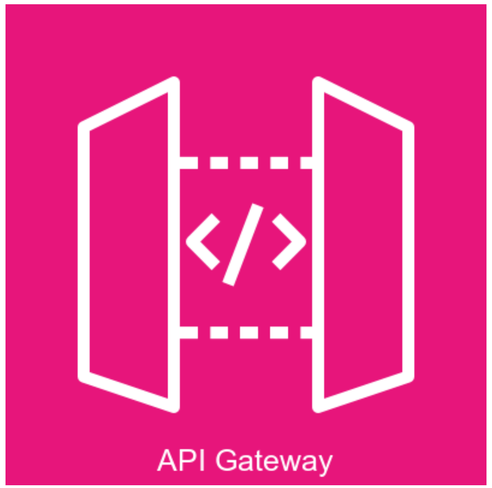
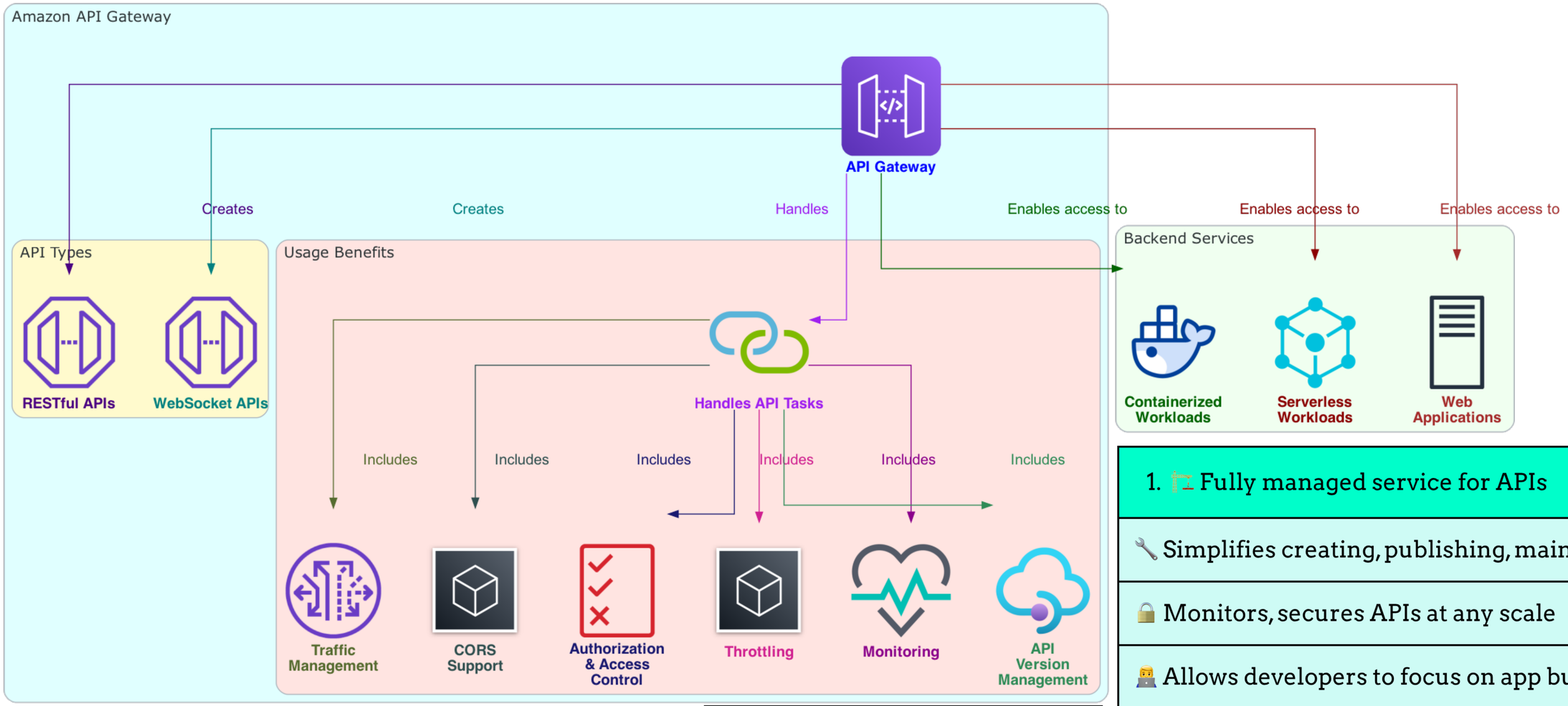



