Red Hat 3scale API Management 2.10

Getting Started

Getting started with your 3scale API Management installation.

Red Hat 3scale API Management 2.10 Getting Started

Getting started with your 3scale API Management installation.

Legal Notice

Abstract

This guide provides information about how you can start working with 3scale.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message .

CHAPTER 1. FIRST STEPS WITH 3SCALE

For your first steps with Red Hat 3scale API Management features, you will learn how to integrate and manage your API, both from customer-facing and internal perspectives.

1.1. PRODUCTS AND BACKENDS FOR 3SCALE APIS

Red Hat 3scale API Management separates your APIs into two main groups:

- Backends: Internal APIs bundled in a product. Backends grant API providers the freedom to map their internal API organization structure to 3scale.

- Products: Customer-facing APIs. Products facilitate the creation of strong and simplified offerings for API consumers.

A product can contain multiple backends, and a backend can be used in multiple products. In other words, to integrate and manage your API in 3scale you need to create both:

- A backend containing at least the URL of your API. The backend can optionally be configured with mapping rules, methods and metrics to facilitate reusability.

- A product for which you define the application plans, and configure APIcast.

1.2. CONFIGURING YOUR FIRST API USING THE 3SCALE WIZARD

The 3scale wizard provides initial assistance when working with products and backends.

IMPORTANT

Run the 3scale wizard only the first time you log in to your 3scale account. If you run the wizard to create subsequent 3scale components, you will overwrite existing components.

Prerequisites

- You need a 3scale account.

Procedure

1. Run the wizard. You can run the 3scale wizard in two ways:

   - The first time you log in to your 3scale account.

   - Running the wizard by replacing **[Your_admin_domain]** in the following URL address: **\https://[Your-admin-domain]-admin.3scale.net/p/admin/onboarding/wizard/intro**

     For example, if your admin domain is *testing-area*, the wizard is available in:

     https://testing-area-admin.3scale.net/p/admin/onboarding/wizard/intro

2. Click **OK, how does 3scale work?**.

3. Watch the introductory animation. Once done, click **Got it! Let's add my API**

4. Create a backend:

a. Specify a name for your backend. For example: **Inventory**

b. Indicate a base URL for the backend. Example: **https://echo-api.3scale.net:443**

c. Click **Add this Backend**.

5. Create a product:

a. Indicate a product name. For example: **Petstore**

b. Click **Add this Product**.

6. Indicate a path to connect your backend to your product, and click **Add the Backend to the Product**.

- You can leave the default value. Path: /

7. To make a test with a GET request, click **Send request**.

- You can specify a GET method.

8. After these steps, you will see a confirmation of a successful request to the API Gateway. For more details about the following configurations, click **Cool, what's next?**.

Once you have taken a look at the suggested additional configurations, you are ready to use 3scale. Click **Got it! Take me to my API on 3scale** and you will see the *Overview* page of your product.

1.3. INITIAL API CONFIGURATIONS TO PERFORM TEST CALLS

Initial configurations will ensure that your API traffic is protected by API keys, tracked, and monitored by 3scale with basic rate limits and controls in place. If this is the first time you are working with 3scale, you can run the wizard to get assistance configuring your first API.

1.4. CREATING BACKENDS FOR YOUR PRODUCTS

A backend is an internal API that is bundled to a product.

Prerequisites

- You need a 3scale account.

Procedure

1. Go to the Dashboard.

2. Under the APIs section, click **Create Backend** in the *Backends* card.

3. Provide the following details:

- *Name*: Backend identifier.

- *System name*: Identifier used for internal purposes.

- *Description*: Optional field containing more details about the backend.

- *Private endpoint*: Base URL of the private API.

4. Click **Create Backend**.

After performing the steps for *Creating backends for your API*, you have an internal API. Add more backends as needed.

Next steps

- Add this backend to a product .

1.5. CREATING NEW PRODUCTS TO TEST API CALLS

As a 3scale API provider, create products to test API calls through these public APIs. A product is a customer-facing API that packages one or more backends.

You can create a new product by following one of these options:

- Define the product manually.

- Import the product from OpenShift.

Here you will find details about the manual definition. If you want to import a product from OpenShift, see Service Discovery.

