Bakta

O. Schwengers, L. Jelonek, M. Dieckmann, S. Beyvers

Aug 01, 2022

CONTENTS

Bakta is a tool for the rapid & standardized annotation of bacterial genomes & plasmids. It provides **dbxref**-rich and **sORF**-including annotations in machine-readable `JSON` & bioinformatics standard file formats for automatic downstream analysis.

This documentations provides information how to install and use Bakta via CLI, web and REST-API.

CHAPTER

ONE

BAKTA CLI

1.1 Contents

1.2 Installation

Bakta can be installed via BioConda, Docker, Singularity and Pip. However, we encourage to use Conda or Docker/Singularity to automatically install all required 3rd party dependencies.

In all cases a mandatory *database* must be downloaded.

1.2.1 BioConda

```
$ conda install -c conda-forge -c bioconda bakta
```

1.2.2 Docker

```
$ sudo docker pull oschwengers/bakta
$ sudo docker run oschwengers/bakta --help
```

Installation instructions and get-started guides: Docker docs

For further convenience, we provide a shell script (`bakta-docker.sh`) handling Docker related parameters (volume mounting, user IDs, etc):

```
$ bakta-docker.sh --db <db-path> --output <output-path> <input>
```

1.2.3 Singularity

```
$ singularity build bakta.sif docker://oschwengers/bakta:latest
$ singularity run bakta.sif --help
```

Installation instructions, get-started and guides: Singularity docs

1.2.4 Pip

```
$ python3 -m pip install --user bakta
```

Bacta requires the following 3rd party executables which must be installed & executable:

- tRNAscan-SE (2.0.8) https://doi.org/10.1101/614032 http://lowelab.ucsc.edu/tRNAscan-SE
- Aragorn (1.2.38) http://dx.doi.org/10.1093/nar/gkh152 http://130.235.244.92/ARAGORN
- INFERNAL (1.1.4) https://dx.doi.org/10.1093%2Fbioinformatics%2Fbtt509 http://eddylab.org/infernal
- PILER-CR (1.06) https://doi.org/10.1186/1471-2105-8-18 http://www.drive5.com/pilercr
- Prodigal (2.6.3) https://dx.doi.org/10.1186%2F1471-2105-11-119 https://github.com/hyattpd/Prodigal
- Hmmer (3.3.2) https://doi.org/10.1093/nar/gkt263 http://hmmer.org
- Diamond (2.0.14) https://doi.org/10.1038/nmeth.3176 https://github.com/bbuchfink/diamond
- Blast+ (2.12.0) https://www.ncbi.nlm.nih.gov/pubmed/2231712 https://blast.ncbi.nlm.nih.gov
- AMRFinderPlus (3.10.23) https://github.com/ncbi/amr
- DeepSig (1.2.5) https://doi.org/10.1093/bioinformatics/btx818

1.2.5 Database download

Bakta requires a mandatory database which is publicly hosted at Zenodo: Further information is provided in the *database* section below.

List available DB versions:

```
$ bakta_db list
...
```

Download the most recent compatible database version we recommend to use the internal database download & setup tool:

```
$ bakta_db download --output <output-path>
```

Of course, the database can also be downloaded manually:

```
$ wget https://zenodo.org/record/5215743/files/db.tar.gz
$ tar -xzf db.tar.gz
$ rm db.tar.gz
$ amrfinder_update --force_update --database db/amrfinderplus-db/
```

In this case, please also download the AMRFinderPlus database as indicated above.

Update an existing database:

```
$ bakta_db update --db <existing-db-path> [--tmp-dir <tmp-directory>]
```

The database path can be provided either via parameter (`--db`) or environment variable (`BAKTA_DB`):

```
$ bakta --db <db-path> genome.fasta

$ export BAKTA_DB=<db-path>
$ bakta genome.fasta
```

For system-wide setups, the database can also be copied to the Bakta base directory:

```
$ cp -r db/ <bakta-installation-dir>
```

As Bakta takes advantage of AMRFinderPlus for the annotation of AMR genes, AMRFinder is required to setup its own internal databases in a `<amrfinderplus-db>` subfolder within the Bakta database `<db-path>`, once via `amrfinder_update --force_update --database <db-path>/amrfinderplus-db/`. To ease this process we recommend to use Bakta's internal download procedure.

1.3 Examples

Simple:

```
$ bakta --db <db-path> genome.fasta
```

Expert: verbose output writing results to *results* directory with *ecoli123* file `prefix` and *eco634* `locus tag` using an existing prodigal training file, using additional replicon information and 8 threads:

```
$ bakta --db <db-path> --verbose --output results/ --prefix ecoli123 --locus-tag eco634 --
↪prodigal-tf eco.tf --replicons replicon.tsv --threads 8 genome.fasta
```

1.4 Input and Output

1.4.1 Input

Bakta accepts bacterial genomes and plasmids (complete / draft assemblies) in (zipped) fasta format. For a full description of how further genome information can be provided and workflow customizations can be set, please have a look at the *Usage* section.

Replicon meta data table:

To fine-tune the very details of each sequence in the input fasta file, Bakta accepts a replicon meta data table provided in `csv` or `tsv` file format: `--replicons <file.tsv>`. Thus, complete replicons within partially completed draft assemblies can be marked & handled as such, *e.g.* detection & annotation of features spanning sequence edges.

Table format:

original sequence id	new sequence id	type	topology	name
`old id`	[`new id` / `<empty>`]	[`chromosome` / `plasmid` / `contig` / `<empty>`]	[`circular` / `linear` / `<empty>`]	[`name` / `<empty>`]

For each input sequence recognized via the `original locus id` a `new locus id`, the replicon `type` and the `topology` as well a `name` can be explicitly set.

Shortcuts:

- `chromosome`: `c`
- `plasmid`: `p`
- `circular`: `c`
- `linear`: `l`

<empty> values (`-` / ``) will be replaced by defaults. If **new locus id** is `empty`, a new contig name will be autogenerated.

Defaults:

- type: `contig`
- topology: `linear`

Example:

original locus id	new locus id	type	topology	name
NODE_1	chrom	`chromosome`	`circular`	`-`
NODE_2	p1	`plasmid`	`c`	`pXYZ1`
NODE_3	p2	`p`	`c`	`pXYZ2`
NODE_4	special-contig-name-xyz	`-`	-	
NODE_5	``	`-`	-	

User provided protein sequences

Bakta accepts user provided trusted protein sequences via `--proteins` in either GenBank (CDS features) or Fasta format. Using the Fasta format, each reference sequence can be provided in a short or long format:

```
# short:
>id gene~~~product~~~dbxrefs
MAQ...

# long:
>id min_identity~~~min_query_cov~~~min_subject_cov~~~gene~~~product~~~dbxrefs
MAQ...
```

Allowed values:

field	value(s)	example
min_identity	`int`, `float`	80, 90.3
min_query_cov	`int`, `float`	80, 90.3
min_subject_cov	`int`, `float`	80, 90.3
gene	`<empty>`, `string`	msp
product	`string`	my special protein
dbxrefs	`<empty>`, `db:id`, `,` separated list	`VFDB:VF0511`

Protein sequences provided in short Fasta or GenBank format are searched with default thresholds of 90%, 80% and 80% for minimal identity, query and subject coverage, respectively.

