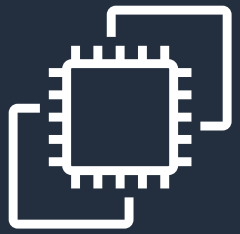
Lambda and API Gateway

Team or presenters name

Date

Broadest and deepest platform choice



Amazon EC2

Virtual server instances in the cloud

Amazon ECS, EKS, and Fargate

Container management service for running Docker on a managed cluster of EC2

AWS Lambda

Serverless compute for stateless code execution in response to triggers

aws

AWS compute offerings

Service	**Amazon EC2**	**Amazon ECS**	**AWS Lambda**
Unit of scale	**VM**	**Task**	**Function**
Level of abstraction	**H/W**	**OS**	**Runtime**

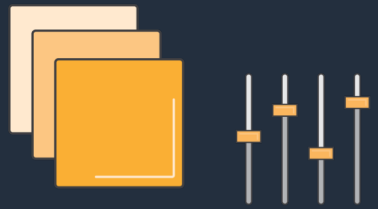
aws

AWS compute offerings

Service	Amazon EC2	Amazon ECS	AWS Lambda
How do I choose?	I want to configure servers, storage, networking, and my OS	I want to run servers, configure applications, and control scaling	Run my code when it's needed

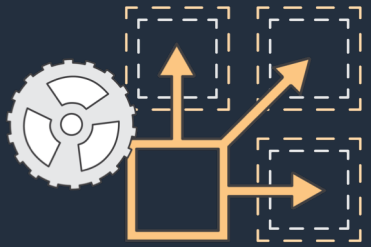
aws

Owning servers means dealing with ...



Operations and management

Scaling



Provisioning and utilization

Availability and fault tolerance

aws

AWS Lambda: Run code in response to events

EVENT SOURCE

FUNCTION

SERVICES (ANYTHING)

Changes in data state

Requests to endpoints

Changes in resource state

Node
Python
Java
C#

aws

Benefits of Lambda and serverless compute

No servers to provision or manage

Scales with usage

Never pay for idle

Availability and fault tolerance built in

aws

AWS Lambda pricing

- Buy compute time in 100 ms increments
- Low request charge
- No hourly, daily, or monthly minimums
- No per-device fees

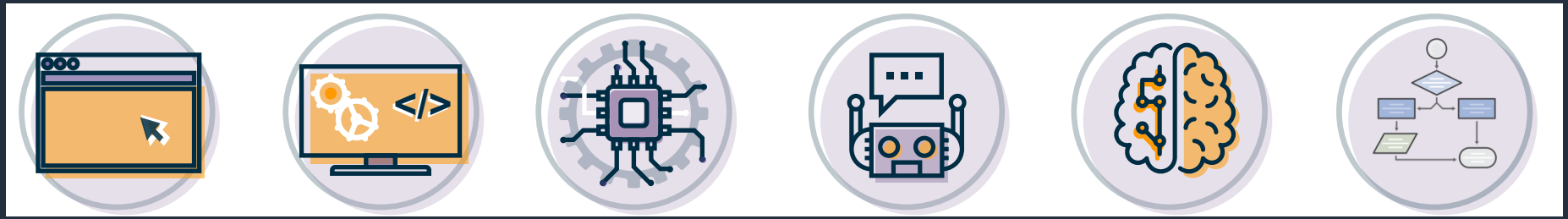
Free Tier

1 million requests and 400,000 GBs of compute every month, every customer

Never pay for idle!

aws

Common use cases



Web Applications

- Static websites
- Complex web apps
- Packages for Flask and Express

Backends

- Apps & services
- Mobile
- IoT

Data Processing

- Real time
- MapReduce
- Batch

Chat Bots

- Powering chat bot logic

Amazon Alexa

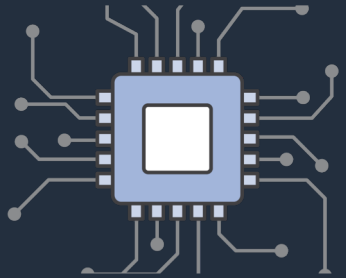
- Powering voice-enabled apps
- Alexa Skills Kit

IT Automation

- Policy engines
- Extending AWS services
- Infrastructure management

aws

What is API Gateway?



Create a unified API frontend for multiple micro-services

DDoS protection & throttling for your backend, caching for performance

Authenticate and authorize requests to a backend

Document your APIs with Swagger

aws

Feedback from our customers on running APIs

Managing multiple versions and stages of an API is difficult.

Monitoring third-party developers' access is time consuming.

Access authorization is a challenge.

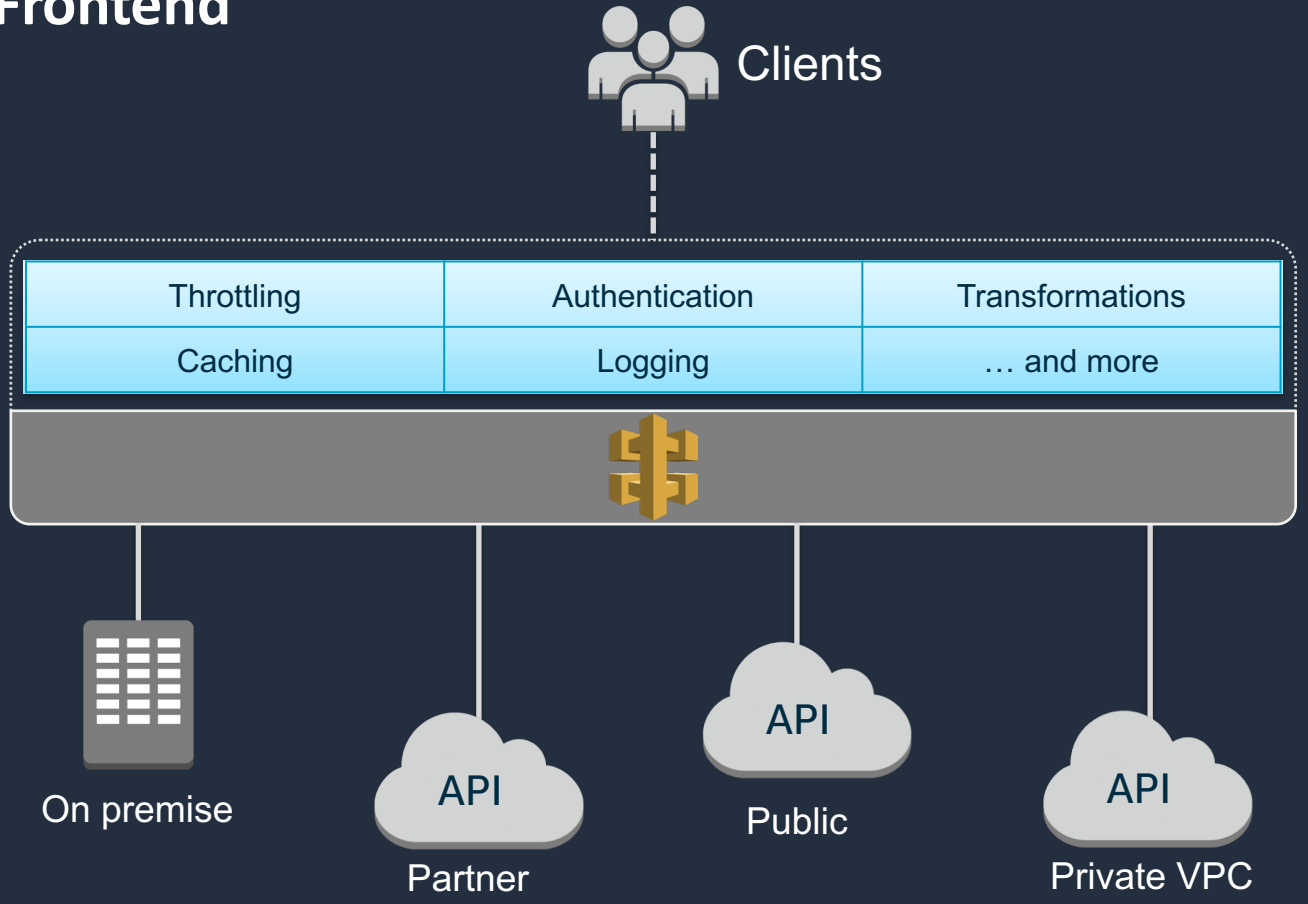
Traffic spikes create an operational burden.