# Amazon API Gateway

# Table of Contents

# What is Amazon API Gateway?

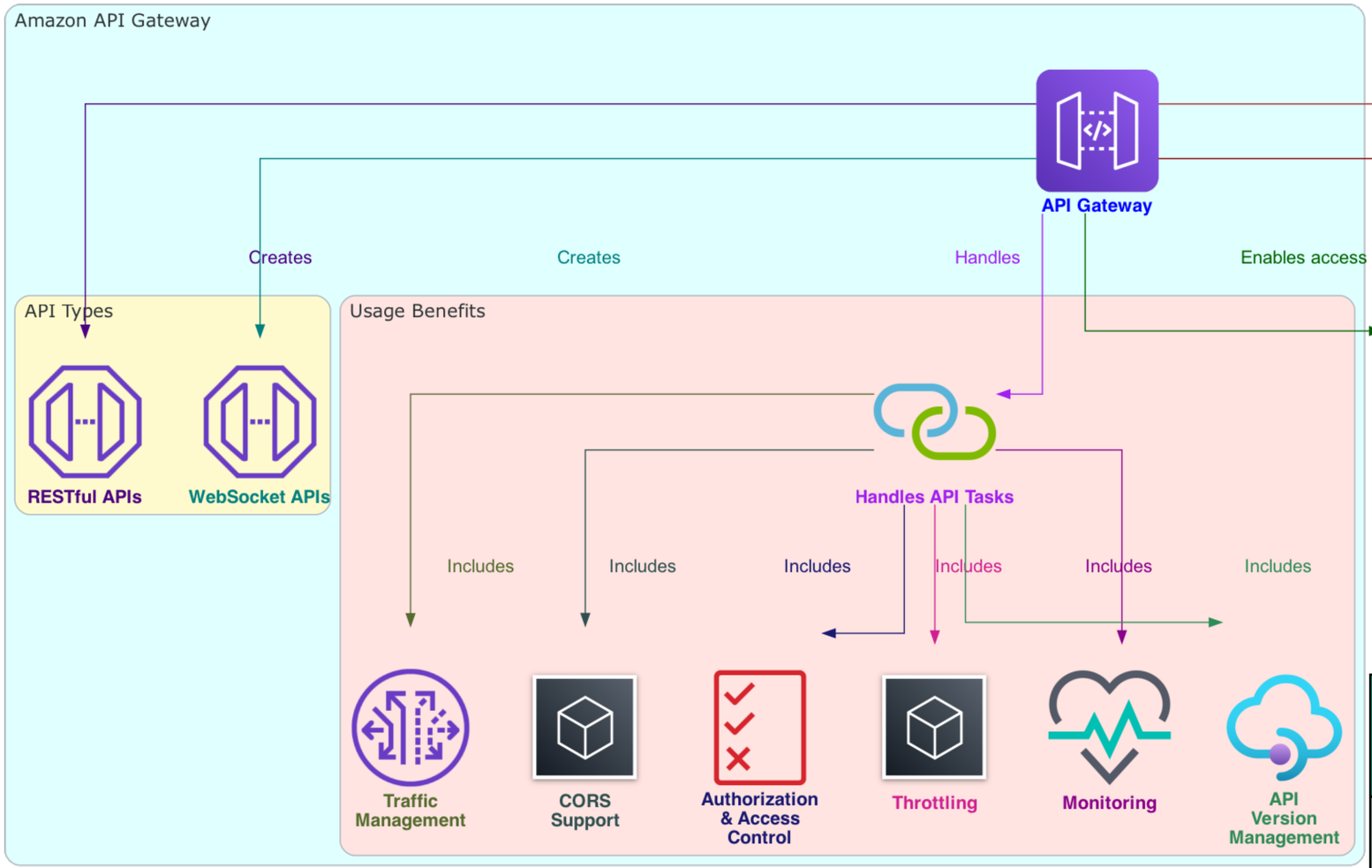



1. 🏗️ Fully managed service for APIs
- 🔧 Simplifies creating, publishing, maintaining APIs
- 🔒 Monitors, secures APIs at any scale
- 👷 Allows developers to focus on app building

2. 🌐 Supports multiple API types
- 🎛️ RESTful APIs
- 🔌 WebSocket APIs
- 🎨 Flexibility in API architecture design

3. 🔄 Enables real-time communication apps
- 🌐 Supports WebSocket APIs
- ↔️ Bidirectional client-server communication

