How to Install TIG Stack (Telegraf, InfluxDB, and Grafana) on Debian 12

The TIG (Telegraf, InfluxDB, and Grafana) Stack is an acronym for a platform of open-source tools to make the collection, storage, graphing, and alerting of system metrics easier. You can monitor and visualize metrics such as memory, disk space, logged-in users, system load, swap usage, uptime, running processes, etc. from one place. The tools used in the stack are as follows:

- **Telegraf** - is an open-source metrics collection agent for collecting and sending data and events from databases, systems, and IoT sensors. It supports various output plugins such as InfluxDB, Graphite, Kafka, etc to which it can send the collected data.
- **InfluxDB** - is an open-source time-series database written in the Go language. It is optimized for fast, high-availability storage and is suitable for anything involving large amounts of time-stamped data, including metrics, events, and real-time analytics.
- **Grafana** - is an open-source data visualization and monitoring suite. It supports various input plugins such as Graphite, ElasticSearch, InfluxDB, etc. It provides a beautiful dashboard and metric analytics allowing you to visualize and monitor any kind of system metrics and performance data.

In this tutorial, you will learn how to install and configure the TIG Stack on a single Debian 12 server.

Prerequisites

1. A server running Debian 12 with a minimum of 1 GB of RAM.
2. A non-sudo user with root privileges.
3. The uncomplicated Firewall(UFW) is enabled and running.
4. A Fully Qualified Domain Name (FQDN) like *grafana.example.com* pointing to your server.
5. An SMTP account with an email service like Amazon SES or Mailgun for getting email notifications for service alerts.
6. Ensure that everything is updated.

```
$ sudo apt update && sudo apt upgrade
```

7. A few essential packages are required for the tutorial and Craft CMS to run. Some of these will already be on your server.

```
$ sudo apt install curl wget nano software-properties-common dirmngr apt-transport-https ca-certificates lsb-release debian-archive-keyring gnupg2 ufw unzip -y
```

Step 1 - Configure Firewall

Before installing any packages, the first step is configuring the firewall to open ports for InfluxDB and Grafana.

Check the status of the firewall.

```
$ sudo ufw status
```

You should see something like the following.

```
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
```

Open port 8086 for InfluxDB and 3000 for the Grafana server.

```
$ sudo ufw allow 8086
$ sudo ufw allow 3000
```

Allow HTTP and HTTPs ports.

```
$ sudo ufw allow http
$ sudo ufw allow https
```

Check the status again to confirm.

```
$ sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
8086                       ALLOW       Anywhere
3000                       ALLOW       Anywhere
80/tcp                     ALLOW       Anywhere
443                        ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
8086 (v6)                  ALLOW       Anywhere (v6)
3000 (v6)                  ALLOW       Anywhere (v6)
80/tcp (v6)                ALLOW       Anywhere (v6)
443 (v6)                   ALLOW       Anywhere (v6)
```

Step 2 - Install InfluxDB

We will use InfluxDB's official repository to install it.

Download the InfluxDB GPG key.

```
$ wget -q https://repos.influxdata.com/influxdata-archive_compat.key
```

Import the GPG key into the server.

```
$ echo '393e8779c89ac8d958f81f942f9ad7fb82a25e133faddaf92e15b16e6ac9ce4c influxdata-archive_compat.key' | sha256sum -c && cat influxdata-archive_compat.key | gpg --dearmor | sudo tee /etc/apt/trusted.gpg.d/influxdata-a
influxdata-archive_compat.key: OK
```

Import the InfluxDB repository.

```
$ echo 'deb [signed-by=/etc/apt/trusted.gpg.d/influxdata-archive_compat.gpg] https://repos.influxdata.com/debian stable main' | sudo tee /etc/apt/sources.list.d/influxdata.list
```

Update the system's repository list.

```
$ sudo apt update
```

You have the option of installing InfluxDB 1.8.x or 2.0.x. However, it is better to use the latest version. Install InfluxDB.

```
$ sudo apt install influxdb2
```

Start the InfluxDB service.

```
$ sudo systemctl start influxdb
```

Check the status of the service.

```
$ sudo systemctl status influxdb
? influxdb.service - InfluxDB is an open-source, distributed, time series database
     Loaded: loaded (/lib/systemd/system/influxdb.service; enabled; preset: enabled)
     Active: active (running) since Tue 2024-01-02 02:39:41 UTC; 1s ago
       Docs: https://docs.influxdata.com/influxdb/
```

```
     Process: 5584 ExecStart=/usr/lib/influxdb/scripts/influxd-systemd-start.sh (code=exited, status=0/SUCCESS)
    Main PID: 5585 (influxd)
       Tasks: 8 (limit: 2299)
      Memory: 53.1M
         CPU: 735ms
      CGroup: /system.slice/influxdb.service
              ??5585 /usr/bin/influxd
.......
```

Step 3 - Create InfluxDB Database and User Credentials

To store the data from Telegraf, you need to set up the Influx database and user.

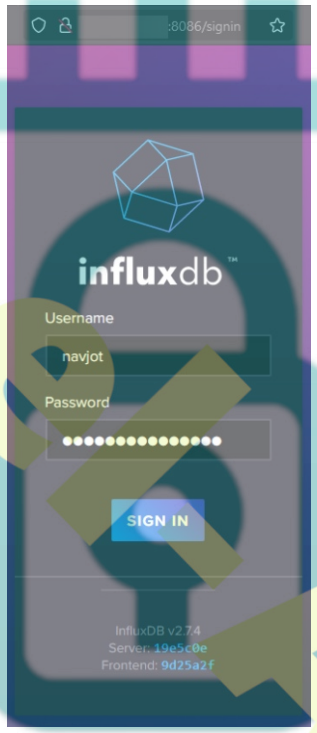
InfluxDB comes with a command-line tool named **influx** for interacting with the InfluxDB server. Think of *influx* as the *mysql* command-line tool.

Run the following command to perform the initial configuration for Influx.

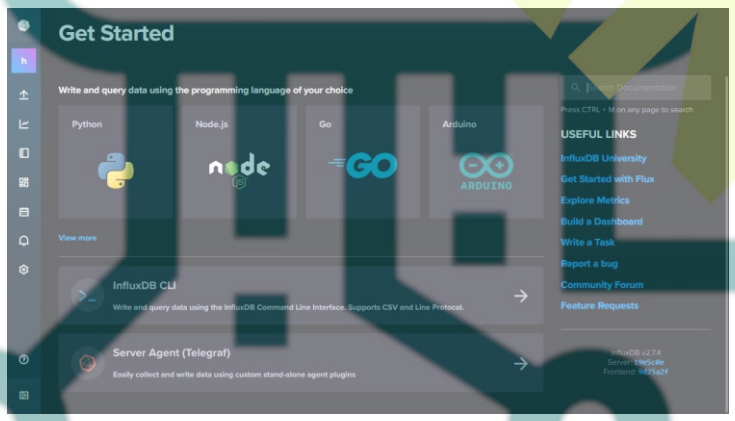
```
$ influx setup
> Welcome to InfluxDB 2.0!
? Please type your primary username navjot
? Please type your password **************
? Please type your password again **************
? Please type your primary organization name howtoforge
? Please type your primary bucket name tigstack
? Please type your retention period in hours, or 0 for infinite 360
? Setup with these parameters?
  Username:          navjot
  Organization:      howtoforge
  Bucket:            tigstack
  Retention Period:  360h0m0s
 Yes
User    Organization    Bucket
navjot  howtoforge      tigstack
```

You need to set up your initial username, password, organization name, the primary bucket name to store data, and the retention period in hours for that data. Your details are stored in the */home/username/.influxdbv2/configs* file.

You can also perform this setup by launching the URL *http://<serverIP>:8086/* in your browser. Once you have performed the initial setup, you can log in to the URL with the credentials created above.

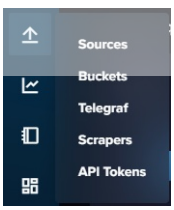


You should be greeted with the following dashboard.

