

amcat-book

2025-12-05

Table of contents

Welcome	4
Acknowledgments	4
1 Why amcat	5
1.1 What is amcat?	6
2 Getting started	8
2.1 Installing and accessing AmCAT	9
2.1.1 Run on our servers	9
2.1.2 AmCAT in your personal computer	9
2.2 Access AmCAT	27
2.2.1 Accessing the GUI	28
2.3 Setup on your own server	33
2.4 Frontend	37
2.4.1 amcat4client	37
2.4.2 API Client	42
3 Document Storage	44
3.1 Manage Documents with the Graphic User Interface	44
3.1.1 Upload	44
3.1.2 Visualize	53
3.2 Manage Documents with the API	57
3.2.1 A Note on the ID Field and Duplicated Documents	66
4 Access Management and Document Sharing	71
4.1 Access Management	71
4.1.1 Access Management via the Graphic User Interface	71
4.1.2 Access Management via the Application Programming Interface	73
4.2 Document Sharing	74
5 Public Document Presentation	75
6 Fast searches in amcat databses	76
7 amcat for document annotation	78

8	Preprocessing and machine learning	79
8.1	Why nlpipeline?	79
9	About amcat	80
	References	82

Welcome

Welcome to the amcat manual.

Acknowledgments

This version of the book was built with the following packages and versions:

package	version
quarto	1.2.313
pandoc	2.19.2
R	4.2.2
Python	3.10
amcat4	
amcat4r	0.0.1.9000
amcat4apiclient	0.9
operating system	Linux 6.1.3-arch1-1

1 Why amcat

An example workflow

Say, there has been a media frenzy recently about a political scandal. You know that it started with a revelation by an anonymous poster on Reddit, spread through social media, and was investigated by respectable media outlets, which created public pressure until the topic was discussed in your country's parliament and a minister or two had to step down. You quickly realise that the case is a treasure trove for investigating your political system and testing several media theories in the process.

You spin up a new instance of the amcat suite on your research server or a new cloud instance and start to collect data through APIs of social media websites, media monitoring sites and scrapers and store everything in the amcat database. You share your data with collaborators, using fine-grained data access control since some of the scraped content is copyrighted. A new research assistant with some knowledge about the case but no technical training starts digging through the data using amcat's user interface and publicly available dashboard to search for potentially relevant terms and queries. One of your collaborators builds on this by producing time-series models and plots in R.

You decide to dig deeper into the content and start training coders and deploy coding tasks through amcat's annotation tool. Coders get through the task quickly as they can access the interface on their phones and code on their bus commutes or whenever else they feel like it. You use your amcat server to preprocess the data and train and validate a heap of advanced machine learning models that complete the coding task of all documents. Your analysis reveals new mechanisms and confirms some of the theories you worked with. After writing up a paper, you submit your results to a journal. The editor asks for replication code and data, so you simply share your R or Python scripts and grant temporary access to your amcat server for a researcher tasked with replicating your results.

After publication, a newspaper picks up your results, leading some interested citizens to play with your dashboard. Even though users do not have access to the full text for copyright reasons, they can query different combinations of keywords, which makes your research transparent for a wider audience. Since the annotations and the preprocessed texts are also available, someone finds they get even better validation scores using a newly created algorithm.

1.1 What is amcat?

The `amcat`-suite consists of several packages for text analysis. It has two main goals: to **standardize** text analysis tasks with **easy to use** software, while offering quality-of-life features for power users. It consists of several different software packages, which are usually used together:

- `amcat4` takes care of **document storage**, provides fine-grained **data access control** (e.g., to restrict access to the parts of a dataset which are copyrighted, proprietary or (privacy-)sensitive) and supports fast queries using [Elasticsearch](#)
- `middlecat` provides **authentication methods** to support the fine-grained **data access control** built into `amcat4` (e.g., to make datasets available for which data owners have restricted full text access)
- `amcat4client` offers a user interface, which makes it easy to **query documents from amcat4 via a web interface**, share data with collaborators or the public and present your corpora to stakeholders, the community or the public
- `amcat4apiclient` provides bindings to **manage and query corpora from the Python programming language** via `amcat4`'s REST API
- `amcat4r` provides bindings to **manage and query corpora from the R programming language** via `amcat4`'s REST API

These core packages can be extended by powerful addons which provide additional features:

- `annotinder` which let's you **manually annotate documents** with an appealing web interface (which also looks great on mobile!) and the possibility to deploy it to the web. There is also an [R client](#)!
- `nlpipe` can be used for **advanced document pre-processing and machine learning** tasks. You can't share your full text? How about letting `nlpipe` apply word embeddings on your corpus and share the embedding instead of the full text!

You can use many of these packages individually or you create a full setup, which would look something like this:

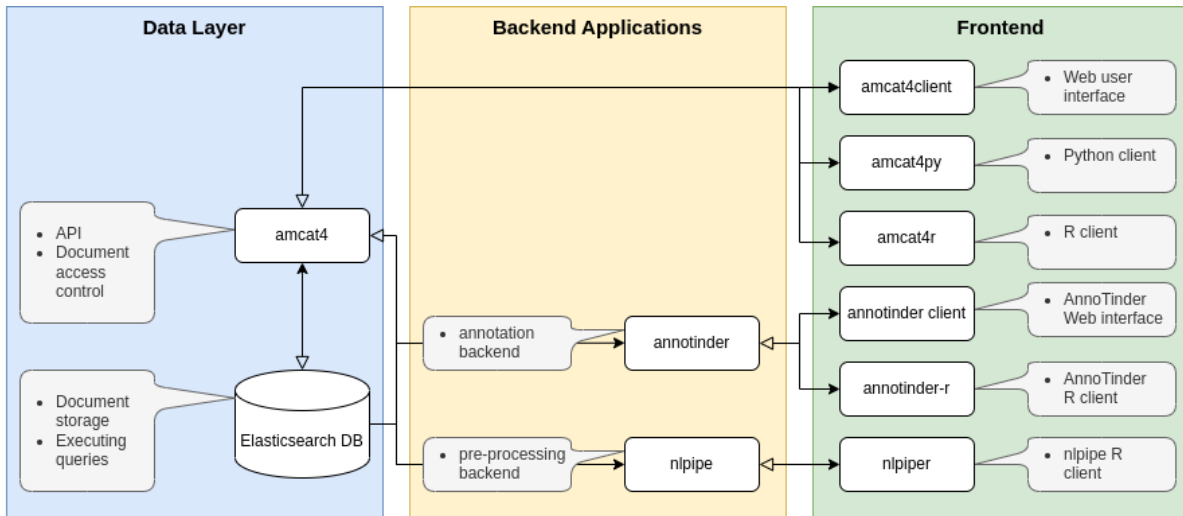


Figure 1.1: amcat-framework

It might seem like this is overly complicated, given that all of the features are also available in other software packages, some of which you will also be familiar with already. However, the main reason that functions are split between different software modules is to make development easier and more transparent. **If you are only interested in amcat's capabilities, you can use it on [our servers](#) or conveniently install everything at once through docker.**

If you want to learn more about the project, have a look at the [about chapter](#).

2 Getting started

! Will change soon

This is work in progress. Passages highlighted in red are likely to change soon.

In this chapter, we show you how to download, set up, and start AmCAT.

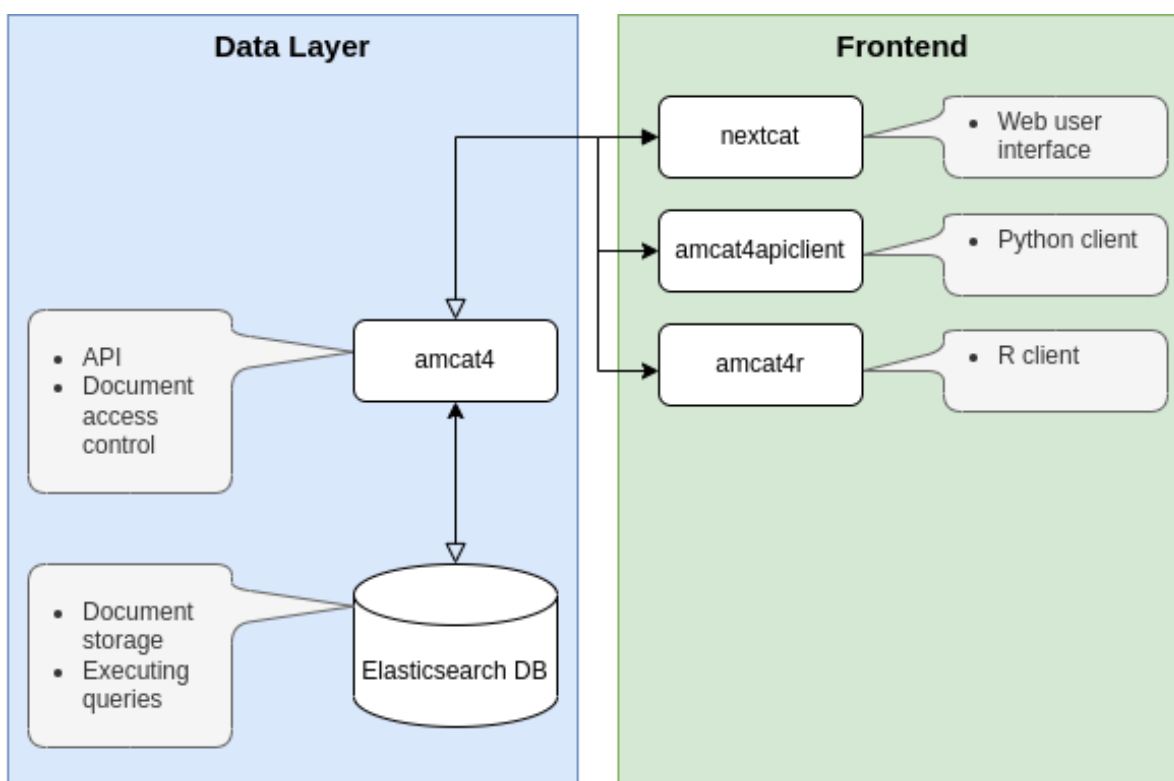


Figure 2.1: AmCAT instance after following this chapter

In the Installing and accessing AmCAT section, it is sufficient if you choose one of the sub-sections to follow. We explain how you can run AmCAT

- [on our servers](#), which we recommend for testing purposes;

- through a Docker image, which we recommend for most people who want to conduct a research project and/or share data online;
- [or install AmCAT directly on your system](#), which we only recommend for advanced users, who want a customised setup.

In the [Frontend](#) section, it makes sense to cover the [amcat4client](#), which provides a react web interface to query data. Then you can select to either install the R or Python client.

2.1 Installing and accessing AmCAT

2.1.1 Run on our servers

! Coming soon...

2.1.2 AmCAT in your personal computer

Let's first install AmCAT in your computer. For this, you need to: 1.Install Docker, 2.Copy the “docker-compose.yml” file in your computer, and 3.Run the file via terminal. It sounds hard, but do not worry, I will visually walk you through each step.

2.1.2.1 Install Docker

💡 Why do we use Docker for installation?

Functionally, Docker containers are a cross-platform installation format that makes it trivially easy to install software packages on Linux, MacOS and Windows without needing users to deal with dependency issues or installation quirks on different systems.¹ A side effect is that we can easily develop AmCAT for all operating systems at once and you can be sure that we do not fall behind on developing AmCAT for your operating system of choice.

Docker is a software that allows easy creation and portability of software in something called containers. AmCAT is stored in one of such containers. To install it, you need to first install Docker: <https://www.docker.com/get-started/>. To install Docker, you can head over to the Docker website to get [Docker Desktop](#) or the [Docker Engine](#) and [Docker Compose](#) to use Docker

¹Technically, it is a little more complicated, as Docker containers have many similarities to virtual machines. However, for most users that technical background is not really important. If you want to learn more, have a look [here](#).

from the command line.² For a visual guide, you can check the video Johannes Gruber (2024) made: <https://www.youtube.com/watch?v=iMyCdd5nP5U>. Remember that to use Docker, you do NOT need to create an account.

Docker Desktop makes installation easier if you do not use a package manager (which many Windows and MacOS users do not). Docker Desktop looks like this:

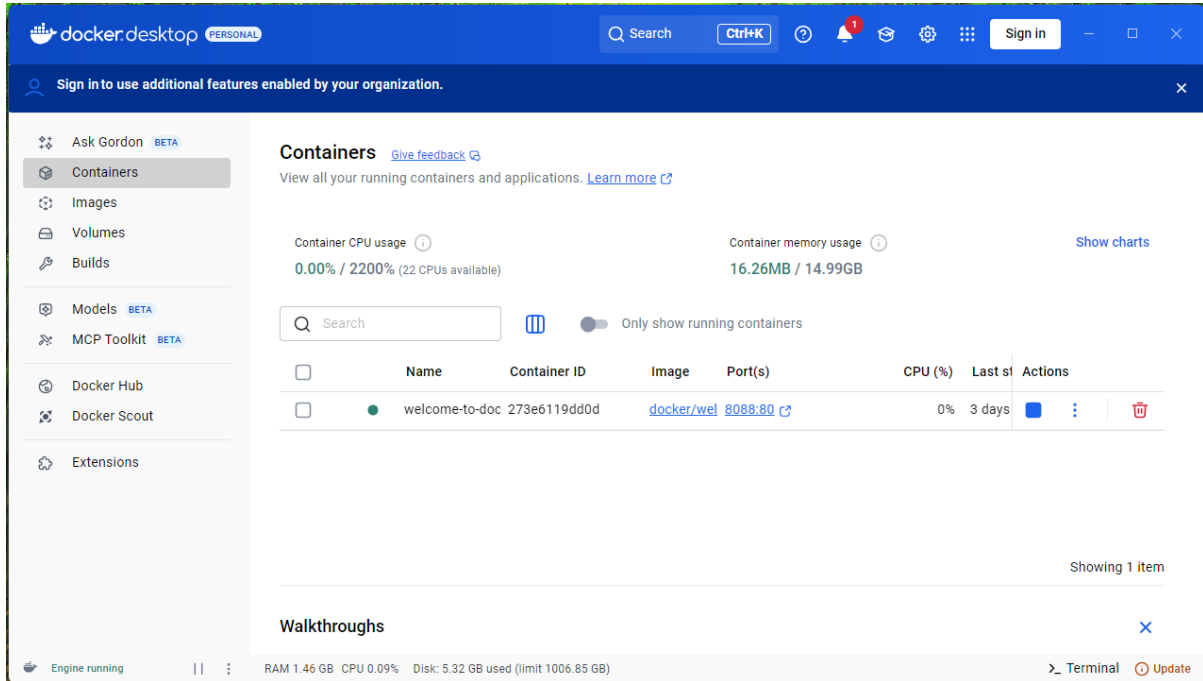


Figure 2.2: Docker_Desktop

2.1.2.2 Copy the docker-compose.yml file in your computer

To install the AmCAT data layer, you should use our Docker Compose file. You can get it from [here](#) (save the file as “docker-compose.yml”).

In general, Gruber (2024) explains how to copy a “docker-compose.yml” file depending on the Operating System (OS). Here, we visually follow the steps for the AmCAT “docker-compose.yml” using a Windows OS. Access the AmCAT “docker-compose.yml” here: <https://raw.githubusercontent.com/ccs-amsterdam/amcat4docker/main/docker-compose.yml>.

