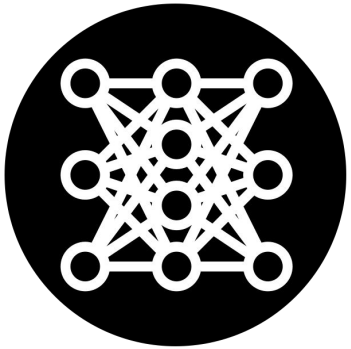
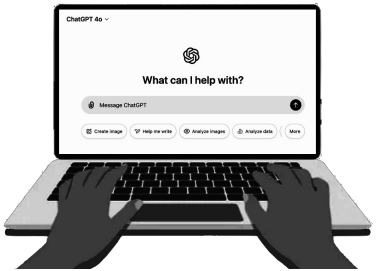


LLM Engineering III

- LLM platforms ➜ Cloud + Local
- Data Persistance
- Databases

LLM Platforms



OpenAI
ChatGPT
Browser Client

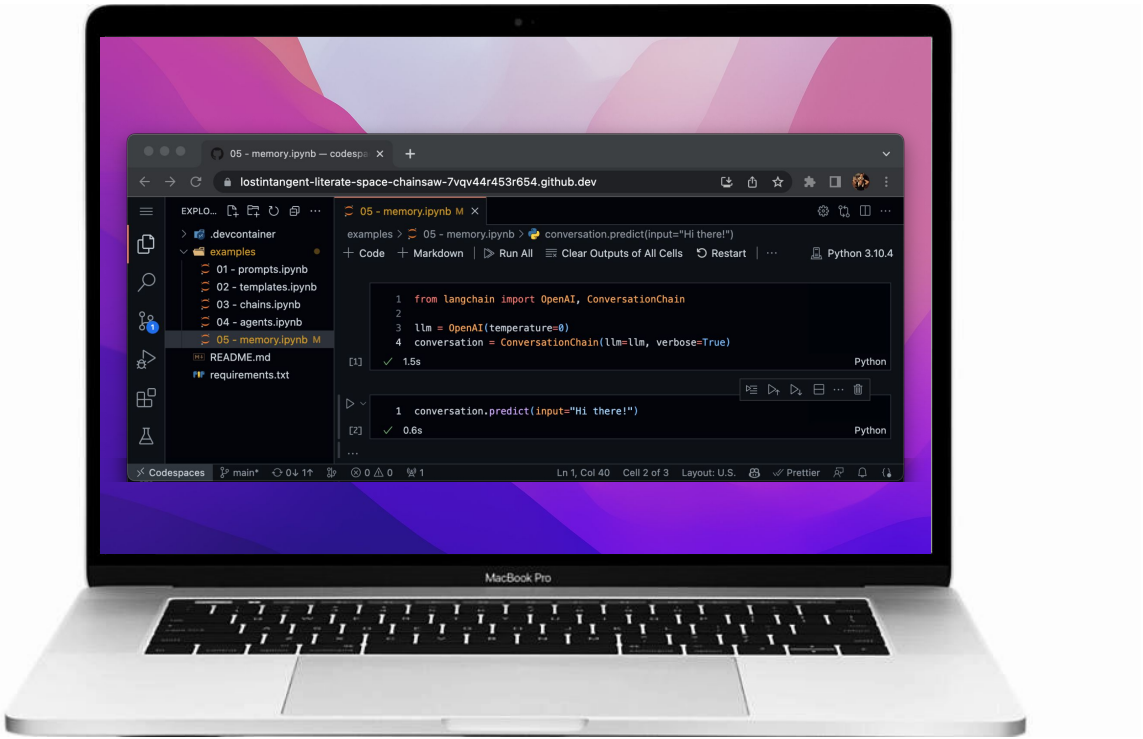
OpenAI
API
Code Call

Ollama
Run Locally
OpenSource

Ollama
CodeSpaces
OpenSource

Ollama in CodeSpaces

Local OS, Browser, Github Cloud, VM, Visual Studio Code



Two ways to run Ollama in codespaces:

1) Install at command line
2) Use a development container

Create a new codespace & install Ollama

1) Install Ollama

Run the following command to download and install Ollama:

```
curl -fsSL https://ollama.com/install.sh | sh
```

2) Verify installation

Type ollama in the terminal to verify the installation:

```
ollama
```

Start Ollama and run a model

1) Start Ollama

Run the following command to start Ollama:

```
ollama serve
```

2) Open a new shell

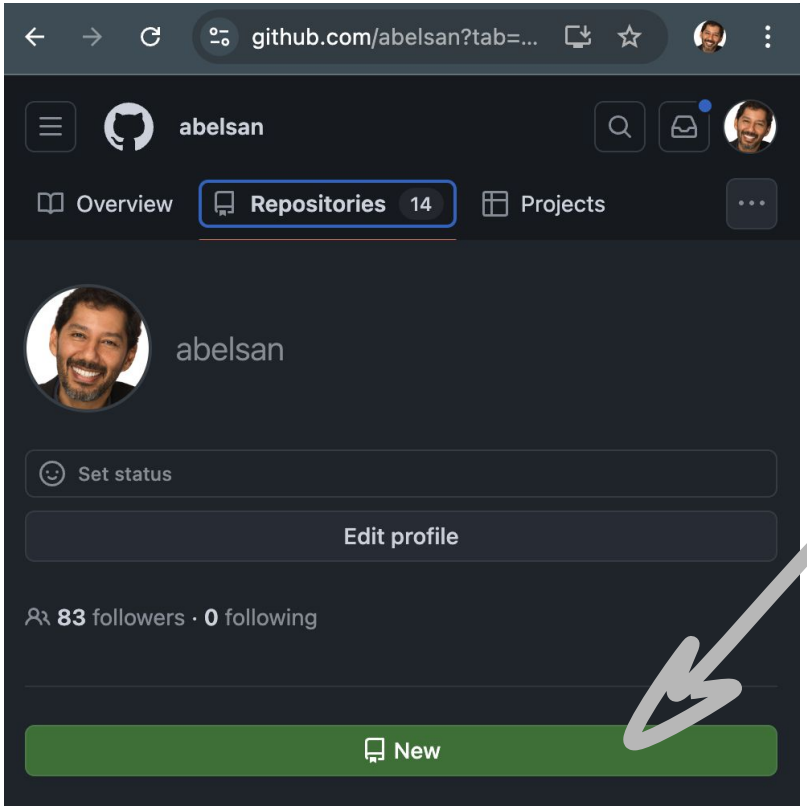
Open a new shell, separate from the one used to launch Ollama:

```
ollama run llama3.2:1b
```

