REST API Practical

Catch up with one of the most popular standards in software industry

Table of contents

Prerequisites

Software Requirements

- Visual Studio 2022 (17.8+) or VS Code
- .NET 8 SDK
- Git
- Postman or similar API testing tool (optional)

Prerequisites

Basic Knowledge

- C# fundamentals
- HTTP & Web API basics
- JSON
- Authentication/Authorization principles

Prerequisites

Nice to have

- Cloud platform knowledge (Azure/AWS)
- Git workflow
- CI/CD concepts

01

Fundamentals

Do you know

- What is HTTP?
- Overview flow of a HTTP request/response
- Some real examples of HTTP

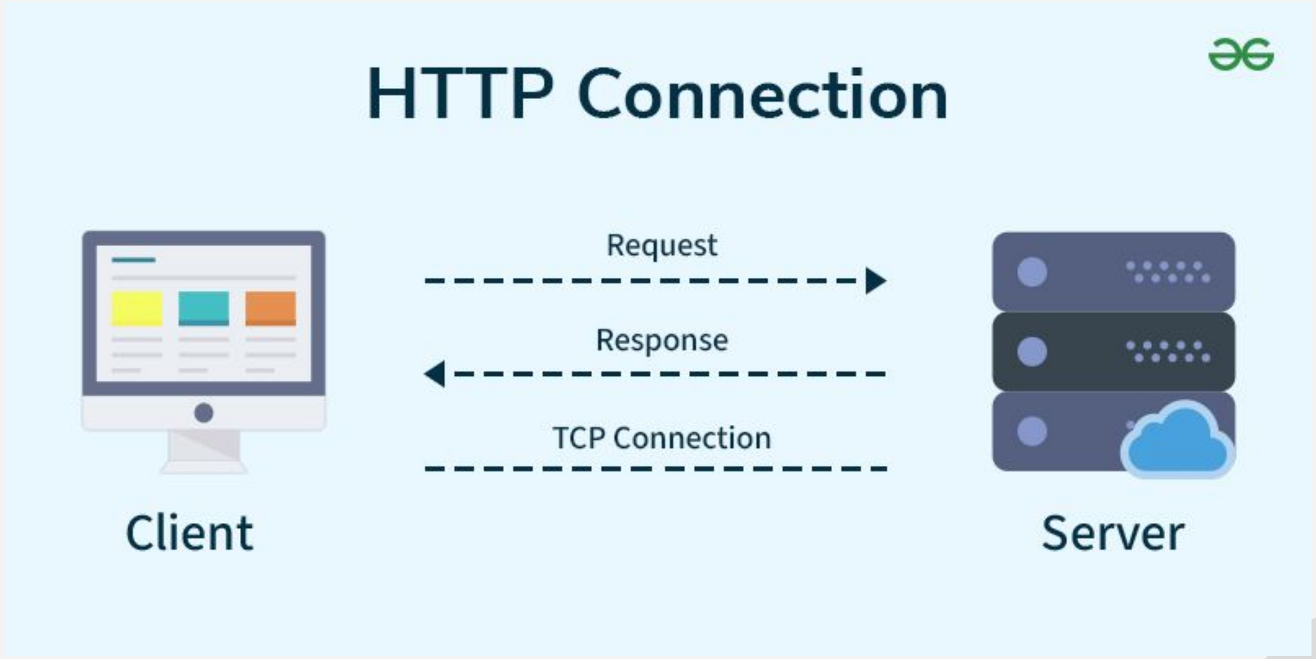
1. Fundamentals

HTTP (Hypertext Transfer Protocol)

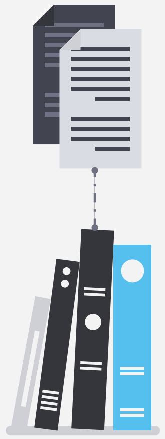
- Client-server communication protocol
- Stateless - each request is independent
- Request-response based
- Foundation of web communication

1. Fundamentals

HTTP (Hypertext Transfer Protocol)



1. Fundamentals



What is REST/RESTful API?

- REST: representational state transfer
- Stateless
- Client-Server
- Resource-Based
- Standard HTTP Methods
- Uniform Interface

1. Fundamentals

RESTful API (HTTP) request - Request URL

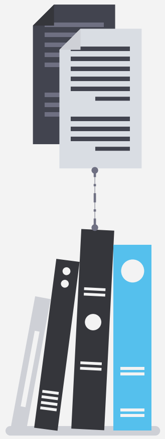
GET https://api.example.com/v1/users/{id}/posts?status=active

- Scheme: http/https
- Host: api.example.com
- Path : v1/users/{id}/posts
  - Path params: {id}
  - Resource: users, posts
- Query params: after "?"
  - status=active

1. Fundamentals

RESTful API (HTTP) request - HTTP methods

- GET: Retrieve data
- POST: Create new resource
- PUT: Update/Replace resource
- PATCH: Partial update
- DELETE: Remove resource



1. Fundamentals

RESTful API (HTTP) request - Headers

- HTTP headers that provide information about the request
- For examples:
  - Content-Type: application/json
  - Authorization: Bearer <token>
  - Accept: application/json
  - Accept-Language: en-US

1. Fundamentals

RESTful API (HTTP) request - Body

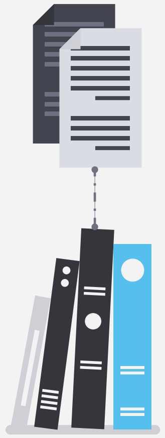
- Part of the request message carrying data from the client to the server.
- Crucial for methods like POST, PUT, and PATCH.

```
{
    "title": "Sample Post",
    "content": "Content here",
    "status": "draft"
}
```

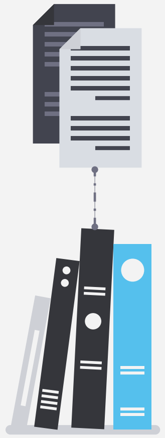
1. Fundamentals

RESTful API (HTTP) response

- Body: response data from API
- Headers: HTTP headers that provide information about the response



1. Fundamentals



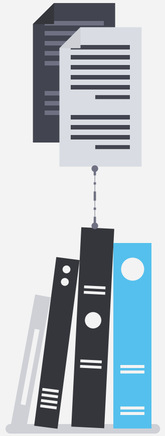
RESTful API (HTTP) response - Status codes

- **2xx** Success:
  - **200** OK, **201** Created, **204** No Content
- **4xx** Client Errors:
  - **400** Bad Request, **401** Unauthorized, **403** Forbidden, **404** Not Found
- **5xx** Server Errors:
  - **500** Internal Server Error, **502** Bad Gateway, **503** Service Unavailable

1. Fundamentals

Resource naming conventions

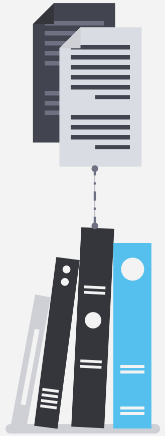
- Use Nouns, Not Verbs
  - ✅ /api/posts
  - ❌ /api/getPost
  - ✅ /api/users/lock
- Plural for Collections
  - ✅ /api/users
  - ❌ /api/user
- Lowercase and Hyphens
  - ✅ /api/blog-posts
  - ❌ /api/blogPosts
  - ❌ /api/blog_posts



1. Fundamentals

Resource relationships

- Nested Resources
  - ✅ /api/posts/{id}/comments
  - ✅ /api/users/{id}/orders
- Query Parameters
  - ✅ /api/posts?category=tech
  - ✅ /api/users?role=admin



1. Fundamentals

Common patterns for reading resources

- **List:** /api/posts
- **Single:** /api/posts/{id}
- **Sub-resources:** /api/posts/{id}/comments
- **Filtering:** /api/posts?status=published
- **Sorting:** /api/posts?sort=date
- **Pagination:** /api/posts?page=1&size=10

Question

- How much percent (%) do you get from all of the previous information?

