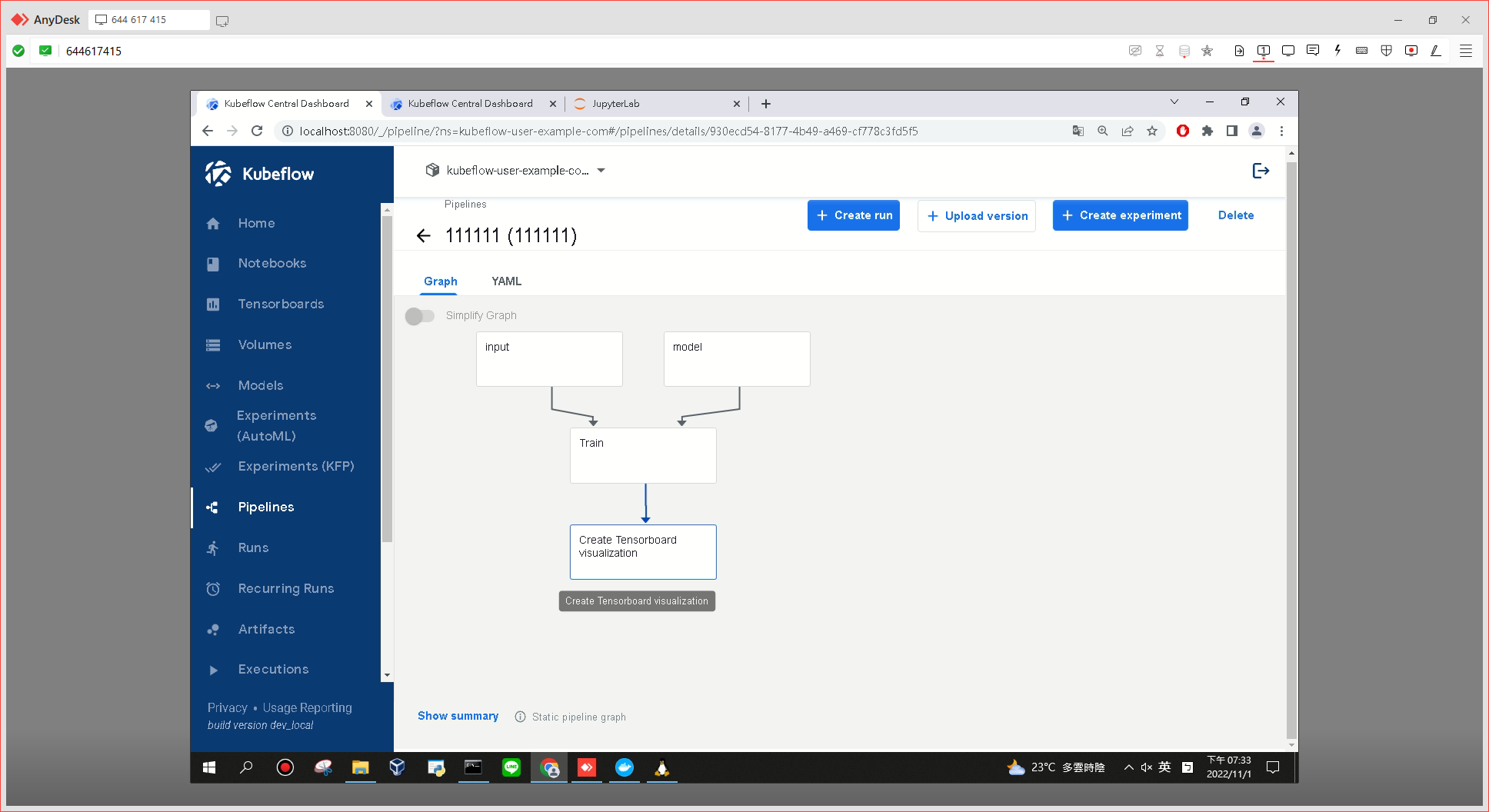
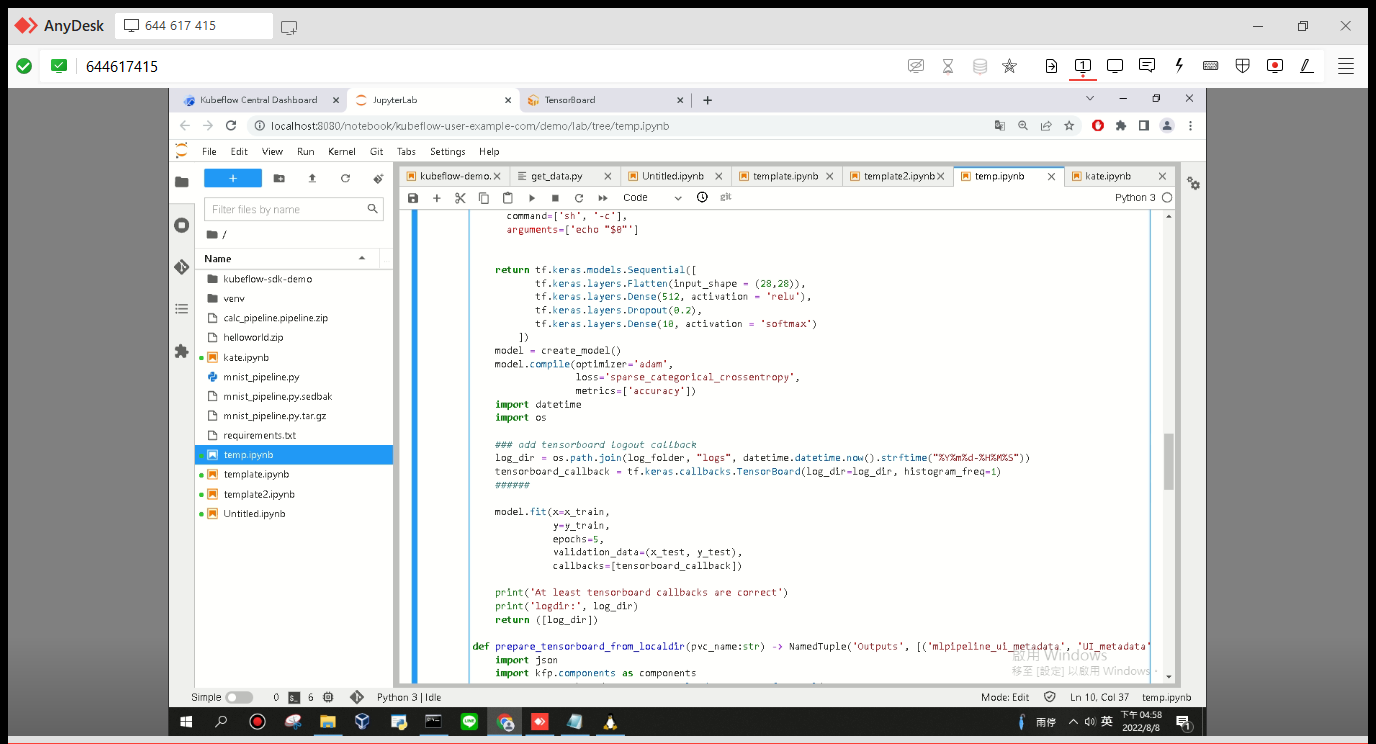
Template mnist example

開發管道之後，可以在Kubeflow管道UI上上傳並共享它。Pipelines組件是一組獨立的用戶代碼，打包為Docker映像，它們在管道中執行一個步驟。例如，一個組件可以負責數據預處理，數據轉換，模型訓練等，實現為 Kubernetes CRD（自定義資源定義）。 