²On Ubuntu and Debian, the Docker Compose version in the repos is too old. You can use this gist to update it: <https://gist.github.com/JBGruber/db4de49ee106889b969a277f800c05b8>

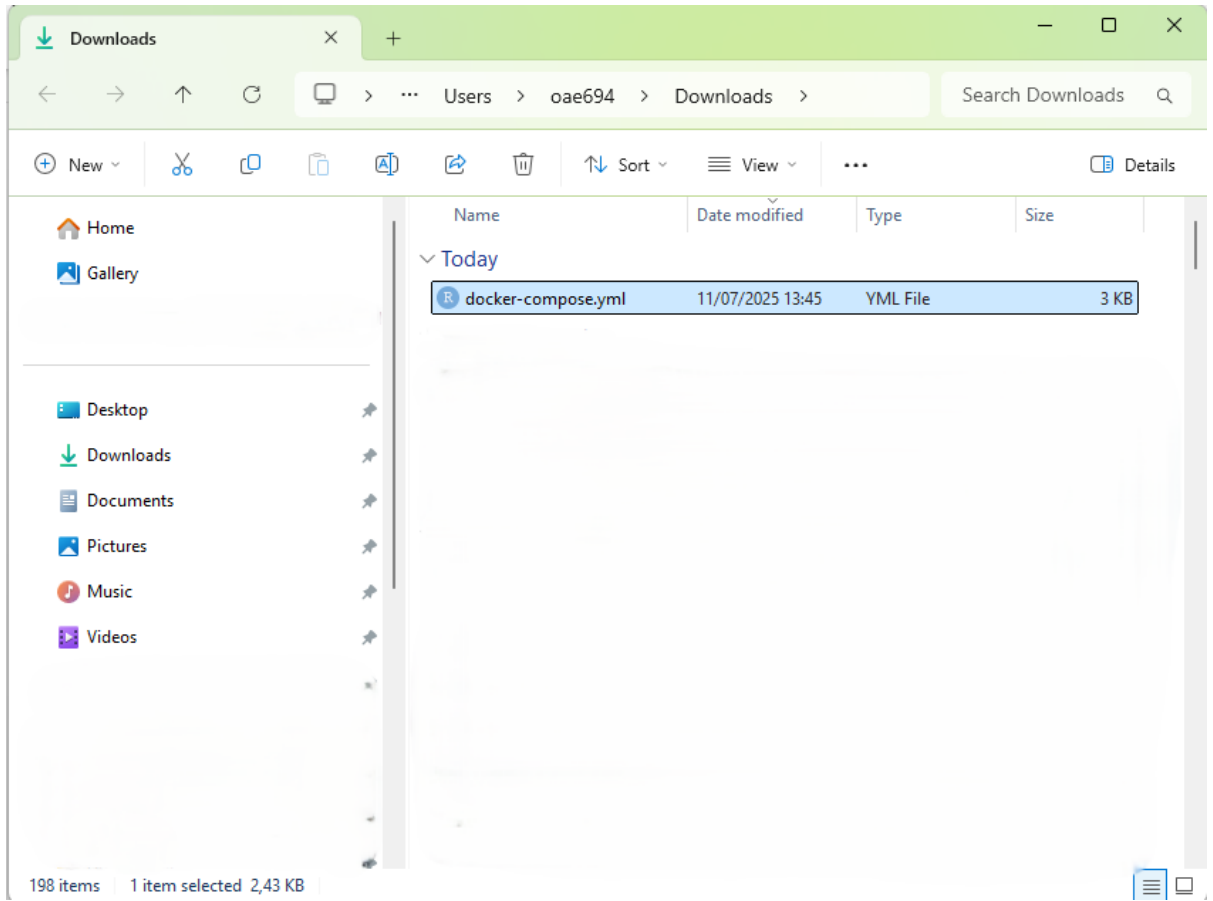


Figure 2.3: AmCAT_DockerCompose1

Now, press the right button of your mouse. That will display the following menu:



```
services:
  web_server:
    image: ccsamsterdam/nginxcat:4.0.14
    build: ./nginxcat
    container_name: nginxcat
    restart: unless-stopped
    networks:
      - amcat-net
    environment:
      - amcat4_client=http://amcat4client:3000
      - amcat4_host=http://amcat4:5000
    ports:
      - 80:80 # [local port]:[container port]
    depends_on:
      - "web_client"
      - "api"
  web_client:
    image: ccsamsterdam/amcat4client:4.0.14
    build: ./amcat4client
    container_name: amcat4client
    restart: unless-stopped
    networks:
      - amcat-net
    environment:
      # this can be changed later, it is just the suggested default
      - NEXT_PUBLIC_AMCAT_SERVER=http://localhost/amcat
    depends_on:
      - "api"
  api:
    image: ccsamsterdam/amcat4:4.0.14
    build: ./amcat4
    container_name: amcat4
    restart: unless-stopped
    networks:
      - amcat-net
    environment:
      # note that these take precedence over values set in `amcat4 config`
      - amcat4_elastic_host=http://elastic8:9200
```

Figure 2.4: AmCAT_DockerCompose2

From there, press the option “Save Page As...” and save it in any location you want:

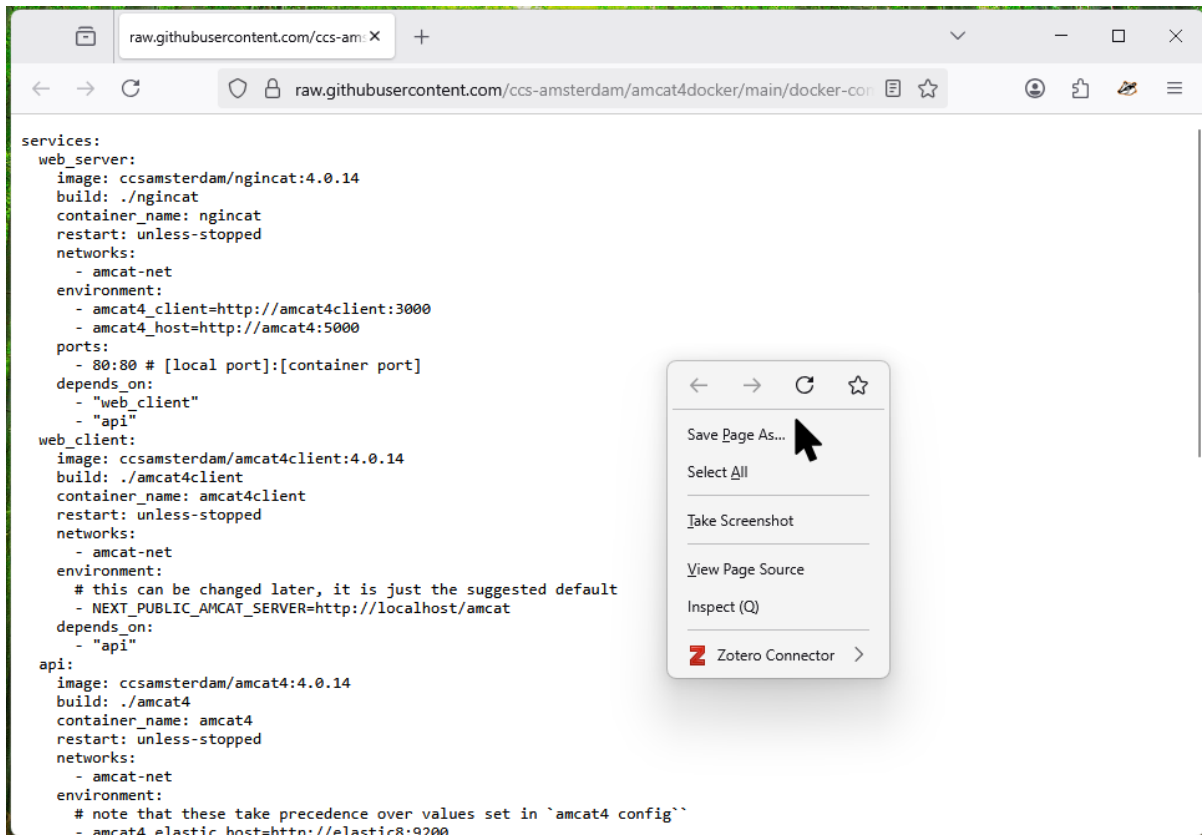


Figure 2.5: AmCAT_DockerCompose3

By default, Windows saves any “weird” file as “txt”:

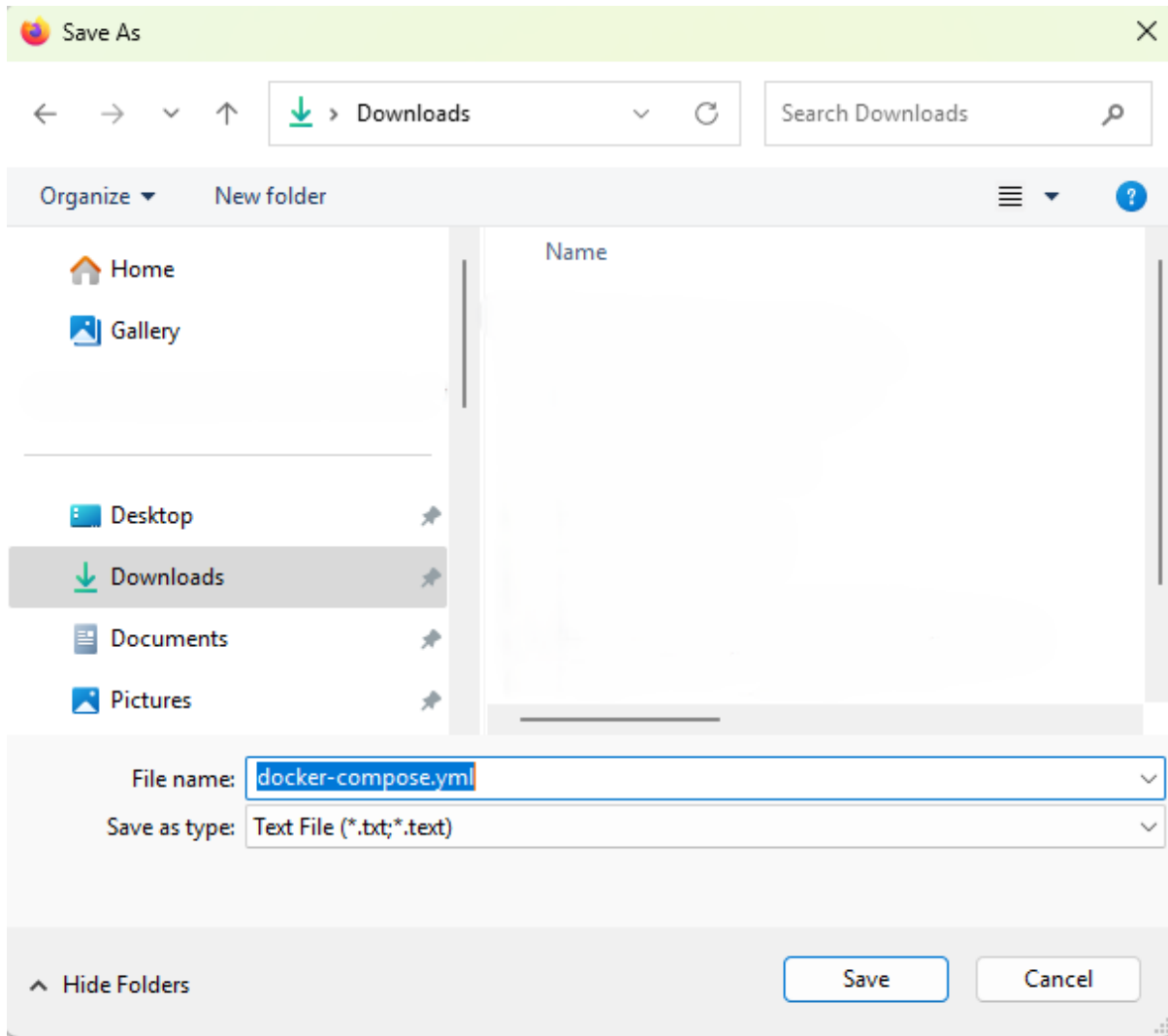


Figure 2.6: AmCAT_DockerCompose4

So, we need to explicitly tell Windows that our file is a “yaml” type. To do so, you need to press View/Show/File name extensions:

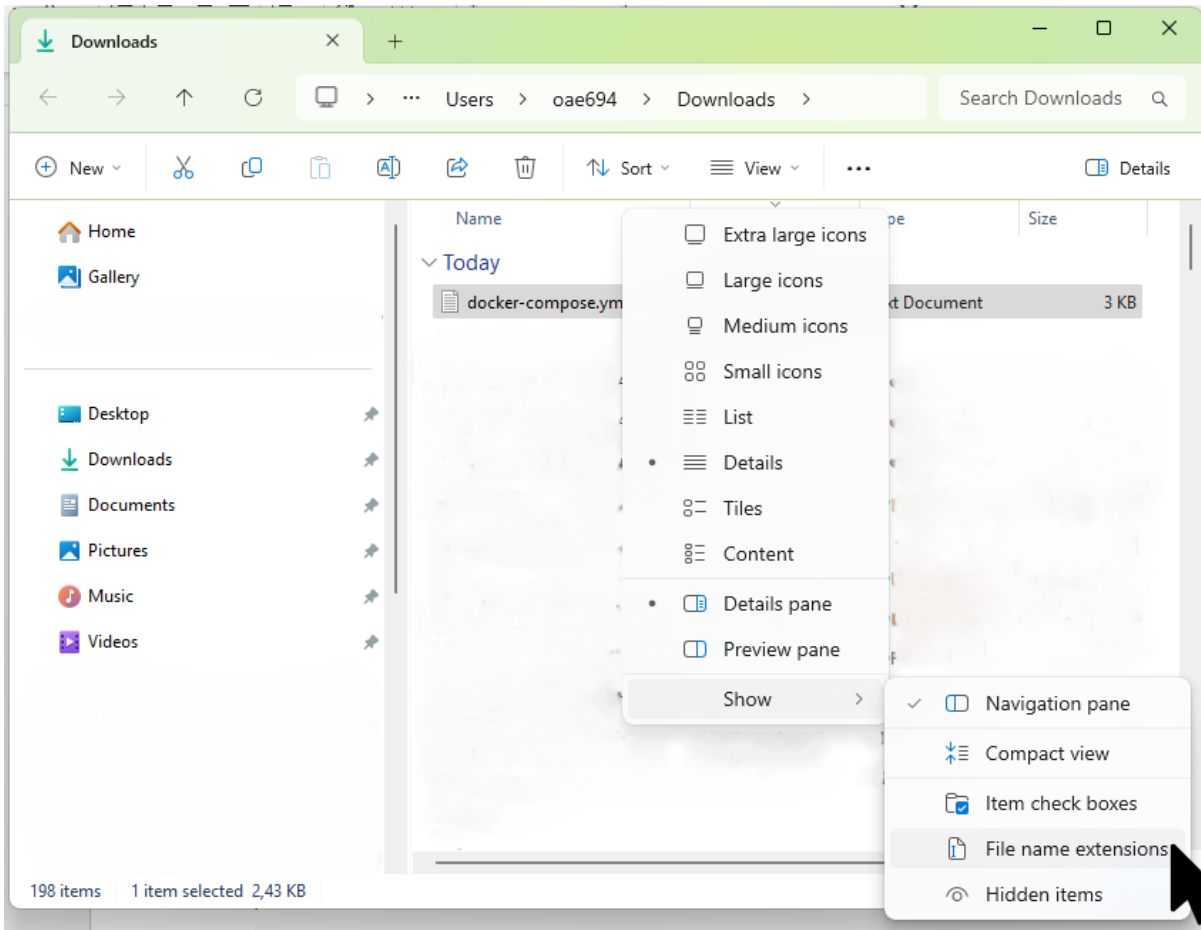


Figure 2.7: AmCAT_DockerCompose5

Now, you can see that Windows added the “txt” extension to the file:

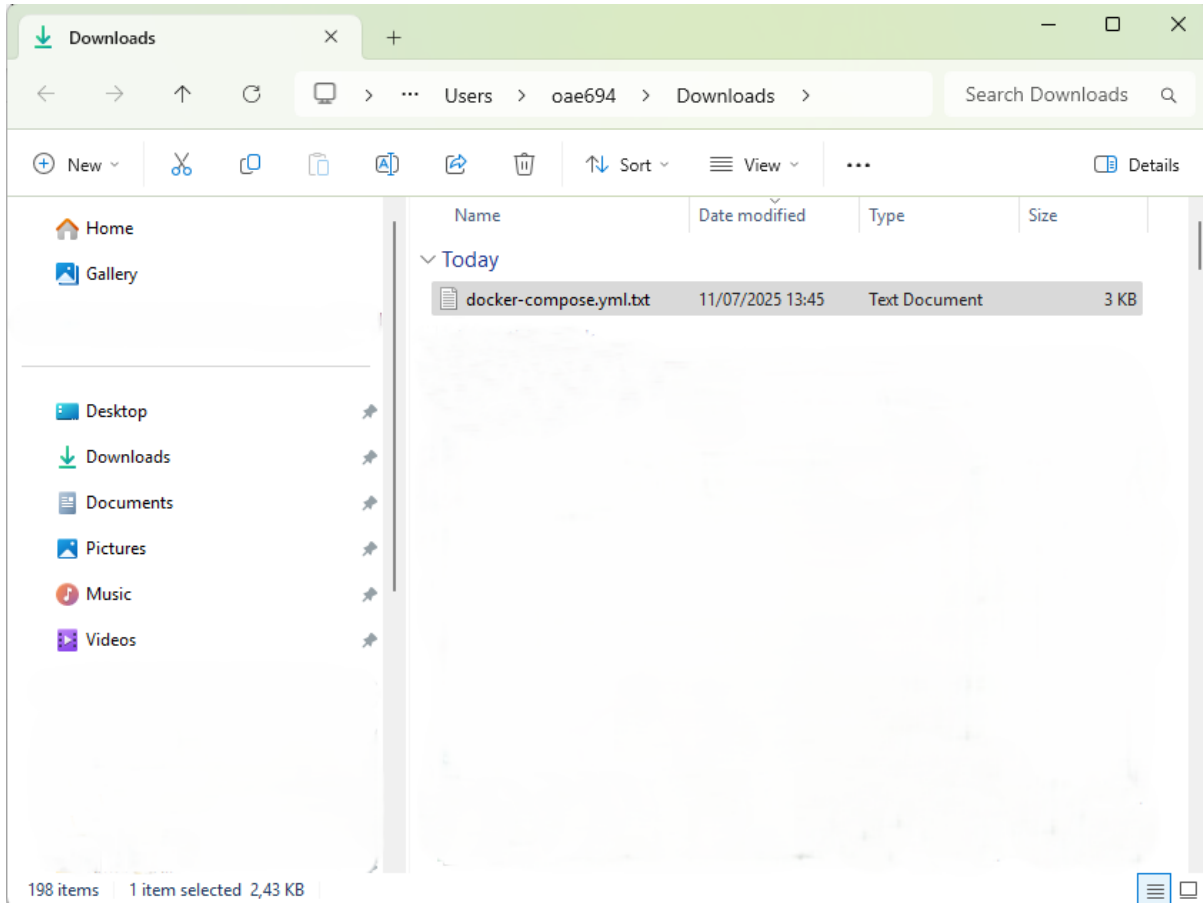


Figure 2.8: AmCAT_DockerCompose6

All we need to do is to delete the extra “.txt”:

