The Orion Papers Pro

AWS Certified Solutions Architect - Professional
Course Manual

Enter

Adrian Cantrill, Training Architect

Linux Academy

AWS Certified Solutions Architect - Professional

Start

Linux Academy

Dear students,

Welcome to Linux Academy's updated AWS Certified Solutions Architect - Professional exam prep course! This course will help you prepare for the AWS Certified Solutions Architect - Professional exam.

To help you on your journey, we've provided an interactive learning tool known as ***The Orion Papers***. This resource is meant to be a non-linear, interactive guide to the course material. You can use it to enhance your learning experience and learn about the required AWS services at your own pace. This interactive guide can be used independently of the video lessons, but we are also going to use it in conjunction with them. Each hands-on lab will also provide a static chart for you to reference.

This course contains a variety of labs that give you hands-on exposure to AWS in live environments. You won't just learn the material; you'll get to practice it until you're confident in your skills! Successfully completing this course and practicing the hands-on labs will put you in a great position for taking the certification exam.

We hope you enjoy the course.

Now let's get certified!

Adrian Cantrill
Craig Arcuri

Global Infrastructure | Account & Services Layer | Physical & Networking Layer

AWS Account
7-Layer OSI Model
HA vs. FT
Regions, AZs, & Edge
JSON 101
DR: RPO/RTO
Data Persistence

Account Root User

Create IAM Entities

IAM

Permissions

The root user is initially the only principal that can authenticate to the account and the only principal authorized to do so.

The root user is *specific* to an account.

AWS

Authenticate

Authorize

Bill

The root user can create IAM principals that can also authenticate to the account and can be authorized to act with granular permissions.

Account Domains

AWS accounts consist of three discrete domains: Authentication, Authorization, and Billing. By default, every AWS account has separate billing, users, and permissions. If an account is exploited, the "blast radius" is limited to that specific account.

Global Infrastructure | Account & Services Layer | Physical & Networking Layer

AWS Account | 7-Layer OSI Model | HA vs. FT | Regions, AZs, & Edge | JSON 101 | DR: RPO/RTO | Data Persistence

JSON (JavaScript Object Notation)

Key/Value Pairs:

```
{ "Name" : "LinuxAcademy",
  "Type" : "Company" }
```

Lists of Things:

```
[ 1, 2, 3]
["Penny", "Roffle", "Winkie", "Truffles"]
```

Key/Value Pairs & Lists:

```
{ "Name" : "Adrian",
"Pets" : ["Penny", "Roffle", "Winkie", "Truffles"] }
```

Nesting and Complex Stuctures:

```
{"Team" : [ {"Name" : "Tom"}, {"Name", "Tia"}, {"Name" :
"Adrian", "Pets" : ["Winkie","Roffle", ... ] } ] }
```


```
{"Team" : [ {"Name" : "Tom"}, {"Name", "Tia"}, {"Name" :
"Adrian", "Pets" : ["Winkie","Roffle", ... ] } ] }
```

Global Infrastructure | Account & Services Layer | Physical & Networking Layer

AWS Account

7-Layer OSI Model

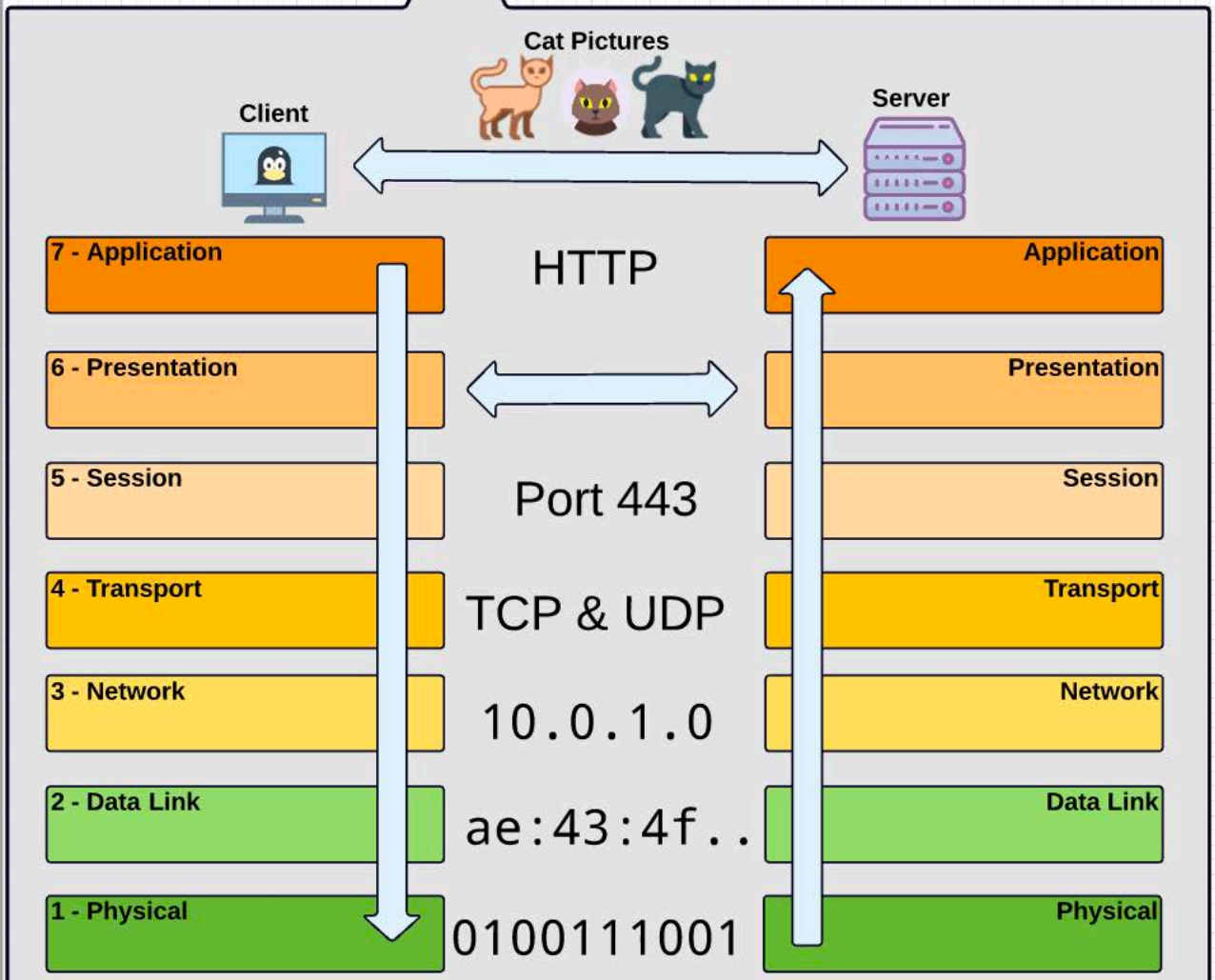
HA vs. FT

Regions, AZs, & Edge

JSON 101

DR: RPO/RTO

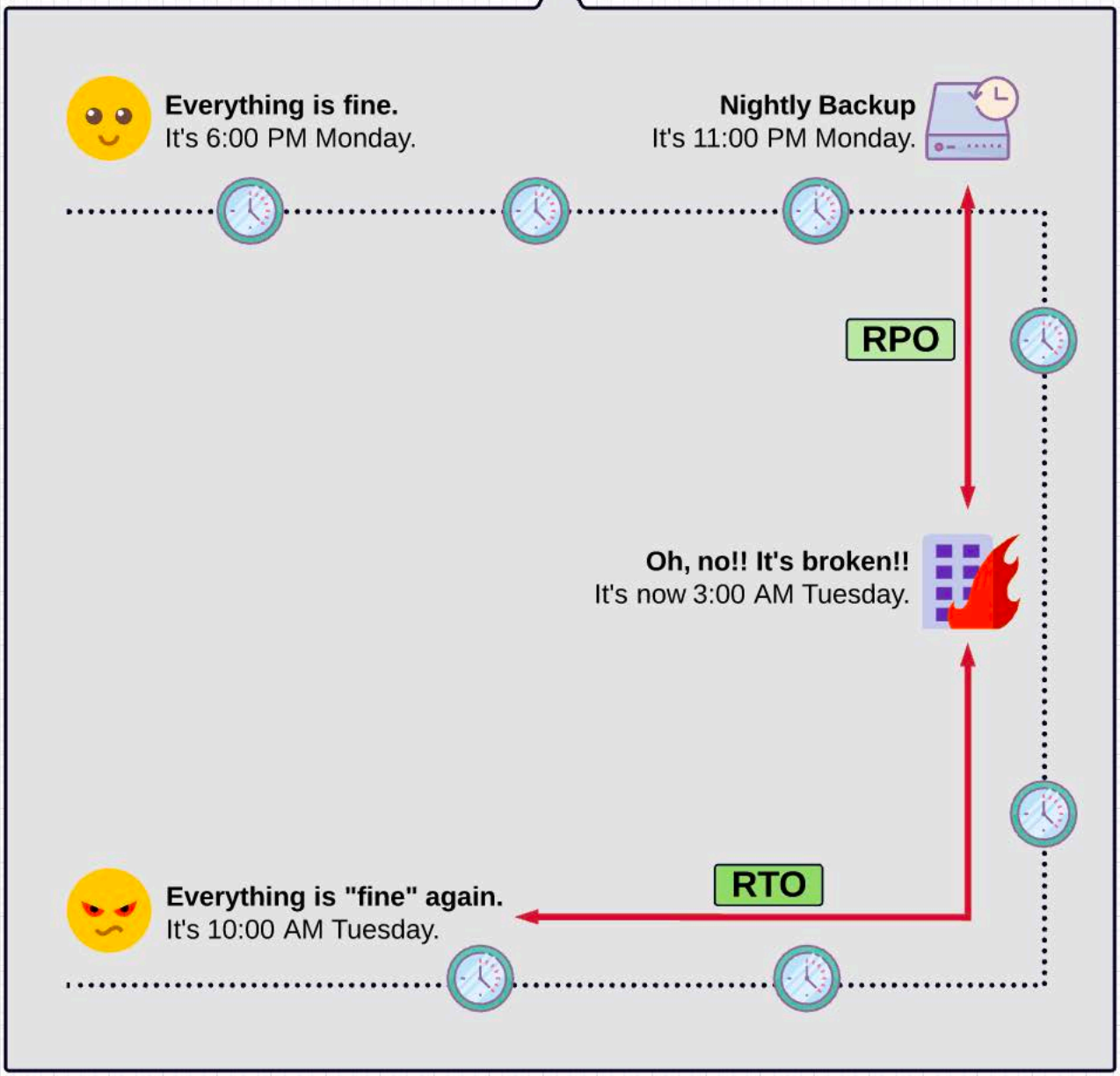
Data Persistence



The OSI 7-layer networking model splits network communication into isolated layers. The protocols at any given layer can be changed or updated without affecting any of the other layers. Communication can occur between hosts as long as both hosts are using the same protocol at the same layer.

Global Infrastructure | Account & Services Layer | Physical & Networking Layer

AWS Account
7-Layer OSI Model
HA vs. FT
Regions, AZs, & Edge
JSON 101
DR: RPO/RTO
Data Persistence



Global Infrastructure | Account & Services Layer | Physical & Networking Layer

AWS Account
7-Layer OSI Model
HA vs. FT
Regions, AZs, & Edge
JSON 101
DR: RPO/RTO
Data Persistence

Everything is fine.
It's 6:00 PM Monday.

Nightly Backup
It's 11:00 PM Monday.

RPO: Recovery Point Objective
The time between when a disaster occurs and the last recoverable copy of key business data was created.

RPO

Oh, no!! It's broken!!
It's now 3:00 AM Tuesday.

Minimize the length of this time period (via regular backups, snapshots, and transaction logs) to avoid business disruptive data loss.

RTO

Everything is "fine" again.
It's 10:00 AM Tuesday.

Global Infrastructure | Account & Services Layer | Physical & Networking Layer

AWS Account

7-Layer OSI Model

HA vs. FT

Regions, AZs, & Edge

JSON 101

DR: RPO/RTO

Data Persistence

Everything is fine.
It's 6:00 PM Monday.

Nightly Backup
It's 11:00 PM Monday.

RPO

RTO: Recovery Time Objective
The time between when a disaster occurs and when the system can be restored to an operational state and handed over to the business for testing.

Oh, no!! It's broken!!
It's now 3:00 AM Tuesday.

Improve this by decreasing restore time, having hot spares tested and ready to use, and enforcing efficient processes.

RTO

Everything is "fine" again.
It's 10:00 AM Tuesday.

Global Infrastructure

Account & Services Layer

Physical & Networking Layer

AWS Account

7-Layer OSI Model

HA vs. FT

Regions, AZs, & Edge

JSON 101

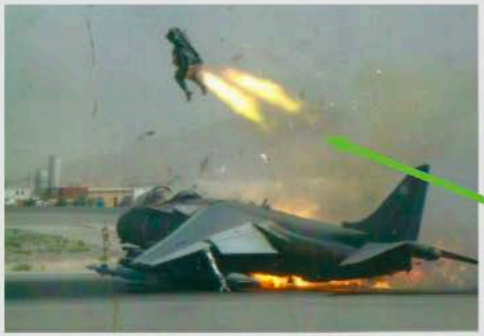
DR: RPO/RTO

Data Persistence

No HA
No FT
No DR

HA
No FT
No DR

HA
FT
No DR



HA
FT
DR

Global Infrastructure | Account & Services Layer | Physical & Networking Layer

AWS Account

7-Layer OSI Model

HA vs. FT

Regions, AZs, & Edge

JSON 101

DR: RPO/RTO

Data Persistence

Ephemeral

Data that is generally local to a resource and is lost when that resource is powered down

Instance Store Volume

Amazon ElastiCache

Transient

Data that exists in a form while it's passed between sources

Queue

Persistent

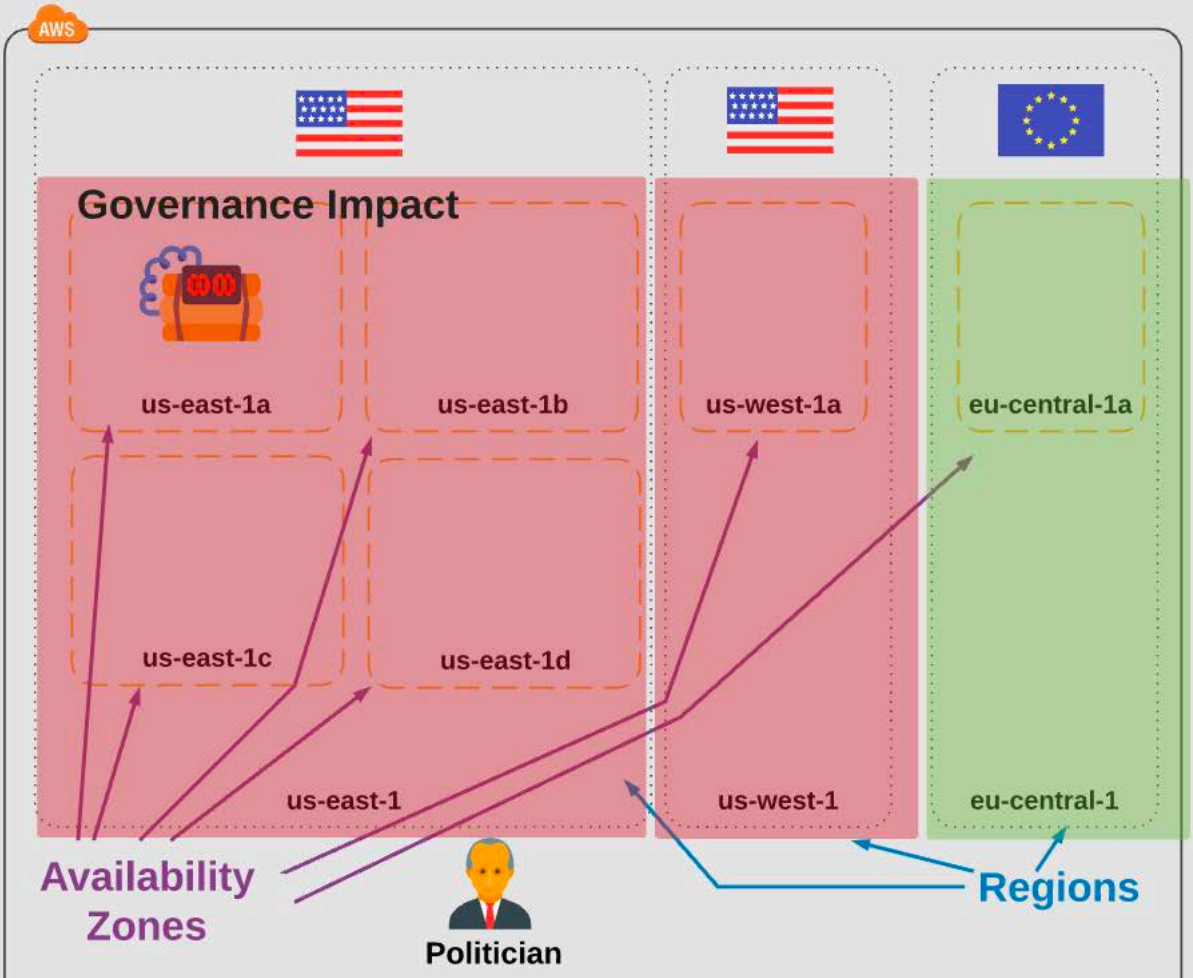
Data that is durable and survives power events (Start, Stop, Restart)

Amazon EBS

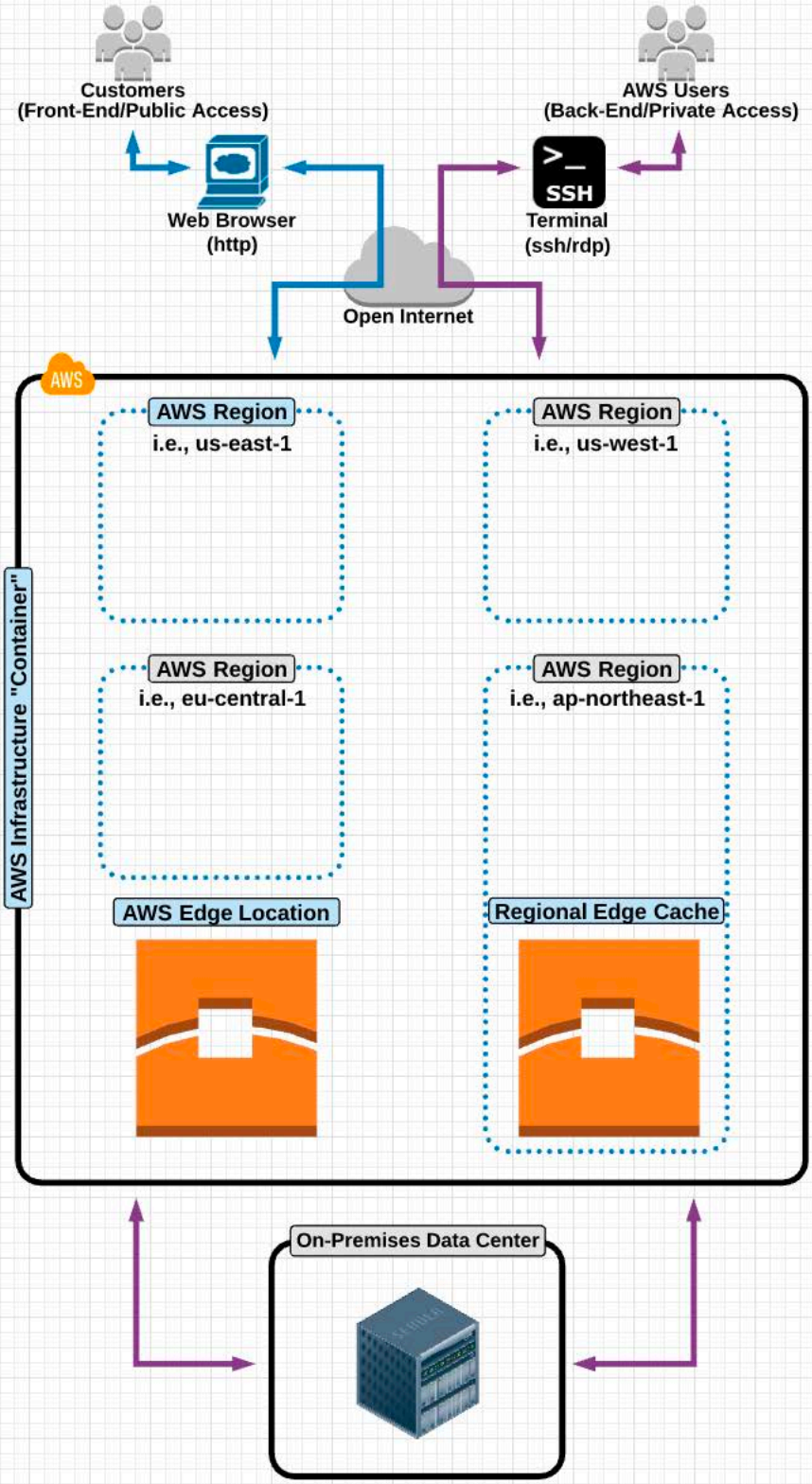
Amazon EFS

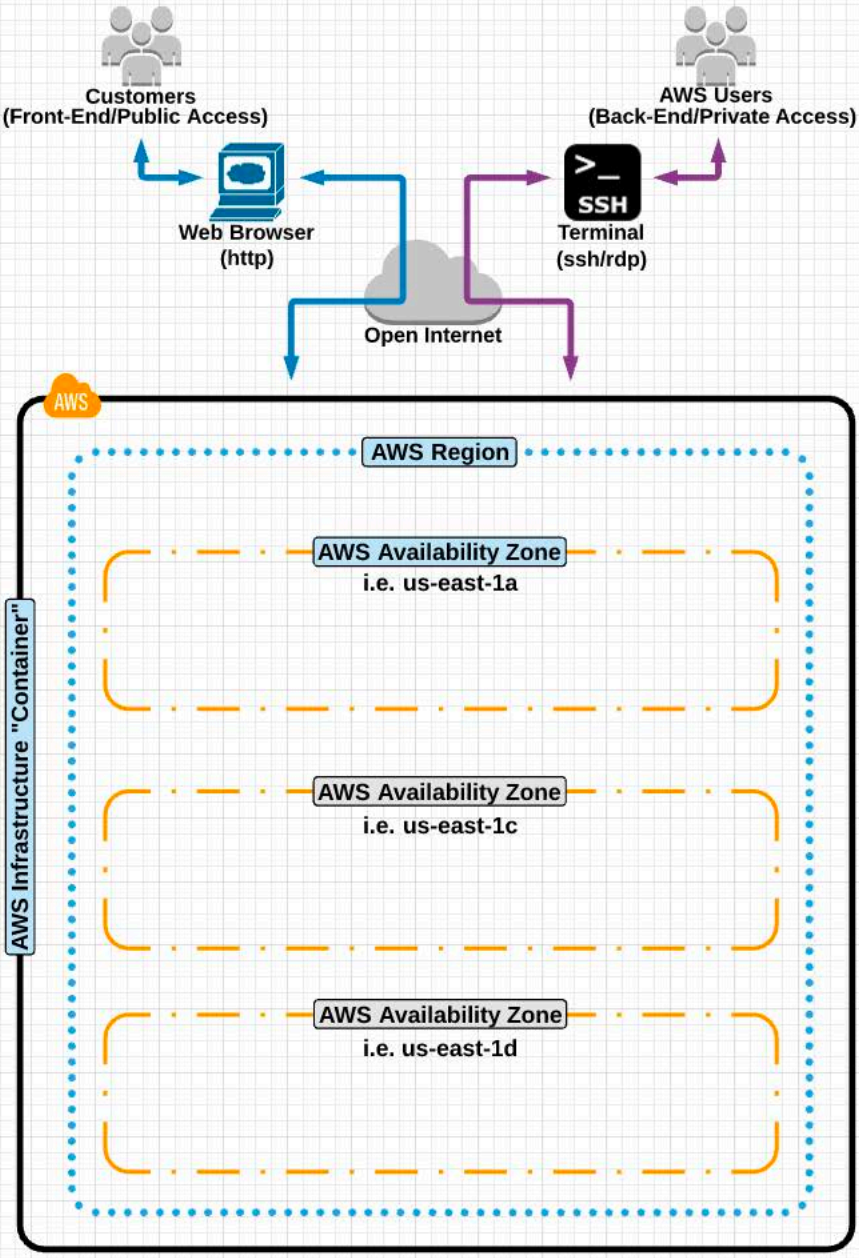
Global Infrastructure | Account & Services Layer | Physical & Networking Layer

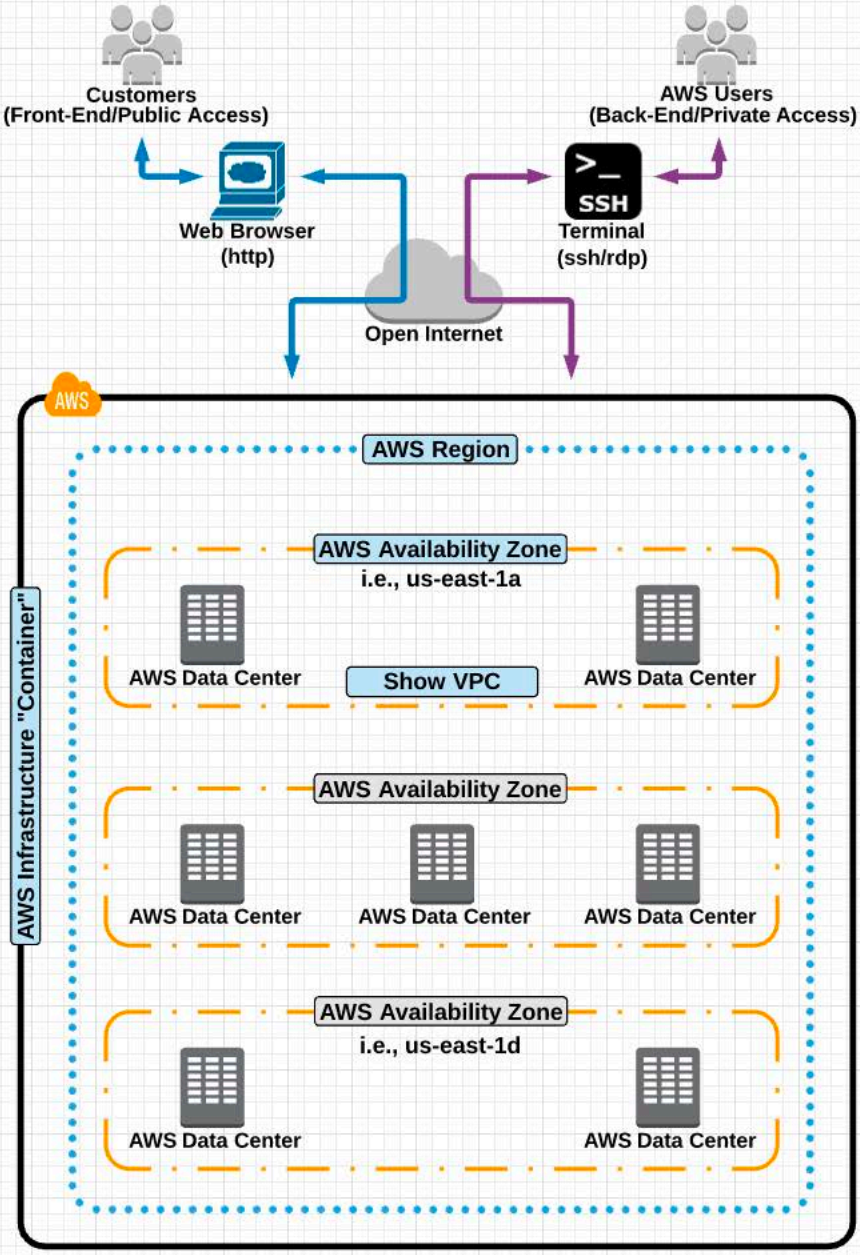
AWS Account
7-Layer OSI Model
HA vs. FT
Regions, AZs, & Edge
JSON 101
DR: RPO/RTO
Data Persistence



Edge locations are smaller pockets of infrastructure, distributed globally in major cities. These locations are used to distribute content and operate other "edge" services.

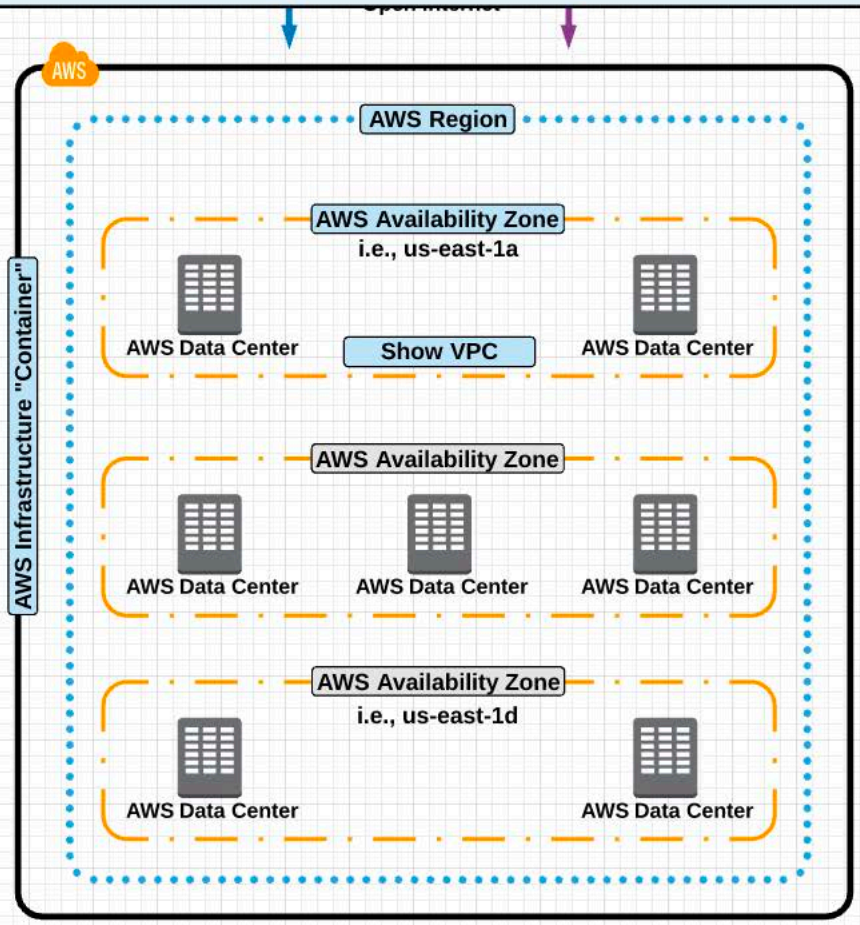

Customers
(Front-End/Public Access)
AWS Users
(Back-End/Private Access)
Web Browser
(http)
SSH
Terminal
(ssh/rdp)
Open Internet
AWS
AWS Infrastructure "Container"
AWS Region
i.e., us-east-1
AWS Region
i.e., us-west-1
AWS Region
i.e., eu-central-1
AWS Region
i.e., ap-northeast-1
AWS Edge Location
Regional Edge Cache
On-Premises Data Center


Customers
(Front-End/Public Access)
AWS Users
(Back-End/Private Access)
Web Browser
(http)
Terminal
(ssh/rdp)
Open Internet
AWS
AWS Region
AWS Availability Zone
i.e. us-east-1a
AWS Availability Zone
i.e. us-east-1c
AWS Availability Zone
i.e. us-east-1d
AWS Infrastructure "Container"

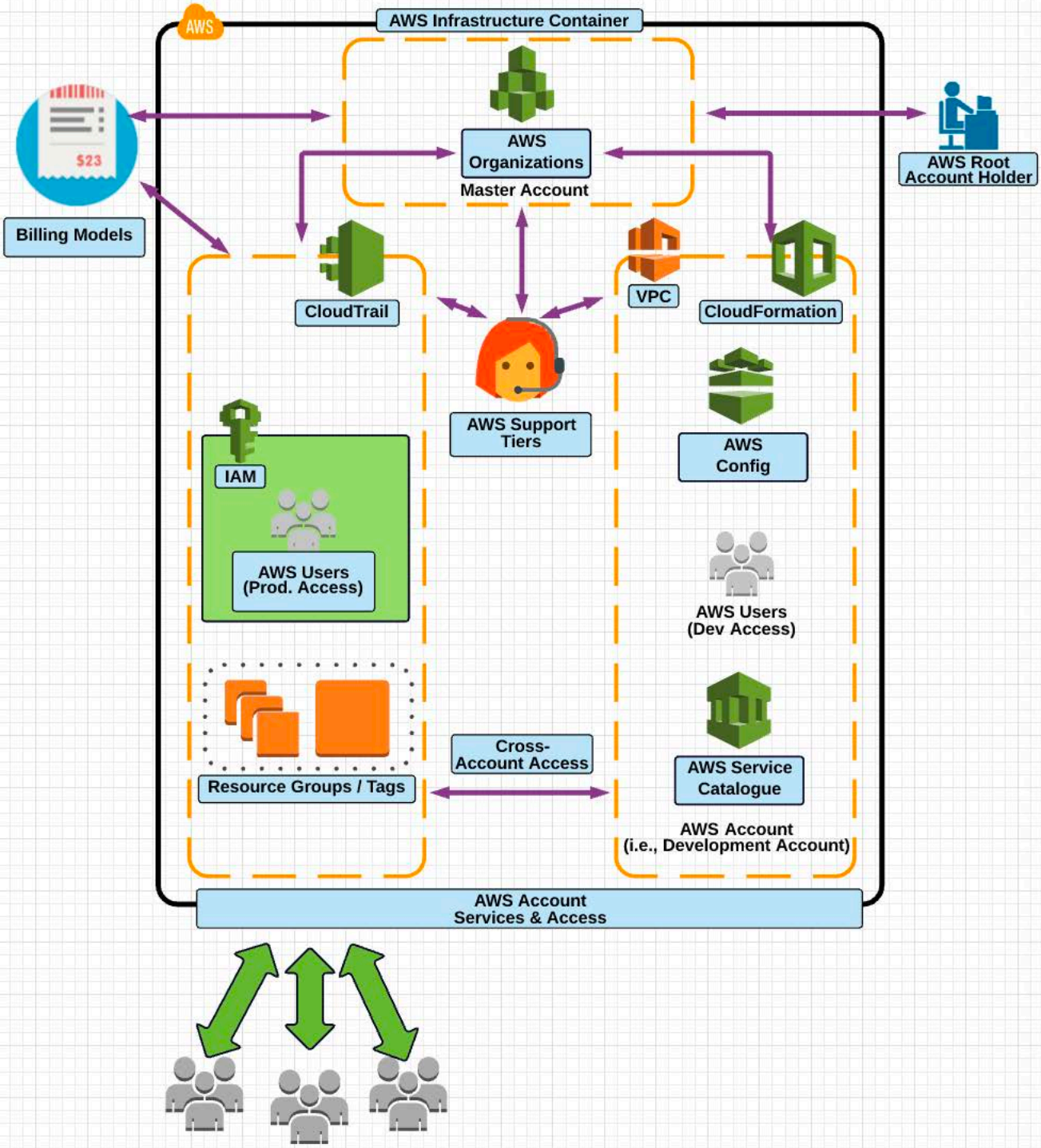

Customers
(Front-End/Public Access)
AWS Users
(Back-End/Private Access)
Web Browser
(http)
SSH
Terminal
(ssh/rdp)
Open Internet
AWS
AWS Infrastructure "Container"
AWS Region
AWS Availability Zone
i.e., us-east-1a
AWS Data Center
Show VPC
AWS Data Center
AWS Availability Zone
AWS Data Center
AWS Data Center
AWS Data Center
AWS Availability Zone
i.e., us-east-1d
AWS Data Center
AWS Data Center

AWS Infrastructure Container

- This represents the boundaries of AWS.
- Everything inside is part of AWS's infrastructure, including all of its physical networking components and services.
- Everything outside represents items that are external to AWS, which either connect to AWS or belong to you or your company (e.g., on-premises servers, the open internet, or your personal computer).

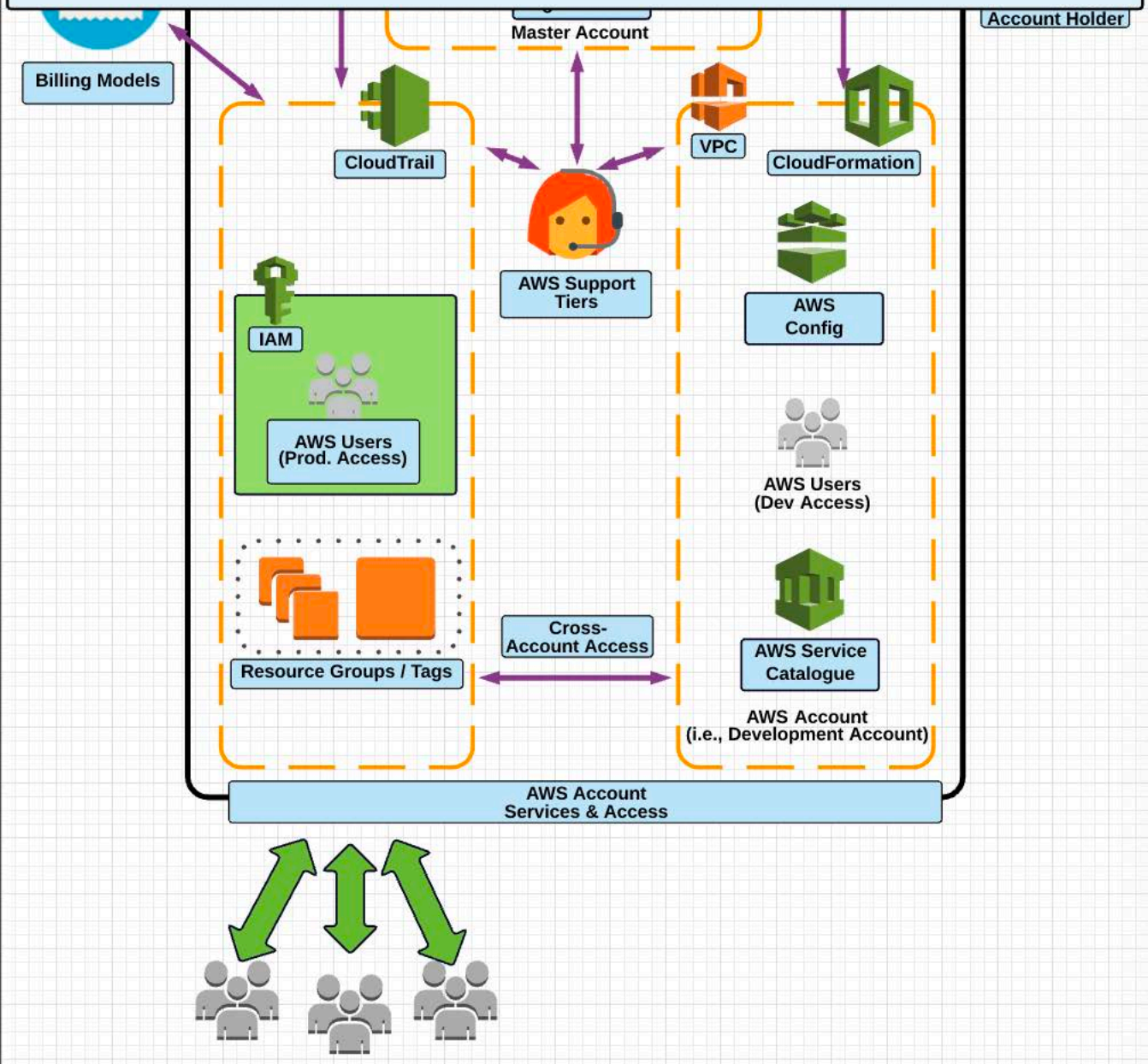


The Account and Services layer represents how we create, access, and manage an AWS account and its services, from interacting with an AWS account and managing user rights, to accessing and using various AWS services and features.



AWS Infrastructure Container

- The AWS infrastructure container represents the boundaries of AWS.
- Everything inside is part of AWS's infrastructure, including all of its physical networking components and services.
- Everything outside represents items that are external to AWS, that either connect to AWS or belong to you or your company (i.e., on-premises servers, the internet, or your personal computer).



Billing Models

Close

AWS provides three billing models:

On-Demand

On-Demand is the default AWS billing model. You pay only for what you consume: per hour, per GB transferred, or per GB stored/month. On-Demand is ideal for ad-hoc usage, where you can't calculate future usage patterns. It provides small usage discounts for high volume but is generally the most expensive AWS billing model. It has standard startup priority.

Reserved

The Reserved billing model provides significant reductions in cost and (optional) capacity reservations if you commit to a 12- or 36-month term. You can choose from three payment options: All Upfront, Partial Upfront, and No Upfront. Full Upfront provides the best cost advantages.

Reservations can be used for EC2 instances, RDS instances, DynamoDB performance, and many other services in AWS.

Reserved terms are useful when your usage is **known** and **steady-state**.
The Reserved billing model can also reserve capacity, providing high-priority startup.

Spot

Spot pricing is ideal for sporadic workloads where you can tolerate interruptions and want the lowest price.

Spot pricing can be significantly cheaper than On-Demand but is 100% dependent on spare capacity. As a result, prices can be higher if capacity is constrained.

The resources you choose to allocate using a Spot pricing plan can be terminated with *very* little notice, so any workloads need to be tolerant of interruption.

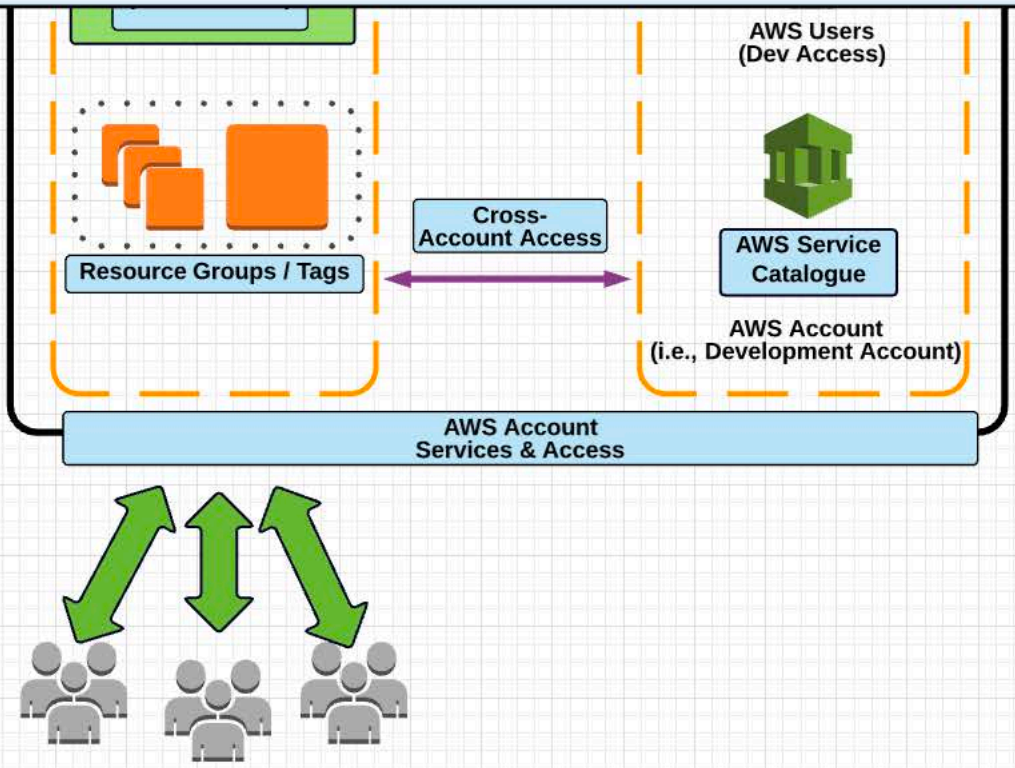
Spot resources have the **lowest** startup priority.

The Root User

- The user that is created when you first create your AWS account is called the **root** user.
- The root user's credentials are the email address and password you used when signing up for your AWS account.
- The root user has *full* administrative rights and access to every part of the account. Note: Privileges cannot be revoked from the root user**.**
- Root users for accounts created within an AWS Organization are created with no password. Password recovery should be used if the password is needed.
- Root users of non-master accounts in an organization can be restricted using Service Control Policies (see the dedicated SCP lesson).

Root User Best Practices

- You should not use the root user for daily work and AWS administration. Because the account cannot be restricted by default, the risk of credential leakage is too high for day-to-day use.
- You should create an IAM admin user that has administrative rights to replace the root account.
- This IAM account should create additional named or service accounts for day-to-day operations.
- You should *always* protect your root account with MFA.
- Store the root account login details and the MFA device somewhere safe and secure (a fire safe or deposit box).



AWS Organizations

Close

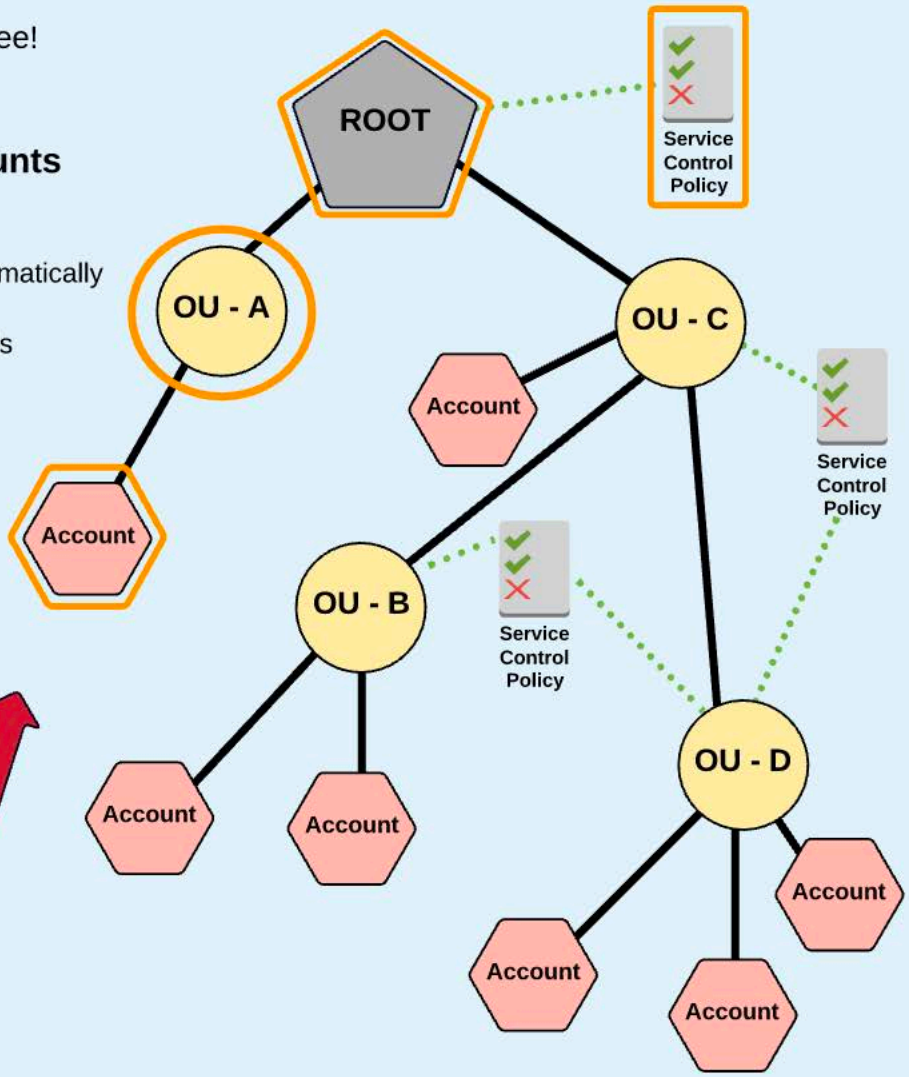
AWS Organizations is a service that lets you consolidate multiple AWS accounts into a single organization. This allows you to manage all accounts within the organization in one place.

AWS Organizations makes billing and permissions easier and allows for the creation of managed accounts. Accounts are organized hierarchically, which provides better security and compliance controls.

Best of all, this service is free!

Manage Multiple Accounts

- IAM policy management
  - Account groups
  - Create accounts programmatically
- Consolidated billing
  - Manage payment methods



Organizations operate in either "Consolidated Billing" or "All Features" mode. This can be modified at any time.

Consolidated Billing

All Features

AWS Organizations

Close

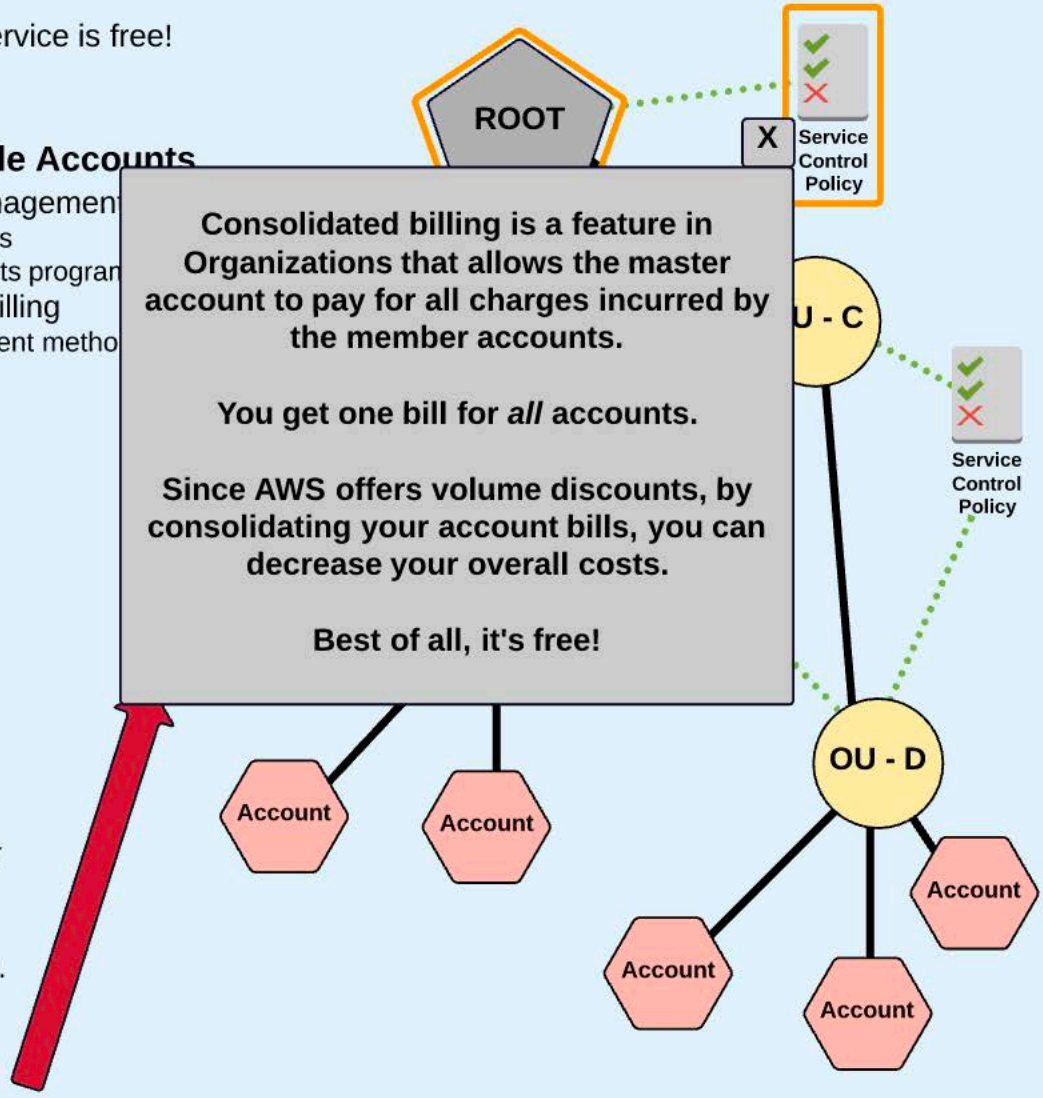
AWS Organizations is a service that lets you consolidate multiple AWS accounts into a single organization. This allows you to manage all accounts within the organization in one place.

AWS Organizations makes billing and permissions easier and allows for the creation of managed accounts. Accounts are organized hierarchically, which provides better security and compliance controls.

Best of all, this service is free!

Manage Multiple Accounts

- IAM policy managemen
  - Account groups
  - Create accounts progran
- Consolidated billing
  - Manage payment metho



Consolidated billing is a feature in Organizations that allows the master account to pay for all charges incurred by the member accounts.

You get one bill for *all* accounts.

Since AWS offers volume discounts, by consolidating your account bills, you can decrease your overall costs.

Best of all, it's free!

Organizations operate in either "Consolidated Billing" or "All Features" mode. This can be modified at any time.

Consolidated Billing

All Features

AWS Organizations

Close

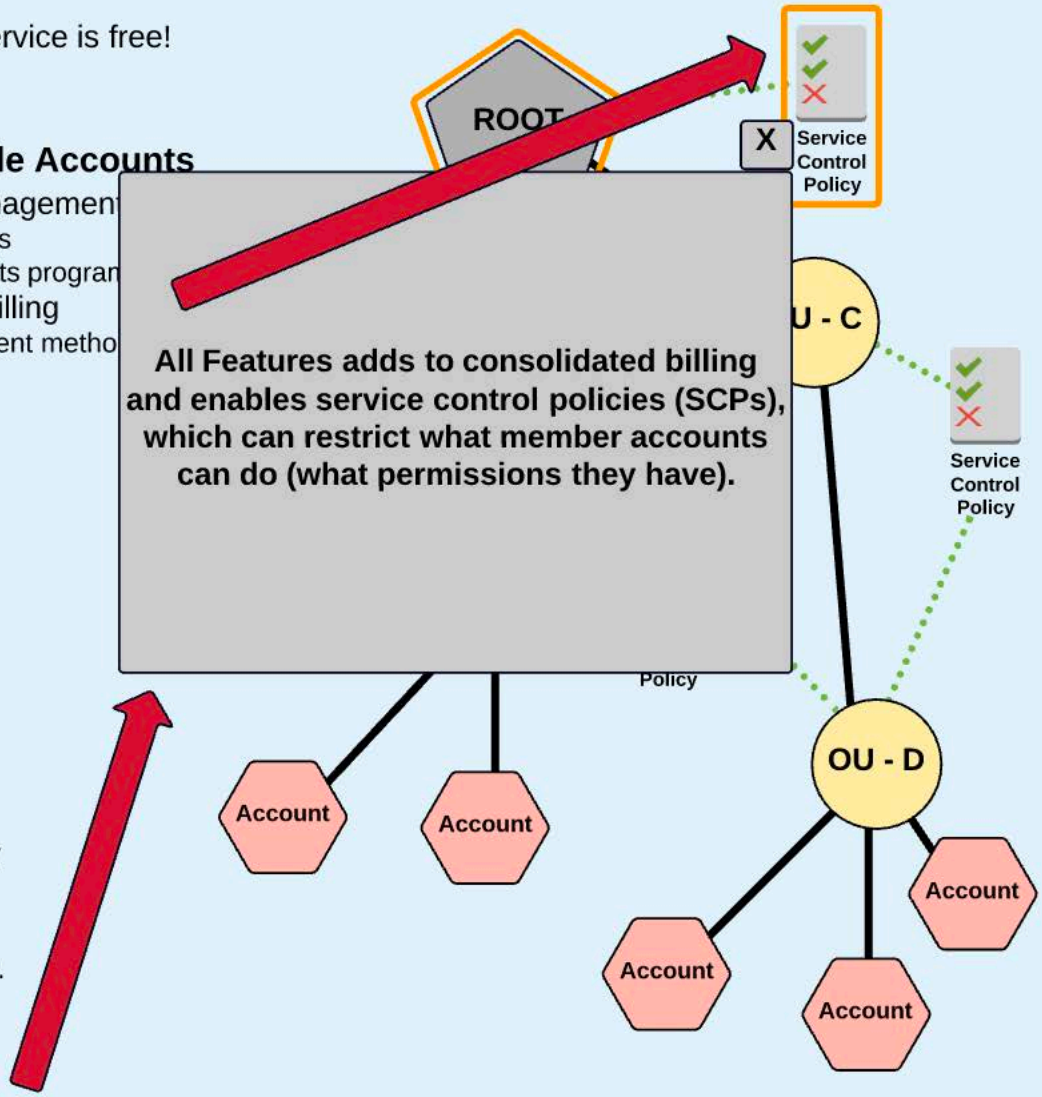
AWS Organizations is a service that lets you consolidate multiple AWS accounts into a single organization. This allows you to manage all accounts within the organization in one place.

AWS Organizations makes billing and permissions easier and allows for the creation of managed accounts. Accounts are organized hierarchically, which provides better security and compliance controls.

Best of all, this service is free!

Manage Multiple Accounts

- IAM policy managemen
  - Account groups
  - Create accounts progran
- Consolidated billing
  - Manage payment metho



All Features adds to consolidated billing and enables service control policies (SCPs), which can restrict what member accounts can do (what permissions they have).

Organizations operate in either "Consolidated Billing" or "All Features" mode. This can be modified at any time.

Consolidated Billing

All Features

CloudTrail

CloudTrail is the primary service we use for logging in AWS.

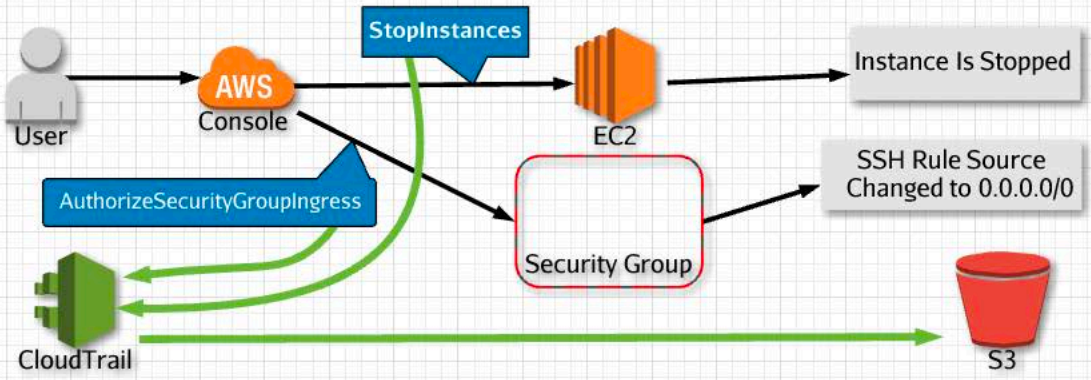
Important Features:

- It logs all the API calls in an AWS account (includes Console, CLI, API/SDK calls)
- It's enabled when your account is created
- Entries can be viewed using the **Event History** (past 90 days)
- **Trail:** A configuration allowing for logs to be sent to an S3 bucket:
  - Single region or multi-region trails can be configured
  - Trails can make multi-account logging possible (more on this later)

Trails have several configuration options:

- **Management events**: Enabling will log control plane events, such as:
  - User login events
  - Configuring security
  - Setting up logging
- **Data events**, which include:
  - Object-level events in S3
  - Function-level events in Lambda
- **Encryption** flexibility:
  - Encrypted in S3 server-side by default but can be changed to KMS
- The logs can be sent to an **S3 bucket** of choice and even prefixed (folders)

Scenario: An IAM user logs in to the AWS Management Console. That user then proceeds to stop an EC2 instance and edit a security group. CloudTrail will log all of these actions. Here is the workflow:



AWS Support Tiers

Close

AWS Support provides three different paid support tiers: Developer, Business, and Enterprise. Each of these tiers has a different cost structure and provides a different feature set, which for a Solutions Architect, is important to understand at a high level.

	Developer	Business	Enterprise
	Recommended if you are experimenting or testing in AWS.	*Recommended if you have production workloads in AWS.*	*Recommended if you have business and/or mission critical workloads in AWS.*
AWS Trusted Advisor Best Practice Checks	7 Core checks	Full set of checks	Full set of checks
Enhanced Technical Support	Business hour email access to Support Engineers Unlimited cases / 1 primary contact	24x7 phone, email, and chat access to Support Engineers Unlimited cases / unlimited contacts (IAM supported)	24x7 phone, email, and chat access to Support Engineers Unlimited cases / unlimited contacts (IAM supported)
Architectural Guidance	General	Contextual to your use-cases	Consultative review and guidance based on your applications
Programmatic Case Management		AWS Support API	AWS Support API
Third-Party Software Support		Interoperability & configuration guidance and troubleshooting	Interoperability & configuration guidance and troubleshooting
Proactive Programs		Access to Infrastructure Event Management for additional fee.	Infrastructure Event Management Well-Architected Reviews Operations Reviews Technical Account Manager (TAM) coordinates access to programs and other AWS experts as needed.
Technical Account Management			Designated Technical Account Manager (TAM) to proactively monitor your environment and assist with optimization.
Training			Access to online self-paced labs
Account Assistance			Concierge Support Team
Pricing	Starts at $29 / month. See pricing detail and example.	Starts at $100 / month. See pricing detail and example.	Starts at $15K / month. See pricing detail and example.

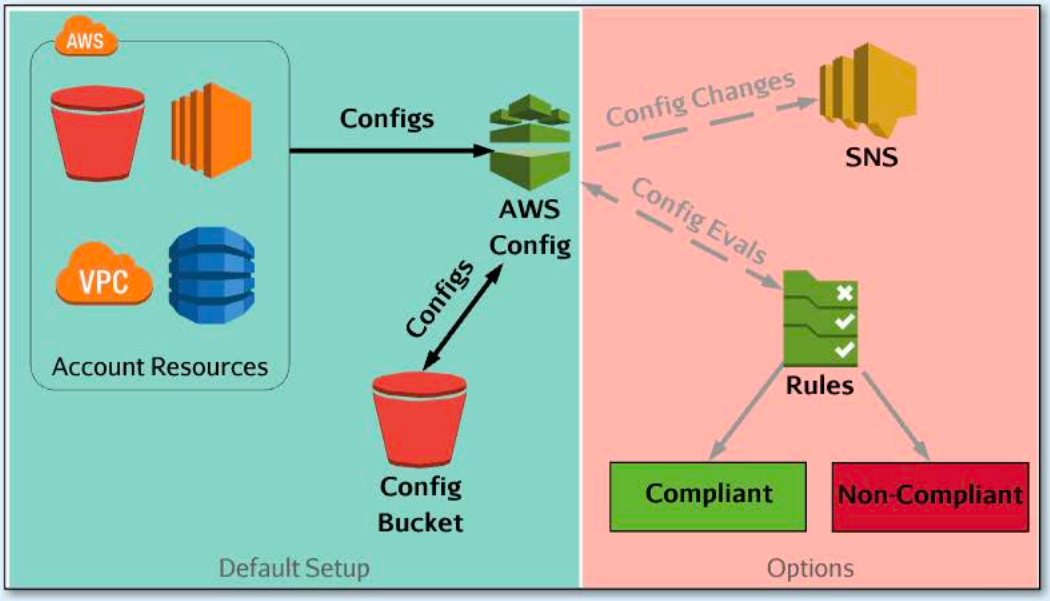
AWS Config

Close

A detailed view of the configuration of AWS resources (EC2, EBS, security groups, VPC, etc.)

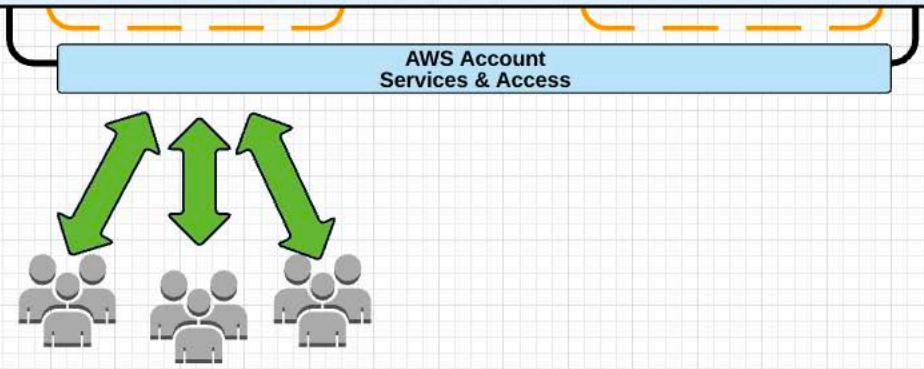
With AWS Config, you can:

- Evaluate resource configurations for desired settings
- Get a snapshot of the current configurations associated with your account
- Retrieve the current or historical configurations of resources in your account
- Receive notification of creations, deletions, and modifications
- View the relationships between resources (e.g., members of a security group)



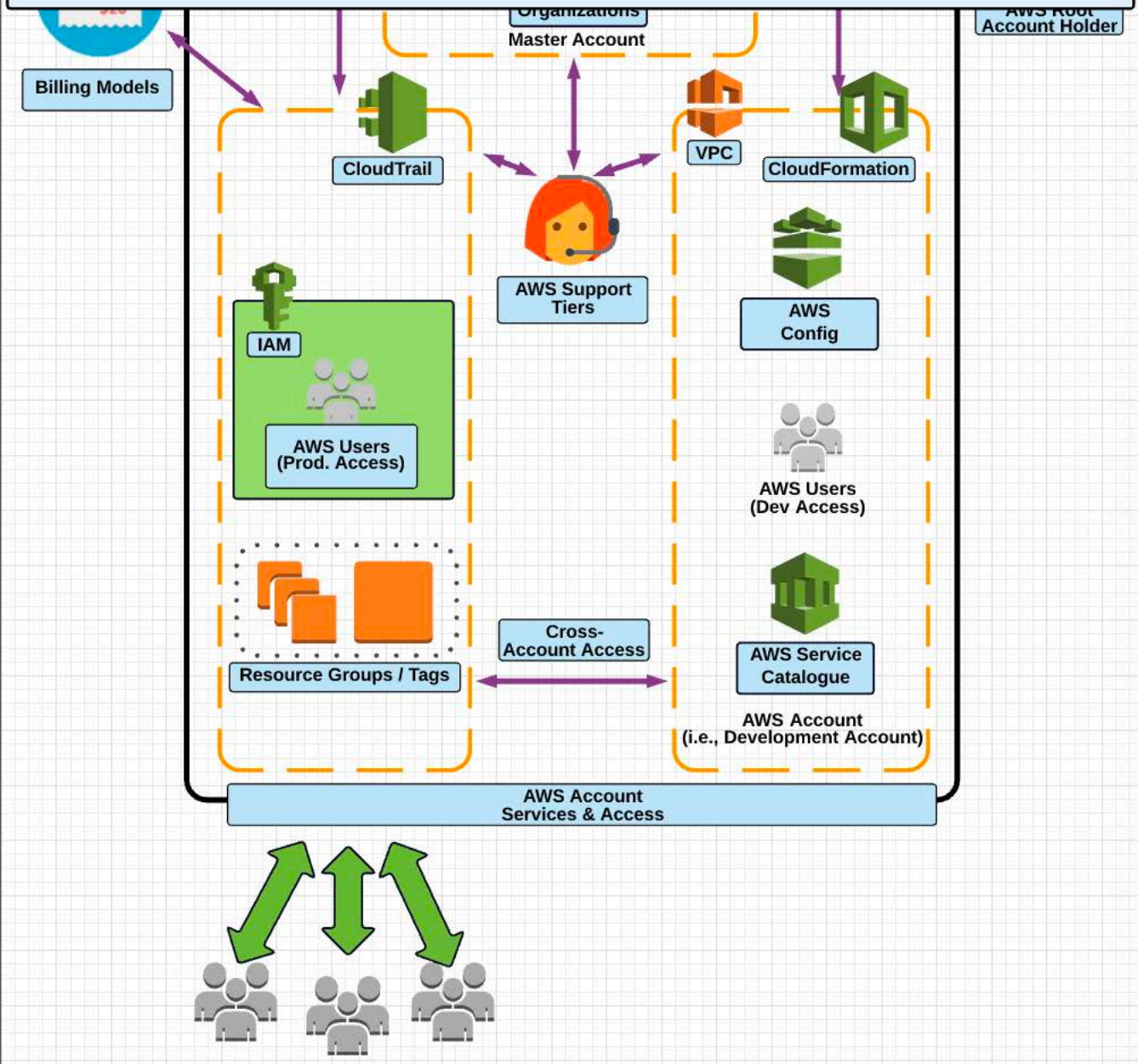
Uses of AWS Config:

- Can be used to identify usage/changes in config over time (current and historical)
- Helps with architecture planning: shows relationships between products and services
- Can generate events related to changes
- Can help keep an implementation in a compliant state
- Assists security teams by showing changes to SGs and NACLs over time
- Fault finding and investigation: last known good config



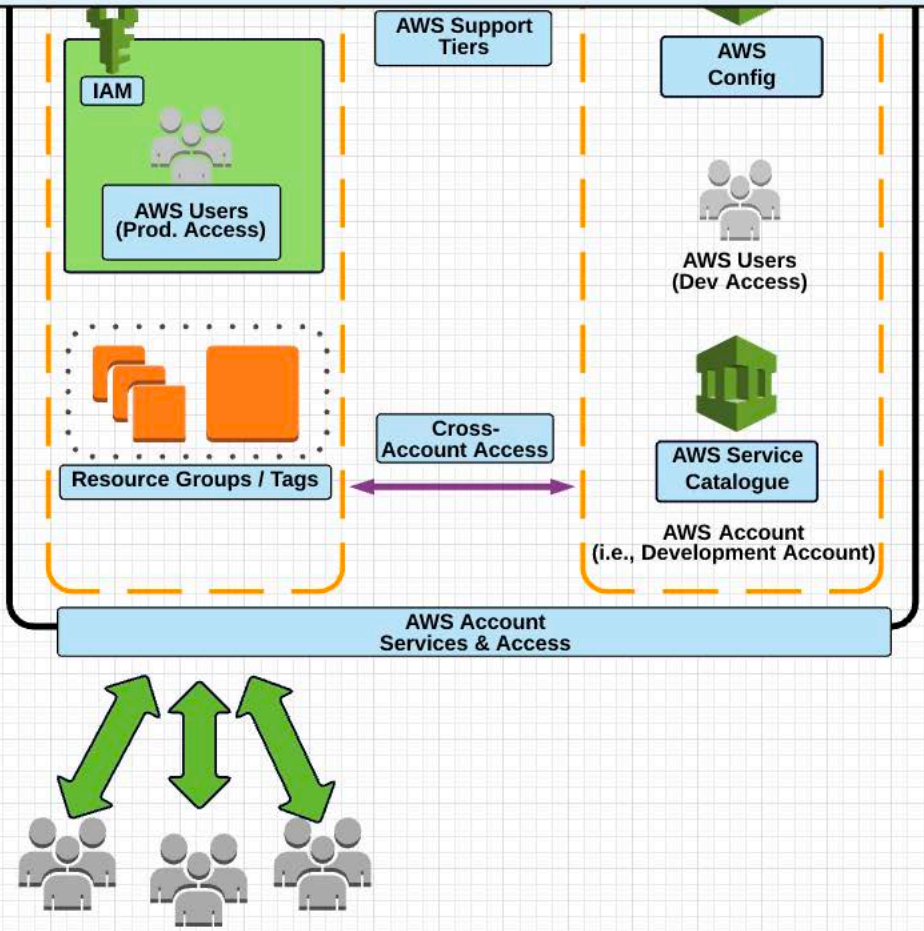
AWS Users

- AWS users are users that you can create (in IAM) who have varying degrees of access (see the example below).
- Permissions are controlled with IAM policies.



Cross-Account Access

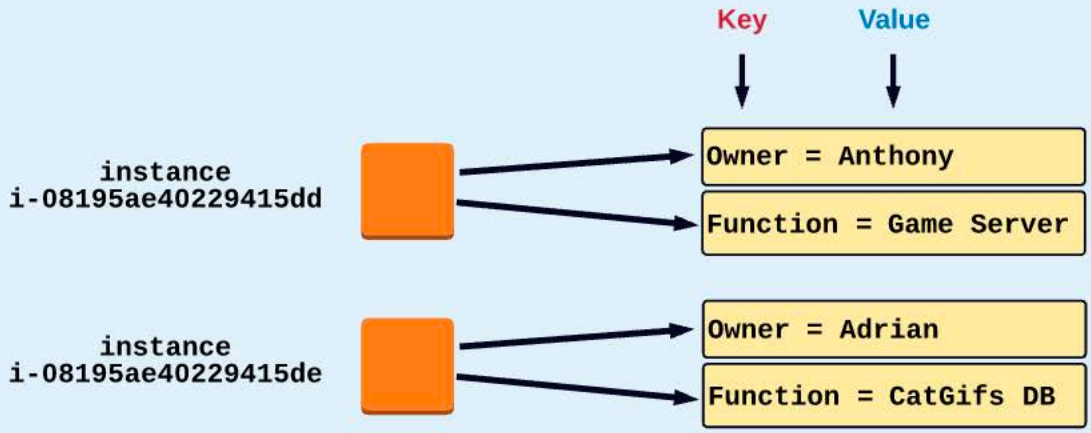
- **Cross-account access** allows IAM users in one account to access resources that are in different AWS accounts that you own (i.e., separate Production and Development accounts; see diagram below).
- Cross-account access is handled through the use of **IAM roles**.
- Users in one account assume a role that grants access to resources in another.
- The benefits of using roles for cross-account access include:
  - No need to create individual IAM users in each account
  - Users don't have to sign out of one account and into another in order to access resources that are in different AWS accounts



Tags and Resource Groups

Close

In AWS, tags are key and value pairs that can be assigned to resources. A tag must contain a key and can optionally have an associated value.



Tags can be used in the UI console under Resource Groups. Both the UI and CLI/API allow you to search and filter by tags. You can also use tags for cost allocation reporting — this has to be enabled globally for your account/organization and then each tag enabled for billing. Tags can also be referenced within IAM policy documents and used to grant/restrict access to resources for IAM users, groups, and roles.

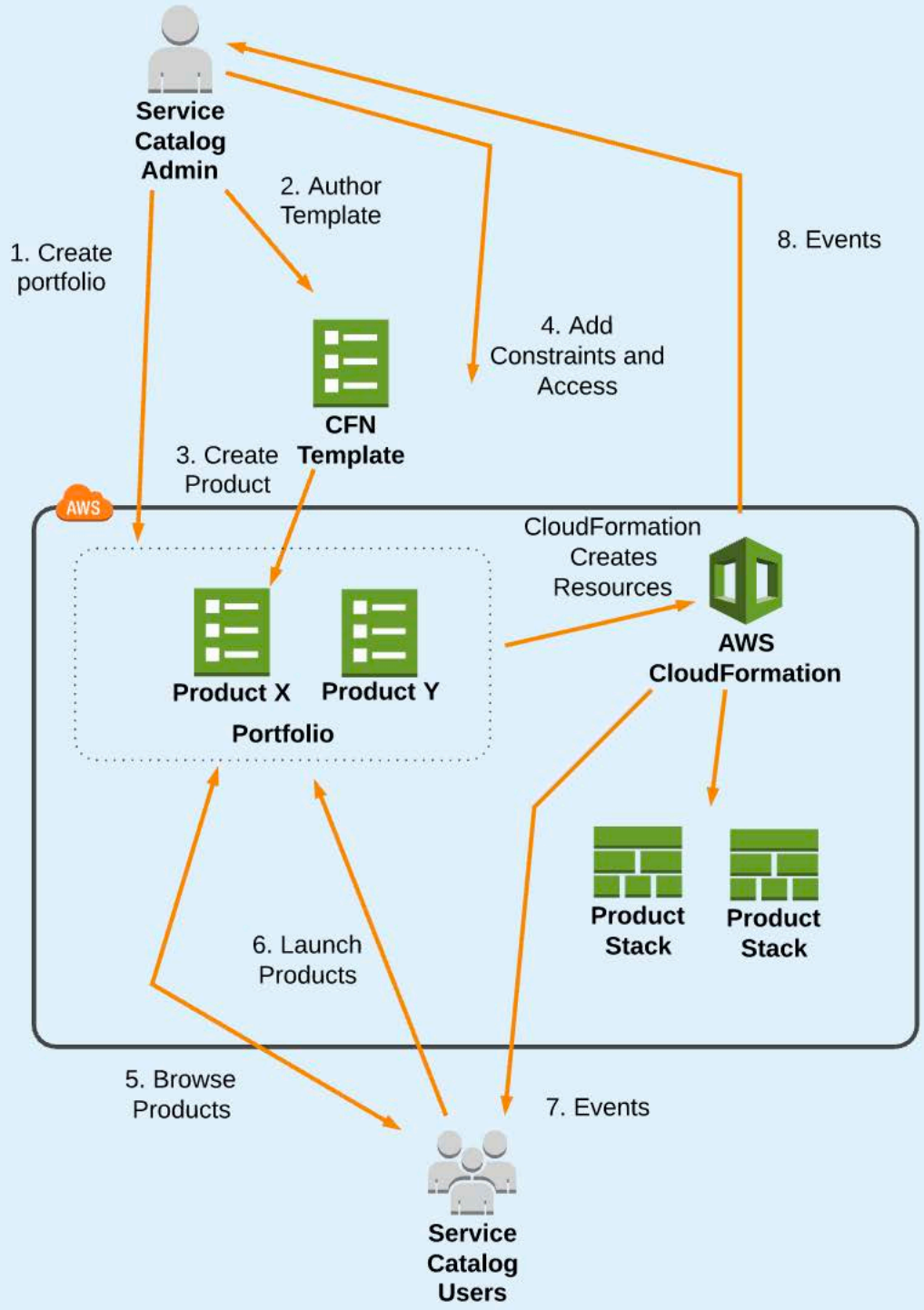
Think carefully about your tagging strategy and categories. All tags should support the business in terms of billing, security, and automation.