1.4.2 Output

Annotation results are provided in standard bioinformatics file formats:

- `<prefix>.tsv`: annotations as simple human readble TSV
- `<prefix>.gff3`: annotations & sequences in GFF3 format
- `<prefix>.gbff`: annotations & sequences in (multi) GenBank format
- `<prefix>.embl`: annotations & sequences in (multi) EMBL format
- `<prefix>.fna`: replicon/contig DNA sequences as FASTA
- `<prefix>.ffn`: feature nucleotide sequences as FASTA
- `<prefix>.faa`: CDS/sORF amino acid sequences as FASTA
- `<prefix>.hypotheticals.tsv`: further information on hypothetical protein CDS as simple human readble tab separated values
- `<prefix>.hypotheticals.faa`: hypothetical protein CDS amino acid sequences as FASTA
- `<prefix>.txt`: summary as TXT

The `<prefix>` can be set via `--prefix <prefix>`. If no prefix is set, Bakta uses the input file prefix.

Additionally, Bakta provides detailed information on each annotated feature in a standardized machine-readable JSON file `<prefix>.json`:

```
{
    "genome": {
        "genus": "Escherichia",
        "species": "coli",
        ...
    },
    "stats": {
        "size": 5594605,
        "gc": 0.497,
        ...
    },
    "features": [
        {
            "type": "cds",
            "contig": "contig_1",
            "start": 971,
            "stop": 1351,
            "strand": "-",
            "gene": "lsoB",
            "product": "type II toxin-antitoxin system antitoxin LsoB",
            ...
        },
        ...
    ],
    "sequences": [
        {
            "id": "c1",
            "description": "[organism=Escherichia coli] [completeness=complete]␣
↪[topology=circular]",
            "sequence": "AGCTTT...",
            "length": 5498578,
            "complete": true,
            "type": "chromosome",
            "topology": "circular"
            ...
        },
        ...
    ]
}
```


Exemplary annotation result files for several genomes (mostly ESKAPE species) are hosted at Zenodo:

1.5 Usage

Usage:

```
usage: bakta [--db DB] [--min-contig-length MIN_CONTIG_LENGTH] [--prefix PREFIX] [--
↪output OUTPUT]
             [--genus GENUS] [--species SPECIES] [--strain STRAIN] [--plasmid PLASMID]
             [--complete] [--prodigal-tf PRODIGAL_TF] [--translation-table {11,4}] [--
↪gram {+,-,?}] [--locus LOCUS]
```

(continues on next page)

(continued from previous page)

```
            [--locus-tag LOCUS_TAG] [--keep-contig-headers] [--replicons REPLICONS] [--
↪compliant] [--proteins PROTEINS]
            [--skip-trna] [--skip-tmrna] [--skip-rrna] [--skip-ncrna] [--skip-ncrna-
↪region]
            [--skip-crispr] [--skip-cds] [--skip-sorf] [--skip-gap] [--skip-ori]
            [--help] [--verbose] [--threads THREADS] [--tmp-dir TMP_DIR] [--version]
            <genome>

Rapid & standardized annotation of bacterial genomes, MAGs & plasmids

positional arguments:
  <genome>              Genome sequences in (zipped) fasta format

Input / Output:
  --db DB, -d DB        Database path (default = <bakta_path>/db). Can also be provided␣
↪as BAKTA_DB environment variable.
  --min-contig-length MIN_CONTIG_LENGTH, -m MIN_CONTIG_LENGTH
                        Minimum contig size (default = 1; 200 in compliant mode)
  --prefix PREFIX, -p PREFIX
                        Prefix for output files
  --output OUTPUT, -o OUTPUT
                        Output directory (default = current working directory)

Organism:
  --genus GENUS         Genus name
  --species SPECIES     Species name
  --strain STRAIN       Strain name
  --plasmid PLASMID     Plasmid name

Annotation:
  --complete            All sequences are complete replicons (chromosome/plasmid[s])
  --prodigal-tf PRODIGAL_TF
                        Path to existing Prodigal training file to use for CDS prediction
  --translation-table {11,4}
                        Translation table: 11/4 (default = 11)
  --gram {+,-,?}        Gram type for signal peptide predictions: +/-/? (default = '?')
  --locus LOCUS         Locus prefix (default = 'contig')
  --locus-tag LOCUS_TAG
                        Locus tag prefix (default = autogenerated)
  --keep-contig-headers
                        Keep original contig headers
  --replicons REPLICONS, -r REPLICONS
                        Replicon information table (tsv/csv)
  --compliant           Force Genbank/ENA/DDJB compliance
  --proteins PROTEINS   Fasta file of trusted protein sequences for CDS annotation

Workflow:
  --skip-trna           Skip tRNA detection & annotation
  --skip-tmrna          Skip tmRNA detection & annotation
  --skip-rrna           Skip rRNA detection & annotation
  --skip-ncrna          Skip ncRNA detection & annotation
  --skip-ncrna-region   Skip ncRNA region detection & annotation
```

(continues on next page)

(continued from previous page)

```
  --skip-crispr         Skip CRISPR array detection & annotation
  --skip-cds            Skip CDS detection & annotation
  --skip-pseudo         Skip pseudogene detection & annotation
  --skip-sorf           Skip sORF detection & annotation
  --skip-gap            Skip gap detection & annotation
  --skip-ori            Skip oriC/oriT detection & annotation

General:
  --help, -h            Show this help message and exit
  --verbose, -v         Print verbose information
  --threads THREADS, -t THREADS
                        Number of threads to use (default = number of available CPUs)
  --tmp-dir TMP_DIR     Location for temporary files (default = system dependent auto␣
↪detection)
  --version             show program's version number and exit
```

1.6 Annotation Workflow

1.6.1 RNAs

1. tRNA genes: tRNAscan-SE 2.0
2. tmRNA genes: Aragorn
3. rRNA genes: Infernal vs. Rfam rRNA covariance models
4. ncRNA genes: Infernal vs. Rfam ncRNA covariance models
5. ncRNA cis-regulatory regions: Infernal vs. Rfam ncRNA covariance models
6. CRISPR arrays: PILER-CR

Bakta distinguishes ncRNA genes and (cis-regulatory) regions in order to enable the distinct handling thereof during the annotation process, *i.e.* feature overlap detection.

ncRNA gene types:

- sRNA
- antisense
- ribozyme
- antitoxin

ncRNA (cis-regulatory) region types:

- riboswitch
- thermoregulator
- leader
- frameshift element

1.6.2 Coding sequences

The structural prediction is conducted via Prodigal and complemented by a custom detection of sORF < 30 aa.

To rapidly identify known protein sequences with exact sequence matches and to conduct a comprehensive annotations, Bakta utilizes a compact read-only SQLite database comprising protein sequence digests and pre-assigned annotations for millions of known protein sequences and clusters.