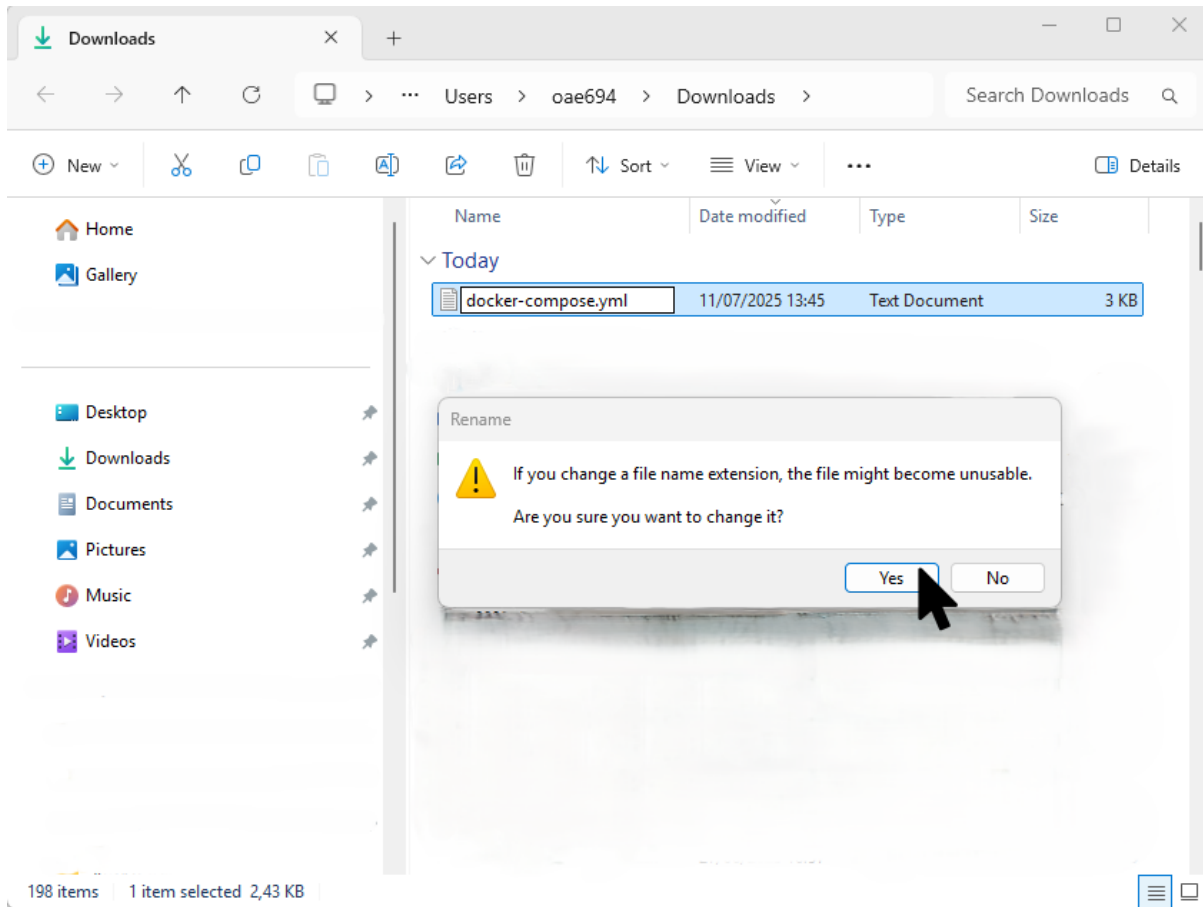


Figure 2.9: AmCAT_DockerCompose7

Now the file type changes to “YML file”.

2.1.2.3 Run the file via terminal

Now, we will run the file via terminal. Gruber (2024) <https://www.youtube.com/watch?v=iMyCdd5nP5U> shows how to do so depending on the OS. Here, we will do it for Windows. So, let's open “Windows PowerShell”:

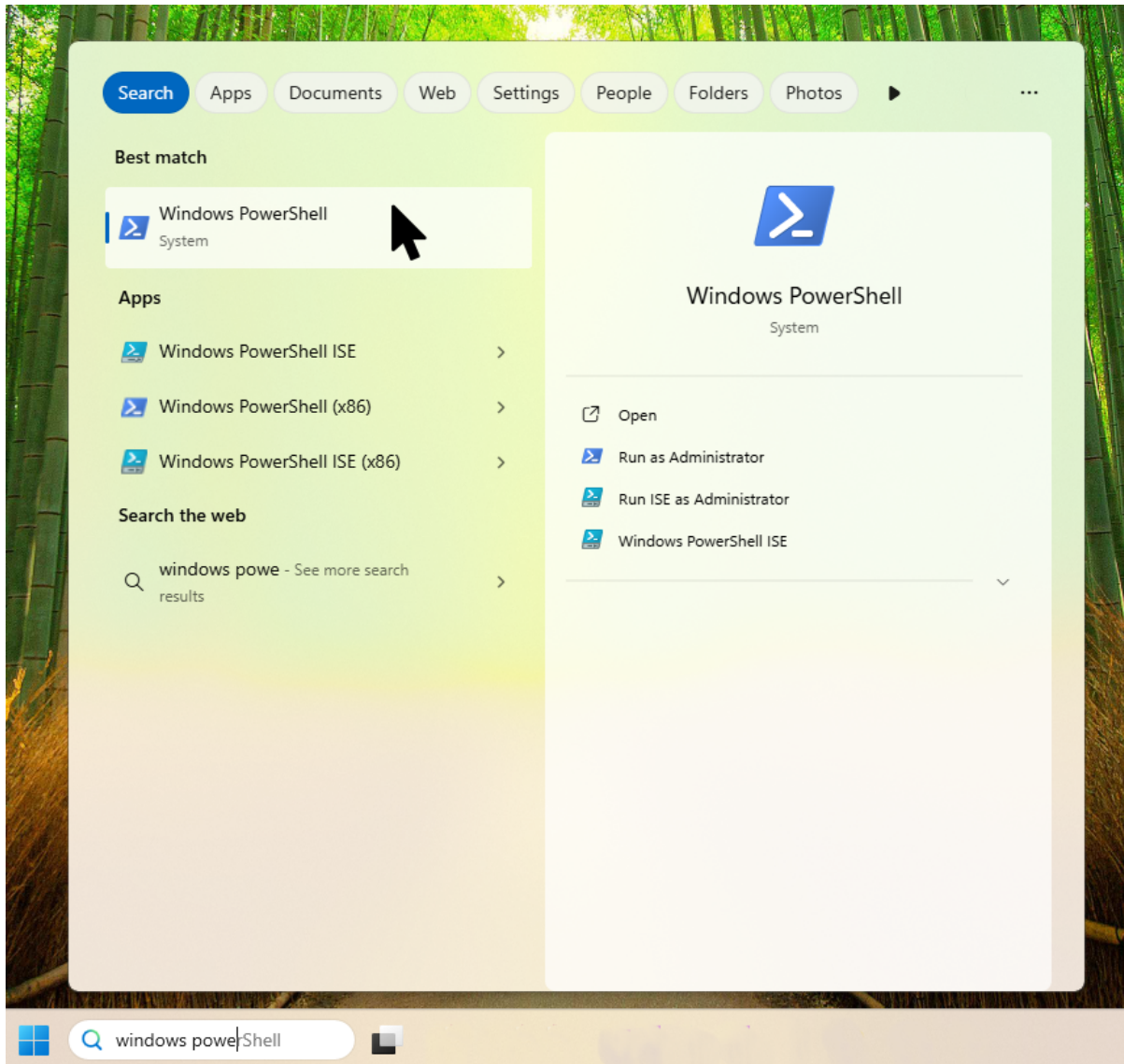


Figure 2.10: Windows Power Shell

This will open the following window:

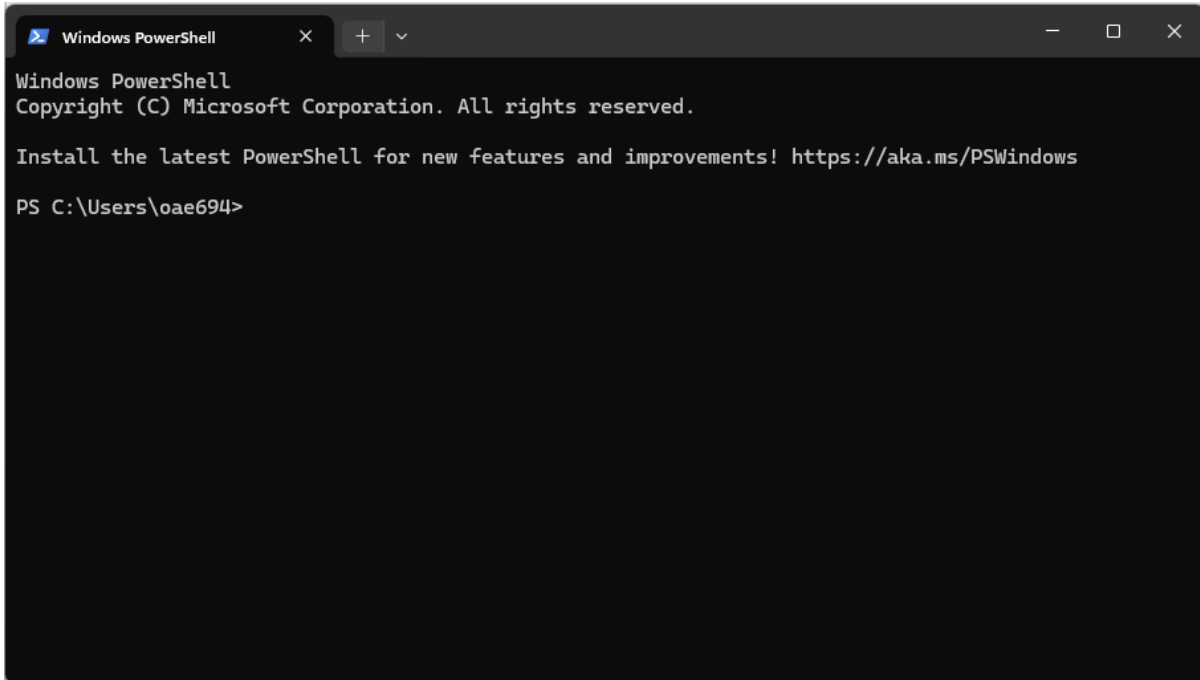
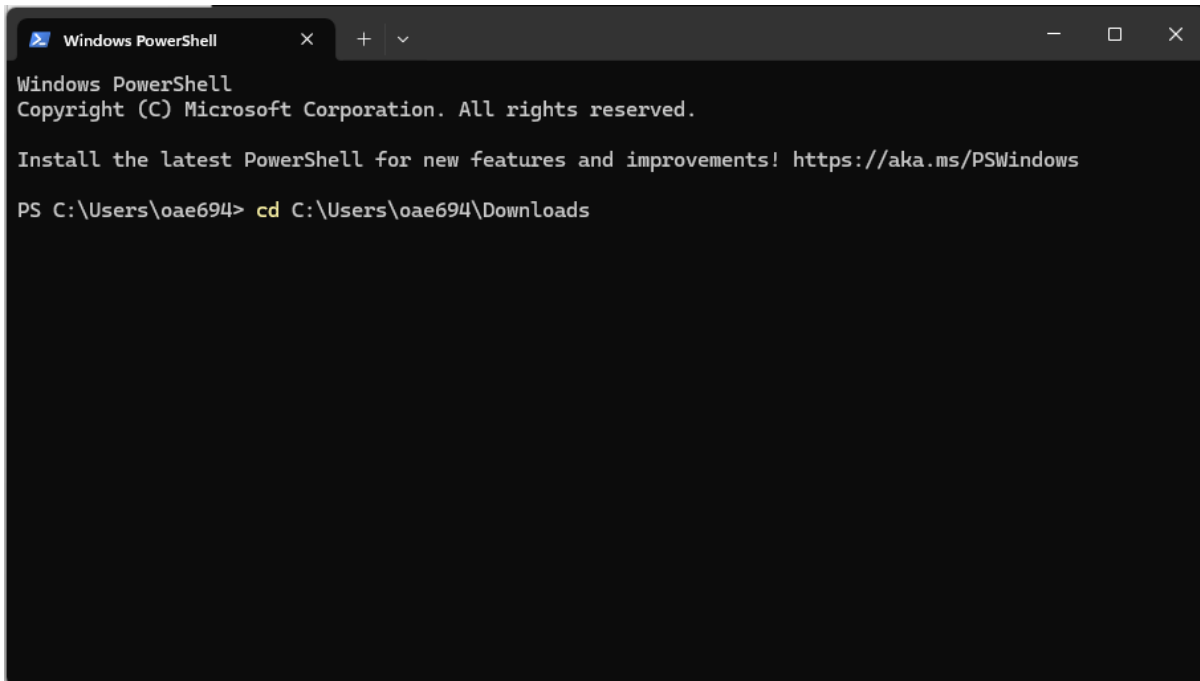


Figure 2.11: Windows Power Shell

In here, we will: (a) move to the folder where the “docker-compose.yml” is, and (b) run the “docker-compose.yml” file.

(a) Move to the folder where the “docker-compose.yml” is

For that, you will use the command `cd` followed by the path where the file is. For example, mine is in the “Download” folder, therefore, my command looks like:



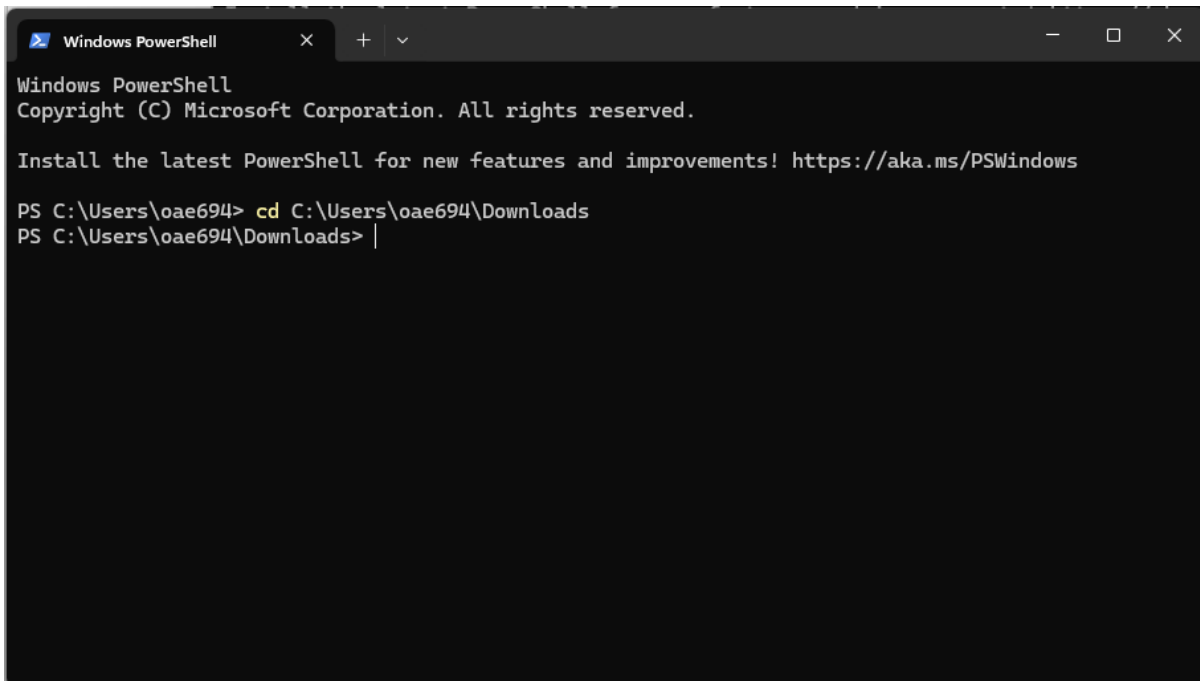
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\oae694> cd C:\Users\oae694\Downloads
```

Figure 2.12: Change Directory

Now press enter. This will modify how the previous link looks like:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

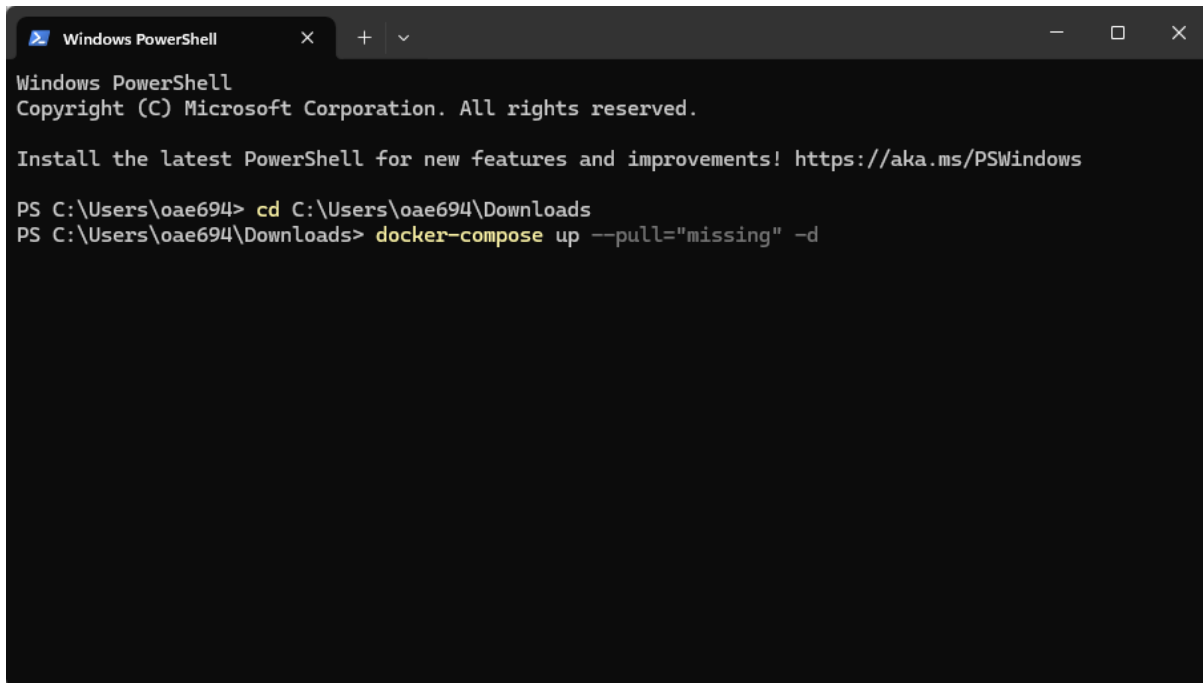
PS C:\Users\oae694> cd C:\Users\oae694\Downloads
PS C:\Users\oae694\Downloads> |
```

