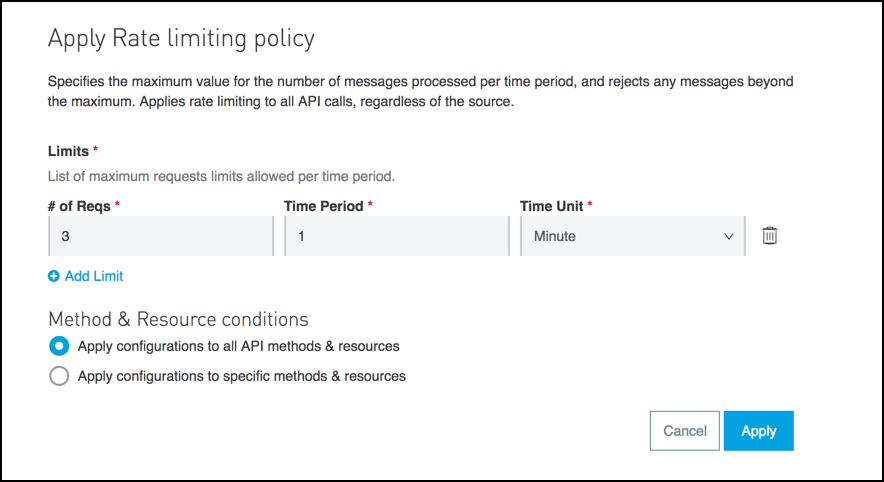
# API Policy Management

An API policy provides a powerful ability on the API Gateway to change the behavior of a given API through configuration. A policy, in other words, is one or more configuration settings that can be directly applied to an existing API deployed and running within the API Gateway. Such policy configuration settings have a wide range of uses, from applying specific security policy to logging incoming request and outgoing response.

In majority of the use cases where API policies are applied, a developer or API administrator needs to provide shared and reusable configuration across more than one API. If such configuration is required for only a single narrow use case, the API developer can simply program that logic inside that API code itself. But for shared configurations across more than one API, applying a default or custom policy is the recommended approach.

An example of a shared reusable configuration that can be applied to more than one API is the rate limiting policy, to monitor and limit the number of API requests. For example, a rate limiting policy can be used within Kong, 3Scales, Mulesoft, or AWS API Gateways. Most of the rate limiting policies across these and other API gateways have similar configuration settings:



## Example above from Mulesoft API Gateway shows a typical policy configuration form, where an API administrator or developer provides policy settings. For a rate limiting policy, the basic inputs should be the number of requests within a specific time period, which instructs that policy when to trigger an exception if the maximum number of requests has been reached.

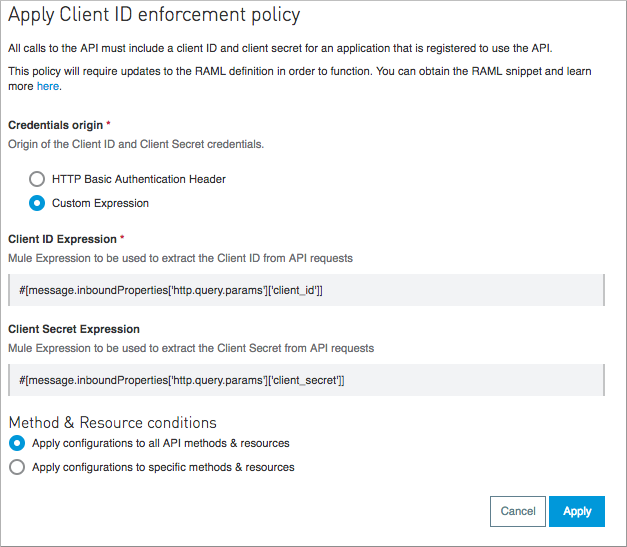
## API Policy Types and Uses

There are many different types of APIs, with most of them grouped in to the following categories:

* Compliance
* Security
* Quality of Service

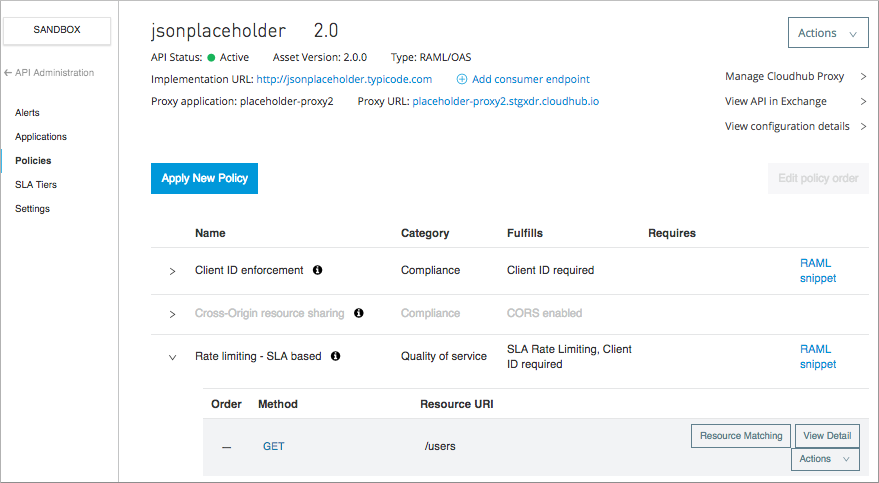
An API Gateway such as Mulesoft, AWS, or Kong, will be bundled with a number of provided policies which have been developed and integrated with the API Gateway product by the product team. Most of the API Gateways also come with a marketplace of third-party and community-developed policies, which can be integrated by adding them to the APIs from the marketplace or Exchange.

It is important to review the number and quality of the provided policies when choosing an API Gateway platform. Certain policies like security policies to validate API security are imperative to successful API management, therefore these security policies should be expected to come provided with the API Gateway product. Lack of provided policies indicates immaturity of the API Gateway platform under review. Certain policies are also valuable as they integrate with specific third parties. For example, Ping or Okta Identity Management integration policies are useful if there is a need to integrate with Ping Identity or Okra security solutions. Mulesoft, for example, has partnership agreements with both Ping Identity and Okta, and therefore Mulesoft API Gateway comes bundled with the provided Ping and Okta OAuth and JWT token validation policies. Developing such security policies from scratch is not a trivial exercise on any of these API Gateway platforms.



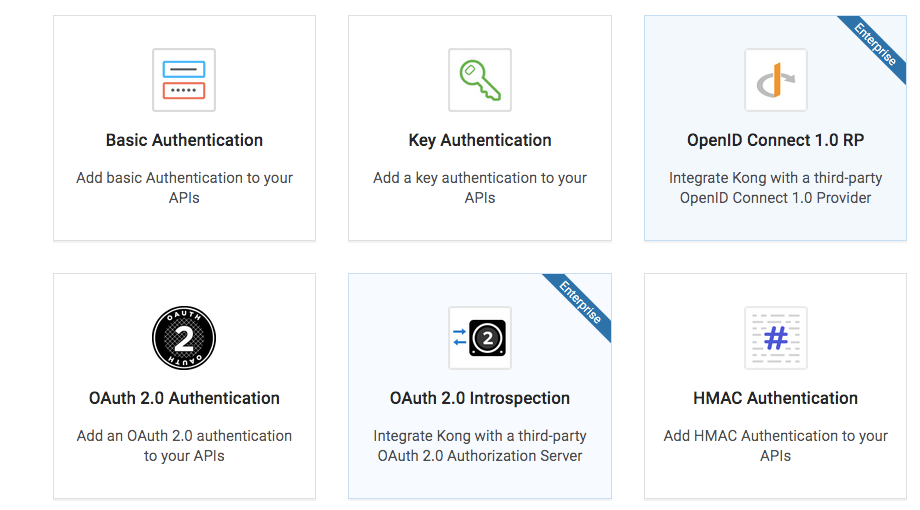
### Compliance Policies

Compliance policies are used to enforce specific protocol or specification compliance on a given API. For example, Cross-Origin Resource Sharing (CORS) compliance or Client ID enforcement policies are provided in a number of API Gateways:

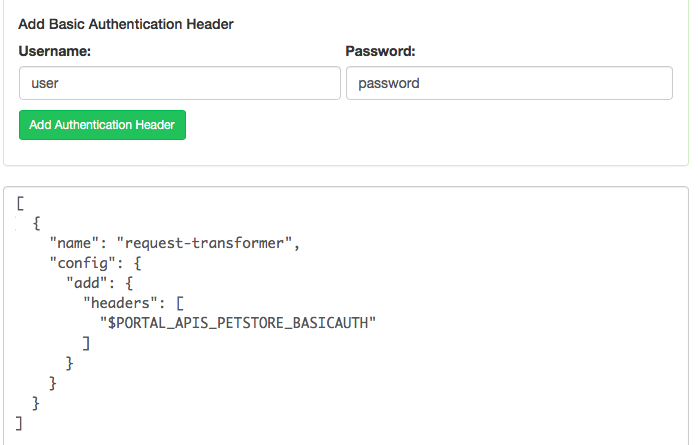


### Security Policies

Security policies are developed to protect APIs with specific security protocols such as Basic Auth, OAuth, JWT, Open ID, SAML, or Certificates. An example of a security policy commonly used is the Basic Auth policy. A provided policy to validate Basic Auth credentials exists in almost all API gateways, including Kong, AWS, Azure, Mulesoft, 3Scales and others. A Kong Basic Auth plugin looks like this:

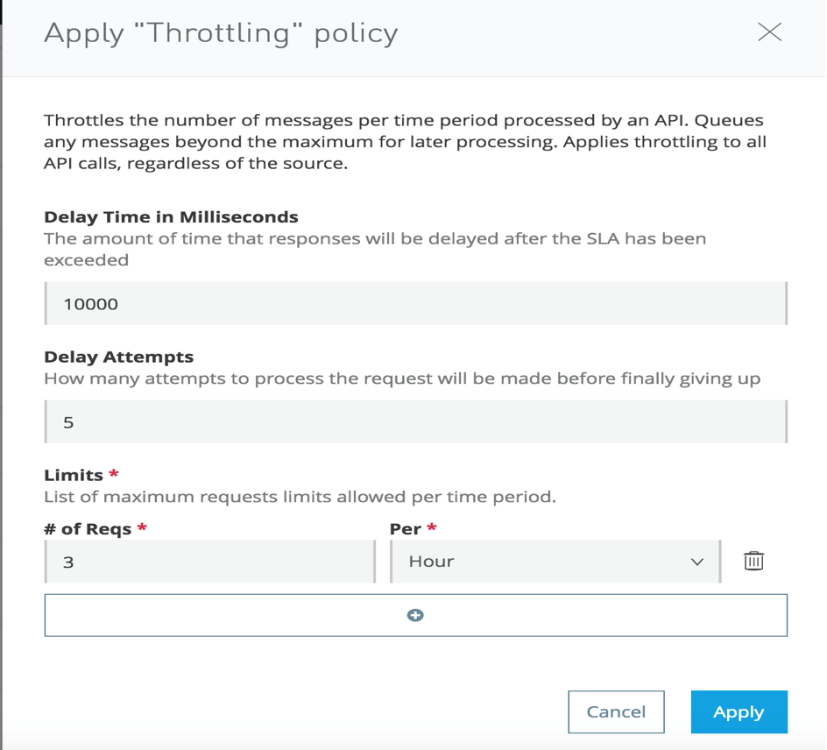


Once Basic Authentication plugin is selected, it needs to be configured to check for a specific username and password passed in the Base64-encoded Authorization header of any API request. A Basic Auth security policy will intercept each request and validate passed-in username and password in the Authorization header by Base64-decoding the header value and comparing the username and password values with the ones in the plugin/policy. In Kong, for example, an administrator of the security policy would enter username and password that have to match those passed from any API invoker:



### Quality of Service Policies

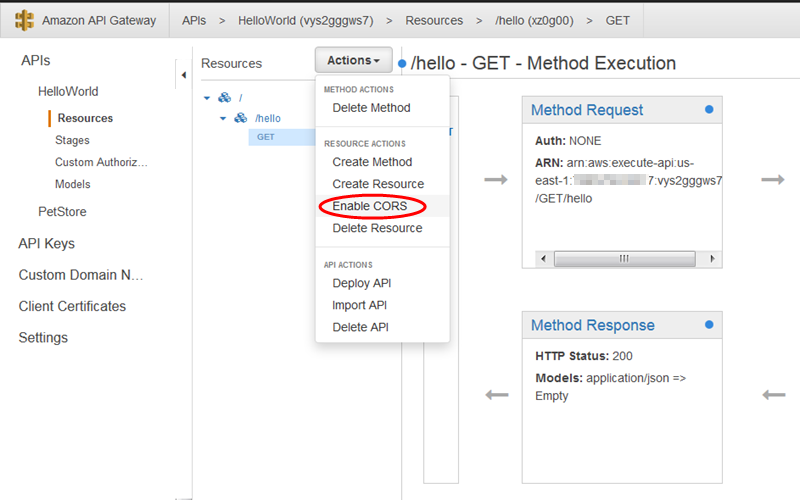
Quality of service policies are typically tied to rate limiting and throttling of incoming API requests. A rate limiting policy is tracking how many incoming API requests are allowed within a given time (e.g., in an hour or in a day). A throttling policy is fairly sophisticated and is not available out of the box with many gateways. Mulesoft has a provided throttling policy which can be used to throttle requests for a given API:



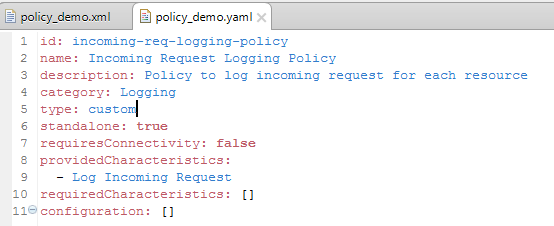
## Developing API Policies

To develop an API policy, different API gateways provide frameworks, plugin SDKs, or policy XML-based templates. AWS and Azure both provide a simple web-based graphical interface to add a new policy or modify an existing policy. Both have an XML-based policy template which can be easy to use. For complicated business logic required, both provide ability to script additional configurations via its built-in policy scripting language. Mulesoft provides ability to create custom XML and YAML based policy files and add policy logic either as a Groovy or Java operations. Kong comes with a Lua-based plugin SDK which can be used to create custom policy plugins.

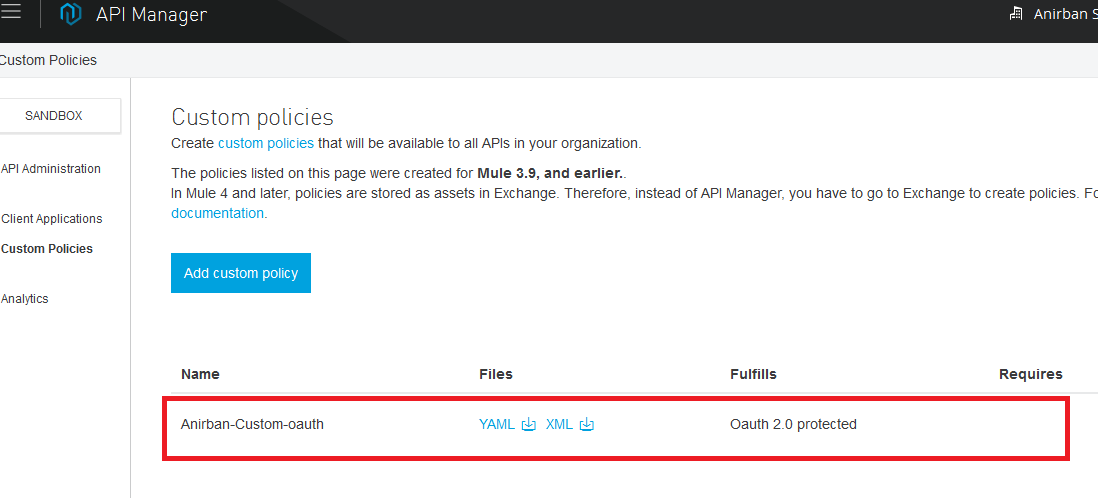
In the AWS API Gateway, a policy administrator would apply one or more policies by adding them via “Actions” in the API “Resources” dashboard. For example, to add CORS compliance policy, an administrator would click on “Enable CORS” on a specific operation of the API (or on all operations, if needed):



Once policy files are created, the policy can be deployed into an API Gateway and associated with one or more APIs. Certain gateways like Mulesoft provides ability to add custom inputs for a policy via its YAML-based policy input configuration. In YAML, a developer can instruct how the policy form should be presented, what form fields should be displayed, and what HTML controls should be associated with each input (e.g., textbox, checkbox, select list, checkbox list, etc.)



When a custom policy is deployed, it will show up as a custom policy or plugin in the API Manager interface of the gateway:



Managing API Policies

To properly manage custom API policies, all custom policy files should be versioned via a source control similar to how API development assets are managed in a source control repository. Mulesoft and Kong provide an ability to create custom policy/plugin projects, and these projects need to be checked in and managed in a source control repository. AWS and Azure manage these policies directly inside their gateways, so there are no external source control repositories required for those platforms.

Once a policy is applied to any given API, a series of regression tests should be executed to validate that policy changes didn’t break any of the existing API behavior. This can be performed via Postman or SoapUI test script execution to validate that the impacted APIs are responding to sample requests as expected.

Provided API policies that come bundled with the API gateway platform are managed and source controlled directly by the API gateway vendor. With each new major API gateway release, the vendor typically releases new and updated provided policies. For example, Mulesoft has released many new provided policies with the Mule 4.x release, which weren’t available in 3.x releases.

It is recommended to have a controlled access to all corporate policies and their application to the existing APIs. Since applying a policy can break the entire API and all of its operations, it is important to have a dedicated team of administrators be responsible for applying these policies to new APIs. Developers can develop and deploy the APIs, but the security and compliance administrator group should be responsible for applying these policies in the API gateway via the management portal.

In many gateway platforms, the API developer role is separated from the API administrator role, who has access to apply different policies.