

---

**Bakta**

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

**Sep 03, 2021**



# CONTENTS

<b>1</b>	<b>Bakta CLI</b>	<b>3</b>
1.1	Contents . . . . .	3
1.2	Installation . . . . .	3
1.3	Examples . . . . .	5
1.4	Input and Output . . . . .	6
1.5	Usage . . . . .	8
1.6	Annotation Workflow . . . . .	9
1.7	Database . . . . .	11
1.8	Genome Submission . . . . .	12
1.9	FAQ . . . . .	13
1.10	Issues and Feature Requests . . . . .	14
<b>2</b>	<b>Bakta Web</b>	<b>15</b>
2.1	Overview . . . . .	15
2.2	Submit options . . . . .	16
2.3	Monitoring Jobs . . . . .	16
2.4	Visualization . . . . .	17
<b>3</b>	<b>Bakta API-Documentation</b>	<b>21</b>
3.1	Usage . . . . .	21
3.2	Endpoints . . . . .	22
<b>4</b>	<b>Features</b>	<b>27</b>
<b>5</b>	<b>Citation</b>	<b>29</b>



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.



## 1.1 Contents

- *Installation*
- *Examples*
- *Input & Output*
- *Usage*
- *Annotation Workflow*
- *Database*
- *Genome Submission*
- *FAQ*
- *Issues & Feature Requests*

## 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
```

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

- tRNAscan-SE (2.0.6) <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 (1.1.2) <https://dx.doi.org/10.1093%2Fbioinformatics%2Fbtt509> <http://eddylab.org/inferral>
- 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 (3.3.1) <https://doi.org/10.1093/nar/gkt263> <http://hmmer.org>
- Diamond (2.0.11) <https://doi.org/10.1038/nmeth.3176> <https://github.com/bbuchfink/diamond>
- Blast+ (2.7.1) <https://www.ncbi.nlm.nih.gov/pubmed/2231712> <https://blast.ncbi.nlm.nih.gov>
- AMRFinderPlus (3.10.1) <https://github.com/ncbi/amr>

On Ubuntu/Debian/Mint you can install these via:

```
$ sudo apt install trnascan-se aragorn inferral pilercr prodigal hmmer diamond-aligner_
↳ncbi-blast+
```

Tested with Ubuntu 20.04 - some older distributions might provide outdated versions, *e.g.* trnascan-se in Ubuntu 18.04. In these cases dependencies must be installed manually.



## 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/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	``	-	-	

## 1.4.2 Output

Annotation results are provided in standard bioinformatics file formats:

- `<prefix>.tsv`: annotations as simple human readable tab separated values
- `<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 readable tab separated values
- `<prefix>.hypotheticals.faa`: hypothetical protein CDS amino acid sequences as FASTA

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,
```

(continues on next page)

```

        "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]
        [--locus-tag LOCUS_TAG] [--keep-contig-headers] [--replicons REPLICONS] [--
↳compliant] [--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 & 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

```

(continues on next page)

(continued from previous page)

```

Path to existing Prodigal training file to use for CDS prediction
--translation-table {11,4}
Translation table: 11/4 (default = 11)
--gram {+,-,?}
Gram type: +/-/? (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

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
--skip-crispr
Skip CRISPR array detection & annotation
--skip-cds
Skip CDS 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 programs 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 un-completed contigs)
2. discard spurious CDS via AntiFam
3. Detection of UPSs via MD5 digests and lookup of related IPS and PCS
4. Sequence alignments of remainder via Diamond vs. PSC (query/subject coverage=0.8, identity=0.5)
5. Assign protein sequences to UniRef90 or UniRef50 clusters if alignment hits meet an identity larger than 0.9 or 0.5, respectively
6. Execution of expert systems:
  - AMR: AMRFinderPlus
  - Alignments: NCBI BlastRules, VFDB
1. Combination of available 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

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 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 (214,847,897)
- **IPS:** UniProt UniRef100 (199,630,327)
- **PSC:** UniProt UniRef90 (90,580,050)
- **PSCC:** UniProt UniRef50 (12,127,845)

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: 150,463,165)
- NCBI COG db (PSC: 3,353,131)
- SwissProt EC/GO terms (PSC: 335,256)
- NCBI AMRFinderPlus (IPS: 6,255, PSC: 43,413)
- ISFinder db (IPS: 27,841, PSC: 10,161)

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 currently 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: 267

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 3.0, 28 Gb zipped, 53 Gb unzipped) are hosted at Zenodo:

## 1.8 Genome Submission

Most genomes annotated with Bakta should be ready-to-submit 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 --compliant --db <db-path> -v --genus Escherichia --species "coli 0157:H7" --
↳ strain Sakai --complete --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
```

