Microsoft

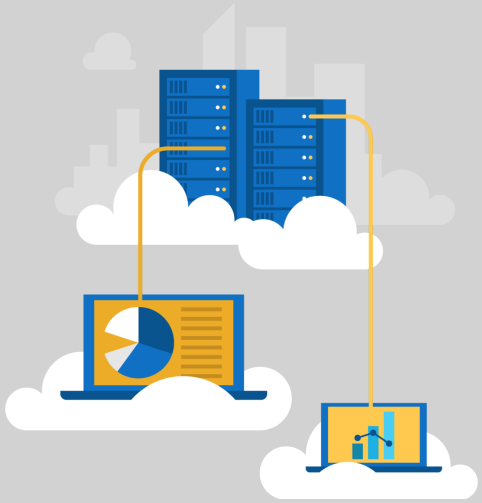
Module 08: Implement API Management

Microsoft

Topics

- API Management overview
- Working with APIs in APIM
- Configure authentication for APIs

Lesson 01: API Management overview



API Management (APIM)

- Streamlines the process of common tasks necessary for creating an API for external use
- Tasks include:
  - Creating a successful and useful developer portal
  - Securing API endpoints from anonymous or unwanted access
  - Managing existing developer access through cache mechanisms, throttling, and other policies
  - Building a monitoring and analytics platform to diagnose issues and monitor adoption
  - Providing business users and developers with deep insights into how each API is specifically used

Terminology

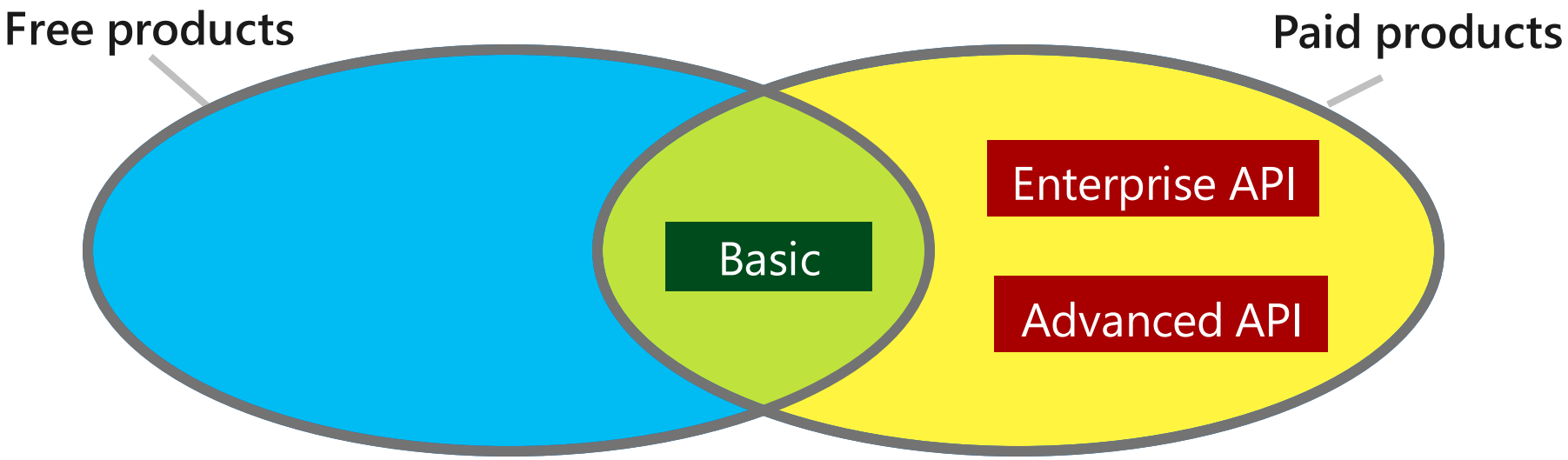
- Backend API:
  - A HTTP service that you implement with your business logic
- Frontend API:
  - A HTTP service façade hosted by API Management to obfuscate your back-end API
- Product:
  - One or more APIs, along with a usage quota and terms of use
- Operation:
  - A specific operation in the front-end API that correlates to a specific request/response from the backend API

Terminology (continued)

