LE QUERNEC Loévan CDOF2

Containerization Technologies – TD 5

Step 0: GitLab

The connection is ok!

Step 1: Populate the database automatically

I already done this in the previous practical work. I created a file named 'init.sql' that does exactly the same thing than before, and it is executed when the container is launching (thanks to the docker-entrypoint-initdb.d directory in the database container). First, the sql file, named 'init.sql', slightly modified with more tables and data:

Containerization Technologies > TD 5 > init.sql > {} SELECT

```sql
-- init.sql
-- create the database
CREATE DATABASE db;

-- move to the database
\l db;

-- create the table
CREATE TABLE courses (course_id serial primary key, title varchar(100), description varchar(200), teacher varchar(100));
CREATE TABLE students (student_id serial primary key, fullname varchar(100), age int);
CREATE TABLE following (student_id int, course_id int, primary key(student_id, course_id));

-- insert some data
INSERT INTO courses (title, description, teacher) VALUES ('Math', 'Math course', 'Mr. Smith');
INSERT INTO courses (title, description, teacher) VALUES ('English', 'English course', 'Mrs. Johnson');
INSERT INTO courses (title, description, teacher) VALUES ('Science', 'Science course', 'Mr. Brown');
INSERT INTO students (fullname, age) VALUES ('John Doe', 20);
INSERT INTO students (fullname, age) VALUES ('Jane Doe', 19);
INSERT INTO students (fullname, age) VALUES ('Jim Doe', 18);
INSERT INTO following (student_id, course_id) VALUES (1, 1);
INSERT INTO following (student_id, course_id) VALUES (1, 2);
INSERT INTO following (student_id, course_id) VALUES (2, 2);
INSERT INTO following (student_id, course_id) VALUES (3, 3);

-- check the data
SELECT * FROM courses;
SELECT * FROM students;
SELECT * FROM following;
```

Then, the line I added in the 'Dockerfile.db' file (the one for the database container):

```dockerfile
# copy the init.sql file to the docker-entrypoint-initdb.d directory
COPY init.sql /docker-entrypoint-initdb.d/
```

Step 2: Port

I define the ports that my app and db containers will use in the file 'up.sh'.

The commands modified:

- '**docker run -d -p 8080:8080 --name app --network my-tiny-network app**';
- '**docker run -d -p 5432:5432 --name db --network my-tiny-network db**'.

```
Containerization Technologies > TD 5 > $ up.sh
# up.sh
#!/bin/bash

docker run -d -p 8080:8080 --name app --network my-tiny-network app
docker run -d -p 5432:5432 --name db --network my-tiny-network db
```

Now we test it, and the connections are ok:

```
blxucreep@DESKTOP-0FKDJG2:/mnt/c/Users/Loeva/OneDrive/Bureau/ESILV/A4 cycle ingé DIA/Semestre 8/~ programmation/
Containerization Technologies/TD 5$ nc -vz localhost 8080
Connection to localhost (127.0.0.1) 8080 port [tcp/http-alt] succeeded!
blxucreep@DESKTOP-0FKDJG2:/mnt/c/Users/Loeva/OneDrive/Bureau/ESILV/A4 cycle ingé DIA/Semestre 8/~ programmation/
Containerization Technologies/TD 5$ nc -vz localhost 5432
Connection to localhost (127.0.0.1) 5432 port [tcp/postgresql] succeeded!
```

Step 3: Database insert

I created two new routes: one to get all the students (not required, but I prefer to test my app with a GET first) and one to insert a student already defined in my script. Here are the routes defined, first, the GET:

```python
# retrieve the data from the database
def get_students():
    try:
        connection = psycopg2.connect(**db_config)
        cursor = connection.cursor()

        # query
        cursor.execute("SELECT * FROM students;")
        data = cursor.fetchall()

        cursor.close()
        connection.close()

        return data
    except:
        return []

# route to display the data
@app.get('/api/get')
def students():
    data = get_students()
    return jsonify(data)
```