Figure 2.13: Change Directory2

(b) Run the “docker-compose.yml” file

Now, you need to run the following command:

```
# docker-compose up --pull="missing" -d
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\oae694> cd C:\Users\oae694\Downloads
PS C:\Users\oae694\Downloads> docker-compose up --pull="missing" -d
```

Figure 2.14: Docker compose

Once it finishes running, it will show the following:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\oae694> cd C:\Users\oae694\Downloads
PS C:\Users\oae694\Downloads> docker-compose up --pull="missing" -d
[+] Running 6/6
  ✓ Network downloads_amcat-net      Created           0.0s
  ✓ Volume "downloads_elastic-volume" Created           0.0s
  ✓ Container elastic8               Started           0.5s
  ✓ Container amcat4                 Started           0.6s
  ✓ Container amcat4client            Started           0.7s
  ✓ Container ngincat                 Started           0.9s
PS C:\Users\oae694\Downloads>
```

Figure 2.15: Docker compose2

If you navigate to Docker Desktop, you will see that AmCAT, together with other dependencies, is now listed as a container under “downloads”:

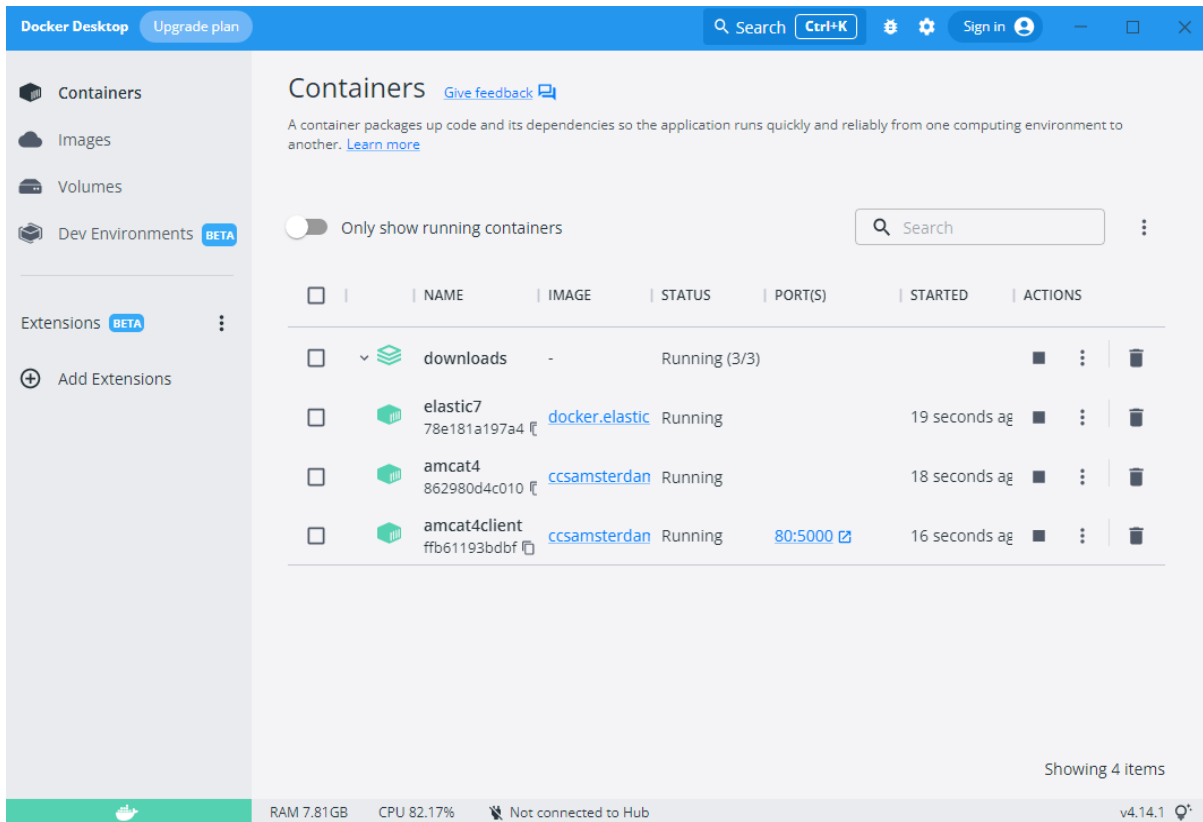


Figure 2.16: Docker Desktop with AmCAT

We tried to write the file to contain sensible defaults for testing it locally. If you plan to work with AmCAT for a research project and/or plan to update the images in the future, you should have a look at the customization option. Otherwise, you can continue.

i Customization

The current default docker-compose.yml looks like this:

```

1 services:
2   web_server:
3     image: ccsamsterdam/nginxcat:4.0.16
4     build: ./nginxcat
5     container_name: nginxcat
6     restart: unless-stopped
7     networks:
8       - amcat-net
9     environment:
10      - amcat4_client=${AMCAT_CLIENT:-http://amcat4client:3000}
11      - amcat4_host=${AMCAT_HOST_INTERNAL:-http://amcat4:5000}
12     ports:
13      - 80:80 # [local port]:[container port]
14     depends_on:
15      - "web_client"
16      - "api"
17   web_client:
18     image: ccsamsterdam/amcat4client:4.0.16
19     build: ./amcat4client
20     container_name: amcat4client
21     restart: unless-stopped
22     networks:
23      - amcat-net
24     environment:
25      - AMCAT_SERVER=${AMCAT_HOST_EXTERNAL:-http://localhost/amcat}
26     depends_on:
27      - "api"
28   api:
29     image: ccsamsterdam/amcat4:4.0.16
30     build: ./amcat4
31     container_name: amcat4
32     restart: unless-stopped
33     networks:
34      - amcat-net
35     environment:
36      # note that these take precedence over values set in `amcat4 config`
37      - amcat4_elastic_host=${ELASTIC_HOST:-http://elastic8:9200}
38      - amcat4_host=${AMCAT_HOST_EXTERNAL:-http://localhost/amcat}
39      - FORWARDED_ALLOW_IPS=*
40     depends_on:
41      - "db"
42   db:
43     image: docker.elastic.co/elasticsearch/elasticsearch:8.18.4
44     container_name: elastic8
45     restart: unless-stopped
46     # for security reasons, the database is only exposed to the other containers in the am
47     # If you want to be able to access it locally, uncomment the following two lines
48     # ports:
49     # - 9200:9200
50     networks:
51      - amcat-net
52     environment:
53      - discovery.type=single-node

```

Setting the containers up from R or Python uses the same settings. You can leave most lines as they are, but we want to draw your attention to a couple of settings you might want to change:

- In lines 12, 13, we set the port to 80 on the host machine. This means you will be able to access the AmCAT client without specifying a port (80 is the default port your browser uses to access an address). If the port is already in use, the container will crash. In this case, change 80 to a different port and access AmCAT through, for example, `localhost:5000`.
- In lines 53, 54, 55, 56, we configured Elasticsearch to form a single-node cluster and use a maximum of 4GB memory.
- In lines 63, 64 we suggested a setting so Elasticsearch will store your data on a volume on the host machine. **If you do not use this, your data will be destroyed when you remove the Elasticsearch container!** We recommend this to make it easier to back up your database and reuse it with a different installation of Elasticsearch (e.g., after an update) in the future. However, the container will not run if it does not have proper access to this folder. See the comment to solve this. Note that the suggested local path is just an example. Learn more about this in the chapter on backups and updates.

Besides the command line, it is possible to download and start the AmCAT containers (Elasticsearch, `amcat4`, and `amcat4client`) from R and python:

2.1.2.4 Command Line

Start a terminal and navigate to the directory where you downloaded the `docker-compose.yml` file³:

```
# docker-compose up --pull="missing" -d
```

Check if the containers are running with:

```
# docker ps
#> CONTAINER ID      IMAGE                                     COMMAND
#> 0628bc852c79      ccsamsterdam/amcat4client:0.0.1        "nginx -g 'daemon o
#> 8134bcd1cbe8      ccsamsterdam/amcat4:0.0.1              "./wait-for-it.sh e
#> 2d59e128e748      docker.elastic.co/elasticsearch/elasticsearch:7.17.7  "/bin/tini -- /usr/
```

³The yml file is written for Docker Compose V2. If you are having trouble, check your version with `docker-compose --version`. [Get the newest version as described here](#).

2.1.2.5 R

```
# install the required packages first:
# remotes::install_github("JBGruber/dockr")
# remotes::install_github("ccs-amsterdam/amcat4r")
# amcat4r::run_amcat_docker()
```

This pulls downloads the default `docker-compose.yml` file from the same link as above. If you want to change the file first, just supply the path to the function afterwards:

```
# amcat4r::run_amcat_docker("docker-compose.yml")
```

Check if the containers are running with:

```
# amcat4r::docker_lc()
#> # A tibble: 3 × 5
#>   name          image          status          id
#>   <chr>         <chr>          <chr>          <chr>
#> 1 /amcat4client ccsamsterdam/amcat4client:0.0.1 Up 58 seconds 0628bc
#> 2 /amcat4       ccsamsterdam/amcat4:0.0.1     Up 59 seconds 8134bc
#> 3 /elastic7    docker.elastic.co/elasticsearch/elasticsearch:7.17.7 Up 58 seconds 2d59e1
```

2.1.2.6 Python

