

---

# **kliko Documentation**

*Release dev*

**Gijs Molenaar**

April 11, 2016



<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	installation . . . . .	3
1.2	Why Kliko? . . . . .	3
1.3	Getting started . . . . .	4
1.4	Contributing . . . . .	5
1.5	Testing . . . . .	5
<b>2</b>	<b>Terminology</b>	<b>7</b>
2.1	Kliko . . . . .	7
2.2	Kliko image . . . . .	7
2.3	Kliko container . . . . .	7
2.4	Kliko runner . . . . .	7
2.5	The kliko.yml file . . . . .	7
2.6	The parameters.json file . . . . .	7
<b>3</b>	<b>The specification</b>	<b>9</b>
3.1	The /kliko.yml file . . . . .	9
3.2	An example kliko.yml file . . . . .	10
<b>4</b>	<b>Command Line Utilities</b>	<b>13</b>
4.1	kliko-run . . . . .	13
4.2	kliko-validate . . . . .	13
<b>5</b>	<b>API</b>	<b>15</b>
5.1	Validation . . . . .	15
5.2	Command line interface generation . . . . .	16
5.3	Docker . . . . .	16
5.4	Django . . . . .	17
<b>6</b>	<b>Indices and tables</b>	<b>19</b>
	<b>Python Module Index</b>	<b>21</b>



Contents:



---

## Introduction

---

KLIKO is a specification, validator and parser for the Scientific Compute Container specification. KLIKO is written in Python.

This documentation is intended for the developer who wants to package up a piece of software into a Kliko container.

### 1.1 installation

Development of Kliko is done on [github](#). You can install Kliko inside a docker container or just on your system:

```
$ python setup.py install
```

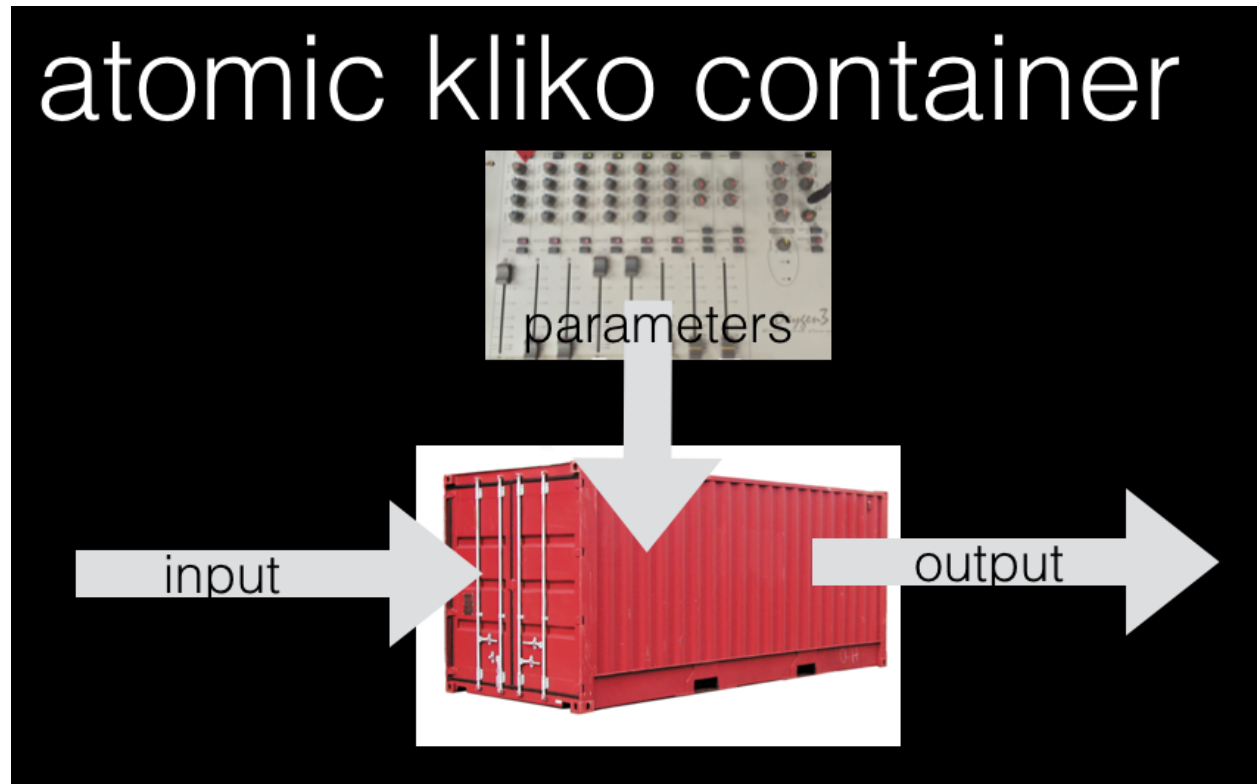
or from pypi:

```
$ pip install kliko
```