Conceptual terms:

- **UPS**: unique protein sequences identified via length and MD5 hash digests (100% coverage & 100% sequence identity)
- **IPS**: identical protein sequences comprising seeds of UniProt's UniRef100 protein sequence clusters
- **PSC**: protein sequences clusters comprising seeds of UniProt's UniRef90 protein sequence clusters
- **PSCC**: protein sequences clusters of clusters comprising annotations of UniProt's UniRef50 protein sequence clusters

CDS:

1. Prediction via Prodigal respecting sequences' completeness (distinct prediction for complete replicons and uncompleted contigs)
2. Discard spurious CDS via AntiFam
3. Detect translational exceptions (selenocysteines)
4. Detection of UPSs via MD5 digests and lookup of related IPS and PCS
5. Sequence alignments of remainder via Diamond vs. PSC (query/subject coverage=0.8, identity=0.5)
6. Assignment to UniRef90 or UniRef50 clusters if alignment hits achieve identities larger than 0.9 or 0.5, respectively
7. Execution of expert systems:

- AMR: AMRFinderPlus
- Expert proteins: NCBI BlastRules, VFDB
- User proteins (optionally via `--proteins <Fasta/GenBank>`)

1. Prediction of signal peptides (optionally via `--gram <+/->`)
2. Combination of IPS, PSC, PSCC and expert system information favouring more specific annotations and avoiding redundancy

CDS without IPS or PSC hits as well as those without gene symbols or product descriptions different from `hypothetical` will be marked as `hypothetical`.

Such hypothetical CDS are further analyzed:

1. Detection of Pfam domains, repeats & motifs
2. Calculation of protein sequence statistics, *i.e.* molecular weight, isoelectric point

sORFs:

1. Custom sORF detection & extraction with amino acid lengths < 30 aa
2. Apply strict feature type-dependent overlap filters
3. discard spurious sORF via AntiFam
4. Detection of UPS via MD5 hashes and lookup of related IPS

5. Sequence alignments of remainder via Diamond vs. an sORF subset of PSCs (coverage=0.9, identity=0.9)
6. Exclude sORF without sufficient annotation information
7. Prediction of signal peptides (optionally via `--gram <+/->`)

sORF not identified via IPS or PSC will be discarded. Additionally, all sORF without gene symbols or product descriptions different from `hypothetical` will be discarded. Due due to uncertain nature of sORF prediction, only those identified via IPS / PSC hits exhibiting proper gene symbols or product descriptions different from `hypothetical` will be included in the final annotation.

1.6.3 Miscellaneous

1. Gaps: in-mem detection & annotation of sequence gaps
2. oriC/oriV/oriT: Blast+ (cov=0.8, id=0.8) vs. MOB-suite oriT & DoriC oriC/oriV sequences. Annotations of ori regions take into account overlapping Blast+ hits and are conducted based on a majority vote heuristic. Region edges are fuzzy - use with caution!

1.7 Database

The Bakta database comprises a set of AA & DNA sequence databases as well as HMM & covariance models. At its core Bakta utilizes a compact read-only SQLite db storing protein sequence digests, lengths, pre-assigned annotations and dbxrefs of UPS, IPS and PSC from:

- **UPS**: UniParc / UniProtKB (213,429,161)
- **IPS**: UniProt UniRef100 (198,494,206)
- **PSC**: UniProt UniRef90 (91,455,850)
- **PSCC**: UniProt UniRef50 (12,323,024)

This allows the exact protein sequences identification via MD5 digests & sequence lengths as well as the rapid subsequent lookup of related information. Protein sequence digests are checked for hash collisions while the db creation process. IPS & PSC have been comprehensively pre-annotated integrating annotations & database *dbxrefs* from:

- NCBI nonredundant proteins (IPS: 153,166,049)
- NCBI COG db (PSC: 3,383,871)
- SwissProt EC/GO terms (PSC: 335,068)
- NCBI AMRFinderPlus (IPS: 6,534, PSC: 48,931)
- ISFinder db (IPS: 45,820, PSC: 10,351)
- Pfam families (PSC: 3,917,555)

To provide high quality annotations for distinct protein sequences of high importance (AMR, VF, *etc*) which cannot sufficiently be covered by the IPS/PSC approach, Bakta provides additional expert systems. For instance, AMR genes, are annotated via NCBI's AMRFinderPlus. An expandable alignment-based expert system supports the incorporation of high quality annotations from multiple sources. This currenlty comprises NCBI's BlastRules as well as VFDB and will be complemented with more expert annotation sources over time. Internally, this expert system is based on a Diamond DB comprising the following information in a standardized format:

- source: *e.g.* BlastRules
- rank: a precedence rank
- min identity

- min query coverage
- min model coverage
- gene lable
- product description
- dbxrefs

Rfam covariance models:

- ncRNA: 798
- ncRNA cis-regulatory regions: 270

ori sequences:

- oriC/V: 10,878
- oriT: 502

To provide FAIR annotations, the database releases are SemVer versioned (w/o patch level), *i.e.* `<major>.<minor>`. For each version we provide a comprehensive log file tracking all imported sequences as well as annotations thereof. The db schema is represented by the `<major>` digit and automatically checked at runtime by Bakta in order to ensure compatibility. Content updates are tracked by the `<minor>` digit.

All database releases (latest 3.1, 28 Gb zipped, 53 Gb unzipped) are hosted at Zenodo:

1.8 Genome Submission

Most genomes annotated with Bakta should be ready-to-submid to INSDC member databases GenBank and ENA. As a first step, please register your BioProject (e.g. PRJNA123456) and your locus_tag prefix (*e.g.* ESAKAI).

```
# annotate your genome in `--compliant` mode:
$ bakta --db <db-path> -v --genus Escherichia --species "coli O157:H7" --strain Sakai --
↪complete --compliant --locus-tag ESAKAI test/data/GCF_000008865.2.fna.gz
```

1.8.1 GenBank

Genomes are submitted to GenBank via Fasta (`.fna`) and SQN files. Therefore, `.sqn` files can be created via `.gff3` files and NCBI's new table2asn_GFF tool. Please have all additional files (template.txt) prepared:

```
# download table2asn_GFF for Linux
$ wget https://ftp.ncbi.nih.gov/toolbox/ncbi_tools/converters/by_program/table2asn_GFF/
↪linux64.table2asn_GFF.gz
$ gunzip linux64.table2asn_GFF.gz

# or MacOS
$ https://ftp.ncbi.nih.gov/toolbox/ncbi_tools/converters/by_program/table2asn_GFF/mac.
↪table2asn_GFF.gz
$ gunzip mac.table2asn_GFF.gz
$ chmod 755 linux64.table2asn_GFF.gz mac.table2asn_GFF.gz

# create the SQN file:
$ linux64.table2asn_GFF -M n -J -c w -t template.txt -V vbt -l paired-ends -i GCF_
↪000008865.2.fna -f GCF_000008865.2.gff3 -o GCF_000008865.2.sqn -Z
```

