

Activity 1 – Hands on creating a QnA chatbot

In this activity, we will learn:

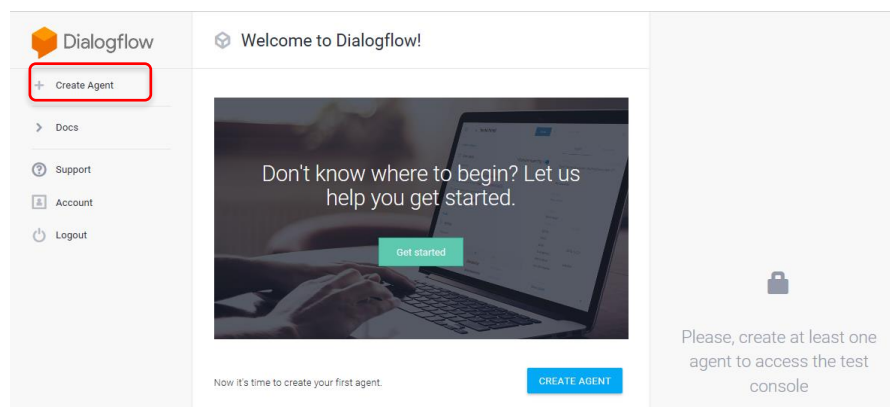
- ☐ Create DialogFlow Knowledge Base.
- ☐ Create an Integration to use a knowledge base
- ☐ Chat with the bot to verify the code is working
- ☐ Link Telegram to bot's channel

1. Prerequisites

- a) To start using Dialogflow Knowledge, you will need to have a Google account. Head over to here (<https://accounts.google.com/signup/v2>) to create one if necessary.

2) Create an Agent

- a) Login in to Dialogflow (<https://console.dialogflow.com/api-client/#/login>).
- b) If it is the first time you login to Dialogflow with this account, you will be asked to allow Dialogflow to access your Google account. Click on **Allow** to continue.
- c) In the next screen, review your account settings. The only required action is to check “*Yes, I have read and accept the agreement*”. Click **ACCEPT** to continue.
- d) You may be asked to authenticate again. If everything is ok, you should see the following screen. Click on **Create Agent**

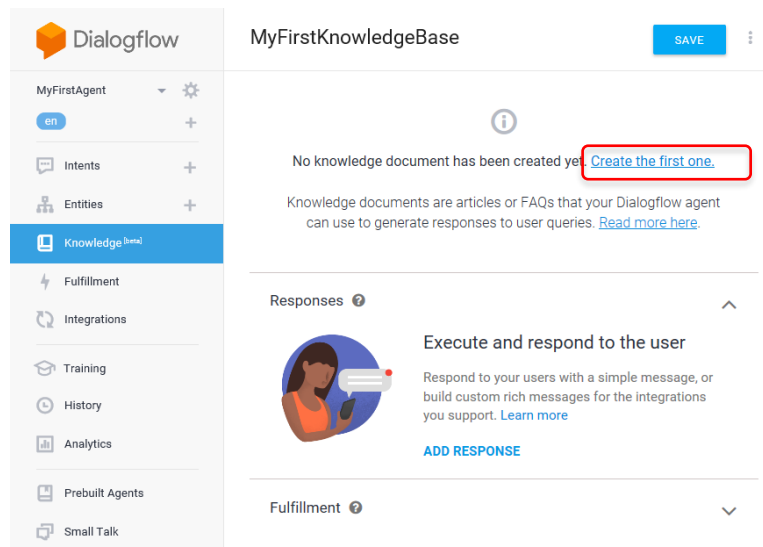


- e) Give your agent an appropriate name. We can leave the **DEFAULT LANGUAGE** and **DEFAULT TIME ZONE** settings. Click **CREATE** to continue. After a while, the screen will refresh and bring you to details configuration page.
- f) Click on **Setting** icon. In the **General** tab, turn on **BETA FEATURES** and click **Save**. Do note the **Dialogflow Knowledge** is still in beta at this point in time.

3) Create Knowledge Base

- a) On the left panel, click **Knowledge [beta]**, and then **CREATE KNOWLEDGE BASE** to create a new knowledge base.

- b) Enter a suitable knowledge base name and click on **SAVE**.
- c) Click on **Create the first one** to create a knowledge document. Knowledge documents are articles or FAQs that your Dialogflow agent can use to generate responses to user queries.



d) Enter values according to your data.

Document Name: this can be anything that you want

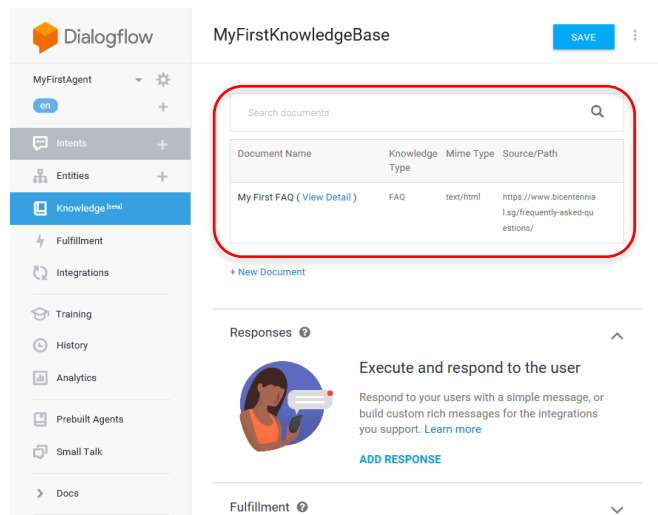
Knowledge Type: this can be selected as FAQ. For FAQ, only text/html and text/csv is supported for now.

MIME type: this is the type of data you are going to feed to the bot. It has options like (text/plain, text/html, text/csv, application/pdf)

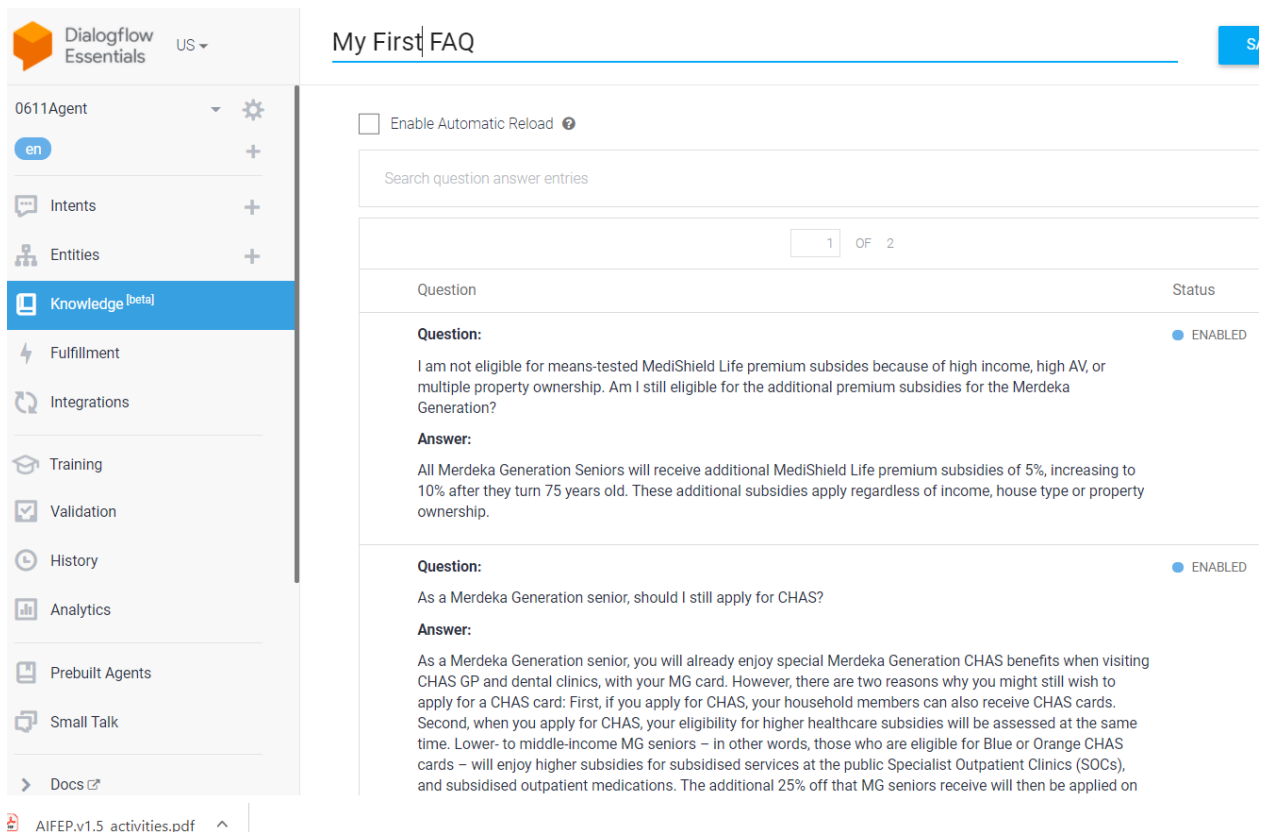
Data Source: this can be provided as a file on cloud storage or local, you can also give this as a URL to a public page. For our purpose of this activity, we will use a URL. Enter <https://www.merdekahgeneration.sg/en/faqs>. You are free to try with another FAQ URLs.

