Performance Testing Tool User Guide

1: Initial Setup/Installation

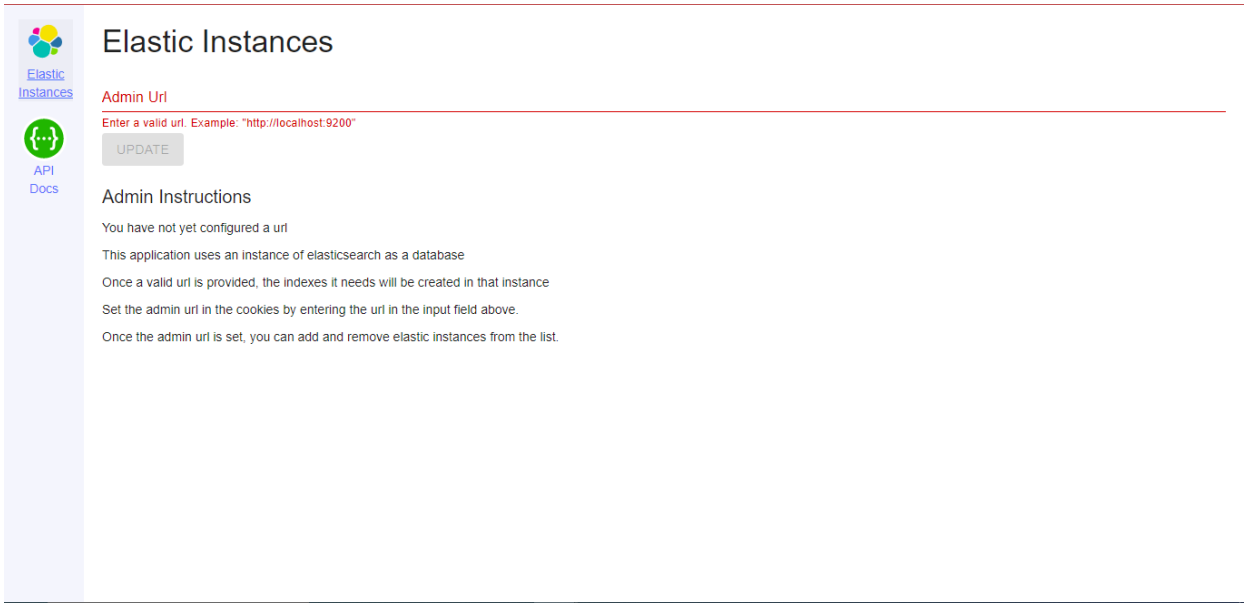
The use of an EC2 instance is recommended for running the application, though it can be run locally on nearly any platform, provided that the Elasticsearch URLs for testing are reachable from wherever the application is installed and running.

You will need to have node installed on the instance. Instructions for installing node on an EC2 instance can be found here:
https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/setting-up-node-on-ec2-instance.html

You will need to start the tool using npm, instructions for this can be found here:
How to npm run start at the background ⭐ | by Ido Montekyo | idomongo | Medium

