
Skyminer

Kratos Communications

Dec 22, 2023

CONTENTS

1	Cassandra dashboard	3
1.1	Configuration	3
1.2	Cassandra metrics	4
1.3	Glossary	5
2	Continuous Integration	7
2.1	Introduction	7
2.2	How it works	7
2.3	Tags	7
2.4	Nightly build	8
3	Jupyter	9
3.1	Adding LaTeX fonts	9
3.1.1	Introduction	9
3.1.2	Japanese characters process	9
3.2	Manage Python packages	10
3.2.1	When new packages have been installed	10
3.2.2	Adding new libraries	10
3.3	List of libraries	10
3.4	How to update skyminer-timeseries-python-api	11
4	Nexus	13
4.1	Introduction	13
4.2	Pushing	13
5	NGINX	15
5.1	Introduction	15
5.1.1	About the nginx docker image	15
5.2	Global hardening	15
5.2.1	Hide Nginx version number	15
5.2.2	Hide Nginx server signature	16
5.2.3	Reduce XSS risks	16
5.2.4	Control the behavior of the referer header	16
5.2.5	Provide clickjacking protection	16
5.2.6	X-XSS protection	16
5.2.7	Prevent sniff mimetype	16
5.2.8	SSL hardening	16
5.2.9	Keep only TLS 1.2	16
5.2.10	Use only strong ciphers	17
5.2.11	Use more secure ECDH curve	17

5.2.12	Defend against the BEAST attacks	17
5.2.13	HTTP Strict Transport Security	17
5.2.14	Disable compression	17
6	Packaging	19
6.1	Requirements	19
6.2	Usage	19
6.2.1	Packaging Skyminer	19
6.2.2	Generating docker images	20
6.2.3	Building Skyminer locally	20
7	Release	21
7.1	Introduction	21
7.2	Step-by-step process	21
7.2.1	Agents	21
7.2.2	Skyminer document	21
7.2.3	Skyminer UI	22
7.2.4	Docker images	22
7.2.4.1	Jupyter	22
7.2.4.2	Grafana	22
7.2.4.3	Other images	23
7.2.5	BIRT plugin	23
7.2.6	Skyminer core	24
8	SSL Certificates	25
8.1	Introduction	25
8.2	Creation	25
9	Development environment	27
9.1	Installation and configuration	27
9.1.1	Login	27
9.1.2	Prerequisites	27
9.1.3	How to install	27
9.1.4	NG Angular	28
9.2	How it works	28
9.3	Module system	28
9.4	Config files	28
9.4.1	dev.py	29
9.4.2	kubernetes.py	29
9.4.2.1	conf_kubernetes	29
9.4.2.2	conf_kubernetes_dev	29
9.5	Templating Engine Templator	29
9.5.1	Variables	29
9.5.2	if conditions	30
9.6	Use cases	30
9.6.1	I want to activate the dev mod of Skyminer	30
9.6.2	I want to update the kubernetes files of one of my services	30
9.6.3	I want to change my number of replicas of one of my services	30
9.6.4	I want to run Angular dev NG Serve behind Skydev ingress	31
9.6.5	I want to check the logs of one of my pods	31
9.6.6	I want to manually delete one of my pod	31
9.6.7	I want to stop Skyminer	31
9.6.8	I want to start Skyminer	31
9.6.9	I want to reboot Skyminer	31
9.6.10	I want to pull from the nexus	31

9.6.11	I want to add the autocompletion of kubectl	32
9.7	./cli.py	32
10	Skyminer Performances	33
10.1	Performance metrics	33
10.1.1	Query Throughput	33
10.1.2	Ingest Throughput	33
10.2	Dimensioning a Skyminer system	34
10.3	Performance trade-offs	34
10.4	Alternative designs	35
10.5	Paths of improvement	35
10.5.1	Cassandra	35
10.5.2	Skyminer	35
11	Testing	37
11.1	API automated testing	37
11.1.1	Tools	37
11.1.1.1	Postman	37
11.1.1.2	Newman	37
11.1.2	Usage	37
11.1.2.1	Postman	37
11.1.2.2	Newman	46
11.2	Manual testing	47
11.2.1	Scenario	47
12	Skyminer Expert features API	49
12.1	POST: /api/skyminer/expert-features/query/results_as_insert_request	49
12.1.1	Tags management	49
12.2	POST: /api/skyminer/expert-features/query/save_results	49
13	Design Notes	51
13.1	Authentication	51
13.1.1	Mechanism	51
13.1.2	Why Keycloak?	51
13.1.3	How authentication is implemented on Skyminer scope?	52
13.1.3.1	Knight	53
13.1.3.2	Securing Skyminer API	54
13.1.3.3	Securing Skyminer WEB UI	55
13.1.3.4	Securing Grafana	55
13.1.3.5	Securing Opensearch and Opensearch Dashboard	55
13.1.3.6	Securing Jupyter notebook	55
13.2	Access Control	55
13.2.1	Mechanism	56
13.2.2	Time Series ACL	56
13.2.2.1	Metrics filters	57
13.2.2.2	Tag filters	57
13.2.2.3	Results filtering	58
13.2.3	OpenSearch & Document API ACL	58
13.2.4	Object API filters	58
13.3	Kubernetes	58
13.3.1	Basic deployment	58
13.3.2	SSO deployment	59
13.4	RocksDb datastore	60
13.4.1	RocksDB Settings	60
13.4.2	General aspects	60

13.4.3	Operations	60
13.4.4	Metrics CF	61
13.4.5	Time Series Index CF	61
13.4.6	Series reverse index CF	61
13.4.7	Data CF	61
13.4.8	Service keystore CF	61
13.4.9	Service keystore change CF	61
13.4.10	Compressed Data CF (possibility for later)	62
13.5	High Cardinality Series	62
13.5.1	High-cardinality problems	62
13.5.1.1	Slow query performance in Cassandra	62
13.5.1.2	UI Responsiveness	62
13.5.1.3	UI usability	62
13.5.2	Proposed improvements	63
13.5.2.1	Query cardinality limit	63
13.5.2.2	Data type filter	63
13.5.2.3	Time series count API endpoint	63
13.5.2.4	UI Improvements: series limit	64
13.5.2.5	UI Improvements: number of series	64
13.5.2.6	UI Improvements: support number of tags in interactive tag selector	64
13.5.2.7	Tags query improvement	65
13.5.2.8	UI Improvements: data type	65
13.5.2.9	Grafana UI Improvements	65
13.5.2.10	Query add number of time series in response	66
13.5.2.11	UI add metrics/tags from response	66
13.5.2.12	User guidelines	66
13.5.2.13	Design alternate storage for high-cardinality series	67
13.5.2.14	Inform user on server load	67
13.6	OpsCenter Installation	67
13.6.1	Install KMP	68
13.6.1.1	How to install Compass	68
13.6.1.2	How to use Compass for demo	69
13.6.1.3	How to install KMP	69
13.6.1.4	How to update KMP for Skyminer	70
13.6.2	Miscellaneous	70
13.7	KMP Message Queue Interface	71
13.7.1	KMP AMQP Messaging patterns	71
13.7.1.1	Request/Reply	71
13.7.1.2	Pub/Sub	71
13.7.1.3	Discovery (Scatter/Gather)	71
13.7.2	Metrics Data ingest	71
13.7.3	Metrics Data query	72
13.7.4	Scalability	73
13.8	Performance History Service Integration	73
13.8.1	How to use	73
13.8.2	Measurand	73
13.8.2.1	Datamodel	73
13.8.2.2	Mapping with Skyminer	74
13.8.3	Java Interface	74
13.8.3.1	MeasurementObj Object	75
13.8.3.2	CallInfo Object	75
13.8.3.3	FinishPost Method	76
13.9	Skyminer-KMP Integration	76
13.9.1	Metrics data storage	76

13.9.2	Metrics analytics service	76
13.9.3	Third-party integrations	77
13.9.4	Data model mapping	77
13.9.5	Steps for integration	77
13.9.5.1	Skyminer REST services for KMP	77
13.9.5.2	Interfacing KMP Message Queue	78
13.9.5.3	Mixed approach for third-party and legacy	78
13.9.6	Authentication & ACL	78
13.9.7	Packaging	78
13.9.7.1	KMP Data & Analytics Service	78
13.9.7.2	Data analytics & Third-party integrations	79
13.9.8	Time Series meta-data	79
13.10	Log collectors comparison articles	79
13.11	Filebeat	80
13.11.1	Introduction	80
13.11.2	Pros and cons	80
13.11.2.1	Pros	80
13.11.2.2	Cons	80
13.11.3	Docker	80
13.11.3.1	Prepare your system	80
13.11.3.2	Docker compose	82
13.11.4	Kubernetes	83
13.11.5	Elastic Stack integration	83
13.11.5.1	Logs TTL aka Index Lifecycle Management (ILM)	83
13.11.5.2	Configure the ILM	83
13.11.6	References	83
13.12	Fluentd	84
13.12.1	Introduction	84
13.12.2	Pros and cons	84
13.12.2.1	Pros	84
13.12.2.2	Cons	84
13.12.3	Docker	84
13.12.3.1	Prepare your system	84
13.12.3.2	Docker compose	86
13.12.4	Kubernetes	90
13.12.5	Standalone Fluentd aka td-agent	90
13.12.5.1	Run the dummy service container	91
13.12.5.2	Configure td-agent	91
13.12.5.3	Code the dummy service	91
13.12.5.4	Run the dummy service	92
13.12.6	Elastic Stack integration	92
13.12.6.1	Logs TTL aka Index Lifecycle Management (ILM)	92
13.12.6.2	Configure the ILM	92
13.12.7	Resources	92
13.13	Skyminer SQL	93
13.13.1	Introduction	93
13.13.2	Technical solution	93
13.13.3	Multicorn	93
13.13.3.1	Multicorn repository	93
13.13.3.2	Foreign Data Wrapper	93
13.13.4	Service availability	93
13.13.5	PostgreSQL FDW Identifiers	94
13.13.6	Update/maintenance of SQL mapping	94
13.13.6.1	Rebuild FDW REST API	94

13.13.6.2 Automated Update	94
13.13.6.3 New metric	95
13.13.6.4 New metric tag	95
13.13.7 Meta data tables (real tables)	95
13.13.7.1 skyminer_sql_metric_info	95
13.13.7.2 skyminer_sql_metric_tags_info	95
13.13.8 Data tables (virtual tables using FDW)	95
13.13.8.1 <metric>	96
13.13.9 Queries	96
13.13.9.1 FDW Tables	96
13.13.9.2 Authentication & ACL	96
14 How to run grafana using datasource plugin locally	97
14.1 Prerequisite	97
14.2 Running Grafana	97
15 State of the art, anomalies/novelties detection in time series	99
15.1 statistical approaches, machine learning approaches, deep learning approaches	99
15.1.1 Proprieties of time series	99
15.1.2 Statistical Approach	100
15.1.3 Machine Learning Approach	100
15.1.4 Deep Learning Approach	101
15.1.5 Summary table	104
15.1.6 References	104
16 Robot Framework	109
16.1 Introduction	109
16.1.1 Installation	109
16.1.2 Built-in features	109
16.1.3 Third-party features	109
16.2 Code and Features	110
16.3 Run Tests	110
16.4 Generate tests documentation	110
16.4.1 Libraries	110
16.4.2 Pandoc	110
16.5 Knowledge Base	111
17 Similarity Search	113
17.1 Introduction	113
17.2 Exact methods	113
17.2.1 Sequential scan (a.k.a. linear scan)	113
17.2.1.1 Overview	113
17.2.1.2 State-of-the-art techniques	114
17.2.2 Indexing (a.k.a. GEMINI)	115
17.2.2.1 Overview	115
17.2.2.2 State-of-the-art techniques	116
17.2.3 Experimental Evaluation	118
17.2.4 Recommendations	119
17.3 Approximate methods	119
17.3.1 State-of-the-Art for Multidimensional Vectors	119
17.3.2 State-of-the-Art for Data Series	120
17.3.3 Recommendations	120
17.4 Conclusion	120

This project is used to group all the documentation needed by developers in order to work on Skyminer projects.

CASSANDRA DASHBOARD

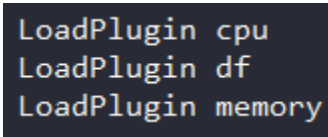
1.1 Configuration

The dashboard contains 11 metrics, with 3 coming from the system and 8 from Cassandra. These metrics are configured in a file named 'collectd.conf'.

The 3 system metrics are :

- CPU, which comes from the plugin 'cpu'
- DISK, which comes from the plugin 'df'
- RAM, which comes from the plugin 'memory'

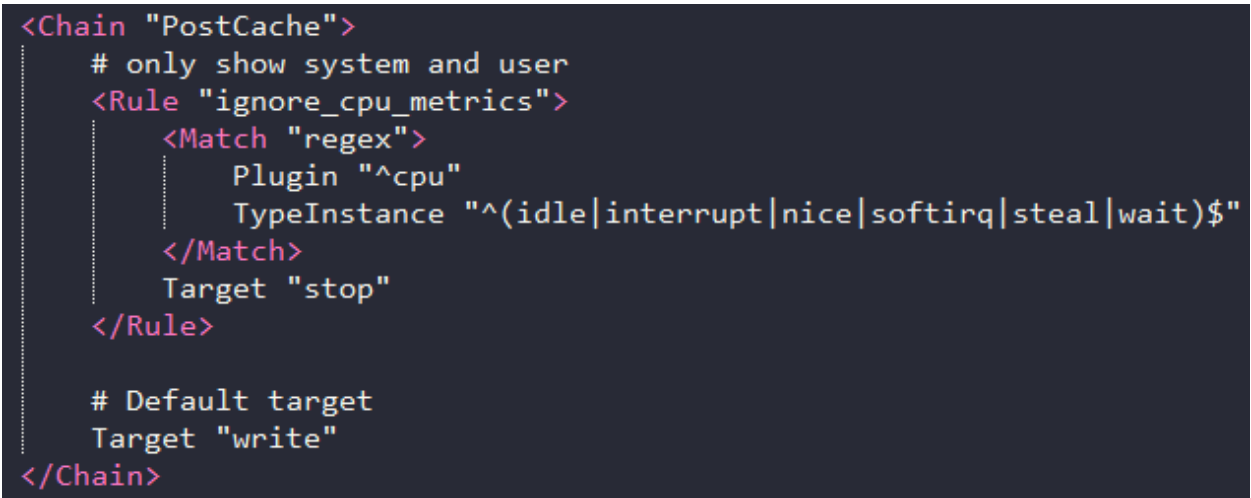
To monitor them, the following lines must be present at the start of collectd.conf :



```
LoadPlugin cpu
LoadPlugin df
LoadPlugin memory
```

Fig. 1: Loading of collectd plugins

This will enable Skyminer to receive all metrics related to cpu, disk, and ram usage. It is then possible to use the 'match_regex' plugin to filter out the ones we don't need :



```
<Chain "PostCache">
  # only show system and user
  <Rule "ignore_cpu_metrics">
    <Match "regex">
      Plugin "^cpu"
      TypeInstance "^(idle|interrupt|nice|softirq|steal|wait)$"
    </Match>
    Target "stop"
  </Rule>

  # Default target
  Target "write"
</Chain>
```

1.2 Cassandra metrics

Cassandra export its metrics using JMX, which are then gathered by collectd. Each Cassandra metric must be defined as a MBean in the collectd.conf file :

```
LoadPlugin java
<Plugin java>
    JVMArg "-verbose:jni"
    JVMArg "-Djava.class.path=/usr/share/collectd/java/collectd-api.jar:/lib/collectd-fast-jmx.jar"
    LoadPlugin "com.e_gineering.collectd.FastJMX"
    <Plugin FastJMX>
        MaxThreads 256
        CollectInternal true

        <MBean "cassandra/cfstats-write-total-latency">
            ObjectName "org.apache.cassandra.metrics:type=ColumnFamily,name=WriteTotalLatency"
            InstancePrefix "cassandra"
            <Value>
                Attribute "Count"
                InstancePrefix "write_total_latency"
                Type "counter"
            </Value>
        </MBean>

        <Connection>
            ServiceURL "service:jmx:rmi:///jndi/rmi://skyminer-cassandra:7199/jmxrmi"
            IncludePortInHostname true
            Username "skyminer"
            Password "skyminer"

            Collect "cassandra/cfstats-write-total-latency"
        </Connection>
    </Plugin>
</Plugin>
```

1.3 Glossary

Metric name	Definition
CPU USAGE	Average percentage of CPUs usage
RAM USAGE	Percentage of occupied RAM (used + cached)
DISK USAGE	Percentage of occupied disk space (used + reserved)
CASSANDRA LIVE DISK USAGE	Current disk space used by Cassandra
KEY CACHE HIT RATE	One minus the miss rate, which is defined as the ratio between key reads and key read requests (i.e. the number of physical reads of a key block from a disk over the number of requests to read a key from the cache)
SSTABLES	Number of Sorted Strings Tables on disk
TOTAL READ LATENCY	Total read latency since last check
TOTAL WRITE LATENCY	Total write latency since last check
READ LA- TENCY PER- CENTILES	75th and 95th percentiles of the read latency
WRITE LA- TENCY PER- CENTILES	75th and 95th percentiles of the write latency
READ OPERA- TIONS	Number of read latency updates
WRITE OPER- ATIONS	Number of write latency updates
MEMTABLE SIZE	Total amount of live data stored in the memtable, excluding any data structure overhead
MEMTABLE COUNT	Number of times flush has resulted in the memtable being switched out
MEMTABLE HEAP SIZE	Amount of memory reserved for the memtables

CONTINUOUS INTEGRATION

2.1 Introduction

Continuous integration is a built-in functionality of Gitlab. When a request is made, either manually or with a scheduled task, it will trigger a pipeline that can test, build, and release a Gitlab project.

2.2 How it works

The CI is configured by a file called `.gitlab-ci.yml` placed at the repository's root. The scripts set in this file are executed by the GitLab Runner.

To read more about the GitLab CI/CD, refer to the following documents:

- [How GitLab CI/CD works](#)
- [GitLab CI/CD basic workflow](#)
- [Step-by-step guide for writing .gitlab-ci.yml for the first time](#)

For **skyminer-core**, the CI calls the script `packaging_tools/scripts/deploy/deploy.sh` which pushes the artifact and the `auto-install.sh` script by `scp` on the distant server. `deploy.sh` then calls the `auto-install.sh` script by `ssh` on the distant server, which clears the old skyminer version and installs the new skyminer.

2.3 Tags

For most project the pipeline is started by tagging the project.

In the case of **skyminer-core**, this will create the release and deploy it to the Nexus. ([Click here for more info about the Nexus](#)) It will deploy the release accessible on servers at the following addresses:

- <http://192.168.48.19/> (using RocksDB datastore)
- <http://192.168.48.29/> (using Cassandra datastore)

2.4 Nightly build

Every day at around midnight, a scheduled task will build and package the master branch of the **skyminer-core** project.

It will also deploy the nightly build on servers accessible at the following addresses:

- <http://192.168.48.18/> (using RocksDB datastore)
- <http://192.168.48.28/> (using Cassandra datastore)

JUPYTER

Jupyter is built as part of skyminer-docker-images. It relies on a skyminer Python image also containing LaTeX.

3.1 Adding LaTeX fonts

3.1.1 Introduction

This process was used to make the pdf generation of Jupyter support Japanese characters.

It consists in:

- Installing the *texlive-lang-cjk* package and a Japanese font in the Jupyter docker
- Enabling the package and setting the font in the LaTeX template located in the *Jupyter-Hidecode-extension** project.

3.1.2 Japanese characters process

1. On the *skyminer-python3-latex-light* branch of the *Skyminer-docker-images* project, add the following lines in the *Dockerfile*:
`RUN DEBIAN_FRONTEND=noninteractive apt -y install --no-install-recommends texlive-lang-cjk RUN DEBIAN_FRONTEND=noninteractive apt -y install --no-install-recommends fonts-ipaexfont`
2. In the project *Jupyter-Hidecode-extension*, edit the file *templates/custom_latex.tplx* and add the following lines in the packages block:
`latex \usepackage{xCJK} \setmainfont{DejaVuSans.ttf} \setCJKmainfont{fonts-japanese-mincho.ttf}`
3. Tag the *skyminer-python3-latex-light* branch of the *Skyminer-docker-images* with a new version number
4. On the *skyminer-jupyter* branch of the *Skyminer-docker-images* project, update the version of the latex-light docker in the first line of the *Dockerfile*
5. Tag the *skyminer-jupyter* branch of the *Skyminer-docker-images* with a new version number
6. Update the version of Jupyter in the *get-env.sh* file of *skyminer-core*
7. Tag *skyminer-core*

3.2 Manage Python packages

3.2.1 When new packages have been installed

New packages have been installed (generate a new requirements.txt):

```
pip list --format=freeze
```

Update outdated packages:

```
pip list --outdated --format=freeze | grep -v '^-\e' | cut -d = -f 1 | xargs -n1 pip_
↪install -U
```

List packages and their licenses:

```
pip-licenses
```

Check why one or several package(s) is(are) required:

```
pipdeptree --reverse --packages <list of package>
```

Display conflicts in packages:

```
pipconflictchecker
```

3.2.2 Adding new libraries

- If SSL filter causes certificate verification issues with pip: Add the following options to pip command line.

```
--trusted-host pypi.python.org --trusted-host pypi.org --trusted-host files.
↪pythonhosted.org
```

- Please document below the libraries added and their purpose.

3.3 List of libraries

Complete list is in requirements.txt.

Some libraries of interest are listed below:

- Altair, plotting library, BSD License, <https://altair-viz.github.io/>
- Gluon-TS, Probabilistic Time Series Modeling, Apache license, <https://github.com/awsmlabs/gluon-ts>
- Lime, model explanation, BSD License, <https://github.com/marcotcr/lime>
- Matplotlib, Plotting library, Derivative of Python Software Foundation License (PSFL) - BSD-like License, <https://github.com/matplotlib/matplotlib/>
- Numpy, large dimension arrays & matrices operations, BSD License, <https://numpy.org/>
- Pandas, data manipulation and analysis, BSD License, <https://pandas.pydata.org/>
- Pmdarima, Bring R's auto.arima functionality to Python in scikit-learn friendly way, MIT License, <https://github.com/tgsmith61591/pmdarima>

- Prophet, Time Series Forecasting, MIT License, <https://github.com/facebook/prophet>
- Pyod, Python Outlier Detection, BSD License, <https://github.com/yzhao062/Pyod>
- Rrcf, Implementation of the Robust Random Cut Forest Algorithm for anomaly detection by Guha et al. (2016), MIT License, <https://github.com/kLabUM/rrcf>
- Ruptures, changepoint detection, BSD License, <https://altair-viz.github.io/>
- Scikit-learn, Machine Learning library, BSD License, <https://scikit-learn.org/>
- Seglearn, Segmentation feature extraction/processing/estimator, BSD License, <https://github.com/dmbee/seglearn>
- Shapely, Manipulation and analysis of geometric objects in the Cartesian plane, BSD, <https://github.com/Toblerity/Shapely>
- Tslearn, Machine learning tools for the analysis of time series, BSD License, <https://github.com/rtavenar/tslearn>
- Vega, visualization grammar, BSD License, <https://vega.github.io/vega-lite/>

Note: Prophet uses two GPL-licensed libraries that are on the way to be removed.

3.4 How to update skyminer-timeseries-python-api

1) In skyminer-timeseries-python-api:

- Update skyminer-timeseries-python-api
- Merge request (target branch master)
- Create new tag (v1.0.xx) / (master branch)

2) In skyminer-docker-images - jupyter branch:

- Dockerfile => edit => version change (v1.0.xx)
- Create new tag (skyminer-jupyter#1.xx) / (jupyter branch)

3) In skyminer-core:

- skyminer-core/packaging_tool/script/common/get-env.sh => edit => jupyter version change (1.xx)
- Merge request (target branch master)

4.1 Introduction

Nexus is a repository manager which is used to store and retrieve build artifacts and facilitate collaboration between developers. In our case we use it with to store docker images, Skyminer releases, and documentation. The docker images are pulled when the Skyminer package is created.

The Skyminer Nexus can be accessed with the following URL: <http://192.168.48.22:8082/>

It is divided into 2 repositories: *Skyminer-dev* for development artifacts, and *Skyminer* for release artifacts.

4.2 Pushing

Pushing zip artifacts and documentation to the Nexus is done by the `push_to_nexus()` function present in `push_to_nexus.sh`.

The files will be pushed to either the **Skyminer** or **Skyminer-dev** repository, depending on the value of `$SKYMINER_VERSION`.

This is done automatically by the CI, either with the nightly build or when the **skyminerKairosDBModule** is tagged.

5.1 Introduction

The SkyminerKairosDB project contains two Nginx configuration files:

- ‘ssl.conf’, which is used when https is enabled
- ‘default.conf’, which is used the rest of the time.

On these two files some instructions were added to reinforce the security of the web server and reduce the number of vulnerabilities.

5.1.1 About the nginx docker image

The ``Skyminer nginx image<https://gitlab-toulouse.kratos.us/Skyminer/skyminer-docker-images/-/tree/skyminer-nginx-proxy>`` is not based on an ``official nginx image<https://hub.docker.com/_/nginx>` because we cannot install the nginx-extras on the official image>`_

5.2 Global hardening

For in-depth information: <https://github.com/trimstray/nginx-admins-handbook#hardening>

These measures are used on both default.conf (http) and ssl.conf (https).

5.2.1 Hide Nginx version number

Disclosing the version of NGINX can be undesirable as it helps potential attackers identifying version specific vulnerabilities.

```
server_tokens off;
```

5.2.2 Hide Nginx server signature

Similar to the version number, it's usually recommended to hide the server signature.

```
more_set_headers "Server: Unknown";
```

5.2.3 Reduce XSS risks

Content security policy reduces the risk and impact of XSS attacks in modern browsers.

```
add_header Content-Security-Policy "default-src 'none'; script-src 'self'; connect-src 'self'; img-src 'self'; style-src 'self';" always;
```

5.2.4 Control the behavior of the referer header

Determine what information is sent along with the requests.

```
add_header Referrer-Policy "no-referrer";
```

5.2.5 Provide clickjacking protection

Help to protect visitors against clickjacking attacks.

```
add_header X-Frame-Options "SAMEORIGIN" always;
```

5.2.6 X-XSS protection

Prevent some categories of XSS attacks by enabling the cross-site scripting filter built into modern browsers.

```
add_header X-XSS-Protection "1; mode=block" always;
```

5.2.7 Prevent sniff mimetype

Stop some attacks by preventing the browser from doing MIME-type sniffing.

```
add_header X-Content-Type-Options "nosniff" always;
```

5.2.8 SSL hardening

These measures are used on `ssl.conf` (https).

