

---

# **Local EnteroBase**

**Local EnteroBase Developer**

**Sep 15, 2020**



**CONTENTS:**

<b>1</b>	<b>Installation</b>	<b>3</b>
1.1	Automatic installation . . . . .	3
1.2	Manual Installation . . . . .	4
<b>2</b>	<b>Local EnteroBase Registration</b>	<b>9</b>
2.1	System Configuration . . . . .	9
2.2	Client Registration . . . . .	9
2.3	Files upload test . . . . .	11
<b>3</b>	<b>User Guide</b>	<b>15</b>
3.1	Login using Warwick EnteroBase . . . . .	15
3.2	Upload Single Strain . . . . .	15
3.3	Upload Muliple Strains . . . . .	17
3.4	Jobs Status . . . . .	18
<b>4</b>	<b>Developing Local EnteroBase</b>	<b>21</b>
4.1	Singularity images creation . . . . .	21
<b>5</b>	<b>Indices and tables</b>	<b>29</b>



The main purpose of this version is to enable our partners (users) to assemble their read files locally and send the strain metadata along with the assembly files to Warwick EnteroBase.



## INSTALLATION

### 1.1 Automatic installation

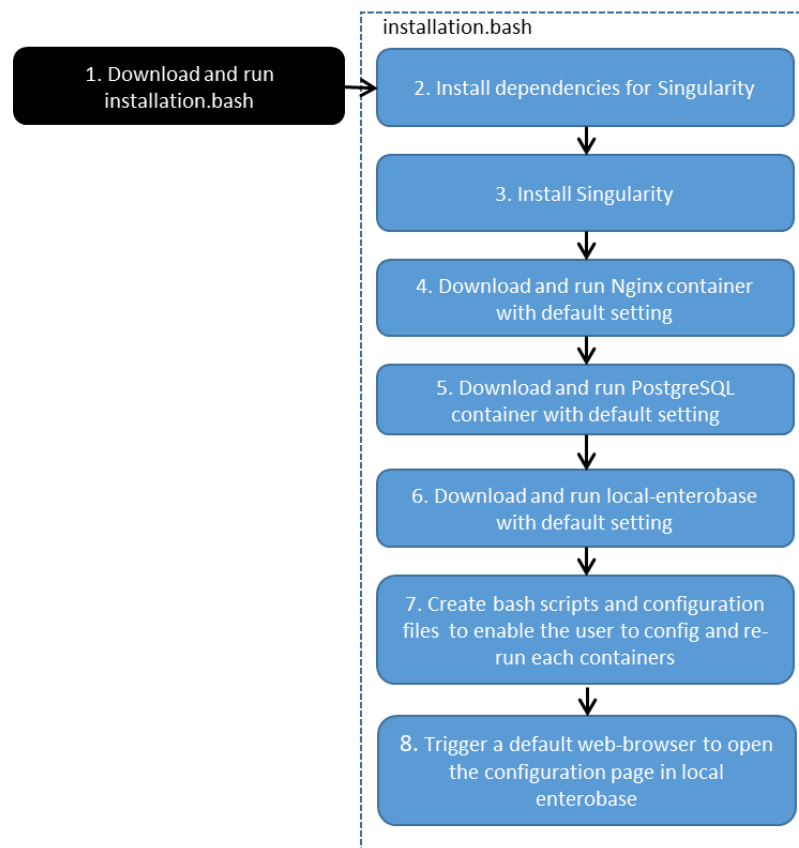
The Installation script “local\_enterobase\_installer.sh” is saved in the “installer” sub-folder.

The script will automatically download, configure and run the system singularity images (i.e. Nginx, PostgreSQL and Unicorn and local EnteroBase) and their dependencies packages. During the installation the user will be asked to:

- Provide his server domain name or IP address
- Set a password for local EnteroBase.

Then system components will be configured using the default parameters and should run without any issue.

The following figure shows the script workflow.



## 1.2 Manual Installation

This section will guide you to download and install the packages which are required to run the local EnteroBase on your machine.

### 1.2.1 Singularity installation

Firstly, the user needs to install singularity software using the following steps as described in “[https://sylabs.io/guides/3.5/user-guide/quick\\_start.html](https://sylabs.io/guides/3.5/user-guide/quick_start.html)”, please note some installation requires the user to have sudo rule, if you do not have, please ask your system administrator to install them for you

#### Install the required packages

- You need to use these two commands to install the dependency packages:

```
sudo apt-get update
sudo apt-get install -y \
build-essential \
libssl-dev \
uuid-dev \
libgpgme11-dev \
squashfs-tools \
libseccomp-dev \
wget \
pkg-config \
git \
cryptsetup
```

#### Install go

- This is used by the singularity container

```
export VERSION=1.13 OS=linux ARCH=amd64
wget https://dl.google.com/go/go$VERSION.$OS-$ARCH.tar.gz
sudo tar -C /usr/local -xzf go$VERSION.$OS-$ARCH.tar.gz
rm go$VERSION.$OS-$ARCH.tar.gz
sudo ln -s /usr/local/go/bin/go /usr/local/bin/go
```

