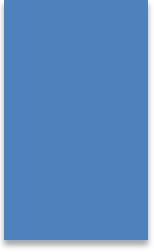


Architecting on AWS



Learn the Art of
Cloud Computing

Agenda

- Benefits of Public Cloud
- What is Cloud Computing
- Cloud Providers
- Introduction to AWS
- AWS Compute Service – EC2.

Benefits of Public Cloud

Capital Expenses

translates to

Operational Costs

Benefits of Public Cloud

Lower Costs

Massive Economies of Scale

Benefits of Public Cloud

**Focus on Your Core Business
Rather than Building Data Centres**

Benefits of Public Cloud

**Uniform Platform
with Global Footprint**

Benefits of Public Cloud

**Speed and Agility
using Automation**

What is Cloud Computing?

- Delivery of IT over the internet.
 - **IAAS** – Delivery of raw infrastructure resources
 - **PAAS** – Delivery of Platforms
 - **SAAS** – Delivery of Software

Characteristics of a Cloud Platform

1. On-demand Self-Service

Characteristics of a Cloud Platform

2. Rapid Elasticity

Characteristics of a Cloud Platform

4. Metered Services

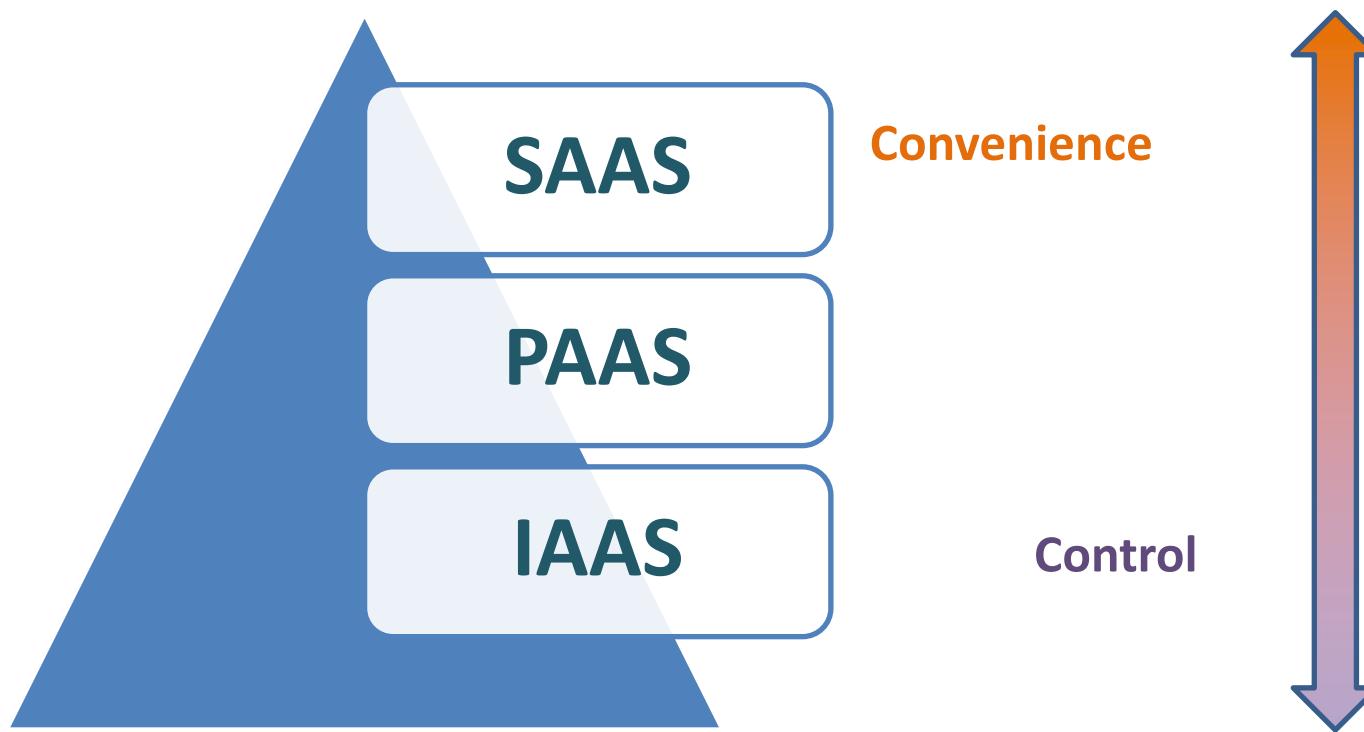
Characteristics of a Cloud Platform

4. Resource Pooling

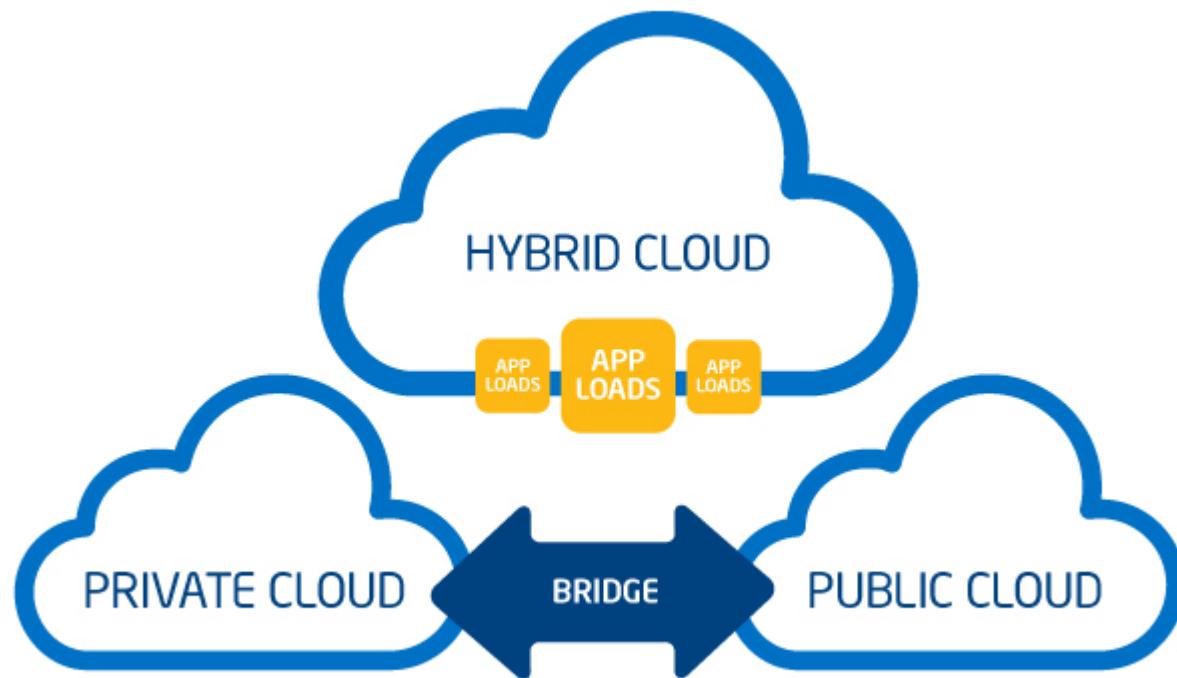
Characteristics of a Cloud Platform

5. Broad Network Access

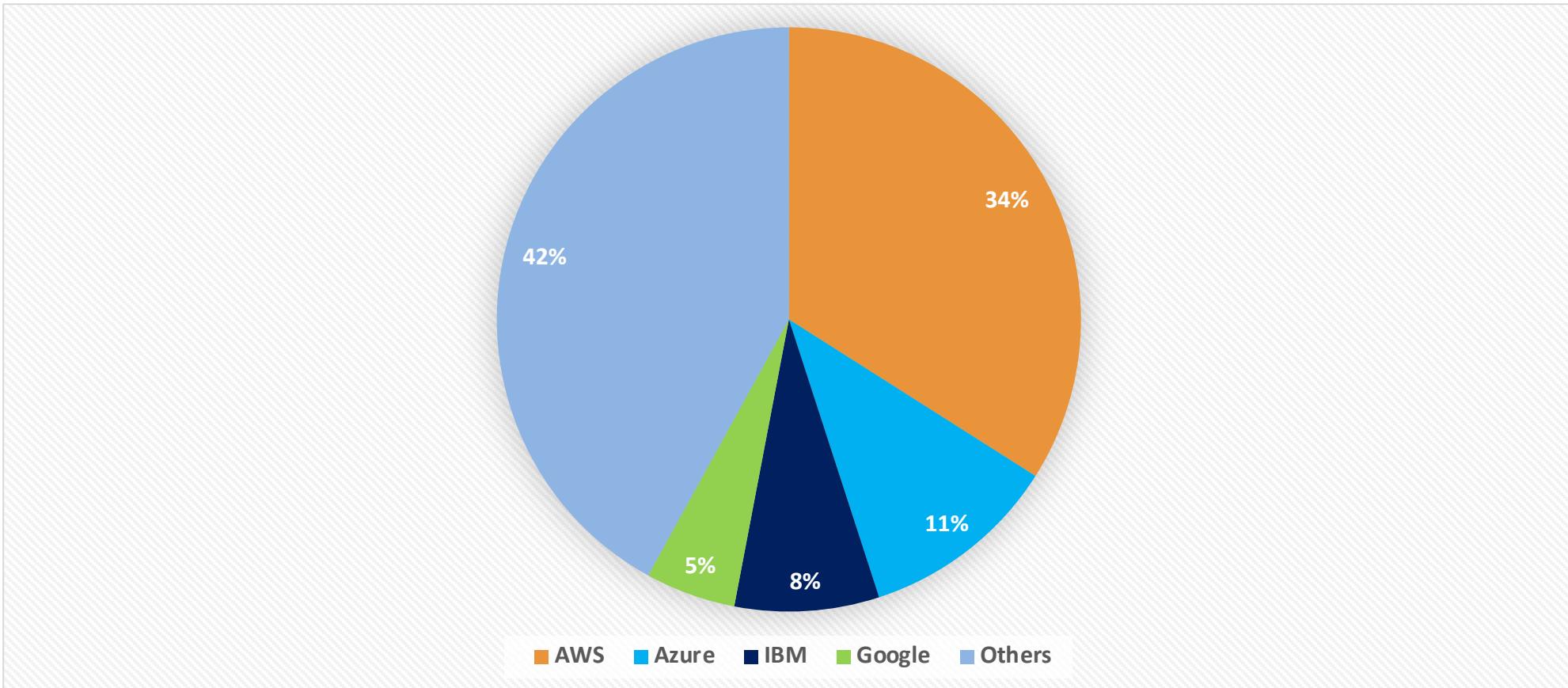
Cloud Service Models



Cloud Deployment Models



Public Cloud Providers



Who is Using Cloud?



airbnb



CleverTap

hulu

slack



P&G

ING



Challenges in Cloud Adoption





Introduction of AWS

Agenda

- Introduction and History of AWS
- Traditional Infra v/s AWS Services
- AWS Data Centre Design
- Q/A

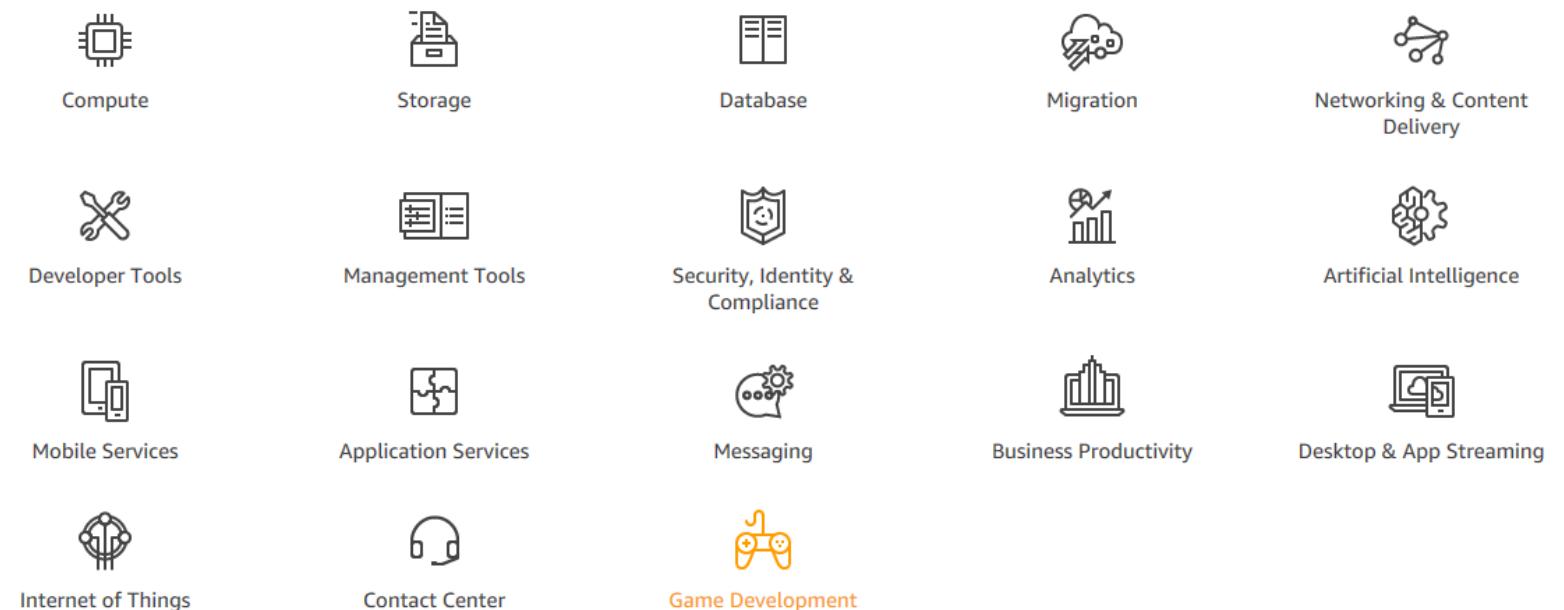
History

- AWS was launched in 2006 with a vision to enable businesses and developers to be able to build highly scalable and sophisticated web scale applications.



Features and Services

- 92+ Services
- 2000+ Features



Service Classification

Managed Services

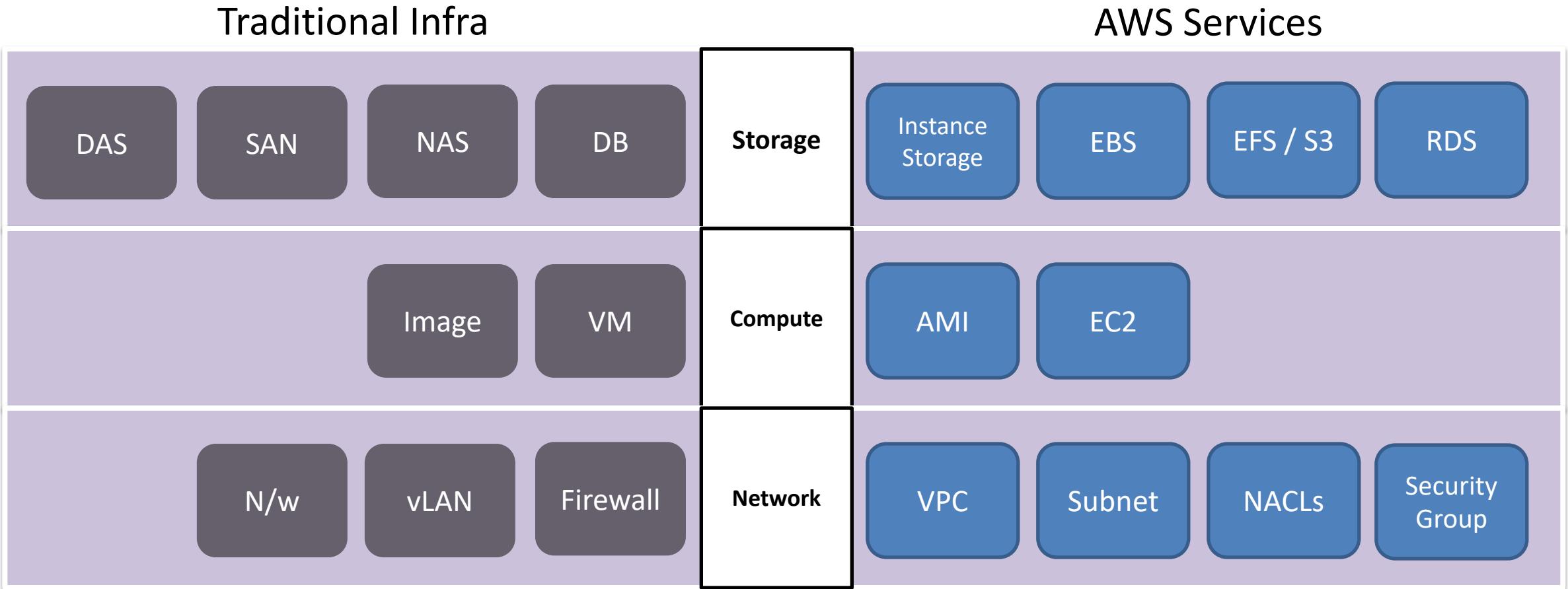
S3, RDS, DynamoDB, CloudFront...

v/s

Unmanaged Services

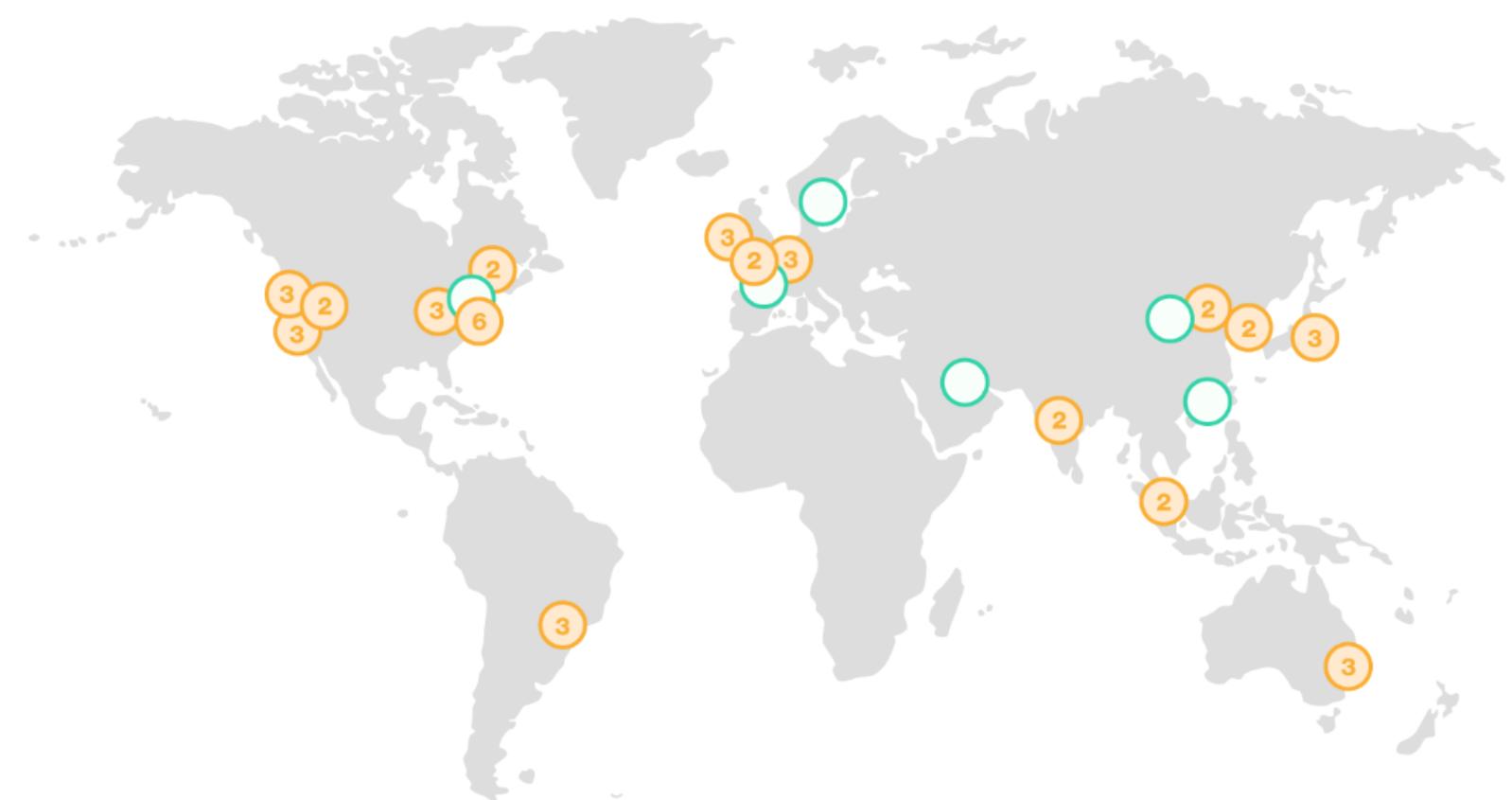
EC2, EBS, RedShift...

Traditional Infra v/s AWS Services



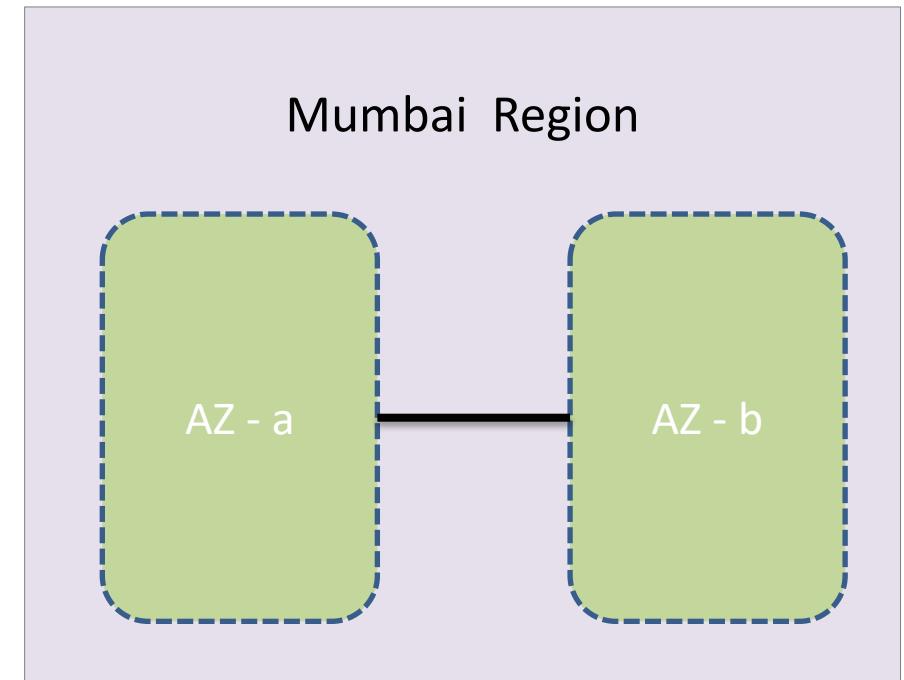
AWS Global Infrastructure

- 16 Regions (+5)
- 46 Availability Zones-AZ
- 100+ Edge Locations



Regions and AZs

- Region is geographical location of presence.
- A region must have at least 2 AZs - located 10 to 100 miles apart from each other - mainly for fault isolation.
- AZ within a region is equivalent to an independent data centre.
- AZs are connected with Dark Fibres for low latency (~5-10 milliseconds).



Edge Location

- It's an extended Infrastructure – a collection of 100+ edge cache locations that is primarily used for the CDN service called as CloudFront.
- It is also used to host Route53 and other edge services like Lambda.



AWS Account Security Using IAM.

Topics

- What is AWS IAM
- IAM Users
- IAM User Credentials
- IAM Policies
- IAM Groups
- IAM Roles
- Best Practices.

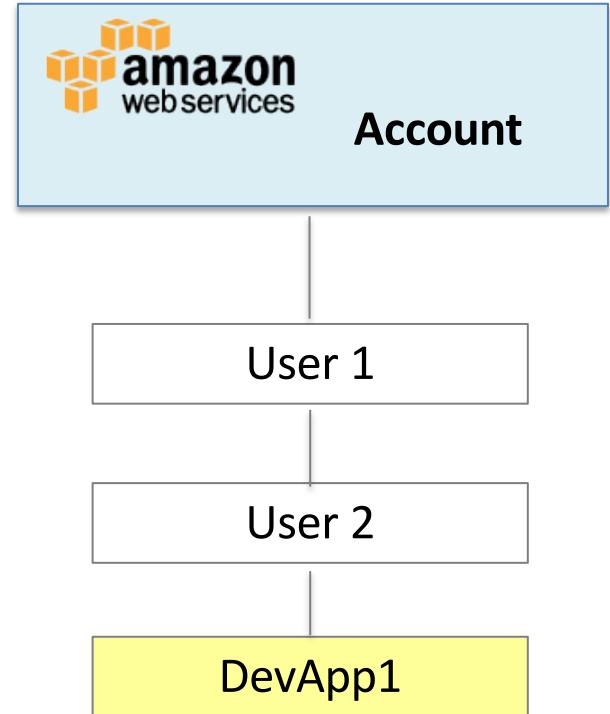


AWS Identity and Access Management

- Centrally **manage access and authentication** of your users to your AWS resources.
 - Offered as a feature of your AWS account for no charge.
 - Create **users**, **groups**, and **roles**, and apply **policies** to them to control their access to AWS resources.
 - Manage what resources can be accessed and how they can be accessed (e.g., terminating EC2 instances).
 - Define required credentials based on context (e.g., **who** is accessing **which service** and **what** are they trying to do?).

IAM Users

- An entity you create in AWS.
- Provides a way to interact with AWS.
- No default security credentials for IAM users.
 - You have to assign them specifically.
- IAM users are not necessarily people.
- **Best practice:** Create a separate IAM user account with administrative privileges even for the root account user.



Types of Security Credentials

Email address and password

Associated with your AWS account (root)

IAM user name and password

Used for accessing the AWS Management Console

Access keys

Typically used with CLI and programmatic requests like APIs and SDKs

Multi-Factor Authentication

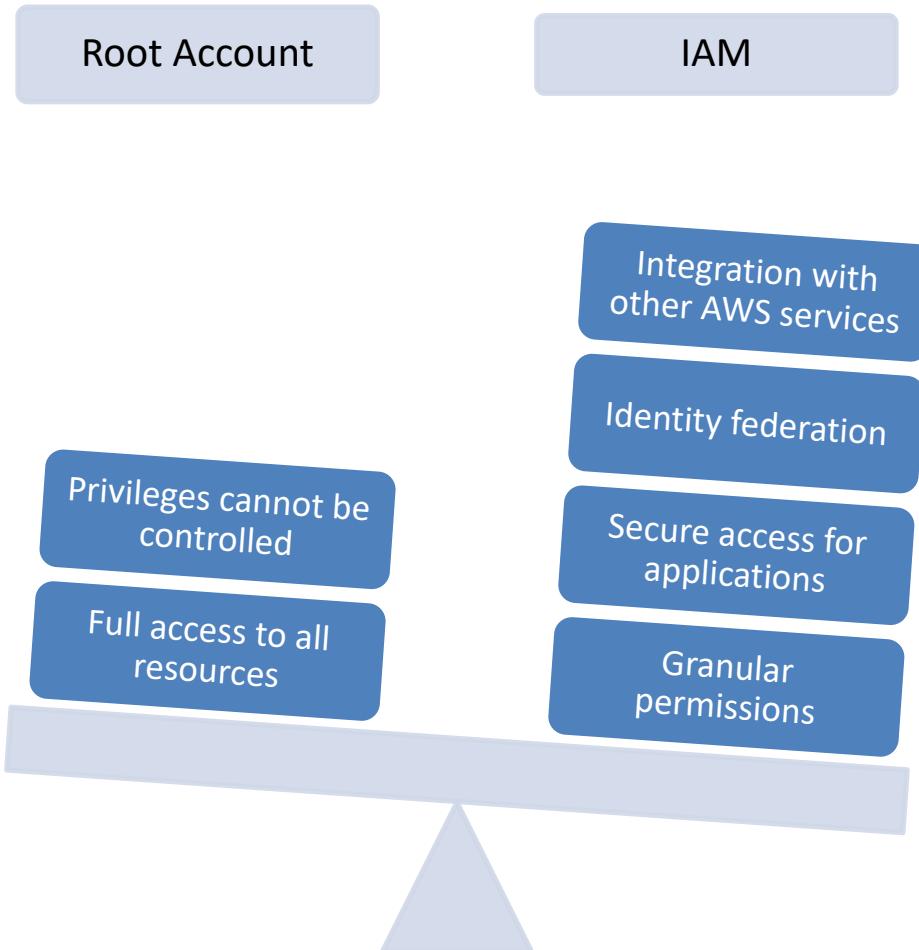
Extra layer of security

Can be enabled for root account and IAM users

Key pairs

Used only for specific AWS services like Amazon EC2

Root Account Access vs. IAM Access



IAM allows you to follow the
least privilege principle.

IAM Permissions

Permissions determine which resources and which operations are allowed to be used.

- All permissions are *implicitly* denied by default.
- If something is *explicitly* denied, it can never be allowed.
- **Best Practice:** Follow the **least privilege** principle.

IAM Policy Example

```
{  
    "Version": "2012-10-17",  
    "Statement": [{  
        "Effect": "Allow",  
        "Action": ["dynamodb:*", "s3:*"],  
        "Resource": ["arn:aws:dynamodb:region:account-number-without-hyphens:table/table-name",  
                    "arn:aws:s3:::bucket-name",  
                    "arn:aws:s3:::bucket-name/*"]  
    },  
    {  
        "Effect": "Deny",  
        "Action": ["dynamodb:*", "s3:*"],  
        "NotResource": ["arn:aws:dynamodb:re  
                        "arn:aws:s3:::bucket-name",  
                        "arn:aws:s3:::bucket-name/*"]  
    }]  
}
```

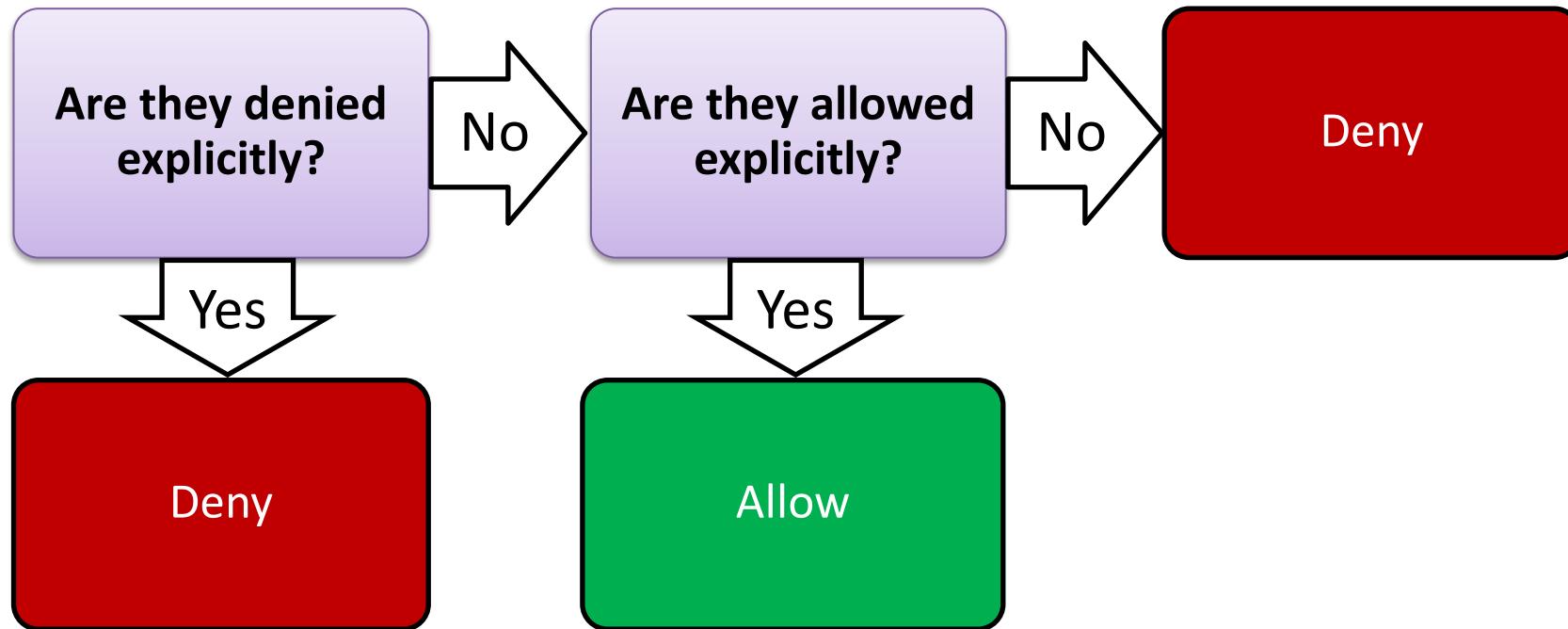
Gives users access to a specific DynamoDB table and...

...Amazon S3 buckets

Explicit deny ensures that the users cannot use any other AWS actions or resources other than that table and those buckets

An explicit deny statement **takes precedence** over an allow statement

How IAM Determines Permissions



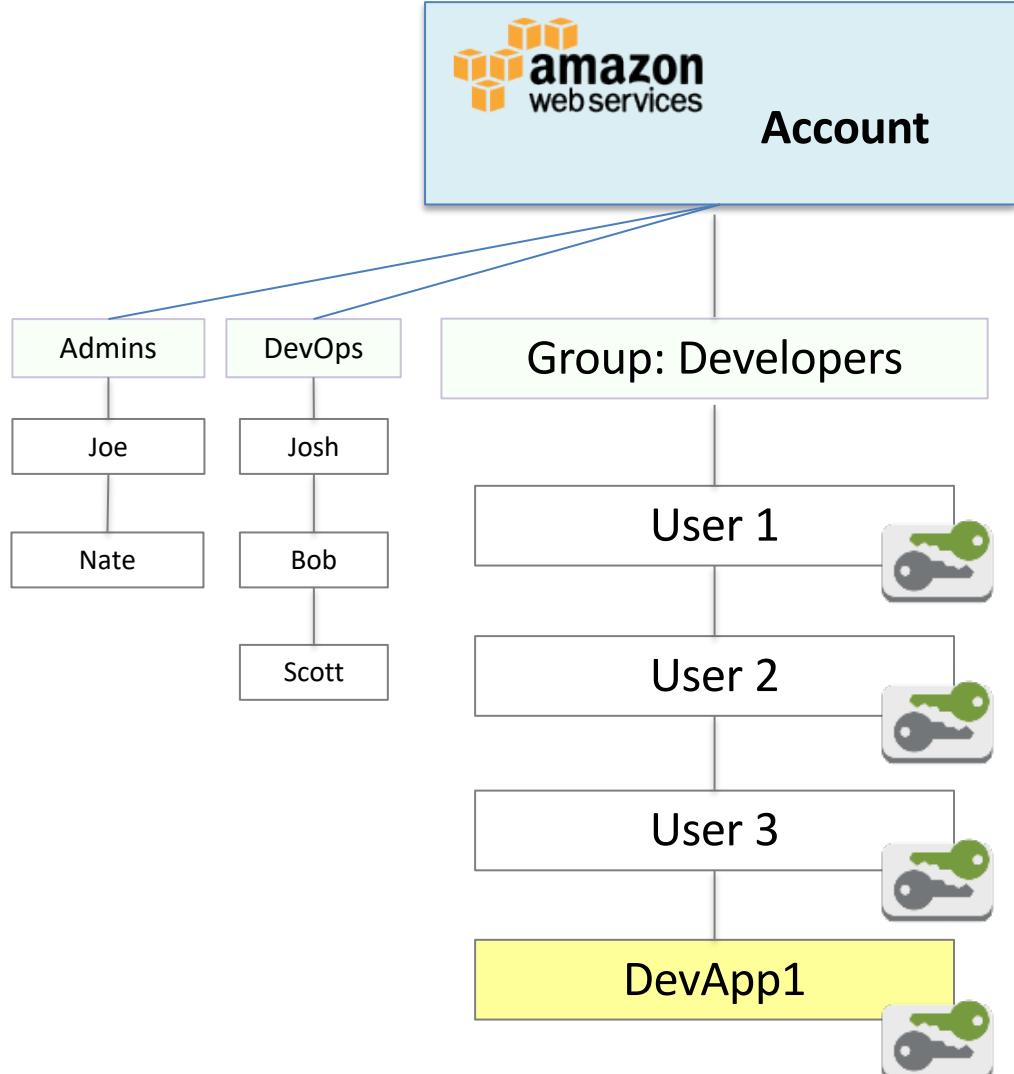
IAM Policies

- An IAM policy is a formal statement of **one or more permissions**.
 - You attach a policy to any IAM entity: user, group, or role.
 - Policies authorize the actions that may, or may not, be performed by the entity.
 - Enables fine-grained access control.
 - A single policy can be attached to multiple entities.
 - A single entity can have multiple policies attached to it.
- **Best practice:** When attaching the same policy to multiple IAM users, put the users in a group and attach the policy to the group instead.



IAM Groups

- Collection of IAM users.
- Specify permissions for the entire group.
- No default groups.
- Groups cannot be nested.
- A user can belong to multiple groups.
- Permissions are defined using IAM policies.



IAM Roles

- Used to **delegate** access to AWS resources.
 - Provides temporary access
 - Eliminates the need for static AWS credentials
- Permissions are:
 - Defined using IAM policies
 - Attached to the role, not to an IAM user or group

IAM Roles

- Use cases:

- Provide AWS resources with access to AWS services.
- Provide access to externally authenticated users.
- Provide access to third parties.
- Switch roles to access resources in:
 - Your AWS account.
 - Any other AWS account (cross-account access).

What are the first things to do with a
new AWS account?

IAM Best Practices



AWS Account Setup...

1. Stop using the root account as soon as possible.

The root account has completely unrestricted access to your resources.

To stop using the root account, take the following steps:

- 1) With the root account, create an IAM user for yourself.
- 2) Create an IAM group, give it full administrator permissions, and add the IAM user to the group.
- 3) Sign in with your IAM user credentials.
- 4) Store your root account credentials in a very secure place. Disable and remove your root account access keys, if you have them.



AWS Account Setup...

2. Require multi-factor authentication for access.
 - a. Require MFA for your root account and all IAM users.
 - b. You can also use MFA to control access to AWS service APIs.
- Software MFA options: AWS Virtual MFA, Google Authenticator, Authenticator (Windows phone app), or SMS notification
- Hardware MFA options: Key fob or display card offered by Gemalto:
<https://safenet.gemalto.com/multi-factor-authentication/>



AWS Account Setup...

3. Enable [AWS CloudTrail](#).

AWS CloudTrail logs all API requests to resources in your account.

- 1) Via the CloudTrail console: create a trail, give it a name, apply it to all regions, and enter a name for the new Amazon S3 bucket that the logs will be stored in.
- 2) Ensure that the Amazon S3 bucket you use for CloudTrail has its access restricted to only those who should have access, such as admins.

AWS Account Setup...

4. Enable a [billing report](#), such as the AWS Cost and Usage Report.
 - a) Billing reports provide information about your usage of AWS resources and estimated costs for that usage.
 - b) AWS delivers the reports to an Amazon S3 bucket that you specify and updates the reports at least once a day.
 - c) The AWS Cost and Usage Report tracks your AWS usage and provides estimated charges associated with your AWS account, either by the hour or by the day.

Network Design using VPC

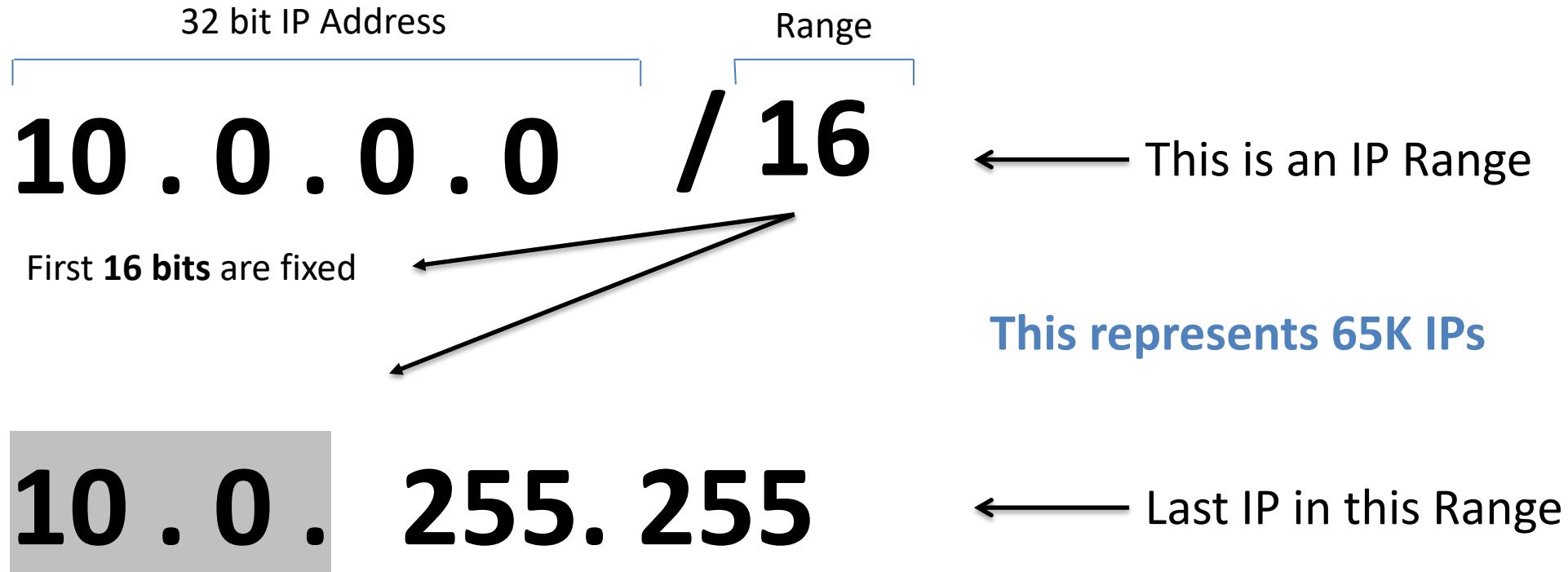
Topics

- IP Address CIDR Notation
- Creating a VPC from Scratch
- Demo
- Network Security
- Design Principles
- Extending VPC to your Data Centre

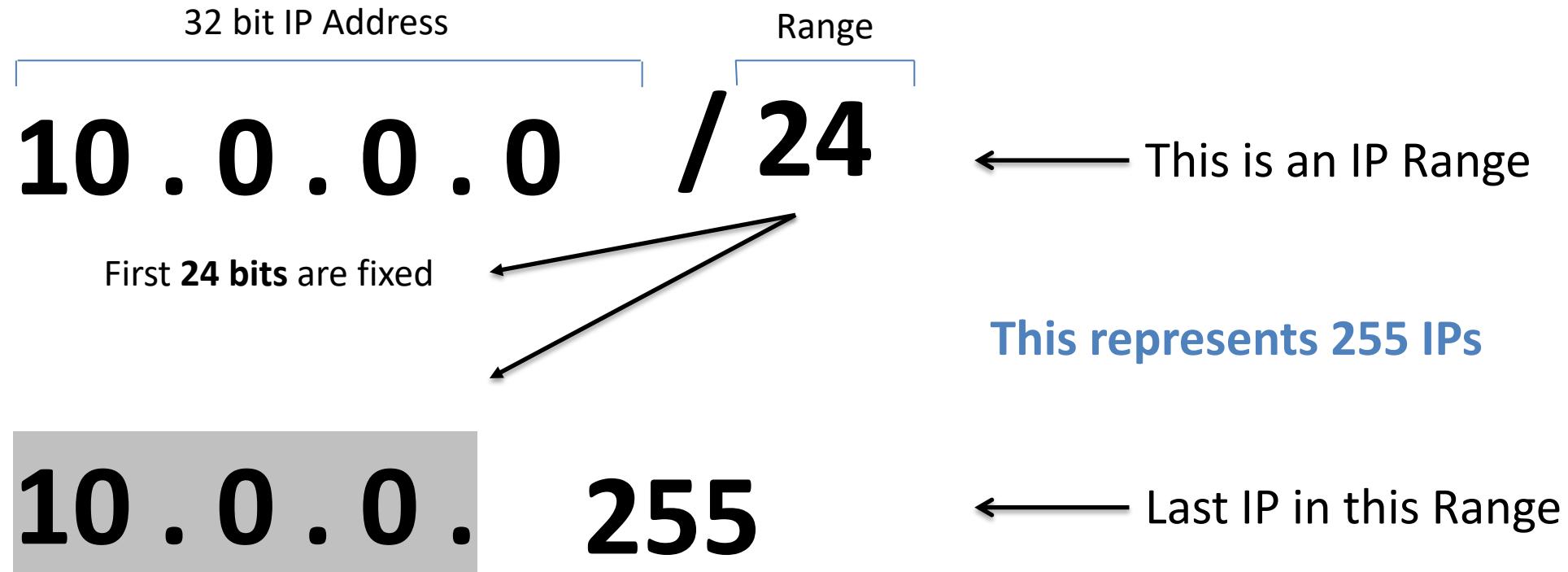
VPC – Virtual Private Cloud

- VPC = Networking in AWS Cloud
- VPC lets you virtually isolate your infrastructure with in AWS Region.
- It gives you full control on Routing Rules, Network firewalls and infrastructure placement.

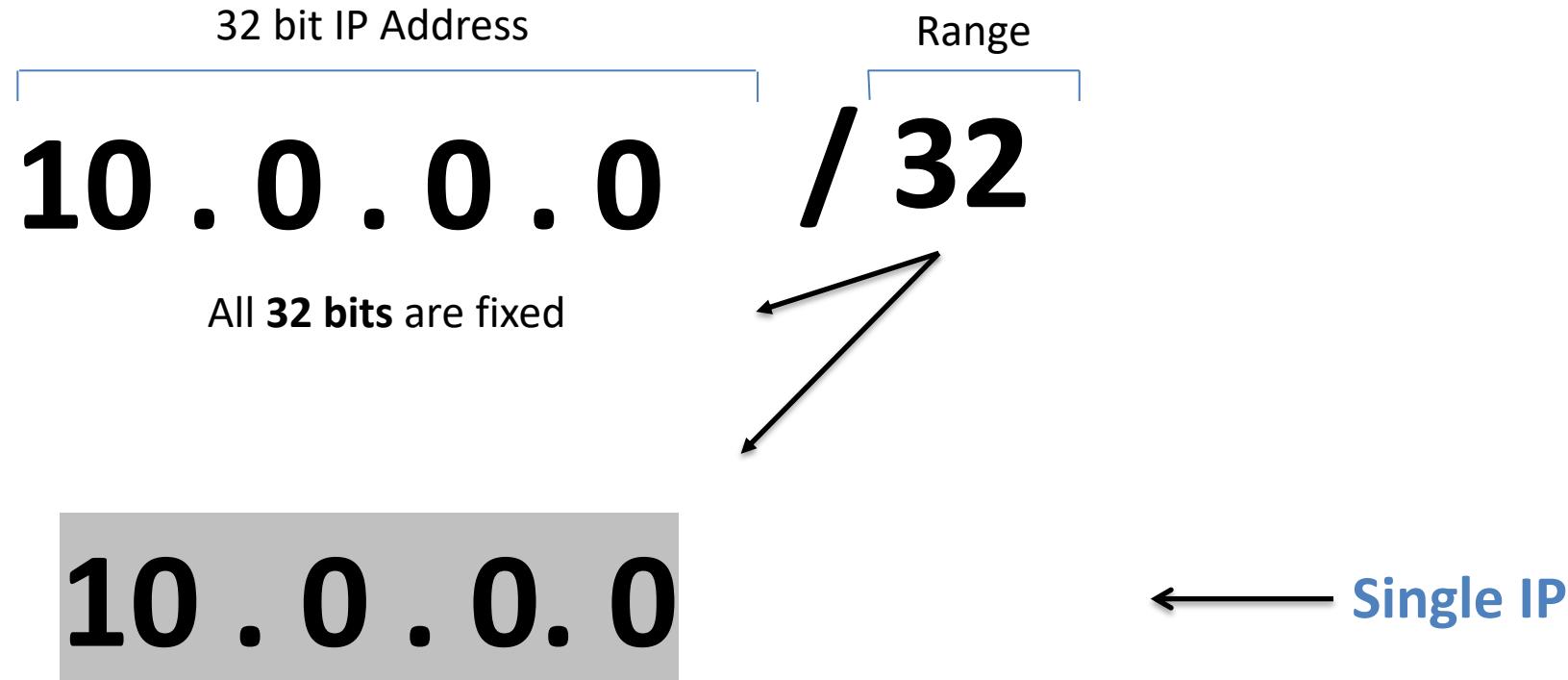
CIDR Notation



CIDR Notation



CIDR Notation



CIDR Notation

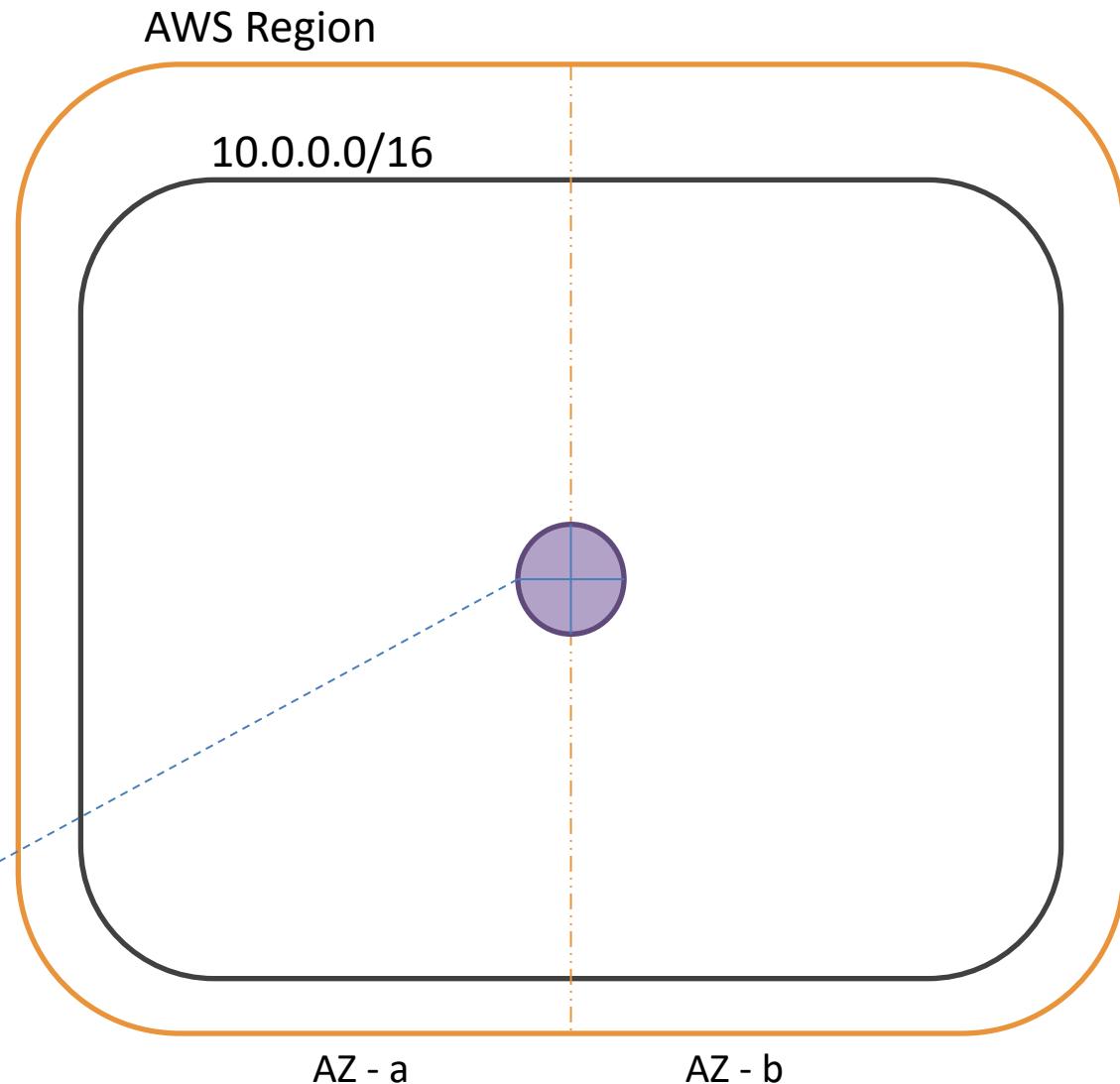
32 bit IP Address Range
0 . 0 . 0 . 0 / 0 ← This is an IP Range

255.255.255.255 ← Entire IPV4 Range

VPC – Virtual Private Cloud

- A VPC spans across multiple Availability Zones within a region.
- VPC comes with an implicit router and a default route table to route the local traffic within that VPC.

