renga-deployer Documentation

Release 0.1.0.dev20170000

Swiss Data Science Center

Contents

1	Local	
2	Docker	5
3	Orchestration with Postgres and load-balancer	7
4	Platform integration	9
	User's Guide 5.1 Installation	11 13
	API Reference 6.1 API Docs	15 15
	Additional Notes 7.1 Contributing 7.2 Changes 7.3 License 7.4 Authors	23 23
Pyt	thon Module Index	25

Renga Deployer Service.

Contents 1

2 Contents

Local

```
$ export FLASK_APP=renga_deployer/app.py
$ flask run
```

The first time you run the app locally, you may need to build the database tables:

```
$ flask shell
>>> from renga_deployer.app import db
>>> db.create_all()
```

4 Chapter 1. Local

Docker

```
$ docker build --tag renga-deployer:latest .
$ docker run -p 5000:5000 -v /var/run/docker.sock:/var/run/docker.sock renga-
deployer:latest
```

For development, mount the code directly and enable flask debug mode:

```
$ docker run -p 5000:5000 \
    -e FLASK_DEBUG=1 \
    -v `pwd`:/code \
    -v /var/run/docker.sock:/var/run/docker.sock \
    renga-deployer:latest
```

You can test the API by pointing your browser to http://localhost:5000/v1/ui

6 Chapter 2. Docker

Orchestration with Postgres and load-balancer

The packages includes two simple orchestration setup files: docker-compose.yml and docker-compose.full.yml.

The first includes only the deployer container linked to a postgres database. To use the deployer with postgres:

```
$ docker-compose up
```

As before, the service is available on port 5000.

The second can be used as a template to define your own production deployment environment including the Traefik load balancer and (self-signed) SSL certificates.

```
$ docker-compose -f docker-compose.full.yml up
```

The service is available on https://localhost/api/deployer. You can access the traefik dashboard on http://localhost:8080/.



Platform integration

The deployer can optionally integrate with other Renga Platform services. To enable integration, set the appropriate environment variables in the form of <code>SERVICE_NAME_URL></code> to point to the api URL. For example, setting <code>KNOWLEDGE_GRAPH_URL</code> will ensure that deployment contexts and executions are automatically added to the knowledge graph. Note that to use the resource manager, you will need to additionally set the <code>DEPLOYER_JWT_KEY</code>.

renga-deployer Documentation, Release 0.1.0.dev20170000			

User's Guide

This part of the documentation will show you how to get started in using Renga-Deployer.

5.1 Installation

Renga-Deployer is on PyPI so all you need is:

```
$ pip install renga-deployer
```

5.2 Configuration

Configuration options can be also provided as environmental variables.

```
renga_deployer.config.DEPLOYER_APP_NAME = 'demo-client'
    Application name.
```

renga_deployer.config.DEPLOYER_AUTHORIZATION_URL = 'http://localhost:8080/auth/realms/Renga_OpenID-Connect authorization endpoint.

```
renga_deployer.config.DEPLOYER_BASE_PATH = '/v1'
Base path for the API.
```

renga_deployer.config.**DEPLOYER_BASE_TEMPLATE** = 'renga_deployer/base.html'
Default base template for the demo page.

```
renga_deployer.config.DEPLOYER_CLIENT_ID = 'demo-client'
Client identifier used for OIDC authentication.
```

renga_deployer.config.DEPLOYER_CLIENT_SECRET = None
 Client credentials used for OIDC authentication.

