Setting up and configuring Edge Microgateway

Edge Microgateway v. 2.3.x

Overview

Note: You can only use Edge Microgateway with paid accounts for Edge Public Cloud. If you're using Edge Private Cloud, you can use Edge Microgateway with both paid and free trial accounts. See the Edge pricing features page (https://apigee.com/about/pricing/apigee-edge-pricing-features) for more details.

This tutorial walks you through the steps required to get an instance of Edge Microgateway up and running.

After completing the steps here, you'll have a fully configured, working Edge Microgateway installation capable of processing API requests. You'll test the setup by making secure API calls through Edge Microgateway to a backend target. You will also learn how to add a spike arrest plugin to the Microgateway.

This guide is divided into these parts:

- **Prerequisite:** Installing Edge Microgateway (#Prerequisite)
- **Part 1:** Configure Edge Microgateway (#Part1)
- **Part 2:** Create entities on Apigee Edge (#Part2)
- **Part 3:** Operate Edge Microgateway (#Operating%20Edge%20Microgateway)
- **Part 4:** Secure Edge Microgateway (#Secure)
- **Part 5:** Add the Spike Arrest plugin (#Part4)
- **Part 6:** View Analytics data on Apigee Edge (#Part5)

Prerequisite: Install Edge Microgateway

Follow the instructions in Installing Edge Microgateway (https://docs.apigee.com/api-platform/microgateway/2.3.x/installing-edge-microgateway-v2.3.x.html).

When you complete the installation, you'll be ready to follow the steps in this tutorial.

When you are finished with the installation, proceed to the next section, "Part 1: Configure Edge Microgateway (#Part1)".

Part 1: Configure Edge Microgateway

In this part you'll use a command-line interface (CLI) command to configure Edge Microgateway to communicate with Apigee Edge. If you are using Apigee Edge Cloud, then follow the Apigee Edge Cloud configuration steps (#Cloud%20config). If you are on Apigee Private Cloud, follow the steps for Apigee Edge Private Cloud (#Private%20config).

Note: Why does Edge Microgateway need to communicate with Edge? See Dependency on Apigee Edge (https://docs.apigee.com/api-platform/microgateway/2.3.x/overview-edge-microgateway-v2.3.x.html#whatyouneedtoknowaboutedgemicrogateway-dependencyonapigeeedge)
.

Apigee Edge Cloud configuration steps

Follow these steps to use Edge Microgateway with Apigee Edge Cloud:

1. If you haven't done so previously, initialize Edge Microgateway (you only need to do this step one time):

   ```
   edgemicro init
   ```

2. (Optional) Print help for the **edgemicro configure** command:

   ```
   edgemicro configure -h
   ```

3. Execute the following command to configure Edge Microgateway:

   ```
   edgemicro configure -o [org] -e [env] -u [username]
   ```

   Where:

   - **org** is your Edge organization name (you must be an org administrator).

- **env** is an environment in your org (such as test or prod).
- **username** is the email address associated with your Apigee account.

Example

```
edgemicro configure -o docs -e test -u jdoe@example.com
```

Output

```
./edgemicro configure -o docs -e test -u jdoe@apigee.com
password:
current nodejs version is v6.1.0
current edgemicro version is 2.2.3-beta
password:
file doesn't exist, setting up
Give me a minute or two... this can take a while...
App edgemicro-auth deployed.
creating vault
adding private_key
adding public_key
configuring host edgemicroservices-us-east-1.apigee.net for region us-east-1

saving configuration information to: /Home/.edgemicro/wwitman-test-config.yaml

vault info:
 -----BEGIN CERTIFICATE-----
MIICpDCCAYwCCQCaDpaTttaDANBgkqhkiG9w0BAQsFADAUMRIwEAYDVQQDEwls
b2NhbGhvc3QwHhcNMTYxAxMjA0NzIyWhcNMTYxMTAyMjA0NzIyWjAUMRIwEAYD
VQQDEwlsb2NhbGhvcwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDA
YbtbsFe50GgBrI8QTih5yzxxxvCfdubGrLWmovwwCCFthno3u8lS54ekOL9jQu
JTJsocJfNhZxvnXifDsCk5Muwx87Z/D0BYLw9ZhM8TMyA2MCkXgC6RfKT69IdJ
jT3X+3D0s+apr3lACsDhgOfaeQfeHAm1SSwH/VemaJqdImGkQMtM8uscMDwG6k
vBbCSNS+hh6ZH1m3tAkcKXqvj/E1hbrHNTWr+uiYFRByUzeo1I607daQD4Lxid
il4Ng3hr3LC1gEzvobWgVyhQ2ovYB57T886H7dGghTnOUxLm2y/TwmQya+44SL
JSsDwauArMF38cRKZZ7VAgMBAAEwDQYJKoZIhvcNAQELBQADEBAHwwu+gOn8Tq
jKOYBTXtOX11HPMTxmyvZ/I57R4roE7Z/lXg/DXwbiFpFG2uamXNKq7yTDJH2i
iiqdERZOGfv7TJMjRlxEnfVHoCV7UKguKq4zeeAEU2x55yFwpWNrarx0kMXRwI
v3WgGfo2bimFQrCjDCNIkDKmYYt4SXIF5zUJmBWPCaK9aJoQb7ARXQO9s2zoOI
XZ5bCUDbehQZ+6LyqC0hgDXiIZYy7R0j93yCbQgoHaHUMEprJEEjE24SHdsWBO
nxGZxB20JLq5AHTm8lbZp7XtvbU0jVpxyrBo2/olFnmpvBVOq9eIp042IVD7nT
J4rPejdK6C4=
-----END CERTIFICATE-----
```

```
The following credentials are required to start edge micro
  key: e88ec9d5da17cd88ca338d532b7c7a8c4f43ddd3da139d95955ab345af30352d
  secret: d7f64854eb6889d06b809dd7c161dba5eeba4a2657277fd819921bf992221a8f

edgemicro configuration complete!
```

