

Sharing geological collection data with GeoCAsE 2.0

Figure

Authors: Tom Dijkema

Dr. Niels Raes

Affiliations: Naturalis Biodiversity Center, Leiden, the Netherlands

Date: 14 January 2022

Version: 0.1

Table of Contents

A. Introduction	4
Introducing the problem, solution and matter	4
A brief but strong introduction on the subject and the technical process	4
B. Configuring a webserver	5
Making use of the Naturalis webserver	5
Getting a server	5
Configuring the server user	6
Security measures	9
C: Using Docker	15
Introduction	15
Naturalis service	15
A brief introduction to Docker	15
Terminology	15
How it works	16
Installing Docker	17
Basic commands	Fout! Bladwijzer niet gedefinieerd.
Advanced methods	20
D: Data(base) preparation	24
Naturalis service	24
Creating the database	24
Preparing the dataset	26
Inserting metadata	29
Defining relations	30
Future datasets	31
E: Installing and using the BioCase Provider Software	33
Naturalis service	33
The purpose of BioCAsE	33
How it works	33
Installing BioCAsE with Docker	Fout! Bladwijzer niet gedefinieerd.
Data source set up	Fout! Bladwijzer niet gedefinieerd.
Table configuration	36

Mapping to ABCD-EFG	38
F: Mobilizing data to GeoCAsE	43
Naturalis service	43
What is GeoCAsE?	43
Publishing data to GeoCAsE	43
G: Useful extras	44
Summary of possible occurring errors	44
MIDS for mapping geological data	44
Multiple images	44

A. Introduction

Introducing the problem, solution and matter

xxx

A brief but strong introduction on the subject and the technical process

xxx

B. Configuring a webserver

To mobilise geological data to the GeoCASE portal, it is mandatory to install the BioCASE software on an online server. The BioCASE instance on a webserver serves as the endpoint from which GeoCASE can pull the data. This chapter describes how to set up a webserver in preparation of the BioCASE installation process.

Making use of the Naturalis webserver

Besides operating an own webserver with a BioCASE installation, it is possible to make use of Naturalis' services. Naturalis offers a webserver with a BioCASE installation that can host multiple datasets from different institutions to facilitate the mobilisation of geological data to the GeoCASE 2.0 portal. If you wish to make use of this service, please contact the contact person of Naturalis as mentioned in the appendix.

Getting a server

To set-up your own webserver, you can approach one of the many server providers and order a package. This package often consists of a VPS (Virtual Private Server) in combination with a terminal and/or operating system to manage the server. It is important that an individual server is acquired and not a hosting package which often is used for hosting simple websites. The reason for this is that we need to be able to edit the property settings of the webserver, use different ports and be able to install specific software. As an example, we used a server package from the TransIP provider. See the specifications below (>):

- Provider: [TransIP](#)
- Package: BladeVPS
- Operating System: CentOS
- Server operating tool: cPanel
- 1 Core
- 1024 MB RAM
- 50 GB SSD

This server contains enough storage space and operating power to share the data with the GeoCASE portal. If necessary, a package can be scaled up relatively easily. The **operating system** we are using is called CentOS (sometimes referred to as AlmaLinux). This is a type of **Linux** distribution. We also included cPanel as a GUI (graphical user interface) to easily interact with the server. cPanel consists of two different interfaces: a so-called WHM (WebHost Manager) interface for managing the server's back-end; and the core cPanel interface that is most focused on managing the server's content. During the purchase you have to define the domain on which you will install the server. We made use of a subdomain, which is an extension of an existing domain. In this case: nlbif.nl. The resulting domain is as follows: biocase.nlbif.nl. If you do not already own a domain, it is mandatory to register a domain name which can also be done at TransIP or almost any other provider.

If the installation of the server or defining the domain causes any problems, make sure to contact the provider you will be using to get support. It is important to set up the webserver in a correct way so this does not interfere with processes on the latter.

After purchasing the server you receive an email with the following information:

- cPanel address, e.g. <https://biocase.nlbif.nl:2083/>

- WHM address, e.g. <https://biocase.nlbif.nl:2087/>
- Username and password

This is crucial information to get to work with the server. In the case of the Naturalis server we can access the cPanel interface at port 2083 and the WHM interface at port 2087.

A port is an important term we will be using in this document. It refers to the different access points of the server. In this case we could access the cPanel interface at port 2083 by typing it in after our registered domain name like: biocase.nlbif.nl:2083. The same goes for the WHM interface which is reachable via: biocase.nlbif.nl:2087. Important to remember is that only one service can be run on one port. Multiple services can be installed to run on the same port, but they cannot be run simultaneously.

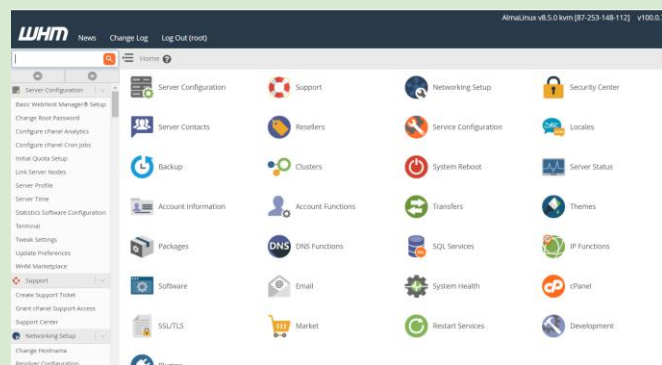
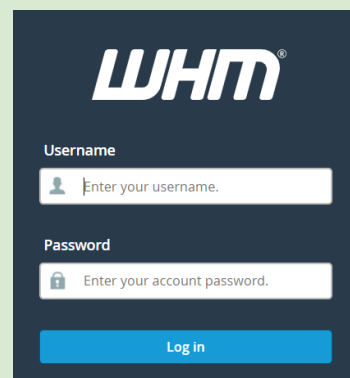
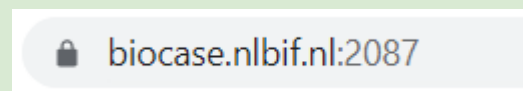
Configuring the server

After receiving the details of our freshly bought server, we need to configure it. First we will take a look at the WHM interface because we need to set up a cPanel user account to access the cPanel interface.

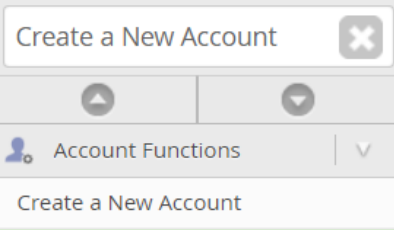
1. Go to the address of your WHM installation by typing in your domain name in the URL bar of your browser and adding the 2087 port, like for example: biocase.nlbif.nl:2087

2. You will see a login form where you should be able to login in with the username and password credentials that were received in the mail from your provider

3. After logging in you will see the main page of the WHM portal. This page offers all kinds of tools that can be used



Create a cPanel user to login to the cPanel interface that is linked to a specific section of the domain. A cPanel user account has access to the main domain section, in our case: biocase.nlbif.nl.

<p>1. Type and click in the search bar on: 'Create a New Account'</p>	
<p>2. Fill in the domain name you registered, for example biocase.nlbif.nl</p>	<p>Domain <input type="text" value="biocase.nlbif.nl"/> ✓</p>
<p>3. Think of a suiting username for the cPanel account and fill it in</p>	<p>Username <input type="text" value="biouser"/> ✓</p>
<p>4. Think of a suiting password and fill it in, or generate a random one</p>	<p>Password <input type="password" value="....."/> ✓</p>
<p>5. Retype the password</p>	<p>Re-type Password <input type="password" value="....."/> ✓</p>
<p>6. Make sure to document your credentials somewhere save</p>	
<p>7. Save the user by pressing 'create'</p>	<p>Create</p>

To check if the user configuration is correct, navigate to the cPanel interface that is located on port 2083 like: <https://biocase.nlbif.nl:2083>. If correct, a similar login form will be shown where you can login with the credentials you just created.