What if I don't want to manage infrastructure at all?

aws

Unified frontend

Unified Frontend



Hosting Backend ASP.NET APIs on AWS

EC2, Elastic Beanstalk

- .NET Framework
- .NET Core
- Use with ALB & auto-scaling
- Choose instance size, memory, disk, vCPUs
- You manage servers

Containers

- .NET Framework
  (Windows containers on ECS)
- .NET Core
  (ECS or EKS, can use Fargate)
- Use with ALB & auto-scaling
- You provide container images

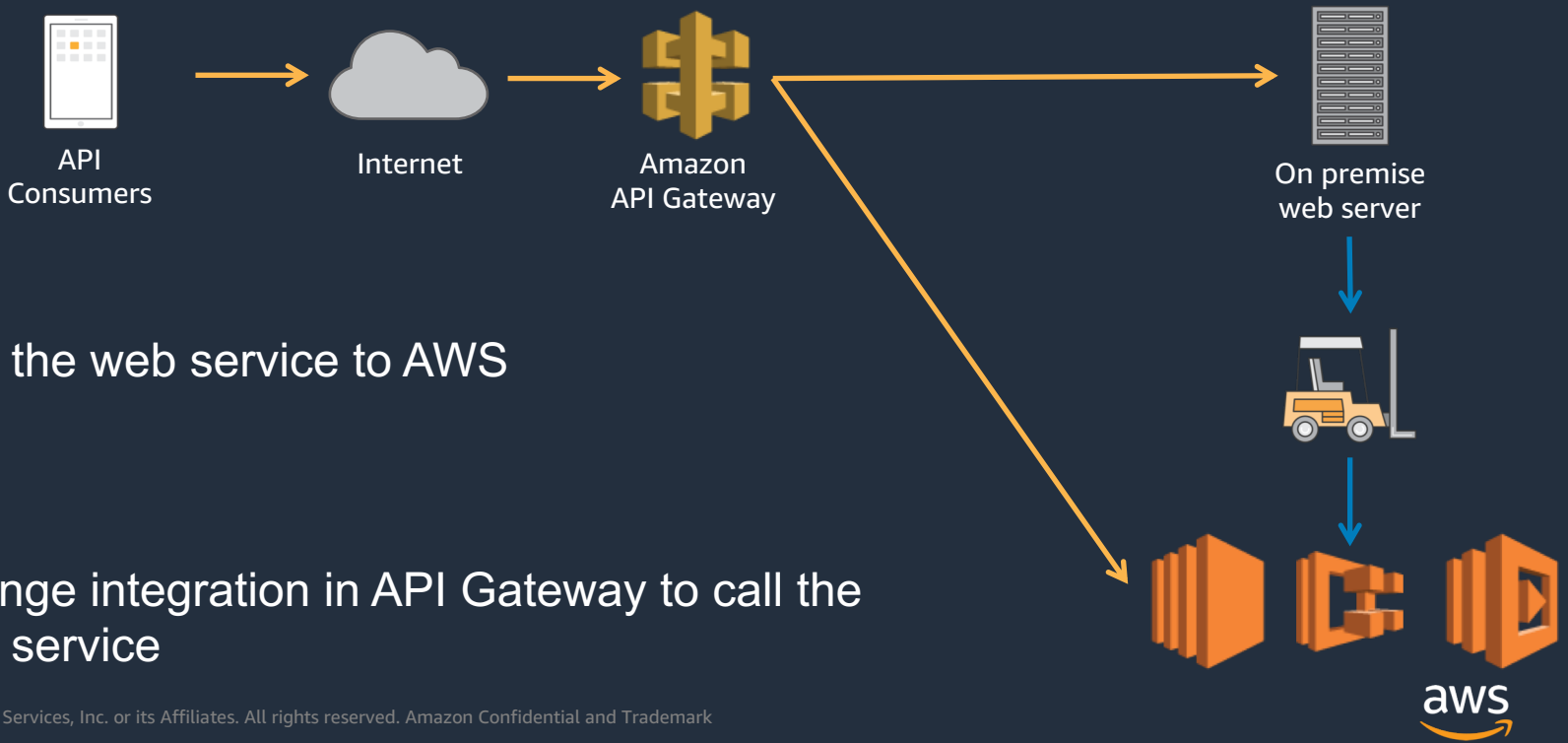
Lambda Functions

- .NET Core
- AWS scales and manages infrastructure
- Fully integrated with API Gateway
- Per for code execution time (per 100 ms)
- You provide the code

