developer.skatelescope.org Documentation

Release 0.1.0-beta

Marco Bartolini

Nov 25, 2019

Home

Welcome to the Square Kilometre Array software documentation portal. Whether you are a developer involved in SKA or you are simply one of our many users, all of our software processes and projects are documented in this portal.

CHAPTER 1

Scope

This documentation applies to the bridging phase of the SKA project, further updates and scope changes will be published on this website. Part of the bridging phase goals will be to consolidate and enrich this portal with more detailed information. It is thus anticipated that in this phase the change rate of the documentation will be very frequent.

CHAPTER 2

SKA Code of Conduct

SKA Organisation (SKAO) is committed to the highest standards of business ethics and as such expects everyone involved in SKAO-related business to uphold the standards and expected professional behavior set out in SKA Code of Ethics page .

The code of ethics applies to every SKA collaborators and it is the reference guide defining the culture of this online community of contributors.

- Download the SKA Code of Ethics

2.1 Howto join the SKA developer community

Todo:

- to be defined as soon as we have onboarding procedures for new members, and specific training events.

2.2 Agile teams and responsibilities

SKA software development is organized in agile teams.

2.2.1 Development team

See https://www.scaledagileframework.com/dev-team/

Todo:

- should we expand this section? The whole portal is dedicated to describe DEV practices and tools . . .

2.2.2 Scrum Master

The Scrum Master of each team is responsible for the process the team follows. A generic description of this role can be found on the SAFe website. The SKA Scrum Masters are also responsible for:

- Meet the team, make sure they know each other and find a nice way to present interests, skills and get to know each other. Lead the team to find a name they like.
- Make sure all team members can access SKA confluence and jira.
- Make sure all team members have access to SKA video conferencing tools.
- Create a team page on the SKA confluence portal describing who belongs to the team and his/her role. This page will serve as an entry point for team related information.
- Use the Team Jira board to plan and report team activity happening in the development sprints.
- Run sprints planning/retrospective/reviews cycles and daily stand-up meetings with the team, making sure the team follows an improvement process.
- Work with the team in order to understand the SKA Definition of Done and development practices.
- Setup and maintain a slack channel for the team according to the slack usage guidelines.
- Setup and maintain a github team including all team members under the SKA organization github account.
- Manage permissions on github repositories the team is working on.
- Maintain consistency between the team composition on the various tools and platforms, and make sure that team members are using those in an appropriate way.
- Take part in the Scrum Of Scrums meeting, coordinating his/her activity with all the SMs participating in the development effort.

2.2.3 Product Owner

See https://www.scaledagileframework.com/product-owner/

Todo:

- Define specifics activities for SKA POs.

2.3 SKA developer community decision making process

Todo:

- this is still TBD with the program management team.

2.4 Recommended readings

This page contains a list of recommended readings that describe the practices adopted by SKA developers.

2.4.1 System Design

- Continuous Delivery describes system design and practices necessary to realize continuos delivery.
- Design Patterns: Elements of Reusable Object-Oriented Software describes the most common design patterns to be found in a software system.

2.4.2 Programming

- Code complete is a practical introduction to software craftmanship.
- The pragmatic programmer is a good introduction to sound programming practices.
- Clean code introduces quality software practices showcasing different examples and good principles from the agile world.
- Extreme programming explained can be extremely useful to teams and developers embracing a more agile way of working for the first time.

2.4.3 Programming Languages

Python

- Python in a Nutshell is a comprehensive Python reference to have on your desk while developing any sort of Python application.
- Python testing with pytest covers pytest framework and related testing best practices in the Python ecosystem.
- Fluent Python is a useful guide to writing effective, idiomatic Python code. This book is recommended reading for intermediate Python developers and those coming from other languages, showing how Python features and idioms should be used most effectively.

C++

- Effective C++ contains useful recipes for sound implementations of common C++ patterns.
- Effective STL focuses on the correct usage of the standard teamplate library.
- Effective Modern C++ follows from the previous series, expanding on the transition to C++11 and C++14 and their new constructs.

CHAPTER 3

SKA developer community

SKA software development is managed in an open and transparent way.

- *Howto join the SKA developer community*
- *Agile teams and responsibilities*
- *SKA developer community decision making process*
- *Recommended readings*

Todo:

- SAFe process implementation
- Community forum

3.1 Git

Git is the version control system of choice used by SKA. Describing the basics of how to use git is out of the scope of this developer portal, but it is fundamental that all developers contributing to SKA get familiar with git and how to use it. These online resources are a good starting point:

- Learn git interactively: https://learngitbranching.js.org/
- Official git reference at: https://git-scm.com/docs
- Interactive Git cheatsheet: http://www.ndpsoftware.com/git-cheatsheet.html

3.1.1 Committing code

When working on a development project, it is important to stick to these simple commit rules:

- Commit often.

- Have the **Jira story ID** at the beginning of your commit messages.
- Git logs shall be human readable in sequence, describing the development activity.
- Use imperative forms in the commit message.

3.1.2 Configure git

Set GIT institutional email address

Setup git so that it uses your institutional email account to sign commits, this can be done in your global git configuration:

```
$ git config --global user.email "your@institutional.email"
$ git config --global user.email
your@institutional.email
```

Or you can configure the mail address on a project basis.

```
$ cd your/git/project
$ git config user.email "your@institutional.email"
$ git config user.email
your@institutional.email
```

3.1.3 Branching policy

Albeit the SKA organisation does not want to be prescriptive about git workflows, two concepts are important to the SKA way of using GIT:

1. The master branch of a repository shall always be stable.
2. Branches shall be short lived, merging into master as often as possible.

Stable means that the master branch shall always compile and build correctly, and executing automated tests with success. Every time a master branch results in a condition of instability, reverting to a condition of stability shall have the precedence over any other activity on the repository.

Master based development

We suggest teams to start developing adopting a master-based development approach, where each developer commits code into the master branch at least daily. While this practice may seem counter intuitive, there is good evidence in literature that it leads to a better performing system. Branches are reduced to a minimum in this model, and the discipline of daily commits into master greatly enhances the communication within the team and the modularity of the software system under construction. The workflow follows these steps:

- As a developer starts working on a story, all his commits related to the story shall contain the story Jira ID in the message. i.e. *AT-51 method stubs*
- The developer continues working on his local master branch with multiple commits on the same story.
- Each day the local master pulls the remote and incorporates changes from others.
- The local master is tested successfully.
- The local commits are pushed onto the remote master.
- The CI pipeline is correctly executed on the remote master by the CI server.

Implemented correctly, this practice leads to having an integrated, tested, working system at the end of each development interval, that can be shipped directly from our master branch with the click of a button.

Story based branching

We support adopting a story-based branching model, often referred to as **feature branching**. This workflow effectively leverages **pull requests** enabling code reviews and continuous branch testing, but it is important to stress the importance of having short lived branches. It is easy to abuse this policy and have long living branches resulting in painful merge activities and dead or stale development lines. Bearing in mind that a *story* by definition is some piece of work a developer should conclude in the time of a sprint, the workflow would follow these steps:

- As a developer starts working from master on a new story, she creates a new branch.
- The new branch shall be named as the story, i.e. *story-AT1-26*.

```
$ git branch
* master
$ git checkout -b my-story-id
$ git branch
master
* my-story-id
```

- All the commit messages contributing to the development of the story begin with the story ID, i.e. *AT1-26 basic testing*.
- The developer makes sure that all tests execute correctly on her local story branch.
- When the story is ready for acceptance the developer pushes the story branch upstream.

```
$ git push -u origin my-story-id
```

- A pull request is created on the DVCS server to merge the story branch into the master branch.
- Reviewers interact with comments on the pull request until all conflicts are resolved and reviewers accept the pull request.
- Pull request is merged into Master.
- The CI pipeline is executed successfully on the master branch by the CI server.

Whenever a team deviates from one of the recommended policy, it is important that the team captures its decision and publicly describe its policy, discussing it with the rest of the community.

See a more detailed description of this workflow at https://guides.github.com/introduction/flow/

3.2 Github

3.2.1 Use institutional email

Create a github account using your **institutional email** address at https://github.com/join?source=login . If you already have an account on github, you shall have your institutional email added to your profile: click on your user icon on the top right corner and select *Settings->Emails->Add email address* .

3.2.2 Setup SSH key

Associate your ssh-key to your user at *Settings->SSH and GPG keys* .

3.2.3 Join SKA Organisation

SKA Organisation can be found on github at https://github.com/ska-telescope/ , The scrum master of your team will make sure you can access it.

3.2.4 Desktop client

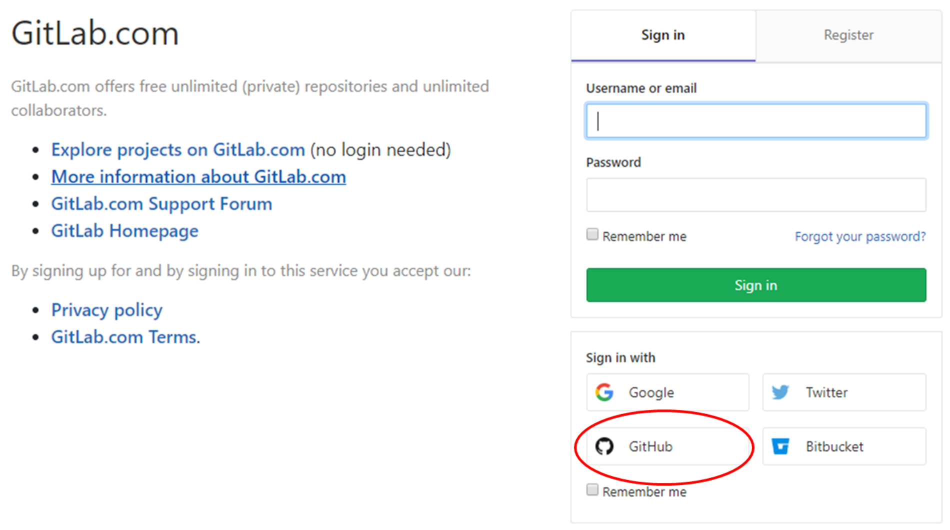
Less experienced developers can use the github desktop client at: https://desktop.github.com/ This definitely lowers the barrier of using git for a number of different users.

3.3 Continuous Integration

3.3.1 Creating a new CICD project

GitLab CI/CD can be used with GitHub or any other Git server. Instead of moving your entire project to GitLab, you can connect your external repository to get the benefits of GitLab CI/CD (documentation pages for github can be found here).

In order to do that, the very first step is to create an account in GitLab. It is recommended to sign_in with the GitHub account. for any information or requests on those aspects, please contact the SCRUM master of your agile team.



GitLab.com

GitLab.com offers free unlimited (private) repositories and unlimited collaborators.

- Explore projects on GitLab.com (no login needed)
- More information about GitLab.com
- GitLab.com Support Forum
- GitLab Homepage

By signing up for and by signing in to this service you accept our:

- Privacy policy
- GitLab.com Terms.

Sign in | Register

Username or email

Password

Remember me | Forgot your password?

Sign in

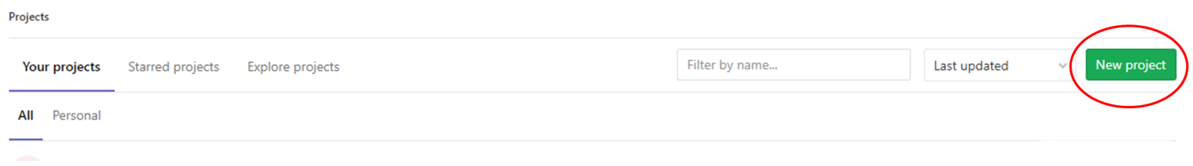
Sign in with

Google | Twitter

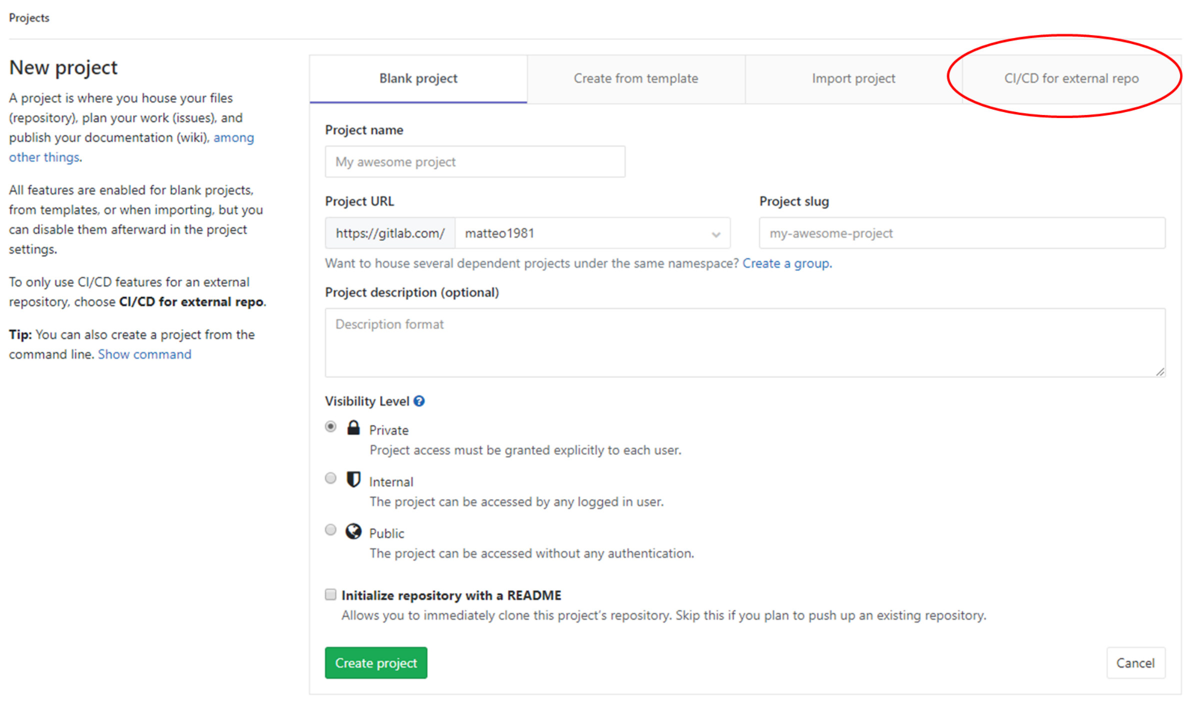
GitHub | Bitbucket

Remember me

Once logged, the home page allows for the creation of a new project (see images below).