### 1.2 Why Kliko?

Kliko was born out of our needs to have a more formal and uniform way of scheduling batch compute tasks on arbitrary public and private cloud platforms. Docker is perfect for encapsulating and distributing software, but the input output flow is not defined. Kliko is an attempt to create a standard way to define compute input, output and parameters.

Kliko assumes your problem looks like this:



Kliko containers can be chained up in a sequence, for example using [NextFlow](#). An other use case is to simplify the parameterized scheduling of compute tasks using [RODRIGUES](#).

## 1.3 Getting started

### 1.3.1 Creating a Kliko container

- Create a Docker container from your application
- Create a script `/kliko` in the container that can parse and use a `/parameters.json` file.
- Add a `kliko.yml` file to the root of the container which defines the valid fields in the parameters file.
- You can validate your kliko file with the `kliko-validate` script installed by the kliko Python library.

### 1.3.2 Running a kliko container

You can run a kliko container in various ways. The most simple way is to use the `kliko-run` script which is installed on your system when you install Kliko. Use `kliko-run <image-name> --help` to see a list of accepted parameters.

If you already have a parameters file you can also run the container manually:

```
$ docker run -v parameters.json:/parameters.json:ro -v input:/input:ro -v output:/output:rw <image-name>
```

Finally you can also run kliko images and visualise results using [RODRIGUES](#), a web based kliko runner.



## 1.4 Contributing

Contributions are more than welcome! If you experience any problems let us know in the bug tracker. We accept patches in the form of github pull requests. Please make sure your code works with python 2 and python3, and is pep8 compatible. Also make sure the test suit actually passes all tests. We use docker in some of the tests so you need to have that installed and configured.

## 1.5 Testing

Note that before you run the test suite you have to create a `klikotest` docker image by running `make` in the `examples` folder.



---

## Terminology

---

### 2.1 Kliko

A specification which defines constraints on a docker container to aid in the scheduling of scientific compute tasks. It is also a Python library that can be used to check if a container confirms the specification.

### 2.2 Kliko image

A Docker image conforming to the kliko specification. An image is an ordered collection of root filesystem changes and the corresponding execution parameters for use within a container runtime. Images are read-only.

### 2.3 Kliko container

A container is an active (or inactive if exited) stateful instantiation of an image.

Read more about Docker terminology in the [Docker glossary](#).

### 2.4 Kliko runner

Something that can run a kliko image. For example the `kliko-run` command line tool, or RODRIGUES.

### 2.5 The `kliko.yml` file

A yaml formatted file conforming to the Kliko specification that defines the parameters a Kliko container is expecting. This is the file you want to create and add to your dockdr image if you want to create a Kliko container.

### 2.6 The `parameters.json` file

A json encoded structure that contains all the parameter values for your compute task. This file is presented to your container at runtime by the container runner, for example RODRIGUES or Nextflow. The valid fields are defined by the Kliko image container and are defined in the `kliko.yml` file.



---

## The specification

---

- Kliko is based on standard docker containers
- A Kliko container should have a `/kliko.yml` file which defines the accepted parameters.
- A Kliko container should have a runnable binary or script named `/kliko`. This will be the entrypoint for the Kliko runner.
- Logging should be written to `STDOUT` and `STDERR`.
- We define two types of compute containers, split IO and joined IO containers. For split IO Input files will be mounted read only into `/input`. Output file should be written to `/output`, which will be mounted by the host. For joined IO containers input & output is the `/work` folder which will be mounted RW.
- Parameters for the computation will be given when the container is run in the form of a file in json format called `/parameters.json`
- Fields with type `file` will enable supply of custom input files. these will be put in the `/input` folder.

