

SophiaConf



Jean Armel Luce



About me ...



I work at Orange



I like to try, test, experiment and push to production

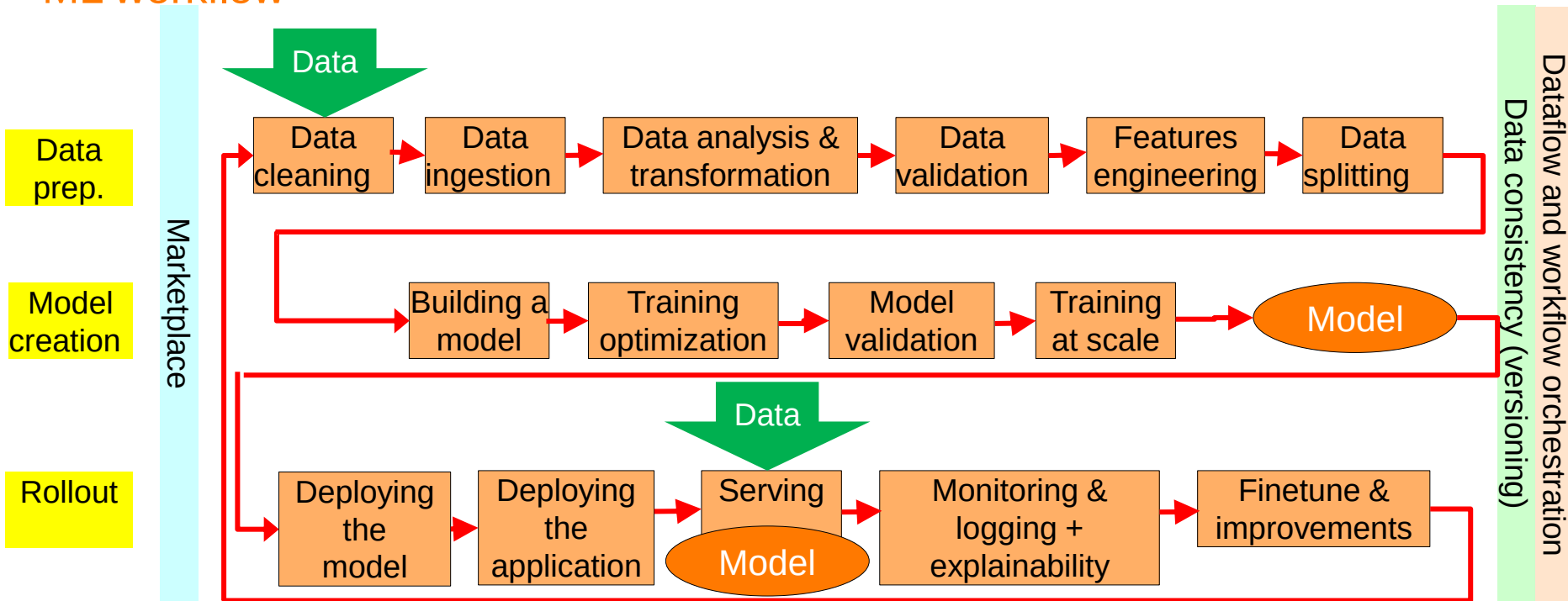


I like to share

- Meetups and local events
- C* Summit EU 2013 (London) : migration from RDBMS to C*
- C* Summit 2014 (San Francisco) : graphs and Hive with C*
- C* Summit 2016 (San Jose) : ML with Spark and C*
- ApacheCon N.A. 2019 (Las Vegas) : C* deployment on top of K8S



ML workflow



Several different actors



... using several different tools
... and some hardware resources





Make it Easy for Everyone to
Develop, Deploy and Manage
Portable, Distributed ML on
Kubernetes

- Kubernetes
- Microservices
- Serverless

Kubernetes : what ? where ?

- Tasks scheduling
- Automated deployments and upgrades
- Self healing
- Metrics & logs collection
- Security
- ...