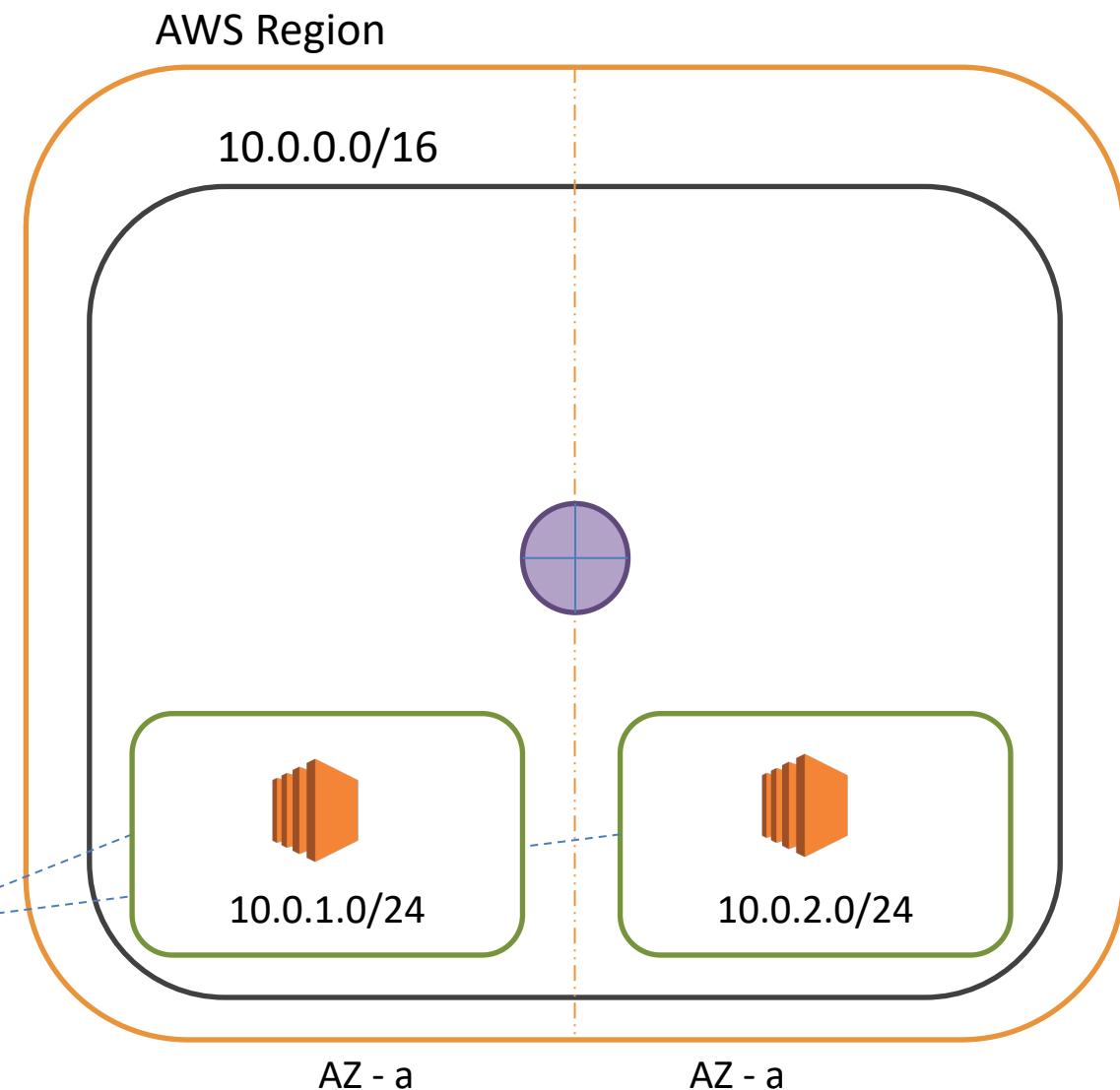
Main RT	
Destination	Target
10.0.0.0/16	Local



VPC – Private Subnets

- A newly created subnets gets associated with the Default Route Table.
- This association makes a newly created subnet as a **Private Subnet**

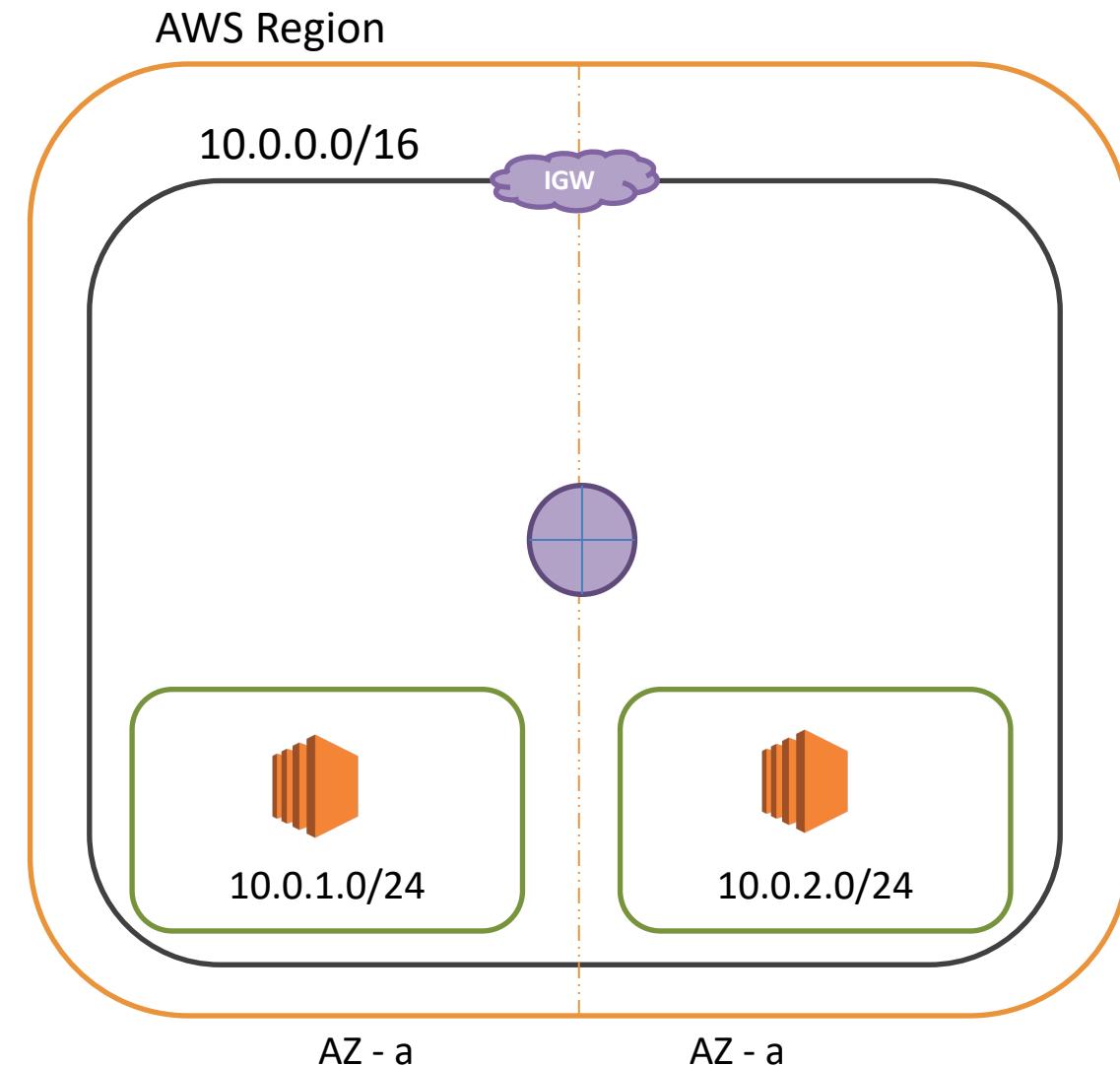
Main RT	
Destination	Target
10.0.0.0/16	Local



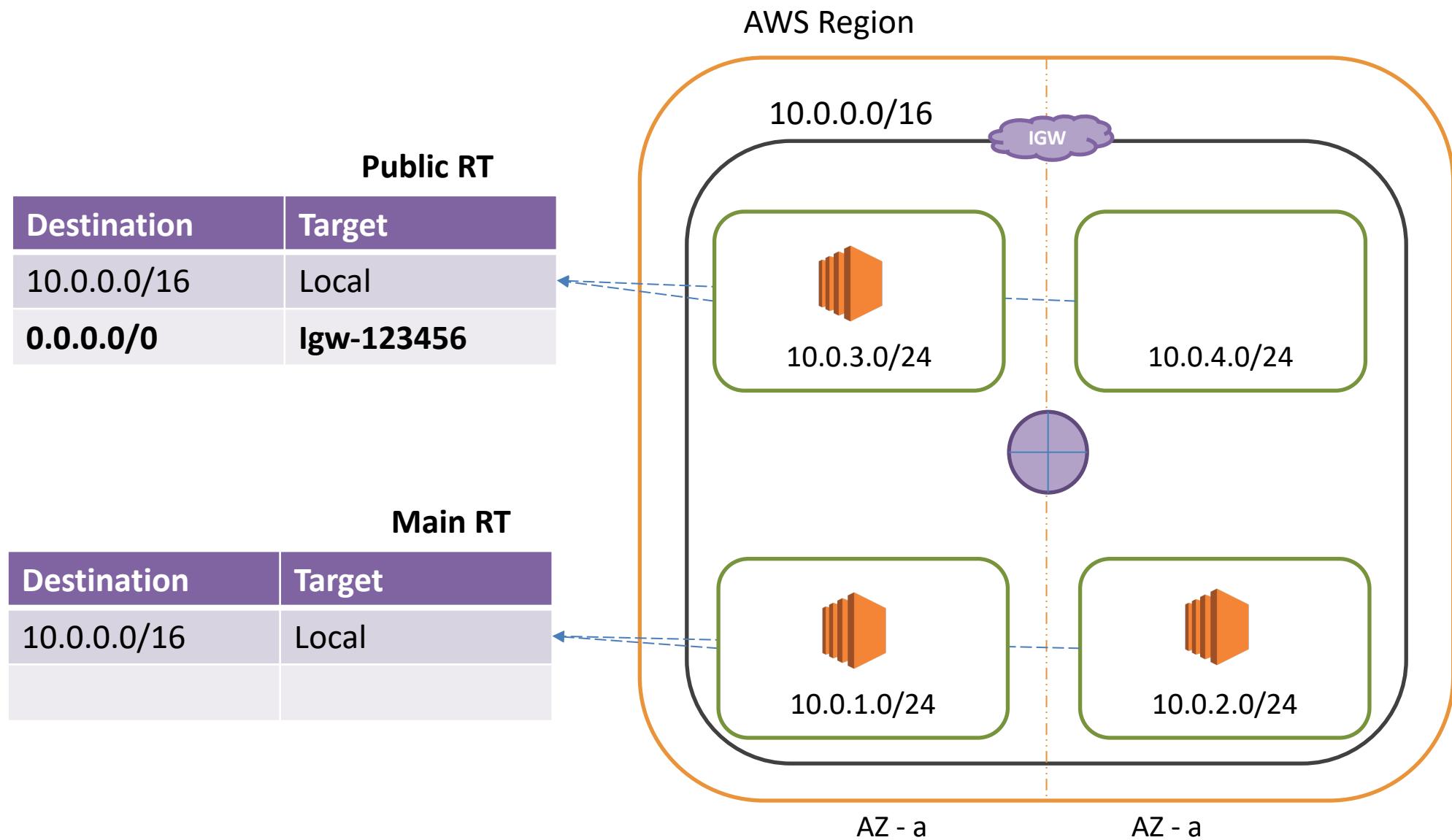
VPC – Internet Gateway

- IGW – Internet Gateway is a virtualized network component that allows external n/w connectivity.
- Attach IGW to your VPC and make a Route Table entry.

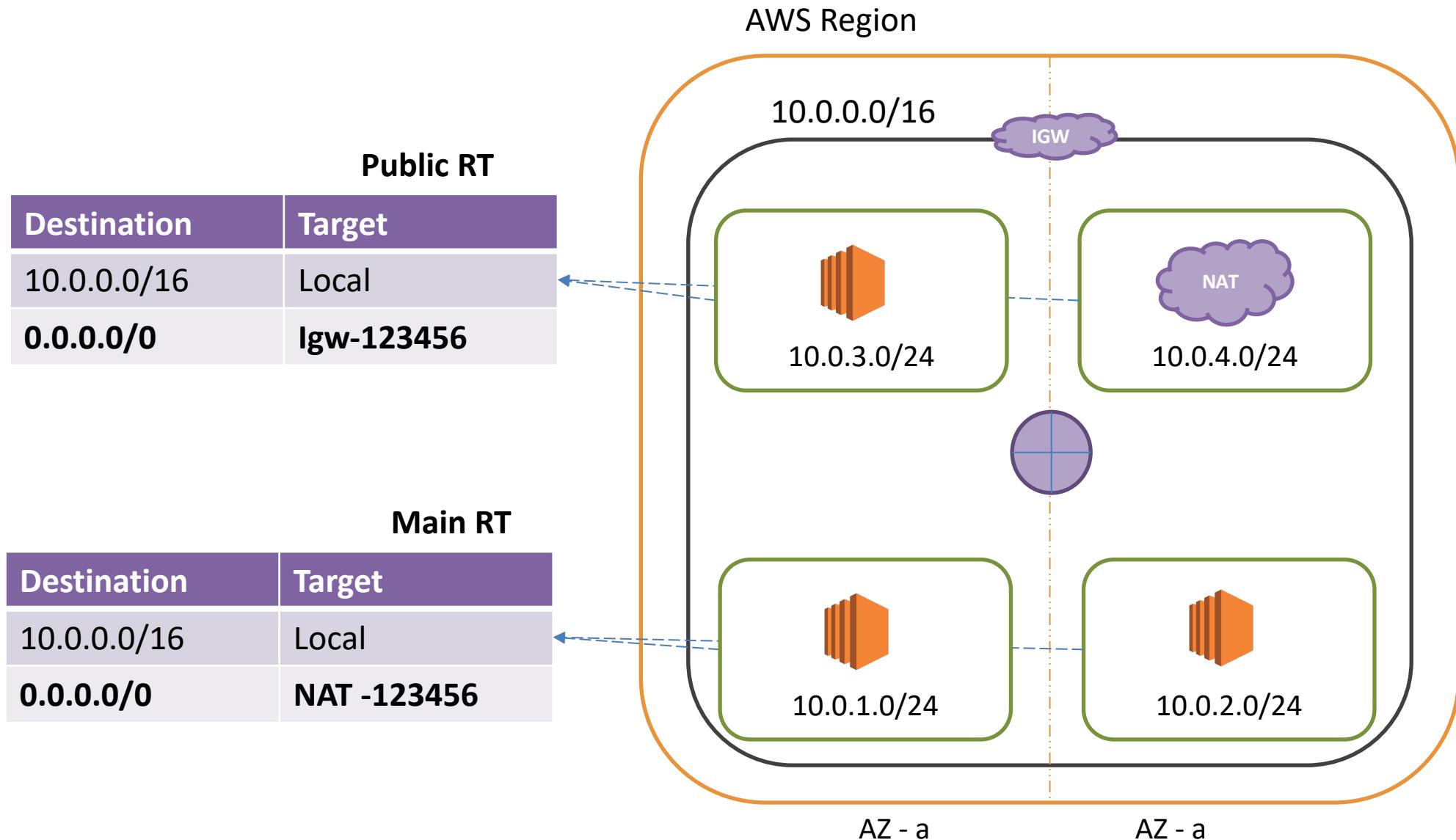
Public RT	
Destination	Target
10.0.0.0/16	Local
0.0.0.0/0	Igw-123456



VPC – Public Subnets

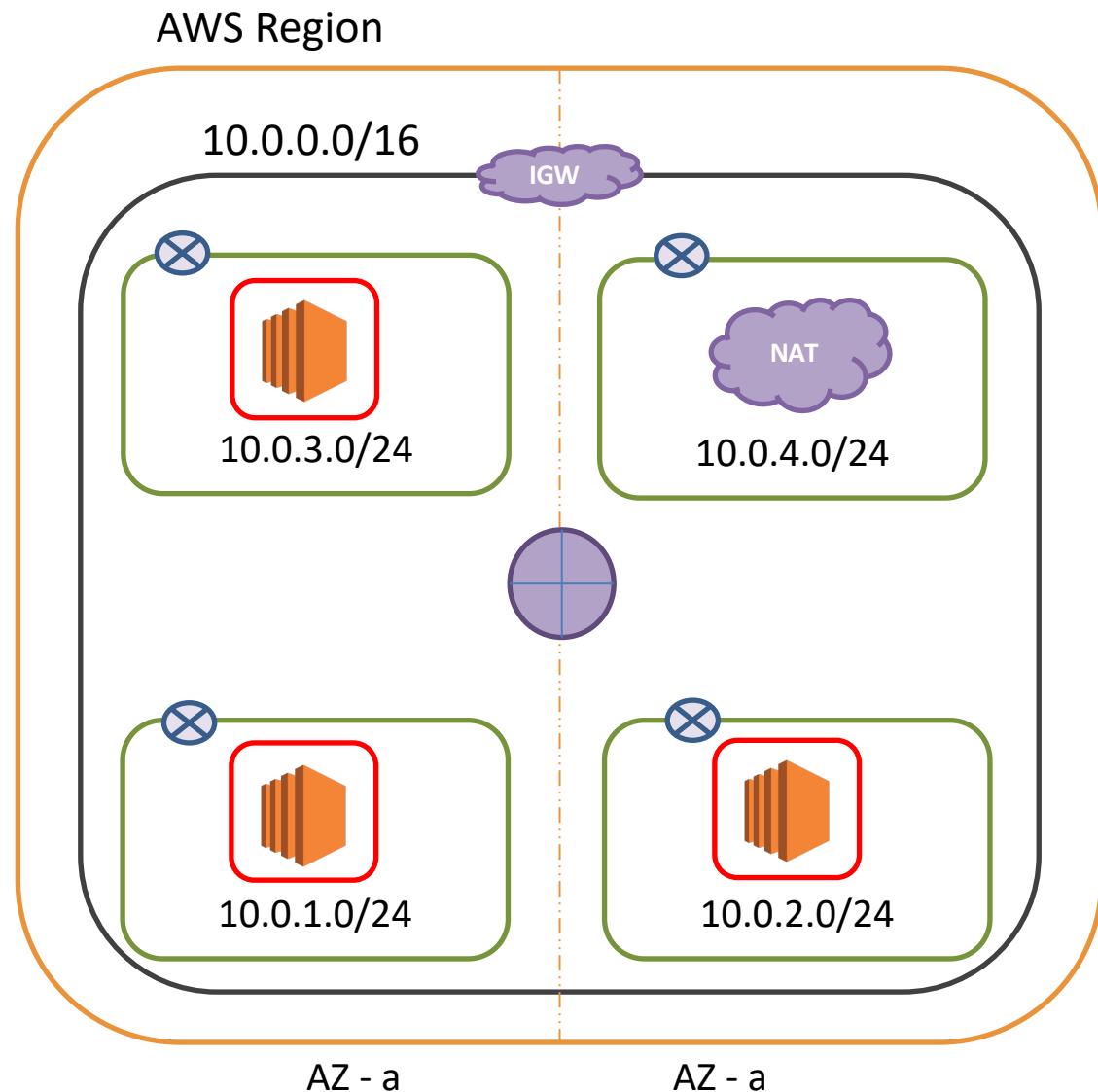


VPC – NAT Gateway



Network Security

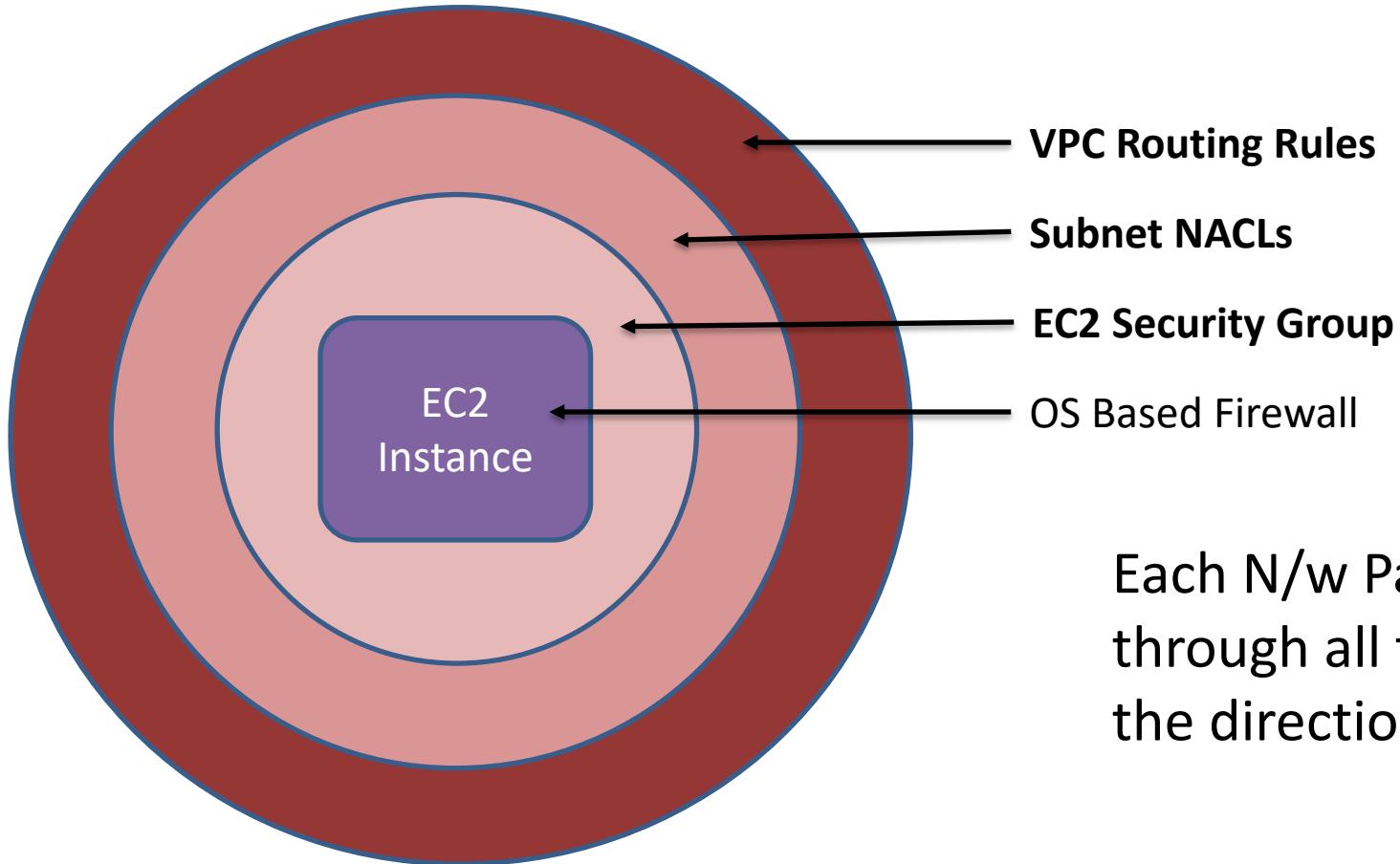
- **NACLs** – Network Access Control Lists are network firewalls that can be setup at subnet level.
- **Security Groups** are Host based firewall.



Network Security

- **NACL**
 - Applied at Subnet Level
 - Stateless
 - All Traffic is Allowed By Default
 - Rules are executed in the order of weight specified against each rule.
- **Security Group**
 - Applied at EC2 Instance Level
 - Statefull
 - All Inbound Traffic is blocked By Default
 - Rules are aggregated. Order is insignificant.

Network Security Defence



Each N/w Packet will pass through all these layers in both the directions.

Design Principles

- How many VPCs would you create ?
 - Multiple VPCs v/s Multiple Accounts.
- What Should be the size of a VPC ?
 - Max allowable size /16. (Smallest is /28)
- What should be the size of your subnets ?
 - /24 or Larger Private Subnet.
 - Relatively smaller Public Subnets.

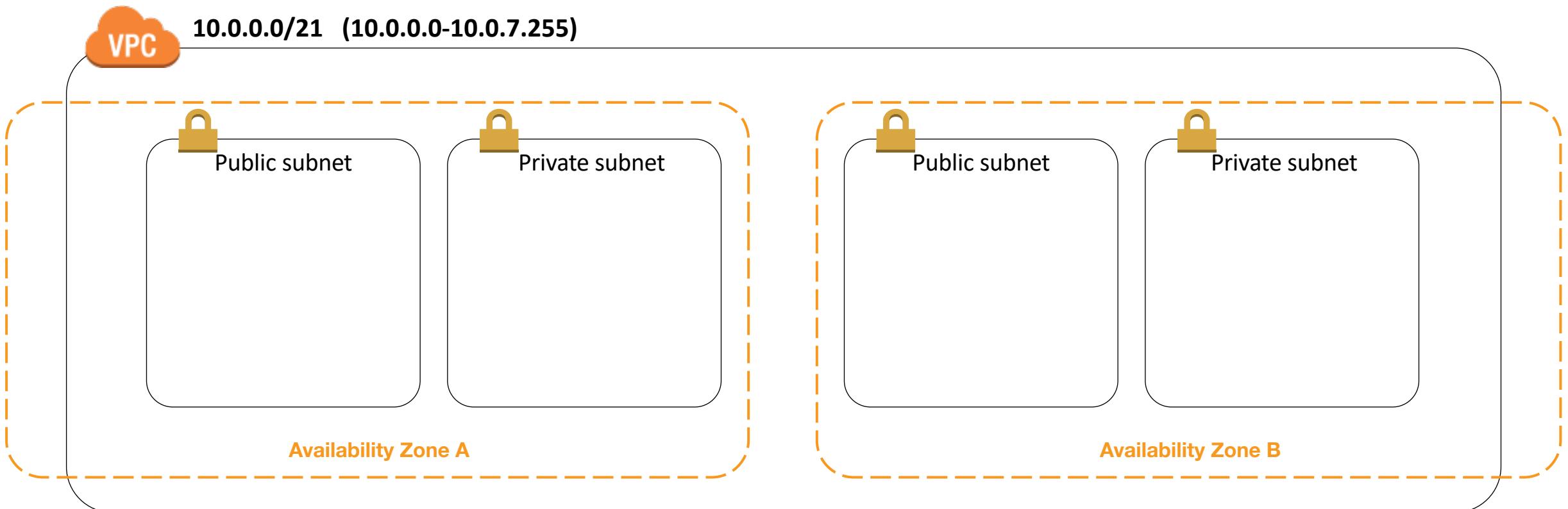
VPCs and IP Addresses

- 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	

Subnets

- Recommendation: Start with one public and one private subnet per Availability Zone.



Next Topics - Extending Your VPC

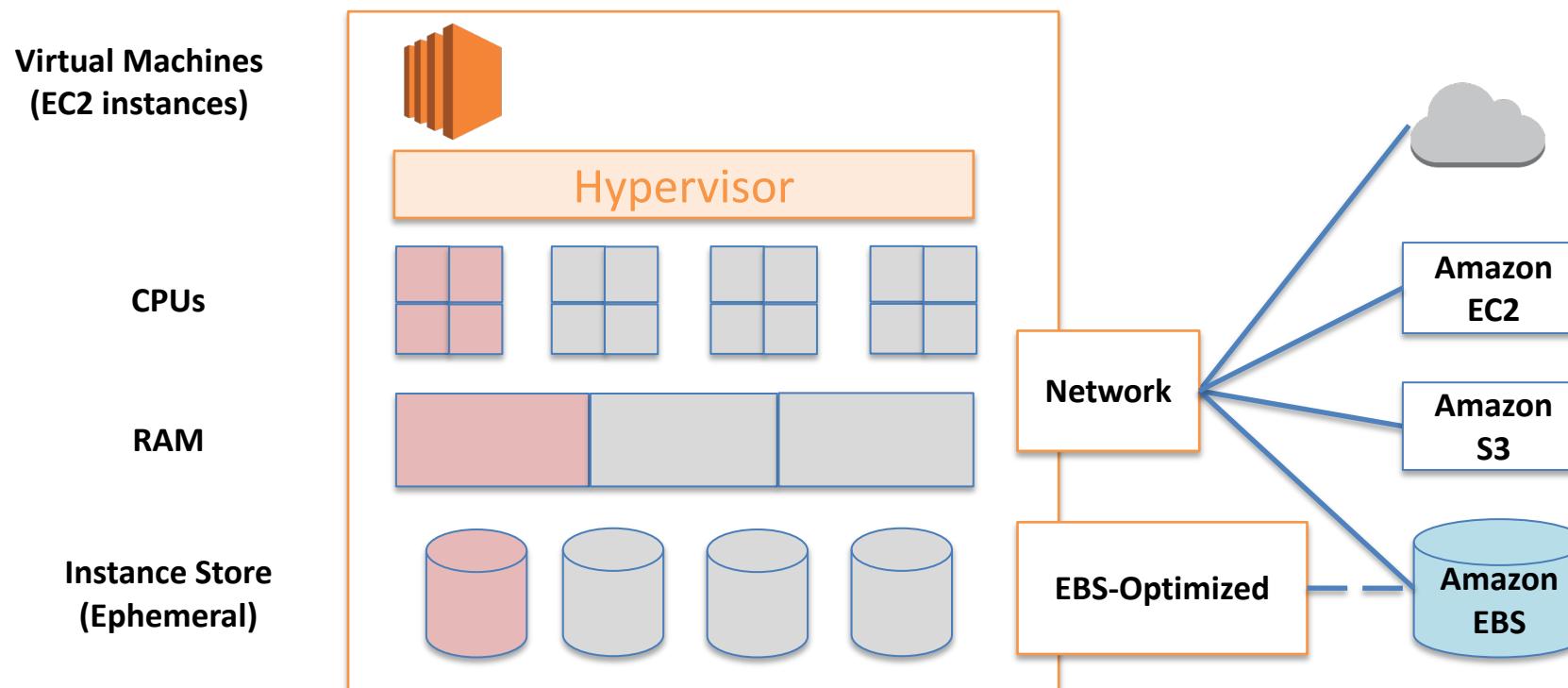
- Within AWS Regions
 - VPC Peering
 - VPN Across 2 AWS regions
- To Your DataCentre
 - VPN IPSec Tunnel over the Internet
 - VPN over **DirectConnect**

Working With EC2.

Topics

- EC2 Internal Architecture
- Demo
- Instance Configurations
- Instance Life Cycle
- Pricing

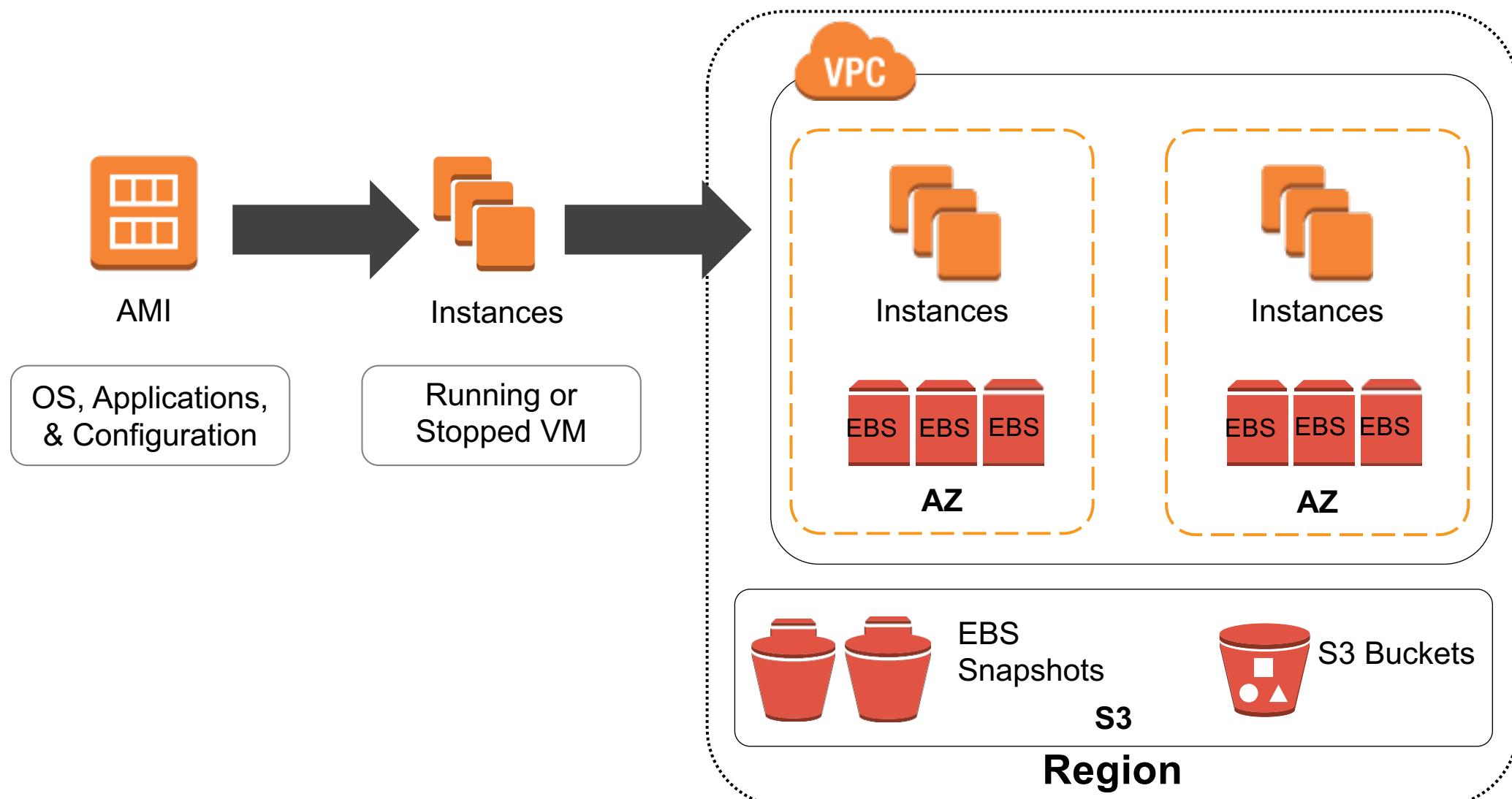
EC2 Internal Architecture



Current Generation Instances

Instance Family	Some Use Cases
General purpose (t2, m4, m3)	<ul style="list-style-type: none">• Low-traffic websites and web applications• Small databases and mid-size databases
Compute optimized (c4, c3)	<ul style="list-style-type: none">• High performance front-end fleets• Video-encoding
Memory optimized (r3)	<ul style="list-style-type: none">• High performance databases• Distributed memory caches
Storage optimized (i2, d2)	<ul style="list-style-type: none">• Data warehousing• Log or data-processing applications
GPU instances (g2)	<ul style="list-style-type: none">• 3D application streaming• Machine learning

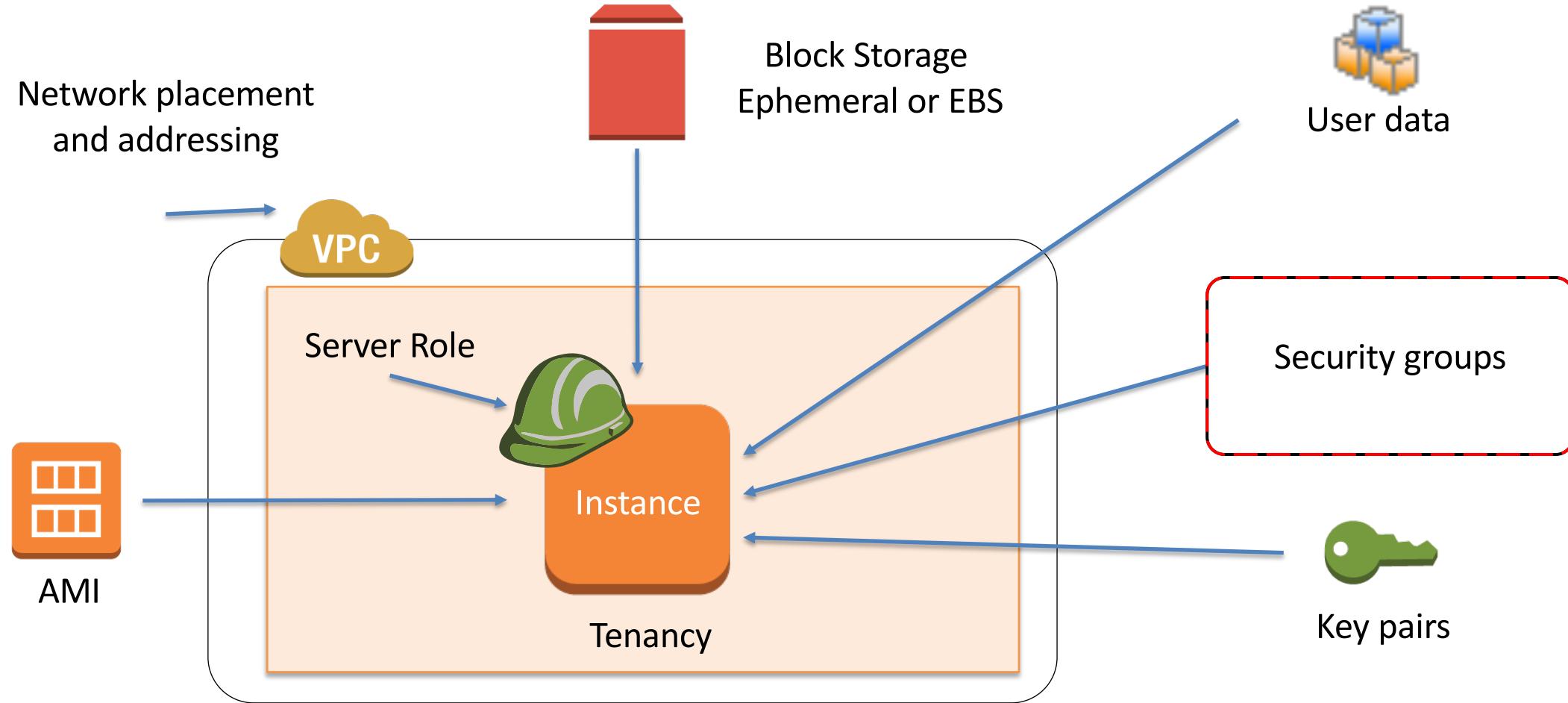
Instances Placement



Demo

- How to launch EC2 Instance

Lunching EC2 Instance



Amazon Elastic Block Store (EBS)

- Network-attached disk storage
- Types of EBS volumes
 - General Purpose (SSD)
 - Provisioned IOPS (SSD)
 - Magnetic options
- Data persist when instance is stopped.
- Data persist when instance is terminated, provided the **DeleteOnTermination** attribute is false.

Instance Store (Ephemeral)

- Instance store volumes are directly attached to a host computer.
- Instance Store SSD volumes have fast disk access suitable for swap files, caches, buffers, and highly replicated data.
- Instance store volume contents are **lost** when an instance is stopped or terminated.
- Instance store volumes can offer up to 315,000 IOPS for some instance types. <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/i2-instances.html#i2-instances-diskperf>

User Data

- User data scripts supplied to initialize instances automatically.
 - Linux script
 - Windows batch or PowerShell scripts
- User data scripts can install any software package
 - Web server
 - Database server
 - Configuration management tools.
- User data script runs once per instance-id by default.

User Data Example Linux

```
#!/bin/sh
```

User data shell scripts must start with the #! characters and the path to the interpreter you want to read the script.

```
yum -y install httpd  
chkconfig httpd on  
/etc/init.d/httpd  
start
```

Install Apache web server
Enable the web server
Start the web server

User Data Example Windows

```
<powershell>
```

```
  Import-Module ServerManager
```

Import the Server Manager module for Windows PowerShell.

```
  Install-WindowsFeature web-server, web-webserver
```

```
  Install-WindowsFeature web-mgmt-tools
```

```
</powershell>
```

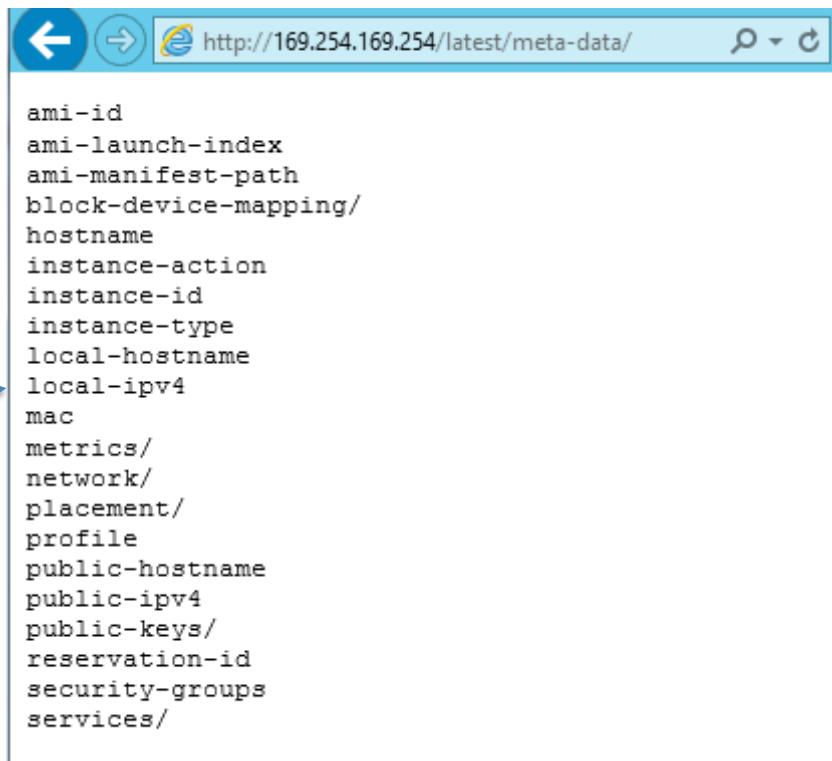
Install IIS

Install Web Management Tools



Retrieving Instance Metadata

- To view all categories of instance metadata from within a running instance, use the following URI:
`http://169.254.169.254/latest/meta-data/`
- On a Linux instance, you can use:
 - `$ curl http://169.254.169.254/latest/meta-data/`
 - `$ GET http://169.254.169.254/latest/meta-data/`
- All metadata is returned as text (content type `text/plain`).



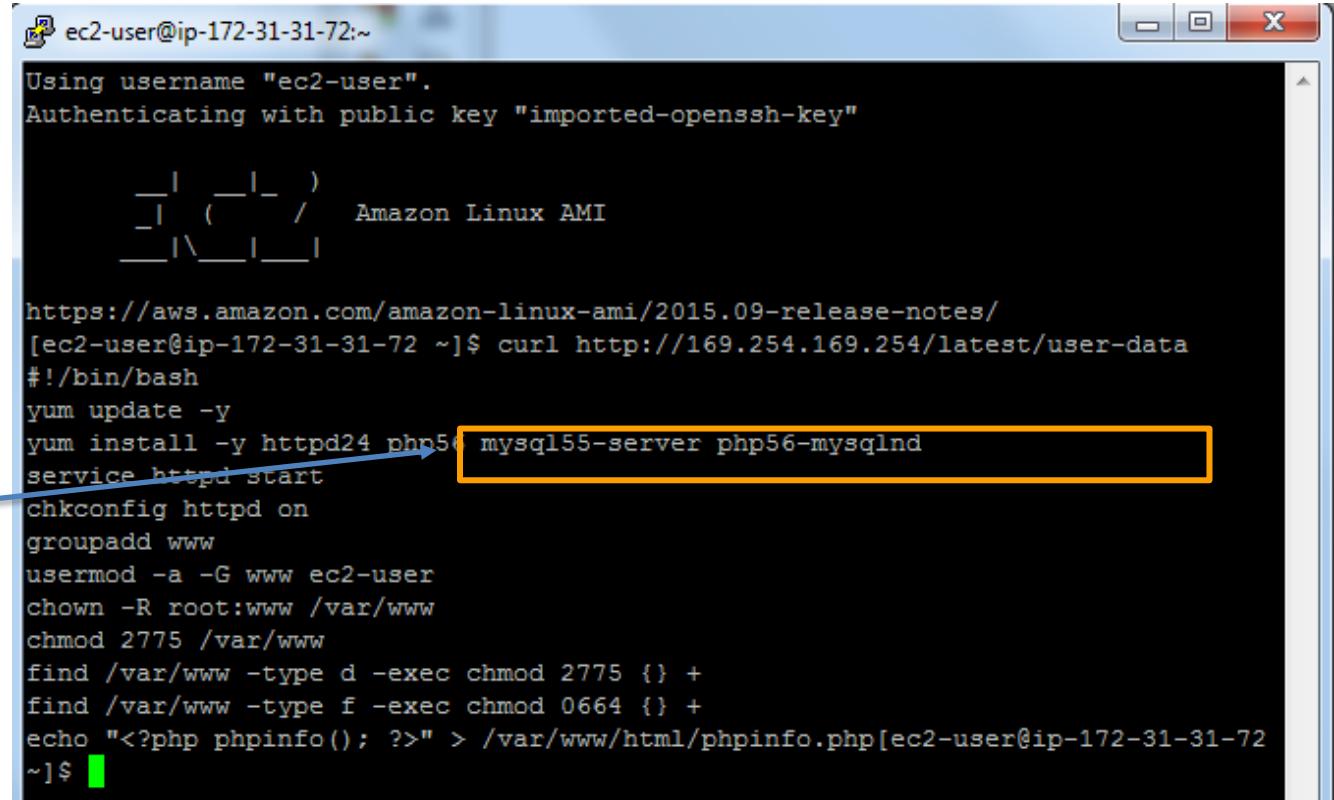
Retrieving User Data

- To retrieve user data, use the following URI:

`http://169.254.169.254/latest/
user-data`

- On a Linux instance, you can use:

- `$ curl
http://169.254.169.254/latest/
user-data/`
- `$ GET
http://169.254.169.254/latest/
user-data/`



The screenshot shows a terminal window titled "ec2-user@ip-172-31-31-72:~". The window displays the following text:

```
Using username "ec2-user".
Authenticating with public key "imported-openssh-key"

[Amazon Linux AMI]

https://aws.amazon.com/amazon-linux-ami/2015.09-release-notes/
[ec2-user@ip-172-31-31-72 ~]$ curl http://169.254.169.254/latest/user-data
#!/bin/bash
yum update -y
yum install -y httpd24 php56 mysql55-server php56-mysqlnd
service httpd start
chkconfig httpd on
groupadd www
usermod -a -G www ec2-user
chown -R root:www /var/www
chmod 2775 /var/www
find /var/www -type d -exec chmod 2775 {} +
find /var/www -type f -exec chmod 0664 {} +
echo "<?php phpinfo(); ?>" > /var/www/html/phpinfo.php[ec2-user@ip-172-31-31-72
~]$
```

A blue arrow points from the URL in the first bullet point to the "curl" command in the terminal window. A yellow box highlights the "mysql55-server" and "php56-mysqlnd" packages in the "yum install" command.

Security Groups

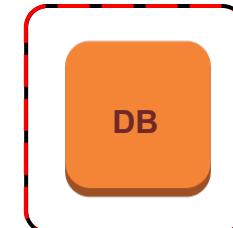
Remote Access 22



Remote Access 22



port 3306



- Restrict access to instances by:
 - Port range
 - IP range
 - Security group or resource ID
- Instances can be associated with multiple security groups.
- Allow data ingress and egress.
- Can be added/modified after launch.
 - EC2-Classic rules are ingress-only, applied only at launch.

Security Groups - Examples

Open HTTP port access from anywhere

ID	Port Range	Source
sg-fdgh234sk	80 (HTTP)	0.0.0.0/0

Open SSH access from a specific IP

ID	Port Range	Source
sg-abc123xyz	22 (SSH)	53.1.2.12/32

Open SSH access from members of a security group (e.g., a bastion host)

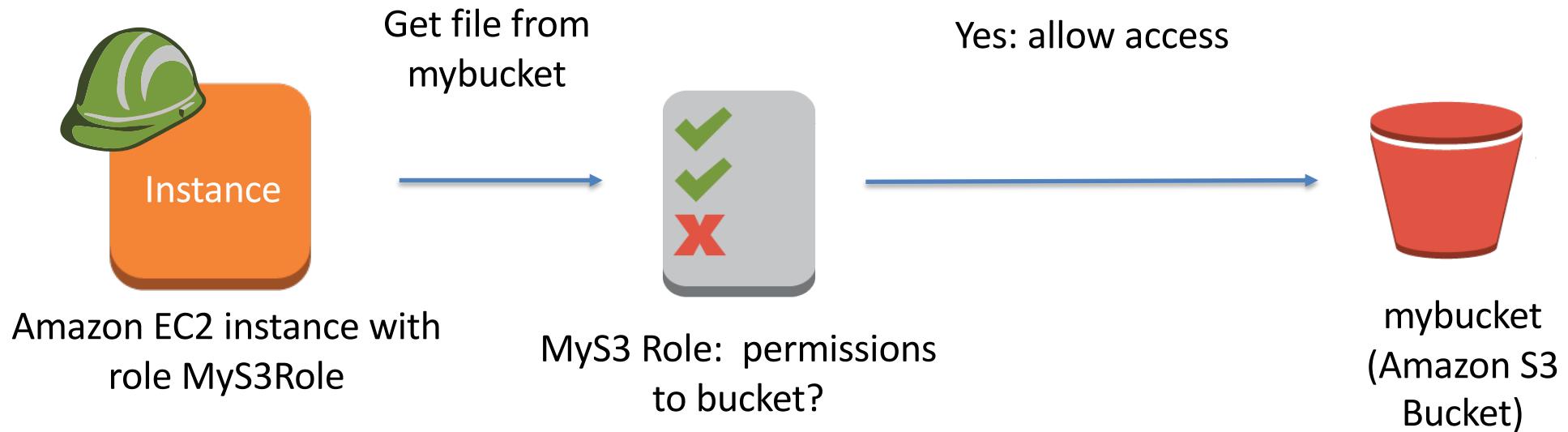
ID	Port Range	Source
sg-4kdf23fb4	22 (SSH)	sg-abc123xyz

Key Pairs

- Remote Access
 - **Linux** - Use SSH to log into an instance since password-only authentication is disabled on Linux by default.
 - **Windows** - Use RDP to log into instances using an encrypted random password that can only be decrypted using a private secret access key.

IAM Roles

- Useful when EC2 instances must access other AWS services, such as Amazon S3.



