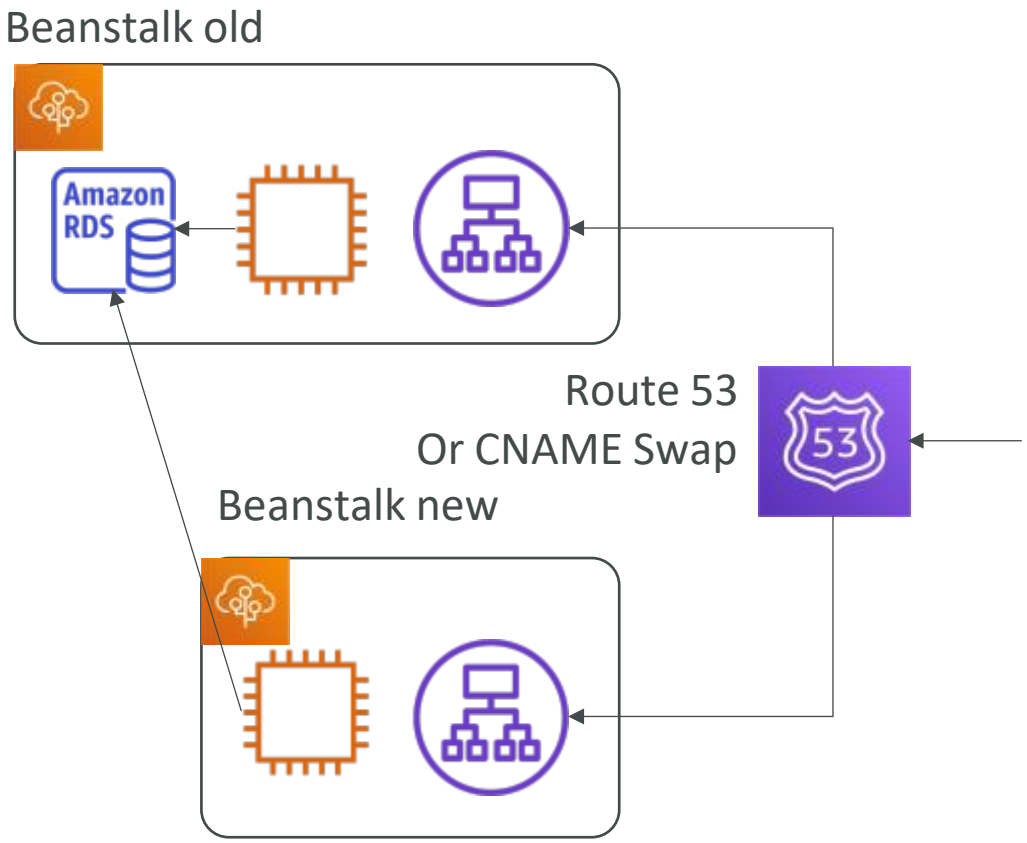
RDS with Elastic Beanstalk

- RDS can be provisioned with Beanstalk, which is great for dev / test
- This is not great for prod as the database lifecycle is tied to the Beanstalk environment lifecycle
- The best for prod is to separately create an RDS database and provide our EB application with the connection string



Elastic Beanstalk Migration: Decouple RDS

1. Create a snapshot of RDS DB (as a safeguard)
2. Go to the RDS console and protect the RDS database from deletion
3. Create a new Elastic Beanstalk environment, without RDS, point your application to existing RDS
3. perform a CNAME swap (blue/green) or Route 53 update, confirm working
4. Terminate the old environment (RDS won't be deleted)
5. Delete CloudFormation stack (in DELETE_FAILED state)