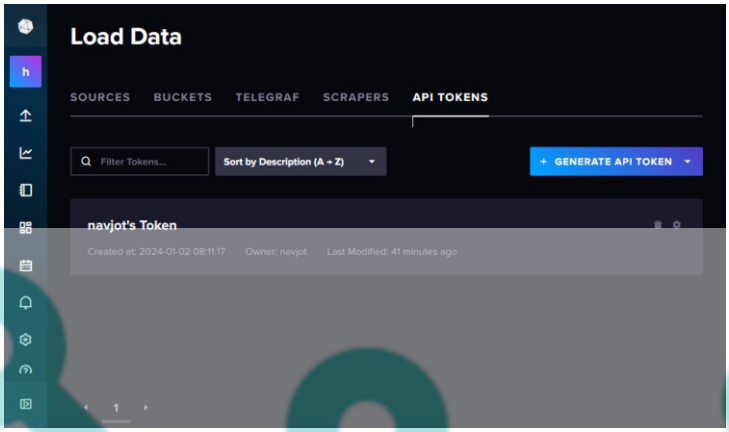


The initial setup process creates a default token that has full read and write access to all the organizations in the database. You need a new token for security purposes which will only connect to the organization and bucket we want to connect to.

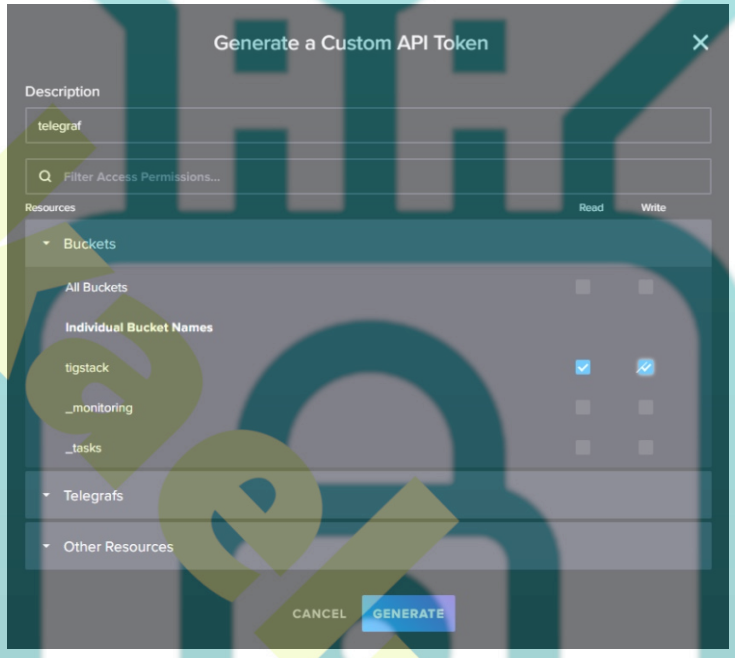
To create a new token, click on the following icon from the left sidebar and click the **API Tokens** link to proceed.



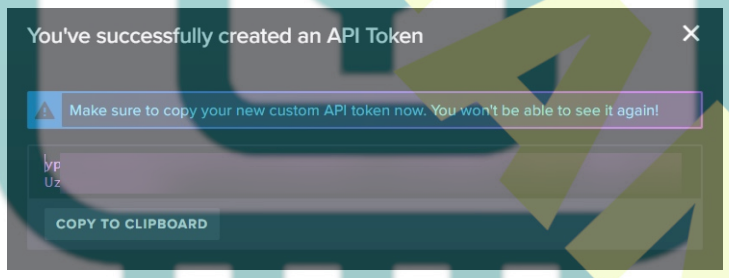
You will be taken to the **API Tokens** page. Here, you will see the default token that we created at the time of the initial configuration.



Click on the **Generate API Token** button and select the **Custom API Token** option to launch a new overlay popup. Give a name to the Token (*telegraf*) and expand the Resources section and select the default bucket we created under both the **Read** and **Write** sections.



Click **Generate** to finish creating the token. Click the **COPY TO CLIPBOARD** button to copy the token. The button might not work in some cases so make sure to confirm before dismissing the popup.



Save it for now since we will need it later on.

This completes the installation and configuration of InfluxDB. Next, we need to install Telegraf.

Step 4 - Install Telegraf

Telegraf and InfluxDB share the same repository. It means you can install Telegraf directly.

```
$ sudo apt install telegraf
```

Telegraf's service is enabled and started automatically during installation.

Telegraf is a plugin-driven agent and has 4 types of plugins:

1. **Input plugins** collect metrics.
2. **Processor plugins** transform, decorate, and filter metrics.
3. **Aggregator plugins** create and aggregate metrics.
4. **Output plugins** define the destinations where metrics are sent including InfluxDB.

Telegraf stores its configuration for all these plugins in the file */etc/telegraf/telegraf.conf*. The first step is to connect Telegraf to InfluxDB by enabling the *influxdb_v2* output plugin. Open the file */etc/telegraf/telegraf.conf* for editing.

```
$ sudo nano /etc/telegraf/telegraf.conf
```

Find the line *[[outputs.influxdb_v2]]* and uncomment out by removing the **#** in front of it. Edit out the code below it in the following way.

```
# # Configuration for sending metrics to InfluxDB 2.0
  [[outputs.influxdb_v2]]
#   ## The URLs of the InfluxDB cluster nodes.
#   ##
#   ## Multiple URLs can be specified for a single cluster, only ONE of the
#   ## urls will be written to each interval.
#   ##   ex: urls = ["https://us-west-2-1.aws.cloud2.influxdata.com"]
    urls = ["http://127.0.0.1:8086"]
#
#   ## Token for authentication.
    token = "$INFLUX_TOKEN"
#
#   ## Organization is the name of the organization you wish to write to.
    organization = "howtoforge"
#
```

```
#   ## Destination bucket to write into.
    bucket = "tigstack"
```

Paste the InfluxDB token value saved earlier in place of the `$INFLUX_TOKEN` variable in the code above.

Search for the line `INPUT PLUGINS` and you will see the following input plugins enabled by default.

```
# Read metrics about cpu usage
[[inputs.cpu]]
  ## Whether to report per-cpu stats or not
  percpu = true
  ## Whether to report total system cpu stats or not
  totalcpu = true
  ## If true, collect raw CPU time metrics
  collect_cpu_time = false
  ## If true, compute and report the sum of all non-idle CPU states
  report_active = false
  ## If true and the info is available then add core_id and physical_id tags
  core_tags = false


# Read metrics about disk usage by mount point
[[inputs.disk]]
  ## By default stats will be gathered for all mount points.
  ## Set mount_points will restrict the stats to only the specified mount points.
  # mount_points = ["/"]

  ## Ignore mount points by filesystem type.
  ignore_fs = ["tmpfs", "devtmpfs", "devfs", "iso9660", "overlay", "aufs", "squashfs"]

  ## Ignore mount points by mount options.
  ## The 'mount' command reports options of all mounts in parathesis.
  ## Bind mounts can be ignored with the special 'bind' option.
  # ignore_mount_opts = []


# Read metrics about disk IO by device
[[inputs.diskio]]
....
....

# Get kernel statistics from /proc/stat
[[inputs.kernel]]
  # no configuration


# Read metrics about memory usage
[[inputs.mem]]
  # no configuration

# Get the number of processes and group them by status
[[inputs.processes]]
  # no configuration


# Read metrics about swap memory usage
[[inputs.swap]]
  # no configuration


# Read metrics about system load & uptime
[[inputs.system]]
  # no configuration
```

You can configure additional input plugins depending upon your requirement including Apache Server, Docker containers, Elasticsearch, iptables firewall, Kubernetes, Memcached, MongoDB, MySQL, Nginx, PHP-fpm, Postfix, RabbitMQ, Redis, Varnish, Wireguard, PostgreSQL, etc.

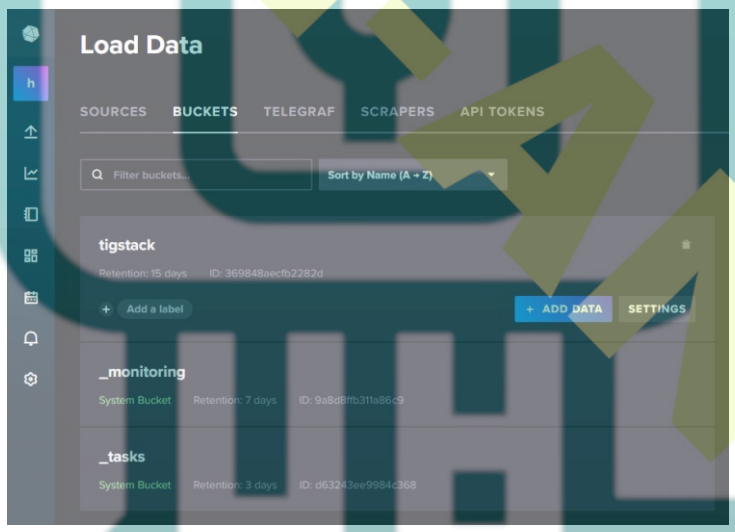
Once you are finished, save the file by pressing **Ctrl + X** and entering **Y** when prompted.