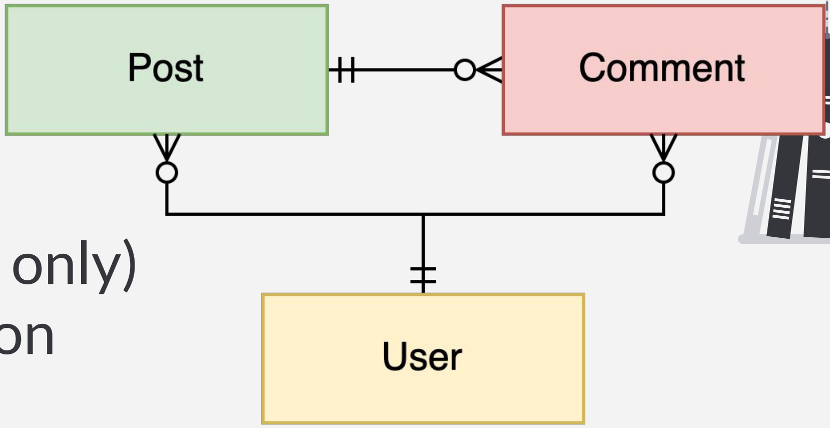
02

Hands-on

2. Hands-on

Project context - Key features

- Resource Management
  - Posts CRUD operations
  - Comments management
  - User management (admin only)
- Authentication & Authorization



2. Hands-on

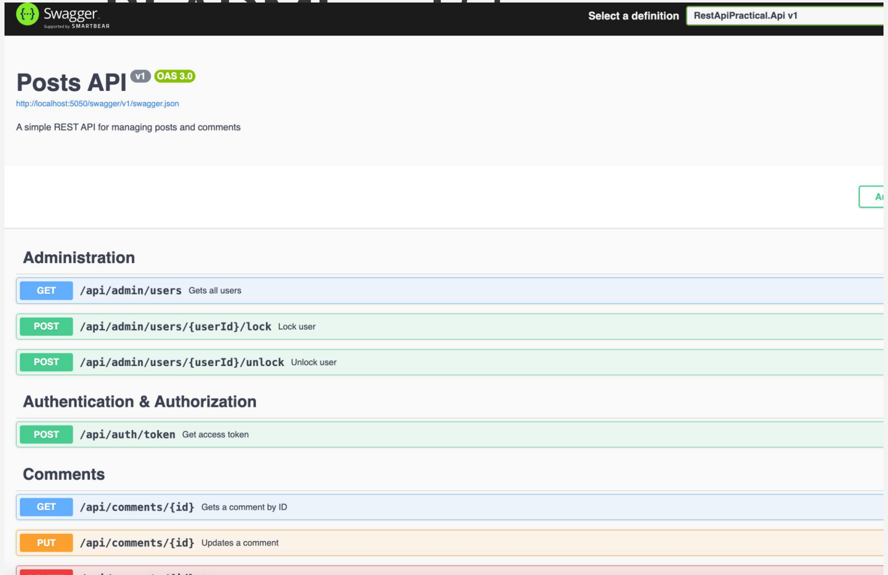
Project context - Architecture

- In-memory database for demonstration
- Minimal API approach
- DTOs for request/response

2. Hands-on

Project setup

- Github: https://github.com/A-Techaholic/rest-api-practical
- Instructio



Posts API v1 OAS 3.0

http://localhost:5050/swagger/v1/swagger.json

A simple REST API for managing posts and comments

Administration

```
GET   /api/admin/users                     Gets all users
POST  /api/admin/users/{userId}/lock       Lock user
POST  /api/admin/users/{userId}/unlock     Unlock user
```

Authentication & Authorization

```
POST  /api/auth/token                      Get access token
```

Comments

```
GET   /api/comments/{id}                   Gets a comment by ID
PUT   /api/comments/{id}                   Updates a comment
```

2. Hands-on

Swagger

- API documentation tool that automatically generates interactive documentation
- Allows testing endpoints directly from the browser interface
- Describes API endpoints, parameters, responses, and authentication methods

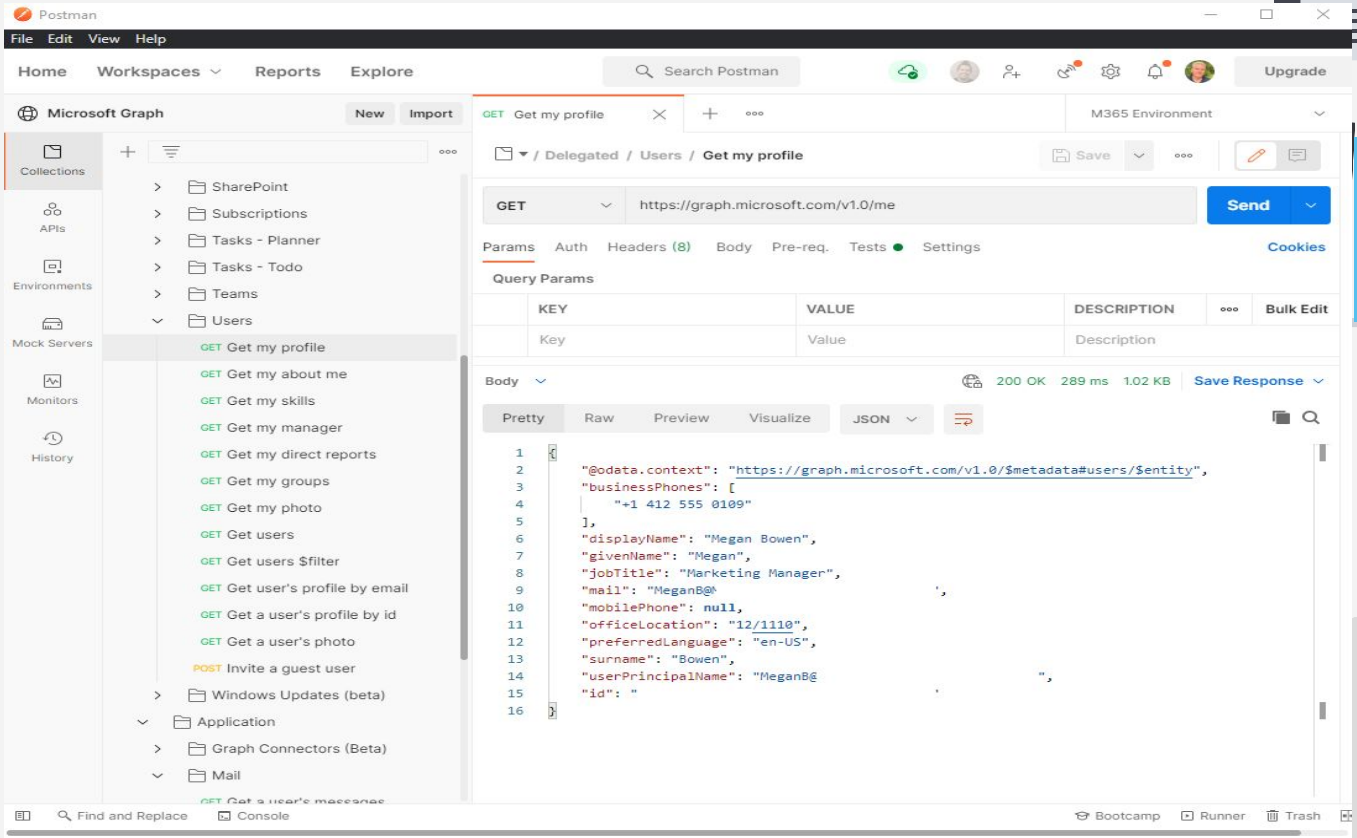
2. Hands-on

Postman

- API testing tool for sending requests and viewing responses
- Allows creating collections of API requests and test suites
- Supports environment variables and request automation
- Enables team collaboration and API documentation sharing
- Provides testing scripts and monitoring capabilities

2. Hands-on

Postman



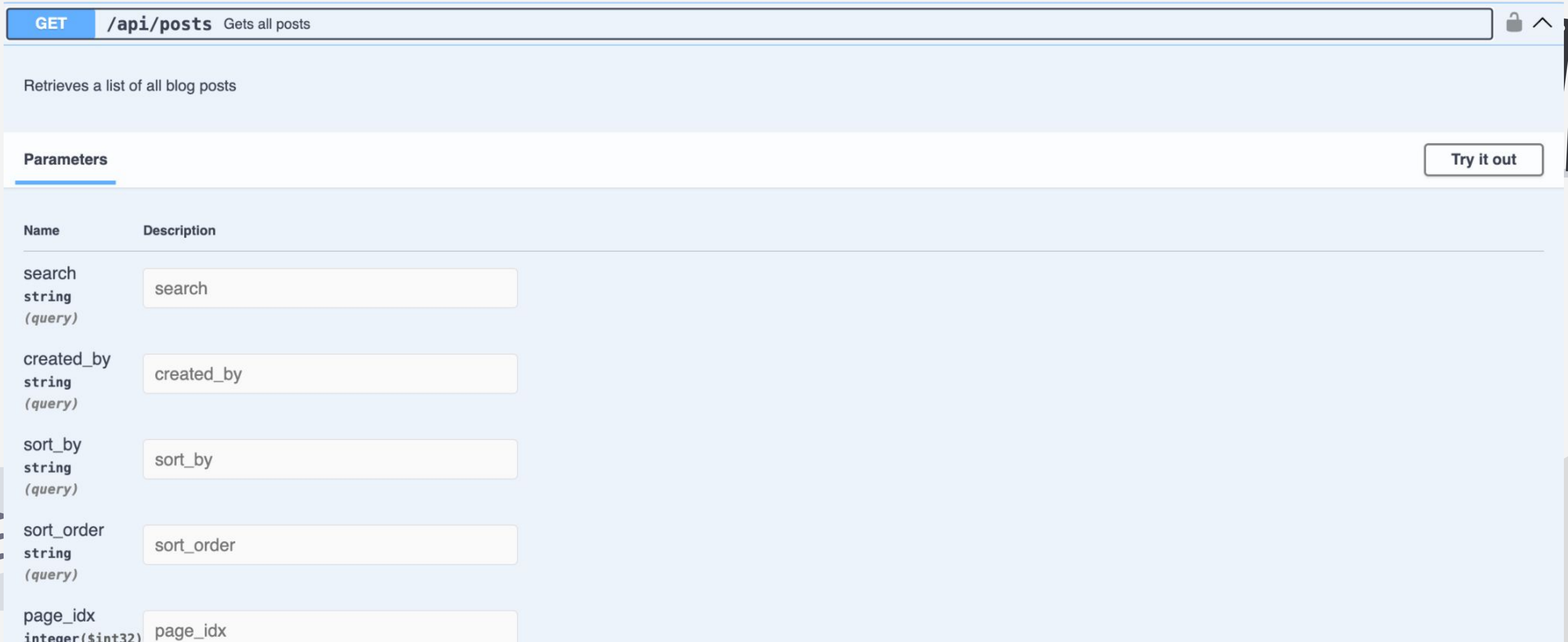
```
{
    "@odata.context": "https://graph.microsoft.com/v1.0/$metadata#users/$entity",
    "businessPhones": [
        "+1 412 555 0109"
    ],
    "displayName": "Megan Bowen",
    "givenName": "Megan",
    "jobTitle": "Marketing Manager",
    "mail": "MeganB@M                         ',
    "mobilePhone": null,
    "officeLocation": "12/1110",
    "preferredLanguage": "en-US",
    "surname": "Bowen",
    "userPrincipalName": "MeganB@                         ",
    "id": "                         '
}
```

