ML Easier at CERN with Kubeflow

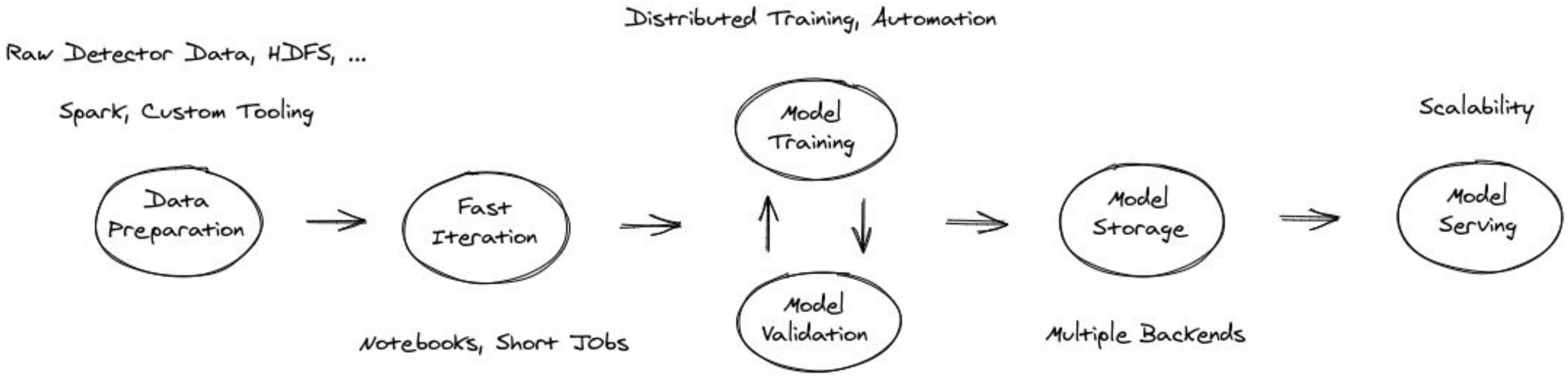
Preparation, Training and Model Serving

Dejan Golubovic, Ricardo Rocha
CERN IT-PW-PI

Contents

Motivation

Offer a platform to manage the full machine learning lifecycle



Motivation

Ensure **efficient usage** of our

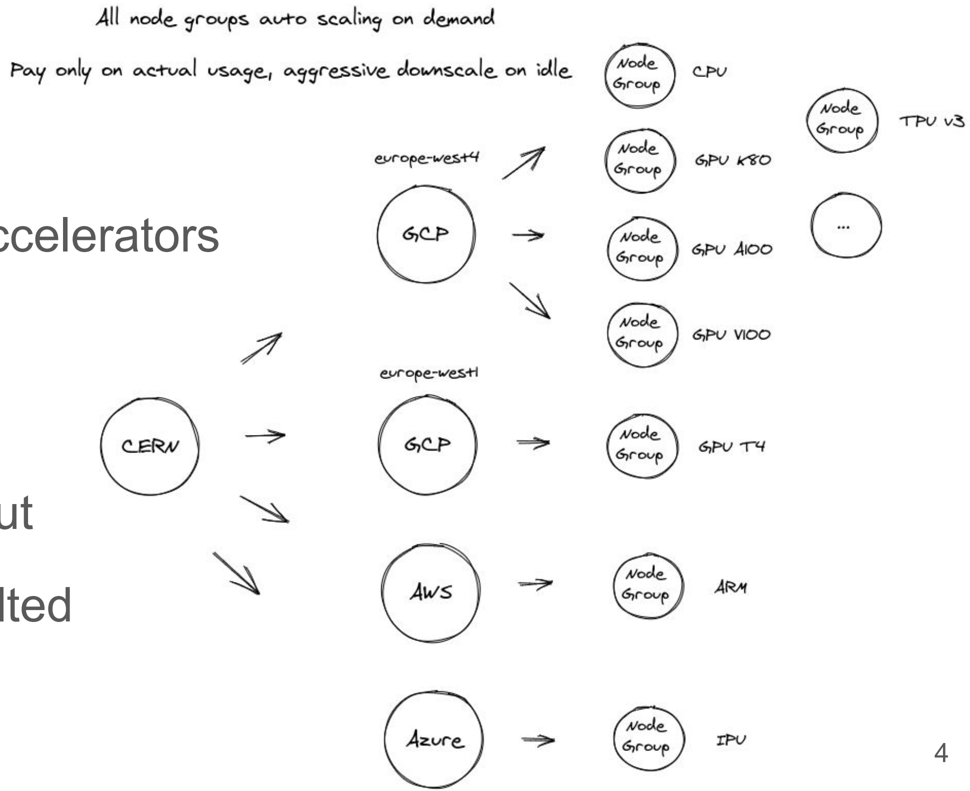
on premises GPU resources

Provide easy access to **public cloud** accelerators

(**GPUs**, **TPUs**, IPUs, FPGAs, …)

Bursting setup already demonstrated, but

access to resources temporarily halted



Infrastructure

Based on **Kubeflow**, the machine learning toolkit for Kubernetes

Kubeflow

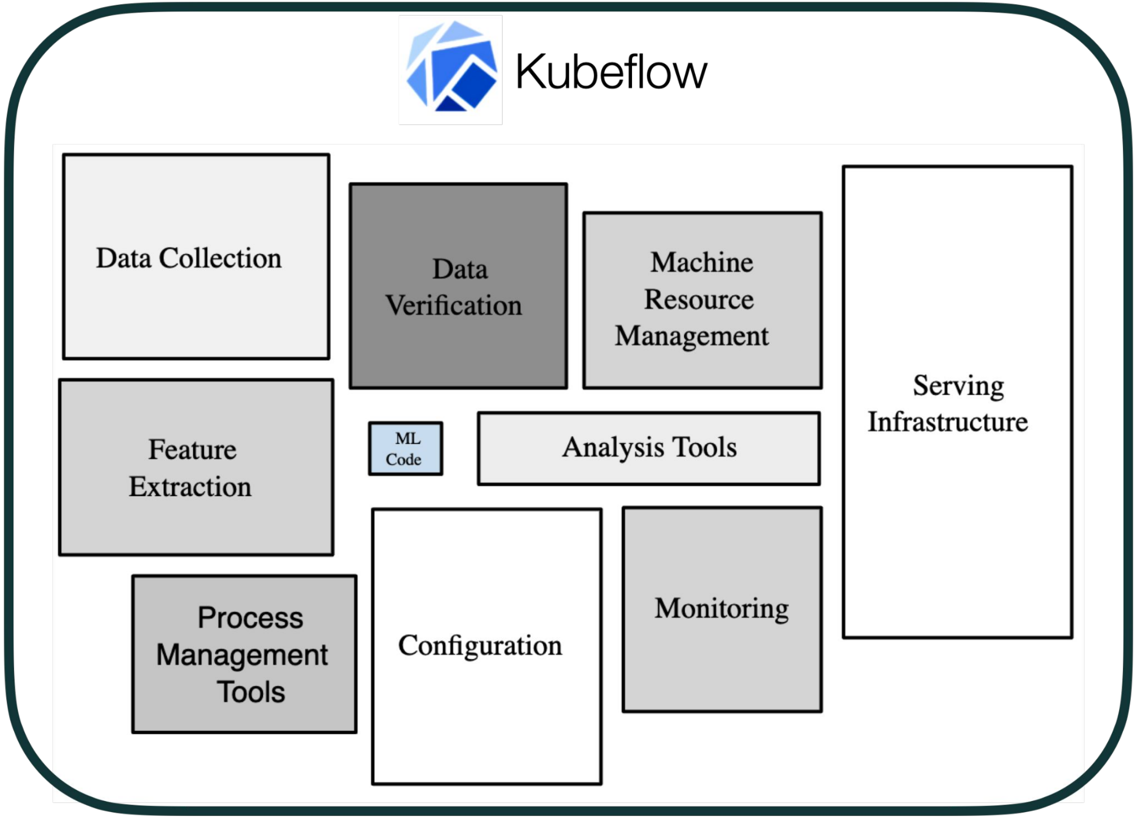
- Open source project started by Google in 2018
  - https://github.com/kubeflow/kubeflow

Declarative API, Operators, Auto healing, Application and Cluster auto scaling

Support for most common frameworks (**TensorFlow**, **PyTorch**, MXNET, …)

In production at multiple companies

- Google, Spotify, Bloomberg, Zillow, Arrikto...


