Anfisa v.0.5 Setup & Administration Reference

This document does not cover specific aspects of annotation preprocessing.

Overview

Anfisa is a Linux-based Python application (Python 3.5+) and provides access via HTTP/HTTPS protocols. It deals with datasets of mutation variants in the human genome. The datasets can be of two different kinds with Anfisa providing different functionality dependent on the kind:

- XL-dataset (XL, eXtra Large) usually represents a whole exome (WES) or a whole genome and can encompass up to 10 million variants. Users can search subsets of variants, and form (secondary) workspaces from them to perform more detailed studies.
- Workspace (WS) is a dataset of a small number of variants (up to 10000). Users can view and tag variants in it. Workspaces are either created as derivative datasets from an XL-dataset or can also be directly ingested into the system as primary datasets. The latter option is used for analyzing gene panels.

Only administrators of the system can create primary datasets in the vault of the system or remove them from there. A primary dataset is created directly from prepared data and can be of type XL or WS. “Normal” users can create secondary datasets only (automatically). These are filtered out from XL datasets and are always of type WS.

Anfisa uses the following external systems:

- MongoDB: this database is used to store information about user activities; it does NOT contain information about datasets.
- Druid OLAP system: this engine is used for effective support of XL-datasets (Druid is not necessary while working without XL-datasets)

There are two variants for Anfisa service configuration:

- Server (uWSGI) mode:
    - Anfisa application being wrapped into a uWSGI container (uWSGI is the common way to handle a python application under a web service)...
    - and this container is used by the main web server, usually Nginx or Apache
- Stand-alone mode: this mode is used for development purposes and may be installed on a personal computer without a server environment

Currently there are two kinds of User Interface (UI) provided by the system:

- Legacy UI: a collection of web pages which gives the user access to the full functionality of the system; unfortunately this kind of UI does not satisfy all criteria for a “good UI”. In particular, it works properly only under Chrome or Firefox browsers.
- NextGen Frontend: this version satisfies the criteria for a “good UI”, but currently does not cover the whole functionality yet. E.g., it does not provide access to XL-datasets

Note: currently the NextGen Frontend is not part of the stand-alone configuration.

Setup

Anfisa installation and configuration

To setup the Anfisa system in Server (uWSGI) mode it is recommended to install the stand-alone variant first, and then upgrade it to the server variant.

1. Stand-alone installation

Following instructions are tested on Ubuntu 18.04 LTS. Still, there shouldn’t be anything too specific.

1.1. Prerequisites

- Standard Python tools: Python 3.5+, pip3, virtualenv (optional)
- MongoDB (version 2.6.10 or higher): If not installed already install it with the default configuration using the official guides (https://docs.mongodb.com/manual/installation/; look for the appropriate Linux distribution using the links in this document). It should be up and running.
- git version control system: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git
- If you want to use the XL-dataset functionality, Druid v0.16 needs to be installed (http://druid.io/).

1.2. Installation

To install Anfisa in the stand-alone variant execute the steps below.

- Create a new directory for the project and go there. From now on, this directory will be referred as **ANFISA_ROOT**:
  ```
  $ mkdir -p $ANFISA_ROOT
  $ cd $ANFISA_ROOT
  ```
- At this point we advise one to create virtual environment using any suitable tool. In this example we use virtualenv:
  ```
  $ pip3 install virtualenv
  $ python -m virtualenv venv
  ```

```
$ source venv/bin/activate
```

- Clone the repository of the system:
- `$ git clone https://github.com/ForomePlatform/anfisa.git`
- Cloning the repository creates the directory anfisa, containing the application. Change into this directory:
  `$ cd anfisa`
  Note: below we will refer to this directory as
  ANFISA_HOME =`/data/projects/Anfisa/anfisa`
- Install dependencies by running:
  `$ pip3 install -r requirements.txt`
- Now try to initialize the working environment for the system
  `$ bash deploy.sh`
- This script asks for an installation directory, i.e. the working directory where the system will store information (case data, intermediate files, indices, log files, etc.);
  "`../a-setup`" is recommended but a different name should work too
  Note: below we will refer to this directory as
  ANFISA_WORK =`/data/projects/Anfisa/a-setup`

Now you are good to go! To run the service in the stand-alone variant use commands printed by deploy script:

```
$ env PYTHONPATH=. python app/hserver.py
$ANFISA_WORK/anfisa_<hostname>.json
```

In a browser (Chrome or Firefox are supported) one can see the service at the following URL: http://localhost:8190/dir
Provided the script `deploy.sh` has worked properly, one should see the directory of Anfisa filled with one workspace, and be able to work with that workspace.
(If it is a server installation and there are no open ports on the computer, use ssh tunneling to access this and other pages).

