SWEMLS: Coursework 4: Running inference on Kubernetes

Google docs version of this document

Background

Our real production environment will use Kubernetes, so in this coursework, we'll deploy our solution from coursework 3 onto a Kubernetes cluster, running on Microsoft's Azure cloud service. We'll push a docker image of our code to a private container registry for the course, and use it to create a kubernetes deployment.

All authentication is via your standard Microsoft college account. Replace `group-name` with the name of your group, eg `serreria`. You won't be able to see, or change, the Kubernetes resources created by other groups.

The simulator you'll connect to when running on Kubernetes is the same simulator we've been using to assess coursework 3, but configured to send messages in real time, rather than as quickly as you can read them. It will still replay messages from the beginning each time you connect to it. If you attempt to connect to it twice, the older connection will be closed.

Tasks

(note long command lines are wrapped in the doc, but are a single line)

- Install the Azure command line interface, `az`
- Install the Kubernetes command line interface, `kubectl`
- Login to azure:
  - `az login`
- Use the course's subscription:
  - `az account set --subscription 4693832c-ac40-4623-80b9-79a0345fcfce`
- Login to our private container registry:
  - `az acr login --name imperialswemlsspring2025`
- Setup kubectl to talk to our Kubernetes cluster:
  - `az aks get-credentials --resource-group imperial-swemls-spring-2025 --name imperial-swemls-spring-2025 --overwrite-existing`
  - `kubelogin convert-kubeconfig -l azurecli`
- At this point, you should be able to verify that you don't yet have anything running on Kubernetes.
  - `kubectl --namespace=group-name get pods`

If everything is going to place, you'll see:

"`No resources found in group-name namespace`"

- Build a docker image for your solution, and give it a tag that includes your group name, and refers to our private repository:
  - `docker build -t imperialswemlsspring2025.azurecr.io/coursework4-group-name .`
- Push the image to the repository:
  - `docker push imperialswemlsspring2025.azurecr.io/coursework4-group-name`
- Adapt the configuration file below, and save it as `coursework4.yaml`. Use it to create a Kubernetes deployment, which will create a pod to run your code.
  - `kubectl apply -f coursework4.yaml`
- If everything has gone according to plan, you'll be able to see your deployment listed with:
  - `kubectl --namespace=group-name get deployments`
- ..and you'll be able to see the output of your code with:
  - `kubectl logs --namespace=group-name -l app=aki-detection`

Configuration example

Here's an example Kubernetes configuration file that:

- Allocates persistent storage for your code that will survive restarts
- Creates a deployment that:
  - Copies the hospital's history.csv from another docker image into the filesystem your code will see, as `/data/history.csv`
  - Mounts the persistent storage as `/state`

You will need to change (at least) the `command` and `args` part of the configuration file. You'll need to make sure that any data you want to keep between restarts is written `/state`. You should keep the `MLLP_ADDRESS` and `PAGER_ADDRESS` parts. Replace all the parts that say *GROUP_NAME* with your group's name.

Notes

- The architecture of the kubernetes cluster nodes is exclusively amd64. If you build docker images on a different architecture (for example, an M1 or M2 mac, known as arm64), you'll need to tell docker to build images for that architecture. That should be as simple are using:
  - `docker build --platform=linux/amd64,linux/arm64`

  in your build command. The amount of magic that triggers should not be underestimated.
- If you're logging from Python to standard output (for example, by using `print` function calls), you may not see any log output until your code has written a significant amount of output - which may be never - as Python buffers the output. You can disable this by setting

an environment variable `PYTHONUNBUFFERED=1` in your container. See the python documentation.

Marking

You'll achieve 100% of the marks for this coursework by consuming any messages at all in the Kubernetes environment. You don't need to submit anything - we'll detect it.