Kubeflow
Data Collection
Data Verification
Machine Resource Management
Serving Infrastructure
Feature Extraction
ML Code
Analysis Tools
Process Management Tools
Configuration
Monitoring

Kubeflow Components and Features

Notebooks

Machine Learning Pipelines

AutoML - Hyperparameter Optimization

Distributed Training

Tensorboards

Model Serving

Notebooks

Easiest way to **start experimenting** with Kubeflow

Integration with other Kubeflow components

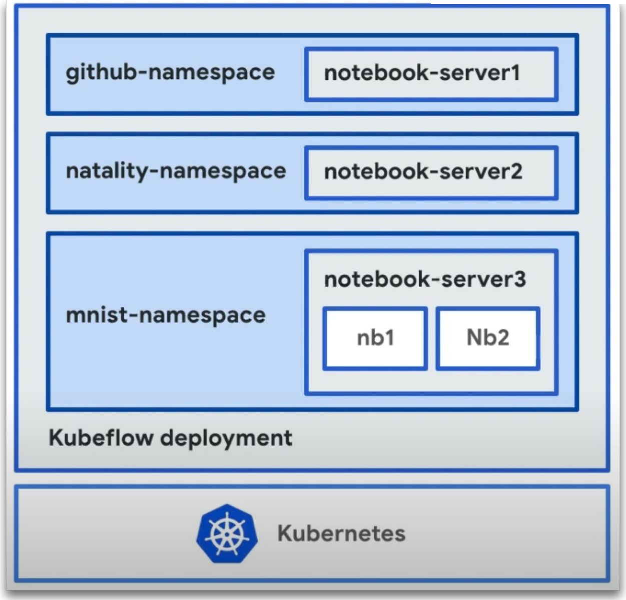
 Pipelines, distributed training, inference, AutoML

Ability to **customize Python environment**

 Or use prebuilt images (Tensorflow, Pytorch)

Select resources (CPU, MEM, GPU)

Good for **experimentation and prototyping phase**



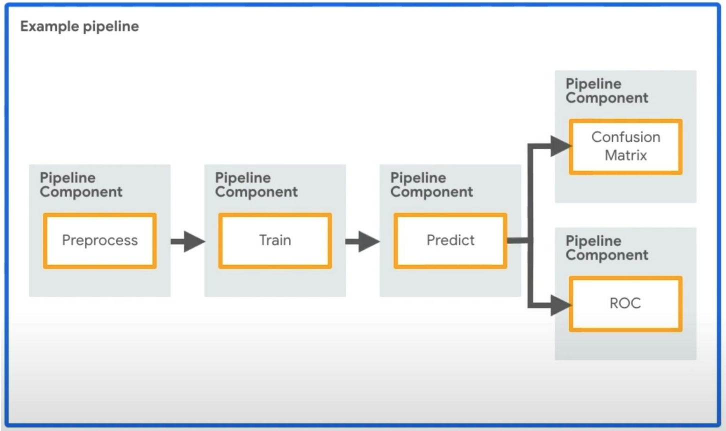
Machine Learning Pipelines

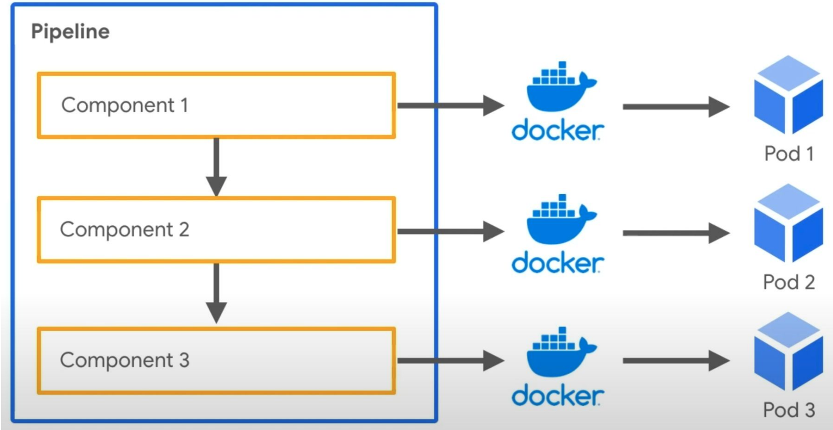
Automated ML workflows

A **user interface** (UI) for managing and tracking experiments, jobs, and runs

An **engine** for scheduling multi-step ML workflows

An **SDK/API** for defining and compiling pipelines and components





Benefits of Machine Learning Pipelines

Clear isolation between components

Can be scheduled to run periodically

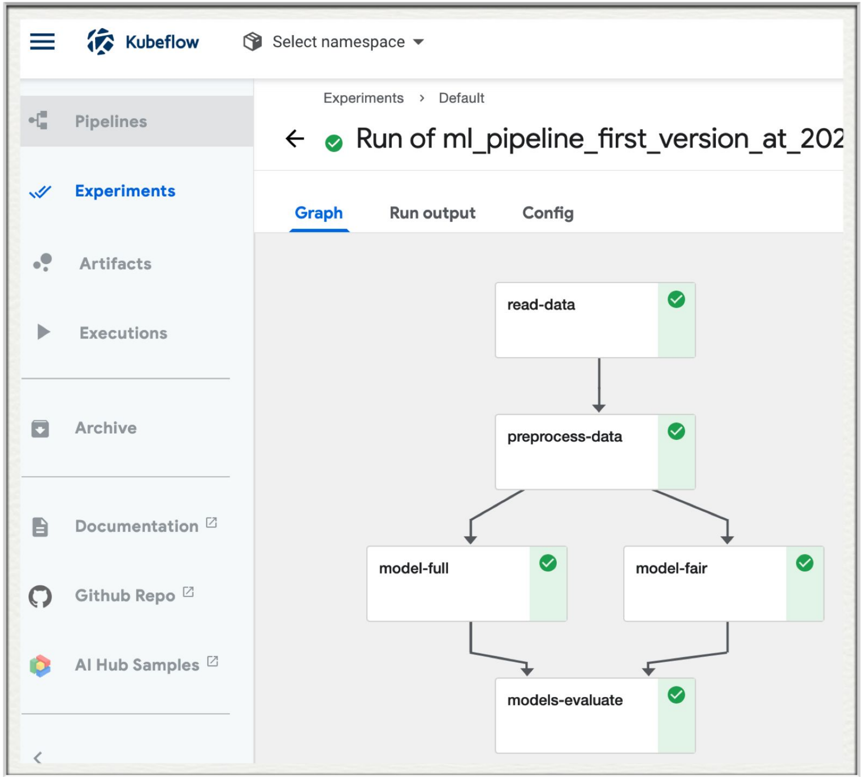
Can run with different input parameters

Versioning

Parallelisation

Non-blocking GPU access

Remote submissions with a client SDK



AutoML (Katib) - Hyperparameter Optimization

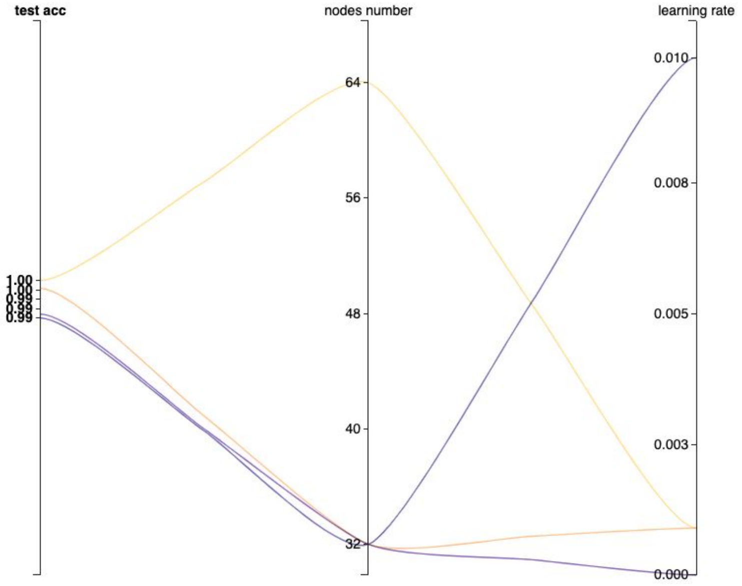
Standardized **development process**

- Create a training script
- Build a Docker image
- Run with various sets of inputs