Restart the Telegraf service once you have finished applying the changes.

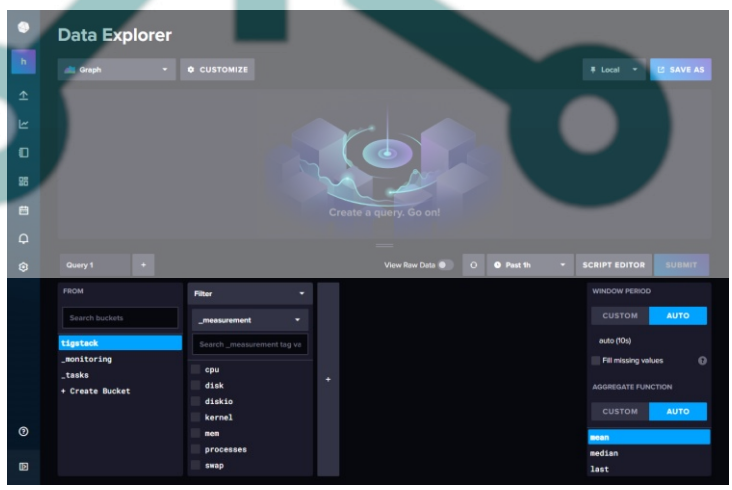
```
$ sudo systemctl restart telegraf
```

Step 5 - Verify if Telegraf stats are being stored in InfluxDB

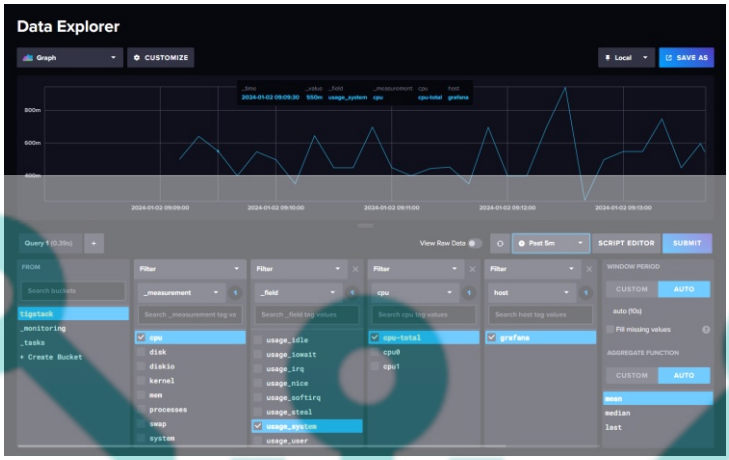
Before proceeding further, you need to verify if Telegraf stats are correctly collected and fed into the InfluxDB. Open the InfluxDB UI in your browser, click the second icon from the left sidebar, and select the **Buckets** menu.



Click on **tigstack** and you should be greeted with the following page.



Click on the bucket name and then click on one of the values in the *_measurement* filter, and keep clicking on other values as and when they appear. Once you are done, click the **Submit** button. You should see a graph at the top. You might need to wait for some time for the data to appear. We shifted the time interval from **Past 1 h** to **Past 5m** to generate a graph over a large area.



This should confirm that the data is being passed on correctly.

Step 6 - Install Grafana

We will use the official Grafana repository to install it. Import the Grafana GPG key.

```
$ sudo mkdir -p /etc/apt/keyrings/
$ wget -q -O - https://apt.grafana.com/gpg.key | gpg --dearmor | sudo tee /etc/apt/keyrings/grafana.gpg > /dev/null
```

Add the repository to your system.

```
$ echo "deb [signed-by=/etc/apt/keyrings/grafana.gpg] https://apt.grafana.com stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list
```

If you want to install Grafana beta, add the following repository instead.

```
$ echo "deb [signed-by=/etc/apt/keyrings/grafana.gpg] https://apt.grafana.com beta main" | sudo tee -a /etc/apt/sources.list.d/grafana.list
```

Update the system repository list.

```
$ sudo apt update
```

Install Grafana.

```
$ sudo apt install grafana
```

Start and Enable the Grafana service.

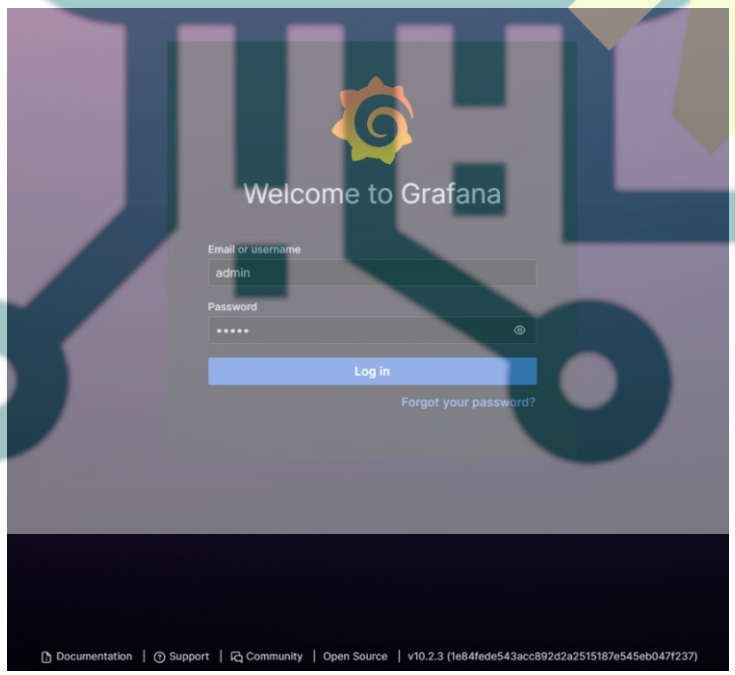
```
$ sudo systemctl enable grafana-server --now
```

Check the service status.

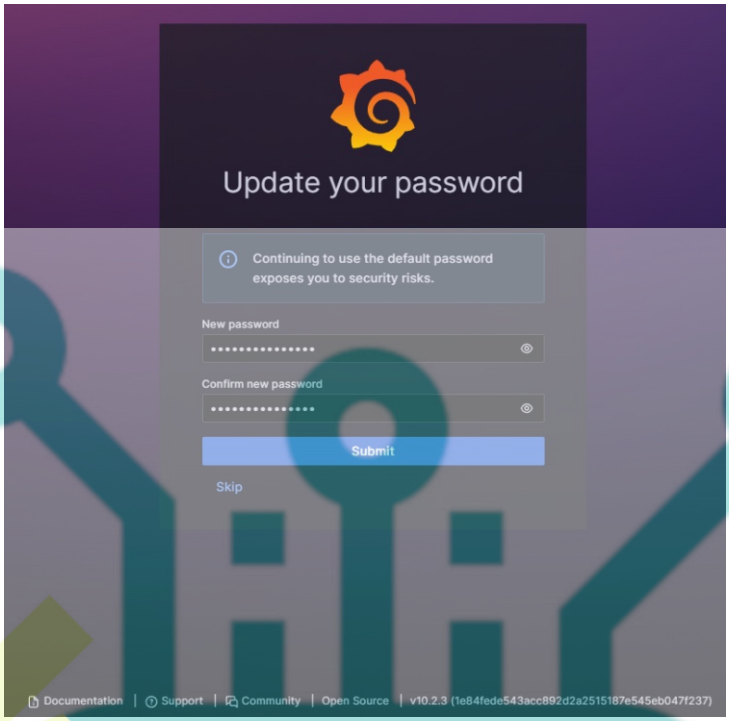
```
$ sudo systemctl status grafana-server
? grafana-server.service - Grafana instance
     Loaded: loaded (/lib/systemd/system/grafana-server.service; enabled; preset: enabled)
     Active: active (running) since Tue 2024-01-02 03:48:01 UTC; 3s ago
       Docs: http://docs.grafana.org
   Main PID: 8769 (grafana)
      Tasks: 7 (limit: 2299)
     Memory: 42.6M
        CPU: 1.804s
     CGroup: /system.slice/grafana-server.service
             ??8769 /usr/share/grafana/bin/grafana server --config=/etc/grafana/grafana.ini --pidfile=/run/grafana/grafana-server.pid --packaging=deb cfg:default.paths.logs=/var/log/grafana cfg:default.paths...
.......
```

Step 7 - Set up Grafana Data Source

Launch the URL *http://<serverIP>:3000* in your browser and the following Grafana login page should greet you.