Anywhere you are running Kubernetes, you should be able to run Kubeflow :

➤ Public clouds

➤ Private clouds



Microservices : main components Kubeflow in 01/2020



PyTorch

scikit-learn

mxnet

TensorFlow

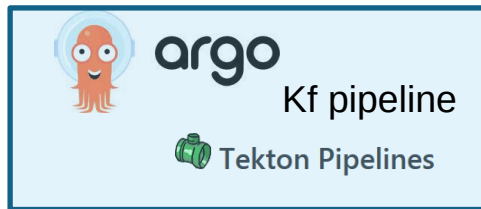
tensorboard tftraining

Kfdef

metadata

modeldb

metacontroller



argo

Kf pipeline

Tekton Pipelines



Kubeflow



Istio

Knative

kubernetes

Kubebench

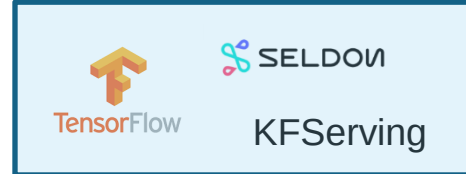


Spark



jupyter

Katib



TensorFlow

SELDON

KFServing



Google Cloud Platform

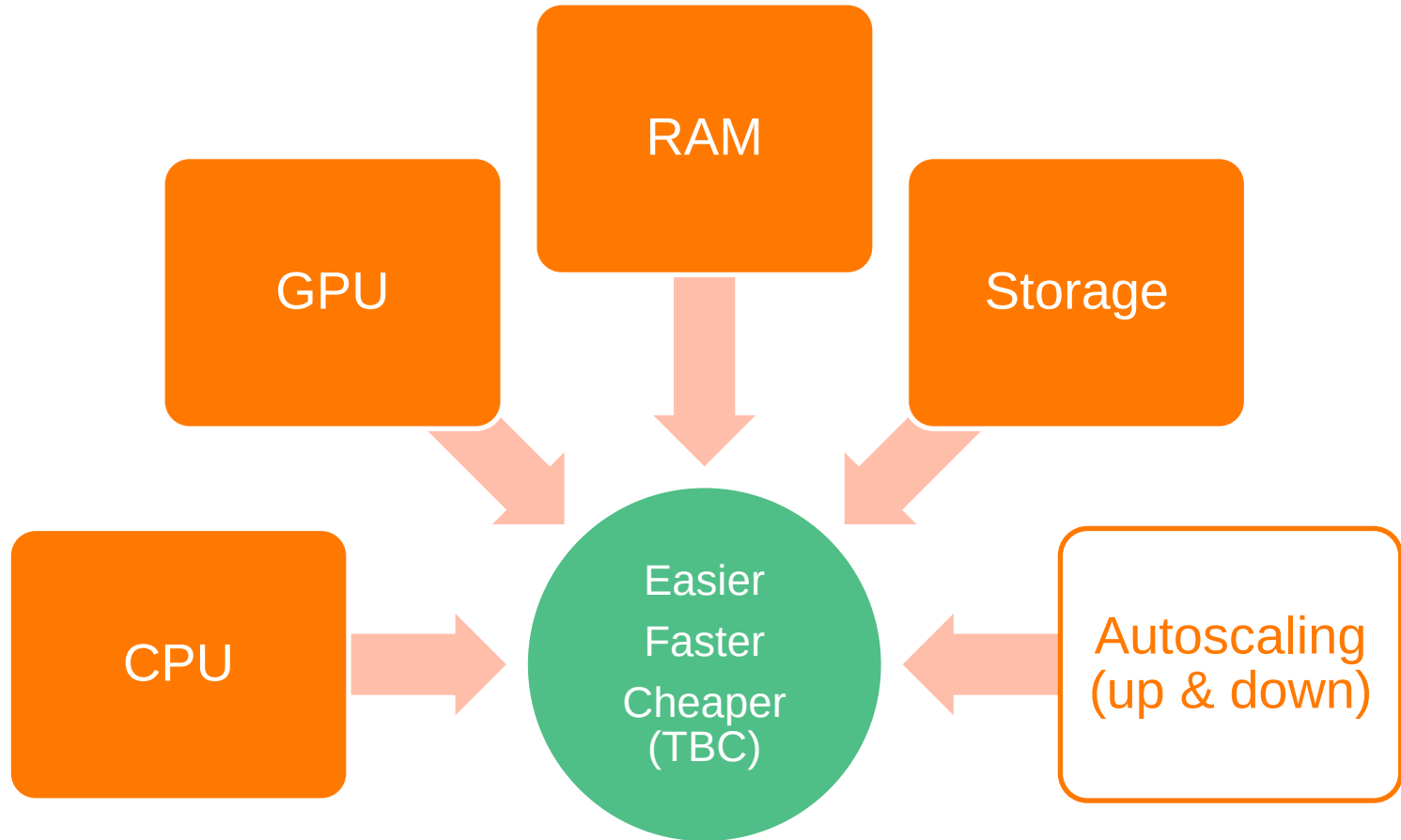
aws

Azure

On premise



Serverless : access to performant hardware



Kubeflow home + working storage

The screenshot shows the Kubeflow console interface for the namespace 'kubeflow-jeanarmel-luce'. The interface includes a left-hand navigation menu with links to Pipelines, Notebook Servers, Katib, Artifact Store, Manage Contributors, GitHub, and Documentation. The main content area is divided into several sections: Quick shortcuts, Recent Notebooks, Recent Pipelines, Recent Pipeline Runs, and a Google Cloud Platform sidebar with links to Stackdriver Logging, Project Overview, Deployment Manager, Kubernetes Engine, and Documentation. Annotations with orange arrows point to specific features: 'Workspace selection' points to the namespace dropdown, 'Notebook servers' points to the 'Notebook Servers' link in the navigation menu, and 'Manage contributors' points to the 'Manage Contributors' link in the navigation menu.

Workspace selection

Notebook servers

Manage contributors

Quick shortcuts

- Upload a pipeline
- View all pipeline runs
- Create a new Notebook server
- View Katib Studies
- View Metadata Artifacts

Recent Notebooks

- kserving_sdk_sample.ipynb
- lost+found
- lost+found
- classification2_clothes.ipynb
- lost+found

Recent Pipelines

- [Tutorial] DSL - Control structures
- [Tutorial] Data passing in python components
- [Demo] TFX - Taxi Tip Prediction Model Trainer
- [Demo] XGBoost - Training with Confusion Matrix