5.2.9 Keep only TLS 1.2

Run only TLS 1.2 and disable SSLv2, SSLv3, TLS 1.0, and TLS 1.1 which all have protocol weaknesses and use older cipher suites.

```
ssl_protocols TLSv1.2;
```


5.2.10 Use only strong ciphers

Enable only the strongest ciphers which are recommended by cryptoexperts.

```
ssl_ciphers "TLS13-CHACHA20-POLY1305-SHA256:TLS13-AES-256-GCM-SHA384:TLS13-AES-128-GCM-SHA256:ECDHE-ECD
```

5.2.11 Use more secure ECDH curve

Select only the strongest curves.

```
ssl_ecdh_curve X25519:secp521r1:secp384r1:prime256v1;
```

5.2.12 Defend against the BEAST attacks

Enable server-side protection to stop BEAST attacks which rely on weaknesses in the client ciphers.

```
ssl_prefer_server_ciphers on;
```

5.2.13 HTTP Strict Transport Security

Tell browsers that the connection should always be encrypted.

```
add_header Strict-Transport-Security "max-age=63072000; includeSubdomains" always;
```

5.2.14 Disable compression

Disable the https compression of requests, which can be exploited for BREACH attacks.

```
gzip off;
```


PACKAGING

6.1 Requirements

To package Skyminer, the following items are required:

- A Linux shell (Linux, Git BASH, WSL, ...)
- A functional Maven environment
- Docker
- ZIP

You also need to set the three following variables in your `.bashrc` file:

- `export REGISTRY=192.168.48.22:8083`
- `export REGISTRY_PASSWORD=DjH-SFao6Fx.`
- `export REGISTRY_USER=syminer-ci`

Don't forget to reload the file to apply the changes:

```
source ~/.bashrc
```

6.2 Usage

The scripts needed to build Skyminer are located in *packaging_tools/scripts/build*

6.2.1 Packaging Skyminer

To package a Skyminer release, run

```
sudo -E ./package.sh
```

This will create a ZIP file in the *release* repository containing:

- Container images
- Installation scripts and conf
- Tools

6.2.2 Generating docker images

To build and generate the Skyminer docker image, run

```
sudo -E ./dockerize.sh
```

You can then check the creation of the docker image with

```
docker images grep skyminer/skyminer
```

6.2.3 Building Skyminer locally

To create a local build of Skyminer, run

```
sudo ./build.sh
```

This will create a build in *package/kairosdb*

7.1 Introduction

A Skyminer release must be created at the end of each sprint.

It consists of a folder containing:

- The agents Compass2Skyminer, Epoch2skyminer, and monics2skyminer
- The BIRT engine
- The Skyminer documentation (admin, user, user unified diff and release note)
- The Skyminer Python API
- The release of the Skyminer core

7.2 Step-by-step process

7.2.1 Agents

Update file `release_vars_and_func.sh` (variable `AGENTS`) if new agents version are available in http://protect\T1\textdollarNEXUS_REGISTRY/repository/skyminer/releases/agents

7.2.2 Skyminer document

Check if *SkyminerDocument* is modified and released.

If it is not released:

1. (Optional) Merge changes into master
2. Tag the master branch
3. Wait for the CI to finish

In the project *skyminer-core*, update the file *get-env.sh* with the value of the new version This project is part of the *skyminer-core* service lib.

7.2.3 Skyminer UI

Check if *skyminer-web-ui* is modified and released.

If it is not released:

1. (Optional) Merge changes into master
2. Tag the master branch
3. Wait for the CI to finish

In the project *skyminer-core*, update the file *get-env.sh* with the value of the new version

7.2.4 Docker images

Skyminer docker images are made in two steps:

1. Each of the docker images has its own main image defined on a branch in the *skyminer-docker-images* project.
2. Each of the Skyminer service uses the previously made images to make its own image of the service.

7.2.4.1 Jupyter

Jupyter extensions

The Jupyter service of Skyminer is delivered with the following Jupyter extensions:

- *skyminer-jupyter-extension*
- *jupyter-hidecode-extension*

Check if each Jupyter Extension modifications is merged. If not, merge changes into master. No tags are needed in the projects, jupyter-skyminer docker CI will use the content of the master branch.

Jupyter service

If some Jupyter Extensions have changed or Jupyter version changed:

1. Tag the *skyminer-jupyter* branch of the *Skyminer-docker-images* project
2. Wait for the CI to finish

In the project *skyminer-core*, update the file *get-env.sh* with the value of the new version

7.2.4.2 Grafana

Grafana plugin

Check if *grafana-plugin* is modified and released.

If it is not released:

1. (Optional) Merge changes into master
2. Tag the master branch

3. Wait for the CI to finish

In the project *skyminer-core*, update the file *get-env.sh* with the value of the new version

Grafana service

If the docker image has been modified:

1. Tag the corresponding branch in the *Skyminer-docker-images* project with a new version
2. Wait for the CI to finish

In the project *skyminer-core*, update the file *get-env.sh* with the value of the new version

Note: Grafana service is composed of the Skyminer “internal” plugin and some open source plugins.

7.2.4.3 Other images

This paragraph is applicable to:

- Cassandra
- Collectd
- ddgt
- Kibana
- Nginx
- PostgreSQL

If a docker image has been modified:

1. Tag the corresponding branch in the *Skyminer-docker-images* project with a new version
2. Wait for the CI to finish

In the project *skyminer-core*, update the file *get-env.sh* with the value of the new version

7.2.5 BIRT plugin

If the BIRT plugin has been modified:

1. From the Nexus, download the file */skyminer/releases/birt/BIRT_FULL.tgz*
2. Extract *BIRT/BIRT-Report-Designer-luna-64bit*
3. Replace the jar files in the *dropins* folder with the new ones
4. Re-create the archive *BIRT-Report-Designer-luna-64bit.zip*
5. Re-create the archive *BIRT_FULL.tgz*
6. Push it manually to the Nexus to replace the old one

7.2.6 Skyminer core

All the following steps are applicable to the *skyminer-core* project.

During the release, make sure that nobody will accept merge request until Skyminer next version is set

Prerequisite:

1. Make sure with the rest of the team that all their changes are ok and that the final release is ready to be built
2. Check and/or update the release note in *documentation/release-notes* with all the changes and bugfixes affecting the end user:
 - Add or update the release specific file, sprint (e.g. *2020-S15.rst*) or version (e.g. *version-2.7.rst*),
 - Update the *index.rst* file relative to the release (sprint or version) by adding the previously defined file.
3. If an agent has been modified, update its version number in the file *packaging_tools/scripts/release/release_vars_and_func.sh*
4. Check all the versions defined in *get-env.sh*
5. Check in *get-env.sh* the `SKYMINER_LATEXDIFF_REFERENCE_VERSION` variable:
 - All sprint versions 2020-S17, 2020-S18 are compared with the previous minor version (in such case 2.6),
 - All minor versions are compared with previous minor version (2.7 vs 2.6),
 - All major versions are compared with previous minor version (3.0 vs 2.9),
 - All patch versions are compared with previous minor version (2.6.1 vs 2.6).
6. Edit the *pom.xml* file to set the final version of Skyminer, for sprint (e.g. 2022-S2) or version (e.g. 2.7)

Release:

7. Tag the project with the name of the version. Wait for the CI to finish

Tests:

8. Connect to the deployed version: 192.168.48.29 or 192.168.48.114, and go over each issue to make sure that the changes are applied and working

Delivery:

9. Go to *packaging_tools/scripts/release*
10. Run `./fetch_skyminer_release.sh`. This will create a folder named *skyminer-release-** containing the Skyminer release
11. Go to this folder, check that all the elements composing the release are there (see *Introduction*)
12. In the documentation folder, check that the pdf files can be opened. Skim through them rapidly to make sure the layout is OK
13. Copy the release folder to `\\shackleton\\I53_Skyminer\\Skyminer-Releases`

Prepare next version:

14. Edit the *pom.xml* file with the next Skyminer sprint version with *dev* in the middle (e.g. 2020-dev-S15), make sure it is merged to master branch

SSL CERTIFICATES

8.1 Introduction

SSL stands for Secure Sockets Layer, a global standard security technology that enables encrypted communication between a web browser and a web server. It is used by million of people to decrease the risk of sensitive information from being stolen or tampered with by hackers and identity thieves.

To create this secure connection, an **SSL certificate** is installed on the web server.

It serves two functions:

- Guaranteeing the identity of the website
- Encrypting the data that is being transmitted

When a certificate is successfully installed on the server, the application protocol will change from HTTP to HTTPS.

8.2 Creation

The certificate is generated by the following command:

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout "$INSTALLATION_DIRECTORY/skyminer/nginx_volumes/certs/nginx-selfsigned.key" -out "$INSTALLATION_DIRECTORY/skyminer/nginx_volumes/certs/nginx-selfsigned.crt" -subj "/C=FR/ST=TOULOUSE/O=Kratos/OU=Skyminer/CN=$DOMAIN_NAME" > /dev/null 2>&1
```

It generates a new X.509 private key (-x509, -newkey) using the RSA algorithm with a 2048-bit key length (rsa:2048) without using a passphrase (-nodes), and then creates the key file 'nginx-selfsigned.key' (-keyout).

The command then generates 'nginx-selfsigned.crt' (-out) with the following information (-subj) :

- C: Country
- ST: State
- O: Organization
- OU: OrganizationalUnit
- CN: Common name or FQDN

The resulting certificate has a validity of one year (-days 365).

The domain name is given by the user before generating the certificate. By default, it is set to the public IP of the server.

DEVELOPMENT ENVIRONMENT

Skydev is the dev environment of Skyminer. It uses Kubernetes to run Skyminer.

9.1 Installation and configuration

9.1.1 Login

user: user

password: user

9.1.2 Prerequisites

- Virtualbox
- VBoxManage in your windows path
- Internet connection

9.1.3 How to install

- Go to the skydev folder
- Execute `./download_dependencies.bat` to download the dependencies of the dev env
- Execute `./config_vm.bat` to install the machine
- Edit the script `update_ip.sh` with the IP of your VM (replace the IP address line 9)
- Open VirtualBox and check that your network interface is in bridge mode
- Run the machine and go to `/skydev`
- (Optional) If you have issues with CRLF, run `sudo apt install dos2unix`, followed by `find . -type f -exec dos2unix {} \;`
- and execute `sudo ./update_ip.sh` and `ip addr` to check that you have the correct IP address
- Edit the config file `/skydev/config/machine.py`, set `host.ip` to your computer ip and `server.ip` to your VM ip
- Connect to your machine using SSH
- Run `cd /skydev/` to access to the dev env folder in the machine

- Run `./cli.py -i -rs` to install Skyminer

9.1.4 NG Angular

Change your env with HTTPS

```
ng serve -host=0.0.0.0 -baseHref=/front/
```

9.2 How it works

Skydev uses a very simple template engine to generate kubernetes yml files. These yml files will be loaded into kubernetes to run Skyminer.

Each folder in the `/template/` corresponds to a Skyminer service.

As the system is very complex, Skydev implements a module system to load or unload Skyminer's services in Kubernetes.

Skydev uses two folders: - `/skydev/` which contains all the files to run the system - `/skyminer-data/` which contains all the files to save the state of the system

Skydev uses `microk8s` to run kubernetes (<https://microk8s.io/docs/commands>) and `kubectl` (<https://kubernetes.io/fr/docs/reference/kubectl/cheatsheet/>)

9.3 Module system

Modules contain commands to start, delete, and update the Skyminer services.

They are located in the folder `/skydev/modules/` and are used by the `./cli.py` tool.

For example installing the cassandra module is done with the command: `./cli.py -installmod skyminer/cassandra`

9.4 Config files

The folder `/skydev/config/` contains the configuration files used to run the system.

- `/skydev/`
 - `/config/`
 - `dev.py`
 - `kubernetes.py`
 - `machine.py`

9.4.1 dev.py

Defines whether the system is in development mode.

9.4.2 kubernetes.py

Contains 2 dictionary: `conf_kubernetes` and `conf_kubernetes_dev`.

9.4.2.1 conf_kubernetes

Defines the different container images for the system, the number of replicas, the config files of each services and the secrets.

9.4.2.2 conf_kubernetes_dev

Defines the files and folders you want to mount in the machine from the different containers.

It must have the following structure:

```
{
  "image_name" : [
    {
      "src" : "where in the container",
      "dst" : "where on the machine"
    }, {
      "src" : "where in the container",
      "dst" : "where onthe machine"
    }
  ]
}
```

9.5 Templating Engine Templator

Templator is a very simple templating engine capable of handling variables and if conditions.

9.5.1 Variables

You can define variables in your template like this `{%myvariable%}`.

9.5.2 if conditions

You can define blocks in your template with:

```
{% if myvariable %}  
...  
{% endif %}
```

If `myvariable` is `true` the output file will be generated with the inside of the block.

9.6 Use cases

9.6.1 I want to activate the dev mod of Skyminer

- Edit the config file `/skydev/config/dev.py`, set `dev` to `True`
- Put the image you want to use in the `conf_kubernetes_dev` dictionary in the config file `/skydev/config/kubernetes.py`
- Run `./cli.py --mountdev <myimage>` (Example: `./cli.py --mountdev skyminer:2019-dev-S15`). The cli tool copies the data of the image inside the folder `/skydev/dev/skyminer/`
- Run `./cli.py --deletemod skyminer/skyminer` to remove the current running Skyminer service
- Run `./cli.py --buildtemplate --installmod skyminer/skyminer` to run the Skyminer service with the folder `/skydev/dev/skyminer` mounted.
- Modify the contents of the folder `/skydev/dev/skyminer`, which will update the Skyminer service. If you update a jar file, you will need to run the command `./cli.py --updatemod skyminer/skyminer`

9.6.2 I want to update the kubernetes files of one of my services

- Update files in `/skydev/`
- If the changes are extensive, run `./cli.py --deletemod skyminer/<mymodule>` to remove the module
- Run `./cli.py --buildtemplate` to rebuild the kubernetes yml files
- Run `./cli.py --installmod skyminer/<mymodule>` to reinstall the module

9.6.3 I want to change my number of replicas of one of my services

- Edit the config file `/skydev/config/kubernetes.py`
- Run `./cli.py --buildtemplate --installmod skyminer/<mymodule>`

9.6.4 I want to run Angular dev NG Serve behind Skydev ingress

- Edit the config file `/skydev/config/dev.py`, set `ng.dev` to `True`
- Run `./cli.py --buildtemplate && kubectl apply -f /skydev/build/ingress/ingress-dev.yml`
- Run `ng serve --host=0.0.0.0 --baseHref=/front/` on your computer
- Now you can access to your ng serve with `https://ip_of_your_vm/front/`

To disable it: `kubectl delete -f /skydev/build/ingress/ingress-dev.yml` ## I want to check if my pods are running

- `kubectl get pods`

9.6.5 I want to check the logs of one of my pods

- `kubectl logs id_of_my_pod`

9.6.6 I want to manually delete one of my pod

- `kubectl delete pod id_of_my_pod`

9.6.7 I want to stop Skyminer

- `microk8s.stop`

9.6.8 I want to start Skyminer

- `microk8s.start`

9.6.9 I want to reboot Skyminer

- `microk8s.stop && microk8s.start`

9.6.10 I want to pull from the nexus

- Edit the file `/etc/docker/daemon.json`

```
{
  "insecure-registries": [
    "192.168.48.22:8083",
    "192.168.48.22:8084"
  ]
}
```

9.6.11 I want to add the autocompletion of kubectl

```
source <(kubectl completion bash)
echo "source <(kubectl completion bash)" >> ~/.bashrc
```

9.7 ./cli.py

```
usage: SKCLI [-h] [-i] [--installkubectl] [--uninstall] [--clear] [-rs] [-bt]
            [-cT] [-cSD] [-imod [INSTALLMOD]] [-umod [UPDATEMOD]]
            [-dmod [DELETEMOD]] [-mdev [MOUNTDEV]] [-ddev [DELETEDEV]]
            [-ldi [LOADDOCKERIMAGES]] [-pdi [PUSHDOCKERIMAGES]]

optional arguments:
  -h, --help            show this help message and exit
  -i, --init            Init Skyminer dev environment
  --installkubectl      Install kubectl
  --uninstall           Uninstall Skyminer dev environment
  --clear              Clear docker, kubernetes, build, and /skyminer-data/
  -rs, --runskyminer    Run skyminer
  -bt, --buildtemplate  Build your template in the folder /skydev/template
  -cT, --cleartemplate  Delete the folder /skydev/template
  -cSD, --clearskyminerdata
                        Delete the folder /skyminer-data/
  -imod [INSTALLMOD], --installmod [INSTALLMOD]
                        Run mod in /skydev/modules
  -umod [UPDATEMOD], --updatemod [UPDATEMOD]
                        Update mod in /skydev/modules
  -dmod [DELETEMOD], --deletemod [DELETEMOD]
                        Delete mod in /skydev/modules
  -mdev [MOUNTDEV], --mountdev [MOUNTDEV]
                        Mount dev images
  -ddev [DELETEDEV], --deletedev [DELETEDEV]
                        Delete folder mounted by dev images
  -ldi [LOADDOCKERIMAGES], --loaddockerimages [LOADDOCKERIMAGES]
                        Load all .tar docker images (default: /skydev/images/)
  -pdi [PUSHDOCKERIMAGES], --pushdockerimages [PUSHDOCKERIMAGES]
                        Push all docker images to repo
```


SKYMINER PERFORMANCES

10.1 Performance metrics

Skyminer is by-design a distributed system with no single point of failure. This provides fault tolerance and linear scalability capabilities.

Skyminer performances are typically measured on a cluster with a throughput per node.

10.1.1 Query Throughput

Query Throughput is less scalable than writes because a query is handled by a single node.

The main bottleneck is the serialization of data to the client as JSON (around 150K datapoints/sec). The second bottleneck is data retrieval from Cassandra (around 400K on a single node, 600K on a small cluster)

Table 1: Skyminer performance indicators

Measured item	throughput (datapoints/sec)	concurrency support
CLUSTER: Data aggregation rate	600K	Yes
CLUSTER: Data serialization to client	150K	Yes
Single-node: Data aggregation rate	400K	Yes
Single-node: Data serialization to client	150K	Yes

10.1.2 Ingest Throughput

The sustained ingest (write) throughput is in average about:

- **100K** datapoints/sec using gzipped JSON.
- **50K** datapoints/sec using regular JSON or Telnet API.

10.2 Dimensioning a Skyminer system

Skyminer is dimensioned in terms of write and read nodes (although a node can do both read and writes).

- **Number of write nodes** = Write throughput/sustained write capability

Number of read nodes is different. A Single node can handle many concurrent queries but the read throughput is not scalable per se.

Therefore if it is required to get better performances than 150K (Raw) or 600K (aggregated) datapoints/sec the client has to parallelize its requests by querying different nodes in parallel.

- **Number of read nodes (raw data)** = Expected Read throughput / 150K
- **Number of read nodes (aggregation of data)** = Expected Read throughput / 600K

If the system needs fault tolerance, add 2 more nodes. (if one or two nodes fail, we can still guarantee the full availability and performances).

Note: For example the client needs to query at 5 Million samples for 10 years of data with an aggregation of 10 minutes average in one second. The number of resulting (aggregated) data points is about 525K samples.

- Serialization of the data implies 4 nodes ($4 * 150K > 525K$).
- Querying the data implies 9 nodes ($9 * 600K > 5 \text{ Million}$)

Therefore the system requires 9 nodes to fulfil the requirement for ad-hoc statistics (11 nodes for tolerance to failures **and** sustain the throughput).

If the 10-min average is precomputed 4 nodes are enough.

In both cases the client will have to send as many queries in parallel to the Skyminer system.

10.3 Performance trade-offs

Skyminer is a distributed system supporting concurrent CRUD operations, and out-of-order data insertion. This has some implications because the system needs consistency and scalability.

The design is simple, we directly write data to Cassandra database, and we rely on Cassandra for the distributed aspect of the data.

Having a system that operates on a local filesystem would permit much better performances for non-concurrent read or write workloads... But such a system would not be scalable, or a very expensive developments costs.

However it is possible to tune and improve performances or to adapt to specific use cases.

Use cases that forbid deletion and updates, and/or forbid data written out-of-order would have better performances on another storage than Cassandra.

Those use cases would not happen on distributed system because of network partitions preventing the system to get the data in-time, enforcing for example a rewrite.

10.4 Alternative designs

Instead of Cassandra a library storage (like LevelDB, Apache parquet or similar format) would work well. The easiest would be to plug-in Prometheus and use Prometheus local file storage.

10.5 Paths of improvement

10.5.1 Cassandra

Cassandra can be tuned to adapt to some workloads. For example there are throttling options that can be increased for faster retrievals.

10.5.2 Skyminer

Skyminer could return data in a binary format with a less interoperable but faster serialization/deserialization than JSON.

This section provides information about Skyminer testing.

11.1 API automated testing

11.1.1 Tools

11.1.1.1 Postman

In the context of API development, Postman is a great REST client to test APIs. However Postman is not just a REST Client, it contains a full-featured testing sandbox that lets you write and execute Javascript based tests for your API.

Postman allows you to build, send and save HTTP requests. You can organize those requests in collections.

Postman also manages Environments, which allows you to contextualize variables and execute queries or series of queries in different configurations (typically dev, recipe, prod).

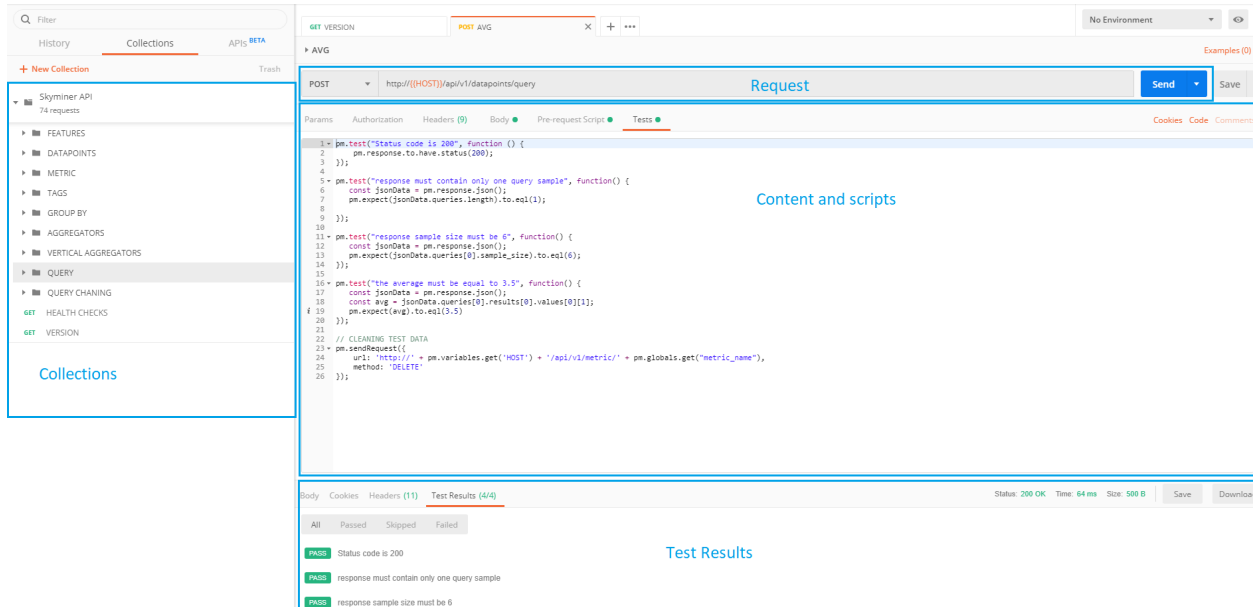
11.1.1.2 Newman

Newman is a command line tool for executing queries and tests of a collection from Postman. It is therefore possible to execute queries outside of Postman. It is really useful for testing API in a continuous integration development mode. This tools allows us to integrate API testing in GitLab CI by adding a new job in the pipeline.

11.1.2 Usage

11.1.2.1 Postman

User interface



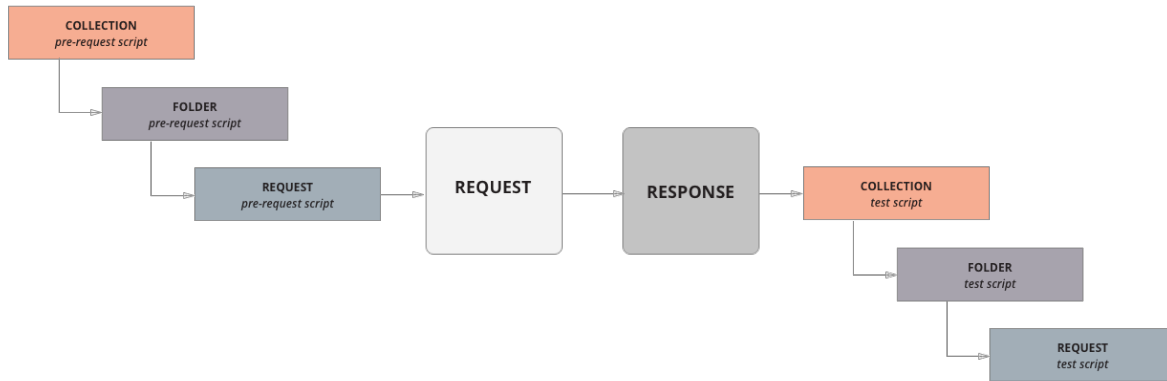
The interface of postman is divided into four parts:

- **Collections:** Postman lets you group requests into collections. That allows you to organize your work. You can also create sub directories in this collection and so on. This is the part you will find on the left side of the application.
- **Request:** This panel allows you build your request to the API. You can define the endpoint and the method (GET, POST, PUT, DELETE ...)
- **Content and scripts:** In this panel you write the body of your request (essentially for POST Request), define headers, write pre-request scripts or the test for the request. You will see examples in the section [Writing a request with test](#).
- **Results:** This panel shows you the results of the query and the tests. The result of query is available by clicking on Body. The results of tests is available by clicking on Test Results.

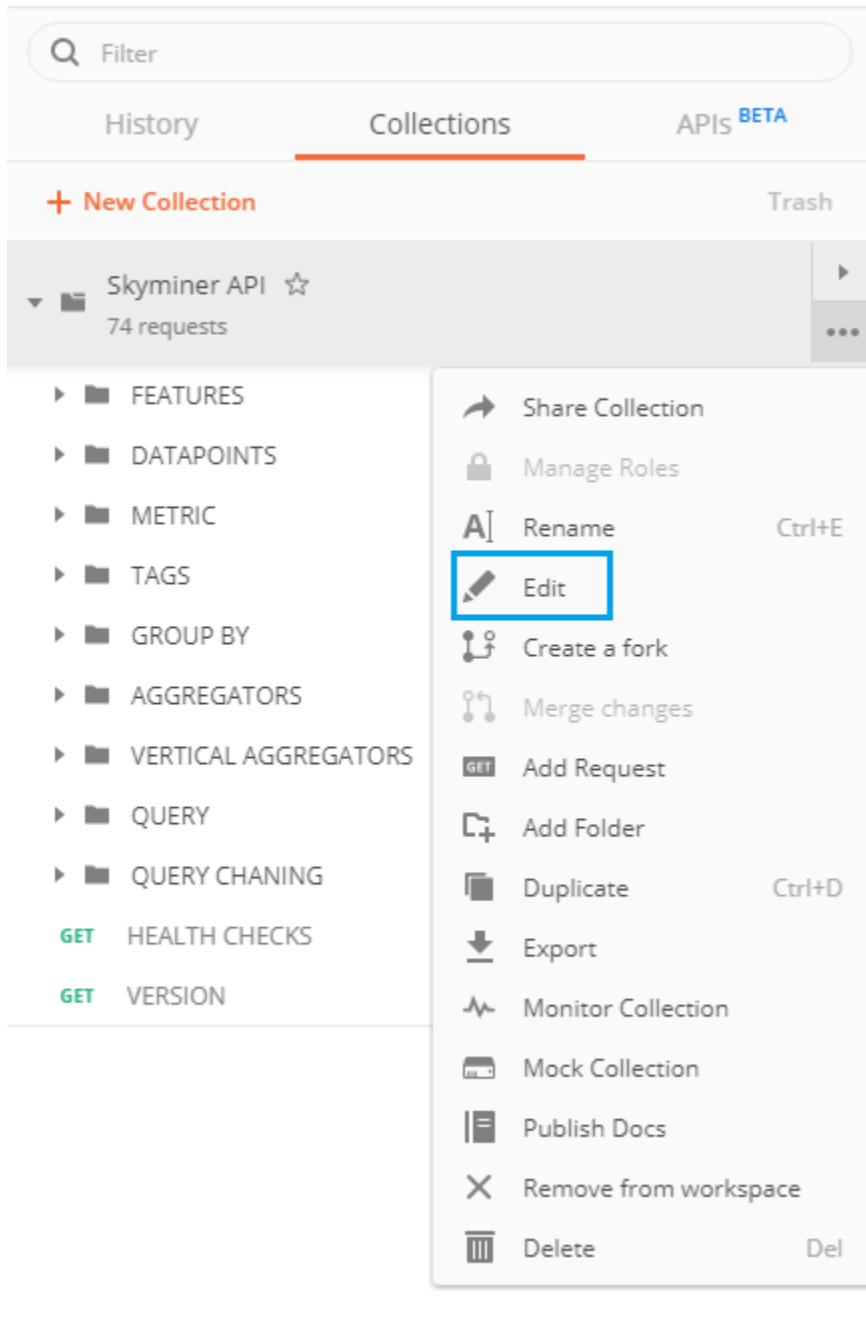
Editing collection

Postman allows you to write tests, pre-request scripts and define variables for a collection or a directory in a collection. All of this will be applied for all the requests of the collection.

The schema below shows you in what order all the scripts are executed.



In our case we define a variable called `HOST` that corresponds to the IP of our test server where Skyminer is deployed and a variable called `METRIC_TEST_NAME` that corresponds to our test metric.



EDIT COLLECTION

×

Name

Skyminer API

Description

Authorization

Pre-request Scripts

Tests

Variables

●

These variables are specific to this collection and its requests. [Learn more about collection variables.](#)

	VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ	...	Persist All	Reset All
<input checked="" type="checkbox"/>	HOST	192.168.48.29	192.168.48.29			
	Add a new variable					

ⓘ

Use variables to reuse values in different places. Work with the current value of a variable to prevent sharing sensitive values with your team. [Learn more about variable values](#)

×

Cancel

Update

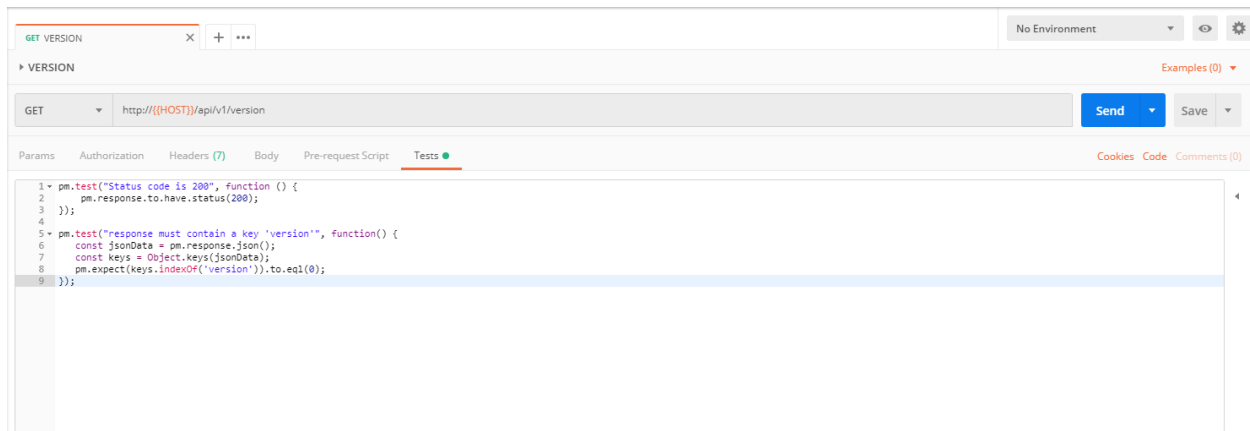
Writing a request with test

A Postman test is essentially JavaScript code executed after the request is sent, allowing access to the `pm.response` object. For more information you can check [examples](#) of Postman tests

Get Request

In order to create a GET request you need to choose GET in the dropdown (5 in the overview section, note that GET is the default choice). Then you need to write the endpoint of the API. Here we use `{{HOST}}` that refers to the collection variable we defined earlier.

For this test nothing is needed in the Body tab, the Header tab or the Pre-request Script tab.



For this request there are two tests.

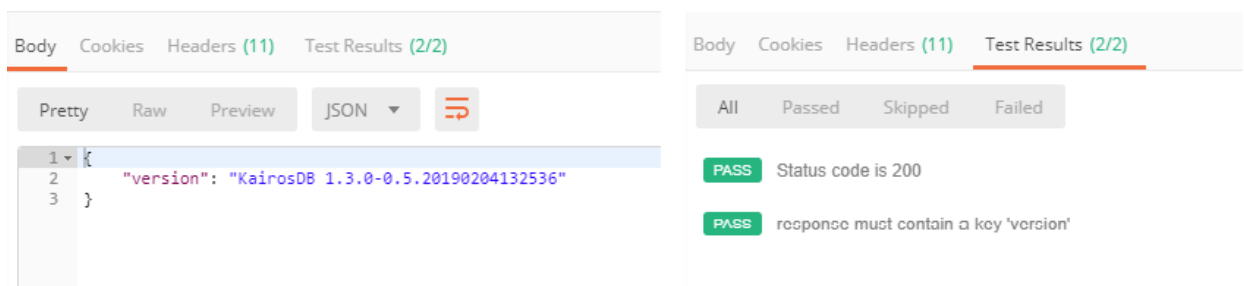
The first one checks the returned code status. Here we expect to receive 200 for success.

