

# Internship Summary 2020

## Gong Na

# 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 colleagues)

- 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 vmpps 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

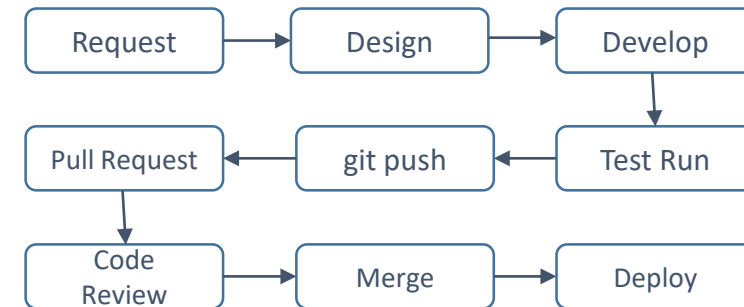
## Task 1: Central Logging

- 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):

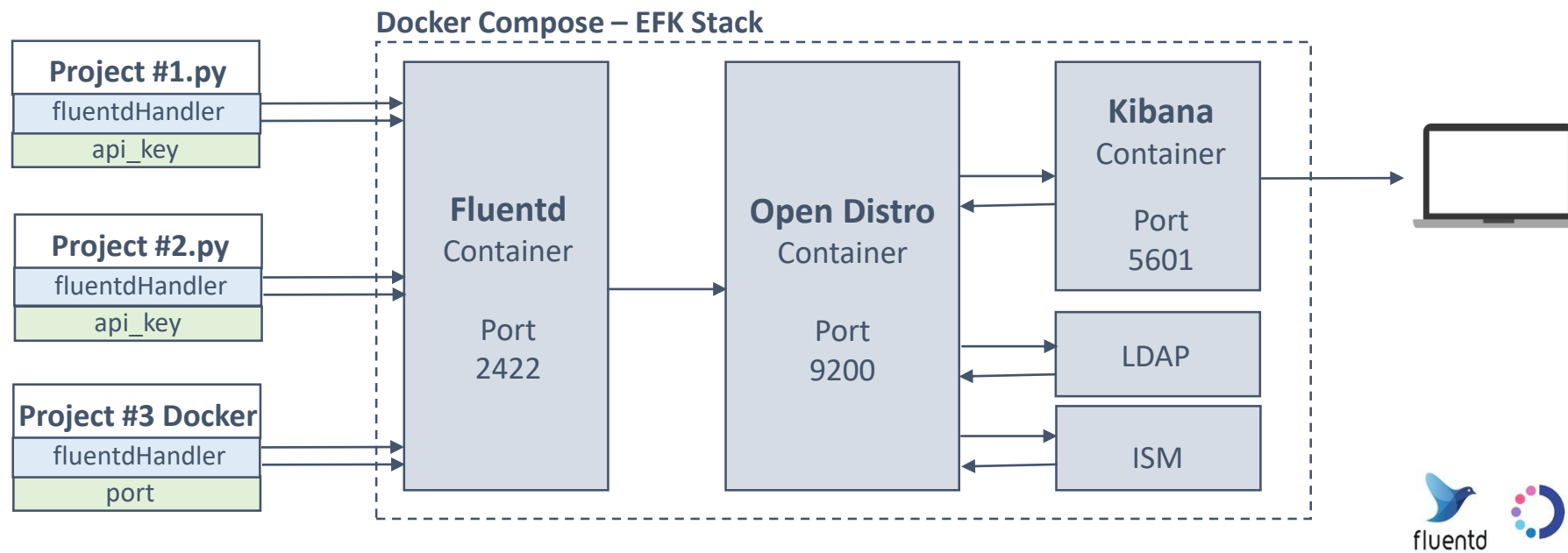




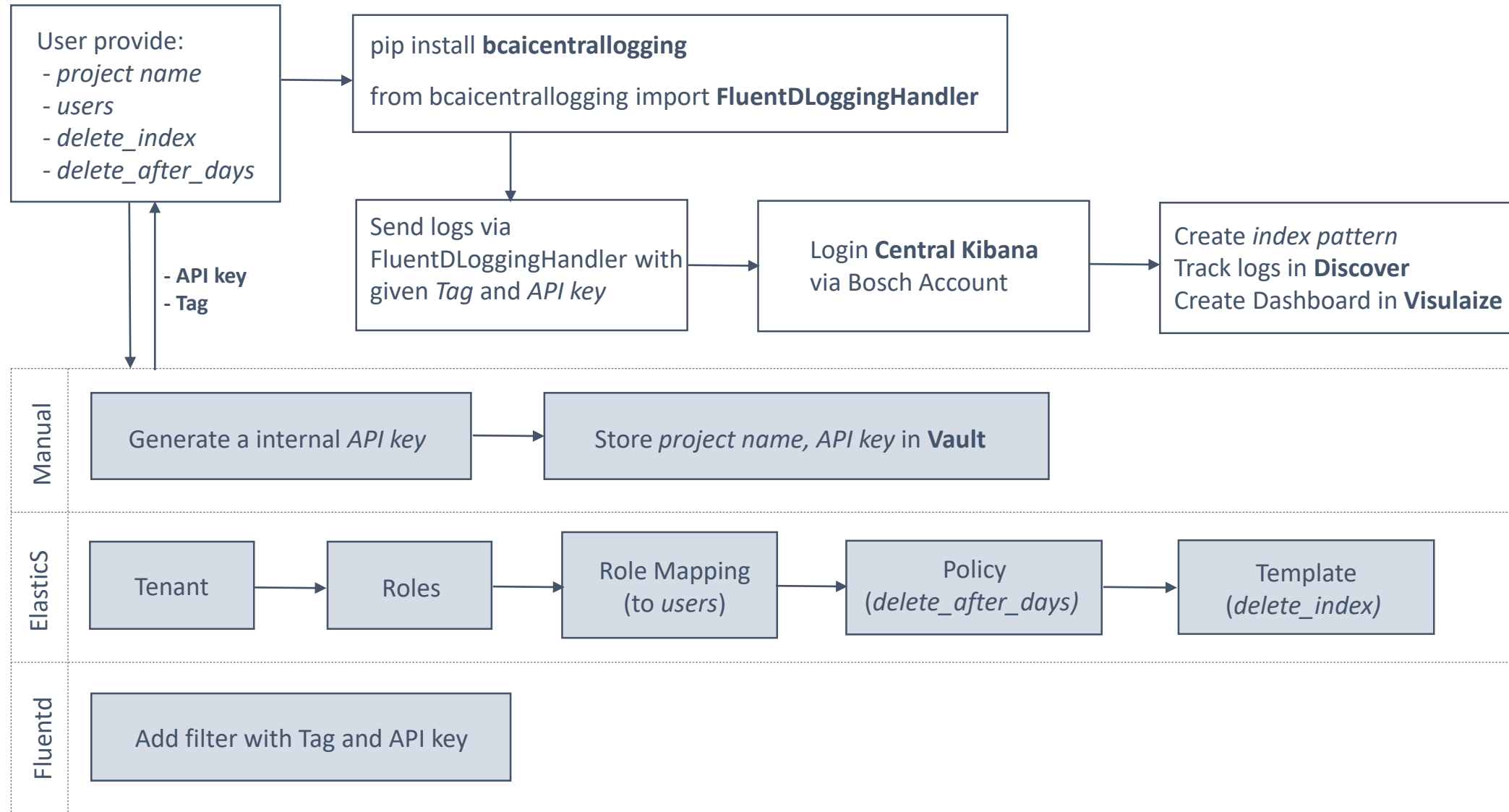
# Central Logging System

# 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





# 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 won't 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

The image displays the Kibana interface for Open Distro for Elasticsearch, showing search results, a login dialog, and index pattern configuration.

**Search Results:** The main panel shows 69 hits for the index pattern `fluentd-vmps_example_*`. The results are displayed in a table with columns for host, where, connection.start, type, INFO, time, api\_key, vmps, message, @log\_name, vmps\_example\_celery, \_id, SIUg24BtiXXDyWKxmzk, \_type, \_doc, \_index, fluentd-vmps\_example\_celery, and \_score. The results are filtered by the index pattern `fluentd-vmps_example_*`.