# What is Amazon API Gateway?

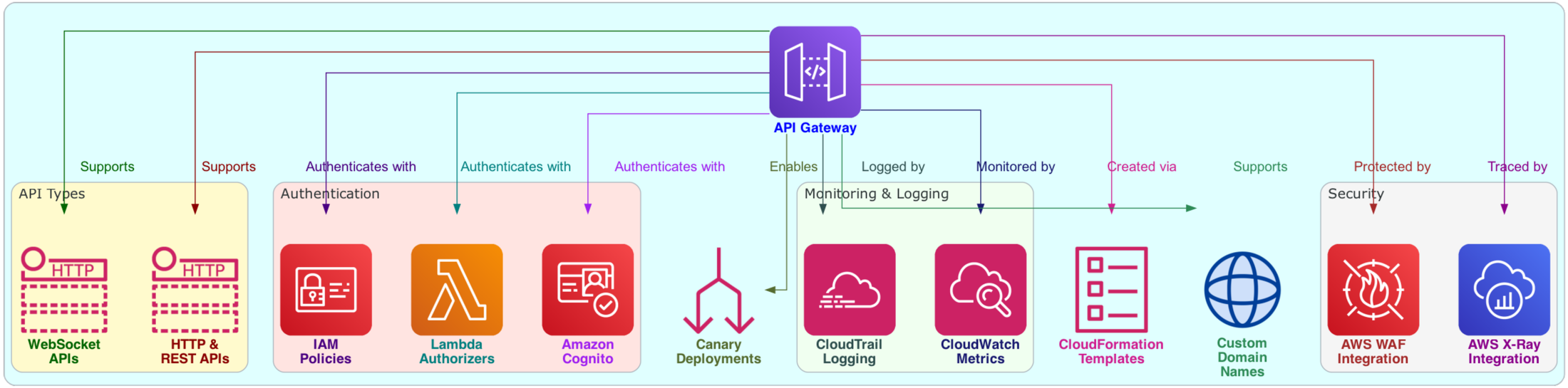



4. 📦 Integrates with various backends
   - 🐋 Containerized workloads
   - ⚡ Serverless architectures
   - 🌎 Traditional web applications

5. 🚦 Handles essential API management tasks
   - 🚥 Traffic management
   - 🤝 CORS support
   - 🔐 Authorization and access control
   - ⏰ Request throttling
   - 📊 API performance and usage monitoring

6. 📈 Cost-effective tiered pricing model
   - 💰 Reduces costs as API usage grows
   - 💸 Pay for API calls and data transfer
   - 🎯 Suitable for varying usage patterns

