



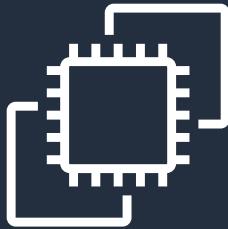
Lambda and API Gateway



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 compute offerings



Service

Amazon EC2

Amazon ECS

AWS Lambda

Unit of scale

VM

Task

Function

Level of abstraction

H/W

OS

Runtime



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

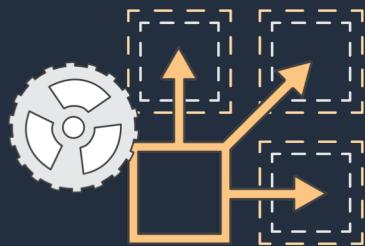
Owning servers means dealing with ...



Operations and management



Scaling

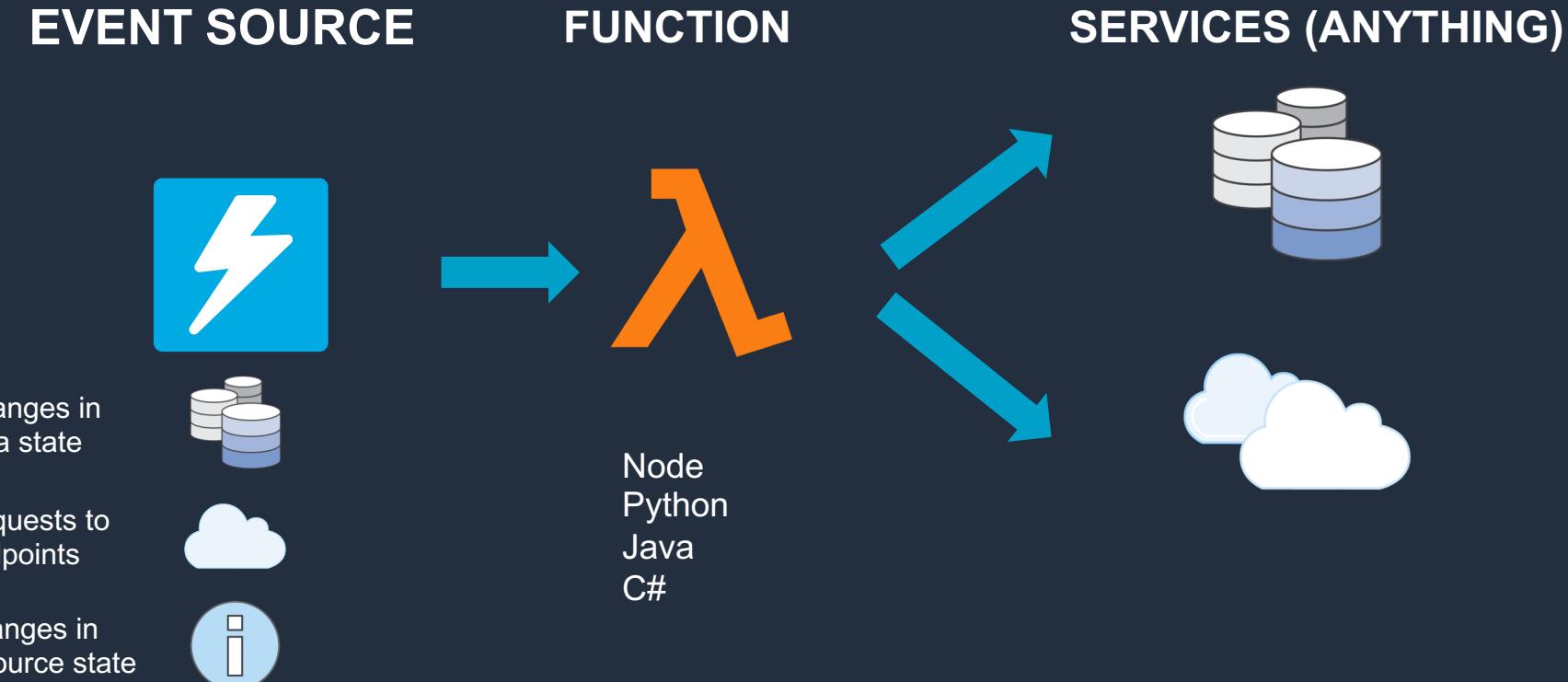


Provisioning and utilization



Availability and fault tolerance

AWS Lambda: Run code in response to events



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 Lambda pricing

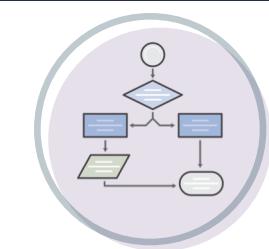
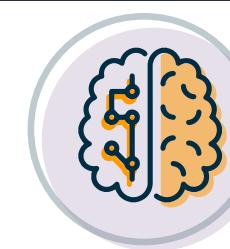
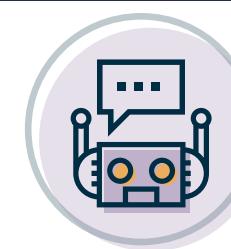
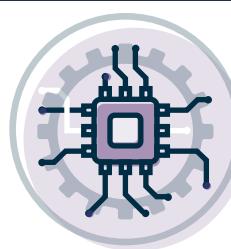
