MuleSoft

Anypoint Platform Operations: Universal API Management

Introducing the course

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

MuleSoft

- Manage APIs deployed in Anypoint Platform or elsewhere in a single control plane using API Manager
- Use Anypoint Exchange and public portals to document, version, and share API specifications
- Control and secure APIs running anywhere using Anypoint Flex Gateway
- Control and secure APIs running in Anypoint Platform using Mule Gateway
- Govern APIs with policies and SLA tiers
- Ensure consistent quality and compliance across all APIs using Anypoint API Governance
- Analyze and monitor APIs using Anypoint Monitoring
- Improve the reliability, performance, and value of APIs using Anypoint Analytics

How the course will work

MuleSoft

- Is primarily hands-on
- Consists of
  - Short lectures (PPT) to introduce a concept
  - Walkthroughs
    - The bulk of class
    - Exercises we do together to learn the content

Course outline

MuleSoft

- Module 1: Introducing universal API management on Anypoint Platform
- Module 2: Discovering and consuming APIs
- Module 3: Managing APIs using Anypoint Flex Gateway
- Module 4: Managing APIs using Anypoint Mule Gateway
- Module 5: Enabling API Governance and enhancing security
- Module 6: Governing APIs with policies and SLA tiers
- Module 7: Versioning managed APIs
- Module 8: Monitoring APIs

MuleSoft

Module 1: Introducing universal API management on Anypoint Platform

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

MuleSoft

- List and describe the functionality included in universal API management on Anypoint Platform
- Describe the gateway options for managing APIs on Anypoint Platform
- Navigate Anypoint Platform and Anypoint API Manager

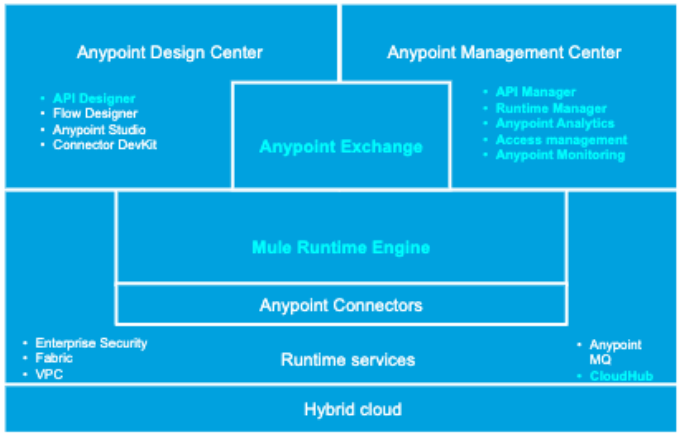
Introducing Anypoint Platform

Anypoint Platform

MuleSoft

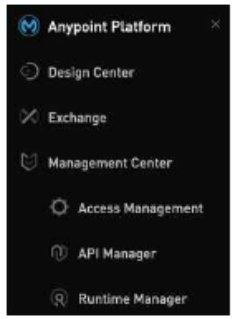
- A **unified**, **highly productive**, **hybrid** integration platform that creates a seamless **application network** of apps, data, and devices by producing and consuming reusable assets
- A collection of runtimes, frameworks, tools, and web applications
  - Tools and frameworks for building applications
  - **Mule runtime** for running applications and applying policies
    - Customer-hosted or MuleSoft-hosted runtimes
  - Web applications for
    - Organizing users into teams and assigning permissions
    - Business groups for resource allocation and grouping of assets
    - Deploying, running, managing, and monitoring applications
    - Defining, managing, discovering, and producing APIs

Anypoint Platform: The components



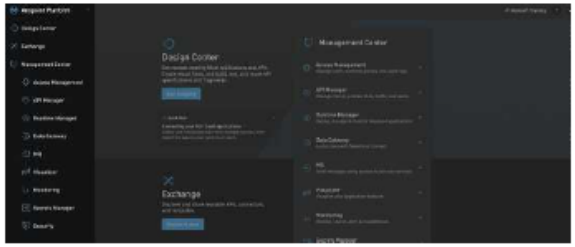
Core functionality of Anypoint Platform related to APIs

- Management Center
  - Manage teams, users, permissions and business groups (Access Management)
  - Manage API instances (API Manager)
  - Deploy apps to MuleSoft-hosted or customer-hosted runtimes (Runtime Manager)
  - Manage and monitor applications (Runtime Manager)
- Design Mule applications, API specifications, and fragments (Design Center)
- Share assets like APIs, examples, connectors, and templates (Anypoint Exchange)
- Publish and document APIs and other assets (Anypoint Exchange)
- Test and simulate APIs (Anypoint Exchange)



Anypoint Platform: The web application

- Availability options
  - In a MuleSoft-hosted cloud environment at
    - https://anypoint.mulesoft.com (U.S. Platform)
    - https://eu1.anypoint.mulesoft.com (EU Platform)
  - In a customer-hosted environment as part of Anypoint Platform Private Cloud Edition



Checking the status of Anypoint Platform services

- https://status.mulesoft.com (U.S. Platform)
- https://eu1-status.mulesoft.com (EU Platform)
- Subscribe to status updates

Performing administrative tasks in Anypoint Platform

- Access management provides administrative and organizational abilities that apply to the various entitlements in Anypoint Platform
  - API Manager, Runtime Manager, Anypoint MQ, and more
- Features of access management includes
  - Managing teams, organizations, users, permissions, and environments
  - Monitoring subscriptions and audit logs

Introducing access management terminology

- Organization – An administrative collection of resources
- Business Group – A child organization
- Users – Anypoint Platform accounts with access to an organization
- Teams - Collections of users
- Environments – Divide up resources within a business group
- Permissions – Define access to resources or assets
  - Teams or users are assigned permissions
  - May be specific to an environment or more generally for a business group

Permission management with teams and business groups

MuleSoft

- Business groups are used for resource grouping and allocation
  - Environments and vCores
  - Exchange assets
- Teams are used to group users reflecting the company structure
  - Allow hierarchical structure
  - All users belong to the Everyone team
  - Child teams inherit permissions from the parent team
- Permissions determine the access level per business product
  - Assign global (Everyone) or team-level permissions

Walkthrough 1-1: Explore Anypoint Platform

MuleSoft

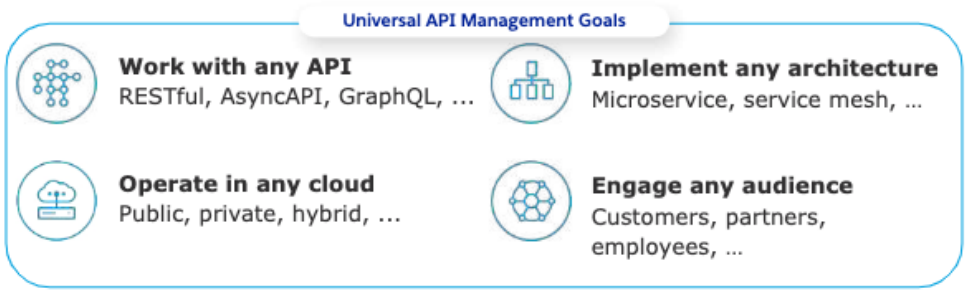
- Log in to Anypoint Platform
- Navigate Anypoint Platform
- Explore support, documentation, and resources

Introducing universal API management on Anypoint Platform

Goals of universal API management (UAPIM)

MuleSoft

Universal API management is the ability to open the scope of full lifecycle management capabilities — **management**, **governance**, **observability**, and **discoverability** — to APIs built and deployed anywhere



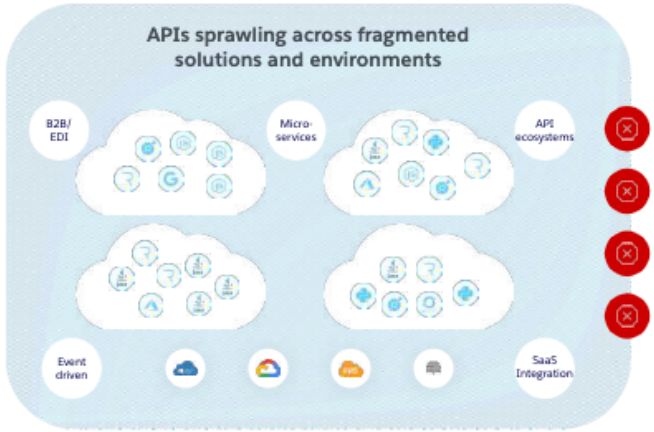
The benefits of universality for API management

MuleSoft

1. **Enabling a digital ecosystem** through universal discovery and a central catalog of all APIs across an organization
2. **Securing data and reducing risk** through universal compliance to standards and best practices, secure API gateways, and ubiquitous operational visibility
3. **Enhancing business agility** by enabling developers to use their environment or architecture of choice
4. **Future-proofing** through investments in open, flexible, and scalable technology platforms that are compatible with both future innovations and previous investments

API sprawl is a barrier to universality

MuleSoft



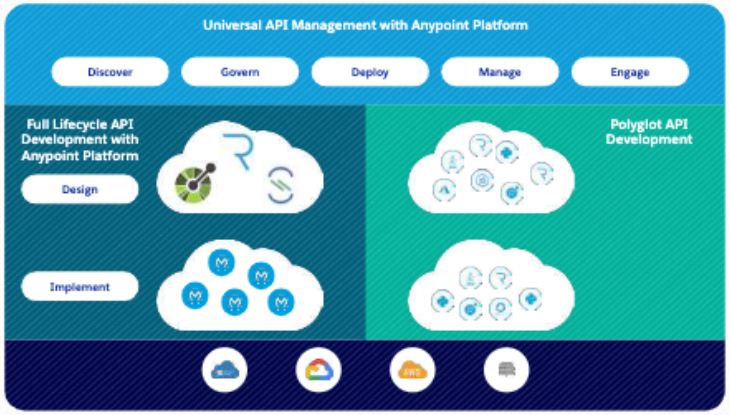
- Limited visibility & access to existing APIs
- Inconsistent enforcement of security and governance
- Slowed pace of change and siloed best practices
- Complex operations and troubleshooting

Universal API management on Anypoint Platform

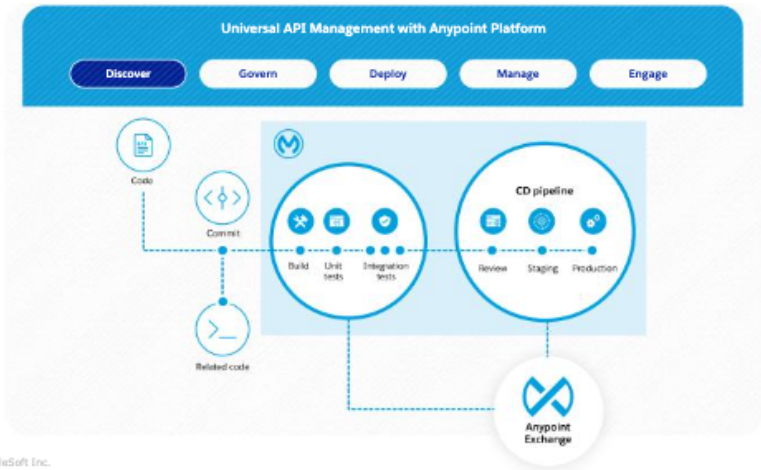
Universal API management on Anypoint Platform is a set of MuleSoft product capabilities that empowers enterprises to manage the full lifecycle of all of their APIs using a single control plane for comprehensive visibility, flexible management, and consistent governance at scale

Developer	API owner	Architect	API Product Manager
Work with any API, built anywhere	**Secure any service, deployed anywhere**	**Manage and govern consistently**	**Create vibrant API ecosystems**
Anypoint API Designer	Anypoint Mule Gateway	Anypoint API Manager	Anypoint Exchange
API Catalog CLI	Anypoint Flex Gateway	Anypoint API Governance	Anypoint API Community Manager

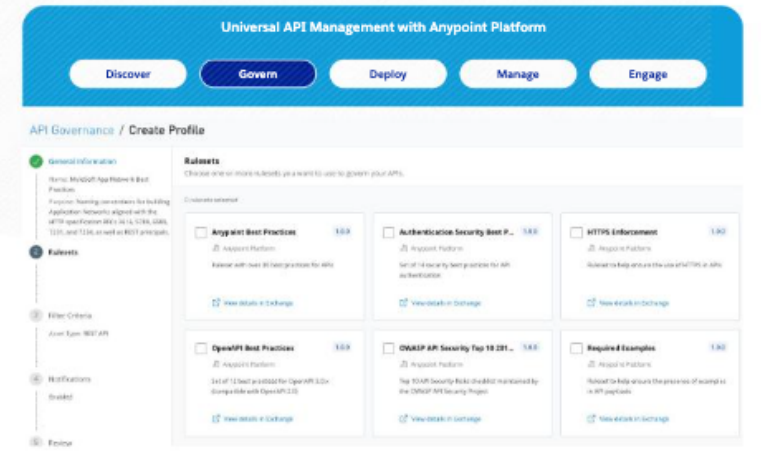
Universal API management + full lifecycle API development with Anypoint Platform



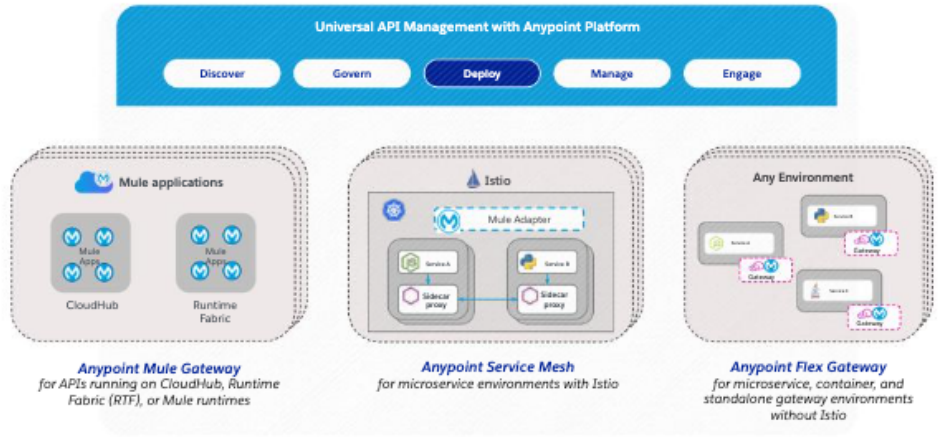
Discover APIs with Exchange and API Catalog CLI



Govern APIs with Anypoint API Governance



Deploy API proxies with multiple runtime options

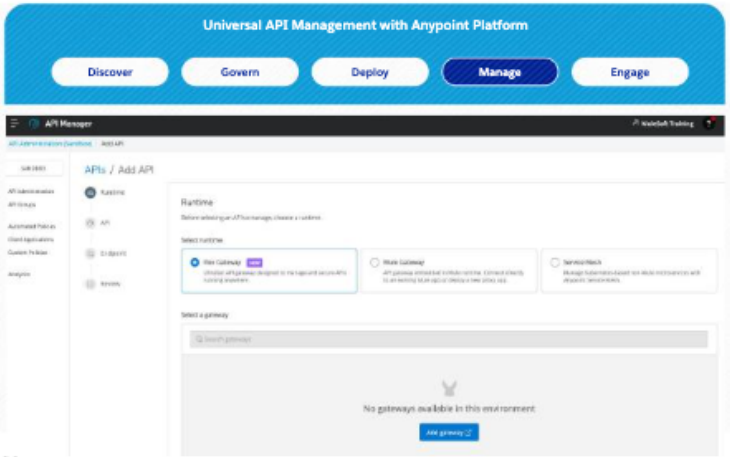


Anypoint Mule Gateway
for APIs running on CloudHub, Runtime Fabric (RTF), or Mule runtimes

Anypoint Service Mesh
for microservice environments with Istio

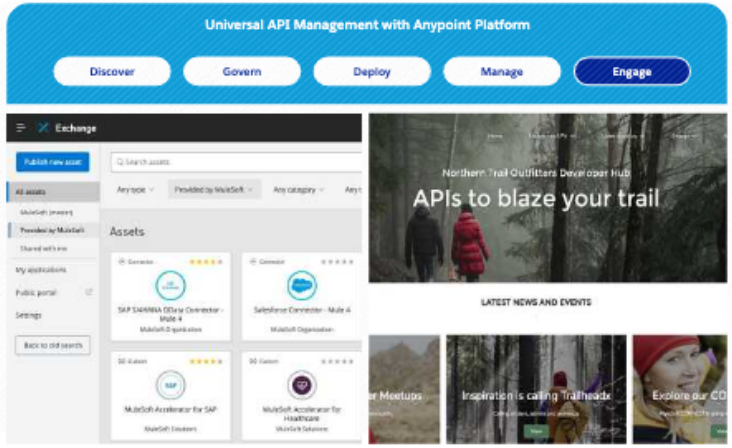
Anypoint Flex Gateway
for microservice, container, and standalone gateway environments without Istio

Manage APIs with Anypoint API Manager



Engage API consumers with Exchange and Anypoint API Community Manager

MuleSoft

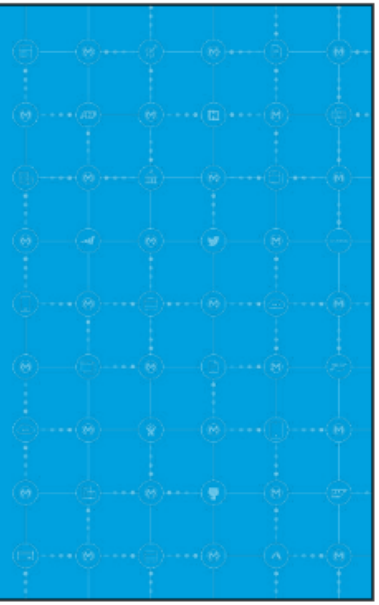


Comparing UAPIM proxy runtime options

MuleSoft

	Anypoint Mule Gateway	Anypoint Service Mesh	Anypoint Flex Gateway
Architecture Pattern	API running on CloudHub, RTF, or Mule runtime	Microservice environment **with Istio**	Microservice, container, and standalone gateway environment **without Istio**
Use Cases	Traditional Mule application & integrations	Service mesh microservices strategy **with Istio**	Standalone gateway, distributed microservices, or Polyglot integration ecosystem - **without Istio**
Role	MuleSoft Developer	Kubernetes Administrator	Any Developer
API Implementation	Mule runtime	Polyglot	Polyglot

Summary



Summary

MuleSoft

- Anypoint Platform is a connectivity platform for connecting any app, data source, device, and API – both in MuleSoft-hosted or customer-hosted runtimes
- Anypoint Platform is a collection of servers, frameworks, tools, and web applications for building, running, managing, and monitoring integration applications and APIs
- Anypoint Platform has a full suite of capabilities for managing the entire API lifecycle
  - Design, build, deploy, manage, and govern
- Universal API management on Anypoint Platform is a set of MuleSoft product capabilities that together provide full lifecycle management capabilities to APIs that are built and deployed anywhere in any architecture or environment

