Big Data, organisation and analysis

Building a REST API service

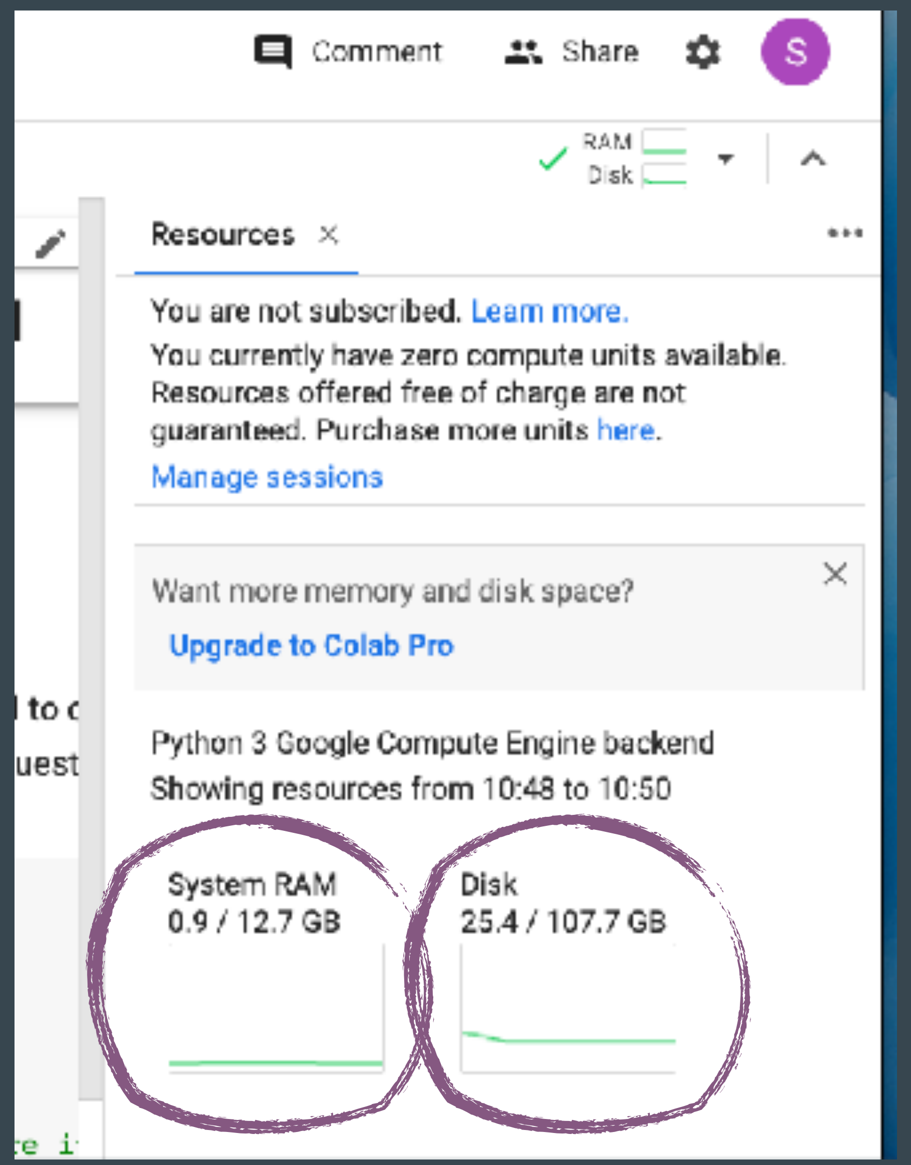
Steffen M. Noe, Spring 2023

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



You are not subscribed. Learn more.
You currently have zero compute units available. Resources offered free of charge are not guaranteed. Purchase more units here.

Manage sessions

Want more memory and disk space?