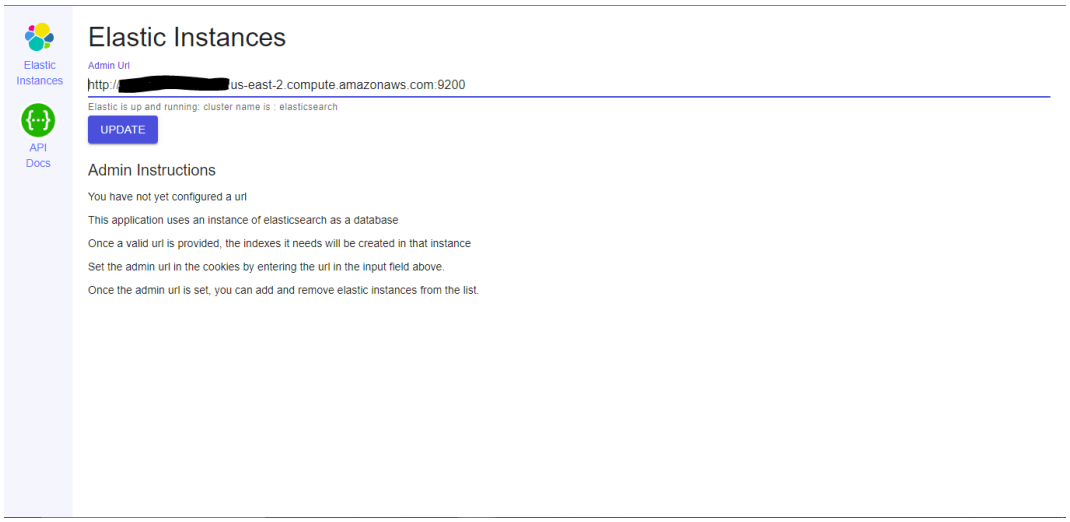
2) Configuring and Using the Tool

Once the tool is installed and running, navigate to the url where it is running and first navigate to the *elastic-instances* page, either by typing it manually or clicking the first item in the navbar to the left

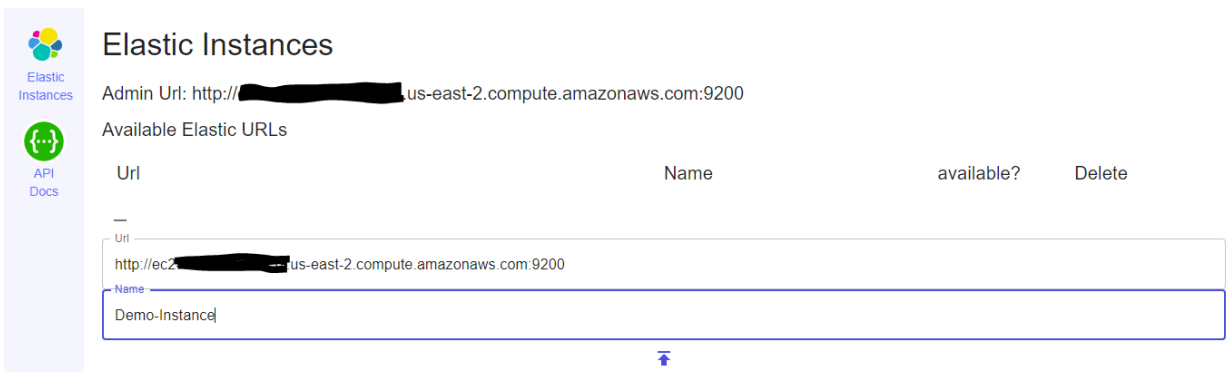


Now enter the URL of the Elasticsearch instance where you want the results from your tests to be stored. When you run tests they will be stored in a new index called “previous-tests.”

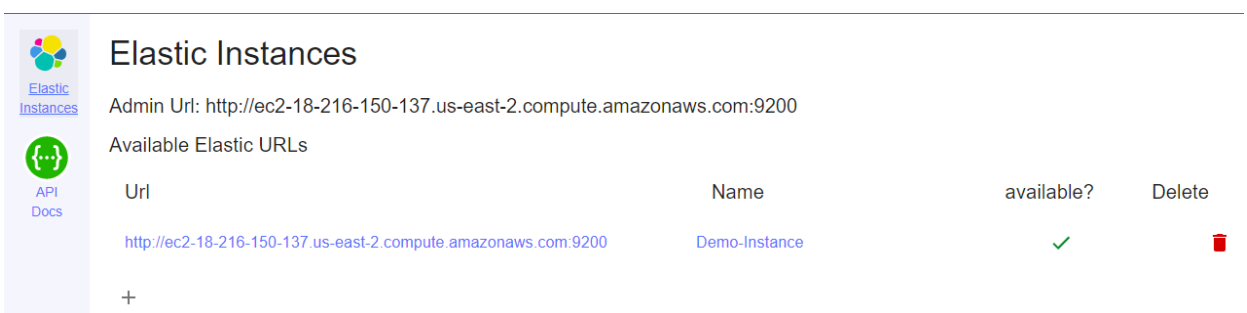
Note that this does not need to be the URL where the instance of Elasticsearch you are testing on is located, although it certainly can be. This is just where the previous tests and information the application needs will be stored.