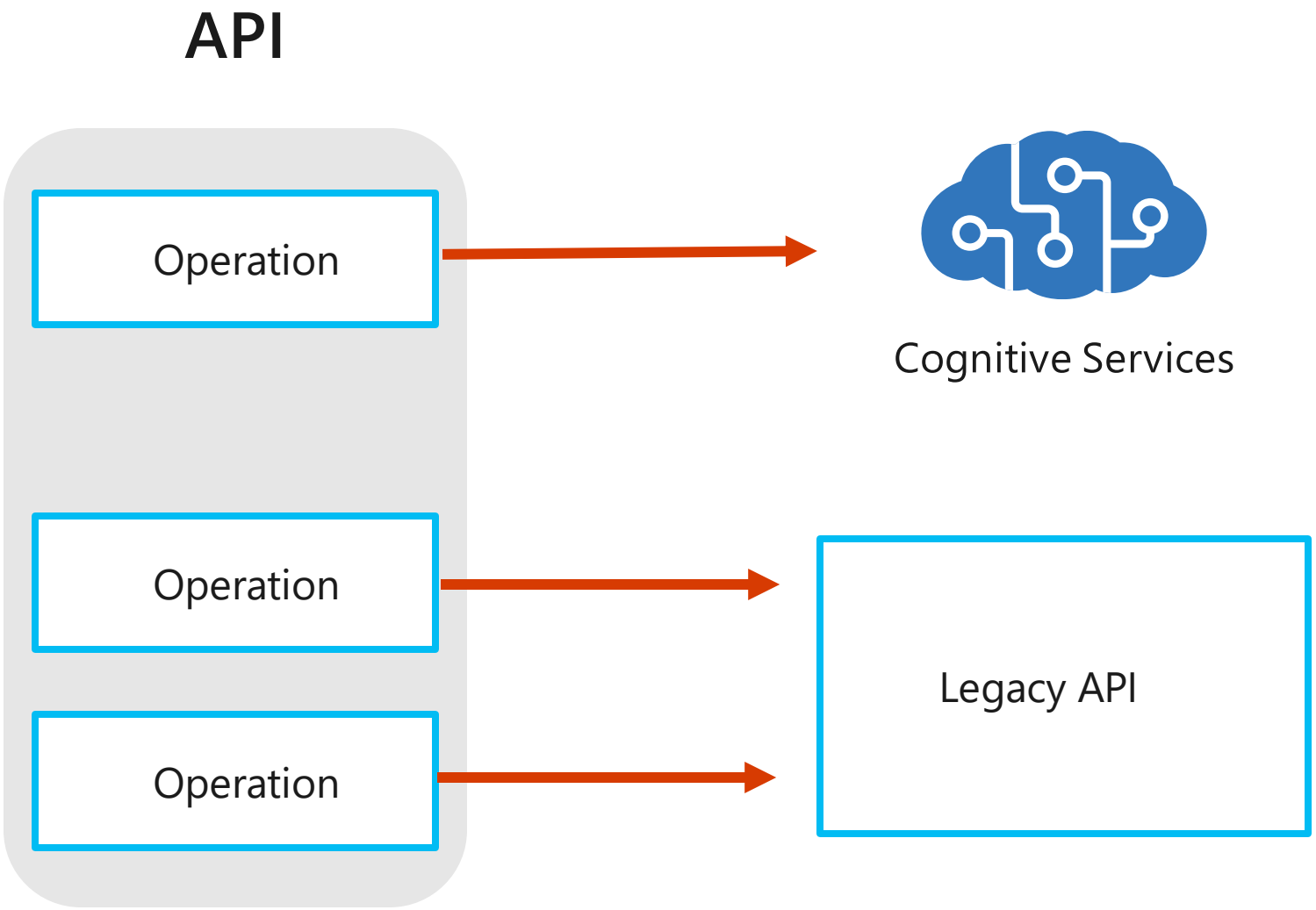
- Version:
  - A breaking change to the front-end API
  - Existing application will not be required to change its code as you update or change the front-end API
- Revision:
  - A non-breaking change to a front-end API
- Developer portal:
  - An interface that developers use to learn about your API and test operations

Products

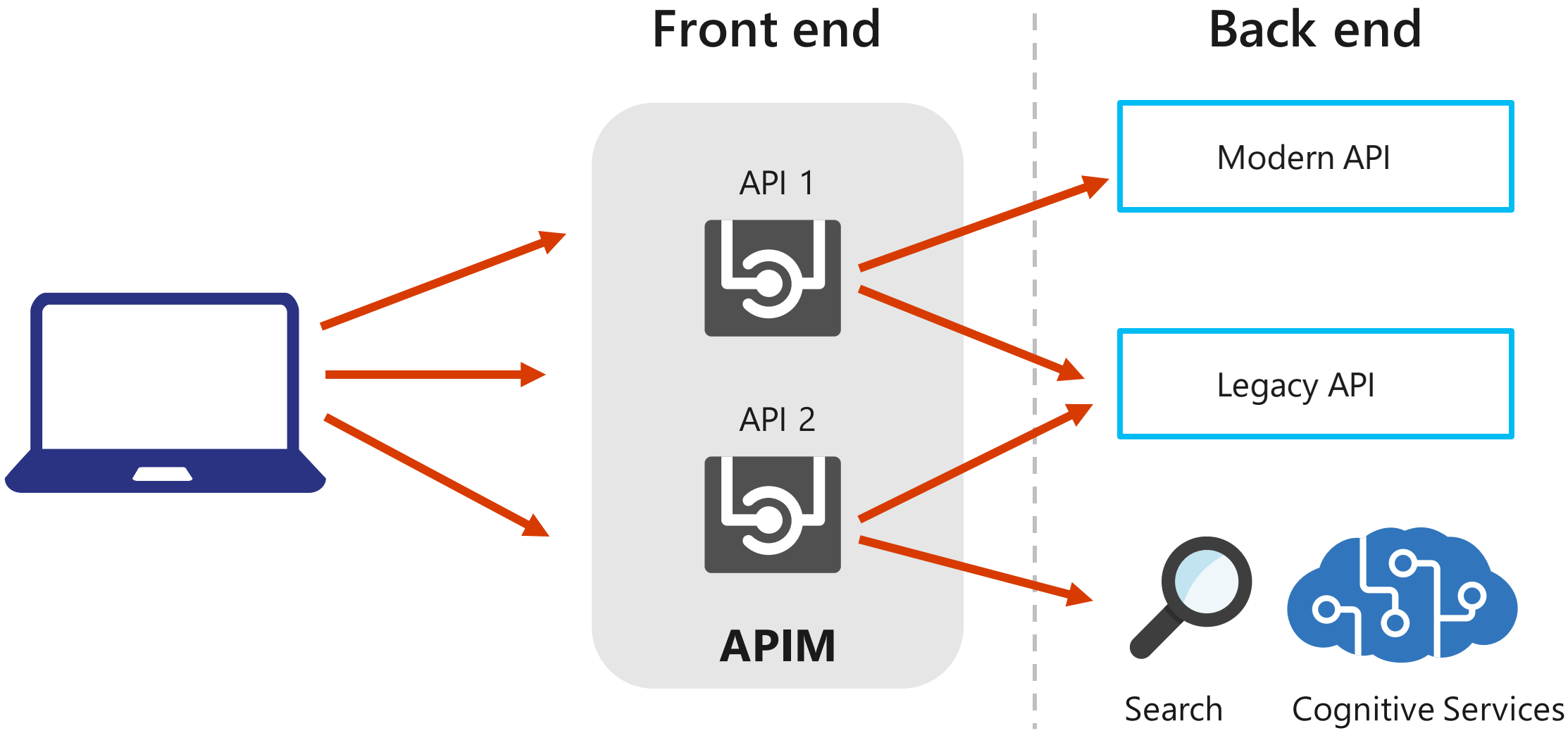
- Contains one or more APIs in a package
- Products can be open or protected:
  - Open products are free to use without any subscription
  - Protected products must be subscribed to before use
- When a product is ready for developers, it can be published for use



