

---

# **Snakeface Documentation**

***Release 0.0.18***

**Johannes Koester**

**Dec 30, 2021**



# GETTING STARTED

<b>1</b>	<b>Getting started with Snakemake Interface</b>	<b>3</b>
<b>2</b>	<b>Support</b>	<b>5</b>
<b>3</b>	<b>Resources</b>	<b>7</b>
3.1	Getting Started . . . . .	7
3.2	Use Cases . . . . .	15
3.3	The Snakeface API . . . . .	15
3.4	Internal API . . . . .	15
	<b>Python Module Index</b>	<b>29</b>
	<b>Index</b>	<b>31</b>



Snakeface is the Snakemake Interface, where you can easily run workflows. To learn more about Snakemake, visit the [official documentation](#)



## **GETTING STARTED WITH SNAKEMAKE INTERFACE**

Snakeface can be used on your local machine to provide a nice interface to running snakemake workflows, or deployed by a group to run shared workflows. See [Use Cases](#) for an overview of different use cases.





## SUPPORT

- In case of **questions**, please post on [stack overflow](#).
- To **discuss** with other Snakemake users, you can use the [mailing list](#). **Please do not post questions there. Use stack overflow for questions.**
- For **bugs and feature requests**, please use the [issue tracker](#).
- For **contributions**, visit Snakemake on [Github](#).



## RESOURCES

**Snakemake Repository** The Snakemake workflow manager repository houses the core software for Snakemake.

**Snakemake Wrappers Repository** The Snakemake Wrapper Repository is a collection of reusable wrappers that allow to quickly use popular tools from Snakemake rules and workflows.

**Snakemake Workflows Project** This project provides a collection of high quality modularized and re-usable workflows. The provided code should also serve as a best-practices of how to build production ready workflows with Snakemake. Everybody is invited to contribute.

**Snakemake Profiles Project** This project provides Snakemake configuration profiles for various execution environments. Please consider contributing your own if it is still missing.

**Bioconda** Bioconda can be used from Snakemake for creating completely reproducible workflows by defining the used software versions and providing binaries.

### 3.1 Getting Started

Snakeface stands for “Snakemake Interface,” and it’s exactly that - an interface for you to easily run and interact with Snakemake workflows. Although it is still in development, the overarching goal is to be flexible to different needs for your deployment. This means that you can both run it quickly as a notebook to test a workflow locally, or deploy it in a cluster environment for your user base. If you have a need for deployment that is not addressed here, please [let us know](#) We recommend that you start by setting up the *Example Workflow*.

#### 3.1.1 Installation

Snakeface can be installed and run from a virtual environment, or from a container.

##### Virtual Environment

First, clone the repository code.

```
$ git clone git@github.com:snakemake/snakeface.git
$ cd snakeface
```

Then you’ll want to create a new virtual environment, and install dependencies.

```
$ python -m venv env
$ source env/bin/activate
$ pip install -r requirements.txt
```

And install Snakeface (from the repository directly)

```
$ pip install -e .
```

## Install via pip

Snakeface can also be installed with pip.

```
$ pip install snakeface
```

Once it's installed, you should be able to inspect the client!

```
$ snakeface --help
usage: snakeface [-h] [--version] [--noreload] [--verbosity {0,1,2,3}]
                [--workdir [WORKDIR]] [--auth {token}] [--port PORT]
                [--verbose] [--log-disable-color] [--log-use-threads]
                [--force]
                [repo] [dest] {notebook} ...

Snakeface: interface to snakemake.

positional arguments:
  repo                  Repository address and destination to deploy, e.g.,
                        <source> <dest>
  dest                  Path to clone the repository, should not exist.

optional arguments:
  -h, --help            show this help message and exit
  --version             print the version and exit.
  --noreload            Tells Django to NOT use the auto-reloader.
  --verbosity {0,1,2,3} Verbosity (0, 1, 2, 3).
  --workdir [WORKDIR]  Specify the working directory.
  --force               If the folder exists, force overwrite, meaning remove
                        and replace.

SETTINGS:
  --auth {token}        Authentication type to create for the interface,
                        defaults to token.

NETWORKING:
  --port PORT           Port to serve application on.

LOGGING:
  --verbose             verbose output for logging.
  --log-disable-color   Disable color for snakeface logging.
  --log-use-threads     Force threads rather than processes.

actions:
  subparsers for Snakeface

  {notebook}           snakeface actions
  notebook             run a snakeface notebook
```

## Setup

As a user, you most likely want to use Snakeface as an on demand notebook, so no additional setup is needed other than installing the package. As we add more deployment types that warrant additional configuration, or in the case of installing Snakeface as a cluster admin, you likely will want to install from the source repository (or a release) and edit the settings.yml file in the snakemake folder before deploying your service. More information will be added as this is developed. If you are interested, you can look at [Settings](#).

### 3.1.2 Example Workflow

#### Downloading Tutorial

You likely want to start with an example workflow. We will use the same one from the *snakemake tutorial* <<https://snakemake.readthedocs.io/en/stable/tutorial/tutorial.html>>\_. We assume that you have already installed snakeface (and thus Snakemake and it's dependencies are on your system). So you can download the example as follows:

```
$ mkdir snakemake-tutorial
$ cd snakemake-tutorial
$ wget https://github.com/snakemake/snakemake-tutorial-data/archive/v5.24.1.tar.gz
$ tar --wildcards -xf v5.24.1.tar.gz --strip 1 "*/data" "*/environment.yaml"
```

This should extract a data folder and an environment.yaml. You should also create the *Snakefile* <<https://snakemake.readthedocs.io/en/stable/tutorial/basics.html#summary>>\_. This Snakefile is the same as in the tutorial, with the addition of adding the environment.yaml to each section.

```
SAMPLES = ["A", "B"]

rule all:
    input:
        "calls/all.vcf"

rule bwa_map:
    input:
        "data/genome.fa",
        "data/samples/{sample}.fastq"
    output:
        "mapped_reads/{sample}.bam"
    conda:
        "environment.yaml"
    shell:
        "bwa mem {input} | samtools view -Sb - > {output}"

rule samtools_sort:
    input:
        "mapped_reads/{sample}.bam"
    output:
        "sorted_reads/{sample}.bam"
    conda:
        "environment.yaml"
    shell:
        "samtools sort -T sorted_reads/{wildcards.sample} "
        "-O bam {input} > {output}"
```

(continues on next page)

(continued from previous page)

```

rule samtools_index:
    input:
        "sorted_reads/{sample}.bam"
    output:
        "sorted_reads/{sample}.bam.bai"
    conda:
        "environment.yaml"
    shell:
        "samtools index {input}"

rule bcftools_call:
    input:
        fa="data/genome.fa",
        bam=expand("sorted_reads/{sample}.bam", sample=SAMPLES),
        bai=expand("sorted_reads/{sample}.bam.bai", sample=SAMPLES)
    output:
        "calls/all.vcf"
    conda:
        "environment.yaml"
    shell:
        "samtools mpileup -g -f {input.fa} {input.bam} | "
        "bcftools call -mv - > {output}"

```

## Running Snakeface

At this point, from this working directory you can run Snakeface. For example, you might run a *Notebook*. Make sure to select `--use-conda` or else the environment above won't be properly sourced. This is one deviation from the main Snakemake tutorial, which has you install dependencies on the command line before running the workflow, and the workflow doesn't have the `conda` sections.

### 3.1.3 Notebook

Make sure that before you run a notebook, you are comfortable with Snakemake and have a workflow with a Snakefile read to run. If not, you can start with the instructions for an example workflow (*Example Workflow*).