```
# coming soon
```

It might take a couple of seconds for Elasticsearch to start up.

2.2 Access AmCAT

Great! So far, we told you that you have installed AmCAT, but you cannot see it yet... AmCAT can be accessed via its Graphic User Interface (GUI) and via its Application Programming Interface (API).

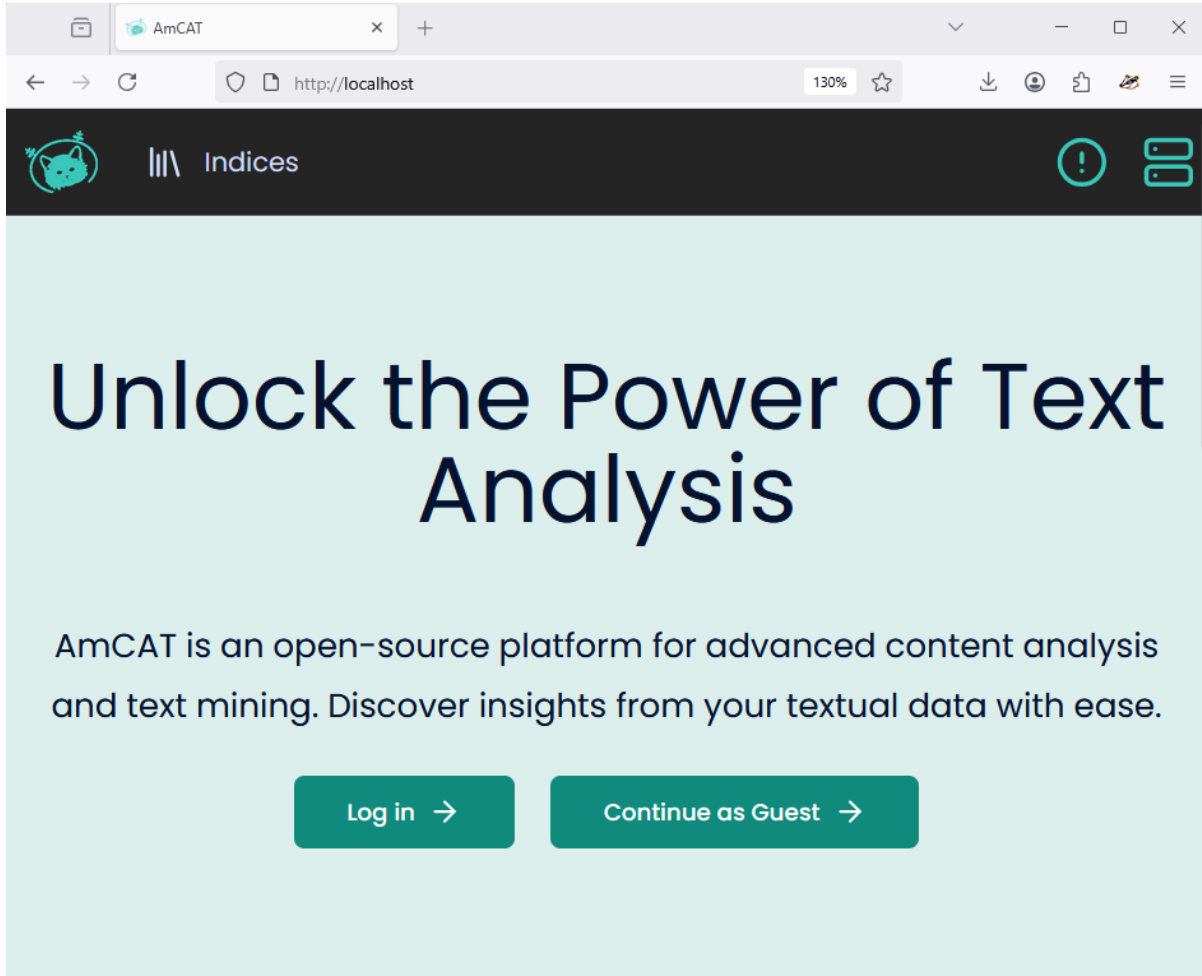
! Important

For AmCAT to work in Windows and MAC, Docker has to be running, i.e. open Docker Desktop.

2.2.1 Accessing the GUI

Open your favorite browser (e.g. Chrome, Firefox, or Edge) and type: <http://localhost/>.

This will open a page that looks like:



Key Features



Figure 2.17: First view of the AmCAT react app in your browser

From there, you should press “Log in” to finish setting up your account, which will open a

window that looks like:

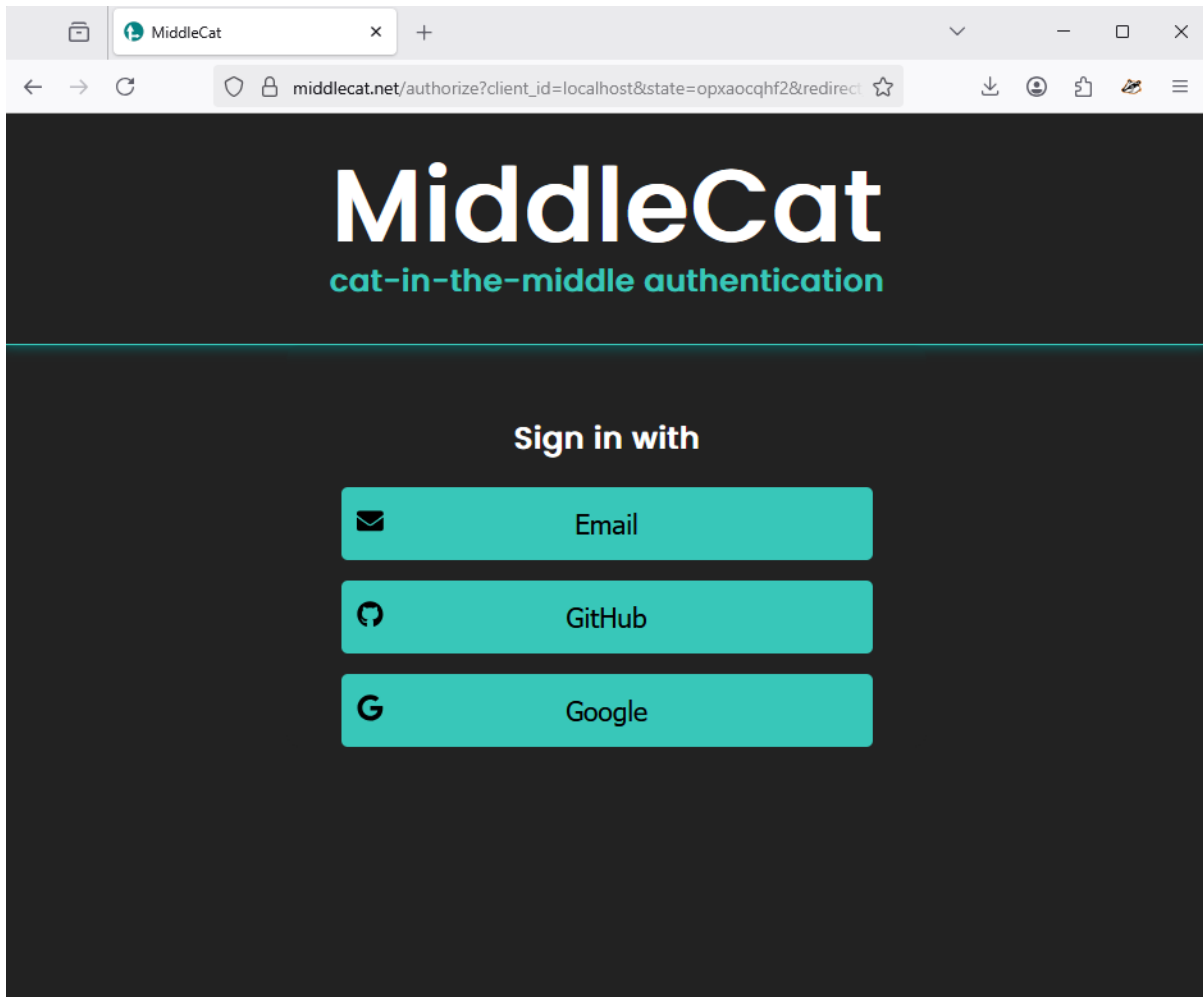


Figure 2.18: AmCAT log in

Now, you can safely log in using your email. This will create a link that will be sent to the email you provided. Just click on it. This will display a window that looks like the one beneath, just press Authorize:

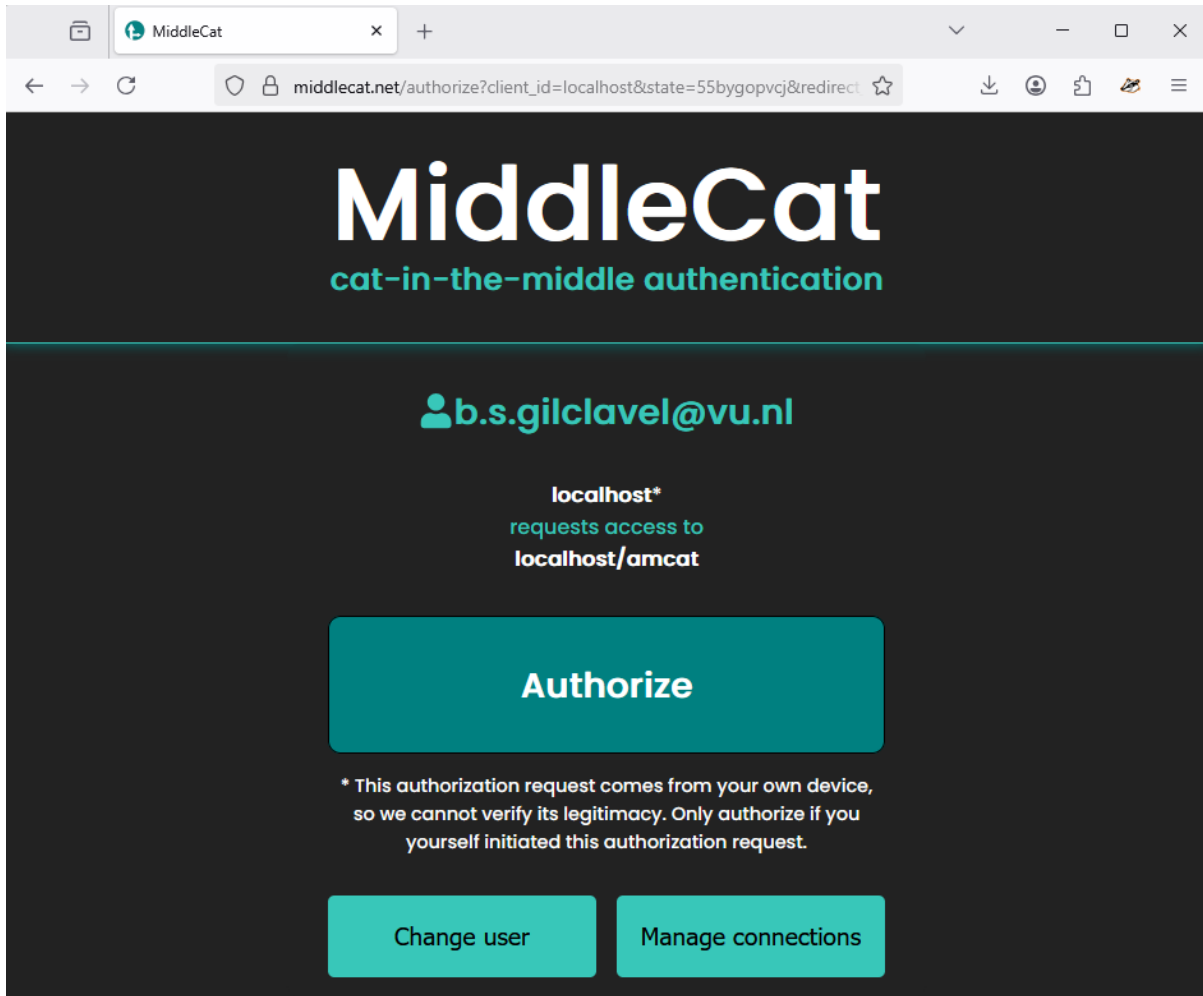


Figure 2.19: AmCAT Authorize

Now, you can just enter the server or check the [authentication options](#):

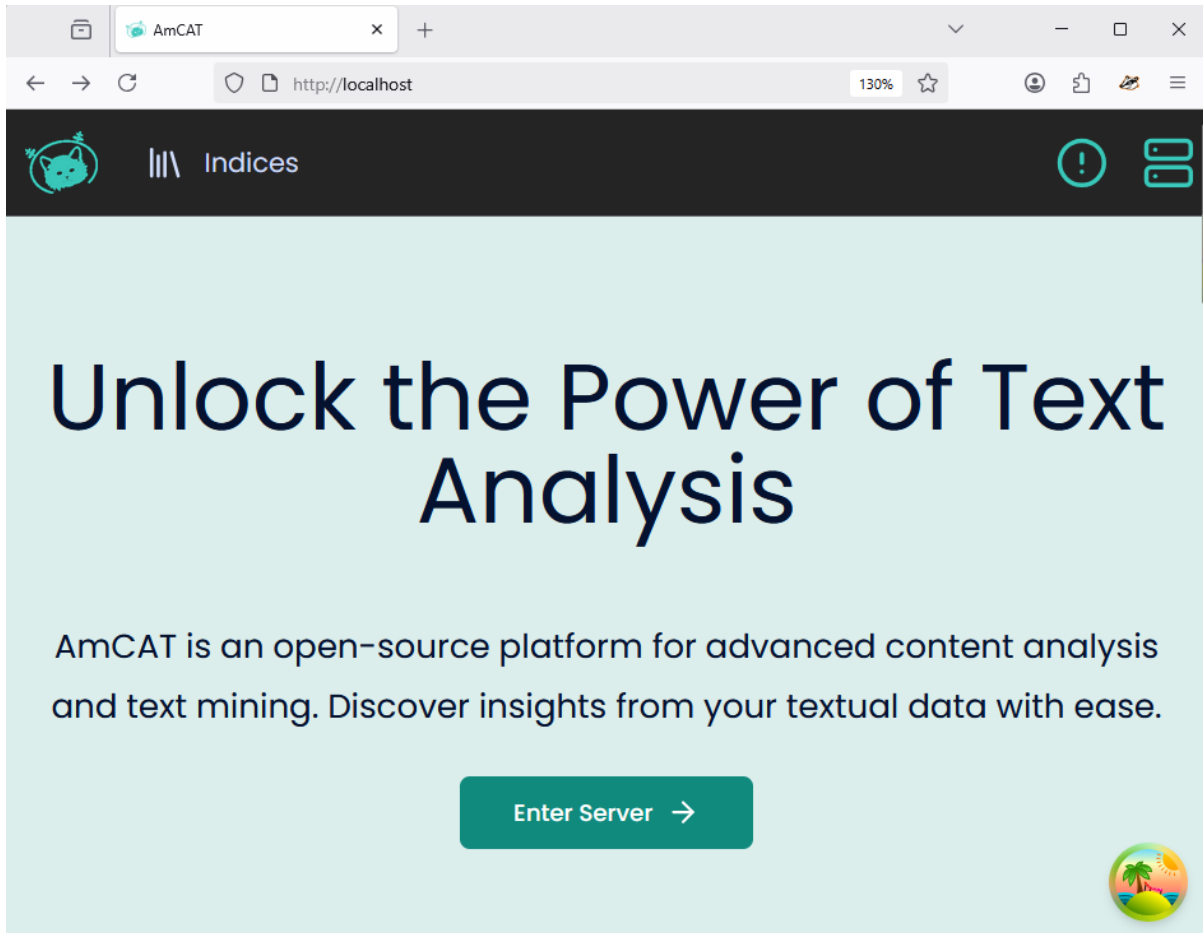


Figure 2.20: AmCAT Authorize

2.2.1.1 Authentication options

If you are thinking on giving access to others, then you should first check the settings and change to your needs. You can do this with `amcat4 config`:

2.2.1.2 Command Line

```
# docker exec -it amcat4 amcat4 config
```

```
~ : fish — Konsole <2>
File Edit View Bookmarks Plugins Settings Help
New Tab Split View Copy Paste Find
johannes@johannes-pc ~
> docker exec -it amcat4 amcat4 config
Reading/writing settings from .env

host: Host this instance is served at (needed for checking tokens)
The current value for host is http://localhost/amcat.
Enter a new value, press [enter] to leave unchanged, or press [control+c] to abort:

elastic_host: Elasticsearch host
The current value for elastic_host is elastic7:9200.
Enter a new value, press [enter] to leave unchanged, or press [control+c] to abort:

auth: Do we require authorization?
Possible choices:
- no_auth: everyone (that can reach the server) can do anything they want
- allow_guests: everyone can use the server, dependent on index-level guest_role authorization settings
- allow_authenticated_guests: everyone can use the server, if they have a valid middlecat login,
and dependent on index-level guest_role authorization settings
- authorized_users_only: only people with a valid middlecat login and an explicit server role can use the server

The current value for auth is AuthOptions.no_auth.
Enter a new value, press [enter] to leave unchanged, or press [control+c] to abort: allow_authenticated_guests

middlecat_url: Middlecat server to trust as ID provider
The current value for middlecat_url is https://middlecat.up.railway.app.
Enter a new value, press [enter] to leave unchanged, or press [control+c] to abort:

admin_email: Email address for a hardcoded admin email (useful for setup and recovery)
The current value for admin_email is None.
Enter a new value, press [enter] to leave unchanged, or press [control+c] to abort: JohannesB.Grubler@gmail.com

system_index: Elasticsearch index to store authorization information in
The current value for system_index is amcat4_system.
Enter a new value, press [enter] to leave unchanged, or press [control+c] to abort:
*** Written .env file to .env ***
johannes@johannes-pc ~
>
```

Figure 2.21: Configuring amcat4

If you choose anything but `no_auth` for authentication options, you should also add a global admin user via:

```
# docker exec -t amcat4 amcat4 add-admin admin@example.com
```

2.2.1.3 R

```
# coming soon
```

2.2.1.4 Python

```
# coming soon
```

You can now use Middlecat to authenticate as the admin user:

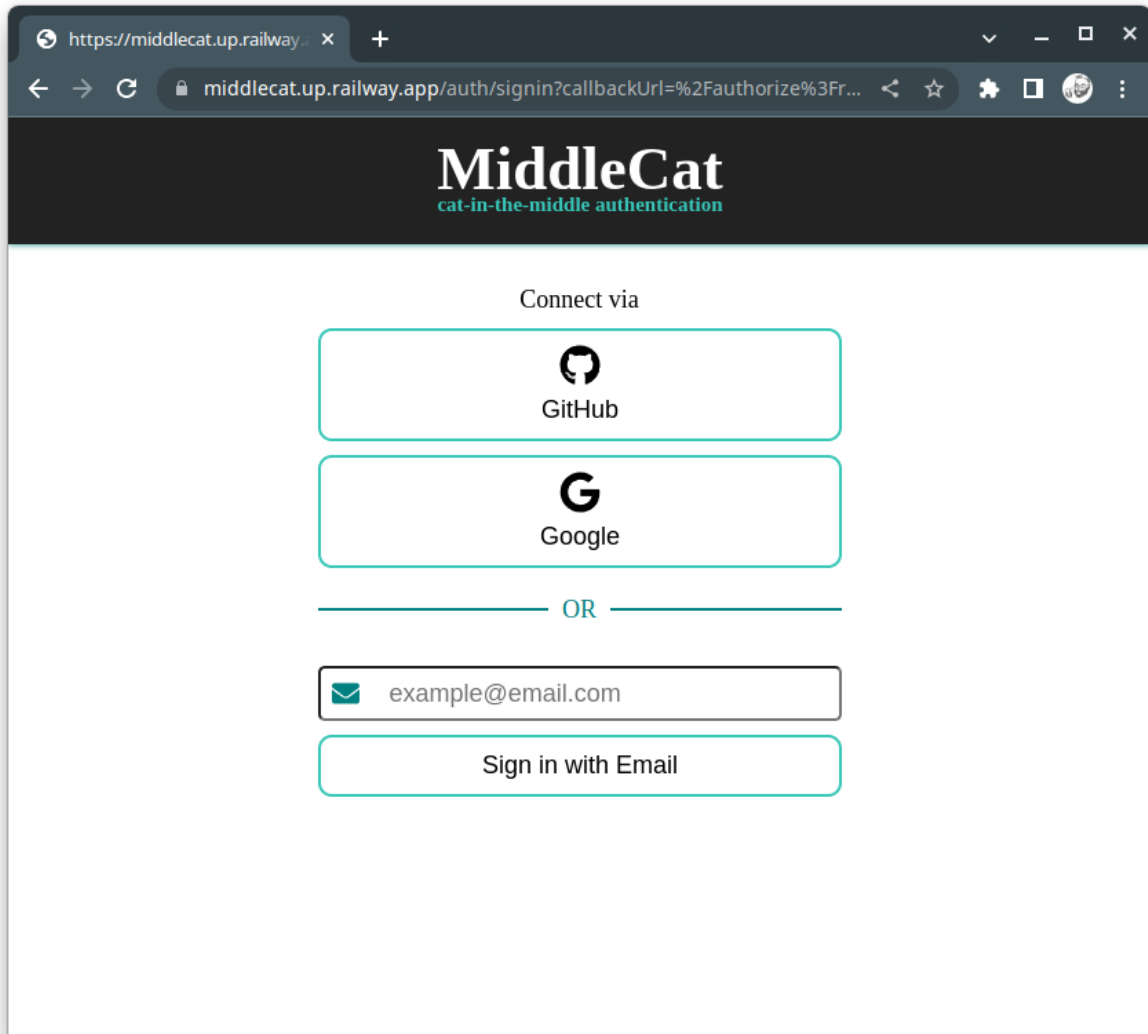


Figure 2.22: After logging into the AmCAT react app in your browser

2.3 Setup on your own server

If you decide not to go with Docker, for example, because you feel you need more control over what is happening, you can also run AmCAT on your system directly. We do not recommend this anymore and have, in fact, switched our own servers over to use the Docker images. If there is something wrong with the images or you simply want to customise the setup, we suggest you head over to the [GitHub repo](#) and change the files as you like.

If you still want to go without Docker, feel free to use the example configuration below. We assume that if you are going this route, you are running a Linux server. Below we show one example setup. Obviously feel free to replace the suggested Linux tools like *systemd* or *nginx* with your own choice.

2.3.0.1 amcat4 – aka AmCAT server

The first piece to set up is the AmCAT server and the Elasticsearch database it interacts with. To download and install Elasticsearch, refer to [their website](#), or, preferably, install it through a package manager. For example, if you are running Debian or Ubuntu or another distro which uses *apt* you can install Elasticsearch 7.x (which we are currently working with) like this:

```
# curl -fsSL https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo gpg --dearmor -o /usr/share/keyrings/elasticsearch.gpg
# echo "deb [signed-by=/usr/share/keyrings/elasticsearch.gpg] https://artifacts.elastic.co/packages/7.x/apt/debian" | sudo tee /etc/apt/sources.list.d/elasticsearch-7.x.list >/dev/null
# sudo apt update
# sudo apt install elasticsearch
```

In the next step, you need to configure Elasticsearch:

```
# sudo nano /etc/elasticsearch/elasticsearch.yml
```

Configure the database to your own liking in terms of user management and exposure. Since we are controlling it through *amcat4*, the only two things that really matter is the address and port (and that *amcat4* still has access after you're done configuring Elasticsearch). So within *elasticsearch.yml*, we only look for two lines:

```
# network.host: localhost
#...
# http.port: 9200
```

You can configure the memory usage of Elasticsearch

```
# echo "-Xms4g" | sudo tee -a /etc/elasticsearch/jvm.options.d/memory.options
```

Leaving the values at their defaults here, we can enable the *systemd* service (skip this step if you've installed Elasticsearch through Docker):

```
# sudo systemctl daemon-reload
# sudo systemctl enable elasticsearch
# sudo systemctl start elasticsearch
```

You can check if everything is working with:

```
# curl -X GET 'http://localhost:9200'
#> {
#>   "name" : "amcat-opted-trekdop0",
#>   "cluster_name" : "amcat-opted",
#>   "cluster_uuid" : "Sx-D89zmSx2zAcwl62u32A",
#>   "version" : {
#>     "number" : "7.17.6",
#>     "build_flavor" : "default",
#>     "build_type" : "deb",
#>     "build_hash" : "f65e9d338dc1d07b642e14a27f338990148ee5b6",
#>     "build_date" : "2022-08-23T11:08:48.893373482Z",
#>     "build_snapshot" : false,
#>     "lucene_version" : "8.11.1",
#>     "minimum_wire_compatibility_version" : "6.8.0",
#>     "minimum_index_compatibility_version" : "6.0.0-beta1"
#>   },
#>   "tagline" : "You Know, for Search"
#> }
```

Next, you want to setup the AmCAT server. You can do this wherever you like, but we will set things up at `/srv/amcat`:

```
sudo git clone https://github.com/ccs-amsterdam/amcat4 /srv/amcat
sudo chown -R $USER:$USER /srv/amcat
cd /srv/amcat
python3 -m venv env
env/bin/pip install -e .[dev]
```

To test if it runs as expected, you can use:

```
env/bin/python -m amcat4 run
#> /srv/amcat/env/lib/python3.9/site-packages/elasticsearch/connection/base.py:200: ElasticsearchWarning
#> warnings.warn(message, category=ElasticsearchWarning)
#> [INFO   :root           ] Starting server at port 5000, debug=True
#> INFO:     Started server process [1001112]
#> [INFO   :uvicorn.error ] Started server process [1001112]
#> INFO:     Waiting for application startup.
#> [INFO   :uvicorn.error ] Waiting for application startup.
#> INFO:     Application startup complete.
#> [INFO   :uvicorn.error ] Application startup complete.
```

```
#> INFO:      Uvicorn running on http://0.0.0.0:5000 (Press CTRL+C to quit)
#> [INFO      :uvicorn.error  ] Uvicorn running on http://0.0.0.0:5000 (Press CTRL+C to quit)
```

To check and adapt the settings of AmCAT use:

```
env/bin/python -m amcat4 config
```

Since you probably don't want to run AmCAT in an open ssh tab all the time, you should set it up as a service, for example, with `systemd`. So head over to `/etc/systemd/system` and create a new file, for example, `amcat.service`.

Here is a small example to set things up:

```
[Unit]
Description=Amcat4 API
After=network.target
Requires=elasticsearch.service

[Service]
Type=simple
User=amcat
Group=amcat
WorkingDirectory=/srv/amcat/amcat4

Environment=AMCAT4_ELASTIC_HOST=http://localhost:9200
Environment=AMCAT4_DB_NAME=/srv/amcat/amcat4.db

ExecStart=/srv/amcat/env/bin/uvicorn \
    --proxy-headers \
    --forwarded-allow-ips='*' \
    --workers=2 \
    --no-access-log \
    --uds /tmp/amcat.socket \
    --root-path /api \
    amcat4.api:app

ExecReload=/bin/kill -HUP ${MAINPID}
RestartSec=1
Restart=always

[Install]
WantedBy=multi-user.target
```

In the above service, we run the AmCAT server as the user `amcat`. To create this user and hand over the ownership of the AmCAT server folder to it use:

```
sudo useradd amcat
sudo chown -R amcat:amcat /srv/amcat
```

Then you can start the service and enable it to run on startup:

```
sudo systemctl daemon-reload
sudo systemctl start amcat.service
sudo systemctl enable amcat.service
```

Now you can check if everything is working with

```
systemctl status amcat.service
amcat.service - Amcat4 API
#>    Loaded: loaded (/etc/systemd/system/amcat.service; enabled; vendor preset: enabled)
#>    Active: active (running) since Thu 2022-11-03 10:39:23 CET; 3min 29s ago
#>   Main PID: 197173 (uvicorn)
#>     Tasks: 4 (limit: 33532)
#>    Memory: 86.8M
#>       CPU: 1.770s
#>   CGroup: /system.slice/amcat.service
#>          197173 /srv/amcat/env/bin/python3 /srv/amcat/env/bin/uvicorn --proxy-header:
#>          197174 /srv/amcat/env/bin/python3 -c from multiprocessing.resource_tracker :
#>          197175 /srv/amcat/env/bin/python3 -c from multiprocessing.spawn import spawn
#>          197176 /srv/amcat/env/bin/python3 -c from multiprocessing.spawn import spawn
```

If something went wrong, you can troubleshoot with `sudo journalctl -eu amcat.service`.

2.4 Frontend

2.4.1 amcat4client

i Note

If you are using AmCAT on our servers or through Docker, you can skip this section and move on to install an API client to start managing AmCAT from either the R or Python.

If you have checked port 5000 of your new AmCAT server while testing it above (i.e., `http://0.0.0.0:5000`), you were probably disappointed by a simple `{"detail": "Not Found"}` message. This is because the client has been split from the main package to make it easier to develop. You can install the React client next to AmCAT in `srv/amcat4client` using:

```
cd /srv/  
sudo git clone https://github.com/ccs-amsterdam/amcat4client.git  
sudo chown -R $USER: amcat4client  
cd amcat4client  
npm install
```

If you get error messages about outdated versions of dependencies (which is likely on Ubuntu and Debian) you should update `Node.js`. On Debian, you can do this likes so:

```
su  
curl -fsSL https://deb.nodesource.com/setup_19.x | bash - &&\  
apt-get install -y nodejs  
exit
```

And the equivalent on Ubuntu:

```
curl -fsSL https://deb.nodesource.com/setup_19.x | sudo -E bash - &&\  
sudo apt-get install -y nodejs
```

See [this repository](#) for instructions for other Linux flavours.

After that, you can build the React app:

```
npm run build
```

If your AmCAT instance will be publicly reachable, you can build the React app permanently attached to only your instance of AmCAT:

```
REACT_APP_FIXED_HOST=https://example.com/api npm run build
```

Once this has finished, you should hand over ownership of the React application to the previously created AmCAT user

```
sudo chown -R amcat:amcat .
```

Now we have an Elasticsearch and amcat4 running. But they are currently not accessible. To solve this, we can use, for example, `nginx` to provide users access to the React frontend and the AmCAT API. Create a new `nginx` config file with, for example, `nano`:

```
sudo nano /etc/nginx/sites-available/amcat.conf
```

Below is a minimal example of the `amcat.conf` file, which you can copy and paste: For more information, visit the [uvicorn documentation website](#).

```
server {
    client_max_body_size 4G;


    listen 5000;

    location /api/ {
        rewrite ^/api/(.*) /$1 break;
        proxy_set_header Host $http_host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection $connection_upgrade;
        proxy_redirect off;
        proxy_buffering off;
        proxy_pass http://amcat;
    }

    location / {
        root /srv/amcat4client/build;
        index index.html;
        try_files $uri $uri/ /index.html;
    }
}

map $http_upgrade $connection_upgrade {
    default upgrade;
    '' close;
}
```

```
upstream amcat {  
    server unix:/tmp/amcat.socket;  
}
```

 warning about http

This setup assumes that your AmCAT sever will only be available in the local network. If it should be accesible via the internet, we **strongly** recommend to enable https. You can find more information about that on the [nginx website](#) or [this guide](#).

To enable the site, use:

```
sudo ln -s /etc/nginx/sites-available/amcat.conf /etc/nginx/sites-enabled/amcat.conf
```

Then simply restart `nginx`, for example, through `systemd`:

```
sudo systemctl restart nginx.service
```

To test if the API is reachable, use this:

```
curl http://localhost:5000/api/  
#> {"detail":"Not Found"}
```

If everything works, you can now access the client at <http://localhost:5000>, or the address of your server, if you installed AmCAT remotely:

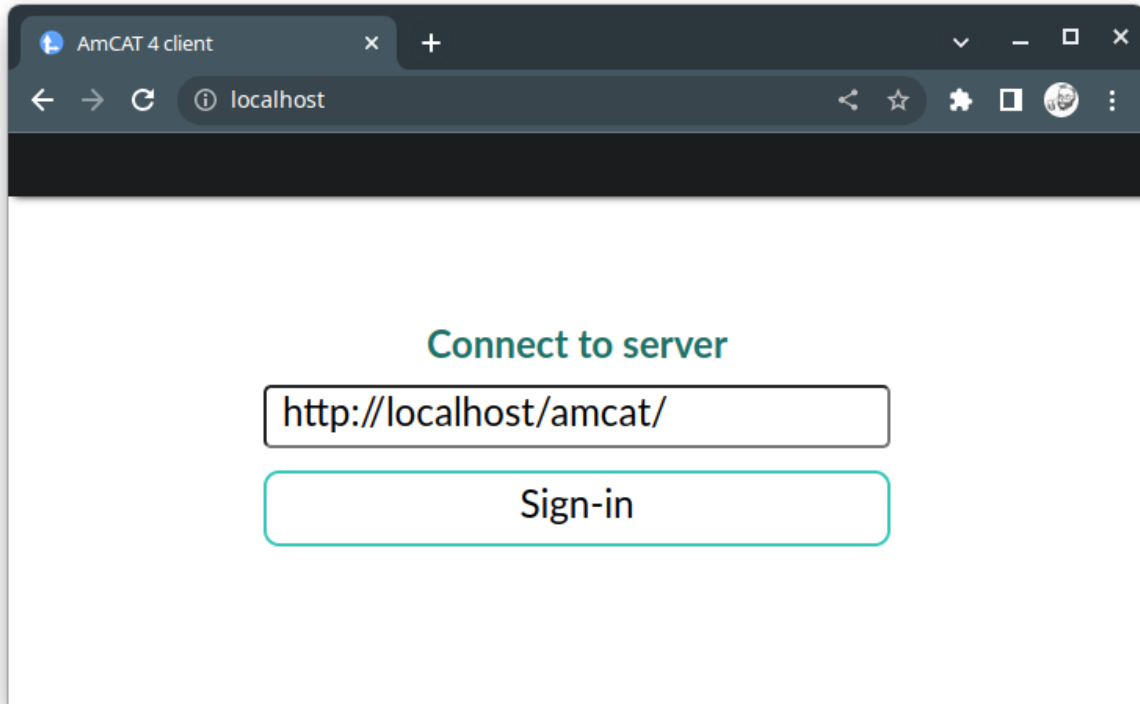


Figure 2.23: First view of the AmCAT react app in your browser

The React app is always running locally in your browser, even if you've accessed it on another computer. So the appropriate host needs to be the route to the AmCAT server. In the example above, I set up an AmCAT instance in my local network on a computer with the IP address 192.168.2.180 and port 5000. To access that host, you need to enter:

i Note

Host: "http://192.168.2.180:5000/api"
Email: "admin"
Password: "admin"

Just replace 192.168.2.180 with the address of the machine you set up AmCAT on.

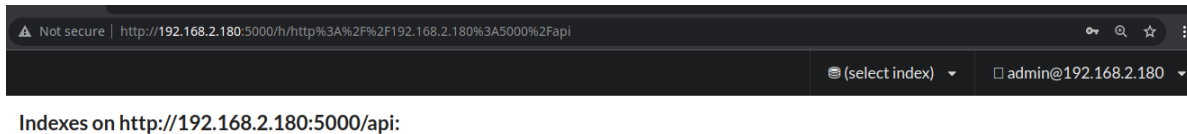


Figure 2.24: After logging into the AmCAT react app in your browser

Success! However, the interface doesn't show much at this point, since we added no data yet. We will do that in the [storage chapter](#).

2.4.2 API Client

2.4.2.1 R

The R client is called `amcat4r` and can be installed via the following command in R (install `remotes` first if you don't have it yet):

```
remotes::install_github("ccs-amsterdam/amcat4r")
```

If you have set up the AmCAT suite as shown above, you should be able to log into the database:

```
library(amcat4r)
login("http://localhost/api", username = "admin", password = "supergeheim")
```

If this does not throw an error, you have set everything up correctly.

2.4.2.2 Python

Install `amcat4py` from the command line through `pip`:

```
pip install amcat4py
```

Then you can open Python and log in:

```
from amcat4py import AmcatClient  
  
amcat = AmcatClient("http://localhost/amcat")
```

If this does not throw an error, you have set everything up correctly.

3 Document Storage

This chapter covers how you can upload, change, query, and delete documents and indexes in AmCAT. Some basic tasks are implemented in the Web user interface, for the rest you need to use one of the clients in R or Python or another means to call the API (e.g., through [cURL](#) as shown below). You can also use the API calls to build your own client.

More information on the API can be found on every amcat4 instance at `/redoc` (e.g., <http://localhost/amcat/redoc>).

Let us know about it and we will promote your new API wrapper package here.

3.1 Manage Documents with the Graphic User Interface

3.1.1 Upload

From the Graphic User Interface (GUI), you can “Create an Index” by uploading it as a csv file.

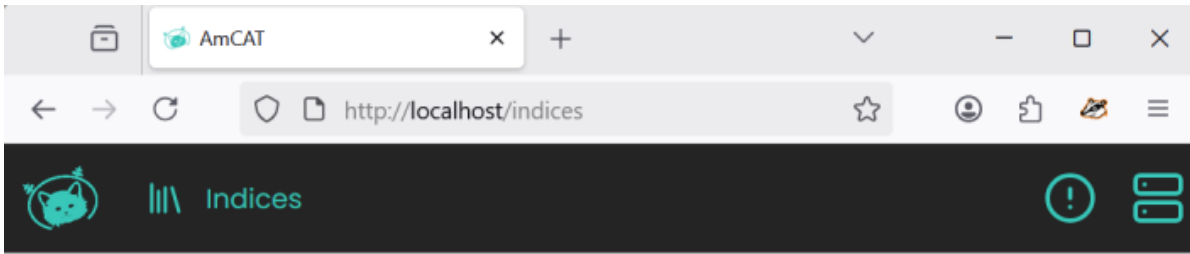
Navigate to your favorite spreadsheet (e.g. Excel) and create something that resembles the following:

	A	B	C	D	E	F
1	author	year	title	chapter	text	
2	Sofia Gil-Clavel	2023	book1	chapter1	"This is some text"	
3	Sofia Gil-Clavel	2023	book1	chapter2	"This is some text"	
4	Sofia Gil-Clavel	2023	book1	chapter3	"This is some text"	
5	Sofia Gil-Clavel	2023	book1	chapter4	"This is some text"	
6	Sofia Gil-Clavel	2023	book1	chapter5	"This is some text"	
7	Sofia Gil-Clavel	2024	book2	chapter1	"This is some text"	
8	Sofia Gil-Clavel	2024	book2	chapter2	"This is some text"	
9	Sofia Gil-Clavel	2024	book2	chapter3	"This is some text"	
10	Sofia Gil-Clavel	2024	book2	chapter4	"This is some text"	
11	Sofia Gil-Clavel	2024	book2	chapter5	"This is some text"	
12	Sofia Gil-Clavel	2025	book3	chapter1	"This is some text"	
13	Sofia Gil-Clavel	2025	book3	chapter2	"This is some text"	
14	Sofia Gil-Clavel	2025	book3	chapter3	"This is some text"	
15	Sofia Gil-Clavel	2025	book3	chapter4	"This is some text"	
16	Sofia Gil-Clavel	2025	book3	chapter5	"This is some text"	
17						

Save it as csv utf-8:

File name:	Example_Index.csv
Save as type:	CSV UTF-8 (Comma delimited) (*.csv)

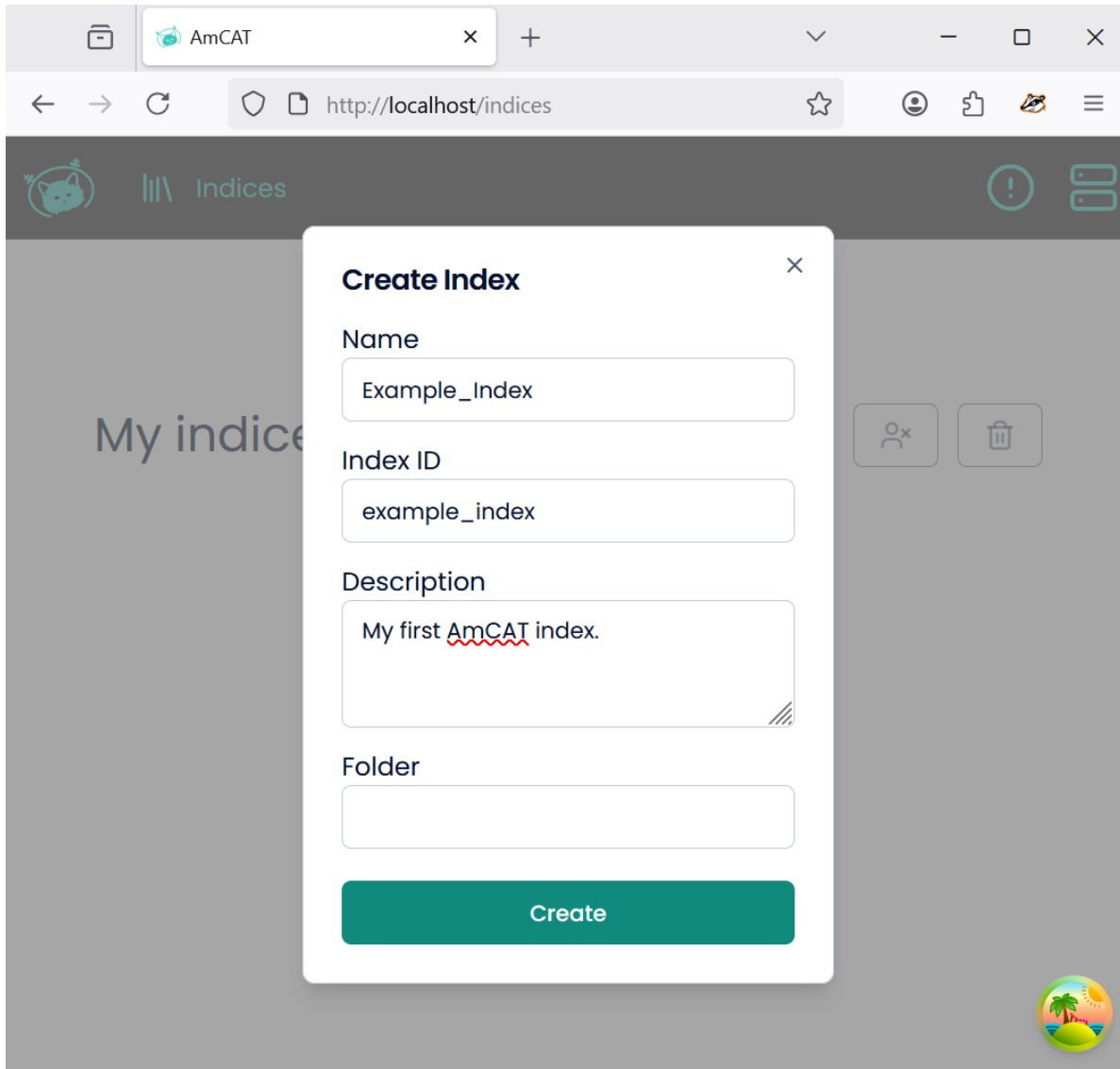
Upload it to AmCAT:



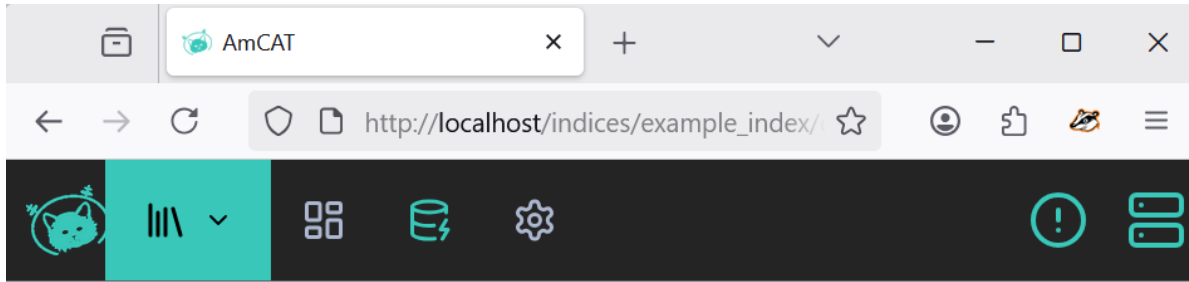
My indices



Fill out each option and press “Create”:



That will display a window with the following information:

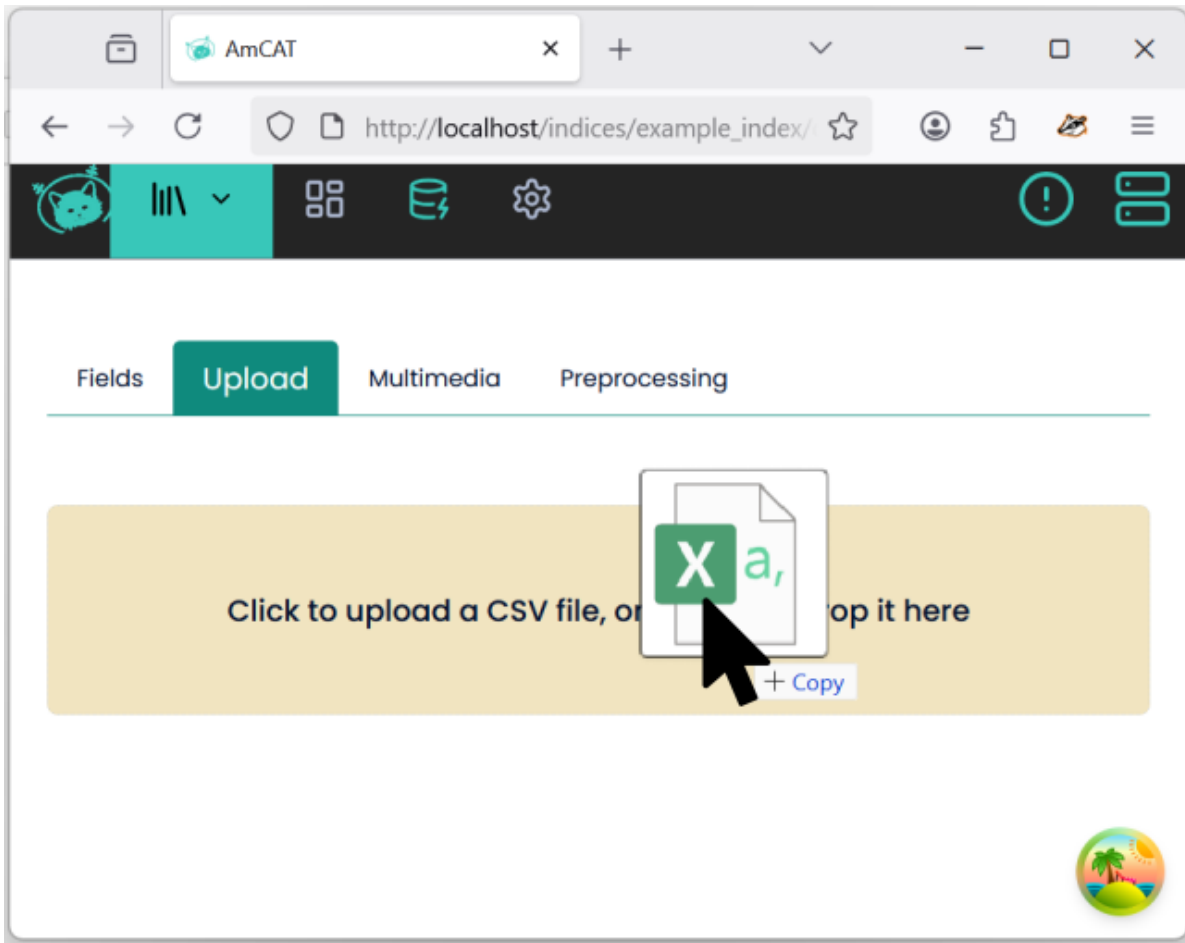


Fields **Upload** Multimedia Preprocessing

Click to upload a CSV file, or drag and drop it here



Now, you can just drag and drop the csv file into the window:









This will display a window with the following information:

Column	Index field	Status
id	Select field ▼ <input type="button" value="automate"/>	<input type="checkbox"/> Not used
author	Select field ▼ <input type="button" value="automate"/>	<input type="checkbox"/> Not used
year	Select field ▼ <input type="button" value="automate"/>	<input type="checkbox"/> Not used
title	Select field ▼ <input type="button" value="automate"/>	<input type="checkbox"/> Not used
chapter	Select field ▼ <input type="button" value="automate"/>	<input type="checkbox"/> Not used
text	Select field ▼ <input type="button" value="automate"/>	<input type="checkbox"/> Not used

Create ▼



From there, you can either choose the type of field by clicking “Select field” or let AmCAT do it for you by clicking “automate”. If in doubt, then choose “automate”. Clicking “automate” on all gives the following result:

Column	Index field	Status
id	<input type="radio"/>  id text	<input checked="" type="checkbox"/> Ready
author	<input type="radio"/>  author keyword	<input checked="" type="checkbox"/> Ready
year	<input type="radio"/>  year integer	<input checked="" type="checkbox"/> Ready
title	<input type="radio"/>  title keyword	<input checked="" type="checkbox"/> Ready
chapter	<input type="radio"/>  chapter keyword	<input checked="" type="checkbox"/> Ready
text	<input type="radio"/>  text keyword	<input checked="" type="checkbox"/> Ready

Creating Index fields

id
 author
 year
 title

 chapter
 text








Upload 15 documents Create ▾



You have to explicitly choose the column(s) that will be used to create the unique identifier. In this example, I explicitly created a column named “id”. So, that is the one I chose as identifier:

Column	Index field	Status
id	<input checked="" type="radio"/>  id text <div style="border: 1px solid orange; padding: 2px; display: inline-block; margin-top: 5px;">Should new column be used as identifier?</div>	<input checked="" type="checkbox"/> Ready


Now, you can press “Upload 15 documents”, which is the number of rows in the csv file.


Column	Index field	Status
id	  id text	<input checked="" type="checkbox"/> Ready
author	<input type="radio"/>  author keyword	<input checked="" type="checkbox"/> Ready
year	<input type="radio"/>  year integer	<input checked="" type="checkbox"/> Ready
title	<input type="radio"/>  title keyword	<input checked="" type="checkbox"/> Ready
chapter	<input type="radio"/>  chapter keyword	<input checked="" type="checkbox"/> Ready
text	<input type="radio"/>  text keyword	<input checked="" type="checkbox"/> Ready

Creating Index identifiers

 **id**

Creating Index fields

 **author**

 **year**

 **title**

 **chapter**

 **text**

[Upload 15 documents](#) [Create](#) ▼




Now you can navigate to your indexes and find your newly made index:

My indices

Example_Index ⋮
My first index



3.1.2 Visualize


One of AmCAT features is that it allows you to perform some basic visualizations. To do so in the main window, you first need to enable them.




































For this, you just need to click on “Enable visualizations in field settings”.


The screenshot shows a web application interface with a dark header bar containing several icons: a cat, a bar chart, a grid, a database cylinder, a gear, a warning triangle, and a sidebar menu. Below the header is a search bar with the text "search" and a filter icon. A navigation bar contains buttons for "Summary", "Aggregate", "Copy", "Update", and "Download". The main content area displays "15 articles" and "1 of 3" with navigation arrows. Three article entries are visible, each with a "book1" label and an "id" field. The first entry has "id_1" and contains three field visualizations: "Sofia Gil-Clavel", "chapter1", and "This is some text". The second entry has "id_2" and contains "Sofia Gil-Clavel", "chapter2", and "This is some text". The third entry has "id_3" and contains "Sofia Gil-Clavel", "chapter3", and "This is some text". Each entry also has "book1" and "2023" labels. A callout box with the text "Enable visualizations in field settings" and a mouse cursor points to the field visualizations. A small globe icon is located in the bottom right corner of the interface.

That will open the following window:

Fields Upload Multimedia Preprocessing

Fields  Add field

Identifier	Field	Type	Display	METAREADER access
	author	 keyword keyword	  H 	 invisible 
	chapter	 keyword keyword	  H 	 invisible 
	id	 text text	  H	 invisible 
	text	 keyword keyword	  H 	 invisible 
	title	 keyword keyword	  H 	 invisible 
	year	 integer long	  H 	 invisible 

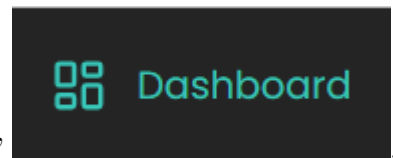


From there, you just need to click on the fields   H  , under the column “Display”, and choose the options “show as heading” and/or “visualize in list”.

Fields ➔ Add field

Identifier	Field	Type	Display	METAREADER access
	author	keyword keyword		invisible ▾
	chapter	keyword keyword		invisible ▾
	id	text text		invisible ▾
	text	keyword keyword		invisible ▾
	title	keyword keyword		invisible ▾
	year	integer integer		invisible ▾

- show in document
- show in list
- show as heading (title)
- visualize in list



You can go back to the main window by clicking on “Dashboard”
There you can see the visualizations you have enabled:

Summary
Aggregate
Copy
Update
Download

15 articles

1 of 3

book1

id_1

Sofia Gil-Clavel
chapter1
"This is some text"
book1

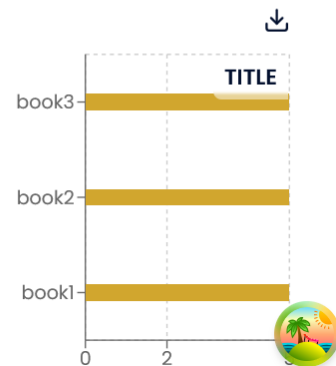
2023

book1

id_2

Sofia Gil-Clavel
chapter2
"This is some text"
book1

2023



3.2 Manage Documents with the API

For this overview, we log into a local `amcat4` (i.e., `http://localhost/amcat`). Replace this with the address to the `amcat4` instance you are working with (e.g., `https://opted.amcat.nl/api`).

We first need to log in:

3.2.0.1 R

```
library(amcat4r)
amcat_login("http://localhost/amcat")
```

3.2.0.2 Python

```
from amcat4py import AmcatClient
amcat = AmcatClient("http://localhost/amcat")
if amcat.login_required():
    amcat.login()
```

3.2.0.3 cURL

There is no dedicated way at the moment to get a token via cURL. You can still use cURL with instances that do not require authentication or by copying the token from Python or R. In these cases, you can make requests with an extra header, for example:

```
AMCAT_TOKEN="YOUR_TOKEN"
curl -s http://localhost/amcat/index/ \
  -H "Authorization: Bearer ${AMCAT_TOKEN}"
```

We can first list all available indexes, as a document collection is called in Elasticsearch and thus in AmCAT:

3.2.0.4 R

```
list_indexes()
```

3.2.0.5 Python

```
amcat.list_indices()
```

3.2.0.6 cURL

```
curl -s http://localhost/amcat/index/
```

You can see that the test index we added in the Data Layer section is here and that it is called “state_of_the_union”. To see everyone who has been granted access to an index we can use:

3.2.0.7 R

```
list_index_users(index = "state_of_the_union")
```

3.2.0.8 Python

```
amcat.list_index_users(index="state_of_the_union")
```

3.2.0.9 cURL

```
curl -s http://localhost/amcat/index/state_of_the_union/users
```

We will learn more about these roles in the chapter on access management. To see what an index looks like, we can query it leaving all fields blank to request all data at once:

3.2.0.10 R

```
sotu <- query_documents(index = "state_of_the_union", queries = NULL, fields = NULL)  
str(sotu)
```

3.2.0.11 Python

```
sotu = list(amcat.query("state_of_the_union", fields=None))
print(len(sotu))
for k, v in sotu[1].items():
    print(k + "(" + str(type(v)) + "): " + str(v)[0:100] + "...")
```

3.2.0.12 cURL

To not clog the output, we save it into file and display only the beginning:

```
curl -s http://localhost/amcat/index/state_of_the_union/documents > sotu.json
# show the first few characters only
head -c 150 sotu.json
```

Knowing now what a document should look like in this index, we can upload a new document to get familiar with the process:

3.2.0.13 R

```
new_doc <- data.frame(
  title = "test",
  text = "test",
  date = as.Date("2022-01-01"),
  president = "test",
  year = "2022",
  party = "test",
  url = "test"
)
upload_documents(index = "state_of_the_union", new_doc)
```

3.2.0.14 Python

```
from datetime import datetime
new_doc = {
  "title": "test",
  "text": "test",
  "date": datetime.strptime("2022-01-01", '%Y-%m-%d'),
  "president": "test",
```

```
"year": "2022",
"party": "test",
"url": "test"
}
amcat.upload_documents("state_of_the_union", [new_doc])
```

3.2.0.15 cURL

```
curl -s -X POST http://localhost/amcat/index/state_of_the_union/documents \
-H "Content-Type: application/json" \
-d '{
    "documents": [
        {
            "title": "test",
            "text": "test",
            "date": "2022-01-01",
            "president": "test",
            "year": "2022",
            "party": "test",
            "url": "test"
        }
    ]
}'
```

Let's see if the the new document is in the index:

3.2.0.16 R

```
query_documents(index = "state_of_the_union", fields = NULL, filters = list(title = "test"))
```

3.2.0.17 Python

```
import pprint
pp = pprint.PrettyPrinter(depth=4)
res=list(amcat.query("state_of_the_union", fields=None, filters={"title": "test"}))
pp.pprint(res)
```

3.2.0.18 cURL

```
curl -s -X POST http://localhost/amcat/index/state_of_the_union/query \  
-H 'Content-Type: application/json' \  
-d '{"filters":{"title":["test"]}}' | jq -r ".results"
```

We will learn more about queries later on in the Writing a Query chapter.

Instead of adding whole documents, you can also change fields in an index. Fields are similar to columns in a table in Excel. However, you need to define the type of a field upon its creation and make sure that you later only add data which adheres to the specifications of the type (otherwise you will get an error). To learn more about the fields in the test index, you can use:

3.2.0.19 R

```
get_fields(index = "state_of_the_union")
```

3.2.0.20 Python

```
amcat.get_fields("state_of_the_union")
```

3.2.0.21 cURL

```
curl -s http://localhost/amcat/index/state_of_the_union/fields
```

You can see that there are five different types in this index: date, keyword, text, url and double. Keyword, text, url are all essentially the same type in R, namely character strings. The date needs to be a POSIXct class, which you can create with `as.Date`. Year should be a double, i.e., a numeric value or integer.

You can add new fields to this, for example, if you want to add a keyword to the documents:

3.2.0.22 R

```
set_fields(index = "state_of_the_union", list(keyword = "keyword"))
```

3.2.0.23 Python

```
amcat.set_fields("state_of_the_union", {"keyword":"keyword"})
```

3.2.0.24 cURL

```
curl -s -X POST http://localhost/amcat/index/state_of_the_union/fields \  
-H 'Content-Type: application/json' \  
-d '{"keyword":"keyword}"'
```

When you now query a document, however, you will not see this new field:

3.2.0.25 R

```
query_documents(index = "state_of_the_union", fields = NULL, filters = list(title = "test"))
```

3.2.0.26 Python

```
res = list(amcat.query("state_of_the_union", fields=None, filters={"title": "test"}))  
pp.pprint(res)
```

3.2.0.27 cURL

```
curl -s -X POST http://localhost/amcat/index/state_of_the_union/query \  
-H 'Content-Type: application/json' \  
-d '{"filters":{"title":["test"]}}' | jq -r ".results"
```

This is because it is empty for this document, just as the url field, which is absent from all documents in this index. We can add something to the new field and see if it shows up:

3.2.0.28 R

```
update_tags(index = "state_of_the_union",
            action = "add",
            field = "keyword",
            tag = "test",
            filters = list(title = "test"))
```

```
query_documents(index = "state_of_the_union",
               fields = c("title", "keyword"),
               filters = list(title = "test"))
```

3.2.0.29 Python

```
test_doc = list(amcat.query("state_of_the_union", fields=["id"], filters={"title": "test"}))
amcat.update_document("state_of_the_union", doc_id=test_doc["_id"], body={"keyword": "test"})
```

```
res=list(amcat.query("state_of_the_union", fields=["title", "keyword"], filters={"title": "test"}))
pp.pprint(res)
```

3.2.0.30 cURL

