BitPath-DevKit

Installing TIG Stack

RADIOSTUDIO

1 Table of Contents

2 Revision History

Version No	Date	Author	Change Log
1.0	29/04/2024	RadioStudio	

3 Install InfluxDB

This guide assumes that you have already provisioned a VM with AWS or GCP. Let's install the latest version of InfluxDB first.

Find the right download from this page and install it using the instructions mentioned on that page.

Enable the service and start it

```
systemctl enable influxdb.service && systemctl start influxdb.service
```

You should now be able to configure it for the first time by visiting the public IP of your VM on port 8086.
Example: http://52.2.96.251:8086/signin

3.1 Enabling HTTPS

```
sudo openssl req -x509 -nodes -newkey rsa:2048 \
  -keyout /etc/ssl/influxdb-selfsigned.key \
  -out /etc/ssl/influxdb-selfsigned.crt \
  -days 1000

######

sudo chmod 644 /etc/ssl/influxdb-selfsigned.crt
sudo chmod 600 /etc/ssl/influxdb-selfsigned.key

#####
chown influxdb:influxdb /etc/ssl/influxdb-selfsigned.*

#####

vi /etc/influxdb/config.toml
###
Add these two values below to the configuration file.
###

tls-cert = "/etc/ssl/influxdb-selfsigned.crt"
tls-key = "/etc/ssl/influxdb-selfsigned.key"
```

###

The final config.toml should look like this.

###

bolt-path = "/var/lib/influxdb/influxd.bolt"

engine-path = "/var/lib/influxdb/engine"

tls-cert = "/etc/ssl/influxdb-selfsigned.crt"

tls-key = "/etc/ssl/influxdb-selfsigned.key"

###

Change the config file to point to /etc/influxdb/config.toml

Just go to **vi /etc/default/influxdb2 and uncomment the line that points to config.toml and comment the old config file**

#INFLUXD_CONFIG_PATH=/etc/influxdb/influxdb.conf

INFLUXD_CONFIG_PATH=/etc/influxdb/config.toml

Save and restart the influx service.

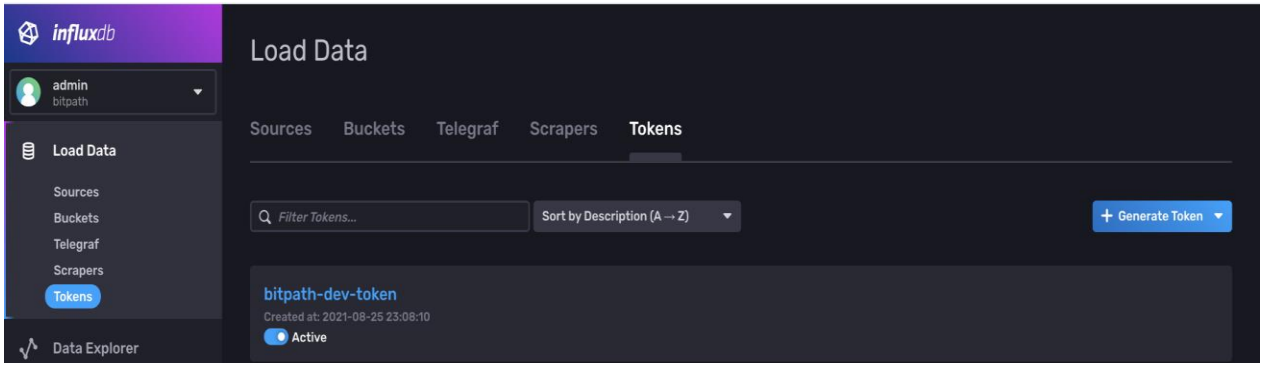
####

service influxd restart

Choose a username and password for the first-time setup. Example:

- Enter a Username for your initial user. - **influxadmin**
- Enter a Password and Confirm Password for your user. - **your password**
- Enter your initial Organization Name (this is a logical unit for your datasets) - **bitpath**
- Enter your initial Bucket Name (this is like a database) - **bitpath-dev**

Once connected, go to tokens and create one. Grafana needs a token to authenticate and talk to InfluxDB. Create a token and save it somewhere for later use.



