

# Praktikum CPS

Versuch 4: „Daten Visualisierung“

1. Microsoft Azure: Portal and DevOps	3
2. IaC - Erstellung von Virtual Machine.	4
3. Verbindung von dem lokalen Rechner mit der cloud-gehosteten VM.	5
3.1. SSH Tunnel	5
3.2. Jupyter Notebook	6
4. Daten Visualisierung.	9
4.1. Installation von benötigten Bibliotheken.	9
4.2. Zugriff zu den Wetterdaten.	10
4.3. Visualisierung	11

## 1. Microsoft Azure: Portal and DevOps

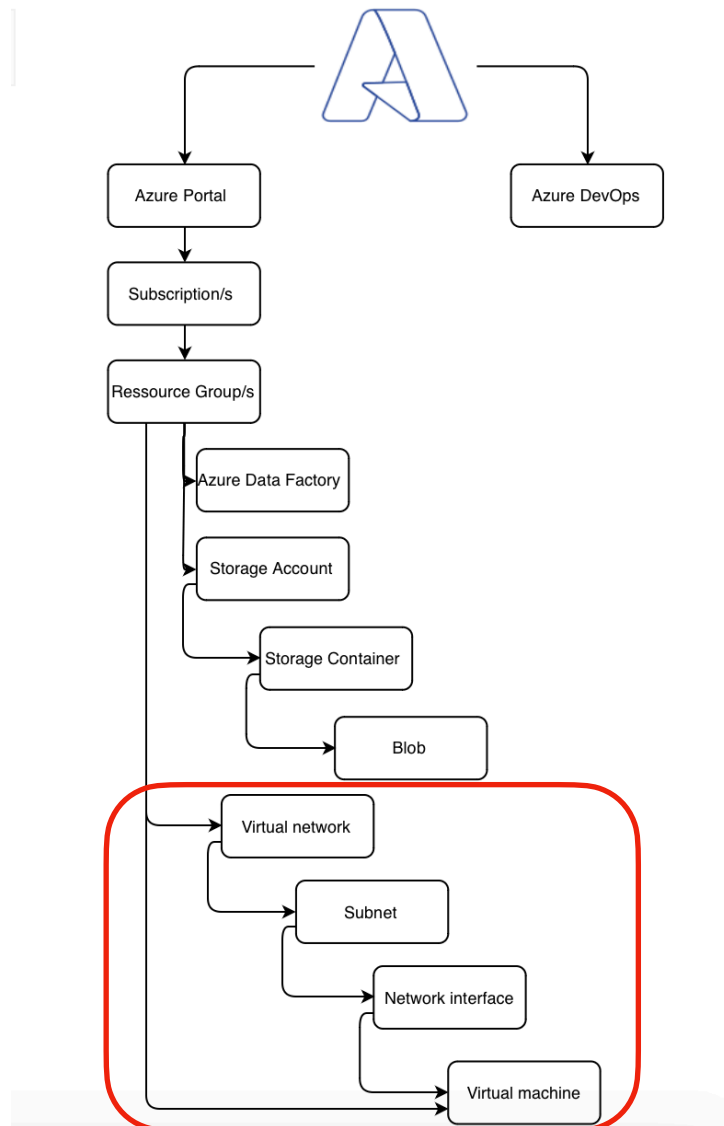


Abbildung 1 - Azure Cloud structure

## 2. IaC - Erstellung von Virtual Machine.

**Bevor Sie beginnen, ein kleiner Hinweis: Nach der Erstellung der VM wird diese sofort hochgefahren. Um unnötige Kosten zu vermeiden, empfiehlt es sich, die VM Manuel zu stoppen, wenn sie nicht benötigt wird.**

1. Öffnen Sie die Datei **main.tf**, die Sie im 1. Versuch erstellt haben.
2. Erstellen Sie mit Hilfe von Terraform die Linux-Virtual Machine. Folgenden Vorlagen können als Referenz verwendet werden.
  - **VM** - [https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/linux\\_virtual\\_machine](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/linux_virtual_machine)
  - **Public IP** - [https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/public\\_ip](https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/public_ip)

