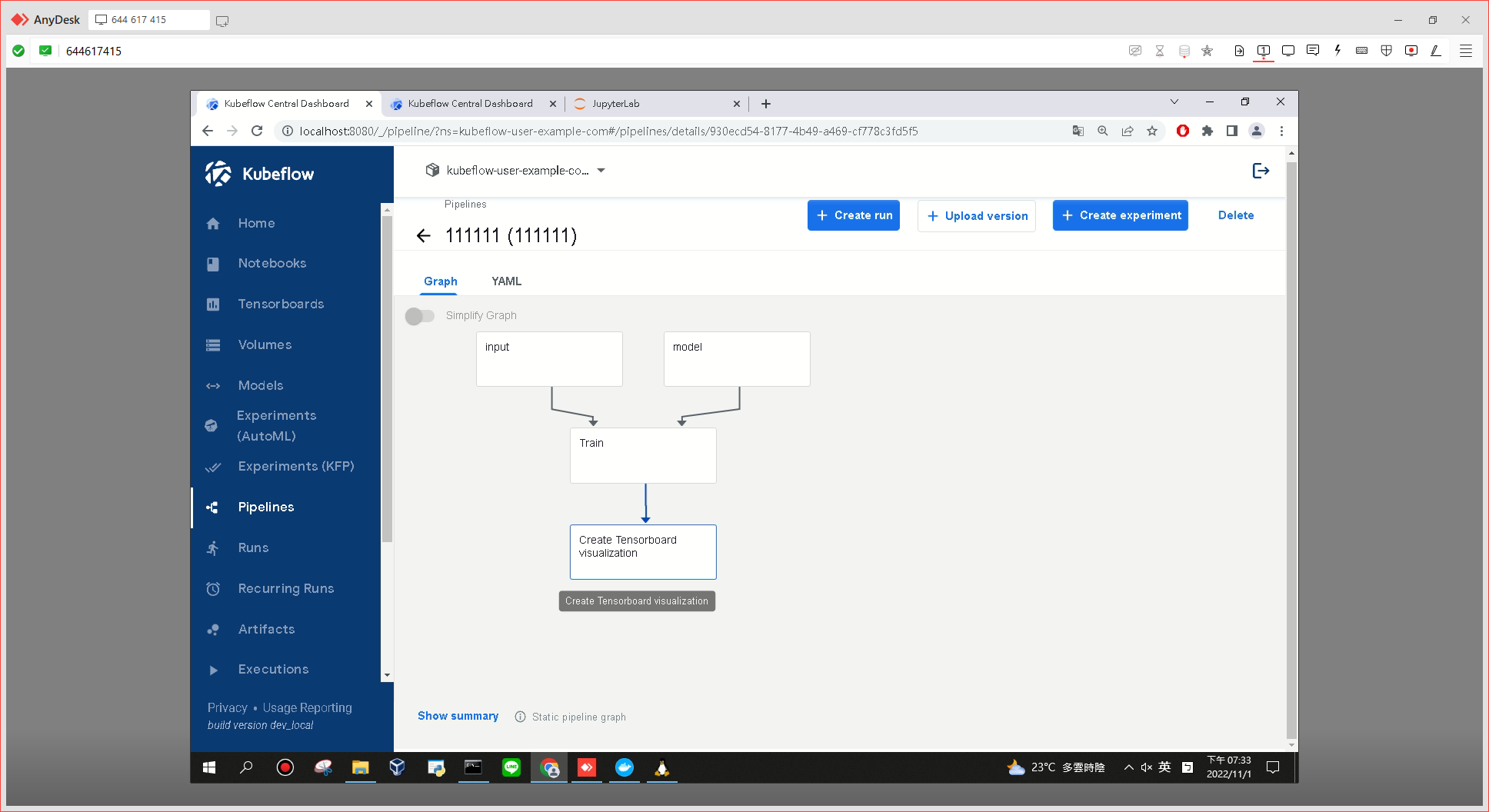
Pipeline template mnist example

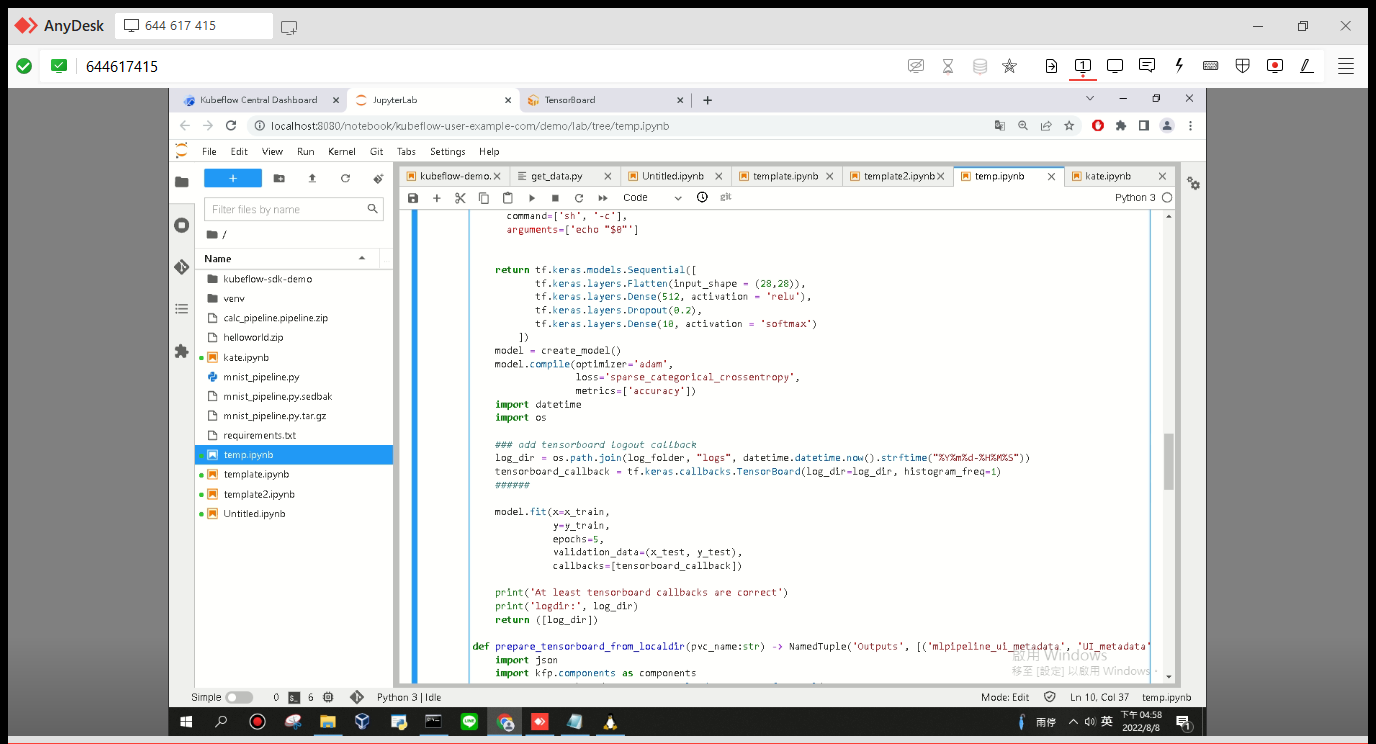
After the pipeline is developed, it can be uploaded and shared on the Kubeflow pipeline UI. A Pipelines component is a set of independent user code, packaged as a Docker image, that executes pipeline in steps. For example, a component can be responsible for data preprocessing, data transformation, model training, etc., implemented as a Kubernetes CRD (Custom Resource Definition). So workflows can be managed using kubectl and integrated with other Kubernetes services such as volumes, secrets, and RBAC.

1. The established pipeline diagram, using the example handwriting recognition, divides the keras model into two components, model and training parts. The entire pipeline flow chart is divided into the parts of building tensorboard, model input layer, and data training.

2. The establishment process is to first extract training data from pvc, import it into the model for training (input), use the keras model (model) to train with the input model, and perform 5 iterations to do basic handwriting recognition processing (mnist func ), and finally visualize the data to the next components to view the training results and model integrity (create tensorboard visualization).



