Architecting on Cloud

John Kidd

Architect responsibilities

Plan

- Set technical cloud strategy with business leads
- Analyze solutions for business needs and requirements

Research

- Investigate cloud services specifications and workload requirements
- Review existing workload architectures
- Design prototype solutions

Build

- Design the transformation roadmap with milestones, work streams, and owners
- Manage the adoption and migration

What is Cloud Native?

- Cloud-native refers to the way an application is built and deployed, specifically leveraging the benefits of the cloud. It implies that the apps exist in the public cloud, rather than within an on-premises datacenter.

- **Traditional development** focuses on the application first and where the infrastructure is hosted as second. Cloud native development is as much a philosophy as it is a set of tools. It's the notion that **we consider all factors from day one**. Through cloud native development we consider where the application is hosted and **what sort of services or technologies can be leveraged** from that platform (e.g. AWS).

Benefits

- Cloud native architectures take full advantage of **on-demand** delivery, **global** deployment, **elasticity**, and **higher-level** services.
- They enable huge improvements in –
  - developer productivity,
  - business agility,
  - scalability,
  - availability,
  - utilization,
  - cost savings.

What about Infrastructure?

Problems with traditional IT approach

Problems with traditional IT approach

- Pay for the rent for the data center
- Pay for power supply, cooling, maintenance
- Adding and replacing hardware takes time
- Scaling is limited
- Hire 24/7 team to monitor the infrastructure
- How to deal with disasters? (earthquake, power shutdown, fire…)

Benefits of Public Cloud

Benefits of Public Cloud

- On-demand self service:
  - Users can provision resources and use them without human interaction from the service provider
- Broad network access:
  - Resources available over the network, and can be accessed by diverse client platforms
- Multi-tenancy and resource pooling:
  - Multiple customers can share the same infrastructure and applications with security and privacy
  - Multiple customers are serviced from the same physical resources
- Rapid elasticity and scalability:
  - Automatically and quickly acquire and dispose resources when needed
  - Quickly and easily scale based on demand
- Measured service:
  - Usage is measured, users pay correctly for what they have used

Amazon Web Services **(AWS)** owns and maintains the network-connected hardware required for these application services, while you provision and use what you need.

Cloud Computing Deployment Models

Cloud

A cloud-based application is fully deployed in the cloud and all parts of the application run in the cloud. Applications in the cloud have either been created in the cloud or have been migrated from an existing infrastructure to take advantage of the benefits of cloud computing. Cloud-based applications can be built on low-level infrastructure pieces or can use higher level services that provide abstraction from the management, architecting, and scaling requirements of core infrastructure.

Hybrid

A hybrid deployment is a way to connect infrastructure and applications between cloud-based resources and existing resources that are not located in the cloud. The most common method of hybrid deployment is between the cloud and existing on-premises infrastructure to extend, and grow, an organization's infrastructure into the cloud while connecting cloud resources to internal system. For more information on how AWS can help you with your hybrid deployment, please visit our hybrid page.

On-premises

Deploying resources on-premises, using virtualization and resource management tools, is sometimes called “private cloud”. On-premises deployment does not provide many of the benefits of cloud computing but is sometimes sought for its ability to provide dedicated resources. In most cases this deployment model is the same as legacy IT infrastructure while using application management and virtualization technologies to try and increase resource utilization.

Cloud Computing Models

Infrastructure as a Service (IaaS)

Infrastructure as a Service, sometimes abbreviated as IaaS, contains the basic building blocks for cloud IT and typically provide access to networking features, computers (virtual or on dedicated hardware), and data storage space. Infrastructure as a Service provides you with the highest level of flexibility and management control over your IT resources and is most similar to existing IT resources that many IT departments and developers are familiar with today.

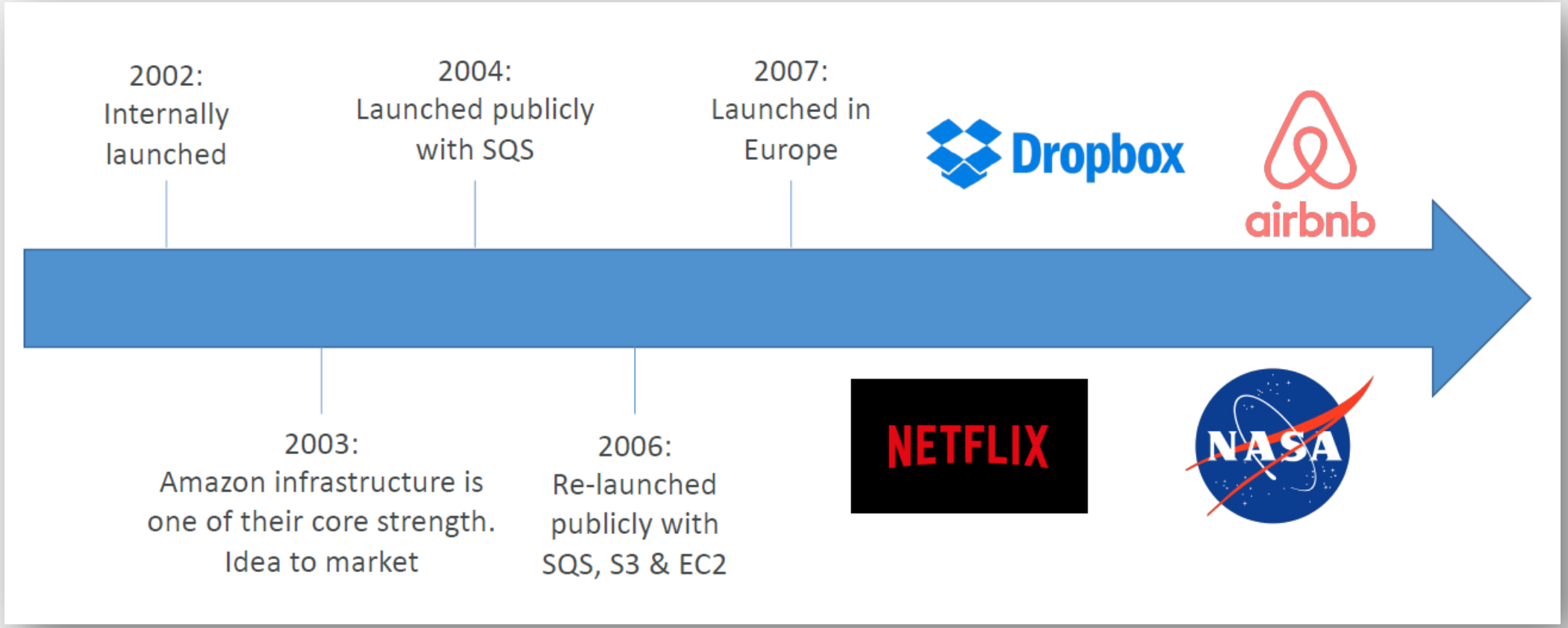
Platform as a Service (PaaS)

Platforms as a service remove the need for organizations to manage the underlying infrastructure (usually hardware and operating systems) and allow you to focus on the deployment and management of your applications. This helps you be more efficient as you don't need to worry about resource procurement, capacity planning, software maintenance, patching, or any of the other undifferentiated heavy lifting involved in running your application.

Software as a Service (SaaS)

Software as a Service provides you with a completed product that is run and managed by the service provider. In most cases, people referring to Software as a Service are referring to end-user applications. With a SaaS offering you do not have to think about how the service is maintained or how the underlying infrastructure is managed; you only need to think about how you will use that particular piece of software. A common example of a SaaS application is web-based email where you can send and receive email without having to manage feature additions to the email product or maintaining the servers and operating systems that the email program is running on.

AWS Cloud History



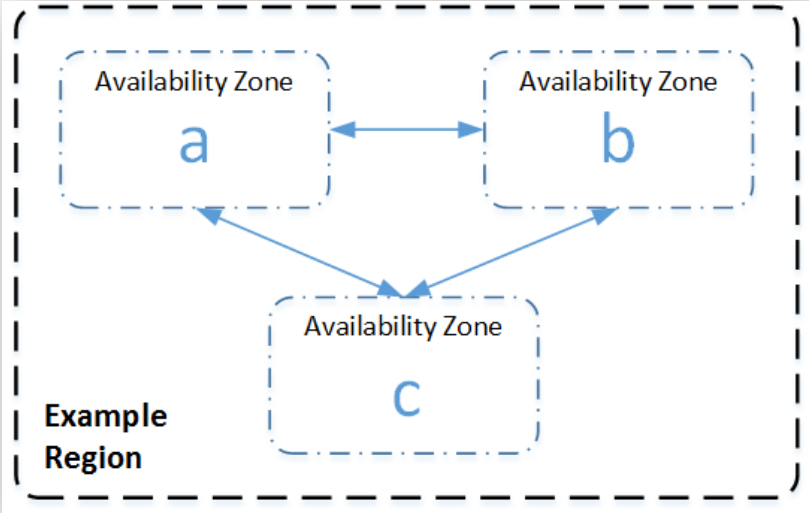
AWS Global Infrastructure Map

Regions & Availability Zones

Regions & Availability Zones

- **Regions**
  - Geographic locations (around the world)
  - Consists of **at least** 3 Availability Zones

- **Availability Zones**
  - Clusters of data centers
  - Isolated in their supplies from other AZ
  - Isolated from failures in other Availability Zones
  - AZs with in a region are connected via private network



Region:
- us-east-1

Availability Zone:
- us-east-1a
- us-east-1b
- us-east-1c

Accessing AWS

- AWS Management Console
  - GUI method via browser
  - Username & password
- AWS CLI
  - Install easily on your laptop / (or use **AWS CloudShell**)
  - Access Key ID & Secret Access Key
- AWS SDK
  - Many programming languages are supported
  - Access Key ID & Secret Access Key

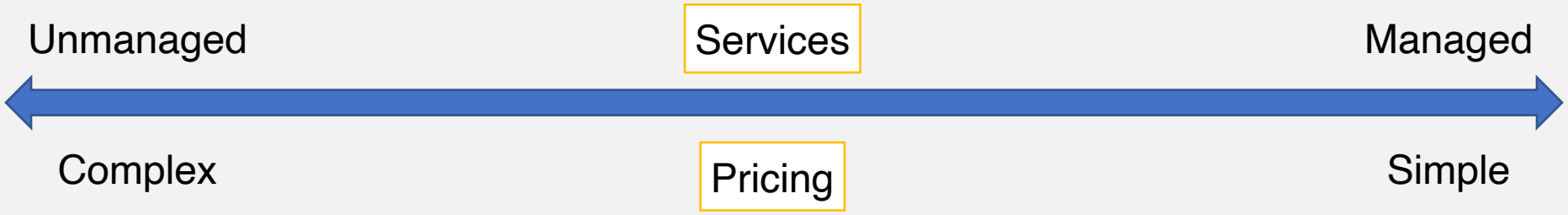
Scope of an AWS Service/Resource

- Global
  - Route53, IAM, CloudFront
- Regional
  - VPC, EFS, Load Balancer, DynamoDB, S3
- AZ based
  - Subnet, EC2, EBS, NAT Gateway, ENI

Data Transfer Considerations

- Data **IN** to AWS is **FREE**.
- Data **OUT** of AWS is **charged**.

The complexity of **Pricing on Cloud** depends on the type of service under consideration.



You need to **understand all the factors** around the pricing of a particular service.

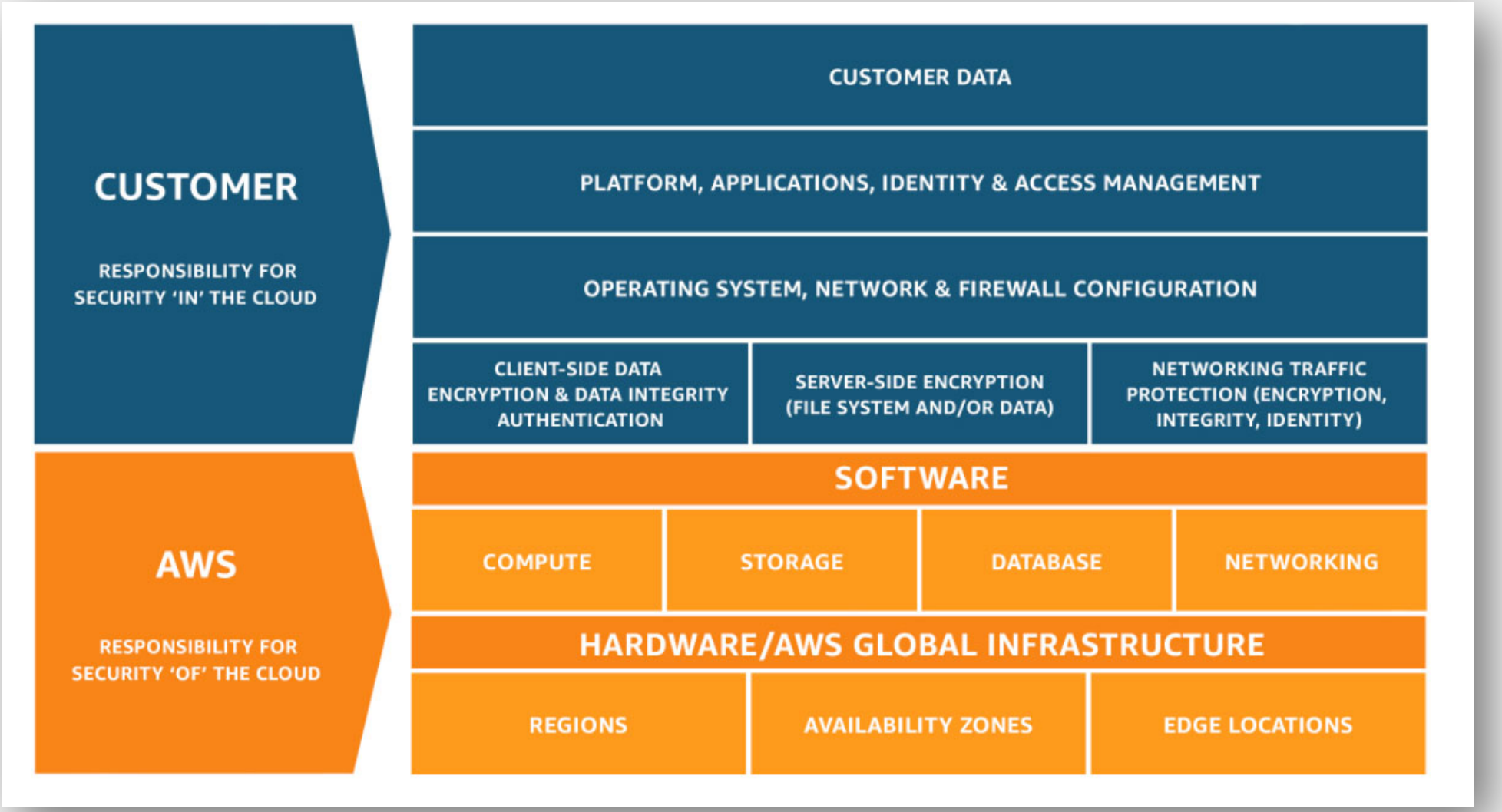
Ensuring HA for an application

Ensuring HA for an application

- Any infrastructure component must be located in **at least 2 AZs**
  - Few AWS services do this automatically, for others you need to architect it

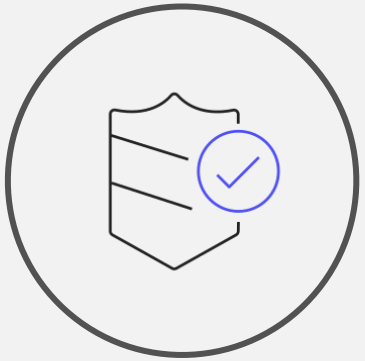
- For even higher Service Availability, **deploy in 2 regions (2 AZs in each region)**

Shared Responsibility Model



AWS Well-Architected Framework

The AWS Well-Architected Tool (AWS WA Tool) helps cloud architects learn, measure, and build using architectural best practices.

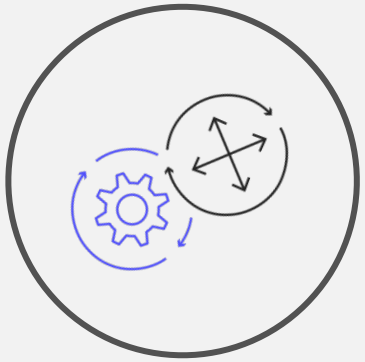


Security

Cost optimization

Reliability

Performance efficiency



Operational excellence

Well-Architected Framework Pillars

Security

- Apply at all layers
- Principle of least privilege
- Multi-factor authentication (MFA)

Cost optimization

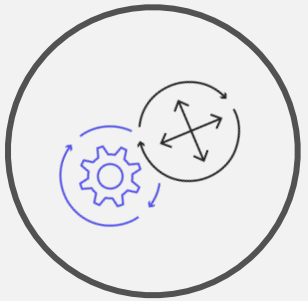
- Analyze and attribute expenditures
- Cost-effective resources
- Stop guessing

Reliability

- Recover from failure
- Test recovery procedures
- Scale to increase availability

Performance efficiency

- Reduce latency
- Use serverless architecture
- Incorporate monitoring



Operational excellence

- Perform operations with code
- Test response for unexpected events

There is a new pillar added recently - **SUSTAINABILITY**

AWS gives a useful WA Tool

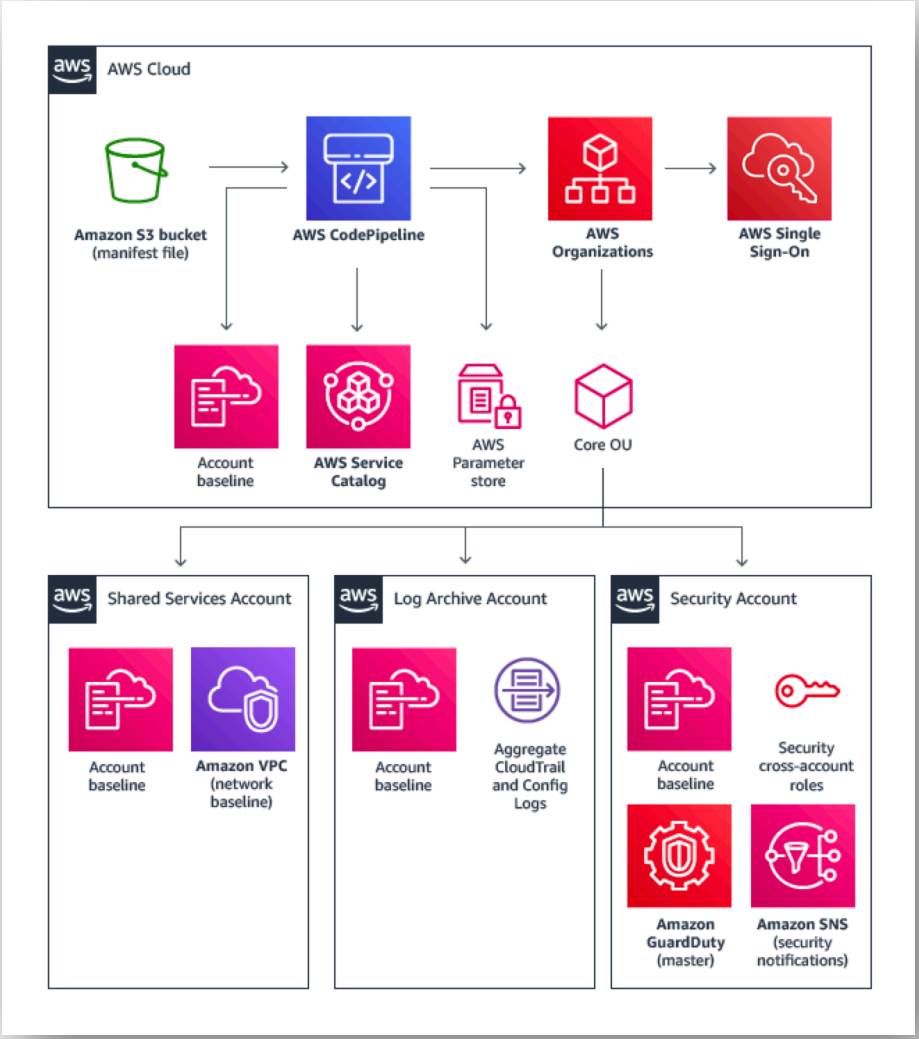
Questions / Quiz Time

IDENTITY & ACCESS MANAGEMENT

How many AWS accounts does a customer need?