Für die Installation von der VM sind folgende Ressourcen notwendig:

- Ressource Group
  - Virtual network
  - Subnet
  - Network interface
  - Public IP Address
  - Virtual machine
3. Für diesen Versuch benötigen Sie eine Linux-basierte VM.
  4. Stellen Sie sicher, dass Sie die Abhängigkeiten zwischen den Ressourcen verstehen.
  5. Nach dem Ausführen von **terraform plan** erhalten Sie wieder eine Meldung darüber, wie viele Ressourcen erstellt werden. Wenn die Anzahl der zu erstellenden Ressourcen Ihren Erwartungen entspricht, führen Sie **terraform apply** aus.
  6. Überprüfen Sie im Azure-Portal, welche Ressourcen jetzt sichtbar sind.

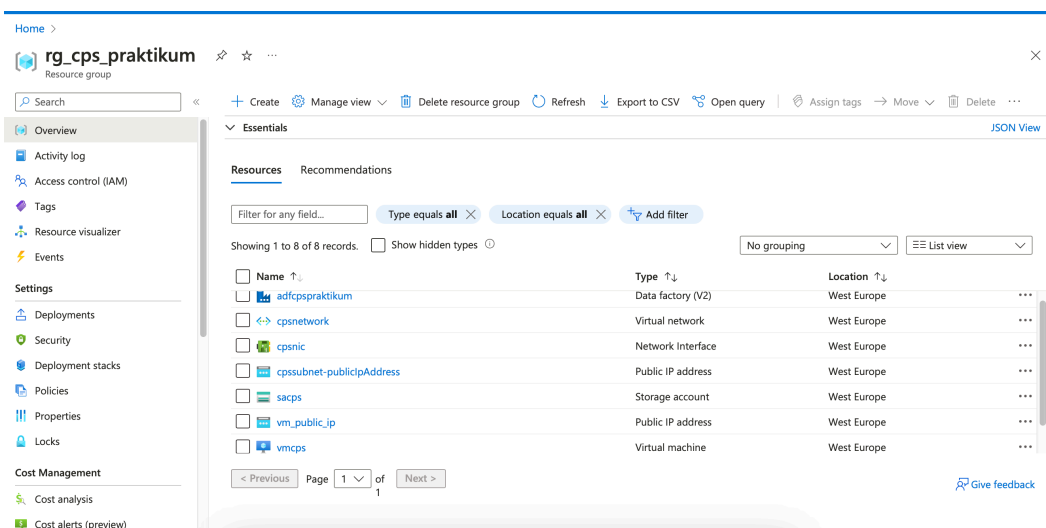


Abbildung 2 - Azure Portal

7. Öffnen Sie die erstellte VM - **vmcps**. Im Overview finden Sie Public IP Address. Merken Sie sich diese.

### 3. Verbindung von dem lokalen Rechner mit der cloud-gehosteten VM.

#### 3.1. SSH Tunnel

Ganz allgemein: Ein **SSH-Tunnel** ist eine verschlüsselte Verbindung zwischen einem lokalen Rechner und einem entfernten Server über das Secure Shell (SSH)-Protokoll. Es wird verwendet, um Daten sicher zwischen einem lokalen und einem entfernten Gerät zu übertragen, indem es eine sichere Kommunikationsleitung durch unsichere Netzwerke, wie das Internet, schafft.