**Login Dialog:** A modal dialog titled "Please login to Kibana" is displayed, asking for a username and password. It includes a "Log in" button.

**Index Pattern Configuration:** The "Step 1 of 2: Define index pattern" dialog is shown. The index pattern is `fluentd-vmps_example_*`. The dialog indicates that the pattern matches 3 indices: `fluentd-vmps_example_celery`, `fluentd-vmps_example_flask`, and `fluentd-vmps_example_python`. The "Next step" button is highlighted.

**Permissions and Roles:** A sidebar menu shows "Permissions and Roles" with sub-items: Role Mappings, Roles, Action Groups, and Tenants.

**Authentication Backends:** A sidebar menu shows "Authentication Backends" with the option "Internal User Database".

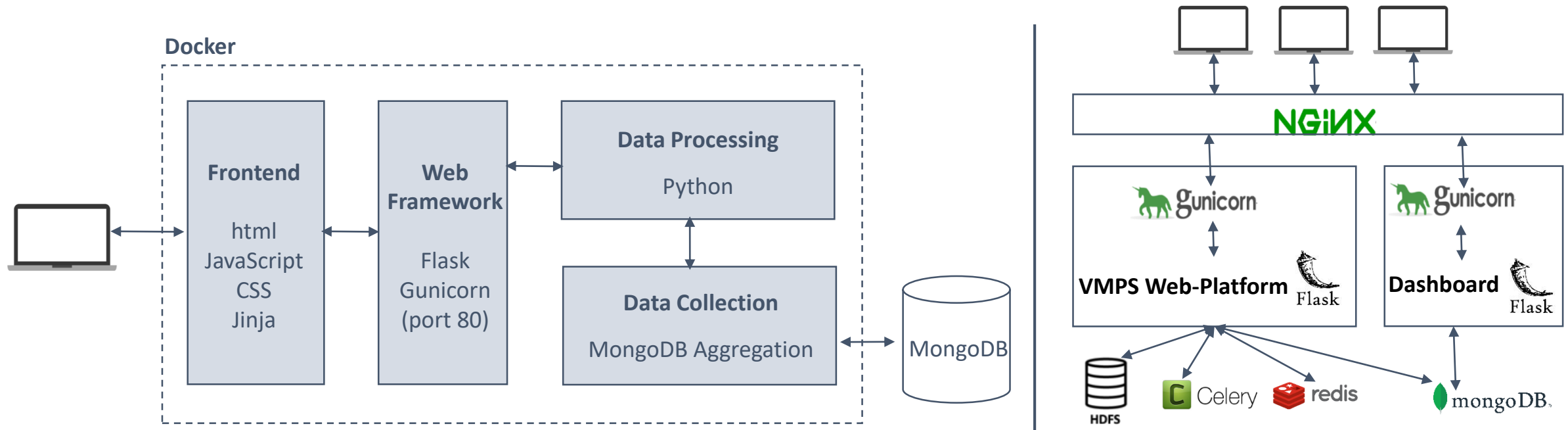
**Dev Tools:** The "Dev Tools" panel shows a console with a query: `GET _search { "query": { "match_all": {} } }`. The query is executed, and the results are displayed in the main panel.