因此工作流可以使用 kubectl 進行管理，並與其他 Kubernetes 服務（如volumes, secrets,和 RBAC）進行整合。

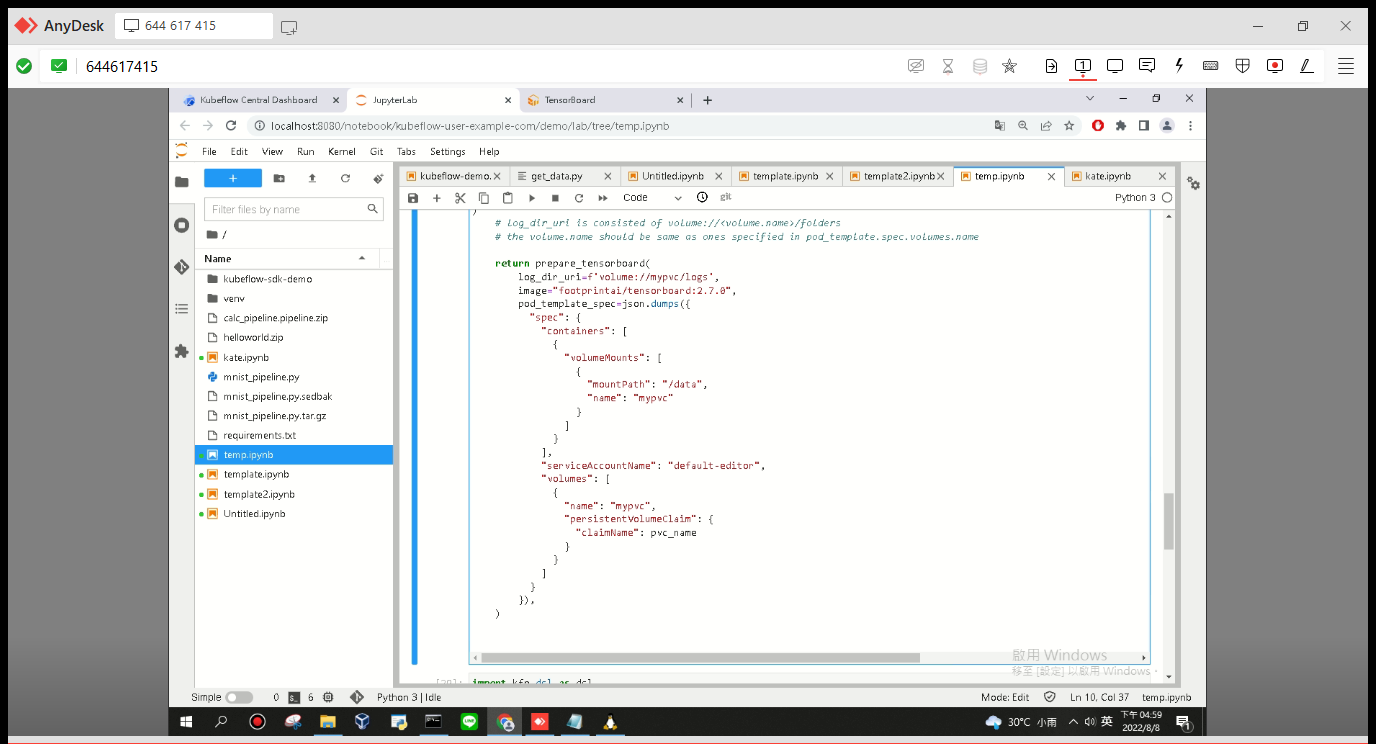
1. 建立出的管道圖，利用範例手寫辨識，將keras模型分成兩個components，分成模型和訓練部分，整個pipeline流程圖分為建立tensorboard、模型輸入層、數據訓練的部分。
2. 建立流程為先從pvc提取訓練數據，匯入模型內進行訓練(input)，藉由keras模型(model)來和輸入模型進行訓練，進行5次迭代來做基本的手寫辨識處理(mnist