Recent Pipeline Runs

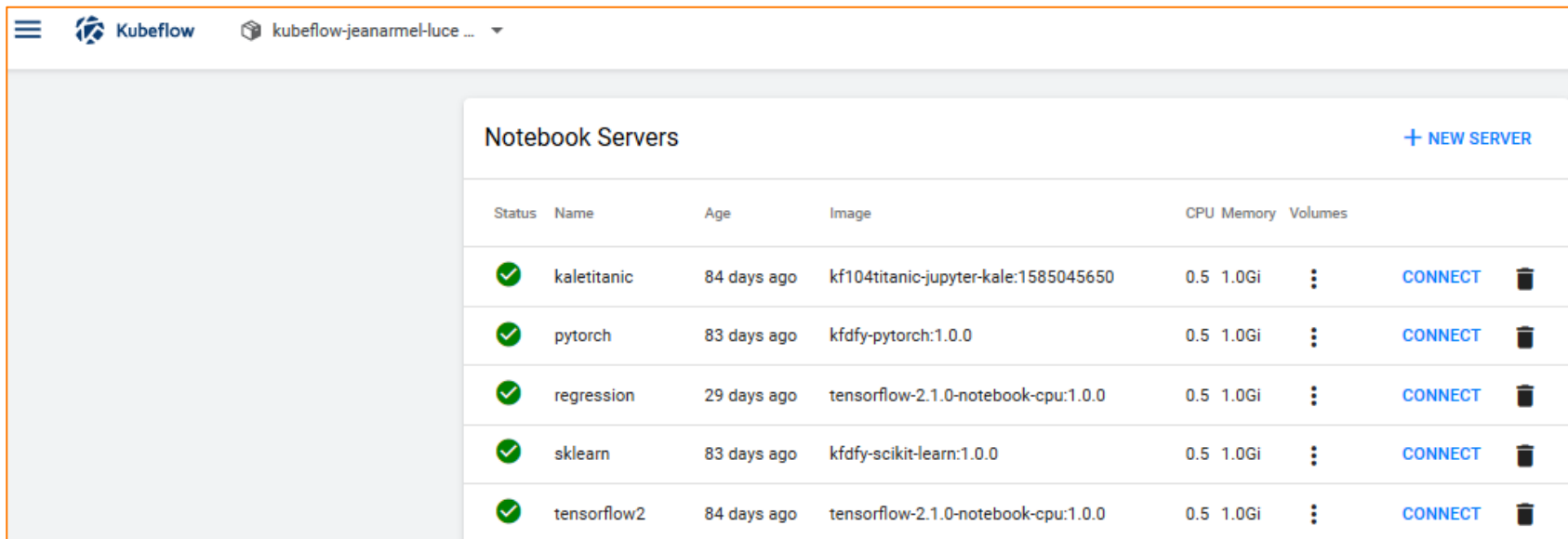
None Found

Google Cloud Platform

- Stackdriver Logging
- Project Overview
- Deployment Manager
- Kubernetes Engine
- Documentation
- Getting Started with Kubeflow
- Minikf
- Microk8s for Kubeflow
- Minikube for Kubeflow
- Kubeflow on GCP
- Kubeflow on AWS
- Requirements for Kubeflow

Jupyter notebooks list

A user may have multiple notebooks in a workspace ...



The screenshot shows the Kubeflow dashboard interface. At the top, there is a navigation bar with the Kubeflow logo and a dropdown menu showing 'kubeflow-jeanarmel-luce ...'. Below the navigation bar, the main content area displays a table titled 'Notebook Servers'. To the right of the title is a '+ NEW SERVER' button. The table has columns for Status, Name, Age, Image, CPU, Memory, Volumes, and actions. There are five rows of notebook servers listed, each with a green checkmark status, a name, an age, an image, CPU and memory specifications, and a 'CONNECT' button with a trash icon.

Status	Name	Age	Image	CPU	Memory	Volumes	
✓	kaletitanic	84 days ago	kf104titanic-jupyter-kale:1585045650	0.5	1.0Gi	⋮	CONNECT 
✓	pytorch	83 days ago	kdfy-pytorch:1.0.0	0.5	1.0Gi	⋮	CONNECT 
✓	regression	29 days ago	tensorflow-2.1.0-notebook-cpu:1.0.0	0.5	1.0Gi	⋮	CONNECT 
✓	sklearn	83 days ago	kdfy-scikit-learn:1.0.0	0.5	1.0Gi	⋮	CONNECT 
✓	tensorflow2	84 days ago	tensorflow-2.1.0-notebook-cpu:1.0.0	0.5	1.0Gi	⋮	CONNECT 

Jupyter notebook creation

A user may have several notebooks. A name is required to distinguish each of them

Image « Docker »

Resources (CPU, GPU, RAM, storage)

The screenshot shows a web form for creating a Jupyter Notebook. It includes sections for Name, Image, CPU / RAM, Workspace Volume, Data Volumes, Configurations, GPUs, and Miscellaneous Settings. Annotations with orange arrows point from text boxes to specific fields: 'A user may have several notebooks. A name is required to distinguish each of them' points to the Name field; 'Image « Docker »' points to the Image dropdown; 'Resources (CPU, GPU, RAM, storage)' points to the CPU, Memory, Workspace Volume, and GPUs sections. A 'Workspace' label with an arrow points to the Namespace field.

Name
Specify the name of the Notebook Server and the Namespace it will belong to.

Name: Namespace:

Image
A starter Jupyter Docker Image with a baseline deployment and typical ML packages.