IAM Roles

- Use IAM Roles on EC2 instances
 - Automatic propagation of access keys to instances
 - Auto-rotation of access keys multiple times daily
 - Use across multiple instances (e.g., Auto Scaling group)

Types of AMI

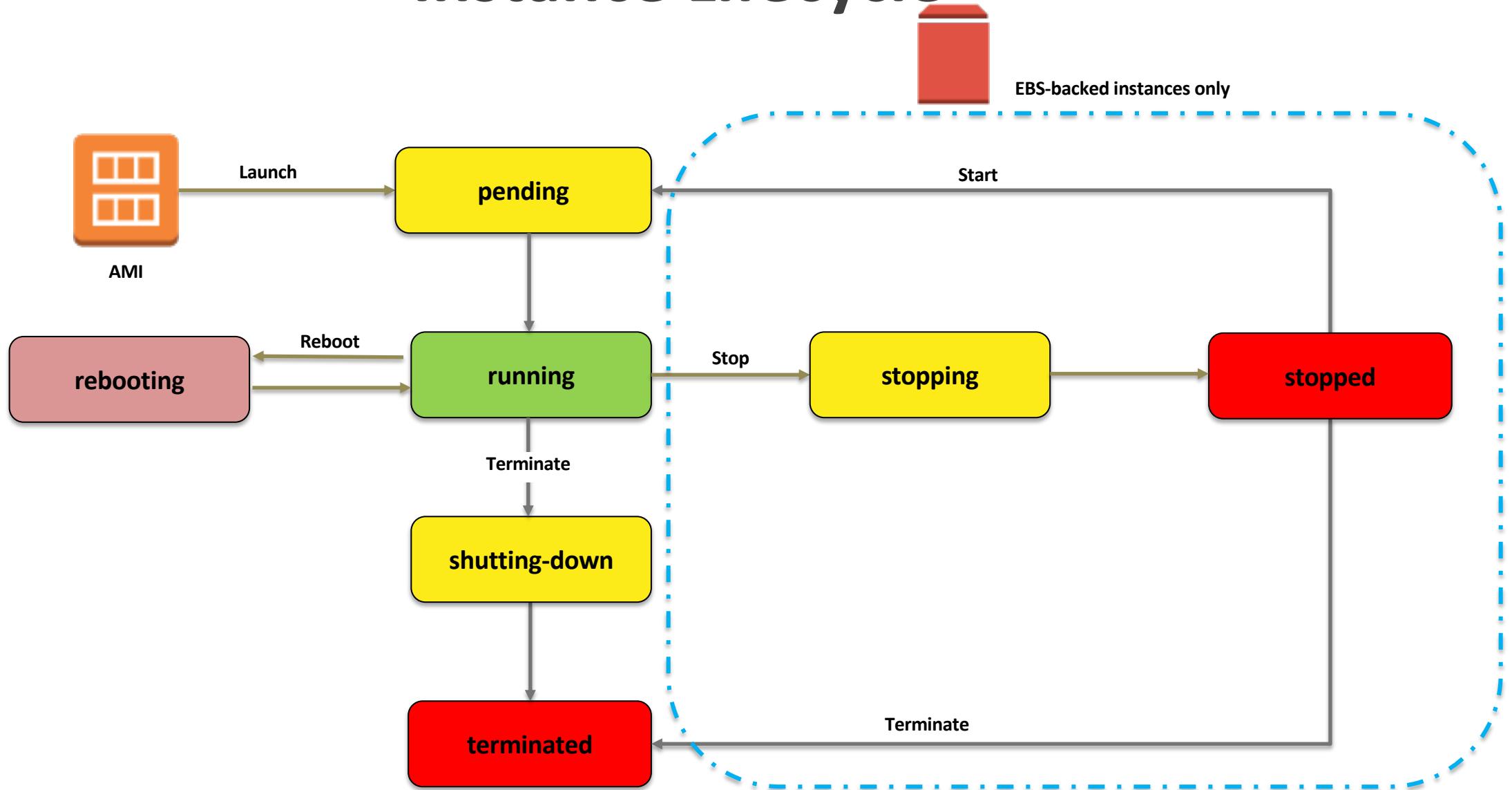
- EBS Backed AMI

Root Volumes will be created on
an **EBS Volume**

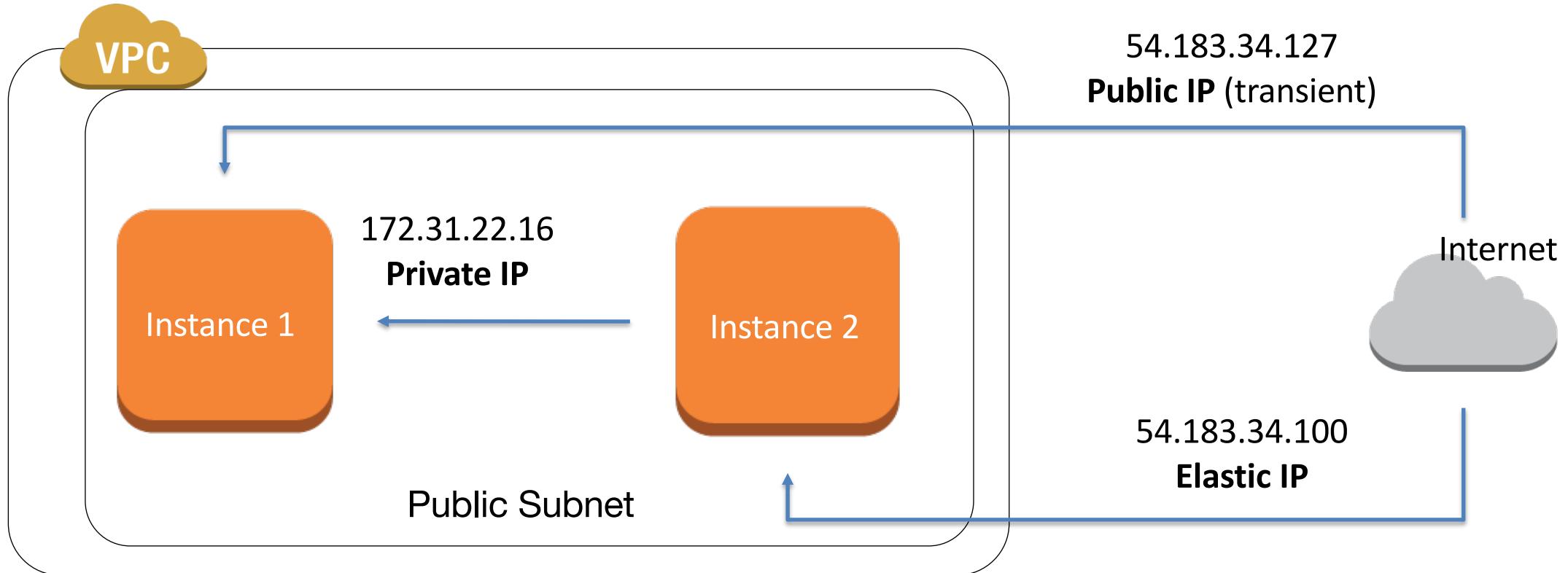
- Instance Storage
Backed AMI

Root Volumes will be created on an
Instance Storage Volume

Instance Lifecycle



Private IPs, Public IPs, Elastic IPs

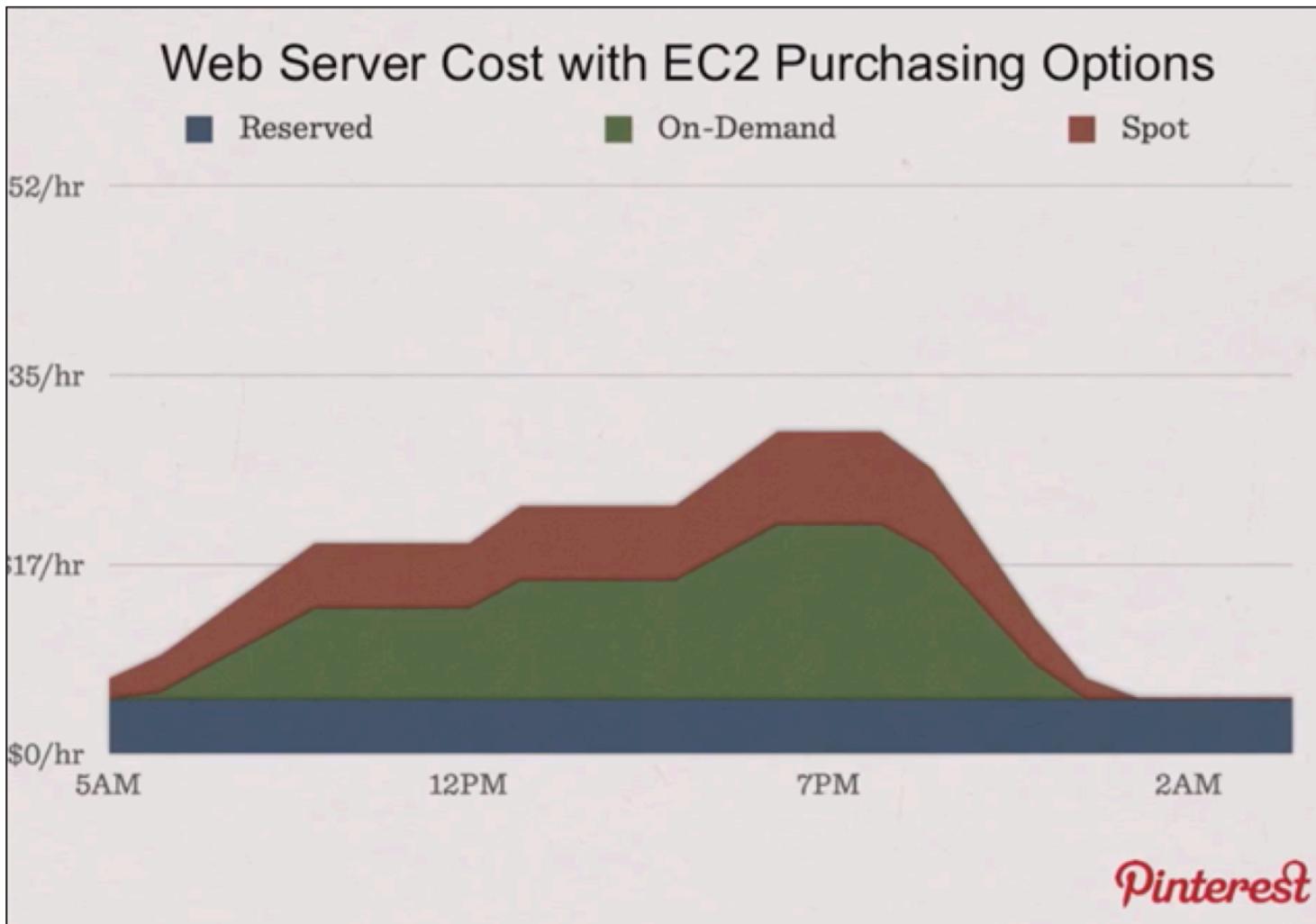


- Public IPs change when an instance stops or terminates.
- Use Elastic IPs for static IP addresses.

EC2 Pricing

On-Demand	Reserved	Scheduled RI	Dedicated	Spot
<ul style="list-style-type: none">• Per Second billing• No long term Commitment	<ul style="list-style-type: none">• 3 Yr or 1 Yr• 35% to 65% Discount• Flexibility to change size	<ul style="list-style-type: none">• For Periodic workload• 10% Discount	<ul style="list-style-type: none">• Single tenancy at the physical host level.	<ul style="list-style-type: none">• Bid for AWS's unutilized capacity.• Upto 80% Cheaper.

Using a Mix of Pricing Types



Reboot vs. Stop vs. Terminate

Characteristic	Reboot	Stop/Start (EBS-backed instances only)	Terminate
Host computer	The instance stays on the same host computer.	The instance runs on a new host computer.	N/A
Private and public IP addresses	Stay the same.	Instance keeps its private IP address and gets a new public IP address.	N/A
Elastic IP addresses (EIP)	EIP remains associated with the instance.	EIP remains associated with the instance.	The EIP is disassociated from the instance.
Instance store volumes	The data is preserved.	The data is erased.	The data is erased.
EBS volume	The volume is preserved.	The volume is preserved.	The 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.

Relational Database RDS

Topics

- What is RDS?
- Multi-AZ Deployment
- Scaling RDS
- Read Replicas
- Pricing

Relational Database Service (RDS)

- **Managed Service**
 - DB Setup, HA Replication, Automated Backup, Scaling.
- **Resizable Capacity**
 - Compute, Database Storage, Storage IO.
- **Deploy MySQL, MariaDB, Microsoft SQL Server, Oracle, and PostgreSQL databases**
- **Amazon Aurora – MySQL, PostgreSQL**

How RDS Backup Works

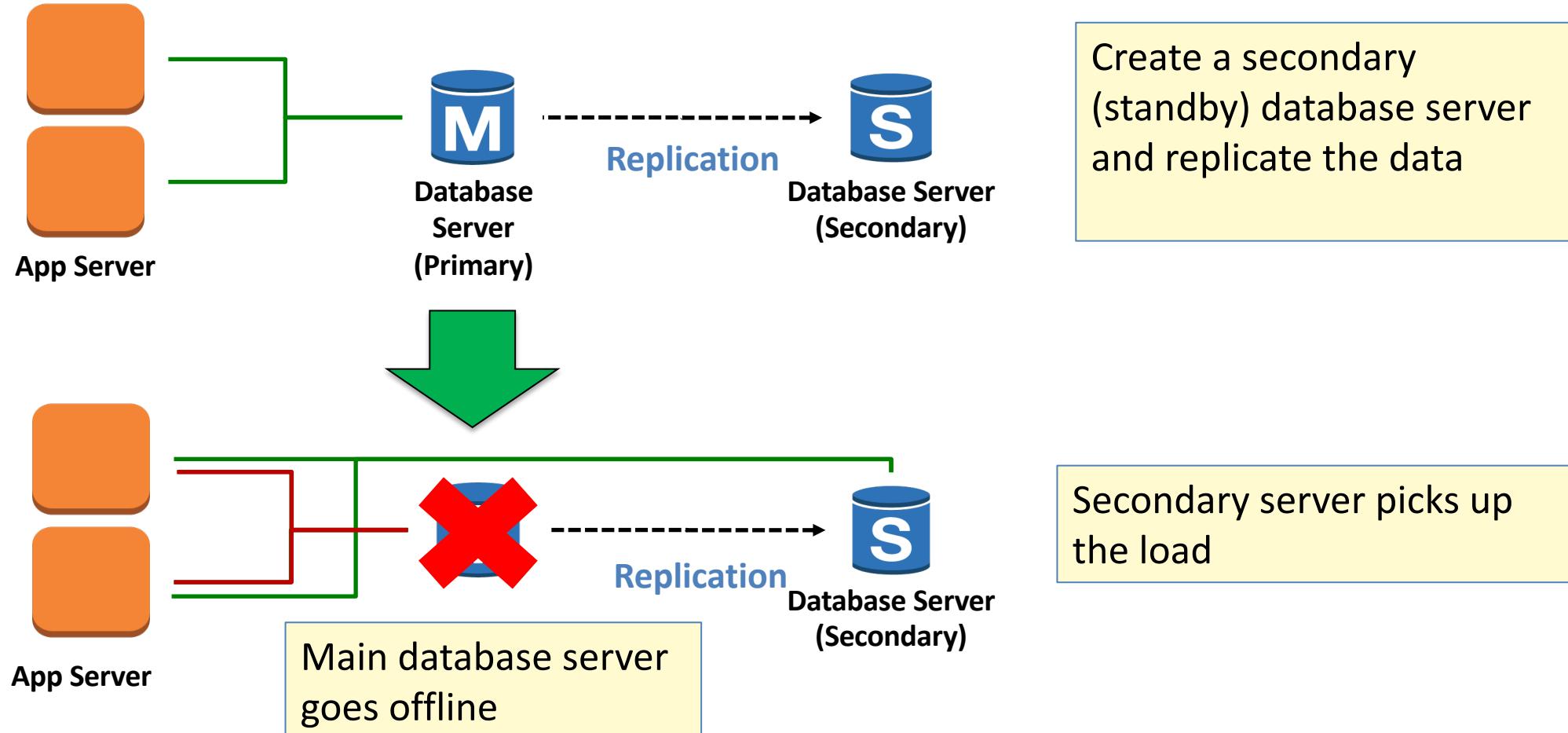
- Automatic Backups:
 - Restore your database to a point in time.
 - Are enabled by default.
 - Let you choose a retention period up to 35 days.
- Manual Snapshots:
 - Let you  database

Cross-Region Snapshots for DR

- Are a copy of a database snapshot stored in a different AWS Region.
- Provide a backup for disaster recovery.
- Can be used as a base for migration to a different region.



RDS Multi-AZ Deployment for HA



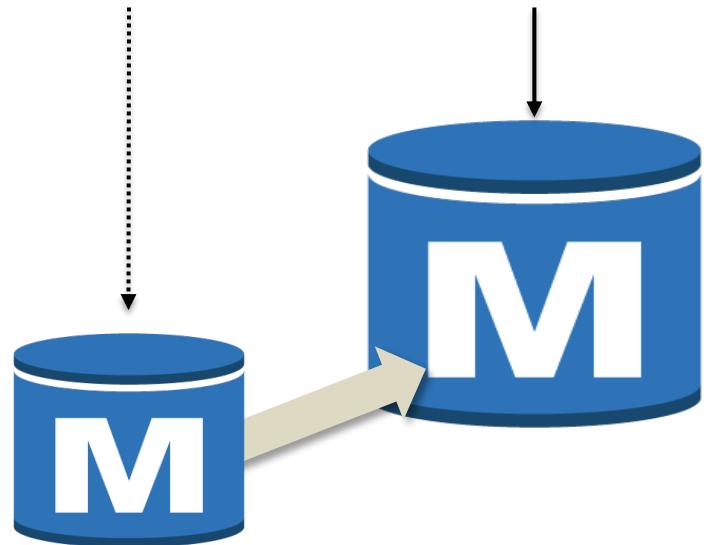
Multi-AZ RDS Deployment for HA

- With Multi-AZ deployment, your database is **synchronously** replicated to another AZ within the same AWS Region.
- Automated **Failover** to the standby **within 30 Sec** in case of primary database failure.
- Database **endpoints** does not change during the failover.
- Planned maintenance is applied first to standby databases.

RDS: Push-button Scaling

- Scale nodes vertically up or down with minimal or no downtime
 - Compute node from micro to 8xlarge and everything in-between.
 - Database Storage.
 - Storage IO from General Purpose to PIOPS.

Push-Button Scaling



RDS Scaling with Read Replicas

- Horizontally scale your database for read-heavy workloads by adding **Read only Node**.
- Example - Offload reporting
- Replication is **asynchronous**; hence, Replica node can lag by few milliseconds to few seconds from the Master node.

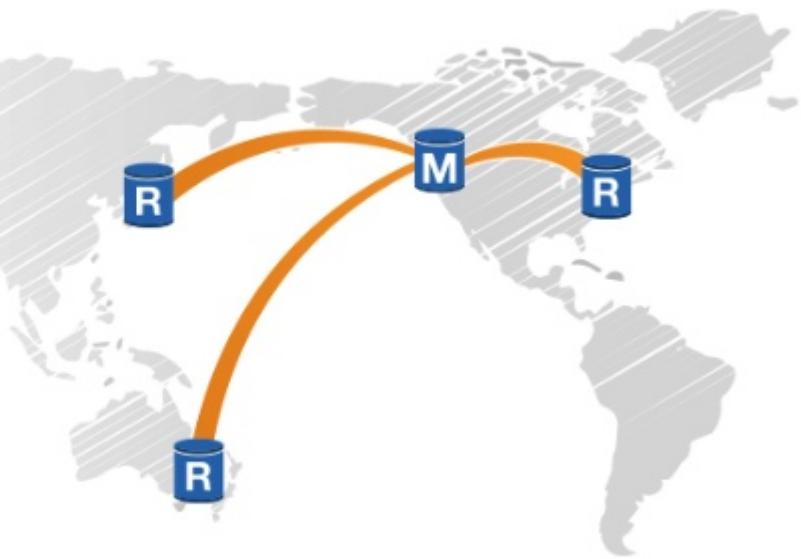
Read Replicas

Within a region

- MySQL
- MariaDB
- PostgreSQL
- Aurora

Cross-region

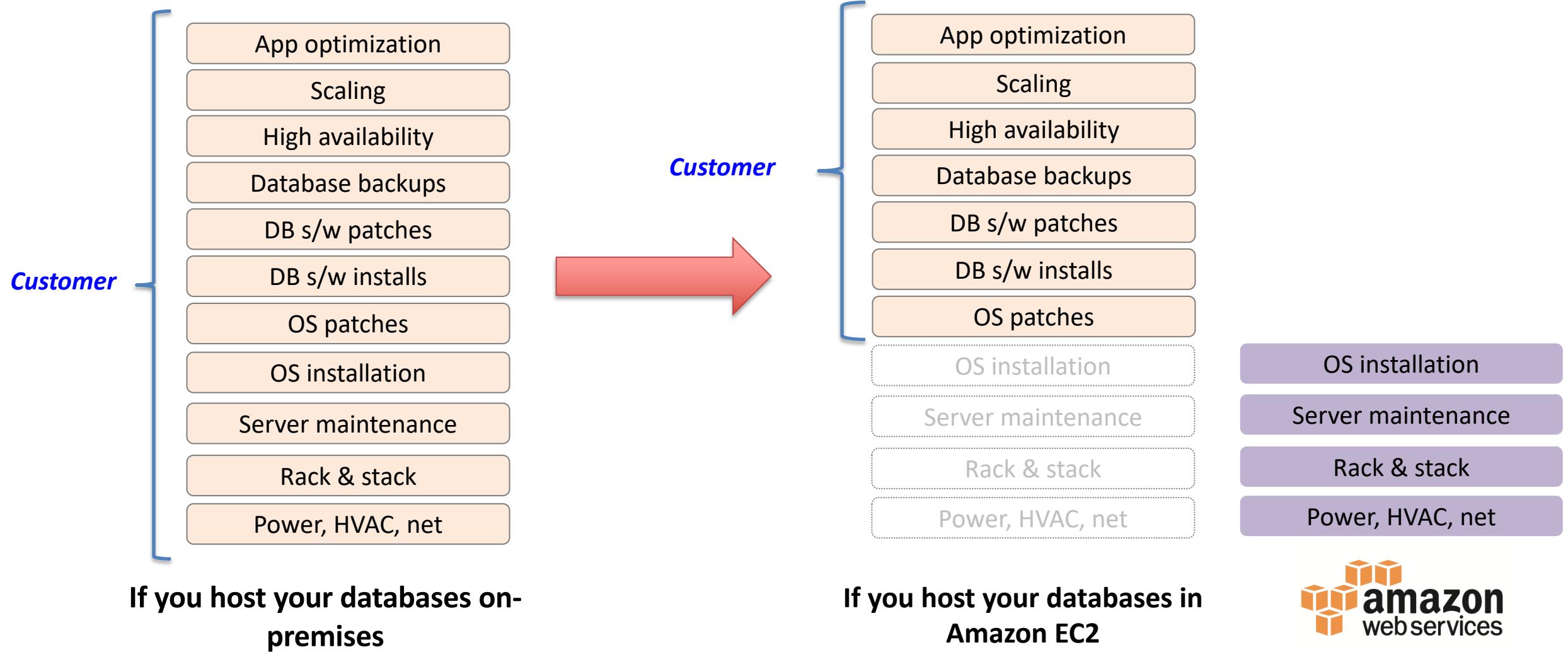
- MySQL
- MariaDB



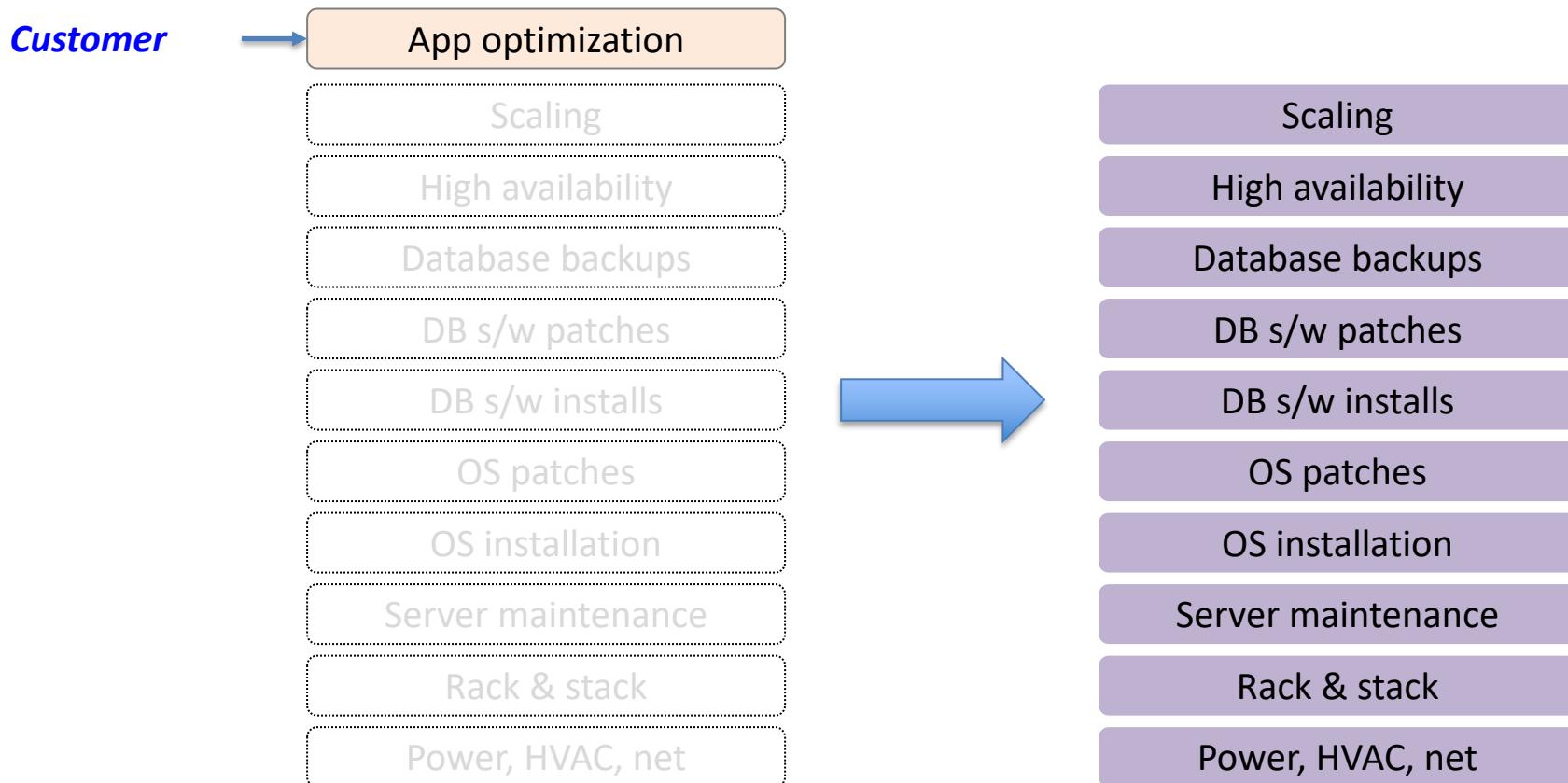
Amazon Aurora Overview

- Highly Optimized version of MySQL and PostgreSQL – up to 5 times faster than standard MySQL
- Drop-in compatible with MySQL 5.6.0
- 99.99% Available.
- Instant crash recovery.
- Database storage scales out automatically by adding 10 GB.
- Up to 15 replicas with ~10-ms replica lag (compared to seconds or minutes with MySQL).

Database Deployment on EC2



Database Deployment on RDS



RDS Pricing

- Compute Node Size
 - Reserved Pricing Available
- Database Storage
 - Snapshots are separately charged.
- Multi-AZ Node and Read Replica Charges.
- Data transfer

Storage Services – S3 and EBS

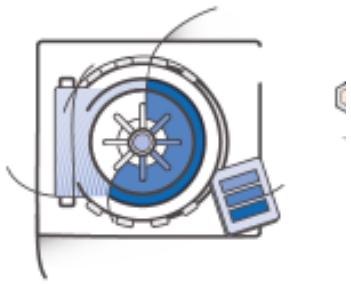
Topics

- Storage Services - Block Storage v/s Object Storage
- Amazon S3 and various storage classes
- Amazon S3 features and Pricing
- Amazon EBS
- EBS Storage types
- EBS Snapshots

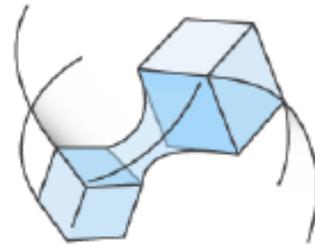
Getting Started: Storage Service



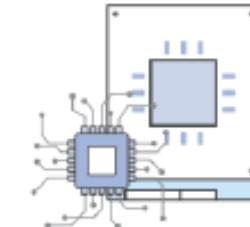
Amazon
S3



Amazon
Glacier



Amazon
EBS



Amazon EC2
Instance
storage

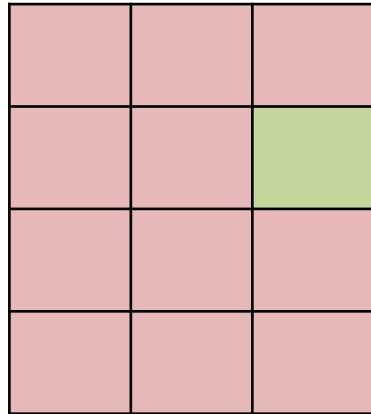
Object

Block

Block vs. Object Storage

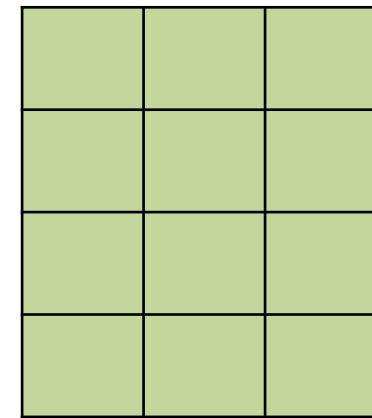


What if you want to change one character in a 1-GB file?



Block Storage

Change one block (piece of the file)
that contains the character



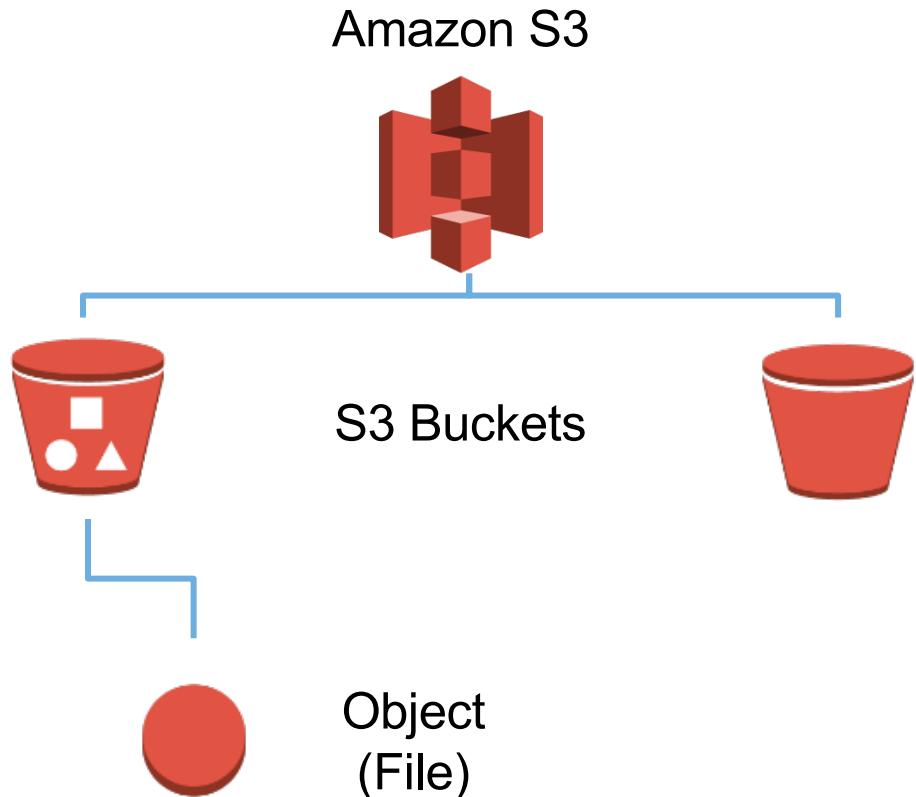
Object Storage

Entire file must be updated

Simple Storage Service (S3)

- Low latency and high throughput **Object Storage**
- Single Object Size up to **5TB**
- Designed for **durability of 99.99999999%** of objects
- Designed for **99.99% availability** over a given year
- Supports **SSL encryption** of data in transit and at rest
- Bucket Policies, Access Rules and IAM.
- **Lifecycle management** for automatic migration of objects

Amazon S3 Concepts



- Amazon S3 stores data as objects within buckets
- An object is composed of a file and optionally any metadata that describes that file
- You can have up to 100 buckets in each account
- You can control access to the bucket and its objects

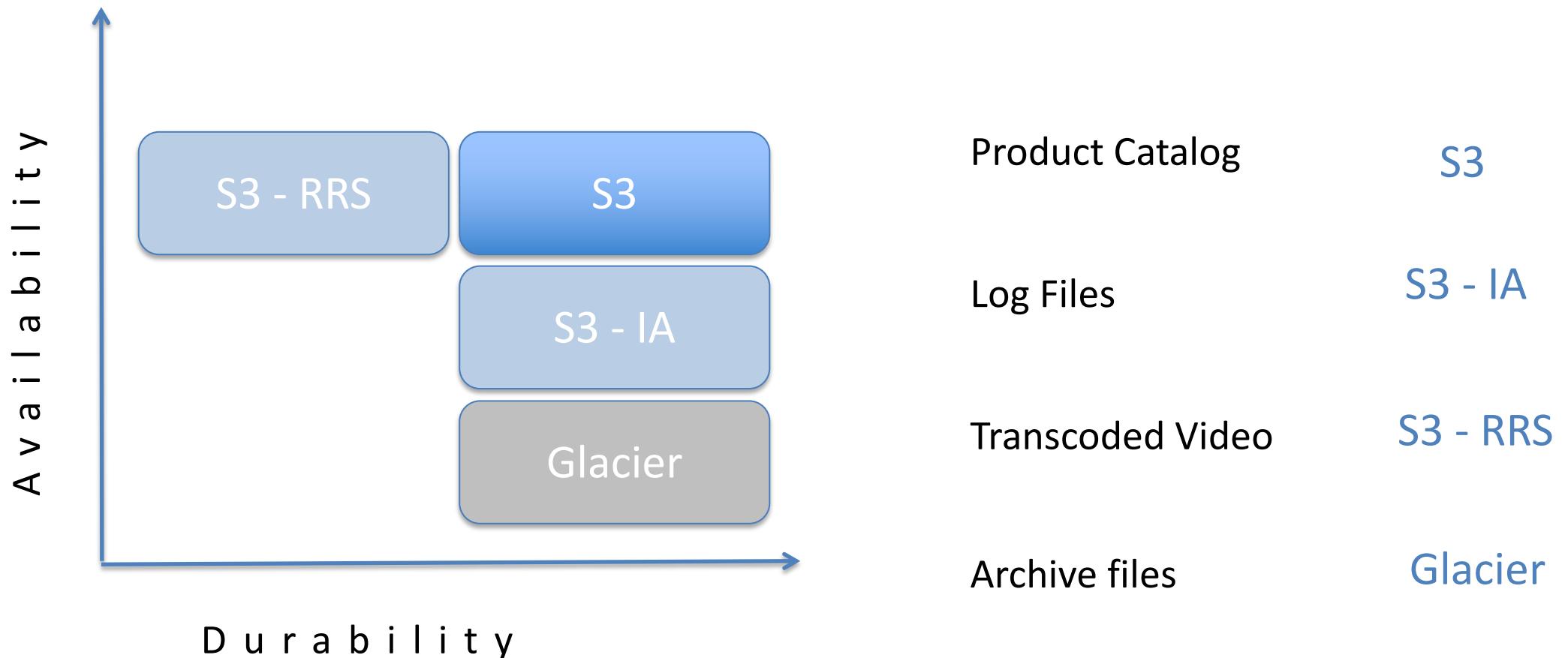
S3 Namespace

- Bucket name has to be **unique** across the entire S3 namespace.
- An object key is the unique identifier for an object within a bucket.
- Folders are just prefixes – internally, it is a flat storage.

<http://example.s3.amazonaws.com/reports/FY2017.html>



S3 Storage Classes



Simple Storage Service (S3)

	Standard	S3 – RRS	S3 – IA	Amazon Glacier
Designed for Durability	99.999999999%	99.99%	99.999999999%	99.999999999%
Cost	\$0.025 per GB	\$0.0175 per GB	\$0.0125 per GB	\$0.005 per GB
Designed for Availability	99.99%	99.99%	99.9%	N/A
Availability SLA	99.9%	99.9%	99%	N/A
Minimum Object Size	N/A	N/A	128KB	N/A
Minimum Storage Duration	N/A	N/A	30 days	90 days

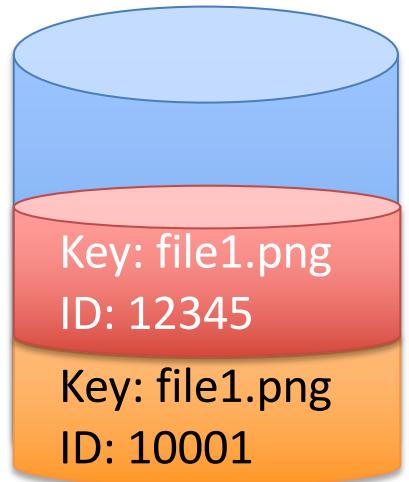
Amazon S3 Object Lifecycle

Lifecycle management defines how Amazon S3 manages objects during their lifetime. Some objects that you store in an Amazon S3 bucket might have a well-defined lifecycle:

- Log files
- Archive documents
- Digital media archives
- Financial and healthcare records
- Raw genomics sequence data
- Long-term database backups
- Data that must be retained for regulatory compliance

Amazon S3 Versioning

- Protects from accidental overwrites and deletes with no performance penalty.
- Generates a new version with every upload.
- Allows easily retrieval of deleted objects or roll back to previous versions.
- Three states of an Amazon S3 bucket
 - Un-versioned (default)
 - Versioning-enabled
 - Versioning-suspended



Versioning Enabled

Amazon S3 Cross Region Replication

- Cross-region replication (CRR) makes it simple to replicate new objects into any other AWS Region for reduced latency, compliance, disaster recovery, and a number of other use cases.
- CRR replicates every object uploaded to your source bucket to a destination bucket in a different AWS region that you choose.
- The metadata, ACLs, and object tags associated with the object are also part of the replication.

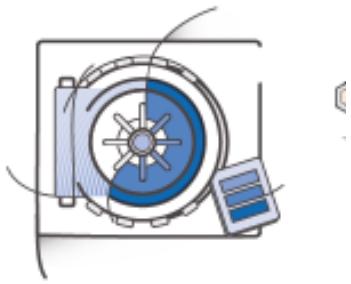
Amazon S3 Pricing

- **Actual Storage Utilized**
 - Per GB/month – prorated daily
 - Varies for different storage classes.
- **Number of Requests**
 - HTTP GET – \$0.4 per million requests
 - HTTP POST - \$5 per million requests
 - **S3 IA** - slightly higher request pricing. (\$1 and \$10)
- **Data Out charges**
 - S3 to EC2 with in same region is free
 - S3 to Cloudfront is free.

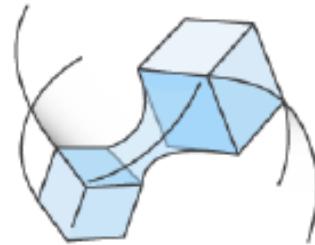
Getting Started: Storage Service



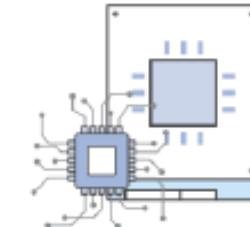
Amazon
S3



Amazon
Glacier



Amazon
EBS

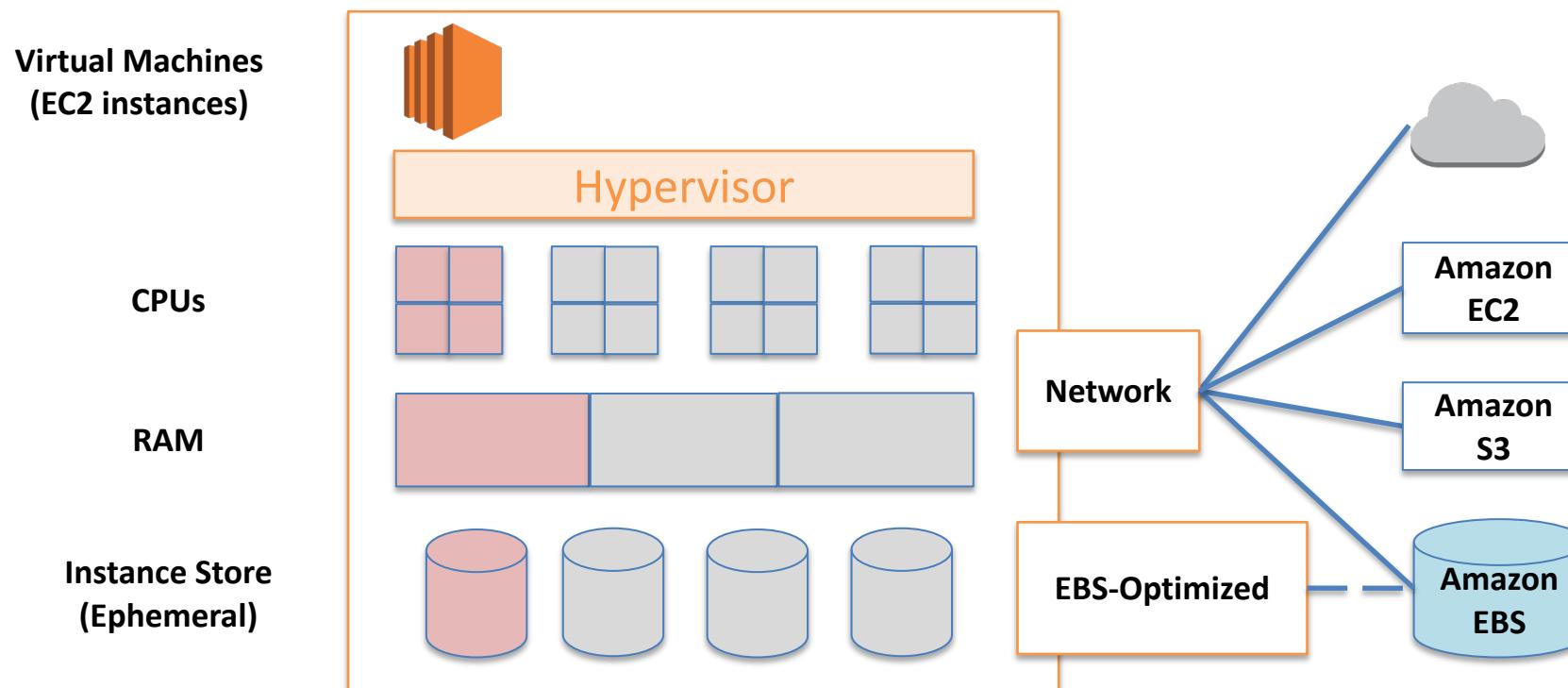


Amazon EC2
Instance
storage

Object

Block

EC2 Internal Architecture



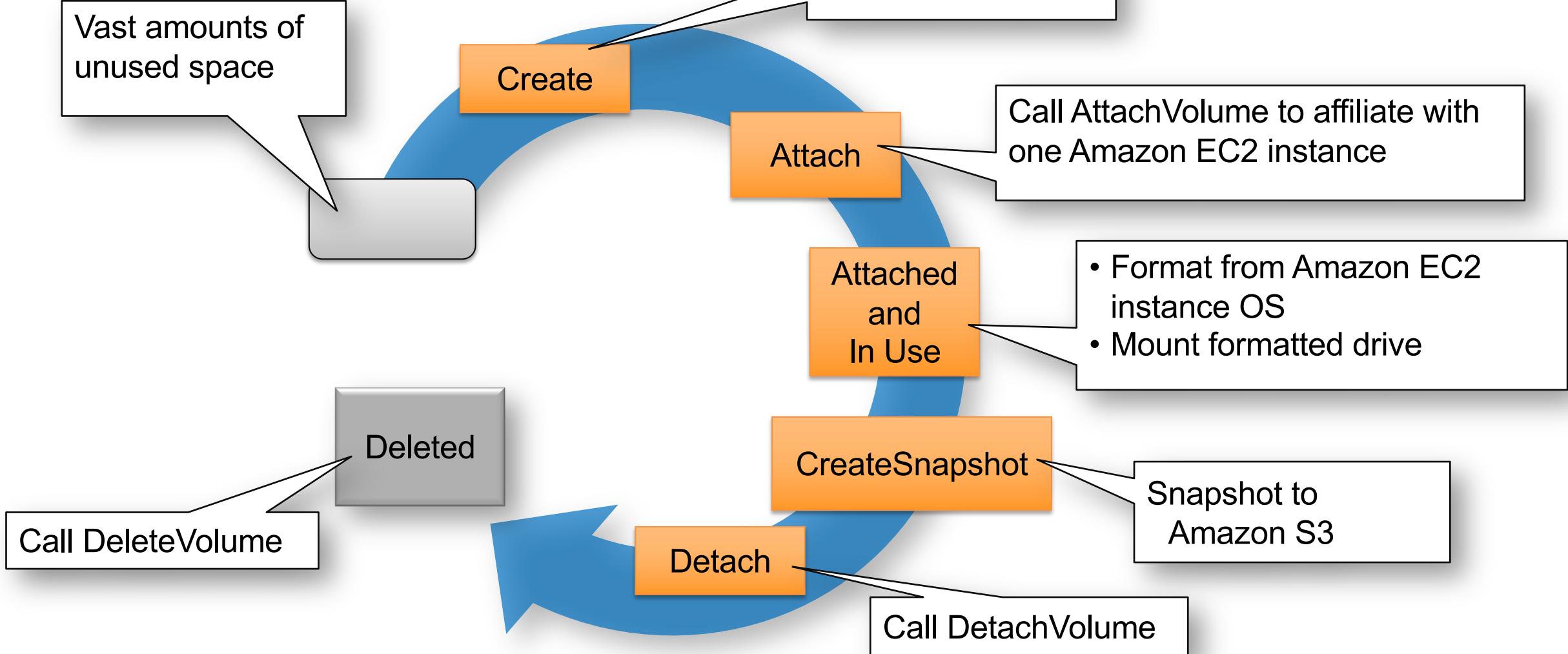
EC2 Instance Storage

- Local, complimentary - **Direct Attached block storage.**
- Availability, number of disks, and size is based on EC2 instance type.
- Storage optimized instances for up to **365,000 Read IOPS and 315,000 Write IOPS.**
- SSD or magnetic.
- **No persistence** - All data is automatically deleted when an EC2 instance stops, fails or is terminated.

EBS – Elastic Block Storage

- **Persistent block level storage**
- **Independent from EC2 lifecycle.**
- **SAN Storage** in cloud - volumes offering consistent and low-latency performance.
- Automatically replicated within its Availability Zone
- Snapshots stored durably in Amazon S3

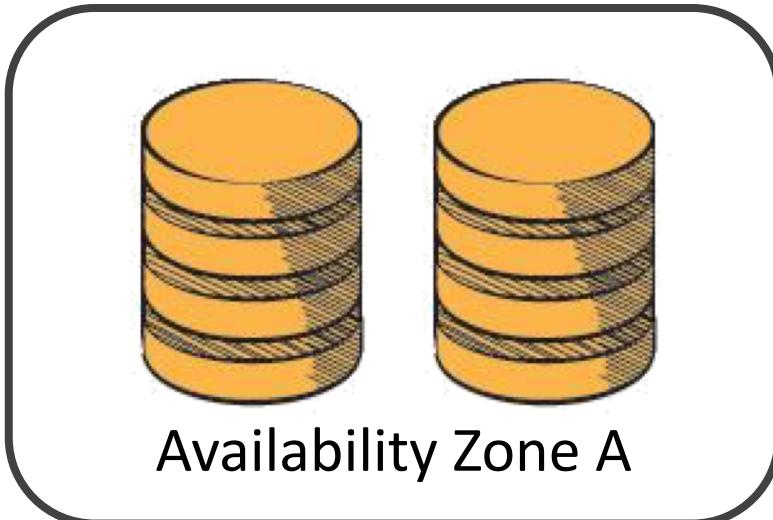
Amazon EBS Life Cycle



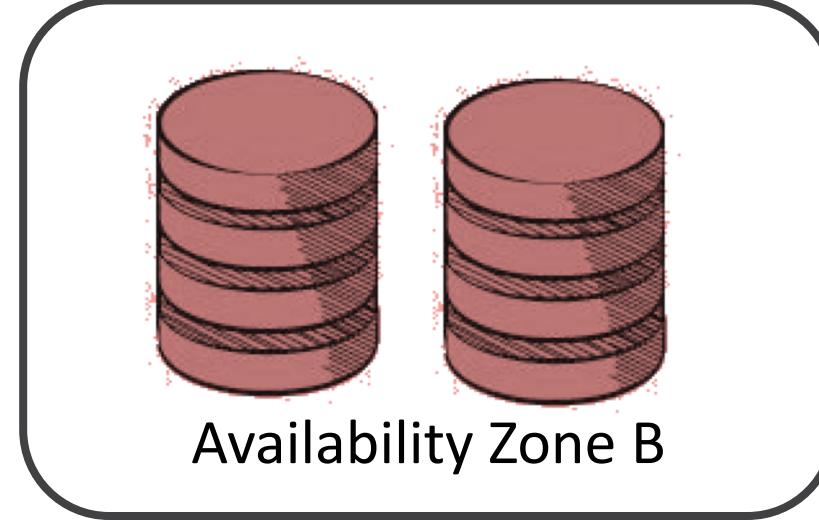
Amazon EBS Scope

Amazon EBS Volumes are Single Availability Scoped

EBS Volume 1



EBS Volume 2



Volume data is replicated across multiple servers with a single AZ.

Amazon EBS Use Cases

- OS – Use for boot/root volume, secondary volumes
- Databases – Scales with your performance needs
- Enterprise applications – Provides reliable block storage to run mission-critical applications
- Business continuity – Minimize data loss and recovery time by regularly backing up using EBS Snapshots
- Applications – Install and persist any application

General Purpose (SSD)

- I/O Performance is based on the volume size
 - Consistent baseline performance of **3 IOPS/GB**
- Earn credits at rate of 3 IOPS/GiB/second.
- Spend credits to burst up to 3,000 IOPS.
 - Maximum credit gives 3,000 IOPS for at least 30 minutes.
 - Only 0.3% historically have run out of credit.
- Initial credit balance sufficient to sustain 3000 IOPS for 30 minutes.
 - Provides a fast initial boot cycle for boot volumes.

General Purpose (SSD)

Volume size (GiB)	Baseline Performance 3*Volume size (IOPS)	Maximum burst duration @ 3000 IOPS (minutes)
1	3	30
100	300	33
250	750	40
500	1500	60
750	2250	120
1000	3000	unlimited
2000	6000	unlimited

* Bursting and I/O credits are only relevant to volumes under 1,000 GiB, where burst performance exceeds baseline performance.

EBS Provisioned IOPS (PIOPS)

- Use when consistent performance is required – for example **Relational Database.**
 - Users can specify performance in terms of IOPS.
 - PIOPS volumes are designed to consistently perform within 10% of the specified IOPS 99.9% of time.
 - Charges for both storage capacity and speed apply.

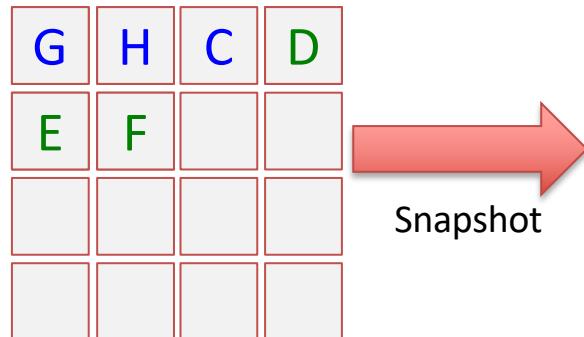
Note: Use with Amazon EBS–optimized instances, which provide dedicated network piping with the EBS.

PIOPS – IOPS and Throughput limits

Provisioned IOPS	Max Throughput (MBps)	Max Random 16 KiB I/O per second	Max Random 256 KiB I/O per second
1250	320	1250	1250
10000	320	10000	1250
20000	320	20000	1250

Amazon EBS Incremental Snapshots

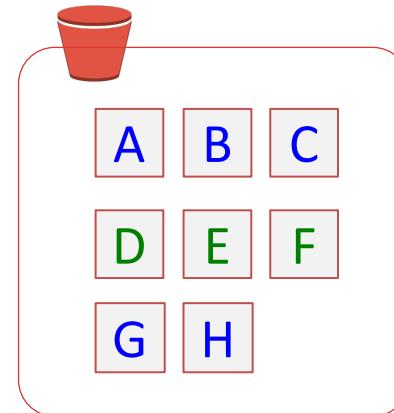
- Snapshots copy block-level data to Amazon S3.
- Snapshots are based on deltas.
 - Only the changes from previous snapshots is copied to S3.



Snapshot 1 = ABC

Snapshot 2 = ABCDEF

Snapshot 3 = GHCDEF



Amazon EBS Pricing

- **Storage Provisioned**
 - Per GB/month – prorated daily
 - Price varies for Magnetic, GP2, PIOPS types of volume.
- **IOPS**
 - GP2, HDD – NO separate IO charges.
 - PIOPS – IO is charged separately.
- **Snapshot**
 - \$0.05 per GB-month of data stored on S3.

Amazon EBS and Amazon S3

	Amazon EBS	Amazon S3
Paradigm	Block storage with file system	Object store
Performance	Very fast	Fast
Redundancy	Across multiple servers in an Availability Zone	Across multiple facilities in a Region
Security	EBS Encryption – Data volumes and Snapshots	Encryption
Access from the Internet?	No (1)	Yes (2)
Typical use case	It is a disk drive	Online storage

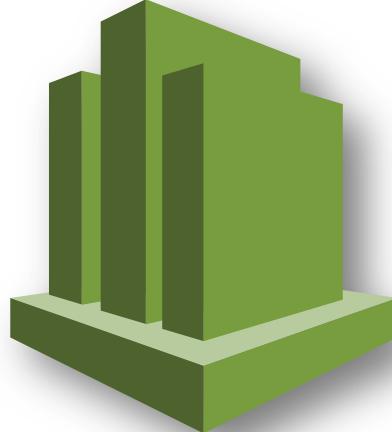
- (1) Accessible from the Internet if mounted to server and set up as FTP, etc.
(2) Only with proper credentials, unless ACLs are world-readable

CloudWatch Monitoring

Topics

- What is CloudWatch?
- CloudWatch Metrics
- CloudWatch Alarms
- CloudWatch Logs
- Pricing

Amazon CloudWatch



Amazon CloudWatch:

- Monitors your instances, collects and processes raw data into readable metrics.
- Sends Alert notifications and triggers auto scaling actions based on specific metrics.

CloudWatch Metrics Examples

CloudWatch Metrics by Category

Your CloudWatch metric summary has loaded. Total metrics: 97

EBS Metrics: 24 EC2 Metrics: 38 S3 Metrics: 18 SNS Metrics: 3 SQS Metrics: 14
Per-Volume Metrics: 24 Per-Instance Metrics: 38 Storage Metrics: 18 Topic Metrics: 3 Queue Metrics: 14

Browse Metrics X

Showing all results (100) for cpu. For more results expand your search to All EC2 Metrics.

Select All | Clear

EC2 > Per-Instance Metrics

InstanceId	InstanceName	Metric Name
<input checked="" type="checkbox"/> i-2be9e804		CPUUtilization
<input type="checkbox"/> i-2ce7e603		CPUUtilization
<input type="checkbox"/> i-5b210d23		CPUUtilization
<input type="checkbox"/> i-8da34f60		CPCreditBalance

Title: CPUUtilization

Average 5 Minutes

CPUUtilization

From: 12 hours ago To: 0 hours ago

Zoom: 1h | 3h | 6h | 12h | 1d | 3d | 1w | 2w

Add to Dashboard

Actions

Copy URL Create Alarm

Time Range

Relative Absolute UTC (GMT)

Left Y-axis

Monitor AWS Resources with CloudWatch

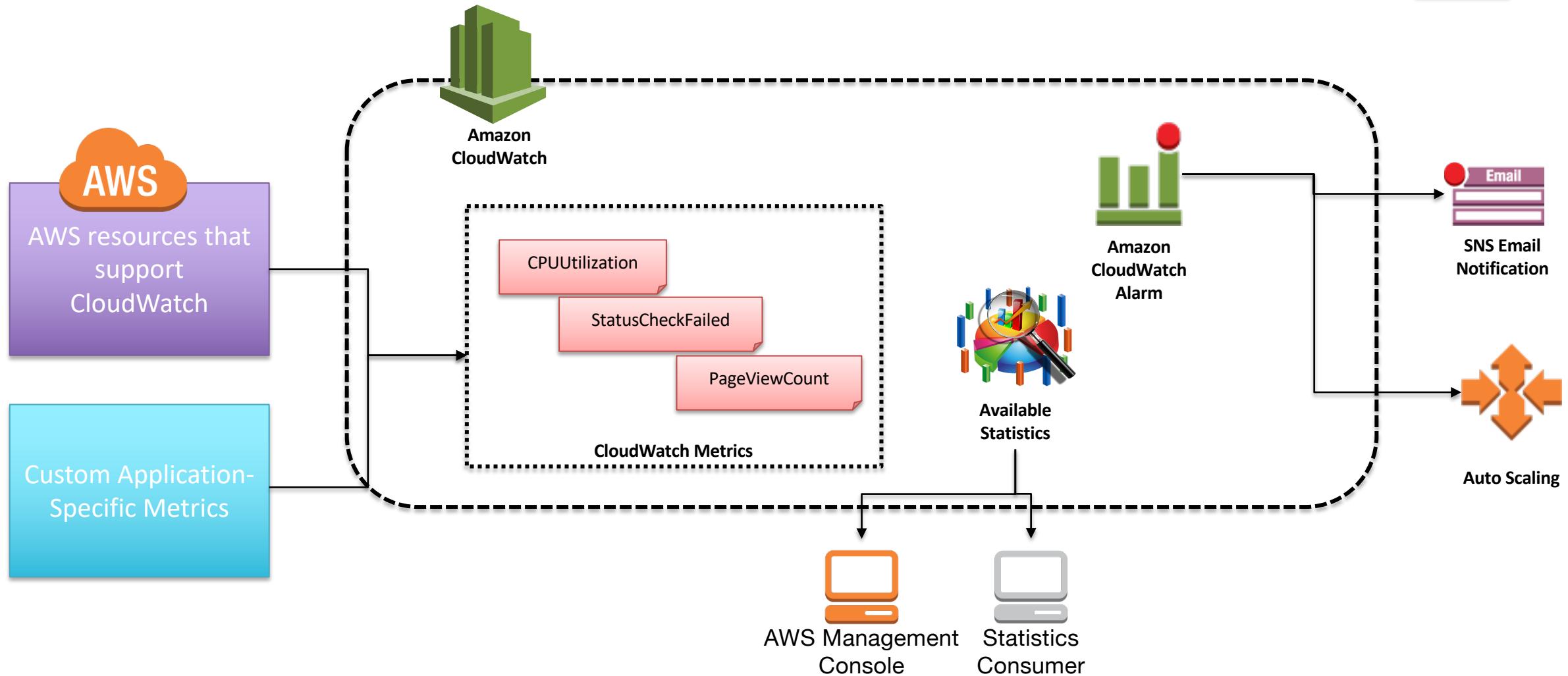
Amazon CloudWatch offers:

- A distributed statistics gathering system that collects and tracks metrics.
- Metrics that are seamlessly collected at the hypervisor level by default.
- **Ability Create and use custom metrics** with the data generated by your own applications and services.