# Features of API Gateway




1. 🌐 Supports multiple API types

🔄 Stateful (WebSocket) APIs

🔍 Stateless (HTTP and REST) APIs

2. 🔐 Powerful, flexible authentication

🆔 AWS IAM policies

🔑 Lambda authorizer functions

👥 Amazon Cognito user pools

3. 🚀 Canary release deployments

🎯 Safely roll out changes

4. 📝 Logging and monitoring

🔍 CloudTrail logging

📊 CloudWatch access and execution logging

⏰ CloudWatch alarms

5. 🏗️ AWS CloudFormation templates

💡 Enable API creation

6. 🌍 Custom domain names support

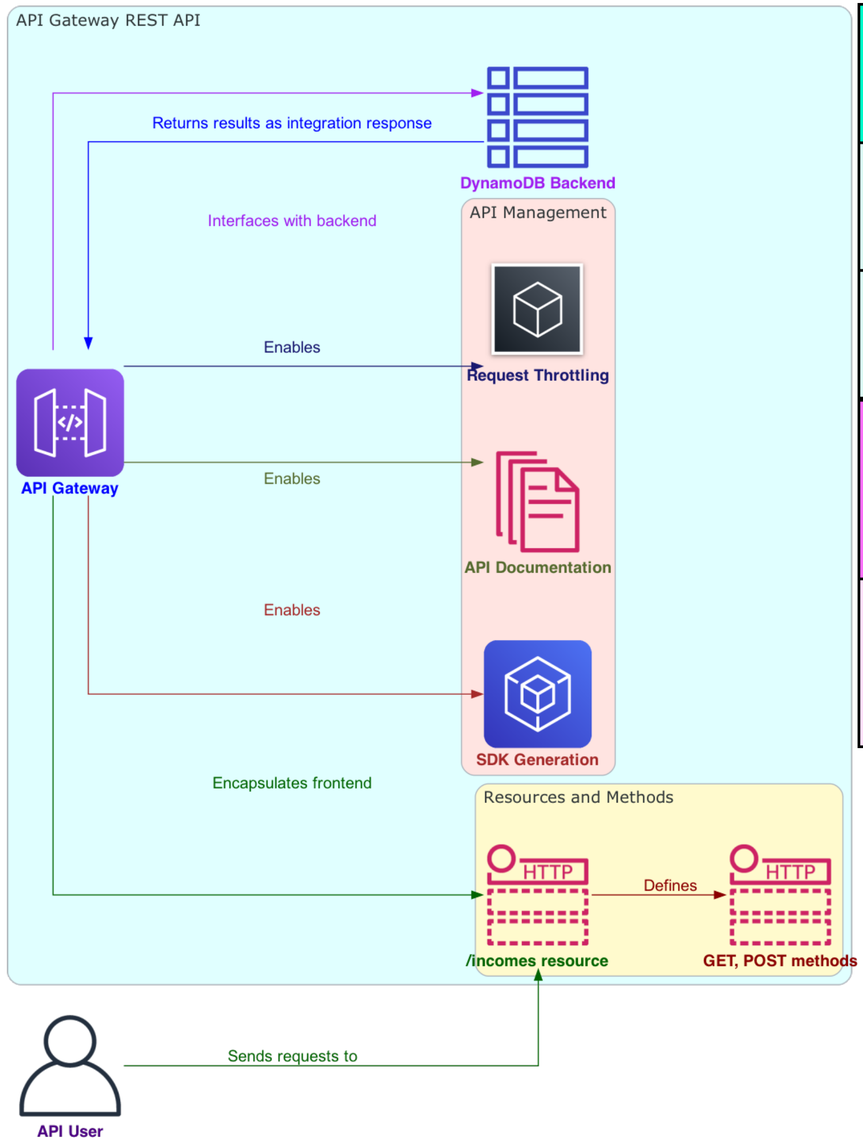
7. 🛡️ AWS WAF integration

🔒 Protect against web exploits

8. 🕵️ AWS X-Ray integration

🔍 Understand and triage latencies

# Building REST APIs with Amazon API Gateway




1. 🧱 Resources and Methods: Building blocks

🎛️ Resources: Logical entities, resource paths

🔍 Methods: REST API requests, responses

4. 🔗 Backend integration through requests/responses

🔄 Integration requests, responses for communication

2. 📡 Resource paths, HTTP methods define endpoints

🔢 Example: /incomes resource path

🧭 HTTP methods: GET, POST, PUT, PATCH, DELETE

🔍 Resource path + HTTP method = API method

➕ Example: POST /incomes to add income

🔍 Example: GET /expenses to query expenses

3. 🔀 Frontend encapsulated by requests/responses

🔍 Method requests, responses abstract backend

5. 🗄️ DynamoDB example: Configuring backend

🔍 Integration request forwards to DynamoDB

⚙️ Specifies action, IAM role, policies, data transformation

🔄 DynamoDB returns result as integration response

6. 🗺️ Mapping integration to method responses

🔄 Configure integration response to map parameters

🔄 Translate output data format if needed

🧩 Define schema/model for payload mapping

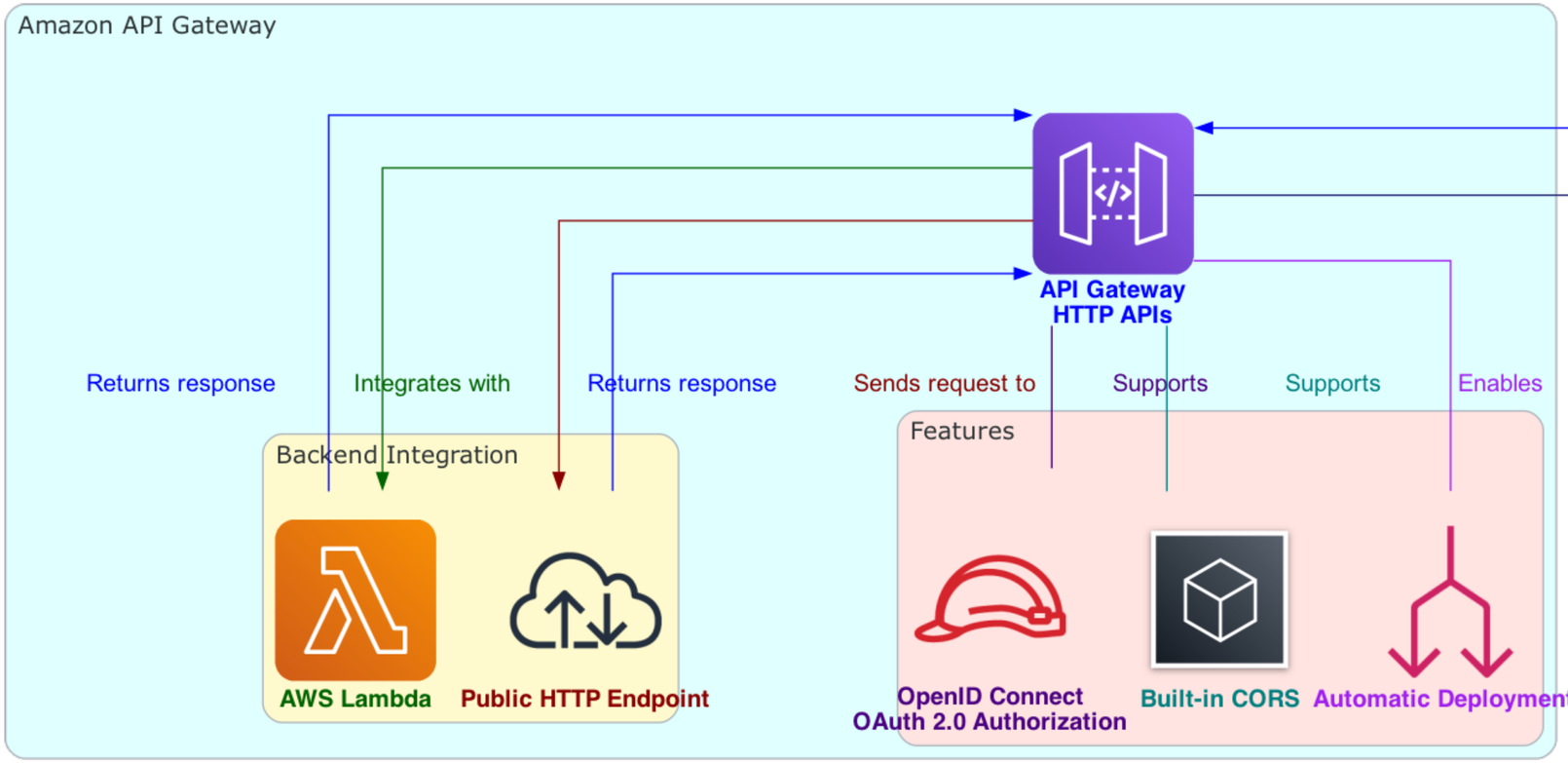
7. 🛠️ Additional API management features

