Seagen®
quantiphi
Solving What Matters

SeaGen - IHC Visualization, TMA Orchestration & Pathology Scoring Model Migration

Technical Design Documentation

Seagen®
quantiphi
Solving What Matters

0 Table of Contents

1 Business Overview

1.1 Objective of the Project

Original Scope:

1. Augment Seattle Genetics' existing pipeline to include the automatic generation of Immunohistochemistry (IHC) visualization images to improve their scientists ability to perform quality checks on the AutoML Vision model.
2. Automate the process of tissue core detection in a Tissue MicroArray (TMA) sample.

Extended Scope:

- The goal is to migrate the Automated Pathology Scoring(APS) application and supporting applications from Azure to GCP to have all the automated workflows relating to pathology scoring.
    - Ingestion
    - Pre-processing
        - Tiling, Quality Check, Tissue core de-arraying, etc
- Pathology scoring of input images will be followed by model inferencing and Output visualization
- Implementation to be hosted within the GCP environment.

1.2 Business Outcome

The end goal would be to have end-to-end automated pipelines to interconnect all ingested data, where the images ingested will be identified and rerouted to proper buckets for classification, quality control, model inference, and visualization.

1.3 Solution Used

Quantiphi automated the generation of Immunohistochemistry (IHC) visualization images and the process of tissue core detection in a Tissue MicroArray (TMA) sample. Quantiphi also migrated the automated pathology scoring application and supporting applications from Azure to GCP and hosted all the automated workflows (relating to pathology scoring) on GCP environment.

2 Cloud Environment

2.1 Services Used

Resource	Description
Cloud Storage	• Cloud Storage is a reliable and secure object storage service that allows world-wide storage and retrieval of any amount of data at any time.
Dataflow	• Dataflow is a managed service for executing a wide variety of data processing patterns.
Vertex AI	• Vertex AI brings AutoML and AI Platform together into a unified API, client library, and user interface. With Vertex AI, both AutoML training and custom training are available options. Whichever option you choose for training, you can save models, deploy models and request predictions with Vertex AI.
BigQuery	• BigQuery is Google Cloud's fully managed, petabyte-scale, and cost-effective analytics data warehouse that lets you run analytics over vast amounts of data in near real time.
Cloud Source Repository	• Cloud Source Repositories are fully featured, private Git repositories hosted on Google Cloud.
Virtual Private Cloud(VPC)	• A Virtual Private Cloud (VPC) network is a virtual version of a physical network, implemented inside of Google's production network, using Andromeda.

Cloud Build	• Cloud Build is a service that executes your builds on GCP.. Cloud Build can import source code from Cloud Source Repository and execute a build to your specifications, and produce artifacts such as Docker containers or Java archives.
Cloud Functions	• Cloud Functions is a lightweight compute solution for developers to create single-purpose, stand-alone functions that respond to Cloud events without the need to manage a server or runtime environment.

TABLE 1 - Cloud Environment Services used in the Project

2.2 Costs

June 2021 - $2829.00
July 2021 - $4770.00
August 2021 - $8002.40 (Up until August 19, 2021)

Total Cost **~ $15601.00 (Up until August 19, 2021)**

2.3 Data Access and Security

Within Seagen's Google Cloud environment, we created secure access roles with access to services required.

Below is an infrastructure diagram to illustrate the security components.

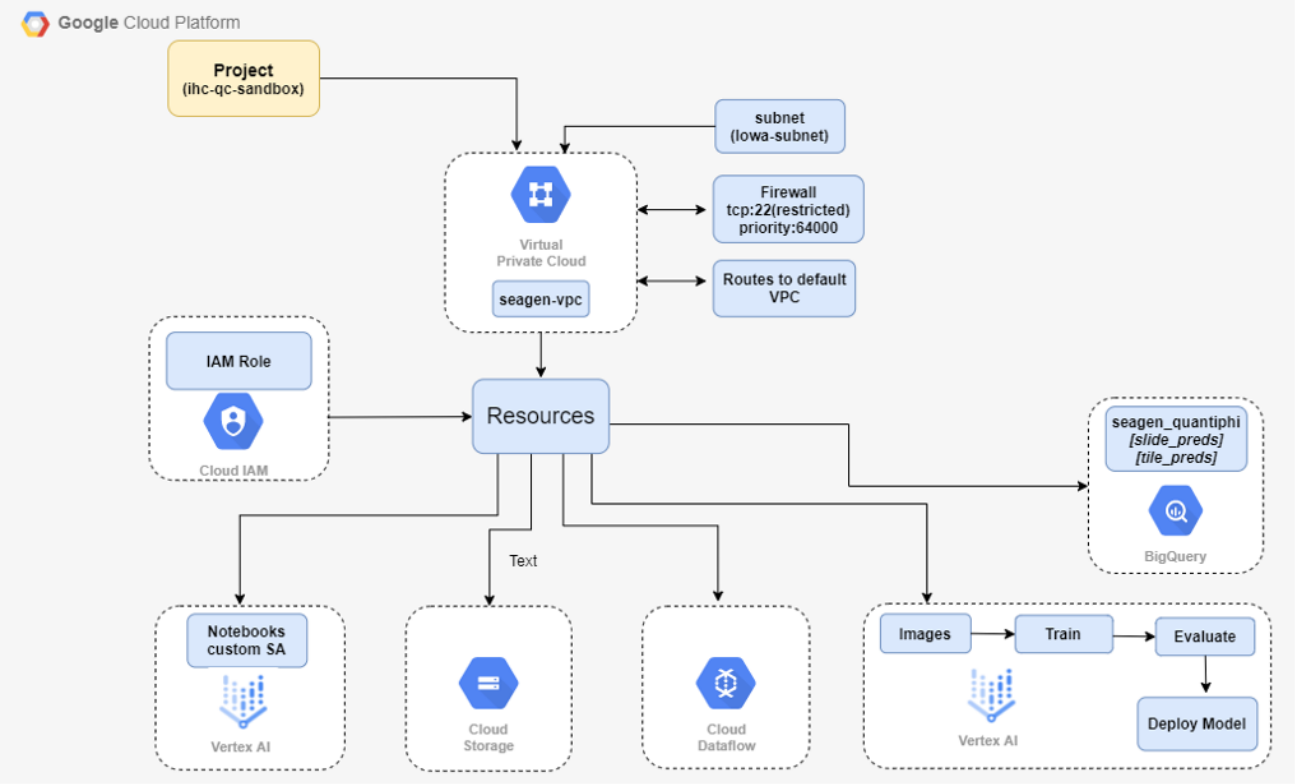


FIGURE 1 - Visualization of Data Access and Security Protocol

4 Solution Architecture

The user uploads a CSV file onto a Cloud Storage bucket that triggers a Cloud Function that launches the Dataflow pipeline. Each of the pipeline steps are explained in further detail under the Data Processing header.