Note: You'll need the returned **key** and **secret** later when you start Edge Microgateway.

Apigee Private Cloud configuration steps

Follow these steps to use Edge Microgateway with Apigee Private Cloud:

1. Print help information for the **edgemicro private configure** command. You can print help this way for any Edge Microgateway CLI command or command option.

   ```
   edgemicro private configure -h
   ```

2. Execute the following command. It requires standard information about your Apigee Edge Private Cloud account: organization name, environment name, username (email address), password, management server IP and router IP. You must be an Edge organization administrator to use this command:

   ```
   edgemicro private configure -o [org] -e [env] -u [username] -r [runtim
   ```

   Where:

   - **`org`** is your Edge organization name (you must be an org administrator).
   - **`env`** is an environment in your org (such as test or prod).
   - **`runtime_url`** is the runtime URL for your private cloud instance.
   - **`mgmt_url`** is The URL of the management server for your private cloud instance.
   - **`username`** is the email address associated with your Apigee account.
   - **`virtual_host`** is a comma-separated list of virtual host names. The default values are **`default,secure`**

> **Note:** By default, the **edgemicro-auth** proxy expects to connect through a virtual host called **secure**. If your Private Cloud installation does not have this virtual host defined, you will receive a configuration error. Be sure to specify on the command line a **virtual_host** that exists. It is generally safe to specify a virtual host called **default**.

Example

```
edgemicro private configure -o docs -e test -u jdoe@example.com -r http://
```

or, if you have a virtual host alias of `myorg-test.mycompany.com`, you would use a command like this:

```
edgemicro private configure -o docs -e test -u jdoe@example.com -r myorg-t
```

Output

```
delete cache config
checking for previously deployed proxies
configuring edgemicro internal proxy
deploying edgemicro internal proxy
deploying  edgemicro-auth  app
copy auth app into tmp dir
copy config into tmp deer
Give me a minute or two... this can take a while...
App edgemicro-auth added to your org. Now adding resources.

checking org for existing vault creating vault adding private_key adding public_
configuring host http://192.168.52.106:9001 for region dc-1

saving configuration information to: /Users/ApigeeCorporation/.edgemicro/jdoe-te
vault info:
-----BEGIN CERTIFICATE-----
MIICpDCCAYwCCQDpIvWlpaZJGDANBgkqhkiG9w0BAQFADAUMRIwEAYDVQQDEwls
b2NhbGhvc3QwHhcNMTYwNDA3MTcxNTM5WhcNMTYwND4MTcxNTM5WjAUMRIwEAYD
VQQDEwlsb2NhbGhvc3QwggEiMA0GCSqGSIb3DQEBAUAA4IBDwAwggEKAoIBAQD3
OAQ+kf5FH0S0yuj05ITqUODuUJspBPberRMbqOZYHcFswhB0Yvg6JKWxKWBDP9o
Xl96dtgH7xPFRqIU0zI452jkMQ1fPz2mSaGwik245yfBku7olooXKRKTRKOUoXa
q3Hld/RPxGSsWtiyyYtKex7tuFdq0Knm1EhowdTRGIgjNvudeYMka/XPRXuykhd
xIDxWj4rdX+4GPx9qT2eTQC5nOAC7XHVL7ys4KqsAiv28vw10u400KstFFS8Qho
7FaE0bOKLolKKadKyA60ha1XIw/uSTD6ZQFWQ+XM3OaRbXePWXnSZioSxXcZT7L
hMUKbsRXw/TCvRB51LgNAgMBAAEwDQYJKoZIhvcNAQELBQADgEBAOuR1OmE/W6j
```

```
gRAarJB5EQuTEpI/9Zpg5c5RAGjzLhkazsycn7pal+IymUjCV7D0oIxTVuTM8ZZ
57kR5hF/C1ZypDN9i+KGP2ovX8WOCCXYtIQECgZPB/L/7/k7BDuKN4vFBvWUe3X
s2oyjnVWy+ASqsW8gHfj8ekhe22bP240Oqkbyg9SZP9ol1tvJX6+M0thYwcTwAd
ft929Icey/MOTQC0jB2qm0gnIx/7KInFXfS5KoFRJoGrWDeckr3RdBo2LhnPaeZ
1gPYIqphZ3HfisF5KlBXoR8oT/Ilym/nq5COlv+3L4tMIk18F7BQZB60SRazifz
pFkIxepyr/0=
-----END CERTIFICATE-----

The following credentials are required to start edge micro
   key: a3f8f3dfe39158fc3c50b274f0af2234246e0d5f6ea4dd09389b645147151ba3
   secret: 3e9904802fb3c0e8ca408128a11119cf13546d54dac30ace944c097a726a1263

edgemicro configuration complete!
```

Note: Important: If you have a virtual host alias defined, then use the alias for the `-r <router-ip>` parameter. You can view virtual hosts in the Edge UI for your organization, under **APIs > Environment Configuration > Virtual Hosts**.