Once this is done you will be able to add the URLs of the instances you will be testing against to a list like so:

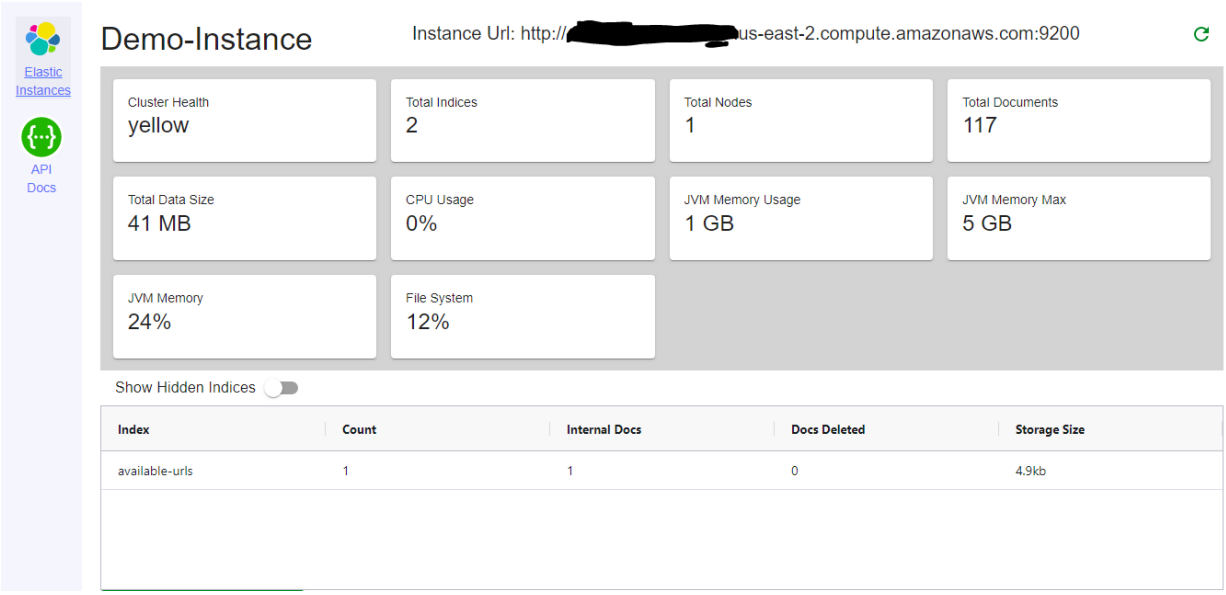


You can always return to this page to add new URLs to test against. The URL field is the base endpoint where the instance can be reached, and the Name field is used only for you to be able to differentiate between the different testing instances you have stored.



