# Accelerating HEP Workloads on Kubernetes

## AUGUST 2021

**AUTHOR(S):**
JASON PRAFUL FRANCIS XAVIER

IT - ML

**SUPERVISOR(S):**
RICARDO ROCHA
DEJAN GOLUBOVIC

CERN openlab

# PROJECT SPECIFICATION

## Motivation

The ability to learn and identify patterns especially when dealing with large datasets has always been difficult for humans to process, with the aid of modern technology this has now been simplified. With the help of machine learning (ML), humans can now parse, identify patterns and deal with large datasets in a short span of time. The importance of machine learning has only grown since then – for instance, ML is used on a daily basis to prevent fraud when transacting over the internet, prevent malware, Natural Language Processing and at CERN we can see the use of ML in Particle Tracking. Over tens of petabytes of data is processed at CERN every year and in order to identify patterns and sift through such a large amount of data, ML comes in handy.

At CERN we have access to tremendous computing power across the institution, however, it is not always obvious for users to find a service fulfilling their needs. In this case when a user would want to find a service to train or deploy their ML model they might be overwhelmed with a wide variety of choices. By centralising the entire process of training large datasets, building pipelines, deploy models and much more the user now has access to an entire ML infrastructure under one portal.This central shared pool also helps improve the overall usage and efficiency of scarce resources such as GPUS rather than having each group owning their own hardware. That way the user not only saves time sifting through multiple services and finding the one to meet their needs but they also save time building their own local infrastructure directly giving them more time to continue with their research.

## Solution

The new ML Service designed and built by CERN is still in its early beta. This easy to use service caters the user an all in one ML platform where they can not only train and deploy their models but also make use of pipelines to automate their ML tasks, and with direct public cloud integration, users will be able to make use of the tremendous amounts of GPU, make use of specialized accelerators like TPUs and FPGAs, and computing power otherwise not possible on-premise.

## Technical

Given that Kubeflow is directly built on top of Kubernetes; the service allows all applications to be containerised which on a large scale provides a tremendous advantage. Kubernetes allows easy deployment of all services in separate pods which can then directly make use of the computing and GPU power available at different nodes. The process of containerisation also allows easy building, termination and deployment of clusters, pods, services with ease.

The advantage of having the entire framework being open-sourced also meant that the tool can be deployed on an on-premise in this case the CERN Private Cloud. CERN Private Cloud hosts Openstack – this cloud computing platform makes it, even more, easier to gain access to bare metal hardware and deploy VMs upon which the Kubernetes service can be run.

The summer internship role was to work on the ML service built on top of Kubernetes and Kubeflow, test all the examples and docs and provide feedback. At the same time provide assistance to users accessing the service. The role also involved implementing additional features and functionalities to the service such as serving multiple models for inference on a single GPU, working on additional libraries and features and integrating them into the system to help aid users.

# ABSTRACT

Deploying a new open-source based service is an extensive and challenging process, from both technical and user-support perspective. The work includes understanding the service architecture and features, deploying the service on local servers, debugging errors and customizing the open-source code to fit the specific requirements of the users' ecosystem.

Integration of the new machine learning service based on Kubeflow is an ongoing process to offer a better user experience for machine learning developers across CERN. Providing on-demand access to Python environments and hardware resources such as GPUs, the new service reduces users' need to set up local infrastructures and allows more time for scientific research. Additionally, the service offers features such as pipelines, automated hyperparameter search, inference services, that can be utilized for developing complex machine learning use cases which go beyond model training.

# Table of Contents

## 1. OBJECTIVE

With petabytes of data flowing through daily at CERN, understanding, filtering, and performing machine learning analysis on this data is vital. As currently there is no centralized place at CERN for providing ML infrastructure and services, and following CERN's commitment to using open-source solutions, the objective of the project is to deploy an open-source based machine learning service. The service is hosted on CERN on-premise servers, making use of the available computing power.