```
test_doc=$(curl -s -X POST http://localhost/amcat/index/state_of_the_union/query \
-H 'Content-Type: application/json' \
-d '{"filters":{"title":["test"]}}' | jq -r ".results[]._id")
curl -s -X PUT http://localhost/amcat/index/state_of_the_union/documents/${test_doc} \
-H 'Content-Type: application/json' \
-d '{"keyword": "test"}'
```

```
curl -s -X POST http://localhost/amcat/index/state_of_the_union/query \
-H 'Content-Type: application/json' \
-d '{"filters":{"title":["test"]}}' | jq -r ".results"
```

Now that we have a better idea of what an index is and how it looks like, we can create a new one>

3.2.0.31 R

```
create_index(index = "new_index", guest_role = "admin")
list_indexes()
get_fields(index = "new_index")
```

3.2.0.32 Python

```
amcat.create_index(index="new_index", guest_role="admin")
```

```
amcat.list_indices()
amcat.get_fields("new_index")
```

3.2.0.33 cURL

```
curl -s -X POST http://localhost/amcat/index/ \
-H 'Content-Type: application/json' \
-d '{
    "name": "new_index",
    "guest_role": "ADMIN"
}'
```

```
curl -s http://localhost/amcat/index/
curl -s http://localhost/amcat/index/new_index/fields
```

As you can see, the newly created index already contains fields. You could now manually define new fields to fit your data. Or you can simply start uploading data:

3.2.0.34 R

```
new_doc <- data.frame(
  title = "test",
  text = "test",
  date = as.Date("2022-01-01"),
  president = "test",
  year = "2022",
  party = "test",
  url = "test"
)
upload_documents(index = "new_index", new_doc)
```

```
get_fields(index = "new_index")
```

3.2.0.35 Python

```
new_doc = {  
    "title": "test",  
    "text": "test",  
    "date": datetime.strptime("2022-01-01", '%Y-%m-%d'),  
    "president": "test",  
    "year": "2022",  
    "party": "test",  
    "url": "test"  
}  
amcat.upload_documents("new_index", [new_doc])
```

```
amcat.get_fields("new_index")
```

3.2.0.36 cURL

```
curl -s -X POST http://localhost/amcat/index/new_index/documents \  
-H "Content-Type: application/json" \  
-d '{  
    "documents": [  
        {  
            "title": "test",  
            "text": "test",  
            "date": "2022-01-01",  
            "president": "test",  
            "year": "2022",  
            "party": "test",  
            "url": "test"  
        }  
    ]  
}'
```

```
curl -s http://localhost/amcat/index/new_index/fields
```

amcat4 guesses the types of fields based on the data. You can see here that this might not be the best option if you care about data types: party and president have been created as text, when they should be keywords; year is now a long type instead of double or integer.

Finally, we can also delete an index:

3.2.0.37 R

```
delete_index(index = "new_index")
```

3.2.0.38 Python

```
amcat.delete_index("new_index")
```

3.2.0.39 cURL

```
curl -s -X DELETE http://localhost/amcat/index/new_index
```

3.2.1 A Note on the ID Field and Duplicated Documents

AmCAT indexes can have all kinds of fields, yet one special field must be present in every document of every index: a unique ID. This ID is usually not that noteworthy, since the user does not really need to take care of it. This changes, however, with the special case of duplicated documents, that is, a document with the exact same information in the same fields. AmCAT does not normally check if your documents are duplicated when you upload them. However, when no ID is present in an uploaded document, as in the example we uploaded above, AmCAT will construct a unique ID from the available data of a document. Let us have another look at that document we titled “test”:

3.2.1.1 R

```
query_documents(index = "state_of_the_union", fields = NULL, filters = list(title = "test"))
```

3.2.1.2 Python

```
res=list(amcat.query("state_of_the_union", fields=None, filters={"title": "test"}))  
pp.pprint(res)
```

3.2.1.3 cURL

```
curl -s -X POST http://localhost/amcat/index/state_of_the_union/query \  
-H 'Content-Type: application/json' \  
-d '{"filters":{"title":["test"]}}' | jq -r ".results"
```

Note that AmCAT has automatically added an `_id` field (in R it is `.id` due to naming conventions) to the document. If we would upload the same document again, the algorithm that constructs the `_id` field would come up with the same value and the document would be replaced by the newly uploaded document. If we wanted to keep a duplicate for some reason, we could accomplish that by either changing at least one value in a field or by assigning an ID column manually:

3.2.1.4 R

```
new_doc <- data.frame(  
  .id = "1",  
  title = "test",  
  text = "test",  
  date = as.Date("2022-01-01"),  
  president = "test",  
  year = "2022",  
  party = "test",  
  url = "test"  
)  
upload_documents(index = "state_of_the_union", new_doc)
```

```
query_documents(index = "state_of_the_union", fields = NULL, filters = list(title = "test"))
```

3.2.1.5 Python

```
from datetime import datetime  
new_doc = {  
  "_id": "1",  
  "title": "test",  
  "text": "test",  
  "date": datetime.strptime("2022-01-01", '%Y-%m-%d'),  
  "president": "test",  
  "year": "2022",
```

```
"party": "test",
"url": "test"
}
amcat.upload_documents("state_of_the_union", [new_doc])
```

```
res=list(amcat.query("state_of_the_union", fields=None, filters={"title": "test"}))
pp.pprint(res)
```

3.2.1.6 cURL

```
curl -s -X POST http://localhost/amcat/index/state_of_the_union/documents \
-H "Content-Type: application/json" \
-d '{
    "documents": [
        {
            "_id": "1",
            "title": "test",
            "text": "test",
            "date": "2022-01-01",
            "president": "test",
            "year": "2022",
            "party": "test",
            "url": "test"
        }
    ]
}'
```

```
curl -s -X POST http://localhost/amcat/index/state_of_the_union/query \
-H 'Content-Type: application/json' \
-d '{"filters":{"title":["test"]}}' | jq -r ".results"
```

As you can see, we now have the example document in the index twice – although with different IDs. To simulate what would have happened without an ID, or rather if the ID had been constructed automatically, we can upload different data, but with the same ID to see what changes:

3.2.1.7 R

```

new_doc <- data.frame(
  .id = "1",
  title = "test",
  text = "A second test",
  date = as.Date("2022-01-02"),
  president = "test",
  year = "2022"
)
upload_documents(index = "state_of_the_union", new_doc)

```

```

query_documents(index = "state_of_the_union", fields = NULL, filters = list(title = "test"))

```

3.2.1.8 Python

```

from datetime import datetime
new_doc = {
  "_id": "1",
  "title": "test",
  "text": "A second test",
  "date": datetime.strptime("2022-01-02", '%Y-%m-%d'),
  "president": "test",
  "year": "2022"
}
amcat.upload_documents("state_of_the_union", [new_doc])

```

```

res=list(amcat.query("state_of_the_union", fields=None, filters={"title": "test"}))
pp.pprint(res)

```

3.2.1.9 cURL

```

curl -s -X POST http://localhost/amcat/index/state_of_the_union/documents \
-H "Content-Type: application/json" \
-d '{
  "documents": [
    {
      "_id": "1",
      "title": "test",
      "text": "A second test",

```

```
    "date": "2022-01-02",
    "president": "test",
    "year": "2022"
  }
]
}'
```

```
curl -s -X POST http://localhost/amcat/index/state_of_the_union/query \  
-H 'Content-Type: application/json' \  
-d '{"filters":{"title":["test"]}}' | jq -r ".results"
```

The document with the ID 1 has been replaced with the new data. This is the normal behaviour of AmCAT: when we tell it to add data to an already present document, identified by the ID, it will be replaced. If a field was present in the old document, but not in the data it is replaced with, this field will be empty afterwards.

4 Access Management and Document Sharing

In this chapter, we describe the different roles you can get a give to access your indexes, and how to use them to safely share your data. Roles are important because they determine how much users can view and modify in your data.

4.1 Access Management

In AmCAT, there are four types of roles:

- Admin: It is a user that can modify other users' roles and perform any changes to the data.
- Writer: A user that can modify the data.
- Reader: A user that can see the data.
- Metareader: A user that can see all the data fields, but only a snippet of the text.

You can add users with their respective roles using the [Graphic User Interface \(GUI\)](#) or via the [Application Programming Interface \(API\)](#).

4.1.1 Access Management via the Graphic User Interface

To modify the access to an index, the index has to already exist. For this example, I will refer to the index “Example Index” that we created in chapter 3 Document Storage. Nevertheless, all the steps apply to any other index.

Go ahead and open the index by clicking on it:

My indices


Search   

Example_Index 

My first index 





 **Index Admin**

From there navigate to “Index Admin” located at the top of the window. In there, you will the XXX: Settings and Users.

Settings Users

Example_Index  Archive 

My first index

Guest role METAREADER

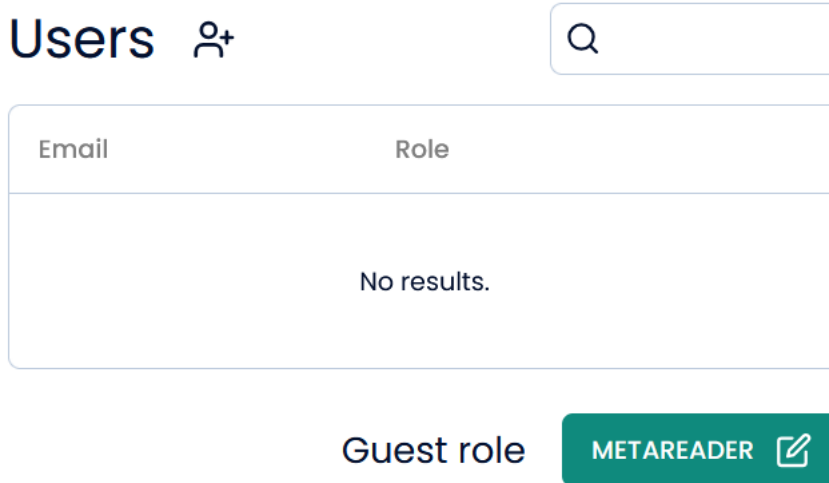
Own role METAREADER

Folder

Image

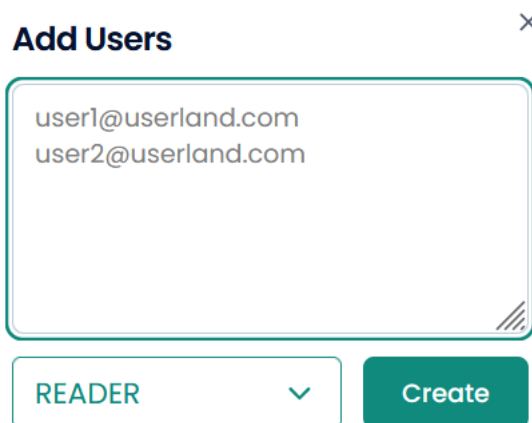
The default settings is that yours and the guest’s roles are Metareaders. This means that both of you can only access the date, title, url etc, and snippets of the text when you search.

To change these roles, you can navigate to “Users”. This will open the following window:



From there, you can add

a user by clicking “Add user” , which will pop up the following window:



Now, you can add all the users’ emails, including your, with their respective roles. Click “Create” when you are done. To see the changes applied, you need to log-off and log-in again.

4.1.2 Access Management via the Application Programming Interface

```
list_users()
```

```
create_user(  
    email = "user@example.com",  
    password = "test",  
    global_role = "writer",  
    index_access = "state_of_the_union"  
)
```

While you are here, it probably makes sense to change the admin password to something secure:

```
modify_user(email = "admin", new_password = "a")
```

```
add_index_user()
```

```
delete_index_user()
```

```
modify_index_user()
```

4.2 Document Sharing

5 Public Document Presentation

Sharing documents with your collaborators and research assistants is not the only scenario in which you want to make documents available to others. Sometimes, you want to share a document collection with a wider public. Whether you want to make the full-text documents available or just let users discover frequencies of terms and word combinations in your text, a dashboard built on top of amcat can be the solution. This chapter introduces you to how you can build a dashboard on top of amcat.

! Coming soon...

6 Fast searches in amcat databses

Now that we have covered basic filtering in AmCAT, we can turn to the next task: identifying all terms that represent a broader concept. Some concepts, such as Ukraine or President Obama, are relatively easy to define using a simple keyword search. Other concepts such as emotions like fear or anger, or general sentiments like positive and negative are difficult to capture with keywords alone.

Other concepts are more manageable. For example, when searching for references to terrorism, it is usually sufficient to include variations such as terror, terrorist, and terrorists. More complex concepts, such as crime, require a wider set of terms because they appear in many different forms and contexts.

AmCAT provides several tools to support the construction of effective search strings. These options are summarized in the table below. Note that all search terms must be written in lowercase, while Boolean operators (AND, OR) must be written in uppercase.

Notation in query screen	Description
obama terrorism; obama OR terrorism	Searches articles that contain obama or terrorism. (A space automatically functions as OR.)
obama AND terrorism	Articles containing obama and terrorism
obama NOT terrorism	Articles containing obama, but not terrorism
“obama fights terrorism”	Searches the exact phrase “obama fights terrorism”
“obama israel”~10	Finds Obama and Israel within 10 words of each other. 10 is a random number, you can choose any other numbers which are higher than 2.
(obama OR Israel) AND (iran OR terroris*)	This search finds articles that contain either Obama or Israel, and at the same time either Iran or any word starting with “terroris” (e.g., terrorism, terrorist).
(“obama iran”~10 OR “obama terrorism”~10 OR “biden iran”~10 OR “biden terrorism”~10)	This search looks for anything where Obama or Biden appears close to Iran or terrorism (maximum 10 words apart).

Notation in query screen	Description
* or ? (wildcards)	A ? in a search term indicates that the place of ? can be changed in a another character. For example: terroris?, can be terrorism, or terrorist. The place of a * can be changed in a character either in a couple of characters. For example: Economi*, can be changed in economic, economical, economic state, but also in longer words too. Note: you can't put a wildcard at the beginning of a word.

7 amcat for document annotation

! Coming soon...

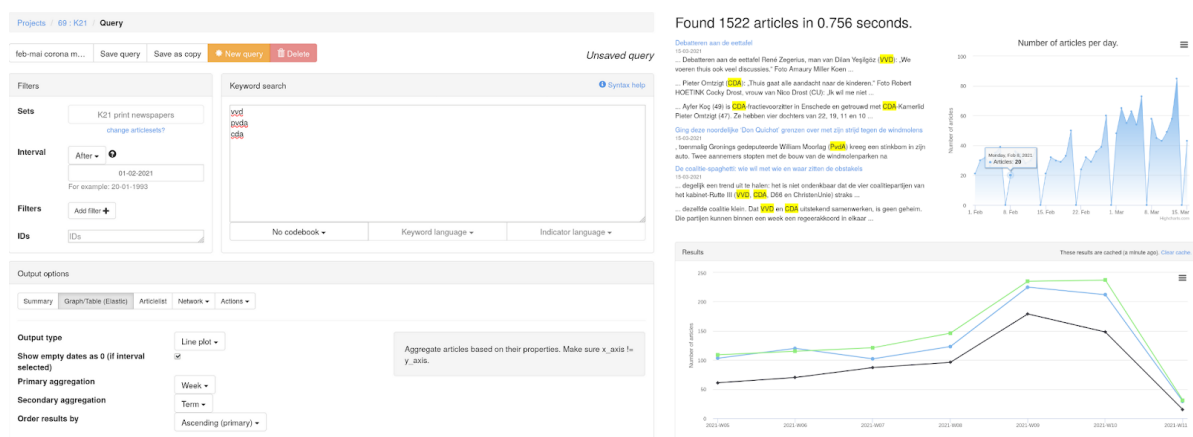
8 Preprocessing and machine learning

! Coming soon...

8.1 Why nlpipeline?

9 About amcat

AmCAT has been in development in various guises since about 2001 as a text research / content analysis platform at the VU. From the start, the core has been a **database** of news articles, an **API** allows other programs and power users to connect to it directly from e.g. Python or R and an intuitive **query interface** for performing quantitative analysis. The screenshots below give some indication of the database and query interface of AmCAT 3.5.



In the last 20 years, the availability of user friendly tools for text analysis has greatly increased. Many proprietary platforms like LexisNexis or Coosto allow basic text analysis directly on the dashboard, while R and Python based toolkits like *quanteda* (Benoit et al. 2018) and *scikit-learn* (Pedregosa et al. 2011) have put very powerful analysis possibilities within the reach of a technically inclined social scientist. However, even with those advances we believe that a tool such as AmCAT can still play an important role for text analysis in the social sciences:

- An intuitive user interface gives a visual overview of the stored texts and **allows non-technical users to interact with the system**, without restricting its use to a single proprietary platform. This also makes it a **valuable resource in teaching** quantitative content analysis.
- A shared storage format, and API makes it easier to **share and reuse** tools and analysis scripts. It also aids **reproducible science** by allowing analyses and data to be shared and validated based on immutable data sets.

- Finally, as communication science data is often proprietary, fine-grained access control allows results to be shared and validated without giving access to the underlying data using **non-consumptive research**.

This is, why AmCAT was redesigned within work package 7 of the [OPTED \(Observatory for Political Texts in European Democracies: A European research infrastructure\)](#) project. The goal of the redesign is to create an infrastructure that is simple and flexible enough to be adapted for various projects at OPTED partner universities, other research insitutions and by basically everyone interested in text analysis.

References

- Benoit, Kenneth, Kohei Watanabe, Haiyan Wang, Paul Nulty, Adam Obeng, Stefan Müller, and Akitaka Matsuo. 2018. “Quanteda: An r Package for the Quantitative Analysis of Textual Data.” *Journal of Open Source Software* 3 (30): 774. <https://doi.org/10.21105/joss.00774>.
- Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. 2011. “Scikit-Learn: Machine Learning in Python.” *Journal of Machine Learning Research* 12: 2825–30.