aws

Unified frontend – Migrating to AWS

1. Use API Gateway in front of an on-premise web service

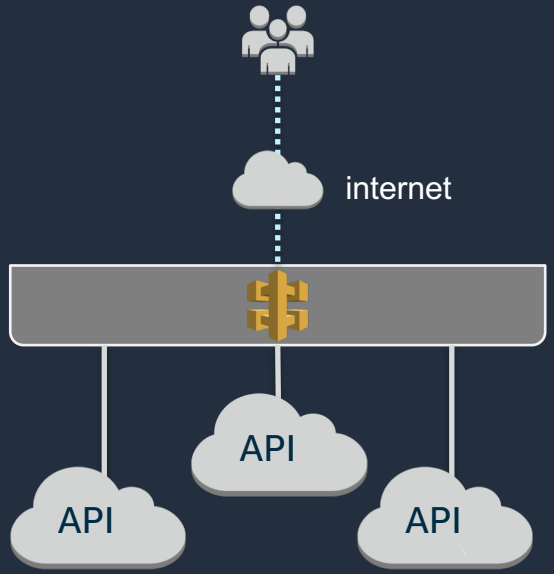


2. Port the web service to AWS

3. Change integration in API Gateway to call the new service

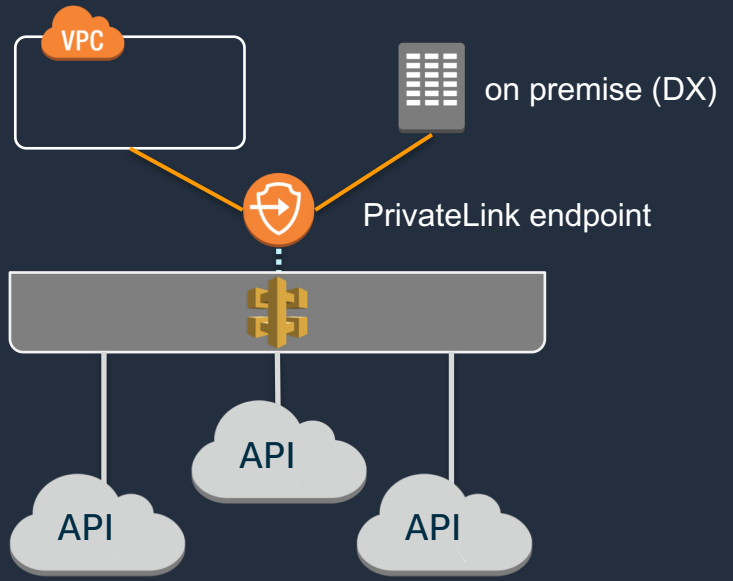
API Gateway APIs: Public or Private