1. Öffnen Sie das Terminal.
2. Führen Sie den folgenden Befehl aus: **ssh -L 8080:localhost:8888 [user name]@[ip address]**

**user name** - Ihr angegebener **admin\_username** für die VM.

**ip address** - Public IP address von VM.

3. Anschließend werden Sie nach einem Passwort gefragt. Geben Sie das Passwort von **admin\_password** an.

Wenn die SSH-Verbindung erfolgreich hergestellt wurde, erhalten Sie folgende Bestätigung:

- Rot - Befehl zur Verbindung
- Blau - Passwortabfrage
- Grün - Username und der Name der VM

```
lime@eduroam-5596-66 ~ % ssh -L 8080:localhost:8888 lime@13.93.36.159
The authenticity of host '13.93.36.159 (13.93.36.159)' can't be established.
ED25519 key fingerprint is SHA256:jytDlFzquq3+m1ITpBAUG18oaRxL7IhoEmcIZJtOL8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '13.93.36.159' (ED25519) to the list of known hosts.
lime@13.93.36.159's password:
Welcome to Ubuntu 22.04.6 LTS (GNU/Linux 5.15.0-1045-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

System information as of Sat Sep 30 10:57:03 UTC 2023

System load:  0.0      Processes:    121
Usage of /:   5.5% of 28.89GB   Users logged in:  0
Memory usage: 8%      IPv4 address for eth0: 10.0.2.5
Swap usage:  0%        IPv4 address for eth0: 10.0.2.4

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

lime@vmcps:~$
```

Abbildung 3 - Bestätigung von SSH Verbindung.

Sie können die IP\_Adresse Ihrer VM mit dem Befehl **curl ifconfig.me** abfragen.

```
lime@vmcps:~$ curl ifconfig.me
13.93.36.159lime@vmcps:~$
```