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

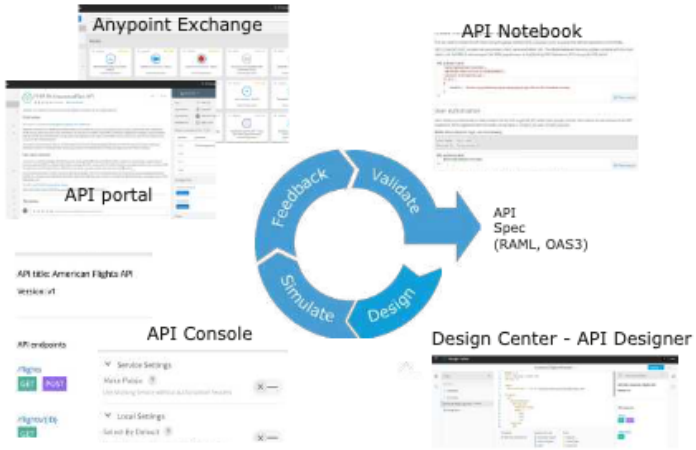
MuleSoft

- Create APIs in Anypoint Design Center using RAML specifications
- Publish, document, and test API specifications in Anypoint Exchange
- Control API access using Anypoint Exchange portals
- Discover APIs through public portals
- Catalog APIs with Anypoint CLI
- Describe how to extend the capabilities of Anypoint Exchange with API Community Manager

Creating API specifications in Anypoint Platform

API development cycle: API specification

MuleSoft



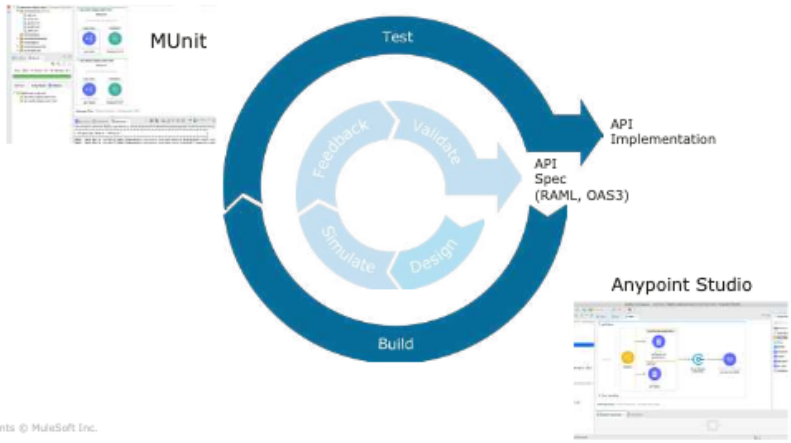
The role of an API specification

MuleSoft

- The **API specification** describes **everything required** of an API client to make requests to **any** API implementation
  - The names of **resource endpoints** and their HTTP **methods**
  - The names and structure of any request and response **parameters** and **payloads**
  - **Required** vs. **optional** parameters
  - The types of **response codes** and the structure of **response data**

API development lifecycle: API implementation

MuleSoft



The role of an API implementation

MuleSoft

- An API implementation
  - Runs on a network that can accept requests from API clients on API endpoints
  - Is **built** to implement the API specification's resource endpoints
    - For invalid API requests, the API implementation can use the API spec to perform validation to "fail fast" and return an error response back to the API client with the context of the failure according to the API specification
  - For valid API requests, carries out the integration, business logic and other transformation logic to perform each method from an API specification, including interactions with any related backend systems
  - Sends back responses consistent with the API specification, either success or errors

Introducing RAML: RESTful API Modeling Language

MuleSoft

- A non-proprietary, vendor-neutral open spec
- A simple, structured, and succinct way of describing RESTful APIs
  - Describes **resources** containing **methods**
  - Method request and response parameters
  - Data schema and other API-related details
  - Can import and combine API fragments

RAML
http://raml.org

- Designed as a top-down specification
  - Breaks down systems and explains the behavior of the various sub-components
  - Helps out the current API ecosystem by encouraging Spec-Driven Development
  - Encourages reuse, enables discovery and pattern-sharing, and promotes the merit-based emergence of best practices

Introducing OAS: OpenAPI Specification Language

MuleSoft

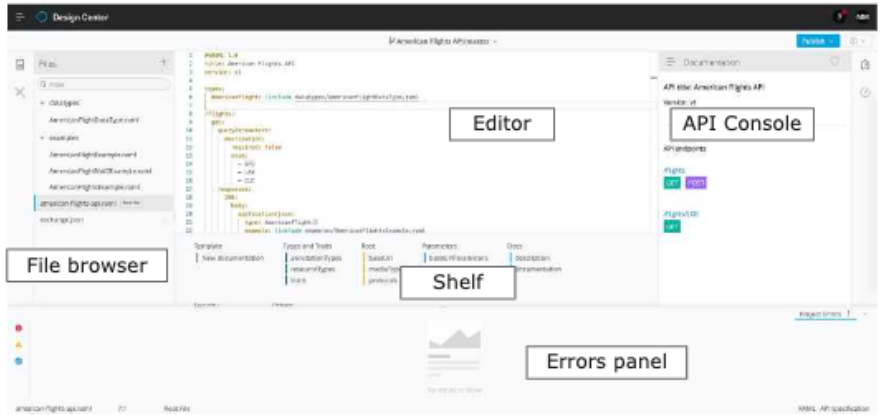
- Standard, programming **language-agnostic interface description language** for REST APIs
- Comparable to RAML
- Can be converted to RAML via AML Modeling Framework (**AMF**)
- Can be written in **YAML** or **JSON** format
- Documentation of **REST API** methods, parameters, and models
- Can be used to create client and server stubs
- Anypoint Platform support
  - Full support for **OAS 2.0**
  - Support for **OAS 3.0** (See documentation)

OPENAPI INITIATIVE
https://www.openapis.org/

Creating API specifications in RAML 1.0 using API Designer

MuleSoft

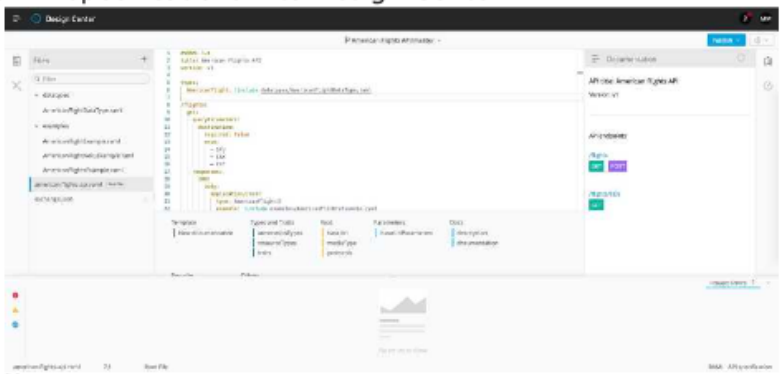
- API Designer is a part of the Design Center entitlement



Walkthrough 2-1: Create API specifications in Design Center

MuleSoft

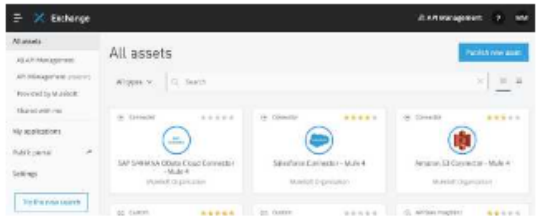
- Create new API specification projects
- Import API specifications into Design Center



Consuming assets from Anypoint Exchange

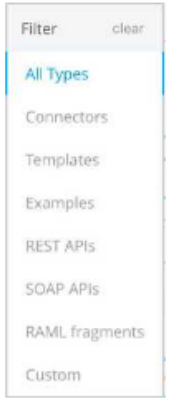
What does Anypoint Exchange contain?

- **Anypoint Exchange** is designed to support the **Center for Enablement** (C4E) concept
  - A shared place where all the assets are published for self-service and reuse by different parts of the organization, now and in the future
  - Each asset is a Maven artifact and contains an asset version
  - Can house how-to and best practices documentation, API specifications, API fragments, templates, connectors, and examples



Anypoint Exchange can contain both public and private assets

- MuleSoft-provided **public** assets are available in all accounts to all Anypoint Platform users
  - You can work with MuleSoft to get connectors you build certified and added
- **Private** content is only available to people in a particular Anypoint Platform organization
  - Assets added by anyone in an Anypoint Platform organization are added to that organization's private Exchange
- The Anypoint Platform organization should populate the private Exchange to contain everything needed to build integration projects
  - Including documentation, diagrams, videos, links, and more

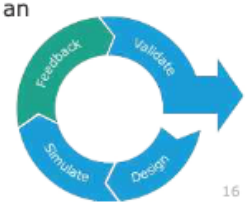


Publishing and documenting APIs using Anypoint Exchange

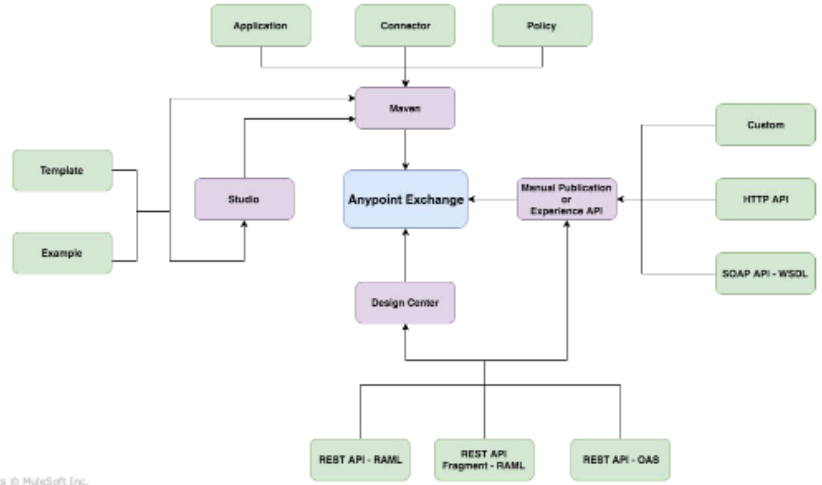
Engaging users through the API lifecycle

MuleSoft

- To build a successful API, you should define it iteratively
  - Get feedback from developers on usability and functionality along the way starting from the design phase
- To do this, provide ways for developers to discover and hit the endpoints in your API
  - Anypoint Exchange promotes discoverability of APIs
- Anypoint Platform makes this easy with Anypoint Exchange portals
  - Anypoint Exchange portals allow developers to learn about an API and try it out
  - Developers can request access to any APIs in an Exchange portal

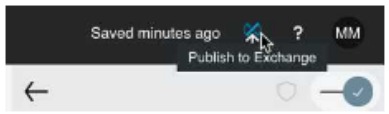


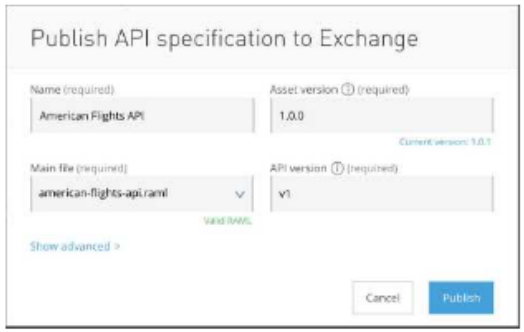
How different assets are published to Anypoint Exchange



Publish API specifications/fragments to Anypoint Exchange from API Designer

- RAML specifications, OAS API specifications, and RAML fragments can be published to Anypoint Exchange from **API Designer**
- **API version** is specified (v1, v2, etc.)
- A unique, sequential, semantic **version of the asset** should also be specified
  - 1.0.0, 1.0.1, 1.0.2, 1.1.0, 2.0.0





Publish APIs to Anypoint Exchange using Exchange API (1/2)

- Use the Exchange experience API to create, share, search, and manage assets in Anypoint Exchange
  - The public portal for the Exchange API can be accessed here
- Asset documentation can also be created and updated using the Exchange API
  - Upload resources, create draft pages, publish drafts
- Authentication methods before executing requests
  - An access token for an Anypoint Platform user
  - Use connected application authentication
- Assets can be published synchronously or asynchronously

Publish APIs to Anypoint Exchange using Exchange API (2/2)

- Example cURL request to publish

```
curl -v \
  -H 'Authorization: bearer {access-token}' \
  -H 'x-sync-publication: true' \
  -F 'name=American Flights System API' \
  -F 'properties.mainFile=american-flights-system-api.raml' \
  -F 'properties.apiVersion=v1' \
  -F 'files.raml.zip=@{path}/american-flights-system-api-1.0.2-raml.zip' \
https://anypoint.mulesoft.com/exchange/api/v2/organizations/{org-id}/assets/{group-id}/
american-flight-system-sapi/1.0.2
```

  - Placeholders in brackets should be substituted with their corresponding values depending on the Anypoint Platform account used

Publish APIs to Anypoint Exchange using Anypoint CLI

MuleSoft

- Anypoint CLI is a convenient command-line tool provided by Anypoint Platform
- Prerequisites and installation guidelines can be found in the docs
- Authentication methods before executing requests
  - Username and password directly
  - Connected app credentials
  - An access token for an Anypoint Platform user
- Example

```
exchange asset upload --classifier raml --apiVersion v1 --name AmericanFlights
--mainFile american-flights-system-api.raml american-flights-system-api/1.0.2
american-flights-system-api-1.0.2-raml.zip
```


Other ways to publish assets to Anypoint Exchange

MuleSoft

- **Mule Maven plugin** allows publishing APIs as part of the Maven lifecyle phases
  - Supports applications, connectors, policies, templates, and examples
  - To publish, configure the pom.xml and settings.xml files as described in the docs; then run

```
mvn deploy
```

  - **Mutable data** such as tags, custom field, categories, and documentation pages can also be included in the same request (See docs)
- **Manually** through the Anypoint Exchange menus to publish new assets
  - Supports API specs such as RAML, RAML fragments, OAS, WSDL
  - Also supports custom assets such as images, text, documents, or compressed files

Creating API portals in Anypoint Exchange

Introducing portals for REST APIs in Exchange

MuleSoft

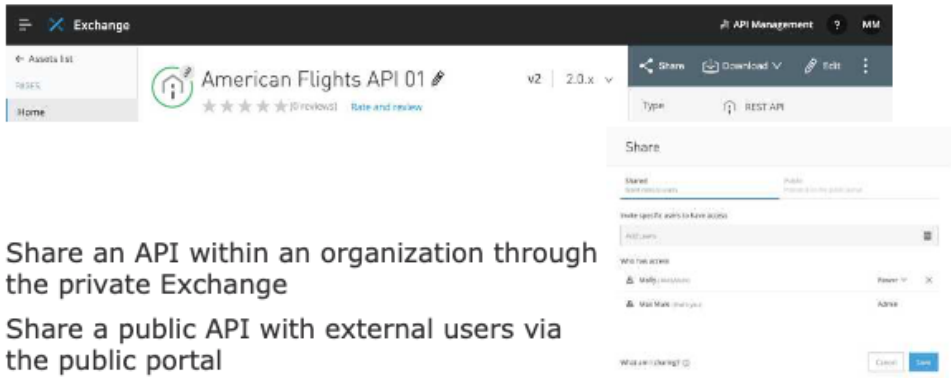
- **API portals** are automatically created for REST APIs published in Exchange
- An API portal has
  - An **automatically generated API endpoint** that uses a **mocking service** to allow the API to be tested without having to implement it
  - An **API Console** that provides a REST client GUI for consuming and testing mocked or real API endpoints, with sample values already filled in
  - A **REST Connect** generated connector that can be used as an **API client** in Mule applications
- **Public portals** are accessible to anyone without logging in
  - Appearance is customizable, e.g., logo, image, text, favicon
- **Private portals** make it easy to securely share API assets with internal developers inside your organization

Customizing API portals in Anypoint Exchange

- After publishing an asset to Anypoint Exchange, it can be enhanced with descriptions, tags, videos, links, and more
  - Text can be written in GitHub Flavored Markdown https://help.github.com/categories/writing-on-github/
  - A visual editor can be used for a WYSIWYG experience
  - Additional pages and terms and conditions can also be added

Sharing Anypoint Exchange portals

- You can share an API in Anypoint Exchange with other internal or external users



- Share an API within an organization through the private Exchange
- Share a public API with external users via the public portal

Walkthrough 2-2: Document, test, and share an Anypoint Exchange asset

MuleSoft

- Publish an API to Anypoint Exchange
- Use Markdown language to add content to an Exchange portal
- Publish and view an updated Exchange portal
- Test an API using the API Console in the API's Exchange portal
- Add a private Anypoint Exchange asset into a public portal

Cataloging APIs with Anypoint CLI

Anypoint API Catalog CLI

MuleSoft

- Designed by MuleSoft to help developers automatically discover API specifications built anywhere
- APIs are cataloged and made available in Anypoint Exchange
- Anypoint API Catalog CLI makes cataloging API specifications automatic with a CI/CD plugin
- Standardizes APIs to get them ready for universal consumption

Using the Anypoint API Catalog CLI

MuleSoft

- Prerequisites
  - Latest long-term support version of Node.js
  - Node Package Manager (NPM)
  - User with **API Catalog Contributor** permission
  - If authenticating via connected app, set **View Environment** scope
- Run the command to install the CLI

```
npm install -g api-catalog-cli@latest
```

- Create a descriptor file

```
api-catalog create-descriptor -d descriptor-filename.yaml --external
```

- Publish assets to Exchange

```
api-catalog publish-asset -d catalog.yaml --host=anypoint.mulesoft.com
--organization=<org-id> --client_id=client_id --client_secret
```

Extending the capabilities of Anypoint Exchange with API Community Manager

What is API Community Manager (ACM)?