```
pm.test("Status code is 200", function () {
  pm.response.to.have.status(200);
});
```

The second one checks if there is a key version in the body of the response from API.

```
pm.test("response must contain a key 'version'", function() {
  const jsonData = pm.response.json();
  const keys = Object.keys(jsonData);
  pm.expect(keys.indexOf('version')).to.eql(0);
});
```

The results of the API call and tests are displayed on the bottom of the page. The body panel shows you the result of the request. The Tests Results panel show you tests that passed or not.



Post Request

In order to create a POST request you need to choose POST in the dropdown (5 in the overview section). Then you need to write the endpoint of the API. Here we use `{{HOST}}` that refers to the collection variable we defined earlier.

The request in the example below asks for an average aggregation on one hour of data. You will need to write a body in Json format, so you need to add a header: `Content-type: application/json` (see picture below).

Params		Authorization		Headers (1)		Body		Pre-request Script		Tests	
▼ Headers (1)											
		KEY				VALUE					
<input checked="" type="checkbox"/>		Content-Type				application/json					

The body corresponds to a query you can find in the Skyminer UI.

Params		Authorization		Headers (1)		Body		Pre-request Script		Tests	
<div> <input type="radio"/> none <input type="radio"/> form-data <input type="radio"/> x-www-form-urlencoded <input checked="" type="radio"/> raw <input type="radio"/> binary <input type="radio"/> GraphQL <small>BETA</small> <input checked="" type="radio"/> JSON (application/json) </div>											
<pre> 1 { 2 "start_absolute": 1561888800000, 3 "end_absolute": 1561892400000, 4 "metrics": [5 { 6 "name": "test_api_metric_name", 7 "aggregators": [8 { 9 "name": "avg", 10 "align_end_time": false, 11 "align_sampling": true, 12 "align_start_time": false, 13 "sampling": { 14 "value": "1", 15 "unit": "HOURS" 16 } 17 } 18] 19 } 20] 21 }</pre>											

In our case we want to test that the average aggregator works fine on a test metric called `test_api_metric_name`. But this metric does not exist on the test server of Skyminer. We need to add data for this metric. To do so, we use Pre-request script.

Params	Authorization	Headers (1)	Body ●	Pre-request Script ●	Tests ●
<pre> 1 const host = pm.variables.get('HOST'); 2 3 // building of a dataset of 6 points 4 let datapoints = []; 5 const number_of_points = 6; 6 // start_absolute - 1 hour 7 const first_timestamp = 1561888800000; 8 // end_absolute + 1 hour 9 const last_timestamp = 1561892400000; 10 const gap = Math.round((1561892400000 - 1561888800000) / (number_of_points + 1)); 11 12 for(i = 1; i <= number_of_points; i++) { 13 datapoints.push({ 14 "name": pm.globals.get("metric_name"), 15 "timestamp": first_timestamp + gap * i , 16 "value": i, 17 "tags": { 18 "host": "host1" 19 } 20 }); 21 } 22 // End of dataset building 23 24 pm.globals.set("metric_name", "test_api_metric_name"); 25 26 // Adding datapoints 27 pm.sendRequest({ 28 url: 'http://' + host + '/api/v1/datapoints', 29 method: 'POST', 30 header: { 31 'Content-Type': 'application/json', 32 }, 33 body: { 34 mode: 'raw', 35 raw: JSON.stringify(datapoints) 36 } 37 }); 38 39 setTimeout(function(){}, 1000); </pre>					

In the script we generate 6 data points for the metric on a time range of one hour.

- `pm.globals.set` allows you to define a global variable which is accessible from tests.
- `pm.sendRequest` allows you to send a request to the API.

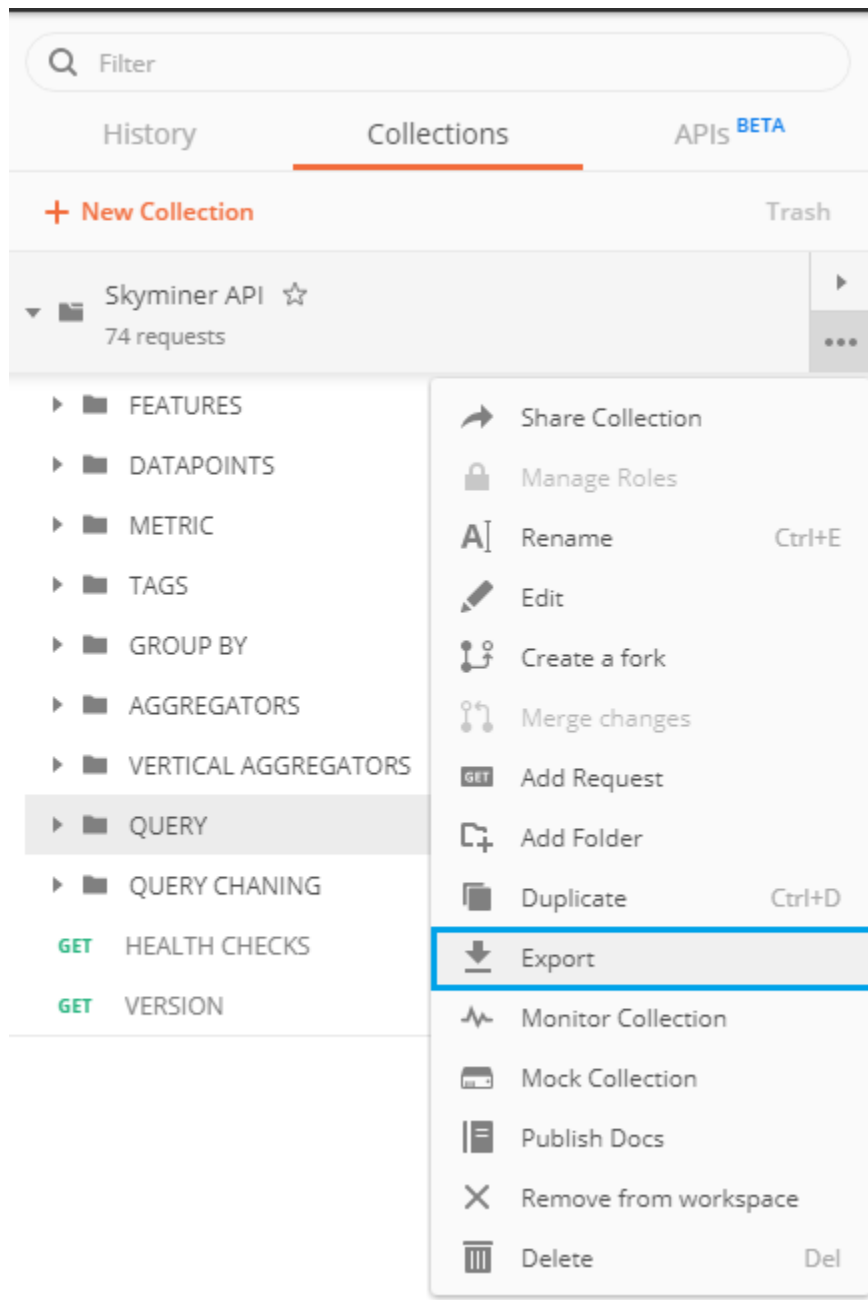
Here is the test for the request:

Params	Authorization	Headers (1)	Body ●	Pre-request Script ●	Tests ●
<pre> 1 pm.test("Status code is 200", function () { 2 pm.response.to.have.status(200); 3 }); 4 5 pm.test("response must contain only one query sample", function() { 6 const jsonData = pm.response.json(); 7 pm.expect(jsonData.queries.length).to.eql(1); 8 9 }); 10 11 pm.test("response sample size must be 6", function() { 12 const jsonData = pm.response.json(); 13 pm.expect(jsonData.queries[0].sample_size).to.eql(6); 14 }); 15 16 pm.test("the average must be equal to 3.5", function() { 17 const jsonData = pm.response.json(); 18 const avg = jsonData.queries[0].results[0].values[0][1]; 19 pm.expect(avg).to.eql(3.5); 20 }); 21 22 // CLEANING TEST DATA 23 pm.sendRequest({ 24 url: 'http://' + pm.variables.get('HOST') + '/api/v1/metric/' + pm.globals.get("metric_name"), 25 method: 'DELETE' 26 }); </pre>					

Note: data **is** deleted **in** order **not** to pollute the Skyminer database **and** to start **from 0** **↳ for** each test set.

Export your collection

In order to work with Newman you will need to export your collection in json format.



11.1.2.2 Newman

Utilisation

Newman, in Skyminer, is essentially used in CI. It allows us to know if everything works fine before delivery. Newman is executed in a docker environment only when you tag the repository. You can use it on your environment, for this you will need to install it.

```
npm install -g newman
```

And then you only need to run it with your collection you export with Postman.

