VERITAS

# Veritas NetBackup Flex Appliance API Guide

API first to enable integration, automation, and reporting.

# Contents

## Introduction

### Executive Summary

Customers want to manage and monitor Veritas NetBackup™ Flex Appliances along with their other IT assets using external management and monitoring tools to achieve operational efficiency. The demands for automation, reporting, and monitoring are increasing on NetBackup Flex Appliances.

The Flex Appliance public APIs allow customers and developers to integrate customized codes into Veritas products. Streamlined operations improve the user experience and enable customers to access performance data and create graphic diagrams like Grafana Prometheus API integration.

### Scope

The purpose of this document is to provide technical details to assist in understanding the Flex Appliance application programming interfaces (APIs). This white paper describes the Flex Appliance APIs and provides some useful examples.

### Target Audience

This document is for customers, partners, and Veritas field personnel who want to learn about integration and automation with the Flex Appliance REST APIs. Basic knowledge of REST APIs is required for this documentation. If you don't yet understand REST or know how to use REST APIs, read Understanding REST and REST APIs.

### NetBackup Flex Appliance APIs

An API is a set of definitions and protocols for building and integrating application software. It is a way to programmatically interact with a separate software component or resource. APIs hide complexity from developers, extend systems to partners, organize code, and make components reusable. You can find Flex Appliance APIs in Flex Appliance Product API Documents.

### NetBackup Flex Appliance APIs

An API is a set of definitions and protocols for building and integrating application software. It is a way to programmatically interact with a separate software component or resource. APIs hide complexity from developers, extend systems to partners, organize code, and make components reusable. You can find Flex Appliance APIs in Flex Appliance Product API Documents.

## Getting Started

You can use the GUI tool Postman or the command line tool cURL for an API testing environment to transfer data through URLs.

HTTP request methods loosely correspond to the paradigm of CRUD (Create, Update, Read, Delete). Although CRUD refers to functions used in database operations, those design principles can also apply to HTTP verbs in a RESTful API.

| Action | Request Method | Definition |
|---|---|---|
| Read | `GET` | Retrieves a resource |
| Create | `POST` | Creates a new resource |
| Update | `PUT` | Updates an existing resource |
| Delete | `DELETE` | Deletes a resource |

Access tokens allow an application to access an API. After a user successfully authenticates and authorizes access, the application receives an access token and then passes the access token as a credential when it calls the target API (see Figure 1). The passed token informs the API that the bearer of the token has been authorized to access the API and perform specific actions specified by the scope that was granted during authorization. From Flex 2.1, you have two options for API access tokens—a metrics token and a support token. Each token has specific APIs it can use. Metrics tokens can call metrics APIs and support tokens can make calls to the logs collect APIs. You can also set a token lifetime to avoid having to regenerate the token too often.

