

# Big Data, organisation and analysis

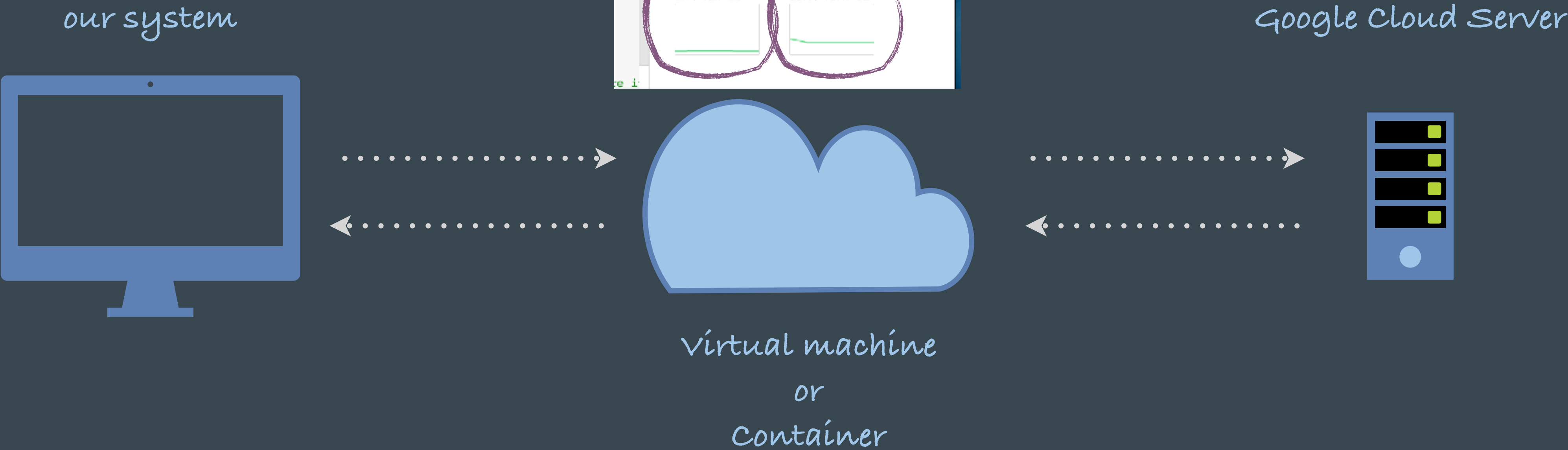
Building a REST API service

# Recap on what we learned

- So far, we have been learning...
  - What is virtualisation and the link with VMs and Containers
  - How these are forming Cloud computing services (e.g., Colab Notebooks)
  - How to create a “middleware” to access data
  - How to access REST APIs