Improved **hardware efficiency**

- Run each trial on a separate GPU

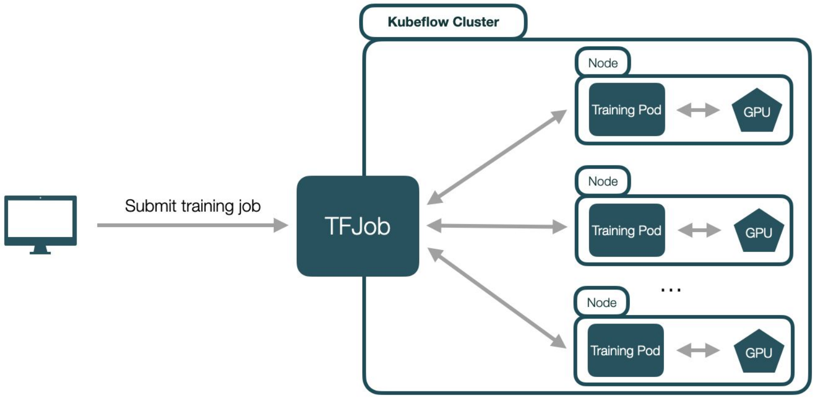
Visualization of **results and metrics**



OVERVIEW TRIALS DETAILS YAML

Trial name	Status	Test acc	Nodes number	Learning rate
test-wze6q-brnmwpbc	Succeeded	0.99593	32	0.001
test-wze6q-kqvz9zcp	Succeeded	0.98831	32	0.01
test-wze6q-lnbhttzs	Succeeded	0.98932	32	0.0001
test-wze6q-qvhz6pm8	Succeeded	0.99796	64	0.001

Distributed Training



Split training jobs across multiple GPUs

TensorFlow, **Pytorch** and other frameworks s

Jobs are split across multiple **local GPUs**

TFJob, PytorchJob… - Kubernetes custom resources for distributed training

Jobs are split across multiple **cluster GPUs**

Operators for Tensorflow and Pytorch to support containerised distribution

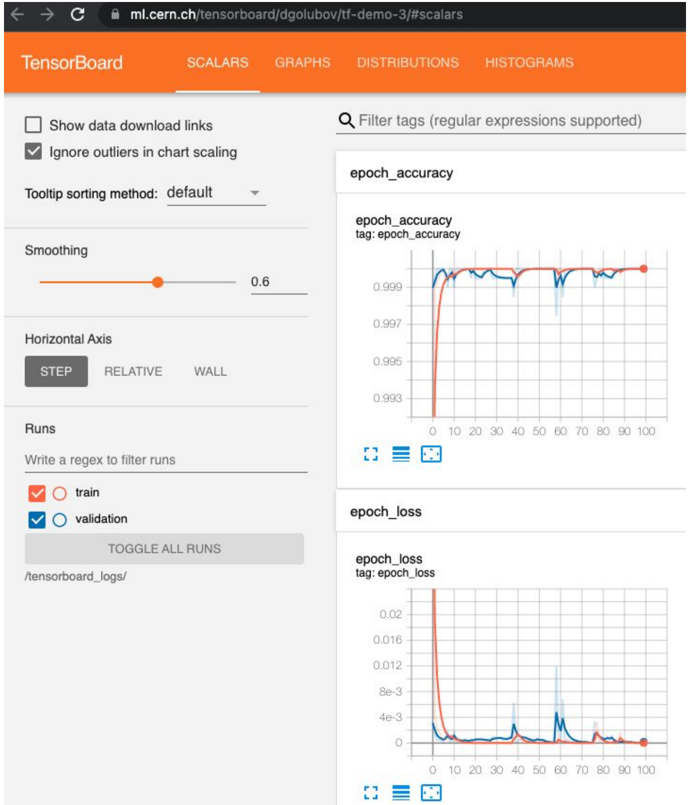
Tensorboards

Measurements and visualizations for ML workloads

- Track **loss and accuracy**
- Visualize **model graph**
- View custom metrics

Kubeflow allows creation of **Tensorboard servers**

- Monitor model training real-time
- Training from any Kubeflow component



Model Serving

Deploy a server to **run inference via http requests**

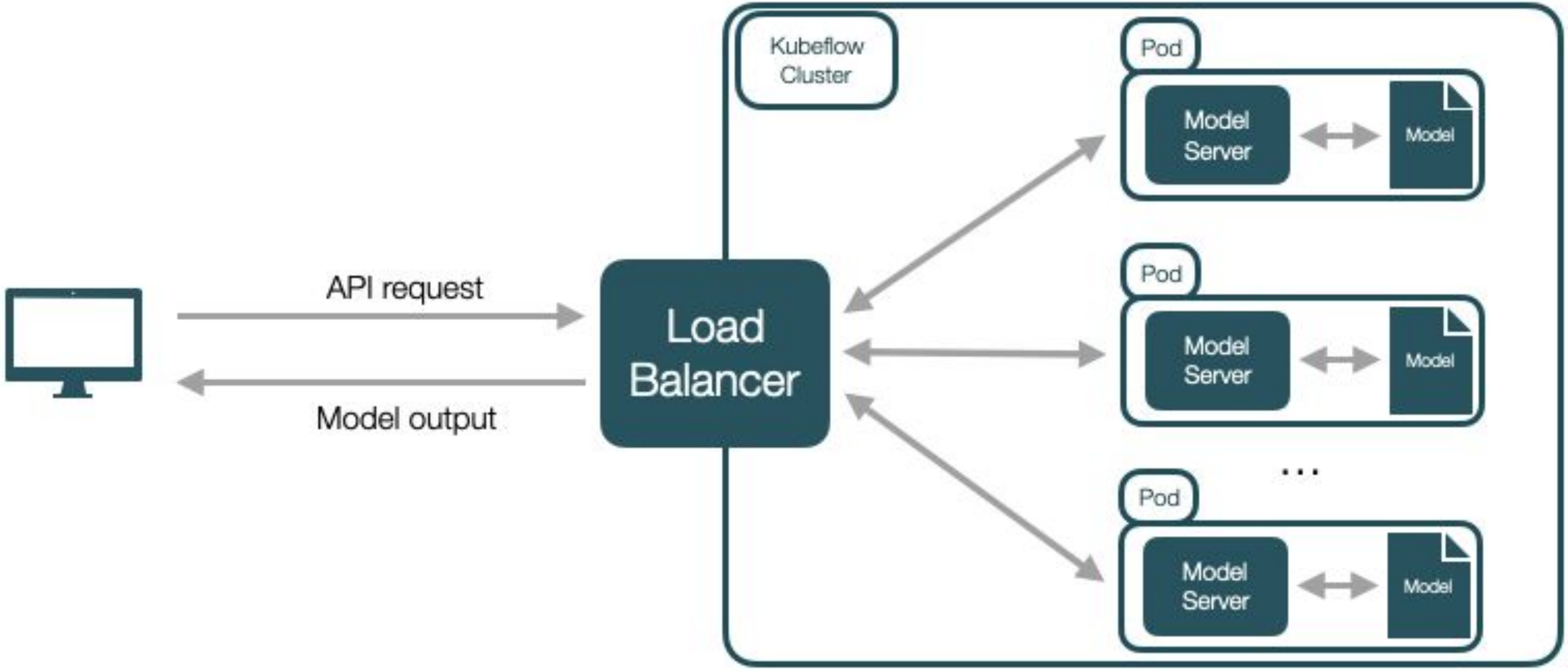
curl -v -H “Host: host” “http://host_ip/v1/models/mnist:predict" -d @./input.json

