

ADVANCED TOPICS

ADVANCED TOPICS

GITLAB AND GITFLOW

GitLab

- Request GitLab access using this [form](#)

GitFlow

- a branching model for Git, created by Vincent Driessen
- very well suited to collaboration and scaling the development team

Key Benefits

- Parallel Development
 - new development are isolated from finished work
 - new development is done in **feature** branches, and will only be merged back to the **develop** branch if new feature is done
 - for *interruptions*, commit and then create new feature
 - when finished with new feature, commit and then checkout previous task
- Collaboration
 - **specific branches for features** so developers can easily collaborate on the same feature

GitFlow

Key Benefits

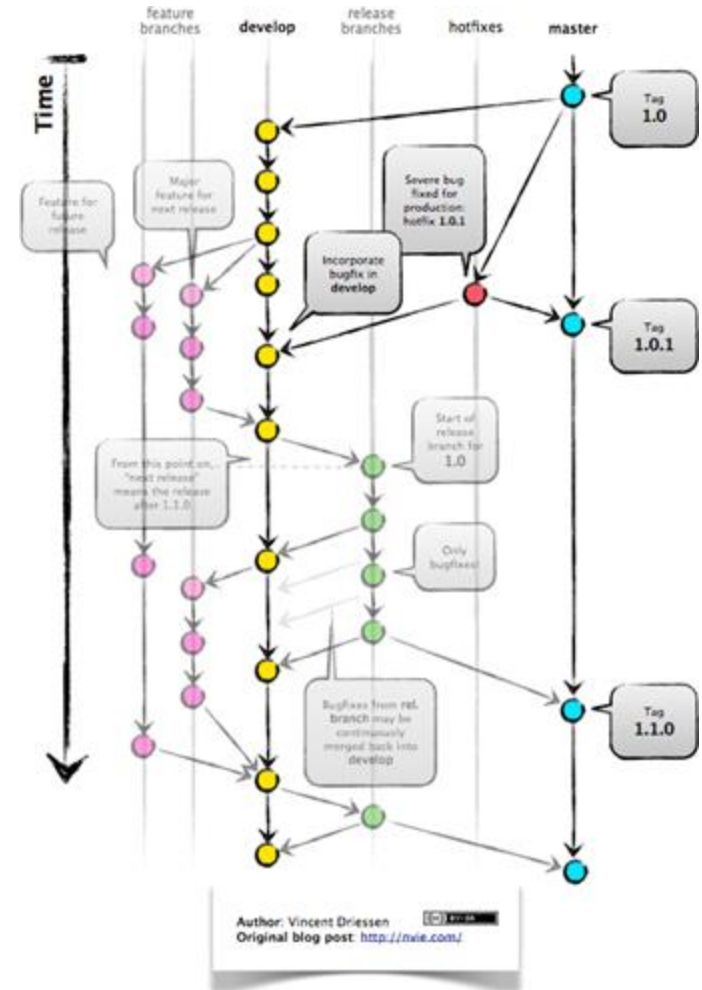
- Release Staging Area
 - **develop** branch - a *staging area* for all completed features that have not been released yet
- Support For Emergency Fixes
 - **hotfix** branches - branches made from a tagged release for making emergency fixes, and when finished, merged back into both **master** and **develop** to make sure the hotfix is not accidentally lost

References:

- <https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow>
- <https://nvie.com/posts/a-successful-git-branching-model/>