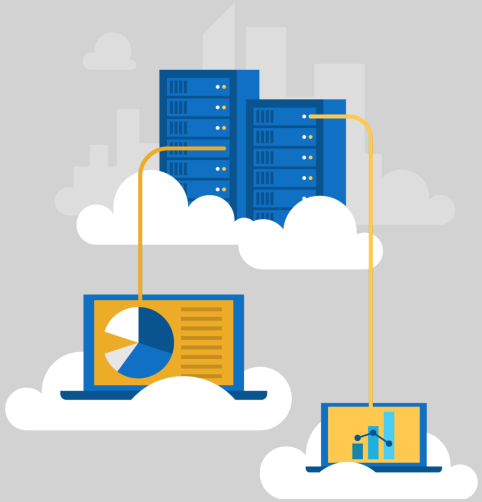
APIs and operations



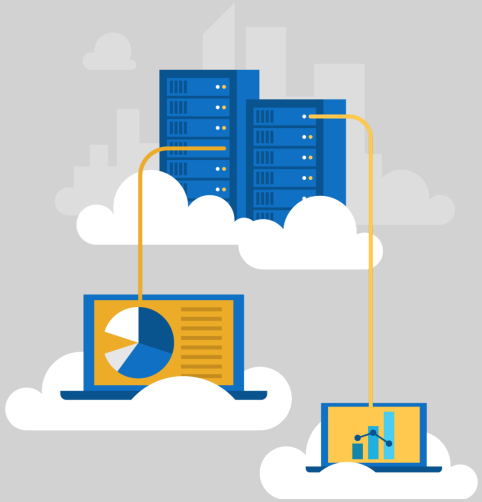
Back-end and front-end APIs



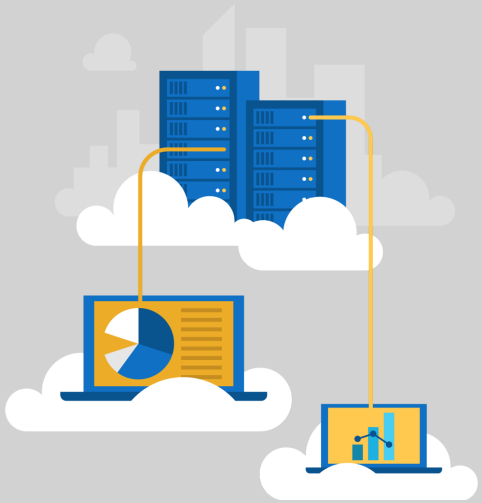
Demonstration: Creating an APIM instance by using Azure CLI



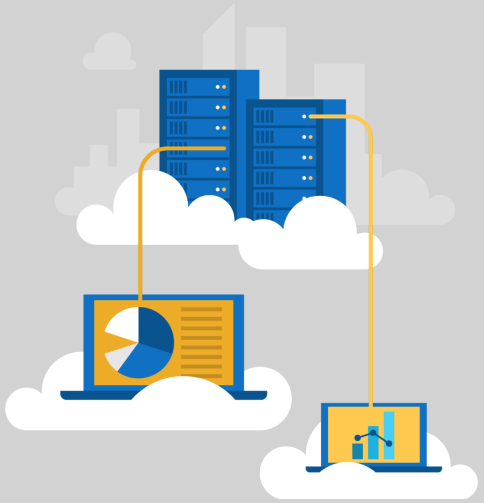
Demonstration: Importing an API by using the Azure portal