Serverless architecture

Automatic scaling per number of requests

Provided via **KServe** component

Model Serving



Demo

Pipelines

AutoML with Distributed Training

Tensorboard

Serving

Resource Management - Current

Resources assigned **per profile**

- Memory, CPU, GPU, Kubernetes resources…
- Kubernetes ***ResourceQuota*** defined for each profile

Personal profiles have a quota of **1 GPU** by default

Quotas for **group profiles** can be increased

- For now by contacting us directly
- Soon via dedicated ServiceNow form

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: kf-resource-quota
  namespace: dgolubov
status:
  hard:
    limits.cpu: "5"
    limits.memory: 10Gi
    limits.nvidia.com/gpu: "1"
    (soon) limits.nvidia.com/gpu.shared: "1"
    requests.cpu: "5"
    requests.memory: 10Gi
    requests.nvidia.com/gpu: "1"
    (soon) requests.nvidia.com/gpu.shared: "1"
```

Resource Management - Upcoming

Support for **time-sliced** NVIDIA GPUs

- **Multiple pods can access a single GPU**, allowing for time sharing
- Useful for smaller workloads, ex. notebooks with infrequent GPU utilization

Support for **physically sliced** NVIDIA GPUs

- **A GPU memory is physically split, without concurrent access**
- Useful for medium to large workloads that require constant GPU access

A ServiceNow form for requesting resources for group profiles

Storage Integration

EOS supported with Kerberos authentication

S3 object storage supported via S3 clients authentication

s3.cern.ch or public cloud providers (Amazon S3, Google Object Storage…)

registry.cern.ch - registry for the built images

Conclusions

Community effort to improve machine learning infrastructure

Kubeflow ongoing active development

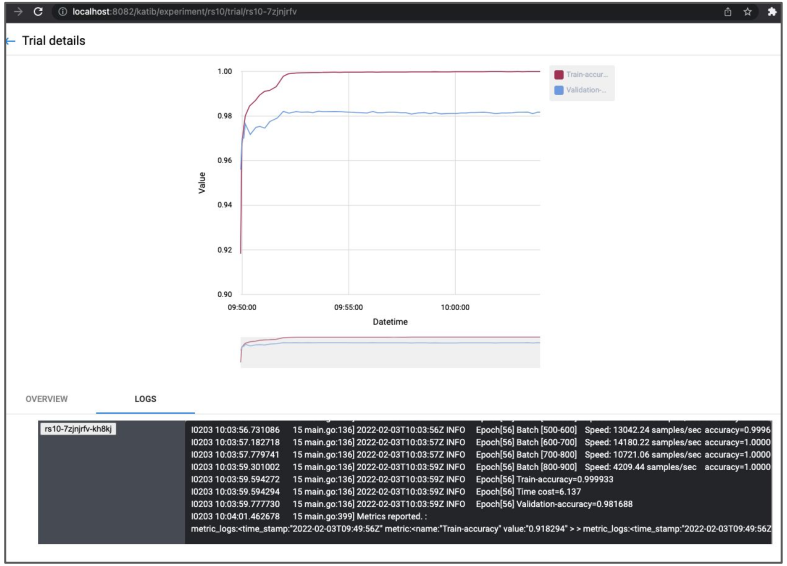
CERN users can influence future developments

High interest in our feedback

Anyone can contribute to open source

https://github.com/kubeflow

Everyone is invited to **provide feedback**!



Previous Talks

KubeCon Europe 2022, May 17 2022

Jet Energy Corrections with GNN Regression using Kubeflow @ CERN

IT Technical Forum CERN, November 19 2021

Centralized Management of Your Machine Learning Lifecycle

KubeCon North America 2021, October 12 2021

A Better and More Efficient ML Experience for CERN Users

KubeCon Europe 2021, May 6 2021

Building and Managing a Centralized ML Platform with Kubeflow at CERN

25th International Conference on Computing in High-Energy and Nuclear Physics, May 20 2021

Training and Serving ML workloads with Kubeflow at CERN

Fast Machine Learning for Science Workshop, Dec 01 2020

Making ML Easier with Kubeflow

Important Links

https://ml.cern.ch - the service landing page

ml.docs.cern.ch - documentation pages

https://gitlab.cern.ch/ai-ml/examples - examples repository

https://mattermost.web.cern.ch/it-dep/channels/ml - Mattermost channel

For any questions, please write here

Others may benefit from your questions!

Q&A

Backup Slides

Where we are today

Already **offering a lot** of what users ask for

Code validation on notebooks / small jobs

Distributed Training, Hyper Parameter Optimization

Model Serving

Direct kubernetes access still required in some cases

Launching distributed training jobs, checking katib logs

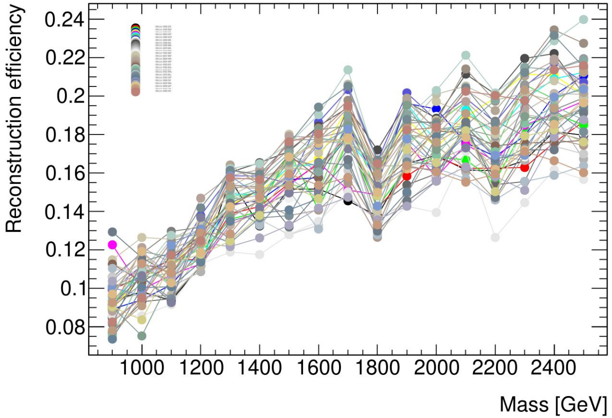
Early Adopters