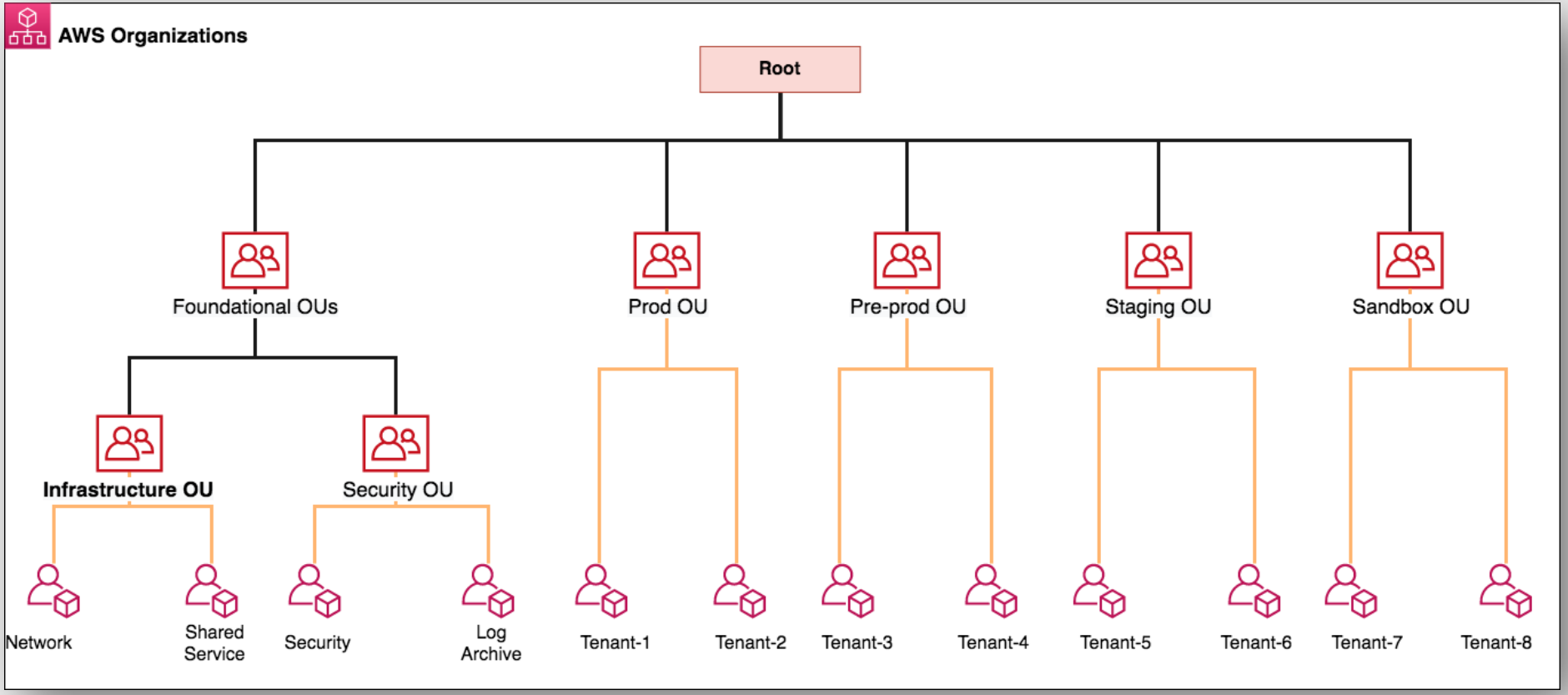
How many AWS accounts does a customer need?

- Typically more than one account is always required.
- First account is the Master Account (or Management Account). Bills of all the accounts get consolidated here. You get tiered discount as well.
- Other accounts (as per the best practices):
  - Log Archive Account
  - Security Account
  - Shared Services Account
  - Other accounts based on Business Unit, Workload, etc.



Managing multiple accounts - AWS Organizations

- SCPs (Service Control Policy) can be applied at any node.
- **Master/Management Account** remains **unaffected** by **SCPs**.



New AWS Account

- New AWS account creation process:
  - ➢Independent account creation
  - ➢Under AWS Organizations
- Free Tier usage & benefits
- Why are Limits/Quota important?

IAM - Users & Groups

- **Root user** gets created by default. It shouldn't be used or shared (E.g. bu01@abc.com)
- Identity and Access Management (IAM) is a global service
- IAM Users are people within your organization, and can be grouped
- IAM Groups only contain users, not other groups
- Users may (or may not) belong to a group
- A user can belong to multiple groups

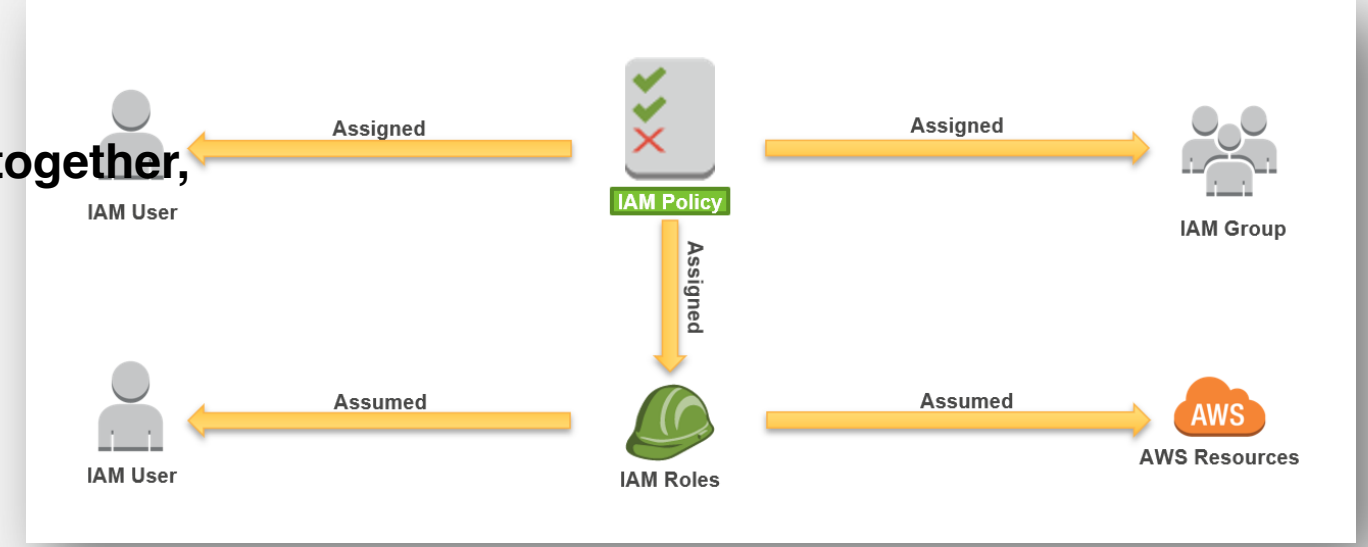
IAM Policies

There are following types of policies:

- **Managed Policies** – these can be attached to multiple IAM entities.
  - Amazon Managed Policies
  - Customer Managed Policies
- **Inline Policies** – these are written specific to a particular IAM entity.
- **Resource based policies**

When two conflicting permissions come together,

DENY always WINS.



Policy Elements

	Description	Required
Effect	Use Allow or Deny to indicate whether the policy allows or denies access.	✔
Principal	Indicate the account, user, role, or federated user to which you want to allow or deny access (only on resource policies).	
Action	Include a list of actions that the policy allows or denies.	✔
Resource	Specify a list of resources to which the actions apply.	✔
Condition	Specify the circumstances under which the policy grants permission.	

IAM - Roles

- Role is an IAM identity that you can create in your account and has specific permissions. An IAM role has some similarities to an IAM user. Roles and users are both AWS identities with permissions policies that determine what the identity can and cannot do in AWS. However, instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it. Also, a role does not have standard long-term credentials such as a password or access keys associated with it. Instead, when you assume a role, it provides you with temporary security credentials for your role session.
- Roles can be used by the following:
  - **People**: Grant cross-account access to employees.
  - **Applications**: Permit applications running on AWS or on premises to make AWS API calls.
  - **AWS Services**: Permit AWS services to make AWS API calls on your behalf.

Examples: EC2 Instance Roles, Roles for Lambda Function, Roles for CloudFormation

IAM Best Practices

- **Never** use Root User's access keys.
- Create individual IAM users.
- Use groups to assign permissions to IAM users.
- Grant **least privilege** always.
- Configure a **strong** password policy.
- **Enable MFA** for all the IAM users.
- Use **roles** for applications that run on Amazon EC2 instances.
- **Delegate by using roles** instead of by sharing credentials.
- **Rotate** credentials regularly.
- Remove unnecessary/inactive users and credentials.
- Use IAM policy conditions for extra security.

AWS Account for the Labs

➢ Please create **only those resources** as instructed during the course.

➢ Kindly **delete all the resources** after a lab gets completed (or when instructed).

- Please **do not abuse this AWS environment** by creating any additional resources.
- Open the given **URL** and bookmark it in your browser for this training.
- Use the credentials & AWS account allocated to you.
- Feel free to ask me doubts while doing the labs. But, you should put your best efforts to explore as well.
- Login and change your password for further use. Please make a note of your password carefully!

Demo

1. Create an IAM role “**myEC2Role**”:
   - Allow this role to be assumed by EC2 service.
   - It should allow upload & download to S3 buckets having names starting with “**ntt**”.

2. Create an IAM user “**Steve**” who could do following:
   - View all resources in the account, except the details of EBS volumes.
   - Do all the EC2 actions; but, could not terminate the EC2 instance with tag “Key: Environment”, “Value: Production”

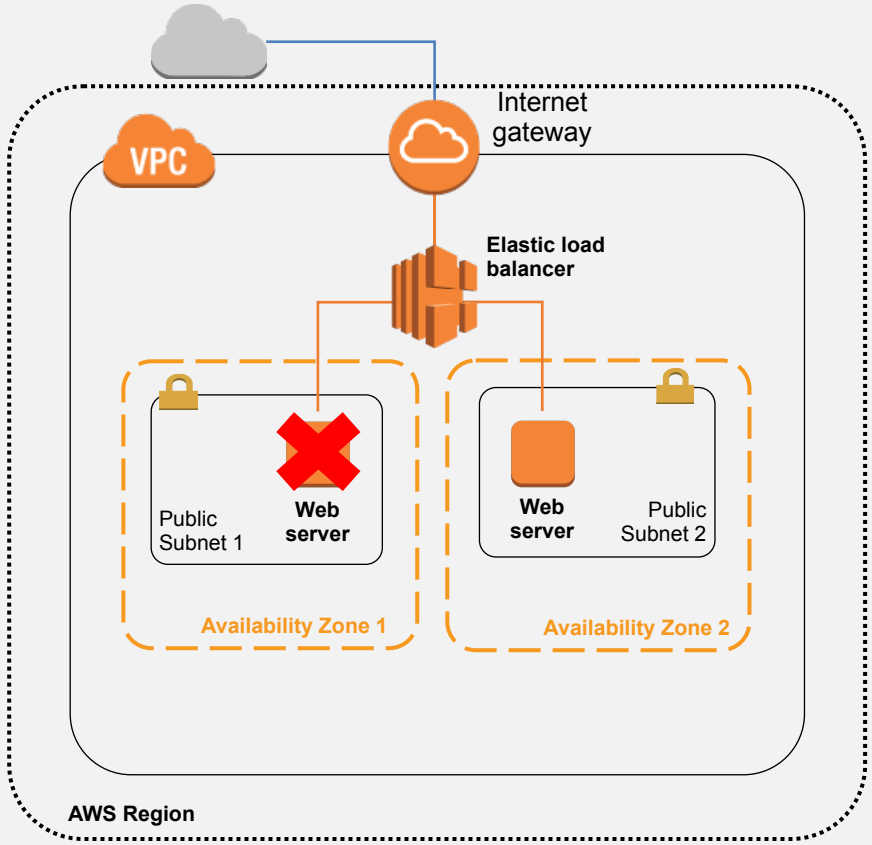
Questions / Quiz Time

How do you choose a region?

- Does the region meet your environment's data sovereignty and compliance requirements?
- How close is the region to your users or data centers?
- Does the region you're considering offer all of the services and features your environment might require?
- Are you choosing the most cost-effective region?

How Many Availability Zones Should I Use?

- Recommendation: Start with two Availability Zones per region.
  - Best practice: If resources in one Availability Zone are unreachable, your application shouldn't fail.
  - Most applications can support two Availability Zones.
  - Using more than two Availability Zones for HA (with in a region) is not usually cost-effective.

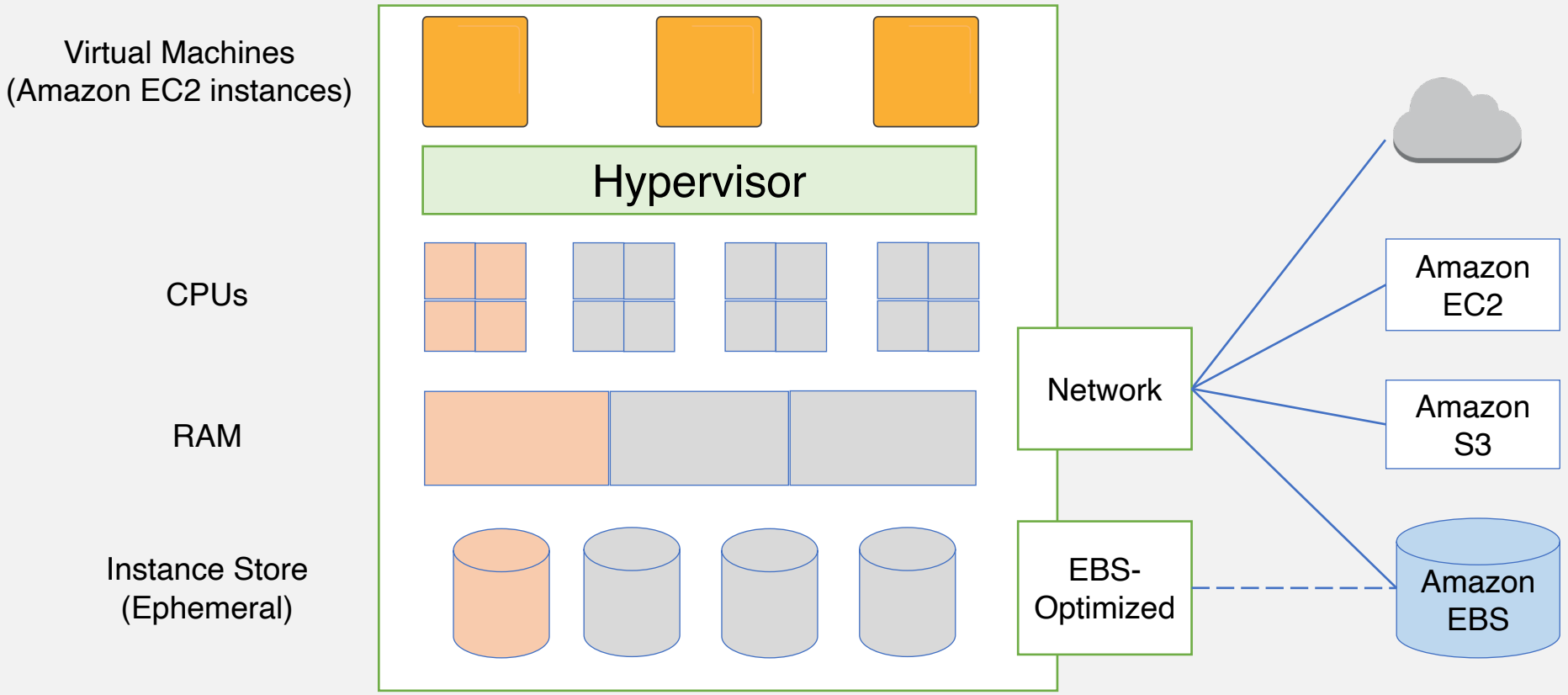


ELASTIC COMPUTE CLOUD (EC2)

EC2 Basics

- **Operating System (OS):** RHEL, SLES, Ubuntu, Amazon Linux, Windows or Mac OS
- How much compute power & cores (CPU)
- How much random-access memory (RAM)
- How much storage space:
  - Network-attached (EBS)
  - Host (Instance Store)
- **Network card:** speed of the card, Public IP address
- **Firewall rules:** security group
- Bootstrap script (runs once during the launch): EC2 User Data