Click **CREATE** to continue.

e) Wait for it to generate all the data from the source that you have provided. This usually takes about 2–5 minutes depending upon the size of your data. You can see 'My First FAQ' knowledge base has been created.



- f) You can look at the content of data inside FAQ by clicking on '**View Detail**'. It should look something like this



- g) Click the menu button (three vertical dots) beside the SAVE button and select **back** in the dropdown menu to return to the Knowledge base view.

4) Create telegram bot

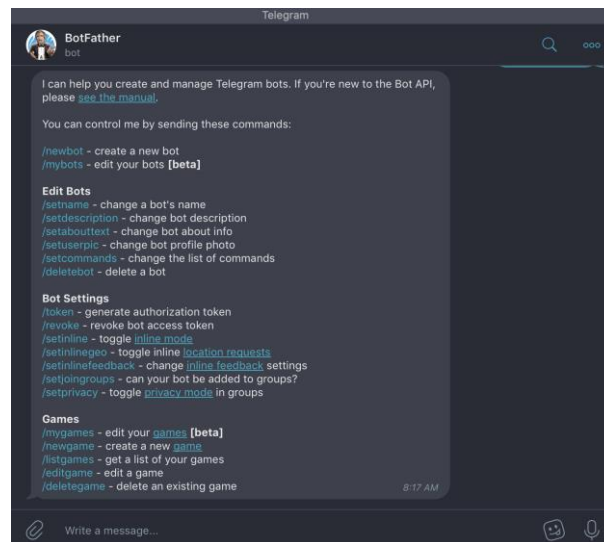
For most integration, you must provide configuration information to run your bot with Dialogflow. Most channels require that your bot have an account on the channel, and others, like Facebook Messenger, require your bot to have an application registered with the channel also.

- a) If you do not have telegram installed on your mobile phone, please install via Google Play store or Apple App Store.

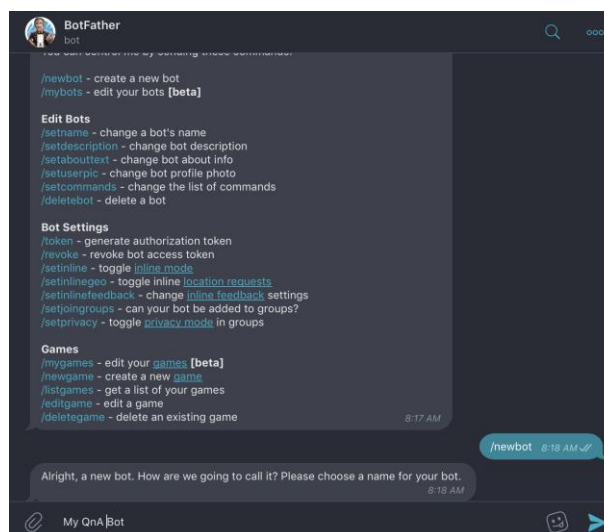
- b) If you have telegram installed on your laptop, use this [link](https://telegram.me/botfather) to connect to Bot Father (<https://telegram.me/botfather>) or add botfather.
Otherwise, if you are using telegram on your mobile, start telegram and search for BotFather.



- c) Click on **Start** to start a conversation with botfather.

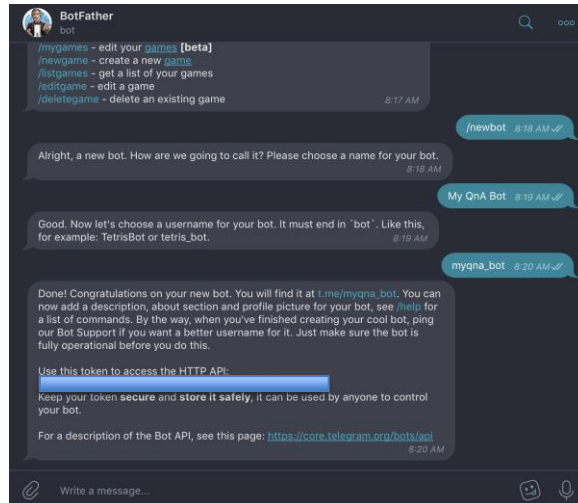


- d) Create a new Telegram bot by sending command /newbot.



- e) Give the bot a friendly name.

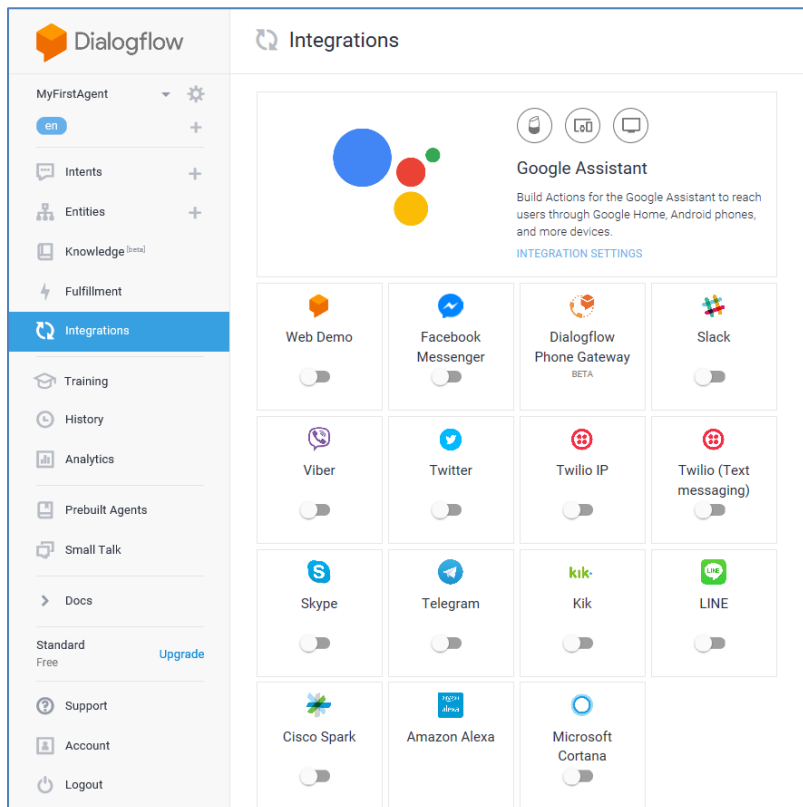
- f) Specify a username. **Name cannot start with number and must end with _bot*



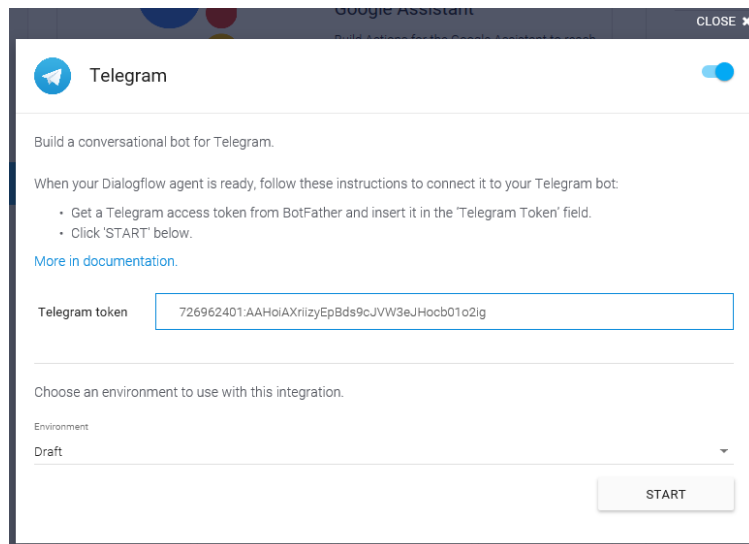
- g) Copy the Telegram bot's access token provide in the screen above. You will need this info to configure integration for the Agent.