Providing underlying technologies such as Openstack, Kubernetes, CSI, and others, CERN private cloud lays the groundwork for deploying Kubeflow, an open-source machine learning framework. The integration of Kubeflow is an ongoing process done by the CERN IT-CM-RPS group.

The objective of the summer internship was to cooperate in the service integration by testing and incorporating additional features, providing user support, and working on the service upgrade. During the internship, multiple features were explored and tested, such as multiple models serving on a single GPU, Nvidia multi-instance GPU partitioning, Feast feature store, and the Goofys filesystem. Additional implemented feature was exposing virtual GPUs to the users. The result of the work during the internship is the enhanced service, with more options for users' machine learning workloads.

## 2. PROJECT PLANNING

An agile approach was followed throughout the process of the Openlab internship. Jira boards were used to plan, delegate, report, and keep an entire record of all the portions and processes during the project. Biweekly sprint meetings were utilised efficiently to monitor and keep track of the open tickets in Jira. These calls were also used to keep the team updated on the progress as well as ask questions.

- Week 1 – Onboarding and testing the instance
- Week 2 – Providing feedback on examples and testing all the instances and pipelines
- Week 3 – Exploring multi instance GPUs (MIG)
- Week 4 – Exploring FEAST library and building an example set
- Week 5 – Integrating and exposing virtual GPUs to frontend
- Week 5-6 – Integrating and providing examples for Goofys S3 filesystem
- Week 6-7 – Creating Merge requests with all updates, comments and bug fixes where needed
- Week 7-8 – Providing user support and testing the production instance for improvements.

| T | Key | Summary | Assignee | Reporter | P | Status | Resolution | Created | Updated ↓ | Due | Development | Σ Progress |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | OS-14395 | ml.cern.ch Down (x509 Error) | Jason Prafulchandra Francis Xavier | Jason Prafulchandra Francis Xavier | | CLOSED | Fixed | 02/Aug/21 | 02/Aug/21 | | | |
| | OS-12708 | expose gpu/vgpu availability on notebook creation form | Jason Prafulchandra Francis Xavier | Ricardo Rocha | | IN REVIEW | Unresolved | 27/Nov/20 | 02/Aug/21 | | | |
| | OS-14373 | Write a report about the internship | Jason Prafulchandra Francis Xavier | Dejan Golubovic | | IN PROGRESS | Unresolved | 23/Jul/21 | 02/Aug/21 | | | |
| | OS-14374 | Prepare a presentation for the summer student event | Jason Prafulchandra Francis Xavier | Dejan Golubovic | | IN PROGRESS | Unresolved | 23/Jul/21 | 02/Aug/21 | | | |
| | OS-14140 | Investigate NVIDIA Triton Inference Server | Jason Prafulchandra Francis Xavier | Dejan Golubovic | | IN PROGRESS | Unresolved | 11/Jun/21 | 29/Jul/21 | | | |
| | OS-14354 | try out goofys to mount s3 buckets in ml notebooks | Jason Prafulchandra Francis Xavier | Ricardo Rocha | | IN REVIEW | Unresolved | 15/Jul/21 | 29/Jul/21 | | | |
| | OS-14283 | drop data volumes from the notebook creation page | Jason Prafulchandra Francis Xavier | Ricardo Rocha | | IN REVIEW | Unresolved | 05/Jul/21 | 29/Jul/21 | | | |
| | OS-13922 | add vgpu selector to kubeflow notebooks | Jason Prafulchandra Francis Xavier | Ricardo Rocha | | IN REVIEW | Unresolved | 17/May/21 | 29/Jul/21 | | | |
| | OS-14278 | Multi instance GPUs with Nvidia A100 and Kubernetes | Jason Prafulchandra Francis Xavier | Jason Prafulchandra Francis Xavier | | OPEN | Unresolved | 04/Jul/21 | 29/Jul/21 | | | |
| | OS-14277 | Issue with ML Service Certificates (X509 Bug) | Jason Prafulchandra | Jason Prafulchandra | | CLOSED | Fixed | 02/Jul/21 | 13/Jul/21 | | | |