The new project page allows to create a project (CI/CD) from an external repository.


Projects

New project

A project is where you house your files (repository), plan your work (issues), and publish your documentation (wiki), among other things.

All features are enabled for blank projects, from templates, or when importing, but you can disable them afterward in the project settings.

To only use CI/CD features for an external repository, choose **CI/CD for external repo**.

Tip: You can also create a project from the command line. Show command

Blank project | Create from template | Import project | CI/CD for external repo

Project name: My awesome project

Project URL: https://gitlab.com/ matteo1981

Project slug: my-awesome-project

Want to house several dependent projects under the same namespace? Create a group.

Project description (optional): Description format

Visibility Level

[✓ Private] Project access must be granted explicitly to each user.

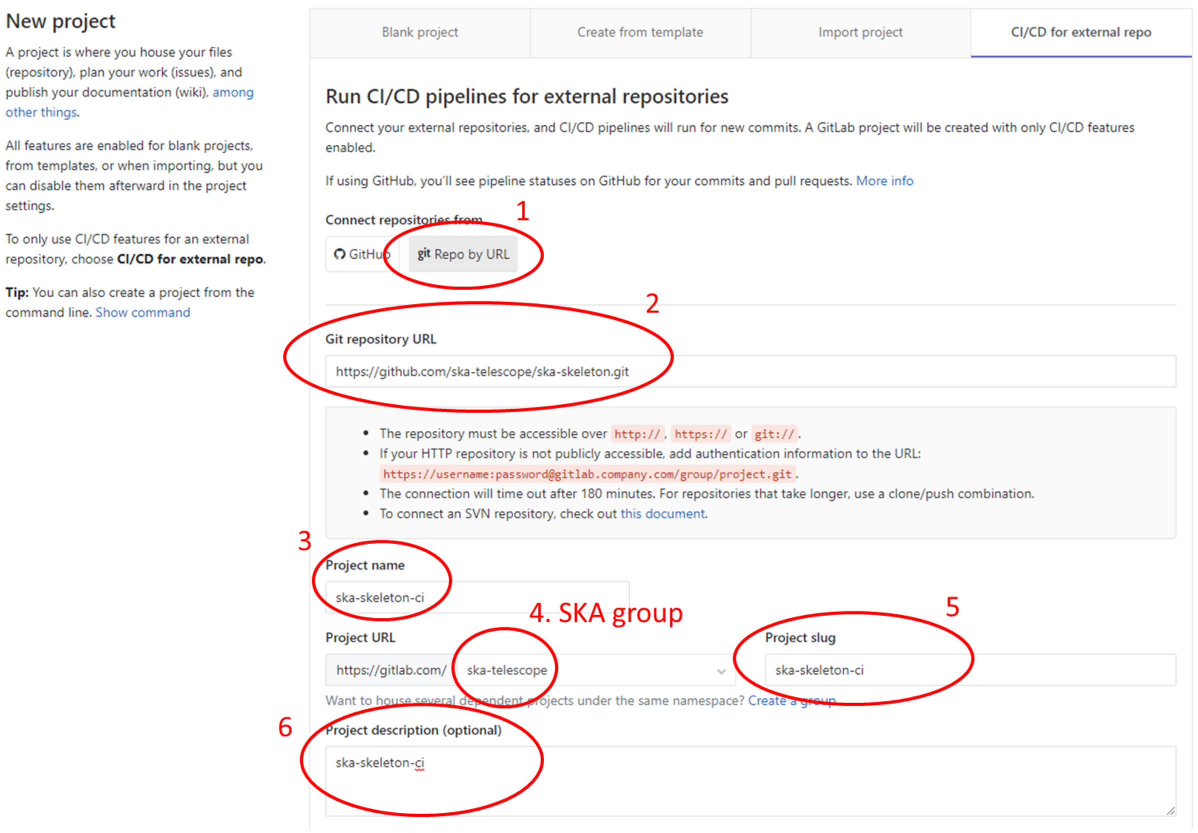
Internal — The project can be accessed by any logged in user.

Public — The project can be accessed without any authentication.

Initialize repository with a README — Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

The step-by-step procedure is the following:

1. Select the option "Repo by URL",
2. Write the repository url (i.e. https://github.com/ska-telescope/ska-skeleton.git),
3. Select a project name (for instance ska-skeleton-ci),
4. Select the correct ska group (i.e. ska-telescope),
5. Select/write a project slug (URL-friendly version of a repository name) and
6. Insert a project description.



New project

A project is where you house your files (repository), plan your work (issues), and publish your documentation (wiki), among other things.

All features are enabled for blank projects, from templates, or when importing, but you can disable them afterward in the project settings.

To only use CI/CD features for an external repository, choose **CI/CD for external repo**.

Tip: You can also create a project from the command line. Show command

Run CI/CD pipelines for external repositories

Connect your external repositories, and CI/CD pipelines will run for new commits. A GitLab project will be created with only CI/CD features enabled.

If using GitHub, you'll see pipeline statuses on GitHub for your commits and pull requests. More info

Connect repositories from

Git repository URL: https://github.com/ska-telescope/ska-skeleton.git

- The repository must be accessible over `http://`, `https://` or `git://`.
- If your HTTP repository is not publicly accessible, add authentication information to the URL: `https://username:password@gitlab.company.com/group/project.git`.
- The connection will time out after 180 minutes. For repositories that take longer, use a clone/push combination.
- To connect an SVN repository, check out this document.

Project name: ska-skeleton-ci

Project URL: https://gitlab.com/ ska-telescope

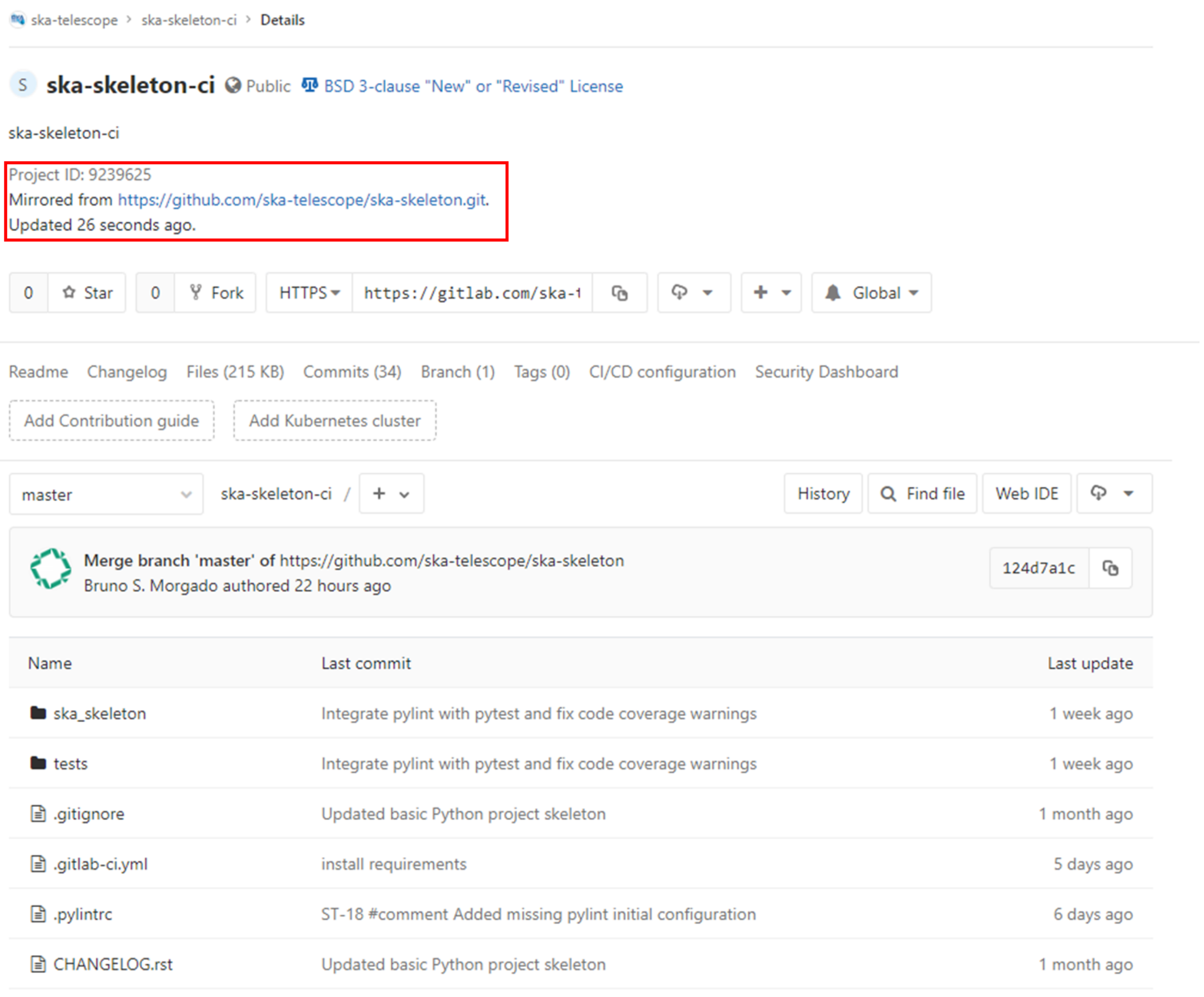
Project slug: ska-skeleton-ci

Want to house several dependent projects under the same namespace? Create a group.

Project description (optional): ska-skeleton-ci

It is also possible to set the visibility of the project (the default is public).

Connecting an external repository will set up repository mirroring (it is indicated in the GitLab project home page) and create a lightweight project where issues, merge requests, wiki, and snippets disabled.


ska-telescope › ska-skeleton-ci › Details
ska-skeleton-ci Public BSD 3-clause "New" or "Revised" License
0 Star 0 Fork HTTPS https://gitlab.com/ska-t Global
Readme Changelog Files (215 KB) Commits (34) Branch (1) Tags (0) CI/CD configuration Security Dashboard
Add Contribution guide Add Kubernetes cluster
master ska-skeleton-ci / + History Find file Web IDE

ska-skeleton-ci

Project ID: 9239625
Mirrored from https://github.com/ska-telescope/ska-skeleton.git.
Updated 26 seconds ago.

