Required installations for the Project to Run:  
  
Install DockerDesktop  
Docker Desktop → Settings → Kubernetes → Enable Kubernetes.  
choco install kubernetes-cli  
choco install kubernetes-helm  
pip install prefect  
Install Python >=3.9  
  
Step-1  
  
**Goal:** create a small project that runs 3 parallel training trials locally with Prefect, writes models & metrics to ./artifacts, and shows Prefect task logs. This validates your Prefect + Python environment before we dockerize / run in K8s.

python -m venv .venv  
.\.venv\Scripts\Activate.ps1  
  
pip install --upgrade pip  
python -m pip install --upgrade pip  
  
python .\flow.py  
  
dir .\artifacts  
type .\artifacts\metrics\_t1.json  
type .\artifacts\metrics\_t2.json  
type .\artifacts\metrics\_t3.json

docker build -t ml-prefect-flow:latest .  
docker run --rm -it ml-prefect-flow:latest  
  
python deploy\_flow.py  
prefect agent kubernetes start --namespace prefect  
prefect deployment run "ml-flow-k8s"  
kubectl get pods -n prefect

**Multiple hyperparameter experiments** can run simultaneously on a cluster.

**Prefect orchestrates** the task scheduling.

**Kubernetes provides** isolation, scalability, and resource control.

Metrics and models are collected automatically in artifacts/ for downstream processing.

┌─────────────────────────┐

│ Prefect Flow.py │

│ (local\_async\_ml\_flow) │

└─────────────┬───────────┘

│

Fan-out tasks in parallel

│

▼

┌──────────────┬──────────────┬──────────────┐

│ train\_model │ train\_model │ train\_model │

│ t1 │ t2 │ t3 │

└───────┬──────┴───────┬──────┴───────┬──────┘

│ │ │

Prefect Kubernetes Agent schedules tasks

│ │ │

▼ ▼ ▼

┌──────────┐ ┌──────────┐ ┌──────────┐

│ Kubernetes│ │ Kubernetes│ │ Kubernetes│

│ Pod t1 │ │ Pod t2 │ │ Pod t3 │

└──────┬───┘ └──────┬───┘ └──────┬───┘

│ │ │

▼ ▼ ▼

Train RandomForest Train RandomForest Train RandomForest

Save model + metrics Save model + metrics Save model + metrics

│ │ │

└───────────┬─────┴───────┬────────┘

▼

Artifacts folder / Prefect logs

What is Perfect?  
  
Prefect is a workflow orchestration tool for data and ML pipelines. Unlike Airflow, Prefect provides:

* **Simpler Python-native API**
* **Asynchronous task execution**
* **Better handling of retries, failures, and dynamic flows**
* Native **Kubernetes, Docker, and cloud integration**

**How does Prefect handle asynchronous execution?**

Prefect’s .submit() method schedules tasks for execution **without blocking** the main flow.

Tasks can run **in threads, processes, or remote agents** (like Kubernetes pods).

**Synchronous:** Tasks execute one after another; each wait for the previous to finish.

**Asynchronous:** Tasks can run in parallel or concurrently without waiting, enabling **faster execution and scalability**, especially for independent ML experiments.

**How is Kubernetes used in an ML pipeline?  
  
 Each ML task (training, preprocessing, validation) runs as a separate pod.**

**Prefect acts as an orchestrator, telling Kubernetes which pod to run and when.**

**Benefits: isolation, scalability, parallelism, resource management.  
  
 Containerize the ML code (Docker).**

**Use Kubernetes to run multiple experiments in parallel.**

**Use Prefect for orchestration and scheduling.**

**Store metrics/models centrally for monitoring and downstream pipelines.**

**Prefect supports automatic retries, state handling, and logging.**

**Failed tasks can be rescheduled without affecting the entire flow.**

**Q7. How do you deploy a Prefect ML pipeline on Kubernetes?  
  
Dockize the ML code (Flow.py) with dependencies.**

**Push Docker image to a registry.**

**Create a Prefect deployment using Kubernetes Job infrastructure.**

**Start a Prefect Kubernetes agent.**

**Run the flow; Prefect schedules tasks as pods.**

**How is this setup used in real-time production?**

**Run multiple model training experiments in parallel.**

**Automate retraining when new data arrives.**

**Isolate resource-intensive ML jobs without affecting other services.**

**Centralize monitoring/logging with Prefect Cloud or Server.**