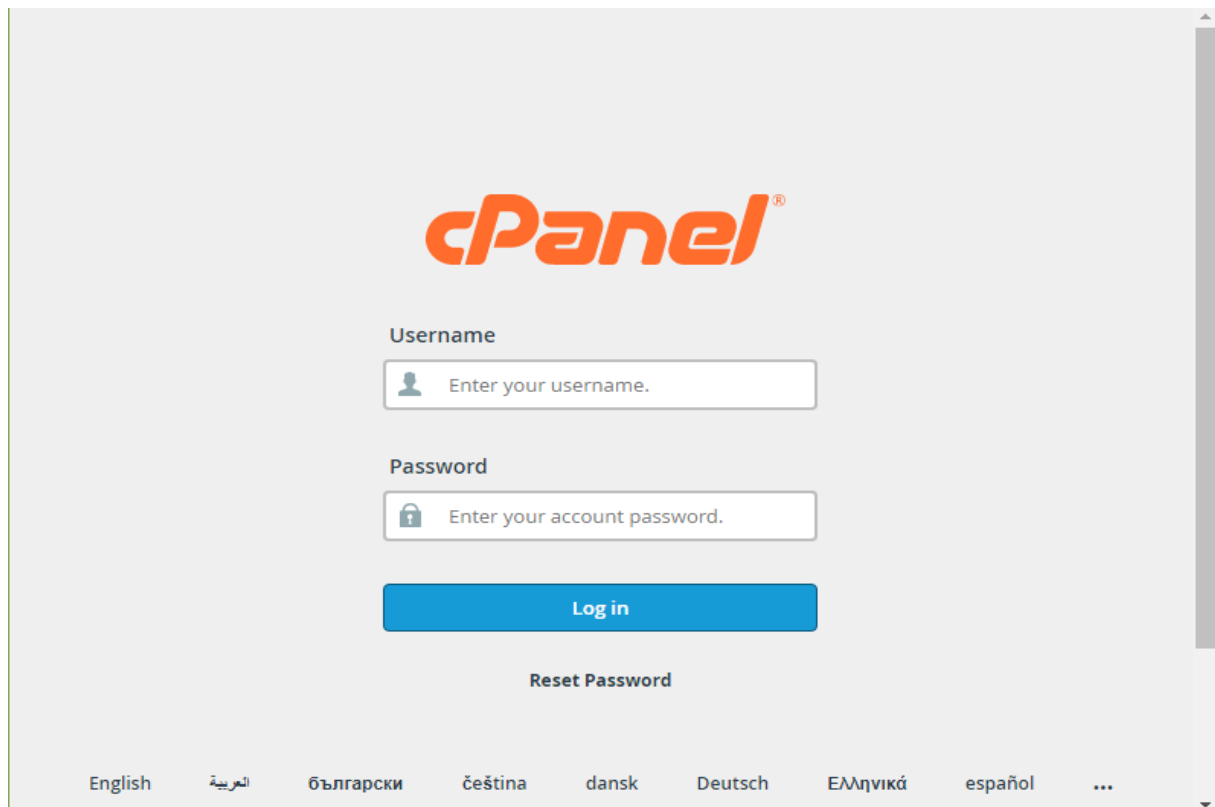


Figure #: cPanel login page.

If the cPanel user is not working, check if you have filled in the correct credentials and domain name. This can be done by searching and clicking in the WHM search bar onto: 'Modify an Account'. Then select the right account, if only one is existent, this account will automatically be displayed.

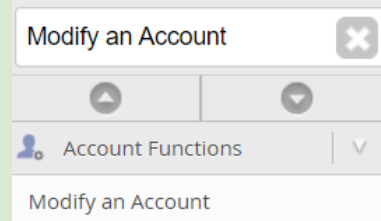
Next we will have to give the user permission to make use of the 'shell'. The shell represents a terminal from where the user can interact with the server. To give permission we need to add the user to the so-called 'Wheel Group'.

Give the user access to the shell:

1. Search and click in the WHM search bar onto 'Manage Wheel Group Users'
2. Select the cPanel username you created from the list and press on 'Add to Group'

The screenshot shows the WHM interface. At the top, there's a search bar with 'Manage Wheel Group Users' entered. Below it, there's a 'Security Center' dropdown menu. The main content area shows a list of users: _imunify, adm, bin, biouser, chrony, clamscan, clamupdate, and cpanel. The 'cpanel' user is highlighted. At the bottom of the list, there's a blue button labeled 'Add to Group'.

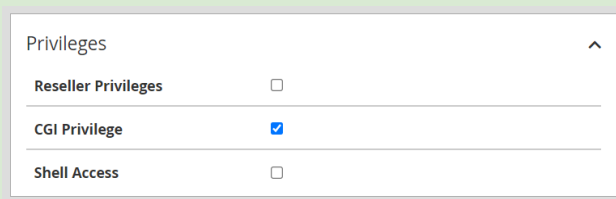
3. Search and click in the search bar onto 'Modify an Account'



4. Click on the account you created (if only one exists, this one will be automatically displayed) and press 'Modify'



5. Expand the privileges element



6. Check the option of 'Shell Access'



7. Save this setting by pressing 'Save'



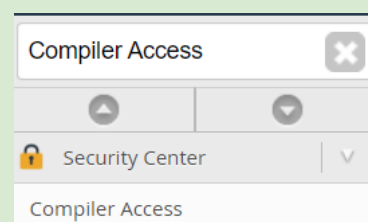
Security measures

To secure the server it is important to configure several security settings. These range from shutting services to implying certain security settings. In this section we will skip through the recommended security settings.

First off, the compiler access. Some servers are built for compiling files, however this is not an activity that is needed for the purpose to share data with GeoCASE. Also, if non-intended users manage to access the server, they can manipulate the compiler. When not used, it is best to disable the compiler access.

Disabling the compiler access:

1. Search and click in the WHM portal onto 'Compiler Access'



2. If enabled, press 'Disable Compilers' to shut access to the compiler

Status
Compilers are enabled for unprivileged users.
Configure
Disable Compilers


An easy way to identify more security issues is by using the security advisor. This is an inbuilt tool that will scan the server settings and identify possible security breaches. Identified issues are categorised in three classes: important, information and verified. Important issues need attention and can often be solved by adjusting the settings. If possible, try to determine the importance of identified issues and apply the necessary changes. Issues labelled with 'Information' indicate important security settings that are set. These should be monitored regularly. Lastly, issues labelled with 'Verified' indicate which components are set well and secure.

Using the security advisor:

1. Search and click in the WHM portal onto 'Security advisor'

Security Advisor	✕
▲	▼
🔒 Security Center	▼
Security Advisor	

2. Run an automatic scan or press 'Scan Again' for a manual scan

	cPanel Security Advisor
Scan Again	

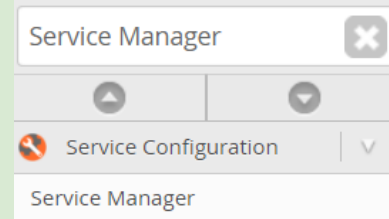
3. Read the advice given by the security advisor

If you do not possess the correct knowledge to improve the security of the server, make sure to contact us or your IT-department (if existent) to review the settings and security advice.

Next, we have a look at the list that shows all the running services. Not all services are mandatory to be run to mobilise data to GeoCAsE. Instead they can be turned off to prevent unwanted visitors and improve the server quality. This can be done by using the Service Manager in the WHM interface.

Shut down unnecessary services:

1. Search and click in the WHM portal onto 'Service Manager'



2. Deactivate all unnecessary services, see the list below for reference

3. Press 'save' to save the settings



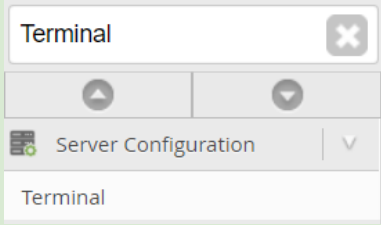
Recommended services to be run (the rest can be turned off):

Service	Description
ChkServd	Responsible for checking, monitoring and restarting services.
ModSecLog	Parses the ModSecurity audit log and stores the events in the modsec database for later viewing or analysis.
cPanel DAV Daemon	-
cPHulk Daemon	cPHulk Brute Force Protection
Cron Daemon	-
cPanel DNS Admin Cache	cPanel DNS Admin Cache Service
Apache Web Server	Web Server
IP Aliases	-
MySQL Server	MySQL Database Server
DNS Server	PowerDNS
Name Service Cache Daemon	-
rsyslog System Logger Daemon	Enhanced System Logger Daemon

Table #: Recommended webserver services to be run

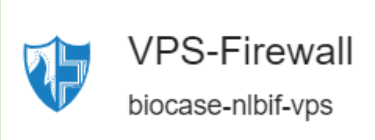
As we will be making use of MySQL to set up a database on the latter, it is recommended to limit MySQL to listen only to queries from the webserver. When not set, it is possible for foreign users to login to the database and make possible changes when they are in possession of the correct credentials. To solve this thread, we can change the bind address which MySQL listens from to be set only on our server address.

Set the MySQL bind address to our server address:

1. Search and click in the WHM portal onto 'Terminal'	
2. Activate MySQL by typing 'mysql'	mysql
3. Change the bind address variable to your servers ip address by using the following command	bind-address = [your ip address];
4. Exit mysql by typing 'quit'	quit
5. Restart MySQL by using the following command	Service mysqld restart
6. To check if your bind address has been changed properly, type in the following command in the activated mysql command line (type mysql in the terminal) to display the variable:	mysql SHOW variables LIKE 'bind_address';

To secure the gateways to our server, like MySQL, even more; we can access the Firewall properties in the online portal of TransIP. This portal can be found by logging in to your TransIP account. In case you are using another provider, a similar portal should be available. When logged in search for Firewall settings. A page containing all the server ports should appear in which we can open or close ones. First off we want to close the unnecessary ports as we will not be using these. Make sure the Firewall is activated, otherwise these changes will have no effect.