Demonstration: Creating and publishing a product



Lesson 02: Working with APIs in APIM

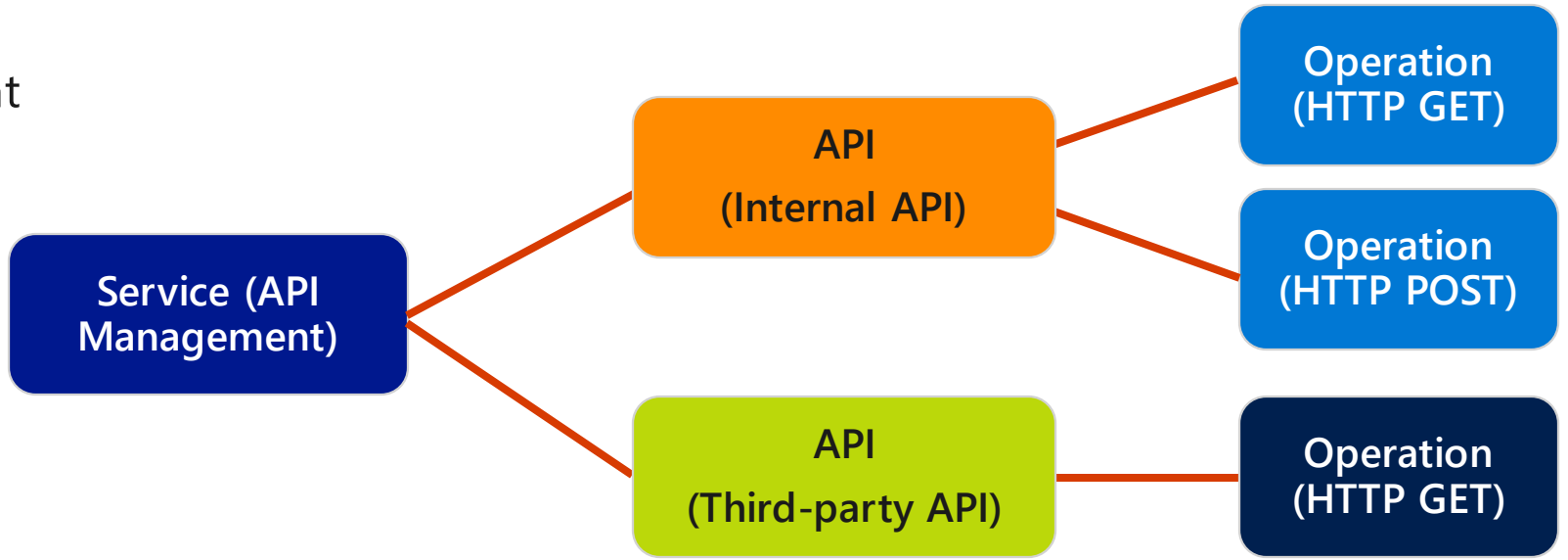


Creating an API Management instance

- Create and manage APIs
- Each API contains one or more sets of operations
- Operations are configurable, granting control over:
  - URL mapping
  - Query and path parameters
  - Request and response content
  - Operation response caching

Service hierarchy

- Create and manage APIs
- Each API contains one or more sets of operations
- Operations are configurable, granting control over:
  - URL mapping
  - Query and path parameters
  - Request and response content
  - Operation response caching