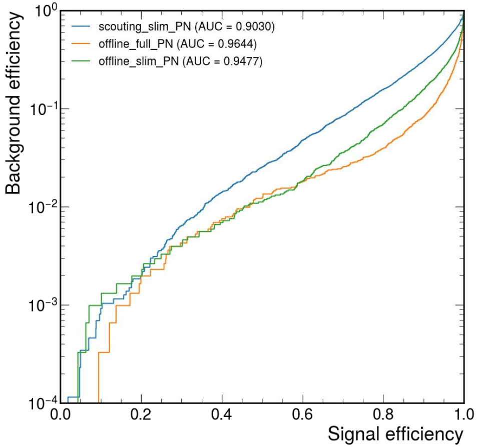
ATLAS Susy Search with Spanet

- PyTorch training
- Prototyping with Jupyter Notebooks
- Hyperparameter optimization with Katib

CMS Jet Tagging with ParticleNet

- Distributed PyTorch training
- Hyperparameter optimization with Katib





Early Adopters

OpenLab 3DGAN

GPU Benchmark

Scaling to 128 GPUs

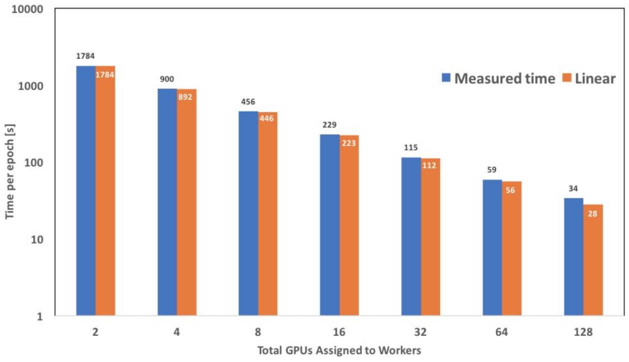
Distributed Tensorflow Training

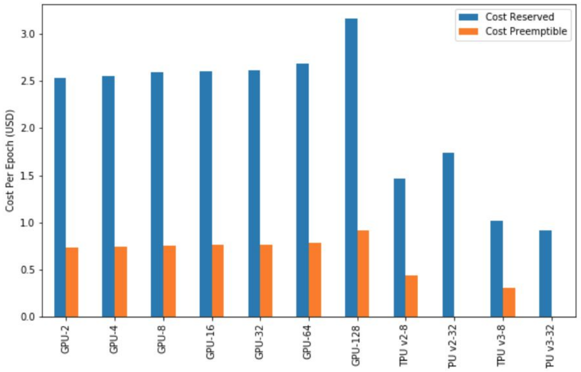
ADMON - Anomaly Detection

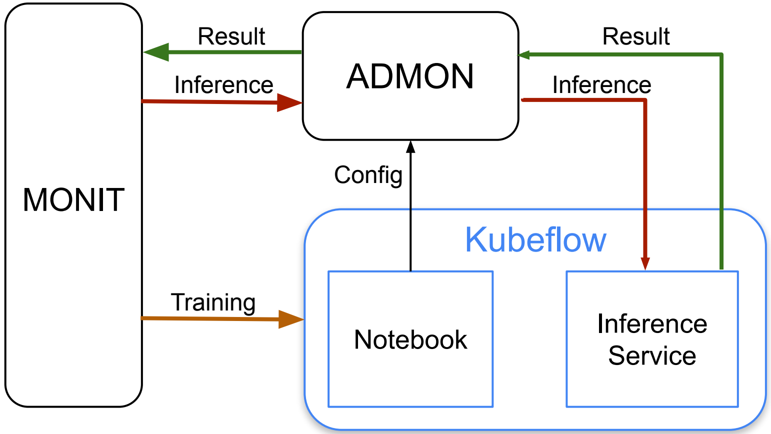
Continuous analysis of MONIT data

Model Serving with KServe

Transformer + Predictor







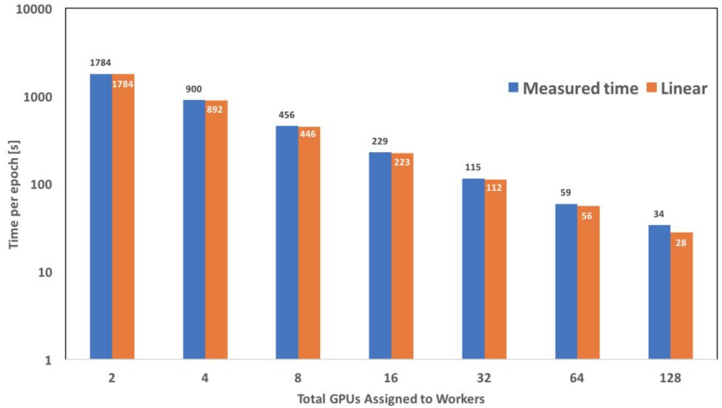
3DGAN OpenLab

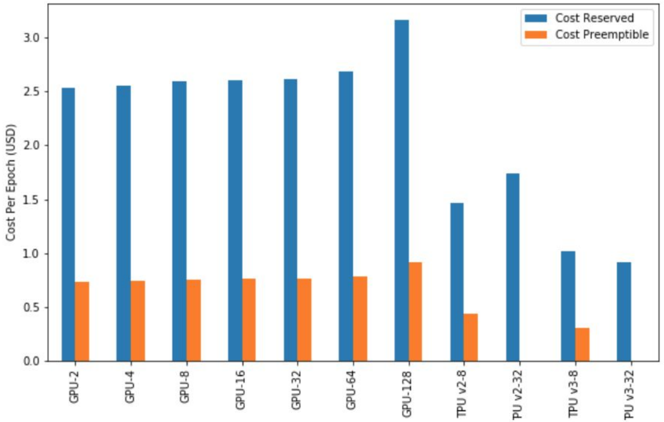
Run distributed training with **TFJob** on a **local cluster** and on a **public cloud**

Using up to 128 preemptible **Google Cloud GPUs** and **TPUs**

Resources from the Cloudbank EU project

Particularly well behaved, performance improvement close to linear





Accelerating GAN training using highly parallel hardware on public cloud, CHEP 2021
https://doi.org/10.1051/epjconf/202125102073

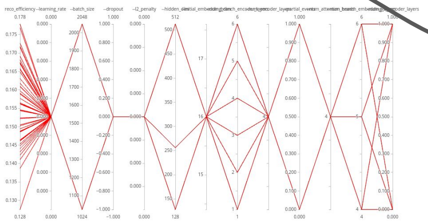
My experience with Kubeflow | Jona Bossio (CERN)

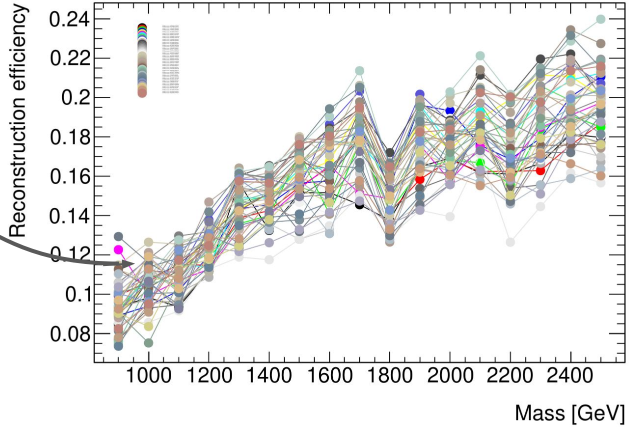
Context:

- ML project within an ATLAS SUSY search
- **Goal**: assign (uniquely) every resonance particle to a set of jets while preserving symmetries and handling a variable-size set of input jets
- **Plan**: Use SPANet (Symmetry Preserving Attention Networks) to resolve such assignment problem

How I used ml.cern.ch so far?

- Jupyter notebook using a custom image (specifications: 1 GPU, CPU 0.5, Memory 1 GB)
    - I used such notebook for training(~1hr)/testing/inference with SPANet
- Kubeflow pipelines for doing a hyperparameter optimization with Katib





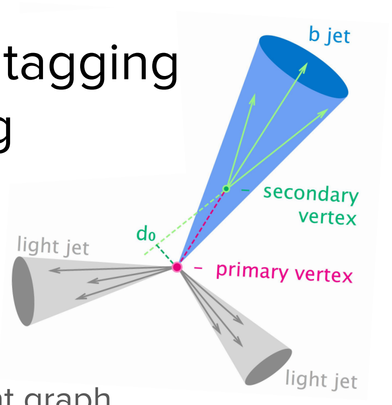
Notes:

- In both cases, inputs are outputs are located on EOS

Current issues/limitations:

- Not enough available machines with a GPU (limiting possible # of simultaneous trials, sometimes no GPU is available)
- Not possible to run over 24hs (limiting the size of the scan in the hyperparameter space) [kerberos token expires after 24hs]

Hackathon for Jet tagging with CMS Scouting



Infrastructure:

- ParticleNet [arXiv:1902.08570] (permutation-invariant graph neural network)

Goal:

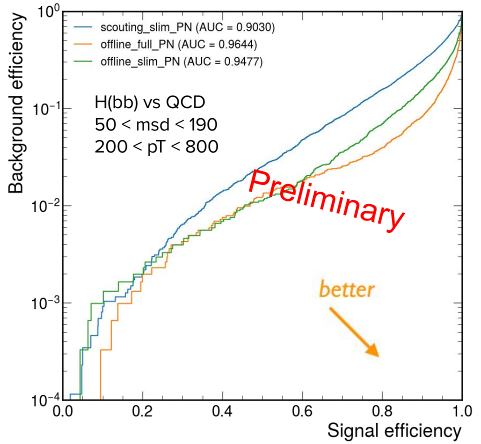
- Study the effects of the CMS Run3 Scouting reconstruction on the jet tagger performance

Team:

- Some experienced and some new to ML/K8s
- Everybody excited to have access to GPU

Our initial experience of Kubeflow:

- Very helpful team
    - Daniel Holmberg provided custom Docker image and example
    - Tutorial by Dejan Golubovic
- Limited to max 24h training (Krb5 authentication)
- Some cases more RAM needed
- Would be helpful to have access to Tensorboard for real-time monitoring
- We will continue to use Kubeflow to complete project



Adelina Lintuluoto (KIT, CERN), Clemens Lange (CERN), Congqiao Li (PKU), Huilin Qu (CERN), Loukas Gouskos (CERN), Phillip Masterson (USCB)

Early Adopters Experiences

Infrastructure

On Premises GPUs: Nvidia V100s and T4s

Full GPU card assignment

Distributed training, hyper parameter optimization, model serving

On Premises vGPUs (soon)

Time sharing of a GPU, up to 4x

Notebooks, code validation, quick iteration

