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)
```

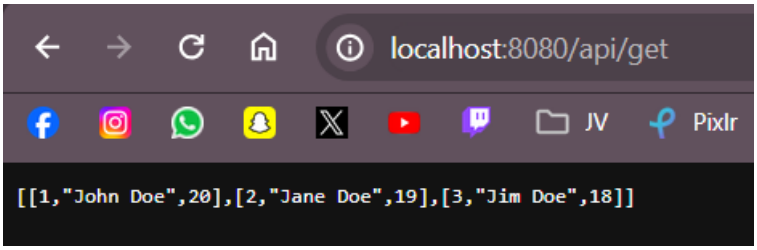
Then the POST:

```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
}
```

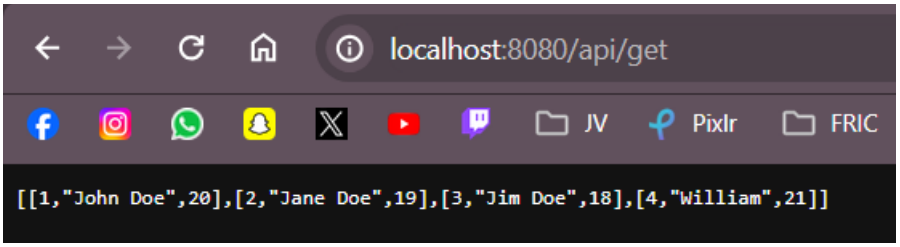
Now I test my GET route, and I can see my students after re-building my images and re-running my containers:



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

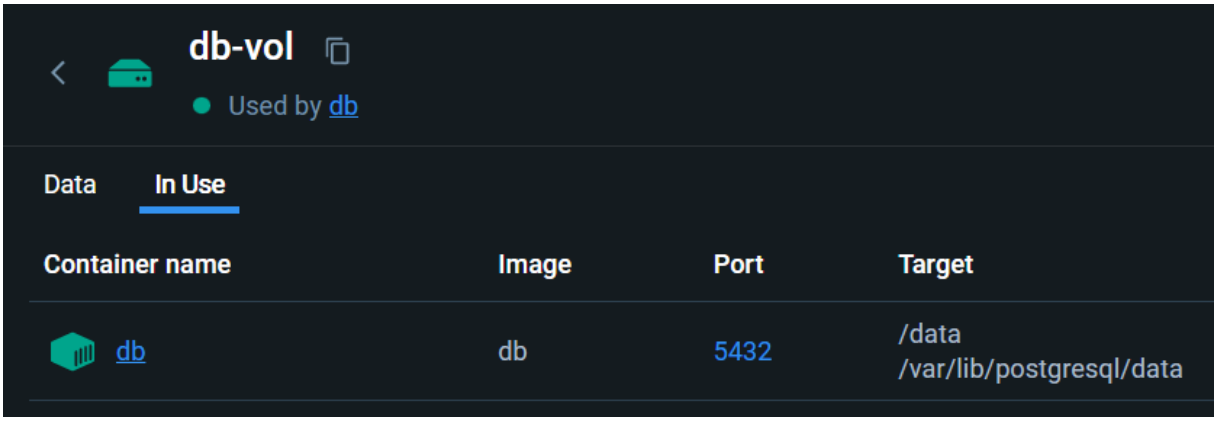
```
# make the sample config easier to munge (and "correct by default")
RUN set -eux; \
        dpkg-divert --add --rename --divert "/usr/share/postgresql/postgresql.conf.sample.dpkg" "/usr/share/postgresql/
        cp -v /usr/share/postgresql/postgresql.conf.sample.dpkg /usr/share/postgresql/postgresql.conf.sample; \
        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':

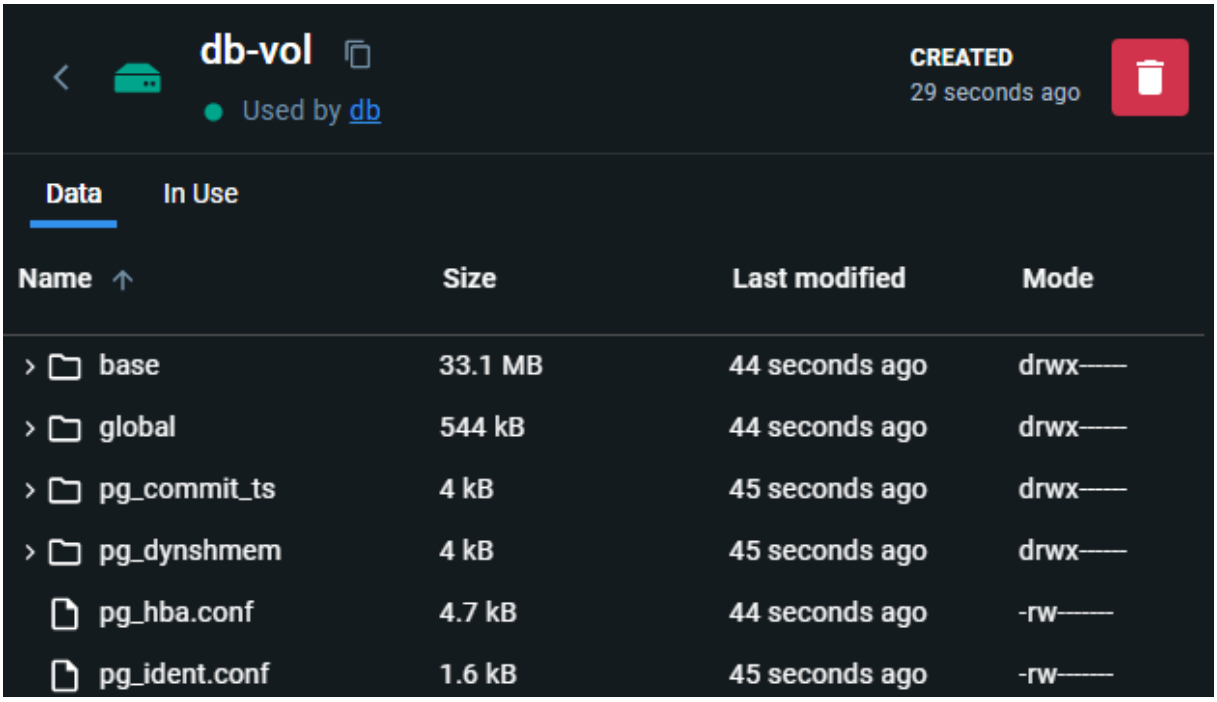
Containerization Technologies > TD 5 > $ up.sh

```
# 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:

```
blxucreep@Blxucreep:/mnt/c/Users/Loeva/OneDrive/Bureau/ESILV/A4 cycle ingé DIA/Semestre 8/~ programmation/
Containerization Technologies/TD 5$ ssh-keygen -b 8192 -t rsa -f ~/.ssh/id_rsa -N ""
Generating public/private rsa key pair.
Created directory '/home/blxucreep/.ssh'.
Your identification has been saved in /home/blxucreep/.ssh/id_rsa
Your public key has been saved in /home/blxucreep/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:R7sBFWo2hBIMyQ5LlM/mDXtSBckCszU2D+WgXbsxIV0 blxucreep@Blxucreep