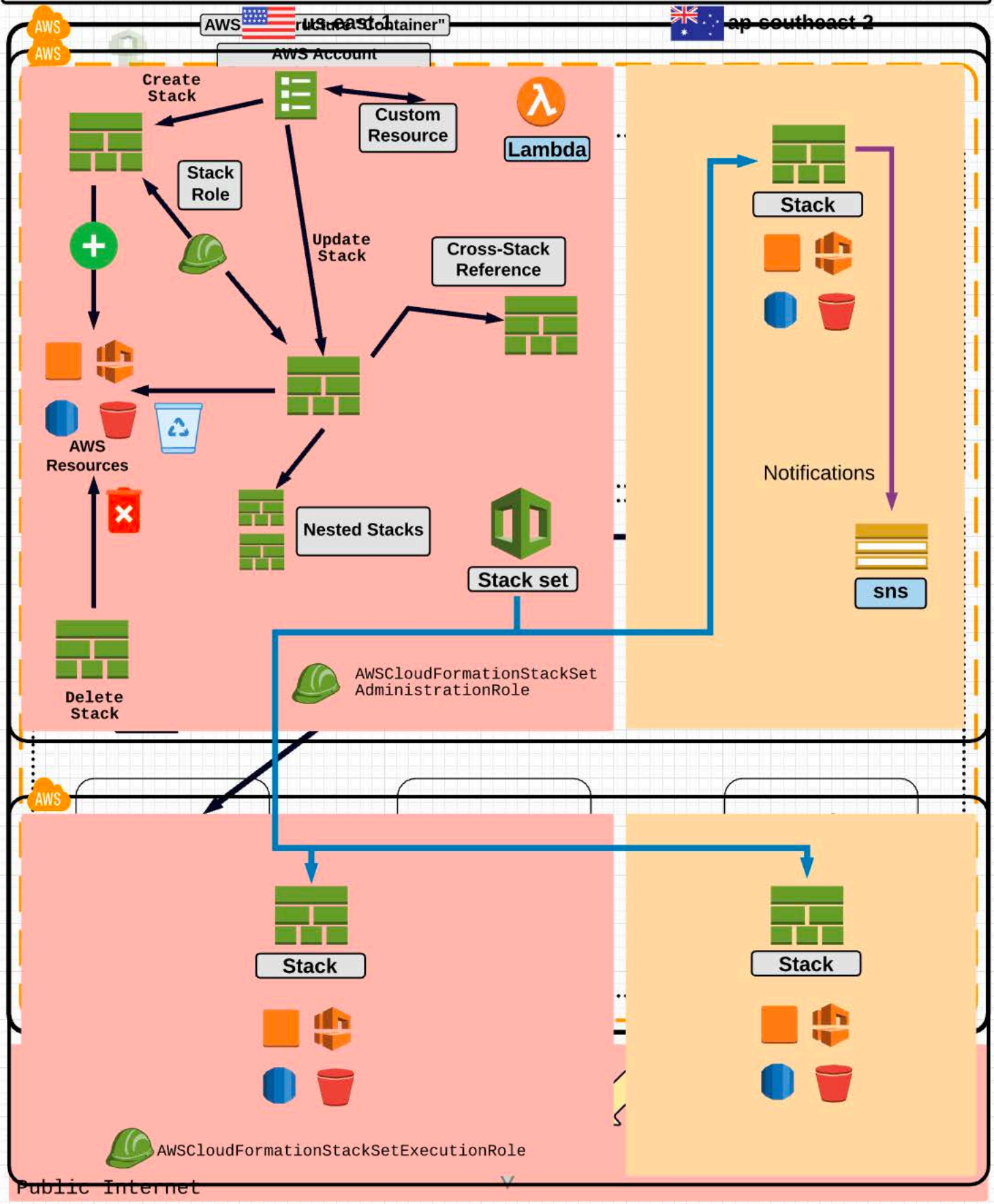
Technical	Automation	Business	Security
Name: Identification	**Backup Type**	**Owner**	**Confidentiality**
Application ID: System components	**Date/Time:** Schedules for a resource	**Department**	**Compliance**
Application Role: The function of the application	**Exclude**: Exclude from automation	**Cost Center**	**Access Restrictions**
Cluster Name/ID: Node identification of cluster members	**Security**: Requirements for encryption, logging, and access restrictions	**Customer or Client**	**Region Restrictions**
Environment: Production, Staging, Test, and Development		**Project**	
Version		**Expiry/Project Close**	
		SLAs	

AWS Service Catalog

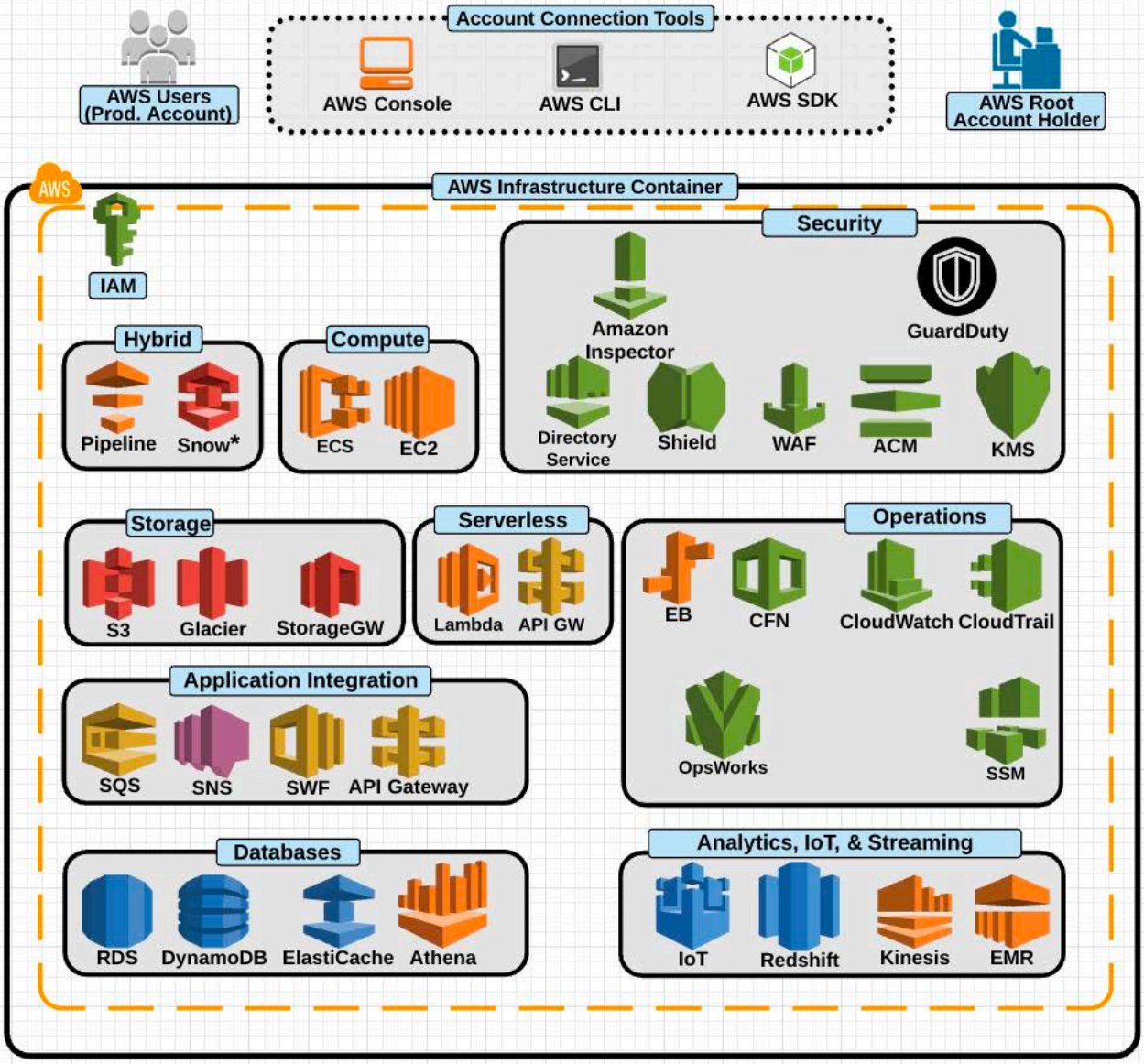
Close

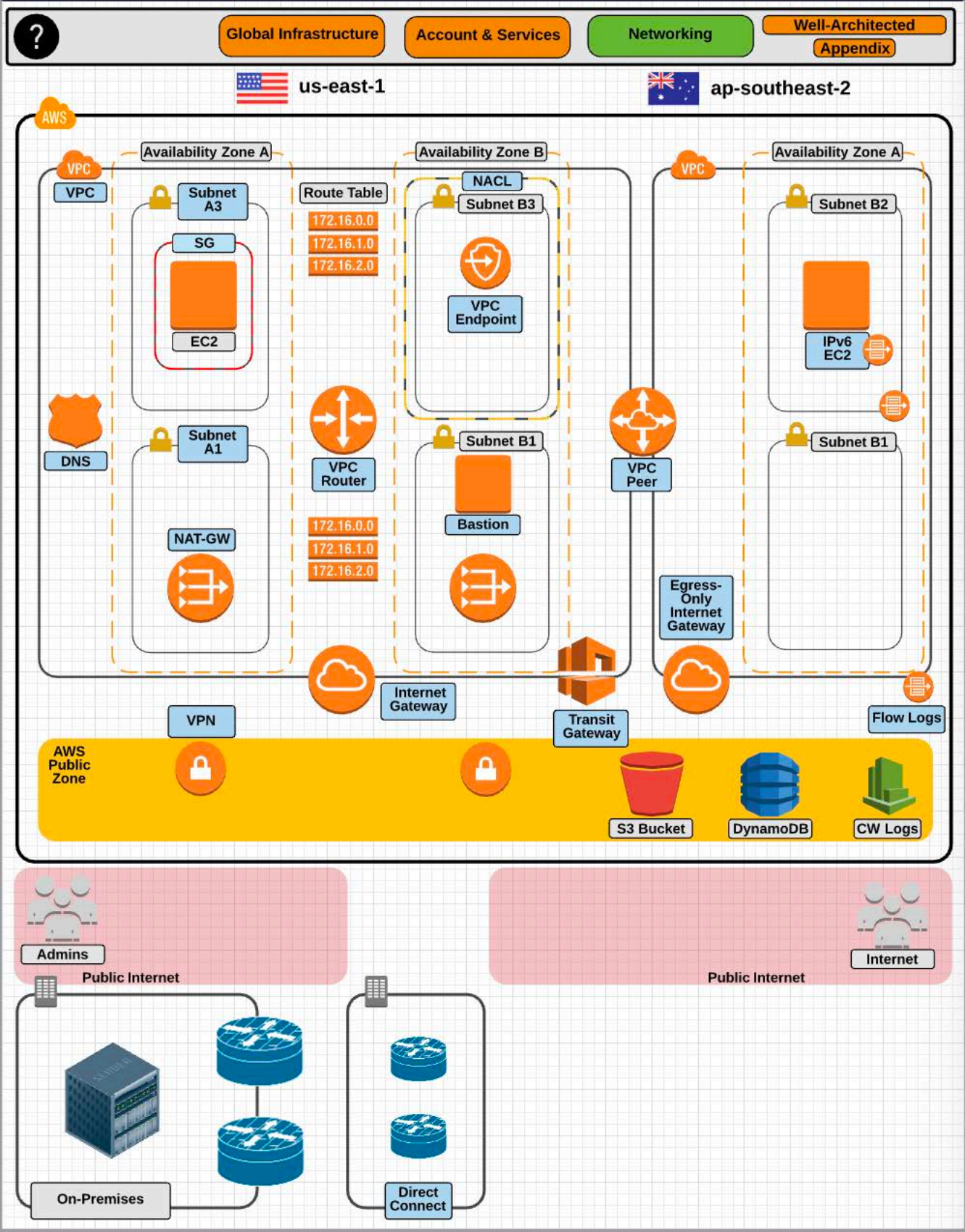
Service Catalog is a product that allows administrators to define products and portfolios (groups of products and configurations). Users can be allowed to self-service deploy these products without the usual IAM permissions required to do so directly with AWS services.

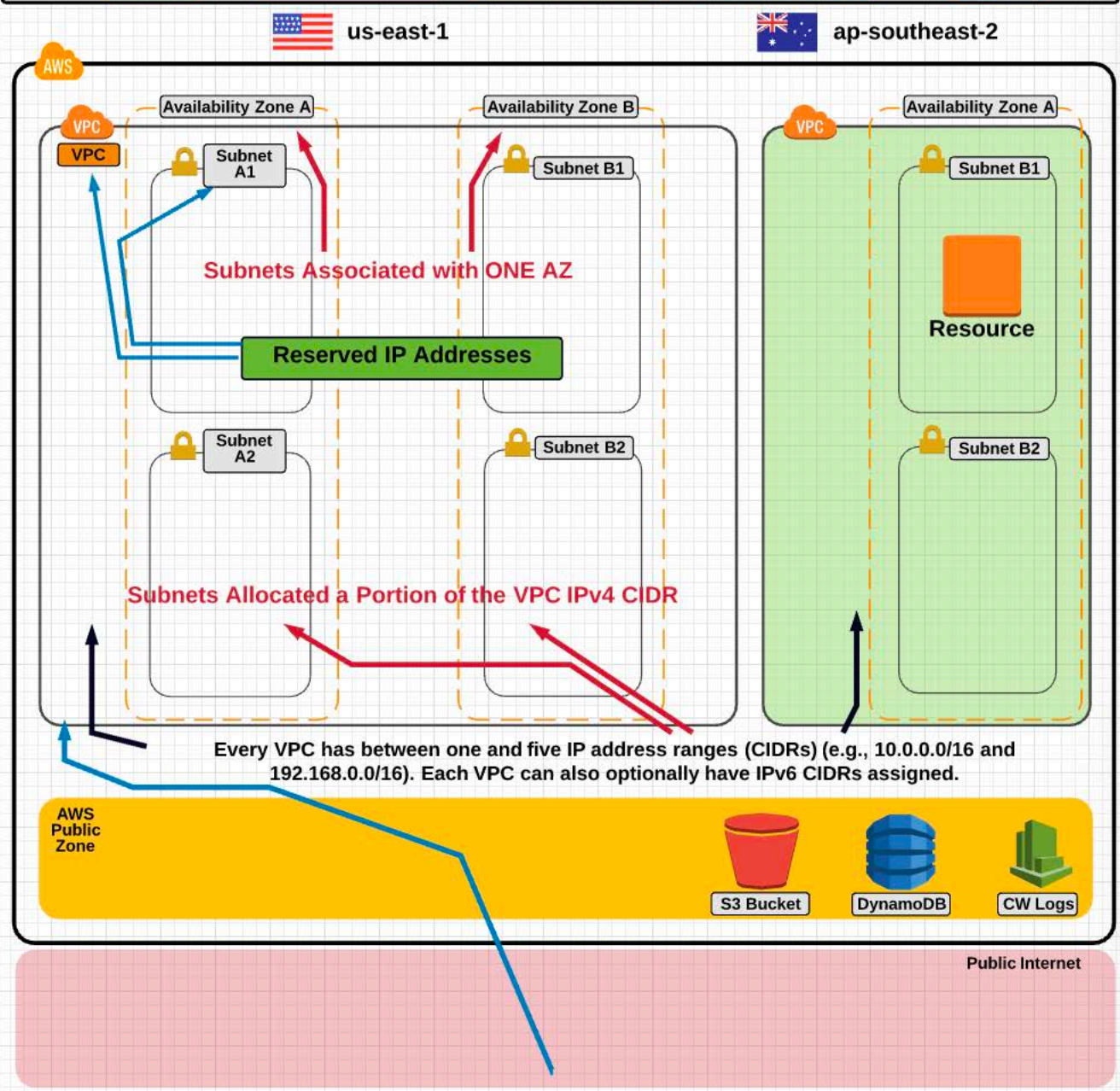



AWS
AWS
us-east-1
ap-southeast-2
AWS Account
Create Stack
Custom Resource
Lambda
Stack Role
Update Stack
Cross-Stack Reference
AWS Resources
Nested Stacks
Stack set
Stack
Notifications
sns
Delete Stack
AWSCloudFormationStackSet AdministrationRole
AWS
Stack
Stack
AWSCloudFormationStackSetExecutionRole
Public Internet

The Account and Services layer represents how we create, access, and manage an AWS account and its services, from interacting with an AWS account and managing user rights, to accessing and using various AWS services and features.



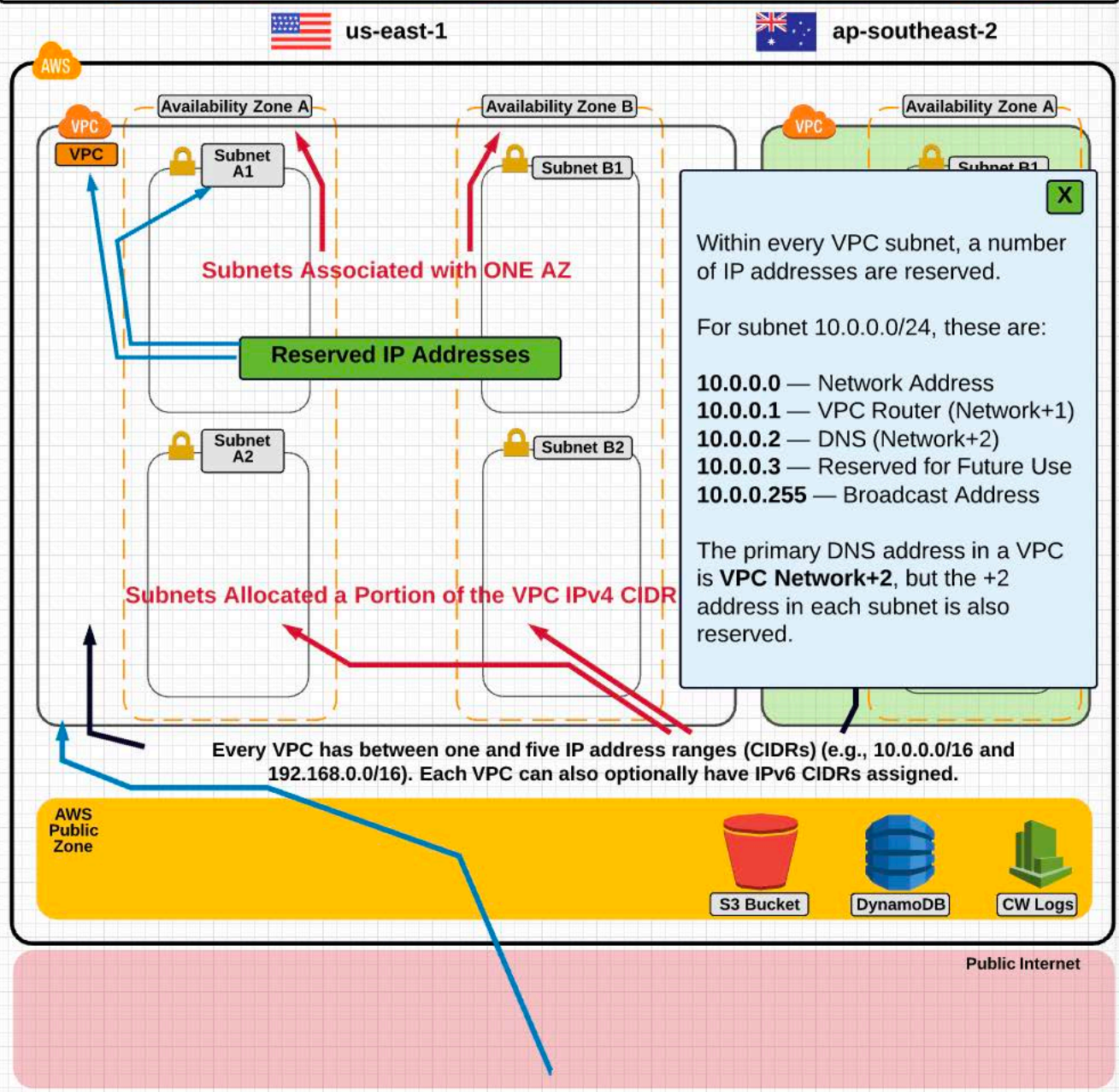

?
Global Infrastructure
Account & Services
Networking
Well-Architected
Appendix
us-east-1
ap-southeast-2
AWS
VPC
Availability Zone A
Availability Zone B
Availability Zone A
VPC
Subnet A3
SG
EC2
Route Table
172.16.0.0
172.16.1.0
172.16.2.0
NACL
Subnet B3
VPC Endpoint
VPC
Subnet B2
IPv6 EC2
DNS
Subnet A1
NAT-GW
VPC Router
172.16.0.0
172.16.1.0
172.16.2.0
Subnet B1
Bastion
VPC Peer
Subnet B1
Egress-Only Internet Gateway
Internet Gateway
Transit Gateway
Flow Logs
VPN
AWS Public Zone
S3 Bucket
DynamoDB
CW Logs
Admins
Public Internet
Internet
Public Internet
On-Premises
Direct Connect



Network Zones: Private (VPC), Public Internet, AWS Public Services

VPC Tenancy Modes: *Default* and *Dedicated*

Default mode lets you select any of the tenancy options for VPC resources later if you need to. Dedicated mode limits your future tenancy options.



X

Within every VPC subnet, a number of IP addresses are reserved.

For subnet 10.0.0.0/24, these are:

10.0.0.0 — Network Address
10.0.0.1 — VPC Router (Network+1)
10.0.0.2 — DNS (Network+2)
10.0.0.3 — Reserved for Future Use
10.0.0.255 — Broadcast Address

The primary DNS address in a VPC is **VPC Network+2**, but the +2 address in each subnet is also reserved.

Network Zones: Private (VPC), Public Internet, AWS Public Services

VPC Tenancy Modes: *Default* and *Dedicated*

Default mode lets you select any of the tenancy options for VPC resources later if you need to. Dedicated mode limits your future tenancy options.

us-east-1 | ap-southeast-2

AWS

VPC

Availability Zone A

Subnet A3

SG

EC2-1

10.10.128.0/20

Subnet A1

NAT-GW

10.10.0.0/20

172.16.0.0
172.16.1.0
172.16.2.0

VPC Router

172.16.0.0
172.16.1.0
172.16.2.0

Availability Zone B

Subnet B3

VPC Endpoint

10.10.144.0/20

NACL

Subnet B1

BASTION

10.10.16.0/20

VPC

IPv6 EC2

Internet Gateway

Internet Gateways (IGWs) require resources with allocated public IPs — they provide no benefit to resources in private subnets.

3.2.45.47

S3 Bucket

DynamoDB

AWS Public Zone

Admins

Public Internet

Internet

Destination	Target
10.10.0.0/16	local
0.0.0.0/0	nat-0d12780b15c2f8c86

Destination	Target
10.10.0.0/16	local
0.0.0.0/0	igw-997a6be1

us-east-1 | ap-southeast-2

AWS

VPC

Availability Zone A

Subnet A3

SG

EC2-1

Availability Zone B

Subnet B3

EC2-2

Subnet B1

SG

BASTION

VPC

VPC Peer

EC2

Security groups (SGs) are attached to AWS network interfaces on AWS resources. Multiple SGs can be associated with an interface, and a single SG can be associated with many interfaces.

SG processing has *no* order — all rules are evaluated at once. There is a **default implicit DENY**.

SGs *cannot* explicitly DENY traffic.

AWS Public Zone

Public Internet

Stateful

Inbound

Type	Protocol	Port Range	Source	
SSH	TCP	22	Custom	0.0.0.0/0

Outbound

Type	Protocol	Port Range	Destination	
All traffic	All	All	Custom	0.0.0.0/0

us-east-1 | ap-southeast-2

AWS

VPC

Availability Zone A

Subnet A3

SG

EC2-1

Availability Zone B

Subnet B3

EC2-2

Subnet B1

SG

BASTION

VPC

VPC Peer

EC2

Security groups (SGs) are attached to AWS network interfaces on AWS resources. Multiple SGs can be associated with an interface, and a single SG can be associated with many interfaces.

SG processing has *no* order — all rules are evaluated at once. There is a **default implicit DENY**.

SGs *cannot* explicitly DENY traffic.

Security groups are *stateful*. If traffic is allowed in, the corresponding return traffic is automatically allowed. This cannot be changed. The same logic applies for outgoing traffic — its return communications are automatically permitted.

This means you don't have to worry about adding ephemeral ports.

Stateful

Inbound

Type	Protocol	Port Range	Source	
SSH	TCP	22	Custom	0.0.0.0/0

Outbound

Type	Protocol	Port Range	Destination	
All traffic	All	All	Custom	0.0.0.0/0

us-east-1 | ap-southeast-2

AWS

VPC

Availability Zone A

Subnet A3

SG

EC2-1

sg-001b643189c09f816

Availability Zone B

Subnet B3

EC2-2

Subnet B1

SG

BASTION

sg-015dfafb1f9b00dd4

VPC

VPC Peer

EC2

Security groups (SGs) are attached to AWS network interfaces on AWS resources. Multiple SGs can be associated with an interface, and a single SG can be associated with many interfaces.

SG processing has *no* order — all rules are evaluated at once. There is a **default implicit DENY**.

SGs *cannot* explicitly DENY traffic.

Security groups are aware of AWS logical resources. This means a security group's *Source* or *Destination* field can reference CIDRs or logical resources including EC2 instances, other security groups, or even itself.

X

Source	
Custom	sg-001b643189c09f816
Custom	sg-015dfafb1f9b00dd4

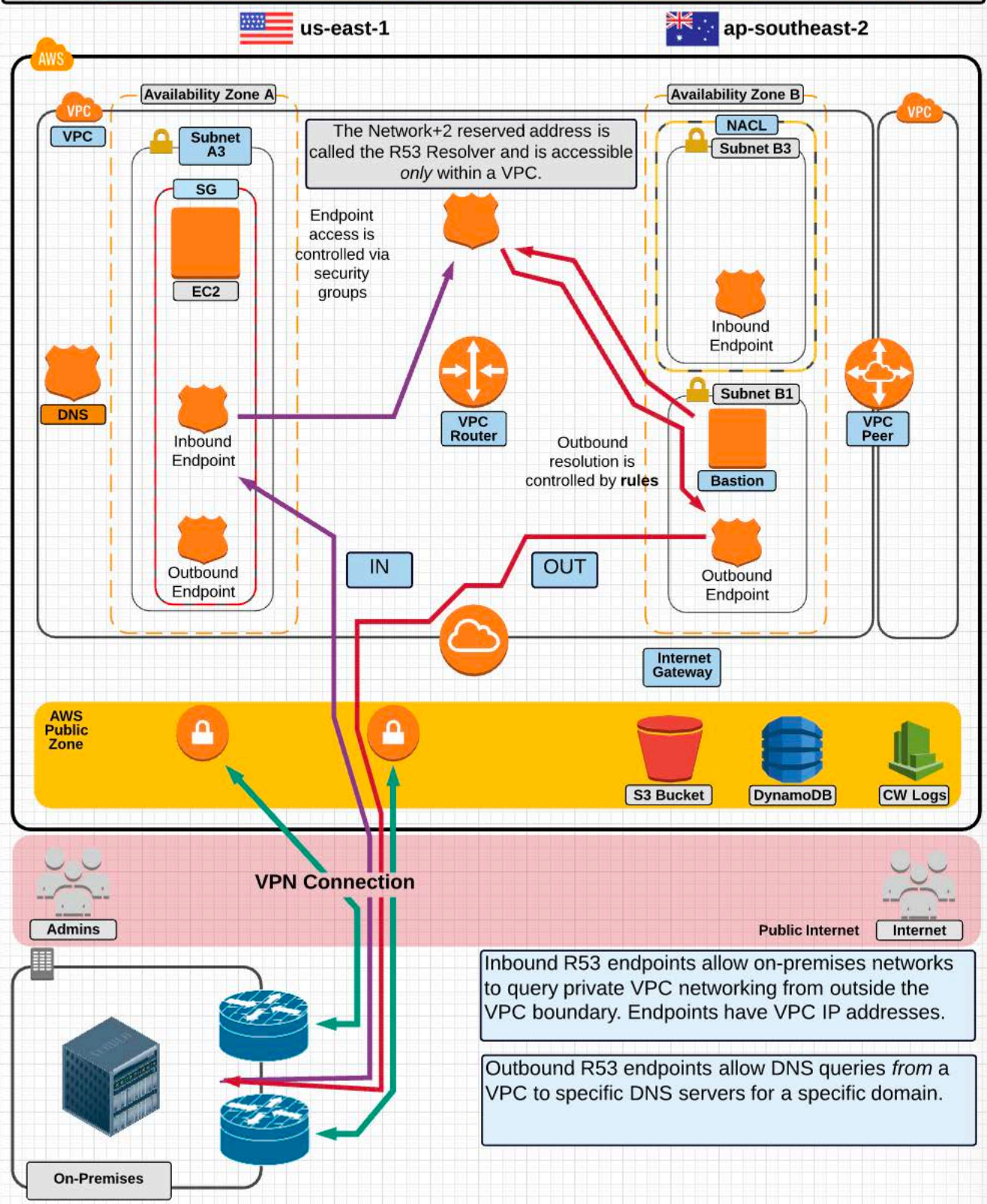
Stateful

Inbound

Type	Protocol	Port Range	Source	
SSH	TCP	22	Custom	0.0.0.0/0

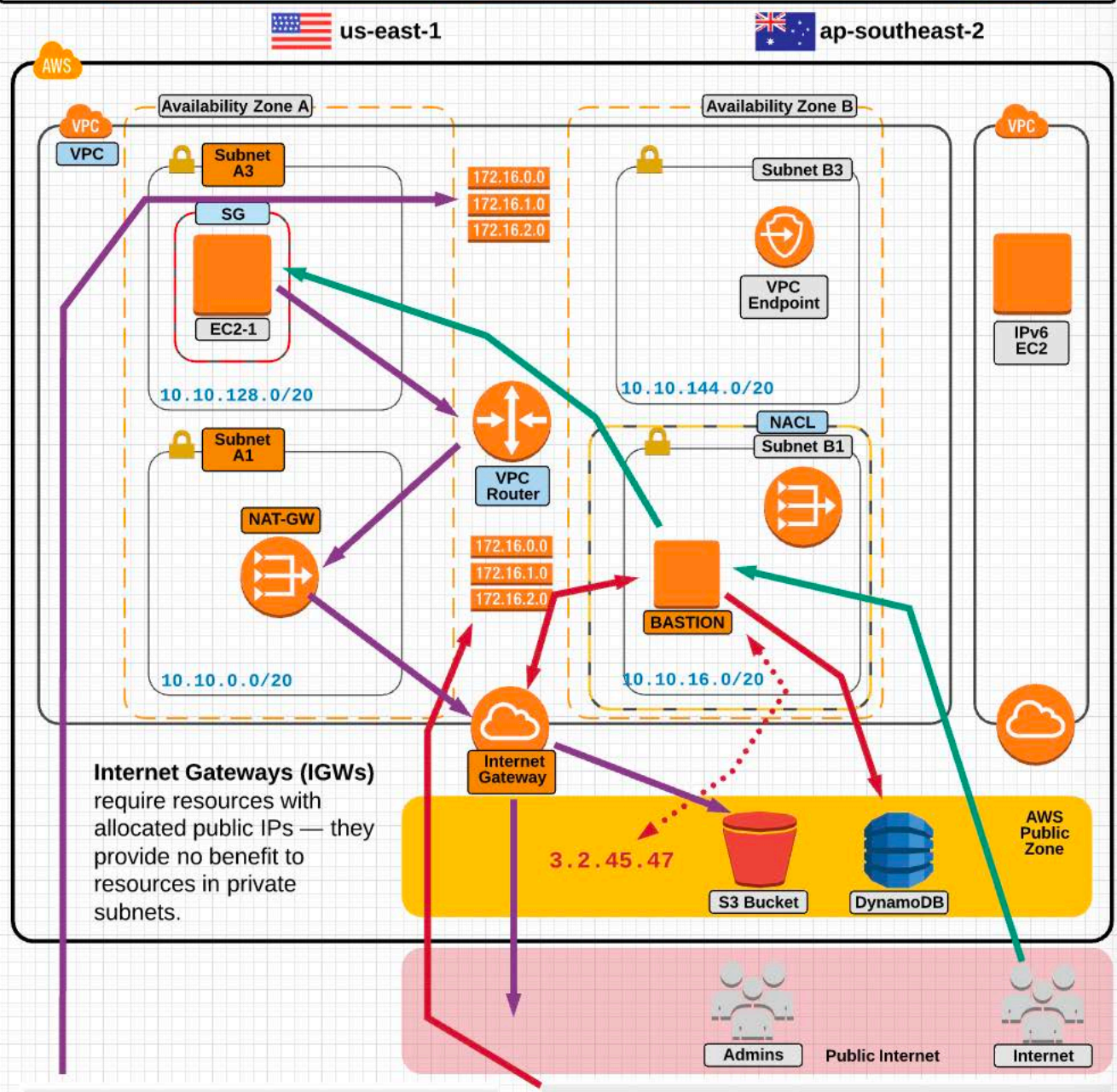
Outbound

Type	Protocol	Port Range	Destination	
All traffic	All	All	Custom	0.0.0.0/0



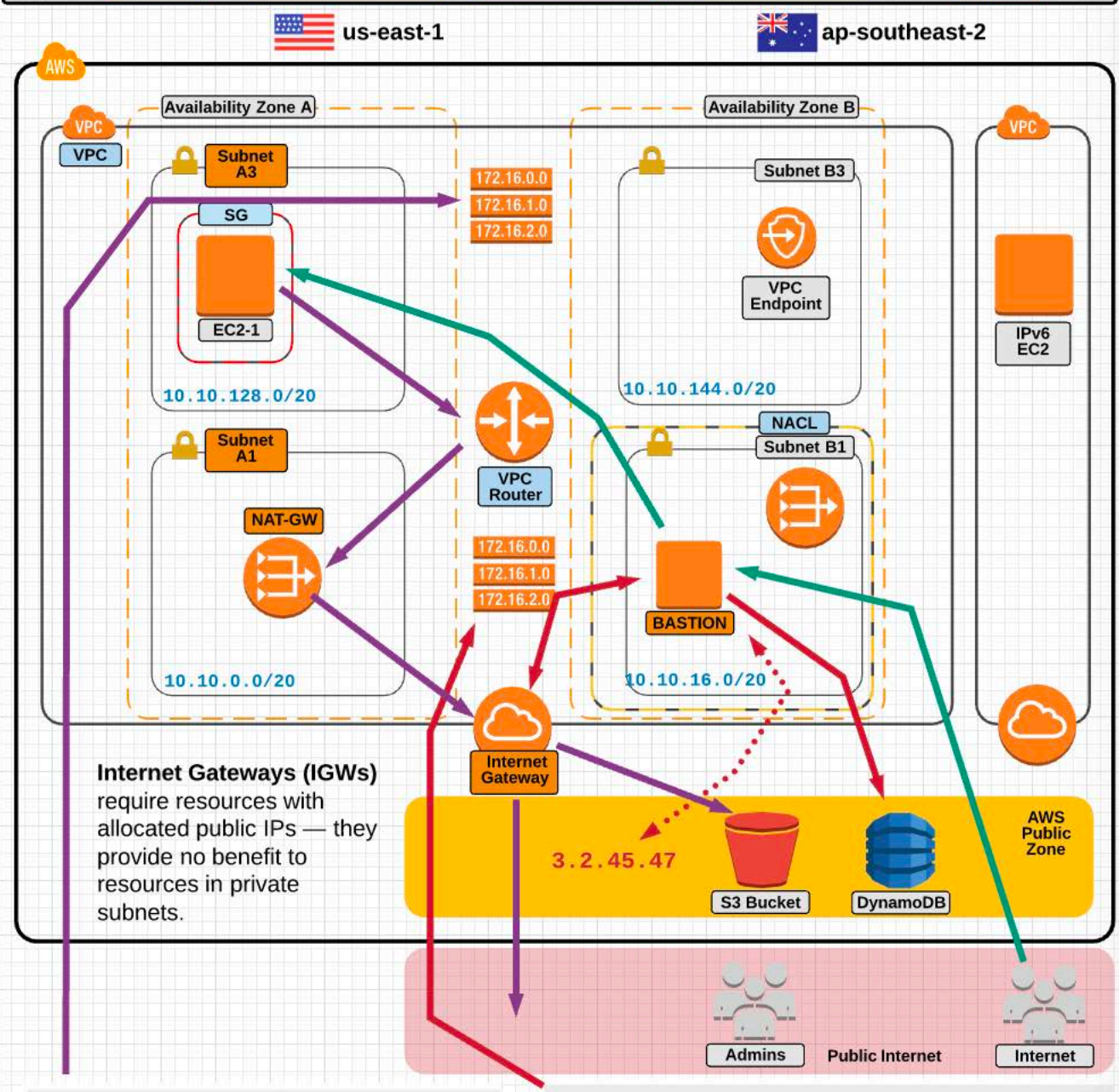
Inbound R53 endpoints allow on-premises networks to query private VPC networking from outside the VPC boundary. Endpoints have VPC IP addresses.

Outbound R53 endpoints allow DNS queries *from* a VPC to specific DNS servers for a specific domain.



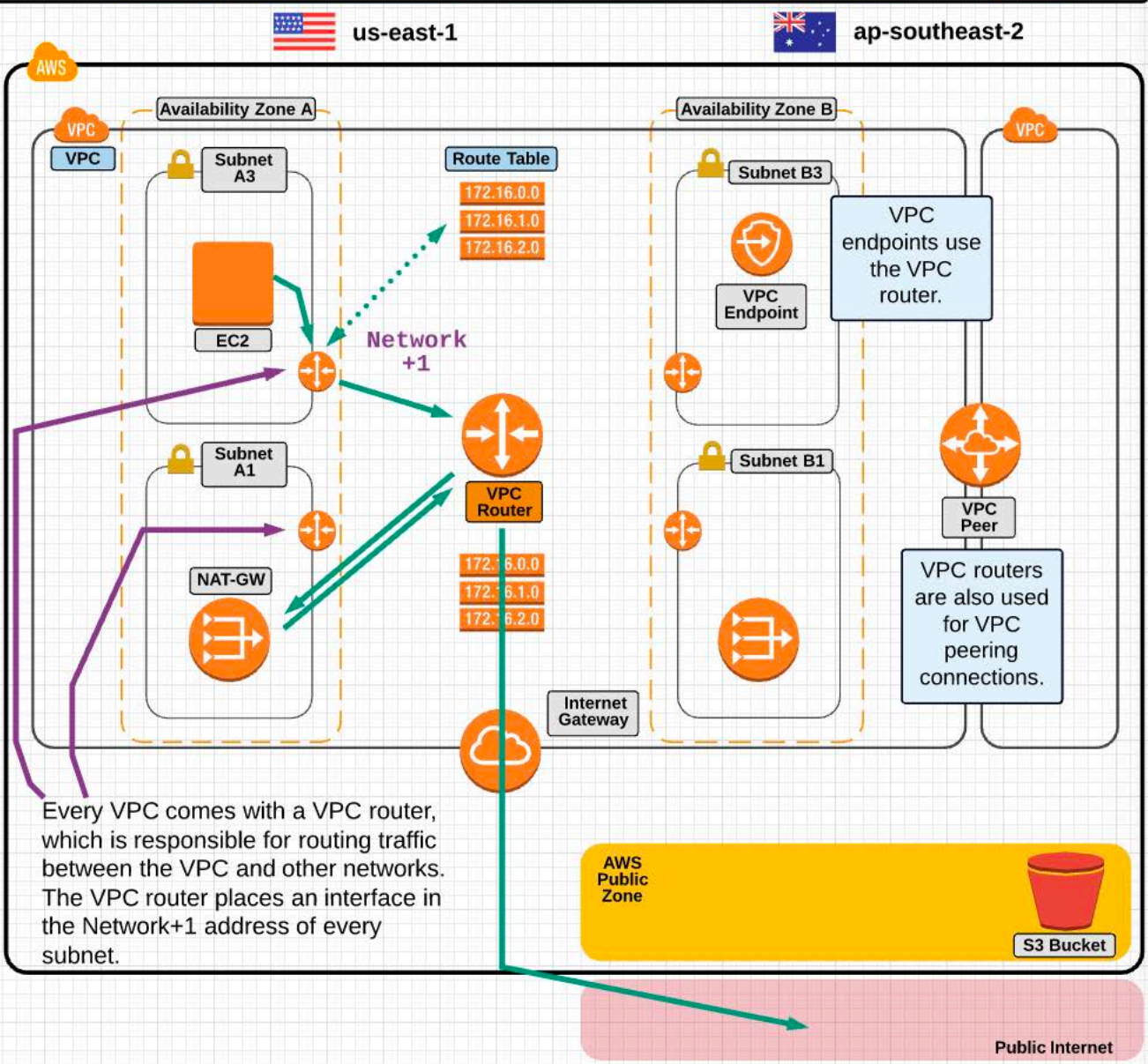
Destination	Target
10.10.0.0/16	local
0.0.0.0/0	nat-0d12780b15c2f8c86

Destination	Target
10.10.0.0/16	local
0.0.0.0/0	igw-997a6be1



Destination	Target
10.10.0.0/16	local
0.0.0.0/0	nat-0d12780b15c2f8c86

Destination	Target
10.10.0.0/16	local
0.0.0.0/0	igw-997a6be1


us-east-1
ap-southeast-2
AWS
VPC
Availability Zone A
Availability Zone B
VPC
Subnet A3
Route Table
172.16.0.0
172.16.1.0
172.16.2.0
Subnet B3
VPC Endpoint
VPC endpoints use the VPC router.
EC2
Network +1
Subnet A1
VPC Router
Subnet B1
VPC Peer
NAT-GW
Internet Gateway
VPC routers are also used for VPC peering connections.
Every VPC comes with a VPC router, which is responsible for routing traffic between the VPC and other networks. The VPC router places an interface in the Network+1 address of every subnet.
AWS Public Zone
S3 Bucket
Public Internet

us-east-1 ap-southeast-2

AWS

VPC

Availability Zone A

Subnet A3

EC2

Subnet A1

NAT-GW

Route Table

172.16.0.0
172.16.1.0
172.16.2.0

Network +1

VPC Router

172.16.0.0
172.16.1.0
172.16.2.0

Internet Gateway

Availability Zone B

Subnet B3

VPC Endpoint

Subnet B1

VPC

VPC Peer

VPC endpoints use the VPC router.

VPC routers are also used for VPC peering connections.

Every VPC comes with a VPC router, which is responsible for routing traffic between the VPC and other networks. The VPC router places an interface in the Network+1 address of every subnet.

AWS Public Zone

S3 Bucket

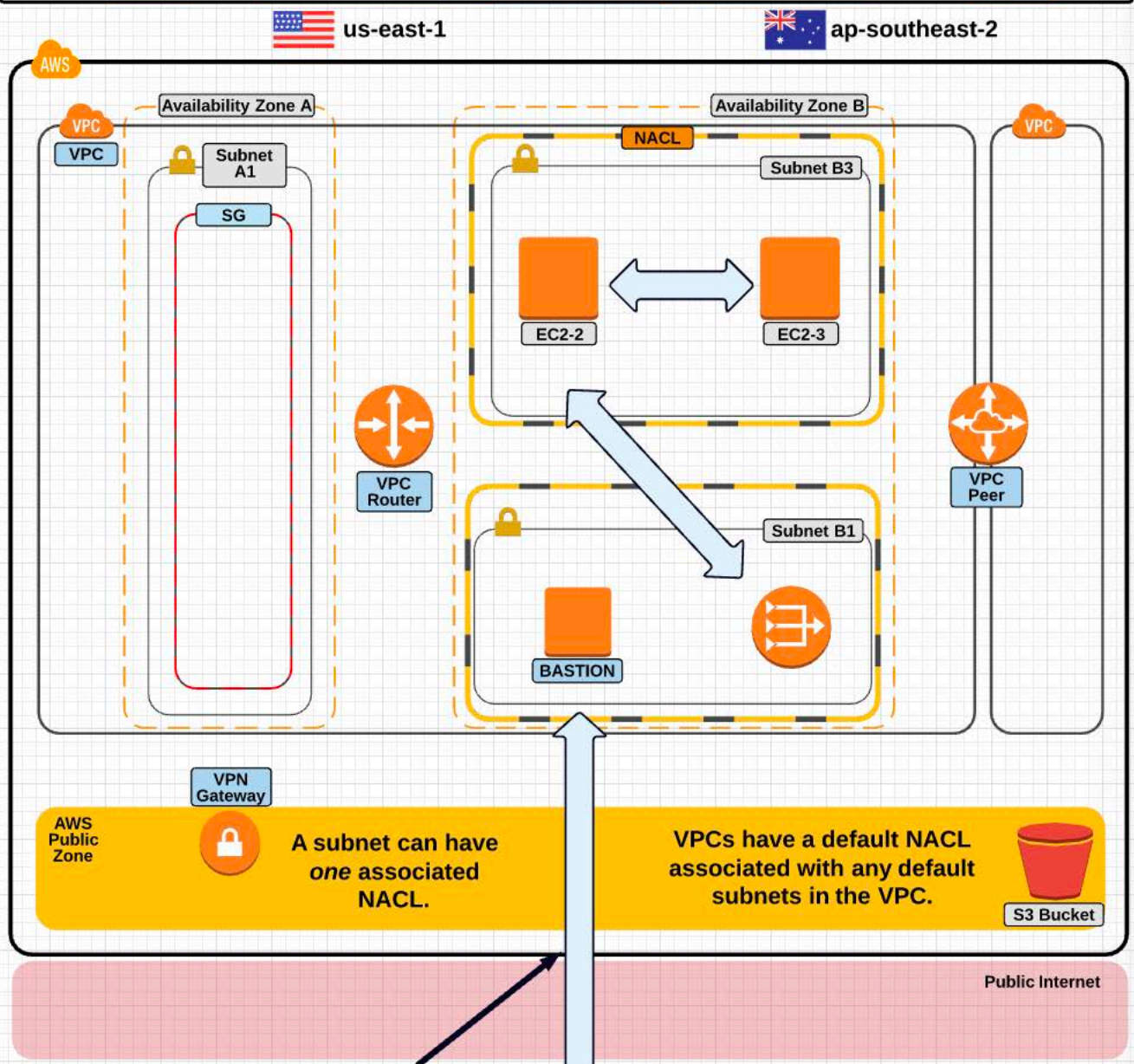
Public Internet

The VPC router is controlled by route tables (RTs).
VPCs come with a default "main" route table.
Custom RTs can be defined and attached to subnets.

See VPN Lesson X

Route tables contain a **default** "**local**" route for the VPC.
Routes map to a **destination** and tell the VPC router where to direct IP traffic (at the **target**). More specific routes take priority (e.g., /32 > /16).

Destination	Target	Status	Propagated
10.10.0.0/16	local	active	No
0.0.0.0/0	igw-997a6be1	active	No

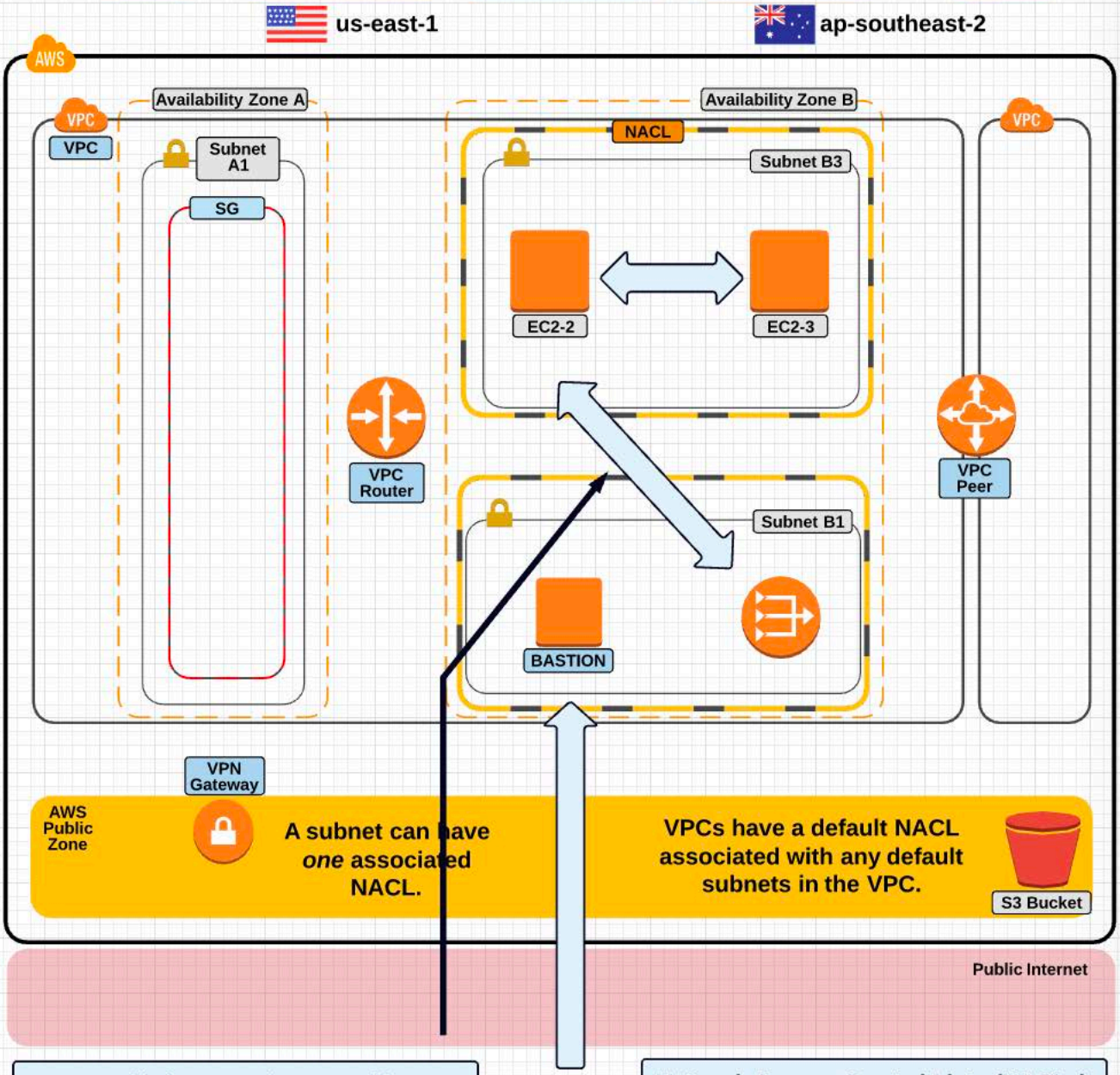


A NACL placed around the bastion and NATGW subnet could DENY inbound traffic from the public internet, control outbound traffic from the NATGW, and filter which EC2 instances are able to reach the NATGW.

NACLs are often used to provide explicit DENY protection to internet subnets.

Network Access Control Lists (NACLs) are firewalls attached to VPC subnets. They filter IP traffic entering or leaving the subnet. NACLs are *stateless*, which means ephemeral port rules are needed in order to permit response traffic.

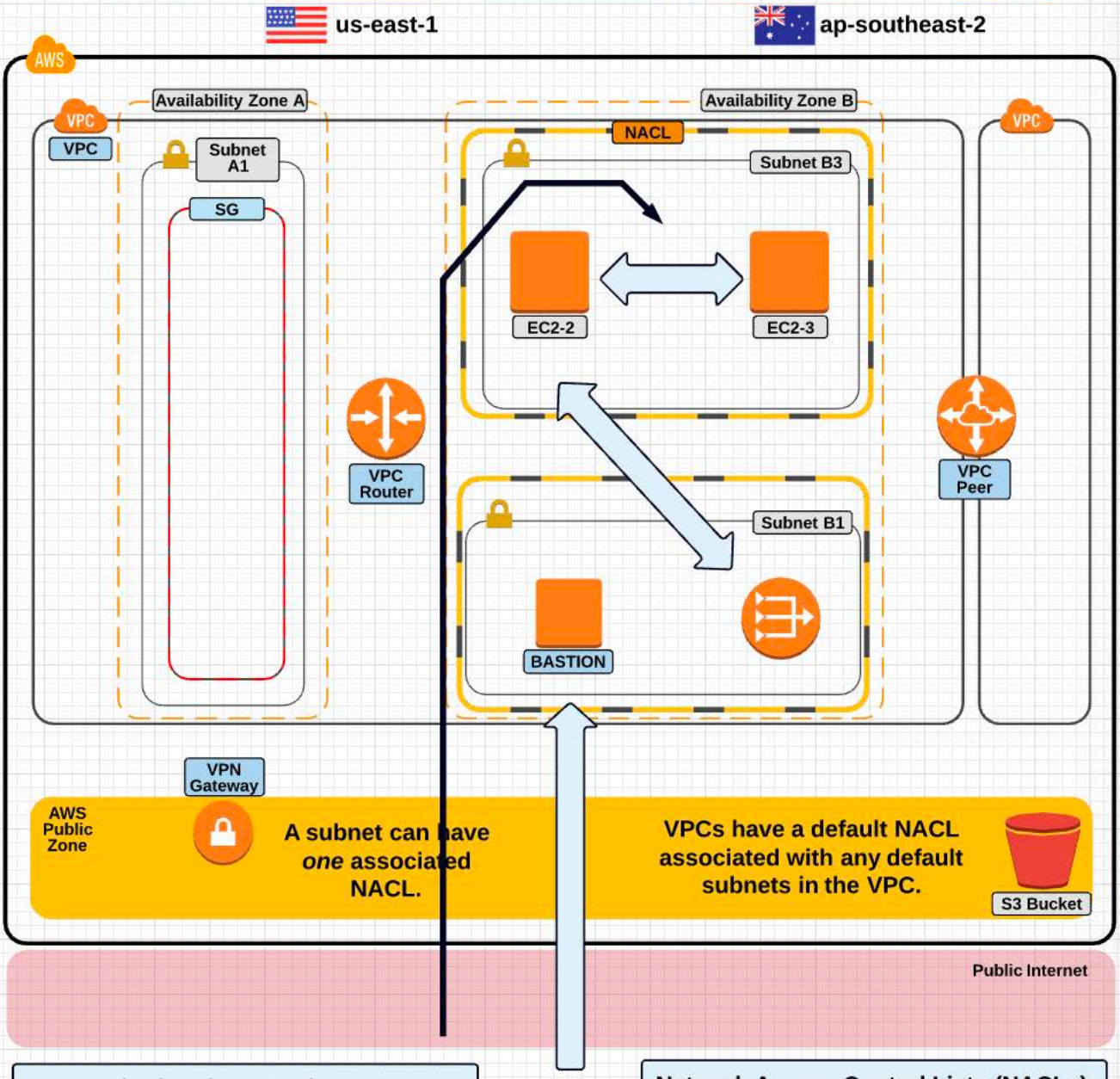
NACLs can explicitly DENY and their rules are processed in order.



Any IP traffic between the top and bottom subnets would be subject to processing out of the top subnet and in to the bottom subnet. The same applies for return traffic.

Network Access Control Lists (NACLs) are firewalls attached to VPC subnets. They filter IP traffic entering or leaving the subnet. NACLs are *stateless*, which means ephemeral port rules are needed in order to permit response traffic.

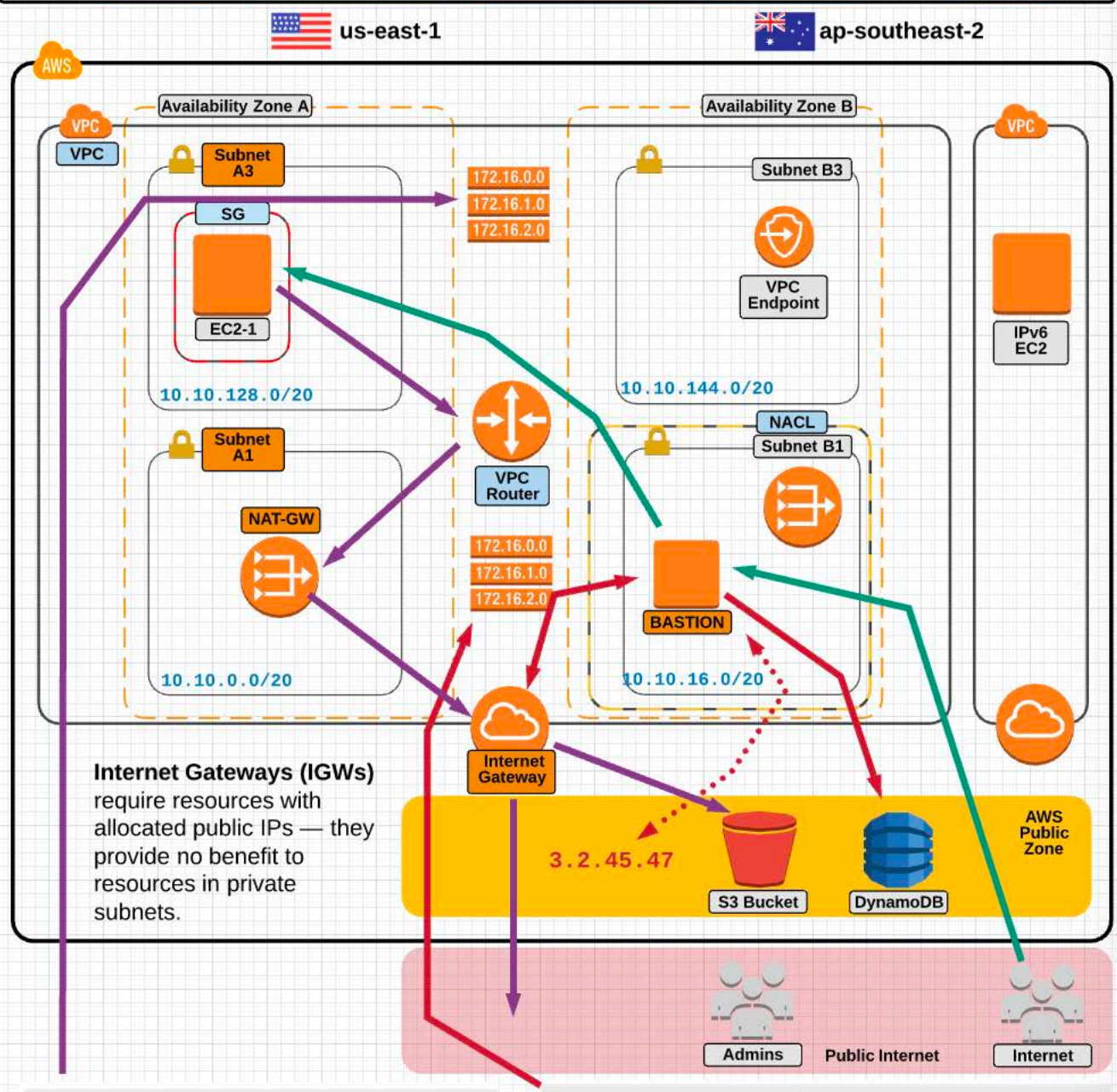
NACLs can explicitly DENY and their rules are processed in order.



Communications between the EC2 instances would be unaffected by any NACLs — NACL processing only occurs when traffic crosses over a subnet boundary (in or out).

Network Access Control Lists (NACLs) are firewalls attached to VPC subnets. They filter IP traffic entering or leaving the subnet. NACLs are *stateless*, which means ephemeral port rules are needed in order to permit response traffic.

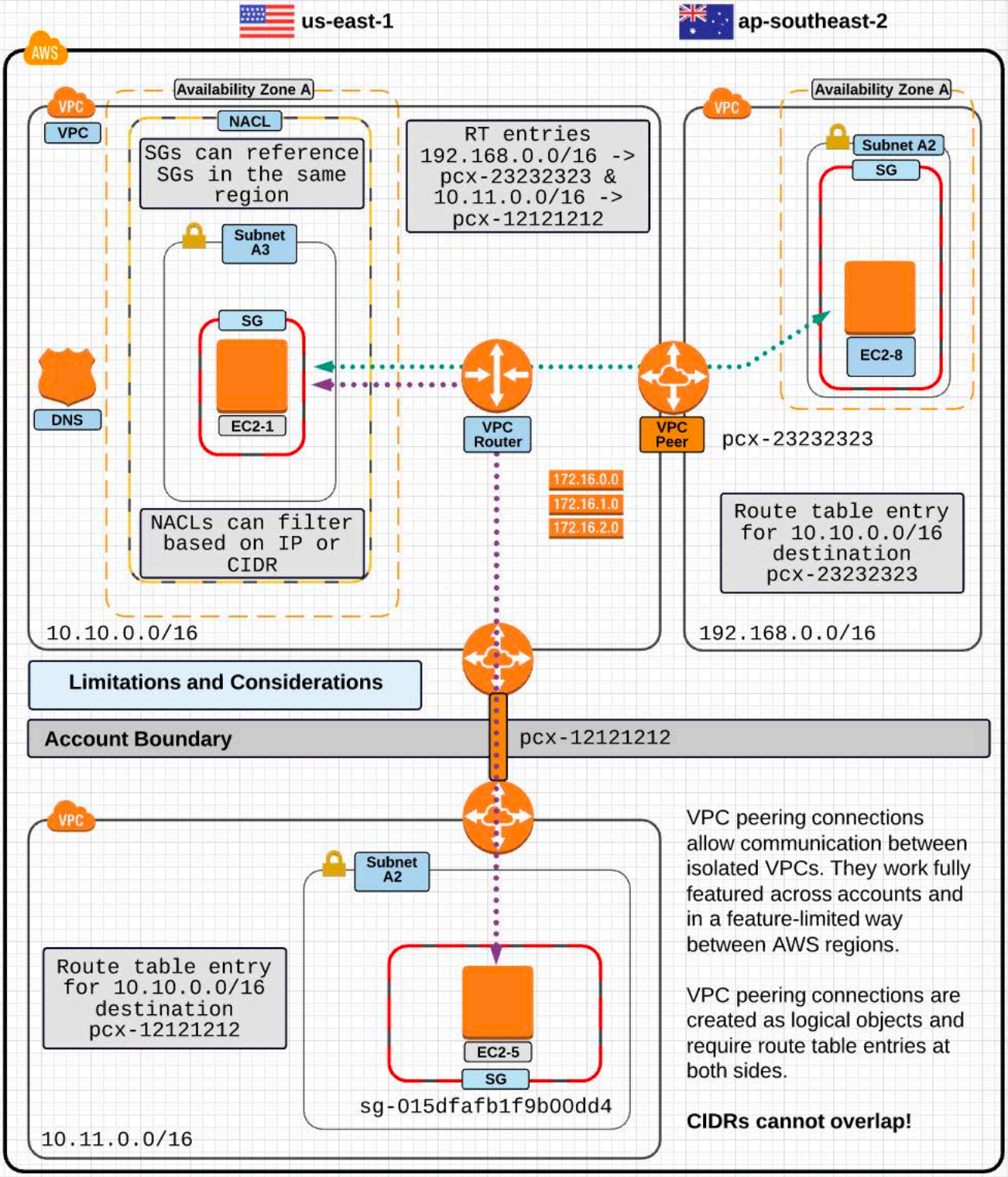
NACLs can explicitly DENY and their rules are processed in order.

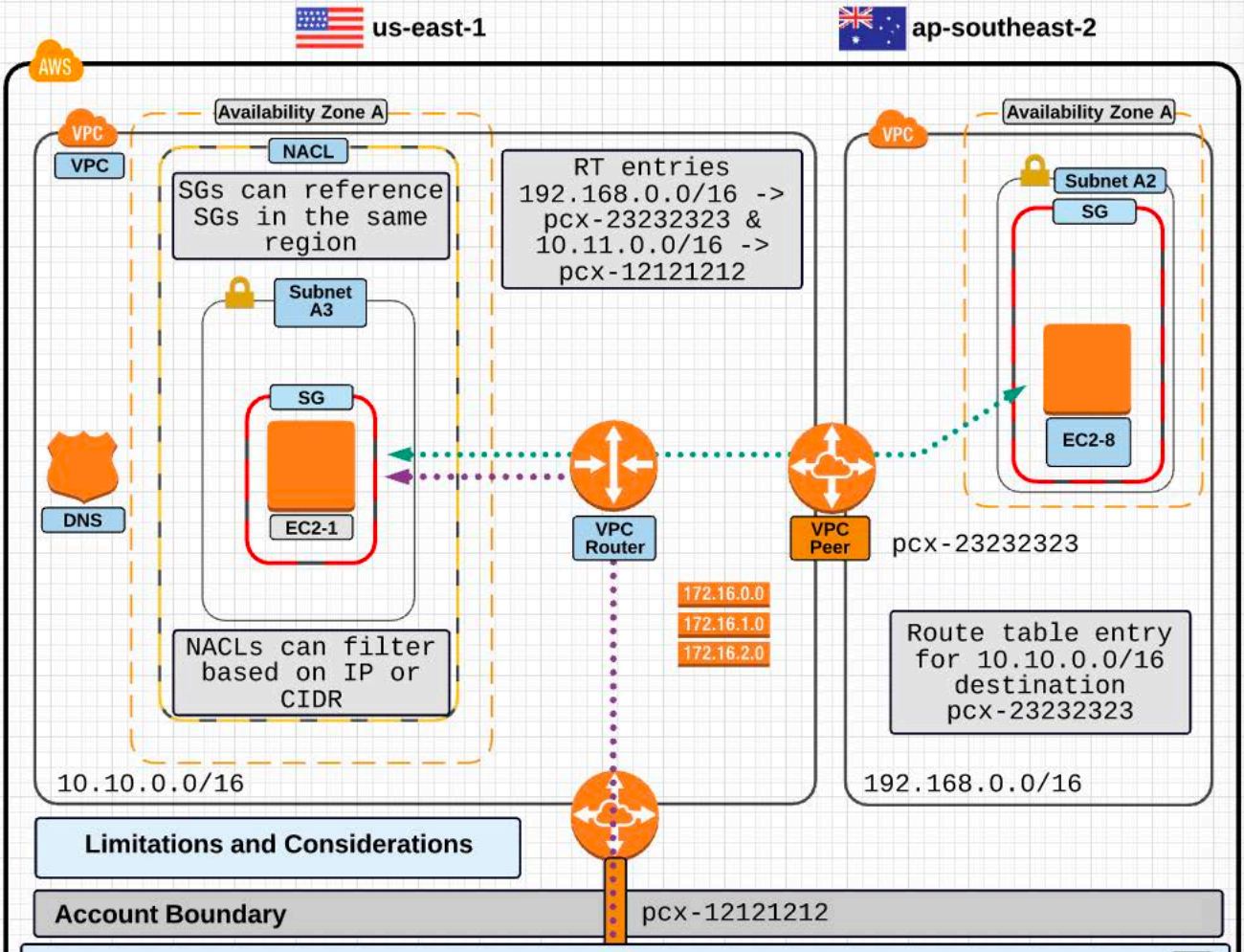


Internet Gateways (IGWs) require resources with allocated public IPs — they provide no benefit to resources in private subnets.

Destination	Target
10.10.0.0/16	local
0.0.0.0/0	nat-0d12780b15c2f8c86

Destination	Target
10.10.0.0/16	local
0.0.0.0/0	igw-997a6be1


us-east-1
ap-southeast-2
AWS
VPC
Availability Zone A
NACL
SGs can reference SGs in the same region
Subnet A3
SG
EC2-1
DNS
NACLs can filter based on IP or CIDR
10.10.0.0/16
RT entries
192.168.0.0/16 -> pcx-23232323 &
10.11.0.0/16 -> pcx-12121212
VPC Router
172.16.0.0
172.16.1.0
172.16.2.0
VPC Peer
pcx-23232323
Subnet A2
EC2-8
Route table entry for 10.10.0.0/16 destination pcx-23232323
192.168.0.0/16
Limitations and Considerations
Account Boundary
pcx-12121212
Route table entry for 10.10.0.0/16 destination pcx-12121212
EC2-5
sg-015dfafb1f9b00dd4
10.11.0.0/16
VPC peering connections allow communication between isolated VPCs. They work fully featured across accounts and in a feature-limited way between AWS regions.
VPC peering connections are created as logical objects and require route table entries at both sides.
CIDRs cannot overlap!
Public Internet



Limitations and Considerations

X

Intra-Region VPC Peering

Peering connections cannot peer VPCs with overlapping CIDRs.

Peering is *not* transitive. Peering A and B, then B and C, does not mean A and C are peered.

SGs can reference groups in remote VPCs. SGs and NACLs can also reference CIDR/IP.

IPv6 is supported, but not by default.

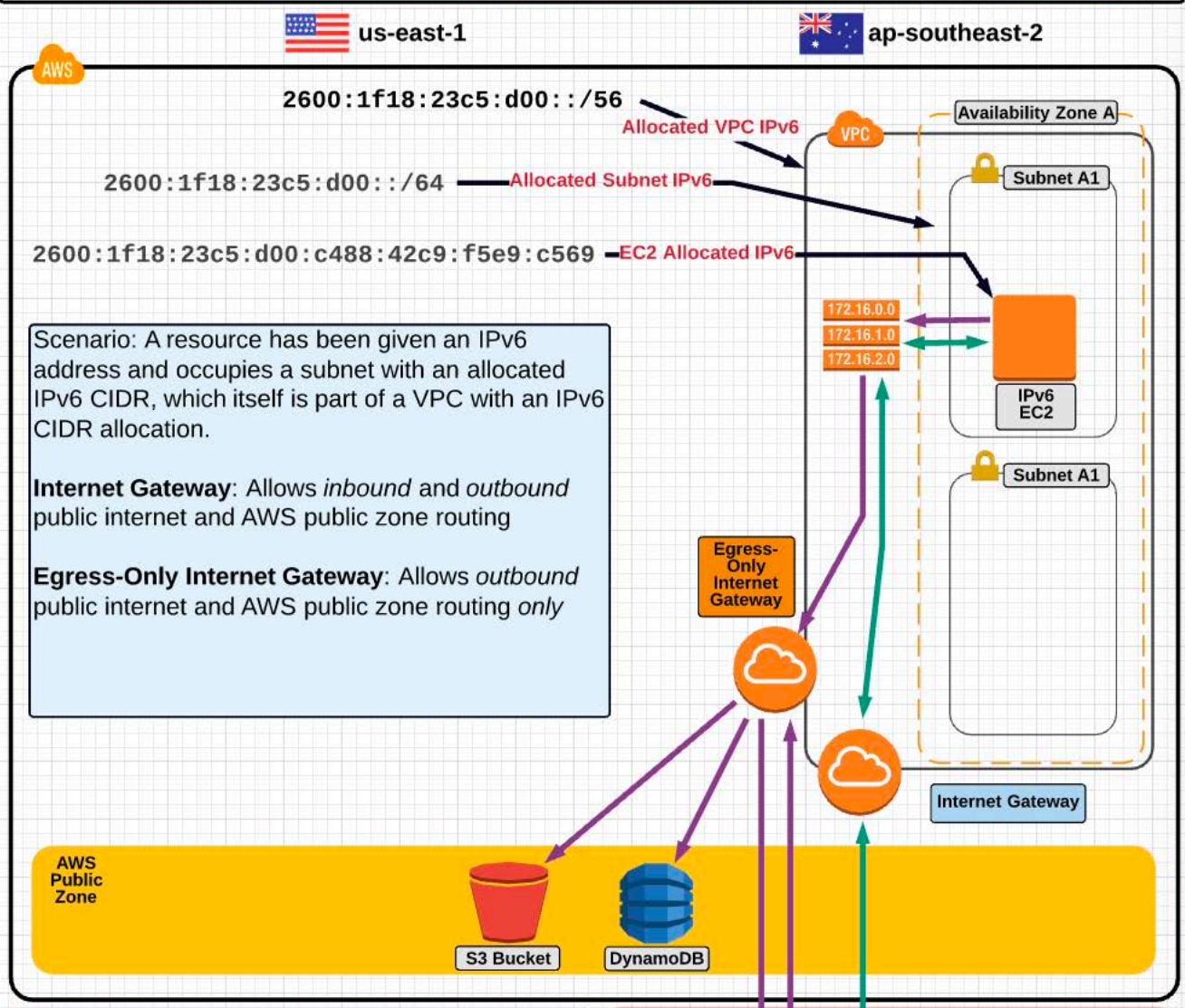
Private DNS hostnames can be resolved.

Inter-region VPC peering has the following additional limitations:

Inter (cross-) region peered VPCs do not support IPv6.

Security group logical referencing does *not* work cross-region.

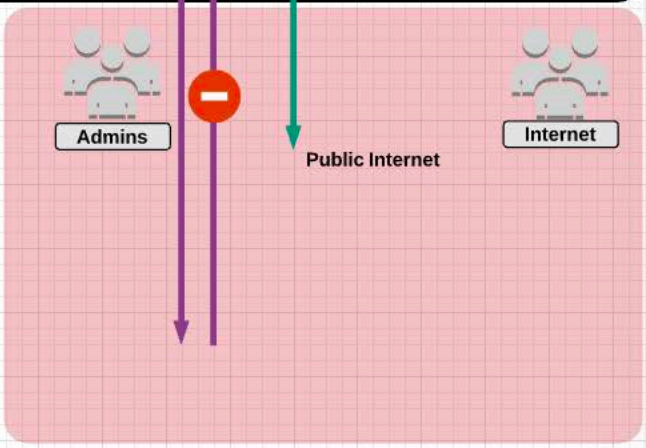
Private DNS hostnames can be resolved but *must* be enabled.



Scenario: A resource has been given an IPv6 address and occupies a subnet with an allocated IPv6 CIDR, which itself is part of a VPC with an IPv6 CIDR allocation.

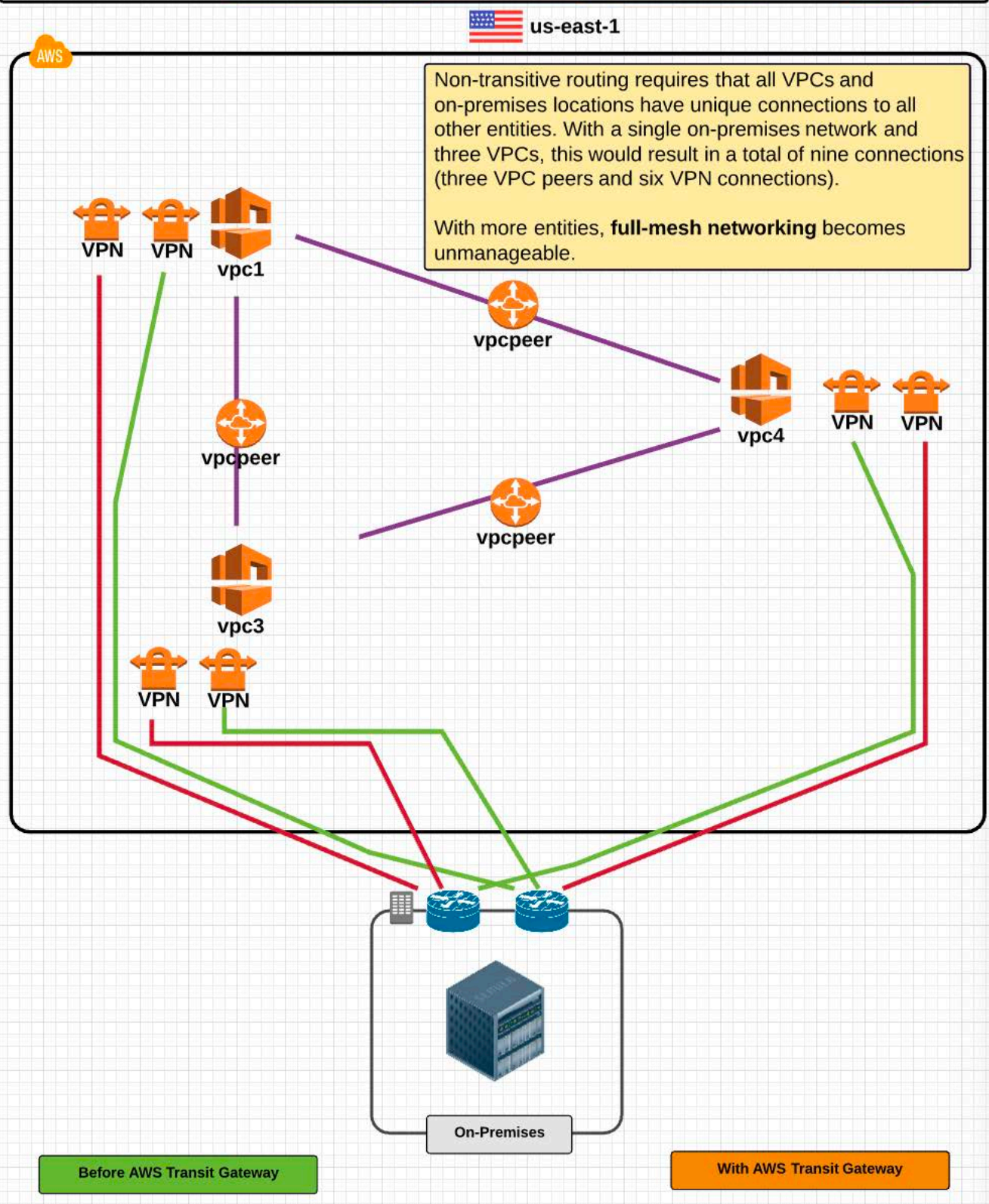
Internet Gateway: Allows *inbound* and *outbound* public internet and AWS public zone routing

Egress-Only Internet Gateway: Allows *outbound* public internet and AWS public zone routing *only*



Resources inside a public subnet with IPv6 allocations are provided with publicly routable addresses because *all* IPv6 addresses are publicly routable.

