Internship Summary 2020

Gong Na

BOSCH
Invented for life

Bosch-Group

Bosch Center for Artificial Intelligence

General Information

Duration: Oct.2019 – Mar.2020

Position: Intern of DevOps/Data Engineer

Supervisor: Martin Kopp (CR/PJ-AI-C4)

Location: Renningen

Task Overview

Task 1: **Set up a Central Logging System by using EFK stack** (work with 3 colleauges)

- Use cases:
  - a) Project A is able to use Central Logging System to track all logs from different components
  - b) All developers of the Project A can access Central Logging System via Bosch Account
  - c) Project A cannot access logs of other projects without permission
  - d) Project A can identify logs from different component with user-defined index pattern
  - e) Central Logging System will delete the expired logs automatically with pre-defined settings
- Contribution:
  - ✓ Proof of concept
  - ✓ [feature] Fluentd Configuration
  - ✓ [feature] Python Fluentd Handler pkg
  - ✓ [feature] Index Sate Management

Task 2: **Develop the VMPS dashboard** (single project – end2end)

- Use cases:
  - a) Customers and managers can check the statistical status of all vmps tasks in real-time manner
  - b) They can select the time frame and different dimension of different charts
- Contribution:
  - ✓ UI design
  - ✓ Frontend, Backend, Deployment

Tool Overview

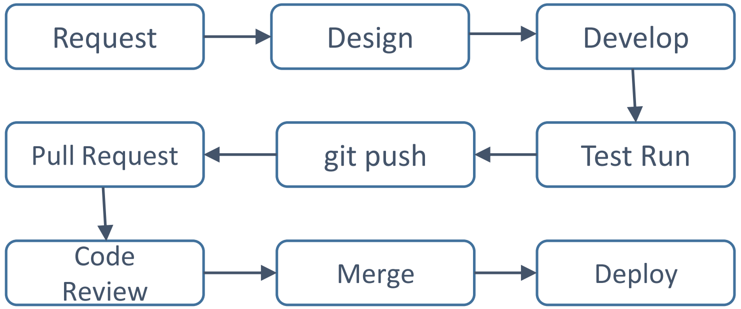
Task 1: Central Logging

- Elasticsearch / Open Distro
- Fluentd
- Kibana
- Docker Compose
- Vault
- Artifactory
- Makefile
- YAML (commonly used for configuration files)
- RestAPI
- JSON format (Elasticsearch console needs)
- DIE: VS Code (Pycharm + JupyterNotebook)

Task 2: VMPS Dashboard

- Docker
- Flask
- MongoDB
- Jinja
- Dashboard pipeline
- Python Web App Architecture: Gunicorn, NGINX

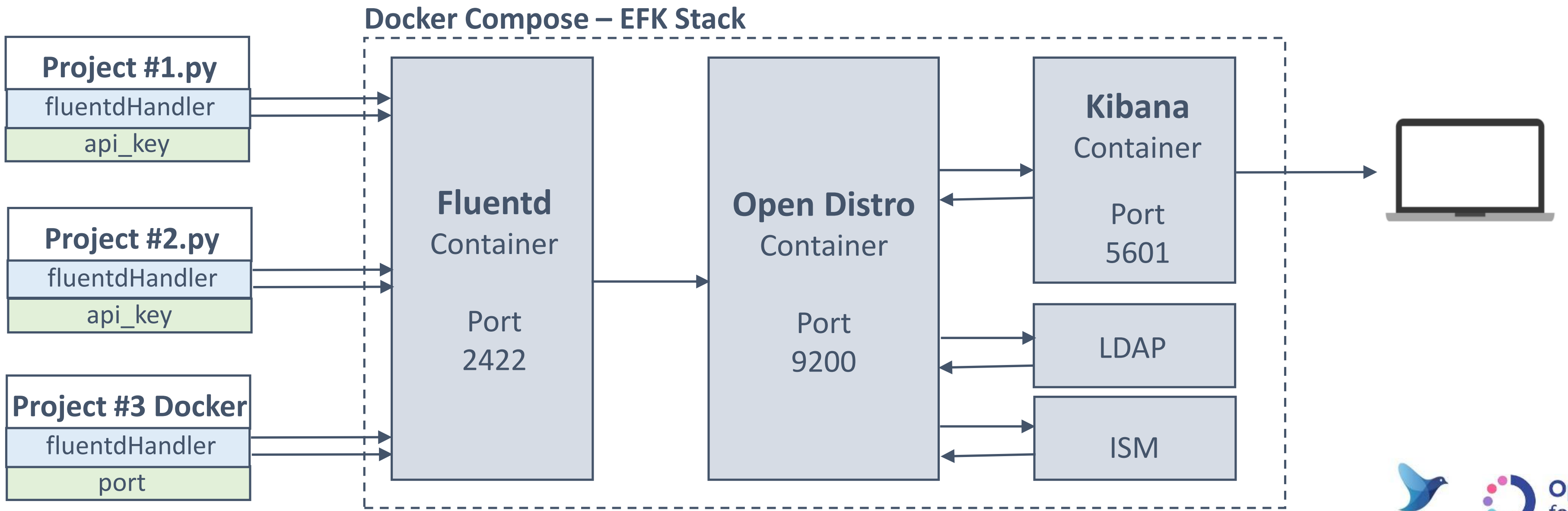
- Agile Development Process **(CI/CD):**