4 Install and Configure Grafana

Install and configure Grafana. This is often done on the same VM as InfluxDB.

```
sudo vi /etc/yum.repos.d/grafana.repo
```

Add the following:

```
[grafana]
name=grafana
baseurl=https://packages.grafana.com/oss/rpm
repo_gpgcheck=1
enabled=1
gpgcheck=1
gpgkey=https://packages.grafana.com/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
```

Install

```
sudo yum install grafana
```

Start service and check the status

```
sudo systemctl daemon-reload && sudo systemctl start grafana-server && sudo
systemctl status grafana-server
```

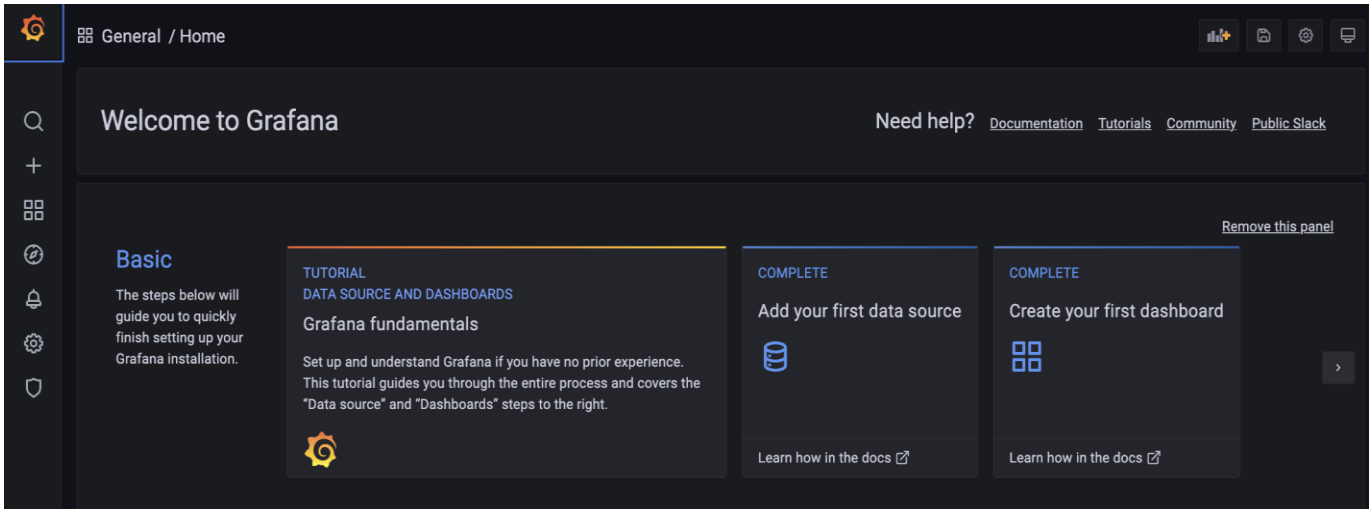
You might also want to enable the Grafana service.

```
sudo systemctl enable grafana-server.service
```