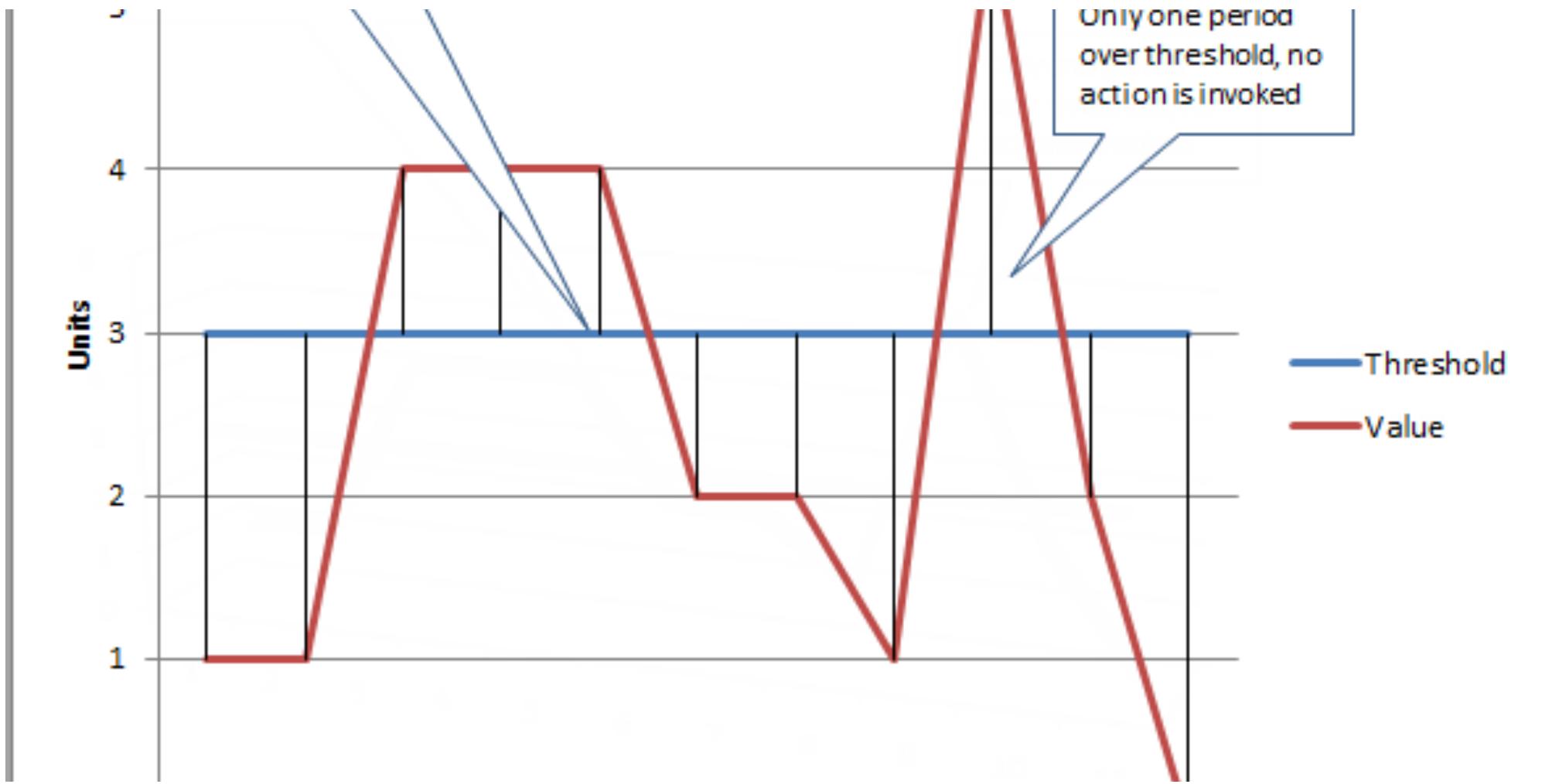
Examples:

- The time web pages take to load
- Request error rates
- Number of simultaneous processes or threads

CloudWatch Architecture



CloudWatch Alarms



CloudWatch Alarm Examples

Amazon EC2



If CPU utilization is > 60% for 5 minutes...

Amazon RDS



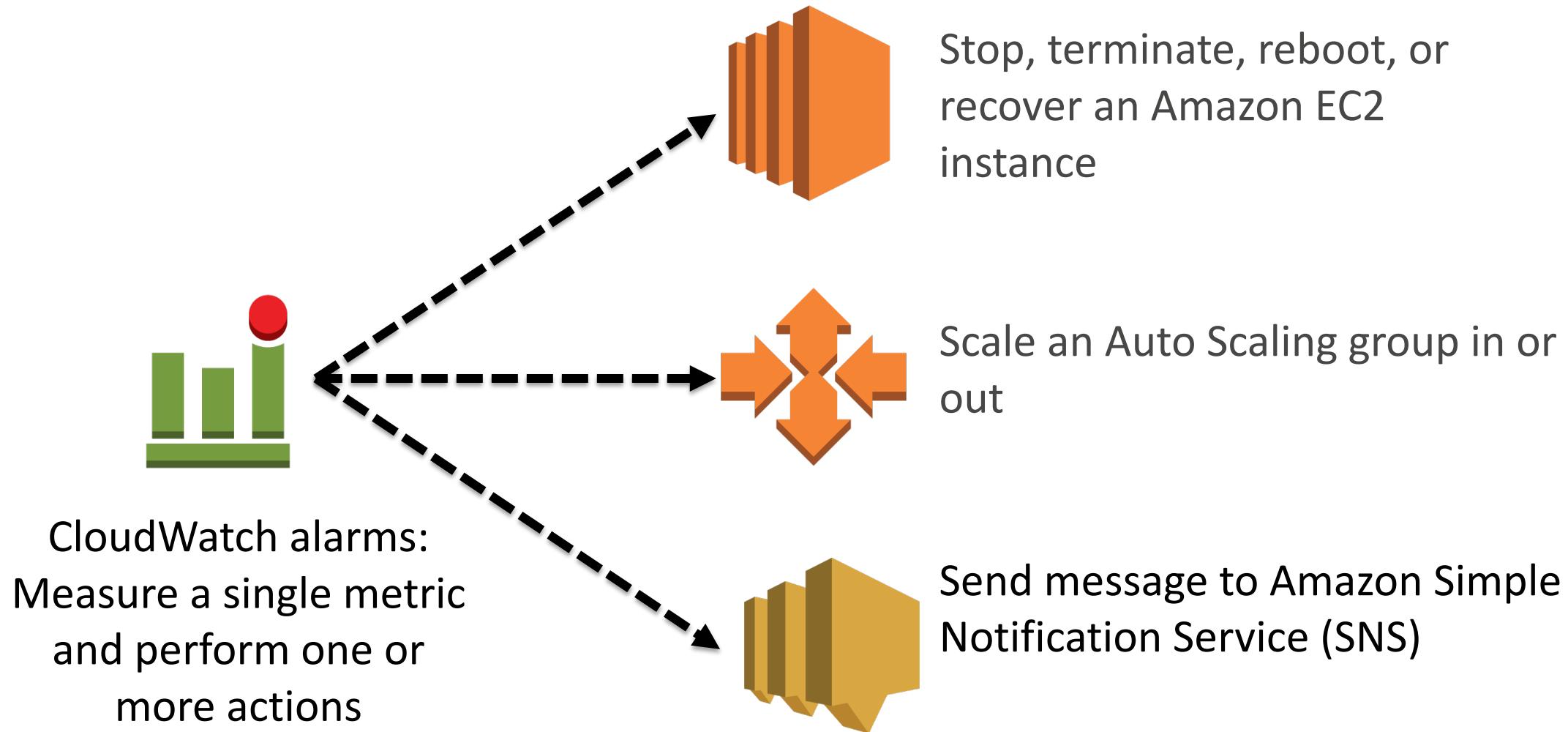
If number of simultaneous connections is > 10 for one minute...

Amazon ELB

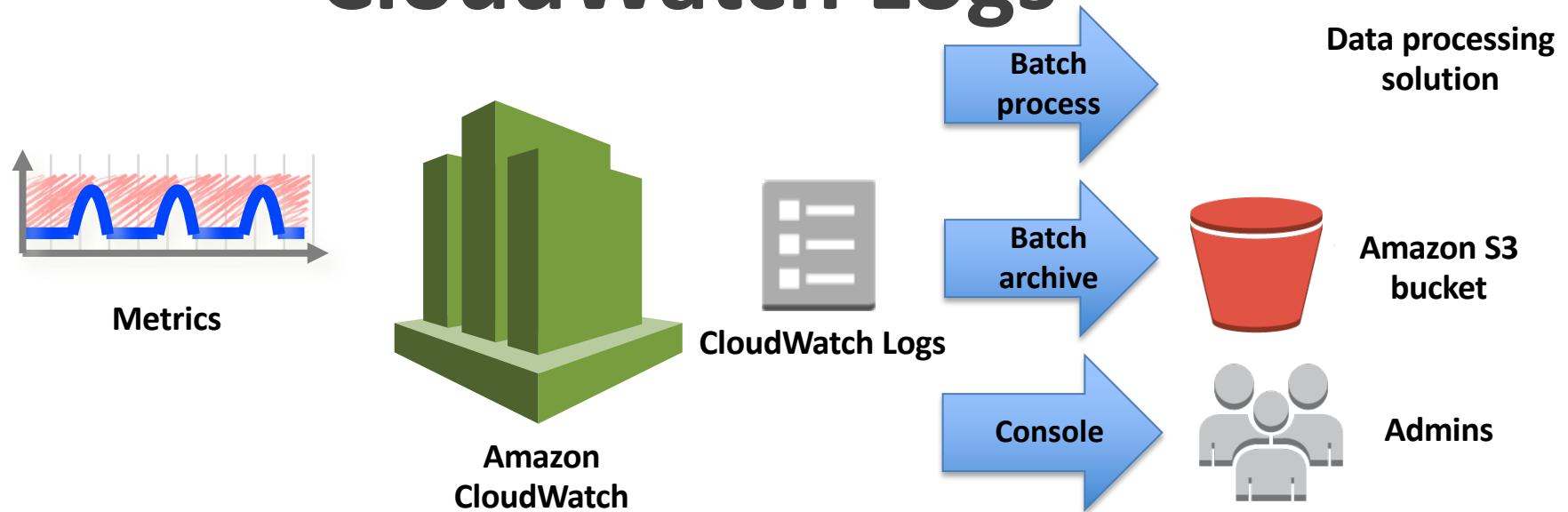


If number of healthy hosts is < 5 for 10 minutes...

CloudWatch Alarms and Actions



CloudWatch Logs



- Your metrics can be stored durably in CloudWatch as CloudWatch Logs.
- Admins and other parties can review CloudWatch logs directly in the AWS Management Console.
- Logs can be stored in Amazon S3, to be accessed by other services or another user.
- Logs can be streamed in real time to data-processing solutions like Amazon Kinesis Streams or AWS Lambda.

CloudWatch Pricing

- Basic Monitoring is available for free
 - 5 Min Interval
- Detailed Monitoring – 1 Min Interval
 - \$2.10 per instance per month.
- Custom Metric
 - \$0.30 per metric per month
- CloudWatch Logs
 - \$0.50 per GB ingested
 - \$0.03 per GB archived per month

Horizontal Scaling using AutoScaling

Topics

- Vertical v/s Horizontal Scaling
- Elastic Load Balancer
- Auto-Scaling
- Auto-Scaling Example

Vertical v/s Horizontal Scaling

- **Vertical scaling**
Scale up and down
- Change in the **specifications** of instances (more CPU, memory, etc.)
- **Horizontal scaling**
Scale in and out
- Change in the **number of instances**
(Add and remove instances as needed)

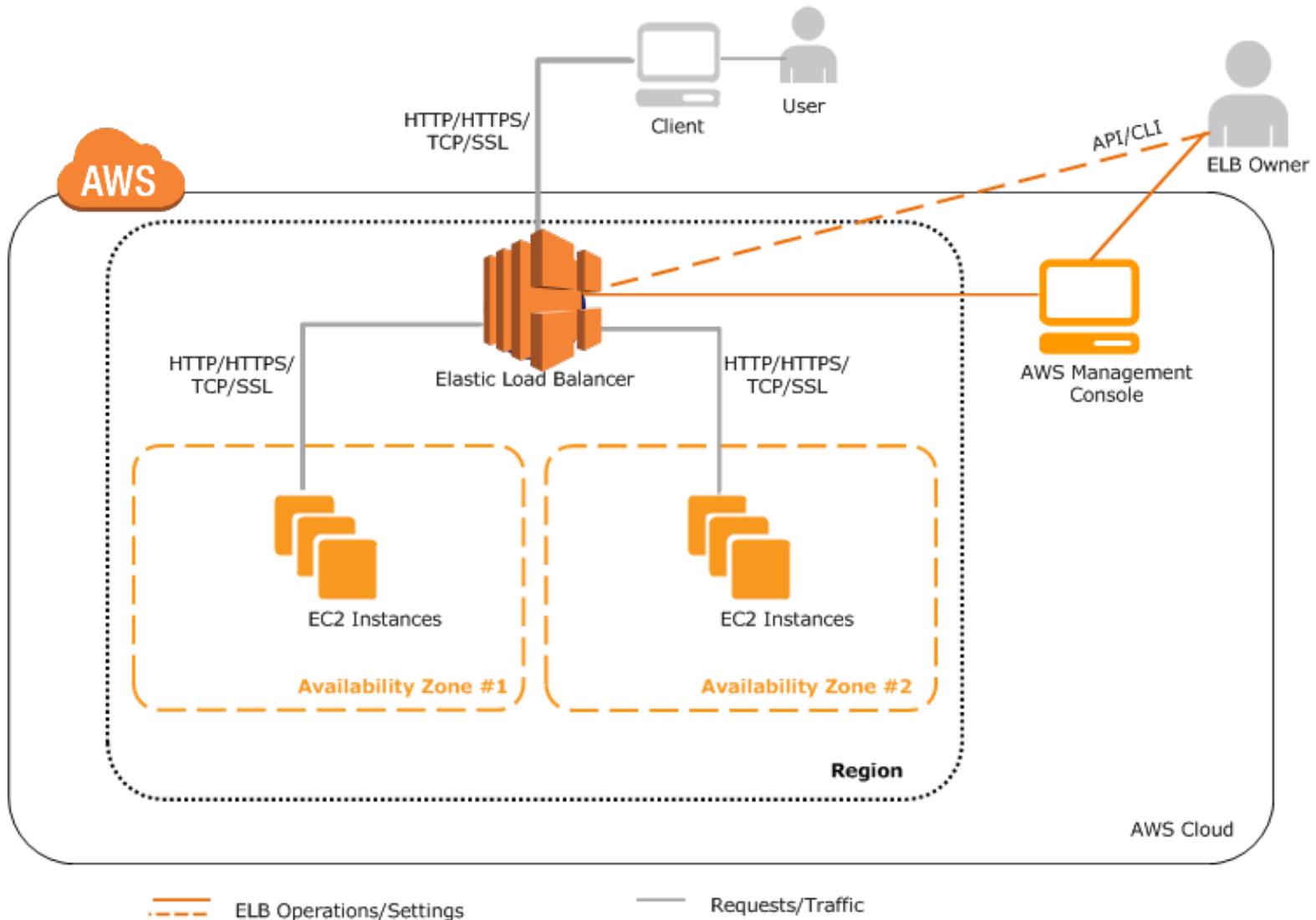


Elastic Load Balancing (ELB)



A managed load balancing service that [distributes incoming application traffic](#) across multiple Amazon EC2 instances.

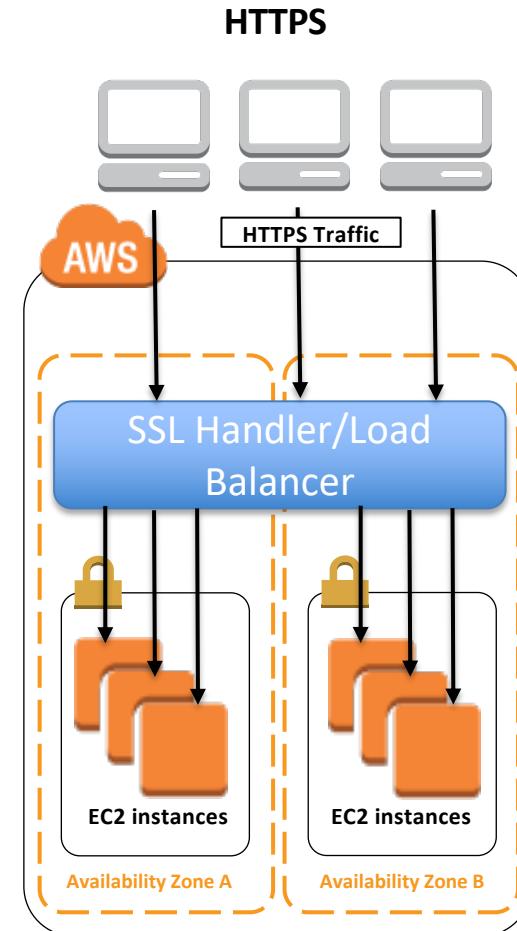
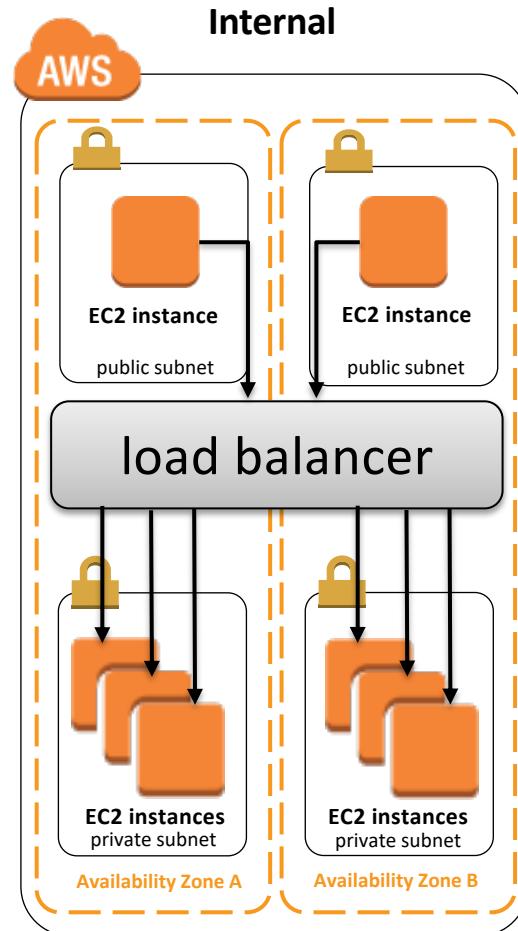
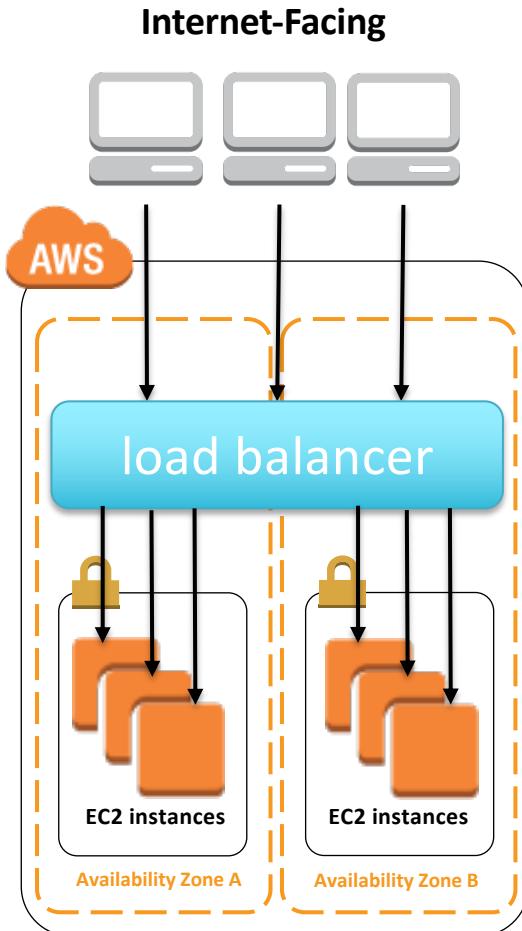
ELB Example



Elastic Load Balancer

- Distributes load between instances.
- Recognizes and responds to unhealthy instances.
- Can be public or internal-facing.
 - Uses HTTP, HTTPS, TCP, and SSL (secure TCP) protocols.
 - Each load balancer is given a public DNS name.
 - **Internet-facing** load balancers have DNS names which publicly resolve to the public IP addresses of the load balancer's nodes.
 - **Internal** load balancers have DNS names which publicly resolve to the private IP addresses of the load balancer's nodes.

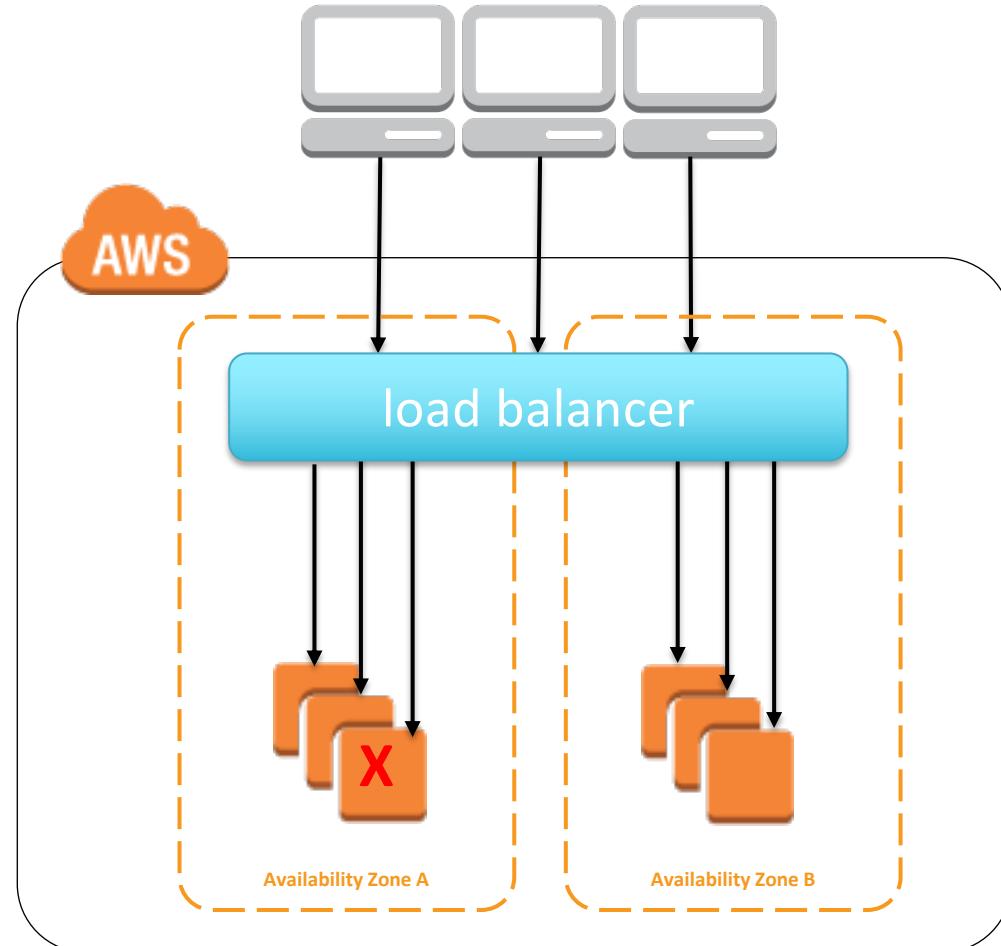
Load Balancer Types



ELB Features

- Health checks
- Cross-zone load balancing
- Proxy Protocol
- Sticky sessions
- Connection draining

How It Works

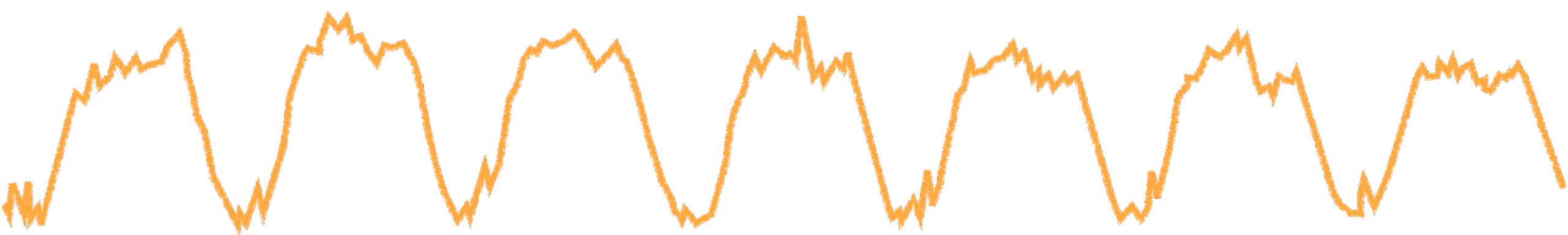


Event Driven Scaling

AUTO SCALING

Typical Weekly Traffic at Amazon.com

Provisioned capacity



Sunday

Monday

Tuesday

Wednesday

Thursday

Friday

Saturday

November Traffic to Amazon.com

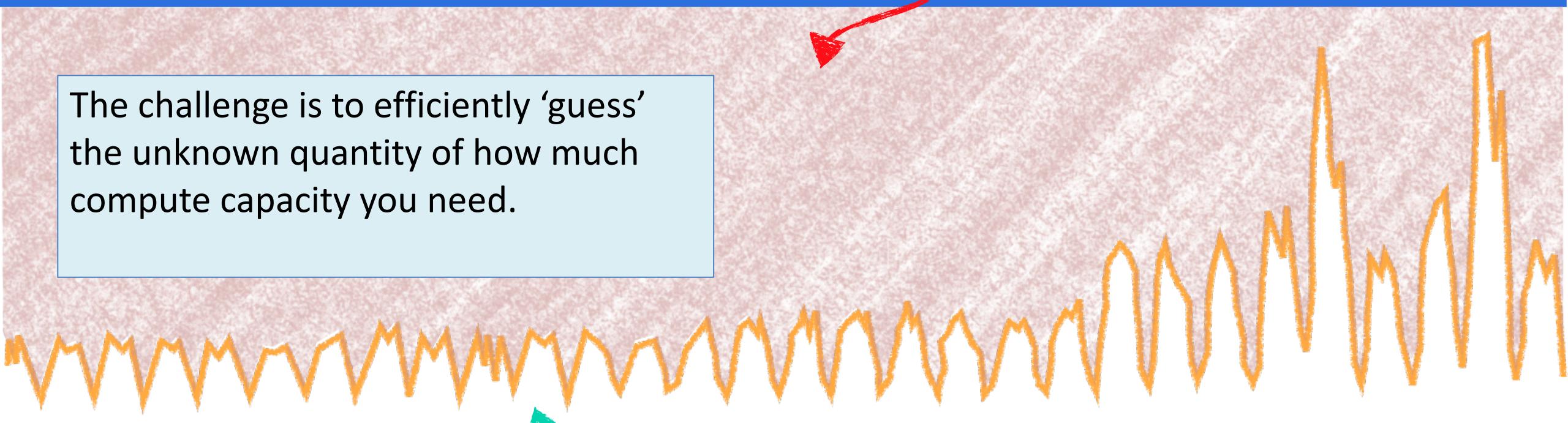
Provisioned capacity

The challenge is to efficiently ‘guess’ the unknown quantity of how much compute capacity you need.

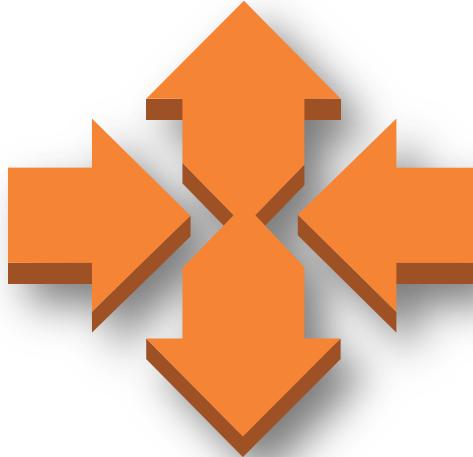
76%

November

24%

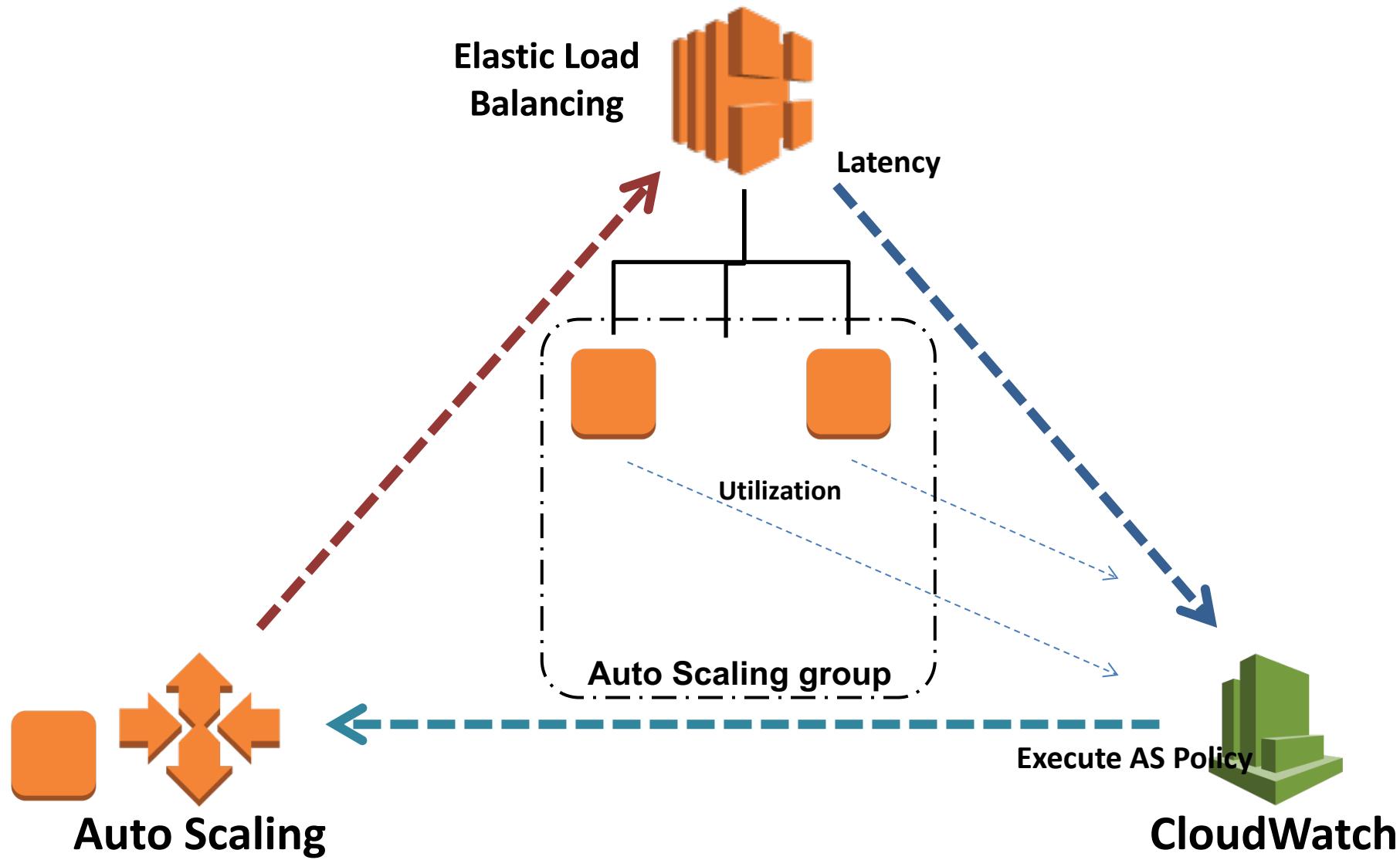


Auto Scaling

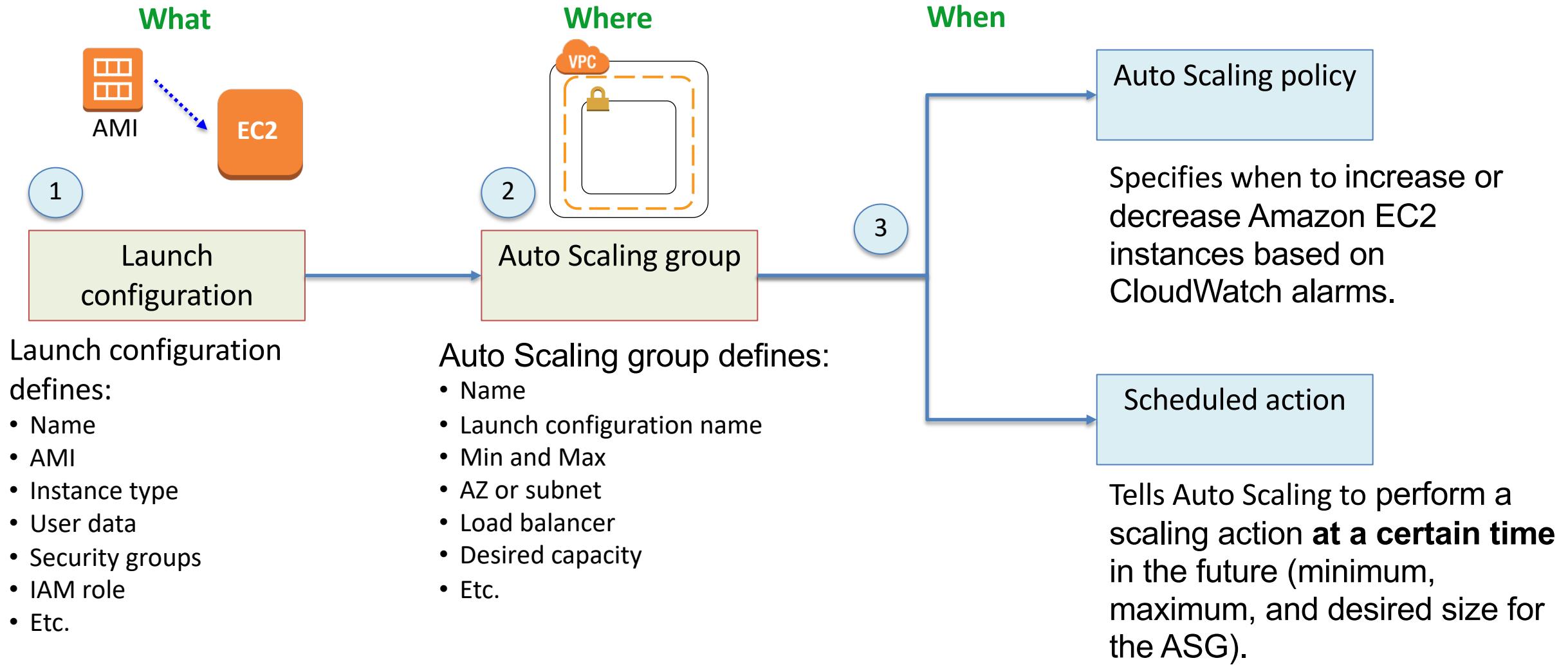


- Launches or terminates instances based on specified conditions.
- Automatically registers new instances with load balancers when specified.
- Can launch across Availability Zones.

How it Works



How to Configure AutoScaling

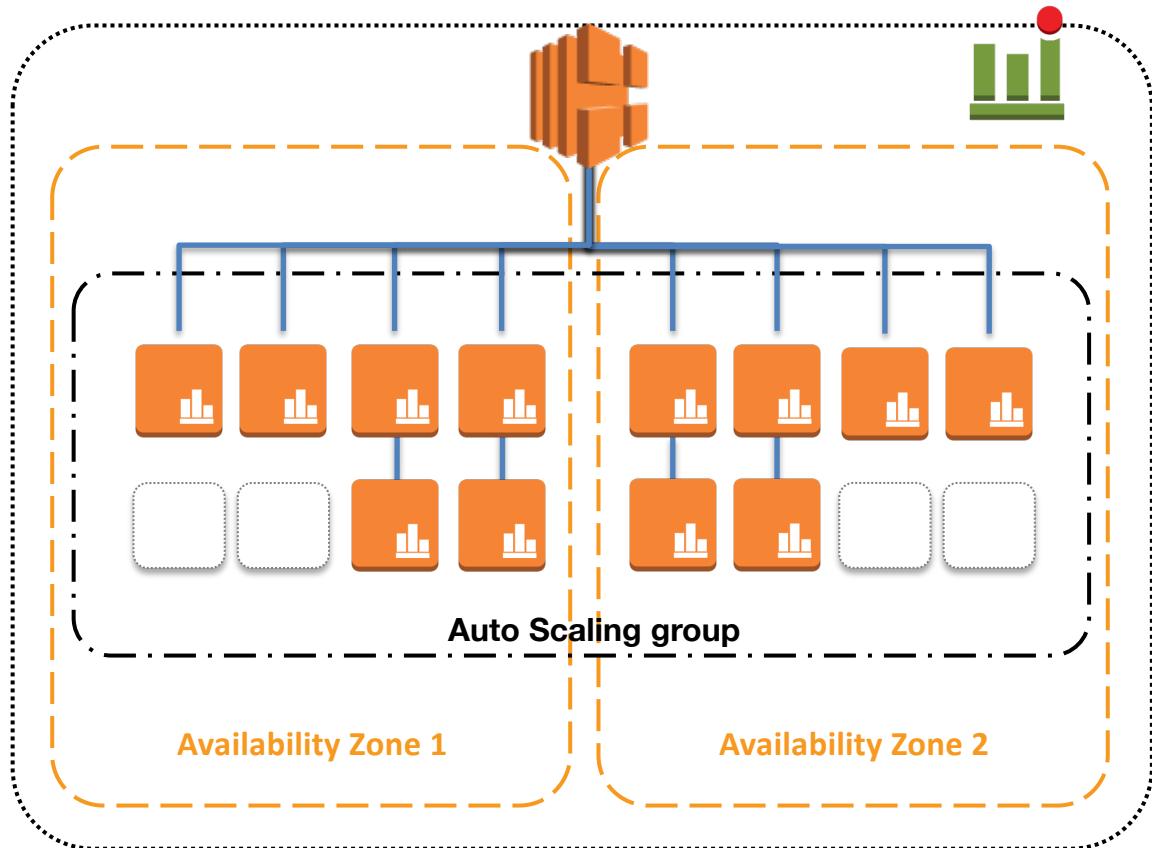


Auto Scaling Example

Scenario:

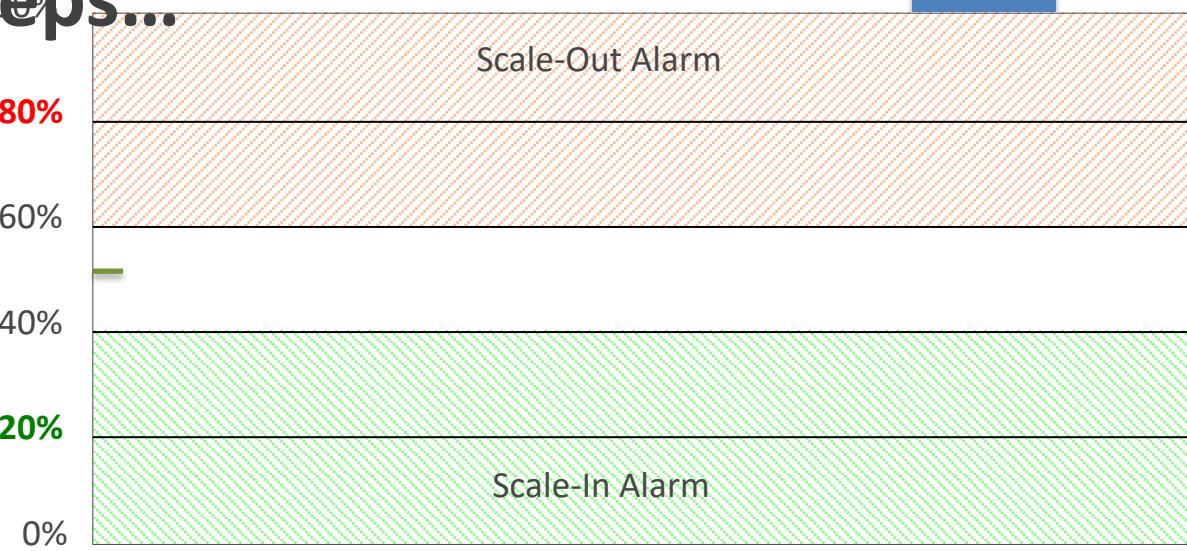
- Auto Scaling group:
 - Minimum = 2
 - Maximum = **12**
- Auto Scaling policy:
 - When CPU utilization is greater than 60%
 - Add 100% of group = **double the capacity**

CPU utilization triggers the alarm: capacity is doubled until CPU utilization drops below 60% or max capacity is reached.



Auto Scaling Steps...

Average CPU Utilization

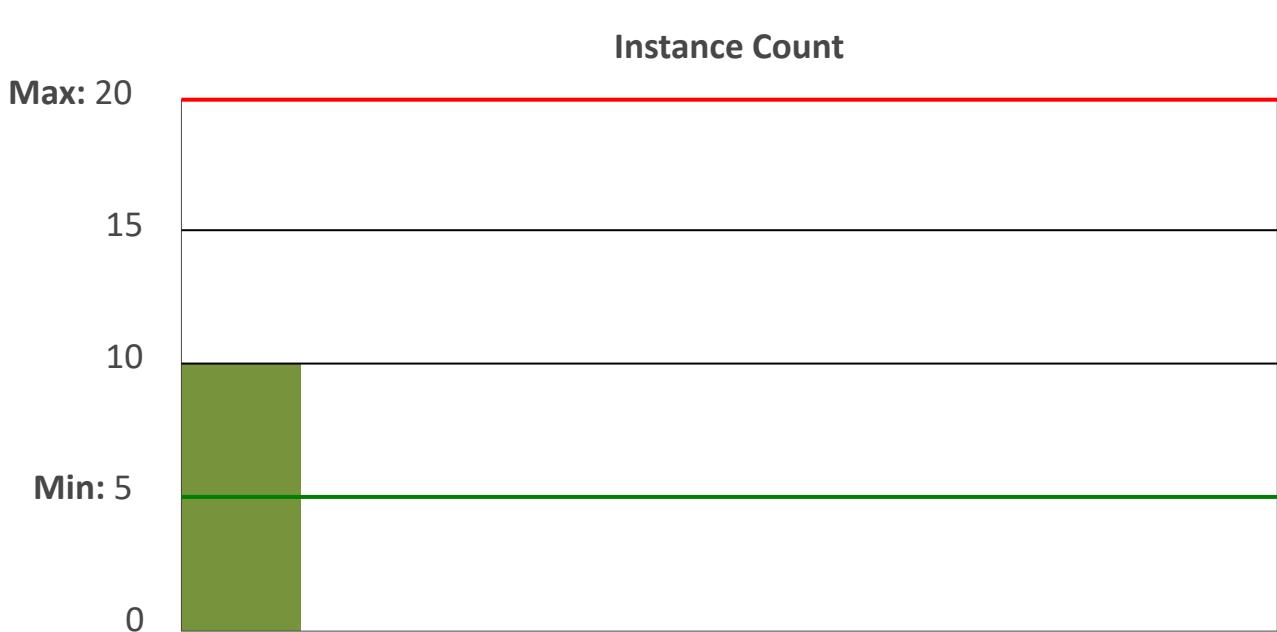


Alarms:

- Add 2 instances when average CPU is 80-100%
- Add 1 instance when average CPU is 60-80%
- Remove 1 instance when average CPU is 20-40%
- Remove 2 instances when average CPU is 0-20%

Limits:

- Minimum: 5 instances
- Maximum: 20 instances



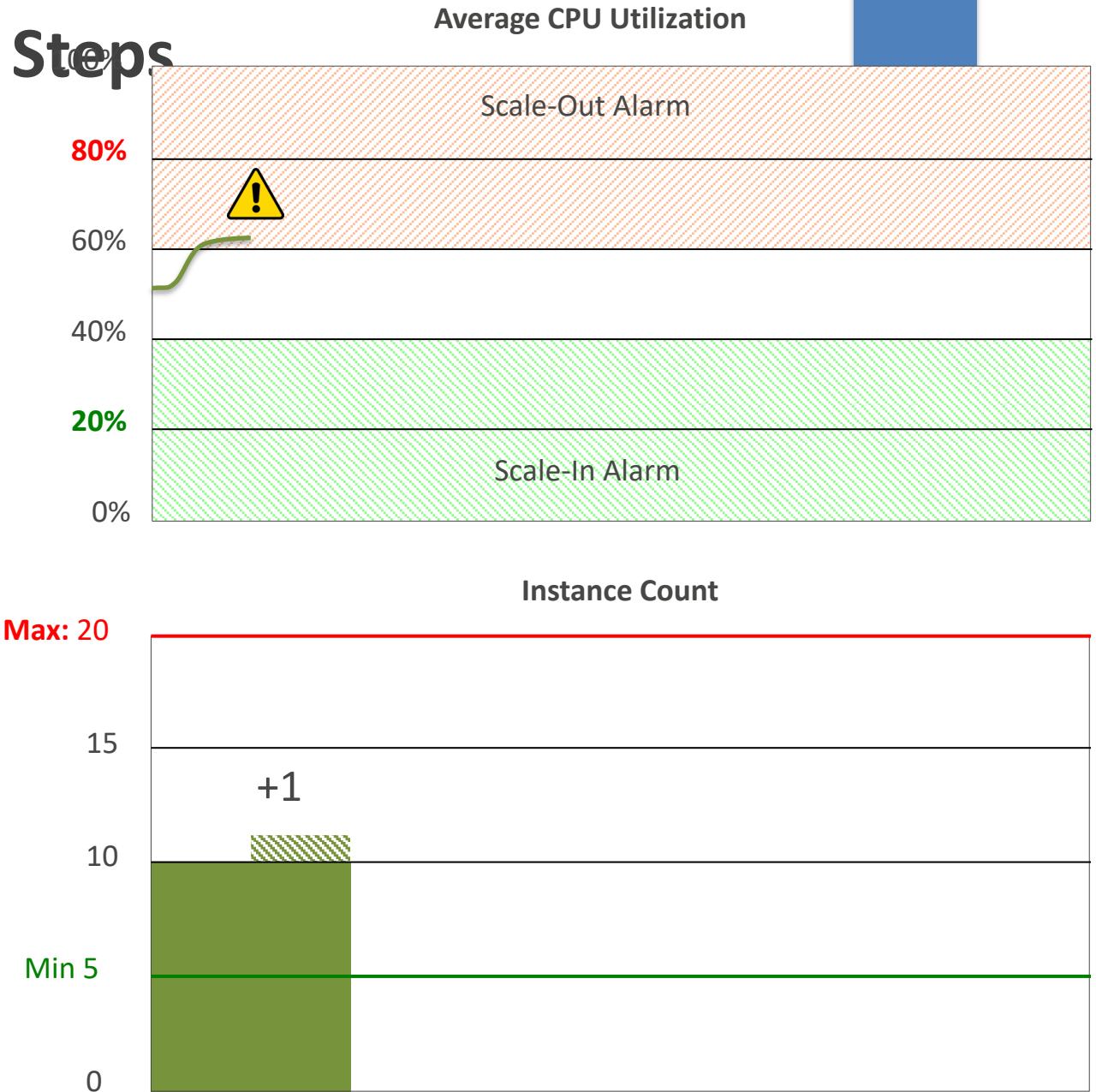
Auto Scaling Steps

As usage increases:

- CPU utilization goes up.

When CPU utilization is 60-80%:

- Scale-out alarm is triggered.
- Add 1 step policy is applied.
- New instance is launched but not added to the aggregated group metrics until after warm up period expires.



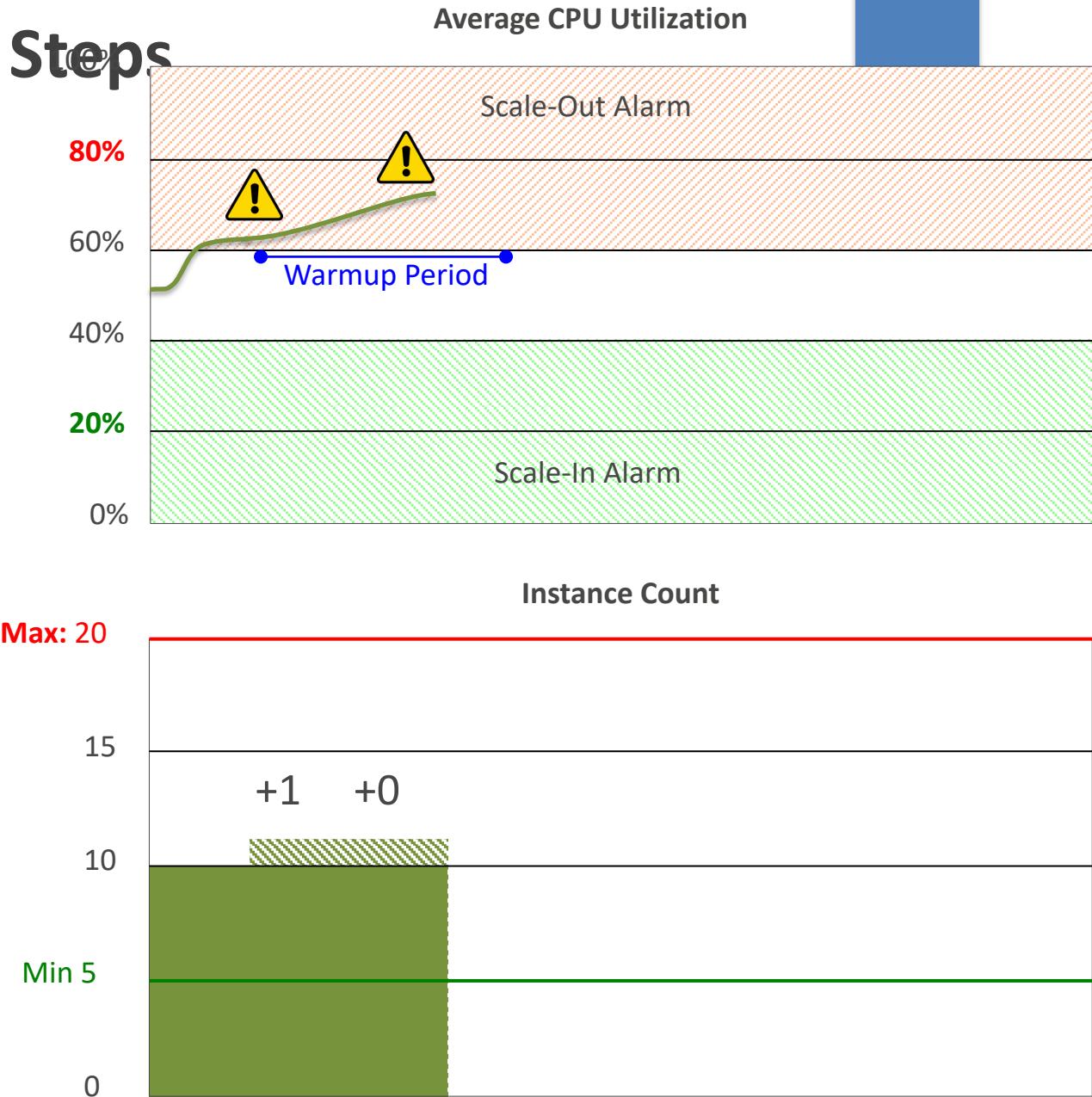
Auto Scaling Steps

As usage increases:

- CPU utilization goes up.

While waiting for new instance:

- CPU utilization remains high.
- Another alarm period is triggered.
- Since current capacity is still 10 during the warmup period, and desired capacity is already 11, **no additional instances are launched**.