Verify the installation

Run this command to verify the installation. If no errors are reported, everything is set up correctly and you will be able to start the Edge Microgateway successfully.

```
edgemicro verify -o [org] -e [env] -k [key] -s [secret]
```

Where:

- **org** is your Edge organization name (you must be an org administrator).
- **env** is an environment in your org (such as test or prod).
- **key** is the key returned previously by the configure command.
- **secret** is the key returned previously by the configure command.

Example

```
edgemicro verify -o docs -e test -k 93b01fd21d86331459ae52f664ae9aeb13eb94
```

About the configuration

All of the configuration done so far allows Edge Microgateway to bootstrap itself to Apigee Edge. After the bootstrapping succeeds, Edge Microgateway retrieves a payload of additional configuration information from Apigee Edge.

What is this configuration information used for? As we'll discover in the next part of this tutorial, when Edge Microgateway starts, it needs to receive a list of special Edge Microgateway-aware API proxies from Apigee Edge. In the next part of this tutorial, you will create a Microgateway-aware proxy. Edge Microgateway restricts clients to calling only the APIs fronted by these Microgateway-aware API proxies, and clients will be required (by default) to present a valid security token for each call. To read more about these proxies, see "What you need to know about Edge Microgateway-aware proxies in the Overview of Edge Microgateway (https://docs.apigee.com/api-platform/microgateway/2.3.x/overview-edge-microgateway-v2.3.x.html).

As an Edge org admin, you'll be interested to know that Edge Microgateway-aware proxies can be added to Edge products, just like any other proxies. Through the use of products and developer apps, you can generate client-specific security tokens to control access to APIs called through Edge Microgateway. Again, the patterns involved are identical to working with any API proxies, products, and developer apps on Apigee Edge. If you'd like to read up on products, start with What is an API product? (https://docs.apigee.com/api-platform/publish/what-api-product.html) in the Edge documentation.

Next we'll walk through how to create Edge Microgateway-aware proxies, and after that, we'll start Edge Microgateway and test the setup.

Part 2: Create entities on Apigee Edge

In this part, you will create these entities on Edge:

- **A microgateway-aware proxy** - This is a special proxy that Edge Microgateway can discover upon startup. Microgateway-aware proxies have a naming convention that you must follow: the name must being with **`edgemicro_`**. For example **`edgemicro_hello`** or **`edgemicro_userinfo`**. When Edge Microgateway starts, it retrieves from Edge a list of microgateway-aware proxies from the same Edge organization and environment that you specified when you started Edge Microgateway.

  For each microgateway-aware proxy, Edge Microgatway retrieves the target URL of the

proxy and its base path. Microgateway-aware proxies also provide a convenient way to associate analytics data generated by Edge Microgateway with a proxy on the Edge platform. As the Microgateway handles API calls, it asynchronously pushes analytics data to Edge. Analytics data will show up in the Edge Analytics UI under the microgateway-aware proxy name(s), as it does for any other proxy.

- **A product, developer, and developer app** - Edge Microgateway uses products, developers, and developer apps to enable OAuth2 access token or API key security. When Edge Microgateway starts, it downloads all of the product configurations from your Apigee Edge organization. It uses this information to verify API calls made through Edge Microgateway with API keys or OAuth2 access tokens.

Read more: See also "What you need to know about Edge Microgateway-aware proxies" in the Overview of Edge Microgateway (https://docs.apigee.com/api-platform/microgateway/2.3.x/overview-edge-microgateway-v2.3.x.html).

1. Create an Edge Microgateway-aware API proxy on Edge

Note: These instructions are based on the Classic Edge user interface. After you log in to Edge, click **Switch to Classic** and then follow the steps in this section.

Note: Edge Microgateway-aware proxies must point to an HTTP target endpoint. In other words, the TargetEndpoint for the proxy must include an HTTPTargetConnection. Edge Microgateway is not designed to work with proxies that use the ScriptTarget element to point to Node.js applications as backend targets. See also Endpoint properties reference (https://docs.apigee.com/api-platform/reference/policies/endpoint-properties-reference.html) and Specify the Node.js target with ScriptTarget (https://docs.apigee.com/api-platform/nodejs/adding-nodejs-existing-api-proxy.html#specifythenodejstargetwithscripttarget)
.

Note: Important: Do not attach policies or make conditional flows in microgateway-aware proxies, because they will never execute ON APIGEE EDGE. Microgateway-aware proxies are never called directly ON EDGE -- they only serve to provide configuration information to Edge Microgateway and as a way to surface analytics data in the Edge analytics system. If you want to add policy functionality, such as quota, spike arrest, or OAuth2 security, you need to use Edge Microgateway plugins. For details, see Use plugins (https://docs.apigee.com/api-platform/microgateway/2.3.x/use-plugins-v2.3.x.html). See also Develop

custom plugins (https://docs.apigee.com/api-platform/microgateway/2.3.x/develop-custom-plugins-v2.3.x.html).