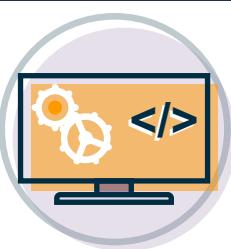
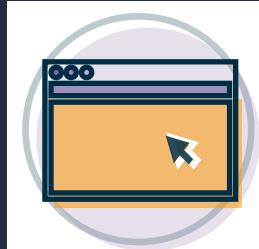
- Buy compute time in 100 ms increments
- Low request charge
- No hourly, daily, or monthly minimums
- No per-device fees

Free Tier

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

Never pay for idle!

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

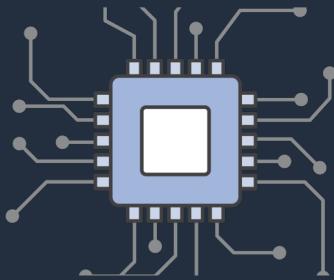
Amazon Alexa

- Powering voice-enabled apps
- Alexa Skills Kit

IT Automation

- Policy engines
- Extending AWS services
- Infrastructure management

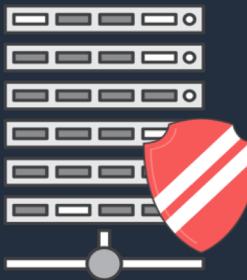
What is API Gateway?



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

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.



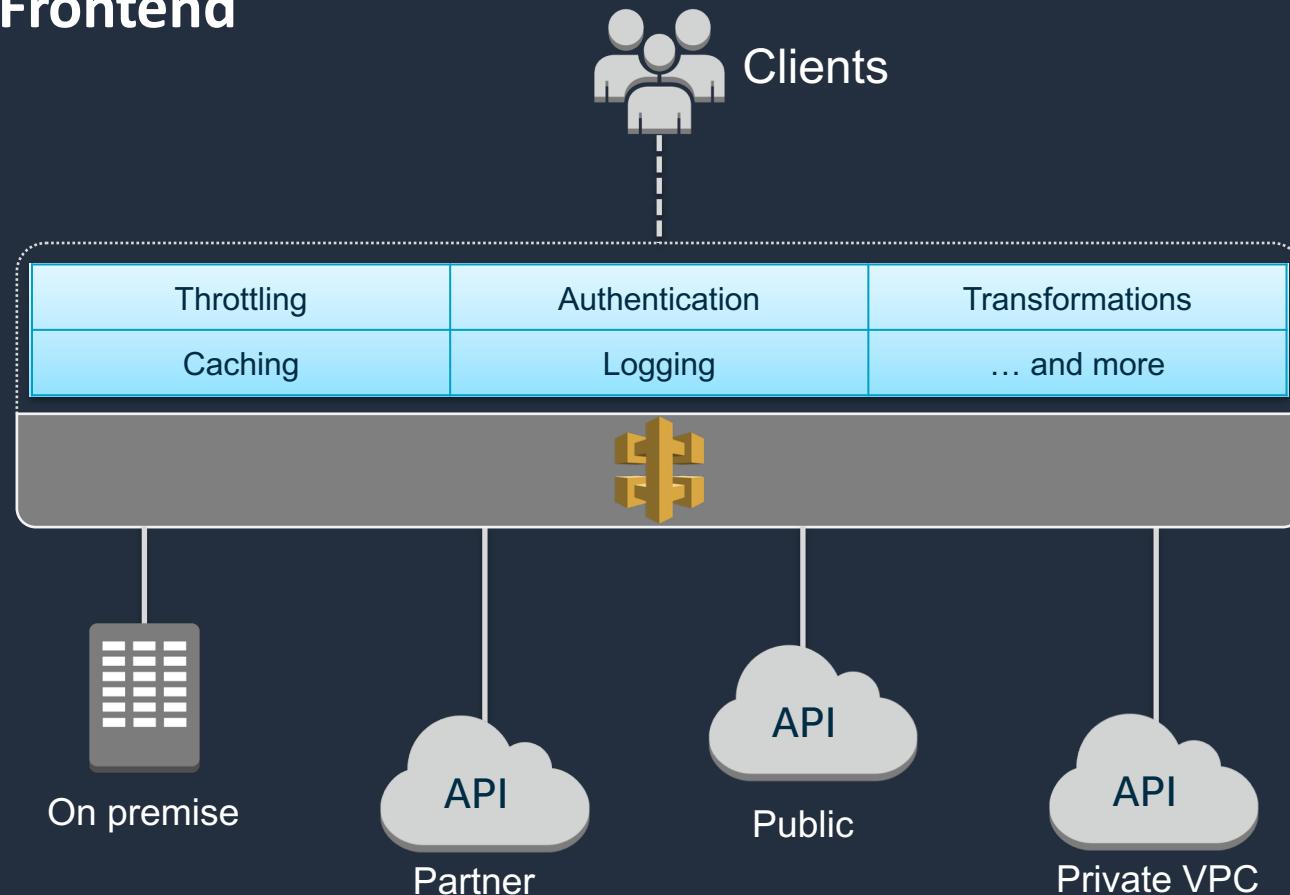
Traffic spikes create an operational burden.