(continues on next page)



(continued from previous page)

```
# 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
```

Exemplary 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 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>`
- **Nice, but I'm missing 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.

## 1.10 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*

## 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 screenshot shows the top navigation menu with links: Submit, Jobs, Viewer, Docs, Download, GitHub. Below the menu is the text "(Software: 1.1.0| DB: 3.0.0)". The main heading is "Bakta Web" with the subtitle "Rapid & standardized annotation of bacterial genomes & plasmids". The central part of the interface features a large text area with the placeholder text "Paste your fasta sequences here or select a fasta file from your computer below...". Below this text area is a file upload control with a "Browse..." button and the text "No file selected."



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.

---

**Organism**

Genus and species (optional)

Strain (optional)

Locus prefix (optional)

Locus tag prefix (optional)

**Annotation**

Complete genome   
  Keep contig headers   
 Min contig length    
 Translation table    
 Mono-/Diderm    
 Prodigal training file

**Replicons**

Original sequence id	Length	New sequence id	Type	Topology	Name
CP010450.1	1791401	Optional...	Contig	linear	Optional...

---



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.

[Submit](#) [Jobs](#) [Viewer](#) [Docs](#) [Download](#) [GitHub](#)

(Software: 1.1.0 | DB: 3.0.0)

### Bakta Web

Rapid & standardized annotation of bacterial genomes & plasmids

Id	Jobname	Submission	Last updated	Status	Link
fa53bf74-7017-486a-826e-91f0a76475c1	sequence.fasta	Sep 1, 2021, 11:06 AM	Sep 1, 2021, 11:06 AM	INIT	

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.

Submit Jobs Viewer Docs Download GitHub

(Software: 1.1.0| DB: 3.0.0)

## Bakta Web

Rapid & standardized annotation of bacterial genomes & plasmids

Id	Jobname	Submission	Last updated	Status	Link
fa53bf74-7017-486a-826e-91f0a76475c1	sequence.fasta	Sep 1, 2021, 11:06 AM	Sep 1, 2021, 11:07 AM	RUNNING	

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

Submit Jobs Viewer Docs Download GitHub

(Software: 1.1.0| DB: 3.0.0)

## Bakta Web

Rapid & standardized annotation of bacterial genomes & plasmids

Id	Jobname	Submission	Last updated	Status	Link
fa53bf74-7017-486a-826e-91f0a76475c1	sequence.fasta	Sep 1, 2021, 11:06 AM	Sep 1, 2021, 11:11 AM	SUCCESSFUL	<a href="#">Link</a>

## 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.

Submit Jobs Viewer Docs Download GitHub

(Software: 1.1.0| DB: 3.0.0)

## Bakta Web

Rapid & standardized annotation of bacterial genomes & plasmids

You can visualize bakta json files with this viewer. The data is visualized inside your browser. None of your data is send to the server.

Browse... No file selected.