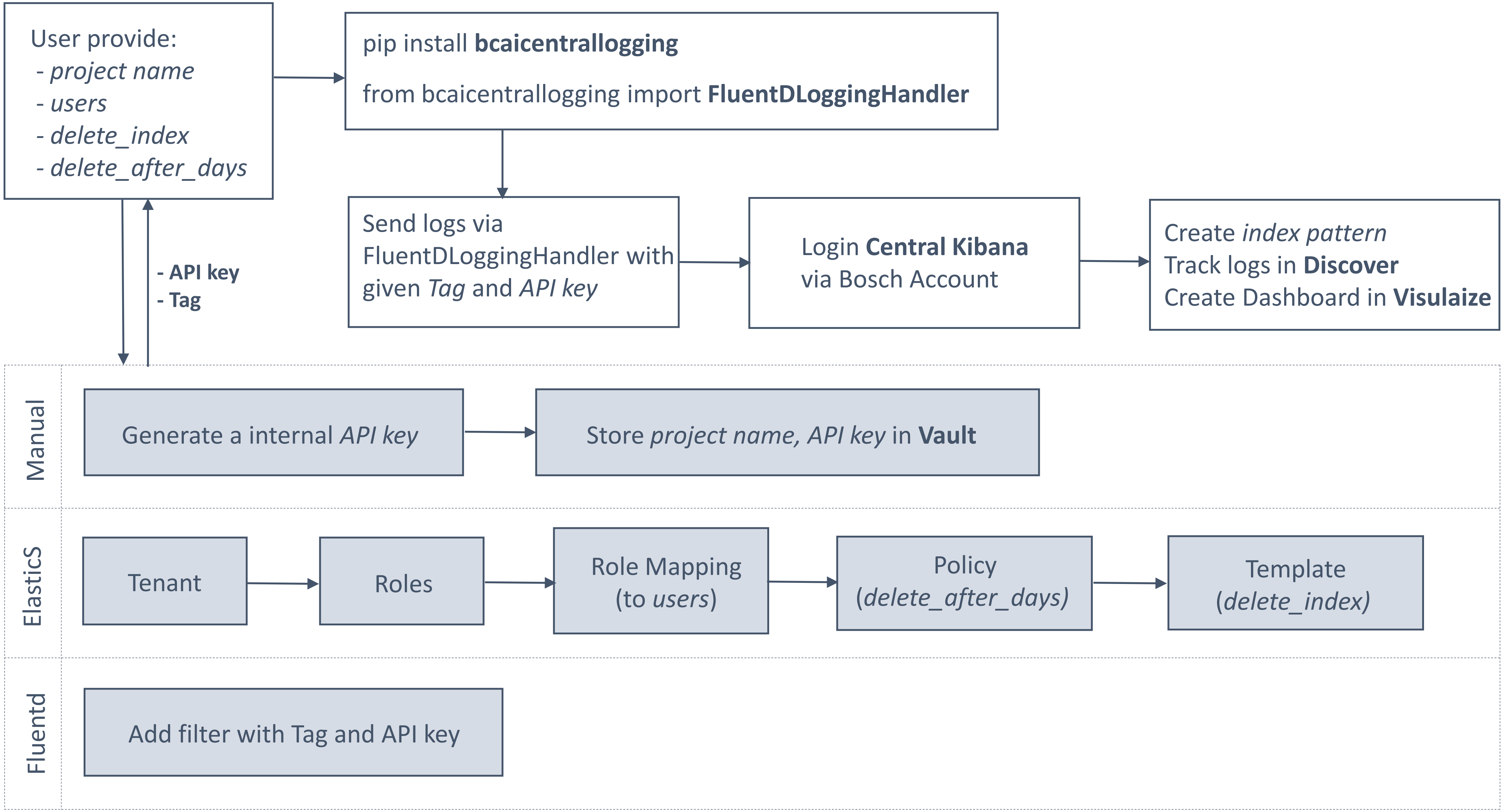
Central Logging System

Central Logging – EFK Stack

- **Fluentd:** a more flexible alternative of LogStash to forward logs to Opendistro based on the tag. It supports LogStash format as well.
- **Elasticsearch**: It provides a distributed, multitenant-capable full-text search engine with a HTTP web interface and schema-free JSON documents. (I understand it as a storage.)
- **Open Distro for Elasticsearch:** an alternative of Elasticsearch which is equipped with security management (e.g. user authentication which requires login and allows to enable LDAP), launched by Amazon Web Service (AWS).
- **Kibana:** UI which lets you visualize the Elasticsearch data and do standard configurations of Elasticsearch.
- **LDAP:** it is like a tree-shaped database to support faster query. Usually used to store data which won't change frequently such as user accounts. Straight forward because companies are always organized in tree structure.
- **Index Sate Management (ISM):** plugin of Open Distro, an alternative of Curator which manages time-series index (delete old index, etc) by creating policies.



Central Logging – Design of Workflow



User provide:

- *project name*
- *users*
- *delete_index*
- *delete_after_days*

pip install **bcaicentrallogging**

from bcaicentrallogging import **FluentDLoggingHandler**