- You can test your download by typing

```
go h
```

#### Download Singularity from a release

- The following command should be used to download the code:

```
export VERSION=3.5.2 && wget https://github.com/sylabs/singularity/releases/
↪download/v${VERSION}/singularity-${VERSION}.tar.gz
tar -xzf singularity-${VERSION}.tar.gz
cd singularity
```



## Compile the Singularity source code

- The code needs to be compiled using the following command:

```
./mconfig
make -C builddir
sudo make -C builddir install
```

- You can test the installation by typing

```
singularity h
```

### 1.2.2 NGINX Web Server

- Pull the nginx singularity image “nginx\_container.sif” from the singularity cloud library using the following command

```
singularity pull --arch amd64 nginx_container.sif library://local_enterobase/
↪default/nginx_local_enterobase:0.1
```

- Please run the following commands to create a folder and copy nginx configuration file (nginx\_local\_enterobase.conf) and certs folder to the created folder.

```
mkdir $HOME/home/nginx_folder
singularity run -B $HOME/nginx_folder:/home/nginx_user --app prep_nginx nginx_
↪container.sif
```

- Then you should alert nginx\_local\_enterobase.conf to set the domain name or IP address in three different places, by replacing (replaced\_by\_your\_server\_uri) string with your actual server uri, e.g. myserver.com.
- In addition, you should provide a valid ssl certificate which should be saved inside the certs folder inside the user's home folder
- In case of your server does not have a valid ssl certificate, you can temporary create a self-signed certificate using the following link.
  - <https://linuxize.com/post/creating-a-self-signed-ssl-certificate/>
- Alternatively, you can use the keys inside certs folder in the user home folder (no modification is needed)
- This should be temporary, as it is important to get a valid ssl certificate from a trusted Certificate Authority.
- To test your configuration, you should run this command:

```
sudo singularity run -B $HOME/nginx_folder:/home/nginx_user --writable-tmpfs --
↪app test_nginx nginx_container.sif
```

- You need to correct any issue which you may get from the previous command, and run the test again.
- You need to run the server using the following command

```
sudo singularity instance start -B $HOME/nginx_folder:/home/nginx_user --
↪writable-tmpfs nginx_container.sif
```

### 1.2.3 PostgreSQL Database server

- Please pull the postgres singularity image file “postgres\_container.sif” from the singularity cloud library using the following command:

```
singularity pull --arch amd64 postgres_container.sif library://local_enterobase/  
↪default/postgres_local_enterobase:0.1
```

- You need to create a folder which will be used to save the databases (for example: \$HOME/postgresql/data) and create another folder (for example \$HOME/postgresqltemp) for the database server temporary use, e.g. postgres\_temp, these two folders will be bonded with related folders inside the postgres container at run time

```
mkdir $HOME/postgresql  
mkdir $HOME/postgresql/data  
mkdir $HOME/postgresql/temp
```

- Then you can run the following command to start up the server

```
SINGULARITYENV_POSTGRES_PASSWORD=local_password singularity instance start -B  
↪$HOME/postgres/data:/var/lib/postgresql/data -B $HOME/postgres/temp:/var/run/  
↪postgresql/ postgres_container.sif postgres -p 5432
```

- Please note that:
  - local\_password (you may use something else), is the database user password for the database user postgres, you will need both of them to configure your local instance
  - The default port number for the database server is 5432, you may change that by replacing 5432 with the new port

### 1.2.4 Gunicorn and the application

- Please pull the singularity image file “local\_enterobase.sif” from the singularity cloud library and save it inside your home folder, you should use this command:

```
singularity pull --arch amd64 local_enterobase.sif library://local_enterobase/  
↪default/local_enterobase:0.1
```

- Please make sure that you have installed and run PostgreSQL container as described before.
- For security reason, you should first set up a system password so you can use the web interface to configure the application, register your client with Warwick EnteroBase and test upload files to Warwick EnteroBase. Use the following command to set up the password after replacing “mypassword” by your own password.

```
singularity run --app set_password local_enterobase.sif -p mypassword
```

- Please note that you may get an error message regarding database configuration when running this command, but you can ignore it at this stage. Then, you should use the following command to run the gunicorn and the application:

```
singularity instance start local_enterobase.sif local_enterobase
```

- To be able to use the application, you need to:
  - Configure the database server which includes Database server URI, Database port number, Database user and Database password.
  - Register your installation with Warwick EnteroBase

- Test uploading 100 files to Warwick EnteroBase
- The local installation configuration file is saved in your home folder (.local\_configuration\_file.yml), you can edit it directly using any text editor (e.g. vim) or it can be alerted using “/update\_system\_configuration” link from the web interface (it will be the default main web page if the database is not configured or not configured correctly).
- After any system configuration change, you need to restart the application using the following commands:

```
singularity instance stop local_enterobase  
singularity instance start local_enterobase.sif local_enterobase
```



## LOCAL ENTEROBASE REGISTRATION

### 2.1 System Configuration

- The local administrator needs to:
  - Configure the database server which includes uri( DATABASE SERVER URI), port (DATABASE PORT), username (DATABASE USER) and password (DATABASE PASSSSWORD)
  - Register your installation with Warwick EnteroBase
  - Test uploading files to Warwick EnteroBase
- In case of using “local\_enterobase\_installer.sh” bash script to install the system, then the script will configure the system with the default setting which should be fine for most of cases.
- The local installation configuration file is saved in the user home folder (\$HOME/.local\_configuration\_file.yml), you can edit it directly using any text editor (e.g. vim) or it can be alerted using “/update\_system\_configuration” link from the web interface (it will be the default main web page if the database is not configured or not configured correctly in case of manual installation).
- After any system configuration change, you need to restart the application using the following commands:

```
singularity instance stop local_enterobase
singularity instance start local_enterobase.sif local_enterobase
```