Prerequisites

- You need a 3scale account.

Procedure

1. Go to the Dashboard. Under the APIs section, click **Create Product** in the *Products* card.

2. Provide the following details:

   a. **Name**: Product identifier.

   b. **System name**: Identifier used for internal purposes. A product **system_name** is used to generate proxy endpoints and domain names. By default, **system_name** is part of a label whose pattern can be as one of the alternatives below:

      - For APIcast staging: **%{system_name}-%{tenant_name}-apicast-staging**

      - For APIcast production: **%{system_name}-%{tenant_name}-apicast-production**

      - When an auto-generated URL label exceeds 63 characters, the system shortens the label as follows: **<truncated-label>-<unique-hash>**

        - **<truncated-label>** is the first 54 or 55 characters of the original URL.

        - **<unique-hash>** is the first 7 characters of a unique SHA-1 hash calculated from the original label.
          For example, this is the URL before truncation:

          https://my-very-long-system-name-also-very-long-tenant-name-apicast-staging.3scale.net

          This is the URL after the truncation:

https://my-very-long-system-name-also-very-long-tenant-name-api-72588d2.3scale.net

c. **Description**: Optional field containing more details about the product.

3. Click **Create Product**.

After these steps, you have a product that represents the public facing API. The next steps are creating backends and adding them to the product .

1.6. ADDING BACKENDS TO YOUR PRODUCTS

By the end of this procedure you will have added a backend to a product. Repeat the procedure to add more backends as you require them.

Prerequisites

- A product. To create one, see Creating new products .

- One or more backends. To create one, see Creating backends for your products .

Procedure

1. Navigate to **[Your_product_name] > Integration > Backends**

2. Click **Add Backend**.

3. From the drop down list, select an existing backend.

4. Specify the routing path in the *Path* textbox. For more details, see The backend path of a specific product.

5. Click **Add to Product**.

After these steps, your product will have a backend. You can follow this procedure again to add more backends to a product or to multiple products.

1.7. THE BACKEND PATH OF A SPECIFIC PRODUCT

When you add a backend to a product, you define the path that the backend uses to communicate with the specified product. This path is not part of the backend.

From the procedure described in Adding backends to your products , APIcast is the API gateway that uses the path of the backend you indicated in step 4. APIcast redirects the traffic to the backend with the specified path matching the prefix of the requested endpoint path.

When defining the path for a backend:

- You can indicate / as the path of one of the backends.

- Paths must be unique inside the product. This means that you cannot add two backends with the same path inside the same product. Neither can you add the same backend twice to the same product.

- You can give the same backend the same path in different products.

This is how the backend path works:

- Each backend added to a product is mounted in the specified path.

- Before redirecting the traffic, the path is removed from the public URL of the request to the product.

Example

Consider this configuration to add a backend to a product:

- Backend: Inventory

- Resource path: **/list**

- Product: Petstore

- Backend path: **/supplies**

These are the URLs used by your configuration:

- Public URL: **<public-api-domain>/supplies/list**

- Private URL: **<private-api-domain>/list**

This is the action flow when a request is sent:

1. The application sends a request.

2. The request is sent to the public URL: **<public-api-domain>/supplies/list**

3. The backend path is removed before redirecting to the private URL: **<private-api-domain>/list**

4. The request is redirected to the **Inventory** backend.

1.8. DEFINING MAPPING RULES

A mapping rule associates a call to an endpoint with designated methods and metrics for tracking and limiting access to your API. You can define mapping rules at the backend and product levels. The advantage of defining mapping rules at the backend level is that you can add a backend to multiple products. To learn more about the metrics or methods for which to collect usage information depending on the requests to your API, both at the product and backend levels, see How APIcast applies mapping rules for capturing usage of 3scale APIs.

Prerequisites

- A backend. To create it, see Creating backends for your products .

Procedure

1. Navigate to the *Backend* you want to define mapping rules for.

2. In the page containing information about the backend, navigate to **Mapping Rules**.

3. Click **Add Mapping Rule**.

4. Specify the following settings:

- *Verb*: The HTTP request verb (**GET**, **POST**, **DELETE**, or **PUT**).

- *Pattern*: The pattern to match. For example, **/hello**.

- *Metric or method to increment* : The metric or method name.