## 3. TOOLS

- Reverse SSH Proxy (for connecting to CERN Internal Network)
- Kubernetes
- Git (for collaboration, MR, pipelines)
- Kubeflow + Kale, Katib (on Kubeflow for pipelines)
- Jupyter Notebooks
- Docker
- Angular.js
- Typescript, Python
- GKE (Google Kubernetes Engine)

## 4. METHODOLOGIES

### a. Remote Access

While developing for this project, multiple different tools were utilised to achieve success. With the era of work from home, more focus was laid on connecting to the resources remotely. CERN allows users to remote tunnel into its servers via an SSH proxy via lxplus. This connection was utilised throughout especially when interacting and deploying projects on the internal network.

### b. Kubernetes

For the majority of the project major focus was laid on working with the Kubernetes cluster – both on-prem and on GKE (Google Kubernetes Engine). In Kubernetes, more focus was laid on configuring the system to handle MIGs, Kubeflow Pipelines, Multi-Model Serving and a complete end to end pipeline to rebuild the architecture.

### c. Kubeflow

Kubeflow was the framework that was worked on for almost the entire course of the project. The process involved upgrading the existing instance, Rebuilding the frontend to show any requirements in the JIRA tickets assigned. Testing and debugging pipelines. Providing user support.

### d. DevOps

Most portions involved refactoring or adding additional code to the repository. This was key, especially when collaborating with the team on the same project. Ensuring Merge Requests were submitted with well-detailed documentation, clean commits and examples to help aid and ease the development process. Pipelines to merge new commits to either the dev environment or production environment. For instance, creating and pushing an entire docker image to a registry.

# 5. FRAMEWORK AND PROCESSES

## a. Exploring and Providing Feedback for Kubeflow Examples

During the initial phase of the project, most of the time was invested to learn and test the Kubeflow framework. Kubeflow is an open-sourced machine learning framework built on top of Kubernetes to allow users to easily build, deploy and scale ML frameworks. Kubeflow offers a plethora of features such as allowing users to run their own notebook servers, run ML pipelines to build, train and publish their models as well as make use of Kubernetes to scale their ML application.

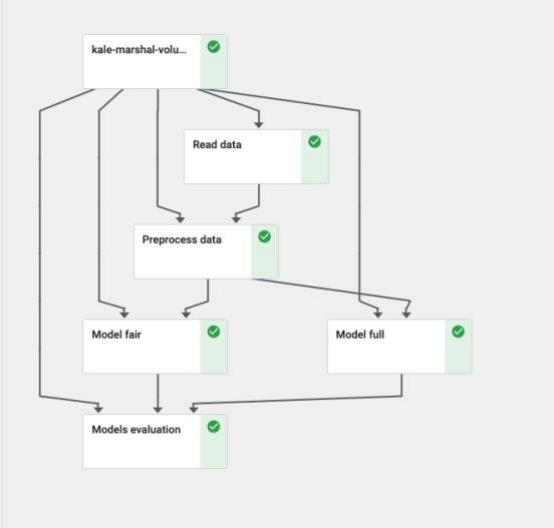


*Figure 1 - Example of an ML Pipeline*

The goal at this stage was to test all the examples and provide feedback as well as potential fixes to the examples available. The goal of these examples is to allow any user new to this framework to familiarize themselves with the platform quickly. These examples included steps for CERN EOS integration, Argo workflows, making use of Kubeflow fairing to build and deploy ML models, learn how to create Kubeflow pipelines (KALE) directly from notebook servers, making use of S3 or EOS for storing models. These are simply some of the many examples provided to users to get them on board with the platform.

While investigating and testing the examples there were a few bugs that were encountered. Changes were then made to the examples to mitigate these bugs. Appropriate merge requests were made upon fixing these bugs as well as adding more examples where needed.

This phase allowed easy adaption to the platform and find portions for potential improvement.