- If the local administrator has installed it using the provided bash file, it will create a bash script “restart\_local\_enterobase.sh” which can be used to restart the singularity container using the following command:

```
./restart_local_enterobase.sh
```

#### 2.1.1 Notes

- It is needed to synchronize the local installation name with the registration form.
- In case of using the automatic installation, it is needed to get the server URL from the user input during the installation.

### 2.2 Client Registration

- The local administrator needs to submit a request to register his client with Warwick EnteroBase by selecting “Register your Local EnteroBase Website” option from the first web page” or use this link “/register\_client”

**Configuration**

Instance name

Instance icon

Database user

Database server uri

Database port

Database password

Warwick temporary token

Warwick client password

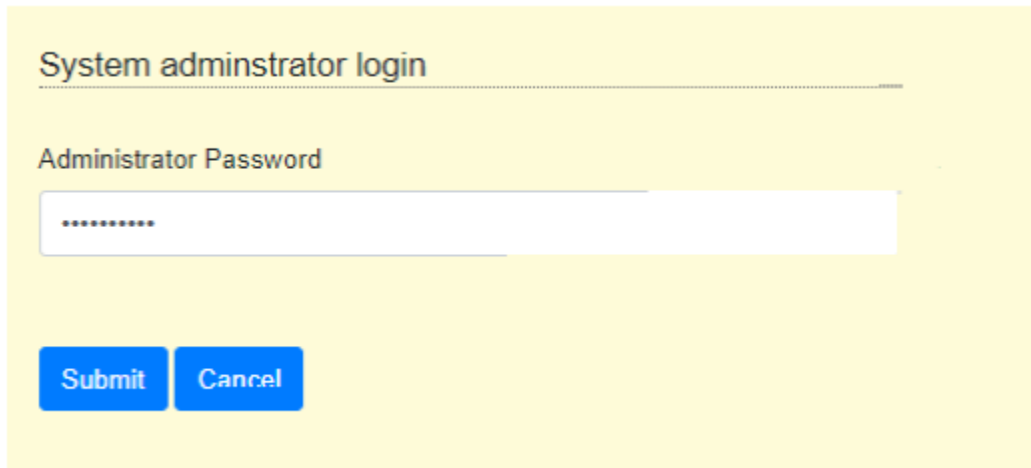
Warwick client id

Working Directory

Allowed process

Fig. 1: Fig 1 Configuration Form

- For security reason, a local log in dialog (Fig. 1) will pop up to allow the local administrator to sign in so no other user can use this option, the local administrator should input the password which he set during the system installation



The image shows a 'System administrator login' dialog box. It has a title bar with the text 'System administrator login'. Below the title bar, there is a label 'Administrator Password' followed by a text input field containing several dots, indicating a password. At the bottom of the dialog, there are two buttons: 'Submit' and 'Cancel'.

Fig. 2: Fig 1 Local login Form

- Then a registration form will pop up (Fig. 2), it is needed to provide your local website name, and URI (which needs to start with <https://>, e.g. <https://myserver.com>) and optionally a brief description. When the user selects submit, this information will be sent to Warwick EnteroBase and a conformation dialog contains the submitted details will pop up.
- An email will be sent to the user (loal administrator) who registered the local instance, the email will contain a Warwick temporary token, it will be used to authorize “test uploading 100 files” from your local installation to Warwick EnteroBase and submit the results (uploading time) to Warwick EnteroBase. This test is a part of the registration process, and should be done once the local administrator received the token so the Warwick EnteroBase administrator can check the registration request.

## 2.3 Files upload test

- Inside the system configuration form (Fig. 1), there is a “Warwick temporary token” field, local administrator should set this value using the token whihc he has received by email. Then local administrator should select “File Uploads Test” option then click “Start” button (Fig. 2).
- After submitting the upload test results, the registration request will be verified by Warwick EnteroBase administrator.
- Once the registration request has been approved, local administrator will receive an email which will contain the client id and a client password which will be used to authorize the communication between the local installation and the Warwick EnteroBase.
- Local administrator needs to update your system configuration using these values and restart the application. At this stage, the system is ready and can be used.

### 2.3.1 Notes

- We should use different database user as using postgres user is not recommended. This can be done in the automatic installation script.

### Local EnteroBase Registration

---

Name\*

URL\*

Description\*

\* This configuration will be used to register your local EnteroBase with [Warwick EnteroBase](#)

Fig. 3: **Fig 2 Client Registration Form**

- Using “Warwick client id” instead of using “Warwick client password” field, this is can be used to authorize upload files test.
- Instance name and icon should be removed. The instance name should be synchronized with the registration form.



**Configuration**

Instance name

Instance icon

Database user

Database server uri

Database port

Database password

Warwick temporary token

Warwick client password

Warwick client id

Working Directory

Allowed process

Fig. 4: Fig 1 system configuration form

Upload Files Test

Press "Start" button to start the test.  
It will test uploading 100 files to Warwick EnteroBase and calculate the elapsed time, the test results will be submitted to Warwick EnteroBase.

