Case Study 2:

API Management and Serverless Architecture

Company Background:

DEF Corporation is a leading fintech company that wants to create a robust API management system to handle multiple APIs from various sources. The company has a existing Function App that handles API requests, but it is not scalable and secure. The company wants to migrate to a serverless architecture using Azure API Management (APIM) and Azure Functions.

Requirements:

1. API Management: The company wants to create a centralized API management system that can handle multiple APIs from various sources.
2. Security: The company wants to ensure that the APIs are secure and protected from unauthorized access.
3. Scalability: The company wants to ensure that the APIs can handle a large number of requests without a significant decrease in performance.
4. Monitoring and Analytics: The company wants to monitor and analyze the API usage to improve the overall performance and security.

Current Infrastructure:

1. Function App: The company has an existing Function App that handles API requests.
2. Azure Subscription: The company has an Azure subscription with access to Azure API Management (APIM) and other Azure services.
3. Terraform: The company wants to use Terraform to automate the deployment of the APIM and Function App.

Challenge:

The company's IT team is responsible for deploying and managing the APIs. However, they face challenges in ensuring the security, scalability, and monitoring of the APIs. They want to migrate to a serverless architecture using APIM and Functions to improve the overall performance and security.

Your Task:

Design and implement a solution that meets the requirements and challenges outlined above. The solution should include the following components:

1. APIM: Create an APIM instance to handle multiple APIs from various sources.
2. Function App: Migrate the existing Function App to a serverless architecture using Azure Functions.
3. Terraform: Use Terraform to automate the deployment of the APIM and Function App.
4. Security: Configure security policies to protect the APIs from unauthorized access.
5. Monitoring and Analytics: Configure monitoring and analytics to track API usage and improve the overall performance.

Deliverables:

1. A detailed design document outlining the solution architecture.
2. A Terraform configuration file that automates the deployment of the APIM and Function App.
3. An APIM configuration file that defines the API policies and security settings.
4. A Function App configuration file that defines the API handlers and security settings.
5. A report that summarizes the solution and its benefits.

APIM Configuration:

The APIM instance should be configured to handle multiple APIs from various sources. The APIM configuration file should include the following settings:

API policies: Define the API policies to handle authentication, authorization, and rate limiting.
Security settings: Configure the security settings to protect the APIs from unauthorized access.
Product settings: Define the product settings to handle API subscriptions and quotas.

Function App Configuration:

The Function App should be configured to handle API requests using Azure Functions. The Function App configuration file should include the following settings:

API handlers: Define the API handlers to handle API requests.
Security settings: Configure the security settings to protect the APIs from unauthorized access.

Monitoring settings: Configure the monitoring settings to track API usage and improve the overall performance.

Terraform Configuration:

The Terraform configuration file should automate the deployment of the APIM and Function App. The Terraform configuration file should include the following settings:

Resource definitions: Define the resource definitions for the APIM and Function App.
Variable definitions: Define the variable definitions for the APIM and Function App.
Output definitions: Define the output definitions for the APIM and Function App.

Monitoring and Analytics:

The company wants to monitor and analyze the API usage to improve the overall performance and security. The solution should include the following monitoring and analytics settings:

API usage tracking: Track API usage to identify performance bottlenecks and security threats.
Error tracking: Track errors to identify and fix issues quickly.
Performance monitoring: Monitor performance to identify areas for improvement.

Security:

The company wants to ensure that the APIs are secure and protected from unauthorized access. The solution should include the following security settings:

Authentication: Configure authentication to protect APIs from unauthorized access.
Authorization: Configure authorization to restrict access to APIs.
Rate limiting: Configure rate limiting to prevent API abuse.

Scalability:

The company wants to ensure that the APIs can handle a large number of requests without a significant decrease in performance. The solution should include the following scalability settings:

Horizontal scaling: Configure horizontal scaling to add more instances as needed.
Vertical scaling: Configure vertical scaling to increase instance capacity as needed.
Load balancing: Configure load balancing to distribute traffic across instances.