Abbildung 4- IP Abfrage

### 3.2. Jupyter Notebook

1. Sie werden folgende Tools benötigen:
  - Python
  - Pip
  - Jupyter Notebook
2. Überprüfen Sie zuerst die Version von Python: **python3 --version**
3. Danach überprüfen Sie die Version von pip: **pip --version**. Wenn Sie die Meldung erhalten : *Command 'pip' not found, but can be installed with:*, soll dann die pip installiert werden.
4. Um pip installieren zu können, machen Sie folgendes:
  - Aktualisieren Sie Ihr Linux-System: **'sudo apt update'**
  - Installieren Sie pip: **'sudo apt install python3-pip'**
  - Überprüfung von pip-Version: **pip3 -- version**.
5. Mit den folgenden Befehlen können Sie Jupyter Notebook installieren:
  - **sudo -H pip3 install jupyter**
  - ggf. **pip install --upgrade zipp**
  - Nach der Installation starten Sie das Notebook mit dem Befehl: **'jupyter notebook --no-browser'**
  - Sie sollen die folgende Meldung erhalten:

```
lime@vmcps:~$ jupyter notebook --no-browser
lime@vmcps:~$ ssh -L 8080:localhost:8888 lime@13.93.36.159 - 141x44
lime@vmcps:~$ jupyter notebook --no-browser
[I 2023-09-30 12:53:40.005 ServerApp] Package notebook took 0.0000s to import
[I 2023-09-30 12:53:40.017 ServerApp] Package jupyter_lsp took 0.0115s to import
[W 2023-09-30 12:53:40.017 ServerApp] A `_jupyter_server_extension_points` function was not found in jupyter_lsp. Instead, a `_jupyter_server_extension_paths` function was found and will be used for now. This function name will be deprecated in future releases of Jupyter Server.
[I 2023-09-30 12:53:40.022 ServerApp] Package jupyter_server_terminals took 0.0051s to import
[I 2023-09-30 12:53:40.023 ServerApp] Package jupyterlab took 0.0000s to import
[I 2023-09-30 12:53:40.063 ServerApp] Package notebook_shim took 0.0000s to import
[W 2023-09-30 12:53:40.063 ServerApp] A `_jupyter_server_extension_points` function was not found in notebook_shim. Instead, a `_jupyter_server_extension_paths` function was found and will be used for now. This function name will be deprecated in future releases of Jupyter Server.
[I 2023-09-30 12:53:40.064 ServerApp] jupyter_lsp | extension was successfully linked.
[I 2023-09-30 12:53:40.068 ServerApp] jupyter_server_terminals | extension was successfully linked.
[I 2023-09-30 12:53:40.074 ServerApp] jupyterlab | extension was successfully linked.
[I 2023-09-30 12:53:40.078 ServerApp] notebook | extension was successfully linked.
[I 2023-09-30 12:53:40.300 ServerApp] notebook_shim | extension was successfully linked.
[I 2023-09-30 12:53:40.363 ServerApp] notebook_shim | extension was successfully loaded.
[I 2023-09-30 12:53:40.365 ServerApp] jupyter_lsp | extension was successfully loaded.
[I 2023-09-30 12:53:40.366 ServerApp] jupyter_server_terminals | extension was successfully loaded.
[I 2023-09-30 12:53:40.368 LabApp] JupyterLab extension loaded from /home/lime/.local/lib/python3.8/site-packages/jupyterlab
[I 2023-09-30 12:53:40.368 LabApp] JupyterLab application directory is /home/lime/.local/share/jupyter/lab
[I 2023-09-30 12:53:40.369 LabApp] Extension Manager is 'pypi'.
[I 2023-09-30 12:53:40.372 ServerApp] jupyterlab | extension was successfully loaded.
[I 2023-09-30 12:53:40.375 ServerApp] notebook | extension was successfully loaded.
[I 2023-09-30 12:53:40.376 ServerApp] Serving notebooks from local directory: /home/lime
[I 2023-09-30 12:53:40.376 ServerApp] Jupyter Server 2.7.3 is running at:
[I 2023-09-30 12:53:40.376 ServerApp] http://localhost:8888/tree?token=18de50a6c9056e08f2f6ded48621b696041298503639be9d
[I 2023-09-30 12:53:40.376 ServerApp] http://127.0.0.1:8888/tree?token=18de50a6c9056e08f2f6ded48621b696041298503639be9d
[I 2023-09-30 12:53:40.376 ServerApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 2023-09-30 12:53:40.379 ServerApp]

To access the server, open this file in a browser:
file:///home/lime/.local/share/jupyter/runtime/jpserver-7657-open.html
Or copy and paste one of these URLs:
http://localhost:8888/tree?token=18de50a6c9056e08f2f6ded48621b696041298503639be9d
http://127.0.0.1:8888/tree?token=18de50a6c9056e08f2f6ded48621b696041298503639be9d
[I 2023-09-30 12:53:40.405 ServerApp] Skipped non-installed server(s): bash-language-server, dockerfile-language-server-nodejs, javascript-typescript-langservers, jedi-language-server, julia-language-server, pyright, python-language-server, python-lsp-server, r-languageserver, sql-language-server, texlab, typescript-language-server, unified-language-server, vscode-css-languageserver-bin, vscode-html-languageserver-bin, vscode-json-languageserver-bin, yaml-language-server
```

Abbildung 5 - Jupyter Notebook: Meldung in dem Terminal

Jetzt ganz wichtig: Um nicht durcheinander zu kommen, müssen Sie folgendes verstehen: Sie können nicht einfach die URL nutzen, die Jupyter Ihnen vorschlägt (in Abbildung 5 rot markiert). Das wäre möglich, wenn Sie den Notebook auf Ihrem eigenem Rechner gestartet hätten. Sie haben jedoch den Notebook auf dem Server hochgefahren, der sich auf einer Azure VM befindet.

Um Zugriff auf das Notebook zu erhalten, sollen die SSH-Verbindung berücksichtigen (in Abbildung 5 grün markiert). Verwenden Sie den Port 8080 der SSH-Verbindung, anstelle von 8888 des Jupyter Notebooks. Geben Sie im Browser also: **https://localhost:8080/** oder **http://localhost:8080/** ein. Falls erforderlich, verwenden Sie Ihr Authentifizierungstoken. (Abbildung 5 gelb markiert)

- Jetzt sehen Sie in Ihrem Browser die Hauptseite von Jupyter Notebook.