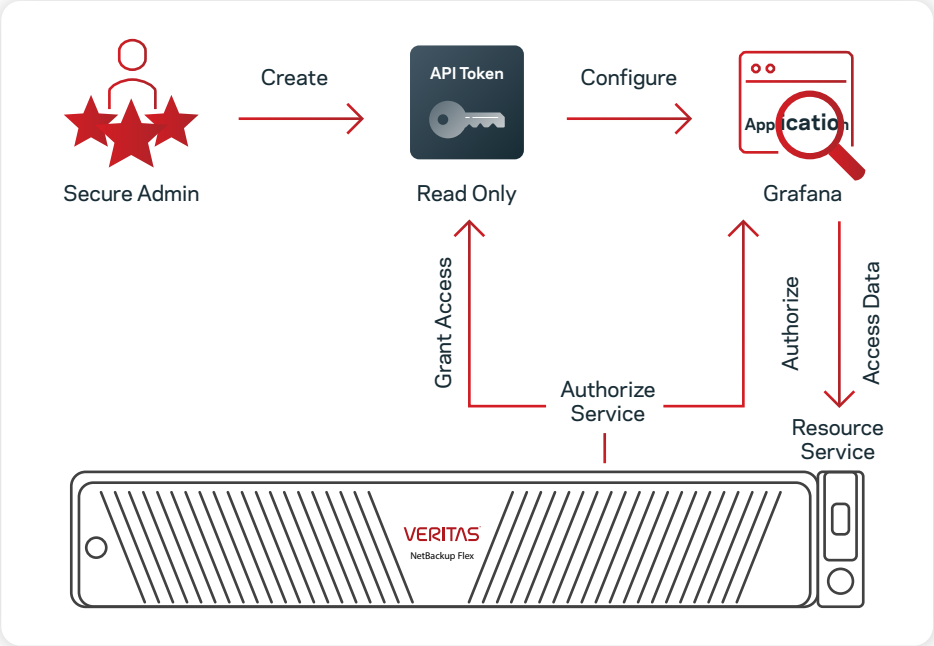
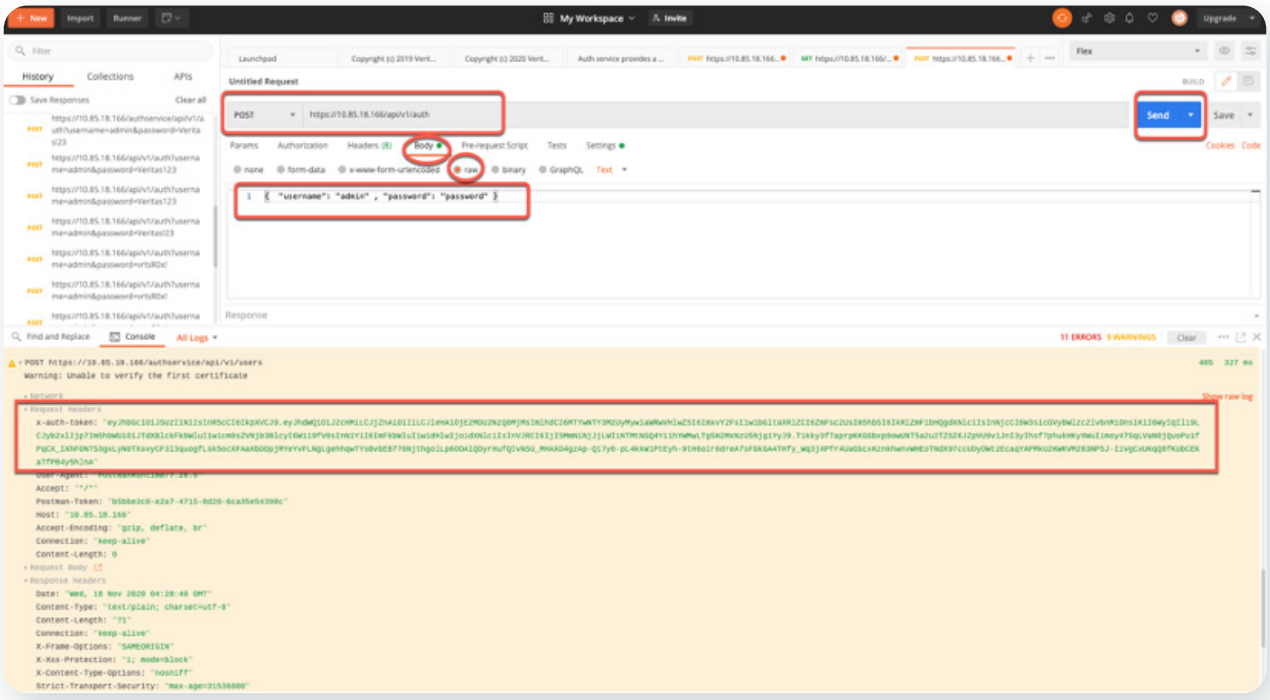


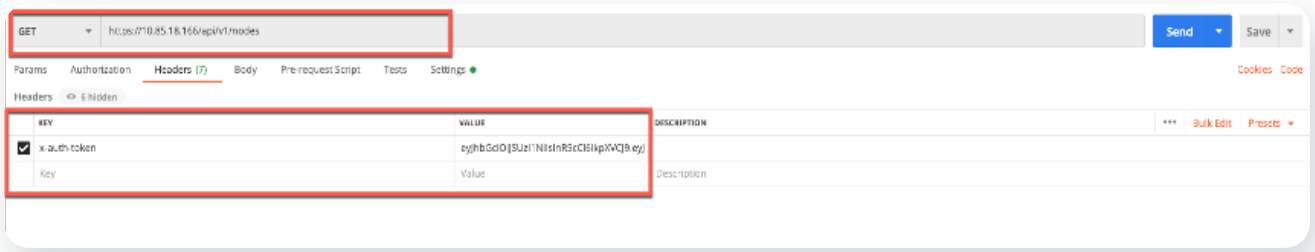

*Figure 1. An overview of the access token request process.*

Before you make any API requests, you need to acquire an access token.

1. Download Postman for your environment.
2. In Postman, select the **POST** method.
3. Enter the following in the request URL: **https://<Flex Appliance Console IP address>/authservice/api/v1/auth**
4. On the **Body** tab, select **Raw**, and enter the Flex Appliance Console username and password. For example: {"username": "admin" , "password": "password" }
5. Click **Send**.



6. Copy the token value and create an x-auth-token key in the header of an API request.




Request URL:

```
Post https://<Flex Appliance Console IP address>/authservice/api/v1/auth
```

Request body:

```
{
     “username”: “APIUser” ,
     “password”: “password”
}
```

A successful response body:

```
{     “username”:”admin”,
     “token”:”eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJhdWQiOiJ2cHMiLCJjZnAiOiIi….”
}
```

## Common Use Cases

This section provides the sequence of APIs for some common use cases.