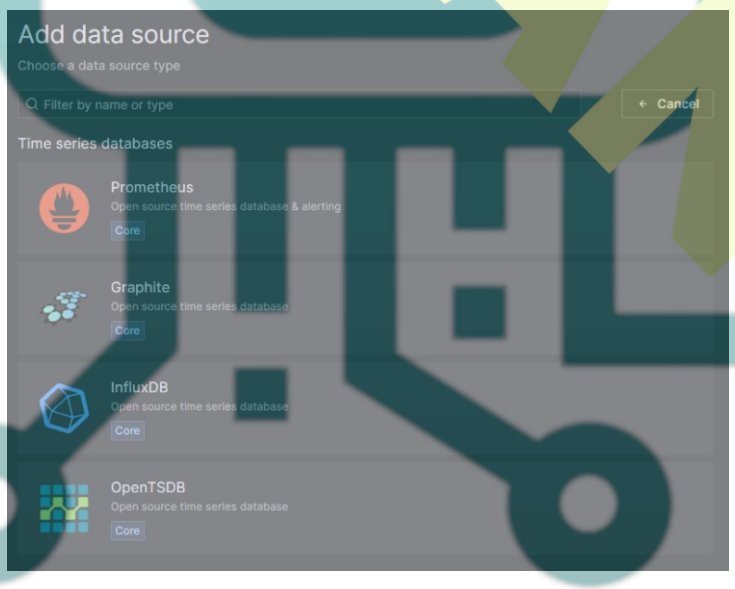


Login with the default username *admin* and password *admin*. Next, you need to set up a new default password.

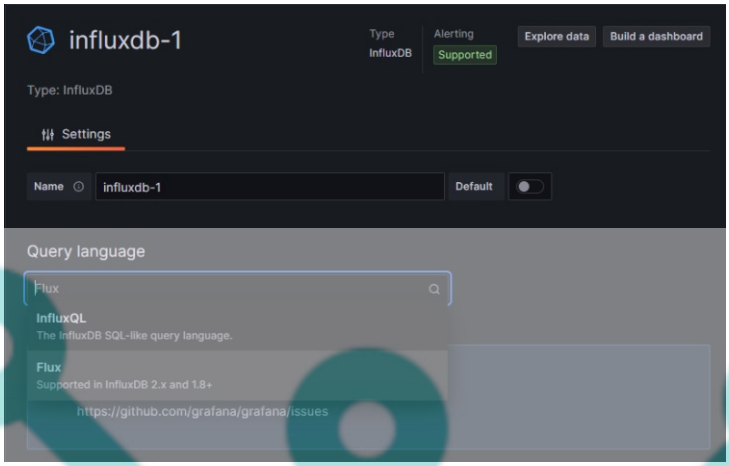


You shall be greeted with the following Grafana homepage. Click on the **Add your first data source** button.

Click the **InfluxDB** button.



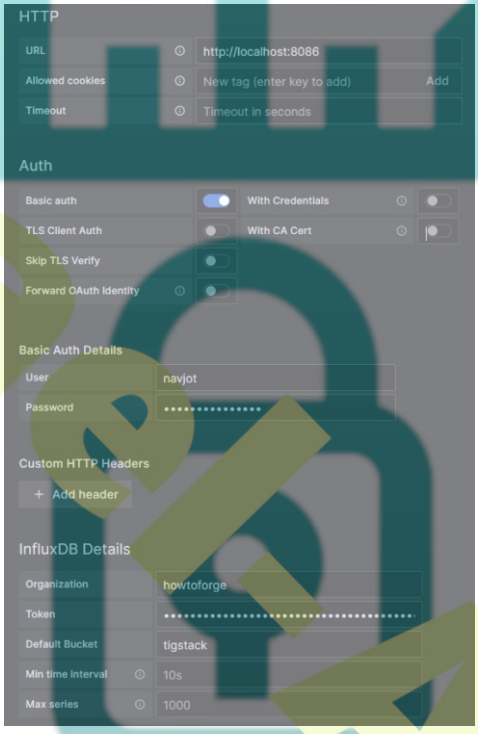
On the next page, select **Flux** from the dropdown menu as the query language. You can use **InfluxQL** as the query language, but it is more complicated to configure since it supports only InfluxDB v1.x by default. Flux supports InfluxDB v2.x and is easier to set up and configure.



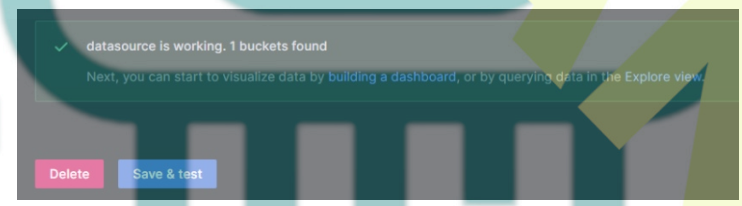
Enter the following values.

URL: *http://localhost:8086* **Basic Auth Details User:** navjot **Password:** *<yourinfluxdbpassword>*

InfluxDB Details Organization: howtoforge **Token:** *<influxdbtoken>* **Default Bucket:** tigstack

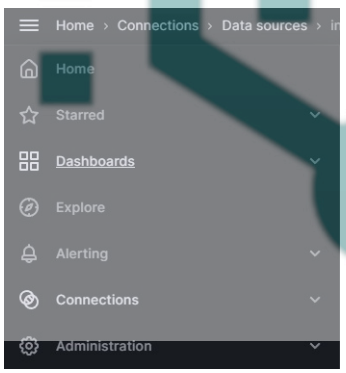


Click on the **Save and test** button and you should see a confirmation message verifying the setup is successful.

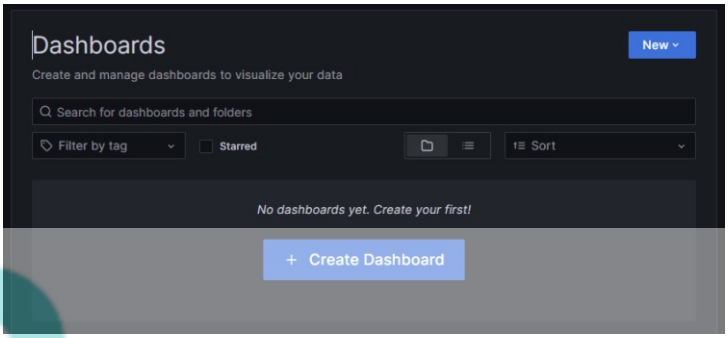


Step 8 - Set up Grafana Dashboards

The next step is to set up Grafana Dashboards. Click the hamburger menu to the left of **Home** and click **Dashboards** to open the Dashboard Create screen.



Click the **Create Dashboard** button to proceed.



On the next page, click on the **Add visualization** button to launch the overlay and click *influxdb-1* to select it as the data source.

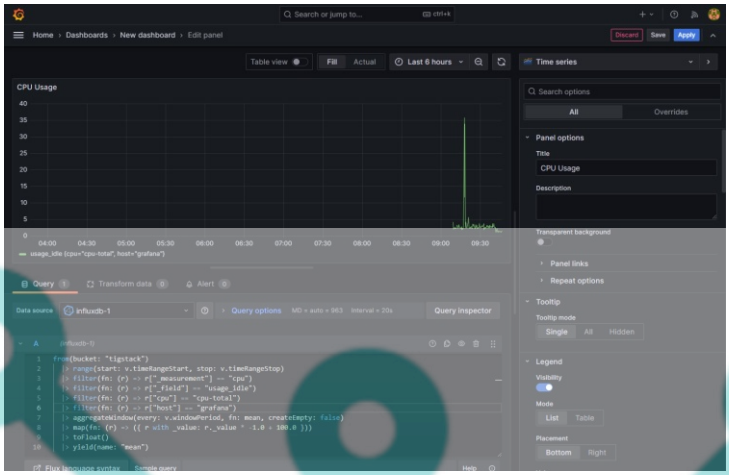
You will be taken to the following **Edit Panel** page.