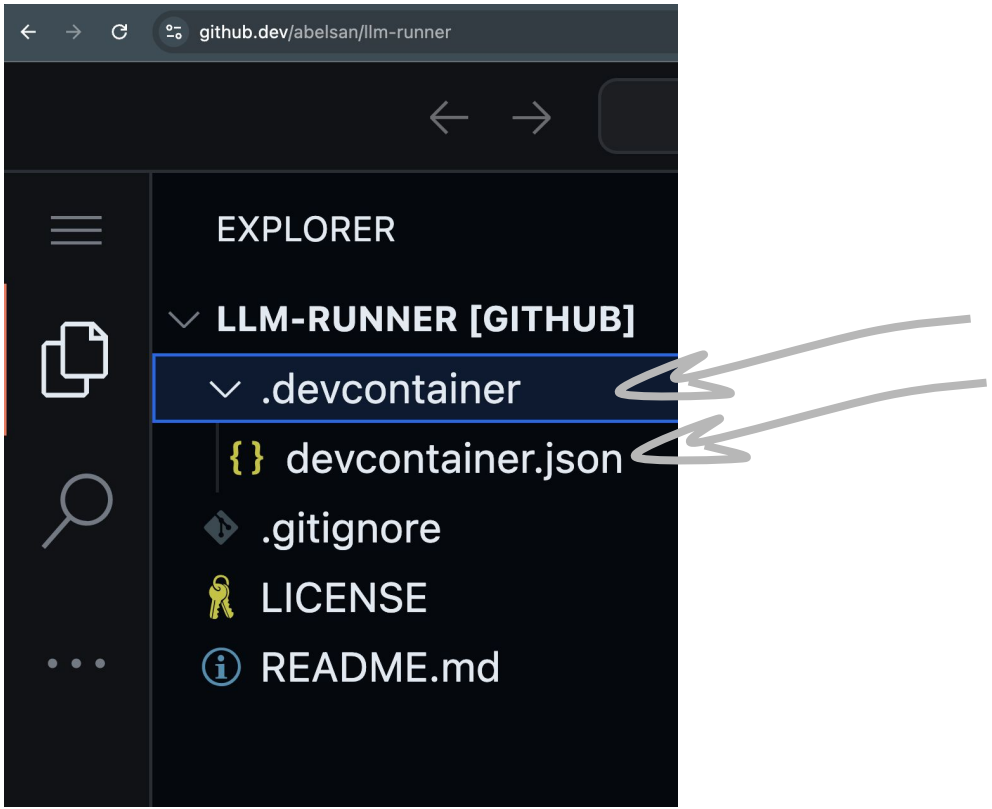
Two ways to run Ollama in codespaces:

1) Install at command line
2) Use a development container

Create a new repository



Add .devcontainer directory and devcontainer.json file



Create development container

The path for development containers within a Github repository is:
`.devcontainer/devcontainer.json`

Community-created Ollama feature in devcontainer.json

```
"name": "Ollama Python Playground",
"image": "mcr.microsoft.com/devcontainers/python:3.12-bullseye",
"features": {
    "ghcr.io/prulloac/devcontainer-features/ollama:1": {}
}
```

For more information on devcontainers:
`https://aka.ms/devcontainer.json`

devcontainer.json

Full address:
https://github.com/abelsan/llm-runner/blob/main/.devcontainer/devcontainer.json

Bit.ly minimized address:

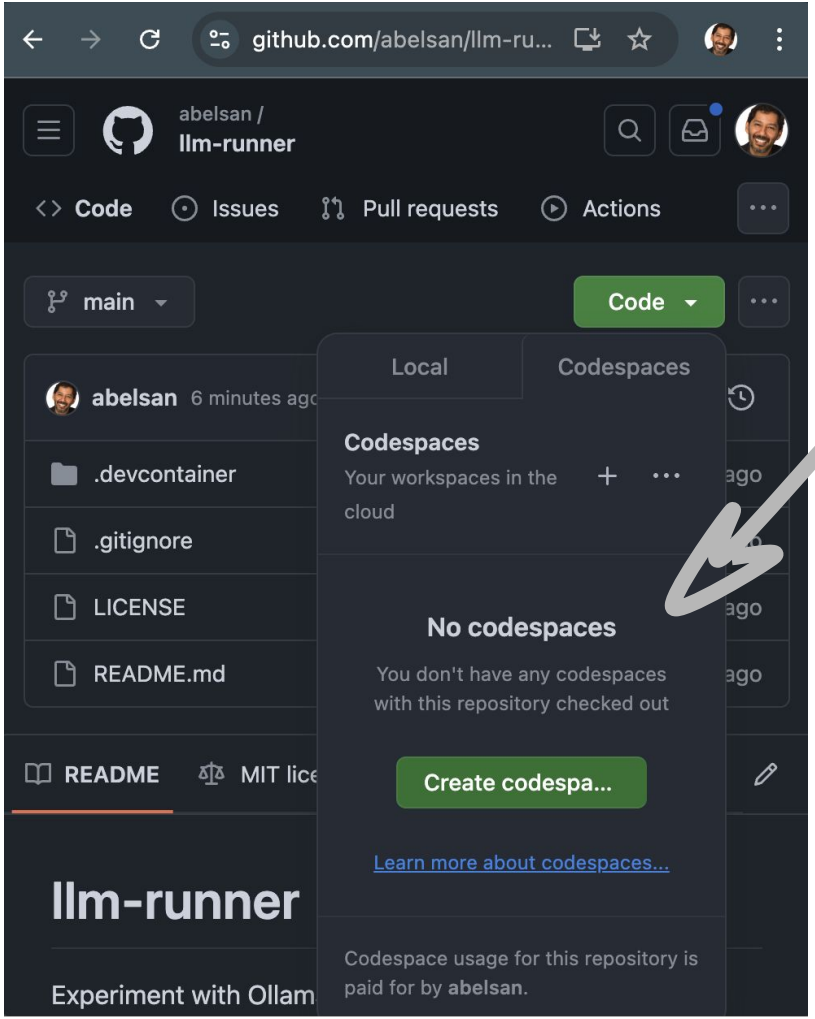
https://bit.ly/4dVxA3g

Sample devcontainer

devcontainer.json

```
{
    "name": "Ollama Python Playground",
    "image": "mcr.microsoft.com/devcontainers/python:3.12-bullseye",
    "features": {
        "ghcr.io/prulloac/devcontainer-features/ollama:1": {}
    },
    "customizations": {
        "vscode": {
            "settings": {
                "python.defaultInterpreterPath": "/usr/local/bin/python",
                "files.exclude": {
                    "__pycache__": true
                }
            },
            "extensions": [
                "ms-python.python"
            ]
        }
    },
    "hostRequirements": {
        "memory": "16gb"
    },
    "remoteUser": "vscode"
}
```