1. Log in to your organization on Apigee Edge.
2. Click **SWITCH TO CLASSIC** to go to the Edge Classic UI.
3. Select **APIs > API Proxies** from the top menu.
4. In the API Proxies page, click **+ API Proxy**.
5. In the Build a Proxy wizard, select **Reverse proxy (most common)**.
6. Click **Next**.
7. In the Details page of the wizard, configure as follows. Be sure to fill in the wizard exactly as shown:

★ **Note: Important:** Edge Microgateway-aware proxy names **must always begin** with the prefix **edgemicro_**. For example: `edgemicro_hello`.

   - Proxy Name: **edgemicro_hello**
   - Proxy Base Path: **/hello**
   - Existing API: **http://mocktarget.apigee.net/**
8. Click **Next**.
9. In the Security page of the wizard, select **Pass through (none)**.
10. Click **Next**.
11. In the Virtual Hosts page of the wizard, accept the defaults.
12. Click **Next**.
13. In the Build page of the wizard, review your proxy settings. Make sure the **test** environment is selected.
14. Click **Build and Deploy**.

2. Create a product

Create a product that contains two proxies:

- Your microgateway-aware proxy: **edgemicro_hello**

- The authentication proxy that was installed by Edge Microgateway: **edgemicro-auth**.

1. In the Edge UI (Classic version), go to **Publish > Products**.
2. In the Products page, click **+ Product**. Fill out the Product Details page as follows:
    - **Name:** EdgeMicroTestProduct
    - **Display Name:** EdgeMicroTestProduct
    - **Environment:** test and prod
    - **Access:** Public
    - **Key Approval Type:** Automatic
    - **Resources:**
        - **API Proxy:** Select **edgemicro_hello**
        - **Revision:** 1
        - **Resource Path:** /**
3. Click **Import Resource**.
4. In Resources, click **+API Proxy**
5. Select **edgemicro-auth**
6. Click **Save**.

3. (Optional) Create a test developer

For the purpose of this tutorial, you can use any existing developer for the next step, creating a developer app. But if you wish, create a test developer now:

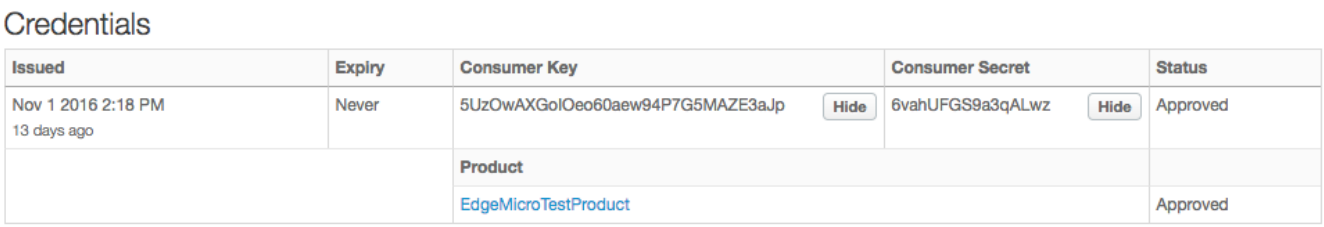
1. Go to **Publish > Developers**.
2. In the Products page, click **+ Developer**.
3. Fill out the dialog to create a test developer.

4. Create a developer app

You are going to use the client credentials from this app to make secure API calls through Edge Microgateway:

1. Go to **Publish > Developer Apps**.

2. In the Developer Apps page, click **+ Developer App**.
3. Fill out the Developer App page as follows:
    a. **Name:** EdgeMicroTestApp
    b. **Display Name:** EdgeMicroTestApp
    c. **Developer:** If you created a test developer, select it. Or, you can use any existing developer for the purpose of this tutorial.
    d. **Credentials:**
        i. Select Expiration: **Never**.
        ii. Click **+ Product** and select **EdgeMicroTestProduct** (the product you just created)
4. Click **Save**.
5. You're back in the Developer Apps list page.
6. Select the app you just created, **EdgeMicroTestApp**.
7. Click **Show** next to the **Consumer Key** and **Consumer Secret**.


Credentials

Issued	Expiry	Consumer Key	Consumer Secret	Status
Nov 1 2016 2:18 PM 13 days ago	Never	5UzOwAXGolOeo60aew94P7G5MAZE3aJp Hide	6vahUFGS9a3qALwz Hide	Approved
		Product		
		EdgeMicroTestProduct		Approved

Note: You'll need to use these keys later when you configure use API Key or OAuth2 access token security for your API.

Part 3: Operate Edge Microgateway

Now that you have a configured Edge Microgateway and at least one Edge Microgateway-aware proxy on Edge, it's time to start up Edge Microgateway. An Edge Microgateway HTTP server will run on your local machine, and you'll make API calls directly to that server.

1. Start Edge Microgateway

Use the **edgemicro start** command to start Edge Microgateway.

