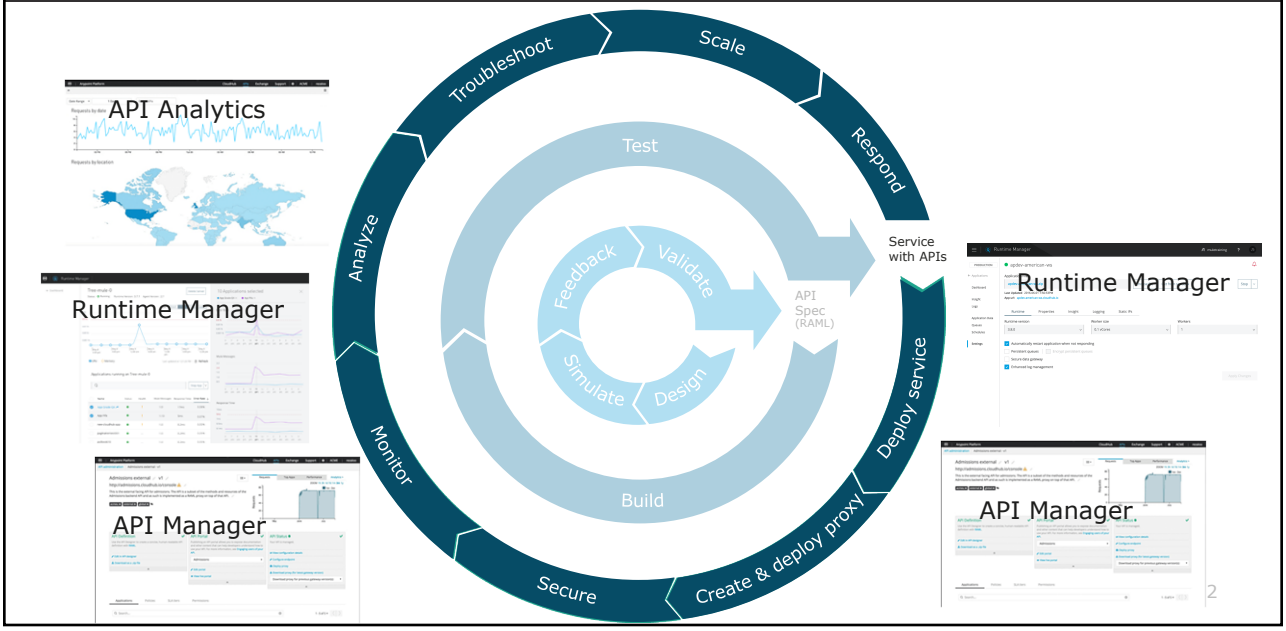
MuleSoft

Module 5: Deploying and Managing APIs

Goal



At the end of this module, you should be able to

MuleSoft

- Describe the options for deploying Mule applications
- Use properties in Mule applications so they can easily move between environments
- Deploy Mule applications to CloudHub
- Use API Manager to create and deploy proxies for APIs
- Use API Manager to restrict access to API proxies

Introducing deployment options

Deploying applications

- During development, applications are deployed to an embedded Mule runtime in Anypoint Studio
- For everything else (testing, Q&A, and production), applications can be deployed to
  - **Customer-hosted Mule runtimes**
  - **CloudHub**
    - Platform as a Service (PaaS) component of Anypoint Platform
    - MuleSoft-hosted Mule runtimes on AWS (Amazon Web Services platform)
    - A fully-managed, multi-tenanted, globally available, secure and highly available cloud platform for integrations and APIs

Customer-hosted runtime

MuleSoft-hosted runtime

CloudHub benefits

- No hardware to maintain
- Continuous software updates
- Provided infrastructure for DNS and load-balancing
- Built-in elastic scalability for increasing cloud capacity during periods of high demand
- Globally available with data centers around the world
- Highly available with 99.99% uptime SLAs (service level agreements) http://status.mulesoft.com/
- Highly secure
  - PCI, HiTrust, and SSAE-16 certified