### 3.1 The `/kliko.yml` file

The kliko file should be in YAML format and has these required fields:

#### 3.1.1 `schema_version`

The version of the kliko specification. note that this is independent of the versioning of the Kliko library.

#### 3.1.2 `name`

Name of the kliko image. For example `radioastro/simulator`.

#### 3.1.3 `description`

A more detailed description of the image.

#### 3.1.4 `author`

Who made the container.

### 3.1.5 email

email adres of the author.

### 3.1.6 url

Where to find the specific kliko project on the web.

### 3.1.7 io

Which IO mode to use, could be `join` or `split`. For `split` IO Input files will be mounted read only into `/input`. Output file should be written to `/output`, which will be mounted by the host. For joined IO containers input & output is the `/work` folder which will be mounted RW.

### 3.1.8 Sections

The parameters are grouped in sections. Sections are just lists of fields.

### 3.1.9 fields

A section consists of a list of fields.

### 3.1.10 field

each field has 2 obligatory keys, a `name` and a `type`. `Name` is a short reference to the field which needs to be unique. This will be the name for internal reference. The `type` defines the `type` of the field and can be one of `choice`, `char`, `float`, `file`, `bool` or `int`.

**Optional keys are:**

- **initial**: supply a initial (default) value for a field
- **max\_length**: define a maximum length in case of string type
- **choices**: define a list of choices in case of a choice field. The choices should be a mapping
- **label**: The label used for representing the field to the end user. If no label is given the name of the field is used.
- **required**: Indicates if the field is required or optional
- **help\_text**: An optional help text that is presented to the end user next to the field.

## 3.2 An example kliko.yml file

Below is an example kliko file.

```
1 schema_version: 2
2 name: kliko test image
3 description: for testing purposes only
4 container: radioastro/klikotest
5 author: Gijs Molenaar
```

```

6 email: gijsmolenaar@gmail.com
7 url: http://github.com/gijzelaerr/kliko
8 io: split
9
10 sections:
11   -
12     name: section1
13     description: The first section
14     fields:
15       -
16         name: choice
17         label: choice field
18         type: choice
19         initial: second
20         required: True
21         choices:
22           first: option 1
23           second: option 2
24       -
25         name: char
26         label: char field
27         help_text: maximum of 10 chars
28         type: char
29         max_length: 10
30         initial: empty
31         required: True
32       -
33         name: float
34         label: float field
35         type: float
36         initial: 0.0
37         required: False
38     -
39     name: section2
40     description: The final section
41     fields:
42       -
43         name: file
44         label: file field
45         help_text: this file will be put in /input in case of split io, /work in case of join io
46         type: file
47         required: True
48       -
49         name: int
50         label: int field
51         type: int
52         required: True

```

Loading a Kliko container with the previous kliko file is loaded up in RODRIGUES will result in the form below:

RODRIGUES - watching yo... x Gijs Molenaar

127.0.0.1:8000/scheduler/job/create/6/

R.O.D.R.I.G.U.E.S. List jobs Create job admin logged in as gijs Log out

## Create new simulation

**The first section**

choice field

char field   
maximum of 10 chars

float field

**The final section**

file  No file chosen

field   
this file will be put in /input in case of split io,  
/work in case of join io

int field

**Job Parameters**

Job description

Processing this form will result in the following parameters.json file which is presented to the Kliko container on runtime:

```
1 {"int": 10, "file": "some-file", "char": "gijs", "float": 0.0, "choice": "first"}
```



---

## Command Line Utilities

---

### 4.1 `kliko-run`

Use this to run the container. Use `kliko-run <image-name> --help` to see a list of accepted `kliko` parameters, which are `kliko` container specific.

### 4.2 `kliko-validate`

Use this script to check if `kliko` container is valid.



Kliko contains various helper functions to validate Kliko files, parameter files based on a kliko definition, generate command line interfaces and django forms from Kliko definitions.

## 5.1 Validation

Kliko and parameter validation related functions.