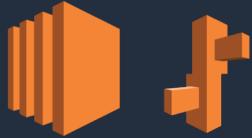
What if I don't want to manage infrastructure at all?

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
- Pay for code execution time (per 100 ms)
- You provide the code

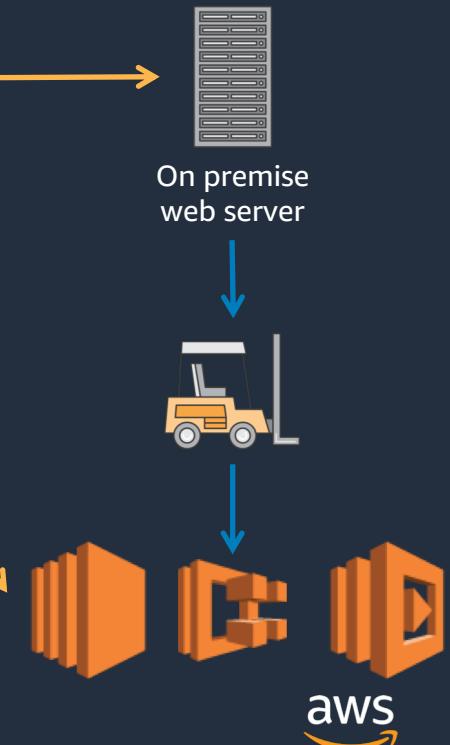
Unified frontend – Migrating to AWS

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



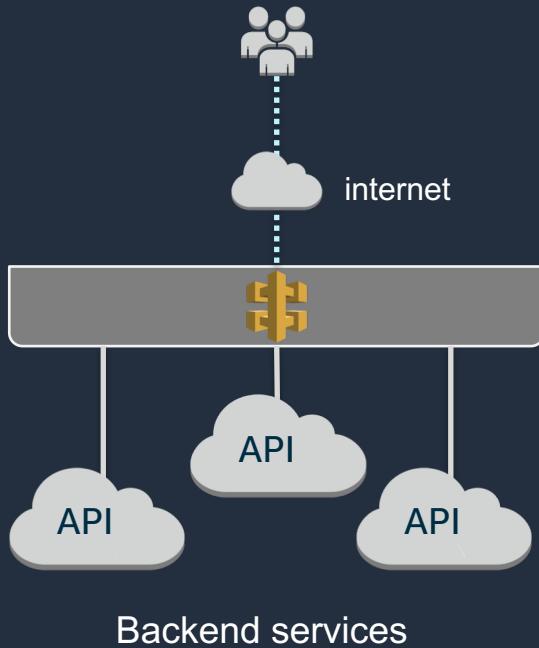
2. Port the web service to AWS

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

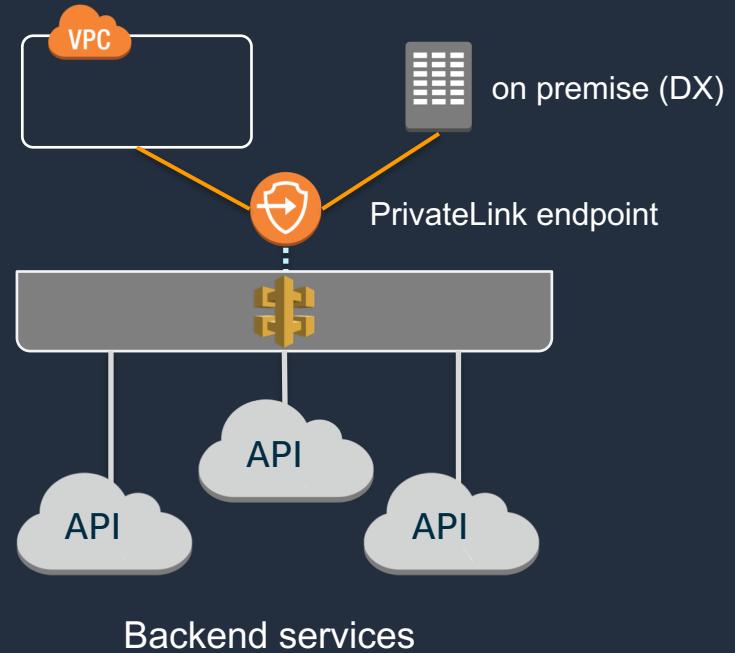


API Gateway APIs: Public or Private

Publicly-accessible