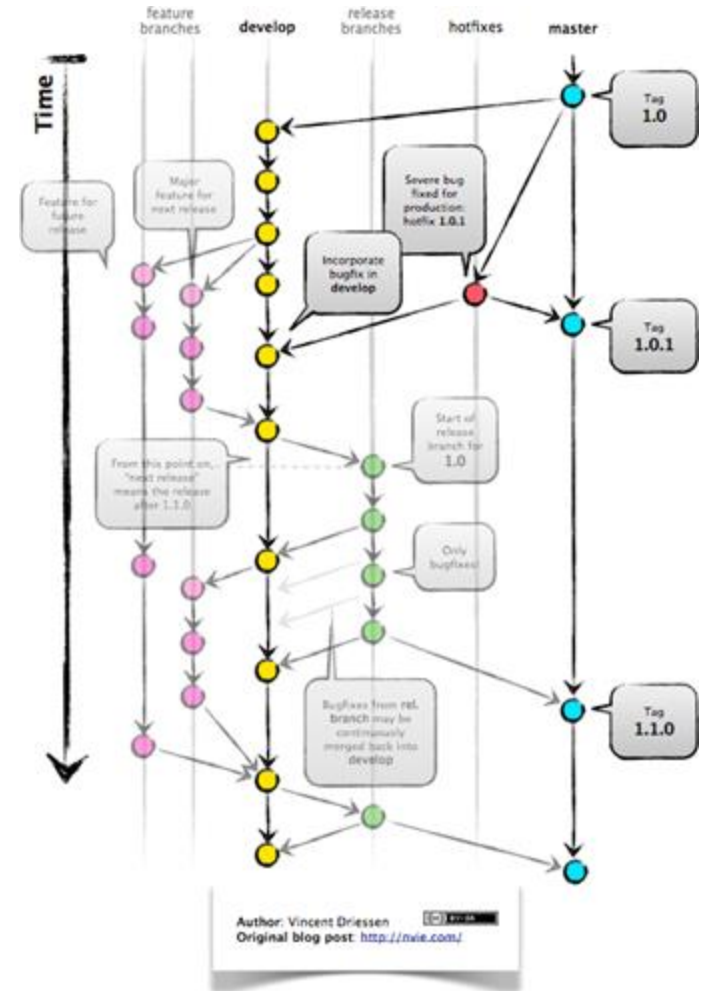
GitFlow

- new development (features) are built in **feature** branches branched off from the **develop** branch and will be merged back to **develop** branch when ready for release
- when ready for release, **release** branch is branched off from **develop** to a suitable test environment, tested, and problems are fixed directly in the release branch.
- **deploy** → **test** → **fix** → **redploy** → **retest**



GitFlow

- when the release is finished, the **release** branch is merged to **master**, and **develop** branches to make sure changes are not accidentally lost
- **master** branch tracks release code only, and the only commits in this branch are merges from **release**, and **hotfix** branches
- **hotfix** branches are branched off directly from a tagged release in the **master** branch, and when finished are merged back to **master**, and **develop**



ADVANCED TOPICS

SECURE CONFIGURATION PROPERTIES

Secure Configuration Properties

- encrypting configuration properties as another security level for applications
 1. Create a source configuration properties file.
 - o .yaml or .properties
 2. Define secure properties by enclosing the encrypted values between the sequence `![value]`.
 3. Encrypt the property using **Secure Properties Tool**.
 4. Configure the file in the project with the **Mule Secure Configuration Properties Extension Module**.
 - o via XML or in Studio
 - o use `secure::` to load the key

Encrypting

Text Strings

```
java -cp
  secure-properties-tool.jar
  com.mulesoft.tools.SecurePropertiesTool \
string \           → method
encrypt \         → operation
Blowfish \        → algorithm
CBC \
                → mode

mulesoft \         → key
"some value to encrypt" → value
```

Properties Inside a File

```
java -cp
  secure-properties-tool.jar
  com.mulesoft.tools.SecurePropertiesTool \
file \           → method

encrypt \         → operation
Blowfish \        → algorithm
CBC \
                → mode

mulesoft \         → key
example_in.yaml    → input file
example_out.yaml   → output
file
```

All Contents of a File

```
java -cp
  secure-properties-tool.jar
  com.mulesoft.tools.SecurePropertiesTool \
file-level \      → method
encrypt \         → operation
Blowfish \        → algorithm
CBC \
                → mode

mulesoft \         → key
example_in.yaml    → input file
example_out.yaml   → output
file
```