Fig. 5: Fig 2 Files upload test form



## USER GUIDE

This section will cover the main local EnteroBase functions

### 3.1 Login using Warwick EnteroBase

- The user should select “Sign in with Warwick EnteroBase” from the main page (Fig. 1)
- The user will be redirected to Warwick EnteroBase login page, once he enters his username and password and select login he will be asked to authorize local EnteroBase installation instance to access his username and email address in addition to submit strains metadata along with assemblies in behalf of him (Fig. 2).
- After the user confirmation, he will be redirected back to the local EnteroBase instance and a page which has a list contains his Warwick EnteroBase active databases (Fig. 3)
- Then the user can select the database which he wants to work with

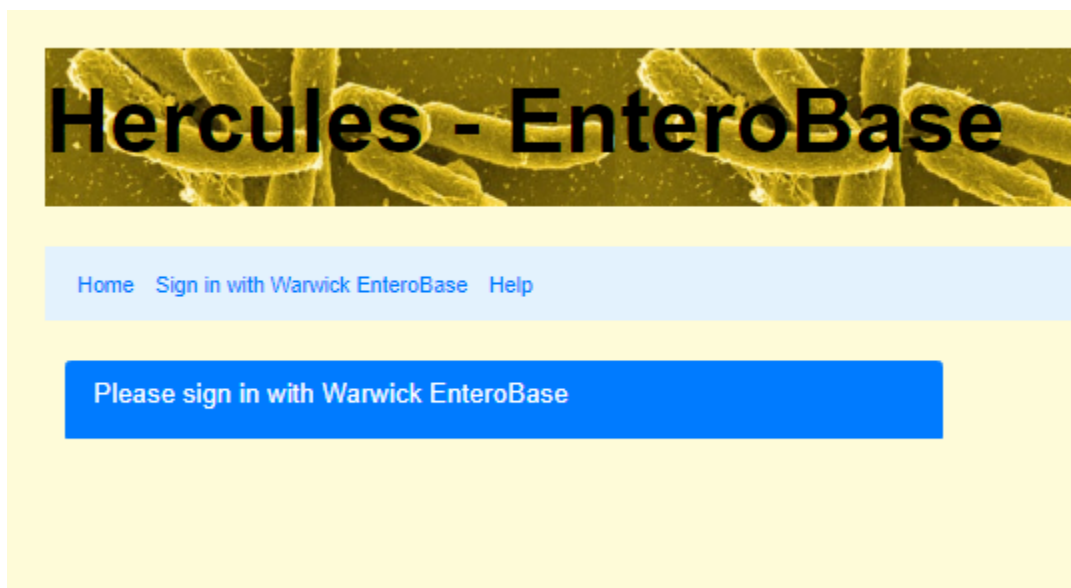


Fig. 1: Fig. 1 First Page

### 3.2 Upload Single Strain

- This is the default database option.