Customer-hosted Mule runtimes

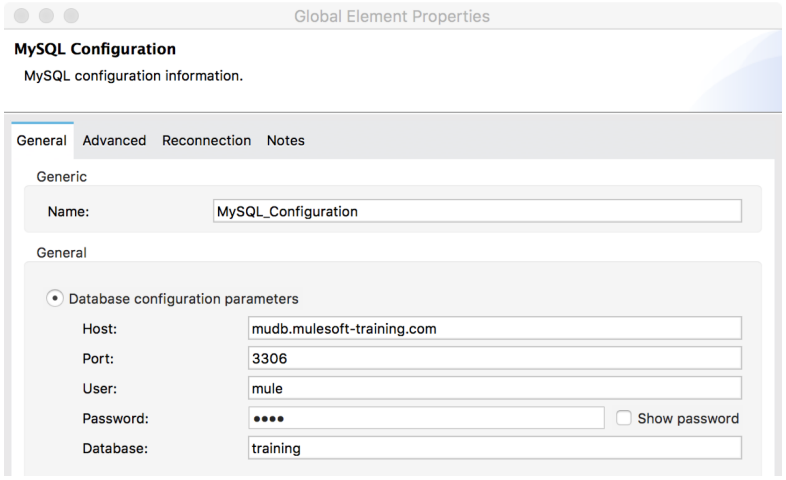
- Easy to install
- Requires minimal resources
- Can run multiple applications
- Uses a Java Service Wrapper that controls the JVM from the operating system and starts Mule
- Can be managed by
  - Runtime Manager in MuleSoft-hosted Anypoint Platform
  - Runtime Manager in customer-hosted Anypoint Platform
    - Anypoint Platform Private Cloud Edition

Preparing an application for deployment

Before deploying

MuleSoft

- Think about anything in your application that might change between development and production…



Using application properties

Application properties

MuleSoft

- Are an alternative to hard-coding credentials & resources
- Are injected into the application at runtime
- Provide an easier way to manage credentials, changes, and settings
- Can be encrypted
- Are defined in .properties files
  - Separate property files can host values specific to an environment
    - app-dev.properties and app-prod.properties

Defining application properties

MuleSoft

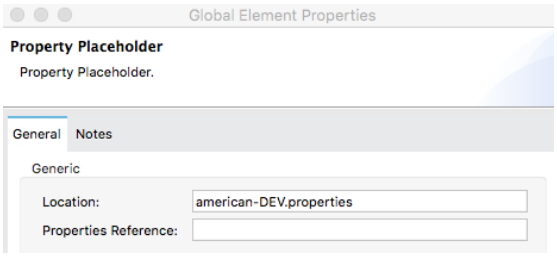
- Create a properties file in the src/main/resources folder

```
american-DEV.properties
```

- Define properties in the properties file

```
db.port = 3306
db.user = mule
```

- Create a Properties Placeholder global element

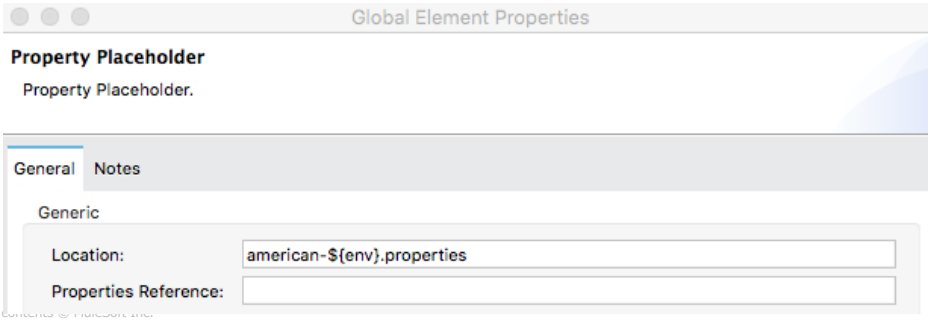


- Use the properties in the application

```
${db.account}
```

Dynamically specify property files

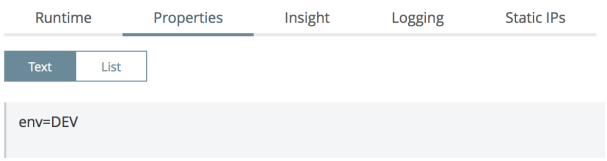
- Resources and credentials often vary from development to production environments
- You can use environment variables in an application whose values must be set when the application starts
- You can use a dynamic value for the location in the Property Placeholder



