# Red Hat 3scale 2.2

## Quickstart

For Use with Red Hat 3scale 2.2

Last Updated: 2018-09-20

# Red Hat 3scale 2.2 Quickstart

For Use with Red Hat 3scale 2.2

## Legal Notice



## Abstract

This guide provides the basic information to help users get started with Red Hat 3scale 2.2.

# Table of Contents

# CHAPTER 1. LAUNCHING YOUR API PROGRAM

This guide will help you get up and running to boost your API with 3scale in no time at all.

It will cover all the key steps to launch your API:

1. **Secure** your API
2. **Configure** your API access policies with application plans
3. **Engage** your developers with a Developer Portal
4. **Go Live**

You have a choice of three different paths to take. These are:

- Prototype. **Completion time**: Less than an hour. **Goal**: Complete an end-to-end integration of 3scale with a simple public API. **Recommended for**: The prototype path is recommended to get the fastest possible overview of how to integrate 3scale and to get an appreciation of 3scale's end-to-end capabilities. Do this first before the basic path. If you've successfully completed the onboarding wizard in the Admin Portal, you can skip this path and go straight to the next one.
- Basic. **Completion time**: Less than one week. **Goal**: Complete all implementation steps to launch your API in production. **Recommended for**: If you want to go live with your API in production and time is of the essence, the basic path will cover most of your needs.
- Advanced. **Completion time**: Several weeks. **Goal**: Optional extras for after you've completed the basic path. Advanced control of your API. Deeper customization of the Developer Portal. **Recommended for**: If you have a more complex requirement, or if you've covered the basic path already, you may be ready to consider advanced options. If you want to use a custom domain and email (in SaaS), they have a long lead time, so you should follow the steps in the go live section to get them set up as soon as possible.

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

## 1.1. HOW TO SECURE, CONTROL, AND PROMOTE YOUR APIS

You can follow through the **prototype**, **basic**, and **advanced** paths individually from end to end, or you can also choose bits and pieces from the three different paths according to your needs. Each path is independent, but they build on top of each other.

### 1.1.1. Prototype

#### 1.1.1.1. Secure your API

Assuming your API is publicly accessible (for Saas) or reachable from your 3scale AMP installation (for on-premises), you can prototype the 3scale access control layer within a few minutes.

The Echo API will serve as en example of a public API. 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

1. Verify that your API is reachable (once the security layer is in place, you can hide or restrict access to the backend host) e.g. https://echo-api.3scale.net/v1/fast/track
2. Go to the API integration wizard: **Dashboard > API > Integration**
3. Before you set up your own API, verify that you can make a test call using the default parameters.
4. Once verified, proceed to enter the private base URL e.g. **`https://echo-api.3scale.net:443`**.
5. Enter the URL path for a valid GET request to make a test call, e.g. **`/v1/fast/track`** , and click **Update & Test Staging Environment**.
6. Now you can copy the cURL statement, which includes **`user_key`** as the default credentials to make calls from the command line:

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

   You can go ahead and play around making different calls. For example try another endpoint, adding the same **`user_key`**.
7. Note: You can get the API keys from the application details page of one of the developer accounts.
8. Success! Your 3scale access control layer will now only allow authenticated calls through to your backend API.

#### 1.1.1.2. Configure your API access policies with application plans

In the previous step, you ensured that only authenticated calls are allowed through to your API. Now you will apply policies to differentiate rate limits.

In 3scale terms, *applications* define the credentials to access your API. An application is always associated with one *application plan*, which 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'll add a policy to the Echo API used in the previous step.

1. Go to your API configuration: **API > Integration**
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. Select *limits* on the *hits* row, and create a new usage limit of 3 per hour.
4. Go to one of your sample applications in **API > Applications**, and make sure that the application is set to the *basic* plan. If not, *change plan* on the application details page.
5. Use the credentials for this app and repeat the previous sample call at least 3 times.
6. Success! You have defined more restrictive access polices for all the applications on the basic plan.

#### 1.1.1.3. Engage your developers with a developer portal

For the prototype, you don't really need to do create any documentation content. It's usually enough to check that the workflows will meet your requirements. For example, while the API is in development and testing, you may want to disable the full self-service workflow:

1. From your Admin Portal, go to the view of your Developer Portal seen by your developers by clicking *Visit developer portal* link.
2. Create a test signup and walk through all the steps.
3. Usually self-service is enabled by default. To change it, go to **Settings > General** and check the box for *account approval required*.
4. Now you can repeat the test signup walkthrough and verify that you need to approve the account in the Admin Portal before the user can log in.
5. Success! You can customize workflows for your developer portal.