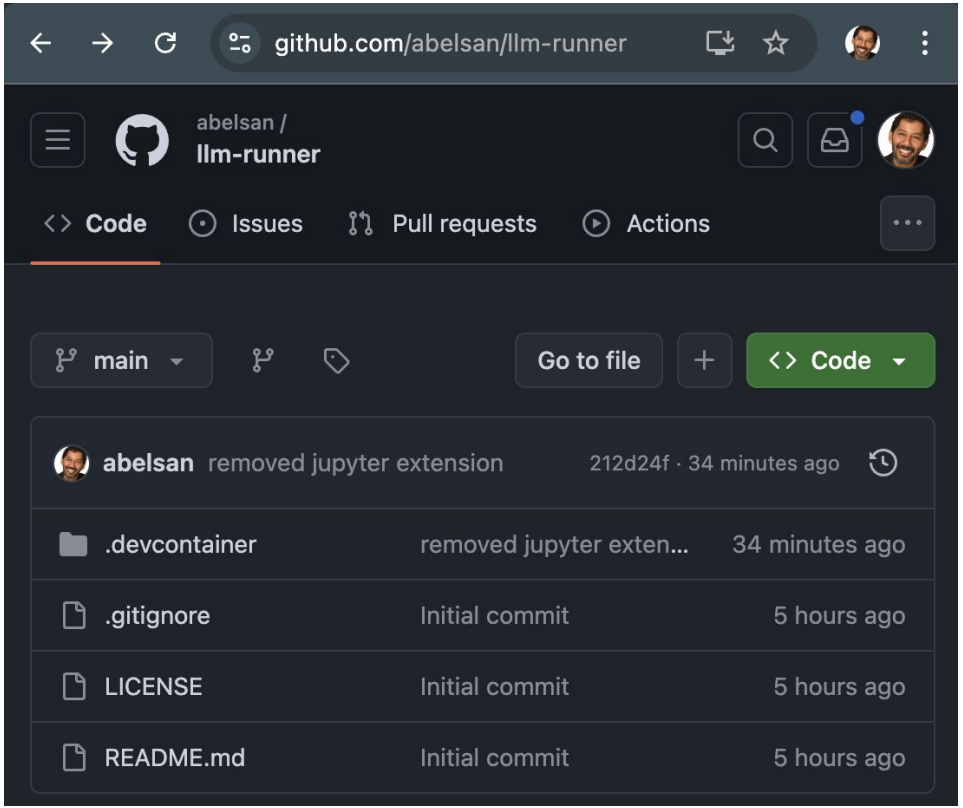
Repository address:
https://github.com/abelsan/llm-runner



Create new codespace with Ollama already in it.

You can also fork from my repository

`https://github.com/abelsan/llm-runner`



Run a model using Ollama

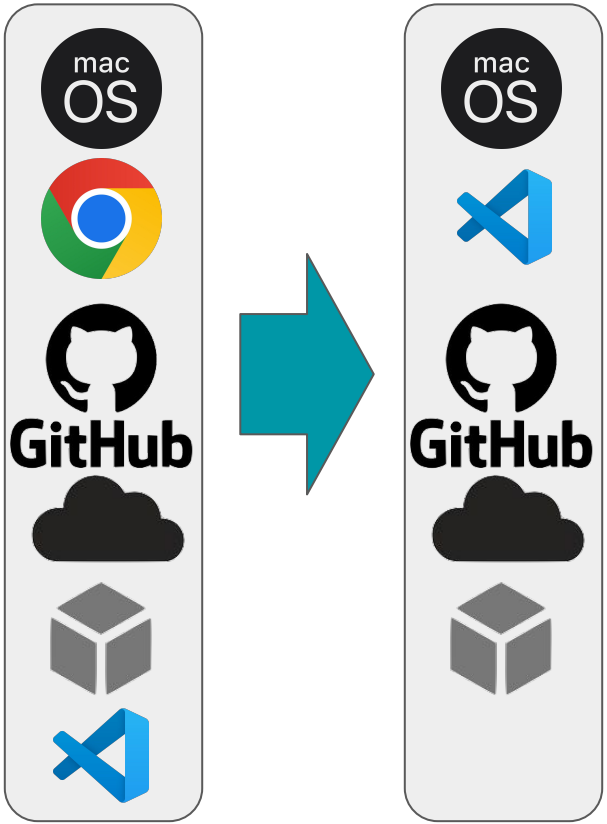
At the command line prompt:

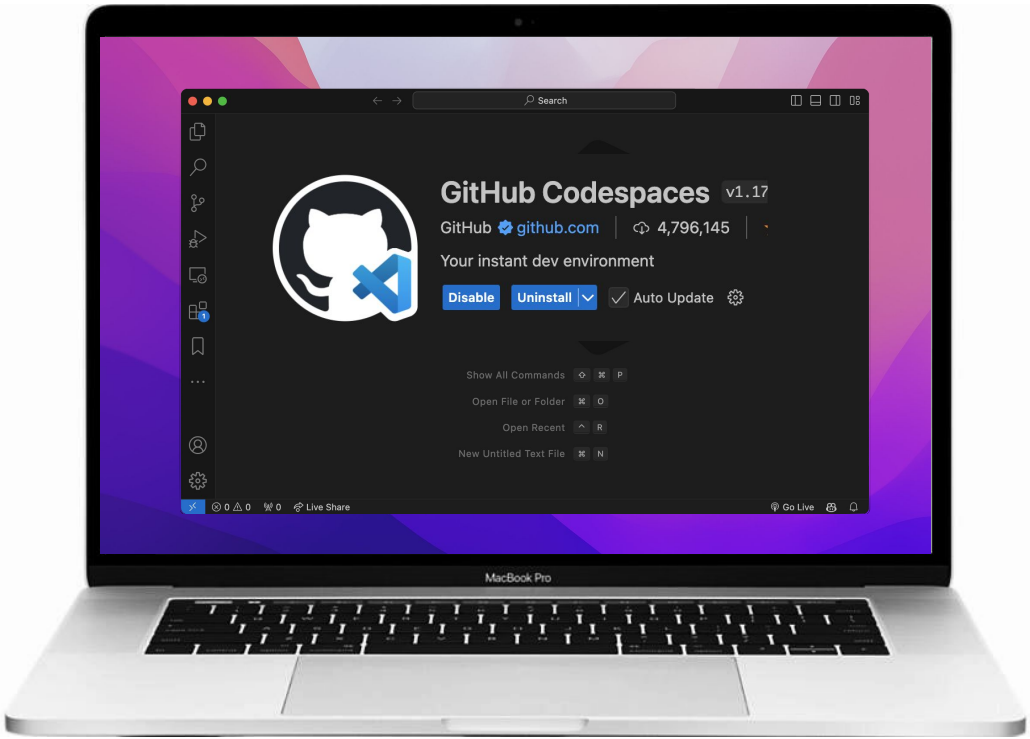
```
ollama run llama3.2:1b
```

Visualization of YouTube comments

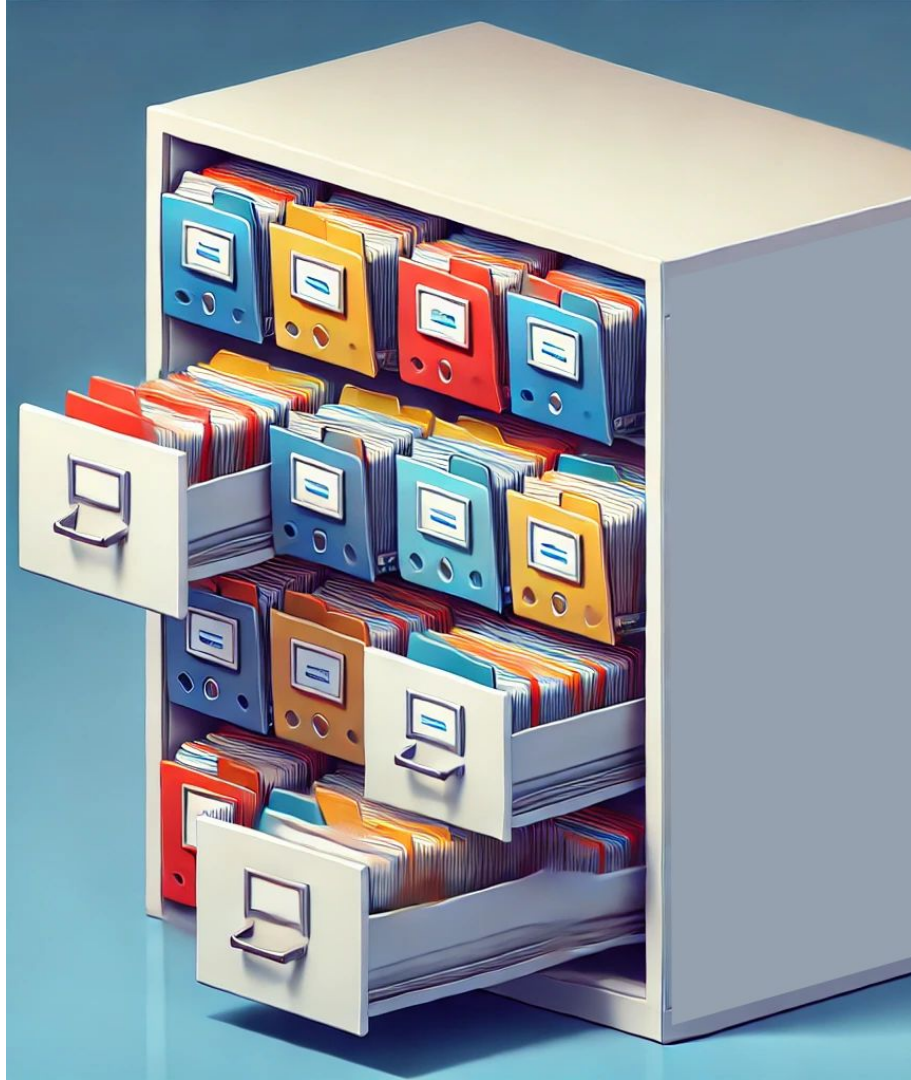
```
# Step 4: Retrieve embeddings for clustering and visualization
embedded_texts = [embeddings.embed_query(text) for text in comments]

# Step 5: Find clusters using K-means and visualize with t-SNE
n_clusters = 4
kmeans = KMeans(n_clusters=n_clusters, init="k-means++", random_state=42)
kmeans.fit(embedded_texts)
labels = kmeans.labels_

# Create a DataFrame for visualization
df = pd.DataFrame({'Text': comments, 'Embedding': embedded_texts, 'Cluster': labels})

# Visualize clusters using t-SNE
tsne = TSNE(n_components=2, perplexity=30, random_state=42, init="random",
learning_rate=200)
vis_dims2 = tsne.fit_transform(embedded_texts)

x = [x for x, y in vis_dims2]
y = [y for x, y in vis_dims2]

for category, color in enumerate(["purple", "green", "red", "blue"]):
    xs = np.array(x)[df.Cluster == category]
    ys = np.array(y)[df.Cluster == category]
    plt.scatter(xs, ys, color=color, alpha=0.3)

    avg_x = xs.mean()
    avg_y = ys.mean()

    plt.scatter(avg_x, avg_y, marker="x", color=color, s=100)

plt.title("Clusters identified visualized in language 2D using t-SNE")
plt.savefig('plot.png')
```