MuleSoft

- Powered by Salesforce Experience Cloud
- Customization, branding, marketing, and engagement capabilities

Comparing Anypoint Exchange with ACM

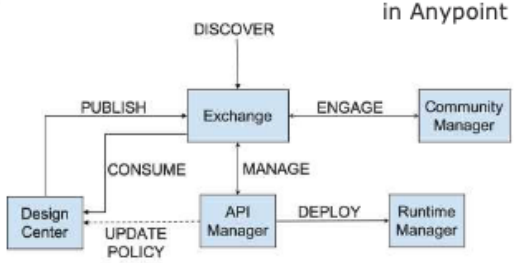
MuleSoft

Anypoint Exchange

- Auto-generated portals
- Discoverable catalog
- Lightweight branding
- Interacts with Design Center and API Manager

API Community Manager

- Branded, personalized experiences
- Forums, blogs, events
- Supports case management
- Engages with APIs already hosted in Anypoint Exchange



With API Community Manager (ACM), you can

MuleSoft

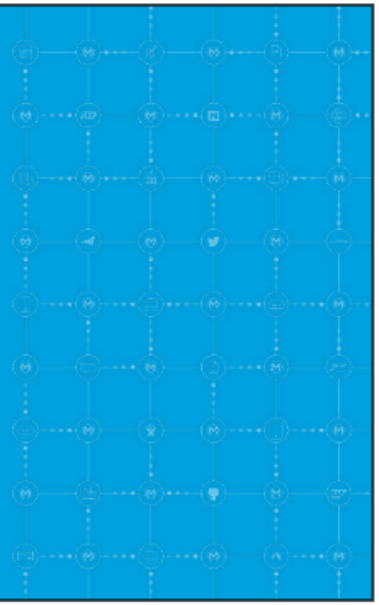
- Style the community to match company branding
- Build portals in minutes with out-of-the-box templates
- Customize and personalize experiences based on API consumer needs
- Auto-populate interactive documentation from Anypoint Exchange
- Support community members with developer forums, chat, and case management
- Measure ecosystem engagement and track API program KPIs in real time

Unique features of ACM

MuleSoft

- **Prebuilt themes** and Lightning components
- **Self-registration** of members
- **Forums** with moderation capabilities
- **Audience targeting** with fine-grained visibility rules
- Out-of-the-box **dashboards** that reflect best practices in community engagement
- Advanced **CSS** styling
- **Authoring** content
  - Internally using ACM stored in Salesforce Content Management System (CMS)
  - Externally using tools such as Adobe Experience Manager, Drupal, or Wordpress

Summary



Summary

MuleSoft

- Anypoint Exchange is a shareable online library of assets like APIs, connectors, templates, examples, and more
- Make an API discoverable by adding it to Anypoint Exchange
  - Everything you add is added to your private Exchange
  - You share REST APIs outside your private organization by adding them to public portals
- Portals in Exchange help document and test REST APIs
  - You can add Markdown or WYSIWYG content into portals
  - Portal pages can contain graphics, text, links, videos, and more

Summary

MuleSoft

- Anypoint Exchange is at the core of the Center for Enablement (C4E) concept
- It is a place where all the assets are published for self-service and reuse by different parts of the organization
- Anypoint Exchange can house how-to and best practices documentation, API specifications, API fragments, templates, connectors, examples, and more
- Anypoint Catalog CLI helps promote a centralized hub for saving and reusing APIs throughout the world
- API Community Manager (ACM) can help create engaging experiences with API communities that allow for high customization and branding

MuleSoft

Module 3: Managing APIs using Anypoint Flex Gateway

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

MuleSoft

- Install and register Flex Gateway
- Expose APIs through Flex Gateway
- Apply policies to APIs managed by Flex Gateway
- Add external log forwarding to Flex Gateway
- Scale Flex Gateway using replicas
- Secure Flex Gateway using TLS

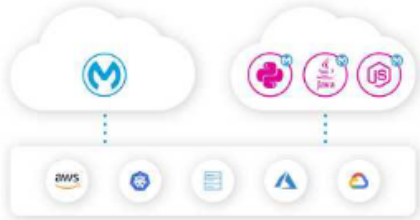
Introducing Anypoint Flex Gateway

Introducing Anypoint Flex Gateway

MuleSoft

Anypoint Flex Gateway is an ultrafast API gateway designed to manage and secure APIs running anywhere. Use Anypoint Flex Gateway to implement modern architectures and

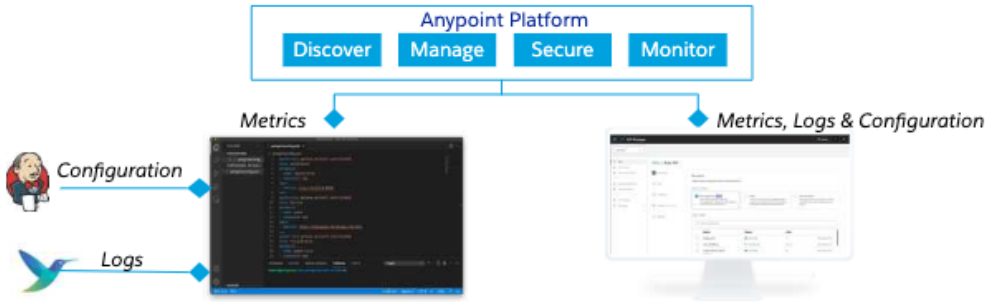
- **Build responsive experiences**
  Reduce application response times and lower costs with high performance on a small footprint
- **Extend and centralize management**
  Manage any service — no matter the size, language, or cloud — with one flexible gateway
- **Deploy with flexibility and efficiency**
  Deploy to virtually any target using the Anypoint Platform control plane or via CI/CD with declarative configurations



Flexible control through local and connected modes

MuleSoft

Control your **Anypoint Flex Gateway** configuration from Anypoint Platform in **Connected Mode** or configure your gateways using your existing CI/CD pipelines in **Local Mode**



Flex Gateway in Local Mode

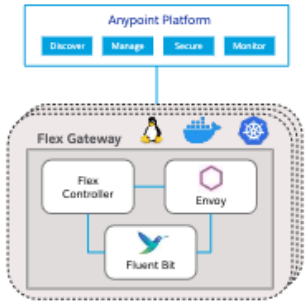
Manage locally with declarative configuration files to seamlessly integrate with existing CI/CD workflows and logging systems

Flex Gateway in Connected Mode

Use Anypoint Platform to centrally manage, secure, and observe all APIs across the enterprise

Flex Gateway components and deployment options

MuleSoft



Anypoint Flex Gateway consists of a lightweight **Envoy** proxy distribution accompanied by two main processes

- A **Controller** process that manages and translates configurations into Envoy-specific formats, either by connecting to Anypoint Platform in *Connected Mode* or reading configuration files in *Local mode*
- A **Fluent Bit** log forwarder process that can be configured to send both runtime and access logs to Anypoint Platform or any Fluent Bit output destination

Developers can deploy Flex Gateways virtually anywhere, including as a **Linux service**, **Docker container** or **Kubernetes Ingress controller** *(in Local Mode)*

Comparing Anypoint gateway options

MuleSoft

	Anypoint Mule Gateway	Anypoint Flex Gateway
Supporting Technology	Java Spring application on Mule runtime	Envoy, Fluent Bit (logging), Redis (optional shared object storage)
Key Capabilities	Use the same technology to both manage APIs and implement them as Mule applications	Support high performance, polyglot API management use cases
Policies	• Apply included or custom policies using the Anypoint Platform control plane • Use Maven to build and deploy Mule 4 custom policies	• Apply included or custom policies using the Anypoint Platform control plane (in Connected Mode) or by using declarative configuration files (in Local Mode) • Create custom policies via WebAssembly (Wasm) extensions that run on Envoy as custom filters *(Policy Development Kit planned)*
Deployment Options	MuleSoft-hosted or customer-hosted	Customer-hosted *(MuleSoft-hosted planned)*

Flex Gateway usage reports

MuleSoft

- **Usage reports**
  - Found at the root organization level
  - Show monthly API calls for all registered Flex Gateways (running in either Connected or Local Modes)
  - Updated on the **5th day of each month**
  - Require **Usage Viewer permission** to access

Shared responsibility model

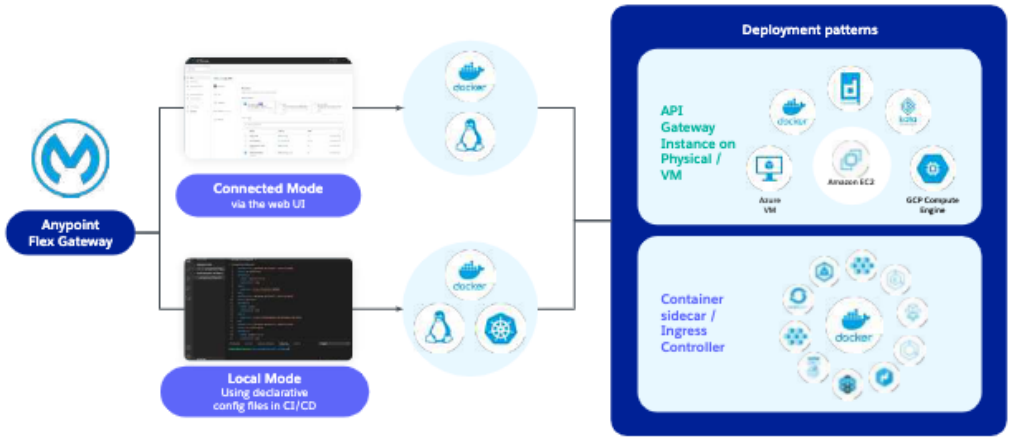
MuleSoft

The successful operation of Anypoint Flex Gateway is a **shared responsibility** model between customers and MuleSoft

MuleSoft Responsibilities	Customer Responsibilities
Provide the Flex Gateway packages Manage, update, and maintain the Anypoint Platform control plane	Use the Anypoint Platform control plane features in a way that complies with company security policies and regulatory requirements Install and configure the Flex Gateway packages Provision, manage, update, and maintain all Flex Gateway runtime infrastructure components Maintain connectivity from the Flex Gateway runtime to the Anypoint Control Plane Do not run third-party software that interferes with normal Flex Gateway operation

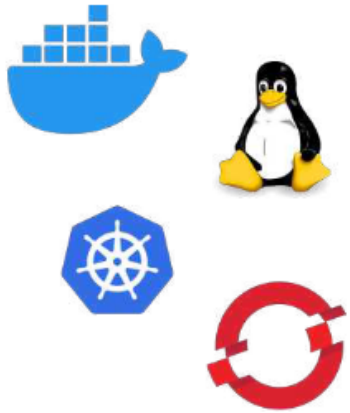
Installing Flex Gateway

Options for running Flex Gateway



Installation options for Flex Gateway include

- In a Docker container for getting started quickly
- As a Linux service running on a virtual machine
- As an ingress controller to secure and balance load for a Kubernetes cluster or an OpenShift cluster



Choosing between Connected and Local Mode

MuleSoft

- Connected Mode
  - Enables management of Mule and non-Mule APIs from a single unified control plane
  - Extending Anypoint Platform capabilities such as API Analytics monitoring, easier reusability through Exchange, and API Manager past typical Mule apps
- Local Mode
  - Using declarative configuration files, independently manage non-Mule APIs
    - Define upstream services
    - Apply policies
    - Configure logging
  - Treating API specification and configuration as code, simplifies API source control and CI/CD automation
  - For more information, refer to the mini course on Local Mode available online

High-level steps for Flex Gateway installation (1/2)

MuleSoft

1. Fulfill the prerequisites (refer to docs)
   - Permission requirements
   - Software requirements based on deployment model
   - Hardware requirements
   - Ports, IPs, and hostname allowlist requirements
2. Choose a deployment model
   - Linux service
   - Docker container
   - Kubernetes Ingress controller
3. Run installation command depending on the deployment model

High-level steps for Flex Gateway installation (2/2)

4. Choose an authentication method
   - Username and password
   - Connected app
   - Token
5. Choose between Local Mode and Connected Mode
6. Run the command to register and run depending on
   - Local or Connected Mode
   - Authentication method
   - Deployment model

See https://docs.mulesoft.com/gateway/1.1/flex-install

Walkthrough 3-1: Install Flex Gateway using Docker

- Create a connected app with the required scopes to install Flex Gateway
- Register Flex Gateway in Connected Mode using Docker
- Start a Flex Gateway replica using Docker

Accessing Flex Gateway logs

Logs management in Flex Gateway

MuleSoft

- Anypoint Flex Gateway pushes metrics to Fluent Bit
- Types of logs
  - Access logs (enabled by default)
  - Runtime logs (message logging policy required)
- Enable access logs by
  - Configuring and applying a Message Logging policy

Logs in Connected Mode vs Local Mode

MuleSoft

	Linux Service	Docker	Kubernetes
Local	Built-in Fluent Bit Declarative configuration via CRD		
	stdout/stderr redirect + log forwarding	Docker logging best practices (e.g., data volume, logging container, sidecar, and log forwarding)	Kubernetes logging best practices (Fluentd / Fluent Bit)
Connected	Same as in Local Mode Anypoint Monitoring and alerts API Manager logs		

Flex Gateway logs stored in API Manager

MuleSoft

- Available for an API instance after applying Message Logging policy
- Basic logs report
  - Inbound and outbound API calls
  - Packet contents
  - Headers
  - Other information in the request
- Access to logs
  - All users in non-production environments
  - Only Admin users in Production environments
- Limitations
  - Stores 100MB per API instance
  - Kept for 30 days
  - Purged automatically by API Manager after 100MB per API instance

Forwarding logs to an external location

- Forwarding logs to an external location can be achieved by
  - Configuring protocol-based outputs
    - File (commonly used for audits)
    - HTTP (e.g., enterprise monitoring solutions)
  - Configuring vendor-specialized Fluent Bit plugins
    - (e.g., Datadog / Splunk / Elasticsearch)
- Message Logging policy must be applied to forward access logs

Walkthrough 3-2: Forward Flex Gateway access and runtime logs to an external location

- Publish an API to Flex Gateway
- Access the runtime logs available within the Docker containers
- Create a configuration file to add a Fluent Bit output
- Examine the Fluent Bit configuration within Flex Gateway
- Apply a Message Logging policy

Scaling Flex Gateway

Flex Gateway replicas

MuleSoft

- A **Flex replica** is an instance of the Flex Gateway
- Only one replica runs by default
- Production environments should run multiple replicas
  - Avoid single point of failure
  - Increased performance if there is higher traffic
- Replicas can be added to a running Flex Gateway
- Replicas have these statuses
  - Connected
  - Disconnected
  - Connected (x of y)

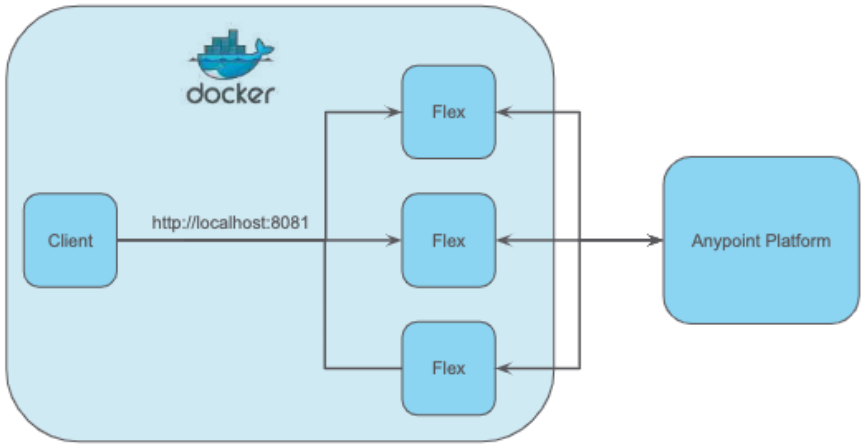
Adding Flex Gateway replicas

MuleSoft

- Running as a Linux service
  1. Make a copy of the registration files
  2. Run the start command used during initial installation
- Running as a Docker container
  1. Select a different port to be used by your replica
  2. Execute the start command used during initial installation

Multiple Flex Gateway replicas running on Docker

MuleSoft



Walkthrough 3-3: Scale a Flex Gateway using replicas

- Add a replica to a Flex Gateway instance
- Apply a Rate Limiting policy to a gateway with multiple replicas
- Check the status of a replica
- Deactivate a gateway replica

Securing Flex Gateway

Gateway policies available for Flex Gateway

Basic Authentication: LDAP	IP Blocklist
Basic Authentication: Simple	JWT Validation
Client ID Enforcement	Message Logging
Cross-Origin Resource Sharing (CORS)	OpenID Connect Access Token Enforcement
Header Injection	Rate Limiting
Header Removal	Rate Limiting: SLA-based
HTTP Caching	Spike Control
IP Allowlist	Transport Layer Security (TLS)

Shared storage configuration in Flex Gateway

- Used by caching and rate limiting policies
- Configured via a custom YAML file
  - Should be of Configuration kind
  - Location for the YAML file is
    - /etc/mulesoft/flex-gateway/conf.d/custom (running as Linux service)
    - /usr/local/share/mulesoft/flex-gateway/conf.d (running in a Docker container)
  - Create a new resource using the YAML file if Flex Gateway is configured as a Kubernetes Ingress controller
- **Redis** should be used in Production workflows

Implementing custom policies in Flex Gateway

- Custom policies in Flex Gateway
  - Implemented using WebAssembly (Wasm) extensions
  - Run on Envoy as custom filters
  - Use the WebAssembly for Proxies (Rust SDK)
- A metadata YAML file contains the policy definition asset and is published to Exchange
- The implementation binary files or WASM files are added to a policy asset in stable state

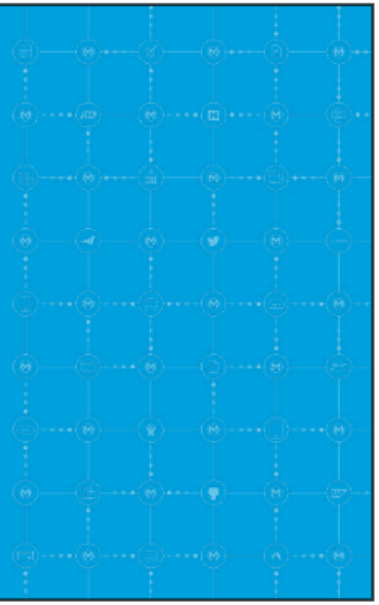
For more information, refer to
https://docs.mulesoft.com/gateway/1.1/policies-custom-flex-implement-rust

https://docs.mulesoft.com/gateway/1.1/policies-custom-flex-getting-started

Walkthrough 3-4: Configure TLS/SSL in Flex Gateway

- Create a certificate and key pair
- Add a TLS policy configuration
- Call an API via HTTPS and check the logs
- Inspect Flex Gateway configuration files

Summary



Summary

MuleSoft

- MuleSoft Anypoint Flex Gateway is a containerized, ultrafast, lightweight API gateway designed to secure and manage APIs running anywhere
- Flex Gateway can be installed in Connected Mode where it is fully connected to the Anypoint Control Plane for management via the API Manager UI, or in Local Mode where it is managed via local configuration files
- In Connected Mode, Flex Gateway logs can be viewed in API Manager or forwarded to third-party software
- Flex Gateway replicas can be configured for high availability and increased performance
- Custom policies can be created using Rust SDK.