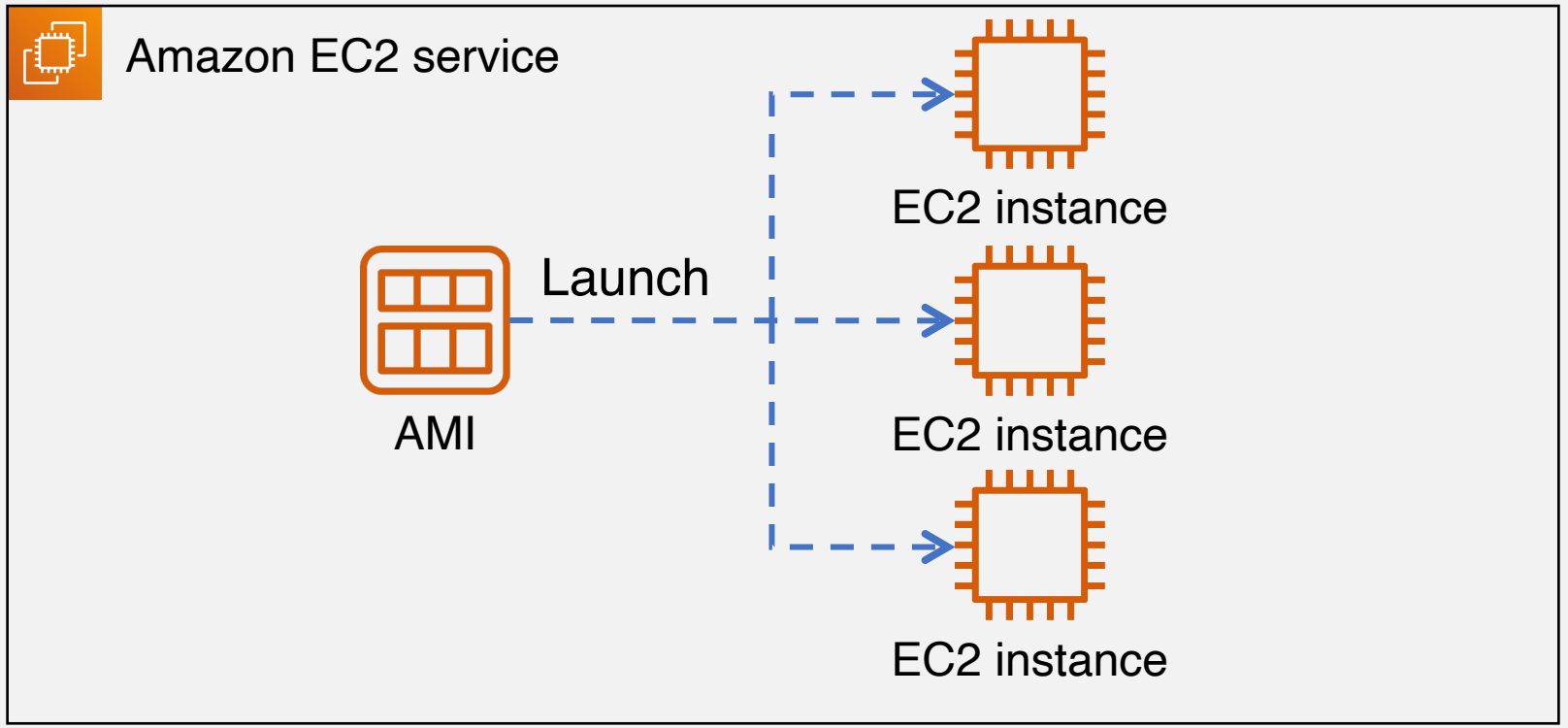
EC2 Architecture



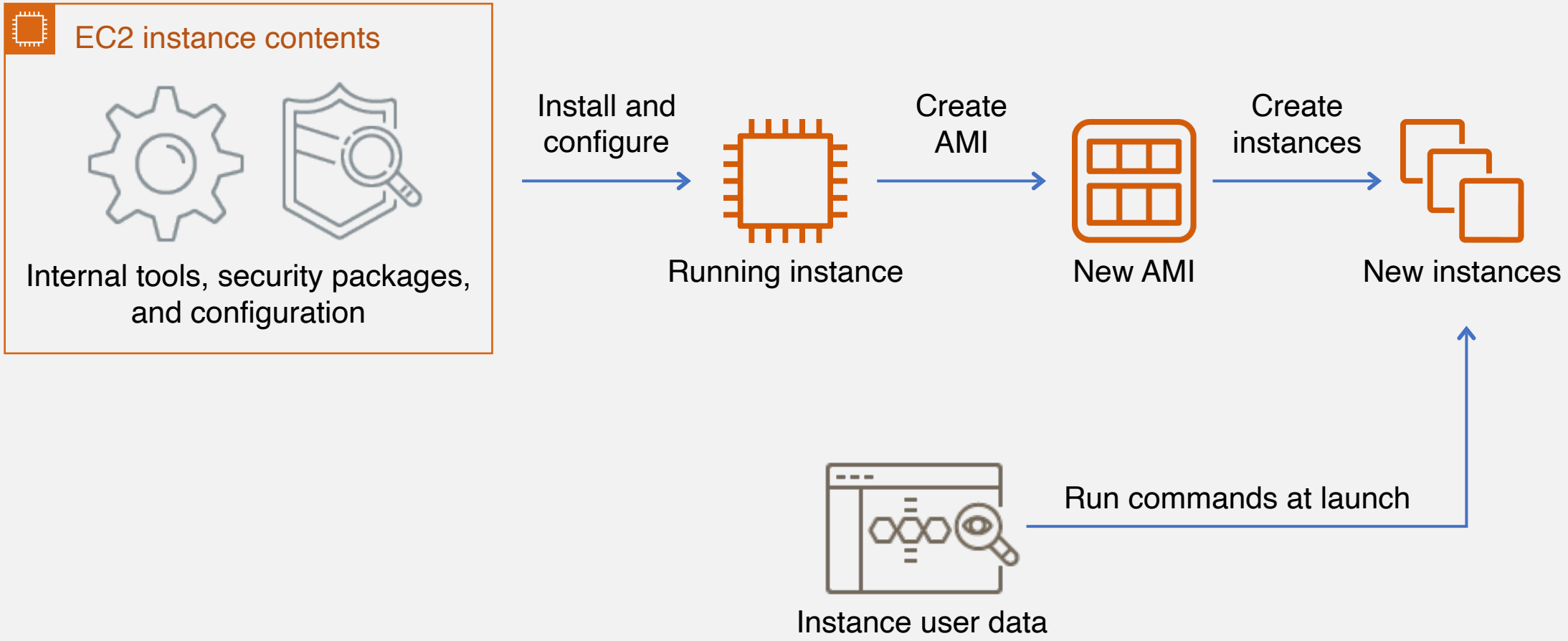
Amazon Machine Image (AMI)

- An AMI includes the following:
  - One or more Amazon Elastic Block Store (Amazon EBS) snapshots.
  - Launch permissions that control which AWS accounts can use the AMI to launch instances.
  - A block device mapping that specifies the volumes to attach to the instance when it's launched.
- When you create an AMI, it captures all the content which is there at the disk level
- A custom AMI gives faster boot / configuration time because all your software is pre-packaged
- AMI is a regional resource. It can be copied across regions
- You can launch EC2 instances from:
  - A Public AMI: AWS provided
  - Your own AMI: you customize and maintain them yourself
  - An AWS Marketplace AMI: an AMI sold or provided by a 3rd party software vendor

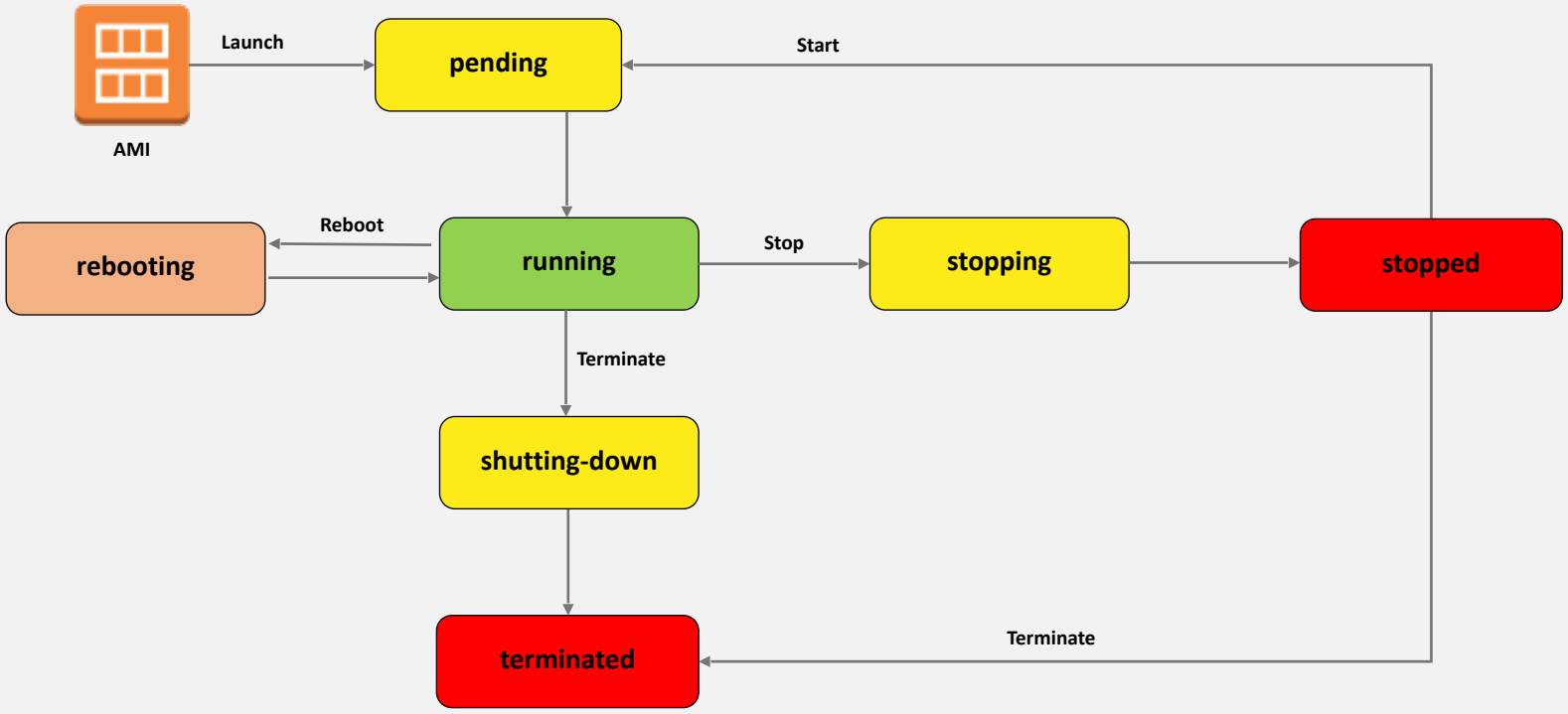
Launching instances from an AMI



Custom AMI creation (Golden AMI approach)



EC2 Lifecycle



Reboot vs. Stop vs. Terminate

Characteristic	Reboot	Stop/Start	Terminate
Host computer	The instance **stays on the same host computer**.	The instance runs on a **new host computer**.	
Public IP address	No change	**New address assigned**	
Elastic IP addresses (EIP)	EIP remains associated with the instance.	EIP remains associated with the instance.	EIP is **disassociated** from the instance.
Instance store volumes	Preserved	**Erased**	**Erased**
EBS volume	Preserved	Preserved	Boot volume is **deleted by default**.
Billing	Instance billing hour doesn't change.	You **stop incurring charges** as soon as state is changed to *stopping*.	You **stop incurring charges** as soon as state is changed to *shutting-down*.

Instance metadata & user data

- **Metadata**
  - Data about the EC2 instance can be used for automation
  - Access it via URL (from the instance) - http://169.254.169.254/latest/meta-data/
- **User data**
  - Runs scripts as root after the instance starts
  - Can be passed to the instance at launch
  - Can be used to perform common automated configuration tasks
  - Access it via URL (from the instance) - http://169.254.169.254/latest/user-data/

EC2 Purchase Options

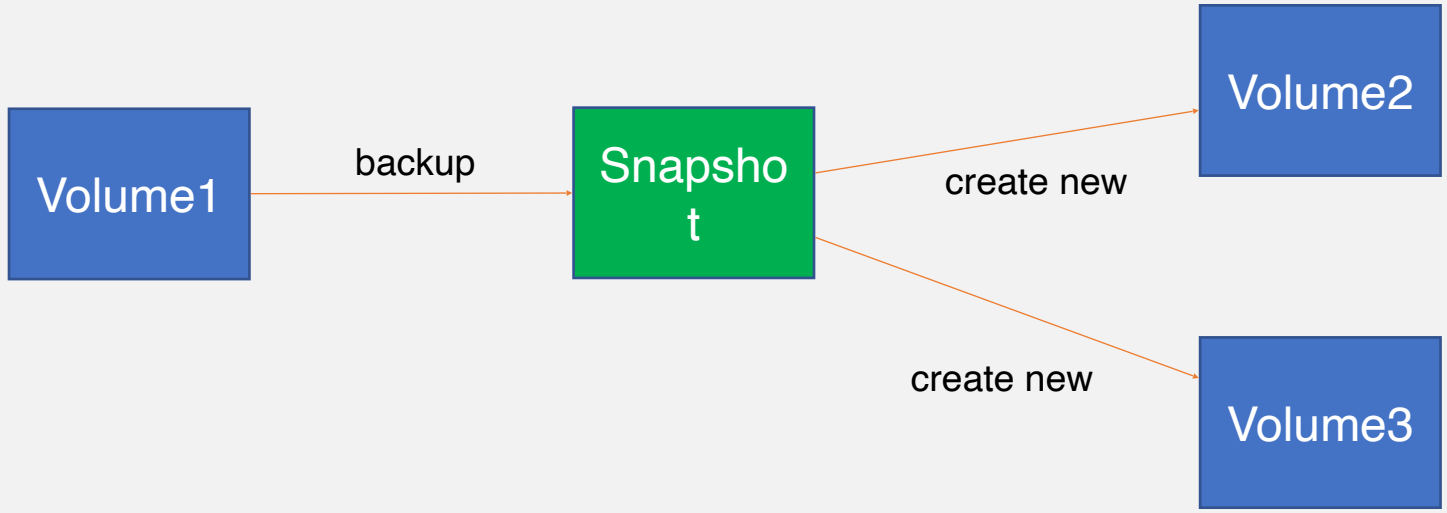
On-Demand	Reserved	Spot Instances
• No commitment • Pay by the hour • Any partial hour converted to full • A new billing cycle starts whenever an instance changes to "Running" state • A billing cycle ends when instance changes to "Stopping" state • Billing cycles don't start at 9am, 10 am etc. • Pay per second (supported for few Operating Systems)	• Two terms available – 1 year or 3 years • 3 Payment options: - Full Upfront - Partial Upfront - No Upfront (not for 3 years term) - Lot of saving in comparison to On-Demand • Gives you Capacity Guarantee as well • You commit the usage for chosen term • You can re-sell on AWS if you choose not to use • Considered for full term (or production usage)	• Unused capacity at AWS is given in market for bidding • Look at pricing history and decide the maximum price • Instances are terminated with 2 minutes notice when market price goes above bid price • If terminated by AWS, last partial hour is free • Optionally, use Spot Block option with bid to block the instance (maximum 6 hours) • Suitable for resumable applications, back end processing

EBS Types

- **SSD**
  - General Purpose SSD (gp2 & gp3) – consistent IOPS, cheaper
  - Provisioned IOPS SSD (io1 & io2) – higher IOPS and costlier
- **HDD**
  - Throughput Optimized HDD (Higher throughput)
  - Cold HDD (lower throughput, cheaper)
  - These cannot act as the boot disk (OS volume)

EBS Snapshots

- Make a backup (snapshot) of your EBS volume at a point in time
- Not necessary to detach volume to do snapshot, but recommended
- A regional resource and stored in Amazon managed S3
- Can copy snapshots across regions



Time allocated: 20 mins

Lab 1

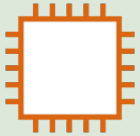
1. Launch an EC2 instance (Amazon Linux, t2.micro). Connect to it via **EC2-connect** option from your browser. Install **Apache web server** on it. Access this public website from a browser. You may use your mobile to access it as well.
2. Stop this EC2 instance and then create an AMI of it. Launch another instance from the AMI and check if all your changes persist? (i.e. webserver installation).
3. After completing this, **terminate all** the EC2 instances. Also, deregister all the AMIs and delete the snapshots & volumes from your account.

Questions / Quiz Time

VIRTUAL PRIVATE CLOUD (VPC)

Public and private IP addresses

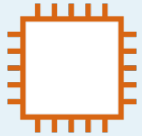
Public IP Address



Public IP: 54.56.9.10

Public IP addresses are IPv4 addresses reachable from the internet.

Private IP Address



Private IP: 172.31.1.90

Private IP addresses are not reachable from the internet.

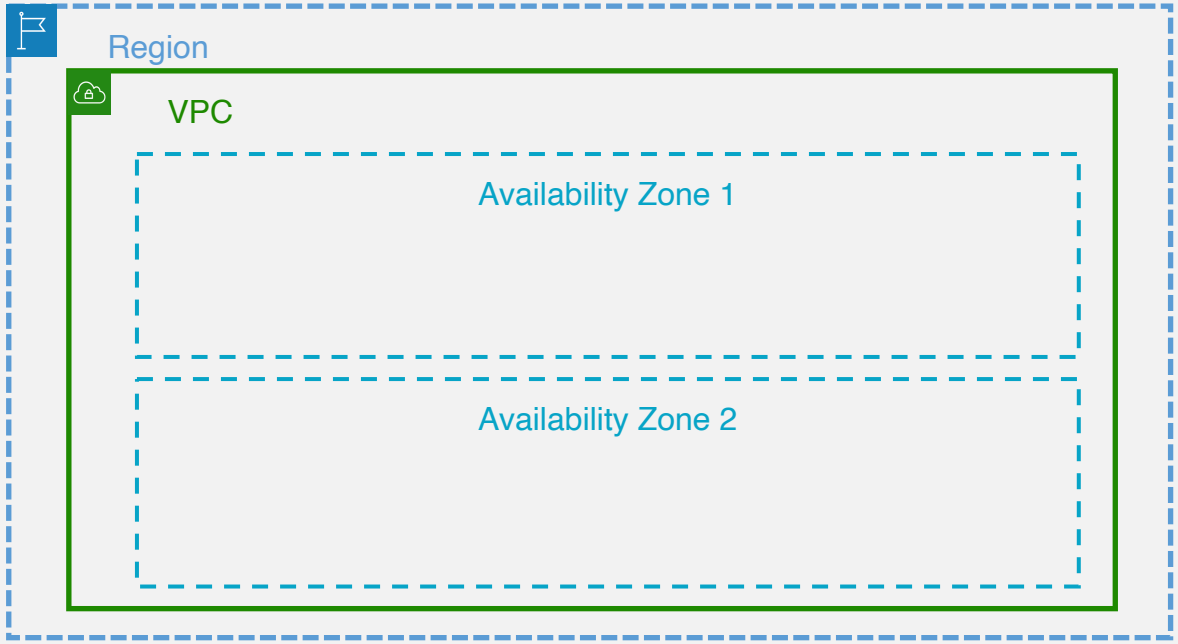
Amazon Virtual Private Cloud (VPC)

- Virtual network; isolated portion of AWS cloud that you design
  - Optional dedicated tenancy
  - Supports logical separation with subnets
  - Fine-grained security
- Private address ranges specified using Classless Inter-Domain Routing (CIDR) notation
- AWS VPCs can use CIDR ranges between /16 and /28.
- For every one step a CIDR range increases, the total number of IPs is cut in half:

CIDR / Total IPs						
/16	/17	/18	/19	/20	/21	/22
65,536	**32,768**	**16,384**	**8,192**	**4,096**	**2,048**	**1,024**
/23	/24	/25	/26	/27	/28	
512	**256**	**128**	**64**	**32**	**16**	

VPC

- VPCs can span across multiple Availability Zones within a region.
- VPCs have an implicit router and a default route table that routes local traffic within the VPC.
- VPCs are private networks until associated with an Internet gateway and a route table rule routing traffic through it.
- **Default Amazon VPCs** are provisioned at account creation in every region. Do not use this for your customer workloads.

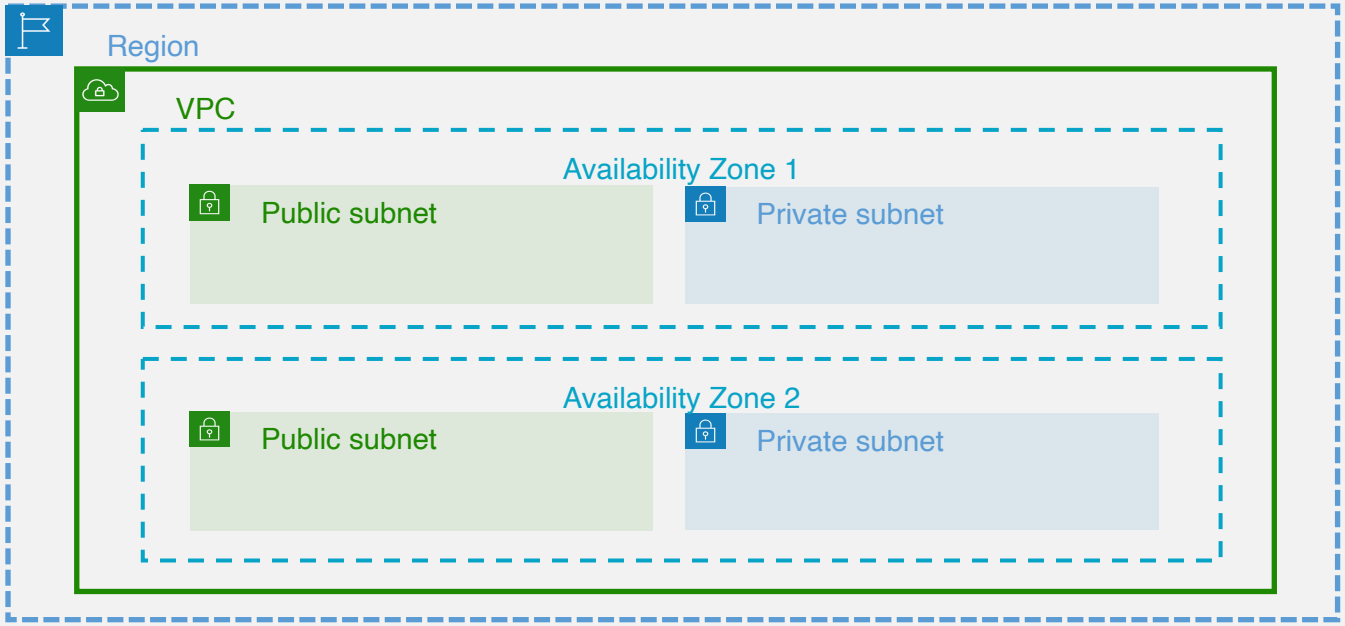


Default route (Can't change)

Destination	Target
172.31.0.0/16	local

Subnets

- Subnets segment VPC address ranges even further.
- Subnets can exist within one and only one Availability Zone.
- Subnet CIDR blocks within a VPC must not overlap.
- Subnet inbound and outbound traffic can be restricted using NACLs.
- **Recommendation**: Allocate substantially more IPs for private subnets than for public subnets.