## b. Multi Instance GPUs

JIRA TICKET

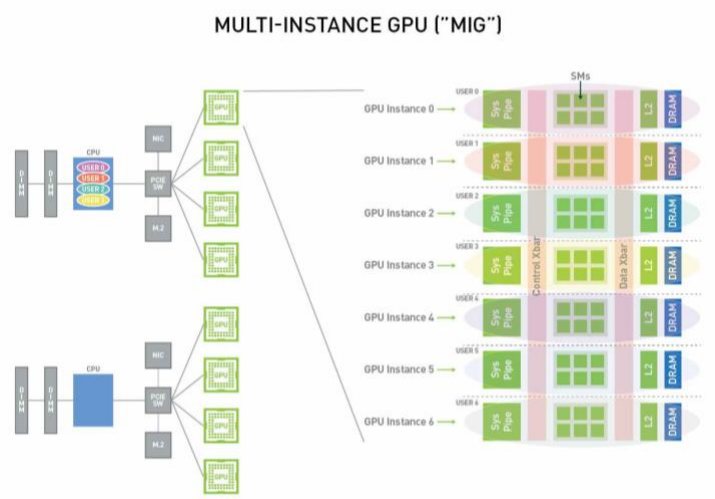


*Figure 2 - MIG Architecture*

One of the disadvantages of the Kubernetes setup was allocating an entire GPU to a single user. This process not only made inefficient use of the available resources, but it also meant that at any given point an entire GPU will be allocated to a user without allowing anyone else to utilise it. The user may or may not use the GPU, regardless, the GPU would remain fixed leaving potential users who may want the resource helpless.

Therefore, the goal of this phase of the project was to explore the possibilities to allow Multi Instance GPUs (MIG). This allows a single GPU to be repartitioned and to be used by multiple services (notebooks, pipelines, distributed training jobs), rather than allocating a single full GPU to a user. This also means that a GPU would be returned to the pool of resources, after the user's task completes.

With this goal in mind, the next steps were to identify the process involved in allowing MIG. Nvidia provides support for MIG on Kubernetes, however only the A100 series supports this feature at the moment. The on-prem Kubernetes cluster hosted V100s and T4s GPUs, which meant that this phase of the project had to be hosted on a cloud environment for development and testing.

To serve multiple instances, the Kubernetes layer had to be separated from the production Kubeflow instance, virtually creating more nodes to allow the users to connect and use.

Nvidia allows this feature on top of Kubernetes with the help of their `k8s-device-plugin` and the `GPU-feature-discovery`. This feature allows multiple users to run GPU workloads concurrently with isolation on a hardware level rather than doing it over software.

The user would now be able to specify the slice and memory requirements of a single A100 GPU instead of taking over an entire GPU.

For example – a single A100-SXM4-40GB GPU can be configured into one of the following configurations

- `2 x 3g.20gb` (20gb x 2 for 2 instances on 1gpu) (3 compute instances, 20gb each)
- `3 x 2g.10gb` (10gb x 3 for 3 instances on 1gpu) (2 compute instances, 10gb each)
- `7 x 1g.5gb` (5gb x 7 for 7 instances on 1gpu) (1 compute instance, 5gb each)

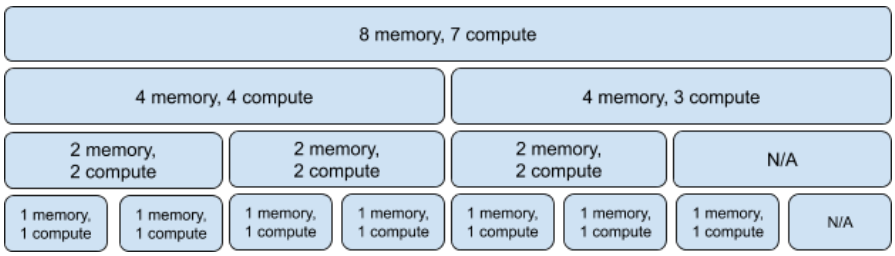


*Figure 3 - Configuration/Split*

With the legacy method a user would request for GPU in the Kubernetes script as – `nvidia.com/gpu: 1`

With MIG, a user can specify the compute requirement and memory allocation and get a fragment of a single GPU instance. This new system would now have the following limit to request GPU resources –

```
nvidia.com/mig-3g.20gb: 1
```

The hurdle at this phase of the project was to have the latest version of Kubernetes along with the latest version of `nvidia-docker2`.