1.8.2 ENA

Genomes are submitted to ENA as EMBL (`.embl`) files via EBI's Webin-CLI tool. Please have all additional files (manifest.tsv, chrom-list.tsv) prepared as described here.

```
# download ENA Webin-CLI
$ wget https://github.com/enasequence/webin-cli/releases/download/v4.0.0/webin-cli-4.0.0.
↪jar

$ gzip -k GCF_000008865.2.embl
$ gzip -k chrom-list.tsv
$ java -jar webin-cli-4.0.0.jar -submit -userName=<EMAIL> -password <PWD> -context␣
↪genome -manifest manifest.tsv
```

Exemplarey manifest.tsv and chrom-list.tsv files might look like:

```
$ cat chrom-list.tsv
STUDY    PRJEB44484
SAMPLE    ERS6291240
ASSEMBLYNAME    GCF
ASSEMBLY_TYPE    isolate
COVERAGE    100
PROGRAM    SPAdes
PLATFORM    Illumina
MOLECULETYPE    genomic DNA
FLATFILE    GCF_000008865.2.embl.gz
CHROMOSOME_LIST    chrom-list.tsv.gz

$ cat chrom-list.tsv
contig_1    contig_1    circular-chromosome
contig_2    contig_2    circular-plasmid
contig_3    contig_3    circular-plasmid
```

1.9 Protein bulk annotation

For the direct bulk annotation of protein sequences aside from the genome, Bakta provides a dedicated CLI entry point `bakta_proteins`:

Examples:

```
$ bakta_proteins --db <db-path> input.fasta

$ bakta_proteins --db <db-path> --prefix test --output test --proteins special.faa --
↪threads 8 input.fasta
```

1.9.1 Output

Annotation results are provided in standard bioinformatics file formats:

- <prefix>.tsv: annotations as simple human readble TSV
- <prefix>.faa: protein sequences as FASTA
- <prefix>.hypotheticals.tsv: further information on hypothetical proteins as simple human readble tab separated values

The <prefix> can be set via --prefix <prefix>. If no prefix is set, Bakta uses the input file prefix.

1.9.2 Usage

```
usage: bakta_proteins [--db DB] [--output OUTPUT] [--prefix PREFIX] [--proteins␣
↪PROTEINS] [--help] [--threads THREADS] [--tmp-dir TMP_DIR] [--version] <input>

Rapid & standardized annotation of bacterial genomes, MAGs & plasmids

positional arguments:
  <input>               Protein sequences in (zipped) fasta format

Input / Output:
  --db DB, -d DB        Database path (default = <bakta_path>/db). Can also be provided␣
↪as BAKTA_DB environment variable.
  --output OUTPUT, -o OUTPUT
                        Output directory (default = current working directory)
  --prefix PREFIX, -p PREFIX
                        Prefix for output files

Annotation:
  --proteins PROTEINS   Fasta file of trusted protein sequences for annotation

Runtime & auxiliary options:
  --help, -h            Show this help message and exit
  --threads THREADS, -t THREADS
                        Number of threads to use (default = number of available CPUs)
  --tmp-dir TMP_DIR     Location for temporary files (default = system dependent auto␣
↪detection)
  --version, -V         show program's version number and exit
```

1.10 FAQ

- **AMRFinder fails** If AMRFinder constantly crashes even on fresh setups and Bakta's database was downloaded manually, then AMRFinder needs to setup its own internal database. This is required only once: amrfinder_update --force_update --database <bakta-db>/amrfinderplus-db. You could also try Bakta's internal database download logic automatically taking care of this: bakta_db download --output <bakta-db>
- **DeepSig not found in Conda environment** For the prediction of signal predictions, Bakta uses DeepSig that is currently not available for MacOS. Therefore, we decided to exclude DeepSig from Bakta's default Conda

dependencies because otherwise it would not be installable on MacOS systems. On Linux systems it can be installed via `conda install -c conda-forge -c bioconda python=3.8 deepsig`.

- **Nice, but I'm mising XYZ…** Bakta is quite new and we're keen to constantly improve it and further expand its feature set. In case there's anything missing, please do not hesitate to open an issue and ask for it!
- **Bakta is running too long without CPU load… why?** Bakta takes advantage of an SQLite DB which results in high storage IO loads. If this DB is stored on a remote / network volume, the lookup of IPS/PSC annotations might take a long time. In these cases, please, consider moving the DB to a local volume or hard drive.

1.11 Issues and Feature Requests

Bakta is brand new and like in every software, expect some bugs lurking around. So, if you run into any issues with Bakta, we'd be happy to hear about it. Therefore, please, execute bakta in verbose mode (`-v`) and do not hesitate to file an issue including as much information as possible:

- a detailed description of the issue
- command line output
- log file (`<prefix>.log`)
- result file (`<prefix>.json`) *if possible*
- a reproducible example of the issue with an input file that you can share *if possible*

CHAPTER

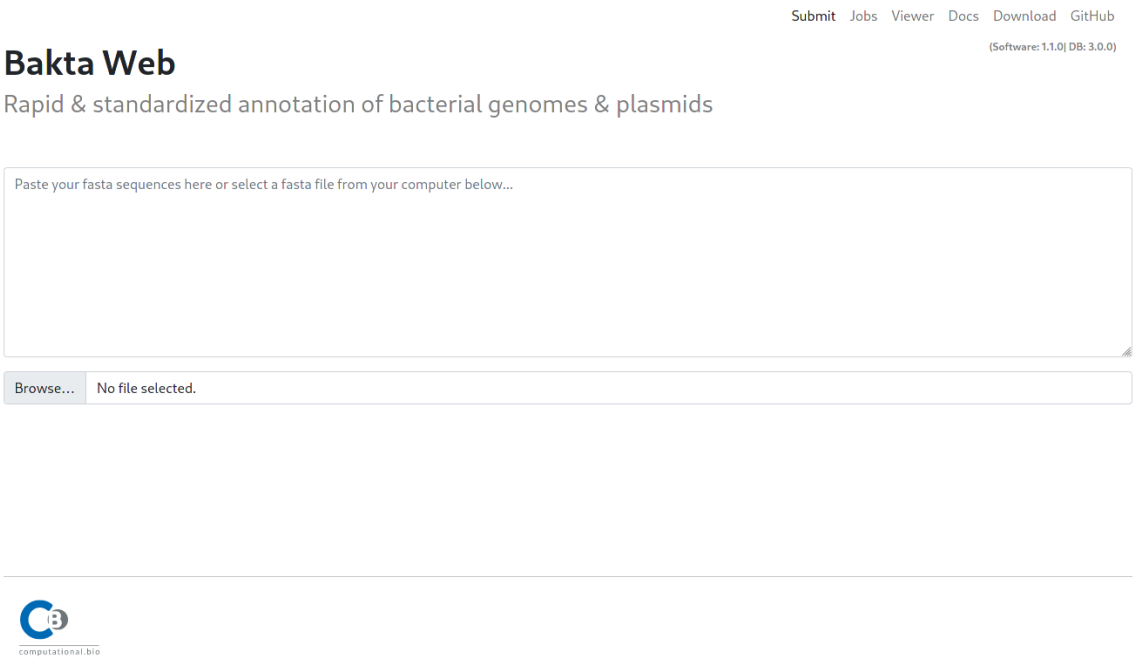
TWO

BAKTA WEB

We provide a dedicated Bakta web version available via https://bakta.computational.bio.