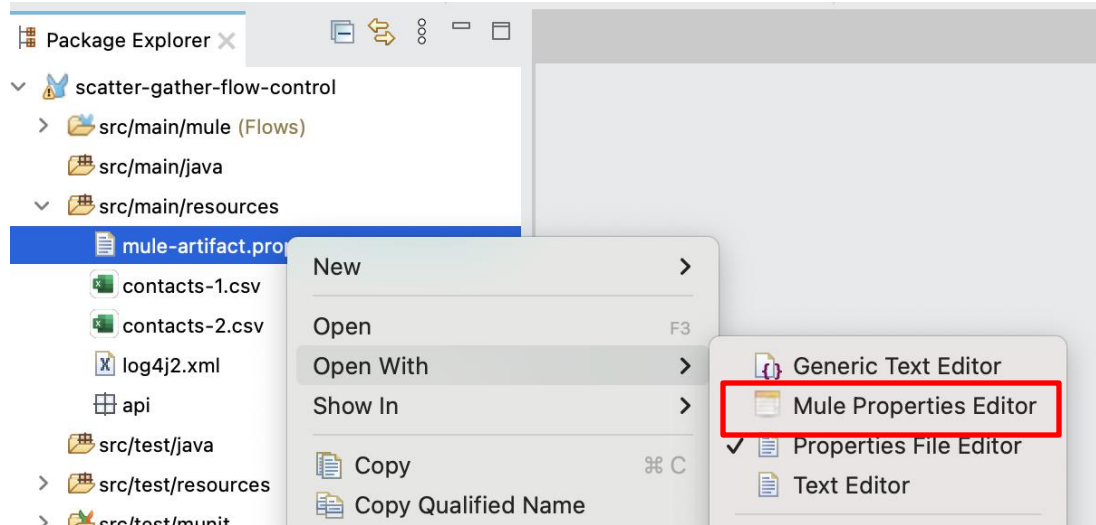
Reference:

- <https://docs.mulesoft.com/mule-runtime/latest/secure-configuration-properties>

Encryption with Anypoint Studio (1/4)

- Encryption Tool Installation
 - <https://help.salesforce.com/s/articleView?id=001123741&type=1>

Encryption with Anypoint Studio (2/4)



Open the properties file using the Mule Properties Editor

Encryption with Anypoint Studio (3/4)

Package Explorer

scatter-gather-flow-control

src/main/mule (Flows)

src/main/java

src/main/resources

mule-artifact.properties

contacts-1.csv

mule-artifact.properties

Name	Value
key	value

Edit property

Type the value of the property to modify

Key: key

Value: value

Encrypt

Cancel

OK

Setup encryption information

Type the key and choose the encryption algorithm.

Algorithm: AES

Key:

Cancel

OK

Double click

Click

Select
Algorithm
and supply
your own
encryption
key

Encryption with Anypoint Studio (4/4)

The screenshot illustrates the final step of encrypting a property in Anypoint Studio. The 'Edit property' dialog is open, showing the 'Key' as 'key' and the 'Value' as '![Cd4ZcRekjgAZXV2rCcrzMw==]'. The 'Value' field is highlighted with a red box. Below the dialog, the 'Package Explorer' on the left shows the project structure, with 'mule-artifact.properties' selected. On the right, the properties file is open, showing a table with the same key and value. The 'Value' field in the table is highlighted with a blue box. A large orange arrow points from the 'OK' button in the dialog to the highlighted value in the table, indicating the action to save the changes.

Edit property
Type the value of the property to modify

Key: key

Value: ![Cd4ZcRekjgAZXV2rCcrzMw==]

Decrypt

Cancel OK

Click

Package Explorer

- scatter-gather-flow-control
 - src/main/mule (Flows)
 - src/main/java
 - src/main/resources
 - mule-artifact.properties
 - contacts-1.csv

*mule-artifact.properties

Name	Value
key	![Cd4ZcRekjgAZXV2rCcrzMw==]

Save the file

ADVANCED TOPICS

MASK PROPERTIES IN CLOUDHUB

Mask Properties in CloudHub

- CloudHub supports safely hidden application properties, in which the name of the property is visible in Anypoint Runtime Manager, but the value is not displayed or retrievable by any user. CloudHub resolves the property at runtime without exposing the sensitive information.