As a result, **NAT gateways** aren't a valid option to provide outgoing-only access to the AWS public zone or public internet. NAT gateways translate between private-only IP addresses to a public address.


us-east-1
AWS
Non-transitive routing requires that all VPCs and on-premises locations have unique connections to all other entities. With a single on-premises network and three VPCs, this would result in a total of nine connections (three VPC peers and six VPN connections).
With more entities, **full-mesh networking** becomes unmanageable.
VPN
VPN
vpc1
vpcpeer
vpc4
VPN
VPN
vpcpeer
vpcpeer
vpc3
VPN
VPN
On-Premises
Before AWS Transit Gateway
With AWS Transit Gateway

us-east-1

AWS

Transit gateways replace virtual private gateways for VPC and on-premises connections. Transit gateways can be attached to VPCs and VPNs (along with DX) and can perform transitive routing, allowing for hub-and-spoke architectures.

vpc1

VPC Attachment

Transit Gateway

VPC Attachment

vpc4

vpc3

VPC Attachment

VPN Attachment

VPN Attachment

VPN

VPN

Transit gateways have multiple route tables, allowing complex routing hierarchies to be defined — from hub-and-spoke architectures to edge consolidation and everything in between.

On-Premises

Before AWS Transit Gateway

With AWS Transit Gateway

us-east-1 | ap-southeast-2

AWS

```
version
account-id
interface-id
srcaddr
dstaddr
srcport
dstport
protocol
packets
bytes
start
end
action
log-status
```

LOG

VPC

Availability Zone A

Subnet B3

EC2

EC2

Subnet B1

Flow Logs

What Isn't Logged?

DHCP
AWS DNS
Metadata
License Activation Requests

AWS Public Zone

CW Logs

S3 Bucket

Public Internet

Admins

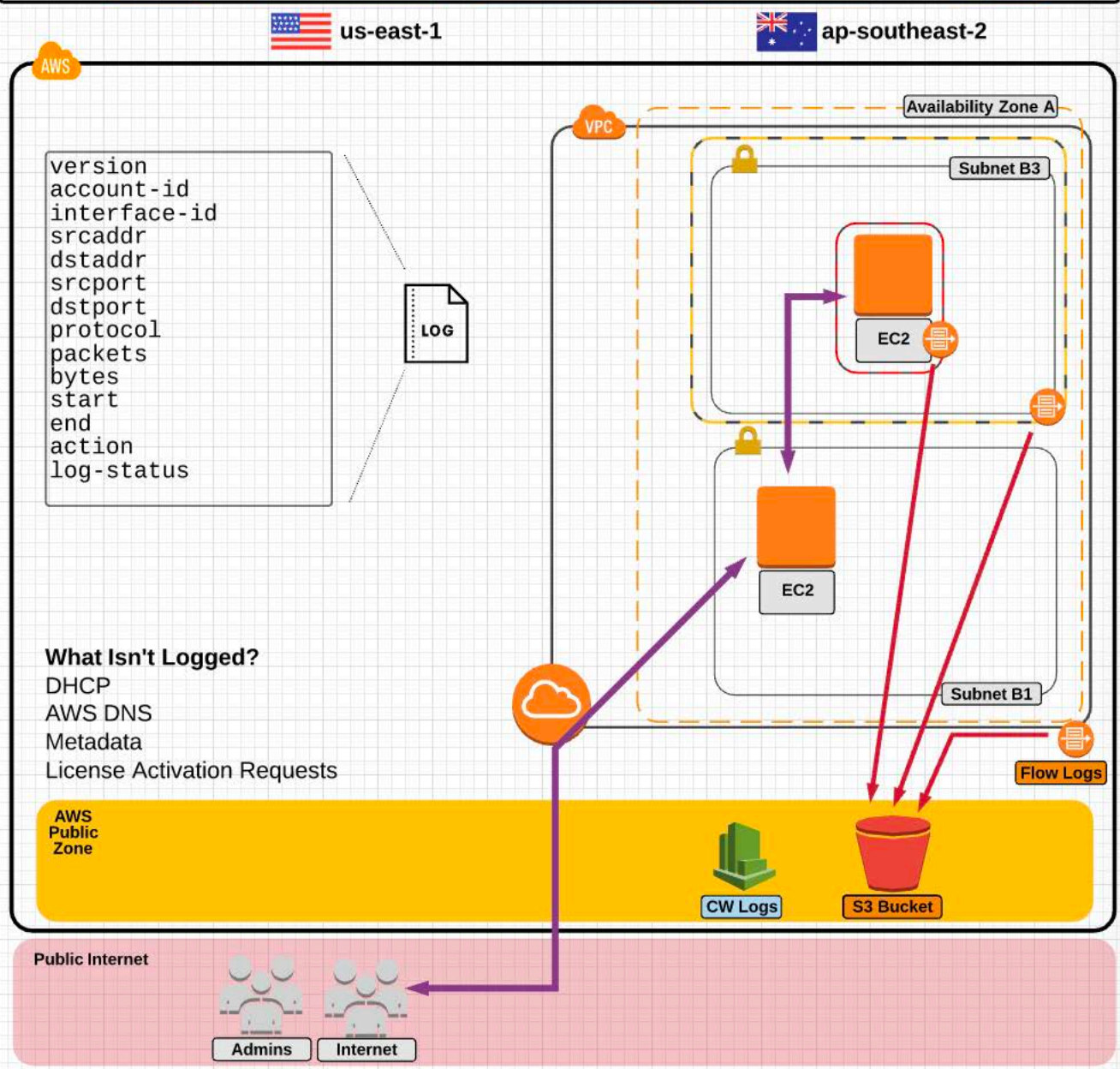
Internet

IAM roles provide permissions to the VPC Flow Logs service, either for CloudWatch log injection or for sending log files to an S3 bucket for storage.

IAM Role

VPC Flow Logs gather *metadata* for traffic flowing across a VPC. They don't log actual traffic, nor are they real-time. They collect data at one of three levels:

- VPC: All traffic to/from/inside the VPC
- Subnet: To/from/inside the subnet
- ENI: To/from the ENI



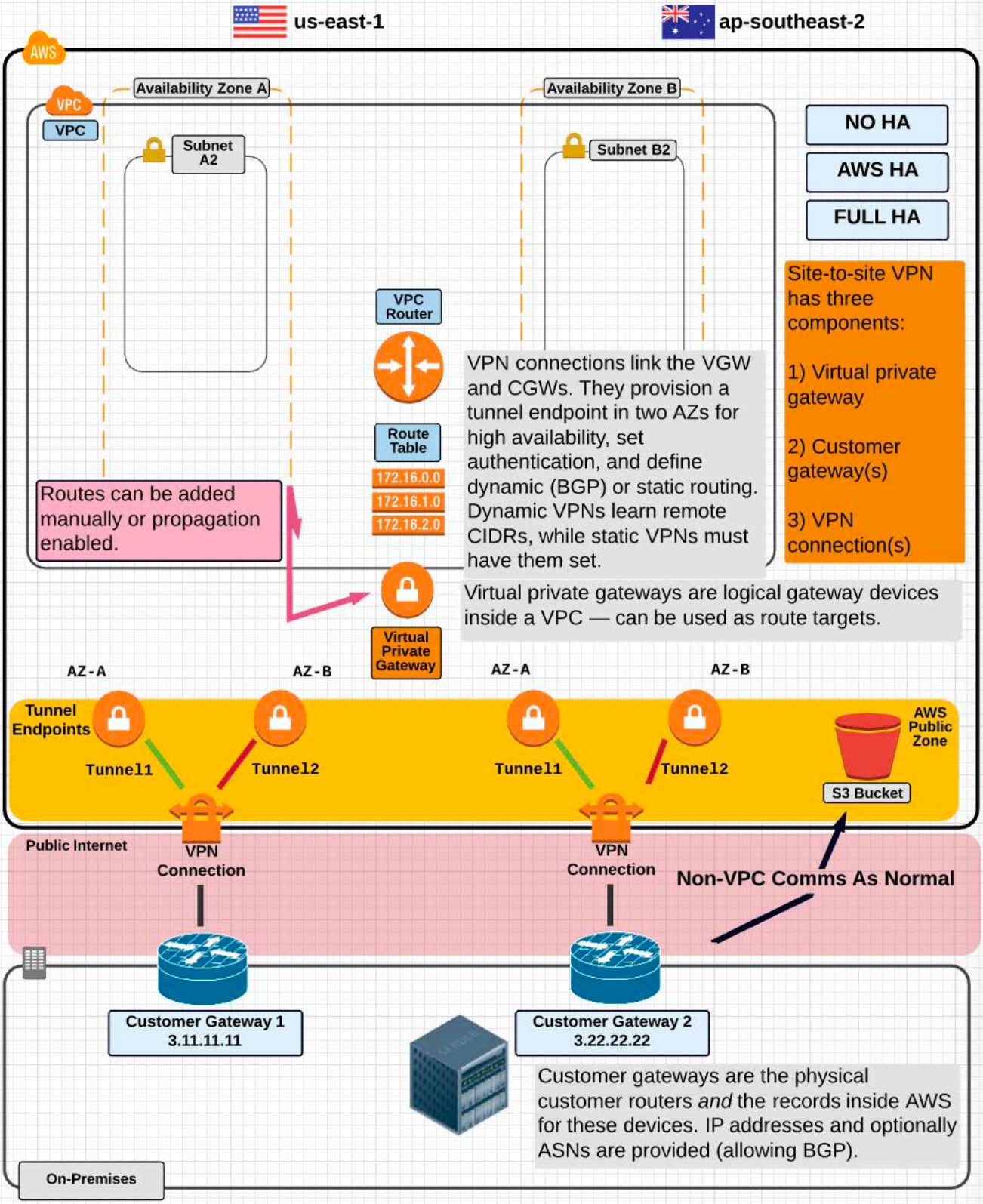
What Isn't Logged?
DHCP
AWS DNS
Metadata
License Activation Requests

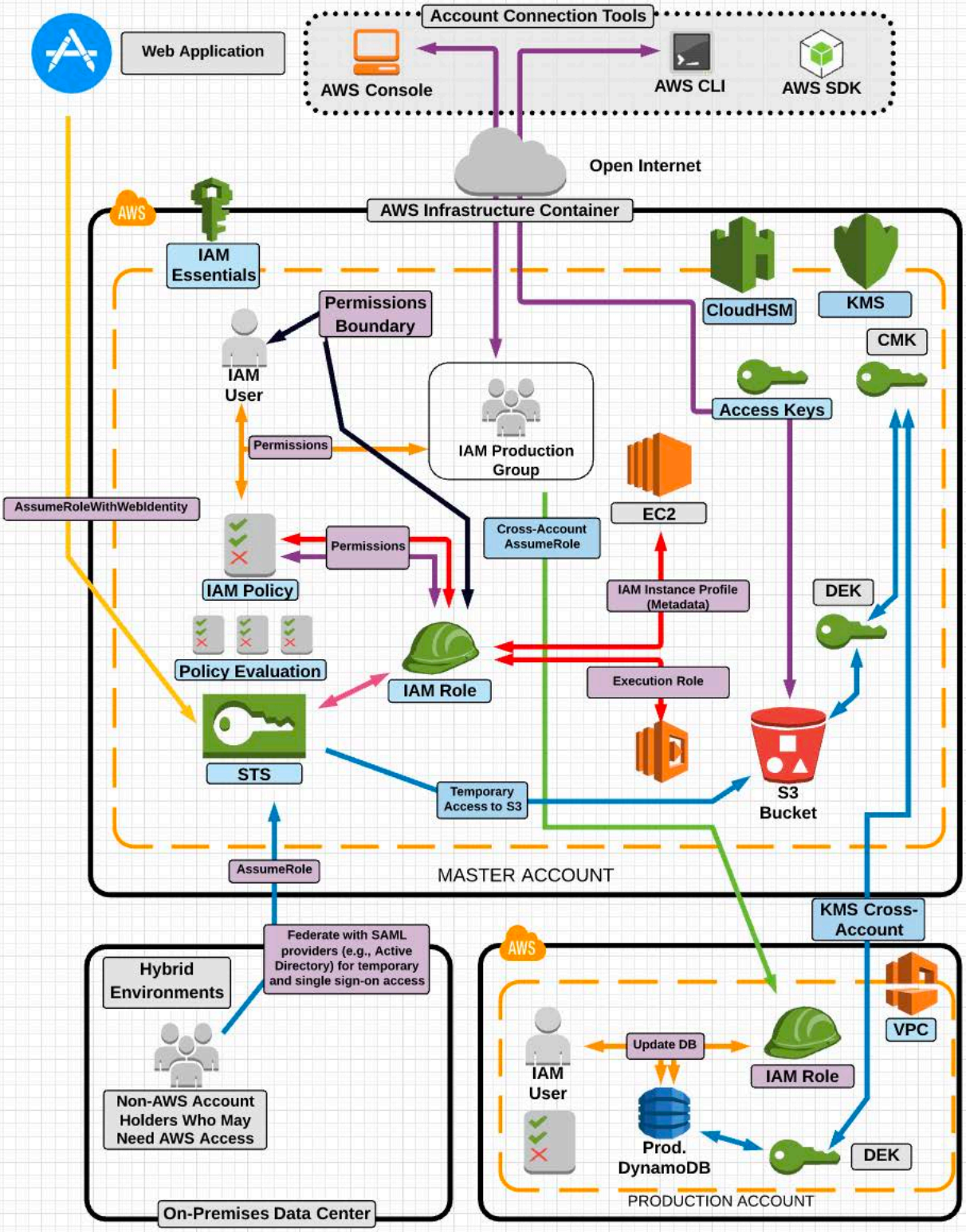
IAM roles provide permissions to the VPC Flow Logs service, either for CloudWatch log injection or for sending log files to an S3 bucket for storage.

IAM Role

VPC Flow Logs gather *metadata* for traffic flowing across a VPC. They don't log actual traffic, nor are they real-time. They collect data at one of three levels:

- VPC: All traffic to/from/inside the VPC
- Subnet: To/from/inside the subnet
- ENI: To/from the ENI


us-east-1
ap-southeast-2
AWS
VPC
VPC
Availability Zone A
Availability Zone B
Subnet A2
Subnet B2
NO HA
AWS HA
FULL HA
Site-to-site VPN has three components:
1) Virtual private gateway
2) Customer gateway(s)
3) VPN connection(s)
VPC Router
Route Table
172.16.0.0
172.16.1.0
172.16.2.0
VPN connections link the VGW and CGWs. They provision a tunnel endpoint in two AZs for high availability, set authentication, and define dynamic (BGP) or static routing. Dynamic VPNs learn remote CIDRs, while static VPNs must have them set.
Routes can be added manually or propagation enabled.
Virtual private gateways are logical gateway devices inside a VPC — can be used as route targets.
Virtual Private Gateway
AZ-A
AZ-B
AZ-A
AZ-B
Tunnel Endpoints
Tunnel1
Tunnel2
Tunnel1
Tunnel2
AWS Public Zone
S3 Bucket
Public Internet
VPN Connection
VPN Connection
Non-VPC Comms As Normal
Customer Gateway 1
3.11.11.11
Customer Gateway 2
3.22.22.22
Customer gateways are the physical customer routers *and* the records inside AWS for these devices. IP addresses and optionally ASNs are provided (allowing BGP).
On-Premises


Web Application
Account Connection Tools
AWS Console
AWS CLI
AWS SDK
Open Internet
AWS
AWS Infrastructure Container
IAM Essentials
Permissions Boundary
IAM User
Permissions
IAM Production Group
CloudHSM
KMS
CMK
Access Keys
AssumeRoleWithWebIdentity
EC2
Cross-Account AssumeRole
Permissions
IAM Policy
IAM Instance Profile (Metadata)
DEK
Policy Evaluation
IAM Role
Execution Role
STS
Temporary Access to S3
S3 Bucket
AssumeRole
MASTER ACCOUNT
KMS Cross-Account
Federate with SAML providers (e.g., Active Directory) for temporary and single sign-on access
Hybrid Environments
Non-AWS Account Holders Who May Need AWS Access
On-Premises Data Center
AWS
VPC
Update DB
IAM User
IAM Role
Prod. DynamoDB
DEK
PRODUCTION ACCOUNT

Identity & Access Management (IAM)

IAM Essentials

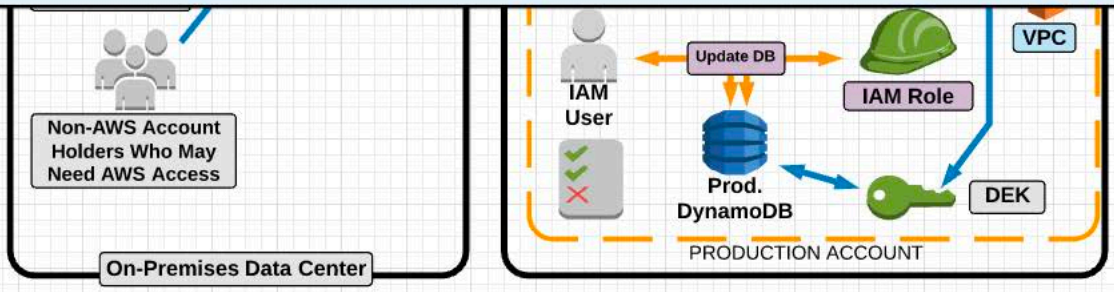
- **IAM (Identity & Access Management)** allows you to control access to AWS services and resources. It lets you manage identities and allocate permissions for different roles, users, or organizations. As a global service, the pool of identities is shared across all regions.

- IAM is used to manage:
  - Users
  - Groups
  - Roles
  - IAM Policies
  - Authentication Attributes: Usernames, Passwords, Access Keys, MFA, and Password Policies

- By default, any new IAM user you create in an AWS account has no permissions policies attached.
- All permissions policies have an implicit deny. If there is no **allow** in the policy, it is equivalent to an implicit **deny**.
- A **deny** always overrides any **allow**.

- For all users (except the root user), permissions must be given that grant access to AWS services. This is done through IAM policies.
- IAM provides identity services but also coordinates with STS to allow identity federation (the use of external identities) to access AWS resources.

- IAM Best Practices:
  - Delete your root access keys.
  - Activate MFA on your root account.
  - Create and use an IAM user with admin privileges instead of the root account.
  - Create individual IAM users.
  - Use groups to assign permissions.
  - Follow the principle of least privilege.
  - Apply an IAM password policy.



Identity & Access Management (IAM)

IAM Permissions and Policies

- A **policy** is a document that formally states one or more permissions.
- By default, all permissions are *implicitly denied*.
- An **explicit deny** always overrides an **explicit allow**.

- IAM provides pre-built policy templates to assign to users and groups. Examples include:
  - **Administrator Access**: Full access to *all* AWS resources
  - **Power User Access**: Admin Access, except it does not allow user/group management
  - **Read-Only Access**: Only view AWS resources (e.g., user can only view what is in an S3 bucket)

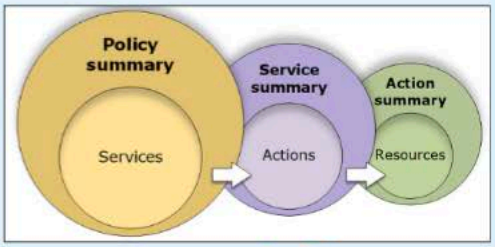
- You can also create custom IAM permissions policies using the **visual editor** or from scratch.
- More than one policy can be attached to a user or group at the same time.
- Policies cannot be directly attached to AWS resources (such as an EC2 instance). You must use roles for this.

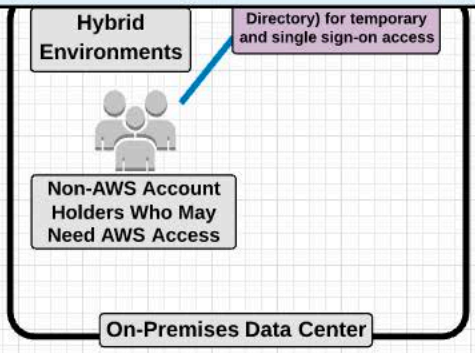
IAM Policy Example: EC2 Full Access

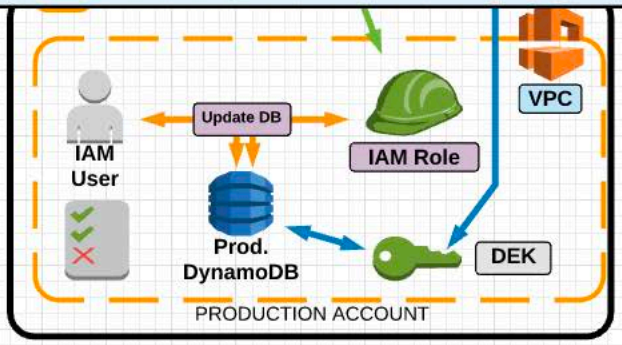
Service	Access level	Resource	Request condition
Allow (6 of 142 services) Show remaining 136			
Auto Scaling	Full access	All resources	None
CloudWatch	Full access	All resources	None
EC2	Full access	All resources	None
ELB	Full access	All resources	None
ELB v2	Full access	All resources	None
IAM	Limited: Write	All resources	Multiple

Boundaries — Boundaries that permissions can't exceed

Access Advisor — Shows what permissions exist and the last access time

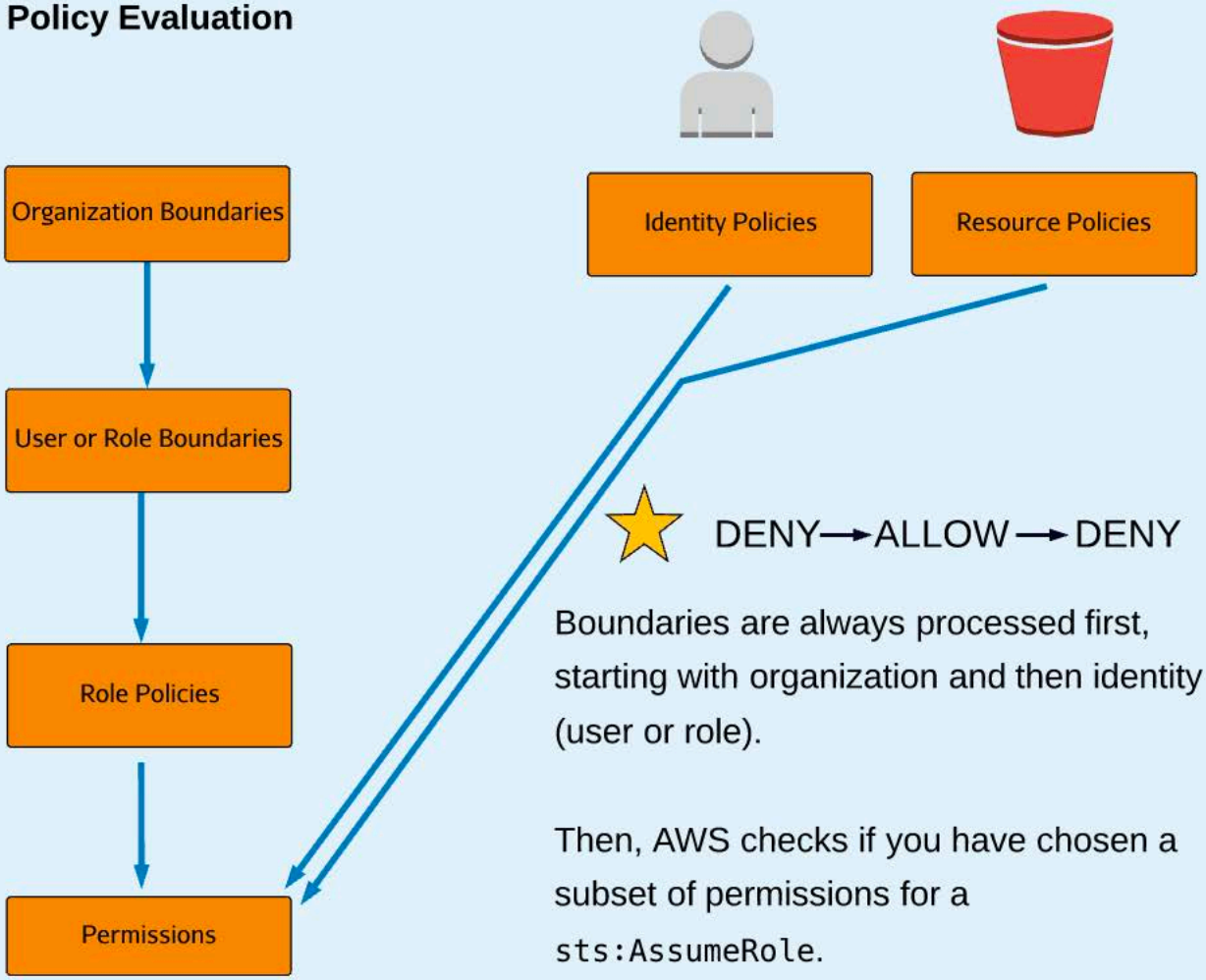






Identity & Access Management (IAM)

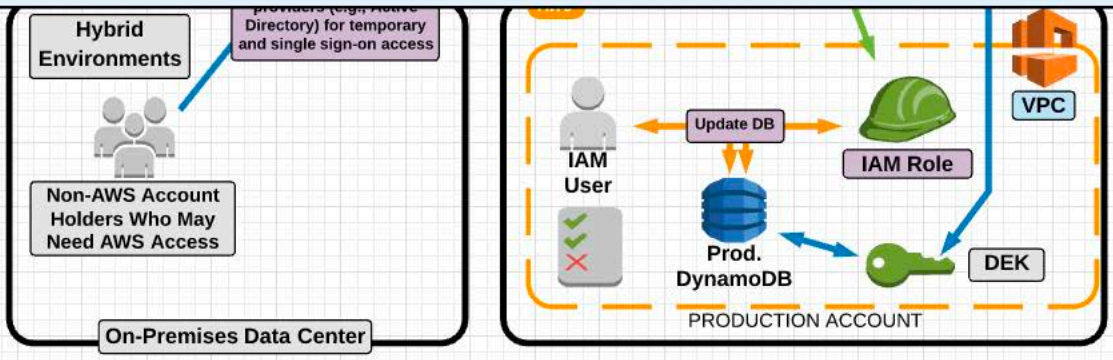
IAM Policy Evaluation



Boundaries are always processed first, starting with organization and then identity (user or role).

Then, AWS checks if you have chosen a subset of permissions for a `sts:AssumeRole`.

The final effective permissions are a merge of identity, resource, and ACL.



Identity & Access Management (IAM)

IAM Security Token Service (STS)

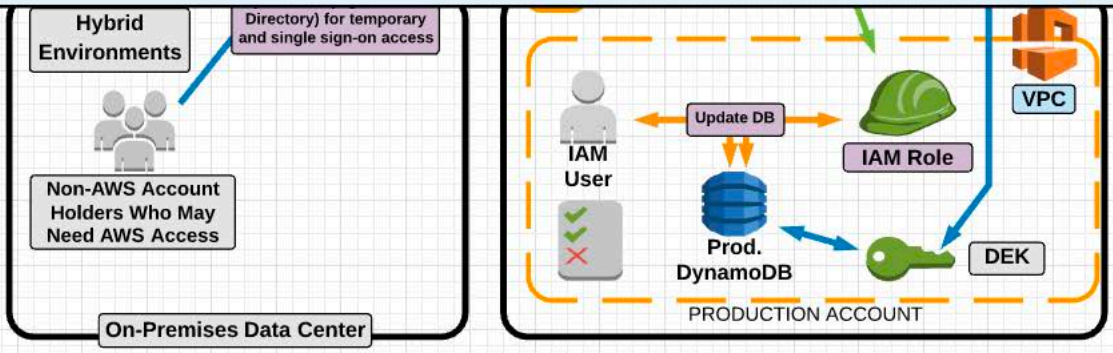
- STS allows you to create temporary security credentials that grant trusted users access to your AWS resources.
- These temporary credentials are for short-term use, with a configurable session duration that can be set anywhere between 15 minutes and 36 hours.
- Once expired, the credentials can no longer be used to access your AWS resources.
- An STS API call returns a credential object containing:
  - A session token
  - An access key ID
  - A secret access key
  - An expiration timestamp

When to use STS:

- Identity Federation
  - Enterprise identity federation (authenticate through your company's network).
    - STS supports SAML (Security Assertion Markup Language), which allows for the use of Microsoft Active Directory or your own solution.
  - Web identity federation (third-party identity providers like Facebook, Google, and Amazon).
- Roles for Cross-Account Access
  - Used for organizations that have more than one AWS account.
- Roles for Amazon EC2 (and Other AWS Services)
  - Allow an application running on an EC2 instance to access other AWS services without having to embed credentials.

For mobile applications, AWS recommends using Cognito rather than STS. Cognito provides additional mobile-specific functionality that makes the flow easier to manage.

AssumeRole | AssumeRoleWithWebIdentity | AssumeRoleWithSAML | Example

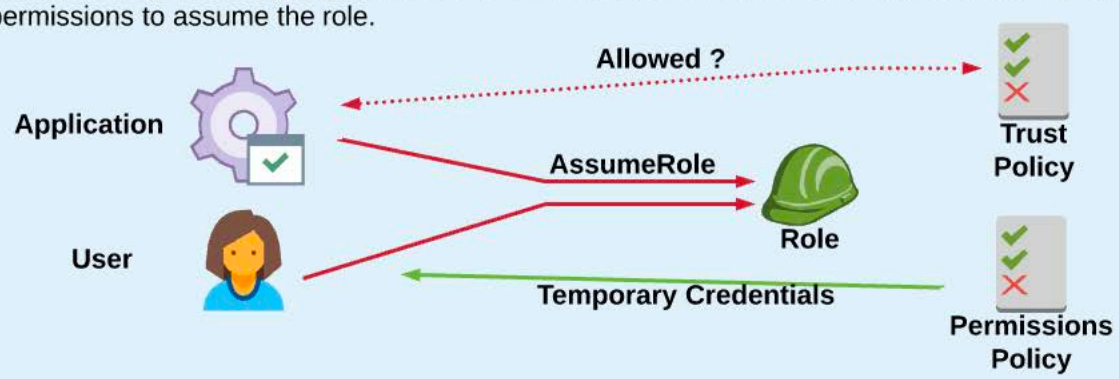


IAM Roles

Like an IAM user, an IAM role is an entity that can be granted permissions to interact with AWS resources. However, unlike IAM users, roles are designed to be assumed by anyone who needs to make use of their permissions.

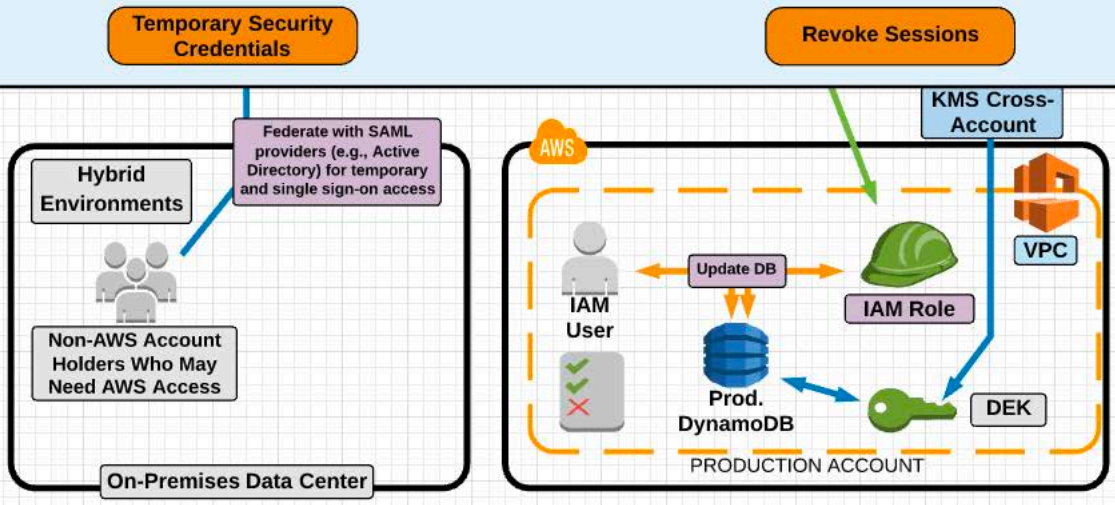
A role has two components: a **trust** policy, which defines the circumstances under which the role can be assumed, and a **permissions** policy, which defines the AWS access rights granted during that `AssumeRole`.

A role has no long-term access credentials (username, password, or key pairs) of its own. Instead, when the role is assumed, STS generates temporary credentials, which can last anywhere from a few minutes to several hours. Once the credentials expire, they are no longer valid. Temporary credentials can be renewed, even before expiration, and will be granted for as long as the requesting identity has permissions to assume the role.



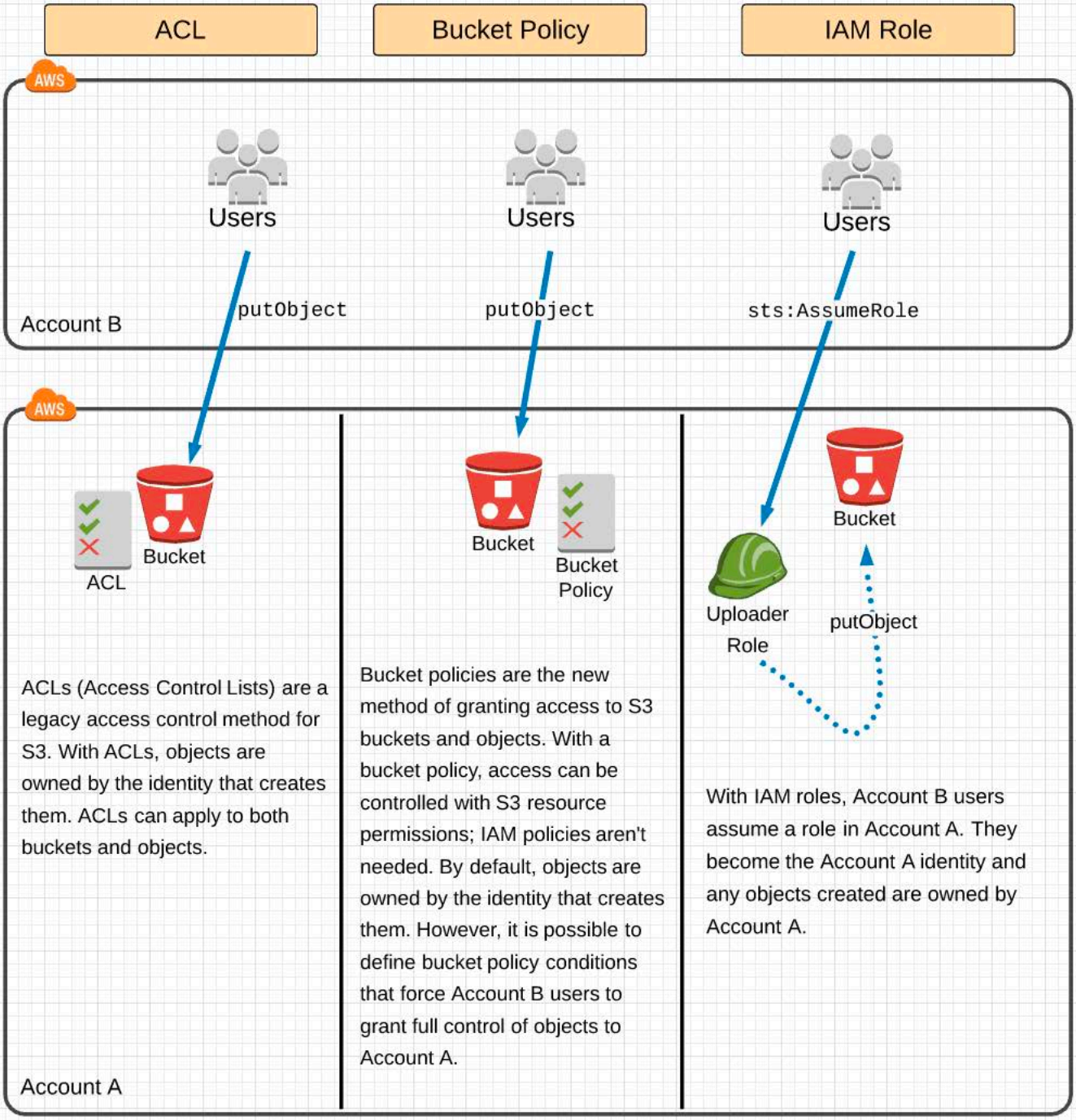
Roles are used extensively in AWS. You grant permissions to services via roles. For example, Lambda gains permissions via its execution role, EC2 is given permissions via an instance role, and many services that use CloudWatch for logging are provided those services via roles.

When an entity assumes a role, it becomes that role. If you assume a role in an external AWS account and then interact with that account, that account *owns* any resources you create. You *are* an entity in that account for as long as you've assumed the role.



Cross-Account Access to S3 Buckets and Objects

There are three main ways to provide access to your S3 buckets from external AWS accounts: IAM Roles, Bucket Policies, and Bucket ACLs.



CloudHSM and On-Premises HSM

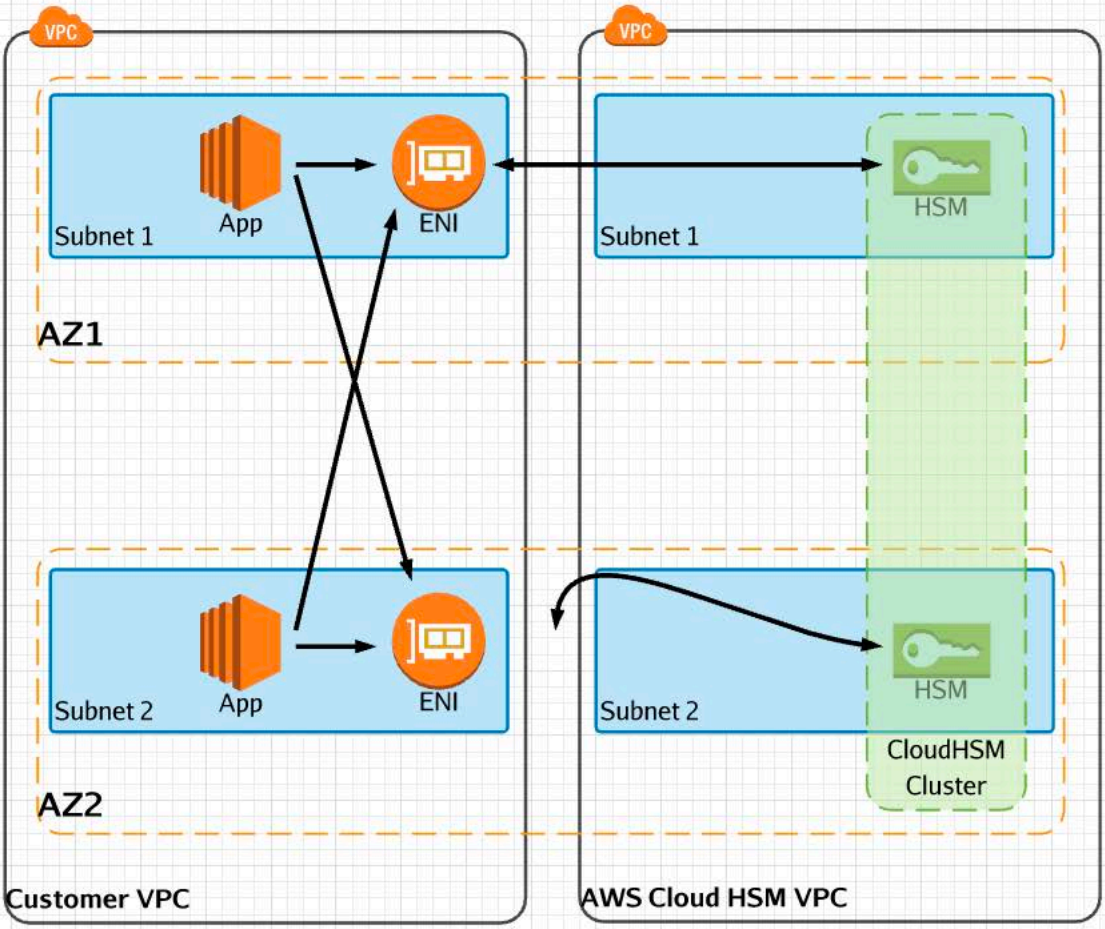
CloudHSM is a dedicated HSM that runs within your VPC, accessible only to you in a single-tenant architecture. AWS manages and maintains hardware but has ***no*** access to the cryptographic component. Interaction with CloudHSM is via industry-standard APIs, ***not* normal AWS APIs**.

- PKCS #11, Java Cryptography Extensions (JCE), Microsoft CryptoNG (CNG)

Keys can be transferred between CloudHSM and other hardware solutions (on-premises). Keys are shared between cluster members. *No* HA unless multiple HSMs are provisioned.

Applications can be ***outside*** the VPC — Direct Connect, peered, or VPN.

On-premises HSM is for if you ***really*** need to **control your own physical hardware.**



Identity & Access Management (IAM)

IAM Access Keys

Access keys are used to sign programmatic requests to AWS. They are required when making calls to AWS from:

- The AWS command line interface (CLI)
- Tools for Windows PowerShell
- AWS SDKs

Access Key Architecture

Access keys are made up of two parts: an **access key ID** and a **secret access key**.

Access Key ID: *AKIAIOSFODNN7EXAMPLE*
Secret Access Key: *wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY*

The access key ID is public. The secret access key is sensitive and should be stored securely. You will need both parts to interact with AWS from the CLI or API.

Important Facts about Access Key Pairs

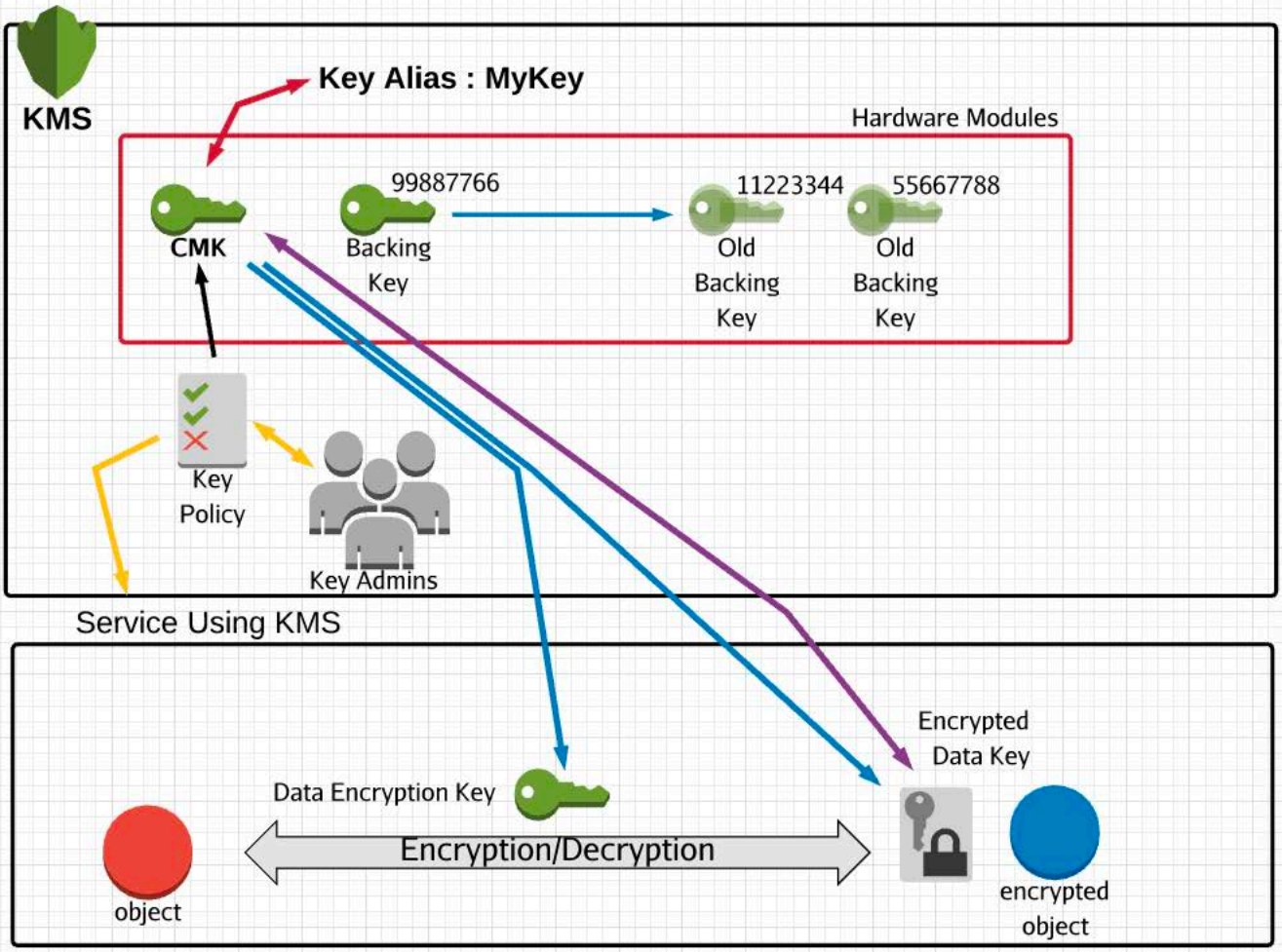
- Access keys are associated with IAM users, *not* groups or roles.
- They can be generated *only once*. The secret access key is *not* stored by AWS. If lost, the access key will need to be regenerated.
- AWS will *not* regenerate the same access key. Once an access key is lost, it cannot be recovered.
- Access keys are long-term credentials. They don't expire and can only be enabled, disabled, or deleted.
- IAM roles do not have associated access keys; they use short-term credentials.
- IAM users can only have two key pairs at a time. If you require new credentials and already have two key pairs, you must deactivate one of your current keys and generate a new one.
- *Never* create or store access keys on an EC2 instance or GitHub repository.
- Access keys grant access to whatever actions and resources the associated IAM user has access to.
- Access keys provide access to AWS from anywhere an API connection is possible.

Key Management Service (KMS)

KMS is a key management service that uses FIPS 140-2 compliant hardware modules to manage access to key material. It integrates fully with IAM and CloudTrail for permissions management and auditing functions. KMS can be optionally used with most AWS services that support encryption.

KMS manages customer master keys (CMKs), with which it can encrypt and decrypt small amounts of data. It can also use CMKs to generate data encryption keys (DEKs), which AWS services can use to encrypt and decrypt data.

All interaction with KMS is via AWS services or AWS APIs. KMS does not support industry standard key management APIs but integrates well with AWS services.



Identity & Access Management (IAM)

IAM Access Keys

Access keys are used to sign programmatic requests to AWS. They are required when making calls to AWS from:

- The AWS command line interface (CLI)
- Tools for Windows PowerShell
- AWS SDKs

Access Key Architecture

Access keys are made up of two parts: an **access key ID** and a **secret access key**.

Access Key ID: *AKIAIOSFODNN7EXAMPLE*
Secret Access Key: *wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY*

The access key ID is public. The secret access key is sensitive and should be stored securely. You will need both parts to interact with AWS from the CLI or API.

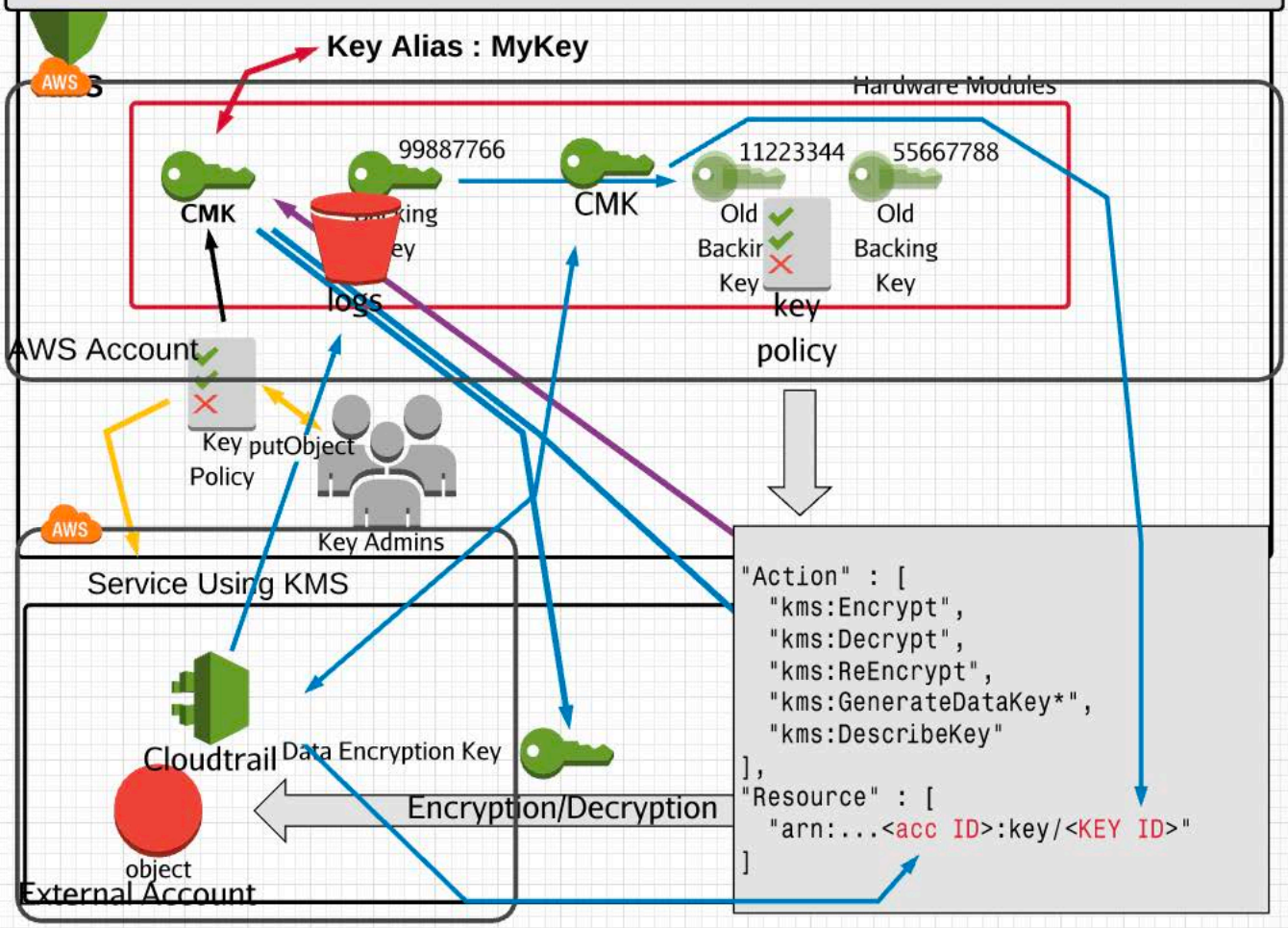
Important Facts about Access Key Pairs

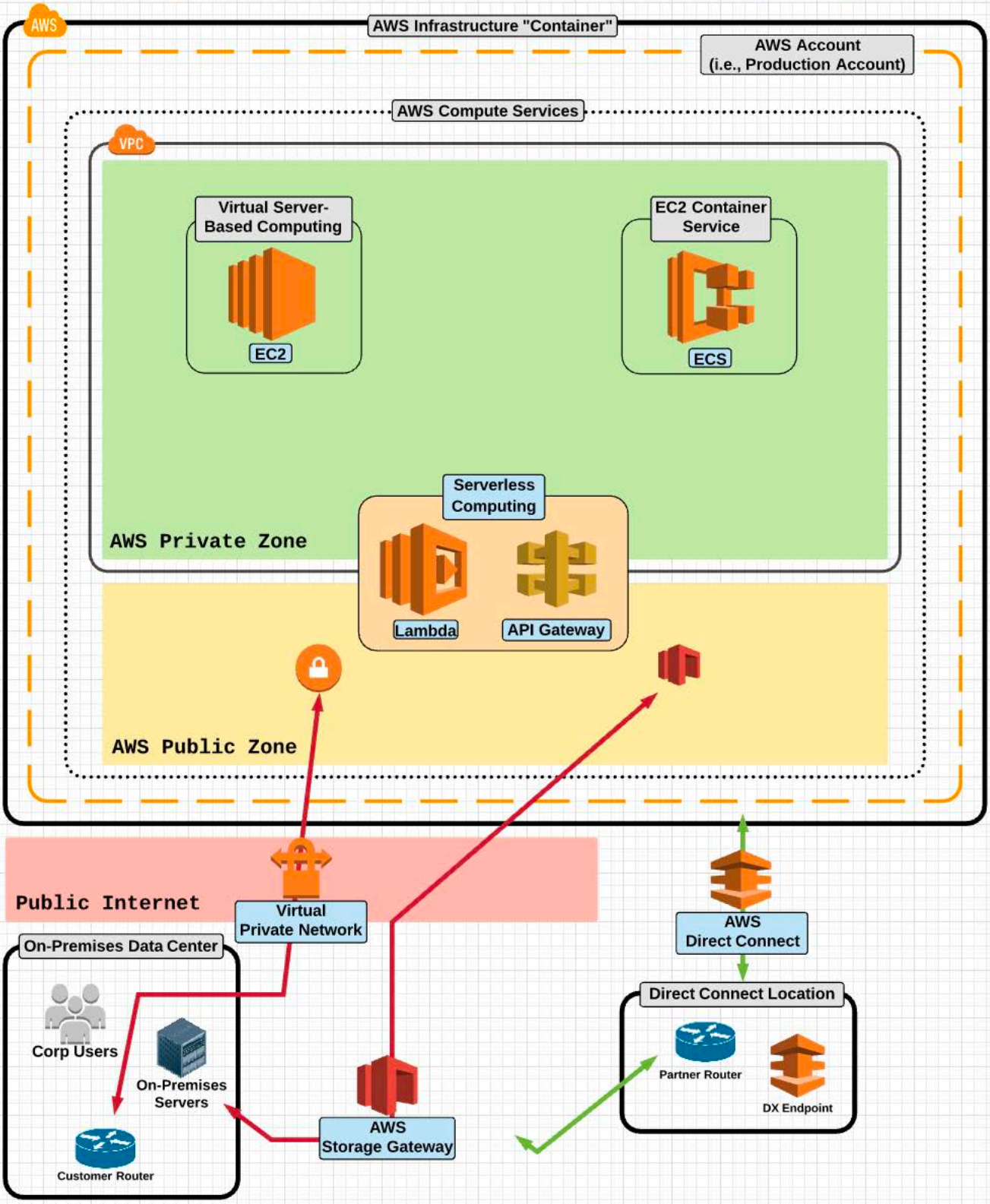
- Access keys are associated with IAM users, *not* groups or roles.
- They can be generated *only once*. The secret access key is *not* stored by AWS. If lost, the access key will need to be regenerated.
- AWS will *not* regenerate the same access key. Once an access key is lost, it cannot be recovered.
- Access keys are long-term credentials. They don't expire and can only be enabled, disabled, or deleted.
- IAM roles do not have associated access keys; they use short-term credentials.
- IAM users can only have two key pairs at a time. If you require new credentials and already have two key pairs, you must deactivate one of your current keys and generate a new one.
- *Never* create or store access keys on an EC2 instance or GitHub repository.
- Access keys grant access to whatever actions and resources the associated IAM user has access to.
- Access keys provide access to AWS from anywhere an API connection is possible.

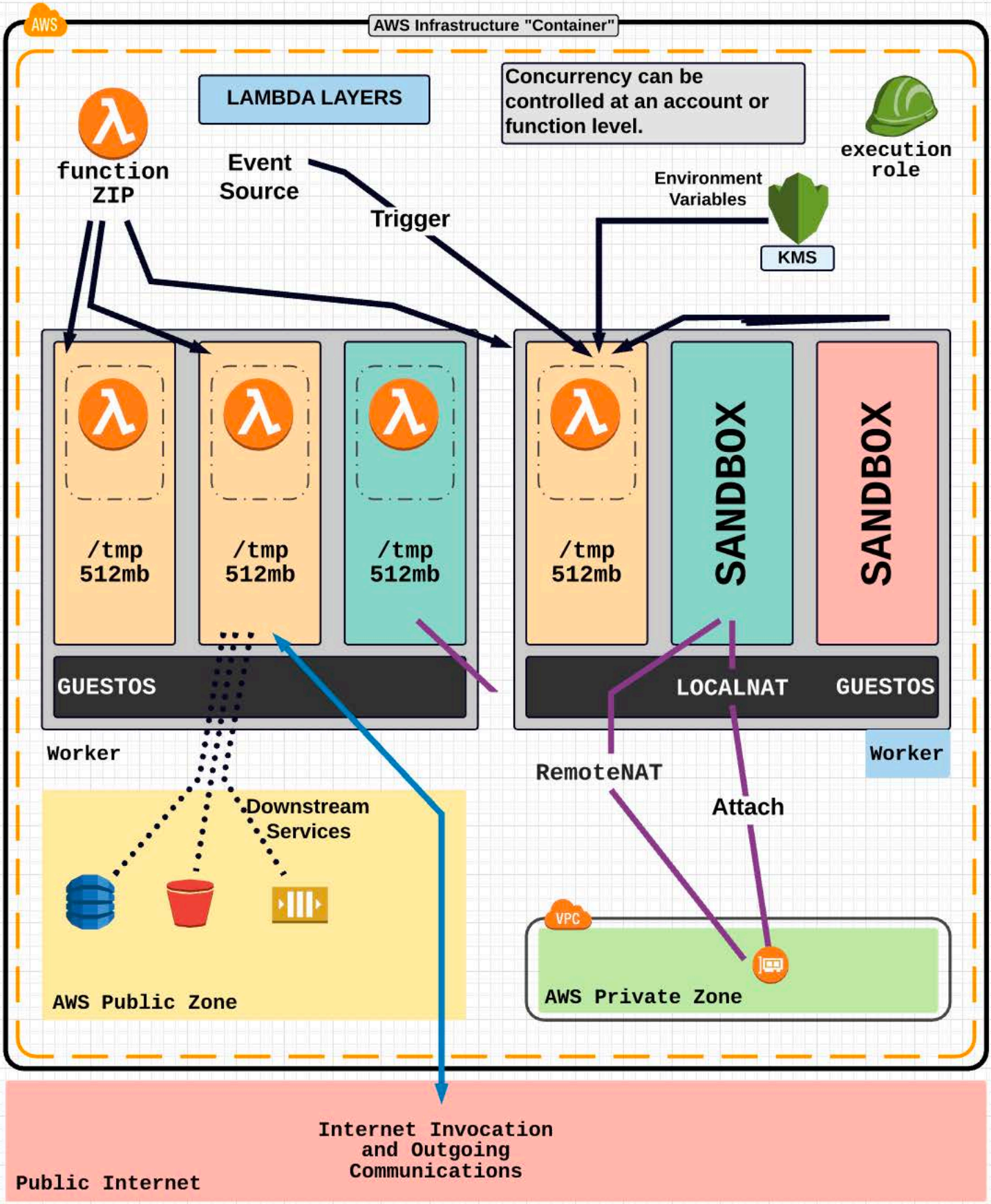
KMS in a Multi-Account Configuration

CMKs can be configured to allow other accounts to *use* them. The key won't *appear* in the external account, but if it is configured using a key policy, that account can interact with the key for cryptographic functions.

Remember: Key *usage* and key *admin* are not the same thing!




AWS
AWS Infrastructure "Container"
AWS Account
(i.e., Production Account)
AWS Compute Services
VPC
Virtual Server-Based Computing
EC2
EC2 Container Service
ECS
Serverless Computing
Lambda
API Gateway
AWS Private Zone
AWS Public Zone
Public Internet
Virtual Private Network
AWS Direct Connect
Direct Connect Location
Partner Router
DX Endpoint
On-Premises Data Center
Corp Users
On-Premises Servers
Customer Router
AWS Storage Gateway


AWS
AWS Infrastructure "Container"
LAMBDA LAYERS
Concurrency can be controlled at an account or function level.
function ZIP
Event Source
Trigger
execution role
Environment Variables
KMS
/tmp 512mb
/tmp 512mb
/tmp 512mb
/tmp 512mb
SANDBOX
SANDBOX
GUESTOS
LOCALNAT
GUESTOS
Worker
Worker
RemoteNAT
Attach
Downstream Services
AWS Public Zone
VPC
AWS Private Zone
Internet Invocation and Outgoing Communications
Public Internet

Lambda Layers

X

Lambda layers provide two additional capabilities to Lambda functions and the engineers using them.

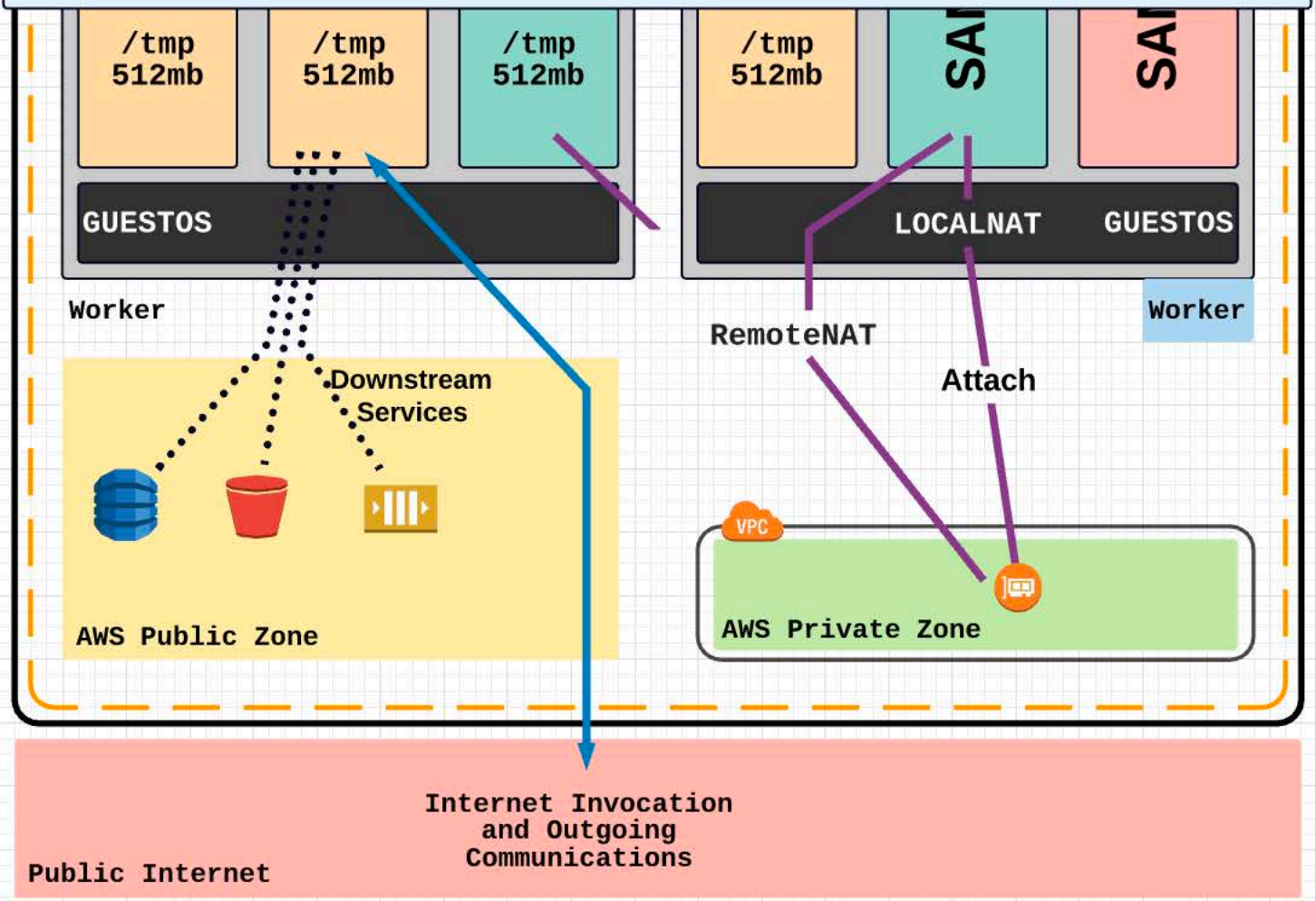
- They can be used to add additional runtime support — runtime support for any language can be packaged into a layer and shared with any AWS account or publicly.
- Layers can also be used to store commonly used libraries and other software dependencies, allowing them to be removed from deployment packages.

Layers are immutable. Once created, they cannot be changed — new versions can be created, but existing versions are static.

Up to five layers can be used with a Lambda function.

Layers are extracted into the */opt* folder of the execution environment.

Total size of all layers cannot exceed 250 MB.



Lambda Worker

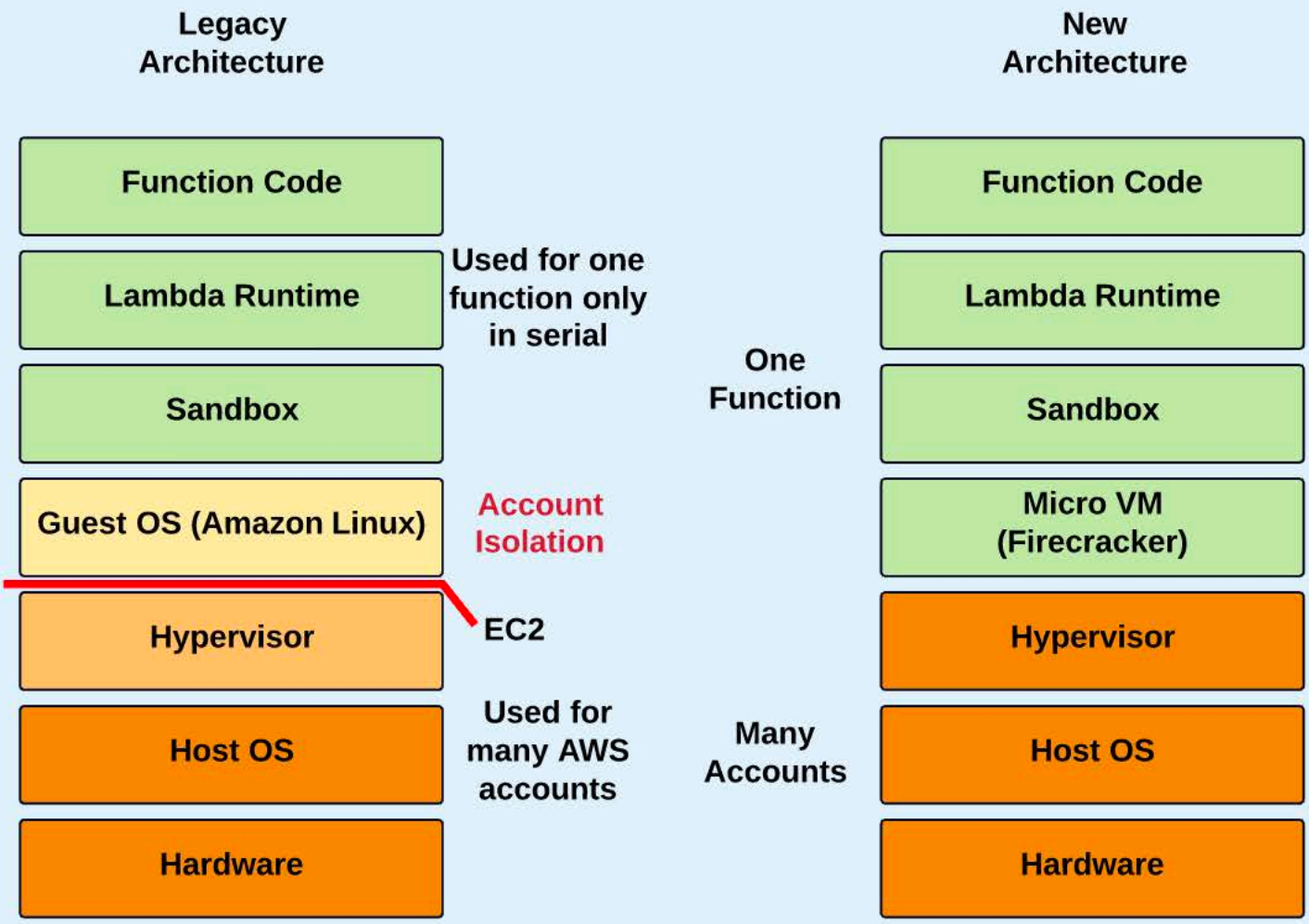
X

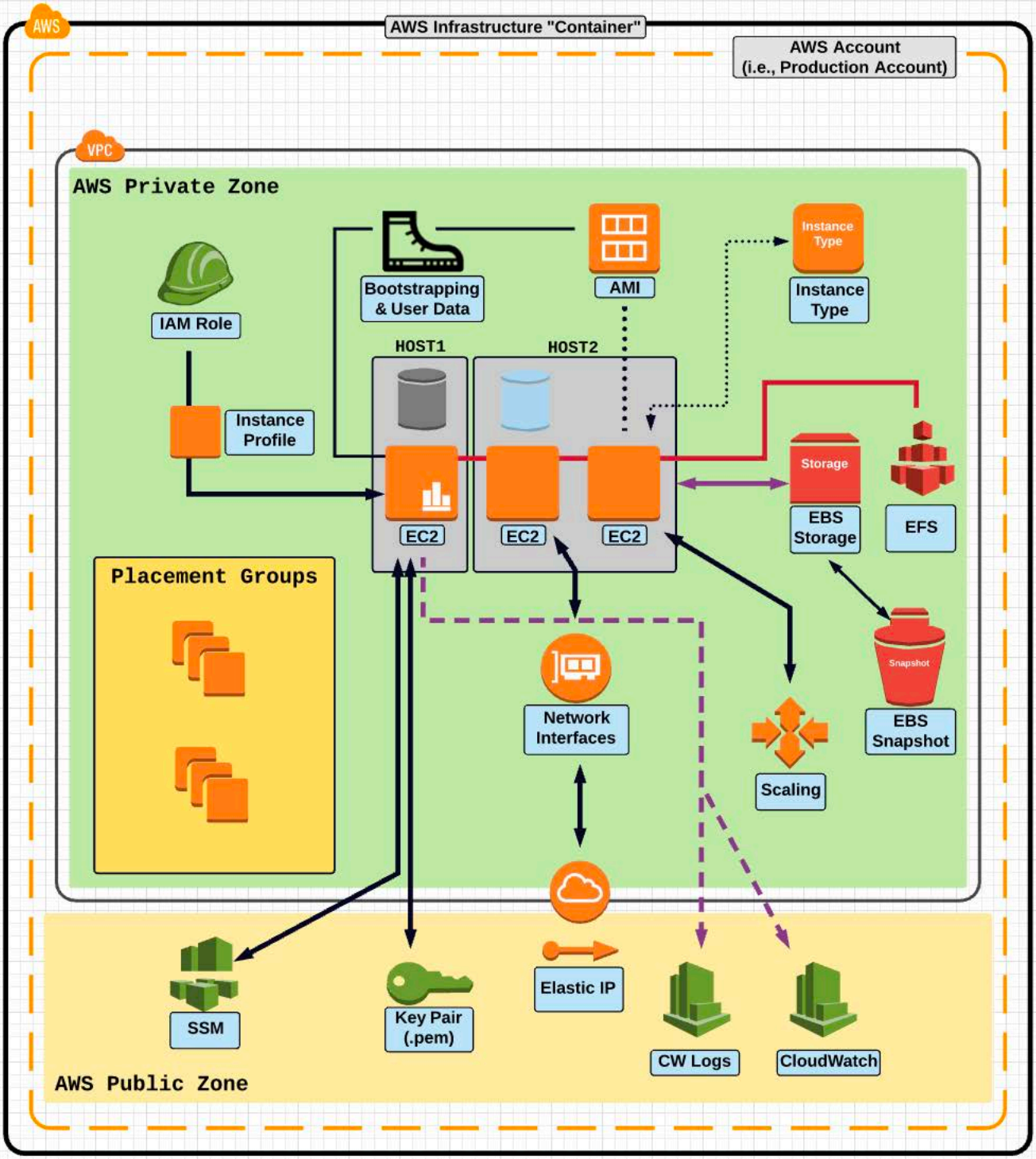
A Lambda worker is a physical compute "unit" that provides capacity to execute *sandboxes*, which are the environments Lambda functions operate within. Sandboxes are specific to a function and can be used for one function invocation at a time, but they can be reused.

Sandboxes, therefore, can be "cold" or "warm" (reused). Provisioning a sandbox takes time, so ideally aim for sandbox reuse as much as possible. Sandboxes are configured to have resources allocated (CPU and memory) influencing cost and performance.

Sandboxes are never used across different functions. Sandboxes operate on a guest operating system, which can run hundreds or thousands of sandboxes, but guest operating systems are isolated to a specific AWS account.

Hardware and host OS are shared across many AWS accounts, and the hypervisor provides guest OS isolation to a specific AWS account.





Public Internet

AWS

AWS Infrastructure "Container"

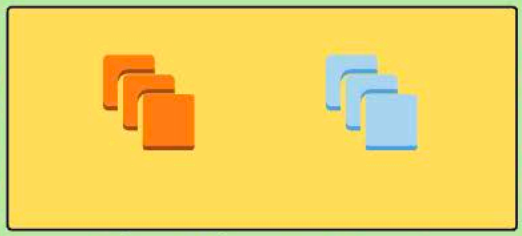
AWS Account
(i.e., Production Account)

AWS Private Zone

VPC

AZ-A

AZ-B



Cluster Placement Group

Cluster placement groups exist in a single AZ and are created to allow high speed and low latency connectivity between instances in the group.

Cluster placement groups reserve physical capacity, keeping instances in physical vicinity. The chances of capacity-related failure increases if different types of instances are used or instances are added at separate times.

Partition Placement Group

Designed to spread large infrastructure sets across infrastructure in isolated fault domains. Used when you need to ensure HA but need physical location control.

Spread Placement Group

Designed for a small number of critical components where you *need* to ensure separation. A spread placement group can be single-AZ or multi-AZ.

Availability Zone

Availability Zone

EC2 Instance Profiles and Instance Roles (IAM Role)

IAM roles provide short-term or temporary access credentials within AWS when used with the sts:AssumeRole API call. Identities within AWS can be granted access to assume roles using that roles TRUST policy.

Instance profiles allow a role to be assigned to a single EC2 instance and for applications running on that instance to assume a role while being abstracted away from any AWS identity. Applications running on an EC2 instance are not "AWS identities," and so it's the instance profile which bridges that gap.

IAM role credentials can be obtained from the EC2 instance metadata if a role is attached via an instance profile:

http://169.254.169.254/latest/meta-data/iam/security-credentials/ROLENAME

The credentials returned take the form of a JSON structure:

```
{
  "Code" : "Success",
  "LastUpdated" : "2012-04-26T16:39:16Z",
  "Type" : "AWS-HMAC",
  "AccessKeyId" : "ASIAIOSFODNN7EXAMPLE",
  "SecretAccessKey" : "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
  "Token" : "token",
  "Expiration" : "2017-05-17T15:09:54Z"
}
```

Credentials are automatically renewed when expired using calls to the instance metadata. Explicitly defined credentials take priority for the CLI tools if present on EC2 instances.

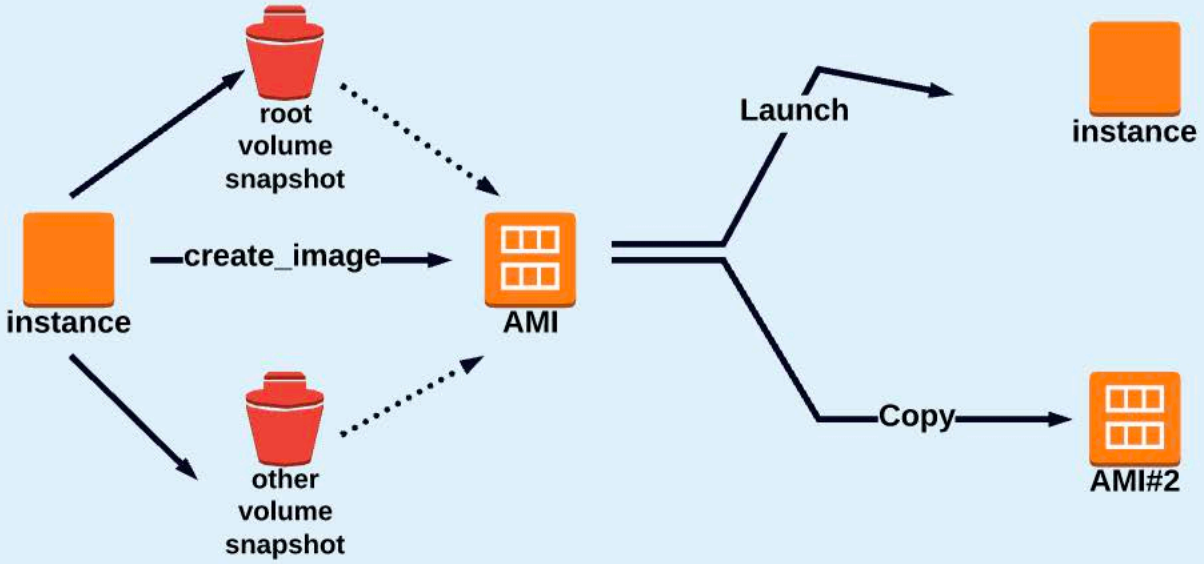
Only a single instance profile can be associated with an EC2 instance, containing a single EC2 role. If associating a role via the console, the instance profile is automatically created and associated. If using APIs, the CLI ,or CloudFormation, the two steps are distinct and must be done explicitly.

Roles (via profiles) can be assigned at the time of creating the instance or afterwards via the console, CLI, or APIs. All applications running on the instance share the role credentials — it's not possible to be more granular.

Amazon Machine Images (AMIs)

X

AMIs are objects containing all the information required to launch an instance, the owner of the AMI, launch permissions (public, explicit, and/or implicit), the architecture and operating system, and a block device mapping of all volumes required.