2.1 Overview

The Bakta mainpage contains multiple sections for the user input. A textfield to paste your fasta sequence as well as a file input to upload your sequence as a Fasta file.

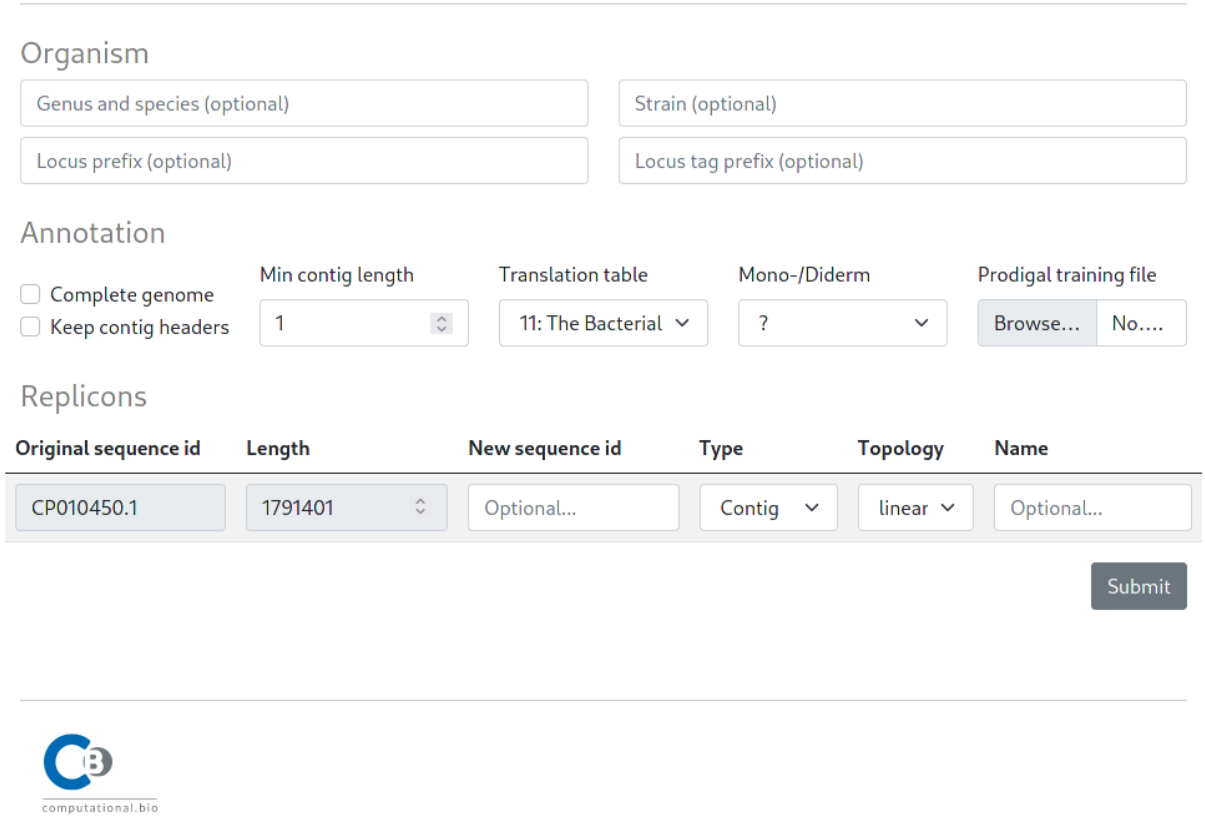


The menu contains multiple subsections:

- Submit: The mainpage for submitting new Jobs
- Jobs: Overview of all submitted Jobs and their status
- Viewer: Visualizing genome annotation results conducted with the local CLI version
- Docs: documentation
- Download: Download the Bakta tool and database
- GitHub: Github repository.

2.2 Submit options

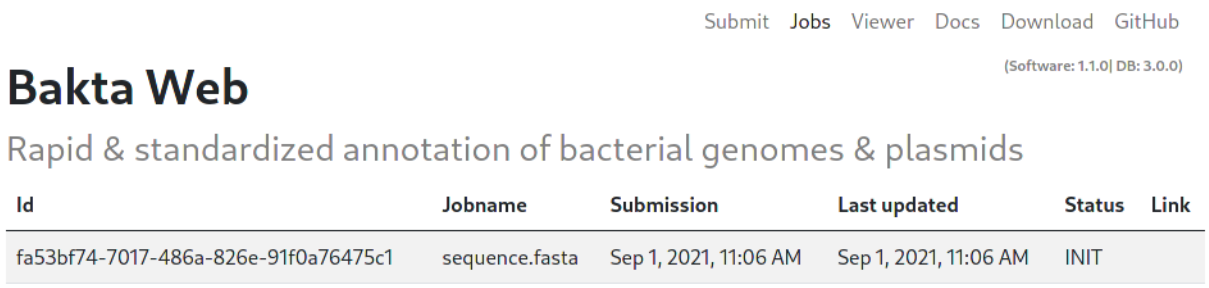
After choosing a sequence, users can set additional options before submitting.



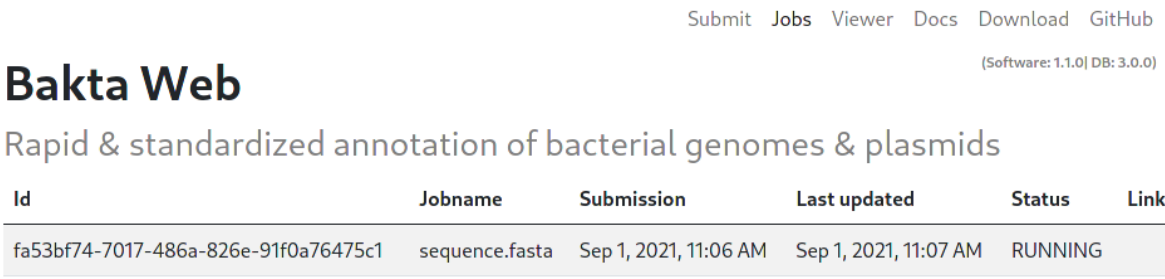
The submit options are split in three sections. An `Organism` section that allows users to specify additional (optional) description tags for the submitted fasta sequence, an `Annotation` section to specify the annotation settings and a `Replicons` section to provide optional sequence metadata, e.g. completeness and topology.