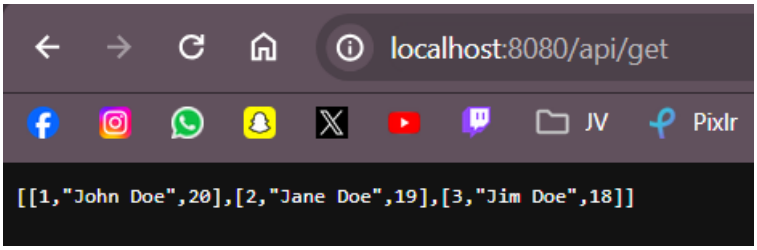
Then the POST:

```python
# insert the data into the database
def insert_student():
    try:
        connection = psycopg2.connect(**db_config)
        cursor = connection.cursor()

        # query
        cursor.execute("INSERT INTO students (fullname, age) VALUES ('William', 21);")
        connection.commit()

        cursor.close()
        connection.close()

        return 'Data inserted'
    except:
        return 'Error inserting data'

# route to insert the data
@app.post('/api/insert')
def insert():
    data = insert_student()
    return jsonify(data)
```

Don't forget to add the database configuration:

```python
# database connection configuration
db_config = {
    'host': 'db', # db container name
    'database': 'db', # database name
    'user': 'postgres', # default postgres user
    'password': 'root', # password
    'port': '5432' # default postgres port
}
```

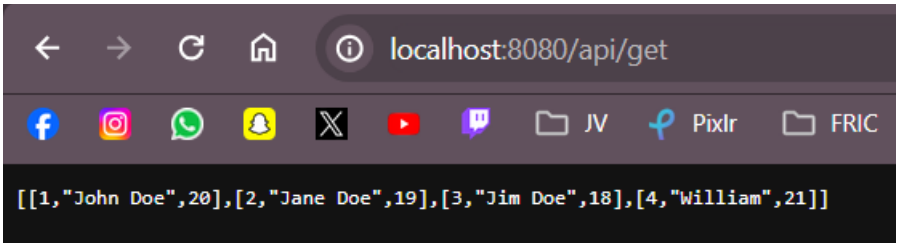
Now I test my GET route, and I can see my students after re-building my images and re-running my containers:



I execute the curl command in the shell:

```
blxucreep@DESKTOP-0FKDJG2:/mnt/c/Users/Loeva/OneDrive/Bureau/ESILV/A4 cycle ingé DIA/Semestre 8/~ programmation/
Containerization Technologies/TD 5$ curl -X 'POST' 'http://127.0.0.1:8080/api/insert'
"Data inserted"
```

Now, we check again our GET route:



And my new student has been inserted, my POST route is working correctly!

Step 4: Create a docker volume

Now, I modified my 'up.sh' script to automatically add a volume to my db container, if it doesn't exist. I did the same for the network, to avoid errors (if I launch these scripts on another computer).

I also added an environment variable 'PGDATA'. In fact, it's the path where the postgres data is stored by default (/var/lib/postgres/data), so I redefine it when launching the container to point the desired directory (/data). Here the Dockerfile from the postgres:14 image:

postgres / 14 / bookworm / **Dockerfile** ↑ Top

Code Blame 226 lines (209 loc) · 9.92 KB Raw

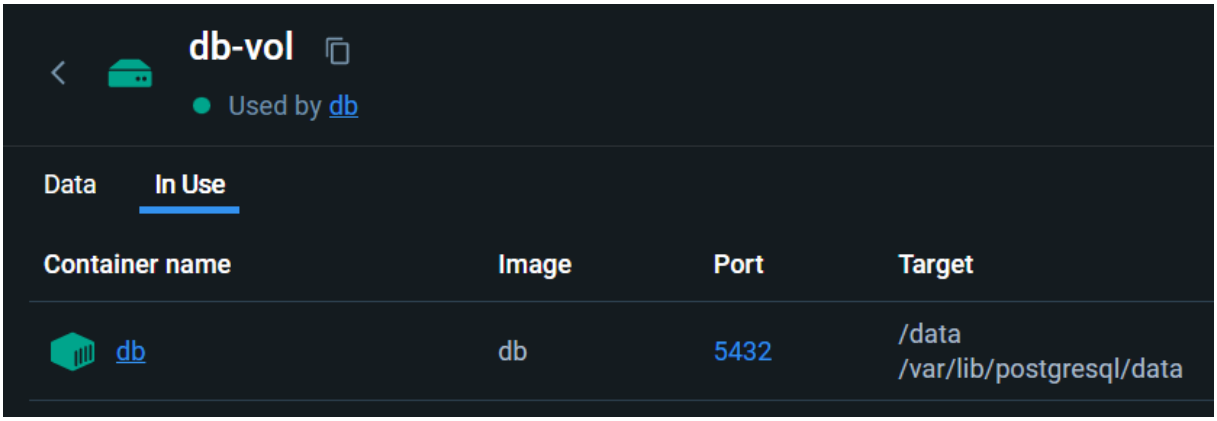
```dockerfile
# make the sample config easier to munge (and "correct by default")
RUN set -eux; \
        dpkg-divert --add --rename --divert "/usr/share/postgresql/postgresql.conf.sample.dpkg" "/usr/share/postgresql/
        cp -v /usr/share/postgresql/postgresql.conf.sample.dpkg /usr/share/postgresql/postgresql.conf.sample; \
        ln -sv ../postgresql.conf.sample "/usr/share/postgresql/$PG_MAJOR/"; \
        sed -ri "s!^#?(listen_addresses)\s*=\s*\S+.*!\1 = '*'!" /usr/share/postgresql/postgresql.conf.sample; \
        grep -F "listen_addresses = '*'" /usr/share/postgresql/postgresql.conf.sample

RUN mkdir -p /var/run/postgresql && chown -R postgres:postgres /var/run/postgresql && chmod 3777 /var/run/postgresql

ENV PGDATA /var/lib/postgresql/data
# this 1777 will be replaced by 0700 at runtime (allows semi-arbitrary "--user" values)
RUN mkdir -p "$PGDATA" && chown -R postgres:postgres "$PGDATA" && chmod 1777 "$PGDATA"
VOLUME /var/lib/postgresql/data
```

Here the script 'up.sh':

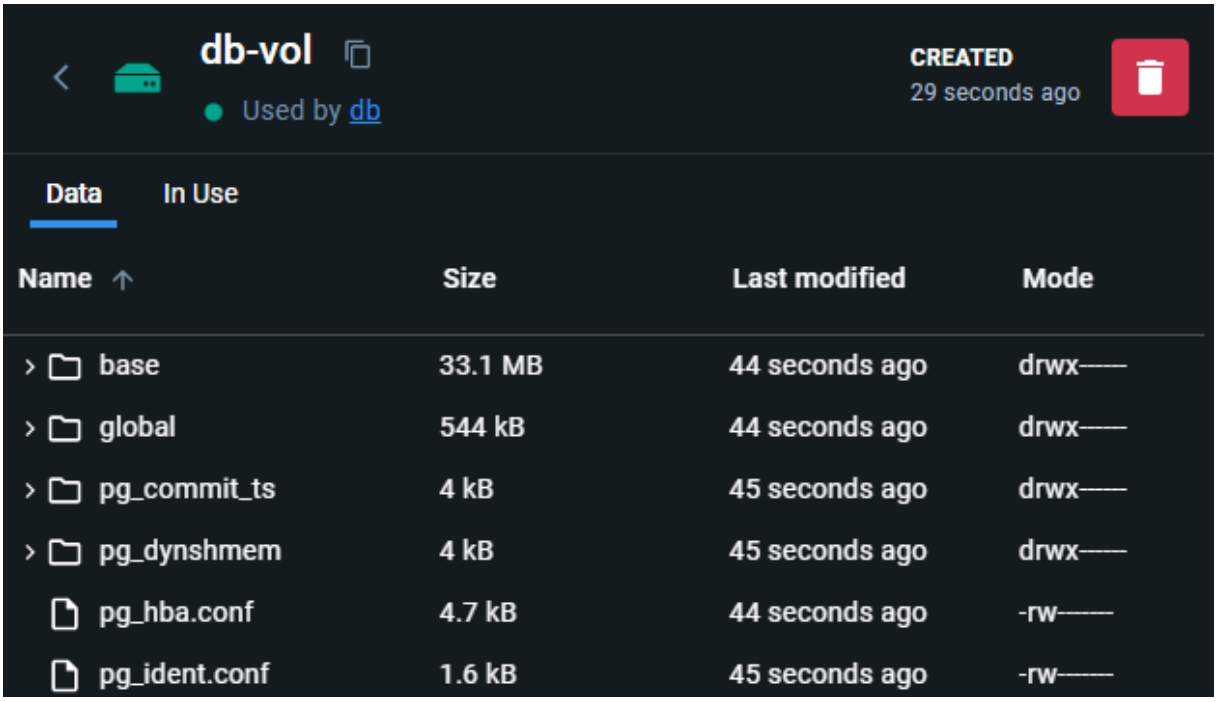
Containerization Technologies > TD 5 > $ up.sh

```bash
# up.sh
#!/bin/bash

# create the my-tiny-network if it doesn't exist
docker network create my-tiny-network || true

# create the db-vol if it doesn't exist
docker volume create db-vol || true

docker run -d -p 8080:8080 --name app --network my-tiny-network app
docker run -d -p 5432:5432 --name db --network my-tiny-network -v db-vol:/data -e PGDATA=/data db
```