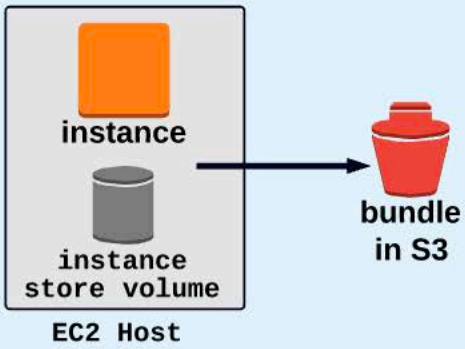


AMIs contain mappings to any volumes, and these mappings reference EBS volume snapshots in the same region.

AMIs are regional based but can be copied between regions, which also copies any volume snapshots.

By default, an AMI has an implicit permission allowing use by the creator. An AMI can be set to public, or accounts can be granted explicit rights to use the AMI and any associated volume snapshots.

AMIs can be sold. Users of those AMIs pay the base EC2 charges for the instance they create from the AMI, in addition to charges set by the creator.



When creating AMIs from instances with instance store root volumes, the process is slightly different — a bundle is created and stored on S3 containing ALL the root vol data.

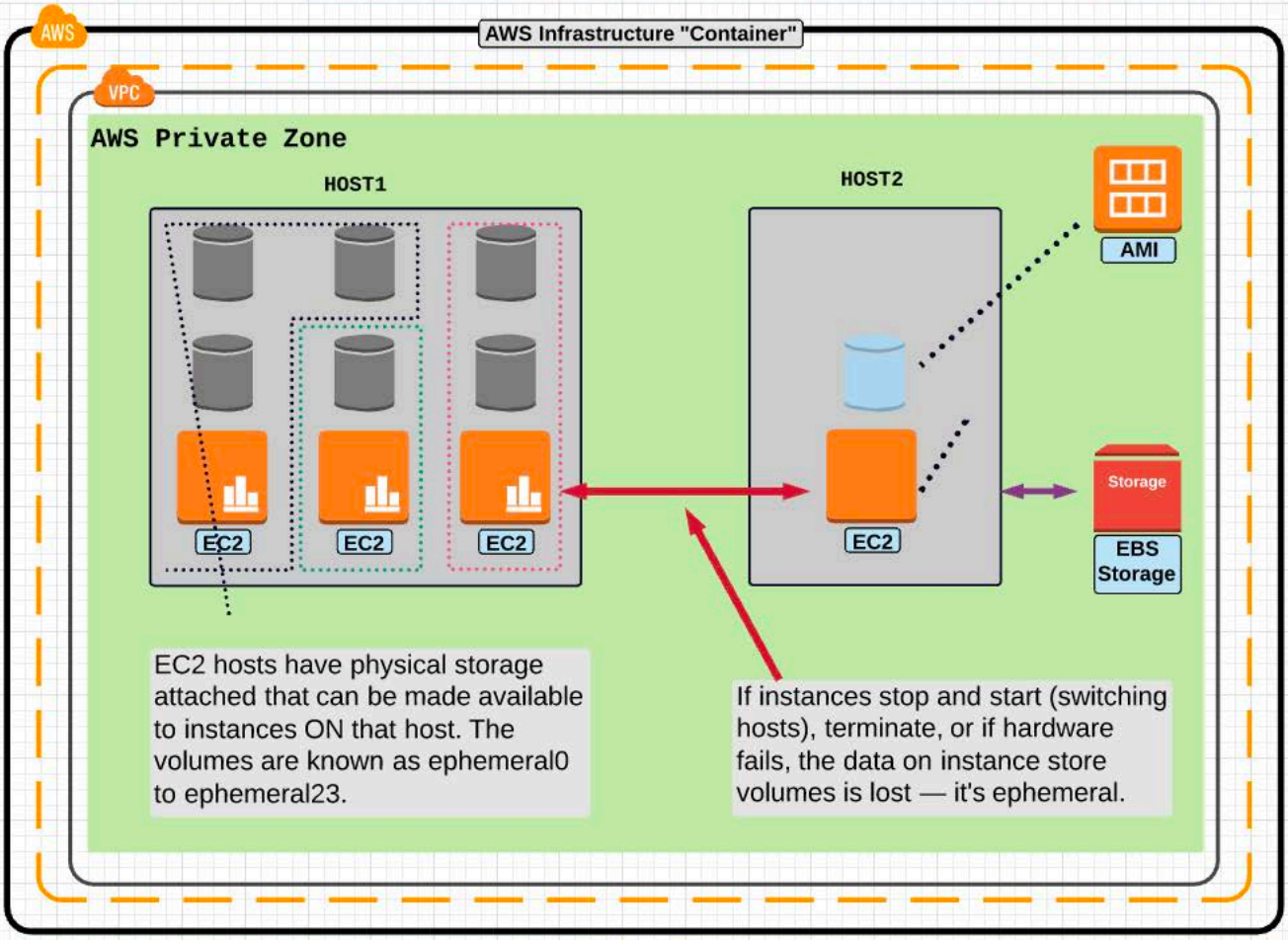
X

	EBS only	NVMe EBS	Instance store	Placement group	Enhanced networking
A1	Yes	Yes	No	Yes	ENA
C4	Yes	No	No	Yes	Intel 82599 VF
C5	Yes	Yes	No	Yes	ENA
C5d	No	Yes	NVMe *	Yes	ENA
C5n	Yes	Yes	No	Yes	ENA
D2	No	No	HDD	Yes	Intel 82599 VF
F1	No	No	NVMe *	Yes	ENA
G3	Yes	No	No	Yes	ENA
H1	No	No	HDD	Yes	ENA
I3	No	No	NVMe *	Yes	ENA
M4	Yes	No	No	Yes	`m4.16xlarge`: ENA All other sizes: Intel 82599 VF
M5	Yes	Yes	No	Yes	ENA
M5a	Yes	Yes	No	Yes	ENA
M5d	No	Yes	NVMe *	Yes	ENA
P2	Yes	No	No	Yes	ENA
P3	`p3dn.24xlarge`: No All other sizes: Yes	`p3dn.24xlarge`: Yes All other sizes: No	`p3dn.24xlarge`: NVMe *	Yes	ENA
R4	Yes	No	No	Yes	ENA
R5	Yes	Yes	No	Yes	ENA
R5a	Yes	Yes	No	Yes	ENA
R5d	No	Yes	NVMe *	Yes	ENA
T2	Yes	No	No	No	No
T3	Yes	Yes	No	No	ENA
u-xtb1.metal	Yes	Yes	No	No	ENA
X1	No	No	SSD	Yes	ENA
X1e	No	No	SSD	Yes	ENA
z1d	No	Yes	NVMe *	Yes	ENA

Instance Types: General Purpose, Compute Optimized, Memory Optimized, Storage Optimized, Accelerated Computing, Burstable

Accelerated Computing: GPU and **FPGA**

NITRO, NITRO, NITRO — if in doubt, the preference is Nitro.

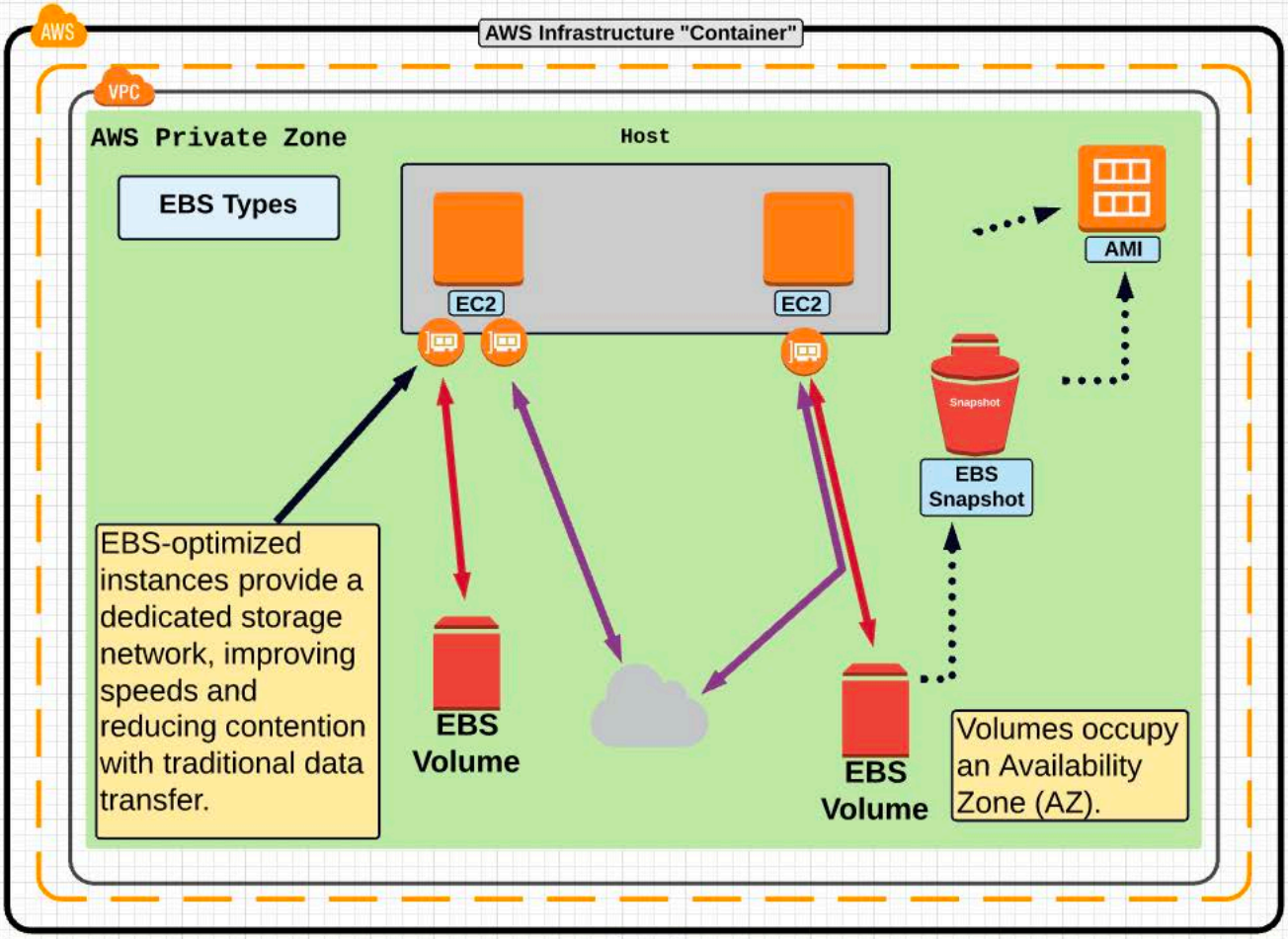


Ideal Patterns

- **Temporary Storage:** For any non-persistent storage needs, instance store volumes are included in the instance cost.
- **High I/O & Throughput:** Instance store offers direct attached storage. Depending on instance type, this offers the highest level of performance available for AWS storage.

Anti-Patterns

- **Shared Storage:** Volumes are attached to one instance only and can't be shared across instances.
- **Persistence:** Volumes will be removed from instances if that instance moves between hosts (stop/start).
- **Durability/Resilience:** No support for snapshots and volumes are single physical disks, so they can and do fail.
- **Elasticity:** Storage is directly tied to physical devices and can't easily be scaled.

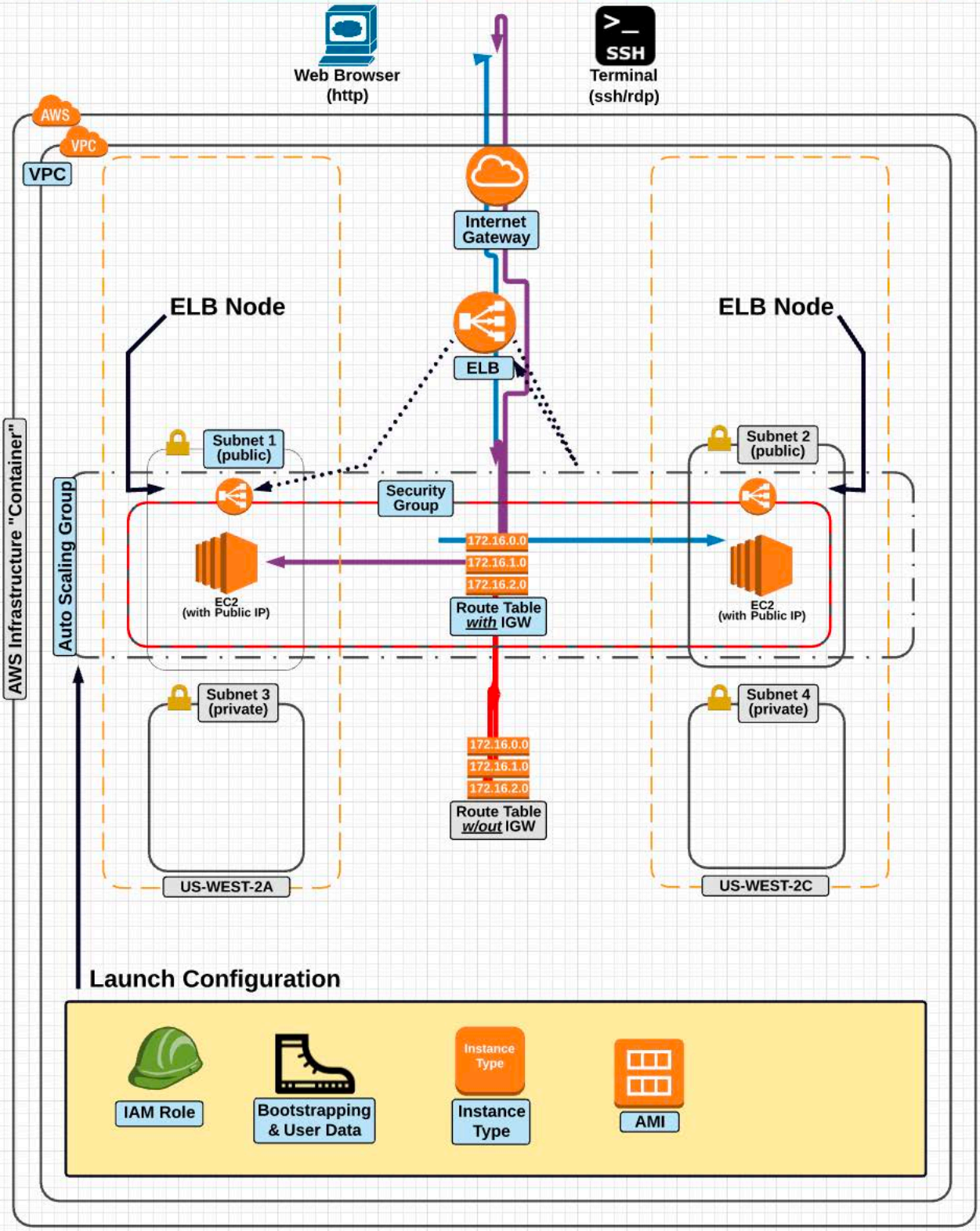


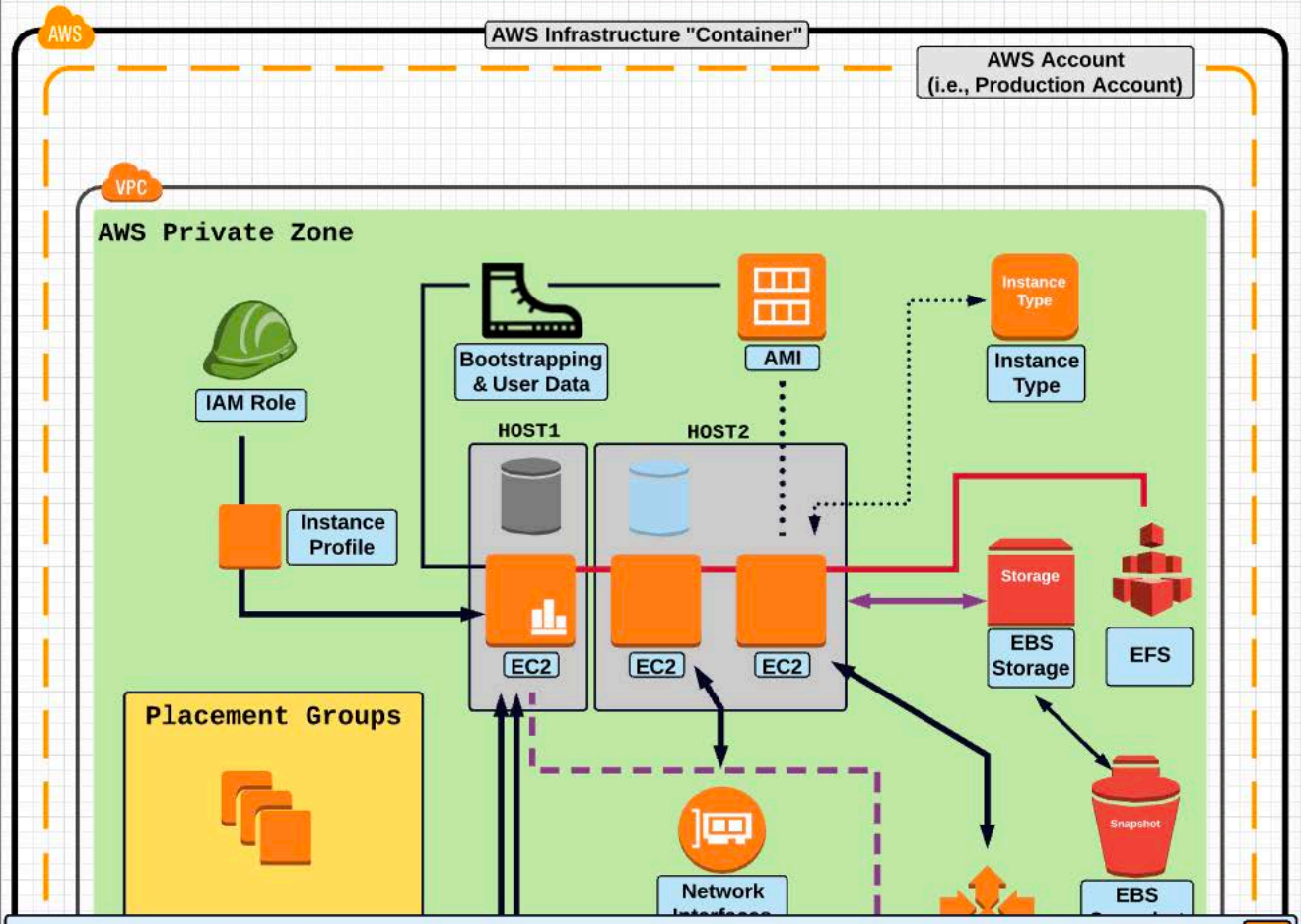
Ideal Patterns

- **Persistence:** A requirement for data to exist beyond the lifecycle of a given instance and able to tolerate reboots/instance failure.
- **Durability:** Data that requires snapshots or resilience. Note: EBS is not the most durable storage service.
- **Elasticity:** Requirements for variable or changing performance demands over time.
- **Provisioned Performance:** Certain EBS volumes can provide guaranteed performance levels.

Anti-Patterns

- **Temporary Storage:** Instance store would be more suited to this requirement. EBS provides persistence and has extra cost.
- **Static Content Distribution:** S3 or S3 + CloudFront is more ideal for object-based distribution.
- **Shared Access:** EBS volumes are attached to a single instance as block storage. They cannot be shared between instances.
- **High Durability:** EBS cannot provide super high 99.5-99.9%+ durability.


Web Browser
(http)
Terminal
(ssh/rdp)
AWS
VPC
VPC
Internet Gateway
ELB Node
ELB Node
ELB
Subnet 1
(public)
Subnet 2
(public)
Security Group
Auto Scaling Group
AWS Infrastructure "Container"
172.16.0.0
172.16.1.0
172.16.2.0
Route Table
with IGW
EC2
(with Public IP)
EC2
(with Public IP)
Subnet 3
(private)
Subnet 4
(private)
172.16.0.0
172.16.1.0
172.16.2.0
Route Table
w/out IGW
US-WEST-2A
US-WEST-2C
Launch Configuration
IAM Role
Bootstrapping
& User Data
Instance Type
Instance
Type
AMI

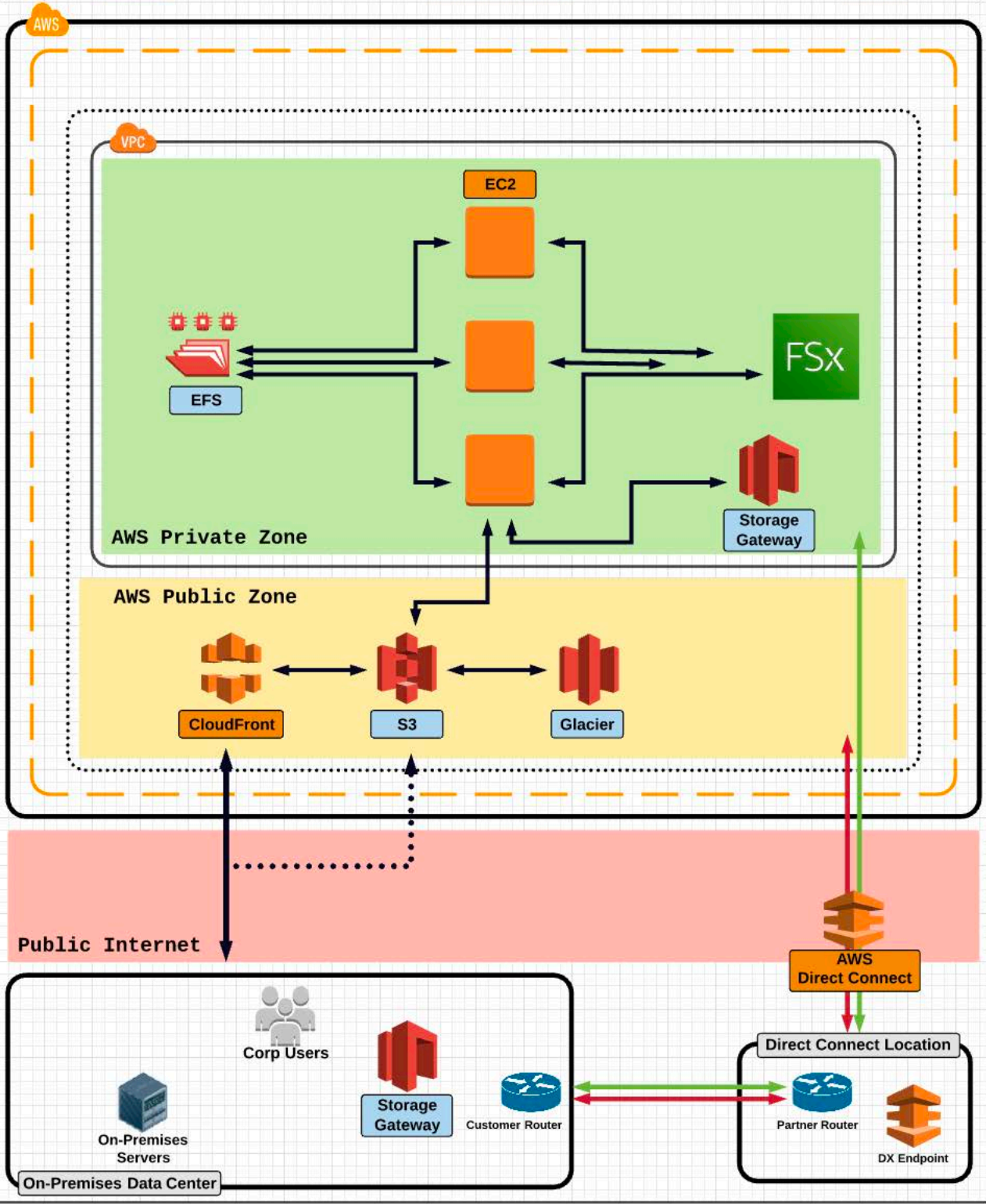


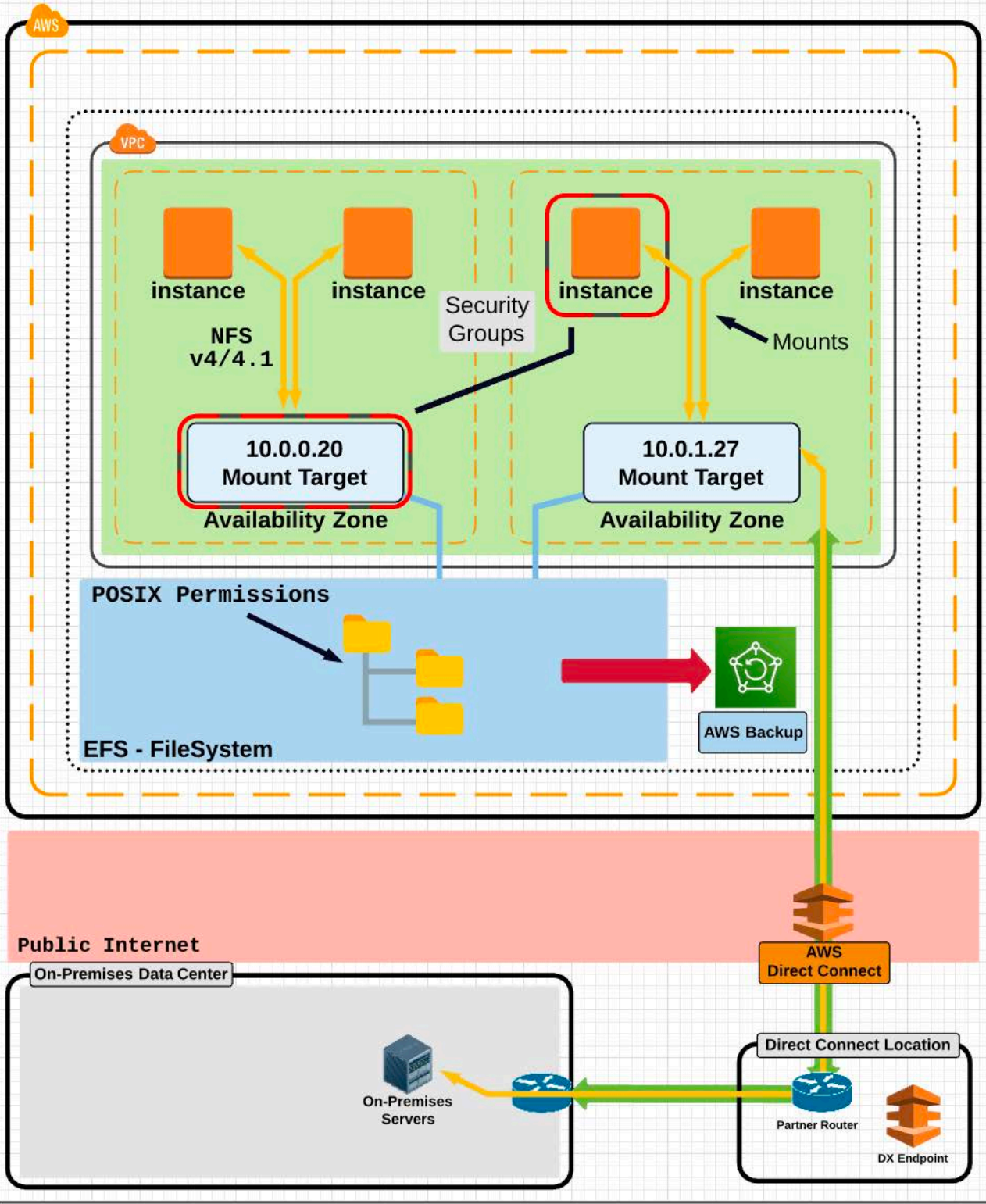
EBS Snapshots

X

Snapshots are point-in-time backups of an EBS volume that are stored in S3. An initial snapshot contains an exact copy of the data on the EBS volume and can take some time to complete.

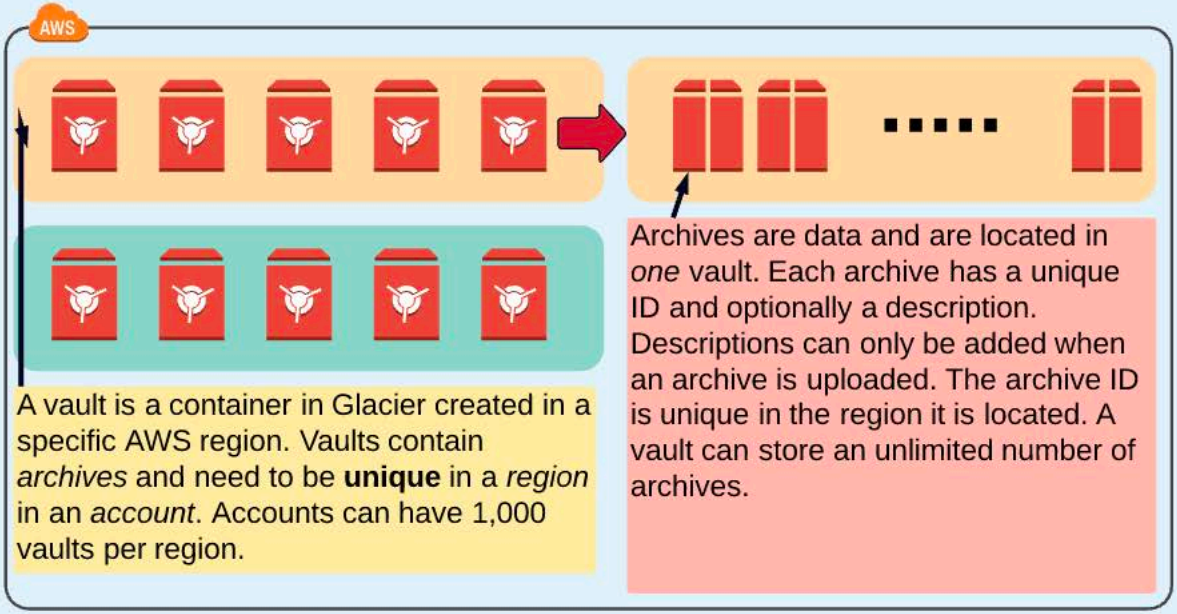
- Snapshots are incremental in nature.
- A snapshot only stores the changes since the most recent snapshot, thereby reducing costs (by only having to pay for storage for the “incremental changes” between snapshots).
- When deleted, blocks required to restore the other snapshots are retained.
- Snapshots can be used to create EBS volumes or used when creating AMIs.
- Snapshots are point-in-time and are known as crash-consistent. Activity should be frozen on highly transactional volumes where consistency is required.
- Ideally, for root volumes the instance should be in a ***stopped*** state.
- Where EBS is an AZ-scoped service, snapshots are in S3 and are region resilient.


AWS
VPC
EC2
EFS
FSx
Storage Gateway
AWS Private Zone
AWS Public Zone
CloudFront
S3
Glacier
Public Internet
AWS Direct Connect
Corp Users
Storage Gateway
Customer Router
On-Premises Servers
On-Premises Data Center
Direct Connect Location
Partner Router
DX Endpoint


AWS
VPC
instance
instance
instance
instance
Security Groups
NFS v4/4.1
Mounts
10.0.0.20 Mount Target
Availability Zone
10.0.1.27 Mount Target
Availability Zone
POSIX Permissions
EFS - FileSystem
AWS Backup
Public Internet
AWS Direct Connect
On-Premises Data Center
On-Premises Servers
Direct Connect Location
Partner Router
DX Endpoint

Glacier

The name *Glacier* is often used to refer to both the product S3 Glacier as well as the storage classes based on Glacier available within S3. S3 Glacier is an isolated product used for long-term archival storage. It's not a product you use directly. Beyond simple operations, it can only be interacted with from the command line or APIs.



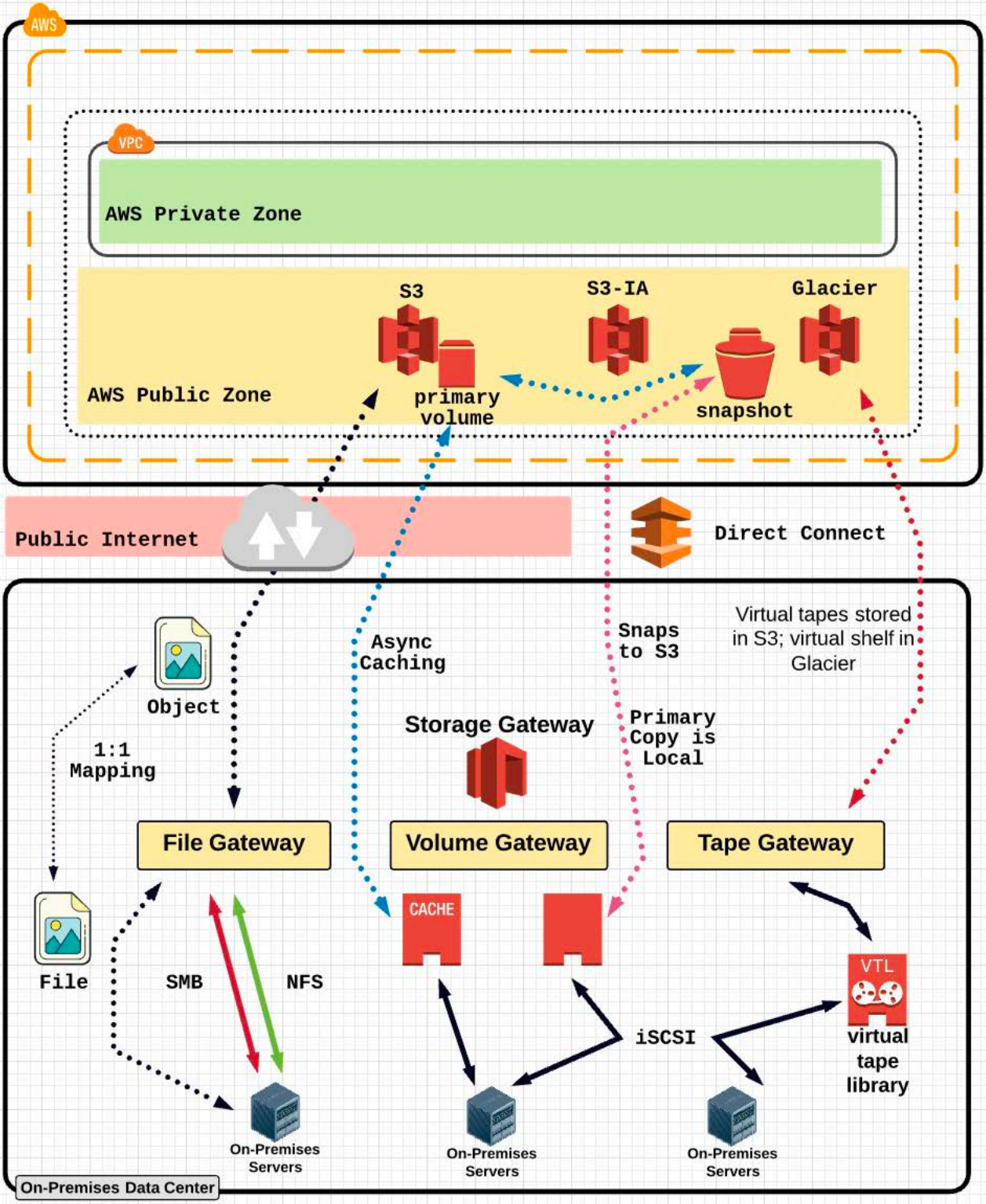
All vault operations are region specific — vaults are created in a region, and listing vaults returns vaults in that region.

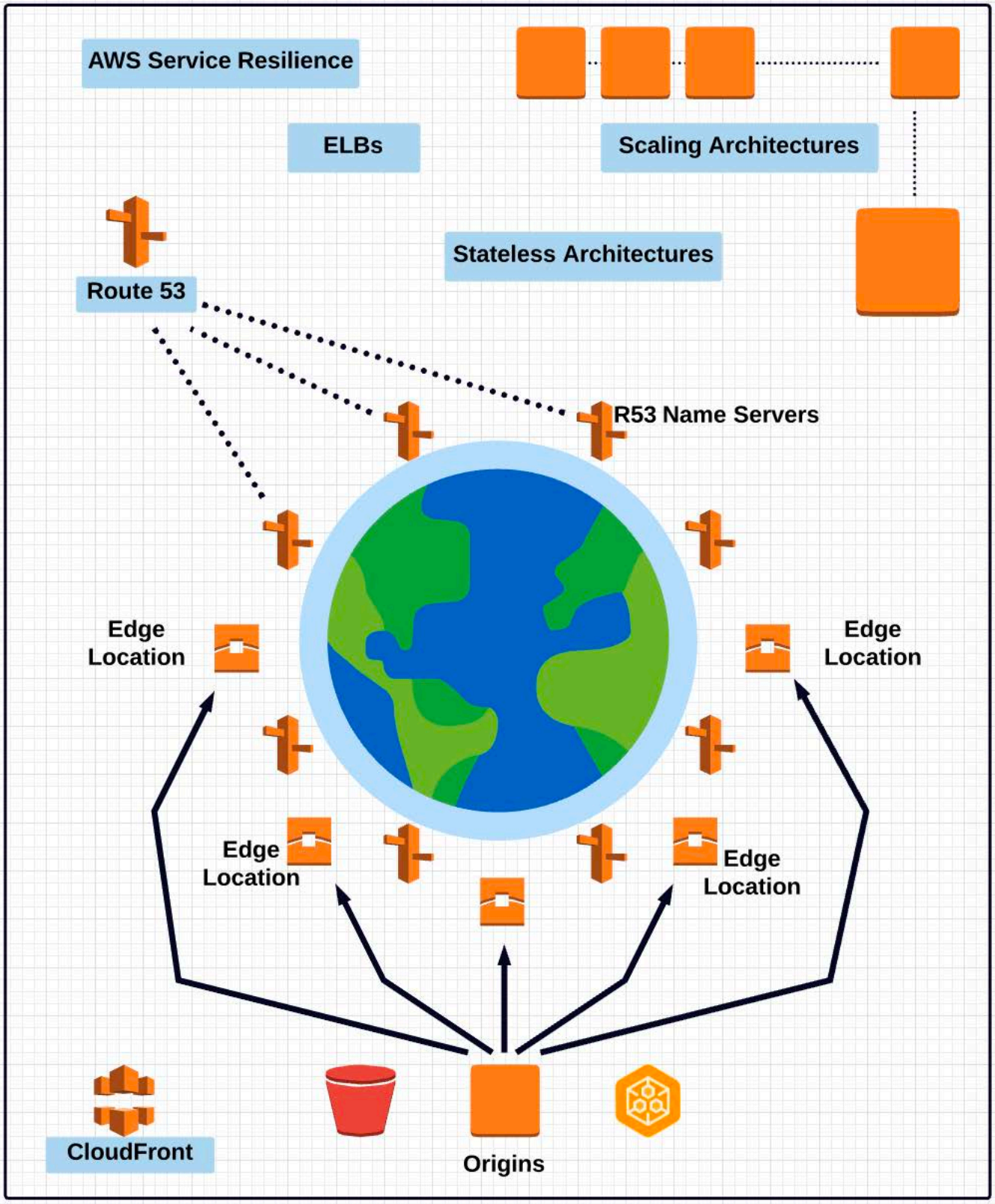
Inventories of vault contents is an asynchronous operation with notifications available from SNS. Glacier performs an automatic inventory of every vault every 24 hours. A vault inventory lists archive IDs, creation date, and the size.

No user-definable metadata is stored with archives in Glacier. Archives don't have a name — only a unique ID and optional description set when initially uploaded. Archives cannot be edited — they can only be deleted and new ones uploaded (given a new archive ID).

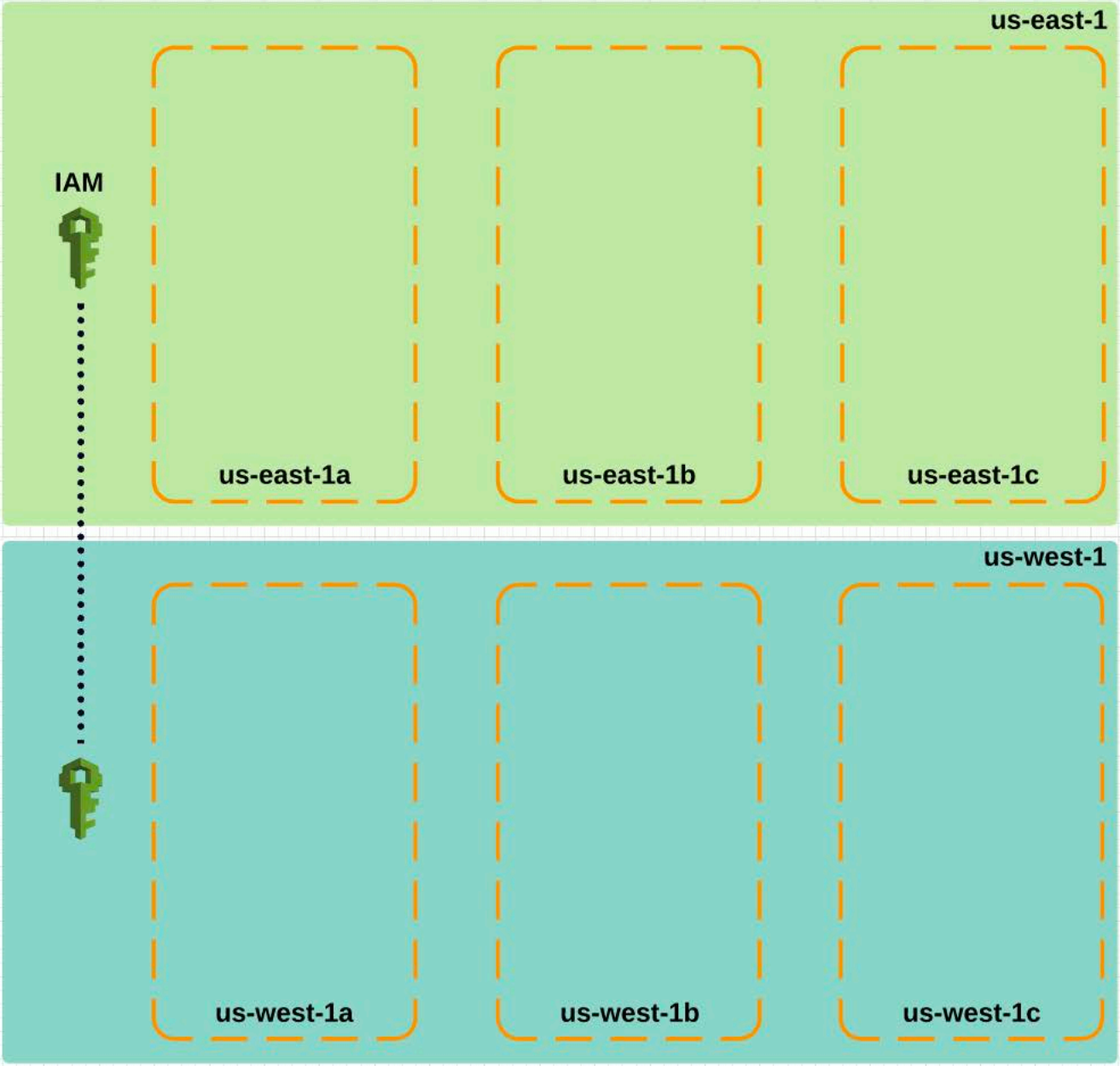
Any jobs to download archives are asynchronous, and there are three speeds:

- **Expedited:** Jobs are typically completed in 1-5 minutes for anything below 250 MB archives.
- **Standard:** Jobs are completed generally within 3-5 hours.
- **Bulk:** Economic option for large amounts — completed within 5-12 hours.

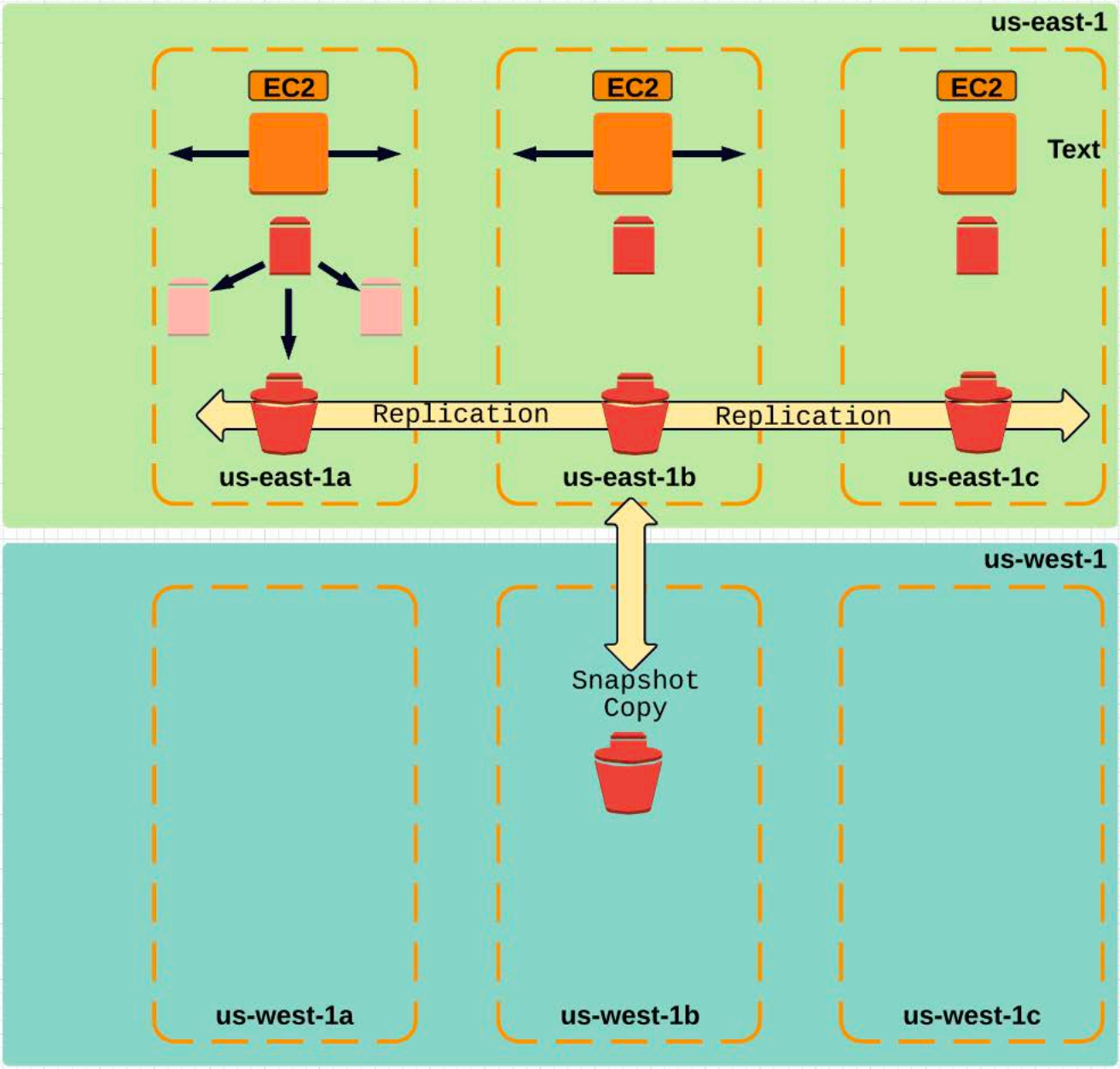

AWS
VPC
AWS Private Zone
AWS Public Zone
S3
S3-IA
Glacier
primary volume
snapshot
Public Internet
Direct Connect
Object
1:1 Mapping
Async Caching
Snaps to S3
Virtual tapes stored in S3; virtual shelf in Glacier
Storage Gateway
Primary Copy is Local
File Gateway
Volume Gateway
Tape Gateway
File
SMB
NFS
CACHE
iSCSI
VTL
virtual tape library
On-Premises Servers
On-Premises Servers
On-Premises Servers
On-Premises Data Center


AWS Service Resilience
ELBs
Scaling Architectures
Stateless Architectures
Route 53
R53 Name Servers
Edge Location
Edge Location
Edge Location
Edge Location
CloudFront
Origins

IAM | EC2/EBS | S3 | R53 | ELB/ALB
VPC | NAT GW | ASG | VPN



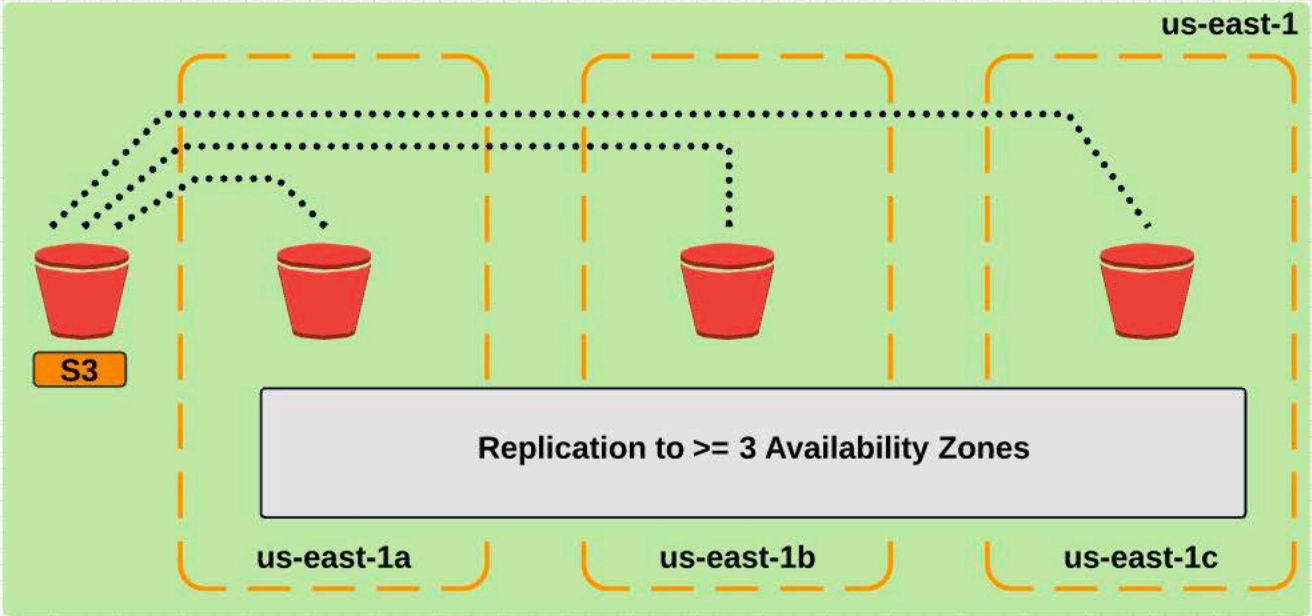
IAM is global — same data globally, accessible from all regions.

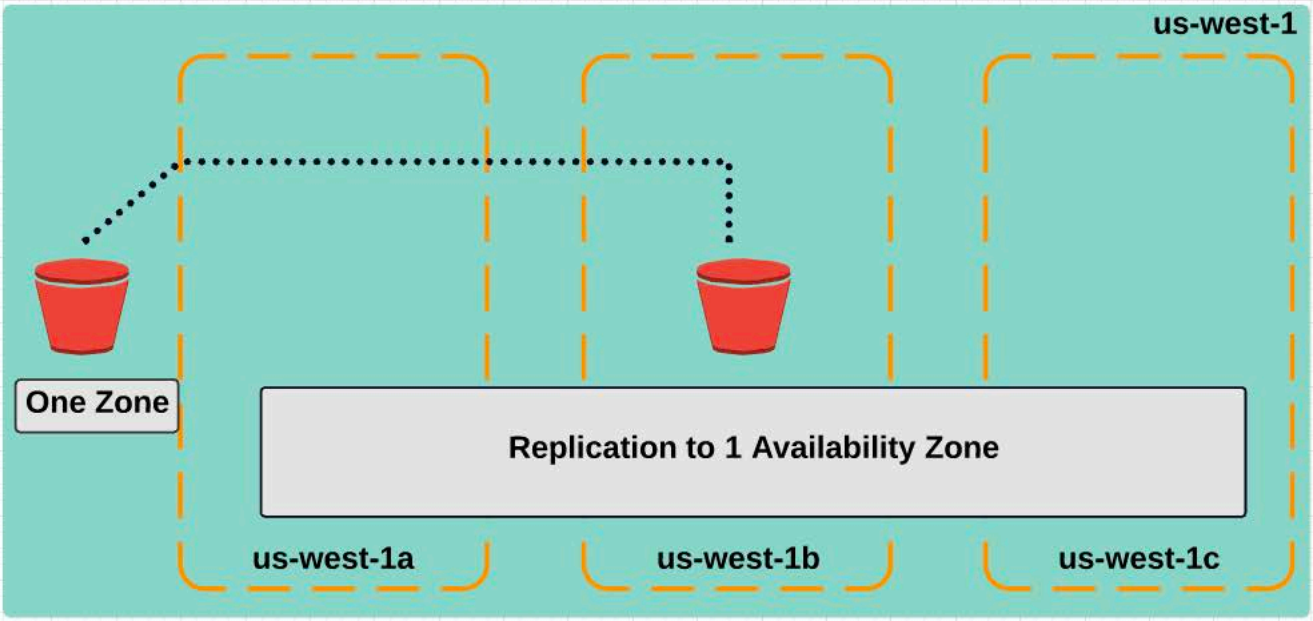


EC2 instances run on EC2 hosts located in specific AZs. Failure of an AZ will cause failure of that host and failure of all instances on the host. EBS volumes are stored in a specific AZ. Replication happens in that AZ, but AZ failure can result in data loss. Snapshots are replicated across AZs and optionally copied cross-region.

?
Back
Global Infrastructure
Account & Services
Networking
Well-Architected
Appendix

IAM
EC2/EBS
S3
R53
ELB/ALB
VPC
NAT GW
ASG
VPN





S3 is a regional service. For most storage classes, data is replicated across three or more AZs in the region a bucket is located in.

?
Back
Global Infrastructure
Account & Services
Networking
Well-Architected
Appendix

IAM | EC2/EBS | S3 | R53 | ELB/ALB

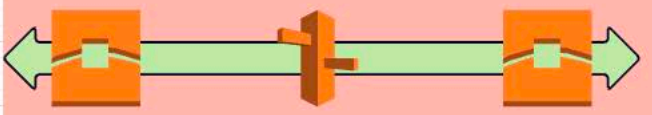
VPC | NAT GW | ASG | VPN

us-east-1

us-east-1a | us-east-1b | us-east-1c

us-west-1

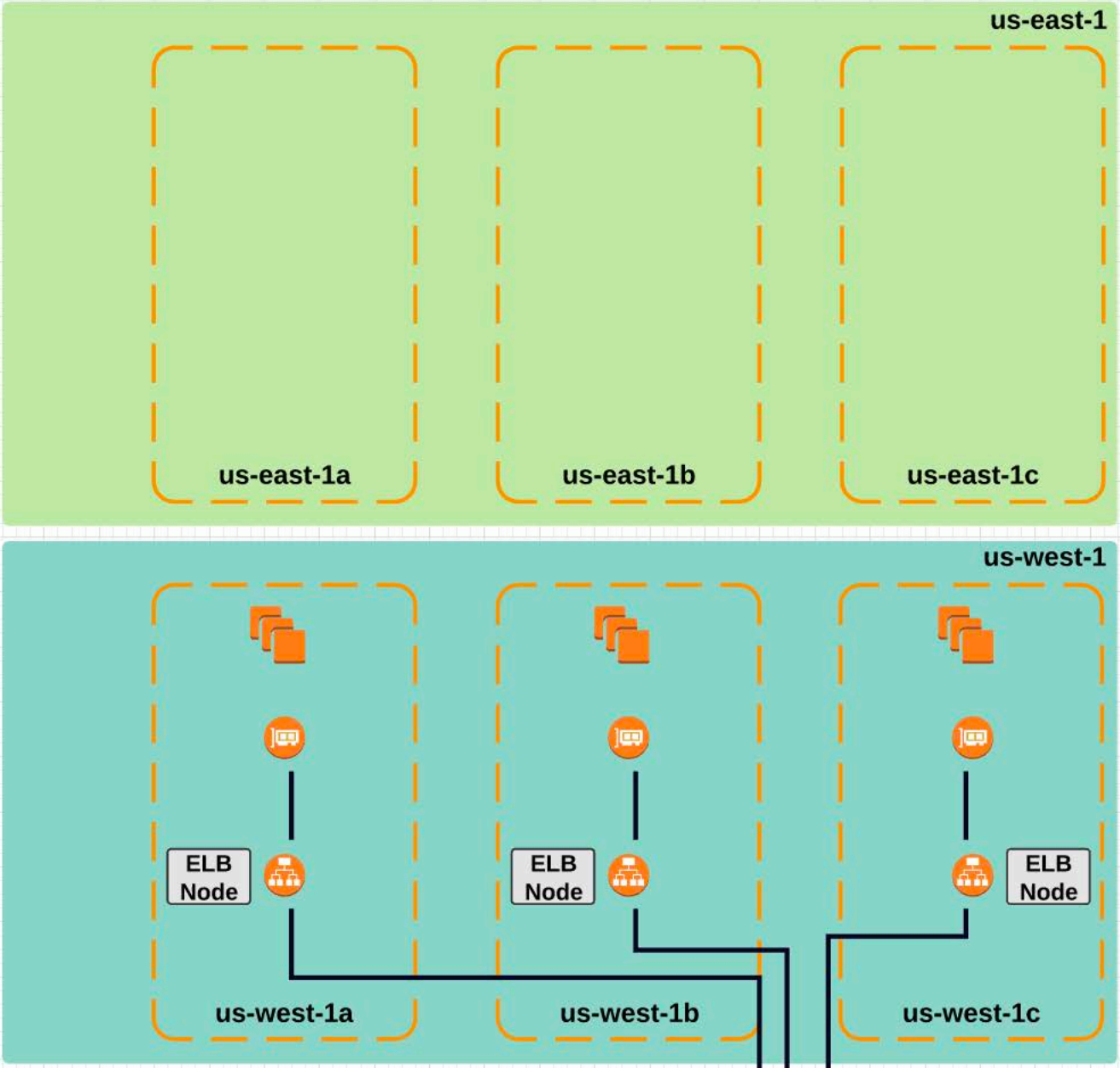
us-west-1a | us-west-1b | us-west-1c



Route 53 operates its name servers from edge locations positioned globally. As such, it's resistant even to issues in AWS regions.

?
Back
Global Infrastructure
Account & Services
Networking
Well-Architected
Appendix

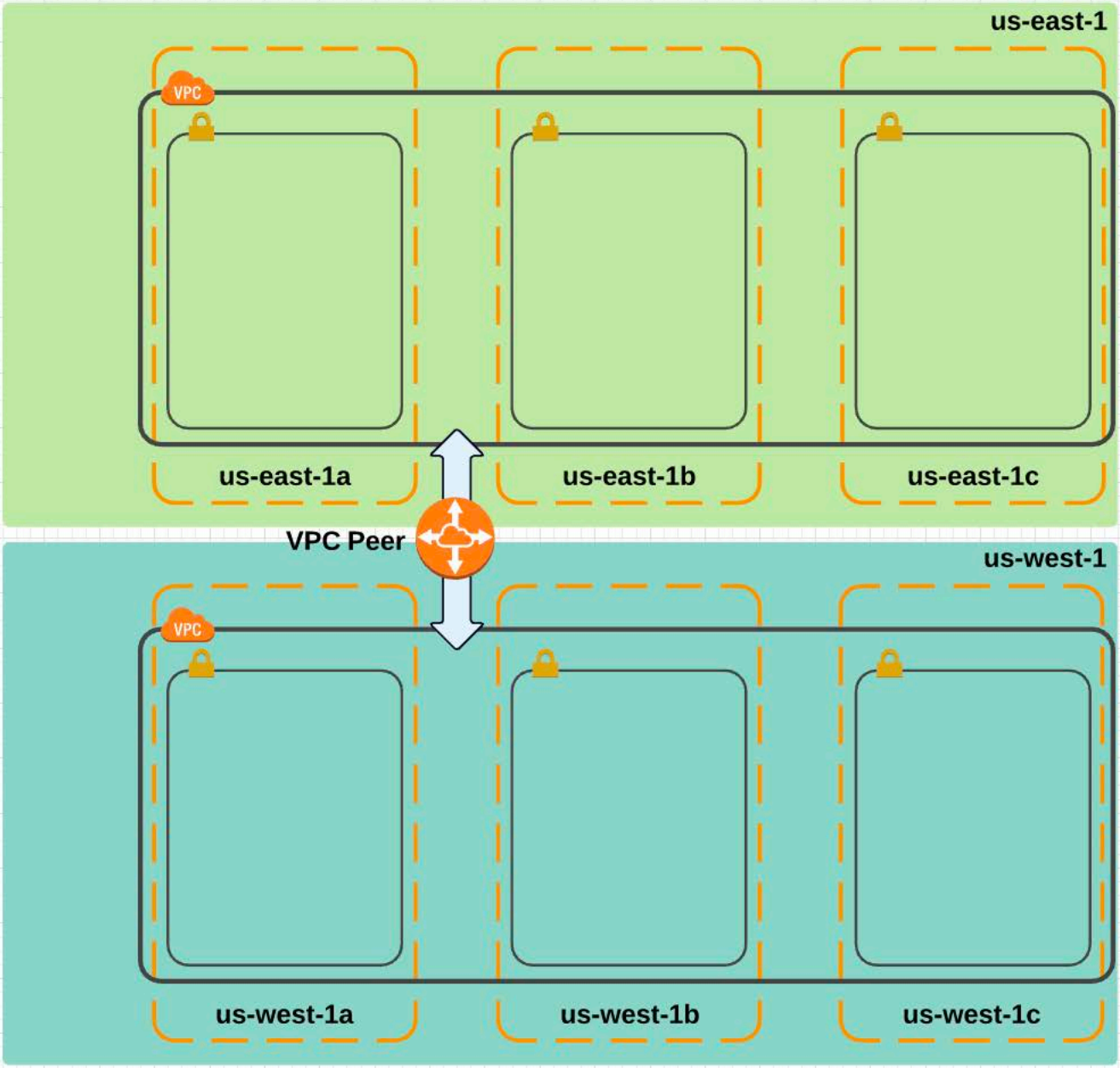
IAM
EC2/EBS
S3
R53
ELB/ALB
VPC
NAT GW
ASG
VPN



ELBs are regional services and are enabled for specific Availability Zones. When enabled, an ELB node is provisioned for an AZ and network interfaces are placed in a specific subnet. The suggested architecture is to enable the ELB in all AZs where instances are placed.

?
Back
Global Infrastructure
Account & Services
Networking
Well-Architected
Appendix

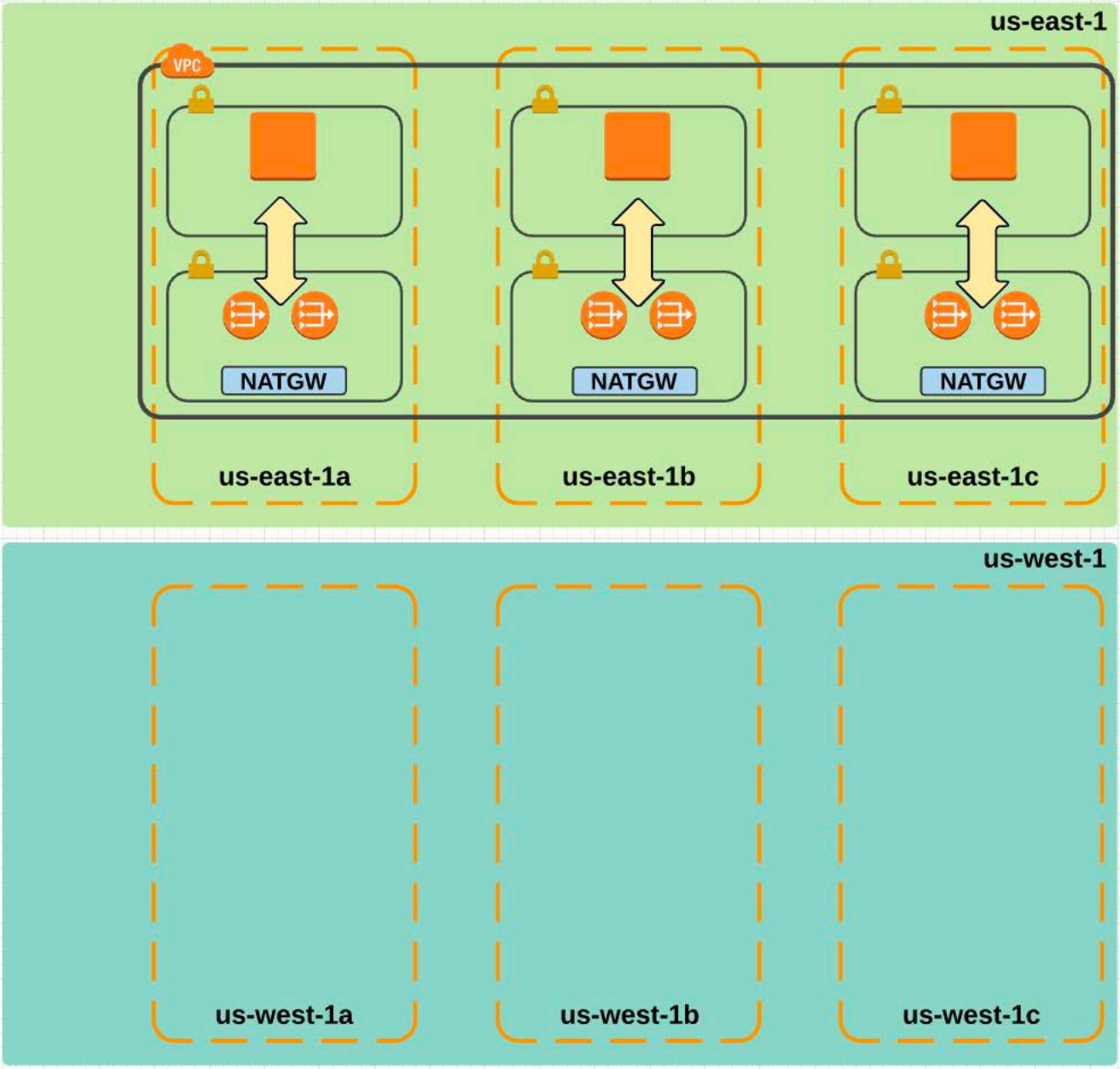
IAM
EC2/EBS
S3
R53
ELB/ALB
VPC
NAT GW
ASG
VPN



VPCs are regional services — they can operate in all Availability Zones in a region. They are tolerant to AZ failure in a region, but certain components such as subnets are linked to a particular AZ. Other components, such as internet gateways and VPC peering, is HA by design.

?
Back
Global Infrastructure
Account & Services
Networking
Well-Architected
Appendix

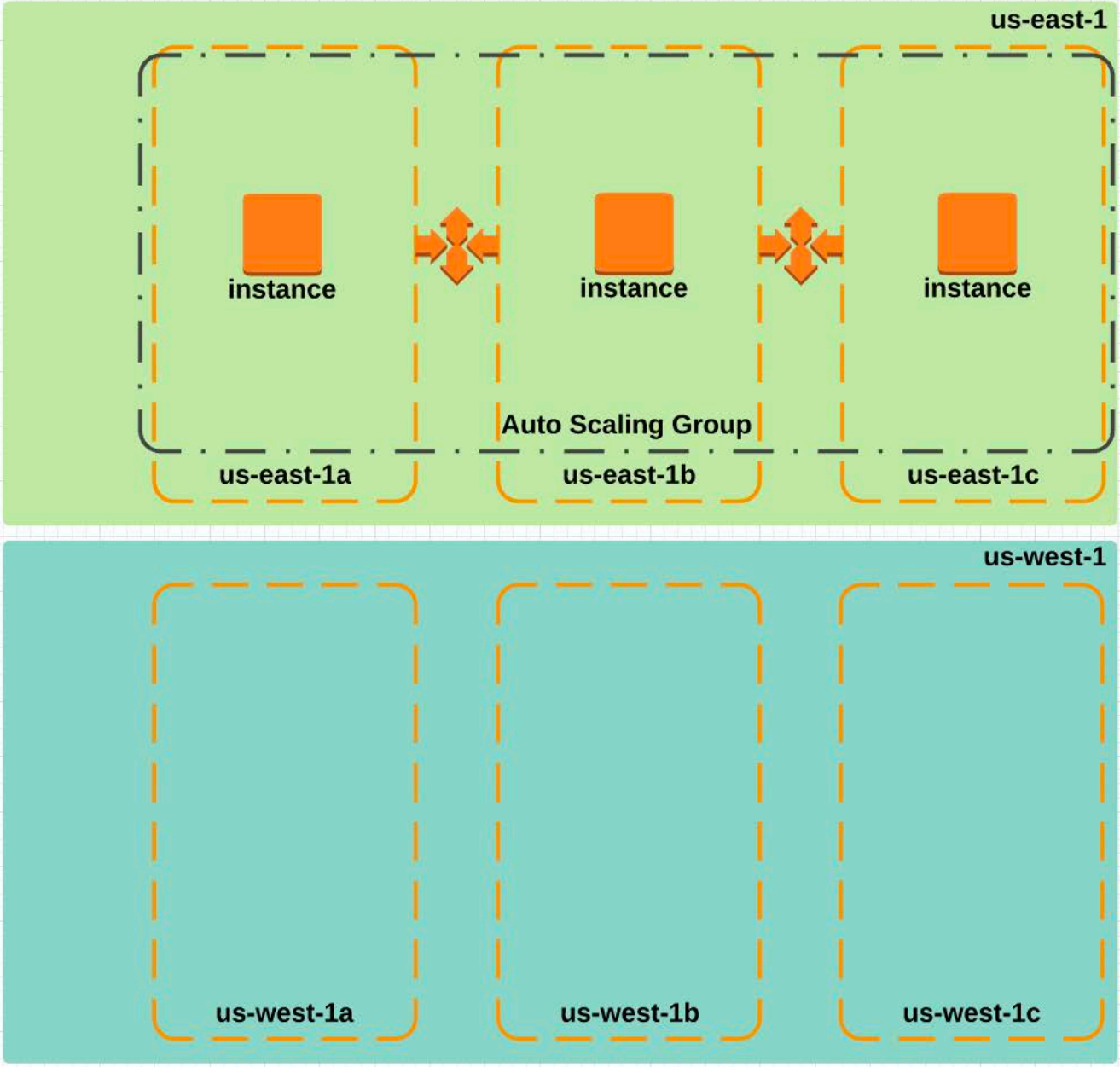
IAM | EC2/EBS | S3 | R53 | ELB/ALB
VPC | NAT GW | ASG | VPN



NAT Gateways operate in a specific subnet that is linked to an Availability Zone. NAT Gateways are implemented with redundancy inside each subnet — but to implement zone-independent redundancy, create a NAT Gateway in each Availability Zone.

?
Back
Global Infrastructure
Account & Services
Networking
Well-Architected
Appendix

IAM
EC2/EBS
S3
R53
ELB/ALB
VPC
NAT GW
ASG
VPN

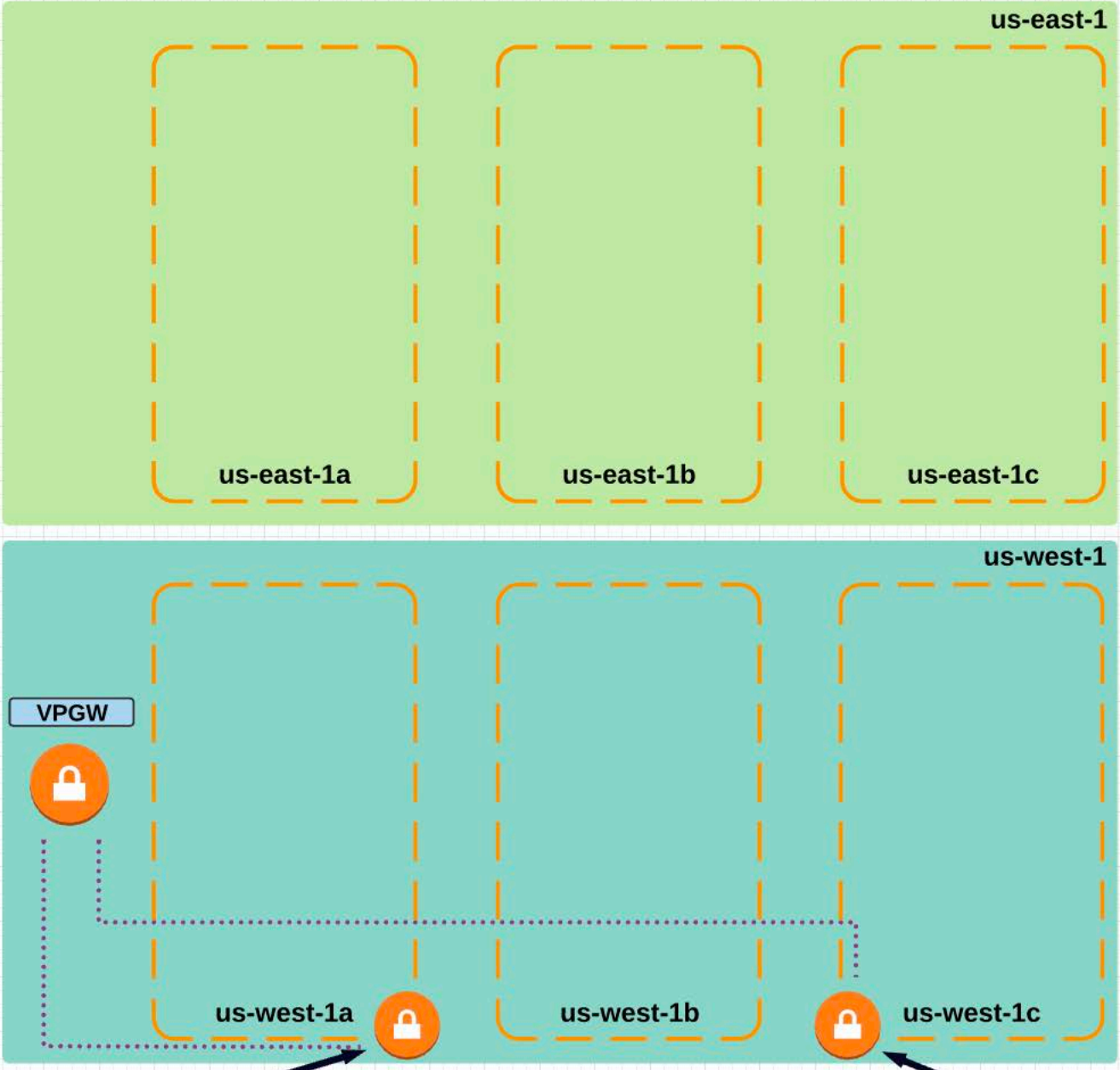


EC2 Auto Scaling groups allow EC2 instances to be provisioned and terminated based on load demands. Because min, max, and desired values are specified, ASGs can manage a unit of compute capable of reacting to AZ failure. ASGs operate in a VPC across subnets in that VPC.

?
Back
Global Infrastructure
Account & Services
Networking
Well-Architected
Appendix

IAM | EC2/EBS | S3 | R53 | ELB/ALB

VPC | NAT GW | ASG | VPN



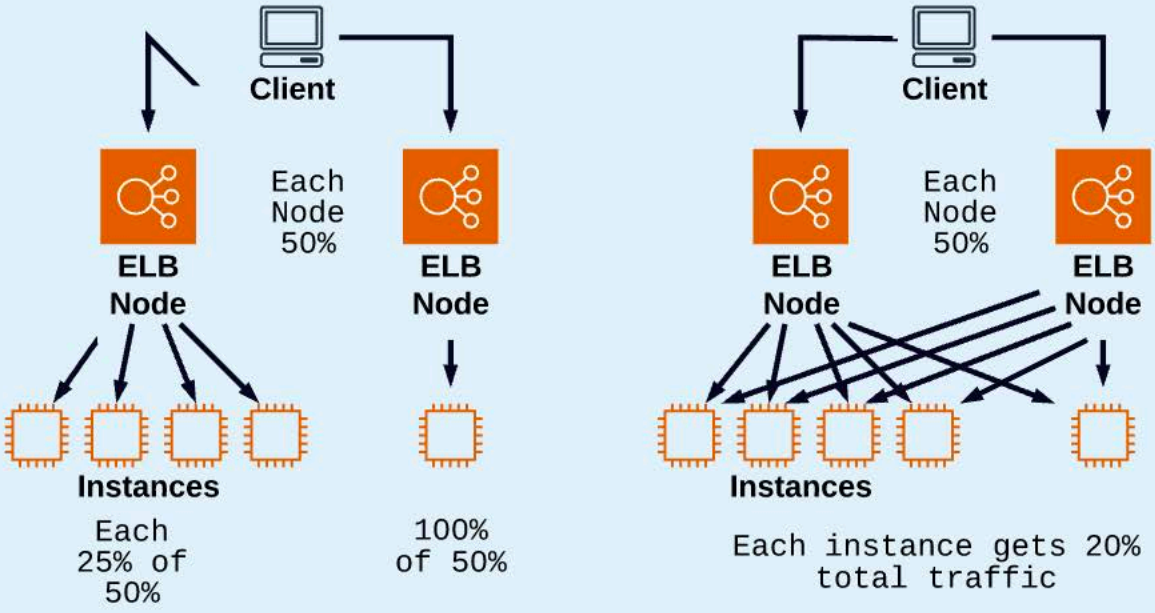
Virtual private gateways provision two IPsec endpoints. These endpoints operate in different Availability Zones and so can tolerate failures of individual Availability Zones.