2.3 Monitoring Jobs

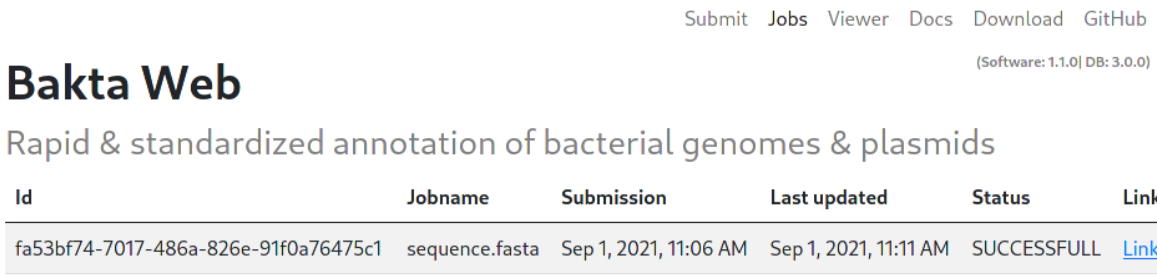
Submitted jobs are monitored automatically in the `Jobs` tab.



All jobs start with the `INIT` status. This indicates an initializing status, as well as a waiting position in the queue. A running Job is indicated by the `RUNNING` status.

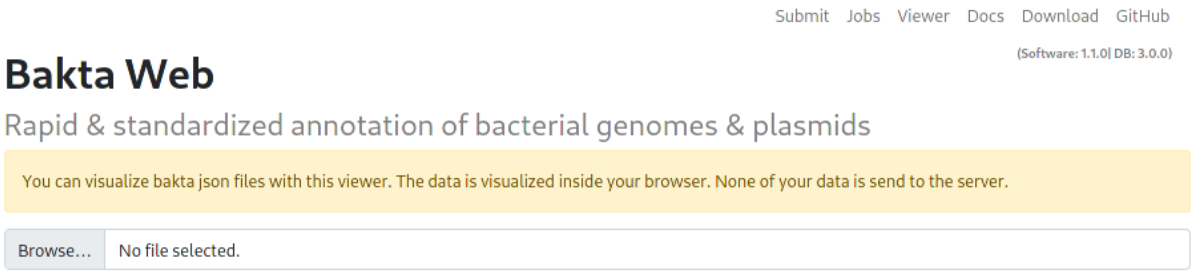


Finished jobs have the `SUCCESFULL` status and include a `LINK` to see the results in the `Viewer` tab.



2.4 Visualization

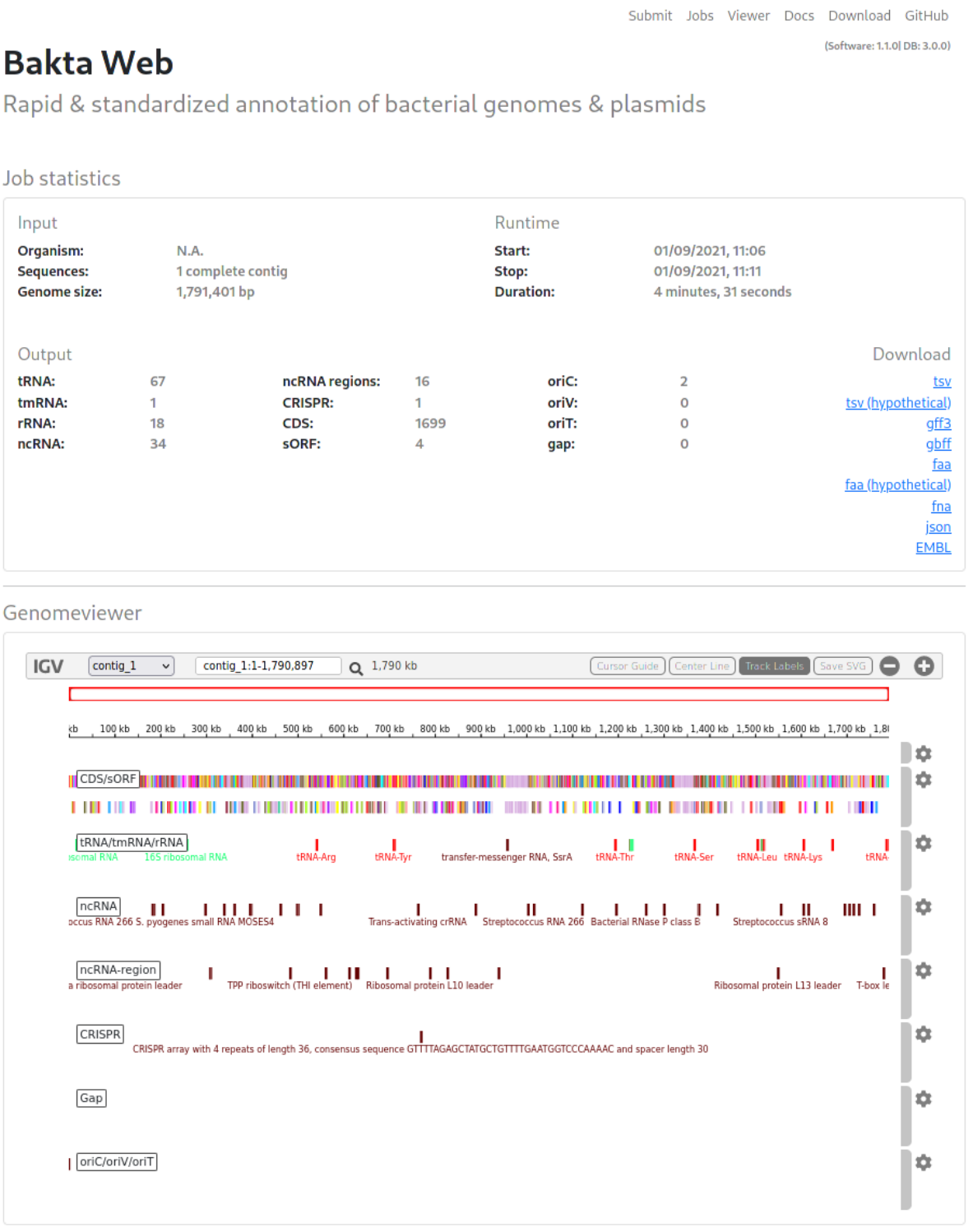
Results can be visualized via the `Viewer` tab, to visualize local files users can choose a local Json file for visualization. This visualization happens entirely and exclusively in the Browser, no data is uploaded to the server.



2.4.1 Results

Visualized results contain three sections:

- Job statistics: Contains a general overview of the annotated genome.
- Genomeviewer: An IGV-based genome browser to visualize annotated features.
- Annotations: A comprehensive list of all annotated features, including DB cross references to multiple common databases.