🔧 SDK generation

📚 API documentation using OpenAPI

⏰ HTTP request throttling for usage management

# 🌐 Creating HTTP APIs with Amazon API Gateway




1. 🚀 RESTful APIs with lower latency and cost
- ⏰ Lower latency
- 💰 Lower cost

2. 🔗 Integration with AWS services and HTTP endpoints
- 🔌 Send requests to Lambda functions
- 🌍 Send requests to public HTTP endpoints
- 🎨 Flexibility in backend integrations

3. 🌟 Example: HTTP API with Lambda backend
- 🔄 Client calls API
- 🚀 API Gateway sends request to Lambda
- 🔄 Lambda returns response to client
- 🤝 Seamless client-backend communication

4. 🔐 OpenID Connect and OAuth 2.0 support
- 🆔 Built-in support for authentication
- 🔒 Built-in support for authorization
- 🛡️ Enhanced API security

5. 🌍 Built-in CORS support
- 🤝 Easier handling of cross-origin requests
- 🔒 Proper access control for APIs

6. 🛠️ Automatic deployments
- 🚀 Simplified deployment process
- 🔄 Easy API updates without manual intervention

# Building WebSocket APIs with Amazon API Gateway

1. 🔄 Bi-directional, real-time communication

↔️ Client and server can send messages anytime

📤 Backend servers push data to users, devices

🚫 No complex polling mechanisms needed

2. 🏗️ Serverless app example: Chat room with Lambda

🌐 Build using API Gateway WebSocket API, Lambda

📤 Send/receive messages to/from users, groups

3. 🔌 Invoke backend services based on message content

⚡ Invoke Lambda, Kinesis, HTTP endpoints

4. 🔒 Secure, server-free real-time apps

🌐 Build secure, real-time communication apps

🚫 No need to manage servers for connections, data

5. 🎯 Use cases

💬 Chat applications

📈 Real-time dashboards (e.g., stock tickers)