I re-launch my scripts, and and my volume is created:



And I can check that I have my data inside:



Step 5: Git submission

The output when I create the ssh key:

```
Loeva@DESKTOP-0FKDJG2 MINGW64 ~
$ ssh-keygen -t rsa -b 4096 -C "LoevanLqc@outlook.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/Loeva/.ssh/id_rsa):
Created directory '/c/Users/Loeva/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/Loeva/.ssh/id_rsa
Your public key has been saved in /c/Users/Loeva/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:JknxThZPvfTPGbVYvkNI3LeTb4UcuStMe0EA6j33lIM LoevanLqc@outlook.com
The key's randomart image is:
+---[RSA 4096]----+
|      . . ooo... |
|       o =  oo+oo|
|      . = ...==+*|
|     . * .  o+*X.|
|      o S ooE.==B|
|       o   o+oo==|
|             o...|
|                 |
|                 |
+----[SHA256]-----+
```

The key:

```
Loeva@DESKTOP-0FKDJG2 MINGW64 ~
$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQCxXF5LPehIkLGO+wmgd7cTEXj39Gp8l8H9sk+mnVJ9
gPrq6R2xogtYIhNnaNprZKPHLARQ1VIjc4r0L3HL8wwfzIJ2Tw2ViIhv/BOOgbNMTQZ31EorhOoXChMl
U+fTHNeDW+tuIJc1S+MSbw8D5cMhM5IvPuCFxNaaDgn+J/es19zUn4RNYeg3gnq5Az+qDmW21fyargIM
DUsVdsrH/nMroX6bM++WzHaxLeT7wDDVyaFJEEZ7LseFesq/bSVoiX8m1j1E8TR1o+67bDcAWHW1wfO2
o+sUVEBevE4OyJzRdmg+5TMbZqSAHEgW/oTxAM81NlLLURgo2vy8RxeXj8aLtl6jb7S9ybqcfuIZduTI
3KUIiql65O7aDi4v/zpj6A9X65DYHZ+J9lkFl75e2uLELO44cICLAoVfTNRVAmcaUxf20Q67lmhOOdhe
Et4AiF3hDUZgNgcUV4ioqK4ktA2/HWaq8NO3yt2lUpy3DVSStSSWFWM1VLPE46kOH2ye1afswruqGOld
+YXajRGWW3WgEbTyvJ4n9vwe//QJYchRhc1ZgtAJrBxY4+6Bd1CIoXXMFmpUtcSQ1otPA1KYFMYsbyYM
2A1IPw7nq/owI45od1XmlafYClAwocXKF49FJEUMVBcXCqCjKBzQgN6XLhrppn3UlrPlCUmyT9q8IXGj
qQ== LoevanLqc@outlook.com
```

My key has been added in GitLab:

SSH Key: Blxucreep-laptop

Key details

Usage type	Created	Last used	Expires
Authentication & Signing	Feb 26, 2024 9:38pm	Never	Never