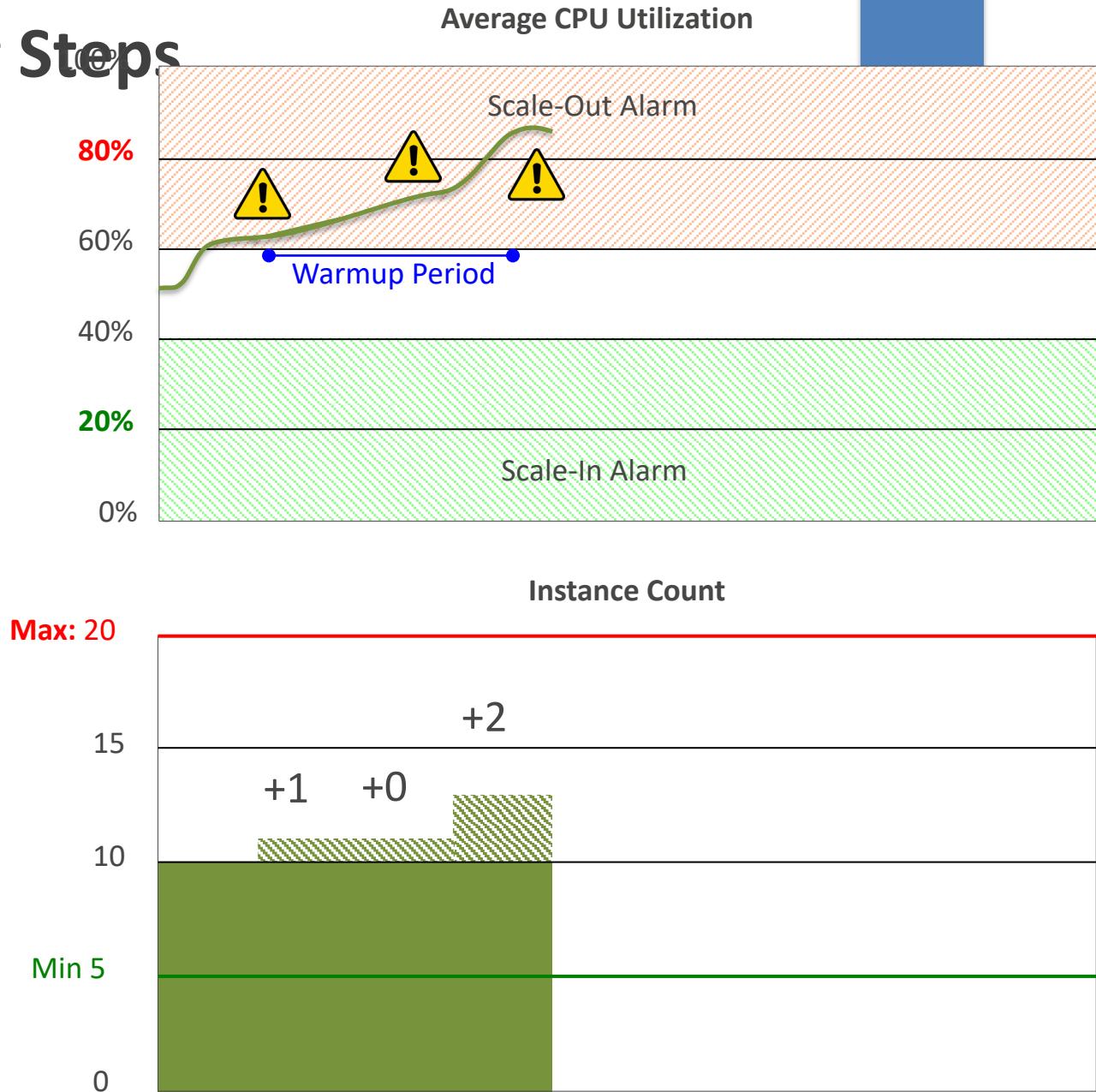
Auto Scaling Steps

As usage increases further:

- CPU utilization goes up.

When CPU utilization is 80-100%:

- Scale-out alarm is triggered.
- Add 2 step policy is applied.
- Since the alarm occurred during a warm up period, two instances are launched less the one instance added during the first alarm.
- Again new instances are not added to aggregated group metrics.



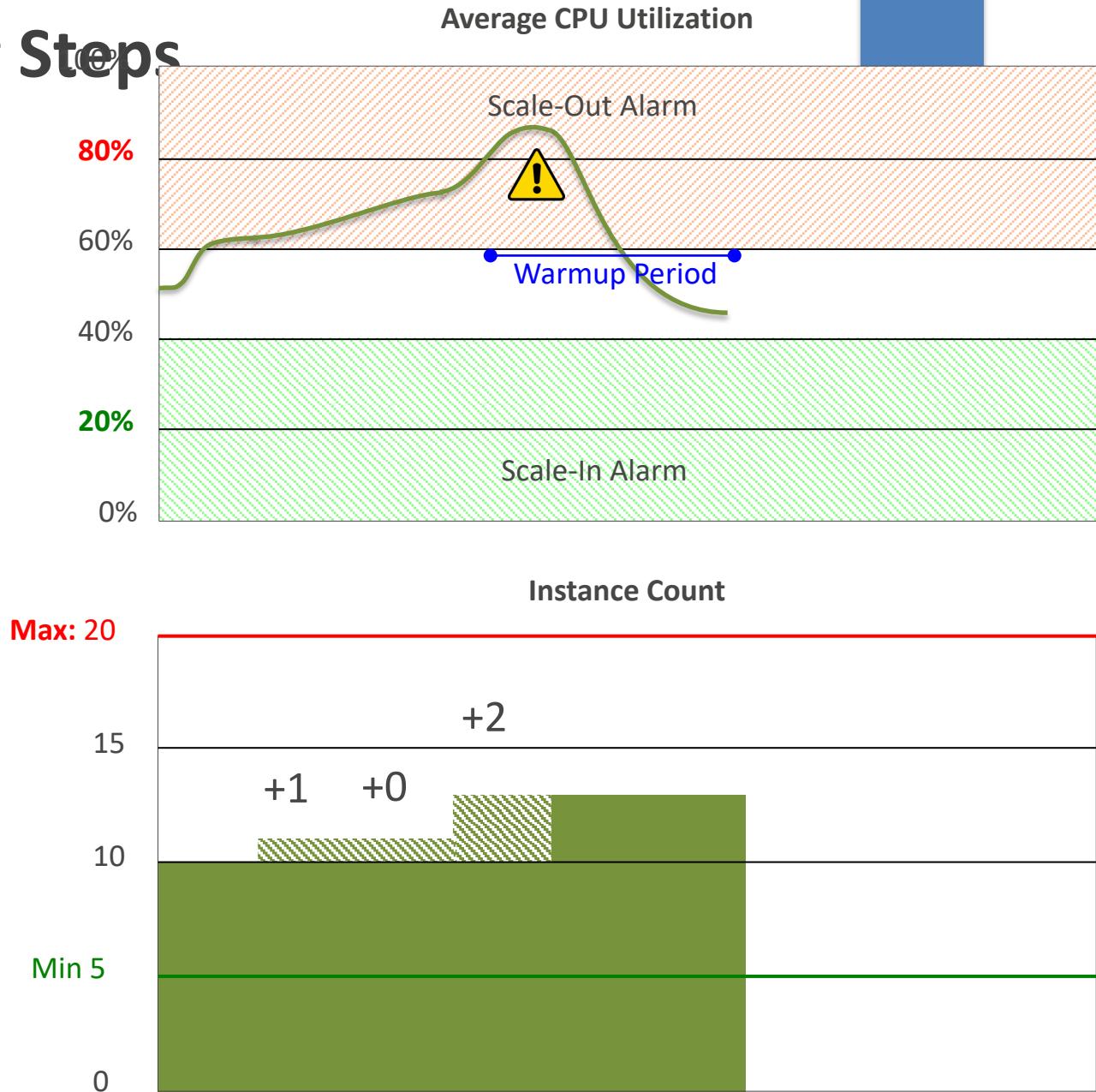
Auto Scaling Steps

As capacity matches usage:

- CPU utilization stabilizes.

When CPU utilization is 40-60%:

- No alarms are triggered.
- After warm up periods expire, new instances are added to the aggregated group metrics.



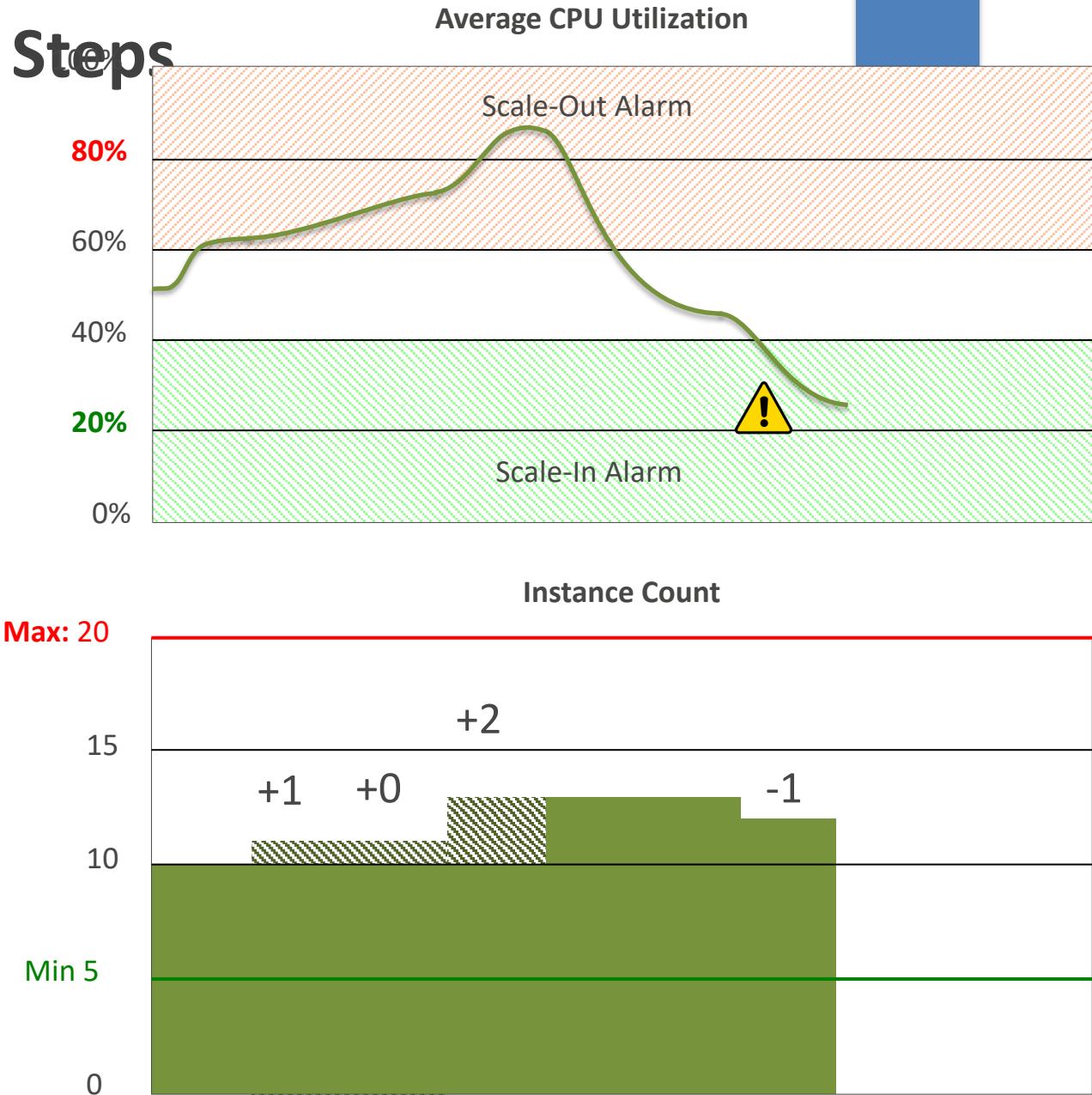
Auto Scaling Steps

As usage decreases:

- CPU utilization goes down.

When CPU utilization is 20-40%:

- Scale-in alarm is triggered.
- Remove 1 step policy is applied.
- An instance is removed from the Auto Scaling group and from the aggregated group metrics.



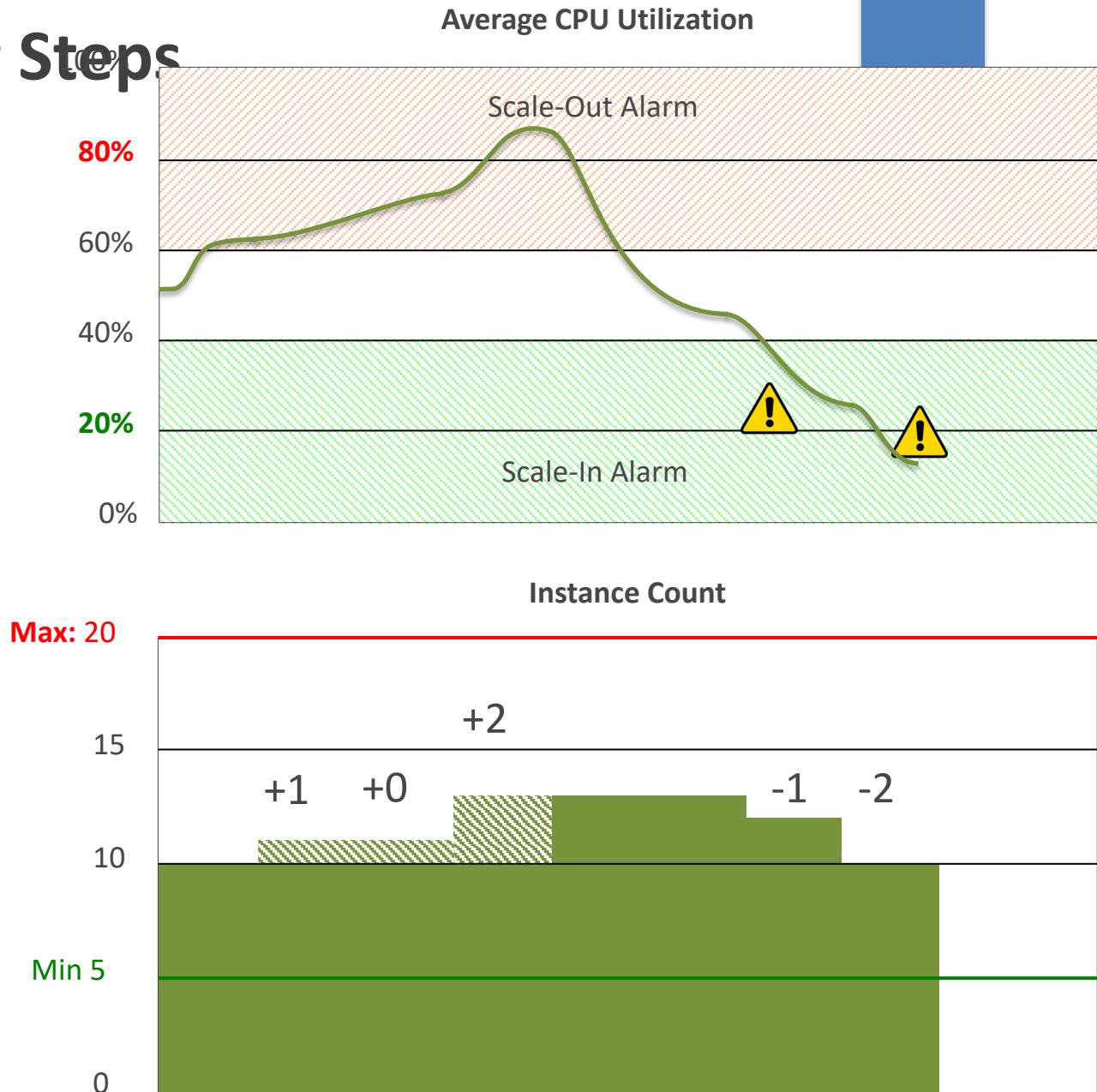
Auto Scaling Steps

As usage decreases:

- CPU utilization goes down further.

When CPU utilization is 0-20%:

- Scale in alarm is triggered.
- Remove 2 step policy is applied.
- Two instances are removed from the Auto Scaling group and from the aggregated group metrics.



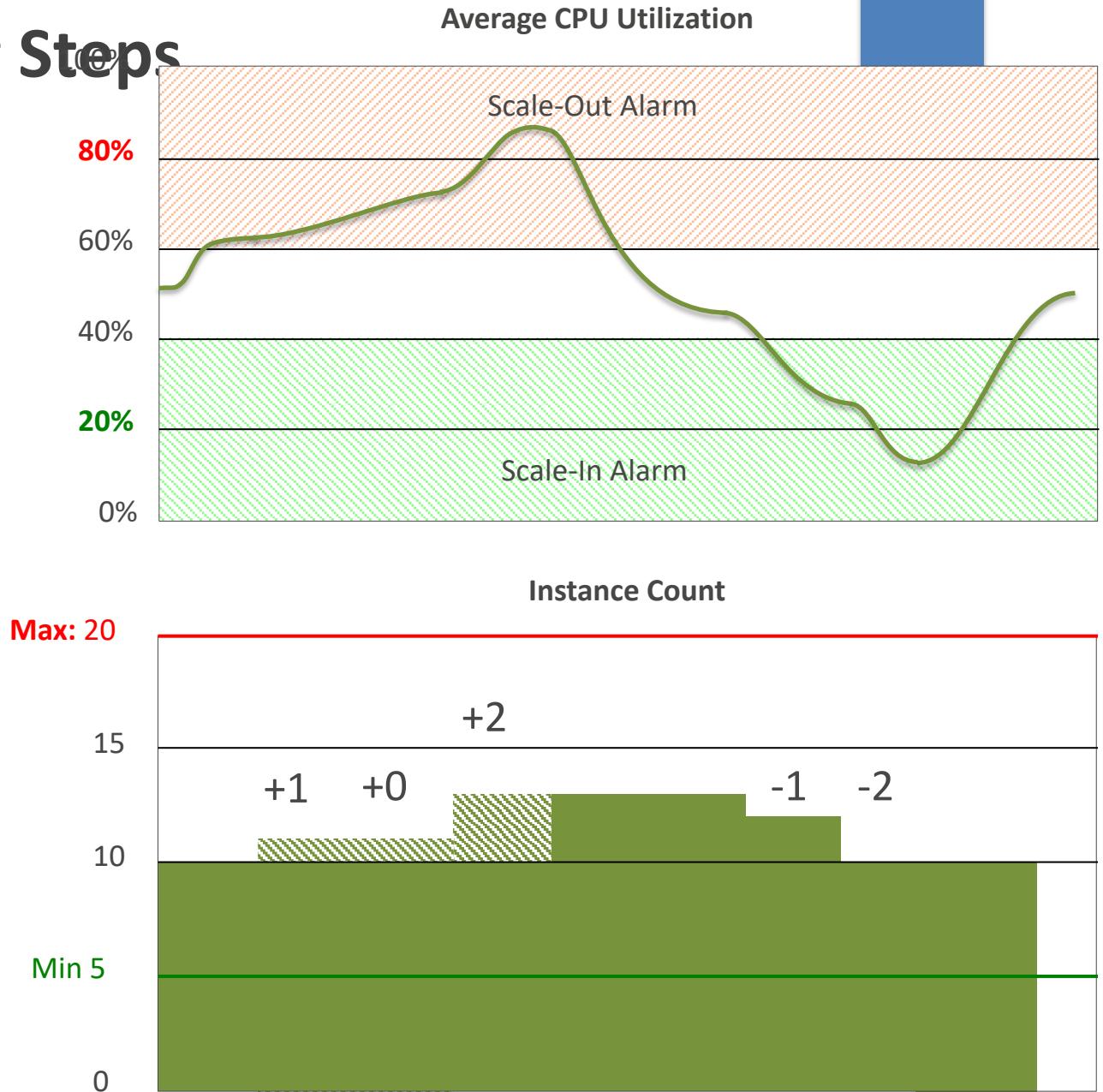
Auto Scaling Steps

As capacity matches usage:

- CPU utilization stabilizes.

When $40\% < \text{CPU Utilization} < 60\%$

- No alarm is triggered.
- No step policies are applied.
- No instances are added or removed from service.



Auto Scaling Considerations

- Avoid Auto Scaling thrashing.
 - Scale out early, scale in slowly.
 - Be more cautious about scaling **in**; avoid aggressive instance termination.
- Set the min and max capacity parameter values carefully.
- Use lifecycle hooks.
 - Perform custom actions as Auto Scaling launches or terminates instances.
- Stateful applications will require additional automatic configuration of instances launched into Auto Scaling groups.

Remember: Instances can take several minutes after launch to be **fully usable**.

Horizontal Scaling using AutoScaling

Topics

- Vertical v/s Horizontal Scaling
- Elastic Load Balancer
- Auto-Scaling
- Auto-Scaling Example

Amazon Route 53: DNS

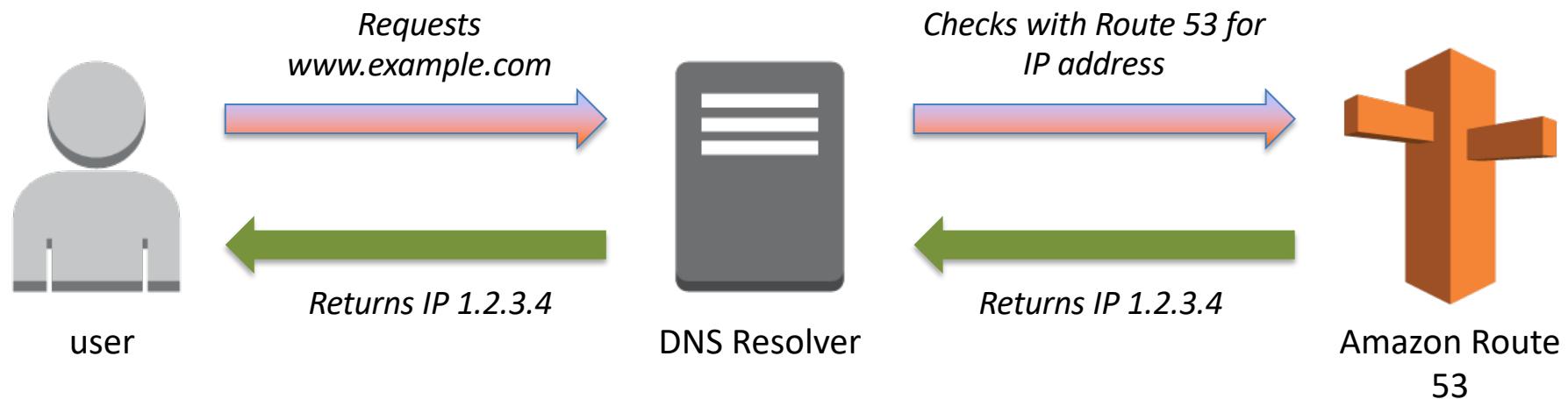
Amazon Route 53 is an authoritative DNS service from AWS with the following characteristics:

- DNS translates domain names (like www.amazon.com) into IP addresses .
- The name refers to the fact that DNS servers respond to queries on port 53.



Amazon
Route 53

DNS Resolution



Amazon Route 53

Reliable

- Redundant locations
- Backed with 100% Service Level Agreement (SLA)

Fast

- Worldwide anycast network
- Fast propagation of changes

Integrated with AWS

- ELB-Alias Queries
- Latency-based routing

Easy to Use

- Console
- Programmatic API
- Domain name management

Cost-Effective

- Inexpensive rates
- Pay-as-you-go model

Flexible

- Geolocation routing
- Weighted round robin
- Self-aliasing

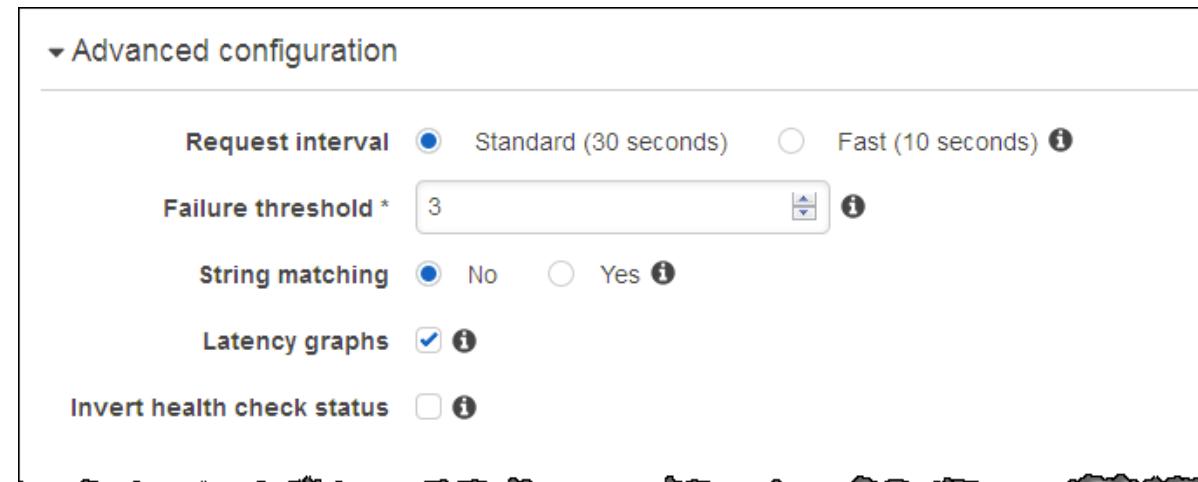
DNS Routing Policies

- **Simple routing:** Single server environments.
- **Weighted round robin:** Assign weights to resource record sets to specify the frequency.
- **Latency-based routing:** Helps to improve your global applications.
- **Health check and DNS failover:** Fail over to a backup site if your primary site becomes unreachable.
- **Geolocation routing:** Specify **geographic locations** by continent, by country, or by state in the United States.

Amazon Route 53 DNS Failover

Improve availability of your applications:

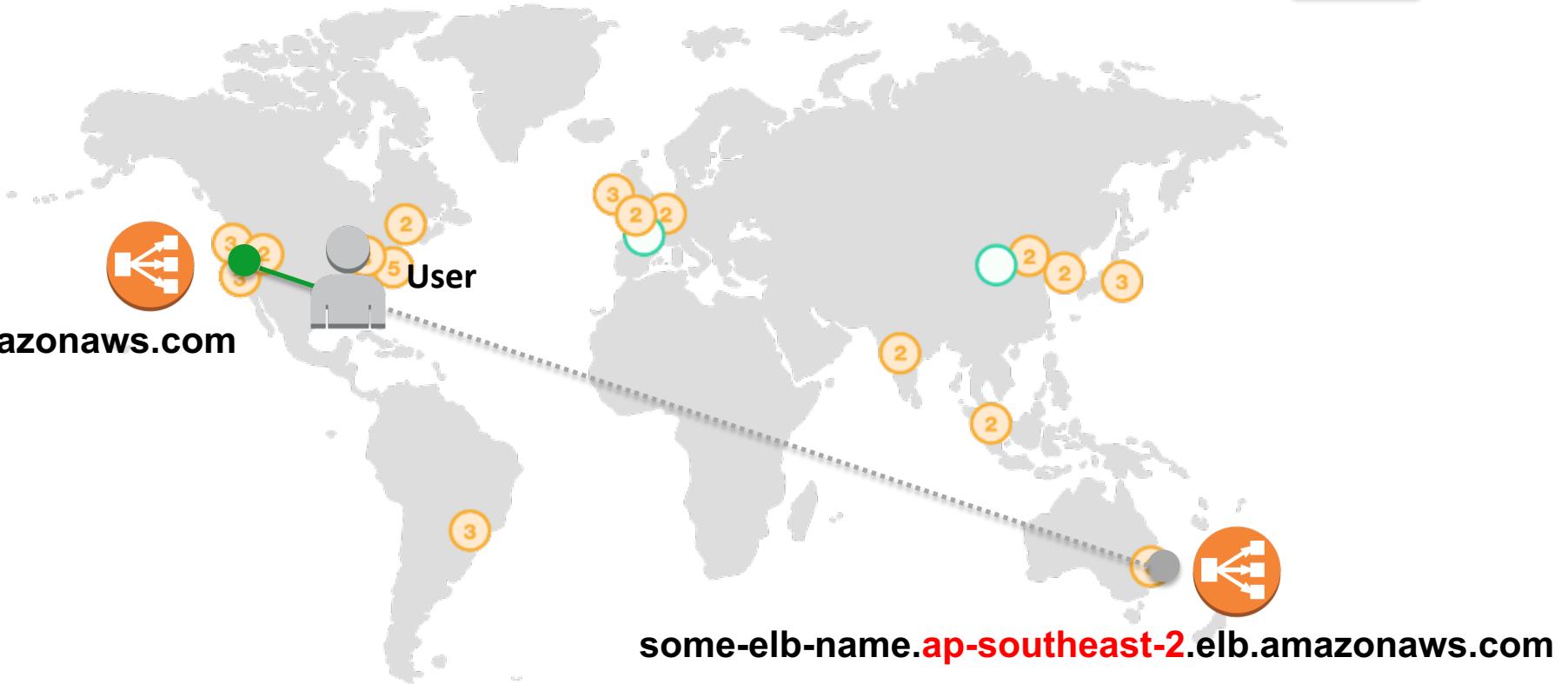
- Configuring backup and failover scenarios for your own applications.
- Enabling highly available multi-region architectures on AWS.
- Improving health checks by combining multiple health checks, domain name–based health checks, string matching, specifying the request interval, and more.



Use Case: Multi-Region Deployment

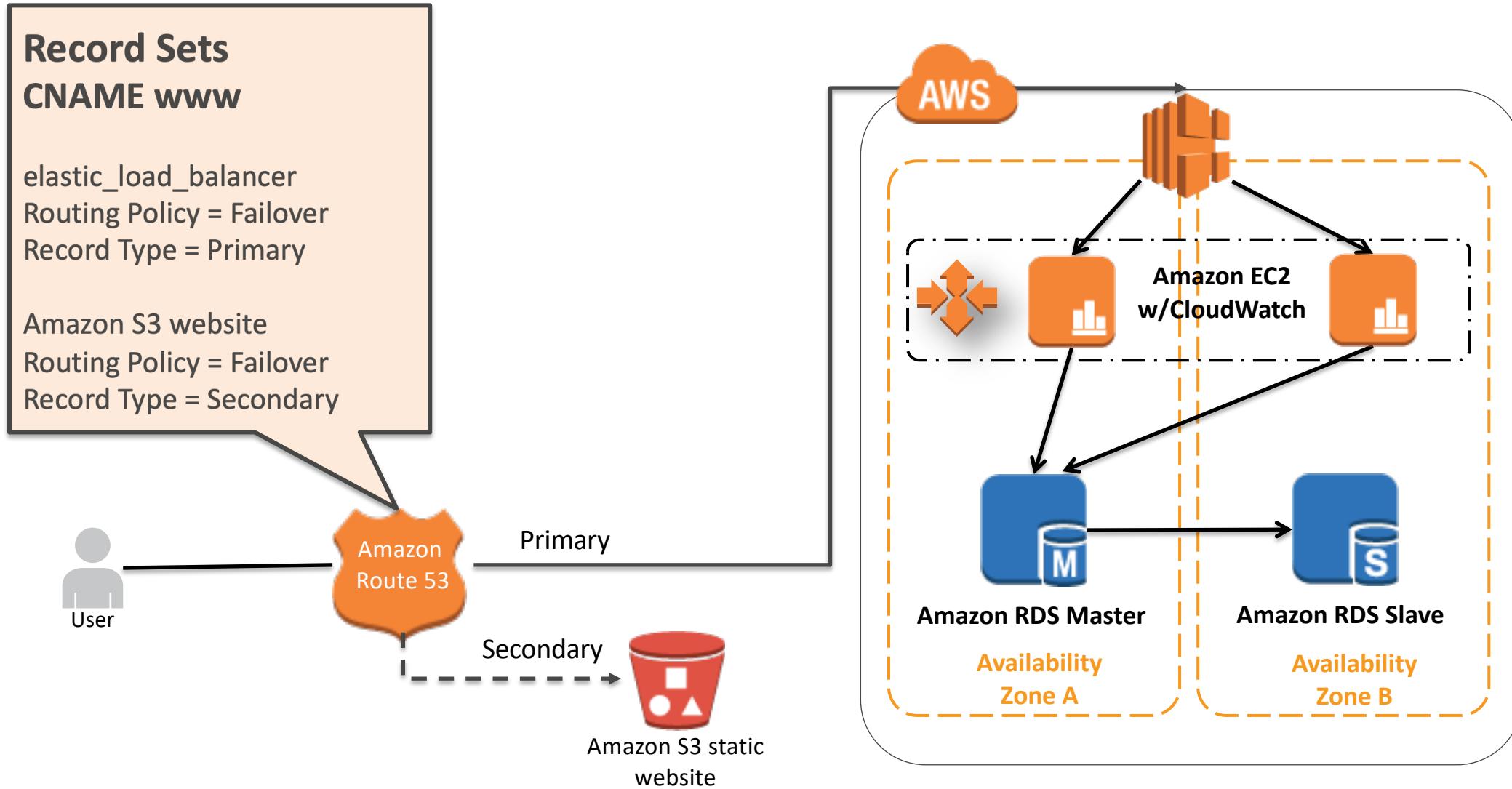


some-elb-name.us-west-2.elb.amazonaws.com

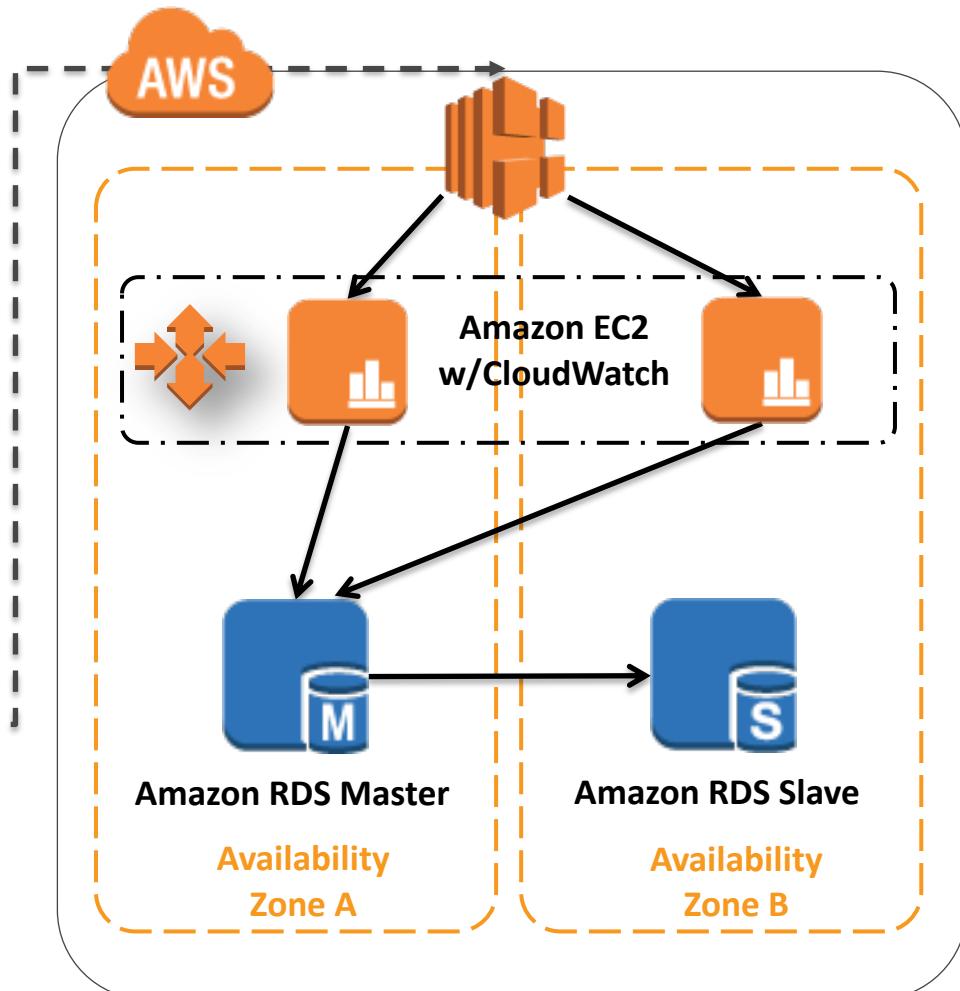
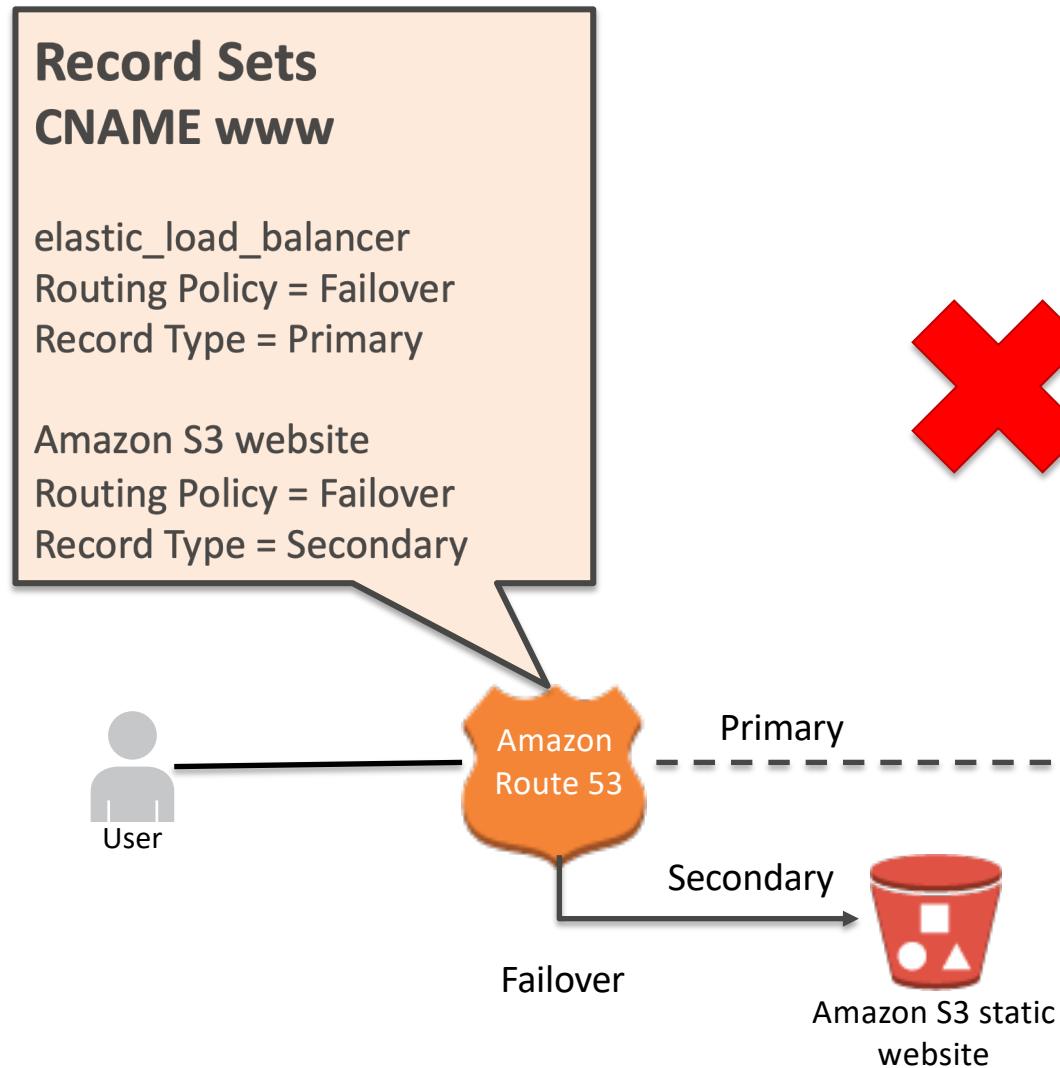


Name	Type	Value
amgogreen.com	ALIAS	some-elb-name.us-west-2.elb.amazonaws.com
amgogreen.com	ALIAS	some-elb-name.ap-southeast-2.elb.amazonaws.com

Typical Architecture



Typical Architecture



AWS Cloudformation

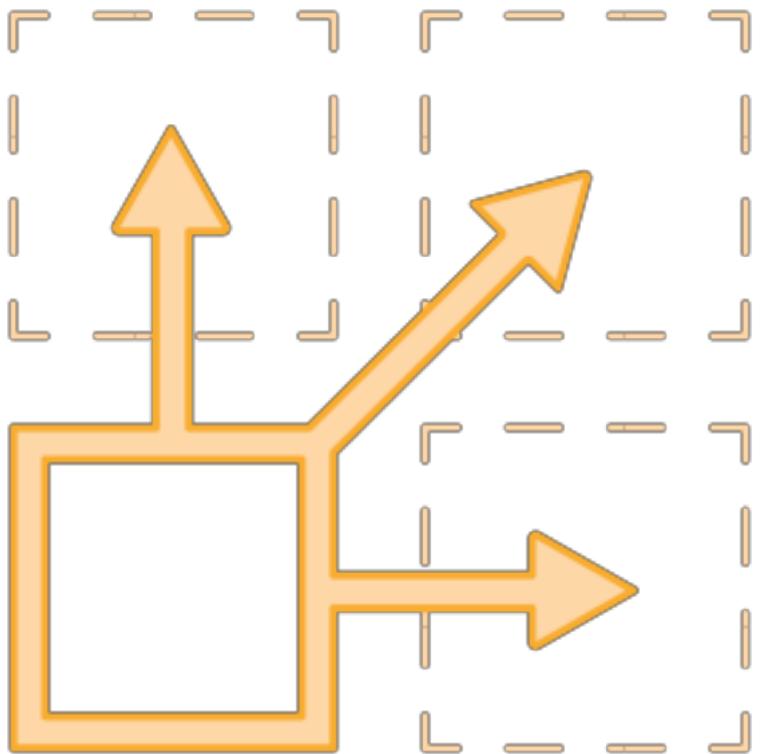
Why AWS CloudFormation?

Some customer challenges

- Creating and managing AWS resources
- Provisioning and updating infrastructure resources in an orderly manner
- Version controlling infrastructure like software

What is CloudFormation?

Templated resource provisioning



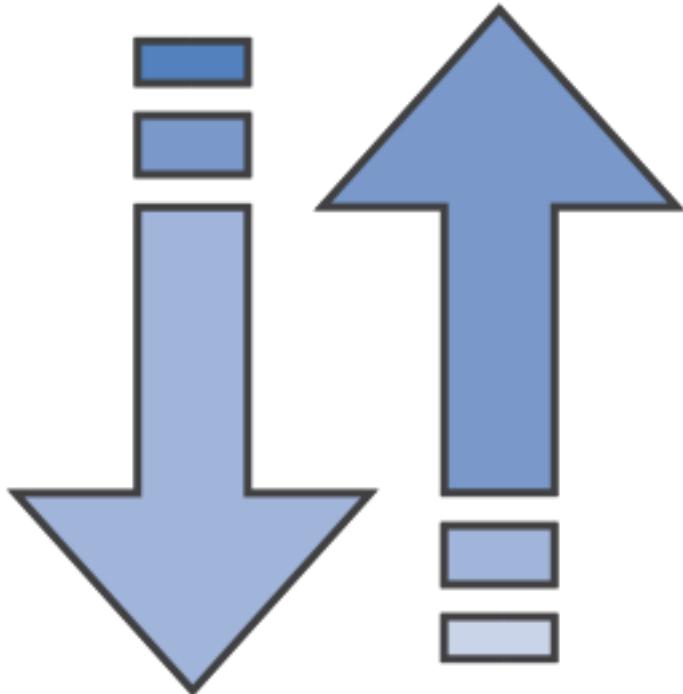
- Create templates to describe the AWS resources used to run your application
- Provision identical copies of a stack

Infrastructure as code



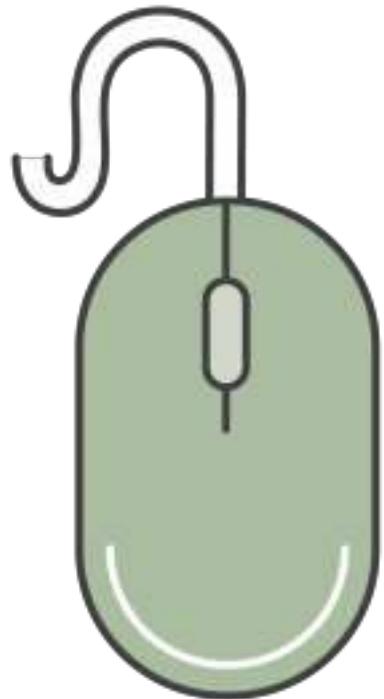
- Templates can be stored in a source control system
- Track all changes made to your infrastructure stack
- Modify and update resources in a controlled and predictable way

Declarative and flexible



- Just choose the resources and configurations you need
- Customize your template through parameters

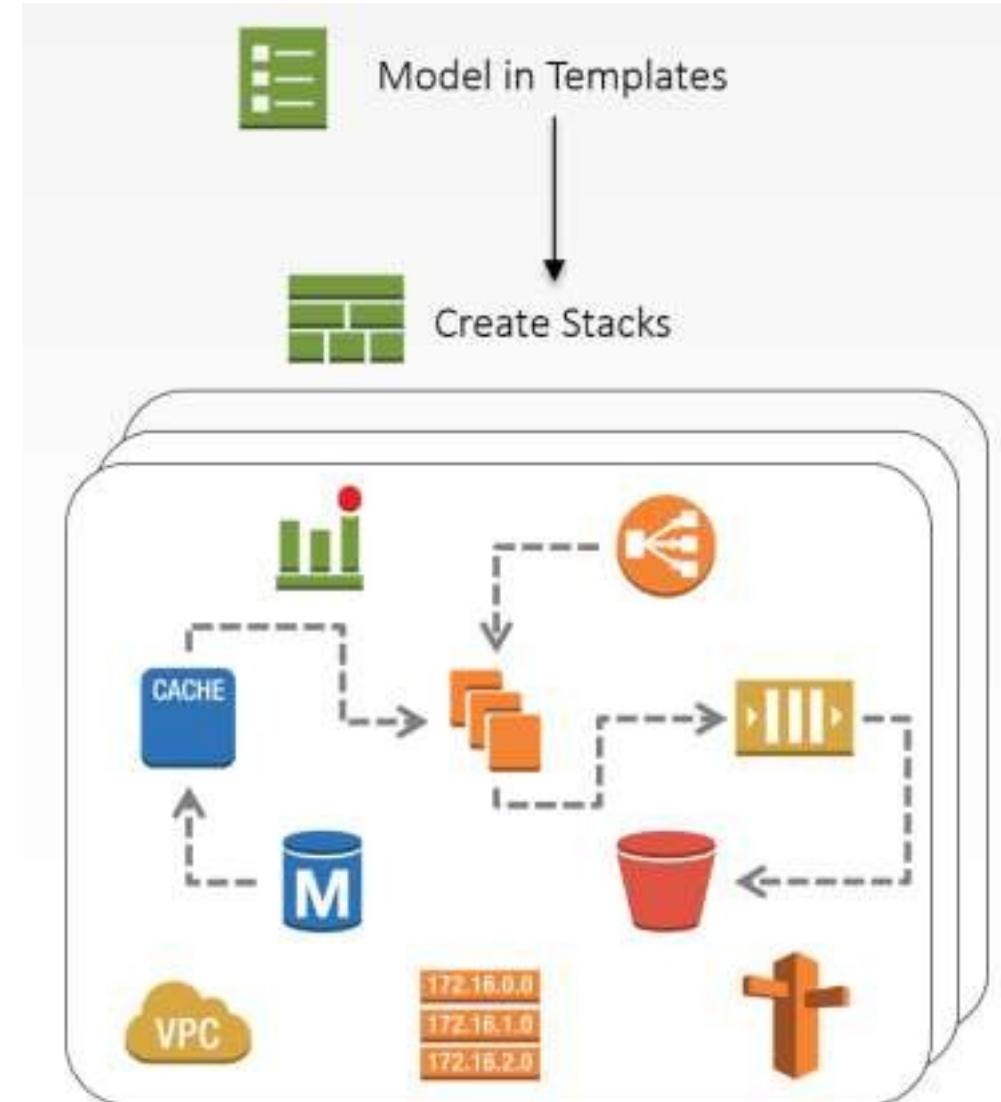
Easy to use



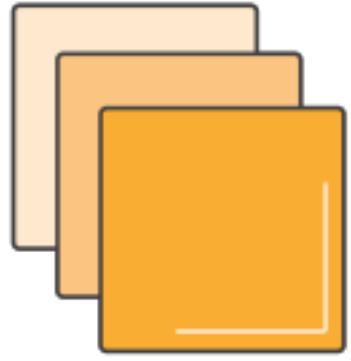
- Access through console, CLI, or SDKs
- Start with one of the many sample templates
- Integrate with your development and management tools

CloudFormation

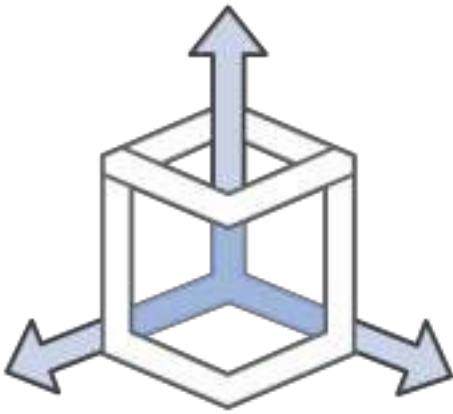
- Create templates of the infrastructure and applications you want to run on AWS
- Have CloudFormation automatically provision the required AWS resources and their relationships from the templates
- Easily version, replicate, or update the infrastructure and applications using the templates
- Integrates with other development, CI/CD, and management tools



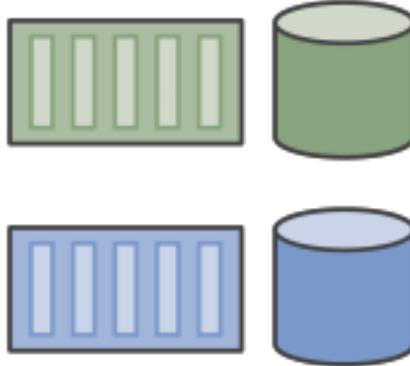
Common use cases



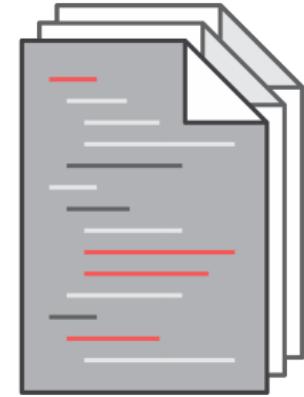
Stack
replication



Infrastructure
scale
out



Blue/green
deployment



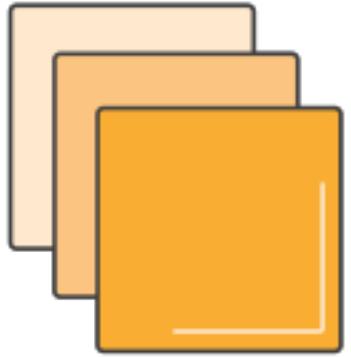
Infrastructure
as code

Pricing

- There is no additional charge for CloudFormation
- Customers pay only for the AWS resources (e.g., EC2 instances, EBS volumes) created using CloudFormation

How do I get started with CloudFormation?