Mask Properties in CloudHub

- to create safely hidden application properties:
1. Add the properties to the appropriate file:
For Mule 4.0 and later, in the `mule-artifact.json` file under the `secureProperties` key, list the property names to safely hide as a comma-separated array:
 2. Deploy the application to CloudHub.
 3. In the **Applications** page, click the application name and then click **Settings**.

```
{
  "configs": [
    "testprops-app.xml"
  ],
  "secureProperties": ["password", "birthdate"],
  "redploymentEnabled": true,
  "name": "secure-properties",
  "minMuleVersion": "4.2.1",
  "requiredProduct": "MULE_EE",
  "classLoaderModelLoaderDescriptor": {
    "id": "mule",
    "attributes": {
      "exportedResources": []
    }
  },
  "bundleDescriptorLoader": {
    "id": "mule",
    "attributes": {}
  }
}
```

Mask Properties in CloudHub

4. In the **Properties** tab, click List and then enter your application.
5. Click **Apply Changes**.
6. Redeploy or restart the app.

The diagram illustrates the process of masking properties in CloudHub. It shows two states of a properties list, connected by a large downward arrow. The top state shows unmasked values, and the bottom state shows masked values (passwords replaced with asterisks). A blue arrow points to the password field in the bottom state.

username	testuser	×
password	testpass123	×
birthdate	11/12/2006	×

↓

username	testuser	×
password	*****	×
birthdate	*****	×

Mask Properties in CloudHub

```
{
  "minMuleVersion": "4.2.2",
  "secureProperties": [
    "dynamodb.accesskey",
    "dynamodb.secretkey",
    "dynamodb.tablename"
  ]
}
```

● sys-whitelist-api-secure-properties

Stop ▾



Application File

sys-whitelist-api-1.0.0-SNAPSHOT-mule-app...

Choose file ▾

Get from sandbox

Last Updated: 2020-05-11 8:43:21AM

App url: sys-whitelist-api-secure-properties.us-e2.cloudhub.io

Runtime	Properties	Insight	Logging	Static IPs
Text	List			
anypoint.platform.client_id	e3b5985a9ade4c80bd397760e15c44cc			×
dynamodb.accesskey	*****			×
dynamodb.secretkey	*****			×
anypoint.platform.client_secret	c4e6468e6E7E4622869f618001844570			×
dynamodb.tablename	*****			×

ADVANCED TOPICS

API Specifications as Exchange Dependencies

API Specifications as Exchange Dependencies

Reference: <https://docs.mulesoft.com/studio/latest/api-development-studio#api-specifications-as-exchange-dependencies>

ADVANCED TOPICS

DEPLOY TO CLOUDHUB
(MAVEN DEPLOYMENT)

Deploy To CloudHub (Maven Deployment)

- Prerequisite:
 - Apache Maven
 - Connected app ([reference](#))
- include these in pom.xml, and configure deployment strategy

Mule Maven Plugin

```
<plugin>
  <groupId>org.mule.tools.maven</groupId>
  <artifactId>mule-maven-plugin</artifactId>
  <version>3.4.0</version>
  <extensions>true</extensions>
  <configuration>
    <classifier>mule-application</classifier>
    <cloudHubDeployment>
      <uri>https://anypoint.mulesoft.com</uri>
      <muleVersion>${app.runtime}</muleVersion>
      <connectedAppClientId>${connectedAppClientId}</connectedAppClientId>
      <connectedAppClientSecret>${connectedAppClientSecret}</connectedAppClientSecret>
      <connectedAppGrantType>client_credentials</connectedAppGrantType>
    </cloudHubDeployment>
  </configuration>
</plugin>
```

Mule Maven Plugin Repository

```
<pluginRepositories>
  <pluginRepository>
    <id>mule-public</id>
    <url>https://repository.mulesoft.org/
      nexus/content/repositories/releases</url>
  </pluginRepository>
</pluginRepositories>
```

Deploy To CloudHub (Maven Deployment)