#### Local Notebook

If you have installed Snakeface on your own, you likely want a notebook. You can run `snakeface` without any arguments to run one by default: This works because the default install settings have set `NOTEBOOK_ONLY` and it will start a Snakeface session.

```
$ snakeface
```

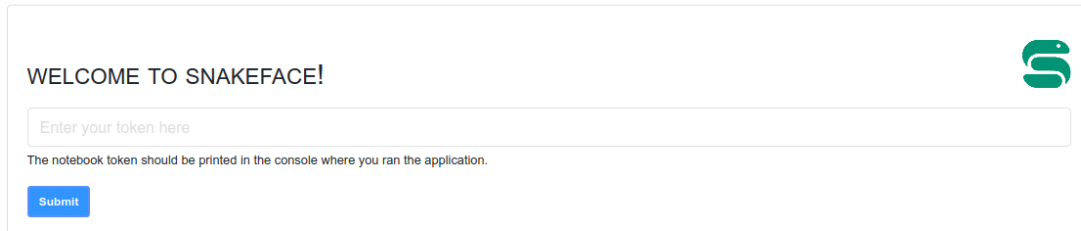
However, if your center is running Snakeface as a service, you will need to ask for a notebook explicitly:

```
$ snakeface notebook
```

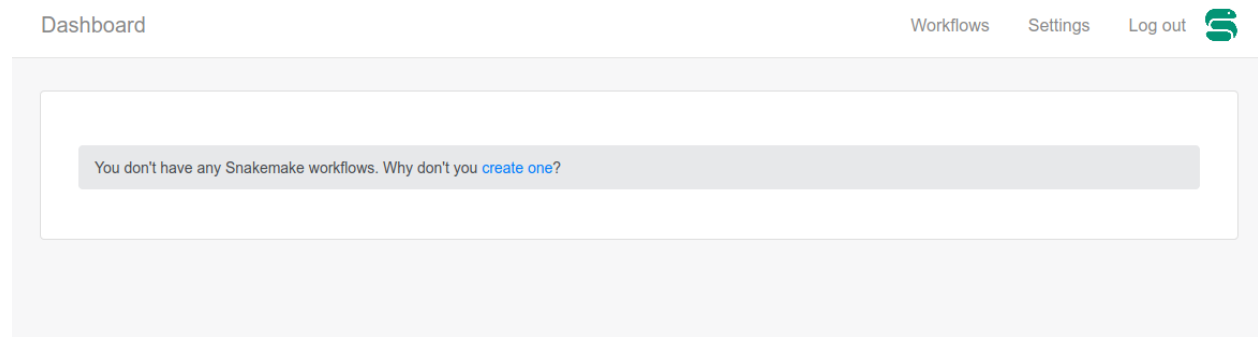
For either of the two you can optionally specify a port:

```
$ snakeface notebook --port 5555
$ snakeface --port 5555
```

For the notebook install, you will be given a token in your console, and you can copy paste it into the interface to log in.


The image shows a web interface for logging into Snakeface. At the top left, it says "WELCOME TO SNAKEFACE!". On the top right is a green logo consisting of a stylized 'S'. Below the welcome message is a text input field with the placeholder text "Enter your token here". Underneath the input field is a small line of text: "The notebook token should be printed in the console where you ran the application." At the bottom left of the form is a blue button with the text "Submit".

You can then browse to localhost at the port specified to see the interface! The first prompt will ask you to create a collection, which is a grouping of workflows. You might find it useful to organize your projects.

The image shows the Snakeface dashboard. At the top, there is a navigation bar with the word "Dashboard" on the left and "Workflows", "Settings", and "Log out" on the right, followed by a green logo. Below the navigation bar is a large light gray box. Inside this box, at the top, is a message: "You don't have any Snakemake workflows. Why don't you [create one](#)?" where "create one" is a blue link.

Next, click on the button to create a new workflow. The next form will provide input fields for all arguments provided by your Snakemake installation. You can select the blue buttons at the top (they are always at the top) to jump to a section, and see the command being previewed at the bottom. The command will always update when you make a new selection.

Create Workflow

Workflows Settings Log out 

MAIN EXECUTION GROUPING REPORTS UTILITIES OUTPUT BEHAVIOR CONDA SINGULARITY ENVIRONMENT MODULES

Run Workflow

### Main

NAME  
Dinosaur Workflow

WORKDIRS\*  
/

Your working directory must be within the path where you launched your notebook.

### EXECUTION

TARGET

Targets to build. May be rules or files.

☒ Dryrun


Do not execute anything, and display what would be done. If you have a very large workflow, use `--dry-run --quiet` to just print a summary of the DAG of jobs.

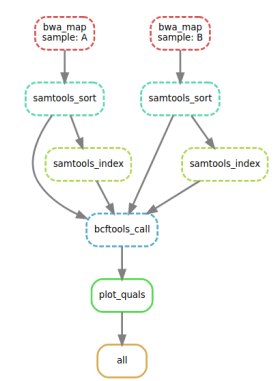
PROFILE

```
snakemake --snakefile /home/vanessa/Desktop/Code/snakeface/snakefile --local-cores 8 --scheduler greedy --wms-monitor http://127.0.0.1:5000 --scheduler-llp-solver COIN_CMO --max-inventory-time 20 --latency-wait 5 --max-jobs-per-second 10 --max-status-checks-per-second 10 --attempts 1 --wrapper-prefix https://github.com/snakemake/snakemake-wrappers/raw --conda-frontend conda
```

Note that if you start running your notebook in a location without any Snakefiles, you will get a message that tells you to create one first. A Snakefile matching some pattern of `snakefile*` (case ignored) must be present. When you've finished your workflow, click on "Run Workflow." If the workflow is invalid (e.g., you've moved the Snakefile, or provided conflicting commands) then you'll be returned to this view with an error message. If it's valid, you'll be redirected to a page to monitor the workflow.

Workflow: 9

Workflows Settings Log out 



```

graph TD
    A[bwa_map sample: A] --> AS[samtools_sort]
    B[bwa_map sample: B] --> BS[samtools_sort]
    AS --> AI[samtools_index]
    BS --> BI[samtools_index]
    AI --> BC[bcfutils_call]
    BI --> BC
    BC --> PQ[plot_qual]
    PQ --> all[all]
  
```

RE-RUN EDIT CANCEL DELETE

Snakefile `/home/vanessa/Desktop/Code/snakemake/snakemake-tutorial/Snakemakefile`

Workdir `/home/vanessa/Desktop/Code/snakemake/snakemake-tutorial`

Command

```
snakemake --snakefile /home/vanessa/Desktop/Code/snakemake/snakemake-tutorial/Snakemakefile --cores 1 --local-cores 8 --wms-monitor http://127.0.0.1:5000 --max-inventory-time 20 --latency-wait 5 --max-jobs-per-second 10 --max-status-checks-per-second 10 --attempt 1 --use-conda --wms-monitor-arg id=9
```

Return Code 1

WMS\_MONITOR\_TOKEN `d09ab3ba0bb073ca02faf42f831a003e154860dd`

Command Line Interaction

To interact with this workflow from the command line, export this variable and provide the following extra arguments

```
--wms-monitor http://127.0.0.1:5000
--wms-monitor-arg id=9
```

Building DAG of jobs...

Using shell: `/bin/bash`

Provided cores: 1 (use `--cores` to define parallelism)

Rules claiming more threads will be scaled down.

Job counts:

```
count jobs
1 all
1 bcfutils_call
```

This page also has metadata for how to interact with your workflow if you choose to run it again with Snakemake from the command line. A token and arguments for monitoring are required. At the bottom part of the page, there is a status table that updates automatically via a Web Socket.