MuleSoft

Module 4: Managing APIs using Anypoint Mule Gateway

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

MuleSoft

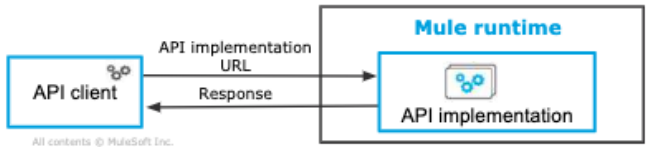
- Describe the two types of API policy enforcement endpoints
- Manage APIs using basic endpoint configurations
- Manage APIs using proxy endpoint configurations
- Deploy API implementations to CloudHub
- Apply automated policies to all API instances

Implementing and managing APIs in Anypoint Platform

Building API implementations as Mule applications

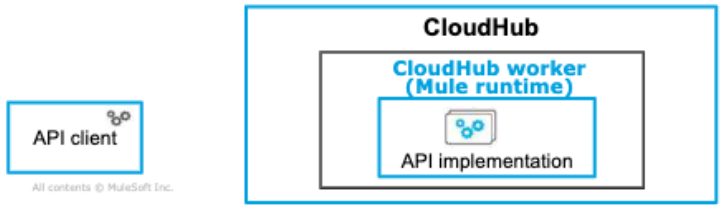
MuleSoft

- Anypoint Platform provides tools to help developers import an API specification from Anypoint Exchange to quickly implement an API as a **Mule application**
- Once coded by developers, a Mule application is packaged as a **deployable archive** JAR file that can be deployed to a **Mule runtime**
- The Mule runtime has API Gateway and API integration capabilities



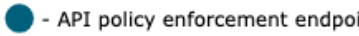
Deploying a Mule application to **CloudHub**

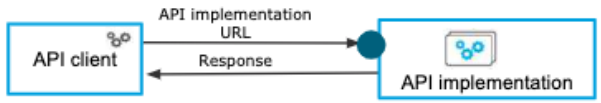
- A MuleSoft-hosted Mule runtime is a cloud-based virtual machine hosted by MuleSoft, also called a **CloudHub worker**
  - Provides various SLAs such as **zero-downtime** redeployments and **scaling** to multiple CloudHub workers, **automatic patches** and **upgrades**, and **centralized logging**
- In this class, you will only deploy API implementations to CloudHub
- More details in the *Anypoint Platform Operations: CloudHub* course



API policies govern access to API implementations

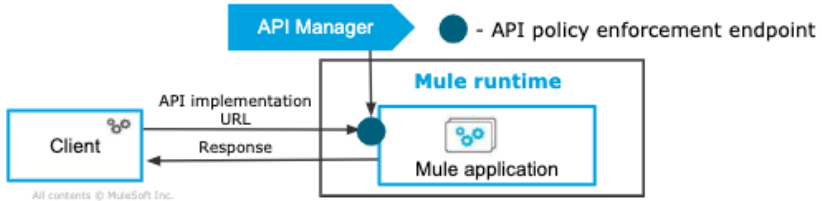
- API policies are non-functional requirement (NFR) rules that govern and control access to API implementation endpoints
  - Types of policies include authorization and authentication, logging, rate limiting, and other qualities of service
  - API policies might be **part of the API implementation** or might be a **separate application**
  - The application that enforces policies is called an **API policy enforcement endpoint**
  - API policies speed up development by offloading NFR code from the API implementation



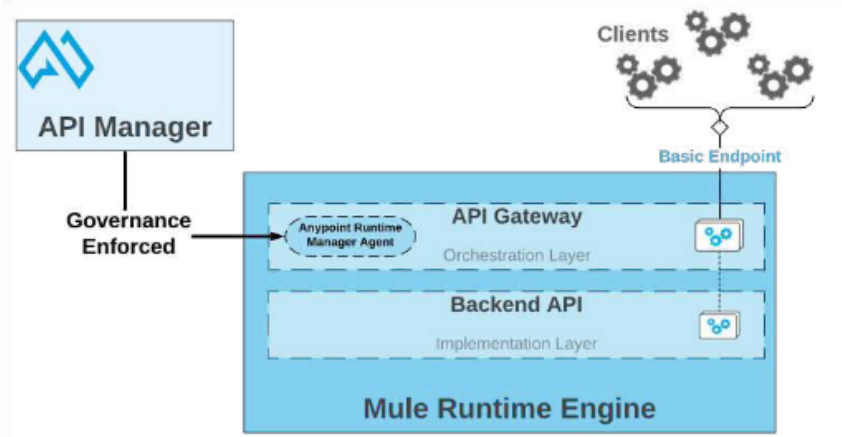


API Manager can define API policies and then inject them into API implementations

- A Mule application can register with API Manager to act as an **API policy enforcement endpoint**
  - Policies are **downloaded** from API Manager and **injected** into each policy enforcement endpoint in order to enforce policies
  - This **decouples** the API **implementation code** and **integration logic** from the **NFR code and logic** of the **policies**
  - API Manager can define, apply, change, and monitor policies for one or more APIs and their API implementations in a central management console



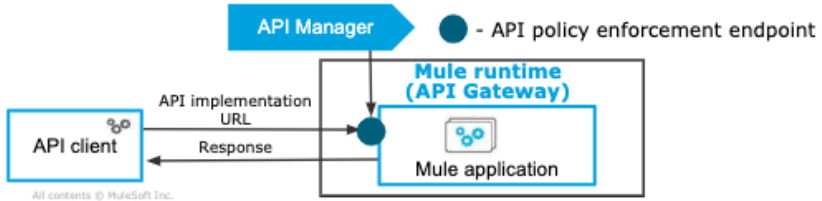
Relationship between API Manager and API Gateway



API Gateway is the part of a Mule runtime that enforces API policies

MuleSoft

- **API Gateway** is an entitlement (libraries) of a Mule runtime
  - Designed and optimized to implement REST APIs from API specifications
  - Supports **governance** of **API implementation endpoints** by applying **API policies** to **API policy enforcement endpoints** that are centrally defined and managed in **API Manager**
  - Available for both customer-hosted and MuleSoft-hosted runtimes



Benefits of using an API Gateway

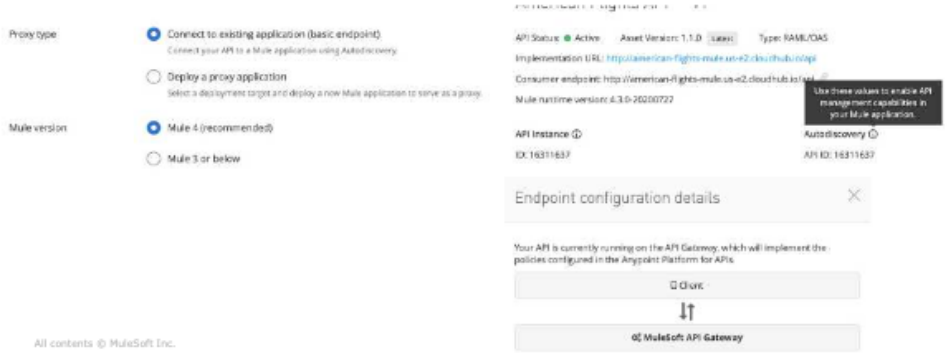
MuleSoft

- **Applies policies** to enforce governance
  - Rate limiting, throttling
  - Caching
  - Security
  - Troubleshooting

- **Separates API management** from API implementation concerns
- **Authorizes API traffic** to pass through the API implementation to backend services
- **Meters API traffic** flowing through the API implementation
- **Logs** all **transactions**
  - Collects and tracks analytics data

Managed APIs define policy enforcement endpoints as **API instances**

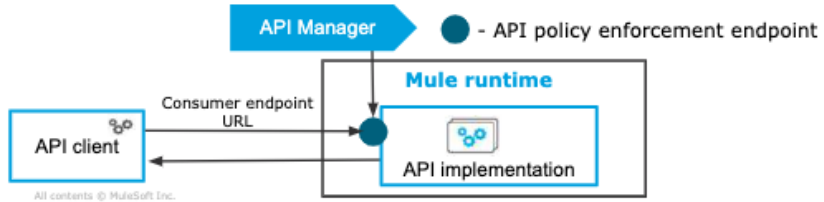
- An API specification is imported into API Manager as an **API instance**
- Each API instance defined in API Manager is specified as either a **basic endpoint** or an **endpoint with proxy**



A **basic endpoint** API instance combines the API policy enforcement with the API implementation

- A **basic endpoint** API instance is configured as a basic API policy enforcement endpoint
  - It is called "basic" because it applies policies in the same Mule application as the API implementation
  - API clients send REST requests on URLs that are part of the actual API implementation endpoints

Basic endpoint

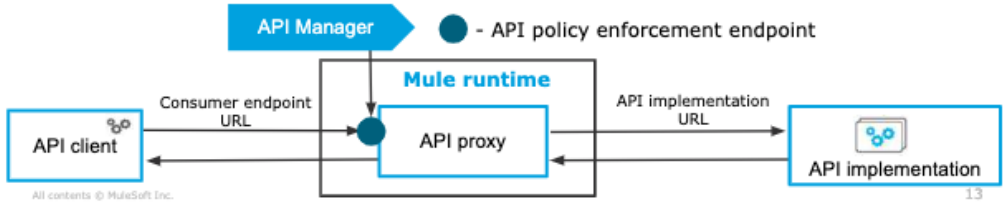


A **proxy** API instance separates the policy enforcement app from the API implementation

MuleSoft

- A **proxy** API instance is an API policy enforcement endpoint that runs "in the middle" in a separate Mule application
  - API clients send requests to URLs in the API proxy
  - The API proxy is then configured to forward validated API client requests to the API implementation, and to forward responses back to the API client
  - API proxies protect and hide the API implementation endpoints from API clients

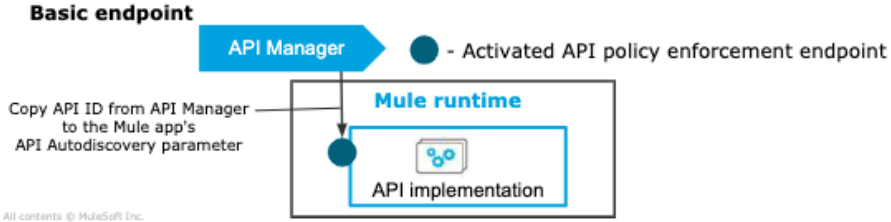
Endpoint with proxy



Configuring and activating a basic API policy enforcement endpoint in a Mule application

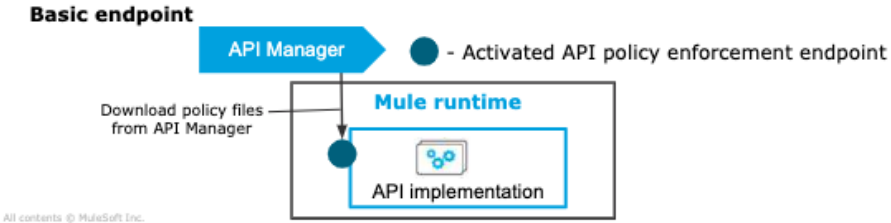
Configuring a basic endpoint API instance in a Mule application

- Because the API basic endpoint is included in the API implementation, developers must expose the **API Autodiscovery** parameter in the API implementation's deployable archive
  - The parameter's value must be configured to deploy the API implementation
  - The value is the API Autodiscovery ID found in API Manager for an API instance
  - Anypoint Platform credentials also need to be configured either in the Mule application or in the Mule runtime



Activating a basic endpoint API instance in a Mule application

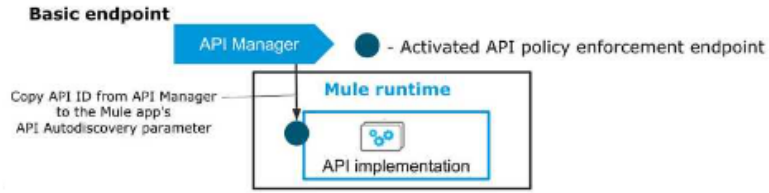
- The basic endpoint API instance API policy enforcement endpoint is activated by deploying the properly configured Mule application to a Mule runtime
  - When the Mule application starts up, API Autodiscovery downloads policies from API Manager and activates the policies in a policy enforcement endpoint
  - Policy changes in API Manager are dynamically pushed to the Mule application and applied at the policy enforcement endpoint



Walkthrough 4-1: Connect an API to a Mule application via Autodiscovery

MuleSoft

- Manage an API by importing it from Anypoint Exchange into API Manager
- Define a basic endpoint API instance in API Manager to apply policies to a Mule application
- Configure a Mule application's API Autodiscovery parameter with the value from API Manager
- Activate an API policy enforcement endpoint by deploying a properly configured Mule application to a Mule runtime



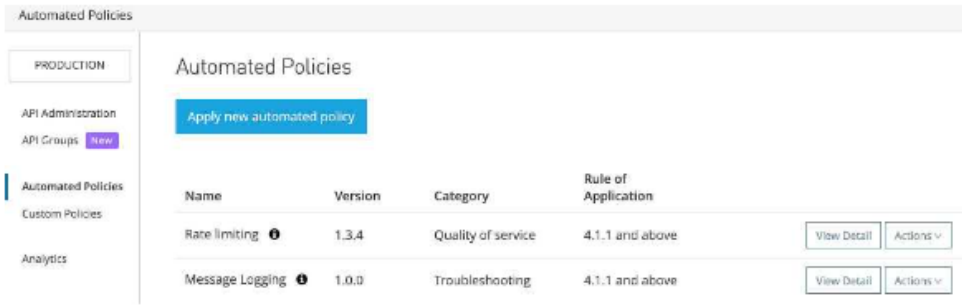
Applying policies to all managed APIs

Common API policies

- A Message Logging policy configures additional customized logging on both inbound and outbound messages to the API policy enforcement endpoint
- JSON and XML threat protection policies
  - Protect against denial-of-service attacks from client requests that try to overwhelm and crash the API endpoint's implementation
  - Can also impose size and depth limits to protect against malicious code being stuffed into string values
  - Length limits include the size of text values, key names, attributes, or comments
  - Other limits include the depth of nested objects or the size of objects and arrays

Applying automated policies to all APIs managed in an API Manager environment

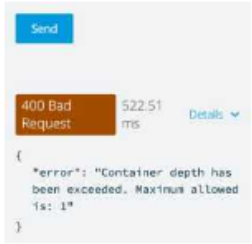
- An **automated policy** is any API policy that is configured to be automatically applied to every managed API instance in an environment



Walkthrough 4-2: Apply policies to a single API instance or to all API instances in an environment

MuleSoft

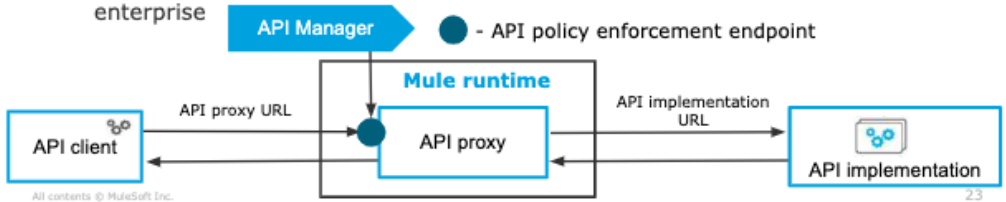
- Define a JSON threat protection policy to be applied to just one particular API instance
- Define an automated Message Logging policy to be applied to every API instance in an API Manager environment
- Test how automated policies are applied to every API instance of any API Manager environment



Deploying a Mule application to serve as an API proxy

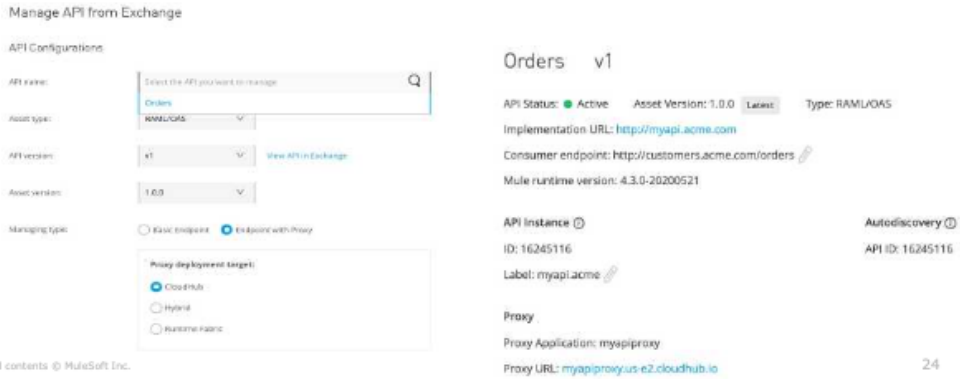
An API proxy does not need to be coded by a Mule developer

- An API endpoint with proxy (API proxy) auto-generates a complete and ready-to-deploy Mule application from a template
  - This separates the policy enforcement code from the API implementation code
  - The auto-generated code has API Autodiscovery configured
  - Includes the API specification when of type RAML/OAS or the WSDL for SOAP services
- The API implementation is a separate application
  - Can run in a separate location and does not have to be a Mule application
  - Might be an existing application owned by a different line of business or enterprise



An endpoint with proxy API instance is immediately deployable from API Manager

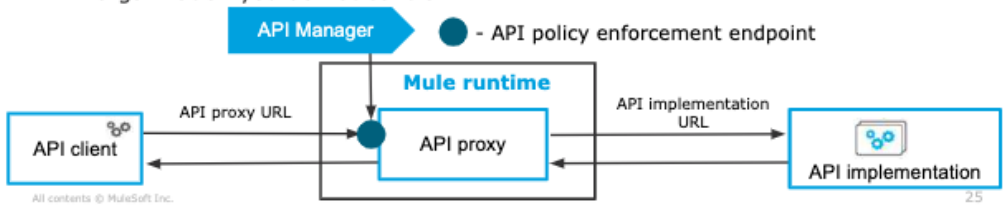
- The API proxy can be created and deployed from API Manager without any coding or help from a Mule developer
  - Can be immediately deployed from API Manager to a Mule runtime



Benefits of proxy API instances

MuleSoft

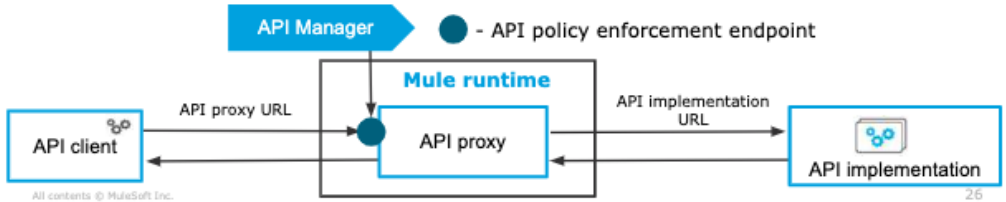
- Minimal or no coding, secure, and maintainable
- Typically used to leverage API Gateway capabilities for
  - Non-Mule applications
  - Mule applications with no API Autodiscovery enabled or other private/closed code
- Allows you to control and govern access on API client requests without having to code the API implementation
  - The API implementation might already be deployed and managed by a different organization you do not control