Setting environment variables

- For development, set environment variables in mule-app.properties

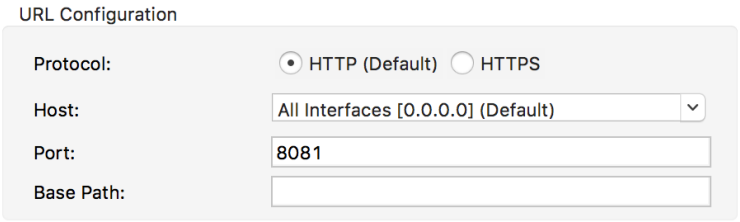
```
env = DEV
```



- For deployment to **CloudHub**
  - Set the values in the Runtime Manager properties tab for the application before you start it
  - If you deploy to CloudHub from Anypoint Studio, these values get automatically added as properties for the application in the Runtime Manager
- For deployment to a **customer-hosted Mule runtime**
  - Variables must be passed to Mule runtime when it starts
  - Set them in wrapper.conf before starting Mule

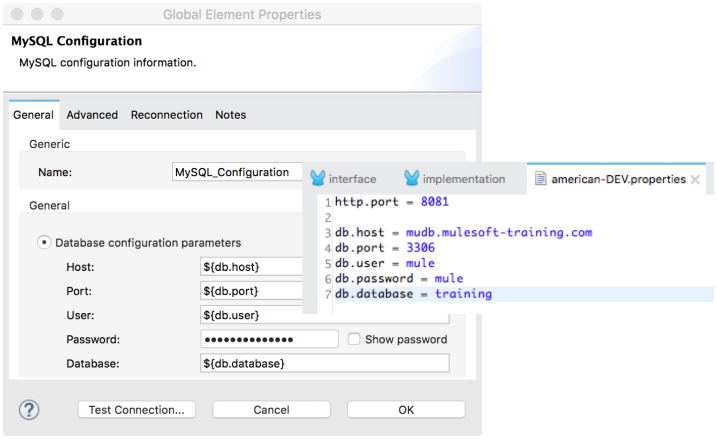
Do you have to modify HTTP Listener connectors?

- No
- CloudHub routes all requests to your application domain URL on port 80 to an endpoint with the matching path that was configured with a host of 0.0.0.0 and port 8081
- If you use a port other than 8081, you need to set the port in a reserved application property called http.port or https.port
  - Traffic on port 80 to a CloudHub application domain URL will then be routed to the port set by that property



Walkthrough 5-1: Prepare an application for deployment using properties

- Create a properties file for an application
- Create a Properties Placeholder element to specify the properties file
- Define and use Database connector properties
- Specify a properties file dynamically

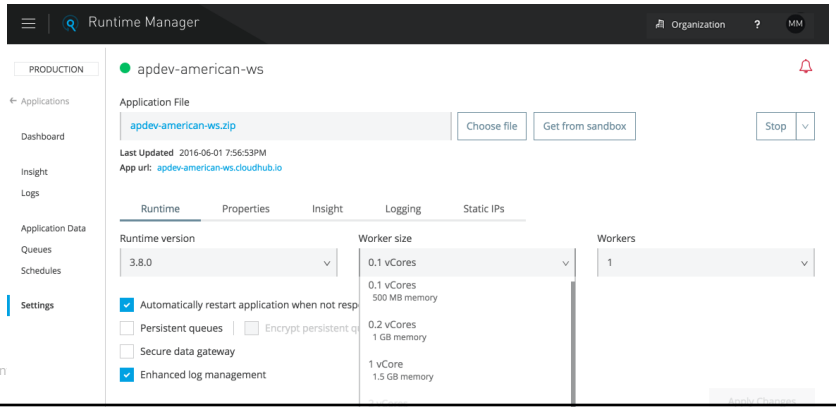


Deploying applications to CloudHub

Deploying applications to CloudHub

MuleSoft

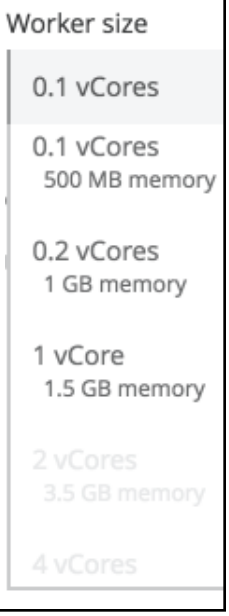
- Can deploy from Anypoint Studio or from Anypoint Platform using Runtime Manager
- You must set worker size and number
  - For apps deployed from flow designer, these values were set automatically



