**API Gateway can route requests to various backend services**

* **AWS Lambda:** Serverless functions for compute tasks without EC2.
* **Amazon EC2:** For traditional server-based services, API Gateway can route requests to EC2 instances.
* **HTTP Endpoints:** Directly calling external HTTP APIs.
* **AWS Services:** Like DynamoDB, S3, or Step Functions, without needing EC2 instances.
* **Security and Access Control**: API Gateway manages access control through API keys, AWS IAM roles and policies, Lambda authorizers, or Amazon Cognito user pools, which don’t require EC2.
* **Scaling and Management**: API Gateway handles scaling automatically and maintains high availability for the APIs.
* **Public EC2 Instances**: Use API Gateway to directly connect using the public URL of the EC2 instance.
* **Private EC2 Instances**: Set up a VPC Link in API Gateway to securely route traffic to private EC2 instances within a VPC.
* **Security**: API Gateway can handle authentication, authorization, and throttling, reducing the burden on EC2.
* **Amazon API Gateway** can route requests to an EC2 instance as an HTTP proxy.

In the context of API Gateway, an **HTTP Proxy** is a setup where API Gateway forwards incoming requests directly to a backend HTTP endpoint (like an EC2 instance) with minimal processing. It essentially acts as a “pass-through” layer, sending the request to the backend server and returning the response without modifying it.

### **Key Points of HTTP Proxy in API Gateway**

* **Direct Routing**: API Gateway forwards the request exactly as it is, preserving headers, query parameters, and body content.
* **Minimal Processing**: API Gateway doesn’t modify or inspect the request. It just relays the data back and forth between the client and backend.
* **Flexible**: This setup is useful if you want API Gateway to act as a front door to your existing HTTP-based services (e.g., those hosted on EC2, another API, or an external service).
* **Security and Throttling**: API Gateway can still manage access control, throttling, and monitoring, while passing requests to the backend.

With an **HTTP proxy integration** to an EC2 instance, **all backend functionality**—such as processing requests and handling business logic—would run on the EC2 instance instead of AWS Lambda.

**Using EC2 as Backend**: When API Gateway routes requests to EC2, the EC2 instance handles all processing, such as executing functions, accessing databases, and generating responses. This approach is ideal if you have a more traditional server-based application or if you need control over the server environment.

**Using AWS Lambda as Backend**: In contrast, if you integrate API Gateway with AWS Lambda, the Lambda functions handle the backend logic. Lambda is a serverless solution, meaning you don’t manage the underlying infrastructure, and it automatically scales with demand. It’s cost-effective for event-driven or sporadic workloads, as you pay per request rather than for ongoing server time.

**Summary**

While API Gateway can be used with EC2 instances as a backend, it does not inherently rely on EC2. It’s a standalone service that can work with various AWS. With an HTTP proxy to EC2, you’re opting for a setup where the backend processing runs on EC2 instead of Lambda. This can be advantageous if your application is stateful, resource-intensive, or requires specific environment configurations that Lambda doesn’t support.