2. Upgrade to server setup

In a server variant Anfisa runs on UWSGi application server with separate so-called NextGen frontend served by a web application server.

2.1. Prerequisites

1. You will need to have root privileges to perform some of the following steps.
2. You need to have a web server installed, Apache or NGINX. Others are good too, but we will provide configuration examples only for those aforementioned.

Before setting up the server variant one needs to answer the following questions:

1. *Which user would run Anfisa?*
   Note: Below we refer to this username as **ANFISA_ADMIN**

2. *What is the URL pointing to the Anfisa application?*
   As a web application Anfisa is run using an address like:
   `http://<server>/<directory>/… (http: or https:)`
   So, one needs to specify this directory. Let's refer to it as **ANFISA_HTML_BASE**. Its name should start and end with symbols '/', and can be as short as '/'.
   When the NextGen Frontend is installed, it can be accessed via this address, and the extended address:
   ANFISA_HTML_APP_BASE = $ANFISA_HTML_BASE + 'app/'
   is used as the base level of the internal REST API and the Legacy UI.
3. *What is the port number for the http socket to be used for uWSGI connection?*
   Should be unique among the sockets running on the computer. Below we will use the number 3041, one is free to choose any other unique number in case of conflict.
4. *What is the name of the MongoDB database which is going to support Anfisa?*
   The name "Anfisa" is recommended.
5. *Where is the Druid system set up?*
   There can be one of three answers:
   1. nowhere - then there will be no XL-datasets support
   2. on the same computer
   3. on a different computer, with access via secure connections

   (see details below in the section about Druid setup)
6. *What is the prefix for names of datasets represented in Druid?*
   The name "Anfisa" is recommended
7. *Does the server provide access to BAM-files for IGV direct support?*
   See below discussion in "IGV direct support" section

And: create the directory $ANFISA_WORK/ui:

```
$ export ANFISA_WORK=/data/projects/Anfisa/a-setup
$ mkdir $ANFISA_WORK/ui
```

In the following sections we would refer to deployment location of static files of the NextGen frontend as **ANFISA_HTML_STATIC**=`/var/www/html/anfisa`

2.2. Configure the application

Copy the configuration file $ANFISA_HOME/anfisa.json to the directory $ANFISA_ROOT and make the following changes to it:

- `"file-path-def": {"WORK": "${HOME}/../a-setup"},`
  Change the value of $WORK to the value of $ANFISA_WORK
- `"html-base": "/anfisa/app",`
  Write the value of $ANFISA_HTML_APP_BASE here (it should end with `"/app"` if it is a server installation)
- `"html-ws-url": "/anfisa"`

Write the value of $ANFISA_HTML_BASE here if the NextGen UI is configured, else keep the default value “ws”

- `"mongo-db": "Anfisa"`
  Change this if a different database name is chosen for the MongoDB
- `"data-vault": "${WORK}/vault",`
  You can change this value to put the vault to any other place on the computer. This directory can be large: it will contain the entire data of the datasets.
- `"http-bam-base": “http://<server>/anfisa/links/”`
  HTTP base directory for access to BAM-files, for IGV direct support, see discussion below. Uncomment this option and set it up correctly if the server provides access to BAM-files, otherwise keep it commented.
- `"dir-files":`
  `["/ui", "${HOME}/int_ui/files"],`
  Drop this line and uncomment the next one:
  `["/ui", "${WORK}/ui"],`
  This instruction and the next one will be used for anti-cache subsystem - see description below; **make sure that you have the directory `$ANFISA_WORK/ui` is created**.
- `"mirror-ui": ["${HOME}/int_ui/files", "${WORK}/ui"],`
  Please uncomment this instruction, see above.
- `"druid": {...}`
  If you are going to use XL-datasets, set up the parameters of Druid properly (see the section below).