### 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.

# Bakta Web

Rapid & standardized annotation of bacterial genomes & plasmids

## Job statistics

Input		Runtime		
<b>Organism:</b>	N.A.	<b>Start:</b>	01/09/2021, 11:06	
<b>Sequences:</b>	1 complete contig	<b>Stop:</b>	01/09/2021, 11:11	
<b>Genome size:</b>	1,791,401 bp	<b>Duration:</b>	4 minutes, 31 seconds	
Output				Download
<b>tRNA:</b>	67	<b>ncRNA regions:</b>	16	<a href="#">tsv</a>
<b>tmRNA:</b>	1	<b>CRISPR:</b>	1	<a href="#">tsv (hypothetical)</a>
<b>rRNA:</b>	18	<b>CDS:</b>	1699	<a href="#">gff3</a>
<b>ncRNA:</b>	34	<b>SORF:</b>	4	<a href="#">gbff</a>
				<a href="#">faa</a>
				<a href="#">faa (hypothetical)</a>
				<a href="#">fna</a>
				<a href="#">json</a>
				<a href="#">EMBL</a>

## Genomeviewer



## Annotations

Show  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	<a href="#">COG:COG0593</a> <a href="#">COG:L</a> <a href="#">GO:0003688</a> <a href="#">GO:0005524</a> <a href="#">GO:0005737</a> <a href="#">GO:0006270</a> <a href="#">GO:0006275</a> <a href="#">RefSeq:WP_002987659.1</a> <a href="#">SO:0001217</a> <a href="#">UniProtKB/TrEMBL:UPI00</a> <a href="#">UniProtKB/TrEMBL:UniRe</a> <a href="#">UniProtKB/TrEMBL:UniRe</a> <a href="#">UniProtKB/TrEMBL:UniRe</a>
contig_1	oriC	1592	1745	?	NPDCAJ_00015		
contig_1	cds	1746	2882	+	NPDCAJ_00020	Beta sliding clamp	<a href="#">RefSeq:WP_011184038.1</a> <a href="#">SO:0001217</a> <a href="#">UniProtKB/TrEMBL:UPI00</a> <a href="#">UniProtKB/TrEMBL:UniRe</a> <a href="#">UniProtKB/TrEMBL:UniRe</a>
contig_1	cds	2957	3154	+	NPDCAJ_00025	Hypothetical cytosolic protein	<a href="#">COG:COG4481</a> <a href="#">COG:S</a> <a href="#">RefSeq:WP_002981933.1</a> <a href="#">SO:0001217</a> <a href="#">UniProtKB/TrEMBL:UPI00</a> <a href="#">UniProtKB/TrEMBL:UniRe</a> <a href="#">UniProtKB/TrEMBL:UniRe</a> <a href="#">UniProtKB/TrEMBL:UniRe</a>
contig_1	cds	3484	4599	+	NPDCAJ_00030	Ribosome-binding ATPase YchF	<a href="#">COG:COG0012</a> <a href="#">COG:J</a> <a href="#">RefSeq:WP_002994729.1</a> <a href="#">SO:0001217</a> <a href="#">UniProtKB/TrEMBL:UPI00</a> <a href="#">UniProtKB/TrEMBL:UniRe</a> <a href="#">UniProtKB/TrEMBL:UniRe</a> <a href="#">UniProtKB/TrEMBL:UniRe</a>
contig_1	cds	4669	5238	+	NPDCAJ_00035	Peptidyl-tRNA hydrolase	<a href="#">COG:COG0193</a> <a href="#">COG:J</a> <a href="#">EC:3.1.1.29</a> <a href="#">GO:0004045</a> <a href="#">GO:0005737</a> <a href="#">GO:0006412</a>

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





## 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 execution. 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 simultaneously 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 SUCCESSFUL status can be retrieved via the *result request*. The response contains a list (Result-Files) 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 a 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"
}
```



## 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 adhering 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 ±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, currently 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.





## CITATION

A manuscript is in preparation. To temporarily cite our work, please transitionally refer to:

---

### Citation

**Schwengers O., Jelonek L., Dieckmann M., Beyvers S., Blom J., Goesmann A. (2021). Bakta: Rapid & standardized annotation of bacterial genomes via alignment-free sequence identification. bioRxiv 2021.09.02.458689; doi: <https://doi.org/10.1101/2021.09.02.458689>**

---

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>