PAL Caching Solution

User Manual

Table of Contents

[**1** **Introduction:** 4](#_Toc160109989)

[**1.1** **Purpose** 4](#_Toc160109990)

[**1.2** **Scope:** 4](#_Toc160109991)

[**2** **Solution Architecture** 4](#_Toc160109992)

[**3** **Key Components** 4](#_Toc160109993)

[**3.1** **Google Kubernetes Engine (GKE)** 4](#_Toc160109994)

[**3.2** **Unleash** 4](#_Toc160109995)

[**3.3** **Caching Solution App** 5](#_Toc160109996)

[**3.4** **Apache Ignite** 5](#_Toc160109997)

[**3.5** **Redis** 5](#_Toc160109998)

[**4** **Deployment** 5](#_Toc160109999)

[**4.1** **Build and push the image to Artifact Registry** 5](#_Toc160110000)

[**4.2** **Deploy the components using YAML files** 5](#_Toc160110001)

[**4.2.1** **Apache Ignite** 5](#_Toc160110002)

[**4.2.2** **Redis** 6](#_Toc160110003)

[**4.2.3** **Postgres** 6](#_Toc160110004)

[**4.2.4** **Unleash web** 6](#_Toc160110005)

[**4.2.5** **Spring boot app** 6](#_Toc160110006)

[**5** **Testing** 6](#_Toc160110007)

[**5.1** **Login to Unleash** 6](#_Toc160110008)

[**5.2** **Token generation** 7](#_Toc160110009)

[**5.3** **Create new feature toggle** 9](#_Toc160110010)

[**5.4** **Application testing using Swagger** 11](#_Toc160110011)

**Revision History**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SNo** | **Version** | **Modification Date** | **Author** | **Reviewer** | **Comments** |
| 1 | V1.0 | 29-02-2024 | Vineeth Roy/Praval Kumar | Shambhu Kumar |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

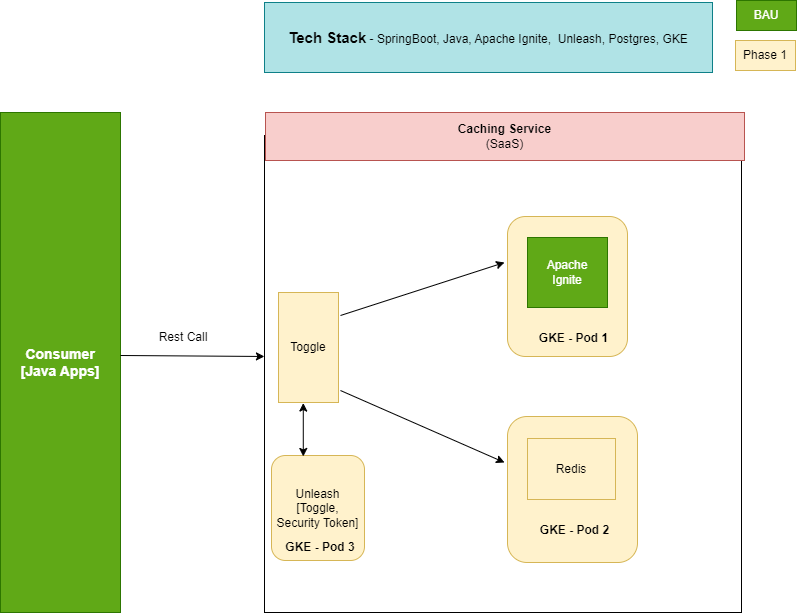
# **Introduction:**

## **Purpose**

## 

## **Scope:**

# **Solution Architecture**



# **Key Components**

## **Google Kubernetes Engine (GKE)**

GKE is Google-managed implementation of the Kubernetes open source container orchestration platform. GKE is the main component of the caching Solution as all the components including Sprint boot application, unleash web, Apache Ignite and Redis are containerized and deployed in GKE. All those components are exposed as a service as well. The Service API, part of Kubernetes, is an abstraction to help to expose groups of Pods over a network. Each Service object defines a logical set of endpoints (usually these endpoints are Pods) along with a policy about how to make those pods accessible. GKE is ideal if a platform is needed that helps to configure the infrastructure that runs the containerized apps, such as networking, scaling, hardware, and security. The main advantages of using GKE is platform management, improved security, cost optimization, reliability, and availability.

## **Unleash**

Unleash is an open source solution for feature management. It streamlines the development workflow, accelerates software delivery, and helps teams to control how and when they roll out new features to end users. Feature flags in Unleash enables the team to work on multiple features simultaneously without the need for separate feature branches. In caching solution, Unleash helps to toggle the underlying cache without rebuilding the application. One cache is enabled at a time and Ignite will be enabled as the default one.

## **Caching Solution App**

The spring boot caching solution application is build using Spring boot, containerized, and deployed in GKE. The application has four rest end points. One each for test, get, put and delete.

## **Apache Ignite**

Ignite is an open-source, in-memory, key-value and distributed computing platform that includes an in-memory data grid (IMDG), in-memory database (IMDB), support for streaming analytics, and a continuous learning framework for machine and deep learning. The main advantages of using Ignite is it provides in-memory speed and unlimited horizontal scalability. The caching solution is integrated with two storages currently – Apache Ignite and Redis. They are deployed in GKE as separate containers.

## **Redis**

Redis is an open-source in-memory storage, used as a distributed, in-memory key-value database, cache, and message broker, with optional durability. Because it holds all data in memory and because of its design, Redis offers low latency reads and writes, making it suitable for use cases that require a cache. The cache solution provides the capability of dynamically using the cache that is required.