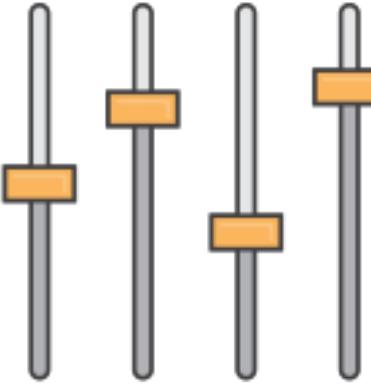
CloudFormation terminology



Stacks



Templates



Parameters



Policies

Basic steps

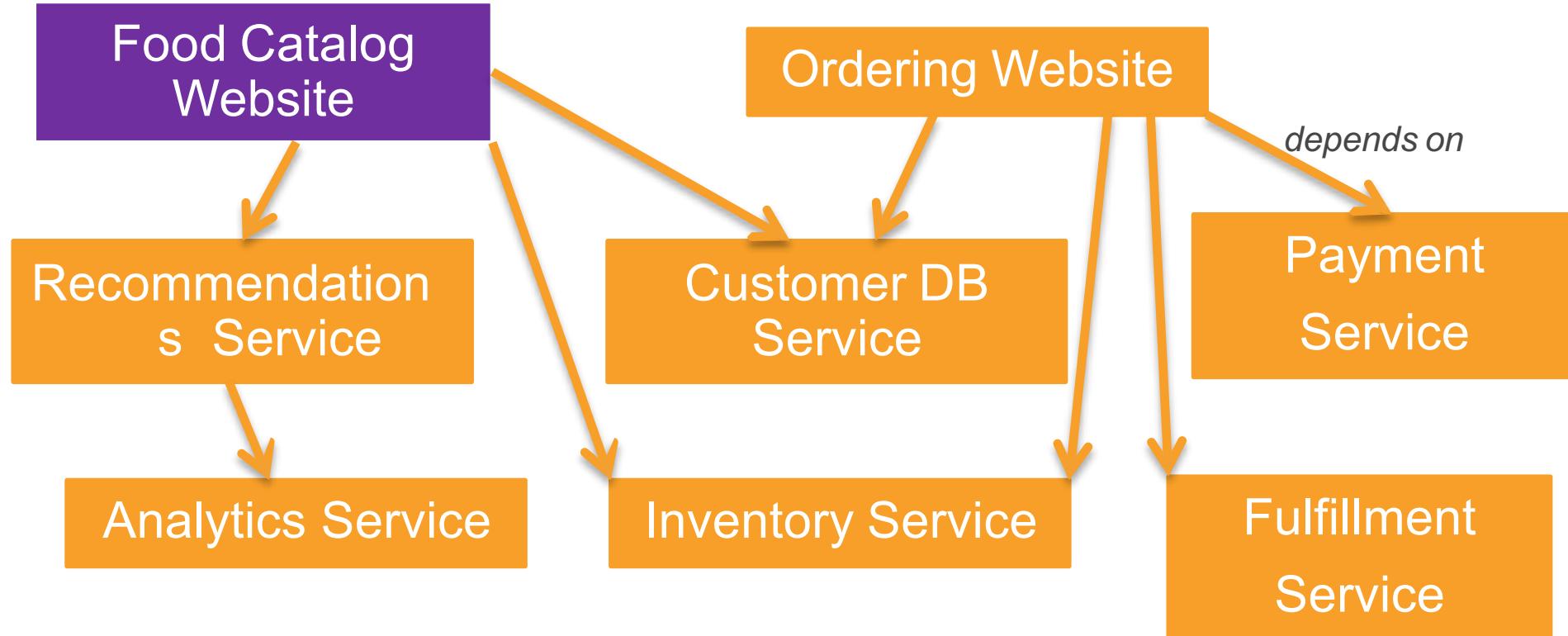


- Create or use an existing template
- Upload template to CloudFormation
- Specify parameter values
- Set up policies, tags, or notification options
- Review and create

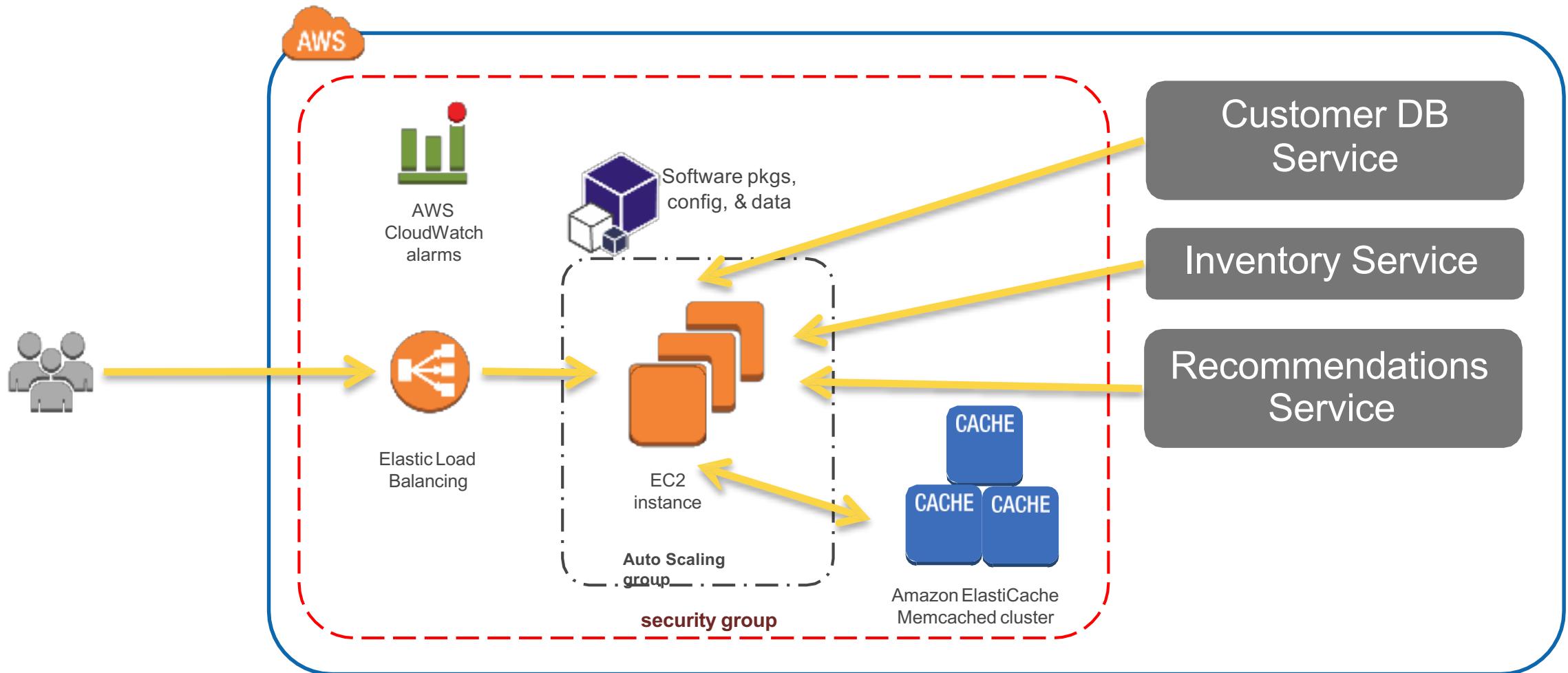
Sample Resources section of a template

```
"Resources" : {  
    "EC2Instance" : {  
        "Type" : "AWS::EC2::Instance",  
        "Properties" : {  
            "InstanceType" : { "Ref" : "InstanceType" } ,  
            "SecurityGroups" : [ { "Ref" : "InstanceSecurityGroup" } ] ,  
            "KeyName" : { "Ref" : "KeyName" } ,  
            "ImageId" : { "Fn::FindInMap" : [ "AWSRegionArch2AMI", { "Ref" : "AWS::Region" } ,  
                                            { "Fn::FindInMap" : [ "AWSInstanceType2Arch", { "Ref" :  
"InstanceType" }, "Arch" ] } ] }  
        }  
    },
```

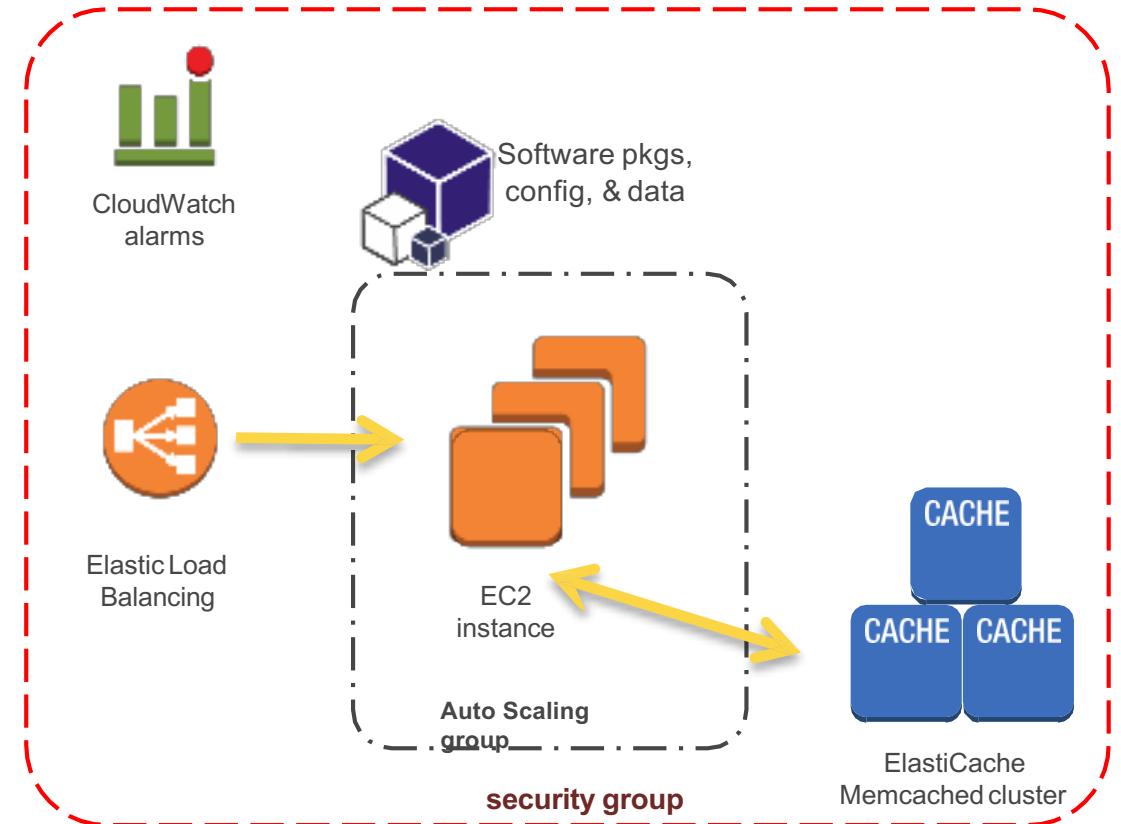
Design—Imagine building a food ordering service



Create template-example for the Food Catalog website



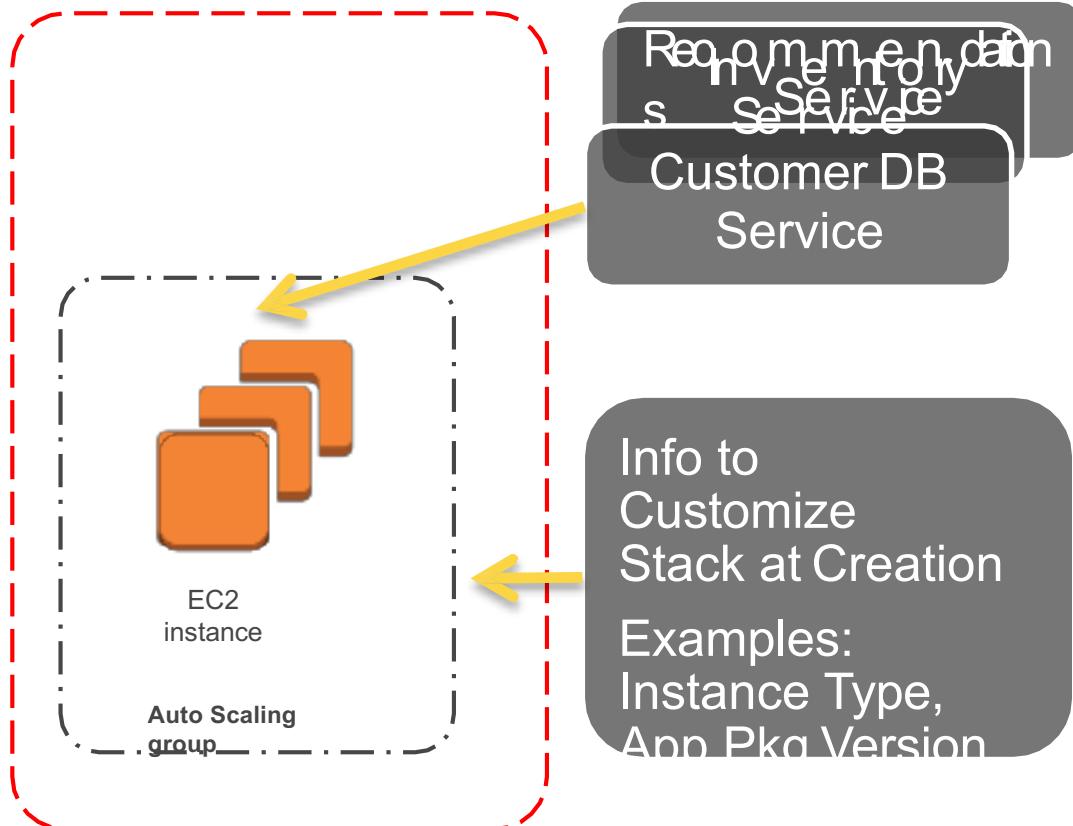
Create template—Resources



CloudFormation template

```
"Resources" : {  
    "SecurityGroup" : {},  
    "WebServerGroup" : {  
        "Type" : "AWS::AutoScaling::AutoScalingGroup",  
        "Properties" : {  
            "MinSize" : "1",  
            "MaxSize" : "3",  
            "LoadBalancerNames" : [ { "Ref" :  
                "LoadBalancer" } ],  
            ...  
        }  
    },  
    "LoadBalancer" : {},  
    "CacheCluster" : {},  
    "Alarm" : {}  
},
```

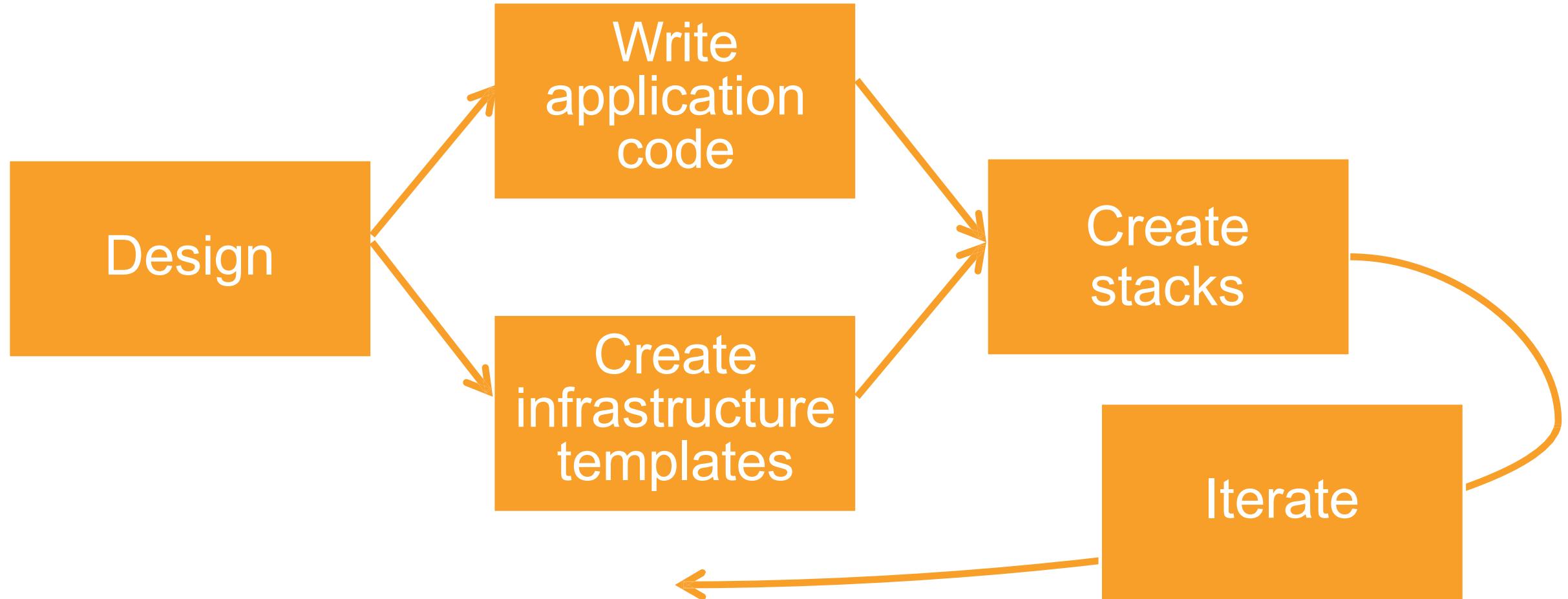
Create template–Parameters



CloudFormation template

```
"Parameters" : {  
    "CustomerDBServiceEndPoint" : {  
        "Description" : "URL of the Customer DBService", "Type" :  
        "String"  
    },  
    "CustomerDBServiceKey" : {  
        "Description" : "API key for the Customer DB  
Service",  
        "Type" : "String",  
        "NoEcho" : "true"  
    },  
    "InstanceType" : {  
        "Description" : "WebServer EC2instance type", "Type"  
        : "String",  
        "Default" : "m3.medium",  
        "AllowedValues" :  
        ["m3.medium","m3.large","m3.xlarge"],  
        "ConstraintDescription" : "Must be a valid  
instance type"  
    }  
}
```

Basic workflow



Infrastructure as code workflow



Code
templates

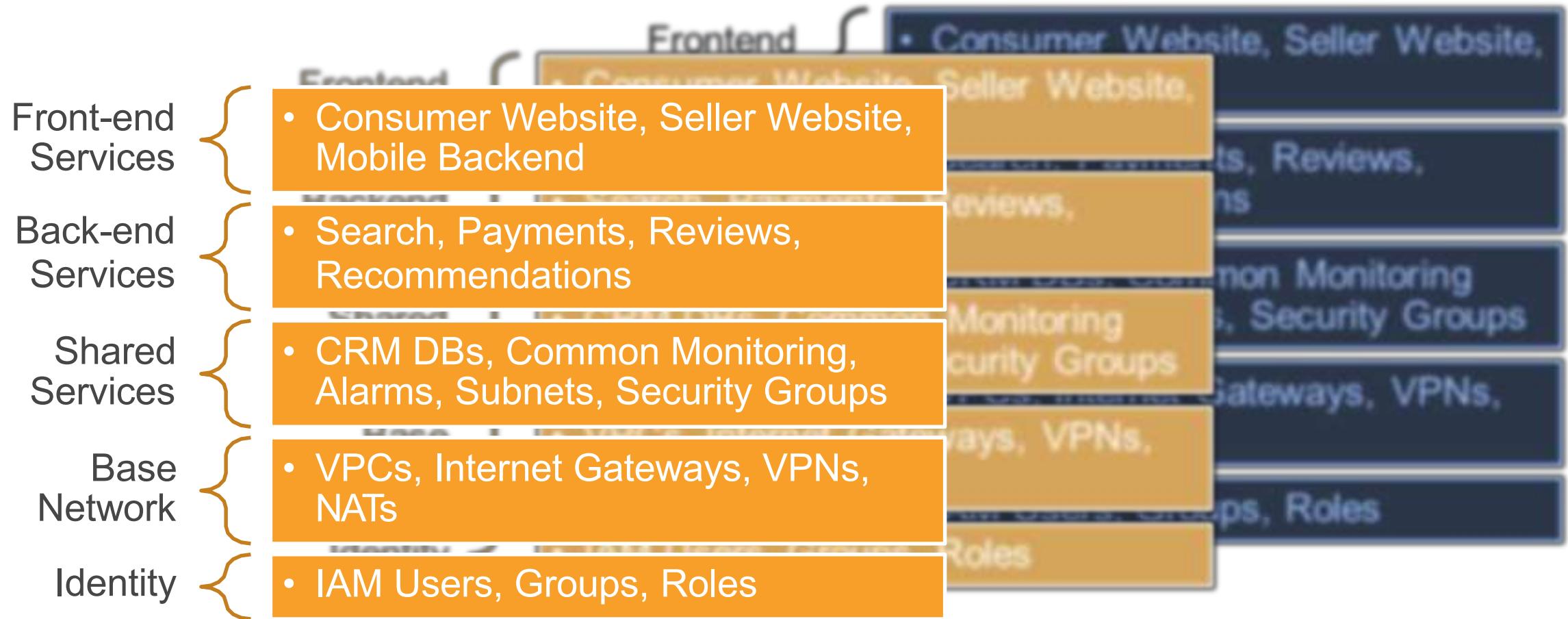
Version
control

Code
review

Integrate

“It’s all software”

“It’s all software”—Organize like it’s software



Best practices (1 of 2)

- Planning and organizing
 - Organize your stacks by lifecycle and ownership
 - Reuse templates to replicate stacks in multiple environments
 - Verify quotas for all resource types
 - Use nested stacks to reuse common template patterns
- Creating templates
 - Do not embed credentials in your templates
 - Use AWS-specific parameter types
 - Use parameter constraints
 - Validate templates before using them

See: <http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/best-practices.html>

Best practices (2 of 2)

- Managing stacks
 - Manage all stack resources through CloudFormation
 - Create change sets before updating your stacks
 - Use stack policies
 - Use CloudTrail to log CloudFormation calls
 - Use code reviews and revision controls to manage your templates

See: <http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/best-practices.html>

:

Environment Setup

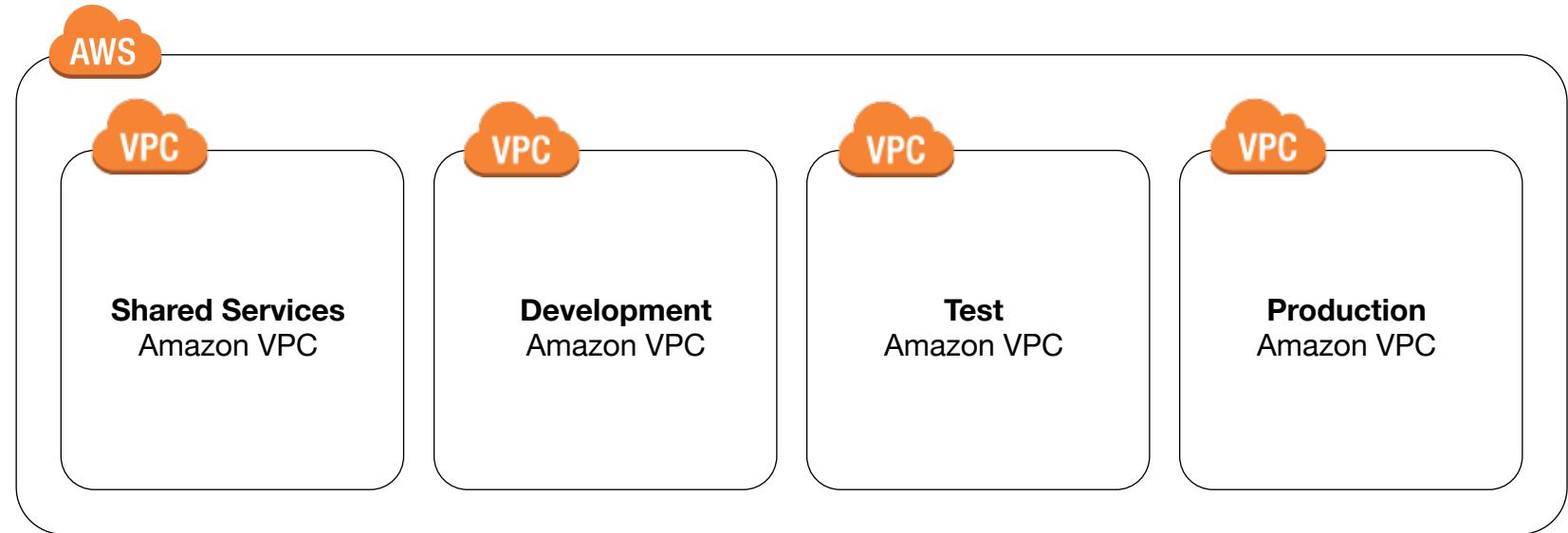
Architecting on AWS

Topics

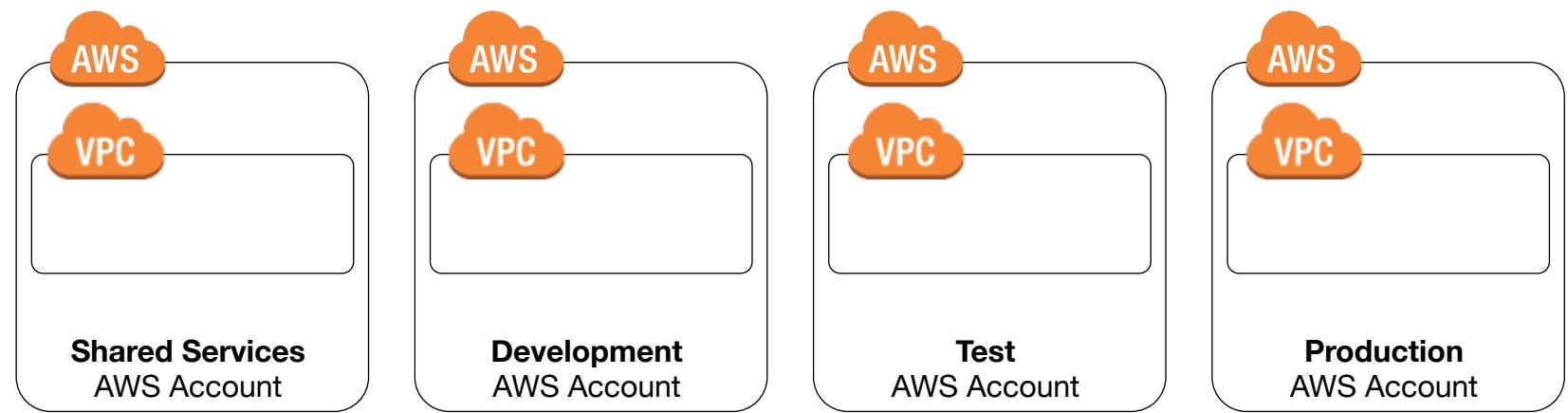
- IAM Identity federation
- VPC VPN Setup
- VPC Peering
- VPC Endpoints

AWS Infrastructure Patterns

Multi-VPC
pattern



Multi-Account
pattern



Choosing a Pattern

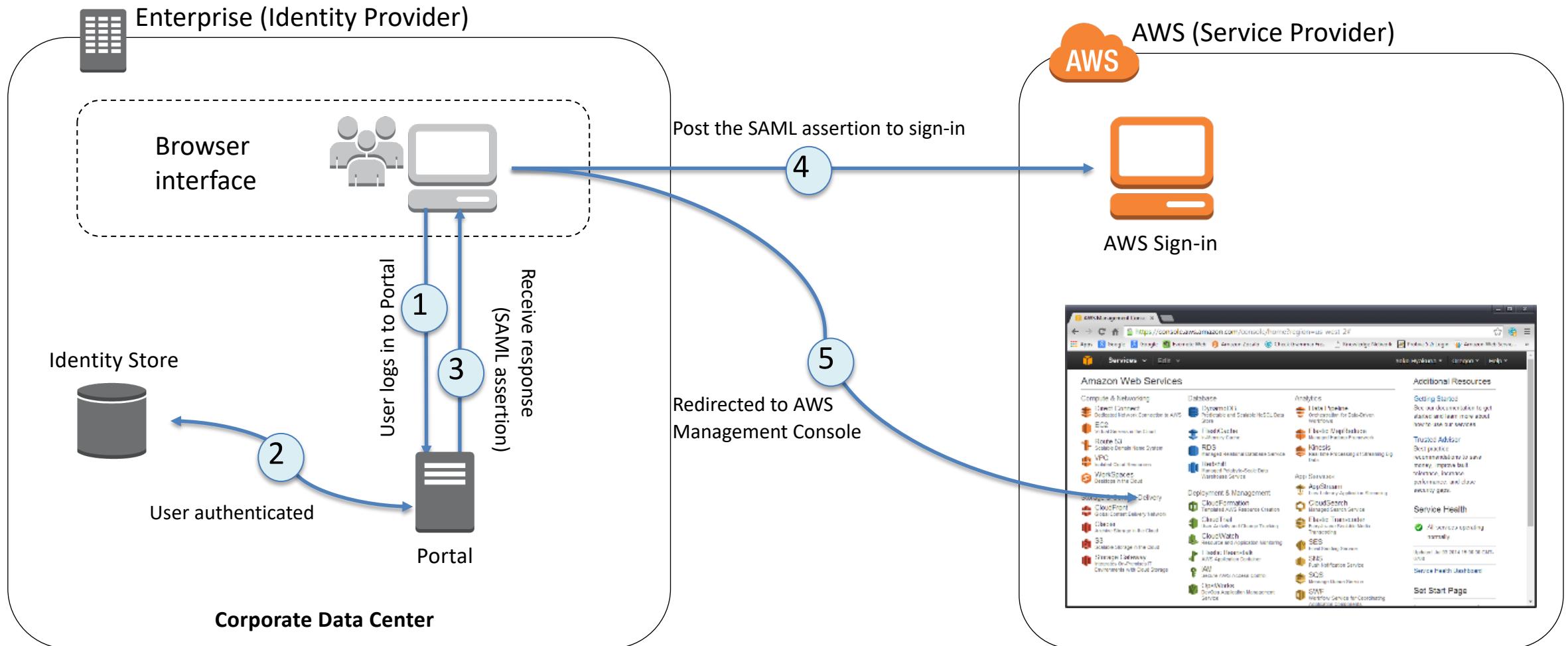
- How do you know which pattern to use?
- The primary factors for determining this are the [complexity](#) of your organization and your [workload isolation](#) requirements:
- Single IT team? [Multi-VPC](#)
- Large organization with many IT teams? [Multi-account](#)
- High workload isolation required? [Multi-account](#)

Multi-Account Pattern

- Uses **two or more AWS accounts** to organize application environments
 - Uses **one or more VPC** per AWS account
- Best suited for:
 - **Large organizations** and **organizations with multiple IT teams**, such as Enterprise-level corporations or government agencies
 - **Medium-sized organizations** that anticipate rapid growth
- Why?
 - **Managing access** and **standards** can be more challenging in more complex organizations.

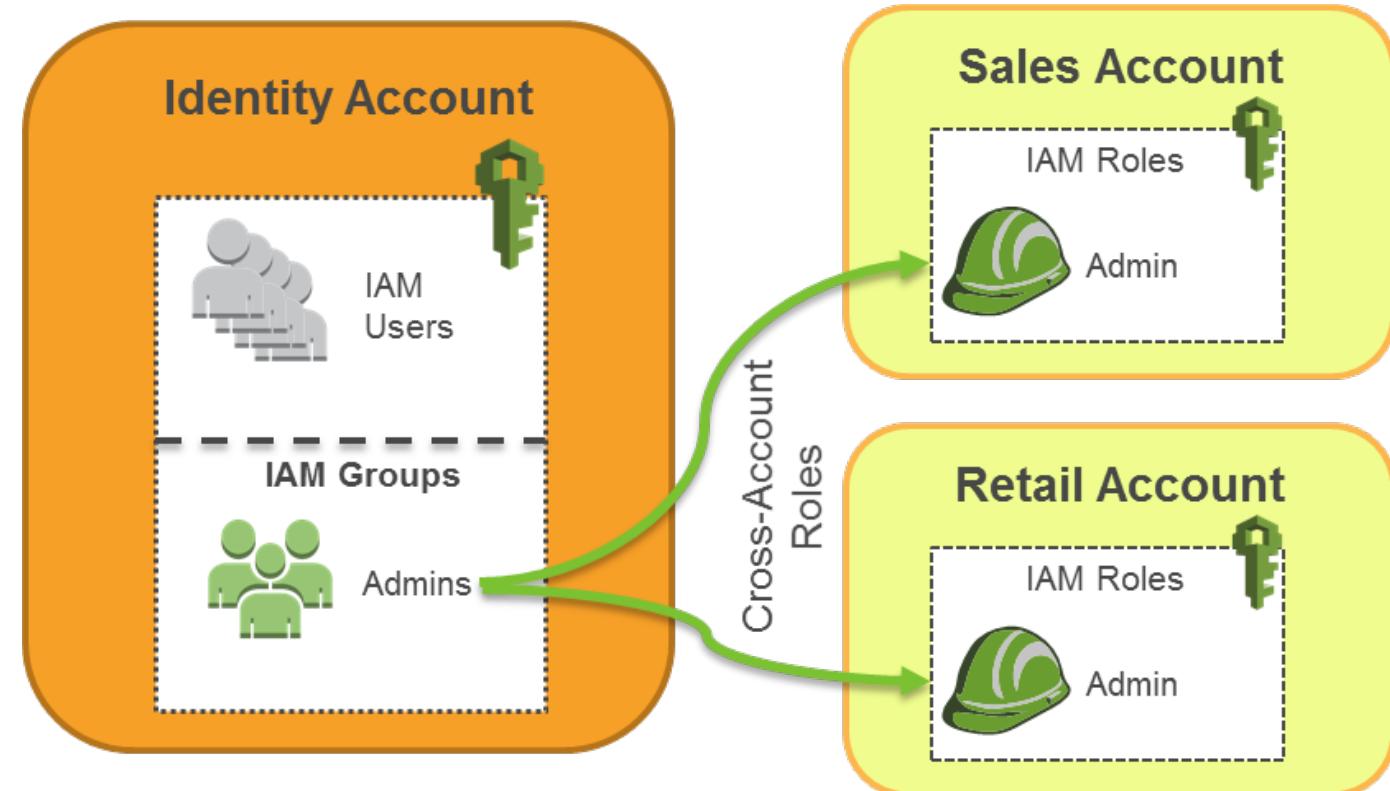
Should you maintain a new identity
for your AWS users?

SSO Federation Using SAML (3 of 3)



Identity Account Structure

- Create and manage all their users in a single account and enable user and group access to resources in other accounts.
- IAM cross-account roles grants predetermined users, groups, or roles in one AWS account permission



Multi-VPC Pattern

- Uses [one AWS Account](#)
 - Uses two or more [VPCs](#) to organize application environments
- Best suited for:
 - [Single team or single organizations](#), such as Managed Service Providers
- Why?
 - Limited teams make [maintaining standards](#) and [managing access](#) far easier.
- Exception:
 - [Governance](#) and [compliance standards](#) may require workload isolation regardless of organizational complexity.

Subnet Sizing

Recommendation: Consider larger subnets over smaller ones (/24 and larger).

Simplifies workload placement:

- Choosing where to place a workload among 10 small subnets is more complicated than with one large subnet.

Less likely to waste or run out of IPs:

- If your subnet runs out of available IPs, you can't add more to that subnet.
 - **Ex.:** If you have 251 IPs in a subnet that's using only 25 of them, you can't share the unused 226 IPs with another subnet that's running out.

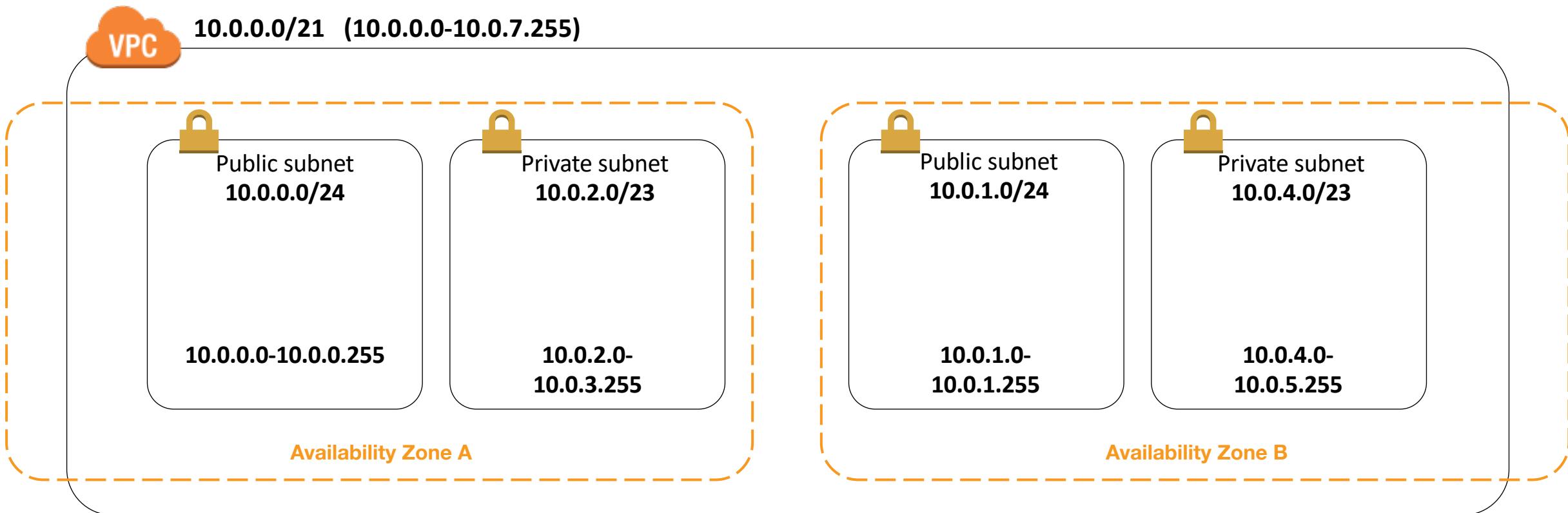
Subnet Types

Which subnet type (public or private) should you use for these resources?

- **Data store instances**
 - Private
- **Batch processing instances**
 - Private
- **Back-end instances**
 - Private
- **Web application instances**
 - Public or private*

Subnets Sizing

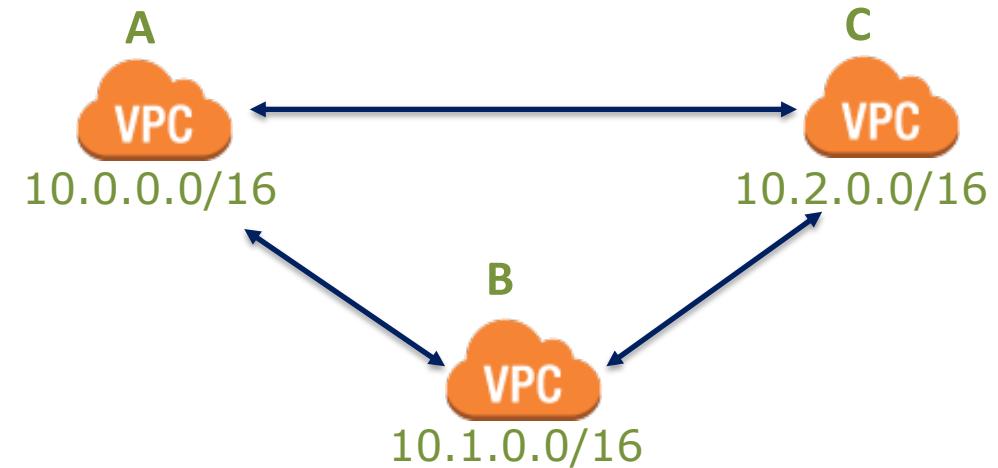
- **Recommendation:** Allocate substantially more IPs for private subnets than for public subnets.



Can you connect multiple VPCs
to each other?

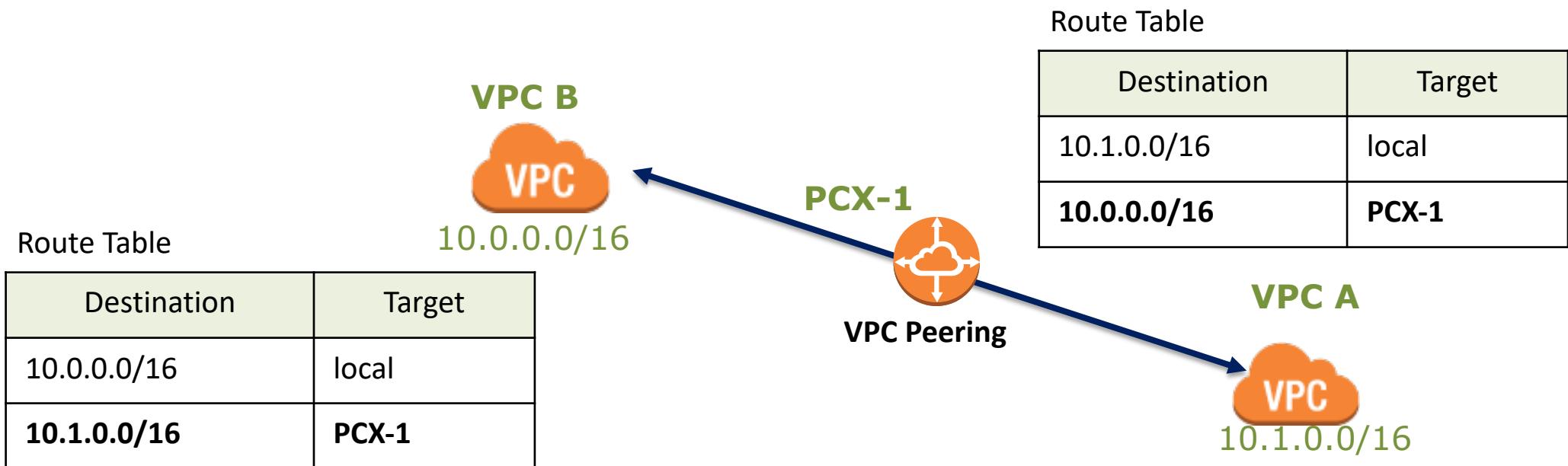
VPC Peering

- VPC peering connection allows you to route traffic between the peer VPCs.
 - Use private IP addresses.
 - VPCs reside in the same region.
 - IP space cannot overlap.
 - Only one between any two VPCs.
 - Transitive peering relationships are not supported
 - Can be established between different AWS Accounts
- Instances in either VPC can communicate with each other as if they are within the same network.



How Does VPC Peering Work?

- No Internet gateway or virtual gateway required
- No single point of failure
- No bandwidth bottlenecks

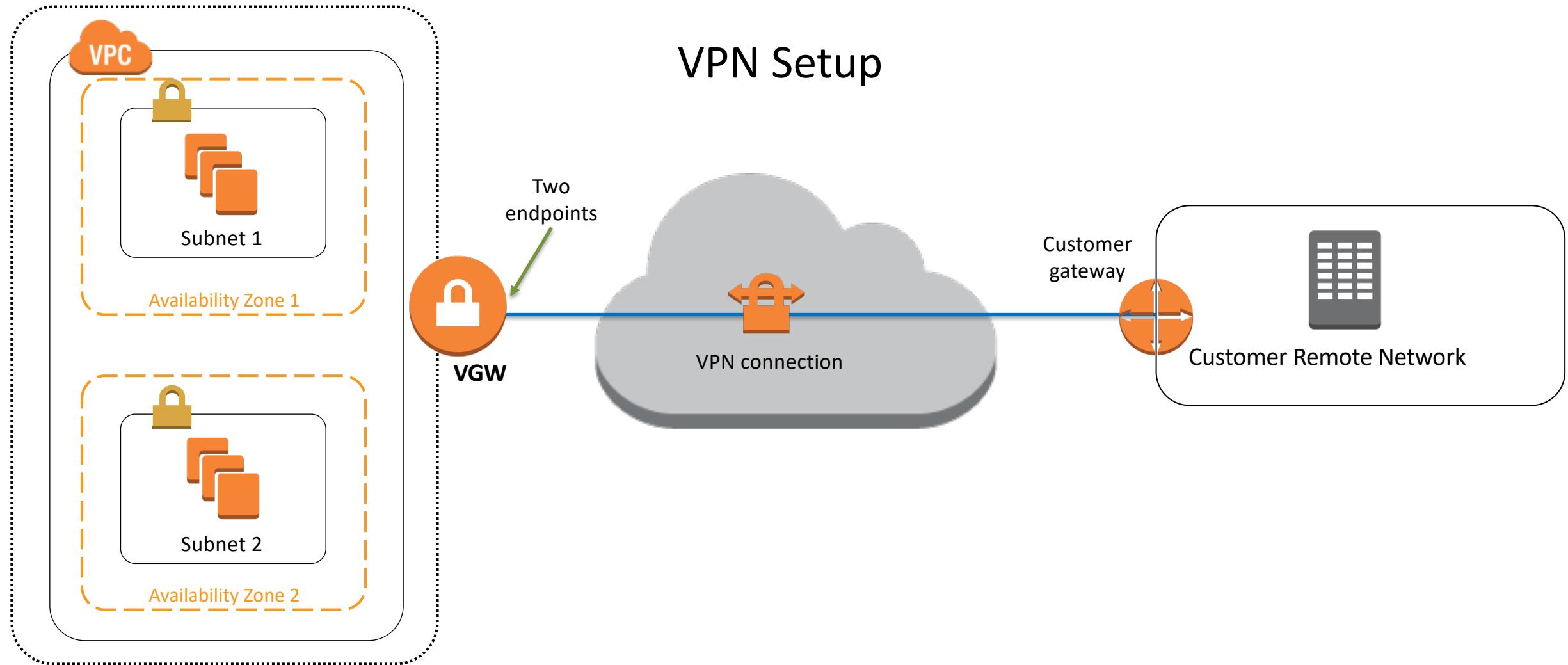


Other Important Considerations

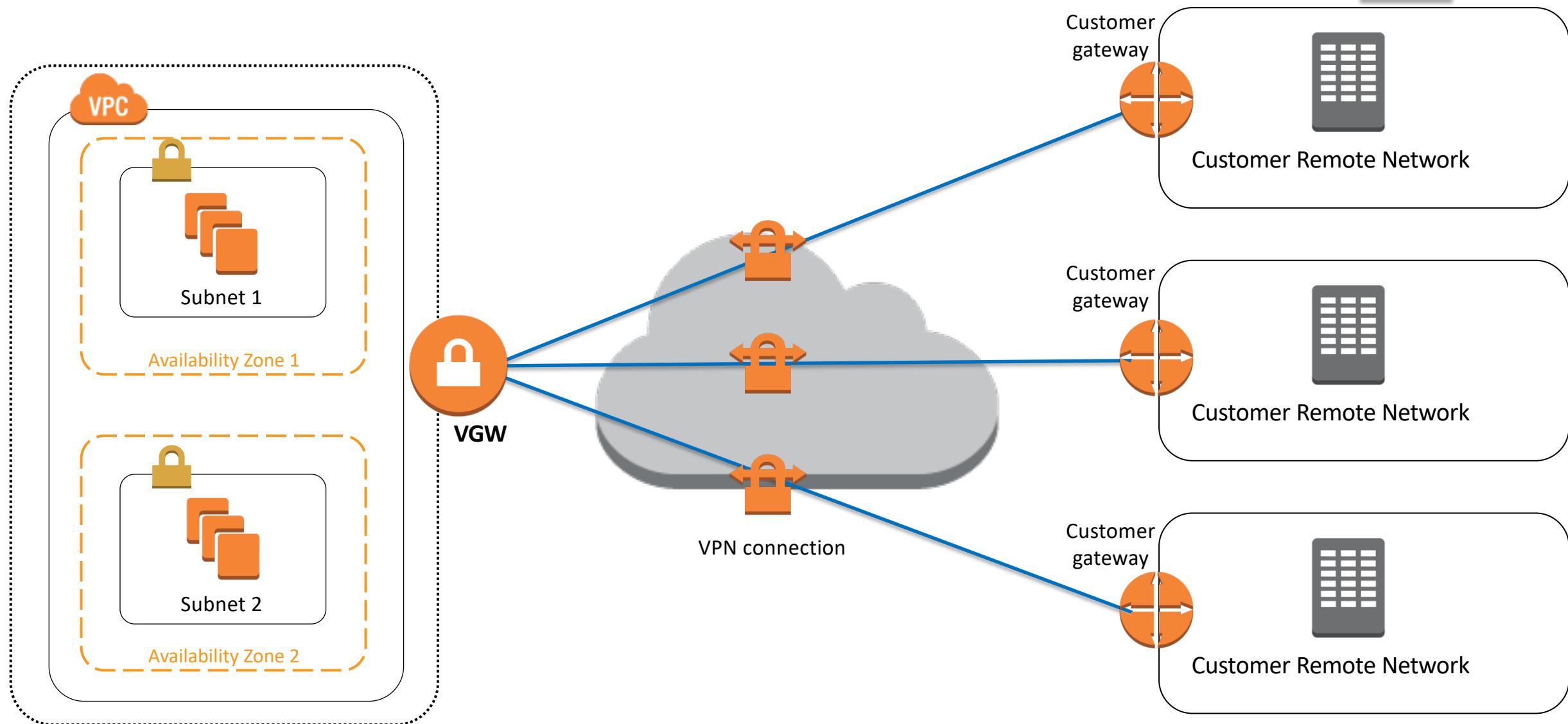
- The majority of AWS services **do not actually sit within a VPC.**
 - For these services, a VPC **cannot provide any isolation** outside of connectivity.
 - Amazon S3 and DynamoDB offer **VPC endpoints** to connect without traversing the public Internet.
 - Network traffic between AWS Regions traverse the AWS global network backbone by default.

How do you integrate on-premises infra with your AWS environment?

Extending On-Premises Network to AWS



Multiple VPN





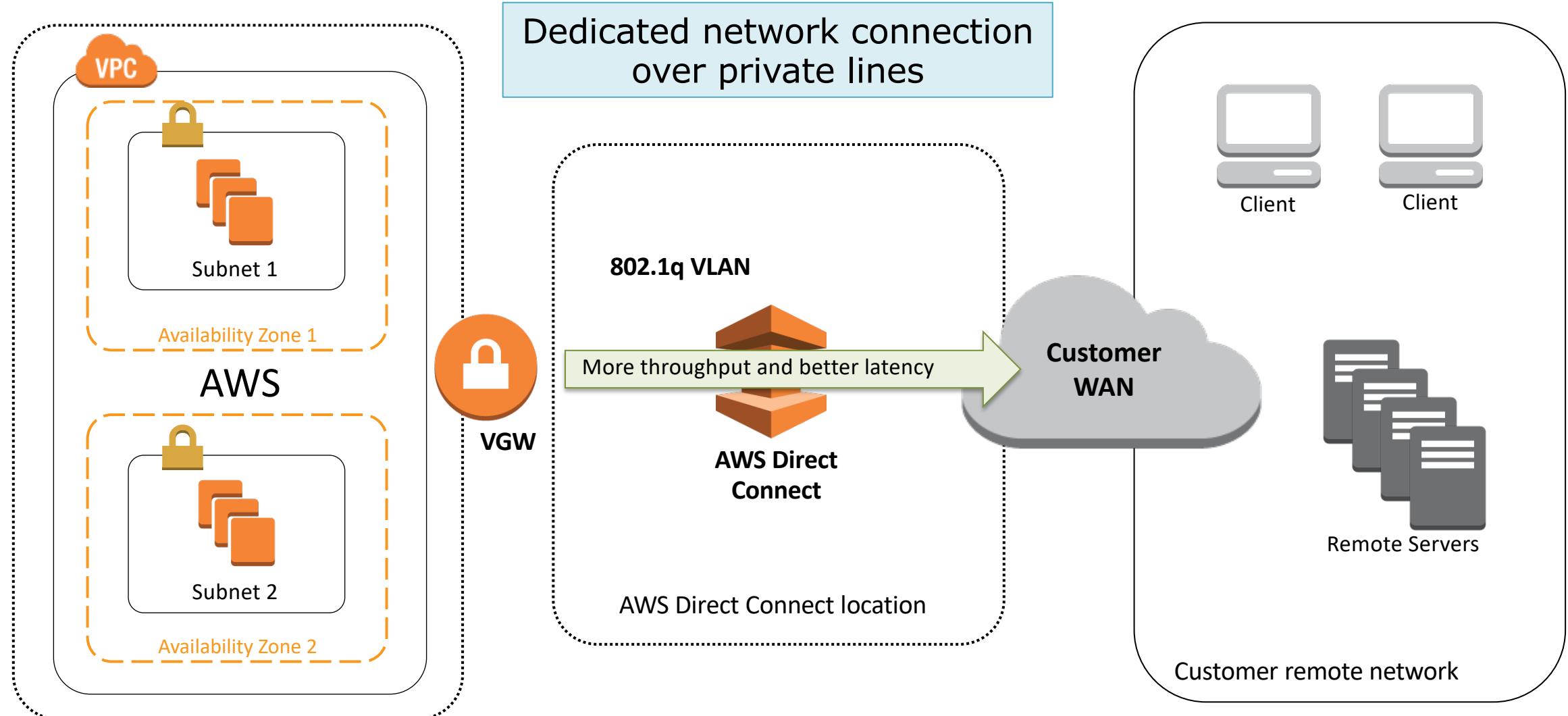
AWS Direct Connect

- AWS Direct Connect provides a private network (MPLS) connection between AWS and your data center.
- It is a network service alternative to using the Internet to access AWS cloud services.

Benefits:

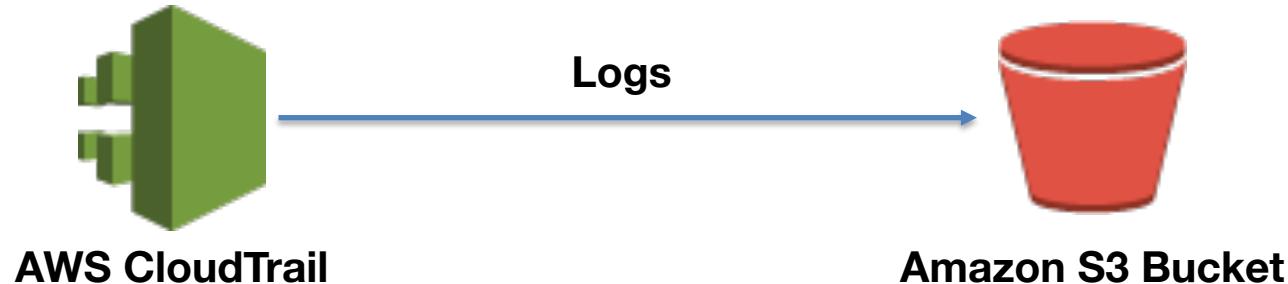
- Reduced network transfer costs
- Improved application performance with predictable metrics
- Transferring large data sets
- Security and compliance
- Hybrid cloud architectures
- Private data center expansion
- Alternative to Internet-based IPSec VPN connections (IPSec VPN connections can be used as a failover)

AWS Direct Connect



AWS CloudTrail

- Records AWS API calls for accounts.
- Delivers log files with information to an Amazon S3 bucket.
- Makes calls using the AWS Management Console, AWS SDKs, AWS CLI and higher-level AWS services.



Analyze AWS Access Logs

- Do you need to track the API calls for one or more AWS accounts?
- Use cases enabled by [CloudTrail](#):
 - Security analysis
 - Track changes to AWS resources
 - Troubleshoot operational issues
 - Compliance aid
 - Use with AWS Config to identify responsibility for a change



Track Resource Changes with AWS Config

- Provides AWS resource [inventory](#), configuration [history](#), and configuration [change notifications](#).
- Provides continuous details on all configuration changes associated with AWS resources.
- Combines with CloudTrail for full visibility into what contributed to a change.
- Enables compliance auditing, security analysis, resource change tracking, and troubleshooting.

Decoupled Architecture using SNS and SQS

Architecting on AWS

Topics

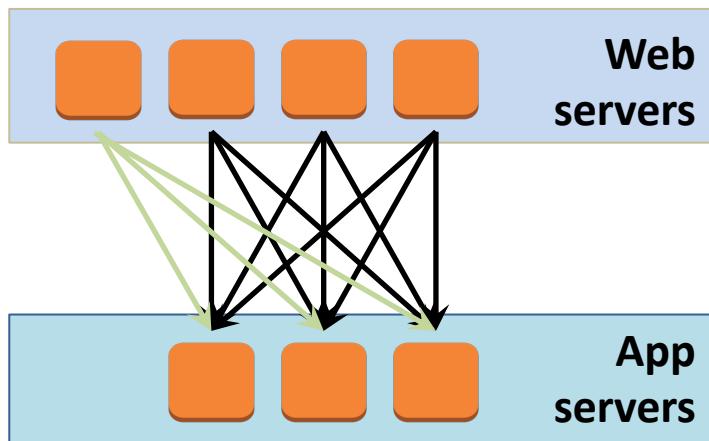
- Loosely Coupled System
- SQS
- SNS
- Architecture Example

Loosely Coupled System

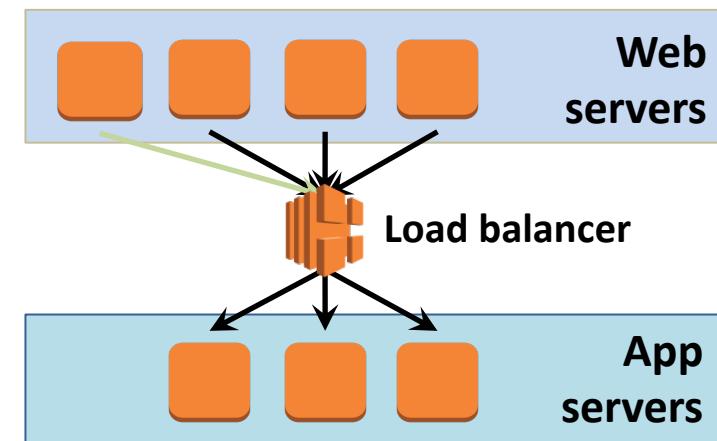
Decoupling

- The more loosely your system is coupled:
 - The more easily it scales.