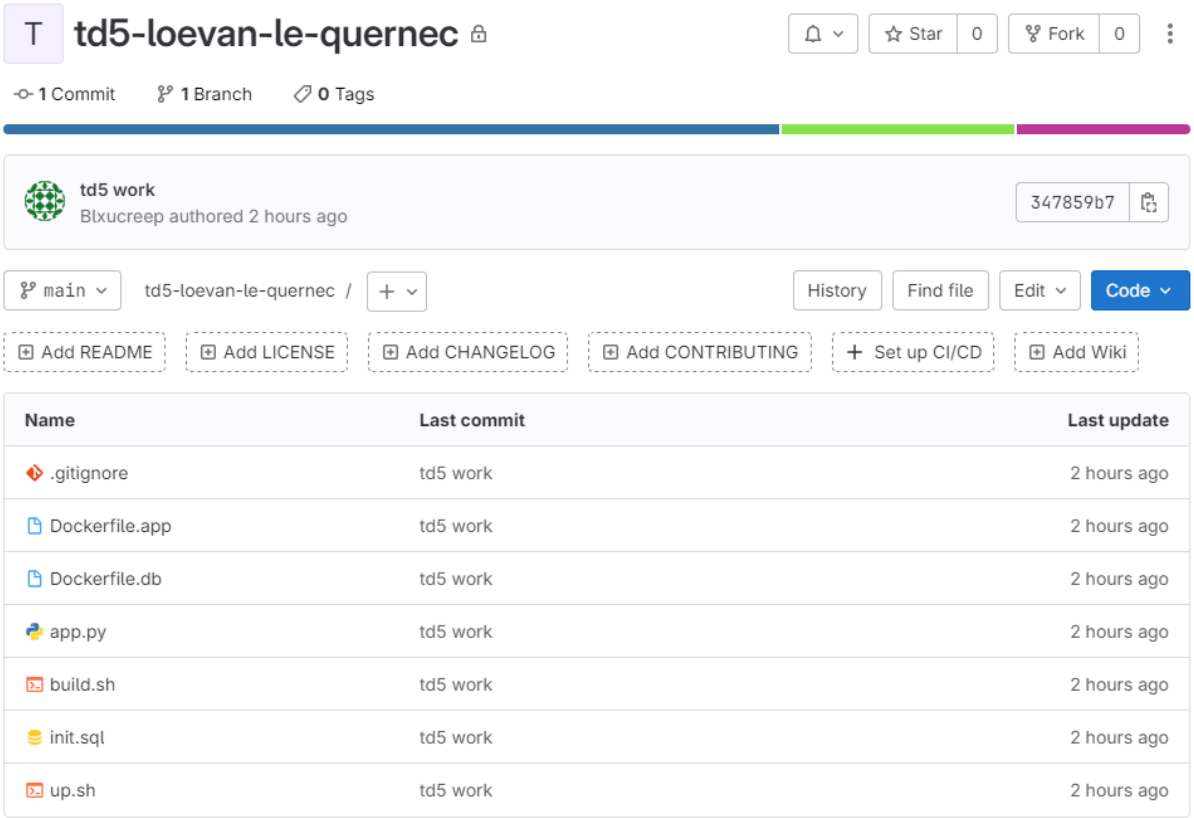
SSH Key

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQCxXF5LPehIkLGO+wmgd7cTEXj39Gp8l8H9sk+mnVJ9gPrq6R2xogtYIhNnaNprZKPHLARQ1VIjc4r0L3HL8wwfzIJ2Tw2ViIhv/BO
```

Fingerprints

MD5	dd:e7:ce:bd:40:50:b7:2c:e7:d1:9e:dd:b9:8b:6b:dc
SHA256	JknxThZPvfTPGbVYvkNI3LeTb4UcuStMe0EA6j33lIM

After executing the commands for pushing an existing folder, all my work is here:



Bonus: GPG key

I did this command, and followed the instructions: '**gpg --full-generate-key**'

```
gpg: /home/blxucreep/.gnupg/trustdb.gpg: trustdb created
gpg: key 820B93E059D72C47 marked as ultimately trusted
gpg: directory '/home/blxucreep/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/home/blxucreep/.gnupg/openpgp-revocs.d/D188C8BA47100337A0928B28820B93E059D72C47.rev'
public and secret key created and signed.