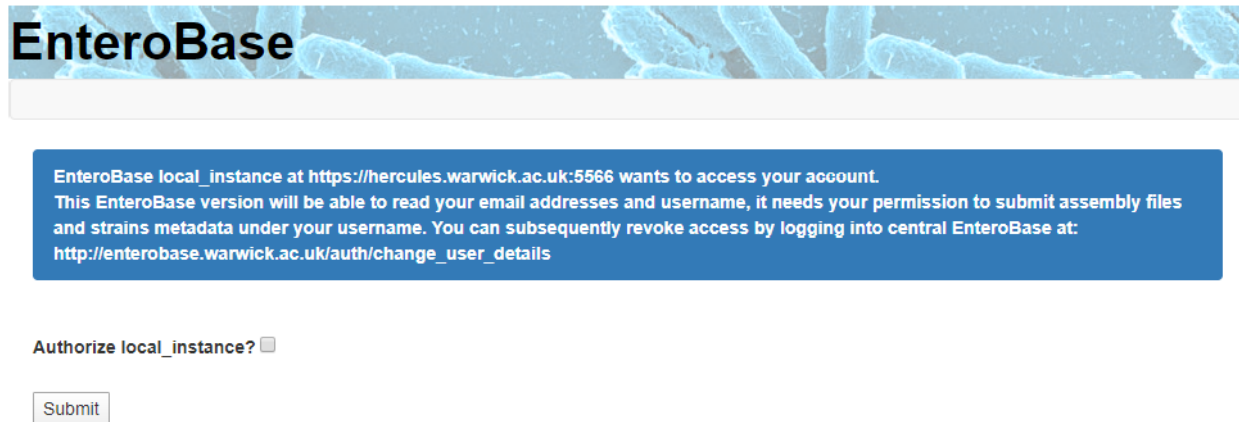


Fig. 2: Fig. 2 Warwick EnteroBase Consent dialog

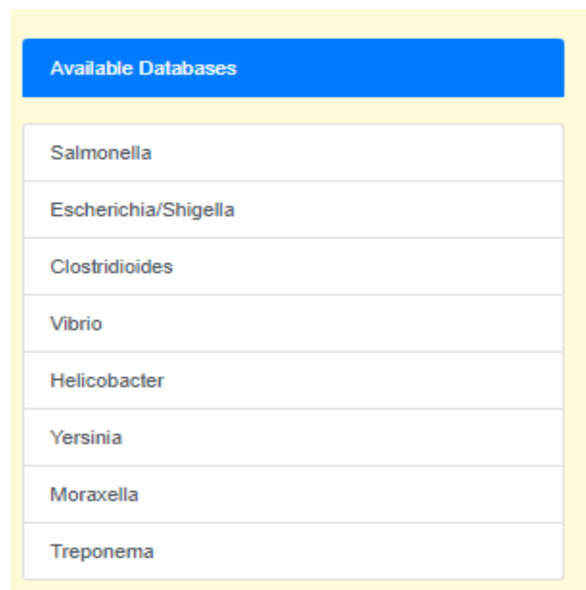


Fig. 3: Fig. 3 Active Databases dialog

- It allows the user to enter the strains metadata and the provide the read files.
- Then it will submit the metdata and upload read files.

Salmonella

Upload Single Strain  
Upload Multiple Strains  
Jobs Status

Add Strain Metadata

Release Period\* (Months) 0

Strain Name*	<input type="text" value="Strain name, required"/>		
Lab Contact*	<input type="text" value="Lab contact, required"/>		
Collection Date*	<input type="text" value="dd/mm/yyyy"/>		
Read File 1*	<input type="button" value="Choose file"/> No file chosen		
Read File 2*	<input type="button" value="Choose file"/> No file chosen		
Source	Source Niche	Source Type	Source Details
	<input type="text"/>	<input type="text"/>	<input type="text"/>
Location	Continent	Country	City
	Europe	United Kingdom	<input type="text"/>
Subspecies	<input type="text"/>		
Disease	<input type="text"/>		
Antibiotic Resistance	<input type="text"/>		
Comment	<input type="text"/>		

Fig. 4: Upload Single Strains dialog

### 3.2.1 Notes

- This option should be discarded later once uploading multiple strains option completely implemented as it allows to submit one or more strains.

## 3.3 Upload Muliple Strains

- This options allows the user to enter one or more strains metadata along with the read files
- The user can write the metadata inside the table or import the metdata from a text file
- The user can select a folder which contains the read files and automatically it imports the paired read files for each strain and creates a row for it.
- The user has the option to export these data to a text file which he can edit and import the modified file.

### 3.3.1 Notes

- This option is not completely implemented.
- **It is needed to use the Zhemini API for source and location**

Salmonella

Upload Single Strain  
Upload Multiple Strains  
Jobs Status

Add Strain/s Metadata

[Add read files](#) [Export to text file](#) [Import from text file](#) [Submit](#)

Name	Read files	Collection Date	Lab Contact	Accession Average Length	Source Niche	Source Type	Source Details	Location Country

Showing all 1 rows

[ADD](#) [DELETE](#)

Fig. 5: Upload Multiple Strains dialog

- This requires developing Warwick EnteroBase API functions
- I have implemented importing the data from a text file which contains the metadata but it needs more testing
- Submitting strains metadata and uploading read files are not implemented yet but we can modify and use the functions which are used to upload single strain.

### 3.4 Jobs Status

- This option allows the user to check his assembly jobs status

Salmonella

Upload Single Strain  
Upload Multiple Strains  
Jobs Status

Jobs status

[Custom View](#)

Name	Uploaded Date	Release Date	Pipeline	Status	Notes	Entero Barcode
Strain one	May 28 2020 16:53	May 28 2020 16:53	QAssembly	Uploaded	-	-
Strain Two	May 28 2020 17:00	May 28 2020 17:00	QAssembly	Uploaded	-	-
Strain Three	May 28 2020 17:04	May 28 2020 17:04	QAssembly	Uploaded	-	-

Showing all 3 rows

Fig. 6: Jobs Status dialog

### 3.4.1 Notes

- We should modify this table to include batch jobs rather than single strain job
- A link for each batch should be provided so the user can check the status of each strain
- **The previous link will open other dialog which allow the user to:**
  - check the status of the assembly,
  - alert metadata,
  - provide new read files in case of the assembly job failed due to read files
  - submit the strains metadata along with read files to Warwick EnteroBase.
- The previous functions are not implemented yet





## DEVELOPING LOCAL ENTEROBASE

### 4.1 Singularity images creation

Local EnteroBase is hosted in [Bitbucket repository](#)

The following figure illustrates the Local EnteroBase folder structure, all the images definition files are saved in the Singularity\_Images sub-folder