The output of the pipeline is:

1. Biomax metadata pushed into a BigQuery table
2. Each tissue core is split from the TMA and saved on a Cloud Storage Bucket
3. The visualizations from the predictions of CD228 and the SLC1A5 models are saved on a Cloud Storage Bucket
4. The path to the tissue core, the visualisations and the confidence scores are saved on another BigQuery table.

HIGH-LEVEL ARCHITECTURE

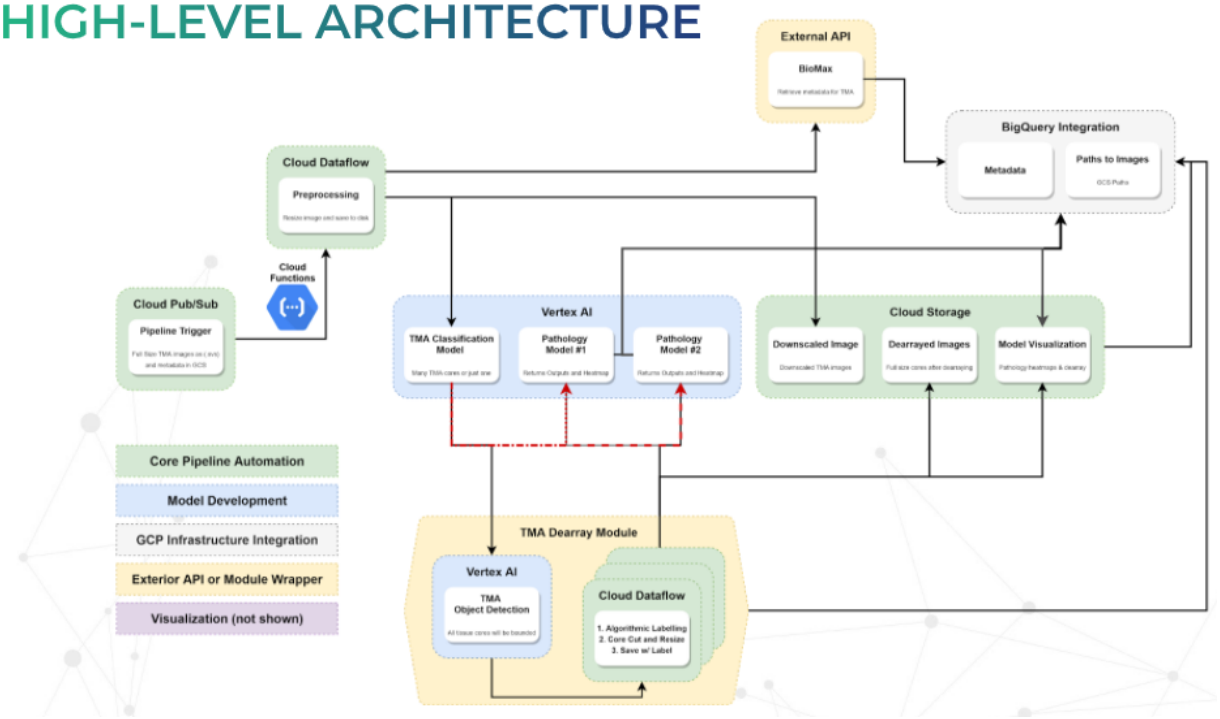
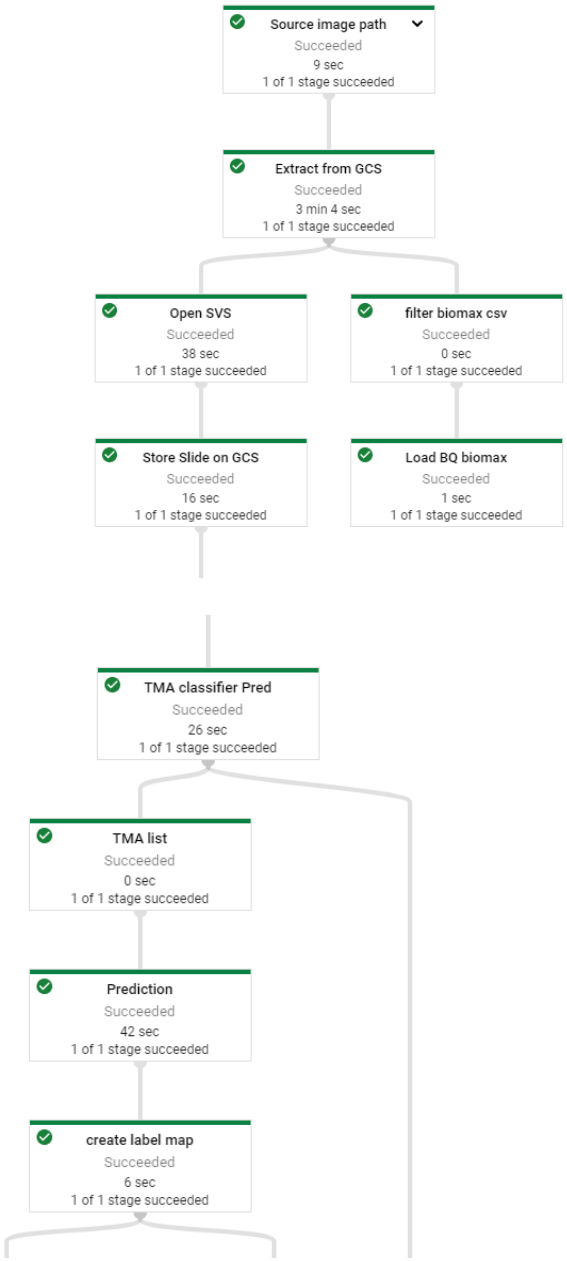