Send logs via FluentDLoggingHandler with given *Tag* and *API key*

Login **Central Kibana** via Bosch Account

Create *index pattern*
Track logs in **Discover**
Create Dashboard in **Visulaize**

Generate a internal *API key*

Store *project name, API key* in **Vault**

Tenant

Roles

Role Mapping (to *users*)

Policy (*delete_after_days)*

Template (*delete_index)*

Add filter with Tag and API key

Central Logging System

Central Logging - Implementation

EFK

- Set up EFK stack using official images via *docker-compose.yml*
- Connect Kibana with Elasticsearch directly in *docker-compose.yml* via *environment* variable (*ELASTICSEARCH_URL, ELASTICSEARCH_HOSTS*)
- Connect fluentd with Elasticsearch via fluent.conf:

```
<store>
   @type elasticsearch
   host odfe-node1
   port 9200
   user  fluentd
   password fluentd
   …
</store>
```

Fluentd.conf

- Add *<filter></filter>* to restrict log forwarding via *API key.* Fluentd will forward the log with tag xxx only if it's api key is correct. Therefore, project A's log wont mess up with Project B's log
- Create a Elasticsearch user *fluentd* with all write permission, this gives authority to Fluentd to forward logs to Elasticsearch.
- Enable LogStash format to generate time-series index
- Add index prefix *logstash_prefix "#{ENV['ODFE_FLUENTD_PREFIX']}${tag}"*