Local OS, Visual Studio Code, Github Cloud, VM



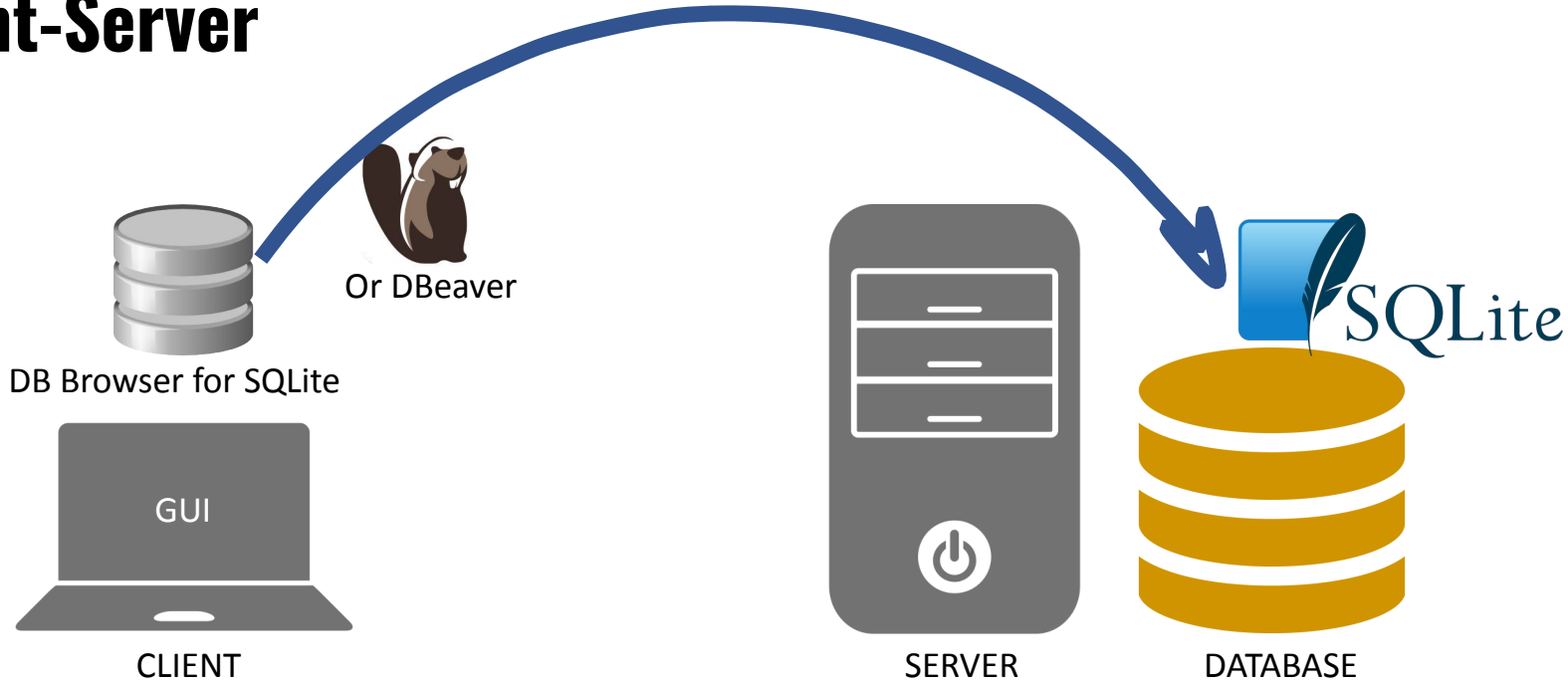


Data persistence

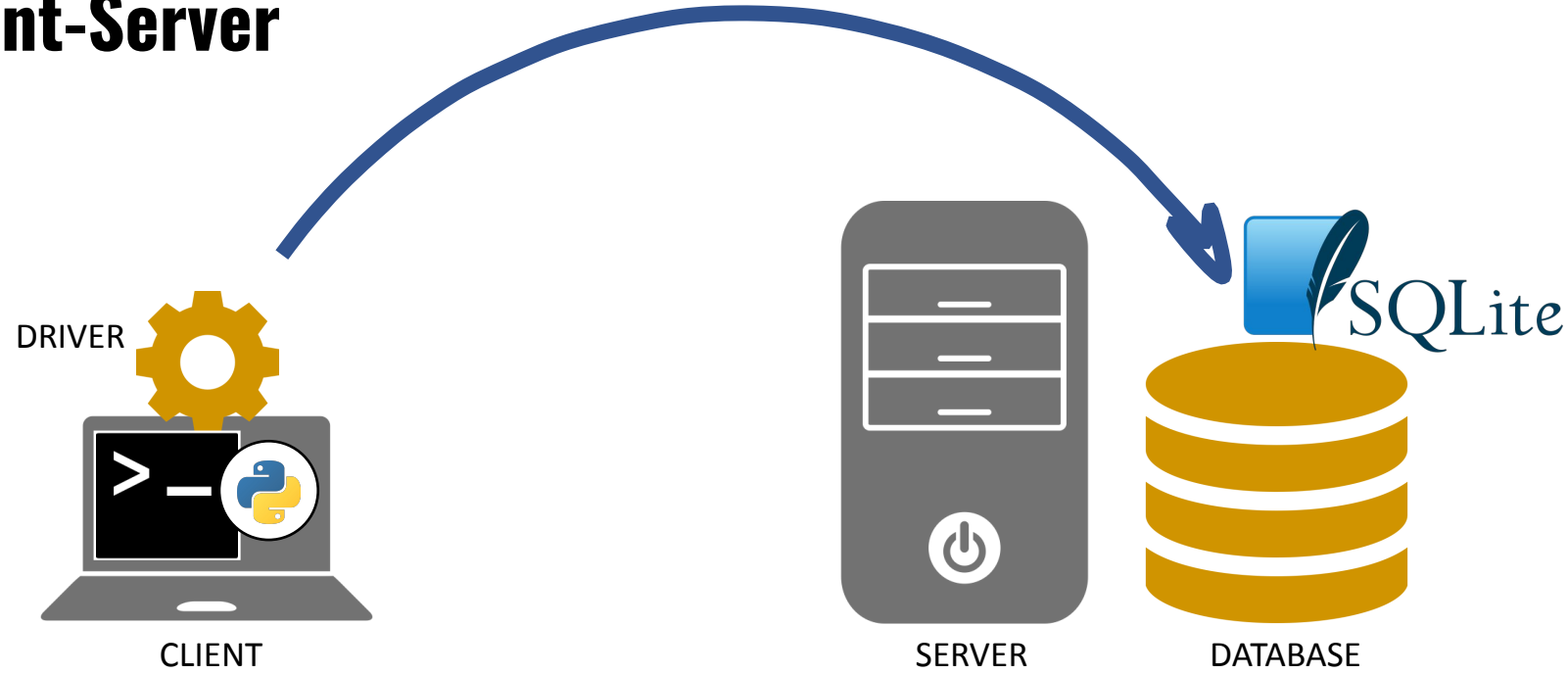


Python + DB

Client-Server



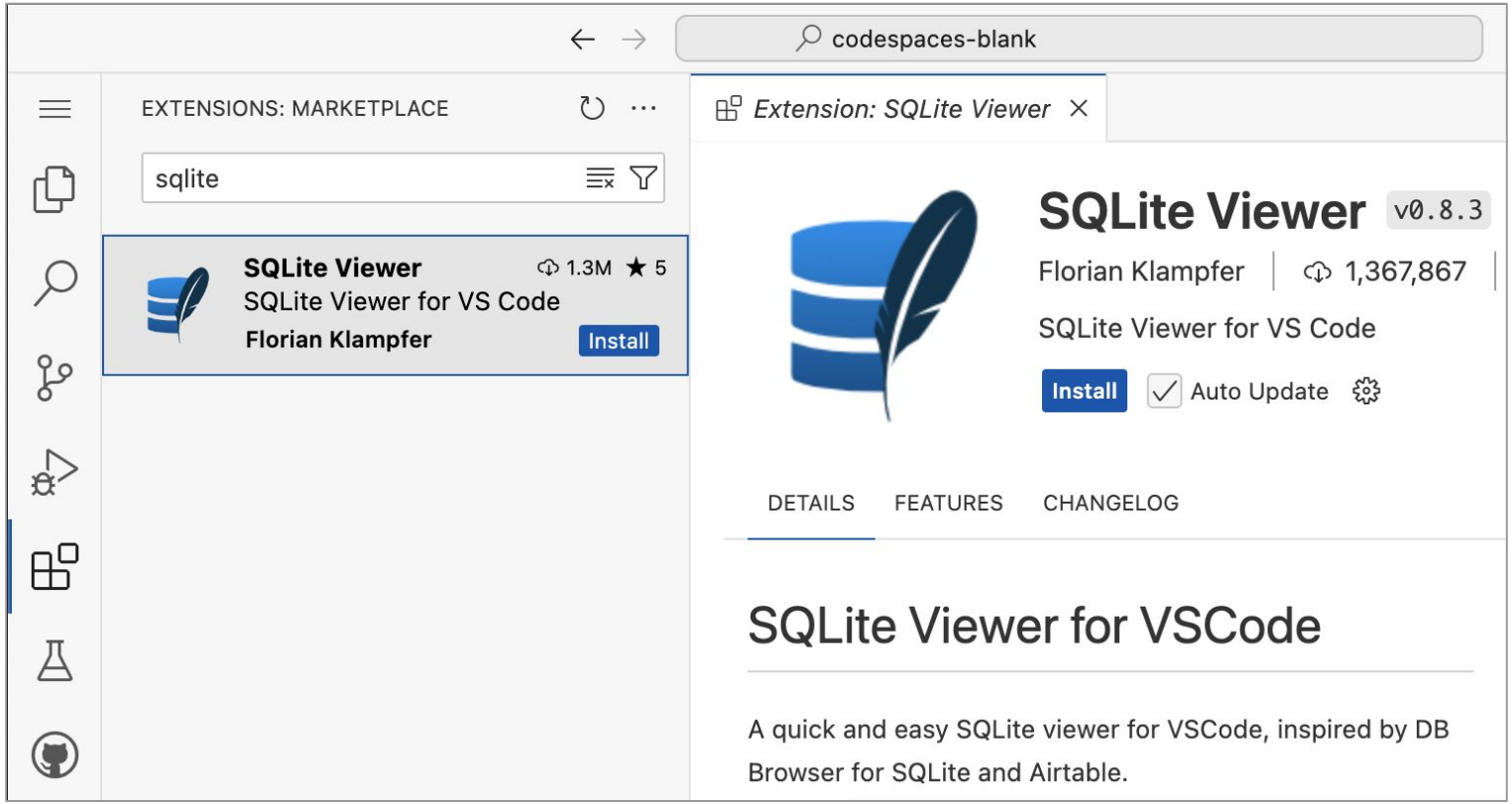
Client-Server



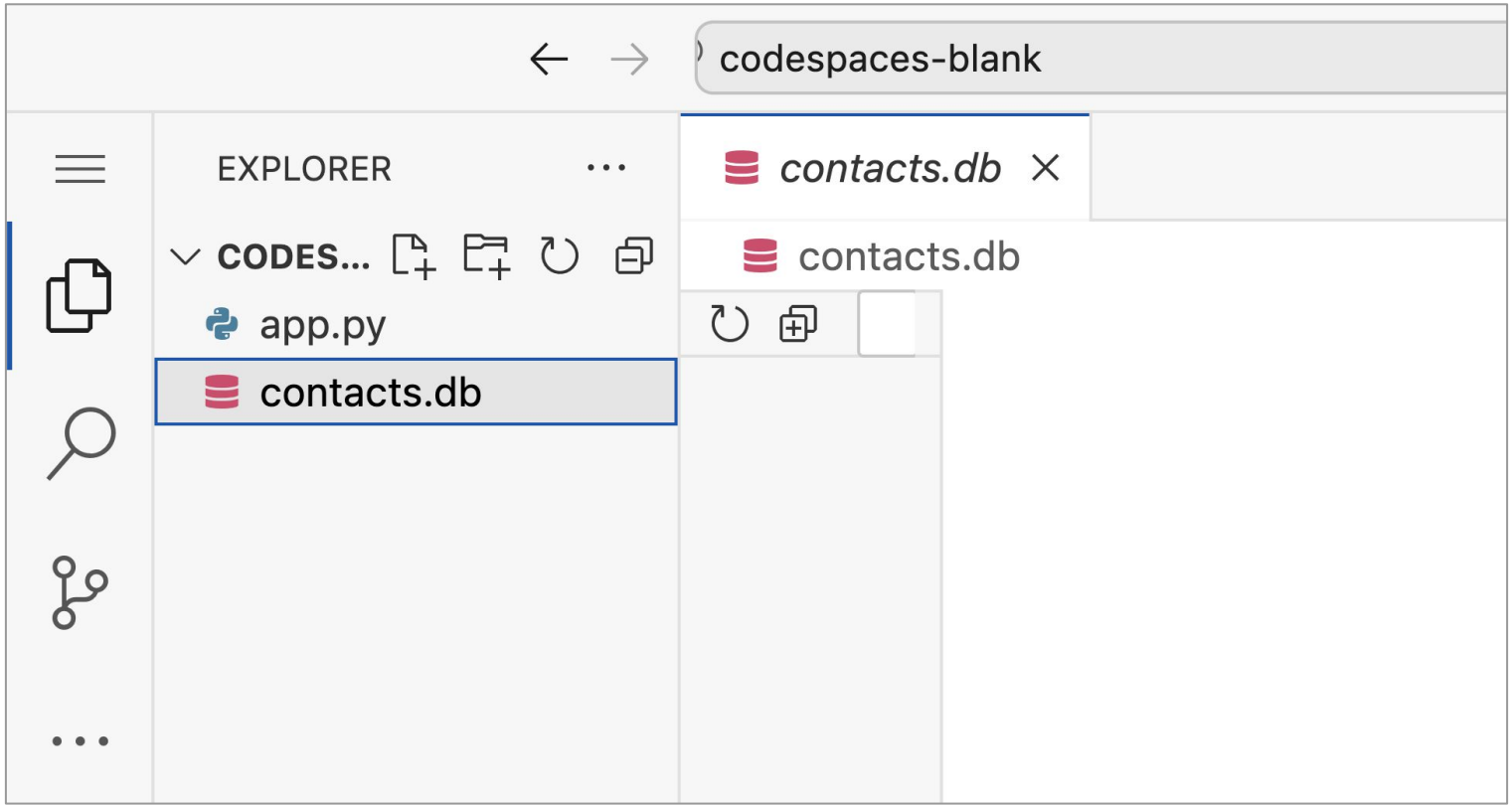
Create Database

```