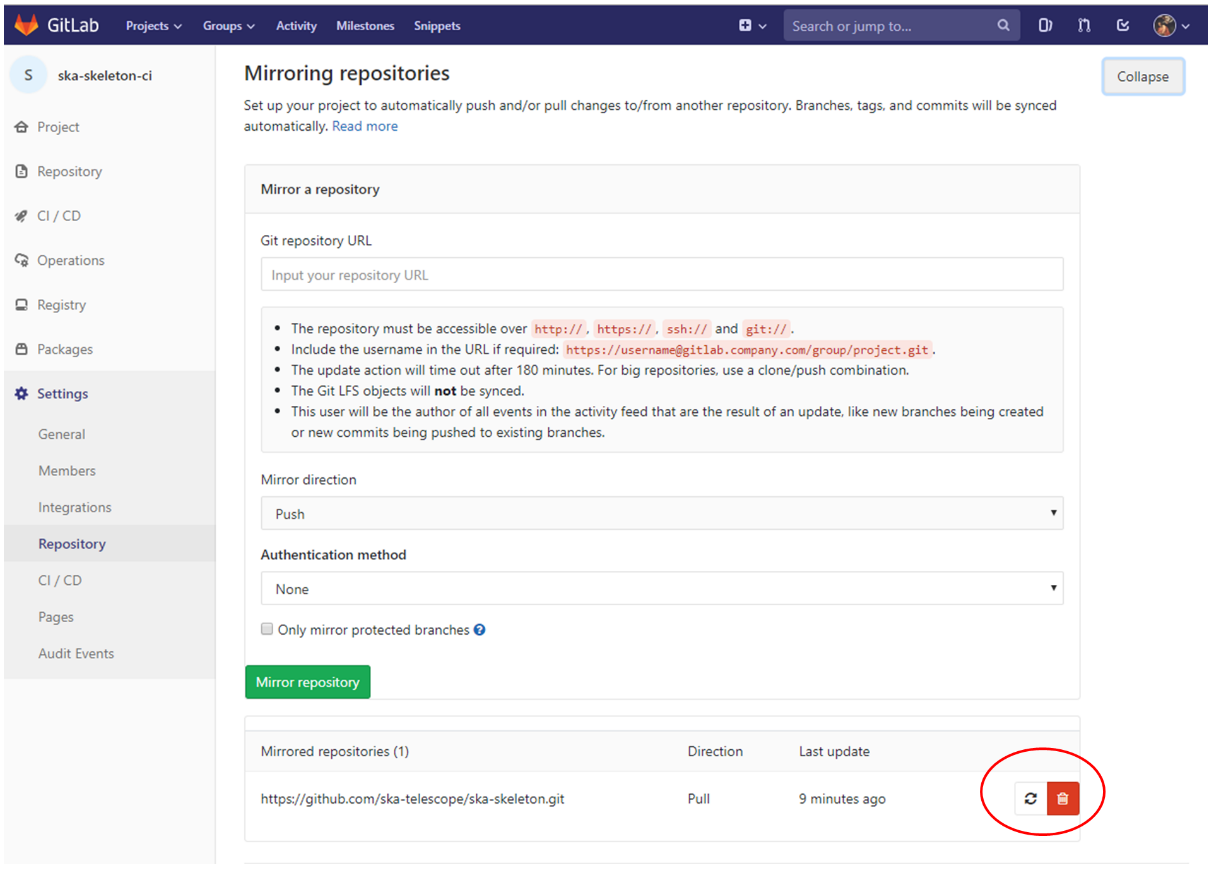
Merge branch 'master' of https://github.com/ska-telescope/ska-skeleton
Bruno S. Morgado authored 22 hours ago
124d7a1c

Name	Last commit	Last update
ska_skeleton	Integrate pylint with pytest and fix code coverage warnings	1 week ago
tests	Integrate pylint with pytest and fix code coverage warnings	1 week ago
.gitignore	Updated basic Python project skeleton	1 month ago
.gitlab-ci.yml	install requirements	5 days ago
.pylintrc	ST-18 #comment Added missing pylint initial configuration	6 days ago
CHANGELOG.rst	Updated basic Python project skeleton	1 month ago

Concerning mirroring, it is important to note the following points:

- Once you activate the pull mirroring feature, the mirror will be inserted into a queue. A scheduler will start every minute and schedule a fixed amount of mirrors for update, based on the configured maximum capacity.
  - If the mirror successfully updates it will be enqueued once again with a small backoff period.
  - If the mirror fails (eg: branch diverged from upstream), the project's backoff period will be penalized each time it fails up to a maximum amount of time
- You should not push commits directly to the repository on GitLab. Instead, any commits should be pushed to the upstream repository. Changes pushed to the upstream repository will be pulled into the GitLab repository, either:
  - Automatically within a certain period of time.
  - When a forced update is initiated.

The update can be forced with the specific button in the repository settings:



Mirroring repositories

Set up your project to automatically push and/or pull changes to/from another repository. Branches, tags, and commits will be synced automatically. Read more

Mirror a repository

Git repository URL

Input your repository URL

- The repository must be accessible over `http://`, `https://`, `ssh://` and `git://`.
- Include the username in the URL if required: `https://username@gitlab.company.com/group/project.git`.
- The update action will time out after 180 minutes. For big repositories, use a clone/push combination.
- The Git LFS objects will **not** be synced.
- This user will be the author of all events in the activity feed that are the result of an update, like new branches being created or new commits being pushed to existing branches.

Mirror direction: Push

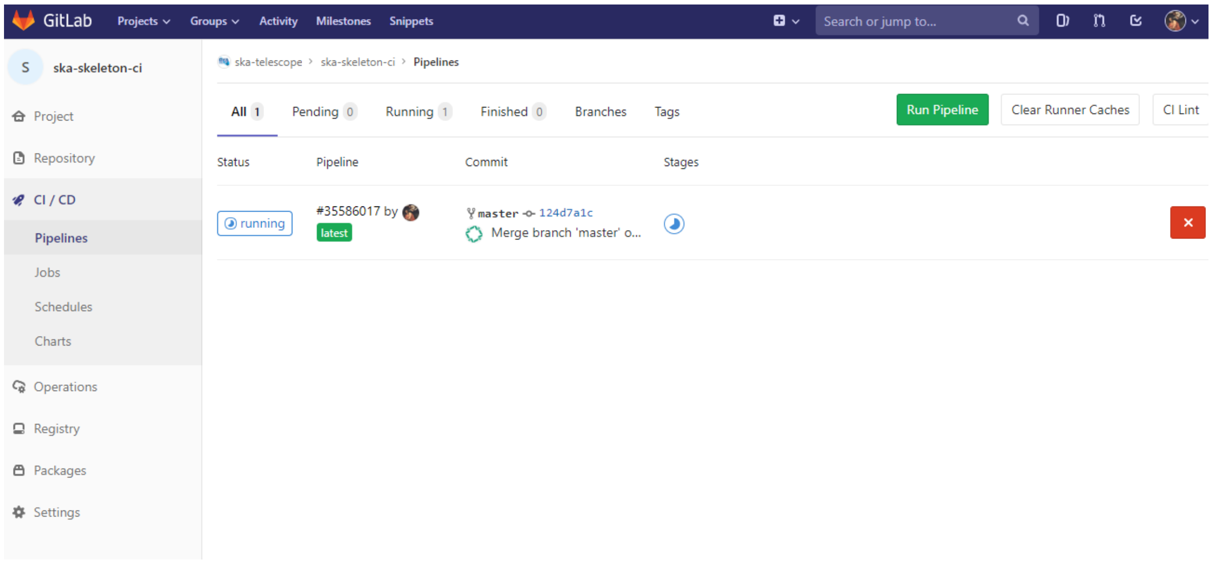
Authentication method: None

Only mirror protected branches

Mirrored repositories (1)	Direction	Last update
https://github.com/ska-telescope/ska-skeleton.git	Pull	9 minutes ago

3.3.2 Configuring a CI pipeline

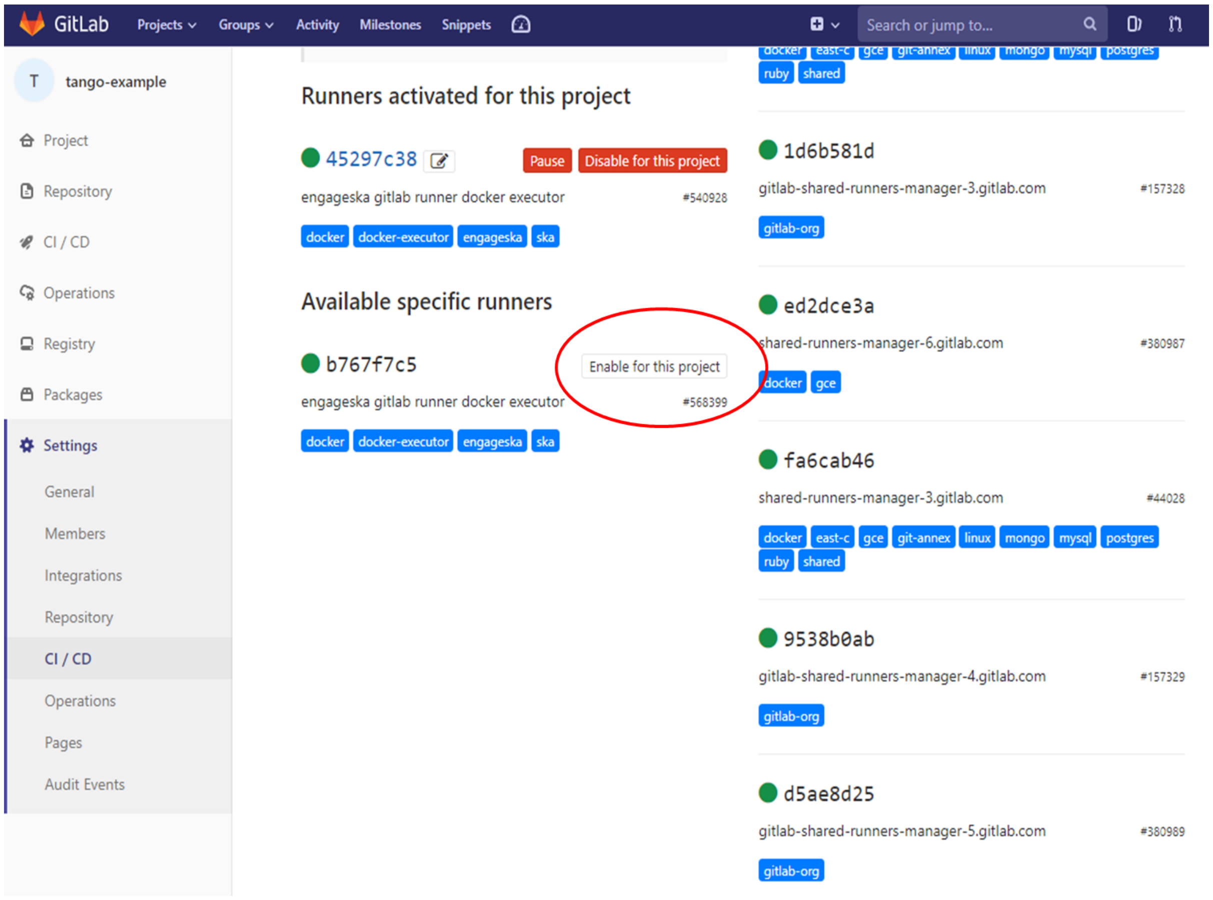
To enable the Gitlab automation, it is needed to insert a configuration file that must be placed in the root of the repository (i.e. GitHub) and called ".gitlab-ci.yml". It mainly contains definitions of how your project should be built. An example of it can be found within the project "ska-skeleton" available here. Once the file is in the root directory, it is possible to run the CI pipeline manually (creating a pipeline) or with a commit in github as soon as the mirroring finishes. The following pipeline was created manually pressing the button "Run pipeline" on a specific branch (i.e. master).



Status	Pipeline	Commit	Stages
running	#35586017 by latest	master 124d7a1c Merge branch 'master' o...	

3.3.3 Using a specific executor

The pipeline by default will run with a shared runner made available from GitLab. It is also possible to assign specific ska runner to the project (by adding the tags). To do that the option must be enabled:



Runners activated for this project

45297c38
engageska gitlab runner docker executor #540928
docker docker-executor engageska ska

Available specific runners

b767f7c5
engageska gitlab runner docker executor #568399
docker docker-executor engageska ska

docker east-c gce git-annex linux mongo mysql postgres ruby shared

1d6b581d
gitlab-shared-runners-manager-3.gitlab.com #157328
gitlab-org

ed2dce3a
shared-runners-manager-6.gitlab.com #380987
docker gce

fa6cab46
shared-runners-manager-3.gitlab.com #44028
docker east-c gce git-annex linux mongo mysql postgres ruby shared

9538b0ab
gitlab-shared-runners-manager-4.gitlab.com #157329
gitlab-org

d5ae8d25
gitlab-shared-runners-manager-5.gitlab.com #380989
gitlab-org

The EngageSKA cluster located at the Datacenter of Institute of Telecommunication (IT) in Aveiro provides some virtual machines available adding the tag “engageska” or “docker-executor” as shown here.

CHAPTER 4

Development tools

4.1 Git

Git is adopted as distributed version control system, and all SKA code shall be hosted in a git repository. The github organization *ska-telescope* can be found at https://github.com/ska-telescope . All SKA developers must have a github account and be added to the organization as part of a team.

- *Git*

4.2 Jira

Every team is tracking daily work in a team-based project on our JIRA server at [https://jira.skatelescope.org]

Todo:

- Create a new project
- Link to issue tracker

4.2.1 Definition of Done

Done-ness is defined differently at different stages of development and for different purposes.

Story

- Code is supplied with an acceptable license
- Code is peer-reviewed (via pull-request process)
- Code is checked into SKA repository with reference to ticket

- Code has tests that have adequate (between 75% and 90%) coverage
- Code compiles cleanly with no warnings
- Code adheres to SKA language specific style
- Code is deployed to continuous integration environment
- Code passes regression testing
- Code passes smoke test
- NFRs are met
- Story is tested against acceptance criteria
- Story is documented
- Story ok-ed by Product Owner

Code documentation

- Public API exposed is clearly documented
- Code is documented inline according to language specific standards
- Documentation is peer-reviewed by stakeholder (e.g. Product Owner for a feature or technical peer for an enabler) via pull-request mechanism.
- Documentation is deployed to externally visible website accessible via the developer portal.