Paste the following code in the Query Editor.

```
from(bucket: "NAMEOFYOUBUCKET")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "cpu")
  |> filter(fn: (r) => r["_field"] == "usage_idle")
  |> filter(fn: (r) => r["cpu"] == "cpu-total")
  |> filter(fn: (r) => r["host"] == "NAMEOFYOURHOST")
  |> aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)
  |> map(fn: (r) => ({ r with _value: r._value * -1.0 + 100.0 }))
  |> toFloat()
  |> yield(name: "mean")
```

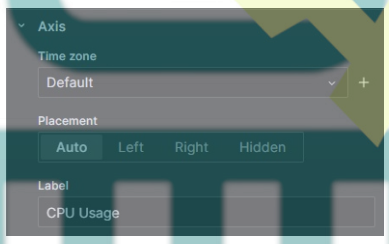
Use the bucket name that we used above. And the name of the host which you can retrieve from the file */etc/hostname*.

The above code will calculate the CPU Usage and generate a graph for it. Give the Panel a Title.

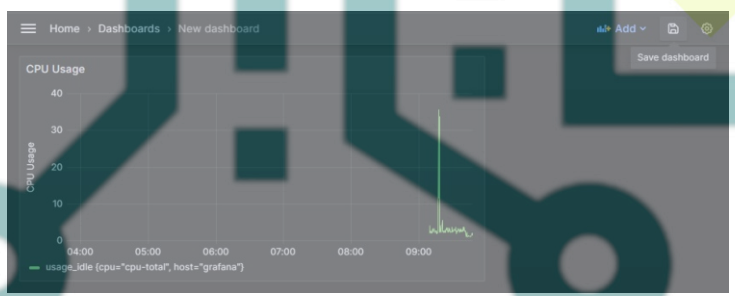


Click the **Query inspector** button and then click the **Refresh** button to verify if your query is working successfully. Click the cross icon to close the inspector.

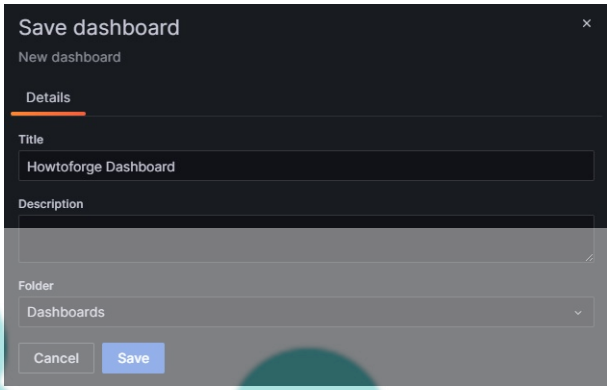
You can also name the axis by using the **Label** field on the right under the **Axis** section.



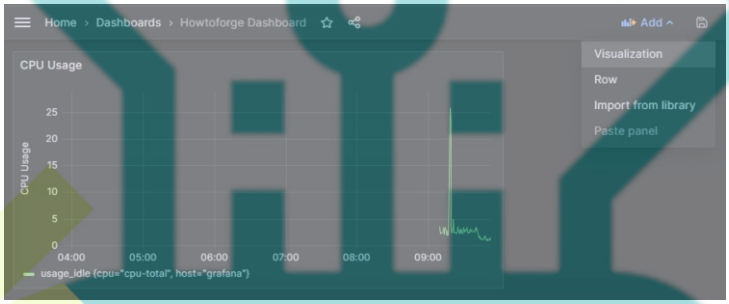
Click the **Apply** button to save the panel. Click the **Save Dashboard** button, once finished.



Give a name to the dashboard and click **Save** to finish.



It will open the dashboard and then click on the **Add Visualization** button to create another panel.



Repeat the process by creating another panel for RAM Usage.

```
from(bucket: "NAMEOFYOUBUCKET")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "mem")
  |> filter(fn: (r) => r["_field"] == "used_percent")
  |> filter(fn: (r) => r["host"] == "NAMEOFYOURHOST")
  |> aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)
  |> yield(name: "mean")
```

Use the following code for displaying the HDD Usage.

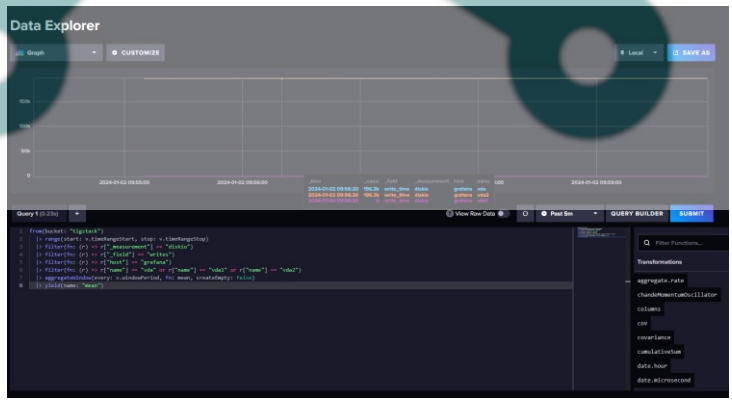
```
from(bucket: "NAMEOFYOURBUCKET")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "disk")
  |> filter(fn: (r) => r["_field"] == "used")
  |> filter(fn: (r) => r["path"] == "/")
  |> filter(fn: (r) => r["host"] == "NAMEOFYOURHOST")
  |> aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)
  |> map(fn: (r) => ({ r with _value: r._value / 1000000.0 }))
  |> toFloat()
  |> yield(name: "mean")
```

You can create an unlimited number of panels.

The above code is based on the Flux Scripting language. Fortunately, you don't need to learn the language to write queries. You can generate the query from the InfluxDB URL. Even though learning the language can benefit in optimizing the queries.

You need to go back to the InfluxDB dashboard and open the **Explore page** to get the query.

Click on the bucket name and then click on one of the values in the *_measurement* filter, and keep clicking on other values as and when they appear. Once you are done, click the **Script Editor** button and you should see the following page. The graph should also be updated.



Copy the query shown and you can now use it in the Grafana dashboard to build your graphs.

Step 9 - Configure Alerts and Notifications

The primary use of setting up monitors is to get alerts on time when the value goes beyond a certain threshold.

The first step is to set the destination where you want to get alerts. You can receive notifications via Email, Slack, Kafka, Google Hangouts Chat, Microsoft Teams, Telegram, etc.

We will be enabling email notifications for our tutorial. To set up Email notifications, we need to configure the SMTP service first. Open the `/etc/grafana/grafana.ini` file for configuring SMTP.

```
$ sudo nano /etc/grafana/grafana.ini
```

Find the following line `[smtp]` in it. Uncomment the following lines and enter the values for the custom SMTP server.

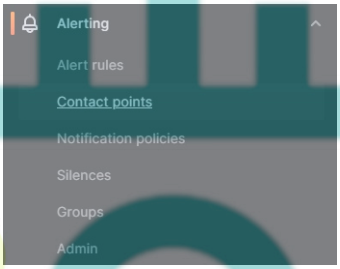
```
[smtp]
enabled = true
host = email-smtp.us-west-2.amazonaws.com:587
user = YOURUSERNAME
# If the password contains # or ; you have to wrap it with triple quotes. Ex """#password;"""
password = YOURUSERPASSWORD
;cert_file =
;key_file =
;skip_verify = false
from_address = user@example.com
from_name = HowtoForge Grafana
# EHLO identity in SMTP dialog (defaults to instance_name)
;ehlo_identity = dashboard.example.com
# SMTP startTLS policy (defaults to 'OpportunisticStartTLS')
;startTLS_policy = NoStartTLS
```

Save the file by pressing **Ctrl + X** and entering **Y** when prompted.