Recall: CloudHub workers

MuleSoft

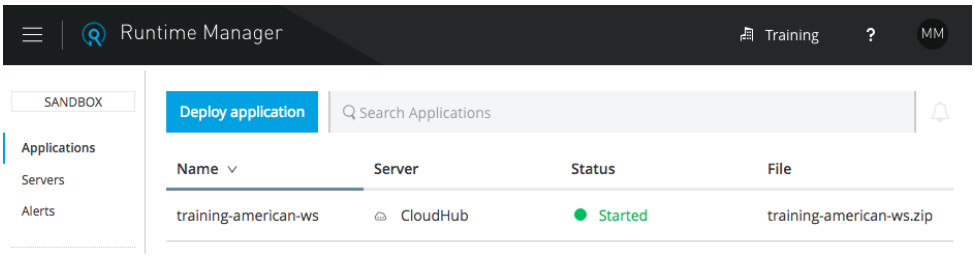
- A worker is a dedicated instance of Mule that runs an app
- Each worker
  - Runs in a separate container from every other application
  - Is deployed and monitored independently
  - Runs in a specific worker cloud in a region of the world
- Workers can have a different memory capacity and processing power
  - Applications can be scaled vertically by changing the worker size
  - Applications can be scaled horizontally by adding multiple workers



Walkthrough 5-2: Deploy an application to CloudHub

MuleSoft

- Deploy an application from Anypoint Studio to CloudHub
- Run the application on its new, hosted domain
- Make calls to the web service
- Update the API implementation deployed to CloudHub

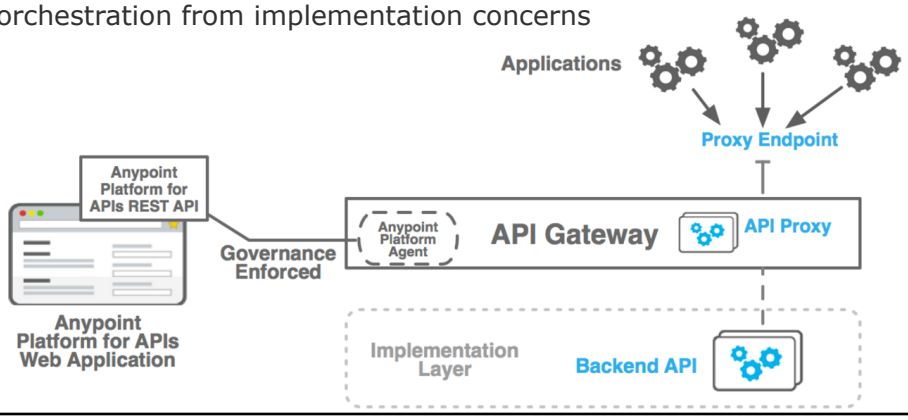


Creating API proxies

Restricting access to APIs

MuleSoft

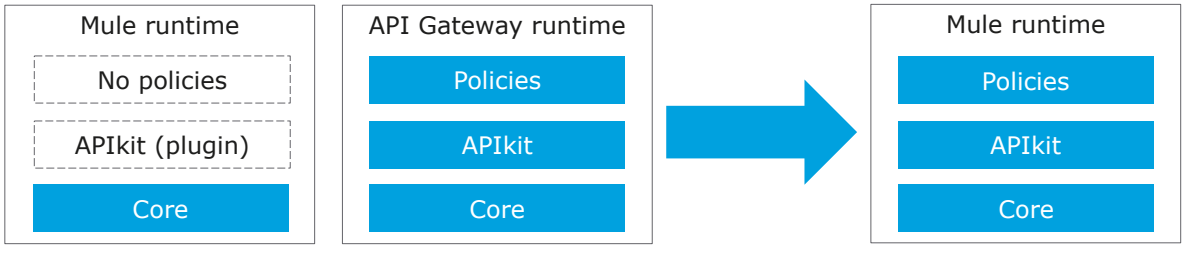
- An **API proxy** is an application that controls access to a web service, restricting access and usage through the use of an API gateway
- The **API Gateway** is a runtime designed and optimized to host an API or to open a connection to an API deployed to another runtime
  - Separates orchestration from implementation concerns



