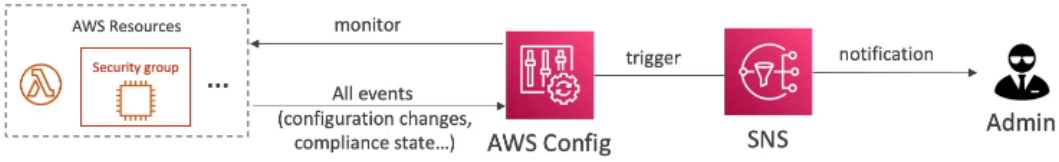


▼ ☑ **AWS Health Dashboard.**

AWS Health Dashboard is a tool that provides personalized views into the status and health of our AWS services and resources. It helps us monitor and manage the health of our AWS infrastructure by delivering alerts and notifications about service disruptions, scheduled maintenance, and other impactful events.

- **Service Health** is the single place to learn about the availability and operations of AWS services. It offers the possibility to subscribe to an RSS feed to be notified of interruptions to each service.

  Service History - displays the status of AWS services across all regions. It shows the current operational status and any ongoing issues.

- **Your Account (Personal Health Dashboard (PHD))** - offers a personalized view into the health of AWS services that are specifically relevant to our AWS environment. It provides alerts and remediation guidance when AWS experiences events that may impact our resources. Shows how AWS outages directly impact us and our AWS resources.

IAM - Advanced

▼ ☑ **AWS Organizations.**

AWS Organizations is a global service that allows us to manage multiple AWS accounts. The main account is called master account. There is API which automates AWS account creation.

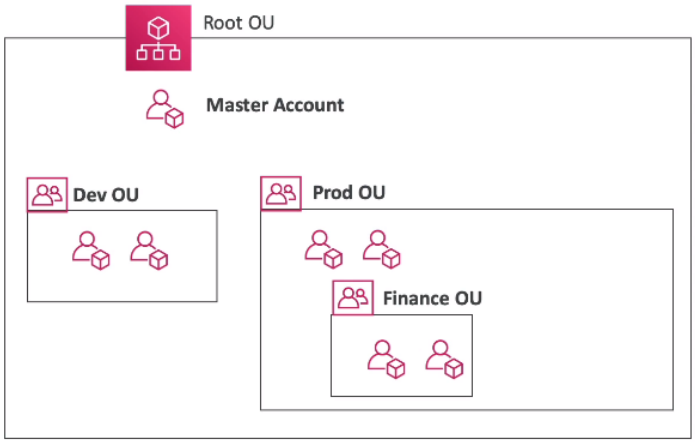
Cost benefits:

- Consolidated Billing across all accounts - we receive a single invoice for all the accounts in our organization, simplifying the billing process.
- Pricing benefits from aggregated usage - higher usage typically results in lower per-unit costs, providing significant savings.
- Pooling of reserved EC2 instances for optimal savings.

Multi Account Strategies

We can create accounts per department, per cost center, per dev/test/prod, for better resource isolation, to have separate per-account service limits, isolated account for logging ...

Good practice is to enable CloudTrail on all accounts and send logs to central S3 account, also send CloudWatch logs to central logging account.

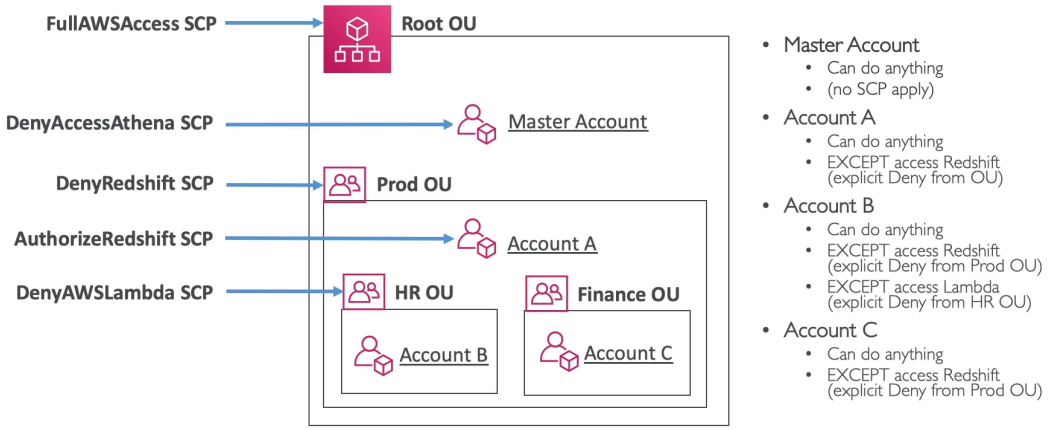


Service Control Policies (SCPs) - allows us to control the services and actions that users and roles can access within the accounts of our organization (whitelist or blacklist IAM actions).

Can be applied to the OU (organizational unit) or account level (does not apply to the master account). It is applied to all the users and roles of the account, including root. SCPs are inherited down the hierarchy. An SCP attached to a root or OU will affect all accounts and OUs beneath it.

It will not affect service-linked roles (service-linked roles enable other AWS services to integrate with organizations and cant be restricted).

SCP must have an explicit allow (does not allow anything by defult).



▼ ☑ **IAM Advanced Policies.**

aws:SourceIp - restrict the client IP from which the API calls are being made.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Deny",
        "Action": "*",
        "Resource": "*",
        "Condition": {
            "NotIpAddress": {
                "aws:SourceIp": [
                    "192.0.2.0/24",
                    "203.0.113.0/24"
                ]
            }
        }
    }
}
```

aws:RequestedRegion - restrict the region the API calls are made to.

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "DenyAllOutsideRequestedRegions",
            "Effect": "Deny",
            "NotAction": [
                "cloudfront:*",
                "iam:*",
                "route53:*",
                "support:*"
            ],
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "aws:RequestedRegion": [
                        "eu-central-1",
                        "eu-west-1",
                        "eu-west-2",
                        "eu-west-3"
                    ]
                }
            }
        }
    ]
}
```

ec2:ResourceTag - restrict based on tags.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:startInstances",
                "ec2:StopInstances"
            ],
            "Resource": "arn:aws:ec2:us-east-1:12345:instance/
            "Condition": {
```

```
                "StringEquals": {
                    "ec2:ResourceTag/Project": "DataAnalytics"
                    "aws:PrincipalTag/Department": "Data"
                }
            }
        }
    ]
}
```

aws:MultiFactorAuthPresent - force MFA.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "ec2:StopInstances",
      "ec2:TerminateInstances"
    ],
    "Resource": ["*"],
    "Condition": {
        "Bool": {
            "aws:MultiFactorAuthPresent": "true"
          }
         }
  }]
}
```

IAM for S3

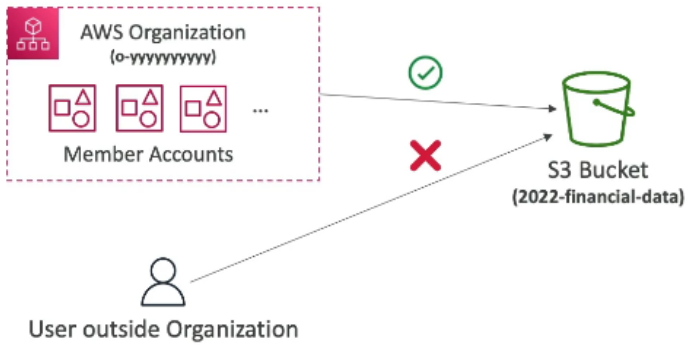
s3:ListBucket - applies to arn:aws:s3:::test (**bucket level permission**).

s3:GetObject, **s3:PutObject**, **s3:DeleteObject** - applies to arn:aws:s3:::test/* (**object level permission**).

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "AllowAuroraToExampleBucket",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:AbortMultipartUpload",
                "s3:ListBucket",
                "s3:DeleteObject",
                "s3:GetObjectVersion",
                "s3:ListMultipartUploadParts"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/*",
                "arn:aws:s3:::amzn-s3-demo-bucket"
            ]
        }
    ]
}
```

aws:PrincipalOrgID - can be used in any resource policies to restrict access to accounts that are member of an AWS Organization.




```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
        "Effect": "Allow",
        "Action": [
            "s3:PutObject",
            "s3:GetObject"
        ],
        "Resource": "arn:aws:s3:::2022-financial-data/*",
        "Condition": {
          "StringEquals": {
            "aws:PrincipalOrgID": "o-sabhong3hu"
          }
        }
      }
    ]
}
```

▼ ☑ **IAM Resource-based Policies vs IAM Roles.**

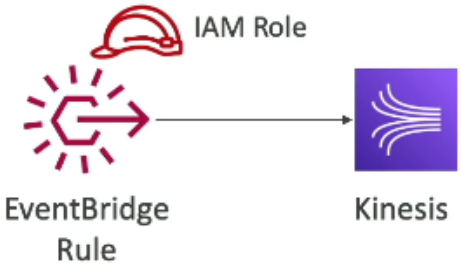
When we want to perform an API call on a S3 bucket cross account we have two options. Attaching a resource-based policy to a resource (e.g. S3 bucket policy) or using a role as a proxy.

- When we assume a role (user, application or service), we give up our original permissions and take the permissions assigned to the role.
- When using a resource-based policy, the principal doesnt have to give up his permissions.

> User in account A needs to scan a DynamoDB table in account A and dump it in an S3 bucket in account B. ⇒ we should use resource-based policy.

It is really important when we use EventBridge. When rule runs, it needs permissions on the target.

We have targets that support resource-based policy (**AWS Lambda, SNS, SQS, S3 buckets, API Gateway**) and targets that supports IAM role (**Kinesis stream, SSM Run Command, ECS task**).



▼ ☑ **IAM Policy Evaluation Logic.**

IAM Permission Boundaries - advanced feature to use a managed policy to set the maximum permissions an IAM entity can get. They are supported for users and roles (not groups). They can be used in combinations of AWS Organizations SCP.

Example:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:*",
                "cloudwatch:*",
                "ec2:*"
            ],
            "Resource": "*"
        }
    ]
}
```

IAM Permission Boundary

+

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:CreateUser",
    "Resource": "*"
  }
}
```

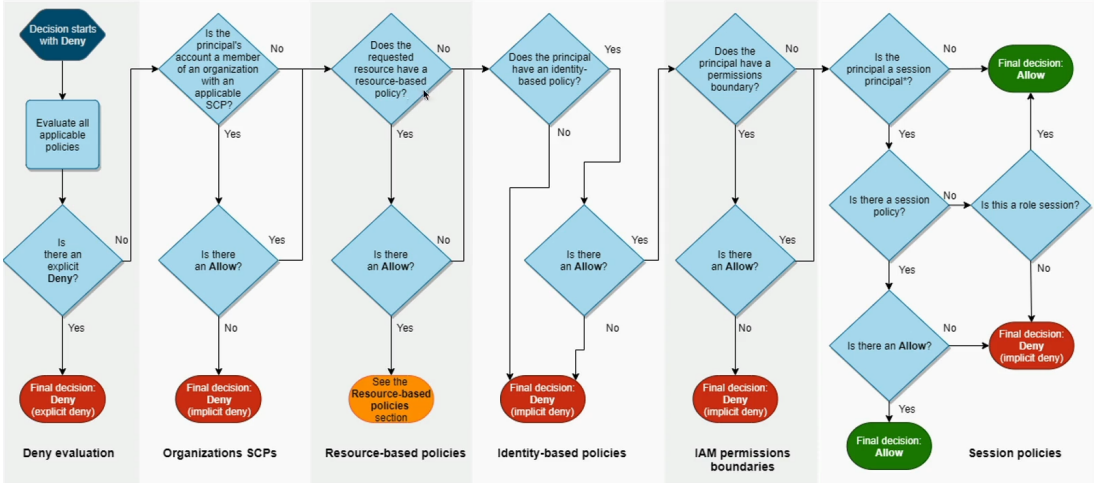
IAM Permissions
Through IAM Policy

= No Permissions

- Can delegate responsibilites to non administrators within their permissions boundaries.
- Allow developers to self-assign policies and manage their own permissions, while making sure they cant escalate their privileges.

- Useful to restrict one specific user (instead of a whole account using Organization & SCP).

IAM Policy Evaluation Logic



- Can you perform sqs:CreateQueue? NO
- Can you perform sqs:DeleteQueue? NO
- Can you perform ec2:DescribeInstances? NO

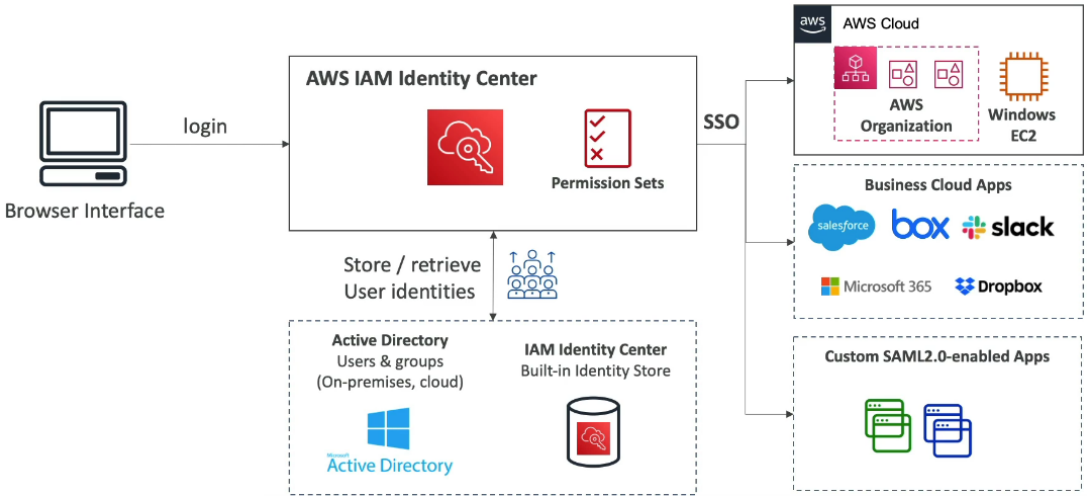
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "sqs:*",
            "Effect": "Deny",
            "Resource": "*"
        },
        {
            "Action": [
                "sqs:DeleteQueue"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

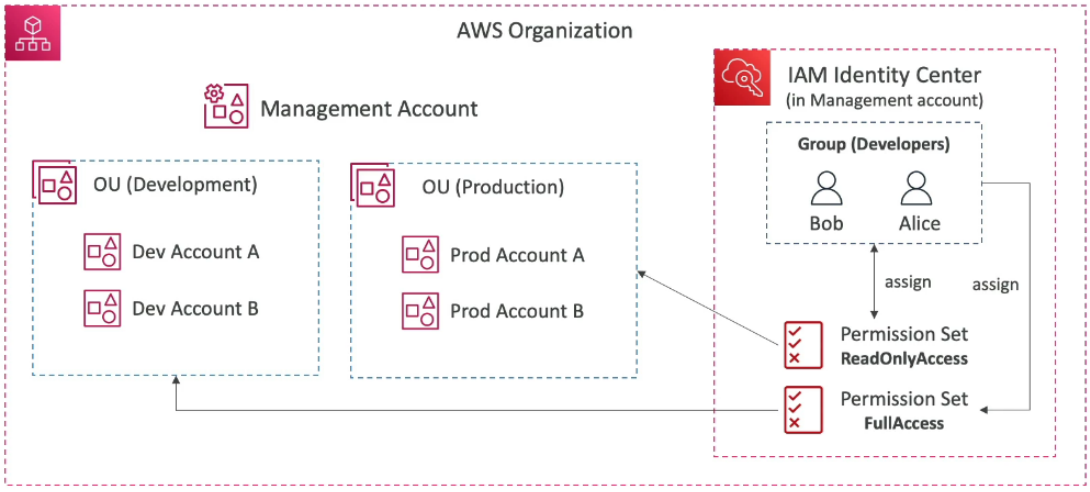
Explicit Deny

▼ ☐ AWS IAM Identity Center.

AWS IAM Identity Center (AWS Single Sign-On (AWS SSO)) is a cloud-based service that simplifies managing access to multiple AWS accounts in AWS Organizations and business applications.

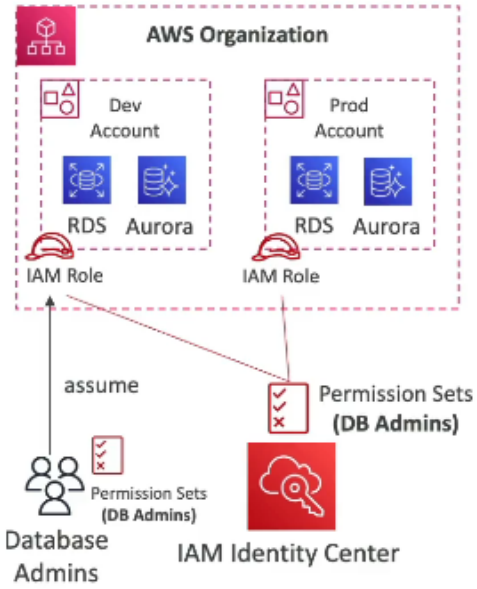
An **Identity Provider** is a service that manages and verifies identities of users within an organization. It authenticates users and provides identity tokens that can be used to grant access to applications and services. IdPs often use standard protocols like SAML 2.0, OpenID Connect.





IAM Identity Center Permissions & Assignments

- **Multi-Account Permissions** - used to manage access across AWS accounts in our AWS Organization. We can create permission sets (collection of one or more IAM policies assigned to users and groups to define AWS access).



- **Application Assignments** - used for SSO access to many SAML 2.0 business applications. It provides required URLs, certificates and metadata.
- **Attribute-Based Acces Control (ABAC)** - fine-grained permissions based on users' attributes stored in IAM Identity Center Identity Store. Used to define permissions once, then modify AWS access by changing the attributes.

▼ ☑ **AWS Directory Services.**

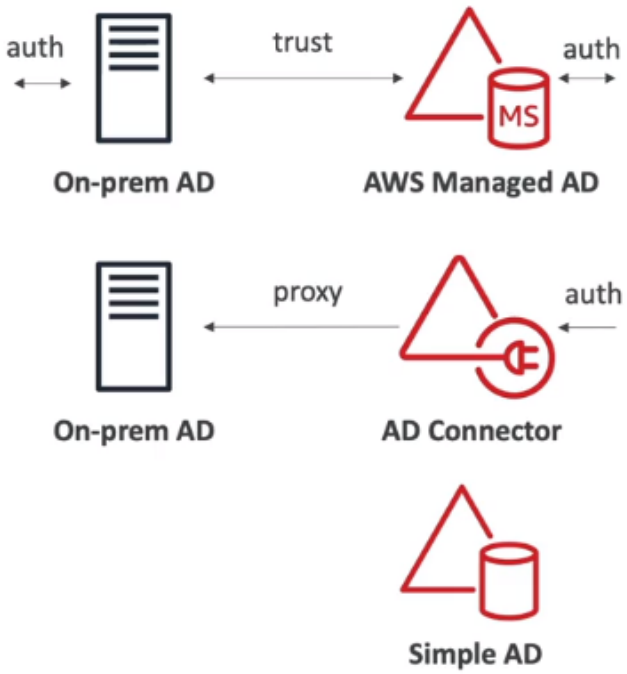
AWS Directory Service is a managed service that makes it easy to set up and run directory services in the AWS cloud or connect your AWS resources with an existing on-premises Microsoft Active Directory (AD).

Microsoft Active Directory (AD) is a directory service developed by Microsoft for Windows domain networks. It is a database of objects: user accounts, computers, printers, file shares, security groups... Objects are organized in trees and a group of trees is a forest.

AWS Directory Services

- **AWS Managed Microsoft AD** - creates our own AD in AWS, manage users locally and supports MFA. It establishers "trust" connections with our on-premise AD. Those directories are deployed across **two AZs in a region** by default and connected to our Amazon VPC.
- **AD Connector** - directory gateway (proxy) to redirect to on-premise AD, supports MFA. Users are managed on the on-premise AD.

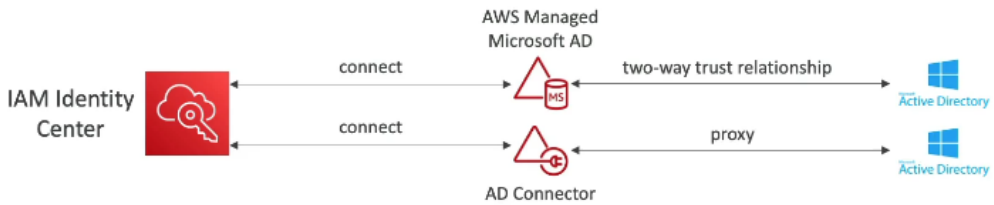
- **Simple AD** - AD-compatible managed directory on AWS. Cannot be joined with on-premise AD.



IAM Identity Center Active Directory Setup

- **Connect to an AWS Managed AD** - integration is out of the box.

- **Connect to a Self-Managed AD** - we need to create two-way trust relationship using AWS Managed Microsoft AD and AD Connector.



▼ ☑ **AWS Control Tower.**

AWS Control Tower is a service that provides a way to set up and govern a secure, multi-account AWS environment based on AWS best practices. It is designed for organizations that need to manage multiple AWS accounts and provides a pre-configured environment called a landing zone. The landing zone includes baseline IAM roles, account structures, and network configurations.

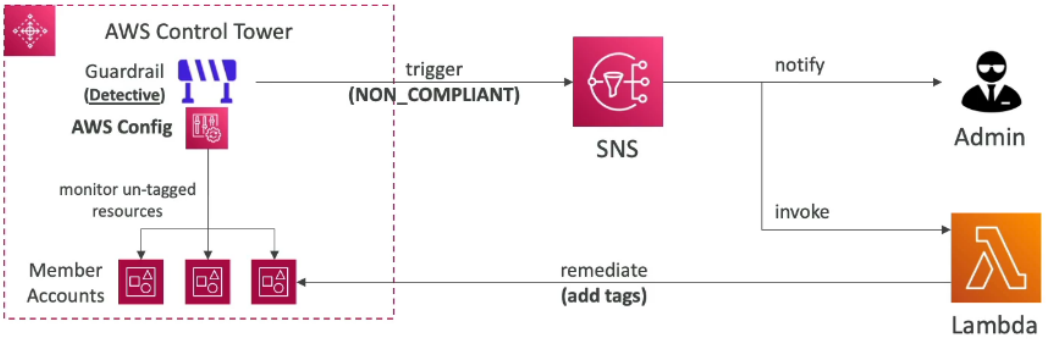
Benefits:

- Automate the set up of our environment.
- Automate ongoing policy management using guardrails.
- Detect policy violations and remediate them.
- Monitor compliance through an interactive dashboard.

Control Tower Guardrails

It provides ongoing governance for our Control Tower evironment (AWS Accounts).

- **Preventive Guardrail** - using SCPs (e.g. **restrict regions across all our accounts**).
- **Detective Guardrail** - using AWS Config (e.g. **identify untagged resources**).



Section IV

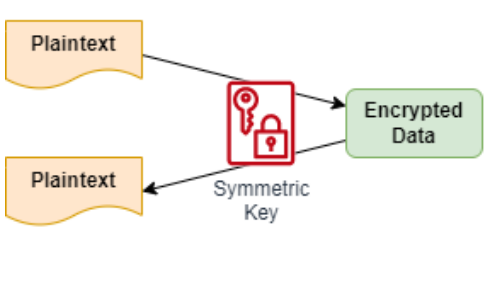
AWS Security & Encryption

▼ ☐ **KMS.**

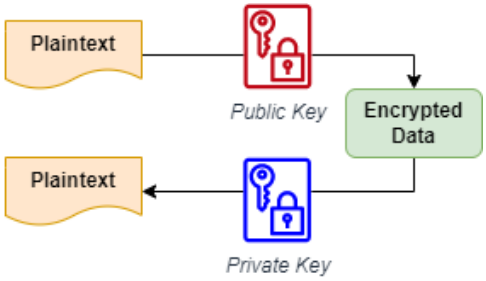
AWS KMS (Key Management Service) is a managed service that allows us to create and manage encryption keys used to encrypt and decrypt our data. It integrates with many other AWS services to simplify the encryption process.

Keys Types

- **Symmetric (AES-256 Keys)** - single encryption key that is used to encrypt and decrypt. AWS services that are integrated with KMS use Symetric CMKs. We never get access to the KMS key unencrypted (must call KMS API to use).
- **Asymmetrict (RSA & ECC Key Pairs)** - public (encrypt) and private (decrypt) key pair. Used for encrypt/decrypt or sign/verify operations. The public key is downloadable, but we cant access the private key unencrypted. It is used for encryption outside of AWS by users who cant call the KMS API.



Symmetric Encryption - the same key is used for both encrypting and decrypting data



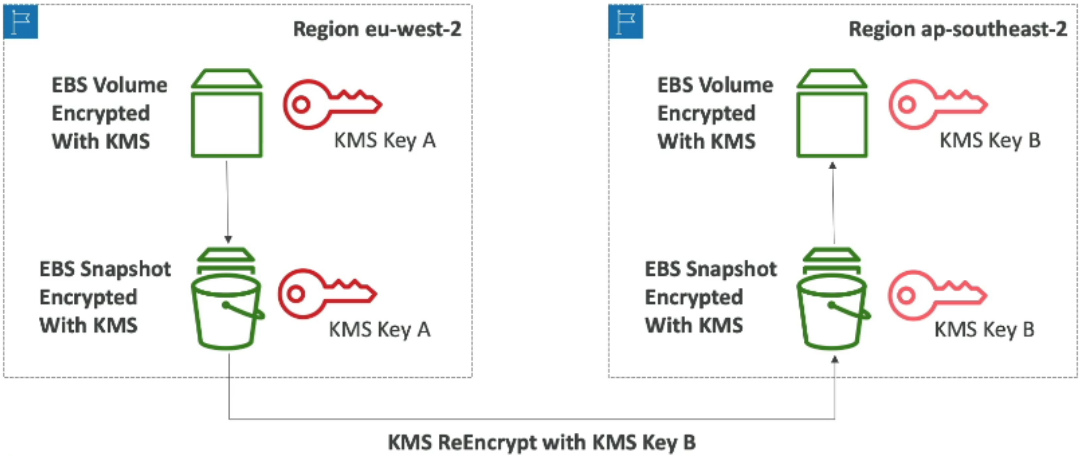
Asymmetric Encryption - the public key is used for encryption and the private key is used for decryption

Types of KMS Keys

- **Customer Managed Key (CMK)** - created, managed and used by the customer. Possibility of rotation policy (new key generated every year). Possibility to bring our own key.
- **AWS Managed Key** - created, managed and used on the customer's behalf by AWS. Used by AWS services.
- **AWS Owned Key** - collection of CMKs that an AWS service owns and manages to use in multiple accounts. AWS can use those to protect resources in our account (we cant view the keys).

Automatic key rotation is automatic every 1 year for AWS managed keys. For created CMKs it must be enabled (can be automatic or on-demand), and for imported KMS keys, only manual rotation is possible using alias.

KMS keys are scoped per region. If we want to to move encrypted EBS volume to another region we need to take snapshot of that EBS volume. Then we need to copy snapshot to another region, we neeed to re-encrypt the snapshot using a different KMS key and then we can restore EBS volume from that snapshot.



KMS Key Policies - control access to KMS keys (similar to S3 bucket policies).

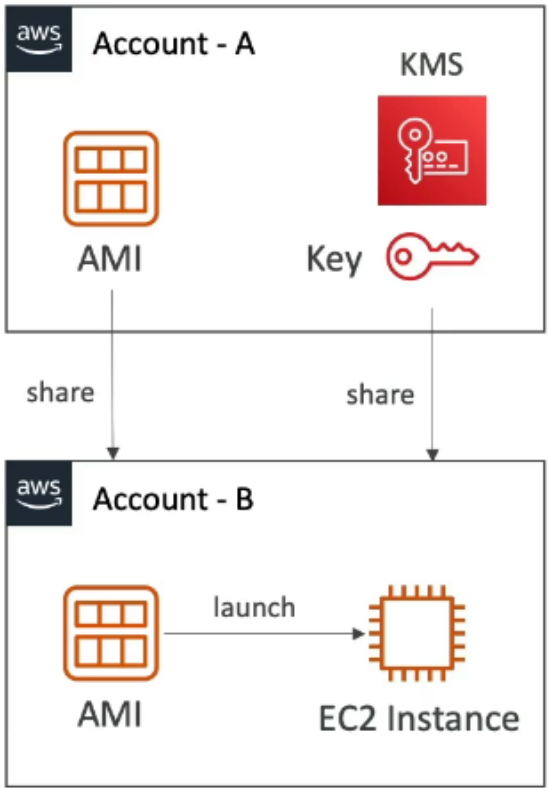
- **Default KMS Key Policy** - created if we dont provide a specific KMS Key Policy. Root user has complete access to the key.
- **Custom KMS Key Policy** - define users, roles that can access the KMS key and who can administer the key. Useful for cross-account access of our KMS key.

To copy snapshot across accounts we need to create a snapshot encrypted with our own KMS key. Then attach a KMS Key Policy to authorize cross-account access and share the encrypted snapshot. In target account we need to create a copy of the snapshot, encrypt it with CMK in our account and then create a volume from snapshot.

▼ **AMI Sharing Process Encrypted via KMS.**

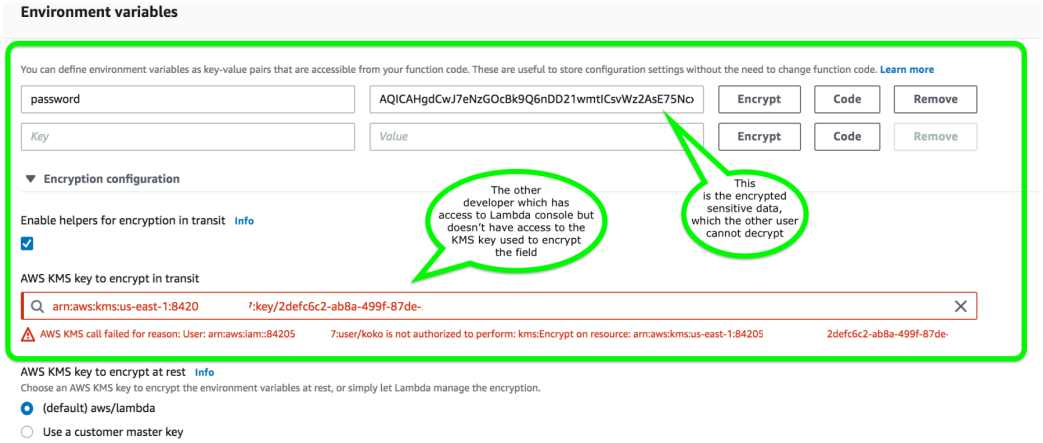
1. AMI in source account is encrypted with KMS key from source account.
2. Must modify the image attribute to add a Launch Permission which corresponds to the specified target AWS account.
3. Share the KMS keys used to encrypt the snapshot AMI references with the target account.

4. The IAM role/user in the target account must have the permissions to DescribeKey, ReEncrypt, CreateGrant, Decrypt.
5. When launching an EC2 instance from AMI, optionally the target account can specify a new KMS key in its own account to re-encrypt the volumes.



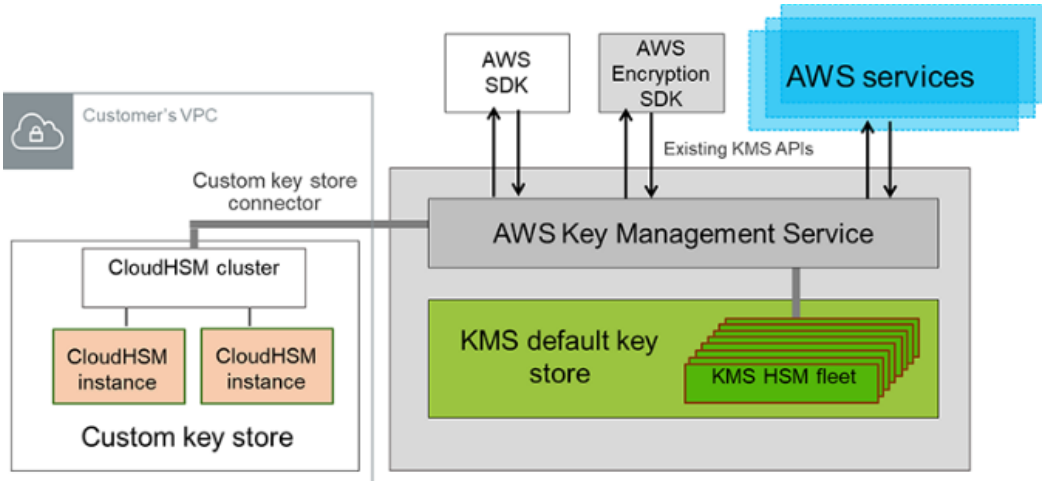
▼ **Encryption of Environment Variables in Lambda.**

Lambda encrypts the environment variables in our function by default, but the sensitive information would still be visible to other users who have access to the Lambda. The best option is to use encryption helpers to secure our environment variables.



▼ KMS - CloudHSM Integration.

The KMS custom key store feature combines the controls provided by AWS CloudHSM. We can configure our own CloudHSM cluster and authorize KMS to use it as a dedicated key store for our keys rather than the default AWS KMS key store. This is suitable if we want to be able to audit the usage of all our keys independently of KMS or CloudTrail.



▼ ☐ KMS Multi-Region Keys.

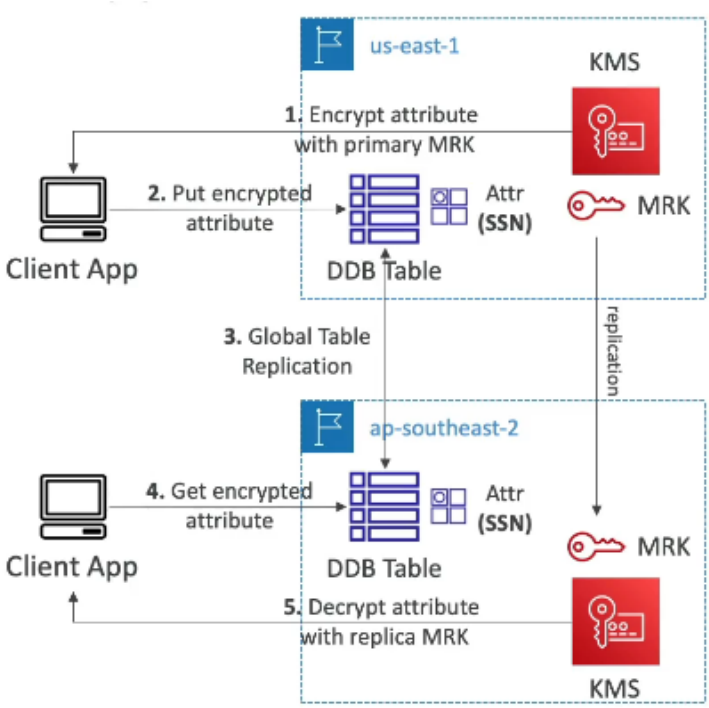
Multi-Region Keys are identical KMS keys in different regions that can be used interchangeably. They have the sam key ID, key material and automatic rotation. We can encrypt in one region and decrypt in other regions (no need to re-encrypt or make cross-region API calls).

KMS Multi-Region is not global (primary + replicas). Each Multi-Region key is managed idependently.

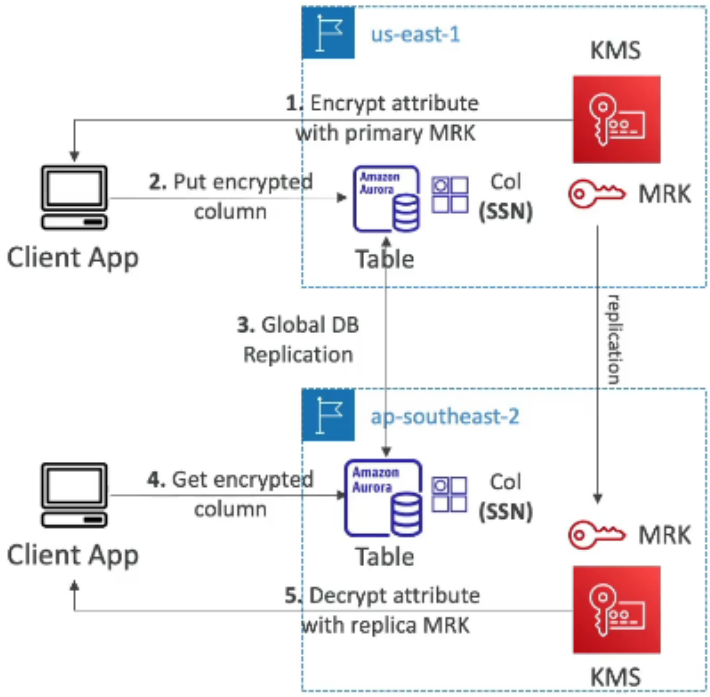
DynamoDB Global Tables/Global Aurora & KMS MRK Client-Side Encryption

We can encrypt specific attributes client-side in our DynamoDB table using DynamoDB Encryption Client. Combined with Global Tables, the client-side encrypted data is replicated to other regions. If we use a MRK replicated in the same region as the DynamoDB Global table, then clients in these regions can use low-latency API calls to KMS in their region to decrypt the data client-side.

Using client-side encryption we can protect specific fields and guarantee only decryption if the client has access to an API key.



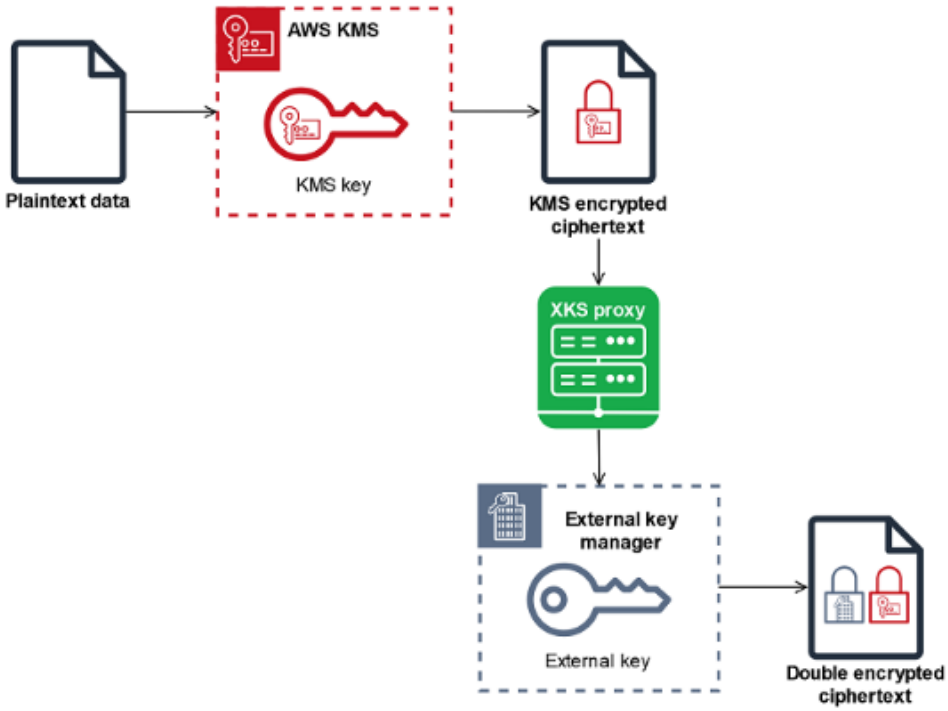
We can encrypt specific attributes client-side in our Aurora table using the AWS Encryption SDK. Combined with Aurora Global Tables, the client-side encrypted data is replicated to other regions. We can protect specific fields even from database admins.



▼ ☐ **KMS XKS.**

External Key Store (XKS) in KMS is a setup where encryption keys are stored and managed outside the primary KMS infrastructure. It enhances security and compliance by allowing organizations to retain control over their encryption keys in a dedicated external system.

XKS Proxy is an intermediary service that acts as a bridge, enabling secure key retrieval and management without requiring direct integration with the external key store itself.



▼ ☑ **S3 Replication with Encryption.**

Unencrypted objects and objects encrypted with SSE-S3 are replicated by default. Object encrypted with SSE-C can also be replicated.

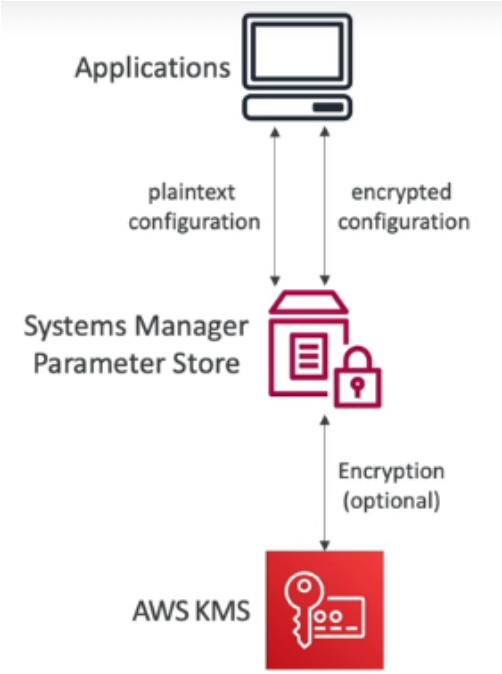
For objects encrypted with SSE-KMS, we need to enable the option:

1. Specify which KMS Key to encrypt the objects within the target bucket.
2. Adapt the KMS Key Policy for the target key.
3. Create IAM Role with kms:Decrypt for the source KMS Key and kms:Encrypt for the target KMS Key.

We can use MRK, but they are currently treated as independent keys by S3 (**the object will still be decrypted and then encrypted**).

▼ ☑ **SSM Parameter Store.**

SSM Parameter Store - it is a secure storage for configuration and secrets (API keys, passwords, configurations ...). It is serverless, scalable and durable. Also it is secure because we control access permissions using IAM.



Parameters Policies - allow us to assign a TTL to a parameter to force updating or deleting sensitive data such as passwords.

Expiration (to delete a parameter)