☐ Custom Image
Image:

CPU / RAM
Specify the total amount of CPU and RAM reserved by your Notebook Server. For CPU-intensive workloads, you can choose more than 1 CPU (e.g. 1.5).

CPU: Memory:

Workspace Volume
Configure the Volume to be mounted as your personal Workspace.

☐ Don't use Persistent Storage for User's home

Type: Name: Size: Mode: Mount Point:

Data Volumes
Configure the Volumes to be mounted as your Datasets.

[+ ADD VOLUME](#)

Configurations
Extra layers of configurations that will be applied to the new Notebook. (e.g. Insert credentials as Secrets, set Environment Variables.)

Configurations:

GPUs
Specify the number and Vendor of GPUs that will be assigned to the Notebook Server's Container.

Number of GPUs: GPU Vendor:

Miscellaneous Settings
Other possible settings to be applied to the Notebook Server.

☒ Enable Shared Memory

[LAUNCH](#) [CANCEL](#)

Workspace

Jupyter notebook

examples/mnist/

jupyter mnist_gcp Last Checkpoint: Last Thursday at 15:37 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Markdown

MNIST end to end on Kubeflow on GKE

This example guides you through:

1. Taking an example TensorFlow model and modifying it to support distributed training.
2. Serving the resulting model using TF Serving.
3. Deploying and using a web app that sends prediction requests to the model.

Requirements

- You must be running Kubeflow 1.0 on Kubernetes Engine (GKE) with Cloud Identity-Aware Proxy (Cloud IAP).
- Run this notebook within your Kubeflow cluster. See the guide to [setting up your Kubeflow notebooks](#).

Prepare model

There is a delta between existing distributed MNIST examples and what's needed to run well as a TFJob.

Basically, you must:

- Add options in order to make the model configurable.
- Use `tf.estimator.train_and_evaluate` to enable model exporting and serving.
- Define serving signatures for model serving.

This tutorial provides a Python program that's already prepared for you: [model.py](#).

Verify that you have a Google Cloud Platform (GCP) account

The cell below checks that this notebook was spawned with credentials to access GCP.

```
In [1]: import logging
import os
import uuid
from importlib import reload
from oauth2client import import GoogleCredentials
credentials = GoogleCredentials.get_application_default()
```

Install the required libraries

Run the next cell to import the libraries required to train this model.

```
In [2]: import notebook_setup
reload(notebook_setup)
notebook_setup.notebook_setup()

pip installing requirements.txt
Checkout kubeflow/tf-operator @9238986
Adding /home/jovyan/git_tf-operator/sdk/python to python path
Configure docker credentials
```

Wait for the message `Configure docker credentials` before moving on to the next cell.

```
In [3]: import k8s_util
# Force a reload of Kubeflow. Since Kubeflow is a multi namespace module,
# doing the reload in notebook_setup may not be sufficient.
import kubeflow
reload(kubeflow)
from Kubernetes import client as k8s_client
from Kubernetes import config as k8s_config
from kubeflow.tfserving import tf_job_client as tf_job_client_module
from IPython.core.display import display, HTML
import yaml
```

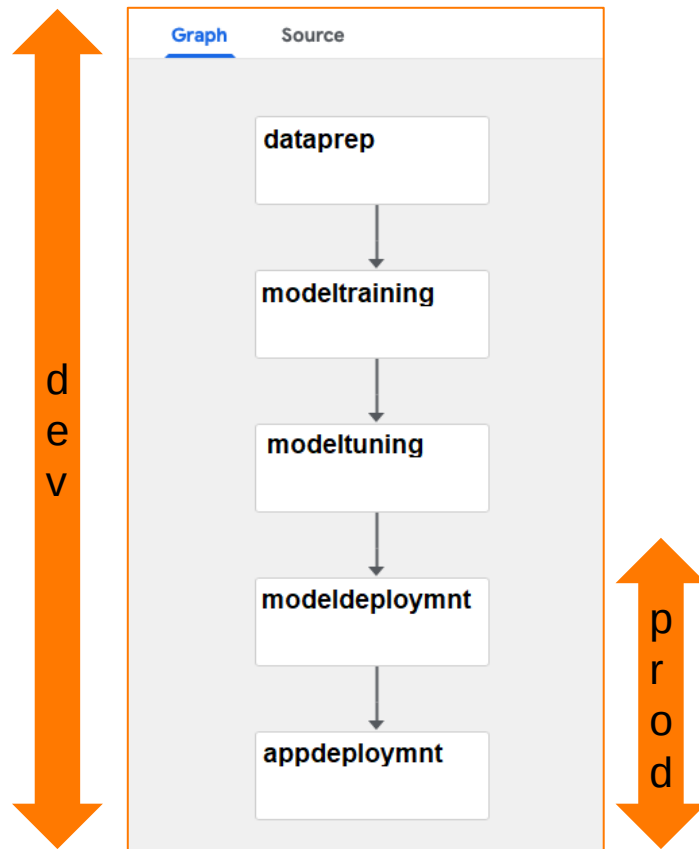
Configure a Docker registry for Kubeflow Fairing