The API Gateway is the point of control

- **Determines which traffic** is authorized to pass through the API to backend services
- **Meters the traffic** flowing through
- **Logs** all transactions, collecting and tracking analytics data
- Applies runtime policies to **enforce governance** like rate limiting, throttling, and caching

Mule runtime includes API Gateway runtime



Pre Mule 3.8

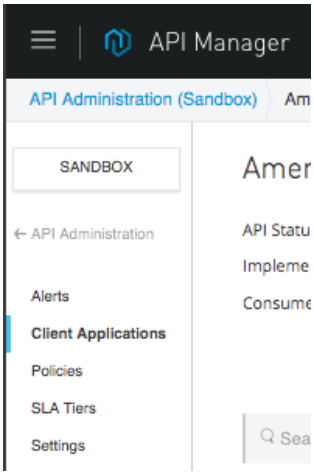
Mule 3.8+

* Mule runtime and API Gateway runtime require separate licenses

Using API Manager to manage access to APIs

MuleSoft

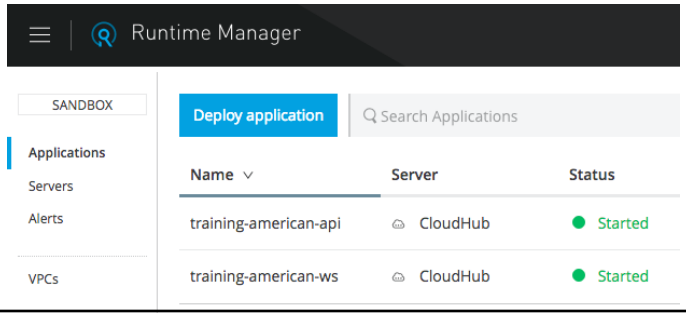
- **Create** proxy applications
- **Deploy** proxies to an API Gateway runtime
  - On CloudHub or a customer-hosted runtime
- Specify throttling, security, and other **policies**
- Specify **tiers** with different types of access
- Approve, reject, or revoke **access** to APIs by clients
- **Promote** managed APIs between environments
- Review **analytics**



Walkthrough 5-3: Create and deploy an API proxy

MuleSoft

- Add an API to API Manager
- Use API Manager to automatically create and deploy an API proxy app
- Set a label and consumer endpoint for a proxy so requests can be made to it from Exchange
- Make calls to the API proxy from API portals for both internal and external developers
- View API request data in API Manager