Issues with proxy API instances

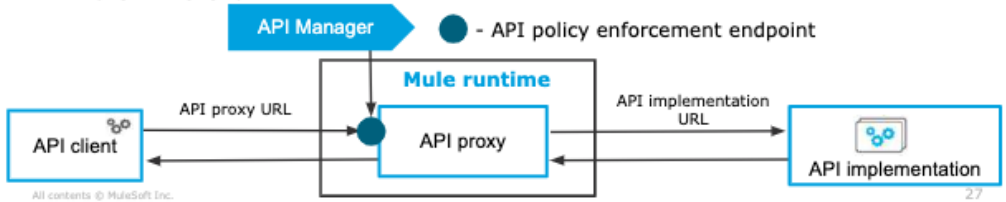
MuleSoft

- The API proxy may need an additional Mule runtime, which counts against licenses
- Access to the API implementation must be secured to avoid API clients bypassing the API proxy
- Adds management overhead, network latency, and more complex troubleshooting



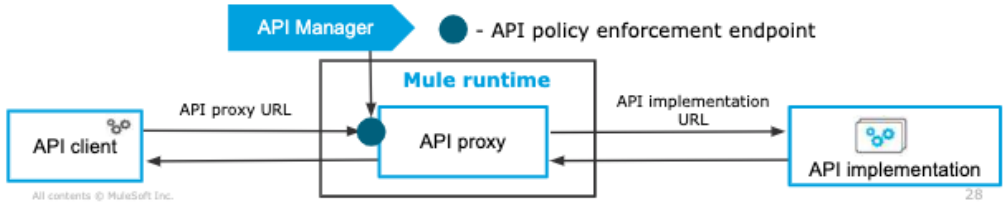
How API client communication is handled by an endpoint with proxy API instance configuration

- The API client sends a request to the API proxy URL
  - Policies are enforced by the API proxy
- For validated requests, the API proxy then sends an HTTP request to the API implementation
  - The API proxy copies any message payload and headers from the request and passes them to the API implementation
  - The API proxy captures message payload and headers from the API implementation response and attaches them to the response message returned to the API client



Walkthrough 4-3: Deploy a Mule application as a proxy

- Manage an API by importing it from Anypoint Exchange into API Manager
- Configure a proxy API instance in API Manager to enforce policies and forward requests to a separate API implementation endpoint
- Test that an API proxy's policy enforcement endpoint properly forwards requests to the API implementation

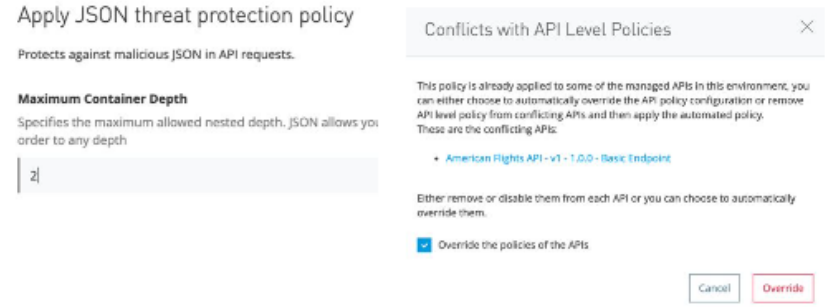


Overriding API instance policies with global automated policies

Automated policies are applied on every API instance

MuleSoft

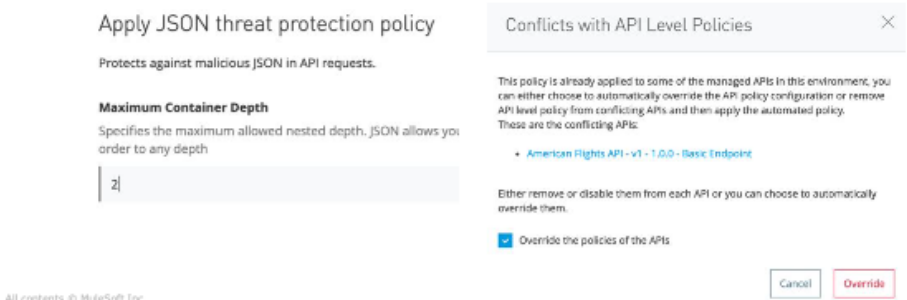
- An automated policy overrides the same policy type activated in any managed API instance in that API Manager environment
- The policy override takes place immediately without having to restart any of the deployed API instance's Mule applications



Walkthrough 4-4: Override an API instance's policy with an automated policy

MuleSoft

- Enable wire logging in an API proxy Mule application
- Create an automated JSON threat protection policy
- Verify the policy applies to both the American Flights API instance and the United Flights API instance

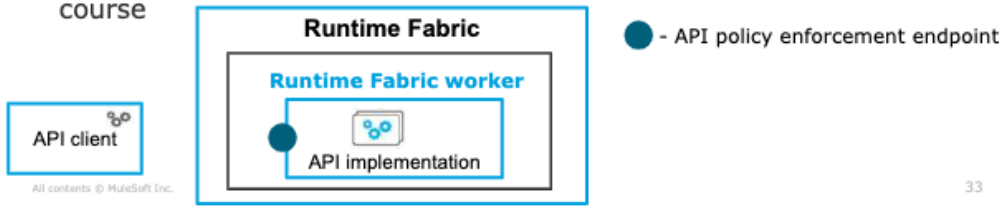


Managing APIs in customer-hosted Mule runtimes

Anypoint Runtime Fabric provides some of the same SLAs as CloudHub for customer-hosted VMs

MuleSoft

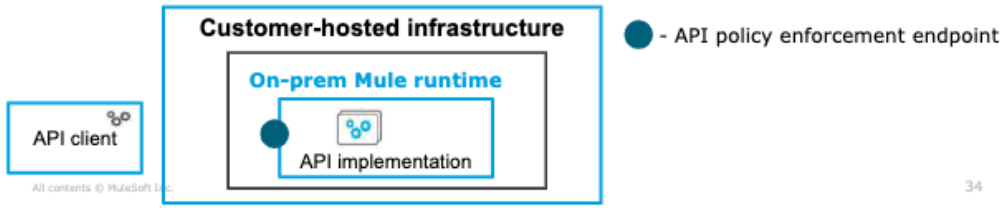
- A **customer-hosted** container service that brings some CloudHub benefits to your on-prem deployments
  - Mule runtimes run in **customer-hosted virtual machines** (also called **workers**) in a data center or private cloud
  - Includes zero-downtime redeployments, isolated applications, horizontal scaling to multiple workers, and easy connection with the MuleSoft-hosted control plane
- The **customer is responsible** for patching and updating the software
- More details in the *Anypoint Platform Operations: Runtime Fabric* course



Deploying a Mule application to a **customer-hosted Mule runtime**

MuleSoft

- A **customer-hosted Mule runtime** or on-prem Mule runtime is a Mule runtime installed in any type of customer-hosted hardware or virtual machine
- The **customer is responsible** for provisioning the hosts and **installing**, **starting**, and **maintaining** the Mule runtime software
- More details in the *Anypoint Platform Operations: Customer-Hosted Runtimes* course

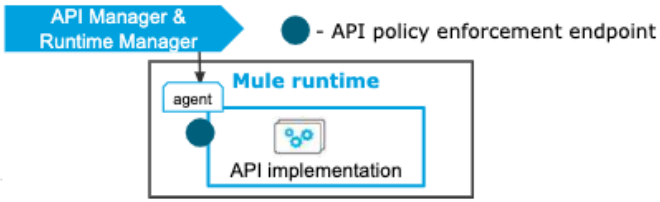


Deploying API instances to a customer-hosted Mule runtime

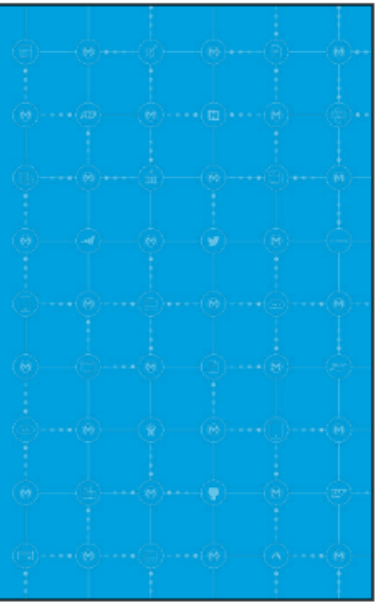
- API instances can be deployed to customer-hosted Mule runtimes
  - To deploy the API instance from API Manager, the Mule runtime must be registered in an Anypoint Runtime Manager environment
  - This is also covered in the *Anypoint Platform Operations: Customer-Hosted Runtimes* course
- Multiple API proxies can be configured to share the same **Mule domain**
  - Allows multiple API proxies to share resources such as a single HTTPS port on one Mule runtime
  - Mule developers can manually create or edit a Mule domain project in Anypoint Studio and combine resources
  - API proxies are Mule applications that can be manually reconfigured in Anypoint Studio to use a different Mule domain

The Anypoint Runtime Manager agent links a Mule runtime with API Manager

- Communication between Anypoint Platform and the API Gateway in the Mule runtime is handled by the **Anypoint Runtime Manager agent** plugin
- For customer-hosted Mule runtimes, the agent installer must be run manually for every Mule runtime installation
  - After installation, the agent helps the Mule runtime connect with Runtime Manager and API Manager to manage the runtime and its Mule applications
  - The agent must be installed for API policy management to function
  - The agent must be patched and updated to match API Manager updates



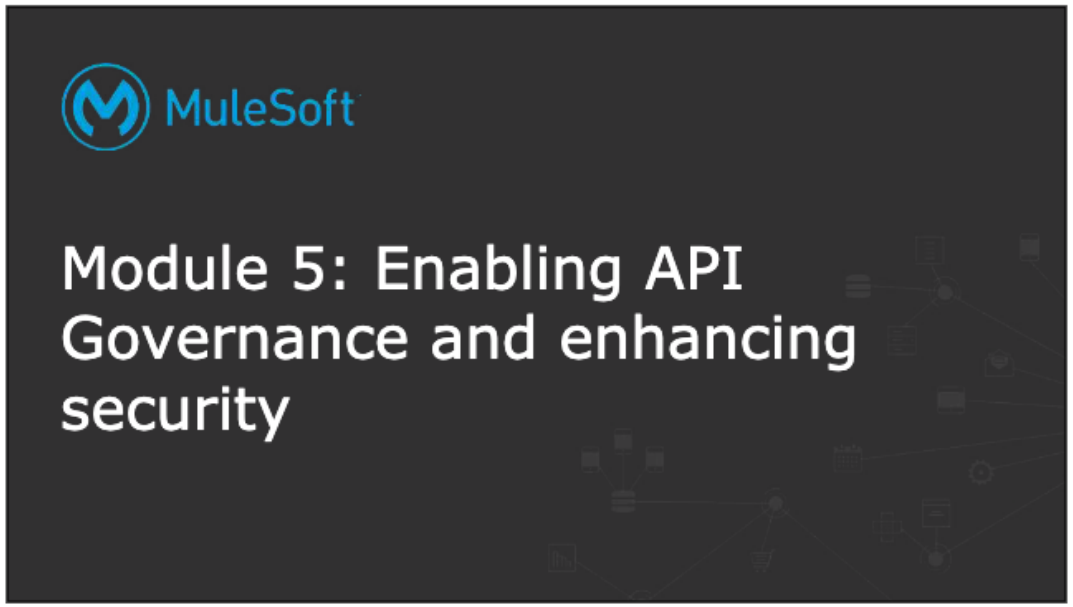
Summary



Summary

MuleSoft

- APIs can be set up to be managed from Anypoint Platform by configuring a basic endpoint or a proxy and the implementation URI
  - Deployment configurations can be a customer-hosted or a MuleSoft-hosted runtime
- A Mule runtime includes the API Gateway libraries
  - An entitlement of a Mule runtime
  - Controls access to APIs by enforcing policies
- An API proxy is an application that controls access to a web service, restricting access and usage by client applications
  - The governance is accomplished through an API Gateway
- Use API Manager to
  - Create and deploy managed API instances



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

- Implement API Governance profiles and rulesets
- Validate API Governance conformance from API Designer and Exchange
- Use Anypoint CLI to automate the API Governance tasks
- Create custom governance rulesets
- Secure managed APIs with the OAuth 2.0 Token Enforcement policy

Introducing API Governance

What is governance?

- Governance involves compliance with **internal and/or external standards** to which an organization is bound
- A **best practice, *when enforced*,** becomes governance
- API Governance ≠ API Management
- **Anypoint API Governance** currently applies governance rulesets at **design time**

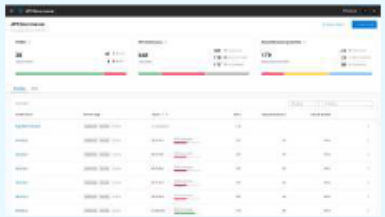
What is Anypoint API Governance?

- Provides frictionless and always-on compliance checks with **non-blocking enforcement at design time**
- Includes **extensible rules** based on popular open standards (W3C, OPA)
- Offers a **central console** to identify compliance and security risks
- Integrates compliance checks with **DevOps automation** and CI/CD pipelines

Empower security *teams* to operationalize governance while developers maintain speed & compliance

Security team / Governance officer

- Gain consistent quality & security
- Ensure conformance to industry & internal regulations
- Observe overall conformance from a single place
- Notify & remediate issues seamlessly
- Automatically govern new APIs



- Self-serve & access rules from Exchange
- Apply standards on any API definition
- Identify & resolve conformance issues
- Automate validations by integrating with CI/CD pipelines
- Follow extensible rules based on open standards (W3C, OPA)

API developer

Applying governance rulesets to APIs

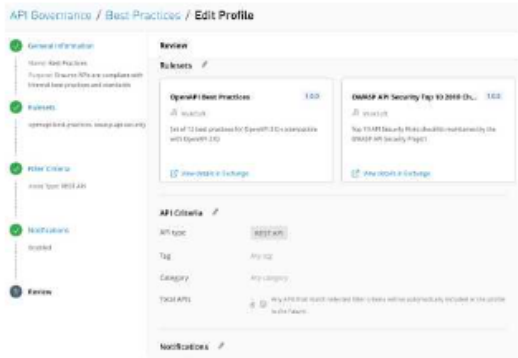
API Governance concepts: governance profiles

MuleSoft

A **governance profile** applies chosen governance rulesets to a select group of APIs

Governance profiles consist of

1. Name and optional description
2. Applicable governance rulesets
3. Filter criteria identifying the APIs to govern
4. Conformance notification rules

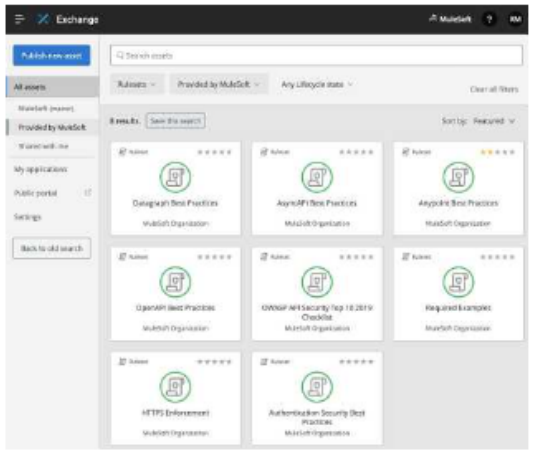


API Governance concepts: governance rulesets

Governance rulesets are collections of rules, or guidelines, that are applied to selected APIs published in Exchange

Provided rulesets in Exchange include

1. Anypoint API Best Practices
2. OpenAPI Best Practices
3. OWASP API Security Top 10
4. Authentication Security Best Practices



Governance ruleset definition and customization

Governance rulesets are defined in YAML files as a collection of rules

- Can be edited to create **custom rulesets**, then published to Exchange
- Each rule name is categorized as a violation (error) or a warning

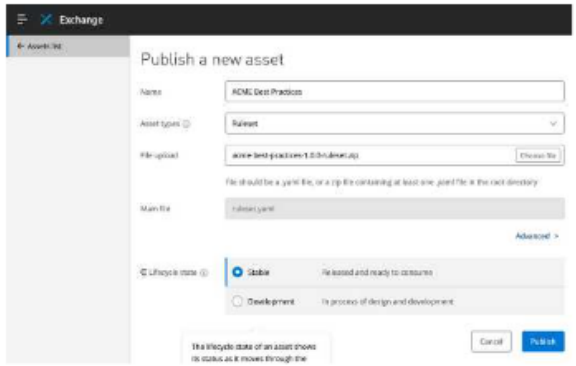
```yaml
profile: ACME Best Practices

description: |
  This ruleset contains over 30 best practices

tags:
  - best-practices

violation:
  #- resource-use-lowercase
  - media-type-headers-response
  - base-url-pattern-server
  - camel-case-fields
  - date-time-representation

warning:
  - api-must-have-description
  - api-must-have-documentation
  - operations-must-have-descriptions
  - responses-must-have-descriptions
  - headers-must-have-descriptions
  - query-params-must-have-descriptions
  - property-shape-ranges-must-have-description
```



Walkthrough 5-1: Create a new API Governance profile and define rulesets to apply to APIs

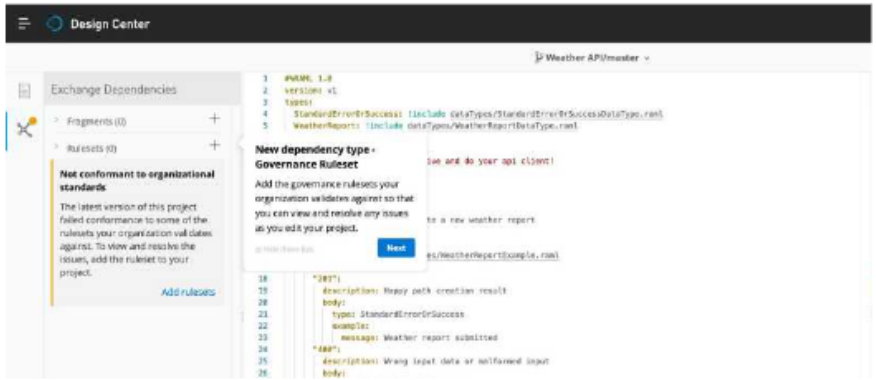
- Set required permissions to use API Governance
- Set up tags and categories for API Governance
- Apply a predefined API Governance ruleset
- Set up alerts and conformance notifications

