

Telegraf, InfluxDB & Grafana Installation, Setup and Configuration to monitor NVMesh clients and targets on CentOS 7

Telegraf is the data collector running in the targets and clients to be monitored.

InfluxDB is the time series database storing the data its receiving from the Telegraf agents.

Grafana is the visualization tool being used to visualize and interface with the data stored in the InfluxDB time series database.

1. Preparation

Download the Telegraf and InfluxDB rpm packages here:

[Telegraf download](#)

[Influxdb download](#)

You need to install the Telegraf package on every target and client you intend to monitor.

Download the Grafana rpm packages here:

[Grafana Download](#)

Download the NVMesh Telegraf plugin here: [NVMesh Plugin download](#)

The InfluxDB and Grafana package and services only needs to be installed on one host/server which will host the InfluxDB instance and/or the Grafana service/instance.

You can install all together on the same host/server or separate hosts/servers.

Curl needs to be installed and working on the host/server with the telegraf packages and the InfluxDB services running to verify and check the installation..

2. Installation

Change into the download folder containing the rpm packages.

1. Install the Telegraf package:
`sudo yum localinstall telegraf-x.x.x.x86_64.rpm`
2. Install the InfluxDB package:
`sudo yum localinstall influxdb-x.x.x.x86_64.rpm`
1. Install the Grafana package:
`sudo yum localinstall grafana-x.x.x.x86_64.rpm`

3. Verify the services, configuration and dashboard setup

1. Check if the InfluxDB service is running:
`sudo systemctl status influxdb`

If the service is NOT running, run/issue the following commands:

```
sudo systemctl daemon-reload
sudo systemctl enable influxdb
sudo systemctl start influxdb
```

Verify if the InfluxDB service is running now:

```
sudo systemctl status influxdb
```

Run the SHOW DATABASES command using curl:

```
curl "http://localhost:8086/query?q=show+databases"
```

If InfluxDB is up and running, you should see an object that contains the `_internal` database:

```
{"results":[{"statement_id":0,"series":[{"name":"databases","columns":["name"],"values":[["_internal"]}]}]}
```

2. Check if the Telegraf service is running:

```
sudo systemctl status telegraf
```

If the service is NOT running, run/issue the following commands:

```
sudo systemctl enable telegraf
```

```
sudo systemctl start telegraf
```

Verify if the Telegraf service is running now:

```
sudo systemctl status telegraf
```

The default installation should create a configuration file with System Stats as an input plugin and InfluxDB as an output plugin.

Double check the configuration file at `/etc/telegraf/telegraf.conf` for the relevant input and output settings.

The OUTPUT PLUGINS section should have the following settings for the InfluxDB output:

```
[[outputs.influxdb]]
## The full HTTP or UDP endpoint URL for your InfluxDB instance.
## Multiple urls can be specified as part of the same cluster,
## this means that only ONE of the urls will be written to each interval.
# urls = ["udp://localhost:8089"] # UDP endpoint example
urls = ["<your_influxDB_host>:8086"] # required
## The target database for metrics (telegraf will create it if not exists).
database = "telegraf" # required

## Retention policy to write to. Empty string writes to the default rp.
retention_policy = ""
## Write consistency (clusters only), can be: "any", "one", "quorum", "all"
write_consistency = "any"

## Write timeout (for the InfluxDB client), formatted as a string.
## If not provided, will default to 5s. 0s means no timeout (not recommended).
timeout = "5s"
# username = "telegraf"
# password = "metricsmetricsmetricsmetrics"
## Set the user agent for HTTP POSTs (can be useful for log differentiation)
# user_agent = "telegraf"
## Set UDP payload size, defaults to InfluxDB UDP Client default (512 bytes)
# udp_payload = 512
```

Next, the INPUT PLUGINS section should have the following settings for the system stats input:

```
# 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
```

```
# Read metrics about disk usage by mount point
```

```
[[inputs.disk]]
```

```
## By default, telegraf gather stats for all mountpoints.
```

```
## Setting mountpoints will restrict the stats to the specified mountpoints.
```

```
# mount_points = ["/"]
```

```
## Ignore some mountpoints by filesystem type. For example (dev)tmpfs (usually
```

```
## present on /run, /var/run, /dev/shm or /dev).
```

```
ignore_fs = ["tmpfs", "devtmpfs"]
```

```
# Read metrics about disk IO by device
```

```
[[inputs.diskio]]
```

```
## By default, telegraf will gather stats for all devices including
```

```
## disk partitions.
```

```
## Setting devices will restrict the stats to the specified devices.
```

```
# devices = ["sda", "sdb"]
```

```
## Uncomment the following line if you need disk serial numbers.
```

```
# skip_serial_number = false
```

```
# 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
```

3. Install and configure the NVMesh telegraf Plugin. Details and How To can be found here: [NVMesh Telegraf Plugin](#)

Run a quick test with Curl to verify the plugin setup:
`curl "http://localhost:8086/query?q=select+*+from+telegraf..cpu"`

If Telegraf is set up right and up and running, a lot of JSON data will appear on your screen. If so, go on.

4. Check if the Grafana service is running:
`sudo systemctl status grafana-server`

If the service is NOT running, run/issue the following commands:

```
sudo systemctl enable grafana-server
sudo systemctl start grafana-server
```

Verify if the Telegraf service is running now:
`sudo systemctl status grafana-server`

Assuming everything is up and running you should be able to connect to and configure Grafana to show NVMesh client and target data. Point your web browser to <http://<the grafana server IP address>:3000>

Grafana configuration and dash board setup

Login with user: *admin* and password: *admin*

The next step is to point and connect Grafana to the InfluxDB data base instance. Configure the data source as shown in the picture below:

Edit data source

Name	InfluxDB	Default	<input checked="" type="checkbox"/>
Type	InfluxDB		

Http settings

Url	http://<InfluxDB server IP address>:8086
Access	proxy

Http Auth

Basic Auth	<input type="checkbox"/>	With Credentials	<input type="checkbox"/>
TLS Client Auth	<input type="checkbox"/>	With CA Cert	<input type="checkbox"/>

InfluxDB Details



Database	telegraf		
User		Password	

Default group by time:

Click <Save & Test>

Next is to setup and configure the NVMesh Plugin Grafana dashboards.


From the top left corner, navigate to Dashboards > Import and click on <Upload .json File> and select the NVMesh Plugin Dashboard JSON file you want to use and install. The preconfigured NVMesh Plugin dasgboards can be found found here: [NVMesh Grafana Dashboards](#). Select InfluxDB as the data source and click on <Import>.

 *Import Dashboard* 



Options

Name

NVMe Target View




InfluxDB

✓ Select a InfluxDB data source

InfluxDB

 Import

Cancel

Back

Repeat the previous step for every dashboard you want to use and install.

Further information can be found here:

Grafana - <http://docs.grafana.org/>

InfluxDB - https://docs.influxdata.com/influxdb/v1.3/introduction/getting_started/