Best Practices for Designing Amazon API Gateway Private APIs and Private Integration

January 2021

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Contents

Abstract

For many enterprise customers, AWS Direct Connect or a virtual private network (VPN) is often used to build a network connection between an on-premises network and an Amazon Web Services (AWS) virtual private cloud (VPC). This adds additional complexity to a network design, and introduces challenges to Amazon API Gateway private API and private integration setup. This whitepaper introduces best practices for deploying private APIs and private integrations in API Gateway, and discusses security, usability, and architecture.

This whitepaper is aimed at developers who use API Gateway, or are considering using it in the future.

Introduction

Many customers use Amazon API Gateway to build RESTful and HTTP APIs. If the use of those APIs is limited to internal clients, customers prefer to use private APIs, because private APIs provide a secure means to invoke APIs via an interface VPC endpoint. API Gateway private integration makes it simple to expose your HTTP/HTTPS resources behind an Amazon VPC, for access by clients outside of the VPC. Additionally, private integration can integrate with private APIs, so the APIs can send requests to a Network Load Balancer (NLB) through a private link. For HTTP APIs, Application Load Balancer (ALB) and AWS Cloud Map are also supported. Private integration forwards external traffic sent to APIs to private resources, without exposing the APIs to the internet.

Based on security requirements, different security measures can be placed at different security layers. To secure VPC resources such as Elastic Network Interface (ENI), associate resources are associated with a security group. VPC endpoints are associated with both the security group and the resource policy. For NLB, Transport Secure Layer (TLS) listeners are used to secure a listener. For ALB, security groups and HTTPS listeners are used.

Compared to regional and edge-optimized API implementation, private API implementation and private integrations add additional components, such as interface VPC endpoints and load balancers. This can lead to additional complexity in application architectures.

This whitepaper includes sample architectures to help understand private APIs, along with private integration implementation and best practices. It also covers security and cost optimizations.

Overview of Amazon API Gateway

Amazon API Gateway is a fully managed service that helps you easily create, publish, maintain, monitor, and secure APIs at any scale. It provides three different types of APIs: *REST*, *WebSocket*, and *HTTP*. Depending on your business needs and architectural patterns, you can use one or more of the API types:

- The **REST** API type has three endpoint types: edge-optimized, regional, and private. Edge-optimized and regional REST APIs are publicly accessible and serve requests over the internet. For customers who need to access an API in a private network, a private REST API is the preferred choice. REST APIs provide an easy means to secure APIs such as resource policies, IAM authentication, and custom authorizers.
- **WebSocket** APIs enable you to build real-time, two-way communication applications such as chat apps and streaming dashboards. Although there is no private endpoint type available, WebSocket APIs provide an option to create a route with a VPC link for private integration.
- **HTTP** APIs are the newest type of APIs in API Gateway. They include enhanced features such as auto deployment and cross-origin resource sharing (CORS) support, improved performance, and low costs. HTTP API private integrations work with Application Load Balancer and AWS Cloud Map, in addition to Network Load Balancer.

Rest API

REST APIs help create APIs that follow the REST architectural style. Developers can use their existing knowledge and apply best practices while building REST APIs in API Gateway.

While designing a REST API, a key consideration is security. Use least privilege access when giving access to APIs. The private endpoint type restricts API access through interface VPC endpoints only. If REST APIs are publicly exposed but integration endpoints exist in a private subnet, private integration offers a way to access the endpoints via a VPC link. You can create a VPC link with a Network Load Balancer. API Gateway creates a VPC endpoint service for API Gateway to access Network Load Balancer.

Private Endpoint Type

To make APIs accessible only from Amazon VPCs, you can use REST APIs with the private endpoint type. The traffic to the APIs will not leave the AWS network. There are three options to invoke a private API through different domain name system (DNS) names:

- Private DNS names

- Interface VPC endpoint public DNS hostnames
- Amazon Route53 alias

While configuring private APIs, there are several key points to consider. The "DNS Names for Private APIs" section provides use cases, pros, and cons about each option.

DNS Names for Private APIs

Table 1 – Private API DNS names

DNS names	Private DNS option on VPCe	Pros	Cons
Private DNS names	Enabled	Easy to set up	DNS issue with regional and edge-optimized APIs.
Interface VPC endpoint public DNS hostnames	Disabled	The domain name is publicly resolvable.	Requires a Host or x-apigw-api-id header in requests.
Route53 alias	Disabled	The domain name is publicly resolvable. The host or `x-apigw-api-id` header is not required.	Requires an interface VPC endpoint association with each private API.

Private DNS Names

This option works when the private DNS option on an interface VPC endpoint is enabled. In addition, to resolve the name, AmazonProvidedDNS should be present in the DHCP options set for the clients in the VPC. Because those are the only requirements, this option is usually easy to use for a simple use case such as invoking a private API within a VPC.

However, if you use a custom DNS server, a conditional forwarder must be set on the DNS that points to the `AmazonProvidedDNS` or Route53 Resolver. Because of the private DNS option enabled on the interface VPC endpoint, DNS queries against

`*.execute-api.amazonaws.com` will be resolved to private IPs of the endpoint. This causes issues when clients in the VPC try to invoke regional or edge-optimized APIs, because those types of APIs must be accessed over the internet. Traffic through interface VPC endpoints is not allowed. The only workaround is to use an edge-optimized custom domain name. See Why do I get an HTTP 403 Forbidden error when connecting to my API Gateway APIs from a VPC? for the troubleshooting steps.