SHOW 20 ENTRIES SEARCH:

Order	Level	Job	Message
0	warning		The flag 'directory' used in rule all is only valid for outputs, not inputs.
1	info		Building DAG of jobs...
2	dag_debug	all	
3	dag_debug	job1	
4	dag_debug	job1	
<div> <div>status</div> <div>selected</div> </div> <div> <div>job</div> <div>job1</div> </div> <div> <div>order</div> <div>4</div> </div>			
5	dag_debug		Producer found, hence exceptions are ignored.
6	dag_debug	all	
7	dag_debug		Producer found, hence exceptions are ignored.
8	info		Nothing to be done.
9	info		Complete log: /home/vanessa/Desktop/Code/snakeface/.snakemake/log/2020-12-30T223439.257769.snakemake.log
10	debug		unlocking
11	debug		removing lock
12	debug		removing lock
13	debug		removed all locks

Showing 1 to 14 of 14 entries Previous 1 Next

Finally, you'll also be able to see your workflows on the dashboard page in the Workflows table.

Dashboard Workflows Settings Log out 

New Workflow

SHOW 10 ENTRIES SEARCH:

ID	STATUS	SNAKEFILE	COMMAND	ACTIONS
5	Error	/home/vanessa/Desktop/Code/snakemake/ /snakemake-tutorial/Snakemakefile	snakemake --snakefile /home/vanessa/Desktop/Code/snakemake/snakemake-tutorial/Snakemakefile --local-cores 8 --wms-monitor http://127.0.0.1:5000 --max-inventory-time 20 --latency-wait 5 --max-jobs-per-second 10 --max-status-checks-per-second 10 --attempt 1 --wms-monitor-arg id=5	<a href="#">View</a>
7	Completed	/home/vanessa/Desktop/Code/snakeface/Snakemakefile	snakemake --snakefile /home/vanessa/Desktop/Code/snakeface/Snakemakefile --cores 1 --local-cores 8 --wms-monitor http://127.0.0.1:5000 --max-inventory-time 20 --latency-wait 5 --max-jobs-per-second 10 --max-status-checks-per-second 10 --attempt 1 --wms-monitor-arg id=7	<a href="#">View</a>
8	Error	/home/vanessa/Desktop/Code/snakemake/ /snakemake-tutorial/Snakemakefile	snakemake --snakefile /home/vanessa/Desktop/Code/snakemake/snakemake-tutorial/Snakemakefile --cores 1 --local-cores 8 --wms-monitor http://127.0.0.1:5000 --max-inventory-time 20 --latency-wait 5 --max-jobs-per-second 10 --max-status-checks-per-second 10 --attempt 1 --wms-monitor-arg id=8	<a href="#">View</a>

Showing 1 to 3 of 3 entries Previous 1 Next

## Continuing A Workflow

If you want to start a workflow from the command line to interact with a snakeface server, or you've already started one with Snakeface and want it to reference the same identifier again, you can easily run snakemake to do this by adding an environment variable for an authorization token, and a workflow id. If you look at the workflow details page above, you'll see that the token and command line arguments are provided for you. You might re-run an existing workflow like this:

```
export WMS_MONITOR_TOKEN=a2d0d2f2-dfa8-4fd6-b98c-f3219a2caa8c
snakemake --cores 1 --wms-monitor http://127.0.0.1:5000 --wms-monitor-arg id=3
```

## Workflow Reports

If you want to add a report file to the workflow, just as you would with command line Snakemake, you'll need to install additional dependencies first:

```
pip install snakemake[reports]
```

And then define your report.html file in the reports field.

### 3.1.4 Settings

Settings are defined in the settings.yml file, and are automatically populated into Snakeface. If you want a notebook, you will likely be good using the defaults.

Table 1: Title

Name	Description
GOOGLE_ANALYTICS_SITE	The url of your website for Google Analytics, if desired
GOOGLE_ANALYTICS_ID	The identifier for Google Analytics, if desired
TWITTER_USERNAME	A Twitter username to link to in the footer.
GITHUB_REPOSITORY	A GitHub repository to link to in the footer
GITHUB_DOCUMENTATION	GitHub documentation (or other) to link to in the footer
USER_WORKFLOW_LIMIT	The maximum number of workflows to allow a user to create
USER_WORKFLOW_RUNS_LIMIT	The maximum number of running workflows to allow
USER_WORKFLOW_GLOBAL_RUNS_LIMIT	Giving a shared Snakeface interface, the total maximum allowed running at once.
NOTEBOOK_ONLY	Only allow notebooks (disables all other auth)
MAXIMUM_NOTEBOOK_JOBS	Given a notebook, the maximum number of jobs to allow running at once
WORKFLOW_UPDATE_SECONDS	How often to refresh the status table on a workflow details page
EXECUTOR_CLUSTER	Set this to non null to enable the cluster executor
EXECUTOR_GOOGLE_LIFE_SCIENCES	Set this to non null to enable the GLS executor
EXECUTOR_KUBERNETES	Set this to non null to enable the K8 executor
EXECUTOR_GA4GH_TES	Set this to non null to enable this executor
EXECUTOR_TIBANNA	Set this to non null to enable the tibanna executor
DISABLE_SINGULARITY	Disable Singularity argument groups by setting this to non null
DISABLE_CONDA	Disable Conda argument groups by setting this to non null
DISABLE_NOTEBOOKS	Disable notebook argument groups by setting this to non null
ENVIRONMENT	The global name for the deployment environment
HELP_CONTACT_URL	The help contact email or url used for the API
SENDGRID_API_KEY	Not in use yet, will allow sending email notifications
SENDGRID_SENDER_EMAIL	Not in use yet, will allow sending email notifications
DOMAIN_NAME	The server domain name, defaults to a localhost address
DOMAIN_PORT	The server port, can be overridden from the command line
REQUIRE_AUTH	Should authentication be required?
PROFILE	Set a default profile (see <a href="https://github.com/snakemake-profiles">https://github.com/snakemake-profiles</a> )
PROFILE	Set a default profile (see <a href="https://github.com/snakemake-profiles">https://github.com/snakemake-profiles</a> )
PRIVATE_ONLY	Make all workflows private (not relevant for notebooks)
ENABLE_CACHE	Enable view caching
WORKDIR	Default working directory (overridden by client and environment)
PLUGINS_LDAP_AUTH_ENABLED	Set to non null to enable
PLUGINS_PAM_AUTH_ENABLED	Set to non null to enable
PLUGINS_SAML_AUTH_ENABLED	Set to non null to enable

### 3.1.5 Authentication

If you don't define an authentication backend (e.g., plugins like ldap, saml, or OAuth 2), then the default authentication model for Snakeface is akin to a jupyter notebook. You'll be given a token to enter in the interface, and this will log you in. This is currently the only authentication supported, as we haven't developed the other deployment types.

## 3.2 Use Cases

Snakeface is intended to be flexible to different needs for your deployment. This means that you can both run it quickly as a notebook *Notebook* to test a workflow, or deploy it in a cluster environment for your user base. If you have a need for deployment that is not addressed here, please [let us know](#)

## 3.3 The Snakeface API

These sections detail the internal functions for Snakeface.

## 3.4 Internal API

These pages document the entire internal API of Snakeface.