### 1.1.2. Basic

#### 1.1.2.1. Secure your API

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

You have the choice of several authentication modes for API traffic. Consult the guide on the available options and configure the settings. Once you set this, you should not switch auth modes again because it can easily invalidate existing credentials.

You also have the choice of 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 APIcast hosted – great to get started quickly, but comes with volume limits and additional latency – or on deploy it on your own servers – for the best performance and completely unrestricted traffic volume. If you prefer to stick with your existing app stack technologies and languages, or if you want tighter control over the integration, the plugin libraries will be a good fit.

**APIcast hosted**

1. Simply follow the onboarding wizard after you log in your Admin Portal for the first time.
2. Continue iterating on your API configuration (such as refining access policies) until you've reached a version you're happy with for production.
3. Promote your APIcast configuration to the production gateway.

**APIcast self-managed**

1. You can follow the guide for APIcast for more details than in the *prototype* steps. This guide also covers some of the ground to configure API access policies.
2. Set up a test installation of your API gateway on your OpenShift servers.
3. Continue iterating on your API configuration (such as refining access policies) until you've reached a version you're happy with for production.
4. Promote your APIcast configuration to the production gateway.

**Code plugin deployment**

1. You can follow the tutorial on using 3scale plugins to set up the integration.
2. Choose the appropriate library for your language.
3. Sample code snippets to make authorization calls are available directly in your Admin Portal under **API > Settings > Integration**.
4. Whenever you need to do some troubleshooting, it's best to start by comparing the plugin behavior to the raw REST API calls using ActiveDocs. Those can be found on the top level corner of your Admin Portal.

*Note: 3scale code plugins are community supported.*

#### 1.1.2.1.1. Configure your API access policies with application plans

In the previous step, you ensured that only authenticated calls are allowed through to your API. Now you will apply policies to differentiate rate limits.

In 3scale terms, *applications* define the credentials to access your API. An application is always associated with one *application plan*, which 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 the *prototype*, you only controlled access based on overall hits on your API. You realize the real flexibilty of 3scale once you start using custom methods and metrics to create more sophisticated tiers for your application plans and for deeper analytic insight to your API. For some brief background, check out the analytics guide. There are a few tips worth bearing in mind:

- The mapping between your API structure and methods or metrics in 3scale is a logical one. As long as you can define a consistent rule, you can report the usage to 3scale. In effect, you determine the level of detail. Generally, it's good to aim for 5-20 methods/metrics.
- The values reported to 3scale can only be incremented. You cannot set absolute values or decrement the counters.
- After adding any new methods or metrics to 3scale, it's important to add the new system names to your integration point (API gateway or code plugin).
- Any changes such as rate limits can be done on the fly without needing to redeploy.

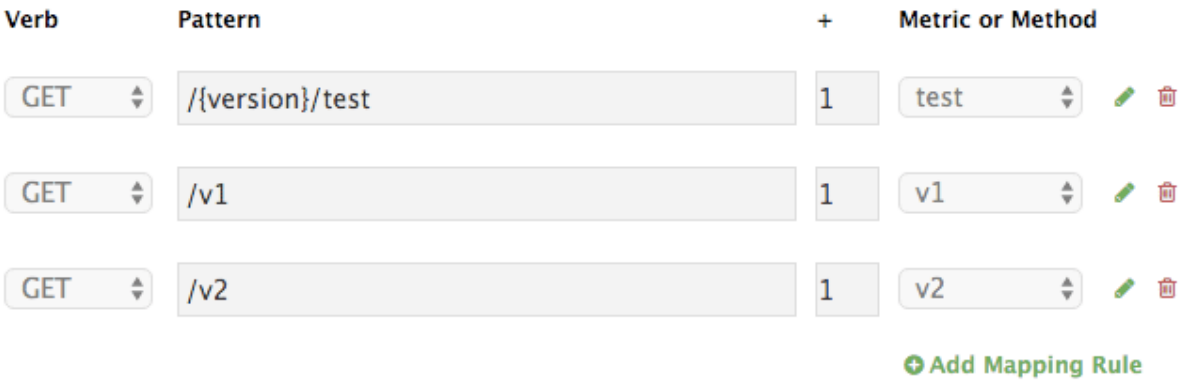
In this example, add polices to the application plan of the Echo API:

1. Go to your *APIs* tab where you can configure all your services.
2. In the 'Application Plans' sectio, select *basic* to edit one of the plans that was generated automatically after signing up to 3scale / deploying your instance.
3. If you have 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 of 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 the the integration wizard in **API > Integration**.
2. Expand the mapping rules section, and add the following mappings:




3. Note: the default mapping for "/" has been removed. If it had been left in place, it would lead to double-counting of hits.

**Code plugin deployment**

1. Follow the instructions and examples in your plugin library to add usage for custom methods and metrics to your 3scale authorization and reporting calls.
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". Test the calls using application credentials associated with the 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 be rejected. That's because of the limit set for the test method.
   - Calls will be rejected because v2 is not allowed in the *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 (for your plugin, it depends how you implement the mapping) for the following call – it would be up to your app to return a 404 not found response. You can always refine the mapping to avoid this:

     ```
     ``curl "https://api-
     ```

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

This basic concept gives you all the flexibility you need to define your API tiers. The important thing is to decide early on what you want to use for your custom methods and metrics. Whenever you make changes to the system names, you need to redeploy the changes as described in the *secure your API* section.

#### 1.1.2.2. Engage your developers with a developer portal

The developer portal guide contains almost all you need to complete a developer portal. Consider writing your content in Textile or Markdown. Some optional steps 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 the *partial* in your CMS called *analytics*.
- Configure your signup workflows.
- Customize your email addresses (doc for SaaS) and the index:https://access.redhat.com/documentation/en-us/red_hat_3scale/2.2/html-single/developer_portal#customize-email[email template content].

### 1.1.3. Advanced

#### 1.1.3.1. Secure your API

**APIcast self-managed**

- The combination of NGINX and Lua provides unlimited API tranformation options. For some inspiration, check out the 3scale developer blog post on how to augment your API.

**Advanced authentication mode: OIDC**

- Secure your APIs using the APIcast integration with OIDC for Red Hat Single Sign-On (RHSSO). Applications in the Red Hat 3scale API Management Platform are synchronized with the Identity Provider (IdP), in this case RHSSO. This solution is currently supported end to end and covers the main OAuth 2.0 flows: Authorization code, Resource password owner, Client credentials, and Implicit grant.

**Code plugin deployment**

- Almost all 3scale customers find performance to be fine. But if you really want to turbo-charge your API, you can cache authorization calls to 3scale using any caching library you're comfortable with.

#### 1.1.3.2. Configure your API access policies with application plans

In the previous step, you ensured that only authenticated calls are allowed through to your API. Now you will apply policies to differentiate rate limits.

In 3scale terms, *applications* define the credentials to access your API. An application is always

associated with one *application plan*, which 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.

Alerts may be configured to send notifications out by email or to the web consoles. Go to your API configuration page **APIs > [Service]** Then go to *settings* and select *alerts*. There, you can configure the alerts you desire as a percentage of your rate limit levels.

3scale gives you the flexibility to decide whether to make rate limits soft (even calls above the limits are allowed through) or hard (calls are rejected before hitting your app). With the code plugin, you consciously 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.

#### 1.1.3.3. Engage your developers with a developer portal

Two advanced areas to explore for the developer portal (after you've completed the basic path):

- 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 a topic for very advanced users because the HTML is rather complex. Ultimately, you can customize virtually any page of your developer portal. Usually the default pages will be perfectly fine with some CSS changes.

## 1.2. GO LIVE

The final checklist before the public launch of your API – remember to request the custom domain and email as soon as possible since they have a long lead time:

1. Set up a custom domain* (SaaS documentation)
2. Set up a custom outbound email address* (SaaS documentation) (*)
3. Remove the developer portal access code, which can be found in **Settings > Developer portal > Domains * Access**.

(*) These steps are optional.

Beyond this point, some extras that you may want to consider:

- Add pricing to generate revenue directly from your API (only available for SaaS accounts).
- Use insight from your API analytics (under *analytics* in your Admin Portal) to refine your application plans.