VPC Endpoint Public DNS Hostnames

If your use case requires the private DNS option to be disabled, consider using interface VPC endpoint public DNS hostnames. When you create an interface VPC endpoint, it also generates the public DNS hostname. When invoking a private API through the hostname, you must pass a Host or `x-apigw-api-id` header.

The header requirement can cause issues when the hostname is used in a web application. For cross-origin, non-simple requests, modern browsers send a preflight request to an endpoint. This option requires clients to send requests with a custom header. Because browsers will not send the custom header for the preflight request, this will cause CORS issues. This option is **not** a preferred option for customers who need to use a private API from a web application.

Route 53 Alias

This Route 53 option resolves the header requirement imposed by the VPC endpoint public DNS hostnames option. Additionally, the Route 53 alias is publicly resolvable, and does not require private DNS to be enabled. Clients in a VPC can access private APIs through the Route 53 alias, as well as other types of APIs such as regional and edge-optimized REST APIs.

Each alias is generated after associating a VPC endpoint to a private API. The association is required every time you create new interface VPC endpoints and private APIs.

Resource-Based Policy

Resource-based policies are attached to a resource like a REST API in API Gateway. For resource-based policies, you can specify who has access to the resource and what actions are permitted.

Unlike regional and edge-optimized endpoint types, private APIs require the use of a resource policy. Deployments without a resource policy **will fail**. For private APIs, there are additional keys within the condition block you can use in the resource policy, such as `aws:sourceVpc` and `aws:SourceVpce`. The `aws:sourceVpc` policy allows traffic to originate from specific VPCs, and `aws:SourceVpce` allows traffic originating from interface VPC endpoints.

Private Integration

Private integrations allow routing traffic from API Gateway to customers' VPCs. The integrations are based on VPC links, and rely on a VPC endpoint service that is tied to network load balancers (NLBs) for REST and WebSocket APIs. VPC link integrations work in a similar way as HTTP integrations. A common use case is to invoke Amazon EC2-hosted applications behind NLBs through VPC links. There are several design considerations in this case:

- For existing applications with a Classic Load Balancer (CLB) or Application Load Balancer (ALB:
  - Create an NLB in front of a CLB or ALB.
    - This creates an additional network hop and infrastructure cost.
  - Route traffic through NLB instead of CLB or ALB.
    - This requires migration from CLB or ALB to NLB to shift traffic and redesign the existing architecture. See Migrate your Classic Load Balancer for the migration process.
- NLB listener type
  - Transmission control protocol (TCP) (Secure Socket Layer (SSL) passthrough or non-SSL traffic)
  - Transport Layer Security (TLS) (terminating the SSL connection on NLB)

Sample Architecture Patterns

When implementing a private API, using an authorizer such as AWS Identity and Access Management (IAM) or Amazon Cognito is highly recommended. This ensures an additional layer of security, and helps verify requests using IAM credentials for IAM authorization, and access/ID tokens for the Amazon Cogito authorizer.

Basic Architecture

In the basic architecture, Amazon EC2 instances and VPC-enabled AWS Lambda functions access a private API through an interface VPC endpoint. The security group attached to the endpoint must allow the Transmission Control Protocol (TCP) port 443. In the private API resource policy, requests from the VPC and interface VPC endpoint should be allowed.