## c. Multiple Model Serving Using a Single GPU

JIRA TICKET

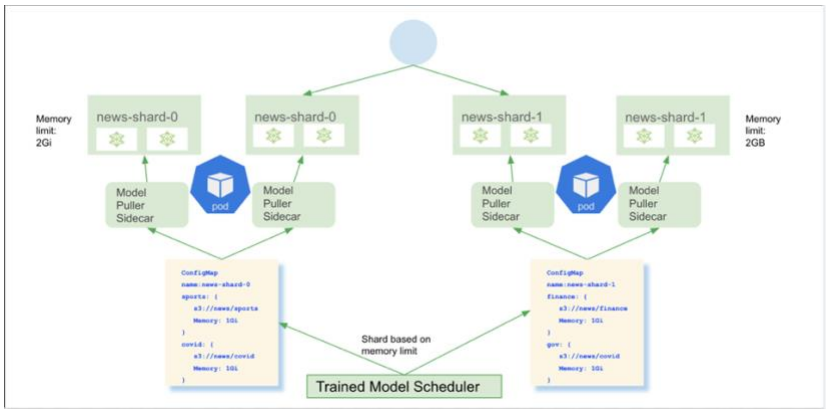


*Figure 4 - Multi-Model Serving Diagram*

The option to split a single GPU and allow it the sections to be used by multiple services is beneficial for using GPUs to run intensive tasks such as model training via notebooks, pipelines, or Katib. When a GPU is used to serve a trained model, there are additional steps to ensure the efficiency. As in the previous sections, it is important to reduce the idle time of GPUs.

Model serving represents making a trained model available to the users and other software components. Kubeflow enables model serving via REST API. When a model serving pod receives a request, it generates predictions by querying a stored model, using a GPU.

Kubeflow additionally facilitates hosting multiple models on a single GPU. With this, a user can make use of a single triton inference server to dynamically serve multiple models. This also means that users don't need to manually split the GPUs and keep track of the memory.

The process is fully dynamic. With KFServing v0.6+ this process is inbuilt and can be enabled by just changing the **`multimodelserver`** flag to **`true`** in the `inferenceservice.yaml` file. The final flag is the number of dependencies that need to be met to enable and install **`KFServing v0.6+.`**

Dependencies included and were not limited to:

Upgrade **`Kubernetes`** to **`v1.16+`**

Upgrade **`Istio`** to **`v1.9.0+`**

Upgrade **`kNative-Serving`** to **`v0.17.4+`**

The production instance currently runs on an older version of Kubeflow and Kubernetes and thereby does not allow enabling the **multimodelserving** feature.

## d. Exploring and Integrating the Feast Feature Store Library

JIRA TICKET

Feast (**Fea**ture **St**ore) is an operational data system for managing and serving machine learning features to models in production. Feast is able to serve feature data to models from a low-latency online store (for real-time prediction) or from an offline store (for scale-out batch scoring or model training). The legacy process of developing features is slow when multiple users/services access datasets at the same time. Furthermore, the features need to be redeveloped multiple times before training and serving. This poses challenges for teams with large number of developers. During this process, inconsistencies arrive especially when data

scientists and engineers need to work together to provide one output. This structure also doesn't allow easy addition of new features and feature reuse.

With Feast the entire process is streamlined. Feast creates and breaks down features that allow teams to work independently and combine the data in a real-time production environment. This process now allows reuse of artifacts between teams and provides a central registry where the teams can collaborate and make use of the catalogue of features available.