The key's randomart image is:
+---[RSA 8192]----+
|.=+%=*+E. o.     |
|.oX.%++o o       |
|.*o..B. * .      |
|. .= .+o + .     |
|  o =.  S +      |
|   + o   . o     |
|    o     .      |
|                 |
|                 |
+----[SHA256]-----+
```

The key:

```
blxucreep@Blxucreep:/mnt/c/Users/Loeva/OneDrive/Bureau/ESILV/A4 cycle ingé DIA/Semestre 8/~ programmation/
Containerization Technologies/TD 5$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAAEAQC5V2COOa3vMypvmBBqnuoK337zIe1I9pQTyMZMQk6uurrZZhqItL926I2l4K8Pr5hQyj
Ex8kyeWIB+lCynioS+GLGqzV1SqnmF1aHcYGTe1N9mPBTWwa3yxe9Eq2v40R7nEwFRsbD6sTWF3DmBvu0SiQL/dvCHkmFHdYxNUoOJkqX9
WyMwTt3xKpQrmhN/amY+U0KlaHsUu+cwi14YkqBOnBpzfztkuaEishG3xb6wojJ/JPaObP9nO7FimJoN3ZH6q2iZqWuGO1c0KaehsbCvy8
Rtrm3MiaavH7rnG1OD9z/DHt2O/+nyP5ktcg9Kn8VuEVjjUEDO0hS7YQkedReiOyBrH0OH0xJ9ZVnZVYiJDwzwtYOvZ6SDbFKkA/qIxIbF
mxGq4GmJQ9nDlQyETNL9OMdHUu67XnsrLzpmHhwVHPFA0Np4A/9d0Fni8FG57puMW+JG3SOzWT2+yN9ikmYws84etKQ2TPZS8Po9zLV9tc
GuRXxxkdF8brtIPFJ63nlshi7GSZ2CGB6aDQ/BmIdaMSnxwdulDJDiXdoOYUp7JAxVyn0jFxZBLbByf+C0XFf6ljKMaYVnHLSAIVGiiiXB
3J/3gSROfVH8ft3zhRJWcJjQxysaelEV6kmay6BzQJJRX5+cjK8QDqmDl/C1+Vom3f9AhmI0z+Dlvh4PA3RLDmO4h0L/iY6E9PhgPPbeVQ
wlrsnq+gDaOL68CyEpVo4GlkltNmhZSwhozFnwQuZ+PFLYO+jADfcxag47P2c1TLbBiiq7FV3qnRwVXCx1nHMhsuBQXuqW+zYFw0Akm942