func)，最後再給數據到下一個components做可視化圖像來查看訓練結果、模型完整度(create tensorboard visualization)。



1. 利用模型的輸入層來做數據訓練，透過管道的數據傳輸，將所訓練的數據傳遞至下一個components，提取keras的完全連接層(Dense)，越多神經元迴歸線的特徵數越多，預測越準確。

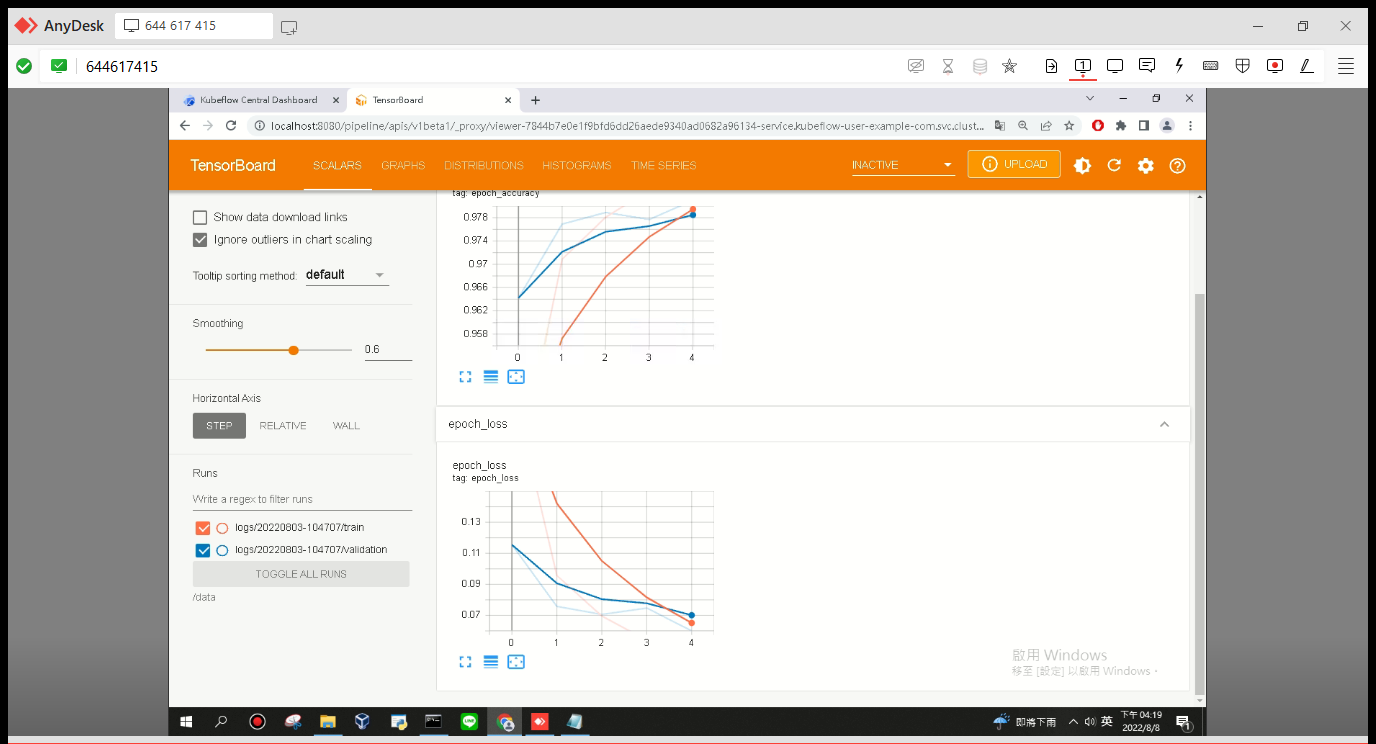