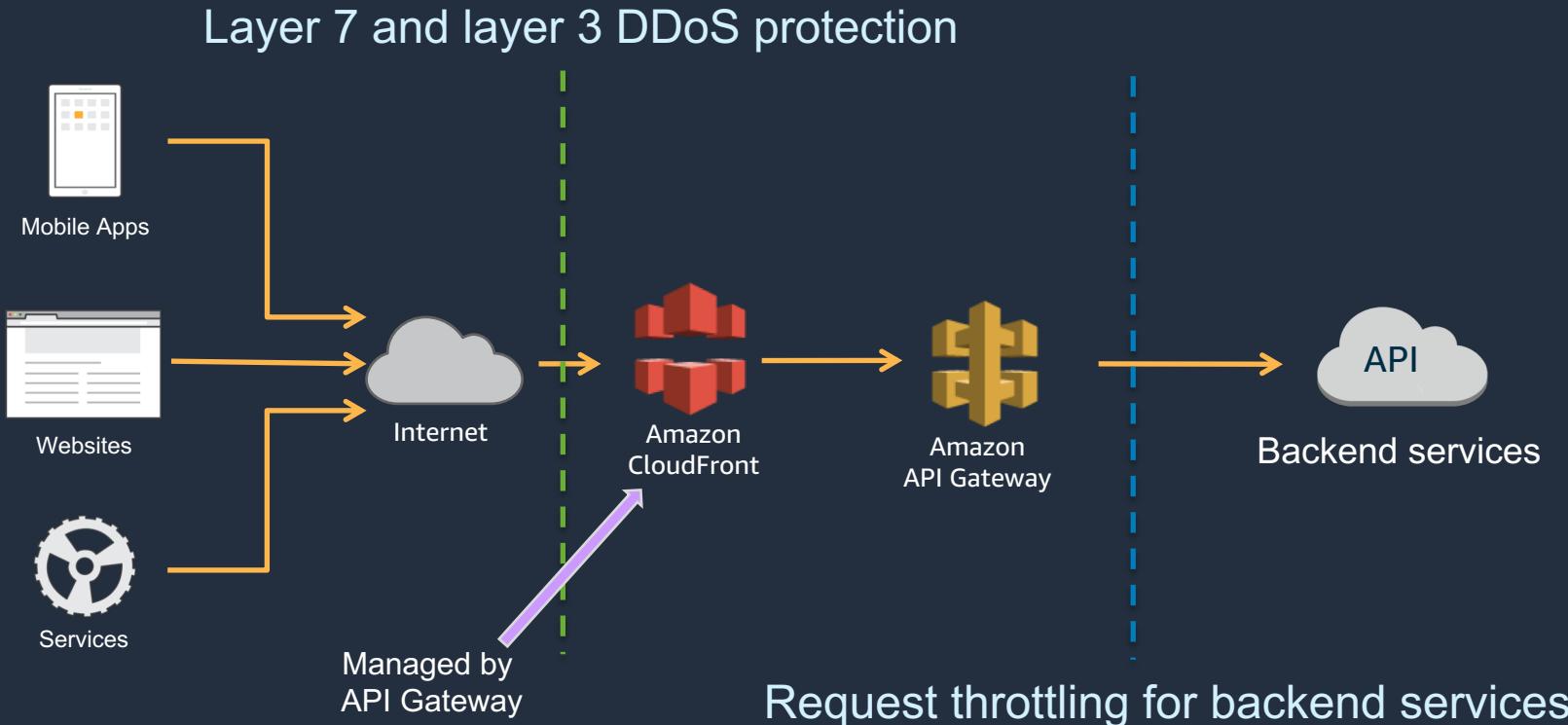


Access from VPCs or AWS Accounts

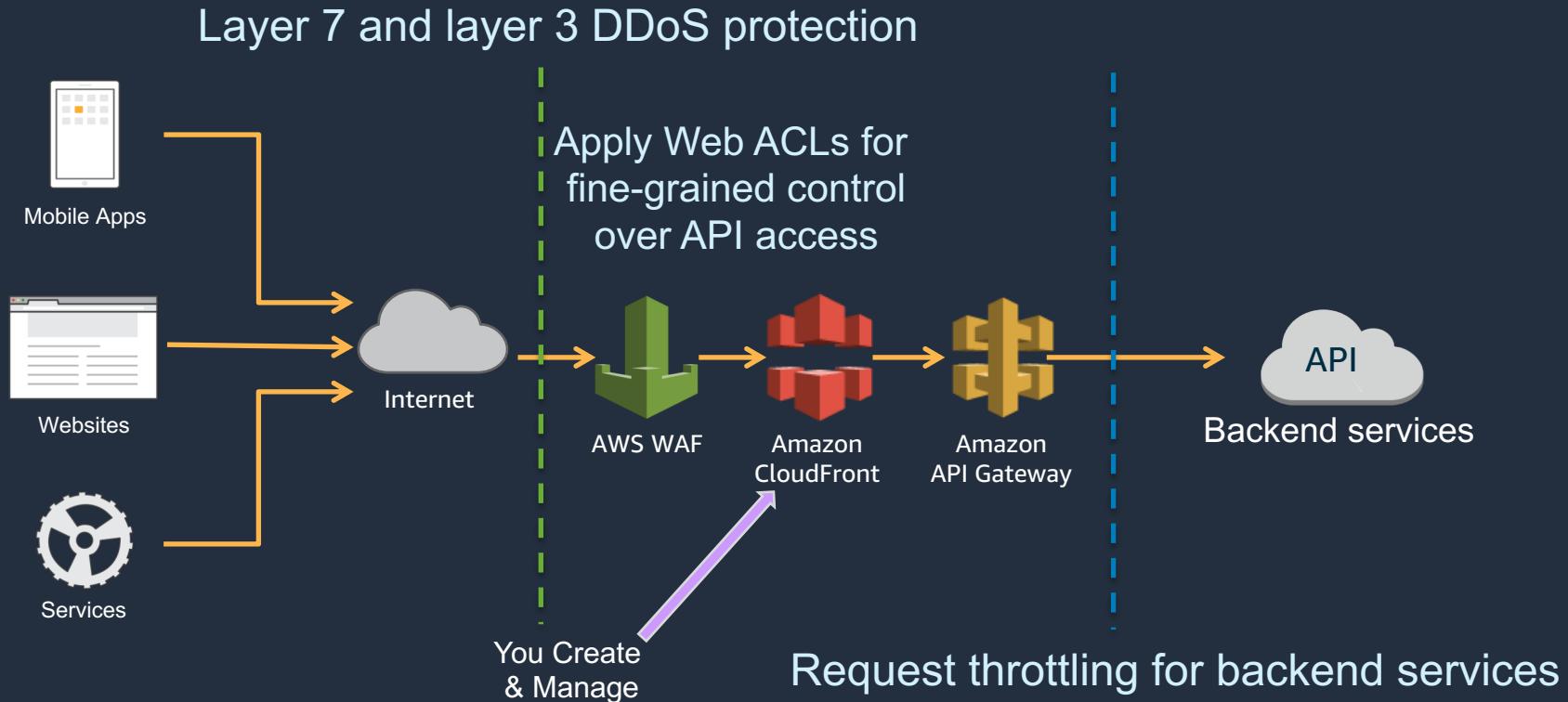


DDoS Protection, Throttling and Caching

DDoS and network protection – edge-optimized APIs



Regional API endpoint + WAF



Default Method Throttling

Settings Logs Stage Variables SDK Generation Export Deployment History Documentation History Canary

Configure the metering and caching settings for the **prod** stage.

Cache Settings

Enable API cache

Default Method Throttling

Choose the default throttling level for the methods in this stage. Each method in this stage will respect these