Submit Jobs Viewer Docs Download GitHub
(Software: 1.1.0| DB: 3.0.0)
Bakta Web
Rapid & standardized annotation of bacterial genomes & plasmids
Job statistics
Input
Organism: N.A.
Sequences: 1 complete contig
Genome size: 1,791,401 bp
Runtime
Start: 01/09/2021, 11:06
Stop: 01/09/2021, 11:11
Duration: 4 minutes, 31 seconds
Output
tRNA: 67
tmRNA: 1
rRNA: 18
ncRNA: 34
ncRNA regions: 16
CRISPR: 1
CDS: 1699
sORF: 4
oriC: 2
oriV: 0
oriT: 0
gap: 0
Download
tsv
tsv (hypothetical)
gff3
gbff
faa
faa (hypothetical)
fna
json
EMBL
Genomeviewer
IGV
contig_1
contig_1:1-1,790,897
1,790 kb
Cursor Guide
Center Line
Track Labels
Save SVG
100 kb 200 kb 300 kb 400 kb 500 kb 600 kb 700 kb 800 kb 900 kb 1,000 kb 1,100 kb 1,200 kb 1,300 kb 1,400 kb 1,500 kb 1,600 kb 1,700 kb
CDS/sORF
tRNA/tmRNA/rRNA
16S ribosomal RNA
tRNA-Arg
tRNA-Tyr
transfer-messenger RNA, SsrA
tRNA-Thr
tRNA-Ser
tRNA-Leu
tRNA-Lys
ncRNA
S. pyogenes small RNA MOSES4
Trans-activating crRNA
Streptococcus RNA 266
Bacterial RNase P class B
Streptococcus sRNA 8
ncRNA-region
TPP riboswitch (THI element)
Ribosomal protein L10 leader
Ribosomal protein L13 leader
CRISPR
CRISPR array with 4 repeats of length 36, consensus sequence GTTTTAGAGCTATGCTGTTTTGAATGGTCCCAAAAC and spacer length 30
Gap
oriC/oriV/oriT

Annotations

Show 100 entries Search:

Sequence	Type	Start	Stop	Strand	Locus tag	Product	DbXrefs
contig_1	oriC	4	235	?	NPDCAJ_00005		
contig_1	cds	236	1591	+	NPDCAJ_00010	Chromosomal replication initiator protein DnaA	COG:COG0593 COG:L GO:0003688 GO:0005524 GO:0005737 GO:0006270 GO:0006275 RefSeq:WP_002987659.1 SO:0001217 UniProtKB/TrEMBL:UPI0C UniProtKB/TrEMBL:UniRe UniProtKB/TrEMBL:UniRe UniProtKB/TrEMBL:UniRe
contig_1	oriC	1592	1745	?	NPDCAJ_00015		
contig_1	cds	1746	2882	+	NPDCAJ_00020	Beta sliding clamp	RefSeq:WP_011184038.1 SO:0001217 UniProtKB/TrEMBL:UPI0C UniProtKB/TrEMBL:UniRe UniProtKB/TrEMBL:UniRe
contig_1	cds	2957	3154	+	NPDCAJ_00025	Hypothetical cytosolic protein	COG:COG4481 COG:S RefSeq:WP_002981933.1 SO:0001217 UniProtKB/TrEMBL:UPI0C UniProtKB/TrEMBL:UniRe UniProtKB/TrEMBL:UniRe UniProtKB/TrEMBL:UniRe
contig_1	cds	3484	4599	+	NPDCAJ_00030	Ribosome-binding ATPase YchF	COG:COG0012 COG:J RefSeq:WP_002994729.1 SO:0001217 UniProtKB/TrEMBL:UPI0C UniProtKB/TrEMBL:UniRe UniProtKB/TrEMBL:UniRe UniProtKB/TrEMBL:UniRe
contig_1	cds	4669	5238	+	NPDCAJ_00035	Peptidyl-tRNA hydrolase	COG:COG0193 COG:J EC:3.1.1.29 GO:0004045 GO:0005737 GO:0006412

Showing 1 to 100 of 1,842 entries Previous 1 2 3 4 5 ... 19 Next

CHAPTER

THREE

BAKTA API-DOCUMENTATION

Bakta provides a open-access REST-API that can be used to annotate own genomes programmatically. The API and the corresponding Swagger documentation can be found here

3.1 Usage

Using the REST-API requires a three staged process. First, a job must be *initialized*, then the initialized job must be *started* and its status monitored, in the last step the *results* of a finished job can be retrieved. These stages must occur in this exact order, any deviation from this order will result in a failing job.

3.1.1 Initialization

Initialization of a Bakta Job must start with a *init request*. The API responds with a unique `jobID` and a corresponding `secret` as well as three pre-authenticated S3 urls (`uploadLinkFasta`, `uploadLinkProdigal`, `uploadLinkReplicons`). `jobID` and the corresponding `secret` must be stored locally and are used as credentials to identify the user in following requests.

The initialization procedure is finished by using the three S3-URLs (`uploadLinkFasta`, `uploadLinkProdigal`, `uploadLinkReplicons`) to upload data to the internal storage system. For this `PUT` requests with raw data as request body should be used.

`uploadLinkFasta` should be used to upload the (fasta) sequence data for annotation. `uploadLinkProdigal` (optional) can be used to upload an additional prodigal training file `uploadLinkReplicons` (optional) should be used to upload a replicon table in `tsv` format that describes the provided replicons in the fasta input file

Note: A `PUT` request to all three S3-URLs is necessary to finish the initialization procedure, optional URLs should be satisfied with a request that has an empty body with length zero.

3.1.2 Job start & Monitoring

After initialization the specific job (identified by `jobID` and a corresponding `secret`) can be scheduled via the *start request*.

Scheduled jobs are monitored via the *job list request*. This requests contains a list of all monitored `jobIDs` and `secrets`. The API responds with a matching JSON list of `jobStatuses` . The *job list request* should be repeated until all jobs have either the `SUCCESSFUL` or `ERROR` status. Recently scheduled jobs have the `INIT` status, currently running jobs `RUNNING`.

Note: The `INIT` status refers to a started job that is currently initialized for excecution. Depending on the current load of the underlying hardware and position in the scheduling queue, it may take a while before a job transitions to the `RUNNING` status. A failed job is always indicated by the `ERROR` status.

If multiple jobs are monitored simultaneosly the finalization procedure can be started for a job with `SUCCESSFUL` status while others are still `RUNNING`. In this case the monitoring should continue in parallel for the remaining jobs and the finished job can be removed from the list.

3.1.3 Getting results

Results for jobs with a `SUCCESFUL` status can be retrieved via the *result request*. The response contains a list (ResultFiles) of different file-formats with corresponding Download URLs. The result files can be retrieved with `GET` requests to the URL, or via a regular Webbrowser.

Currently the following file formats for results are provided:

- **EMBL**
- **FAA**
- **FAAHypothetical**
- **FNA**
- **GBFF**
- **GFF3**
- **JSON**
- **TSV**
- **TSVHypothetical**

More information about the structure of these output formats can be found in the CLI Documentation

Note: The `JSON` output format can be visualized locally via the WebUI at https://bakta.computational.bio.

3.2 Endpoints

3.2.1 /api/v1/job/init

The init endpoint is used to initialize a new job. Initialized jobs can be started via the *start request*.

HTTP-Method: **POST**

Expected request body:

```
{
  "repliconTableType": "CSV",
  "name": "string"
}
```

`repliconTableType` describes the file format of the provided replicontable, this should be either `CSV` or `TSV`. `name` is an arbitrary name, usually the name of the fasta input file.