5) Integrate telegram with Dialogflow

- a) Back to Dialogflow dashboard, click on **Integration**. switch on Telegram.



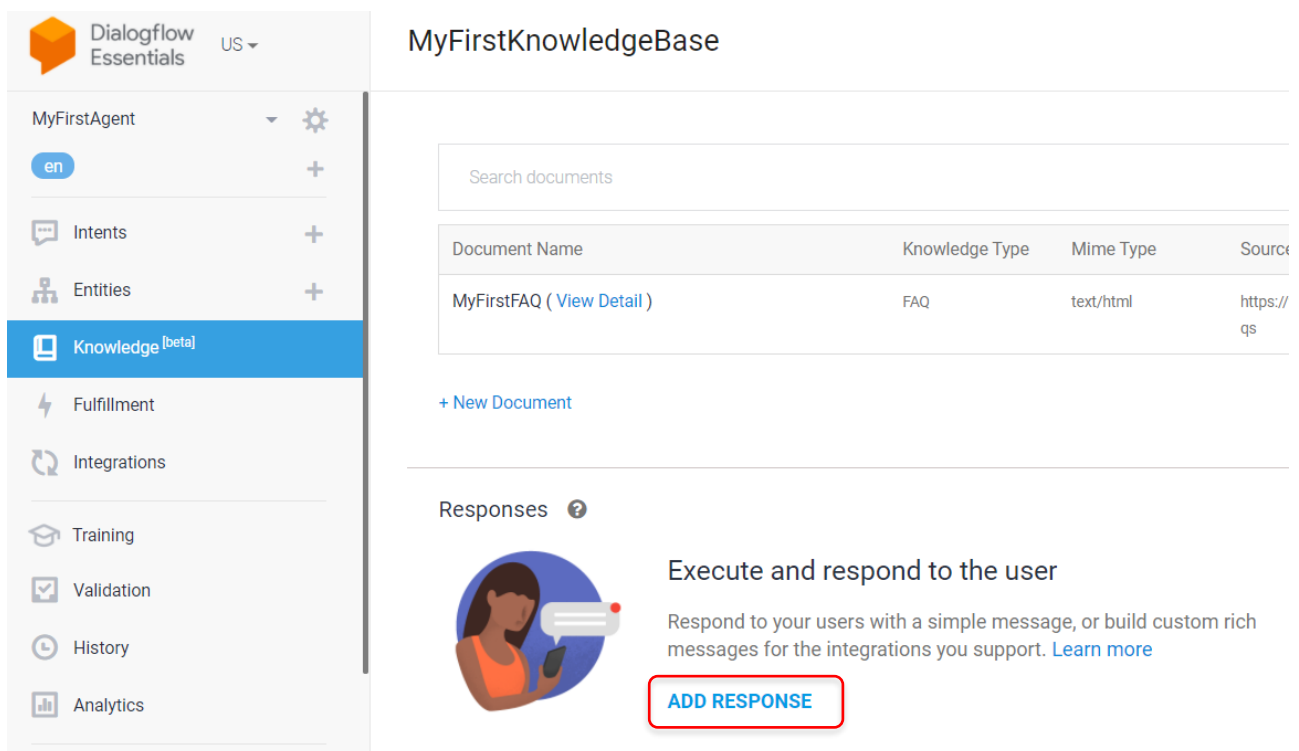
- b) In the Telegram configuration pop up, enter the telegram bot's access token you obtained in Step 4. Click **START** and after a while, you should see a flash message saying the bot is started.



c) Click on Close (top right) to finish the configuration.

6) Add Responses

a) Return back to your created Knowledge Base and click **Add Response** to create a new response.



Document Name	Knowledge Type	Mime Type	Source
MyFirstFAQ (View Detail)	FAQ	text/html	https://qs

+ New Document

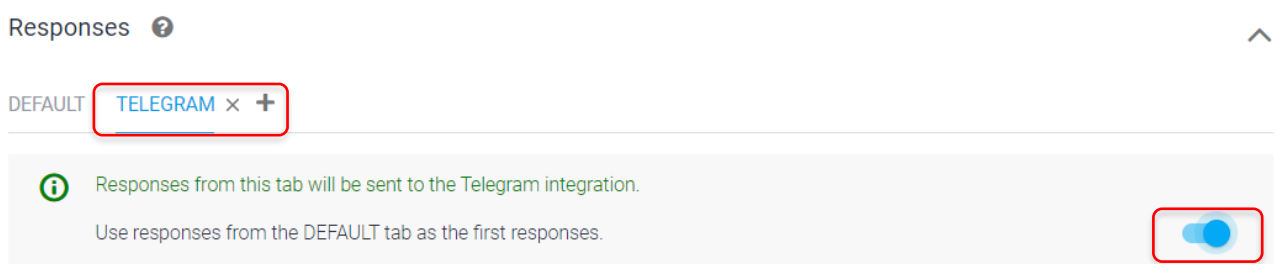
Responses ?

Execute and respond to the user

Respond to your users with a simple message, or build custom rich messages for the integrations you support. [Learn more](#)

ADD RESPONSE

b) Switch on to use responses from the DEFAULT tab to TELEGRAM an. Click **Save**.



Responses ?

DEFAULT **TELEGRAM** x +

Responses from this tab will be sent to the Telegram integration.
Use responses from the DEFAULT tab as the first responses.

TELEGRAM

7) Small Talk

- a) Your agent can learn how to support small talk without any extra development. By default, it will respond with predefined phrases. (By default, if you send “How are you?”, agent will respond with “Sorry, can you say that again?”)

Dialogflow Small Talk SAVE

MyFirstAgent en

- Intents
- Entities
- Knowledge (beta)
- Fulfillment
- Integrations
- Training
- History
- Analytics
- Prebuilt Agents
- Small Talk**

Your agent can learn how to support small talk without any extra development. By default, it will respond with predefined phrases. Use the form below to customize responses to the most popular requests.

User: How are you?
Agent: Wonderful as always. Thanks for asking.

User: You're so sweet.
Agent: Thanks! The feeling is mutual.

☒ Enable

Based on [Actions on Google policy](#), enabling Small Talk in its entirety will cause your action to be rejected. See [Import the prebuilt agent](#) for steps on how to import and select subsets of Small Talk features that comply with Action on Google's policy.

Small Talk Customization Progress 0%

About agent 0%

Courtesy 0%

8) Test your agent from Telegram

- a) Launch telegram and add your bot. You can search for your bot by adding a @ to the username you used in step 3. **same as how we searched for BotFather*
- b) Click start to start a conversation with the bot.
- c) Start asking your bot!
E.g. Do I qualify for Merdeka? How to appeal?

Activity wrap-up:

We learn how to

- ☐ Create a Dialogflow agent.
- ☐ Create a Dialogflow knowledge base
- ☐ Use Small Talk to add personality to agent
- ☐ Integrate Telegram to agent

Activity 2 – Training an Image Classifier

In this activity, we will:

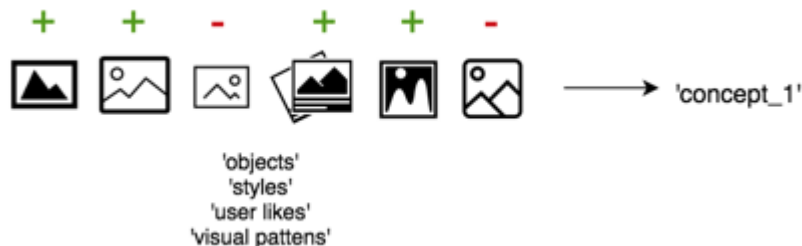
- ☐ Learn how to build a classifier through Clarifai AI service.
- ☐ Learn how to train a (supervised learning) image classification model
- ☐ Learn to evaluate the model
- ☐ Deploy your model to a chatbot

NOTE for on-campus training:

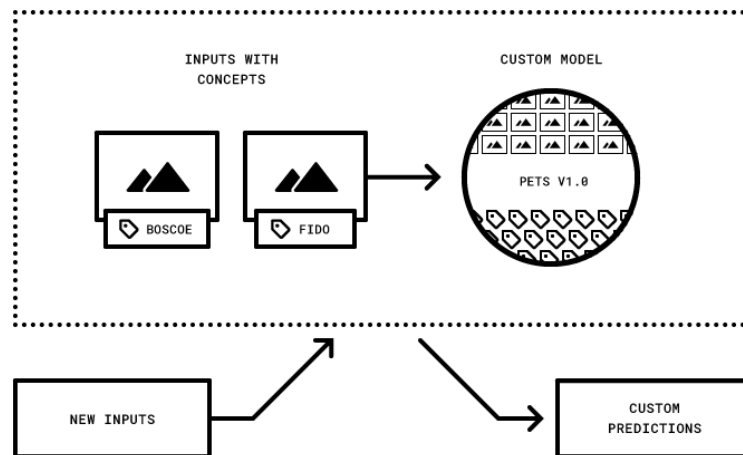
- ☐ **Some security and network settings may disable proper registration on Clarifai. Please check that you are using Guest@RP wireless network!**

Custom Training

Custom Training is about teaching computers to see the world in a way that is specific to your own content and context. You can think of Custom Training as a series of inputs where you ultimately teach a neural network what **concept_1** is and what it is not. We define the training data, taxonomy (the model's 'concepts', this is equivalent to “labelling”) and other performance characteristics.



