# Bonsai

***Release 0.1***

**Markus Johansson**

**Apr 22, 2024**

# CONTENTS

CHAPTER

ONE

# INSTALLATION

You can use `docker-compose.yml` file to quickly install Bonsai for demo and development purposes. This requires that you have docker and *docker-compose* installed. Depending on your environment you might need to edit the docker-compose file, api config, and web client config files.

To install Bonsai, first clone the repository and start the softwares.

```
# clone Bonsai
git clone git@github.com:Clinical-Genomics-Lund/bonsai.git
# navigate to project directory
cd bonsai
# start the bonsai software stack
docker-compose up -d
```

Use the Bonsai api command line interface to create the required database indexes.

```
docker-compose exec api bonsai_api index
```

Create an admin user with the CLI. There are three built in user roles (*user*, *uploader*, and *admin*). The user role has permission to retrieve data and comment on isolates and should be the default user role. *Uploader* has permission to create and modify data but cannot view isoaltes, this role is inteded for uploading data to the database. The *admin* has full permission to view, create, modify and delete data.

```
docker-compose exec api bonsai_api create-user -u admin                \
                                               -p admin                \
                                               --fname Place           \
                                               --lname Holder          \
                                               -m place.holder@mail.com \
                                               -r admin
```

Use the upload_sample.py script to add analysis result and genome signature file to the database.

```
./scripts/upload_sample.sh                                              \
   --api localhost:8011                                                 \
   --group <optional: group_id of group to associate sample with>       \
   -u <username>                                                        \
   -p <password>                                                        \
   --input /path/to/input.json
```

## 1.1 Accessing the web interface

To access the web interface, access the URL https://localhost:8000 in your web browser.

(If this doesn't work, you might want to run `docker container ls` and make sure that a container based on the image `bonsai_frontend` is available in the list).

## 1.2 Data persistance

The data is not persitant between docker container updates by default as all data is kept in the container. You have to mount the mongo database and the API genome signature database to the host OS to make the data persitant. The volume mounts can be configured in the `docker-compose.yaml` file. If you mount the databases to the host OS you have to ensure that they have correct permissions so the container have read and write access to these files.

Use the following command to get the user and group id of the user in the container.

```
$ docker-compose run --rm mongodb id
# uid=1000(worker) gid=1000(worker) groups=1000(worker)
```

Use `chown -R /path/to/volume_dir 1000:1000` to change the permission of the folders you mount to the container.

The following are an example volume mount configuration. See the docker-compose documentation for more information on volume mounts.

```
services:
   mongodb:
      volumes:
         - "./volumes/mongodb:/data/db"

   api:
      volumes:
         - "./volumes/api/genome_signatures:/data/signature_db"
```

CHAPTER

TWO

# FREQUENTLY ASKED QUESTIONS

CHAPTER

THREE

# ADMINISTRATION

Users and admins administer Bonsai through either the API command line interface or the Bonsai front end. Bonsai has three predefined user roles (*user*, *uploader*, and *admin*) with different levels of privileges. The uploader role has only write permissions and is intended for automation scripts that upload new samples or create groups. A user with an uploader role cannot retrieve information from the database to minimize the threat vector.

The user can view samples and groups and post comments but not upload new results or manage sample groups. The user role should be enough for most users.

The admin user role grants full permission to create, modify, and read all data through the API entry points.

## 3.1 Index database

The database must be indexed for Bonsai to work correctly. The database indexes support and speed up common queries and enforces restriction on the data. For instance, will the indexes prevent duplicated sample IDs and sample group IDs? The indexes are created using the Bonsai API command line interface. Note that if you are running the containerized version of Bonsai, you must execute the commands in the container.

## 3.2 Create users

Users are also created through the API. You must specify the username, password, and user role when you create the user.

## 3.3 Upload samples

JASEN analysis results are uploaded to Bonsai using HTTP requests to the Bonsai API where the analysis result in JSON format and signature file is uploaded separately. Bonsai includes a simple upload bash script that can upload a sample and assign it to a group. The script is located in the scripts directory.

```
./scripts/upload_sample.sh                                          \
   --api localhost:8011                                             \
   --group <optional: group_id of group to associate sample with> \
   -u <username>                                                    \
   -p <password>                                                    \
   --input /path/to/input.json
```

The script demonstrates basic sample management using the API routes and these could be included in your automations for sample processing.

## 3.4 Create and manage groups of samples

Groups can be created and managed through the front end by admin users. Groups can be created and modified from the */groups/edit* view. Groups can also be managed through the API, which allows for automation and/ or integration into sample processing pipelines.