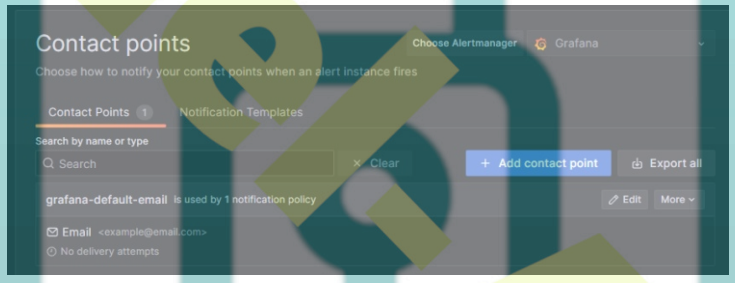
Restart the Grafana server to apply the settings.

```
$ sudo systemctl restart grafana-server
```

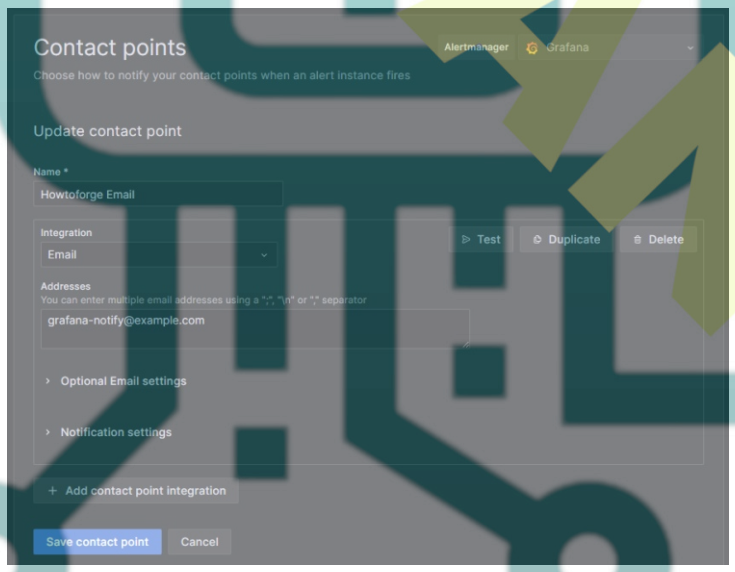
Open the Grafana page, click on the Alert icon, and click on **Contact points**.



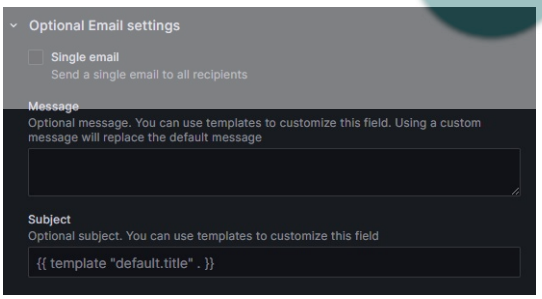
Grafana automatically creates and sets up a default email contact point that must be configured with the correct email address. Click on the edit button across the `grafana-default-email` contact point.



Enter the details to set up the Email notification channel.

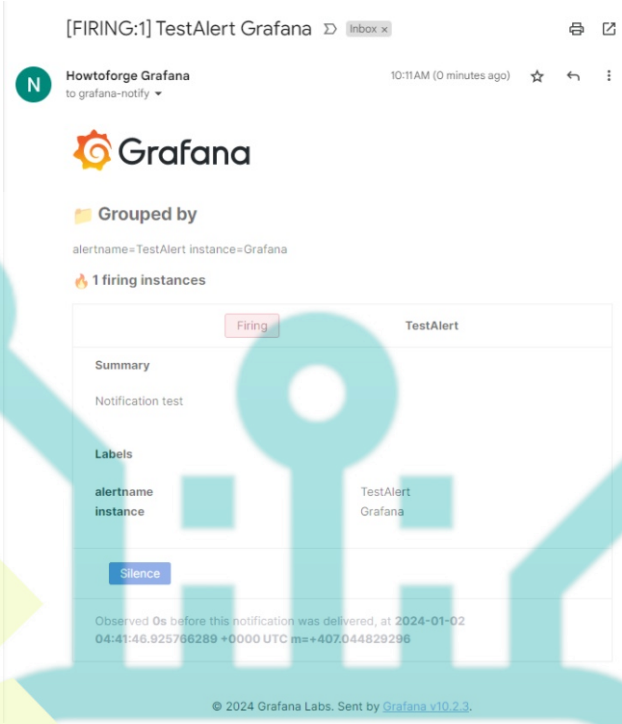


If you want to send an additional message, click the **Optional Email settings** link and enter the message.



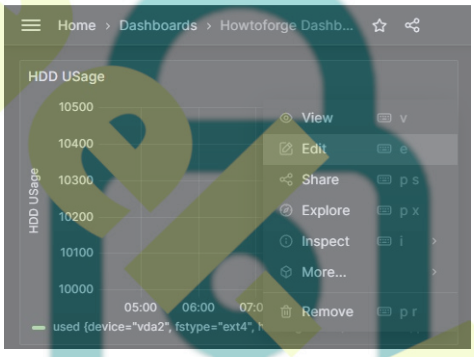
Click **Test** to open the popup and then click the **Send test notification** button to see if the email settings are working. Click **Save contact point** when finished.

You should get the following email confirming the settings.

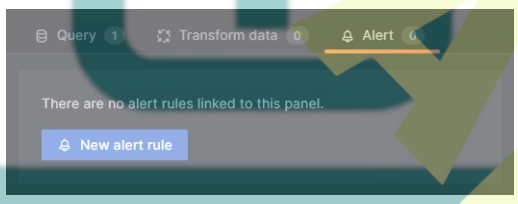


Now that we have set up notification channels, we need to set up alerts on when to receive these emails. To set up the alerts, you need to go back to the dashboard panels.

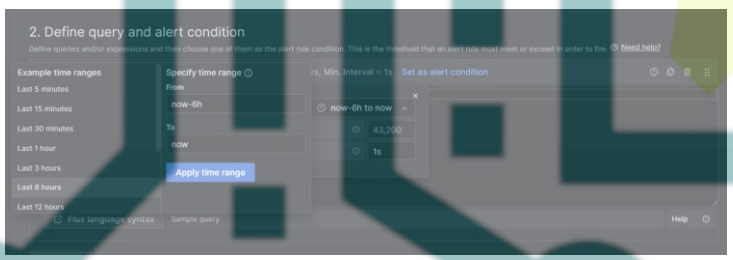
Go back to the **Dashboards** screen. Click on the dashboard we just created and you will get its homepage with different panels. To edit the panel, click on the name of the panel, and a dropdown menu will pop up. Click on the **Edit** link to proceed.



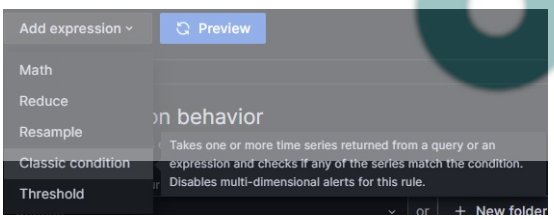
Click on the **Alert Panel** and click on the **New alert rule** button to set up a new alert. We are creating an alert for the CPU usage panel.

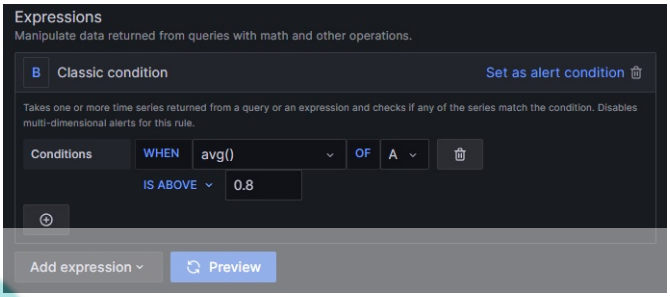


You can now configure the conditions under which Grafana will send the alert. Click the **Options** link dropdown menu and select the default time range (now-6h to now) to change the time range to the **Last 15 minutes** which means it will check from 15 minutes ago to now.