If your content is visually distinct and easy to identify, 25-50 positive examples per concept will provide robust and accurate predictions. A **'concept' is synonymous with 'tag', 'category' or 'keyword'**. Concepts are your business' world view as to what an object, visual pattern, or style may represent.



What makes for a well-built concept?

- **Accurate labels.** Mis-labelled images introduces noise into your model and can lead to weak or confusing predictions.
- **Balanced training data.** Skewed training sets where several concepts have 5-20x as many positive training images as others may affect model performance.
- **Matching training and prediction context.** It's crucial that your training images for your concepts resemble the conditions and context of imagery you'll be making predictions on.

As an example, training a flower identification model solely with stock photography and then attempting to predict on user generated smartphone photos will not be ideal.

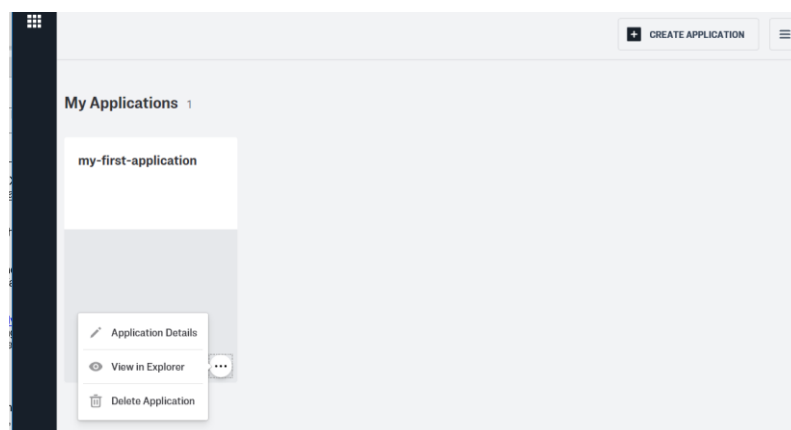
1. Create an account with Clarifai

- a) Sign up for an account at clarifai.com (<https://portal.clarifai.com/signup>)

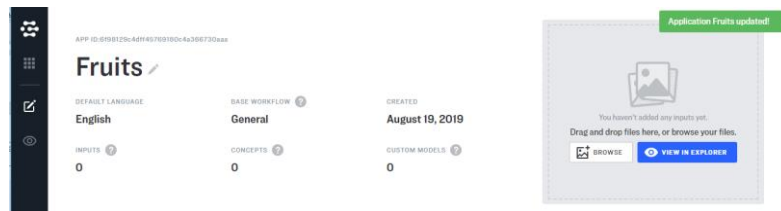
2. Edit your application

Operations are tied to an account and application, and any model that you create and add images to will be contained within a specific application. By default, you'll have an application in your account already so let's change that one's name to whatever you'd like. We are going to name ours "Fruits" for practical reasons

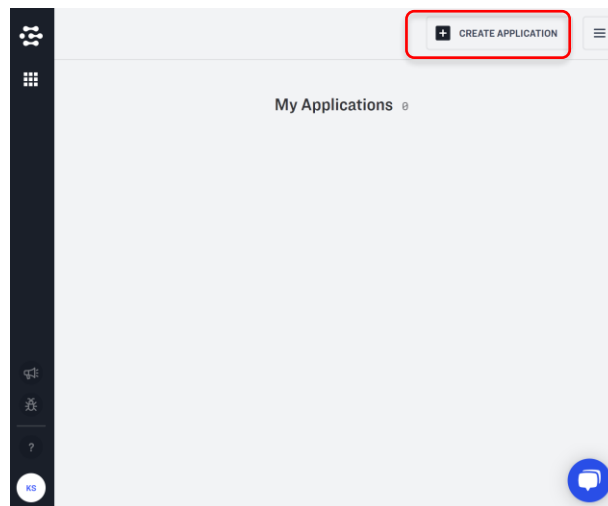
- a) Click on the details button of the default application and select Application Details.



- b) Change the application name to Fruits.



- c) Alternatively, you can also use “Create New Application” to create a new application and fill in the details as shown below.



Create an application

Create your application here!

APP-ID ?

Fruits

DISPLAY NAME (OPTIONAL)

short descriptive name

DESCRIPTION (OPTIONAL)

short description about your application

DEFAULT LANGUAGE

English (en)

BASE WORKFLOW ?

General

Note: an API Key will be created automatically for this application.

CANCEL CREATE

For non-locked down laptops, you can perform step 3 – 9. If you are using a locked-down and cannot access the URL in step 3, skip to step 10.

3. Copy the API key from the Application created: https://cognimate.me:2635/vision_home

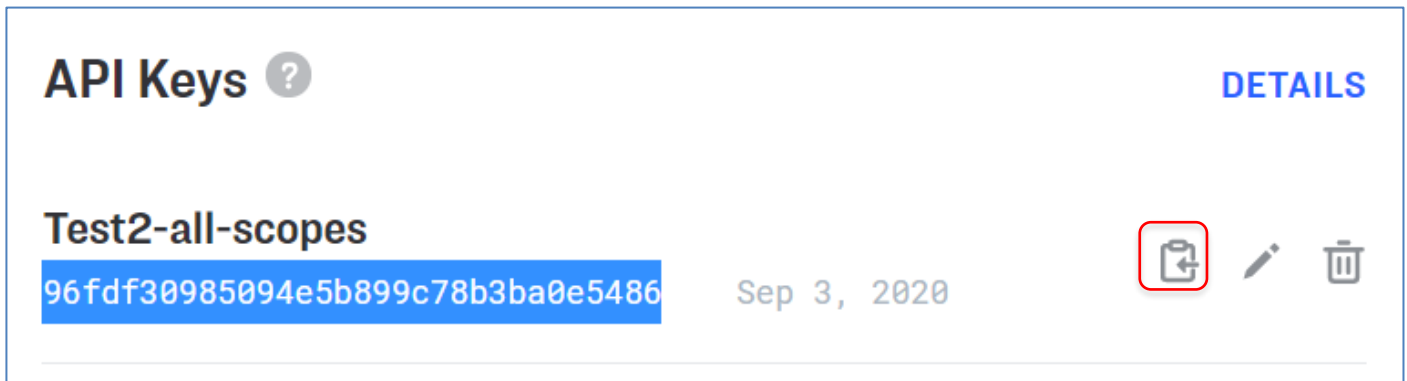


API Keys ?

DETAILS

Test2-all-scopes

96fdf30985094e5b899c78b3ba0e5486

Sep 3, 2020

4. Give your project a name and click on the “Set” button.

What is your project name?

Set

5. Paste the API key that you had copied earlier into the empty field and click on the “Set” button.

What is your Clarifai key? ?

Set Key

[How-to guide](#) [Sign up & get keys](#)

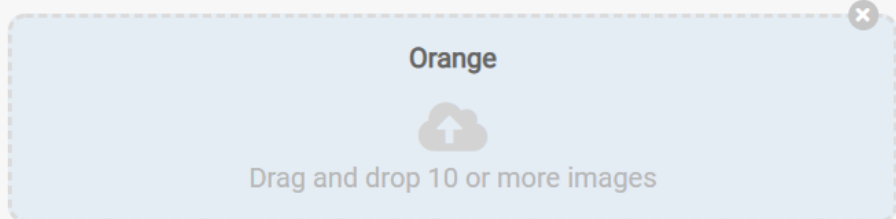
6. Key in the category of item that you want to train and click on “Add Category”.