- In order to build Docker images from your notebook, you need a Docker registry to store the images.
- Below you set some variables specifying a [Container Registry](#).
- Kubeflow Fairing provides a utility function to guess the name of your GCP project.

Pipelines

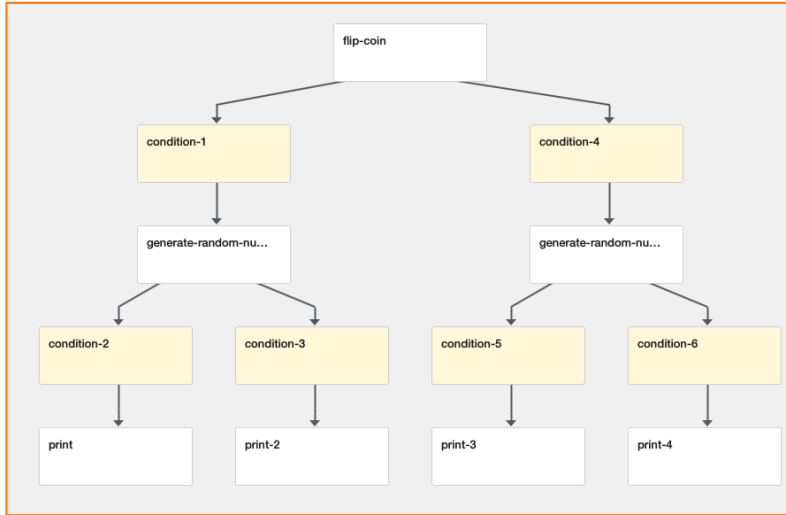
A *pipeline* is a description of an ML workflow, including all of the *components* in the workflow and how they combine in the form of a graph. (See the screenshot below showing an example of a pipeline graph). The pipeline includes the definition of the inputs (parameters) required to run the pipeline and the inputs and outputs of each component.

A *pipeline component* is a self-contained set of user code, packaged as a [Docker image](#), that performs one step in the pipeline. For example, a component can be responsible for data preprocessing, data transformation, model training, and so on.