FIGURE 2 - Visual representation of the solution architecture

5 Data Processing Pipelines



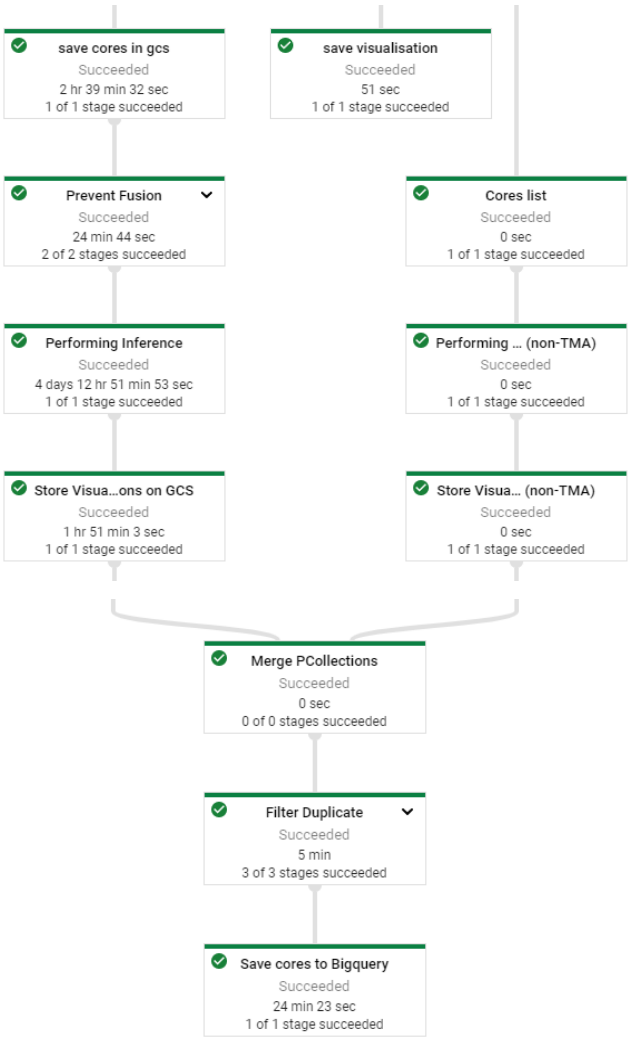


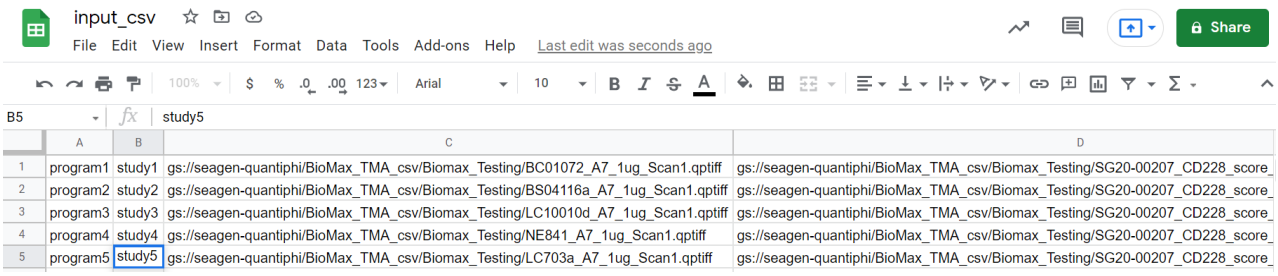
FIGURE 3 - Data Processing Pipeline

5.1 Source Image Path

The function reads the GCS image paths from the input CSV file and creates PCollections to read different files parallely.

5.2 Extract from GCS

- In this stage the file is downloaded from the GCS bucket as a blob object and stored in the Dataflow worker's local environment. It creates Pcollections for Biomax excel files and TMA(or Non TMA) images.



	A	B	C	D
1	program1	study1	gs://seagen-quantiphi/BioMax_TMA_csv/Biomax_Testing/BC01072_A7_1ug_Scan1.qptiff	gs://seagen-quantiphi/BioMax_TMA_csv/Biomax_Testing/SG20-00207_CD228_score_
2	program2	study2	gs://seagen-quantiphi/BioMax_TMA_csv/Biomax_Testing/BS04116a_A7_1ug_Scan1.qptiff	gs://seagen-quantiphi/BioMax_TMA_csv/Biomax_Testing/SG20-00207_CD228_score_
3	program3	study3	gs://seagen-quantiphi/BioMax_TMA_csv/Biomax_Testing/LC10010d_A7_1ug_Scan1.qptiff	gs://seagen-quantiphi/BioMax_TMA_csv/Biomax_Testing/SG20-00207_CD228_score_
4	program4	study4	gs://seagen-quantiphi/BioMax_TMA_csv/Biomax_Testing/NE841_A7_1ug_Scan1.qptiff	gs://seagen-quantiphi/BioMax_TMA_csv/Biomax_Testing/SG20-00207_CD228_score_
5	program5	study5	gs://seagen-quantiphi/BioMax_TMA_csv/Biomax_Testing/LC703a_A7_1ug_Scan1.qptiff	gs://seagen-quantiphi/BioMax_TMA_csv/Biomax_Testing/SG20-00207_CD228_score_

FIGURE 4 - Input CSV

- Pipeline further branches out in two parts A) Biomax csv load B) Image pre-processing for TMA classifier model(tma_dearray_011021).

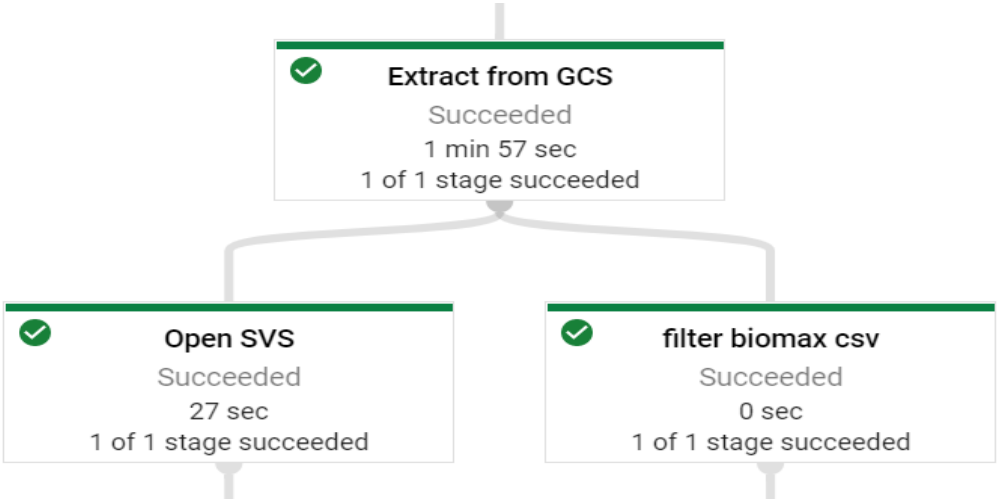


FIGURE 5 -Extract from GCS

5.3 Filter Biomax CSV

This step uses Apache Beam to filter PCollections having Biomax excel file path.

- Load BQ Biomax:
  - The function reads the biomax excel files and loads the data into the BigQuery table named Biomax_Metadata. The data is read from each file and loaded into the BQ table along with the filename.

- The schema for BQ table:

Field Name	Type	Mode
Name	RECORD	REPEATED
Name.Position	STRING	
Name.Number	STRING	
Name.Age	STRING	
Name.Sex	STRING	
Name.Organ_AnatomicSite	STRING	
Name.Pathology_diagnosis	STRING	
Name.TNM	STRING	
Name.Grade	STRING	
Name.Stage	STRING	
Name.Type	STRING	
Name.Tissue_ID	STRING	
TMA_Name	STRING	NULLABLE

TABLE 2 - BigQuery Table Schema

- The function is developed to discard the following cases -

Edge Case1	Only 'xlsx' files are allowed for Biomax.
Edge Case2	Xlsx file should have at least 11 columns.
Edge Case3	Function will ignore the file if the data for it is already present in BQ table

TABLE 3 - Edge Cases

5.4 Open SVS

- The step opens Image and gets the lowest tier dimensions and respective downsampling factor.
- It performs downsampling and returns the downsampled RGB and Gray scale images and downsampling factors. The downsampled images are stored in the respective job output folder.
- Example- Downsampled image-