### 3.4.1 snakeface package

#### Submodules

#### snakeface.argparser module

**class** snakeface.argparser.**SnakefaceArgument** (*action, required=False*)

Bases: object

A Snakeface argument takes an action from a parser, and is able to easily generate front end views (e.g., a form element) for it

**boolean\_field**()

generate a boolean field (radio button) via a jinja2 template

**choice\_field**()

generate a choice field for using a pre-loaded jinja2 template

**field**()

generate a form field for the argument

**property field\_name**

**property is\_boolean**

**load\_template** (*path*)

Given a path to a template file, load the template with jinja2

**text\_field**()

generate a text field for using a pre-loaded jinja2 template

**update\_choice\_fields** (*updates*)

**class** snakeface.argparser.SnakefaceParser

Bases: object

A Snakeface Parser is a wrapper to an argparse.Parser, and aims to make it easy to loop over arguments and options, and generate various representations (e.g., an input field) for the interface. The point is not to use it to parse arguments and validate, but to output all fields to a front end form.

**property** command

Given a loaded set of arguments, generate the command.

**property** errors

**get** (name, default=None)

A general get function to return an argument that might be nested under a group. These objects are the same as linked in `_groups`.

**property** groups

yield arguments organized by groups, with the intention to easily map into a form on the front end. The groups seem to have ALL arguments each, so we have to artificially separate them.

**include\_argument** (name, group)

Given an argument name, and a group name, skip if settings disable it

**load** (argdict)

Load is a wrapper around set - we loop through a dictionary and set all arguments.

**property** required

**set** (name, value)

Set a value for an argument. This is typically what the user has selected.

**property** snakefile

**snakefiles** = []

**to\_dict** ()

the opposite of load, this function exports an argument

**validate** ()

ensure that all required args are defined

## snakeface.client module

snakeface.client.get\_parser()

snakeface.client.main()

main entrypoint for snakeface

## snakeface.apps.api module

**class** snakeface.apps.api.permissions.AllowAnyGet

Bases: rest\_framework.permissions.BasePermission

Allows an anonymous user access for GET requests only.

**has\_permission** (request, view)

Return *True* if permission is granted, *False* otherwise.

snakeface.apps.api.permissions.check\_user\_authentication(request)

Given a request, check that the user is authenticated via a token in the header.

`snakeface.apps.api.permissions.get_token(request)`

The same as `validate_token`, but return the token object to check the associated user.

**class** `snakeface.apps.api.views.CreateWorkflow(**kwargs)`

Bases: `ratelimit.mixins.RatelimitMixin`, `rest_framework.views.APIView`

Create a snakemake workflow. Given that we provide an API token, we expect the workflow model to already be created and simply generate a run for it.

`get(request)`

`ratelimit_block = True`

`ratelimit_key = 'ip'`

`ratelimit_method = 'GET'`

`ratelimit_rate = '1000/1d'`

`renderer_classes = (<class 'rest_framework.renderers.JSONRenderer'>,)`

**class** `snakeface.apps.api.views.ServiceInfo(**kwargs)`

Bases: `ratelimit.mixins.RatelimitMixin`, `rest_framework.views.APIView`

Return a 200 response to indicate a running service. Note that we are not currently including all required fields. See: <https://ga4gh.github.io/workflow-execution-service-schemas/docs/#operation/GetServiceInfo>

`get(request)`

`ratelimit_block = True`

`ratelimit_key = 'ip'`

`ratelimit_method = 'GET'`

`ratelimit_rate = '1000/1d'`

`renderer_classes = (<class 'rest_framework.renderers.JSONRenderer'>,)`

**class** `snakeface.apps.api.views.UpdateWorkflow(**kwargs)`

Bases: `ratelimit.mixins.RatelimitMixin`, `rest_framework.views.APIView`

Update an existing snakemake workflow. Authentication is required, and the workflow must exist.

`post(request)`

`ratelimit_block = True`

`ratelimit_key = 'ip'`

`ratelimit_method = 'POST'`

`ratelimit_rate = '1000/1d'`

`renderer_classes = (<class 'rest_framework.renderers.JSONRenderer'>,)`

**snakeface.apps.main module**

```
class snakeface.apps.main.consumers.WorkflowConsumer(*args, **kwargs)
```

Bases: `channels.generic.websocket.AsyncJsonWebsocketConsumer`

```
async connect()
```

```
async disconnect(close_code)
```

Called when a WebSocket connection is closed.

```
async receive(text_data)
```

Called with a decoded WebSocket frame.

```
async update_workflow_status()
```

```
snakeface.apps.main.consumers.async_get_statuses(workflow_id)
```

Return a dictionary of workflow statuses on success. If the workflow doesn't exist, then return False and we disconnect from the socket.

```
snakeface.apps.main.consumers.get_statuses(workflow_id)
```

Return a dictionary of workflow statuses on success. If the workflow doesn't exist, then return False and we disconnect from the socket.

```
class snakeface.apps.main.forms.WorkflowForm(data=None, files=None,
auto_id='id_%s', prefix=None, initial=None, error_class=<class
'django.forms.utils.ErrorList'>, label_suffix=None,
empty_permitted=False, instance=None, use_required_attribute=None,
renderer=None)
```

Bases: `django.forms.models.ModelForm`

```
class Meta
```

Bases: `object`

```
fields = ['name', 'workdirs']
```

```
model
```

alias of `snakeface.apps.main.models.Workflow`

```
base_fields = {'name': <django.forms.fields.CharField object>, 'workdirs': <django.f
```

```
declared_fields = {'workdirs': <django.forms.fields.ChoiceField object>}
```

```
property media
```

Return all media required to render the widgets on this form.

```
class snakeface.apps.main.models.JSONField(verbose_name=None, name=None, pri-
mary_key=False, max_length=None,
unique=False, blank=False, null=False,
db_index=False, rel=None, default=<class
'django.db.models.fields.NOT_PROVIDED'>,
editable=True, serialize=True,
unique_for_date=None,
unique_for_month=None,
unique_for_year=None, choices=None,
help_text='', db_column=None,
db_tablespace=None, auto_created=False,
validators=(), error_messages=None)
```

Bases: `django.db.models.fields.Field`

**db\_type** (*connection*)

Return the database column data type for this field, for the provided connection.

**from\_db\_value** (*value, expression, connection*)

**get\_prep\_value** (*value*)

Perform preliminary non-db specific value checks and conversions.

**to\_python** (*value*)

Convert the input value into the expected Python data type, raising `django.core.exceptions.ValidationError` if the data can't be converted. Return the converted value. Subclasses should override this.

**value\_to\_string** (*obj*)

Return a string value of this field from the passed obj. This is used by the serialization framework.

**class** `snakeface.apps.main.models.Workflow` (*\*args, \*\*kwargs*)

Bases: `django.db.models.base.Model`

A workflow is associated with a specific git repository and one or more workflow runs.

**exception DoesNotExist**

Bases: `django.core.exceptions.ObjectDoesNotExist`

**exception MultipleObjectsReturned**

Bases: `django.core.exceptions.MultipleObjectsReturned`

**add\_date**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**command**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**contributors**

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

`Pizza.toppings` and `Topping.pizzas` are `ManyToManyDescriptor` instances.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

**dag**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**data**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**error**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**get\_absolute\_url** ()

**get\_label** ()

```
get_next_by_add_date (*, field=<django.db.models.fields.DateTimeField: add_date>,
                      is_next=True, **kwargs)
get_next_by_modify_date (*, field=<django.db.models.fields.DateTimeField: modify_date>,
                        is_next=True, **kwargs)
get_previous_by_add_date (*, field=<django.db.models.fields.DateTimeField: add_date>,
                        is_next=False, **kwargs)
get_previous_by_modify_date (*, field=<django.db.models.fields.DateTimeField: modify_date>,
                            is_next=False, **kwargs)
get_private_display (*, field=<django.db.models.fields.BooleanField: private>)
get_report ()
    load the report file, if it exists.
get_status_display (*, field=<django.db.models.fields.TextField: status>)
has_edit_permission ()
    If we are running in a notebook environment, there is just one user that has edit access to anything. Otherwise, the user must be an owner
has_report ()
    returns True if the workflow command has a designated report, and the report file exists
has_view_permission ()
id
    A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.
property members
property message_fields
modify_date
    A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.
name
    A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.
objects = <django.db.models.manager.Manager object>
output
    A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.
```

**owners**

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

`Pizza.toppings` and `Topping.pizzas` are `ManyToManyDescriptor` instances.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.