03

CRUD Operations

3. CRUD Operations

Reading resources (GET)


GET /api/posts Gets all posts

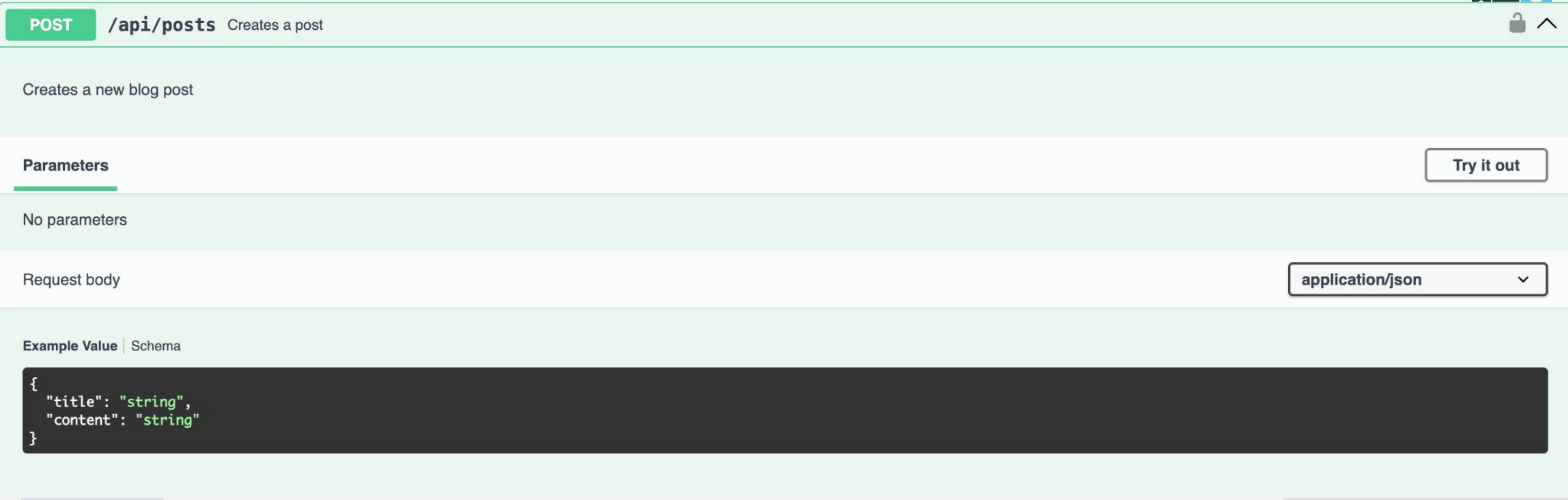
Retrieves a list of all blog posts

Parameters

Name	Description
search string *(query)*	search
created_by string *(query)*	created_by
sort_by string *(query)*	sort_by
sort_order string *(query)*	sort_order
page_idx integer($int32)	page_idx

3. CRUD Operations

Creating resources (POST)



POST `/api/posts` Creates a post

Creates a new blog post

Parameters

No parameters

Request body

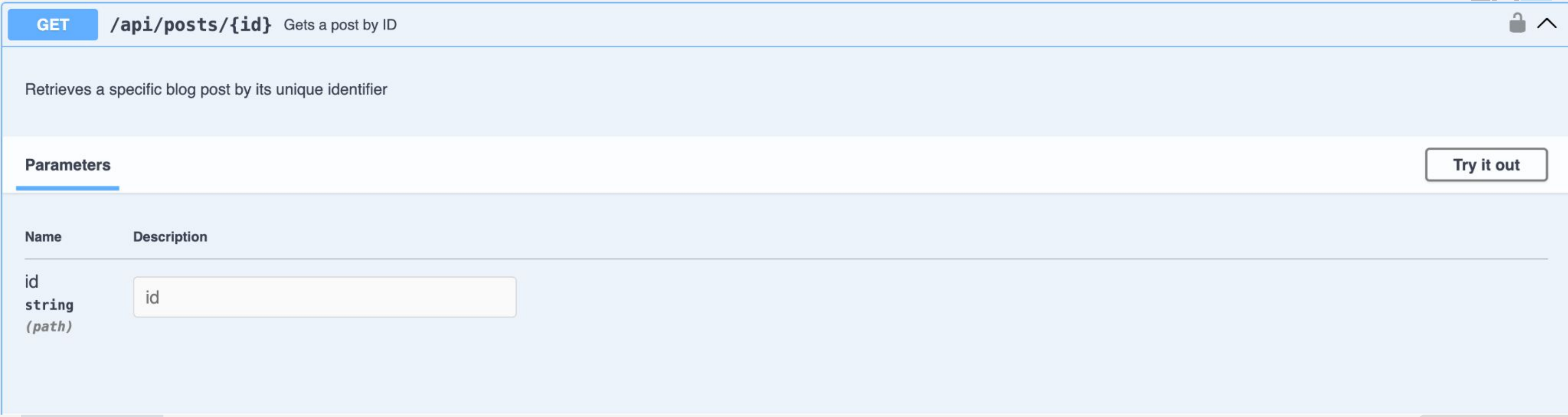
application/json

Example Value | Schema

```
{
  "title": "string",
  "content": "string"
}
```

3. CRUD Operations

Reading resource details (GET)



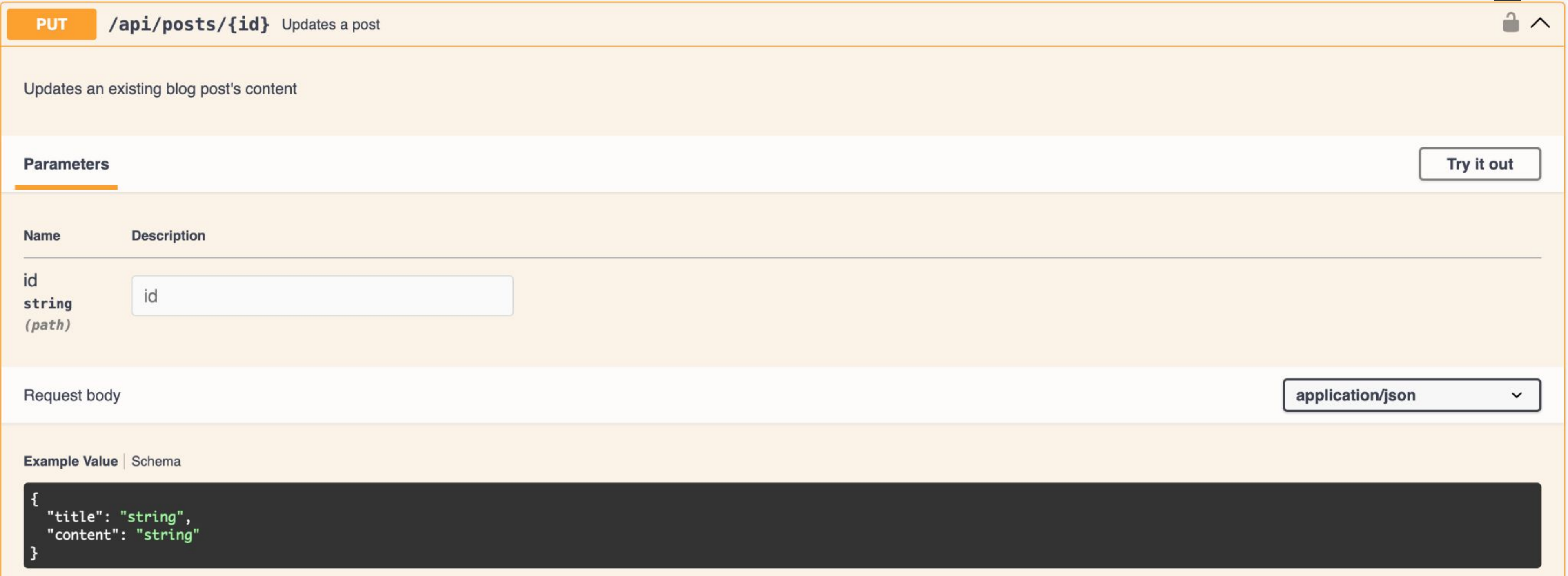
/api/posts/{id} Gets a post by ID

Retrieves a specific blog post by its unique identifier

Name	Description
id string (path)	id

3. CRUD Operations

Updating resources (PUT)