```
{
    "Type":"Expiration",
    "Version":"1.0",
    "Attributes":{
        "Timestamp":"2020-12-02T21:34:33.000Z"
    }
}
```

ExpirationNotification (EventBridge)

```
{
    "Type":"ExpirationNotification",
    "Version":"1.0",
    "Attributes":{
        "Before":"15",
        "Unit":"Days"
    }
}
```

NoChangeNotification (EventBridge)

```
{
    "Type":"NoChangeNotification",
    "Version":"1.0",
    "Attributes":{
        "After":"20",
        "Unit":"Days"
    }
}
```

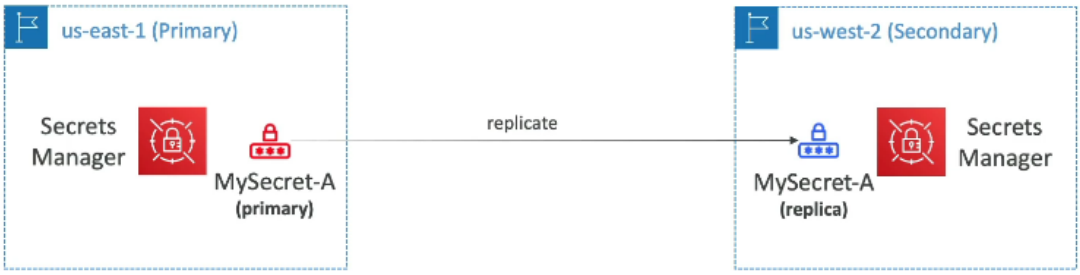
▼ ☑ AWS Secrets Manager.

ASM is a newer service which is used for storing secrets. It has capability to force rotation of secrets every X days. We can automate generation of secrets on rotation using Lambda. All secrets are encrypted using KMS.

Can be integrated with Amazon RDS (MySQL, Postgres, Aurora). Mostly meant for RDS integration.

We can replicate secrets across multiple regions. Secrets Manager keeps read replicas in sync with the primary secret. There is an ability to promote a read replica secret to a standalone secret.

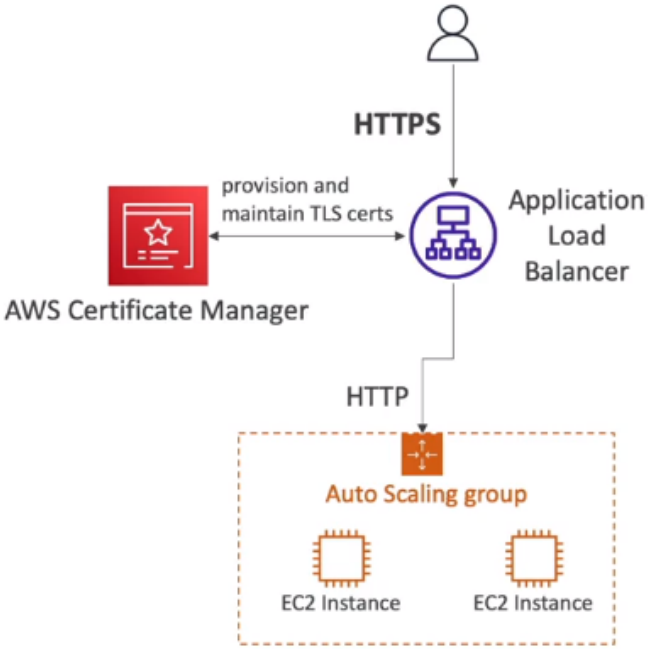
Use cases: multi-region apps, disaster recovery strategies, multi-region databases...



▼ ☑ **AWS ACM & IAM Certificate Store.**

ACM lets us easily provision, manage and deploy TLS certificates. Used to provide HTTPS for websites. It supports both public and private TLS certificates.

Can be integrated with (load TLS certificates on) ELB, CloudFront distributions, APIs on API Gateway.



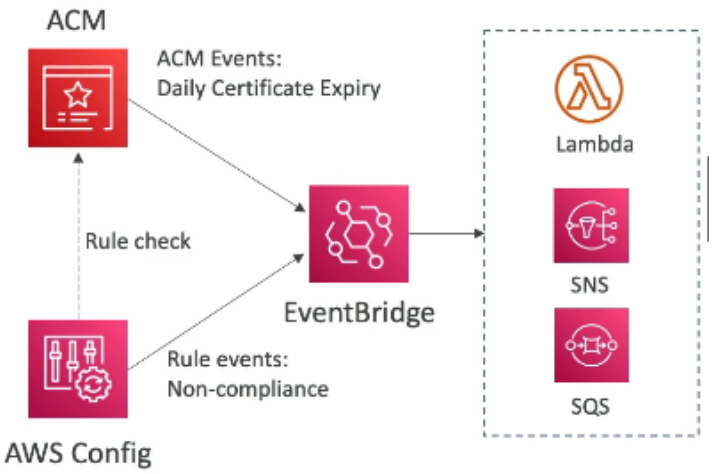
ACM Private CA - for enterprise customers building a public key infrastructure (PKI) inside the AWS cloud and is intended for private use within an organization. We can create our own CA hierarchy and issue certificates with it for

authenticating internal users, computers, applications, services... Certificates issued by a private CA are trusted only within our organization, not on the internet.

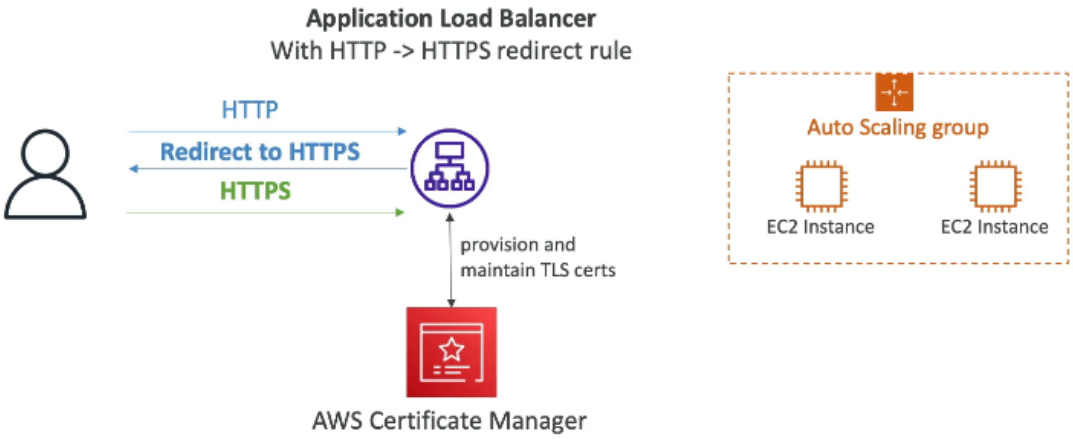
Importing Public Certificates

There is an option to generate the certificate outside of ACM and then import it. Because there is **no automatic renewal**, we must import a new certificate before expiry. ACM sends daily expiration events, starting 45 days prior to expiration (number of days can be configured). Those events are appearing in EventBridge.

AWS Config has a managed rule named *acm-certificate-expiration-check* to check for expired certificates.



Integration with ALB

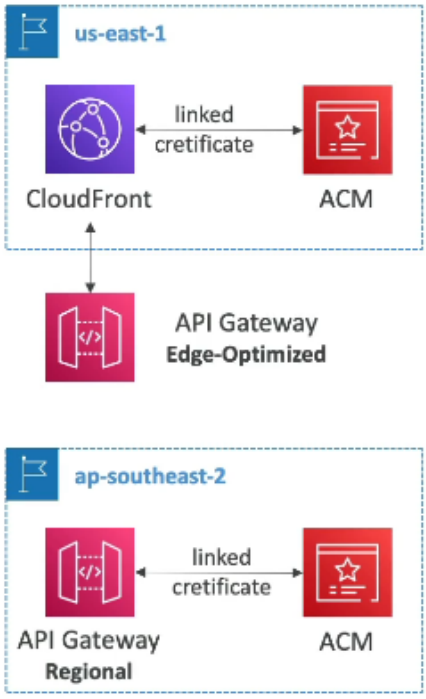


Integration with API Gateway

First, we need to create a custom domain name in API Gateway. ACM can be integrated with Edge-Optimized and Regional endpoints in API Gateway.

- Edge-Optimized Endpoint (default) - TLS Certificate must be in the same region as CloudFront.
- Regional Endpoint - TLS Certificate must be imported on API Gateway in the same region as the API Stage.

Then we need to setup CNAME or Alias record in Route 53.



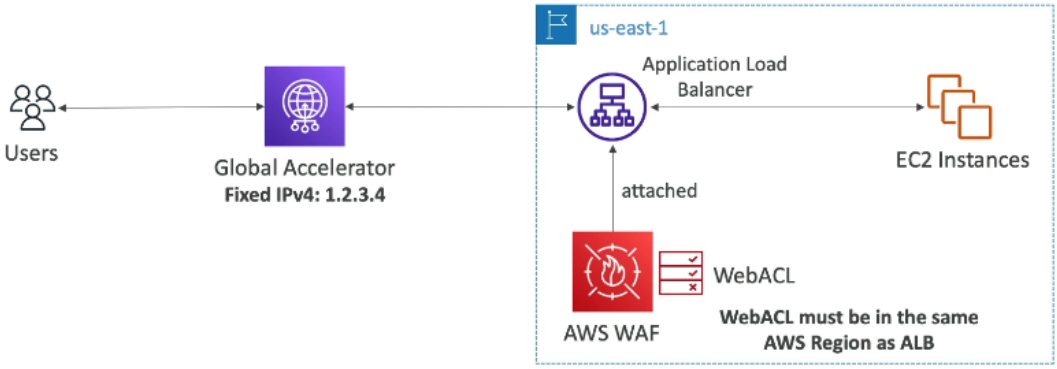
IAM Certificate Store is a system or service used to manage and organize digital certificates within an IAM framework.

▼ ☑ **AWS WAF.**

AWS WAF (Web Application Firewall) - protects our web applications from common web exploits (**Layer 7**). It can be deployed on **ALB, API Gateway, CloudFront, AppSync** and **Cognito User Pool**.

On WAF we can define **Web ACL** (Web Access Control List). Rules can include IP addresses, HTTP headers, HTTP body or URI strings. It can protect us from SQL Injection and XSS. There are Rate-based rules which are used for DDoS protection. A **rule group** is a reusable set of rules that we can add to a web ACL. Web ACL are regional except for CloudFront.

If we want to get a fixed IP in our application while using WAF with an ALB, we have to use Global Accelerator to get fixed IP for application and then enable WAF on our ALB.

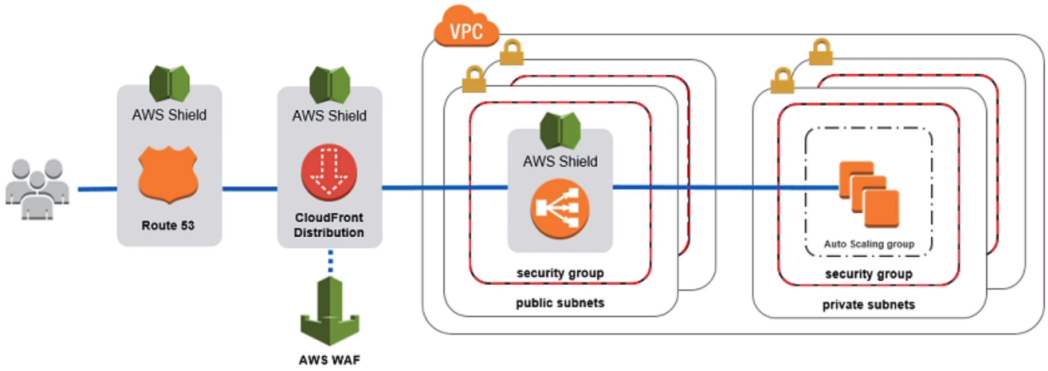


▼ ☑ **AWS Shield.**

DDoS Protection on AWS:

- **AWS Shield Standard** - protects against DDoS attack for our website and applications, its free for all customers.
- **AWS Shield Advanced** - 24/7 premium DDoS protection ($3000 per month).
- **CloudFront & Route 53** - combined with AWS Shield, provides attack mitigation at the edge.

We should be ready to scale during attack - leverage AWS Auto Scaling.



▼ ☑ **Firewall Manager.**

AWS Firewall Manager manages security rules in all accounts of an AWS Organization. Rules are applied to new resources as they are created (good for compliance) across all and future accounts in our organization.

Common set of security rules: VPC Security Groups for EC2, WAF rules, AWS Shield Advanced, AWS Network Firewall ...

▼ ☑ **Amazon GuardDuty.**

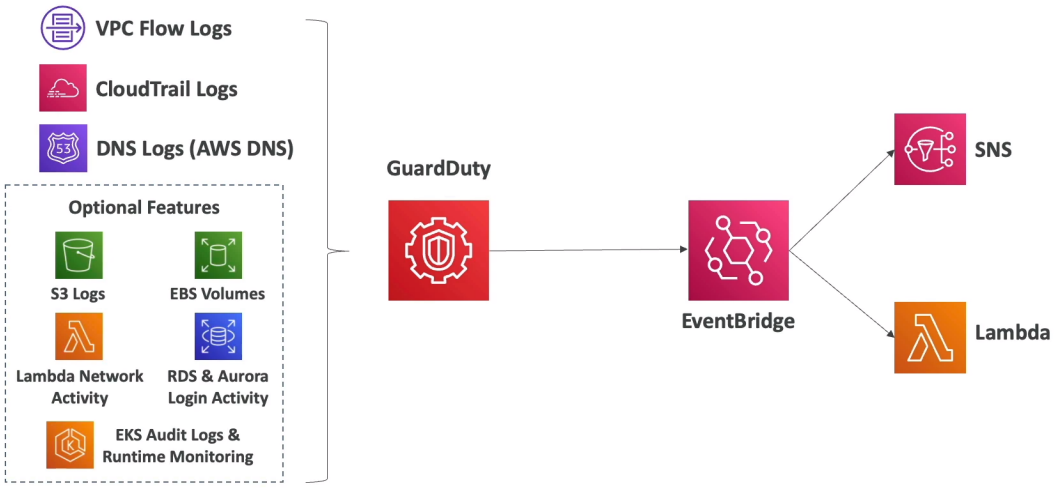
GuardDuty is an intelligent threat discrovery service which protects our AWS account. It uses ML algorithms, anomaly detection and 3rd party data. Focuses on real-time threat detection.

Input data can include:

- **CloudTrail Event Logs** - unusual API calls, unauthorized deployments.
- **VPC Flow Logs** - unusal internal traffic, unusual IP address.
- **DNS Logs** - compromised EC2 instances sending encoded data within DNS queries.
- **Optional Features** - EKS Audit Logs, RDS & Aurora, EBS, Lambda, S3 Data Events...

We can setup EventBridge rules to be notified in case of findings. EventBridge rules can target AWS Lambda or SNS.

GuardDuty can protect us against CryptoCurrency attacks.



▼ ☑ **Amazon Inspector.**

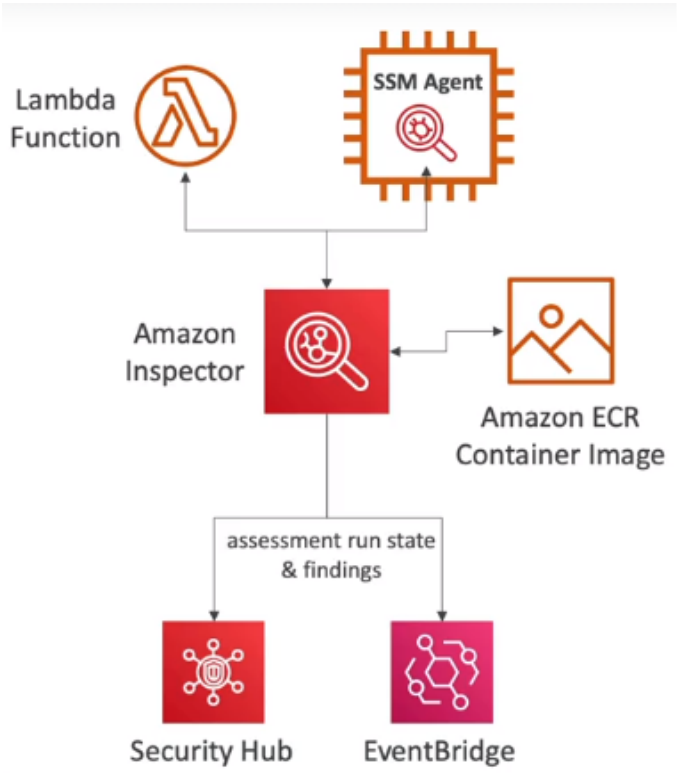
Amazon Inspector is a security assessment service that helps improve the security and compliance of applications deployed on AWS. Inspector

automatically assesses applications for vulnerabilities or deviations from best practices (CVE).

Can be used for:

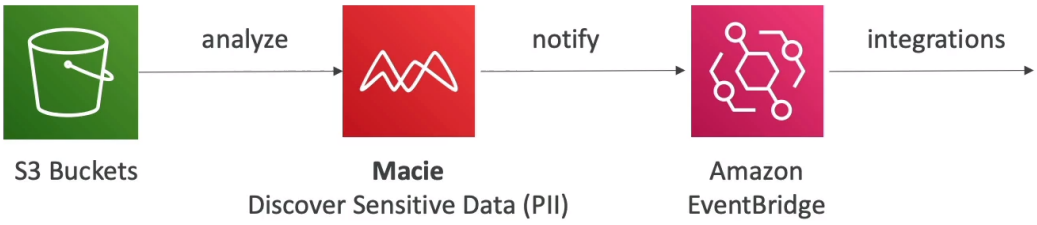
- EC2 Instances
  - Leveraging the SSM agent.
  - Analyze against unintended network accessibility.
  - Analyze the running OS against known vulerabilities.
- Container images push to Amazon ECR
  - Assessment of container images as they are pushed.
- Lambda functions
  - Identifies software vulnerabilities in function code and package dependencies.
  - Assessment of funtions as they are deployed.

Can be integrated with AWS Security Hub and send findings to EventBridge.



▼ ☑ **Amazon Macie.**

Amazon Macie is a fully managed data security and data privacy service that uses ML and pattern matching to discover and protect our sensitive data in AWS. Macie helps idenfity and alert us to sensitive data in our **S3 Buckets**, such as personally identifiable information (PII).



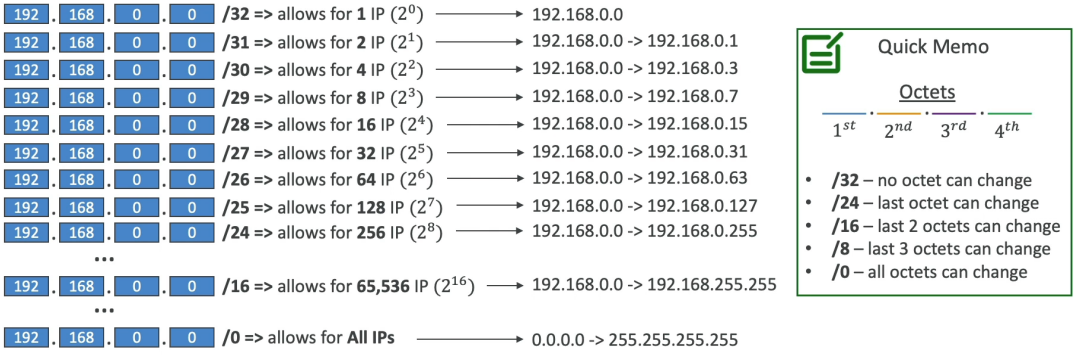
Networking - VPC

▼ ☑ **CIDR, Private & Public IP.**

CIRD (Classless Inter'Domain Routing) - a method for allocationg IP addresses. It is used in Security Groups rules and AWS networking in general. It help to define an IP address range.

CIDR consists of two components:

- **Base IP** - represents an IP contained in the range.
- **Subnet Mask** - defines how many bits can change in the IP.



Private IP addresses are used within a VPC and are not routable over the internet. They are used for internal communication between resources within the same VPC or between different VPCs if they are peered or connected via a VPN.

Public IP addresses are routable over the internet. They are used to allow resources to communicate with external networks and to be accessible from the internet.

▼ ☑ **VPC & Subnet.**

VPC (Virtual Private Cloud) is a private network where we can deploy our resrouces (regional resource). All new AWS accounts have a default VPC. New EC2 instances are launched into the default VPC if no subnet is specified. Default VPC has internet connectivity and all EC2 instances inside it have public IPv4 addresses.

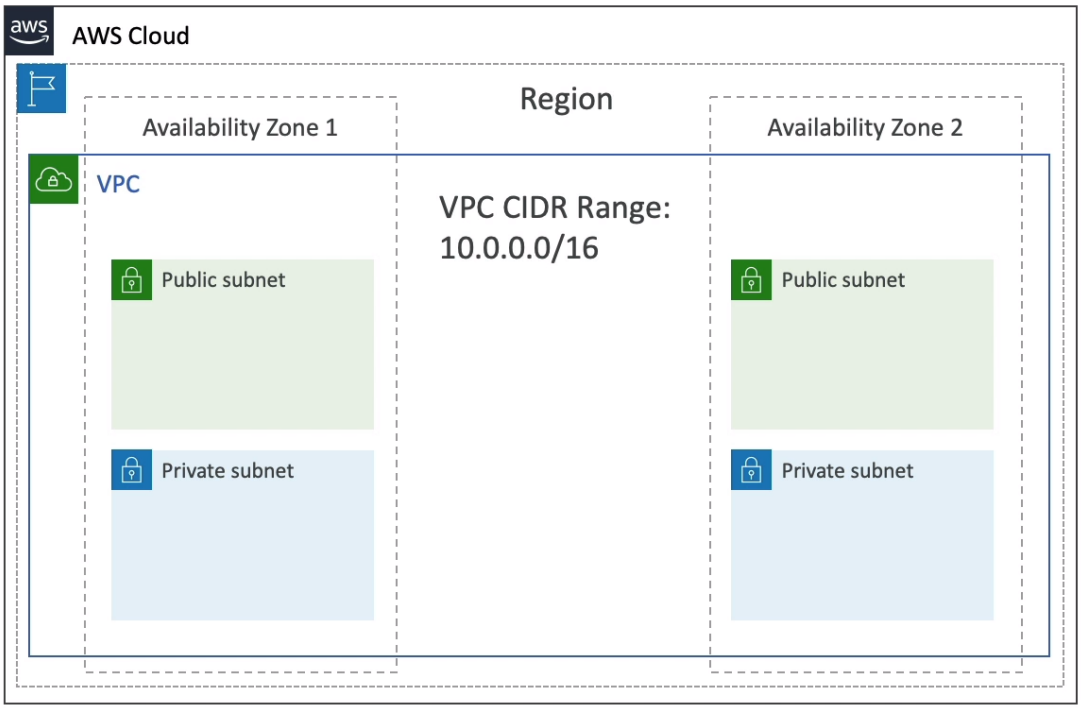
Max CIDR per VPC is five. For each CIDR:

- Min size is /28 (16 IP addresses)
- Max size is /16 (65536 IP addresses)

Our VPC CIDR should not overlap with our other networks (e.g. corporate).

Subnets allow us to partition our network inside our VPC (AZ resource).

- Public subnet - subnet that is accessible from the internet.
- Private subnet - subnet that is not accesible from internet