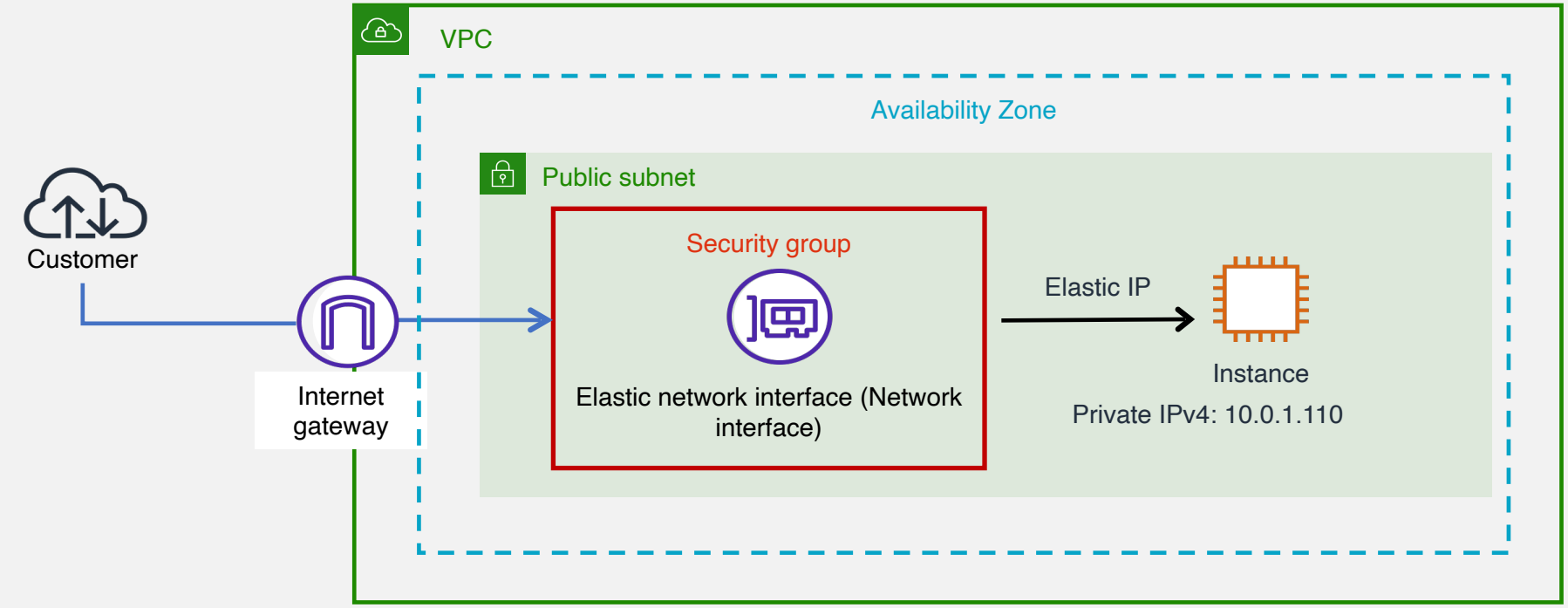
Other VPC concepts

- **Internet gateways** permit communication between instances in your VPC and the internet.
- They are horizontally scaled, redundant, and highly available by default.
- They provide a target in your subnet route tables for internet-routable traffic.
- **Public subnets** allow external communication **through public IP addresses**.
- There is no automatic outbound routing. You access the internet by way of the internet gateway.
- **NAT Gateway** helps instances in the private subnet to access internet.
- Traffic in the VPC stays local.

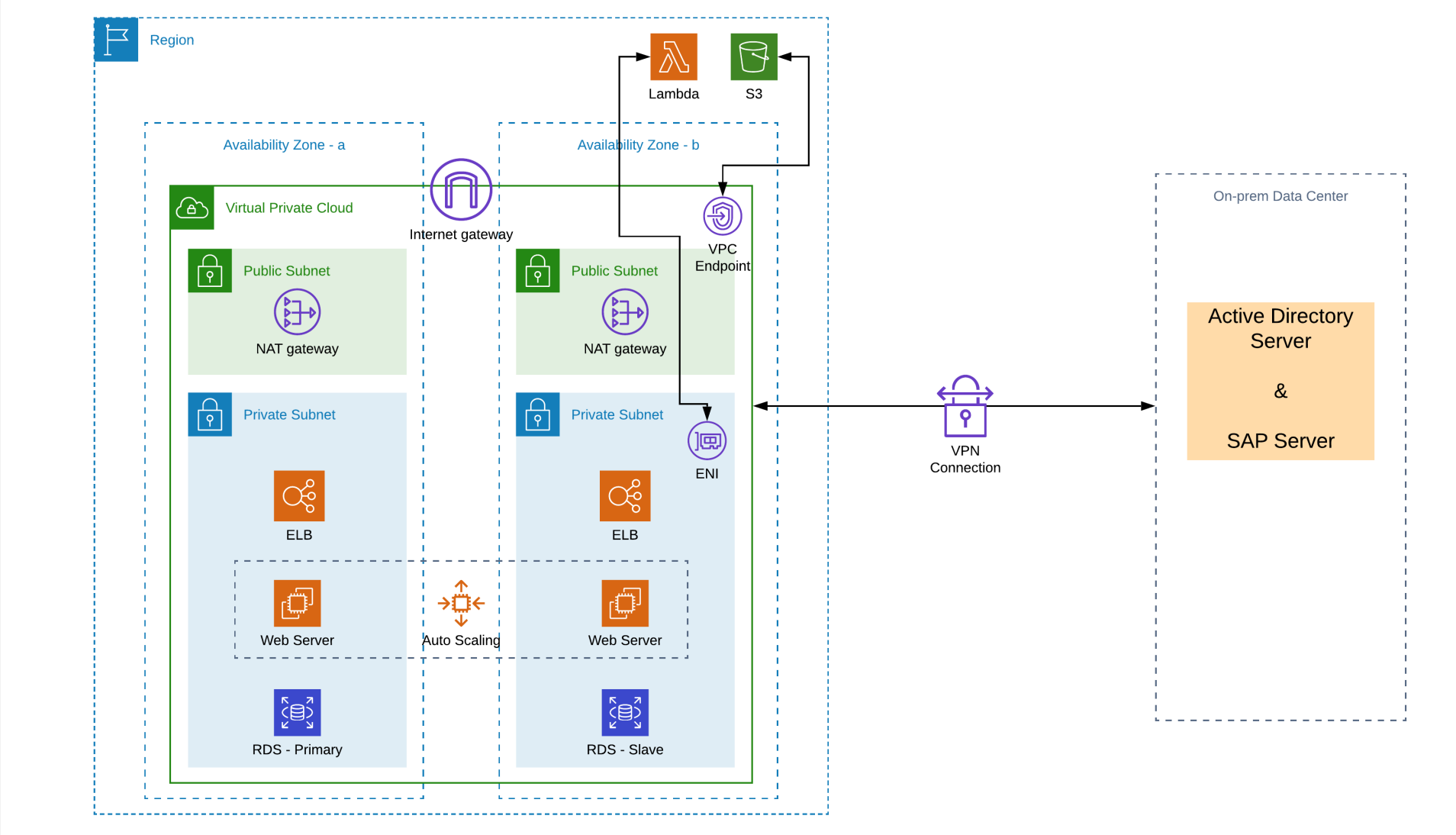
Elastic Network Interface

An elastic network interface is a virtual network interface that:

- Can be moved across EC2 instances in the **same Availability Zone**
- Maintains its private IP address, Elastic IP address, and MAC address



VPC in action – A Production Setup



Security Groups vs NACL

Security Group	Network ACL
Associated to an elastic network interface and implemented in the hypervisor (instance level)	Associated to a subnet and implemented in the network
Supports Allow rules only	Supports Allow rules and Deny rules
A stateful firewall	A stateless firewall
All rules are evaluated before deciding whether to allow traffic	All rules are processed in order when deciding whether to allow traffic
Needs to be manually assigned to instances	Automatically applied when instances are added to subnet
Requires configuration to allow communication	Allows communication by default

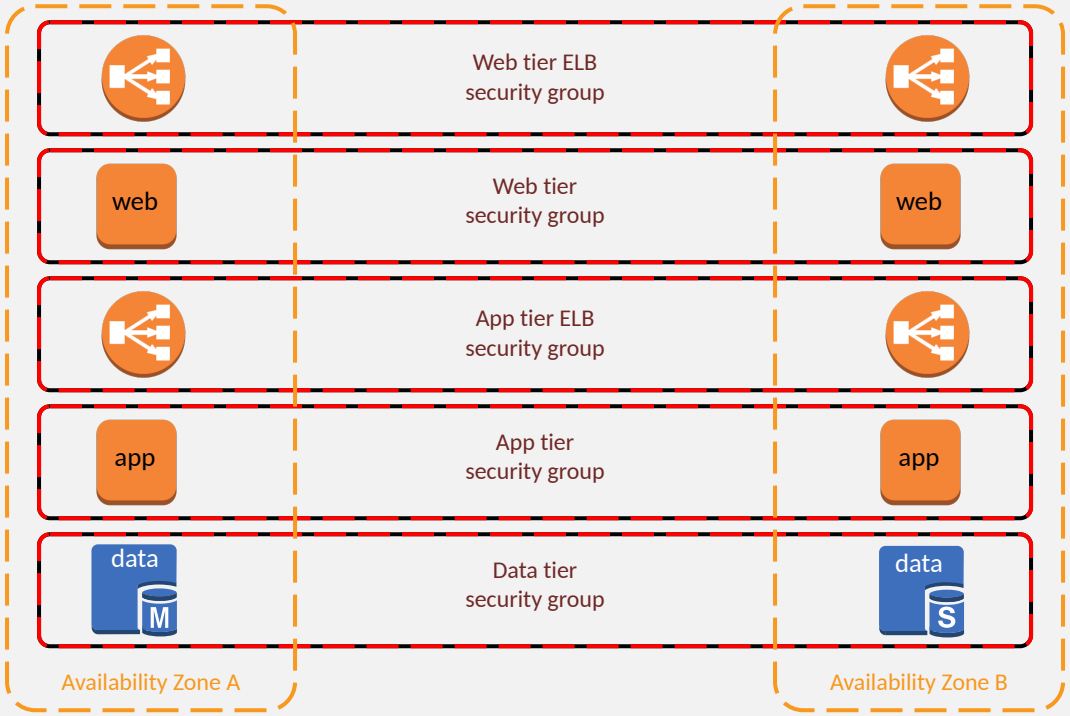
VPC Endpoints

VPC Endpoints allow connectivity to AWS services in a private manner via Amazon's network (without going via internet).

There are 2 types of VPC endpoints:

1. **Gateway Endpoint** – works for S3, DynamoDB.
2. **Interface Endpoint** – works for CloudWatch, Systems Manager, SQS, SNS, and many other services.

Security Group Chaining



Inbound Rule
Allow TCP Port 443
Source: 0.0.0.0/0 (Any)

Inbound Rule
Allow TCP Port 80
Source: Web tier ELB

Inbound Rule
Allow TCP Port 8080
Source: Web tier

Inbound Rule
Allow TCP Port 8080
Source: App tier ELB

Inbound Rule
Allow TCP Port 3306
Source: App tier

Questions / Quiz Time

Time allocated: 45 mins

Lab 2

- Create a new VPC with CIDR **10.0.0.0/26**
- Divide this into 4 subnets of equal size across 2 AZ (e.g. a1, a2, b1, b2)
- Make a1 & b1 as Public subnets. a2 & b2 as Private Subnets.
- Create a NAT gateway in Public subnet and update Route table of Private subnet to use it.
- Launch a (t2.micro) **Linux** instance in Public subnet and another instance in Private subnet.
- Connect to Public & Private instances separately using EC2 Connect.
- Verify if the internet is accessible on both the instances!
- **Delete the NAT Gateway and then release the Elastic IP**. Check again if the internet is accessible on both the instances!
- Stop the Public Instance. Terminate the Private Instance.

DATABASES ON AWS

Unmanaged vs. Managed Services

Unmanaged:

Scaling, fault tolerance, and availability are managed by you.

Amazon Elastic Compute Cloud (EC2)

Managed:

Scaling, fault tolerance, and availability are typically built in to the service.

Amazon Relational Database Service (RDS)

Relational and Non-Relational Databases

	Relational	Non-Relational
Data Storage	Rows and Columns	Key-Value
Schemas	Fixed	Dynamic
Querying	Using SQL	Focused on collection of documents
Scalability	Vertical	Horizontal

Relational

ISBN	Title	Author	Format
9182932465265	Cloud Computing Concepts	Wilson, Joe	Paperback
3142536475869	The Database Guru	Gomez, Maria	eBook

Non-Relational

```
{
        ISBN: 9182932465265,
        Title: “Cloud Computing Concepts”,
        Author: “Wilson, Joe”,
        Format: “Paperback”
}
```

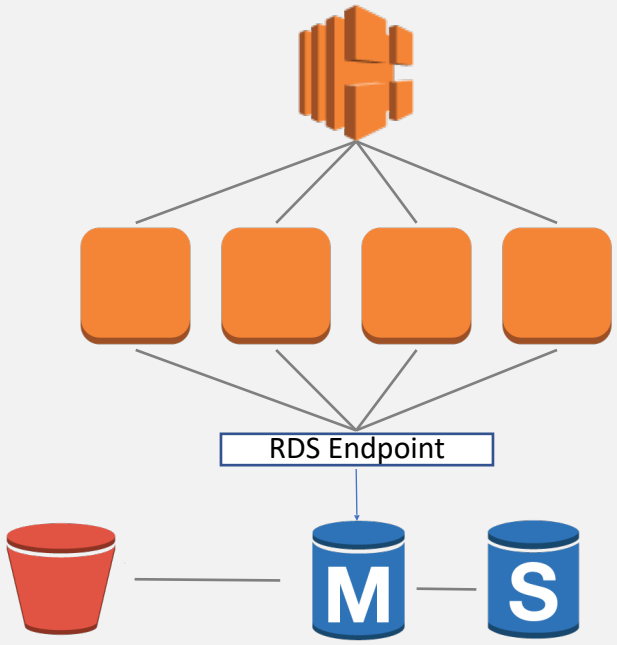
Different Database Services

- Amazon RDS, Aurora – Relational
- Amazon DynamoDB – NoSQL
- Amazon ElastiCache – In memory
- Amazon Redshift – Datawarehouse
- AWS Database Migration Service – For migrating data

RDS

- RDS could be provisioned as Single-AZ or Multi-AZ.
- With Multi-AZ operation, your database is **synchronously** replicated to an instance in another Availability Zone in the same AWS Region.
- **Failover** to the standby automatically occurs in case of master database failure. There will be delay for few seconds for sure.
- Planned maintenance is applied first to standby databases.
- Offers MySQL, PostgreSQL, SQL Server, Oracle, MariaDB.

A Resilient, Durable Application Architecture



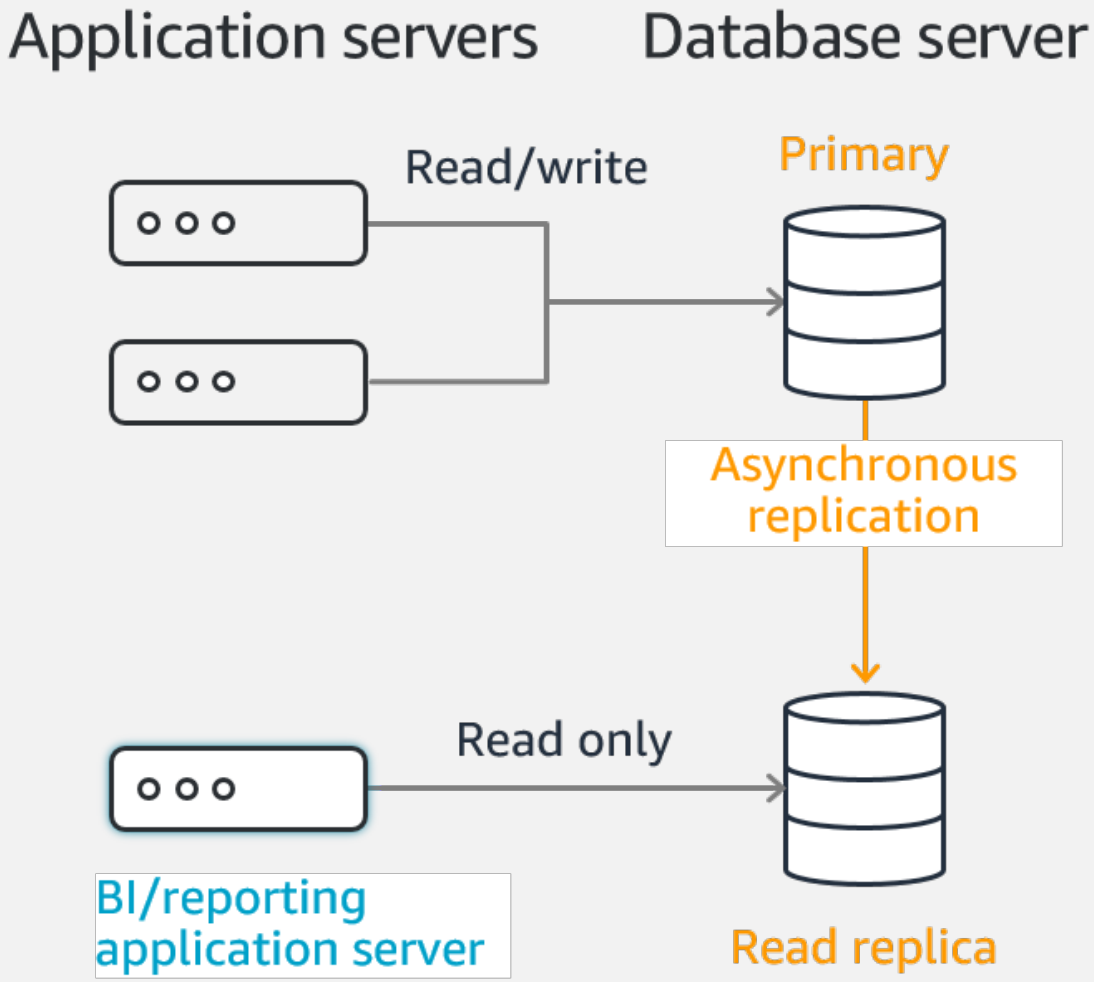
Elastic Load Balancing load balancer instance

Application, in Amazon EC2 instances

Amazon RDS database instances: Master and Multi-AZ standby

DB snapshots in Amazon S3

RDS Read Replica



Global replication: Logical

Faster disaster recovery and enhanced data locality

- Promote read replica to a master
  for faster recovery
  in case of a disaster
- Bring data close to your customer's
  applications
  in different regions
- Promote to a master
  for easy migration

RDS Backup & Restore

Automatic Backups:

- Restore your database to a point in time.
- Are enabled by default.
- Let you choose a retention period up to 35 days.

Manual Snapshots:

- Let you build a new database instance from a snapshot.
- Are initiated by the user.
- Persist until the user deletes them.
- Are stored in Amazon managed S3.

Restore Process:

- You can restore till current time **minus ten minutes** (which uses Automatic backups).
- New endpoint gets created for the restored DB.