**private**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**reset ()**

Empty all run related fields to prepare for a new run.

**retval**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**snakefile**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**snakemake\_id**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**status**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**thread**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**update\_command** (*command=None, do\_save=False*)

Given a command (or an automated save from the signal) update the command for the workflow.

**update\_dag** (*do\_save=False*)

given a snakefile, run the command to update the dag

**workdir**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**workflowstatus\_set**

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Parent.children is a ReverseManyToOneDescriptor instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

```
class snakeface.apps.main.models.WorkflowStatus(*args, **kwargs)
```

Bases: `django.db.models.base.Model`

A workflow status is a status message send from running a workflow

**exception DoesNotExist**

Bases: `django.core.exceptions.ObjectDoesNotExist`

**exception MultipleObjectsReturned**

Bases: `django.core.exceptions.MultipleObjectsReturned`

**add\_date**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

```
get_next_by_add_date (*, field=<django.db.models.fields.DateTimeField: add_date>,
                      is_next=True, **kwargs)
```

```
get_next_by_modify_date (*, field=<django.db.models.fields.DateTimeField: modify_date>,
                          is_next=True, **kwargs)
```

```
get_previous_by_add_date (*, field=<django.db.models.fields.DateTimeField: add_date>,
                           is_next=False, **kwargs)
```

```
get_previous_by_modify_date (*, field=<django.db.models.fields.DateTimeField: modify_date>,
                              is_next=False, **kwargs)
```

**id**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**modify\_date**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**msg**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**objects** = <django.db.models.manager.Manager object>

**workflow**

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

Child.parent is a ForwardManyToOneDescriptor instance.

**workflow\_id**

```
snakeface.apps.main.models.update_workflow(sender, instance, **kwargs)
```

```
snakeface.apps.main.tasks.doRun(wid, uid)
```

The task to run a workflow

```
snakeface.apps.main.tasks.run_is_allowed(request)
```

Given a request, check that the run is allowed meaning: 1. If running a notebook, we aren't over quota for jobs  
2. If not running a notebook, we aren't over user or global limits

```
snakeface.apps.main.tasks.run_workflow(request, wid, uid)
```

Top level function to ensure that the user has permission to do the run, and we direct to the correct function (notebook or not written, another backend)

```
snakeface.apps.main.tasks.serialize_workflow_statuses(workflow)
```

A shared helper function to serialize a list of workflow statuses into json.

```
class snakeface.apps.main.utils.CommandRunner
```

Bases: object

Wrapper to use subprocess to run a command. This is based off of pypi vendor distlib SubprocesMixin.

**reader** (*stream, context*)

Get output and error lines and save to command runner.

**reset** ()

**run\_command** (*cmd, env=None, cancel\_func=None, cancel\_func\_kwargs=None, \*\*kwargs*)

**class** snakeface.apps.main.utils.**ThreadRunner** (*group=None, target=None, name=None, args=(), kwargs=None, \*, daemon=None*)

Bases: threading.Thread

We need to be able to run a Snakemake job as a thread, and kill it if an exception is raised based on it's id

**set\_workflow** (*workflow*)

**property thread\_id**

Return the id of the thread, either attributed to the class or by matching the Thread instance

snakeface.apps.main.utils.**get\_snakefile\_choices** (*path=None*)

Given the working directory set on init, return all discovered snakefiles.

snakeface.apps.main.utils.**get\_tmpfile** (*prefix="", suffix=""*)

get a temporary file with an optional prefix. By default, the file is closed (and just a name returned).

**Parameters prefix** (-) – prefix with this string

snakeface.apps.main.utils.**get\_workdir\_choices** (*path=None*)

Given the working directory set on init, return potential subdirectories.

snakeface.apps.main.utils.**read\_file** (*filename*)

Write some text content to a file

snakeface.apps.main.utils.**write\_file** (*filename, content*)

Write some text content to a file

snakeface.apps.main.views.**edit\_or\_update\_workflow** (*request, parser, workflow=None*)

A shared function to edit or update an existing workflow.

snakeface.apps.main.views.**view\_workflow\_report** (*request, wid*)

If a workflow generated a report and the report exists, render it to a page

## snakeface.apps.base module

snakeface.apps.base.views.**warmup** ()

## snakeface.apps.users module

snakeface.apps.users.decorators.**login\_is\_required** (*function=None, login\_url=None, redirect\_field\_name='next'*)

Decorator to extend login required to also check if a notebook auth is desired first.

**class** snakeface.apps.users.forms.**TokenForm** (*data=None, files=None, auto\_id='id\_%s', prefix=None, initial=None, error\_class=<class 'django.forms.utils.ErrorList'>, label\_suffix=None, empty\_permitted=False, field\_order=None, use\_required\_attribute=None, renderer=None*)

Bases: django.forms.forms.Form

**base\_fields** = {'token': <django.forms.fields.CharField object>}

```
declared_fields = {'token': <django.forms.fields.CharField object>}
```

**property media**

Return all media required to render the widgets on this form.

```
class snakeface.apps.users.models.CustomUserManager(*args, **kwargs)
```

Bases: `django.contrib.auth.base_user.BaseUserManager`

**add\_staff** (*user*)

Intended for existing user

**add\_superuser** (*user*)

Intended for existing user

```
create_superuser(username, email, password, **extra_fields)
```

```
create_user(username, email=None, password=None, **extra_fields)
```

```
class snakeface.apps.users.models.User(id, password, last_login, is_superuser, username,
                                         first_name, last_name, email, is_staff, is_active,
                                         date_joined, active, agree_terms, agree_terms_date,
                                         notebook_token)
```

Bases: `django.contrib.auth.models.AbstractUser`

**exception DoesNotExist**

Bases: `django.core.exceptions.ObjectDoesNotExist`

**exception MultipleObjectsReturned**

Bases: `django.core.exceptions.MultipleObjectsReturned`

**active**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**agree\_terms**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**agree\_terms\_date**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**auth\_token**

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Place.restaurant` is a `ReverseOneToOneDescriptor` instance.

**get\_credentials** (*provider*)

return one or more credentials, or None

**get\_label** ()

```
get_next_by_date_joined(*, field=<django.db.models.fields.DateTimeField: date_joined>,
                        is_next=True, **kwargs)
```

```
get_previous_by_date_joined(*, field=<django.db.models.fields.DateTimeField:
date_joined>, is_next=False, **kwargs)
```

**get\_providers()**

return a list of providers that the user has credentials for.

**groups**

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

`Pizza.toppings` and `Topping.pizzas` are `ManyToManyDescriptor` instances.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

**has\_create\_permission()**

`has_create_permission` determines if the user (globally) can create new collections. By default, superusers and admin can, along with regular users if `USER_COLLECTIONS` is `True`. Otherwise, not.

**id**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**logentry\_set**

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

**notebook\_token**

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

**objects = <snakeface.apps.users.models.CustomUserManager object>**

**social\_auth**

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

**property token**

The user token is for interaction with creating and updating workflows

**user\_permissions**

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

`Pizza.toppings` and `Topping.pizzas` are `ManyToManyDescriptor` instances.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

#### **workflow\_contributors**

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

`Pizza.toppings` and `Topping.pizzas` are `ManyToManyDescriptor` instances.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

#### **workflow\_owners**

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

`Pizza.toppings` and `Topping.pizzas` are `ManyToManyDescriptor` instances.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

`snakeface.apps.users.models.create_auth_token(sender, instance=None, created=False, **kwargs)`

Create a token for the user when the user is created (with `OAuth2`)

1. Assign user a token
2. Assign user to default group

Create a `Profile` instance for all newly created `User` instances. We only run on user creation to avoid having to check for existence on each call to `User.save`.