AWS reserves 5 IP addresses (first 4 and last 1) in each subnet. These IP addresses are not available for use and cant be assigned to an EC2 instance.

Example: If CIDR block is 10.0.0.0/24, then reserved IP addresses are:

10.0.0.0	Network Address
10.0.0.1	For the VPC router
10.0.0.2	For mapping to Amazon-provided DNS
10.0.0.3	For future use
10.0.0.255	Network Broadcast (AWS does not support broadcast in VPC)

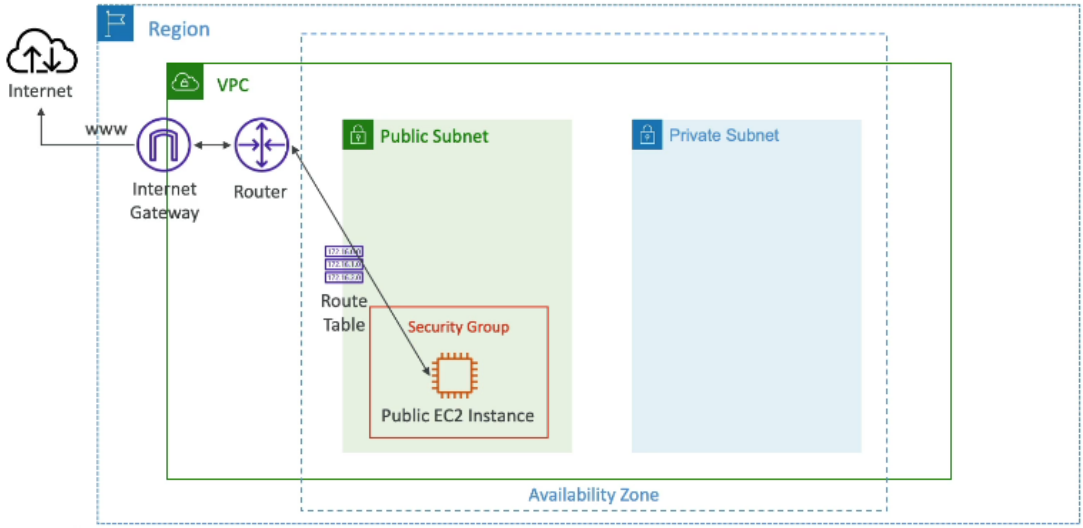
▼ ☑ **Internet Gateways & Route Tables.**

Internet Gateway is a horizontally scaled, redundant, and highly available VPC component that allows instances within our VPC to connect to the internet, and

for the internet to initiate connections with instances within our VPC (if allowed by security rules).

One VPC can only be attached to one IGW and vice versa.

Routing Table is a set of rules (routes) that determine where network traffic is directed. Every subnet in our VPC must be associated with a route table.

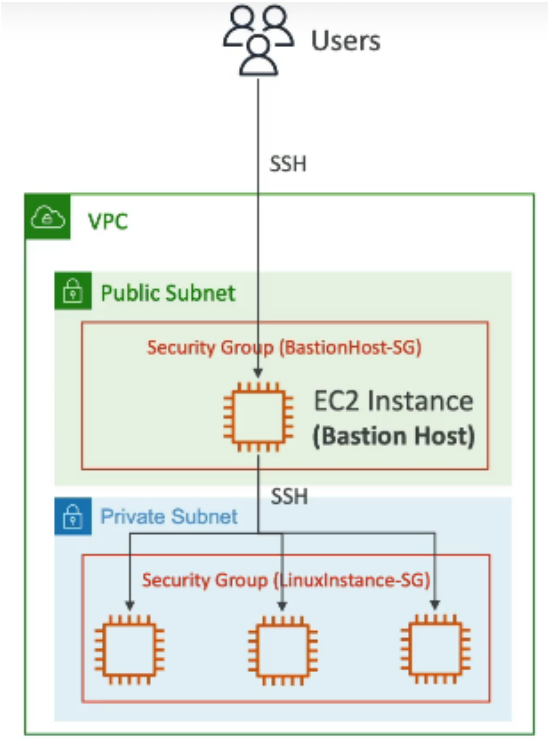


▼ ☑ **Bastion Hosts.**

Bastion Host is a special-purpose instance designed to provide secure access to our private network in the AWS cloud. It acts as a bridge or "**jump box**" for administrative tasks, allowing us to securely manage and administer our instances in private subnets.

Bastion Host security group must allow inbound from the internet on port 22 from restricted CIDR, for example the public CIDR of our corporation.

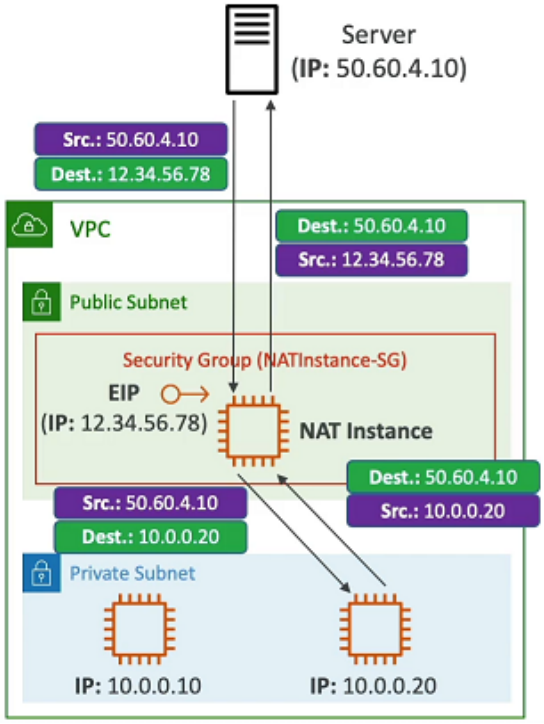
Security Group of the EC2 instances must allow the Security Group of the Bastion Host, or the private IP of the Bastion Host.



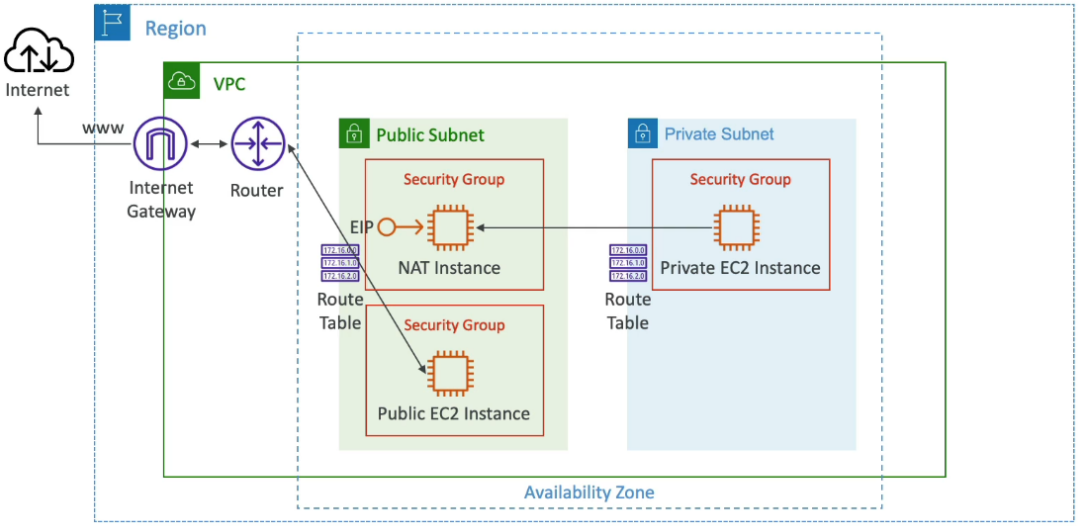
▼ ☑ **NAT Instances & NAT Gateways.**

NAT Instance allows EC2 instance in private subnets to connect to the internet. They must be launched in a public subnet and we must disable EC2 source/destination check setting and have Elastic IP attached to it.

Route Tables must be configured to route traffic from private subnets to the NAT Instance.

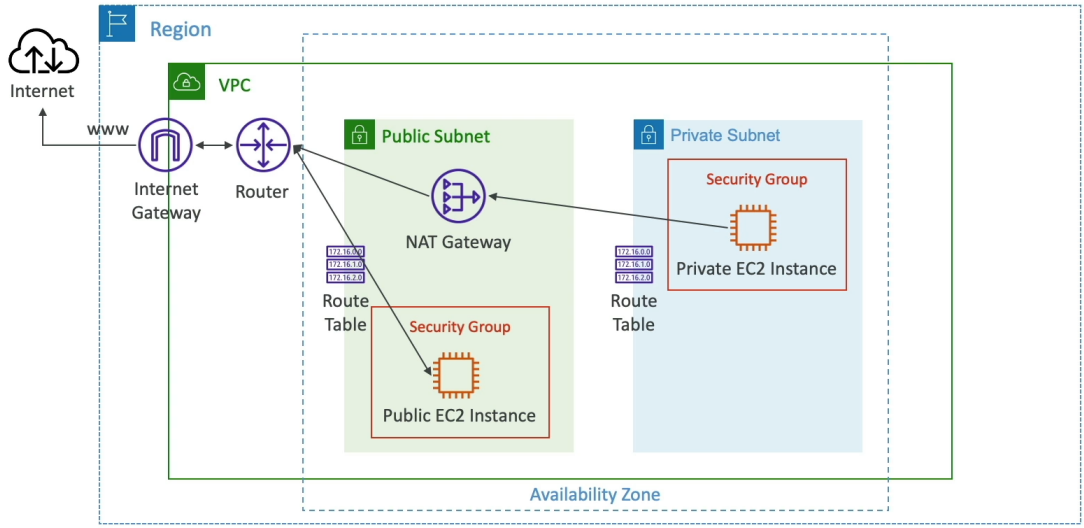


There is a pre-configured Amazon Linux AMI for NAT Instance. NAT instances are not highly available, we need to create an ASG in multi-AZ and internet traffic bandwidth depends on EC2 instance type.

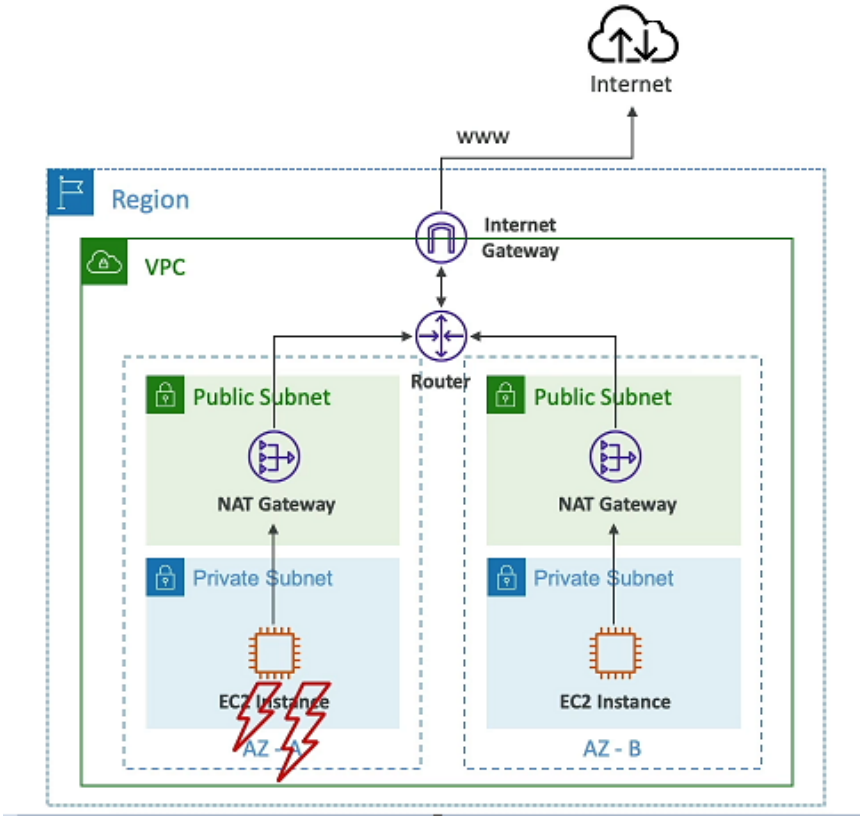


NAT Gateway is AWS managed NAT with higher bandwidth, high availability and no administration. We pay per hour for usage and bandwidth. NAT Gateway is created in a specific AZ and uses an Elastic IP.

It cant be used by EC2 instance in the same subnet (only from other subnets). It requires an Internet Gateway.



NAT Gateway is resilient within a single AZ. We must create multiple NAT Gateways in multiple AZs for fault-tolerance. There is no cross-AZ failover needed because if an AZ goes down it doesnt need NAT.



NAT Instance vs NAT Gateway

	NAT Gateway	NAT Instance
Availability	Highly available within AZ (create in another AZ)	Use a script to manage failover between instances
Bandwidth	Up to 100 Gbps	Depends on EC2 instance type
Maintenance	Managed by AWS	Managed by you (e.g., software, OS patches, ...)
Cost	Per hour & amount of data transferred	Per hour, EC2 instance type and size, + network $
Public IPv4	✔	✔
Private IPv4	✔	✔
Security Groups	✖	✔
Use as Bastion Host?	✖	✔

▼ ☑ **NACL & Security Groups.**

NACL - firewall which controls traffic from and to subnet. It can have ALLOW and DENY rules. Rules only include IP addresses. Newly created NACLs will deny everything. We can have one NACL per subnet, new subnets are assigned the Default NACL.

Default NACL accepts everything inbound/outbound with the subnets it is associated with. We should not modify the Default NACL, instead we should create custom NACLs.

Default NACL for a VPC that supports IPv4

Inbound Rules

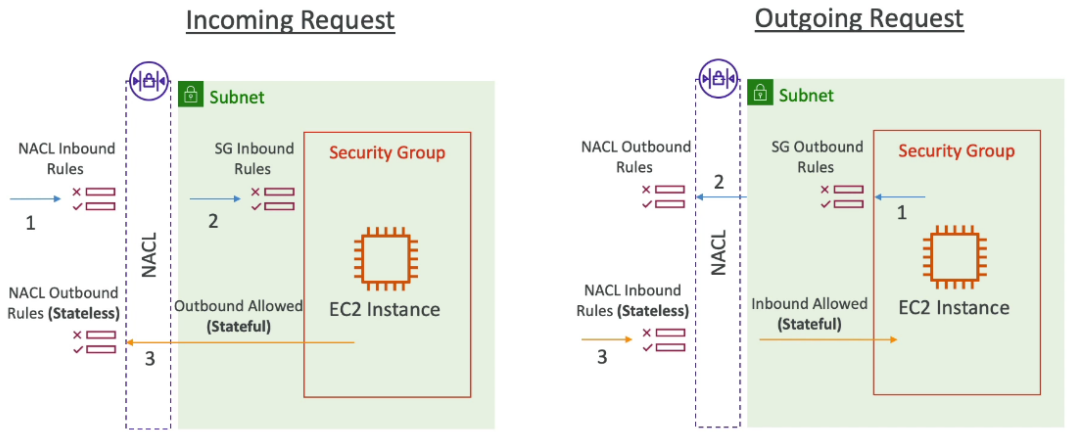
Rule #	Type	Protocol	Port Range	Source	Allow/Deny
100	All IPv4 Traffic	All	All	0.0.0.0/0	ALLOW
*	All IPv4 Traffic	All	All	0.0.0.0/0	DENY

Outbound Rules

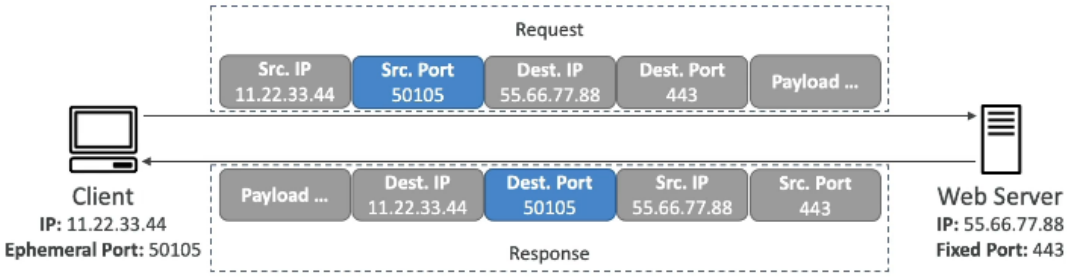
Rule #	Type	Protocol	Port Range	Destination	Allow/Deny
100	All IPv4 Traffic	All	All	0.0.0.0/0	ALLOW
*	All IPv4 Traffic	All	All	0.0.0.0/0	DENY

Rules have a number (1-32766), higher precedence = lower number. First rule match will drive the decision.

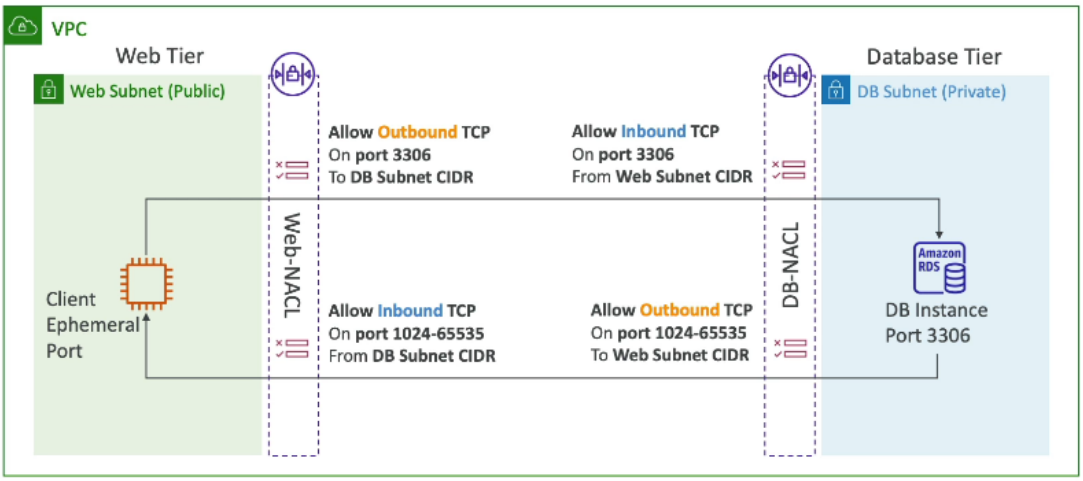
Security Groups - firewall that controls traffic to and from an EC2 instance. It can have only ALLOW rules. Rules include IP addresses and other security groups.



Ephemeral Ports - temporary, short-lived ports used for outbound connections. They are typically assigned dynamically and used for communication between a client and a server.



NACL with Ephemeral Ports



- **Client → Web Server:** The client connects to the web server using an ephemeral port.
- **Web Server → Database:** The web server connects to the RDS instance over port 3306 to retrieve or store data.
- **Response from Database → Web Server:** The RDS instance sends responses back to the web server using an ephemeral port.
- **Response from Web Server → Client:** The web server sends the response back to the client.

NACL rules ensure that the web tier can communicate with the database tier while limiting traffic to only the necessary ports. The use of a private subnet for the database ensures it is not directly accessible from the internet, enhancing security.

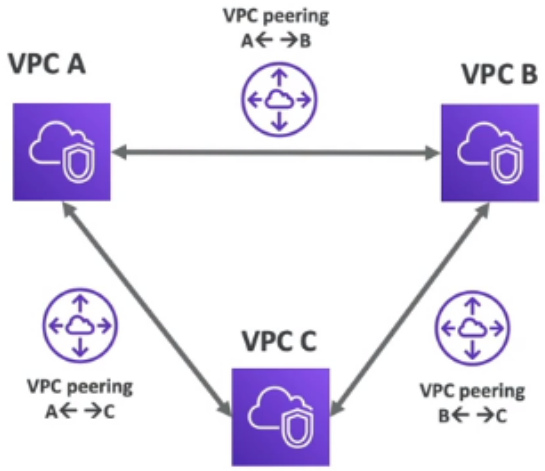
Security Group vs NACL

Security Group	NACL
Operates at the instance level	Operates at the subnet level
Supports allow rules only	Supports allow rules and deny rules
Stateful: return traffic is automatically allowed, regardless of any rules	**Stateless:** return traffic must be explicitly allowed by rules (think of ephemeral ports)
All rules are evaluated before deciding whether to allow traffic	Rules are evaluated in order (lowest to highest) when deciding whether to allow traffic, first match wins
Applies to an EC2 instance when specified by someone	Automatically applies to all EC2 instances in the subnet that it's associated with

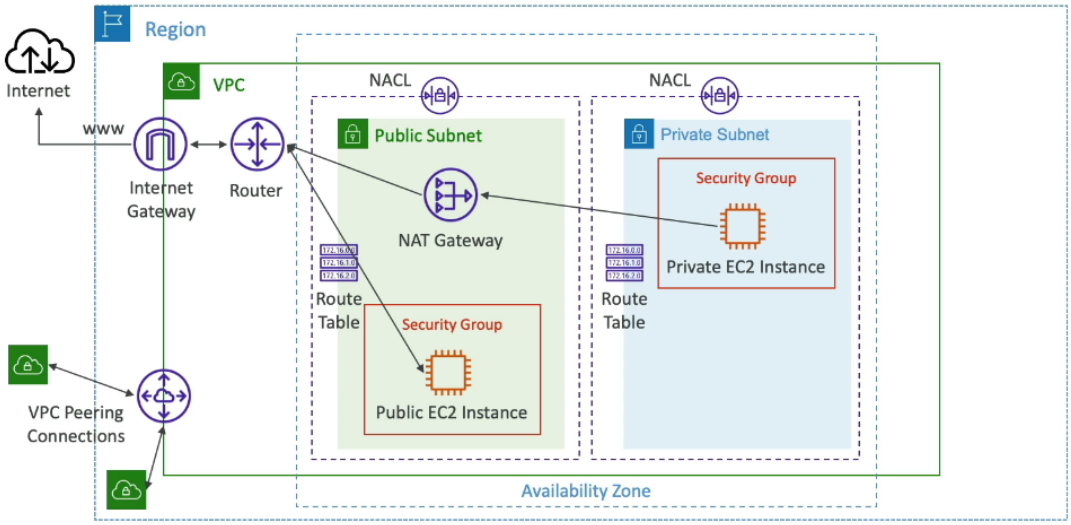
▼ ☑ **VPC Peering.**

VPC Peering connects two VPCs privately using AWS network. It makes two VPCs behave as if they were in the same network. They must not have overlapping CIDR (IP address range).

VPC Peering connection is not transitive (must be established for each VPC that need to communicate with one another).

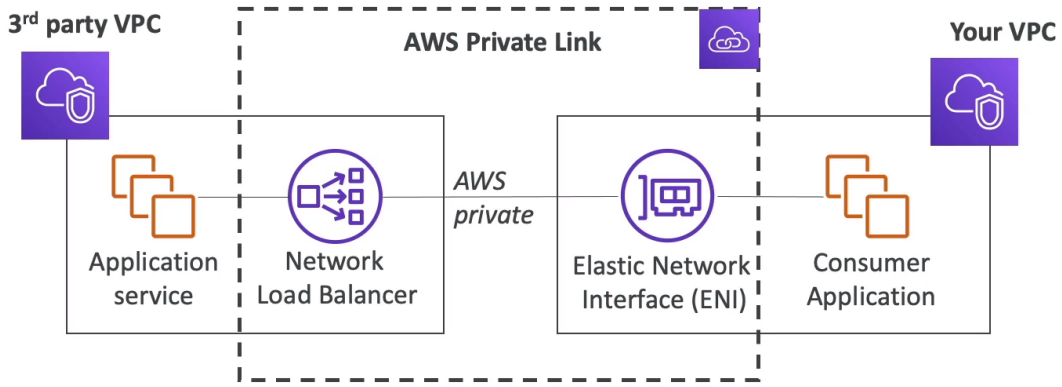


We must update route tables in each VPC's subnets to ensure EC2 instances can communicate with each other. We can create VPC Peering connection between VPCs in different AWS accounts/regions. Also we can reference a security group in a peered VPC (works cross accounts in same region).



▼ ☑ **VPC Endpoints & AWS PrivateLink.**

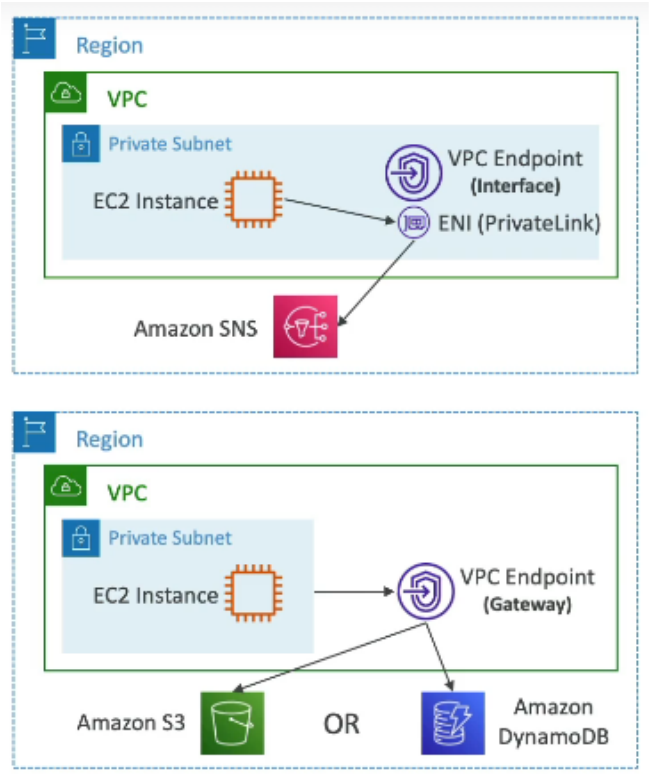
AWS PrivateLink allows us to privately access services hosted on AWS, including our own applications, third-party services, and AWS services, without exposing our traffic to the public internet. It simplifies the network architecture, improves security. Does not require VPC peering, internet gateway, NAT, route tables, but it requires a NLB on service VPC and ENI on customer VPC.

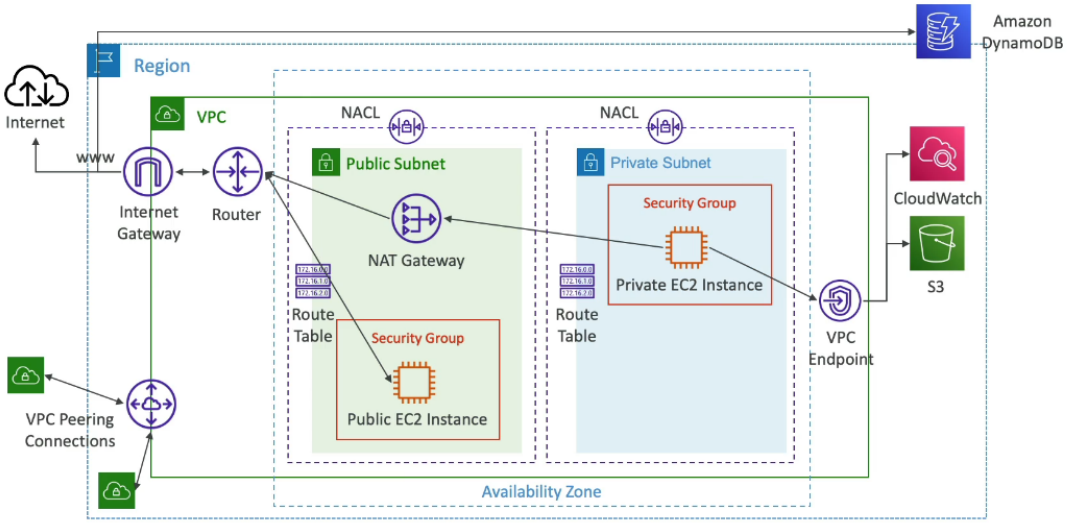


VPC Endpoints (powered by AWS PrivateLink) allow us to connect to AWS Services using a private network instead of the public internet. It gives us enhanced security and lower latency to access AWS services.

- **Interface Endpoints** (powered by PrivateLink) - provisions an ENI (private IP address) as an entry point (must attach a Security Group). It supports most AWS services. We pay per hour and per GB of data processed.
- **Gateway Endpoints** - provisions a gateway and must be used as a target in a route table (does not use security groups). It supports S3 and DynamoDB and it is free.

  We can attach an endpoint policy that controls access to the service to which we are connecting. We can modify the endpoint policy attached to our endpoint and add or remove the route tables used by the endpoint. An endpoint policy does not override or replace IAM user policies or service-specific policies.





▼ ☑ **VPC Flow Logs.**

VPC Flow Logs capture information about IP traffic going into our interfaces.