```
newman run mycollection.json
```

11.2 Manual testing

11.2.1 Scenario

1. Connect to Skyminer
2. Use dashboards (Grafana)
 1. Open ModCod Analysis dashboard, change carrier using list selector on top
 2. Open "SLA Monitoring" dashboard, change carrier using list selector on top
 3. Open "Incident statistics" dashboard
 4. Open "Incident site details" dashboard
 5. Open some random other dashboards, check contents
 6. Edit an existing dashboard, check features work
 7. Create a new dashboard with new query, check features work
 8. create alerts - test alerting
1. Use query UI
 1. Select a metric
 2. Check first / last samples are displayed
 3. Add tag filtering
 4. Add group-by
 5. Query and plot data, check plot
 6. Test query to correlations
 7. Test query to jupyter
 8. Test export to CSV in the available formats (open in excel for check)
 9. Test time override (remove it afterwards)
10. Test event-based query
11. Test search features
12. Test search metric
 1. Use Jupyter notebooks
 1. Test Skyminer plugin : create a query
 2. Test Skyminer plugin : edit a query
 3. Test Skyminer plugin : get & plot data
 4. Test export to PDF - with some code, without code
 5. Test notebook: HPA degradation
 1. Use correlations

1. Mode: search
2. Reference query: 1 week of data from sunday 2 weeks ago 0:00 to last Sunday 00:00 select skyminer.demo.carrier_eirp, filter by tags: carrier=carrier_1000, group-by tag: carrier, aggregator: average 30min align sampling & align start time, remove safeguards
3. Search series: same query, just remove the filter with carrier_1000
4. Similarity measure : discarding linear with reliability at 10
5. Results format: Threshold = 0.5, return best = 50
6. Computation method = threaded
7. Launch query, check there are good correlations (with perfect score of 1.0 for carrier_100 to itself), and some correlations with good scores but bad reliability (appear in red)
8. use interactive features (mouse over, click)
9. Now Select Mode: matrix
10. Same searched series: 1 week of data from sunday 2 weeks ago 0:00 to last Sunday 00:00 select skyminer.demo.carrier_eirp, filter by tags: group-by tag: carrier, aggregator: average 30min align sampling & align start time, remove safeguards
11. Similarity measure : discarding linear with reliability at 50
12. Results format: Threshold = 0.5, return best = 100, uncheck “Return perfect scores”
13. Computation method = memory cache threaded
14. Launch query, check there are good correlations, and possibly some correlations with good scores but bad reliability
15. use interactive features (mouse over, click)
16. Do the same but enable checkbox “Return perfect scores”
 1. Upgrade / rollback test
 1. Ensure there’s some data and dashboards / notebooks
 2. Upgrade to new release
 3. Perform tests to check that data , dashboards and notebooks are still present and work
 4. Rollback to previous release
 5. Perform tests to check that data , dashboards and notebooks are still present and work

SKYMINER EXPERT FEATURES API

12.1 POST: `/api/skyminer/expert-features/query/results_as_insert_request`

Outputs the results of the query as a JSON document that can be used as a POST request on Skyminer. Accepts a JSON query. Response is 200 with a JSON document.

Metrics can be renamed using alias aggregator in the source query. More tags can be added using tags aggregator in the source query.

12.1.1 Tags management

- Tags from original data are by default included.
- If several values are associated to a tag key they are concatenated with " / " separator with a limit in number and size of generated value.
- it can be disabled by providing `expert.include_tags` set to `false` in the query
- In all cases the different groups definitions are added as tags (the same way that the detailed CSV results columns are built)

12.2 POST: `/api/skyminer/expert-features/query/save_results`

Save results of the query in Skyminer. Accepts a JSON query. Response is 204 (No content).

The results are saved as a new metric name and include a tag `source_product` with value `skyminer_query_results`.

The new metric name is computed as follows:

- A configurable prefix, by default `skyminer.saved_results`.
- A configurable suffix, by default an empty string

Prefix and suffix can be respectively changed to another value by setting `expert.save_as_metric_prefix` and `expert.save_as_metric_suffix` to a valid string value in the query JSON. The string can be empty.

Tags are managed the same way as for the `results_as_insert_request` feature above

TTL, (Time to live of the data) if supported by datastore can be set using the `expert.save_as_metric_ttl_seconds` property to an integer value in seconds, by default the TTL is set to zero (no expiry).

Metrics can be renamed using alias aggregator in the source query. More tags can be added using tags aggregator in the source query.

DESIGN NOTES

This section provides information about WIP design of some upcoming components or features.

13.1 Authentication

This page describes authentication lists system.

13.1.1 Mechanism

Authentication is done using OAuth2 that provides SSO (Single Sign-On) for the platform.

This requires an authentication server.

If integrated with KMP, the authentication server will be the KMP one (either using a proxy in *Skyminer* containers, or by directly accessing it).

For development purposes (and maybe also production), *Keycloak* is the chosen one (free and opensource).

13.1.2 Why Keycloak?

Keycloak is an open source software product to allow single sign-on with Identity and Access Management aimed at modern applications and services. As of March 2018, this *JBoss* community project is under the stewardship of *Red Hat* who use it as the upstream project for their *RH-SSO product*.

Among the many features, *Keycloak* includes:

- User registration
- Social login
- Single Sign-On/Sign-Off across all applications belonging to the same Realm
- 2-factor authentication
- LDAP integration
- Kerberos broker
- Multitenancy with per-realm customizable skin

There are 2 main components:

- Server
- [Application adapters](#) (many servers supported, a gatekeeper is also provided and already dockerized)

Please note that *Keycloak* supports both *SAML* and *OpenId* protocols.

For Skyminer, we use Keycloak version 22.0.4. Earlier versions had a bug on URL containing brackets, which is problematic for skyminer as some links contain json parameters. If Keycloak is deployed with Skyminer, it is done with `<installation_directory>/keycloak/docker-compose.yml`.

All clients are created in Skyminer Realm:

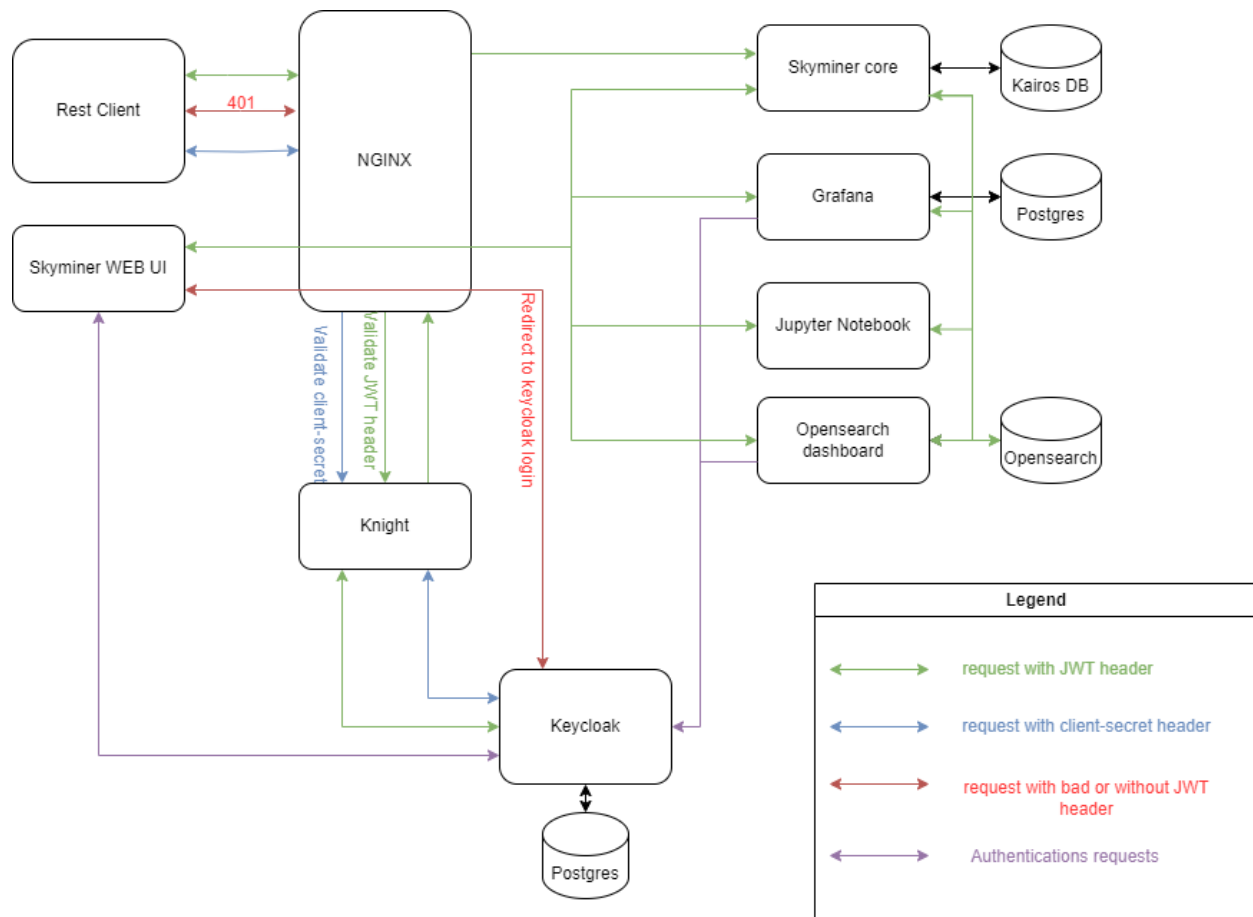
- **skyminer-service-account**: Client to authenticate outside of the context of a user. It uses the OAuth 2.0 Client Credentials Grant Type, which allows to obtain JWT token without user name/pwd but using client-secret.
- **skyminer-grafana**: Client to authenticate Grafana.
- **skyminer-opensearch**: Client to authenticate Opensearch and Opensearch dashboard.
- **skyminer-query-ui**: Client to authenticate Skyminer front.

Keycloak is deployed with 3 demo users :

- **skyminer-admin**
- **skyminer-editor**
- **skyminer-viewer**

13.1.3 How authentication is implemented on Skyminer scope?

Here is a view of Skyminer architecture in authenticated scope:

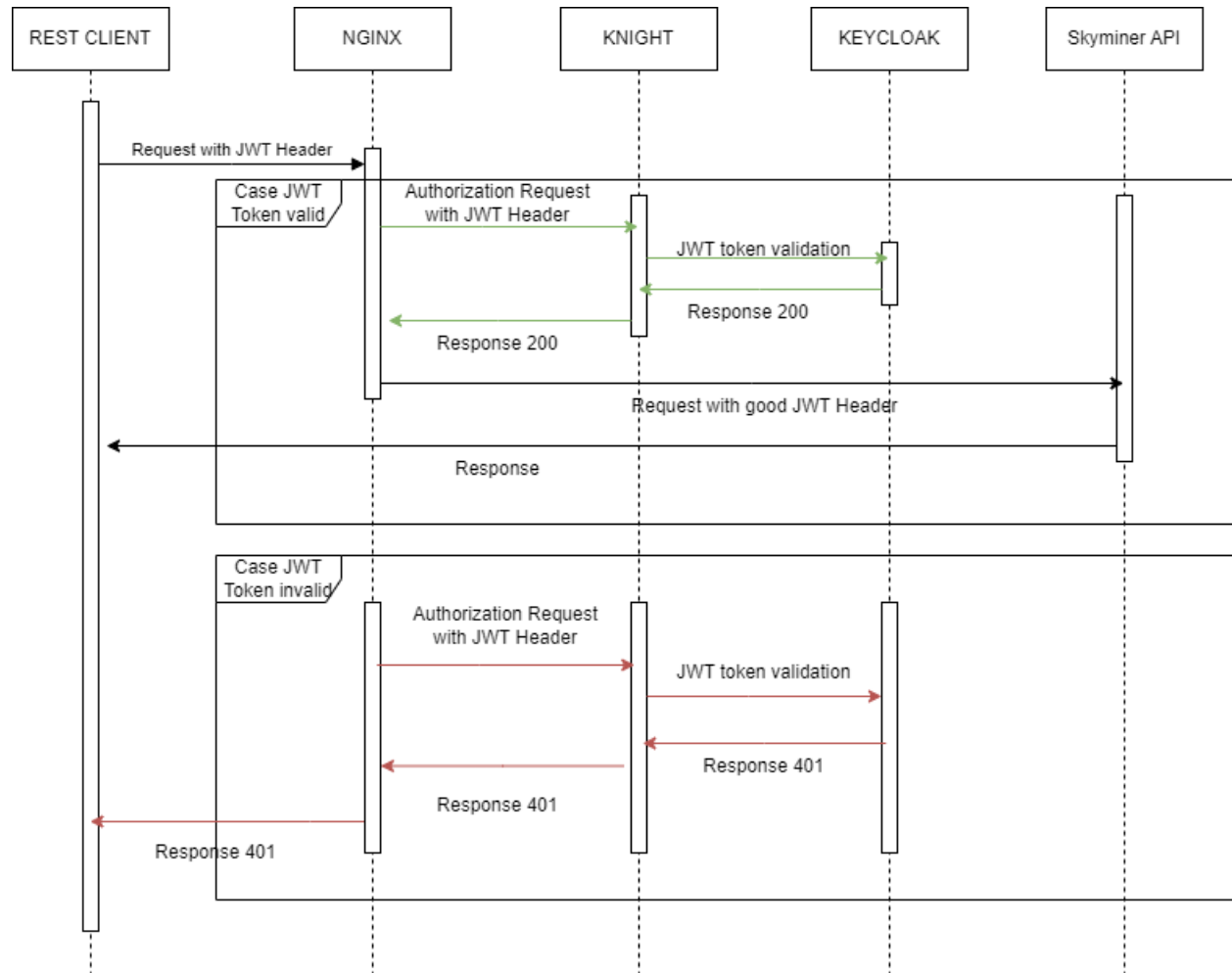


13.1.3.1 Knight

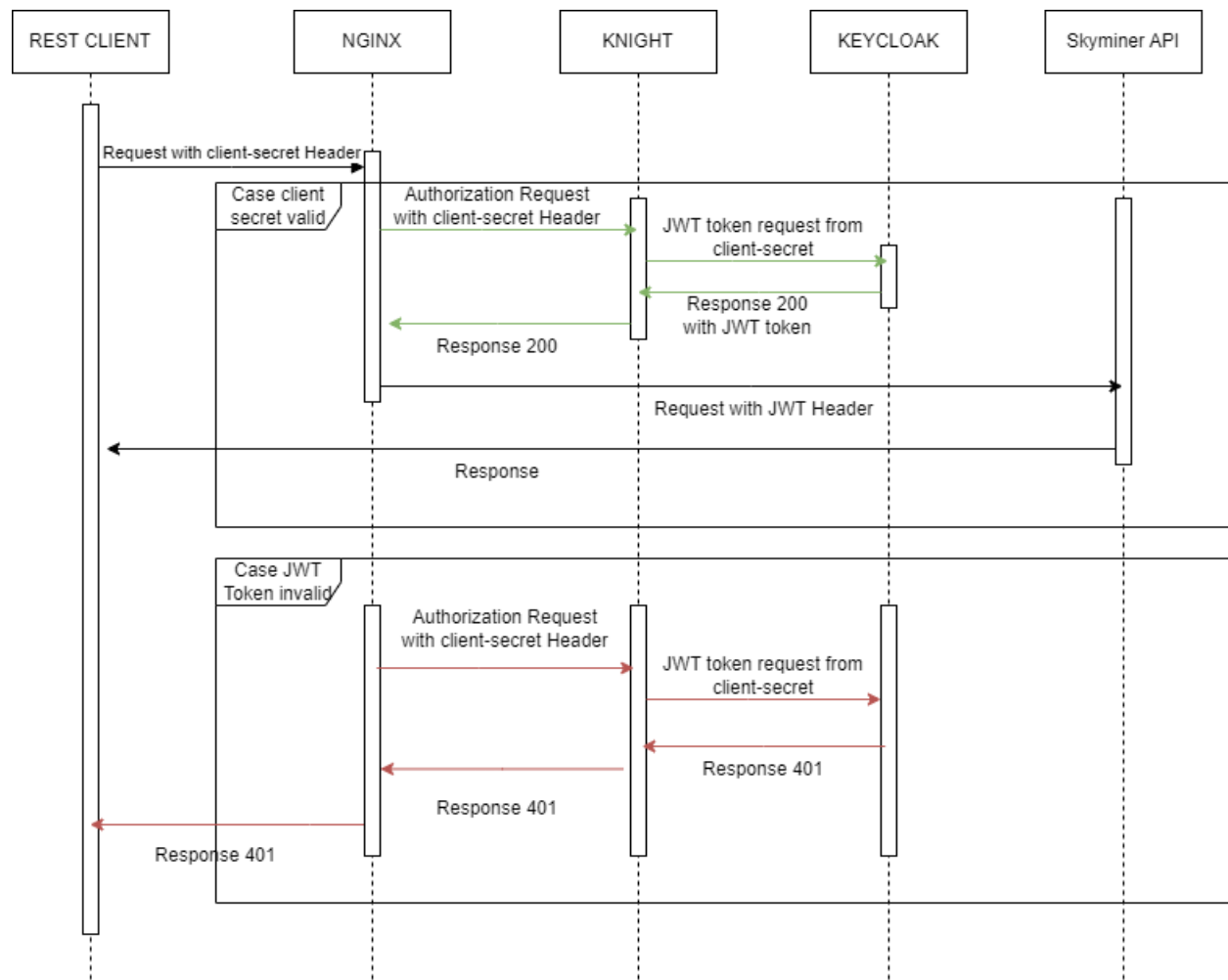
Knight is the security gateway used by Skyminer. In Skyminer scope, it has 2 ways of validating a request:

- Validate JWT token in request Authorization Header.
- Check presence of 'client-secret' header in request and get a token from Keycloak using said client-secret. This use case should be used by service clients.

Here is a sequence diagram of JWT validation process:



Here is a sequence diagram of client-secret validation process :



Knight is configured in file `<installation_directory>/skyminer/knight/config.json`. Please see https://gitlab-toulouse.kratos.us/research_and_development/openspace-service-knight for more information on configuration.

Knight uses a cache to avoid doing too much requests to Keycloak.

13.1.3.2 Securing Skyminer API

Skyminer API and skyminer-document-api are secured by Knight. Human clients should get a JWT from Keycloak, and then request API. Service client should use client-secret in their header. This secret is available on Keycloak UI (realm Skyminer, client skyminer-service-account, credentials tab).

Endpoints `/api/skyminer/version` and `/api/skyminer/license` are not protected, as they are used by Skyminer homepage which is not secured.

13.1.3.3 Securing Skyminer WEB UI

Skyminer WEB UI uses Keycloak.js library:

- If user is not authenticated, he is redirected to Keycloak login page.
- If user is authenticated, he has access to Skyminer Query Web UI and each request done to backend has JWT header.

This authentication use the keycloak skyminer-query-ui client.

Skyminer front is configured in file `<installation_directory>/skyminer/front/config/config.json`

13.1.3.4 Securing Grafana

Grafana supports OAUTH2 authentication. We only have configuration to secure Grafana with Keycloak. Configuration is done in file `<installation_directory>/skyminer/grafana/env/grafana.env`, using environments variables. All environments variables beginning with `GF_AUTH` concern grafana authentication. Grafana uses Keycloak skyminer-grafana client.

When user is authenticated, a Grafana session is created and managed in cookie.

13.1.3.5 Securing Opensearch and Opensearch Dashboard

Opensearch authentication is configured in `<installation_directory>/skyminer/opensearch/config/opensearch-security/config.yml` file. Part `openid_auth_domain` contains OpenId authentication configuration.

Opensearch dashboard authentication is configured in `<installation_directory>/skyminer/opensearch-dashboard/config/opensearch_dashboards.yml` file. Part `opensearch_security` contains authentication configuration.

Opensearch and opensearch-dashboard use Keycloak skyminer-opensearch client.

When user is authenticated, an Opensearch dashboard session is created and managed in cookie.

13.1.3.6 Securing Jupyter notebook

In progress. Jupyter notebook must be upgraded to JupyterLab to properly manage authentication with Keycloak.

Jupyter Notebook uses his legacy authentication, and request done to backend have to use the client-secret. Development is not done yet.

13.2 Access Control

To be determined. Find below previous notes.

13.2.1 Mechanism

Role-based ACL are defined. Users are mapped to roles, and all roles have access rules defined.

Case of authorized data query:

Case of unauthorized data query:

13.2.2 Time Series ACL

The Time series ACL is complex to integrate in the security gateway.

There are two components:

- metrics filters
- tag filters

Metrics & tags are defined together. Only one filtering is applied, if it exists the exact metric_name match is used by the filtering mechanism, otherwise the last rule is applied.

Therefore we can define first generic rules (with all match regexp) with lowest order then specific rules.

If a user belongs to different roles the metric filters rules are merged (order-preserving). For example if user is role1 and role2, the rules are merged and applied in their respective order. If the priority_order is not defined then the position of the rule is used.

Time filters are applied only if ranges are defined:

- if start time is outside a range: change start time to start time of next range
- if end time is outside a range: change to end time of previous range
- if start time and end time are both after last allowed range: reject query
- if end time and start time are before first allowed range: reject query
- if start and end belong to different ranges: reject query

For meta data queries (tag queries) if time ranges defined:

- set start time to the maximum between provided start time and start time of first range
- set end time to the minimum between provided end time and end time of last range

Example of a role definition in JSON:

```
{
  "role": "A_ROLE_NAME",
  "time_series_acl": [{
    "priority_order": 1,
    "metric_regexp": ".*",
    "time_range_filters": [{
      "name": "first_range",
      "start_time_epoch_millis": 0,
      "end_time_epoch_millis": 1580000600000
    }, {
      "name": "another_range_ending_in_2099",
      "start_time_epoch_millis": 1584489600000,
      "end_time_epoch_millis": 4077475200000
    }
  ]
},
```

(continues on next page)

(continued from previous page)

```

        "tag_filters": [{
            "tag_name": "satellite",
            "tag_values": ["sat1"]
        },
        {
            "tag_name": "source_product",
            "tag_values": ["epoch", "compass"]
        }
    ],
    {
        "priority_order": 10,
        "metric_name": "metric2",
        "tag_filters": []
    },
    {
        "metric_name": "eirp",
        "tag_filters": [{
            "tag_name": "satellite",
            "tag_values": ["sat1", "sat2"]
        }]
    }
]
"document_acl" : [
    ...
],
"object_acl" : [
    ...
]
}

```

13.2.2.1 Metrics filters

Only some metrics could be allowed for a role. The rules support regular expressions (or alike).

If the metric is not allowed the gateway returns a FORBIDDEN error to the client. If the metric is allowed the query goes through an additional tag filter

13.2.2.2 Tag filters

Tag filters are defined with a whitelist of allowed tags (no regular expression possible since it's not supported by *Skyminer*)

If the user specifically requests a non-allowed tag in the query tag filters the query is rejected (FORBIDDEN) Otherwise the gateway if needed injects appropriate tag filters into the query.

Note: Example: user toto is allowed to see only metric1 with tag satellite=sat1 and metric2 with no tag filter.

- If toto requests metric3 the query is rejected by the security gateway.
- If toto requests metric1 without filter the satellite=sat1 filter is automatically injected by the security gateway
- If toto requests metric1 without filter the satellite=sat2 the query is rejected by the security gateway

- If toto requests metric2 without filter no filtering is applied
-

13.2.2.3 Results filtering

The security gateway filters the metadata in the results so unauthorized data shall not be visible to the user. It is implemented as a simple post-processing of the JSON results.

There are two filters:

- Filtering metric names on metric queries (/api/v1/metricnames)
- Filtering tag results on tag queries (/api/v1/datapoints/query/tags)

Only the matching metrics are kept for metric names filtering. Only the authorized tag values are kept for filtered metric tags.

Note: Example: user toto is allowed to see only metric1 with tag satellite=sat1 and metric2 with no tag filter.

- If toto requests metric3 the query is rejected by the security gateway.
 - If toto requests metric1 without filter the satellite=sat1 filter is automatically injected by the security gateway
 - If toto requests metric1 without filter the satellite=sat2 the query is rejected by the security gateway
 - If toto requests metric2 without filter no filtering is applied
-

13.2.3 OpenSearch & Document API ACL

For documents, the API will be filtered using the same mechanisms as for the time series:

- allow/forbid indices (same mechanisms as for metrics)
- allow only some key-values for indices (this will be similar to tag filtering)

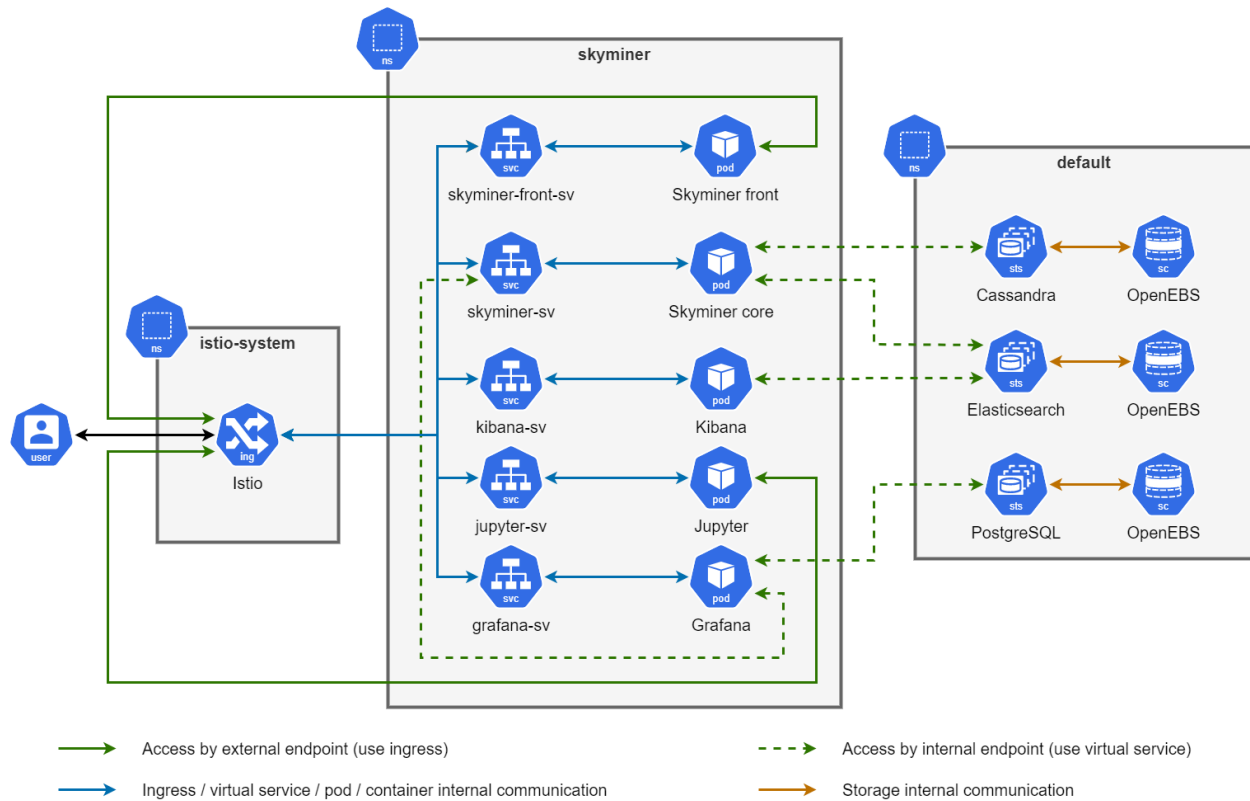
13.2.4 Object API filters

No object API is currently implemented yet in *Skyminer*. If it happens the mechanism will be identical using meta data with a concept of indices and key-value pair labels (or tags).

13.3 Kubernetes

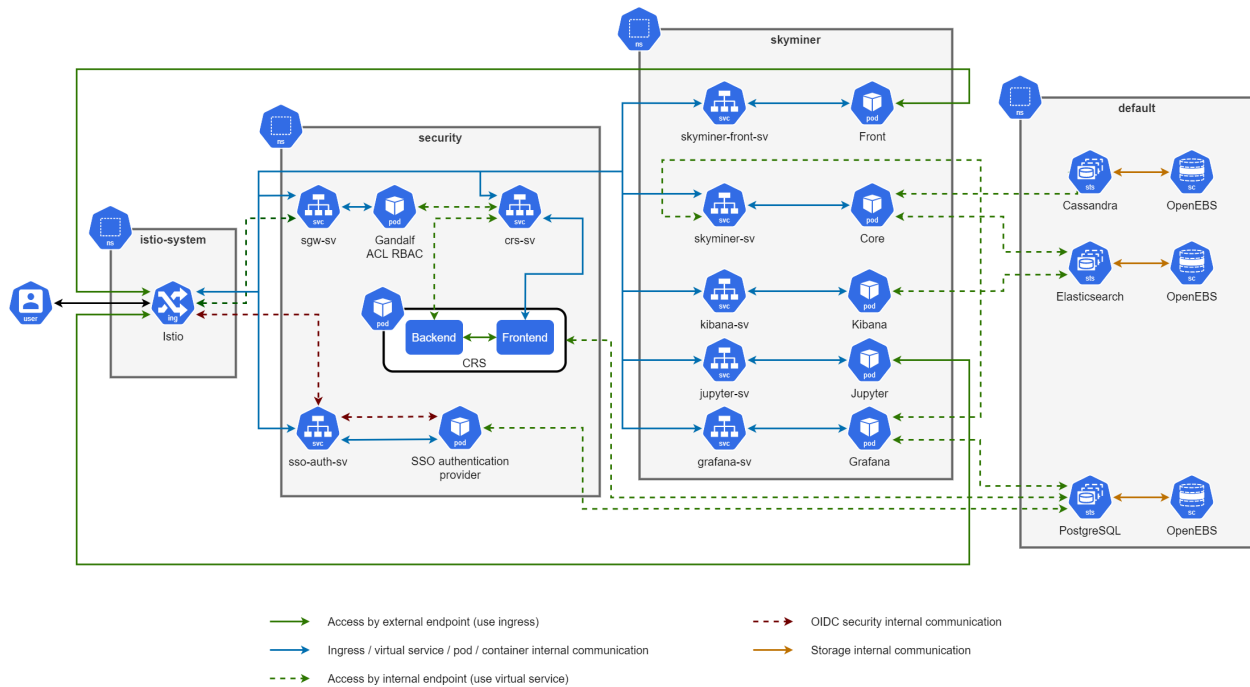
13.3.1 Basic deployment

Here is a graph of a kubernetes deployment of *Skyminer* without SSO.



13.3.2 SSO deployment

Here is a graph of a kubernetes deployment of *Skyminer* with SSO.



13.4 RocksDb datastore

RocksDB is a local filesystem storage system. It works as a key-value storage system.

This is a perfect candidate for storing data for single node or embedded systems.

Moreover RocksDB is included in a Java wrapper in a Maven artifact (For Linux & Windows x64) with JNI bindings available rightaway !

Schema as described below for experimentation of a simple data store for Skyminer.

cf.

- <https://github.com/facebook/rocksdb/wiki/RocksJava-Basics>
- <https://github.com/facebook/rocksdb/wiki/RocksDB-FAQ>
- <https://github.com/facebook/rocksdb/>
- Can be used in Java using <https://mvnrepository.com/artifact/org.rocksdb/rocksdbjni>

13.4.1 RocksDB Settings

Compaction needs to be a Leveled compaction strategy Other settings and tuning TBD after experiments

13.4.2 General aspects

Each time series (unique set of metric name, type and tags) is assigned a unique ID (incremental). Given the fact that this datastore is aimed to be a local filesystem, there is no conflict in ID assignement.

The datastore will cache:

- metric names (all)
- time series ID (only recently pushed ones)

Queries & deletes can be done by range scans (rocks iterators).

13.4.3 Operations

query metric names:

- metrics: return all values (cached)

query metric tags:

- time series index: range scan for matching time series

Add data point:

- metrics: check metric exists, record it otherwise
- time series index: check if time series exist (unique combination of metric + tags key/value pairs)
- time series index: create time series ID if needed
- data: insert data point

Query database:

- time series index: range scan for matching time series

- data: for each series in data: range scan for matching data points (using time interval)

Delete data points:

- time series index: range scan for matching time series
- data: for each series in data: range delete for matching data points (using time interval)

Delete metric:

- time series index: range scan for matching time series
- data: for each series in data, range delete for matching data points (using time interval)
- time series index: delete for matching time series

13.4.4 Metrics CF

Key : metric name (UTF String) value: none

13.4.5 Time Series Index CF

Key: Series decriptor in JSON encoded to bytes (with metricname , type , and tags) value: series ID (long, using VLE)

13.4.6 Series reverse index CF

Key: series ID (long, using VLE) Value: Series decriptor in JSON encoded to bytes (with metricname , type , and tags)

13.4.7 Data CF

Each row is one data point.

Key: series ID (long, using VLE), timestamp ms (long, full 8 bytes) value: value (depends)

13.4.8 Service keystore CF

Key: service (UTF String), service key (UTF String), key (UTF String) Value: value (UTF string), last change timestamp (long 8 bytes)

13.4.9 Service keystore change CF

This column family is required since a delete should also update last modification time of a service/service_key.

Key: service (UTF String), service key (UTF String) Value: last change timestamp (long VLE)

13.4.10 Compressed Data CF (possibility for later)

Each row is 5-minutes bucket.

timestamp base = 5 minutes increments from Epoch time in seconds

Key: series ID (long, VLE), timestamp base (long, VLE) Where to put a flag for compression - and what kind of compression value: timestamp offset, value offset

Bloom filter on 8 bytes : series ID, portion of timestamp base ?

The key shall include a flag to indicate how data is compressed/encoded (Last bits of timestamp base? A reserved byte?)

Compression: delta of delta compression for timestamps and long values, Xor+VLE compression for doubles - Like Gorilla

13.5 High Cardinality Series

13.5.1 High-cardinality problems

High cardinality series cause several problems.

13.5.1.1 Slow query performance in Cassandra

Queries are done in 2 steps: gather row keys then read row When too many rows to read Cassandra IUO is intensive and it becomes slow

- Cassandra has to read as many row keys as time series and time buckets from disk
- KairosDB puts the time series data in cache on disk
- KairosDB Reads data from disk

Eventually it can take minutes to get 70-100k row keys (as measured on EBS at 3K IOPS and 250MiBps limitations)

13.5.1.2 UI Responsiveness

- UI asks for first & last samples - it can take a long time on high cardinality as it has to read as many rows
- UI asks for all tags keys values (in absolute) and in case of tag filters tag per tag - this can involve lots of operations on server and big response payload

13.5.1.3 UI usability

- The UI becomes unresponsive waiting for first & last sample
- Too many tags cause issues to display the values

13.5.2 Proposed improvements

13.5.2.1 Query cardinality limit

Complexity: Low

Impact: High

Probability of implementation: **Certain**

Priority: **High**

Purpose: Do not allow query to hang for too long to keep user experience correct

- Add a parameter to limit the number of time series in a query (configurable limits in skyminer properties file e.g. skyminer.query.max_numbner_of_time_series), by default use 10,000
- Add an option to override this limitation in the safeguards and query payload: max_numbner_of_time_series
- When limit is reached, the query fails and returns a message such as “Number of time series requested (X) exceeds allowed maximal of Y, please add tag filter to reduce number of time series involved”

13.5.2.2 Data type filter

Complexity: Low

Impact: Low

Probability of implementation: **Certain** (provides awareness and control on results)

Priority: **High**

Purpose: Add better user control on query

- Add an optional parameter in query payload (part of metric): e.g. data_type
- This field allows to use an API data type (e.g. number, string, complex, ... etc as available in the data types API)
- This can be used to reduce cardinality in case the metric has several data types

13.5.2.3 Time series count API endpoint

Complexity: Low

Impact: High

Probability of implementation: **Certain**

Priority: **High**

Purpose: Add better information to user ahead of time to improve experience

- Add a time-series search capability in Skyminer custom REST services
- Add an interface the indexed Cassandra datastore and the RocksDB datastore will implement
- This will leverage Cassandra row keys / RocksTB TS Index
- This will leverage OpenSearch ts-index if it exists

e.g. /api/skyminer/time-series-query

13.5.2.4 UI Improvements: series limit

Complexity: Low

Impact: Low

Probability of implementation: **High**

Priority: **High**

Purpose: Add complete control on limits to the user

Add limit on cardinality as part of manageable safeguards in the UI

13.5.2.5 UI Improvements: number of series

Complexity: Low

Impact: High

Probability of implementation: **Certain**

Priority: **High**

Purpose: Add information to the user ahead of time to improve experience

Return number of time series with tags cardinality

- Everytime a metric is selected or a query modified the UI will query this API
- UI displays the information
- If too many time series (or too many tags) the UI provide the information that query needs filtering to display all tags

13.5.2.6 UI Improvements: support number of tags in interactive tag selector

Complexity: TBD

Impact: High

Probability of implementation: **High**

Priority: Medium

Purpose: Improve user experience in case large number of tags needs to be handled interactively

- When more than 100 tags only show 100 first then show a “...” button
- And enforce the user to search to display matches
- Or use an infiniscroll with limited number of items visible at all times

13.5.2.7 Tags query improvement

Complexity: Low

Impact: High

Probability of implementation: **High**

Priority: **High**

Purpose: Keep user experience correct when there are too many items (response time, latency, performances of selectors)

- Improve tag query results: add an optional “warn” field (warning, or message, name TBD)
- Add a parameter to limit the number of tags (configurable limits in skyminer properties file), by default use 5,000
- In case of too many tags (limit reached) provide a warn message such as “Some tags [X,Y,X] have too many values please filter data to reduce cardinality and get all values”
- UI Shall be able to display this message

13.5.2.8 UI Improvements: data type

Complexity: Low

Impact: Low

Probability of implementation: **High**

Priority: Medium

Purpose: Keep user experience as smooth as possible

- Do tag query before asking for first & last sample
- Add data type with first last samples
- Add data type in the metrics filters
- If data type is used use data type filter

13.5.2.9 Grafana UI Improvements

Complexity: Medium/High

Impact: High

Probability of implementation: **High**

Priority: **High**

Purpose: Keep user experience as smooth as possible

- Make so that the Grafana query plugin works properly with high tag cardinality

13.5.2.10 Query add number of time series in response

Complexity: Low

Impact: Medium

Probability of implementation: Medium

Priority: Medium

Purpose: Give more awareness to user

- Provide user the information on how many time series were involved in the response (for awareness)
- Make this visible in the UI

13.5.2.11 UI add metrics/tags from response

Complexity: Medium

Impact: High

Probability of implementation: Medium

Priority: **High**

Purpose: Give more awareness to user

- Provide user the information for each metrics results on tags and cardinality of values for each tag (and allow to explore values)

13.5.2.12 User guidelines

Complexity: Medium

Impact: High

Probability of implementation: **High**

Priority: **High**

Purpose: Give information to user and system architects / engineers

- Anticipate use cases with high-cardinality series
- Document the use case
- Provide advise to tune the data model (split in metrics maybe)
- Provide advise on how to tune the system (separate physical SSD disks for Cassandra commotlog, data, kairosdb write cache, and kairosdb query cache)

13.5.2.13 Design alternate storage for high-cardinality series

Complexity: High

Impact: High

Probability of implementation: Low

Priority: Low

Purpose: Improve the system as a whole by supporting differently very special use cases

- Define what storage could be used for such series (e.g. OpenSearch)
- Do some design in preparation to this move
- Exercise the design

13.5.2.14 Inform user on server load

Complexity: Medium

Impact: High

Probability of implementation: Medium

Priority: Low

Purpose: Inform user when server is becoming slow (maybe because of accepting data, doing I/O, not enough memory...etc), using internal self-monitoring

- Add a service endpoint next to health (or as part of health monitor). E.g. /skyminer/health/performance-monitor
- this endpoint returns
 - Average query time on the target server for the past hour or so, on the cluster (if several servers), and status of CPU and memory, nb of queries...etc.
 - An indicator of the current status (e.g. OK, SLOW, DEGRADED, FAILED) derived from these metrics (low memory available, CPU usage too high, queries start to take too long... TBD)
- UI polls this API and informs the user about the metrics and if service is degraded

13.6 OpsCenter Installation

This documentation is about:

- KMP installation and some use
- How to show Compass data on KMP
- How to configure KMP to send KMP data to Skyminer

13.6.1 Install KMP

First of all you have to find a KMP version.

You can find KMP here:

- Official release : [kmp-generic-release-local](#)
- Development version : [kmp-generic-dev-local](#)

You can install KMP in standalone but you will need data to test your installation. Compass is the data generator we will install.

13.6.1.1 How to install Compass

Compass needs a PostgreSQL instance to work. So install PostgreSQL before installing Compass. The version of the DB depends on the version of the Compass you have.

1st install

Compass use a *setup.exe* file to install. Choose the default directory: *C:\Compass*

Use MariaDB and all the defaults.

An error occurred: a problem with MTS “Failed to start MTS”. This error is known by Compass Team but not fixed. The installer still announcing that we must restart the computer, don’t do it and stop installation. You need to install twice without uninstalling the previous one to have a working Compass.

2nd install

More options are available, just add everything except the documentation. MariaDB is now configured.

New problem “Cannot turn Compass into RUN mode”. Another error is detected: Microsoft Windows must be installed in English. M.Potier helps to configure Windows but still doesn’t work => The local account created on Compass installation is incomplete.

Moreover the account (NewpointCompass) seems not created at the good place: *C:\Users\LocalService\Application Data* instead of *C:\Windows\System32\config\systemprofile\AppData\Local*. M.Potier copy the files from his own installation into both directories.

With this new configuration, Compass services must be stopped:

```
net stop compass
net stop fepxml
net stop mysql
```

and restarted:

```
net start mysql
net start fepxml
net start compass
```

13.6.1.2 How to use Compass for demo

Start Compass and launch UI:

- Check that Compass is responding: `fepstate`
- Turn Compass into STOP mode: `fepstate 0`
- Turn Compass into RUN mode: `fepstate 2`
- Enter credentials: login=admin, password=newpoint
- Start Compass UI: `fepview` or *Compass client* in the installed programs
- Login with login=admin, password=newpoint
- Check that this simulator add data and raise error

Start Compass DEMO simulator:

- Start Compass : `fepstate 2`
- Start Compass UI : `fepview` or Compass client
- Credentials : admin and newpoint
- Go to overview -> System -> Driver Buses -> DEMO (dble click)
- Click on green button Start (top right corner) then OK

Now Compass DEMO generate data for you.

To stop drivers : `fepstate 1`. This will stop all drivers but not Compass = IDLE state.

13.6.1.3 How to install KMP

Follow the instructions of the KMP installation guide except that PostgreSQL password is kratos

Go to LDAP configuration paragraph of the Administration Guide and configure the authentication token

- Go to Compass directory: `C:\Compass\rest` (by default) and copy *jwt.keys*
- Go to KMP directory: `C:\Program Files (x86)\Kratos\KMP\bin` (by default) and paste *jwt.keys*

To restart the KMP:

- Restart the service (if installed as a service)
- `launch.bat` (in `C:\Program Files (x86)\Kratos\KMP\bin`) if you want to do it manually

KMP is available locally at this url [KMP URL](#)

Configure KMP for Compass

- Go to Settings -> Loaders
- **Add new Loaders**
 - Loader type = Compass
 - hostname = 127.0.0.1
 - Node Id is auto detected = “your pc name”

Check that data coming from Compass are displayed in KMP:

- Some simulated error and warning are displayed
- In inventory menu, the Inventory/Loaders/Sites are available

How to show Compass data on KMP:

- Go to [KMP URL](#)
- Credentials : admin - kratos
- Configure Compass loaders (see above)
- Create a new dashboard ('+' button at bottom)
- Add chart widget (line chart for example) with drag and drop
- **Go to Settings of this widget (top right corner) and add :**
 - Title
 - Measurement 1: asset = devN description (see Compass DEMO) and Metric = *fi1 then Confirm and Save
 - Save the dashboard (the disk on top of the page) with a significant name
 - This will unset the Edit mode and you will have a chart if Compass is running
 - Your dashboard is now available in the dashboard list

13.6.1.4 How to update KMP for Skyminer

Skyminer is embedded on the perf_microservice of the [kmp_microservices](#) and available in skyminer repo [kmp-microservices](#)

Clone the [kmp_microservices](#)

You have to work on the official repository and push local change on local repository before proposing to official release.

Skyminer artifacts

Skyminer artifact is not available in the KMP repository so the file is temporary available in the perf_microservice.

Build will use it (build.gradle) to compile the service.

The *SkyminerJavaClient.jar* must be deployed in the KMP/libs to work.

To see KMP data on Skyminer:

- Go to Metric
- Search for OPENSOURCE (generic prefix set by KMP) or fi1 (one of the data sent to KMP by Compass)

13.6.2 Miscellaneous

How to test a version of KMP without installing it

Example with version 1.3.0-enigma on installed version 1.2.0.

- Unzip KMP archive. You have 2 directory : bin and lib
- Rename the 2 same directories in your KMP installation (ex: bin.old and lib.old)
- Copy the new directories in the installation
- **Configure it (actions from old to new directories)**
 - **bin directory**
 - * Compass : Copy the CompassLoaderConfig dir and the jwt.keys file

- * Skyminer : Copy the rest_args.txt configured with new parameters for perf microservices
- * Configure the postgres.txt file (in local_args.d dir) with your postgres password
- * Create the empty file server.json (probably specific to this upgrade)
- **lib directory**
 - * Skyminer : Copy SkyminerJavaClient jar and replace the perf_microservice jar

13.7 KMP Message Queue Interface

This chapter describes how the integration between Skyminer-core (as metrics service) is done with KMP.

13.7.1 KMP AMQP Messaging patterns

13.7.1.1 Request/Reply

- RPC-like, where a single request is sent, and a single response is received.
- Support for progress and cancellation messages back to the requester
- Support for scalability by adding more processors that receive the requests
- Support for specifying where a request is serviced
 - Allows spatial-locality of processing of data
 - Sender indicates this before sending the message

13.7.1.2 Pub/Sub

- One-way messages are sent to interested consumers
- Use capabilities of RMQ to perform message routing to consumers
- Support for scalability by scaling out the number of consumers to process the throughput from the publishers

13.7.1.3 Discovery (Scatter/Gather)

- Messages are broadcasted to consumers, which can decide to respond
- Sender waits a certain amount of time to receive responses

13.7.2 Metrics Data ingest

Data ingest is done using a Publish/Subscribe model.

Metrics service (Skyminer-core) subscribes to a topic that provides metrics to ingest. The linear scalability is used in conjunction with the linear scalability of AMQP: as many ingest services as needed can be provisionned.

- Exchange Type: Topic
- Exchange Name: KMP.PubSub
- Routing Key: Indicates the source of the published data: `site.host.module.messageType.instance`

- Queue Name: Message Type [Service Name] (ex. `Metrics.Ingest [MetricsService]`)
- Wildcards can be used to
 - Substitute for exactly one word `*` (star)
 - Substitute for zero or more words `#` (hash)
- Queue name is very important to support load balanced consumers
- Scalability is achieved by creating more than 1 consumer bound to a queue: every Skyminer-core instance
 - Create N-Consumers
 - Specify `Channel.SetQos(1)` to indicate that a consumer shouldn't receive a new message until it has finished with the previous message.

Note:

- If it is possible the Skyminer service will use meta-data from the routing key to tag the metrics: site, host, module and instance will then be used to tag the data origin.
 - If the meta-data is not available to consumer it will have to be inserted in the message data itself.
 - If both are used, the value inserted in the message supersedes the meta-data from AMQP
-

The message may contain one or several datapoints for batch sending.

13.7.3 Metrics Data query

Data query is done using a Request/Reply model.

- Exchange Type: Direct
- Exchange Name: Message Type
- Routing Key is defined per the need from the publisher:
 - `site.host`: Specify which host should service the request.
 - `site`: Specify which site should service the request.
 - `instance` : Specify which instance should service the request.
 - `<empty>` : Sender doesn't care where the request is serviced.
- Queue Name: Message Type [Service Name] (ex. `Canonical.ClassifyRequest [ClassifyService]`)
- Scalability is achieved by creating more than 1 consumer bound to a queue.
 - Create N-Consumers
 - Specify `Channel.SetQos(1)` to indicate that a consumer shouldn't receive a new message until it has finished with the previous message.

13.7.4 Scalability

- Skyminer REST API can handle about 100K datapoints/second.
- AMQP needs to be tested for performances (per node)
- If the throughput is very high, probably the issue will be RabbitMQ brokers that become bottlenecks.

A solution might be to use Kafka queues for high-performances metrics ingest.

13.8 Performance History Service Integration

The Performance History Service is responsible for storing performance data, historical performance data, metric definitions and measurand definitions. We want to save this information in Skyminer.

13.8.1 How to use

In the installation folder of KMP, edit the file `res_args.txt`. Change the line `-i com.kratos.perfhandler.PerfHistHandler`, remove arguments and add `skyminerEnabled=true skyminerUrl=http://ip.of.skyminer/ skyminerDataDirectory="C:\my\folder"`

Example:

```
-i com.kratos.perfhandler.PerfHistHandler skyminerEnabled=true skyminerUrl=http://192.168.48.29/ skyminerDataDirectory="C:\Program Files (x86)\Kratos\KMP\SkyminerData"
```

Optional arguments:

`skyminerSleepDelayBetweenQueriesMs` (default: 5000)

`skyminerMaxCacheFileAgeMs` (default: 86400000)

13.8.2 Measurand

A Measurement is a single sample of a Measurand (Metric on a source Asset). It includes an mid, a timestamp, sampleId, and a value. The value may be a string, number, or object.

13.8.2.1 Datamodel

id: amplifier.power/KMP/Asset/5678

mid: amplifier.power/KMP/Asset/5678

value: 64.46

nvalue: 64.46

time: <sampletime in microseconds>

13.8.2.2 Mapping with Skyminer

name: OPENSOURCE_ + mid
timestamp: time / 1000
value: value
tags:
 host: KMP
 id: id

13.8.3 Java Interface

```
// !! NOT USED
// ??
public void Activate() throws Exception {}

// Returns all measurements stored in the file as a List of Maps <- Useless ?
public List<MeasurementObj> getAllMeasurementsList() throws Exception {}

// !! NOT USED
// In Skyminer you can't really get a measurement by ID
public MeasurementObj getMeasurementItemById(String id) throws Exception {}

// !! NOT USED
// Return all measurements that were recorded after startTime and before endTime as a
↳List of Maps
public List<Map> getMeasurementsByTime(Long startTime, Long endTime) throws Exception {}

// !! NOT USED
public <T> List<T> GetList(String tname, CallInfo call) throws Exception {}

// !! NOT USED
public <T> T GetInstance(String tname, String id, CallInfo call) throws Exception {}

// !! NOT USED
// I think it's for update
public ApiResult FinishPut(IObjApi api, String id, Object obj, CallInfo call) throws
↳Exception {}

// Used but it can only return non-complex object for the moment
public <T> List<T> Get(IObjApi api, CallInfo call) throws Exception {}

// !! NOT USED
public <T> T Get(IObjApi api, String id, CallInfo call) throws Exception {}

// Used for pushing data in Skyminer
public ApiResult FinishPost(IObjApi api, String id, Object obj, CallInfo call) throws
↳Exception {}

// !! NOT USED
public ApiResult FinishDelete(IObjApi api, String id, CallInfo call) throws Exception {}
```

13.8.3.1 MeasurementObj Object

A MeasurementObj is a “measurand”

```
public MeasurementObj (MeasurementObj x) throws Exception
{
    //The property is a primitive.
    this.id = x.id;

    //The property is a primitive.
    this.mid = x.mid;

    //The property is a primitive.
    this.value = x.value;

    //The property is a primitive.
    this.nvalue = x.nvalue;

    //The property is a primitive.
    this.time = x.time;

    //The property is a primitive.
    this.sampleId = x.sampleId;

    //The property is not a primitive.
    //The property is not a container.
    //The property is an object.
    this.objectValue = x.objectValue;
}
```

13.8.3.2 CallInfo Object

The CallInfo Object contains information about the request.

Example:

```
String mid = "";
Long time = 0L;
for (Filter f : call.getFilters()) {
    if (f.getProperty().equals("mid")) {
        mid = f.getValue();
    }
    if (f.getProperty().equals("time")) {
        time = Long.parseLong(f.getValue())/1000;
    }
}
```

13.8.3.3 FinishPost Method

FinishPost is used to push data in Skyminer.

If you want to get the information about the point you will push in Skyminer, you will have to cast the Object obj (third parameter of the function) to a MeasurementObj.

Example:

```
public ApiResult FinishPost(IObjApi api, String id, Object obj, CallInfo call) throws Exception {
    MeasurementObj mObj = (MeasurementObj) obj;
    String jsonFormat = "{\n" +
        "    \"name\": \"OPENSOURCE_\"+ mObj.getMid() + \"\", \"n\" +
        "    \"timestamp\": \"+ (mObj.getTime()/1000) + \"\", \"n\" +
        "    \"value\": \" + mObj.getNvalue() + \"\", \"n\" +
        "    \"tags\": {\n" +
        "        \"host\": \"KMP\", \"n\" +
        "        \"id\": \"\" + mObj.getId() + \"\" \"n\" +
        "    } \n" +
        " }";
    postToSkyminer(jsonFormat);
    return new ApiResult(api, obj, id);
}
```

13.9 Skyminer-KMP Integration

The integration of Skyminer with KMP provides key data storage & analytics and third-party integrations to KMP, and provide a better sustainability for Skyminer platform.

13.9.1 Metrics data storage

Skyminer provides the data storage service for metrics in KMP.

The interface is done by subscribing to KMP message bus via new micro-services (i.e. **skyminer-kmp** microservices), or alternatively in a module activated in skyminer-core.

The module is easier to deliver.

13.9.2 Metrics analytics service

Skyminer provides the data query & analytics service for metrics in KMP.

The interface is done by subscribing to KMP message bus via new micro-services (i.e. **skyminer-kmp** microservices), or alternatively in a module activated in skyminer-core.

The module is easier to deliver.

13.9.3 Third-party integrations

One of biggest strengths of Skyminer is the third-party integrations (e.g. Grafana, Jupyter, BIRT...). These integrations make no sense in KMP right away, but are valuable as an optional add-on for customers that need to access data and run their own processings.

13.9.4 Data model mapping

The regular time series data model should work fine for KMP.

The tag key/value pairs will indicate as many information elements as available such as:

- domain
- host
- site
- satellite/antenna/transponder/service/device
- asset_uuid
- asset_name
- alarm_status
- status

...etc.

13.9.5 Steps for integration

13.9.5.1 Skyminer REST services for KMP

First step might be to use Skyminer REST API that are already available and run Skyminer container as part of KMP.

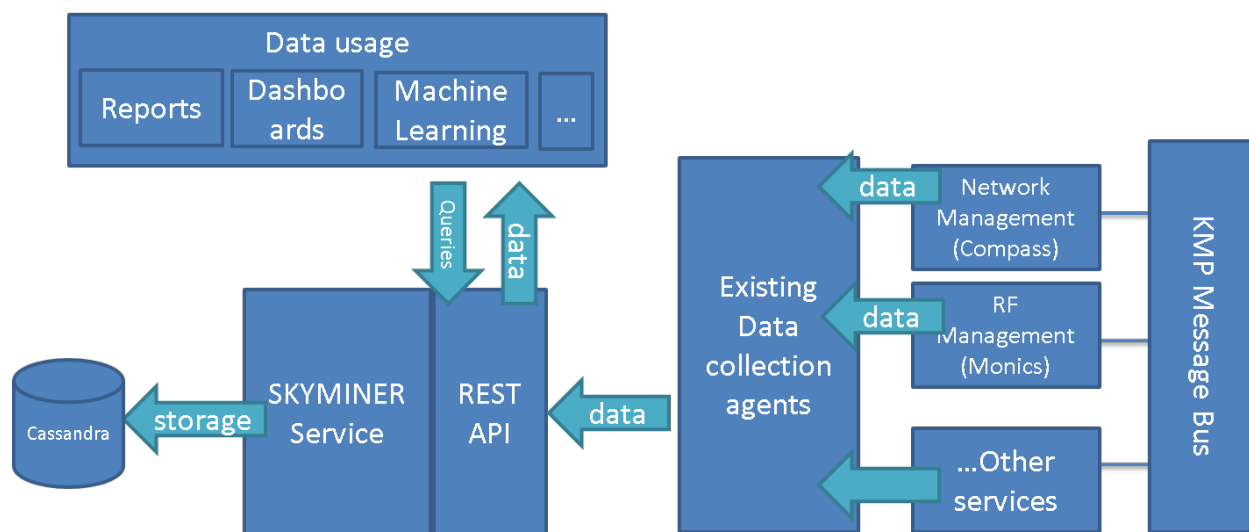


Fig. 1: Skyminer integrated as an additional container providing REST services to KMP.

13.9.5.2 Interfacing KMP Message Queue

Then a second step to include Skyminer plugged into the KMP message bus for data ingest and serving queries

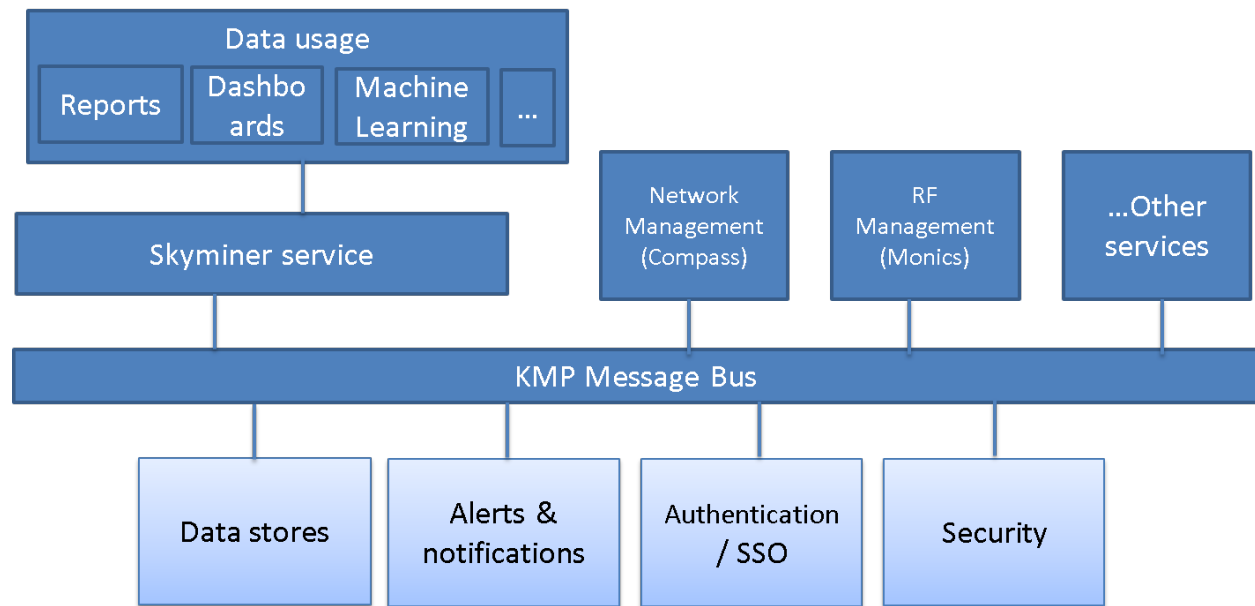


Fig. 2: Skyminer integrated as an additional container interoperating with KMP via AMQP Message Bus.

13.9.5.3 Mixed approach for third-party and legacy

If the whole Skyminer package is used or for third-party data integration a mixed approach is always possible.

13.9.6 Authentication & ACL

Authentication & ACL are managed by KMP.

If third-party integration is provided, an authentication filter using KMP rules on Skyminer data will be developed.

13.9.7 Packaging

13.9.7.1 KMP Data & Analytics Service

The KMP services are packaged without the user knowledge of Skyminer in the tools. KMP provides domain-specific and cross-domain capabilities and uses Skyminer as data and analytics service in the background.

The user never accesses Skyminer data but uses KMP features that generate data stored in the service and display data from the service.

The software baseline is the skyminer-core with the data storage backend without any of the other containers (frontend, proxy, grafana, jupyter... etc.).

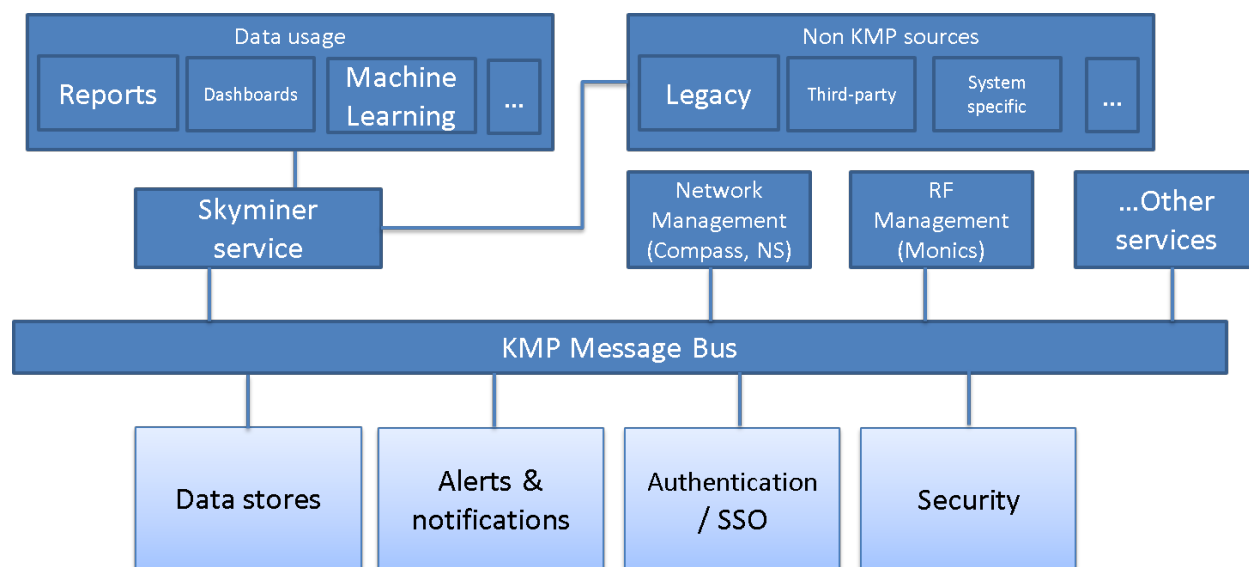


Fig. 3: Skyminer integrated as an additional container interoperating with KMP via AMQP Message Bus and third-party using REST.

13.9.7.2 Data analytics & Third-party integrations

The Third-party integration bundle is provided as an option (this is the actual Skyminer platform).

All containers are delivered and operate together with the KMP data & analytics service (former skyminer-core).

Skyminer UI and correlation analysis is part of this software baseline.

13.9.8 Time Series meta-data

Time Series meta-data is included either in Message meta-data or in message data itself. Same meta-data information can be provided at different levels, the lowest level is then taking precedence.

13.10 Log collectors comparison articles

[Fluentd vs Logstash](#)

[Filebeat vs Logstash](#)

[Ultimate log collectors benchmarking 2020](#)

13.11 Filebeat

13.11.1 Introduction

The following sections shows how you can use Filebeat with Docker or Kubernetes.

Note: According to https://www.elastic.co/guide/en/beats/libbeat/current/breaking-changes-7.13.html#_beats_may_not_be_sending_data_to_some_distributions_of_elasticsearch filebeats is not operating properly with openSearch

13.11.2 Pros and cons

13.11.2.1 Pros

- Easy to manage thanks to the integrated auto-discover
- Non-invasive (no modification needed for Docker and Kubernetes inside your containers)
- Can be secured (SSL or TLS)
- Can be connected to either Fluentd or Logstash
- Can collect logs from lots of inputs
- `docker logs` command still work with your containers
- Automatically generates all the metadata needed to watch your logs with Kibana
- Can create the dashboards in Kibana automatically
- Can stop the log collection without changing `docker-compose` configuration

13.11.2.2 Cons

- Needs more resources than some competitors (Fluentd, ...)
- Not able to advance parse your logs (needs Fluentd or Logstash to do so)
- Not many outputs (can be extended thanks to Fluentd or Logstash)

13.11.3 Docker

13.11.3.1 Prepare your system

On the host machine:

```
cd <installation_directory>
sudo mkdir filebeat
sudo touch filebeat/filebeat.docker.yml
sudo touch filebeat/ilm.json
```

Edit `filebeat.docker.yml`:


```
filebeat.config:
  modules:
    path: ${path.config}/modules.d/*.yaml
    reload.enabled: false

filebeat.autodiscover:
  providers:
    - type: docker
      hints.enabled: true

processors:
- add_cloud_metadata: ~

output.opensearch:
  hosts: '${OPENSEARCH_HOSTS:opensearch:9200}'
  username: '${OPENSEARCH_USERNAME:}'
  password: '${OPENSEARCH_PASSWORD:}'

setup.ilm.policy_file: "/usr/share/filebeat/ilm.json"
setup.ilm.check_exists: false
setup.ilm.overwrite: true
```

Edit ilm.json:

```
{ "policy":
  {
    "phases":
    {
      "hot":
      {
        "actions":
        {
          "rollover":
          {
            "max_size": "25GB"
          }
        }
      },
      "delete":
      {
        "min_age": "30d",
        "actions":
        {
          "delete": {}
        }
      }
    }
  }
}
```

13.11.3.2 Docker compose

To enable the log collection on the machine, add the following in `docker-compose.yml`:

```
skyminer-kibana:
  image: 192.168.48.22:8083/skyminer/skyminer-kibana:1.0
  container_name: skyminer-kibana
  restart: always
  ports:
    - "5601:5601"
  environment:
    ELASTICSEARCH_HOSTS: '["http://elasticsearch:9200"]'

skyminer-filebeat:
  image: docker.elastic.co/beats/filebeat:7.8.0
  container_name: skyminer-filebeat
  restart: always
  privileged: true
  user: root
  logging:
    driver: "json-file"
    options:
      max-size: "50m"
      max-file: "10"
  volumes:
    - "<installation_directory>/skyminer/filebeat/filebeat.docker.yml:/usr/share/
↪filebeat/filebeat.yml:z"
    - "<installation_directory>/skyminer/filebeat/ilm.json:/usr/share/filebeat/ilm.
↪json:z"
    - "/var/lib/docker/containers:/var/lib/docker/containers:ro"
    - "/var/run/docker.sock:/var/run/docker.sock:ro"
  environment:
    - BEAT_STRICT_PERMS=false
    - ES_HOST=<host_ip>
    - ES_PORT=9200
    - KIBANA_HOST=<host_ip>
    - KIBANA_PORT=5601
```

Then made it up (even if other containers are running):

```
docker-compose up -d
```

13.11.4 Kubernetes

Everything is easy, well-defined and clear to run Filebeat on Kubernetes.

See [Run Filebeat on Kubernetes](#).

13.11.5 Elastic Stack integration

13.11.5.1 Logs TTL aka Index Lifecycle Management (ILM)

Filebeat creates automatically the index template (`filebeat-<version>-*`) with a specific ILM (`filebeat`)... If you provide the Kibana URL in the Filebeat configuration.

It only creates the Hot phase by default, with the following configuration values:

- Maximum index size: 50 GB
- Maximum age: 30 days

Subsequently, the roll-up occurs when the indice reaches 50 GB or when its age is 30 days.

A new indice is created on each rollover (named `filebeat-<version>-<date>-<roll_id>`)

In our case, we defined the ILM policy we wanted with the file `ilm.json`, so the resulting ILM is different from the default one.

13.11.5.2 Configure the ILM

The ILM can be configured in the Filebeat configuration file.

See [Configure ILM](#).

Please note the ILM configuration json file shall be mounted to the Filebeat container.

The `setup.ilm.policy_file` field of the Filebeat configuration file must be the Filebeat inside-container absolute file path.

Alternatively, the Elasticsearch API can be also used to define the ILM policy.

13.11.6 References

[Inputs](#)

[Outputs](#)

[Run Filebeat on Docker](#)

[Run Filebeat on Kubernetes](#)

[Configure Filebeat](#)

[ILM \(get started\)](#)

[Configure ILM](#)

13.12 Fluentd

13.12.1 Introduction

The following sections show how you can use Fluentd with Docker or Kubernetes.

13.12.2 Pros and cons

13.12.2.1 Pros

- Can be extended with lots of plugins
- Can be secured (SSL or TLS)
- Can be connected to Logstash
- Can receive logs from either Fluentbit or Filebeat
- Can collect logs from Docker using the provided logging driver and stream them to the Fluentd aggregator
- Proposes an API to post logs (TCP and UDP)
- Proposes a logging library that can be used by our applications (like Log4j)
- Can use an agent to tail the log files and stream them to the Fluentd aggregator
- Can parse your logs and extract information from them (Logstash is not mandatory for that)

13.12.2.2 Cons

- `docker logs` command does not work with your containers when using the logging driver
- Quite invasive as it needs agents installed in the containers or the docker logging driver
- With compose, forces the EFK (Elastic, Fluentd, Kibana) to be started and ready before running your application containers
- Cannot stop log collection without changing `docker-compose` configuration

13.12.3 Docker

13.12.3.1 Prepare your system

On the host machine:

```
cd <installation_directory>
sudo mkdir fluentd
sudo touch fluentd/fluent.conf
sudo touch fluentd/fluentd-template.json
```

Edit `fluent.conf`:

```

<source>
  @type forward
  @id input1
  @label @mainstream
  port 24224
</source>

<filter **>
  @type stdout
</filter>

<label @mainstream>
  <match docker.**>
    @type file
    @id output_docker1
    path /fluentd/log/docker.*.log
    symlink_path /fluentd/log/docker.log
    append true
    time_slice_format %Y%m%d
    time_slice_wait 1m
    time_format %Y%m%dT%H%M%S%z
  </match>
  <match skyminer.**>
    @type elasticsearch
    host elasticsearch
    port 9200
    logstash_format true
    logstash_prefix fluentd
    include_tag_key true
    type_name access_log
    tag_key @log_name
    enable_ilm true
    ilm_policy_id fluentd-policy
    ilm_policy { "policy": { "phases": { "hot": { "actions": { "rollover": { "max_size":
→ "25GB" } } }, "delete": { "min_age": "30d", "actions": { "delete": {} } } } } } }
    template_name fluentd-template
    template_file /fluentd/fluentd-template.json
  </match>
</label>

```

Edit fluentd-template.json:

```

{
  "index_patterns": ["fluentd-*"],
  "settings": {
    "number_of_shards": 1,
    "number_of_replicas": 1,
    "index.lifecycle.name": "fluentd-policy"
  }
}

```

13.12.3.2 Docker compose

To enable the log collection on the machine, modify as following:

- docker-compose.yml:

```
version: '2.2'

services:
  cassandra:
    image: cassandra-skyminer:2020-dev-S17
    container_name: skyminer-cassandra
    restart: always
    ports:
      - "7000:7000"
      - "9042:9042"
      - "7199:7199"
    environment:
      - "CASSANDRA_CLUSTER_NAME=SKYMINER"
      - "CASSANDRA_SEEDS=<host_ip>"
      - "CASSANDRA_BROADCAST_ADDRESS=<host_ip>"
      - "LOCAL_JMX=no"
    volumes:
      - "<installation_directory>/skyminer/cassandra/data:/var/lib/cassandra/data:z"
      - "<installation_directory>/skyminer/cassandra/commitlog:/var/lib/cassandra/
↵commitlog:z"
      - "<installation_directory>/skyminer/cassandra_jmx/jmxremote.password:/etc/
↵cassandra/jmxremote.password:z"
    logging:
      driver: "fluentd"
      options:
        fluentd-address: "<host_ip>:24224"
        tag: skyminer.logs.cassandra

  skyminer:
    image: skyminer:2020-dev-S17
    container_name: skyminer-service
    restart: always
    ports:
      - "2003:2003"
      - "2004:2004"
      - "4242:4242"
    expose:
      - "8080/tcp"
    environment:
      - "CASSANDRA_HOST=cassandra:9042"
      - "VIRTUAL_PORT=8080"
      - "VIRTUAL_HOST=skyminer-service"
    volumes:
      - "<installation_directory>/skyminer/conf:/opt/skyminer/conf:z"
      - "<installation_directory>/skyminer/queue:/opt/skyminer/queue:z"
      - "<installation_directory>/skyminer/license:/opt/skyminer/license:z"
    links:
      - "cassandra:cassandra"
```

(continues on next page)

(continued from previous page)

```

logging:
  driver: "fluentd"
  options:
    fluentd-address: "<host_ip>:24224"
    tag: skyminer.logs.service

grafana:
  image: grafana-skyminer:2020-dev-S17
  container_name: skyminer-grafana
  restart: always
  environment:
    - "VIRTUAL_HOST=skyminer-grafana"
    - "VIRTUAL_PORT=3000"
    - "GF_SECURITY_ADMIN_USER=admin"
    - "GF_SECURITY_ADMIN_PASSWORD=5t0bgYXC1PdLZRC8RtzorfsZXPsvtK4TVKfIkmHnUSNE2eZ44y"
    - "GF_DATABASE_HOST=<host_ip>:5432"
    - "GF_DATABASE_USER=grafana"
    - "GF_DATABASE_PASSWORD=74W0shNwonDjbZgs2avs501UYXzeqbAW30NZ0je2HG47v3NIgG"
    - "GF_ALERTING_ENABLED=true"
    - "GF_EXPLORE_ENABLED=false"
    - "GF_DATABASE_TYPE=postgres"
    - "GF_AUTH_ANONYMOUS_ENABLED=true"
    - "GF_AUTH_ANONYMOUS_ORG_NAME=Main Org."
    - "GF_AUTH_ANONYMOUS_ORG_ROLE=Editor"
    - "GF_SERVER_ROOT_URL=http://<host_ip>/grafana"
    - "GF_PLUGINS_ALLOW_LOADING_UNSIGNED_PLUGINS=grafana-skyminer-datasource"
  expose:
    - "3000/tcp"
  volumes:
    - "<installation_directory>/skyminer/grafana-plugins:/var/lib/grafana/plugins:z"
    - "<installation_directory>/skyminer/grafana-dashboards/all.yml:/etc/grafana/
    ↪provisioning/dashboards/all.yml:z"
    - "<installation_directory>/skyminer/grafana-dashboards/dashboards:/var/lib/
    ↪grafana/dashboards:z"
    - "<installation_directory>/skyminer/grafana-dashboards/home.json:/usr/share/
    ↪grafana/public/dashboards/home.json:z"
  links:
    - "skyminer:skyminer"
  logging:
    driver: "fluentd"
    options:
      fluentd-address: "<host_ip>:24224"
      tag: skyminer.logs.grafana

nginx-proxy:
  image: nginx-skyminer:2020-dev-S17
  container_name: skyminer-nginx
  restart: always
  ports:
    - "80:80"
    - "443:443"
  volumes:

```

(continues on next page)

(continued from previous page)

```

- "/var/run/docker.sock:/tmp/docker.sock:ro"
- "<installation_directory>/skyminer/nginx_volumes/conf:/etc/nginx/conf.d:z"
- "<installation_directory>/skyminer/nginx_volumes/certs:/etc/nginx/certs:ro,z"
- "<installation_directory>/skyminer/nginx_volumes/dhparam:/etc/nginx/dhparam:ro,
↪z"
  logging:
    driver: "fluentd"
    options:
      fluentd-address: "<host_ip>:24224"
      tag: skyminer.logs.nginx

skyminer-collectd:
  image: skyminer-collectd:2020-dev-S17
  container_name: skyminer-collectd
  restart: always
  volumes:
    - "<installation_directory>/skyminer/collectd/collectd.conf:/etc/collectd/
↪collectd.conf:z"
  logging:
    driver: "fluentd"
    options:
      fluentd-address: "<host_ip>:24224"
      tag: skyminer.logs.collectd
  depends_on:
    - cassandra
    - skyminer
  environment:
    - SKYMINER_LOCAL_HOSTNAME="localhost.localdomain"
    - SKYMINER_LOCAL_IP="<host_ip>"

skyminer-jupyter:
  image: skyminer-jupyter:2020-dev-S17
  container_name: skyminer-jupyter
  restart: always
  ports:
    - "8787:8787"
  volumes:
    - "<installation_directory>/skyminer/jupyter/notebooks:/jupyter:z"
    - "<installation_directory>/skyminer/jupyter/jupyter_notebook_config.py:/root/.
↪jupyter/jupyter_notebook_config.py:z"
    - "<installation_directory>/skyminer/jupyter/SkyminerTS:/usr/lib/python3.6/
↪SkyminerTS:z"
    - "<installation_directory>/skyminer/jupyter/SkyminerTSPlot:/usr/lib/python3.6/
↪SkyminerTSPlot:z"
  cpus: 0.8
  logging:
    driver: "fluentd"
    options:
      fluentd-address: "<host_ip>:24224"
      tag: skyminer.logs.jupyter

skyminer-front:

```

(continues on next page)

(continued from previous page)

```

image: skyminer-front:2020-dev-S17
container_name: skyminer-front
restart: always
ports:
  - "81:80"
logging:
  driver: "fluentd"
  options:
    fluentd-address: "<host_ip>:24224"
    tag: skyminer.logs.front
volumes:
  - "<installation_directory>/skyminer/customization/customer_logo:/var/www/img/
↪customer_logo:z"
  - "<installation_directory>/skyminer/customization/customer_logo:/var/www/query/
↪assets/images/customer_logo:z"
  - "<installation_directory>/skyminer/front/config:/var/www/query/assets/config:z"

```

- docker-compose-monitoring.yml:

```

version: '2.2'

services:
  opensearch:
    restart: always
    image: opensearch-skyminer:2020-dev-S17
    container_name: skyminer-opensearch
    environment:
      - node.name=skyminer-opensearch
      - cluster.initial_master_nodes=skyminer-opensearch
      - cluster.name=kairos-elastic
      - transport.tcp.port=9300
      - bootstrap.memory_lock=true
      - http.cors.enabled=true
      - http.cors.allow-origin=*
      - http.cors.allow-headers=Content-Type, Access-Control-Allow-Headers,
↪Authorization, X-Requested-With
      - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
    ulimits:
      memlock:
        soft: -1
        hard: -1
    volumes:
      - ./opensearch/esdata:/usr/share/opensearch/data:z
    ports:
      - 9200:9200
      - 9300:9300

  skyminer-kibana:
    image: 192.168.48.22:8083/skyminer/skyminer-kibana:1.0
    container_name: skyminer-kibana
    restart: always
    ports:

```

(continues on next page)

(continued from previous page)

```

    - "5601:5601"
  environment:
    ELASTICSEARCH_HOSTS: '["http://opensearch:9200"]'
  depends_on:
    - skyminer-fluentd
#   volumes:
#     - "<installation_directory>/skyminer/kibana/kibana.yml:/usr/share/kibana/config/
↪kibana.yml:z"

skyminer-fluentd:
  image: 192.168.48.22:8083/skyminer/skyminer-fluentd-aggregator:1.0
  container_name: skyminer-fluentd
  restart: always
  ports:
    - "24224:24224"
    - "24224:24224/udp"
  volumes:
    - "<installation_directory>/skyminer/fluentd/fluent.conf:/fluentd/etc/fluent.
↪conf:z"
    - "<installation_directory>/skyminer/fluentd/fluentd-template.json:/fluentd/
↪fluentd-template.json:z"
  depends_on:
    - opensearch

```

Then made it up:

```

script docker-compose -f docker-compose-monitoring.yml up -d && sleep 60 && docker-
↪compose up -d

```

13.12.4 Kubernetes

Everything is easy, well-defined and clear to run Fluentd on Kubernetes.

See [Kubernetes logging](#).

13.12.5 Standalone Fluentd aka td-agent

Here we define how you can integrate any machine or container using the `td-agent` without using the Docker logging driver.

We use the `tail` input of Fluentd to tail the log file. It detects automatically the file log rotation.

In our case, we will define a dummy service in a container named `ubuntu-fluentd-agent-test` to simulate a service which will log through a rotating log file.

The `skyminer-fluentd` container is required and must be fully functional when starting the dummy service.

13.12.5.1 Run the dummy service container

On the host machine, run:

```
docker run -it --name="ubuntu-fluentd-agent-test" --network="skyminer_default" ubuntu_
↪bash -c "apt-get update && apt-get install python ruby-dev g++ make vim sudo -y && gem_
↪install fluentd --no-doc && mkdir fluent && bash"
```

13.12.5.2 Configure td-agent

When the container is started and you have its prompt command interactable, you can edit `fluent/fluent.conf`:

```
<source>
  @type tail
  path /test.log
  pos_file /var/log/td-agent/test.log.pos
  tag skyminer.test.log
  <parse>
    @type json
  </parse>
</source>

<match skyminer.test.log>
  @type forward
  send_timeout 10s
  recover_wait 10s
  hard_timeout 10s

  <server>
    name myserver1
    host skyminer-fluentd
    port 24224
  </server>
</match>
```

13.12.5.3 Code the dummy service

After configuring the Fluentd agent (td-agent), edit `test-log.py`:

```
import time, os

log_index = 1
file = open("/test.log", "w")
indice = 0
while True:
    if indice != 0 and indice % 100 == 0:
        file.close()
        os.rename("/test.log", "/test." + str(log_index) + ".log")
        log_index = log_index + 1
        file = open("/test.log", "w")
```

(continues on next page)

(continued from previous page)

```
file.write('{"time": "' + str(int(time.time())) + '", "container_name": "ubuntu-  
↪fluentd-agent-test", "log": "rotation #' + str(log_index) + ', indice #' + str(indice)↪  
↪+ '" }\n')  
indice = indice + 1  
time.sleep(1)
```

13.12.5.4 Run the dummy service

Now you can start the Python script and td-agent to run the dummy service:

```
script python test-log.py &  
fluentd -c ./fluent/fluent.conf
```

13.12.6 Elastic Stack integration

13.12.6.1 Logs TTL aka Index Lifecycle Management (ILM)

Fluentd does not create any index template or specific ILM by default.

Nonetheless, Fluentd creates an indice per day named `<logstash_prefix>-<date>`.

Subsequently, the user shall create his own index template and the ILM policy to use with this template.

The ILM can be configured also using the [fluent-plugin-elasticsearch ILM parameters](#).

In our case, we defined the index template and the associated ILM, resulting to an indice name as `<logstash_prefix>-<application>-<date>-<roll_id>`.

13.12.6.2 Configure the ILM

The user shall use the `Stack Management` section of Kibana to configure the ILM.

Alternatively, the Elasticsearch API can be also used to define the ILM policy or [fluent-plugin-elasticsearch ILM parameters](#).

See above, we defined our own template and policy to associate to.

13.12.7 Resources

[Docker logging with compose](#)

[Kubernetes logging](#)

[Parse logs](#)

[Format logs](#)

[Configure Fluentd](#)

[ILM \(get started\)](#)

[fluent-plugin-elasticsearch ILM parameters](#)

13.13 Skyminer SQL

13.13.1 Introduction

Skyminer SQL is a service that allows users to request Skyminer metrics through SQL requests.

13.13.2 Technical solution

Skyminer SQL is based on the following:

- A new PostgreSQL container is available to host this service : [skyminer-postgresql](#)
- A PostgreSQL Foreign Data Wrapper called [Multicorn](#)
- A Skyminer Python Foreign Data Wrapper, **skyminerfdw.py**, to request Skyminer API for data: Fixed, embedded in docker images [python](#)
- A service that initiates and maintains the FDW tables up-to-date

More informations about Foreign Data Wrapper for PostgreSQL: [FDW](#)

13.13.3 Multicorn

Multicorn documentation is available here: [readthedocs](#)

Multicorn source code is available here: [github](#)

13.13.3.1 Multicorn repository

Multicorn is available on [PGXN](#) in zip format and deployed in Kratos Nexus with the following command:

```
curl -v -u $REGISTRY_USER_DEV:$REGISTRY_PASSWORD_DEV --upload-file multicorn-$VERSION.  
tar.gz http://192.168.48.22:8082/repository/skyminer/multicorn/$VERSION/
```

13.13.3.2 Foreign Data Wrapper

Multicorn is available with some data wrappers but none is available for KairosDB.

To create a new data wrapper, see [implements](#) or [conference](#)

13.13.4 Service availability

Skyminer SQL has no data availability constraint so:

- It is installed in all the Skyminer installations
- It uses the **15432** port instead of the default one which is used by grafana_postgres service
- Credentials are auto generated and displayed at the end of installation
- It has its own database **skyminer-sql**, locally hosted

13.13.5 PostGreSQL FDW Identifiers

Skyminer data is case-sensitive. PostGreSQL must be configured to use case-sensitive collation. All table names, column names are managed with quoted identifiers to avoid confusion with keywords, and case problems.

Identifiers must follow rules: character set, start character to be alphabetic, limited to 31 characters. Changing this would require special compilation of postgres and is impractical.

A mechanism to ensure compliance to Postgres identifiers rules and uniqueness is used for metric names and tag names using aliases.

Aliasing with uniqueness is achieved as follows:

- If name does not start with an alphabetic character add an underscore character _ in front
- If name contains invalid characters: sanitize by removing them
- If sanitized name length is ≤ 31 characters and is equal to the original name, the name as-is used as alias, otherwise unique alias is computed as follows
- Compute a hash/checksum of the original name -not the sanitized name- (using murmur3 32-bit algorithm)
- Truncate sanitized name to 22 characters, concatenated with an underscore _ character and the results of the checksum encoded in base64

13.13.6 Update/maintenance of SQL mapping

The FDW is maintained by a service, which also provides a REST API.

13.13.6.1 Rebuild FDW REST API

A REST service allows to maintain the FDW mappings

- GET: /skyminer/sql-fdw/rebuild : rebuild all the FDW mappings
- GET: /skyminer/sql-fdw/rebuild/<metric> : rebuild FDW mappings for a particular metric

13.13.6.2 Automated Update

The FDW tables shall be automatically maintained to represent an accurate view of Skyminer tables.

This is done by a watchdog service, at regular intervals this service is used to update the tables. The frequency shall be configurable (by default set to 10 minutes for all metrics and 1 minute for “one metric” refresh).

- Each 10 minutes: Query metrics, proceed to creation of FDW tables for all missing metrics
- Each 1 minute: The metric that was updated the least recently is refreshed/updated
- Each query for a metric: Check last update of FDW tables. If last update is expired Query all metrics/tags, proceed to update of FDW tables for missing tags (by default 1 hour expiry)

13.13.6.3 New metric

- Create a transaction
- Compute alias for metric
- Get metric tags
- Compute alias for metric tags
- Create the FDW table for metric and its tags
- Insert the information into the meta data tables
- Commit transaction

13.13.6.4 New metric tag

- Create a transaction
- Compute alias for metric tag
- ALTER the FDW table for adding the metric tag
- Insert the information into the meta data table
- Commit transaction

13.13.7 Meta data tables (real tables)

13.13.7.1 skyminer_sql_metric_info

- **metric_name** character varying, PK
- **table_name** character varying (alias of the metric name for FDW)
- **last_update** timestampz

13.13.7.2 skyminer_sql_metric_tags_info

- **metric_name** character varying, PK
- **tag_name** character varying, PK
- **tag_value** character varying, PK
- **column_name** character varying, Unique (alias of the tag name for FDW)

13.13.8 Data tables (virtual tables using FDW)

One FDW table is created for each metric. One FDW column is created for each possible tag key of the metric.

The data FDW tables need to be re-created each time the tags set change.

Only one FDW class should be required, the name of the metric (also the name of the table)

13.13.8.1 <metric>

- **v_timestamp** timestamptz
- **v_numeric** numeric
- **v_json** character varying
- **<tag>** character varying - one column for each tag

13.13.9 Queries

13.13.9.1 FDW Tables

Queries: Mandatory filter on `value_timestamp` field

13.13.9.2 Authentication & ACL

Authentication: need to use LDAP or SSPI auth - can we retrieve user identity in FDW Multicorn interface? Additional option may include user token in the query (e.g `SELECT * from "my.skyminer.metric.name" WHERE v_timestamp>"some time" AND auth_token='ABDHGDF'`)? If the identity can be retrieved it can be used by FDW to provide identity to ACL gateway.

Management of ACL for meta-data is complicated. A way is to use Skyminer API and use the tables only for internal use.

HOW TO RUN GRAFANA USING DATASOURCE PLUGIN LOCALLY

14.1 Prerequisite

You need to have a built datasource plugin under <path_to_data_source_plugin>.

14.2 Running Grafana

- Run grafana container with the Skyminer Data Source plugin:

```
docker run -d -p 3000:3000 -e "GF_PLUGINS_ALLOW_LOADING_UNSIGNED_PLUGINS=grafana-  
↪skyminer-datasource" -v <path_to_data_source_plugin>:/var/lib/grafana/plugins/  
↪skyminer --name=grafana grafana/grafana:7.4.3
```

- Login grafana on localhost:3000 (admin/admin)
- Configure datasource plugin to get data from Skyminer VM of your choice
 - Configuration -> Data Sources -> Add Data source -> Select Skyminer
 - Enter the URL of the VM of your choice
 - Check Skip TLS Verify
 - Save & test

STATE OF THE ART, ANOMALIES/NOVELTIES DETECTION IN TIME SERIES

15.1 statistical approaches, machine learning approaches, deep learning approaches

Overview

- *State of the art, anomalies/novelities detection in time series*

In this document, I present a state of the art for the anomalies/novelities detection in time series. Many data are treated in the form of time series and many industrial problems have to detect anomalies in them. In our case, we want to detect novelties or anomalies in time series from satellite data. This can allow to prevent failure, for example. We have 2 main ways to detect anomalies on time series :

- The first way is to forecast time series with a training set and predict your new data with it. The anomalies appear when you look at the differences between prediction and real values.
- The second way is to use distance or density, for example, to do clusters of embedding.

15.1.1 Proprieties of time series

- **Trend** : A time-series has a trend, if its mean μ is not constant, but increases or decreases over time. A trend can be linear or non-linear.
- **Seasonality** : Seasonality is the periodic recurrence of fluctuations.
- **Stationarity** : A stationary time series is one whose properties do not depend on the time at which the series is observed. Thus, time series with trends, or with seasonality, are not stationary.

This state of the art was largely inspired by (Braei & Wagner, April 2020)¹

¹ Braei, M., & Wagner, S. (April 2020). Anomaly Detection in Univariate Time-series: A Survey on the State-of-the-Art. Retrieved from <https://arxiv.org/pdf/2004.00433.pdf>

15.1.2 Statistical Approach

- **AR:** The autoregressive model specifies that the output variable depends linearly on its own previous values and on a stochastic term.

$$AR(p) : X_t = c + \sum_{i=1}^p \varphi_i X_{t-i} + \varepsilon_t$$

- **MA :** Moving-average model

$$MA(q) : X_t = \mu + \varepsilon_t + \sum_{i=1}^q \theta_i \varepsilon_{t-i}$$

- **ARMA :** mixture of moving average and auto-regressive model

$$ARMA(p, q) : X_t = c + \varepsilon_t + \sum_{i=1}^p \varphi_i X_{t-i} + \sum_{i=1}^q \theta_i \varepsilon_{t-i}$$

- **ARIMA :** ARMA + differentiation
- **Exponential Smoothing :** Alternative to moving average
- **Seasonal ESD (S-ESD) and Seasonal Hybrid ESD (S-H-ESD)**²: using of extreme studentized deviate to detect anomalies. The algorithm was decomposed into 4 steps :
 - Extract seasonal component using STL Variant
 - Compute median
 - Compute residual
 - Detect anomalies vector XA using ESD
- **VAR, others models (multivariate ...)**

To see more about the statistical approach you can see the thesis (Statistiques en grande dimension pour la détection d'anomalies dans les données fonctionnelles issues des satellites)³

15.1.3 Machine Learning Approach

- **K-Means Clustering :** Subsequence time-series Clustering (STSC). Sliding window method, calculate centroid (need to give a number of cluster)
- **DBSCAN :** Based on density. We have to specify a distance (neighbors) and a minimum number of normal points on each cluster.
- **Local Outlier Factor (LOF) :** The Local Outlier Factor (LOF) algorithm is an unsupervised anomaly detection method which computes the local density deviation of a given data point with respect to its neighbors. It considers as outliers the samples that have a substantially lower density than their neighbors. It can be used to outlier detection and novelty detection.
- **Isolation Forest :** The Isolation Forest 'isolates' observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature. Not based on distance or density. (Outlier detection)
- **One-Class Support Vector Machines (OC-SVM) :** Works on distances, maximize the margin between normal and anomalies. Possibility to use kernel methods. Semi-supervised method, need to train on "normal time series".

² Hochenbaum, J., Vallis, O. S., & Kejariwal, A. (2017). Automatic Anomaly Detection in the Cloud Via Statistical Learning. arXiv e-prints. Retrieved from <https://arxiv.org/abs/1704.07706>

³ Barreyre, C. (2018). Statistiques en grande dimension pour la détection d'anomalies dans les données fonctionnelles issues des satellites. INSA de Toulouse. Retrieved from <https://tel.archives-ouvertes.fr/tel-01885331>

- **Gaussian distribution (EllipticEnvelope)** : Gaussian distribution is also called normal distribution. (More to detect outlier)
- **Extreme Gradient boosting (XGBoost, XGB)** : Used as regression model to forecast. To detect anomalies, we have to compute prediction error and use a threshold.

PCA (principal component analysis) : allows to reduce the dimension of data / **Fourier Transform** / **Wavelet decomposition** can be efficient to detect the most obvious anomalies. But if you want to detect precise anomalies it can be more difficult.

15.1.4 Deep Learning Approach

We eliminate any problems of preprocessing (if we have to do a preprocessing, it is simpler than other approaches)

- **MLP** (Multi-Layer Perceptron): Possibility to use MLP to predict time series with NNAR (Neural Network Autoregression Model). This model is equal to ARIMA model but no seasonal restriction exists. Hyperparameters : depth, width, length of the window, learning rate, optimization function.
- **CNN** (Convolutional Neural Network): Train faster because we use convolutional layers to extract features, CNNs work more than MLPs on local patterns. DeepAnT architecture (2018) . Uni or multi-variated. In contrast to the LSTM based approach, CNN based DeepAnT is not data hungry.

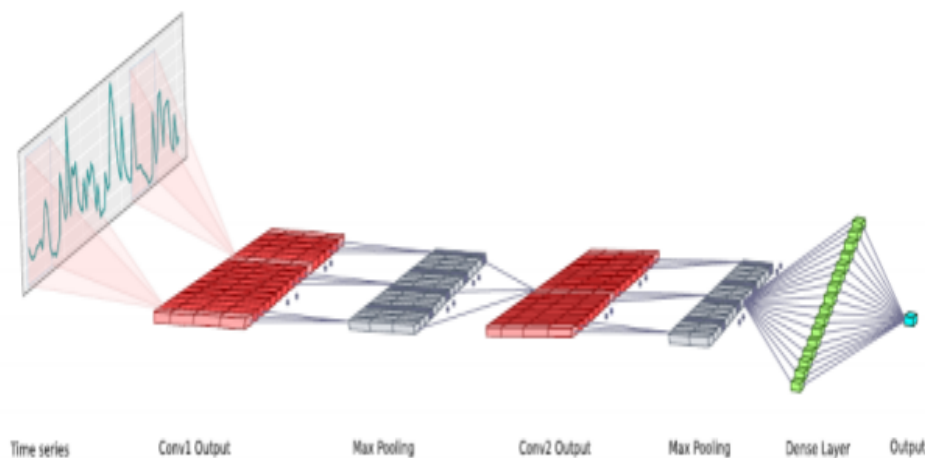


Fig. 1: DeepAnT architecture

- **ResNet** (Residual Neural Network) : CNNs with Residual blocks. Used to avoid vanishing gradient problem which occurs sometimes in complex CNNs. ResNet have achieved good results on computer vision.
- **LSTMs** : RNN (Recurrent Neural Network) construct to avoid vanishing gradient and to learn sequences. The main objectif of LSTMs is to keep in memory important information and transmit this information between synopsis of a same layer. They adapt many filters to treat the information in memory. We predict time series and we compute the error vector. This vector is used to fit to a multivariate Gaussian distribution and given a threshold, we can detect anomalies.
- **GRU** (Gated recurrent unit): Particular LSTM (simplified structure). Perform well as LSTM and more faster (because it is simpler)
- **LSTNet** : (Modeling Long- and Short-Term Temporal Patterns with Deep)⁴ Created for multivariate time series problems. First layer is a convolutional layer w/o pooling, we obtain a matrix of size number of filters * Time.

⁴ Lai, G., Chang, W.-C., Yang, Y., & Liu, H. (2018). Modeling Long- and Short-Term Temporal Patterns with Deep. ACM Conference (SIGIR'18), (p. 11). New York, NY, USA. Retrieved from <https://arxiv.org/pdf/1703.07015.pdf>

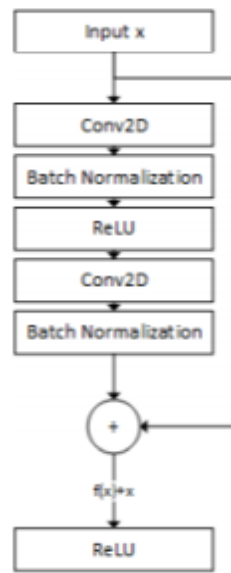


Fig. 2: ResNet architecture

Next we have a recurrent component (GRU + ReLU) followed by a recurrent skip component (to capture long term dependencies, hyperparameter p : number of hidden cells skipped through), a dense layer is used to combine these two results. We can use an alternative approach with attention mechanism which allows us to alleviate the problem of p (LSTNet-Skip for the first model, LSTNet-Attn for the second). At a same time, we add a linear model VAR. This allows us to focus on the local scaling issue. The final prediction is a sum of these two models.

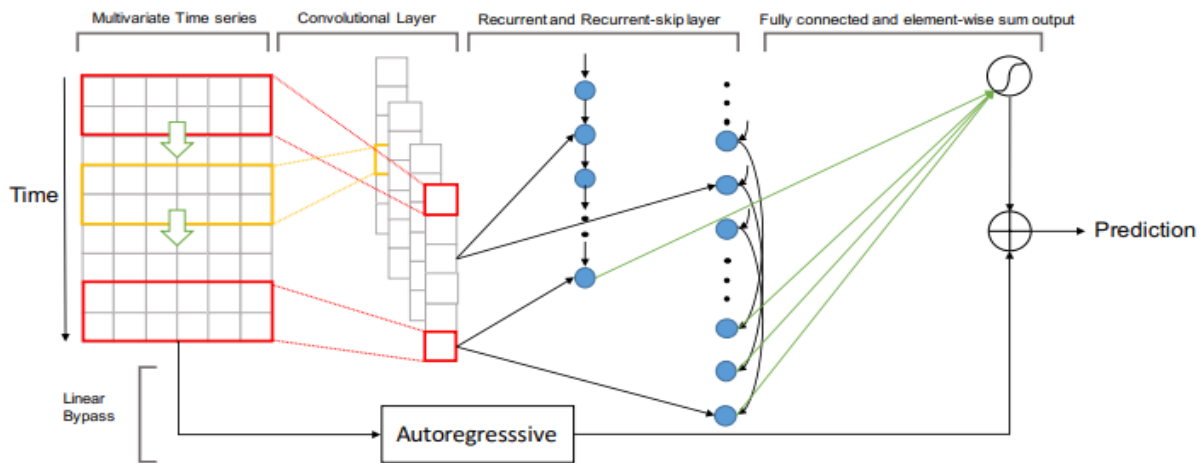


Fig. 3: LSTNet architecture

- **Deep SVDD** : (Deep One-Class Classification)⁵ One class classification for high dimensional data. Deep Support Vector Data Description (Deep SVDD), trains a neural network while minimizing the volume of a hypersphere that encloses the network representations of the data. Minimizing the volume of the hypersphere forces the network to extract the common factors of variation since the network must closely map the data points to the center of the sphere.

⁵ Ruff, L., Gornitz, N., Deecke, L., Siddiqui, S. A., Binder, A., Muller, E., & Kloft, M. (2018). Deep One-Class Classification. 35 th International Conference on Machine (p. 10). Stockholm, Sweden: PMLR 80,. Retrieved from <http://proceedings.mlr.press/v80/ruff18a/ruff18a.pdf>

- **THOC** : (Timeseries Anomaly Detection using Temporal Hierarchical One-Class Network)⁶
- **DROCC** : (Deep Robust One-Class Classification)⁷ possible on time series ?
- **ES-RNN** : Hybrid model, using LSTMs and exponential smoothing.
- **N-BEATS** : It was the first deep learning model to give better results than statistical models on M4 data set. It uses doubly residual stacking and can be interpret seasonality and trend of time series.

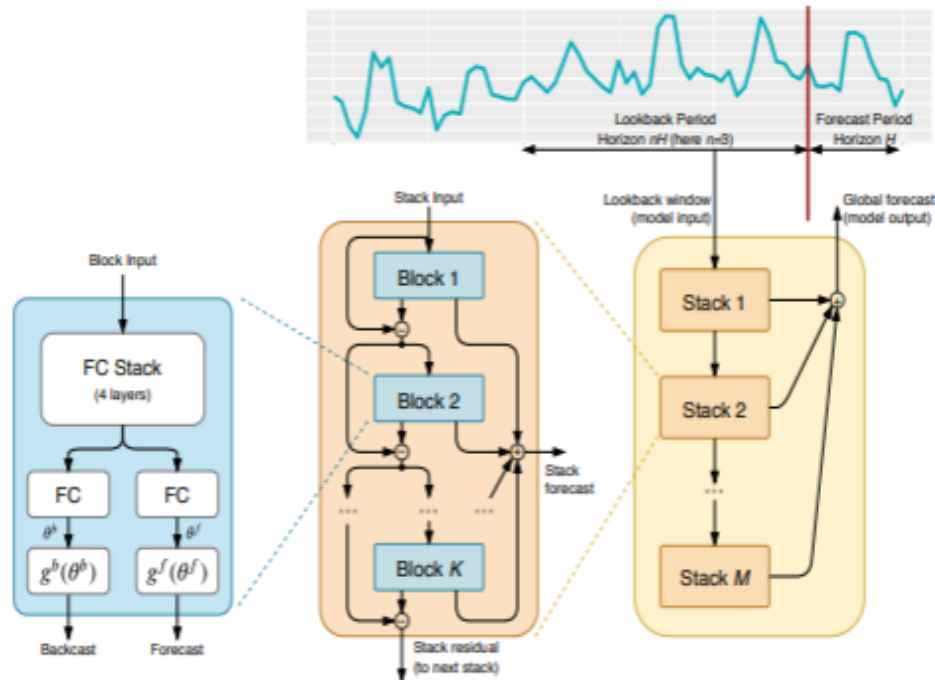


Fig. 4: N-BEATS architecture

- **Autoencoders** : Reduce dimensionality using an encoder (project the vector of the input in latent space) and reconstruct the signal with decoder. Train on normal data and predict a new value : if the new value error is superior to a given threshold, we detect it as an anomaly. If subsequences length increases, it is difficult to obtain good results.
- **USAD** : (UnSupervised Anomaly Detection on Multivariate Time Series)⁸ Autoencoders + GAN (Generative Adversarial Network), allows us to detect if input doesn't contain an anomaly (GAN), using an AE architecture, to eliminate problems of convergences during GAN training. 2 phases of training, first, we train 2 AE, with a same encoder. In a second time, we give the produce signal of AE1 to AE2 and it has to distinguish if is a normal signal or a created signal.

⁶ Shen, L., Li, Z., & Kwok, J. T. (2020). Timeseries Anomaly Detection using Temporal Hierarchical One-Class Network. 34th Conference on Neural Information Processing Systems (NeurIPS 2020), (p. 11). Vancouver, Canada. Retrieved from <https://proceedings.neurips.cc/paper/2020/file/97e401a02082021fd24957f852e0e475-Paper.pdf>

⁷ Goyal, S., Raghunathan, A., Jain, M., Simhadri, H., & Jain, P. (2020). Deep Robust One-Class Classification. 37 th International Conference on Machine (p. 16). PMLR 119. Retrieved from <https://arxiv.org/pdf/2002.12718.pdf>

⁸ Audibert, J., Michiardi, P., Guyard, F., Marti, S., & Zuluaga, M. A. (August 2020). USAD: UnSupervised Anomaly Detection on Multivariate Time Series. KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, (pp. 3395-3404). Virtual Event CA USA. Retrieved from <https://www.kdd.org/kdd2020/accepted-papers/view/usad-unsupervised-anomaly-detection-on-multivariate-time-series>

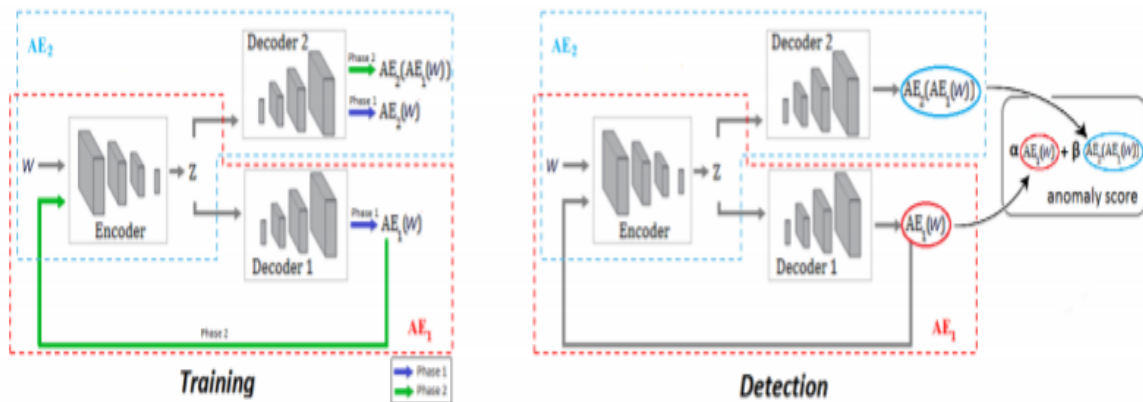


Fig. 5: USAD architecture

- **Transformers** :⁹ Faster than RNN to train (parallelizable). Distance between 2 distant tokens isn't taken in consideration, i.e. we can take long-term dependencies. Multi-head allows us to give many attention of relation between tokens.
- **Informer** : (Beyond Efficient Transformer for Long Sequence Time-Series Forecasting).¹⁰ Created to optimize the Transformer model. Indeed, in the case of forecasting timeseries the transformer takes a lot of GPU computing power. A recent paper, present Informer which gives some new techniques like ProbSparse to optimize the self-attention mechanism.

15.1.5 Summary table

To conclude, the statistical approach seems to be an old vision of this problem, but many studies demonstrate that these methods have good results. The machine learning approach seems more difficult to execute. The problem is that we have to do a very good preprocessing to achieve good results. The Deep learning approach can help us to get around this problem. Indeed, DL allows to produce information thanks to raw data. A lot of papers relate to new methods. Currently, the mechanism of attention seems to have good results for sequences with long term dependencies. However, each data set is different and it is difficult to find a general method.

This state of the art reports some models, but in no way, all of the models are mentioned.

15.1.6 References

⁹ Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (June 2017). Attention Is All You Need. arXiv e-prints, 15. Retrieved from <https://arxiv.org/pdf/1706.03762.pdf>

¹⁰ Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., & Zhang, W. (December 2020). Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. <https://arxiv.org/pdf/2012.07436v2.pdf>, 12.

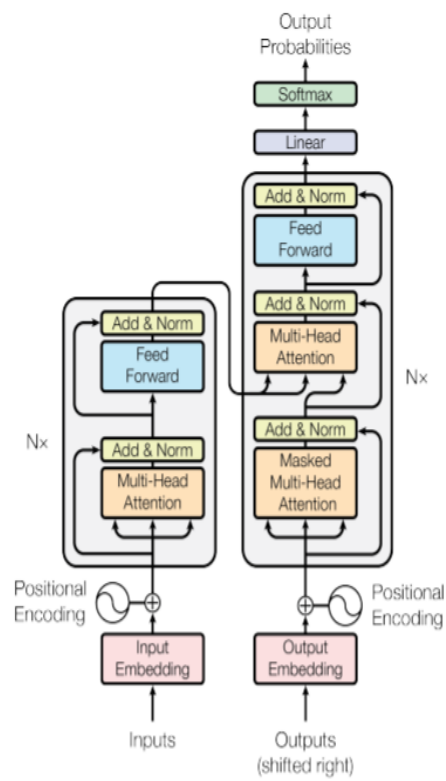


Fig. 6: Transformer architecture

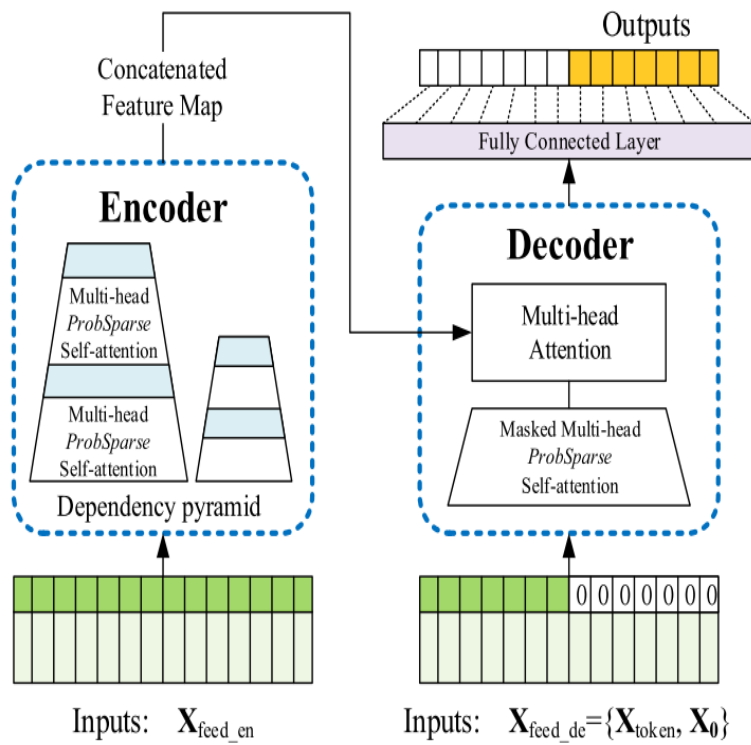


Fig. 7: Informer architecture

<i>Model name</i>	<i>Uni-variate/Multi-variate</i>	<i>Prediction/Clustering</i>
<i>Statistical approach</i>		
AR	Uni-variate	Prediction
MA	Uni-variate	Prediction
ARMA	Uni-variate	Prediction
ARIMA	Uni-variate/Multi-variate	Prediction
Exponential Smoothing	Uni-variate	Prediction
VAR	Multi-variate	Prediction
<i>Machine Learning approach</i>		
K-Means	Uni-variate	Clustering
DBSCAN	Uni-variate	Clustering
LOF	Uni-variate	Clustering
Isolation Forest	Uni-variate	Clustering
OC-SVM	Uni-variate	Clustering
Gaussian Distribution	Uni-variate	Clustering
XGBoost	Uni-variate	Prediction
<i>Deep Learning approach</i>		
DeepAnT	Uni-variate/ Multi-variate	Prediction
Resnet	Uni-variate/ Multi-variate	Prediction
LSTMs	Uni-variate/ Multi-variate	Prediction
GRU	Uni-variate/ Multi-variate	Prediction
LSTNet	Multi-variate	Prediction
Deep SVDD/ THOC	Uni-variate	Clustering
DROCC	?	Clustering
ES-RNN	Uni-variate/ Multi-variate	Prediction
N-BEATS	Uni-variate/ Multi-variate	Prediction
AE	Uni-variate/ Multi-variate	Prediction
USAD	Uni-variate/ Multi-variate	Prediction
Transformers	Uni-variate/ Multi-variate	Prediction
Informers	Uni-variate/ Multi-variate	Prediction

Fig. 8: Summary table

ROBOT FRAMEWORK

16.1 Introduction

Skyminer uses Robot Framework for Functional Acceptance Tests

Links:

- [Robot Framework](#)
- [Robot Framework Github Repo](#)

16.1.1 Installation

```
pip install robotframework
```

16.1.2 Built-in features

- [OperatingSystem](#)
- [Process](#)
- [Screenshot](#)

16.1.3 Third-party features

- [Robot & Selenium](#)

```
pip install robotframework-seleniumlibrary
```

- [Pypi](#)
- [Documentation](#)

- [RPA framework](#)

```
pip install rpaframework
```

- [Pypi](#)
- [Github](#)
- Used for JSON: [How to](#)

- Used for REST API: [How to](#)

16.2 Code and Features

- *lib/skyminer_variables.py*: Good practice file to define common variables with Python code
- *lib/skyminer_resource.robot*: Good practice robot file for common configuration and keywords.
 - **Important** All the keywords defined in Skyminer starts with *Sk* to not have possible conflicts with open source keywords libraries.
- *skyminer_<feature>.robot*: Test file for feature

16.3 Run Tests

Run a single test:

```
robot -v SKYMINER_VERSION:2021-dev-S2 -d output skyminer_system.robot
```

SKYMINER_VERSION is a **must have** variable in cli.

16.4 Generate tests documentation

16.4.1 Libraries

```
pip install mdutils
```

This library is used to convert output.xml produced by Robot to a Markdown file with formatted results.

Documentation [Example](#)

16.4.2 Pandoc

- Official site: [Pandoc](#)
- Documentation: [Manual](#)

HTML output:

```
pandoc -s skyminer-tests.md -c kratos.css --metadata title="Skyminer Tests Documentation" -o skyminer-tests.html
```

kratos.css is a classless CSS file based on example found [here](#)

PDF output:

```
pandoc -t pdf --pdf-engine wkhtmltopdf skyminer-tests.md -o skyminer-tests.pdf
```

PDF engine used by Pandoc could be [wkhtmltopdf](#) but the result needs more refinement.

16.5 Knowledge Base

- Tuto
- Dockerize
- Example
- Tests good practice
- Misc

SIMILARITY SEARCH

Overview

- *Similarity Search*
 - *Introduction*
 - *Exact methods*
 - *Approximate methods*
 - *Conclusion*

17.1 Introduction

Similarity search is the operation of finding the set of data series in a collection, which is close to a given query series according to some definition of distance. Most data mining algorithms (e.g. clustering, classification, outlier detection, etc.) use similarity measures and similarity search as subroutines. Interpretable mining of heterogeneous data objects is possible through similarity search. A lots of methods have been proposed in the past two decades. There are two effective types of Similarity Search: exact and approximate. The aim is to present state-of-the-art techniques in similarity search on time series.

17.2 Exact methods

The similarity search algorithms that always produce correct and complete answers are called exact. There are two effective Exact Similarity Search Methods in different field, sequential scan and indexing.

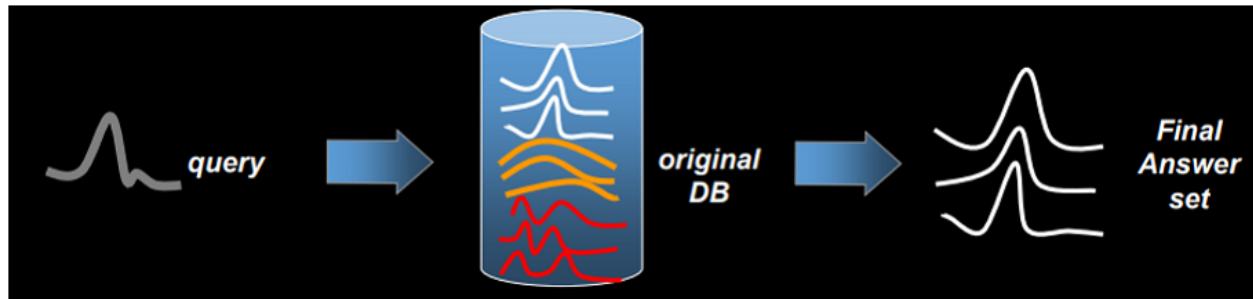
17.2.1 Sequential scan (a.k.a. linear scan)

17.2.1.1 Overview

1

A trivial solution to the similarity search problem is sequential scanning, also known as linear scanning, where the query is compared against all the data objects in the data base. This method presents good accuracy for similarity search. Yet, one of its main problems lies in the computational complexity of its execution. Therefore, its application becomes not viable for a series that presents a large amount of points. Even though many improvements has been found

¹ <https://www.cs.unm.edu/~mueen/Tutorial/CIKM2016Tutorial.pdf>



to speed up this method. Recent works have shown that sequential scans can be performed efficiently, such techniques are mostly applicable when the dataset consists of a single, very long data series, and queries are looking for potential matches in small subsequences of this long data series. Such approaches, in general, do not provide any benefit when the dataset is composed of a large number of small data series. Therefore, indexing is required in order to efficiently support data exploration tasks.

17.2.1.2 State-of-the-art techniques

1. UCR Suite²

The UCR Suite is an optimized sequential scan algorithm for exact subsequence matching. It use all applicable speedup techniques (early abandoning, just-in-time normalizations, reordering...)

Characteristics:

- Very fast for subsequence matching.
- It works for Euclidean distance and DTW(Dynamic Time Warping).
- It does not require parameters to be set.
- It requires zero preprocessing time.
- The same idea works for both streaming data, and batch offline search.
- Finally UCR-Suite is the fastest sequential scan method under DTW

2. MASS³

MASS is an exact subsequence matching algorithm, which computes the distance between a query, and every subsequence in the series, using the dot product of the DFT(Discrete Fourier Transform) of the series and the reverse of the query.

Characteristics:

- Very fast for subsequence matching.
- Produces a “distance profile” of the query to the subsequences of the time series. Every distance is reported, nothing is abandoned.
- It Can be used to answer K-NN queries, range queries, and density estimation all at the same time.
- Data and query independent execution.

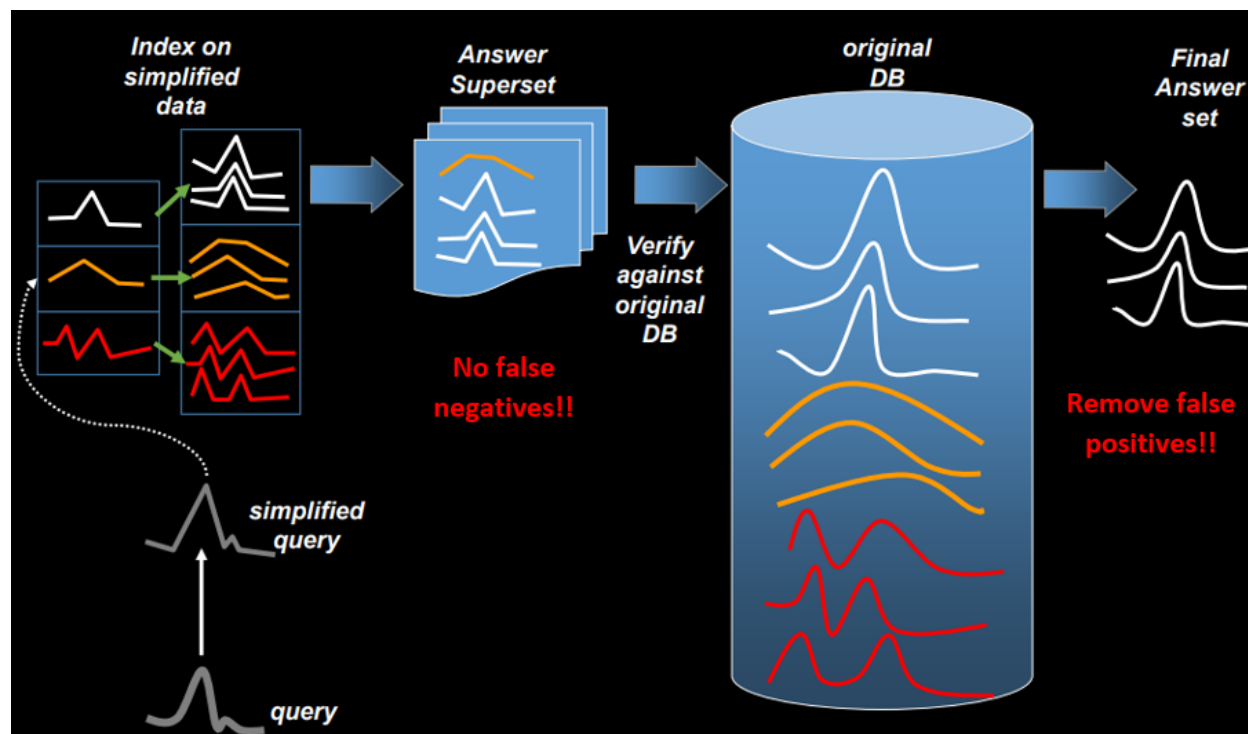
² C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, Z. Zimmerman, D. F. Silva, A. Mueen, and E. Keogh. Time series joins, motifs, discords and shapelets: a unifying view that exploits the matrix profile. Data Mining and Knowledge Discovery, pages 1–41, 2017. <https://www.cs.unm.edu/~mueen/FastestSimilaritySearch.html>

³ T. Rakthanmanon, B. J. L. Campana, A. Mueen, G. E. A. P. A. Batista, M. B. Westover, Q. Zhu, J. Zakaria, and E. J. Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In Q. Yang, D. Agarwal, and J. Pei, editors, KDD, pages 262–270. ACM, 2012. <https://www.cs.ucr.edu/~eamonn/UCRsuite.html>

- Can be further optimized when multiple queries are issued
- Finally MASS is the fastest sequential scan method under Euclidean distance

17.2.2 Indexing (a.k.a. GEMINI)

17.2.2.1 Overview



4

The first step is to define a dimensionality reduction technique for the data. That is, reducing the size (i.e., length) of the data, one is able to efficiently discard many of the time series which are not similar to the query, and therefore efficiently reduce the number of potential “false positives”. This could be achieved by mapping the objects of our database to a new representation space (called feature space), where the distance function could be computed efficiently. The distance between two objects in the feature space should be underestimated with respect to the corresponding distance in the original space (this property of the distance function is known as lower bounding), otherwise “false negatives” could possibly break down the effectiveness of the similarity search method.

Indexing methods outperform sequential scan for whole matching. But Subsequences have large overlaps which they do not exploit so sequential scan can be better for subsequence matching especially on small dataset. In addition most of indexing methods can't handle variable length query, require parameters to be set and long index build time. Indexing is not efficient for performing a small number of queries.

⁴ <https://www.cs.unm.edu/~mueen/Tutorial/CIKM2016Tutorial.pdf>

17.2.2.2 State-of-the-art techniques

1. DSTree⁵

The DSTree approach uses the EAPCA summarization technique, which allows, during node splitting, the resolution of a summarization to increase along two dimensions: vertically and horizontally. (Instead, SAX-based indexes allow horizontal splitting by adding a breakpoint to the y-axis, and SFA allows vertical splitting by adding a new DFT coefficient.) In addition to a lower bounding distance, the DSTree also supports an upper bounding distance. It uses both distances to determine the optimal splitting policy for each node.

Characteristics:

- Very fast on very large dataset
- Very fast during query answering
- Very long index building time

2. VA+file⁶

The VA+file is an improvement of the VA-file method⁷. While both methods create a filter file containing quantization-based approximations of the high dimensional data, and share the same exact search algorithm, the VA+file does not assume that neighboring points (dimensions) in the sequence are uncorrelated. It thus improves the accuracy of the approximations by allocating bits per dimension in a non-uniform fashion, and partitioning each dimension using a k-means.

Characteristics:

- Very fast on long time series

3. SFA trie⁸

The SFA approach first summarizes the series using SFA of length 1 and builds a trie with a fan out equal to the alphabet size on top of them. As leaves reach their capacity and split, the length of the SFA word for each series in the leaf is increased by one, and the series are redistributed among the new nodes. The maximum resolution is the number of DFT coefficients given as a parameter. SFA implements lower-bounding to prune the search space, as well as a bulk-loading algorithm.

Characteristics:

- Subsequence matching implementation

iSAX Index Family is a family of indexing methods based on the iSAX summarization technique

Timeline depicted on top; implementation languages marked on the right. Solid arrows denote inheritance of index design; dashed arrows denote inheritance of some of the design features; two new versions of iSAX2+/ADS+ marked with asterisk support approximate similarity search with deterministic and probabilistic quality guarantees.⁹

4. iSAX2+¹⁰

⁵ Y. Wang, P. Wang, J. Pei, W. Wang, and S. Huang. A data-adaptive and dynamic segmentation index for whole matching on time series. PVLDB, 6(10):793–804, 2013. <https://www.cs.sfu.ca/~jpei/publications/time%20series%20index%20VLDB13.pdf>

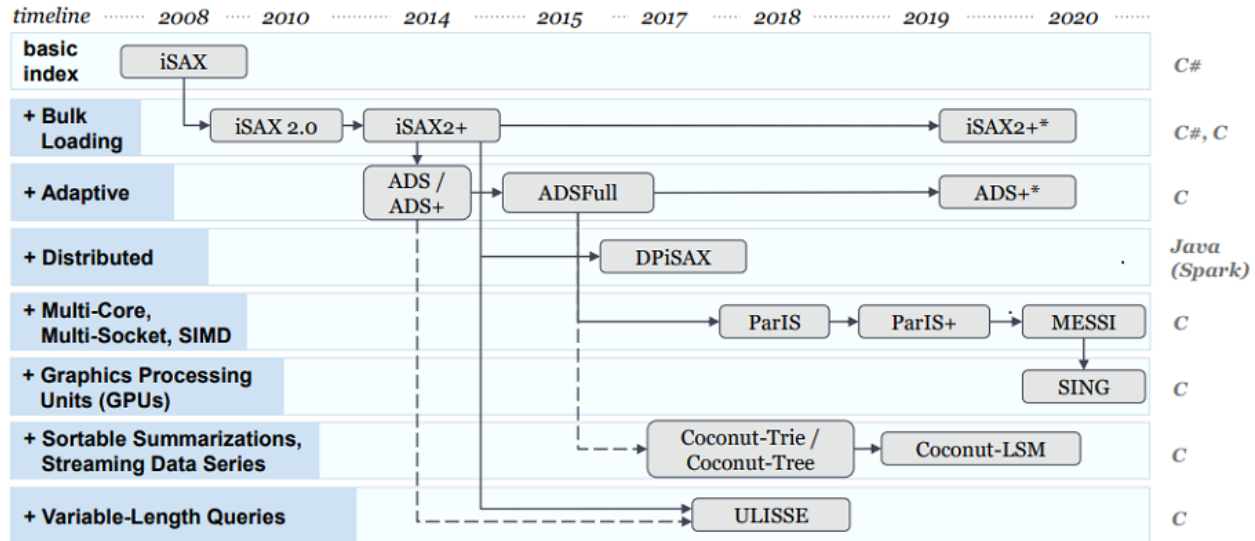
⁶ H. Ferhatosmanoglu, E. Tuncel, D. Agrawal, and A. E. Abbadi. Vector approximation based indexing for nonuniform high dimensional data sets. In In Proceedings of the 9th ACM Int. Conf. on Information and Knowledge Management, pages 202–209. ACM Press, 2000.

⁷ R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In Proceedings of the 24th International Conference on Very Large Data Bases, VLDB '98, pages 194–205, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

⁸ P. Schafer and M. Hoggqvist. Sfa: A symbolic fourier approximation and index for similarity search in high dimensional datasets. In Proceedings of the 15th International Conference on Extending Database Technology, EDBT '12, pages 516–527, New York, NY, USA, 2012. ACM. <https://openproceedings.org/2012/conf/edbt/SchaferH12.pdf>

⁹ <http://helios.mi.parisdescartes.fr/~themisp/publications/iscc20-bigsequencemanagement.pdf>

¹⁰ A. Camerra, J. Shieh, T. Palpanas, T. Rakthanmanon, and E. J. Keogh. Beyond one billion time series: indexing and mining very large time series collections with isax2+. Knowl. Inf. Syst., 39(1):123–151, 2014.



iSAX2+ improved the splitting policy and added bulk-loading support to the original iSAX index it is a direct improvement of iSAX2.0¹¹

Characteristics:

- Very fast on very small dataset
- Very fast on short series
- Average indexing and query answering time

5. ADS+¹²

ADS+ is the first query adaptive data series index. It first builds an index tree structure using only the iSAX summarizations of the raw data, and then adaptively constructs the leaves and incorporates the raw data during query answering. It first performs a fast ng-approximate search in the tree in order to acquire an initial best-so-far (bsf) distance, then prunes the search space by using the bsf and the lower bounds between the query and all iSAX summaries. Using that, it performs a skip-sequential search on the raw data that were not pruned. ADS-FULL is a non-adaptive version of ADS, that builds a full index using a double pass on the data.

Characteristics:

- Very fast index building
- Very fast on long data series

iSAX family extensions

6. Coconut¹³

Coconut is the current solution for limited memory devices and streaming time series

7. ULISSE¹⁴

ULISSE is the current solution for variable-length queries

¹¹ A. Camerra, T. Palpanas, J. Shieh, and E. J. Keogh. isax 2.0: Indexing and mining one billion time series. In G. I. Webb, B. Liu, C. Zhang, D. Gunopulos, and X. Wu, editors, ICDM, pages 58–67. IEEE Computer Society, 2010.

¹² K. Zoumpatianos, S. Idreos, and T. Palpanas. ADS: the adaptive data series index. VLDBJ, 25(6):843–866, 2016.

¹³ Haridimos Kondylakis, Niv Dayan, Kostas Zoumpatianos, and Themis Palpanas. 2018. Coconut: A Scalable Bottom-Up Approach for Building Data Series Indexes. PVLDB 11, 6 (2018), 677–690.

¹⁴ M. Linardi and T. Palpanas. ULISSE: ULtra compact Index for Variable-Length Similarity SEarch in Data Series. In ICDE, 2018.

*iSAX family parallelization*8. **DPiSAX**¹⁵

DPiSAX is the current solution for distributed processing (Spark), it balances work of different worker nodes, it performs 2 orders of magnitude faster than centralized solution

9. **ParIS**¹⁶

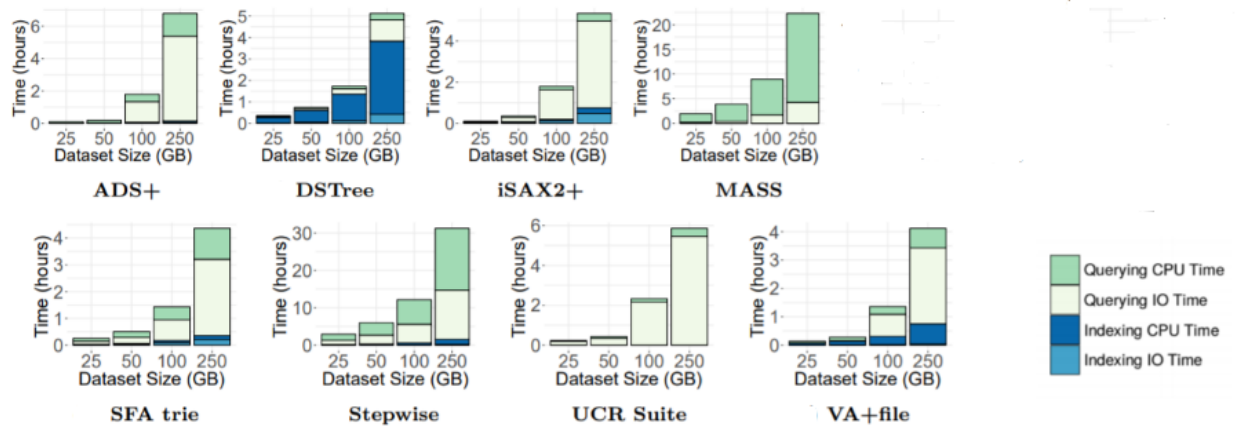
ParIS is the current solution for modern hardware, it masks out the CPU cost, it answers exact queries in the order of a few secs, 3 orders of magnitude faster then single-core solutions

10. **MESSI**¹⁷

MESSI is the current single-node parallel solution + in-memory data, it answers exact queries at interactive speeds: ~50msec on 100GB

11. **SING**¹⁸

SING is the current single-node parallel solution + GPU + in-memory data, it answers exact queries at interactive speeds: ~32msec on 100GB

17.2.3 Experimental Evaluation

The figure shows evaluation of indexing and search efficiency of the methods on whole matching by varying the dataset size. We used two datasets of size 25GB and 50GB that fit in memory and two datasets of size 100GB and 250GB that do not fit in memory (total RAM was 75GB), with the Synth-Rand query workload (100 queries).¹⁹

¹⁵ D. E. Yagoubi, R. Akbarinia, F. Masseglia, and T. Palpanas. 2017. DPiSAX: Massively Distributed Partitioned iSAX. In ICDM. 1135–1140.

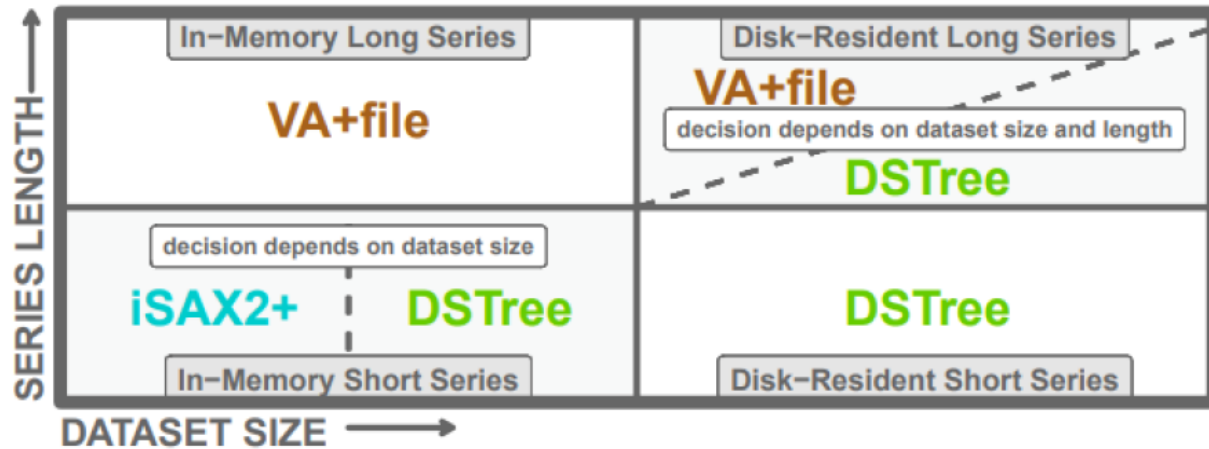
¹⁶ Botao Peng, Panagiota Fatourou, and Themis Palpanas. 2020. ParIS+: Data Series Indexing on Multi-core Architectures. TKDE (2020).

¹⁷ B. Peng, P. Fatourou, and T. Palpanas. MESSI: In-memory data series indexing. In ICDE, 2020.

¹⁸ Botao Peng, Panagiota Fatourou, and Themis Palpanas. 2021. SING: Sequence Indexing Using GPUs. In ICDE.

¹⁹ K. Echihabi, K. Zoumpatianos, T. Palpanas, and H. Benbrahim. The Lernaean Hydra of Data Series Similarity Search: An Experimental Evaluation of the State of the Art. PVLDB, 12(2):112–127, 2018. <http://helios.mi.parisdescartes.fr/~themisp/publications/pvldb20-lernaeanhydra2.pdf>

17.2.4 Recommendations



Recommendations for whole matching (Indexing and answering 10K queries)²⁰

17.3 Approximate methods

Algorithms that do not satisfy the exact property are called approximate. There are three types of approximate algorithms:

- An **e-approximate** algorithm guarantees that its distance results have a relative error no more than ϵ , i.e., the approximate distance is at most $(1 + \epsilon)$ times the exact one.
- A **d-e-approximate** algorithm, guarantees that its distance results will have a relative error no more than ϵ (i.e., the approximate distance is at most $(1 + \epsilon)$ times the exact distance), with a probability of at least d .
- An **ng-approximate** (no-guarantees approximate) algorithm does not provide any guarantees (deterministic, or probabilistic) on the error bounds of its distance results.

17.3.1 State-of-the-Art for Multidimensional Vectors

12. HNSW²¹

HNSW is an in-memory ng-approximate method that belongs to the class of proximity graphs that exploit two fundamental geometric structures: the Voronoi Diagram (VD) and the Delaunay Triangulation (DT).

Characteristics:

- Very fast for small time series
- ng-approximate

13. QALSH²²

²⁰ K. Echihabi, K. Zoumpatianos, T. Palpanas, and H. Benbrahim. The Lernaean Hydra of Data Series Similarity Search: An Experimental Evaluation of the State of the Art. PVLDB, 12(2):112–127, 2018. <http://helios.mi.parisdescartes.fr/~themisp/publications/pvldb20-lernaeanhydra2.pdf>

²¹ Y. A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. CoRR, abs/1603.09320, 2016.

²² Q. Huang, J. Feng, Y. Zhang, Q. Fang, and W. Ng. Query-aware Locality-sensitive Hashing for Approximate Nearest Neighbor Search. PVLDB, 9(1):1–12, 2015.

The LSH family²³ encompasses a class of randomized algorithms that solve approximate nearest neighbor problem in sub-linear time. The main intuition is that two points that are nearby in a high dimensional space, will remain nearby when projected to a lower dimensional space. In this work, we select QALSH to represent the class of LSH techniques because it's considered the state-of-the-art in terms of accuracy.

Characteristics:

- Very fast for small time series
- d-e-approximate

14. **Flann**²⁴

Flann is an in-memory ensemble technique used to approximate nearest neighbors search in high-dimensional spaces. Given a dataset and a desired search accuracy, Flann selects and auto-tunes the most appropriate algorithm among randomized kd-trees and a new proposed approach based on hierarchical k-means trees.

Characteristics:

- Very fast for small time series
- ng-approximate

17.3.2 State-of-the-Art for Data Series

15. **iSAX2+**

New versions of iSAX2+ support all types of approximate similarity search.

15. **ADS+**

New versions of ADS+ support all types of approximate similarity search.

16. **DSTree**

New versions of DSTree support all types of approximate similarity search.

17.3.3 Recommendations

*Recommendations for whole matching (Answering queries)*²⁵

17.4 Conclusion

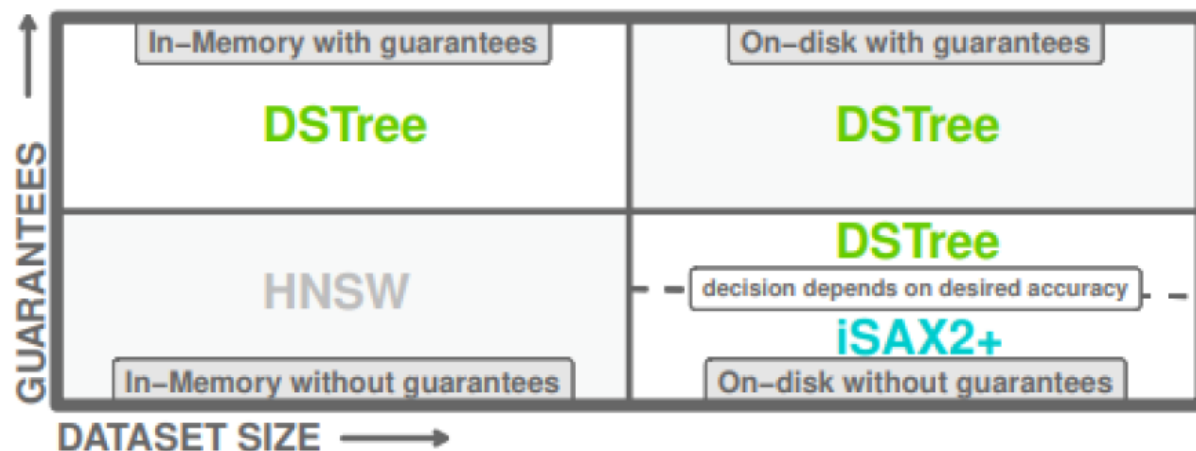
For the newcomers, I advise you to consult the newcomers.rst file.

²³ A. Andoni, P. Indyk, and I. P. Razenshteyn. Approximate Nearest Neighbor Search in High Dimensions. CoRR, abs/1806.09823, 2018.

²⁴ M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In VISAPP International Conference on Computer Vision Theory and Applications, pages 331–340, 2009

²⁵ K. Echihabi, K. Zoumpatianos, T. Palpanas, and H. Benbrahim. Return of the lernaean hydra: Experimental evaluation of data series approximate similarity search. PVLDB, 2019. <http://helios.mi.parisdescartes.fr/~themisp/publications/pvladb20-lernaeanhydra2.pdf>

²⁶ Pinos Bruno, Notebooks jupyter 2021 (draft) file://shackleton/153_Skyminer/Technical_Research/2021/similarity_search_internship/test_notebook/



Algorithm	Exact	Approximate	Representation	Subsequence Matching	Whole matching	Index Type	Linear Scan	Tested
MASS	✓		DFT	✓			✓	✓
UCR-Suite	✓		Raw	✓			✓	✓
DSTree	✓	✓	EAPCA		✓	Tree		
VA+file	✓	✓	DFT		✓	Filter file		
SFA trie	✓		SFA	✓	✓	Tree		
iSAX2+	✓	✓	iSAX		✓	Tree		
ADS+	✓	✓	iSAX		✓	Tree		
HNSW		✓	Raw		✓	Graph		✓
QALSH		✓	Signatures		✓	LSH		
Flann		✓	Raw		✓	Tree		

Fig. 1: Overview of all similarity search SOTA techniques

Algorithm	Language	Library	Index building time (s)	Query answering time (ms)	Metric	Comments
MASS	Python	Stumpy		386	Euclidean	
MASS	Python	mass-ts		1107	Euclidean	
MASS	Python	matrix-profile-ts		481	Euclidean	
UCR-Suite	Python	ucrdtw		565	DTW	1NN search only
UCR-Suite	R	IncDTW		582	DTW	
kd-tree	Python	scikit-learn	75	825	Euclidean	Preprocessing
kd-tree	Python	tslearn	127	599	Euclidean	Preprocessing
ckd-tree	Python	scipy	22	116	Euclidean	Preprocessing
hsw	Python	nmslib	240	7	Euclidean	Preprocessing

Fig. 2: Subsequence matching Test on one time series of length 3 millions and Pattern of length 180²⁶