Mule Maven Plugin (more detailed)

```
<plugin>
  <groupId>org.mule.tools.maven</groupId>
  <artifactId>mule-maven-plugin</artifactId>
  <version>${mule.maven.plugin.version}</version>
  <extensions>true</extensions>
  <configuration>
    <classifier>mule-application</classifier>
    <cloudHubDeployment>
      <uri>https://anypoint.mulesoft.com</uri>
      <muleVersion>${app.runtime}</muleVersion>
      <connectedAppClientId>${connectedAppClientId}</connectedAppClientId>
      <connectedAppClientSecret>${connectedAppClientSecret}</connectedAppClientSecret>
      <connectedAppGrantType>client_credentials</connectedAppGrantType>
      <applicationName>${cloudhub.application.name}</applicationName>
      <environment>${cloudhub.environment}</environment>
      <workers>${cloudhub.workers}</workers>
      <workerType>${cloudhub.worker.type}</workerType>
      <objectStoreV2>${cloudhub.objectstorev2}</objectStoreV2>
      <region>${cloudhub.region}</region>
      <properties>
        <anypoint.platform.client_id>${anypoint.platform.client_id}</anypoint.platform.client_id>
        <anypoint.platform.client_secret>${anypoint.platform.client_secret}</anypoint.platform.client_secret>
      </properties>
    </cloudHubDeployment>
  </configuration>
</plugin>
```


Deploy To CloudHub (Maven Deployment)

- in terminal run the following commands:
`mvn clean package deploy -DmuleDeploy`
- add runtime properties in the command
`-DconnectedAppClientId=connectedAppClientId`
`-DconnectedAppClientSecret=connectedAppClientSecret`

Reference:

- <https://docs.mulesoft.com/mule-runtime/latest/deploy-to-cloudhub>
- <https://docs.mulesoft.com/access-management/connected-apps-developers>

Deploy To CloudHub (Maven Deployment)

sys-whitelist-api-mac-test

CloudHub

Started

4.2.2

2020-05-06 13:37:38

```
--- mule-maven-plugin:3.3.5:deploy (default-deploy) @ sys-whitelist-api ---  
Deploying artifact sys-whitelist-api-mac-test  
Creating application: sys-whitelist-api-mac-test  
Starting application: sys-whitelist-api-mac-test  
Checking if application: sys-whitelist-api-mac-test has started  
Artifact sys-whitelist-api-mac-test deployed
```

BUILD SUCCESS

```
-----  
Total time: 20:05 min  
Finished at: 2020-05-06T13:38:39+08:00  
-----
```

GET

Params Auth Headers (9) **Body** Pre-req. Tests Settings

This request does not have a body

Body Cookies Headers (5) Test Results 200 OK 2.60 s 252 B

Pretty Raw Preview Visualize

```
1 {  
2   "transactionId": "20200204-004-003",  
3   "mobileNumber": "9171234567",  
4   "status": "0"  
5 }
```

ADVANCED TOPICS

DEPLOY TO CLOUDHUB 2.0
(MAVEN DEPLOYMENT)

Deploy To CloudHub 2.0 (Maven Deployment)

- Prerequisite:
 - Apache Maven
 - Connected app ([reference](#))
- include these in pom.xml, and configure deployment strategy

Mule Maven Plugin

```
<plugin>
  <groupId>org.mule.tools.maven</groupId>
  <artifactId>mule-maven-plugin</artifactId>
  <version>3.7.1</version>
  <extensions>true</extensions>
  <configuration>
    <cloudhub2Deployment>
      <uri>https://anypoint.mulesoft.com</uri>
      <provider>MC</provider>
      <environment>${environment}</environment>
      <target>${targetName}</target>
      <muleVersion>${muleVersion}</muleVersion>
      <connectedAppClientId>${connectedAppClientId}</connectedAppClientId>
      <connectedAppClientSecret>${connectedAppClientSecret}</connectedAppClientSecret>
      <connectedAppGrantType>client_credentials</connectedAppGrantType>
    </cloudhub2Deployment>
  </configuration>
</plugin>
```

Mule Maven Plugin Repository

```
<pluginRepositories>
  <pluginRepository>
    <id>mule-public</id>
    <url>https://repository.mulesoft.org/
      nexus/content/repositories/releases</url>
  </pluginRepository>
</pluginRepositories>
```

Deploy To CloudHub 2.0 (Maven Deployment)