- *Increment by*: The metric increment number. For example, **1**.

- *Last?*: If this mapping rule should be considered as the last one, to stop processing other mapping rules.

- *Position*: Number that indicates the position of the execution of the mapping rule, to sort the mapping rules.

5. Click **Create Mapping Rule**.

Next steps

After these steps, the mapping rule is added to **Backends** under **[Your_API_backend] > Mapping Rules**. The mapping rule is also available for each product currently using the backend. To have the mapping rule active at the product level, promote the latest configuration under the **Products** tab in **[Your_product_name] > Integration > Configuration**

Example

After you promote the configuration, 3scale activates the backend mapping rules at the product level. The mapping rules follow the backend path specified in the product. For example, suppose you have this configuration:

- Pattern of the mapping rule at the backend: **/thousands**

- Backend is added to a product with path: **/unitprice**

The mapping rule at the product level is: **/unitprice/thousands**.

1.9. CREATING 3SCALE APPLICATION PLANS FOR PRODUCTS

A 3scale application plan defines the rules such as limits, pricing, and features for using your API product. For more information, refer to Application plans and Designating methods and adding metrics for capturing usage details.

Prerequisites

- A product. To create one, see Creating new products.

Procedure

1. Navigate to **[Your_product_name] > Applications > Application Plans**

2. Click **Create Application Plan**.

3. On the **Create Application Plan** page, enter a name and a system name for your new plan. A system name must be unique in your 3scale installation.

4. Click **Create Application Plan**.

1.10. CREATING APPLICATIONS FOR THE DEFAULT ACCOUNT TO TEST API CALLS

As a 3scale user, create applications for the default *Developer* account. An application subscribes to an application plan. As a result of this subscription, 3scale provides the user key required to call an API product.

An application is always associated with an application plan. Applications are stored within developer accounts. In basic 3scale plans only a single application is allowed. In enterprise plans, multiple applications per account are allowed.

Prerequisites

- An application plan. To create one, see Creating 3scale application plans for products .

Procedure

1. Navigate to **Audience > Accounts > Listing**

2. Click the default *Developer* account.

3. Go to the **Application** tab.

4. Click **Create Application**.

5. Choose an application plan.

6. **Service plan** contains the product plan configured by the Account Manager, which will be associated to your application.

7. Specify an application name.

8. In the *Description* field, enter information about this application.

9. Click **Create Application**.

You can see your new application in **Dashboard > Audience > Accounts > Applications > Listing**

1.11. SENDING REQUESTS TO YOUR PRODUCT TO TEST THE INTEGRATION OF A BACKEND

As a 3scale API provider, you can send a command line request to a product to test the integration of a backend based on the first mapping rule added to the product.

Before you can send a test request, you must promote an APIcast configuration that includes the backend that you want to test. A specific APIcast configuration consists of the backends you have added to a product with the corresponding mapping rules, applications, and application plans.

3scale directs requests to the backend of a product according to the path specified in the request call. For each backend of a product, you configured the backend path when you added the backend to the product. In other words, each backend has its own path.

Prerequisites

- One or more backends that you added to a product .

- A mapping rule for each backend in a product.

- An application plan to define the access policies.

- An application that subscribes to the application plan.

Procedure

1. Promote the new APIcast configuration to staging:

   a. Navigate to **[Your_product_name] > Integration > Configuration**

   b. Under *APIcast Configuration*, click **Promote v.[n] to Staging APIcast**

      - **v.[n]** indicates the version number to be promoted.

      - If there are no changes to be promoted, there is a grayed button with the text **Nothing to promote**.

2. Under *Staging APIcast*, promote the APIcast configuration to production by clicking **Promote v.[n] to Production APIcast**.

   - **v.[n]** indicates the version number to be promoted.

   - If there are no changes to be promoted, you will see a grayed button with the text **Nothing to promote**.

3. To test requests to your API product, copy the command provided in *Example curl for testing* and run it in a terminal.

   - The **curl** command example is based on the first mapping rule of the product.

   - After you run the command, you should get an HTML response containing results from the backend you are testing.

   - If you do not get a response, delete the catch-all mapping rule from your product, promote the new APIcast configuration to staging and then to production, and run the example **curl** command.