3. Use the input layer of the model for data training, transfer the trained data to the next components through the data transmission of the pipeline, and extract the fully connected layer (Dense) of keras. The more neurons, the more features of the regression line. The more accurate the prediction is.

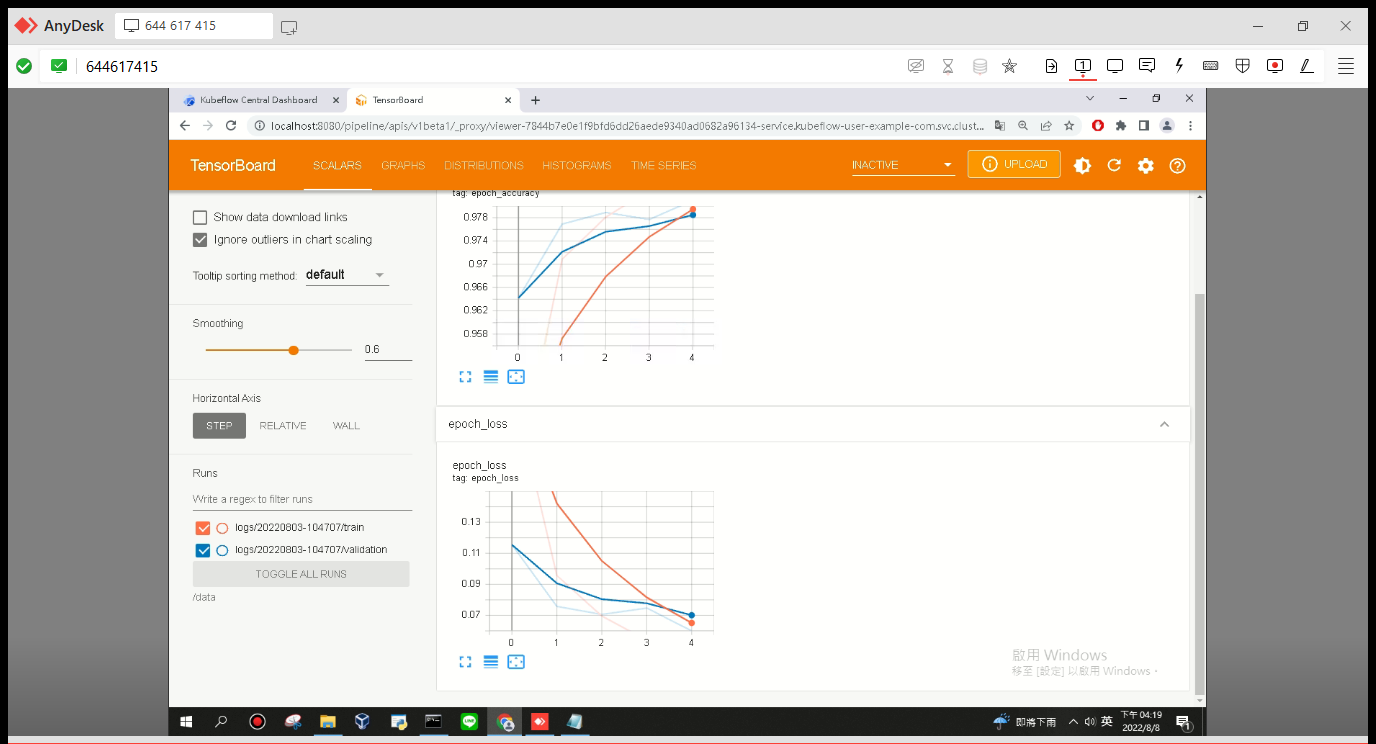
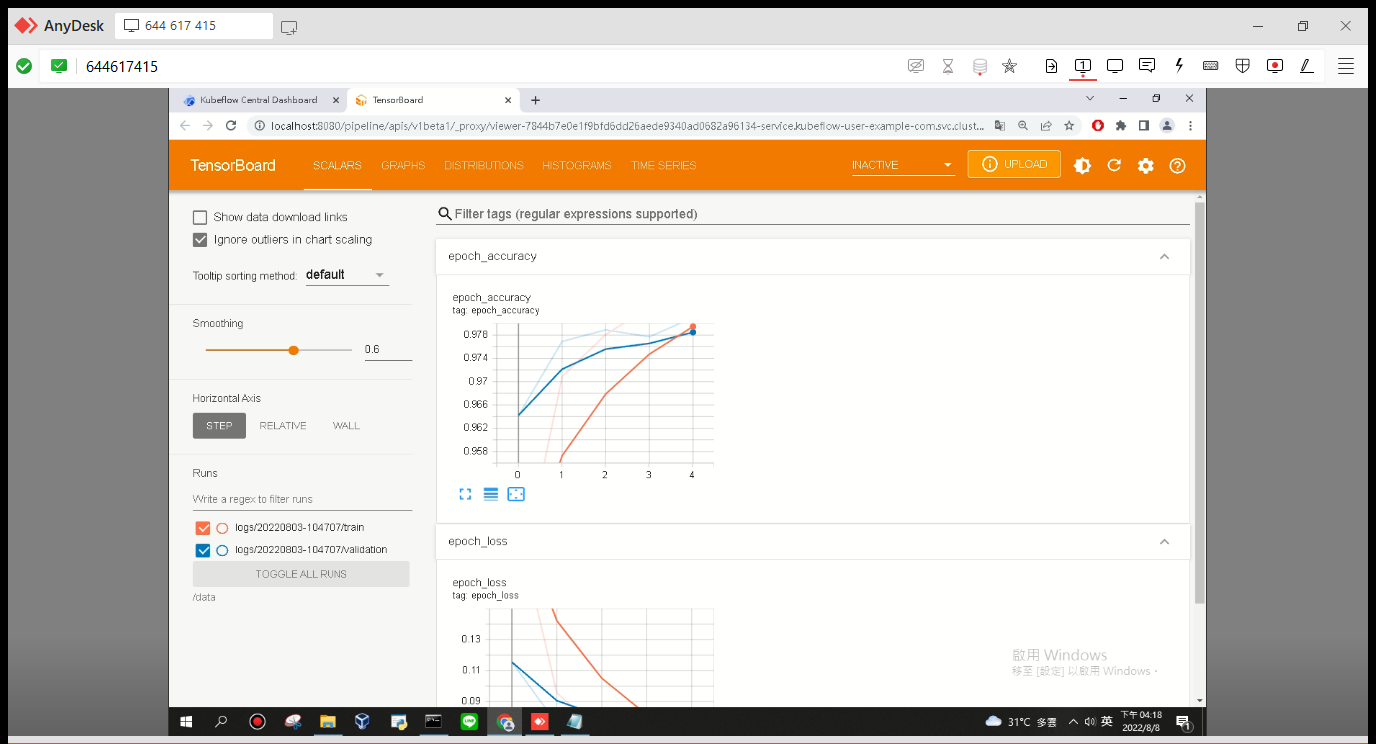


