

## 1. MLFlow Flask S3

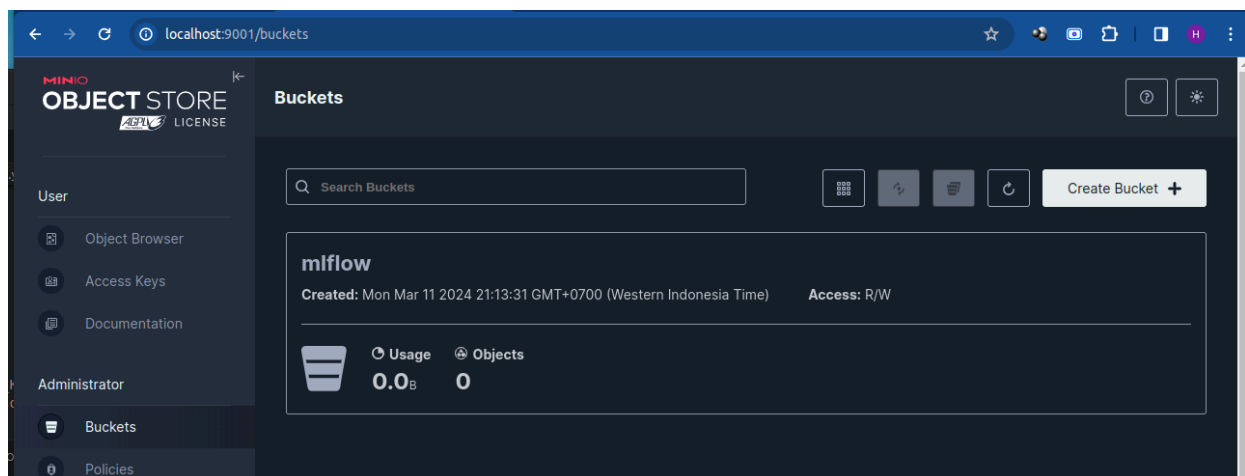
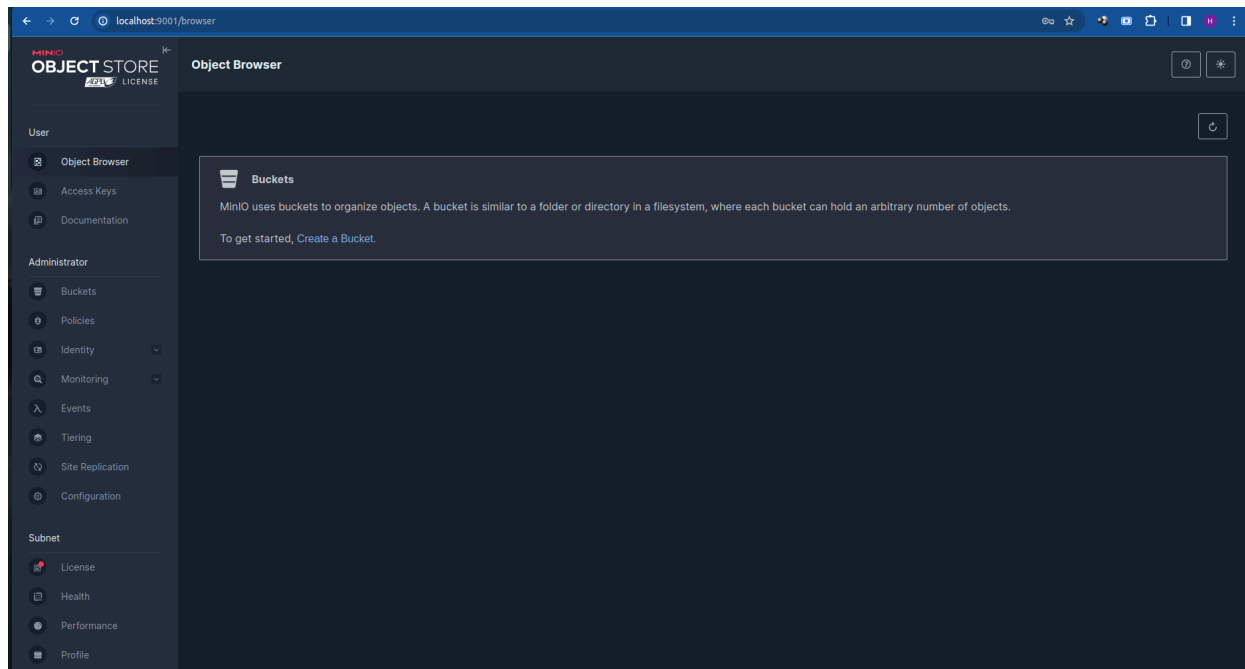
### a. docker logs mlflow\_flask\_s3

```
(ml_flask) hendy@lenovo-ubuntu:~/Documents/digital_skola_de/ml-regression-flask$ docker logs mlflow_flask_s3
MinIO Object Storage Server
Copyright: 2015-2024 MinIO, Inc.
License: GNU AGPLv3 <https://www.gnu.org/licenses/agpl-3.0.html>
Version: RELEASE.2024-03-10T02-53-48Z (go1.21.8 linux/amd64)

API: http://172.20.0.2:9000 http://127.0.0.1:9000
WebUI: http://172.20.0.2:9001 http://127.0.0.1:9001

Docs: https://min.io/docs/minio/linux/index.html
Status: 1 Online, 0 Offline.
STARTUP WARNINGS:
- The standard parity is set to 0. This can lead to data loss.
```