Next steps

You can confirm the different responses when changing metrics and methods, such as limits and pricing rules. For any of the application plans of a product, modify the methods and metrics when testing requests to your product. For more details, refer to adding methods and metrics.

Every time you modify the product configuration and before making calls to your API, you must promote the updated configuration to the staging and production environments. When there are pending changes to be promoted to the staging environment, there is an exclamation mark in the Admin Portal, next to the **Integration** menu item.

1.12. ADDITIONAL RESOURCES

- APIs as a Product: Get the value out of your APIs

- APIs as a Product: Get started in no time

CHAPTER 2. LAUNCHING YOUR APIS

In this chapter, you will learn about some key steps to launch your APIs with Red Hat 3scale API Management. To use this guide, there are these assumptions:

- 3scale has an internal API called **API Backend**, which is a backend.

- 3scale has a customer-facing API called **Echo API**, which is a product. This is the API that you will expose through the Developer Portal.

The guide covers the following steps to launch your API product:

1. Secure the product.

2. Configure the product access policies with application plans.

3. Engage your developers with a Developer Portal.

4. Go live.

2.1. PATHS TO LAUNCH YOUR FIRST API

To begin working with 3scale, you can choose one of the three paths to launch and expose your API product. Your *API product* is the customer-facing API you are showing to the world.

The timing guidelines depend on the complexity of your API product and the resources you plan to dedicate to the effort. You will spend most of your time on refining your API product and preparing content for your Developer Portal. If you already have a stable product and content for documentation, you can go live within a week.

These are the paths to launch your first API product:

Prototype

- Goal: Complete an end-to-end integration of 3scale with a simple API product, which will be exposed to the public.

- Recommended for: This path helps you get a general overview of the end-to-end capabilities of 3scale. You must do this path before going through Basic. If you have successfully completed the onboarding wizard in the Admin Portal, you can skip this path and go to the next one.

- Completion time: Less than an hour.

Basic

- Goal: Complete all implementation steps to launch your API product in production.

- Recommended for: If you want to go live with your API product in production and you have limited time, the Basic path will cover most of your needs.

- Completion time: Less than a week.

Advanced

- Goal: Optional extras after you have completed the Basic path, such as advanced control of your API product, and deeper customization of the Developer Portal.

- Recommended for: If you have a more complex requirement or if you have covered the Basic path already, you may be ready to consider advanced options.

- Completion time: Several weeks.

2.2. FOLLOWING THE PROTOTYPE PATH

You can follow through the Prototype path individually from end to end. Alternatively, you can choose to perform some steps from this path according to your needs. Each path can be independent, but Prototype, Basic, and Advanced paths build on top of each other.

2.2.1. Securing your API

You can prototype the 3scale access control layer within a few minutes, assuming one of the following cases:

- In 3scale Hosted (SaaS), your product is publicly accessible.

- In 3scale on-premises, your product is reachable from the 3scale installation.

**Echo API** serves as an example of a public product. It has the following features:

- It is a simple API that accepts any path and returns information about the request (path, request parameters, headers, etc.) in the response body.

- It is accessible at the following URL: https://echo-api.3scale.net

- The first time you activate 3scale, a product is created for each existing API. In this first time, there is a one-on-one relationship between the product and the API backend. In other words, you will see: **Echo API**, a product containing **API Backend**.

To secure your **Echo API** product, follow these steps:

1. Verify that your product is reachable. Example: https://echo-api.3scale.net/v1/fast/track

   - After the security layer is in place, you can hide or restrict access to the backend host.

2. Navigate to **[Your_product_name] > Integration > Configuration**

3. For **[Your_product_name]**, confirm that the private endpoint has been set in the default API Backend. Example: **https://echo-api.3scale.net:443**

4. Click the button to promote to staging.

5. Copy the cURL statement, which includes the **user_key** as the default credential, to make calls from the command line:

   ```
   curl "https://api-2445581407825.staging.apicast.io:443/v1/fast/track?
   user_key=287d64924e6120d215b1000ac07c063b"
   ```

   You can make different calls. For example try another endpoint, adding the same **user_key**.
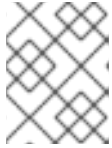
> **NOTE**
>
> You can get the API product keys from the application details page, located in one of the developer accounts.