Identifying and monitoring conformance

Enforcing conformance with API Designer

MuleSoft

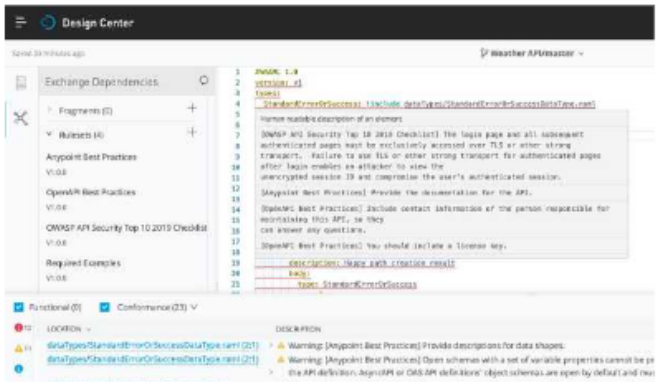
Add governance rulesets in API Designer to enforce conformance at design time



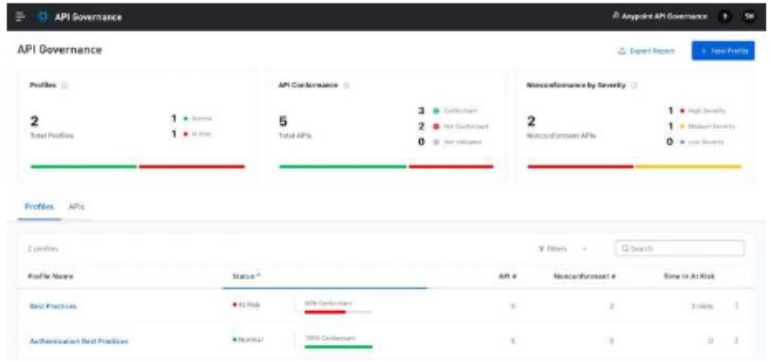
Ruleset violation handling in API Designer

MuleSoft

- Ruleset violations appear in API Designer in popup messages and update dynamically
- Ruleset errors and warnings also appear as Project Errors with type **Conformance**



API Governance dashboard: conformance status



Profile Status

- **Normal:** >= 70% of APIs are conformant
- **At Risk:** <70% of APIs are conformant

Non-conformance Severity

- **High:** 0 - 40% rulesets passed
- **Medium:** 41% - 80% rulesets passed
- **Low:** 81% - 99% rulesets passed

Walkthrough 5-2: Identify and monitor conformance with API Governance

- View the status of API conformance to rulesets
- Create a conformance report
- View conformance status in Exchange
- Identify conformance issues in API Designer

Creating a custom governance ruleset

Custom governance ruleset options

MuleSoft

- Modify an existing ruleset to meet individual needs
- Create new custom rulesets

Creating Custom Governance Rulesets | MuleSoft Documentation

Customizing existing rulesets

- Best approach if an existing ruleset meets most requirements but needs modification

Steps

1. Download the existing ruleset
2. Modify the ruleset YAML file to meet new requirements
3. Validate the changes (*`governance ruleset validate`* command)
4. Generate the ruleset file as an asset document
5. Publish the custom ruleset to Exchange using Exchange UI or CLI commands

Creating new rulesets

- Use if no provided ruleset can be modified to meet needs

Steps

1. Search the MuleSoft Ideas Portal for new rulesets that have already been submitted
2. If not found, submit the ruleset idea to the portal

   https://help.mulesoft.com/s/ideas

Automate API Governance with Anypoint CLI

MuleSoft

Anypoint CLI Command	Description
governance api validate	Validates an API definition against a specified governance ruleset
governance document	Creates the documentation file for a governance ruleset definition
governance profile create	Creates a governance profile
governance profile delete	Deletes a governance profile
governance profile info	Lists information for a specific governance profile ID
governance profile list	Lists all governance profiles for an organization
governance profile update	Updates a governance profile
governance ruleset validate	Validates a governance ruleset definition's format

Promoting APIs between environments

Separating APIs and other Anypoint Platform resources by using environments

- Each business group can define its own environments
  - Environments help manage the entire lifecycle of APIs and other Anypoint Platform resources
- An environment is configured as either production or sandbox quality
  - Production-quality environments offer enhanced SLAs for Mule applications deployed to CloudHub workers
- APIs and Mule applications (API proxies and other API implementations) can be divided between different environments and promoted between environments

Environments

Add environment

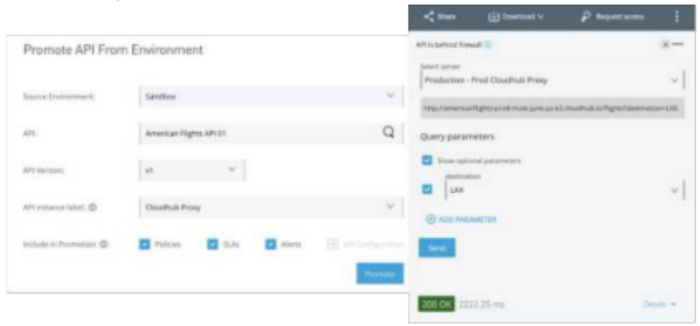
Name	Type
Development	Sandbox
Production	Production
SIT	Sandbox
Staging	Production
UAT	Sandbox

Promoting APIs from one environment to another

- The ability to switch between environments in API Manager helps in promoting a managed API instance from one environment to another
- This functionality allows you to fetch the configurations from the source environment and transfer them to the target environment
  - API creators or version owners can decide to include policies, SLAs, alerts, and API configurations like implementation URLs or deployment targets

Walkthrough 5-3: Promoting a managed API from Sandbox to Production environment

- Create an environment named Production
- Promote a managed API instance from a Sandbox to a Production environment
- Deploy the API implementation in Production



Introducing the OAuth 2.0 Token Enforcement policy

Introducing OAuth2.0

- **OAuth 2.0** is an industry standard protocol for authorization
  - OAuth 2.0-compatible identity providers can centralize **authorization granting** tasks
  - Anypoint Platform supports OpenAM, PingFederate, or Dynamic Client Registration-compliant identity providers
- OAuth 2.0 allows a **service provider** (such as an API instance) to provide access to a **consumer** (such as an API client) on behalf of a **user**
  - Without needing to give the consumer the user's credentials (such as a username and password)
  - The service provider can use OAuth 2.0 to limit access (authorization) in a more restricted way than what the service provider allows to the actual user

Choosing an OAuth 2.0 provider to be used with Anypoint Platform

- Types of supported **third-party OAuth 2.0** providers
  - OpenAM
  - PingFederate
  - OpenID Connect
- The **Mule OAuth 2.0 provider** is an alternative provider developed by MuleSoft
  - Used with OAuth 2.0 Access Token Enforcement Using Mule Oauth Provider policy
  - Uses a custom Mule OAuth 2.0 identity provider
  - **Should only be used for demo purposes**

OAuth 2.0 provides access tokens to applications to safely make API requests over the public internet

MuleSoft

- An **access token**
  - Is a **security credential** that identifies a particular application
  - Authenticates a request from the application to an API resource
- An **authorization grant** type is a particular way for a client application to acquire an access token
  - **Implicit authorization grant**
    - An access token is automatically returned from the authorization request
    - Used for client-side devices that cannot securely store any **client credentials**
  - Other grant types include **authorization code**, **resource owner credentials**, and **client credentials**

OAuth2.0 authorization using **client credentials**

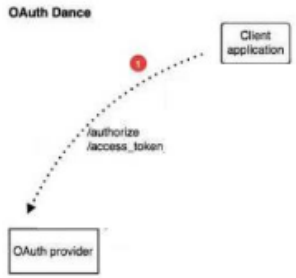
MuleSoft

- A **client credentials** grant type allows client applications to request access tokens outside the context of any **user**
  - This way, the client application does not need to know or share a username or password
  - The access tokens can expire and be automatically updated for better security
  - This is the least secure option, but the easiest to demonstrate using an Anypoint Platform trial account
- A valid **access token** is securely generated from an OAuth 2.0 provider when the client application presents a valid **client ID** and **client secret** to the OAuth 2.0 provider
  - The client ID is a public value, but should be hard to guess
  - The client secret is optional and should only be known by the client application and the OAuth 2.0 provider

How the OAuth dance works in Anypoint Platform: Requesting a valid access token

MuleSoft

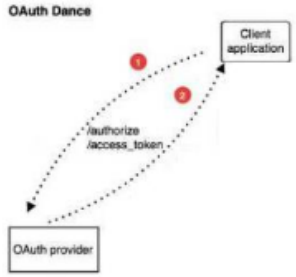
1. The client application requests an access token from the OAuth 2.0 provider
   - The client application presents **client credentials** (a **client ID** and **client secret**) to the OAuth 2.0 provider to authenticate and authorize the request
   - These client credentials allow developers to **authenticate** a client application's identity without using a username and password
   - The client credentials are unique to the client application
   - The client credentials should be more obscure than a username, so harder to guess by public "phishing" attacks



How the OAuth dance works in Anypoint Platform: Receiving a valid access token

MuleSoft

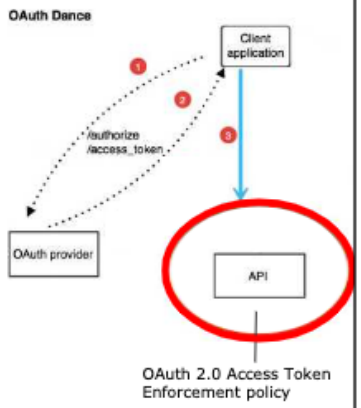
2. The OAuth 2.0 provider securely returns an **access token** to the client application
   - The access token is a safe way to transmit the client ID and client secret over the public internet
   - No one "in the middle" can generate a valid access token



How the OAuth dance works in Anypoint Platform: Making API requests

MuleSoft

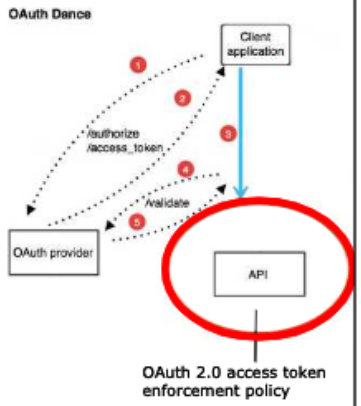
3. The client application then includes the access token in API requests
   - Either as an authentication header or a query parameter in a request to the API
   - When the access token expires, the same client credentials can generate a new valid access token from the OAuth 2.0 provider



How the OAuth dance works in Anypoint Platform: Policy enforcement

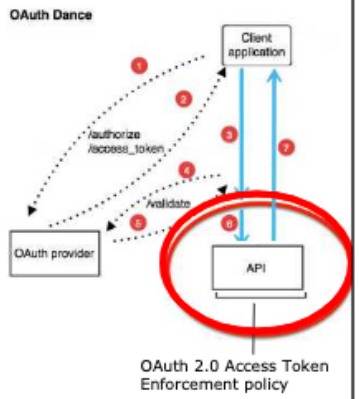
MuleSoft

4. The OAuth 2.0 API policy implementation **intercepts** this request and communicates with the OAuth 2.0 provider to validate the access token
   - The API policy implementation might be in a separate API proxy or in a basic API endpoint that also includes the API implementation
5. The validated token is **allowlisted** and kept on record in the API policy implementation until expiration
   - Any further requests that contain this access token are not validated against the OAuth provider



How the OAuth dance works in Anypoint Platform: Calling authorized API endpoints

6. If the access token is valid, the API request is authorized and then forwarded to the API implementation
   - For a basic API endpoint, this is the same Mule application
   - For an API proxy, the request is sent to another URL
7. The API implementation responds to the client application directly



Securing an API using the OpenID Connect Access Token Enforcement policy

- Add the OAuth2.0 provider instance to the managed API
  - Configure the policy to use the OAuth provider
  - Apply the OpenId Connect Access Token Enforcement policy to the API
- Add the type of security you want implemented for the API to the API's RAML definition
  - Define an OAuth2.0 security scheme
  - Specify the security scheme to be used by a resource(s)

Walkthrough 5-4: Add Okta as an external client provider in Anypoint Platform

- Register for a trial account with Okta
- Set up the authorization server in Okta
- Add Okta as an external client provider in Anypoint Platform

Modifying an API specification to support OpenID Connect access token enforcement

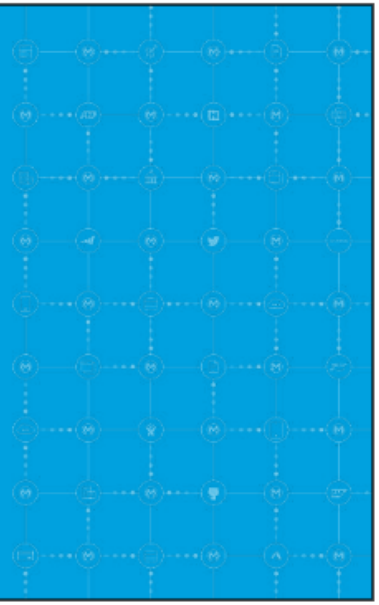
API specification modifications for OAuth 2.0

- API specifications must include a security scheme in order to be able to apply the OAuth 2.0 policy
- Security schemes specify what kind of security is being used by the API, such as basic authentication, client ID/secret, OAuth 2.0, and so on
- By reading the API specification, a user can tell what security will be used to restrict access to the API

Walkthrough 5-5: Apply an OpenId Connect Access Token Enforcement policy

- Update a API RAML definition to specify the OAuth 2.0 security scheme
- Apply the OpenID policy to the API
- Create an API client application to request access to the API implementation
- Test the OpenID policy using Okta as the external provider

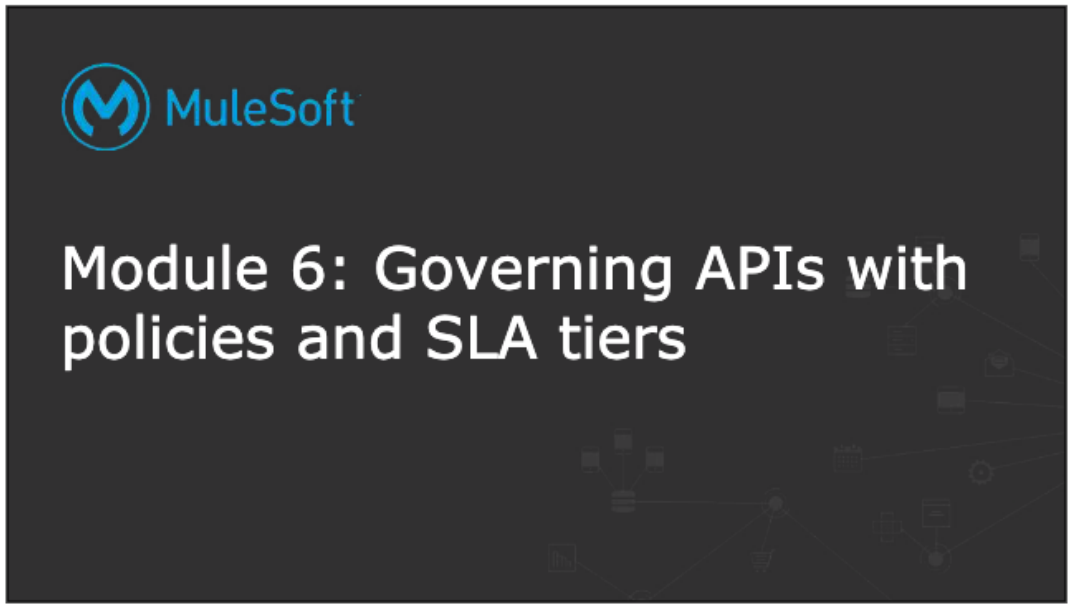
Summary



Summary

MuleSoft

- MuleSoft API Governance enables the application of governance rules during the design stage of API development
- Governance rulesets are grouped into governance profiles, which can then be applied to multiple APIs
- Governance rulesets are cataloged and discovered in Anypoint Exchange
- Rulesets are applied to APIs in Anypoint Design center and conformance is displayed in there as well as in the API Governance dashboard
- The OAuth 2.0 Access Token Enforcement policy can be used to enhance security of APIs



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

- Distinguish between the types of API policies available in Anypoint API Manager
- Enforce service-level agreement (SLA) tiers for APIs or groups of APIs
- Apply the Rate Limiting - SLA-Based policy to APIs
- Apply the Spike Control policy to APIs
- Define the order of execution of policies
- Apply Custom Policies to APIs

Introducing how API policies are managed, defined, and enforced in Anypoint Platform

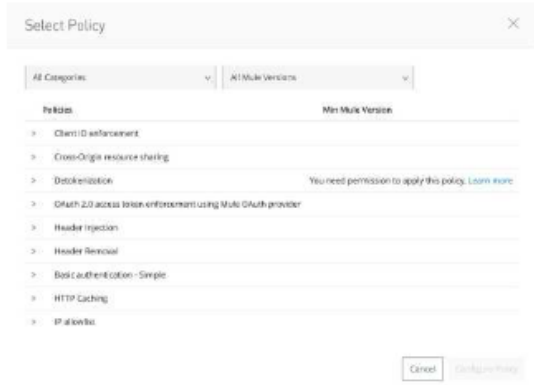
The types of API policies that can be defined in API Manager

MuleSoft

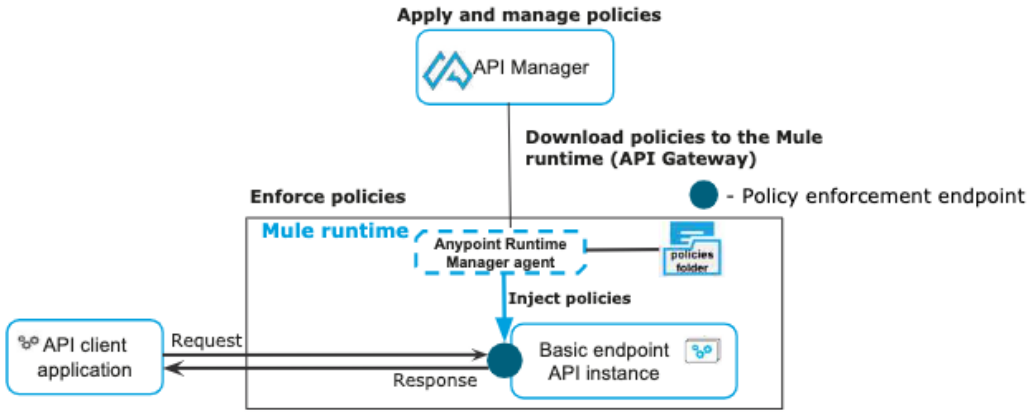
- You've already seen how API policies are configured in API Manager
  - They define various non-functional requirements that can be applied generically to any API, independent of the API's business context, such as data schema, supported HTTP methods, or responses
  - Multiple policies can be applied to APIs, and their execution order matters
- Types of policies include
  - Compliance (Client ID, CORS)
  - Security/authentication (OAuth 2.0, Basic HTTP Authentication, JWT)
  - Quality of service (Rate-Limiting)
  - Troubleshooting, logging, and request and response transformation
  - Custom policies that can be coded and packaged in policy definition JAR files and configured with policy configuration YAML files

Common out-of-the-box policies

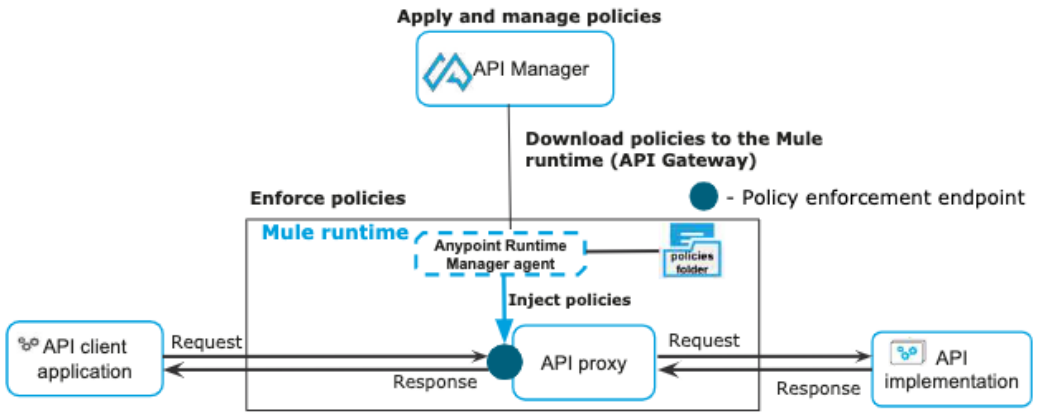
- OpenId Connect Access Token Enforcement policy
- Rate-Limiting SLA-Based policy
- CORS policy
- IP allowlist and blocklist policies
- LDAP security manager policy



Policies are downloaded as policy files and stored in a policy enforcement endpoint's Mule runtime



Policies are downloaded and applied the same way for both types of API instances



Some policies require parameters in the API specification

- For example, client ID enforcement and SLA-based rate limiting
  - Requires client_id and client_secret headers be set by all API client requests
  - If the API specification was developed without taking into account the these requirements, a new version of the API specification must be created and used to replace/update the current API policy enforcement endpoint Mule applications
- Breaking changes in the API specification may require the API instance (the API policy enforcement endpoint Mule application) to be **recoded** and **redeployed**
  - For a **basic endpoint** API instance, a Mule developer may need to add the required parameters or other changes and then export a new Mule deployable archive JAR file
  - For an **endpoint with proxy** API instance, if required, API Manager can automatically generate the new version of the API policy enforcement endpoint Mule application (the API proxy) and redeploy

Bundling related APIs with API groups

What is an **API Group**?