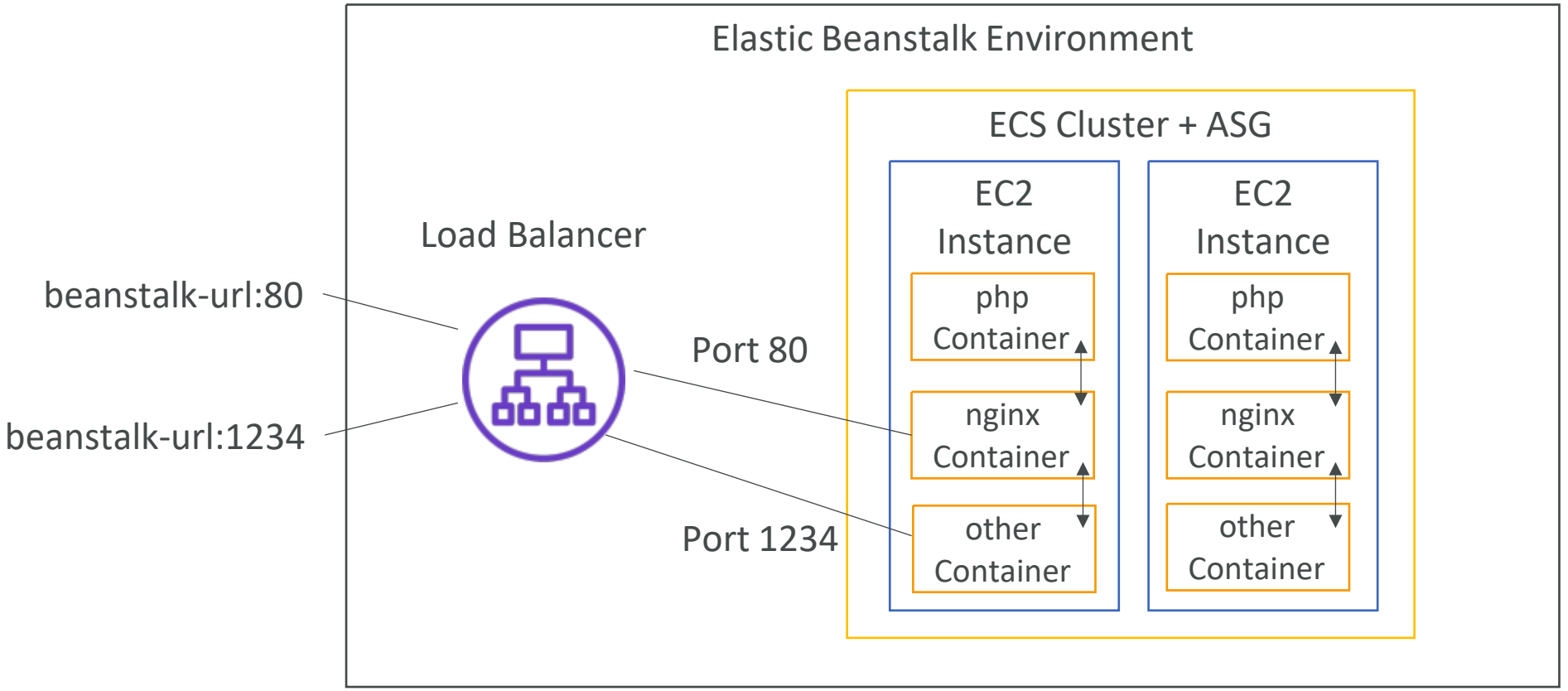
Elastic Beanstalk – Single Docker

- Run your application as a single docker container
- Either provide:
  - **Dockerfile**: Elastic Beanstalk will build and run the Docker container
  - **Dockerrun.aws.json (v1):** Describe where *already built* Docker image is
    - Image
    - Ports
    - Volumes
    - Logging
    - Etc…
- Beanstalk in Single Docker Container does not use ECS

Elastic Beanstalk – Multi Docker Container

- Multi Docker helps run multiple containers per EC2 instance in EB
- This will create for you:
  - ECS Cluster
  - EC2 instances, configured to use the ECS Cluster
  - Load Balancer (in high availability mode)
  - Task definitions and execution
- Requires a config **Dockerrun.aws.json (v2)** at the root of source code
- **Dockerrun.aws.json** is used to generate the **ECS task definition**
- Your Docker images must be pre-built and stored in ECR for example