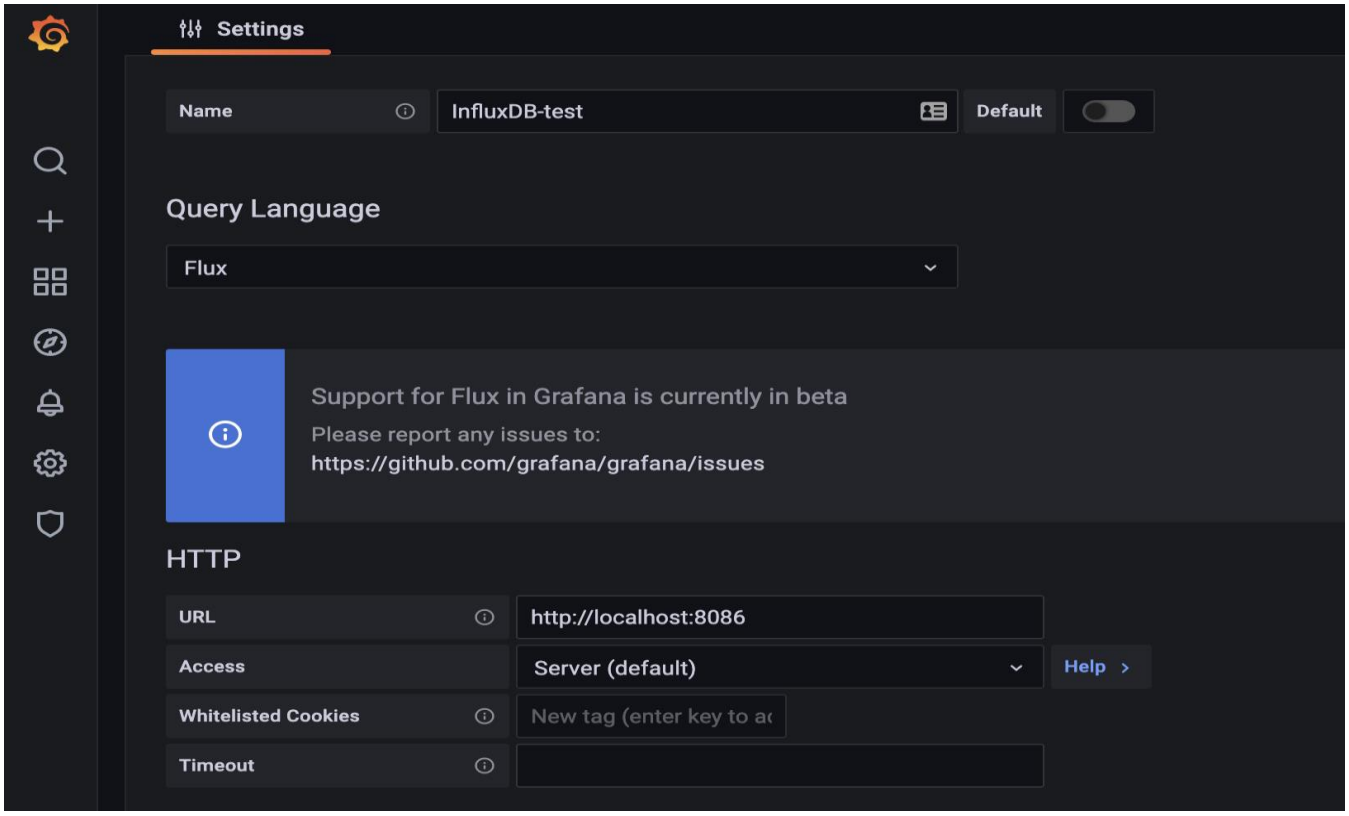
Grafana should be up and running on 3000

```
curl http://10.87.141.223:3000/
```

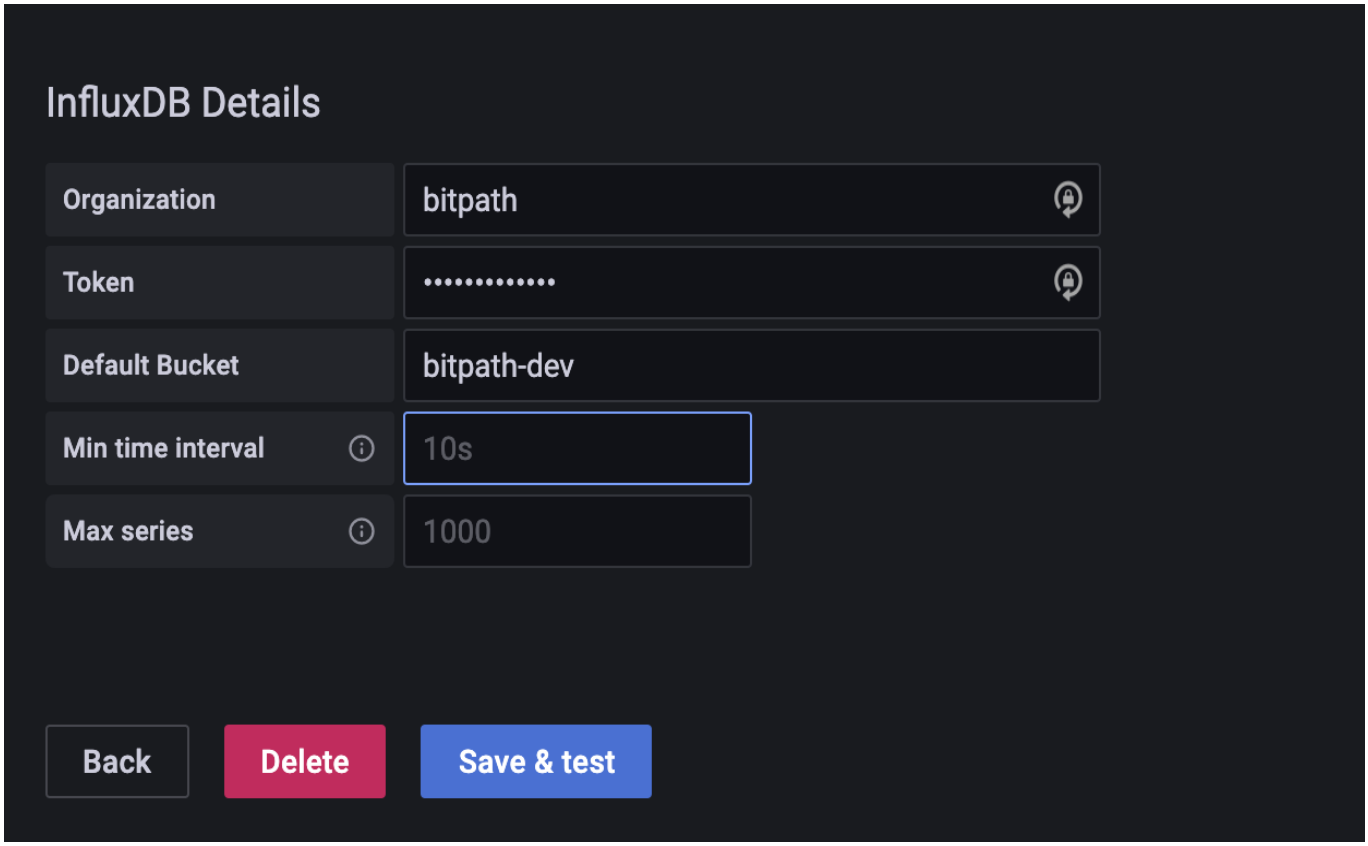
Access Grafana's UI using a browser and click on add your first datasource