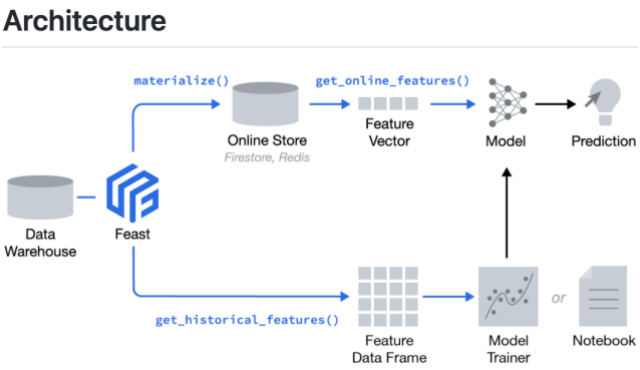


*Figure 5 - FEAST Architecture*

Given the fact that Kubeflow runs on Kubernetes, this makes the entire process even easier as Feast has native support on top of Kubernetes via helm. This, however, wasn't made use of during the course of this project, rather the modules were installed standalone on the notebook server and a central registry was used to store the data.

The main goal of this phase of the project was to provide the users with an example and a guide for using Feast library. Appropriate merge requests were submitted post-completion.

### e. Exposing Virtual GPUs

JIRA TICKET

Virtual GPUs are created by partitioning a single GPU in such a way that a single GPU makes use of time-sharing to cater resources to multiple users. Virtual GPUs provide significant acceleration when compared to CPUs, and increase the quantity of resources in the cluster. The work on this task included implementation of exposing the vGPUs to the users during the notebook server creation.

To implement this functionality there had to be some refactoring done to both the backend and frontend. On the frontend, an additional option for selecting vGPUs was added to a notebook server creation form. This option is used to propagate information to the backend, which allocates the specified resource.

The backend codebase is written in **`Python`** while the frontend is written in **`Angular`** based on **`Typescript`**.

With the help of this refactor the user would now be able to request for vGPUS and the server would then dynamically allocate based on the profile field. Appropriate merge requests consisting of the examples were submitted post-completion.

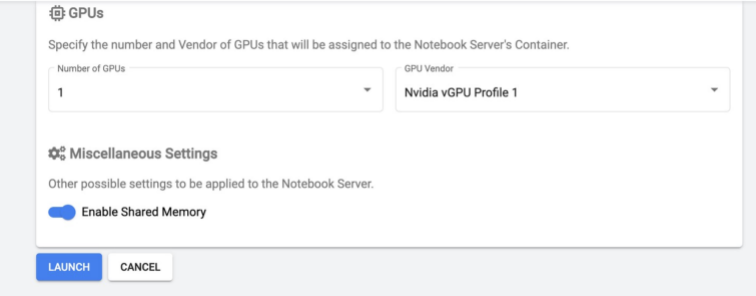


*Figure 6 - vGPU Selector Frontend*

Request payload from frontend to server:


*Figure 7 - vGPU Sample Backend Request*

## f. Mounting S3 Buckets to Notebooks Using Goofys File System

JIRA TICKET

Goofy's is a high-performance Amazon S3 file system library that allows mounting an S3 bucket as a file system. Goofys mounts an S3 bucket as a local filesystem with file structure enabling any user to navigate through with ease.

This is a huge benefit when integrated with the ML service as now any user can mount S3 buckets directly to the notebook just as a local file system. The performance benefit obtained makes the entire experience seamless.

The goal was to test the library and create examples to enable any user to swiftly learn and start using his/her S3 bucket with the notebook environment.

The documentation provided with the library made the entire process swift and easy to adapt and integrate. Appropriate merge requests consisting of the examples were submitted post-completion.