Elastic Load Balancing (ELB)

X

ELB Essentials | Classic LB | Application LB | Network LB

- **Load balancing** is a common method used for distributing incoming traffic among servers.
- Incoming connections are made *to* the load balancer, which distributes them in some way to associated resources.
- **Elastic Load Balancing (ELB)** is a service that provides a highly available and scalable load balancer as a service. It's available in three forms: Classic (CLB), Application (ALB), and Network (NLB).
- Elastic Load Balancing can be paired with Auto Scaling to enhance high availability and fault tolerance, as well as allow for automated scalability and elasticity.
- An ELB has its own DNS record set (A record) that allows for direct access from its external side.
- Cross-zone load balancing: An ELB can load balance traffic to instances located across multiple Availability Zones. This allows for a highly available and fault-tolerant architecture.



- An ELB can be **public facing**, meaning it accepts traffic from the public internet, or an **internal** load balancer, which is only accessible from within a VPC and is often used to decouple tiers of an application.
- ELBs will automatically stop serving traffic to an instance that becomes unhealthy (via **health checks**).
- A CLB or ALB can help reduce compute power on an EC2 instance by allowing for an SSL certificate to be applied directly to the ELB. SSL encryption and description is "**offloaded**" by the ELB in a process called *SSL offload*.

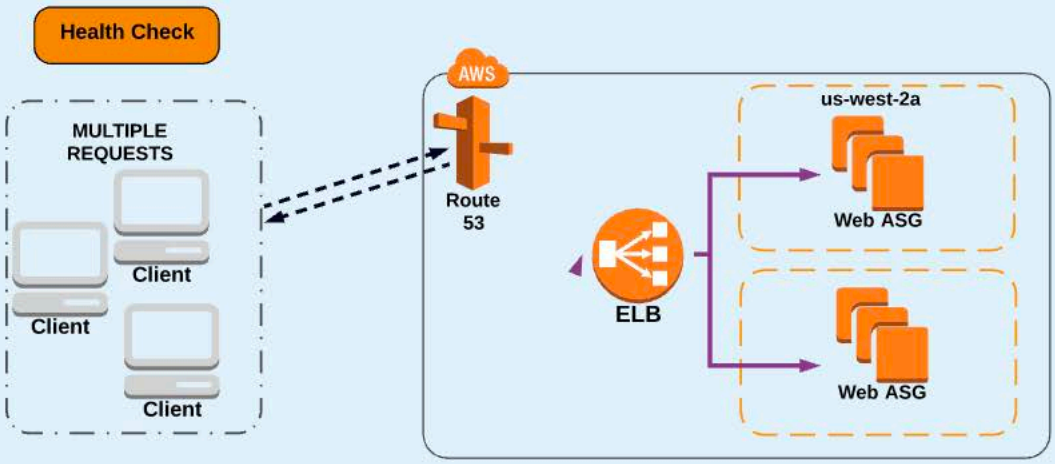
Elastic Load Balancing (ELB)

X

ELB Essentials | Classic LB | Application LB | Network LB

Classic Elastic Load Balancer

- A "classic" elastic load balancer is designed for balancing traffic (load) to multiple EC2 instances.
- It was originally introduced with ec2-classic and is *not* recommended for use within a VPC.
- It isn't a Layer 7 device, so there are no granular routing "rules" — all instances get routed to evenly, and no special routing requests can be made based on specific paths in the request (e.g., /images/).
- A listener is created that listens to client requests and routes them to the correct destination on the backend.
- There are several protocols supported, including:
  - TCP
  - SSL
  - HTTP
  - HTTPS
- Utilizing SSL/TLS at the load balancer level is considered SSL offloading; all of the cipher suite work is done at the load balancer, which frees up resources on the servers.
- Classic Load Balancers are usually associated with Auto Scaling groups, allowing for a stateless architecture.
- One SSL certificate and cipher configuration *per* listener (effectively per CLB).
- You do *not* assign public IPs to load balancers! Communication is based on an A record (DNS).



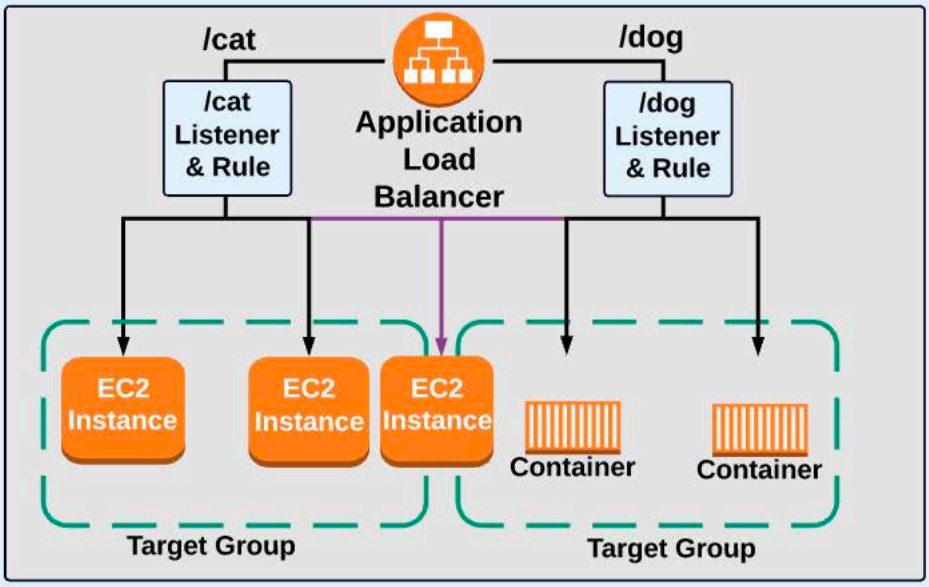
Elastic Load Balancing (ELB)

X

ELB Essentials | Classic LB | Application LB | Network LB

Application Load Balancer

- Application Load Balancers operate and understand up to and including Layer 7 of the OSI 7-Layer networking model.
- ALBs are the recommended load balancer type to use within VCPs rather than Classic Load Balancers.
- They perform better than Classic Load Balancers, are almost always cheaper, and can take routing decisions based on Application level (Layer 7) specifics.
- Content rules are used to direct traffic toward target groups.
- Content-based rules (set up on the listener) can be configured using:
  - **Host-based rules**: Route traffic based on the *host field* of the HTTP header.
  - **Path-based rules**: Route traffic based on the *URL path* of the HTTP header.
  - This allows you to structure your application as smaller services and even monitor/Auto Scale based on traffic to specific **"target groups."**
  - You can balance traffic to multiple ports.
- An ALB also supports ECS and EKS, HTTPS, HTTP/2, WebSockets, access logs, sticky sessions, and AWS WAF (web application firewall).
- ALBs allow routing requests to multiple applications on one EC2 such as containers.
- ALBs can be used with other AWS services such as ECS for microservices.
- Targets can belong to multiple target groups.
- ALBs can send traffic to Lambda functions.
- Mutiple SSL certificates can be managed by a single ALB using SNI.



Elastic Load Balancing (ELB)

X

ELB Essentials | Classic LB | Application LB | Network LB

Network Load Balancer

Network Load Balancers operate at Layer 4 of the OSI 7-Layer model. The Network Load Balancer is designed for *extreme performance*. It's capable of handling millions of requests per second and can scale to handle volatile and unpredictable traffic levels.

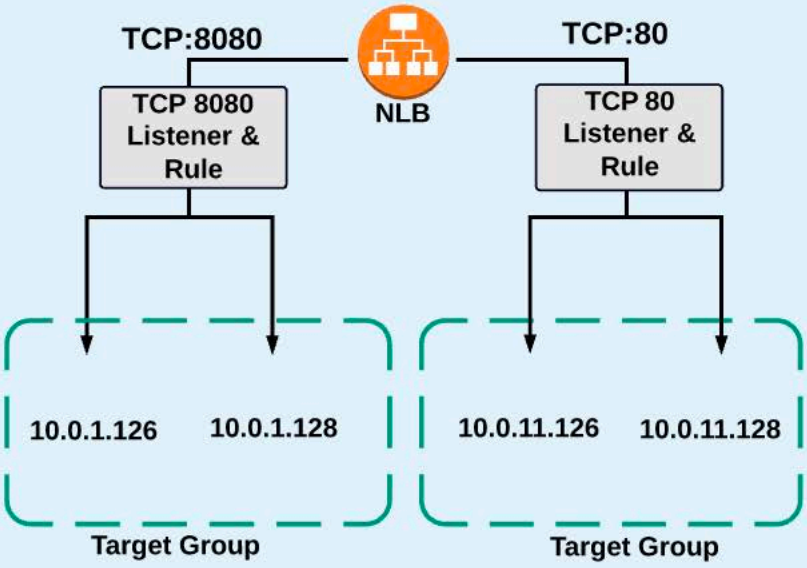
Network Load Balancers support static IPs and ultra low latency.

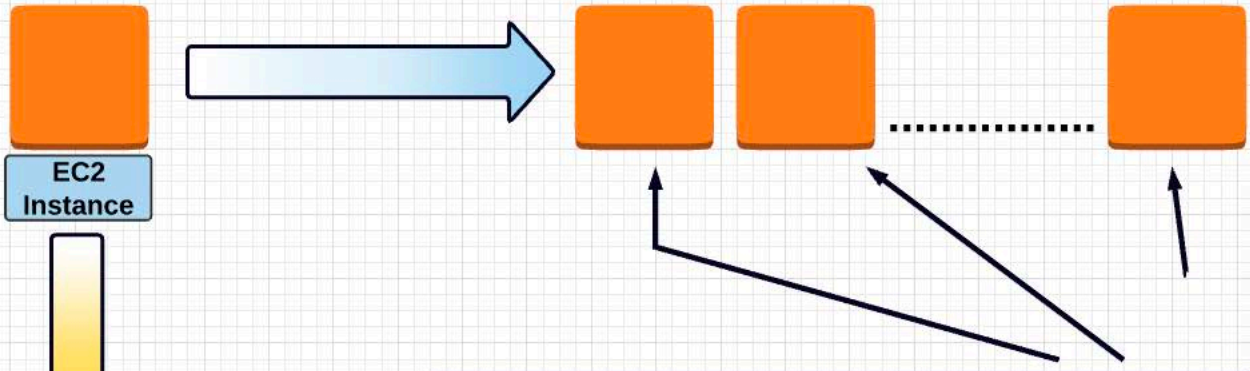
Network Load Balancers share the same core configuration entities with ALBs:

- Listeners
- Target groups
- Targets

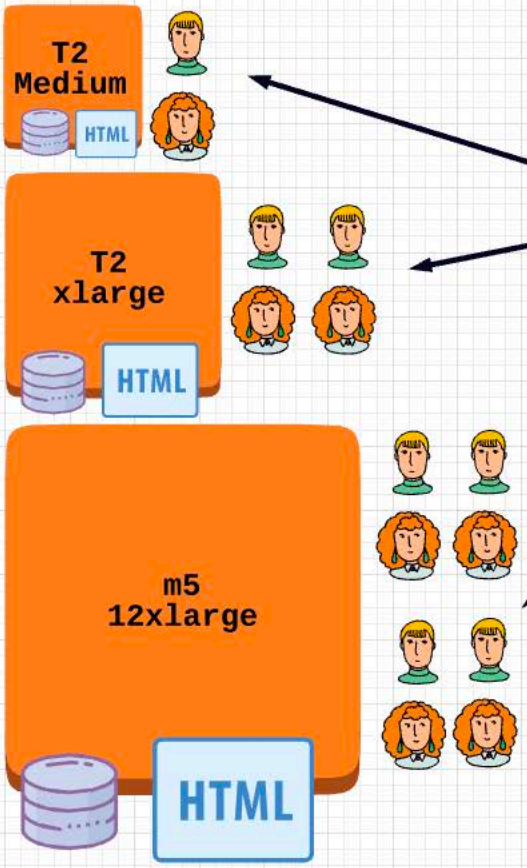
NLBs are assigned a single IP address per Availability Zone and can be assigned an EIP, including ones you own for the lifetime of the NLB. This allows for *much* easier firewall management because this static IP can be whitelisted.

Network Load Balancers **preserve the client IP.** It's a passthrough device — they don't alter the source IP packets.

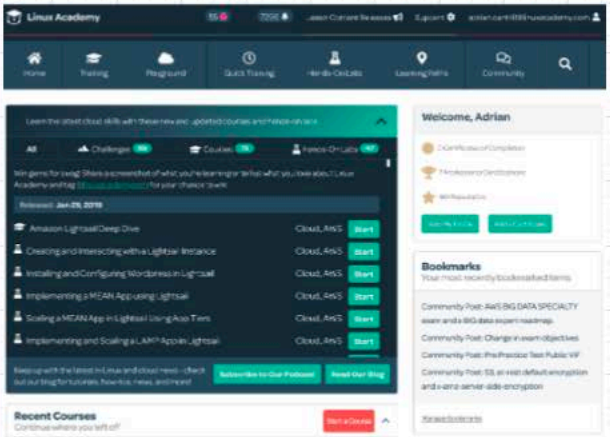




Horizontal scaling means increasing or decreasing system load by increasing or decreasing the number of instances serving the demand. Elasticity is an evolution of this process where this scaling is fully automatic based on system load.

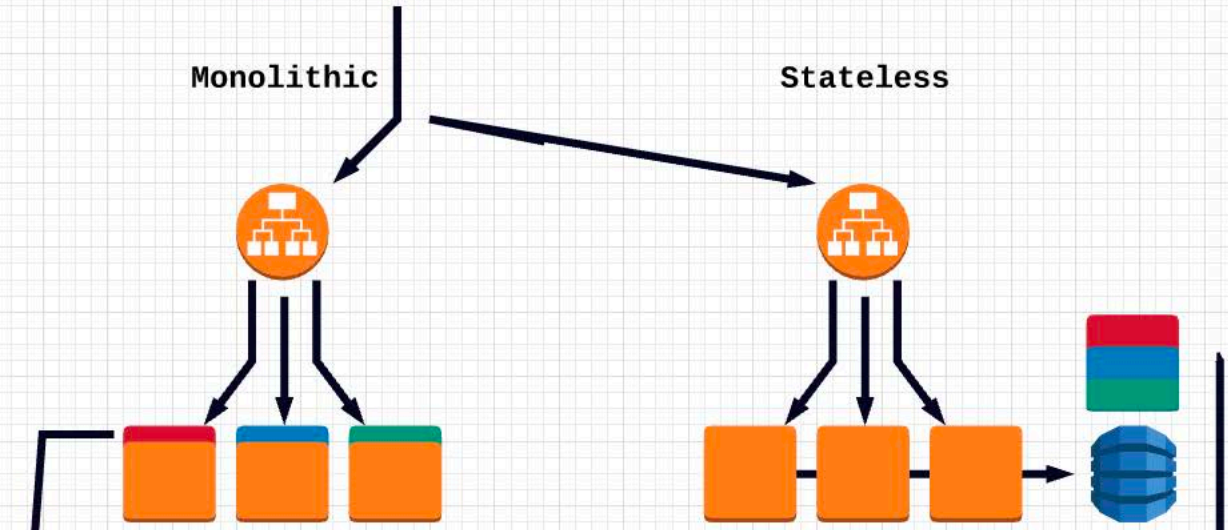


Vertical scaling means increasing load demands by increasing the size of compute resources. Extra CPU, memory, and disk capacity allow systems to handle additional load. Vertical scaling is generally used when applications are monolithic and the state is stored with the application on the server.



With a monolithic and unscalable architecture, the "state" is stored on the application server. Examples of the state include a shopping cart or the login state of a web application.

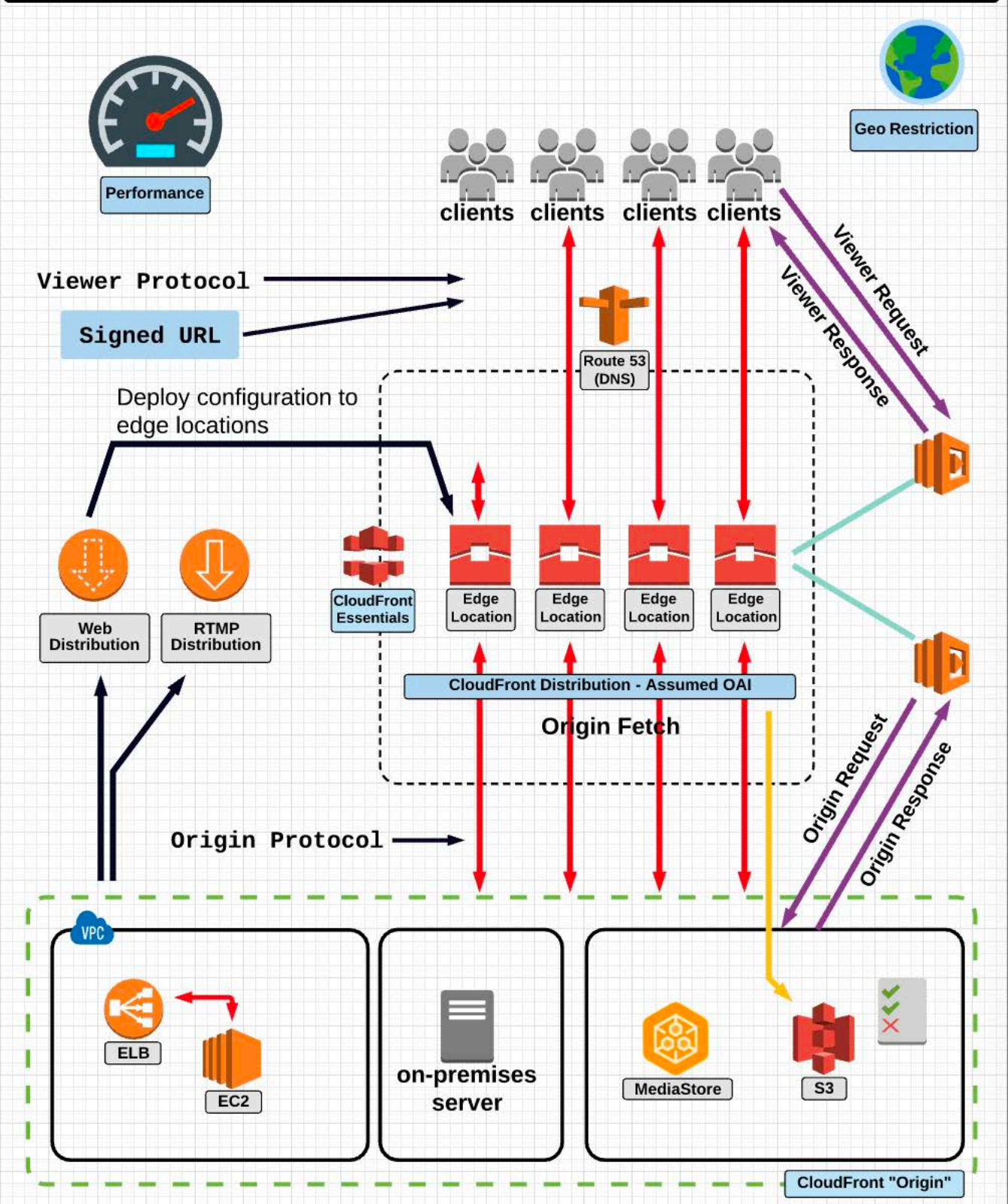
Using a stateless architecture, the state is held separately from the application servers.



Stateless architectures allow load to be evenly distributed across any available instances and allow traffic to be redistributed when failure occurs without disrupting the user experience.

Session data is stored on the application servers. Each session store is unique to the server. If the server fails, the load balancer will redirect clients to alternative servers. Because the session data will be lost, the user experience will be disrupted.

With stateless servers and applications, the session state is moved to a separate store such as DynamoDB. With this style of architecture, when an individual instance fails and traffic is migrated to another instance, session data is still available.


Performance
Geo Restriction
clients
clients
clients
clients
Viewer Protocol
Signed URL
Route 53 (DNS)
Viewer Request
Viewer Response
Deploy configuration to edge locations
Web Distribution
RTMP Distribution
CloudFront Essentials
Edge Location
Edge Location
Edge Location
Edge Location
CloudFront Distribution - Assumed OAI
Origin Fetch
Origin Request
Origin Response
Origin Protocol
VPC
ELB
EC2
on-premises server
MediaStore
S3
CloudFront "Origin"

CloudFront Performance Considerations

X

- Regional caches
  - Save content ejected from cache close to edge location

- Point entire domain to CloudFront to speed up dynamic content
  - Connection optimizations
  - Dynamic objects are still cached

- CloudFront performance can be affected by:
  - File size and type of file
  - Having to remake the request from the edge location to the origin
    - Downloading the object from the origin takes time.
    - As well as writing it to cache and responding to the end-user request.
    - The more requests that have to go to the origin, the higher the load is on your source — which can also cause latency and load performance issues.

  - Query strings (request to the origin to serve a specific object) reduce cache “hits”:
    - http://bestcatpicsintheworldever/?query=redcats
    - It reduces performance because query strings are often unique, so it reduces the cache hits and also requires extra “work” in order to forward to the origin location.

- CloudFront performance can be increased by:
  - Longer cache periods increase performance (less frequent requests to the source).

CloudFront Essentials

X

- CloudFront is a global CDN that delivers content to users from the nearest edge location.
- CloudFront retrieves content from origins.
- CloudFront distribution (web or RTMP)
  - Define origins for content (static and dynamic)
  - CloudFront can integrate with Route 53 for “alternate” CNAMEs.
    - This allows you to create a URL such as http://bestcatpicsintheworldever.com that works with your distribution.
  - Cache behaviors
    - PATH pattern (images/*.jpg)
    - Min, Max, Default TTL
    - Query string forwarding and caching
    - HTTP methods (GET, POST, PUT, HEAD, etc.)

Updating Cached Files:

- Caching is done based on the cache key.
- In order to serve a new version of an object, either create a new object with a new name or create an “invalidation” on the CloudFront distribution based on the object name.
- "Invalidations" can be costly.
- Set TTL = 0

X

Restricting S3 to CloudFront

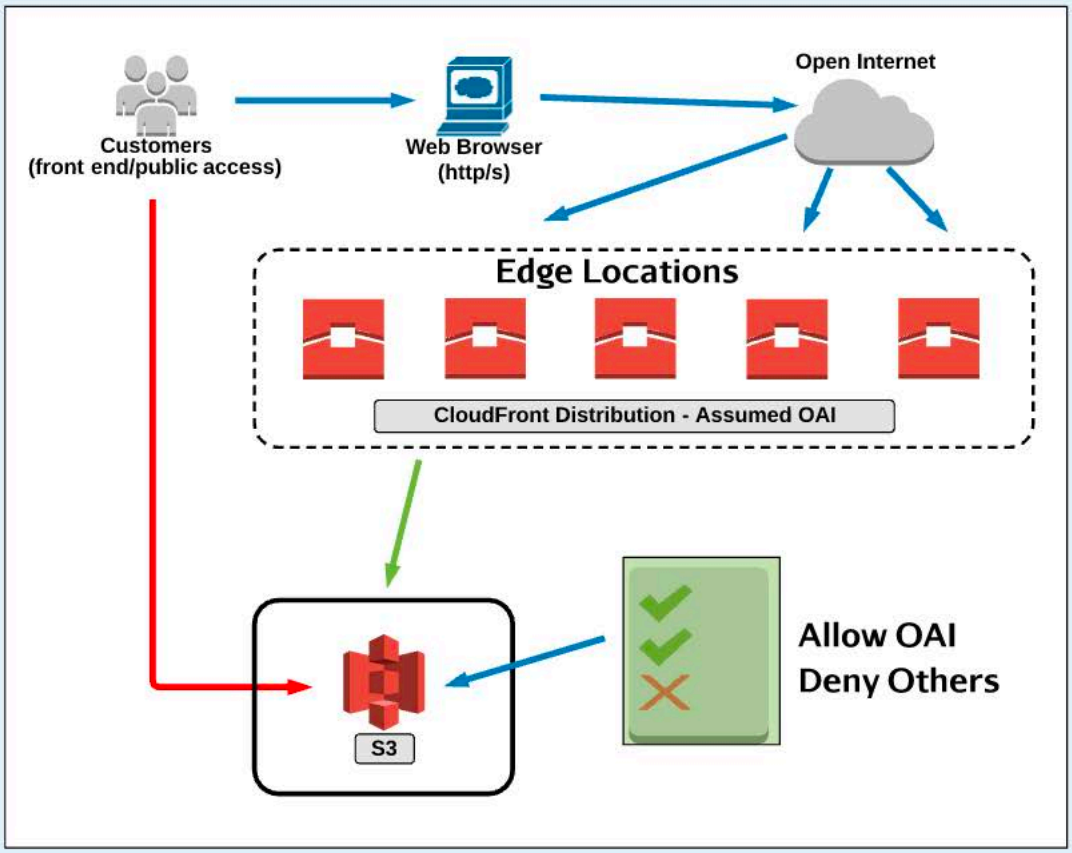
By default, when using CloudFront with S3, CloudFront is optional, and S3 can be accessed directly. This can be changed by creating an origin access identity (**OAI**).

What's an OAI?

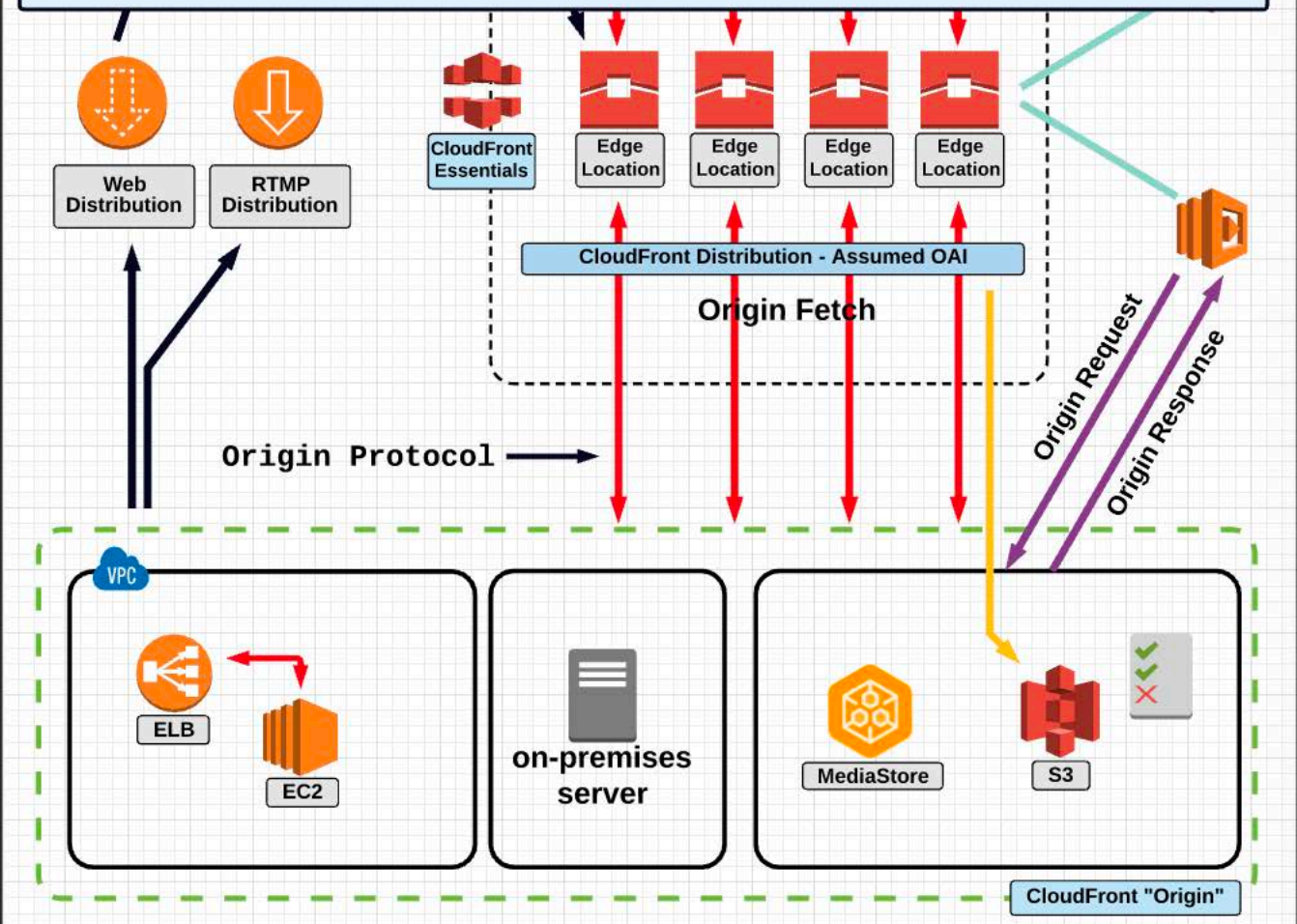
An OAI is a "virtual" identity. A distribution can be configured to use it so that, when accessing S3, CloudFront assumes this identity.

How Is an OIA Used and Why?

To use an OAI, public permissions are removed from your S3 bucket policy and permissions for the OAI are added. Only the CloudFront using that OAI can access your S3 bucket.

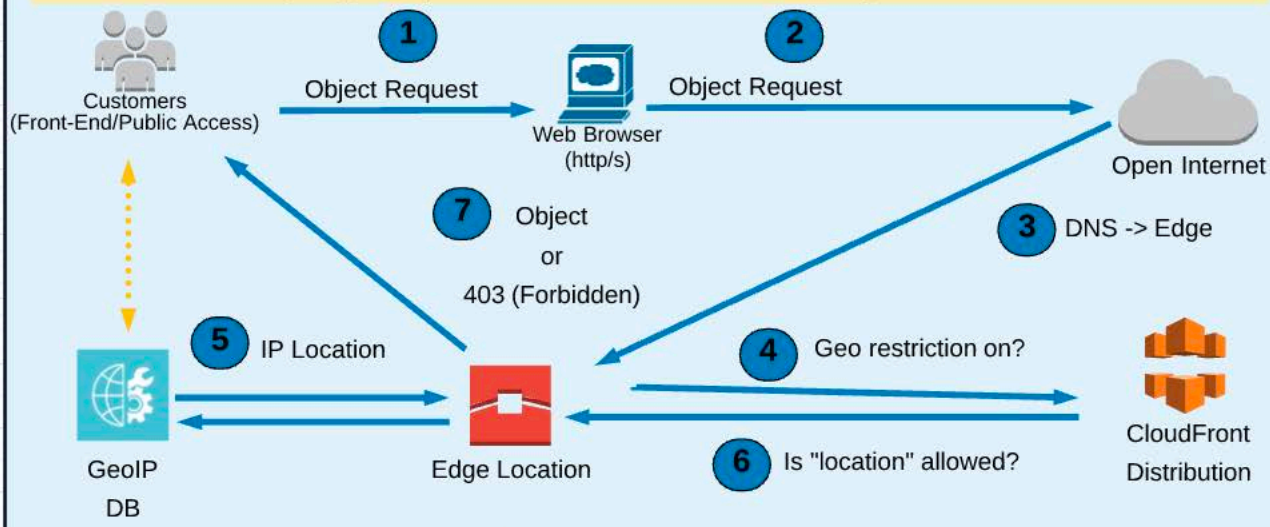


```
{
    "Version": "2008-10-17",
    "Id": "PolicyForCloudFrontPrivateContent",
    "Statement": [
        {
            "Sid": "1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin
Access Identity XXXXXXXXXXXX"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::mybucket/*"
        }
    ]
}
```

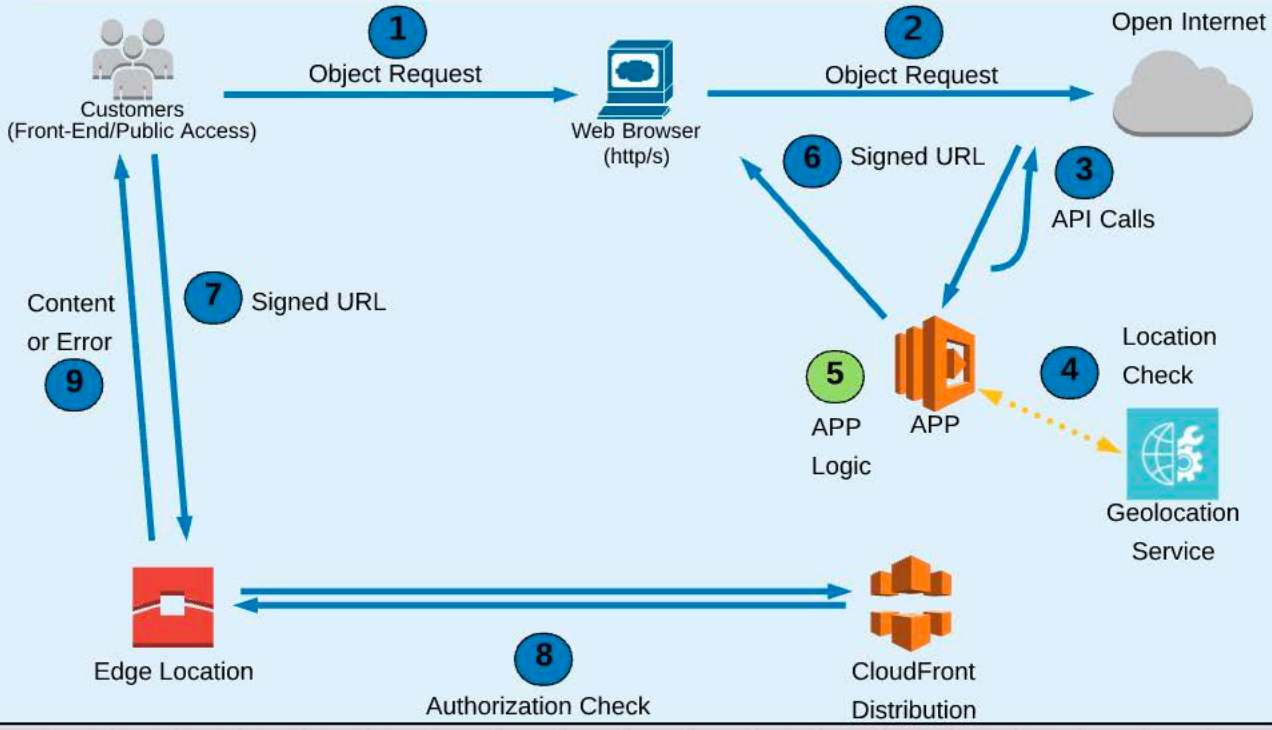


Geo Restriction — CloudFront can restrict content in one of two ways:

CloudFront geo restriction is a simple implementation.
Whitelist *or* blacklist, and it works on country restrictions *only*.
Location is based on IP country location — backed by a GeoIP database (~99.8% accuracy)
No restrictions on *anything else* (session/cookie/content/browser, etc.).

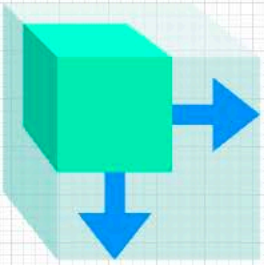


Third-party geo restriction needs a server/serverless application — signed URLs are used. A third-party geolocation service is used for extra accuracy. Your application can apply additional restrictions such as session, browser, or account. Location can be *much* more accurate: city, locale, latitude/longitude in some cases.



?
Back
Global Infrastructure
Account & Services
Networking
Well-Architected
Appendix

EC2 Self-Managed Databases



Database Models

SQL Databases

Aurora

RDS

Athena

NoSQL Databases

DynamoDB

Neptune

QLDB

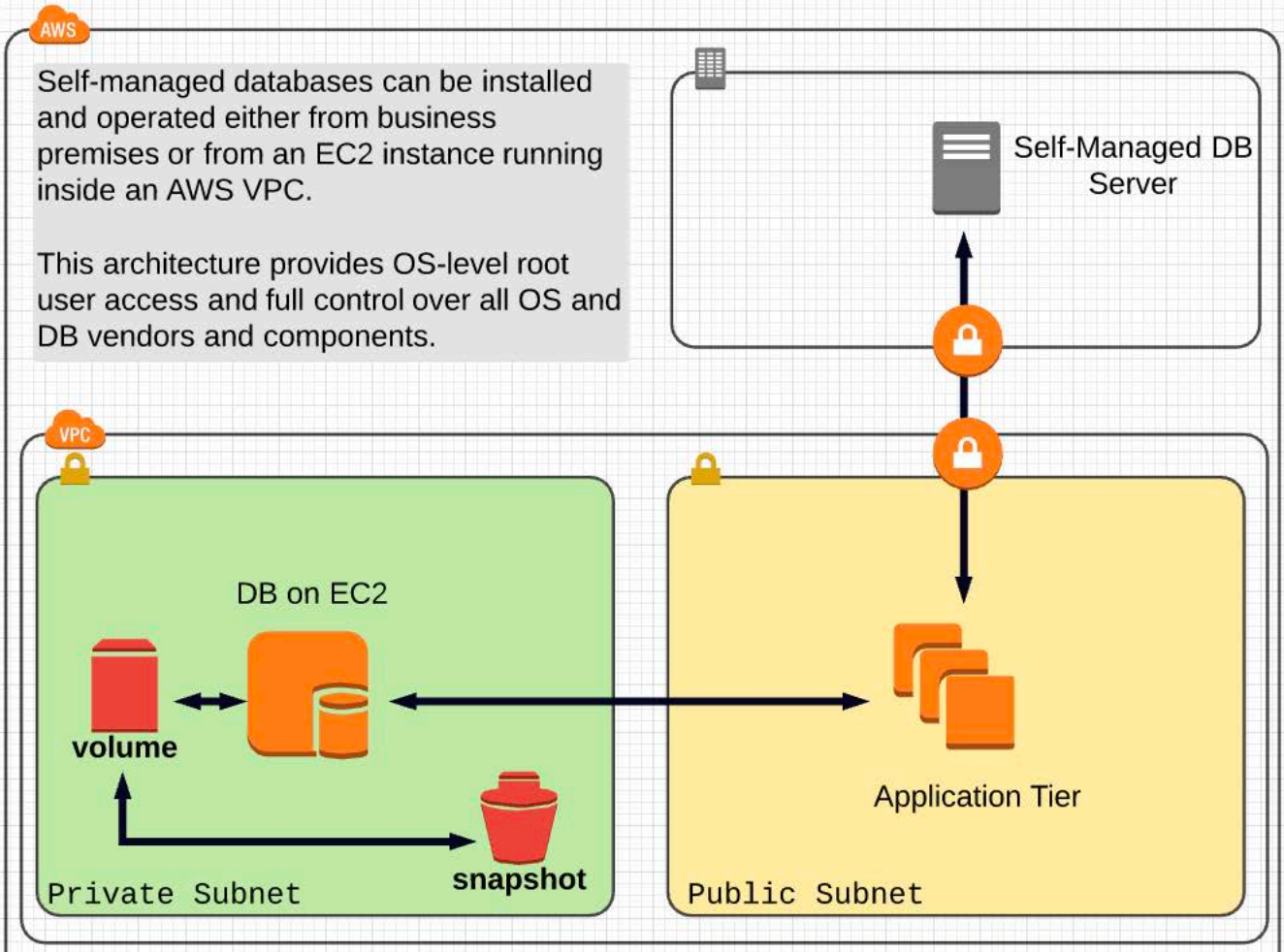
Timestream

DocumentDB

Elasticache

Self-managed databases can be installed and operated either from business premises or from an EC2 instance running inside an AWS VPC.

This architecture provides OS-level root user access and full control over all OS and DB vendors and components.



Risks and Limitations

Running a database (DB) on EC2 adds admin overhead. The system administrator is responsible for installation, configuration, performance management,, and ongoing backup and DR issues.

Additionally, the DB runs on EC2, bringing additional risk of data or service loss if an AZ fails. The EC2 instance and backing EBS volume are both "AZ" services.

Backups can be performed via EBS snapshots to S3. These backups are replicated across multiple AZs in the region but are self-managed and the responsibility of the system's administrator.

If HA is required, the system administrator is responsible for any sync or async replication in addition to management of additional DB nodes.

Relational Database

Relational database management systems (RDBMS) are used when the data to be managed has formal and fixed relationships. Data is stored on disk as rows, and entire rows must be parsed even if individual attributes are all that's needed. Reading 1 attribute from 10,000 records requires 10,000 rows to be read from disk.

Every table has a schema that defines a fixed layout for each row, defined when the table is created. Every row in the table **needs** to have all the attributes and the correct data types.

RDBMS conforms to the ACID system: **A**tomicity, **C**onsistency, **I**solation, and **D**urability. This does impact the ability to achieve high performance levels and limits scalability, but for most applications, the tradeoff is worth it.

Data stored in RDBMS is stored in a normalized form, which means removing repeating data and storing only single values in attributes. To query data from related tables (see below), table joins are used, where attributes are matched based on primary and foreign keys.

SQL (or Structured Query Language) is used to interact with most SQL (relational) database products.

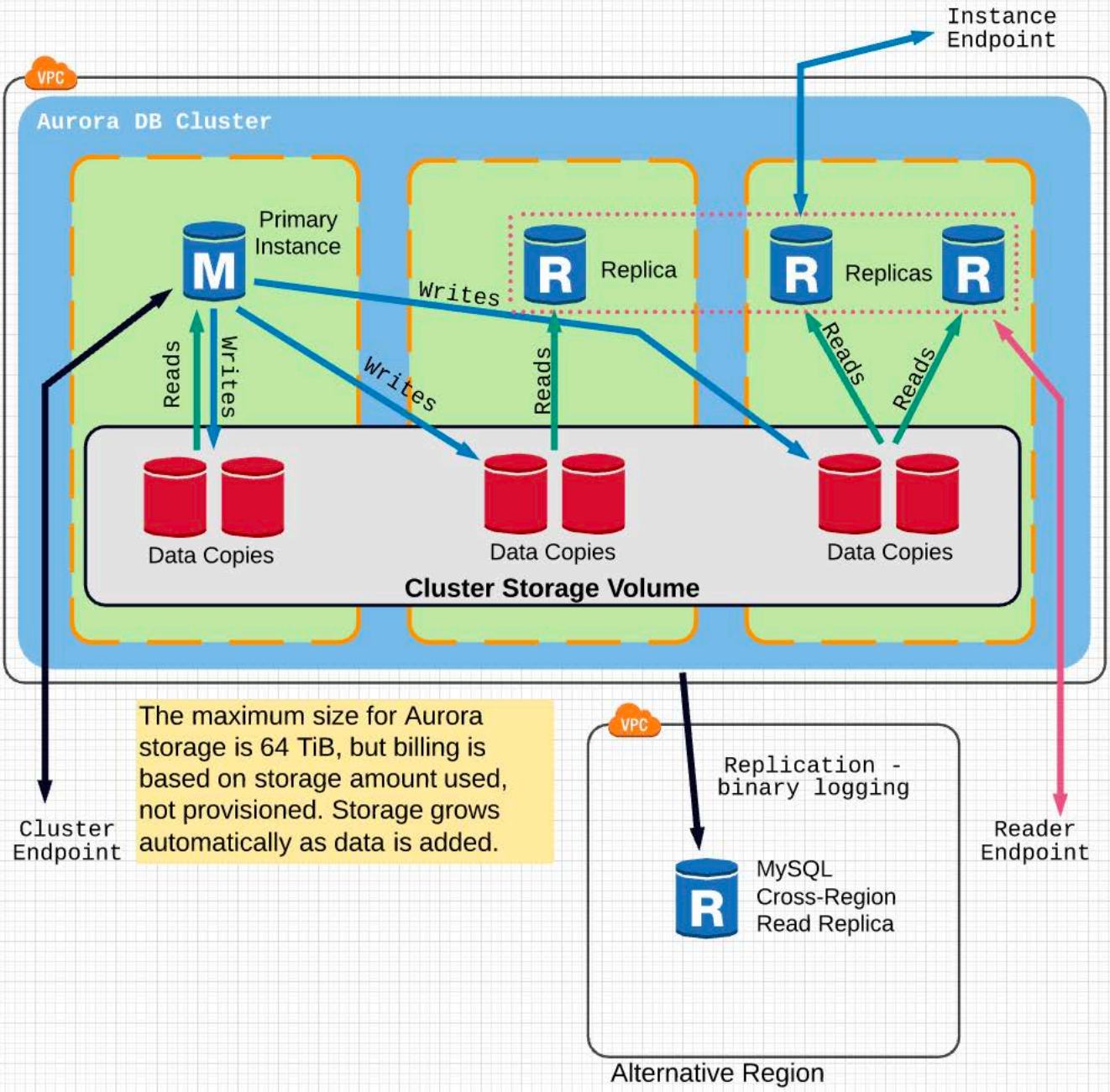
ID	Name	Color
0001	Roffle	B/W
0002	Penny	All the colors
0003	Winkie	White
0004	Truffles	Mixed

ID	Human Slave
0001	Adrian
0001	Nat
0002	Adrian
0002	Nat
0003	Adrian
0003	Nat
0004	Adrian
0004	Nat

Fixed relationships exist between tables based on table keys. Queries join tables to use information from both.

Aurora | Aurora Serverless | Aurora Global

Amazon Aurora
A fully managed relational database product that provides **MySQL** and **PostgreSQL** compatibility. Aurora has been designed to deliver up to 5x the performance of MySQL and 3x the performance of PostgreSQL while maintaining compatibility.

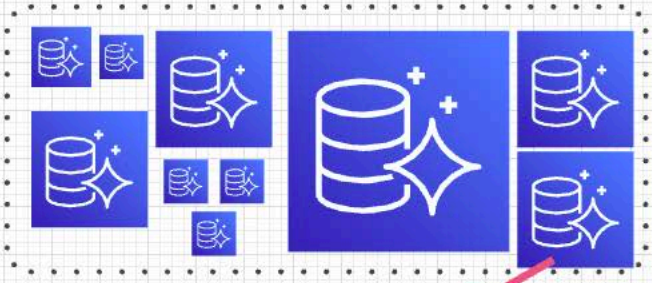


The maximum size for Aurora storage is 64 TiB, but billing is based on storage amount used, not provisioned. Storage grows automatically as data is added.

Aurora | Aurora Serverless | Aurora Global

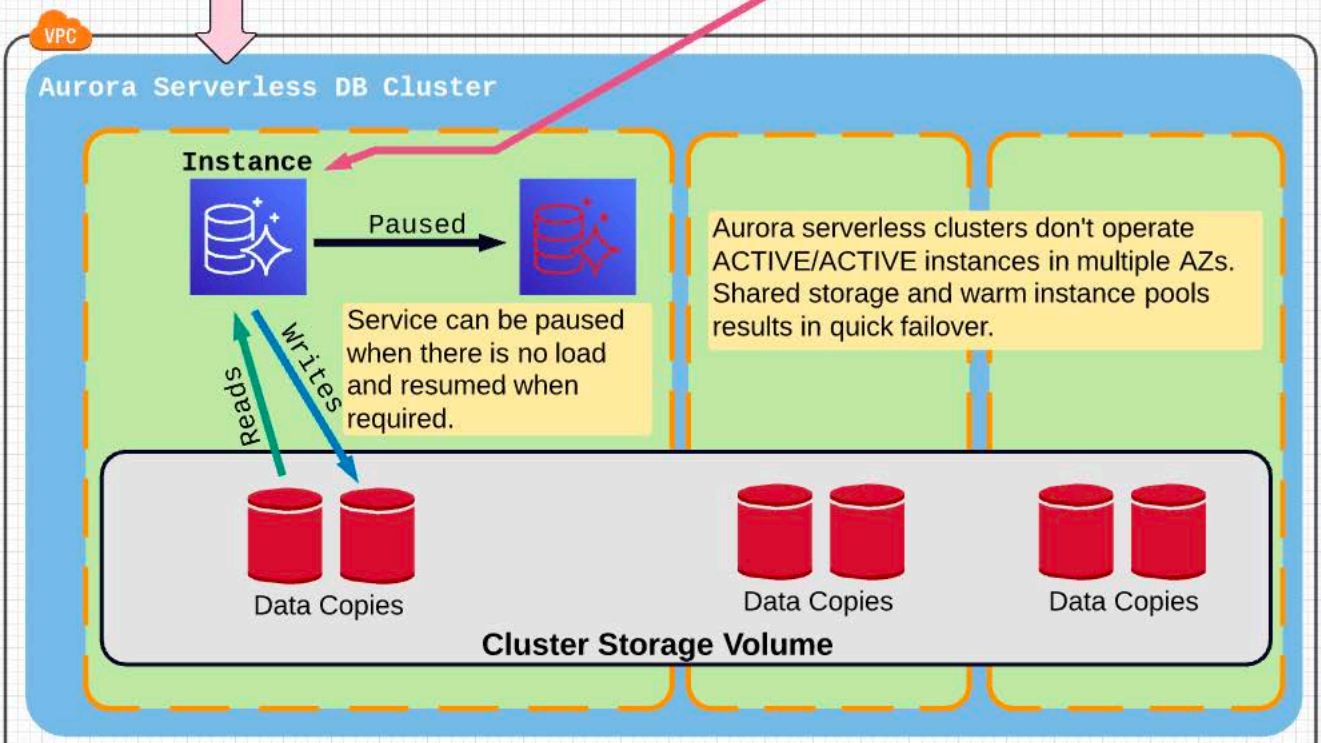
Aurora maintains a warm pool of instances (without any data) ready to swap into an existing serverless cluster as capacity is needed.

Connections are managed by a shared proxy layer that brokers connections and stops interruptions due to scaling events or failures.

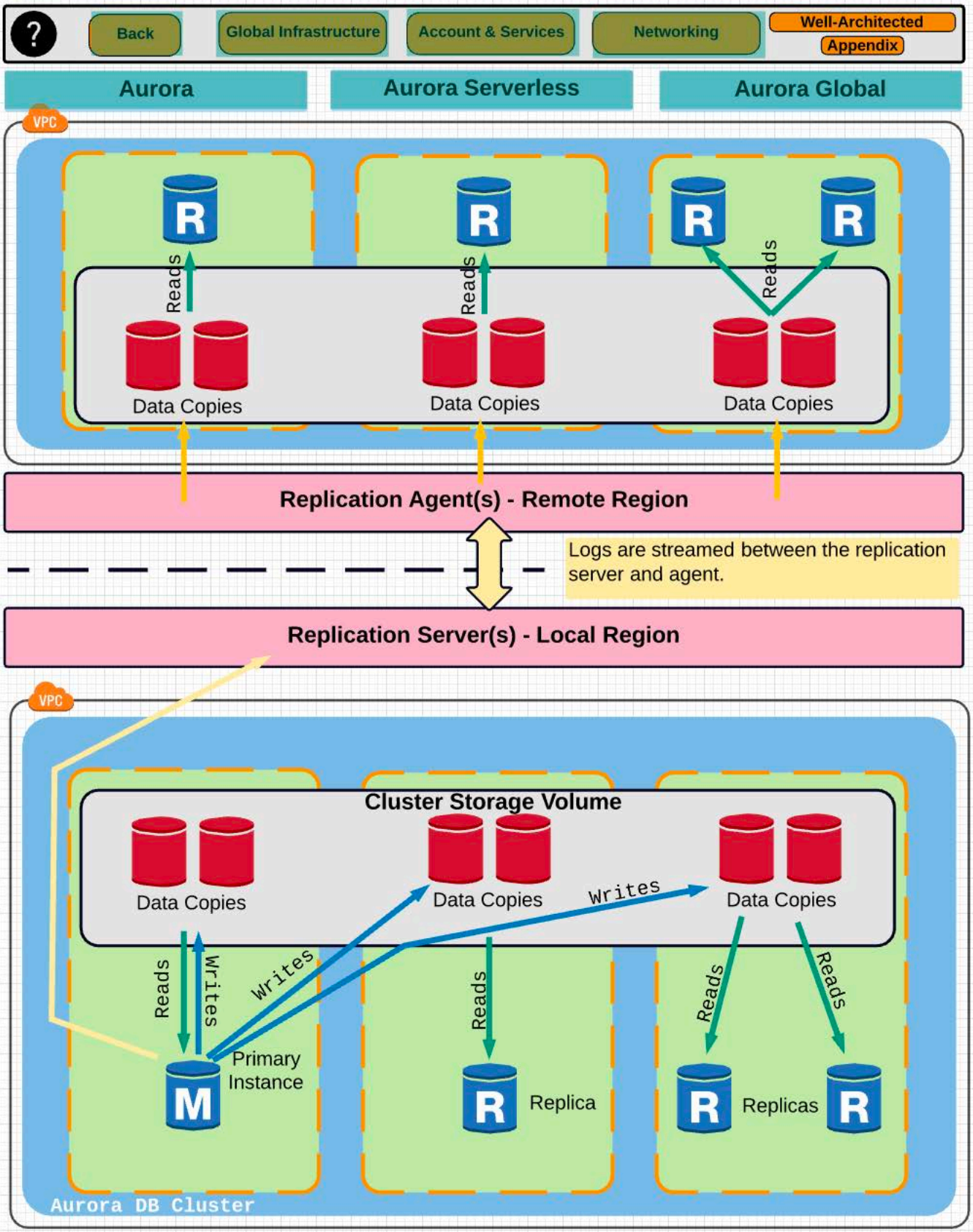


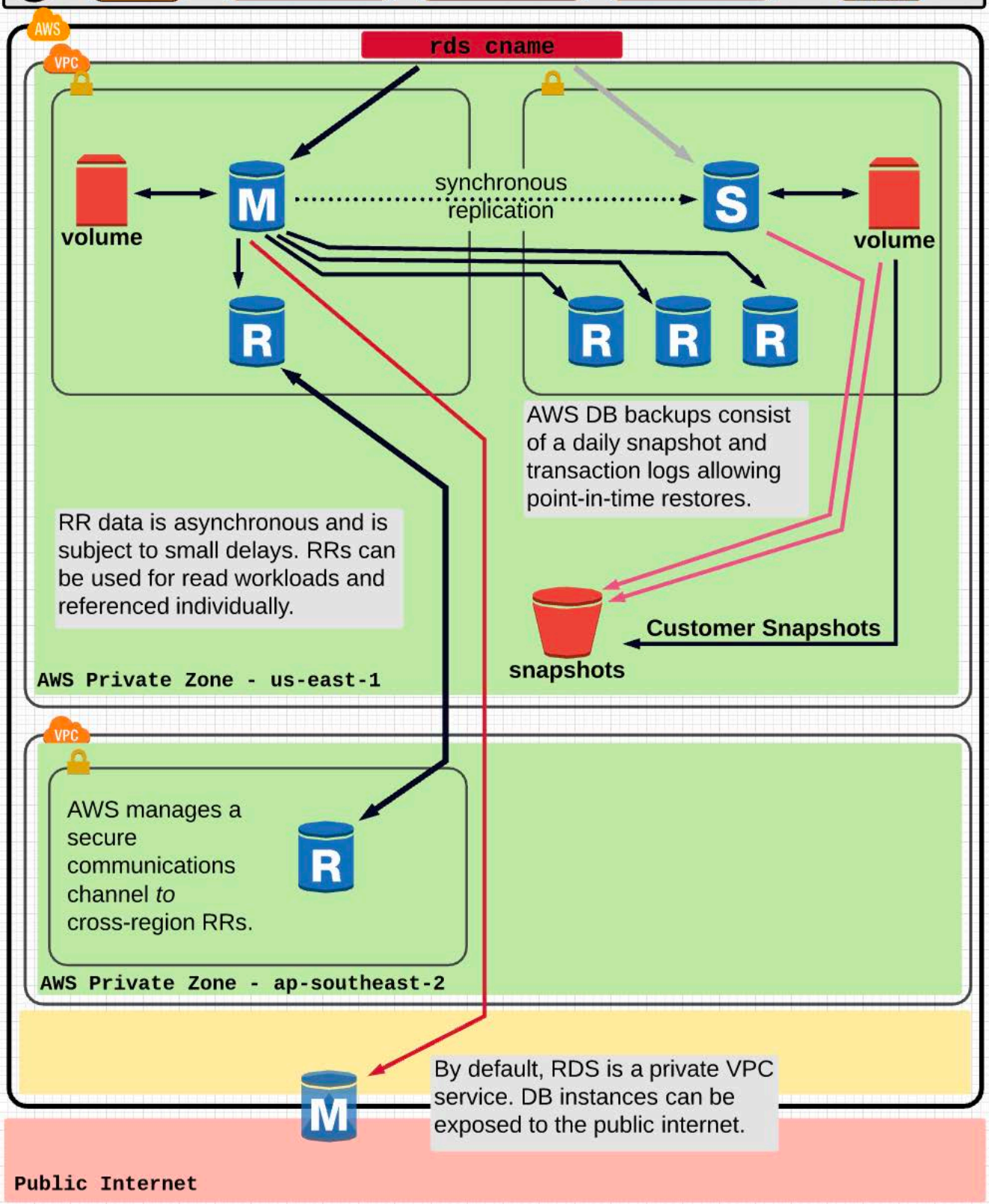
Warm Instance Pool

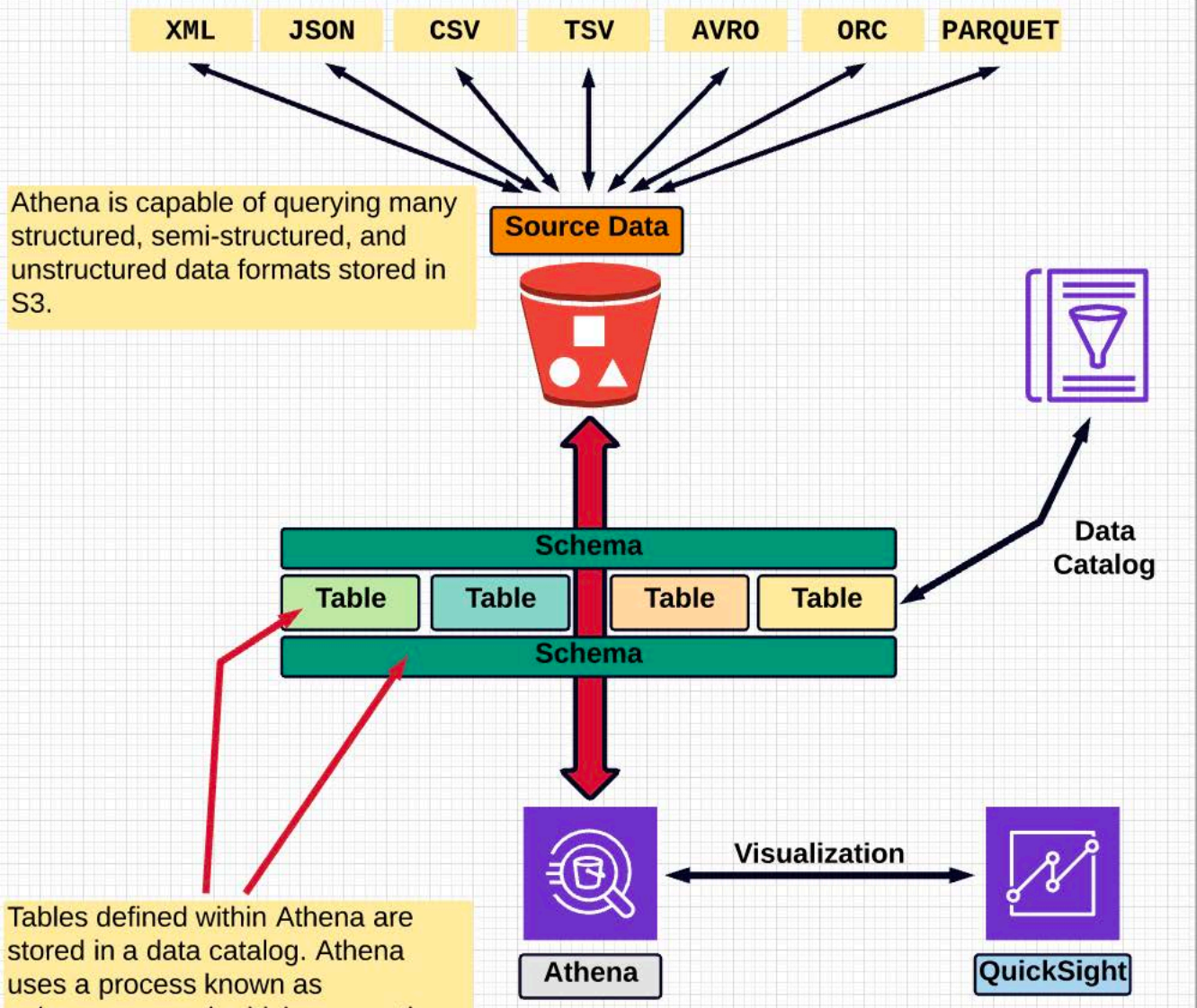
Connections to the databases can be achieved either by using traditional MySQL SDKs/CLIs or via the new AWS **DATA API**.



An Aurora Capacity Unit (ACU) is 1vCPU and 2 GB RAM. For a given cluster, a min/max size is defined. Aurora scales **up** and **down** based on CPU utilization and connections.


?
Back
Global Infrastructure
Account & Services
Networking
Well-Architected
Appendix
Aurora
Aurora Serverless
Aurora Global
VPC
R
Reads
Data Copies
Replication Agent(s) - Remote Region
Logs are streamed between the replication server and agent.
Replication Server(s) - Local Region
VPC
Cluster Storage Volume
Data Copies
Writes
Reads
Primary Instance
M
Replica
Replicas
Aurora DB Cluster


AWS
VPC
rds cname
volume
M
synchronous replication
S
volume
R
R
R
R
AWS DB backups consist of a daily snapshot and transaction logs allowing point-in-time restores.
RR data is asynchronous and is subject to small delays. RRs can be used for read workloads and referenced individually.
snapshots
Customer Snapshots
AWS Private Zone - us-east-1
VPC
AWS manages a secure communications channel *to* cross-region RRs.
R
AWS Private Zone - ap-southeast-2
M
By default, RDS is a private VPC service. DB instances can be exposed to the public internet.
Public Internet



Athena is capable of querying many structured, semi-structured, and unstructured data formats stored in S3.

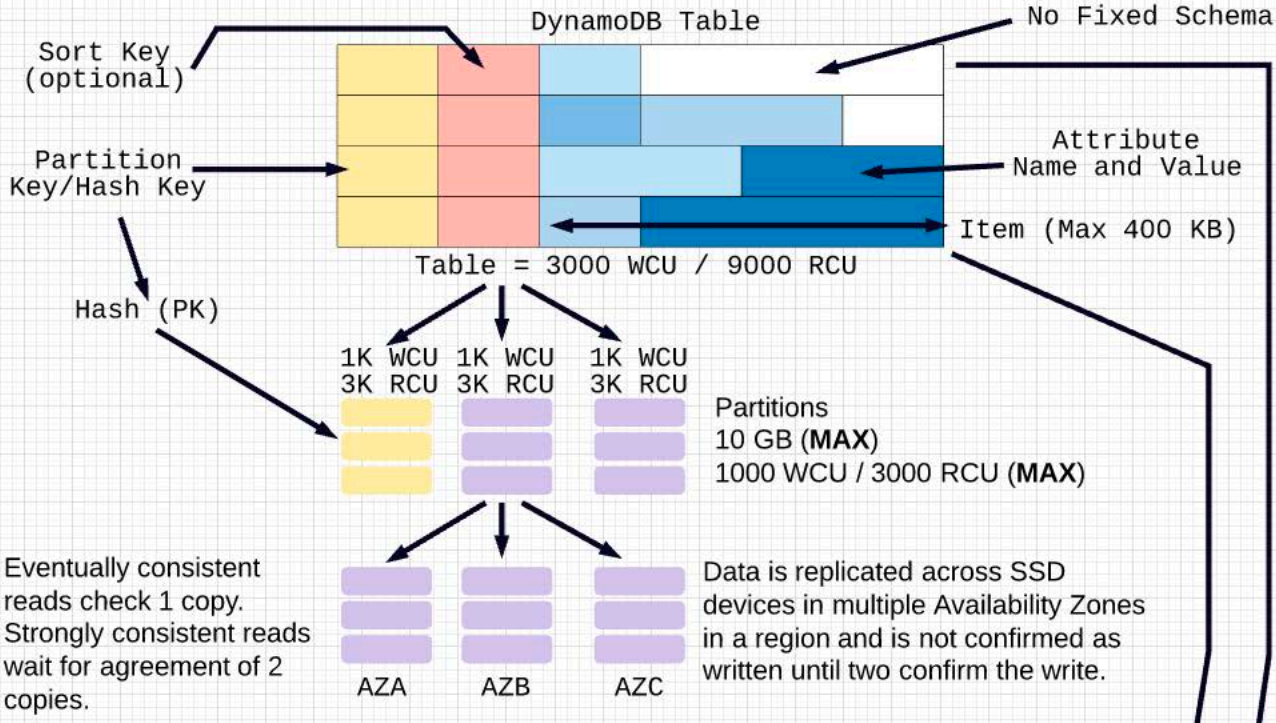
Tables defined within Athena are stored in a data catalog. Athena uses a process known as *schema-on-read*, which means the schema is overlaid on the data and used every time objects in S3 are queried.

Because of this architecture, there is no need for data loading or transformation either in advance or at query time.

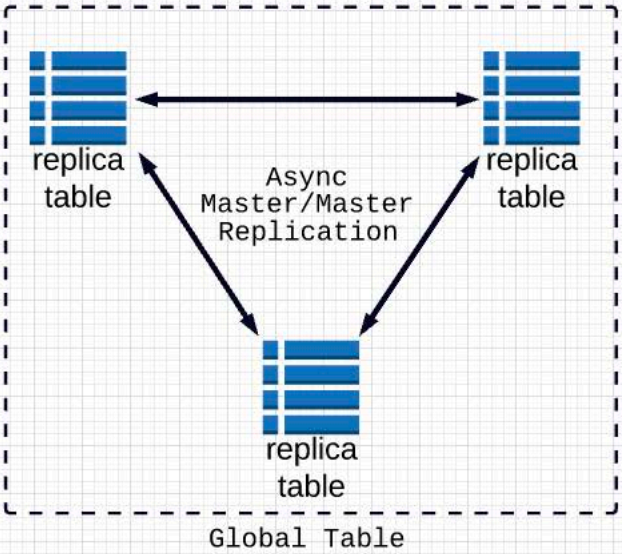
No data is modified within S3 during the query process.

Athena is a serverless query service allowing data stored in S3 to be queried using standard SQL. There is no base cost for using Athena — costs are based on the queries executed.

Athena can be used to query most AWS logging and data output types in addition to most common standard data formats.



Tables can be queried or scanned. Scans are inefficient. Querying can only be done based on partition keys (PK) and any optional sort keys (SK). Local secondary indexes (LSIs) can be added to allow queries against the same PK but alternative SKs. Global secondary indexes (GSIs) allow querying against alternative PKs and SKs.



Streams can be configured to store changes to the DynamoDB table. Streams can either list the **keys** being changed, new image, old image, or new and old images.

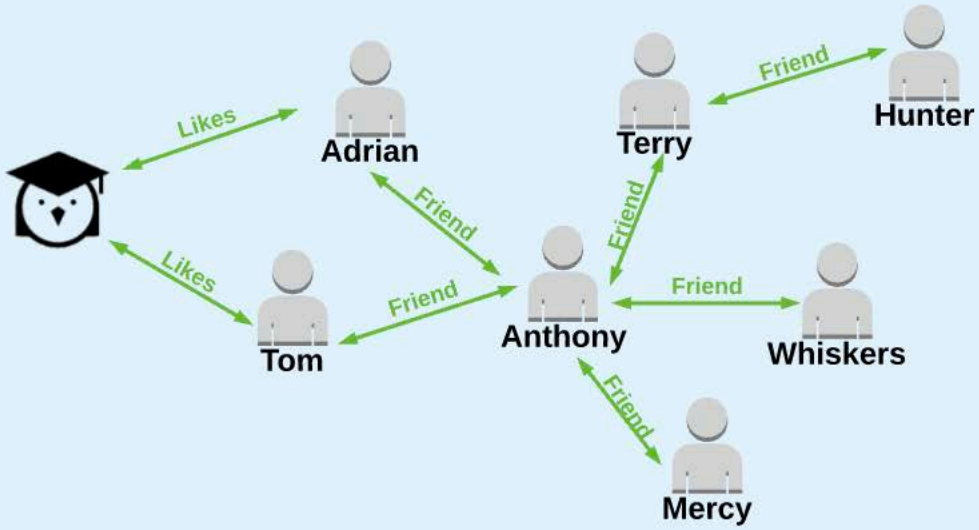
Lambda functions can be configured as triggers for when there is stream activity.

Neptune Essentials

X

- Neptune is a fully managed, NoSQL graph database service provided by AWS.
- Open-source APIs:
  - TinkerPop Gremlin — use Gremlin Console client
  - RDF SPARQL — use RDF4J Console
- Highly available
- Store up to 64 TB and up to 15 Read Replicas
- Server-side encryption

- Graph database:



- Popular use cases include:
  - Social network
  - Knowledge graphs
  - Fraud detection
  - Recommendation engine

Amazon Quantum Ledger Database is a fully immutable database using cryptographically verifiable ledger technology to track every change to data in a way that cannot be manipulated. QLDB uses a document storage model, allowing data with complex, nested structures to be stored and queried.

QLDB is an ACID-based database with strong transactional consistency. This means either an entire transaction is completed or it's fully rolled back. $47 moved from Account A to Account B must complete with the full amount moved or not at all.

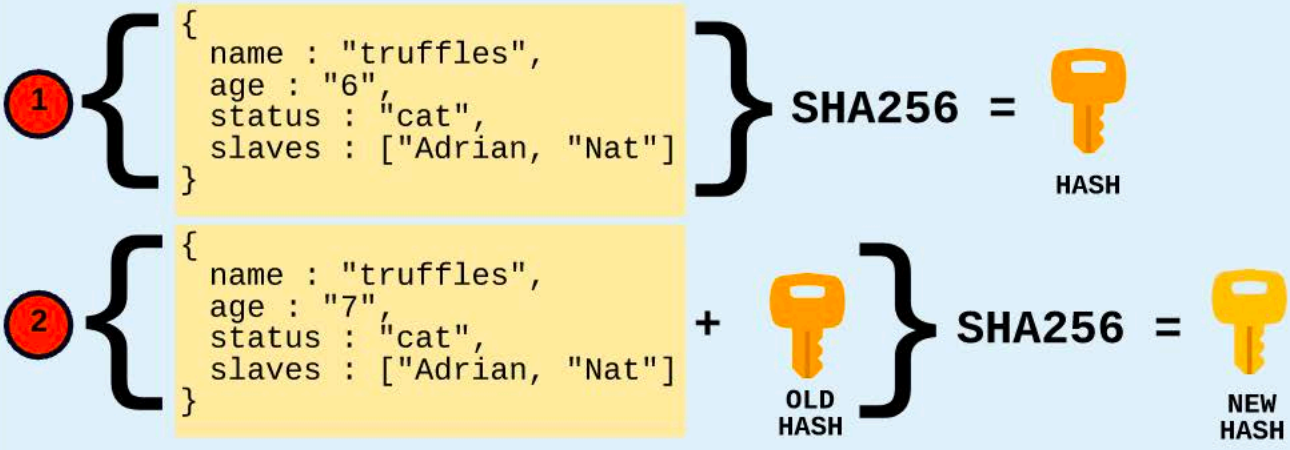
QLDB is fully serverless. It is a DBaaS product and requires no IO provisioning.

```
{
  name : "truffles",
  age : "6",
  status : "cat",
  slaves : ["Adrian, "Nat"]
}
```

SQL is supported allowing familiar queries

```
SELECT *
FROM things
where things.status = 'cat'
```

Changes in QLDB are cryptographically verifiable using a "hash-chained" journal. When data is initially added, a "hash" is created of that data. When a new version is added, a new hash is created of current data plus the old hash — this forms a chain. It makes tampering or changes immediately detectable. And the more data added, the harder it would be to "fake" this chain.



Because QLDB's data history is verifiable, it makes it ideally suited to applications where the validity of data is a priority (e.g., manufacturing, finance, healthcare, payroll, supply chain, insurance, and voting).

X

Amazon Quantum Ledger Database is a fully immutable database using cryptographically verifiable ledger technology to track every change to data in a way that cannot be manipulated. QLDB uses a document storage model, allowing data with complex, nested structures to be stored and queried.

QLDB is an ACID-based database with strong transactional consistency. This means either an entire transaction is completed or it's fully rolled back. $47 moved from Account A to Account B must complete with the full amount moved or not at all.

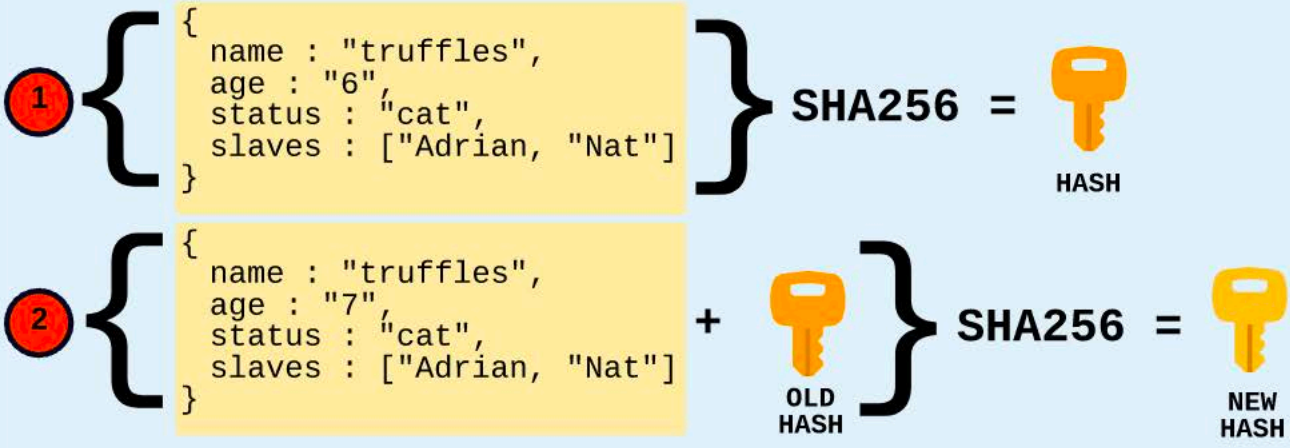
QLDB is fully serverless. It is a DBaaS product and requires no IO provisioning.

```
{
  name : "truffles",
  age : "6",
  status : "cat",
  slaves : ["Adrian, "Nat"]
}
```

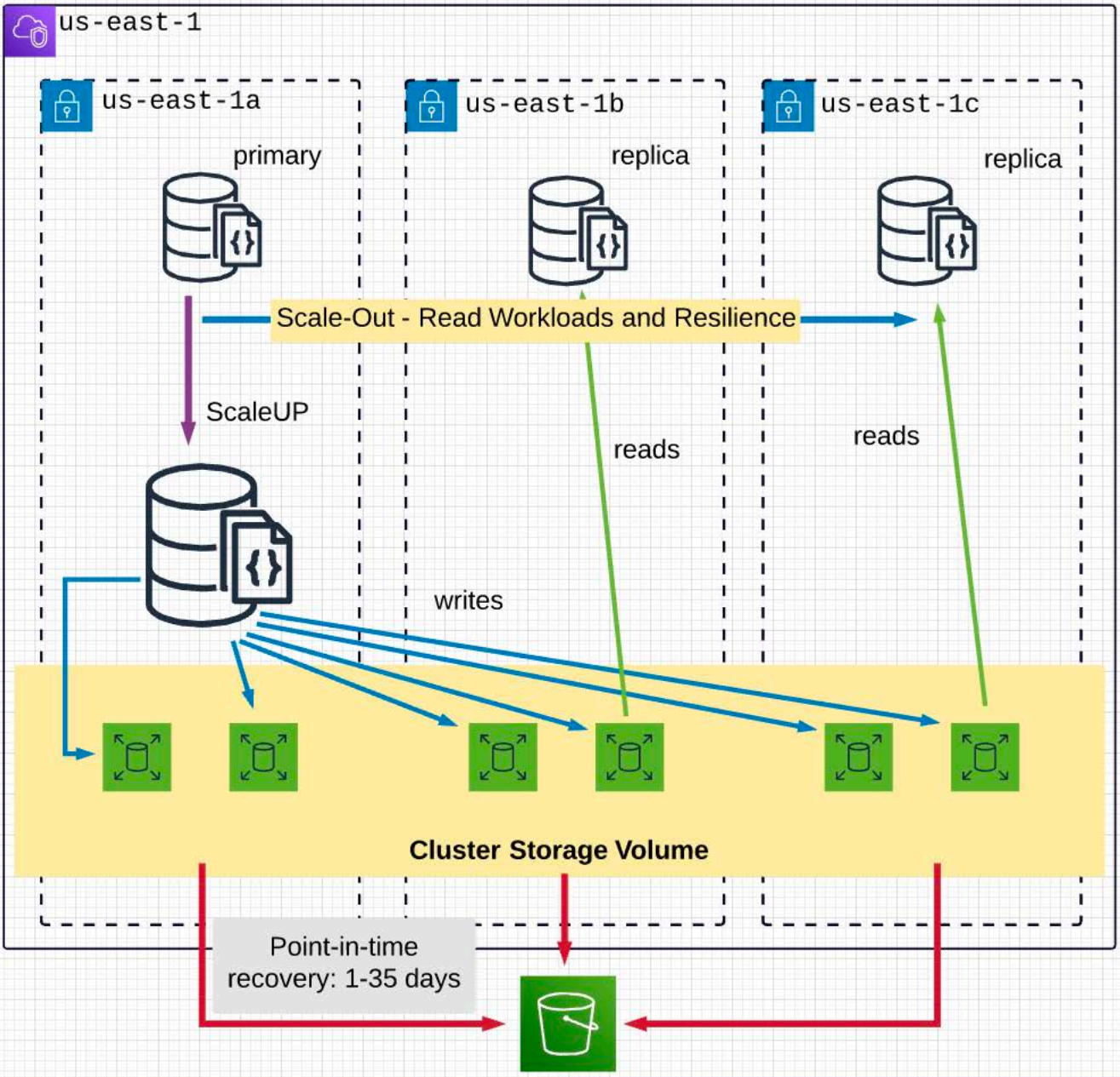
SQL is supported allowing familiar queries

```
SELECT *
FROM things
where things.status = 'cat'
```

Changes in QLDB are cryptographically verifiable using a "hash-chained" journal. When data is initially added, a "hash" is created of that data. When a new version is added, a new hash is created of current data plus the old hash — this forms a chain. It makes tampering or changes immediately detectable. And the more data added, the harder it would be to "fake" this chain.