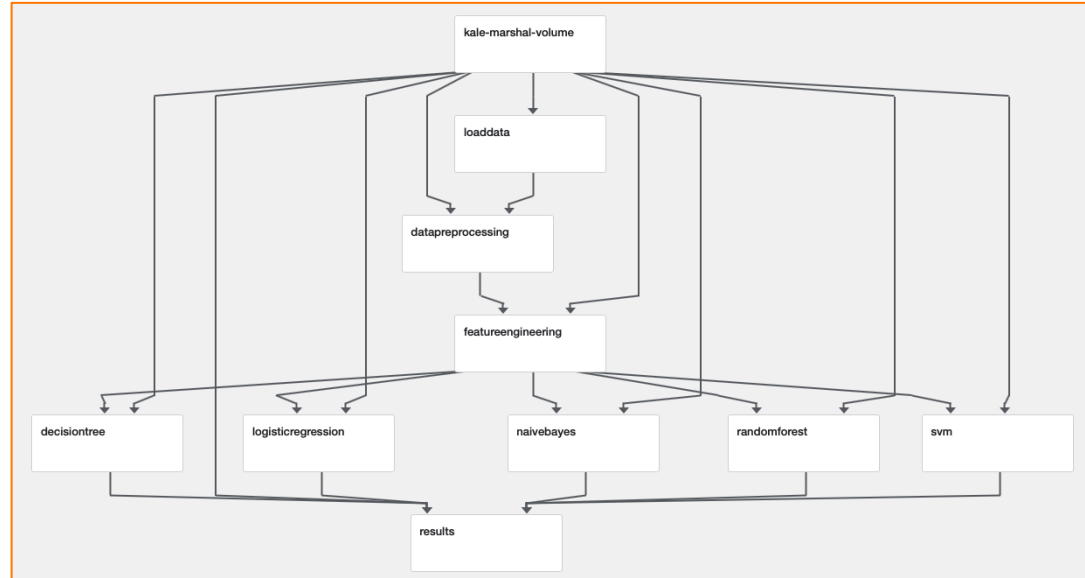


Pipelines – Conditions & parallelization

Conditions



Parallelization



KFServing

- Inference service

apiVersion: "serving.kubeflow.org/v1alpha2"

kind: "InferenceService"

metadata:

name: "flowers-sample"

spec:

default:

predictor:

tensorflow:

storageUri: "gs://kfserving-samples/models/tensorflow/flowers"

canaryTrafficPercent: 10

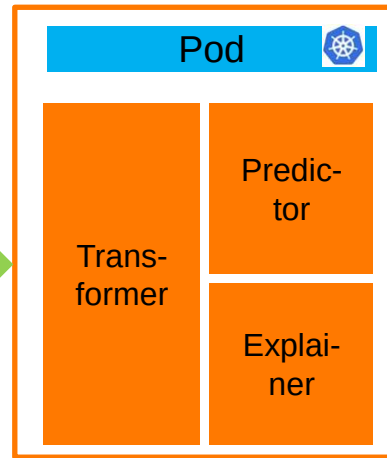
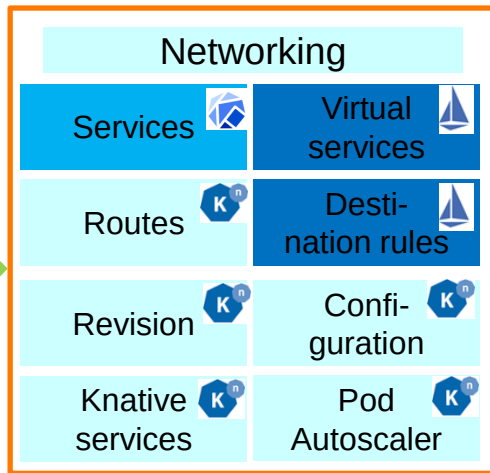
canary:

predictor:

10% of traffic is sent to this model

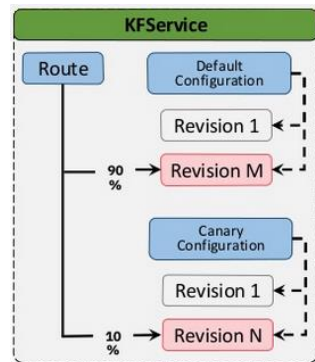
tensorflow:

storageUri: "gs://kfserving-samples/models/tensorflow/flowers-2"



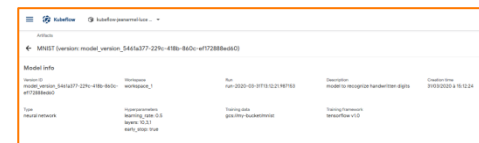
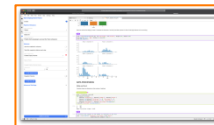
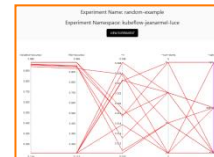
- Transformer + Explainer

