Instructions

- Solve the problems on the following page. Where applicable use python.
- Feel free to use an external library of your choice.
- There is no time limit or necessary right answer. You do not need to complete it, but make a solid effort so that we can talk about what you have done in a meaningful way!
- We are looking at the quality of your solution, the approach used and key decisions you made. Make sure you can explain it when asked.
- Feel free to ask questions and use the team here at Vector as a resource / sounding board. Assume you are working here and this is your project. How would you go about it?
- What we are interested in seeing is the way you think. Create a repository on gitlab / github and share it with us.
- Most importantly, enjoy it!

Problem

Part 1

- Build a multi-class image classifier on the fashion MNIST dataset using a Convolutional Neural Network (CNN) based model.
- Your library will need to be flexible enough to train an image classifier on a different dataset and be written in a way to allow anyone to use it. We will try to run it internally on our own dataset using instructions you provide. Make it as easy as possible for us to use it.
- Don't spend too long trying to push the accuracy up.
- Feel free to use open source code / guidelines.

Part 2

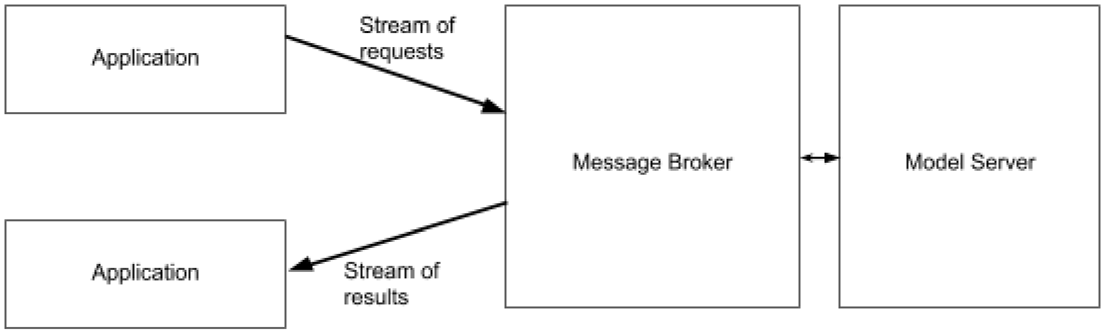
- Build a unified API in python to send and receive messages to / from Apache Kafka and Google Pub/Sub.
- You will have to choose the appropriate client libraries.
- The inputs to the function and the outputs should be as unified as possible.

Part 3

- Let's do something real world now! We have multiple machine learning services that are coordinated via a message broker.

  Here, you have to design and build a robust system to classify fashion images. (Here, we can use the fashion MNIST validation set to mock the input) The system will have a single client consuming a single machine learning service.

- Use the model from Part 1 and the library from Part 2 to build such an application. It does have to be robust, scalable and able to process requests asynchronously.
- Note that this is not a REST API based system but rather one which can process requests in a non-blocking way and (theoretically) put the results somewhere else (like a database). You can mock this by printing to the console.
- Look below for a simple schematic


Application
Stream of requests
Message Broker
Model Server
Application
Stream of results