# **Deployment**

To deploy and run the image in the Kubernetes cluster the following steps are followed.

## **Build and push the image to Artifact Registry**

The first step for deployment is to build the image for the caching application. Jib maven plugin is used to create the image and push it to the gcp artifact registry. Create a new repo in the artifact registry with sufficient permissions for storing the image. Sample command for building and pushing the image is shown below.

*mvn -DskipTests com.google.cloud.tools:jib-maven-plugin:build -Dimage=<gcp-region>-docker.pkg.dev/${GOOGLE\_CLOUD\_PROJECT}/<repo-name>/<image-name>*

## **Deploy the components using YAML files**

Login to Kubernetes cloud shell by entering the command

*gcloud container clusters get-credentials <cluster-name> --region <region> --project <project-name>*

After logging in, start the components in the order as it’s given in the sub section.

### **Apache Ignite**

Configure the yaml file ‘Ignite.yaml’ with the required details. The file contains both container and service related details. The Ignite service named as apache-ignite-service is configured to run on port 10800. To deploy and run, the following command is invoked.

*kubectl create -f Ignite.yaml*

### **Redis**

The yaml file for Redis is named as ‘Redis.yaml’. The file contains both container and service related details. The Redis service named as redis-service is configured to run on port 6379. To deploy and run, the following command is invoked.

*kubectl create -f Redis.yaml*

### **Postgres**

To run Unleash PostgresSQL database is required. Postgres is started in a separate container and as an individual service. The service is named as ‘unleash-db’ and configured to run on port 5432. The service is exposed to access through browser. To deploy and run, the following command is invoked.

*kubectl create -f Postgres.yaml*

### **Unleash web**

Unleash service ‘unleash-web’ is configured to run on port 4242 and the service is exposed. The yaml file is named as ‘Unleash.yaml’. To deploy and run, the following command needs to be invoked.

*kubectl create -f Unleash.yaml*

### **Spring boot app**

The spring boot caching app configurations are made in ‘Springboot.yaml’ file. The app service named as ‘bb-entrypoint’ is configured to run on port 7070. The service is exposed. The container is configured to access the custom image stored in Artifact Registry. To deploy and run, invoke the command below.

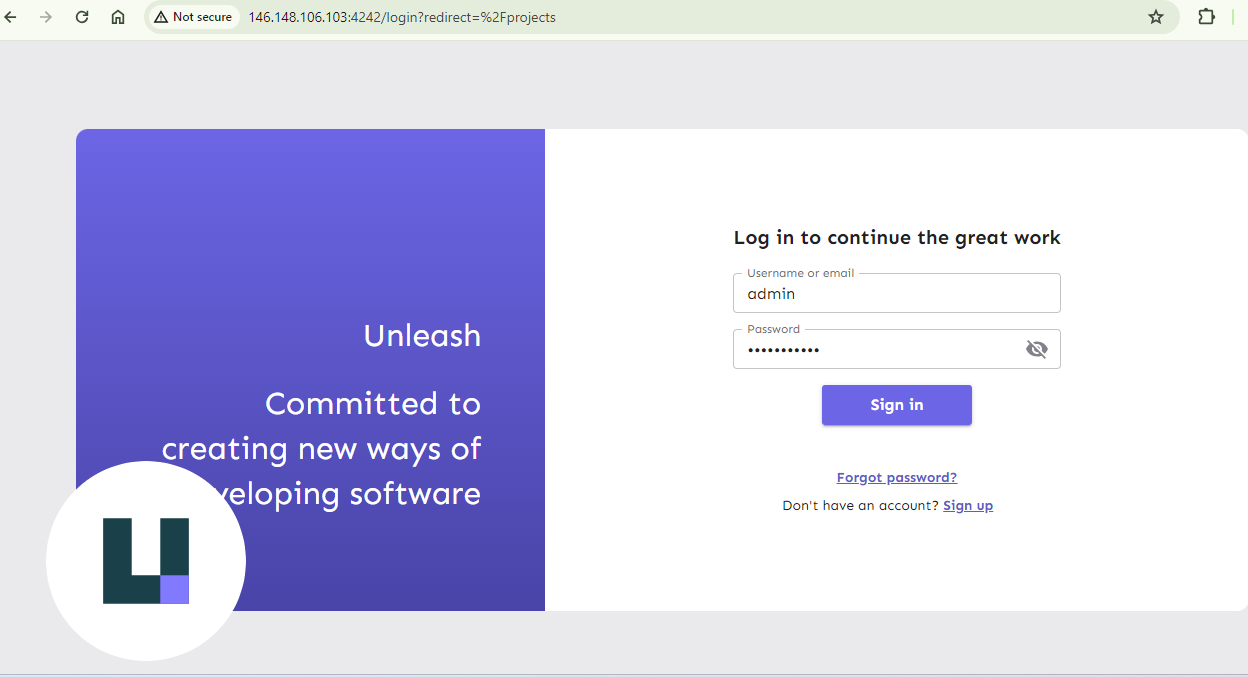
*kubectl create -f Springboot.yaml*

# **Testing**

For testing the application, required to login to Unleash and then create a new admin token. If feature toggle is not created earlier then need to create a new feature toggle. After configuring use Swagger to test the application. Please follow the steps below for testing the Caching solution application.

