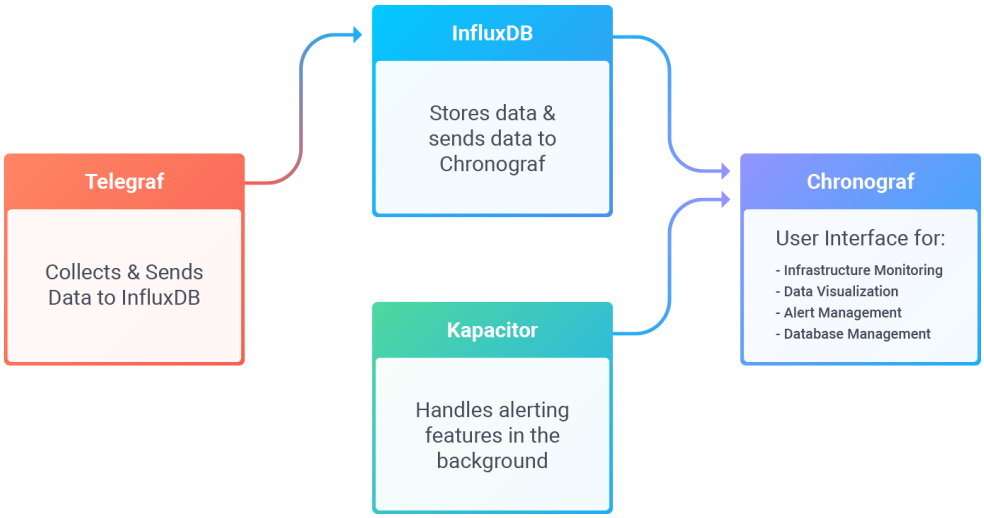
Introduction

The purpose of this service description document is to provide a detailed description of the TICK Stack service, realized by using the Influxdata technology. It describes a well-defined baseline of the service and it's implementation. The aim is to document the key aspects of the TICK Stack service technical design. This Includes:

- overview of the Influxdata platform (TICK STACK):
    - Telegraf;
    - InfluxDB;
    - Chronograf;
    - Kapacitor.
- detailed description about telegrafs agents and input plugins:
    - mapping of network interfaces;
    - mapping of cpu cores;
    - mapping of storage devices;
    - mapping of snmp enabled devices.
- detailed description of the data analysis dashboards:
    - variables, drilldowns and alerts.

On this particular case, TICK Stack is being deployed in an automated manner, resorting to an Ansible playbook created for such purpose. The different components of the TICK Stack are configured to run inside Docker Containers, running inside a VM hosted by the Openstack platform of the IT Datacenter. The created playbook is responsible to apply the most recent software updates, install package dependencies for the proper execution of the tick stack platform tools, creating the containers with the latest version available of the different tools, create and map data storage, importing and applying the necessary configurations as well as importing the different components needed for the data visualization platform. Besides the TICK Stack tools, Grafana is also incorporated in the package of tools used, being Grafana a 3rd party data visualization tool, which offers a larger variety of plugins, data visualization methods, alerts and data sources.

Influxdata Overview



Influxdata is a time series data platform built to integrate real-time analytics, event handling and time-based data with Open Source plugins. A time series database deals with specific workloads and requirements. They need to ingest millions of data points per second, performing real-time queries across these large data sets in a non-blocking manner, downsampling and evicting high-precision low-value data. It optimizes data storage, reducing its costs and performing complex time-bound queries to extract meaningful insight from the data. Influxdata is a complete platform for handling all time series data, from humans, sensors or machines - seamlessly collecting, storing, visualizing and turning insight into action. With both fast deployment and fast performance, Influxdata delivers real value in real time.

TICK Stack - InfluxDB

InfluxDB is a time series database designed to handle high write and query loads. It is an integral component of the TICK Stack. It is meant to be used as a backing store for any use case involving large amounts of timestamped data, including DevOps monitoring, application metrics, IoT sensor data (in both cases the data is collected by Telegraf) and real-time analytics.

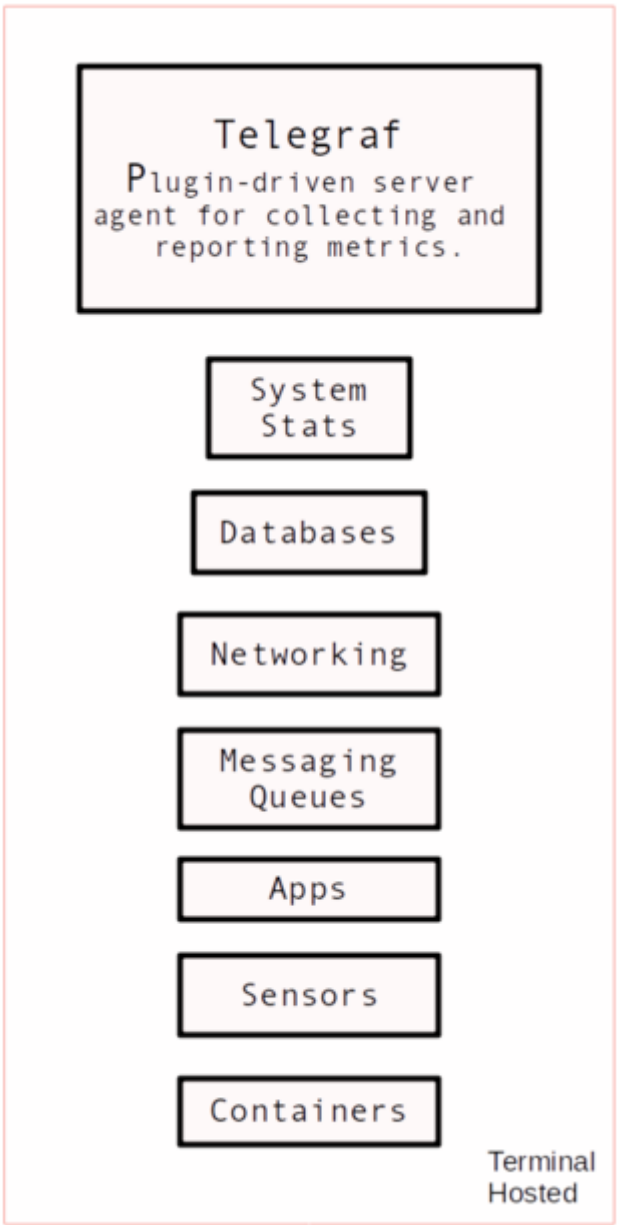
A fresh installation of InfluxDB, has no databases (apart from the system _internal), so the first step after setting up the service is to create a database by opening up InfluxDB CLI with the command "influx" and typing in CREATE DATABASE . Since the entire TICK Stack is being deployed inside a Docker Container, this can also be achieved by setting a group of environmental variables in which the database name, user and password can be specified. As far as needed configuration, there isn't much to be done in InfluxDB aside from the creation of the database and the definition of the port in which it will be accessible. All of those, as referred above, can be setup when the Docker Container is being deployed.