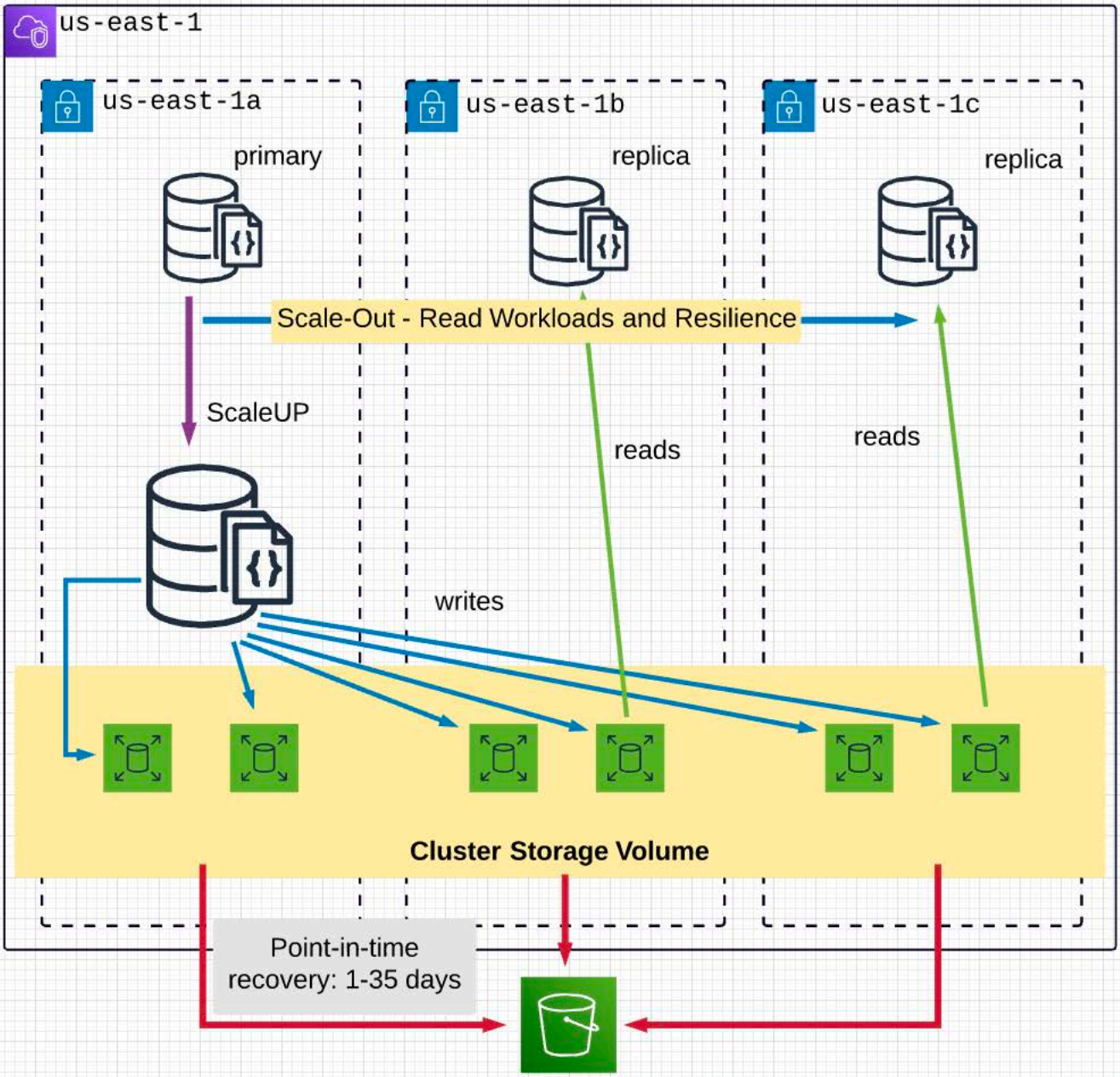
Because QLDB's data history is verifiable, it makes it ideally suited to applications where the validity of data is a priority (e.g., manufacturing, finance, healthcare, payroll, supply chain, insurance, and voting).



DocumentDB is a managed document storage database compatible with MongoDB. It supports continual backup and point-in-time recovery. It can scale horizontally for durability and read workloads, as well as vertically for write-heavy workloads.

SQL vs DocumentDB

```
Table <-> Collection
Row <-> Document
Column <-> Field
Primary Key <-> ObjectId
```



DocumentDB is a managed document storage database compatible with MongoDB. It supports continual backup and point-in-time recovery. It can scale horizontally for durability and read workloads, as well as vertically for write-heavy workloads.

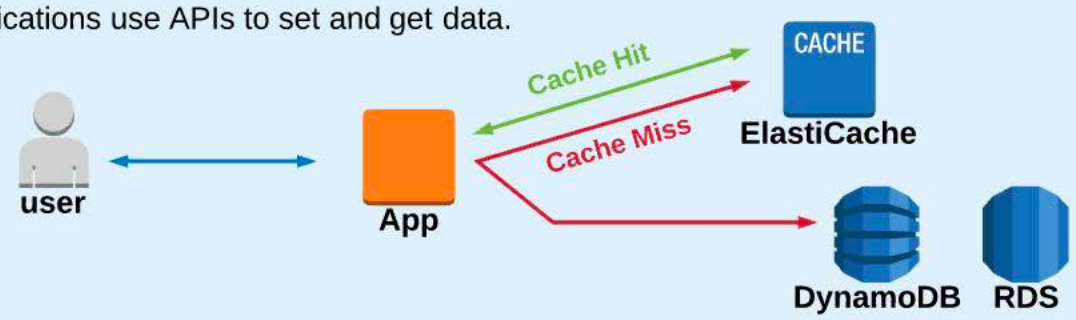
SQL vs DocumentDB

```
Table <-> Collection
Row <-> Document
Column <-> Field
Primary Key <-> ObjectId
```

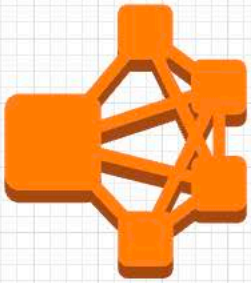
ElastiCache

X

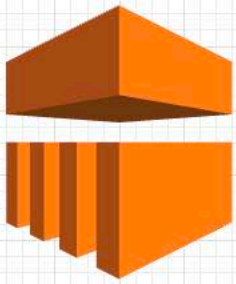
- ElastiCache is a fully managed, in-memory data store and cache service.
- ElastiCache is used to improve database performance by caching results of queries that are made to a database — thus, improving speed and reducing database load.
- ElastiCache provisions node clusters that you can scale.
- ElastiCache can host web sessions.
- Applications use APIs to set and get data.



	Memcached	Redis (cluster mode disabled)	Redis (cluster mode enabled)
Engine versions	1.4.x	2.8.x and later	3.2.x and later
Data types	Simple ‡	2.8.x - Complex * Complex †	3.2.x and later - Complex †
Data partitioning	Yes	No	Yes
Cluster is modifiable	Yes	Yes	3.2.10 and later - Limited
Online resharding	No	No	3.2.10 and later
Encryption	No	3.2.6, 4.0.10 and later	3.2.6, 4.0.10 and later
Compliance certifications			
Compliance Certification FedRAMP	No	3.2.6, 4.0.10 and later	3.2.6, 4.0.10 and later
HIPAA	No	3.2.6, 4.0.10 and later	3.2.6, 4.0.10 and later
PCI DSS	No	3.2.6, 4.0.10 and later	3.2.6, 4.0.10 and later
Multi-threaded	Yes	No	No
Node type upgrade	No	Yes	No
Engine upgrading	Yes	Yes	Yes
High availability (replication)	No	Yes	Yes
Automatic failover	No	Optional	Required
Pub/Sub capabilities	No	Yes	Yes
Sorted sets	No	Yes	Yes
Backup and restore	No	Yes	Yes
Geospatial indexing	No	2.8.x - No 3.2.x and later - Yes	Yes

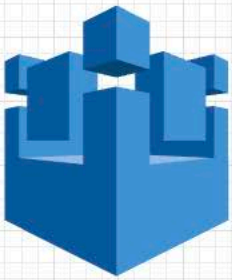


MapReduce



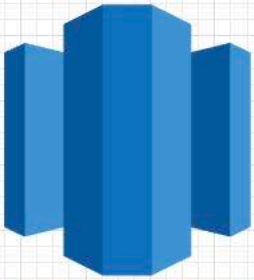
EMR

Kinesis



IoT

QuickSight

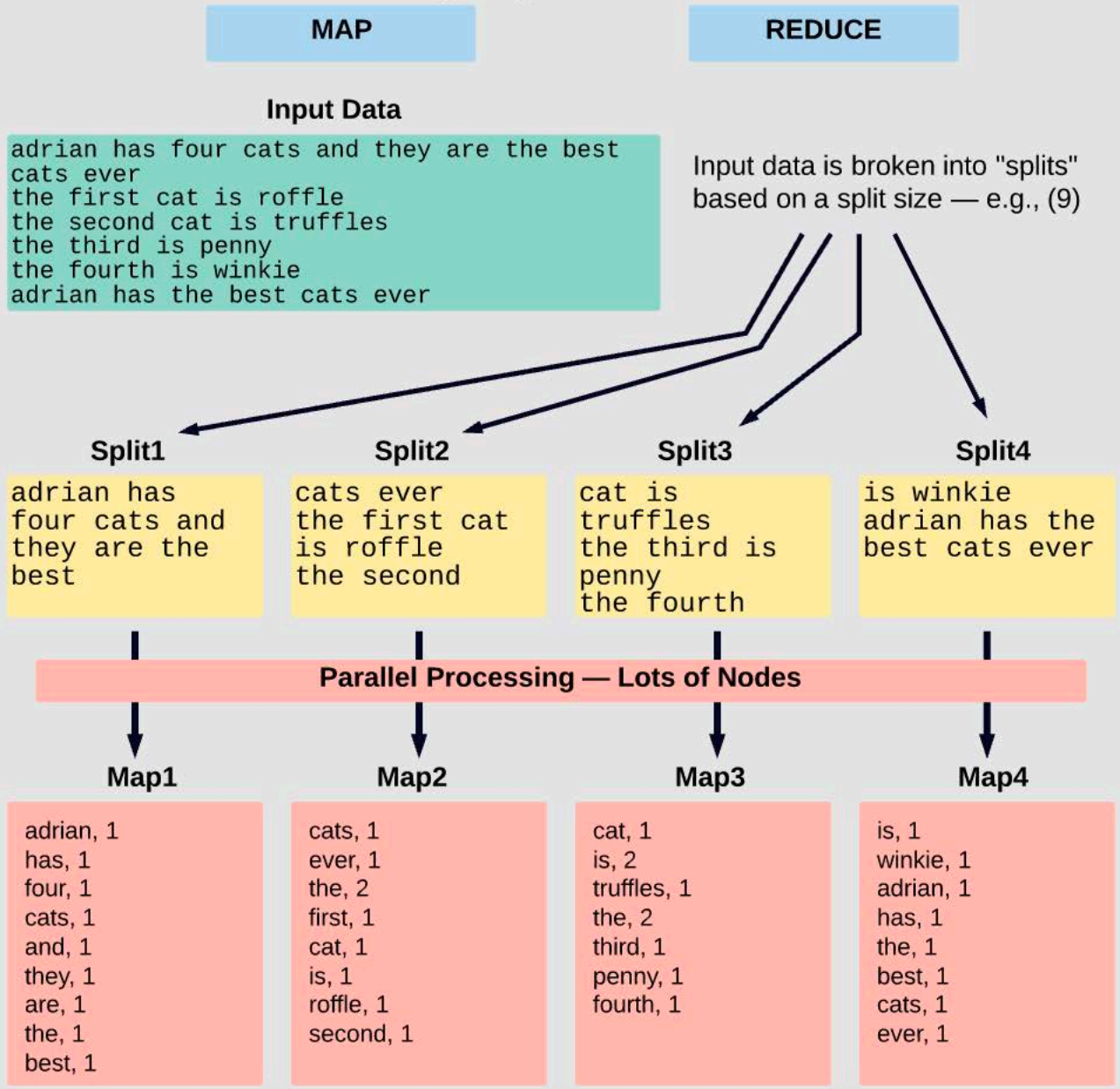


Redshift

Elasticsearch

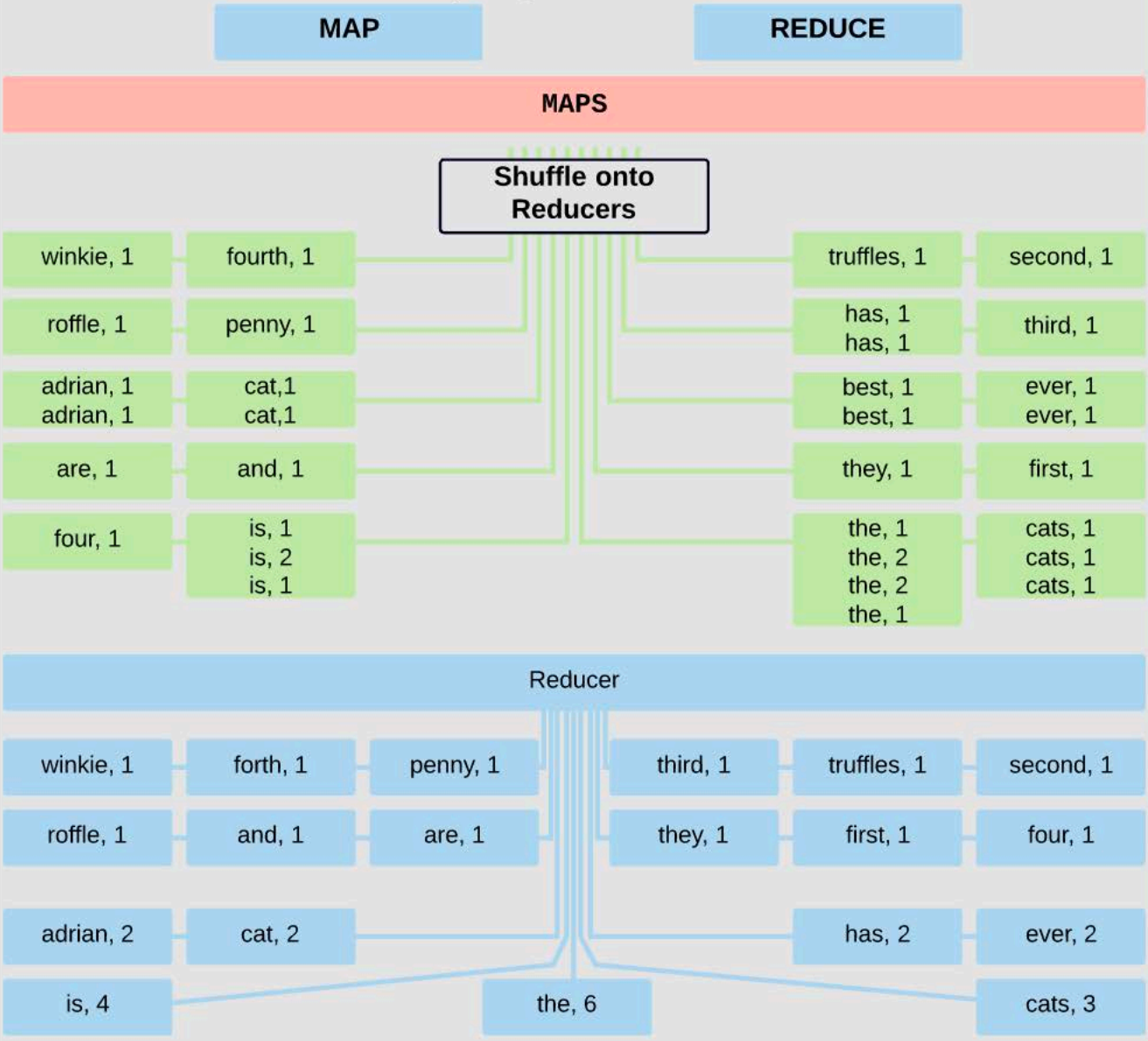
MapReduce

MapReduce is a data analysis architecture capable of processing huge quantities of data in a parallel way. It's suitable for data that can be modularized with the same basic operations executed at scale. MapReduce is a process with two main stages: **map** and **reduce**. At a high level, the idea is to split the data into manageable bits, perform operations at scale on those bits, and then recombine for reporting.



MapReduce

MapReduce is a data analysis architecture capable of processing huge quantities of data in a parallel way. It's suitable for data that can be modularized with the same basic operations executed at scale. MapReduce is a process with two main stages: **map** and **reduce**. At a high level, the idea is to split the data into manageable bits, perform operations at scale on those bits, and then recombine for reporting.



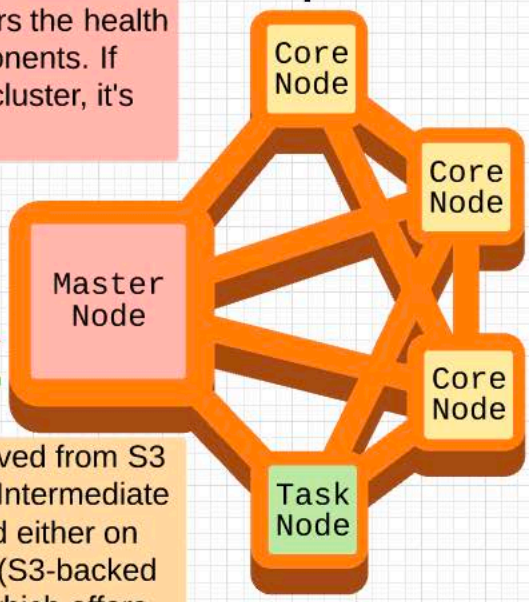
winkie 1, fourth 1, truffles 1, second 1, roffle 1, penny 1, has 2, third 1, adrian 2, cat 2, best 2, ever 2, are 1, and 1, they 1, first 1, four 1, is 4, the 6, cats 3

AWS

The master node manages the cluster. It acts as the HDFS name node, it controls the distribution of workloads, and monitors the health of all the cluster components. If you need to SSH to a cluster, it's via the master node.

EMR clusters can have zero or more core nodes that are managed by the master node. They act as HDFS data nodes, run the task tracker, and can execute mapping and reduce tasks inside the cluster.

Spot instances *can* be used as core nodes but risk HDFS data loss if they fail.



Data can be retrieved from S3 and output to S3. Intermediate data can be stored either on HDFS or EMRFS (S3-backed cluster storage), which offers persistence beyond the lifetime of the cluster and resilience from core node failure.

Task nodes are optional in an EMR cluster — they can be used to execute tasks but don't have any involvement in HDFS cluster storage.

Task nodes are ideal when spot instances are used. The processes managing task execution run on core nodes — the compute can occur on task nodes.

VPC

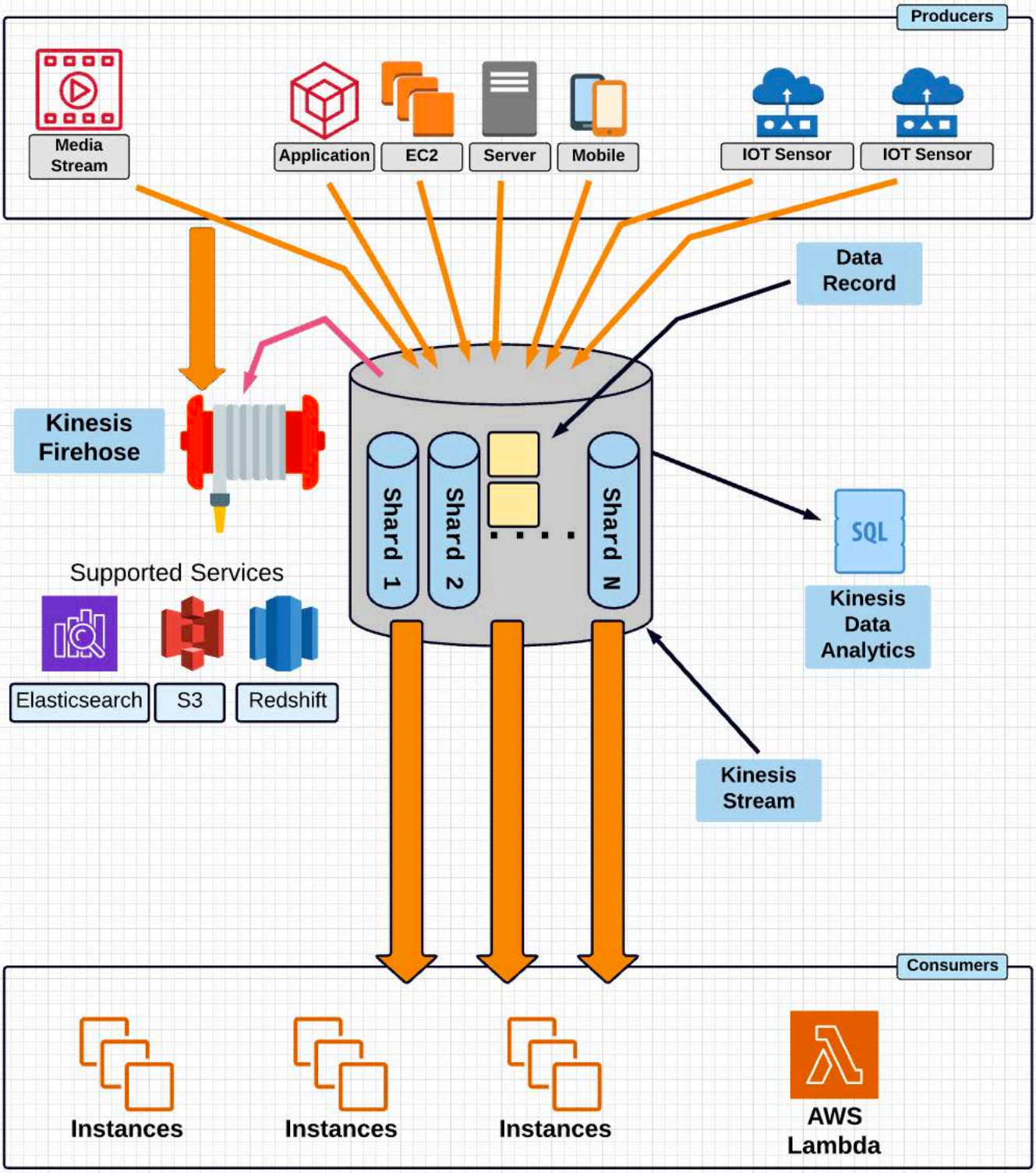
Public or Private Subnet

By default, all cluster instances are within the same Availability Zone to allow fast and efficient intra-cluster communications.

security group

Master / Core / Task Nodes

Kinesis allows the collection of large streams of data records and live video streams in real time. Data can be consumed by multiple consumers .



Kinesis Data Firehose

X

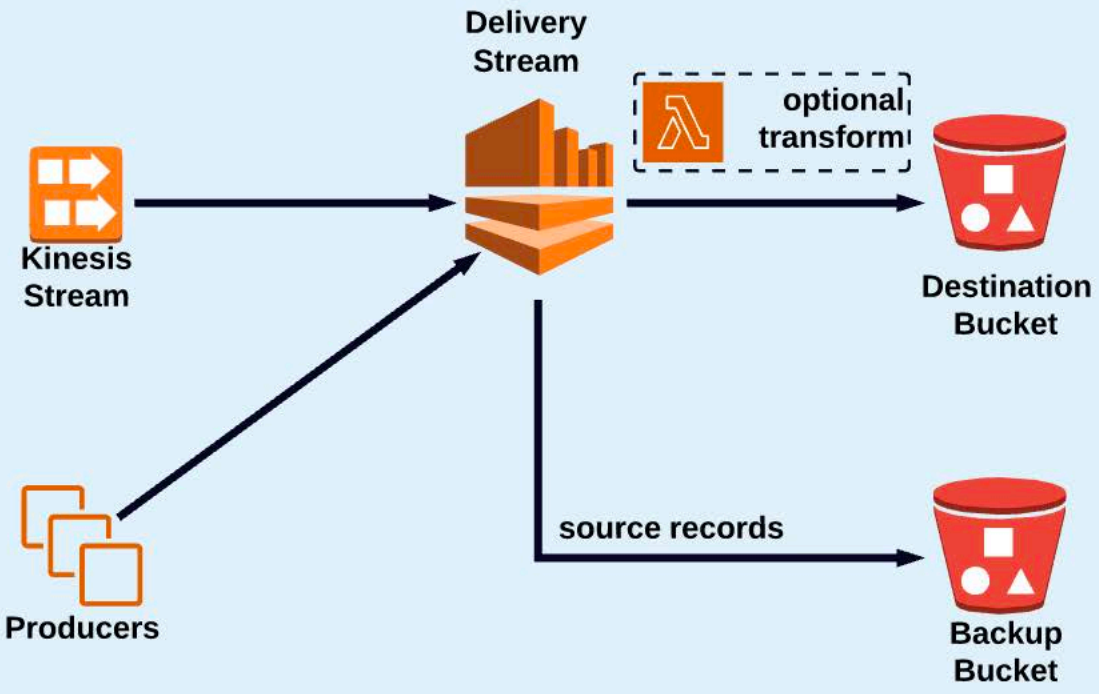
Data Firehose is a managed service that can deliver real-time data to supported destinations. It consumes data from either a Kinesis stream or traditional producers.

Firehose currently supports a number of destinations, including:

- Simple Storage Service (S3)
- Redshift
- Elasticsearch
- Splunk

It also allows for data transformation during the movement between producer and destination using Lambda functions. Examples include:

- Apache log to JSON or CSV
- Syslog to JSON or CSV



X

Kinesis Stream

A Kinesis data stream can be used to collect, process, and analyze a large amount of incoming data. A data stream is accessible from inside AWS VPCs or from the public internet by an unlimited number of producers.

Kinesis streams can scale from low amounts of incoming data to an unlimited level of load using a multi-shard architecture.

Kinesis streams scale to meet capacity requirements and include storage for all incoming data for the stream retention period — 24 hours is standard, with the ability to increase to seven days for an additional charge.

Data Records (see below) are added by **producers** to a Kinesis stream and can be read by **consumers**.

Kinesis Shard

Kinesis shards are how Kinesis streams scale. A stream has at least one shard that provides 1 MiB of ingestion capacity and 2 MiB of consumption capacity. Shards can be added to streams to scale the performance on that stream.

shards_required = max (ingestion_in_KiB/1024, read_requirements_in_KiB/2048)

Per producer bandwidth = stream bandwidth / number of producers
Per consumer bandwidth = stream bandwidth / number of consumers (* **enhanced fanout**)

1,000 written data records per second per shard - 1 MiB per record.

Kinesis Data Record

A data record is the basic entity written to and read from Kinesis streams. A data record consists of a **sequence number**, a **partition key**, and a **data blob.** The data blob can be up to 1 MB in size and is not altered by Kinesis in any way. The sequence number is allocated by Kinesis and is unique within its shard.

X

Kinesis Stream

A Kinesis data stream can be used to collect, process, and analyze a large amount of incoming data. A data stream is accessible from inside AWS VPCs or from the public internet by an unlimited number of producers.

Kinesis streams can scale from low amounts of incoming data to an unlimited level of load using a multi-shard architecture.

Kinesis streams scale to meet capacity requirements and include storage for all incoming data for the stream retention period — 24 hours is standard, with the ability to increase to seven days for an additional charge.

Data Records (see below) are added by **producers** to a Kinesis stream and can be read by **consumers**.

Kinesis Shard

Kinesis shards are how Kinesis streams scale. A stream has at least one shard that provides 1 MiB of ingestion capacity and 2 MiB of consumption capacity. Shards can be added to streams to scale the performance on that stream.

shards_required = max (ingestion_in_KiB/1024, read_requirements_in_KiB/2048)

Per producer bandwidth = stream bandwidth / number of producers
Per consumer bandwidth = stream bandwidth / number of consumers (* **enhanced fanout**)

1,000 written data records per second per shard - 1 MiB per record.

Kinesis Data Record

A data record is the basic entity written to and read from Kinesis streams. A data record consists of a **sequence number**, a **partition key**, and a **data blob.** The data blob can be up to 1 MB in size and is not altered by Kinesis in any way. The sequence number is allocated by Kinesis and is unique within its shard.

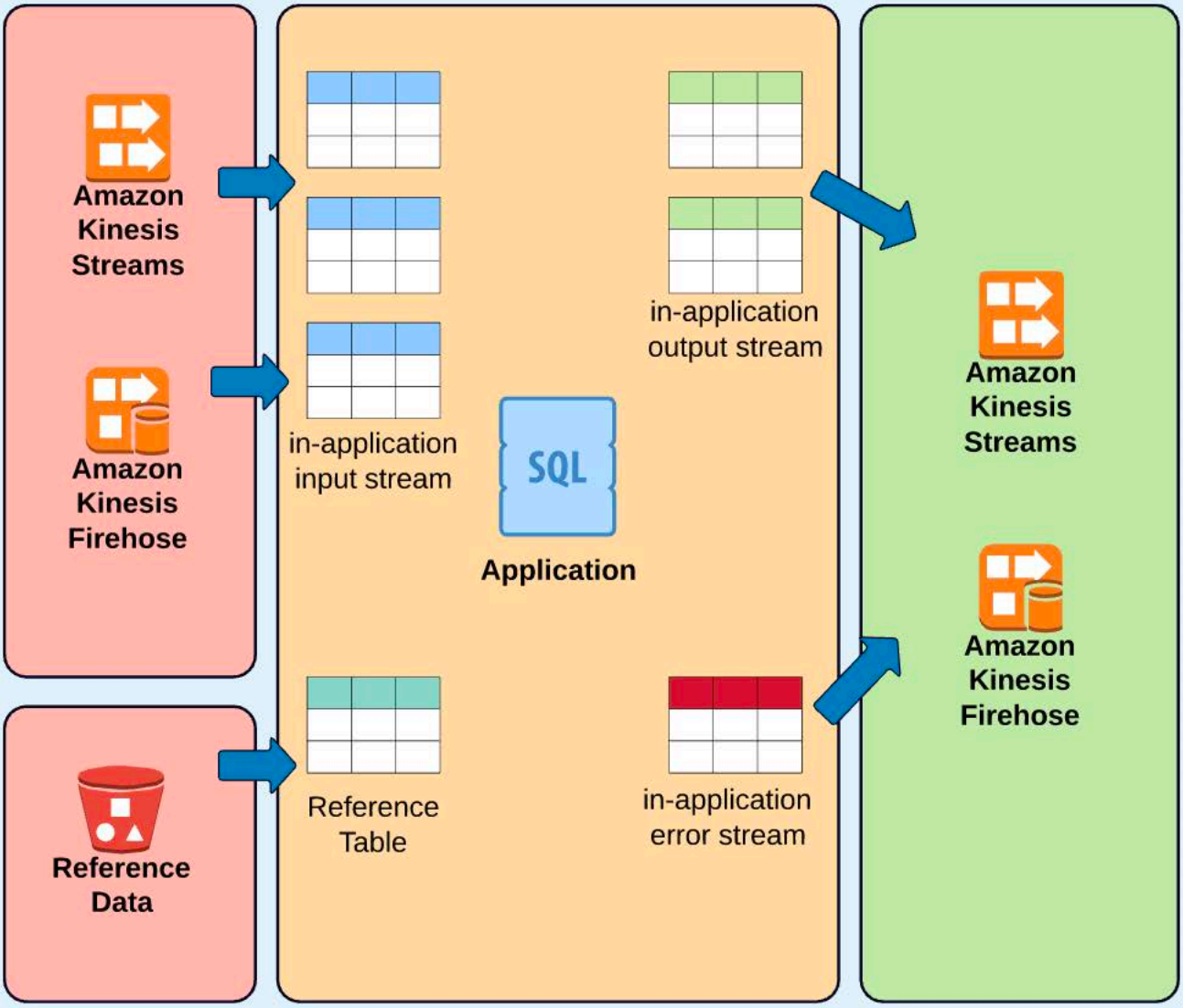
Kinesis Data Analytics:

X

Kinesis Data Analytics for SQL applications is a Kinesis feature allowing real-time processing and analysis of stream data using standard Structured Query Language (SQL).

Data Analytics can be used for a few types of scenarios:

- Time-series analytics
- Real-time dashboards
- Real-time metrics



X

Kinesis Producers

A Kinesis producer is any device that puts **data records** into a Kinesis stream. Examples of producers include:

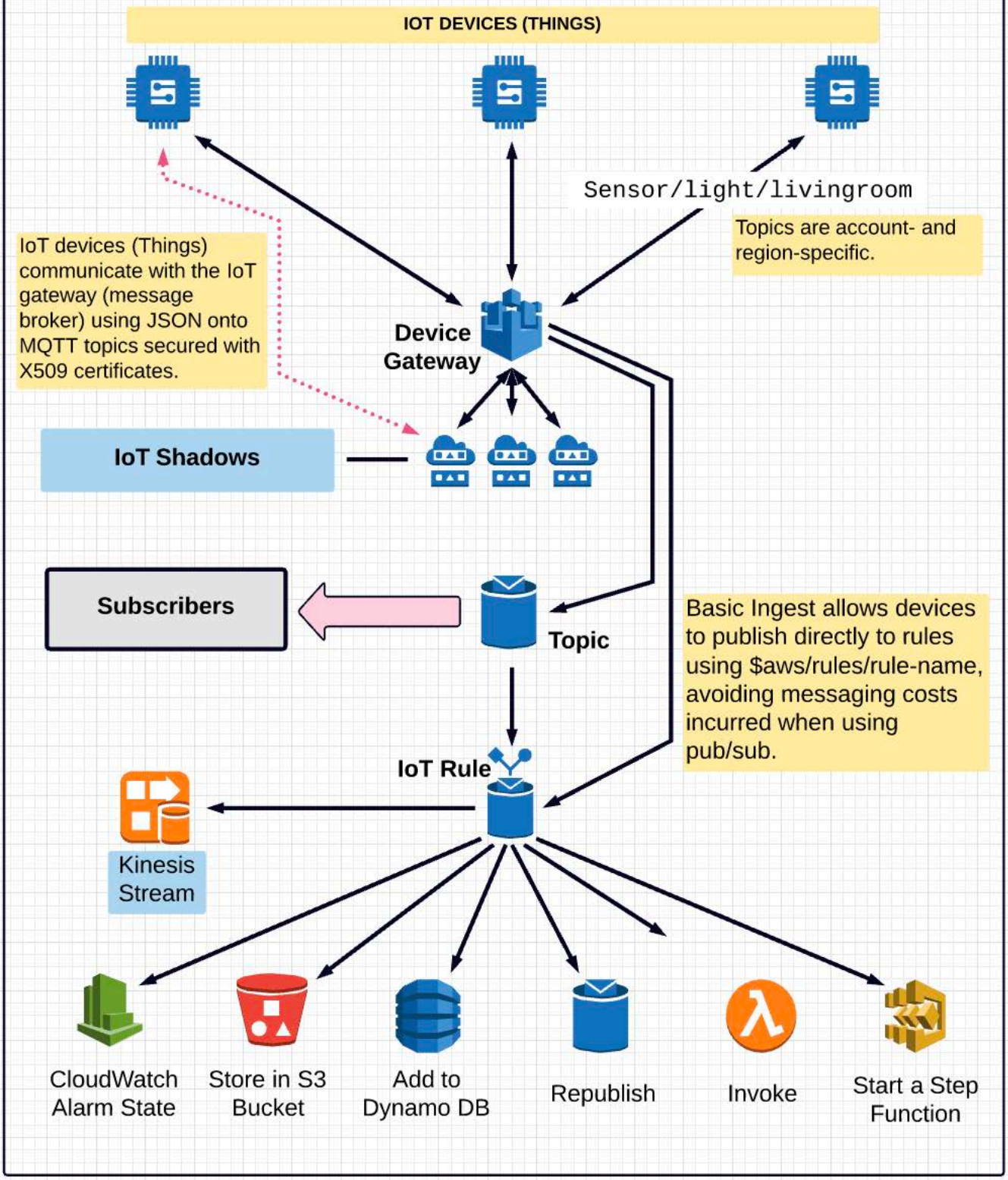
- IoT devices
- Mobile applications
- Applications
- EC2 instance
- On-premises server

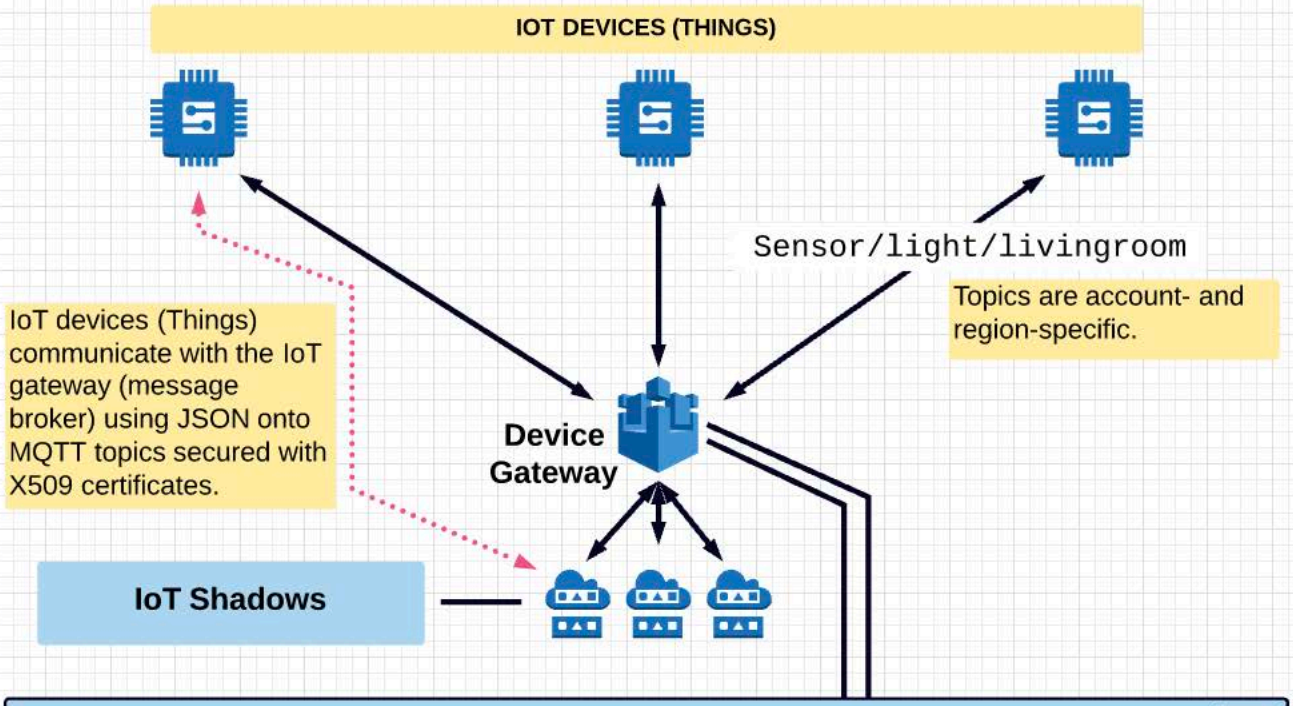
A producer puts data into a stream by specifying the name of the stream, a partition key, and the data to be added. The partition key allows Kinesis to select the **stream** used to ingest the data.

Kinesis Consumers

A consumer is any entity that consumes records from a Kinesis stream. Examples of consumers include:

- EC2 instances running the Kinesis Consumer Library (KCL)
- Lambda functions set to invoke based on stream data records
- Kinesis Firehose


IOT DEVICES (THINGS)
Sensor/light/livingroom
Topics are account- and region-specific.
IoT devices (Things) communicate with the IoT gateway (message broker) using JSON onto MQTT topics secured with X509 certificates.
Device Gateway
IoT Shadows
Subscribers
Topic
Basic Ingest allows devices to publish directly to rules using $aws/rules/rule-name, avoiding messaging costs incurred when using pub/sub.
IoT Rule
Kinesis Stream
CloudWatch Alarm State
Store in S3 Bucket
Add to Dynamo DB
Republish
Invoke
Start a Step Function



IoT Device Shadows

X

Communication between an IoT device and the IoT gateway may be sporadic and unreliable. IoT maintains a device shadow that stores the state of the device as last reported. The shadow can be queried rather than communicating directly. The device shadow can also be written to, for example, to set a desired state, which the device will receive the next time it communicates with the gateway.

```
$aws/things/<thing-name>/shadow/update
$aws/things/<thing-name>/shadow/update/documents
```

```
{
  "reported" : {
    "thingname" : "catfood_bowl",
    "location" : {
      "lat" : 64.7511
      "lon" : -147.3494
    },
    "status" : "full"
  }
}
```

Amazon QuickSight

Amazon QuickSight is a service that provides data visualization. It can integrate with other services to obtain data and present it in a visual and interactive way.

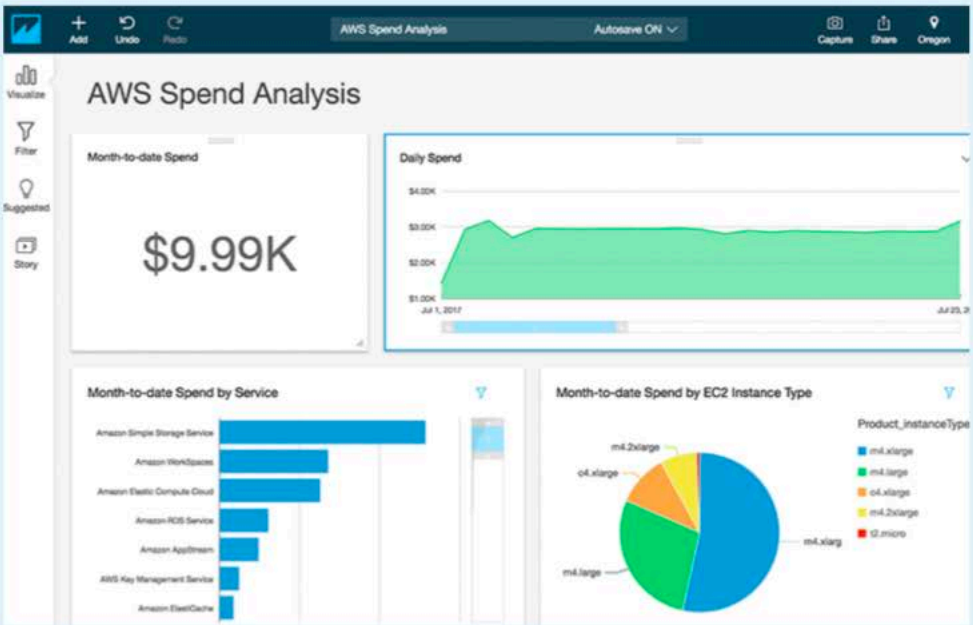
QuickSight is ideally suited when you need to collate data from multiple sources, interact with data visually, or adjust data in an interactive way.

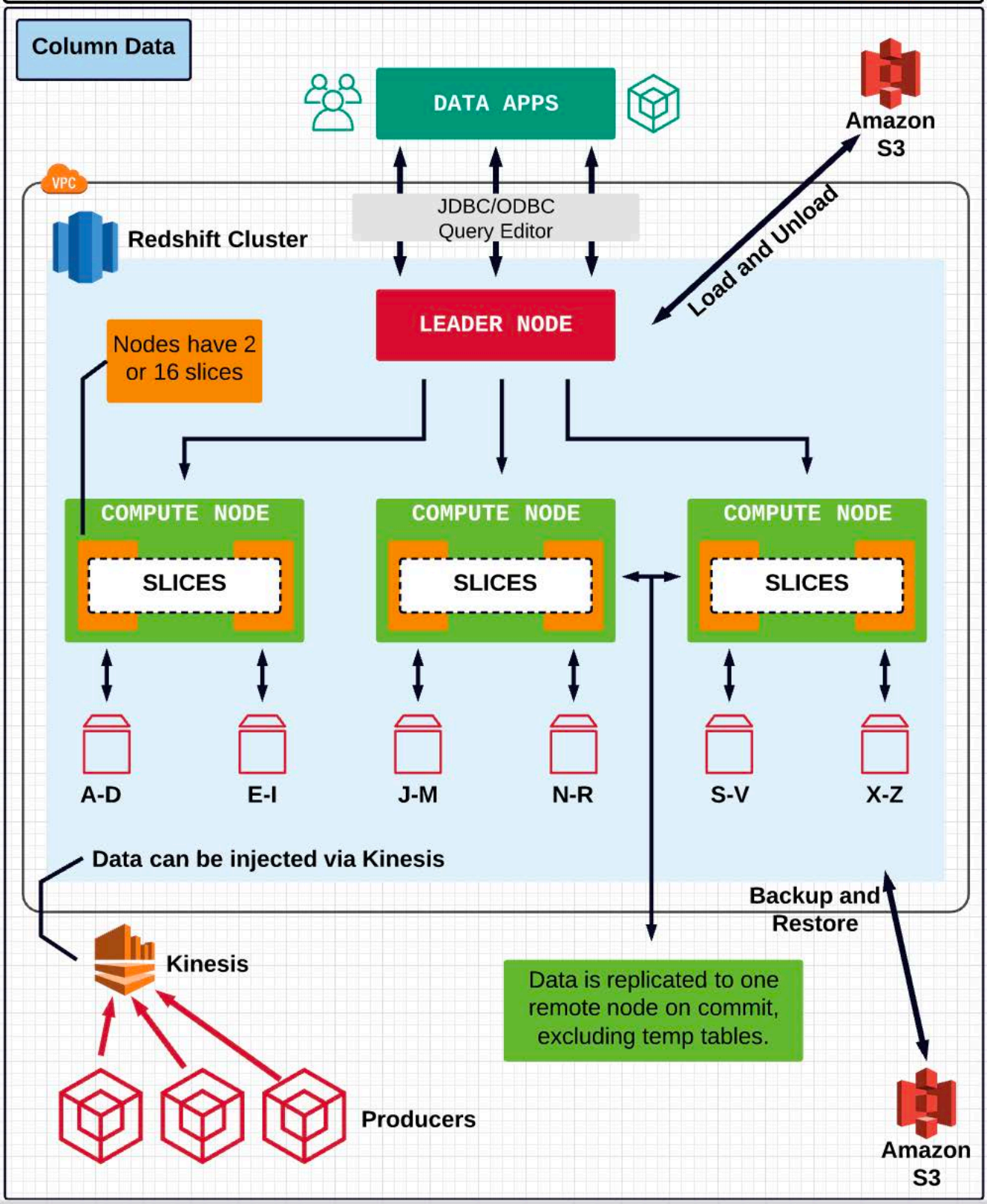
This could be dashboards for a business process, monitoring and metrics for an infrastructure platform, or a visual indication of a fleet of IoT devices.

QuickSight can interact with a large number of data sources, including:

- Amazon Athena
- Amazon Aurora
- Amazon Redshift
- Amazon S3
- Apache Spark 2.0
- MariaDB
- MS SQL 2012+
- MySQL 5.1+

QuickSight can also interact with unsupported data sources by using supported integrations as an intermediary.




Column Data
DATA APPS
Amazon S3
VPC
Redshift Cluster
JDBC/ODBC
Query Editor
Load and Unload
LEADER NODE
Nodes have 2 or 16 slices
COMPUTE NODE
SLICES
COMPUTE NODE
SLICES
COMPUTE NODE
SLICES
A-D
E-I
J-M
N-R
S-V
X-Z
Data can be injected via Kinesis
Backup and Restore
Kinesis
Data is replicated to one remote node on commit, excluding temp tables.
Producers
Amazon S3

```
select max(Age), min(Age) from cats
```

Column Based

CATID	Name	Age
0001	Roffle	6
0002	Penny	5
0003	Winkie	3
0004	Truffles	7
...	...	...
1000000000000001	Mr Wiggles	57

Row Based

CATID	Name	Age
0001	Roffle	6
0002	Penny	5
0003	Winkie	3
0004	Truffles	7
...	...	...
1000000000000001	Mr Wiggles	57

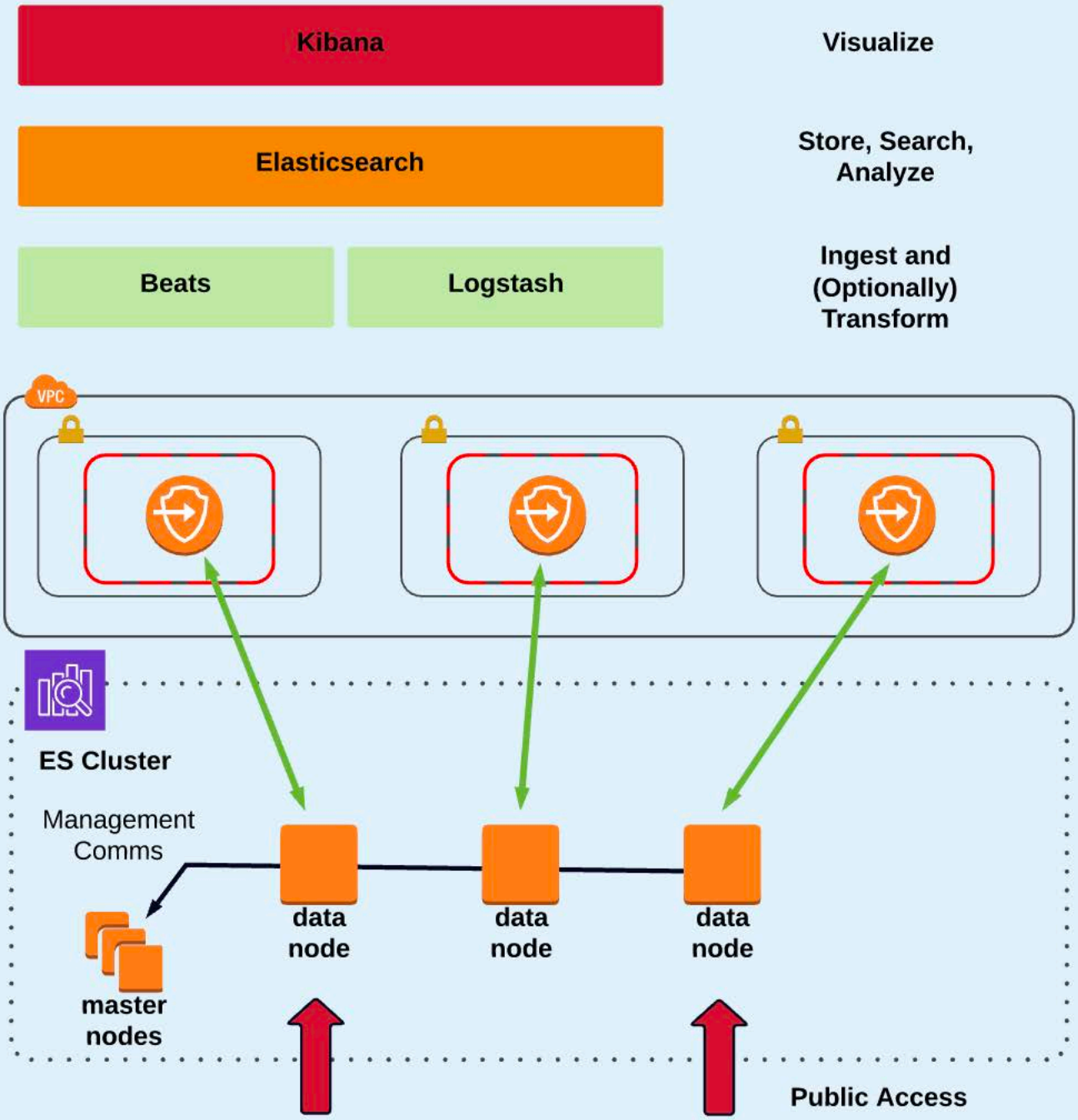
Row-based databases store data in a row arrangement on disk.
Column-based databases store data column by column.

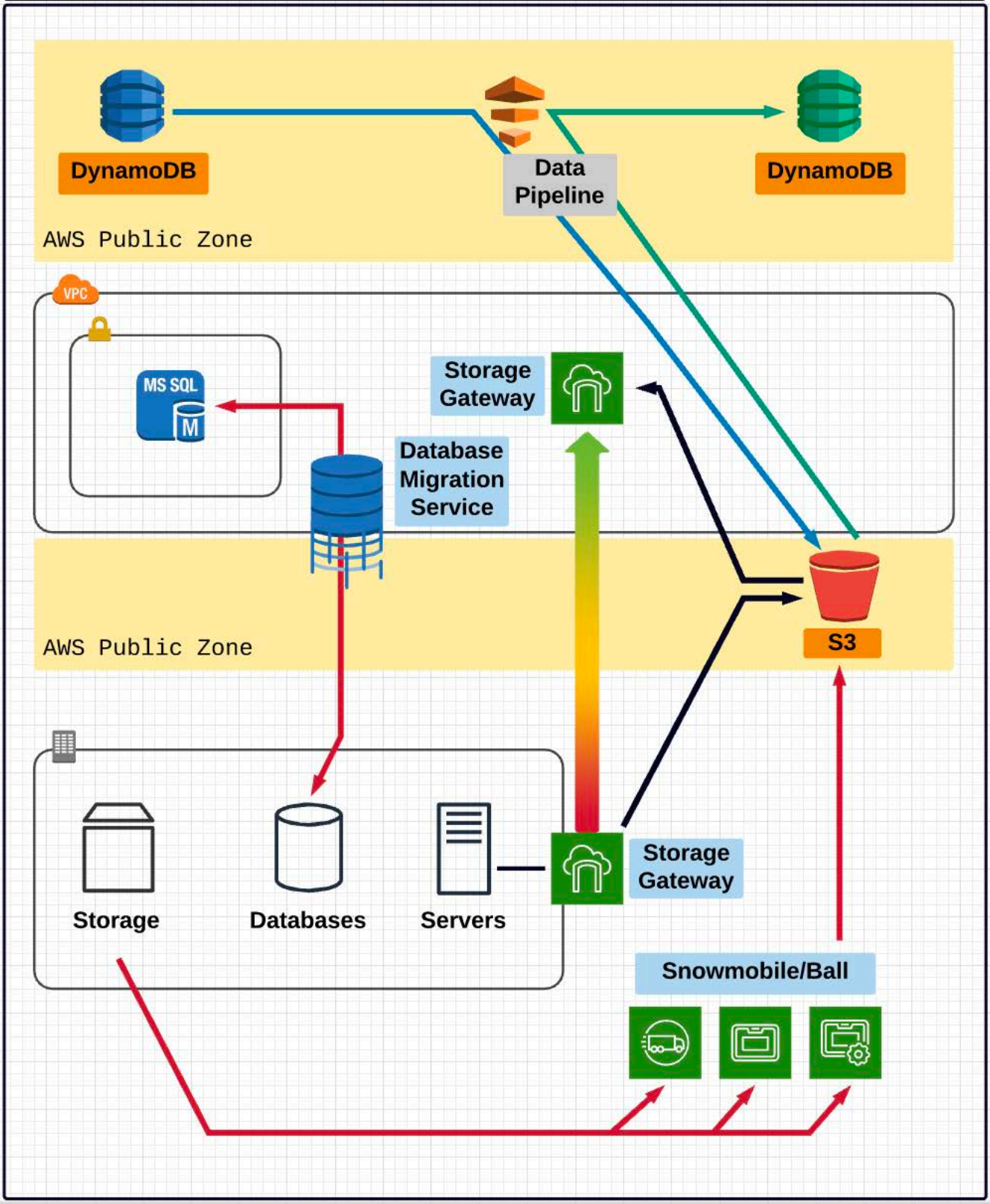
For queries involving specific columns, the data used when querying column-based databases is significantly reduced.

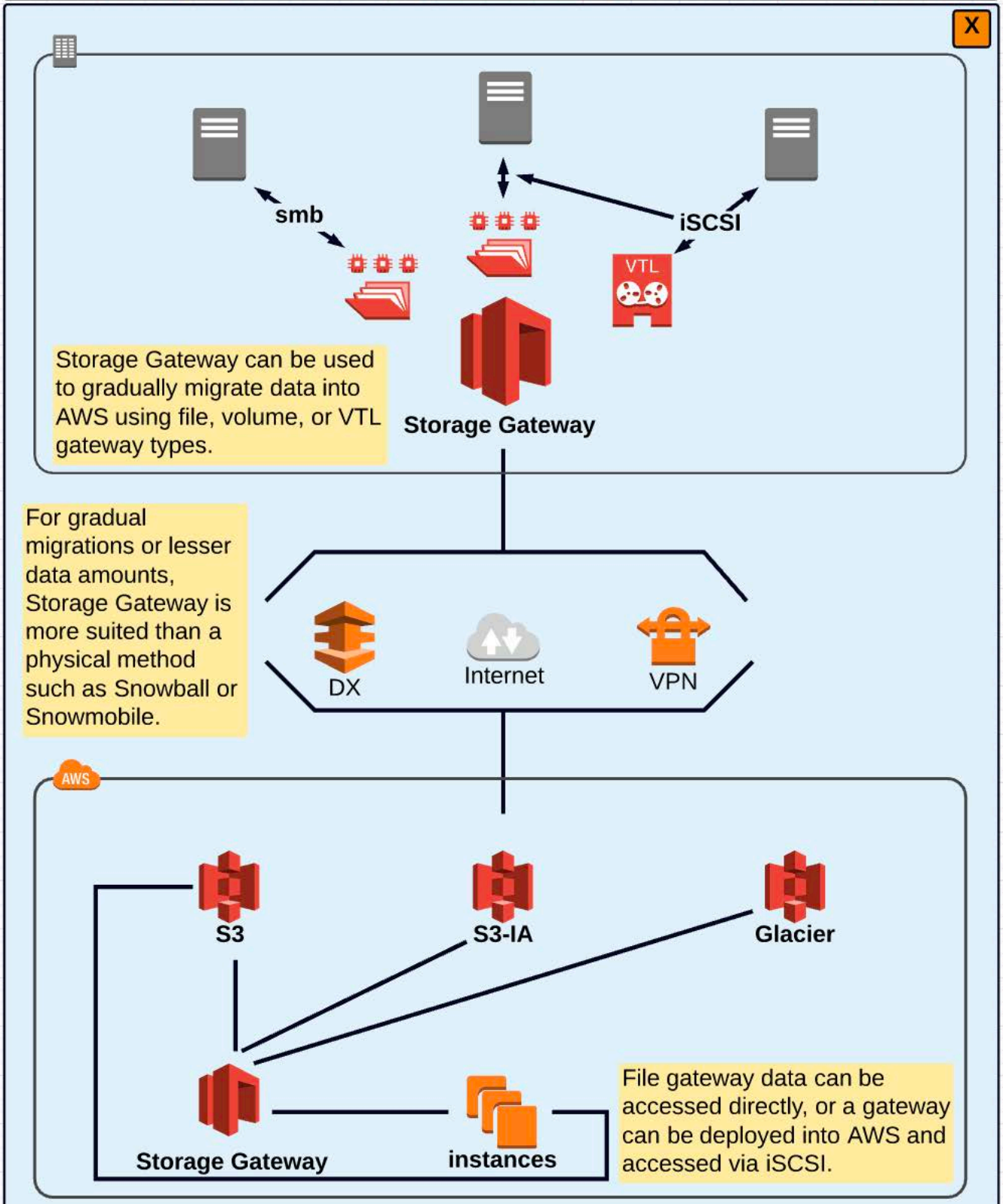
In the example above with five rows, the column-based database reads five attributes. The same query in a row-based database reads 15. The difference would be further amplified with billions of rows.

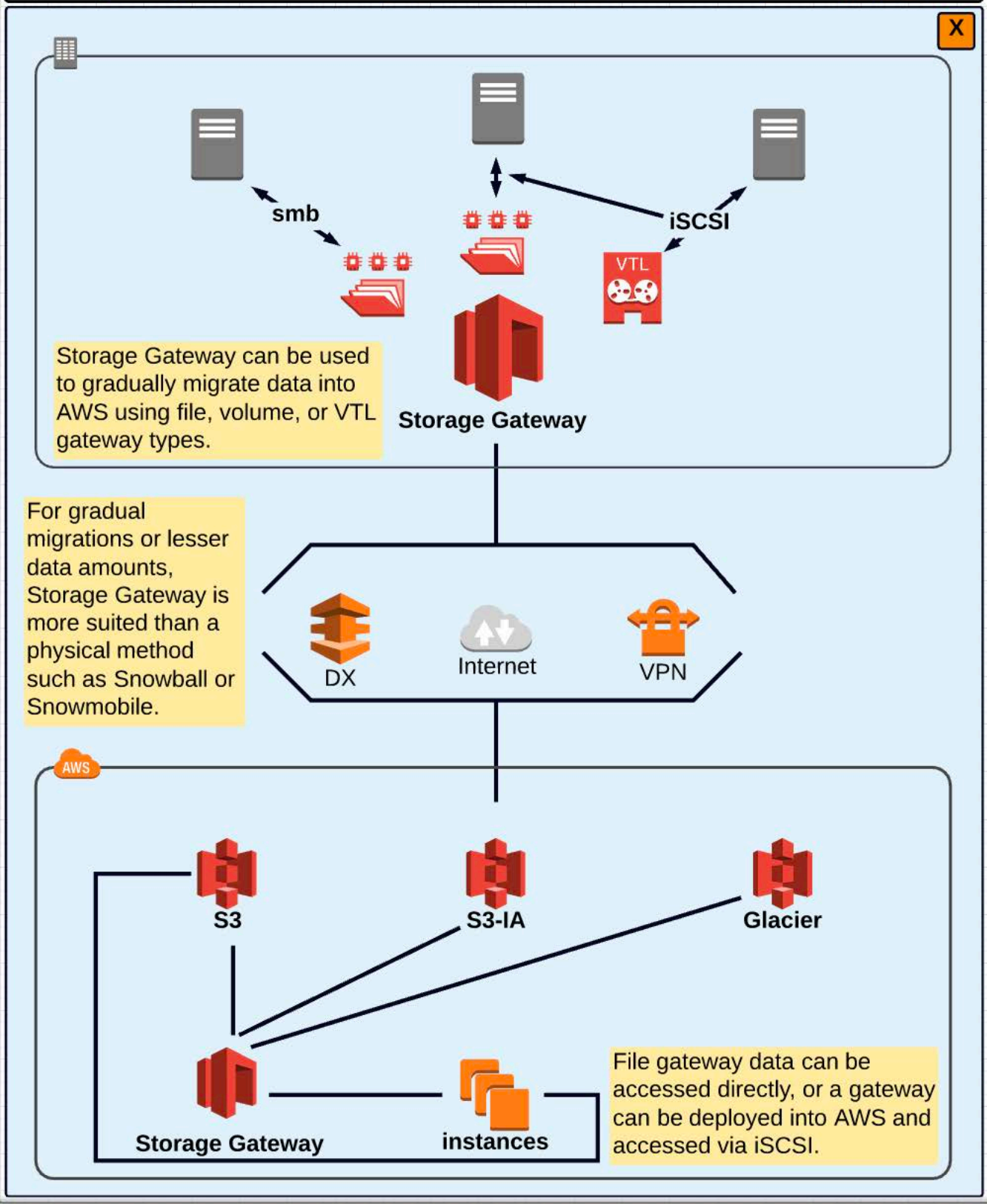
Elasticsearch (ES)

Amazon Elasticsearch is a managed service providing an implementation of the open-source Elasticsearch API and comes with Kibana and Logstash, providing a full implementation of the ELK stack.




DynamoDB
Data Pipeline
DynamoDB
AWS Public Zone
VPC
MS SQL
M
Storage Gateway
Database Migration Service
AWS Public Zone
S3
Storage
Databases
Servers
Storage Gateway
Snowmobile/Ball


X
smb
iSCSI
VTL
Storage Gateway
Storage Gateway can be used to gradually migrate data into AWS using file, volume, or VTL gateway types.
For gradual migrations or lesser data amounts, Storage Gateway is more suited than a physical method such as Snowball or Snowmobile.
DX
Internet
VPN
AWS
S3
S3-IA
Glacier
Storage Gateway
instances
File gateway data can be accessed directly, or a gateway can be deployed into AWS and accessed via iSCSI.


X
smb
iSCSI
VTL
Storage Gateway can be used to gradually migrate data into AWS using file, volume, or VTL gateway types.
Storage Gateway
For gradual migrations or lesser data amounts, Storage Gateway is more suited than a physical method such as Snowball or Snowmobile.
DX
Internet
VPN
AWS
S3
S3-IA
Glacier
Storage Gateway
instances
File gateway data can be accessed directly, or a gateway can be deployed into AWS and accessed via iSCSI.

X

Snowball

Snowballs are storage appliances provided by AWS that can store 50 TB or 80 TB of data. The device is economical when transferring more than 10 TB of data between an on-premises location and AWS or vice versa. Data is encrypted in transit to/from the Snowball and at rest on the device. Snowballs are shipped to a customer, loaded up via storage interfaces, and shipped back to AWS where the data is loaded onto S3.

Architecturally, Snowballs should be used when larger quantities of data transfer is required (i.e., anything beyond 10 TB of data). The process of requesting a snowball, loading it with data, and sending it back to AWS can achieve faster effective transfer rates than using the internet/DX/VPN.

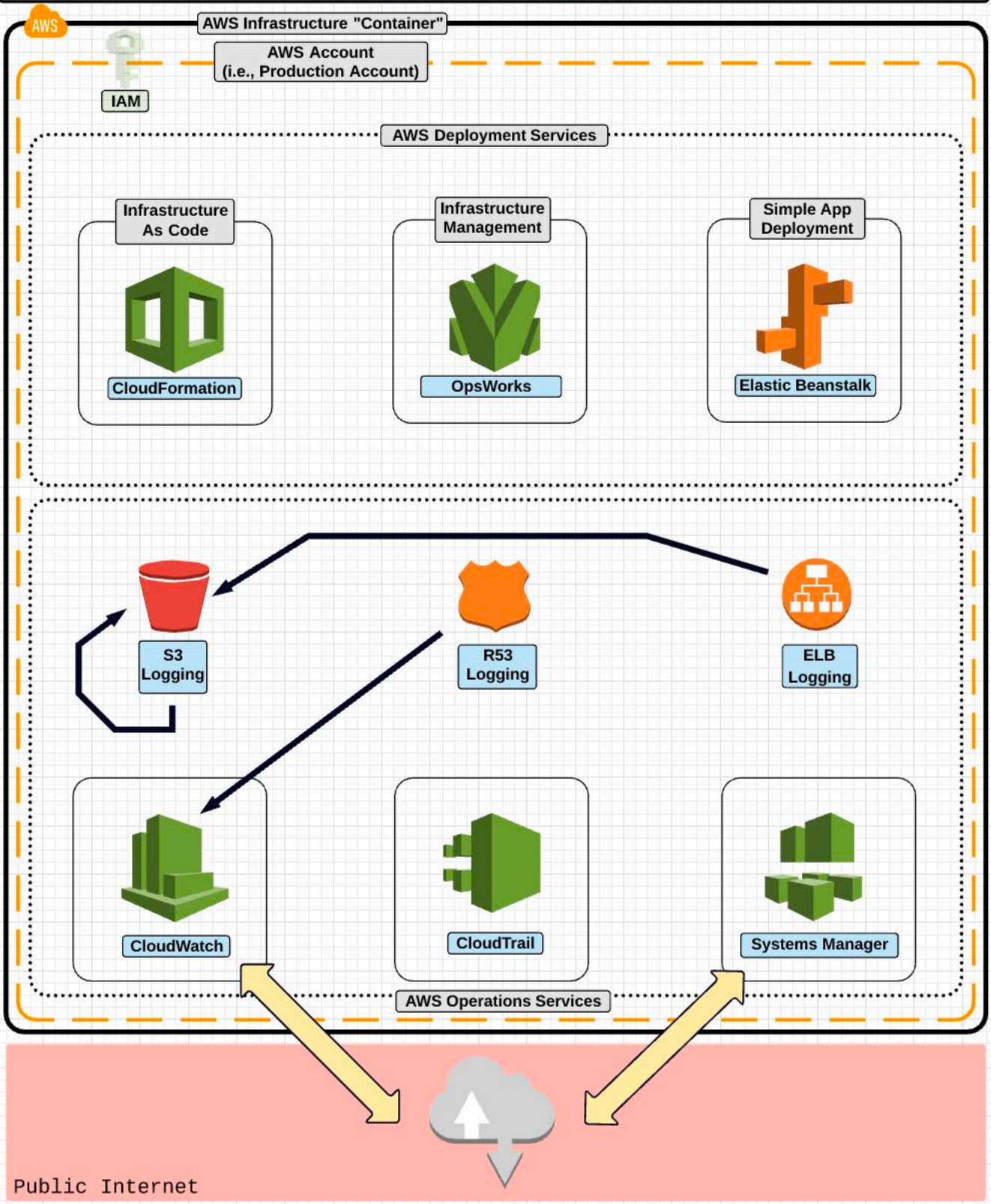
Snowball Edge

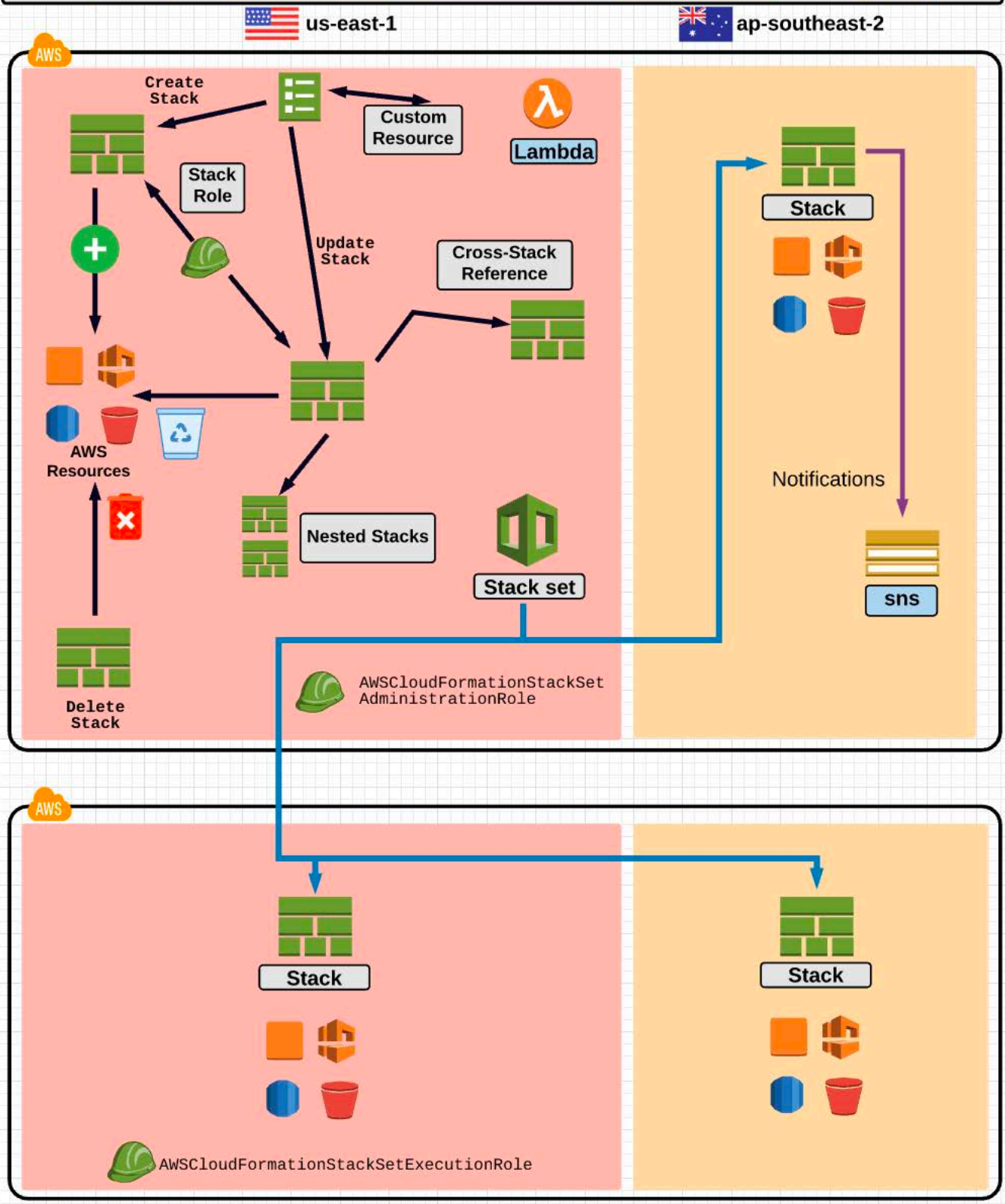
Snowball Edges are similar to standard Snowballs, but they also provide local compute services, allowing them to run processing at the edge, such as Lambda functions and instances. They are used when data needs to be processed or analyzed during the upload (e.g., remote locations with poor connectivity). There are currently three different Snowball Edge versions: **Snowball Edge Storage Optimized**, **Snowball Edge Compute Optimized**, and **Snowball Edge Compute Optimized with GPU**. Up to 100 TB can be stored per device.

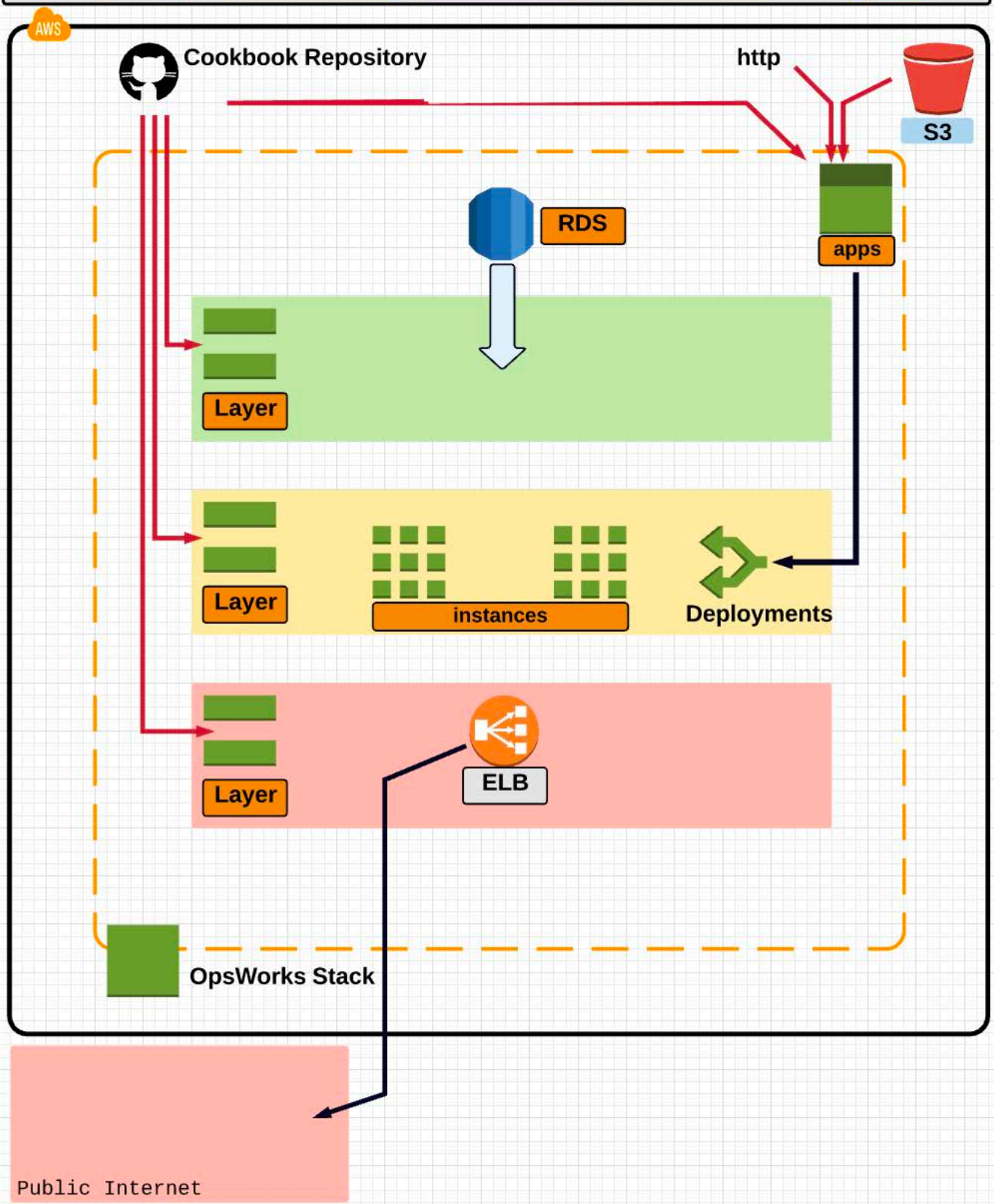
Snowmobile

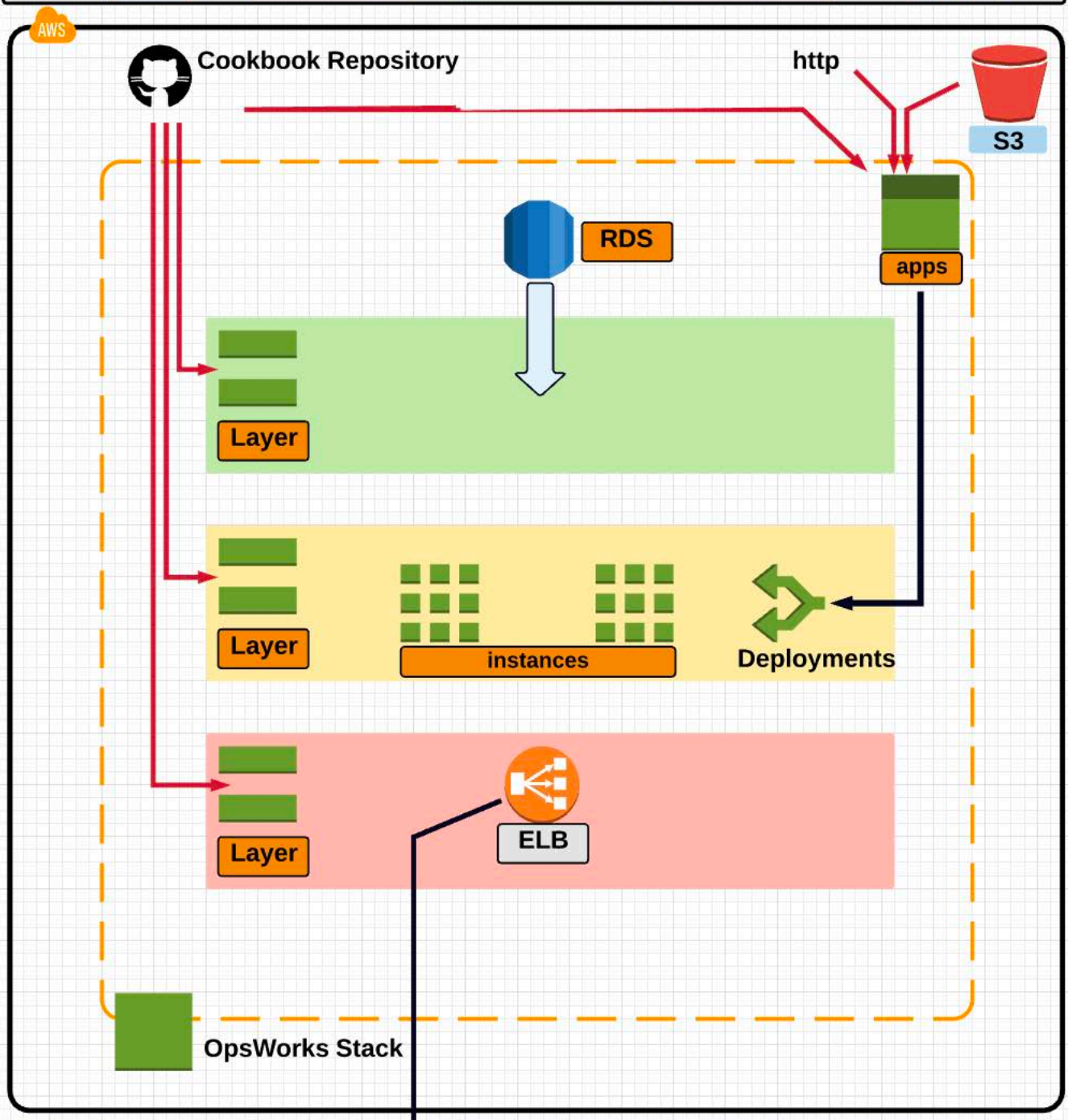
The Snowmobile is a mobile data center. It can be ordered from AWS, driven to location, connected to your data center, and loaded with huge quantities of data before being returned and loaded into AWS. Generally, the Snowmobile is used for data center migrations, or when 10 PB or more needs to be migrated from a single location. For less than 10 PB or when multiple locations are involved, use one or more Snowball/Snowball Edge. Each Snowmobile can store 100 PB, and they can be used in parallel.