Elastic Beanstalk + Multi Docker ECS



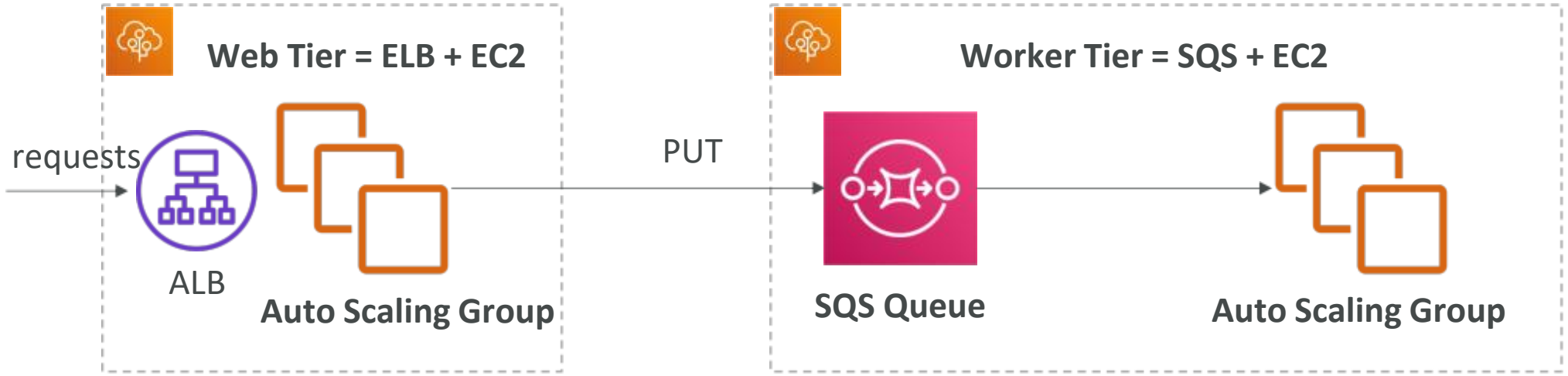
Elastic Beanstalk and HTTPS

- **Beanstalk with HTTPS**
  - Idea: Load the SSL certificate onto the Load Balancer
  - Can be done from the Console (EB console, load balancer configuration)
  - Can be done from the code: .ebextensions/securelistener-alb.config
  - SSL Certificate can be provisioned using ACM (AWS Certificate Manager) or CLI
  - Must configure a security group rule to allow incoming port 443 (HTTPS port)

- **Beanstalk redirect HTTP to HTTPS**
  - Configure your instances to redirect HTTP to HTTPS: https://github.com/awsdocs/elastic-beanstalk-samples/tree/master/configuration-files/aws-provided/security-configuration/https-redirect
  - OR configure the Application Load Balancer (ALB only) with a rule
  - Make sure health checks are not redirected (so they keep giving 200 OK)

Web Server vs Worker Environment

- If your application performs tasks that are long to complete, offload these tasks to a dedicated **worker environment**
- **Decoupling** your application into two tiers is common
- Example: processing a video, generating a zip file, etc
- You can define periodic tasks in a file **cron.yaml**



Elastic Beanstalk – Custom Platform (Advanced)

- Custom Platforms are very advanced, they allow to define from scratch:
  - The Operating System (OS)
  - Additional Software
  - Scripts that Beanstalk runs on these platforms
- Use case: app language is incompatible with Beanstalk & doesn't use Docker
- To create your own platform:
  - Define an AMI using **Platform.yaml** file
  - Build that platform using the **Packer software (open source tool to create AMIs)**
- Custom Platform vs Custom Image (AMI):
  - Custom Image is to tweak an existing Beanstalk Platform (Python, Node.js, Java…)
  - Custom Platform is to create an entirely new Beanstalk Platform

AWS CICD

CodeCommit, CodePipeline, CodeBuild, CodeDeploy, ...

CICD – Introduction

- We have learned how to:
  - Create AWS resources, manually (fundamentals)
  - Interact with AWS programmatically (AWS CLI)
  - Deploy code to AWS using Elastic Beanstalk
- All these manual steps make it very likely for us to do mistakes!

- We would like our code "in a repository" and have it deployed onto AWS
  - Automatically
  - The right way
  - Making sure it's tested before being deployed
  - With possibility to go into different stages (dev, test, staging, prod)
  - With manual approval where needed
- To be a proper AWS developer… we need to learn AWS CICD