### b. http://localhost:9001/login



## 2. MLFlow Flask Server

### a. docker logs mlflow\_flask\_server -f

```
(ml_flask) hendy@lenovo-ubuntu:~/Documents/digital_skola_de/ml-regression-flask$ docker logs mlflow_flask_server -f
[2024-03-11 13:48:01 +0000] [23] [INFO] Starting gunicorn 21.2.0
[2024-03-11 13:48:01 +0000] [23] [INFO] Listening at: http://0.0.0.0:5000 (23)
[2024-03-11 13:48:01 +0000] [23] [INFO] Using worker: sync
[2024-03-11 13:48:01 +0000] [24] [INFO] Booting worker with pid: 24
[2024-03-11 13:48:01 +0000] [25] [INFO] Booting worker with pid: 25
[2024-03-11 13:48:01 +0000] [26] [INFO] Booting worker with pid: 26
[2024-03-11 13:48:01 +0000] [27] [INFO] Booting worker with pid: 27
```

### b. http://localhost:5001/

The screenshot shows the MLFlow 2.6.0 Experiments interface. The top navigation bar includes 'Experiments' and 'Models' tabs. The 'Experiments' tab is active, showing a list of experiments. The 'Default' experiment is selected. The interface includes a search bar, view toggles (Table view, Chart view, Artifact view), filters for 'Time created' and 'State Active', and a table with columns for Run Name, Created, Dataset, and Duration. The table is currently empty, and a message at the bottom states 'No runs logged' with a link to learn more about logging runs.

c.

## 3. Regression Flask

### a. docker logs regression-flask

```
(ml_flask) hendy@lenovo-ubuntu:~/Documents/digital_skola_de/ml-regression-flask$ docker logs regression-flask
* Serving Flask app 'regression'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.20.0.4:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 745-347-018
```

b. `http://localhost:5002/predict-salary?exp=7`

localhost:5002/predict-salary?exp=7

# BadRequestKeyError

`werkzeug.exceptions.BadRequestKeyError: 400 Bad Request: The browser (or proxy) sent a request that this server could not understand.`  
`KeyError: 'run_id'`

Traceback (most recent call last)

File "/usr/local/lib/python3.10/site-packages/flask/app.py", line 2551, in `__call__`  
return self.wsgi\_app(environ, start\_response)

File "/usr/local/lib/python3.10/site-packages/flask/app.py", line 2532, in `wsgi_app`  
response = self.handle\_exception(e)

File "/usr/local/lib/python3.10/site-packages/flask/app.py", line 2520, in `wsgi_app`  
response = self.full\_dispatch\_request()

File "/usr/local/lib/python3.10/site-packages/flask/app.py", line 1825, in `full_dispatch_request`  
rv = self.handle\_user\_exception(e)

File "/usr/local/lib/python3.10/site-packages/flask/app.py", line 1823, in `full_dispatch_request`  
rv = self.dispatch\_request()

File "/usr/local/lib/python3.10/site-packages/flask/app.py", line 1799, in `dispatch_request`  
return self.ensure\_sync(self.view\_functions[rule.endpoint])(\*\*view\_args)

File "/app/server/regression.py", line 22, in `predict`  
logged\_model = f'runs/{data["run\_id"]}/model'

File "/usr/local/lib/python3.10/site-packages/werkzeug/datastructures.py", line 375, in `__getitem__`  
raise exceptions.BadRequestKeyError(key)

`werkzeug.exceptions.BadRequestKeyError: 400 Bad Request: The browser (or proxy) sent a request that this server could not understand.`  
`KeyError: 'run_id'`

The debugger caught an exception in your WSGI application. You can now look at the traceback which led to the error.

To switch between the interactive traceback and the plaintext one, you can click on the "Traceback" headline. From the text traceback you can also create a paste of it. For code execution mouse-over the frame you want to debug and click on the console icon on the right side.

You can execute arbitrary Python code in the stack frames and there are some extra helpers available for introspection:

- `dump()` shows all variables in the frame
- `dump(obj)` dumps all that's known about the object

Brought to you by DONT PANIC, your friendly Werkzeug powered traceback interpreter.