Cloud Machine Learning Engine
(Cloud ML Engine)

Telung Pan Ph.D.
telung@mac.com

Network Concepts

- In Google Cloud Platform, networks provide data connections into and out of your cloud resources (mostly Compute Engine instances).

- Google Cloud Platform supports Projects, Networks, and Subnetworks to provide flexible, logical isolation of unrelated resources.

Projects

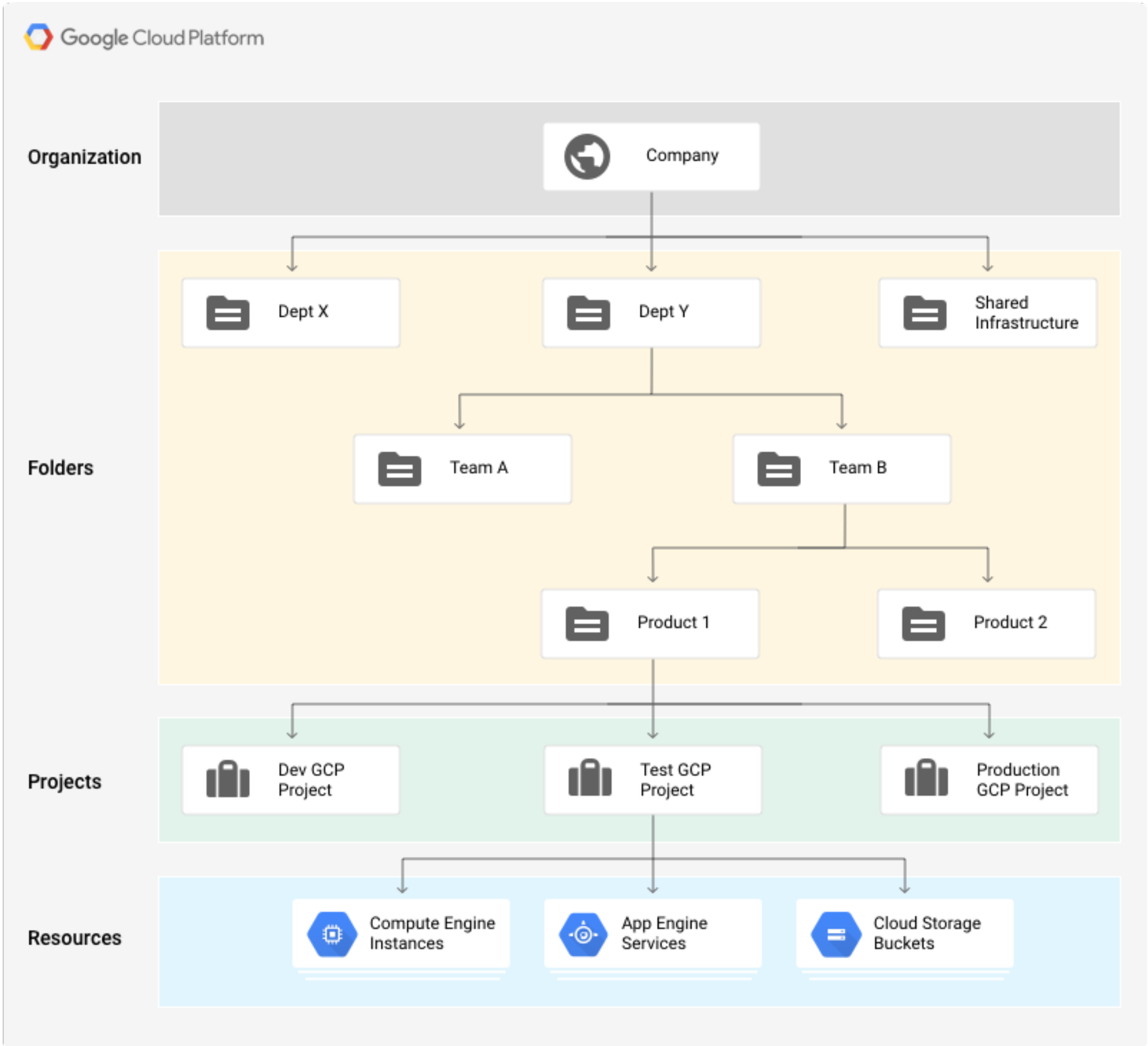
- Projects are the outermost container and are used to group resources that share the same trust boundary.

- Many developers map Projects to teams since each Project has its own access policy (IAM) and member list.

- Projects also serve as a collector of billing and quota details reflecting resource consumption.

- Projects contain Networks which contain Subnetworks, Firewall rules, and Routes (see below architecture diagrams for illustration).

Projects and Teams



Basic gcloud Commands

- gcloud auth login
- gcloud components update
- gcloud auth list
- gcloud config list project

Create an API Key

- Since we'll be using curl to send a request to the Vision API, we'll need to generate an API key to pass in our request URL. To create an API key, navigate to:

  ***APIs* 和服務 > 憑證 >** 建立憑證 > API 金鑰

- Next, copy the key you just generated. Click Close.

- Now that you have an API key, save it to an environment variable to avoid having to insert the value of your API key in each request. You can do this in Cloud Shell. Be sure to replace <your_api_key> with the key you just copied.

  export API_KEY=<YOUR_API_KEY>

Upload an image to a cloud storage bucket

- Creating a Cloud Storage bucket
- There are two ways to send an image to the Vision API for image detection:
  - Sending the API a base64 encoded image string
  - Or, passing it the URL of a file stored in Google Cloud Storage.