`snakeface.apps.users.utils.get_notebook_token(request, verbose=True)`

If a notebook token isn't defined, generate it (and print to the console) The token is used to generate a user to log the user in.

`snakeface.apps.users.utils.get_notebook_user()`

Get the notebook user, if they have logged in before.

`snakeface.apps.users.utils.get_or_create_notebook_user(token)`

Get or create the notebook user. Imports are done in the function because Django startup (`settings.py`) uses these functions.

`snakeface.apps.users.utils.get_username()`

get the username based on the effective uid. This is for a notebook execution, and doesn't add any additional security, but rather is used for personalization and being able to create an associated django user.

`snakeface.apps.users.views.notebook_login(request)`

Given the user doesn't have a token in the request session, ask for it.

## snakeface.settings module

```
class snakeface.settings.Settings (dictionary)
    Bases: object

    convert a dictionary of settings (from yaml) into a class

snakeface.settings.generate_secret_key (filename)
    A helper function to write a randomly generated secret key to file
```

## snakeface.logger module

```
class snakeface.logger.ColorizingStreamHandler (nocolor=False,
                                                stream=<_io.TextIOWrapper
                                                name='<stderr>' mode='w'
                                                encoding='UTF-8'>,
                                                use_threads=False)

    Bases: logging.StreamHandler

    BLACK = 0
    BLUE = 4
    BOLD_SEQ = '\x1b[1m'
    COLOR_SEQ = '\x1b[%dm'
    CYAN = 6
    GREEN = 2
    MAGENTA = 5
    RED = 1
    RESET_SEQ = '\x1b[0m'
    WHITE = 7
    YELLOW = 3

    can_color_tty()

    colors = {'CRITICAL': 1, 'DEBUG': 4, 'ERROR': 1, 'INFO': 2, 'WARNING': 3}

    decorate (record)

    emit (record)
        Emit a record.

        If a formatter is specified, it is used to format the record. The record is then written to the stream with a trailing newline. If exception information is present, it is formatted using traceback.print_exception and appended to the stream. If the stream has an 'encoding' attribute, it is used to determine how to do the output to the stream.

    property is_tty

class snakeface.logger.Logger
    Bases: object

    cleanup ()

    debug (msg)

    error (msg)
```

```
exit (msg, return_code=1)  
handler (msg)  
info (msg)  
location (msg)  
progress (done=None, total=None)  
set_level (level)  
set_stream_handler (stream_handler)  
shellcmd (msg)  
text_handler (msg)  
    The default snakemake log handler. Prints the output to the console. :param msg: the log message dictio-  
    nary :type msg: dict  
warning (msg)  
snakeface.logger.setup_logger (quiet=False, printshellcmds=False, nocolor=False, stdout=False,  
                                debug=False, use_threads=False, wms_monitor=None)
```



## PYTHON MODULE INDEX

### S

- `snakeface`, [15](#)
- `snakeface.apps.api.permissions`, [16](#)
- `snakeface.apps.api.urls`, [17](#)
- `snakeface.apps.api.views`, [17](#)
- `snakeface.apps.base.urls`, [23](#)
- `snakeface.apps.base.views`, [23](#)
- `snakeface.apps.main.consumers`, [18](#)
- `snakeface.apps.main.forms`, [18](#)
- `snakeface.apps.main.models`, [18](#)
- `snakeface.apps.main.routing`, [22](#)
- `snakeface.apps.main.tasks`, [22](#)
- `snakeface.apps.main.urls`, [22](#)
- `snakeface.apps.main.utils`, [22](#)
- `snakeface.apps.main.views`, [23](#)
- `snakeface.apps.users.decorators`, [23](#)
- `snakeface.apps.users.forms`, [23](#)
- `snakeface.apps.users.models`, [24](#)
- `snakeface.apps.users.urls`, [26](#)
- `snakeface.apps.users.utils`, [26](#)
- `snakeface.apps.users.views`, [26](#)
- `snakeface.argparser`, [15](#)
- `snakeface.client`, [16](#)
- `snakeface.logger`, [27](#)
- `snakeface.settings`, [27](#)



## A

active (*snakeface.apps.users.models.User* attribute), 24

add\_date (*snakeface.apps.main.models.Workflow* attribute), 19

add\_date (*snakeface.apps.main.models.WorkflowStatus* attribute), 21

add\_staff() (*snakeface.apps.users.models.CustomUserManager* method), 24

add\_superuser() (*snakeface.apps.users.models.CustomUserManager* method), 24

agree\_terms (*snakeface.apps.users.models.User* attribute), 24

agree\_terms\_date (*snakeface.apps.users.models.User* attribute), 24

AllowAnyGet (class in *snakeface.apps.api.permissions*), 16

async\_get\_statuses() (in module *snakeface.apps.main.consumers*), 18

auth\_token (*snakeface.apps.users.models.User* attribute), 24

## B

base\_fields (*snakeface.apps.main.forms.WorkflowForm* attribute), 18

base\_fields (*snakeface.apps.users.forms.TokenForm* attribute), 23

BLACK (*snakeface.logger.ColorizingStreamHandler* attribute), 27

BLUE (*snakeface.logger.ColorizingStreamHandler* attribute), 27

BOLD\_SEQ (*snakeface.logger.ColorizingStreamHandler* attribute), 27

boolean\_field() (*snakeface.argparser.SnakefaceArgument* method), 15

## C

can\_color\_tty() (*snake-*

*face.logger.ColorizingStreamHandler* method), 27

check\_user\_authentication() (in module *snakeface.apps.api.permissions*), 16

choice\_field() (*snakeface.argparser.SnakefaceArgument* method), 15

cleanup() (*snakeface.logger.Logger* method), 27

COLOR\_SEQ (*snakeface.logger.ColorizingStreamHandler* attribute), 27

ColorizingStreamHandler (class in *snakeface.logger*), 27

colors (*snakeface.logger.ColorizingStreamHandler* attribute), 27

command (*snakeface.apps.main.models.Workflow* attribute), 19

command() (*snakeface.argparser.SnakefaceParser* property), 16

CommandRunner (class in *snakeface.apps.main.utils*), 22

connect() (*snakeface.apps.main.consumers.WorkflowConsumer* method), 18

contributors (*snakeface.apps.main.models.Workflow* attribute), 19

create\_auth\_token() (in module *snakeface.apps.users.models*), 26

create\_superuser() (*snakeface.apps.users.models.CustomUserManager* method), 24

create\_user() (*snakeface.apps.users.models.CustomUserManager* method), 24

CreateWorkflow (class in *snakeface.apps.api.views*), 17

CustomUserManager (class in *snakeface.apps.users.models*), 24

CYAN (*snakeface.logger.ColorizingStreamHandler* attribute), 27

## D

dag (*snakeface.apps.main.models.Workflow* attribute),

19  
data (snakeface.apps.main.models.Workflow attribute),  
19  
db\_type() (snakeface.apps.main.models.JSONField  
method), 18  
debug() (snakeface.logger.Logger method), 27  
declared\_fields (snake-  
face.apps.main.forms.WorkflowForm attribute),  
18  
declared\_fields (snake-  
face.apps.users.forms.TokenForm attribute),  
23  
decorate() (snakeface.logger.ColorizingStreamHandler  
method), 27  
disconnect() (snake-  
face.apps.main.consumers.WorkflowConsumer  
method), 18  
doRun() (in module snakeface.apps.main.tasks), 22