1. Be sure you have the keys that were returned previously when you ran the **edgemicro configure** command. That output looked something like this:

   ```
   You need key and secret while starting edgemicro instance

   key: da4778e7c240a5d4585fc559eaba5083328828ac9f3a7f583e8b73e
   secret: 3aad7439708b4aeb38ee08e87189921ad00e6fc1ba8a8ae9f929ee2
   ```

2. (Optional) Print help information for the **`edgemicro start`** command.

   ```
   edgemicro start -h
   ```

3. To start Edge Microgateway, execute the following command:

   ```
   edgemicro start -o [org] -e [env] -k [key] -s [secret]
   ```

   Where:

   - **`org`** is your Edge organization name (you must be an org administrator).
   - **`env`** is an environment in your org (such as test or prod).
   - **`key`** is the key returned previously by the configure command.
   - **`secret`** is the key returned previously by the configure command.

   Example

   ```
   edgemicro start -o docs -e test -k 701e70e718ce6dc1880616b3c39177d64a8
   ```

   Output

   The start command retrieves a lot of configuration information from Apigee Edge (which scrolls into the terminal window). In the output, you'll see a list of microgateway-aware proxies and products that were discovered. At the end of the output, you should see something like this:

   ```
   ...

   PROCESS PID : 9757
   installed plugin from analytics
   ```

```
installed plugin from oauth
eb725020-a2b0-11e6-8a52-6de156e3a6e2 edge micro listening on port 8000
installed plugin from analytics
installed plugin from oauth
installed plugin from analytics
installed plugin from oauth
installed plugin from analytics
installed plugin from oauth
eb77ce60-a2b0-11e6-8a88-b96278c0c198 edge micro listening on port 8000
eb78b8c0-a2b0-11e6-bf36-717b986c91fe edge micro listening on port 8000
eb77f570-a2b0-11e6-883e-472b9104351e edge micro listening on port 8000
```

What happened?

Look at the terminal where you ran the **edgemicro config** command. Scrolling up through the standard output, you can see that the command retrieves a payload of Edge Microgateway configuration information from Apigee Edge. This information includes:

- The public key we created and stored previously in the Apigee vault.
- A JSON representation of all Edge Microgateway-aware proxies that exist in the organization/environment. These are all proxies that are named with the prefix **edgemicro_**.
- A JSON representation of all of the API products that exist in the organization/environment.

With this information, Edge Microgateway knows which proxies and proxy paths it is allowed to process. It uses the product information to enforce security (in exactly the same way as any API proxy does on Apigee Edge, where developer app keys have an association with products). We'll go through the steps to secure Edge Microgateway shortly.

2. Test Edge Microgateway

With Edge Microgateway running, you can call the proxy. The configuration for the **edgemicro_hello** proxy was downloaded from Edge when you started Edge Microgateway. Remember, the proxy basepath is **/hello**.

To test Edge Microgateway, we start with the base path and add a resource path **/echo**. Note that everything after the base path (including any query parameters) is simply passed through to the backend target:

```
curl -i http://localhost:8000/hello/echo
{"error":"missing_authorization","error_description":"Missing Authorization head
```

The error occurs because you did not send a valid API key or access token with the request. By default, Edge Microgateway requires either an API key or an access token on every API call. In the next step of the tutorial, we'll secure this API properly and show you how to obtain a valid access token and include it with the request.

Note: Debugging problems: If you have a problem, you can restart Edge Microgateway in debug mode using the **`--debug`** flag. For details on running in debug mode, see "Debugging and troubleshooting" in Operation and configuration reference for Edge Microgateway (https://docs.apigee.com/api-platform/microgateway/2.3.x/operation-and-configuration-reference-edge-microgateway-v2.3.x.html). You can also take a look at the Edge Microgateway log files when debugging a problem. For details, see "Managing log files in Operation and configuration reference for Edge Microgateway (https://docs.apigee.com/api-platform/microgateway/2.3.x/operation-and-configuration-reference-edge-microgateway-v2.3.x.html).

4. Stop Edge Microgateway

1. In a separate terminal window, `cd` to **the same directory** where you started Edge Microgateway.
2. Enter the stop command:

```
edgemicro stop
```

Note: There are three commands that you must run from the same directory where you started Edge Microgateway. These include **stop**, **reload**, and **status**.

Part 4: Secure Edge Microgateway

You can secure API calls made through Edge Microgateway using an API key or an access token.

- Secure API calls with an OAuth2 access token (#part4secureedgemicrogateway-secureapicallswithanoauth2accesstoken)
- Secure API calls with an API Key (#apikey%20option)

Secure API calls with an OAuth2 access token

Note: Edge Microgateway does not invoke regular Edge API proxies. An access token generated for Edge Microgateway cannot be used to invoke Edge proxies protected by the OAuthV2 policy. Conversely, access token granted to access Edge proxies cannot be used to access APIs called through Edge Microgateway. To use OAuth2 with Edge Microgateway you must follow the specific instructions provided below in this section.

Follow these steps if you want to authenticate API calls with an OAuth2 access token:

1. Get the required keys

1. In the Edge UI, navigate to the Developer App you created previously, as described in Part 2: Create entities on Apigee Edge (#Part2). The name of the App was EdgeMicroTestApp.