/api/posts/{id} Updates a post

Updates an existing blog post's content

Name	Description
id string *(path)*	id

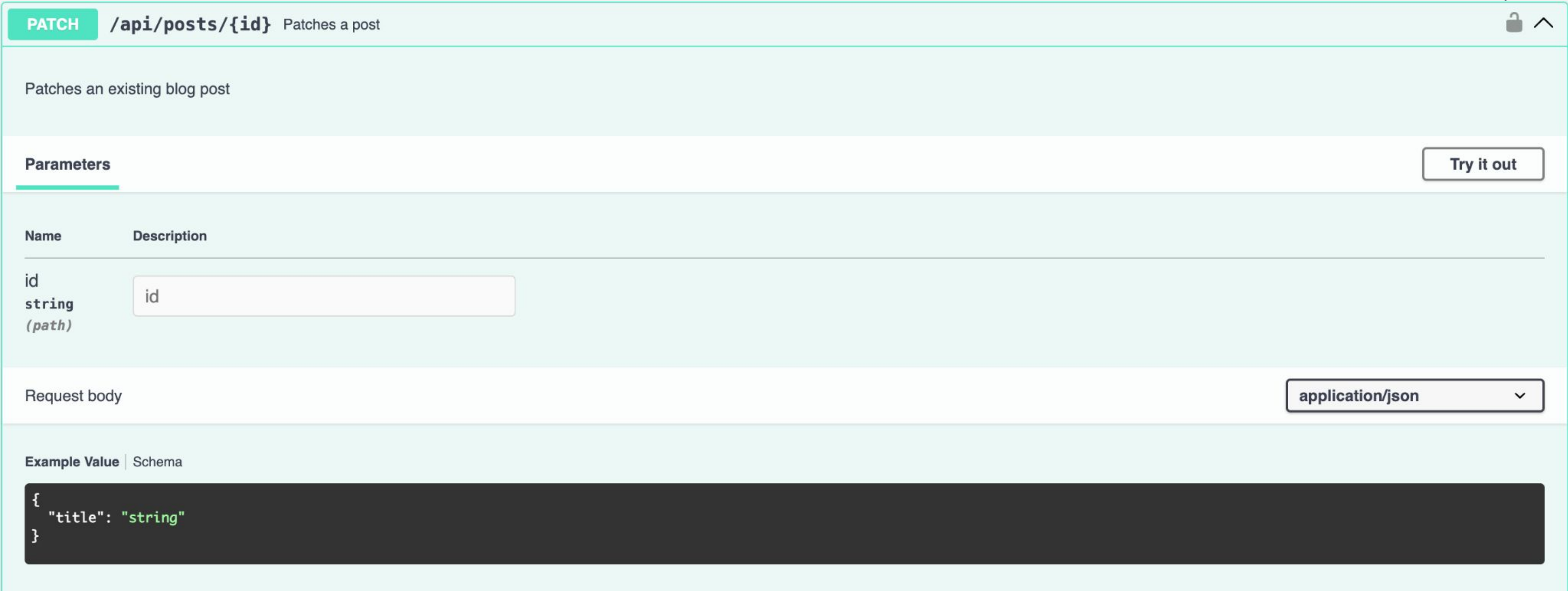
Request body

Example Value | Schema

```
{
  "title": "string",
  "content": "string"
}
```

3. CRUD Operations

Updating resources (PATCH)



PATCH /api/posts/{id} Patches a post

Patches an existing blog post

Name	Description
id string *(path)*	id

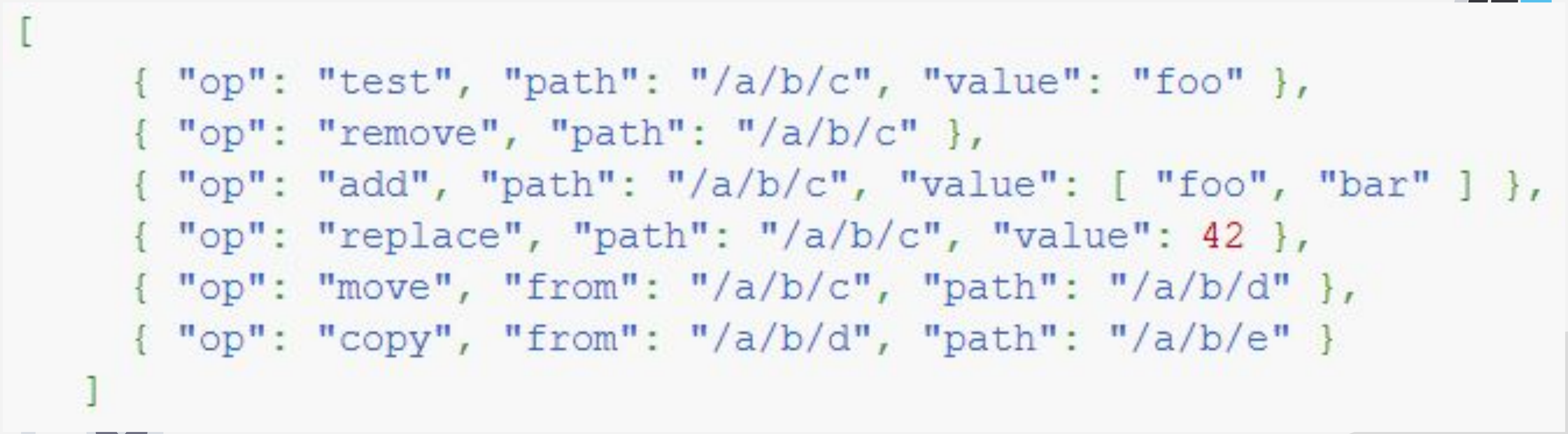
Request body — application/json

Example Value | Schema

```
{
  "title": "string"
}
```

3. CRUD Operations

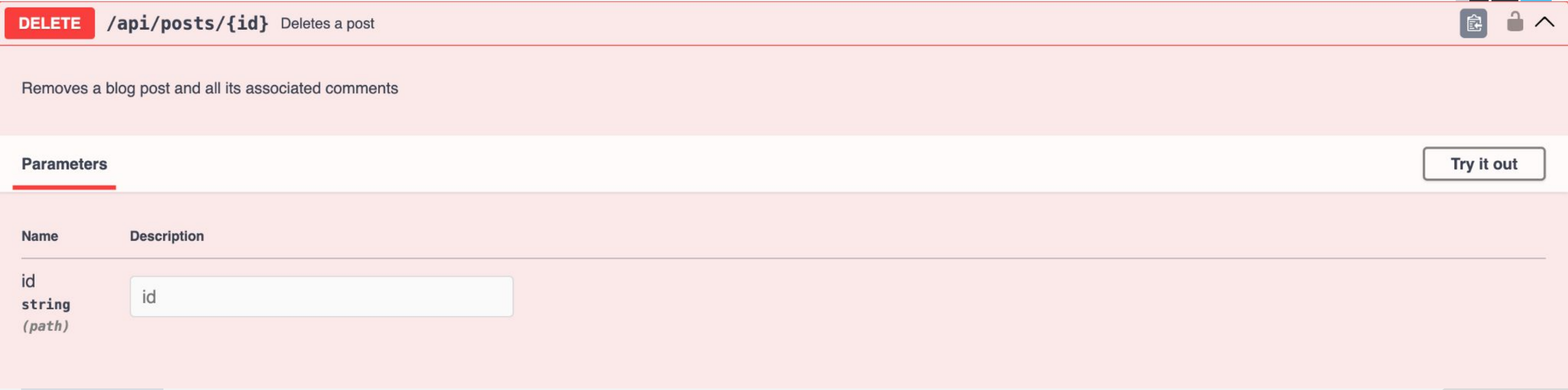
Updating resources (PATCH with JSON Patch)



```
[
     { "op": "test", "path": "/a/b/c", "value": "foo" },
     { "op": "remove", "path": "/a/b/c" },
     { "op": "add", "path": "/a/b/c", "value": [ "foo", "bar" ] },
     { "op": "replace", "path": "/a/b/c", "value": 42 },
     { "op": "move", "from": "/a/b/c", "path": "/a/b/d" },
     { "op": "copy", "from": "/a/b/d", "path": "/a/b/e" }
   ]
```

