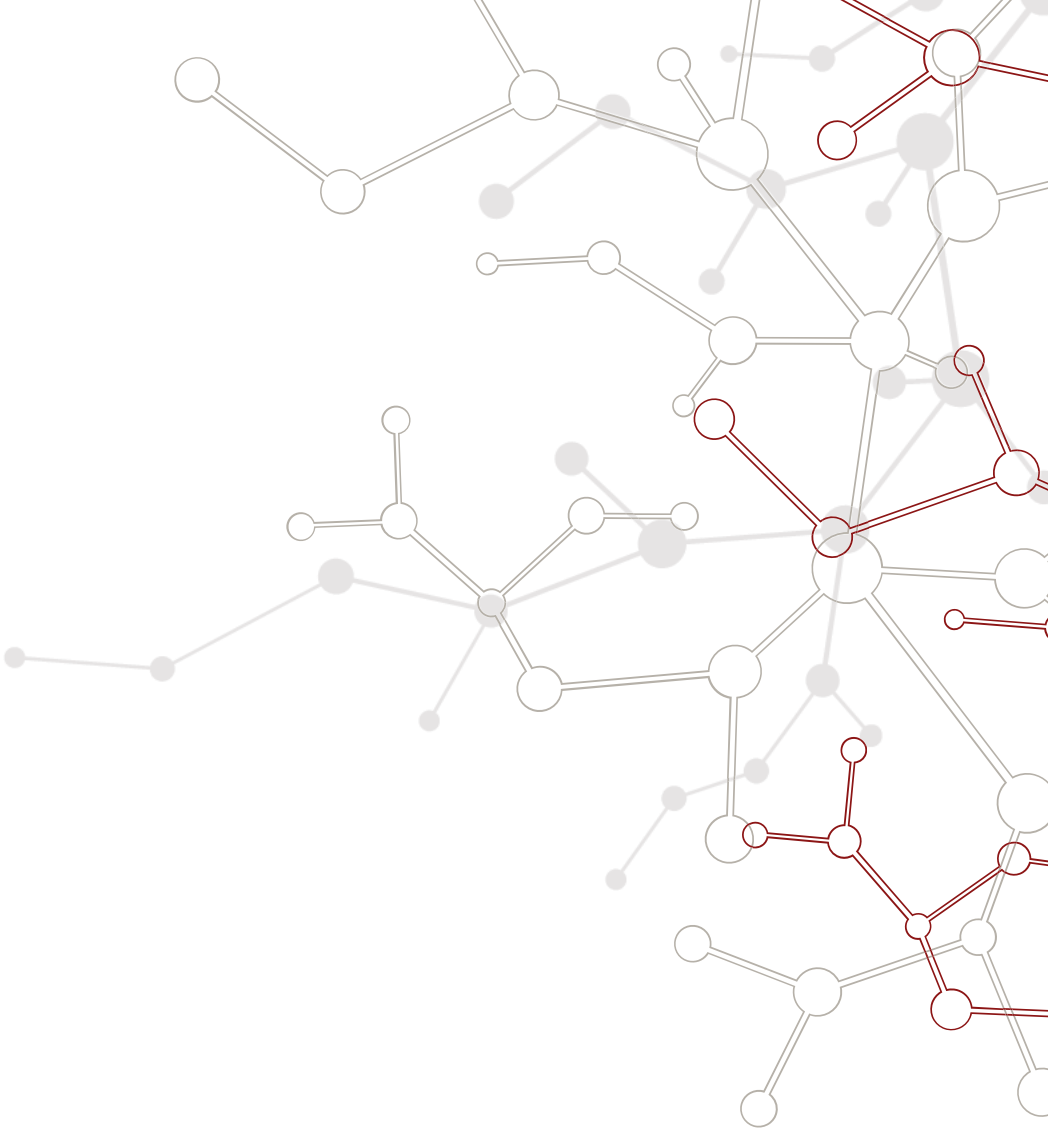
EED Software development

Claudio Bisegni / Dev Soni

Date May 25 2023



Agenda

- proof of concept for new ELOG application
- overview of the new web application architecture base on Kubernetes

ELOG Proof of Concept

POC Objective:

- try to merge the data into the same database for MCC and Physics elog
- create a modern unique Elogs application

Data sources:

MCC Elog data -> from oracle(MCCO)

Pysics Elog data -> from filesystem

What has been implemented:

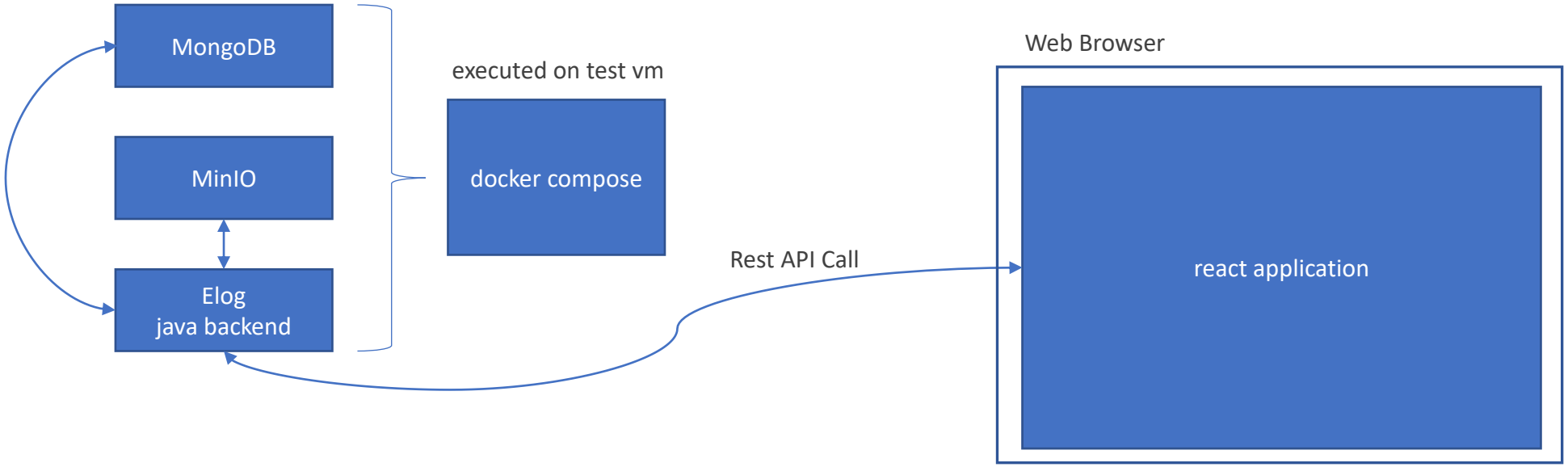
- MCC and Physic Log import procedure (only 01/2023 data)
- view log with pagination
- date and logbook filtering
- image visualization

To develop

ELOG Proof of Concept

Key features :

- mongodb permit to save data with different structure in the same collection using BSON (Binary JSON - https://bsonspec.org)
- MinIO, S3 compatible storage (https://min.io) used for the physics log images
- Java backend that expose REST API for search and manages the data
- React UI, use REST API of the backend for present the data to the user



To develop

ELOG Proof of Concept – MCC data structure

The exported record from Oracle have been converted to JSON before pushed to MongoDB


```
{
"ELOG_ID": 1188604,
"LOGDATE": "2023-01-27 18:49:40",
"PROGDATE": "2023-01-30 18:49:35",
"COMMITDATE": "2023-01-27 18:49:47",
"TITLE": "Attention: The Interlock Checklist for the BTH is IMCOMPLETE. Must be completed before leaving it in NO ACCESS and enable hazards.",
"PRIORITY": "VIP",
"TEXT_TYPE": "text/plain",
"TEXT_LENGTH": 128,
"TEXT": "The incomplete BTH Interlock Checklist is located in the Safety Assurance Forms to be Completed folder located on the EOIC desk.",
"OS_USERNAME": "apache",
"ENTRY_TYPE": "LOGENTRY",
"LOGBOOK": "MCC",
"LASTNAME": "Duong",
"FIRSTNAME": "Tri",
"USERNAME": "thduong2",
"USER_PRECEDENCE": 1,
"PROGRAM_ID": 152,
"PROGRAM_NAME": "Web Entry CGI",
"PROGRAM_CATEGORY": "USER",
"HOSTNAME": "mccelog",
"XML_FILENAME": "20230127_184940_104534.xml",
"SEGMENT_LIST": "LCLS_NC",
"SEGMENT": "LCLS_NC",
"SEGMENT_NAME": "LCLS NC Linac"
},
```