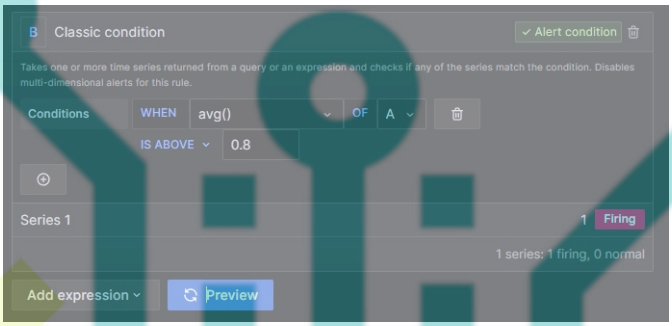


By default, the selected alert type is Grafana managed alert. There are two expressions selected by default. Delete them by pressing the trash button against them. Select the **Add expression** dropdown and select the **Classic condition** as the expression type.





Click the **Set as alert condition** to select the chosen expression for sending the alerts. Doing so will change the expression box as shown below.

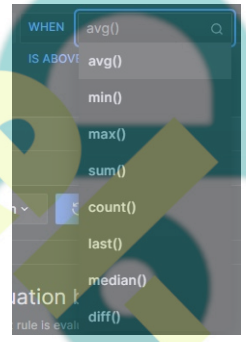


Conditions

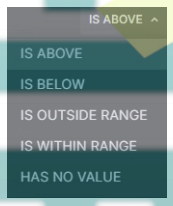
Grafana works on a query of the following format to determine when to launch an alert.

```
avg() OF query(A) IS ABOVE 0.8
```

- **avg()** controls how the value for each series should be reduced to a comparable value against the threshold. You can click on the function name to select a different function such as avg(), min(), max(), sum(), count(), etc.

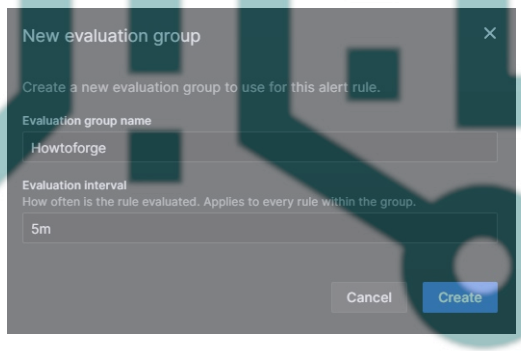


- **query(A)** The letter in the bracket defines what query to execute from the **Metrics** tab.
- **IS BELOW 14** Defines the type of threshold and the threshold value. You can click on **IS BELOW** to select a different threshold type.

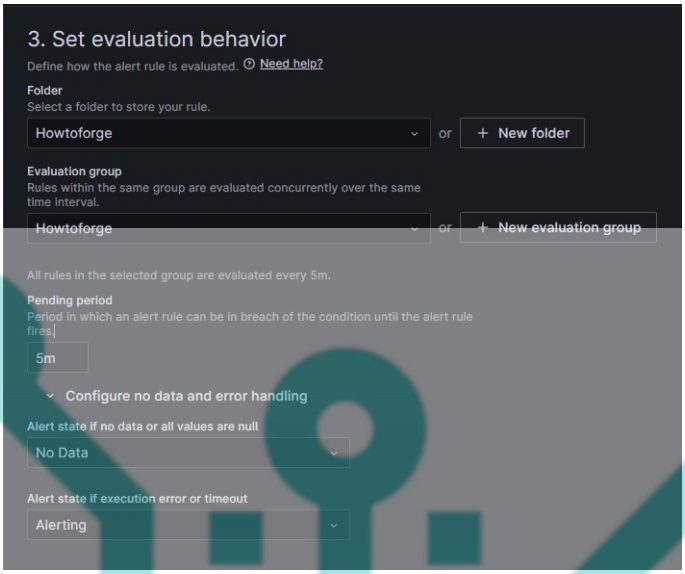


You can add a second condition below it by clicking on the **+** button beneath the first condition. Currently, you can only use **AND** and **OR** operators between multiple conditions.

Next, we will set the evaluation behavior. Click the **New folder** button to create a folder to store your rules. Click the **New evaluation group** button to create a group to club rules which will get evaluated after the same time interval. Set the time interval as 5m while creating the group.



Once finished, the page should look like the following. Set the **Alert state if execution error or timeout** to **Alerting**.



Rule

- **Name** - Enter a descriptive name for the alert
- **Folder** - Create or select a pre-existing folder to store your notification rule.
- **Group** - Enter a name for your alert group. Alerts in a single group are evaluated after the same time interval.
- **Pending** - Specify how often Grafana should evaluate the alert. It is also called an evaluation interval. You can set any value you desire here.

No Data & Error Handling

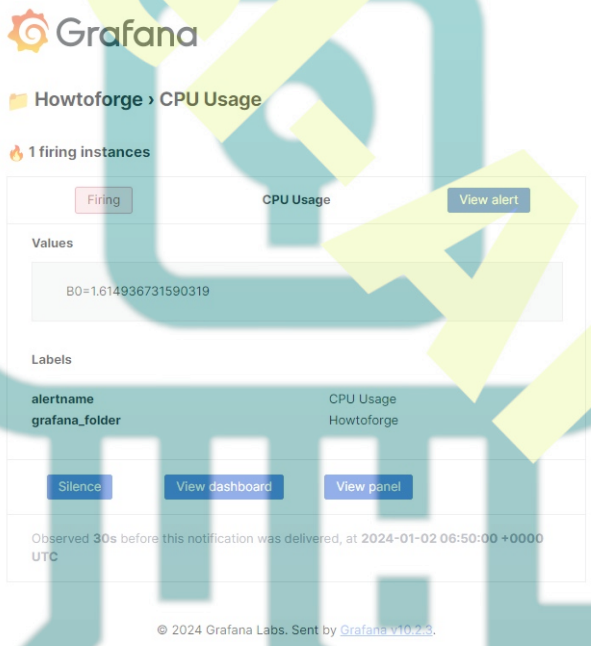
You can configure how Grafana should handle queries that return no data or only null values using the following conditions:

1. **No Data** - Set the rule state to *NoData*
2. **Alerting** - Set the rule state to *Alerting*
3. **Ok** - Set the alert rule state to *Ok*, as you will get an alert even if things are okay.

You can tell Grafana how to handle execution or timeout errors.

1. **Alerting** - Set the rule state to *Alerting*
2. **Ok** - Set the alert rule state to *Ok*, as you will get an alert even if things are okay.
3. **Error** - Set the alert rule state to Error to indicate there is an issue.

Once you are finished, click the button **Preview alerts** to see if everything is working fine. Click the **Save rule and exit** button on the top right to finish adding the alert. You should now start getting alerts on your email. Following is an example of one such email.



Step 10 - Install Nginx

Debian 12 ships with an older version of Nginx. You need to download the official Nginx repository to install the latest version.

Import Nginx's signing key.

```
$ curl https://nginx.org/keys/nginx_signing.key | gpg --dearmor \
    | sudo tee /usr/share/keyrings/nginx-archive-keyring.gpg >/dev/null
```

Add the repository for Nginx's mainline version.

```
$ echo "deb [signed-by=/usr/share/keyrings/nginx-archive-keyring.gpg] \
http://nginx.org/packages/mainline/debian `lsb_release -cs` nginx" \
    | sudo tee /etc/apt/sources.list.d/nginx.list
```

Update the system repositories.

```
$ sudo apt update
```

Install Nginx.

```
$ sudo apt install nginx
```

Verify the installation. On Debian systems, the following command will only work with *sudo*.

```
$ sudo nginx -v
nginx version: nginx/1.25.3
```

Start the Nginx server.

```
$ sudo systemctl start nginx
```

Check the service status.

```
$ sudo systemctl status nginx
? nginx.service - nginx - high performance web server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; preset: enabled)
     Active: active (running) since Tue 2024-01-02 09:21:10 UTC; 5s ago
       Docs: https://nginx.org/en/docs/
    Process: 12964 ExecStart=/usr/sbin/nginx -c /etc/nginx/nginx.conf (code=exited, status=0/SUCCESS)
   Main PID: 12965 (nginx)
      Tasks: 3 (limit: 2299)
     Memory: 2.9M
        CPU: 86ms
     CGroup: /system.slice/nginx.service
             ??12965 "nginx: master process /usr/sbin/nginx -c /etc/nginx/nginx.conf"
             ??12966 "nginx: worker process"
             ??12967 "nginx: worker process"

Jan 02 09:21:10 grafana systemd[1]: Starting nginx.service - nginx - high performance web server...
Jan 02 09:21:10 grafana systemd[1]: Started nginx.service - nginx - high performance web server.
```

Step 11 - Install SSL

We need to install Certbot to generate the SSL certificate. You can either install Certbot using Debian's repository or grab the latest version using the Snapd tool. We will be using the Snapd version.