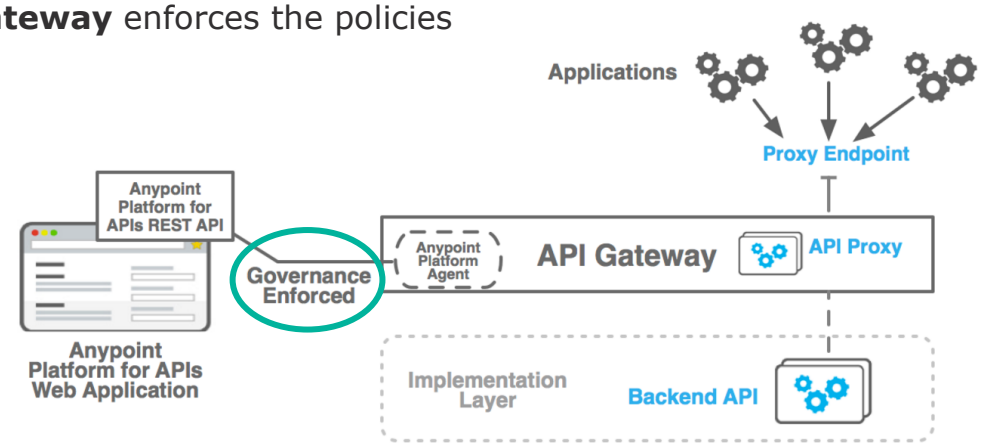


Managing access to APIs

Managing access to APIs

MuleSoft

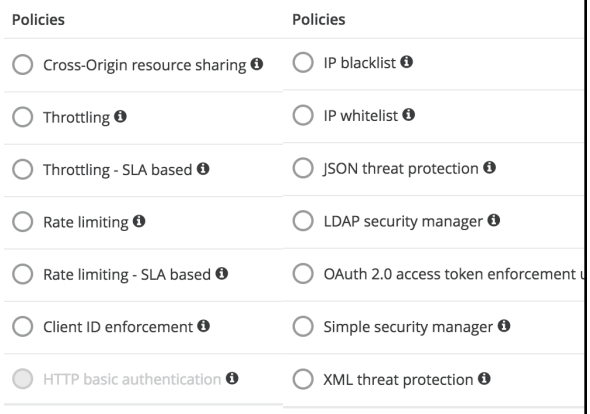
- Use **API Manager** to manage access to API proxies
  - Define SLA tiers
  - Apply runtime policies
- The **API Gateway** enforces the policies



Applying policies to restrict access

MuleSoft

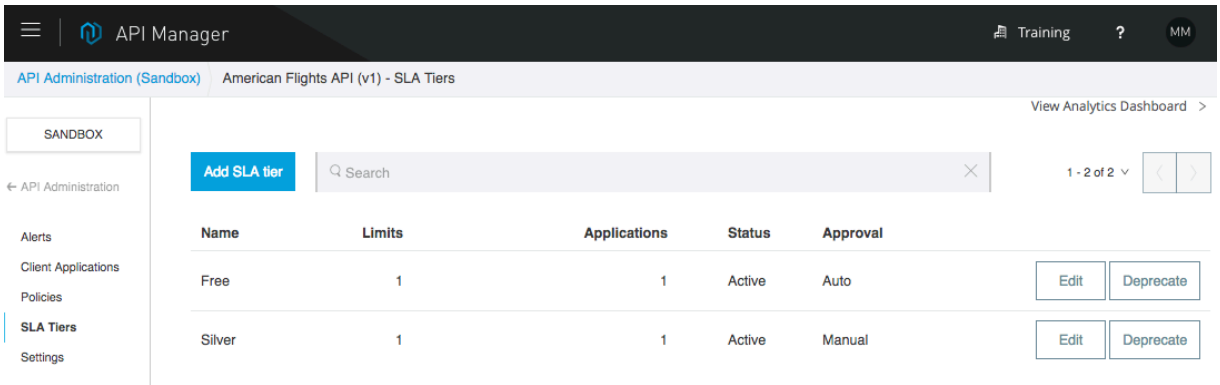
- There are **out-of-the box** policies for many common use cases
  - Rate limiting
  - Throttling
  - Security
- You can define **custom** policies (using XML and YAML)
- You can apply **multiple** policies and set the order



Using SLA tiers to restrict access by client ID

MuleSoft

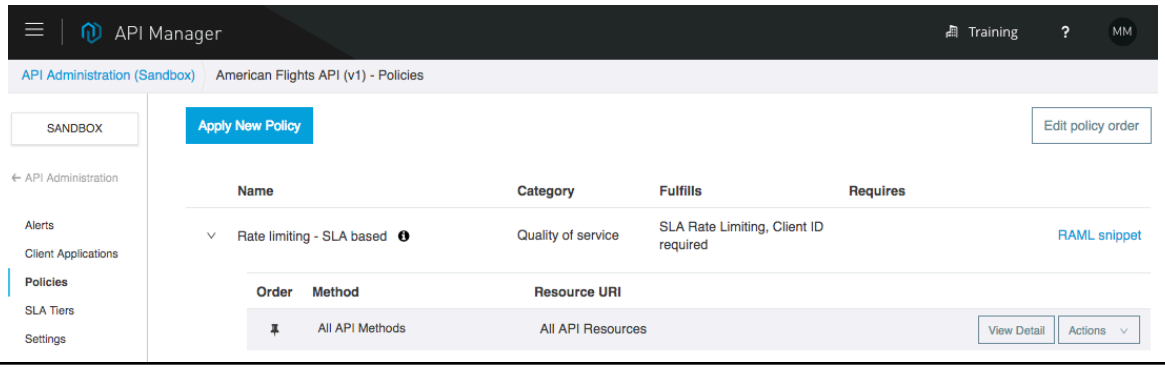
- A **S**ervice **L**evel **A**greement tier defines the # of requests that can be made per time frame to an API
  - Request approval can be set to automatic (free) or manual (for tiers that cost $)