Feature

- Feature has been demonstrated to relevant stakeholders
- Feature meets the acceptance criteria
- Feature is accepted by Feature owner
- Feature is integrated in an integration environment
- Code documentation is integrated as part of the developer portal
- Architectural documentation is updated to reflect the actual implementation

Todo:

- Testing Guidelines
- Writing Command-Line Scripts
- C or Cython Extensions

4.2.2 Python Coding Guidelines

This section describes requirements and guidelines.

Interface and Dependencies

- All code must be compatible with Python 3.5 and later.
- The new Python 3 formatting style should be used (i.e. `"{0:s}".format("spam")` instead of `"%s" % "spam"`).

Documentation and Testing

- Docstrings must be present for all public classes/methods/functions, and must follow the form outlined by PEP8 Docstring Conventions.
- Unit tests should be provided for as many public methods and functions as possible, and should adhere to Pytest best practices.

Data and Configuration

- All persistent configuration should use python-dotenv. Such configuration `.env` files should be placed at the top of the `ska_skeleton` module and provide a description that is sufficient for users to understand the settings changes.

Standard output, warnings, and errors

The built-in `print(...)` function should only be used for output that is explicitly requested by the user, for example `print_header(...)` or `list_catalogs(...)`. Any other standard output, warnings, and errors should follow these rules:

- For errors/exceptions, one should always use `raise` with one of the built-in exception classes, or a custom exception class. The nondescript `Exception` class should be avoided as much as possible, in favor of more specific exceptions (*IOError*, *ValueError*, etc.).
- For warnings, one should always use `warnings.warn(message, warning_class)`. These get redirected to `log.warning()` by default.
- For informational and debugging messages, one should always use `log.info(message)` and `log.debug(message)`.

The logging system should use the built-in Python logging module.

Coding Style/Conventions

- The code will follow the standard PEP8 Style Guide for Python Code. In particular, this includes using only 4 spaces for indentation, and never tabs.
- The `import numpy as np`, `import matplotlib as mpl`, and `import matplotlib.pyplot as plt` naming conventions should be used wherever relevant. `from packagename import *` should never be used, except as a tool to flatten the namespace of a module. An example of the allowed usage is given in *Acceptable use of from module import **.
- Classes should either use direct variable access, or Python's property mechanism for setting object instance variables. `get_value`/`set_value` style methods should be used only when getting and setting the values requires a computationally-expensive operation. *Properties vs. get_/set_* below illustrates this guideline.
- Classes should use the builtin `super()` function when making calls to methods in their super-class(es) unless there are specific reasons not to. `super()` should be used consistently in all subclasses since it does not work otherwise. *super() vs. Direct Calling* illustrates why this is important.

- Multiple inheritance should be avoided in general without good reason.
- `__init__.py` files for modules should not contain any significant implementation code. `__init__.py` can contain docstrings and code for organizing the module layout, however (e.g. `from submodule import *` in accord with the guideline above). If a module is small enough that it fits in one file, it should simply be a single file, rather than a directory with an `__init__.py` file.

Unicode guidelines

For maximum compatibility, we need to assume that writing non-ASCII characters to the console or to files will not work.

Including C Code

- When C extensions are used, the Python interface for those extensions must meet the aforementioned Python interface guidelines.
- The use of Cython is strongly recommended for C extensions. Cython extensions should store `.pyx` files in the source code repository, but they should be compiled to `.c` files that are updated in the repository when important changes are made to the `.pyx` file.
- In cases where C extensions are needed but Cython cannot be used, the PEP 7 Style Guide for C Code is recommended.

Examples

This section shows examples in order to illustrate points from the guidelines.

Properties vs. get_/set_

This example shows a sample class illustrating the guideline regarding the use of properties as opposed to getter/setter methods.

Let's assuming you've defined a `':class:`Star`'` class and create an instance like this:

```
>>> s = Star(B=5.48, V=4.83)
```

You should always use attribute syntax like this:

```
>>> s.color = 0.4
>>> print(s.color)
0.4
```

Using Python properties, attribute syntax can still do anything possible with a get/set method. For lengthy or complex calculations, however, use a method:

```
>>> print(s.compute_color(5800, age=5e9))
0.4
```

super() vs. Direct Calling

By calling `super()` the entire method resolution order for `D` is precomputed, enabling each superclass to cooperatively determine which class should be handed control in the next `super()` call:

```python
# This is safe

class A(object):
    def method(self):
        print('Doing A')

class B(A):
    def method(self):
        print('Doing B')
        super().method()


class C(A):
    def method(self):
        print('Doing C')
        super().method()

class D(C, B):
    def method(self):
        print('Doing D')
        super().method()
```

```
>>> d = D()
>>> d.method()
Doing D
Doing C
Doing B
Doing A
```

As you can see, each superclass's method is entered only once. For this to work it is very important that each method in a class that calls its superclass's version of that method use `super()` instead of calling the method directly. In the most common case of single-inheritance, using `super()` is functionally equivalent to calling the superclass's method directly. But as soon as a class is used in a multiple-inheritance hierarchy it must use `super()` in order to cooperate with other classes in the hierarchy.

Note: For more information on the the benefits of `super()`, see https://rhettinger.wordpress.com/2011/05/26/super-considered-super/

Acceptable use of `from module import *`

`from module import *` is discouraged in a module that contains implementation code, as it impedes clarity and often imports unused variables. It can, however, be used for a package that is laid out in the following manner:

```
packagename
packagename/__init__.py
packagename/submodule1.py
packagename/submodule2.py
```

In this case, `packagename/__init__.py` may be:

```python
"""
A docstring describing the package goes here
"""
```

(continues on next page)

(continued from previous page)

```
from submodule1 import *
from submodule2 import *
```

This allows functions or classes in the submodules to be used directly as `packagename.foo` rather than `packagename.submodule1.foo`. If this is used, it is strongly recommended that the submodules make use of the `__all__` variable to specify which modules should be imported. Thus, `submodule2.py` might read:

```
from numpy import array, linspace

__all__ = ['foo', 'AClass']

def foo(bar):
    # the function would be defined here
    pass

class AClass(object):
    # the class is defined here
    pass
```

This ensures that `from submodule import *` only imports `':func:`foo'` and `':class:`AClass'`, but not `':class:`numpy.array'` or `':func:`numpy.linspace'`.

Acknowledgements

The present document's coding guidelines are derived from project Astropy's codding guidelines.

CHAPTER 5

Development guidelines

5.1 Definition of Done

The definition of done is used to guide teams in planning and estimating the size of stories and features:

- *Definition of Done*

5.2 Python coding guidelines

A Python skeleton project is created for use within the SKA Telescope. This skeleton purpose is to enforce coding best practices and bootstrap the initial project setup. Any development should start by forking this skeleton project and change the apropriate files.

- *Python Coding Guidelines*

5.2.1 List of projects

The following list is automatically extracted from our github organisation page at [https://skatelescope.org/ska-telescope]

- p1
- p2

5.2.2 Create a new project

Todo:

- create a new github repository
- starting from ska-skeleton

5.2.3 Documenting a project

Todo:

- the docs folder
- using readthedocs as the ska-skeleton project
- adding textual documentation
- adding automatically extracted documentation
- documenting the public API

5.2.4 Licensing your project

SKA organisation promotes a model of open and transparent collaboration. In this model collaboration is made possible using permissive licenses, and not by pursuing the ownership of code by the organisation. Copyright will thus belong to the institutions contributing to source code development for the lifetime of the project, and software developed for the SKA telescope will be available to the wider community as source code. Redistribution of the SKA software will always maintain the original copyright information, acknowledging the original software contributor.

License File

Every software repository shall be licensed according to the SPDX standard. Every repository shall contain a LICENSE file in its top directory, indicating the copyright holder and the license used for the software in its full textual representation.

Licenses

SKA office will automatically accept the BSD 3-clause new LICENSE and any exception to this shall be justified and agreed with SKA office Software Quality Assurance engineer. A template of the license is presented at the end of this page. Existing repositories already published with permissive licenses such as Apache 2.0 or MIT licenses will also be accepted as part of the handover procedure, while new repositories are encouraged to adopt the recommended BSD license.

Copyright Information

Copyright information shall be included in the license file, clearly stating the year and the institution the copyright applies to, in the form:

```
Copyright <years> <institution>

An example of this for the SKA organisation would be:

Copyright 2018 SKA Organisation
```

A non exhaustive list of possible copyright notices, based on pre construction SKA collaborators, may include one or more of the following:

```
Copyright 2018 AIT Aveiro
Copyright 2018 ASTRON
Copyright 2018 ATC
Copyright 2018 CSIRO
Copyright 2018 ICRAR
Copyright 2018 INAF
Copyright 2018 NCRA
Copyright 2018 SARAO
Copyright 2018 University of Malta
Copyright 2018 University of Manchester
Copyright 2018 University of Oxford
...
```

A single license file can contain multiple copyright notices, indicating the major contributors to the software repositories. It is not in the scope of the copyright notice to maintain an updated list of single contributors which can always be extracted from the DVCS server system in a more reliable and maintainable way. Whenever a license assumes that copyright is explicitly stated as part of the header of every source code file, this can be summarized into a single centralized COPYRIGHT file in the top directory of the repository, containing all copyright attributions and referred to by the single header comments in the source code:

```
Copyright 2018 The Foo Project Developers. See the COPYRIGHT file at the top-level␣
↪directory of this distribution.
```

It will be the duty of single repository administrators to make sure that copyright notices are maintained and updated according to the institution contributing to the project.

BSD 3-Clause "New" or "Revised" License text template

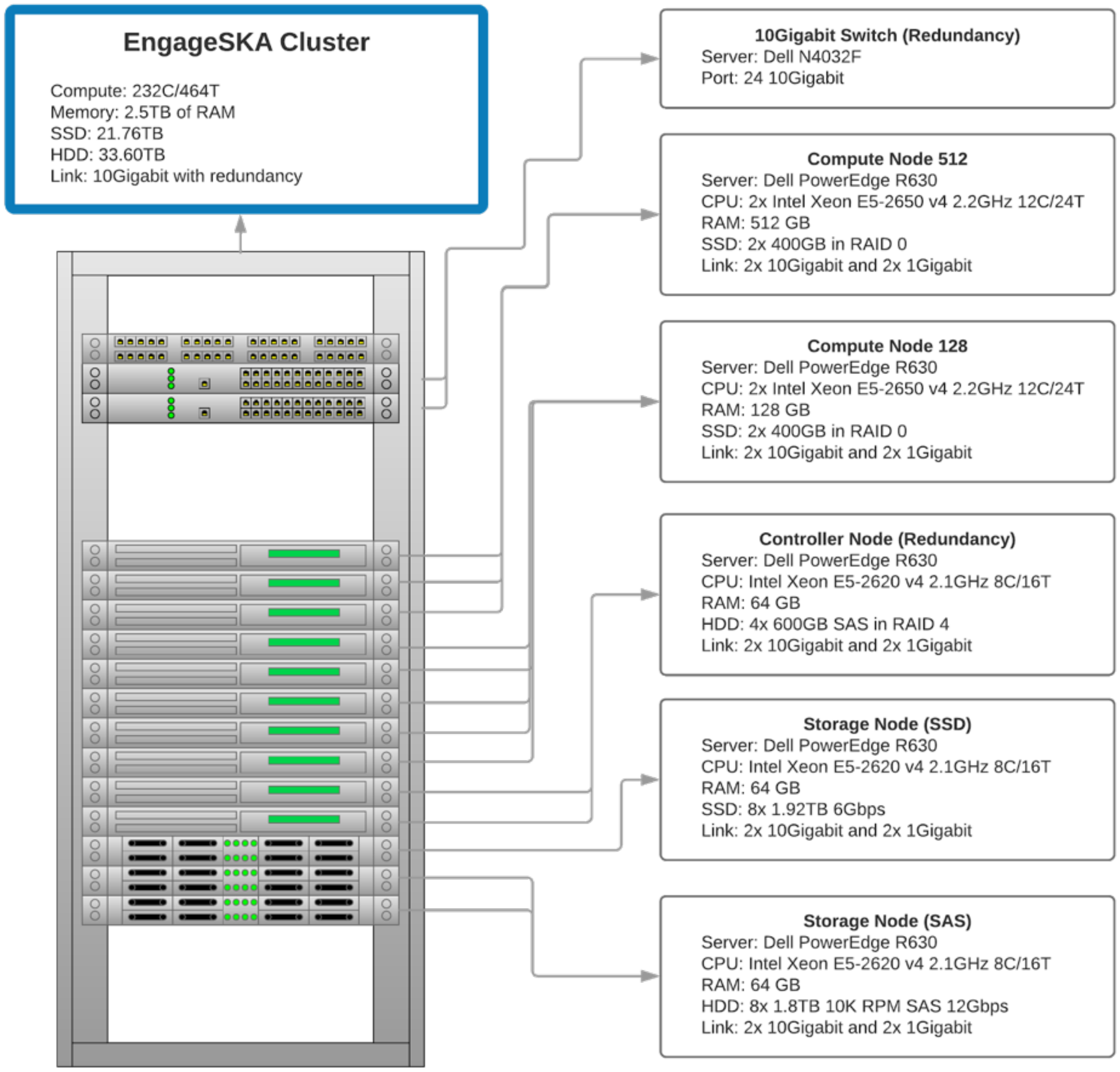
```
Copyright <YEAR> <COPYRIGHT HOLDER>

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice,
this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright
notice, this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its
contributors may be used to endorse or promote products derived from this
software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
```