RDS Optimizes Developer Productivity

- Let developers focus on innovation:
  - Query construction
  - Query optimization

Offload the operational burdens:

- ✓ Migration
- ✓ Backup and recovery
- ✓ Patching
- ✓ Software upgrades
- ✓ Storage upgrades
- ✓ Frequent server upgrades
- ✓ Hardware crash

RDS Pricing

- You have both the models available:
    - On-demand pricing
    - Reserved pricing

Amazon Aurora

Enterprise database at open source price

Delivered as a **managed** service

Speed and **availability** of high-end commercial databases

Simplicity and **cost-effectiveness** of open source databases

Drop-in **compatibility** with MySQL and PostgreSQL

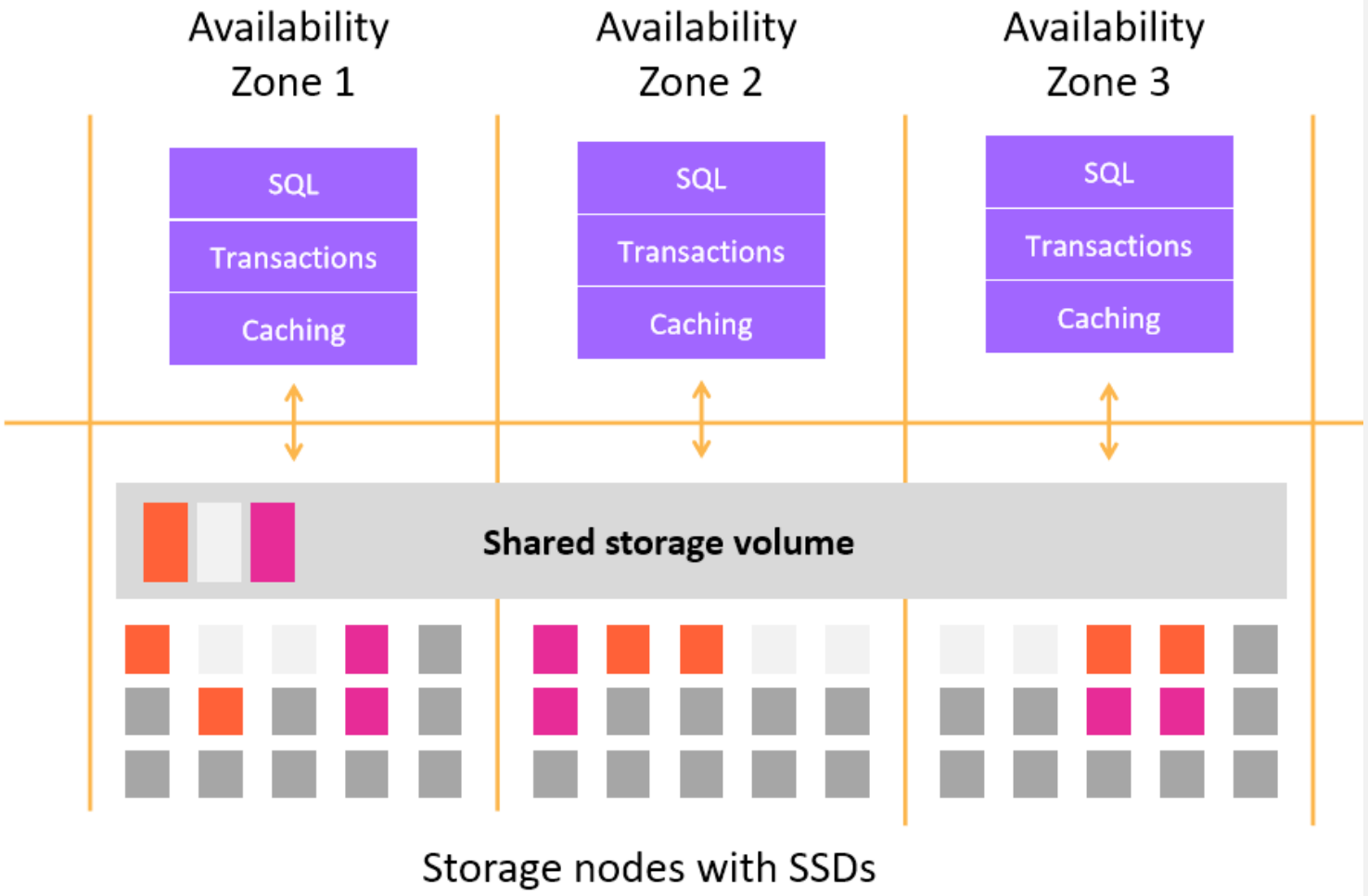
Simple **pay-as-you-go** pricing

Scale-out, distributed architecture

Purpose-built log-structured distributed storage system designed for databases

Storage volume is striped across hundreds of storage nodes distributed over three different Availability Zones

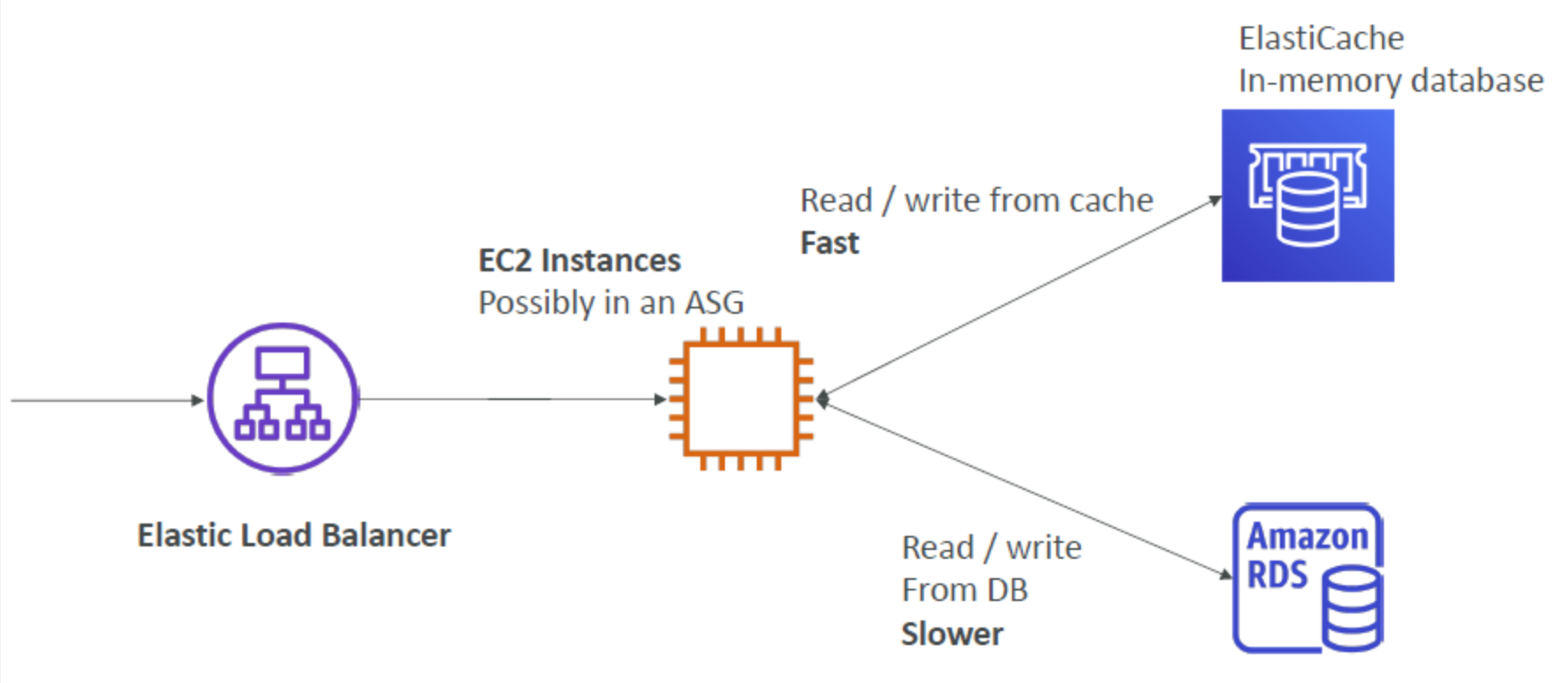
Six copies of data, two copies in each Availability Zone to protect against AZ+1 failures



Amazon ElastiCache

- Caches are in-memory databases with high performance, low latency. Helps reduce load off databases for read intensive workloads
- **Fully-managed in-memory** data store (caching service, to boost DB read performance). AWS takes care of OS maintenance / patching, optimizations, setup, configuration, monitoring, failure recovery and backups
- It is a **remote caching service**, or a side cache i.e. separate dedicated caching instance
- Two flavors (both are open-source key-value stores)
  - Amazon ElastiCache for Redis
  - Amazon ElastiCache for Memcached
- **Sub-millisecond latency** for real-time applications
- The application logic needs to implement the storage and retrieval of data from the cache

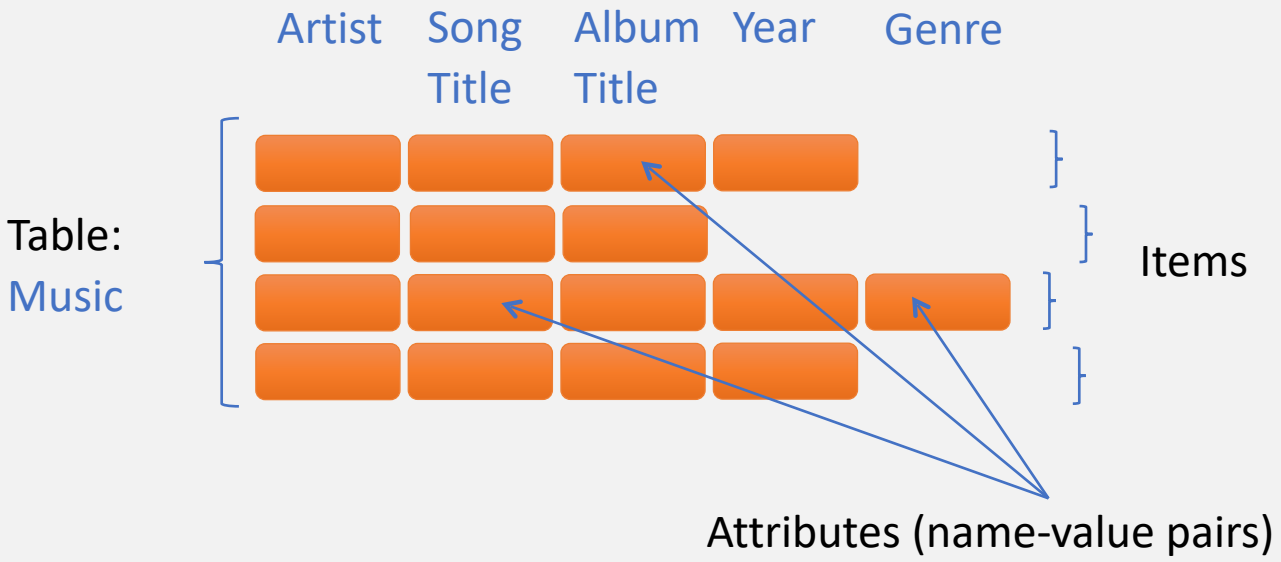
Amazon ElastiCache Example



DynamoDB

- Allows you to store any amount of data with **no limits**.
- Provides fast, predictable performance using **SSDs**.
- Allows you to easily provision and change the **request capacity** needed for each table.
- Is a **fully managed, NoSQL** database service.
- No throughput limits.
- **Single-digit** milliseconds latency.
- Regional resource with data stored in multiple AZs.

DynamoDB Data Model



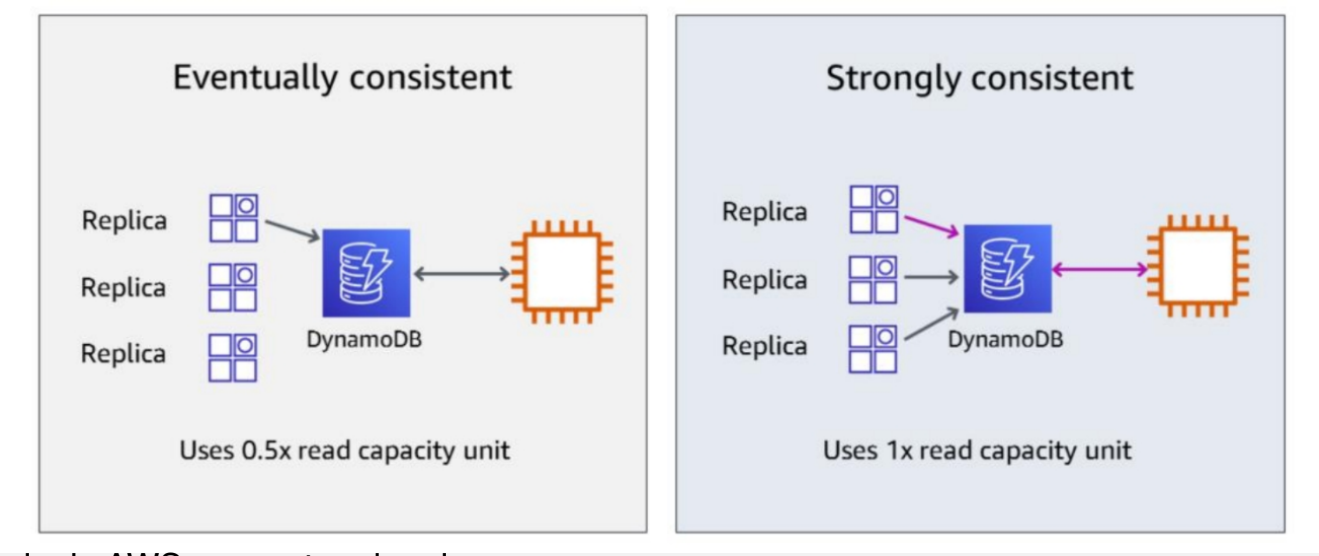
Primary Keys

Table: Music
Partition Key: Artist
Sort Key: Song Title

(DynamoDB maintains a sorted index for both keys)
You should choose a Partition Key that avoids Hot Partitions.

Consistency Options

- **Read Consistency:** strong consistency, eventual consistency, and transactional
- **Write Consistency:** standard and transactional
- **Strong Consistency**
  - The most up-to-date data
  - Must be requested explicitly
- **Eventual Consistency**
  - May or may not reflect the latest copy of data
  - Default consistency for all operations
  - 50% cheaper than strong consistency
- **Transactional Reads and Writes**
  - For ACID support across one or more tables within a single AWS account and region
  - 2x the cost of strongly consistent reads
  - 2x the cost of standard writes

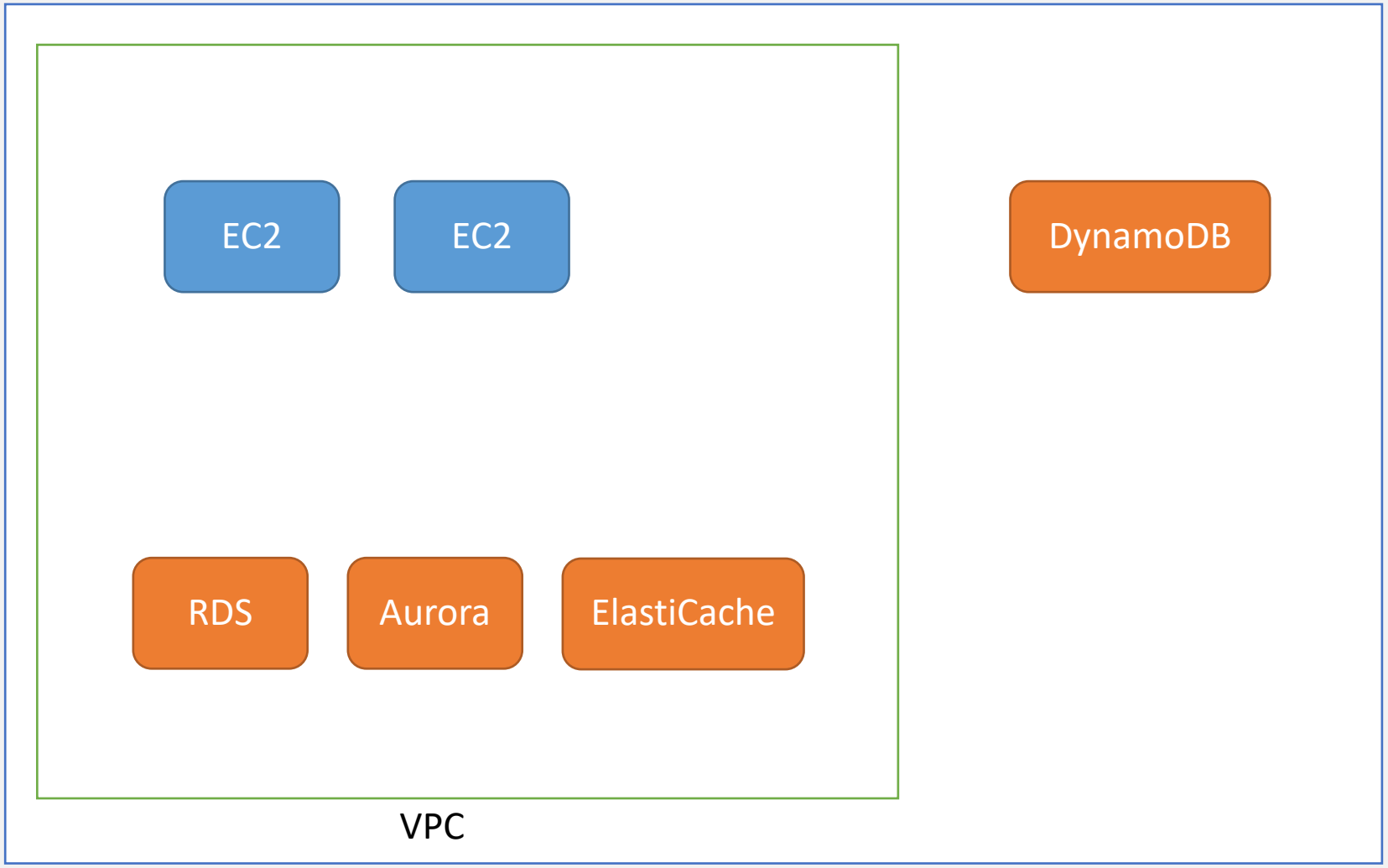


DynamoDB Throughput

- **Provisioned capacity mode**
  - With provisioned capacity mode, you specify the number of reads and writes per second that you expect your application to require. You can use auto scaling to automatically adjust your table's capacity based on the specified utilization rate to ensure application performance while reducing costs.

- **On-demand capacity mode**
  - With on-demand capacity mode, DynamoDB charges you for the data reads and writes your application performs on your tables. You do not need to specify how much read and write throughput you expect your application to perform because DynamoDB instantly accommodates your workloads as they ramp up or down.

Connectivity of different Databases on AWS



Inside and outside of VPC

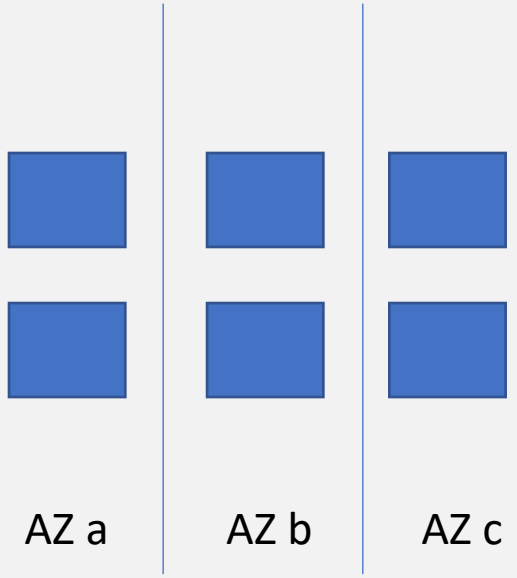
Time allocated: 40 mins

Lab 3

- Launch a Multi-AZ RDS (**PostgreSQL**) instance in private subnets of your VPC. (type – db.t3.micro, Storage – GP SSD – 30GB)
- Once it is ready, connect it via the Public EC2 instance you had launched earlier.
- Follow the instructions and use **psql** to access the RDS database.
  - Create a table and insert data into it.
- Use python code on your EC2 instance to access the data RDS table.
- Check the private IP of the RDS instance serving you. Make a note of it.
  - You can use “ping –a <<RDS endpoint>>”
- Now, reboot your RDS **with failover option** and check the private IP of the RDS instance serving you. Make a note of this IP as well.
- Verify these in the ENI section (under EC2 Dashboard).