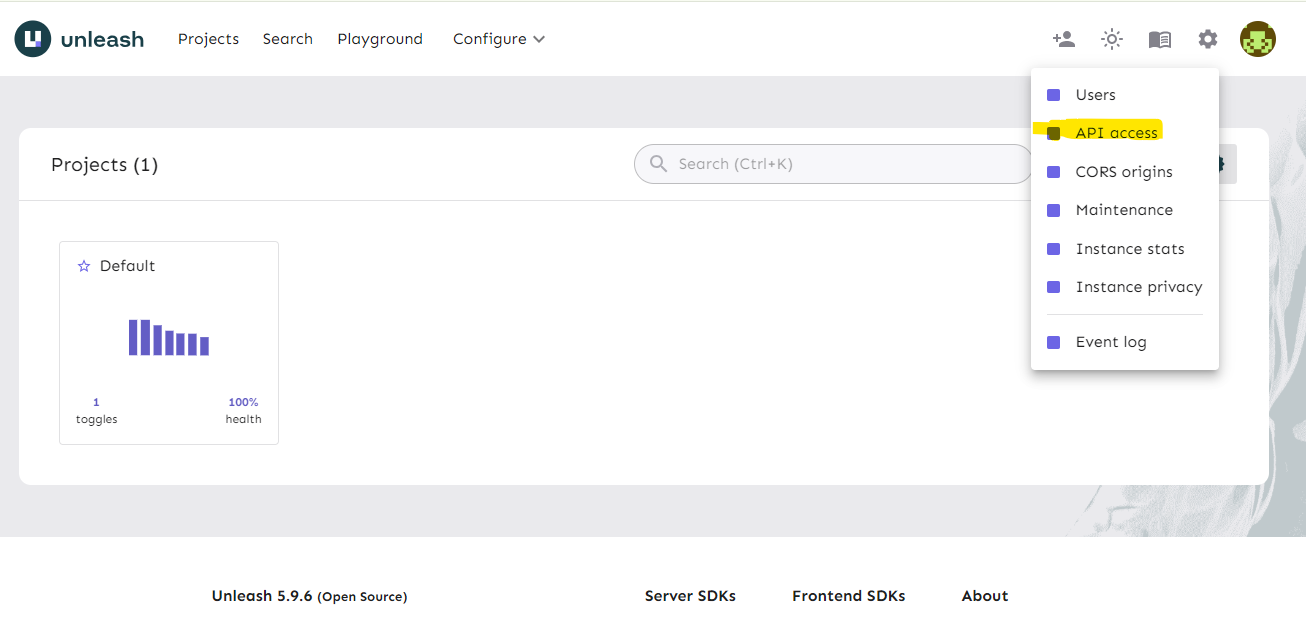
## **Login to Unleash**

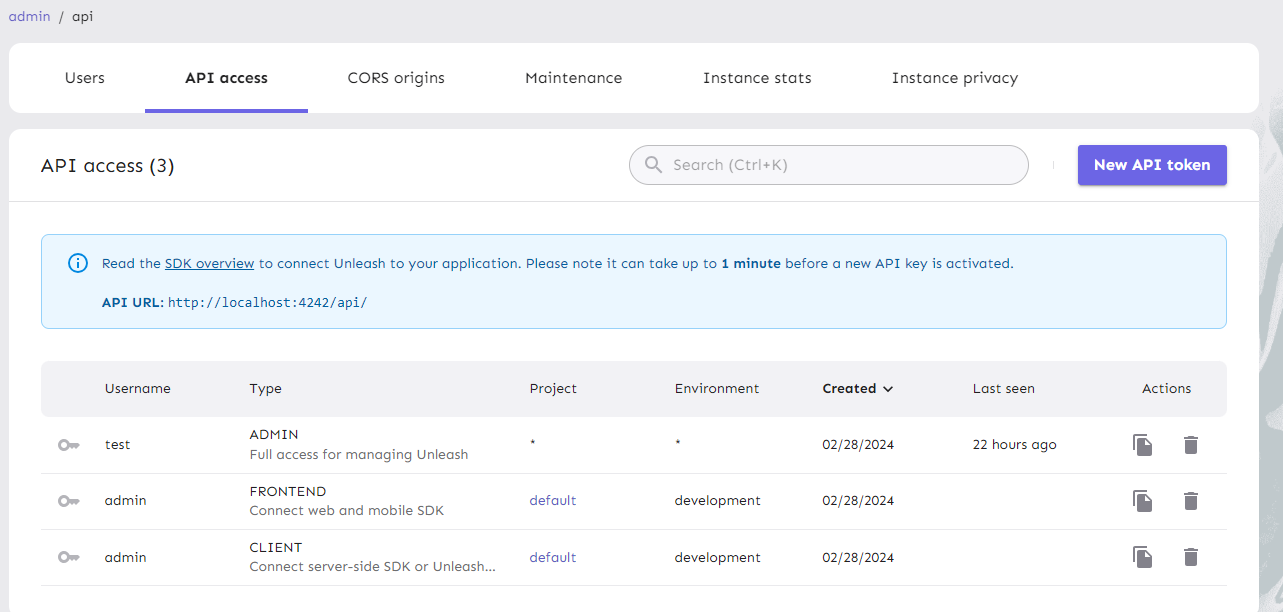
From browser access the URL http://<public-ip>:4242/. Login to unleash using admin credentials. The default password to login is ‘unleash4all’.