CHAPTER 6

Projects

6.1 AIT cluster

6.1.1 Cluster specs


EngageSKA Cluster
Compute: 232C/464T
Memory: 2.5TB of RAM
SSD: 21.76TB
HDD: 33.60TB
Link: 10Gigabit with redundancy
10Gigabit Switch (Redundancy)
Server: Dell N4032F
Port: 24 10Gigabit
Compute Node 512
Server: Dell PowerEdge R630
CPU: 2x Intel Xeon E5-2650 v4 2.2GHz 12C/24T
RAM: 512 GB
SSD: 2x 400GB in RAID 0
Link: 2x 10Gigabit and 2x 1Gigabit
Compute Node 128
Server: Dell PowerEdge R630
CPU: 2x Intel Xeon E5-2650 v4 2.2GHz 12C/24T
RAM: 128 GB
SSD: 2x 400GB in RAID 0
Link: 2x 10Gigabit and 2x 1Gigabit
Controller Node (Redundancy)
Server: Dell PowerEdge R630
CPU: Intel Xeon E5-2620 v4 2.1GHz 8C/16T
RAM: 64 GB
HDD: 4x 600GB SAS in RAID 4
Link: 2x 10Gigabit and 2x 1Gigabit
Storage Node (SSD)
Server: Dell PowerEdge R630
CPU: Intel Xeon E5-2620 v4 2.1GHz 8C/16T
RAM: 64 GB
SSD: 8x 1.92TB 6Gbps
Link: 2x 10Gigabit and 2x 1Gigabit
Storage Node (SAS)
Server: Dell PowerEdge R630
CPU: Intel Xeon E5-2620 v4 2.1GHz 8C/16T
RAM: 64 GB
HDD: 8x 1.8TB 10K RPM SAS 12Gbps
Link: 2x 10Gigabit and 2x 1Gigabit

6.1.2 Access the cluster

The EngageSKA cluster locates at the Datacenter of Institute of Telecommunication (IT) in Aveiro. To have access to the cluster, it is required to be inside the facilities or have VPN credentials to access the IT network remotely.

6.1.3 Access to the network using VPN

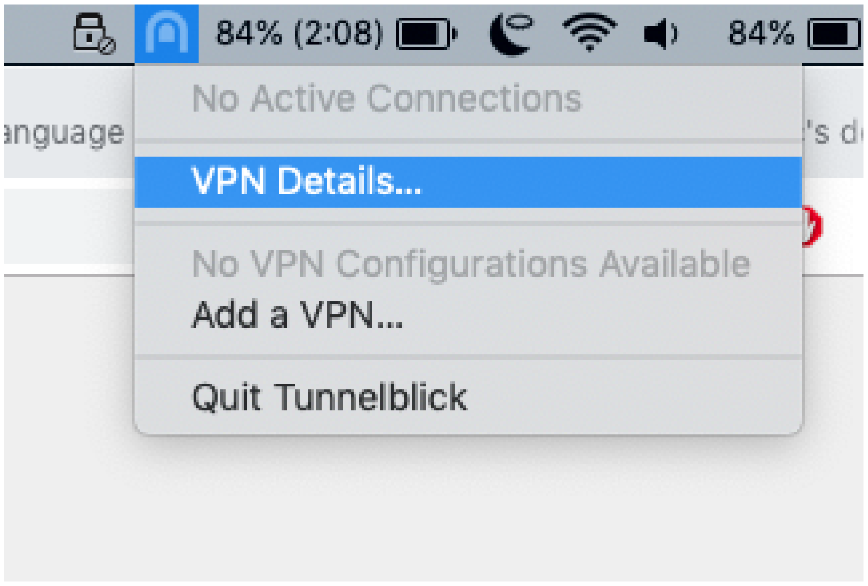
At the moment, VPN credentials are sent individually and is required to send an email to Dzianis Bartashevich (bartashevich@av.it.pt) with the knowledge from Marco Bartolini (M.Bartolini@skatelescope.org).

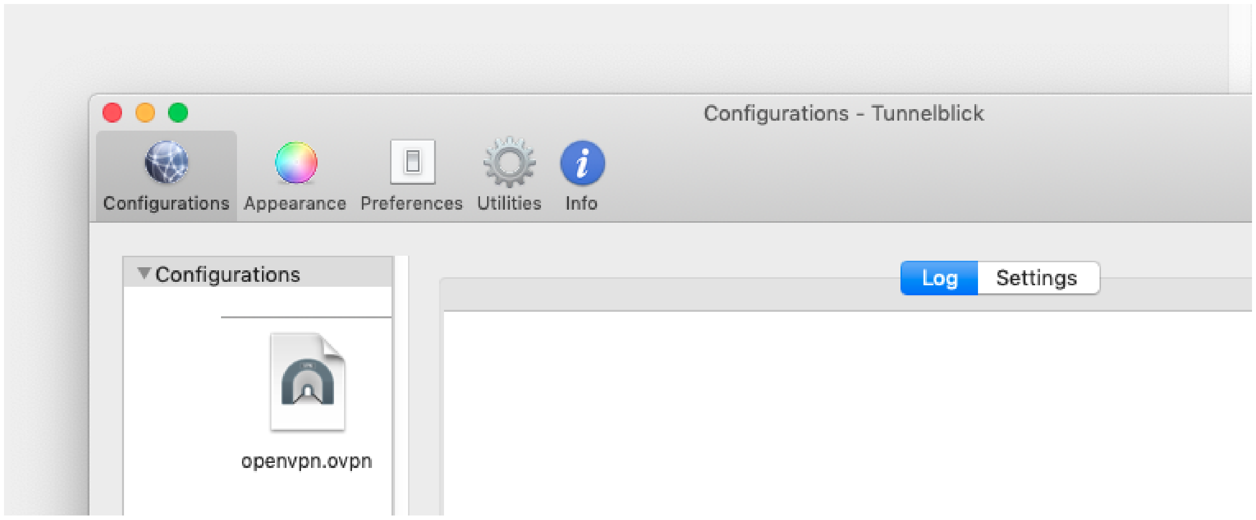
You will receive an .ovpn key to access the private network.

In order to connect you will need an OpenVPN client:

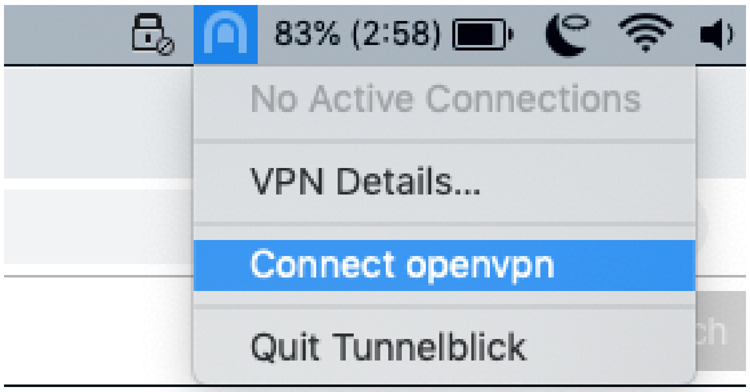
- Windows and Linux: https://openvpn.net/community-downloads/
- MacOS: https://tunnelblick.net/

Now just drag or add the .ovpn file to VPN the client. The follow tutorial is for tunnelblick, but it is similar to the other VPN clients.

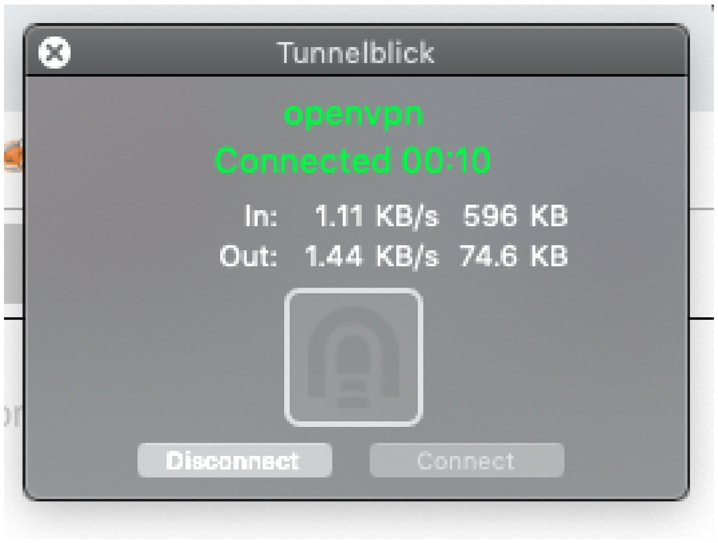




Now that you have configured the VPN credentials click on the VPN client icon located at the toolbar and connect to the private network.

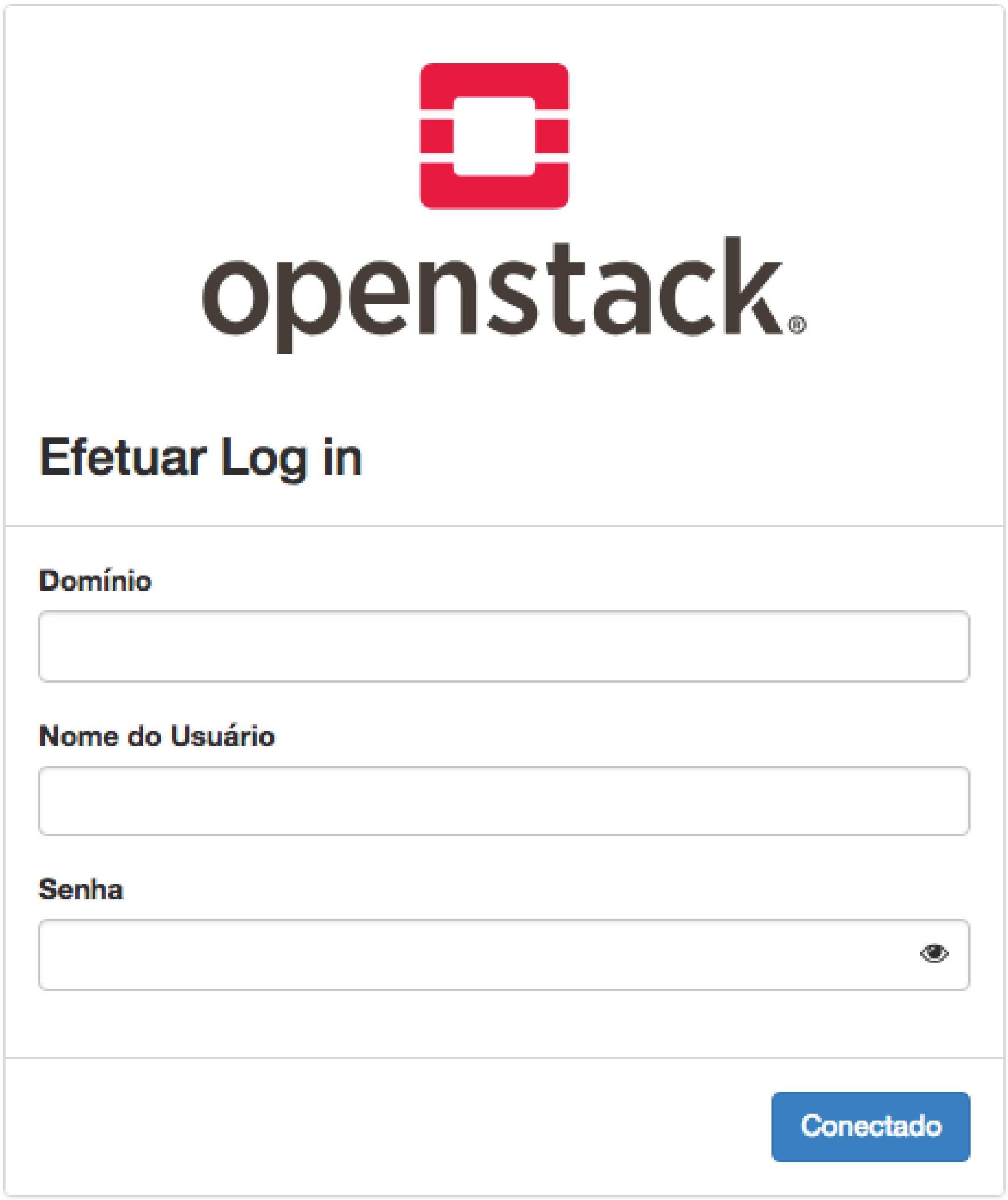


If all went okay, you are successfully connected.



```
 In:   1.11 KB/s  596 KB
Out:   1.44 KB/s  74.6 KB
```

6.1.4 Access to the OpenStack platform (Virtualization)



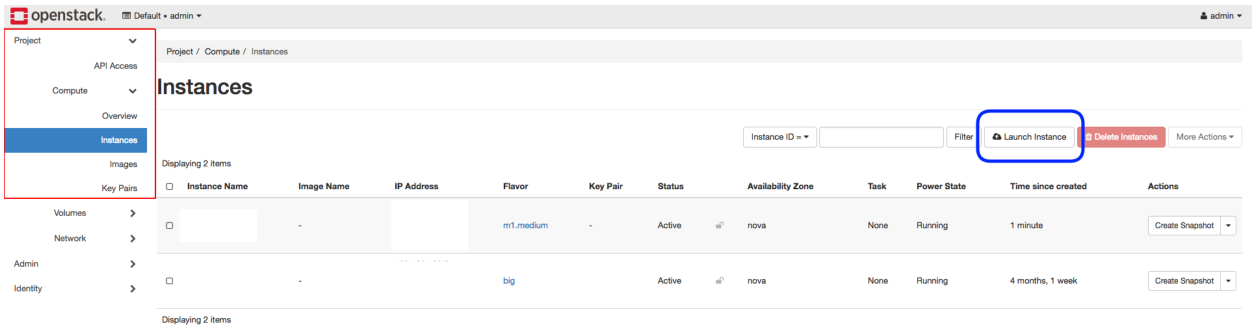
The OpenStack platform requires authentication in order to use it.