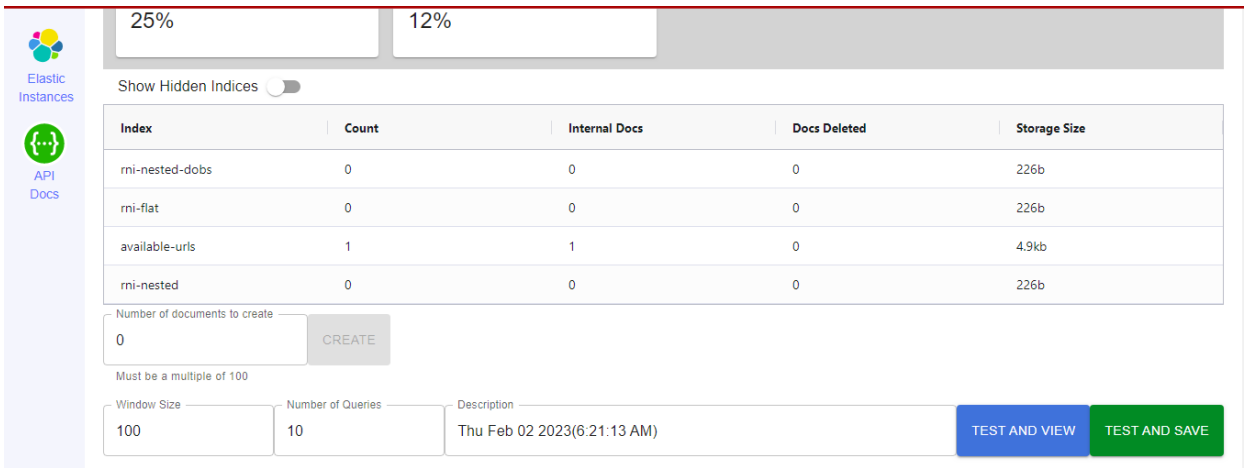
The available tab will be a green check mark if the instance is running and reachable. The delete button will remove it from the list of available instances to test against.

You can now use the link in the url field to drill down into that specific instance. After a brief loading period, you will be presented with some basic information about the elasticsearch instance, including a list of the current indexes.



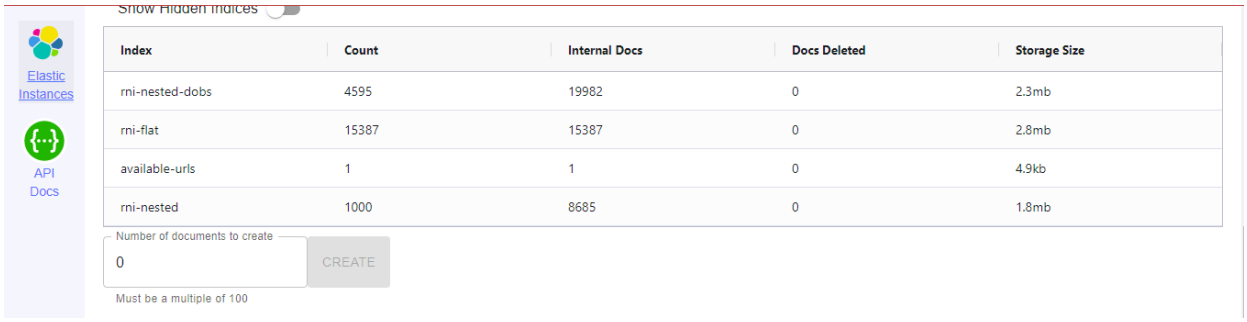
Note that in this instance, there is an index called “available urls” present because I am using this instance as the admin url for this test for writing this doc. You will see some basic information about the index in the table. The count field will show you the count of docs in the index. Note that the “Internal Docs” field is used to display the total number of internal documents that Lucene is using under the hood. Under the hood, Elasticsearch flattens out the nested documents, and it is useful to use this number to understand the amount of overhead needed to store nested field types that are normally abstracted by Elasticsearch.

Now you can scroll down and click the button labeled “Create Testing Indices” and you will be presented with options for populating the indices with mock data. The indices used are ‘rni-nested’, ‘rni-nested-dobs’, and ‘rni-flat’