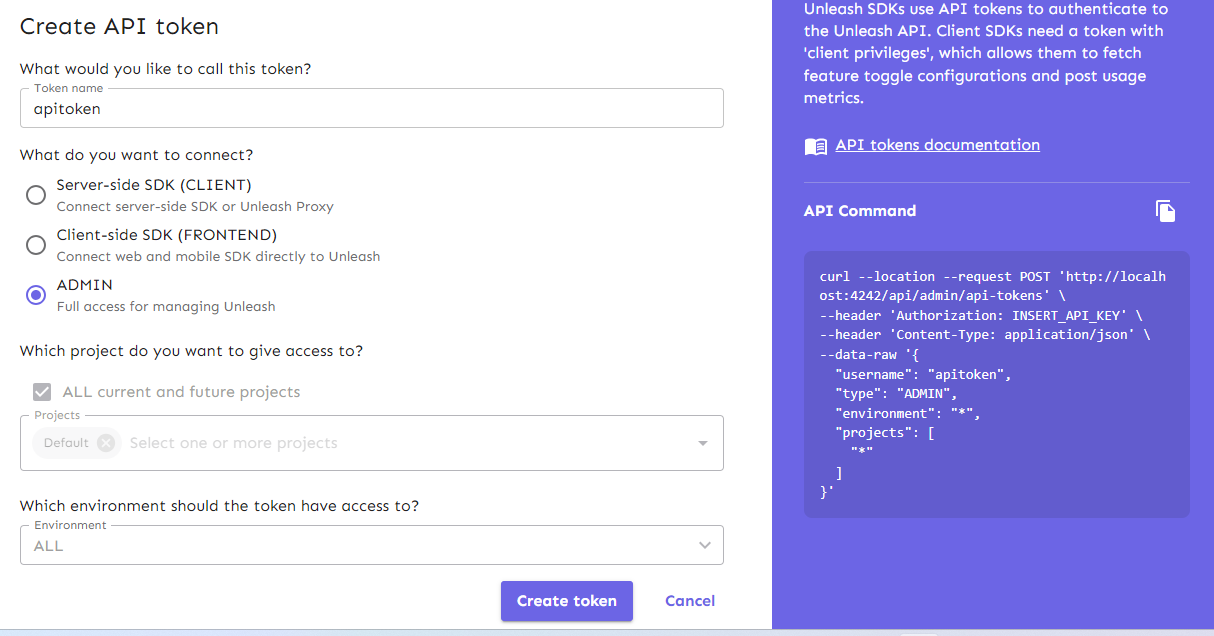
****

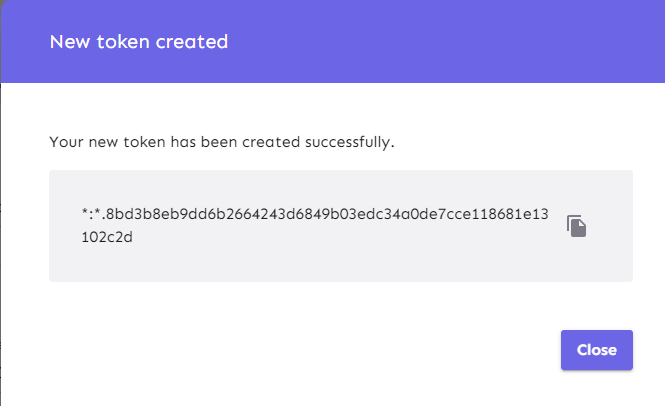
## **Token generation**

From Settings, click ‘API access’. A new window will be opened where there is option to create a ‘New API token. Give a token name, select ‘ADMIN’ and then click ‘Create token. New token will be generated. Its useful to save this token somewhere so that it can be copied and pasted later when required. Screenshots for token generation are pasted below.