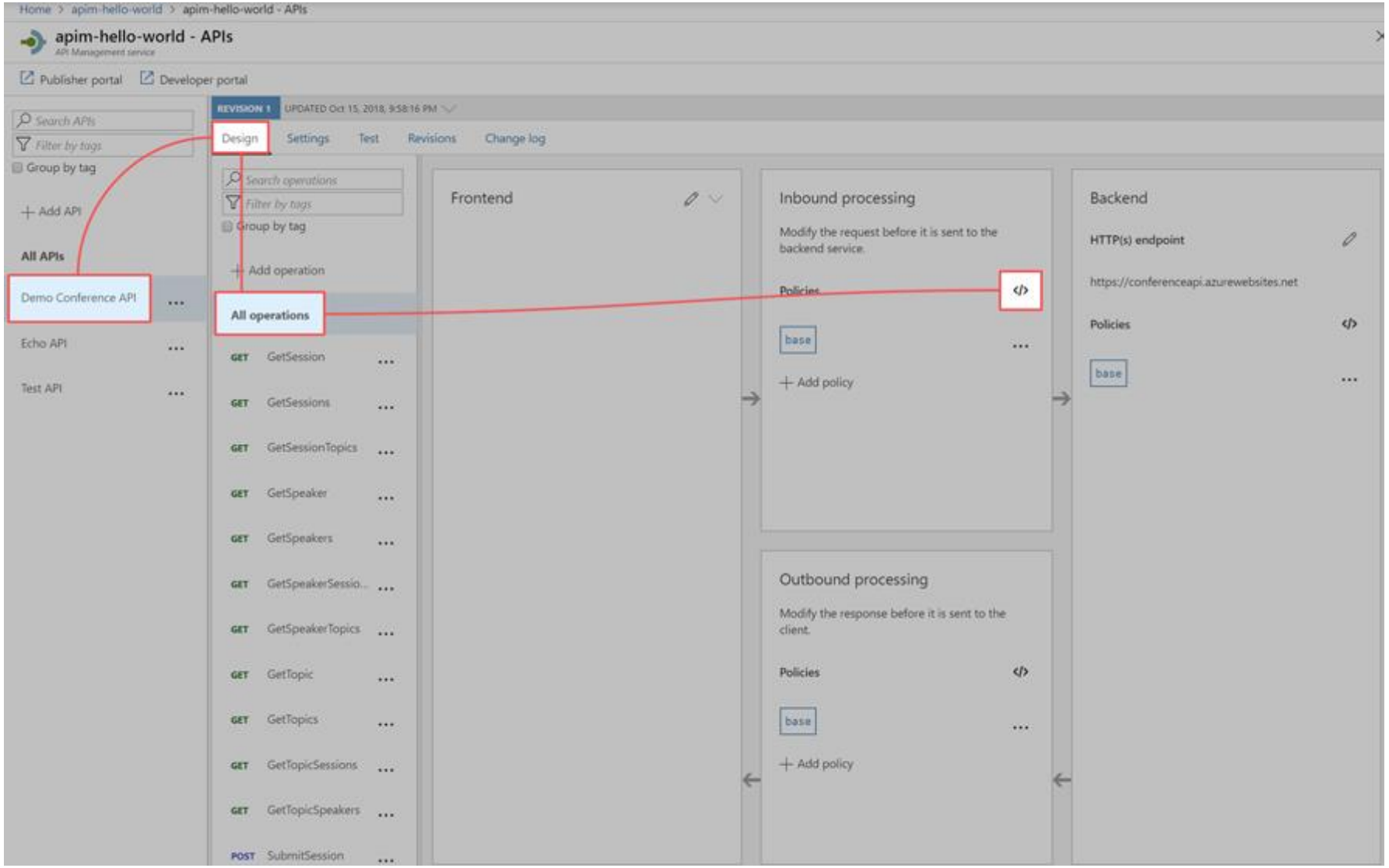


Policies

- Collection of statements that are executed sequentially at the request or response of an API
- Are a quick way to change the behavior of an API without code changes to the actual back-end API application
- A comprehensive list of policy options can be found at API Management policies (https://aka.ms/AA4gbik)

Editing policies

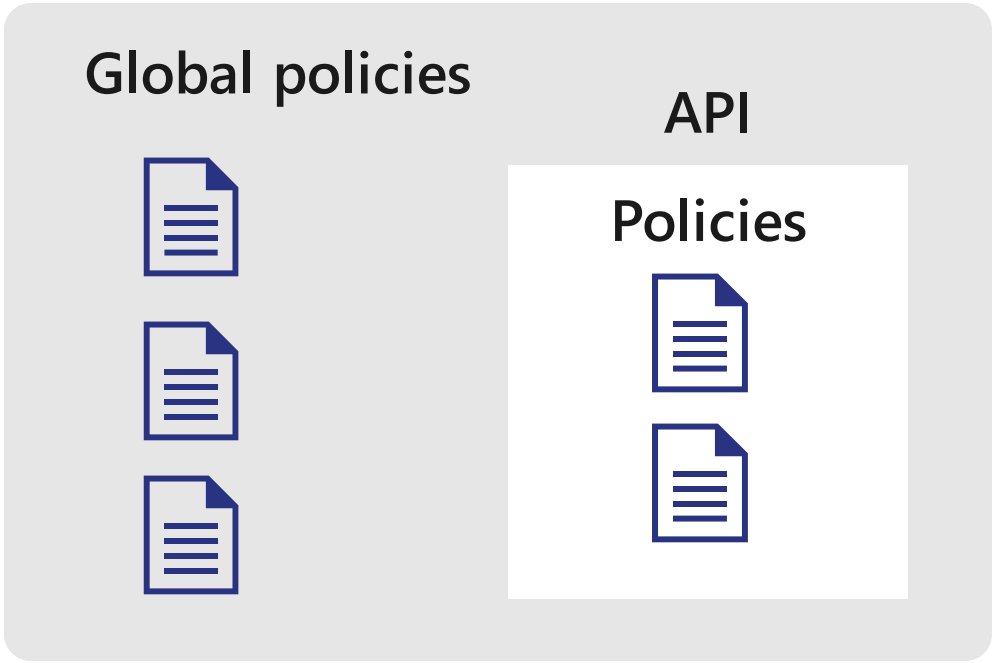
```
<policies>
    <inbound>
        <base />
    </inbound>
    <backend>
        <base />
    </backend>
    <outbound>
        <base />
    </outbound>
    <on-error>
        <base />
    </on-error>
</policies>
```



Policy scopes

```
<policies>
    <inbound>
        <cross-domain />
        <base />
        <find-and-replace from="xyz" to="abc" />
    </inbound>
</policies>
```

Global policies are invoked here.