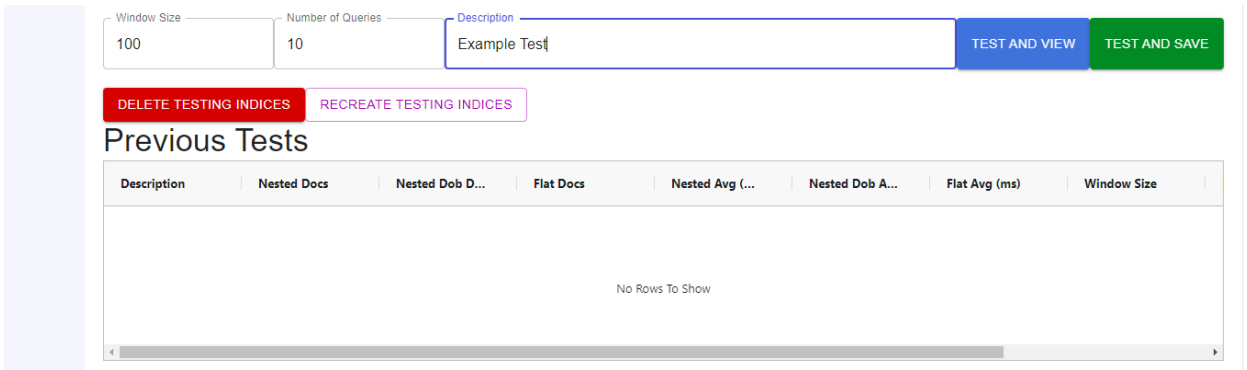


Now you can begin to populate the testing indices with mock data. The process used for this will generate bulk objects that contain the given number of nested documents and the

corresponding flattened out versions. For instance, adding 1000 nested documents will yield the following:



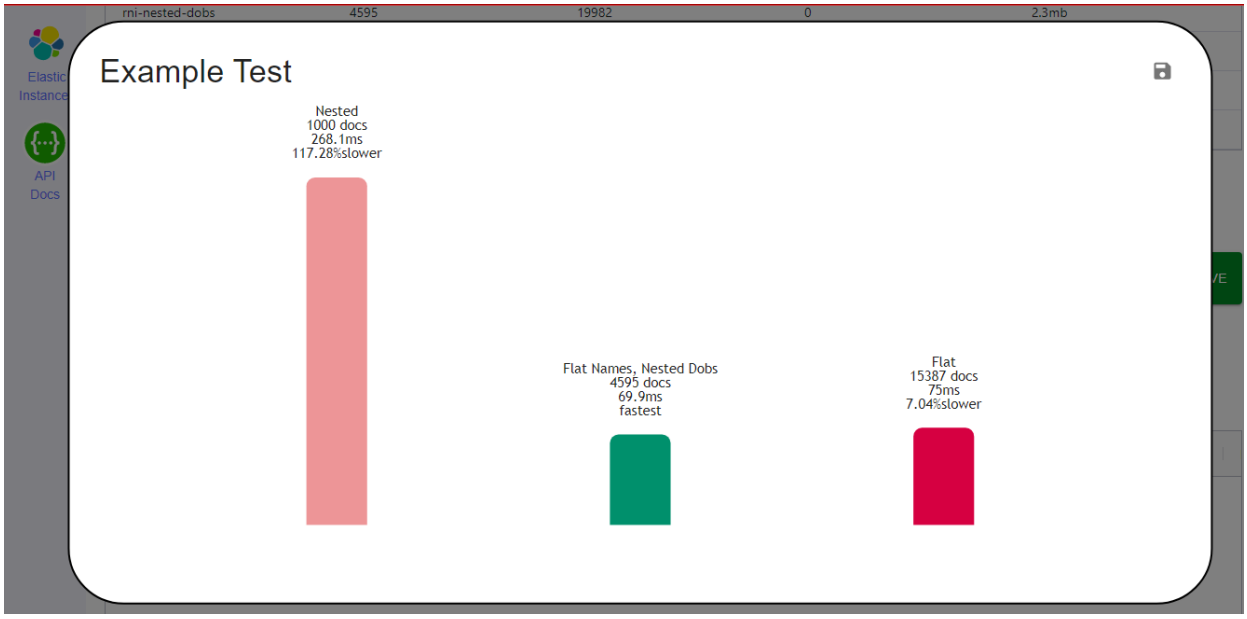
You can now begin executing your first test using the form below that. You will see options for configuring window size and number of queries, and name for the test.



The number of queries is the number of queries that data will be gathered and averaged, for instance a query number of 100 will run 100 queries against each of the different indexes and provide the resulting min, max, and avg values for the time of each query.