****

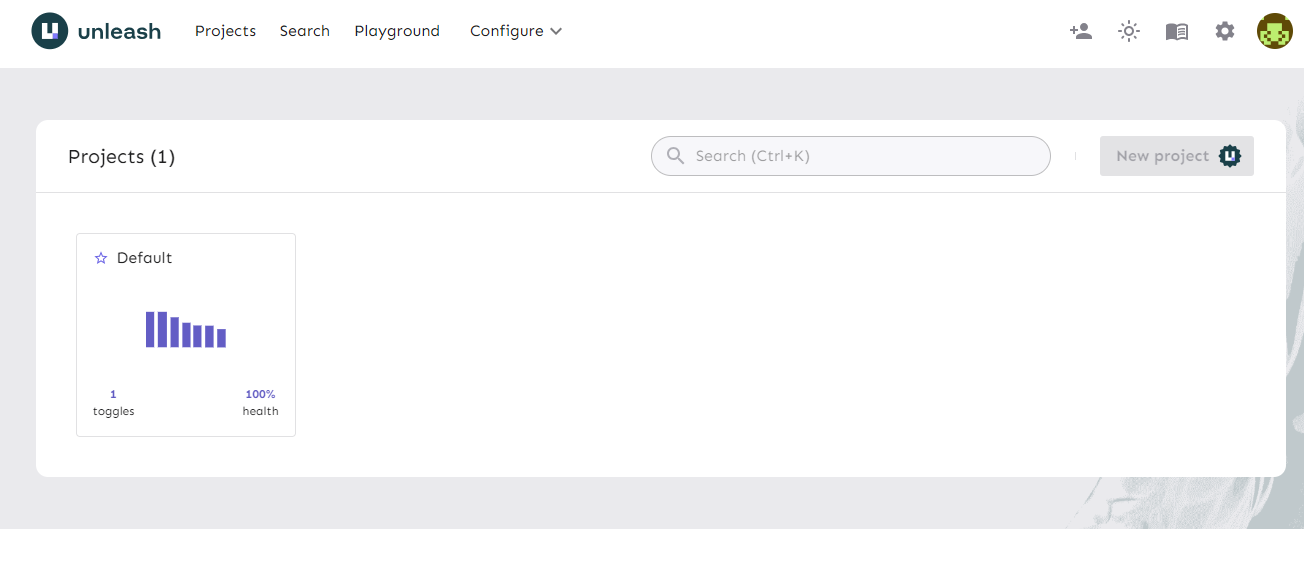
****

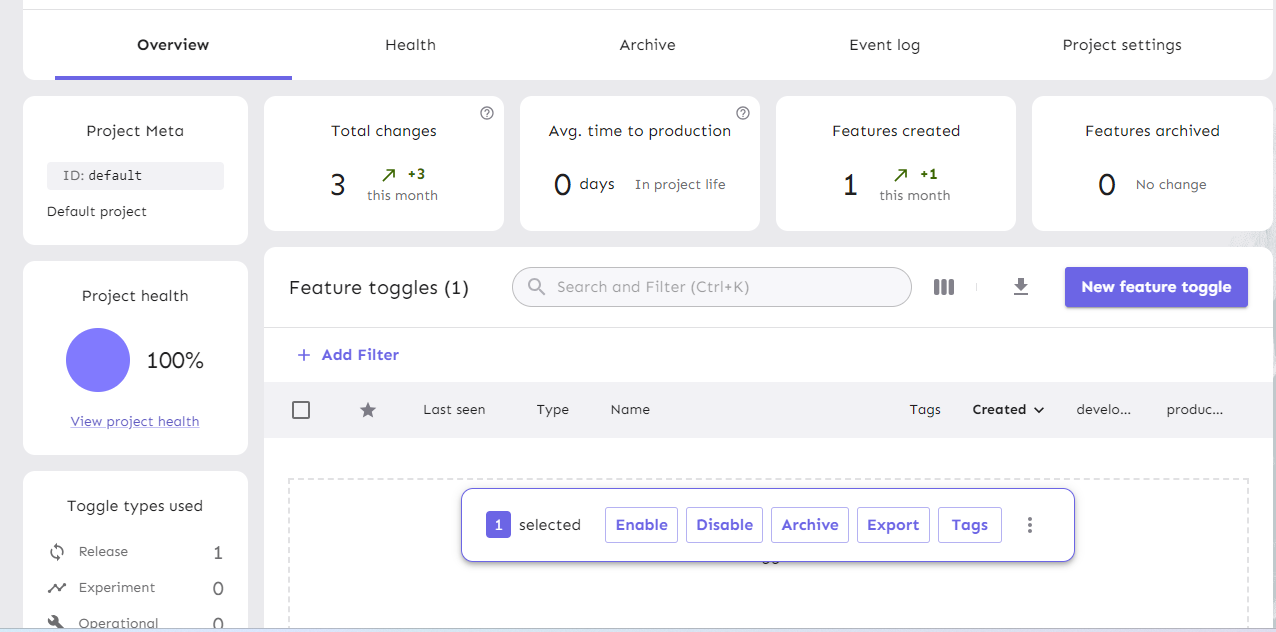
****

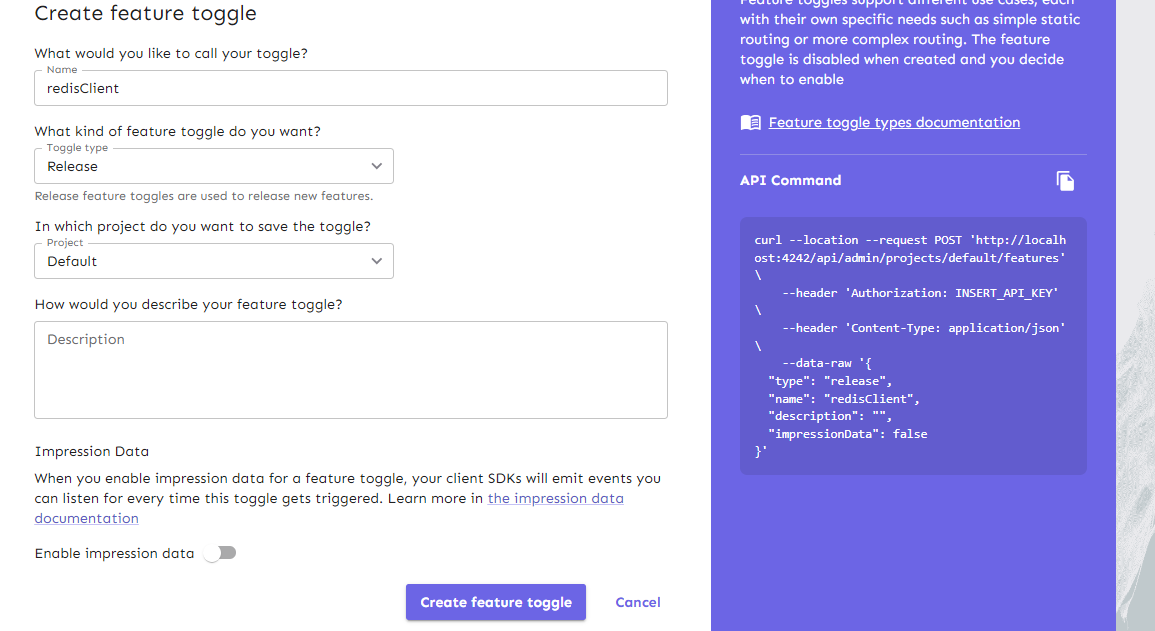
****

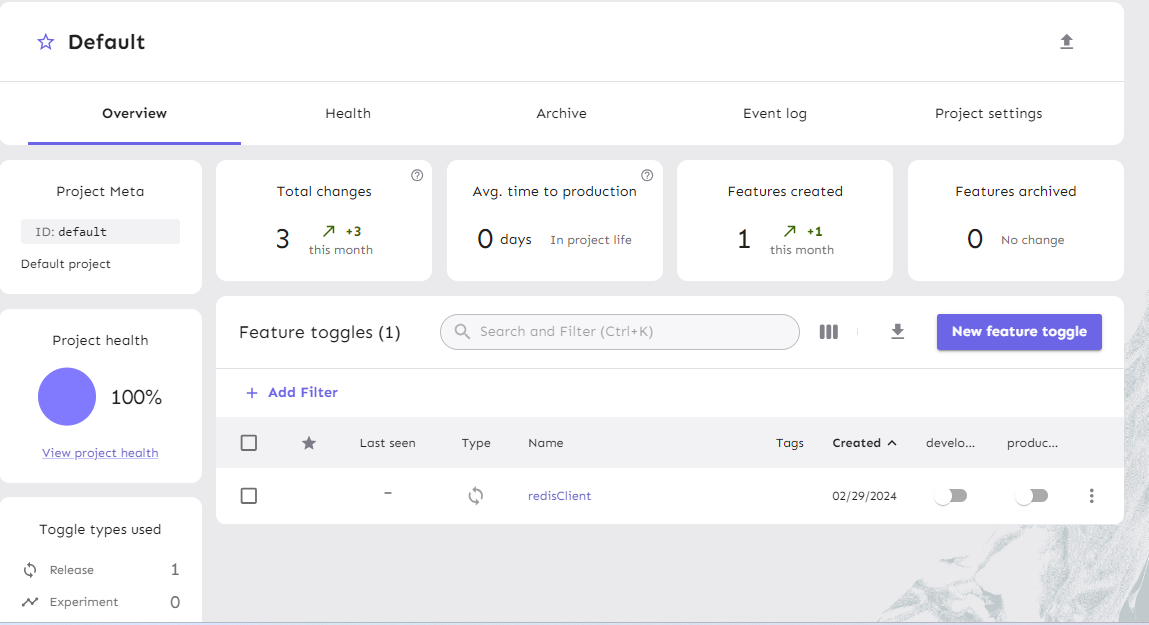
## **Create new feature toggle**

Create a new feature toggle if its not available by referring the screenshots below. Use ‘redisClient’ as the name of the feature toggle. By default the ‘development’ feature is disabled and Ignite will be the underlying cache. By enabling the ‘development’ feature, the cache will be toggled to Redis.