qc65qDfoeDK7TOk3nR/ePX2KnEey4wx/q9kbJqKWYNNnl0X1jXs0Rgp1DyJDZEnbcqkJlb10XggVPPlu9hertGmbQ2hsAqKM7s0G39q3kb
daVSrDOe7zWSOPFwE0oyLMiYPgst/TbJObmiz9eqChKDDi95v37QZUfCLCHTY/mPWdQX65oNrfgnkrvmXAG7NeAnMzHNaCkfvVh8vxAxwN
Qh1tLWN12ZucSN/x5Uxpxomeh/2pQnpRAvo9Ctg8mrDcayr5gM90ur0NITJCD4/043bw9hphDUouWBs1C7QDsPeD3vEyd7pS5NVnB5kRtI
ZAc3TwfcA+0ytS6dF1FV+OOO3zfbRZKGrdbfhi1pID6GJ4JPSj1JjvVBkeX4RfopUOn2jVPxeBZSnocUhsHrb18X4khFhBFkYXWNxaAqZP
cYItu4qO63EeJVyh5qPgifDAv7nT4wplMJXRtZcwogf4MFH3BVewIGH2u0nUbUXIM/7M2kvbQA6iM5sHe0xdEi0QdtYXXW1ESWnwG8kzQc
mBKA8Xh24UmJt8fhwF2ZHu0IL3 blxucreep@Blxucreep
```

My key has been added in GitLab:

SSH Key: blxucreep@Blxucreep

Key details

Usage type	Created	Last used	Expires
Authentication & Signing	Feb 26, 2024 4:00pm	Never	Never

SSH Key

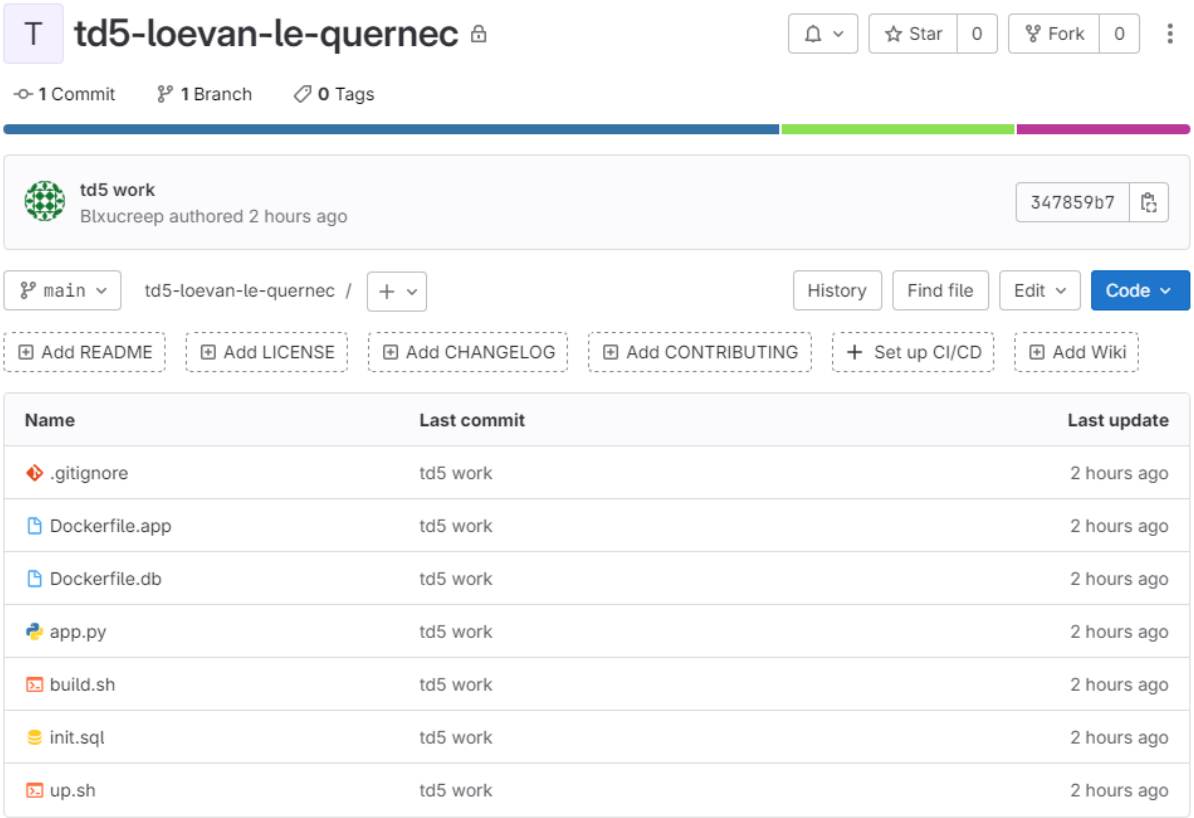
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAAEAQC5V2COOa3vMypvmBBqnuoK337zIe1I9pQTyMZMQk6uurrZZhqItL926I2l4K8Pr5hQyjEx8kyeWIB+lCynioS+GLGqzV1SqnmF1aHcYG1

Fingerprints

MD5	ed:27:f5:a6:c2:2c:2e:61:74:b7:87:44:c1:4d:f9:d6
SHA256	R7sBFWo2hBIMyQ5LlM/mDXtSBckCszU2D+WgXbsxIV0

Delete

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: D188C8BA47100337A0928B28820B93E059D72C47.

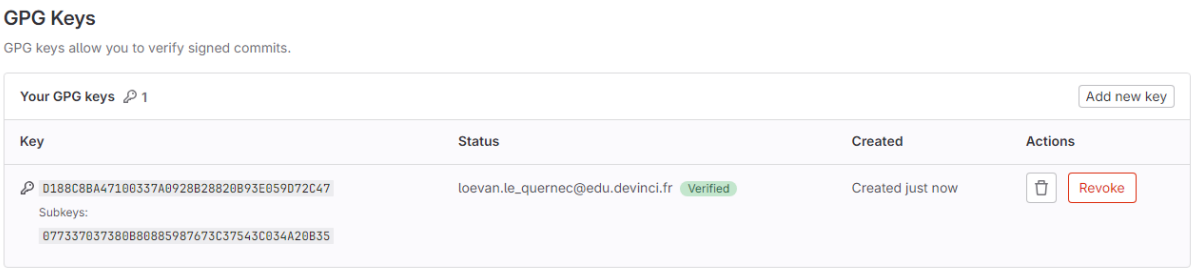
```
blxucreep@Blxucreep:/mnt/c/Users/Loeva/OneDrive/Bureau/ESILV/A4 cycle ingé DIA/Semestre 8/~ programmation/
Containerization Technologies/TD 5$ git config user.signingkey D188C8BA47100337A0928B28820B93E059D72C47
```