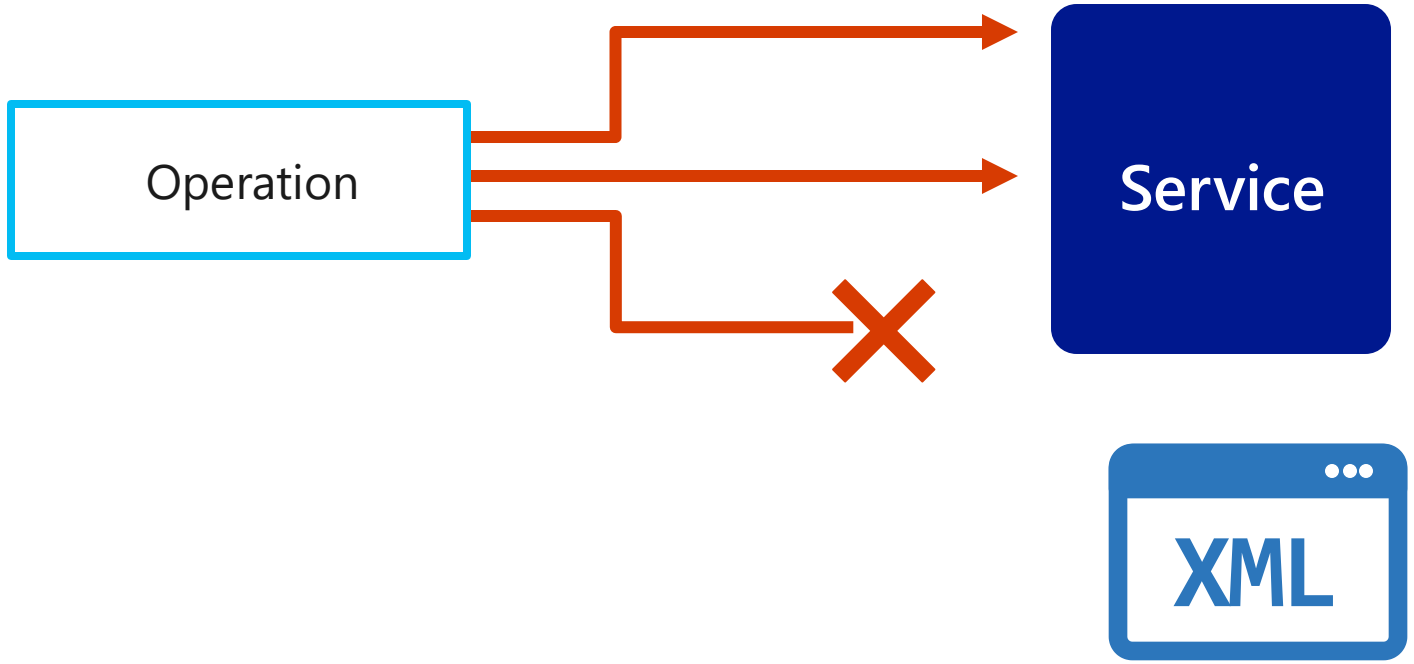
Demonstration: Transforming an API by using policies

Advanced policy scenarios – control flow

```
<choose>
    <when condition="Boolean expression | Boolean constant">
        <!-- one or more policy statements to be applied if the above condition is true -->
    </when>
    <when condition="Boolean expression | Boolean constant">
        <!-- one or more policy statements to be applied if the above condition is true -->
    </when>
    <otherwise>
        <!-- one or more policy statements to be applied if none of the above conditions are true -->
    </otherwise>
</choose>
```

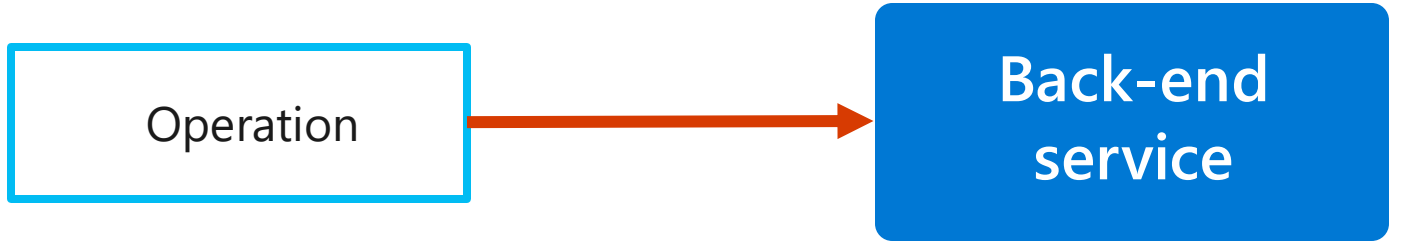
Advanced policy scenarios – limit concurrency

```
<limit-concurrency key="expression" max-count="number">
    <!-- nested policy statements -->
</limit-concurrency>
```



Advanced policy scenarios – forward request

```
<forward-request timeout="time in seconds" follow-redirects="true | false"/>
```

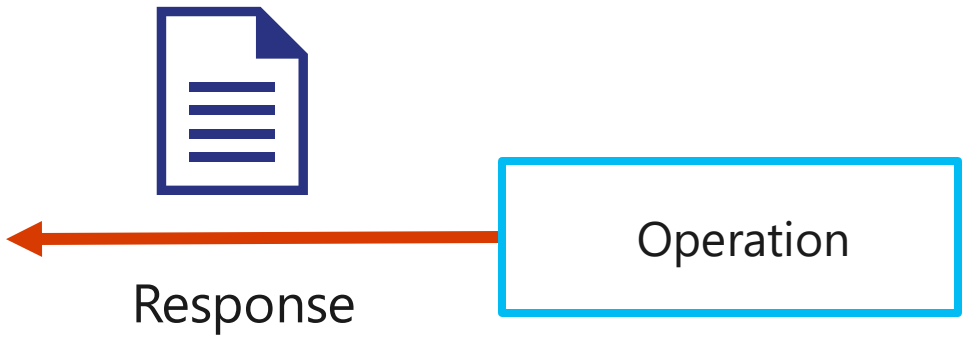


Advanced policy scenarios – log to Event Hub

```
<log-to-eventhub logger-id="id of the logger entity" partition-id="index of the
partition where messages are sent" partition-key="value used for partition assignment">
    <!-- Expression returning a string to be logged -->
</log-to-eventhub>
```