2.3. Install and setup the NextGen Frontend

1. Ensure you have `npm` installed. Install it in the default configuration using the official guides (https://nodejs.org/en/download/package-manager/).
2. Go to the temporary location where you would like to build the NextGen Frontend, for example:
   `$ cd /tmp`
3. Clone the sources from the repository:
   `$ git clone https://github.com/ForomePlatform/Anfisa-Front-End.git`
4. `$ cd Anfisa-Front-End`
5. Download and install project dependencies with:
   `$ npm install`
6. Create file `.env.production.local` using the text editor you prefer, it should look like the following:
   `BASE_URL=`**<ANFISA_HTML_BASE>**
   `VUE_APP_API_URL=`**<ANFISA_HTML_APP_BASE>**

   Initiate the build process:
   `$ npm run build -- --mode=production`

Note: for some reason, `npm install` sometimes silently fails to install node-sass module. If you encounter errors like that:
`Error: ENOENT: no such file or directory, scandir '/tmp/Anfisa-Front-End/node_modules/node-sass/vendor'`
You may want to consult with answers in this issue:
https://github.com/sass/node-sass/issues/1579

7. At the end of the build process you will get a copy of the UI in the local directory `dist/`
   Copy the entire directory to the destination which your web server uses for static files,
   If it is not in place already, create the folder:
   `$ sudo mkdir -p` **<HTML_STATIC>**

   Afterwards, copy all files and folders to this destination:
   `$ cp -R dist/` **<HTML_STATIC>/<ANFISA_HTML_BASE>**

8. Ensure that the new copy has proper access permissions.

2.4. Create the uWSGI container descriptor

In the directory $ANFISA_ROOT create the file `uwsgi.anfisa.ini` with the following content (replace the conventional names used in this document with their proper values):

```
[uwsgi]
socket = 127.0.0.1:3041
chdir = $ANFISA_ROOT
wsgi-file = $ANFISA_HOME/app/run.py
pythonpath = $ANFISA_HOME
processes = 1
threads = 30
logger = file:logfile=$ANFISA_WORK/logs/uwsgi.log,maxsize=500000
lazy
```

Note that the number 3041 is an HTTP socket. It should be unique among the HTTP sockets running on the computer, and can be changed to any other unique number within.

2.5. Register the uWSGI container

As root (e. g. using sudo), create the file `/etc/systemd/system/anfisa.service` with the following contents (replace placeholders used in this document with their proper values):

```
[Unit]
Description=uWSGI Anfisa
User=$ANFISA_ADMIN

[Service]
User=$ANFISA_ADMIN
```

```
Group=$ANFISA_ADMIN_GROUP
ExecStart=$UWSGI_EXE \
          --ini $ANFISA_ROOT/uwsgi.anfisa.ini \
          --virtualenv $ANFISA_ROOT/venv
# Requires systemd version 211 or newer
RuntimeDirectory=uwsgi
Restart=always
KillSignal=SIGQUIT
Type=notify
StandardError=syslog

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

Note: you can obtain uWSGI executable location with following:

```
$ cd $ANFISA_ROOT
$ source venv/bin/activate
$ which uwsgi
```

Also take care of permissions for this file:

```
$ sudo chmod 0644 /etc/systemd/system/anfisa.service
```

Now we need to notify systemd of the new service:

```
$ sudo systemctl daemon-reload
```

And start the service:

```
$ sudo systemctl start anfisa
```

2.6. Setup web server configuration

We provide you with configurations templates for two popular web servers.

2.6.1 Nginx

Insert the following configuration directives into configuration file, for example:
`/etc/nginx/sites-enabled/default`
It governs the behaviour of the web server with respect to the NextGen frontend application (all placeholders in the following configuration need to be replaced with their actual values):

```
#####
#####   Anfisa
#####
location <ANFISA_HTML_APP_BASE> {
    include uwsgi_params;
    uwsgi_read_timeout 300;
```