Tightly coupled



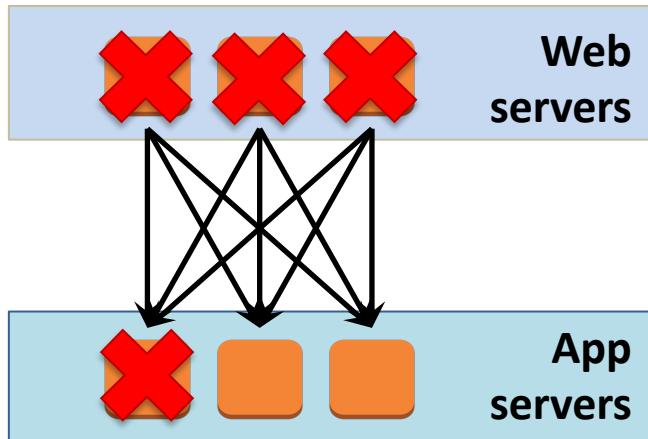
Loosely coupled



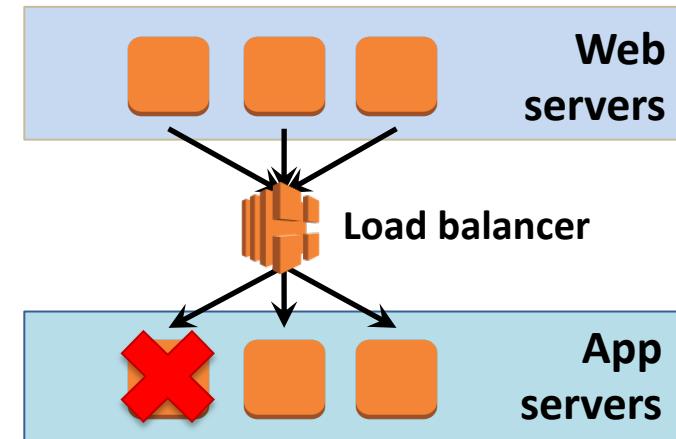
Decoupling

- The more loosely your system is coupled:
 - The more easily it scales.
 - The more fault-tolerant it can be.

Tightly coupled



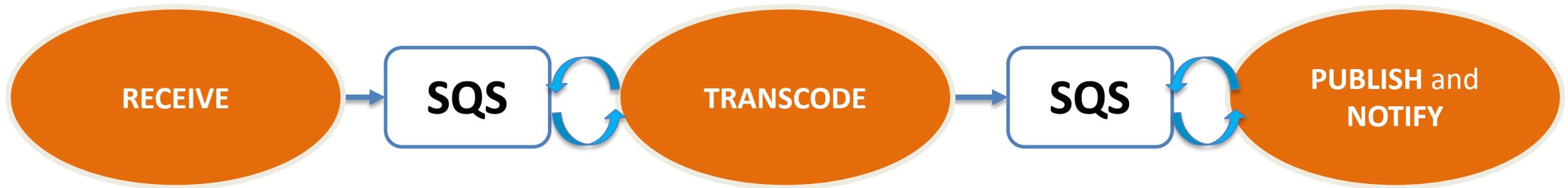
Loosely coupled



Tightly Coupled Systems



Loosely Coupled Systems





Amazon Simple Queue Service (SQS)

Amazon SQS is a fully managed [message queueing service](#). Transmit [any volume of messages](#) at [any level of throughput](#) without losing messages or requiring other services to be always available.

Message



- Generated by one component to be consumed by another.
- Can contain 256 KB of text in any format.

Amazon SQS



- Ensures delivery of each message at least once.
- Supports multiple readers and writers on the same queue.
- Does [not](#) guarantee first in, first out.

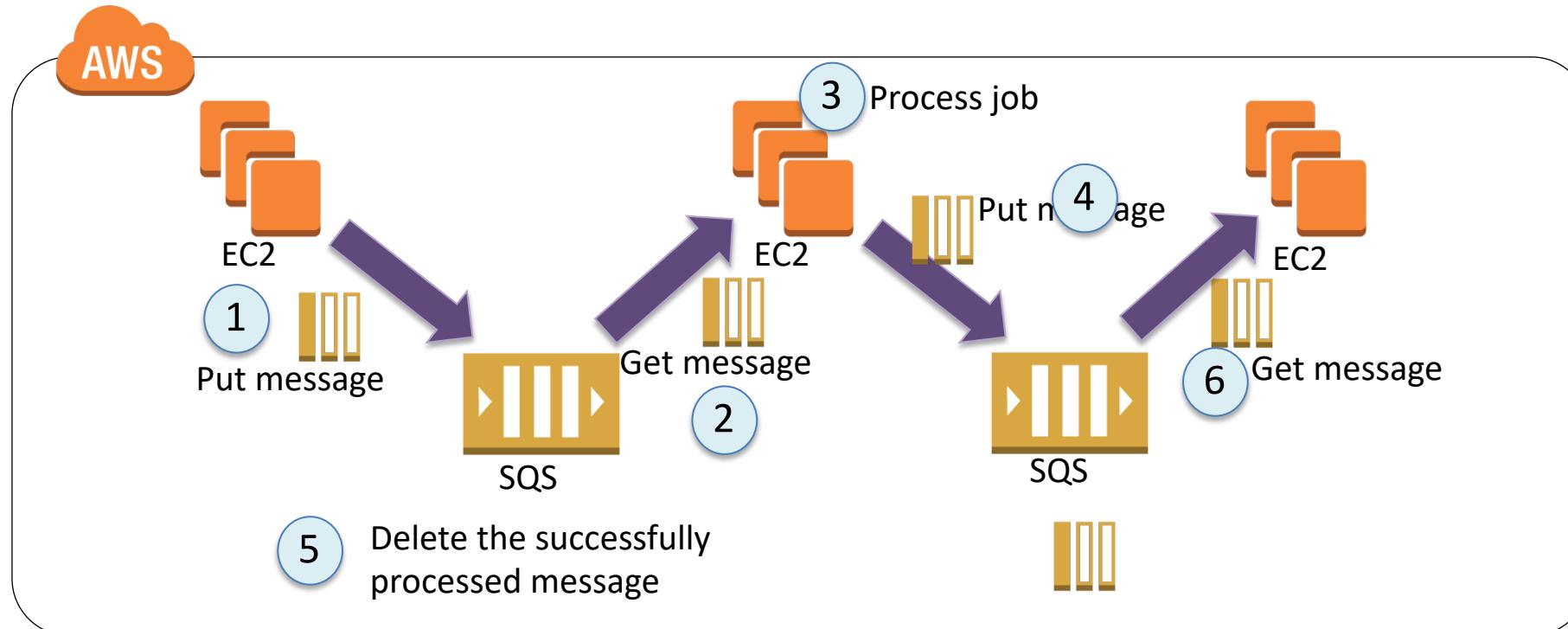
Queues



- Repository for messages awaiting processing.
- Acts as a buffer between the components which produce and receive data.

Loose Coupling with Amazon SQS

Asynchronous Message Processing



Why Choose SQS?

- Extremely scalable
 - Potentially millions of messages.
- Extremely reliable
 - All messages are stored redundantly on multiple servers and in multiple data centers.
- Simple to use
 - Messages get sent in, messages get pulled out.
- Simultaneous read/write
- Secure
 - API credentials are needed.

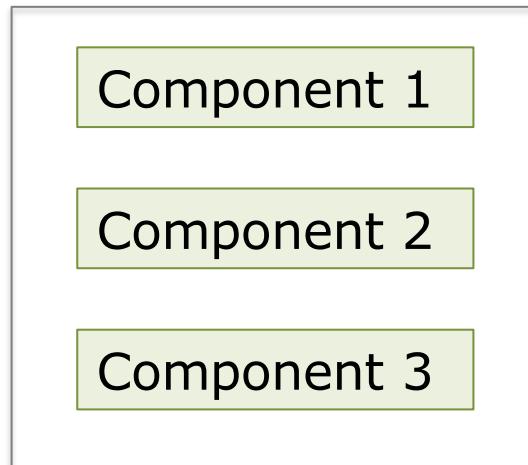
Properties of Distributed Queues



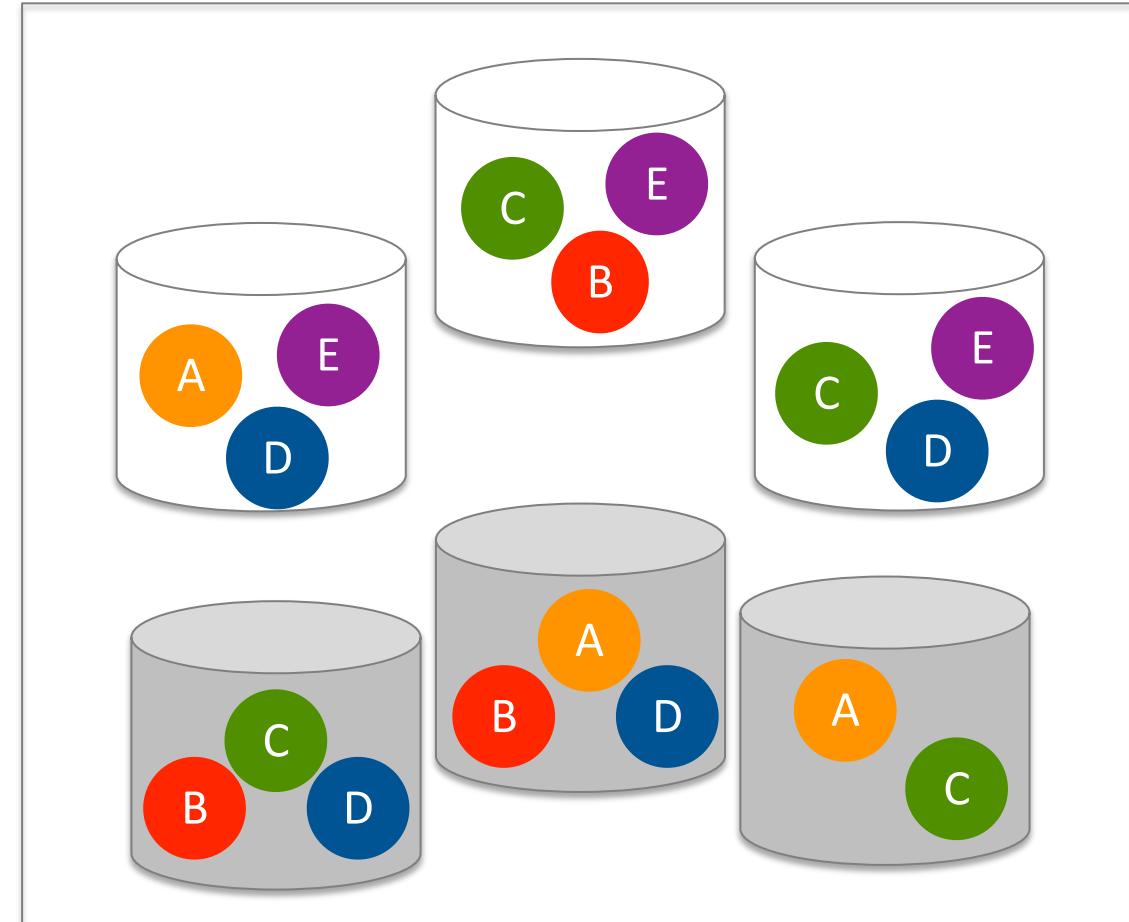
Your queue
(Distributed on SQS servers)

- Message Order
- At-Least-Once Delivery
- Message Sample

Your distributed system's components



Messages received from sampled servers



SQS Key Feature: Visibility Timeout

Visibility timeout prevents multiple components from processing the same message.

- When a message is received, it becomes “locked” while being processed. This prevents it from being processed by other components.
- The component that receives the message processes it and then deletes it from the queue.
- If the message processing fails, the lock will expire and the message will be available again (fault tolerance).

SQS Feature: Dead Letter Queues

- A dead letter queue is a queue of messages that **could not** be processed.
- Why use a dead letter queue?
 - It can sideline and isolate the unsuccessfully processed messages.

Note: A dead letter queue must reside in the same account and AWS region as the other queues that use the dead letter queue.

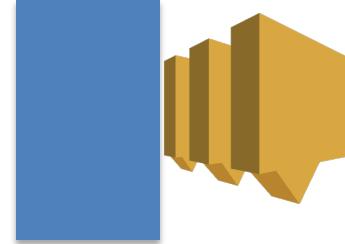
Sharing a Queue

- Shared queues
 - Queues can be shared with other AWS accounts and anonymously.
 - A **permission** gives access to another person to use your queue in some particular way.
 - A **policy** is the actual document that contains the permissions you granted.
- Who pays for shared queue access?
 - The queue **owner** pays for shared queue access.

Amazon SQS Use Cases

- Work queues
- Buffering batch operations
- Request offloading
- Fan-out
- Auto Scaling

Amazon Simple Notification Service (SNS)

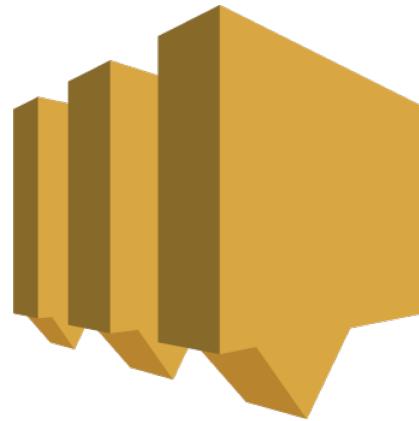


Amazon SNS enables you to [set up, operate, and send notifications](#) to subscribing services other applications.

- Messages published to topic.
- Topic subscribers receive message.
- Subscriber types:
 - Email (plain or JSON)
 - HTTP/HTTPS
 - Short Message Service (SMS) clients (USA only)
 - Amazon SQS queues
 - Mobile push messaging
 - AWS Lambda Function

Characteristics of Amazon SNS

- Single published message
- Order is not guaranteed
- No recall
- HTTP/HTTPS retry
- 256 KB max per message

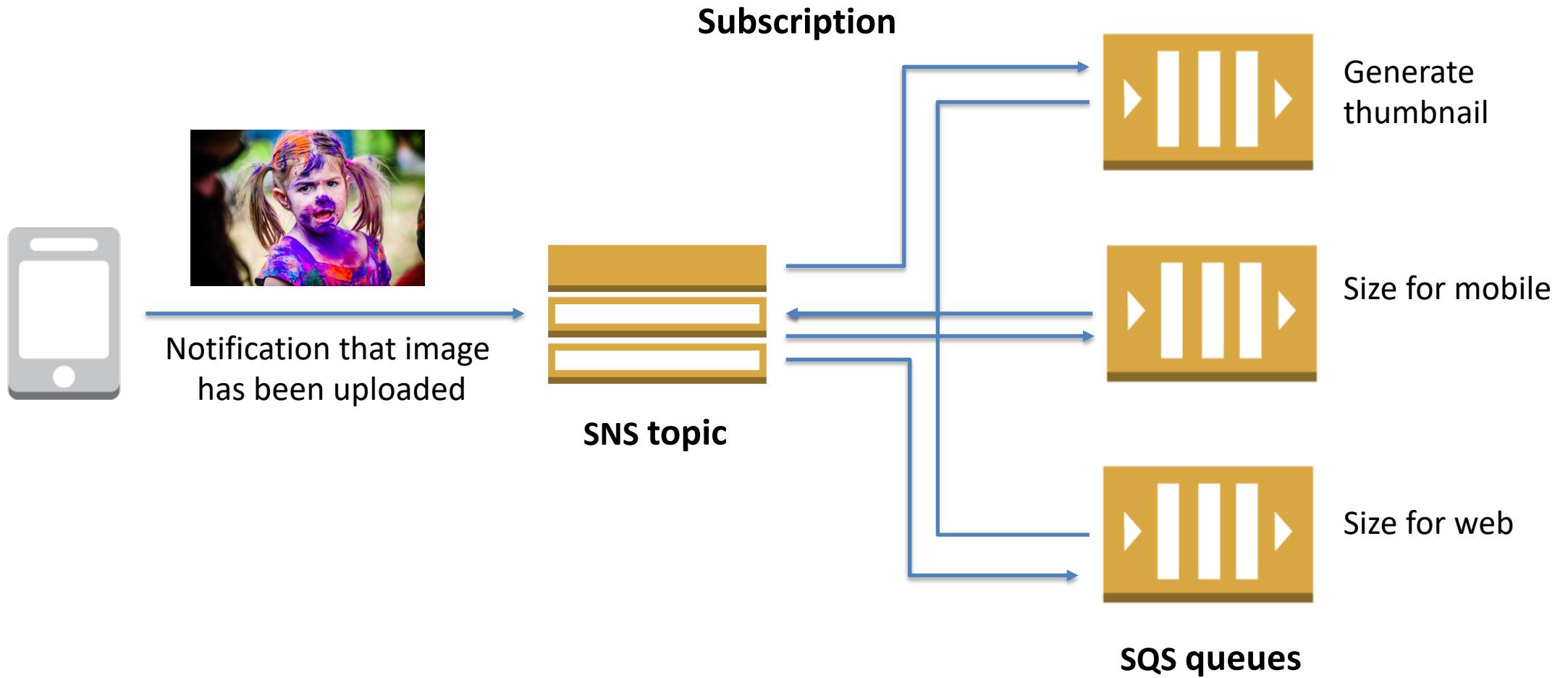


Amazon SNS v/s Amazon SQS

Amazon SQS and Amazon SNS are both messaging services within AWS.

	Amazon SNS	Amazon SQS
Message persistence	No	Yes
Delivery mechanism	Push (Passive)	Poll (Active)
Producer/consumer	Publish/subscribe	Send/receive

Use Case: Fan-Out



End-to-End Scenario: Image Processing

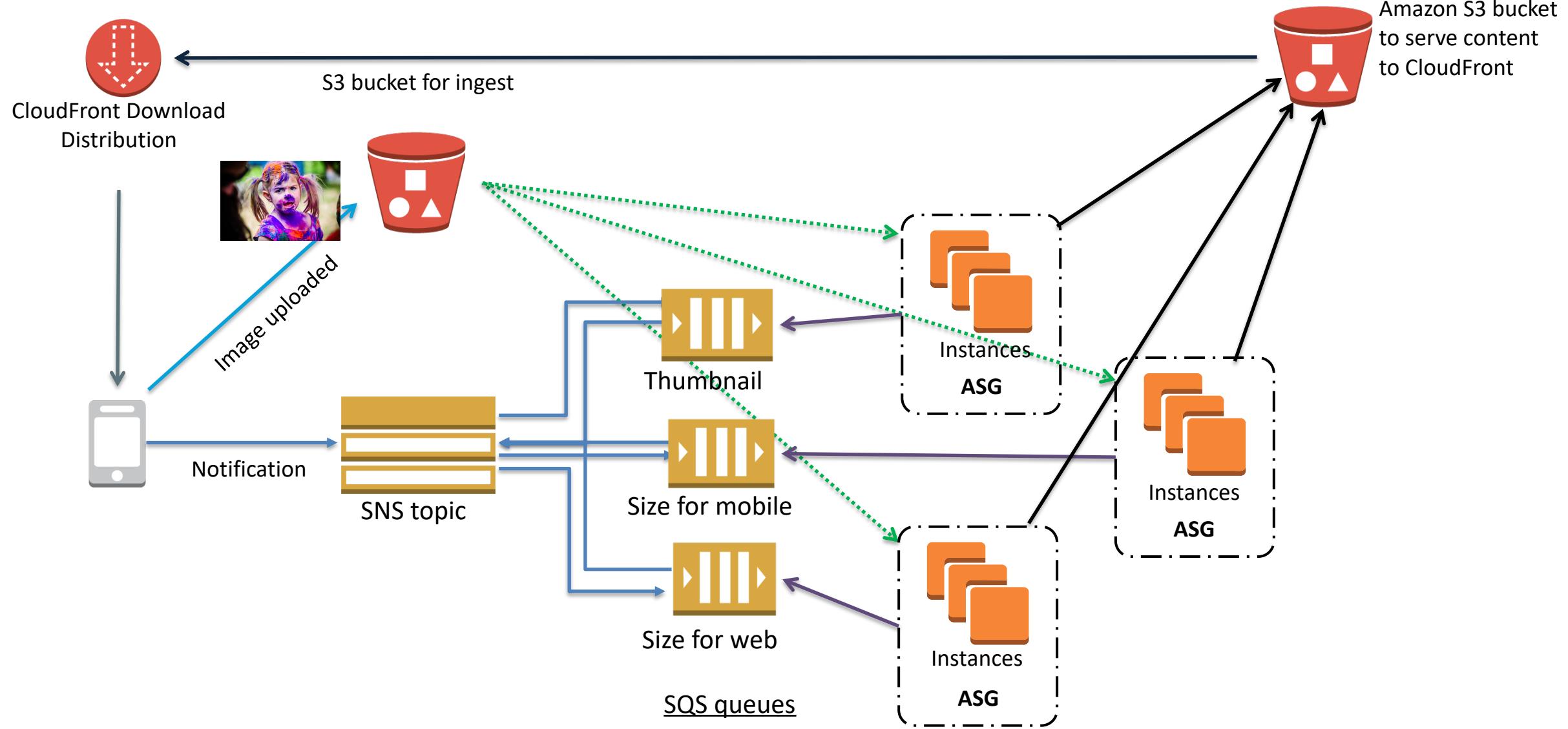
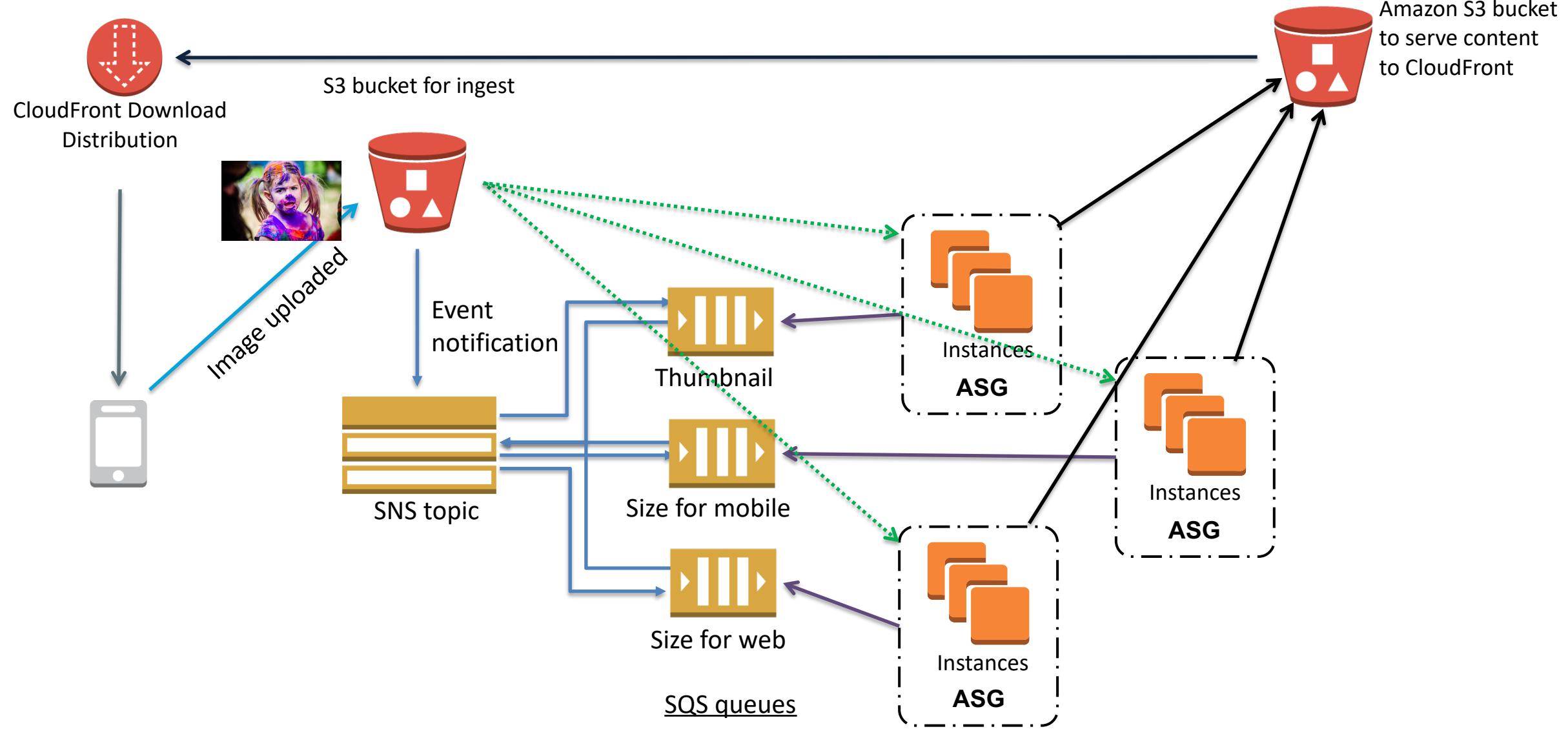


Image Processing – S3 Event Notifications



Serverless Microservices

Lambda, DynamoDB, API gateway

Architecting on AWS

Topics

- What is Microservice Architecture?
- AWS Lambda
- DynamoDB
- AWS API Gateway
- Architecture Example

Best Practice: Design Services, Not Servers

Managed services and serverless architectures can provide greater **reliability** and **efficiency** in your environment.

Anti-pattern

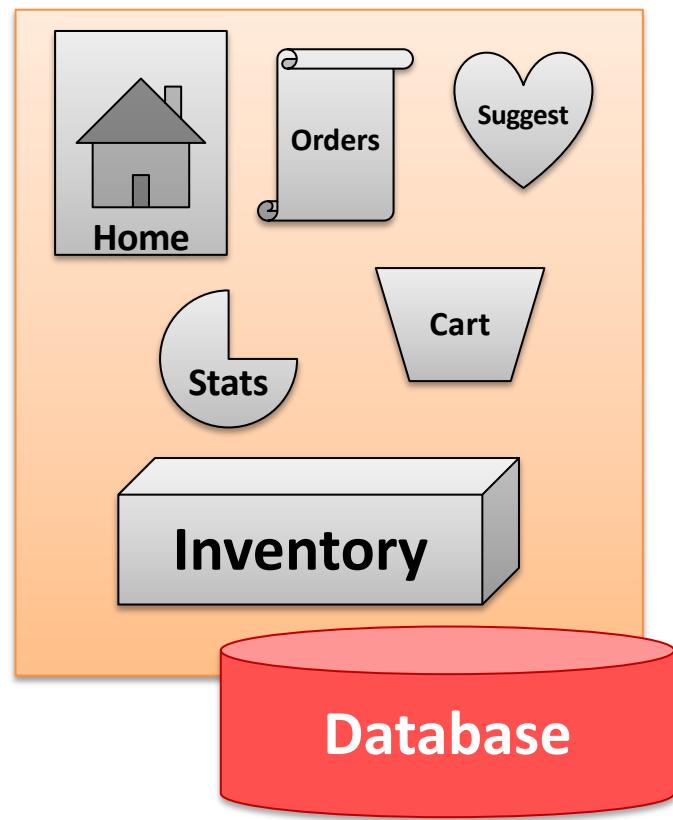
- Simple applications run on persistent servers.
- Applications communicate directly with one another.
- Static web assets are stored locally on instances.
- Back-end servers handle user authentication and user state storage.

Best practice

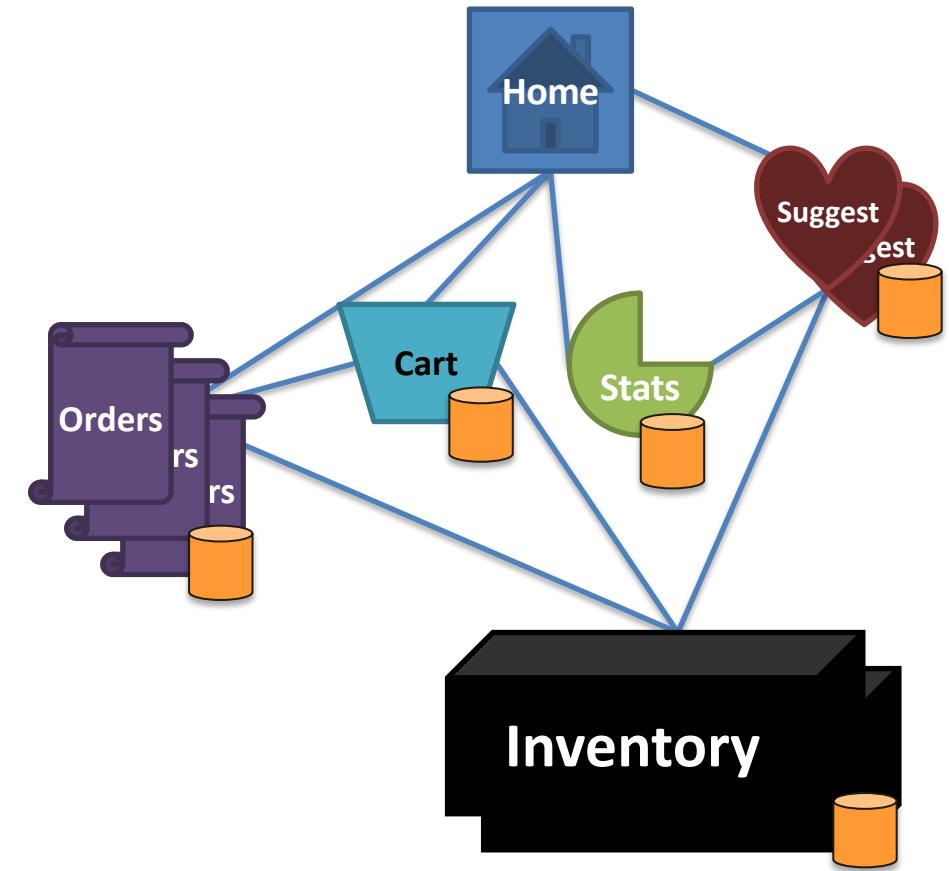
- Serverless solution is provisioned at time of need.
- Message queues handle communication between applications.
- Static web assets are stored externally, such as on Amazon S3.
- User authentication and user state storage are handled by managed AWS services.

Monolith vs Decouple Application

Traditional app architectures are
monolithic:



Microservice-based architectures are
loosely coupled:



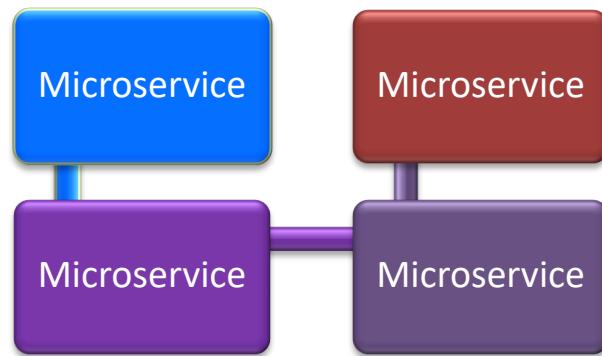
Service-Oriented Architecture (SOA)

- An architectural approach in which application components **provide services to other components** via a communications protocol.
- **Services** are self-contained units of functionality.

Microservices Architectures

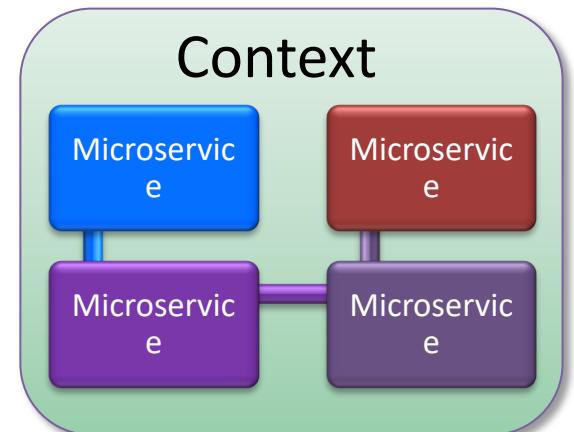
■ Microservices

- Small, independent processes within an SOA.
- Each process is focused on doing one small task.
- Processes communicate to each other using language-agnostic APIs.



Microservices Bounding Concept

- Each business domain can be divided into **contexts**.
 - Contexts are made up of microservices.
 - Each context has its own functions, objects, and language.
 - Contexts can comprise single operations like:
 - Transaction handling
 - User registration, authentication, tracking
 - Content publishing or syndication
 - Catalog, warehouse management



Best Practices

- Change components without breaking them:
 - The interface is a contract.
 - Modifying capabilities should not affect consumers.
- Use a simple API:
 - Lowers the cost of using your service.
 - More complexity means more resistance to change.
 - Abstraction: The less you share, the less will break.
- Keep it technology-agnostic

AWS LAMBDA



AWS Lambda



- Fully managed compute service that runs stateless code (Node.js, Java, Python, and C#) in response to an event or on a time-based interval.
- Allows you to run code without managing infrastructure like Amazon EC2 instances and Auto Scaling groups.

AWS Lambda

- AWS Lambda handles:
 - Servers
 - Capacity needs
 - Deployment
 - Scaling and fault tolerance
 - OS or language updates
 - Metrics and logging
- AWS Lambda enables you to:
 - Bring your own code (even native libraries).
 - Run code in parallel.
 - Create back ends, event handlers, and data processing systems.
 - Never pay for idling resources!

How to Use AWS Lambda

1. Upload your code to AWS Lambda (in .zip form)
 - You can also write your code directly into an editor in the console or import it from an Amazon S3 bucket.
2. Specify the event source.
3. Specify its necessary compute resources (128MB-1.5GB of memory).
4. Specify timeout period.
5. Specify the VPC whose resources it needs to access (if applicable).
6. Launch the function.

That's it. From there, AWS deploys and manages it for you.



Serverless Computing with Lambda

AWS Lambda starts code within milliseconds of an event such as:

- An image upload
- In-app activity
- A website click
- Output from a connected device
- Consider AWS Lambda if:
 - You're using entire instances to run simple functions or processing applications.
 - You don't want to worry about HA, scaling, deployment, or management.

Triggers for AWS Lambda Functions



Amazon
DynamoDB



Amazon
S3



Amazon
SNS



Amazon
Kinesis



Amazon
SES



AWS
Config



Scheduled Events



Amazon
Cognito



AWS SDKs via
Amazon API
Gateway



Amazon
CloudWatch



AWS
CodeCommit



AWS
CloudFormation

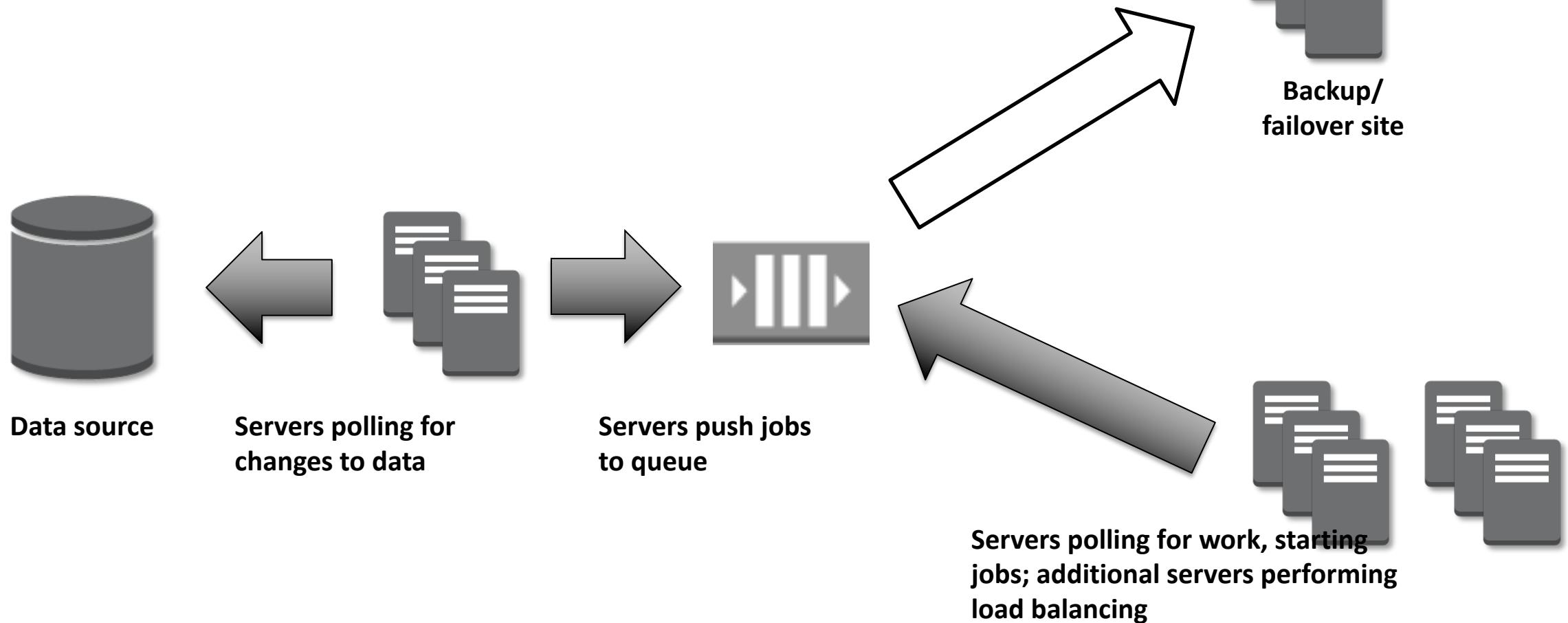


HTTPS via API
Gateway

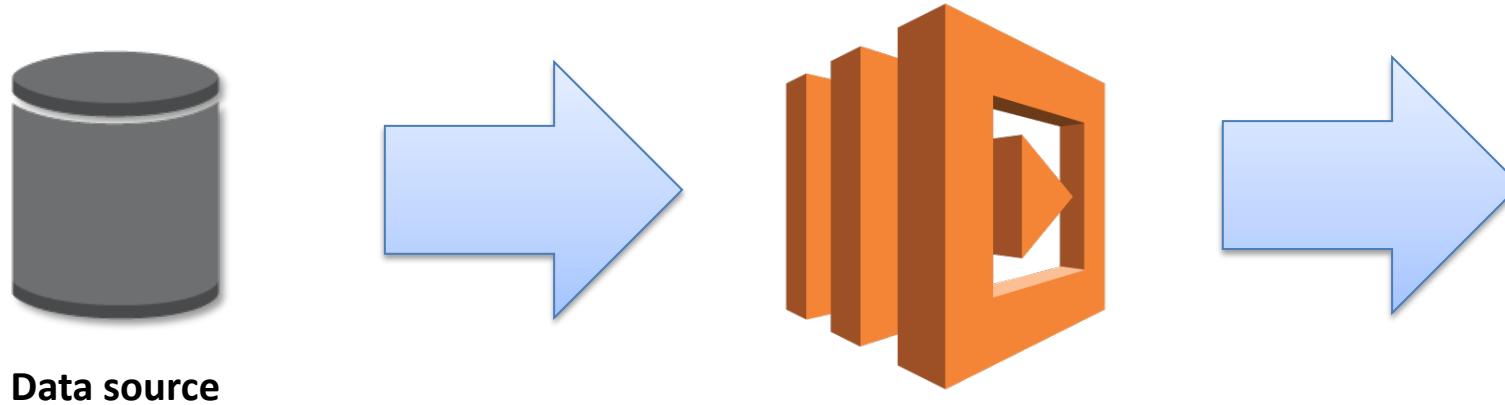
Resource Sizing with AWS Lambda

- AWS Lambda currently offers **23 different levels** of resource allocation, which range from:
 - 128 MB of memory and the lowest CPU power, to
 - 1.5 GB of memory and the highest CPU power
- **More resources = lower latency** for CPU-intensive workloads.
- Compute price **scales** with resource level.
- Functions can run **between 100 ms and five minutes** in length.
- **Free tier:** 1M free requests and 400,000 **GB-s/mo** of compute time.

Typical Data Processing Solution



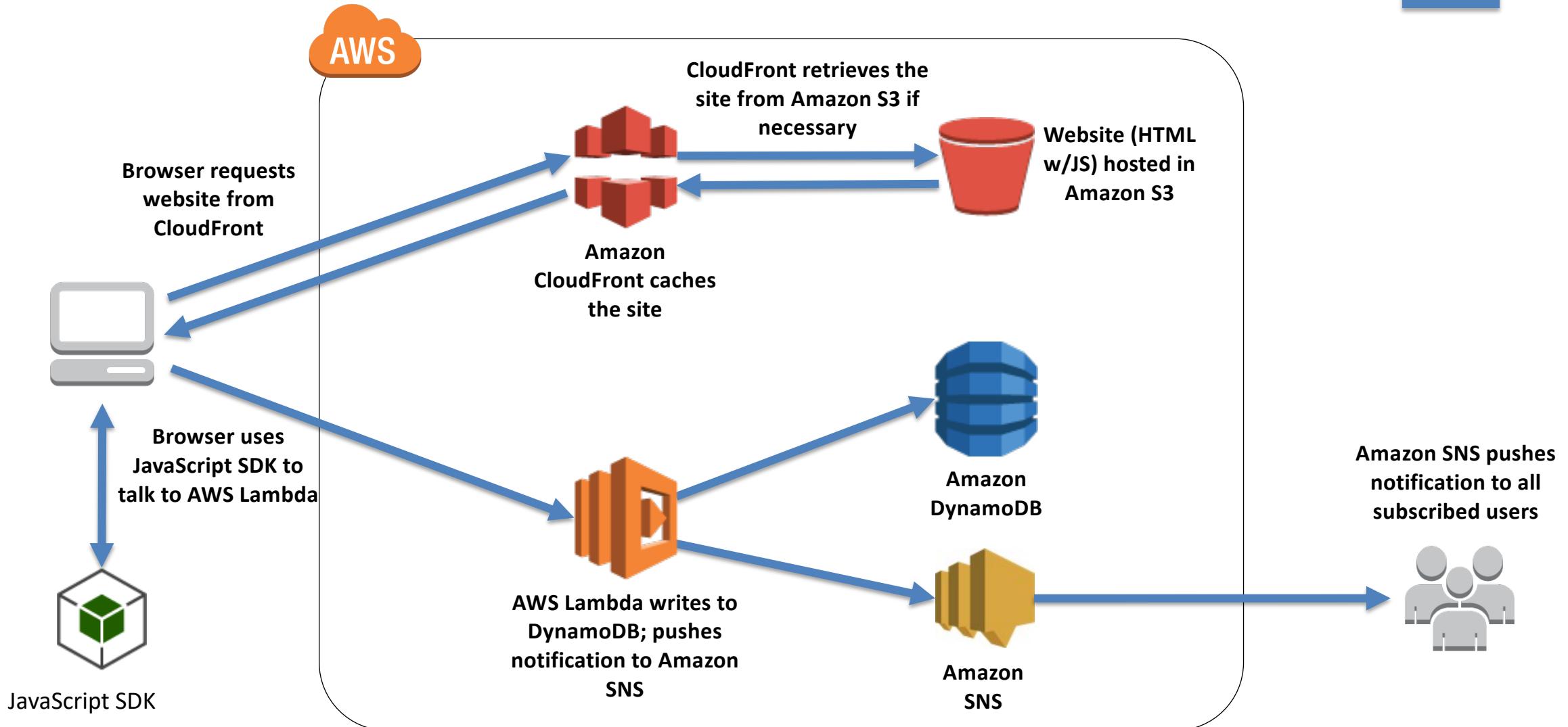
Data Processing with AWS Lambda



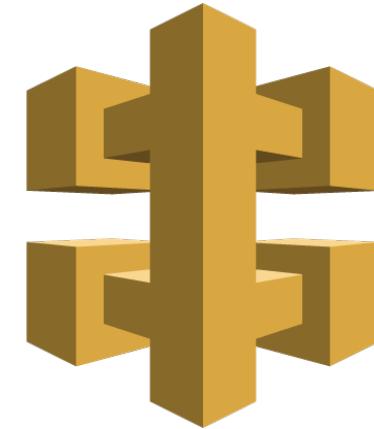
AWS Lambda handles:

- Listening/polling
- Queueing
- Processing
- Auto scaling
- Redundancy
- Load balancing

Using AWS Lambda as a Web Server



AMAZON API GATEWAY



Amazon API Gateway

- Allows you to [create APIs](#) that act as "front doors" for your applications [to access data, business logic, or functionality](#) from your back-end services.
- Fully managed and handles all tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls.
- Can handle workloads running on:
 - Amazon EC2
 - AWS Lambda
 - Any web application

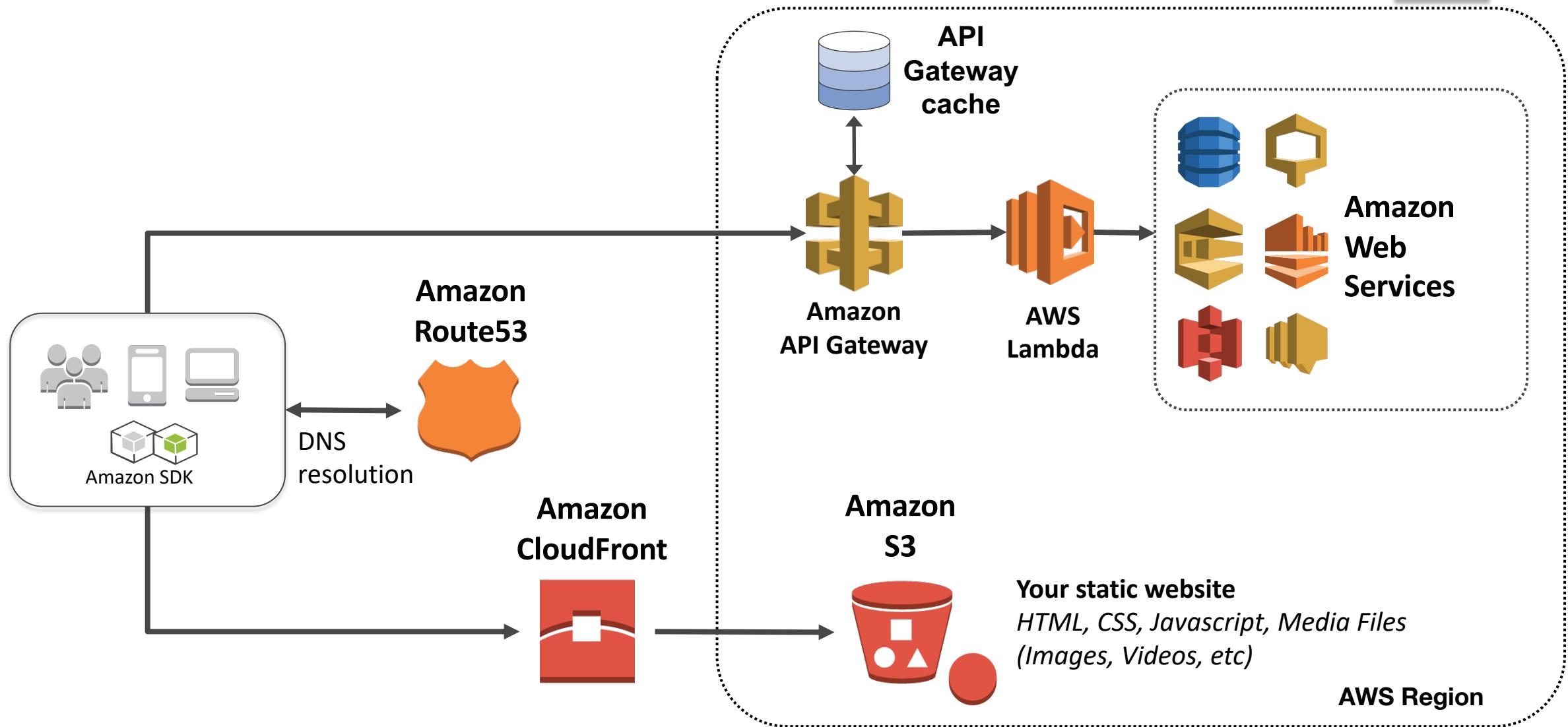
Features of Amazon API Gateway

- Host and use multiple versions and stages of your APIs.
- Create and distribute API keys to developers.
- Leverage signature version 4 to authorize access to APIs.
- Throttle and monitor requests to protect your back end.
- Deeply integrated with AWS Lambda.
- Integrates with the AWS Marketplace.

Benefits of Amazon API Gateway

- Managed cache to store API responses
- Reduced latency and Distributed Denial of Service (DDoS) protection through Amazon CloudFront
- SDK generation for iOS, Android, and JavaScript
- OpenAPI Specification (Swagger) support
- Request/response data transformation

Serverless Architecture Using API Gateway



Designing a Web-Scale Storage on AWS

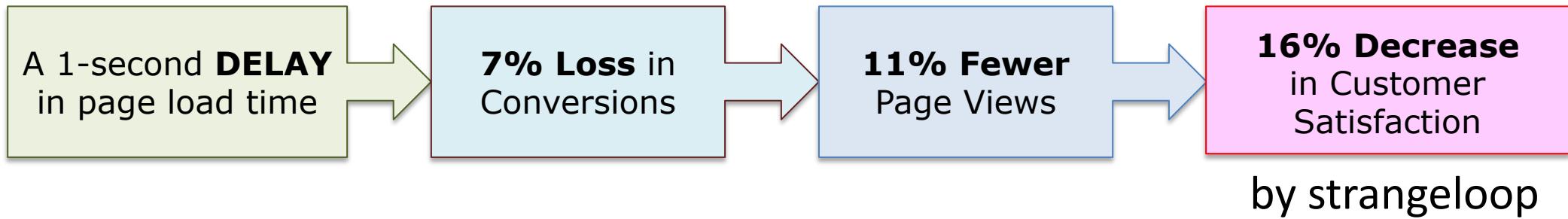
Architecting on AWS

Topics

- Types of Content to be Stored.
- CloudFront CDN
- Static Data Storage
- NoSQL Database
- Relational Database

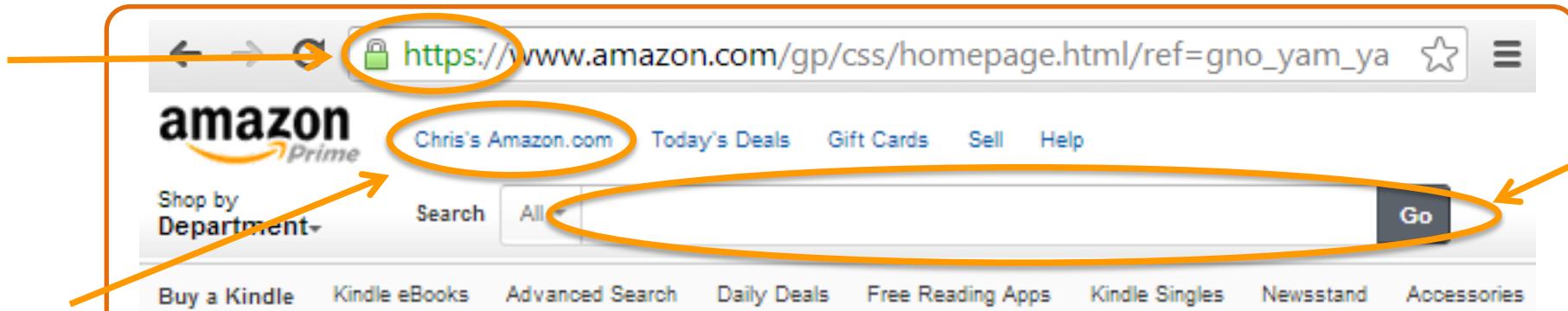
Reality of Web-based Applications

- Your application being unavailable :
 - Leads to revenue loss.
 - Has an impact on customer loyalty and your brand image.
- Performance translates to:
 - High page views.
 - Better customer experience.
 - Higher conversion rates.



Variety of Content on a Website

Secure



**Dynamic
(Zero TTL)**

Static

Can be
cached!



Revolutionary on-device
tech support

Exclusively on Kindle Fire HDX tablets—live on-device tech support from an Amazon expert is just a tap away with the new "Mayday" button

Live Support with Mayday

NEW—Simply tap the "Mayday" button to be connected for free to an Amazon expert who can co-pilot you through any feature by drawing on your screen, walking you through how to do something yourself, or doing it for you—whatever works best. Mayday is available 24x7, 365 days a year, and it's free. Throughout the process, you will be able to see your Amazon Tech advisor live on your screen, but they won't see you. 15 seconds or less is the Mayday response time goal.

Watch it in Action

Kindle Fire HDX is easy to use right out of the box. But when you need extra assistance, the "Mayday" button is there. See how it works in these short commercials.



User
Input

Video

Designing Web-Scale Storage

1. Store static assets in Amazon S3.

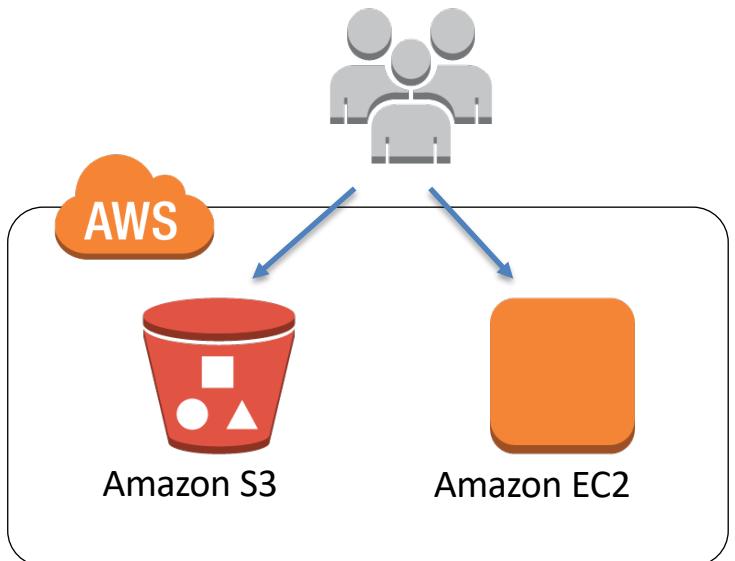
Web Storage Pattern

■ Problem

- Delivering large files from a web server could become a problem in terms of network latency.
- User-generated content needs to be distributed across all your web servers.

■ Solution

- Store static asset files (CSS, multimedia, etc.) in Amazon S3 and deliver the files from there.
- Objects that are stored in Amazon S3 can be accessed directly by users if set to being public.



Why Store Static Content in S3?

- Items stored in Amazon S3 are highly durable:
 - Amazon S3 **redundantly stores** your objects in multiple facilities in a region.
 - Amazon S3 **verifies the integrity** of your data using checksums.
 - Versioning allows you to **preserve, retrieve, and restore every version** of every object.
- The Amazon S3 data consistency model:
 - Provides **read-after-write** consistency for PUTS of new objects.
 - Provides **eventual consistency** for overwrite PUTS and Deletes.