3. CRUD Operations

Deleting resources (DELETE)



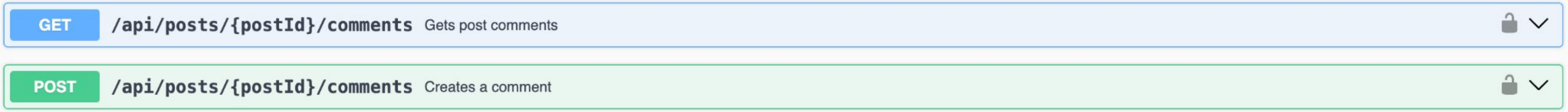
DELETE `/api/posts/{id}` Deletes a post

Removes a blog post and all its associated comments

Name	Description
id `string` *(path)*	id

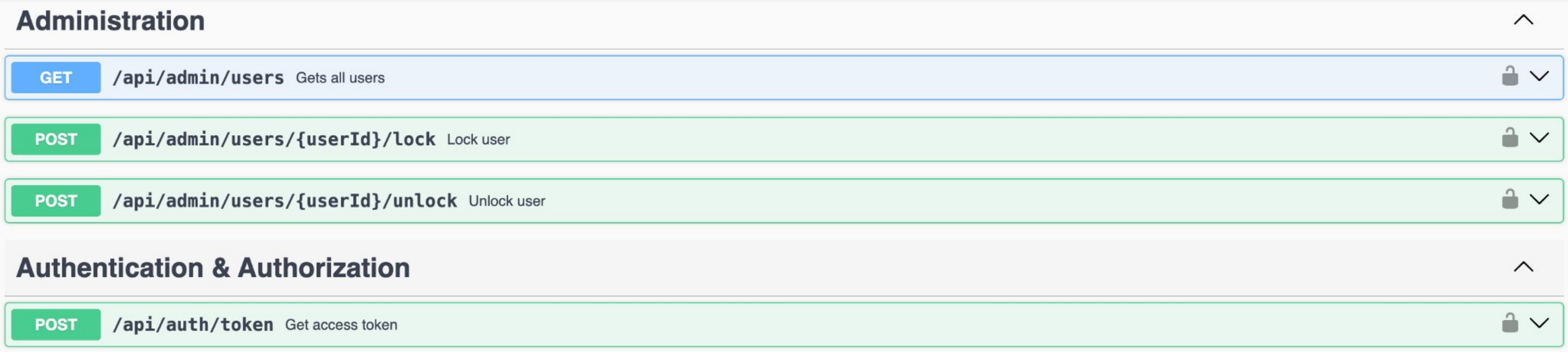
3. CRUD Operations

Others



GET /api/posts/{postId}/comments Gets post comments

POST /api/posts/{postId}/comments Creates a comment



Administration

GET /api/admin/users Gets all users

POST /api/admin/users/{userId}/lock Lock user

POST /api/admin/users/{userId}/unlock Unlock user

Authentication & Authorization

POST /api/auth/token Get access token

3. CRUD Operations

Real-world API examples

- Youtube: Videos | YouTube Data API | Google for Developers
- Wordpress: Reference – REST API Handbook | Developer.WordPress.org
- GitHub Pull requests: REST API endpoints for pull requests - GitHub Docs

Question

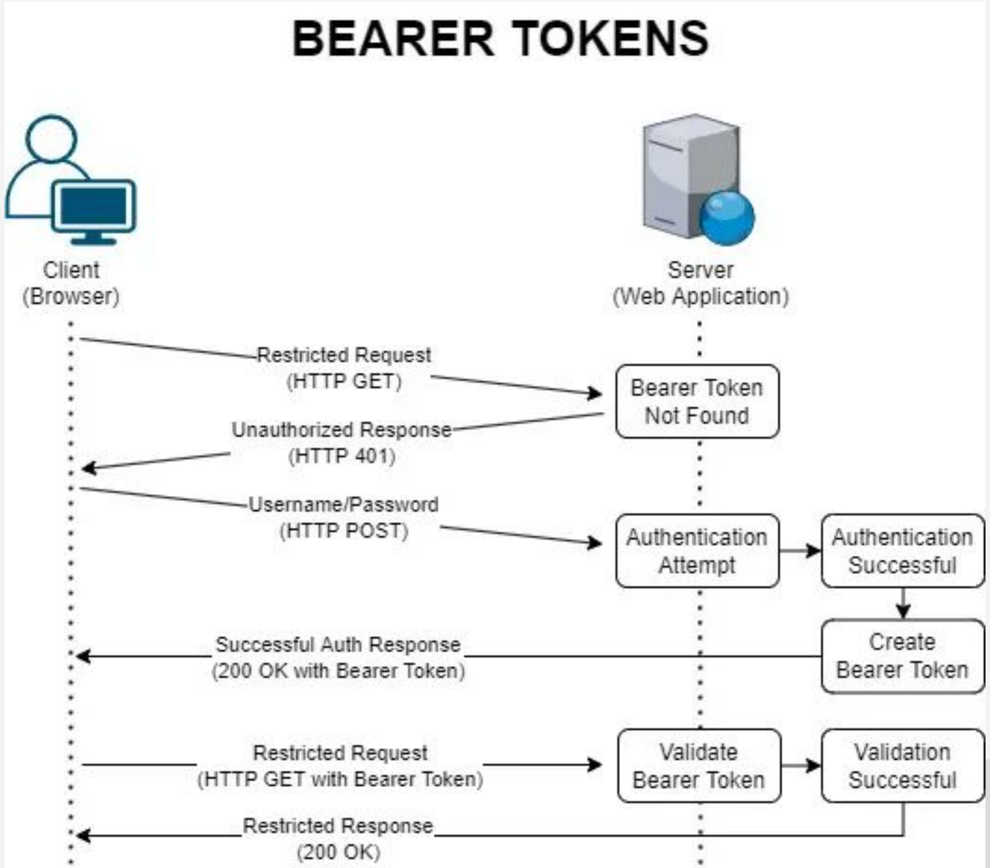
- How do you secure your API?

04

Authentication & Authorization

4. Authentication & Authorization

Bearer (Token) authentication


BEARER TOKENS
Client (Browser)
Server (Web Application)
Restricted Request (HTTP GET)
Bearer Token Not Found
Unauthorized Response (HTTP 401)
Username/Password (HTTP POST)
Authentication Attempt
Authentication Successful
Create Bearer Token
Successful Auth Response (200 OK with Bearer Token)
Restricted Request (HTTP GET with Bearer Token)
Validate Bearer Token
Validation Successful
Restricted Response (200 OK)