```
    uwsgi_pass 127.0.0.1:3041;
}
location <ANFISA_HTML_BASE> {
    root <HTML_STATIC>;
    try_files $uri $uri/ /<ANFISA_HTML_BASE>/index.html;
}
location ~ <ANFISA_HTML_APP_BASE>/ui {
    rewrite ^<ANFISA_HTML_APP_BASE>/ui/(.*)$ /$1 break;
    root <ANFISA_WORK>/ui;
}
location ~ <ANFISA_HTML_APP_BASE>/ui/images {
    rewrite ^<ANFISA_HTML_APP_BASE>/ui/images/(.*)$ /$1 break;
    root <ANFISA_HOME>/int_ui/images;
}
```

The meaning of the above instructions is as follows:

1. The first instruction establishes connection to the uWSGI container with the main Anfisa application when requests (URL) starts with **<ANFISA_HTML_APP_BASE>**.

   For example, in the notation of this document, a request to the directory page will have this URL: `http://<site>/`**<ANFISA_HTML_APP_BASE>**`/dir`
   It is necessary to get access to the kernel REST API of the application and to the Legacy UI. The directory path for these requests should end in '/app/'.

   Note that we use here the socket number 3014, it can be changed to anything else, as long as it is the same as in `uwsgi.anfisa.ini` (see above)
2. The second instruction deploys the NextGen Frontend, when the request starts with directory path **<ANFISA_HTML_BASE>**. For example, in notations of this document the top page of the NextGen Frontend will have this URL:
   `http://<site>/anfisa/#/`
3. The last two instructions forward content of the files used in the internal UI:
   a. one forwards files (with extensions .js and .css) from the mirror anti-cache directory **<ANFISA_WORK>**`/ui/`
   b. the other forwards the images from the directory **<ANFISA_HOME>**`/int_ui/images`
4. There can be one more instruction here if the server provides access to BAM-files for direct IGV support, see discussion below.

Finally, you need to test new configuration:

```
$ sudo nginx -t
```

If everything is ok, reload:

```
$ sudo systemctl reload nginx
```

To ensure that system is up, visit http://localhost/<ANFISA_HTML_BASE> and you should see the main application page. Look for workspaces in the menu to ensure that connection to the main Anfisa application is configured correctly.

2.6.2 Apache

[WRITE IT!!!]

2.7. IGV direct support

Anfisa provides functionality to run IGV local application (https://software.broadinstitute.org/software/igv/download) over any variant in scope. To perform this call the server should provide HTTP/HTTPS access for BAM-files included in case. The setting “http-bam-base" in anfisa.json configuration file serves for this purpose. However, one needs to set up this access. It is not necessary to use the same WEB-server for these files, BAM-files can be located somewhere else.

In a simple example configuration, NGINX simply serves BAM-files from the location on the drive. Files are organized on disk as follows:

```
<BAM_FILES_LOCATION>/{case}/{sample}.hg19.bam
<BAM_FILES_LOCATION>/{case}/{sample}.hg19.bam.bai
```

NGINX configuration in turn contains the following:

```
location /bams {
    root <BAM_FILES_LOCATION>;
}
```

Finally, Anfisa configuration (anfisa.json) contains the following line:

```
"http-bam-base": "https://<site>/bams"
```

2.8. Druid

At the moment of this document being written, Apache Druid v.0.16.0-incubating is the most recent one, and this exact version is assumed. Best source of information on Druid installation and configuration is it’s documentation:
https://druid.apache.org/docs/0.16.0-incubating/design/index.html

In the following section we assume that Druid is installed and properly configured according to its documentation.

2.8.1. Connection configuration

When Druid is installed on the same machine as Anfisa, one needs to uncomment “`druid`” section of the `anfisa.json` configuration:

```
"druid": {
```

```
    "vault-prefix": "Anfisa",
```
Prefix is added to Druid names of datasets. It allows to use single Druid instance for multiple instances of Anfisa.
```
    "index": "http://<DRUID_IP>:8081/druid/indexer/v1/task",
    "query": "http://<DRUID_IP>:8888/druid/v2",
    "sql":   "http://<DRUID_IP>:8888/druid/v2/sql",
    "coord": "http://<DRUID_IP>:8081/druid/coordinator/v1"
```
Settings define addresses of four different kinds of requests to Driud.
Settings are configured for Druid version v.0.16.0.
For Druid version v.0.13 ports of requests have different values:
index: 8090, query/sql 8082, coord: 8081
```
    "-scp": {...}
}
```

2.8.2. Separate machine configuration

In case of a separate machine configuration, Anfisa needs to copy data to the machine with Druid in order to perform data ingestion. This is done via `scp`.

In this section we will use:

- Instance with Anfisa installation — <ANFISA_PC>
- Instance with Druid installation — <DRUID_PC>

Configuration steps:

1. One needs to create data directory, which would receive data.
2. SSH keypair needs to be created on a <ANFISA_PC>:
   ```
   $ ssh-keygen
   ```
   Important: passphrase should be empty.
3. Public key of the new keypair needs to be added to the end of the `/home/<user>/.ssh/authorized_keys` file on the <DRUID_PC>:
4. **Important:** you have to manually perform first login from <ANFISA_PC> to the <DRUID_PC>.
   ```
   $ ssh -i <PATH_TO_PRIVATE_KEY> <user>@<DRUID_PC>
   ```
5. Uncomment "`scp`" subsection of the "`druid`" section in the `anfisa.json`.
   ```
   "scp": {
       "dir": "<DATA_DIR>",
       "key": "<PATH_TO_PRIVATE_KEY>",
       "host": "<USER>@<DRUID_PC>",
       "exe": "/usr/bin/scp"
   }
   ```
   Where:
   a. **<DATA_DIR>** is a path of an existing directory on <DRUID_PC>. This is the target directory, which would receive data.
   b. **<PATH_TO_PRIVATE_KEY>** is a path to the private key on <ANFISA_PC>.

Administration

Running the service: Stand-alone and server mode

Running the service

The service itself and the utilities (see below) should be run under the same user **ANFISA_ADMIN.**

In standalone mode the service is started with either of the following command sequences:

```
$ cd $ANFISA_HOME
$ python3 -m app.run [path to configuration file]
```

or

```
$ env PYTHONPATH="$ANFISA_HOME"
$ python3 -m app.run [path to configuration file]
```

In server mode, while setting the uWSGI container, also run the process by the user ANFISA_ADMIN and use the correct configuration file.
The following commands start resp. stop the uWSGI service:

```
$ sudo systemctl start anfisa
$ sudo systemctl stop anfisa
```

Sharing data between multiple users

The current version of the service makes no distinction between users who share access to datasets and perform different actions in a commonly shared environment. It is supposed that these users form a team of experts, and can make agreements about sharing conflicts in an informal way.

Using server configuration (anfisa.json)

In both modes the main configuration file is important. There is the file from the repository `$ANFISA_HOME/anfisa.json`. This file is the default configuration of the service. It is assumed that it is enough to run the stand-alone mode of the service in the default configuration.

In the server mode, as well as in the stand-alone one but with some modifications, one needs to make a copy of this file and put it in another directory. `$ANFISA_ROOT` is recommended. The name of this file can be the same, `anfisa.json`, or can be changed.

It is important to provide access to the same configuration for administrator utilities and for the stand-alone mode to use it in the start service command.

REST API

REST API is the kernel of the system. It is a variety of HTTP requests built within the concept of REST API (ask Google about it). In short, these requests satisfy certain architectural conditions and their responses have the form of JSON objects. Both Legacy UI and NextGen Frontend provide HTML pages, and use these requests in a way hidden from the users to transfer data and actions upon the service. If the NextGen Frontend was perfect, there would be no need to use the internal UI at all. However, the configuration aspects for REST API and the internal UI are close to each other.

File content transfer aspect

The aspect of transferring file content is a technical one. However, it produces some complexity in the configuration and needs a detailed explanation.

Any WEB-service needs to transfer file content in response to HTTP requests. When we have a “main server” (NGINX/Apache/...), it is good practice to configure it to perform such requests. So in the server mode two kinds of files (used by internal UI) are transferred on the “main server” level: control files *.js and *.css, and images. (See also the next note, on anti-cache mechanism).

In the stand-alone mode there is no such thing as “main server”, so file transfer requests should be supported by the service itself. The option `"dir-files"` in service configuration (`anfisa.json`) regulates these operations.

Therefore we have two different mechanisms in different modules to do the same thing, and may it look too complex in the context of this document.

There is a third kind of files that should be transferred. Any Export operation produces an Excel file (*.xlsx) that should be downloaded by a client (see below in the section about Export). These transfers happen rarely, and the internal service mechanism (via “`dir-files`”) is sufficient for them, so there is no need to configure the “main server” for their support.

Anti-cache mirroring mechanism

It is used for purposes of the internal UI in the server mode. The problem it solves is the following. The internal UI uses some files (with extensions *.js and *.css), and these files are checked out from the repository. So after a push from the repository these files can change. If these files were used by the UI directly, there would be a possibility that the user’s browser will ignore changes in such a file and use some outdated cached copy of its previous version instead of the fresh version of it. The workaround for this problem is to create a mirror directory, copy into it all the necessary files but slightly modify their names in such a way that different versions of the same file will have different names.

This mechanism is recommended for the server mode. However, it can be set up in the stand-only mode as well.

The configuration file of the Anfisa service: anfisa.json

The configuration file is a file in JSON format, usually named `anfisa.json`. Below is a fragmented example of this file, with comments. The JSON format does not provide a proper comment support, so in the text below, “commenting out a property” means essentially breaking its name, usually by prefixing ‘--’.