At the moment, OpenStack credentials are sent individually and it is required to send an email to Dzianis Bartashevich (bartashevich@av.it.pt with the knowledge from Marco Bartolini (M.Bartolini@skatelescope.org). In the next phase, OpenStack could support GitHub authentication.

To access the OpenStack platform go to http://192.168.93.215/dashboard (require VPN) and login with your credentials.

Virtual machine deployment

- At the sidebar, go to Project -> Compute -> Instances and click on the “Launch Instance” button:

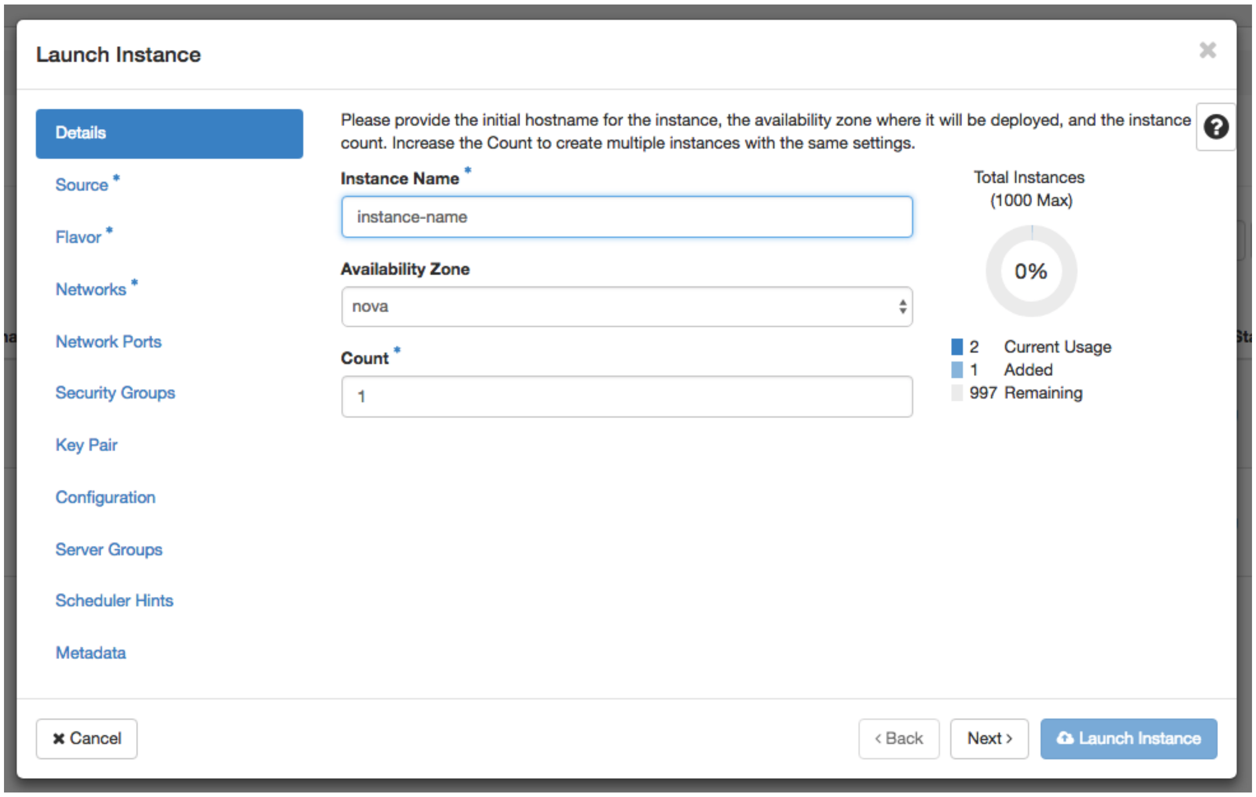


Displaying 2 items

Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
	-		m1.medium	-	Active	nova	None	Running	1 minute	Create Snapshot
	-		big		Active	nova	None	Running	4 months, 1 week	Create Snapshot

Displaying 2 items

- At this stage a menu will pop-up and will ask to specify virtual machine caracteristics, chose an name for virtual machine:



Please provide the initial hostname for the instance, the availability zone where it will be deployed, and the instance count. Increase the Count to create multiple instances with the same settings.

Instance Name *: instance-name

Availability Zone: nova

Count *: 1

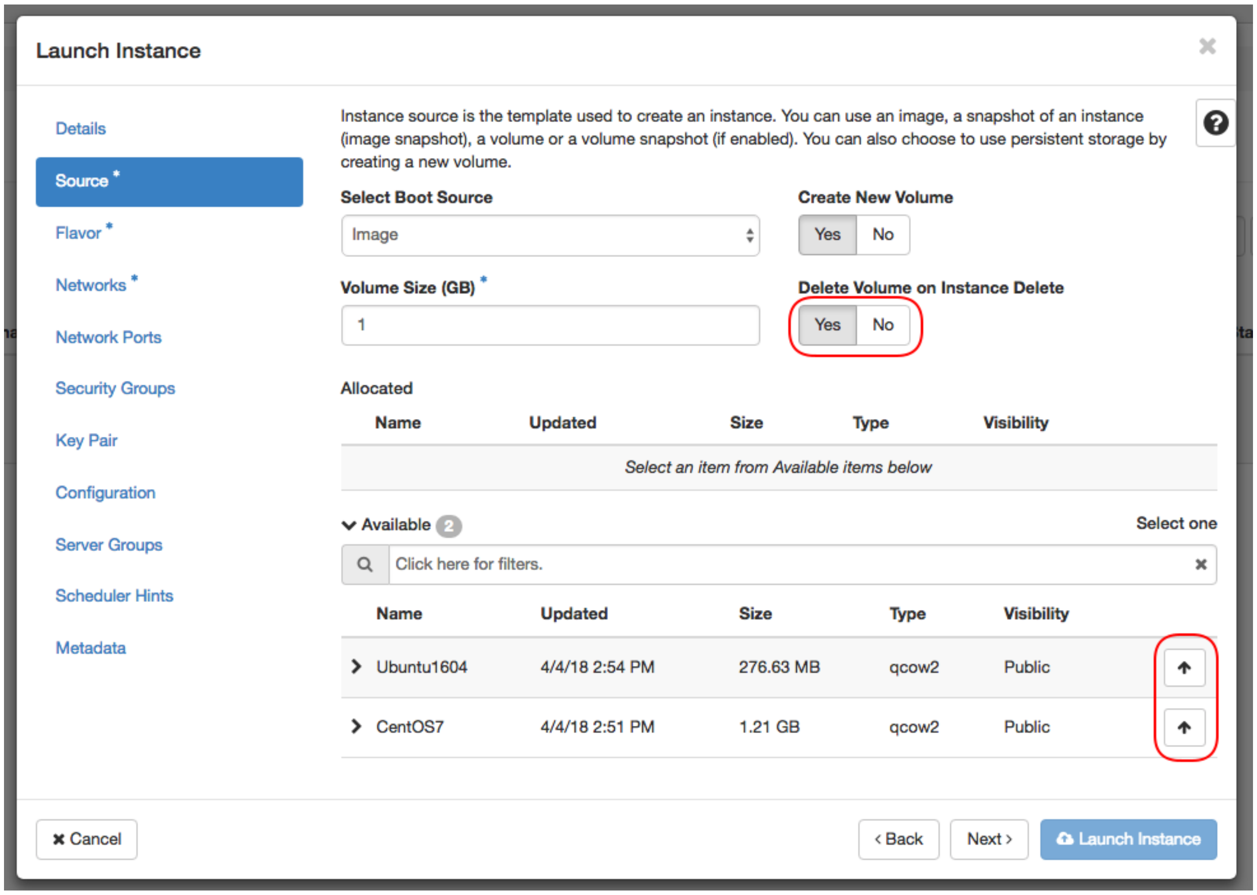
Total Instances (1000 Max)

0%

■	2	Current Usage
■	1	Added
□	997	Remaining

- Select the Operating System you want your VM to have:

NOTE: Please choose the option “Yes” at “Delete Volume on Instance Delete” so when you decide to delete the instance the volume will be also deleted and not occupy unnecessary space



Launch Instance

Instance source is the template used to create an instance. You can use an image, a snapshot of an instance (image snapshot), a volume or a volume snapshot (if enabled). You can also choose to use persistent storage by creating a new volume.

Select Boot Source: Image

Create New Volume: [✓ Yes] No

Volume Size (GB) *: 1

Delete Volume on Instance Delete: [✓ Yes] No

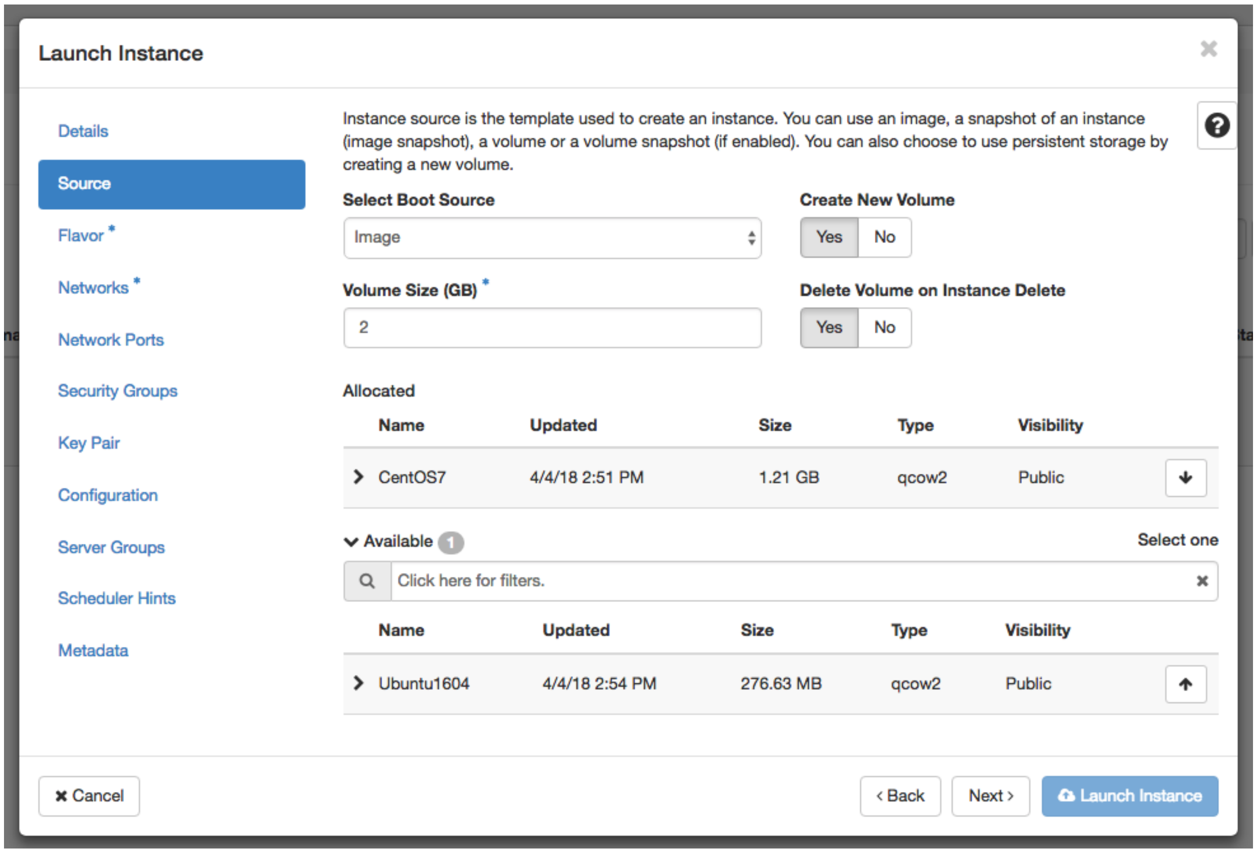
Allocated

Name	Updated	Size	Type	Visibility
Select an item from Available items below				

Available 2 — Select one

Click here for filters.