Your 3scale access control layer will now only allow authenticated calls through to your backend API.

2.2.2. Configuring API access policies with application plans

After following the steps under Section 2.2.1, "Securing your API", only authenticated calls are allowed through to your APIs. In this section you will apply policies to differentiate the rate limits.

In 3scale, *applications* define the credentials to access your product. An application is always associated with one *application plan* that determines the access policies. Applications are stored within *developer accounts*. In the basic 3scale plans only a single application is allowed; but, in the higher plans, multiple applications per account are allowed.

In this example, you add a policy to the **Echo API** product used in the preceding section with these steps:

1. Navigate to **[Your_product_name] > Applications > Application Plans**

2. In the **Application plans** section, go to the *Basic* application plan to edit one of the plans that was generated by the sample data after installing or signing up for 3scale.

3. Under **Metrics, Methods, Limits & Pricing Rules** select *limits* in the *hits* row, and create a new usage limit of 3 per hour.

4. Find one of your sample applications, by navigating to **[Your_product_name] > Applications > Listing**. Ensure that the application is set to the *basic* plan. If not, *change plan* on the application details page.

5. Use the credentials for this application and repeat the previous sample call at least three times.

You have now successfully defined more restrictive access polices for all the applications on 3scale Basic plan.

2.2.3. Engaging developers with a Developer Portal

For the Prototype path, you do not need to create any documentation content. It is usually enough to check that the workflows meet your requirements.

While the product is in development and testing, you can disable the full self-service workflow with these steps:

1. From your Admin Portal, navigate to the **Audience** space and click **Visit Portal** link in the **Developer Portal** menu.

2. Create a test signup and walk through all the steps.

3. Usually self-service is enabled by default. To change it, go to **Audience > Accounts > Usage Rules** and select the *account approval required* checkbox.

4. Repeat the test signup walkthrough and verify that you need to approve the account in the Admin Portal before the user can log in.

You can now successfully customize workflows for your Developer Portal.

2.3. FOLLOWING THE BASIC PATH

You can follow through the Basic path individually from end to end. Alternatively, you can choose to perform some steps from this path according to your needs. Each path can be independent, but Prototype, Basic, and Advanced paths build on top of each other.

2.3.1. Securing your API

For a full production implementation, you need to make some fundamental decisions about how to structure your product and implement integration with 3scale.

You have the choice of several authentication modes for product traffic. Consult the guide on the available options and configure the settings.

> **IMPORTANT**
>
> After you set the authentication, you should not switch authentication modes because this action might invalidate existing credentials.

You also have several deployment options for the API traffic manager layer . APIcast, the NGINX based API gateway, is the favorite amongst 3scale customers due to its combination of ease of configuration and performance. You can use hosted APIcast for a quick start, but comes with volume limits and additional latency. Alternatively, you can deploy it on your own servers for the best performance and completely unrestricted traffic volume.

Hosted APIcast

1. Follow the onboarding wizard after you log in to your Admin Portal for the first time.

2. Continue iterating on your product configuration, such as refining access policies, until you have reached a version suitable for production.

3. Promote your APIcast configuration to the production gateway.

Self-managed APIcast

1. Set up a test installation of your API gateway on your OpenShift servers.

2. Continue iterating on your API configuration, such as refining access policies until you have reached a version suitable for production.

3. Promote your APIcast configuration to the production gateway.

4. For further details about self-managed APIcast, see Installing APIcast. Additionally, APIcast policies covers some concepts to configure the API access policies

2.3.2. Configuring API access policies with application plans

After following the steps under Section 2.3.1, "Securing your API", only authenticated calls are allowed through to your product. In this section you will apply policies to differentiate rate limits.

In 3scale, *applications* define the credentials to access your API product. An application is always associated with one *application plan* that determines the access policies. Applications are stored within

developer accounts. In the basic 3scale plans, only a single application is allowed. In the higher plans, multiple applications per account are allowed.

