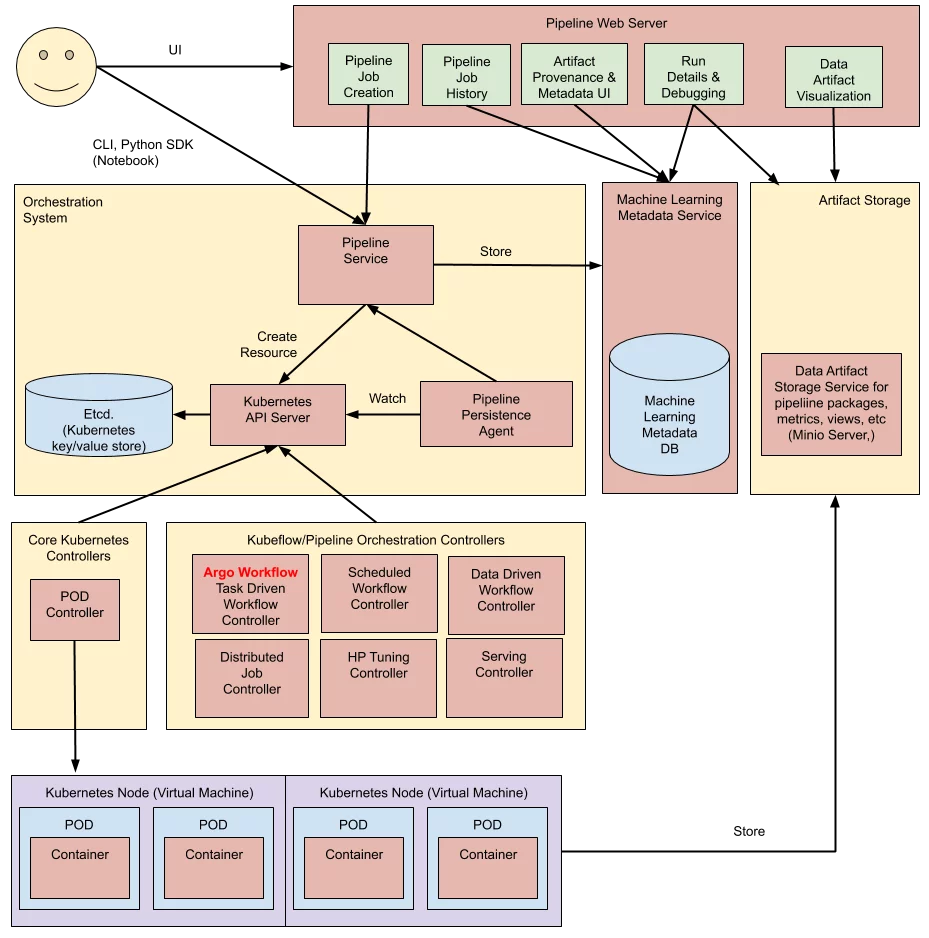
**Brief description of my infrastructure**

Brief description of the infrastructure, pipelines, and log with an example of API usage (for

example when you send a request to working infrastructure)

The following diagram illustrates how the Kubeflow Pipelines platform is structured. I borrow it from [run.ai](https://www.run.ai/guides/kubernetes-architecture/kubeflow-pipelines-the-basics-and-a-quick-tutorial/) to help me describe my infrastructure.

Figure 1 Kubeflow Pipeline platform https://26o9b711vqkt46183d39wgdp-wpengine.netdna-ssl.com/wp-content/uploads/2021/04/KF-image-1-pipelines-architecture.png.webp

My infrastructure can be described from bottom up as follows;

1. Cluster level iris-clf3

I have created a cluster called iris-clf3 with 2 worker nodes, using kind. I apply pods with 5 containers. (Kubenetes Nodes-Virtual machines with POD and Container in figure 1). I can exec into each node and each pod with kubectl (Kubernetes API Server). I also create a local registry at localhost:5000 to store images (Etcd Kubernetes key/value store).

kubectl

2. Namespace (Orchestration System)

Now I have pods that are working, I need to connect them together. To make use of pipelines, I install kubeflow via kustomize as my Pipeline Service in figure1. I set-context so that my namespace is kubeflow, then I can port-forward from kubectl to kubeflow and access the ui (Pipeline Job Creation in figure 1) at localhost:8080. I initiated a run of myfirstpipeline for the training task and created myinferencepipeline. They are stored in /mlpipeline.

--namespace, -n kubeflow

3. Service

To save the work done by the pods and containers, I deployed a database iris-clf-MySql via kustomize(Artifact storage in figure 1) . There are 2 Services; the persistentVolume itself named MySql-for-iris and the MySql client/frontend persistentVolumeClaim named iris-says-POST .

*/kustomize*/kustomization-MySql.yaml

4. serviceAccount

For monitoring and visualization, I set up a Data Artifact visualization (as in figure1) with Prometheus and Grafana as outlined. Kubernetes API server (kubectl) was started with kube-apiserver command. Then I deployed Promotheus operator/serviceAccount at the cluster. Pods, and services to be added.

/operator\_k8s/prometheus\_rbac.yaml, ImpactTech/prometheus.yaml

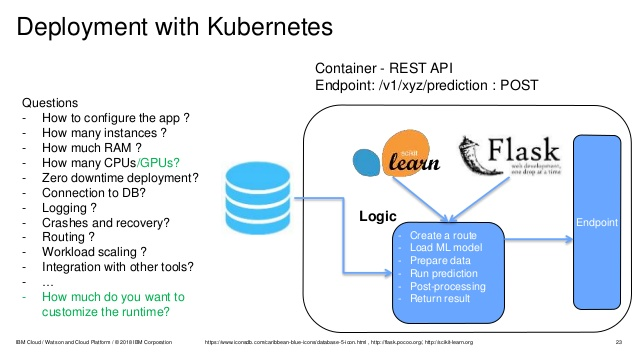
kube-apiserver

5. predictionAPI in a separate namespace (-n ai\_services)

For this part of the workflow, I would like to illustrate it with Figure 2 below. Assuming that the left hand side database is the same one as Artifact Storage from Figure 1. This means starting from the storage, I will use the services created in 3. to connect the database into the pods. Then with myinference pipeline created in 2., it runs the works specified in myfirstpipeline (training and optimize) and return the prediction. The prediction got sent to the Endpoint whenever there is a POST request.

I can exec into the Container REST API and curl the Endpoint to get the prediction.

The submission is work-in-progress.

Figure 2 Deployment of ML models into production with Kubernetes https://www.slideshare.net/Hadoop\_Summit/how-to-deploy-machine-learning-models-into-production