INFLUX PLUGIN

The influx plugin continuously collects and sends time series data to an influxdb database.

The influx plugin was introduced in the 13.x *Mimic* release.

ENABLING

To enable the module, use the following command:

```
ceph mgr module enable influx
```

If you wish to subsequently disable the module, you can use the equivalent *disable* command:

```
ceph mgr module disable influx
```

CONFIGURATION

For the influx module to send statistics to an InfluxDB server, it is necessary to configure the servers address and some authentication credentials.

Set configuration values using the following command:

```
ceph config-key set mgr/influx/<key> <value>
```

The most important settings are `hostname`, `username` and `password`. For example, a typical configuration might look like this:

```
ceph config-key set mgr/influx/hostname influx.mydomain.com
ceph config-key set mgr/influx/username admin123
ceph config-key set mgr/influx/password p4ssw0rd
```

Additional optional configuration settings are:

interval: Time between reports to InfluxDB. Default 5 seconds.
database: InfluxDB database name. Default “ceph”. You will need to create this database and grant write privileges to the configured username or the username must have admin privileges to create it.
port: InfluxDB server port. Default 8086
ssl: Use https connection for InfluxDB server. Use “true” or “false”. Default false
verify_ssl: Verify https cert for InfluxDB server. Use “true” or “false”. Default true

DEBUGGING

By default, a few debugging statments as well as error statements have been set to print in the log files. Users can add more if necessary. To make use of the debugging option in the module:

- Add this to the ceph.conf file.:

  ```
  [mgr]
      debug_mgr = 20
  ```

- Use this command `ceph tell mgr.<mymonitor> influx self-test`.
- Check the log files. Users may find it easier to filter the log files using *mgr[influx]*.

INTERESTING COUNTERS

The following tables describe a subset of the values output by this module.

POOLS

Counter	Description
bytes_used	Bytes used in the pool not including copies
max_avail	Max available number of bytes in the pool
objects	Number of objects in the pool
wr_bytes	Number of bytes written in the pool
dirty	Number of bytes dirty in the pool
rd_bytes	Number of bytes read in the pool
raw_bytes_used	Bytes used in pool including copies made

OSDS

Counter	Description
op_w	Client write operations
op_in_bytes	Client operations total write size
op_r	Client read operations
op_out_bytes	Client operations total read size

Counter	Description
op_wip	Replication operations currently being processed (primary)
op_latency	Latency of client operations (including queue time)
op_process_latency	Latency of client operations (excluding queue time)
op_prepare_latency	Latency of client operations (excluding queue time and wait for finished)
op_r_latency	Latency of read operation (including queue time)
op_r_process_latency	Latency of read operation (excluding queue time)
op_w_in_bytes	Client data written
op_w_latency	Latency of write operation (including queue time)
op_w_process_latency	Latency of write operation (excluding queue time)
op_w_prepare_latency	Latency of write operations (excluding queue time and wait for finished)
op_rw	Client read-modify-write operations
op_rw_in_bytes	Client read-modify-write operations write in
op_rw_out_bytes	Client read-modify-write operations read out
op_rw_latency	Latency of read-modify-write operation (including queue time)
op_rw_process_latency	Latency of read-modify-write operation (excluding queue time)
op_rw_prepare_latency	Latency of read-modify-write operations (excluding queue time and wait for finished)
op_before_queue_op_lat	Latency of IO before calling queue (before really queue into ShardedOpWq) op_before_dequeue_op_lat
op_before_dequeue_op_lat	Latency of IO before calling dequeue_op(already dequeued and get PG lock)

Latency counters are measured in microseconds unless otherwise specified in the description.