Mule Maven Plugin (more detailed)

```
<plugin>
  <groupId>org.mule.tools.maven</groupId>
  <artifactId>mule-maven-plugin</artifactId>
  <version>3.7.1</version>
  <extensions>true</extensions>
  <configuration>
    <cloudhub2Deployment>
      <uri>https://anypoint.mulesoft.com</uri>
      <provider>MC</provider>
      <environment>${environment}</environment>
      <target>${targetName}</target>
      <muleVersion>${muleVersion}</muleVersion>
      <connectedAppClientId>${connectedAppClientId}</connectedAppClientId>
      <connectedAppClientSecret>${connectedAppClientSecret}</connectedAppClientSecret>
      <connectedAppGrantType>client_credentials</connectedAppGrantType>
      <applicationName>${appName}</applicationName>
      <replicas>1</replicas>
      <vCores>1</vCores>
      <deploymentSettings>
        <http>
          <inbound>
            <publicUrl>${publicUrl}</publicUrl>
            <forwardSslSession>true</forwardSslSession>
            <lastMileSecurity>true</lastMileSecurity>
          </inbound>
        </http>
      </deploymentSettings>
    </cloudhub2Deployment>
  </configuration>
</plugin>
```

Deploy To CloudHub 2.0 (Maven Deployment)

- in terminal run the following commands:
`mvn clean package deploy -DmuleDeploy`
- add runtime properties in the command
`-DconnectedAppClientId=connectedAppClientId`
`-DconnectedAppClientSecret=connectedAppClientSecret`

Reference:

- <https://docs.mulesoft.com/mule-runtime/latest/deploy-to-cloudhub-2>
- <https://docs.mulesoft.com/access-management/connected-apps-developers>

ADVANCED TOPICS

HTTPS IMPLEMENTATION LOCAL

HTTPS Implementation Local

- **Keystore** is a server-side asset that stores the private keys and the certificates with their public and private keys
 1. Create the **keystore** and certificate
 2. Include the **keystore** to the resources
 3. Change the HTTP Protocol to HTTPS Protocol.
 4. Go to **TLS Tab** add the **Key Store Configuration**.
 5. Test using the https:// endpoint.

References:

- <https://dzone.com/articles/using-https-in-mule>
- <https://docs.mulesoft.com/mule-runtime/4.6/tls-configuration>

The image shows two screenshots of the MuleSoft configuration interface. The top screenshot displays the 'General' tab of a 'Connection' configuration, where the 'Protocol' is set to 'HTTPS', the 'Host' is 'All interfaces [0.0.0.0] (default)', and the 'Port' is '8081'. The bottom screenshot shows the 'TLS' tab, specifically the 'Key Store Configuration' section. It includes fields for 'Path' (set to 'keystore/keystore.jks'), 'Alias' (set to 'mule'), 'Key Password', and 'Password', each with a 'Show password' checkbox. The 'Type' is set to 'JKS' and the 'Algorithm' is left blank. There is also an 'Insecure' checkbox and an 'Advanced' section at the bottom for 'Enabled Protocols'.

ADVANCED TOPICS

API AUTODISCOVERY

API Autodiscovery

- a mechanism that manages an API from API Manager by pairing the deployed application to an API created on the platform
- API Management includes tracking, enforcing policies if you apply any, and reporting API analytics
- critical to the Autodiscovery process is identifying the API by providing the API name and version

References:

- <https://developer.mulesoft.com/tutorials-and-howtos/quick-start/deploying-managing-your-first-api>
- <https://docs.mulesoft.com/api-manager/1.x/api-auto-discovery>
- <https://docs.mulesoft.com/mule-gateway/mule-gateway-config-autodiscovery-mule4>

API Autodiscovery

1. Go to **Management Center > API Manager**.
2. Click the **Manage API button** and then **Manage API from Exchange**.
3. Select the API you want to manage, and the other fields will be automatically filled. Tick the checkbox for Mule 4 application. Save.
4. After that, you will see an **API ID for Autodiscovery**. Copy this.
5. Get the Environment credentials from **Management Center > Access Management > Environments** and select the environment you will be using. Get the **Client ID** and **Client Secret**, these will be used in the deployment.