Upgrade to Colab Pro

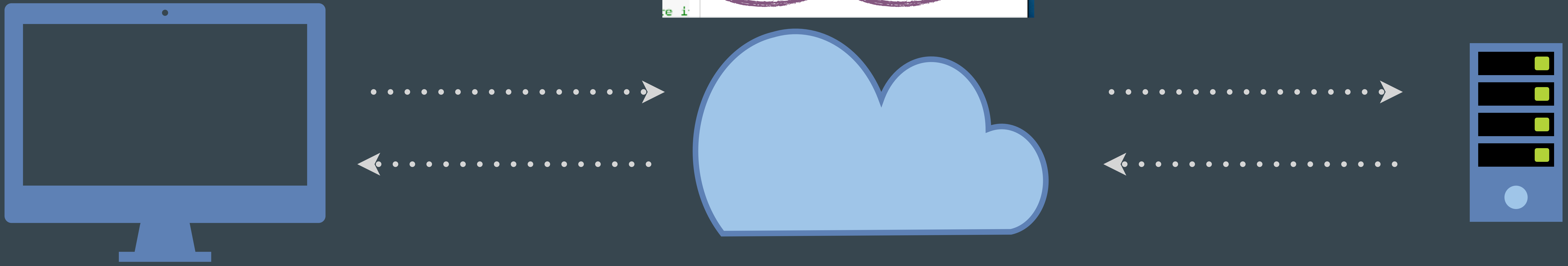
Python 3 Google Compute Engine backend
Showing resources from 10:48 to 10:50

System RAM
0.9 / 12.7 GB

Disk
25.4 / 107.7 GB

our system

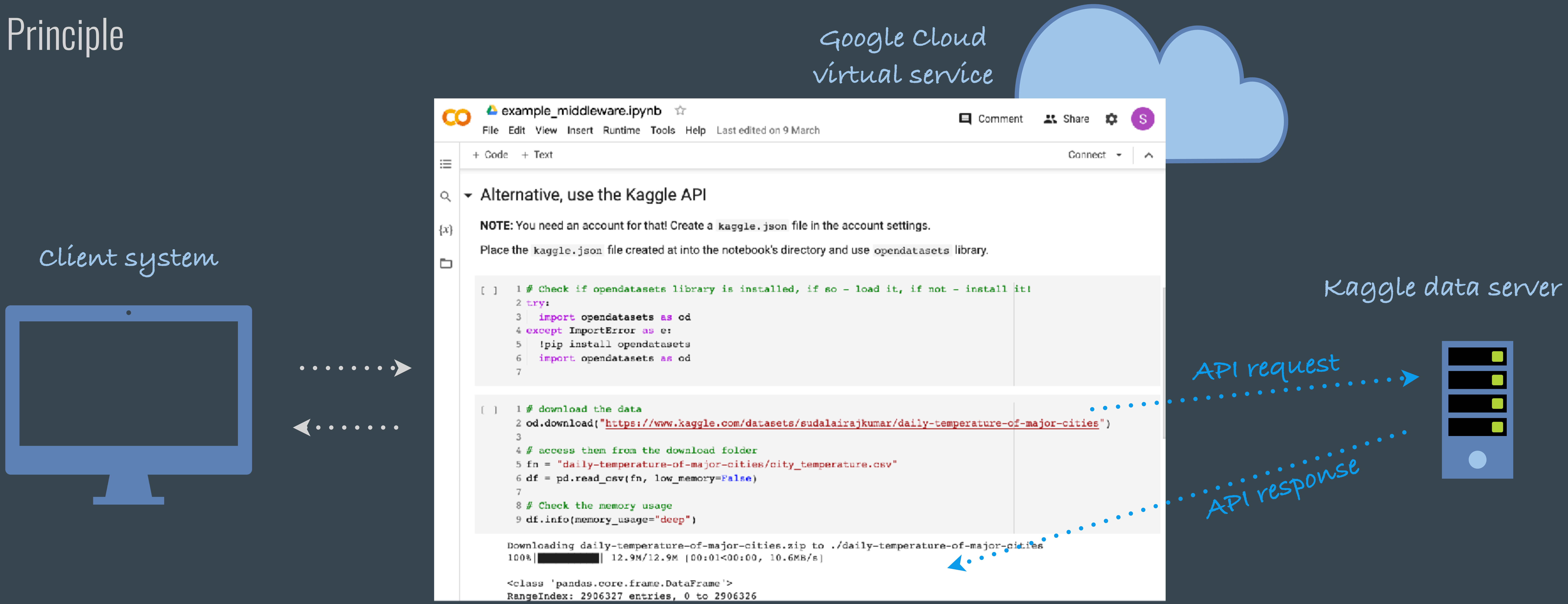
Google Cloud Server



Virtual machine
or
Container

Data access via API

Principle



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.