What categories should your model have? ?

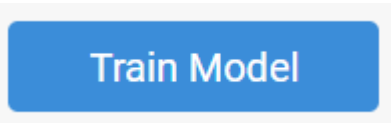
Add Category

7. Drag and drop 10 or more images of the item of choice. (Suggest to have two or more categories)

What categories should your model have? ?

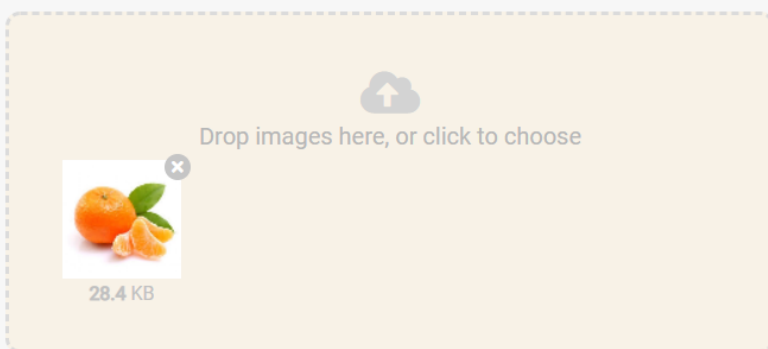


8. Click on “Train Model” once you the images are uploaded.



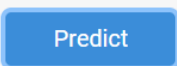
9. Drag and drop an image of choice from the “test_image” folder which you had downloaded earlier followed by clicking on the “Predict” button.

Upload an image to test your model



— or —

Enter an image URL



Category: Orange
Confidence score: 100.00%

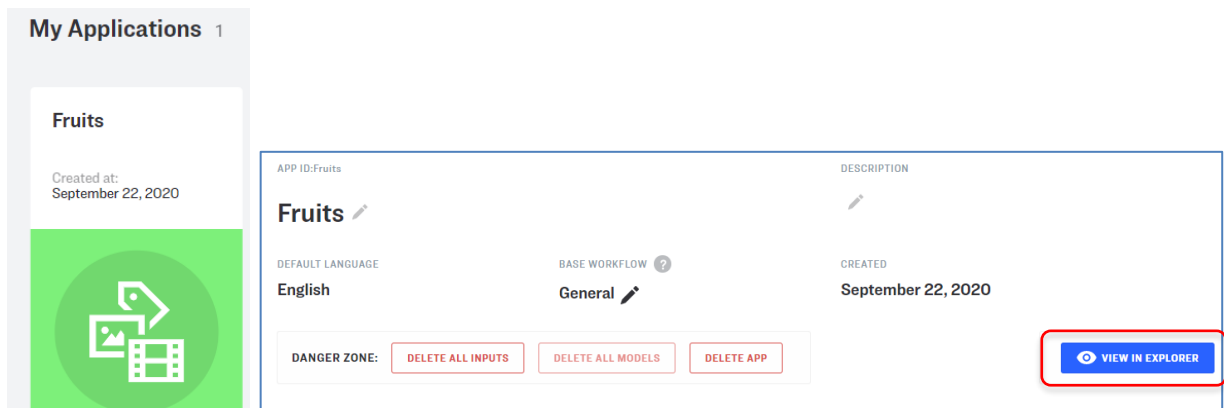
*The predicted category will show together with the confidence level.


^If you have done the above, please skip to step 15.

For locked down laptops, proceed from this step onwards.

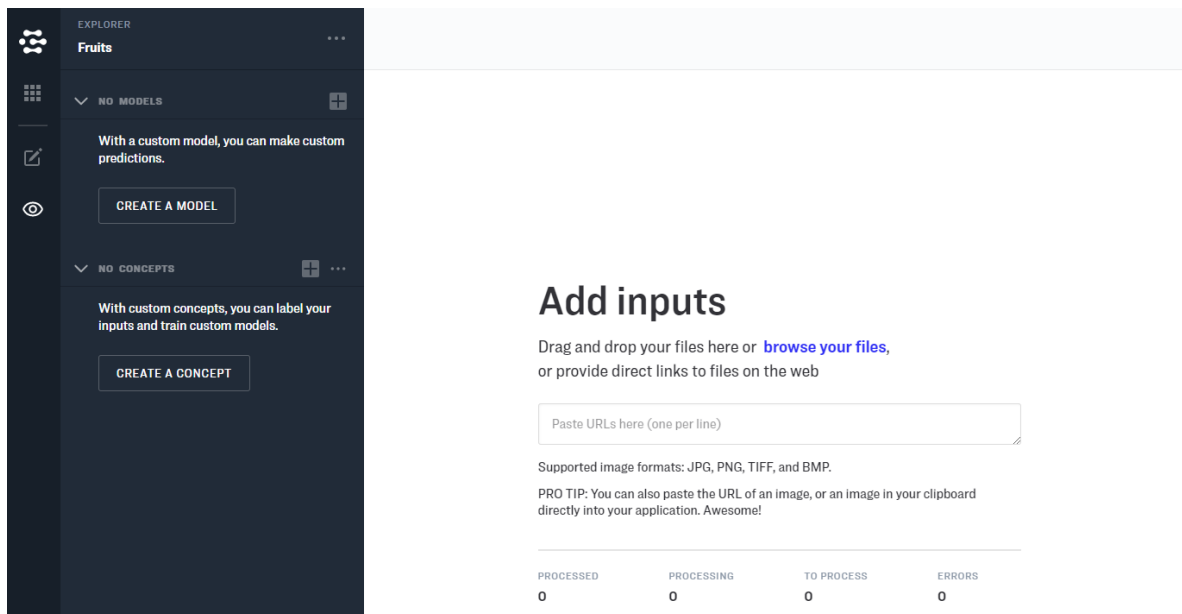
10. Add images to your application:

- a. Select your created **Application** and click on **VIEW IN EXPLORER** button to start adding images to your application.



- b. Click  on the top right and **BROWSE FILES**.

Custom models are built by training on your own data, and they will be able to make predictions specific to your own unique content and context.



- c. Start uploading your images, say, those banana images. It may take a while to upload the images. Once uploaded, you will see tick mark with the uploaded images.

Drag & Drop your files here

BROWSE FILES


Supported image formats: JPG, PNG, TIFF, BMP, WEBP, CSV, and TSV

Or provide direct links to files on the web

PRO TIP: You can also paste an asset URL, or an asset in your clipboard directly into your application. Awesome!

PROCESSED	PROCESSING	TO PROCESS	ERRORS
30	0	0	0

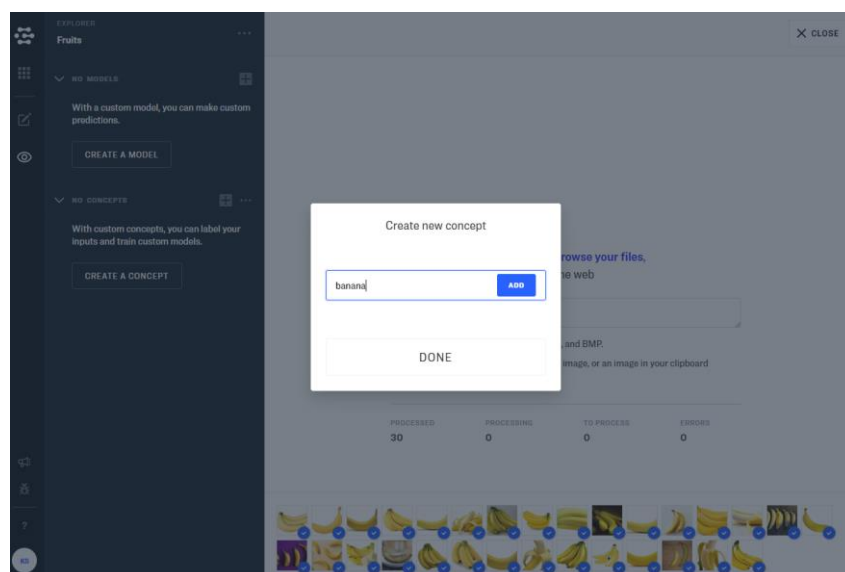
Please don't close this page while upload is in progress.



*Note: It may take a while to upload the images.