### Creating Flex Users

Request method and URL:

```
Post https://<Flex Appliance Console IP address>/authservice/api/v1/users/local/APIUser
```

Request body:

```
{
     subject: “APIUser”,
     password: “password”,
     fullName: “API User”,
     idpType: “local”
}
```

A successful response body:


```
{
"users":[{"id":"0f70bfc7-df56-432e-b4c0-dbafa7bbf709","subject":"APIUser","fullName":"API
User","disabled":false,"roles":[{"name":"Administrator","roleScopes":[]}],
"subType":"user",
"idpType":"local",
"idpID":"00000000-0000-0000-0000-000000000000","userScopes":[],"expired":true},{"id":"06364110-ec48-4f40
a206-e2d8875740ca","subject":"admin","fullName":"Default
```


### Deploying a NetBackup Instance

To create a NetBackup primary instance, you need to call a sequence of APIs:

- Configure at least one network interface. You can configure a physical interface, add a VLAN tag, or create a bond.
- Add at least one tenant.

#### Creating a Tenant

The following API adds a new tenant to the database. The request returns an error if the tenant is already present.

Request method and URL:

```
Post https://<Flex Appliance Console IP address>/api/v1/tenants
```

Request body example:

```
{
  "applications":[],
  "emails":[],
  "id":"",
  "location":"veritas",
  "name":"apiuser",
  "network":{"domainName":"xxxxxx.xxxxx.com",
       "searchDomains":["xxxxx.xxxxxx.com"],
       "nameServers":["xxx.xxx.xxx.xxx"],
       "etcHostsPath":""
       },
  "storage":{}
}
```

A successful response body:

```
{
  “id”:”8954616793330828111”,
  “name”:”apiuser”,
  “location”:”veritas”,
  “emails”:[],
  “network”:{
       “domainName”:”xxxxx.veritas.com”,
       “searchDomains”:[“xxxxx.veritas.com”],
       “nameServers”:[“172.8.2.3”],
       “etcHostsPath”:””
       },
  “storage”:{},
  “applications”:[]
}
```

**Creating a Bond**

The following API creates a bond on the specified node.

Request method and URL:

```
Post https://<Flex Appliance Console IP address>/api/v1/network/bonds
```

Request body:

```
{
  interfaces: [“nic2”, “nic3”],
  mode: “802.3ad”,
  node: “davidcl01vm073”
}
```

A successful response body example:

```
{“task-id”:”655f4cab81523a9fa6badd9e82e092a1”}
```

**Configuring a Network VLAN**

The following API adds a virtual LAN (VLAN) tag to a network interface on a node.

Request method and URL:

```
Post https://<Flex Appliance Console IP address>/api/v1/network/vlans
```

Request body:

```
{
  node_name: "davidcl02vm149",
  macvlan_tag: "",
  macvlan_interface: "Bond1",
  macvlan_subnet: "10.85.34.0/23",
  macvlan_gateway: "10.85.34.1",
  ipv6:"0"
}
```

A successful response body:

```
{"status":"Done"}
```

**Creating an Instance**

The following API creates an instance using user questions from the application profile. It includes a precheck of the network information and storage space. The request returns a task ID you can use to check the status of the operation.

Request method and URL:

```
Post https://<Flex Appliance Console IP address>/api/v1/instance
```

Request body:

```
{
  application.name: "refcontainer",
  application.version: "1.0.0",
  envvars:
       [{id: "AUTH_refcontainer/main_1.0.0", value: "yes"},…]

       { id: "AUTH_refcontainer/main_1.0.0", value: "yes"},
       {id: "ADMIN_USER_refcontainer/main_1.0.0", value: "admin"},
       {id: "ADMIN_PASSWORD_refcontainer/main_1.0.0", value: "P@ssw0rd"},
       network:
  {
       {id: "hostname_refcontainer/main_1.0.0", value: "r3"},
       {id: "interface_refcontainer/main_1.0.0", value: "bond0"},
       {id: "ipaddress_refcontainer/main_1.0.0", value: "10.85.16.3"},
```

```
        {id: "tenant_refcontainer/main_1.0.0", value: "9147453964303818591"},

        {id: "domainname_refcontainer/main_1.0.0", value: " xxxx.xxxxx.com "},

        {id: "nameservers_refcontainer/main_1.0.0", value: "xxx.xxx.xxx.xxx"},

        {id: "searchdomains_refcontainer/main_1.0.0", value: "xxxx.xxxxx.com"},

        {id: "etchosts_refcontainer/main_1.0.0", value: ""}

        {id: "static-routes":[{"nic":"bond0","gateway":" xxx.xxx.xxx.xxx ","ipaddress":"10.85.24.0/21"}],"
        default-network-interface":"bond0"}]

        volume:

        {id: "data-vol1_refcontainer/main_1.0.0", value: "4GB"}

        {id: "data-vol2_refcontainer/main_1.0.0", value: "2GB"}

}
```