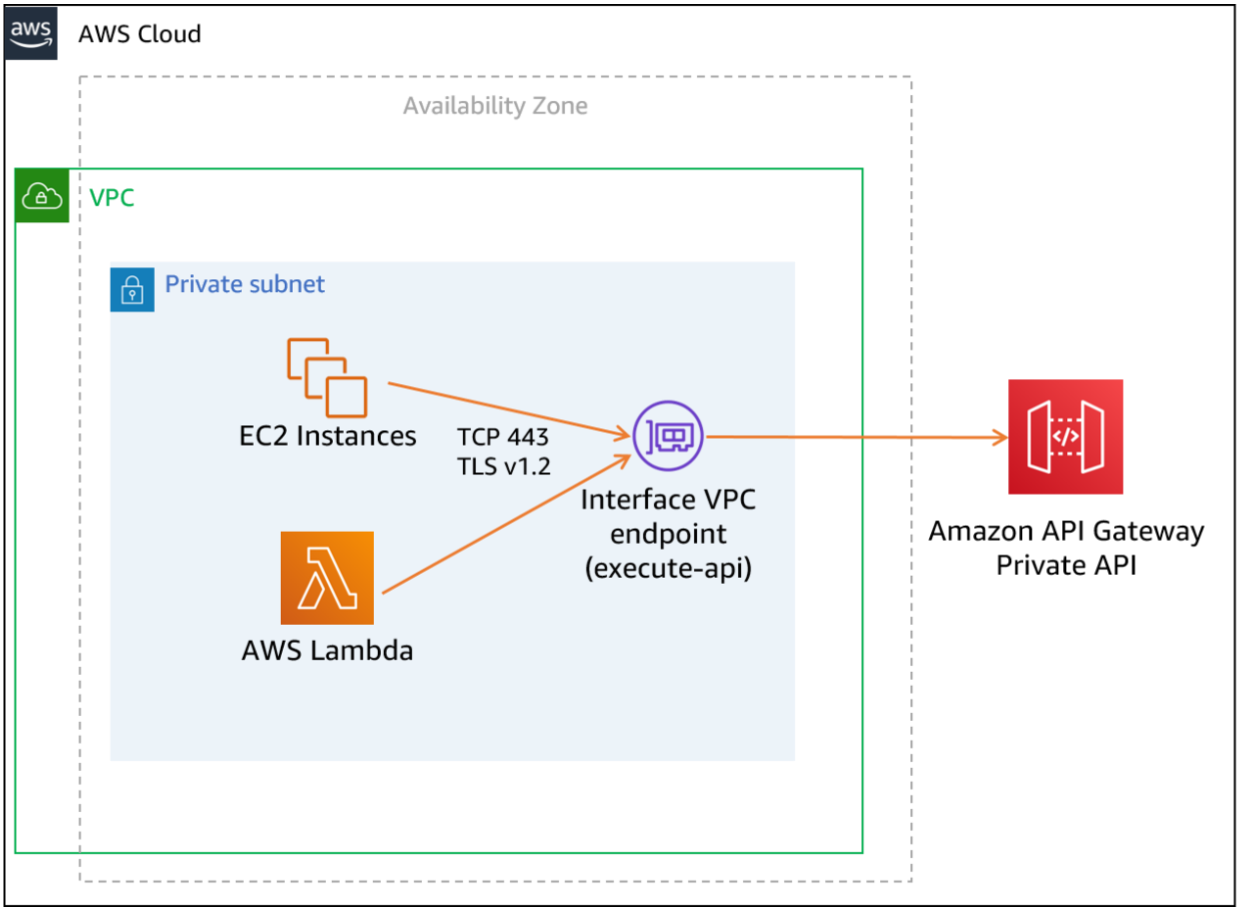


Figure 1 – REST private API basic architecture

Cross-Account Architecture

If you want to allow access to a private API from other accounts, an interface VPC endpoint in a different account can be used to invoke the API. However, they both must exist in the same Region, such as us-east-1 (N. Virginia). Additionally, the private API resource policy must allow access from the other account’s VPC or interface VPC endpoint.