11.Add concept(s)

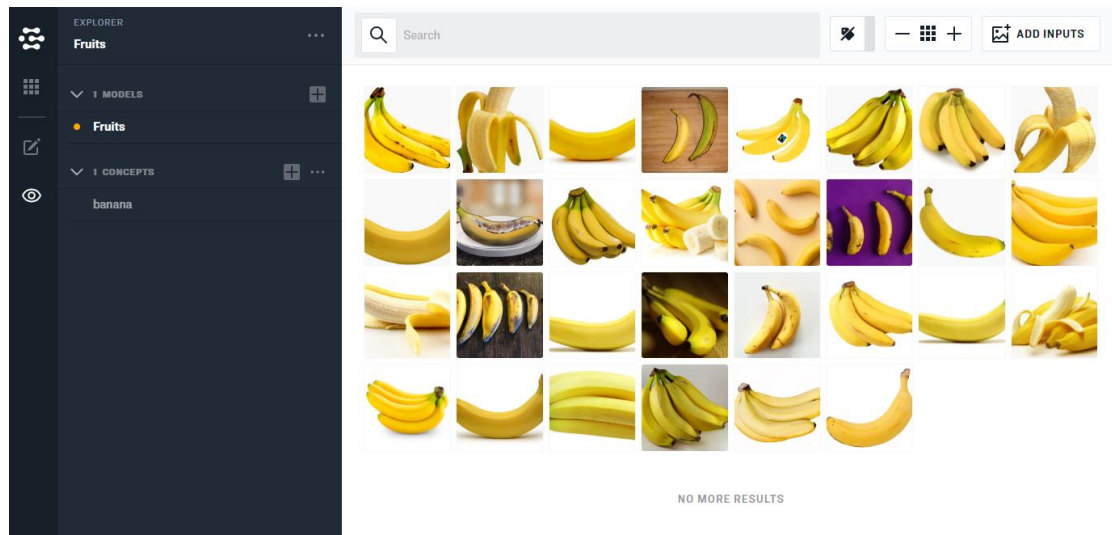
- Click on **CREATE A CONCEPT**.
- Enter banana and then click on **ADD** and then **DONE**.



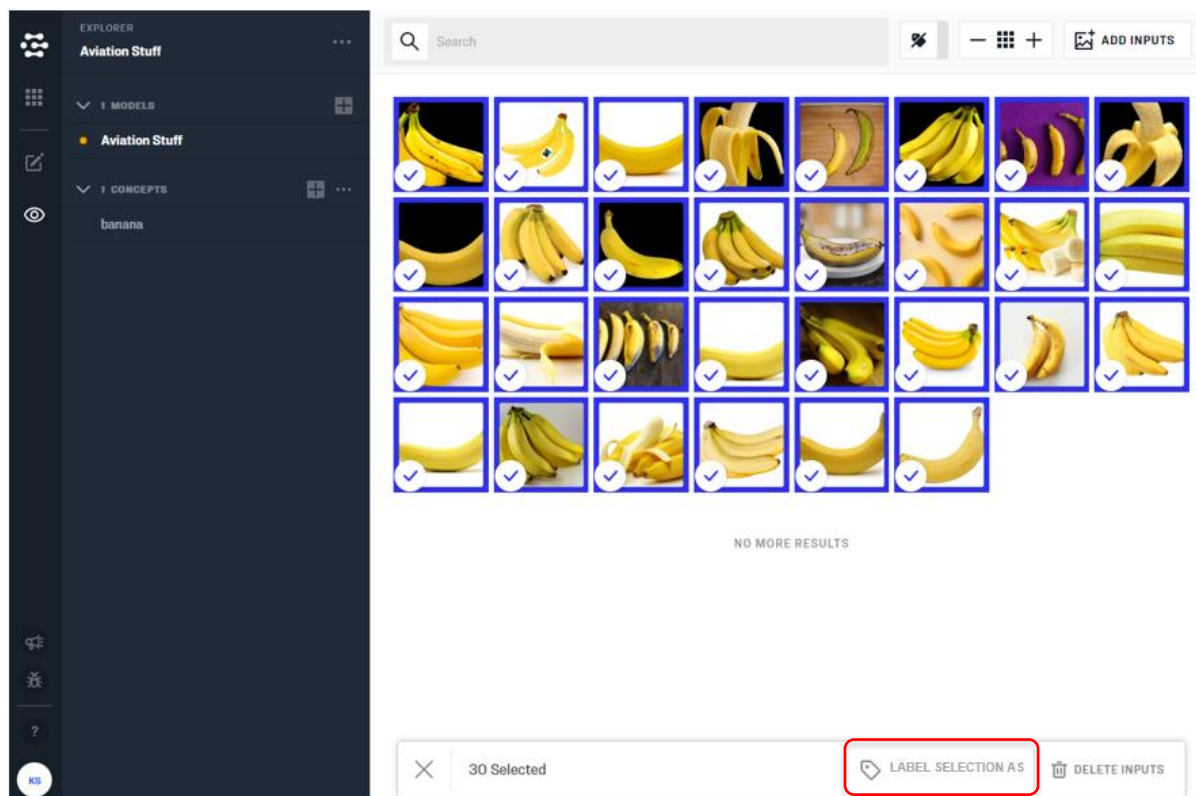
12.Label your images with those concepts

Now that you have some concepts, let's add them to your images! All images in your application are referred to as inputs so if you ever see that lingo, the names are essentially interchangeable. As you add concepts to images, you'll notice the counts in the Concept Panel on the left showing the updated total of how many images are labelled with each, respectively.

- a) Click on CLOSE (if necessary) to bring you to the following screen.



- b) Select all the banana images and select **ADD CONCEPTS**.



- c) Select **banana**

Label selected Inputs (30)

ADD CONCEPTS TO INPUT

Create new concept ADD

Include Concepts

☒ banana

☐ Create new model with selected concepts

Embed model version ID(Optional)

bb186755eda04f9cbb6fe32e816be104

LABEL 30 INPUTS WITH SELECTED CONCEPTS

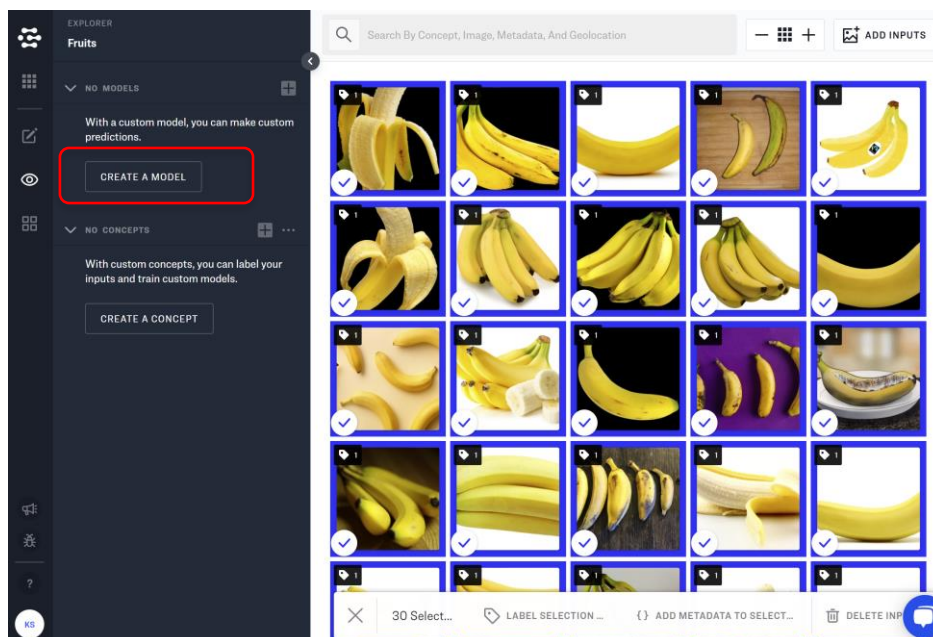
d) Click **LABEL 30 INPUTS WITH SELECTED CONCEPTS**.