A successful response body example:

```
{"task-id":"655f4cab81523a9fa17263getw526asd"}
```

**Updating an Instance**

The following API…

Request URL: PUT https://<Flex Appliance Console IP address>/api/`v1/instances/{instanceID}`

Request body:

```
  {"network":{"nic_ip":[{"nic":"xxxxx","ipaddress":"xxx.xxx.xxx.xxx"},

  {"nic":"bond_vlan.976","ipaddress":"10.255.6.21"}],

  "domain-name":"xxx.xxx.xxx",

  "name-servers":"",

  "search-domains":"",

  "etc-hosts":"",

  "default-network-interface":"bond_vlan.977",

  "static-routes":[{"nic":"bond_vlan.976","gateway":"10.255.6.1","ipaddress":"10.85.24.0/21"},

  {"nic":"bond_vlan.977","ipaddress":"10.255.7.0","gateway":"10.255.7.1"}]},

  "config-version":"fZhy"}
```

A successful response body example:

```
{"status": "SUCCESS", "message": ""}
```

**Checking Task Status**

Retrieves the details of a task.

Request URL: GET https://<Flex Appliance Console IP address>/api/`v1/tasks/{taskID}`

A successful response body:

```
{
"id":"2907843d2b802585de9df85572bb37bb",
 "status":{…},
"type":"Upgrade instance",
"details":{…},
.
.
.
}
```

Detail check Flex API guide.

## Upgrading an Instance

The following API…

Request method and URL:

```
Put https://<Flex Appliance Console IP address>/api/v1/instances/<instanceID>/upgrade
Example: https://<Flex Appliance Console IP address>/api/v1/instances/wLzy/upgrade
```

Request body:

```
{
  application.upgrade.version: "8.2",
  envvars: [{id: "ENV_NB_SUSJOB_netbackup/main_8.2", value: "1"}],
  volume: []
}
```

A successful response body:

```
{"task-id":"284d43b98ae6d16d8384a20f76820d1c"}
```

## Integrating with Grafana

Grafana is an open-source analytics and monitoring solution that lets you query, visualize, alert, and understand your metrics, no matter where they are stored. You can use Flex Appliance APIs with Grafana to create, explore, and share usage and performance dashboards. To install Grafana, refer to the official Grafana documentation. To check Grafana metrics, look at these examples:

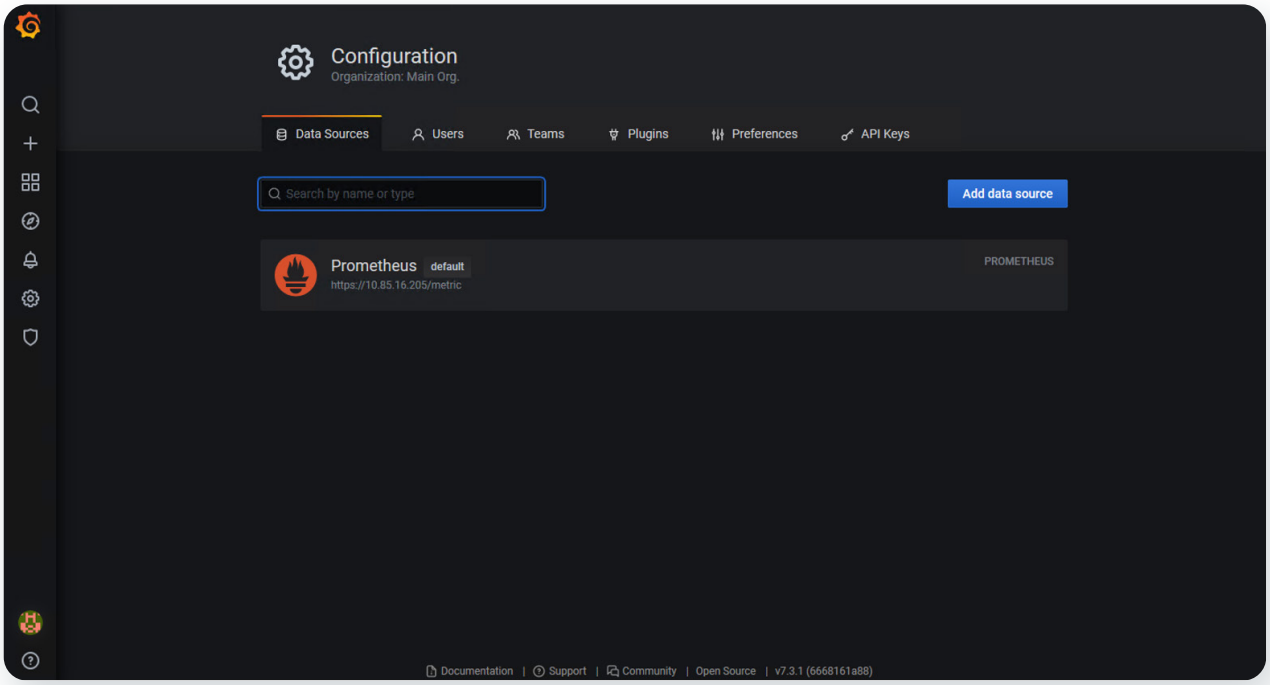
https://github.com/prometheus/node_exporter