Public Cloud GPUs, TPUs

Available on demand

Not yet fully integrated into the centralized service

A bit of history...

First kubeflow trials at CERN started ~2.5 years ago

Helping a few users scale out machine learning use cases

- From a single GPU to distributed training with 10s GPUs

Gradual expansion to a preview service (ongoing)

- Slowly onboarding new users
- Exploring the remaining bits of the ML lifecycle

A bit of history...

First kubeflow trials at CERN started ~2.5 years ago

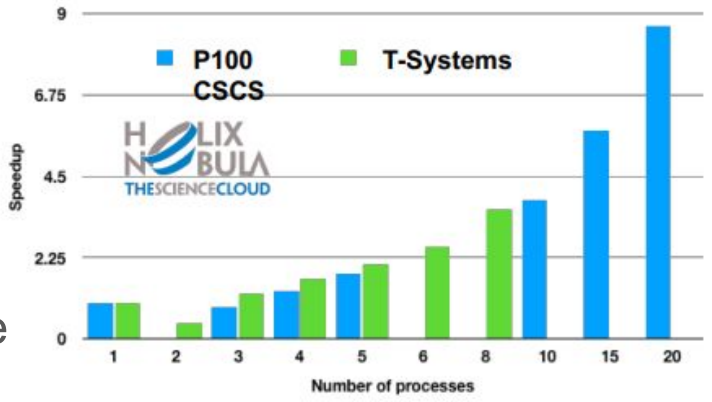
Helping a few users scale out machine learning use cases

From a single GPU to distributed training with 10s GPUs

Gradual expansion to a preview service (ongoing)

Slowly onboarding new users

Exploring the remaining bits of the ML lifecycle



[Sofia Vallecorsa: Deep Learning at CERN openlab: High Energy Physics and Beyond]

Coming Next

Enforced Quotas

Shared profiles based on ldap / egroups

Improved integration with **other services** - continuous integration, notebooks/swan, etc

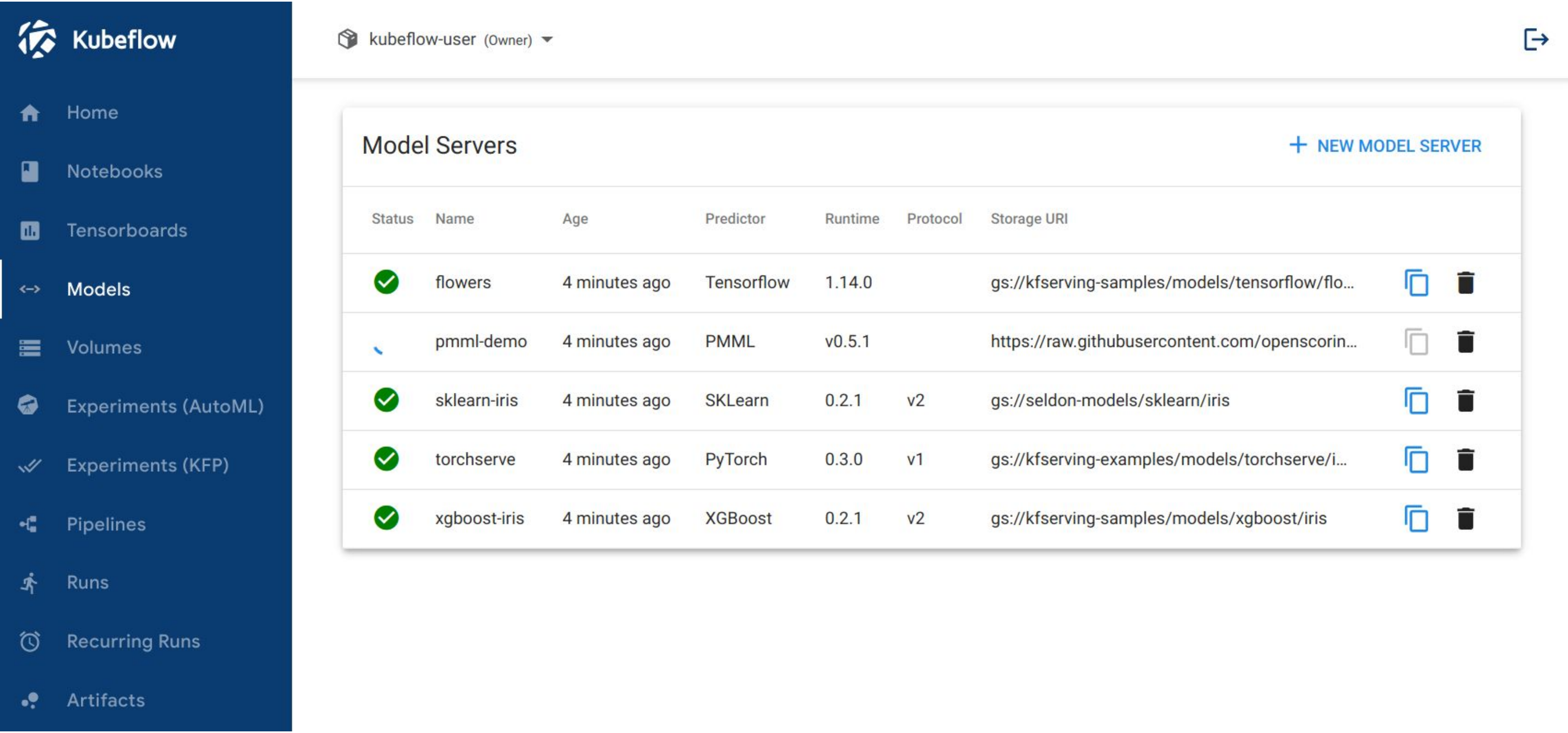
Security improvements

Prevent vulnerable workloads, runtime verification, etc

(ASDF) https://indico.cern.ch/event/1054454/

Artifact navigation

Experiment with a dataset catalog and automated feature discovery


Kubeflow
kubeflow-user (Owner)
Home
Notebooks
Tensorboards
Models
Volumes
Experiments (AutoML)
Experiments (KFP)
Pipelines
Runs
Recurring Runs
Artifacts
Model Servers
NEW MODEL SERVER
Status Name Age Predictor Runtime Protocol Storage URI
flowers 4 minutes ago Tensorflow 1.14.0 gs://kfserving-samples/models/tensorflow/flo...
pmml-demo 4 minutes ago PMML v0.5.1 https://raw.githubusercontent.com/openscorin...
sklearn-iris 4 minutes ago SKLearn 0.2.1 v2 gs://seldon-models/sklearn/iris
torchserve 4 minutes ago PyTorch 0.3.0 v1 gs://kfserving-examples/models/torchserve/i...
xgboost-iris 4 minutes ago XGBoost 0.2.1 v2 gs://kfserving-samples/models/xgboost/iris

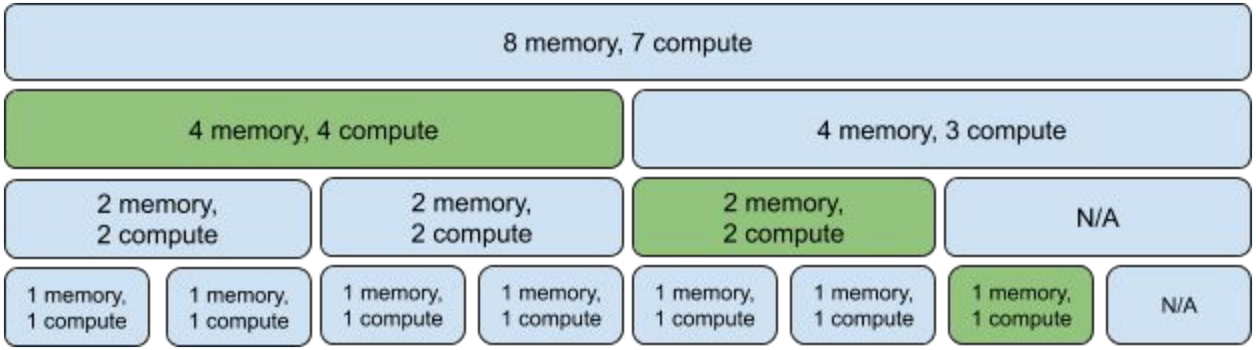
Infrastructure

Expanding soon: Nvidia A100s

Coming to CERN IT beginning of next year

Physical partition of each card up to 7x - Multi Instance GPU (MIG)

Many other layouts possible



Demo

Current Limitations I

Pipelines and pipeline artifacts isolation

Experiments and runs are isolated, but pipelines are not

The fix is coming up, ~3 months

Katib jobs logging

Not there yet, Katib jobs can be monitored using `kubectl`

Automatic credential renewal - coming up in ~1 month

Accessing **cookies for model serving**, currently done manually

Current Limitations II

Running a **Tensorboard server**

Notebook culling not in place yet

Remove unused notebooks periodically

GPU / vGPU availability numbers for notebook creation

No built-in support yet for model versioning, model catalog

Best Practices for Scalable ML

Code and Datasets Storage

Containers are **ephemeral**, notebook servers can crash

Keep code and datasets on **persistent storage** that can be easily accessed

- Code - **Github** or **GitLab**, commit regularly
- Datasets - **EOS** or **S3** object storage

Model Training

Develop models with scalability in mind

Develop model training to be **distributed** across multiple GPUs

Only **prototype** using a single GPU

TF Distributed - TFJob

Pytorch Distributed - PyTorchJob

TFJob Example