CHAPTER

FOUR

# DEPLOYMENT INTO A PRODUCTION ENVIRONMENT

The following should be done when deploying Bonsai to a production environment.

1. Change the *SECRET_KEY* variable in the Bonsai app and API configuration. Use a production-ready WSGI server such as gunicorn instead of the default Flask development server. 2. Configure the *BONSAI_API_URL* ` in the frontend config or set it as an environmental variable. 3. It should be set to the URL where you host the Bonsai API.

You should use SSL certificates in a production environment to protect the data sent from being intercepted. We also recommend enabling authentication to your Mongo database and setting up specific accounts for database administrators and software to protect your data. See the Mongo documentation for more information on authentication and authorization mechanisms.

CHAPTER

FIVE

# BONSAI DEVELOPER DOCUMENTATION

Bonsai consists of a front end, an API layer, and several services that perform long-running or computationally intensive tasks. The front end uses the API to query or update the database and to create jobs for the services to execute. Redis is used as a queue and for caching the results of API queries. The services listen for submitted jobs, perform the task, and return the results to Redis. The front end can then request status updates of submitted jobs using the unique job id it received when it requested the job.

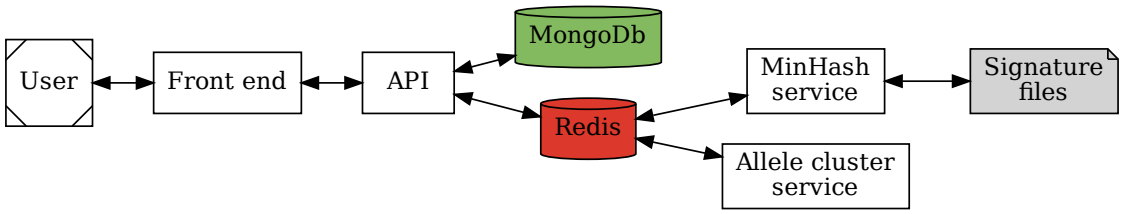


Fig. 1: Overview of Bonsai software stack.

## 5.1 Services

### 5.1.1 MinHash clustering

The MinHash clustering service subscribes to the Redis database and listens for jobs on the *minhash* queue.

#### Tasks

minhash_service.tasks.**add_signature**(*sample_id: str*, *signature*) → str

Find signatures similar to reference signature.

**Parameters**

- **str** (*sample_id*) – the sample_id
- **str]** (*signature Dict[str,*) – sourmash signature file in JSON format

**Returns**

path to the signature

**Return type**
str

minhash_service.tasks.**remove_signature**(*sample_id: str*) → Dict[str, str | bool]

Remove a signature from the database and index.

**Parameters**
**str** (*sample_id*) – the sample_id of the signature to remove

**Returns**
The status of the removed job

**Return type**
Dict[str, str | bool]

minhash_service.tasks.**similar**(*sample_id: str*, *min_similarity: float = 0.5*, *limit: int | None = None*) → List[SimilarSignature]

Find signatures similar to reference signature.

**Parameters**

- **str** (*sample_id*) – The id of reference sample
- **float** (*min_similarity*) – Minimum similarity score
- **None** (*limit int |*) – Limit the result to x samples, default to None

**Returns**
list of the similar signatures

**Return type**
SimilarSignatures

minhash_service.tasks.**cluster**(*sample_ids: List[str]*, *cluster_method: str = 'single'*) → str

Cluster multiple sample on their sourmash signatures.

**Parameters**

- **List[str]** (*sample_ids*) – The sample ids to cluster
- **int** (*cluster_method*) – The linkage or clustering method to use, default to single

**Raises**
**ValueError** – raises an exception if the method is not a valid MSTree clustering method.

**Returns**
clustering result in newick format

**Return type**
str

minhash_service.tasks.**find_similar_and_cluster**(*sample_id: str*, *min_similarity: float = 0.5*, *limit: int | None = None*, *cluster_method: str = 'single'*) → str

Find similar samples and cluster them on their minhash profile.

**Parameters**

- **str** (*sample_id*) – The id of reference sample
- **float** (*min_similarity*) – Minimum similarity score
- **None** (*limit int |*) – Limit the result to x samples, default to None
- **int** (*cluster_method*) – The linkage or clustering method to use, default to single

**Raises**
ValueError – raises an exception if the method is not a valid MSTree clustering method.

**Returns**
clustering result in newick format

**Return type**
str

### 5.1.2 Allele clustering service

The allele clustering service subscribes to the Redis database and listens for jobs on the *allele_clustering* queue. The exposed functions are located in *tasks.py*.