rate and burst settings. Your current account level throttling rate is **10000** requests per second with a burst of **5000** requests. [Read more about API Gateway throttling](#)

Enable throttling ⓘ

Rate requests per second

Burst requests

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

SampleUsagePlan

Details API Keys Marketplace

Name and description

ID 681uzn

Name* PartnersUsagePlan

Description Usage plan for partners consuming our API

Throttling

Enable throttling

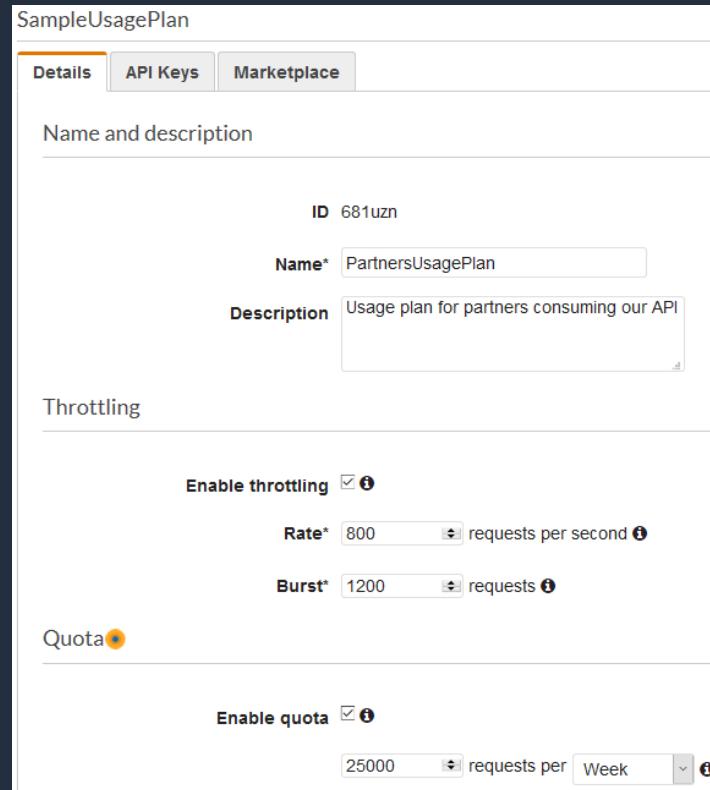
Rate* 800 requests per second i

Burst* 1200 requests i

Quota ?