Storage Services

- Overview of S3, EBS, EFS
- Pricing of S3 (different storage classes), EBS, EFS
- Right use-cases for S3, EBS, EFS



Storage Services

S3

- Object based storage
- Gives 11 9s of reliability
- Uses global namespace
- S3 bucket is a regional resource.
- You interact via AWS APIs
- WORM usage

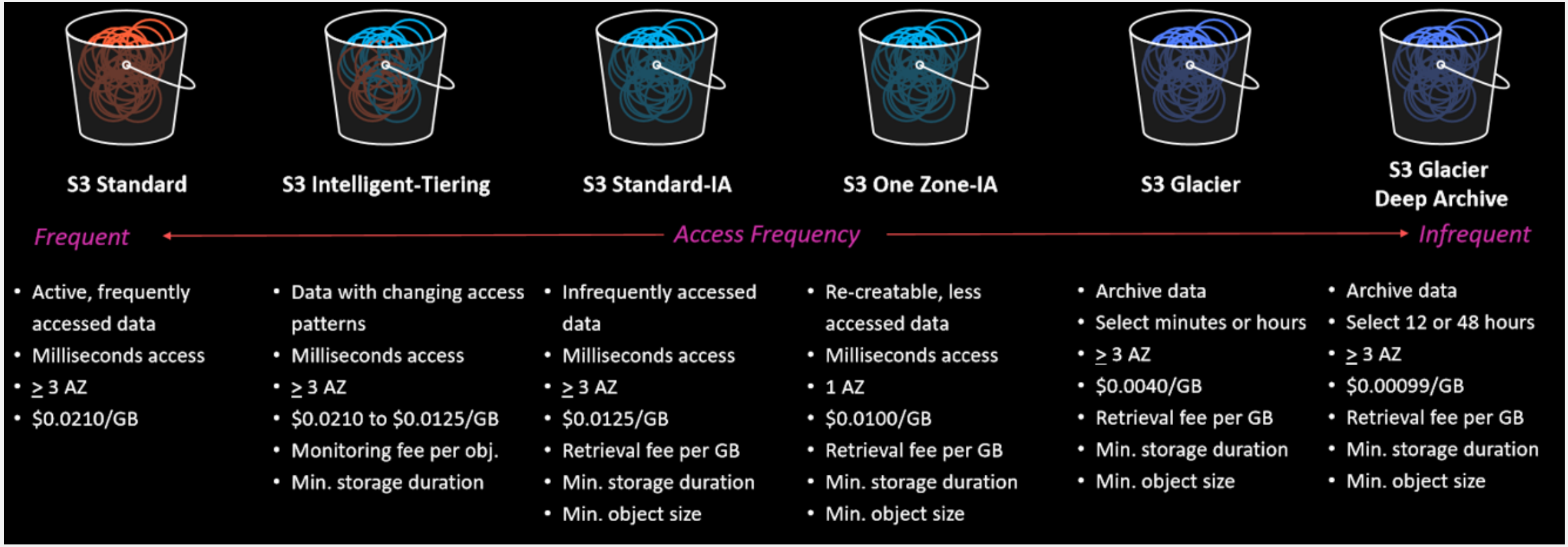
EBS

- Block based storage (disk)
- Used with EC2
- Read / write via OS level operation
- It is located in an AZ
- OS & DB usage

EFS

- It can be mounted to multiple EC2 instances
- Spans across AZs in a region
- Read/write possible from multiple instances
- Works only with Linux

S3 Storage Classes



Pricing Factors:

1. Amount of Storage
2. Data transfer (out to internet)
3. Number of requests (GET, PUT, DELETE, etc.)

Time allocated: 30 mins

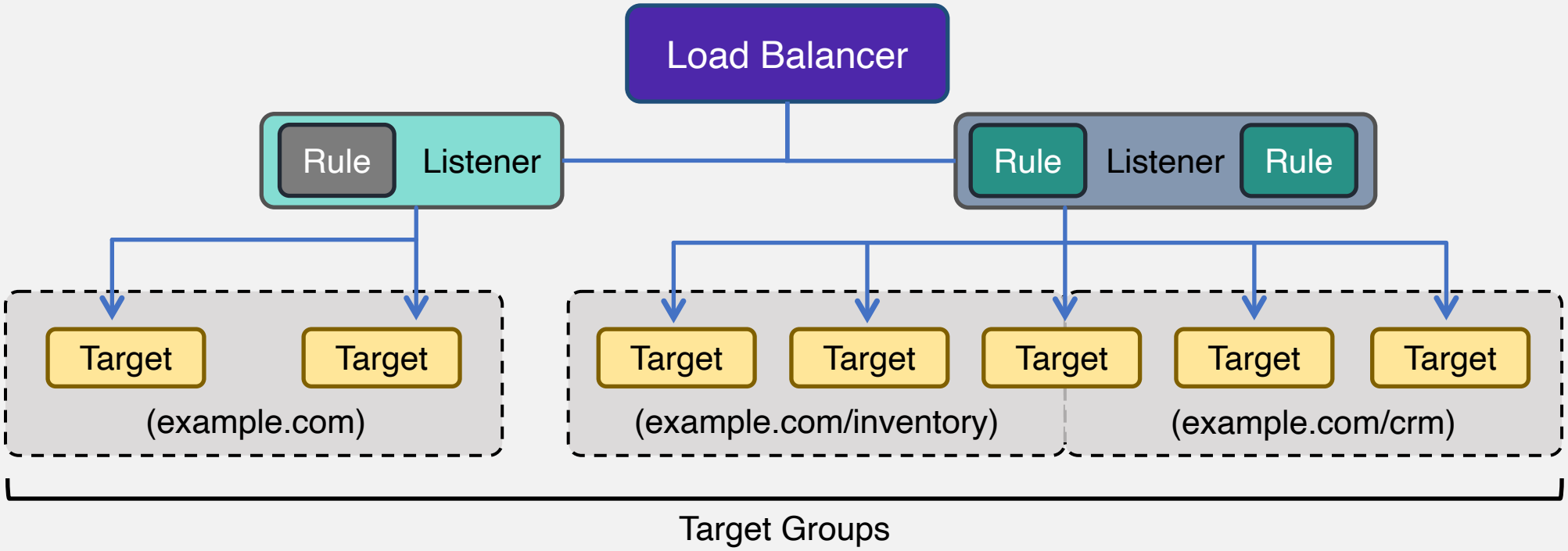
Lab 4

- Activate **CloudShell**
- Execute “aws configure” & “aws sts get-caller-identity”
- Create new buckets and execute the following commands using AWS CLI https://docs.aws.amazon.com/cli/latest/reference/s3/
- Setup Static website using S3.
- Delete all the S3 buckets at the end.

CLI Command to empty a bucket: `aws s3 rm s3://bucket001 --recursive`

Load Balancer

- A **load balancer** distributes incoming application traffic across multiple targets, such as EC2 instances, in multiple Availability Zones to increase the availability of your application.



Load Balancer features

- ✓ Automatically distributes traffic across multiple targets
- ✓ Provides high availability
- ✓ Incorporates security features
- ✓ Performs health checks

Load Balancer types

- **Application Load Balancer (HTTP and HTTPS)**
  - Flexible application management
  - Advanced load balancing of traffic
  - Operates at the request level (Layer 7)
- **Network Load Balancer (TCP & UDP)**
  - Extreme performance and static IP
  - Load balancing of TCP traffic
  - Operates at the connection level (Layer 4)

Time allocated: 25 mins discussion & 10 mins presentation

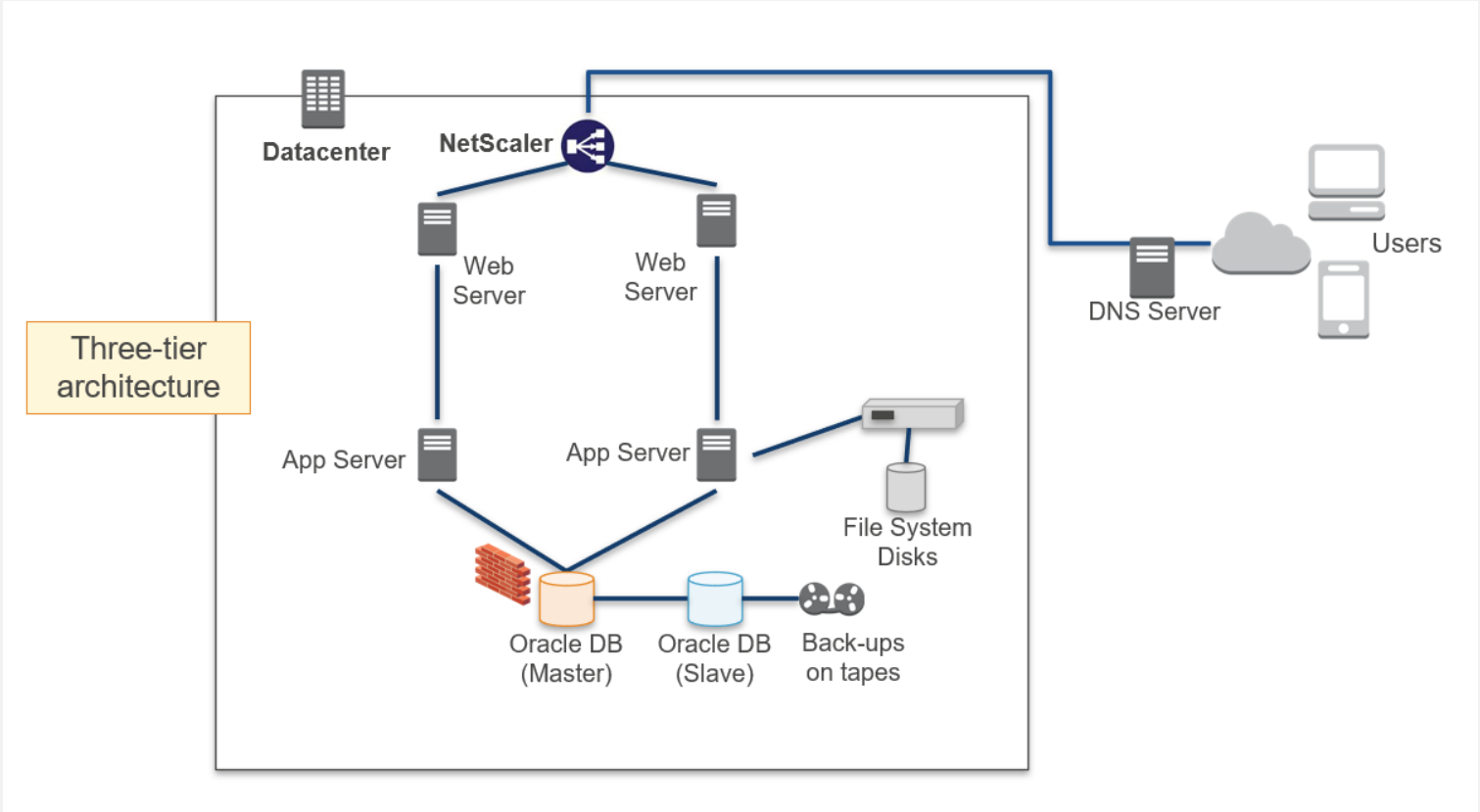
Group Activity 1

A company has acquired another small company "**myCRM**". They provide a hosted CRM tool to end users. Customers access the web application to:

- View and log customer contact information – maintaining a record of all customer information viewable on all platforms
- Upload and access customer contract documents from anywhere – capable through browser and mobile apps
- Track status of customer forms as they proceed through the sales process – view the workflow and provide feedback on next steps or delivery expectations.

- Users (consumers) connect to the myCRM application using their user ID and password (stored in the application database). Active Directory authenticates myCRM employees who need to access the system (e.g., engineers and customer support personnel). AD is present on-prem and is currently not moving to Cloud.
- They are determined to host this application at AWS as soon as possible to get a better infrastructure.
- Present a detailed architecture for this application on AWS. Please make notes on your design decisions. At this time, there is no plan to do any code change. It should be possible to move the application with minimal changes.

Group Activity 1



Route53

- Route53 is a Managed DNS (Domain Name System)
- DNS is a collection of rules and records which helps clients understand how to reach a server through URLs.
- In AWS, the most common records are:
  - A: hostname to IPv4
  - AAAA: hostname to IPv6
  - CNAME: hostname to hostname
  - Alias: hostname to AWS resource.

Route53 Routing Policies

- Simple
- Weighted
- Latency
- Failover
- Geolocation

Monitoring

- CloudWatch Metrics
- CloudWatch Events
- CloudWatch Logs
- Use of CloudWatch with other AWS services and on-premises

AWS CloudWatch Metrics

- CloudWatch provides metrics for every services in AWS
- Metric is a variable to monitor (CPUUtilization, NetworkIn, etc.)
- Metrics belong to namespaces
- Dimension is an attribute of a metric (instance id, environment, etc.)
- Metrics have timestamps
- Can create CloudWatch dashboards of metrics
- Integrated with most of the AWS services
- EC2 Standard & Detailed Monitoring
- Standard & Custom Metrics

CloudWatch Dashboards

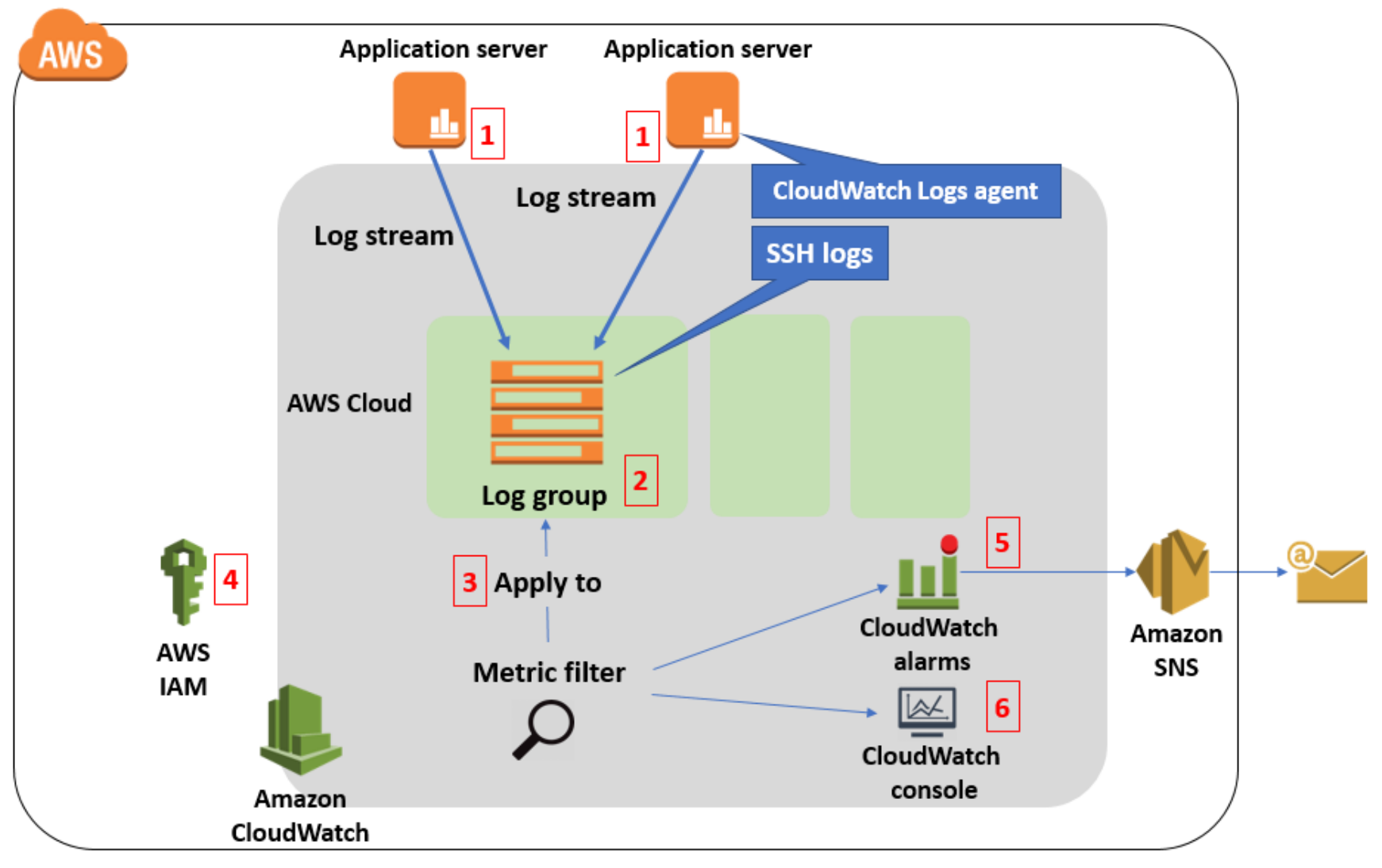
- Great way to setup custom dashboards for quick access to key metrics and alarms
- **Dashboards are global**
- **Dashboards can include graphs from different AWS accounts and regions**
- You can change the time zone & time range of the dashboards
- You can setup automatic refresh (10s, 1m, 2m, 5m, 15m)
- Dashboards can be shared with people who don't have an AWS account (public, email address, 3rd party SSO provider through Amazon Cognito)

CloudWatch Logs

- Applications can send logs to CloudWatch using the AWS SDK

- CloudWatch **can collect log from**:
  - Elastic Beanstalk: collection of logs from application
  - ECS: collection from containers
  - AWS Lambda: collection from function logs
  - VPC Flow Logs: VPC specific logs
  - API Gateway
  - CloudTrail based on filter
  - CloudWatch log agents: for example on EC2 machines
  - Route53: Log DNS queries

CloudWatch Logs



CloudWatch Events

Event Pattern: Intercept events from AWS services (Sources)

- Example sources: EC2 Instance Start, CodeBuild Failure, S3, Trusted Advisor
- Can intercept any API call with **CloudTrail integration**

Schedule or Cron (example: create an event every 4 hours)

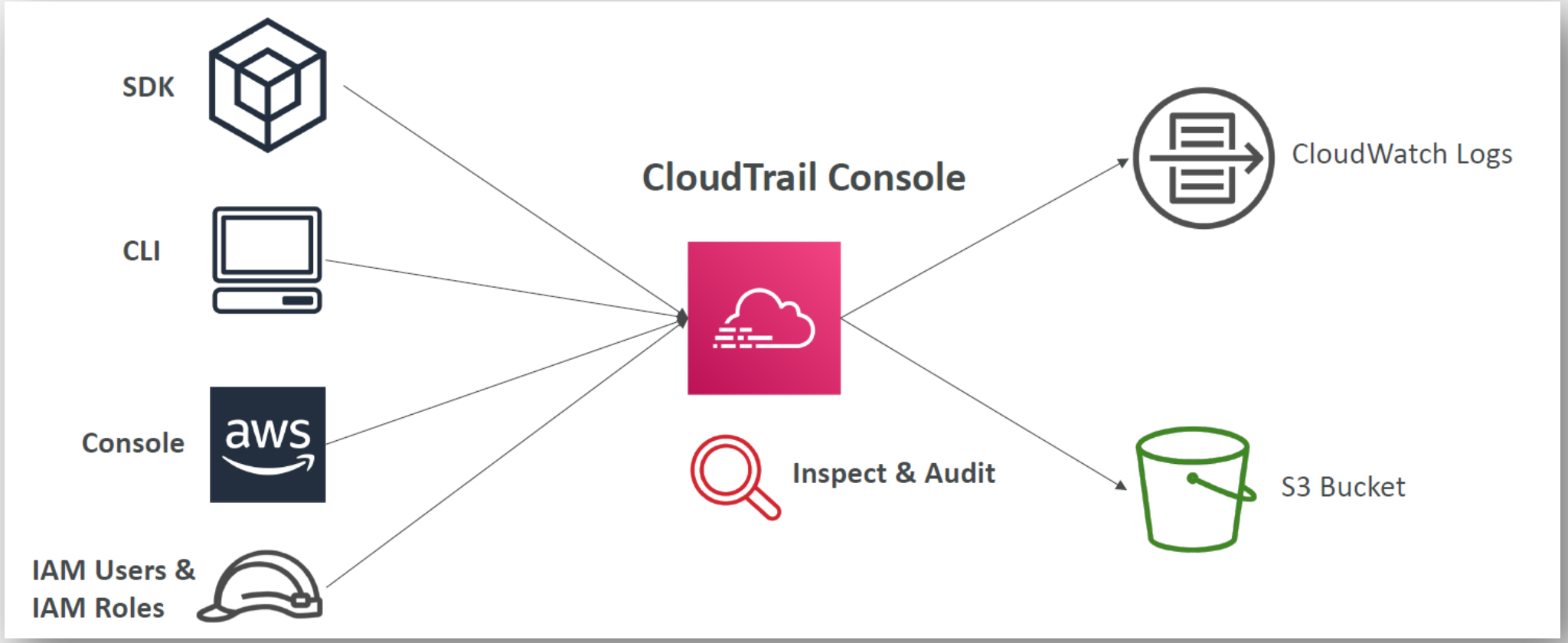
A JSON payload is created from the event and passed to a target