#### Tasks

allele_cluster_service.tasks.**cluster**(*profile: str*, *method: str*) → str

Cluster multiple sample on their allele profiles.

**Parameters**

- **str** (*method*) – a string representation of a tsv table of the allele profiles
- **str** – the MStree clustering method

**Raises**
ValueError – raises an exception if the method is not a valid MSTree clustering method.

**Returns**
cluster in newick format

**Return type**
str

CHAPTER

SIX

# LOGIN SYSTEMS

Bonsai currently support two login systems, LDAP authentication and a simple username and password authentication. These login sytems are mutually exclusive so by chosing one the other is being disabled.

## 6.1 Simple authentication

The default authentication method is logging using a username and password. The password is being salted and stored in the mongo database.

## 6.2 LDAP authentication

Authentication can be made against an institutional LDAP3 server. You need an existing LDAP authentication server to user this authentication method. The users need to have an account in Bonsai in addition to having an entry on the LDAP server, because the user roles are determined from the Bonsai account.

Use either the API CLI or the Admin panel to create a new user.

The LDAP3 server connection is configured using environmental variables. At minimum you need to configure the following variables.

Table 1: Minimal configuration `csv-table`

| Variable name | Description |
|---|---|
| LDAP_HOST | Server host or IP |
| LDAP_PORT | Server port |
| LDAP_BASE_DN | Base distinguished name (DN) for searching the server |
| LDAP_BIND_DN | Admin bind DN |
| LDAP_SECRET | Optional password for the admin bind DN |
| LDAP_SEARCH_ATTR | Attribute to validate username against |

See the *config.py* for all variables that can be configured using environment variables.

### 6.2.1 Example configuration

Here is a example of an LDAP based authentication configuration using docker-compose. We use a demo LDAP server populated with Futurama characters.

Listing 1: Example configuration

```
version: '3.9'
# usage:
# (sudo) docker-compose up -d
# (sudo) docker-compose down
services:
   mongodb:
      image: mongo:4.4.22
      ports:
         - "8813:27017"
      expose:
         - "27017"
      volumes:
         - "./volumes/mongodb:/data/db"
      networks:
         - bonsai-net

   redis:
      image: redis:7.0.10
      networks:
         - bonsai-net

   openldap:
      image: ghcr.io/rroemhild/docker-test-openldap:master
      container_name: openldap
      ports:
         - "10389:10389"
         - "10636:10636"
      networks:
         - bonsai-net
      privileged: true

   api:
      container_name: bonsai_api
      build:
         context: api
         network: host
      depends_on:
         - mongodb
      ports:
         - "8811:8000"
      environment:
         - DB_HOST=mongodb
         - REDIS_HOST=redis
         - LDAP_HOST=openldap
         - LDAP_PORT=10389
         - LDAP_BIND_DN=cn=admin,dc=planetexpress,dc=com
```

(continues on next page)

(continued from previous page)

```
        - LDAP_SECRET=GoodNewsEveryone
        - LDAP_BASE_DN=dc=planetexpress,dc=com
        - LDAP_USER_LOGIN_ATTR=mail
        - LDAP_USE_SSL=false
        - LDAP_USE_TLS=false
    networks:
        - bonsai-net
    command: "uvicorn app.main:app --reload --log-level debug --host 0.0.0.0"

  app:
    container_name: bonsai_app
    build:
      context: frontend
      network: host
    depends_on:
        - mongodb
        - api
    ports:
        - "8812:5000"
    environment:
        - FLASK_APP=app.app:create_app
        - FLASK_ENV=development
        - "BONSAI_API_URL=http://mtlucmds2.lund.skane.se:8811"
    networks:
        - bonsai-net
    command: "flask run --debug --host 0.0.0.0"

networks:
  bonsai-net:
    driver: bridge
    ipam:
      driver: default
      config:
      - subnet: 172.0.30.0/24
```

CHAPTER

SEVEN

# CHANGELOG

Starting with version `2.0` will the project use semantic versioning.

## 7.1 [Unreleased]

### 7.1.1 Added

### 7.1.2 Changed

- Show disabled IGV button for samples without BAM or reference genome.