★ **Note:** Be sure that you have both the **edgemicro_hello** and **edgemicro-auth** proxies listed in the product associated with the app. Also, make sure the developer you associated with the app is active.

2. In the Developer App page, show the Consumer Key and the Consumer Secret, and copy them. **These values are required to obtain an access token in the next step.**

2. Get an access token

There are two ways to get an access token. We'll show you both methods.

Using the CLI to get an access token

The **first method** is convenient, and follows the pattern we've used throughout the tutorial. The **second method** is generally more useful for client app developers who need to request tokens. The actual token endpoint is implemented in the **edgemicro-auth** proxy that was deployed when you configured Edge Microgateway.

1. (Optional) View help for the **token get** command:

```
edgemicro token get -h
```

2. Generate the token, substituting your Consumer Key and Consumer Secret values from the developer app you created on Apigee Edge in the **-i** and **-s** parameters:

```
edgemicro token get -o [org] -e [env] -i [consumer_key] -s [consumer_s
```

Where:

- **org** is your Edge organization name (you must be an org administrator).
- **env** is an environment in your org (such as test or prod).
- **consumer_id** is the Consumer ID in the Developer App you created previously.
- **consumer_secret** is the Consumer Secret in the Developer App you created previously.

Example

```
edgemicro token get -o docs -e test -i G0IAeU864EtBo99NvUbn6Z4CBwVcS2
```

Output (Sample)

```
current nodejs version is v4.4.4
{ token: 'eyJ0eXAiOiJKV1QiLCJhbGciSUzI1NiJ9.eyJhcHBsaWNhdGl
vbl9uYW1lIjoiNWNiMGY0tOWMzOC00YmJjLWIzNzEtZGMxZTQzOGYxZGMxI
iwiY2xpZW50X2lkIjoiNVVdBWEdvSU9lbzYwYWV3OTRQN0c1TUFaRTNhSnA
iLCJzY29wZXMiOltdLCJhcGlfcHJvjdF9saXN0IjpbIkVkZ2VNaWNyb1Rlc
3RQcm9kdWN0IlmlhdCI6MTQ3ODIwODMzMiwiZXhwIjoxNDc4MjEwMTMxfQ.
v3Q8Rs0W9FO_XpERwIAMMXhjGamRmmmWgrAy1wJv0-99oajx5ASI5qjNubM
nCF14xxwlVXHcz1VfedA8Nx7Ih145zhWWuPKL9muzhXXnVBRFeBMeeLqJk4
QQ7Bhst7xH_68nJSTE20Egu95DHOCUemfrepDFH0VggY4BmzvwLkBNfz_te
E_YqwKZbpGOLMdKK1vMk0fk0x19SieIMS_aJYeOAklrzUhnFxWJFrsAWsax
NLx_BbQcZzYfK1JSDUrhRNVBuPHMnGjNA_oHw2J4qa6Hsp62PfchG-DIW_t
-0BFnYf3rYXmqyNORqFrOMdl4VxqQ' }
```

(Optional) Using the API to get a token

If you're used to calling Edge proxies using curl or another HTTP client, you'll be interested to know that you can call the token endpoint directly, rather than using the **edgemicro token** command. Here's a curl example. Just substitute your org and environment names in the URL,

and pass the colon-separated Consumer Key:Consumer Secret values in a Basic Authentication header:

```
curl -i -X POST --user [client_id]:[client_secret] "http://[org]-[env].apig
```

Where:

- **org** is your Edge organization name (you must be an org administrator).
- **env** is an environment in your org (such as test or prod).
- **client_id** is the Consumer ID in the Developer App you created previously.
- **client_secret** is the Consumer Secret in the Developer App you created previously.

Output (Sample)

The command, whether you used the **edgemicro token** CLI command or called the endpoint using curl, returns a signed access token that can be used to make client calls. Something like this:

```
MIICpDCCAYwCCQDpIvWlpaZJGDANBgkqhkiG9w0BAQFADAUMRIwEAYDVQQDEwls
b2NhbGhvc3QwHhcNMTYwNDA3MTcxNTM5WhcNMTYwND4MTcxNTM5WjAUMRIwEAYD
VQQDEwlsb2NhbGhvc3QwggEiMA0GCSqGSIb3DQEBAUAA4IBDwAwggEKAoIBAQD3
OAQ+kf5FH0S0yuj05ITqUODuUJspBPberRMbqOZYHcFswhB0Yvg6JKWxKWBDP9o
Xl96dtgH7xPFRqIU0zI452jkMQ1fPz2mSaGwik245yfBku7olooXKRKTRKOUoXa
q3Hld/RPxGSsWtiyyYtKex7tuFdq0Knm1EhowdTRGIgjNvudeYMka/XPRXuykhd
xIDxWj4rdX+4GPx9qT2eTQC5nOAC7XHVL7ys4KqsAiv28vw10u400KstFFS8Qho
7FaE0bOKLolKKadKyA60ha1XIw/uSTD6ZQFWQ+XM3OaRbXePWXnSZioSxXcZT7L
hMUKbsRXw/TCvRB51LgNAgMBAAEwDQYJKoZIhvcNAQELBQADgEBAOuR1OmE/W6j
gRAarJB5EQuTEpI/9Zpg5c5RAGjzLhkazsycn7pal+IymUjCV7D0oIxTVuTM8ZZ
57kR5hF/C1ZypDN9i+KGP2ovX8WOCCXYtIQECgZPB/L/7/k7BDuKN4vFBvWUe3X
s2oyjnVWy+ASqsW8gHfj8ekhe22bP240Oqkbyg9SZP9ol1tvJX6+M0thYwcTwAd
ft929Icey/MOTQC0jB2qm0gnIx/7KInFXfS5KoFRJoGrWDeckr3RdBo2LhnPaeZ
1gPYIqphZ3HfisF5KlBXoR8oT/Ilym/nq5COlv+3L4tMIk18F7BQZB60SRazifz
pFkIxepyr/0=
```