Close the unnecessary ports:

1. Open the Firewall section in your providers online environment	
2. Check if the Firewall is up and running (image translates to: 'activate VPS-Firewall for this VPS'; 'On')	VPS-Firewall inschakelen voor deze VPS. <input checked="" type="checkbox"/> Aan
3. Close all unnecessary ports, use the table below as a reference. This states all the ports that are opened on Naturalis' server	

Service	Port (range)
cPanel - FTP passive port range	30000-50000
cPanel - WHM	2086
cPanel	2082
cPanel - SSL	2083
cPanel - WHM SSL	2087
cPanel - Webmail	2095
cPanel - Webmail SSL	2096
cPanel - FTP	21
cPanel - HTTP	80
cPanel - HTTPS	443
cPanel - IMAP	143
cPanel - IMAPS	993
cPanel - SMTP	587
cPanel - SMTP	465
cPanel - SMTP	25
cPanel - SSH	22

Table #: Firewall ports of the webserver

At last, we want to add two ports to create access to MySQL and our future BioCAsE installation. MySQL should be reachable via port 3306 from any point of the server. We will install BioCAsE on port 8080 as an extension of 80.

Open a port for MySQL and BioCAsE:

1. Add port 3306 to the Firewall' list that is globally reachable from the server's ip address

MySQL

MySQL

87.253.148.112/32

3306

TCP

2. Add port 8080 to the Firewall' list

Custom

Biocase - Docker

	8080	TCP
--	------	-----

C: Using Docker and installing BioCAsE

Introduction

Before we will install the BioCAsE Provider Software for mobilising geological records to GeoCAsE, it is good to understand how the Docker software functions. We provide this introduction to Docker because it is by far the most easy and efficient way to install and manage BioCAsE. It also is recommended by the original developers of BioCAsE.

Naturalis Service

When making use of the Naturalis server with the BioCAsE instance to mobilise your collection data, it is not necessary to deploy a Docker installation of BioCAsE on a webserver. By making use of the Naturalis service you will make use of the server of Naturalis and add your own dataset to the BioCAsE installation. You will have your own publisher details and dataset metadata information along with the records that you share with GeoCAsE. However, background information on how to deploy a Docker installation of BioCAsE on a webserver might be useful knowledge.

A brief introduction to Docker

A common problem in the world of software development and distribution is the takeover of software. Software often requires certain packages and/or settings that not every computer or server owns by default. This can result in missing system requirements and nonfunctional software. To ensure that the BioCAsE software and all of its dependencies are correctly installed a Docker container with the BioCAsE software and its dependencies can be deployed on a webserver. Docker is an open source platform for developing, transferring and executing software that solves the dependency problem. It does so by offering complete application packages called containers. These containers are easy to run and contain the complete image of the software and its dependencies. This way, no further installations than the container itself need to be done. Besides, containers can easily be updated or removed without harming the source data.

A Docker image of the BioCAsE software is available for free use https://wiki.bgbm.org/bps/index.php/Installation#Installing_BioCAsE_from_a_Docker_image. This means that by using Docker we can install and sustain a BioCAsE installation with all its dependencies on our webserver.

Terminology

In this chapter we use the following terms:

Client: The Docker client functions as the primary way to interact with Docker. Commands can be sent via the client to Dockerd, the daemon, to execute certain actions such as creating, updating or removing containers.

Container: When using Docker an application will be stored inside of a container. A container consists of an isolated environment that runs on the operating system of the host such as Linux or Windows. A container contains all dependencies of an application as defined in the image.

Daemon: The daemon is the engine of Docker. It listens for commands from the client and on noticing, will execute these. A daemon is also capable of communicating with other daemons.

Dockerfile: A Dockerfile represents the template for the construction of an image. During the process of building an image in a container, the Dockerfile also indicates which commands should be executed. For example to install certain dependencies. Docker only uses Dockerfiles that are identified as relevant for building the layers of the image. A layer represents a part of the application. Building a container can be done by using multiple layers which translates to Dockerfiles. The process of building layer after layer allows Docker to be a light and relatively fast program due to the distribution of available workspace of the webserver.

Images: In the context of Docker an image does not mean a picture, but rather a readable template with exact instructions on how to build a container. In most cases, an image is based on an existing application with some favourable changes. In addition, it is possible to create an image from scratch by using a Dockerfile. This can be done by creating an empty Dockerfile and filling it with instructions on how to build the image.

Registry: A Docker registry is where images are stored. The Docker Hub is a public registry where everyone can share and make use of images. It is also possible to make a private registry to save images.

REST API: An API is a web service based endpoint that can be used by applications to extract certain information. A REST API (Representational State Transfer) is a type of API that makes use of the 'RESTful' architecture. This basically represents an optimised way of building the inner structure.

How it works

Docker makes use of a client-server architecture as seen in **figure #**. The client communicates with the Docker daemon (fig #; DOCKER_HOST) that handles heavy tasks like building, running and distributing containers. The client and Docker daemon can function on the same system, but a client can also be connected with an external Docker daemon. The communication between the client and the Docker daemon is established via a REST API.

The Registry (Fig #; Registry) serves as a place where images are saved. The Docker Hub is a public registry that automatically observes for available images. When Docker does not detect an image on the local system, it will automatically search in the Docker Hub for a matching one. In addition, it is possible to run a private registry in Docker Hub.

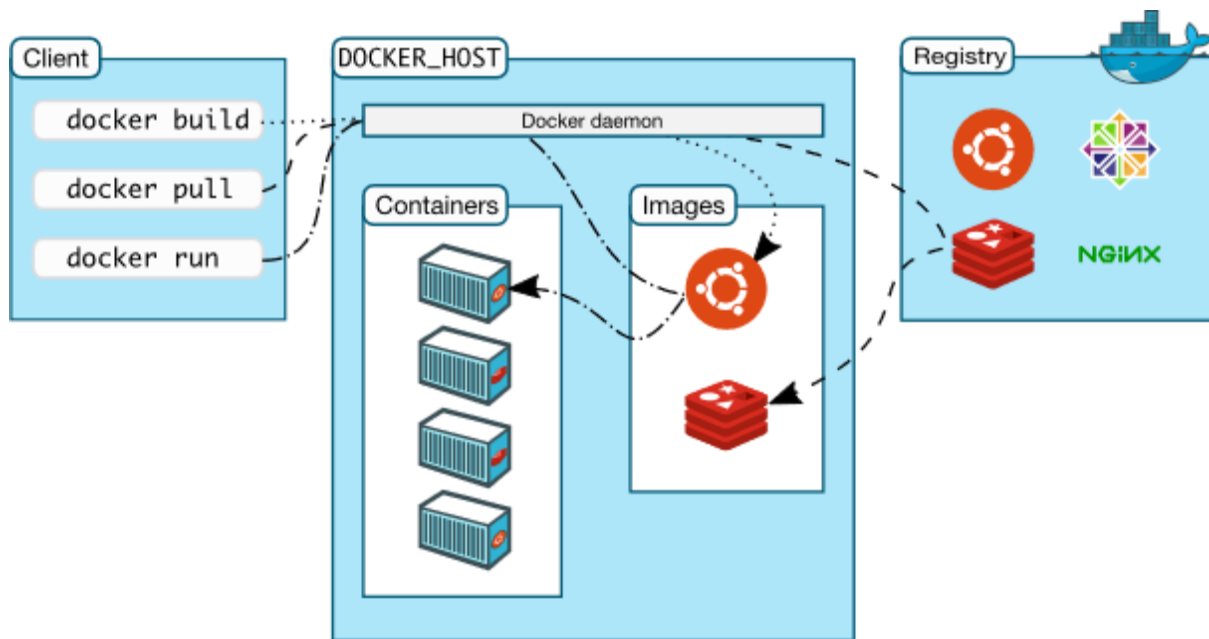


Figure #: The Docker Architecture

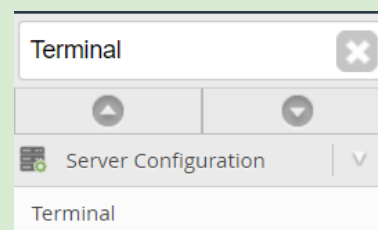
When a container is executed through a command, Docker is activated. The program creates an isolated environment around the defined content of a container which has its own folder structure. Every contextual element from a container that Docker requires needs to be defined in an image. Docker itself does not add anything to the contextual mix. Because of this, the image also contains configuration options for the container like: **environment variables**, a standard **command line** to execute and other **metadata**. The above stated requirements is in fact all that is needed to set up a basic Docker application.

Installing Docker

Installing Docker is a relatively simple task that can be done in two separate ways. The first is to download an installer from the Docker website (<https://docs.docker.com/get-docker/>). This installer, specified for the Windows, Linux or iOS operating systems, installs a graphical interface for using Docker. The latter is to install Docker via the command line in the **terminal**. Because we are going to install Docker on an online server, we use the command line in the terminal as this is the most straightforward way. As mentioned in the server installation chapter we selected CentOS as the operating system. This is a Linux distribution and the main reason why we use Linux commands in this manual.

Install Docker via the terminal in the WHM interface:

1. Search and click on 'Terminal'



2. When the terminal is loaded, type

```
sudo yum install docker-ce docker-ce-cli containerd.io
```

in the following command

This command will have Yum install the Docker software and its dependencies. Yum is a free to use installer for Linux that for example can be used to install, update or remove software. We ran the command with 'sudo', which means we wanted to run the command as administrator. To check if Docker is installed correctly, try to start the daemon by using the Docker start command, again from the terminal.