Then the public key, using this command:

‘gpg --armor --export D188C8BA47100337A0928B28820B93E059D72C47’

```
blxucreep@Blxucreep:/mnt/c/Users/Loeva/OneDrive/Bureau/ESILV/A4 cycle ingé DIA/Semestre 8/~ programmation/
Containerization Technologies/TD 5$ gpg --armor --export D188C8BA47100337A0928B28820B93E059D72C47
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQGNBGXc28gBDACyV73idg6BHDFRbr+BOLTJbCjJVoscYsl2Sii8dz6zRbyZdAKi
O8xMvJFxhZm9JIito4W1h6yuZsONrN3KjNtEE1vBCnEEA70pA2/GVva4yrjC3YwC
fdQNnPJQySPgJErLw4ZJLMS7w7CU/gcvyt9ZVNkbHGitkB2/r2IsVlFw/9O4sb+L
Hv7AgRfwCHO9CEpUJj1OVW75ip9eauop9ZMTP50oV0Wy6lgUZWl4bj0DY4eCjKq4
i3aLViJVI5BhqQqYq1T2TgPYEg2d1Mx1t5ZVMQBBVTEOZBubjDrJcGLldF+dAWhR
+sj7/JfMA6wqz4Lgf4FQ6AuxLE5xgCze38oKd0u2vbIQ2OCBdYUcJHYr5esSUFPf
71WcwQvdBwto96xog40J3+FgStPuNlk/XML6nDeGYXYhJEE40FNbCsVx0joQaOEm
feti2Pgkj/FalMo7otfpXUL9vjyL1dnFDZPs6Kaccd59FZA9YFv/gbaEeLX0yUDz
qmOAkc/aUM9TNnUAEQEAAbQ1TG/DqXZhbiBMZSBRdWVybmVjIDxsb2V2YW4ubGVf
cXVlcm5lY0BlZHUuZGV2aW5jaS5mcj6JAc4EEwEKADgWIQTRiMi6RxADN6CSiyiC
C5PgWdcsRwUCZdzbyAIbAwULCQgHAgYVCgkICwIEFgIDAQIeAQIXgAAKCRCCC5Pg
WdcsR0Q/C/9EoC8NUz+u3ZLnYwqDnyvfRTxrBjm2eWcssAhf8RwD8pudY9m+y8kG
IrcnQK5EqYxsUfSHIJQFrFAX9+TCPynlcjeqeUevllNPc2MoGSE9Wrw7ROc/XDDP
Gn2LtG8VG7ZdaX06eBSI/7JT1QyD8frroIFA3icBH3e3cRdY/wvS0Ca11Hcm88/D
XJNd3+ahXb1T5KuSSvQToavBRkXb529zmOXzpF/se+lIhFs0UjGausVNxby3eBml
sSxQna2uwNWIkN8xzikwA13mdIMmsy9DnPyOdD1F/daUHBzf6QFnIrA6gso08rR1