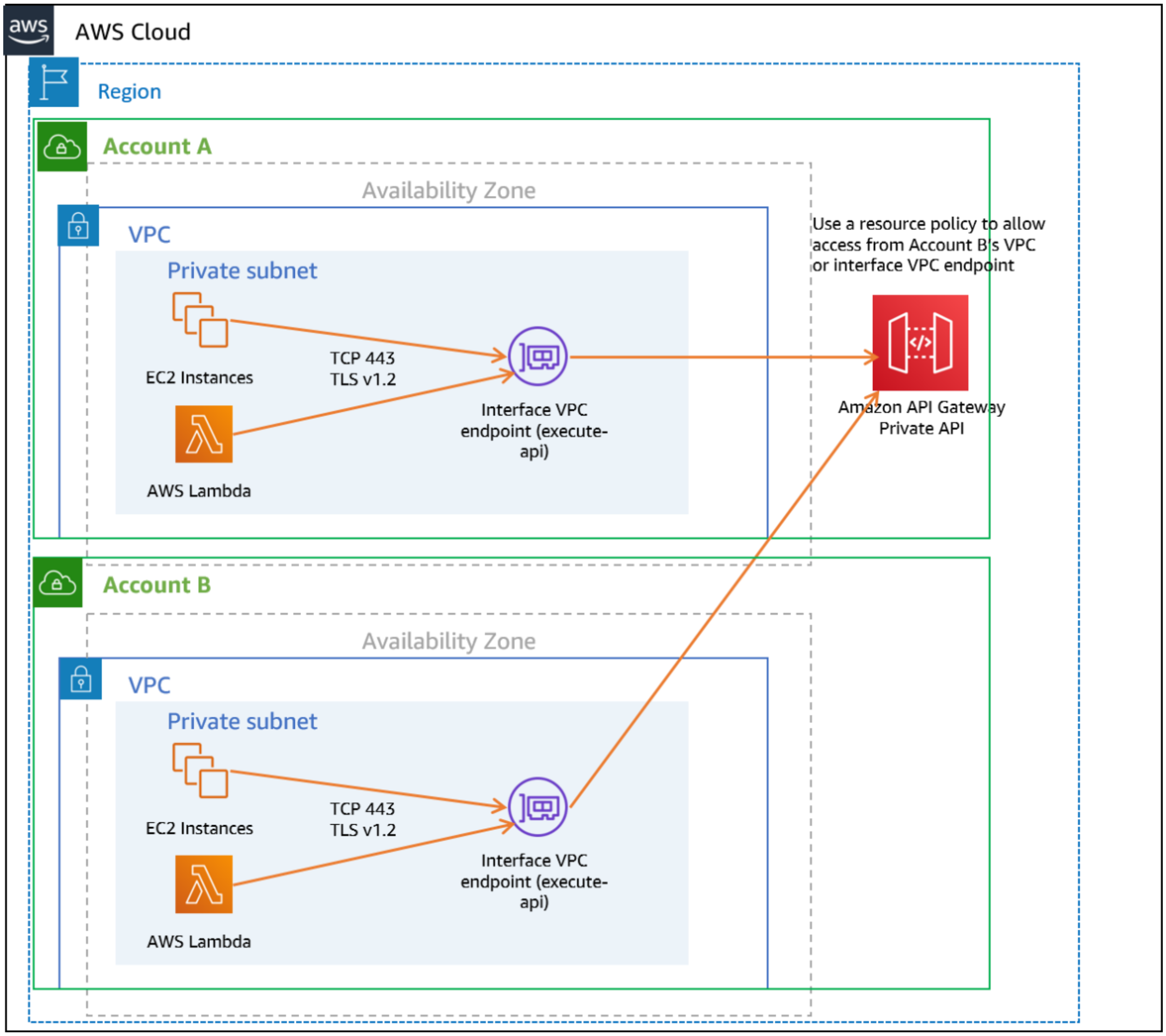
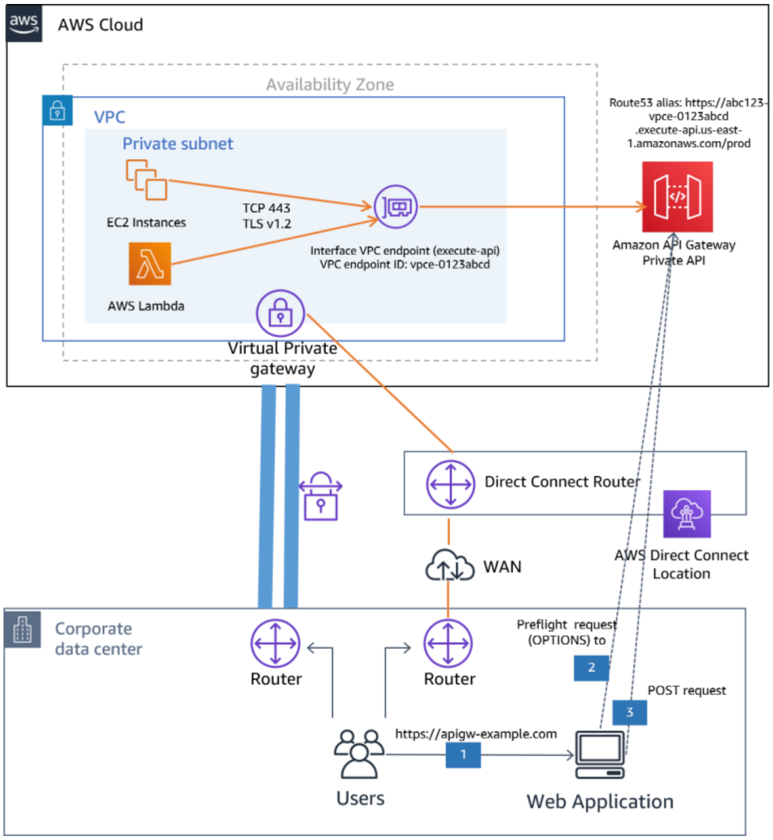

AWS Cloud
Region
Account A
Availability Zone
VPC
Private subnet
EC2 Instances
TCP 443
TLS v1.2
AWS Lambda
Interface VPC endpoint (execute-api)
Use a resource policy to allow access from Account B's VPC or interface VPC endpoint
Amazon API Gateway Private API
Account B
Availability Zone
VPC
Private subnet
EC2 Instances
TCP 443
TLS v1.2
AWS Lambda
Interface VPC endpoint (execute-api)

Figure 2 – REST private API cross-account architecture

On-Premises Architecture

If you have users accessing from on-premises locations, you will need a Direct Connect or VPN connection between the on-premises networks and your VPC. All requests must still go through interface VPC endpoints. For the on-premises architecture, VPC endpoint public DNS hostnames or Route 53 alias records are good options when invoking private APIs. If on-premises users access the network through a web application, Route 53 alias records are a better approach to avoid CORS issues. If the Route 53 alias record option does not work, one solution is to create a conditional forwarder on an on-premises DNS pointing to a Route 53 resolver. See Resolving DNS queries between VPCs and your network.

The following diagram shows a sample architecture where on-premises clients access a web application hosted in the on-premises network. The web application uses a private API for its API endpoint. For the private API endpoint, a Route 53 alias is used. Because a Route 53 alias record is publicly resolvable, there is no need to set up a conditional forwarder on on-premises DNS servers to resolve the hostname.