1. Start the Docker daemon, type in the following command

```
sudo systemctl start docker
```

2. Docker should run by now. To test if Docker is running, type in the following command

```
sudo systemctl status docker
```

3. This should display Dockers details including an active statement that says it is running or not.

```
Active: active (running) since Mon 2021-12-13 22:16:25  
CET; 1 months 8 days ago
```

If Docker does not start or run well, try to reinstall the software. This can be done by removing Docker with: `sudo yum remove docker`; and installing it once more with: `sudo yum install docker`.

Lastly we will add Docker to our start-up applications. This will automatically start Docker when the server is rebooted. This is optional, you can also choose to start Docker manually.

Indicate Docker as a start-up application from the terminal:

1. Type the following command in the terminal

```
sudo systemctl enable docker.service
```

2. Then type in this command

```
sudo systemctl enable containerd.service
```

3. This should enable Docker as a start-up application. You can test this by rebooting the server. Use the following commands if you want to disable this behaviour type

```
sudo systemctl disable docker.service  
sudo systemctl disable containerd.service
```

Basic commands

To get familiar with Docker's functionalities, this next paragraph describes the basic commands to manage a container application like BioCASEl. Every described action can be executed via the terminal (See section xx). It is not mandatory to execute the following actions but serves to get experienced with Docker. The basic commands relate to a simple standard. As an example we look into creating, starting, stopping and demolishing a car like a real live situation.

Creating: To create a car in real life, you would first need a template for the chassis. The same applies to a virtual Docker container. This template is called an image and represents the application

and all of its dependencies that are needed. To create a container with a certain image we can use a locally stored image, or grab one from the Docker Hub. If Docker does not recognize a local image, it will automatically search on the Docker Hub for a corresponding one.

There are several commands which can be used to create a container. The most straightforward one is the **build** command. This command builds a container from a locally stored image. It makes use of the image's location and can be given a name with the **-t** tag. The command shown below could be used when executing from the directory where the image is saved:

Build a container with the **build** command

```
docker build -t <container name> .
```

Another way to create a container is by using the **run** command. This command is very useful because it makes use of the auto-search function. As mentioned before, this will automatically search on the Docker Hub for an image if no image is found locally. Besides, the **run** function will also start the container, we will discuss that shortly. The command below will search in the Docker Hub for the BioCASE image if not found locally, download and install it to a new container:

Run a container with the **run** command

```
docker run -dp 8080:80 biocase/bps
```

The container will be named after the image name, in this case: biocase/bps. Important to notice is the port that we defined with **-p** tag. By running this command the application will be set to run on port 8080:80, which is an extension of the default 80 port for the **HTTP** (Hypertext Transfer Protocol). Also, the **-d** indicates a container to be run as a background process. The **run** command can also be used to run a container that was built with the **build** command. Likewise the port will again be defined in the **run** command for the existing container.

Starting: After creating a car we would want to take it for a drive. This means we will have to start the engine by turning the key. The same applies for a Docker container. After creating, a container can be activated using the start command. This command needs an identification key to the container. A key, in Docker terms, can be for example the name, a tag or id of a container. To acquire a key, we need to display the properties of our Docker container.

Start a container:

1. Acquire the container's properties

```
docker ps -a
```

2. With the key id, start the container with the following command:

```
docker start <container id>
```

Stopping: To stop a car from running, you turn the key anti-clockwise. The same goes for a Docker container. To stop a container use the stop command with the identification key.

Stop a container from running with the **stop** command

```
docker stop <container id>
```

Important to notice is that when a Docker container is stopped, all services will be shut down. This means all data stored in the Docker container will be lost. Docker supports volumes and multi-container applications to prevent data from disappearing when a Docker container is stopped. These methods will be pointed out in the advanced paragraph. The BioCAsE image supports volumes so data will persist. More on that later in **Chapter D** on the BioCAsE software.

Demolishing: When a car is broken and ready to be demolished it will be crushed or taken apart for parts. When a Docker container is no longer needed, it can be deleted. The **rm** function removes a container by using an identification key. Remember that all elements from the image will be deleted from the container. The image itself will not be removed.

Remove a container with the **rm** command

```
docker rm <container id>
```

By knowing how to create, start, stop and demolish Docker containers you will be able to host an application such as BioCAsE and proceed to the next chapter. By interest, you may want to have a look at the advanced methods paragraph for a more in depth look at Docker's capabilities and requirements to run BioCAsE.

Advanced methods

A couple of advanced methods give more insight into Docker's capabilities. It also shows how to retain data when reinstalling or removing Docker containers.

Volumes: Docker volumes are an option to retain data. A volume has the ability to communicate with a specific path in the **file hierarchy** of the **host machine**. When Docker detects a change in the container, like adding or editing a record, the volume will communicate this action with the host machine. The host machine keeps track of this data so the information can be accessed when rebooting the container. It basically comes down to external files, such as text or CSV, that save the data and can be read when restarting a container.

There are two types of volumes. The first one, called a **named volume**, uses external files. With this method it is also possible to create a working directory to dynamically update an application. The second one is based on setting up a **network**. Within a network multiple images can communicate with each other. An example we will use for the BioCAsE software is a network consisting of a BioCAsE image and a MySQL database. The MySQL database contains the data, so when a container is restarted the data is retained. Networks can also be used for different aims such as the preservation of data and connection between multiple applications like API' for functional purposes. The establishment of a network consisting of multiple images in a single Docker container is called a

multi-container application.

Docker Compose: A third method (besides **build** and **run**) to create a Docker container is to use Docker Compose. Docker compose is an inbuilt set of functionalities that makes the creation of containers, volumes and networks simpler. It uses a single file that extracts all necessary information. For example, a Docker Compose file defines the images, port, working directory, volumes and environmental variables for network applications. BioCAsE makes use of a Compose file so that no complicated commands need to be executed in the terminal. Instead, the following command can be run to set-up the entire application by letting Docker Compose read the file:

Execute a Docker Compose file with the **up** command

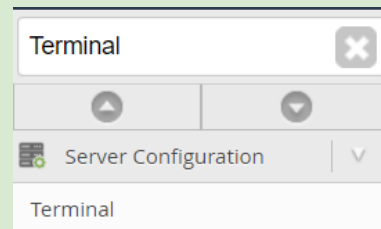
```
docker-compose up -d
```

Installing BioCAsE with Docker

With the Docker knowledge we can install BioCAsE. This process has been simplified by the developers of BioCAsE who have created a **Make file**. A Make file is very useful because it creates a set of rules that apply to the local environment where it is executed. These rules can be used to set-up the Docker installation of BioCAsE by using a single command. The rules also create two extra command opportunities for updating and removing the installation. Before executing the Make command, make sure you have installed Docker on the webserver. To note: we will be using port 8080 as extended from 80 to run the application.

Install BioCAsE with the Make file in the WHM interface:

1. Open the terminal by searching and clicking onto 'Terminal'



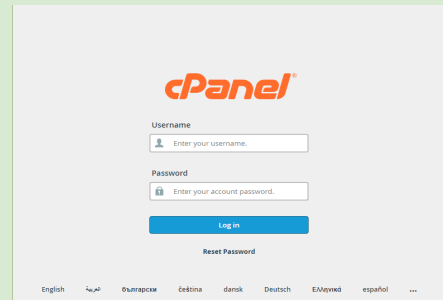
2. Navigate to the public repository of the cPanel user that is linked to the registered domain, in the case of Naturalis the user is called: biouser on the domain of biocase.nlbif.nl. Enter the given commands one after another

```
cd home  
  
cd biouser  
  
cd public_html
```

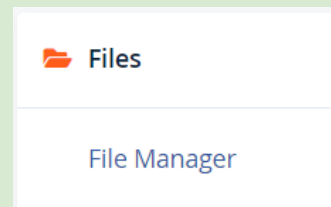
3. Get the Make file by using the following command

```
wget http://ww2.biocase.org/svn/bps2/trunk/Makefile
```

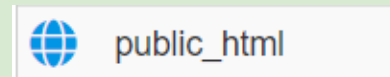
4. Open and login to the cPanel interface at port 2083 in a new browser tab (do not close the WHM interface just yet)



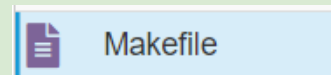
5. Find and click on 'file manager' in the 'Files' tab. This will open another new browser tab



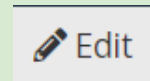
6. Select 'public_html' from the cPanel user's directory



7. Select the Make file



8. Press 'Edit' to open the file in the online editor



9. Change the port variable to 8080

PORT=8080

10. Change the container variable to a relevant name (biocase is the default)

CONTAINER=biocase_test

11. Save by pressing 'save changes'

Save Changes

12. Return to the terminal in the WHM interface

13. Type in the following command to install and start the BioCASE installation:

make install

With this last command, the Make file will create a container with the given name containing the BioCASE installation and run it on port 8080 to the extent of 80. To check if your installation is running, navigate to your domain name and add the port, followed by the BioCASE extension, as for example: biocase.naturalis.nl:8080/biocase. You can also check the container's status by using the Docker command for displaying all containers:

Check the status of a Docker container with the **ps** command

docker ps -a

Using the Make file to create the container is not of harm to Docker's Natural commands such as: start, stop, update or remove. However, the Make file itself does also provide these methods.