Enforcing access to APIs using SLA tiers

MuleSoft

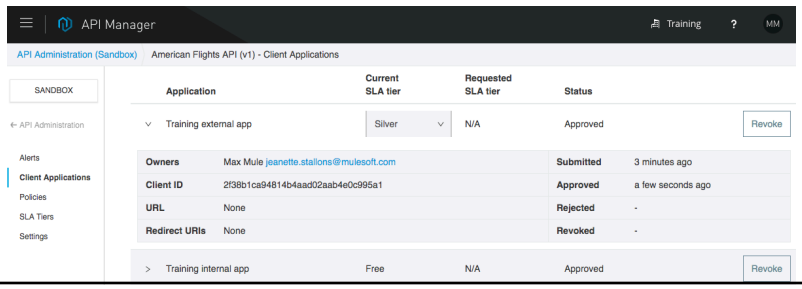
- To enforce, apply an **SLA based** rate limiting or throttling policy
- SLA based policies require all applications that consume the API to
  - **Register** for access to a specific tier
    - From an API portal in private or public Exchange
  - **Pass their client credentials** to calls made to the API



Walkthrough 5-4: Restrict API access with policies and SLAs

MuleSoft

- Add and test a rate limiting policy
- Add SLA tiers, one with manual approval required
- Add and test a rate limiting SLA policy
- Request application access to SLA tiers from private & public API portals
- Approve application requests to SLA tiers in API Manager

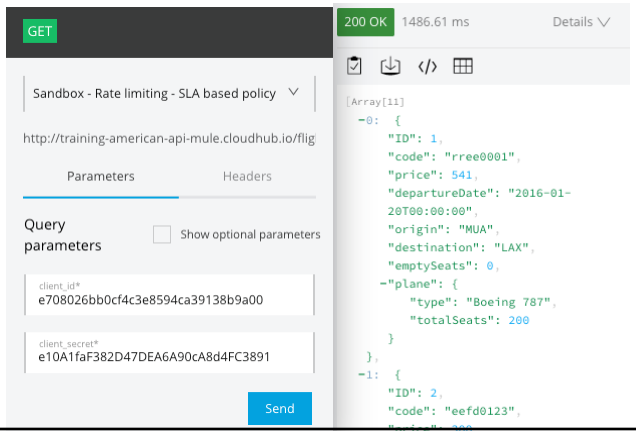


32

Making calls to APIs with client ID enforcement policies from API portals

Walkthrough 5-5: Make calls to an API with a client ID based policy from API portals

- Add authentication query parameters to an API specification
- Update a managed API to use a new version of an API specification
- Call a governed API with client credentials from API portals