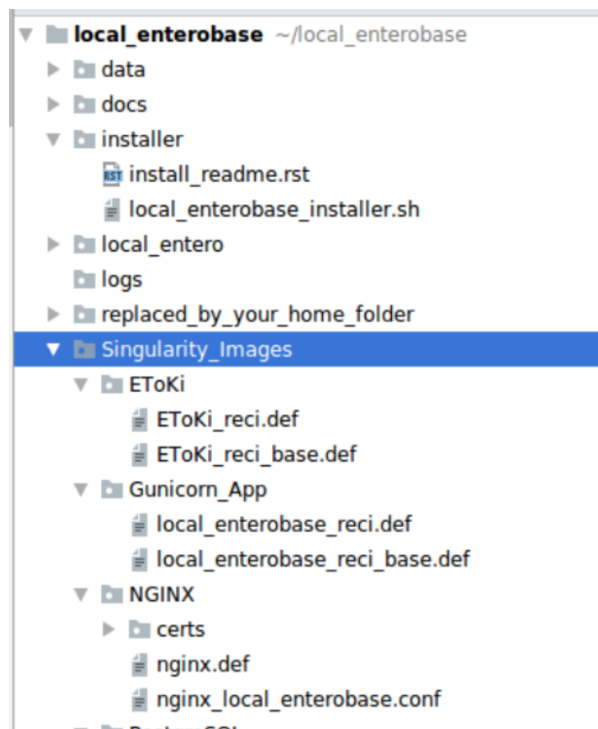


Fig. 1: Local EnteroBase folder structure

#### 4.1.1 NGINX Image

##### Required files and folders

In “local\_enterobase/Singularity\_Images/NGINX” sub-folder

- Recipe file: **nginx.def**

- Default settings for nginx: **nginx\_local\_enterobase.conf**
- temporary ssl certificate: **“certs” folder**

### Content of the recipe file: nginx.def

- The first two lines in the def file instruct to pull the nginx docker image

```
Bootstrap: docker
From: nginx
```

- The following section copies the customized NGINX configuration file “nginx\_local\_enterobase.conf”, as well as “certs” folder which contains temporary ssl certificate to a sub-folder inside the container.

```
%files
local_enterobase/Singularity_Images/NGINX/nginx_local_enterobase.conf /var/www/
local_enterobase/Singularity_Images/NGINX/certs/ /var/www/
```

- The following section creates an entry point which will be used to copy the configuration and temporary ssl certificate files to the host machine. It also creates a ‘logs’ folder. Please note that “/home/nginx\_user/” is a virtual folder in the container which will be mapped at run time to a real folder in the machine which will host the container as it will be described in “Running NGINX Container” section.

```
%apprun prep_nginx
if ! test -f /home/nginx_user/nginx_local_enterobase.conf; then cp /var/www/nginx_
↪local_enterobase.conf /home/nginx_user/; fi
if ! test -d /home/nginx_user/certs; then cp -R /var/www/certs /home/nginx_user; fi
chmod 700 /home/nginx_user/certs
if ! test -d /home/nginx_user/logs; then mkdir /home/nginx_user/logs; fi
```

- The following section creates another entry point, for validating the configuration file and checking the server status

```
%apprun test_nginx
nginx -t -c /home/nginx_user/nginx_local_enterobase.conf
```

- Finally, the *startscript* section is used to start the server as a singularity instance

```
%startscript
nginx -c /home/nginx_user/nginx_local_enterobase.conf
```

### Build NGINX Image

Assuming that the local\_enterobase repository is saved in the sub-folder “local\_enterobase” from the current working folder, the following command is used to build the NGINX singularity image which its name is “nginx\_container.sif”:

```
sudo singularity build nginx_container.sif local_enterobase/Singularity_Images/NGINX/
↪nginx.def
```

### Running NGINX container

- The user needs to create a sub-folder in his machine, the Nginx configuration file and certs folder will be copied to this sub-folder. It will be binded to the “/home/nginx\_user” in the container.

```
mkdir $HOME/nginx_folder
```

Then to copy the nginx configuration file and certs folder to the sub-folder (\$HOME/nginx\_folder), it is required to bind this sub-folder (\$HOME/nginx\_folder) to the “home/nginx” inside the container and run the container with prepnginx:

```
singularity run -B $HOME/nginx_folder/:/home/nginx_user --app prepnginx nginx_
↪container.sif
```

- Then the user needs to edit the configuration file to add their server name or IP address in three different places, by replacing (replaced\_by\_your\_server\_uri) string with your actual server uri, e.g. myserver.com.

```
vi $HOME/nginx_folder/nginx_local_enterobase.conf
```

- If the user wants to get a fresh copy from the customized configuration file, he needs to delete the current configuration file, then run the singularity image with **–app prepnginx** option which is illustrated before.

```
rm $HOME/nginx_folder/nginx_local_enterobase.conf
singularity run -B $HOME/nginx_folder/:/home/nginx_user --app prepnginx_
↪nginx_container.sif
```

- Use the following command to validate the configuration of the server. Change the ‘nginx\_local\_enterobase.conf’ if there is any error message.

```
sudo singularity run -B $HOME/nginx_folder:/home/nginx_user --writable-tmpfs --app_
↪testnginx nginx_container.sif
```

- Use the following command to start the Nginx server as singularity instance:

```
sudo singularity instance start -B $HOME/nginx_folder:/home/nginx_user --writable-
↪tmpfs nginx_container.sif nginx_sing nginx_sing
```

- The instance name is nginx\_sing

## 4.1.2 PostgreSQL Image

### Required files and folders

In “local\_enterobase/Singularity\_Images/PostgreSQL” sub-folder

- Recipe file: **postgres\_simple.def**

### Content of the recipe file: postgres\_simple.def

- The first two lines in the def file pull postgres:alpine docker image

```
BootStrap: docker
From: postgres:alpine
```