Expected response body:

```
{
  "uploadLinkFasta": "string",
  "uploadLinkProdigal": "string",
  "uploadLinkReplicons": "string",
  "job": {
    "secret": "string",
    "jobID": "string"
  }
}
```

The response contains three S3-URLs (`uploadLinkFasta`, `uploadLinkProdigal`, `uploadLinkReplicons`). These URLs are pre-authenticated and can be used to upload data to the internal storage using **PUT** requests. For a detailed, step-by-step guide to use these URLs see *Usage*. Additionally the init-request-response contains a job description with an unique `jobID` and a corresponding `secret` that are used by future request to identify and authorize the initialized job.

3.2.2 /api/v1/job/start

This endpoint is used to start a job that has been initialized via the *init request*.

HTTP-Method: **POST**

Expected request body:

```
{
  "job": {
    "secret": "string",
    "jobID": "string"
  },
  "config": {
    "hasProdigal": true,
    "hasReplicons": true,
    "translationalTable": 0,
    "completeGenome": true,
    "keepContigHeaders": true,
    "minContigLength": "string",
    "dermType": "UNKNOWN",
    "genus": "string",
    "species": "string",
    "strain": "string",
    "plasmid": "string",
    "locus": "string",
    "locusTag": "string"
  },
}
```

A successful response is indicated by a *200* status code and an empty response body.

3.2.3 /api/v1/job/list

Endpoint to query the current status of one (or more) running jobs.

HTTP-Method: **POST**

Expected request body:

```
{
  "jobs": [
    {
      "secret": "string",
      "jobID": "string"
    }
  ]
}
```

Response:

```
{
  "jobs": [
    {
      "jobID": "string",
      "jobStatus": "INIT",
      "started": "2021-07-02T11:41:10.675Z",
      "updated": "2021-07-02T11:41:10.675Z",
      "name": "string"
    }
  ],
  "failedJobs": [
    {
      "jobID": "string",
      "jobStatus": "NOT_FOUND"
    }
  ]
}
```

3.2.4 /api/v1/job/result

Endpoint to query the results of a finished job.

HTTP-Method: **POST**

Request:

```
{
  "secret": "string",
  "jobID": "string"
}
```

Response:

```
{
  "jobID": "string",
  "ResultFiles":
```

(continues on next page)

(continued from previous page)


```
    {
      "EMBL": "S3-URL",
      "FAA": "S3-URL",
      "FAAHypothetical": "S3-URL",
      "FNA": "S3-URL",
      "GBFF": "S3-URL",
      "GFF3": "S3-URL",
      "JSON": "S3-URL",
      "TSV": "S3-URL",
      "TSVHypothetical": "S3-URL"
    },
  "started": "2021-07-14T11:10:31.838Z",
  "updated": "2021-07-14T11:10:31.838Z",
  "name": "string"
}
```


3.2.5 /api/v1/version

Method that can be used to determine the internal database and Bakta version.

HTTP-METHOD: **GET**

Response:


```
{
  "toolVersion": "string",
  "dbVersion": "string",
  "backendVersion": "string"
}
```

CHAPTER

FOUR

FEATURES

- **Bacteria & plasmids only** Bakta was designed to annotate bacteria and plasmids, only. This decision by design has been made in order to tweak the annotation process regarding tools, preferences & databases and to streamline further development & maintenance of the software.
- **FAIR annotations** To provide standardized annotations adhearing to FAIR principles, Bakta utilizes a comprehensive & versioned custom annotation database based on UniProt's UniRef100 & UniRef90 protein clusters (`FAIR` -> DOI/DOI) enriched with dbxrefs (`GO`, `COG`, `EC`) and annotated by specialized niche databases. For each db version we provide a comprehensive log file of all imported sequences and annotations.
- **Protein sequence identification** Fostering the FAIR aspect, Bakta identifies identical protein sequences (**IPS**) via `MD5` hash digests which are annotated with database cross-references (**dbxref**) to RefSeq (`WP_*`), UniRef100 (`UniRef100_*`) and UniParc (`UPI*`). By doing so, IPS allow the surveillance of distinct gene alleles and streamlining comparative analysis as well as posterior (external) annotations of `putative` & `hypothetical` protein sequences which can be mapped back to existing CDS via these exact & stable identifiers (*E. coli* gene ymiA ...more). Currently, Bakta identifies ~198 mio, ~185 mio and ~150 mio distinct protein sequences from UniParc, UniRef100 and RefSeq, respectively. Hence, for certain genomes, up to 99 % of all CDS can be identified this way, skipping computationally expensive sequence alignments.
- **Small proteins / short open reading frames** Bakta detects and annotates small proteins/short open reading frames (**sORF**) which are not predicted by tools like `Prodigal`.
- **Fast** Bakta can annotate a typical bacterial genome in 10 $\pm$5 min on a laptop, plasmids in a couple of seconds/minutes.
- **Expert annotation systems** To provide high quality annotations for certain proteins of higher interest, *e.g.* AMR & VF genes, Bakta includes & merges different expert annotation systems. Currently, Bakta uses NCBI's AMRFinderPlus for AMR gene annotations as well as a generalized protein sequence expert system with distinct coverage, identity and priority values for each sequence, currenlty comprising the VFDB as well as NCBI's BlastRules.
- **Comprehensive workflow** Bakta annotates ncRNA cis-regulatory regions, oriC/oriV/oriT and assembly gaps as well as standard feature types: tRNA, tmRNA, rRNA, ncRNA genes, CRISPR, CDS.
- **GFF3 & INSDC conform annotations** Bakta writes GFF3 and INSDC-compliant (Genbank & EMBL) annotation files ready for submission (checked via GenomeTools GFF3Validator and ENA Webin-CLI for GFF3 and EMBL file formats, respectively for representative genomes of all ESKAPE species).
- **Reasoning** By annotating bacterial genomes in a standardized, taxon-independent, high-throughput and local manner, Bakta aims at a well-balanced tradeoff between fully-featured but computationally demanding pipelines like PGAP and rapid highly-customizable offline tools like Prokka. Indeed, Bakta is heavily inspired by Prokka (kudos to Torsten Seemann) and many command line options are compatible for the sake of interoperability and user convenience. Hence, if Bakta does not fit your needs, please try Prokka.

CHAPTER

FIVE

CITATION

> **Schwengers O., Jelonek L., Dieckmann M. A., Beyvers S., Blom J., Goesmann A. (2021). Bakta:** rapid and standardized annotation of bacterial genomes via alignment-free sequence identification. Microbial Genomics, 7(11). https://doi.org/10.1099/mgen.0.000685

Bakta is standing on the shoulder of giants taking advantage of many publicly available databases. If you find any of those used within Bakta useful, please credit these primary sources, as well:

- UniProt: https://doi.org/10.1093/nar/gky1049
- RefSeq: https://doi.org/10.1093/nar/gkx1068
- Rfam: https://doi.org/10.1002/cpbi.51
- AMRFinder: https://doi.org/10.1128/AAC.00483-19
- ISFinder: https://doi.org/10.1093/nar/gkj014
- AntiFam: https://doi.org/10.1093/database/bas003
- Mob-suite: https://doi.org/10.1099/mgen.0.000206
- DoriC: https://doi.org/10.1093/nar/gky1014
- COG: https://doi.org/10.1093/bib/bbx117
- VFDB: https://doi.org/10.1093/nar/gky1080