3. Check the configuration in Edge Microgateway

1. Open the file **~/.edgemicro/org-env-config.yaml.** See also "Where is Edge Microgateway installed" in Installing Edge Microgateway

(https://docs.apigee.com/api-platform/microgateway/2.3.x/installing-edge-microgateway-v2.3.x.html)

.

2. Make sure these oauth plugin properties are set to **false**. They're false by default, but it's a good idea to double-check:

```
oauth:
   allowNoAuthorization: false
   allowInvalidAuthorization: false
```

3. Also in the **`org-env-config.yaml`** file, be sure that the oauth plugin is added to the **`plugins:sequence`** element, like this::

```
plugins:
   dir: ../plugins
   sequence:
   - oauth
```

4. If you make any changes to the file, **reload** the changes into the running Edge Microgateway instance. This command reconfigures Edge Microgateway with zero-downtime:

```
edgemicro reload -o [org] -e [env] -k [key] -s [secret]
```

Where:

- **`org`** is your Edge organization name (you must be an org administrator).
- **`env`** is an environment in your org (such as test or prod).
- **`key`** is the key returned previously by the configure command.
- **`secret`** is the key returned previously by the configure command.

Example

```
edgemicro reload -o docs -e test -k 701e70ee718ce6dc188016b3c39177d64a
```

4. Call the API securely

With an access token in hand, you can now make the API call securely. For example:

```
curl -i -H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJhcHB
9uYW1lIjoiYmU2YmZjYjAtMWQ0Ni00Y2IxLWFiNGQtZTMxNzRlNTAyMDZkIiwiY2xpZW50X2lkIjoiOG
VhIT1ZwbmhURExhYW9FVG5STVpwWk0iLCJzY29wZXMiOltdLCJhcGlfcHJvZHVjdF9saXN0IjpbIk1pY
eVRlQcm9kdWN0Il0sImCI6MTQzNTM0NzY5MiwiZXhwIjoxNDM1MzQ5NDkxfQ.PN30Y6uK1W1f2ONPEsB
IsjWGfwpz-p6Vak8r767tAT4mQAjuBpQYv7_IU4DxSrnxXQ_q536QYCP4p4YKfBvyqbnW0Rb2CsPFziy
s0p4czcK63SjONaUpxV9DbfGVJ_-WrSdqrqJB5syorD2YYJPSfrCcgKm-LpJc6HCylElFDW8dHuwApaW
4A8Rr-WhTIxDTX7TxkrfI4THgXAo37p3au3_7DPB_Gla5dWTzV4j93xLbXPUbwTHzpaUCFzmPnVuYM44
64RgPmIFUxSqBWGQU7Z1w2qFmWuaDljrMDoLEreI2g" http://localhost:8000/hello/echo
```

The API returns headers and other information from the mock server.

Securing the API with an API key

If you wish to use an API key for authorization, follow these steps:

1. Get the API key

1. In the Edge UI, navigate to the Developer App you created previously, as described in Part 2: Create entities on Apigee Edge (#Part2). The name of the App was EdgeMicroTestApp.

★ **Note:** Be sure that you have both the **edgemicro_hello** and **edgemicro-auth** proxies listed in the product associated with the app. Also, make sure the developer you associated with the app is active.

2. In the Developer App page, show the Consumer Key and copy it. **This value is the API key.** You'll use this key to make authenticated API calls.

2. Check the configuration in Edge Microgateway

1. Open the file `~/.edgemicro/org-env-config.yaml`. See also "Where is Edge Microgateway installed" in Installing Edge Microgateway (https://docs.apigee.com/api-platform/microgateway/2.3.x/installing-edge-microgateway-v2.3.x.html)
.
2. Make sure these oauth plugin properties are set to **false**. They're false by default, but it's a good idea to double-check:

```
oauth:
   allowNoAuthorization: false
   allowInvalidAuthorization: false
```

3. Also in the **`org-env-config.yaml`** file, be sure that the oauth plugin is added to the **`plugins:sequence`** element, like this::

```
plugins:
   dir: ../plugins
   sequence:
   - oauth
```

4. If you make any changes to the file, **reload** the changes into the running Edge Microgateway instance. This command reconfigures Edge Microgateway with zero-downtime:

   `edgemicro reload -o [org] -e [env] -k [key] -s [secret]`

   Where:

   - **`org`** is your Edge organization name (you must be an org administrator).
   - **`env`** is an environment in your org (such as test or prod).
   - **`key`** is the key returned previously by the configure command.
   - **`secret`** is the key returned previously by the configure command.