There are also Subnet Flow Logs and ENI Flow Logs. VPC Flow logs data can go to S3, CloudWatch Logs and Kinesis Data Firehose.

srcaddr & dstaddr - identify problematic IP.

srcport & dstport - identify problematic ports.

action - success or failure of the request due to Security Group/NACL.

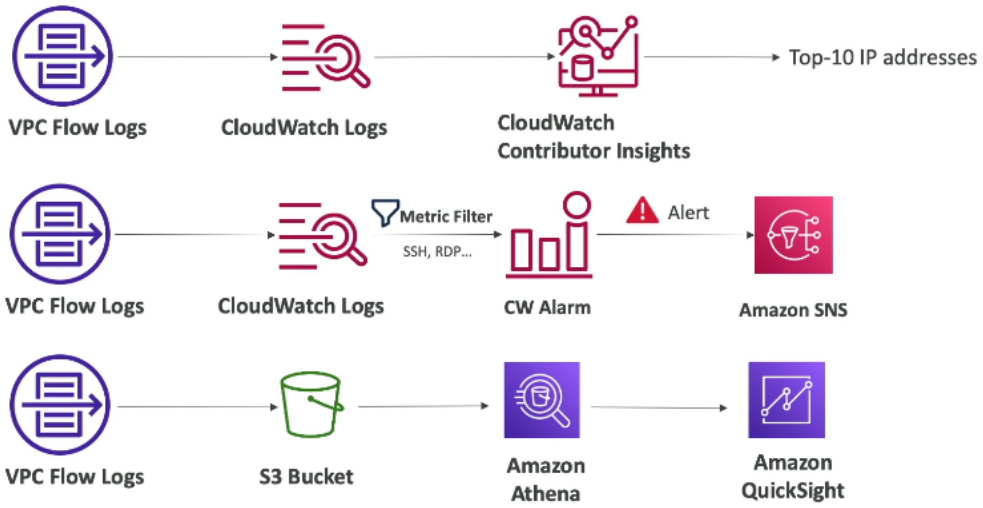
We can query VPC Flow Logs using Athena on S3 or CloudWatch Logs Insights.

Troubleshooting SG & NACL

We can look at the "ACTION" field to troubleshoot problems.

- Inbound REJECT ⇒ NACL or SG problem.
- Inbound ACCEPT, Outbount REJECT ⇒ NACL problem.
- Outbound REJECT ⇒ NACL or SG problem.
- Outbount ACCEPT, Inbound REJECT ⇒ NACL problem.

▼ **VPC Flow Logs Architectures.**



▼ ☐ **Site-to-Site VPN, VGW & CGW.**

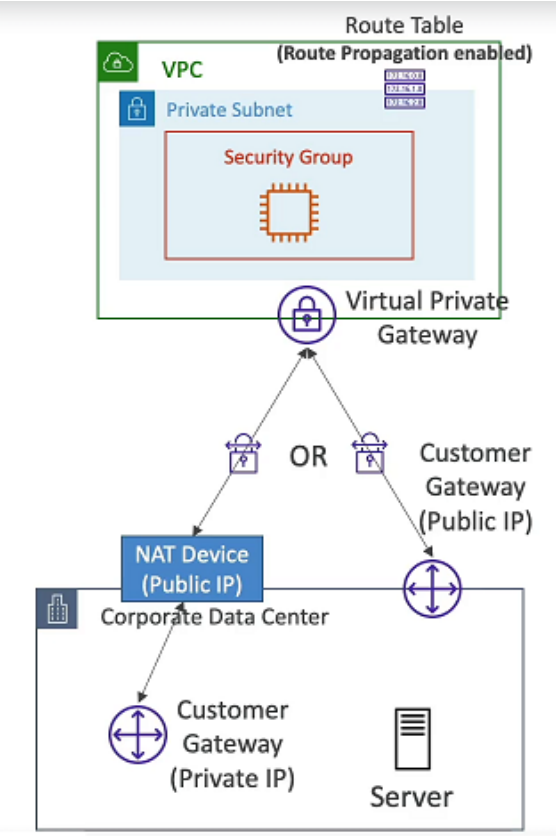
Site-to-Site VPN is used to connect an on-premises VPN to AWS. The connection is automatically encrypted and it goes over the public internet.

To establish Site-to-Site VPN:

- **On-premises** must use a **Customer Gateway (CGW)** - a physical device or software application on the customer's side of the VPN connection. It can be a hardware VPN device, such as a router or firewall, or it can be software-based.
- **AWS** must use a **Virtual Private Gateway (VGW)** - the VPN concentrator on the Amazon side of the VPN connection within the AWS infrastructure. It is a logical entity that represents a VPN gateway capable of handling multiple VPN connections simultaneously.

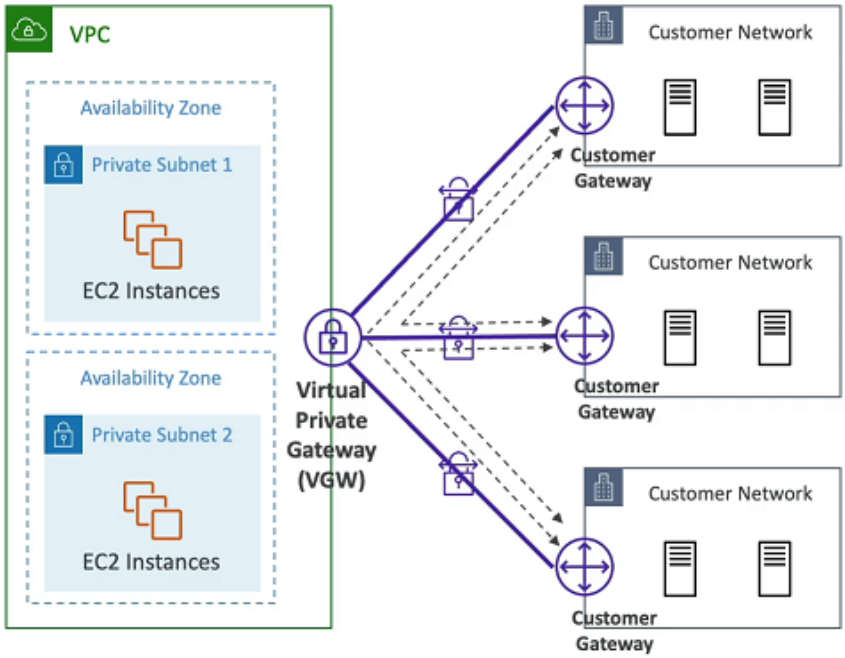
On-premises should use public internet-routable IP address for its CGW device or if it's behind a NAT device that is enabled for NAT traversal (NAT-T), it should use the public IP address of the NAT device.

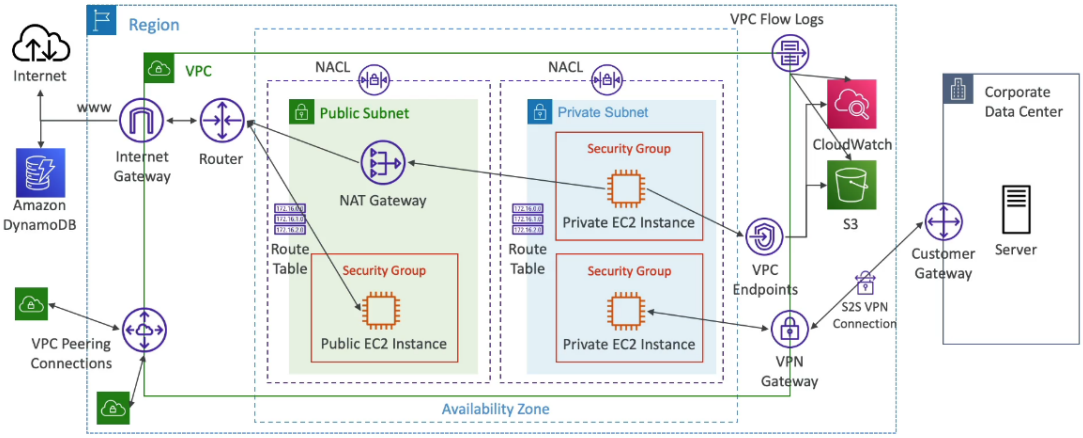
We **must enable Route Propagation for VGW in the route table that is associated with our subnets**.



> If we need to ping our EC2 instances from on-premises, we need to add the ICMP protocol on the inbound of our security groups.

AWS VPN CloudHub - provides secure communication between multiple sites, if we have multiple VPN connections. To set it up, we need to connect multiple VPN connections on the same VGW, setup dynamic routing and configure route tables.





▼ ☐ **DX & DX Gateway.**

Direct Connect provides a dedicated **private** connection from a remote network to our VPC. Dedicated connection must be setup between our data center and AWS DX locations. We need to setup a VGW on our VPC. With DX we can access both public and private resources on same connection.

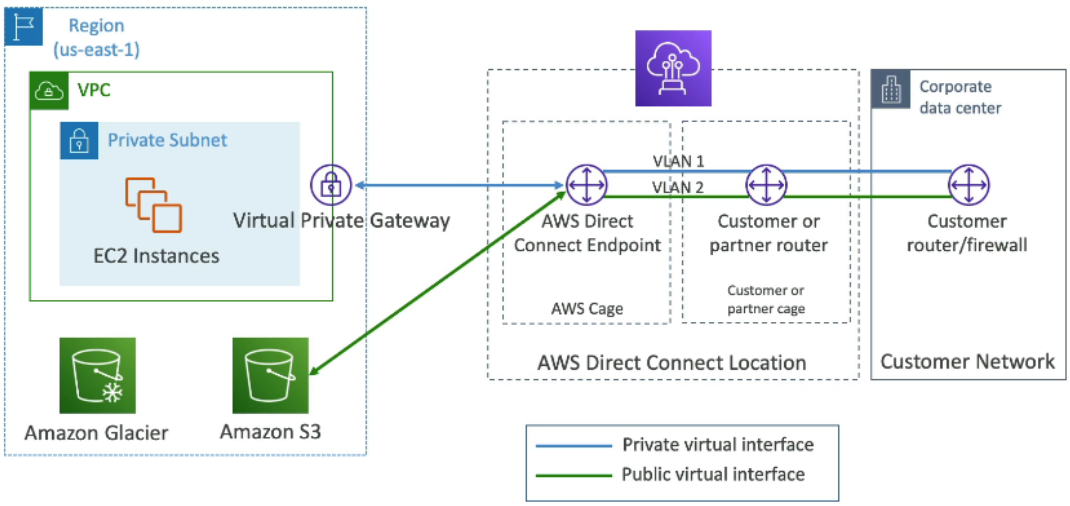
Use cases:

- Increase bandwidth throughput (**working with large data sets**).

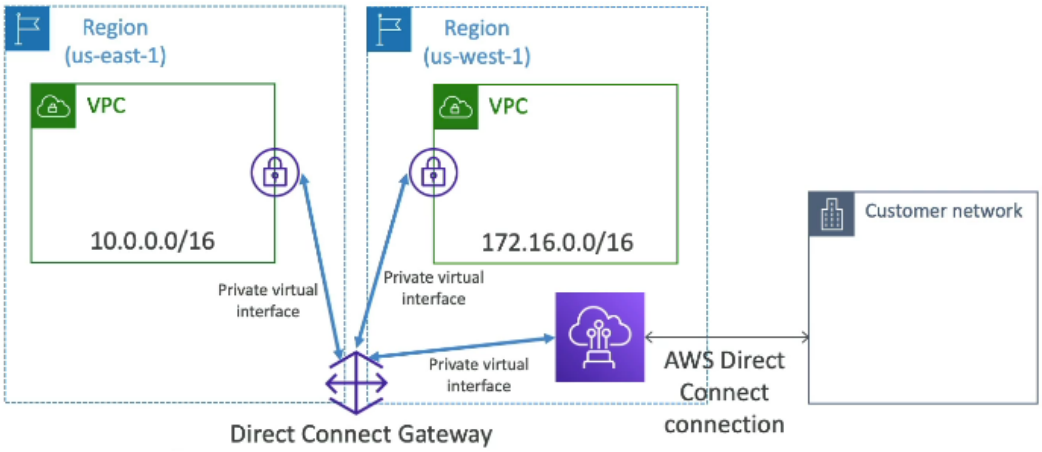
- More consistent network experience (applications using real-time data feeds).
- Hybrid environments.

Direct Connect uses **Virtual Interface** (VIFs) to logically separate different types of traffic. There are three primary types:

- **Private VIF**: Used to connect to our VPC for accessing resources like EC2 instances, RDS databases, etc. Used when we want a private link between our on-premises environment and AWS.
- **Public VIF**: Used to access AWS public services (e.g., S3, DynamoDB, etc.) over a private connection. Public VIFs connect to AWS public endpoints and can be used to route traffic to multiple AWS Regions.
- **Transit VIF**: This allows us to connect multiple VPCs across different AWS Regions using a single connection and a transit gateway.



If we want to setup a DX to one or more VPC in many different regions (same account), we must use a **DX Gateway**.



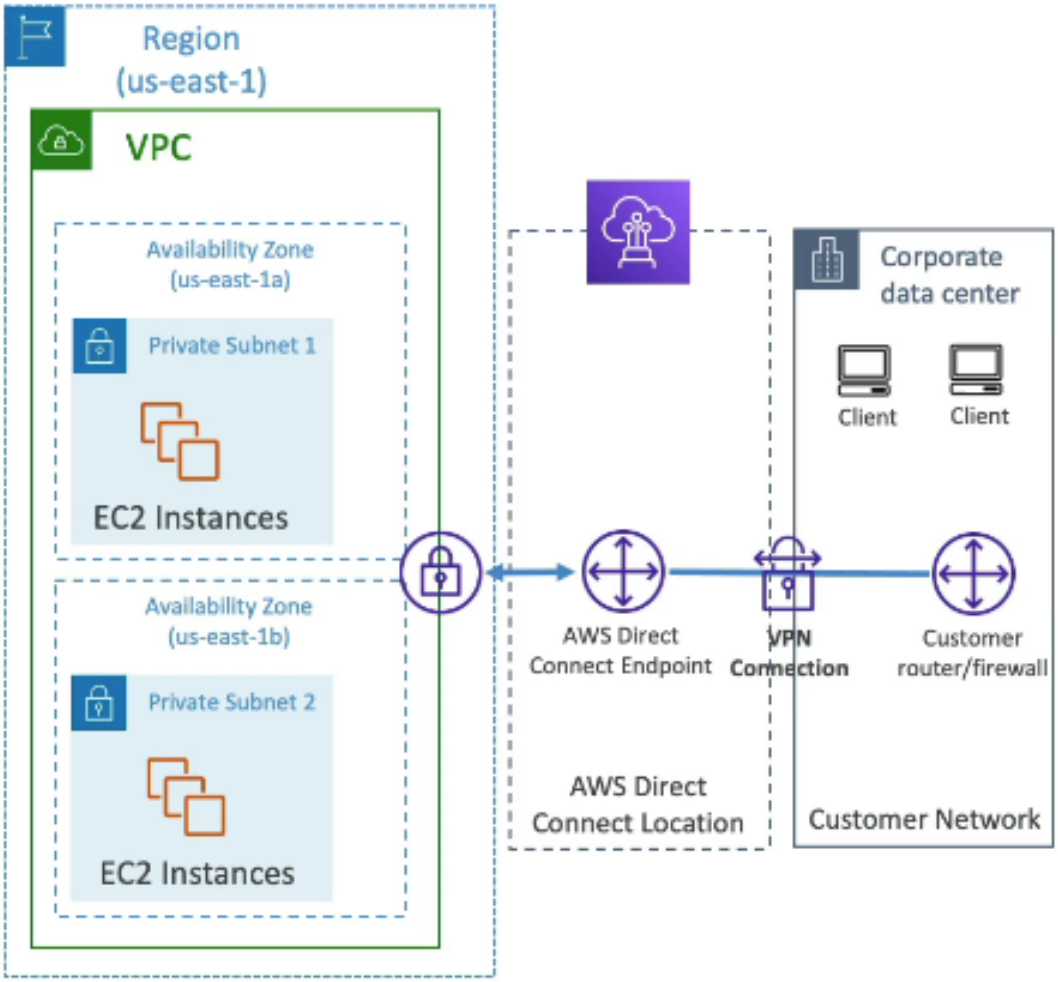
DX Connection Types

- **Dedicated Connections**: 1Gpbs, 10 Gbps, 100 Gbps capacity. A physical ethernet port is dedicated to a customer. We make request to AWS first, then it will be completed by AWS DX partners.
- **Hosted Connections**: 50Mbps, 500 Mbps, to 10 Gbps. Connection requests are made via AWS DX partners. Capacity can be added or removed on demand.

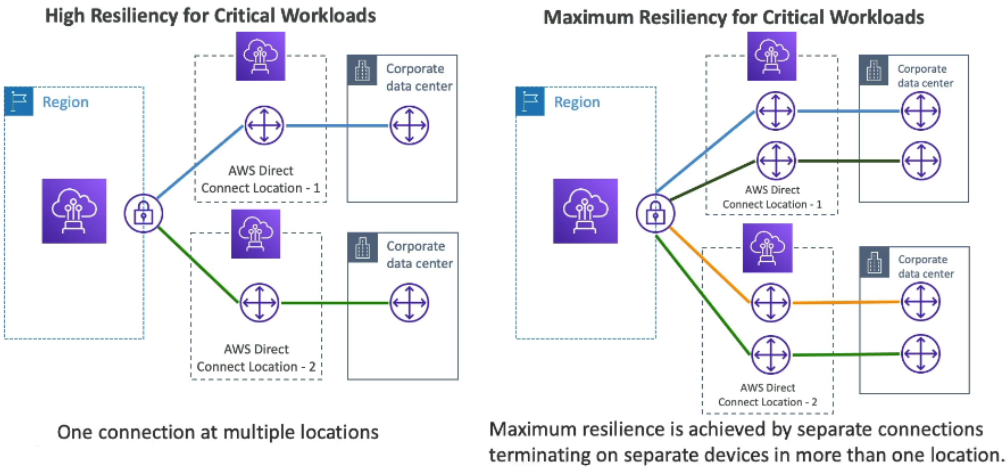
Lead times are often longer than one month to establish a new connection.

DX Encryption

Data in transit is not encrypted but is private. DX + VPN provides an IPsec-encrypted private connection.

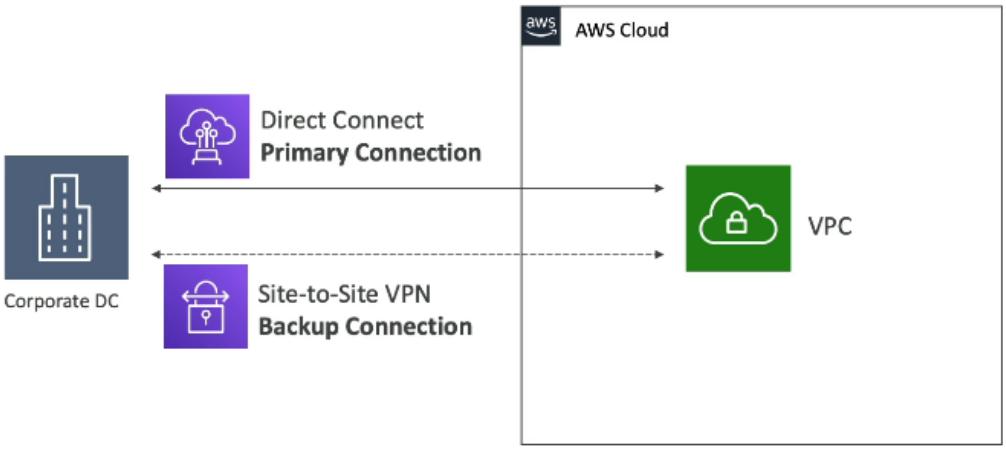


▼ **Solution Architecture - DX Resiliency Patterns.**



▼ **Solution Architecture - Site-to-Site VPN as a Backup.**

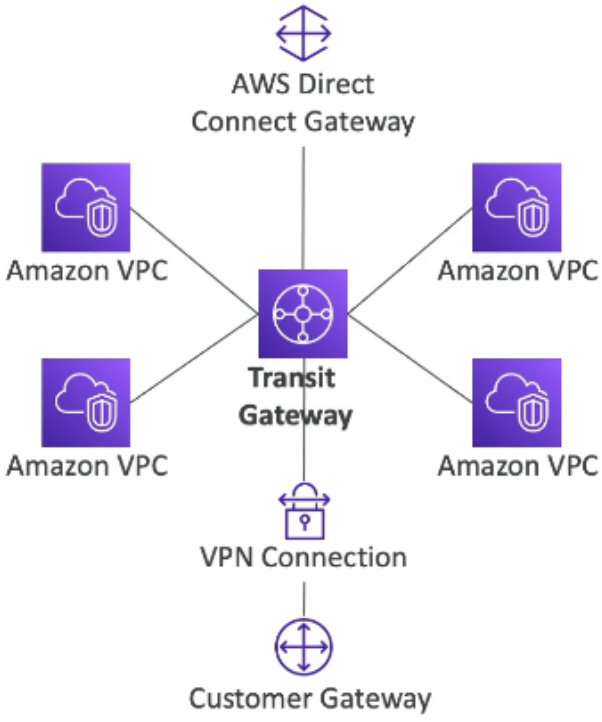
In case DX fails, we can set up a backup DX connection (expensive) or a Site-to-Site VPN connection (cheaper).



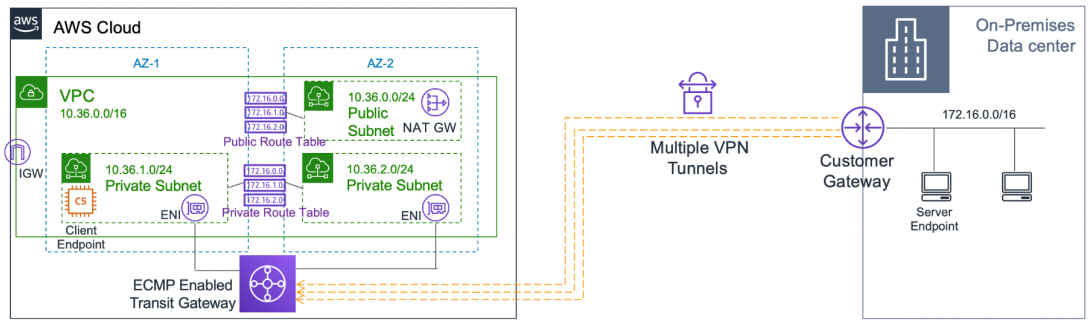
▼ ☐ **Transit Gateway.**

Transit Gateway is used for having transitive peering between thousands of VPC and on-premises. We need only one single gateway to provide this functionality. It works with DX Gateway and VPN connections. We must configure route tables to limit which VPC can talk with other VPC.

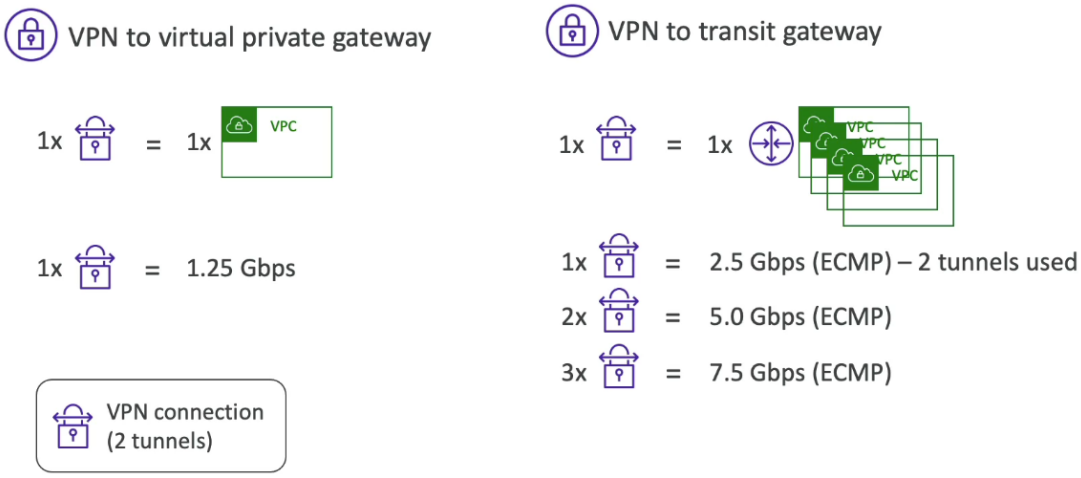
We can peer Transit Gateways across regions and share cross-account using RAM. It supports IP Multicast (**not supported by any other AWS service**).



ECMP (*Equal-cost multi-path routing*) - routing strategy that allows to forward a packet over multiple best paths. It is used to create multiple Site-to-Site VPN connections to increase the bandwidth of our connection to AWS.

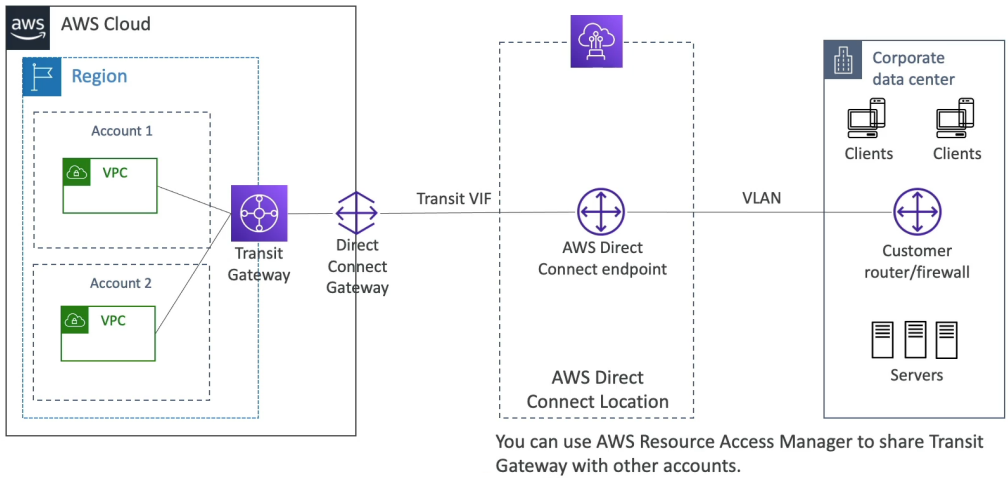


Throughput with ECMP



▼ **Solution Architecture - Share DX between Multiple Accounts.**

- **Cost Efficiency:** By sharing a DX connection, we reduce the cost of maintaining multiple dedicated connections.
- **Centralized Management:** It centralizes network management and simplifies configuration, especially if we have multiple AWS accounts.
- **Reduced Complexity:** It reduces the complexity of setting up and managing multiple DX connections.

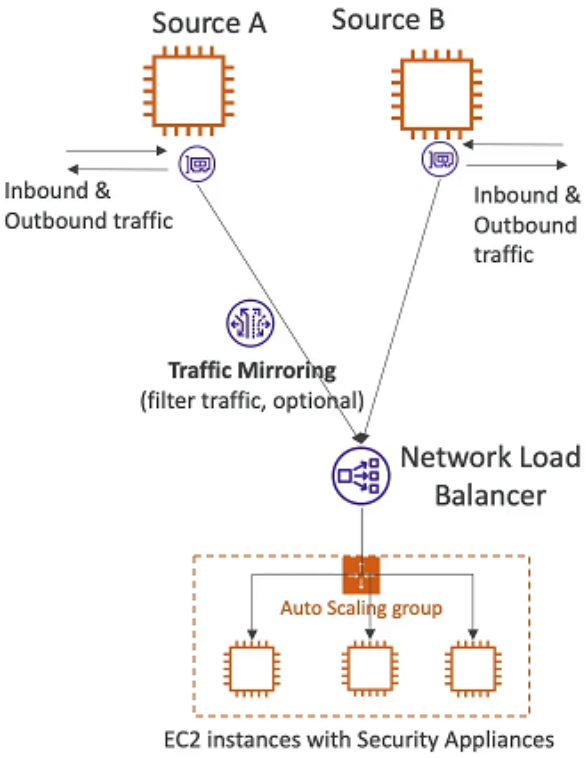


You can use AWS Resource Access Manager to share Transit Gateway with other accounts.

▼ ☐ **VPC Traffic Mirroring.**

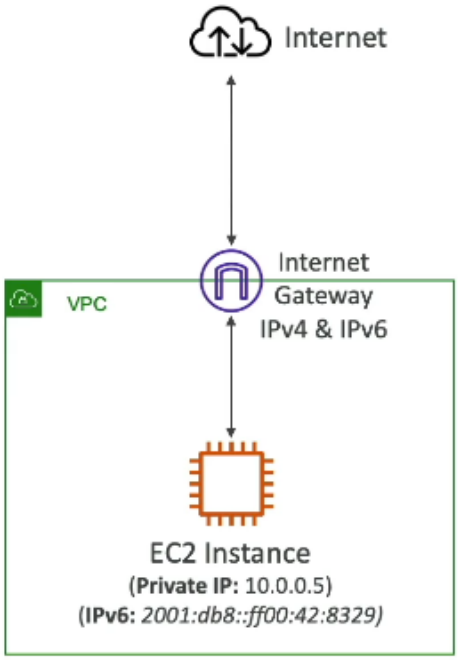
Traffic Mirroring - allows us to capture and inspect network traffic in our VPC. It routes the traffic to security appliances that we manage.

We can capture the source traffic (ENIs) and target traffic (ENI or NLB). It's possible to capture all packets or capture the packets of our interest (optionally, truncate packets). Source and target can be in the same VPC or different VPCs (VPC Peering). Traffic mirror filters can't inspect the actual packet of the incoming and outgoing traffic. It is **primarily used for monitoring and analysis**.



▼ ☑ **IPv6 for VPC.**

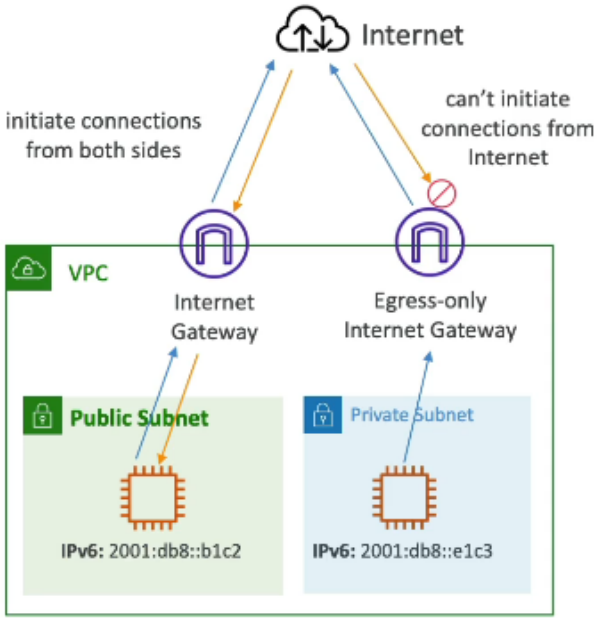
Every IPv6 address in AWS is public and Internet-routable (no private range). IPv4 cannot be disabled for our VPC and subnets. We can enable IPv6 to operate in **dual-stack mode**. Our EC2 instances will get at least a private internal IPv4 and a public IPv6.



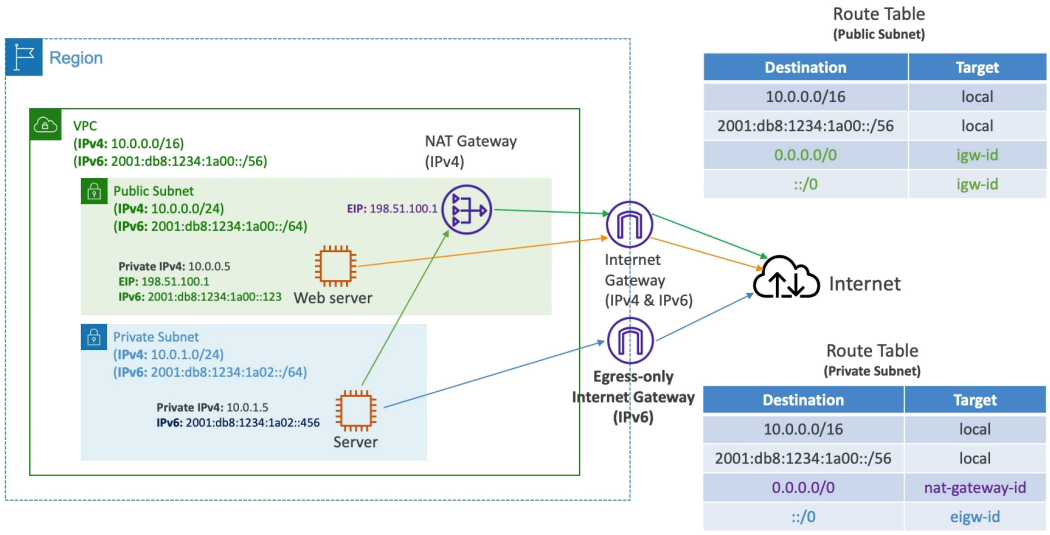
If we cannot launch an EC2 instance in our subnet. It's not because it cannot acquire an IPv6, it's because there are no available IPv4 in our subnet. To solve that we should create a new IPv4 CIDR in our subnet.

▼ ☑ **Egress-Only Internet Gateway.**

Egress-Only Internet Gateway is used for IPv6 only (similar to a NAT Gateway bur for IPv6). It allows outbound connections over IPv6 for instances in our VPC while preventing the internet to initiate an IPv6 connection to our instances. We must update the route tables.



IPv6 Routing



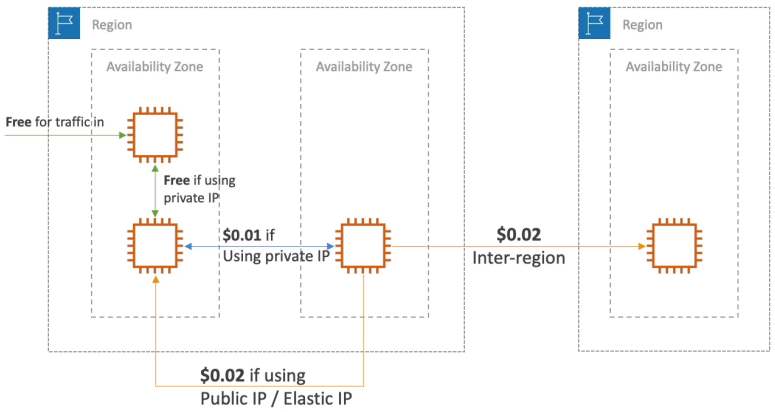
Route Table
(Public Subnet)

Destination	Target
10.0.0.0/16	local
2001:db8:1234:1a00::/56	local
0.0.0.0/0	igw-id
::/0	igw-id

Route Table
(Private Subnet)

Destination	Target
10.0.0.0/16	local
2001:db8:1234:1a00::/56	local
0.0.0.0/0	nat-gateway-id
::/0	eigw-id

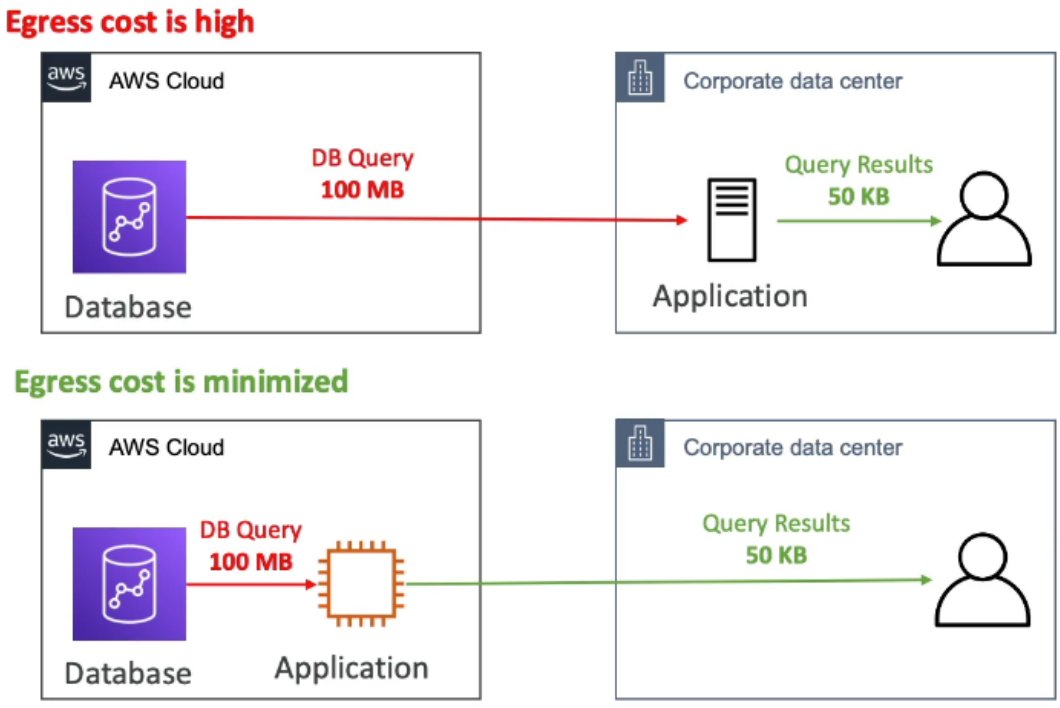
▼ ☑ **Networking Costs in AWS.**



- Use Private IP instead of Public IP for good savings and better network performance
- Use same AZ for maximum savings (at the cost of high availability)

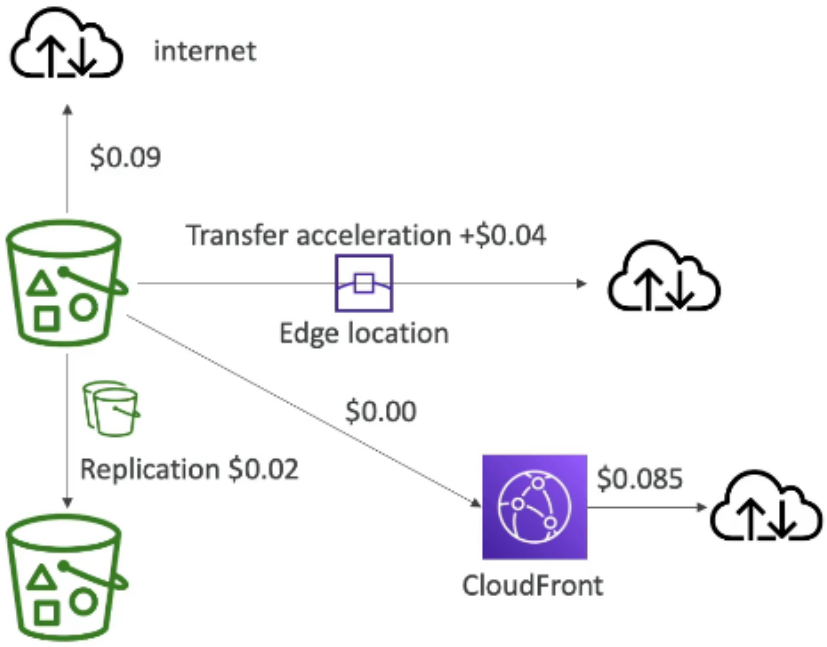
Egress traffic - outbound traffic (AWS → Outside). DX locations that are co-located in the same AWS Region result in lower cost for egress network.

Ingress traffic - inbound traffic (Outside → AWS), typically free.

