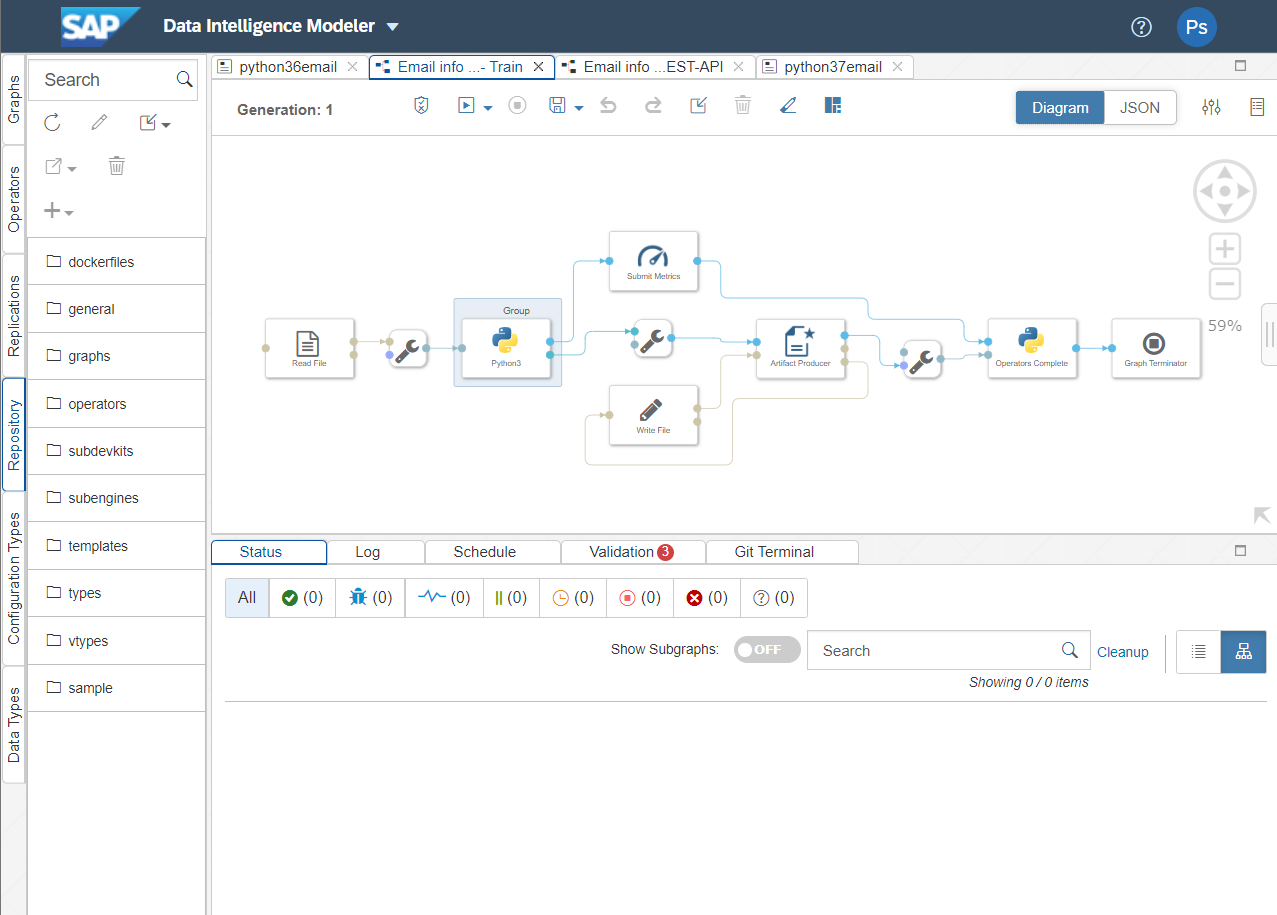
# Email Information Extraction Pipeline on SAP DI:

1. Created a Python Producer Pipeline for training on SAP DI in DI Modeller as shown below:

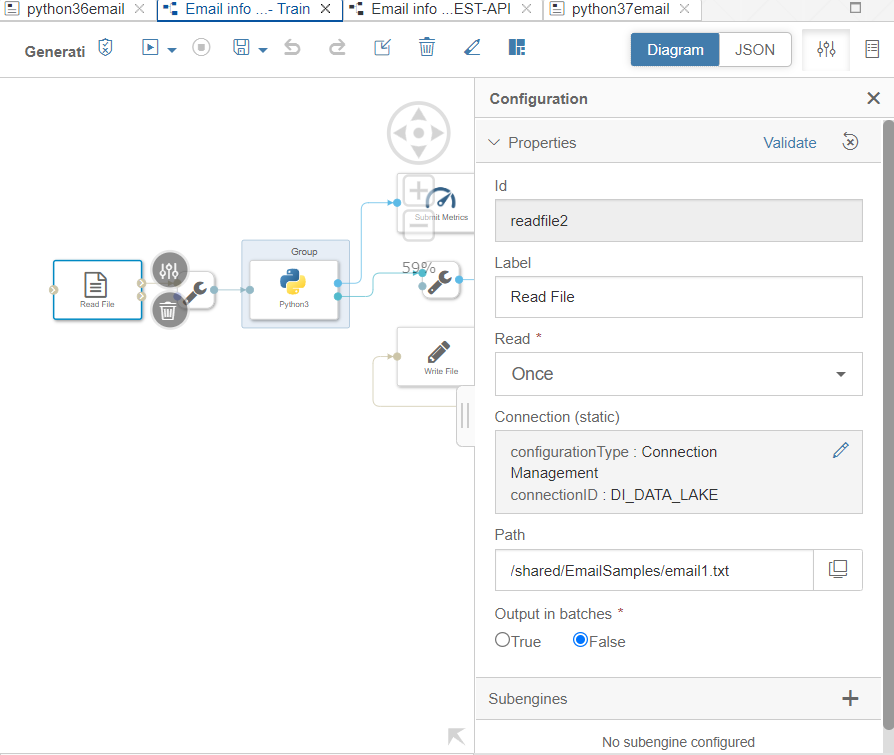


1. The input file location of sample email is as show below:

Graphical user interface, application, Word

Description automatically generated

1. Configure the Read File as below:



1. Create a Docker File with the below written commands:

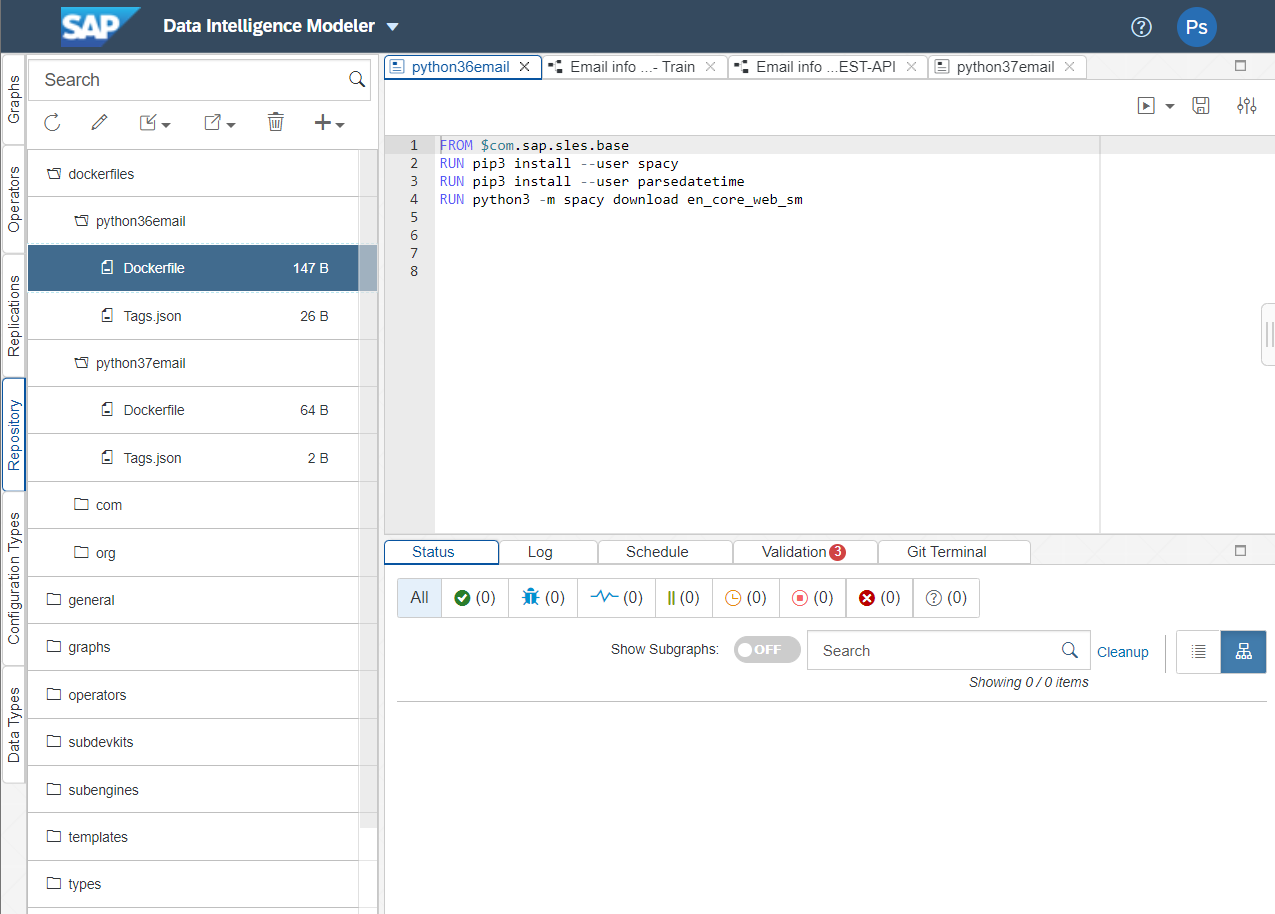
FROM $com.sap.sles.base

RUN pip3 install --user spacy

RUN pip3 install --user parsedatetime

RUN python3 -m spacy download en\_core\_web\_sm

Docker file image shown below:



1. Write Training Code in the python script operator as below:

# Example Python script to perform training on input data & generate Metrics & Model Blob

def on\_input(data):

    import spacy

    nlp = spacy.load("en\_core\_web\_sm")

    # to send metrics to the Submit Metrics operator, create a Python dictionary of key-value pairs

    metrics\_dict = {"kpi1": "1"}

    # send the metrics to the output port - Submit Metrics operator will use this to persist the metrics

    api.send("metrics", api.Message(metrics\_dict))

    # create & send the model blob to the output port - Artifact Producer operator will use this to persist the model and create an artifact ID