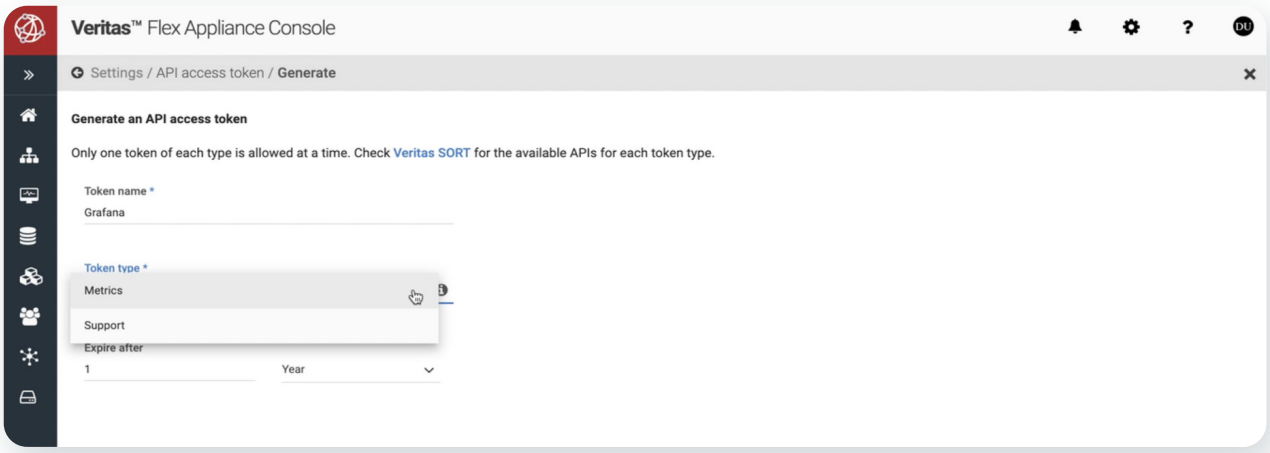
https://github.com/google/cadvisor/blob/master/docs/storage/prometheus.md

Following are the steps to configure a Prometheus data source in Grafana to receive data from the Flex metrics API:

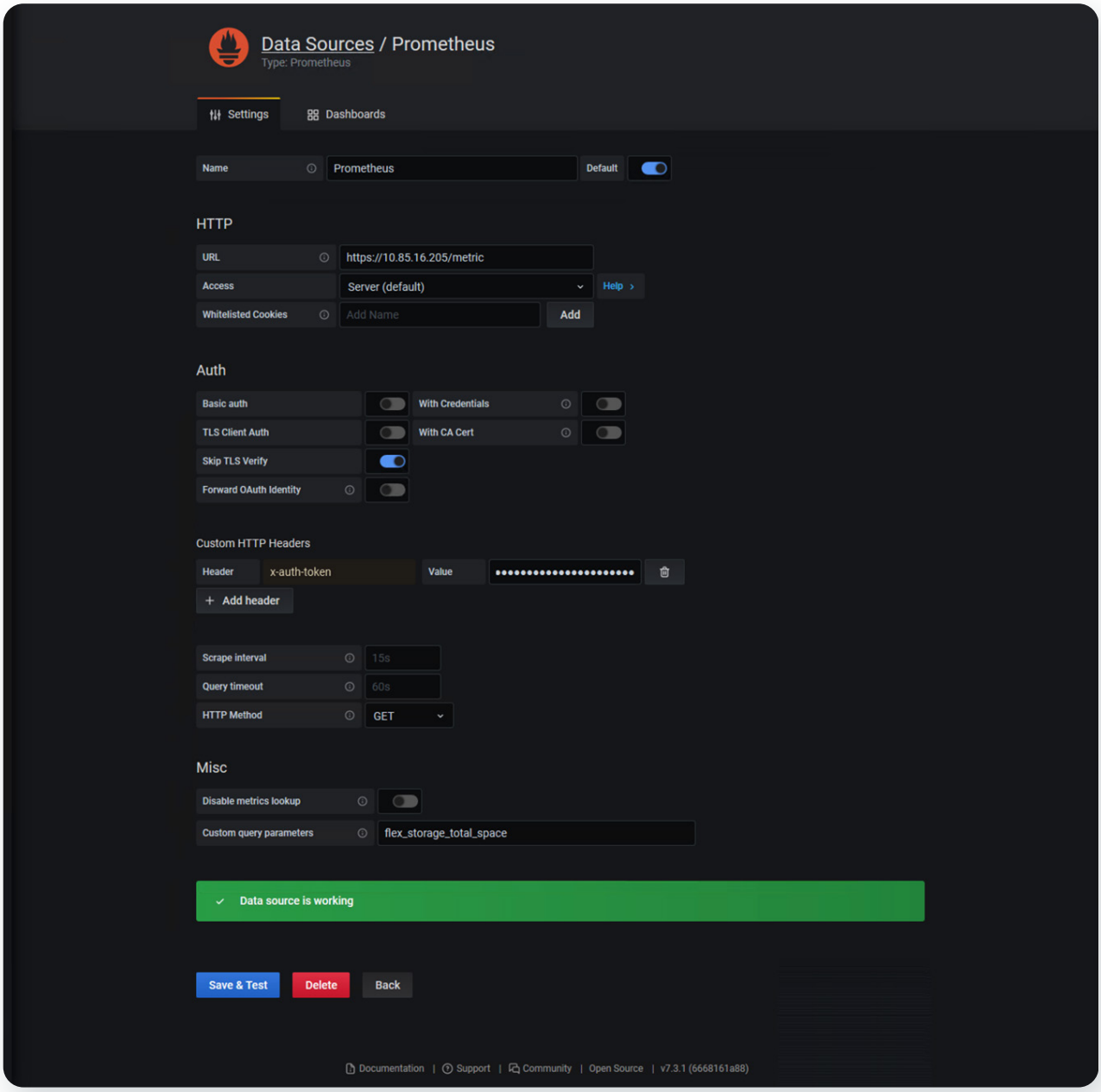
### Creating a Prometheus Data Source

1. Log on to Grafana with a web browser. Click on Configuration > Data Sources and add Prometheus data sources.



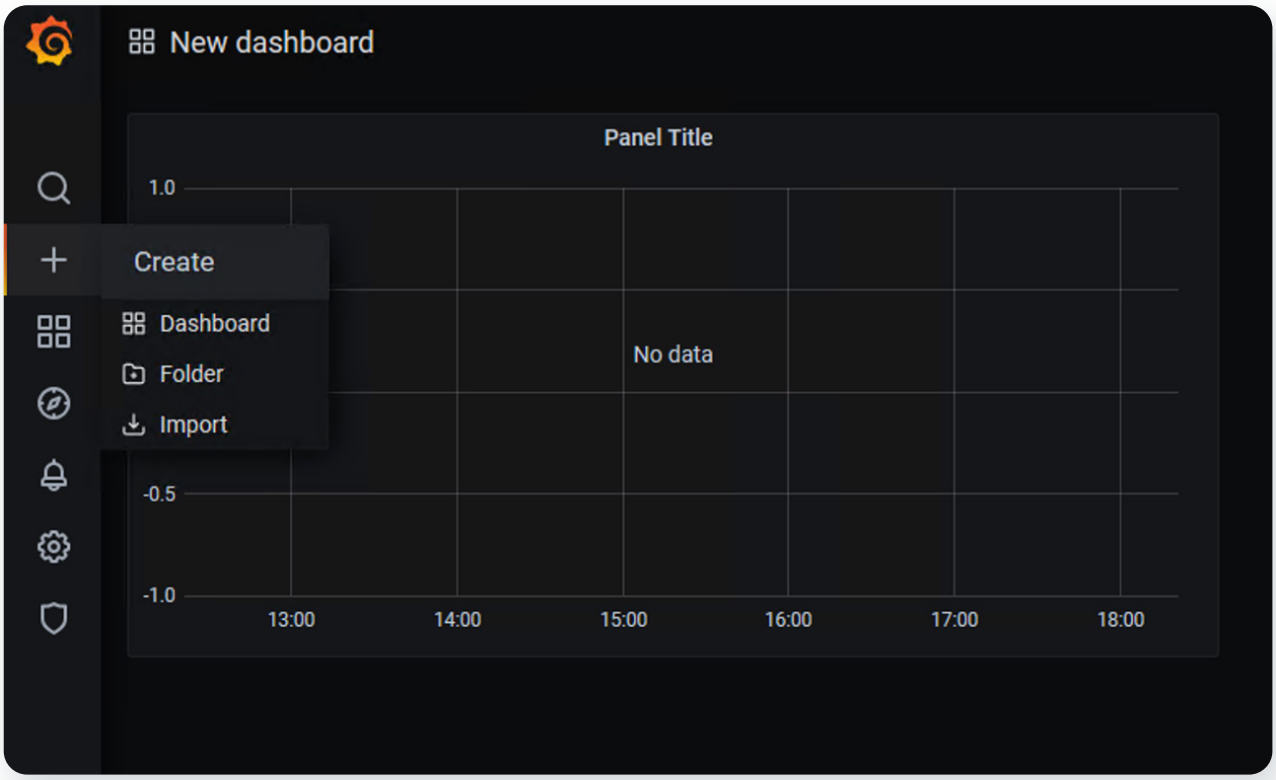



2. Configure the data source with the following details. Click on **Save and test**. Once it gives a green signal, the data source is ready for use. Paste the metrics token from step 2 into X-Auth-Token value field.