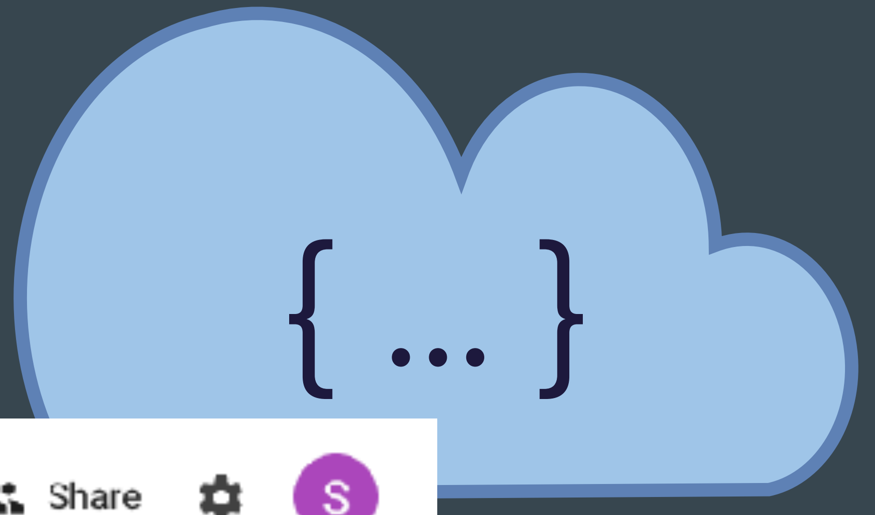
```
[ ]
# Check if opendatasets library is installed, if so - load it, if not - install it!
try:
  import opendatasets as od
except ImportError as e:
  !pip install opendatasets
  import opendatasets as od

```

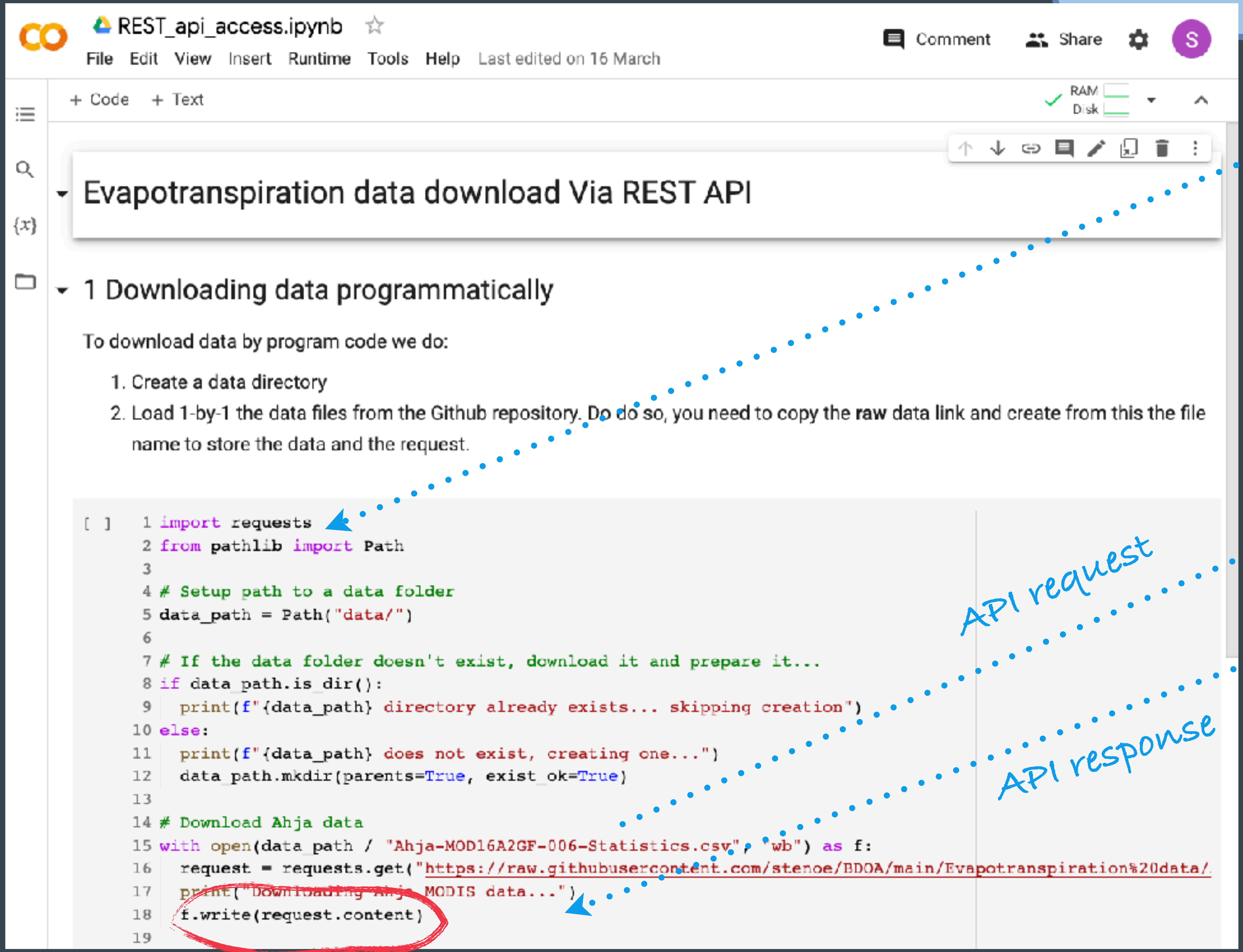
```
[ ]
# download the data
od.download("https://www.kaggle.com/datasets/sudalairajkumar/daily-temperature-of-major-cities")

# access them from the download folder
fn = "daily-temperature-of-major-cities/city_temperature.csv"
df = pd.read_csv(fn, low_memory=False)

# Check the memory usage
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
```