4. Authentication & Authorization

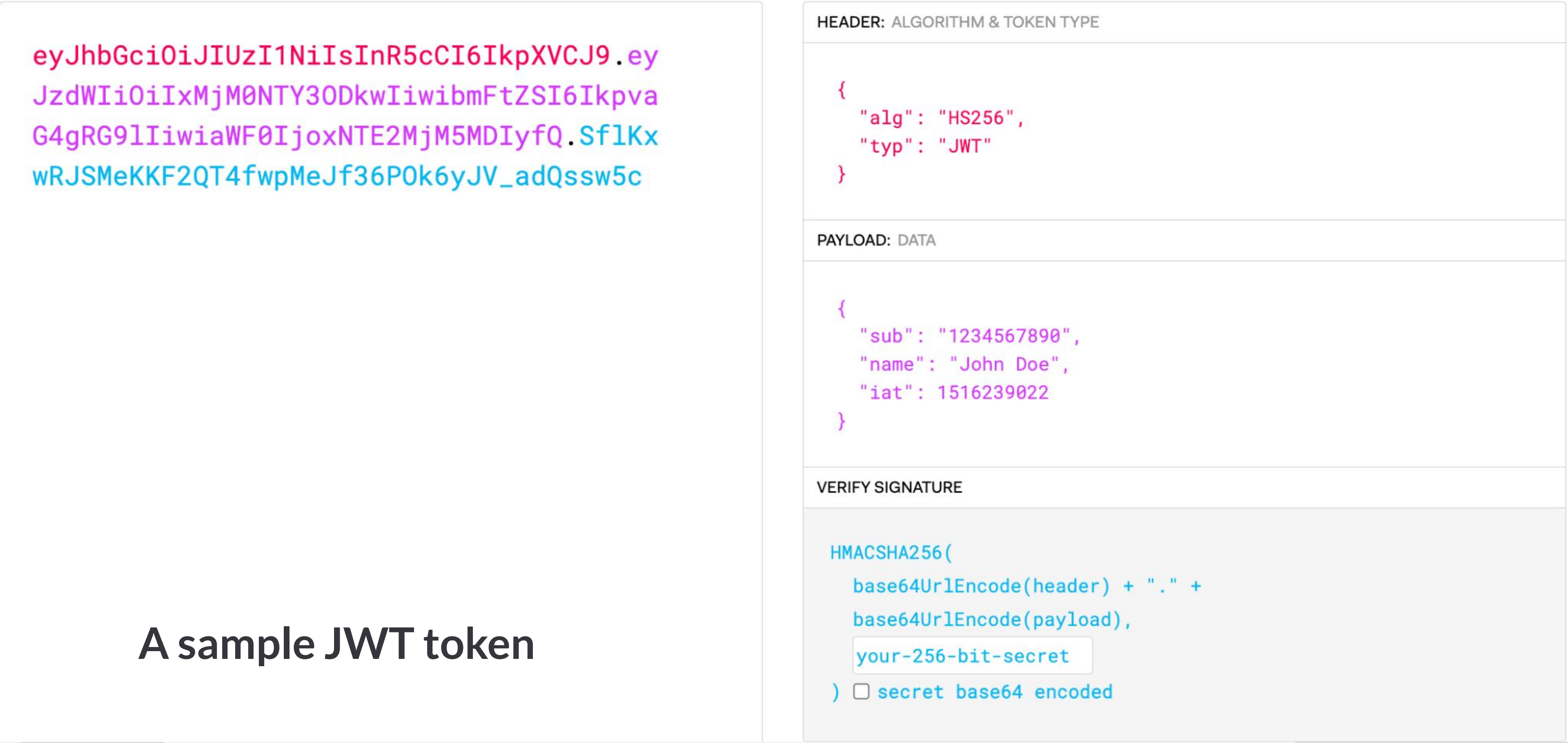
Token vs Cookies authentication

	Token authentication	**Cookies authentication**
Storage location	Client-side	Browser cookies
State management	Stateless	Stateful (session)
Cross-domain	Simple	Need to deal with CORS
Mobile support	Good for mobile app	Limited supports
Data and expiration	Self-contained info (JWT)	Server-controlled
API Integration	Good for API	Less suitable

4. Authentication & Authorization

JWT token

- Token: is simply a piece of data that represents authorization or authentication.
- JWT token: A self-contained token that carries its own information.



```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
JzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpva
G4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKx
wRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

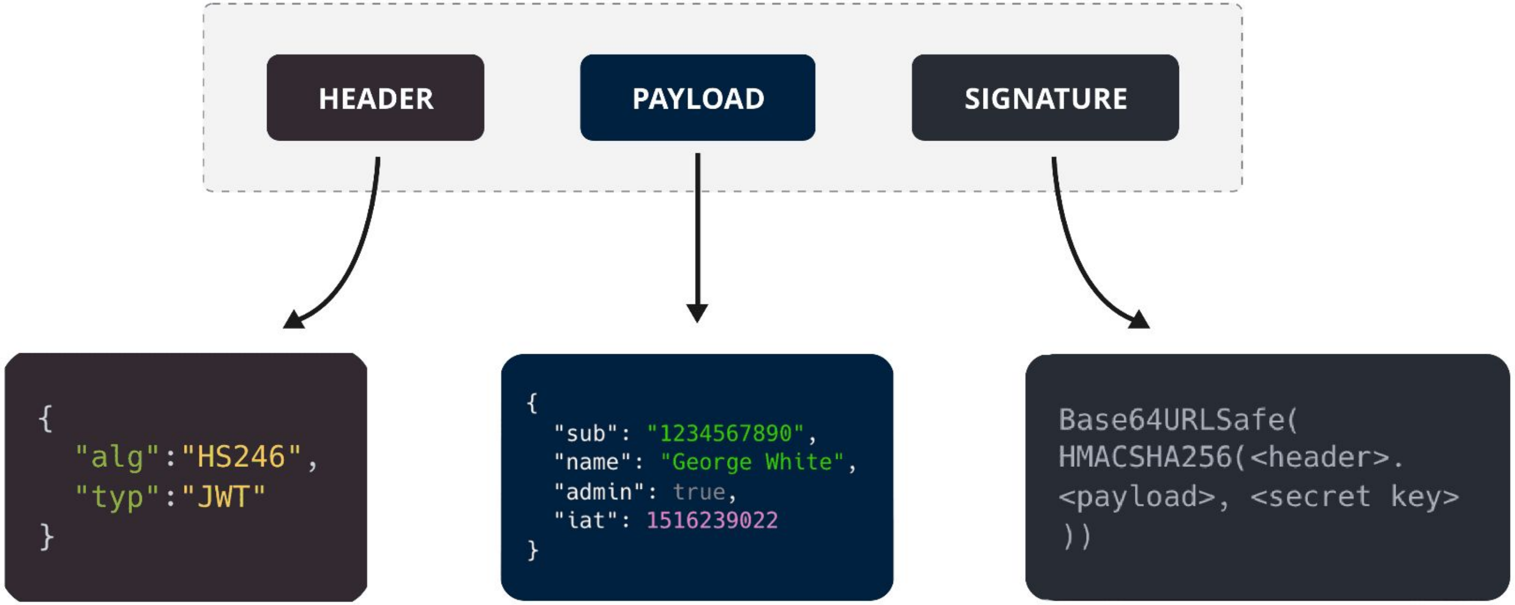
```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

A sample JWT token

Structure of a JSON Web Token (JWT)



```
{
  "alg":"HS246",
  "typ":"JWT"
}
```

```
{
  "sub": "1234567890",
  "name": "George White",
  "admin": true,
  "iat": 1516239022
}
```

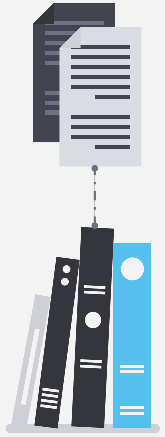
```
Base64URLSafe(
HMACSHA256(<header>.
<payload>, <secret key>
))
```