Should Flask-SQLAlchemy will track modifications of objects.

```
renga deployer.config.DEPLOYER DEFAULT BASE URL = '/'
    The default base url that is passed inside the deployed container via the DEPLOYER_BASE_URL environment
    variable. If an ingress is enabled in a K8S deployer, this variable is changed dynamically per execution.
renga_deployer.config.DEPLOYER_DEFAULT_VALUE = 'foobar'
    Default value for the application.
renga deployer.config.DEPLOYER DOCKER CONTAINER IP = None
    Specific IP for docker-deployed containers.
renga_deployer.config.DEPLOYER_JWT_ISSUER = 'http://localhost:8080/auth/realms/Renga'
    JWT issuer used for token verification.
renga_deployer.config.DEPLOYER_JWT_KEY = None
    Public key used to verify JWT tokens.
renga_deployer.config.DEPLOYER_K8S_CONTAINER_IP = None
    Specific IP for kubernetes-deployed containers.
renga_deployer.config.DEPLOYER_K8S_INGRESS = None
    The class of the ingress controller, for example 'nginx', to be used for the endpoints. Set to None (default) or
    False to disable ingress
renga_deployer.config.DEPLOYER_SWAGGER_UI = False
    Enable Swagger UI.
renga_deployer.config.DEPLOYER_TOKEN_SCOPE_KEY = None
    Key inside JWT containing scopes.
    Use 'https://rm.datascience.ch/scope' in combination with resource manager.
renga_deployer.config.DEPLOYER_TOKEN_URL = 'http://localhost:8080/auth/realms/Renga/protoce
    OpenID-Connect token endpoint.
renga_deployer.config.DEPLOYER_URL = 'http://localhost:5000'
    Base URL for the service.
renga_deployer.config.KNOWLEDGE_GRAPH_URL = None
    Push contexts and executions to the KnowledgeGraph.
renga_deployer.confiq.RENGA_AUTHORIZATION_CLIENT_ID = None
    Client id for fetching the service access token.
renga deployer.config.RENGA AUTHORIZATION CLIENT SECRET = None
    Client secret for fetching the service access token.
renga_deployer.config.RENGA_ENDPOINT = 'http://localhost'
    URL for other platform services.
renga_deployer.config.RENGA_LOGGING_CONFIG = None
    Logging configuration file path.
renga_deployer.config.RESOURCE_MANAGER_URL = None
    Obtain and validate ResourceManager authorization tokens.
renga_deployer.config.SENTRY_DSN = None
    The default Sentry environment variable key.
renga_deployer.config.SQLALCHEMY_DATABASE_URI = 'sqlite:///deployer.db'
    The URI of the database to be used for preserving internal state.
renga deployer.confiq.SQLALCHEMY TRACK MODIFICATIONS = False
```

```
{\tt renga\_deployer.config.WSGI\_NUM\_PROXIES} \ = \ {\tt None}
```

The number of proxy servers in front of the app.

Disable proxy fixer by setting value evaluating to False.

```
renga_deployer.config.from_env(config)
```

Load configuration options from environment variables.

5.3 Usage

Renga Deployer Service.

5.4 Example application

First install Renga-Deployer, setup the application and load fixture data by running:

```
$ pip install -e .[all]
$ cd examples
$ ./app-setup.sh
$ ./app-fixtures.sh
```

Next, start the development server:

```
$ export FLASK_APP=app.py FLASK_DEBUG=1
$ flask run
```

and open the example application in your browser:

```
$ open http://127.0.0.1:5000/
```

To reset the example application run:

```
$ ./app-teardown.sh
```

5.3. Usage 13

API Reference

If you are looking for information on a specific function, class or method, this part of the documentation is for you.

6.1 API Docs

6.1.1 Deployer

```
Deployer sub-module.
```

```
class renga_deployer.deployer.Deployer(engines=None, **kwargs)
     Handling the executions of contexts.
     Create a Deployer instance.
          Parameters engines – dict of engine name:uri pairs
     create (spec)
          Create a context with a given specification.
     classmethod from_env(prefix='DEPLOYER_')
          Create a Deployer from environment variables.
     get_host_ports(execution)
          Fetch hostname and ports for the running execution.
     get_logs (execution)
          Ask engine to extract logs.
     launch (context=None, engine=None, **kwargs)
          Create new execution for a given context.
     stop (execution, remove=False)
          Stop a running execution, optionally removing it from engine.
```

6.1.2 Engines

```
Engine sub-module.
```

class renga_deployer.engines.DockerEngine

Class for deploying contexts on docker.