CICD – Introduction

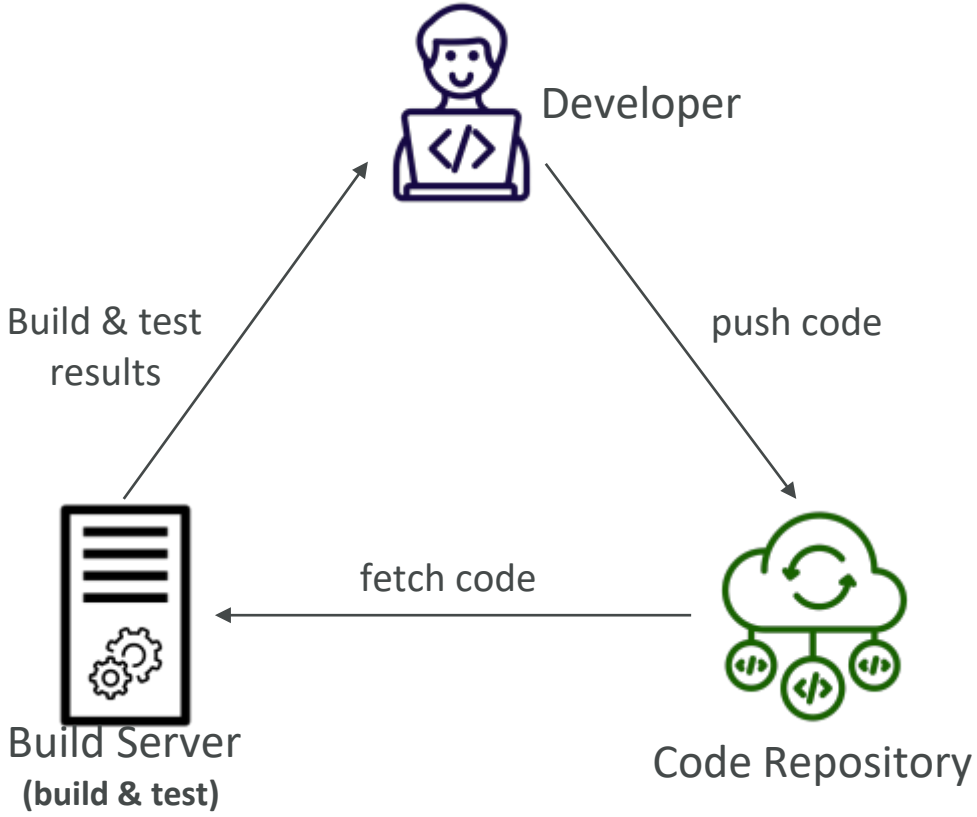
- This section is all about automating the deployment we've done so far while adding increased safety

- We'll learn about:
  - **AWS CodeCommit** – storing our code
  - **AWS CodePipeline** – automating our pipeline from code to Elastic Beanstalk
  - **AWS CodeBuild** – building and testing our code
  - **AWS CodeDeploy** – deploying the code to EC2 instances (not Elastic Beanstalk)
  - **AWS CodeStar** – manage software development activities in one place
  - **AWS CodeArtifact** – store, publish, and share software packages
  - **AWS CodeGuru** – automated code reviews using Machine Learning

Continuous Integration (CI)

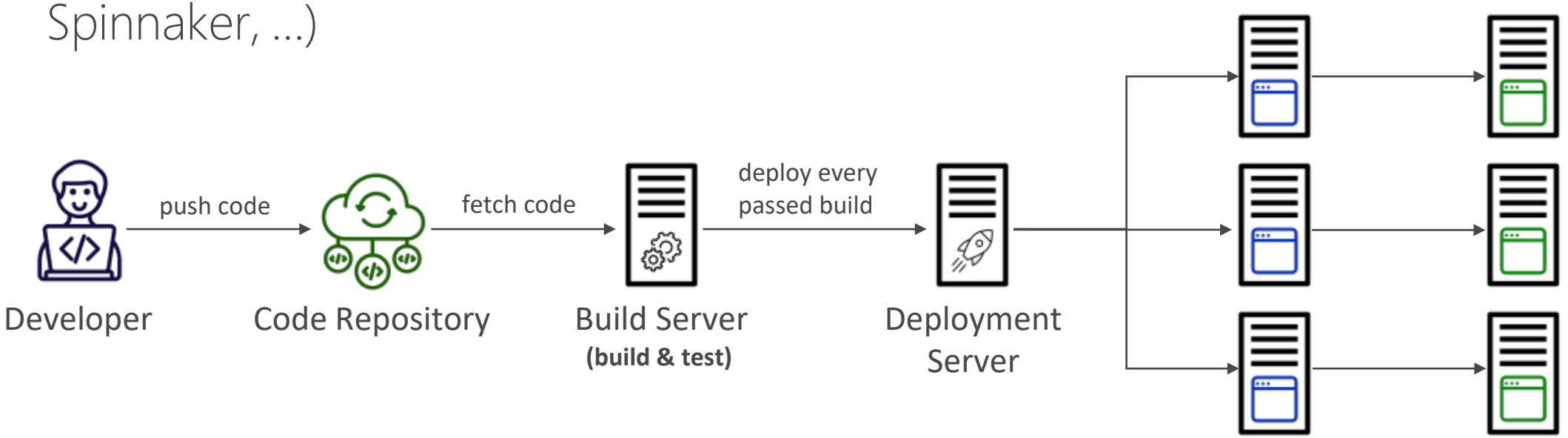
- Developers push the code to a code repository often (e.g., GitHub, CodeCommit, Bitbucket…)
- A testing / build server checks the code as soon as it's pushed (CodeBuild, Jenkins CI, …)
- The developer gets feedback about the tests and checks that have passed / failed