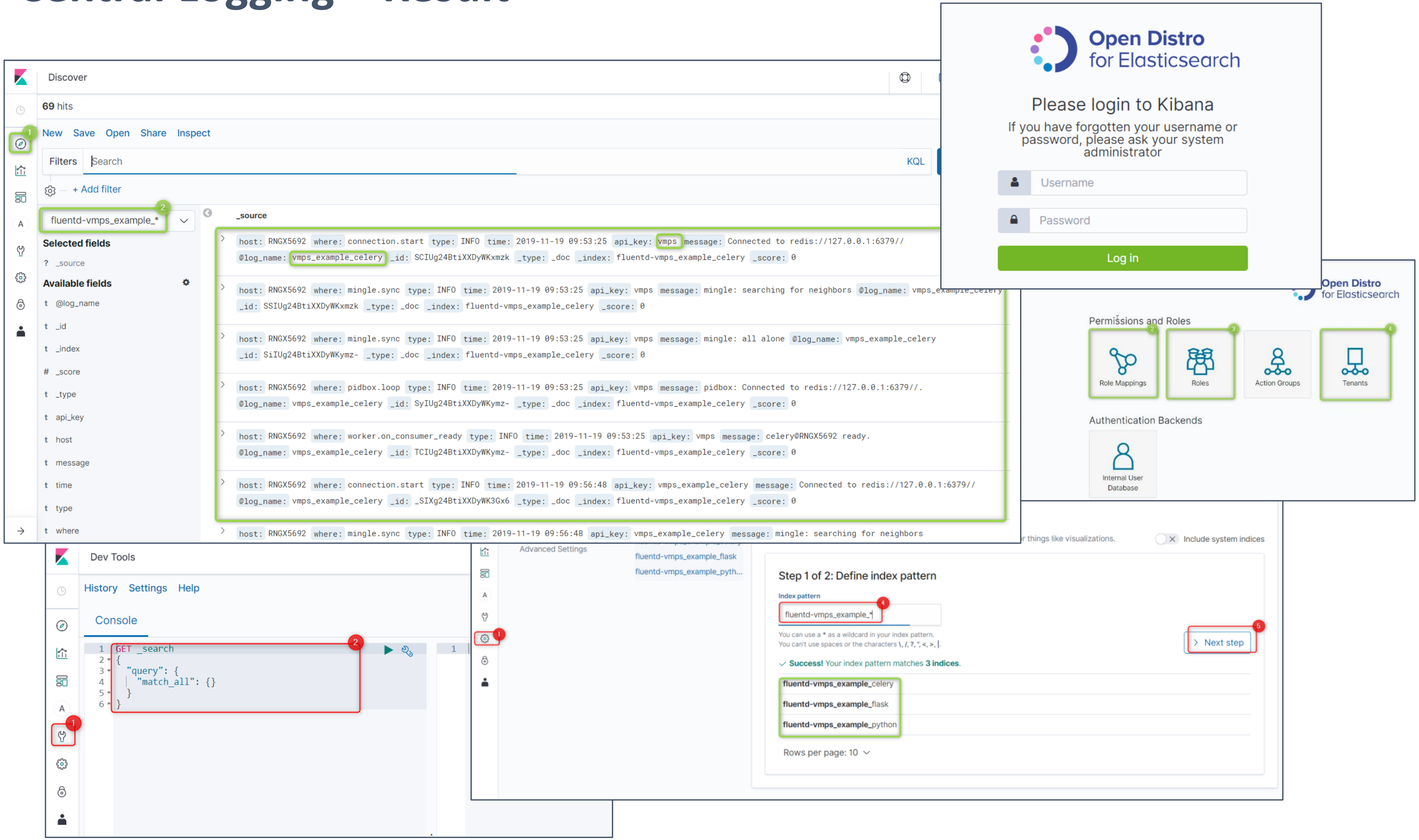
Fluentd Python pkg

- Write a python fluentd handler to record the events from Python application
- Deploy it by uploading to **Artifactory**
- Therefore, others can do pip install

Automatic Tooling

- Store user inputs in *inventory.yml*
- Automate the Elasticsearch configuration via REST api (you can do it manually via Kibana UI or console) and adding Fluentd filter
- Restart/Stop docker-compose or Fluentd via Makefile
- Run it with Vault token (if you do not have access to Vault, you cannot execute central logging system)