Initialize the docker engine.

class EXECUTION_STATE_MAPPING

State mappings for the Docker engine.

client

Create a docker client from local environment.

get_execution_environment (execution) → dict

Retrieve the environment specified for an execution container.

get_host_ports (execution)

Return host ip and port bindings for the running execution.

get_logs (execution)

Extract logs for a container.

get_state (execution)

Return the status of an execution.

launch (execution, **kwargs)

Launch a docker container with the context image.

logger

Create a logger instance.

stop (execution, remove=False)

Stop a running container, optionally removing it.

class renga_deployer.engines.Engine

Base engine class.

get_execution_environment(execution) → dict

Retrieve the environment specified for an execution container.

get_host_port (execution)

Retrieve the host/port where the application can be reached.

get_logs (execution)

Extract logs for a container.

get_state (execution)

Check the state of an execution.

launch (context, **kwargs)

Create new execution environment for a given context.

stop (execution, remove=False)

Stop an execution.

class renga_deployer.engines.K8SEngine (config=None, timeout=10)

Class for deploying contexts on Kubernetes.

Create a K8SNode instance.

class EXECUTION_STATE_MAPPING

State mappings for the K8S engine.

```
get_execution_environment (execution) → dict
    Retrieve the environment specified for an execution container.

get_host_ports (execution)
    Return host ip and port bindings for the running execution.

get_logs (execution, timeout=None, **kwargs)
    Extract logs for the Job from the Pod.

get_state (execution)
    Get status of a running job.

launch (execution, engine=None, **kwargs)
    Launch a Kubernetes Job with the context spec.

logger
    Create a logger instance.

stop (execution, remove=False)
    Stop a running job.
```

6.1.3 Extension

Renga Deployer Service.

```
class renga_deployer.ext.RengaDeployer(app=None)
    Renga-Deployer extension.
    Extension initialization.

deployer
        Returns a local app Deployer.

init_app(app)
        Flask application initialization.

init_config(app)
        Initialize configuration.
```

6.1.4 Models

Models sub-module.

```
class renga_deployer.models.Context(**kwargs)
```

Execution context.

Additionally it contains two columns created and updated with automatically managed timestamps.

A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in kwargs.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

```
classmethod create(spec=None)
```

Create a new context.

creator

Creator of the context.

6.1. API Docs 17

```
id
          Context identifier.
     jwt
          JWT with which the context has been created.
     spec
          Context specification.
class renga_deployer.models.Execution(**kwargs)
     Represent an execution of a context.
     Additionally it contains two columns created and updated with automatically managed timestamps.
     A simple constructor that allows initialization from kwargs.
     Sets attributes on the constructed instance using the names and values in kwarqs.
     Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any
     mapped columns or relationships.
     check state(states, engine)
          Check whether the execution is in one of the specified states.
     context id
          Context identifier from which the execution started.
     engine
          Engine name.
     engine id
          Internal identifier returned by an engine.
     classmethod from_context(context, **kwargs)
          Create a new execution for a given context.
     id
          Execution identifier.
     jwt
          JWT with which the execution has been created.
     namespace
          Namespace name.
class renga_deployer.models.ExecutionStates
     Valid execution states.
renga deployer.models.db = <SQLAlchemy engine=None>
     Core database object.
renga_deployer.models.load_jwt()
     Load JWT from a context.
6.1.5 Utils
Utility functions.
renga_deployer.utils.decode_bytes(func)
     Function wrapper that always returns string.
renga_deployer.utils.dict_from_labels(labels, separator='=')
```

Create a multidict from label string.

```
renga_deployer.utils.join_url(*args)
    Join together url strings.

renga_deployer.utils.resource_available(func)
    Function wrapper to catch that something is not available.

Example:
    while not resource_available(get_logs()): # this loop continues until the logs are available pass logs = get_logs()

renga_deployer.utils.validate_uuid(s, version=4)
    Check that a string is a valid UUID.

renga_deployer.utils.validate_uuid_args(*names)
    Check that input arguments are valid UUIDs.
```

6.1.6 Views

Renga Deployer Service.

```
renga_deployer.views.index()
    Basic view.
```

6.1. API Docs 19

Additional Notes

Notes on how to contribute, legal information and changes are here for the interested.

7.1 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

7.1.1 Types of Contributions

Report Bugs

Report bugs at https://github.com/SwissDataScienceCenter/renga-deployer/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

Write Documentation

Renga-Deployer could always use more documentation, whether as part of the official Renga-Deployer docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/SwissDataScienceCenter/renga-deployer/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome:)

7.1.2 Get Started!

Ready to contribute? Here's how to set up renga-deployer for local development.