Start BioCAsE with the <i>make start</i> command	make start
Stop BioCAsE with the <i>make stop</i> command	make stop
Update BioCAsE with the <i>make update</i> command	make update
Remove BioCAsE with the <i>make remove</i> command	make remove

Now BioCAsE is installed, navigate through the interface to get some feeling of the application. One more thing we want to do before the next step is to change the default password. A password is required to login to BioCAsE's managing system. The default password that comes with every installation is: ACDC. Without changing this, everyone who knows of BioCAsE could login and change settings or data sources.

Change the BioCAsE password:

1. Navigate to your BioCAsE installation (for example: biocase.nlbif.nl:8080/biocase)	biocase.nlbif.nl:8080/biocase/
2. Click on 'Config Tool'	Config Tool Configure new datasources, general options, the querytool, statistics, etc.
3. Now click on 'System Administration'	System administration Configure the general system options and all datasources
4. Fill in the default password: ACDC	Please authenticate with your password: <input type="password" value="...."/> <input type="button" value="Submit"/>
5. Change the password to something strong and memorable	Admin password: <input type="password" value="....."/>
6. Press on 'Update config' to save the settings	<input type="button" value="Update config"/>

Now you can login with your new password. Do not forget to document the password in a safe place.

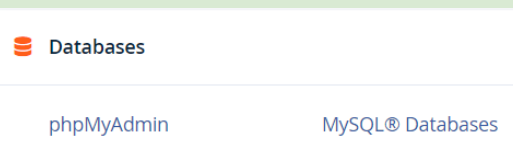
To exchange data with GeoCASE it is required to set-up an online database containing the data related to the geological collection specimens. Exchanging data with the GeoCASE portal depends on a BioCASE software installation (**Chapter C**) on a webserver (**Chapter B**) that connects to the online database. This chapter describes how a dataset can be imported as a database table in a MySQL online database.

When making use of Naturalis services we encourage you to read the paragraphs on the data transformation to MySQL database tables. It describes the most efficient way that your data can be handled and accepts different data types including CSV, MS Excel, JSON, text and MySQL table exports. As an example we will be using a DarwinCore to CSV. When your data is transformed, Naturalis will store it in the MySQL database on the biocase.nl/bif.nl server.

- Institutions,
- Collections,
- Collection metadata
- Occurrence (specimens)

To create the online database with the tables, we need to start MySQL on the webserver. The easiest way is by using a graphical interface, e.g. the cPanel interface. We also use phpMyAdmin, which is an inbuilt database management tool. PhpMyAdmin allows us to execute otherwise complicated tasks on the command line via an easy to understand interface.

- 2. Scroll down to the databases tab and click on 'MySQL Databases'**



3. Now, create a new database with a relevant name (rather not use captions, spaces, etc.). Fill in the name and click 'Create Database' to finish

We will now create the three tables. Three because the occurrences table will be created later using the specimen dataset. We provide two ways to create a table: a manual and automated one. To manually create a table, follow the instructions below. To automatically create the tables, use the database file that is enclosed with this manual. In phpMyAdmin, select the database followed by import. Upload the database file and select SQL as its format. Press 'go' and if correct, the three tables should exist by now. If this method worked, you can skip to the next paragraph.

For the manual creation of a table, you need to provide the required datatable variables.

1. Open phpMyAdmin through the cPanel interface on port 2083

2. Select the created database

3. Create a new table by selecting 'New'. This will be done three times to create three tables

4. Fill in the table names at the top

- Meta
- Sources
- Source_institutions

(Example uses the Meta table)

5. Add the required variables. Pay attention to the:

- Name (name of the variable)
- Type (datatype of the variable)
- Length/Values (length of the variable)
- Default (default value of the variable)
- Null (empty value of the variable)
- A_I (auto increment value of

the variable)

See **figure #** (meta table), **figure #** (Sources table) and **figure #** (Source_institutions table) for reference.

Click 'Go' to save the table.



Meta:

Name	Type	Length/Values	Default	Collation	Attributes	Null	A..I
id	INT		None			<input type="checkbox"/>	<input checked="" type="checkbox"/>
sourceId	VARCHAR	30	NULL	utf8mb4_0900_ai		<input checked="" type="checkbox"/>	<input type="checkbox"/>
individualName	VARCHAR	50	None	utf8mb4_0900_ai		<input type="checkbox"/>	<input type="checkbox"/>
description	TEXT		None	utf8mb4_0900_ai		<input type="checkbox"/>	<input type="checkbox"/>
technicalContact	VARCHAR	50	None	utf8mb4_0900_ai		<input type="checkbox"/>	<input type="checkbox"/>
electronicMail	VARCHAR	60	None	utf8mb4_0900_ai		<input type="checkbox"/>	<input type="checkbox"/>
language	VARCHAR	20	None	utf8mb4_0900_ai		<input type="checkbox"/>	<input type="checkbox"/>
title	VARCHAR	50	None	utf8mb4_0900_ai		<input type="checkbox"/>	<input type="checkbox"/>
dateModified	DATETIME		CURRENT_TIME			<input type="checkbox"/>	<input type="checkbox"/>

Figure #: Required variables in the meta table.

Sources:

Name	Type	Length/Values	Default	Collation	Attributes	Null	A..I
id	VARCHAR	30	None	utf8mb4_0900_ai		<input type="checkbox"/>	<input type="checkbox"/>
RORId	VARCHAR	20	NULL	utf8mb4_0900_ai		<input checked="" type="checkbox"/>	<input type="checkbox"/>

Figure #: Required variables in the sources table.

Source_institutions:

Name	Type	Length/Values	Default	Collation	Attributes	Null	A..I
RORId	VARCHAR	30	None	utf8mb4_0900_ai		<input type="checkbox"/>	<input type="checkbox"/>
title	VARCHAR	50	None	utf8mb4_0900_ai		<input type="checkbox"/>	<input type="checkbox"/>

Figure #: Required variables in the source_institutions table.

Preparing the dataset

Before you upload data to the online database and create the occurrences table, it is important to prepare the geological dataset you want to mobilise. Here the Petrology dataset of Naturalis is used as an example. This dataset is a DarwinCore text file export from the Naturalis Collection Management System. Because plain text files are not supported by phpMyAdmin the data needs to be transformed to CSV format. This can be done by using Microsoft Excel.

1. Open Microsoft Excel



2. Click on open file and select the DarwinCore *.txt file (make text files visible in MS Excel by checking 'show all files')

Occurrence Alle bestanden
Extra Openen Annuleren

3. In the opening editor, start importing rows at line one (image translates to: start import at row: 1)

Importeren starten bij rij: 1

4. Check 'my files contain headers' (image translation) and proceed to the next step

☒ Mijn gegevens bevatten kopteksten.

5. Depending on the type of separator used in the text file, check the separator, Naturalis uses a comma as the default separator (you can recognize the correct separator if Excel shows the rows separated by lines in the preview image).

Scheidingstekens

☒ Tab
☒ Puntkomma
☐ Komma
☐ Spatie
☐ Overige:

catalogNumber	basisOfRecord	class	collectionCode
RGM.1340010	OtherSpecimen		Mineralogy
RGM.999	OtherSpecimen		Mineralogy
RGM.99999	OtherSpecimen		Petrology
RGM.99961.0	OtherSpecimen		Petrology
RGM.99960	OtherSpecimen		Mineralogy

6. Press proceed and at last complete to convert to a Excel file (image translates to: complete)

Voltooien

7. Save the file by default and choose 'CSV UTF-8 (comma separated)' as the file type. The image translates to: CSV (list separator).

petrology_occurrences.csv

CSV (gescheiden door lijstscheidingstekens) (*.csv)

Now it is time to create the fourth table: the occurrences table, based on the CSV file.

1. Open phpMyAdmin in the cPanel interface

Databases

phpMyAdmin

2. Select the created database

☒ biouser_biocase
☐ information_schema

3. Choose 'Import' from the above tab row



4. Press 'Choose file' (image translation) and select the CSV file we generated.

Bestand kiezen

5. Make sure the import starts at row zero since this contains our headers

Skip this number of queries (for SQL) starting from the first one:

6. Check if the format is set to CSV, if not select CSV

Format:

CSV

7. Define the separator you used for creating the CSV, in the case of Naturalis a point comma (;)

Columns separated with:



8. Check the option that declares that our first line contains headers and thus the names of the fields

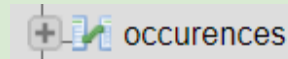
☒ The first line of the file contains the table column names

9. Click 'Go' and wait till the program is finished (this can take a few minutes depending on the size of the dataset)

Go

The table is now created and added to the online database. Change the name of the table to a descriptive name, e.g. petrology_occurrences.

1. Select the database and then the just created table to rename (will be named after your original datafile)



2. Click on 'Operations'

Operations

3. Fill in the new name at 'Rename table to'

Table options

Rename table to

petrologyOccurrences

4. Press 'Go' to save the name

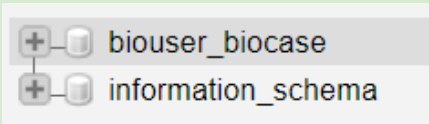
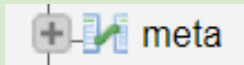

Go

If the upload process of the CSV file fails, try to identify the error phpMyAdmin returns. A possible reason could be a limited timeout. This means the process of uploading is taking longer than the maximum allowed time frame for executing processes. Another reason could be that a wrong separator was given. Check if you used the correct separator and retry the upload. When reuploading

check If the headers of the database were still created. If so, start on line one and do not check the option that declares the first line contains headers. If your are still commencing errors, please contact the contact person of Naturalis.