AWS Cloud
Availability Zone
VPC
Private subnet
EC2 Instances
TCP 443
TLS v1.2
AWS Lambda
Interface VPC endpoint (execute-api)
VPC endpoint ID: vpce-0123abcd
Virtual Private gateway
Route53 alias: https://abc123-vpce-0123abcd .execute-api.us-east-1.amazonaws.com/prod
Amazon API Gateway
Private API
Direct Connect Router
AWS Direct Connect Location
WAN
Corporate data center
Router
Router
Preflight request (OPTIONS) to
2
POST request
3
https://apigw-example.com
1
Users
Web Application

Setup:

- Private API is associated with the VPC endpoint vpce-0123abcd.
    - This generates a Route53 alias to invoke a private API.
- On-premises network and a VPC are connected through Direct Connect

1. On-premises users access a web application hosted in the on-premises network.
2. For non-simple requests, a web browser makes a preflight request (OPTIONS) to the private API (https://abc123-vpce-0123abcd.execute-api.us-east-1.amazonaws.com/prod)
3. When the preflight response include appropriate CORS headers such as Access-Control-Allow-Origin: *, the web browser makes an HTTP request such as POST to the private API (https://abc123-vpce-0123abcd.execute-api.us-east-1.amazonaws.com/prod).

Figure 3 – REST private API on-premises architecture

Private Integration Architecture with ECS

Amazon Elastic Container Service (Amazon ECS) is a fully managed container orchestration service. Customers can use ECS to run their most sensitive and mission critical applications because of its security, reliability, and scalability. For private integration in REST APIs, one common design pattern is to use an NLB to route traffic to an Amazon ECS cluster in private subnets. Many customers deploy ECS as their backend compute service. The following diagram shows clients in one VPC accessing an ECS cluster in another VPC through a private API and private integration.

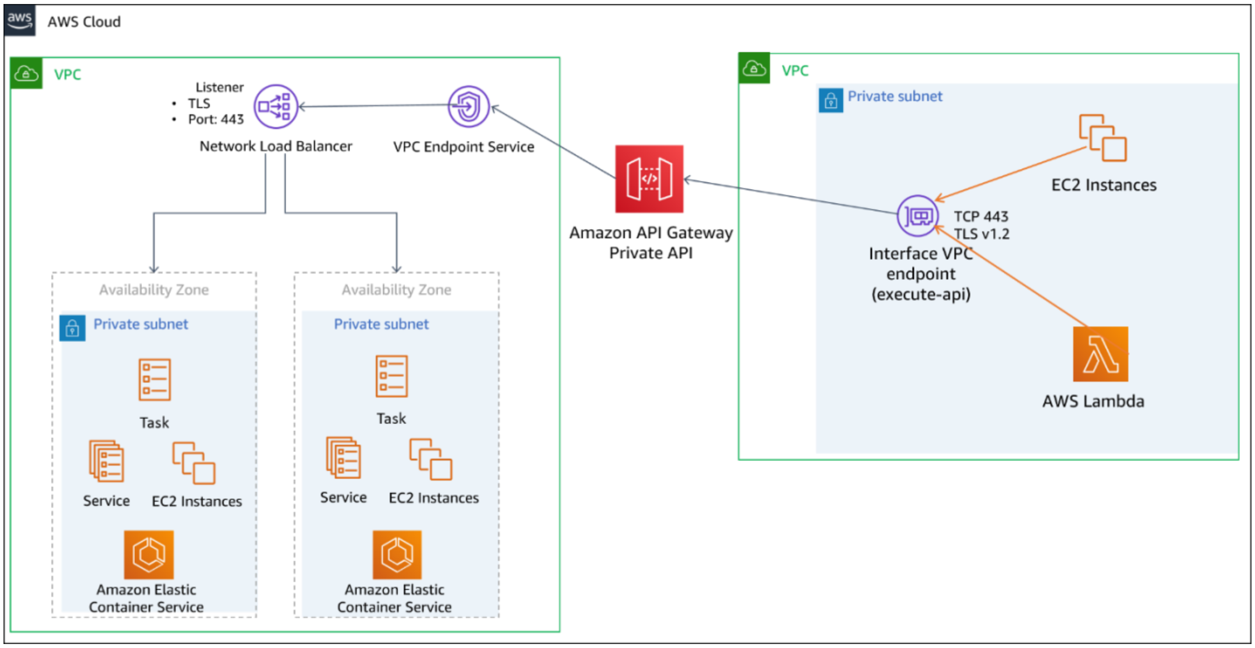


Figure 4 – Cross-VPC ECS Access via Private Integration with Private API

WebSocket API

WebSocket APIs offer APIs that the client can access through the WebSocket protocol. Unlike REST and HTTP APIs, WebSocket APIs allow bidirectional communications. WebSocket APIs are often used in real-time applications such as chat applications, collaboration platforms, multiplayer games, and financial trading platforms.

Private Integration

Private integrations with WebSocket APIs are very similar to those using REST APIs. The difference is how responses are handled, because integration responses are optional in WebSocket API routes. However, integration requests to the VPC links work the same way as requests to REST APIs, so the same design considerations apply to WebSocket APIs.

Sample Architecture Pattern

Currently, WebSocket APIs are offered only with a regional endpoint type. The APIs must be accessed over the internet. Using a private integration, requests through APIs can be routed to EC2 instances or VPC resources through an NLB privately. You can perform TLS termination on a TLS listener of the NLB, or pass the TLS traffic through to the target group instances. If the TLS termination happens on the target group instances, you can implement client certificates generated by API Gateway to enhance security. See [Generate and configure an SSL certificate for backend authentication](#).