?
Back
Global Infrastructure
Account & Services
Networking
Well-Architected
Appendix




us-east-1
ap-southeast-2
AWS
Create Stack
Custom Resource
Lambda
Stack Role
Update Stack
Cross-Stack Reference
AWS Resources
Nested Stacks
Stack set
Delete Stack
AWSCloudFormationStackSet AdministrationRole
Stack
Notifications
sns
AWS
Stack
Stack
AWSCloudFormationStackSetExecutionRole

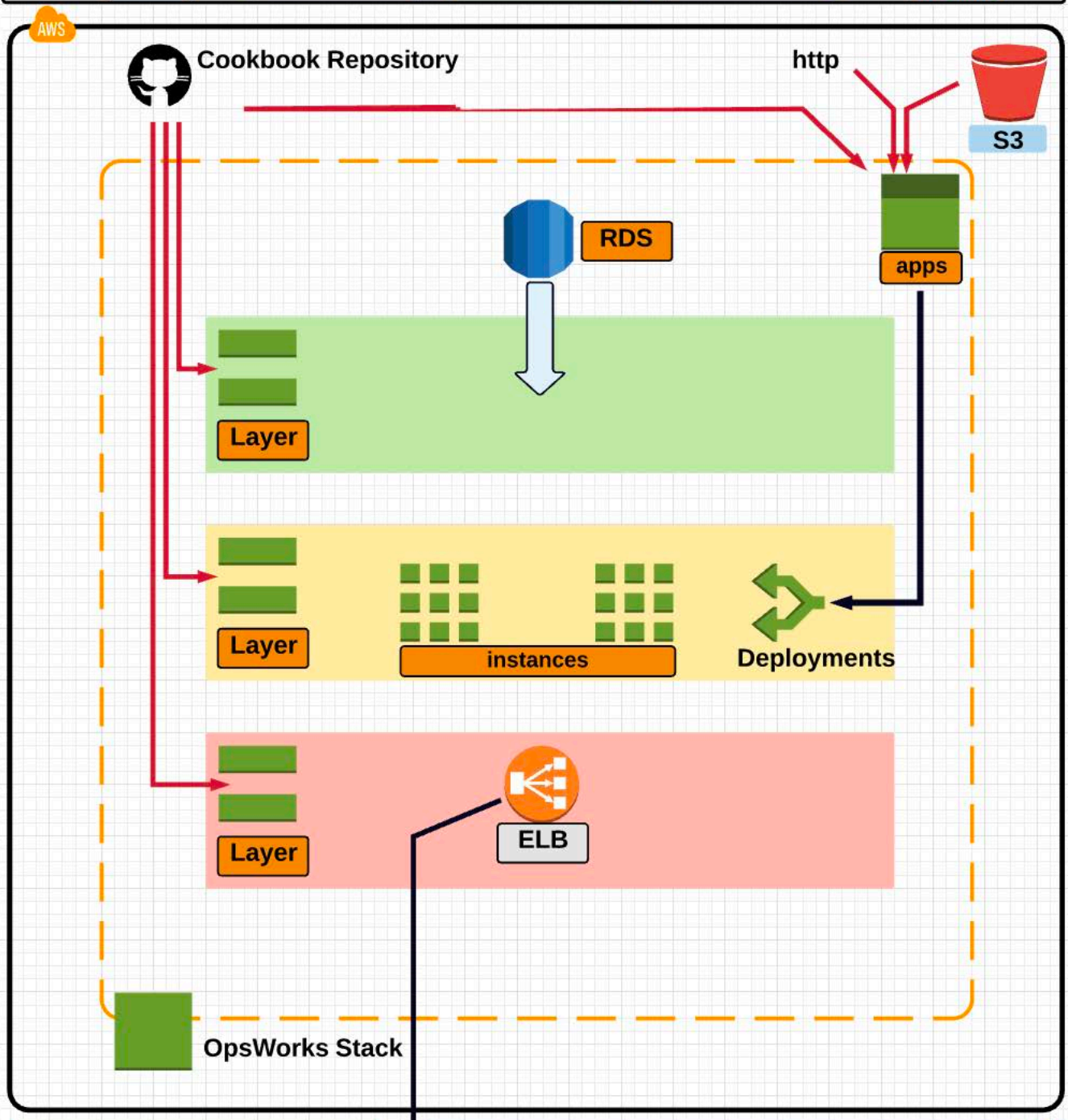

AWS
Cookbook Repository
http
S3
RDS
apps
Layer
Layer
instances
Deployments
Layer
ELB
OpsWorks Stack
Public Internet



Public Internet

A layer within OpsWorks represents a grouping of configurations (recipes). Layers are either an OpsWorks layer, an ECS layer, or an RDS service layer.

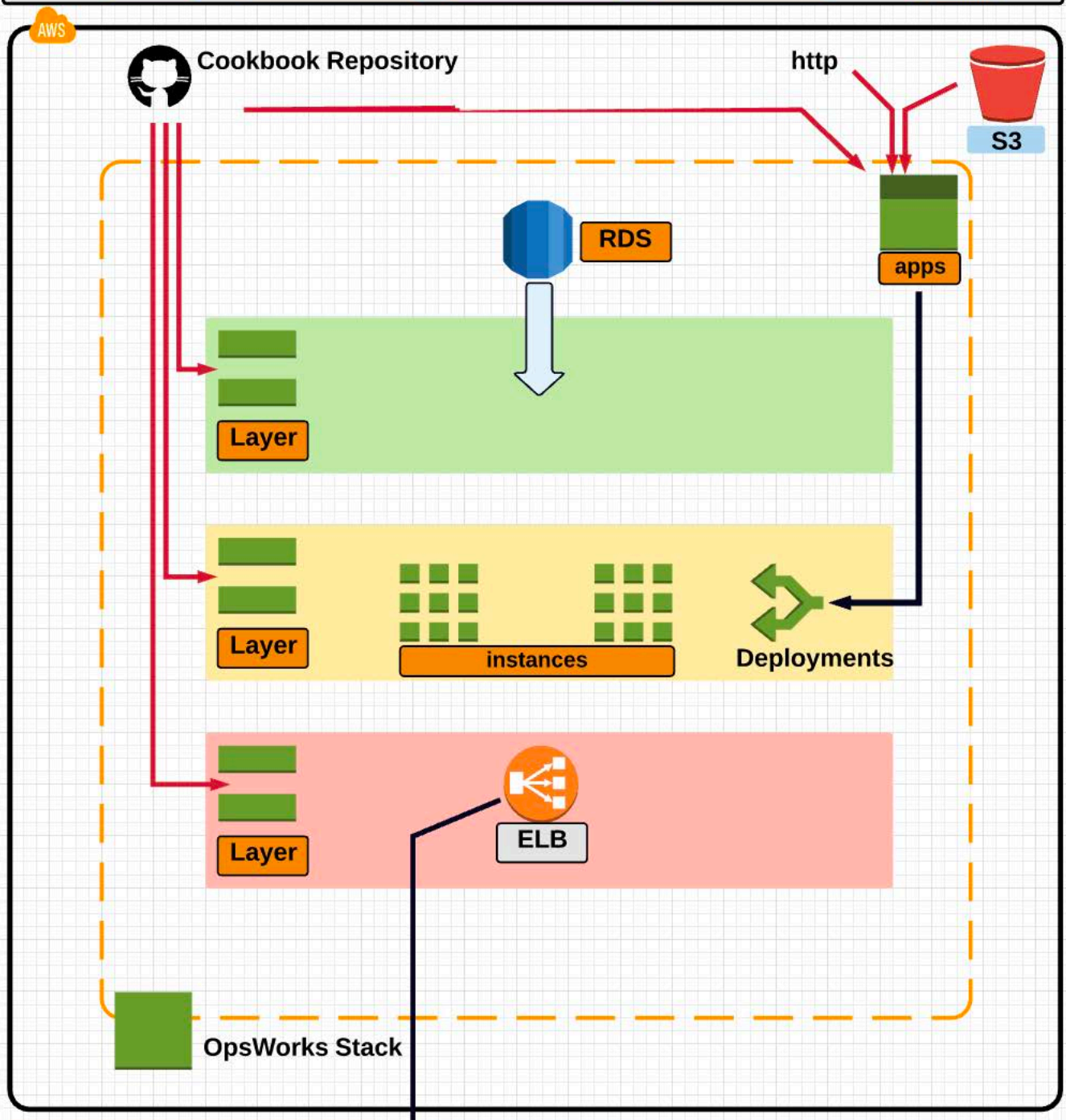
X



Public Internet

RDS instances in the same AWS region can be registered with OpsWorks and placed in an RDS service layer. Removing an RDS instance from a layer does *not* delete the RDS instance — it only removes it from the service layer.

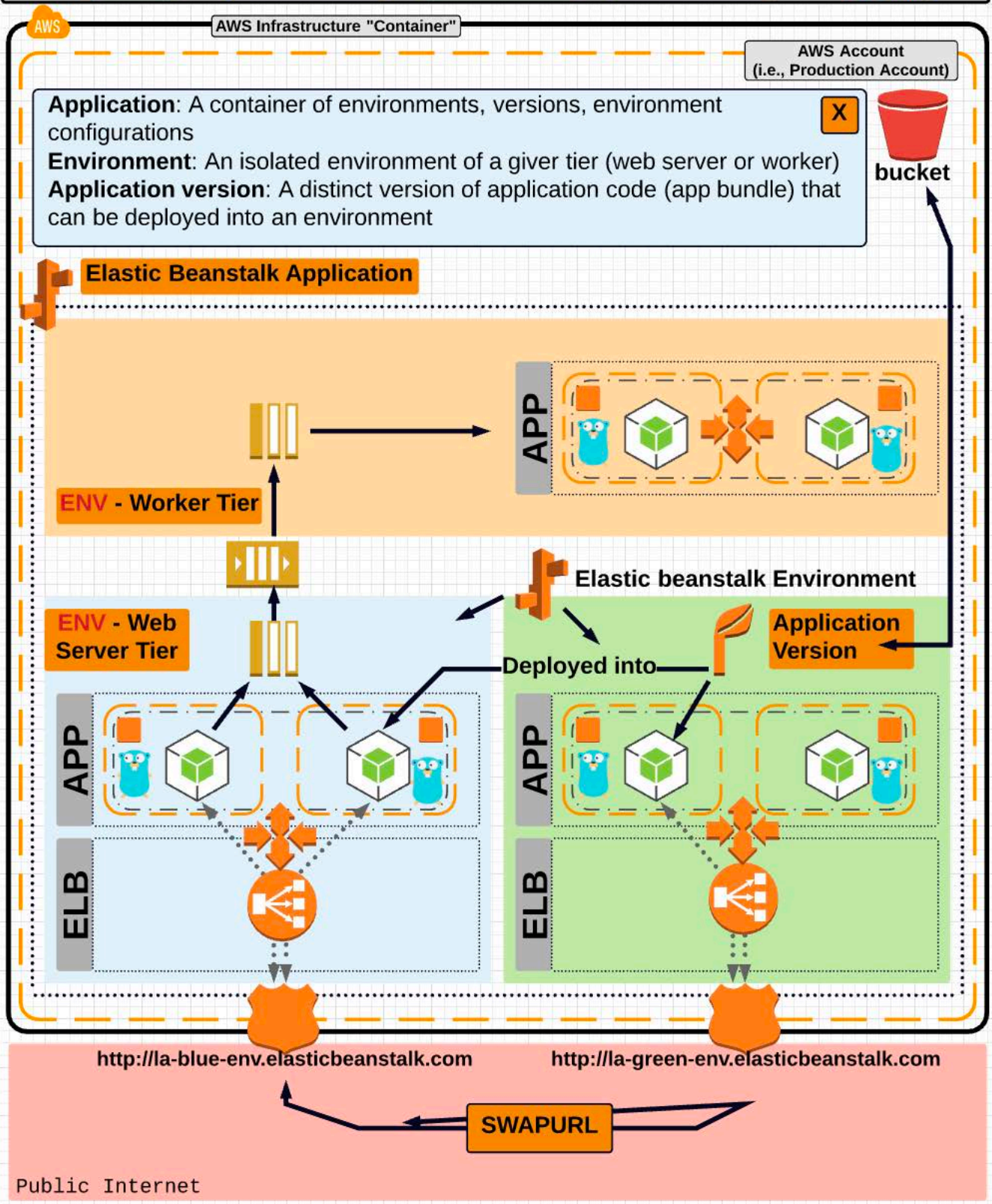
X

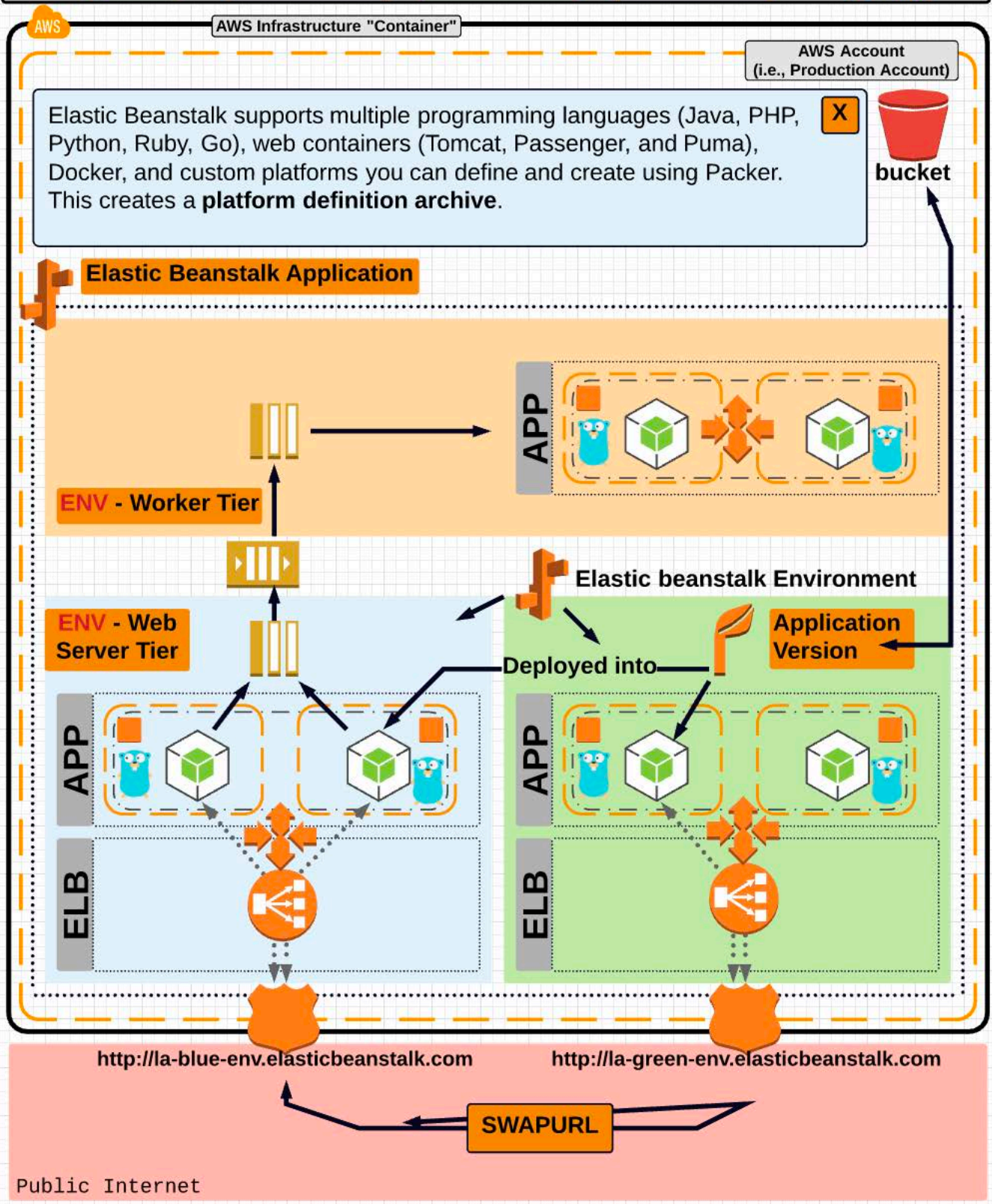


Public Internet

An app in OpsWorks is code that runs on an OpsWorks instance. Apps can be sourced from GitHub, S3, or downloaded via HTTP. App deployment is done via the deploy recipe. Apps can be connected to an RDS instance.

X


AWS
AWS Infrastructure "Container"
AWS Account
(i.e., Production Account)
Application: A container of environments, versions, environment configurations
Environment: An isolated environment of a giver tier (web server or worker)
Application version: A distinct version of application code (app bundle) that can be deployed into an environment
X
bucket
Elastic Beanstalk Application
APP
ENV - Worker Tier
Elastic beanstalk Environment
ENV - Web Server Tier
Application Version
Deployed into
APP
ELB
APP
ELB
http://la-blue-env.elasticbeanstalk.com
http://la-green-env.elasticbeanstalk.com
SWAPURL
Public Internet


AWS
AWS Infrastructure "Container"
AWS Account
(i.e., Production Account)
Elastic Beanstalk supports multiple programming languages (Java, PHP, Python, Ruby, Go), web containers (Tomcat, Passenger, and Puma), Docker, and custom platforms you can define and create using Packer. This creates a platform definition archive.
X
bucket
Elastic Beanstalk Application
APP
ENV - Worker Tier
Elastic beanstalk Environment
ENV - Web Server Tier
Application Version
Deployed into
APP
ELB
APP
ELB
http://la-blue-env.elasticbeanstalk.com
http://la-green-env.elasticbeanstalk.com
SWAPURL
Public Internet

AWS Infrastructure "Container"

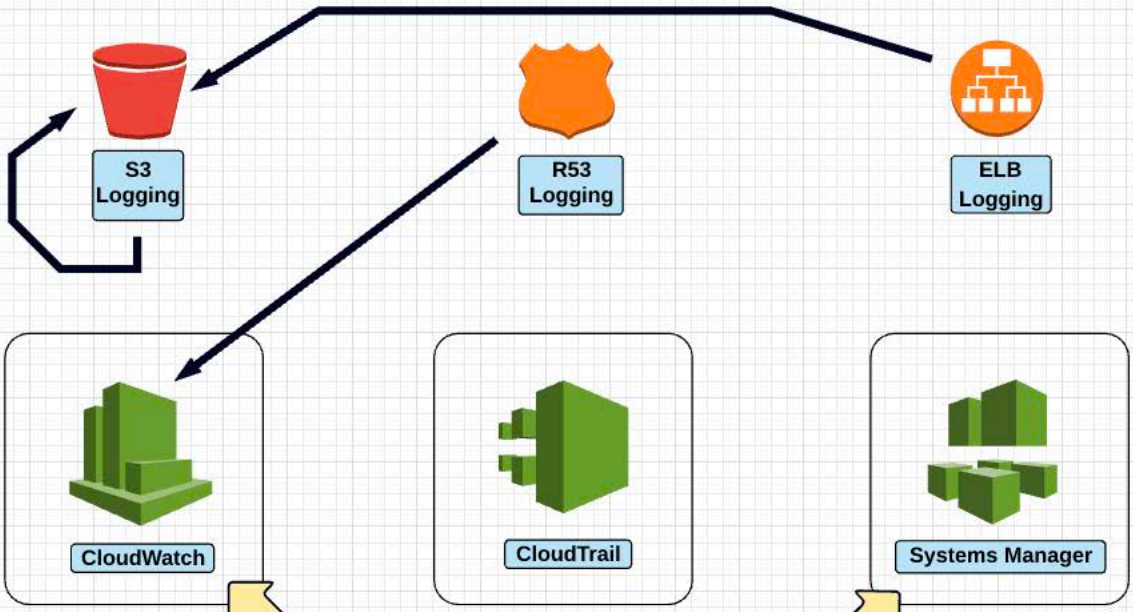
AWS Account (i.e., Production Account)

S3 Logging

X

S3 server access logging provides detailed access logs for a specific S3 bucket. Logs for a source bucket are delivered to a target bucket in the same region.

The Log Delivery group needs to be granted access via an entry in the Bucket ACL.



AWS Operations Services

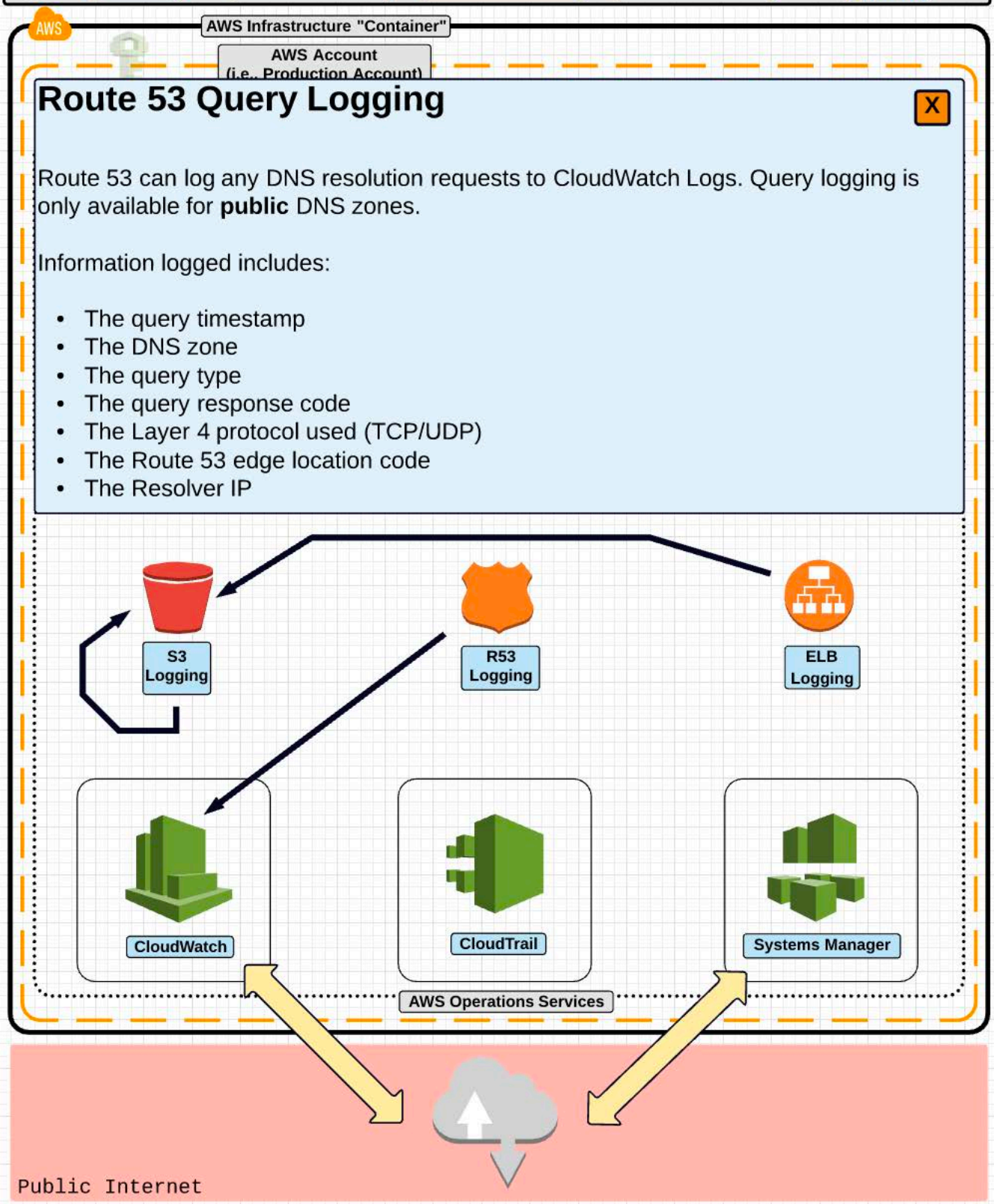
Public Internet

Route 53 Query Logging

Route 53 can log any DNS resolution requests to CloudWatch Logs. Query logging is only available for **public** DNS zones.

Information logged includes:

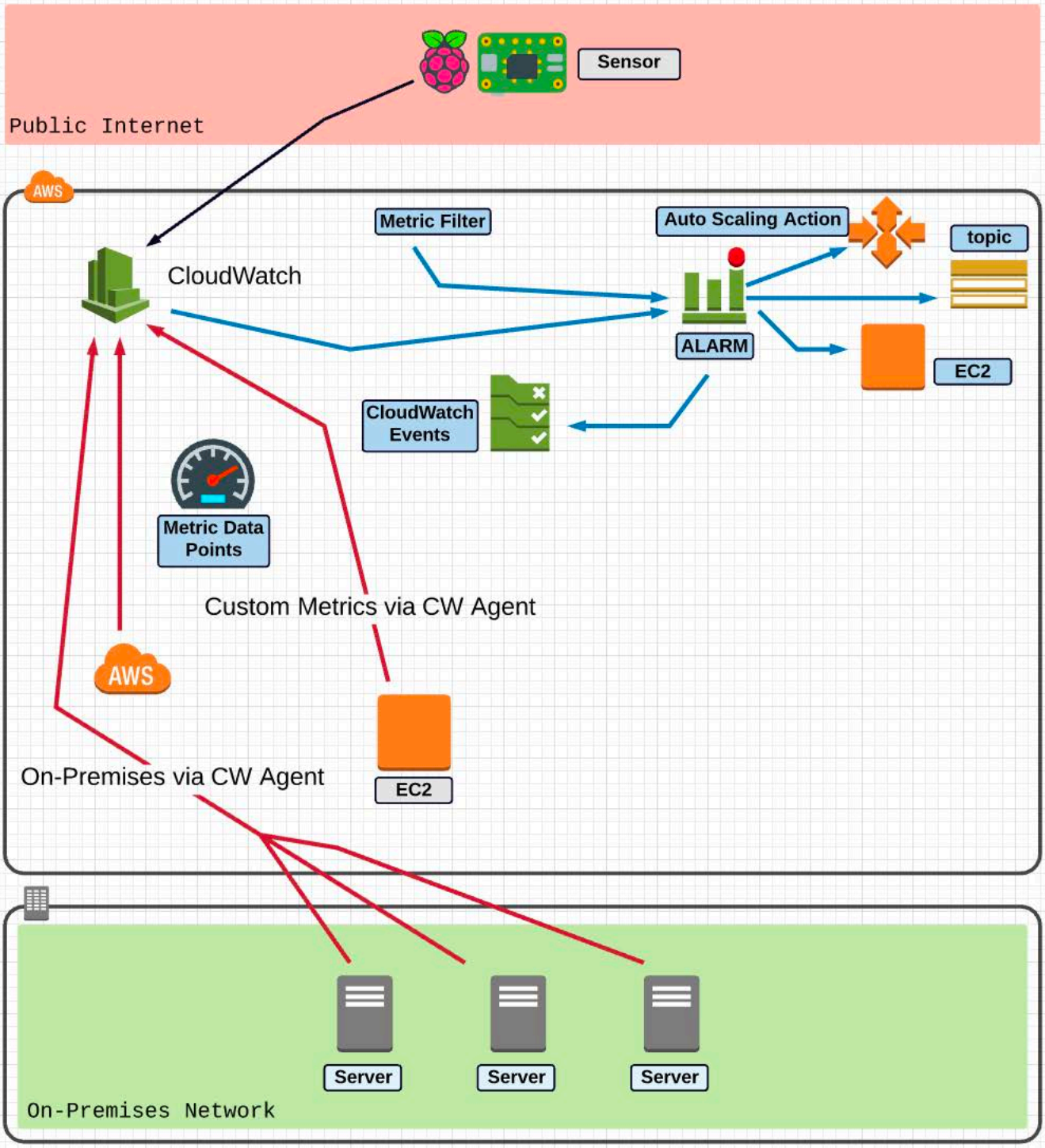
- The query timestamp
- The DNS zone
- The query type
- The query response code
- The Layer 4 protocol used (TCP/UDP)
- The Route 53 edge location code
- The Resolver IP



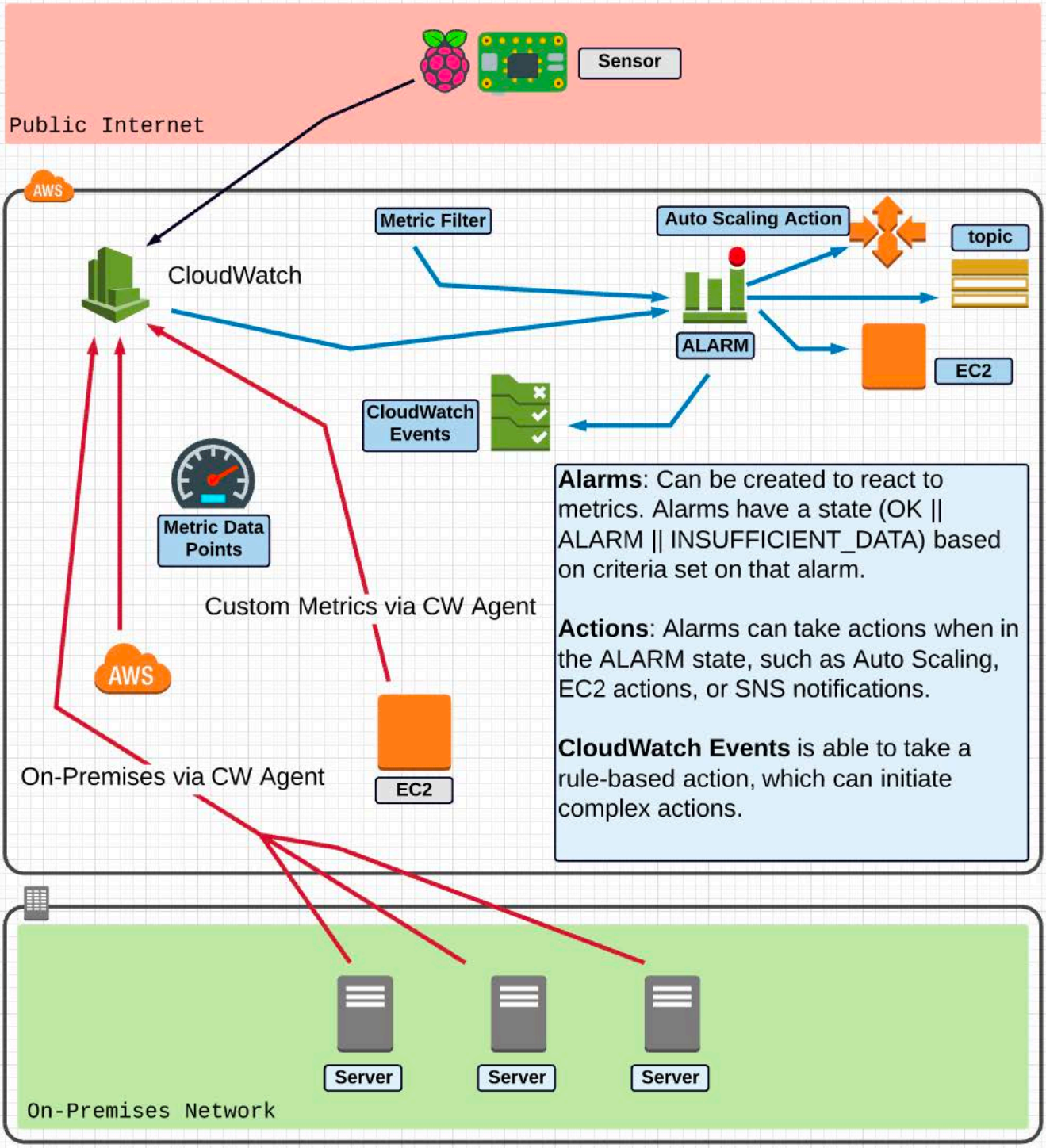
CloudWatch

CloudWatch Logs

CloudWatch Events



CloudWatch | CloudWatch Logs | CloudWatch Events

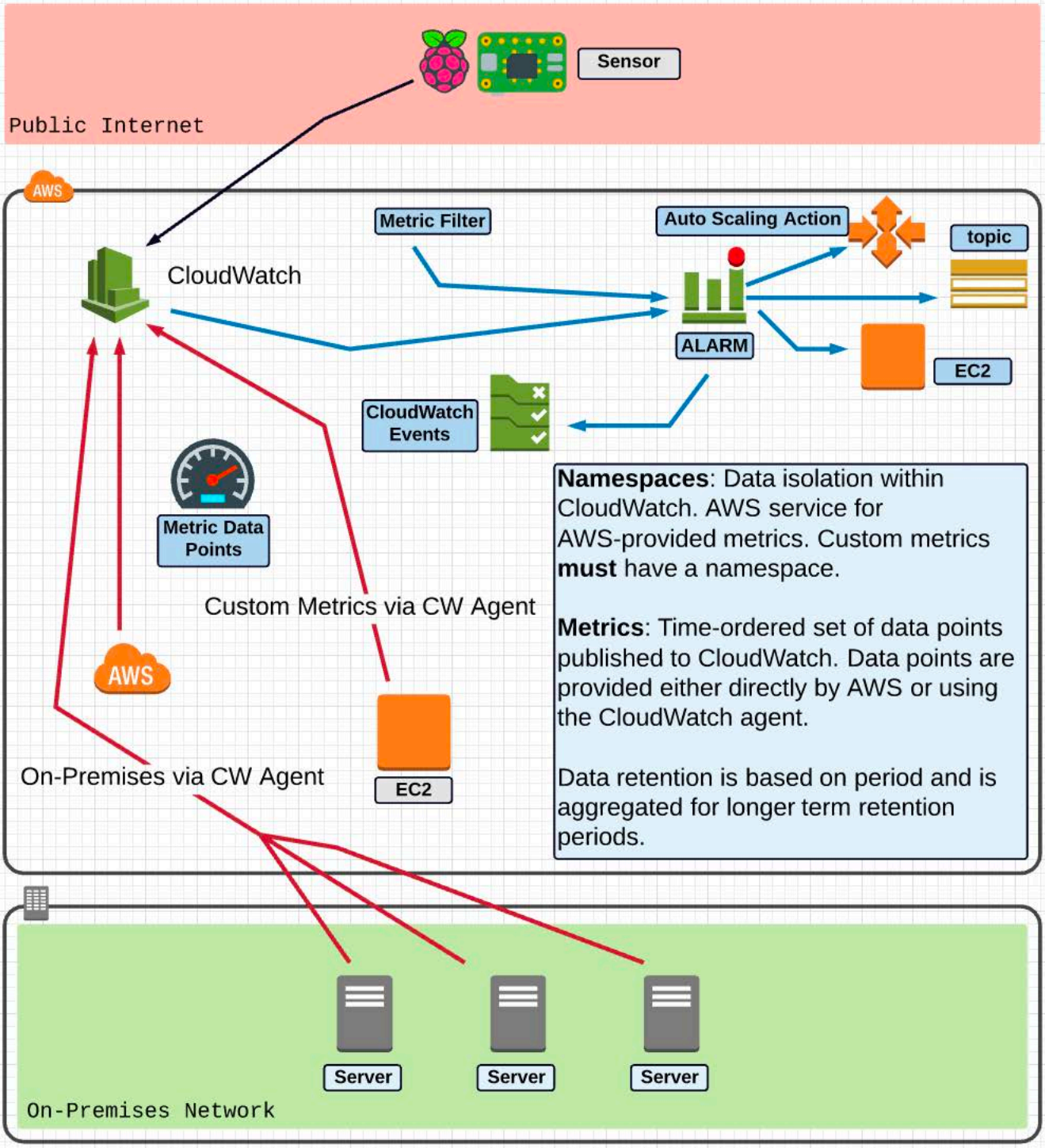


Alarms: Can be created to react to metrics. Alarms have a state (OK || ALARM || INSUFFICIENT_DATA) based on criteria set on that alarm.

Actions: Alarms can take actions when in the ALARM state, such as Auto Scaling, EC2 actions, or SNS notifications.

CloudWatch Events is able to take a rule-based action, which can initiate complex actions.

CloudWatch | CloudWatch Logs | CloudWatch Events

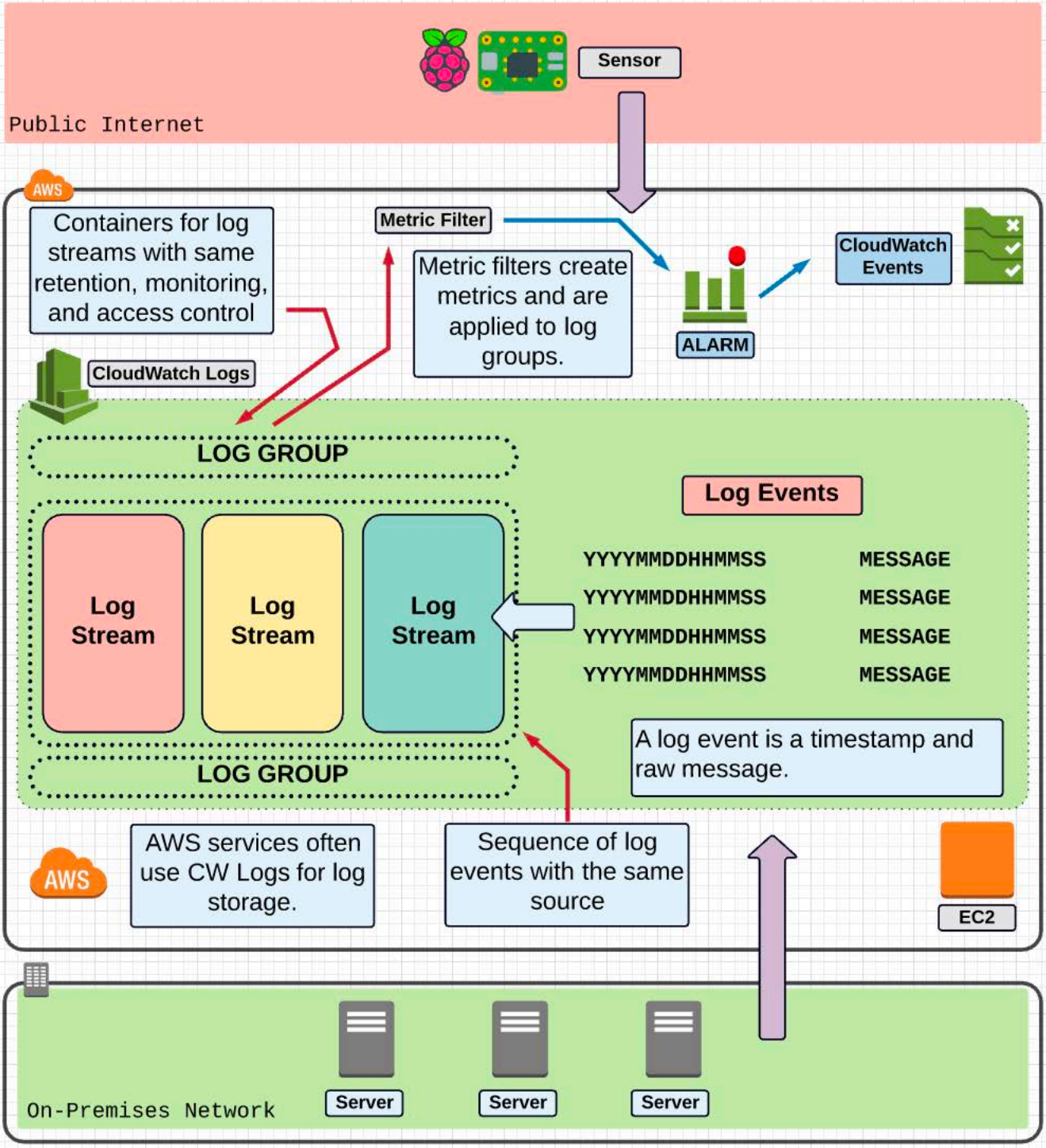


Namespaces: Data isolation within CloudWatch. AWS service for AWS-provided metrics. Custom metrics **must** have a namespace.

Metrics: Time-ordered set of data points published to CloudWatch. Data points are provided either directly by AWS or using the CloudWatch agent.

Data retention is based on period and is aggregated for longer term retention periods.

CloudWatch | CloudWatch Logs | CloudWatch Events



CloudTrail

CloudTrail is the primary service we use for logging in AWS.

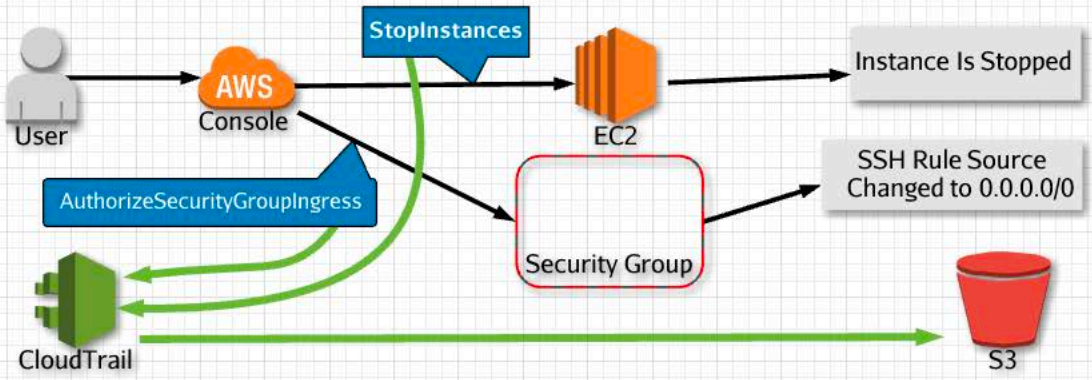
Important Features:

- It logs all the API calls in an AWS account (includes Console, CLI, API/SDK calls)
- It's enabled when your account is created
- Entries can be viewed using the **Event History** (past 90 days)
- **Trail:** A configuration allowing for logs to be sent to an S3 bucket:
  - Single region or multi-region trails can be configured
  - Trails can make multi-account logging possible (more on this later)

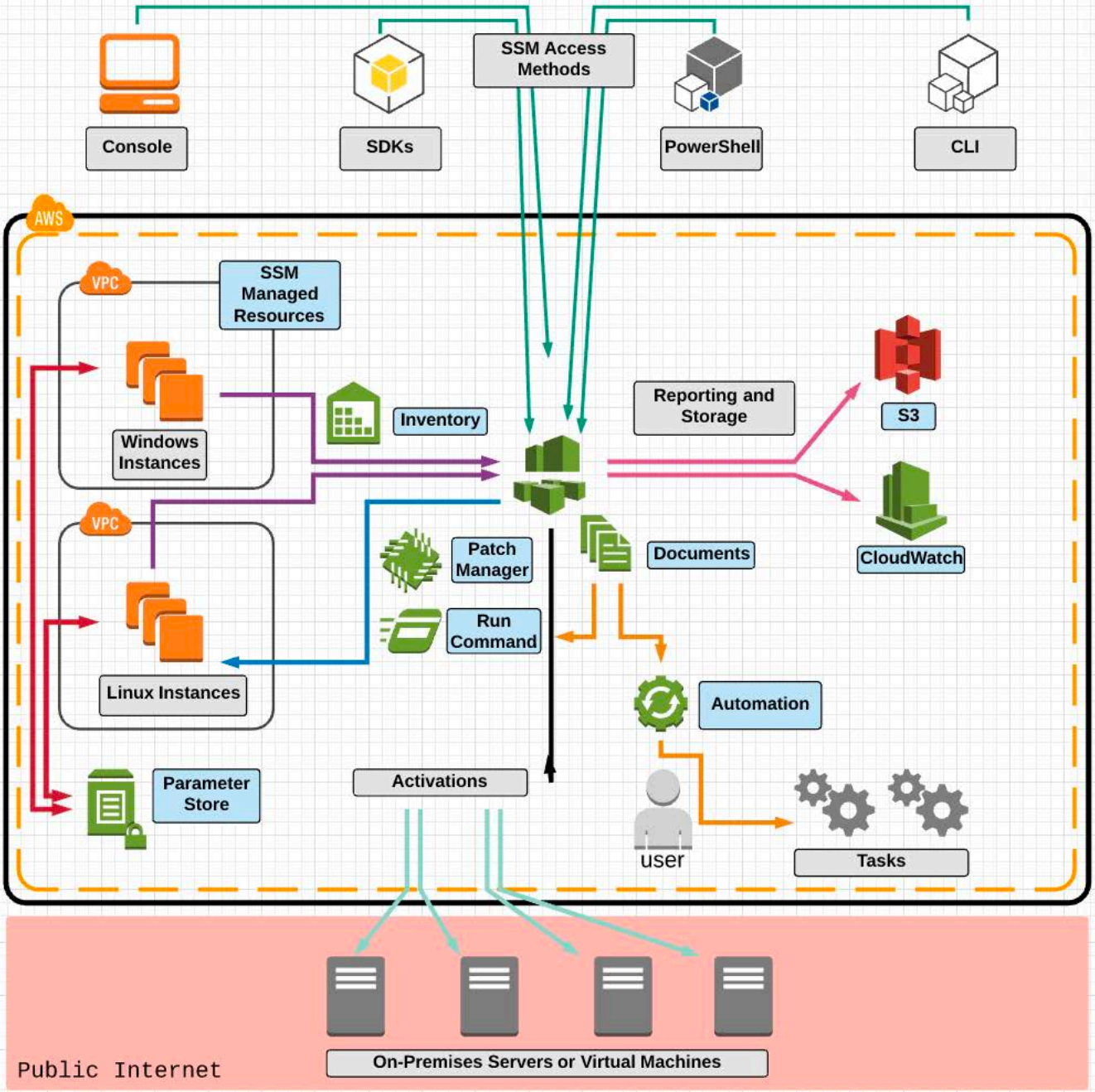
Trails have several configuration options:

- **Management events**: Enabling will log control plane events, such as:
  - User login events
  - Configuring security
  - Setting up logging
- **Data events**, which include:
  - Object-level events in S3
  - Function-level events in Lambda
- **Encryption** flexibility:
  - Encrypted in S3 server-side by default but can be changed to KMS
- The logs can be sent to an **S3 bucket** of choice and even prefixed (folders)

Scenario: An IAM user logs in to the AWS Management Console. That user then proceeds to stop an EC2 instance and edit a security group. CloudTrail will log all of these actions. Here is the workflow:



?
Back
Global Infrastructure
Account & Services
Networking
Well-Architected
Appendix



A managed instance is an EC2 instance or on-premises servers or virtual machines configured to use Systems Manager. An EC2 instance or on-premises server becomes a managed instance by installing the SSM Agent and either assigning IAM permissions and an activation code if the compute resource is not an EC2 instance.

X

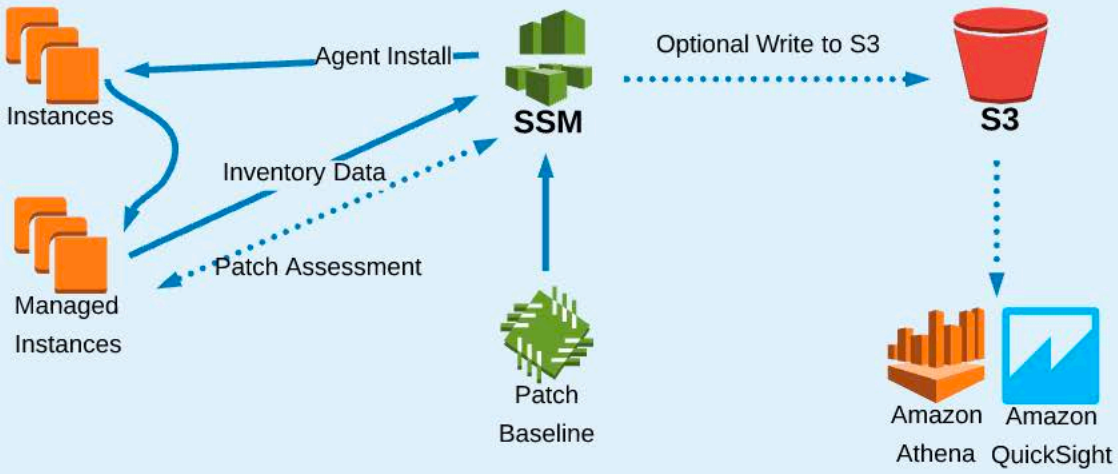
Insights (Inventory and Compliance)

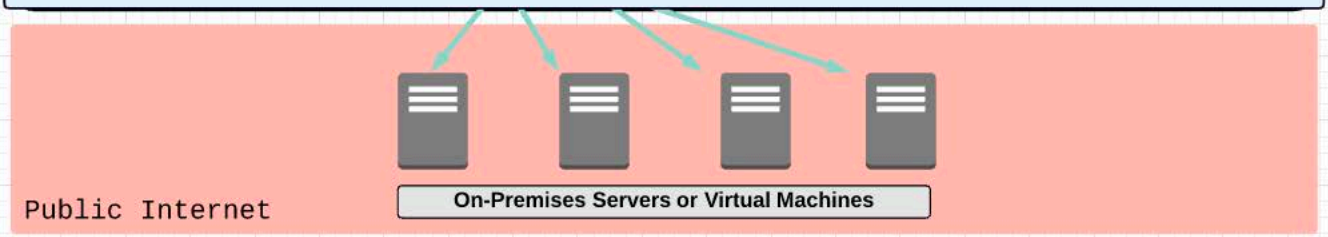
X

Systems Manager offers two insight features: inventory and compliance. Both are supported by SSM State Manager.

Inventory periodically scans EC2 instances or on-premises servers/VMs, retrieving details of installed applications, AWS components, network config, Windows updates, detailed information on an instance/VM, details on running services, Windows roles, and optional custom data SSM can collect on your behalf.

Compliance allows that data to be compared against a baseline, providing a compliant or non-compliant state to a resource. Compliance uses State Manager, SSM patching, and custom compliance types.



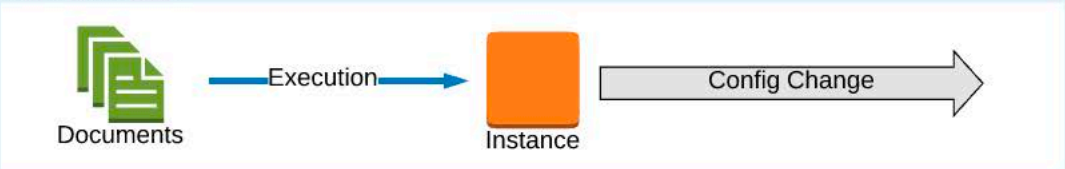


Actions

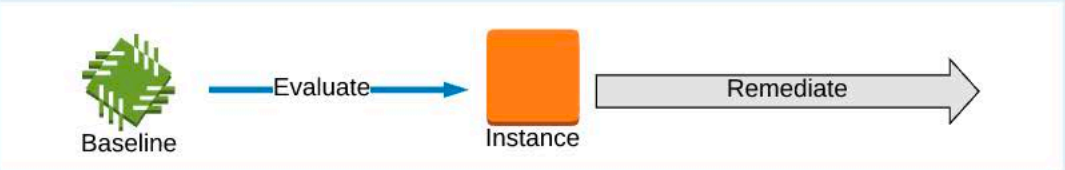
Actions are the operational engine part of Systems Manager. Actions is the part of Systems Manager that performs collections, runs commands, controls patching, and manages the general state of *managed instances*.

Automation

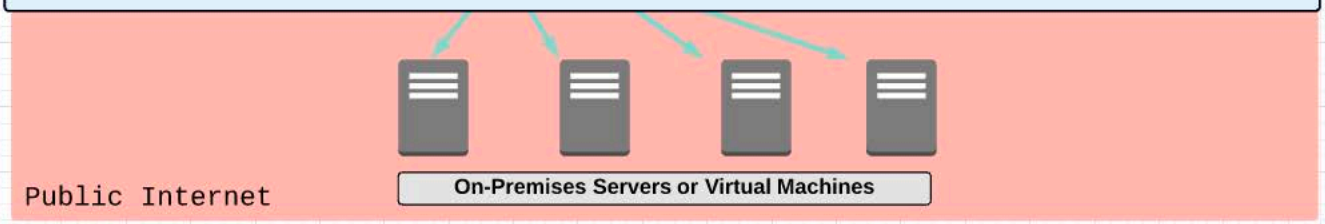
Run Command



Patch Manager



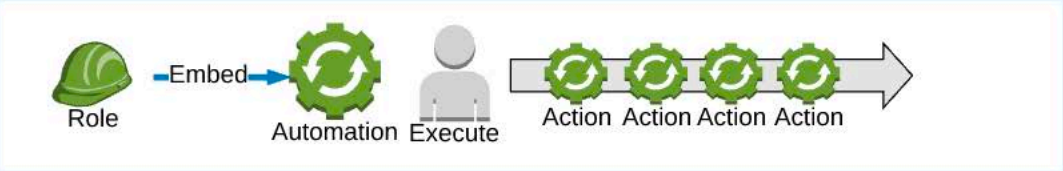
State Manager is a desired state engine. You define the "desired state" in the form of a Systems Manager document, which can be a command document or a policy document. A command document is used by Run Command and State Manager, while a policy document defines desired states and is only used by State Manager. A document is associated with one or more managed instances.



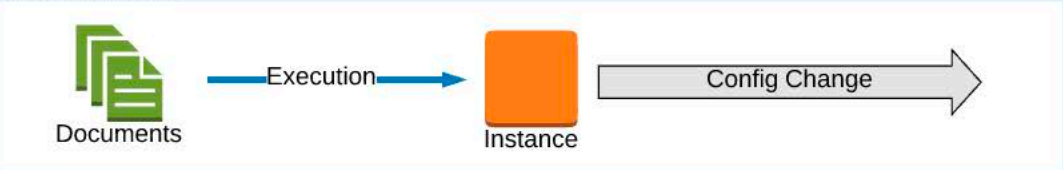
Actions

Actions are the operational engine part of Systems Manager. Actions is the part of Systems Manager that performs collections, runs commands, controls patching, and manages the general state of *managed instances*.

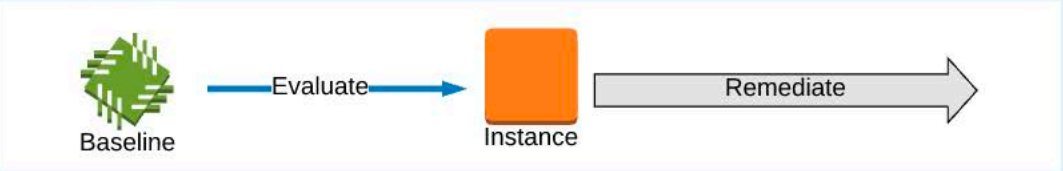
Automation



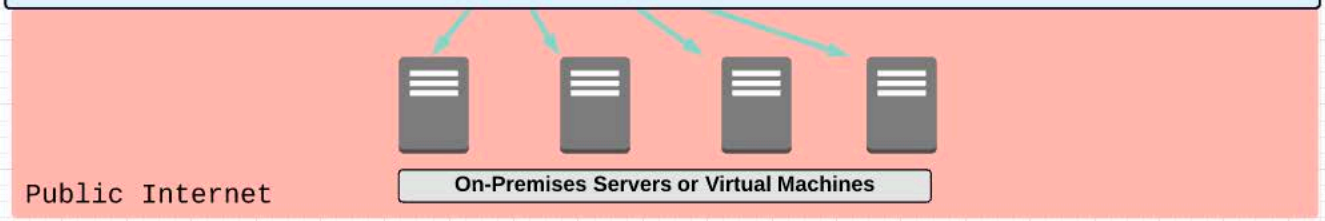
Run Command



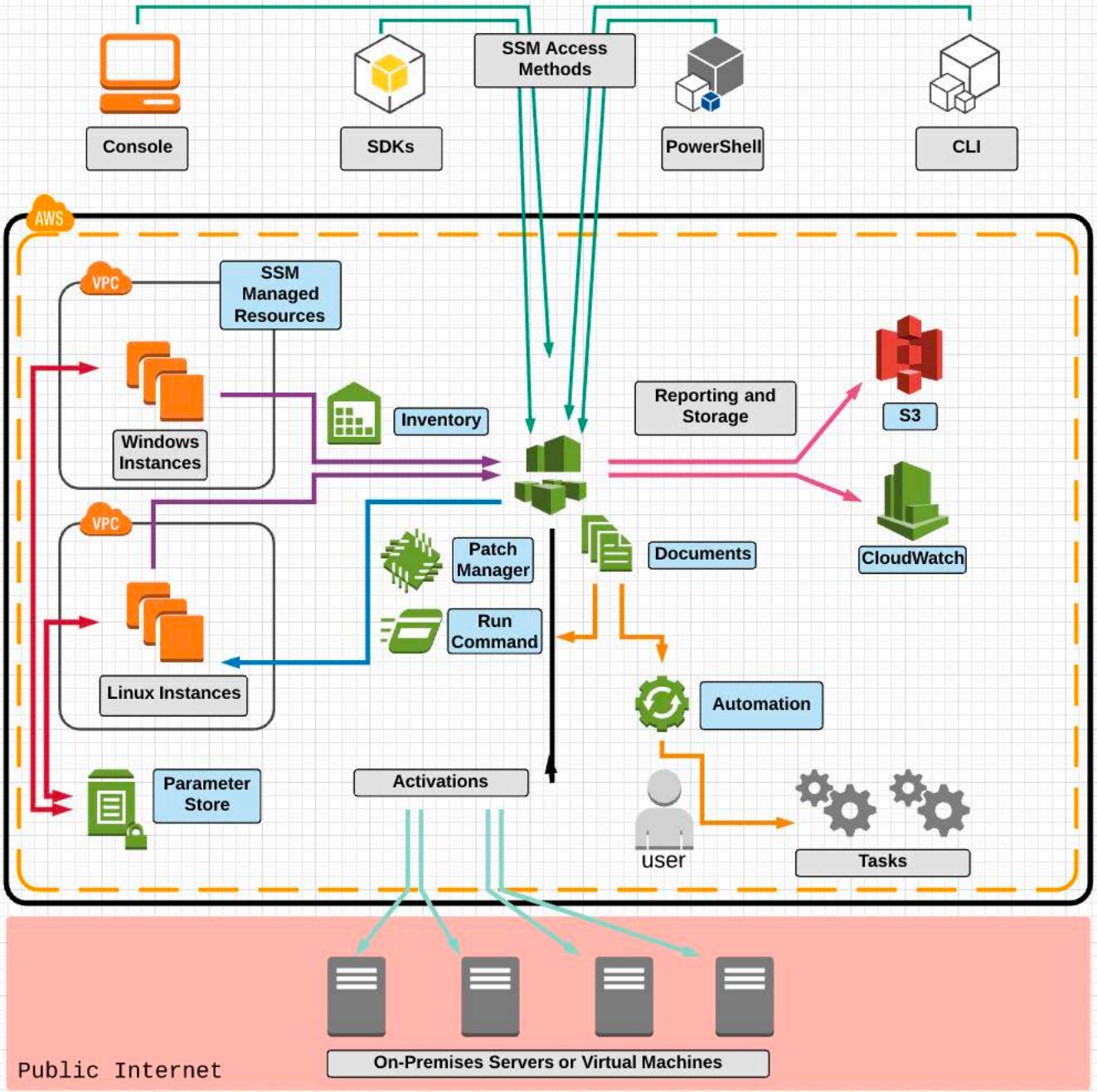
Patch Manager



State Manager is a desired state engine. You define the "desired state" in the form of a Systems Manager document, which can be a command document or a policy document. A command document is used by Run Command and State Manager, while a policy document defines desired states and is only used by State Manager. A document is associated with one or more managed instances.



? Back | Global Infrastructure | Account & Services | Networking | Well-Architected | Appendix



Systems Manager Documents define actions that Systems Manager performs and can be used by State Manager, Run Command, and Automation. They are controlling directives and can be written in YAML or JSON.

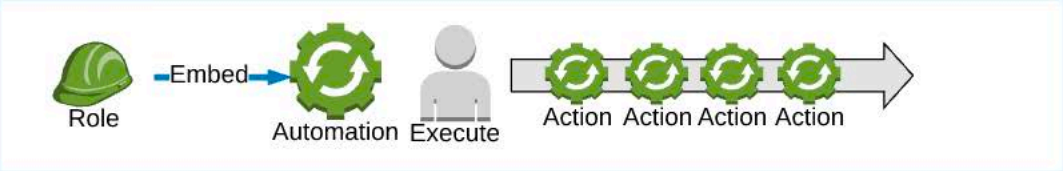
X

Actions

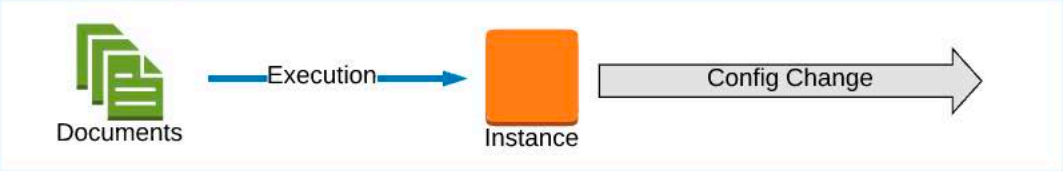
X

Actions are the operational engine part of Systems Manager. Actions is the part of Systems Manager that performs collections, runs commands, controls patching, and manages the general state of *managed instances*.

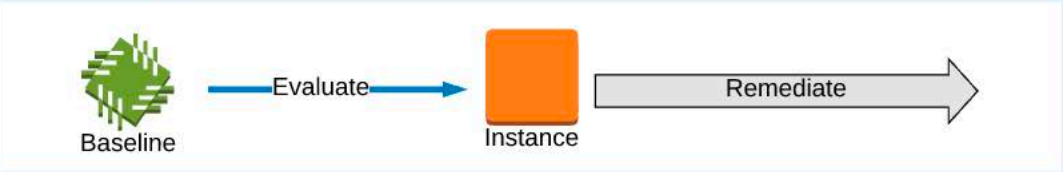
Automation



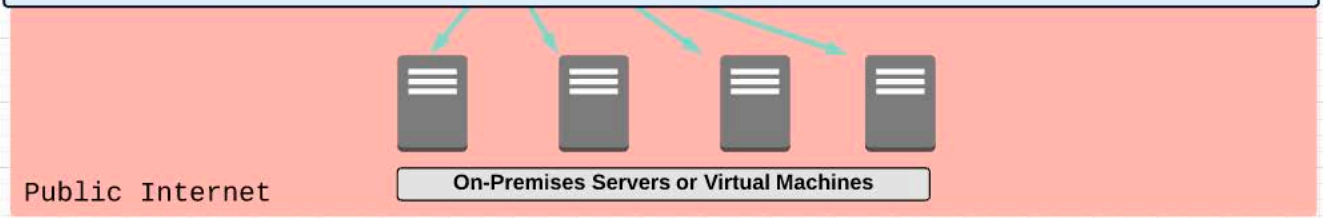
Run Command



Patch Manager



State Manager is a desired state engine. You define the "desired state" in the form of a Systems Manager document, which can be a command document or a policy document. A command document is used by Run Command and State Manager, while a policy document defines desired states and is only used by State Manager. A document is associated with one or more managed instances.

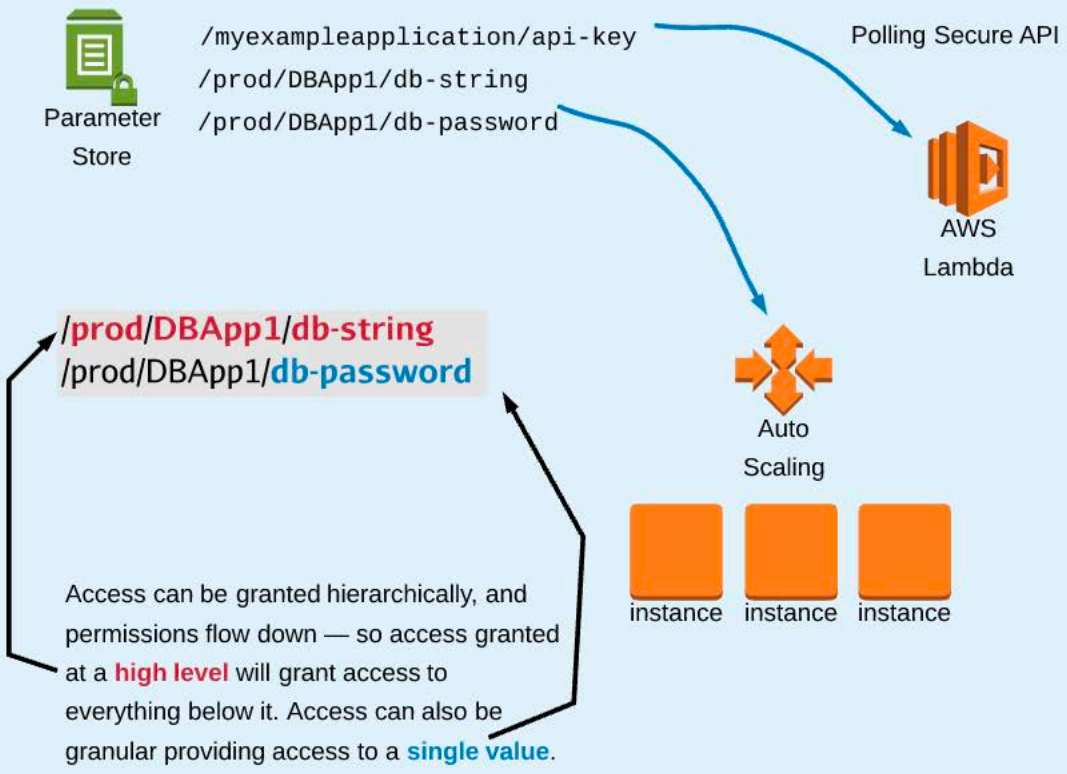


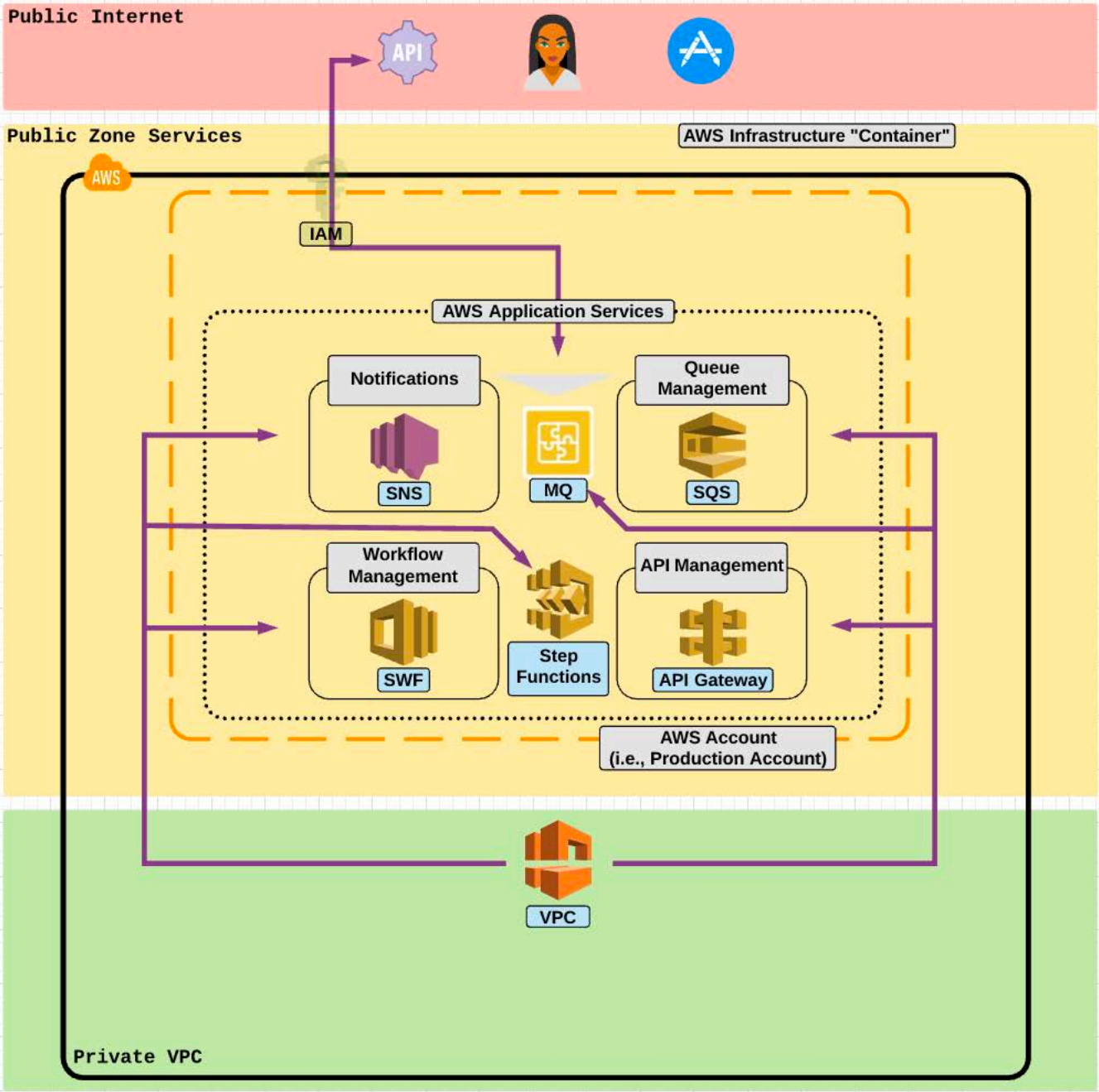
AWS Systems Manager Parameter Store

AWS Parameter Store provides secure storage for configuration data and secrets. Values can be stored as plaintext or as encrypted data using KMS. Data can be referenced using a unique name, providing you have permissions to access the data.

Key Features:

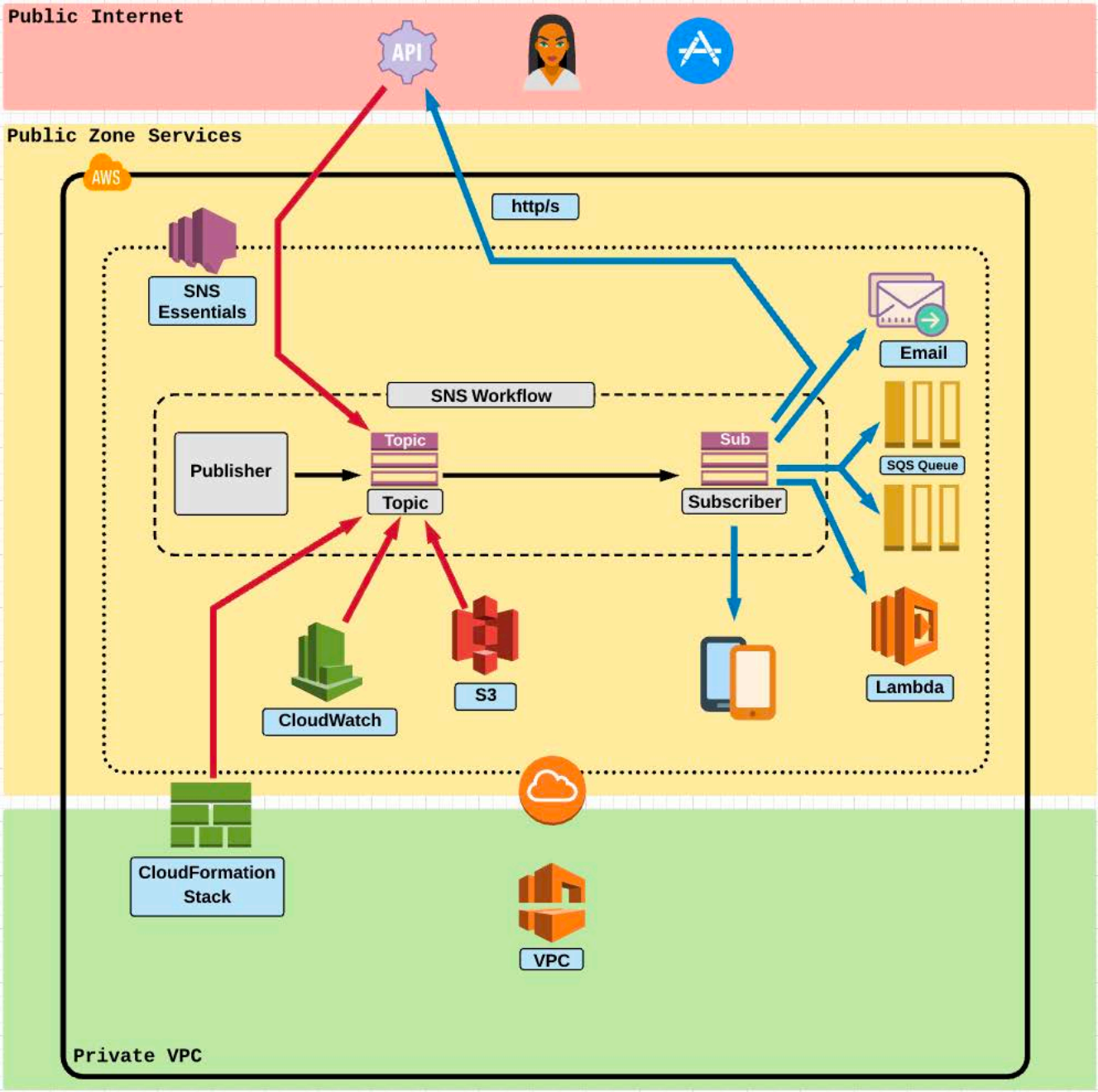
- Configuration and data is separated from code — no chance of leakage via Git
- Data is stored hierarchically, which aids management
- Data is versioned, and access can be controlled and audited
- Parameter Store integrates with many AWS services, including EC2, ECS, Lambda, CodeBuild/Deploy, and many more
- Can also be used for automated deployment using CloudFormation
- Serverless, resilient, and scalable





Application integration services act as supporting functionality to other applications, products, and systems. They generally allow the use of a certain architecture or offload certain parts of application functionality.

Application integration services are generally public zone services, meaning they can be accessed from within private AWS networks or from remote locations over the public internet.