# ------------------------------------------------------
#  Python SQLite "sqlite3" documentation:
#     https://docs.python.org/3/library/sqlite3.html
# ------------------------------------------------------

import sqlite3

# connecting to db, creates db
connection = sqlite3.connect("contacts.db")
```

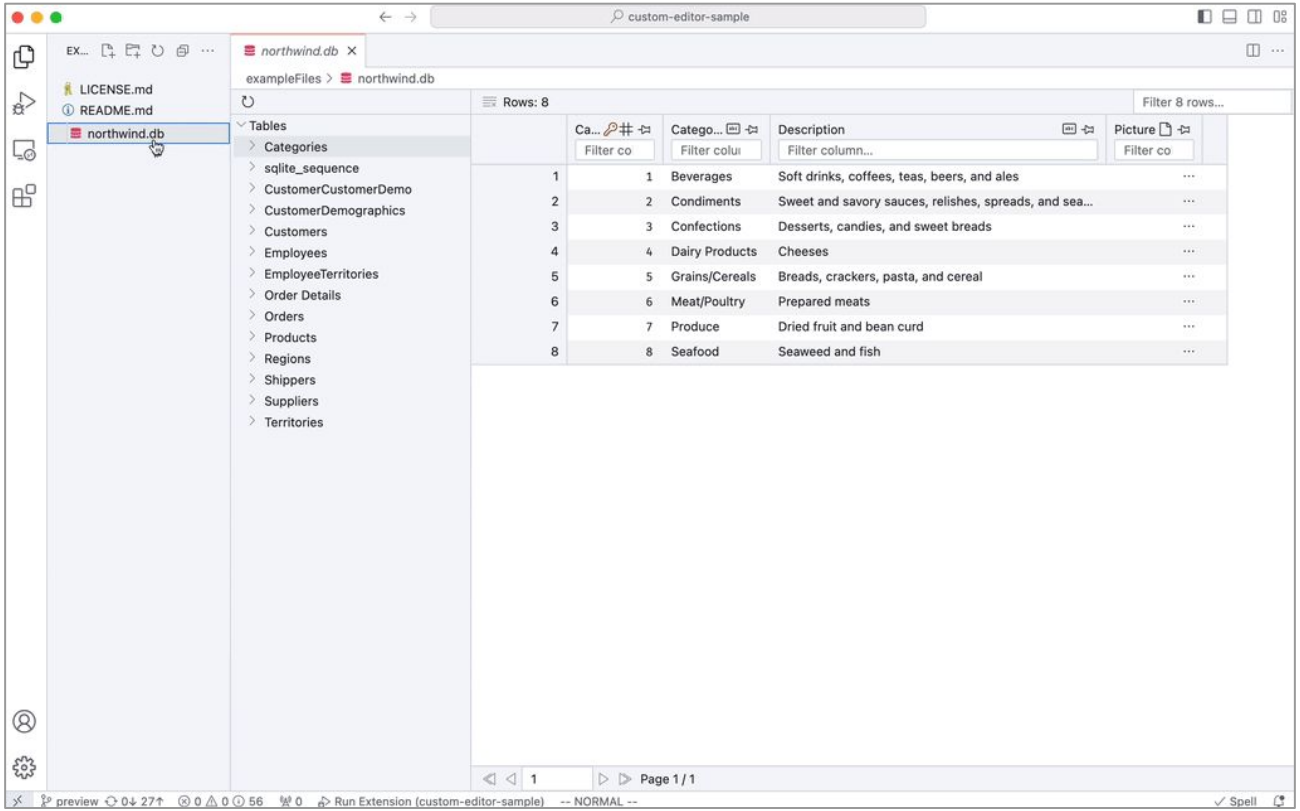
SQLite Viewer for VSCode



SQLite Viewer for VSCode



SQLite Viewer for VSCode



SQL Query Calling Pattern

```
# connect to db
connection = sqlite3.connect("contacts.db")

#  create db cursor
cursor = connection.cursor()

# create table
rows = cursor.execute("""

     Your SQL goes here

""")
```