pub   rsa3072 2024-02-26 [SC]
      D188C8BA47100337A0928B28820B93E059D72C47
uid                      Loévan Le Quernec <loevan.le_quernec@edu.devinci.fr>
sub   rsa3072 2024-02-26 [E]
```

My GPG key ID: 8ED417CD0CE73E89.

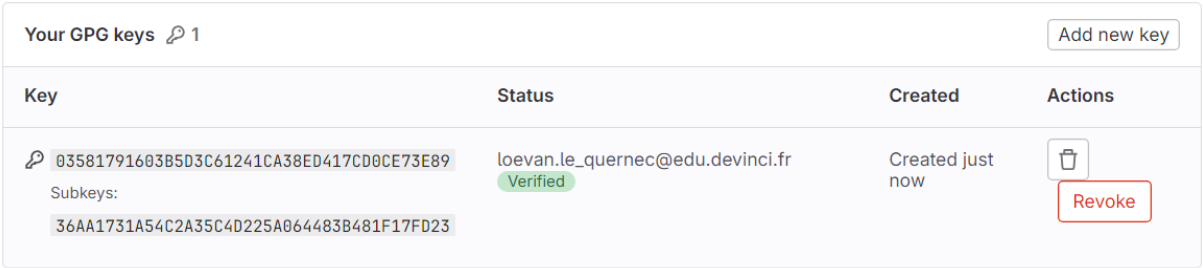
```
Loeva@DESKTOP-OFKDJG2 MINGW64 ~
$ gpg --list-secret-keys --keyid-format=long
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   2  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 2u
/c/Users/Loeva/.gnupg/pubring.kbx
---------------------------------
sec   rsa3072/849DC4EAAF10CB2F 2024-02-26 [SC]
      20EC8AA0A6D71D8AAD163C1D849DC4EAAF10CB2F
uid                 [ultimate] Blxucreep <LoevanLqc@outlook.com>
ssb   rsa3072/B9BFC468EBA81669 2024-02-26 [E]

sec   rsa3072/8ED417CD0CE73E89 2024-02-26 [SC]
      03581791603B5D3C61241CA38ED417CD0CE73E89
uid                 [ultimate] loevan.le-quernec <loevan.le_quernec@edu.devinci.fr>
ssb   rsa3072/64483B481F17FD23 2024-02-26 [E]
```

Then the public key, using this command:

‘**gpg --armor --export 8ED417CD0CE73E89**’