Central Logging – Result



```
host: RNGX5692 where: connection.start type: INFO time: 2019-11-19 09:53:25 api_key: vmps message: Connected to redis://127.0.0.1:6379//
@log_name: vmps_example_celery _id: SCIUg24BtiXXDyWKxmzk _type: _doc _index: fluentd-vmps_example_celery _score: 0

host: RNGX5692 where: mingle.sync type: INFO time: 2019-11-19 09:53:25 api_key: vmps message: mingle: searching for neighbors @log_name: vmps_example_celery
_id: SSIUg24BtiXXDyWKxmzk _type: _doc _index: fluentd-vmps_example_celery _score: 0

host: RNGX5692 where: mingle.sync type: INFO time: 2019-11-19 09:53:25 api_key: vmps message: mingle: all alone @log_name: vmps_example_celery
_id: SiIUg24BtiXXDyWKymz- _type: _doc _index: fluentd-vmps_example_celery _score: 0

host: RNGX5692 where: pidbox.loop type: INFO time: 2019-11-19 09:53:25 api_key: vmps message: pidbox: Connected to redis://127.0.0.1:6379//.
@log_name: vmps_example_celery _id: SyIUg24BtiXXDyWKymz- _type: _doc _index: fluentd-vmps_example_celery _score: 0

host: RNGX5692 where: worker.on_consumer_ready type: INFO time: 2019-11-19 09:53:25 api_key: vmps message: celery@RNGX5692 ready.
@log_name: vmps_example_celery _id: TCIUg24BtiXXDyWKymz- _type: _doc _index: fluentd-vmps_example_celery _score: 0

host: RNGX5692 where: connection.start type: INFO time: 2019-11-19 09:56:48 api_key: vmps_example_celery message: Connected to redis://127.0.0.1:6379//
@log_name: vmps_example_celery _id: _SIXg24BtiXXDyWK3Gx6 _type: _doc _index: fluentd-vmps_example_celery _score: 0

host: RNGX5692 where: mingle.sync type: INFO time: 2019-11-19 09:56:48 api_key: vmps_example_celery message: mingle: searching for neighbors
```

```
GET _search
{
  "query": {
    "match_all": {}
  }
}
```

Step 1 of 2: Define index pattern

Index pattern: fluentd-vmps_example_*

You can use a * as a wildcard in your index pattern.
You can't use spaces or the characters \, /, ?, ", <, >, |.