REST API access via requests package

Principle





Evapotranspiration data download Via REST API

1 Downloading data programmatically

To download data by program code we do:

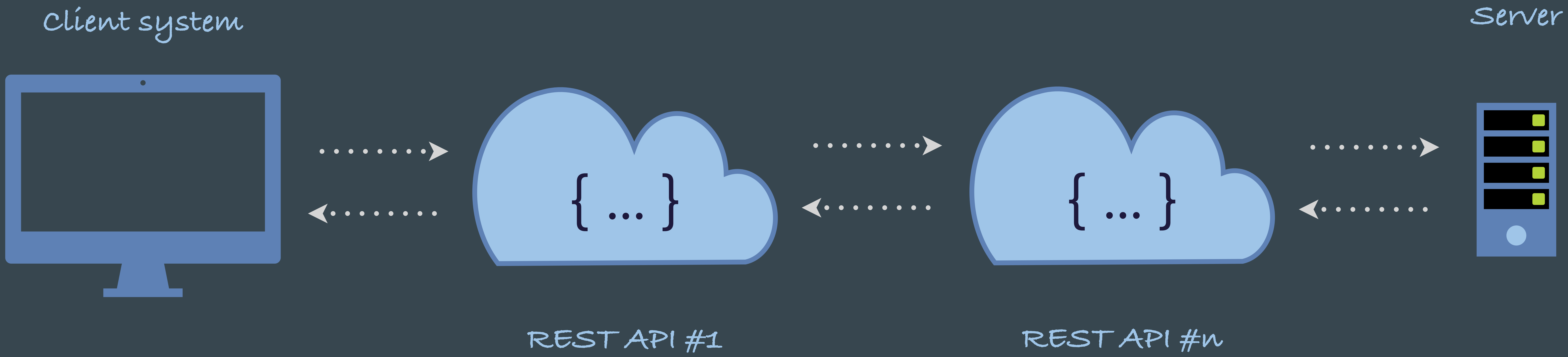
1. Create a data directory
2. Load 1-by-1 the data files from the Github repository. Do do so, you need to copy the raw data link and create from this the file name to store the data and the request.

```
[ ]
import requests
from pathlib import Path

# Setup path to a data folder
data_path = Path("data/")

# If the data folder doesn't exist, download it and prepare it...
if data_path.is_dir():
    print(f"{data_path} directory already exists... skipping creation")
else:
    print(f"{data_path} does not exist, creating one...")
    data_path.mkdir(parents=True, exist_ok=True)

# Download Ahja data
with open(data_path / "Ahja-MOD16A2GF-006-Statistics.csv", "wb") as f:
    request = requests.get("https://raw.githubusercontent.com/stenoe/BDOA/main/Evapotranspiration%20data/
    print("Downloading Ahja MODIS data...")
    f.write(request.content)