Advanced policy scenarios – mock response

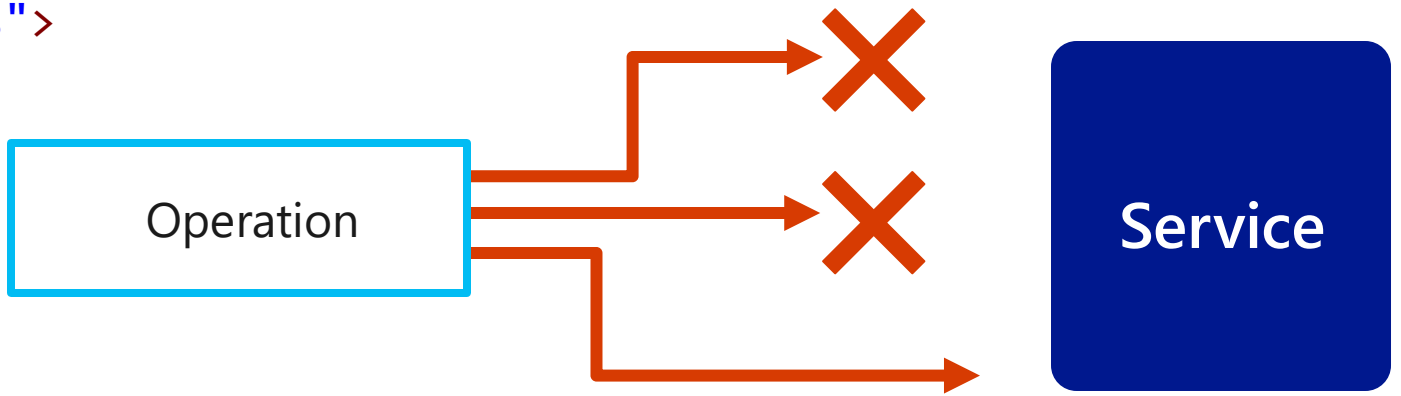
```
<mock-response status-code="code" content-type="media type"/>
```



Advanced policy scenarios – retry

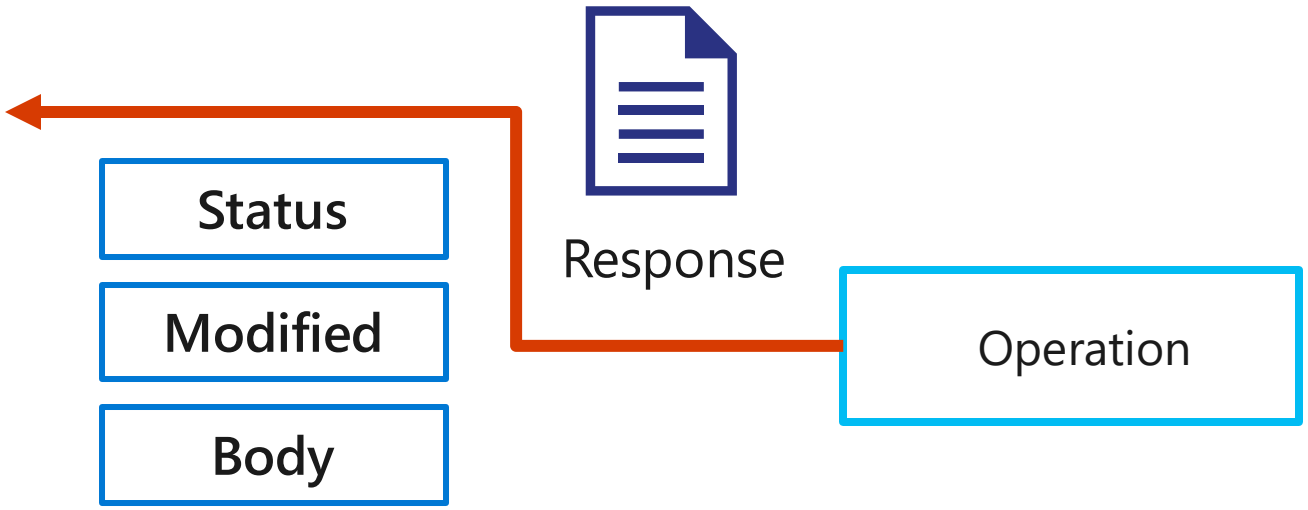
```
<retry
    condition="boolean expression or literal"
    interval="retry interval in seconds"
    max-interval="maximum retry interval in seconds"
    delta="retry interval delta in seconds"
    count="number of retry attempts">

    <!-- One or more child policies.
         No restrictions -->
</retry>
```