# Google Cloud service

## Principle



# Data access via API

## Principle

Google Cloud  
virtual service

Client system



```
example_middleware.ipynb
File Edit View Insert Runtime Tools Help Last edited on 9 March

+ Code + Text
Connect

Alternative, use the Kaggle API

NOTE: You need an account for that! Create a kaggle.json file in the account settings.
Place the kaggle.json file created at into the notebook's directory and use opendatasets library.

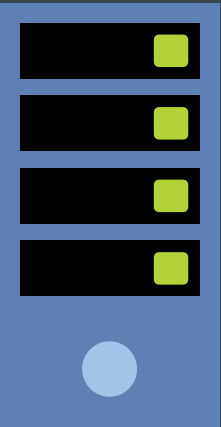
[ ] 1 # Check if opendatasets library is installed, if so - load it, if not - install it!
2 try:
3     import opendatasets as od
4 except ImportError as e:
5     !pip install opendatasets
6     import opendatasets as od
7

[ ] 1 # download the data
2 od.download("https://www.kaggle.com/datasets/sudalairajkumar/daily-temperature-of-major-cities")
3
4 # access them from the download folder
5 fn = "daily-temperature-of-major-cities/city_temperature.csv"
6 df = pd.read_csv(fn, low_memory=False)
7
8 # Check the memory usage
9 df.info(memory_usage="deep")

Downloading daily-temperature-of-major-cities.zip to ./daily-temperature-of-major-cities
100% [██████████] 12.9M/12.9M [00:01<00:00, 10.6MB/s]

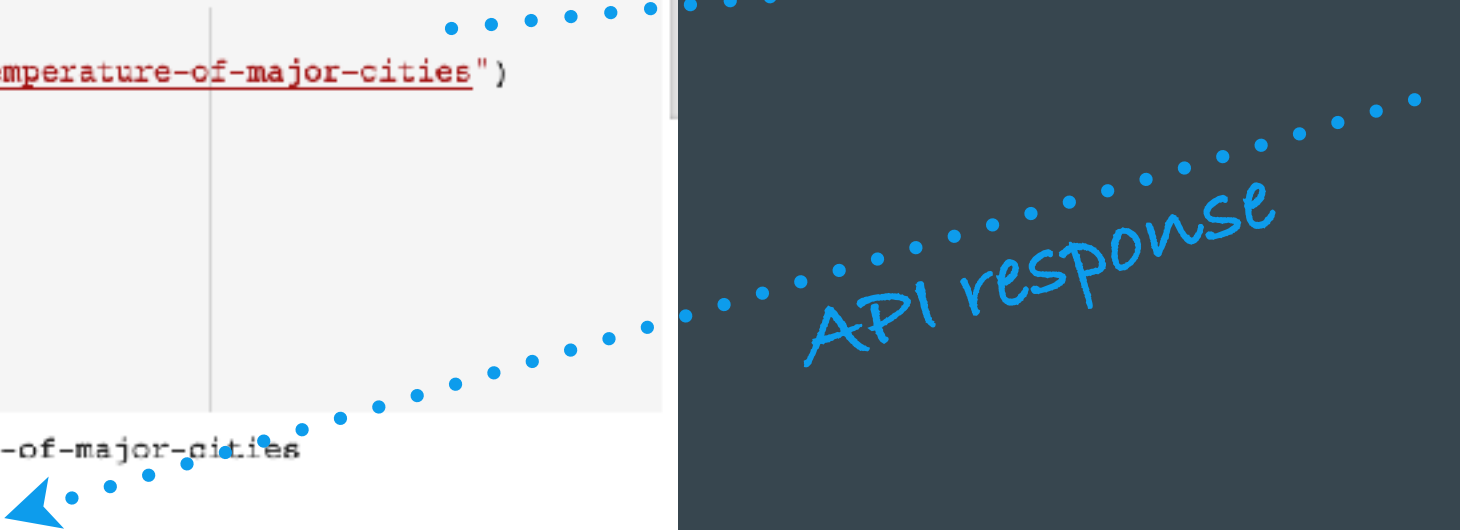
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2906327 entries, 0 to 2906326
```

Kaggle data server



API request

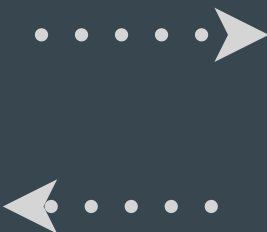
API response



# REST API access via **requests** package

## Principle

Client system



```
REST_api_access.ipynb
File Edit View Insert Runtime Tools Help Last edited on 16 March

+ Code + Text

Evapotranspiration data download Via REST API

1 Downloading data programmatically

To download data by program code we do:

1. Create a data directory
2. Load 1-by-1 the data files from the Github repository. Do so, you need to copy the raw data link and create from this the file name to store the data and the request.

[ ] 1 import requests
    2 from pathlib import Path
    3
    4 # Setup path to a data folder
    5 data_path = Path("data/")
    6
    7 # If the data folder doesn't exist, download it and prepare it...
    8 if data_path.is_dir():
    9     print(f"{data_path} directory already exists... skipping creation")
   10 else:
   11     print(f"{data_path} does not exist, creating one...")
   12     data_path.mkdir(parents=True, exist_ok=True)
   13
   14 # Download Ahja data
   15 with open(data_path / "Ahja-MOD16A2GF-006-Statistics.csv", "wb") as f:
   16     request = requests.get("https://raw.githubusercontent.com/stenoe/BDOA/main/Evapotranspiration%20data/")
   17     print("Downloading Ahja MODIS data...")
   18     f.write(request.content)
   19
```