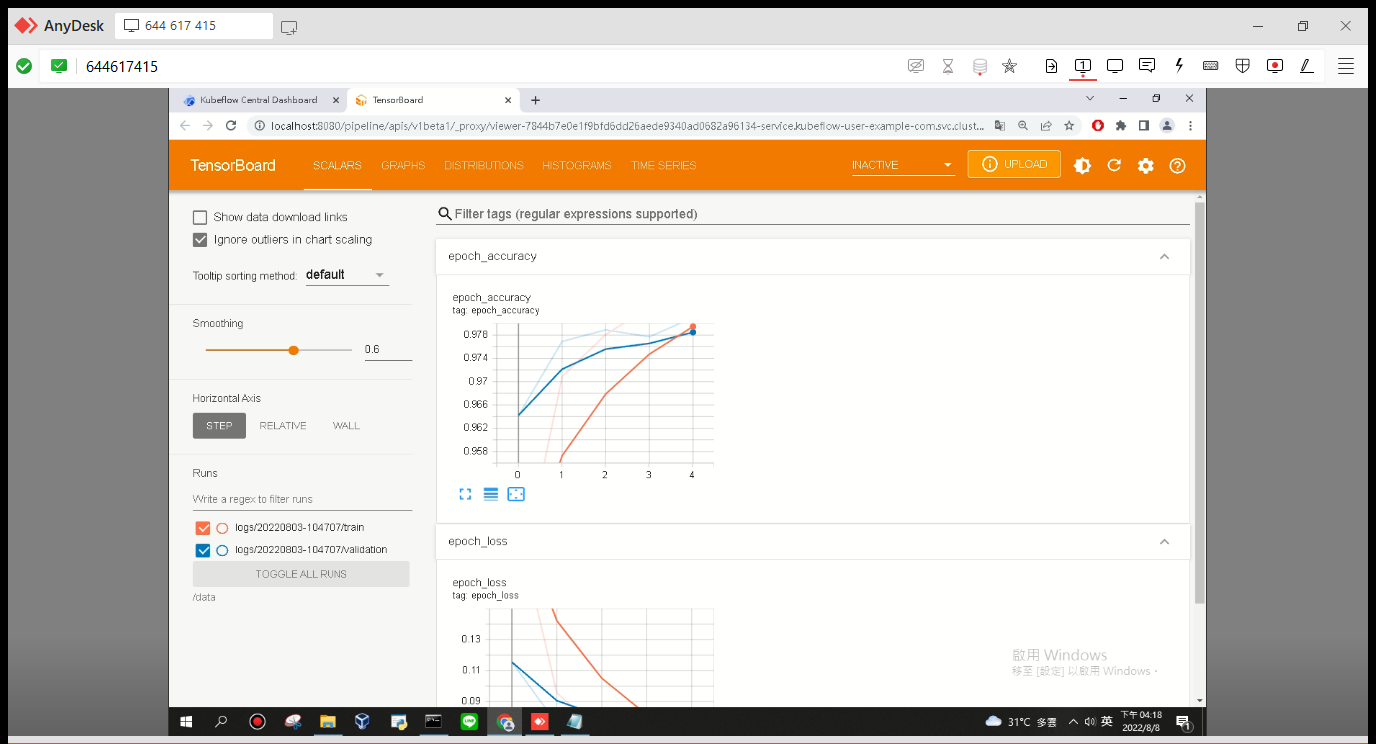


1. 輸入的數據，從image中提取數值，將data和所需套件、參數存入pvc裡，從整個工作流程來看，會將得到的數據透過管線呈現的方式傳遞給mnist func components進行數據的訓練。

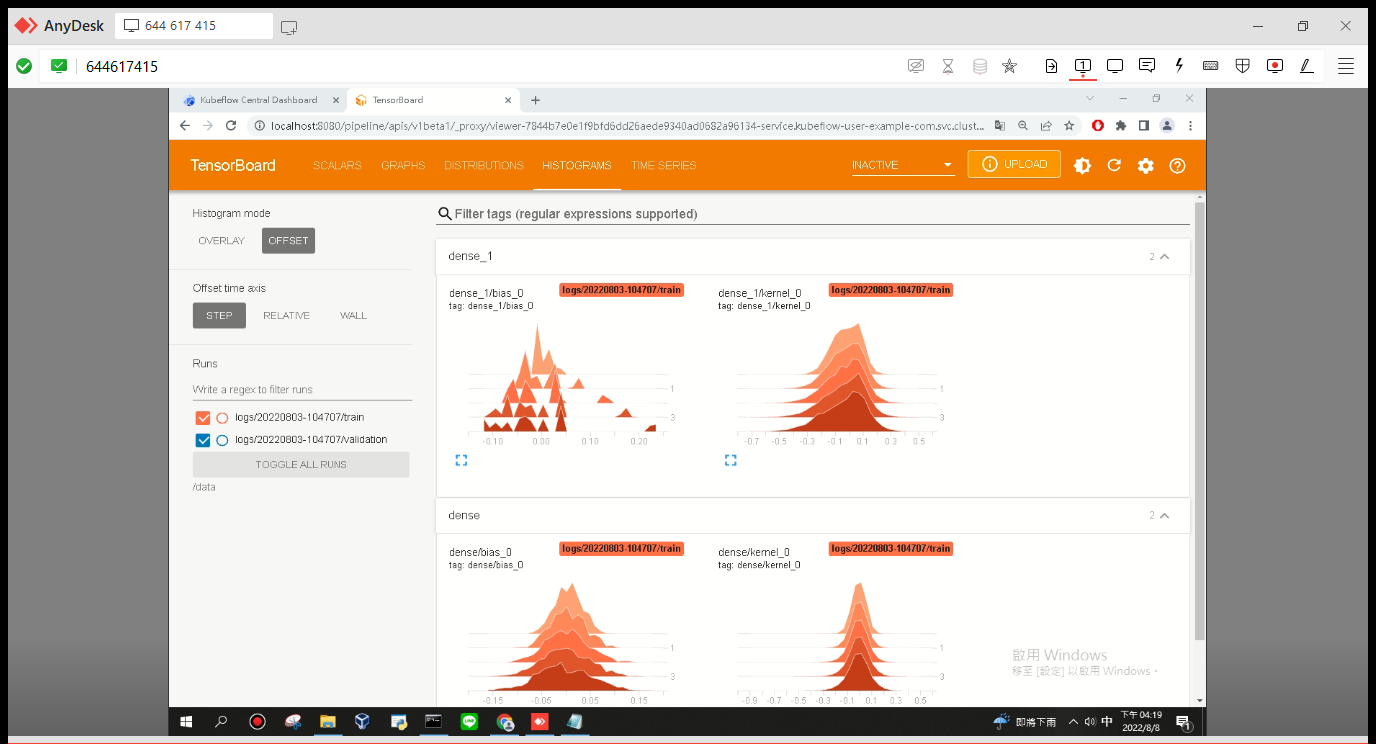


1. 模型運行完的結果輸出，利用Tensorboard查看圖表，為了輸出本地資料，需要先新增viewer來取代舊的，在Yaml修改Metadata為我現在的id，為讓kubeflow可以找到並運行現在新的資料，執行 TensorBoard 會在本機的 8080 連接埠開啟一個網頁伺服器。在設計pipeline components時添加tf.keras.callbacks.TensorBoard回調可確保創建和存儲，--logdir 所指定的目錄就在 TensorFlow 程式中使用寫入資料的目錄。

* Scalars:顯示Accuracy和Loss。模型迭代5次，Accuracy藍色為訓練集的準確率，橘色為驗證集的樣本，驗證集的Loss 是從第1個epoch 才開始記錄的，並且驗證集的Loss 是所有驗證集樣本的Loss 均值，所以曲線更加平滑，而訓練集的Loss 是batch size 的數據，因此震盪幅度較大。



* Histograms:其中一層不同時間顯示張量的直方圖，每一個epoch 的梯度呈正態分佈，權值分佈好，梯度接近於0，模型收斂很快。前後網絡層的梯度都很小，因Loss 很小且沒有下降，模型已經接近收斂。



* 以pipeline components建立yaml檔輸出至kubeflow，供k8s做容器編排部屬，在本地端利用minikube CLI查看pod 的logs。