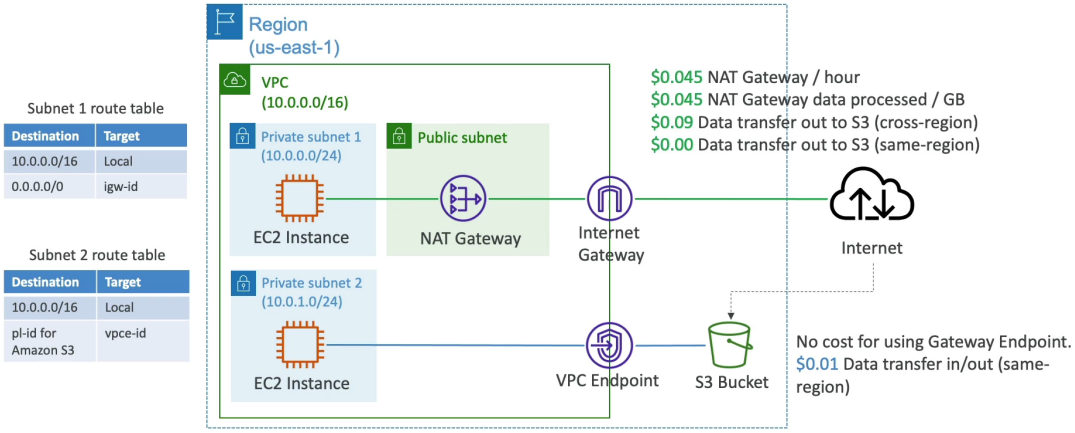


We should try to keep as much internet traffic within AWS to minimize costs.

S3 Data Transfer Pricing



NAT Gateway vs VPC Endpoint Pricing



Subnet 1 route table

Destination	Target
10.0.0.0/16	Local
0.0.0.0/0	igw-id

Subnet 2 route table

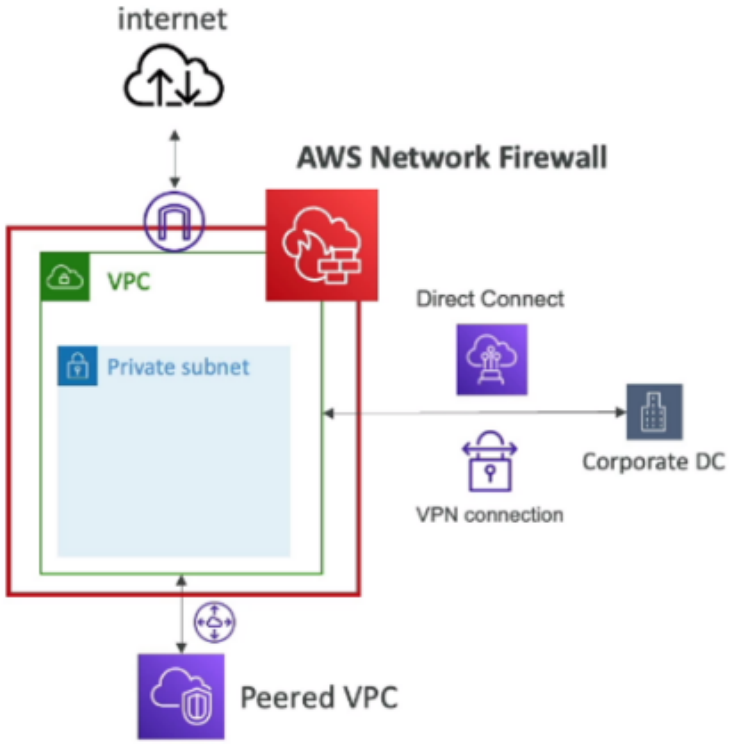
Destination	Target
10.0.0.0/16	Local
pl-id for Amazon S3	vpce-id

▼ ☑ AWS Network Firewall.

AWS Network Firewall protects our entire VPC (from layer 3 to layer 7). We can inspect traffic in any direction:

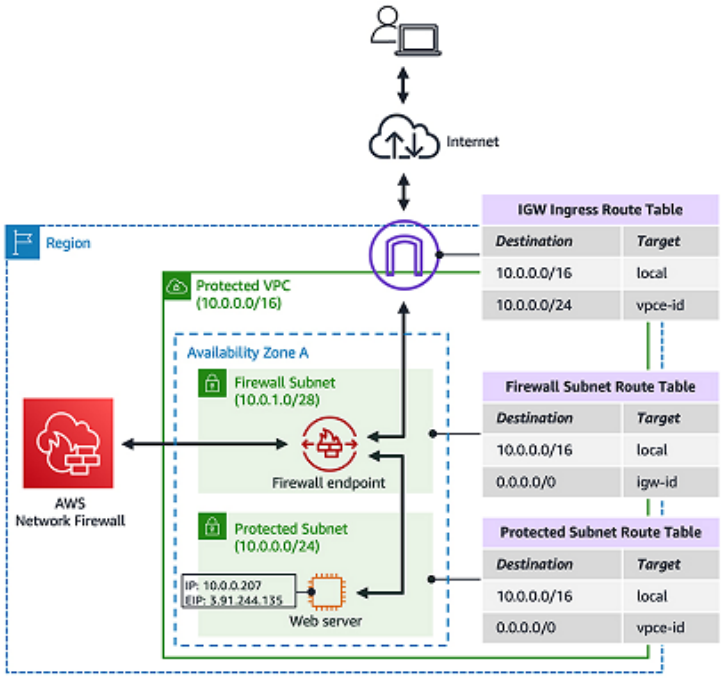
- VPC to VPC traffic
- Outbound to internet
- Inbound to internet

- To/from Direct Connect & Site-to-Site VPN



Internally, the AWS Network Firewall uses the AWS Gateway Load Balancer. Rules can be centrally managed cross-account to apply to many VPCs.

Network Firewall Endpoint - a virtual appliance deployed within our VPC that inspects and filters network traffic based on the firewall rules we configure. It provides the actual interface through which the firewall processes the traffic. It's responsible for applying the policies and rules we define to **control incoming and outgoing network traffic**. We can allow, drop or alert for the traffic that matches the rules. All rule matches can be sent to S3, CloudWatch Logs and KDF.



▼ ☑ **HPC & Networking.**

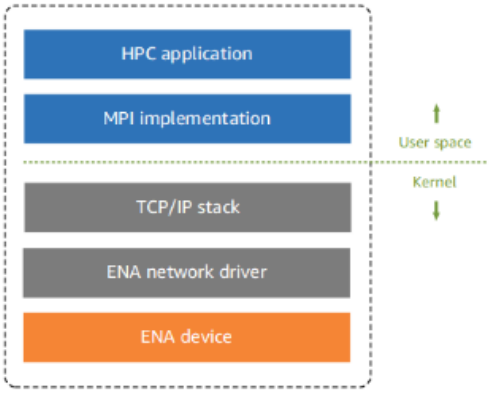
Enhanced Networking via SR-IOV (Single Root I/O Virtualization) - a technology that allows network traffic to bypass the hypervisor and directly access the network interface card (NIC).

It's typically used in instances that need high throughput and low latency but do not require the advanced features of ENA or EFA.

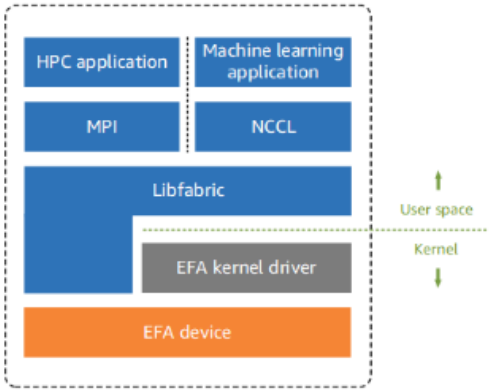
Elastic Network Adapter (ENA) - a custom network interface designed by AWS that supports high-performance networking. It offers better flexibility and advanced features compared to SR-IOV. ENA is suited for workloads that need higher network bandwidth (**up to 100 Gbps**).

Elastic Fabric Adapter (EFA) - a network interface designed for tightly coupled HPC and ML workloads that require inter-instance communication with low latency and high throughput. **EFA builds on ENA**, offering additional features like hardware-based message passing.

The OS-bypass capabilities of EFAs are not supported on Windows instances. If we attach an EFA to a Windows instance, the instance functions as an ENA without the added EFA capabilities.



Traditional HPC software stack in EC2



HPC software stack in EC2 with EFA

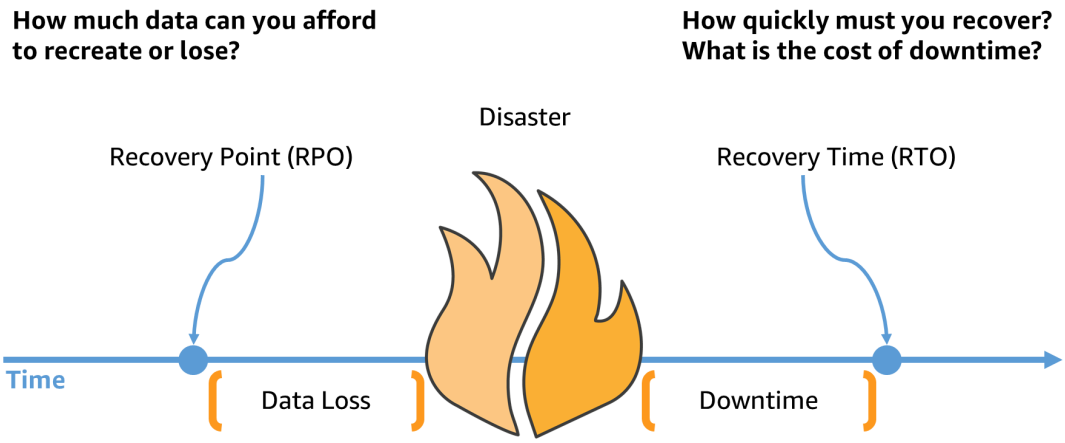
AWS ParallelCluster is an open-source cluster management tool that simplifies the deployment and management of HPC clusters in the AWS Cloud. It allows us to easily set up a multi-node cluster for our HPC workloads. There is an ability to enable EFA on the cluster and improve network performance.

Disaster Recovery & Migrations

▼ ☐ **Disaster Recovery in AWS.**

RPO (Recovery Point Objective) - indicates the maximum amount of data that can be lost measured in time. It's the point in time to which your data must be recovered after an incident.

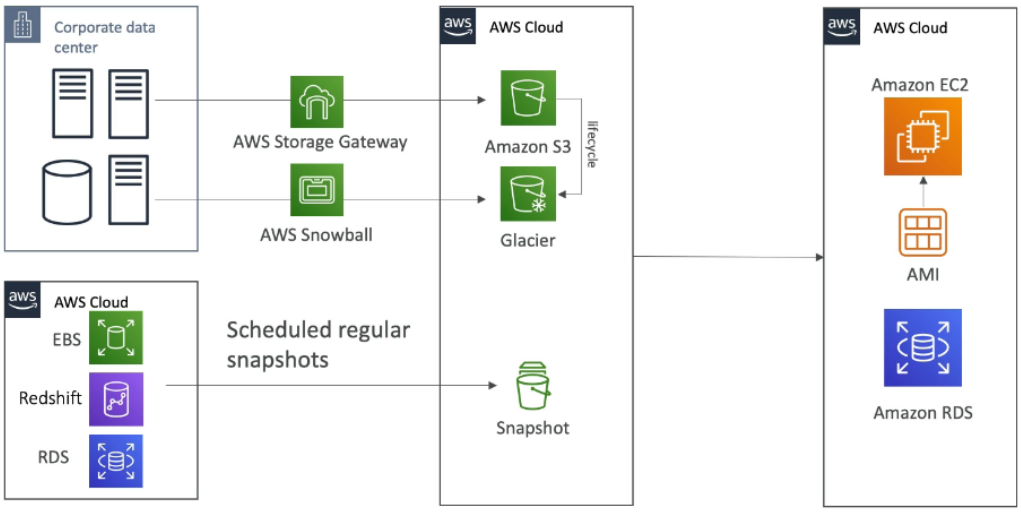
RTO (Recovery Time Objective) - maximum acceptable time that it takes to restore a system or service after a disruption. It reflects how quickly you need to recover after a failure.



Disaster Recovery Strategies

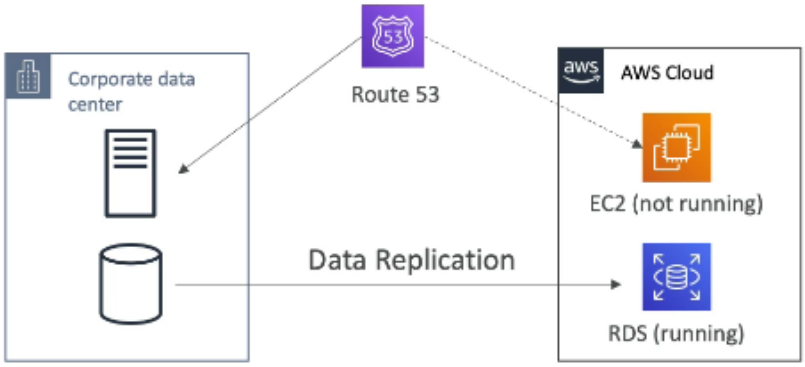
- **Backup & Restore** - most basic DR strategy where data is backed up to AWS storage services and restored in the event of a disaster. Suitable for non-critical applications where downtime can be tolerated.

  RTO and RPO are typically longer because the restoration process involves re-provisioning infrastructure and retrieving data.

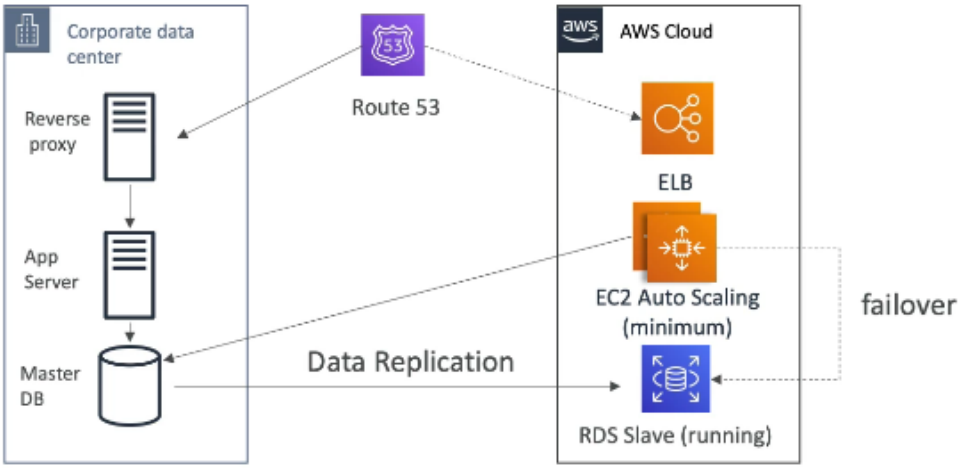


- **Pilot Light** - a minimal version of an environment is always running in AWS, which can be scaled up to a full-scale production environment when needed. Suitable for critical applications where a minimal version can handle some traffic until full scaling occurs.

Faster than the backup and restore strategy but still involves some time for scaling up and provisioning additional resources. The RPO can be near real-time since core components are active.

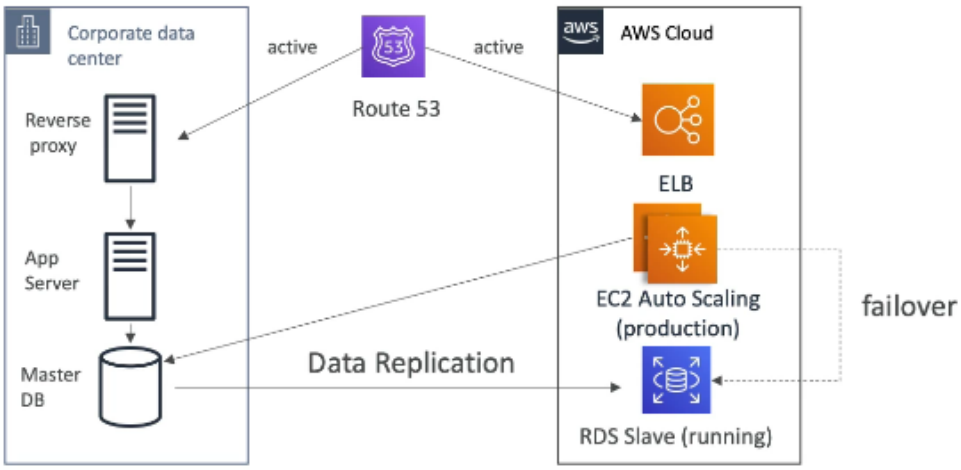


- **Warm Standby** - a scaled-down version of a fully functional environment is running in AWS, which can be quickly scaled up to handle production load. Suitable for critical applications requiring reduced RTO and RPO.



- **Multi-Site (Hot-Site)** - the most robust DR strategy where multiple active sites run simultaneously, providing full redundancy and load balancing. Suitable for mission-critical applications requiring near-zero downtime and high availability.

  This approach offers the lowest RTO and RPO, potentially near zero, as both sites are live and in sync.



Disaster Recovery Strategies Comparison

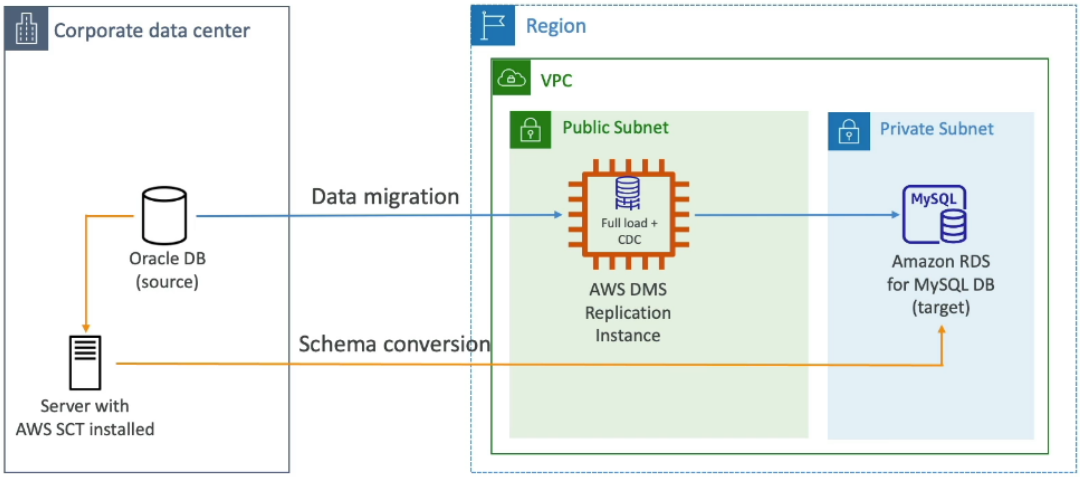
Strategy	RTO	RPO	Cost	Complexity	Typical Use Case
Backup and Restore	Hours	Hours	Low	Low	Low-criticality workloads with tolerant downtime
Pilot Light	Minutes	Minutes	Moderate	Moderate	Critical apps needing quicker recovery but cost-sensitive
Warm Standby	Minutes	Near real-time	Higher	Moderate	Important applications requiring minimal downtime
Multi-Site	Seconds	Near zero	High	High	Mission-critical apps demanding high availability

▼ ☑ **DMS.**

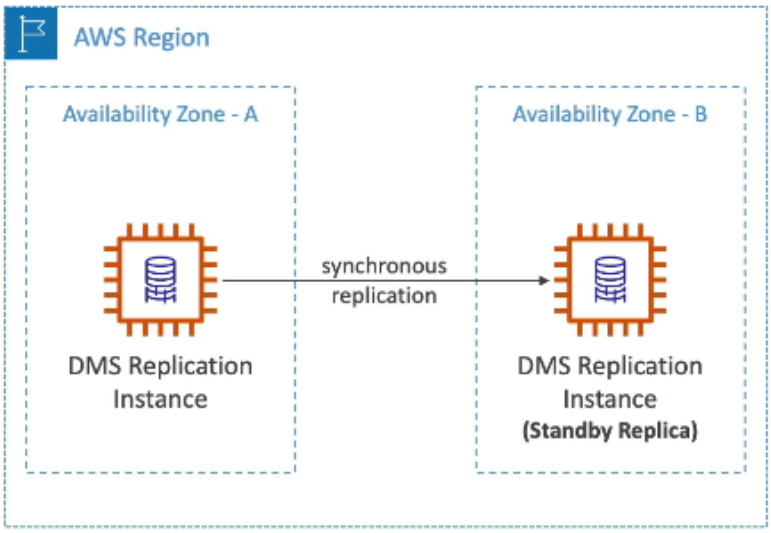
With **DMS** we can quickly and securely migrate databases to AWS. The source database remains available during the migration. DMS supports both homogeneous and heterogeneous migrations. It can migrate data from various sources like on-premises databases or other AWS services. DMS includes a schema conversion tool (AWS SCT) that helps convert the schema of source databases to match the target database schema.

- **Homogeneous migrations** - involves moving data between two systems that have the same DBMS.
- **Heterogeneous migrations** - involves moving data between two systems that have different DBMS.

It can be used for continuous data replication with minimal latency (keep source and target databases in sync for ongoing data changes).

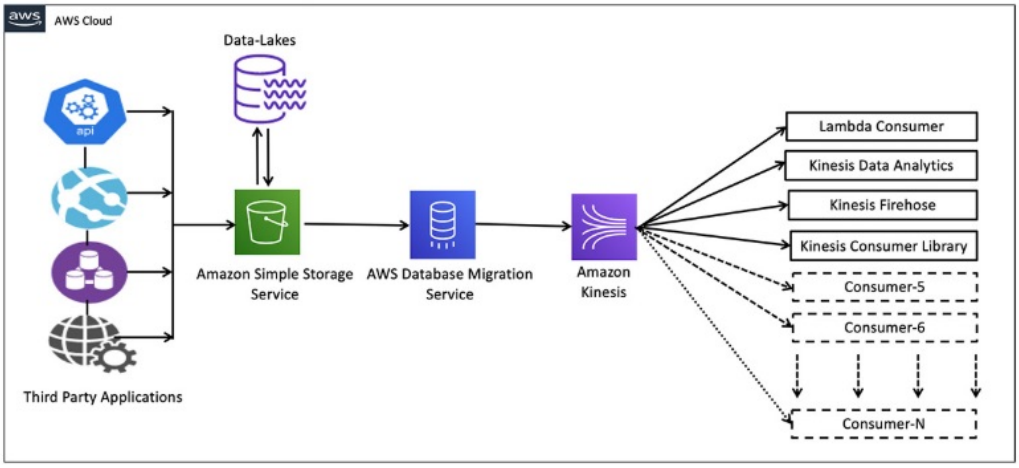


When Multi-AZ is enabled, DMS provision and maintains a synchronously stand replica in a different AZ. It eliminates I/O freezes and provides data redudancy.



▼ Solution Architecture - S3 to KDS

AWS DMS lets us expand the existing application to stream data from Amazon S3 into Amazon Kinesis Data Streams for real-time analytics without writing and maintaining new code.



▼ ☑ **RDS & Aurora Migrations.**

RDS MySQL → Aurora MySQL

- **Option 1**: DB Snapshots from RDS MySQL restored as MySQL Aurora DB.
- **Option 2**: Create an Aurora Read Replica from our RDS MySQL and when the replication lag is 0, promote it as its own DB cluster (can take time and money).

External MySQL → Aurora MySQL

- **Option 1**: Use Percona XtraBackup to create a file backup in S3 and then create an Aurora MySQL DB from S3.
- **Option 2**: Create an Aurora MySQL DB and use mysqldump utiliy to migrate MySQL into Aurora (slower than option 1).

RDS Postrgres → Aurora Postgres

- **Option 1**: DB Snapshots from RDS Postgres restored as Postgres Aurora DB.
- **Option 2**: Create an Aurora Read Replica from our RDS Postgres and when the replication lag is 0, promote it as its own DB cluster (can take time and money).

External Postgres → Aurora Postgres

- Create backup and put it in S3 then import it using the aws_s3 Aurora extension.

If both databases are up and running, use DMS.

▼ ☑ **On-premises Strategies with AWS.**

- **Amazon Linux 2 AMI as VM** (.iso format)
- **VM Import/Export** - migrate existing applications into EC2, create a DR repository strategy for our on-premise VMs. Can export back the VMs from EC2 to on-premise.
- **AWS Application Discovery Service** - gather information about our on-premise servers to plan a migration.
- **AWS Database Migration Service** - replicate on-premise ⇒ AWS, AWS ⇒ AWS, AWS ⇒ on-premise.
- **AWS Server Migration Service** - incremental replication of on-premise live servers to AWS.

▼ ☑ **AWS Backup.**