The name field will default to a date/time string for the time of execution, or can be edited to make the test easily identifiable in the future.

The button labeled “Test and View” can be used to run the test and display the results in a modal, which the user can then save using the “Save” Icon in the modal that opens with a graph of the results.



The button labeled “Test and Save” will not open the modal and will instead simply execute the test and save the results to the “previous-tests” index. It should then be viewable in the table of previous test results.

Previous Tests

Description	Nested Docs	Nested Dob D...	Flat Docs	Nested Avg (ms)	Nested Dob Av...	Flat Avg (ms)	Window Size	Number of Que...	Delete
Example Test #2	1000	4595	15387	216.4	82.4	75.6	100	10	DELETE
Example Test	1000	4595	15387	239.7	69.4	51.3	100	10	DELETE

Note that this table is optimized for user experience, so some data about the test is abstracted away. This data can be accessed using the api endpoint, ‘{url}/api/fetch-previous-tests’ which will return all of the data, which will look like so:

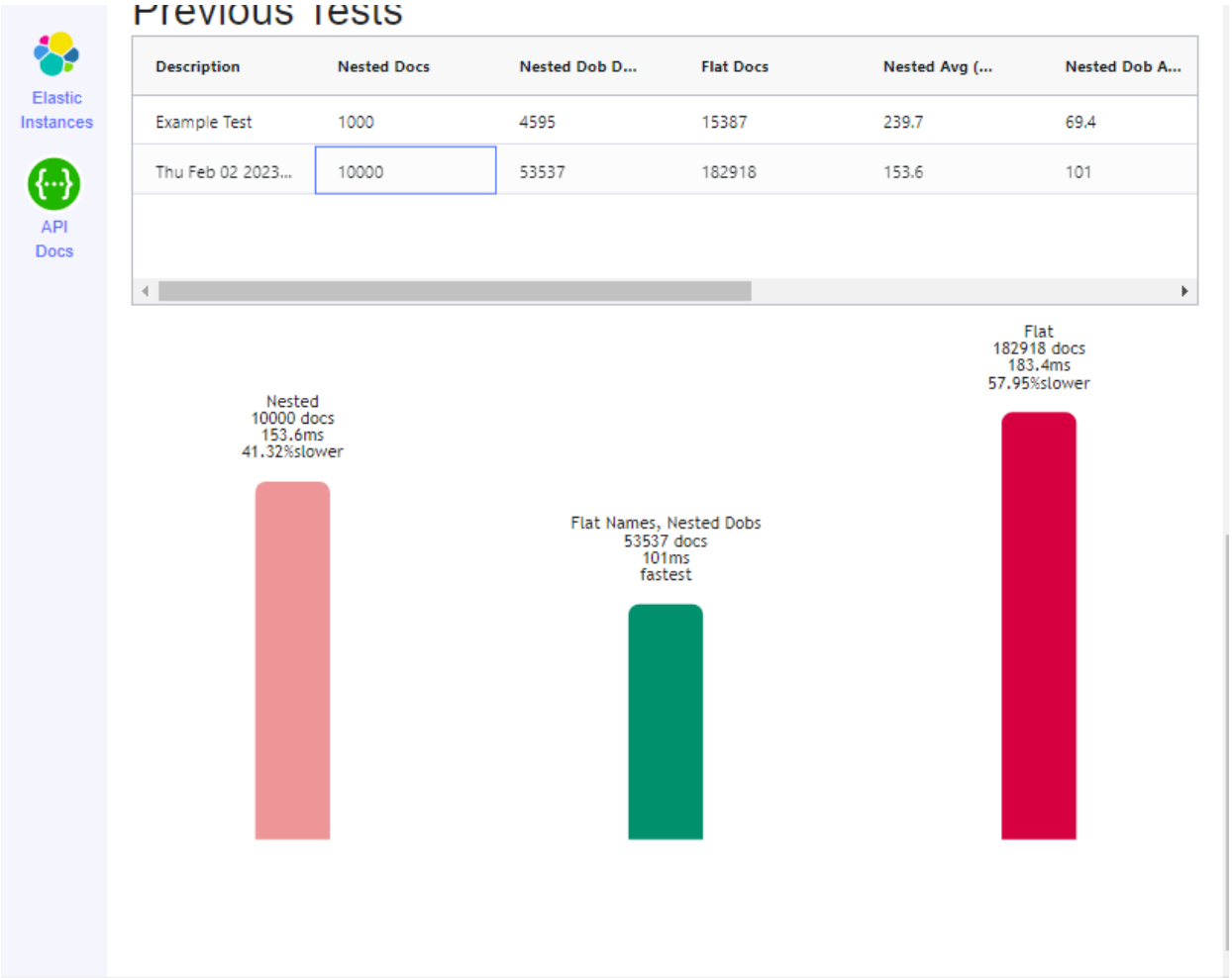
JavaScript

```
{
    "url":
"httpc2-18-216-150-137.us-east-2.compute.amazonaws.com:9200",
    "urlName": "Demo-Instance",
    "window": 100,
    "timeExecuted": "Thu Feb 02 2023(6:42:46 AM)",
    "description": "Example Test",
    "nestedIndexCount": 1000,
    "nestedDobsIndexCount": 4595,
    "flatIndexCount": 15387,
```

```
    "nestedTooks": [
        247,
        167,
        657,
        169,
        121,
        201,
        142,
        113,
        157,
        423
    ],
    "nestedDobsTooks": [
        159,
        56,
        51,
        55,
        38,
        66,
        37,
        92,
        65,
        75
    ],
    "flatTooks": [
        67,
        48,
        48,
        59,
        46,
        89,
        28,
```