```
{
```

```
"file-path-def": {"WORK": "${HOME}/../a-setup"},
```

- The directory macro ${HOME} is predefined; here we define more macros, in particular the macro ${WORK}

```
"host": "0.0.0.0",
"port": 8190,
```

Host and port are used in the stand-alone configuration to setup the server on a specific port, ignored in the server (UWIGI container) mode

```
"html-base": "/anfisa/",
```

Used in the internal UI: its HTML pages need to know their location in terms of URL addresses, just to make correct reference to REST API or other pages of the internal UI, in case of server setup with NextGen UI should end in ‘/app’

```
"html-ws-url": "ws",
```

Address of the NextGen UI base page if the NextGen UI is set up, else keep the default value `"ws"`

```
"html-title": "Anfisa",
```

Title prefix used in the pages of the Legacy UI

```
"mongo-db": "Anfisa",
```

The database in MongoDB used by the system

```
"data-vault": "${WORK}/vault",
```

The location of the vault directory, see details below

```
"http-bam-base": "http://<server>/anfisa/links/"
```

HTTP base directory for access to BAM-files, used in IGV-links. Uncomment this option and set it up correctly if the server provides access to BAM-files, otherwise keep it commented.

```
"export": {
```

Configuration of export functionality, see details below

```
"excel-template":
    "${WORK}/export/SEQaBOO_output_template_20190317.xlsx",
```

The template used to configure the Excel export styles.

During evaluation of the script deploy.sh the file is being downloaded from URL:

"https://www.dropbox.com/s/4dvunn3dusqc636/SEQaBOO_output_template_20190317.xlsx"

```
"work-dir": "${WORK}/export/work"
```

The directory where the service stores exported files

```
    },
    "dir-files": [
```

Setup of the mechanism of forwarding files as request results

```
        ["/ui/images", "${HOME}/int_ui/images"],
        ["/ui", "${HOME}/int_ui/files"],
```

Requests for images and other sources, actual in stand-alone case. Should transfer the content of files located in the specific directory in $ANFISA_HOME; used in the internal UI in the stand-alone mode; in the server mode the same task is solved by configuration of the “main server”, Nginx or Apache

```
        ["--/ui", "${WORK}/ui"],
```

Requests for the source files when the anti-cache mechanism is on; in the server mode, to be used in the internal UI instead of the previous instruction; (in server setup drop two leading ‘-’ to make it working, and comment out the previous instruction)

```
        ["/excel", "${WORK}/export/work"]],
```

This line sets the directory used to place the content of exported Excel files, supposing that they are going to be immediately downloaded by an external client

```
    "--mirror-ui": ["${HOME}/int_ui/files", "${WORK}/ui"],
```

This instruction turns the anti-cache mechanism on; it consists of the paths to the source and target directories for mirroring (drop two leading ‘-’ to make it working)

```
    "druid": {
        ...
```

See the section about Druid configuration above

```
    },
    "logging": {
        …
```

Some standard Python way to configure the logging of a service. Please pay attention to one specific line of this stuff:

```
...
        "filename": "${WORK}/logs/anfisa.log"
```

This line contains the configuration of the path to the logging directory

```
    ...},
    "doc-report-css": "${HOME}/int_ui/files/report.css",
    "doc-pygments-css": "${HOME}/int_ui/files/py_pygments.css",
```

These two options are used to configure styling of documentation pages for datasets

```
    "run-options": []
```

Some additional option to configure Anfisa service. Currently out of use.