ELOG Proof of Concept – Physics data structure

Physics LOG are stored on a file system directory Structure

A valid “log” is composed by

- Xml file for description
- PNG preview image
- PS image

```
...
├── 03
│   ├── 16.01
│   │   └── init.xml
│   ├── 17.01
│   │   ├── 2023-01-17T14:45:06.png
│   │   └── init.xml
│   ├── 18.01
│   │   ├── 2023-01-18T15:21:42-00.png
│   │   ├── 2023-01-18T15:21:42-00.ps
│   │   ├── 2023-01-18T15:21:42-00.xml.BAK
│   │   ├── 2023-01-18T15:21:42-01.xml
│   │   ├── 2023-01-18T15:34:06-00.png
│   │   ├── 2023-01-18T15:34:06-00.ps
│   │   ├── 2023-01-18T15:34:06-00.xml
│   │   ├── 2023-01-18T15:39:26-00.png
│   │   ├── 2023-01-18T15:39:26-00.ps
│   │   ├── 2023-01-18T15:39:26-00.xml.BAK
│   │   ├── 2023-01-18T15:39:26-01.xml
│   │   ├── 2023-01-18T15:40:14-00.png
│   │   ├── 2023-01-18T15:40:14-00.ps
│   │   ├── 2023-01-18T15:40:14-00.xml
│   │   └── init.xml
...
```

Content from /03/18.01/ 2023-01-11T15:17:49-01.xml

xml content

```
<severity>NONE</severity>
<location>not set</location>
<keywords>none</keywords>
<time>15:34:06</time>
<isodate>2023-01-18</isodate>
<author>Matlab</author>
<category>USERLOG</category>
<title>Laser Heater Alignment</title>
<metainfo>2023-01-18T15:34:06-00.xml</metainfo>
<file>2023-01-18T15:34:06-00.png</file>
<link>2023-01-18T15:34:06-00.ps</link>
<text></text>
```

mapping on Elog Key

```
{
"LOGDATE": < isodate >,
"COMMITDATE": < isodate >,
"PROGDATE": < isodate >,
"OS_USERNAME": < author >,
"TITLE": < title >,
"LOGBOOK": "PHYSICS-LOG", #fixed for all entries
"ENTRY_TYPE": < category >,
"PRIORITY": < severity >,
"TEXT": < text >,
"SEGMENT": < location >,
"TAGS": < keywords >,
"FILE_PREVIEW": < file >,
"FILE_PS":< link >
}
```

conversion to JSON

```
{
"LOGDATE": “2023-01-18T15:34:06-00”,
"COMMITDATE": “2023-01-18T15:34:06-00”,
"PROGDATE": “2023-01-18T15:34:06-00”,
"OS_USERNAME": “Matlab”,
"TITLE": “Laser Heater Alignment”,
"LOGBOOK": "PHYSICS-LOG",
"ENTRY_TYPE": “USERLOG”,
"PRIORITY": “NONE”,
"TEXT": “”,
"SEGMENT": “not set”,
"TAGS": “none”,
"FILE_PREVIEW": “pelogs-data/03/18.01/2023-01-18T15:34:06-00.png”
"FILE_PS":” pelogs-data/03/18.01/2023-01-18T15:34:06-00.ps”
}
```