Name	Updated	Size	Type	Visibility
Ubuntu1604	4/4/18 2:54 PM	276.63 MB	qcow2	Public
CentOS7	4/4/18 2:51 PM	1.21 GB	qcow2	Public


Launch Instance

Instance source is the template used to create an instance. You can use an image, a snapshot of an instance (image snapshot), a volume or a volume snapshot (if enabled). You can also choose to use persistent storage by creating a new volume.

Select Boot Source: Image

Create New Volume: [✓ Yes] No

Volume Size (GB) *: 2

Delete Volume on Instance Delete: [✓ Yes] No

Allocated

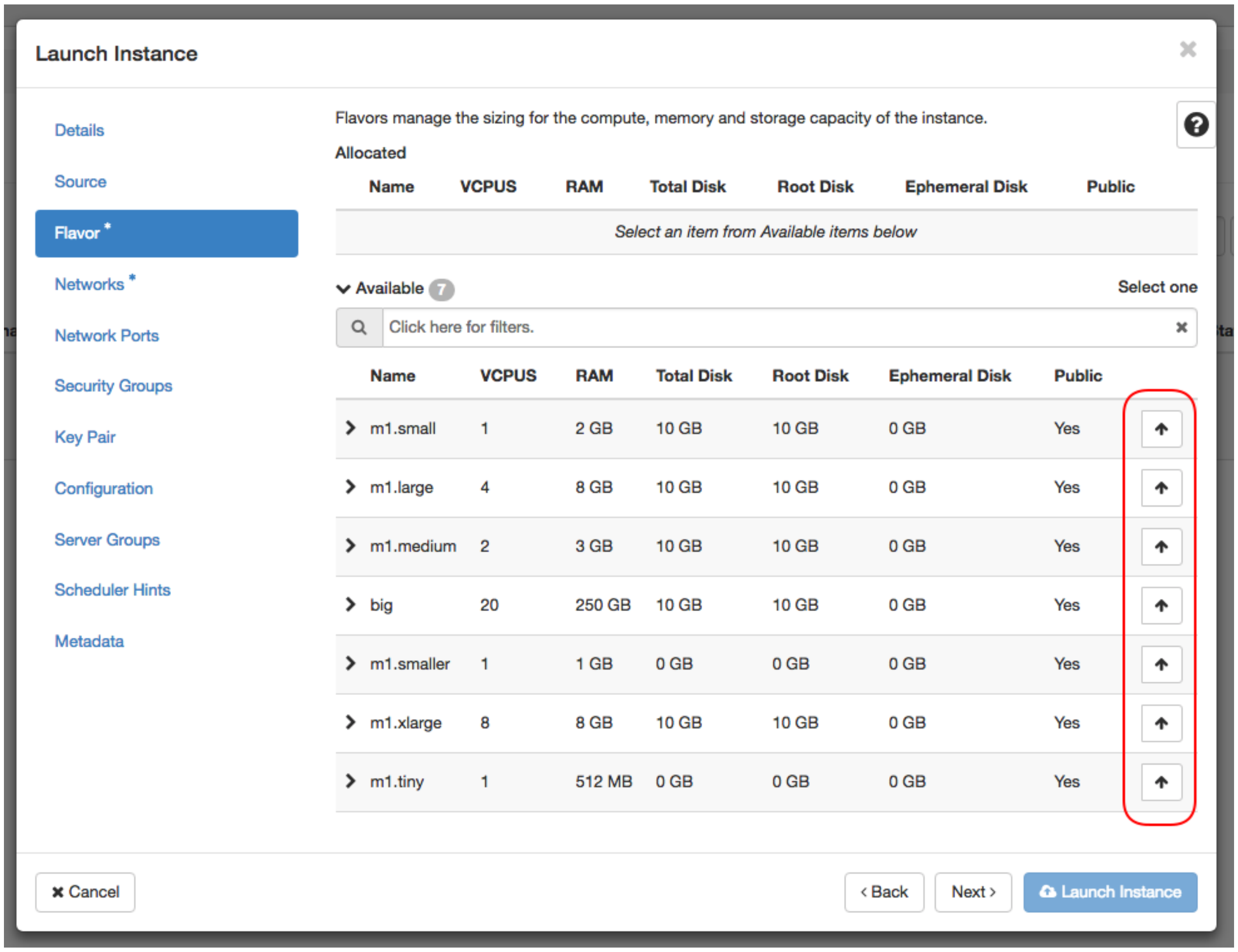
Name	Updated	Size	Type	Visibility
CentOS7	4/4/18 2:51 PM	1.21 GB	qcow2	Public

Available 1 — Select one

Click here for filters.

Name	Updated	Size	Type	Visibility
Ubuntu1604	4/4/18 2:54 PM	276.63 MB	qcow2	Public

- Select the flavor which you want your VM to have:



Flavors manage the sizing for the compute, memory and storage capacity of the instance.

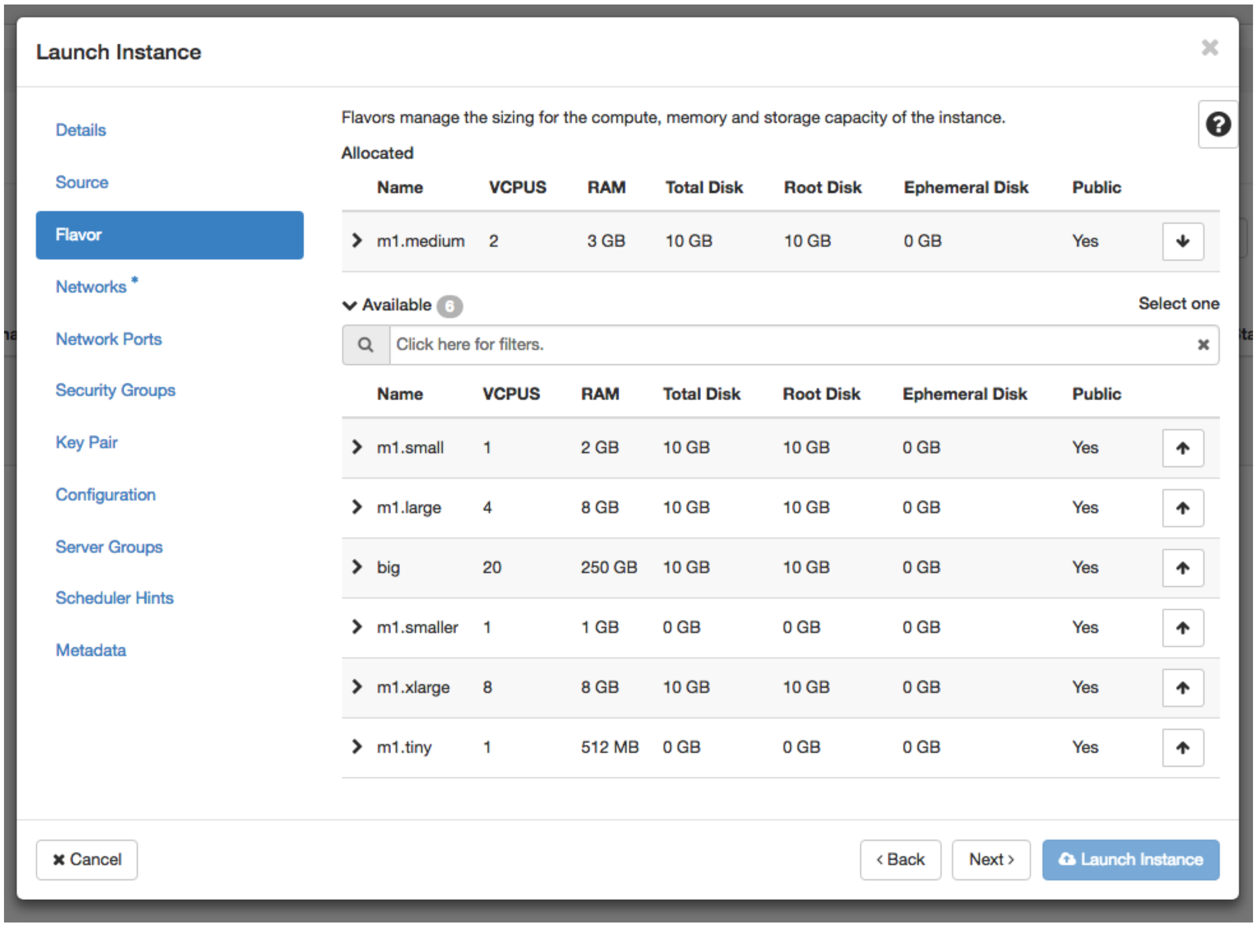
Allocated

Name	VCPUS	RAM	Total Disk	Root Disk	Ephemeral Disk	Public
Select an item from Available items below						

Available 7 Select one

Click here for filters.

Name	VCPUS	RAM	Total Disk	Root Disk	Ephemeral Disk	Public
m1.small	1	2 GB	10 GB	10 GB	0 GB	Yes
m1.large	4	8 GB	10 GB	10 GB	0 GB	Yes
m1.medium	2	3 GB	10 GB	10 GB	0 GB	Yes
big	20	250 GB	10 GB	10 GB	0 GB	Yes
m1.smaller	1	1 GB	0 GB	0 GB	0 GB	Yes
m1.xlarge	8	8 GB	10 GB	10 GB	0 GB	Yes
m1.tiny	1	512 MB	0 GB	0 GB	0 GB	Yes



Launch Instance

Flavors manage the sizing for the compute, memory and storage capacity of the instance.

Allocated

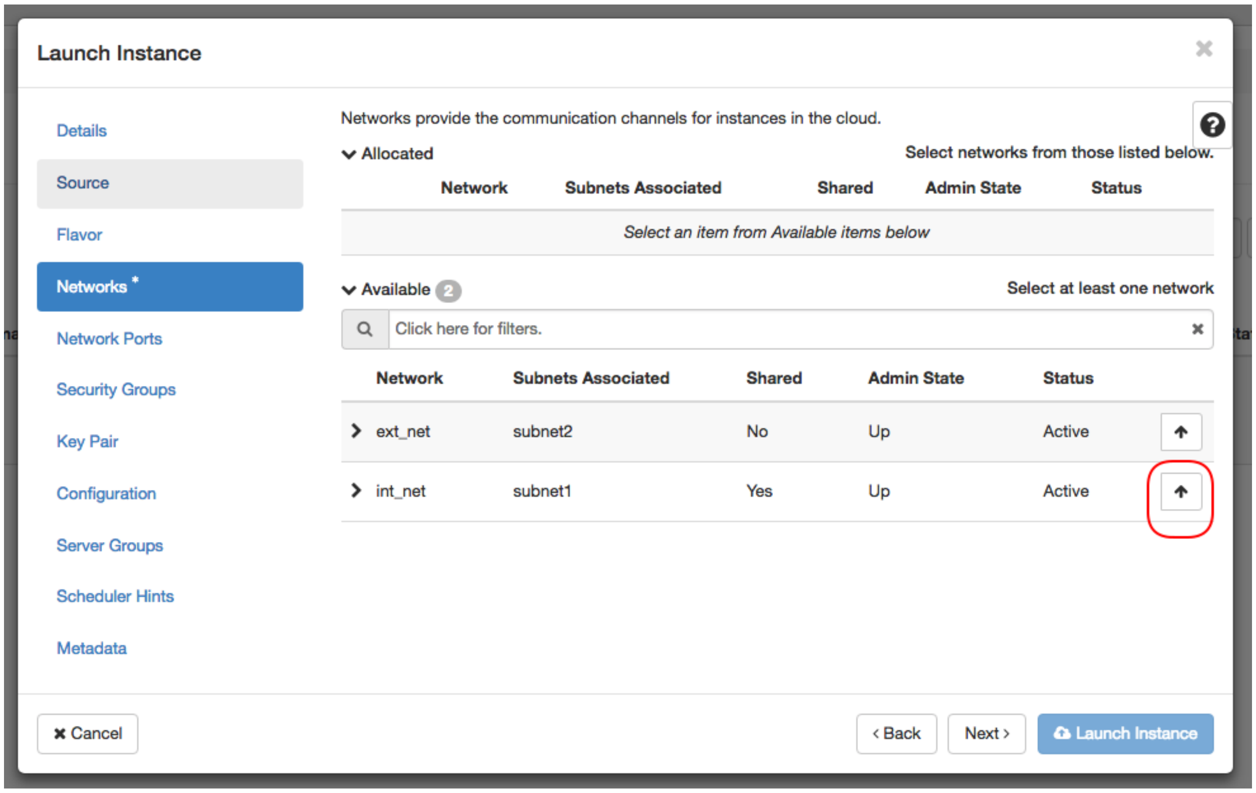
Name	VCPUS	RAM	Total Disk	Root Disk	Ephemeral Disk	Public
m1.medium	2	3 GB	10 GB	10 GB	0 GB	Yes

Available 6 — Select one

Click here for filters.