- Canary deployment



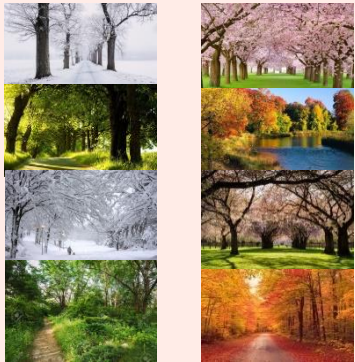
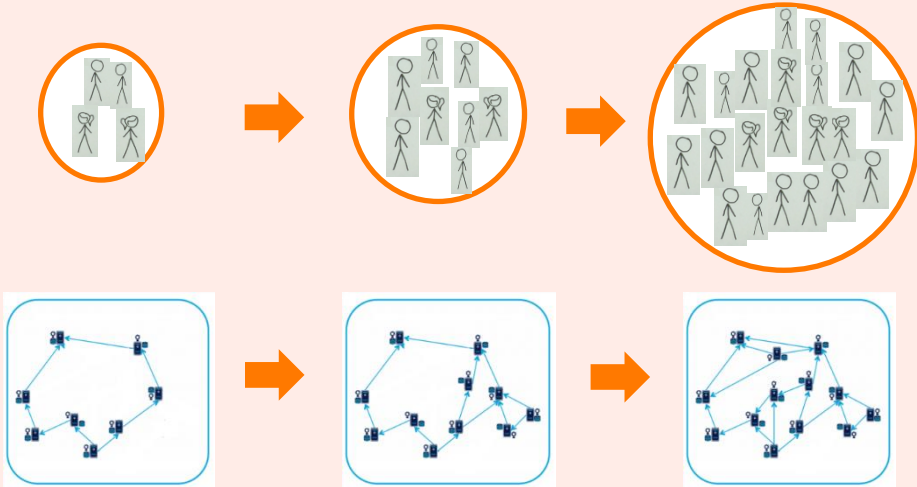
Other microservices or libraries

- Katib : hyperparameters tuning
- Kale : generation of pipelines
- Metadata : tracking and managing metadata of ML workflows
- etc ...



Model Info	Hyperparameters	Run	Description	Created At
Model ID: mnist_version_5461677-22fu-4f8b-84dc-vf7288a48c2	Hyperparameters: learning_rate: 0.1, batch_size: 128	Run ID: 2021-03-07T10:12:14Z	MNIST to tensorflow handwritten digits	2021-03-07 10:12:14
Type: neural network	Hyperparameters: learning_rate: 0.1, batch_size: 128	Training mode: supervised	Training framework: TensorFlow v2.3	

Kubeflow – Main dates

When	What
December 2017	Kubeflow was open sourced at Kubecon USA
	growth 
February 2020	Kubeflow 1.0 was announced to the public

Kubeflow versioning

- Kubeflow 1.0 published in Q1 2020
- Kubeflow embeds several applications with different levels of maturity
 - Each application has its own version id
- Starting from the release of Kubeflow v1.0, the Kubeflow community attributes *stable status* to those applications and components that meet a defined level of stability, supportability, and upgradability.

Give it a try !!! The easiest ways :

- Installation in a cloud. 3 ways :

1. kfctl : is the control plane for deploying and managing Kubeflow → deployment in ~30 minutes (AWS or GCP)
2. operator is currently in incubation phase
3. MiniKF (see below)



- Installation in your laptop : MiniKF

- Installation via vagrant + virtualbox
- Comes with : Minikube + Rok + Kubeflow



⚠ System requirements : 12GB RAM, 2CPUs, 50GB disk space mini

Kubeflow main actors / users



Conclusions

- Faster, easier, cheaper, secure → better TTM
- Provides applications that users love for training phase (notebooks, katib, ...) or serving (A/B testing, explainer, transformers, ...)
- Automation
- Very active open source community
- Still young
- Requires some skills (K8S) for maintenance

Q & A



obrigado

Dank U

Merci

mahalo

Köszí

спасибо

Grazie

Thank
you

mauruuru

Takk

Gracias

Dziękuję

Děkuju

danke

Kiitos