Exercise: Connect to DB and create table

```
# connect to db
connection = sqlite3.connect("contacts.db")

#  create db cursor
cursor = connection.cursor()

# create table
rows = cursor.execute("""
    -- ------------------------------------
    -- create table
    -- call table contacts
    -- add columns firstname, lastname, email
    -- ------------------------------------
""")
```

Replace with your code

Connect to DB and create table

```
# connect to db
connection = sqlite3.connect("contacts.db")

#  create db cursor
cursor = connection.cursor()

# create table
rows = cursor.execute("""
    CREATE TABLE contacts(
        firstname,
        lastname,
        email)
""")
```

Exercise: Populate table

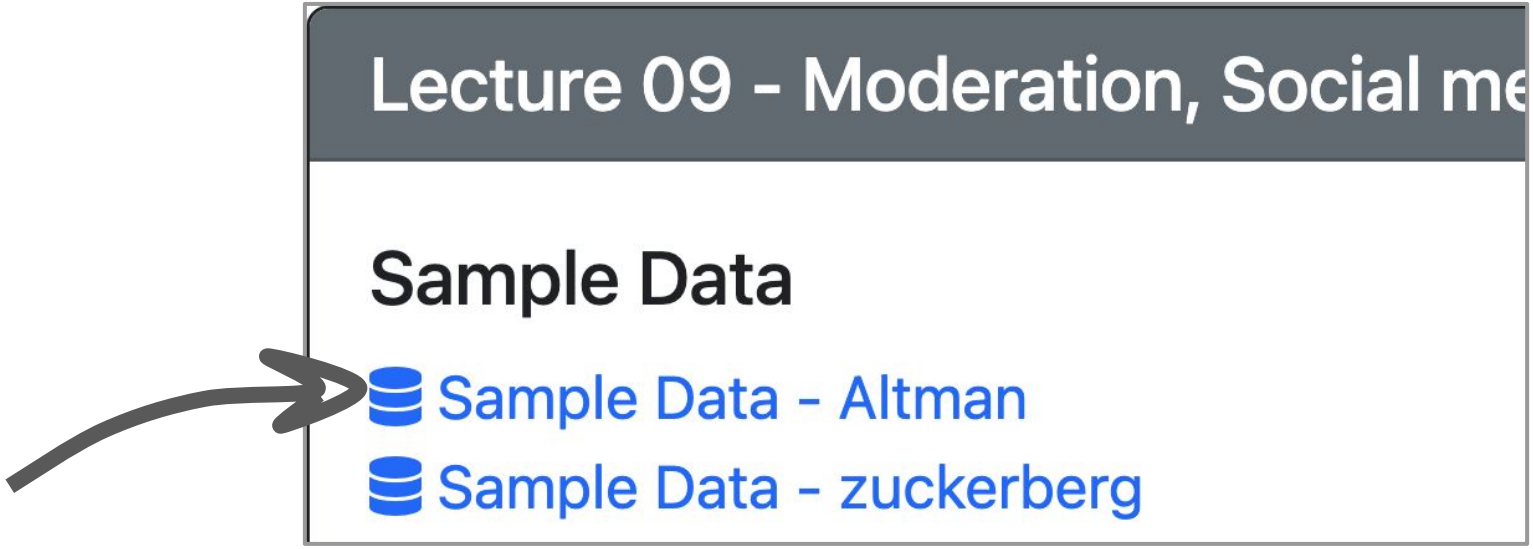
```
# write to table
cursor.execute("""
    -- ----------------------------
    --    Your Code:
    --      Add 3 three contacts
    --      to the contacts table
    -- ----------------------------
""")

# commit changes
connection.commit()
```

Replace with your code

Populate table

```
# write to table
cursor.execute("""
    INSERT INTO contacts VALUES
        ('peter', 'parker', 'peter@mit.edu'),
        ('clark', 'kent', 'clark@mit.edu'),
        ('bruce', 'wayne', 'bruce@mit.edu')
""")

# commit changes
connection.commit()
```

Download Altman's data - 11,491 comments



Read JSON data - count comments

```
# ----------------------------------------
#   Verify that you can read the json file
# ----------------------------------------
import json

file = open('data.json')
data = json.load(file)

for index, item in enumerate(data):
   print(item['cid'],index)
   # print(item['text'],index)
```

Active Learning - load data into “comments.db”

- Write code to create “comments” database
- Write code to create “comments” table
- Write code to load comments into database
- Load all 11,491 comments into database

Comments table

```
# create table
rows = cursor.execute('''
    CREATE TABLE comments(
        id,
        cid,
        text,
        time,
        author,
        channel,
        votes,
        photo,
        heart,
        reply,
        time_parsed)
''')
```

Write comments to database

```
# write to database
def write_to_db(index, item):
    print(index)
    cursor.execute("INSERT INTO comments VALUES
(?,?,?,?,?,?,?,?,?,?,?,'','','','')",
          (index,
           item['cid'],
           item['text'],
           item['time'],
           item['author'],
           item['channel'],
           item['votes'],
           item['photo'],
           item['heart'],
           item['reply'],
           item['time_parsed'])
    )

# commit changes
connection.commit()
```

Test - count rows

```
import sqlite3

# connect to db
connection = sqlite3.connect("comments.db")

#  create db cursor
cursor = connection.cursor()

# count rows
rows = cursor.execute('SELECT COUNT(*) FROM
comments').fetchall()
print(rows)
```

Search for “haters”

```
SELECT * FROM comments
WHERE LOWER(text) LIKE '%hate%'
  OR LOWER(text) LIKE '%stupid%'
  OR LOWER(text) LIKE '%idiot%'
  OR LOWER(text) LIKE '%racist%'
  OR LOWER(text) LIKE '%disgust%'
  OR LOWER(text) LIKE '%nazi%'
  OR LOWER(text) LIKE '%kill%';
```

Delete

```
# delete all rows
cursor.execute("DELETE FROM members")
cursor.execute("DROP TABLE members")
```

Clean up

```
# close connection
connection.close()
```

See SQLite documentation for more

https://www.sqlite.org/docs.html