Inserting metadata

Now all required tables are created it is now possible to add the meta data. To add data to the Meta, Sources and Source_institutions tables you use the insert function. This function can be found by selecting the database in phpMyAdmin and activating the 'Insert' tab (Fig. #). Fill in the fields with the corresponding information. Examples of the Naturalis tables are shown below. Important to notice are the id' which are used to create relations later on. As for example the 'sourcelid' refers from the Meta to the Sources table; and the 'RORid' refers from the Sources to the Source_institutions table.

1. Select the database	
2. Select the Meta, Sources and Source_institutions tables, one for one (Meta table as example in image)	
3. Insert the required variables as visualised per table in the images below	

Meta:

Column	Type	Function	Null	Value
id	int			1
sourcelid	varchar(30)		<input type="checkbox"/>	Petrology - Petrology
individualName	varchar(50)			Curator Petrology
description	text			Petrology collection of Naturalis Biodiversity Center
technicalContact	varchar(50)			Naturalis Biodiversity Center
electronicMail	varchar(60)			frontofficecollectie@naturalis.nl
language	varchar(20)			en
title	varchar(50)			Naturalis Mineralogy Collection
dateModified	datetime			2021-11-16 00:00:00

[Go](#)

Figure #: Insert content to the Meta table.

Sources:

Column	Type	Function	Null	Value
id	varchar(30)			Petrology
RORId	varchar(20)		<input type="checkbox"/>	0566bfb96 - 0566bfb96

Figure #: Insert content to the Sources table

Source_institutions:

Column	Type	Function	Null	Value
RORId	varchar(30)			0566bfb96
title	varchar(50)			Naturalis Biodiversity Center

Figure #: Insert content to the Source_institutions table

Defining relations between database tables

The last step in preparing the database is creating the connections between the tables. These so-called relations are defined by foreign key constraints. A constraint is constructed by linking a certain field of one table to an unique field of another table. Unique fields are often identifiers as id' like the ROR identifier. An example of a relationship is the connection between the Sources and Source_institutions tables. In this example the ROR identifier value of the Sources table can be linked to the unique ROR identifier value of the Source_institutions table which indicates that the Petrology collection belongs to the Naturalis institution. The relations are an integral part of databases and are essential for BioCASE to function.

In the following section you will create relations between the Meta, Sources and Occurrences tables. The metadata and occurrences both belong to a collection and should refer to the collection by using the collection code, or something similar. As described above, the Sources need to link to the Source_institutions. A relationship between database tables is defined by selecting the database, then a table and clicking on 'Structure'.

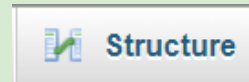
1. Select the database	<pre>graph TD; A[+] --> B[biouser_biocase]; B --> C[+] ; C --> D[information_schema];</pre>
2. Select the Meta, Occurrences and Sources table one per one (Meta	<pre>graph TD; A[+] --> B[meta];</pre>

table as example in the image)

3. Click on 'Structure'

4. Now click on 'Relation view'

5. Take over the values as shown in **figure #** (Meta table), **figure #** (Occurrences table) and **figure #** (Sources table). Note that the database and table fields could be different in your database due to alternate naming



Column	Foreign key constraint
	Database
sourceId	biouser_biocase

(INNODB)

Table	Column
sources	id

Meta:

Foreign key constraints	
Actions	Constraint properties
Drop	meta_ibfk_1
ON DELETE	SET NULL
ON UPDATE	CASCADE

Column	Foreign key constraint (INNODB)
	Database
sourceId	biouser_biocase

+ Add column

Figure #: Relations of the meta table

Occurrences (specimens):

Foreign key constraints	
Actions	Constraint properties
Drop	occurrences_ibfk_1
ON DELETE	SET NULL
ON UPDATE	CASCADE

Column	Foreign key constraint (INNODB)
	Database
collectionCode	biouser_biocase

+ Add column

Figure #: Relations of the occurrences (specimens) table

Sources:

Foreign key constraints	
Actions	Constraint properties
Drop	sources_ibfk_1
ON DELETE	SET NULL
ON UPDATE	CASCADE

Column	Foreign key constraint (INNODB)
	Database
RORId	biouser_biocase

+ Add column

Figure #: Relations of the sources table

Additional datasets

Now that the relationships between the tables of our MySQL database are defined it is suited for use by the BioCASE software. In the case of a new dataset, e.g. a Mineralogy collection, this will have to be defined separately in a second Occurrences table. This table will be created in the same manner the occurrences table was created for the Petrology collection, including the definition of relations. Additionally, a new record has to be added to the Meta table that describes the metadata for the Mineralogy collection. Also a new record in the Sources table would be created representing the

Mineralogy collection. No changes need to be made to the Source_institutions tables, unless a collection from a different institution will be added.

E: Installing and using the BioCASE Provider Software

After all preparations we can finally install and use the BioCASE Provider Software to mobilise geological data to GeoCASE. This process contains some specific settings that will be discussed to optimise the usefulness of the data. Once installed and configured, we will be able to register our BioCASE instance at GeoCASE and publish the data online.

Naturalis server

When making use of the Naturalis service, it is still mandatory to create a new data source in BioCASE by adding a new Occurrences table to the database as shown in **Chapter D**. Also, the configuration of this dataset and mapping process needs to be done in BioCASE. The configuration of a data source in BioCASE can be done relatively easily and could be a good experience on how to handle your data in this online environment. As Naturalis will be making use of the existing server and BioCASE installation, it is not necessary to install a new variant of the BioCASE software on a different location.

We would recommend reading the paragraphs about: *data source set up*, *table configuration* and *data mapping*.

The purpose of BioCASE

BioCASE and its software called the BioCASE Provider Software, is software which creates an access point to online database services from where data can be requested. Much like **API's**, certain commands can be given to request data via a **XML** protocol. XML (Extensible Markup Language) is a type of structured **syntax** that can easily be read by both humans and computers. BioCASE supports a number of different data standards in which data can be requested that are widely recognized in the biological and geological communities, e.g. DarwinCore and ABCD ([Access to Biological Collection Data](#)) with its EFG ([Extension for Geosciences](#)) extension. By using BioCASE, geological collections can be shared in the public domain through a recognized data standard in an usable online endpoint. At present the use of BioCASE is a requirement when sharing data with the GeoCASE portal.

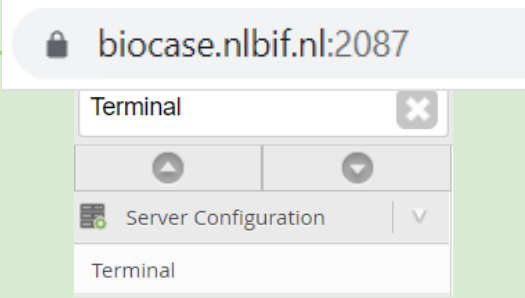
How it works

BioCASE is a so-called **PyWrapper, a software** package that allows primary biodiversity/geology data publishing from any arbitrarily structured database. BioCASE makes use of a database to create the access point to request data. The process of installing BioCASE is described in **Chapter C**. Setting up an online MySQL database is described in **Chapter D**. The next step is to configure BioCASE. First, a connection with the database needs to be made which requires the creation of a database user with admin privileges. Secondly, the database tables can be configured in BioCASE by for instance defining the primary keys and relations between tables. A primary key identifies an unique record in a database table. The 'RORid' for example is the primary identifier variable for institutions in the Source_institutions table. Lastly, the data should be mapped to a data standard. Because GeoCASE is an Earth Science Collections Portal we will focus on the ABCD-EFG standard

Data source set up

With the BioCASE software installed on the webserver, as described in **Chapter C**, we can now set up a data source containing the collection data. The source, as defined in **Chapter D**, consists of a database with four tables. In order to use this database, it is mandatory to create a MySQL user account with the reading privileges. This MySQL user will be able to login to the database from the

BioCAsE software and consult the stored data. In the following example we use the method that works for MySQL. If you are using a different database type, please look up an equivalent method.

<ol style="list-style-type: none">1. Login to the WHM portal on port 20872. Search and click on 'Terminal'	
<ol style="list-style-type: none">3. Start MySQL in the terminal with typing: 'mysql'4. Use the following command to create a new user, specifically for BioCAsE. Think of a relevant name and password; and make sure to document these in a safe place.	<pre>mysql CREATE USER 'username'@% IDENTIFIED BY 'password';</pre>

Worthwhile mentioning is the use of the percentage sign, this indicates that the MySQL user has access to all sections of the server. If specified as localhost for example, this could clash with our BioCAsE software. Because BioCAsE runs on port 8080 extended as 80, we would not be able to connect to MySQL and use the source data.

If correct, a user is made. To check this fact, use the following command in the MySQL terminal to list all MySQL users. If your user is not present, try to add the user again.

Select all users from within MySQL with the select command	<pre>mysql select user from mysql.user;</pre>
--	--

To finish the task, we give the user reading privileges to be able to read out the database.