```yaml
apiVersion: kubeflow.org/v1
kind: TFJob
metadata:
  generateName: tfjob
  namespace: your-user-namespace
spec:
  tfReplicaSpecs:
    PS:
      replicas: 1
      restartPolicy: OnFailure
      template:
        metadata:
          annotations:
            sidecar.istio.io/inject: "false"
        spec:
          containers:
          - name: tensorflow
            image: gcr.io/your-project/your-image
            command:
              - python
              - -m
              - trainer.task
              - --batch_size=32
              - --training_steps=1000
```

```yaml
Worker:
  replicas: 3
  restartPolicy: OnFailure
  template:
    metadata:
      annotations:
        sidecar.istio.io/inject: "false"
    spec:
      containers:
      - name: tensorflow
        image: gcr.io/your-project/your-image
        command:
          - python
          - -m
          - trainer.task
          - --batch_size=32
          - --training_steps=1000
```

Tensorflow Distributed Example

```
strategy = tf.distribute.MultiWorkerMirroredStrategy()

with strategy.scope():
  model = tf.keras.Sequential([
    tf.keras.layers.Dense(2, input_shape=(5,)),
  ])
  optimizer = tf.keras.optimizers.SGD(learning_rate=0.1)

def dataset_fn(ctx):
  x = np.random.random((2, 5)).astype(np.float32)
  y = np.random.randint(2, size=(2, 1))
  dataset = tf.data.Dataset.from_tensor_slices((x, y))
  return dataset.repeat().batch(1, drop_remainder=True)
dist_dataset = strategy.distribute_datasets_from_function(dataset_fn)