- The following section runs the image as a singularity instance. It accepts the arguments from the user at run time when starting the service:

```
%startscript
docker-entrpoint.sh "$@"
```

### Building PostgreSQL image

- Assuming that the local\_enterobase repository is saved in the sub-folder “local\_enterobase” from the current working folder, run the following command to build the PostgreSQL singularity image whihc its name is “postgres\_container.sif”:

```
sudo singularity build postgres_container.sif local_enterobase/Singularity_Images/  
↪PostgreSQL/postgres_simple.def
```

### Running PostgrSQL container

- The user needs to create two folders, for example

```
mkdir $HOME/postgresql  
mkdir $HOME/postgresql/data  
mkdir $HOME/postgresql/temp
```

- The user needs to map previous two folders when running the container,
  - “\$HOME/postgres/data” folder is used for saving the databases
  - “\$HOME/postgres/temp” folder is used by the postgres server for temporary files
- Use the following command to start up the server

```
SINGULARITYENV_POSTGRES_PASSWORD=password singularity instance start -B $HOME//  
↪postgresql/data:/var/lib/postgresql/data -B $HOME/postgresql/temp:/var/run/  
↪postgresql/ postgres_container.sif postgres -p 5432
```

- Please note that:
  - password (something else may be used), is the database user password for the database user “postgres”.The user will need both of them to configure the database server connection.
  - The default port number for the database server is 5432, it is possible to change it by replacing 5432 with the new port.
  - The instance name is “postgres”

## 4.1.3 Unicorn and Application Image

### Required files and folders

In “local\_enterobase/Singularity\_Images/Gunicorn\_App” sub-folder

- Recipe file for the base image: **local\_enterobase\_reci\_base.def**
- Recipe file for the application image : **local\_enterobase\_reci.def**

### Content of the recipe files

- This image is built in two stages:
  - The first stage is building the base image which contains the dependency packages
  - The second one is based on the image which is generated from the first stage and adding local EnteroBase application and the container entry points.

- The reason for using two stages is to reduce the total build time
  - Building of the first image takes long time and it is not changed often.
  - While building of the second one takes much less time, and we need to re-build this image when changing the Local EnteroBase code which happens frequently.

The first image is built using “**local\_enterobase\_reci\_base.def**” recipe file.

- The first two lines in the recipe files pull the a singularity image which is based on ubuntu 18.04

```
Bootstrap:library
From: ubuntu:18.04
```

- Then, at the post section, it downloads and installs the different dependency packages.

The second image is built using “**local\_enterobase\_reci.def**” recipe file

- The first two lines instruct that the image is built based on a local image, then sets the local image name.

```
Bootstrap:localimage
From : local_base_image.sif
```

- Then, at the files section, it copies the local enterobase project code to the image.

```
%files
    $HOME/local_enterobase/ /var/www/
```

- Then it creates an entry point to create a password for the local administrator

```
%apprun set_password "@@"
python3 /var/www/local_enterobase/manage.py set_local_password "$@"
```

- This entry point takes one argument which is the password
- Another entry point is created to run Gunicorn as a singularity instance

```
%startscript
cd /var/www/local_enterobase/
APP_PATH="/var/www/local_enterobase/"
PYTHONPATH=$APP_PATH:$PYTHONPATH
echo $PYTHONPATH
gunicorn -b 0.0.0.0:8000 "local_entero:create_app('production')" --timeout 300 --
↪name "local_entero" --log-file=$HOME/logs/gunilog.log --bind=unix:$HOME/sock
```

- This instructs that Gunicorn runs Local EnteroBase, listens to port 8000 and sets time out to be 300 seconds.

## Building the images

- Assuming that the local\_enterobase repository is saved in the sub-folder “local\_enterobase” from the current working folder, run the following command to build a singularity image named as “local\_base\_image.sif”, the local\_base\_image is built using the following command:

```
sudo singularity build local_base_image.sif local_enterobase/Singularity_Images/
↪Gunicorn_App/local_enterobase_reci_base.def
```

- The following command is used to build the second image:

```
sudo singularity build local_enterobase.sif local_enterobase/Singularity_Images/Gunicorn_App/local_enterobase_reci.def
```

### Running the container

- The user needs to set up a system password so they can use the web interface to configure the application, registers local EnteroBase instance with Warwick EnteroBase and tests upload files to Warwick EnteroBase.

- The local admin should use the following command to set up the password after replacing “mypassword” to their own passwords:

```
singularity run --app set_password local_enterobase.sif -p mypassword
```

- Please note that the user may get error messages regarding database configuration when running the previous command, but these messages can be ignored at this stage

- Then the user needs to use the following command to run the gunicon and the application as a singularity instance which is called “local\_enterobase” (the user may choose to use any other name).

```
singularity instance start local_enterobase.sif local_enterobase
```

- To be sure that the local\_enterobase instance is running, the local admin can use the following command to list all the running Singularity instances:

```
singularity instance list
```

- The output will include the running instance name “i.e. local\_enterobase”, it will look something like this:

INSTANCE NAME	PID	IP	IMAGE
local_enterobase	23456		/home/khaled/local_enterobase.sif

- If the user wants to restart the system, they should stop the instance first, then run it again using the following two commands:

```
singularity instance stop local_enterobase  
singularity instance start local_enterobase.sif local_enterobase
```

### 4.1.4 EToKi Image

#### Required files and folders

In “local\_enterobase/Singularity\_Images/EToKi” sub-folder

- Recipe file for the base image: **EToKi\_reci\_base.def**
- Recipe file for the application image : **EToKi\_reci.def**

#### Content of the recipe files

- This image is built in two stages:
  - The output of the first stage is the base image which contains the dependency packages
  - The second one is build based on the image which is created from the first stage then it clones the EToKi source code from Github in addition to the installation of the python packages inside the requirements file
  - The purpose for doing this is to reduce the build time as
    - \* The the first image building time is long and it is not changed frequently,

- \* It is required to build the second stage image when the EToKi code is changed which happens often and the build time for this image is short.