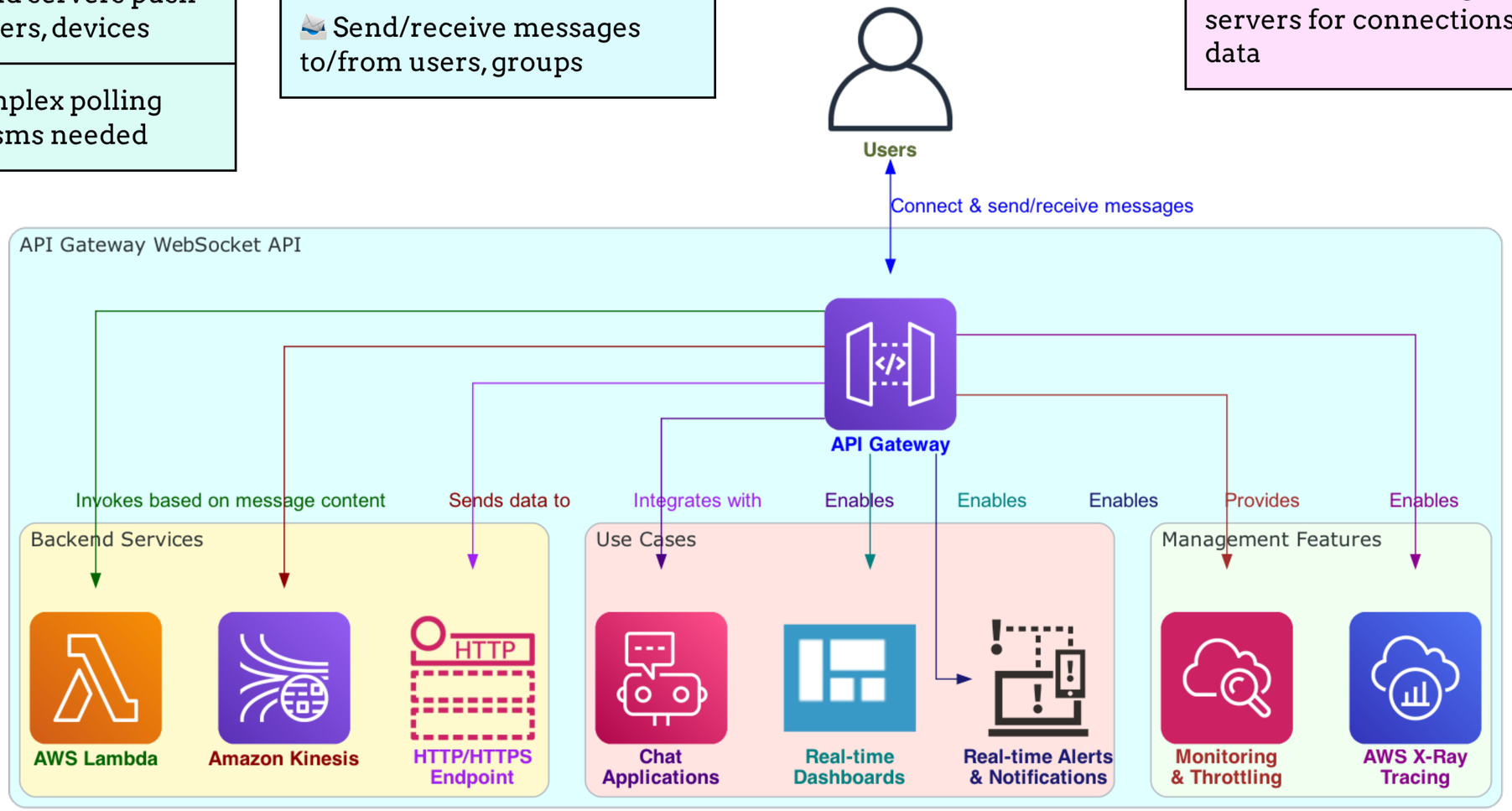
🚨 Real-time alerts and notifications

6. 🛠️ Management features

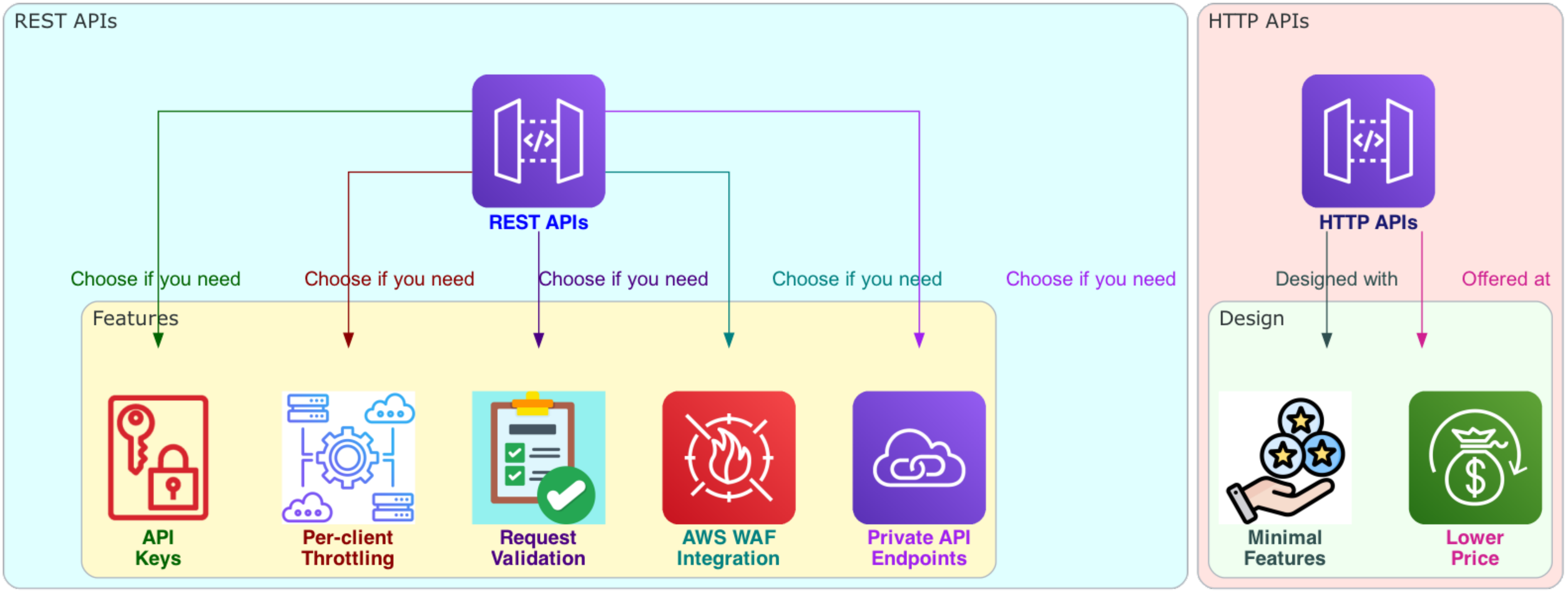
📊 Monitoring and throttling connections, messages