## E

edit\_or\_update\_workflow() (in module snake-  
face.apps.main.views), 23  
emit() (snakeface.logger.ColorizingStreamHandler  
method), 27  
error (snakeface.apps.main.models.Workflow at-  
tribute), 19  
error() (snakeface.logger.Logger method), 27  
errors() (snakeface.argparser.SnakefaceParser prop-  
erty), 16  
exit() (snakeface.logger.Logger method), 27

## F

field() (snakeface.argparser.SnakefaceArgument  
method), 15  
field\_name() (snake-  
face.argparser.SnakefaceArgument property),  
15  
fields (snakeface.apps.main.forms.WorkflowForm.Meta  
attribute), 18  
from\_db\_value() (snake-  
face.apps.main.models.JSONField method),  
19

## G

generate\_secret\_key() (in module snake-  
face.settings), 27  
get() (snakeface.apps.api.views.CreateWorkflow  
method), 17  
get() (snakeface.apps.api.views.ServiceInfo method),  
17  
get() (snakeface.argparser.SnakefaceParser method),  
16

get\_absolute\_url() (snake-  
face.apps.main.models.Workflow method),  
19  
get\_credentials() (snake-  
face.apps.users.models.User method), 24  
get\_label() (snakeface.apps.main.models.Workflow  
method), 19  
get\_label() (snakeface.apps.users.models.User  
method), 24  
get\_next\_by\_add\_date() (snake-  
face.apps.main.models.Workflow method),  
19  
get\_next\_by\_add\_date() (snake-  
face.apps.main.models.WorkflowStatus  
method), 22  
get\_next\_by\_date\_joined() (snake-  
face.apps.users.models.User method), 24  
get\_next\_by\_modify\_date() (snake-  
face.apps.main.models.Workflow method),  
20  
get\_next\_by\_modify\_date() (snake-  
face.apps.main.models.WorkflowStatus  
method), 22  
get\_notebook\_token() (in module snake-  
face.apps.users.utils), 26  
get\_notebook\_user() (in module snake-  
face.apps.users.utils), 26  
get\_or\_create\_notebook\_user() (in module  
snakeface.apps.users.utils), 26  
get\_parser() (in module snakeface.client), 16  
get\_prep\_value() (snake-  
face.apps.main.models.JSONField method),  
19  
get\_previous\_by\_add\_date() (snake-  
face.apps.main.models.Workflow method),  
20  
get\_previous\_by\_add\_date() (snake-  
face.apps.main.models.WorkflowStatus  
method), 22  
get\_previous\_by\_date\_joined() (snake-  
face.apps.users.models.User method), 24  
get\_previous\_by\_modify\_date() (snake-  
face.apps.main.models.Workflow method),  
20  
get\_previous\_by\_modify\_date() (snake-  
face.apps.main.models.WorkflowStatus  
method), 22  
get\_private\_display() (snake-  
face.apps.main.models.Workflow method),  
20  
get\_providers() (snake-  
face.apps.users.models.User method), 24  
get\_report() (snake-  
face.apps.main.models.Workflow method),

20  
 get\_snakefile\_choices() (in module *snakeface.apps.main.utils*), 23  
 get\_status\_display() (snakeface.apps.main.models.Workflow method), 20  
 get\_statuses() (in module *snakeface.apps.main.consumers*), 18  
 get\_tmpfile() (in module *snakeface.apps.main.utils*), 23  
 get\_token() (in module *snakeface.apps.api.permissions*), 16  
 get\_username() (in module *snakeface.apps.users.utils*), 26  
 get\_workdir\_choices() (in module *snakeface.apps.main.utils*), 23  
 GREEN (snakeface.logger.ColorizingStreamHandler attribute), 27  
 groups (snakeface.apps.users.models.User attribute), 25  
 groups() (snakeface.argparser.SnakefaceParser property), 16

## H

handler() (snakeface.logger.Logger method), 28  
 has\_create\_permission() (snakeface.apps.users.models.User method), 25  
 has\_edit\_permission() (snakeface.apps.main.models.Workflow method), 20  
 has\_permission() (snakeface.apps.api.permissions.AllowAnyGet method), 16  
 has\_report() (snakeface.apps.main.models.Workflow method), 20  
 has\_view\_permission() (snakeface.apps.main.models.Workflow method), 20

## I

id (snakeface.apps.main.models.Workflow attribute), 20  
 id (snakeface.apps.main.models.WorkflowStatus attribute), 22  
 id (snakeface.apps.users.models.User attribute), 25  
 include\_argument() (snakeface.argparser.SnakefaceParser method), 16  
 info() (snakeface.logger.Logger method), 28  
 is\_boolean() (snakeface.argparser.SnakefaceArgument property), 15  
 is\_tty() (snakeface.logger.ColorizingStreamHandler property), 27

## J

JSONField (class in *snakeface.apps.main.models*), 18

## L

load() (snakeface.argparser.SnakefaceParser method), 16  
 load\_template() (snakeface.argparser.SnakefaceArgument method), 15  
 location() (snakeface.logger.Logger method), 28  
 logentry\_set (snakeface.apps.users.models.User attribute), 25  
 Logger (class in *snakeface.logger*), 27  
 login\_is\_required() (in module *snakeface.apps.users.decorators*), 23

## M

MAGENTA (snakeface.logger.ColorizingStreamHandler attribute), 27  
 main() (in module *snakeface.client*), 16  
 media() (snakeface.apps.main.forms.WorkflowForm property), 18  
 media() (snakeface.apps.users.forms.TokenForm property), 24  
 members() (snakeface.apps.main.models.Workflow property), 20  
 message\_fields() (snakeface.apps.main.models.Workflow property), 20  
 model (snakeface.apps.main.forms.WorkflowForm.Meta attribute), 18  
 modify\_date (snakeface.apps.main.models.Workflow attribute), 20  
 modify\_date (snakeface.apps.main.models.WorkflowStatus attribute), 22  
 module  
   snakeface, 15  
   snakeface.apps.api.permissions, 16  
   snakeface.apps.api.urls, 17  
   snakeface.apps.api.views, 17  
   snakeface.apps.base.urls, 23  
   snakeface.apps.base.views, 23  
   snakeface.apps.main.consumers, 18  
   snakeface.apps.main.forms, 18  
   snakeface.apps.main.models, 18  
   snakeface.apps.main.routing, 22  
   snakeface.apps.main.tasks, 22  
   snakeface.apps.main.urls, 22  
   snakeface.apps.main.utils, 22  
   snakeface.apps.main.views, 23  
   snakeface.apps.users.decorators, 23  
   snakeface.apps.users.forms, 23

snakeface.apps.users.models, 24  
snakeface.apps.users.urls, 26  
snakeface.apps.users.utils, 26  
snakeface.apps.users.views, 26  
snakeface.argparser, 15  
snakeface.client, 16  
snakeface.logger, 27  
snakeface.settings, 27  
msg (snakeface.apps.main.models.WorkflowStatus attribute), 22

## N

name (snakeface.apps.main.models.Workflow attribute), 20  
notebook\_login() (in module snakeface.apps.users.views), 26  
notebook\_token (snakeface.apps.users.models.User attribute), 25

## O

objects (snakeface.apps.main.models.Workflow attribute), 20  
objects (snakeface.apps.main.models.WorkflowStatus attribute), 22  
objects (snakeface.apps.users.models.User attribute), 25  
output (snakeface.apps.main.models.Workflow attribute), 20  
owners (snakeface.apps.main.models.Workflow attribute), 20