- Find bugs early, then fix bugs
- Deliver faster as the code is tested
- Deploy often
- Happier developers, as they're unblocked



Continuous Delivery (CD)

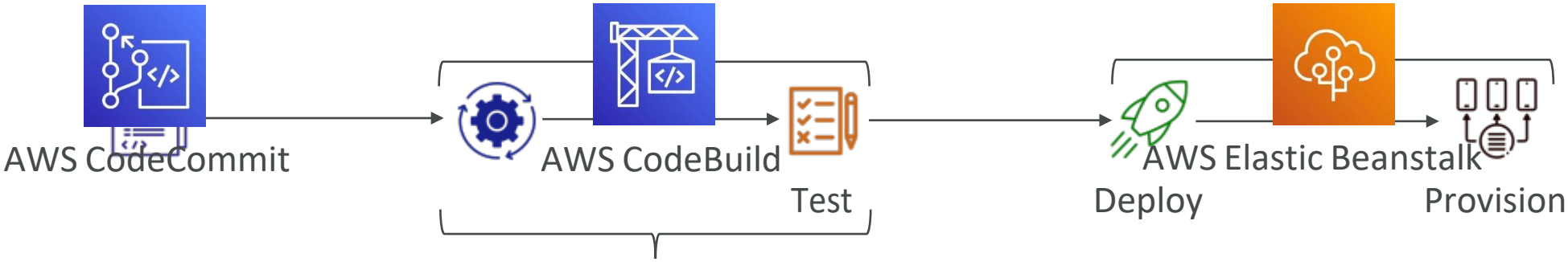
- Ensures that the software can be released reliably whenever needed
- Ensures deployments happen often and are quick
- Shift away from "one release every 3 months" to "5 releases a day"
- That usually means automated deployment (e.g., CodeDeploy, Jenkins CD, Spinnaker, …)



Application Server **v1**

Application Server **v2**

Technology Stack for CICD



Code

Build

GitHub

Bitbucket

any 3rd party Code Repository

Jenkins CI

any 3rd party CI Servers

AWS CodeDeploy

EC2 Instances

On-premises Instances

AWS Lambda

Amazon ECS

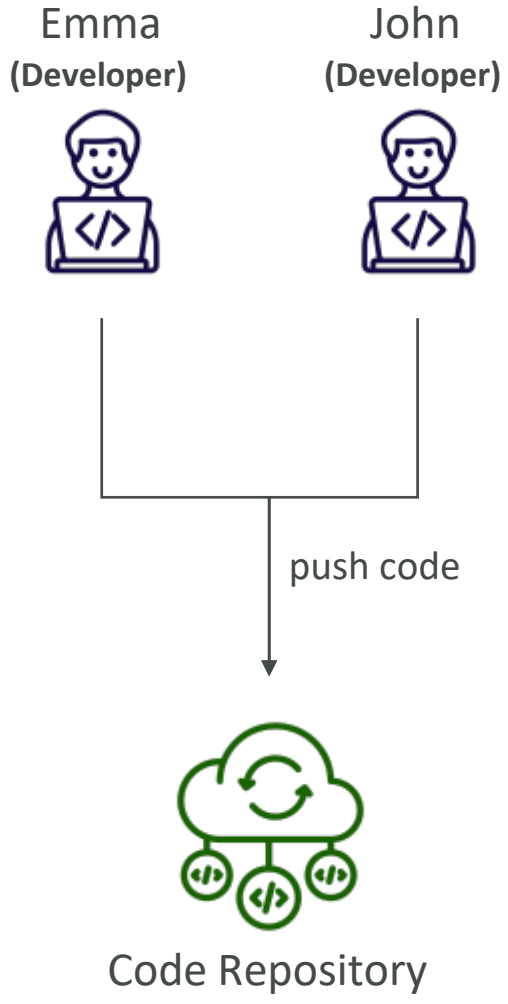
orchestrate using:

AWS CodeCommit

- **Version control** is the ability to understand the various changes that happened to the code over time (and possibly roll back)
- All these are enabled by using a version control system such as **Git**
- A Git repository can be synchronized on your computer, but it usually is uploaded on a central online repository
- Benefits are:
  - Collaborate with other developers
  - Make sure the code is backed-up somewhere
  - Make sure it's fully viewable and auditable

AWS CodeCommit

- Git repositories can be expensive
- The industry includes GitHub, GitLab, Bitbucket, …
- And **AWS CodeCommit:**
  - Private Git repositories
  - No size limit on repositories (scale seamlessly)
  - Fully managed, highly available
  - Code only in AWS Cloud account => increased security and compliance
  - Security (encrypted, access control, …)
  - Integrated with Jenkins, AWS CodeBuild, and other CI tools



CodeCommit – Security

- Interactions are done using Git (standard)
- **Authentication**
  - SSH Keys – AWS Users can configure SSH keys in their IAM Console
  - HTTPS – with AWS CLI Credential helper or Git Credentials for IAM user
- **Authorization**
  - IAM policies to manage users/roles permissions to repositories
- **Encryption**
  - Repositories are automatically encrypted at rest using AWS KMS
  - Encrypted in transit (can only use HTTPS or SSH – both secure)
- **Cross-account Access**
  - Do NOT share your SSH keys or your AWS credentials
  - Use an IAM Role in your AWS account and use AWS STS (**AssumeRole** API)

CodeCommit vs. GitHub

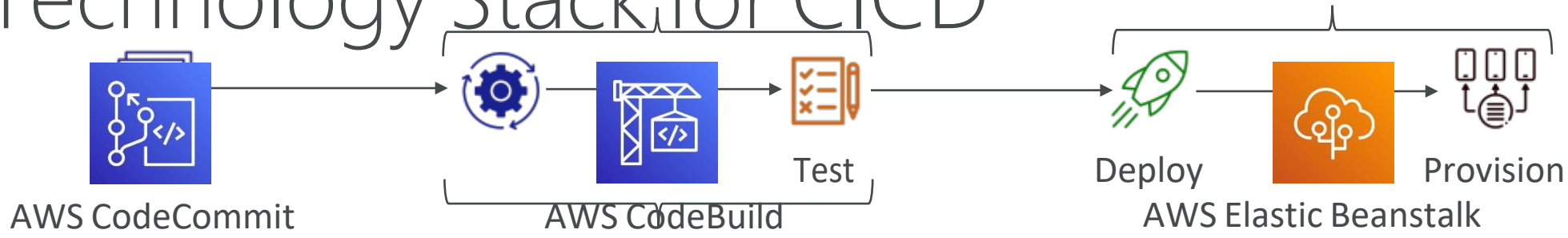
	CodeCommit	GitHub
Support Code Review (Pull Requests)	✔	✔
Integration with AWS CodeBuild	✔	✔
Authentication (SSH & HTTPS)	✔	✔
Security	IAM Users & Roles	GitHub Users
Hosting	Managed & hosted by AWS	- Hosted by GitHub - GitHub Enterprise: self hosted on your servers
UI	Minimal	Fully Featured

AWS CodePipeline

- Visual Workflow to orchestrate your CICD
- **Source** – CodeCommit, ECR, S3, Bitbucket, GitHub
- **Build** – CodeBuild, Jenkins, CloudBees, TeamCity
- **Test** – CodeBuild, AWS Device Farm, 3rd party tools, …
- **Deploy** – CodeDeploy, Elastic Beanstalk, CloudFormation, ECS, S3, …
- Consists of stages:
  - Each stage can have sequential actions and/or parallel actions
  - Example: Build ➔ Test ➔ Deploy ➔ Load Testing ➔ …