Enable quota

25000 requests per Week i



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

Settings Logs Stage Variables SDK Generation Export Deployment History Documentation History Canary

Configure the metering and caching settings for the **Prod** stage.

Cache Settings

Enable API cache

Enabling API cache increases cost and is not covered by the free tier. [See pricing for more details](#)

Cache capacity

Encrypt cache data

Cache time-to-live (TTL)

Per-key cache invalidation

Require authorization

Handle unauthorized requests

Authentication & Authorization

API Gateway Authorization Options



AWS Signature Version 4 "Sigv4"

- Easy signing using the generated SDK



Amazon Cognito User Pools

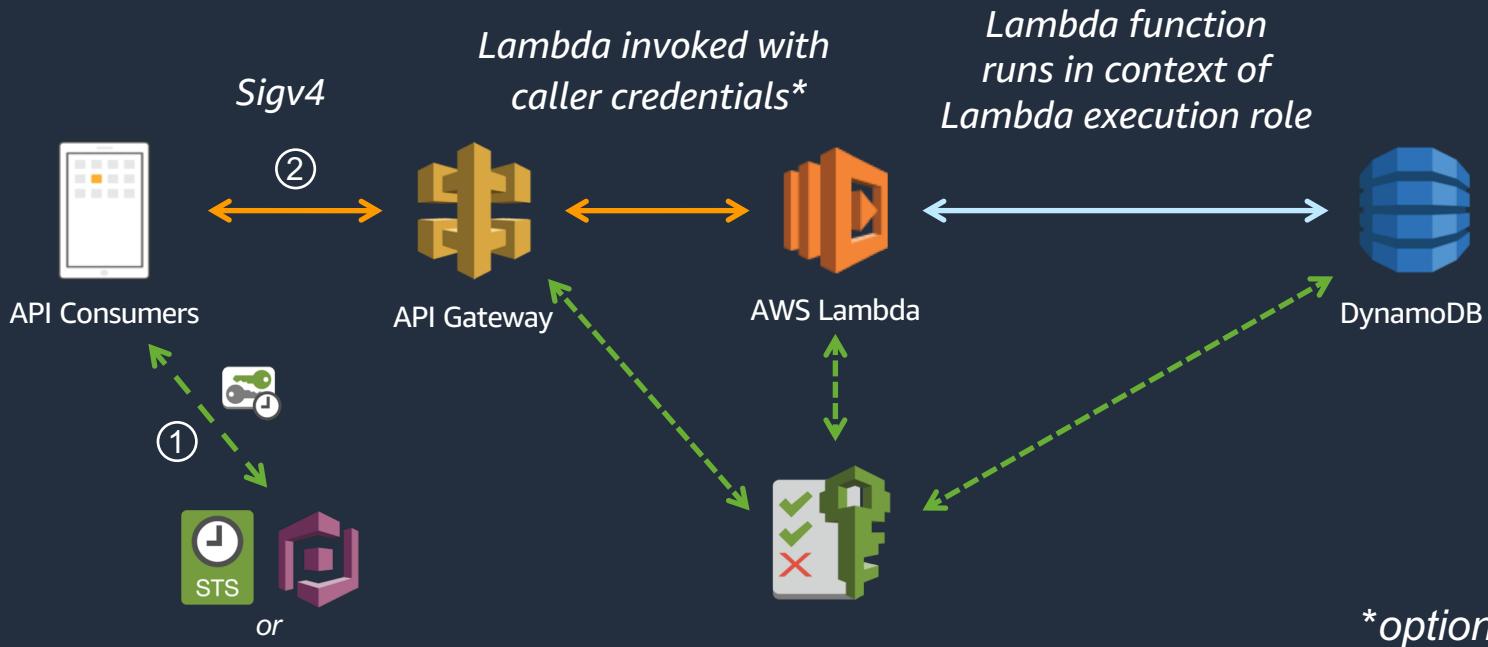
- Authenticate with Cognito
- Pass identity or access token to API Gateway