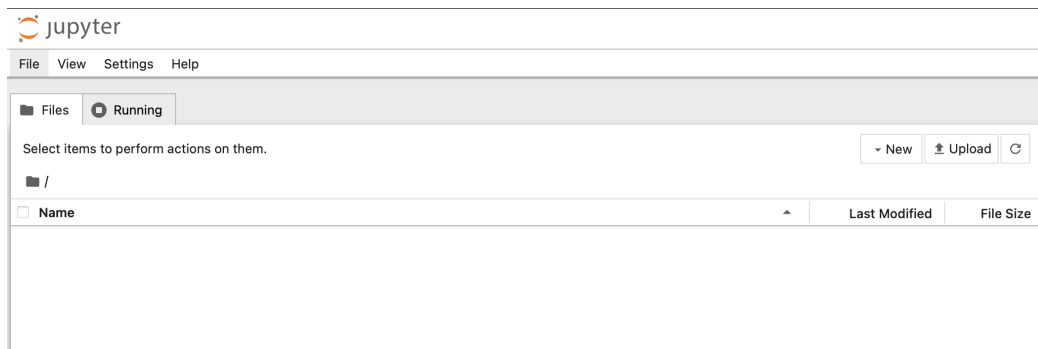


Abbildung 6 - Mainpage von Jupyter Notebook

- Unter ‚Files‘ finden Sie alle Dateien, die für das Notebook zugänglich sind
- Unter ‚Running‘ können Sie sehen, welche Kernels gerade hochgefahren sind.
- Jetzt öffnen Sie ein neues Fenster in VSCode und wählen Sie einen Ordner als Arbeitsverzeichnis aus. (Um Verwirrung zu vermeiden, verwenden Sie nicht dasselbe Fenster und Arbeitsverzeichnis wie für Infrastrukturerstellung).
- Erstellen Sie eine Datei mit dem Namen - **wetterdaten.ipynb**. Das Suffix **.ipynb** steht für Jupyter Notebook.
- In der oberen rechten Ecke finden Sie die Option ‚**Select Kernel**‘. Klicken Sie auf ‚**Select Another Kernel**‘ -> ‚**Existing Jupyter Server**‘ und geben Sie die zuvor verwendete localhost URL. Gegebenenfalls nutzen Sie Ihre Authentifizierungstoken. (Abbildung 5 gelb markiert).
- Führen Sie dann print-Anweisung aus.
- In Ihrem Browser finden Sie jetzt unter „Running“ verbundenen Kernel für den Jupyter Server und Ihr **wetterdaten.ipynb** Notebook.

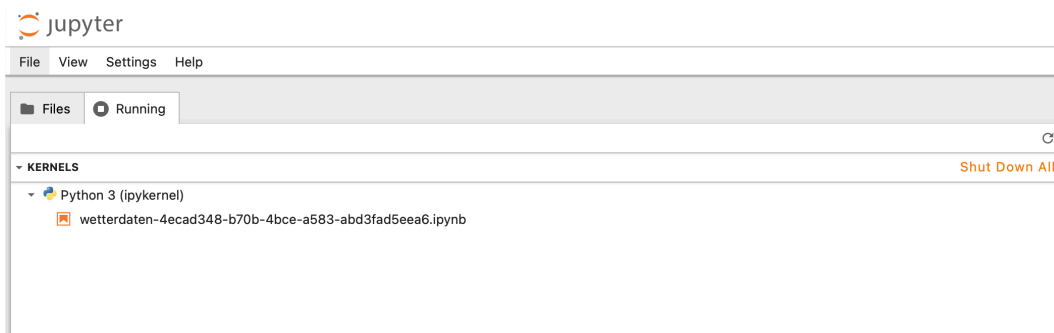


Abbildung 7 - Jupyter Notebook: Kernels



## 4. Daten Visualisierung.

### 4.1. Installation von benötigten Bibliotheken.

Für die Visualisierung können Sie folgende Bibliotheken verwenden:

- **azure-storage-blob** - Zugriff auf Daten, die sich in Azure Blob Storage befinden.
- **pandas** - Zur Datenmanipulation
- **matplotlib** - Zur grafischen Darstellung von Daten