****

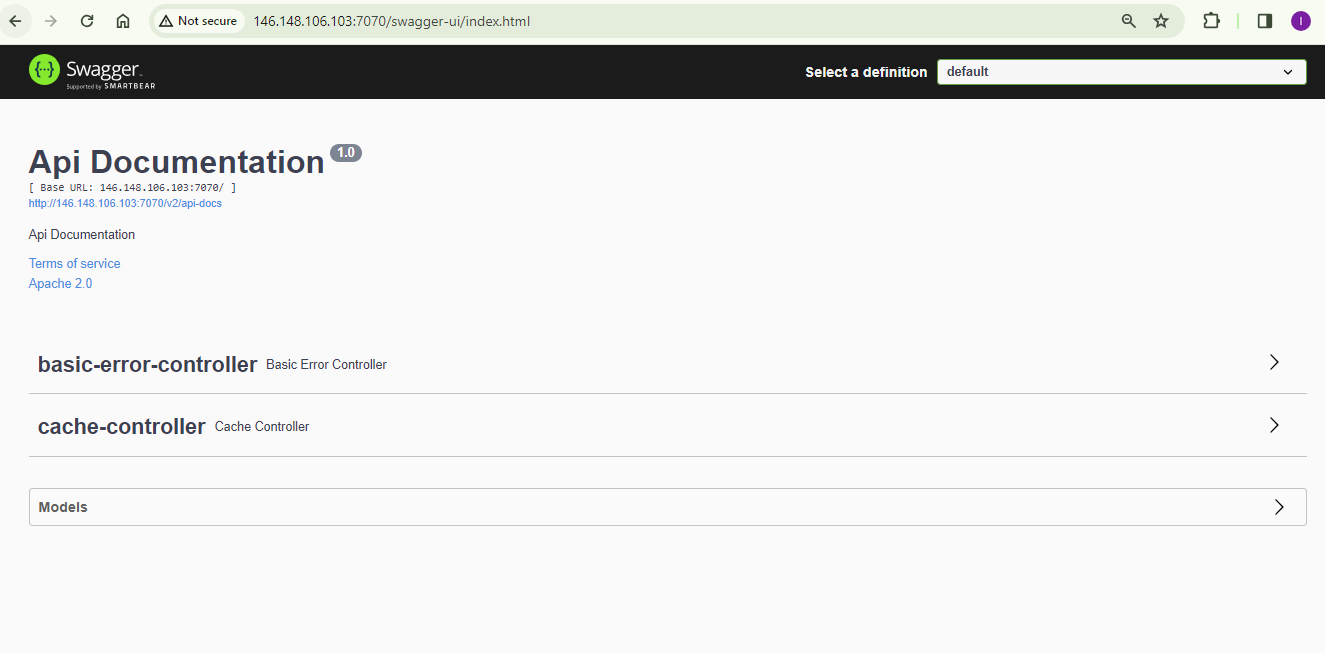
****

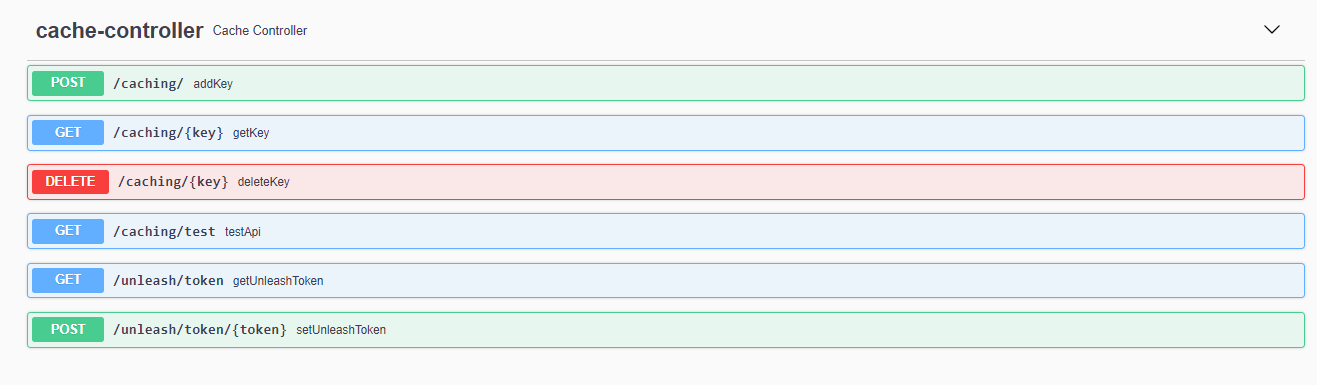
****

****

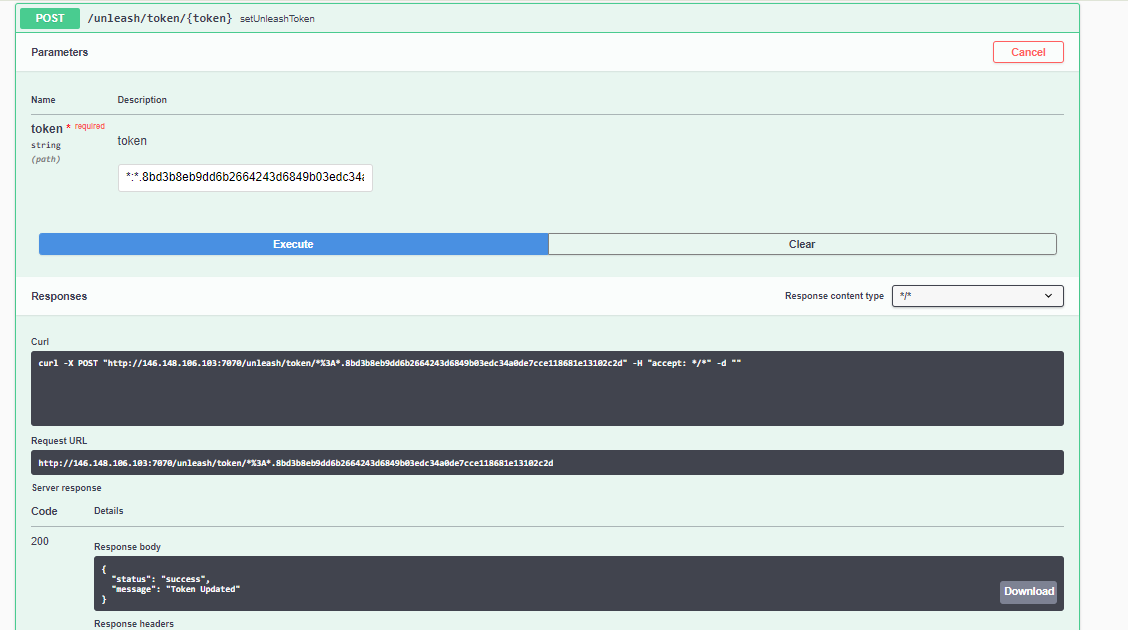
## **Application testing using Swagger**

* Login to Swagger using the url http://<public-ip>:7070/swagger-ui/index.html. No credentials are required. Click ‘Cache-controller’.