Publicly-accessible



Backend services

Access from VPCs or AWS Accounts

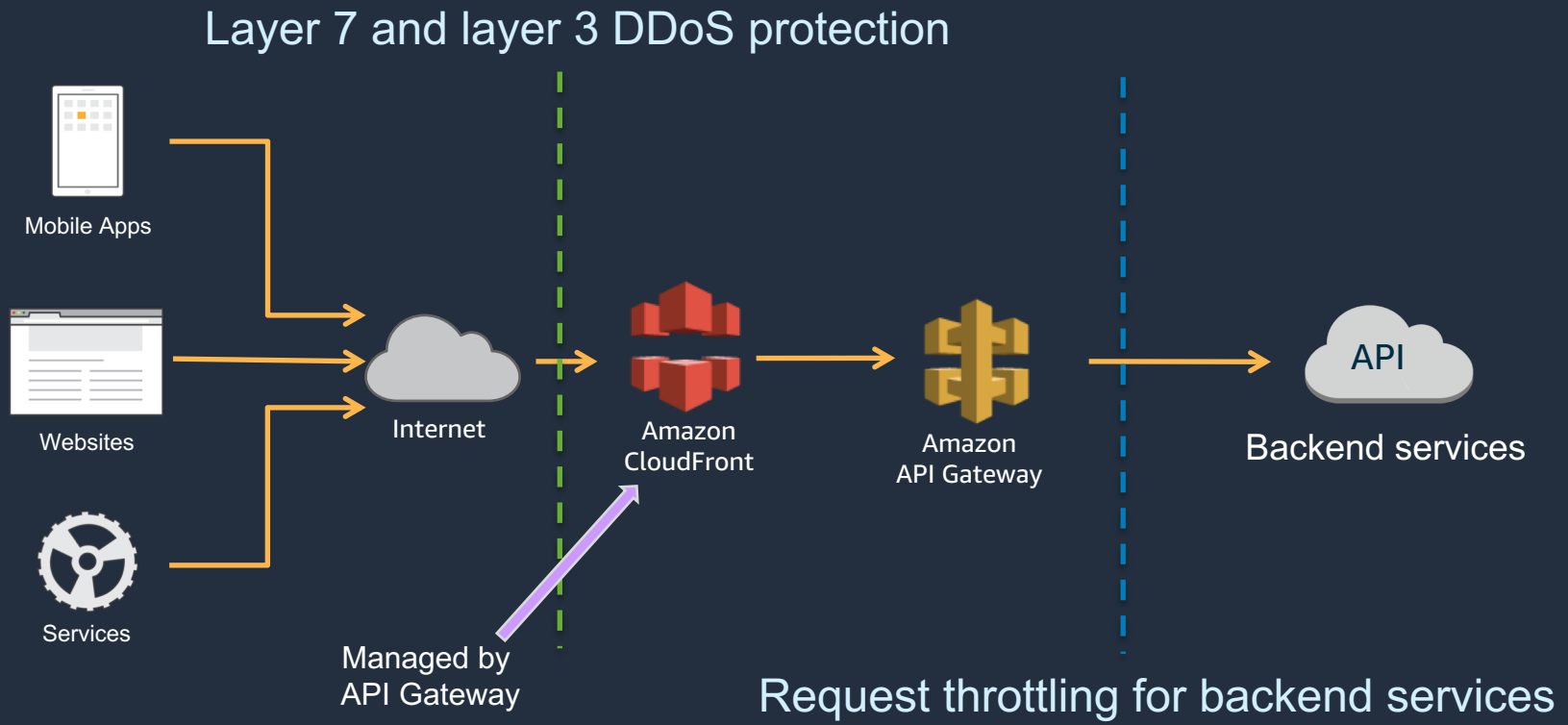


Backend services

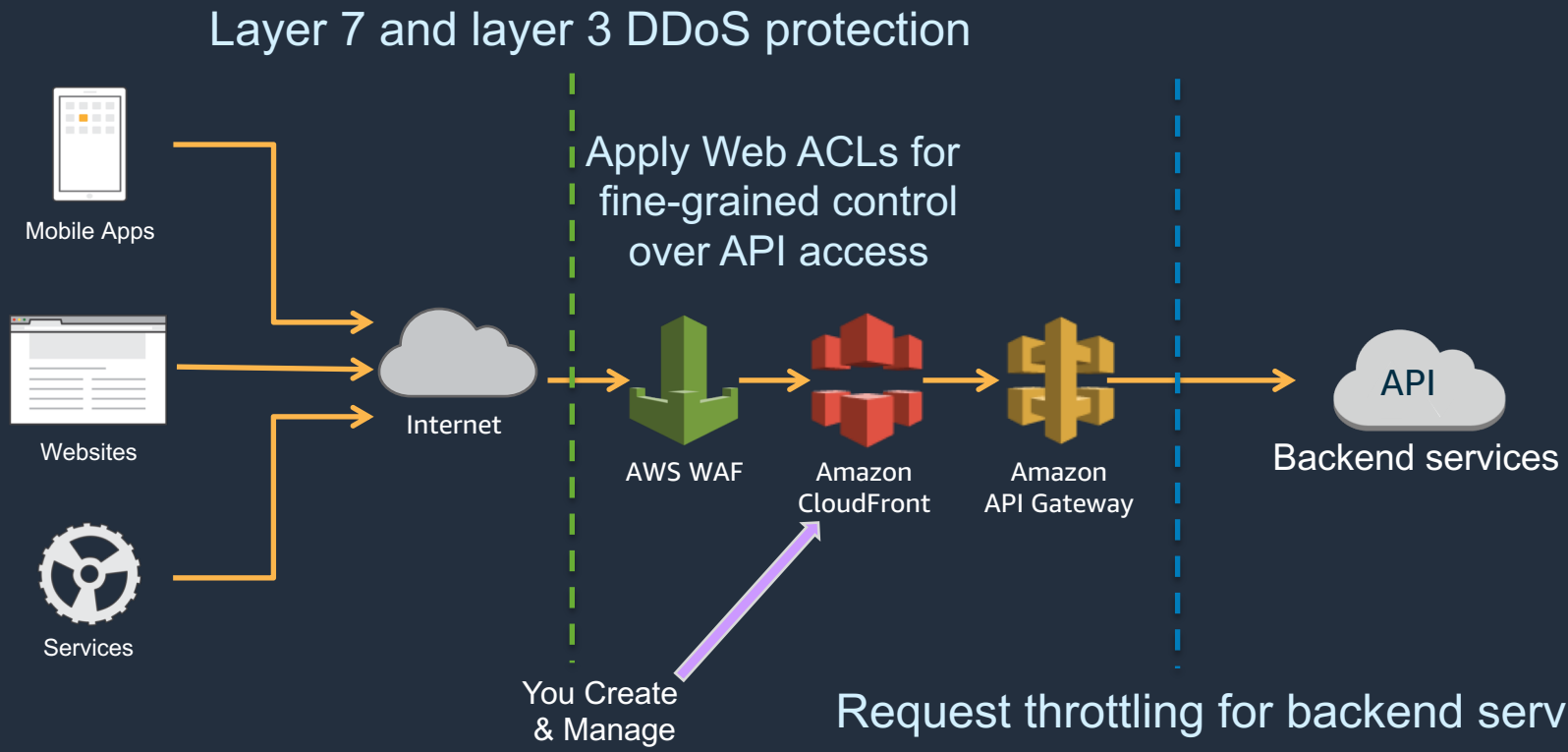
aws

DDoS Protection, Throttling and Caching

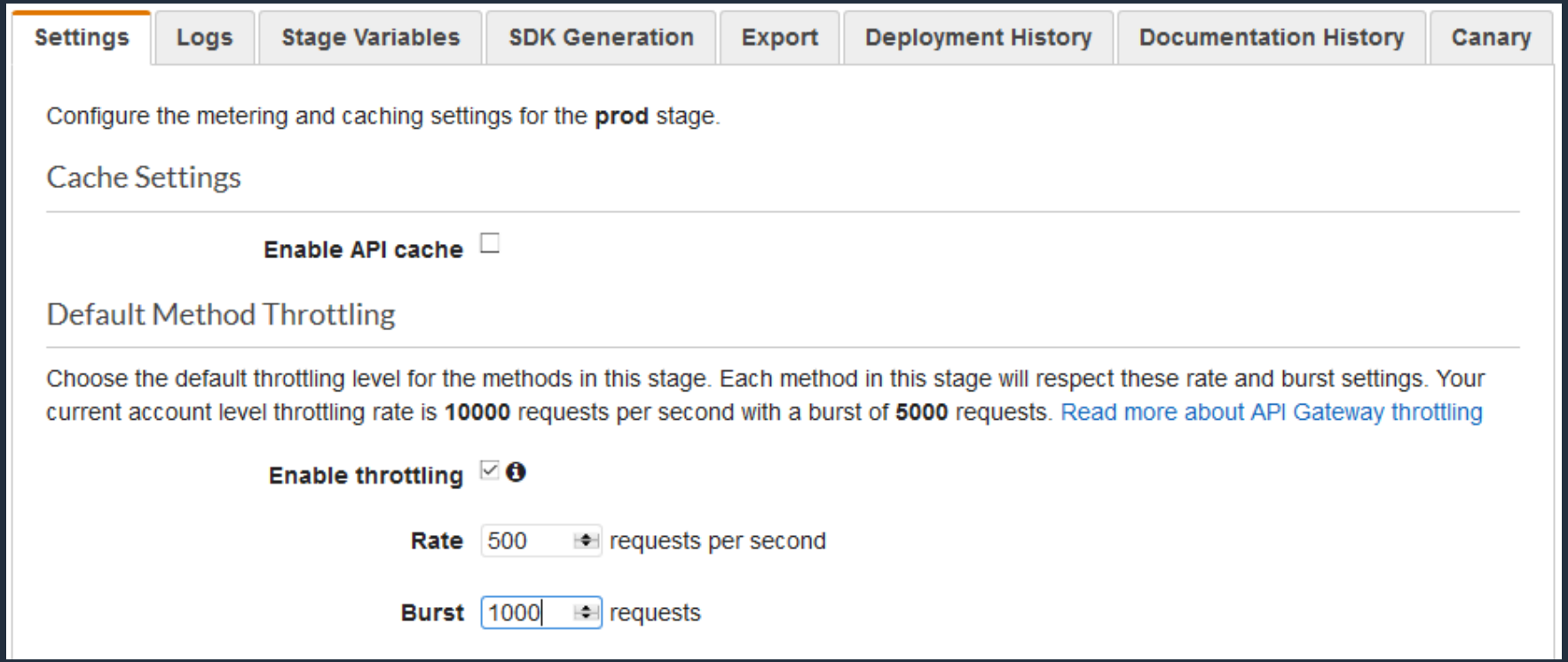
DDoS and network protection – edge-optimized APIs