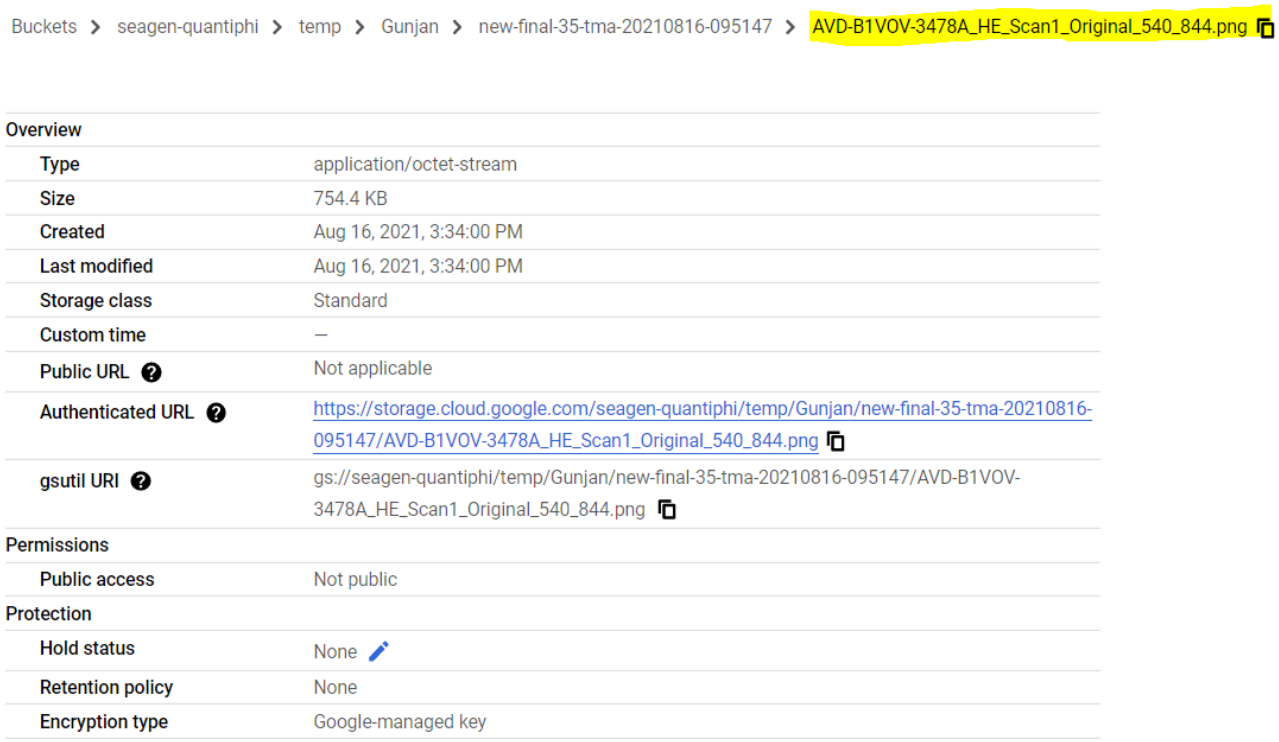


FIGURE 6 - Open SVS

5.5 Store Slide(s) on GCS

Files are uploaded to the GCS bucket based on the output location defined for the job in the configuration file.

5.6 TMA Classifier Prediction

- The TMA classifier model is deployed to Google Cloud VertexAI and the API is called through the end-point.
- The end point is defined in the configuration file. The REST API is called and base64 encoded downsampled images are sent in request parameters.
- Json response contains the numpy arrays for display names and confidences. The step predicts the image as TMA or Non-TMA.