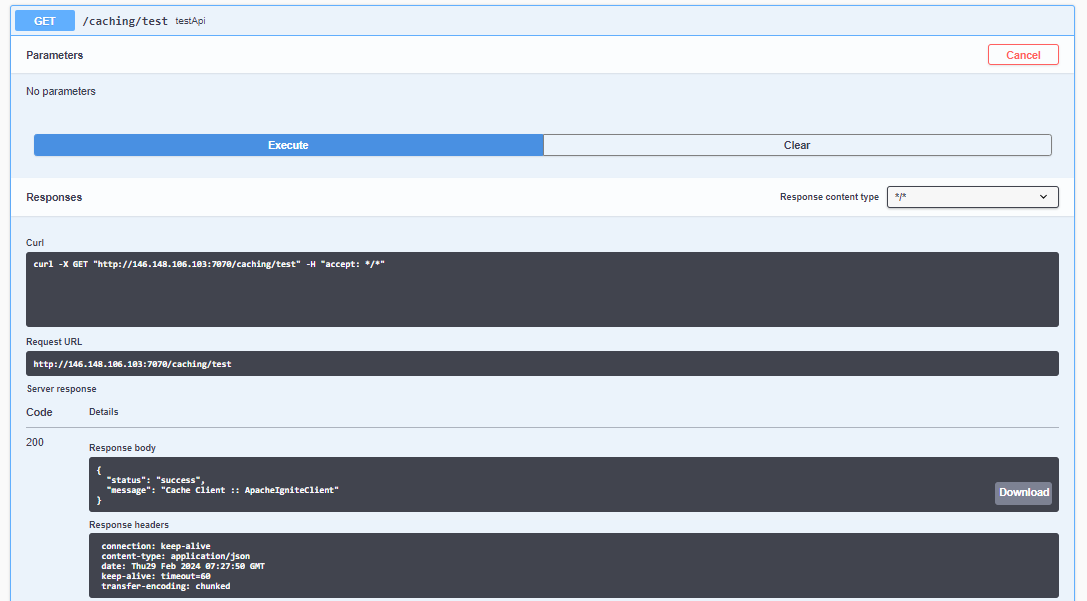
****

****

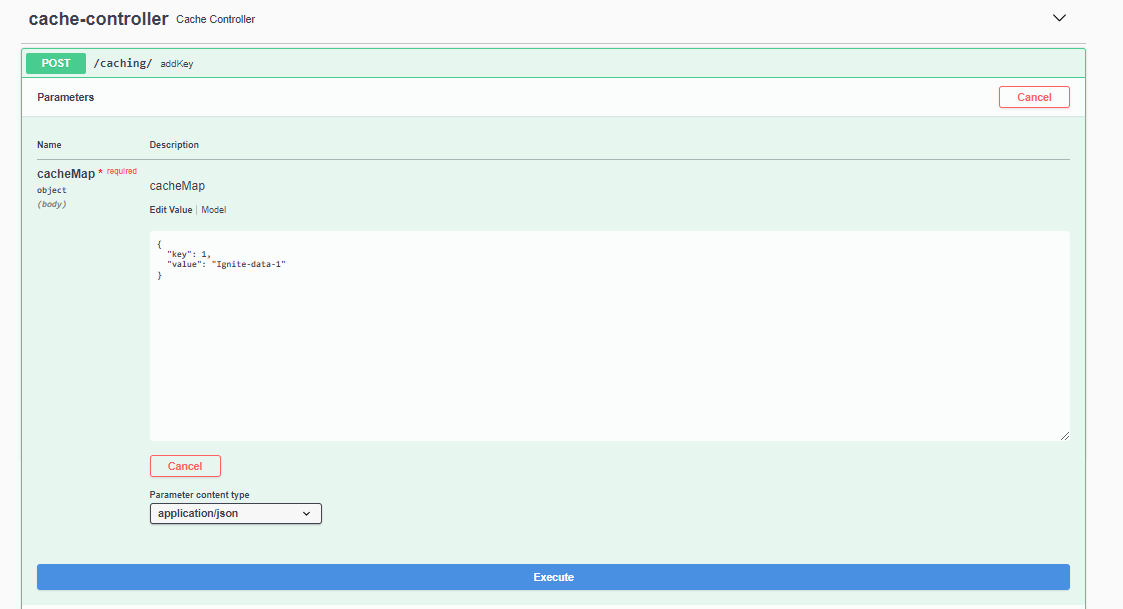
* Execute the below post request to update the latest token in the application so that the same token need not to be passed in the subsequent requests.

****

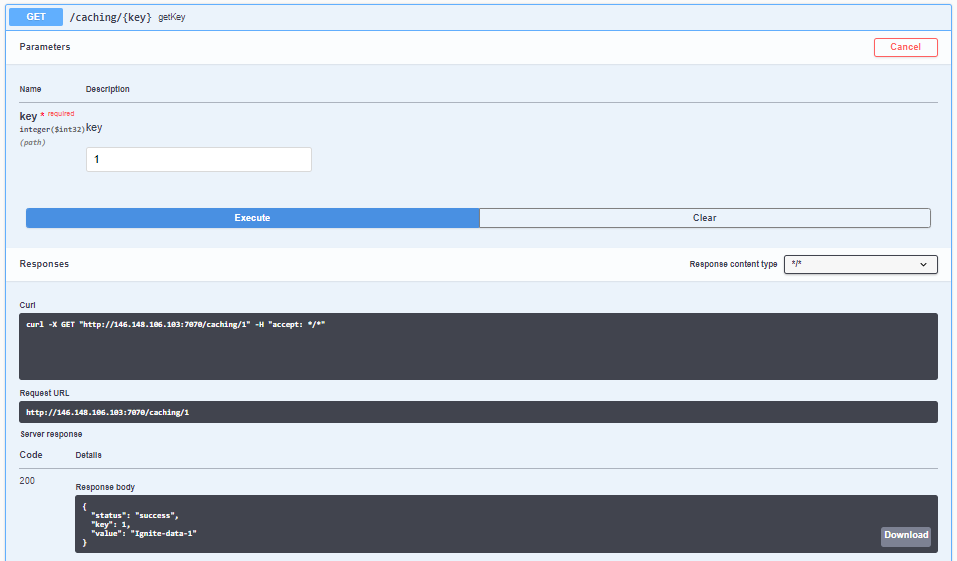
* Execute the below test request to check which is the underlying cache.

****

* Add a new key by clicking the below POST request. Edit the key and value and execute.

****

* Retrieve the key by clicking the below GET request. Enter the key to get the corresponding value.

****

* Continue testing by toggling the feature and changing the cache.