- Compute: Lambda, Batch, ECS task
- Integration: SQS, SNS, Kinesis Data Streams, Kinesis Data Firehose
- Orchestration: Step Functions, CodePipeline, CodeBuild
- Maintenance: SSM, EC2 Actions

AWS CloudTrail

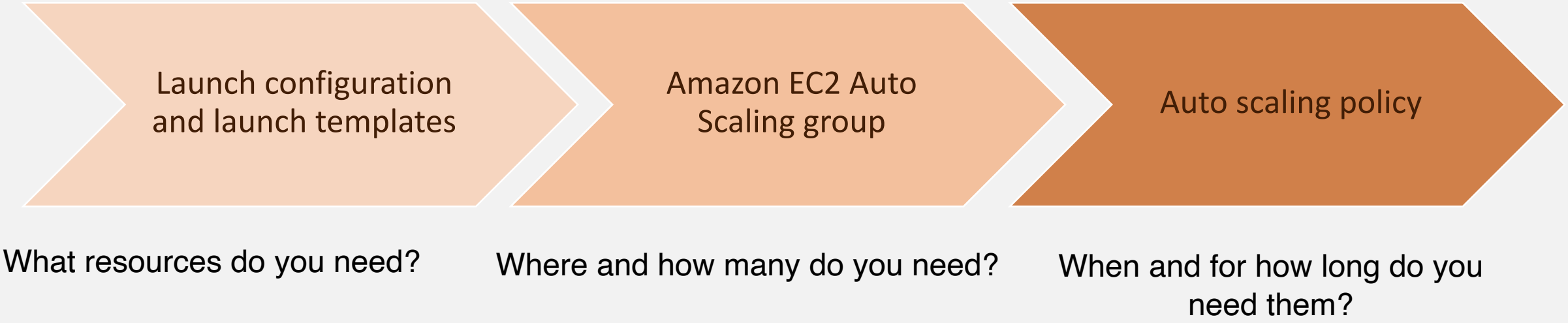
- Provides governance, compliance and audit for your AWS Account
- CloudTrail is enabled by default!
- Get an history of events / API calls made within your AWS Account by:
  - Console
  - SDK
  - CLI
  - AWS Services
- Can put logs from CloudTrail into CloudWatch Logs or S3
- A trail can be applied to All Regions (default) or a single Region.
- If a resource is deleted in AWS, investigate CloudTrail first!

AWS CloudTrail – Information Flow



EC2 Auto Scaling

- Amazon EC2 Auto Scaling helps you ensure that you have the correct number of EC2 instances available to handle the load for your application.
  - It could be a constant number of instances running all the time.
  - It could be a scale-out and scale-in requirement as well.

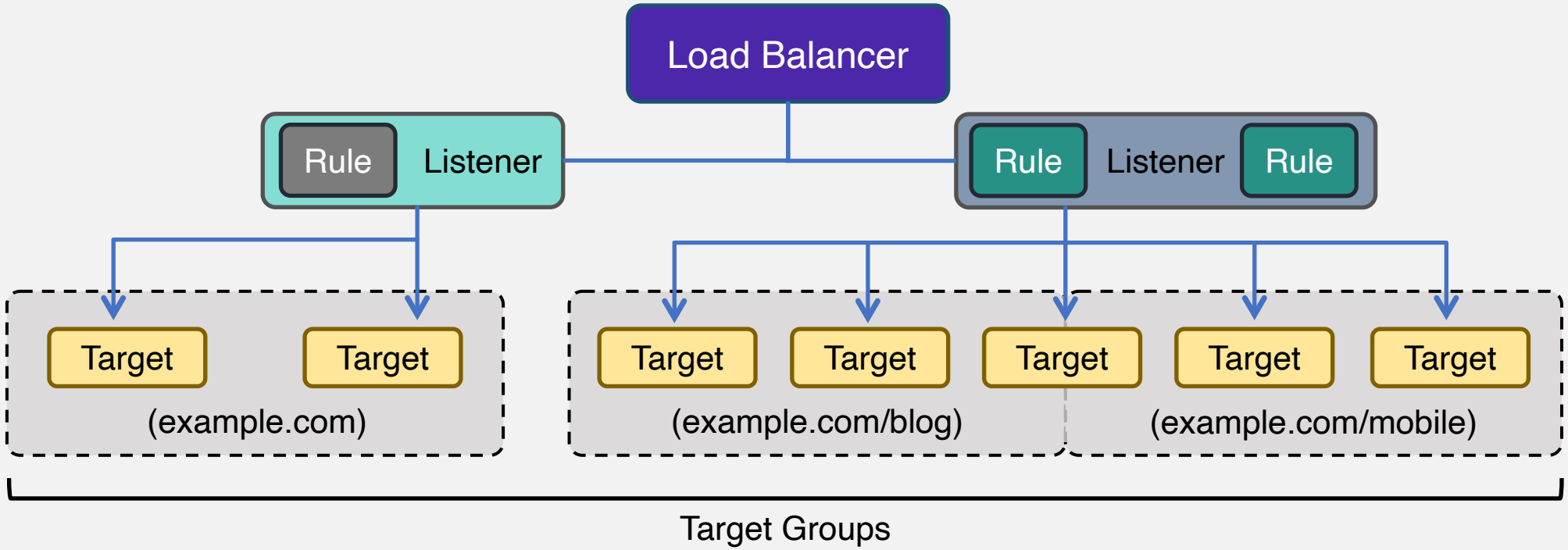


EC2 Auto Scaling – Policy types

- **Scheduled** – scaling happens on a scheduled time
- **Dynamic** – scaling happens based on a defined condition/metric value
- **Predictive** – scaling happens based on past load patterns

How to use Auto Scaling feature?

- Your application should be stateless to get full benefits of Auto scaling feature.
- Session information should be stored in an external storage (e.g. ElastiCache, DynamoDB, etc.)
- Session stickiness should not be a requirement.



Time allocated: 60 mins

Lab 5

1. Create a launch template with the specified AMI
2. Create an Auto Scaling group & Load Balancer
3. Test your application
4. Test high availability of your application tier by terminating an instance
5. Load the application and verify scale-out and scale-in

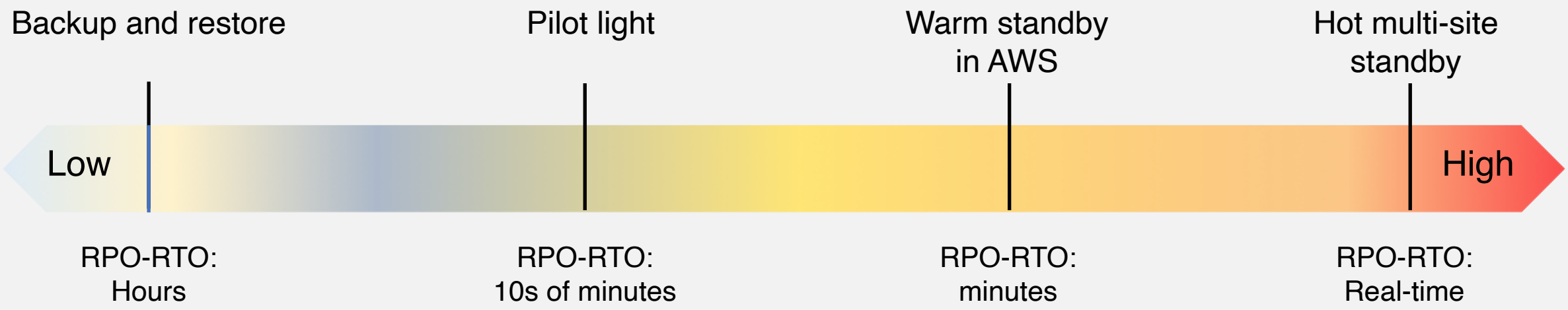
Availability concepts

- **High availability** minimizes downtime and cost. It protects against failure. It keeps your business running with very little downtime and fast recovery, at low cost.
- **Fault tolerance** is often confused with high availability, but fault tolerance refers to the built-in redundancy of an application's components to ensure that there is no service interruption, but at a higher cost.
- **Backup** is critical to protect data and to ensure business continuity. At the same time, it can be a challenge to implement well. The pace at which data is generated is growing exponentially.
- **Disaster recovery** is often overlooked as a component of availability. If a natural disaster makes one or more of your components unavailable or destroys your primary data source, can you restore service quickly and without lost data?
- **RTO** – How long can the application be unavailable?
- **RPO** – How often does data need to be backed up?

DR Strategies

- **Backup & restore** - take backups off current systems and store them in Amazon S3. Know which AMI to use or build your own as needed. Know how to restore systems from backups, how to switch to a new system, and how to configure the deployment. When reacting to a disaster, retrieve backups from Amazon S3 and bring up your required infrastructure. Use CloudFormation to automate deployment of core networking. Next, restore system from backup, switch over to the new system, and adjust DNS records to point to AWS.
- **Pilot light** – in this approach, you replicate your data from one Region to another and provision a copy of your core workload infrastructure. Resources required to support data replication and backup, such as databases and object storage, are always on. The pilot light architecture is relatively inexpensive to implement.
- **Fully working low-capacity standby** - is an elevated version of the pilot light. The warm standby approach involves ensuring that there is a scaled down, but fully functional, copy of your production environment in another Region. This approach extends the pilot light concept and decreases the time to recovery because your workload is always on in another Region.
- **Multi-site active-active** – this approach will have full infrastructure in your DR region as well as on your existing region. The data replication method that you employ will be determined by the recovery point that you choose. This architecture potentially has the least downtime of all.

DR Strategies - comparison



- Lowest priority use cases
- Solutions: Amazon S3, Storage Gateway

$

- Lower RTO and RPO requirements
- Core services
- Scale AWS resources in response to a DR event

$$

- Higher priority solutions require RTO and RPO in minutes
- Business-critical services

$$$

- Highest priority
- Auto-failover of your environment in AWS to a running duplicate

$$$$

Group Activity 2

Time allocated: 60 mins discussion & 15 mins presentation

Your client has recently experienced tremendous sales growth and must support significant amounts of new users.

This means:

- Significant increase in users simultaneously accessing the system.
- Increased number of customer interactions to manage.
- Increased contracts to create and track.
- Increased storage needs.

Due to this volume increase, you have been requested to **redesign** an entire solution. The CTO has given you the following requirements:

1. Reliability

a) Deployment is required in 2 regions: US East Coast and Southeast Asia (only one site active at a time)
b) Must be able to remain fully functional during any component failure

2. Security

a) User authentication for web browser clients (will be based on app, uses credentials stored in app DB)
b) Minimal exposure for any items in the public subnet
c) Security for Data at Rest and Data in Transit

Group Activity 2

3. Performance Efficiency

a) Three-tier architecture with Web, App, and DB tiers

b) Web and App tiers should be loosely coupled and scalable

c) Web-tier should be stateless

d) DB tier runs relational database

e) Session information should be durably stored in a fault tolerant way

f) Must scale to handle the increasing workload

g) Solution should be designed to deliver greatest performance for the end user

4. Cost Minimization

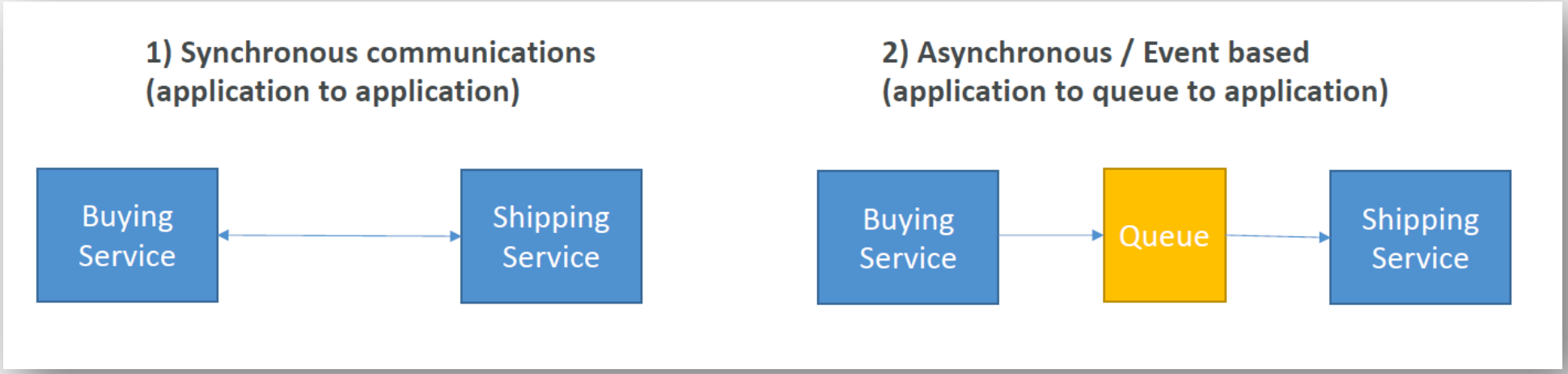
a) Make architectural decisions which have justifiable costs.

b) Cost Optimization includes both tangible and intangible costs

NOTE: It is a **complete re-architecting** so you can adopt all the best practices & recommendations, but you should back your choices against the requirements given.

Decoupled Systems

- When we start deploying multiple applications, they will inevitably need to communicate with one another
- There are two patterns of application communication



Simple Queue Service (SQS)

- Fully managed service, used to decouple applications
- Unlimited throughput, unlimited number of messages in queue
- Default retention of messages: 4 days, maximum of 14 days
- Low latency (<10ms on publish and receive)
- Limitation of 256KB per message sent
- Can have duplicate messages (at least once delivery, occasionally)
- Can have out of order messages (best effort ordering)

SQS – Producing Messages

- Produced to SQS using the AWS SDK
- The message is persisted in SQS until a consumer deletes it
- Message retention: default 4 days, up to 14 days

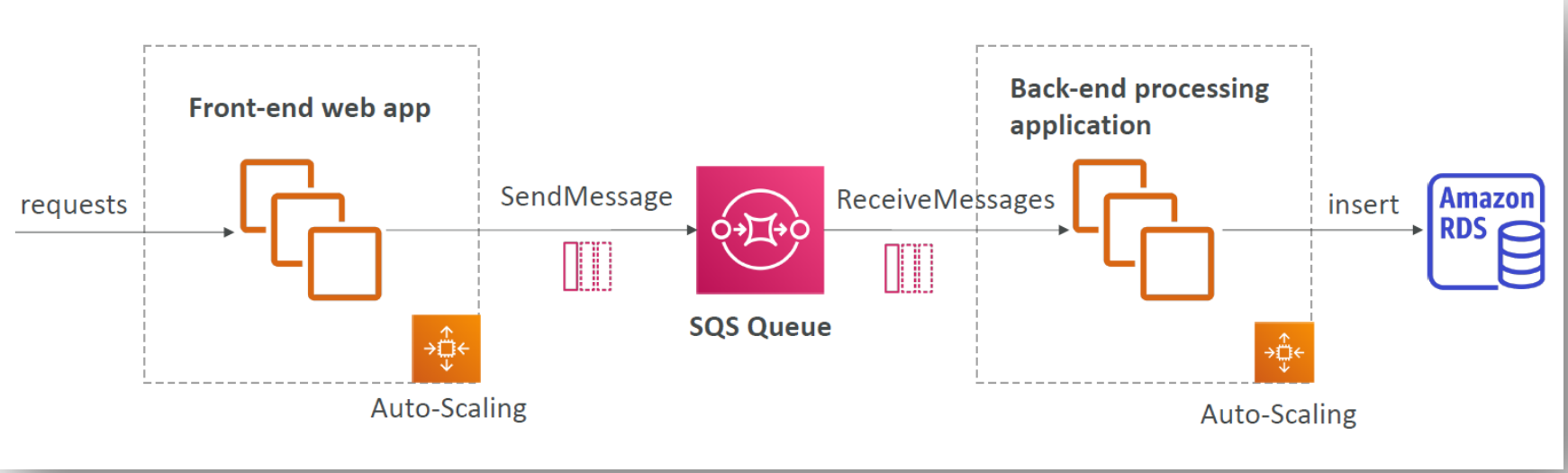
Example: send an order to be processed

- Order id
- Customer id
- Any attributes you want

SQS standard: unlimited throughput

SQS – Consuming Messages

- Consumers (running on EC2 instances, servers, or AWS Lambda)…
- Poll SQS for messages (receive up to 10 messages at a time)
- Process the messages (example: insert the message into an RDS database)
- Delete the messages using the **AWS** API



SQS – Security

Encryption:

- In-flight encryption using HTTPS API
- At-rest encryption using KMS keys
- Client-side encryption if the client wants to perform encryption/decryption itself

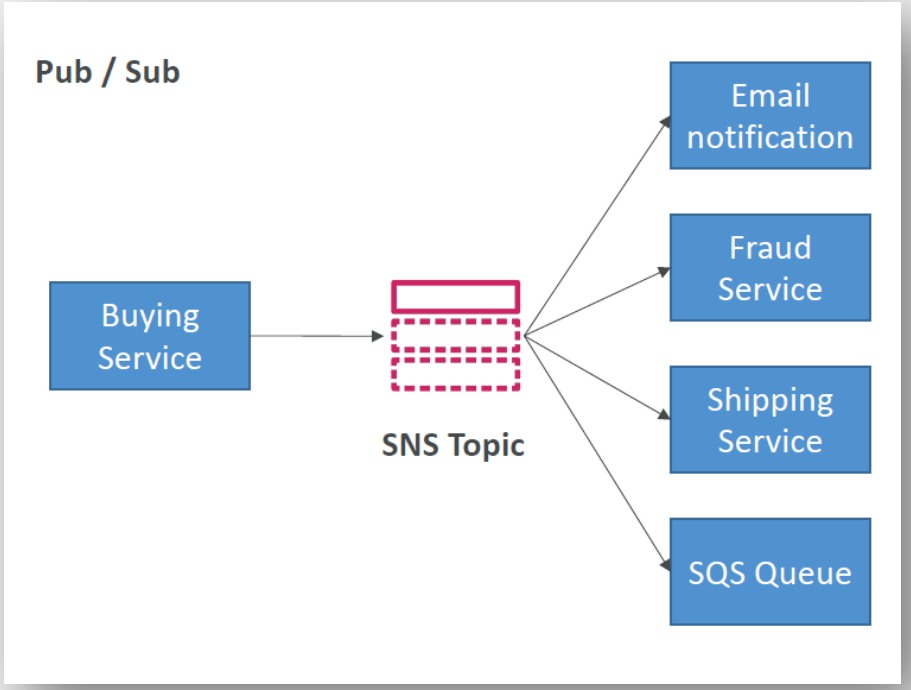
Access Controls: IAM policies to regulate access to the SQS API

SQS Access Policies (similar to S3 bucket policies)

- Useful for cross-account access to SQS queues
- Useful for allowing other services (SNS, S3, etc.) to write to an SQS queue

Simple Notification Service (SNS)

- The “event producer” only sends message to one SNS topic
- As many “event receivers” (subscriptions) as we want to listen to the SNS topic notifications
- Each subscriber to the topic will get all the messages
- Up to 10,000,000 subscriptions per topic
- 100,000 topics limit
- Subscribers can be:
  - SQS
  - HTTP / HTTPS
  - Lambda
  - Emails
  - SMS messages
  - Mobile Notifications



SNS Integration (examples)

- Many AWS services can send data directly to SNS for notifications
- CloudWatch (for alarms)
- Auto Scaling Groups notifications
- Amazon S3 (on bucket events)
- CloudFormation (upon state changes => failed to build, etc.)