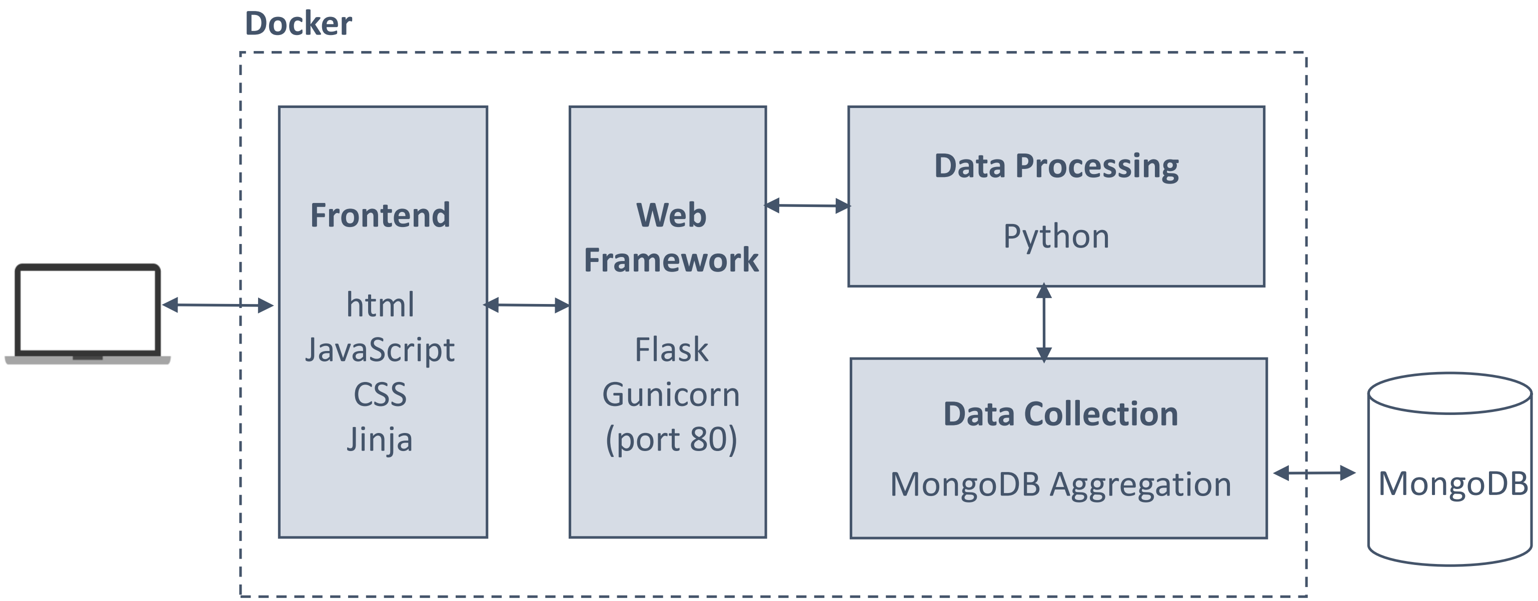
✓ Success! Your index pattern matches 3 indices.

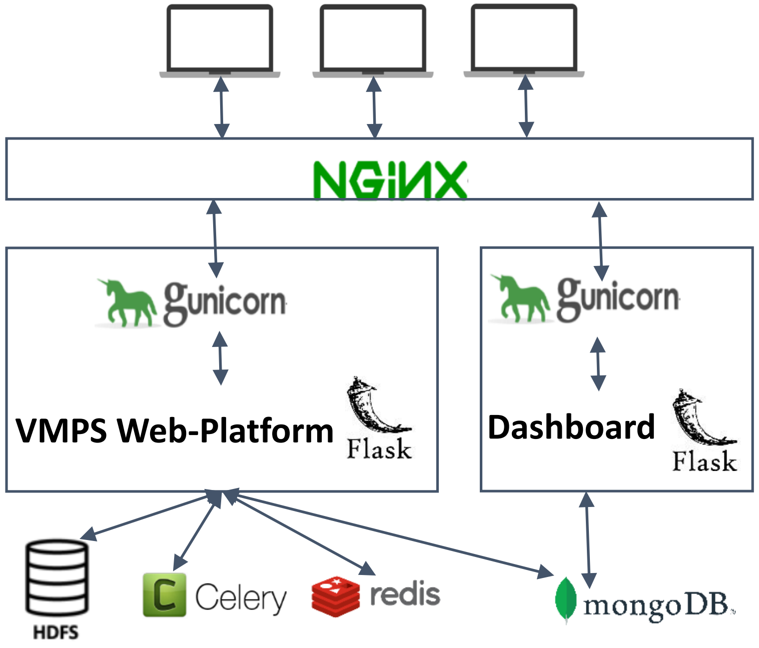
fluentd-vmps_example_celery
fluentd-vmps_example_flask
fluentd-vmps_example_python