    import pickle

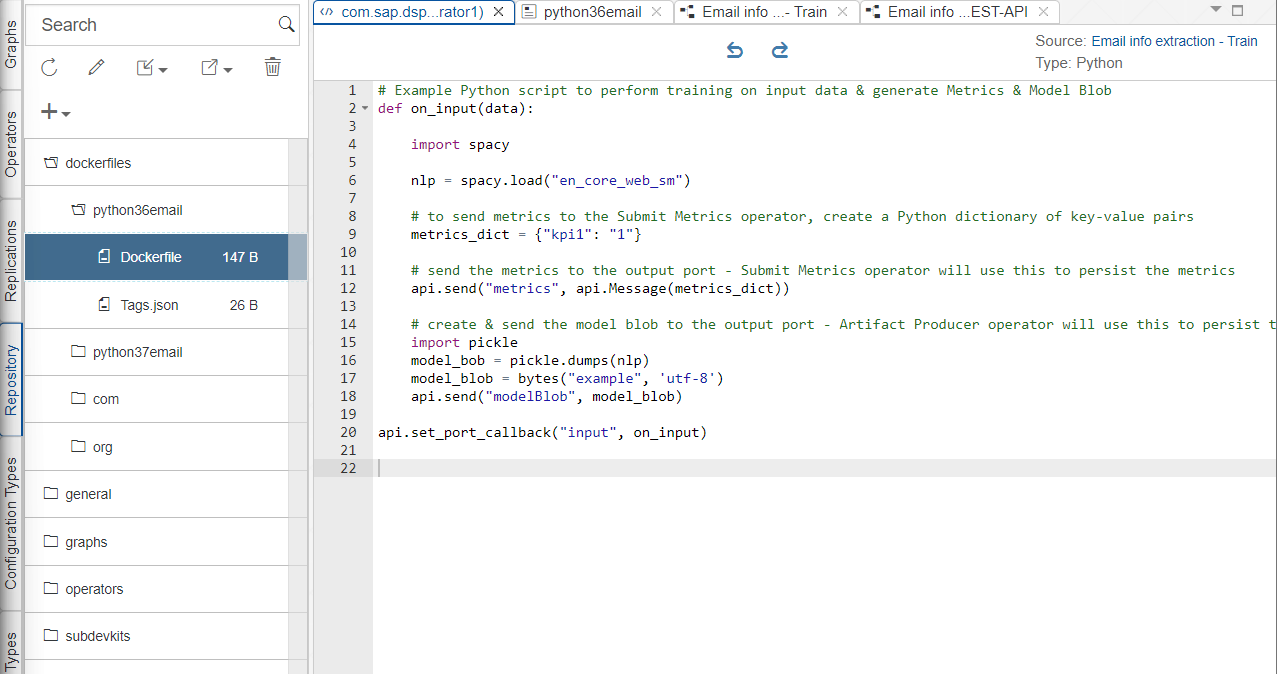
    model\_bob = pickle.dumps(nlp)

    model\_blob = bytes("example", 'utf-8')

    api.send("modelBlob", model\_blob)

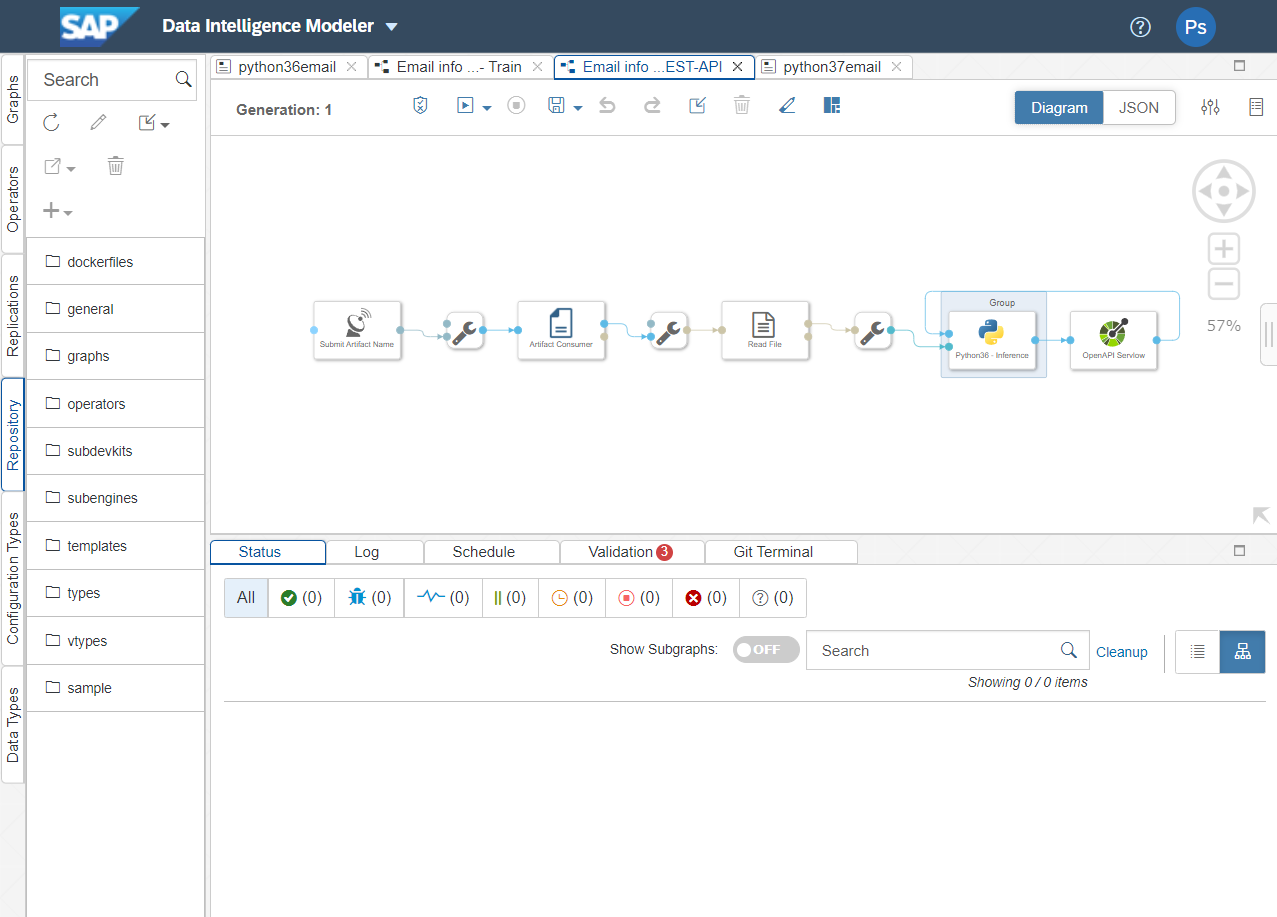
api.set\_port\_callback("input", on\_input)