`kliko.validate.convert_to_parameters_schema` (*kliko*)

Convert a kliko schema into a validator for the parameters generated with a kliko schema.

**Parameters** `kliko` (*str*) – a kliko definition

**Returns** A structure for a pykwalify validator

`kliko.validate.validate` (*kliko\_file*='/*kliko.yml*', *paramaters\_file*='/*parameters.json*')

Validate the kliko and paramaters file and parse the parameters file. Should be run inside the Kliko container.

**Parameters**

- `kliko_file` (*str*) – Path to a kliko file
- `paramaters_file` (*str*) – path to a parameters file

**Returns** The validated and parsed paramaters file

`kliko.validate.validate_kliko` (*kliko*, *version*=2)

validate a kliko yaml string

**Parameters** `kliko` – a parsed kliko object

**Returns** a (nested) kliko structure

**Return type** dict

**Raises** an exception if the string can't be parsed or is not in the following the Kliko schema

`kliko.validate.validate_parameters` (*parameters*, *kliko*)

validate a set of parameters given a kliko definition

**Parameters**

- `parameters` (*dict*) – A structure that should follow the given kliko structure
- `kliko` (*dict*) – A nested dict which defines the valid parameters in Kliko format

**Returns** the parsed parameters

**Return type** str

**Raises** an exception if the string can't be parsed or is not in the defining valid parameters

## 5.2 Command line interface generation

Command line utilities for Kliko

`kliko.cli.directory_exists` (*path*)  
check if a directory exists

`kliko.cli.file_exists` (*path*)  
check if a file exists

`kliko.cli.first_parser` (*argv*)  
This is only used when script is invoked with 0 or 1 args (should be kliko image name).

`kliko.cli.generate_kliko_cli_parser` (*kliko\_data*, *parent\_parser=None*)  
Generate a command line parser from a Kliko structure.

**Parameters** `kliko_data` (*dict*) – A nested kliko structure

**Returns** a configured argument parser

**Return type** `argparse.ArgumentParser`

`kliko.cli.kliko_runner` (*argv*)

`kliko.cli.prepare_io` (*parameters*, *target=None*)

**Parameters**

- **parameters** – A dict containing the parameters
- **target** – output or work dir for kliko

**Returns** (path to parameters file, path to target folder, path to input folder)

**Return type** tuple

`kliko.cli.second_parser` (*argv*, *kliko\_data*)  
Used when kliko image is known, so we can extract the parameters.

## 5.3 Docker

Helper functions for using Kliko in combination with Docker

`kliko.docker.extract_params` (*docker\_client*, *image\_name*)

**Parameters**

- **docker\_client** (*docker.docker.Client*) – a docker client object
- **image\_name** (*str*) – name of the image to use for kliko.yml extraction

**Returns** content of the param schema

**Return type** str

## 5.4 Django

Helper functions for using Kliko in combinaton with Django

`kliko.django_form.generate_form` (*parsed*)  
Generate a django form from a parsed kliko object

**Parameters** `params` – A parsed kliko file.

**Returns** `form_utils.forms.BetterForm`



---

## Indices and tables

---

- `genindex`
- `modindex`
- `search`





## k

`kliko.cli`, 16

`kliko.django_form`, 17

`kliko.docker`, 16

`kliko.validate`, 15



## C

`convert_to_parameters_schema()` (in module `kliko.validate`), 15

## D

`directory_exists()` (in module `kliko.cli`), 16

## E

`extract_params()` (in module `kliko.docker`), 16

## F

`file_exists()` (in module `kliko.cli`), 16

`first_parser()` (in module `kliko.cli`), 16

## G

`generate_form()` (in module `kliko.django_form`), 17

`generate_kliko_cli_parser()` (in module `kliko.cli`), 16

## K

`kliko.cli` (module), 16

`kliko.django_form` (module), 17

`kliko.docker` (module), 16

`kliko.validate` (module), 15

`kliko_runner()` (in module `kliko.cli`), 16

## P

`prepare_io()` (in module `kliko.cli`), 16

## S

`second_parser()` (in module `kliko.cli`), 16

## V

`validate()` (in module `kliko.validate`), 15

`validate_kliko()` (in module `kliko.validate`), 15

`validate_parameters()` (in module `kliko.validate`), 15