AWS Backup is a fully-managed service to centrally manage and automate backups across AWS services. It can have on-demand and scheduled backups. Supports PITR.

Cross-Region Backup - backing up data from one AWS region to another. This is useful for disaster recovery scenarios where we want to ensure data availability even if an entire AWS region becomes unavailable due to a disaster or outage.

Cross-Account Backup - backing up data from AWS resources in one AWS account to AWS Backup vaults in another AWS account. This is useful for scenarios where we want to centralize backups for multiple AWS accounts in a single backup location.

Backup Vault Lock - enforces WORM state for all the backups that we store in our AWS Backup Vault. It is additional layer of defense to protect our backups against inadvertend or malicious delete operations. Even root user cannot delete backups when enabled.

▼ ☑ **Application Discovery Service & Application Migration Service.**

Application Discovery Service helps enterprises in planning their migration to AWS by discovering on-premises applications and **collecting configuration, usage, and behavior data**. It provides insights into application dependencies and resource utilization.

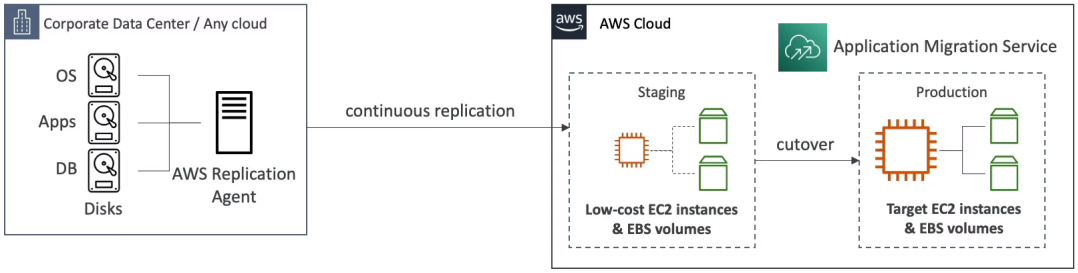
Agentless Discovery (**AWS Agentless Discovery Connector**) - uses network-based data collection methods to gather information about on-premises servers and applications without requiring the installation of agents.

Agent-based Discovery (**AWS Application Discovery Agent**) - uses lightweight agents installed on on-premises servers to collect detailed data about system configurations, running processes, network connections, and performance metrics.

Resulting data can be viewed within AWS Migration Hub.

Application Migration Service (MGN) simplifies and the migration of applications from physical, virtual, and cloud infrastructure to AWS. It automates the migration process, reducing the time, cost, and effort involved in traditional lift-and-shift migrations.

Implementation begins by installing the AWS Replication Agent on our source servers. When we launch Test or Cutover instances, AWS Application Migration Service automatically converts our source servers to boot and runs natively on AWS.



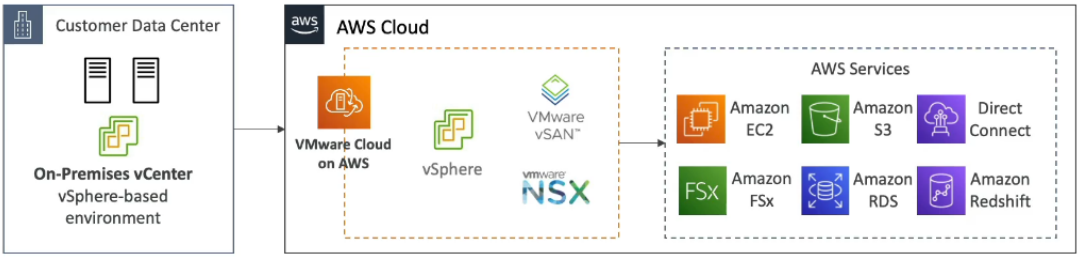
▼ ☑ **Transfering Large Datasets into AWS.**

- **Over the internet/Site-to-Site VPN** - ideal for small to moderate data volumes where latency isn't a significant concern.
- **Over Direct Connect** - suitable for transferring large datasets where consistent, high-throughput, and low-latency connectivity is required.
- **Over Snowball** - ideal when dealing with extremely large datasets or when we have limited or unreliable internet connectivity.
- **For on-going replication/transfers** - Site-to-Site VPN/DX with DMS or DataSync.

▼ ☑ VMware Cloud on AWS.

Some customers use VMware Cloud to manage their on-premises Data Center. They want to extend Data Center capacity to AWS, but keep using the VMware Cloud software.

VMware Cloud on AWS is a service that allows businesses to run VMware's virtualization technologies on AWS infrastructure. Essentially, it extends our VMware environment to the AWS cloud



▼ ☐ AWS Deployment Strategies.

All-at-Once (Big Bang) - deploys the new version of the application to all instances at once. It is simple and fast, with no need for complex infrastructure setups. It has high risk of downtime if the new version has issues, no easy rollback. If something goes wrong, the entire system can be affected.

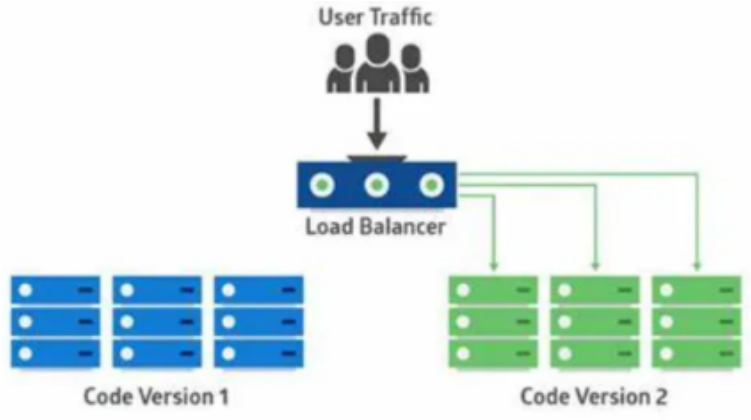
Use Cases: **Non-critical applications where downtime is acceptable or where rollback procedures are straightforward**.

Rolling Deployment - gradually replaces instances of the old version with instances of the new version. Reduces the risk of full-scale failure and can be quickly fixed or rolled back.

Use Cases: **Applications that need to minimize downtime but can tolerate temporary inconsistencies**.

Blue-Green Deployment - two identical environments are maintained: one (blue) running the current version and another (green) running the new version. Traffic is switched from the blue environment to the green one once the new version is ready.

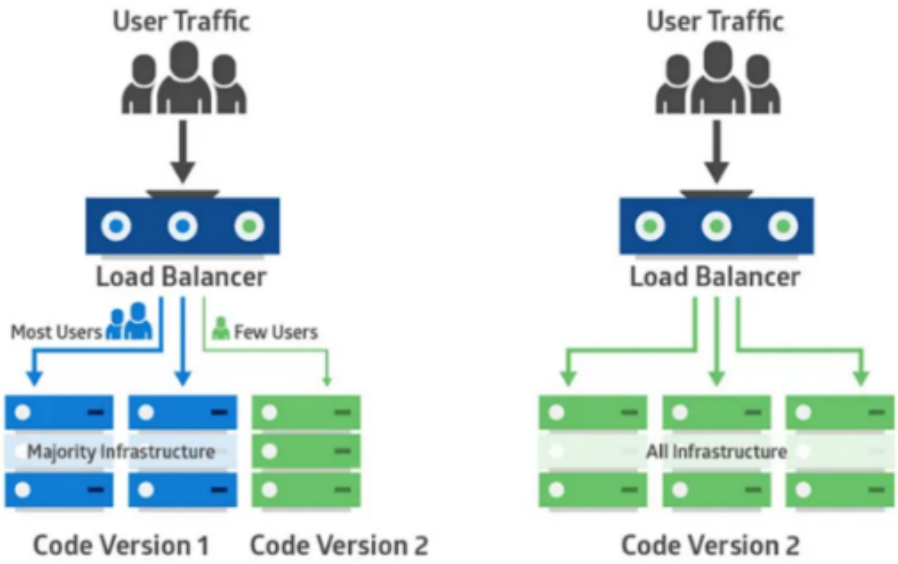
It is zero-downtime deployment, easy rollback by switching traffic back to the blue environment. Testing and validation can be done on the green environment.



Use Cases: **Applications requiring zero downtime and where rollback needs to be swift and reliable**.

Canary Deployment - small portion of the traffic is directed to the new version while the rest remains on the old version. Gradually, more traffic is shifted to the

new version as confidence in its stability grows. It allows for real-world testing with minimal risk.



Use Cases: Applications with a **large user base or critical services where early feedback and incremental rollout are important**.

A/B Testing Deployment - similar to Canary deployment, but with a focus on running two or more versions simultaneously to compare performance or user experience. Different users are directed to different versions.

Use Cases: **Testing new features, UI/UX changes, or optimizations**.

Shadow Deployment - new version is deployed alongside the old version, but the new version does not serve production traffic. Instead, it processes a copy of the production traffic to see how it behaves.

Use Cases: Validating major updates or entirely new services without risking production stability.

Other Services

▼ ☑ **CloudFormation.**

CloudFormation is a declarative way of outlining our AWS Infrastructure, for any resources.

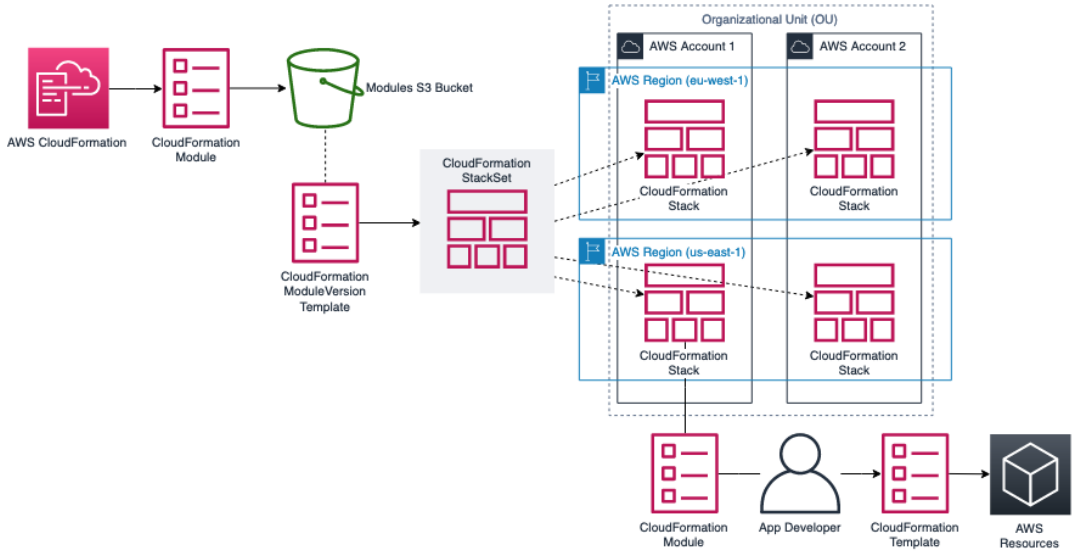
> Example: We want security group, two EC2 instances using that security group, an S3 bucket and ELB in front of these machines. CloudFormation creates all those for us, in the right oreder, with exact configuration that we specify.

Benefits of CloudFormation

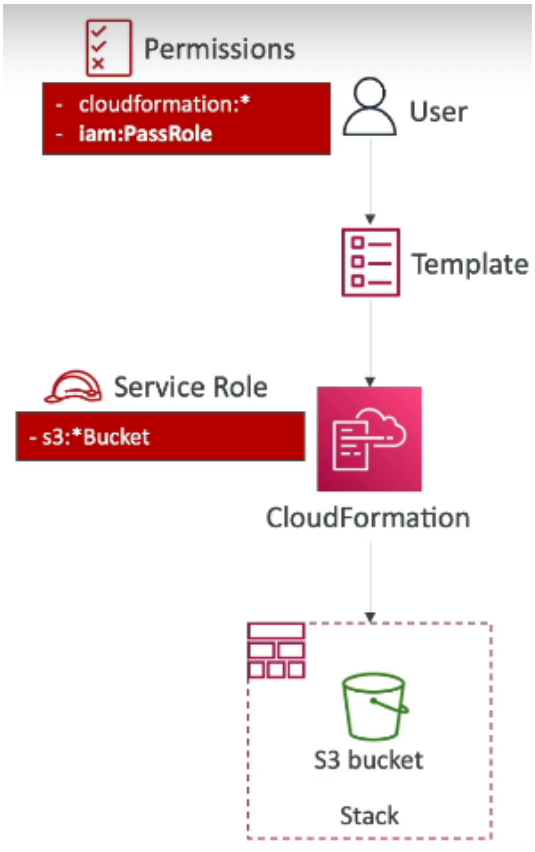
- Infrastructure as code - No resources are manually created, which is excellent for control. Changes to the infrastructure are reviewed through code.
- Cost - Each resources within stack is tagged with an identifier so we can easily see how much a stack costs us. We can estimate the costs of our resources using the CloudFormation template.
- Productivity - Ability to destroy and re-create an infrastructure on the cloud on the fly. We can automate generation of diagrams for our templates.

Application Composer - A visual tool designed to simplify the process of building serverless applications on AWS. It enables developers to drag and drop various AWS services to create an architecture diagram that represents their application.

- **Stacks** are the core unit of deployment in CloudFormation. A stack is a collection of AWS resources that we can manage as a single unit. When you create, update, or delete a stack, CloudFormation automatically provisions or deletes the associated resources in a predictable way.
- **Template** is a JSON or YAML formatted text file that describes the resources AWS CloudFormation will provision in our stack.
- **StackSets** extend the functionality of stacks by allowing us to manage stacks across multiple accounts and regions.



CloudFormation Service Role - IAM role that allows CloudFormation to create/update/delete stack resources on our behalf. It is useful if we want to achieve the least privilege principle, but we dont want to give the user all the required permissions to create the stack resource. User must have iam:PassRole permissions.

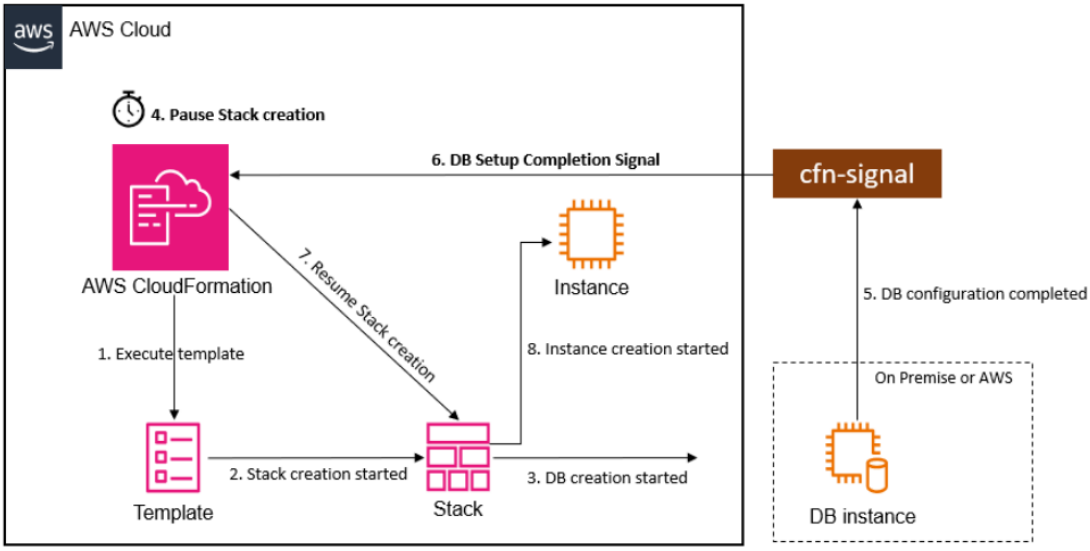


▼ ☑ **CloudFormation Policy Attributes.**

CreationPolicy attribute allows us to specify a policy for how CloudFormation should handle the creation of a resource. This is often used with resources that need additional time to become fully functional or require validation before they are considered created. We can prevent atrribute status from reaching create complete until AWS CloudFormation receives a specified number of success signals or the timeout period is exceeded. To signal a resource, we can use the cfn-signal helper script or SignalResource API.