   **URL**: http://<Flex Appliance Console IP address>/metric
   **Access**: Server(default)
   **Skip TLS Verify**: true
   **Custom HTTP header**
   **Header**: X-Auth-Token
   **Value**: <authorization token from the /auth API>

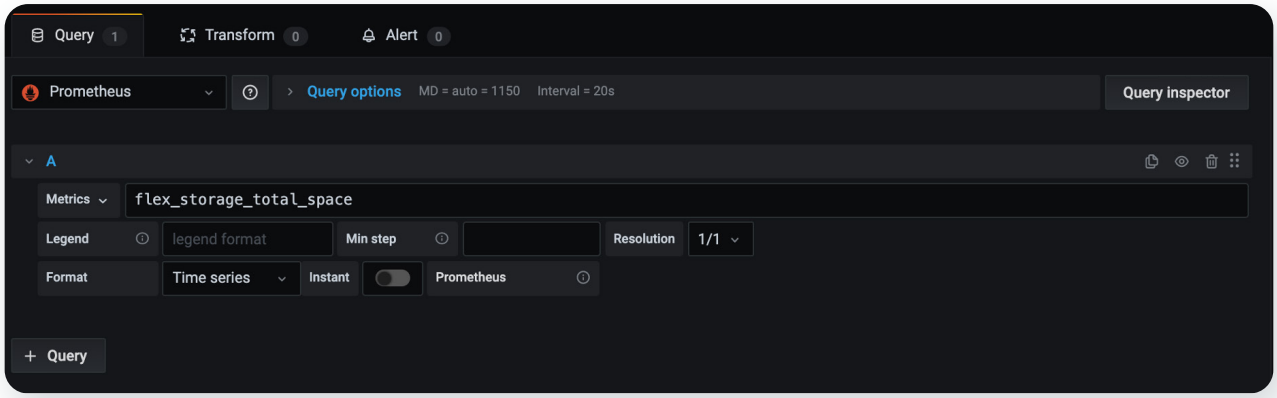


### Creating a Dashboard for Visualization of Data from Metric APIs

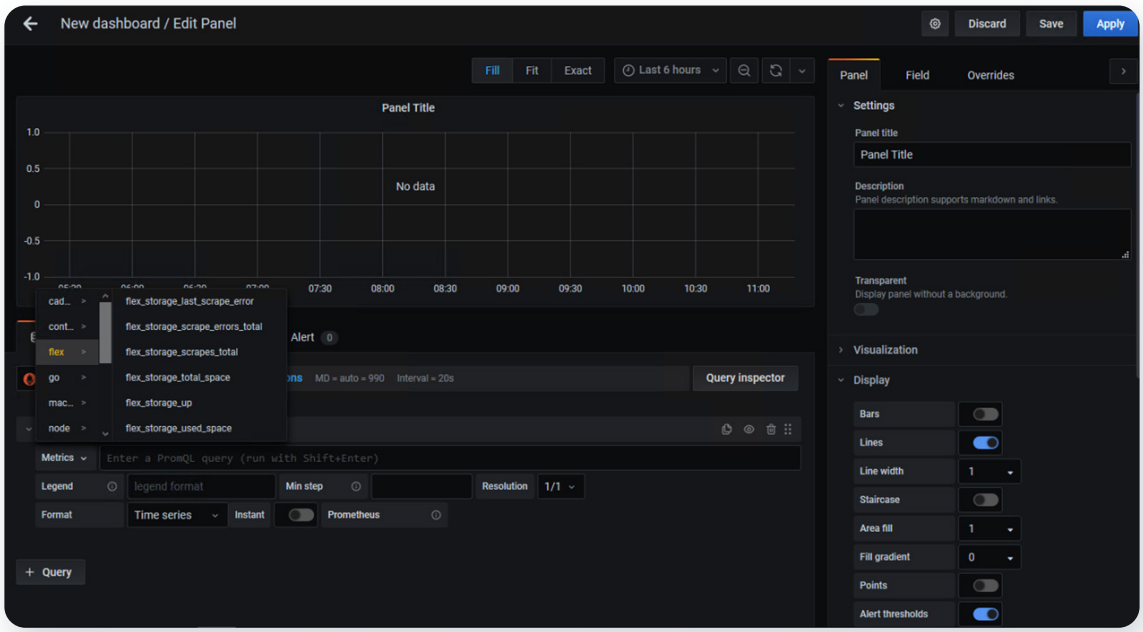
1. On the Grafana home page, click on the + symbol on the left panel to create a dashboard and add a panel.