- Model Endpoint on vertexAI -

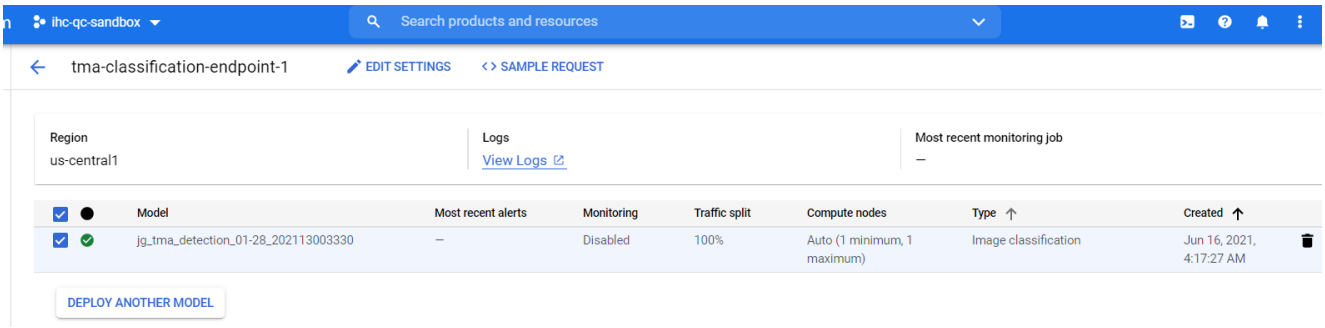


FIGURE 7 - Model Endpoint on VertexAI

- Pipeline branches into two parts:
  - TMA list
  - Cores list

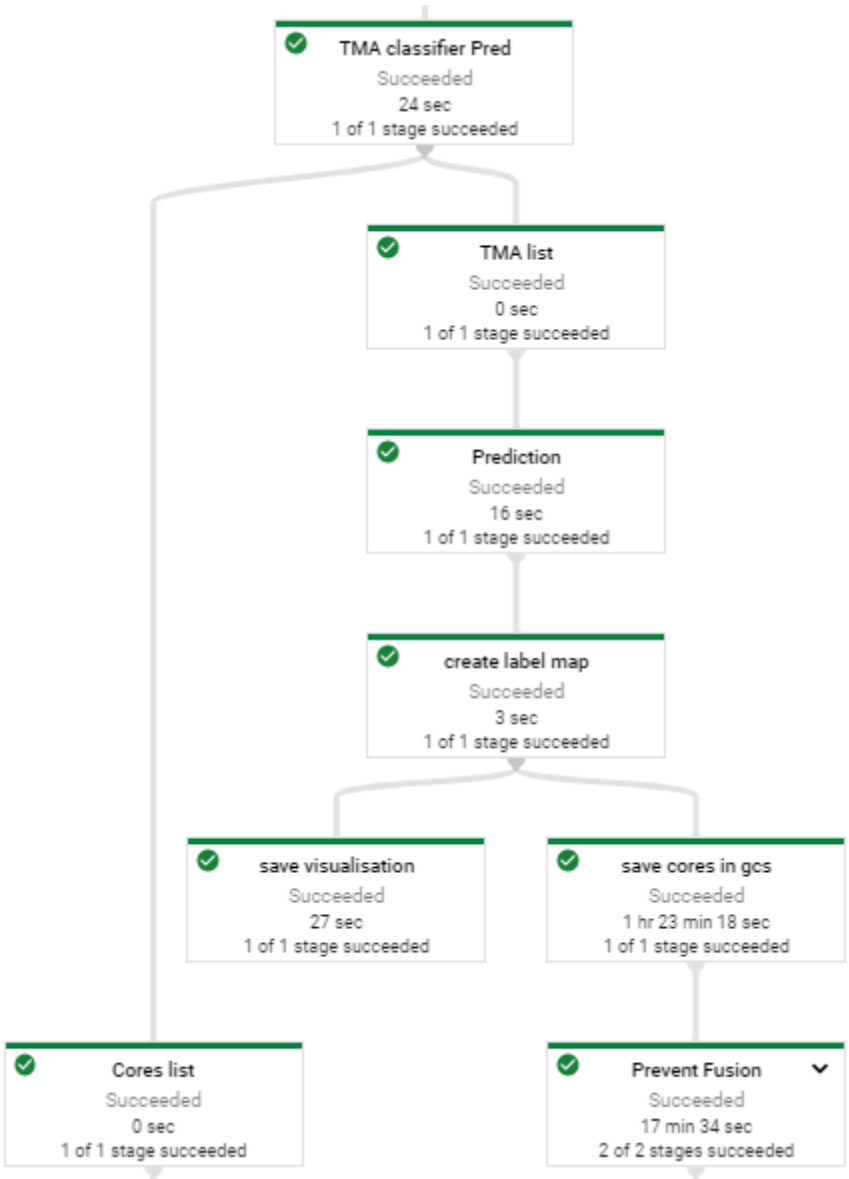


FIGURE 8 - TMA Classifier Prediction flow

5.7 TMA list & Prediction

- TMA images are encoded and passed to the prediction stage.
- TMA images are encoded into base64 format and GCP API for VertexAI is called for the specified endpoint.
- The model (tma_derray_01121) is deployed on VertexAI which is used to predict bounding boxes corresponding to the TMA array cores.

Model - tma_derray_01121-:

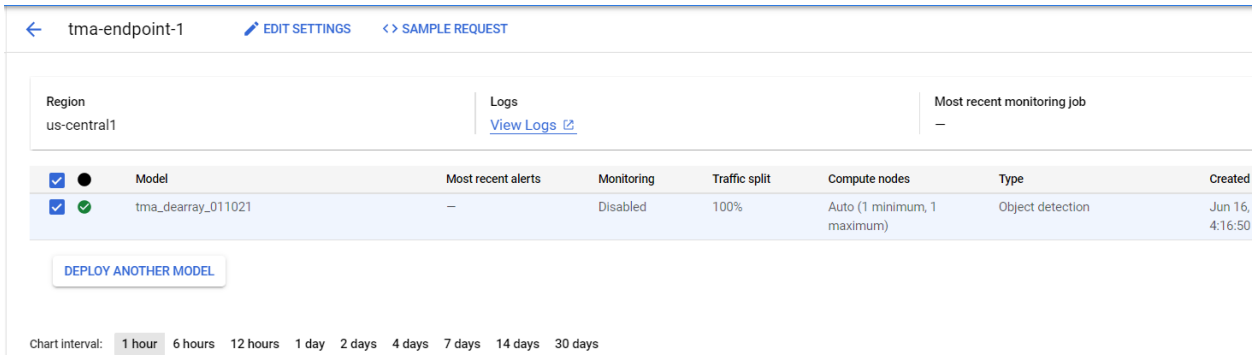


FIGURE 9 - tma_derray_01121

5.8 Create label map and save visualization

- The function creates a bounding box label map and based on that bounding box label map is created.
- It returns a dictionary mapping rows (labelled with capital letters starting at `A`) to a numpy array where each row in the array is the bounding box coordinates for a tissue sample (sorted from left to right).
- Save visualization function creates a visualization based on label map and saves the output to a gcs bucket. The output path can be set up in the configuration file.

- Sample visualization image -

FIGURE 10 - Visualization Image sample

5.9 Save Cores in GCS

- The function crops the individual cores from the TMA image and saves the cores into their respective image directory in the output gcs path.

Cores for TMA image - BL2081_ASCT2_Bladder_Cancer_TMA

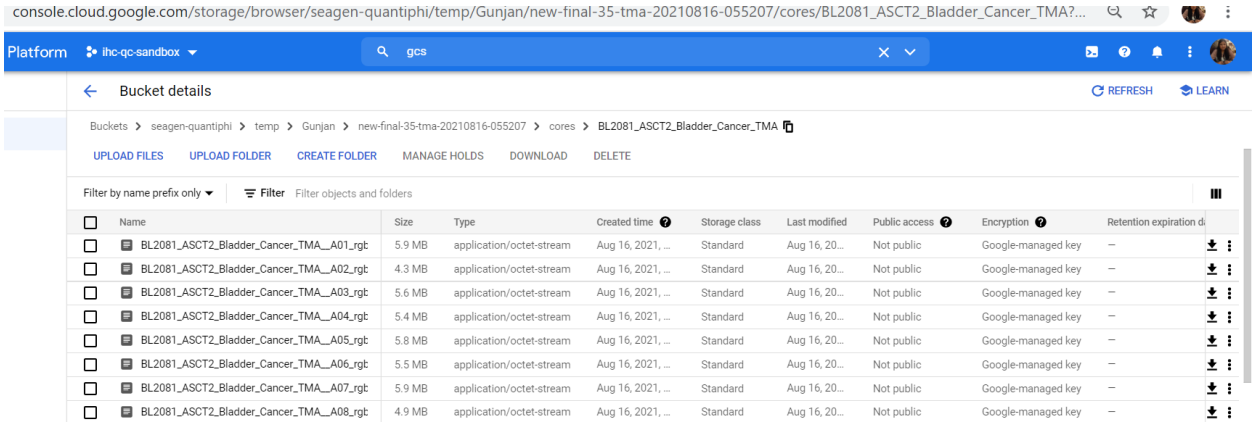


FIGURE 11 - Cores for TMA images

5.10 Prevent Fusion

This stage is responsible for parallelization of the workload in the later stages of the pipeline. This is accomplished by the Apache Beam's Reshuffle function

5.11 Performing Inference(TMA)

In this stage of the pipeline, we send request to the two pathology models i.e. CD228 & SLC1A5 hosted on vertex AI and get the predictions

- The models are deployed to a GPU endpoint, with n1-standard-4 machine type and NVIDIA_TESLA_P100 accelerator type
- The endpoint name of CD228 model is CD228_endpoint_GPU and endpoint id is 1796285340437184512
- The endpoint name of SLC1A5 model is SLC1A5_endpoint_GPU and endpoint id is 3949005962320281600
- The input to this step is a path to a single core, and the output is path to visualization image and the predictions