- 1. Fork the SwissDataScienceCenter/renga-deployer repo on GitHub.
- 2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/renga-deployer.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv renga-deployer
$ cd renga-deployer/
$ pip install -e .[all]
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

5. When you're done making changes, check that your changes pass tests:

```
$ ./run-tests.sh
```

The tests will provide you with test coverage and also check PEP8 (code style), PEP257 (documentation), flake8 as well as build the Sphinx documentation and run doctests.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -s
    -m "component: title without verbs"
    -m "* NEW Adds your new feature."
    -m "* FIX Fixes an existing issue."
    -m "* BETTER Improves and existing feature."
    -m "* Changes something that should not be visible in release notes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

7.1.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

- 1. The pull request should include tests and must not decrease test coverage.
- 2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring.
- 3. The pull request should work for Python 2.7, 3.3, 3.4 and 3.5. Check https://travis-ci.org/ SwissDataScienceCenter/renga-deployer/pull_requests and make sure that the tests pass for all supported Python versions.

7.2 Changes

Version 0.1.0 (released TBD)

• Initial public release.

7.3 License

Copyright 2017 - Swiss Data Science Center (SDSC) A partnership between École Polytechnique Fédérale de Lausanne (EPFL) and Eidgenössische Technische Hochschule Zürich (ETHZ).

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

7.4 Authors

Renga Deployer Service.

• Swiss Data Science Center <contact@datascience.ch>

7.2. Changes 23

renga-deployer Documentation, Release 0.1.0.dev20170000)

Python Module Index

r

```
renga_deployer, 13
renga_deployer.config, 11
renga_deployer.deployer, 15
renga_deployer.engines, 16
renga_deployer.ext, 17
renga_deployer.models, 17
renga_deployer.utils, 18
renga_deployer.views, 19
```

26 Python Module Index

Index

C	DEPLOYER_JWT_KEY (in module
check_state() (renga_deployer.models.Execution method), 18	renga_deployer.config), 12 DEPLOYER_K8S_CONTAINER_IP (in module
client (renga_deployer.engines.DockerEngine attribute), 16	renga_deployer.config), 12 DEPLOYER_K8S_INGRESS (in module
Context (class in renga_deployer.models), 17 context_id (renga_deployer.models.Execution attribute), 18	renga_deployer.config), 12 DEPLOYER_SWAGGER_UI (in module renga_deployer.config), 12
create() (renga_deployer.deployer.Deployer method), 15	DEPLOYER_TOKEN_SCOPE_KEY (in module
create() (renga_deployer.models.Context class method), 17	renga_deployer.config), 12 DEPLOYER_TOKEN_URL (in module
creator (renga_deployer.models.Context attribute), 17	renga_deployer.config), 12 DEPLOYER_URL (in module renga_deployer.config),
D	12
db (in module renga_deployer.models), 18	dict_from_labels() (in module renga_deployer.utils), 18 DockerEngine (class in renga_deployer.engines), 16
decode_bytes() (in module renga_deployer.utils), 18	DockerEngine.EXECUTION_STATE_MAPPING (class
Deployer (class in renga_deployer.deployer), 15	in renga_deployer.engines), 16
deployer (renga_deployer.ext.RengaDeployer attribute),	
DEPLOYER_APP_NAME (in module	E
renga_deployer.config), 11	Engine (class in renga_deployer.engines), 16
DEPLOYER_AUTHORIZATION_URL (in module renga_deployer.config), 11	engine (renga_deployer.models.Execution attribute), 18 engine_id (renga_deployer.models.Execution attribute),
DEPLOYER BASE PATH (in module	18
renga_deployer.config), 11	Execution (class in renga_deployer.models), 18
DEPLOYER_BASE_TEMPLATE (in module	ExecutionStates (class in renga_deployer.models), 18
renga_deployer.config), 11	F
DEPLOYER_CLIENT_ID (in module	•
renga_deployer.config), 11 DEPLOYER_CLIENT_SECRET (in module	from_context() (renga_deployer.models.Execution class method), 18
renga_deployer.config), 11	from_env() (in module renga_deployer.config), 13
DEPLOYER_DEFAULT_BASE_URL (in module	from_env() (renga_deployer.deployer_Deployer class
renga_deployer.config), 11	method), 15
DEPLOYER_DEFAULT_VALUE (in module	G
renga_deployer.config), 12 DEPLOYER_DOCKER_CONTAINER_IP (in module	<u> </u>
renga_deployer.config), 12	get_execution_environment() (renga_deployer.engines.DockerEngine
DEPLOYER_JWT_ISSUER (in module	method), 16
renga_deployer.config), 12	<i>"</i>