Sample Architecture

Figure 5 shows a sample architecture where WebSocket API users access a route key mapped to a VPC link integration method. The NLB has a TLS listener for the domain “example.com”, and listens on TCP port 443. The target group for the listener points to ECS services.

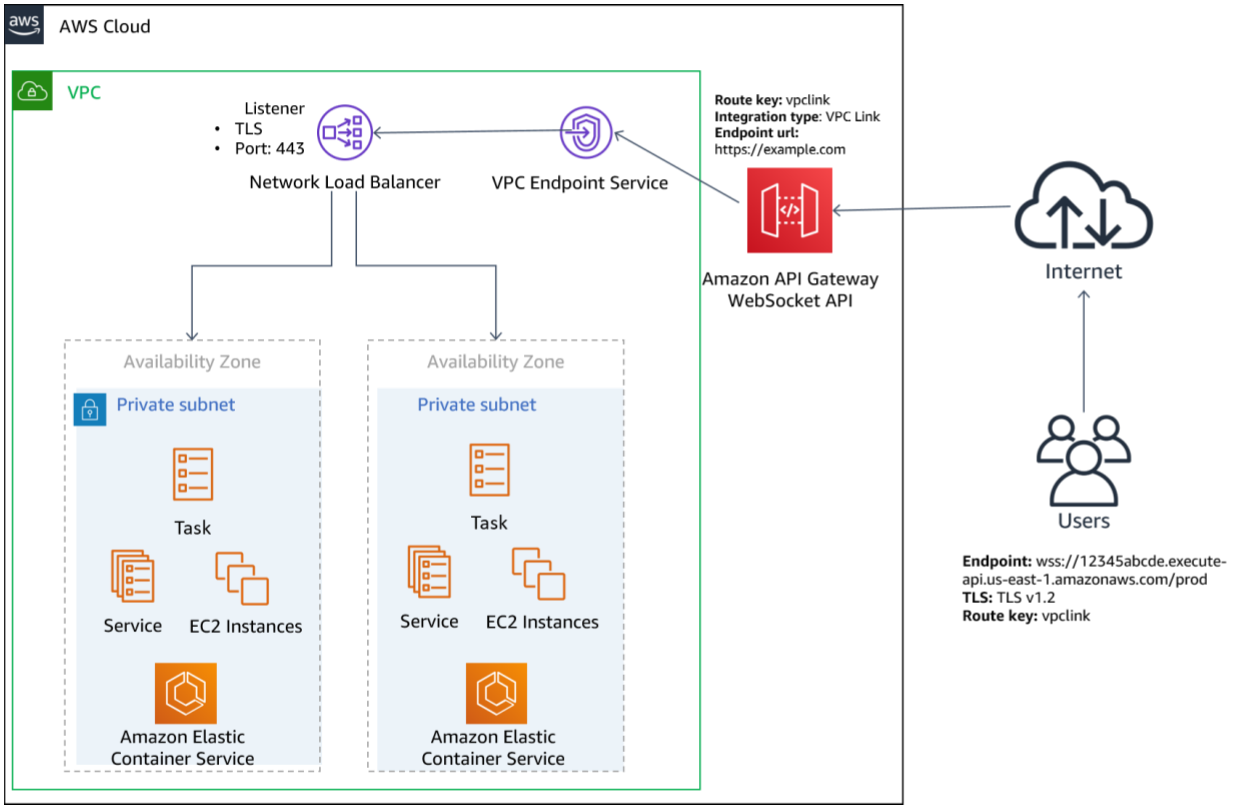


Figure 5 – WebSocket API private integration with ECS

HTTP API

HTTP API is a new flavor of API Gateway. Benefits of using the API include delivering enhanced features, improved performance, and an easier developer experience. In addition, HTTP APIs come with reduced request pricing.

For private integrations, HTTP APIs offer additional integration endpoints for a VPC link, such as ALBs, NLBs, and AWS Cloud Map. For any existing applications or micro services that have ALBs or AWS Cloud Map to route traffic, you can use the same setup. HTTP APIs can route traffic to those endpoints through a VPC link.

Private Integration

Because HTTP APIs offer three different private integration targets, you should consider which integration target best suits your use case. Depending on the backend service, one or more targets can be used by creating multiple VPC endpoints.

Table 2 – HTTP API Private Integration

Integration target	Listener	Use cases
NLB	TCP or TLS listener	TLS passthrough is possible High throughput
ALB	HTTP or HTTPS listener	Layer 7 routing Content-based routing
AWS Cloud Map	Namespace/Service AWS Cloud Map parameters (optional)	Service discovery

Sample Architecture Patterns

ALB Architecture (ECS)

HTTP API private integration allows NLB and ALB for integration targets for load balancers. If you have any backend service fronted with ALBs, you can use the existing setup without re-architecture. Because ALBs allow different routing options, such as path-based routing, this option provides flexibility on the ALB routing level. To create listener rules to achieve path-based routing, see Listener rules for your Application Load Balancer.

Figure 6 shows private integration with ALB in HTTP API. The ALB uses path-based routing rules to route traffic to two different ECS services.