MuleSoft

- A versioned asset type in Exchange that users can use as a filter
- Contains one or more API instances
  - Can span multiple business groups and environments
  - Must share the same Identity Provider (IdP)
- Created in API Manager, published to Exchange for discoverability
- Created by a user with API Group Administrator permission
- Can be assigned an SLA tier
- Supported by API Community Manager

Advantages of using API Groups

- API clients can **request access** to a group of APIs instead of to individual instances
- **Manage SLAs** of related APIs at once from a common place
- **Discover** API Groups as assets in Anypoint Exchange

Deprecating, deleting, and promoting API Groups

- **Deprecate** API Groups
  - Does not deprecate the asset in Anypoint Exchange
  - No new contracts are accepted afterwards
  - To deprecate an API version belonging to a group, first deprecate the group version
- **Delete** API Groups
  - First delete all associated API Group instances, one by one
- **Promote** API Group instances from one environment to another
  - The source environment remains unchanged

Modifying API Groups

- Modify an existing API Group if business requirements change
  - Add or remove new instances
  - If already published to Exchange, a new group instance should be created to modify the version of an API instance
- Restrictions for modifying
  - Group instance has no contracts
  - All instances belong to the same identity provider (IdP)

Walkthrough 6-1: Create, modify, and publish an API Group

- Create a new API Group within an environment
- Add multiple API instances to an existing API Group
- Create an API Group SLA tier
- Apply a default SLA limit to an API Group
- Publish an API Group to Anypoint Exchange
- Request access to an API Group

Limiting requests to APIs per client identity

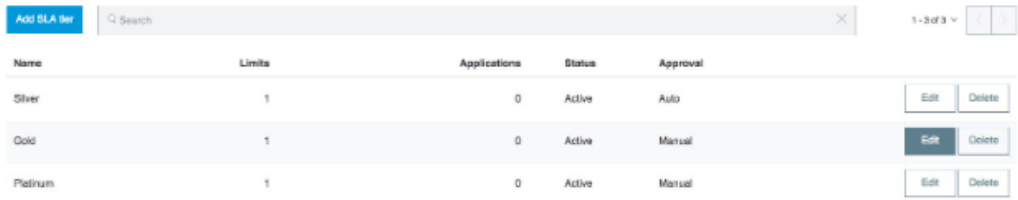
Introducing service level agreement (SLA) tiers

MuleSoft

- A **service level agreement** (**SLA**) tier determines the **quality of service** for a consumer application
  - Defines the number of requests that can be made per time frame to an API
- Layered SLA tiers can be used to impose multiple throughput limits to applications that consume the API
  - The visibility of these throughput limits can be controlled by the API Version Owner or the Organization Administrator
- Applications request access to the API for a specific SLA tier
- SLAs are a packaging tool
  - API owners can group applications using the API into SLAs

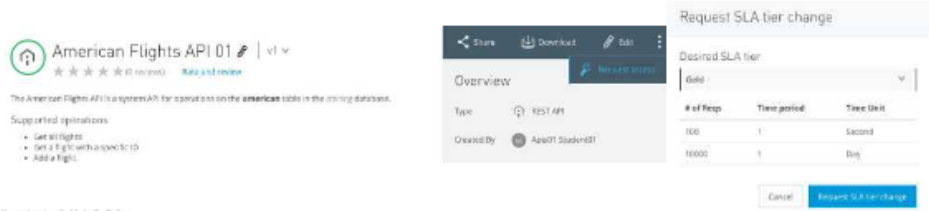
Enforcing SLA tiers

- SLAs could be used by themselves
  - To enable manual approval for client applications
- SLA-based policies require all applications that consume the API to
  - Register for access to a specific SLA tier
  - Pass their **client credentials** in calls to the API



Requesting access to SLA tiers

- When an API is managed by API Manager, anyone (such as an API client developer) can request access to the API from the public Exchange
  - The request includes a client application name
  - If the API uses SLA tiers, the request includes a **desired SLA tier**

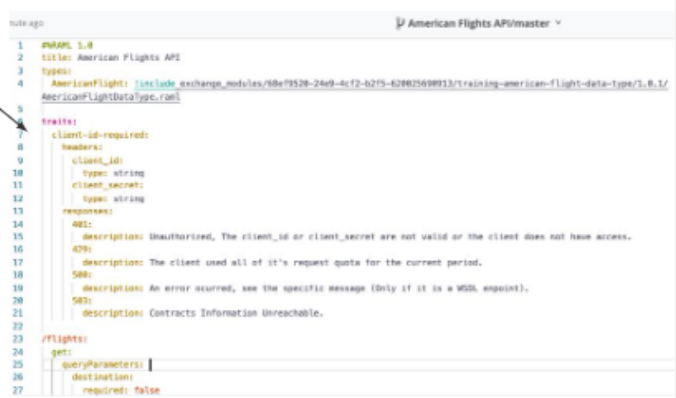


Approving SLA tier access requests from API Manager

- For tiers with manual approval, emails are sent to the Organization Administrators when developers request access for applications
- Organization Administrators can review the applications in API Manager and approve, reject, or revoke requests

Modifying an API specification to support client ID enforcement policies

1. Copy the traits RAML snippet from API Manager to the API specification
   - This is what you will do in the next walkthrough
   - The best practice is instead to use an API fragment containing the required trait(s)
2. Publish a new API version to API Manager
3. Update an API instance to use the new API version so new policies can be applied (like Rate Limiting - SLA Based)



Walkthrough 6-2: Create and enforce SLA tiers through policies

- Apply an SLA-based rate limiting policy to enforce group tiers
- Create API instance SLA tiers for additional levels of throughput
- Add multiple SLA limits to a tier
- Update an API specification and its minor asset version number to support client ID enforcement in API policies
- Apply an SLA tier to an API instance

Applying competing quality of service policies to APIs

Introducing global **rate limiting** policies

MuleSoft

- Rate limiting policies (not SLA-based)
  - Restricts the number of requests the API accepts from **all applications** within a certain **time window**
  - After the threshold is exceeded, all new **API requests** are **rejected** until the configured time window expires
- SLA-based rate limiting policies
  - Use the client ID as a reference to impose quotas **per API client application**
  - After the threshold set by the SLA tier has been exceeded for that **particular client application**, new **API requests** are **rejected** until the configured **time window** expires

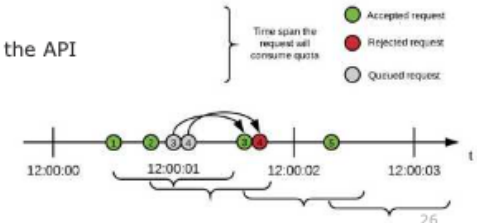
Introducing **spike control** policies

MuleSoft

- Spike control policies help limit the number of requests processed by APIs in a given period of time
  - Unlike a Rate Limiting - SLA Based policy, a Spike Control policy is **not** a per-API client quota enforcement to limit the number of requests an API client can make
  - It **is** a limit set to protect the entire API implementation and corresponding backend services
  - Excessive API requests are **queued** and **retried** after a configured delay time
- For **per API client** quota enforcement, use a **Rate Limiting - SLA Based** policy instead of a **Spike Control** policy

How a Spike Control policy queues and retries API requests

- A Spike Control policy uses a sliding **time window** algorithm
  - New API requests are queued (not rejected) for a certain **time window** after the quota limit is reached
  - The **queuing limit** configures the number of queued APIs allowed
    - New API requests that exceed the queuing limit in the sliding time window are rejected
  - After a configured **delay time** expires, queued API requests are resent
  - Each queued API can be requeued and retried up to a configured **amount of retries**
    - If a queued API is **retried** too many times, the API request is **rejected**



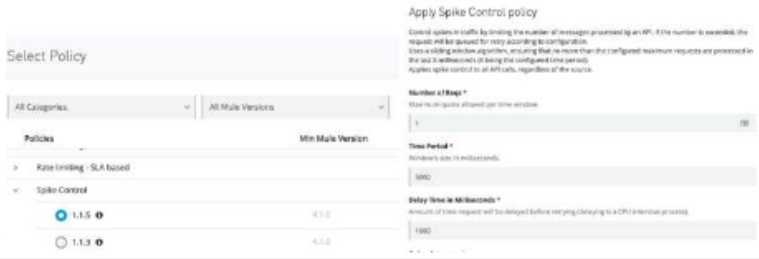
Changing the order of execution of API policies

- Some policies (like CORS) are configured to run before the rest of the applied policies
  - API Manager allows policies to be reordered
  - Policies are applied in the top-down order specified in the API level policies list
  - Automated policies cannot be reordered

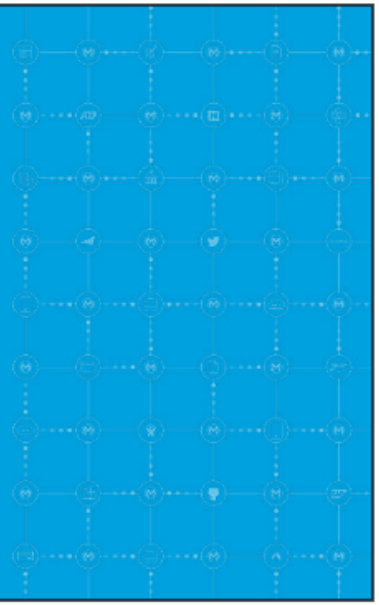
Walkthrough 6-3: Apply a Spike Control policy to limit requests from all API clients

MuleSoft

- Apply a Spike Control policy to an API instance
- Specify the total number of messages to be processed in a time period from all API clients
- Test an API policy enforcement endpoint governed by a Spike Control policy
- Set the order of execution of already applied policies



Summary



Summary

MuleSoft

- Create API Groups to bundle related APIs for easier management
- Manage API access from API Manager by
  - Defining SLA tiers
  - Applying runtime policies
  - Securing API access with OAuth 2.0
- Anypoint Platform has out-of-the box policies for rate limiting, security enforcement, quality of service, and more
- SLA tiers define the number of requests that can be made per time frame to an API
- Apply Custom Policies to extend capabilities of Anypoint Platform

MuleSoft

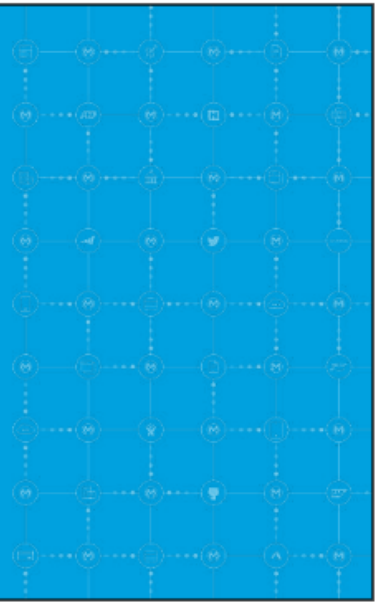
Module 7: Versioning managed APIs

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

MuleSoft

- Version APIs and API groups in Anypoint Design Center
- Document changes in API versions inside portals
- Deprecate old versions of APIs and API groups

Versioning APIs



Determine when and if to version an API

MuleSoft

- Always version when additions or changes result in a **breaking change** to client applications
  - Adding mandatory parameters to existing methods
  - Changing an authentication mechanism
  - Breaking change of data model in API interface
  - Changes to quality of service SLAs (such as quotas or response times)
- Versioning helps to handle future changes even if you do not know what those changes are yet
- **The best practice is to version as little as possible**
  - When possible, add to the existing service in a non-breaking manner
  - Don't version APIs for a basic underlying data model change, like adding fields

MuleSoft semantic versioning best practices

- MuleSoft recommends using version numbers that follow the semantic versioning guidelines
- Reference: http://semver.org
- Version numbers are expressed in a triplet of numbers <major>.<minor>.<patch>
  - Example: version 3.2.1
- **Semantic versioning** quickly advertises and communicates the **meaning** (the semantics) of the changes contained in a new version

Semantic versioning of APIs guides the choice of the next API version number

- Change the **major** version number for **breaking changes**
  - Removing or replacing a method or resource
  - Removing or replacing a required parameter, header, or data model element
  - Documenting changes to SLAs in the API implementations
  - Changing the meaning (the domain) of values returned in API responses, such as adding allowed payment options or adding support for mobile apps
- Change the **minor** version number for **non-breaking changes**
  - Adding a new method or resource
  - Adding a new optional parameter, header, or data model element
- Change the **patch** version number for tiny **non-breaking changes**
  - Changing something in the API specification to handle a bug
  - Updating the API specification's documentation

How to implement versioning

MuleSoft

- Add the version number to the URL
- Add the version to the **accept header**
- Add the version in a **custom request header** of the resource

```
1  #%RAML 1.0
2  title: ACME Banking API
3  version: v1
4  baseUri: http://acme-api.com/{versionNum}
5  baseUriParameters:
6    versionNum: string
```


Adding a version to the URL

MuleSoft

- Update the RAML file to include the version number in the resource name or as a URI parameter
  - Pro
    - API version is in the URL – easy to view and use
  - Con
    - Seems like a representation of different versions of the resource

```
1  #%RAML 1.0
2  title: ACME Banking API
3  version: v1
4  baseUri: http://acme-api.com/{versionNum}
5  baseUriParameters:
6    versionNum: string
```

Adding a version in the accept header

MuleSoft

- Modify the **accept header** to specify the version
  - Pro
    - URIs used by API clients remain stable between API updates
  - Con
    - Users must carefully construct the request with the header and configure appropriately

```
Accept: application/vnd.example.v1+json
Accept: application/vnd.example+json;version=1.0
```


Adding the version in a custom request header

MuleSoft

- Add a **custom request header** with the API version
  - Pros
    - URIs used by API clients remain stable between API updates
    - Does not modify any standard headers
  - Cons
    - When the header is not set with the version number, do you return an error message or route to the new version?
    - How do you handle breaking changes with new API versions every time?
    - They are not a **semantic** way of describing a resource
    - The new custom request headers must be documented and read by API client developers

```
Accept-version: v1
Accept-version: v2
```

Versioning APIs in Anypoint Platform

- Add new version of the API specification in Design Center
  - Always have the master branch be the latest version
  - Create branches with version names v1, v2, v3, etc., for previous versions
  - While publishing to Exchange, the new asset version gets created according to the API version
    - The API version is a string representing the major number of the semantic version number
    - The asset version sets the complete semantic version triplet <major>.<minor>.<patch>

Updating portals for new API version

- When you publish a new API version asset to Anypoint Exchange, the API portal from the previous API version is carried over to the new API version
  - Saves time if documentation across versions overlap
  - Makes the content and structure uniform across all API versions
- Ensure you update the API portal for the most recent API version
- The API reference section in Exchange changes according to the RAML API specification

Walkthrough 7-1: Add a new major API version from API Manager

- Create a new version of a RAML API specification in Design Center
- Configure a proxy endpoint to use the new major API version
- Deploy an API proxy that uses a new major API version

Deprecating older versions of APIs

Before deprecating an old version

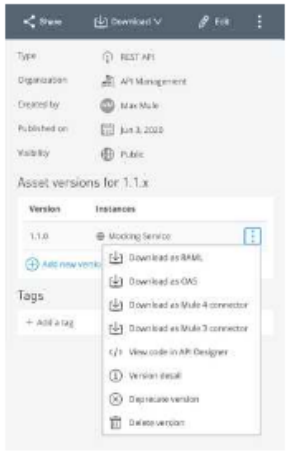
MuleSoft

- Contact developers who own applications that use the API and communicate with them about the new version
  - Ensure service is not interrupted and give time for migration to the new version
  - Ensure developers have time to test and give feedback on it before the new API version goes into production
  - App developers are required to request access to the new version
  - Applications use same client ID and secret for the new API version
  - Offer a realistic migration window so that developers have time to switch to the new version

Deprecating old API versions

MuleSoft

- Deprecation needs to be carried out in Anypoint Exchange
  - Deprecate each asset version that belongs to the old API version
- Set the old API asset version(s) to **Deprecated** to prevent developers from signing up for access to your old API version
- Provide API calls for a finite amount of time until deprecation cut-off date occurs



Walkthrough 7-2: Deprecate an old version of an API

- Create a new group instance to remove an API version
- Deprecate an API group version
- Deprecate an old version of an API in Anypoint Exchange
- Deprecate an API instance in API Manager