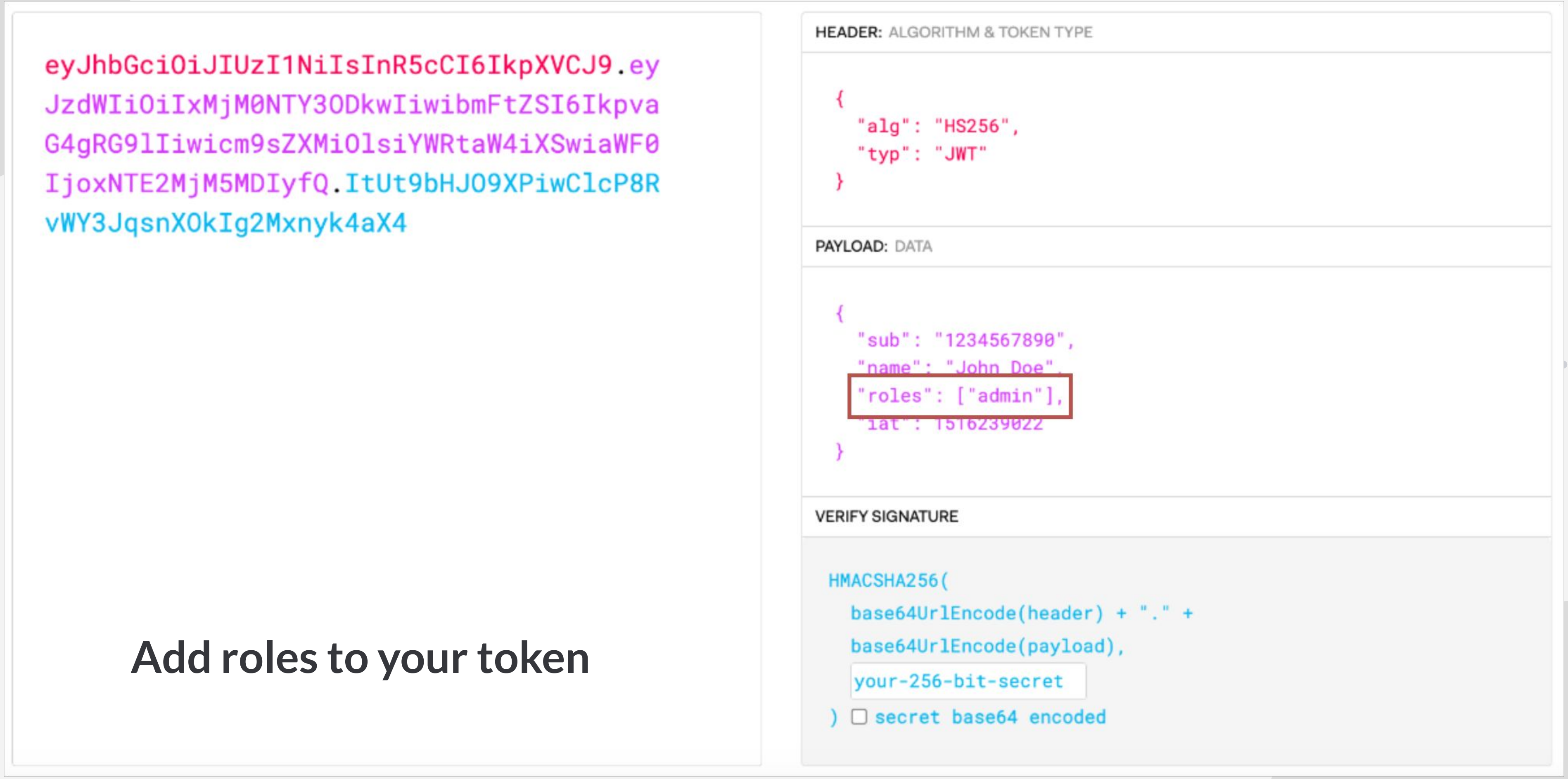
4. Authentication & Authorization

Role-based access control

- Users: Individual accounts
- Roles: Job functions/titles

→ Limit resource access by user roles





```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
JzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikpva
G4gRG9lIiwicm9sZXMiOlsiYWRtaW4iXSwiaWF0
IjoxNTE2MjM5MDIyfQ.ItUt9bHJO9XPiwClcP8R
vWY3JqsnXOkIg2Mxnyk4aX4
```

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "roles": ["admin"],
  "iat": 1516239022
}
```

VERIFY SIGNATURE

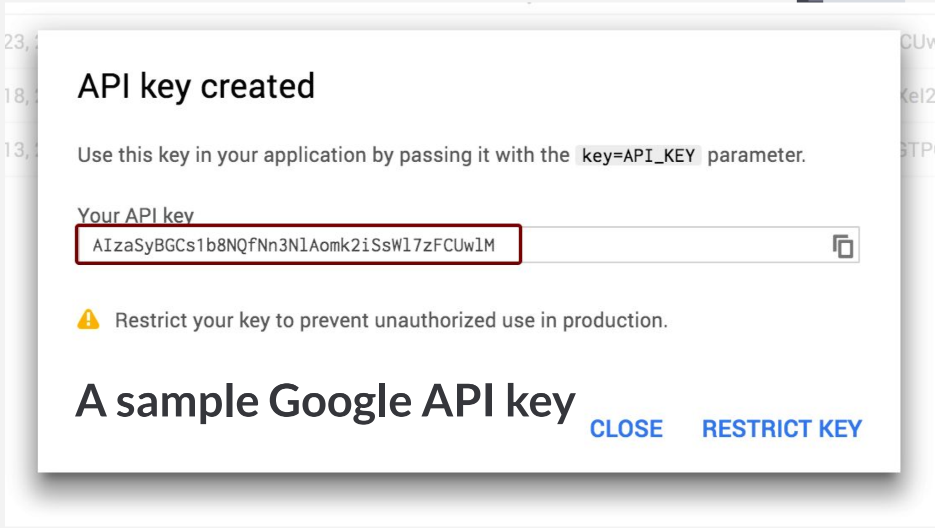
```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

Add roles to your token

4. Authentication & Authorization

API keys

- Common Uses
  - Third-party API access
  - Service authentication
  - Usage tracking
  - Rate limiting
- Limitations
  - Not for user authentication
  - Less secure than OAuth/JWT
  - No built-in expiration
  - No standard format


API key created

Use this key in your application by passing it with the `key=API_KEY` parameter.

Your API key

```
AIzaSyBGCs1b8NQfNn3NlAomk2iSsWl7zFCUwlM
```

Restrict your key to prevent unauthorized use in production.

A sample Google API key

4. Authentication & Authorization

OAuth 2.0

- Authorization framework that enables apps to access user data without passwords
- Uses tokens instead of credentials for secure delegated access
- Involves four roles: Resource Owner, Client, Auth Server, Resource Server
- Supports different flows (grant types) for different use cases
- Issues access tokens and refresh tokens for secure API access

Question

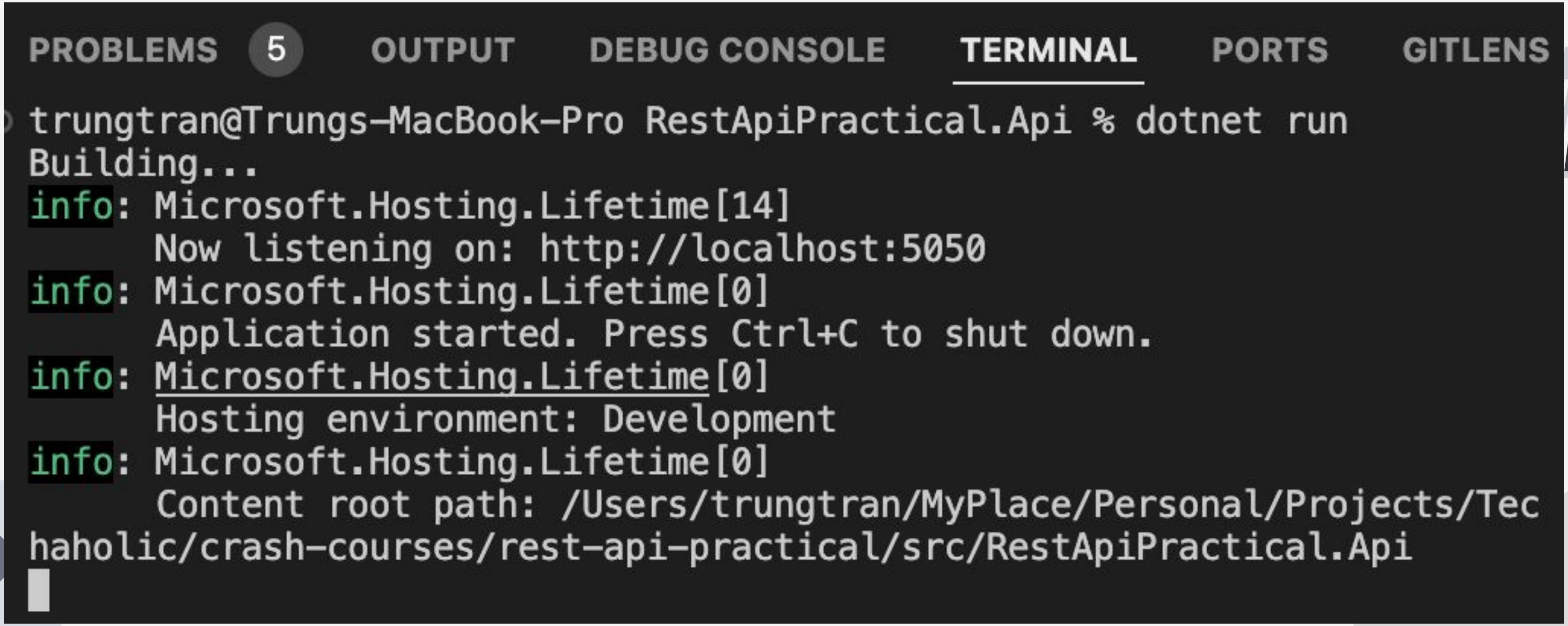
- How do you share your API to others?