4. For the input data, extract the value from the image, and store the data and the required packages and parameters in the pvc. From the perspective of the entire workflow, the obtained data will be passed to the mnist func components for data processing by way of pipeline presentation.

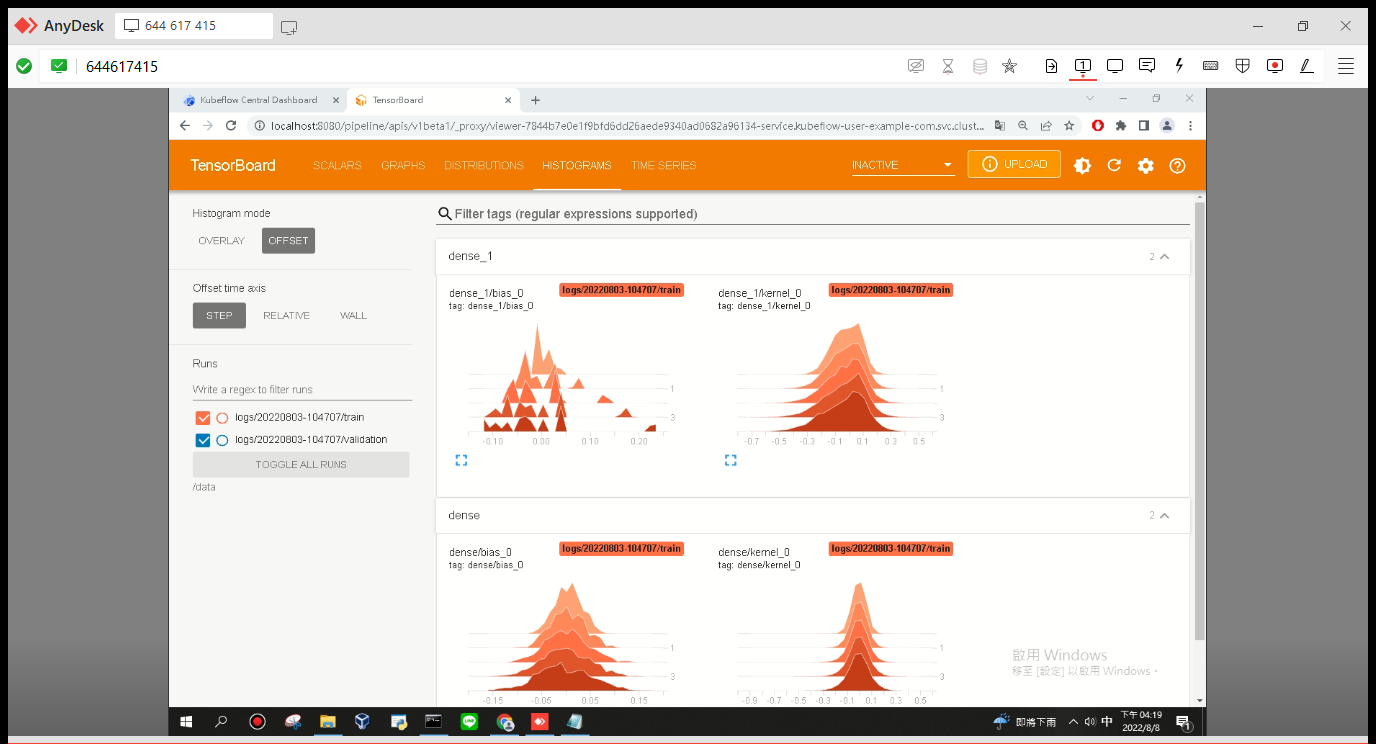
5. After the model runs, use Tensorboard to view the graph. In order to output local data, you need to add a new viewer to replace the old one. Change the Metadata in Yaml to my current id, so that kubeflow can find and run the new data. , running TensorBoard will open a web server on the local 8080 port. Adding the tf.keras.callbacks. TensorBoard callback when designing the pipeline components ensures that the directory specified by --logdir is the directory where the data is written in the TensorFlow program.

5. After the model runs, use Tensorboard to view the graph. In order to output local data, you need to add a new viewer to replace the old one. Change the Metadata in Yaml to my current id, so that kubeflow can find and run the new data. , running TensorBoard will open a web server on the local 8080 port. Adding the tf.keras.callbacks.TensorBoard callback when designing the pipeline components ensures that the directory specified by --logdir is the directory where the data is written in the TensorFlow program.

 Scalars: Display Accuracy and Loss. The model iterates 5 times, Accuracy blue line is the accuracy of the training set, orange is the sample of the validation set, the Loss of the validation set is recorded from the first epoch, and the Loss of the validation set is the Loss of all the samples of the validation set The mean value, so the curve is smoother, and the Loss of the training set is the data of the batch size, so the fluctuation range is large.



 Histograms: One of the layers displays histograms of tensors at different times. The gradient of each epoch is normally distributed, the weight distribution is good, the gradient is close to 0, and the model converges quickly. The gradients of the front and rear network layers are very small. Because the Loss is small and does not drop, the model is close to convergence.



 Use the pipeline components to create a yaml file and output it to kubeflow, for k8s as a subordinate to the container orchestration, and use the minikube CLI to view the pod logs on the local side.