- Manual approval can be defined at any stage

Technology Stack for CICD



Code

GitHub

Bitbucket

any 3rd party

Build

Jenkins CI

Code Repository

AWS CodeDeploy

EC2 Instances

On-premises Instances

AWS Lambda

Amazon ECS

a

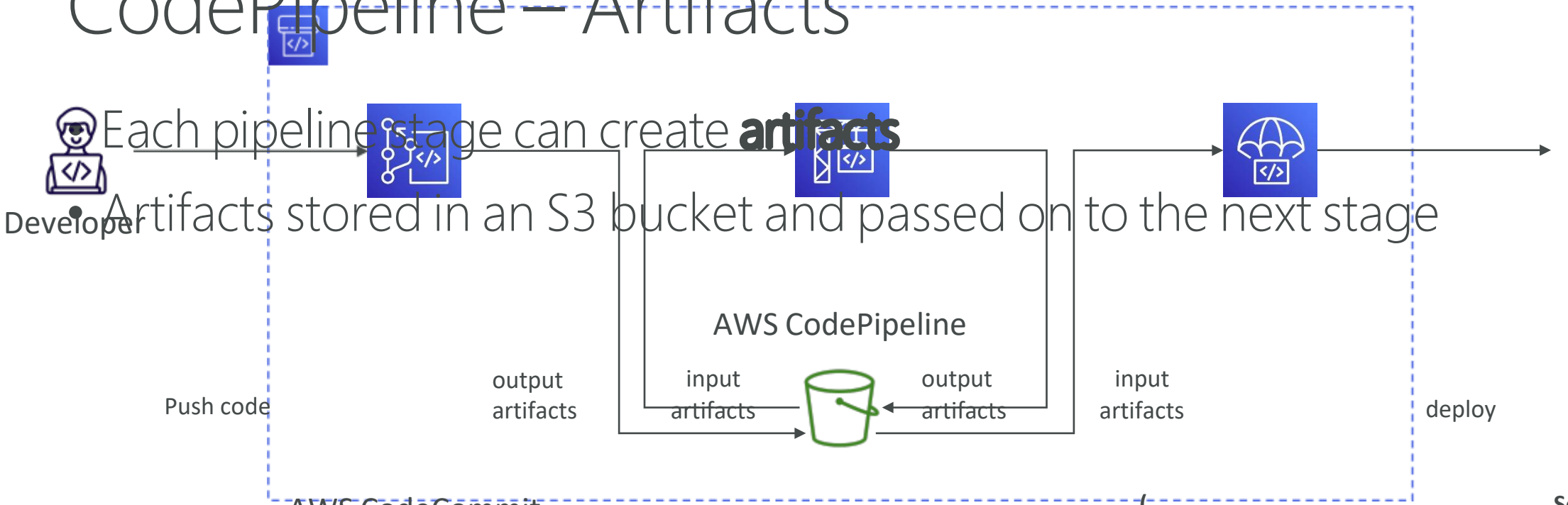
n

y

3!" party CI Servers

CodePipeline – Artifacts

- Each pipeline stage can create **artifacts**
- Artifacts stored in an S3 bucket and passed on to the next stage



(

So

urce)

AWS CodeBuild
(Build)

AWS CodeDeploy
(Deploy)