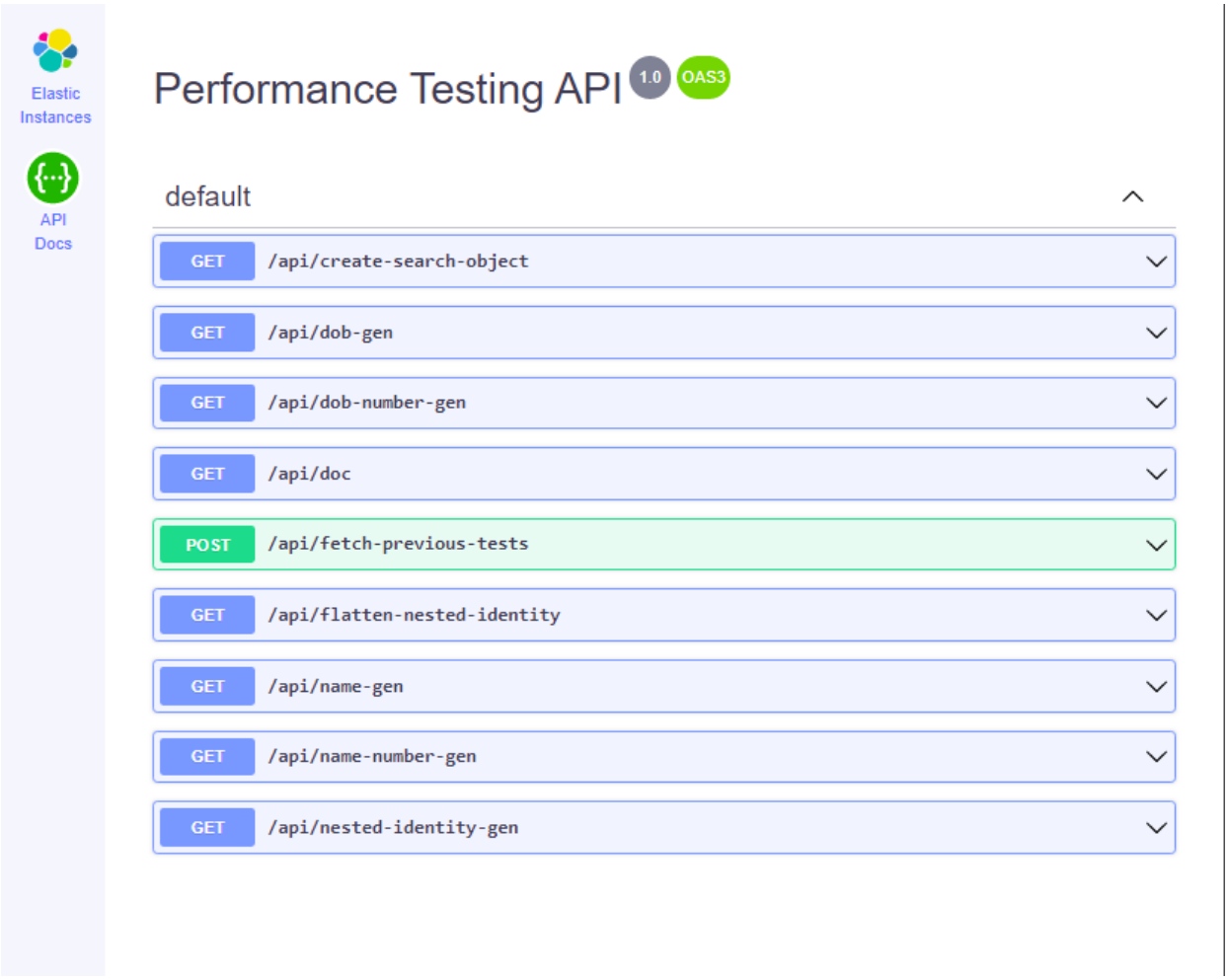
```
        31,
        41,
        56
    ],
    "nested_avg": 239.7,
    "nested_dobs_avg": 69.4,
    "flat_avg": 51.3,
    "nested_min": 113,
    "nested_dobs_min": 37,
    "flat_min": 28,
    "nested_max": 657,
    "nested_dobs_max": 159,
    "flat_max": 89,
    "numberOfQueries": 10
}
```