Performance Benefit

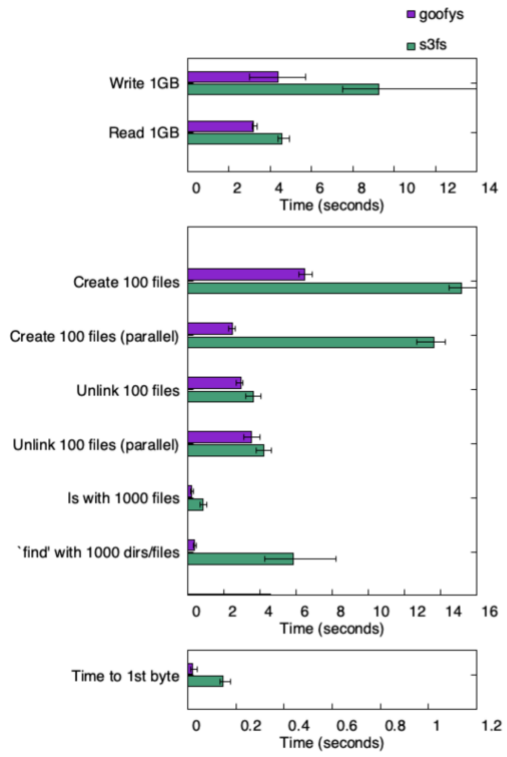


*Figure 8 - Performance Benefit Chart of Goofys*

### g. User Support

During the early beta of this project, there were multiple engineers and users from CERN utilising the service. Users were provided with consistent support on an everyday basis where most of their technical hurdles encountered on the course of using the service were addressed.

Alongside, there were instances where support was provided to debug and fix the problems. Problems such as GPU unavailability, trouble connecting to EOS buckets and version compatibility were addressed. Docker images were then created in line with the users' requirements, documented instructions were sent to the users to help fix the problem faced.

## 6. CONCLUSION

### a. Future Work and Improvements

Given the improvements and the impact the project has, it would bring in a tremendous number of users onboard in a short period of time. We have identified a few areas where improvements can be made in order to provide a better experience for the machine learning developers across CERN.

The continuation of the work during the internship will include the automation of GPUs allocation. This dynamic allocation will provide a better utilization of GPUs, allocating them only for the actual tasks.

Additionally, there is a work in progress to upgrade from `Kubeflow v1.1` to `Kubeflow v1.3.` This upgrade brings in new features such as multi model serving, enriched UI and VSCode notebook servers.

## b. Acknowledgements

The past few weeks have been extremely productive for me as a developer and I have obtained various skills and technical expertise in data handling, expanding my knowledge and hands-on experience with Kubernetes, Kubeflow, Google Kubernetes Engine and CERN's Open-stack service.

I am grateful to CERN for providing me with this opportunity to be a part of an extremely talented team. I had a great time engaging with other interns from across the world. The Summer Open Labs internship is an excellent initiative to work on real-life projects and I feel proud to have contributed my part in making a difference. Additionally, the lectures organised by the team discussing Quantum computing, High energy physics, Filtering Data, etc. have got me curious into knowing more about the advancements at CERN and I look forward to working at the institution in future. I thank my supervisor Ricardo Rocha and Dejan Golubovic, for the support throughout the internship without whom my learning wouldn't have been as effective.

Finally, I wish all the best to the team at CERN for their future endeavours.

# 7. REFERENCES

1. https://kccnceu2021.sched.com/event/iE1w/building-and-managing-a-centralized-ml-platform-with-kubeflow-at-cern-ricardo-rocha-dejan-golubovic-cern?iframe=no&w=100%&sidebar=yes&bg=no
2. https://indico.cern.ch/event/924283/contributions/4105328/
3. https://indico.cern.ch/event/948465/contributions/4323970/
4. https://github.com/kubeflow/kfserving/blob/master/docs/MULTIMODELSERVING_GUIDE.md#integration-with-model-servers
5. https://developer.nvidia.com/blog/getting-kubernetes-ready-for-the-a100-gpu-with-multi-instance-gpu/
6. https://docs.nvidia.com/datacenter/cloud-native/kubernetes/mig-k8s.html
7. https://github.com/feast-dev/feast
8. https://github.com/kahing/goofys