S3 Bucket

CodePipeline – Troubleshooting

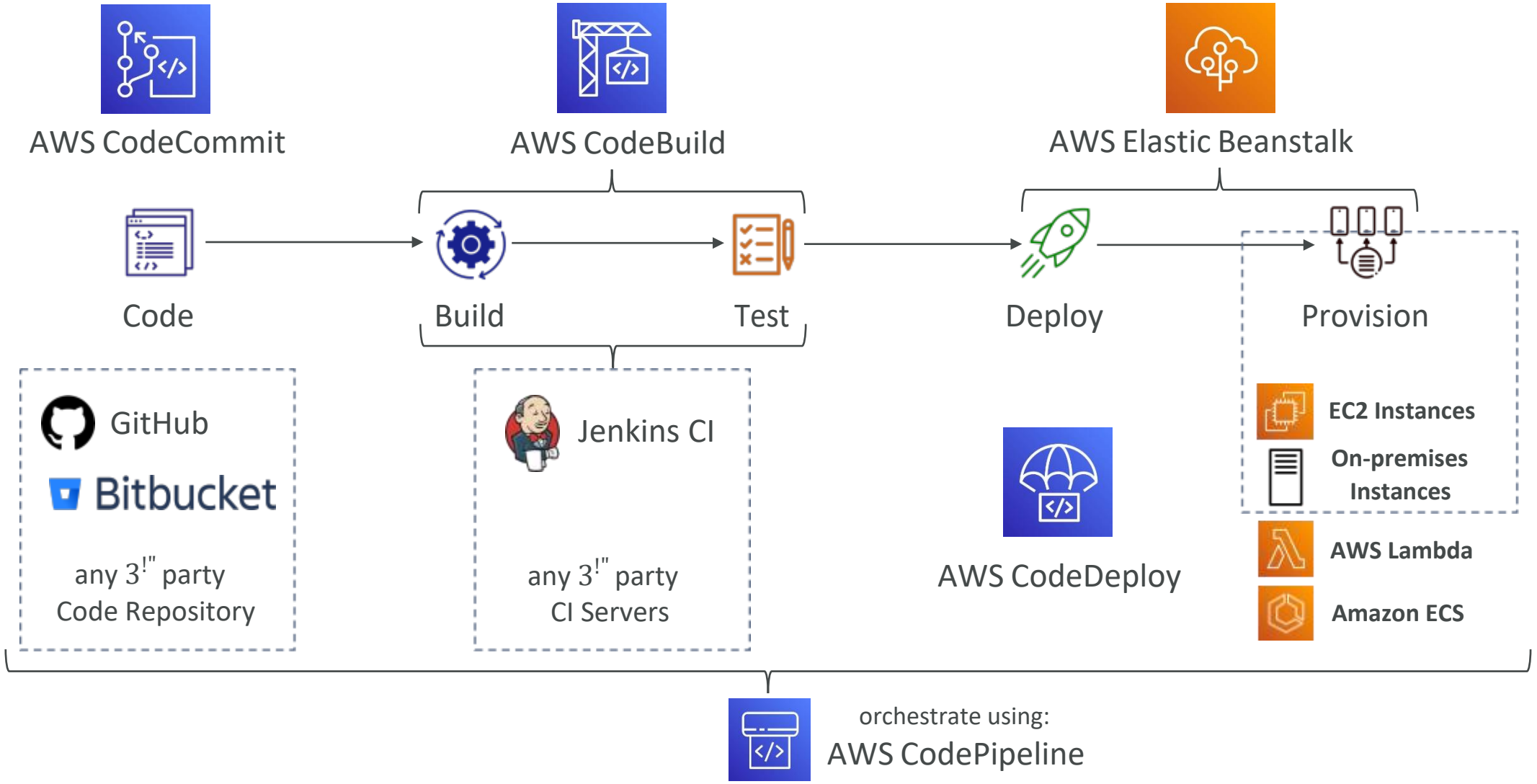
- For CodePipeline Pipeline/Action/Stage Execution State Changes
- Use **CloudWatch Events (Amazon EventBridge).** Example:
  - You can create events for failed pipelines
  - You can create events for cancelled stages

- If CodePipeline fails a stage, your pipeline stops, and you

can get information in the console

- If pipeline can't perform an action, make sure the "IAM Service Role" attached does have enough IAM permissions (IAM Policy)
- AWS CloudTrail can be used to audit AWS API calls

Technology Stack for CICD



AWS CodeBuild

- A fully managed continuous integration (CI) service
- Continuous scaling (no servers to manage or provision – no build queue)
- Compile source code, run tests, produce software packages, …
- Alternative to other build tools (e.g., Jenkins)
- Charged per minute for compute resources (time it takes to complete the builds)
- Leverages Docker under the hood for reproducible builds
- Use prepackaged Docker images or create your own custom Docker image
- Security:
  - Integration with KMS for encryption of build artifacts
  - IAM for CodeBuild permissions, and VPC for network security
  - AWS CloudTrail for API calls logging

AWS CodeBuild

- **Source** – CodeCommit, S3, Bitbucket, GitHub
- **Build instructions**: Code file **buildspec.yml** or insert manually in Console
- **Output logs** can be stored in Amazon S3 & CloudWatch Logs
- Use CloudWatch Metrics to monitor build statistics
- Use CloudWatch Events to detect failed builds and trigger notifications
- Use CloudWatch Alarms to notify if you need "thresholds" for failures

- **Build Projects can be defined within CodePipeline or CodeBuild**

CodeBuild – Supported Environments

- Java
- Ruby
- Python
- Go
- Node.js
- Android
- .NET Core
- PHP
- Docker – extend any environment you like

CodeBuild – How it Works