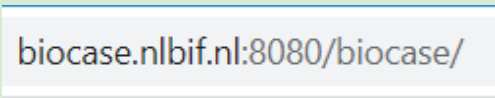
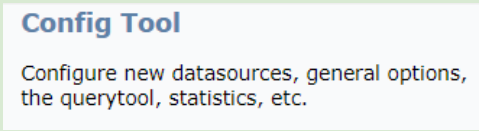
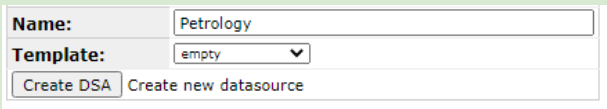
<ol style="list-style-type: none">1. Again, open the MySQL terminal by typing 'mysql' in the WHM terminal	<pre>mysql</pre>
<ol style="list-style-type: none">2. Select the database, the one we created in Chapter C	<pre>use <database name></pre>
<ol style="list-style-type: none">3. Use the following command to grant read privileges on all tables to the MySQL user	<pre>GRANT SELECT ON * . * TO 'username'@%;</pre>

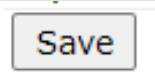

If SELECT eventually does not work with the BioCASE software, check if the user will function with a grant on all privileges. This could be harmful to the database by certain applications. However, it is stated in the BioCASE documentation that it only reads the database and could therefore be trusted (). Use this command to grant a user all privileges:

```
GRANT ALL PRIVILEGES ON * . * TO 'username'@%;
```

With our freshly created user account it is possible to connect BioCASE with the MySQL database. This process requires the user account information, the MySQL database name, and the IP address of the server domain. The IP address is often found in the initial confirmation email of the set up of your webserver, or simply login to the WHM portal and open the terminal. The IP address is displayed on top (see fig. x; here the IP address is 87.253.148.112).

Connect BioCASE to the MySQL database:

1. Open BioCASE at you domain, e.g.: http://87.253.148.112:8080/biocase/ or http://biocase.nlbif.nl:8080/biocase	
2. Click on 'Config tool'	
3. Now click on 'System administration' and login with your BioCASE password	
4. Add a new data source and give it a name, e.g. 'Petrology' for a petrology collection, and click on 'Create DSA'.	
5. Click on 'Edit connection parameters'	
6. Fill in the correct details as we have defined in earlier steps: <ul style="list-style-type: none">• DBMS → mysql• Host → IP address of the webserver• Database → Name of the database• User → Username of the MySQL user	

<ul style="list-style-type: none"> • Password → Password of the MySQL user • Encoding → The encoding type, set to utf-8 per default 	
7. Click 'Save' to save the configuration	
8. If the connection is successful, the red painted 'no connection' will transform to a green 'OK'.	<div> Database connection <p>The connection to the database is OK.  Edit Connection parameters</p> </div>

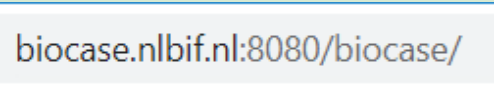

After clicking on 'Save' the MySQL database connection parameters are saved in the configuration file of BioCAsE, meaning that they will be preserved on a new start-up of the Docker container. BioCAsE will also try to connect with the database.

If the connection fails, review your connection parameters on potential typing errors. Also try to give the MySQL users overall privileges as stated in the previous error section. Make sure to contact the contact person of Naturalis if this still does not work.

Database table configuration

The second step is to configure our database tables in the BioCAsE Provider Software. This is a requirement because BioCAsE will use these definitions to query the database. This is why the relationships between tables in the MySQL database need to be correctly defined as described in **Chapter D**. The configuration of the relationships between the database tables is initiated with the set up of the BioCAsE connection and represents a one to one copy from the MySQL database. A few elements need further definition: primary keys, foreign keys and variable types.

Configuring the database tables in BioCAsE:

1. Open BioCAsE at your domain, as for example: biocase.nlbif.nl:8080/biocase/	
2. Click on the data source you added in the previous step, for example a Petrology dataset	<div> DataSources <p>Each data source in a BioCAsE ser</p> <ul style="list-style-type: none"> • demo • Mineralogy • Petrology </div>
3. Next click on 'Edit DB structure' (login if necessary)	<div> Database Structure <p>You have configured 5 table aliases.  Edit DB structure</p> </div>

This page shows the entire configuration of the database. Figure x shows a table in which the elements can be defined. After configuration of the table relationships between the Primary keys and the Foreign keys, BioCAsE renders an image of the database structure (Figure x).

4. Add the tables by selecting the name from the drop down menu and clicking 'Add'

New alias	petrologyOccurrences	petrologyOccurrences
Add		

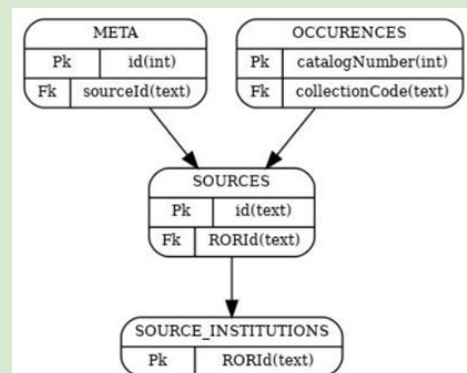
5. Define the primary key (i.e. unique record Id) for each table. Use the example as an indicator (Fig. x). Mind the field types; e.g. the catalogNumber of records in the 'occurrences' table is alphanumeric hence 'text'

catalogNumber	text	
New Primary Key		
--new--	integer	Add

6. Define the foreign keys, as in relations, from each table to the other tables (Fig x)

FK to alias	
sources	
collectionCode	text
New Attribute	
--new--	integer
Add	
new FK to alias	
--none--	Add

7. Check if the rendered image complies with the example figure. Names may differ based on your dataset, but types should be similar)



8. Click 'Save' to save the table configuration

Save

	Table/View	Alias	Primary Key Attribute(s)	Foreign Key(s)
Delete alias	meta	meta	<div>id integer </div> <div>New Primary Key</div> <div>--new-- integer </div> <div>Add</div>	<div>FK to alias</div> <div>sources</div> <div>sourceId text </div> <div>New Attribute</div> <div>--new-- integer </div> <div>Add</div> <div>new FK to alias</div> <div>--none-- Add</div>
Delete alias	occurrences	occurrences	<div>catalogNumber text </div> <div>New Primary Key</div> <div>--new-- integer </div> <div>Add</div>	<div>FK to alias</div> <div>sources</div> <div>collectionCode text </div> <div>New Attribute</div> <div>--new-- integer </div> <div>Add</div> <div>new FK to alias</div> <div>--none-- Add</div>
Delete alias	source_institutions	source_institutions	<div>RORid text </div> <div>New Primary Key</div> <div>--new-- integer </div> <div>Add</div>	<div>new FK to alias</div> <div>--none-- Add</div>
Delete alias	sources	sources	<div>id text </div> <div>New Primary Key</div> <div>--new-- integer </div> <div>Add</div>	<div>FK to alias</div> <div>source_institutions</div> <div>RORid text </div> <div>New Attribute</div> <div>--new-- integer </div> <div>Add</div> <div>new FK to alias</div> <div>--none-- Add</div>

Figure #: Table configuration in BioCASE

Mapping data to the ABCD-EFG data standard

The last step of this chapter describes the mapping process that the BioCASE Provider Software offers. This process declares the data fields of the chosen data standard. As such, we create connections between the database fields and the data standard. Once the database fields are mapped to the data standard the data can be exported from BioCASE to XML documents.

First, a data standard has to be selected. We use the ABCD-EFG standard as this standard is widely recognized in the geological community and is used by the GeoCASE portal. The ABCD-EFG data standard proposes a set of mandatory elements which were prepared in **Chapter D** when the database tables were created. The other fields in the dataset, which are a lot, are made up of optional fields. However, when defining certain variables like the licence type for the object, some additional variables are mandatory like 'language'. Required fields are marked in red.

Mapping required fields for using the ABCD-EFG data standard:

1. Open BioCASE at you domain, e.g. biocase.nlbif.nl:8080/biocase

biocase.nlbif.nl:8080/biocase/

2. Click on your configured dataset

3. In the schemas section, select ABCDEFG_2.06.xml from the dropdown menu (Fig. x) and click 'Create'

DataSources

Each data source in a BioCASE ser

- demo
- Mineralogy
- Petrology

Schemas

These are the XML schemas supported by this datasource and the number of
Click on the name to edit the mapping:

Map your DB to a new schema:

Recycle bin: Restore a schema ma a, or press
Recycle bin is empty.

DarwinCore Archives

No DarwinCore archives created so
You can create, remove or update archive page

--select--
--select--
ABCDDNA_2.06.xml
ABCDEFG_2.06.xml
ABCD_1.20.xml
ABCD_2.06.xml
AfricanTypes_2.00.xml
BioCASE-MetaProfile_1.24.xml
DBSynchron_1.00.xml
DarwinCoreExtQueryTool_2.xml
DarwinCore_2.xml
HISPID_5.xml
LIDO.xml
LIDO_gml.xml
MCPD.xml
SPICE_1.3.xml
TCS_1.01.xml
gcp_passport_1.04.xml

This will open the mapping section displaying all the premature fields of the ABCD-EFG standard. Now, link the fields, that are marked red, to their counterparts in the database. We will be showing an example of the title field from the metadata.