```
Loeva@DESKTOP-0FKDJG2 MINGW64 ~
$ gpg --armor --export 8ED417CD0CE73E89
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQGNBGXdCGQBDADSE8JZCeHWJ6pFwRCAlfX1pIzdpIdENW/1bVrxIJl6rFpbH0BI
PwjMfX55ONdV/Kyaa5EtGlBjg4JR/Floqhb54t+2frjjlbvPnSEAA2BxQjHuZzQu
3iznGpu7/bFyN0Ioo3Ip9upVZGKuU7AJkfe/CzcObcuL4OK0T4YHlrfyi2FZHcdo
5jXPK1MzPc3Trd4khBD8TZusgvGdvqGQKQhwUNzkS7uK3cm09Pga7yNOKl3FC7r1
sEaLo1pKx6tKmfdfmhN30tI69LHi4ti74D+X+1GyLq1tqzdtEbaR1QFS4JiqUA2N
IDEf0JwG9DS5QfbcxjF+aenBC8azsWVPdK9Wwn0UiTf+TBaNYDTZ6b5BipHXzUdT
2PoVcxGoOmU3lHC/GdD1N9/SZEe2tqEmFwfZwL+bW+WWw0qgCy9kjuGS6X99Wsk3
ncIiikP3jx8f7nocTEAhVFDgopXPfxeSQewOdijNEN8+WGiS7W9DPPICPPkYwnAG
EoQc0vVlMHfY4KMAEQEAAbQ0bG9ldmFuLmxlLXF1ZXJuZWMgPGxvZXZhbi5sZV9x
dWVybmVjQGVkdS5kZXZpbmNpLmZyPokBzgQTAQgAOBYhBANYF5Fg0108YSQco47U
F80M5z6JBQJl3QhkAhsDBQsJCAcCBhUKCQgLAgQWAgMBAh4BAheAAAoJEI7UF80M
5z6J9y4L/0A+kzBpbTXv8ahGgCzV5ucT/qxxdr5qzKc+RfEHZsxQiORGveq36mX5
dV0NOpXhfgA57i5PEOvwkdpMJswk0unwg0kCHPaHj4RWi/tHj1nZ9j4+dcwe3aVl
BWfQRymReTWgq6rGs7y+fuFZ5RRfzSp0mp4beFhbv2Z5CStr8zexLJfpa5wtG0PK
AWp4m0q673EmjwEa89w4MMPYfq8tG+icy9ezfHMeBRT2j4OLV6ZpUoHXCxAwCZVN
KMWn+WJpnWj1jbqAc1/AUs2Oh+8UtPnVJmL6GNjISLaYQV/exQZC1i7FpxQQFoJn
I26JHvJYQA2m0J875qxNB1ueuKljLd1vTR32YGN3fO/lTdhyQtNy15t+9smX1nIs
xYK6Rj8n6QqBBqQoBqXedmDYIdS1OjsyXruX72gdSwNbi1EQTweyZ9mmfqXa9L5k
vMSymt2ojQlNeJcXVBJSJEDR0XY/pSph0h8hJ9ovx+Af23sPHIm/iF7goWQzwLua
QpGJqnL2JbkBjQRl3QhkAQwApI+tdjtvke5P3E3ltpP7+70K1p4L9VF5wMU5toHR
bnnWJT6G+t2k5WxkGT+OqdfV2oK43Sc79/LP7+m2/BpCcvRUSBvp/gY1pHBZ4Gzl
aKBGY92wUQJgj2ATsxVEbqRrxSYp0SSk9iKxp0dTWi2+ik68nafjtaAYBrbOAqml
zWuKNluu5WTZL8qWC0QBJY5GviwSwpln1V5Y15aeXxz5xUitJ6OWDyHXKfsQU38S
BfawzdL7d6AVR5ty9s2/aeRrCvreRMcr6PL04b85OwtyHPJJFip00IZ3LsHD4SiI
PzW1KNaaCcvcPnlV5AI1ZKY/ESOnSrhS6voLpxMjb9a9/pc8H3ejSfXgExC0ur9o
x12MrIQx4wKjC4mXr1kc9QBt5ZLoMXGwKt5nbh0QtVyyagsXupMu7LQZEgZPdDQc
4ciuNkeNIWBNqyosfgvm+o07/NKlca5X0v/eDifrzHEfegLyS6MLFJxw/PvV/GN9
e9+uFrdukUyr3sMH225wzu5jABEBAAGJAbYEGAEIACAWIQQDWBeRYDtdPGEkHKOO
1BfNDOc+iQUCZd0IZAIbDAAKCRC01BfNDOc+idpcDACNxesTcX+BdM5mAQMSrD6/
odE+8WjqMhzUDjVvQ8xl0lrQaHmXKRqya3PwjOblYR9h/OfnreC0t5e2C+gRLn67
0JMdmMIHQYt/jQYDGnPov4HU2T7FavJnG6EGmAIbz+qf8DE4140vcIr2BcR/i54X
Pv/r1pwqf6vLdSPIcWajcIe6QJm34KUrTg4oEpQzg5C0DGbWhaz3PQf8ASz8Ilzk
SjM74pZVuPyN16WfS+HrVGVpINEUn4jRDvhbjjvcxZQzqXVp2fkAHE2sb6l+tb08
em/fuhmv5nox8lAjD3hCl7hkt51Qbl7qwSO9nkyEAcoiEKFZBwv43dY4wMxa6ysH
p0S1kCeP9izbKWSOtGFbON+/VZqQhEOR6jVfUKCrib7JpUABBEqa8JOjM8hCVYG4
9AoHUvkfOfwjCgXDgWMV2VdERNzvcC4JhNLCZ2RhwIG2EGG7BG2OJiT7UMupjg1f
FhOxCZukh3H2aUHt1Estk8h+JkLVQhR06jpAXH2alh8=
=ZpVo
-----END PGP PUBLIC KEY BLOCK-----
```

I copied this in GitLab, by creating a new GPG key:



You can see that my last commit (the one with adding this .pdf file) is signed with my newly created GPG key. I used this command to commit (with signature ‘-S’): ‘**git commit -S -m “adding report”**’.