The first image is built using “**EToKi\_reci\_base.def**” recipe file.

- The first two lines in the recipe file instruct that this image is based on singularity image for ubuntu 18.04

```
Bootstrap:library
From: ubuntu:18.04
```

- Then at %post section of the recipe file, the main dependency packages are installed.

The recipe file of the second stage is “**EToKi\_reci.def**”

- The first two lines in the recipe file instruct that the image is built based on the local image (which is generated from the first stage)

```
Bootstrap:localimage
From : ET_Python.sif
```

- Then at %post section, it:
  - Clones the EToKi source code and save it inside the image
  - Installs all the python dependency packages which are saved in the EToKi requirements.txt file
- Create an entry point to copy configure.ini file to the user home folder

```
%apprun cp_configure
cp /code/EToKi/modules/configure.ini $HOME
```

- Another entry point is created to run EToKi with the different options as a Singularity instance.

```
%runscript
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/code/EToKi/externals/SPAdes-3.13.0-Linux/bin
APP_PATH="/code/EToKi/"
PYTHONPATH=$APP_PATH:$PYTHONPATH
SPDPATH=SPAdes-3.13.0-Linux/bin
PYTHONPATH=$SPDPATH:$PYTHONPATH
echo $PYTHONPATH
cd /code/EToKi
python3 /code/EToKi/EToKi.py "$@"
```

## Building the EToKi images

- Assuming that the local\_enterobase repository is saved in the sub-folder “local\_enterobase” from the current working folder, run the following command to build the EToKi singularity image which its name is “ET\_Python.sif”:

```
sudo singularity build ET_Python.sif local_enterobase/Singularity_Images/EToKi/EToKi_
↪reci_base.def
```

The following command is used to build the second image:

```
sudo singularity build EToKi.sif local_enterobase/Singularity_Images/EToKi/EToKi_reci.
↪def
```

- You may get error messages while building the image, something like that:

```
E: You don't have enough free space in /var/cache/apt/archives/.
FATAL:   failed to execute %post proc: exit status 100
FATAL:   While performing build: while running engine: while running /usr/
↳local/libexec/singularity/bin/starter: exit status 255
```

- If you got the previous error messages you may not have enough disk space to create the singularity temporary files, using sandbox option (**-sandbox**) may solve this issue.

```
sudo singularity build --sandbox EToKi.sif local_enterobase/Singularity_Images/EToKi/
↳EToKi_reci.def
```

- The previous command will build into a folder instead of single file. This can be good for the development and testing.
- I have tried to use different disk to be used for singularity temporary files but did not help
- I have used virtual machine for building the images and have worked fine

## Running the container

- The user needs to run the following command to copy the required configure data file (configure.ini) to the his home folder:

```
singularity run --app cp_configure EToKi.sif
```

- Then, the user should download:
  - usearch software, it is needed to submit a free licence request, the user should receive an email which contains a download link
  - Kraken database, the user can download it using this link [ftp://ftp.ccb.jhu.edu/pub/data/kraken2\\_dbs/minikraken2\\_v2\\_8GB\\_201904\\_UPDATE.tgz](ftp://ftp.ccb.jhu.edu/pub/data/kraken2_dbs/minikraken2_v2_8GB_201904_UPDATE.tgz)

The user should save both of them to the same folder e.g. \$HOME/EToKi\_externals and run the following command to configure EToKi:

```
singularity run -B $HOME/configure.ini:/code/EToKi/modules/configure.ini -B $HOME/
↳EToKi_externals:/EToKi/externals EToKi.sif configure --link_krakenDB /EToKi/
↳externals/minikraken2/ --usearch /EToKi/externals/usearch8.0.1623_i86linux32
```

EToKi is ready to run but at run time, the configure.ini and the folder which contains u-search and the kraken database must be bound to the ones at the host machine. This can be done by adding the following option:

```
-B $HOME/configure.ini:/code/EToKi/modules/configure.ini -B $HOME/EToKi_externals:/
↳EToKi/externals
```

- Please note that I have tested the container with assembly options, although other EToKi's options should be visible but,
- This is the assembly command, assuming that the files are saved in \$HOME/EToKi/examples, is:

```
singularity run -B $HOME/configure.ini:/code/EToKi/modules/configure.ini -B $HOME/
↳EToKi_externals:/EToKi/externals EToKi.sif prepare --pe $HOME/EToKi/examples/S_R1.
↳fastq.gz,$HOME/EToKi/examples/S_R2.fastq.gz -p $HOME/EToKi/examples/prep_out
```



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`