ELOG Proof of Concept – MCC vs Physics log structure

- The use of MongoDB simplify the management of documents with heterogeneous set of key/value elements
- MCC-Elog data structure has driven the mapping of Physics Elog data import


```
{
"ELOG_ID": 1188604,
"LOGDATE": "2023-01-27 18:49:40",
"PROGDATE": "2023-01-30 18:49:35",
"COMMITDATE": "2023-01-27 18:49:47",
"TITLE": "Attention: The Interlock Checklist for the BTH is IMCOMPLETE. Must be completed before leaving it in NO ACCESS and enable hazards.",
"PRIORITY": "VIP",
"TEXT_TYPE": "text/plain",
"TEXT_LENGTH": 128,
"TEXT": "The incomplete BTH Interlock Checklist is located in the Safety Assurance Forms to be Completed folder located on the EOIC desk.",
"OS_USERNAME": "apache",
"ENTRY_TYPE": "LOGENTRY",
"LOGBOOK": "MCC",
"LASTNAME": "Duong",
"FIRSTNAME": "Tri",
"USERNAME": "thduong2",
"USER_PRECEDENCE": 1,
"PROGRAM_ID": 152,
"PROGRAM_NAME": "Web Entry CGI",
"PROGRAM_CATEGORY": "USER",
"HOSTNAME": "mccelog",
"XML_FILENAME": "20230127_184940_104534.xml",
"SEGMENT_LIST": "LCLS_NC",
"SEGMENT": "LCLS_NC",
"SEGMENT_NAME": "LCLS NC Linac"
},
```



```
{
"LOGDATE": “2023-01-18T15:34:06-00”,
"COMMITDATE": “2023-01-18T15:34:06-00”,
"PROGDATE": “2023-01-18T15:34:06-00”,
"OS_USERNAME": “Matlab”,
"TITLE": “Laser Heater Alignment”,
"LOGBOOK": "PHYSICS-LOG",
"ENTRY_TYPE": “USERLOG”,
"PRIORITY": “NONE”,
"TEXT": “”,
"SEGMENT": “not set”,
"TAGS": “none”,
"FILE_PREVIEW": “pelogs-data/03/18.01/2023-01-18T15:34:06-00.png”
"FILE_PS":” pelogs-data/03/18.01/2023-01-18T15:34:06-00.ps”
}
```

Proof of concept architecture

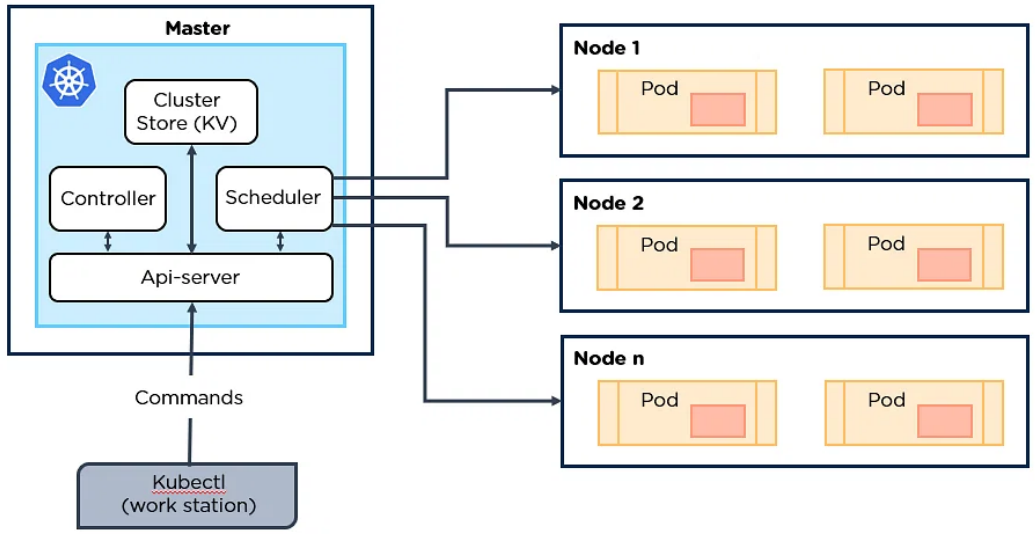
Demo

To develop

Overview of new application Kubernetes architecture

Based on Kubernetes

(an open-source system for automating deployment, scaling, and management of containerized applications)



Key Features

- helps run and organize lots of docker containers, on many computers.
- handles the distribution of workload across different machines in the cluster, automatically scaling the application based on demand.
- monitors the health of containers and restarts them if they fail. Additionally, Kubernetes enables easy updates and rollbacks of applications without any downtime.
- Automatically management of:
  - network
  - http load balancer
  - volume claims
  - scale management