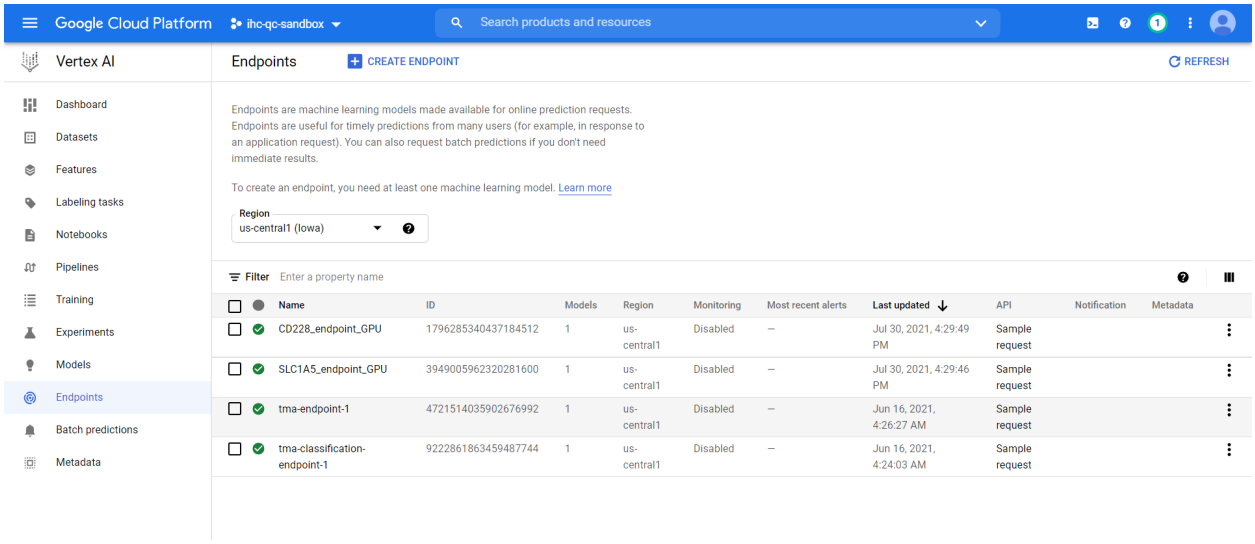


FIGURE 12 - List of Endpoints on VertexAI

The first two endpoints are the two pathology models

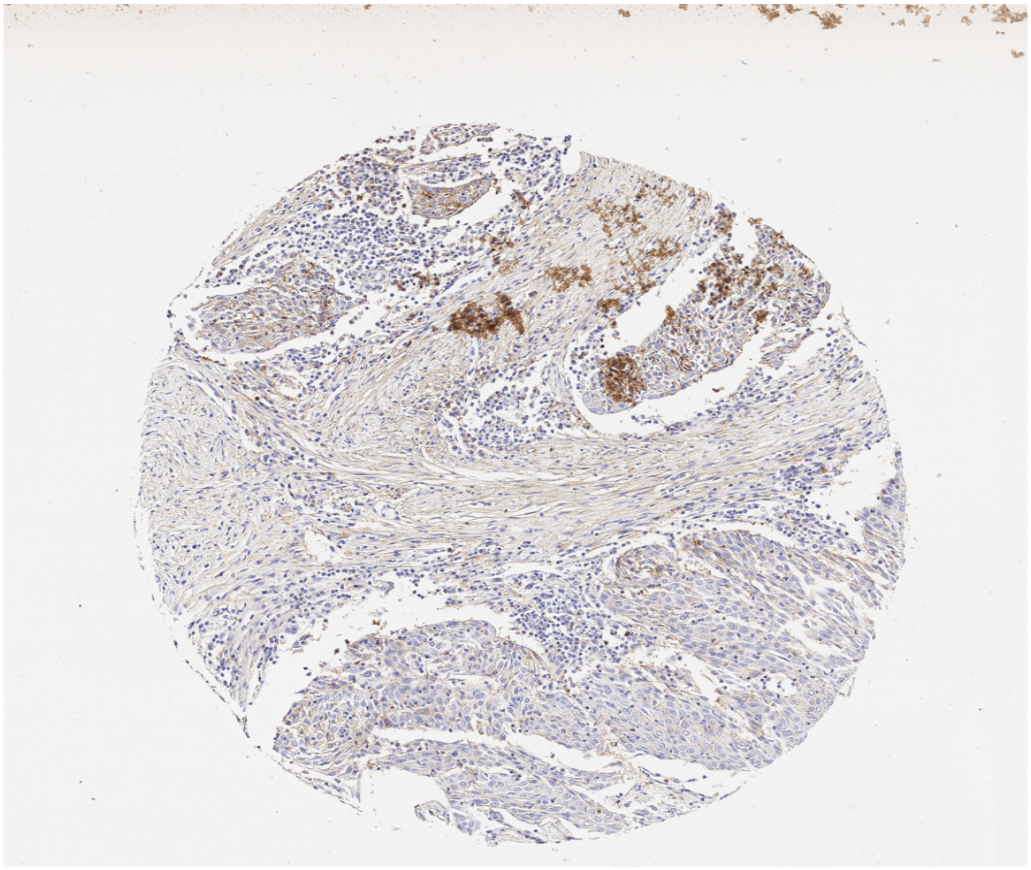


FIGURE 13 - Input Core Image

FIGURE 14 - Visualization output of CD228 model

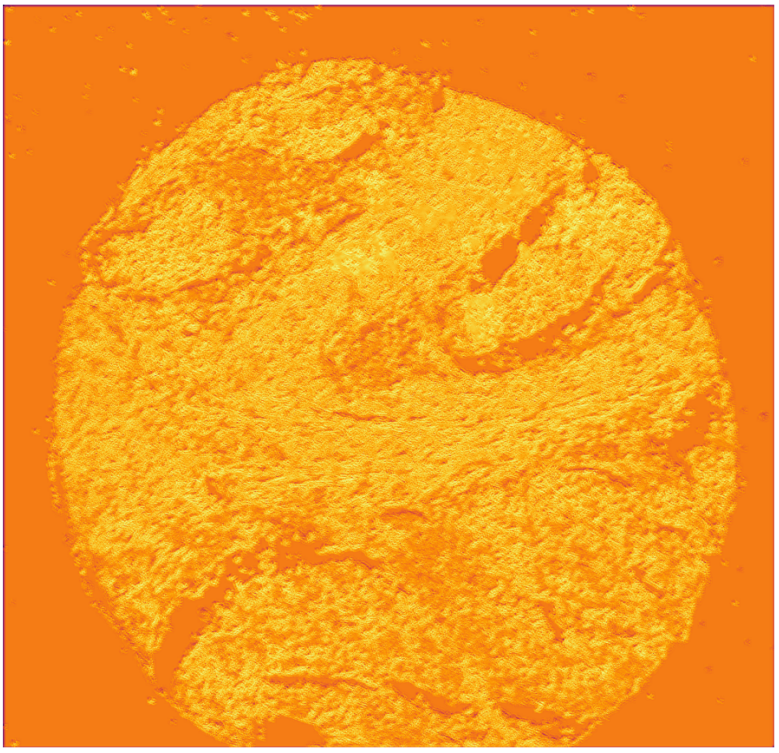


FIGURE 15 - Visualization output of SLC1A5 model

5.12 Performing Inference (Non-TMA)

This stage is similar to the Performing Inference, except that the input to this function is the non-TMA images instead of TMA cores

5.13 Store Visualizations on GCS

This stage of the pipeline is used to store the visualization output of the pathology models in the GCS bucket. This function takes as input the predictions and the local path to output visualization, and stores them in the GCS bucket. The visualization is saved as a PNG whereas the predictions are saved as a TEXT.