🕵️ AWS X-Ray tracing for messages

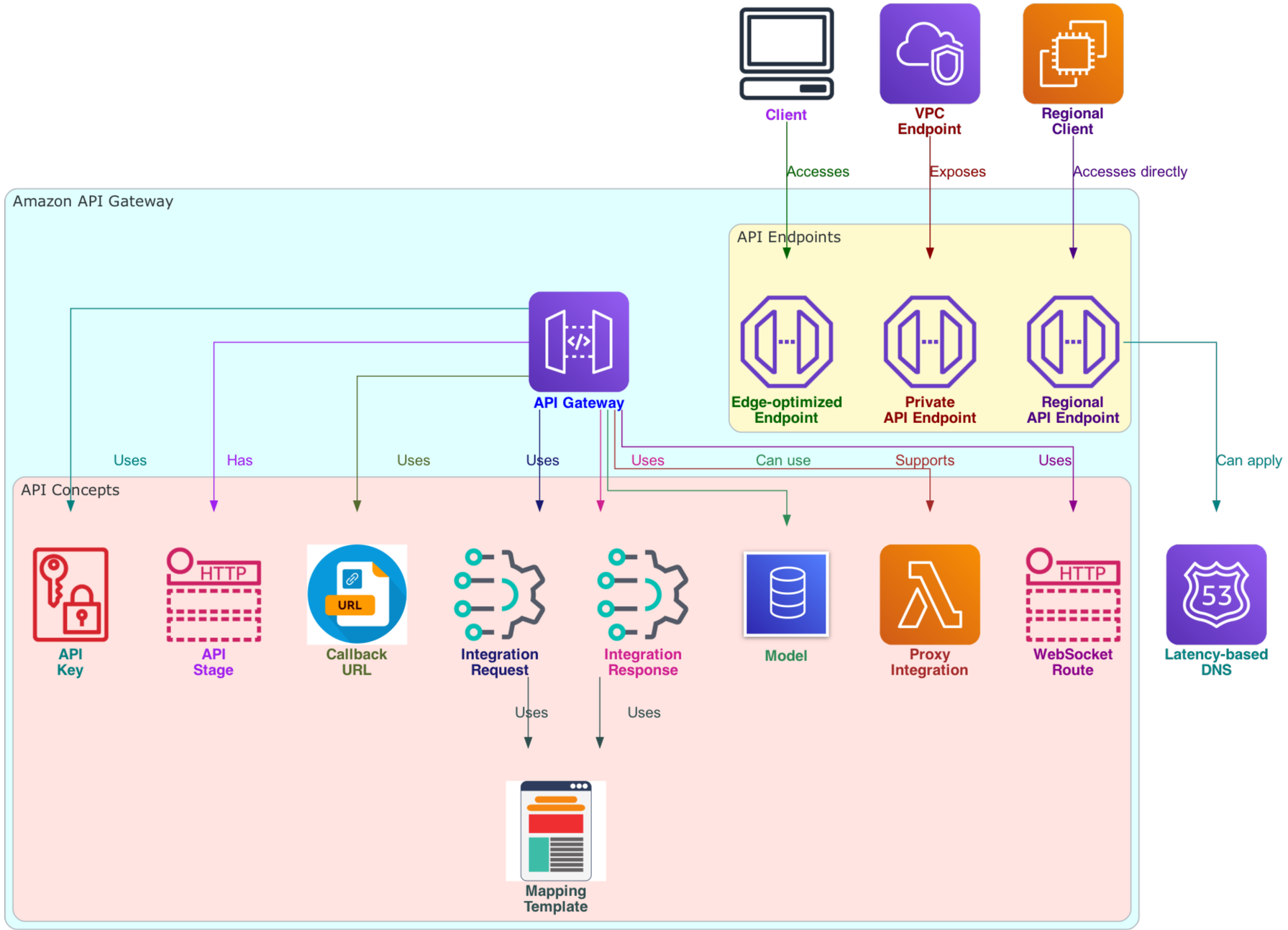
🔌 Easy integration with HTTP/HTTPS endpoints