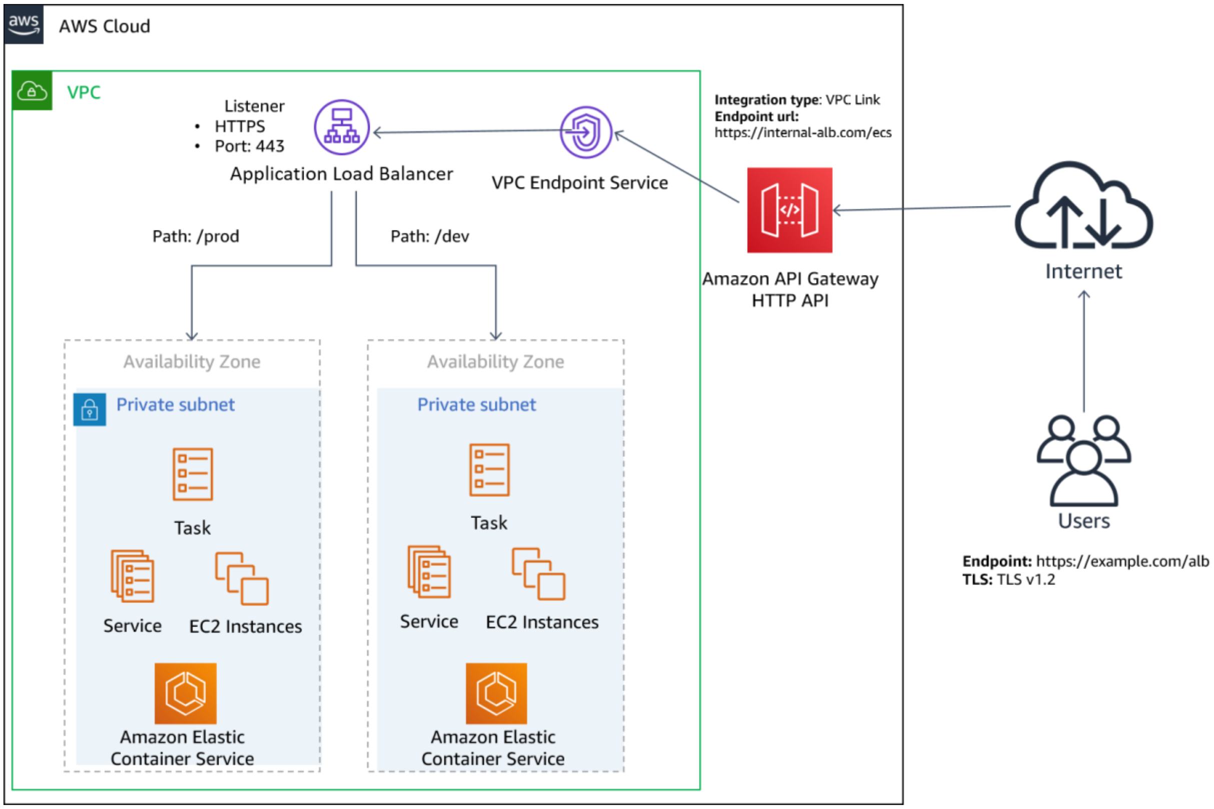


Figure 6 – HTTP API private integration with ALB

Cloud Map Architecture (Microservices)

With the AWS Cloud Map target option, you can use AWS Cloud Map to discover services like ECS and EC2-based services. Using AWS Cloud Map as a front-end service for microservices, you can leverage a private integration with an AWS Cloud Map target in HTTP APIs to route requests to different endpoints.