Debian 12 comes doesn't come with Snapd installed. Install Snapd package.

```
$ sudo apt install snapd
```

Run the following commands to ensure that your version of Snapd is up to date.

```
$ sudo snap install core && sudo snap refresh core
```

Install Certbot.

```
$ sudo snap install --classic certbot
```

Use the following command to ensure that the Certbot command can be run by creating a symbolic link to the `/usr/bin` directory.

```
$ sudo ln -s /snap/bin/certbot /usr/bin/certbot
```

Verify if Certbot is functioning correctly.

```
$ certbot --version
certbot 2.8.0
```

Run the following command to generate an SSL Certificate.

```
$ sudo certbot certonly --nginx --agree-tos --no-eff-email --staple-ocsp --preferred-challenges http -m name@example.com -d grafana.example.com
```

The above command will download a certificate to the `/etc/letsencrypt/live/grafana.example.com` directory on your server.

Generate a **Diffie-Hellman group** certificate.

```
$ sudo openssl dhparam -dsaparam -out /etc/ssl/certs/dhparam.pem 4096
```

Check the Certbot renewal scheduler service.

```
$ systemctl list-timers
```

You will find `snap.certbot.renew.service` as one of the services scheduled to run.

```
NEXT                         LEFT          LAST                         PASSED    UNIT                         ACTIVATES
-----------------------------------------------------------------------------------------------------------------------------
Tue 2024-01-02 15:24:52 UTC 6h left       Mon 2024-01-01 15:24:52 UTC 17h ago   systemd-tmpfiles-clean.timer systemd-tmpfiles-clean.service
Tue 2024-01-02 20:05:29 UTC 10h left      Tue 2024-01-02 09:02:47 UTC 21min ago apt-daily.timer               apt-daily.service
Tue 2024-01-02 20:35:00 UTC 11h left      -                            -        snap.certbot.renew.timer     snap.certbot.renew.service
```

Do a dry run of the process to check whether the SSL renewal is working fine.

```
$ sudo certbot renew --dry-run
```

If you see no errors, you are all set. Your certificate will renew automatically.

Step 12 - Configure Nginx for Grafana and InfluxDB

Open the file `/etc/nginx/nginx.conf` for editing.

```
$ sudo nano /etc/nginx/nginx.conf
```

Add the following line before the line `include /etc/nginx/conf.d/*.conf;`.

```
server_names_hash_bucket_size 64;
```

Save the file by pressing **Ctrl + X** and entering **Y** when prompted.

Create and open the file `/etc/nginx/conf.d/grafana.conf` for editing.

```
$ sudo nano /etc/nginx/conf.d/grafana.conf
```

Paste the following code in it. Replace `grafana.example.com` with your domain name.

```
map $http_upgrade $connection_upgrade {
  default upgrade;
  '' close;
}

server {

    listen 443 ssl reuseport;
    listen [::]:443 ssl reuseport;

    http2 on;

    server_name grafana.example.com;

    access_log  /var/log/nginx/grafana.access.log;
    error_log   /var/log/nginx/grafana.error.log;

    ssl_certificate     /etc/letsencrypt/live/grafana.example.com/fullchain.pem;
    ssl_certificate_key  /etc/letsencrypt/live/grafana.example.com/privkey.pem;
    ssl_trusted_certificate /etc/letsencrypt/live/grafana.example.com/chain.pem;
```

```
    ssl_session_timeout  5m;
    ssl_session_cache shared:MozSSL:10m;
    ssl_session_tickets off;

    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_prefer_server_ciphers on;
    ssl_ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DH
    ssl_ecdh_curve X25519:prime256v1:secp384r1:secp521r1;
    ssl_stapling on;
    ssl_stapling_verify on;
    ssl_dhparam /etc/ssl/certs/dhparam.pem;

    resolver 1.1.1.1 1.0.0.1 [2606:4700:4700::1111] [2606:4700:4700::1001] valid=60s;
    resolver_timeout 2s;

    location / {
        proxy_set_header Host $http_host;
        proxy_pass http://localhost:3000;
    }

    location /api/live {
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection $connection_upgrade;
        proxy_set_header Host $http_host;
        proxy_pass http://localhost:3000;
        }

    location /influxdb/ {
        access_log /var/log/nginx/influx.access.log;
        error_log /var/log/nginx/influx.error.log;
        rewrite ^/influxdb$ /influxdb/ permanent;
        rewrite ^/influxdb/(.*)$ /$1 break;
        proxy_cookie_path ~*^/api /influxdb/api;
        proxy_connect_timeout 600s;
        proxy_http_version 1.1;
        proxy_pass http://localhost:8086;
        proxy_read_timeout 600s;
        proxy_send_timeout 600s;
        proxy_set_header Authorization $http_authorization;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $http_host;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto https;
        proxy_set_header X-Real-IP $remote_addr;
        sub_filter '<base href="/">' '<base href="/influxdb/">';
        sub_filter 'src="/' 'src="/influxdb/';
        sub_filter 'href="/' 'href="/influxdb/';
        sub_filter 'data-basepath="' 'data-basepath="/influxdb/';
        sub_filter 'n.p="/"' 'n.p="/influxdb/"';
        sub_filter 'o.p="/"' 'o.p="/influxdb/"';
        sub_filter '/api/' '/influxdb/api/';
        sub_filter 'api/v2/query' 'influxdb/api/v2/query';
        sub_filter '/health' '/influxdb/health';
        sub_filter types text/css text/javascript application/javascript application/json;
        sub_filter_once off;
        }
}
# enforce HTTPS
server {
    listen 80;
    listen [::]:80;
    server_name grafana.example.com;
    return 301 https://$host$request_uri;
}
```

Save the file by pressing **Ctrl + X** and entering **Y** when prompted.

Verify your Nginx configuration.

```
$ sudo nginx -t
```

Restart the Nginx server.

```
$ sudo systemctl restart nginx
```

Configure Telegraf for HTTPS

Open the file `/etc/telegraf/telegraf.conf` for editing.

```
$ sudo nano /etc/telegraf/telegraf.conf
```

Search for the section `[[outputs.influxdb_v2]]` and change the value of the URL to `https://grafana.nspeaks.com/influxdb` so that the data between InfluxDB and Telegraf is secured.

```
urls = ["https://grafana.example.com/influxdb"]
```

Save the file by pressing **Ctrl + X** and entering **Y** when prompted.

Restart the Telegraf service.

```
$ sudo systemctl restart telegraf
```

Configure Grafana for HTTPS

Next, we need to configure Grafana for HTTPS access. Open the `/etc/grafana/grafana.ini` file for editing.

```
$ sudo nano /etc/grafana/grafana.ini
```

Find the `[server]` section and change the `domain` variable, and `root_url` as follows.

```
# The public facing domain name used to access grafana from a browser
;domain = localhost
domain = grafana.example.com

# Redirect to correct domain if host header does not match domain
# Prevents DNS rebinding attacks
;enforce_domain = true

# The full public facing url you use in browser, used for redirects and emails
# If you use reverse proxy and sub path specify full url (with sub path)
;root_url = %(protocol)s://%(domain)s:%(http_port)s/
root_url = %(protocol)s://%(domain)s
```

Save the file by pressing **Ctrl + X** and entering **Y** when prompted.

Restart the Grafana Server.

```
$ sudo systemctl restart grafana-server
```

Close Firewall Ports for InfluxDB and Grafana

You should also close the InfluxDB and Grafana ports.

```
$ sudo ufw delete allow 8086
$ sudo ufw delete allow 3000
```

Grafana should be accessible at `https://grafana.example.com` and InfluxDB UI and API should both be accessible at the URL `https://grafana.example.com/influxdb`.

Conclusion

This concludes the tutorial about installing and configuring the TIG Stack on a Debian 12 server. If you have any questions, post them in the comments below.