Regional API endpoint + WAF

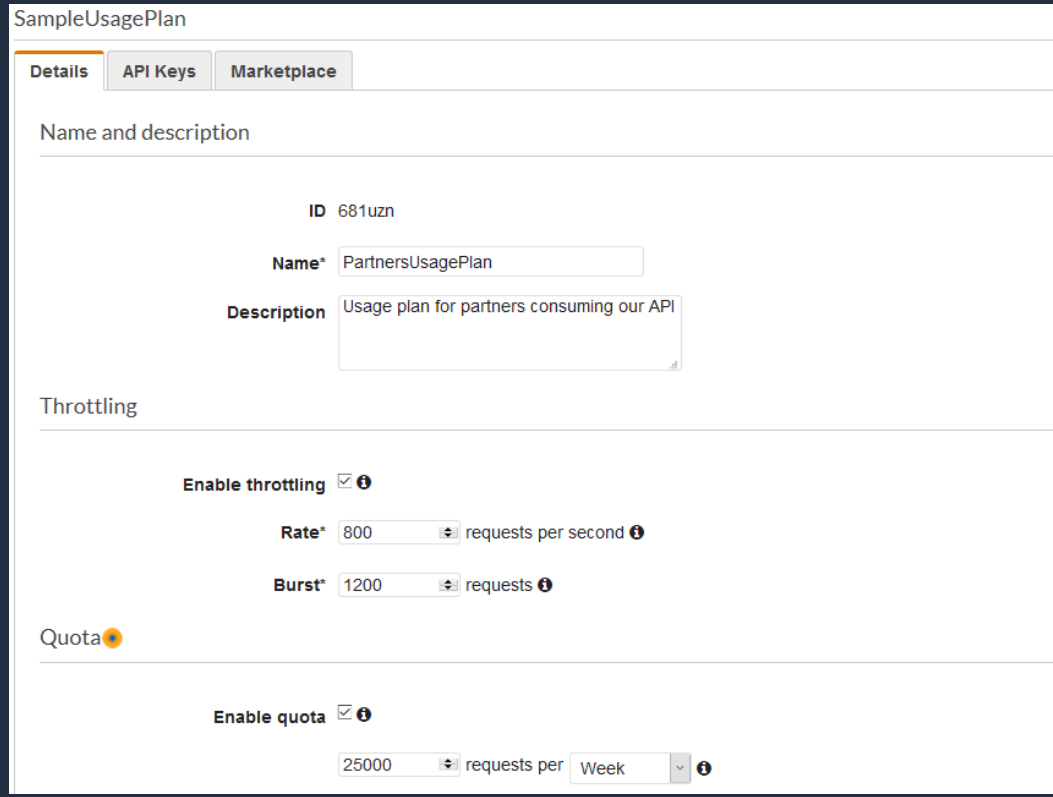


Default Method Throttling



Usage Plans

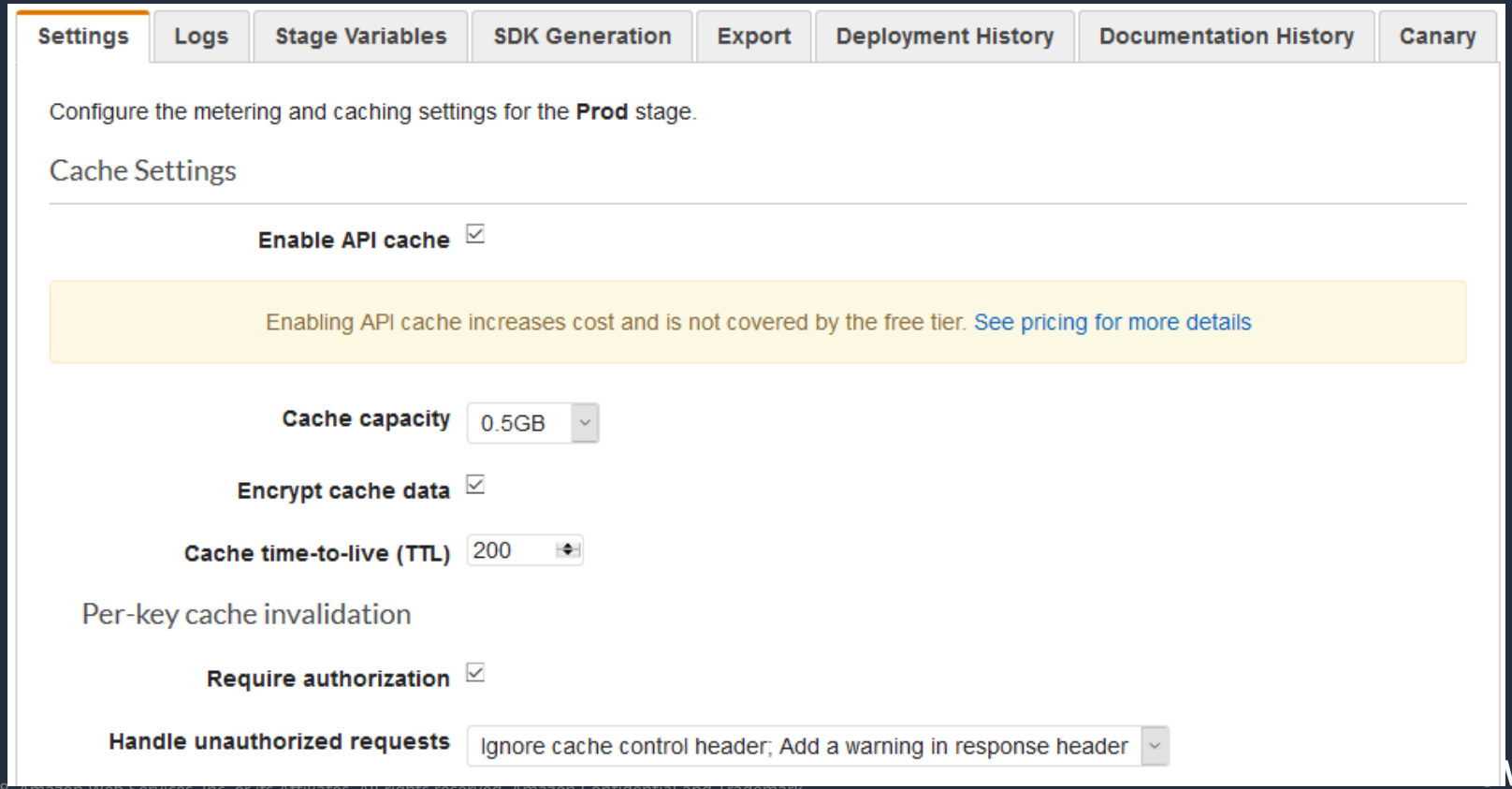
- Associate w/multiple API Keys
- Associate w/multiple APIs and Stages
- Set usage quotas, grant temporary extensions
- Vend your API on the AWS SaaS Marketplace
- Export usage data as JSON or CSV for entire plan, or individual API Keys



Caching

- You can configure a cache key and the Time to Live (TTL) of the API response
- Cached items are returned without calling the backend
- A cache is dedicated to you, by stage
- You can provision between 0.5GB to 237GB of cache
- You can optionally encrypt the cache for sensitive data
- Apply cache settings globally, or override per method

Caching



Authentication & Authorization

API Gateway Authorization Options

AWS Signature Version 4 “Sigv4”