Advanced policy scenarios – return response

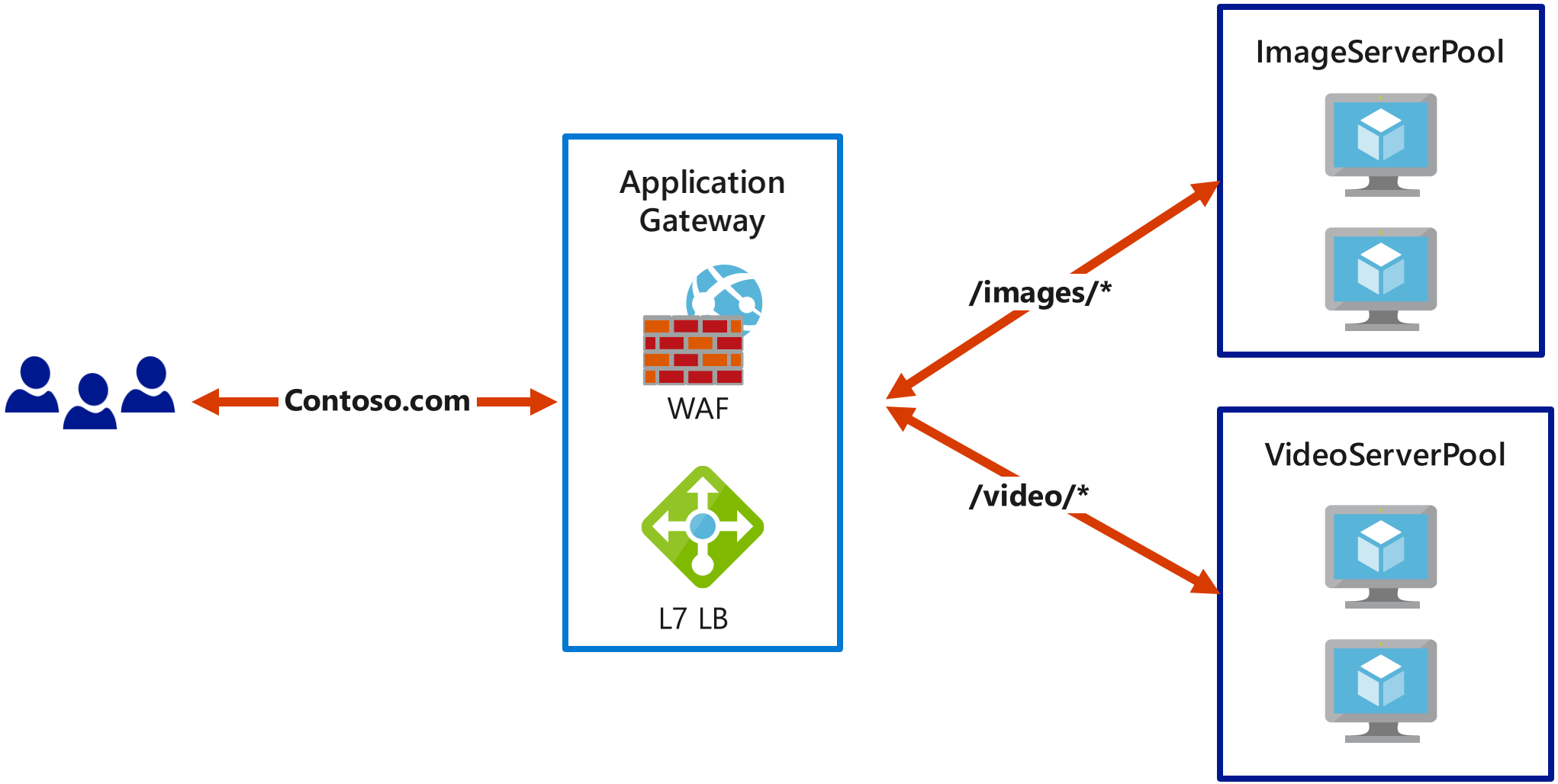
```
<return-response response-variable-name="existing context variable">
    <set-header/><set-body/><set-status/>
</return-response>
```



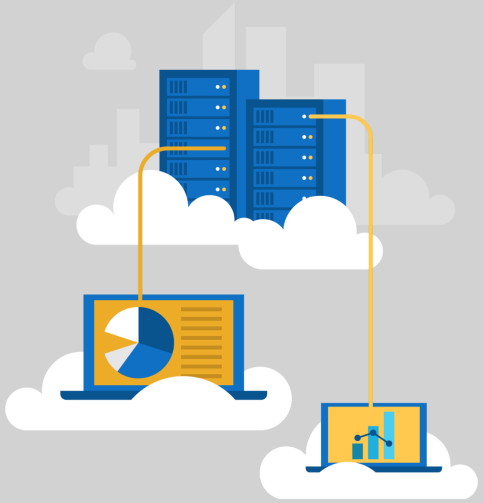
Application Gateway

- Enables management of traffic to your web applications
- Operates as a web traffic load balancer, which allows for more specific traffic routing than a traditional load balancer
- Can handle additional scenarios such as:
  - Custom routing
  - Session affinity
  - SSL termination
  - Firewall management
  - Redirection

Application Gateway (continued)



Lesson 03: Configure authentication for APIs



Subscriptions

- Subscriptions tie **Developers** together with **Products**
- A Developer will sign up for a subscription to get access to various products
  - The subscription will grant the Developer access to subscription keys
  - The subscription keys can be used to access specific products

Client certificates

```
<!-- checking the expiration date -->
<choose>
    <when condition="@(context.Request.Certificate == null ||
context.Request.Certificate.NotAfter < DateTime.Now)" >
        <return-response><set-status code="403" reason="Invalid client certificate"
        /></return-response></when></choose>
<!-- checking the issuer and subject -->
<choose>
    <when condition="@(context.Request.Certificate == null ||
context.Request.Certificate.Issuer != "trusted-issuer" ||
context.Request.Certificate.SubjectName != "expected-subject-name")" >
        <return-response><set-status code="403" reason="Invalid client certificate"
        /></return-response></when></choose>
```

Client certificates (continued)

```
<!-- checking the thumbprint -->
<choose>
    <when condition="@(context.Request.Certificate == null ||
context.Request.Certificate.Thumbprint != "desired-thumbprint")" >
        <return-response><set-status code="403" reason="Invalid client certificate"
        /></return-response></when></choose>
<!-- checking a thumbprint against certificates uploaded to API Management -->
<choose>
    <when condition="@(context.Request.Certificate == null ||
!context.Deployment.Certificates.Any(c => c.Value.Thumbprint ==
context.Request.Certificate.Thumbprint))" >
        <return-response><set-status code="403" reason="Invalid client certificate"
        /></return-response></when></choose>
```

Lab: Creating a multi-tier solution by using services in Azure

Duration

Lab sign-in information