   Example

   `edgemicro reload -o docs -e test -k 701e70ee718ce6dc188016b3c39177d64a`

3. Call the API securely with an API key

Call the API with the **x-api-key** header as follows. The Consumer Key value you copied from the Developer App is the API key. By default, Edge Microgateway expects you to pass the key in a header called **x-api-key**, like this:

```
curl -i http://localhost:8000/hello/echo -H "x-api-key: [apikey]"
```

Where:

- **`apikey`** is the Consumer Key value taken from EdgeMicroTestApp.

For example:

```
curl -i http://localhost:8000/hello/echo -H 'x-api-key: XsU1R4zGXz2ERxa0ilYQ5szw
```

You now have a fully functioning and secure Edge Microgateway. In the next part of the tutorial, we'll take a look at plugins that add functionality to Edge Microgateway.

Part 5: Add a Spike Arrest plugin

In this part, we'll add a rate-limiting feature called spike arrest to your instance of Edge Microgateway.

What are plugins?

A plugin is a Node.js module that adds functionality to Edge Microgateway. Plugin modules follow a consistent pattern and are stored in a location known to Edge Microgateway, enabling the microgateway to discover and load them automatically. You can read more about plugins in the Use plugins (https://docs.apigee.com/api-platform/microgateway/2.3.x/use-plugins-v2.3.x.html).

Adding the spike arrest plugin

Spike Arrest protects against traffic spikes. It throttles the number of requests processed by an Edge Microgateway instance.

In Edge Microgateway, spike arrest is implemented as a plugin module. To enable it, you need to add it to the Edge Microgateway configuration file.

1. Open the file **~/.edgemicro/org-env-config.yaml.** See also "Where is Edge Microgateway installed" in Installing Edge Microgateway (https://docs.apigee.com/api-platform/microgateway/2.3.x/installing-edge-microgateway-v2.3.x.html)
   .
2. Add the following element. You can add it anywhere in the file.

```
spikearrest:
   timeUnit: minute
   allow: 10
   buffersize: 0
```

3. Add **`spikearrest`** to the **`edgemicro:sequence`** element, as shown below. The sequence configuration property tells Edge Microgateway the order in which the plugin modules are executed.

```
edgemicro:
  home: ../gateway
  port: 8000
  max_connections: -1
  max_connections_hard: -1
  logging:
    level: info
    dir: /var/tmp
    stats_log_interval: 60
  plugins:
    dir: ../plugins
  sequence:
    - spikearrest
    - oauth
```

4. Save the config file.
5. Reload Edge Microgateway with the **reload** command. **You must run this command from the directory where you started Edge Microgateway.**

```
edgemicro reload -o [org] -e [env] -k [key] -s [secret]
```

   Where:

   - **`org`** is your Edge organization name (you must be an org administrator).
   - **`env`** is an environment in your org (such as test or prod).
   - **`key`** is the key returned previously by the configure command.
   - **`secret`** is the key returned previously by the configure command.

   Example

```
edgemicro reload -o docs -e test -k 701e70ee718ce6dc188016b3c39177d64a
```

6. Try calling the API several times in quick succession. After the second call, Edge Microgateway returns this error:

```
{"message":"SpikeArrest engaged","status":503}
```

```
{ "message" : "SpikeArrest engaged", "status" :503}
```

The reason is that spike arrest smooths out the number of calls that can be made over the specified time unit. So, in this case, you can make 10 calls in a minute, or one every 6 seconds.

For more information, see "How does spike arrest work?" in the Use plugins (https://docs.apigee.com/api-platform/microgateway/2.3.x/use-plugins-v2.3.x.html).

Extra credit: Adding the quota plugin

Following the same pattern used to configure spike arrest, you can add other plugins, like the quota plugin. Like with spike arrest, the quota plugin is included with every Edge Microgateway installation. A quota specifies the number of request messages that an app is allowed to submit to an API over a specified time interval (minutes or hours).

To learn how quotas work, see "Using the quota plugin" in Use plugins (https://docs.apigee.com/api-platform/microgateway/2.3.x/use-plugins-v2.3.x.html).

Part 6: Viewing analytics on Apigee Edge

We now have a fully functioning Edge Microgateway instance, let's see what's it's been up to! By default, the analytics plugin module is added to Edge Micro. This module silently pushes analytics data from Edge Micro to Apigee edge, where it is consumed by the Edge Analytics system. Let's see:

1. Log in to your organization on Apigee Edge.
2. Select **Analytics > Proxy Performance**.
3. In the Proxy Performance dashboard, select the **edgemicro_hello proxy**.
4. The graph shows you information about the the proxy's traffic patterns, such as total traffic, average response time, average target response time, and more.

You can read more about Edge Analytics dashboards on the Analytics Dashboards home page in the Edge documentation. To learn more about plugins, see Use plugins (https://docs.apigee.com/api-platform/microgateway/2.3.x/use-plugins-v2.3.x.html)