```
Resources:
  MyInstance:
    Type: AWS::EC2::Instance
    CreationPolicy:
      ResourceSignal:
```

```
        Count: 1
        Timeout: PT15M
```



DependsOn attribute specifies that the creation of one resource depends on the creation of another resource. This is useful when you need to control the order in which resources are created or updated.

```
Resources:
  MyBucket:
    Type: AWS::S3::Bucket

  MyBucketPolicy:
    Type: AWS::S3::BucketPolicy
    DependsOn: MyBucket
```

UpdatePolicy attribute defines how CloudFormation should handle updates to a resource. This is useful for managing rolling updates or handling update failures for certain resources.

```
Resources:
  MyAutoScalingGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
```

```
    UpdatePolicy:
      AutoScalingRollingUpdate:
        MinInstancesInService: 1
        MaxBatchSize: 1
        PauseTime: PT5M
```

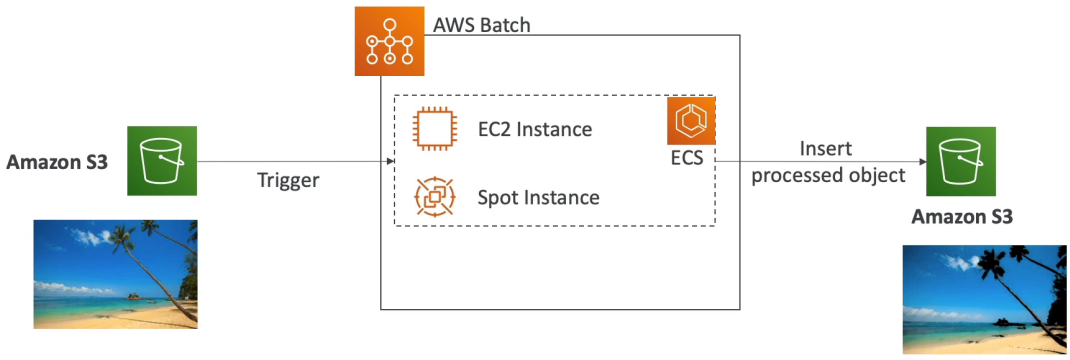
UpdateReplacePolicy attribute specifies what CloudFormation should do with a resource when it is replaced during an update. This is useful for controlling the behavior of resources like RDS instances or S3 buckets when they are replaced.