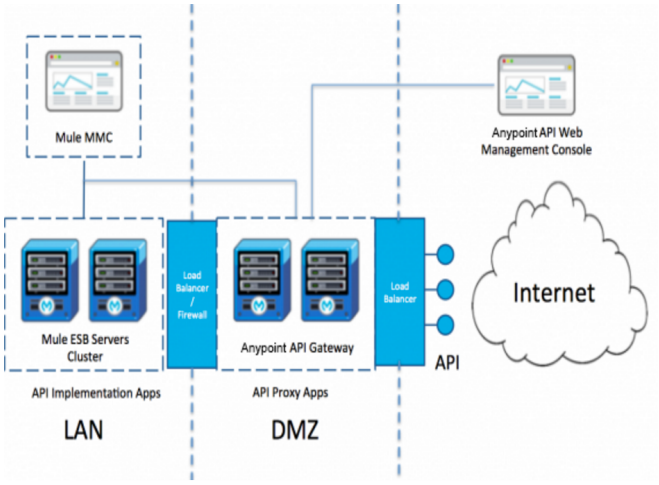


Restricting access to API implementations

On-prem API implementations and API proxies

MuleSoft

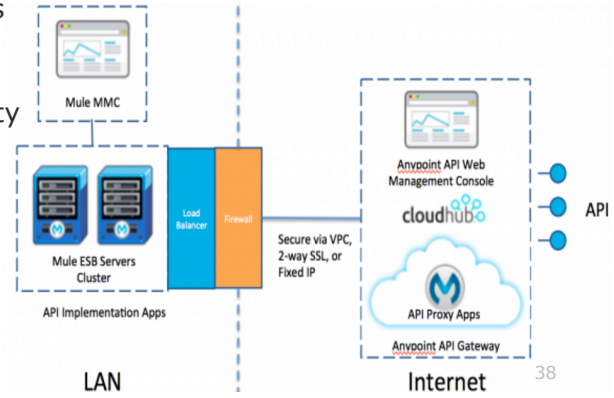
- Set up an API Gateway cluster inside a DMZ to run the API proxy applications



On-prem API implementations and cloud API proxies

MuleSoft

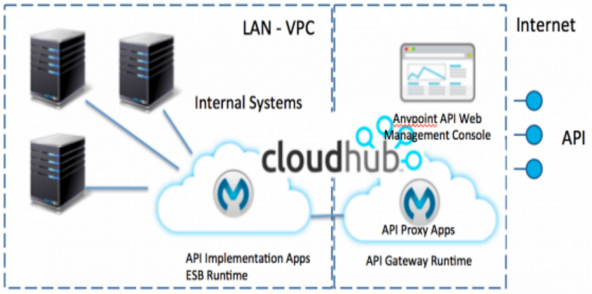
- Set up secure communication between the proxy applications and the internal on-prem runtimes using a Virtual Private Cloud (VPC)
  - A VPC is a private and isolated network of your CloudHub workers
- Connect this network to other VPCs or data centers via a VPN connection
  - This allows CloudHub workers to access resources behind a corporate firewall
  - You can use an IPSec gateway or AWS Direct Connect for VPN connectivity
- See here for setting up VPC https://docs.mulesoft.com/runtime-manager/virtual-private-cloud



Cloud API implementations and API proxies

MuleSoft

- Option 1
  - Do not use separate API proxy applications and instead specify policies for the service API implementation applications
- Option 2: Use VPC
  - Leave the workers running API proxy applications outside the VPC and put the workers running API implementations inside the VPC
  - Use ports 8091 or 8092 in your API implementations



Summary

Summary: Deployment

MuleSoft

- Deploy applications to MuleSoft-hosted or customer-hosted Mule runtimes
- **CloudHub** is the Platform as a Service (PaaS) component of Anypoint Platform
  - Hosted Mule runtimes (workers) on AWS
- Use **application properties** to avoid hard-coding endpoint properties, credentials, and resources
  - Define them in a .properties file whose location is specified in a Properties Placeholder global element
  - Dynamically specify a properties file when the application starts by parameterizing its name and setting the variable
    - As an application property with the Runtime Manager
    - As an argument in on-prem Mule runtime wrapper.conf file

Summary: Access control

- An **API proxy** is an application that controls access to a web service, restricting access and usage through the use of an API gateway
- The **API Gateway runtime** controls access to APIs by enforcing policies
  - Is part of the Mule runtime but requires a separate license
- Use **API Manager** to
  - Create and deploy API proxies
  - Define SLA tiers and apply runtime policies
    - Anypoint Platform has out-of-the box policies for rate-limiting, throttling, security enforcement, and more
    - SLA tiers defines # of requests that can be made per time to an API
  - Approve, reject, or revoke access to APIs by clients
  - Promote managed APIs between environments
  - Review API analytics

Anypoint Platform Operations training courses

- This module was just an introduction to deploying and managing applications and APIs

- Anypoint Platform Operations:
  - CloudHub (1 day in-person or 2 days online)
  - Customer-Hosted Runtimes (2 days)
  - API Management (1 day)