model.compile()
model.fit(dist_dataset)
```

Containerised Workloads

Build Docker images for your ML workloads

Allows for **reproducibility**

Mobility - can run anywhere

Fast deployment

Integration with multiple Kubeflow components

- Pipelines
- Distributed training
- Hyperparameter optimization

Building Docker Images

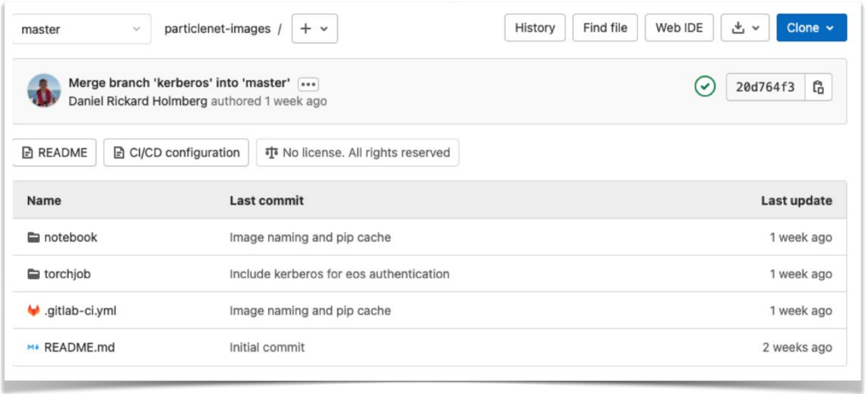
Automate builds with **Gitlab CI**

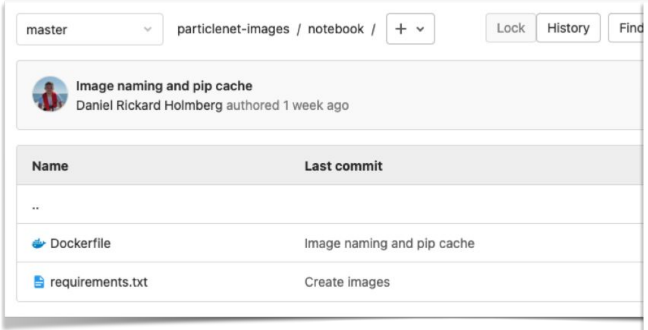
Trigger image build with every push

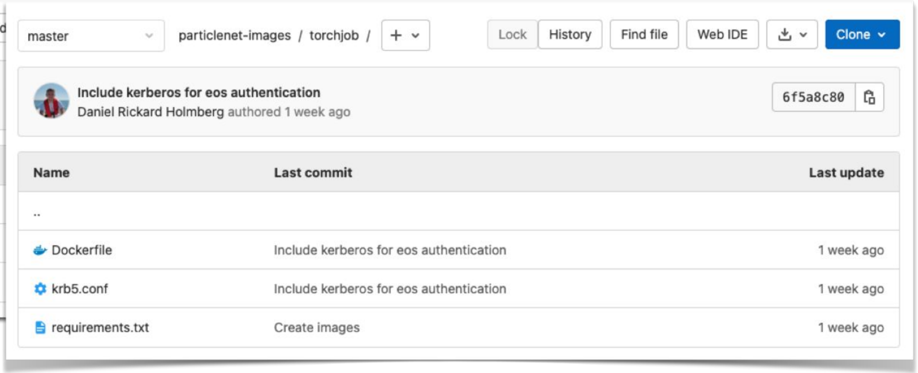
Store images on CERN registries

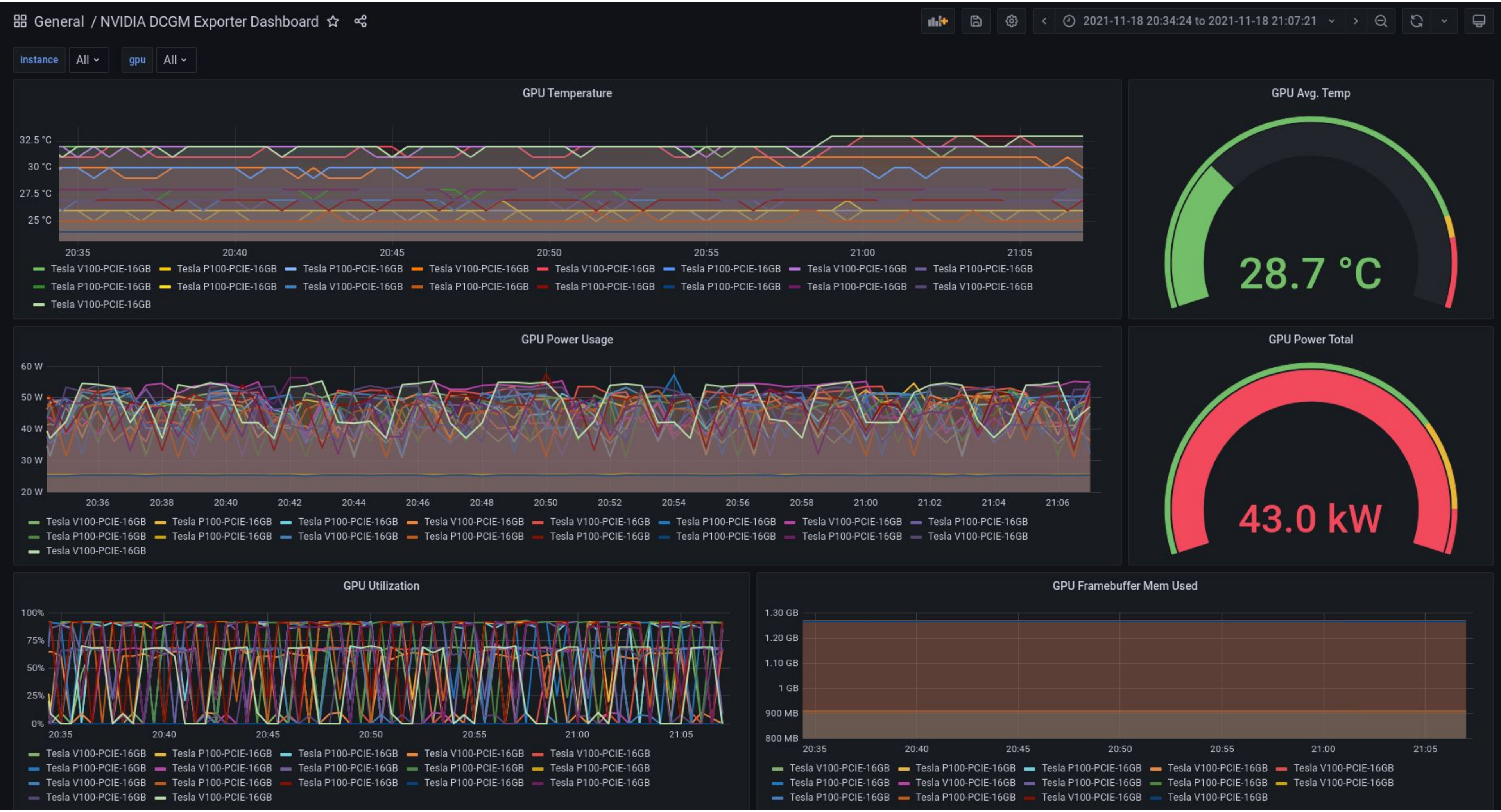
- https://gitlab.cern.ch/ci-tools/docker-image-builder
- https://clouddocs.web.cern.ch/containers/registry/gitlab.html

Building Docker Images








General / NVIDIA DCGM Exporter Dashboard
2021-11-18 20:34:24 to 2021-11-18 21:07:21
instance All
gpu All
GPU Temperature
32.5 °C
30 °C
27.5 °C
25 °C
20:35 20:40 20:45 20:50 20:55 21:00 21:05
Tesla V100-PCIE-16GB Tesla P100-PCIE-16GB Tesla P100-PCIE-16GB Tesla V100-PCIE-16GB Tesla V100-PCIE-16GB Tesla P100-PCIE-16GB Tesla V100-PCIE-16GB Tesla P100-PCIE-16GB
Tesla P100-PCIE-16GB Tesla P100-PCIE-16GB Tesla V100-PCIE-16GB Tesla P100-PCIE-16GB Tesla P100-PCIE-16GB Tesla P100-PCIE-16GB Tesla P100-PCIE-16GB Tesla V100-PCIE-16GB
Tesla V100-PCIE-16GB
GPU Avg. Temp
28.7 °C
GPU Power Usage
60 W
50 W
40 W
30 W
20 W
20:36 20:38 20:40 20:42 20:44 20:46 20:48 20:50 20:52 20:54 20:56 20:58 21:00 21:02 21:04 21:06
Tesla V100-PCIE-16GB Tesla P100-PCIE-16GB Tesla P100-PCIE-16GB Tesla V100-PCIE-16GB Tesla V100-PCIE-16GB Tesla P100-PCIE-16GB Tesla V100-PCIE-16GB Tesla P100-PCIE-16GB
Tesla P100-PCIE-16GB Tesla P100-PCIE-16GB Tesla V100-PCIE-16GB Tesla P100-PCIE-16GB Tesla P100-PCIE-16GB Tesla P100-PCIE-16GB Tesla P100-PCIE-16GB Tesla V100-PCIE-16GB
Tesla V100-PCIE-16GB
GPU Power Total
43.0 kW
GPU Utilization
100%
75%
50%
25%
0%
20:35 20:40 20:45 20:50 20:55 21:00 21:05
Tesla V100-PCIE-16GB Tesla P100-PCIE-16GB Tesla P100-PCIE-16GB Tesla V100-PCIE-16GB Tesla V100-PCIE-16GB
Tesla P100-PCIE-16GB Tesla V100-PCIE-16GB Tesla P100-PCIE-16GB Tesla P100-PCIE-16GB Tesla P100-PCIE-16GB
Tesla V100-PCIE-16GB Tesla P100-PCIE-16GB Tesla P100-PCIE-16GB Tesla P100-PCIE-16GB Tesla P100-PCIE-16GB
Tesla V100-PCIE-16GB Tesla V100-PCIE-16GB
GPU Framebuffer Mem Used
1.30 GB
1.20 GB
1.10 GB
1 GB
900 MB
800 MB
20:35 20:40 20:45 20:50 20:55 21:00 21:05
Tesla V100-PCIE-16GB Tesla P100-PCIE-16GB Tesla P100-PCIE-16GB Tesla V100-PCIE-16GB Tesla V100-PCIE-16GB
Tesla P100-PCIE-16GB Tesla V100-PCIE-16GB Tesla P100-PCIE-16GB Tesla P100-PCIE-16GB Tesla V100-PCIE-16GB
Tesla P100-PCIE-16GB Tesla P100-PCIE-16GB Tesla V100-PCIE-16GB Tesla V100-PCIE-16GB

Hyperparameter Jobs

Make sure the script runs on a **single GPU in a notebook server**

Make sure it uses a GPU + it completes

Make sure the script runs on a **single GPU as a Katib job with 1 trial**

Then carefully expand to 2, 4, 10, 20 trials, and monitor closely

Once sure it works, run a complete search

Be aware that **some combinations of HP might crash the script**

Prepare the script for these edge cases (exception handling etc)

Store metrics in a preferred format in two places

In the container storage to be accessed by the UI

On EOS or S3 for persistent storage