5.14 Merge PCollections

This stage joins the TMA and non-TMA pipelines to give a single output.

5.15 Filter Duplicate

This stage is inserted to prevent duplicate entries in the Bigquery

5.16 Save Cores to Bigquery

This stage saves the paths of original slide, downscaled image, dearrayed image, cores, CD228 visualization of the cores and SLC1A5 visualization of the cores to the Bigquery.

Row	original_svs	downscaled_image
1	https://storage.cloud.google.com/ihc_dataset/TMA_detection/testing/not_tma/29748.svs	https://storage.cloud.google.com/seagen-qua
2	https://storage.cloud.google.com/ihc_dataset/TMA_detection/testing/not_tma/29748.svs	https://storage.cloud.google.com/seagen-qua
3	https://storage.cloud.google.com/ihc_dataset/TMA_detection/testing/not_tma/29814.svs	https://storage.cloud.google.com/seagen-qua
4	https://storage.cloud.google.com/ihc_dataset/TMA_detection/testing/not_tma/29748.svs	https://storage.cloud.google.com/seagen-qua
5	https://storage.cloud.google.com/ihc_dataset/TMA_detection/testing/not_tma/29777.svs	https://storage.cloud.google.com/seagen-qua
6	https://storage.cloud.google.com/ihc_dataset/TMA_detection/testing/not_tma/29795.svs	https://storage.cloud.google.com/seagen-qua
7	https://storage.cloud.google.com/ihc_dataset/Quantiphi/completed/H&E/AVD-B1VOV-3478A_HE_Scan1.qptiff	https://storage.cloud.google.com/seagen-qua

FIGURE 16 - Preview of the Bigquery Table - TMA_paths

Field name	Type	Mode	Policy Ta
original_svs	STRING	NULLABLE	
downscaled_image	STRING	NULLABLE	
downscaled_dearray_visualisation	STRING	NULLABLE	
program	STRING	NULLABLE	
st	STRING	NULLABLE	
cores	RECORD	REPEATED	
name	STRING	NULLABLE	
path	STRING	NULLABLE	
CD228_visualization_path	STRING	NULLABLE	
CD228_mi_value	FLOAT	NULLABLE	
CD228_md_value	FLOAT	NULLABLE	
CD228_ci_value	FLOAT	NULLABLE	
CD228_cd_value	FLOAT	NULLABLE	
SLC1A5_visualization_path	STRING	NULLABLE	
SLC1A5_s_value	FLOAT	NULLABLE	
SLC1A5_i_value	FLOAT	NULLABLE	
SLC1A5_f_value	FLOAT	NULLABLE	

Collapse node

FIGURE 17 - Schema of TMA_paths Bigquery table

6 Convert Model to Base64

This section details out the method to modify the model to be able to host it on Vertex AI:

1. Modify the input signature of the model to have:
    a. shape = [None]
    b. dtype = tf.string
    c. name = 'image_bytes'
2. Add a few layers in the beginning of the model. They are as follows:
    a. tf.io.decode_base64: This is required to decode the base64 encoded string. This takes as input, a tensor of type string and returns a tensor of type string.
    b. tf.io.decode_image: This will convert the string to an image with the desired shape and dtype. In our case, the output shape = (2048, 2048, 3) and dtype = uint16
    c. Changing dtype of the image to tf.float32 and re-scaling it back to what the original model requires
    d. tf.reverse(img, axis=[-1]): This is required because the Opencv saves the image as PNG in 'BGR' format. Hence the tf.reverse would change the orientation of the image back to 'RGB'
3. Save the new model as a SavedModel format using tf.saved_model.save
4. Upload the SavedModel to a GCS bucket and import it in Vertex AI

Seagen®
quantiphi
Solving What Matters

Code Snippet to modify the model:

```
import tensorflow as tf

model_sm = tf.keras.models.load_model('SLC1A5_model/')

@tf.function(input_signature=[tf.TensorSpec(
              shape=[None],
              dtype=tf.string,
              name='image_bytes')])
def f_with_input_signature(image_bytes):
    img = tf.io.decode_base64(image_bytes)
    image_features = tf.map_fn(lambda x: tf.io.decode_image(x, dtype=tf.uint16),
                               img, fn_output_signature=tf.uint16)
    img = tf.cast(image_features, tf.float32)/65535.0
    image = tf.reverse(img, axis=[-1])
    return {'attention_layer': model_sm(image)[1],
            's': model_sm(image)[0][0],
            'i': model_sm(image)[0][1],
            'f': model_sm(image)[0][2]}

tf.saved_model.save(model_sm, 'Vertex_SLC1A5_model', signatures=f_with_input_signature)
```

Method to send request to the Vertex AI model:

1. Given: An image of shape = (2048, 2048, 3) & dtype = float32
   CD228 preprocessed images have min = -1.732051 & max = 2
   SLC1A5 preprocessed images have min = 0 & max = 1
2. First we scale the images between 0 & 65535 (for uint16).
   For CD228 images, this was done as follows
   `tf.cast(65535.0*(cd228_core_img + 1.732051)/3.732051, tf.uint16)`
   And for SLC1A5 images as follows
   tf.cast(65535.0*slc1a5_core_img, tf.uint16).numpy()
3. Then they were saved as PNG using cv2.imwrite
4. The PNG was then read as bytes and subsequently Base64 encoded using `base64.urlsafe_b64encode`
5. It was then sent as a request to the Vertex AI Model

7 Production Environment

7.1 Code Repository

The Entire code source can be found in the staging branch of code repository - **seattle-quantiphi**, which is mirrored from Quantiphi's safe and secure private Gitlab repository.