Public Internet
API
Public Zone Services
AWS
http/s
SNS Essentials
Email
SNS Workflow
Publisher
Topic
Topic
Sub
Subscriber
SQS Queue
CloudWatch
S3
Lambda
CloudFormation Stack
VPC
Private VPC

Application Services

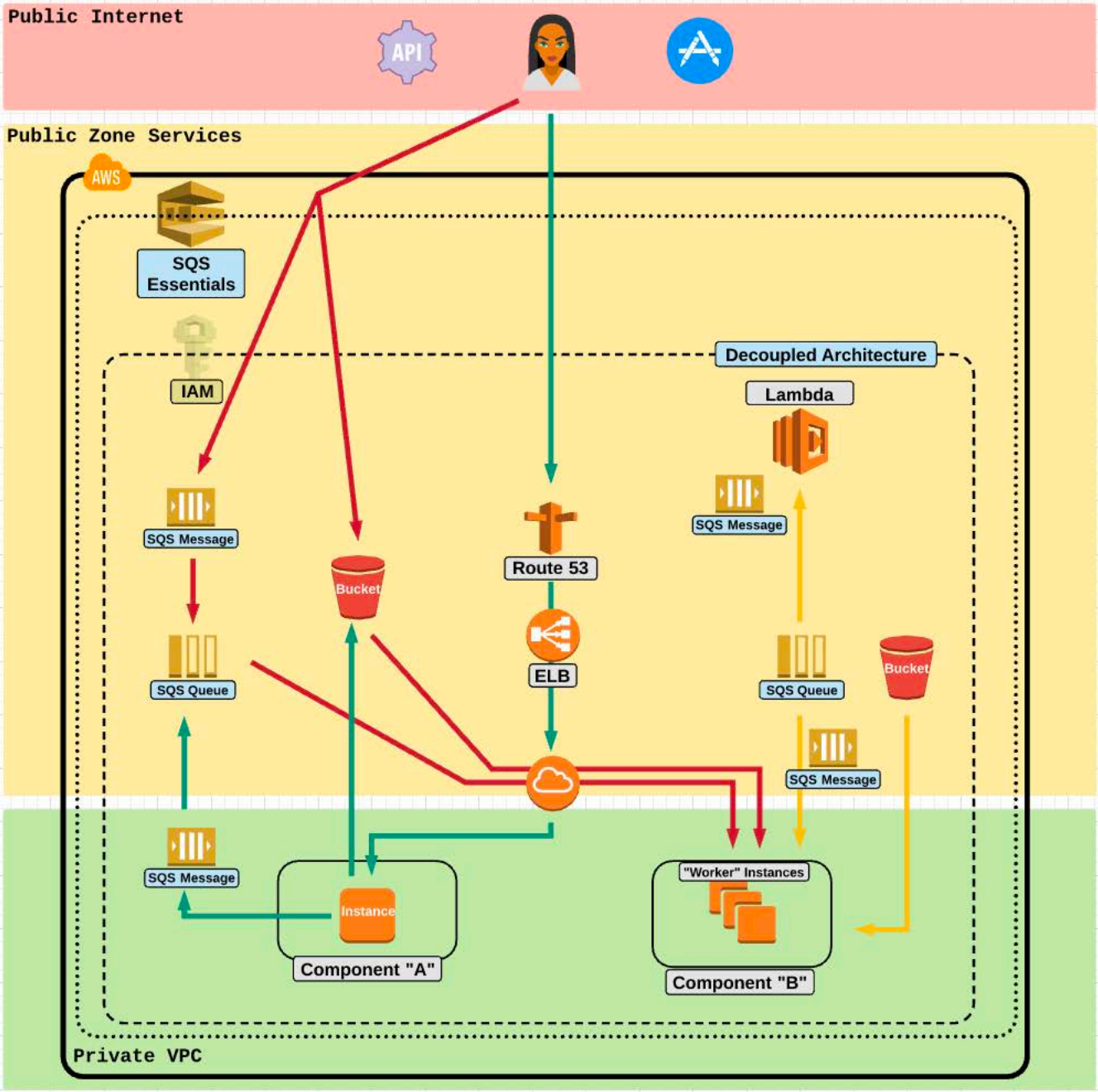
X

SNS Essentials

- SNS coordinates and manages the sending and delivery of messages to subscribers.
- We are able to use SNS to receive notifications when events occur in our AWS environment.
- SNS is integrated into many AWS services, so it is easy to set up notifications based on events that occur in those services.
- With CloudWatch and SNS, a full-environment monitoring solution can be created that notifies administrators of alerts, capacity issues, downtime, changes in the environment, and more.
- This service can also be used for mobile push messages.

SNS Components:

- Topic:
  - Object to which you publish your message (<= 256 KB)
  - Subscribers subscribe to topic to receive message.
- Subscriber:
  - An endpoint to which a message is sent. Messages are simultaneously pushed to subscribers.
  - Available endpoints include:
    - HTTP
    - HTTPS
    - Email
    - Email-JSON
    - SQS
    - Application, mobile app notifications (iOS/Android/Amazon/Microsoft)
    - Lambda
    - SMS (cellular text message)
- Publisher:
  - The entity that triggers the sending of a message
  - Examples include:
    - Application
    - S3 event
    - CloudWatch alarm


Public Internet
API
Public Zone Services
AWS
SQS Essentials
IAM
Decoupled Architecture
Lambda
SQS Message
Bucket
Route 53
ELB
SQS Queue
SQS Message
Bucket
SQS Queue
SQS Message
SQS Message
Instance
Component "A"
"Worker" Instances
Component "B"
Private VPC

Application Services

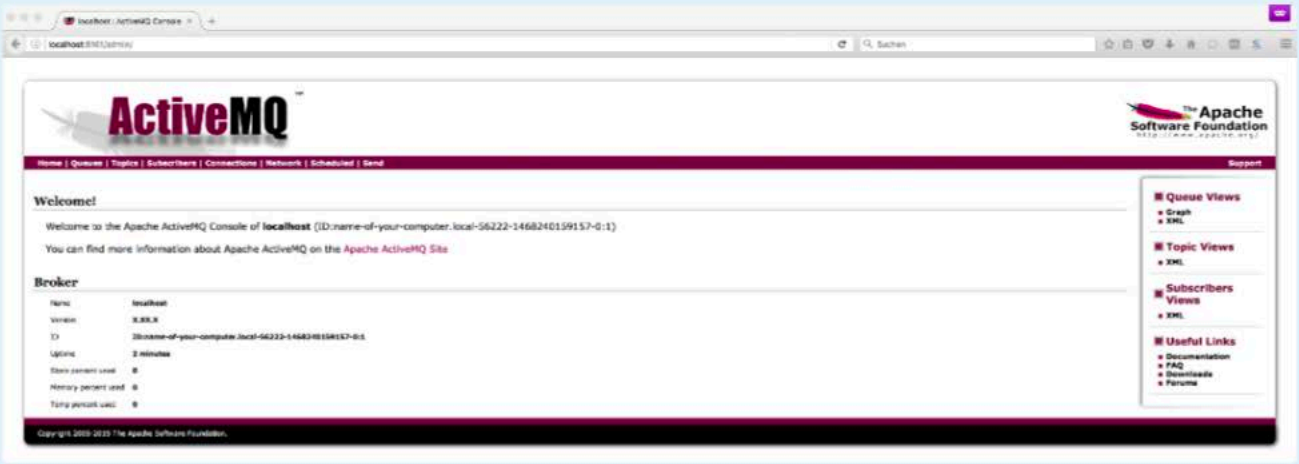
X

Amazon MQ Essentials

- An open-source message broker
- Managed Apache ActiveMQ
- Compatible with JMS API or protocols such as AMQP, MQTT, OpenWire, and STOMP
- Can function as a queue or topic
- One-to-one or one-to-many

MQ Components:

- Broker:
  - Single instance *or*
  - Highly available pair (active/standby)
  - Managed updates
- ActiveMQ Web Console:

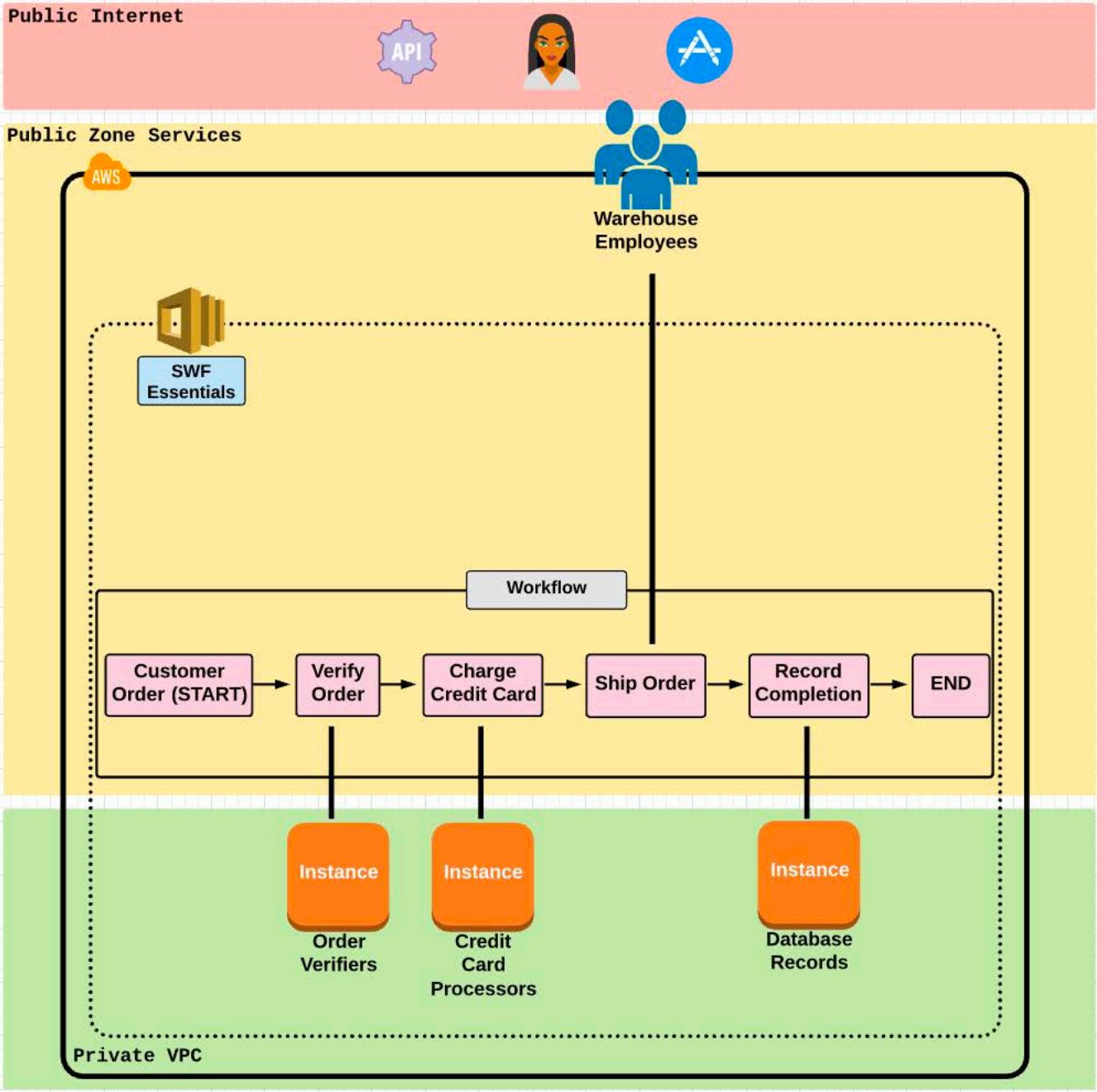


VPC

Private VPC

Application integration services act as supporting functionality to other applications, products, and systems. They generally allow the use of a certain architecture or offload certain parts of application functionality.

Application integration services are generally public zone services, meaning they can be accessed from within private AWS networks or from remote locations over the public internet.


Public Internet
API
Public Zone Services
AWS
Warehouse Employees
SWF Essentials
Workflow
Customer Order (START)
Verify Order
Charge Credit Card
Ship Order
Record Completion
END
Instance
Order Verifiers
Instance
Credit Card Processors
Instance
Database Records
Private VPC

Application Services

X

Simple Workflow Essentials

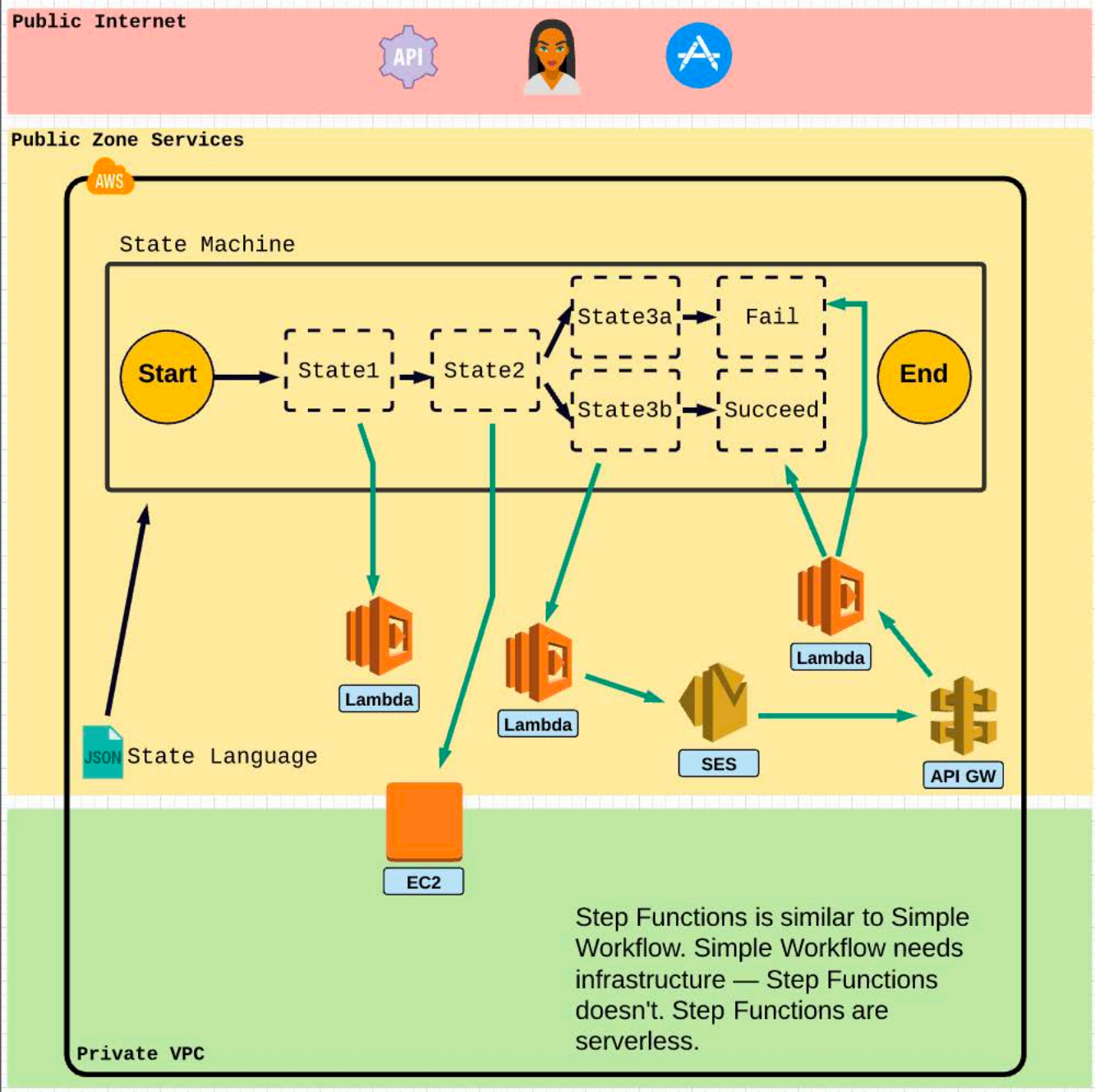
- SWF is a fully managed “workflow” orchestration service provided by AWS.
- A SWF **workflow** allows an architect/developer to implement distributed, asynchronous applications as a workflow.
- A **workflow** coordinates and manages the execution of activities that can be run asynchronously across multiple computing devices.
- SWF has consistent execution.
- Guarantees the order in which tasks are executed.
- There are no duplicate tasks.
- The SWF service is primarily an API into which an application can integrate its workflow service. This allows the service to be used by non-AWS services, such as an on-premises data center.
- A workflow execution can last up to one year.

Components of SWF

- **Starter***:* Starts the workflow
- **Workflow**: A sequence of steps required to perform a specific task
  - A workflow is also commonly referred to as a **decider**.
- **Activities**: A single step (or unit of work) in the workflow
- **Tasks**: What interacts with the “workers” that are part of a workflow
  - Activity task: Tells the worker to perform a function
  - Decision task: Tells the decider the state of the workflow execution, which allows the decider to determine the next activity to be performed
- **Worker**: Responsible for receiving a task and taking action on it
  - Can be any type of component, such as an EC2 instance or even a person

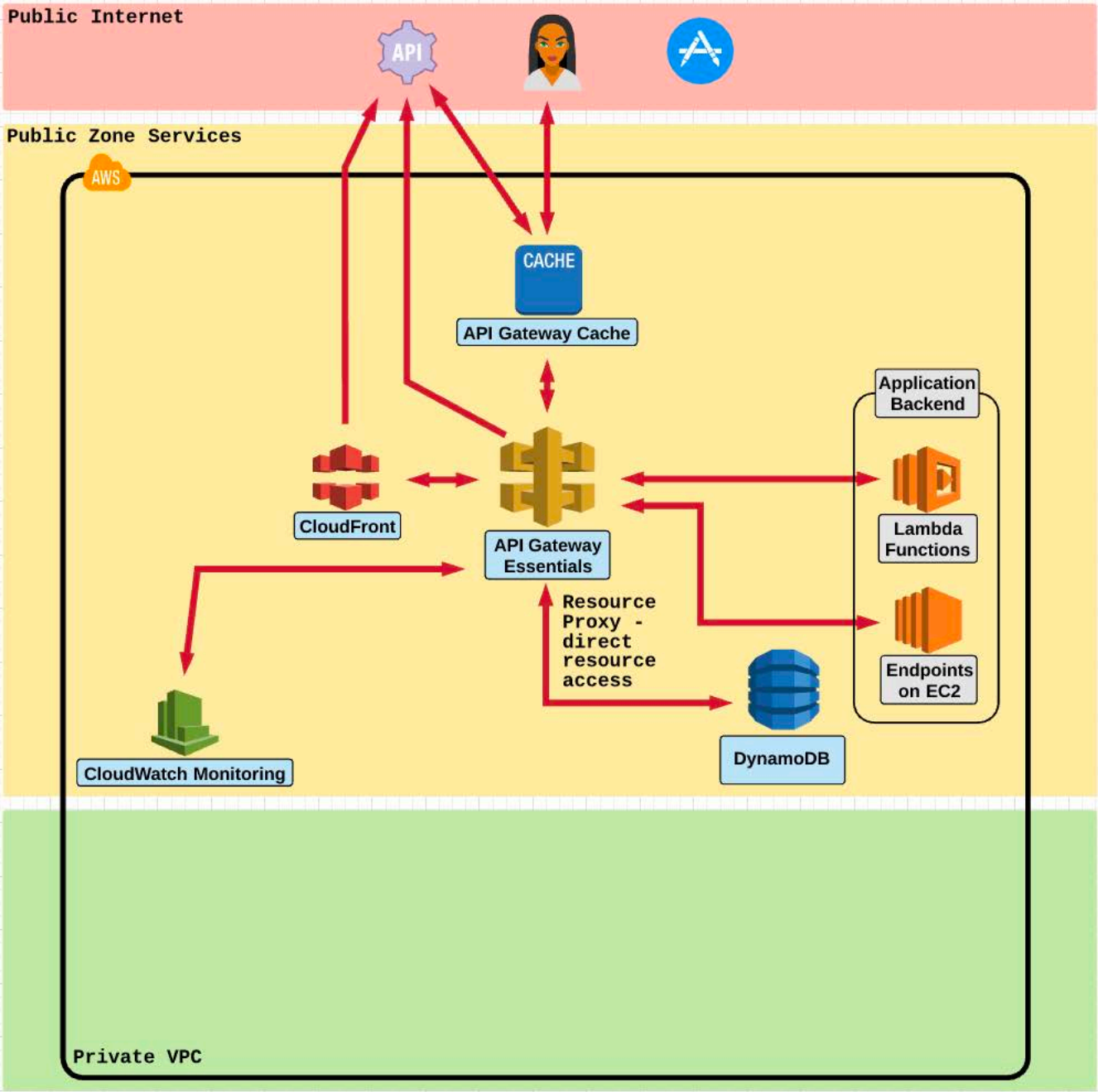
AWS Flow Framework for Java

- Simplifies working with SWF by providing objects and classes



State machines consist of seven types of "state": **Task**, **Choice**, **Parallel**, **Wait**, **Fail**, **Succeed**, and **Pass**. Task states perform the work; Choice adds branching based on input; Parallel splits into multiple or joins data from multiple streams; Wait adds a delay; Fail fails execution; Succeed finishes with a success status; and Pass passes its input to its output.

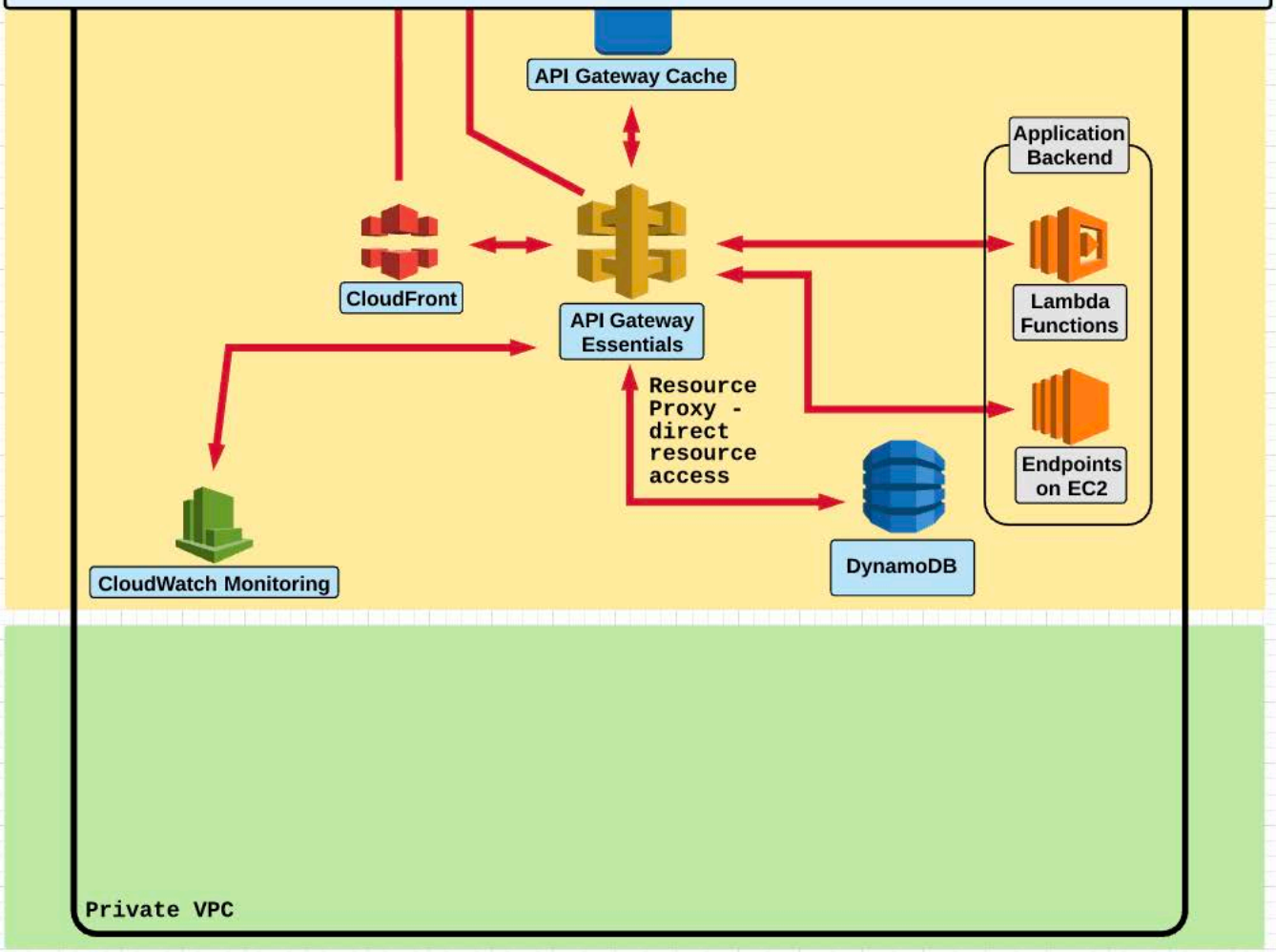
All of the state types are used to construct serverless workflow-style processes.


Public Internet
API
Public Zone Services
AWS
CACHE
API Gateway Cache
Application Backend
CloudFront
API Gateway Essentials
Lambda Functions
Resource Proxy - direct resource access
Endpoints on EC2
CloudWatch Monitoring
DynamoDB
Private VPC

Application Services

API Gateway Caching

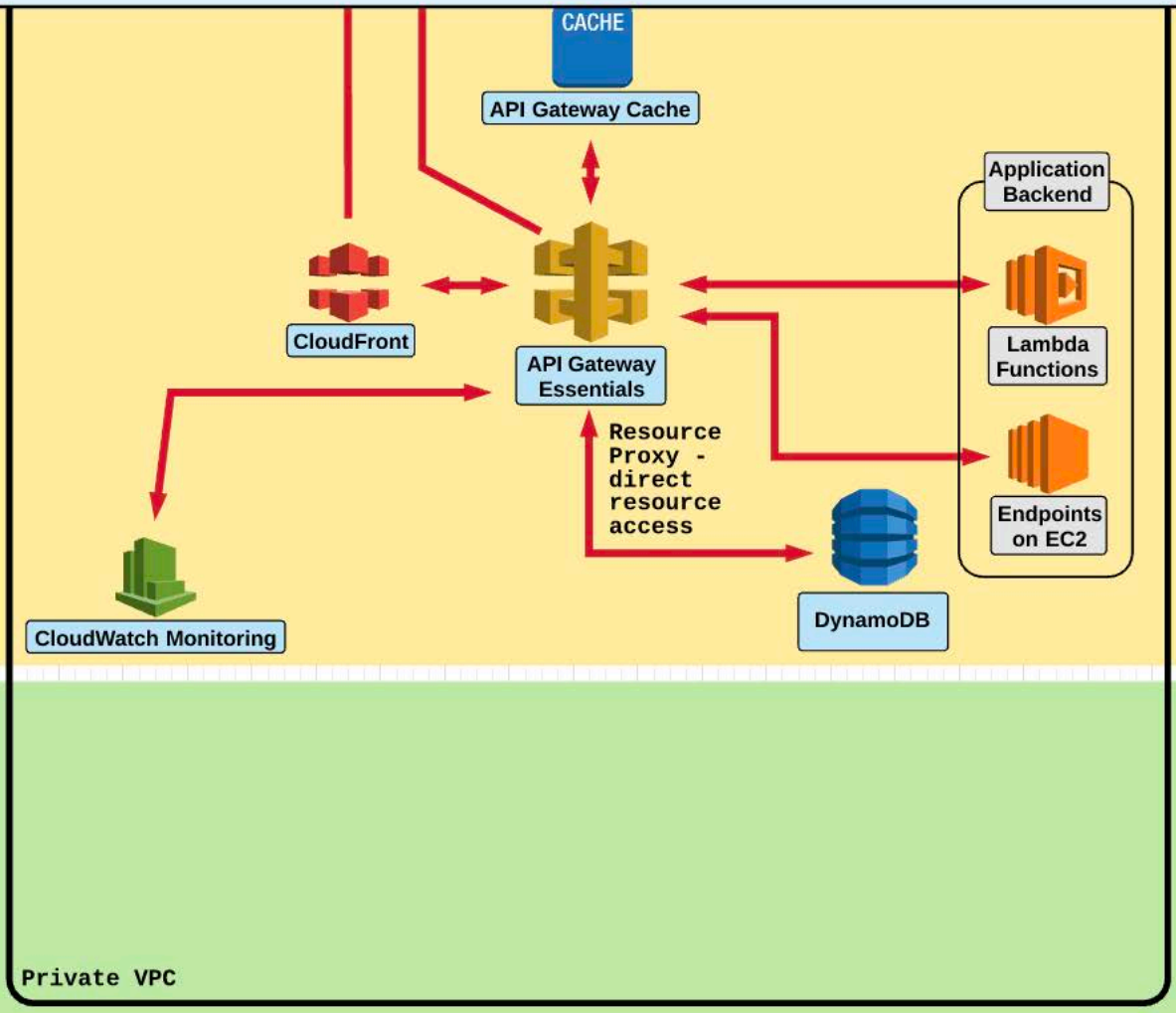
- API Gateway will cache API responses so that duplicate API requests do not have to hit your backend.
  - This reduces load on your backend and speeds up calls to your backend.
- You can configure a cache key and time to live (TTL) of the API response.
- Caching can be set up on a per-API or per-stage basis.



Application Services

API Gateway: CloudFront

- API Gateway benefits from using CloudFront infrastructure:
  - Built-in DDoS attack protection and mitigation
  - All CloudFront Edge Locations become entry points for your API into your backend.
- Summary: Benefits are reduced latency and improved projection.



Application Services

API Gateway Essentials

- API Gateway is a fully managed service that allows you to create and manage your own APIs for your application.
- API Gateway acts as a "front door" for your application, allowing access to data/logic/functionality from your backend services.

API Gateway Main Features

- Build RESTful APIs with:
  - Resources
  - Methods (e.g., GET, POST, PUT)
  - Settings
- Deploy APIs to a "stage" (different environments — e.g., dev, beta, production).
  - Each stage can have its own throttling, caching metering, and logging.
- Create a new API version by cloning an existing one.
  - You can create and work on multiple versions of an API (API version control).
- Roll back to previous API deployments.
  - A history of API deployments are kept.
- Custom domain names
  - Custom domain names can point to an API or stage.
- Create and manage API keys for access *and* meter usage of the API keys through Amazon CloudWatch Logs.
- Set throttling rules based on the number of requests per second (for each HTTP method).
  - Requests over the limit would be throttled (HTTP 429 response).
- Security using Signature Version 4 to sign and authorize API calls
  - Temporary credentials generated through Amazon Cognito and Security Token Service (STS)

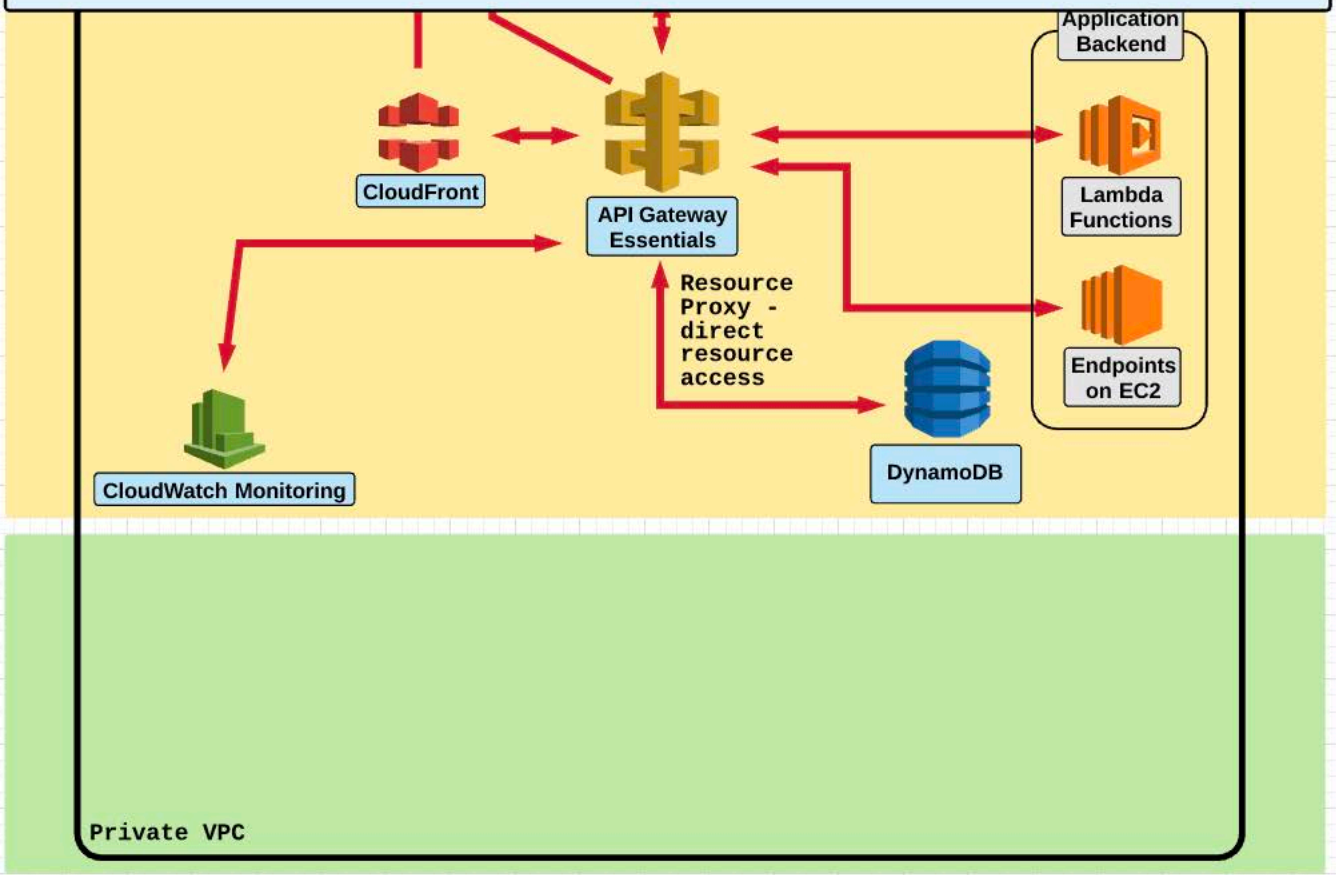
Benefits of API Gateway

- Ability to cache API responses
- DDoS protection via CloudFront
- SDK generation for iOS, Android, and JavaScript
- Supports Swagger (a very popular framework of API dev tools)
- Request/response data transformation (e.g., JSON IN to XML OUT)

Application Services

API Gateway: CloudWatch

- CloudWatch can be used to monitor API Gateway activity and usage.
- Monitoring can be done on the API or stage level.
- Throttling rules are monitored by CloudWatch
- Monitoring metrics include such statistics as:
  - Caching
  - Latency
  - Detected errors
- Method-level metrics can be monitored.
- You can create CloudWatch alarms based on these metrics.



AWS CSA Professional Blueprint

Overview

AWS provides a blueprint for each of their certification exams. You can use this to guide your studies and focus on certain services and concepts within AWS.

Not all services and concepts are tested equally. There are different weights that add up to a total of 100%.

Below is the breakdown for each domain (five total) on the current exam:

DOMAIN NUMBER	DOMAINS	WEIGHT PERCENTAGE
1	Design for Organizational Complexity	12.5
2	Design for New Solutions	31
3	Migration Planning	15
4	Cost Control	12.5
5	Continuous Improvement for Existing Solutions	29
TOTAL		*100*

Check out the full blueprint breakdown here: AWS CSAP Blueprint

AWS Well-Architected Framework

The Well-Architected Framework is a series of best-practice recommendations and questions to ask when designing and developing cloud architectures.

The framework consists of the following five pillars:

Operational Excellence	**Run and monitor systems to provide business value.**
Security	**Protect and monitor systems.**
Reliability	**Recover from failure and mitigate disruptions.**
Performance Efficiency	**Use computing resources efficiently.**
Cost Optimization	**Avoid or eliminate unnecessary expenses.**

Select a resource from above to explore individually or click *Services* to move on.

Services

AWS Well-Architected Framework

Operational Excellence

"The ability to run and monitor systems to deliver business value and to continually improve supporting processes and procedures."

AWS lays out the following six design principles for operational excellence:

1. **Perform operations as code.**
2. **Annotate documentation.**
3. **Make frequent, small, reversible changes.**
4. **Refine operations procedures often.**
5. **Anticipate failure.**
6. **Learn from all operational failures.**

AWS breaks down cloud-based operational excellence into three areas:

1. **Prepare** — Understand your workloads and expected behaviors.
2. **Operate** — Measure success with business achievements and customer outcomes.
3. **Evolve** — Maintain a continuous lifecycle of improvement over time.

"Make me excellent!"

AWS

Check out the operational excellence pillar documentation: OE Pillar

AWS Well-Architected Framework

Operational Excellence

Prepare

You need to be able to understand your workloads and what kinds of behaviors to expect.

There are three areas to consider:

- Operation Priorities
- Design for Operations
- Operational Readiness

Operational Priorities — Understand your workload, roles, and business goals.

Design for Operations — Implement engineering practices to allow for quick, safe fixes.

Operational Readiness — Ensure you have consistent processes and deployments.

Check out the Prepare component in depth here: Whitepaper

AWS Well-Architected Framework

Operational Excellence

Prepare

You need to be able to understand your workloads and what kinds of behaviors to expect.

There are three areas to consider:

X

Utilize AWS services to educate teams involved in your business goals and requirements.

AWS provides the following resources for you:

- AWS Support Center
- AWS Developer Forums
- AWS Knowledge Center

AWS also recommends and provides these services:

- AWS Cloud Compliance
- AWS Trusted Advisor
- Business Support
- Enterprise Support

Trusted Advisor

Operational Priorities

Design for Operations

Operational Readiness

Ensure you have consistent processes and deployments.

Check out the Prepare component in depth here: Whitepaper

AWS Well-Architected Framework

Operational Excellence

Prepare

You need to be able to understand your workloads and what kinds of behaviors to expect.

There are three areas to consider:

- **Operation Priorities**
- **Design for Operations**
- **Operational Readiness**

X

Operational Priorities

Design for Operations

Operational Readiness

Your workload design should include the following:

- How you will deploy it
- How you will update it
- How you will keep it operating

AWS suggests that you treat your workload as code.

AKA Infrastructure as Code!

They also suggest setting up CI/CD deployment pipelines.

That last major recommendation is to enable custom metric logging.

CloudFormation

AWS Key Services:

- AWS CloudFormation
- AWS Developer Tools
- AWS X-Ray
- Amazon CloudWatch

X-Ray

Check out the Prepare component in depth here: Whitepaper

AWS Well-Architected Framework

Operational Excellence

Prepare

You need to be able to understand your workloads and what kinds of behaviors to expect.

There are three areas to consider:

- Operation Priorities
- Design for Operations
- Operational Readiness

X

Operational Priorities

Design for Operations

Operational Readiness

Use consistent processes to mark when you are able to go live with your workload!

Use some type of checklist.

Script your deployments and utilize parallel environments.

Ensure your team can actually run and support your workload!

Config

Systems Manager

AWS Key Services

- AWS Config
- Amazon EC2 Systems Manager
- AWS Lambda

Lambda

Check ou

AWS Well-Architected Framework

Operational Excellence

Operate

Success in this area is "measured by the outcomes and metrics you define."

You need to consider two main concepts:

- Understanding Operational Health
- Responding to Events

Grasping both of these points leads to quicker identification of impacts to your workload and faster response times.

Operational Health — Understand your workload's health easily!

Respond to Events — Anticipate events — both planned and unplanned.

Check out the Operate component in depth here: White Paper

AWS Well-Architected Framework

Operational Excellence

Operate

Success in this area is "measured by the outcomes and metrics you define."

You need to consider two main concepts:

- Understanding Operational Health
- Responding to Events

Grasping both of these points

X

Understand your workload!

Use the appropriate metrics for your needs and requirements.

AWS Example: Send CloudWatch logs to Elasticsearch, and use Kibana to visualize your operational health.

AWS Key Services

- Amazon CloudWatch Logs
- Amazon ES
- Personal Health Dashboard
- Service Health Dashboard

Operational Health

Respond to Events

Anticipate events — both planned and unplanned.

Check out the Operate component in depth here: White Paper

AWS Well-Architected Framework

Operational Excellence

Operate

Success in this area is "measured by the outcomes and metrics you define."

You need to consider two main concepts:

- Understanding Operational Health

Grasping both of these points

Operational Health

Respond to Events

X

Design your workloads with events in mind!

Planned events: site promotions, advertising traffic, or DR tests.

Unplanned events: AZ losses and hardware issues.

Isolate your resources to determine how business will be impacted if any of them go down.

Do your best to automate processes and fixes!

Key AWS Services

- Amazon CloudWatch
- CloudWatch Events
  - Amazon SNS
  - Auto Scaling
- EC2 Systems Manager
- AWS Lambda

Check out the Operate component in depth here: White Paper

AWS Well-Architected Framework

Operational Excellence

Evolve

AWS defines the evolution component as a "continuous cycle of improvement over time."

You should always try to improve your workloads and architectures by making small, incremental changes. Don't make a few massive changes.

AWS lays out the following two principles:

- Learning from Experience
- Sharing Learnings

Having a firm grasp of these topics will help your operations evolve smoothly over time!

Learning from Experience — Learn from mistakes and failures!

Sharing Learnings — Always be sure to participate in knowledge sharing.

Check out the Evolve component in depth here: Whitepaper

AWS Well-Architected Framework

Operational Excellence

Evolve

AWS defines the evolution component as a "continuous cycle of improvement over tim

X

You should always try to imp
incremental cha

AWS la

Having a firm grasp of these to

Learning from Experience

Sharing Learnings

Learn from all of your failures because they *will* happen. Analyze failures, and then work on improvements.

AWS recommends using mirrored, temporary environments for testing deployments, etc.

Use *all* logging services, such as CloudTrail and CloudWatch.

Lastly, AWS suggests you perform cross-team reviews that cover *all* aspects of the business.

S3

Athena

AWS Key Services

- Amazon QuickSight
- Amazon Athena
- Amazon S3
- Amazon CloudWatch
- Amazon ES

QuickSight

CloudWatch

Check out the Evolve component in depth here: Whitepaper

AWS Well-Architected Framework

Operational Excellence

Evolve

AWS defines the evolution component as a "continuous cycle of improvement over time."

You should always try to improve your workloads and architectures by making small incremental changes. Don't make a few massive changes.

AWS la

Having a firm grasp of these to

Learning from Experience

Sharing Learnings

Check out the

X

All knowledge learned should be shared!

Knowledge sharing helps you avoid future challenges that may affect your workloads.

Once again, AWS encourages you to treat infrastructure as code. This allows for easier sharing.

Identity and Access Management (IAM) allows you to define permissions on a granular level for what you want to share.

Remember the least privilege principle!

IAM

SNS

AWS Key Services

- IAM
- Amazon SNS
- AWS CodeCommit
- AWS Lambda
- AMIs

CodeCommit

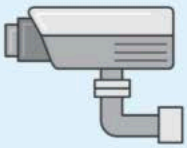
AWS Well-Architected Framework

Security

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

The security pillar involves protecting all information and assets while ensuring business risks are low and mitigated rapidly.

AWS recommends the following guidelines for maintaining strong security:



1. Implement a strong identity foundation.
2. Enable traceability.
3. Apply security at every layer.
4. Automate security.
5. Protect data in transit and at rest.
6. Prepare for security events.

Review the Shared Security Model documentation: LINK

Shared Responsibility Model

AWS breaks down security into five areas:

IAM | **Detective Controls** | **Infrastructure Protection** | **Data Protection** | **Incident Response**

AWS Well-Architected Framework

Security

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

The security pillar involves protecting all information and assets while ensuring business risks are low and mitigated rapidly.

AWS recommends the following guidelines for maintaining strong security:

1. Implement a strong identity foundation.
2. Enabl
3. Apply
4. Autom
5. Protec
6. Prepa

X

Implement the principle of *least* privilege.

Separation of duties!

Enforce centralized privilege management.

Review the Shared Security Model documentation: LINK

Shared Responsibility Model

AWS breaks down security into five areas:

IAM | **Detective Controls** | **Infrastructure Protection** | **Data Protection** | **Incident Response**

AWS Well-Architected Framework

Security

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

The security pillar involves protecting all information and assets while ensuring business risks are low and mitigated rapidly.

AWS recommends the following guidelines for maintaining strong security:

1. Implement a strong identity foundation.
2. Enable traceability.
3. Apply security at every
4. Automate security.
5. Protect data in transit a
6. Prepare for security eve

X

Set up monitoring, alerts, and auditing for any changes or activities within your environments!

In addition, heavily leverage logging and metrics services.

Review the Shared Security Model documentation: LINK

Shared Responsibility Model

AWS breaks down security into five areas:

IAM | **Detective Controls** | **Infrastructure Protection** | **Data Protection** | **Incident Response**

AWS Well-Architected Framework

Security

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

The security pillar involves protecting all information and assets while ensuring business risks are low and mitigated rapidly.

AWS recommends the following guidelines for maintaining strong security:

1. Implement a strong identity foundation.
2. Enable traceability.
3. Apply security at every layer.
4.
5.
6.

X

You should take a layered defense (i.e., "defense in depth") approach to security.

***No* service or resource should be exempt from this practice.**

Review the Shared Security ... **red Responsibility Model**

AWS breaks down security into five areas:

IAM | Detective Controls | Infrastructure Protection | Data Protection | Incident Response

AWS Well-Architected Framework

Security

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

The security pillar involves protecting all information and assets while ensuring business risks are low and mitigated rapidly.

AWS recommends the following guidelines for maintaining strong security:

1. **You should leverage software-based security platforms and mechanisms.**

   Any code-based deployments should take into account architecture security and controls.
2.
3.
4. Automate security.
5. Protect data in transit and at rest.
6. Prepare for security events.

Review the Shared Security Model documentation: LINK **Shared Responsibility Model**

AWS breaks down security into five areas:

IAM | **Detective Controls** | **Infrastructure Protection** | **Data Protection** | **Incident Response**

AWS Well-Architected Framework

Security

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

The security pillar involves protecting all information and assets while ensuring business risks are low and mitigated rapidly.

AWS recommends the following guidelines for maintaining strong security:

1. **You should classify and categorize data by sensitivity.**

   After classification, you should use encryption, tokenization, etc., where necessary.

   Reduce human access to data streams.

   X
2.
3.
4.
5. Protect data in transit and at rest.
6. Prepare for security events.

Review the Shared Security Model documentation: LINK **Shared Responsibility Model**

AWS breaks down security into five areas:

IAM | **Detective Controls** | **Infrastructure Protection** | **Data Protection** | **Incident Response**

AWS Well-Architected Framework

Security

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

The security pillar involves protecting all information and assets while ensuring business risks are low and mitigated rapidly.

AWS recommends the following guidelines for maintaining strong security:

1. Implement a strong identity foundation.
2. Enable traceability.
3.
4.
5.
6. Prepare for security events.

X

Have an incident management process set up!

Test for situations that may occur, such as DR or application failures.

Review the Shared Security Model documentation: LINK

Shared Responsibility Model

AWS breaks down security into five areas:

IAM | **Detective Controls** | **Infrastructure Protection** | **Data Protection** | **Incident Response**

AWS Well-Architected Framework

Security

Identity and Access Management (IAM)

IAM

IAM is one of the most critical components of AWS!

You control *all* access and authentication to your resources this way.

The AWS documentation discusses best practices for implementing and using IAM in the following sections:

- Protecting AWS Credentials
- Fine-Grained Authorization

Protecting AWS Credentials

You want to protect all credentials within your account.

Follow best practices such as rotating access keys, enforcing password policies, and requiring MFA.

MFA

STS

Role

Fine-Grained Authorization

Always follow the principle of least privilege.

Specifically define roles that are required for users and applications.

Use AWS Organizations as a central account management point.

Organizations

Role

Defined Needs

AWS Well-Architected Framework

Security

Detective Controls

You can use detective controls to identify potential threats.

Using these tools can help you meet compliance requirements.

You want to keep track of all resources and audit *all* changes.

The AWS docs lay out a few ways to accomplish this in the following sections:

- Capture and Analyze Logs
- Integrate Auditing Controls with Notification and Workflow

Capture and Analyze

AWS allows for easier asset management. You can programmatically describe instances, monitor logs, etc.

You can use API-driven services to easily collect, filter, and analyze logs.

Config
CloudWatch
S3
CloudTrail
elasticsearch
EMR
Athena

Auditing Controls

Logging is a *critical* tool for analyzing risks and correcting misbehaviors.

AWS best practice is to integrate security logging with notifications and event handling.

CloudWatch
Event
Inspector
Rule
Config

AWS Well-Architected Framework

Security

Infrastructure Protection

This entails all control methods needed to comply with your industry best practices and AWS guidelines.

You want to protect against unauthorized access and attacks.

AWS separates Infrastructure Protection into three parts:

- Protecting Network and Host-Level Boundaries
- System Security Configuration and Maintenance
- Enforcing Service-Level Protection

Protection

Design your topology carefully.

Consider public vs. private components.

VPC

Direct Connect

Config and Maintenance

Keep a strict security posture!

Always maintain least privilege principles.

Automate security procedures.

Inspector

Systems Manager

Automation

Enforcement

Protect service endpoints via IAM.

Least privilege principle!

IAM policies are critical here.

KMS

S3

AWS Well-Architected Framework

Security

Data Protection

Data and resource protection should be the foundation of your architecture's design.

AWS breaks down Data Protection into five categories:

- Data Classification
- Encryption/Tokenization
- Protecting Data at Rest
- Protecting Data in Transit
- Data Backup/DR

Data Classification

Classify data based on sensitivity.

Macie

KMS

Encryption

Encrypt data!

Scope out tokenization approaches.

DynamoDB

CloudHSM

KMS

Data at Rest

Any data persisting for *any* duration.

Encryption and access control!

S3

KMS

EBS

Data in Transit

Any data moving from one system to another — *not* just external.

Encryption, VPNs, certificates.

CloudFront

ELB

ACM

Backups and DR

Define your backup, replication, and recovery approach!

Automate when you can.

Test your backups!

EBS

S3

AWS Well-Architected Framework

Security

Incident Response

All organizations should implement a response and mitigation plan.

Nobody is immune!

Teams should aim to be proactive, not reactive.

AWS provides one main approach for this:

- Clean Room

Clean Room

IAM

Teams should maintain situational awareness!

Tag your resources.

CloudFormation

If possible, form an incident response team.

Use APIs to automate your routine tasks.

EC2

Be able to identify root causes for incidents.

Step Functions

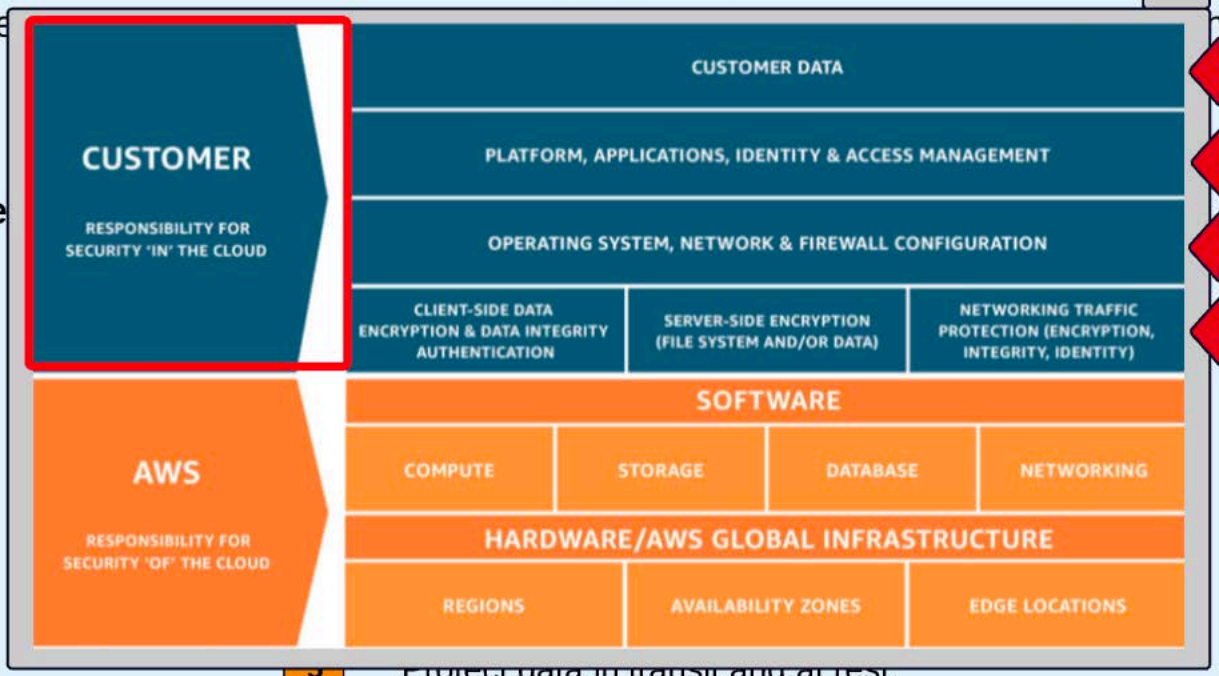
AWS Well-Architected Framework

Security

X

"The ... ness value

The ... ensuring



5 Protect data in transit and at rest.

6 Prepare for security events.

Review the Shared Security Model documentation: LINK

Shared Responsibility Model

AWS breaks down security into five areas:

- **IAM**
- **Detective Controls**
- **Infrastructure Protection**
- **Data Protection**
- **Incident Response**

AWS Well-Architected Framework

Reliability

"The ability to recover from and mitigate service disruptions."

AWS stresses the importance of service availability within your architecture. You need to know how to calculate availability (five nines!) as well as maintain the most cost-efficient architectures.

The reliability pillar is split into a few key points:

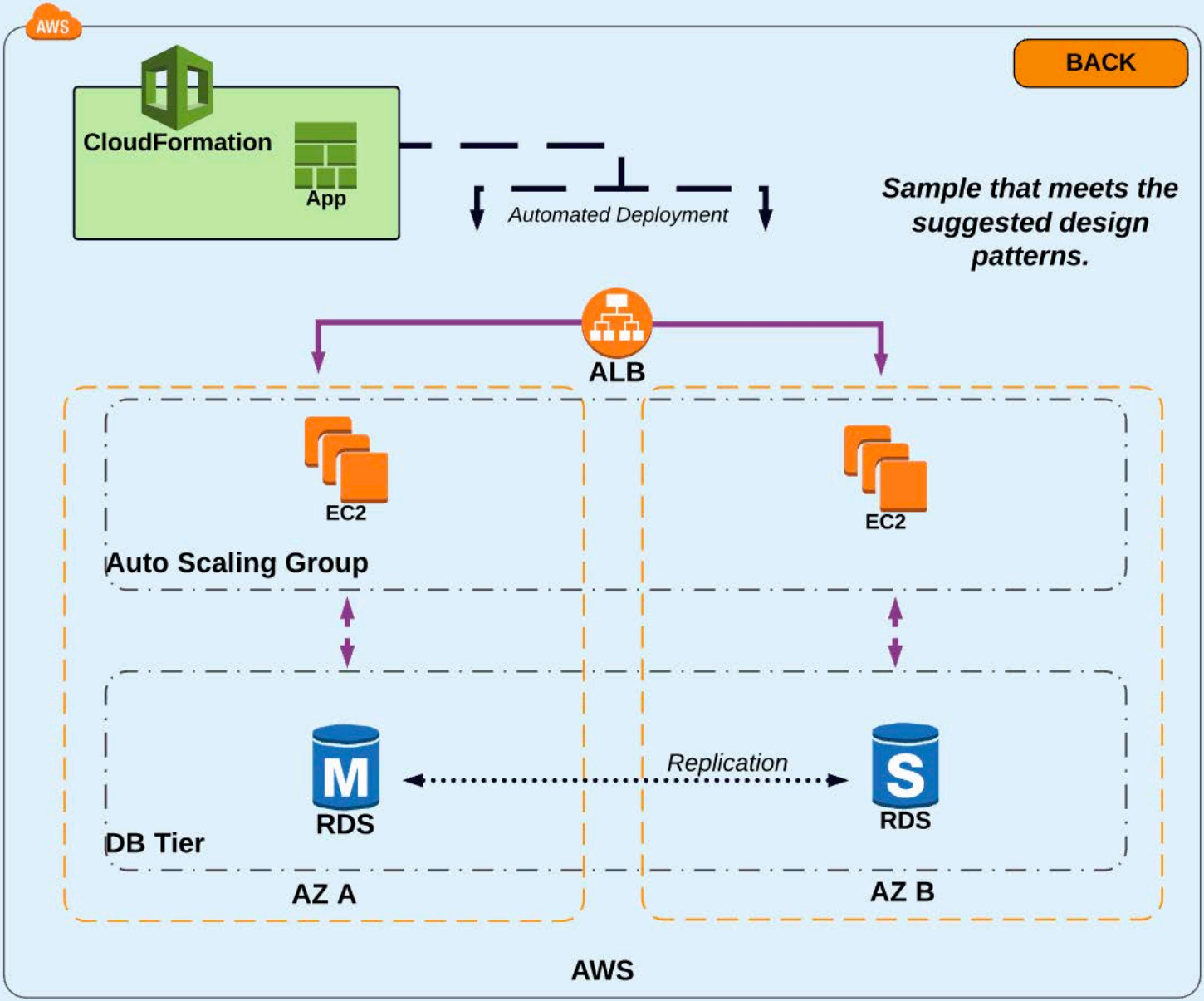
Example

- Always test your recovery solutions.
- Design with self-healing in mind.
- Architect around the ability to scale dynamically.
- Focus on automating all deployments.
- Enable monitoring and alarming.
- Calculate your resource needs. Don't guess!
- Remember least privilege!

Key Services

Check out the reliability pillar documentation: Reliability Pillar

AWS Well-Architected Framework



AWS Well-Architected Framework

Performance Efficiency

"The ability to use computing resources efficiently to meet system requirements and to maintain that efficiency as demand changes and technologies evolve."

Don't reinvent the wheel! AWS can help you deploy efficient services.
You can reach users worldwide with a single click.
Use event-driven architectures when possible!

AWS defines four areas of performance efficiency:

- **Selection**
- **Review**
- **Monitoring**
- **Trade-offs**

Selection Choose the right instance and storage options.

Review Re-evaluate when AWS announces new features and services.

Monitoring Verify that resources perform as expected.

Trade-Offs Consider caching and read replicas.

Check out the performance efficiency pillar documentation: Performance Efficiency

AWS Well-Architected Framework

Performance Efficiency

Selection

Selection involves implementing the correct solution based on your specific workload needs.

There is no one-size-fits-all solution for architectures.

It's important to optimize compute resources for many reasons, including cost and performance. There are different instance types for a reason. Some are better suited than others for certain tasks.

Four areas of selection:

- Compute
- Storage
- Databases
- Network

Compute

Auto Scaling

Storage

S3

EBS

Databases

DynamoDB

RDS

Network

VPC

Route 53

Direct Connect

Read more about the selection component here: Selection

AWS Well-Architected Framework

Performance Efficiency

Review

Continuously reviewing your architecture is critical.

Sooner or later, AWS will release new technologies and services that make your existing architecture less efficient than it could be. It will happen!

AWS recommends a data-driven approach with scheduled performance reviews.

Implement code deployment pipelines! Automate things when you can.

The two key sections here are:

- **Benchmarking**
- **Load Testing**

Benchmarking Test to provide useful data about performance.

Load Testing Use real workloads to get realistic load test results.

Read more about the review component here: Review

AWS Well-Architected Framework

Performance Efficiency

Review

Continuously reviewing your architecture is critical.

Sooner or later, AWS will release new technologies and services that make your existing architecture less efficient than it could be. It will happen!

AWS recommends a data-driven approach with scheduled performance reviews.

Implement code deployment pipelines! Automate things when you c

X

Benchmarking is usually done at the start.

Usually quicker than load testing.

Pre-warm your resources so that results will be realistic.

CloudFormation

CodeDeploy

Benchmarking

Load Testing

Use real workloads to get realistic load test results.

Read more about the review component here: Review

AWS Well-Architected Framework

Performance Efficiency

Review

Continuously reviewing your architecture is critical.

Sooner or later, AWS will release new technologies and services that make your existing architecture less efficient than it could be. It will happen!

AWS recommends a data-driven approach with scheduled performance reviews.

Implement code deployment pipelines! Automate things when you can.

The two key sections here are:

- Benchmarking

X

Benchmarking

Load Testing

Test real workloads in order to gain an understanding of production performance.

Automate tests during code deployment pipeline.

Use metrics! CloudWatch is king.

Parallelize your processes and workloads if possible.

CloudWatch

Read more about the review component here: Review

AWS Well-Architected Framework

Performance Efficiency

Monitoring

After implementing your solutions, you need to monitor their performance.

Set up alarms so you know when thresholds are breached.

Be proactive in your approach — not reactive.

AWS breaks down monitoring into two sections:

- **Active and Passive**
- **Phases**

Active and Passive — Monitoring falls into one of these two categories.

Phases — Five categories fall under this section.

Read more about the monitoring component here: Monitoring

AWS Well-Architected Framework

Performance Efficiency

Monitoring

X

After implementing y

Set up alarm

Be pro

AWS br

Active and Passive

Phases

Active Monitoring

Continuously simulating user activity.

Should be continuous, lightweight, and predictable.

Can be run in *all* environments!

CloudWatch

Passive Monitoring

Common for web-based workloads.

Collects metrics from a browser.

Used to gain understanding of:

- UX
- Geo performance
- API impact

Read more about the monitoring component here: Monitoring

AWS Well-Architected Framework

Performance Efficiency

Monitoring

After implementing your solutions, you need to monitor their performance.

Set up alarm

Be pro

AWS br

Active and Passive

Phases

X

There are five phases within this category:

Generation

Your scope of monitoring, metrics, and thresholds

Aggregation

Complete view from all sources

CloudWatch

Real-Time Processing and Alarming

Recognize and respond to events

Alarms

Storage

Data management and retention policies

S3

Analytics

Dashboards, reporting, and insights

EMR

Read more about the monitoring component here: Monitoring

AWS Well-Architected Framework

Performance Efficiency

Trade-Offs

Consider possible trade-offs to determine optimal approaches.

For example, you may have to sacrifice a bit of durability for higher performance.

AWS allows for global deployments within a few minutes!

AWS breaks down trade-offs into four areas:

- **Caching**
- **Partitioning or Sharding**
- **Compression**
- **Buffering**

Caching

Increase performance using caching.

Compression

Compress to reduce resource requirements.

Partition/Shard

Split data across multiple schemas.

Buffering

Leverage queues for different rate times.

Read more about the trade-offs component here: Trade-Offs

AWS Well-Architected Framework

X

Databases are a key resource for maintaining truthful data.

Caching improves database efficiency.

There are a few different levels:

Application Level

Trade-off at app level by using app-level caches (in-memory caching).

Database Level

Use read replicas if possible.

Geographic Level

CDN usage!

RDS **ElastiCache** **CloudFront**

Caching

Increase performance using caching.

Compression

Compress to reduce resource requirements.

Partition/Shard

Split data across multiple schemas.

Buffering

Leverage queues for different rate times.

Read more about the trade-offs component here: Trade-Offs

AWS Well-Architected Framework

X

Trade longer computing times for reduced storage and network requirements.

Compression can happen at different levels, including the file system, data files, and even web-based resources.

CloudFront is excellent for this, as it can compress at the edge location!

Consider non–network-based data transfer solutions for large-scale jobs.

Redshift allows for better compressions due to the column-based storage.

Amazon Redshift | **Amazon CloudFront** | **Edge** | **AWS Snowball**

Caching

Increase performance using caching.

Compression

Compress to reduce resource requirements.

Partition/Shard

Split data across multiple schemas.

Buffering

Leverage queues for different rate times.

Read more about the trade-offs component here: Trade-Offs

AWS Well-Architected Framework

X

Some RDS implementations are limited to vertical scaling only.

Once you reach the vertical scaling limitations, consider sharding data.

Sharding into multiple schemas offers performance gains for write-heavy loads.

Any sharding changes require work at the application layer as well.

DynamoDB does this for you!

Amazon DynamoDB

Amazon DynamoDB Accelerator

Caching

Increase performance using caching.

Compression

Compress to reduce resource requirements.

Partition/Shard

Split data across multiple schemas.

Buffering

Leverage queues for different rate times.

Read more about the trade-offs component here: Trade-Offs

AWS Well-Architected Framework

X

Buffering leverages queues for accepting work.

Always use durable storage when you can.

Queuing allows you to go at the customer's required pace.

Amazon SQS is the main service for queue-based messaging.

You can also use Kinesis since it uses data streams.

When using queues, be sure to keep latency requirements in mind.

Amazon SQS

Amazon Kinesis

Caching

Increase performance using caching.

Compression

Compress to reduce resource requirements.

Partition/Shard

Split data across multiple schemas.

Buffering

Leverage queues for different rate times.

Read more about the trade-offs component here: Trade-Offs

AWS Well-Architected Framework

Cost Optimization

"The ability to avoid or eliminate unneeded cost or suboptimal resources."

Cost optimization is a continuous process of refinement.

To be fully optimized, you should be fully utilizing *all* resources.

AWS has five design principles to follow:

- Adopt a consumption model
- Measure overall efficiency
- Stop spending money on data centers
- Analyze and attribute expenditure
- Use managed services

Cost-Effective Resources — Choosing the right instance and storage options

Matching Supply and Demand — Scale according to load

Expenditure Awareness — Use cost allocation tags

Optimizing Over Time — Continually reevaluate

AWS Well-Architected Framework

Performance Efficiency

Cost-effective Resources

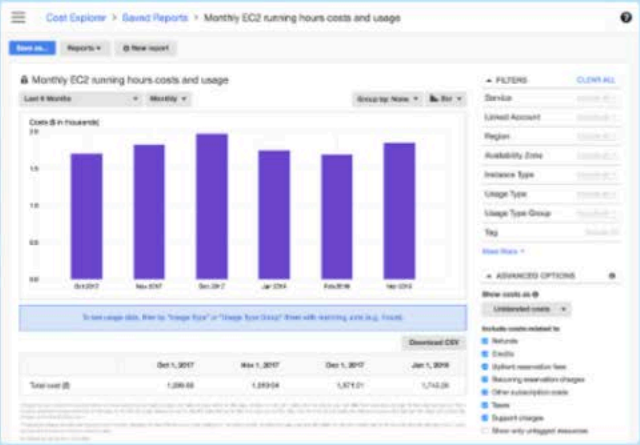
This is all about choosing the correct services and resources for your use case.

You can leverage AWS discounted resources like spot and reserved instances.

Be sure to use AWS geographic tools as well.

Managed services take care of a lot of overhead spending! Use them!

Key Services



Amazon CloudFront

AWS Lambda

Amazon SES

Amazon SQS

Amazon Route 53

Spot instance

Amazon SNS

AWS Well-Architected Framework

Performance Efficiency

Meeting Supply and Demand

The inviting feature about the cloud is the "pay for what you use" pricing points.

There is a fine line between accurately planning for need and still maintaining high availability, etc.

AWS lays out three main approaches for us:

- Demand based
- Buffer based
- Time based

Demand Based

Using this approach usually entails leveraging Auto Scaling services.

You want to focus on elasticity here.

You can leverage CloudWatch metrics to scale accurately.

Auto Scaling | Amazon CloudWatch | Application Load Balancer

Buffer Based

This approach uses a queue to buffer your requests.

SQS is a great choice for this.

You need to keep in mind acceptable delay times here.

Amazon SQS | Amazon Kinesis | Amazon DynamoDB

Demand Based

Use this when you have a predictable workload timing.

You can schedule resources to scale during the highest point of need.

Make sure your workload is consistent!

Auto Scaling | AWS CloudFormation

AWS Well-Architected Framework

Performance Efficiency

Expenditure Awareness

The key here is to understand your business cost drivers!

Be aware of where your costs are coming from. It is easy to lose track when you have multiple environments deployed.

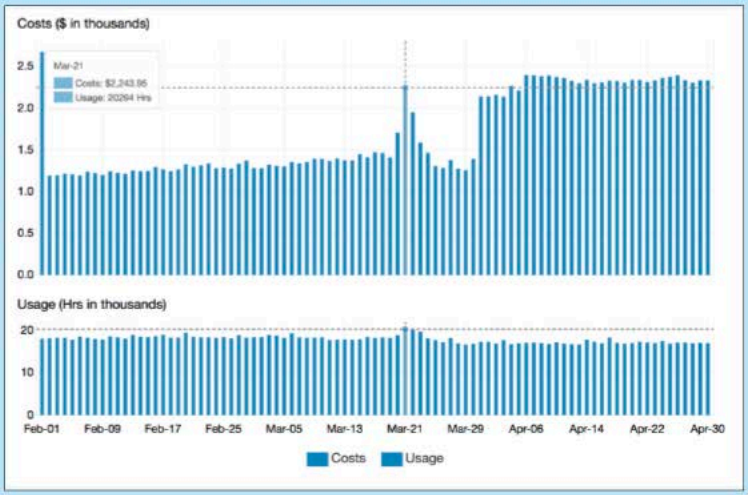
Keep in mind the following key factors:

- **Stakeholders**
- **Visibility and Controls**
- **Cost Attribution**
- **Tagging**
- **Entity Lifecycle Tracking**

Structure your accounts appropriately.

***Always* use resource tagging!**

Use Cost Explorer to track your costs.



AWS Well-Architected Framework

Performance Efficiency

Optimizing Over Time

AWS encourages you to use different approaches over time.

You want to always be checking your implementations for potentially better solutions.

- Measure, Monitor, Improve
- Staying Ever Green

Measure, Monitor, Improve

There are four key concepts:

- Establish a cost optimization function
- Establish goals and metrics
- Gather insight and perform analysis
- Report and validate

Plenty of tools are available for you to use to perform these tasks.

Amazon CloudWatch

Auto Scaling

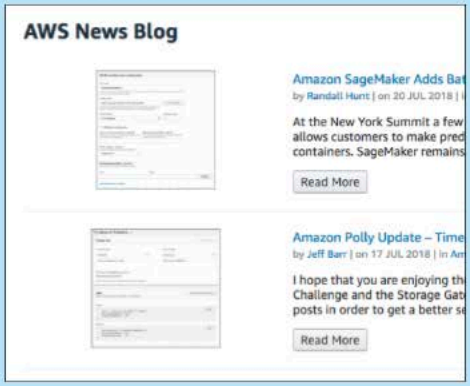
AWS Trusted Advisor

Staying Ever Green

This simply means to *always* review the new services offered.

Leverage managed services when you can!

AWS Trusted Advisor



Appendix

Welcome to the Appendix for the Orion Papers

Here you will find helpful resources and links to help you explore AWS.

Select a resource in the navigation panel above to explore the different parts of this appendix.

Appendix

What Is a Solutions Architect?

- In a broad sense, a solutions architect is responsible for interpreting customer requirements and defining a solution following architectural design principles.
- They provide implementation guidance based upon best practices throughout the lifecycle of the project.
- Being a successful solutions architect on AWS means being competent in the following areas:
  - Designing resilient architectures that are highly available and/or fault tolerant
  - Achieving maximum performance efficiency by leveraging caching and the inherent scalability and elasticity of AWS
  - Identifying the most cost-efficient solutions for compute and storage
  - Knowing how to secure applications, data, and networks
  - Choosing design features that enable operational excellence

A cloud architect is an IT professional who is responsible for overseeing a company's cloud computing strategy. This includes cloud adoption plans, cloud application design, and cloud management and monitoring. Cloud architects oversee application architecture and deployment in cloud environments — including public cloud, private cloud, and hybrid cloud. Additionally, cloud architects act as consultants to their organization and need to stay up to date on the latest trends and issues.

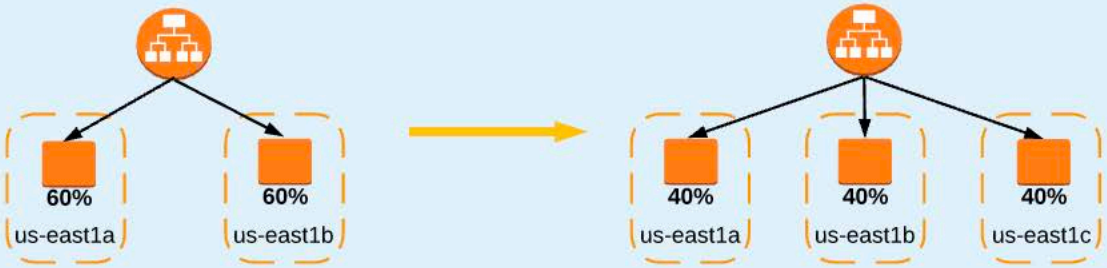
—TechTarget.com

Appendix

CSA Terminology

High Availability
Refers to architectures that continue to remain available to end users in the event of a component or systems failure. On AWS, multi-AZ architectures allow your applications to remain available in the event of an AZ outage.

Fault Tolerance
Refers to architectures that not only remain available during an outage (high availability) but also suffer no degradation in performance. Fault-tolerant architectures usually require extra redundancy and should be traded off with cost concerns.



Scalability
Refers to the ability of a system to easily increase in size and capacity in a cost-effective way (usually based on usage demand). Scaling can be **vertical** (increase the capacity of a single instance or server) or **horizontal** (add or terminate the number of instances).

Elasticity
Refers to the ease of a system's ability to change or adapt. Examples include automatically scaling out or in, updating firewall rules, and remapping IP addresses.

Cost-Efficient
Refers to making the trade-offs required to make a system as inexpensive as possible while meeting all functional requirements.

Secure
Refers to following proper security guidelines and practices to secure a system at every layer.

Appendix

What Is a Solutions Architect? | Terminology | About the Exam | Helpful Links | Learning Activities | Exit

The Certified Solutions Architect Exam (Released February 2018)

About the Exam

- ***Length***: 130 minutes
- ***Number of Questions***: 65
- ***Format:*** Multiple choice
- ***Cost***: $150 USD
- ***Register for the exam***: http://amzn.to/22V3swr

- The exam must be taken at a designated testing center (located in/near most major cities)
- Results are provided immediately after completing the exam

What is covered on the exam?

How to Prepare for the Exam

- Watch and follow along with all video lessons
- Complete every Live Environment hands-on lab (at least once)
- Pass every section quiz (strive for 100%)
- Pass the final (practice) exam a total of three times
- Memorize the instructor flash card deck (or create and memorize your own)
- Read the provided AWS whitepapers (this should not be overlooked)
- Use the Orion Papers for review and study
- Participate in the Linux Academy Community or start/participate in a study group

Test Day — Exam Tips and Tricks

"Tips and Tricks for Taking an AWS Certification Exam" (blog post): http://bit.ly/2nxE1Su

Exam Content

The exam questions are divided up into five domains.

Each domain corresponds to a pillar of the **Well-Architected Framework.**

Domain	% of Exam	Framework Pillar
Domain 1: Design Resilient Architectures	34%	Reliability
Domain 2: Define Performant Architectures	24%	Performance Efficiency
Domain 3: Specify Secure Applications and Architectures	26%	Security
Domain 4: Design Cost-Optimized Architectures	10%	Cost Optimization
Domain 5: Define Operationally Excellent Architectures	6%	Operational Excellence
TOTAL	100%	

Appendix

Helpful Links

Here are some useful links that will assist you in your journey to learn AWS and become certified.

Linux Academy Resources

- **Linux Academy**: linuxacademy.com
- **Certified Solutions Architect (associate) course page**: http://bit.ly/2nhTt3e
- **Linux Academy Community**: http://bit.ly/2nhUKaI
  - Share your success with the community when you pass the exam
  - Share any feedback, thoughts, and/or tips and tricks
- **Linux Academy blog**: http://bit.ly/2nN6QMU
- **Instructor LinkedIn:** https://bit.ly/2Hp0wWd
  - Add me as a contact
  - Let's endorse each other's AWS skills and grow our network

AWS Resources

- **Amazon Web Services**: aws.amazon.com
- **AWS Documentation**: http://amzn.to/1T9k1Og
- **AWS CLI Reference Guide**: http://amzn.to/1l1HXoW
- **AWS Whitepapers**: http://amzn.to/2nBp2sp
  - Architecting for the Cloud: AWS Best Practices http://bit.ly/2iJG1Ht
  - AWS Well-Architected Framework: http://amzn.to/2hhUCVH
- **AWS certification exams info page**: http://amzn.to/20k8G2u
- **AWS certification exams sign-up page**: http://bit.ly/1Yi4KgL

Appendix

Linux Academy Hands-On Labs

Here is a comprehensive list of all live hands-on labs provided by Linux Academy as part of the Certified Solutions Architect (associate) level course. The purpose of live labs are to provide you with real AWS environments to practice what you have learned in the video lessons and help develop your real-world AWS skills.

Building a VPC from Scratch: http://bit.ly/2oi20Fj
Recap: Provisioning an EC2 Instance: http://bit.ly/2mQytp7
EC2 Backup Solutions with AMIs and Snapshots: http://bit.ly/2o8Tb3p
Accessing an Instance's User Data and Metadata: http://bit.ly/2nB4SyB
Setting up an ELB and Auto Scaling Group: http://bit.ly/2oigdSK
Building a More Secure Application with a Bastion Host and NAT Gateway: http://bit.ly/2nwyo74
Using S3 for Static Web Hosting: http://bit.ly/2nwBEPE
Configuring Backup and Archiving Solutions in S3: http://bit.ly/2nBbD3j
Configuring Route 53 DNS Records Sets: http://bit.ly/2nwva3i
Configuring a CloudFront Distribution: http://bit.ly/2oi3bod
VPC Peering: http://bit.ly/2nhDMco
RDS Lab: http://bit.ly/2nhMbwJ
Create and Use an SNS Topic with S3 Events: http://bit.ly/2ozEAdG
CloudWatch Sandbox: http://bit.ly/2nAZ3B8
CloudTrail Sandbox: http://bit.ly/2nwEILJ
CloudFormation Lab: http://bit.ly/2mQvQU0
Elastic Beanstalk Lab: http://bit.ly/2nwEOD5