In *Prototype*, you can only control access based on overall hits on your product. The flexibility of 3scale is evident after you start using custom methods and metrics to create more sophisticated tiers for your application plans and for deeper analytic insight to your product. For more details, see the analytics guide.
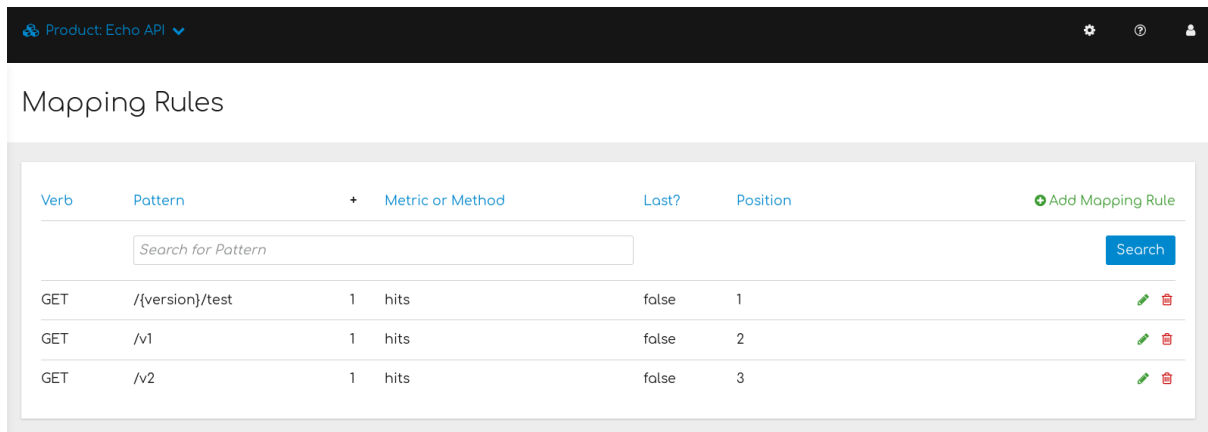
> **IMPORTANT**
>
> - The mapping between your API structure and methods or metrics in 3scale is logical. You can get reports of the product usage from 3scale if you define a consistent rule. You must determine the level of detail. Generally, you should aim between 5 and 20 methods/metrics.
>
> - The values reported to 3scale can only be incremented. You cannot set absolute values or decrease the counters.
>
> - After adding any new methods or metrics to 3scale, it is important to add the new system names to your integration point, such as the API gateway or code plugin.
>
> - You can make changes, such as rate limits, at runtime without redeploying.

In this example, to add polices to the application plan of the **Echo API** product, perfom these steps:

1. Find the product you want to work on.

2. In the **Application Plans** section, select *basic* to edit one of the plans that was generated automatically after signing up to 3scale/deploying your instance.

3. If you find a rate limit for *hits*, remove it.

4. Add a *new method* to the plan under the *hits* metric with the system name "test".

5. Set a rate limit for the test method to 5 per hour.

6. Add two *new metrics* with system names **v1** and **v2**.

7. Under the v2 metric, disable access by clicking on the *enabled* column. This has the same effect as setting a rate limit of zero.

APIcast deployment

1. Go to **[Your_product_name] > Integration > Configuration**

2. Expand the mapping rules section and add the following mappings:

NOTE

The default mapping for "/" has been removed. If you use this default mapping, it will lead to double-counting of hits.

Code plugin deployment

1. To add usage for custom methods and metrics to your 3scale authorization and reporting calls, follow the instructions and examples in your plugin library.

2. Ensure a mapping from the URL structure to the custom method, **test**.

3. Ensure a mapping from the URL to the custom metrics **v1** and **v2**.

4. Test the calls using application credentials associated with the 3scale Basic plan.

   - Calls will be allowed:

     ```
     curl "https://api-2445581407825.staging.apicast.io:443/v1/test?
     user_key=287d64924e6120d215b1000ac07c063b"
     ```

     After 5 calls, the calls will start to get rejected. This is because of the limit set for the test method.