API request

API response

REST API

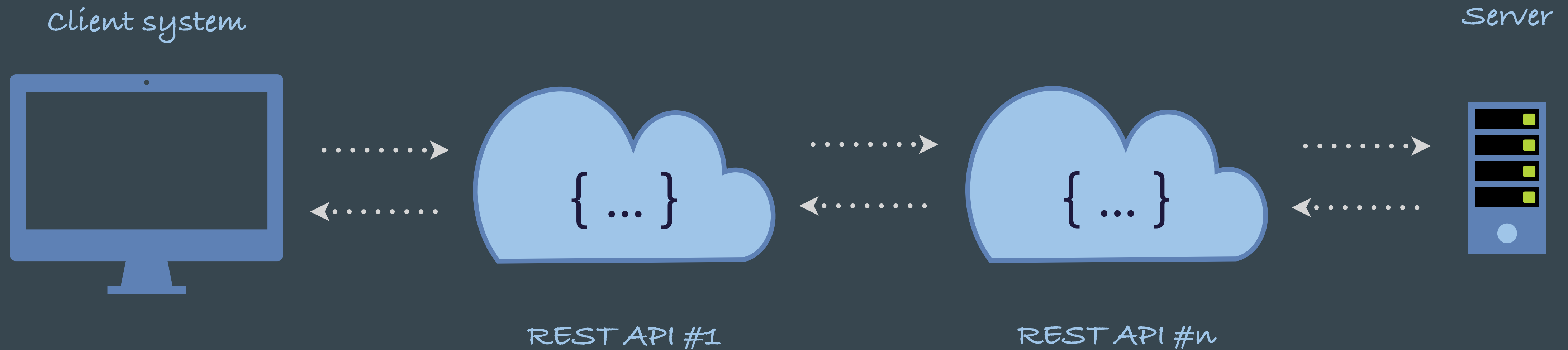
Server



writing data to file on the virtual server

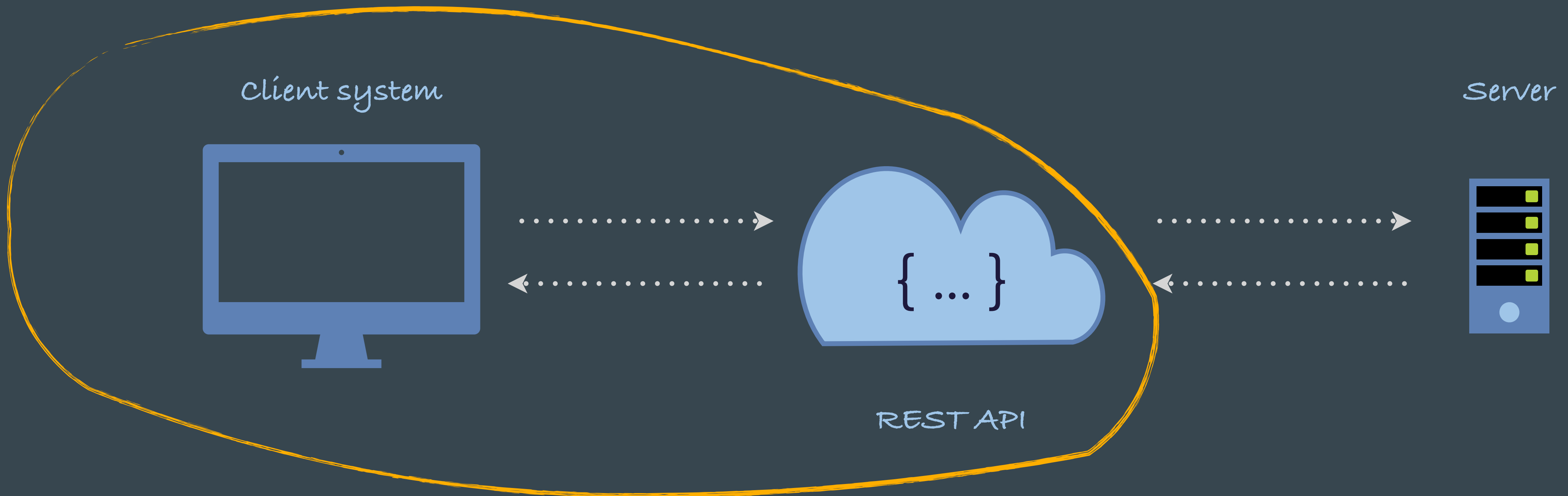
# REST APIs can be chained

There can be more than one API layer, it can be many!



# We need to know also how to build the “other side” - the server

- REST API is a client - server system
- So far we know how to use it as a **client**





# Build a simple Docker server with REST API

- Step 1: Download Docker Desktop
- Step 2: Choose a Docker image (Some Linux version as example)
- Step 3: Create a simple REST API server



# REST API

## Principle