- Easy signing using the generated SDK

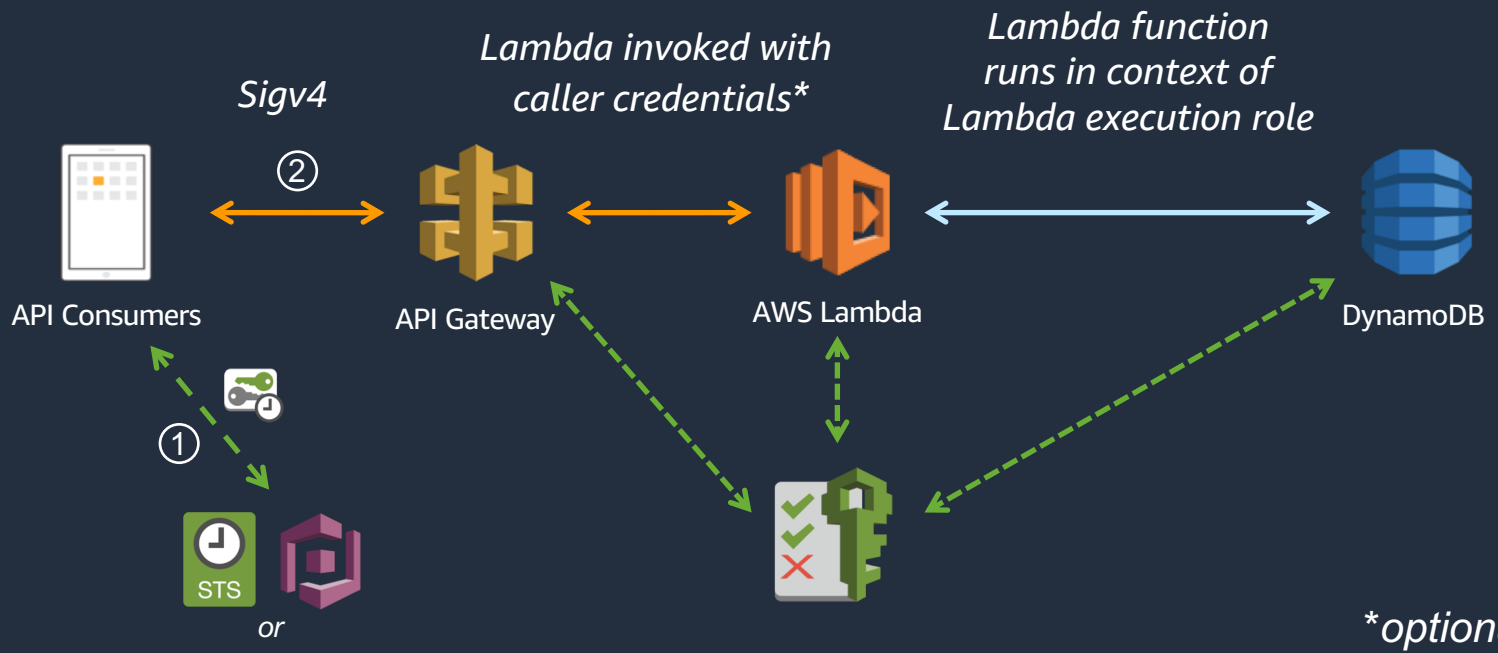
Amazon Cognito User Pools

- Authenticate with Cognito
- Pass identity or access token to API Gateway

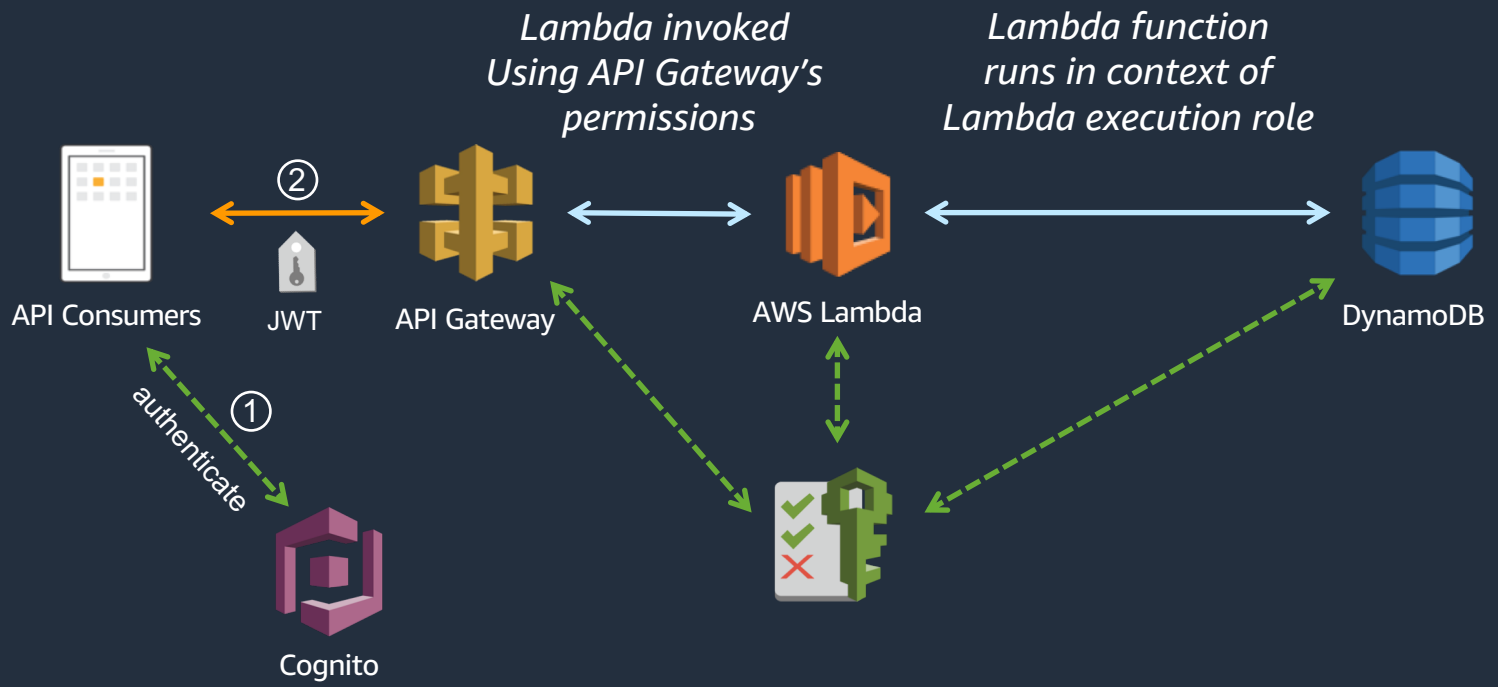
Lambda Authorizer *(formerly Custom Authorizer)*