get_execution_environment()	launch() (renga_deployer.engines.K8SEngine method),
(renga_deployer.engines.Engine method),	17
16	load_jwt() (in module renga_deployer.models), 18
get_execution_environment() (renga_deployer.engines.K8SEngine method),	logger (renga_deployer.engines.DockerEngine attribute),
16	logger (renga_deployer.engines.K8SEngine attribute), 17
get_host_port() (renga_deployer.engines.Engine	
method), 16	N
get_host_ports() (renga_deployer.deployer.Deployer method), 15	namespace (renga_deployer.models.Execution attribute),
get_host_ports() (renga_deployer.engines.DockerEngine method), 16	R
get_host_ports() (renga_deployer.engines.K8SEngine method), 17	RENGA_AUTHORIZATION_CLIENT_ID (in module renga_deployer.config), 12
get_logs() (renga_deployer.deployer.Deployer method), 15	RENGA_AUTHORIZATION_CLIENT_SECRET (in module renga_deployer.config), 12
get_logs() (renga_deployer.engines.DockerEngine	renga_deployer (module), 13
method), 16	renga_deployer.config (module), 11
get_logs() (renga_deployer.engines.Engine method), 16 get_logs() (renga_deployer.engines.K8SEngine method),	renga_deployer.deployer (module), 15
get_logs() (renga_ucproyer.engmes.RosEngme method),	renga_deployer.engines (module), 16 renga_deployer.ext (module), 17
get_state() (renga_deployer.engines.DockerEngine	renga_deployer.models (module), 17
method), 16	renga_deployer.utils (module), 18
get_state() (renga_deployer.engines.Engine method), 16	renga_deployer.views (module), 19
get_state() (renga_deployer.engines.K8SEngine method), 17	RENGA_ENDPOINT (in module
17	renga_deployer.config), 12
	RENGA_LOGGING_CONFIG (in module renga_deployer.config), 12
id (renga_deployer.models.Context attribute), 17	RengaDeployer (class in renga_deployer.ext), 17
id (renga_deployer.models.Execution attribute), 18	resource_available() (in module renga_deployer.utils), 19
index() (in module renga_deployer.views), 19	RESOURCE_MANAGER_URL (in module
<pre>init_app() (renga_deployer.ext.RengaDeployer method), 17</pre>	renga_deployer.config), 12
init_config() (renga_deployer.ext.RengaDeployer	S
method), 17	SENTRY_DSN (in module renga_deployer.config), 12
J	spec (renga_deployer.models.Context attribute), 18
	SQLALCHEMY_DATABASE_URI (in module
join_url() (in module renga_deployer.utils), 18 jwt (renga_deployer.models.Context attribute), 18	renga_deployer.config), 12
jwt (renga_deployer.models.Execution attribute), 18	SQLALCHEMY_TRACK_MODIFICATIONS (in mod-
	ule renga_deployer.config), 12 stop() (renga_deployer.deployer.Deployer method), 15
K	stop() (renga_deployer.engines.DockerEngine method),
K8SEngine (class in renga_deployer.engines), 16	16
K8SEngine.EXECUTION_STATE_MAPPING (class in	stop() (renga_deployer.engines.Engine method), 16
renga_deployer.engines), 16 KNOWLEDGE_GRAPH_URL (in module	stop() (renga_deployer.engines.K8SEngine method), 17
renga_deployer.config), 12	V
	validate_uuid() (in module renga_deployer.utils), 19
L	validate_uuid_args() (in module renga_deployer.utils), 19
launch() (renga_deployer.deployer.Deployer method), 15	W
launch() (renga_deployer.engines.DockerEngine method), 16	
launch() (renga_deployer.engines.Engine method), 16	WSGI_NUM_PROXIES (in module renga_deployer.config), 12

28 Index