Below the table of previous tests will be a graph displaying the results of the currently selected test. Clicking on a new test in the table will change the selected test.



3) Using the API Doc

The second link in the menu bar on the left will allow the user to access a Swagger ui with the endpoints for the application mapped.



- *create-search-object*
    - Used to create a default search object. If you would like to create custom search objects, this is the endpoint that will need to be expanded upon. It exists in the ~/pages/api directory in the code
    - The default endpoint will return a search object with a randomly generated name, dob, and a default window size of 100
- *dob-gen*
    - Used to generate a random dob for searching
    - The dates generated are between 1900 and the current date
- *dob-number-gen*
    - Used to generate the number of dobs to created for a nested object
    - This number is between 1 and 242 following the distribution given to us
- *doc*
    - Returns the Swagger json doc for this page
    - This will allow you to plug this into postman
- *fetch-previous-tests*
    - Takes in an admin url string and returns all of the previous tests save there
- *flatten-nested-identity*

    - Returns a randomly generated nested identity with the corresponding flattened objects
    - The logic for flattening nested objects lives in this file at pages/api/flatten-nested-identity
- *name-gen*
    - Returns a randomly generated name
    - Format is “First Middle Last”
    - The name may or may not include a middle name
    - The names are generated from a list of the most common male and female names and most common surnames
    - This file, ~/pages/api/name-gen is where the lists of names used are located. If you wish to add to or remove from these, the arrays can be modified
- *name-number-gen*
    - Returns a random number of aliases for an identity
    - This number is between 1 and 1380 following the distribution given
- *nested-identity-gen*
    - Returns a randomly generated nested identity
    - Example:

```javascript
{
  "ucn": "447b235f431bbbc45d463a7f02b6afd1",
  "aliases": [
    {
      "primary_name": {
        "data": "Joyce Joyce Allen",
        "entityType": "PERSON"
      }
    }
  ],
  "birth_dates": [
    {
      "birth_date": "1944-09-01"
    }
  ]
}
```