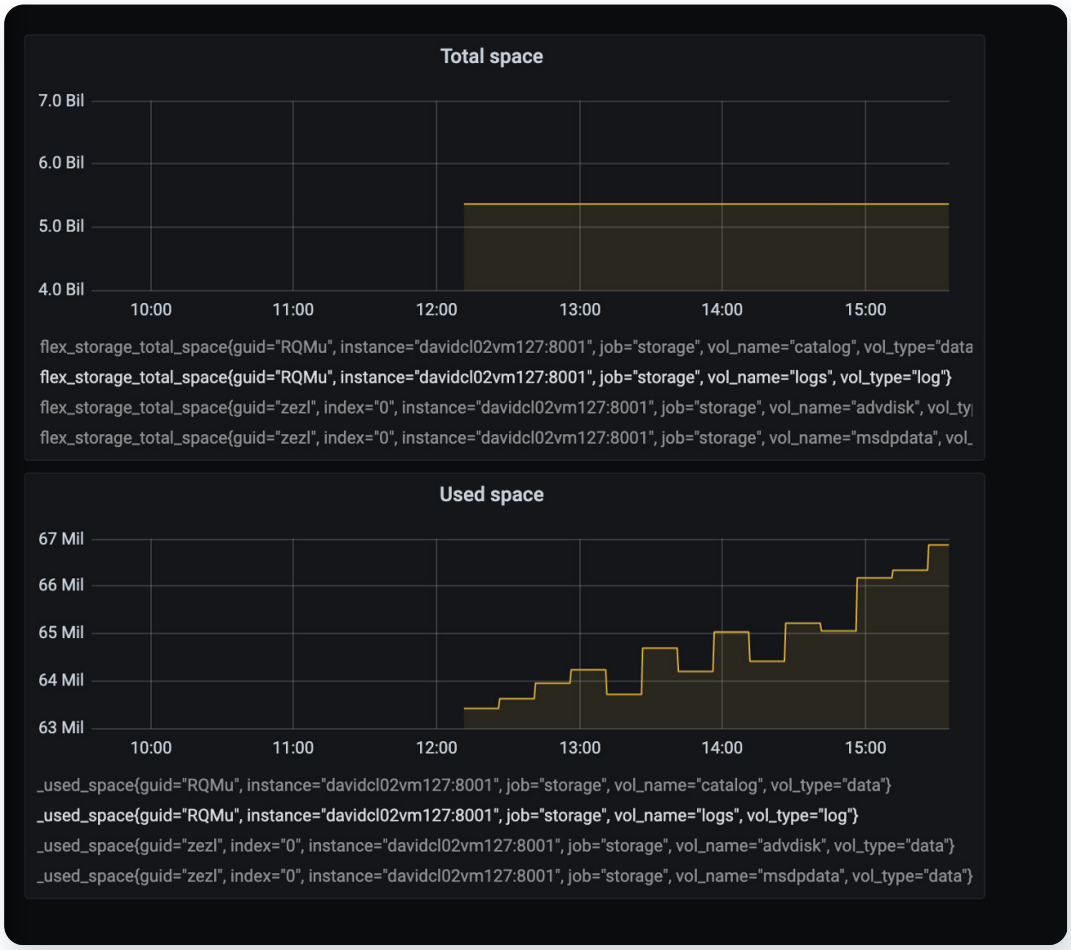

2. Select the query to the Prometheus data source.



3. Select predefined Flex metrics. For example, **flex_storage_total_space** or **flex_storage_total_used_space**.




4. Click on **Save** and **Apply**. You are directed to the dashboard overview page. On this page, you can see the panel that shows the data from the query metrics you selected. It should look similar to the following image:




The top panel is created with the query metric **flex_storage_total_space**, and the bottom panel is created with the query metric **flex_storage_used_space**.

## References

- Flex API documents

## Versions

| Flex Version | Date | Author | Key Updates |
|---|---|---|---|
| 2.0 | Dec 2020 | Rachel Zhu | Original document. |
| 2.0.1 | Jun 2021 | Rachel Zhu | Added static route |
| 2.1 | Nov 2021 | Rachel Zhu | API Metrics Token |




VERITAS

2625 Augustine Drive
Santa Clara, CA 95054
+1 (866) 837 4827
veritas.com

For global contact information visit:
veritas.com/company/contact





V1522 02/22