```
}
```

Directory structure: vault, datasets, logs, export directory

Strictly speaking, there is no real necessity in the existence of two “standard’ directories used in this document: `$ANFISA_ROOT` and `$ANFISA_WORK`. But the system strongly requires all the subdirectories placed under $ANFISA_WORK. One can place them anywhere on the computer and modify configuration (`anfisa.json`) correspondingly.

- $ANFISA_WORK/**vault** - vault directory
  Information of all the datasets supported by the system is placed here: one dataset - one directory. (There might be a serious need to place this directory in another location: it can happen if the size of a dataset grows and one needs to move the directory to another disk. Just move it, and change the “data-vault” line in the config file correspondingly)

- Subdirectories of the vault directory
  Directories for subsets should be created by calls of the app.storage utility made by user $ANFISA_ADMIN. Removal of an XL-dataset should be also done using this utility, because connection to Druid is required in this process. The removal of a workspace also can be done this way, but it is just equivalent to removal of the corresponding directory.
  Please note that this directory contains the empty file `active`. To turn a dataset out of use in terms of the service one needs to remove this file and restart the service.To re-activate the dataset just create the file `active` once again (by the utility touch)

- $ANFISA_WORK/**export** - export directory
  This is the place where the excel-template file is located. The main need is in the subdirectory
  $ANFISA_WORK/**vault**/work
  This is the place where the system stores all the Excel files generated for export. Each file here is used only once, but there is no automatic procedure to clean files from here later. This clearance should be done by the system administrator periodically

- $ANFISA_WORK/**logs** - log directory
  In the stand-alone mode only the `anfisa.log` file is stored here, plus its old portions. In the server mode there are two log files: `anfisa.log` and `uwsgi.log`. The former collects all errors that occurred “inside service logic”, the latter collects meaningful errors at the start of the service.
  There is no automatic procedure to cleanse this directory either. Administrator should do it periodically.

- $ANFISA_WORK/**ui** - mirroring directory

Used in the anti-cache mechanism in the server mode. See details above

Dataset internal structure

Minimal dataset data is just annotated JSON file: <dataset>.json.gz
In expanded form, dataset data forms directory with the following:

- Inventory of dataset: <dataset>.cfg
- Annotated JSON: <dataset>.json.gz
- Optional subdirectory doc/ with supplementary documentation materials
- Optional: BAM-files for samples in dataset case
- May be more files

Anfisa dels either with separate annotated JSON file or with the whole structure referred by inventory file.
During evaluation of the script deploy.sh sampling dataset PGP3140_panel_hl is being loaded by URL <...>, extracted and loaded to Anfisa using app.storage administration util.

Datasets configuration: storage.dir

To manage and configure datasets, the file `storage.dir` is used. It should be placed in the same directory as `anfisa.json`. The file is in JSON format and contains the full path to `anfisa.json` as well as a description of the dataset(s), for example:

```
{
   "anfisa.json": "/data/projects/Anfisa/anfisa.json",
   "datasets": {
      "PGP3140_wgs_hlpanel": {
         "kind": "ws",
         "inv":
"/data/bgm/cases/pgp3140_wgs_hlpanel/pgp3140_wgs_hlpanel.cfg"
      },
      ...
   }
}
```

In setting up a new dataset, the file `storage.dir` must be edited manually to add the new dataset to it. The file `storage.dir` is only used when running the app.`storage` utility which handles the datasets. It is therefore safe to make changes to the file if the app.`storage` utility is run immediately after the change and thus the syntactic consistency of the file is checked.

The allowed values of the `"kind"` parameter are `"ws"` or `"xl"`. Please be careful not to load too large datasets (over 5000 records) with the “kind” option set to “ws”!

It is strongly recommended to include in the description of each dataset the link to the dataset inventory file, in the `"inv"` field (as in the example above). However, in some urgent cases it is possible to use a field labelled `"a-json"` instead of `"inv"` and define there the path directly to the annotated JSON file (whose name normally ends in `".json.gz"`).

Administration of datasets: app.storage

The utility used to create or drop a dataset in the vault of the system. It is recommended to run it from $ANFISA_HOME directory, or use PYTHONPATH env variable to set this directory as python base.
Here are the options of the utility:

```
$ cd $ANFISA_HOME
$ python3 -m app.storage --help
usage: storage.py [-h] [-d DIR] [-c CONFIG] [-m MODE] [-k KIND] [-s SOURCE]
                  [-i INV] [-f] [-C] [--mongo MONGO]
                  [--reportlines REPORTLINES]
                  name

positional arguments:
  name                  Dataset name

optional arguments:
  -h, --help            show this help message and exit
  -d DIR, --dir DIR     Storage directory control file
  -c CONFIG, --config CONFIG
                        Anfisa configuration file, used only if --dir is
                        unset, default = anfisa.json
  -m MODE, --mode MODE  Mode: create/drop/druid-push/doc-push
  -k KIND, --kind KIND  Kind of dataset: ws/xl, default = ws, actual if
   --dir is unset
  -s SOURCE, --source SOURCE
                        Annotated json, actual if --dir is unset
   and mode = create
  -i INV, --inv INV     Annotation inventory
  -f, --force           Force removal, actual if mode = create
  -C, --nocoord         Druid: no use coordinator
  --mongo MONGO         Mongo name, default=name
  --reportlines REPORTLINES
                        Portion for report lines, default = 100
```

Examples of calls:

```
$ python3 -u -m app.storage -m create -k ws -f -s ~/tmp/PGP3140.json PGP3140
$ python3 -m app.storage -m drop -k ws PGP3140
```

Comments.

- The part of arguments “-m app.storage” determines the “main source” of storage utilities, the following arguments are arguments of the app.storage script
- **--dir** option is preferred, it should point to storage.dir file with configuration of datasets; if this option is set, options --config, --source, --kind are out of sense
- **--config** option is only used if --dir is unset. By default it points to the default service configuration file, which is wrong if service is configured in a way that differs from the standard stand-alone mode.
- **--mode** option determines one of the following operations:
    - **create** dataset
    - **drop** dataset
      supplementary commands:
    - **druid-push** - reinstalls only the Druid part of xl-dataset
    - **doc-push** - reinstalls the documentation for dataset
- **--kind** option is set to ws by default, use xl if you need an xl-dataset, actual only if --dir is unset

- either **--inv** or **--source** option are required for the creation mode
    - **--inv** should point to inventory file of dataset located in directory of dataset, it is more convenient way to setup dataset, and the single way to attach documentation to dataset, we recommend use this variant
    - **--source** should point to a file with annotated records ready to be ingested into the Anfisa vault
      It understands the following formats: *.json, *.json.gz, *.json.bz2
      For collections of such files: use ‘*’ for patterning the collection. The files will be read in alphabetical order

- **--force** option is used if one needs to refill an existing dataset with a fresh version of source, and is out of use otherwise
- Name of database: in case of an xl-database it must begin with the prefix “xl_” or “XL_”. Secondary workspaces created by this xl-dataset will have the same name but with other prefixes, of the form “ws<*number*>_”.

Administration of MongoDB-based data: app.adm_mongo

This utility is used to maintain the portion of data that is set by the users of the system. It is recommended to run it from $ANFISA_HOME directory, or use the PYTHONPATH env variable to set this directory as python base.
Here are the options of the utility:

```
$ cd $ANFISA_HOME
$ python3 -m app.adm_mongo --help
usage: adm_mongo.py [-h] [-H HOST] [-P PORT] [-d DATABASE] [-c CONFIG] [-C]
```

```
                    command [command ...]

positional arguments:
  command               Commands, use help command for list

optional arguments:
  -h, --help            show this help message and exit
  -c CONFIG, --config CONFIG
                        Anfisa config file(anfisa.json), use it instead of
                        host/port/database
  -C, --config_default  Use it for config = ./anfisa.json
  -H HOST, --host HOST  MongoDB host
  -P PORT, --port PORT  MongoDB port
  -d DATABASE, --database DATABASE
                        Anfisa database in MongoDB
```

Comments:

- **--config** is used in the standard way to run the utility: just set path to the correct system configuration file (anfisa.json) here
- **--config_default** is a short way to set the system configuration file to "./anfisa.json" which is relevant in the default stand-alone mode
- **--HOST, --PORT, --DATABASE** can be set instead of the system configuration file to get access to a specific database in terms of MongoDB

The utility allows the system administrator to manipulate user data in terms of datasets: back them up, restore or drop. Output of any command is in JSON format.
The types of data provided by the utility currently cover only the needs of workspace maintenance: tagging of records, filters defined by users, and rules. Support of the data in XL-datasets is under development in the current state of the system and is not covered properly yet.

To see the complete list of supported commands use the command "help":

```
$ python3 -m app.adm_mongo help
 ===Anfisa/MongoDB administration tool===
 * List of commands *
      ds-list
      filter-list ds
      tag-list ds
      dump-filters ds
      dump-tags ds
      dump-rules ds
      load-tags ds datafile
      load-filters ds datafile
      load-rules ds datafile
      del-filter ds filter_name
      del-tag ds tag_name
      drop-filters ds
```

```
drop-tags ds
drop-rules ds
drop-ds ds
```

Comments:

- **ds** here means the name of a dataset
- **datafile** in the commands for rules is the path to a file where the data on rules are being stored or restored

Examples of utility use:

```
$ python3 -m app.adm_mongo -c anfisa.json ds-list
["PGP3140", "xl_PGP3140", "ws2_PGP3140", "ws1_PGP3140"]
$
$ python3 -m app.adm_mongo -c anfisa.json dump-rules PGP3140
[["exon_dist", 5], ["af", 0.01], ["af_seq_a_boo", 0.0007], ["af_popmax", 0.01],
["an_popmax", 2000], ["af_in_db", 0.05], ["severity", 1], ["gq", 20], ["fs",
30], ["qd", 4]]
```