```
Resources:
  MyBucket:
    Type: AWS::S3::Bucket
    UpdateReplacePolicy: Retain
```

☑ AWS Batch.

AWS Batch is a fully managed batch processing service at any scale (efficiently runs 100 000s of computing batch jobs on AWS). A batch job is a job with a start and an end.

Batch jobs are defined as Docker images and run on ECS. Batch will dynamically launch EC2 instances or Spot Instances and will provision the right amount of compute/memory.



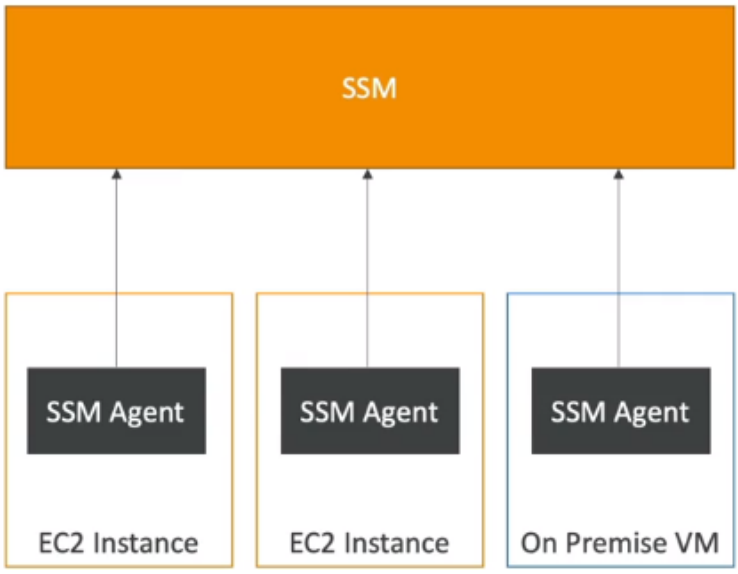
☑ AWS SSM.

SSM is a hybrid service that helps us manage our EC2 and On-Premises systems at scale. We can get operational insights about the state of our infrastructure.

Most important features of SSM:

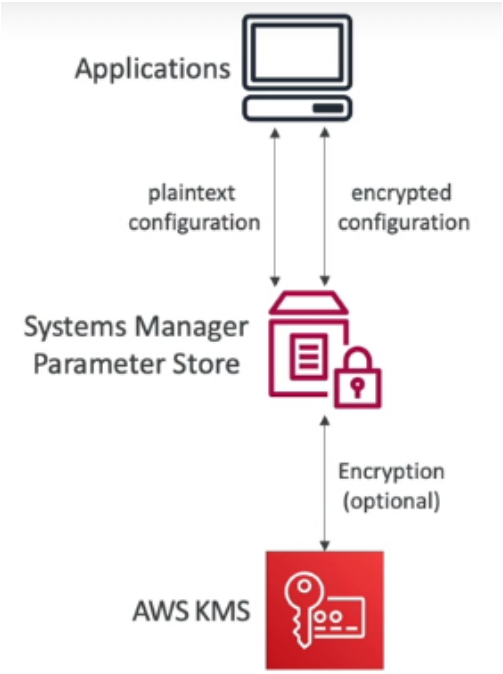
- Patching automation for enhanced compliance.
- Run commands across an entire fleet of servers.
- Store parameter configuration with the SSM Parameter Store.

We need to install the SSM agent onto the systems we control (installed by default on Amazon Linux AMI). If an instance cant be controlled with SSM, it's probably an issue with the SSM agent. With SSM agent we can run commands, patch and configure our servers.



SSM Session Manager - allows us to start SSH on our EC2 and on-premises servers. No SSH access, bastion hosts or SSH keys needed. Also if we want better security we can disable port 22.

SSM Parameter Store - it is a secure storage for configuration and secrets (API keys, passwords, configurations ...). It is serverless, scalable and durable. Also it is secure because we control access permissions using IAM.



▼ ☐ **SSM Advanced.**

- **SSM Run Command**

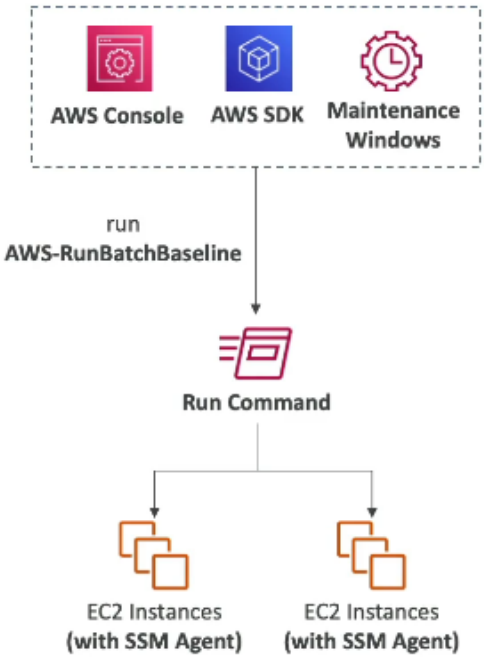
  It lets us to execute a script or just run a command. We can run command across multiple instances (using resource groups) and we dont need SSH. It can be integrated with IAM & CloudTrail and can be invoked using EventBridge.

  Command output can be shown in the AWS Console, sent to S3 bucket or CloudWatch Logs. We can also send notifications to SNS about command status.

- **SSM Patch Manger**

  It automates the process of patching managed instances (OS updates, applications updates, security updates). It supports EC2 instances and on-premises servers. Can scan instances and generate patch compliance.

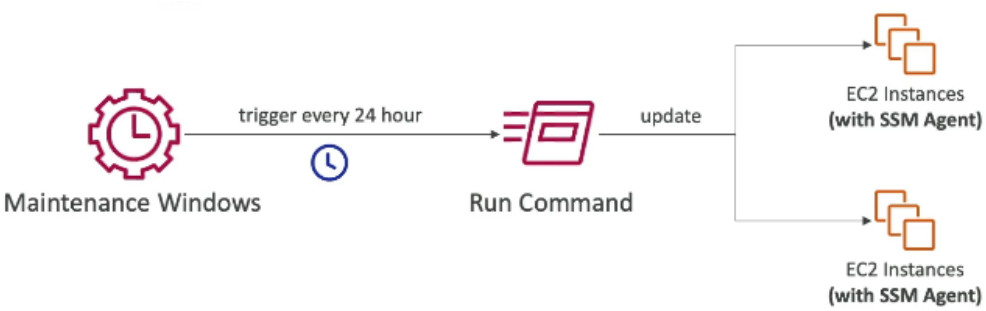
  We can patch on-demand or on a schedule using Maintenance Windows.



- **SSM Maintenance Windows**

  It defines a schedule for when to perform actions on our instances (e.g. OS patching, updating drivers, installing software).

  Maintenance Windows contains: shcedule, duration, set of registered instances and set of registered tasks.



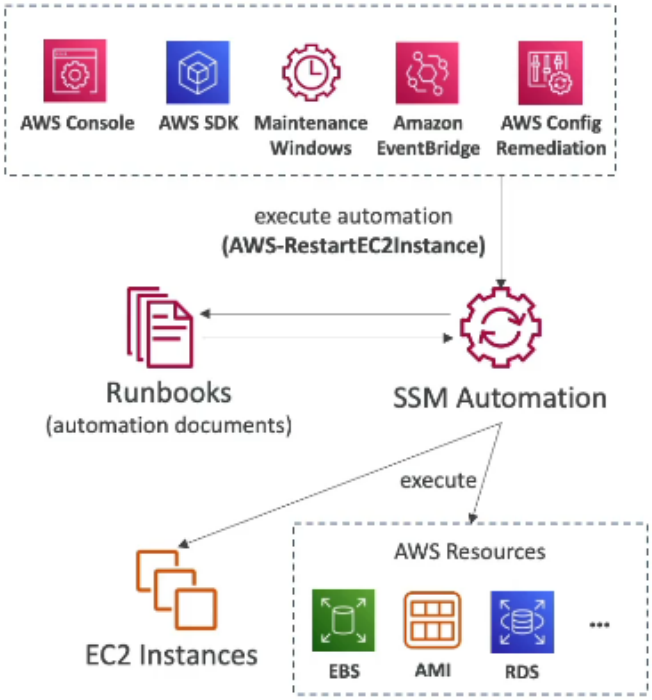
- **SSM Automation**

  It simplifies common maintenance and deployment tasks of EC2 instances and other AWS resources (e.g. restart instances, create an AMI, EBS snapshot).

Automation Runbook - define actions performed on our EC2 instances or AWS resources.

Automation can be triggered using:

- AWS Console, AWS CLI or SDK
- EventBridge
- Using Maintenance Windows
- AWS Config



▼ ☑ **AWS Elastic Beanstalk.**

Elastic Beanstalk is a **PaaS** offered by AWS. It is developer-centric view of deploying an application on AWS (**web and non web**). It abstracts much of the infrastructure management, allowing developers to focus on writing code (we still have full control over the configuration) and developing features rather than dealing with the underlying infrastructure. Only the application code is the responsibility of the developer. Can be used for deploying Docker containers.

Beanstalk is free but we need to pay for the underlying instances.

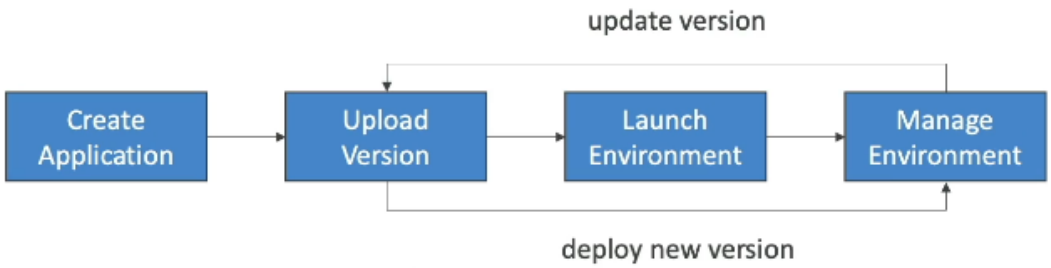
Elastic Beanstalk architecture models:

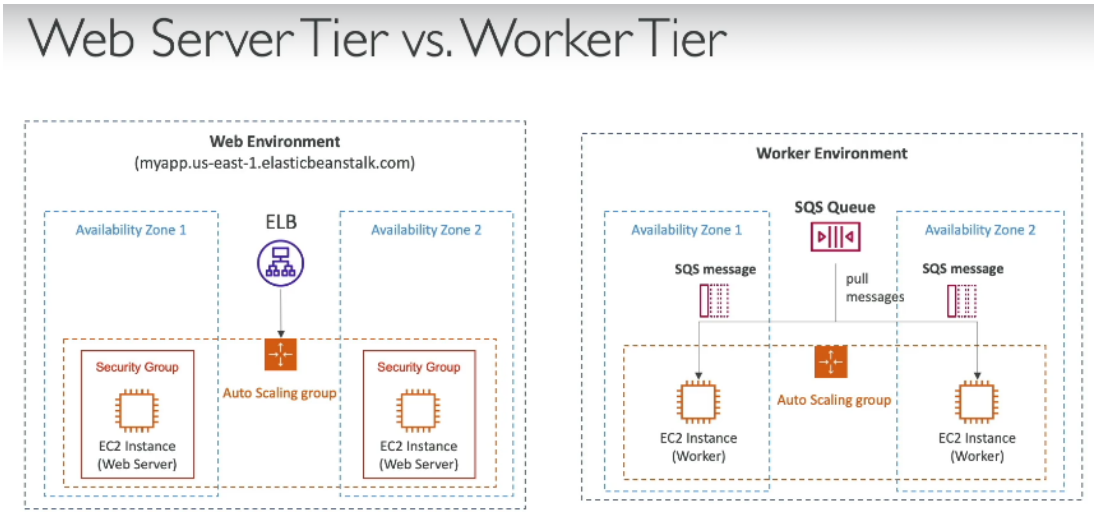
- **Single Instance deployment** - good for development.
- **LB + ASG** - great for production or pre-production web apps.
- **ASG only** - great for non-web apps in production.

Health Monitoring - a crucial feature that helps ensure the reliability and availability of our applications. It involves tracking the status of the environment's resources and application instances, detecting issues, and taking appropriate actions to maintain the desired state of the environment. It pushes metrics to CloudWatch.

Elastic Beanstalk Components

- Application - collection of Elastic Beanstalk components (environments, versions, configurations, ...).
- Application Version - an iteration of our application code.
- Environment - collection of AWS resources running an application version (only one application version at a time).





▼ ☑ **AWS Amplify.**

AWS Amplify is a set of tools and services designed to help developers build scalable and full-stack applications on AWS. It provides a straightforward and comprehensive way to develop web and mobile applications, leveraging the power of AWS cloud services.

▼ ☑ **Amazon SES.**

SES (Simple Email Service) is a fully managed service used to send email securely, globally and at scale. It allows inbound and outbound emails. We can send emails using our application, AWS Console, APIs or SMTP.

We have access to reputation dashboard, performance insights and anti-spam feedback. It provides statistics such as email deliveries, bounces, feedback loop results, email open, etc...

It supports DomainKeys Identified Mail (DKIM) and Sender Policy Framework (SPF).

Use cases: transactional, marketing and bulk email communications.

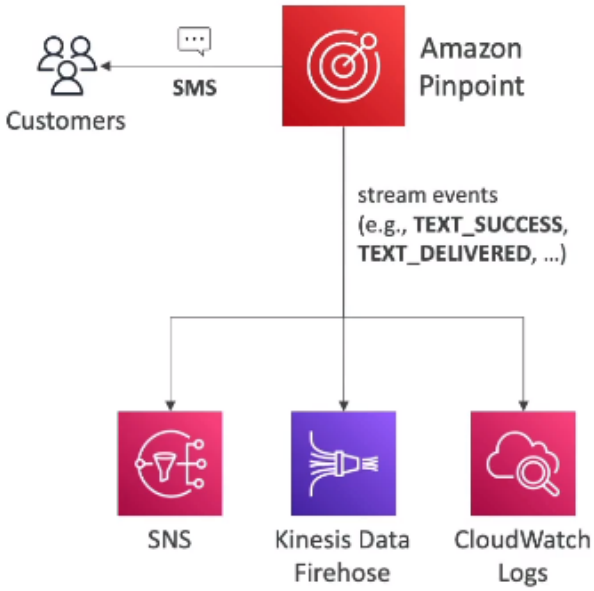
▼ ☑ **Amazon Pinpoint.**

Pinpoint is a scalable 2-way (outbound/inbound) marketing communications service. It supports email, SMS, push, voice and in-app messaging and has ability to segment and personalize messages with the right content to customers. Also we have possibility to receive replies.

Use cases: run campaigns by sending marketing, bulk, transactional SMS messages.

Pinpoint vs SNS & SES

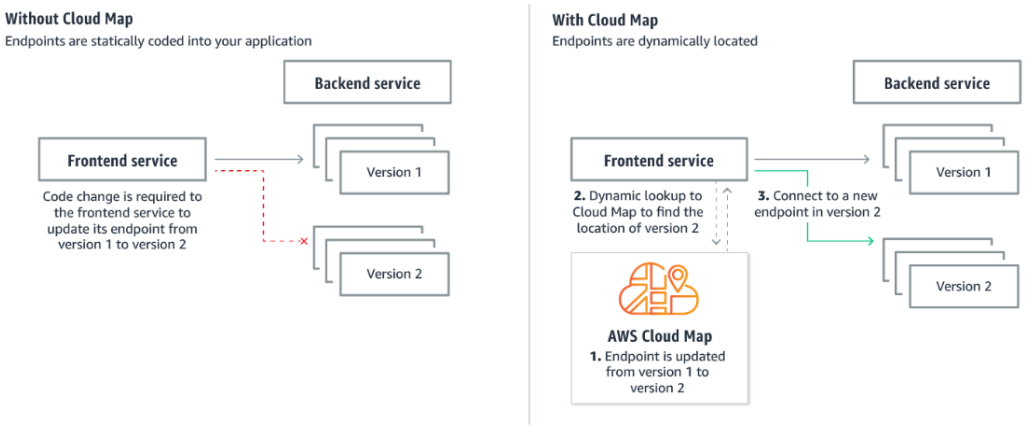
- In SNS & SES we manage each message's audience, content and delivery schedule.
- SQS focuses on reliable message queuing and decoupling applications.
- SES focuses on reliable email delivery and managing email reputation.
- In Pinpoint we create message templates, delivery schedules, highly-targeted segments and full campaigns. Focuses on customer engagement and analytics.



▼ ☑ **AWS Cloud Map.**

Cloud Map is a service that allows us to easily create and manage custom names for our application resources. It simplifies the process of service discovery in cloud-based applications by maintaining a dynamic directory of all our services and their locations.

It constantly monitors the health of every IP-based component of our application and dynamically updates the location of each microservice as it is added or removed.



▼ ☑ **AWS Cost Anomaly Detection.**

Cost Anomaly Detection continously monitor our cost and usage using ML to detect unsual spends. It learns our unique, historic spend patterns to detect one-time cost spike and/or continous cost increases (we dont need to define tresholds).

We can monitor AWS services, member accounts, cost allocation tags or cost categories. It will send us anomaly detection report with root-cause analysis and we can get notified with alerts using SNS.

▼ ☑ **AWS Cost Explorer.**

Cost Explorer - allows users to visualize, understand, and manage their AWS costs and usage over time. We can create custom reports that analyze cost and usage data on high level (total cost and usage across all accounts).

With Cost Explorer we are able to access our optimal savings plan. Also we can forecast usage up to 12 months based on previous usage.

▼ ☑ **AWS Trusted Advisor.**

Trusted Advisor is a service that helps customers optimize their AWS environments, improve performance, and save money. It provides real-time guidance to help users follow AWS best practices.

It lets us analyze our AWS accounts and provides recommendation on six categories:

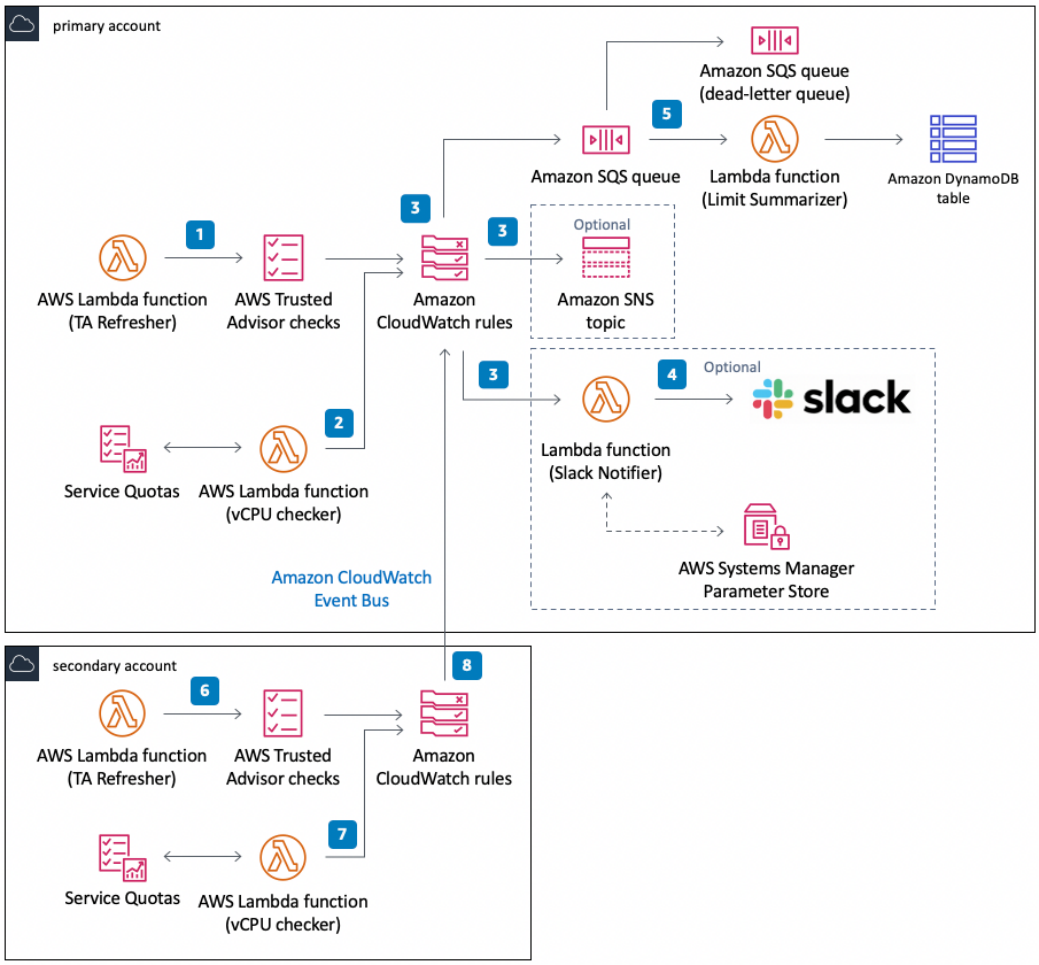
- Cost optimization.
- Performance.
- Security.
- Fault tolerance.
- Service limits.
- Operational Excellence.

For Business & Enterprise Support plan:

- Full Set of Checks.
- Programmatic Access using AWS Support API.

▼ **Solution Architecture - Quota Monitor.**

Pattern uses an AWS Lambda function that runs once every 24 hours. The Lambda function refreshes the AWS Trusted Advisor Service Limits checks to retrieve the most current utilization and quota data through API calls. Amazon CloudWatch Events captures the status events from Trusted Advisor. It uses a set of CloudWatch Events rules to send the status events to all the targets we choose during initial deployment of the solution: SQS queue, SNS topic or a Lambda function for Slack notifications.



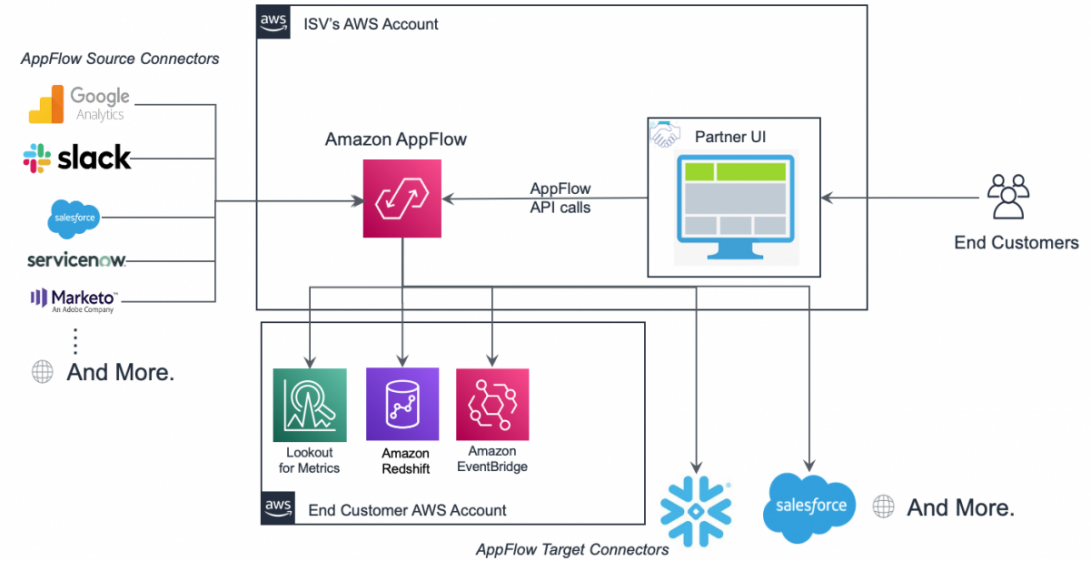
▼ ☑ **Amazon AppFlow.**

AppFlow is a fully managed integration service that enables us to securely transfer data between SaaS applications and AWS. Data is encrypted over the public internet or privately over AWS PrivateLink.

Sources: Salesfroce, SAP, Zendesk, Slack, ServiceNow...

Destinations: AWS Services (S3, Redshift) or non-AWS (SnowFlake, Salesforce).

We can define integration to run on a schedule, in response to events or on demand.



We dont spend time writing the integrations, we can leverage APIs immediately.

WhitePapers & Architectures

▼ ☑ **AWS Well Architected Framework.**

Well Architected Framework is a set of best practices designed to help cloud architects build secure, high-performing, resilient, and efficient infrastructure for their applications.

AWS Cloud Best Practices - Design Principles

- **Scalability** - vertical & horizontal.
- **Disposable Resources** - servers should be disposable & easily configured.
- **Automation** - serverless, IaaS, auto scaling ...
- **Loose Coupling**
- **Services, not Servers** - dont use just EC2, use managed services, databases, etc.

Well Architected Framework **Pillars** - key areas of focus that AWS recommends when designing and evaluating architectures on AWS.

▼ ☑ **WhitePapers Links.**

1. Architecting for the cloud:

https://d1.awsstatic.com/whitepapers/AWS_Cloud_Best_Practices.pdf (Archived)

2. WhitePapers related to well-architected framework:

https://aws.amazon.com/blogs/aws/aws-well-architected-framework-updated-white-papers-tools-and-best-practices/

3. Disaster recovery whitepaper:

https://d1.awsstatic.com/whitepapers/aws-disaster-recovery.pdf

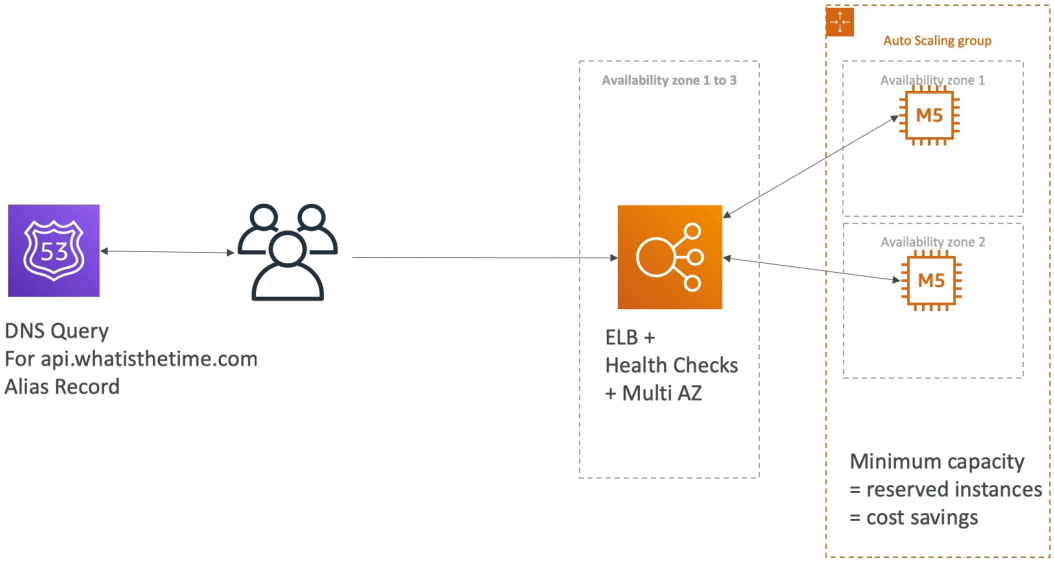
Well-Architected framework whitepaper:

https://d1.awsstatic.com/whitepapers/architecture/AWS_Well-Architected_Framework.pdf

▼ **1. WhatsTheTime App**

Requirements:

- It allows people to know what time it is.
- We dont need a database.
- Start small and can accept downtime.
- We want to fully scale vertically and horizontally, no donwtime.



▼ **2. Clothes Store App.**

Requirements:

- It allows people to buy clothes online.
- There is a shopping cart and users should not lose their shopping cart.
- Website can handle hundreds of users at the same time.
- We need to scale, maintain horizontal scalability and keep our web application as stateless as possible.
- Users should have their details in a database.

Architecture

We can use basic architecture with Route 53, ELB (Multi-AZ) and ASG (3 AZ).

Session

We could use sticky session to save shoping cart but it will violate EC2 performance. Using web cookies is better but we have security risks. The best way to save session data is to store it in ElastiCache.

Database

We can store user data in RDS. Because there are hundreds of users we can use RDS Read Replicas to scale reads (there is alternative using Lazy Loading

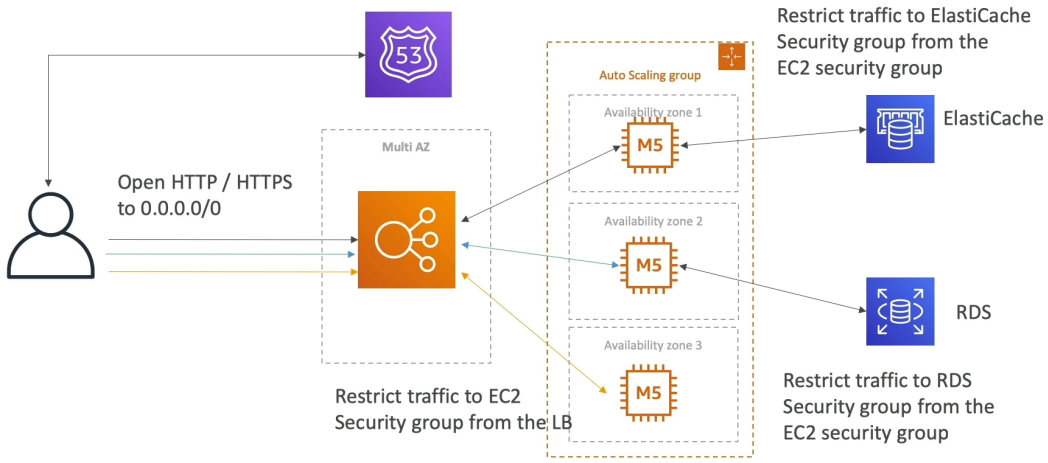
pattern with ElastiCache).

Availability

Using Multi-AZ RDS deployment can be used for disaster recovery.

Security

Open HTTP/HTTPS to 0.0.0.0/0 for users. We need to restrict traffic to EC2 security group from the load balancer and restrict traffic to ElastiCache and RDS security groups from the EC2 security group.

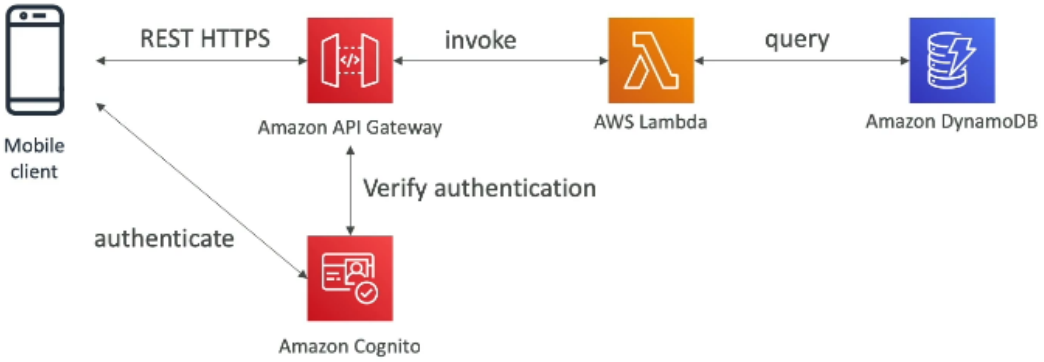


▼ 3. MyTodoList Mobile App.

Requirements:

- Expose as REST API with HTTPS.
- Serverless architecture.
- Users should be able to directly interact with their own folder in S3.
- Users should authenticate through a managed serverless service.
- Users can write and read to-dos, but they mostly read them.
- The database should scale and have some high read throughput.

REST API Layer

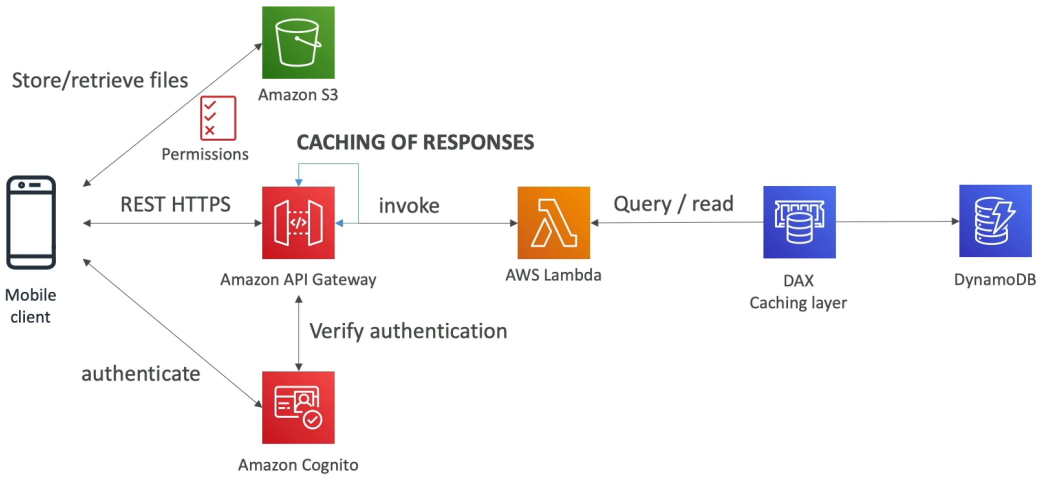


Giving Users Access to S3

We will use Cognito to generate us temporary credentials so we can use S3 to store and retrieve files.

High Read Throughput

We can use DAX as a caching layer, so reads will be cached and DynamoDB will not need many RCU. Another alternative is to cache responses at API Gateway level.



▼ 4. MyBlog Serverless Website.

Requirements:

- Website should scale globally.
- Blogs are rarely written, but often read.

- Some of the website is purely static files, the rest is a dynamic REST API.
- Caching must be implement where possible.
- Any new users that subscribes should receive a welcome email.
- Any photo uploaded to the blog should have a thumbnail generated.

Serving static content, globally and securely

We can store our static content in S3 and we can use CloudFront to serve it globally. To secure it we will add Origin Access Control which will ensure that our S3 bucket can only be accessed by CloudFront. For this we will add a bucket policy to only auhorize the CloudFront distribution.

Serverless REST API

We will have REST HTTPS which will go through API Gateway, invoking a Lambda function and query/read from DynamoDB (can use DAX for cache). If we go global we can use DynamoDB Global Databases.

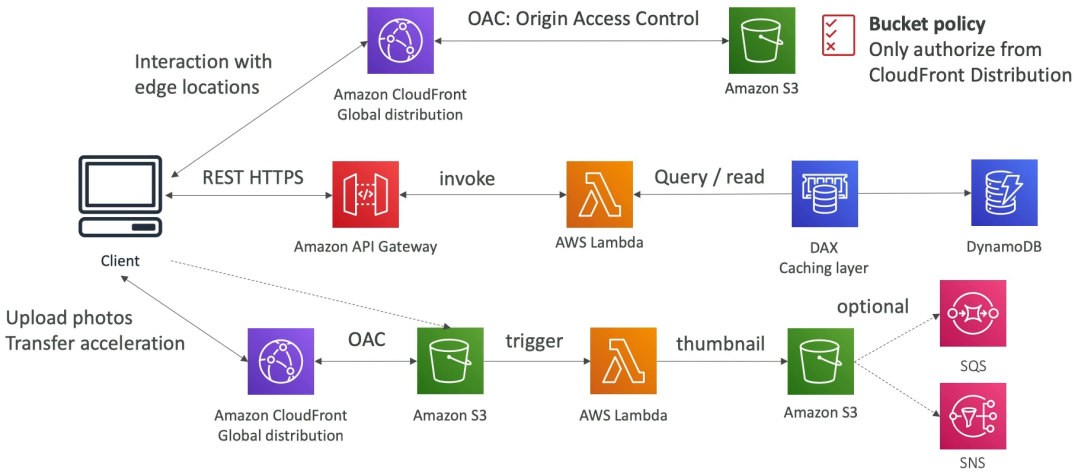
Welcome Email

For user welcome email we can use DynamoDB Stream which will invoke a Lambda function (need IAM role) which will trigger SES to send an email.

Thumbnail Generation

When image is uploaded we need to store it in S3 bucket (can use again OAC and CloudFront - Transfer Acceleration). Uploading to S3 will trigger Lambda

which will create thumbnail and store it in another or same S3 bucket.

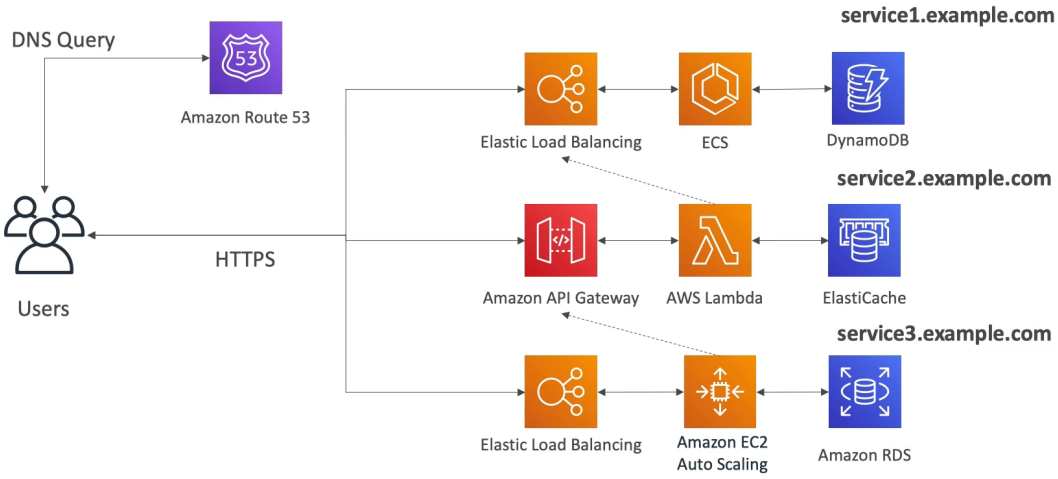


▼ 5. Microservices Architecture.

We are free to design each microservice the way we want.

Synchronous patterns: API Gateway, Load Balancers.

Asynchronous patterns: SQS, Kinesis, SNS, Lambda.



Challenges with microservices:

- Repeated overhead for creating each new microservice.
- Issues with optimizing server density/utilization.

- Complexity of running multiple versions of multiple microservices simultaneously.

Challenges solved by serverless patterns:

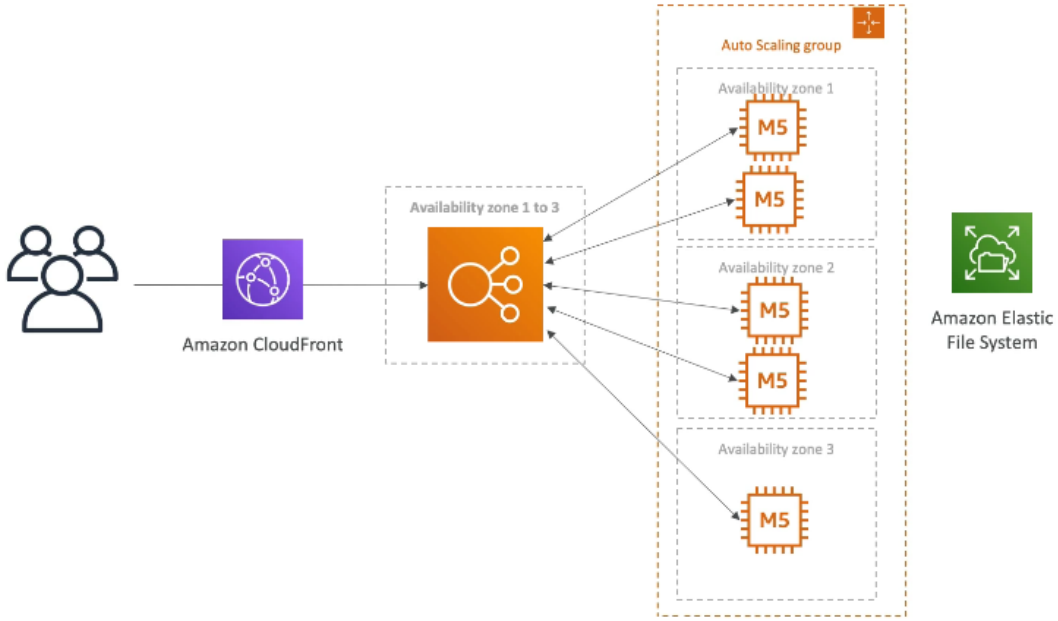
- API Gateway, Lambda scale automatically and we can pay per usage.
- We can easily clone API and reproduce enviornments.

▼ 6. Software Updates Distribution.

We have an application running on EC2 that distributes software updates once in a while. When a new software update is out, we get a lot of requests and the content is distributed in mass over the network and it's very costly.

We can use CloudFront. It will cache software update files at the edge and no changes are needed to the architecture. Software update files are not dynamic, they are static (never changing).

EC2 isntances arent serverless, but CloudFront is and it will scale for us. We will save in availability, network bandwidth cost, etc.



▼ 7. Big Data Ingestion Pipeline.

Requirements:

- We want the ingestion pipeline to be fully serverless.

- We want to collect data in real time and transform that data.
- We want to query the transformed data using SQL.
- The reports created using the queris should be in S3.
- Reported data should be loaded into a warehouse and we want to create dashboards.

Streaming

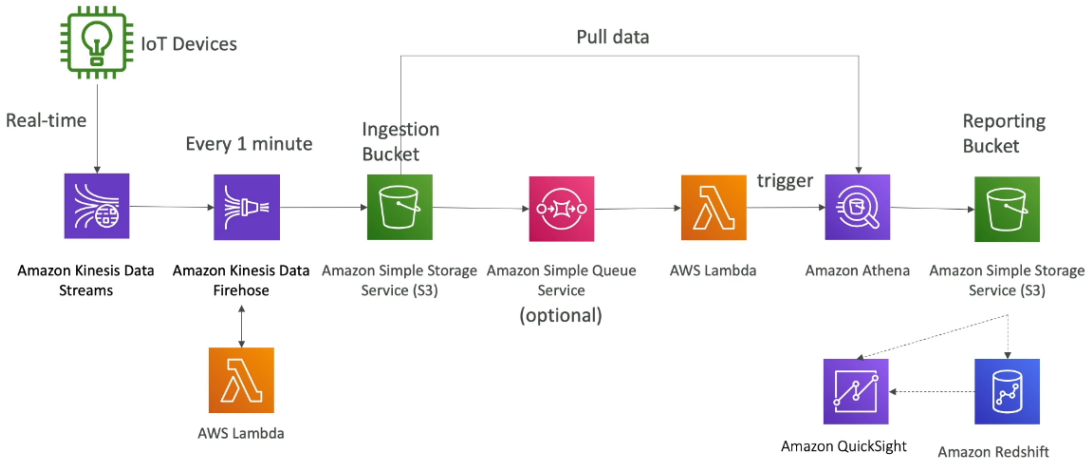
IoT devices send real-time data to Kinesis Data Streams. Data is periodically sent to Kinesis Data Firehose, which can trigger AWS Lambda for processing and then store it in S3 (Ingestion Bucket).

Processing

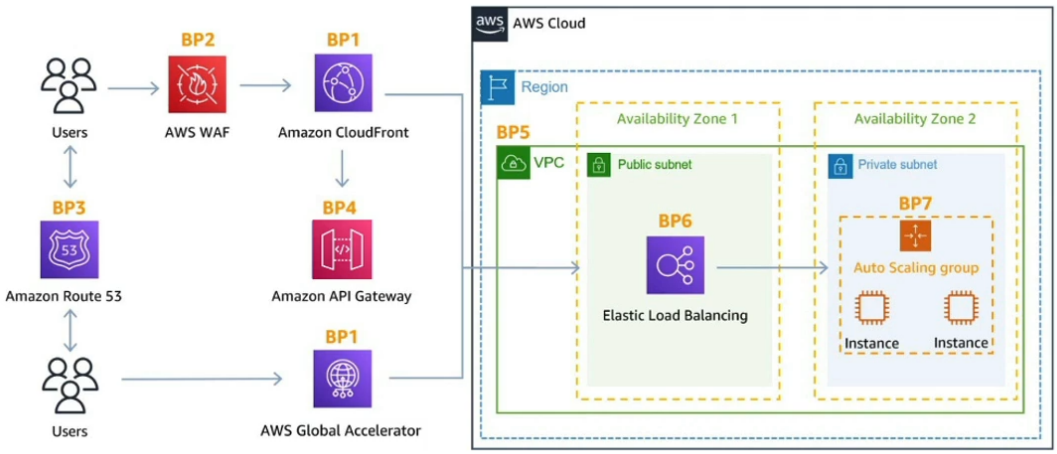
The data may go through SQS before further processing by another Lambda function. The processed data is analyzed using Amazon Athena, with results stored in the Reporting Bucket.

Analytics

The data in the Reporting Bucket or Redshift can be visualized using QuickSight or further analyzed in Redshift.



▼ 8. DDoS Protection Best Practices.



Edge Location Mitigation

BP1 - CloudFront ⇒ Web application delivery at the edge and we are protected from DDoS common attacks (SYN floods, UDP reflection...).

BP1 - Global Accelerator ⇒ Access our application from the edge. We can integrate it with Shield for DDoS protection.

BP3 - Route 53 ⇒ It has DDoS protection mechanism.

Best Practices for DDoS Mitigation

Infrastructure layer defense (BP1, BP3, BP6) ⇒ Protects EC2 against high traffic.

EC2 with ASG (BP7) ⇒ Helps in case of sudden traffic surges including a flash crowd or DDoS attack.

ELB (BP6) ⇒ Scales with the traffic increase and will distribute the traffic to many EC2 instances.

Application Layer Defense

Detect & filter malicious web requests (BP1, BP2) ⇒ CloudFront cache static content and serve it from the edge locations, protecting our backend. WAF is used on top of CloudFront and ALB to filter and block requests based on request signatures.

Shield Advanced (BP1, BP2, BP6) ⇒ Shield Advanced automatically creates, evaluates and deploys WAF rules to mitigate layer 7 attacks.

Attack Surface Reduction

Obfuscating AWS resources (BP1, BP4, BP6) ⇒ We can hide our backend resources (Lambda functions, EC2 instances)

Security groups and Network ACLs (BP5) ⇒ We can filter traffic based on specific IP at the subnet or ENI level. Elastic IP are protected by Shield Advanced.

Protecting API endpoints (BP4) ⇒ Hide our backend. If we use edge-optimized mode or CloudFront + regional mode we can get more control for DDoS protection. Also if we add WAF on top of it then we can get filtering of any HTTP request.

Other

▼ **Important Ports.**

Service	Port
FTP	21
SSH	22
SFTP	22
HTTP	80
HTTPS	443
PostgreSQL	5432
MySQL	3306
Oracle RDS	1521
MSSQL Server	1433
Remote Desktop	3389

▼ **Instantiating Applications Quickly.**

EC2 Instances

- Golden AMI - install our applications, OS dependecies, etc... beforehand and launch our EC2 instance from the Golden AMI.
- Bootstrap using User Data - for dynamic configuration.
- Hybrid - mix Golden AMI and User Data (Elastic Beanstalk).

RDS

- Restore from a snapshot - the database will have schemas and data ready.

EBS

- Restore from a snapshot - the disk will already be formatted and have data.