InfluxDB Ansible Playbook Docker Config

```
docker_container:
    name: tick_influxdb
    image: influxdb:latest
    state: started
    restart_policy: always
    ports:
    - "8086:8086"
    - "8086:8086/udp"
    env:    #environmental variables
        INFLUXDB_DB="{{ database }}"    #specify the database name to be created
        INFLUXDB_ADMIN_ENABLED="true"    #enable administration account
        INFLUXDB_ADMIN_USER="{{ admin }}"    #specify the admin username
        INFLUXDB_ADMIN_PASSWORD="{{ admin_password }}"    #specify the admin password
        INFLUXDB_USER="{{ user }}"    #specify a username for the database
        INFLUXDB_USER_PASSWORD="{{ user_password }}"    #specify the created user
password
    volumes:
      - influxdb_storage:/var/lib/influxdb
      - influxdb_data:/etc/influxdb
```

TICK Stack - Telegraf

Telegraf is part of the TICK Stack and is a plugin-driven server agent for collecting and reporting metrics. Telegraf has integrations to source a variety of metrics, events and logs, directly from the containers and systems it's running on, pull metrics from third-party APIs, or even listen for metrics via a snmp service. It also has output plugins to send metrics to a variety of other datastores, services and message queues, including InfluxDB.



In order for the data collected by Telegraf to be registered it is necessary to specify in it's configuration the InfluxDB host ip address as well as the database in which the data will be stored.

Telegraf Ansible Playbook Docker Config

```
docker_container:
    name: tick_telegraf
    image: telegraf:latest
    state: started
    restart_policy: always
    ports:
    - "8092:8092"
    - "8094:8094"
    - "8125:8125"
    env:
      INFLUX_URL="{{ influxdb_host }}"    #ip address of the InfluxDB host
      INFLUX_DATABASE="{{ influxdb_database }}"    #database name
      INFLUX_USER="{{ influxdb_user }}"    #previously created InfluxDB user
      INFLUX_PASSWORD="{{ influxdb_password }}"    #password of InfluxDB user
    volumes:
      - telegraf_storage:/var/lib/telegraf
      - telegraf_data:/etc/telegraf
```

Besides the Docker Container configuration it is also necessary to configure Telegraf itself. Such configuration is stored in the file telegraf.conf, which can be found both inside the container (/etc/telegraf), as well as in the data storage volume created in Docker into which the container files are pointed to (/var/lib/docker/volumes/telegraf_data/_data). The configuration file contains all the input and output plugins configuration, specifying which will be used to collect metrics and where to store it.

```
###############################################################################
#                            TELEGRAF CONFIGURATION                           #
###############################################################################

[agent]
  interval = "10s"
  round_interval = true
  metric_batch_size = 1000
  metric_buffer_limit = 10000
  collection_jitter = "0s"
  flush_interval = "10s"
  flush_jitter = "0s"
  debug = false
  quiet = true
  hostname = ""
  omit_hostname = false

[global_tags]
  user = "$INFLUX_USER"

###############################################################################
#                              OUTPUT PLUGINS                                 #
###############################################################################

[[outputs.influxdb]]
  urls = ["$INFLUX_URL"]
```

```
  database = "$INFLUX_DATABASE"
  skip_database_creation = true
  password = "$INFLUX_PASSWORD"

###############################################################################
#                            INPUT PLUGINS                                    #
###############################################################################

[[inputs.cpu]]
  percpu = true
  totalcpu = true
  collect_cpu_time = false
  report_active = true

[[inputs.disk]]
  ## By default stats will be gathered for all mount points. ##

[[inputs.diskio]]
  ## By default, telegraf will gather stats for all devices including disk partitions. ##

[[inputs.mem]]
  # Read metrics about memory usage #

[[inputs.processes]]
  # Get the number of processes and group them by status #

[[inputs.swap]]
  # Read metrics about swap memory usage #

[[inputs.system]]
  # Read metrics about system load & uptime #

[[inputs.net]]
  # Read metrics about network interface usage #

[[inputs.netstat]]
  # Read TCP metrics such as established, time wait and sockets counts. #

[[inputs.temp]]
  # Read metrics about temperature #
```