VMPS Dashboard

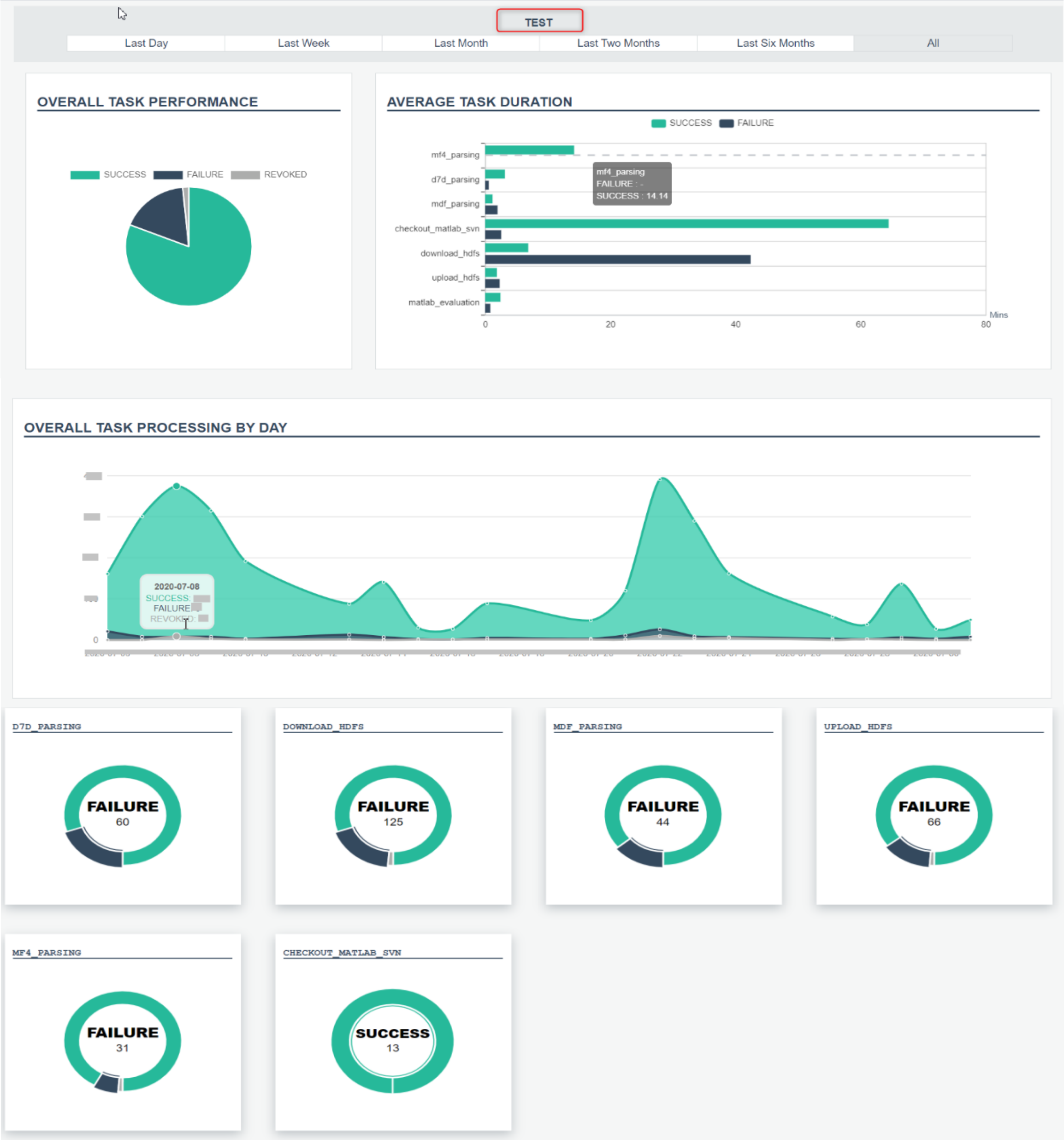
VMPS Dashboard

- **Flask:** It’s a lightweight , extensible web framework for building web applications with Python.
- **Gunicorn:** It’s a web server which enable a faster flask.
- **NGINX**: It’s an open-source web server and reverse proxy, which tries to distribute the requests across multiple servers or instances in a cluster. It aims to minimize the response time and maximize the throughput by avoiding the overload on any single resource.
- **MongoDB:** it’s like a tree-shape database to support faster query. Usually used to store data which wont change frequently like user account. Straight forward because company always organized in tree structure.
- **Jinja:** Template Inheritance; enable loops in html; consume data from Flask.





VMPS Dashboard - Demo



Thanks

BOSCH

Invented for life

Bosch-Group

Bosch Center for Artificial Intelligence