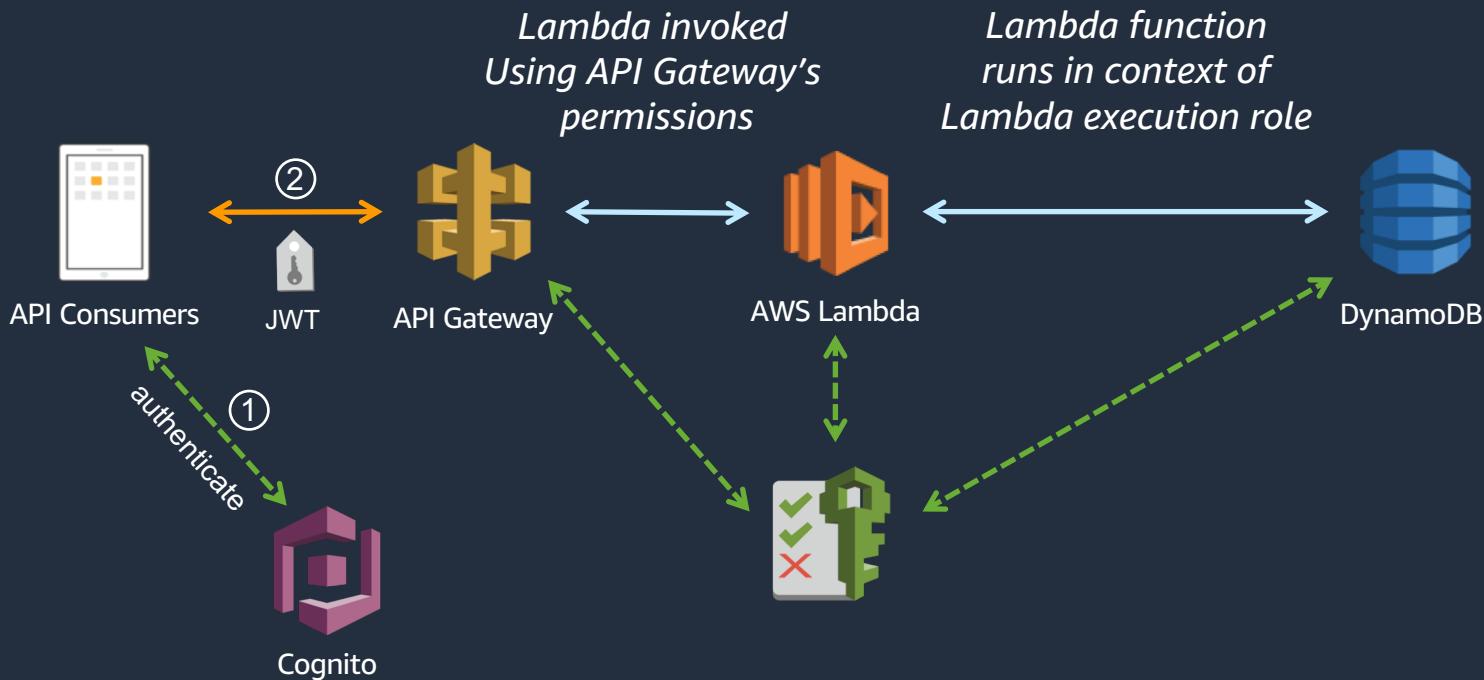
Lambda Authorizer (*formerly Custom Authorizer*)

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

Authorization – AWS Sigv4



Authorization – Cognito User Pool

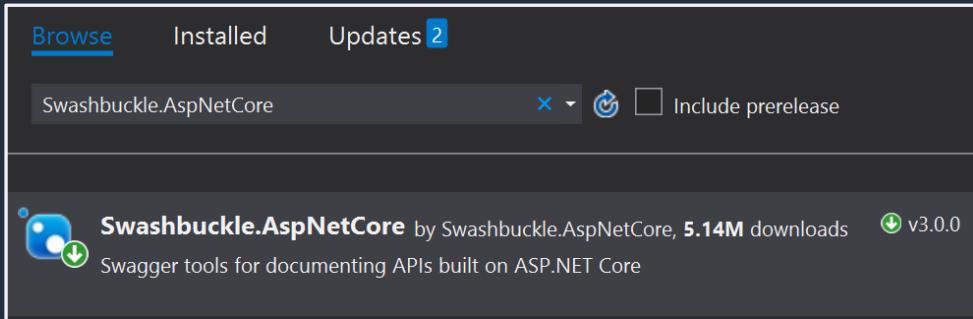


Authorization – Custom Authorizer



Document APIs with Swagger

Generating Swagger with Swashbuckle



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

A screenshot of a browser window displaying a raw JSON document. The URL in the address bar is 'localhost:17955/swagger/v1'. The JSON content is a complex object representing an API specification, including properties for swagger version (2.0), info (version v1, title Hackathon WebAPI Challenge API), paths (including /api/Values), and various operations like GET, POST, and GET with ID.