API Autodiscovery

6. Go to Anypoint Studio. Create an **API Autodiscovery Configuration**. Set the API ID as from step 4. Set your main flow as the Flow Name.
7. Deploy the application with the following properties in Cloudhub:

The screenshot shows the 'Properties' tab in Anypoint Studio. At the top, there are tabs for 'Runtime', 'Properties' (selected), 'Insight', 'Logging', and 'Static IPs'. Below these are two sub-tabs: 'Text' and 'List' (selected). The main area contains a table with three rows of properties. The first row has 'anypoint.platform.client_id' and 'YOUR_CLIENT_ID'. The second row has 'anypoint.platform.client_secret' and 'YOUR_CLIENT_SECRET'. The third row has 'key' and 'value'. Each row has a close button (X) on the right. At the bottom right, there are two buttons: 'Cancel' and 'Deploy Application'.

Runtime	Properties	Insight	Logging	Static IPs
<div>Text List</div>				
anypoint.platform.client_id		YOUR_CLIENT_ID		X
anypoint.platform.client_secret		YOUR_CLIENT_SECRET		X
key		value		
<div>Cancel Deploy Application</div>				

6. You can now go back to API Manager and see that the "Status" column of your API is now marked as Active with a green dot next to it. This means the app has successfully registered with the gateway and you can manage it.

ADVANCED TOPICS

CLIENT ID ENFORCEMENT POLICY


Client ID Enforcement Policy

- restricts access to a protected resource by allowing requests only from registered client applications
- ensures that the client credentials sent on each request have been approved to consume the API.

Reference:

- <https://docs.mulesoft.com/gateway/latest/policies-included-client-id-enforcement>

Client ID Enforcement Policy

1. Go to **Management Center > API Manager**.
2. Select the API to add the policy and the select **Policies**.
3. Click **Apply New Policy** button and select **Client ID Enforcement**.
4. Go to **Exchange** and select the API.
5. On the top right corner, click the  button, and **Request Access** button and this will give **client_id** and **client_secret** of the application.
6. Add **client_id** and **client_secret** to the header of your request.

ADVANCED TOPICS

HTTP PROXY IN CLOUDHUB

HTTP Proxy in CloudHub

1. Go to **API Manager**.
2. Select the API you want to put a proxy on.
3. Go to **Settings**, and in the API Configuration part, select **Endpoint with Proxy**.
4. Fill the Implementation URI field.
5. Under Advanced Options, set the scheme to HTTP.
6. Save.

Advanced options ▾

Proxy version: 3.0.0 ▾

[For more information on different proxy versions please visit this page](#)

Validations:

☒ Validate the inbound requests against the provided specification.

Strict validations (optional)

☐ Query params

Scheme:

☒ HTTP ☐ HTTPS

Port:

8081 ⓘ

Response timeout: ⓘ
(Optional)

10000

☐ Reference user domain ⓘ

Save

API Configuration ▾

ADD A TAG

Managing type:

☐ Basic Endpoint ☒ Endpoint with Proxy

Proxy deployment target:

☒ CloudHub

☐ Hybrid

Your proxy is already deployed, to change this setting please remove the application from cloudhub

Mule version:

☒ Select if you are managing this API using Mule 4 or above.

Your proxy is already deployed, to change this setting please remove the application from cloudhub

Implementation URI:

<http://sys-whitelist-api-clientid.us-e2.cloudhub.io/api/>

TLS Context for outbound traffic:

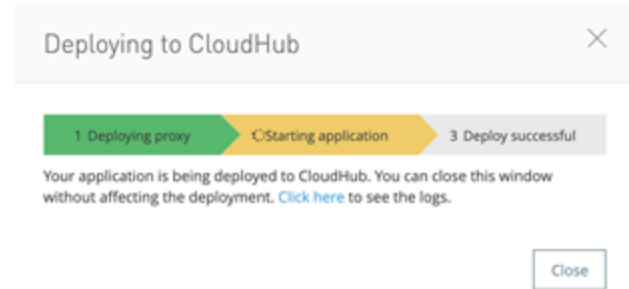
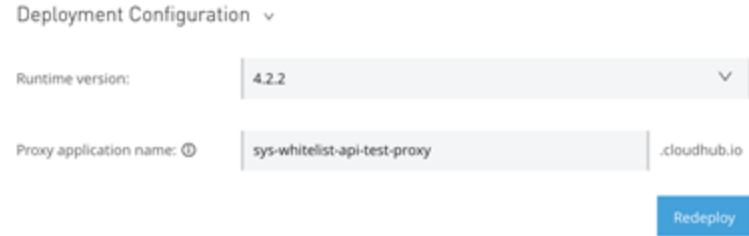
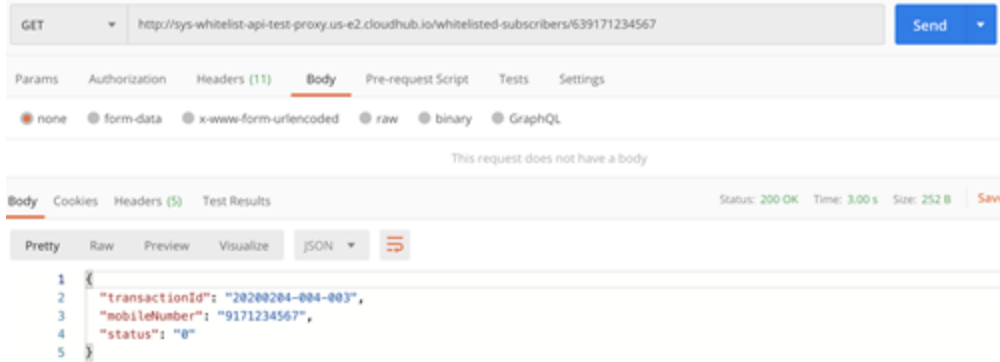
[+ Add TLS Context](#) Used to secure outbound traffic

Path:

/

HTTP Proxy in CloudHub

- Under the Deployment Configuration, specify the **Runtime Version** and the **Proxy Application Name**, then **Deploy/Redeploy**.
- Wait for the proxy to be deployed.
- Test if the proxy is reachable.



References:

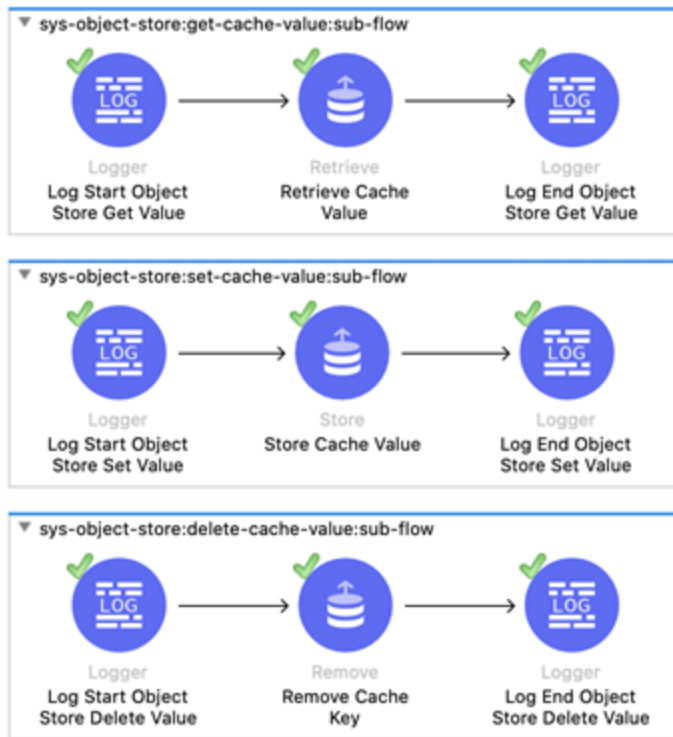
- <https://docs.mulesoft.com/api-manager/2.x/proxy-deploy-cloudhub-latest-task>

ADVANCED TOPICS

OBJECT STORE INSTEAD OF REDIS

Object Store Instead of Redis

- things to note:
 - Unlike Redis, Object Store is not capable of automatically expiring the cache key. You will need to revise the implementation for the expiration (i.e. including expiryDate in the cache value, checking if the expiryDate is expired or not)



END