shown below :



Save the graph and execute the graph, training will be done.

1. Now Create a different Python Consumer graph for creating REST API as shown below.



1. Write the below inference code in Python Script Operator and group with the same docker file which was created for Training pipeline.

import json

import spacy

import re

import pickle

import datetime

from datetime import datetime

import parsedatetime as pdt

# Global vars to keep track of model status

model = None

model\_ready = False

# Validate input data is JSON

def is\_json(data):

  try:

    json\_object = json.loads(data)

  except ValueError as e:

    return False

  return True

# When Model Blob reaches the input port

def on\_model(model\_blob):

    global model

    global model\_ready

    import pickle

    #model = pickle.loads(model\_blob)

    model = spacy.load("en\_core\_web\_sm")

    model\_ready = True

    api.logger.info("Model Received & Ready")

def get\_date(date\_list, current\_date):

  from datetime import datetime

  import parsedatetime as pdt

  cal = pdt.Calendar()

  now = current\_date

  ans = []

  #print("now: %s" % now)

  for time\_string in date\_list:

    if(time\_string.find('to') != -1):

        a,b = time\_string.split('to')

        a = cal.parseDT(a,now)[0]

        b = cal.parseDT(b,now)[0]

        return (a.strftime("%d/%m/%Y"),b.strftime("%d/%m/%Y"))

    x = cal.parseDT(time\_string, now)[0]

    #print("%s:\t%s" % (time\_string, x))

    ans.append(x.strftime("%d/%m/%Y"))

  return get\_start\_end\_date(ans, current\_date)

def get\_start\_end\_date(date\_list, current\_date):

  if(len(date\_list) == 0):

    return (current\_date.strftime("%d/%m/%Y"), current\_date.strftime("%d/%m/%Y"))

  elif(len(date\_list) == 1):

    return (date\_list[0], date\_list[0])

  else:

    listt = []

    for l in date\_list:

      listt.append(datetime.strptime(l, '%d/%m/%Y').date())

    x = min(listt)

    y = max(listt)

    return (x.strftime("%d/%m/%Y"), y.strftime("%d/%m/%Y"))

# Client POST request received

def on\_input(msg):

    error\_message = ""

    success = False

    prediction = None # This line needs to be added

    final\_result = []

    try:

        api.logger.info("POST request received from Client - checking if model is ready")

        if model\_ready:

            api.logger.info("Model Ready")

            api.logger.info("Received data from client - validating json input")

            user\_data = msg.body.decode('utf-8')

            # Received message from client, verify json data is valid

            if is\_json(user\_data):

                api.logger.info("Received valid json data from client - ready to use")

                # apply your model

                # obtain your results

                #text = json.loads(user\_data)['email']

                text = '''

Dear all,

New week, new prizes: yesterday we had the joy of learning that Henriette Michaud had received the Prix de l'Essai 2022 from the Académie Française for Freud in Bloomsbury 3123559: https://www.academie-francaise.fr/sites/academie-francaise.fr/files/palmares\_2022\_vf.pdf

Tomorrow, Hélène Carrère d'Encausse will receive an honorary prize for her work at the Hossegor book fair (Alexandra Kollontaï large format: 7757456 and plural to be published in November: 6912274).