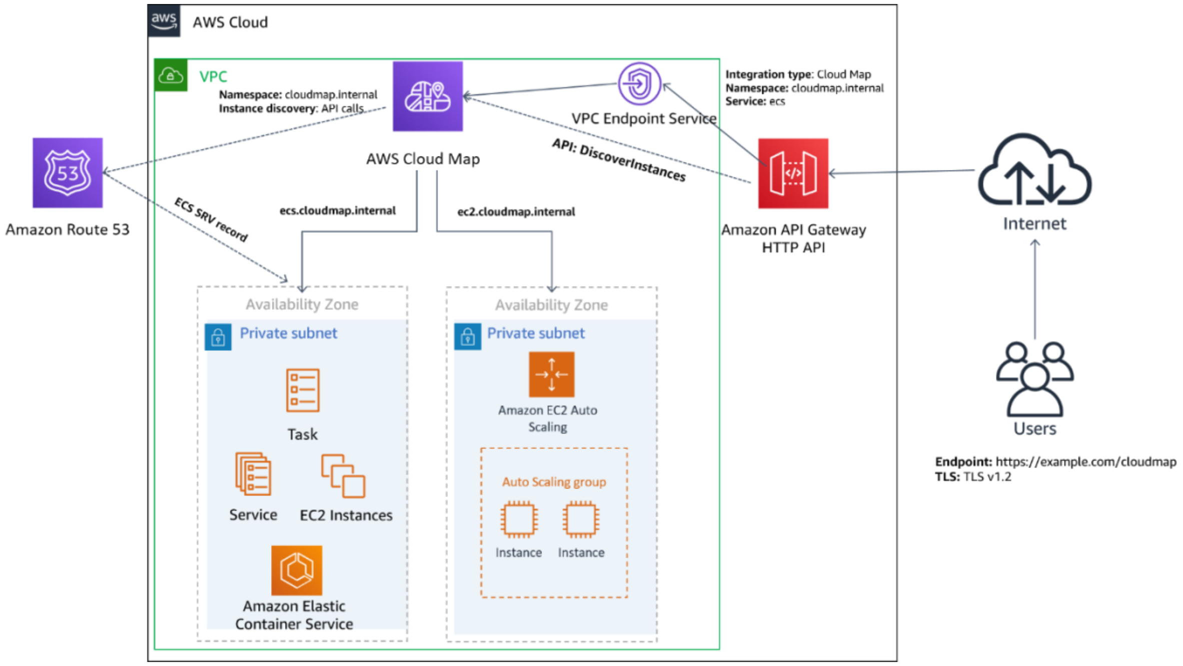


Figure 7 – HTTP API private integration with Cloud Map

Security

Private APIs and private integration offer an extra layer of security from a network standpoint, because communications are limited within a private network. However, malicious users can potentially gain access to private networks, so it's a best practice to implement an authorizer for APIs. REST and WebSocket offer the same set of authorizers, such as IAM, Amazon Cognito, and Lambda authorizers. Currently, HTTP APIs come with a JSON Web Token (JWT) authorizer. Serverless Application Lens covers identity and access management in serverless API in depth.

Table 3 – Authorizations

Authorization type	Available API type	Use case
IAM	REST, WebSocket, HTTP	If clients have IAM user or role credentials, they can sign the request with IAM credentials.

Authorization type	Available API type	Use case
Amazon Cognito	REST, WebSocket	This is commonly used for web and mobile applications where end users log in through Amazon Cognito user pools or federated identity providers.
Lambda	REST, WebSocket, HTTP	A Lambda authorizer enables developers to design a business logic around authorization. This can act as a JWT. authorizer, or validate other types of tokens.
JWT	HTTP	The JWT authorizer is available only for HTTP APIs, and allows clients to pass a JWT token, including tokens from Amazon Cognito.

Cost Optimization

Infrastructure cost is an important factor when choosing application architectures. For application use cases that require REST or HTTP APIs, HTTP APIs offer lower pricing tiers. For existing REST APIs, consider migrating to HTTP APIs. When planning for migration, see Choosing between HTTP APIs and REST APIs to compare HTTP API and REST API supported features.

For serverless API cost optimization, Serverless Application Lens covers cost optimization best practices such as cost-effective resources, matching supply and demand, expenditure awareness, and optimizing over time in Cost Optimization Pillar section.

For REST and HTTP API pricing, see Amazon API Gateway pricing. You may incur additional charges if you use API Gateway in conjunction with other AWS services, or transfer data out of AWS.

Table 4 – REST and HTTP API pricing

Endpoint type	Pricing
REST	Free tier: one million API calls per month for up to 12 months. API calls: 1. First 333 million requests (per month): $3.50 (per million) 2. Next 667 million requests (per month): $2.80 (per million) 3. Next 19 billion requests (per month): $2.38 (per million) 4. Over 20 billion requests (per month): $1.51 (per million) Caching: Billed per hour based on the cache memory size (not eligible for free tier).
HTTP	Free tier: one million API calls per month for up to 12 months. API calls (us-east-1); 1. First 300 million requests (per month): $1.00 (per million) 2. 300+ million requests (per month): $0.90 (per million) HTTP APIs are metered in 512 KB increments.

For private integration with REST and WebSocket APIs, a Network Load Balancer is required. The NLB cost is billed per hour, so while a VPC link remains active, you pay for the NLB. For a use case where requests to a REST or HTTP API are made infrequently, such as five requests per day, a VPC-enabled Lambda function can be a more cost-effective option. VPC-enabled Lambda functions can access VPC resources. Because Lambda bills per request and code execution duration, using a VPC-enabled Lambda function can cost less. See Elastic Load Balancing pricing and AWS Lambda Pricing.

Table 5 – Private integration vs. Lambda pricing

Integration/Lambda	Cost	Use cases
Private integration (NLB)	Billed per hour regardless of use.	If there is a backend service hosted in ECS or other target such as EC2 instances that can be directly integrated with NLB, using an NLB to route traffic simplifies the architecture.
VPC-enabled Lambda	Lambda pricing is billed on-demand, so if a Lambda function is not used, there is no charge.	If there is any private resource like RDS which cannot be directly accessed by NLB, using a VPC-enabled Lambda function is a good alternative.

Conclusion

Amazon API Gateway provides different API types and endpoint types. This paper primarily covered private API and integration design patterns, and best practices. Additionally, it covered security and cost optimization. You can leverage the information provided in this whitepaper to determine the best-suited architecture for your application.

Contributors

Contributors to this document include:

- Takaki Matsumoto, Cloud Support Engineer II, Premium Support

Further Reading

For additional information, see:

- AWS Well-Architected Framework
- Serverless Applications Lens - AWS Well-Architected Framework

Document Revisions

Date	Description
January 2021	First publication