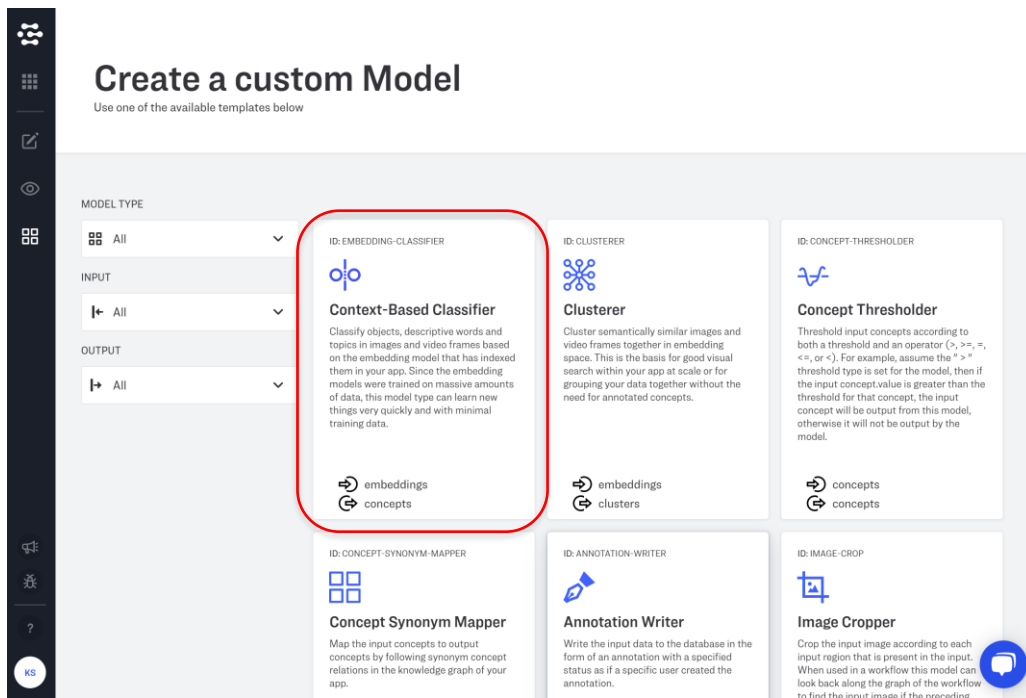
13.Add more images and concepts

Repeat step 3 to 5 for the rest of the fruits folders (except test_images)

14.Create and Train the model

Once all of your images are labelled, let's go ahead and create your first model by clicking CREATE A MODEL.





- a) Click on **Context-based Classifier**. Enter a name (*avoid space within name*) for the model and other details as shown. Click on **CREATE MODEL**.

Create a Context-Based Classifier Model
 Classify objects, descriptive words and topics in images and video frames based on the embedding model that has indexed them in your app. Since the embedding models were trained on massive amounts of data, this model type can learn new things very quickly and with minimal training data.

MODEL ID (OPTIONAL) ?
 user-defined ID

DISPLAY NAME ?
 Fruits

OUTPUT_INFO.DATA.CONCEPTS * ?

CONCEPT ID	CONCEPT NAME
coconut	coconut
banana	banana

Concept Name

OUTPUT_INFO.OUTPUT_CONFIG.CONCEPTS_MUTUALLY_EXCLUSIVE ?
 If you expect multiple concepts for this model to be present per image, set 'Concepts Mutually Exclusive' to false. ☐ NO

OUTPUT_INFO.OUTPUT_CONFIG.CLOSED_ENVIRONMENT ?
 If you will be training with images that do not contain any concepts for this model, set 'Closed Environment' to false. ☐ NO

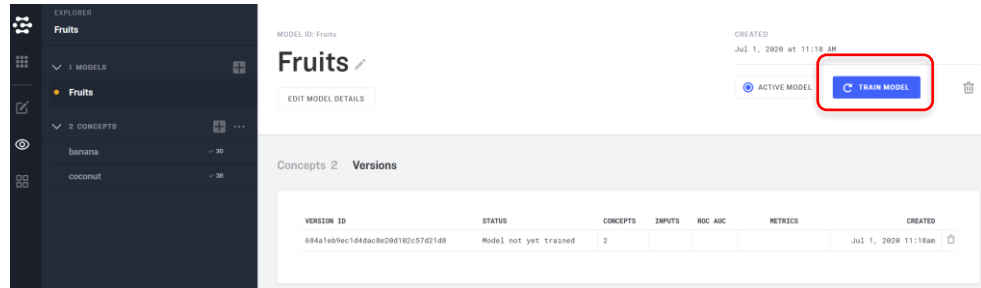
OUTPUT_INFO.OUTPUT_CONFIG.EMBED_MODEL_VERSION_ID ?
 bb186755eda04f9cbb6fe32e816be104

CREATE MODEL

Let's go ahead and train your first model. You can either do this via the little 3-dot menu next to the model name or you can click on the model name and then click the Train Model button on the ensuing page.

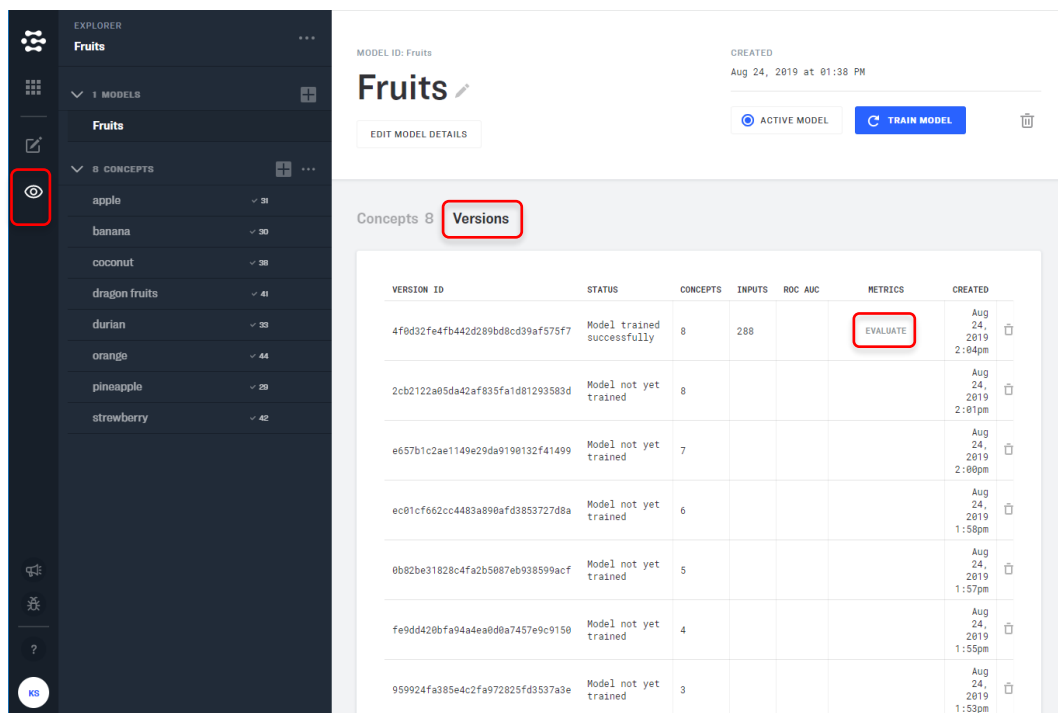
- b) Click on the application (ie Fruits). You should see the following. Click on **TRAIN MODEL**.

** you will see at the bottom right screen on the status of model being trained and completed.*



15. Evaluate your model

- a) Click the **Explorer**, you should be able to see the model name. **Refresh** the webpage if the model name isn't there.
- b) Click on the model name that appears on the left panel to bring you to the screen above.
- c) Click on **Versions**. It will take you to the following screen:



- d) To evaluate your model, click the **evaluate** option under Metrics. This will take a short amount of time depending on the number of images added to your model. Our simple model should be evaluated in seconds. Once the evaluation is completed, the "Evaluate" option will be changed to a "View" option. Click it, and you will see the evaluation results.

-
- The diagram illustrates the process of splitting labeled data into training and testing sets. At the top, a box labeled "Application Data" contains "ALL LABELED INPUTS". An arrow points down to a large container representing the data split process. This container is divided into two main sections: "TEST DATA" and "TRAINING DATA". The "TEST DATA" section shows a single blue square labeled "SPLIT 1". The "TRAINING DATA" section shows four gray squares, with the first one labeled "SPLIT 2". This pattern continues for "SPLIT 3" and "SPLIT K", where the blue square moves to the second, third, and K-th positions respectively, while the others remain gray. Ellipses (...) indicate that there are more splits and training data points than shown.

There is a value called ***probability threshold***, which determines the point at which concepts will be classified as either positive or negative. For example, an image is counted as belonging to a particular concept, such as a pineapple, only if its prediction probability of that image for the pineapple is higher than the threshold value. The default threshold value is 0.5. You can change it as you want.