Summary

Summary

MuleSoft

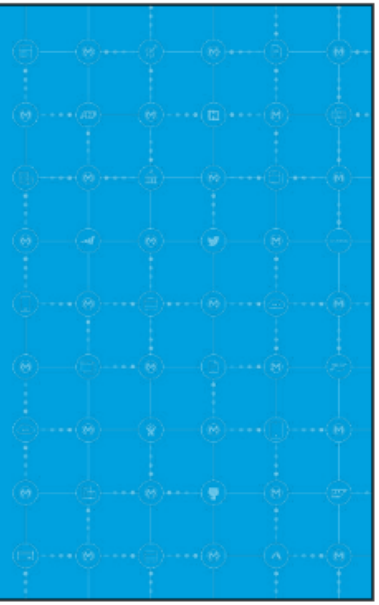
- Managing the lifecycle of an API within Anypoint Platform is a transparent and orderly process
  - It helps to manage new version instances of an API on the API Administration page
- Only update API versions when absolutely required
  - If additions or updates to the API does not break any existing API implementations, you do not need to update the API version
- Anypoint Platform simplifies communicating and engaging with developers by sharing Anypoint Exchange portal content with them

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

MuleSoft

- Enable Anypoint Monitoring for applications and APIs
- Monitor APIs using the built-in dashboards in Anypoint Monitoring
- Create custom dashboards and charts with Anypoint Monitoring
- Create API alert notifications using Anypoint Analytics
- Create and run custom reports from Anypoint Analytics
- Enable API analytics in third-party software

Monitoring APIs with Anypoint Monitoring

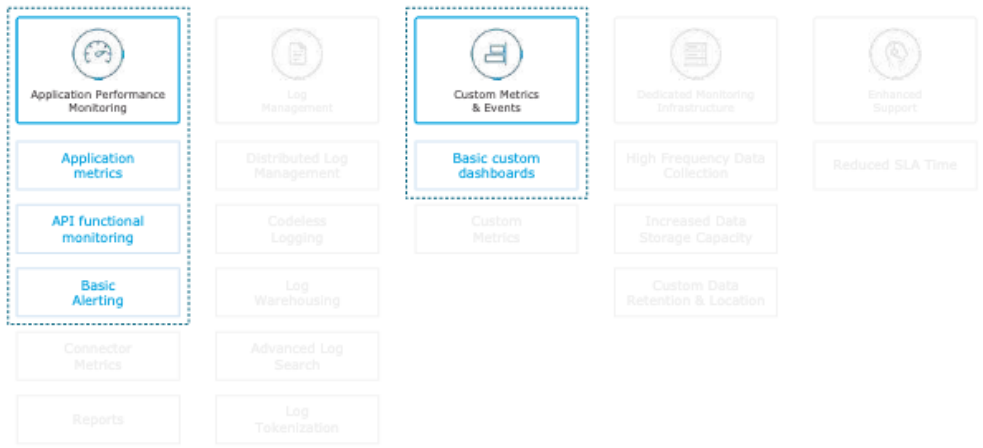


Introducing Anypoint Monitoring

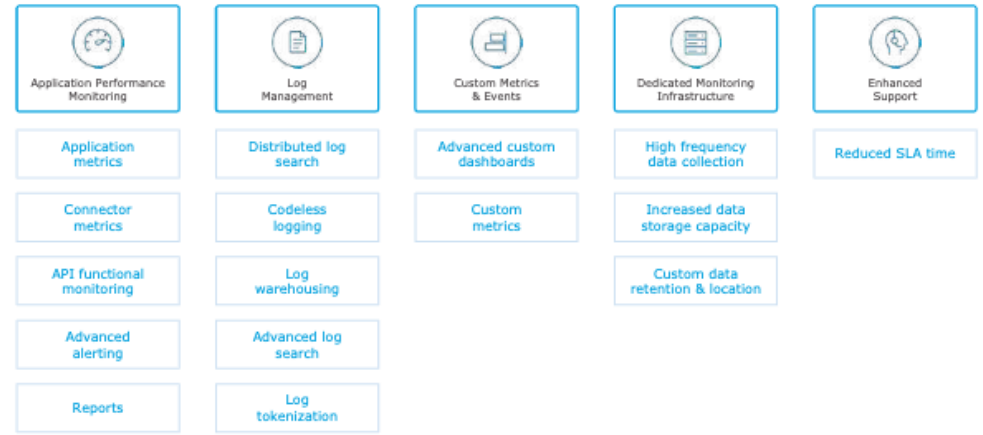
MuleSoft

- Default standard solution for monitoring the application network
- Enables
  - Application performance monitoring
  - Log management
  - Custom metrics and events
- Provides rich features such as
  - Built-in and custom dashboards
  - Alerting
  - API functional monitoring
- Feature access is determined by the subscription level tier
  - Gold/Platinum or Titanium

Features for Gold/Platinum customers



Features for Titanium customers

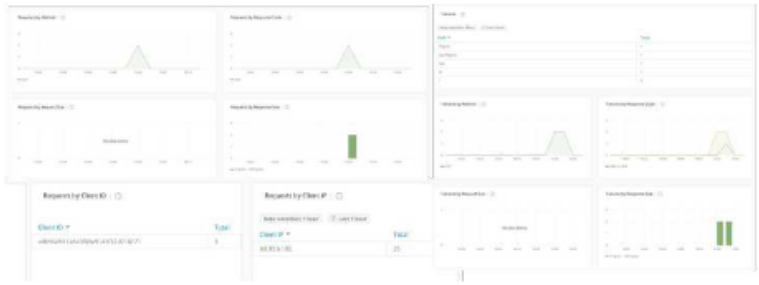


Enabling Anypoint Monitoring for APIs

- Users require **Monitoring Administrator** permission to have full access to Anypoint Monitoring
- Users require **Monitoring Viewer** permission for read-only access
- Applications need a property set to enable monitoring
  - *anypoint.platform.config.analytics.agent.enabled=true*
  - Can be set at deployment time or through CI/CD pipeline
  - Can be set via UI in the settings menu of Anypoint Monitoring; requires restart
- Can be enabled in customer-hosted runtimes; see docs
- Enabled by default in Runtime Fabric applications

Walkthrough 8-1: Explore Anypoint Monitoring built-in dashboards

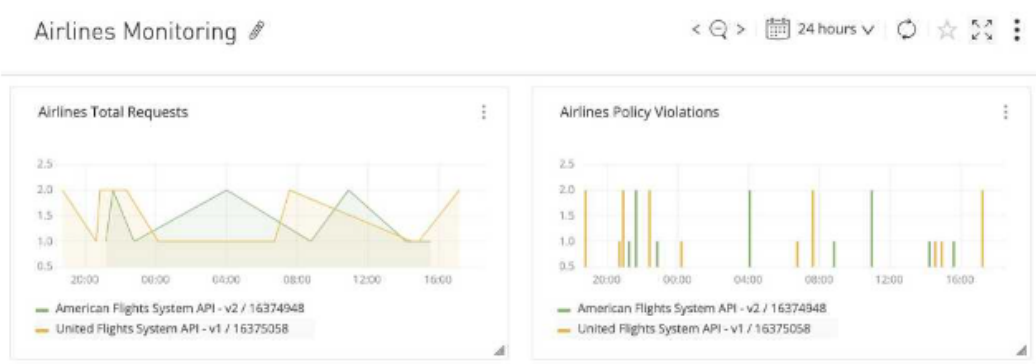
- Set required permissions to use Anypoint Monitoring
- Enable Anypoint Monitoring for applications on CloudHub
- Navigate to the Overview dashboard in Anypoint Monitoring
- Discover other built-in dashboards available with advanced metrics



Creating custom dashboards

- Customize how your data is visualized
- Combine applications and API metrics in a single view
- Add multiple charts (**widgets**)
  - Graph
  - Singlestat
  - Table
  - Text (plain, Markdown, HTML)
- Trigger alerts based on a graph in a custom chart (Titanium)
- Download the data behind a dashboard to a CSV file (Titanium)

Walkthrough 8-2: Create a custom dashboard in Anypoint Monitoring

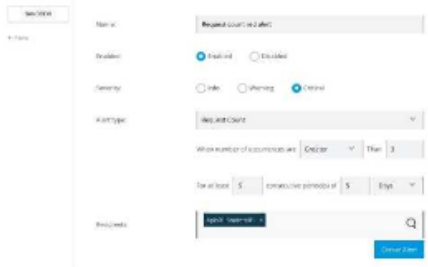


Creating API operational alerts with API Manager

Introducing API alerts in API Manager

MuleSoft

- An API alert is an alarm that signals Anypoint Platform users or other email addresses
  - Different from Runtime Manager alerts that are used to flag CPU usage, deployment success/failure
  - Flags certain type of behavior related to an API request or response



Types of alerts that can be created

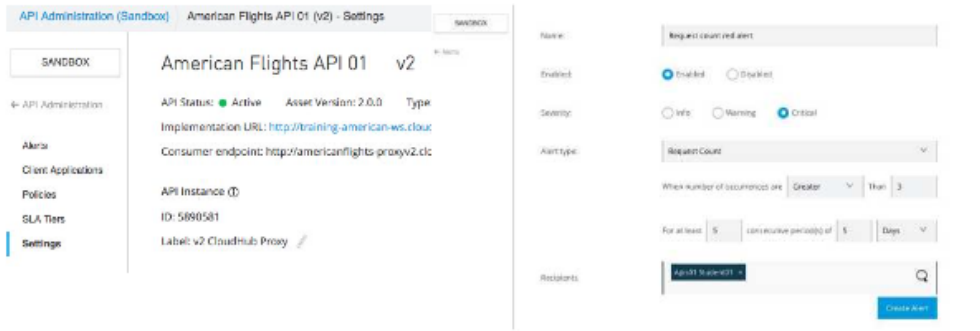
- Policy violation
  - Triggers an alert when one or more policies that govern the API are violated
- Request Count
  - Triggers an alert when users request access to the API more times than allowed in a certain period of time
- Request Code
  - Triggers an alert when the API returns one of these codes when a request is sent - 400,401, 403, 404, 408, 500, 502 or 503
- Response Time
  - Triggers an alert when the response time of the API exceeds a specified period

Receiving API alert notification

- When an API alert is set in API Manager
  - Any email address can also be added to be notified by email when an alert is triggered
  - Organization users can also be added as alert recipients
- Selected recipients receive two email notifications
  - One email describes the alert
  - The second email notifies the user when the alert is resolved

Walkthrough 8-3: Create an API operational alert with API Manager

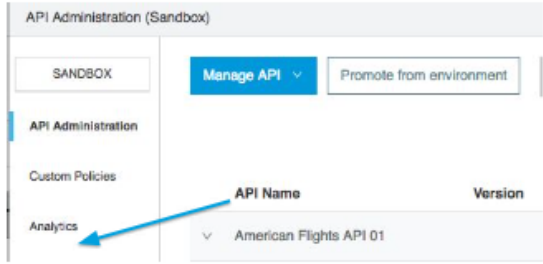
- Explore the metrics available in the API list view
- Create an API operational alert
- Configure and trigger an API operational alert



Introducing Anypoint Analytics

The Anypoint Analytics Dashboard

- Gives insight into API usage and performance
- Helps to view dashboards (both overview and custom), create and manage charts and reports
- Can be accessed by users who have Admin access to the API environments in API Manager



The Overview Dashboard

- Default dashboard in API analytics
- Contains a set of charts
  - Requests by date
    - Line chart representing number of requests
  - Requests by location
    - Map chart showing the number of requests for each country of origin
  - Requests by application
    - Bar chart showing the number of requests from each of the top five registered applications
  - Requests by platform
    - Ring chart showing the number of requests broken down by platform

The Anypoint Analytics Event API

MuleSoft

- Gives access to raw data so that it can be analyzed or transformed
- Contains an API endpoint to fetch analytics data
  - Data can be used in third-party tooling, or stored in a database
- The Manage Reports tab in the Anypoint Analytics Dashboard provides an interface to query the Analytics Event API endpoints
- The API portal for the Analytics Event API can be found here

Retrieving reports via API

MuleSoft

- The Analytics Event API endpoints can be accessed using cURL commands, any programming language, or REST client
  - Obtain an access token using Anypoint Platform login credentials
  - Requires an additional **API Platform token**
  - Copy the report URL from the Manage Reports tab in the API Analytics Dashboard
  - Make a request to the endpoint URL by appending the access token with the call
  - If the request is successful, the report information gets saved in a CSV or JSON file depending on the input parameters

Walkthrough 8-4: Create a custom report in the Analytics Dashboard

MuleSoft

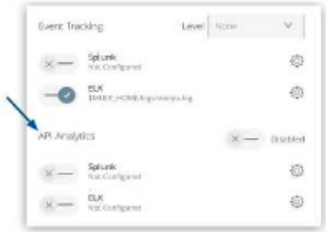
- Create a custom CSV report in the Analytics Dashboard
- Run a custom report in the Analytics Dashboard

Forwarding Anypoint API analytics events to third-party software

Enabling API analytics in third-party software

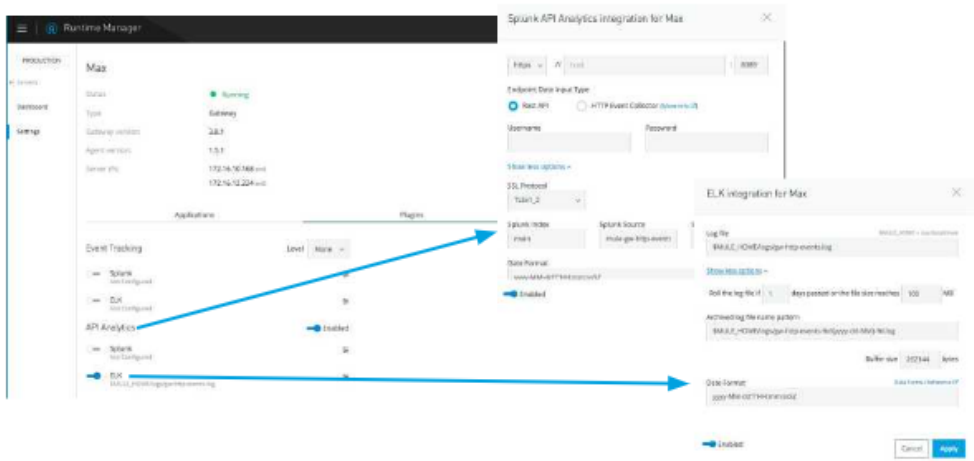
MuleSoft

- Not supported as a Runtime Manager feature for CloudHub applications
- Anypoint Platform supports monitoring, analyzing, and visualizing data for APIs in third-party software like Splunk and ELK
- To enable the API analytics in Splunk or ELK, configure the plugin in the server that runs the API proxy in Runtime Manager

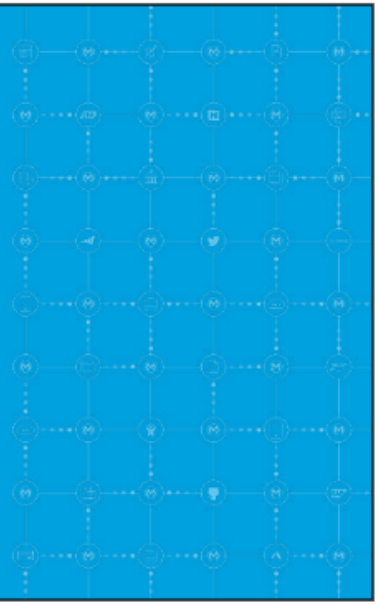


Configuring third-party API analytics integration

MuleSoft



Summary



Summary

MuleSoft

- Enable Anypoint Monitoring to get insights into your application network usage and performance
- Only users with Monitoring Administrator or Monitoring Viewer permission can access Anypoint Monitoring
- API alerts triggers email notifications related to API request/response behavior
- The Analytics Event API helps in creating a report that queries the desired data and exposes it via an API endpoint
- The Manage Reports tab in Anypoint Analytics Dashboard provides an interface to query the Event API endpoints

In this module, you will

MuleSoft

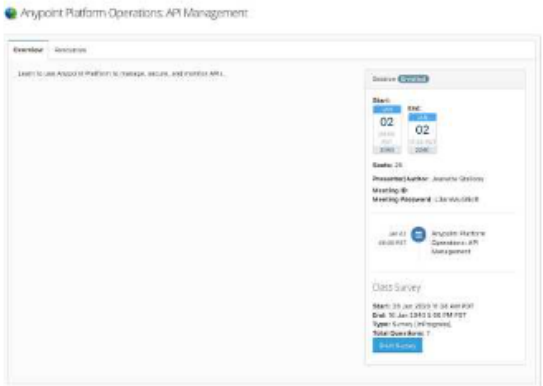
- Take the course survey
- Learn where to go from here

Take the course survey

Course survey

MuleSoft

- Please take the course survey via the link provided by your instructor
- **Please complete the survey now!**
  - We want your feedback!



Where to go from here

Take additional MuleSoft training courses

- Anypoint Platform:
  - Flow Design
  - API Design
- Anypoint Platform Development:
  - Fundamentals
  - Production-Ready Development Practices
  - Production-Ready Integrations
  - DataWeave
- Anypoint Platform Operations:
  - CloudHub and CloudHub 2.0
  - Customer-Hosted Runtimes
  - Runtime Fabric on Self-Managed Kubernetes
  - Runtime Fabric on Virtual Machines
- Anypoint Platform Architecture:
  - Application Networks
  - Integration Solutions

training.mulesoft.com

Learn and build on your own

MuleSoft

- Read more
  - Documentation: developer.mulesoft.com/docs
  - Support knowledge base: help.mulesoft.com/s/support
  - Blog: blogs.mulesoft.com
- See more code
  - Templates: www.mulesoft.com/exchange
- Report and track issues
  - Mule kernel Jira: www.mulesoft.org/jira/browse/MULE
  - Mule: Submit through the Support Portal

Interact with the community

MuleSoft

- Ask questions
  - Forums
    - General forum: help.mulesoft.com/s/forum
    - Course-specific forum: training.mulesoft.com/help/apops-apis
  - Stack Overflow: stackoverflow.com/search?q=mule
- The MuleSoft Help Center: help.mulesoft.com
  - Forum – ask questions
  - Discussion groups – browse topics
  - Resources – get technical resources
  - Training – expand your expertise
  - Support – support and subscriptions
  - Community – network with your peers

Keep current

- Conferences and events: www.mulesoft.com/events
- Blog: blogs.mulesoft.com
- Webinars: www.mulesoft.com/webinars
- Roadmap webinar
  - Get invited by selecting Roadmap in email preferences resources.mulesoft.com/preference-customers.html
- Facebook: www.facebook.com/mulesoft
- Twitter: twitter.com/mulesoft

Influence MuleSoft's roadmap

- Attend quarterly customer roadmap webinars
- Fill out customer surveys
- Leave feedback (ratings and comments) on documentation pages
- Open customer support that can result in enhancement requests

Thank you