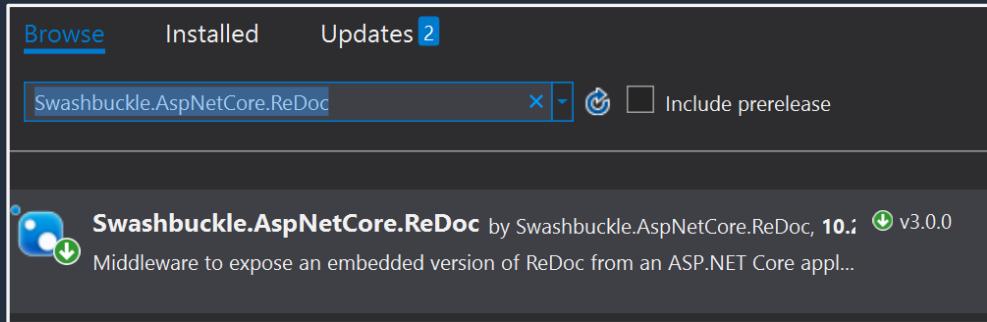
A screenshot of a browser window showing the generated Swagger UI for the 'Hackathon WebAPI Challenge API'. The URL is 'localhost:17955/swagger/inc'. The page has a green header with the word 'swagger'. Below the header, it says 'Select a spec' and 'Hackathon WebAPI Challenge API V1'. The main content area is titled 'Hackathon WebAPI Challenge API' with a link to '/swagger/v1/swagger.json'. It shows a section for 'Values' with three operations listed: a blue 'GET' button for '/api/Values', a blue 'POST' button for '/api/Values', and another blue 'GET' button for '/api/Values/{id}'. The background of the UI is light blue.

Swashbuckle NuGet meta-package

Automatically includes:

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

ReDoc UI with Swashbuckle



The screenshot shows the ReDoc UI for an API. On the left, a sidebar lists methods: 'ApiValuesGet' (GET), 'ApiValuesPost' (POST), 'ApiValuesByIdGet' (GET), 'ApiValuesByIdPut' (PUT, highlighted in grey), and 'ApiValuesByIdDelete' (DELETE). The main area shows the 'ApiValuesByIdPut' endpoint. It includes sections for 'PARAMETERS', 'Path Parameters' (with 'id' as a required integer), 'REQUEST BODY' (with 'string' type), and 'Responses' (with a green box for '200 Success'). To the right, a dark panel shows the 'PUT /api/Values/{id}' method with a sample request body: "string".

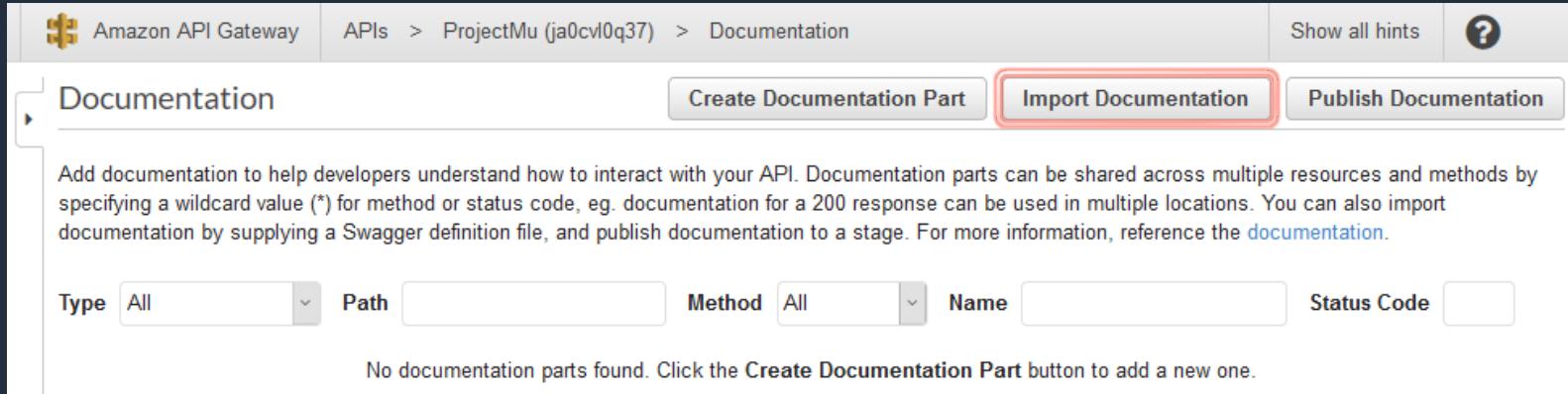
Swashbuckle ReDoc NuGet package

- 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



The screenshot shows the AWS API Gateway Documentation interface. At the top, there's a navigation bar with the Amazon API Gateway logo, followed by 'APIs > ProjectMu (ja0cvl0q37) > Documentation'. To the right are links for 'Show all hints' and a help icon. Below the navigation is a main content area titled 'Documentation' with three buttons: 'Create Documentation Part', 'Import Documentation' (which is highlighted with a red box), and 'Publish Documentation'. A descriptive text block explains how to add documentation parts and import Swagger definitions. Below this are search filters for 'Type' (set to 'All'), 'Path', 'Method' (set to 'All'), 'Name', and 'Status Code'. A message at the bottom states 'No documentation parts found. Click the Create Documentation Part button to add a new one.'

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

Q&A

Name of presenter

Thank you!