Choose InfluxDB from data sources and fill in the following details.



Under InfluxDB details, fill in the organization, bucket name and access token you created during InfluxDB configuration, save and test.



You are now ready to start ingesting data from your edge devices and visualizing them.

5 Install and Configure Telegraf Agent on Edge devices

SSH to your Raspberry Pi, and install and configure Telegraf. Install necessary binaries

```
sudo apt-get update && sudo apt-get install apt-transport-https
```

We are on Bullseye for Raspberry 4 - so add the influx key

```
wget -qO- https://repos.influxdata.com/influxdb.key | sudo apt-key add -

source /etc/os-release

test $VERSION_ID = "10" && echo "deb https://repos.influxdata.com/debian buster
stable" | sudo tee /etc/apt/sources.list.d/influxdb.list
```

5.1 Install and enable service

```
sudo apt-get update && sudo apt-get install telegraf && systemctl enable
telegraf.service
```

Start the service once and check the status to see if it's running. There will be some errors initially because the default configuration will try to talk to a local influxdb but check if the service starts and then stop again.

```
systemctl start telegraf.service
systemctl status telegraf.service

# stop service after this
```

Rename the default config file and create a new telegraf.conf

```
mv /etc/telegraf/telegraf.conf /etc/telegraf/telegraf.conf.bak

vi /etc/telegraf/telegraf.conf
```