Other good news that we are delighted about:

Next week marks the start of Aurélie Valognes and Virginie Grimaldi's summer promotion:

La Ritournelle 4551395 will be on RTL from next Monday until 10 July and then on France Bleu for the biggest summer run from Thursday 28 July to Wednesday 3 August.

Il nous restera ça 4702273 will accompany holidaymakers during RTL's most popular hours from Thursday 28 July to Sunday 31 July and then from Friday 12 August to Monday 15 August.

Cardinal Robert Sarah (Catechism of the Spiritual Life 2504824) is in the spotlight this weekend in Le Figaro Magazine with a major four-page interview (enclosed). Next week, he will receive exceptional media visibility.

Challenges published this week a portrait of Nicolas Forissier for L'ennemi intérieur 6180412 (en pj).

Sarah Briand's novel, Les pépins de Grenade 1282225, is receiving good media coverage. Femme Actuelle gave it a special mention this week (en pj), while Anne-Marie Revol praised the novel in Patricia Loison's 23H last night: https://www.francetvinfo.fr/replay-jt/franceinfo/21h-minuit/23-heures/jt-le-23h-jeudi-30-juin-2022\_5230810.html

Janine Boissard's new novel, Quand la belle se réveillera 4082000, continues to be honoured, as shown by the fine reviews in the newspaper Centre presse (en pj) and Bruxelle culture (en pj).

And all the press review of the week in attachments!

Have a good weekend,

Sincerely yours,

Pauline

'''

                nlp = model

                ###########################################

                import re

                pattern = pattern = r"(\d{%d})"%7

                email = []

                for s in text.split('\n'):

                    if(re.findall(pattern, s)):

                        email.append(s)

                        #print(s,"\*\*\*")

                #print(len(email))

                import re

                final\_struct\_list = []

                for s in email:

                  struct\_list = []

                  doc2 = nlp(s)

                  #displacy.render(doc2, style='ent', jupyter=True)

                  pattern = r"(\d{%d})"%7

                  #print("Product ID: ", end=" ")

                  prod\_id = re.findall(pattern, str(doc2))

                  struct\_list.append(prod\_id)

                  #print(prod\_id)

                  date\_list = []

                  for ent in filter(lambda e : e.label\_ == 'DATE', doc2.ents):

                    date\_list.append(ent.text)

                  start\_date, end\_date = get\_date(date\_list,datetime(2022,7,28))

                  struct\_list.append(start\_date)

                  #print("Start Date: ", start\_date)

                  struct\_list.append(end\_date)

                  #print("End Date: ", end\_date)

                  #print("Description: ", end=" ")

                  temp\_str = ""

                  for e in doc2.ents:

                    temp\_str = temp\_str + str(e)+" "

                  struct\_list.append(temp\_str)

                  #print(temp\_str)

                  final\_struct\_list.append(struct\_list)

                  #print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

                #print(final\_struct\_list)

                final\_result = final\_struct\_list

                ###############################################

                success = True

            else:

                api.logger.info("Invalid JSON received from client - cannot apply model.")

                error\_message = "Invalid JSON provided in request: " + user\_data

                success = False

        else:

            api.logger.info("Model has not yet reached the input port - try again.")

            error\_message = "Model has not yet reached the input port - try again."

            success = False

    except Exception as e:

        api.logger.error(e)

        error\_message = "An error occurred: " + str(e)

    if success:

        # apply carried out successfully, send a response to the user

        msg.body = json.dumps({'marathon\_minutes\_prediction': final\_result})

    else:

        msg.body = json.dumps({'Error': error\_message})

    new\_attributes = {'message.request.id': msg.attributes['message.request.id']}

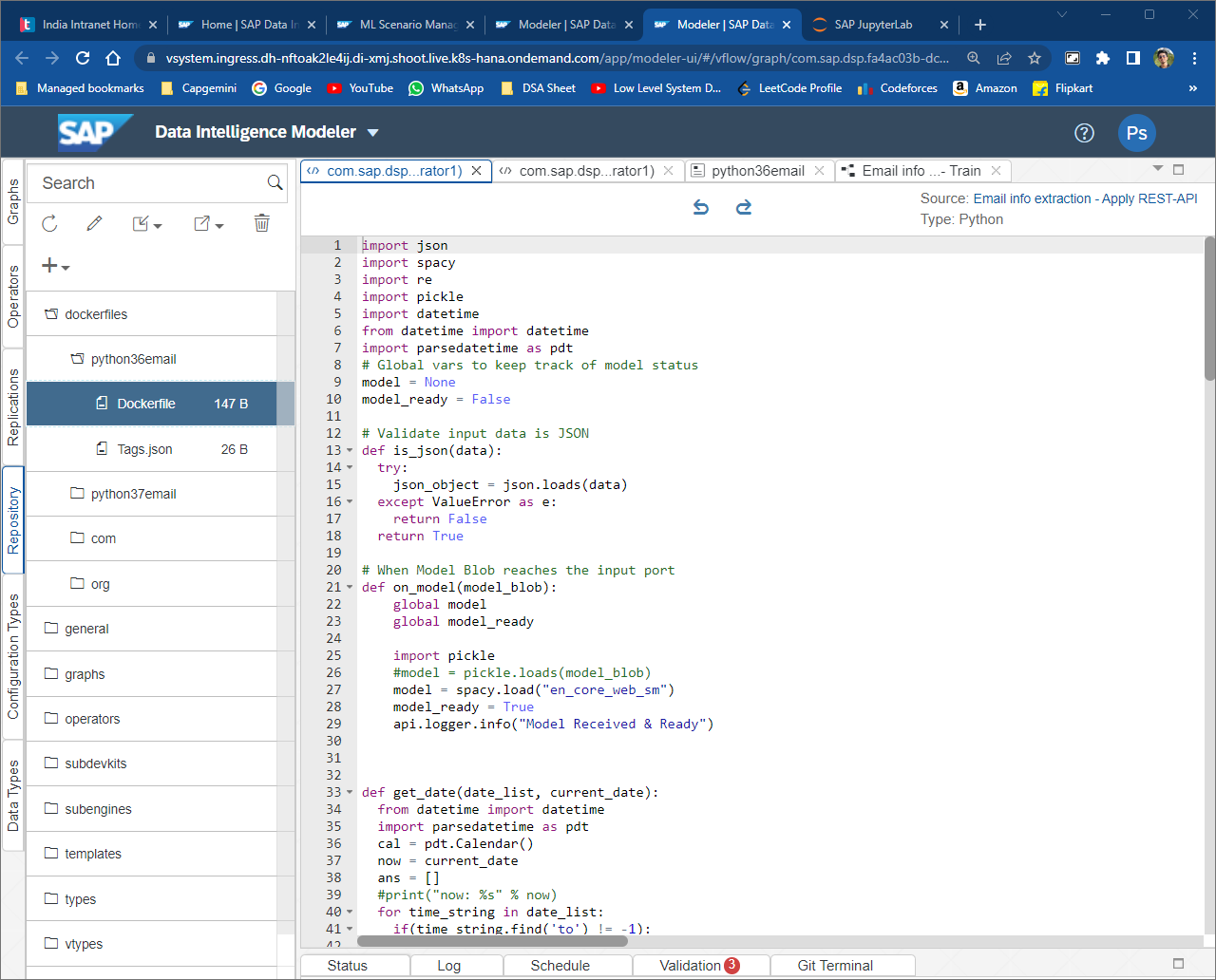
    msg.attributes =  new\_attributes

    api.send('output', msg)

api.set\_port\_callback("model", on\_model)

api.set\_port\_callback("input", on\_input)

Should look as shown below.



1. Save and Execute the graph, you can check the output with postman using the API link you get.