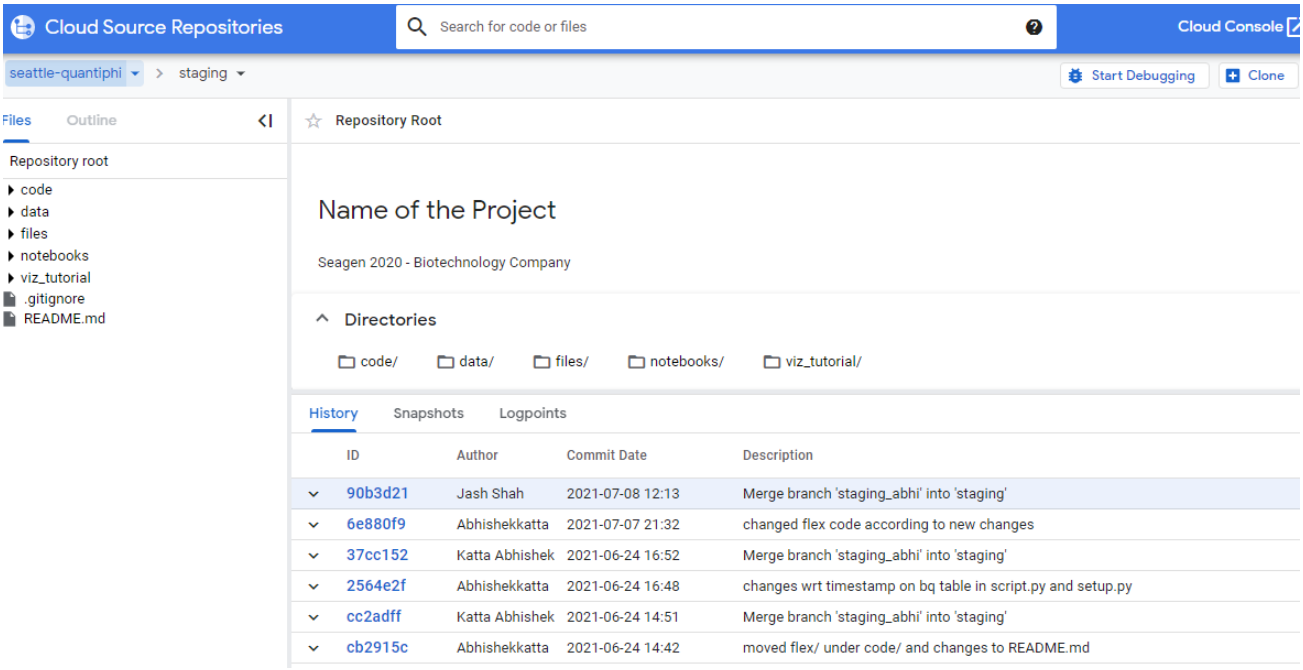


FIGURE 18 - Cloud Source Repository

7.2 Virtual Private Cloud (VPC)

A Virtual Private Cloud Network has been created in the Project - **seagen-vpc**, which provides connectivity for the Google Cloud products built on Compute Engine VMs (Notebooks, Dataflow instances).

Name	Region	Subnets	MTU	Mode	IP address ranges	Gateways
seagen-vpc		2	1460	Custom		
	us-central1	iowa-subnet			192.168.4.0/22	192.168.4.1
	us-west1	oregon-subnet			192.168.0.0/22	192.168.2.1

FIGURE 19 - Virtual Private Cloud (VPC)

7.3 Dataflow

7.3.1 IHC

7.3.1.1 Create Dataflow Template

Note: Please ignore this section if the template is already built.

To create the Dataflow Template please follow the below steps:

1. Clone the source code from the Cloud Source Repository using gcloud or SSH.

```
gcloud source repos clone seattle-quantiphi --project=ihc-qc-sandbox
```

or

```
git clone
```

```
ssh://<email-id>@source.developers.google.com:2022/p/ihc-qc-sandbox/r/seattle-quantiphi.
```

*Note: replace <email-id> with the user email-id which has access to the Cloud Source Repository

2. Switch to the seattle-quantiphi directory.

```
cd seattle-quantiphi
```

3. Checkout the staging branch.

```
git checkout staging
```

4. Navigate to the IHC code Directory.

```
cd IHC/code/flex
```

5. Run the below Shell command.

```
/bin/bash build_flex_template_script.sh
```

7.3.1.2 Run Dataflow Job

1) Using Cloud Functions:

Upload an input CSV file to the bucket **seagen_ihc_input_cf**, the Cloud Function will be triggered and a dataflow job will be run.

Note: The input file should be a csv containing gcs slide path in each row. An example input csv with 10 slides can be found here:- gs://seagen_dataflow/input_csv/input_paths.csv

2) Using gcloud:

Execute the below command in the current directory on the terminal:

```
/bin/bash run_dataflow_flex_job.sh
```

Example gcloud command to run a dataflow job:

```
gcloud dataflow flex-template run flex_job_1 \
--template-file-gcs-location "gs://seagen_dataflow/templates/ihc/v1.json" \
--parameters config_file="gs://abhi_test_dataflow/config/config.json" \
--parameters input_path="gs://seagen_dataflow/input_csvs/input_paths_1.csv" \
--parameters output_path="gs://abhi_test_dataflow/results_runner/test" \
--region "us-west1" \
--subnetwork "regions/us-west1/subnetworks/oregon-subnet"
```

***Note: The Dataflow Jobs are run inside the seagen-vpc network on a subnet(ex. oregon-subnet) with 1024 usable addresses. Please make sure the number of workers running never increases over 1024 workers as it would throw errors.**

7.3.2 TMA

7.3.2.1 Create Dataflow Template

Note: Please ignore this section if the template is already built.

To create the Dataflow Template please follow the below steps:

1. Clone the source code from the Cloud Source Repository using SSH or gcloud.

```
gcloud source repos clone seattle-quantiphi --project=ihc-qc-sandbox
```

or

```
git clone
ssh://<email-id>@source.developers.google.com:2022/p/ihc-qc-sandbox/r/seattle-quantiphi.
```

*Note: replace <email-id> with the user email-id which has access to the Cloud Source Repository

2. Switch to the **seattle-quantiphi** directory.

```
cd seattle-quantiphi
```

3. Checkout the staging branch.

```
git checkout staging
```

4. Navigate to the IHC code Directory.

```
cd TMA/flex_tma
```

5. Run the below Shell command.

```
/bin/bash build_flex_template_tma.sh
```

7.3.2.2 Run Dataflow Job

1) Using Cloud Functions:

Upload an input CSV file to the bucket **seagen_tma_input_cf**, the Cloud Function will be triggered and a dataflow job will be run.

Note: The input file should be a csv containing gcs slide path in each row. An example input csv with 10 slides can be found here:- gs://seagen_dataflow/input_csv/file_30.csv

2) Using gcloud:

Execute the below command in the current directory on the terminal:

```
/bin/bash run_dataflow_flex_job_tma.sh
```

Example gcloud command to run a dataflow job:

```
gcloud dataflow flex-template run flex-tma-v2 \
--template-file-gcs-location "gs://seagen_dataflow/templates/tma/v2.json" \
--parameters config_file="gs://seagen_dataflow/config_tma2/config.json" \
--parameters input_path="gs://seagen_dataflow/input_csvs/file_tma_35.csv" \
--parameters output_path="gs://seagen_dataflow/results_runner/output-tma-flex2" \
--region "us-west1" \
--subnetwork "regions/us-west1/subnetworks/oregon-subnet"
```

***Note: The Dataflow Jobs are run inside the seagen-vpc network on a subnet(ex. oregon-subnet) with 1024 usable addresses. Please make sure the number of workers running never increases over 1024 workers as it would throw errors. With the current worker configuration for TMA Dataflow jobs the maximum number of workers is set to 200 , which allows a maximum of 5 Dataflow jobs to run in Parallel.**

8 Recommendations and Next Steps

1. Retrain the CD228 and the SLC1A5 models on smaller image input size instead of 2048 x 2048
2. Create an MLOps pipeline for each model deployed