- Version 1.5

Concepts 8 Versions

▼ Evaluation Summary Table ⓘ

Model Accuracy Score (ROC AUC MAC AVG):1

Current Prediction Threshold is 0.5. This means an input 'counts' as a predicted concept if the prediction probability for that concept is greater than or equal to 0.5.

Of the 7 images actually labeled **apple**:

True Positive: 7 were predicted as **apple** with probability greater than or equal to 0.5

False Negative: 0 were predicted as **apple** with prediction probability less than 0.5

Recall Rate: 100% (=7/7) of the images actually labeled **apple** were predicted as **apple**.

Of the 8 images predicted as **apple** with prediction probability greater than or equal to 0.5:

True Positive: 7 were labeled as **apple**.

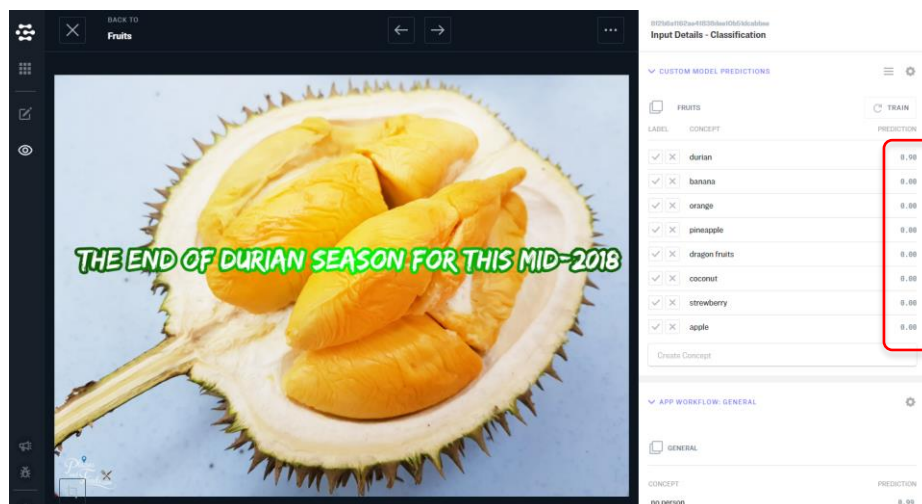
False Positive: 1 were not labeled as **apple**

Precision Rate: 87.5% (=7/8) of the images predicted as **apple** were labeled as **apple**.

CONCEPT	K-SPLIT AVG ACCURACY SCORE (ROC AUC)	I-SPLIT						
		TOTAL LABELED	TOTAL PREDICTED	TRUE POSITIVES	FALSE NEGATIVES	FALSE POSITIVES	RECALL RATE	PRECISION RATE
durian	1.000	6	6	6	0	0	1.000	1.000
apple	1.000	7	8	7	0	1	1.000	0.875

16. Test your model (Upload some external images and see how the model is performing)

- Click on the application, **ADD INPUTS** to upload a test image. (for this exercise, use an image from the test_images directory)
- Click on the test image to test the image classification.

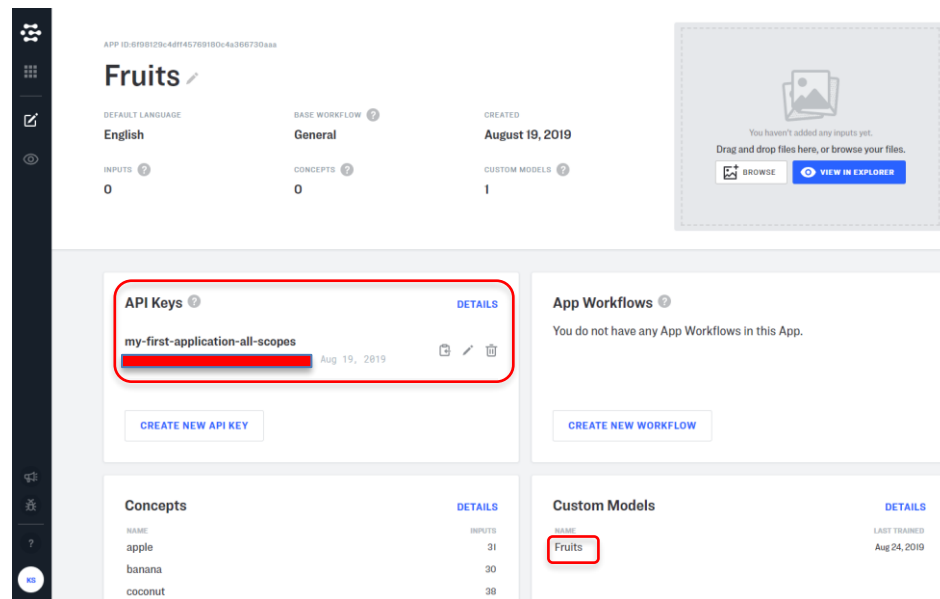


17. Adding negative examples when the model is getting confused (optional)

Up until this point we have only added positive examples to the model (e.g. saying "this is definitely concept X"), but we haven't added any negatives that say the opposite (i.e. "this is not concept X"). A robust and well performing concept is typically made up of both positive and negative examples with a 3 or 4:1 ratio, respectively, but adding too many will be counterproductive so be careful. If you see in one instance that you're getting a false positive for "apple", for example, you may want to teach the system that it's wrong.

18. Model Deployment and usage

- Once you trained a model, an API key is automatically created for you. To use this model in an application, you will need this API key and the model name. Click on Application Details to get this information from the screen.



19. Use the model via a web app

- Launch your web browser and go to <http://kwseow.pythonanywhere.com/forms>. This is a demo site created to utilise your model to perform image classification.
- Key in your Clarifai API key and the name of the model you trained in the previous step.
- Select a photo you want to test with and click on submit. After a short while you should see the result of the classification of your model. See the following screen capture:

AI4E Clarifai image classification demo

Clarifai API Key

Clarifai custom model name

No file chosen

Success! durian|0.9999454
pineapple|0.0007022619
coconut|3.5732985e-05
dragon fruits|1.692772e-05
orange|6.3478947e-06
banana|3.1590462e-06
strewberry|1.7881393e-07
apple|2.9802322e-08

Using model:
Fruits|Fruits



d) The following code snippet provide gives you an ideal of how this is done.

```

01 @app.route("/forms", methods=['GET', 'POST'])
02 def myforms():
03     form = ReusableForm(request.form)
04
05     #print(form.errors)
06     if request.method == 'POST':
07         form = ReusableForm()
08
09         if form.validate_on_submit():
10             #write_to_disk(name, surname, email)
11             f = form.photo.data
12             filename = secure_filename(f.filename)
13             f.save(os.path.join('./mysite/static/photos', filename ))
14         try:
15             #api_key = ""
16             api_key = form.api_key.data
17             os.environ["CLARIFAI_API_KEY"] = api_key
18             clarifai_app = ClarifaiApp()
19             #custom model
20             #model_name="Fruits"
21             model_name=form.model_name.data
22             model = clarifai_app.models.get(model_name=model_name)
23             response = model.predict_by_filename(os.path.join('./mysite/static/photos',
24 filename ))
25             msg = ""
26             for concept in response['outputs'][0]['data']['concepts']:
27                 msg += "%s\n"%(concept['name'],concept['value'])
28             tmp_model = response['outputs'][0]['model']
29             msg += "\nUsing model:\n%s\n"%(tmp_model['id'],tmp_model['name'])
30             msg = msg.replace('\n', '<br>')
31         except ApiError as e:
32             msg = 'Error status code: %d\n' % e.error_code
33             msg += 'Error description: %s\n' % e.error_desc
34             if e.error_details:
35                 msg += 'Error details: %s' % e.error_details
36
37             if e.error_code == 21200:
38                 api_key=form.api_key.data
39                 os.environ["CLARIFAI_API_KEY"] = api_key
40                 clarifai_app = ClarifaiApp()
41                 msg += "\nAvailable models:"
42                 for model in clarifai_app.models.get_all():
43                     msg += "\n%s"%model.model_name
44
45             msg = msg.replace('\n', '<br>')
46
47             #flash('Hello: {}'.format(filename))
48             flash('{}'.format(msg))
49             #filename = secure_filename(form.file.data.filename)
50             #form.file.data.save('uploads/' + filename)
51             #flash('Hello: {} {} {}'.format(name, surname,filename))
    
```

```
47
48     else:
49         flash('Error: All Fields are Required')
50
51     return render_template('form.html', form=form)
```

Activity wrap-up:

We learn how to:

- ☐ Train and evaluate an image classifier
- ☐ Deploy the model for use in a chatbot and a web browser