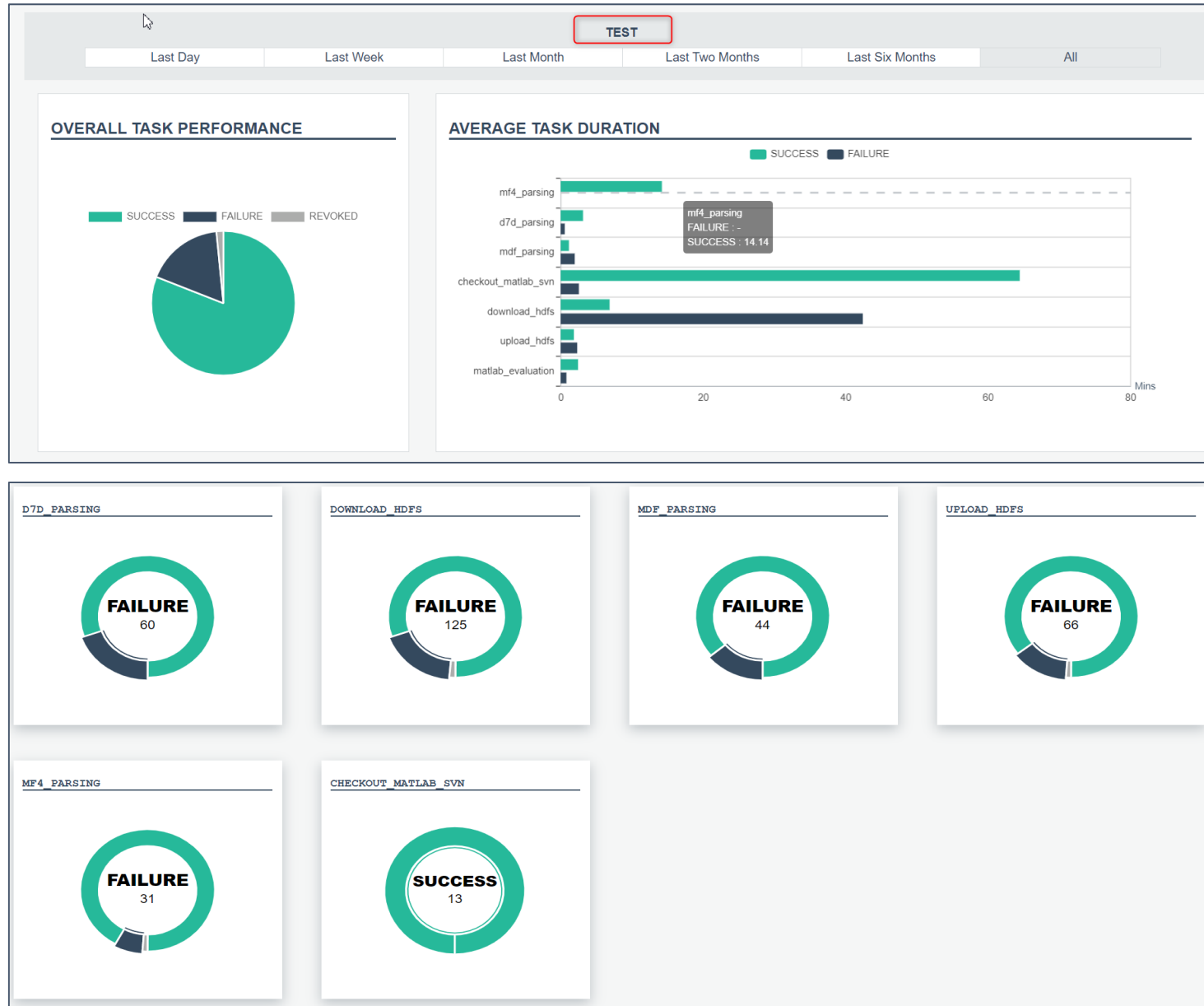
# VMPS Dashboard

# 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