05

Deployment

5. Deployment

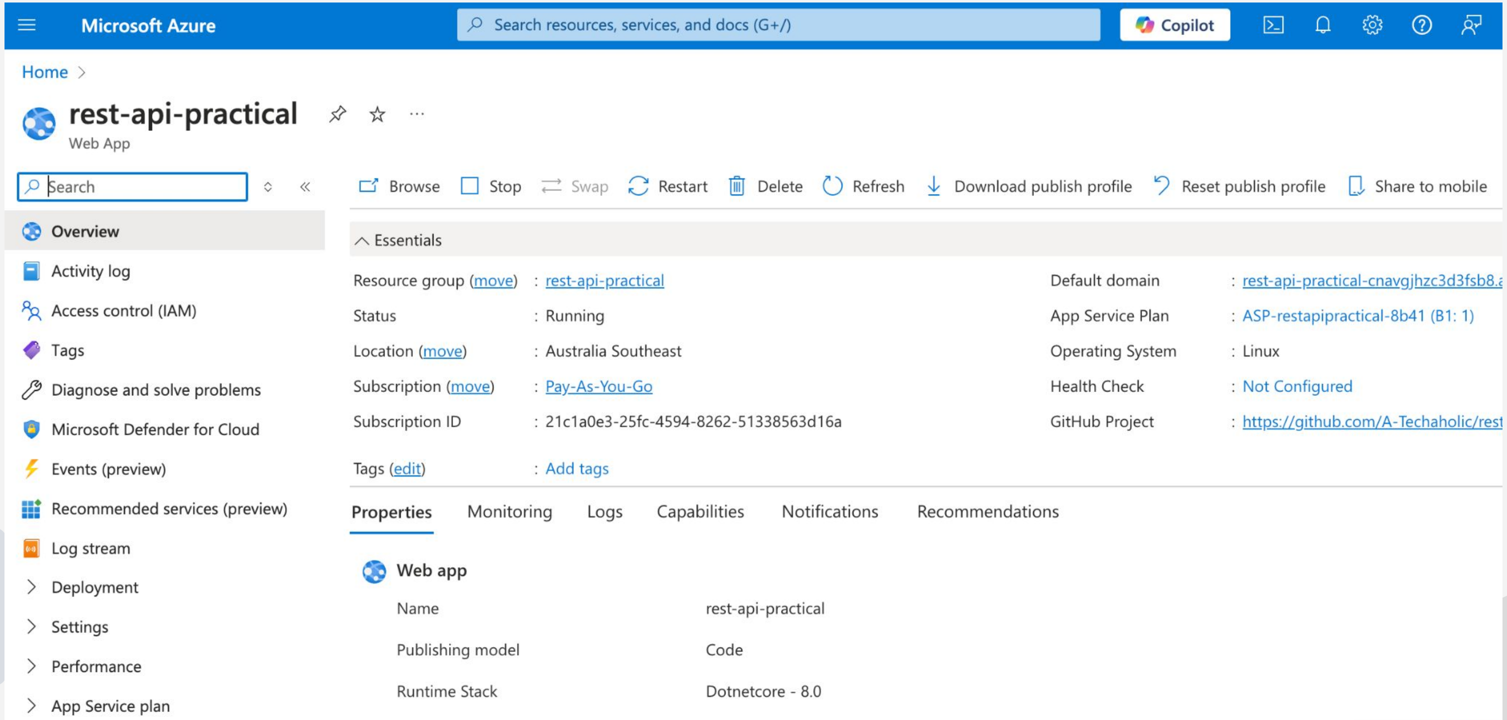
Local/Development



```
trungtran@Trungs-MacBook-Pro RestApiPractical.Api % dotnet run
Building...
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5050
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /Users/trungtran/MyPlace/Personal/Projects/Tec
haholic/crash-courses/rest-api-practical/src/RestApiPractical.Api
```

5. Deployment

Cloud hosting



rest-api-practical
Web App

Essentials

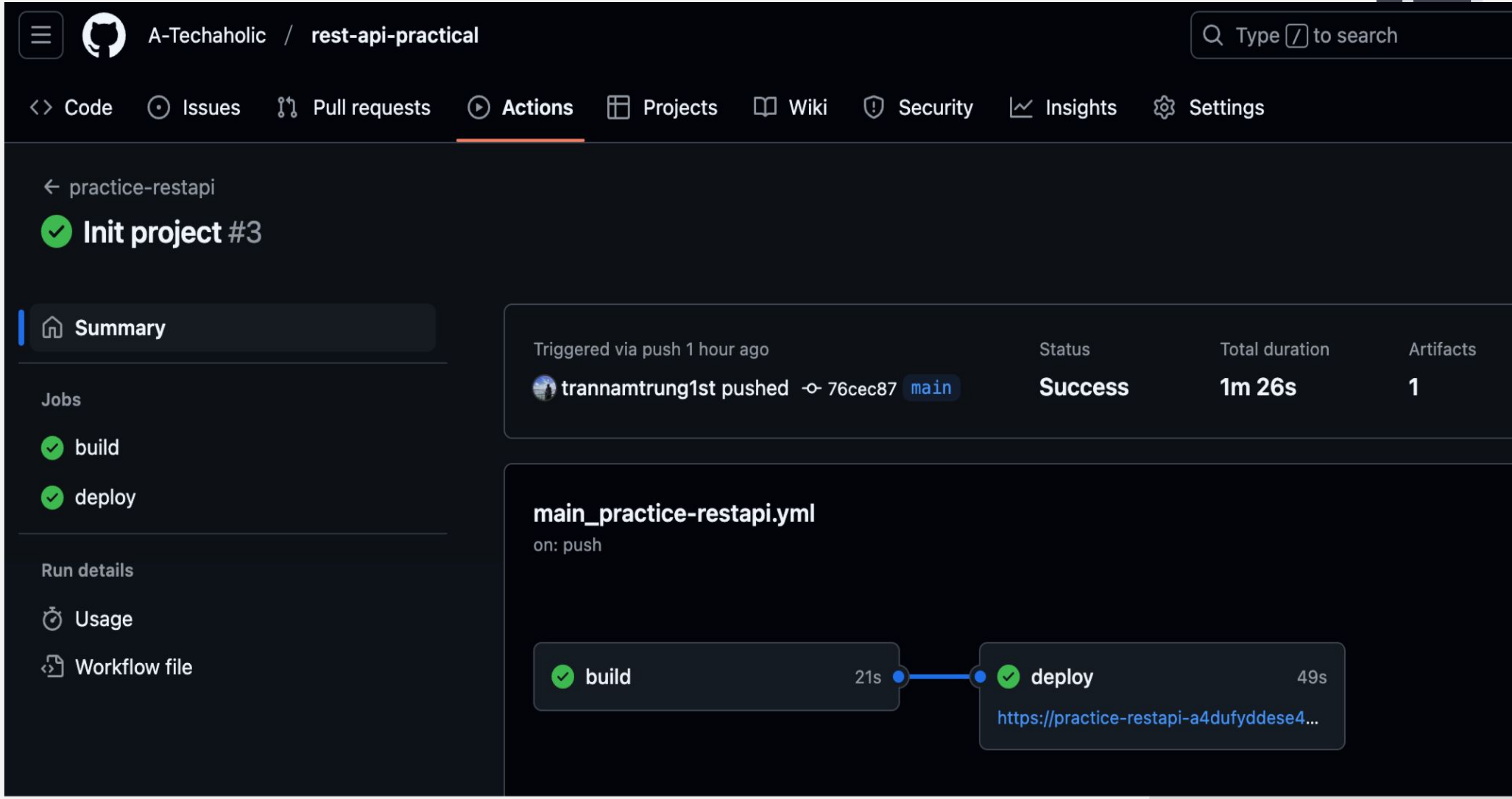
Resource group (move)	: rest-api-practical	Default domain	: rest-api-practical-cnavgjhzc3d3fsb8.
Status	: Running	App Service Plan	: ASP-restapipractical-8b41 (B1: 1)
Location (move)	: Australia Southeast	Operating System	: Linux
Subscription (move)	: Pay-As-You-Go	Health Check	: Not Configured
Subscription ID	: 21c1a0e3-25fc-4594-8262-51338563d16a	GitHub Project	: https://github.com/A-Techaholic/rest
Tags (edit)	: Add tags		

Web app

Name	rest-api-practical
Publishing model	Code
Runtime Stack	Dotnetcore - 8.0

5. Deployment

CI/CD



← practice-restapi

Init project #3

Summary

Jobs

build

deploy

Run details

Usage

Workflow file

Triggered via push 1 hour ago

trannamtrung1st pushed 76cec87 main

Status: Success

Total duration: 1m 26s

Artifacts: 1

main_practice-restapi.yml

on: push

build 21s

deploy 49s

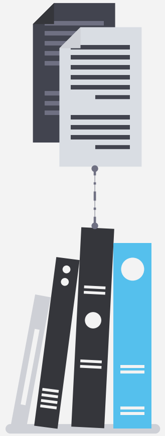
https://practice-restapi-a4dufyddese4...

06

Advanced Topics

6. Advanced Topics

- Versioning strategies
- Rate limiting
- Caching
- Documentation (Swagger/OpenAPI)
- Error handling

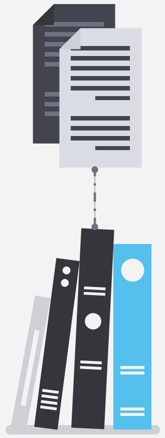


07

Best Practices

7. Best Practices

- Security considerations: OWASP API Security Project
- Performance optimization
- Testing strategies
- Monitoring and logging



08

Summary, QnA

8. Summary, QnA

Summary

- RESTful API Fundamentals
- CRUD Operations
- Authentication & Authorization
- Deployment
- Advanced Topics
- Best Practices

References

- REST API Tutorial
- REST - Wikipedia
- HTTP Status Codes Glossary - WebFX
- JWT
- OAuth 2.0
- Role-Based Access Control - Auth0
- Swagger Documentation | Swagger Docs
- Postman Learning Center
- Minimal APIs quick reference | Microsoft Learn
- ASP.NET Core security topics | Microsoft Learn
- OWASP API Security Project
- REST API Design - Resource Modeling | Thoughtworks

Thanks!

Do you have any questions?

trannamtrung1st@gmail.com

https://www.linkedin.com/in/trung-tran99/