   - Calls will be rejected because **v2** is not allowed in the 3scale Basic plan:

     ```
     curl "https://api-2445581407825.staging.apicast.io:443/v2/test?
     user_key=287d64924e6120d215b1000ac07c063b"
     ```

   - Calls will be rejected because there is no mapping rule set for missing:

     ```
     curl "https://api-2445581407825.staging.apicast.io:443/missing?
     user_key=287d64924e6120d215b1000ac07c063b"
     ```

   - These calls will be allowed for NGINX depending on how you have implemented the mapping for your plugin. For the following call, it will be up to your application to return a 404 not found response. To avoid this, refine the mapping:

     ```
     curl "https://api-2445581407825.staging.apicast.io:443/noversion/test?
     user_key=287d64924e6120d215b1000ac07c063b"
     ```

This basic concept provides the flexibility you need to define your API tiers, both at the product and backend levels. You should decide early on what you want to use for your custom methods and metrics. Whenever you make changes to the system names, redeploy the changes as described in Section 2.3.1, "Securing your API" for the Basic path.

2.3.3. Engaging developers with a Developer Portal

You can find information to create and provide APIs using the Developer Portal. Consider writing your content in Textile or Markdown. Following are optional steps that you may want to consider:

- Configure ActiveDocs to bring interactive capabilities to your documentation and make it easier for developers to explore.

- Add a favicon.

- Add your Google Analytics tracker code by editing *partial* in your CMS called *analytics*.

- Configure your signup workflows.

- Customize your email addresses (document for 3scale Hosted -SaaS-) and the email template content.

2.4. FOLLOWING THE ADVANCED PATH

You can follow through the Advanced path individually from end to end. Alternatively, you can choose to perform some steps from this path according to your needs. Each path can be independent, but Prototype, Basic, and Advanced paths build on top of each other.

2.4.1. Securing your API

To secure your product, you have the following alternatives:

Advanced authentication mode: OpenID Connect (OIDC)

Secure your products using the APIcast integration with OpenID Connect for Red Hat Single Sign-On (RH-SSO). Applications in 3scale are synchronized with the Identity Provider (IdP), in this case RH-SSO. Currently, this is an end-to-end supported solution. It covers the main OAuth 2.0 flows:

- Authorization code

- Resource owner password

- Client credentials

- Implicit grant

Code plugin deployment

If you want to increase the performance of your product, you can cache authorization calls to 3scale using any caching library.

2.4.2. Configuring API access policies with application plans

With the options listed under Section 2.4.1, "Securing your API", you ensured that only authenticated calls are allowed through to your product. In this section, you apply policies to differentiate rate limits.

In 3scale, *applications* define the credentials to access your product. An application is always associated with one *application plan* that determines the access policies. Applications are stored within *developer accounts*. In the Basic 3scale plans, only a single application is allowed. In the higher plans, multiple applications per account are allowed.

Alerts may be configured to send notifications by email or to the web consoles:

1. Go to your API Settings page: **[Your_product_name] > Integration > Settings**.

2. Go to the *Alerts* section on the page. Here, you can configure the alerts that you want as a percentage of your rate limit levels.

3scale gives you the flexibility to decide on the level of rate limits:

- Soft rate limits: Even calls above the limits are allowed through.

- Hard rate limits: Calls are rejected before hitting your application.

With the code plugin, you need to decide which type to implement. On the other hand, APIcast defines hard limits by default. These can be customized in the Lua file to avoid rejecting over-limit calls.

2.4.3. Engaging developers with a Developer Portal

After you have completed the Basic path, following are two advanced areas you can explore for the Developer Portal:

- Liquid markup provides tags and drops that provide direct access to system objects and allow you to introduce dynamic rendering of developer portal pages.

- All 3scale system pages can be customized. This is for advanced users because the HTML is complex. Ultimately, you can customize virtually any page of your Developer Portal. Usually the default pages will be perfectly fine with some CSS changes.

2.5. GOING LIVE

This section describes the final checklist before the public launch of your API product.

> **NOTE**
>
> Raise request for the custom domain and email as soon as possible because they have a long lead time.

1. Set up a custom domain. For further details, refer to Custom Domain in 3scale Hosted (SaaS).

2. Optionally, set up a custom outbound email address. To see how to perform this step, see Configure Email Domain.

3. Remove the Developer Portal access code from **Audience > Developer Portal > Domains & Access**.

Following are some extra points for consideration:

- Add pricing to generate earnings directly from your API product. This feature is only available for 3scale Hosted (SaaS) accounts.

- Use insight from your product analytics, in your Admin Portal located under *Analytics*, to refine your application plans.