The telegraf config file is where you configure the metrics you want to collect in the form of inputs and where this data gets written in the form of outputs. The sample configuration is given below

```
# Configuration for telegraf agent

[agent]

  ## Default data collection interval for all inputs

  interval = "10s"
```

```
  ## Rounds collection interval to 'interval'
  ## ie, if interval="10s" then always collect on :00, :10, :20, etc.
  round_interval = true

  ## Telegraf will send metrics to outputs in batches of at most
  ## metric_batch_size metrics.
  ## This controls the size of writes Telegraf sends to output plugins.
  metric_batch_size = 1000

  ## Maximum number of unwritten metrics per output.  Increasing this value
  ## allows for longer periods of output downtime without dropping metrics at the
  ## cost of higher maximum memory usage.
  metric_buffer_limit = 10000

  ## Collection jitter is used to jitter the collection by a random amount.
  ## Each plugin will sleep for a random time within the jitter before collecting.
  ## This can be used to avoid many plugins querying things like sysfs at the
  ## same time, which can have a measurable effect on the system.
  collection_jitter = "0s"

  ## Default flushing interval for all outputs. Maximum flush_interval will be
  ## flush_interval + flush_jitter
  flush_interval = "10s"
  ## Jitter the flush interval by a random amount. This is primarily to avoid
  ## large write spikes for users running many telegraf instances.
  ## ie, a jitter of 5s and interval 10s means flushes will happen every 10-15s
  flush_jitter = "0s"

  ## By default or when set to "0s", precision will be set to the same
  ## timestamp order as the collection interval, with the maximum being 1s.
  ##   ie, when interval = "10s", precision will be "1s"
  ##       when interval = "250ms", precision will be "1ms"
  ## Precision will NOT be used for service inputs. It is up to each individual
  ## service input to set the timestamp at the appropriate precision.
  ## Valid time units are "ns", "us" (or "µs"), "ms", and "s".
  precision = ""
```

```
## Log at debug level.
# debug = false
## Log only error level messages.
# quiet = false

## Log target controls the destination for logs and can be one of "file",
## "stderr" or, on Windows, "eventlog".  When set to "file", the output file
## is determined by the "logfile" setting.
# logtarget = "file"

## Name of the file to be logged to when using the "file" logtarget.  If set to
## the empty string then logs are written to stderr.
# logfile = ""

## The logfile will be rotated after the time interval specified.  When set
## to 0 no time based rotation is performed.  Logs are rotated only when
## written to, if there is no log activity rotation may be delayed.
# logfile_rotation_interval = "0d"

## The logfile will be rotated when it becomes larger than the specified
## size.  When set to 0 no size based rotation is performed.
# logfile_rotation_max_size = "0MB"

## Maximum number of rotated archives to keep, any older logs are deleted.
## If set to -1, no archives are removed.
# logfile_rotation_max_archives = 5

## Pick a timezone to use when logging or type 'local' for local time.
## Example: America/Chicago
# log_with_timezone = ""

## Override default hostname, if empty use os.Hostname()
hostname = "bitpath-dev-xx"
## If set to true, do no set the "host" tag in the telegraf agent.
omit_hostname = false
```

```
[[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 at each interval.
  ## urls exp: http://127.0.0.1:9999
  urls = []

  ## Token for authentication.
  token = ""

  ## Organization is the name of the organization you wish to write to; must
exist.
  organization = ""

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

# ----------------------------------------------------- Defaults ------------------
-------------------------

# Read CPU metrics
[[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

# Gather metrics from disks
[[inputs.disk]]
  interval = "5m"
```

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

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

# Read HDD temp metric from the hddtemp daemon
#[[inputs.hddtemp]]

# Read metrics about system load & uptime
[[inputs.system]]
  ## Uncomment to remove deprecated metrics.
  fielddrop = ["uptime_format"]

# 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

# Read metrics about the IRQ
```

```
[[inputs.interrupts]]
   cpu_as_tag = false

#[[inputs.linux_sysctl_fs]]

# Gather metrics about network interfaces
[[inputs.net]]
   ignore_protocol_stats = false

# Collect TCP connections state and UDP socket counts
[[inputs.netstat]]
  # no configuration

# Gather metrics from /proc/net/* files
#[[inputs.nstat]]

# ----------------------------------------------------- Top processes (default) --
------------------------------


[[inputs.procstat]]
  pattern = "."
  pid_tag = false
  pid_finder = "native"
  fieldpass = [
      "num_threads",
      "cpu_usage",
      "memory_rss",
  ]

[[processors.topk]]
  # see: https://github.com/influxdata/telegraf/blob/release-
1.17/plugins/processors/topk/README.md
  namepass = ["*procstat*"]
  aggregation = "max"
  k = 10
  fields = [
```

```
      "num_threads",
      "cpu_usage",
      "memory_rss",
  ]

[[processors.regex]]
  namepass = ["*procstat*"]
  [[processors.regex.tags]]
    key = "process_name"
    pattern = "^(.{60}).*"
    replacement = "${1}..."

# ----------------------------------------------------- Raspberry Pi ------------
---------------------------------

# Read RPi CPU temperature
[[inputs.exec]]
  commands = [ '''sed -e 's/^\([0-9]\{2\}\)\(.*\)$/\1.\2/'
/sys/class/thermal/thermal_zone0/temp''' ]
  name_override = "sys"
  data_format = "grok"
  grok_patterns = ["%{NUMBER:thermal_zone0:float}"]

# Vcgencmd input
[[inputs.exec]]
  commands = ["/var/lib/telegraf/vcgencmd.sh"]
  timeout = "7s"
  data_format = "influx"
```

Note that you must supply the correct authentication token, hostname, etc.

Start the service

Start the service now, and it should start shipping data to InfluxDB.

```
systemctl start telegraf.service
```

6 References