Best Practice

- Choosing a region:
 - Performance depends on the [proximity](#) to your users.
 - Co-locating with compute and other AWS resources affects performance.
- Pay attention to your [object naming scheme](#) if:
 - You want consistent performance from a bucket.
 - You want a bucket capable of routinely exceeding 100 PUT/LIST/DELETE or [300 GET requests per second](#).

If Bucket is Under Heavy Load...

- Amazon S3 automatically partitions your buckets according to the prefixes of your files.

Partition 1

```
/album1/photo1.jpg  
/album1/photo2.jpg  
/album1/photo3.jpg  
/album1/photo4.jpg
```



If a process requests the photos from album1, all requests will go to the same partition.

Partition 2

```
/album2/photo1.jpg  
/album2/photo2.jpg  
/album2/photo3.jpg  
/album2/photo4.jpg
```

Anti-pattern – Hot Partitioning

If Bucket is Under Heavy Load...

Add a hex hash prefix to the key to reduce the chance that a request reads from the same partition multiple times at once:

Partition 1

/2e4f/album2/photo7.jpg

/3a79/album1/photo2.jpg

/7b54/album1/photo3.jpg

/8761/album2/photo6.jpg



Partition 2

/921c/album1/photo4.jpg

/b71f/album1/photo1.jpg

/ba65/album2/photo5.jpg

/c34a/album2/photo8.jpg

If a process requests the first four photos, the requests will be *split* between the partitions.



Best practice

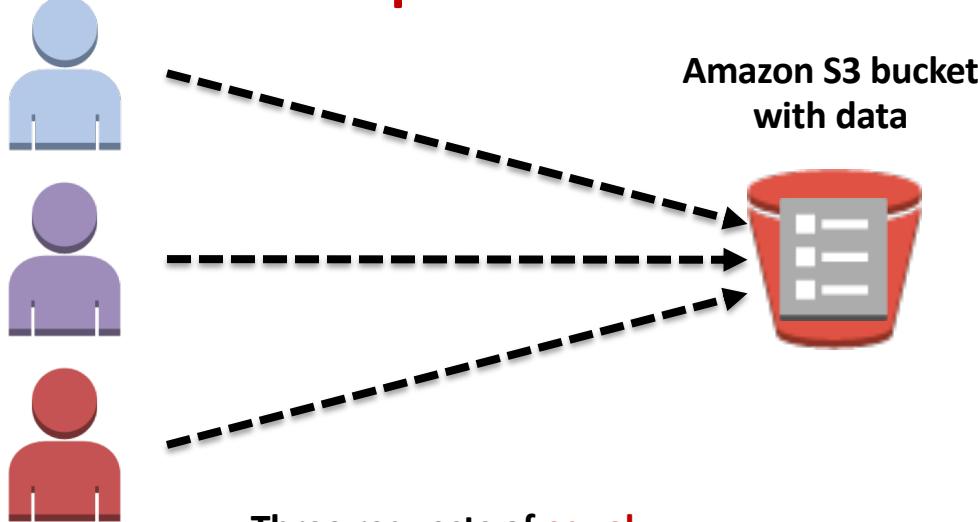
Designing Web-Scale Storage

1. Store static assets in Amazon S3.
2. **Serve frequently accessed assets from Amazon CloudFront.**

Best Practice: Use Caching

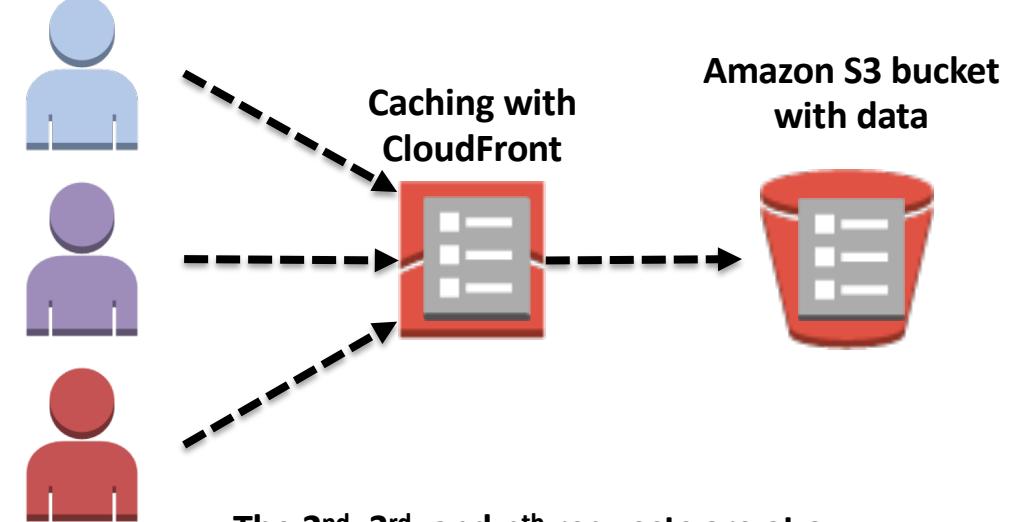
Use caching to minimize redundant data retrieval operations.

Anti-pattern



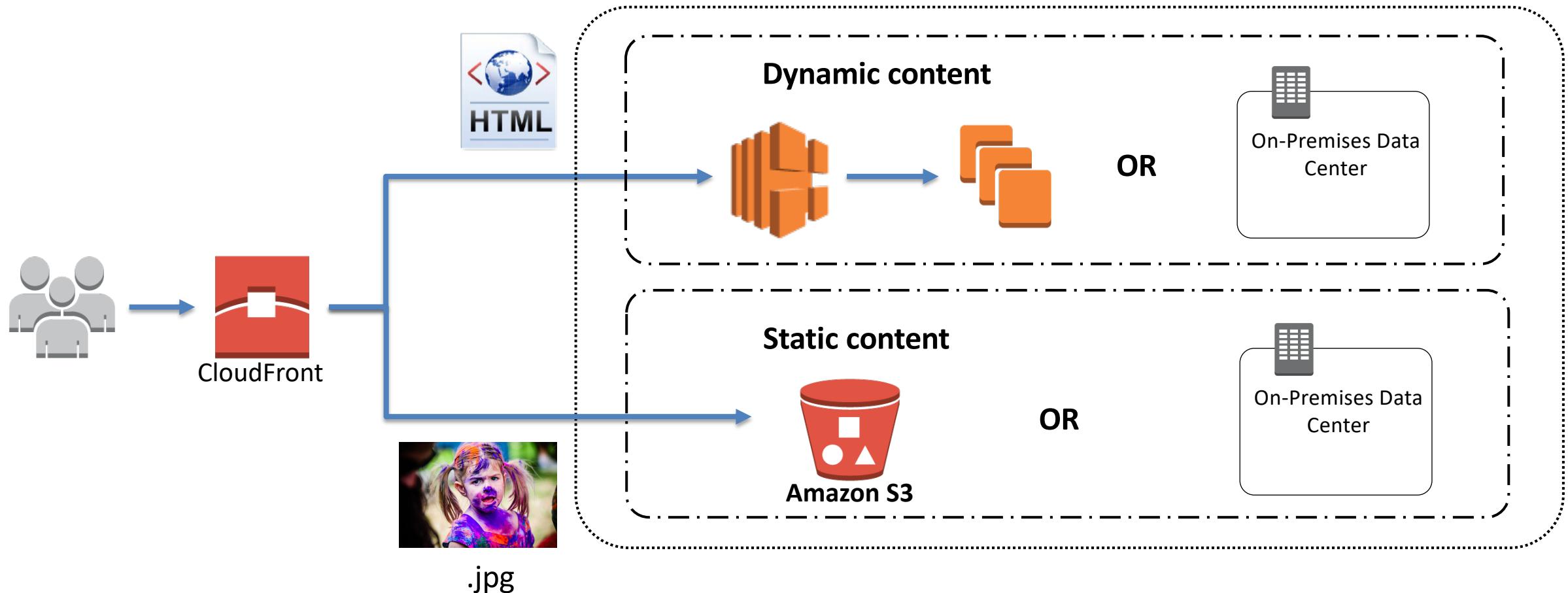
Three requests of **equal** latency and cost

Best practice



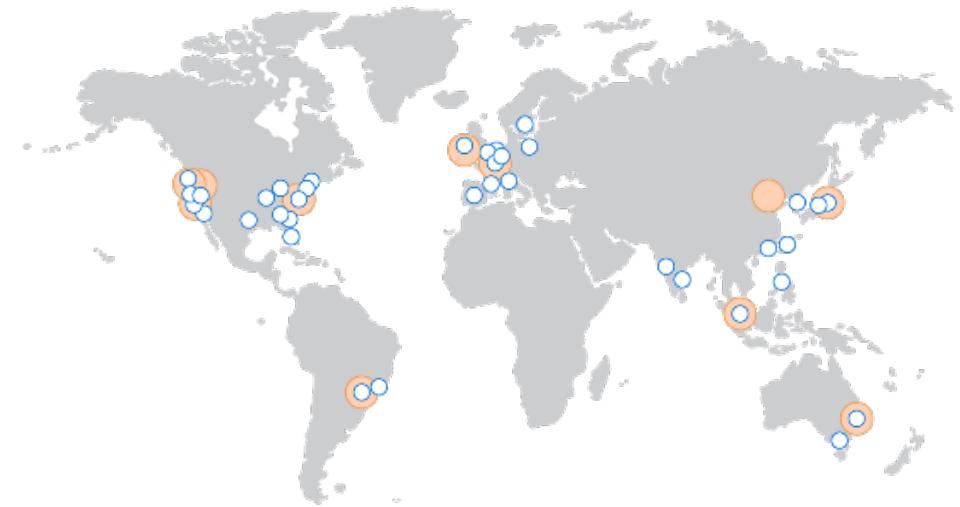
The 2nd, 3rd, and nth requests are at a **lower** latency and cost

Cache Static and Re-usable Content



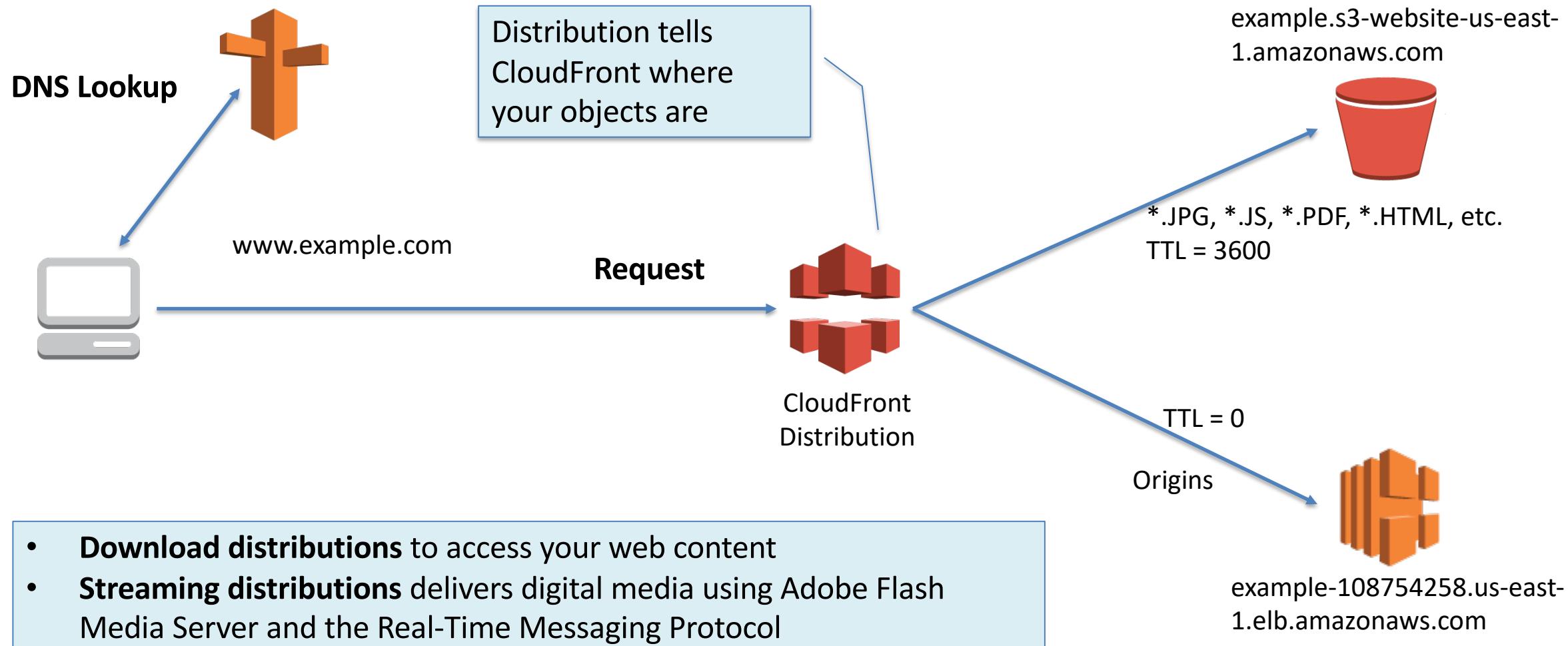
How Can Amazon CloudFront Help?

- Content Delivery Network (CDN)
 - Your content can be cached all around the world.
 - Better user experience.
 - Less stress on your core infrastructure.
- Key Features
 - TCP/IP optimizations for the network path.
 - SSL/TLS termination close to viewers.
 - POST/PUT upload optimizations.
 - Latency-based routing.
 - Regional Edge Caches



Independent of Region

CloudFront Distributions



How Do You Enable CloudFront?

- Use a separate CNAME for static content.
 - Static content is cached, dynamic content straight from origin.
 - More effort to set up and manage.
- Point entire URL to CloudFront.
 - Easiest to manage.
 - Use URL patterns to stage dynamic content.
 - ALL content goes through edge locations.

How to Expire Contents

- Time to Live (TTL)
 - Fixed period of time (expiration period).
 - Set by *you*.
 - GET request to origin from CloudFront will use **If-Modified-Since** header.
- Change object name
 - **Header-v1.jpg** becomes **Header-v2.jpg**.
 - New name forces refresh.
- Invalidate object
 - Last resort: very inefficient and very expensive.

Designing Web-Scale Storage

1. Store static assets in Amazon S3.
2. Serve frequently accessed assets from Amazon CloudFront.
- 3. Store non-relational data in a NoSQL database such as Amazon DynamoDB.**

Choose the Right Database Solutions

Choose from an array of [relational database](#) engines, [NoSQL](#) solutions, [data warehousing](#) options, and [search-optimized](#) data stores.

Things to consider:

- Read and write needs
- Total storage requirements
- Typical object size and nature of access to these objects
- Durability requirements
- Latency requirements
- Maximum concurrent users to support
- Nature of queries
- Required strength of integrity controls

Relational/SQL and NoSQL Databases

	Relational/SQL	NoSQL
Data Storage	Rows and Columns	Key-Value, Documents, and Graphs
Schemas	Fixed	Dynamic
Querying	Using SQL	Focused on collection of documents
Scalability	Vertical	Horizontal

Relational/SQL

ISBN	Title	Author	Format
9182932465265	Cloud Computing Concepts	Wilson, Joe	Paperback

NoSQL

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

NoSQL Databases

- Can be an alternative to relational databases for some types of applications.
- Can process large amounts of data with high availability (depending on the NoSQL solution, configuration, and architecture).
- Form a broad category with different implementations and data models.
- Have the common feature of distributed fault tolerance.

NoSQL Databases

- Does your app need transaction support, ACID compliance, joins, SQL?
- Can it do without these for all, some, or part of its data model
 - Refactor database hotspots to NoSQL solutions.
- NoSQL databases can offer increases in flexibility, availability, scalability, and performance.

NoSQL Databases

- NoSQL on Amazon EC2:
 - Cassandra, HBase, Redis, MongoDB, Couchbase, and Riak
- Managed NoSQL:
 - Amazon DynamoDB
 - ElastiCache with Redis
 - Amazon Elastic Map Reduce supports HBase

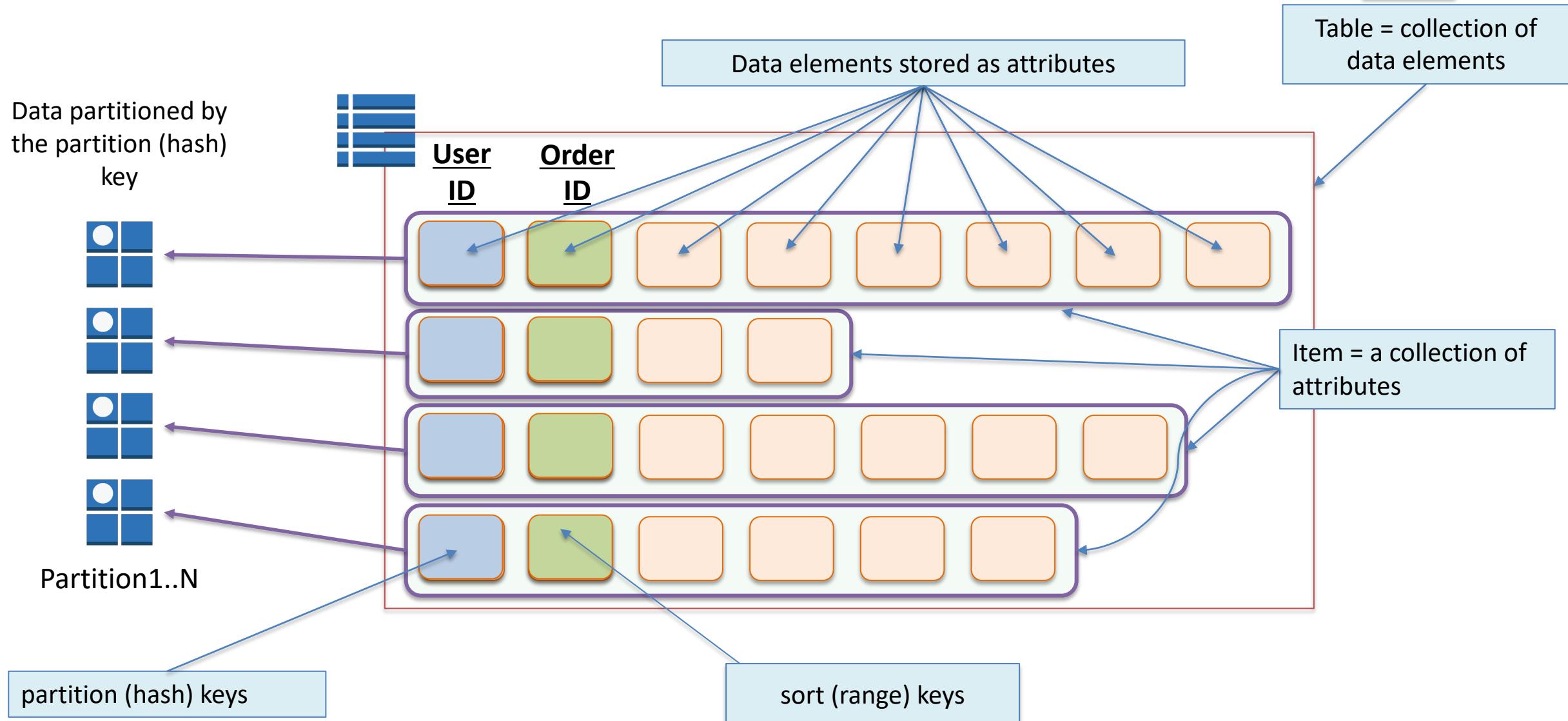
NoSQL Use Cases

- Think about when you need NoSQL instead of SQL.
- Leverage managed services like Amazon DynamoDB.
- **Use cases:**
 - Leaderboards and scoring
 - Rapid ingest of clickstream or log data
 - Temporary data needs (cart data)
 - Hot tables
 - Metadata or lookup tables
 - Session data

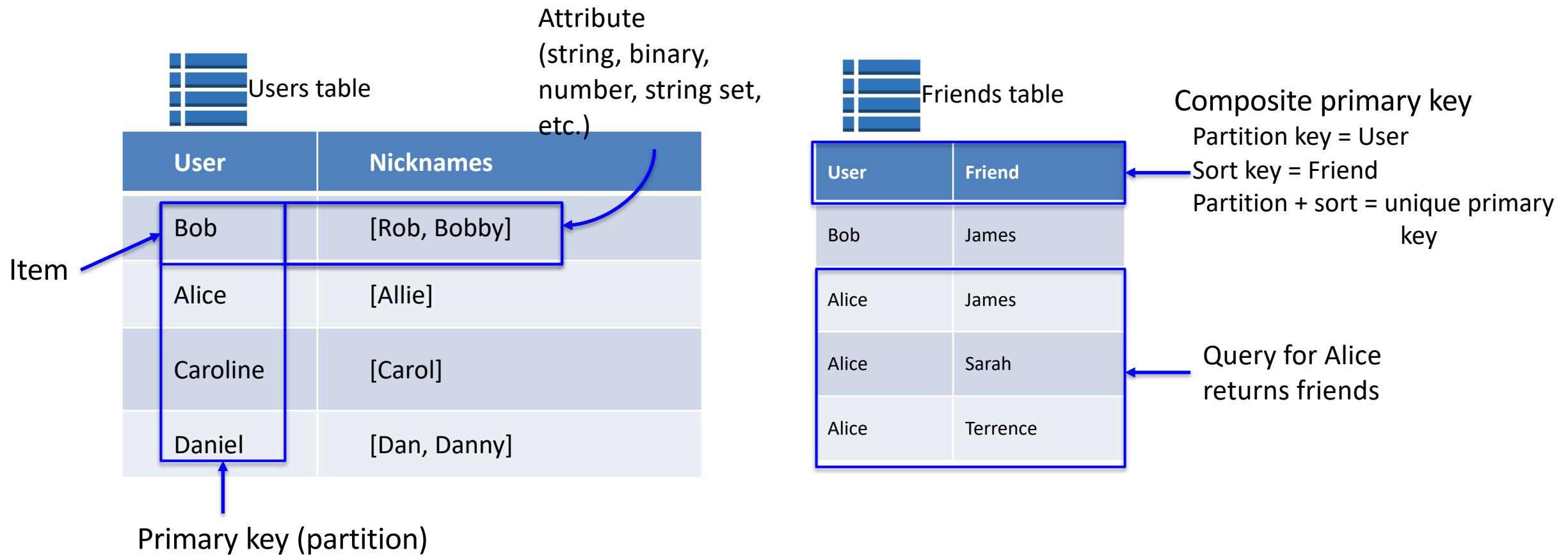
Amazon DynamoDB

- Low latency
 - SSD-based storage nodes
 - Latency = single-digit milliseconds
- Massive and seamless scalability
 - No table size or throughput limits
 - Live repartitioning for changes to storage and throughput
- Predictable performance
 - Provisioned throughput model
- Durable and available
 - Consistent, disk-only writes
- Fully managed NoSQL database service — **Zero Administration!**

Amazon DynamoDB Data Model



Use Case: Social Network



Amazon DynamoDB Consistency

- Synchronously replicates data across three facilities in an AWS Region.
- Read consistency: manner and timing of the read operation of a successfully written or updated data item.
- You can specify, at the time of the read request, whether a read should be **eventually** or **strongly** consistent.

Amazon DynamoDB Best Practices

- Keep item size small.
- Store metadata in DynamoDB and large BLOBs in Amazon S3.
- Use table per day, week, month, etc., for storing time series data.
- Use conditional or Optimistic Concurrency Control (OCC) updates.
- Avoid hot keys and hot partitions.

Designing Web-Scale Storage

1. Store static assets in Amazon S3.
2. Serve frequently accessed assets from Amazon CloudFront.
3. Store non-relational data in a NoSQL database such as Amazon DynamoDB.
- 4. Store relational data in Amazon RDS.**

Well Architected Framework

Architecting on AWS

Well Architected Framework

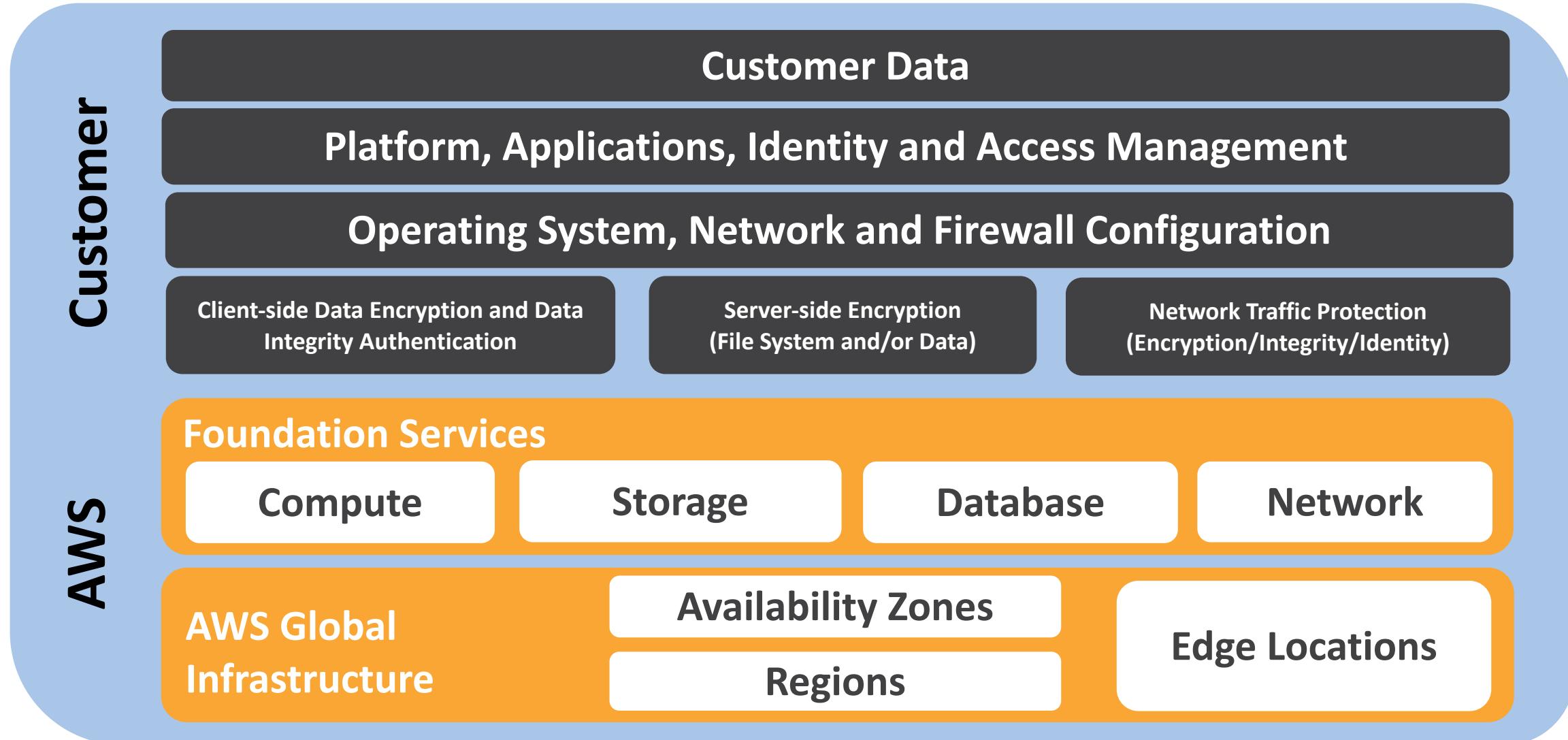
- Security
- Cost Optimization
- Reliability
- Performance
- Operational Efficiency

Security

Topics

- Account Security - Detective Controls
- Network Security
- Data Security - Encryption
- DDOS Mitigation

Shared Responsibility – AWS



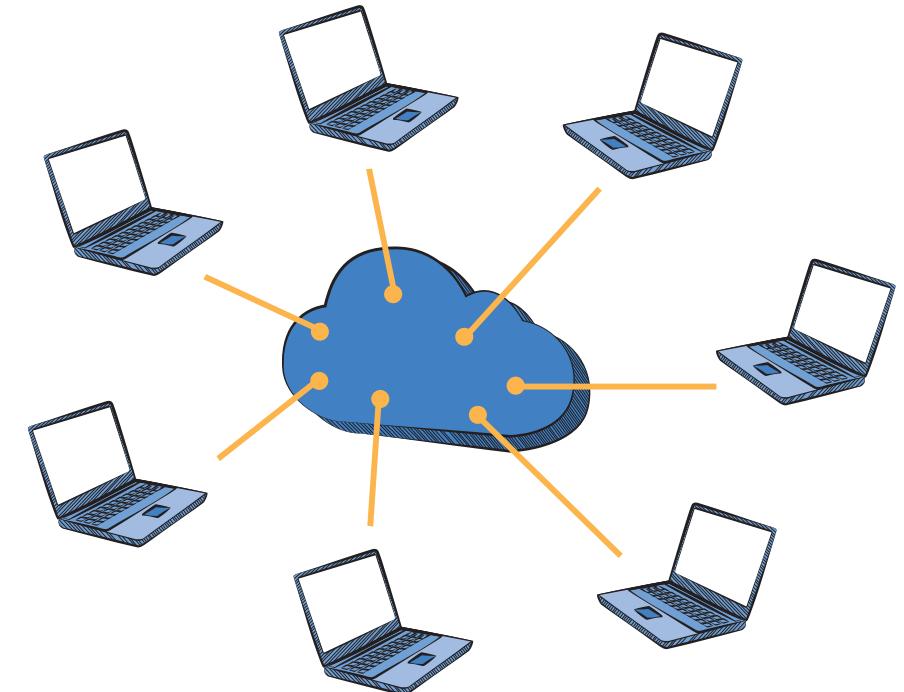
Physical Security

- 24/7 trained security staff
- AWS data centers in nondescript and undisclosed facilities
- Two-factor authentication for authorized staff
- Authorization for data center access



Hardware, Software, and Network

- Automated change-control process
- Bastion servers that record all access attempts
- Firewall and other boundary devices
- AWS monitoring tools



Certifications and Accreditations



ISO 9001, ISO 27001, ISO 27017, ISO 27018, IRAP (Australia), MLPS Level 3 (China), MTCS Tier 3
Certification (Singapore) and more ...

Best Practices

Build security into every layer of your infrastructure.

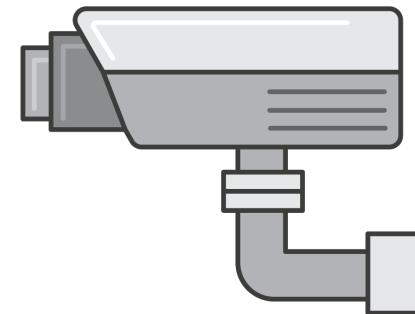
Things to consider:

- Isolate parts of your infrastructure
- Encrypt data in transit and at rest
- Enforce access control granularly, using the principle of least privilege
- Use multi-factor authentication
- Leverage managed services
- Log access of resources
- Automate your deployments to keep security consistent

Well-Architected Aspect: Security

The ability to **protect information**, systems, and assets while delivering **business value** through risk assessments and mitigation strategies.

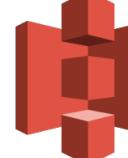
- Identity and access management
- Detective controls
- Infrastructure protection
- Data protection
- Incident response



Security: Design Principles

- Enable traceability
- Implement a principle of least privilege
- Focus on securing your system
- Automate security best practices

Key Services for Security

Areas	Key Services				
Data protection	 Elastic Load Balancing				 AWS Key Management Service (KMS)
Privilege management	 AWS IAM	 MFA token			
Infrastructure protection	 Amazon VPC				
Detective controls	 AWS CloudTrail	 AWS Config	 Amazon CloudWatch		

Amazon VPC Flow Logs

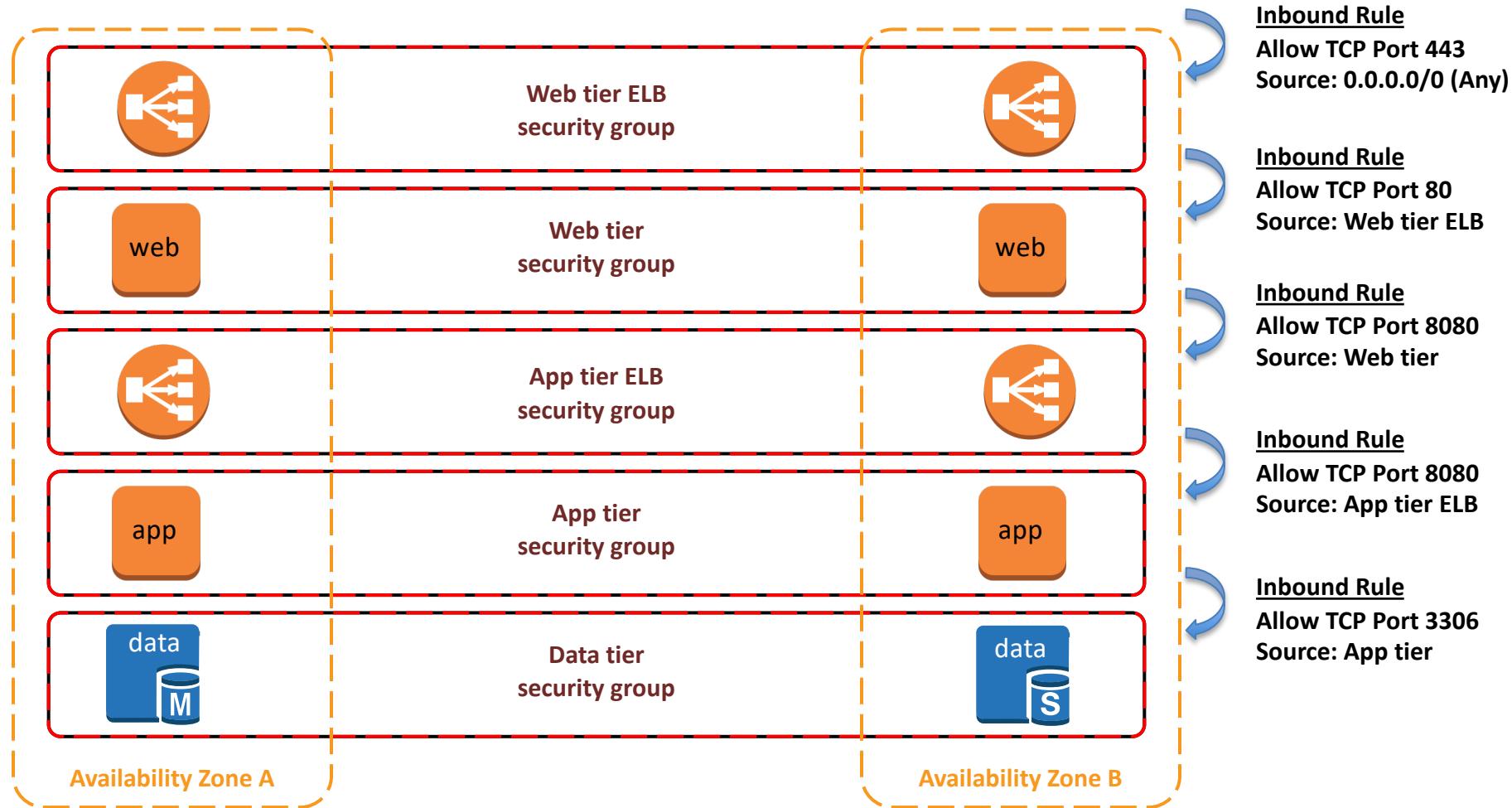
- Captures traffic flow details in your VPC.
 - Accepted and rejected traffic
- Can be enabled for VPCs, subnets, and ENIs.
- Logs published to CloudWatch Logs.

Use cases:

- Troubleshoot connectivity issues.
- Test network access rules.
- Monitor traffic.
- Detect and investigate security incidents.

Security Group Chaining

Security group rules per tier



DDoS (Distributed Denial of Service) Attack

- A Denial of Service (DoS) attack attempts to make your website or application **unavailable** to your end users.
- To achieve this, attackers use a variety of techniques that **consume network or other resources**, thus interrupting access for legitimate end users.
- The attackers use **multiple hosts** to orchestrate an attack against a target.

DDoS Protection

Protecting against attacks is a **shared responsibility** between AWS and you.

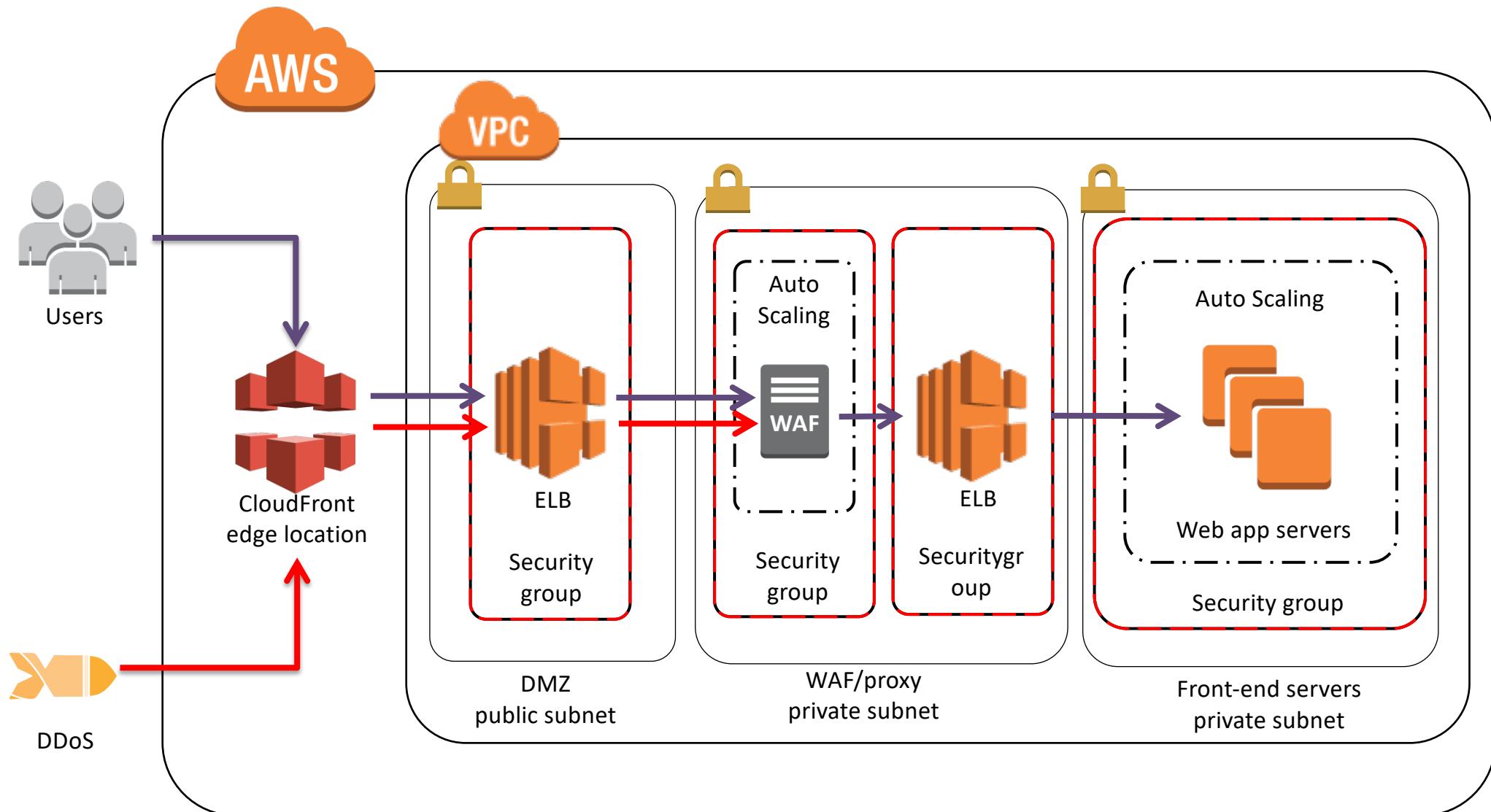
AWS

- AWS API endpoints are hosted on large, Internet-scale, world-class infrastructure.
- Proprietary DDoS mitigation techniques are used.
- AWS networks are multi-homed across a number of providers to achieve Internet access diversity.

Customer

- Front your application with AWS Services
- Safeguard exposed resources
- Minimize the attack surface
- Evaluate soft limits and request increases ahead of time
- Learn normal behavior
- Create a plan for attacks

DDoS Mitigation Example



Securing Data

CloudFront Custom SSL Support

- By default, your content is delivered to viewers over HTTPS by using a CloudFront distribution domain name such as <https://dxxxxx.cloudfront.net/image.jpg>
- Custom SSL certificate support features let you use your own domain name and your own SSL certificate.
- Server Name Indication (SNI) Custom SSL
 - Allows multiple domains to serve SSL traffic over the same IP address.
- Dedicated IP Custom SSL
 - To deliver content to browsers that do not support SNI.

How to Make Content Private

- Restrict access to objects in your Amazon S3 bucket.
- Require that users use signed URLs.
 - Create CloudFront key pairs for your trusted signers.
 - Write the code that generates signed URLs.
 - Typically, you'll write an application that automatically generates signed URLs.
 - Alternatively, use a web interface to create signed URLs.
 - Add trusted signers to your distribution.
 - **Note:** Once you add a trusted signer to your distribution, users must use signed URLs to access the corresponding content.

Origin Access Identity to Restrict Access

- Restrict access to Amazon S3 content by creating an **origin access identity (OAI)**, which is a special CloudFront user.
 - CloudFront origin access identity gets the objects from Amazon S3 on your users' behalf.
 - Direct access to the objects through Amazon S3 URLs will be denied.
- Procedure:
 1. Create an origin access identity and add it to your distribution.
 2. Change the permissions either on your Amazon S3 bucket or the objects in your bucket so that only the origin access identity has read permission.

IAM to Control API Access to CloudFront

Control a user's API access to CloudFront with IAM policies.

Group policy example:

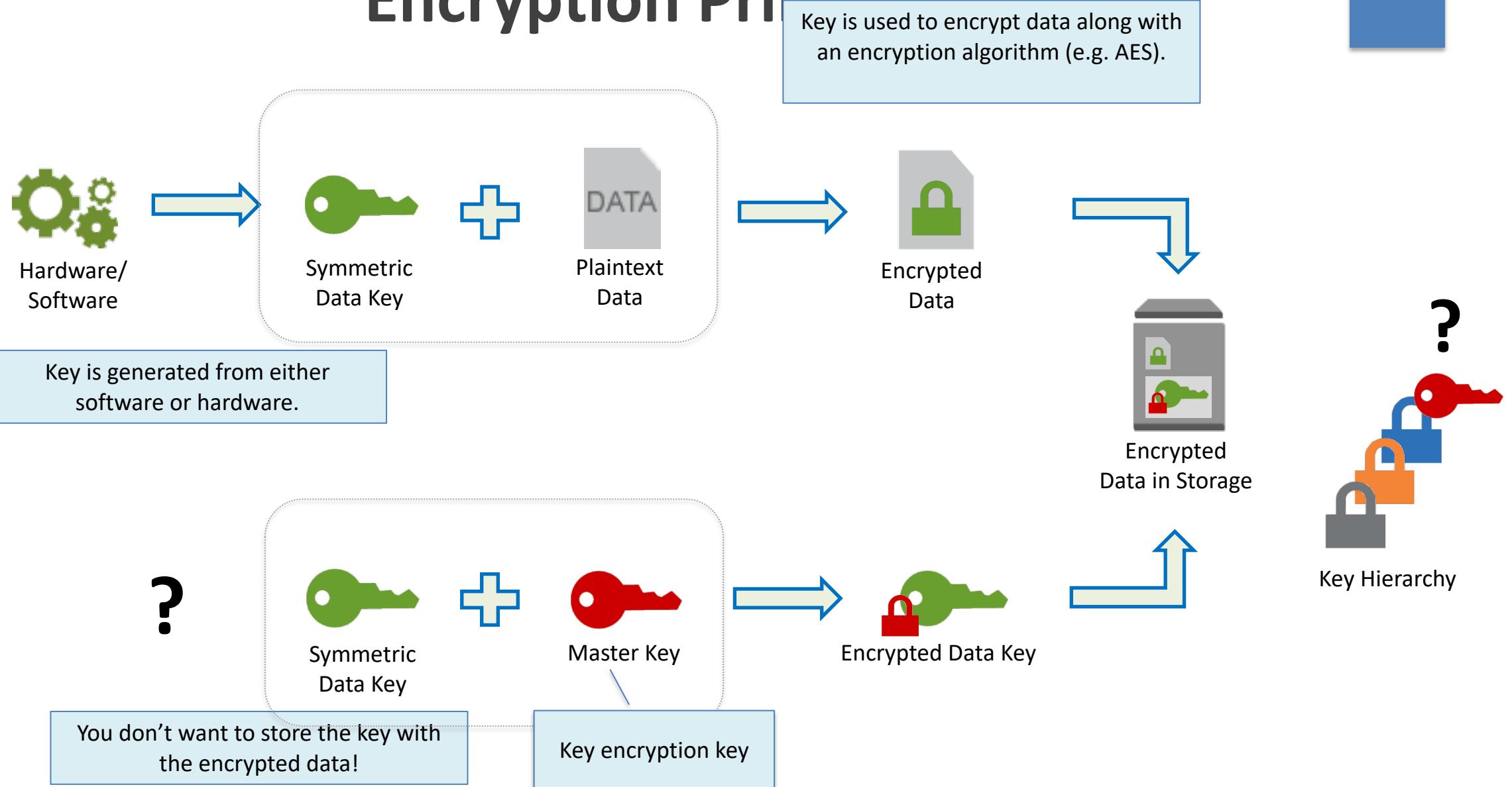
```
{  
    "Version": "2012-10-17",  
    "Statement": [ {  
        "Effect": "Allow",  
        "Action": ["cloudfront:*"],  
        "Resource": "*",  
        "Condition": {  
            "Bool": {  
                "aws:SecureTransport": "true"  
            }  
        }  
    }]  
}
```

Grant permission to access all CloudFront actions for the group this policy is attached to...

...with condition that actions require use of SSL/TLS

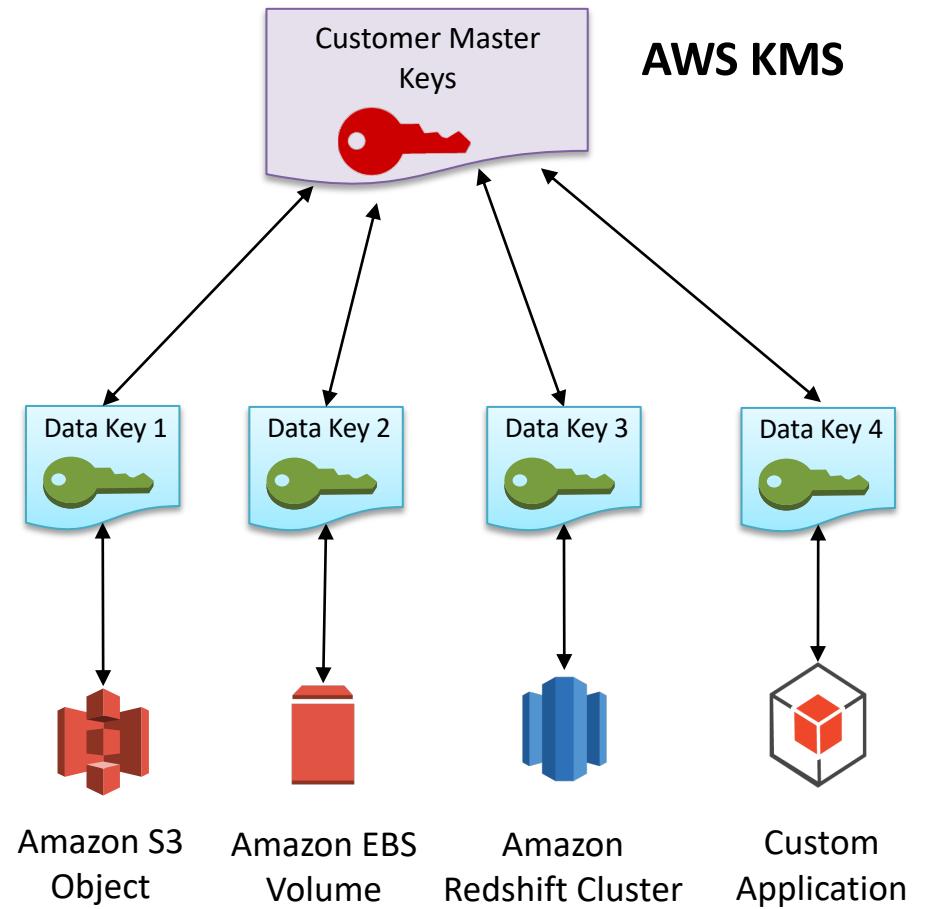
Encrypting Data

Encryption Primer



AWS Key Management Service (KMS)

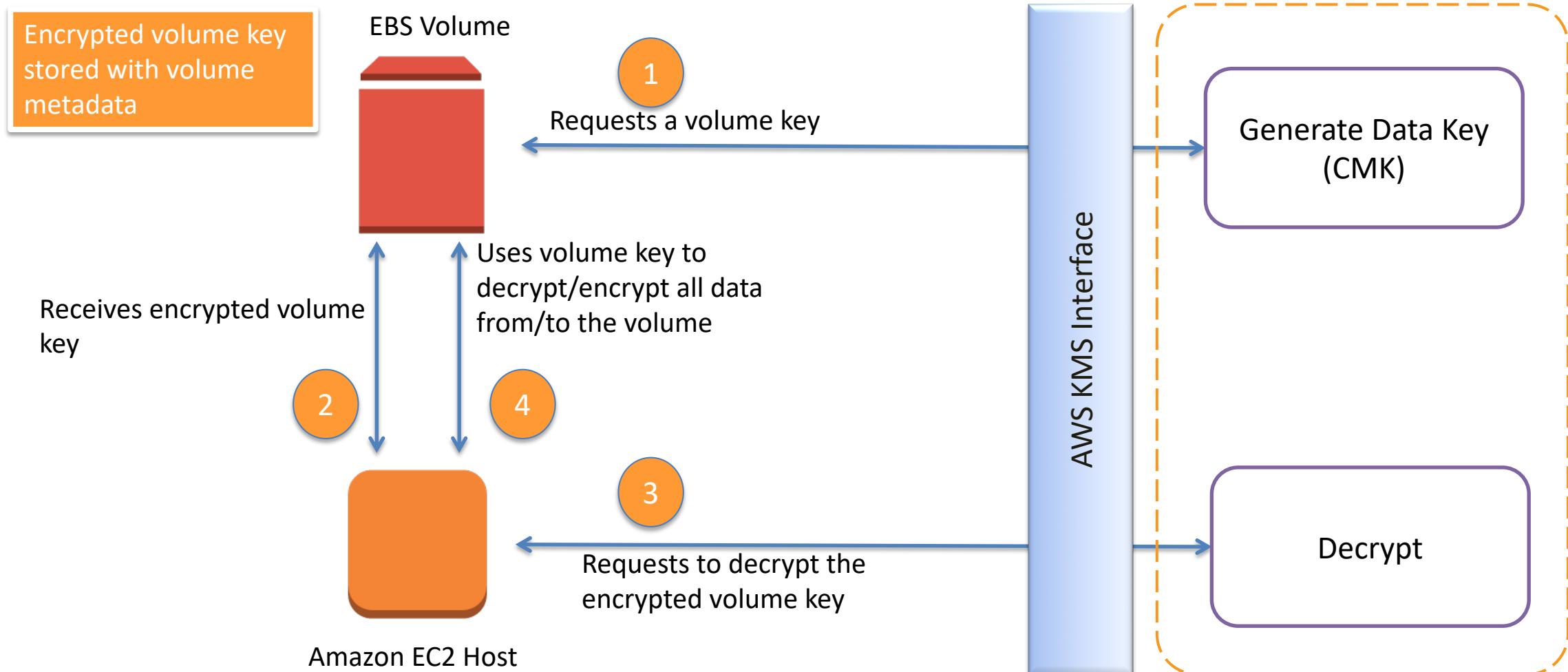
- AWS KMS is a managed encryption service that enables you to easily encrypt your data.
 - Two-tiered key hierarchy using envelope encryption.
 - Data keys are unique.
 - AWS KMS master keys encrypt data keys.
 - AWS KMS master keys never leave the AWS KMS system.



Benefits of Using AWS KMS

- Only data keys are available directly to the customer, and these are unique to each item encrypted.
 - If one was compromised, it would not allow decryption of other objects.
- The risk of a compromised data key is limited.
- The performance for encrypting large data is improved.
- It is easier to manage a small number of master keys than millions of data keys.

EBS Volume Encryption Using AWS KMS



AWS CloudHSM



- Protects your cryptographic keys using a dedicated, tamper-resistant Hardware Security Module (HSM).
- Helps you comply with strict cryptographic key management requirements.
 - Designed to meet [FIPS 140-2](#) and [Common Criteria EAL4+](#) standards.

AWS CloudHSM vs AWS KMS

AWS CloudHSM	AWS KMS
Single-tenant HSM	Multi-tenant AWS service
Customer-managed durability and availability	Highly available and durable key storage and management
Customer managed root of trust	AWS managed root of trust
Broad third-party app support	Broad support for AWS services
Symmetric and asymmetric options	Symmetric encryption only

S3 Security Features

- Data stored in Amazon S3 is private by default, requires AWS credentials for access
 - Access to Amazon S3 can be over HTTP or HTTPS
 - Amazon S3 logging allows auditing of access to all objects
 - Amazon S3 supports access control lists and policies for every bucket, prefix (directory/folder), and object
- Amazon S3 provides server-side encryption (AES-256) using AWS maintained keys or customer provided keys
 - AWS encryption keys are further encrypted with a rotating key
 - Can also encrypt data before storage in Amazon S3 (client-side encryption)

Cost

- Leveraging the Elasticity
- Eliminating Waste
- AWS Pricing Philosophy
- AWS Trusted Advisor

Reliability

- Fault-tolerance
- Scalable
- Disaster Recovery
 - Backup and Restore
 - Pilot Light
 - Fully Working Low-Capacity Standby
 - Multi-Site Active-Active

Performance

- Compute – Right Sizing
- Storage – Right Storage Service
- Caching
- Continues Improvement on KPIs

Operational Efficiency

- Operations as code
- Annotated documentation
- Make frequent small, incremental changes
- Refine operations procedures frequently
- Anticipate failure
- Learn from all operations failure

Thank You