### 7.1.3 Fixed

## 7.2 [v0.4.1]

### 7.2.1 Added

- Display LIMS id in samples view

### 7.2.2 Changed

- Sample name is being displayed instead of sample id on the samples view
- Dockerfile chown step for api and frontend

### 7.2.3 Fixed

- Fix minhash sample id lookup by storing sample_id as signautre name when signature is written to disk.
- Links to a sample from the samples tables now works when Bonsai is hosted under a sub-path
- Fixed so the samples could be added to the minhash index
- Fixed nameing of signature sketches and updating filename
- Fixed broken URL that prevented finding similar samples
- Fixed storing of selected samples in browser session that prevented samples to be added to the basket from the groups view.

- Fixed broken URLs in dendrogram in samples view
- Fixed crash when reading empty sourmash index
- Fixed crash in resistance/variants view when a samples did not have SNVs or SV variants

## 7.3 [v0.4.0]

### 7.3.1 Added

- Added Sample name, LIMS ID, and Sequencing run as selectable columns
- Sample name in sample view table links to sample
- New upload script (`upload_sample.py`) that takes a upload config in YAML format as input

### 7.3.2 Fixed

### 7.3.3 Changed

- Sample id is assigend by concating `lims_id` and `sequencing_run`
- Sample id is not displayed by default
- API route POST /samples/ returns `sample_id`
- Removed `upload_sample.sh`

## 7.4 [v0.3.0]

### 7.4.1 Added

- Add button to remove samples from the group- and groups view.
- Added view for analyszing variants with filtering
- Added IGV genome browser integration to variant analysis view.
- Bonsai support display of SV and SNV variants.
- A user can classify variant as accepted or as rejected based and annotate why it was dissmissed.
- A user can annotate that a variant yeilds resistance to additional anitbiotics
- Placeholder Export to LIMBS button to the sidebar in the variants view
- Added CLI command for exporting AMR prediction to a LIMS tsv file

### 7.4.2 Fixed

- 500 error when trying to get a sample removed from the database
- Frontend properly handles non-existing samples and group
- Fixed typo in similar samples card that caused invalid URLs
- Fixed default fontend config to work with default docker-compose file

### 7.4.3 Changed

- Removed "passed qc" column from tbprofiler result
- Changed default app port to 8000 and api port to 8001

## 7.5 [v0.2.1]

### 7.5.1 Added

- Bulk QC status dropdown in group view

### 7.5.2 Fixed

- Fixed crash when clustering on samples without a MLST profile
- Fixed bug that prevented adding samples to the basket in groups without "analysis profile" column
- Fixed issue that prevented finding similar samples in group view

### 7.5.3 Changed

- Display the sum of kraken assigned reads and added reads in spp card by default.
- Froze uvicorn to version 0.25.0
- Updated fastapi to version 0.108.0

## 7.6 [v0.2.0]

### 7.6.1 Added

- Improved output of create_user API CLI command
- bonsai_api create-user command have options for mail, first name and last name.
- Open samples by clicking on labels in the similar samples card in the samples view.
- Optional "extended" HTTP argument to sample view to view extended prediction info

### 7.6.2 Fixed

- Fixed crash in create_user API CLI command
- Resistance_report now render work in progress page
- Removed old project name from GrapeTree header
- Fixed issue that prevented node labels in GrapeTree from being displayed.

### 7.6.3 Changed

- The role "user" have permission to comment and classify QC
- Updated PRP to version 0.3.0

## 7.7 [v0.1.0]

### 7.7.1 Added

- Find similar samples by calculating MinHash distance
- Added async similarity searches and clustering
- User can choose clustering method from the sample basket
- Added spp prediction result to sample view
- Find similar samples default to 50
- Added STX typing for *Escherichia coli* samples
- CLI command for updating tags for all samples

### 7.7.2 Fixed

- Fixed calculation of missing and novel cgmlst alleles
- Fixed so sample metadata is displayed in GrapeTree

### 7.7.3 Changed

- Renamed client to frontent and server to api
- Renamed cgvis to Bonsai
- Complete rewrite of the project
- Removed N novel cgmlst alleles from cgmlst qc card
- Use data models from JASEN Pipleine Result Processing application
- Use pre-defined table columns in edit groups view

## 7.8 Previous release

Starting with version `2.0`, the project was renamed from cgvis to Bonsai.

Analyze outbreak, antimicrobial resistance and virulence factors in bacteria.

**Note:** This project is under active development and Bonaai and the documentation is likely to change.

Intended to visualize results from JASEN pipeline.

Check out the *Installation* section for installation instructions.

# INDEX