03GEronI59hOdWqnHZ1iG9vjdeawxHKKvjdtQipEDq1NyzTKzOazuspJQ2F1/Fj9
RkThz+Je5QyfEgLjykyv+QHD9xkunz8u/cNPDDiuRLCC8EJQhobYfHTcGQzr7pRX
fuAdB5iKuRn35tm33dvhi66lIW00ujRwobbQjSDRTTt1+EnuL8c1VVKqwQ6AVEc+
k3rr/TAL1Jq5AY0EZdzbyAEMAKGNWqJYcZfFT1hXXaXT0a48MQKY7PWeQpFKmoVS
CZgNf4KinEQeT9TeEaNYyPAm7/vH6PWIJW0M8hC0Zt1Vzd6wcmO40cT5OTKmZp2v
SHAns1Ad934N/CRP2T1Ngzm7ZkcU3qNQRUNaNKoWXedO86nMPopxP4ZWARrCKVk3
jO1/tNku6FEbav/wu6t8QnW/+GuMTtdek6OvkkwTO+5IL/6ZxDtAAxn1jhJqaX3Y
MdDPqQQpJCZAiKWe5XwH5P1vdZbJruOjM1AaGYAfaG2CP+Gki7C7sYHozsHY35fC
q7UdkrPNVEJQIymfKf+tFTaRnLUbRw4Rm0Wbdfg0vbPFsRdR/1xzUfvB4saizjfX
Zgx/YGjZssUGSHpq8k/jo18BndP4juDVeZApMejd9YN0qkqtLoHfS7CrRWCSOlbw
IKBNcLTaGBSdLtHwu6iCqZY6YIVPCRtU1TgXOa/8l8ZNpv7zvXugmFaz8pBis2bq
ca6olJJQxUj59nF5rM9lc5enqwARAQABiQG2BBgBCgAgFiEE0YjIukcQAzegkoso
gguT4FnXLEcFAmXc28gCGwwACgkQgguT4FnXLEeJowwArZuYVay2dP5iEY+IThyR
uGx6uIyTpA0CDqFvJ8V8nI2Y8LDPuzoTfHEauXEU8HCcIiUCJ2UwBcl1EJDF0Hi2
zFjLEphURqxUhEdUhECY3LK2fHAIAYibzzwTY2SaFa9APumveHtDuuLEnOTjRKdJ
IIrVrsGIn7LnEHLayjUrIMgYi/F9sQbbrEKNCwbt/L5FNBYWVA3iGCNnbkAbV2Jo
BONlOOSHcDRP3TzMi4P4y45XjHHWM7++5n7/e3maGpGXNaev9GNOg/qEG9FOdtaf
yI6QTibESISUjM9JurtIDVhx9kxEs5yRp/0vmEv/3izIWyCaPcVReTELVq/g7ltm
/ZNekCNbJBCzlSH9yuRu8GtJzLJ7bG+RyA1q2e4tU87rAh83raDStWwKxZSgoDF9
pIkZwWfCOjXGo00a/jMq9jLGWVXQtHdeRcHe2v5YXtQs7GIKPFYxmNs5oKw6GA9r
MzB8DdwHxNhr/ceZ/sR2Q7yO9yvGytypLT8ktMLGVFJj
=wlBr
-----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”**’.