## P

post() (snakeface.apps.api.views.UpdateWorkflow method), 17  
private (snakeface.apps.main.models.Workflow attribute), 20  
progress() (snakeface.logger.Logger method), 28

## R

ratelimit\_block (snakeface.apps.api.views.CreateWorkflow attribute), 17  
ratelimit\_block (snakeface.apps.api.views.ServiceInfo attribute), 17  
ratelimit\_block (snakeface.apps.api.views.UpdateWorkflow attribute), 17  
ratelimit\_key (snakeface.apps.api.views.CreateWorkflow attribute), 17  
ratelimit\_key (snakeface.apps.api.views.ServiceInfo attribute), 17

ratelimit\_key (snakeface.apps.api.views.UpdateWorkflow attribute), 17  
ratelimit\_method (snakeface.apps.api.views.CreateWorkflow attribute), 17  
ratelimit\_method (snakeface.apps.api.views.ServiceInfo attribute), 17  
ratelimit\_method (snakeface.apps.api.views.UpdateWorkflow attribute), 17  
ratelimit\_rate (snakeface.apps.api.views.CreateWorkflow attribute), 17  
ratelimit\_rate (snakeface.apps.api.views.ServiceInfo attribute), 17  
ratelimit\_rate (snakeface.apps.api.views.UpdateWorkflow attribute), 17  
read\_file() (in module snakeface.apps.main.utils), 23  
reader() (snakeface.apps.main.utils.CommandRunner method), 22  
receive() (snakeface.apps.main.consumers.WorkflowConsumer method), 18  
RED (snakeface.logger.ColorizingStreamHandler attribute), 27  
renderer\_classes (snakeface.apps.api.views.CreateWorkflow attribute), 17  
renderer\_classes (snakeface.apps.api.views.ServiceInfo attribute), 17  
renderer\_classes (snakeface.apps.api.views.UpdateWorkflow attribute), 17  
required() (snakeface.argparser.SnakefaceParser property), 16  
reset() (snakeface.apps.main.models.Workflow method), 21  
reset() (snakeface.apps.main.utils.CommandRunner method), 23  
RESET\_SEQ (snakeface.logger.ColorizingStreamHandler attribute), 27  
retval (snakeface.apps.main.models.Workflow attribute), 21  
run\_command() (snakeface.apps.main.utils.CommandRunner method), 23  
run\_is\_allowed() (in module snakeface.apps.main.tasks), 22  
run\_workflow() (in module snake-

*face.apps.main.tasks*), 22

## S

`serialize_workflow_statuses()` (in module *snakeface.apps.main.tasks*), 22

`ServiceInfo` (class in *snakeface.apps.api.views*), 17

`set()` (*snakeface.argparser.SnakefaceParser* method), 16

`set_level()` (*snakeface.logger.Logger* method), 28

`set_stream_handler()` (*snakeface.logger.Logger* method), 28

`set_workflow()` (*snakeface.apps.main.utils.ThreadRunner* method), 23

`Settings` (class in *snakeface.settings*), 27

`setup_logger()` (in module *snakeface.logger*), 28

`shellcmd()` (*snakeface.logger.Logger* method), 28

`snakeface`

module, 15

`snakeface.apps.api.permissions`

module, 16

`snakeface.apps.api.urls`

module, 17

`snakeface.apps.api.views`

module, 17

`snakeface.apps.base.urls`

module, 23

`snakeface.apps.base.views`

module, 23

`snakeface.apps.main.consumers`

module, 18

`snakeface.apps.main.forms`

module, 18

`snakeface.apps.main.models`

module, 18

`snakeface.apps.main.routing`

module, 22

`snakeface.apps.main.tasks`

module, 22

`snakeface.apps.main.urls`

module, 22

`snakeface.apps.main.utils`

module, 22

`snakeface.apps.main.views`

module, 23

`snakeface.apps.users.decorators`

module, 23

`snakeface.apps.users.forms`

module, 23

`snakeface.apps.users.models`

module, 24

`snakeface.apps.users.urls`

module, 26

`snakeface.apps.users.utils`

module, 26

`snakeface.apps.users.views`

module, 26

`snakeface.argparser`

module, 15

`snakeface.client`

module, 16

`snakeface.logger`

module, 27

`snakeface.settings`

module, 27

`SnakefaceArgument` (class in *snakeface.argparser*), 15

`SnakefaceParser` (class in *snakeface.argparser*), 15

`snakefile` (*snakeface.apps.main.models.Workflow* attribute), 21

`snakefile()` (*snakeface.argparser.SnakefaceParser* property), 16

`snakefiles` (*snakeface.argparser.SnakefaceParser* attribute), 16

`snakemake_id` (*snakeface.apps.main.models.Workflow* attribute), 21

`social_auth` (*snakeface.apps.users.models.User* attribute), 25

`status` (*snakeface.apps.main.models.Workflow* attribute), 21

## T

`text_field()` (*snakeface.argparser.SnakefaceArgument* method), 15

`text_handler()` (*snakeface.logger.Logger* method), 28

`thread` (*snakeface.apps.main.models.Workflow* attribute), 21

`thread_id()` (*snakeface.apps.main.utils.ThreadRunner* property), 23

`ThreadRunner` (class in *snakeface.apps.main.utils*), 23

`to_dict()` (*snakeface.argparser.SnakefaceParser* method), 16

`to_python()` (*snakeface.apps.main.models.JSONField* method), 19

`token()` (*snakeface.apps.users.models.User* property), 25

`TokenForm` (class in *snakeface.apps.users.forms*), 23

## U

`update_choice_fields()` (*snakeface.argparser.SnakefaceArgument* method), 15



`update_command()` (*snakeface.apps.main.models.Workflow* method), 21

`update_dag()` (*snakeface.apps.main.models.Workflow* method), 21

`update_workflow()` (*in module snakeface.apps.main.models*), 22

`update_workflow_status()` (*snakeface.apps.main.consumers.WorkflowConsumer* method), 18

`UpdateWorkflow` (*class in snakeface.apps.api.views*), 17

`User` (*class in snakeface.apps.users.models*), 24

`User.DoesNotExist`, 24

`User.MultipleObjectsReturned`, 24

`user_permissions` (*snakeface.apps.users.models.User* attribute), 25

## V

`validate()` (*snakeface.argparser.SnakefaceParser* method), 16

`value_to_string()` (*snakeface.apps.main.models.JSONField* method), 19

`view_workflow_report()` (*in module snakeface.apps.main.views*), 23

## W

`warmup()` (*in module snakeface.apps.base.views*), 23

`warning()` (*snakeface.logger.Logger* method), 28

`WHITE` (*snakeface.logger.ColorizingStreamHandler* attribute), 27

`workdir` (*snakeface.apps.main.models.Workflow* attribute), 21

`Workflow` (*class in snakeface.apps.main.models*), 19

`workflow` (*snakeface.apps.main.models.WorkflowStatus* attribute), 22

`Workflow.DoesNotExist`, 19

`Workflow.MultipleObjectsReturned`, 19

`workflow_contributors` (*snakeface.apps.users.models.User* attribute), 26

`workflow_id` (*snakeface.apps.main.models.WorkflowStatus* attribute), 22

`workflow_owners` (*snakeface.apps.users.models.User* attribute), 26

`WorkflowConsumer` (*class in snakeface.apps.main.consumers*), 18

`WorkflowForm` (*class in snakeface.apps.main.forms*), 18

`WorkflowForm.Meta` (*class in snakeface.apps.main.forms*), 18

## Y

`YELLOW` (*snakeface.logger.ColorizingStreamHandler* attribute), 27