To develop

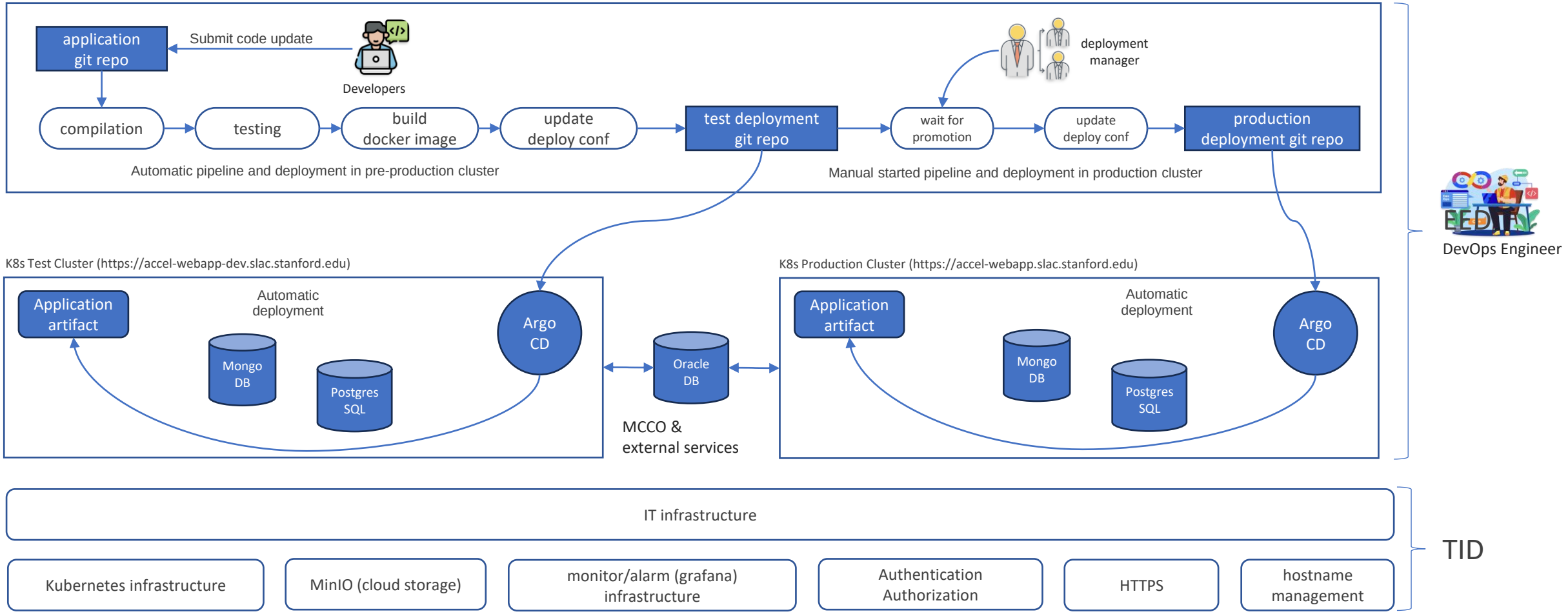
Overview of new application Kubernetes architecture

What's provided

- MongoDB cluster(https://mongodb.com), highly scalable document database. (done)
- Postgres cluster (done)
  - but if needed also MariaDB can be installed.
- MinIO (https://min.io, Amazon S3 compatible object storage) a distributed modern storage for the application's managed data. (from TID)
- ArgoCD (https://argoproj.github.io), tool for a Declarative Continuous Delivery for Kubernetes. (installed by TID help)
  - ensures that the deployed applications match the desired state defined in the Git repository.
  - If any differences are found, it reconciles them by applying the necessary changes to bring the actual state in line with the desired state.
- documentation: https://confluence.slac.stanford.edu/display/EEDWAD/EED-Web+Application+Development+Home (TODO)
- template based on 'copier': (java example) https://github.com/slaclab/eed-java-backend-template, (Python and React -> TODO)
- general pipeline for build software, test software, build docker image, change deploy configuration (Python and React -> TODO, java-> almost DONE)
- Production (TODO) and Test environment (DONE)
  - test environment hostname: https://accel-webapp-dev.slac.stanford.edu/xxxx

Best practices from GitOps approach

uses Git to manage and automate the deployment of applications. It treats infrastructure and application configurations as code stored in a Git repository. Changes made in Git trigger automatic updates to the target environment, ensuring that the desired state matches the actual state. It brings simplicity, traceability, and automation to the deployment process.



- ArgoCD (**Declarative Continuous Delivery for Kubernetes**) https://github.com/argoproj/argo-cd