# Choosing between REST APIs and HTTP APIs

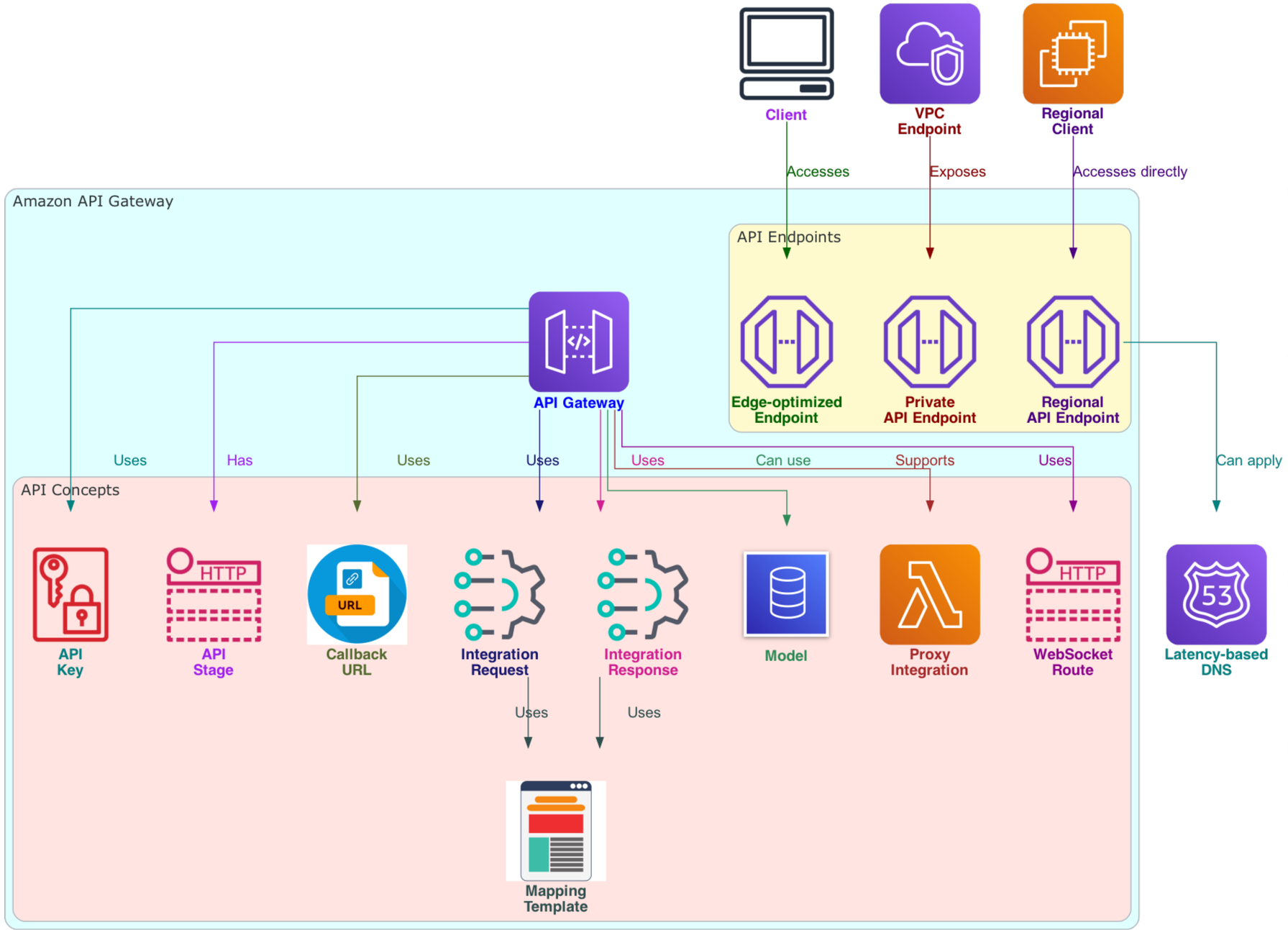


# Amazon API Gateway concepts




1. 🌐 **API Endpoint**: 🏠 Hostname for deployed API, 🌍 Region-specific, 🔌 Supported endpoint types: 🚀 Edge-optimized API endpoint (☁️ Uses CloudFront for cross-region access, 🌎 Routes requests to nearest POP), 🔒 Private API endpoint (🌉 Exposed through VPC endpoints, 🛡️ Securely accesses private API resources), 🌍 Regional API endpoint (🎯 Serves clients in same AWS Region, ⏩ Bypasses CloudFront for in-region requests, 🌐 Supports latency-based routing)
2. 🔑 **API Key**: 🔡 Alphanumeric string, 👨‍💻 Identifies app developer, 🔐 Controls API access with authorizers, usage plans
3. 🎭 **API Stage**: 📜 Logical reference to API lifecycle state, 🆔 Identified by API ID and stage name
4. 📞 **Callback URL**: 🔄 Stored for WebSocket clients, 📨 Used to send messages from backend

# Amazon API Gateway concepts




5. 🔄 **Integration Request/Response**: 🛠️ Internal interface for WebSocket/ REST APIs, 🗺️ Maps request/response formats
6. 📝 **Mapping Template**: 📜 VTL script for data transformation, ⚙️ Specified in integration request/response, 🔄 Passes data as-is or transforms it
7. 📊 **Model**: 📋 Data schema for request/response payload, 🔧 Required for generating SDK, validating payloads, 💡 Useful for generating sample mapping template
8. 🔌 **Proxy Integration**: 🔧 Simplified API Gateway integration, 🌐 HTTP proxy or Lambda proxy, 🔄 Passes entire request/response
9. 🛤️ **Route**: 🗺️ Directs WebSocket messages to integration, 🔍 Based on route key in message body, 🔀 Default route for non-matching keys or proxy

# Thanks for Watching