- Authorization performed by Lambda function
- Handle OAuth, SAML, or other strategies

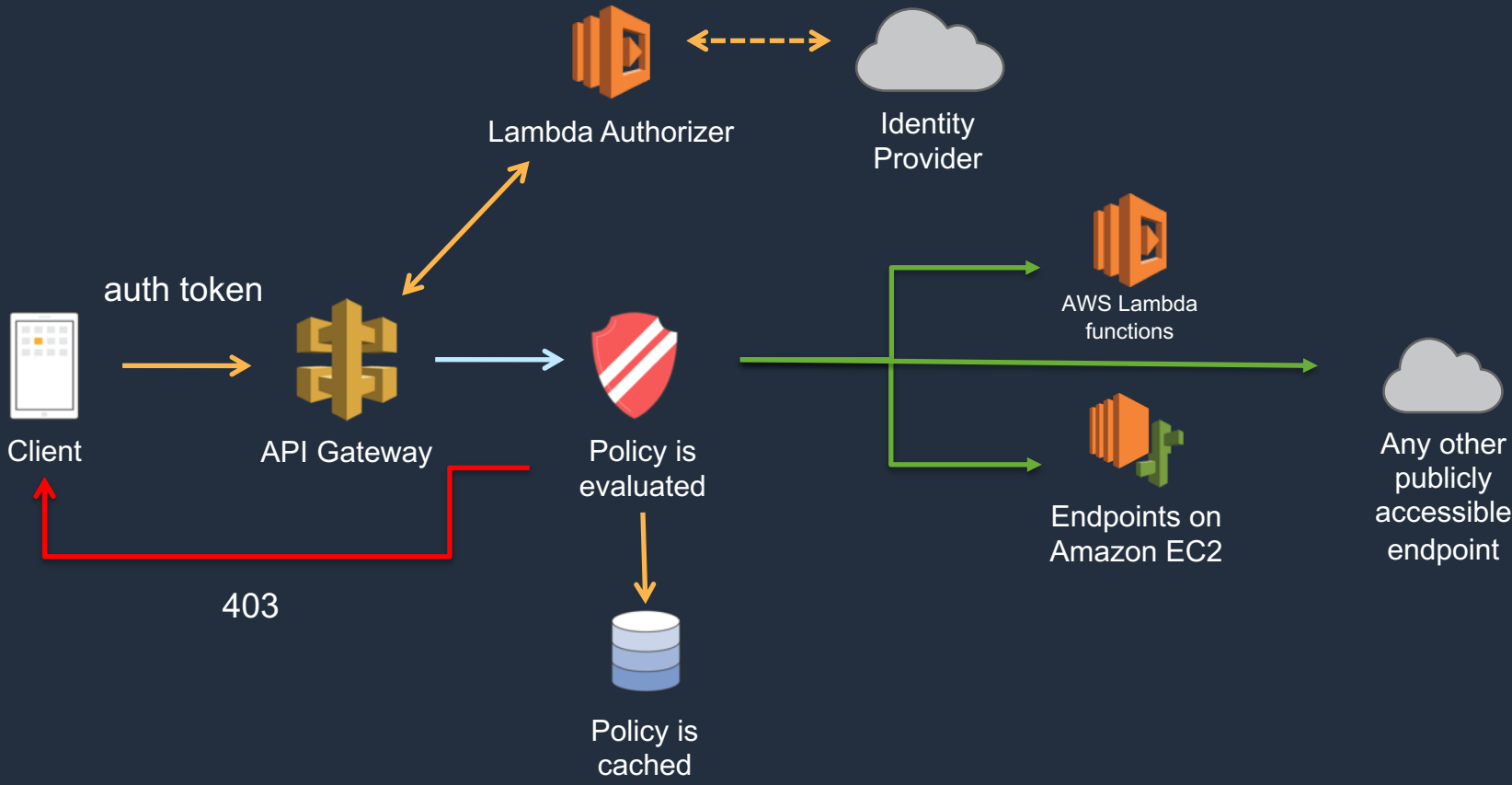
Authorization – AWS Sigv4



Authorization – Cognito User Pool



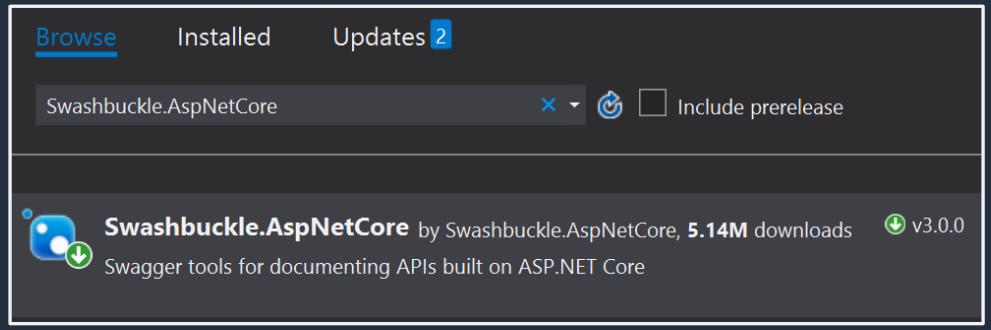
Authorization – Custom Authorizer



Document APIs with Swagger

aws

Generating Swagger with Swashbuckle

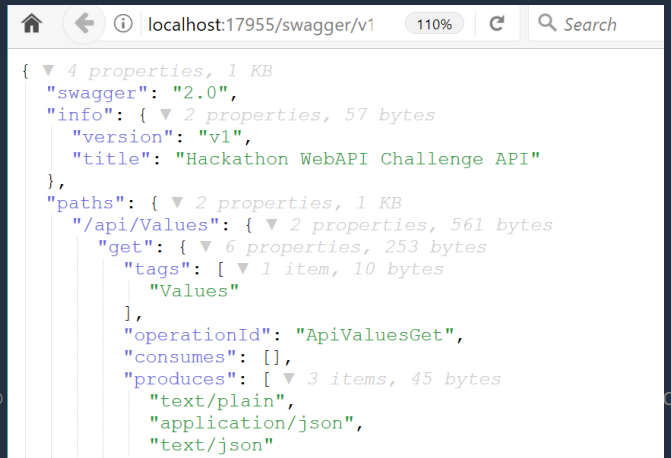


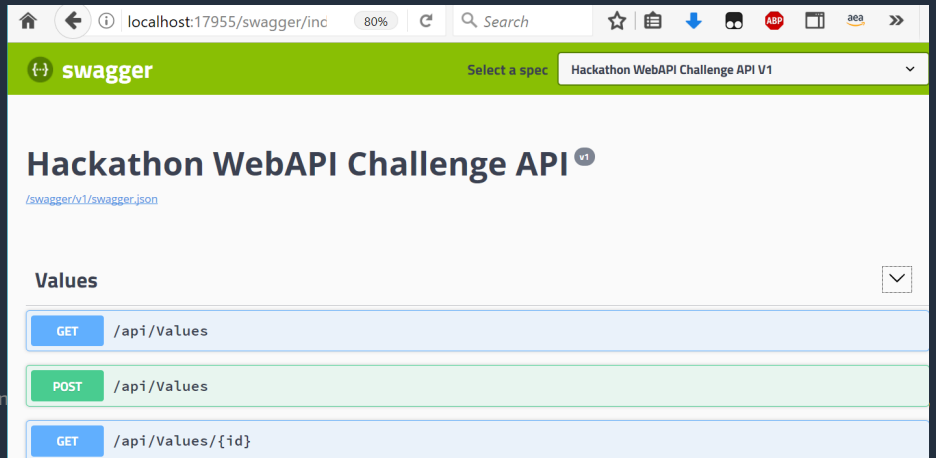
Swashbuckle NuGet meta-package

Automatically includes:

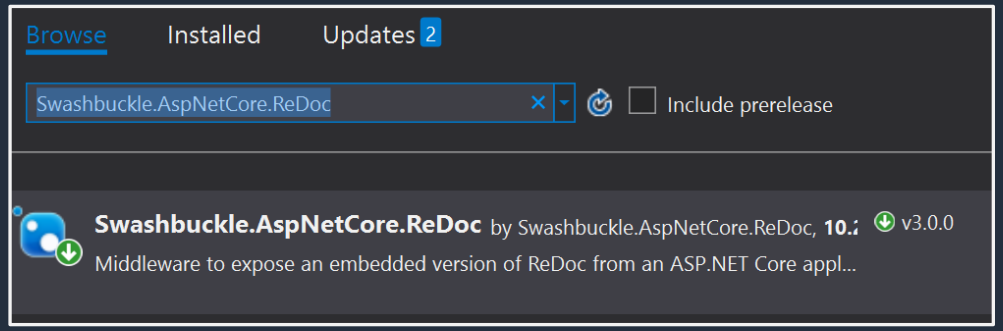
- Swashbuckle.AspNetCore.Swagger
- Swashbuckle.AspNetCore.SwaggerGen
- Swashbuckle.AspNetCore.SwaggerUI

Swashbuckle can automatically generate Swagger files for your ASP.NET APIs, and even generate the UI for easy consumption:





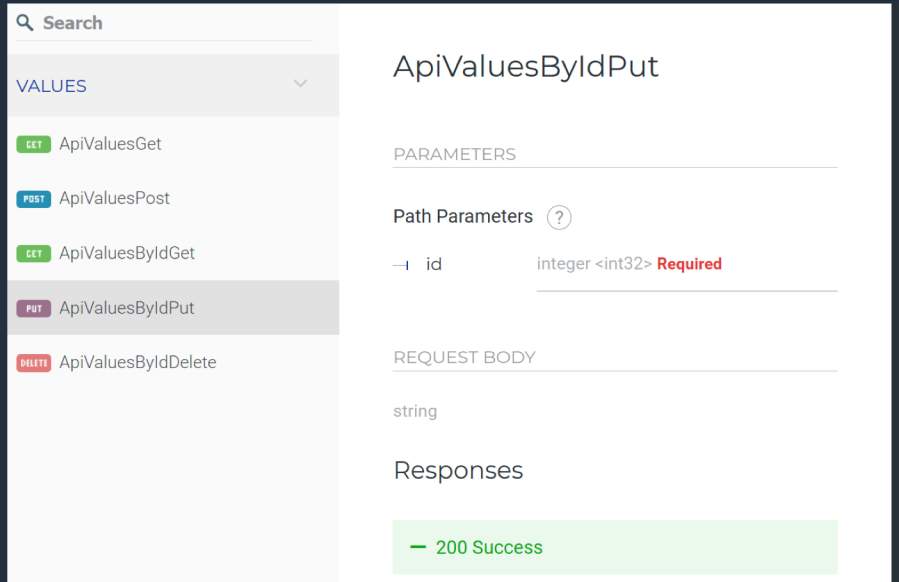
ReDoc UI with Swashbuckle



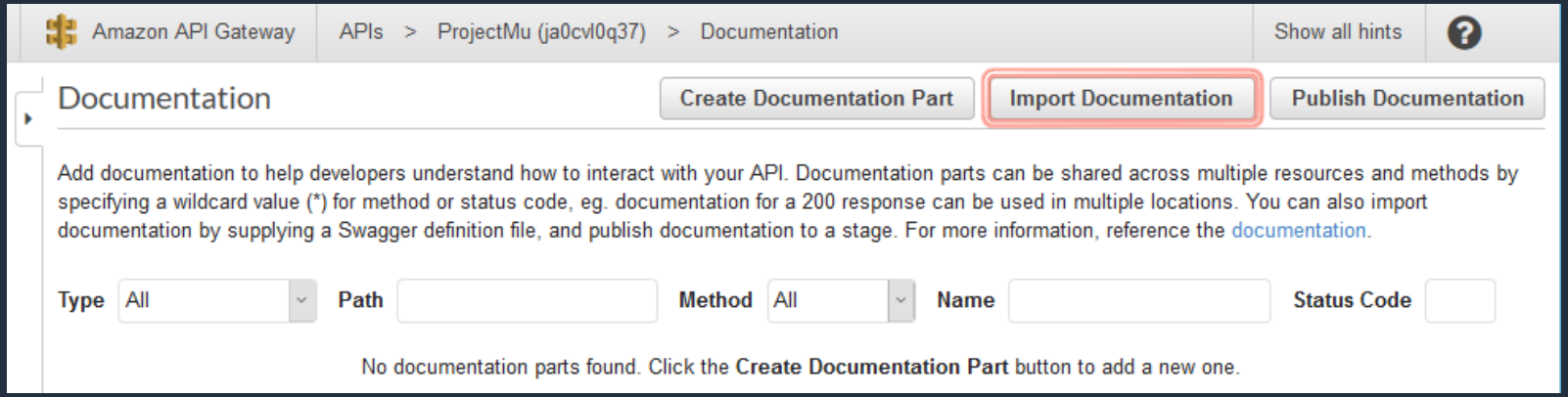
Swashbuckle ReDoc NuGet package

- Swashbuckle.AspNetCore.ReDoc

This package generates a documentation UI using a React.js front-end, and is an alternative to the Swagger-ui



Importing Swagger Documentation



Import your swagger file (json) to populate your API Gateway API documentation. You can then control who has access to the documentation with IAM policies.

API Gateway uses extensions to the Swagger spec to support AWS-specific authorization and API Gateway-specific API integrations.

For details, refer to:
docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-swagger-extensions.html

aws

Q&A

Name of presenter

Thank you!