SNS – How to publish

- Topic Publish (using the SDK)
  - Create a topic
  - Create a subscription (or many)
  - Publish to the topic
- Direct Publish (for mobile apps SDK)
  - Create a platform application
  - Create a platform endpoint
  - Publish to the platform endpoint

SNS - Security

Encryption:

- In-flight encryption using HTTPS API
- At-rest encryption using KMS keys
- Client-side encryption if the client wants to perform encryption/decryption itself

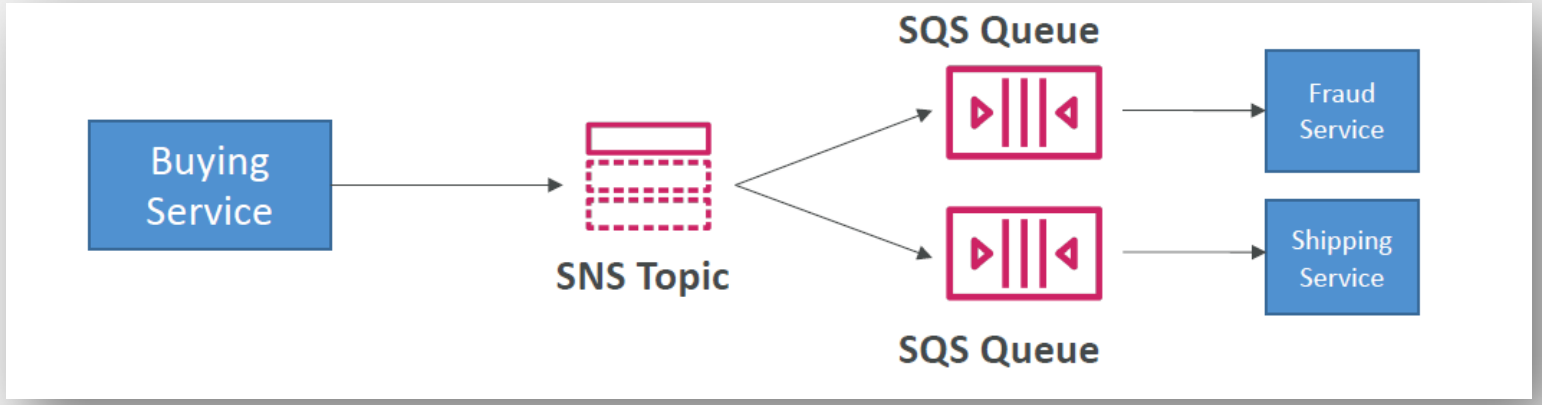
Access Controls: IAM policies to regulate access to the SNS API

SNS Access Policies (similar to S3 bucket policies)

- Useful for cross-account access to SNS topics
- Useful for allowing other services (S3 etc.) to write to an SNS topic

SNS + SQS (Fan Out)

- Push once in SNS, receive in all SQS queues that are subscribers
- Fully decoupled, no data loss
- SQS allows for: data persistence, delayed processing and retries of work
- Ability to add more SQS subscribers over time
- Make sure your SQS queue access policy allows for SNS to write



SERVERLESS

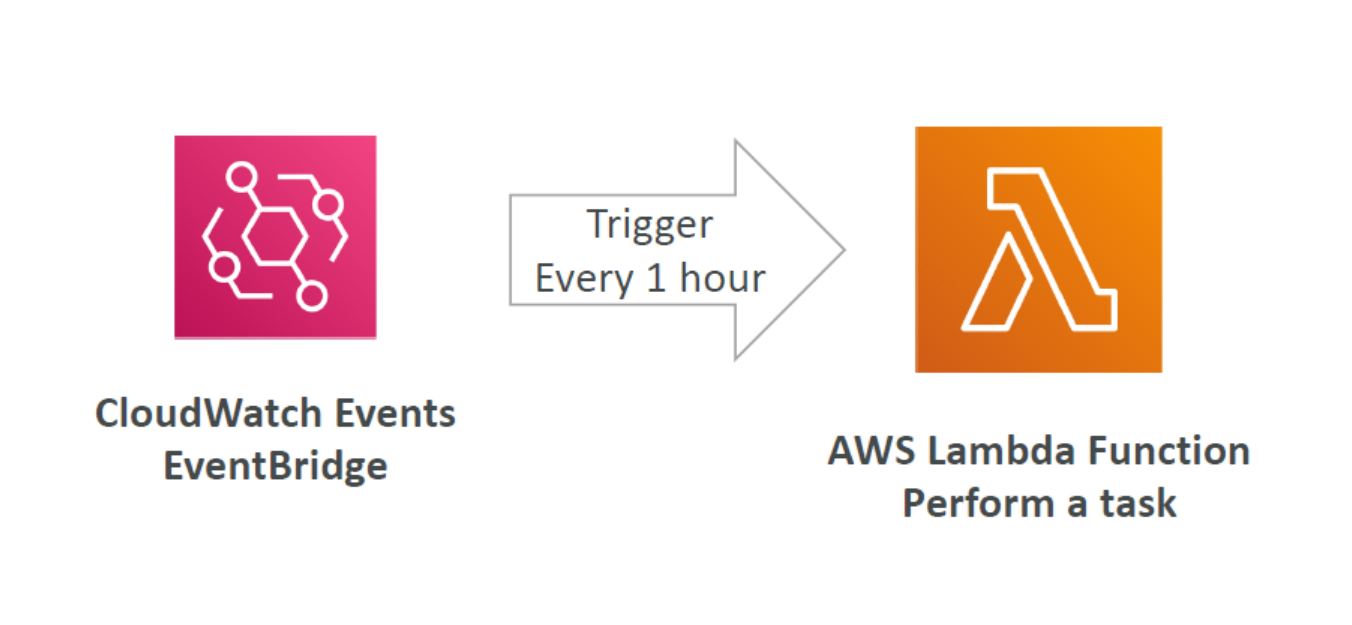
Serverless approach

- Serverless is a new paradigm in which the developers don’t have to manage servers anymore
- They just deploy code in the form of functions
- Serverless **does not mean there are no servers**, it means you just don’t manage / provision / see them

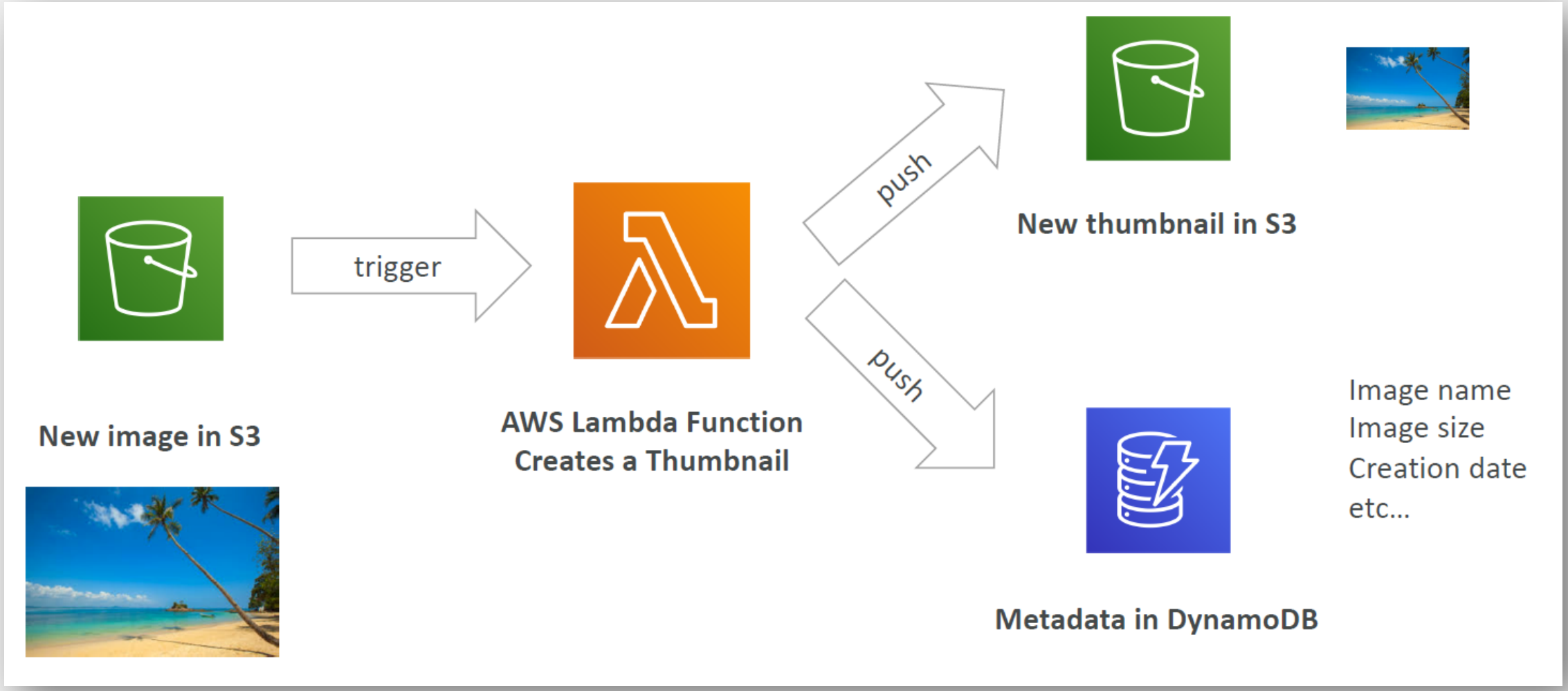
AWS Lambda Overview

- No servers to maintain - serverless
- Supports code in many languages - C#, Java, Python, node.js, Go, etc.
- Offers many levels of resources:
  - 128 MB of memory and the lowest CPU power, to
  - 10 GB of memory and the highest CPU power
- A function can run up to 15 minutes when invoked
- Pricing depends on
  - resource level chosen
  - **actual** time of execution (rounded off to nearest 1ms)
  - number of invocation requests
- Saves money as compute is used **only when required** (no resources are blocked/running all the time)
- Supports IAM role & VPC connectivity

Example: Serverless CRON Job



Example: Serverless Thumbnail creation



AWS Secrets Manager

- A service meant for storing secrets
- Capability to force **rotation of secrets** every X days
- Automate generation of secrets on rotation (uses Lambda – custom code to interact with the respective service)
- Integration with Amazon RDS (MySQL, PostgreSQL, Aurora)
- Secrets are encrypted using KMS
- Mostly meant for RDS integration

AWS Key Management Service (KMS)

- AWS KMS makes it easy for you to create and manage cryptographic keys and control their use across a wide range of AWS services and in your applications.

- AWS KMS is integrated with AWS CloudTrail to provide you with logs of all key usage to help meet your regulatory and compliance needs.

- Three types of Customer Master Keys (CMK):
  - AWS Managed Service Default CMK: free
  - User Keys created in KMS: $1 / month
  - User Keys imported (must be 256-bit symmetric key): $1 / month

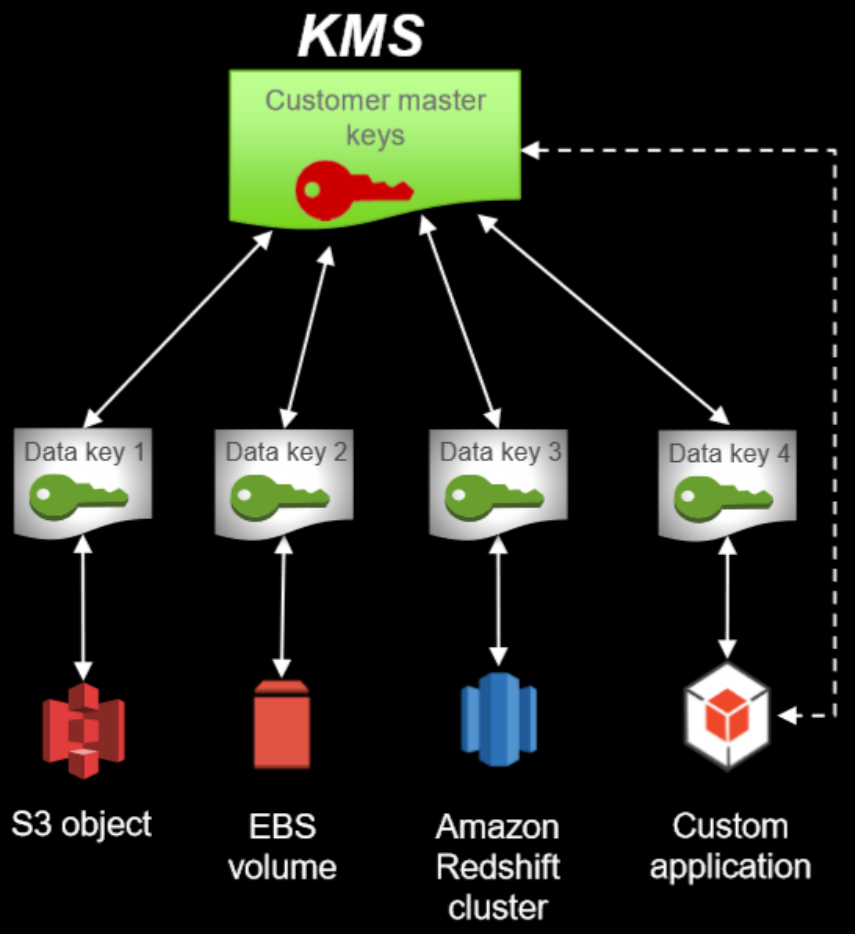
Client Integration with KMS

Two-tiered key hierarchy using envelope encryption

- Unique data key encrypts customer data
- KMS master keys encrypt data keys

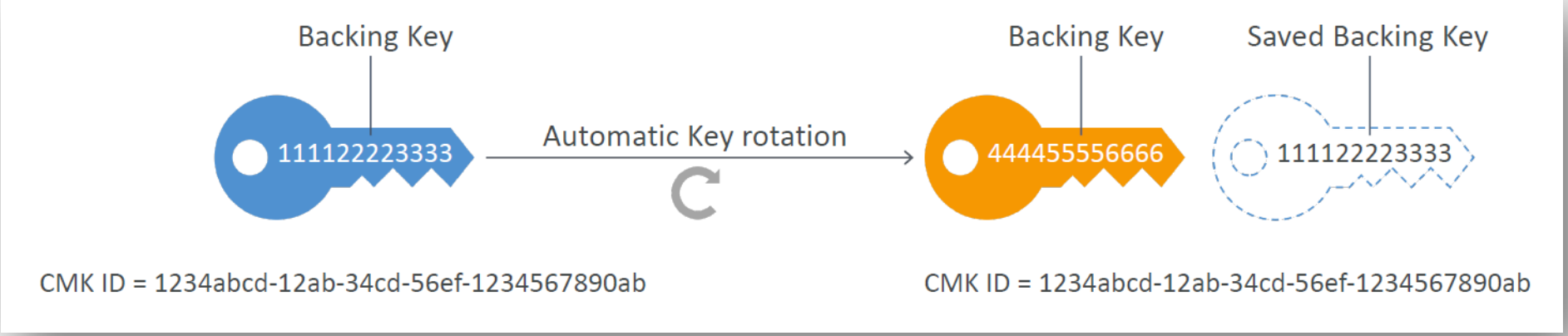
Benefits

- Limits risk of compromised data key
- Better performance for encrypting large data
- Easier to manage small number of master keys than millions of data keys
- Centralized access and audit of key activity



KMS Automatic Key Rotation

- For Customer-managed CMK (not AWS managed CMK)
- If enabled: automatic key rotation happens every 1 year
- Previous key is kept active so you can decrypt old data
- New Key has the same CMK ID (only the backing key is changed)



Infrastructure as Code

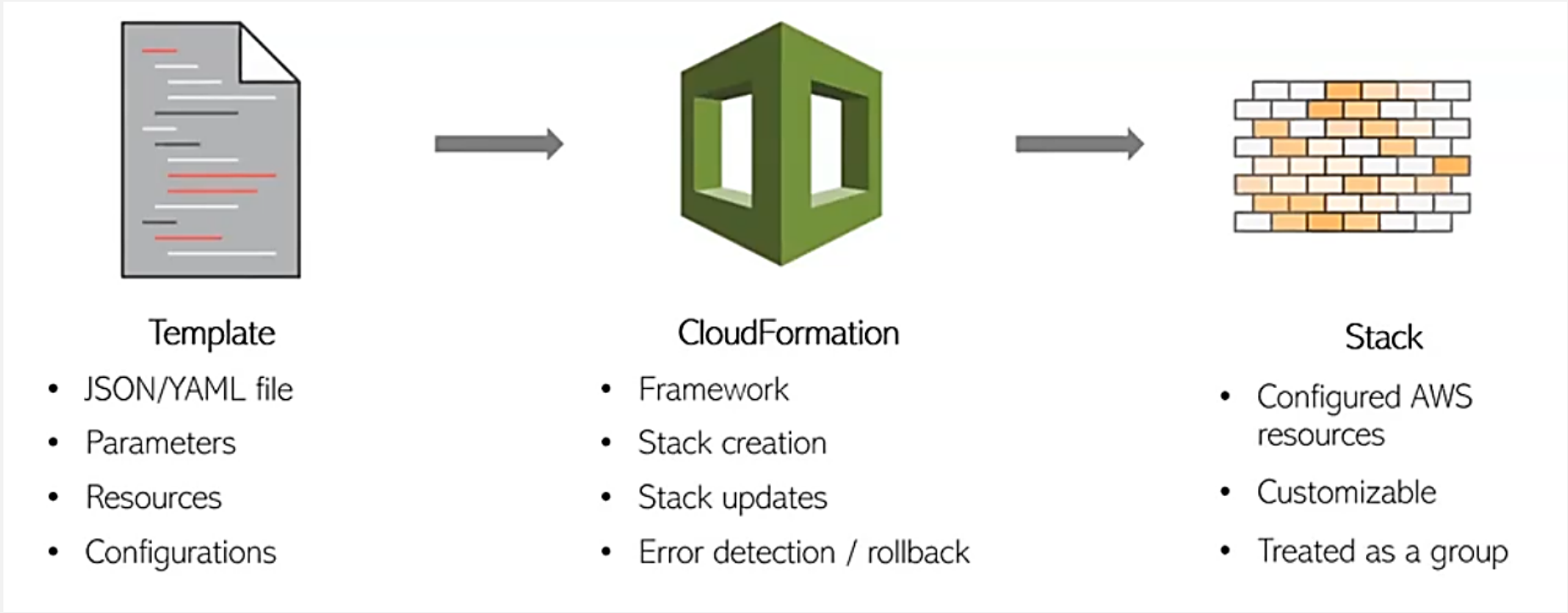
Infrastructure as Code

- IaC is a practice in which traditional infrastructure management techniques are **supplemented by or replaced** with **code-based** tools and **software development** techniques.
- Cloud gives you the option to interact with infrastructure via code (simple API calls).
- What you want >> Resource
- The way you want to customize it >> Attributes

How could it help my Organization

- Save time & bring standards (reduce manual intervention)
- Version controlled and hence ability to go back
- Foundation step to enable self-service in your organization

The Flow


Template
JSON/YAML file
Parameters
Resources
Configurations
CloudFormation
Framework
Stack creation
Stack updates
Error detection / rollback
Stack
Configured AWS resources
Customizable
Treated as a group

Time allocated: 40 mins

Lab 6

- AWS resource deployment using Terraform Code.
- Configure terraform in the Public EC2 instance and execute the steps.
- Verify the created resources in AWS Management Console.