Installieren und importieren Sie diese Bibliotheken wie folgt:

```
1 pip install azure-storage-blob
2 pip install pandas
3 pip install matplotlib
```

Abbildung 8 - Bibliothekeninstallation

```
1 from azure.storage.blob import BlobServiceClient
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import json
✓ 0.0s
```

Abbildung 9 - Import-Anweisungen

## 4.2. Zugriff zu den Wetterdaten.

- Um Zugriff auf die gespeicherten Daten in Azure Blob Storage zu erhalten, verwenden Sie folgenden Befehlen:

```
1 account_name = 'ACCOUNT NAME'
2 account_key = 'ACCOUNT KEY'
3 container_name = 'CONTAINER NAME'
4
5
6 connect_str = 'DefaultEndpointsProtocol=https;AccountName=' + account_name + ';AccountKey=' + account_key + ';EndpointSuffix=core.windows.net'
7 blob_service_client = BlobServiceClient.from_connection_string(connect_str)
8
9 container_client = blob_service_client.get_container_client(container_name)
```

Abbildung 10 - Verbindung mit Storage Container

Before Sie diesen Code ausführen, passen Sie die Value der Variablen an.

Ihre Account key finden Sie unter ‚**Storage Account**‘ -> ‚**Access key**‘

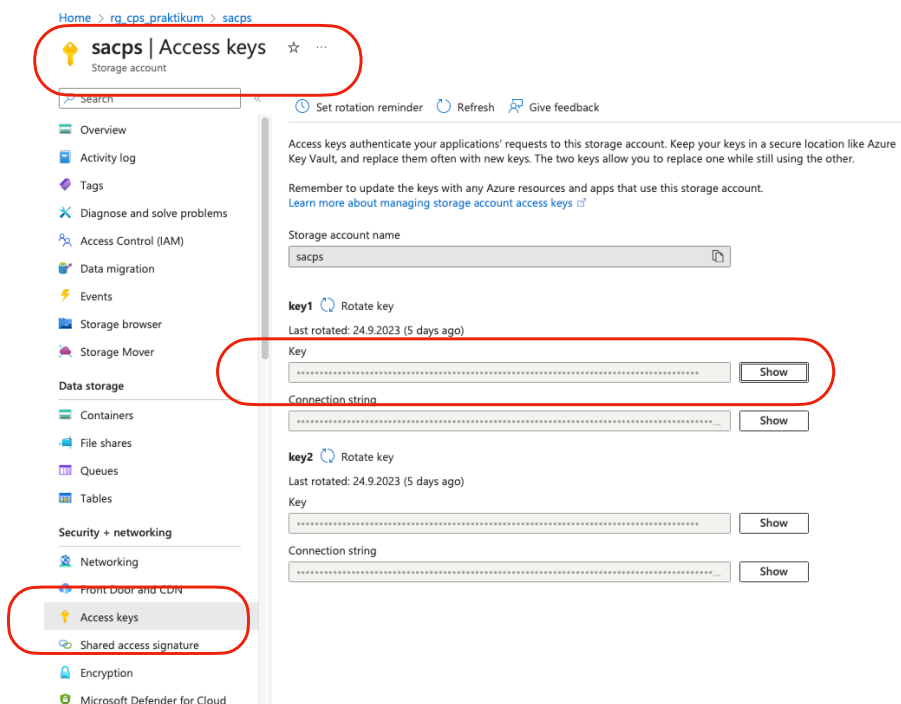


Abbildung 11 - Storage Account Access Key

Im Text Format:

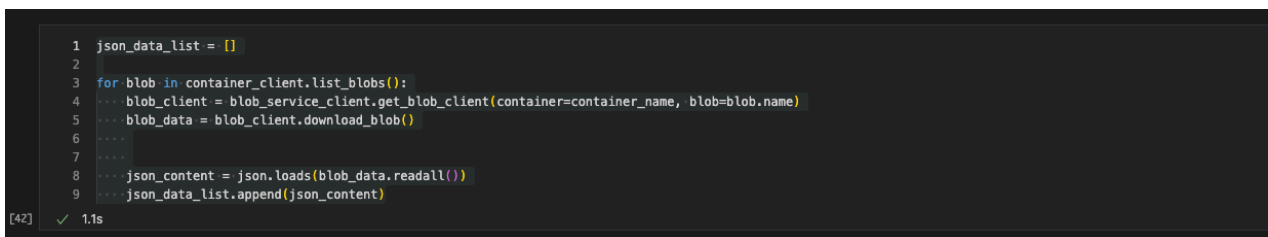
```
account_name = 'ACCOUNT NAME'
account_key = 'ACCOUNT KEY'
container_name = 'CONTAINER NAME'
```

```
connect_str = 'DefaultEndpointsProtocol=https;AccountName=' + account_name +
';AccountKey=' + account_key + ';EndpointSuffix=core.windows.net'
blob_service_client = BlobServiceClient.from_connection_string(connect_str)

container_client = blob_service_client.get_container_client(container_name)
```

### 4.3. Visualisierung

Jetzt können Sie die in Azure gespeicherten Daten lesen.

A screenshot of a terminal window with a dark background. It displays a Python script that iterates through blobs in a container, downloads each blob, and loads its content as JSON. The script includes line numbers 1 through 9. At the bottom left, there is a status bar showing '[42]' and a green checkmark followed by '1.1s'.

```
1 json_data_list = []
2
3 for blob in container_client.list_blobs():
4     blob_client = blob_service_client.get_blob_client(container=container_name, blob=blob.name)
5     blob_data = blob_client.download_blob()
6     ...
7     ...
8     json_content = json.loads(blob_data.readall())
9     json_data_list.append(json_content)
```

Abbildung # - Daten lesen

Auch im Text Format:

```
json_data_list = []

for blob in container_client.list_blobs():
    blob_client = blob_service_client.get_blob_client(container=container_name,
blob=blob.name)
    blob_data = blob_client.download_blob()

    json_content = json.loads(blob_data.readall())
    json_data_list.append(json_content)
```

Mit den folgenden Befehlen können Sie die Daten untersuchen. Wie viele Spalten gibt es? Wie viele Einträge sind vorhanden? Und wie kann der komplette Dataframe angezeigt werden.

```
> ~
1 df = pd.DataFrame(json_data_list)
[]

1 print(len(df.index)) 🔦
[]

1 df
[]

1 print(df.columns)
2 🔦
[]

1 print(df.head())
2 🔦
[]
```

Abbildung # - Data Frame untersuchen

Und schließlich lassen Sie die Daten anzeigen.

```
> ~
1 df['dt'] = pd.to_datetime(df['dt'], unit='s')
2
3 #
4 temperatures_k = [entry['temp'] for entry in df['main']]
5 temperatures_celsius = [temp - 273.15 for temp in temperatures_k]
6
7 #
8 plt.figure(figsize=(10, 6))
9 plt.plot(df['dt'], temperatures_celsius, marker='o', linestyle='-', color='b')
10 plt.title("Temperature Over Time (Celsius)")
11 plt.xlabel("Time")
12 plt.ylabel("Temperature (°C)")
13 plt.xticks(rotation=45)
14 plt.tight_layout()
15 plt.show()
]
```

Abbildung # - Die Daten plotten.

Und im Text Format:

```
df['dt'] = pd.to_datetime(df['dt'], unit='s')

#
temperatures_k = [entry['temp'] for entry in df['main']]
temperatures_celsius = [temp - 273.15 for temp in temperatures_k]

#
plt.figure(figsize=(10, 6))
plt.plot(df['dt'], temperatures_celsius, marker='o', linestyle='-', color='b')
plt.title("Temperature Over Time (Celsius)")
plt.xlabel("Time")
plt.ylabel("Temperature (°C)")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```