1. Click on the plus icon next to 'title', this will open a new window

2. In this window on the first row, select the 'metadata table' and then the 'title' field. Also make sure to set the field 'type correspondent' to the database, in this case to text

3. Press 'Save' to save the mapping. This will close the window and reload the page displaying BioCASE

- Metadata
- Description
- Representation
@language +
Title +

Literal 1
DB Col1 meta title text

Save

Congratulations, you have just mapped your first element! If correct the overview page of the mapping will display this change. The email field will have turned green and the value we bound from the meta table will be displayed. Now repeat the above steps for all the visible, red fields. In the end the result should look something like **figure #**.

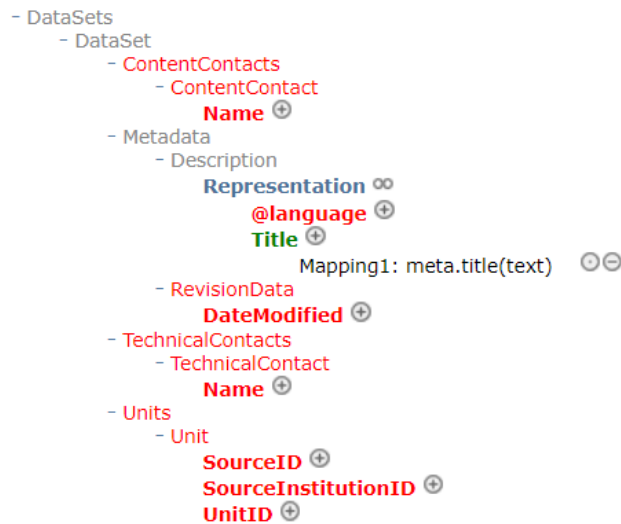


Figure #: Prerequisite ABCD-EFG fields

With these fields filled we are able to create the most simplest export of our data. This is a good time to test the basic fields because we have used all four base tables from the database. If this test succeeds, it is right to assume our database configuration functions correct and we can proceed with the mapping. To execute the test we will be using the manual query form that is built into BioCAsE. To use this query form we will first need to define our endpoint. This endpoint represents the URL from which data can be retrieved by using the BioCAsE protocol.

Configure the endpoint:

1. Open the system administration section from the home menu of BioCAsE
2. In the server configuration section, change the webserver domain to the domain of your server installation as for example:
`http://biocase.nlbif.nl:8080`

System administration
 Configure the general system options and all datasources

Server Configuration

Webserver domain:

With our endpoint correctly defined we can execute a request via the BioCAsE protocol and test our initial mapping.

Test our mapping:

1. Click on your configured datasource

DataSources
 Each data source in a BioCAsE server

- demo
- Mineralogy
- Petrology

2. Now click on the ABCDEFG_2.06.xml schema

Schemas

These are the XML schem
Click on the name to edit

ABCDEFG_2.06.xml :

2. From the mapping overview page of the data source in BioCAsE, click on 'test mapping'

Test Mapping!

3. From the templates menu, select 'ABCD2 search'

Replace form with templates for a :
ABCD scan, ABCD search, ABCD2 scan, ABCD2 search,

4. Remove the filter portion of the statement

```
<?xml version='1.0' encoding='UTF-8'?>
<request xmlns='http://www.biocase.org/schemas/protocol/1.3'>
  <header><type>search</type></header>
  <search>
    <requestFormat>http://www.tdwg.org/schemas/abcd/2.06</requestFormat>
    <responseFormat start='0' limit='10'>http://www.tdwg.org/schemas/abcd/2.06</responseFormat>
    <filter>
      <like path='/DataSets/DataSet/Units/Unit/Identifications/Identification/Result/TaxonIdentified/Scienti'>
        </filters>
      </filter>
    <count>false</count>
  </search>
</request>
```

5. Make sure the wrapper option contains the endpoint of your server, with the addition of the PyWrapper and data source statement

Wrapper:

http://biocase.nlbif.nl:8080/biocase/pywrapper.cgi?dsa=Petrology

6. Press 'submit' to execute the query

Submit

After processing the query, BioCAsE will display its response. This response should contain the fields that you defined in the initial mapping. If the result contains errors, read why the error is given and try to look this up in the mapping, database configuration or endpoint definition.

Furthermore, warnings indicate things that did not process well such as mapped fields which are empty in the database. They are not mandatory and therefore do not stop the protocol from working. The return script from BioCAsE is pretty readable for humans, please try to fix as many of the warnings as possible.

When satisfied with the initial result, you can continue to map more fields out of the broad selection the ABCD-EFG standard has to offer. The method for mapping stays the same, however, you will need to enable the visibleness of these fields. To set an example we will be mapping another field. We have chosen to map the scientific name, because this is an often recurring field.

Map the scientific name:

1. Click on your configured datasource

DataSources

Each data source in a BioCAsE ser

- demo
- Mineralogy
- Petrology

2. Navigate to the mapping overview page in BioCAsE from the schemas section

Schemas

These are the XML schem
Click on the name to edit

ABCDEFG_2.06.xml :

3. Check the checkbox that says 'show all concepts'. This will reload the page and show all the fields of the ABCD-EFG standard

Show all concepts ☒

4. Press on CTRL and F on your keyboard to initiate your browsers search function

5. Type and search for 'scientificname'. This should give multiple hits, each for a different kind of natural group. We want to map the scientific name for a geological collection.

scientificname 1/9

6. Filter through the search results until you find a scientific name element called: 'FullScientificNameString' in the category of 'MineralRockIdentified'

- MineralRockIdentified
 Certainty
 - ClassifiedName
 FullScientificNameString
 - MineralRockNameAtomised
 AuthorTeamAndYear
 MineralRockClassification
 ScientificNameString

7. Press the plus button next to 'FullScientificNameString'

- MineralRockIdentified
 Certainty
 - ClassifiedName
 FullScientificNameString

8. Bind the value of the occurrences (specimens) table to the field

Literal 1
DB Col1 petrologyOccurrences scientificName text

9. Press 'save' to save the mapping

Save

Now we have mapped the scientific name to the correct equivalent of the ABCD-EFG standard. Bear in mind that the sorting process is very important. You could want to be mapping the scientific name for geological purposes to 'MineralRockIdentified', but end up mapping it to 'Mycology'. It is important to scan the names of the groups you map to and ask for someone to review the mapping (especially if you lack knowledge of the biological and geological terms).

Repeat the mapping process for all the fields you can or want to map. *Chapter G* offers a very useful list of the most used fields for mapping geological data to the GeoCAsE portal. Use this list to check the quality of the data you are going to submit.

F: Mobilising data to GeoCAsE

The last step of this manual is the mobilisation of geological data to the GeoCAsE portal. This step is relatively short but of most importance for making the data available for biodiversity scientists.

Naturalis service

When using the Naturalis service, it is best to send an email to GeoCAsE together with the contact person of Naturalis. It is important to discuss the way of handling and presenting data in GeoCAsE, such as incorporating the data under the name of Naturalis or keeping it bound to a separate institution. Since Naturalis' server already is registered at GeoCAsE, it would only be necessary to include new details as the endpoint of the data source.

What is GeoCAsE?

GeoCAsE, which is short for: Geoscience Collection Access Service, is a data network including an internet portal that is developed for creating an universal point where geological data can be found by biodiversity scientists and other interested parties. From this point of view it has a similar goal to DiSSCo, but focused on the geological aspect of data from institutions. GeoCAsE will someday be a part of the DiSSCo network. Hence, why it is a useful step to mobilise data to GeoCAsE as a preparation for the future infrastructure that can request this data.

Mobilising data to GeoCAsE can be done using a XML based tool that transforms data to a readable data format. ABCD-EFG and DarwinCore are both supported (). As mentioned earlier we will be using ABCD-EFG, as this is a widely accepted data standard in the geological community. With the preparations of all previous chapters, we are able to transform our source data to ABCD-EFG and enable GeoCAsE to pull this data.

Publishing data to GeoCAsE

The method for publishing data to GeoCAsE is probably the most simple we have discussed yet. We simply have to communicate the URL of the endpoint of the BioCAsE installation to the people who manage the GeoCAsE portal. The organisation who manages GeoCAsE is called TalTech, also known as the Technical University of Tallinn. They manage GeoCAsE via a back-end tool which also can be used to add new collections to the portal. By providing the endpoint URL of the BioCAsE installation, TalTech will be able to pull the source data via a similar request as we have seen in *Chapter E* when using the query tool. The request will activate the BioCAsE protocol and let it generate a response. This response will be used to generate the data pages in the portal.

To send the endpoint URL, you can make use of regular email. Send an email containing your information, such as the institution name and include the endpoint URL to the following address: geocase@mfn.berlin. This address refers to a German region. This is because the project originates in Berlin. After receiving the email the people behind GeoCAsE will get in contact with you and handle your data. Suggestions could be for improving the data mapping or providing additional information regarding your institution.

If everything worked, congratulations! Your data should be available at the GeoCAsE portal. Navigate to geocase.org, choose search and filter on your institution's or data's details to find the data.

G: Useful extras

Summary of possible occurring errors

xxx

MIDS for mapping geological data

xxx

Multiple images

xxx