- gsutil mb gs://g9220812
- gustil cp Downloads/sign.jpg gs://g9220812
- gsutil ls gs://g9220812

```
pandelongde-MacBook-Air:~ telung$ gsutil cp Downloads/sign.jpg gs://g9220812
Copying file://Downloads/sign.jpg [Content-Type=image/jpeg]...
- [1 files][170.5 KiB/170.5 KiB]
Operation completed over 1 objects/170.5 KiB.
```

Create your Vision API request

```
{
  "requests": [
    {
      "image": {
        "source": {
            "gcsImageUri": “gs://g9220812/sign.jpg"
        }
      },
      "features": [
        {
          "type": "TEXT_DETECTION",
          "maxResults": 10
        }
      ]
    }
  ]
}
```

Call the Vision API's text detection method

- In Cloud Shell, call the Vision API with curl:

```
curl -s -X POST -H "Content-Type: application/json" --
data-binary @ocr-request.json  https://
vision.googleapis.com/v1/images:annotate?key=$
{API_KEY}
```

- The OCR method is able to extract lots of text from our image.

- The first piece of data you get back from textAnnotations is the entire block of text the API found in the image. This includes the language code (in this case fr for French), a string of the text, and a bounding box indicating where the text was found in our image.

- Then you get an object for each word found in the text with a bounding box for that specific word.

- Run the following curl command to save the response to an ocr-response.json file so it can be referenced later:

- curl -s -X POST -H "Content-Type: application/json" --data-binary @ocr-request.json https://vision.googleapis.com/v1/images:annotate?key=${API_KEY} -o ocr-response.json

Sending text from the image to the Translation API

- The Translation API can translate text into 100+ languages. It can also detect the language of the input text. To translate the French text into English, all you need to do is pass the text and the language code for the target language (en-US) to the Translation API.

- First, create a translation-request.json file and add the following to it:

  ```
  {
  "q": "your_text_here",
  "target": "en"
  }
  ```

- q is where you'll pass the string to translate.

- Save the file.

- Run this Bash command in Cloud Shell to extract the image text from the previous step and copy it into a new translation-request.json (all in one command):

  STR=$(jq .responses[0].textAnnotations[0].description ocr-response.json) && STR="${STR//\"}" && sed -i "s|your_text_here|$STR|g" translation-request.json

- Now you're ready to call the Translation API. This command will also copy the response into a translation-response.json file:

  curl -s -X POST -H "Content-Type: application/json" --data-binary @translation-request.json https://translation.googleapis.com/language/translate/v2?key=${API_KEY} -o translation-response.json

- Run this command to inspect the file with the Translation API response:

  cat translation-response.json

- Awesome, you can understand what the sign said!

  {
  "data": {
  "translations": [
  {
  "translatedText": "THE PUBLIC GOOD the despatches For Obama, the mustard is from Dijon",
  "detectedSourceLanguage": "fr"
  }
  ]
  }
  }

- In the response, translatedText contains the resulting translation, and detectedSourceLanguage is fr, the ISO language code for French. The Translation API supports 100+ languages, all of which are listed here.

- In addition to translating the text from our image, you might want to do more analysis on it. That's where the Natural Language API comes in handy. Onward to the next step!

Analyzing our image's text with the Natural Language API

- The Natural Language API helps us understand text by extracting entities, analyzing sentiment and syntax, and classifying text into categories. Use the analyzeEntities method to see what entities the Natural Language API can find in the text from your image.

- To set up the API request, create a nl-request.json file with the following:

```
{
"document":{
"type":"PLAIN_TEXT",
"content":"your_text_here"
 },
"encodingType":"UTF8"
}
```

- In the request, you're telling the Natural Language API about the text you're sending:
    - type: Supported type values are PLAIN_TEXT or HTML.
    - content: pass the text to send to the Natural Language API for analysis. The Natural Language API also supports sending files stored in Cloud Storage for text processing. To send a file from Cloud Storage, you would replace content with gcsContentUri and use the value of the text file's uri in Cloud Storage.
    - encodingType: tells the API which type of text encoding to use when processing the text. The API will use this to calculate where specific entities appear in the text.
- Run this Bash command in Cloud Shell to copy the translated text into the content block of the Natural Language API request:

  STR=$(jq .data.translations[0].translatedText translation-response.json) STR="${STR//\"}" sed -i "s|your_text_here|$STR|g" nl-request.json

- The nl-request.json file now contains the translated English text from the original image. Time to analyze it! Call the analyzeEntities endpoint of the Natural Language API with this curl request:

  curl "https://language.googleapis.com/v1/documents:analyzeEntities?key=${API_KEY}" \ -s -X POST -H "Content-Type: application/json" --data-binary @nl-request.json

- For entities that have a wikipedia page, the API provides metadata including the URL of that page along with the entity's mid. The mid is an ID that maps to this entity in Google's Knowledge Graph. To get more information on it, you could call the Knowledge Graph API, passing it this ID. For all entities, the Natural Language API tells us the places it appeared in the text (mentions), the type of entity, and salience (a [0,1] range indicating how important the entity is to the text as a whole). In addition to English, the Natural Language API also supports the languages listed here.

References

- https://cloud.google.com/ml-engine/docs/deploying-models?hl=zh_TW&_ga=2.134531188.-406726904.1448067839
- https://cloud.google.com/ml-engine/docs/getting-started-training-prediction
- https://cloud.google.com/ml-engine/docs/training-overview#input_data
- https://cloud.google.com/ml-engine/docs/data-prep
- https://cloud.google.com/ml-engine/docs/training-overview
- https://cloud.google.com/ml-engine/docs/training-steps
- https://cloud.google.com/ml-engine/docs/deploying-models