```

writing data to file on the virtual server

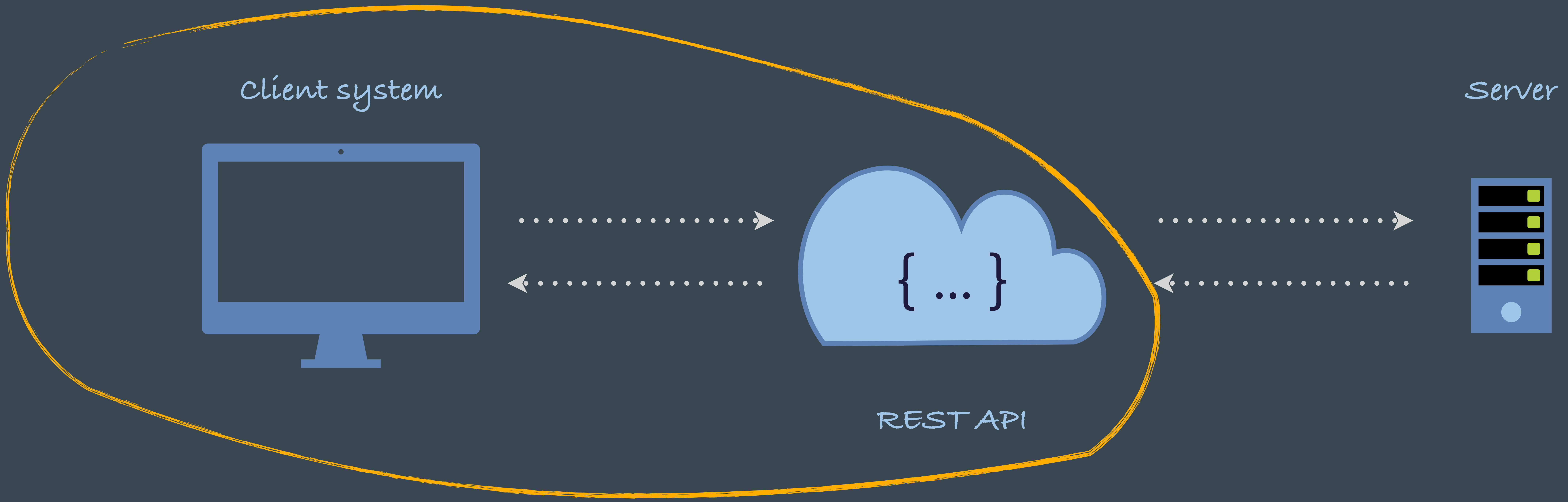
REST APIs can be chained

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

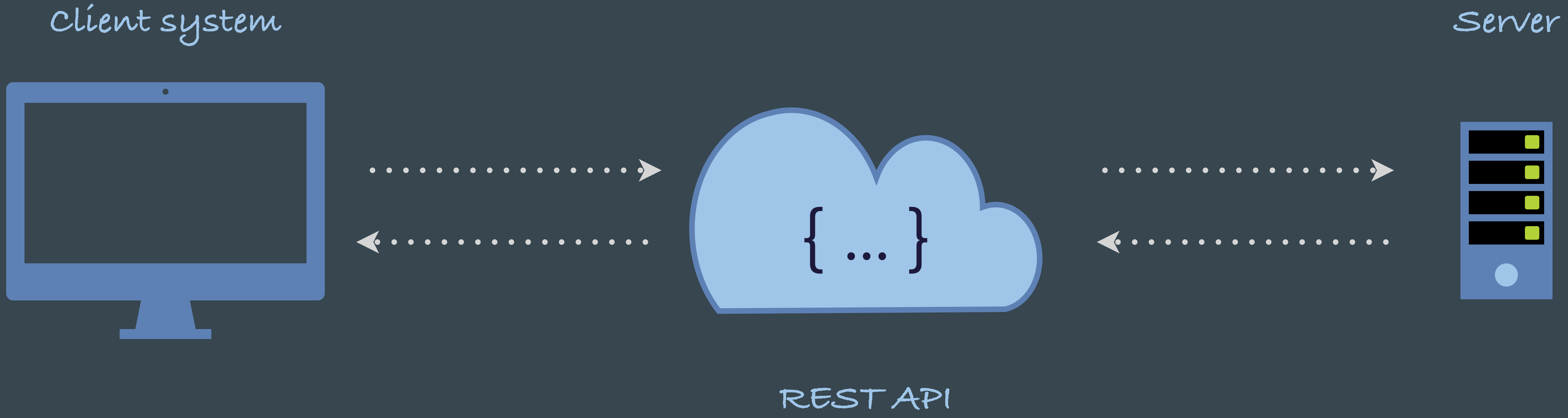


Build a simple Docker server with REST API

- 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


Client system
{ … }
REST API
Server