Name	VCPUS	RAM	Total Disk	Root Disk	Ephemeral Disk	Public
m1.small	1	2 GB	10 GB	10 GB	0 GB	Yes
m1.large	4	8 GB	10 GB	10 GB	0 GB	Yes
big	20	250 GB	10 GB	10 GB	0 GB	Yes
m1.smaller	1	1 GB	0 GB	0 GB	0 GB	Yes
m1.xlarge	8	8 GB	10 GB	10 GB	0 GB	Yes
m1.tiny	1	512 MB	0 GB	0 GB	0 GB	Yes

- Select private network (int-net):



Networks provide the communication channels for instances in the cloud.

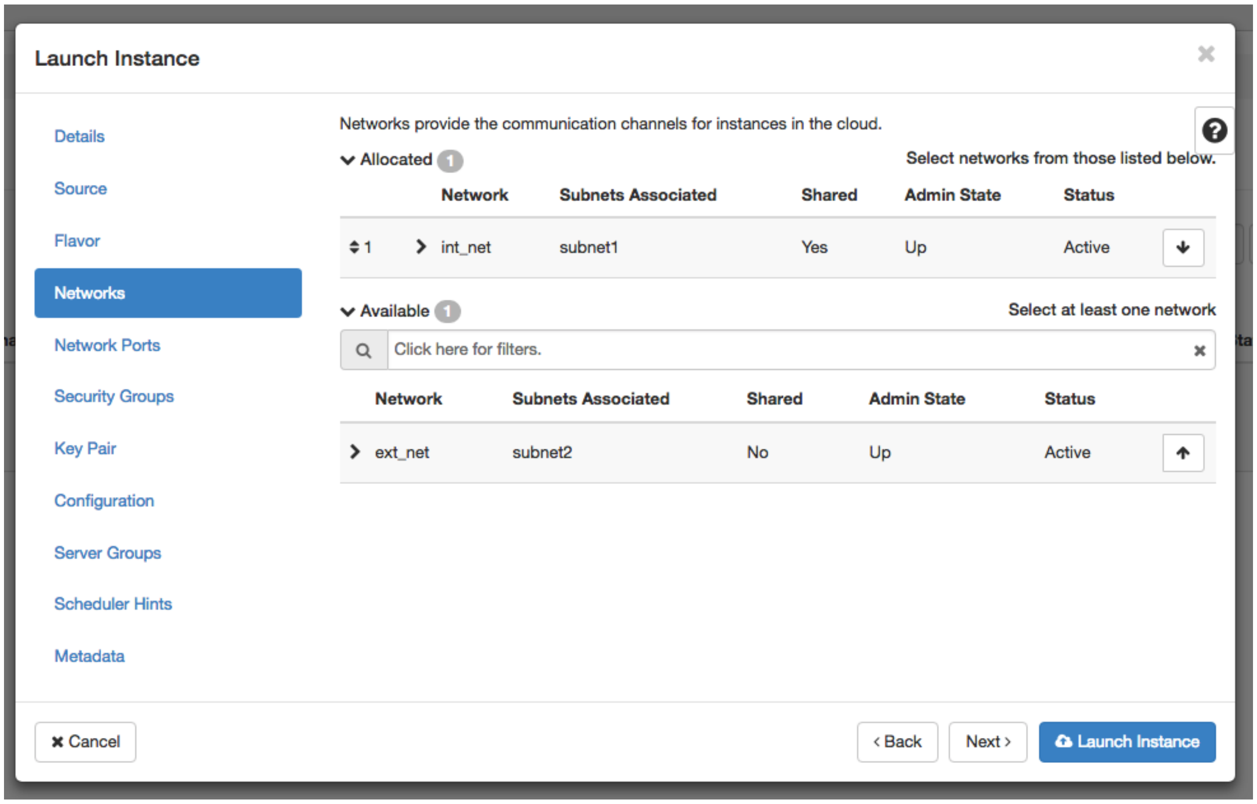
Allocated — Select networks from those listed below.

Network	Subnets Associated	Shared	Admin State	Status
Select an item from Available items below				

Available 2 — Select at least one network

Click here for filters.

Network	Subnets Associated	Shared	Admin State	Status
ext_net	subnet2	No	Up	Active
int_net	subnet1	Yes	Up	Active



Networks provide the communication channels for instances in the cloud.

Allocated 1 — Select networks from those listed below.

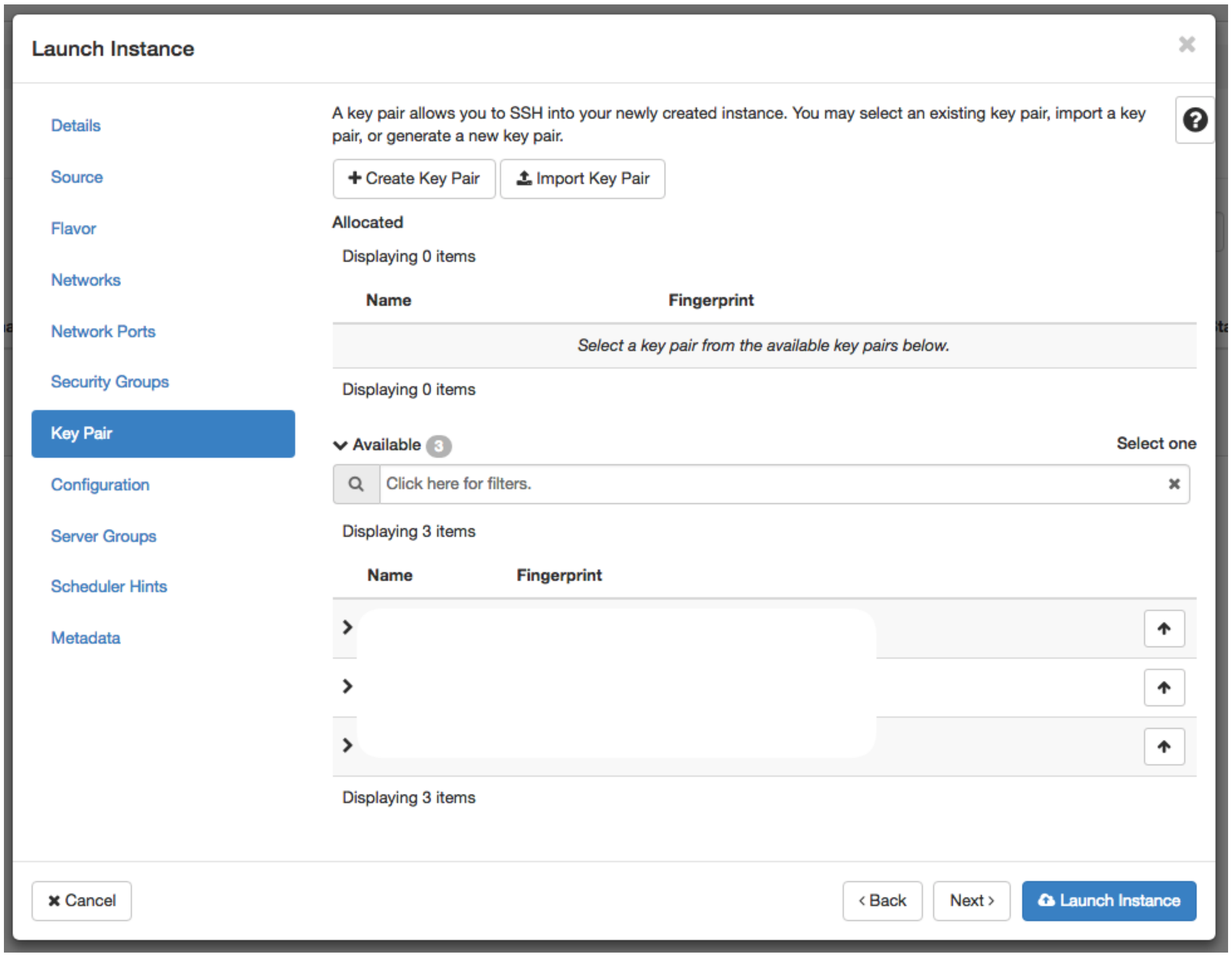
	Network	Subnets Associated	Shared	Admin State	Status
1	int_net	subnet1	Yes	Up	Active

Available 1 — Select at least one network

Click here for filters.

Network	Subnets Associated	Shared	Admin State	Status
ext_net	subnet2	No	Up	Active

- Create or use ssh key to enable ssh access to the VM:



Launch Instance

A key pair allows you to SSH into your newly created instance. You may select an existing key pair, import a key pair, or generate a new key pair.

Allocated

Displaying 0 items

Name	Fingerprint
Select a key pair from the available key pairs below.	

Displaying 0 items

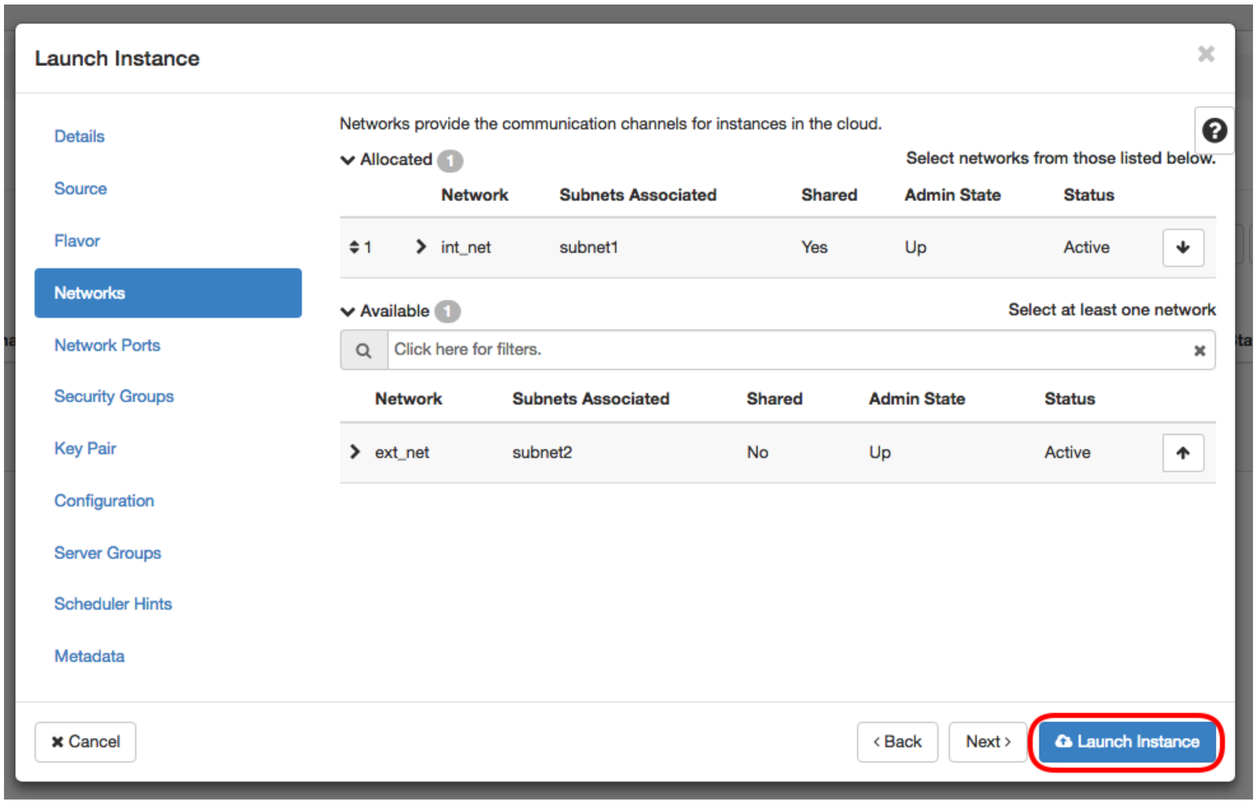
Available 3 — Select one

Displaying 3 items

Name	Fingerprint

Displaying 3 items

- In the end press on “Launch Instance” button at the bottom. This initiates the virtual machine deployment. It could take a while:



Networks provide the communication channels for instances in the cloud.

Allocated 1 — Select networks from those listed below.

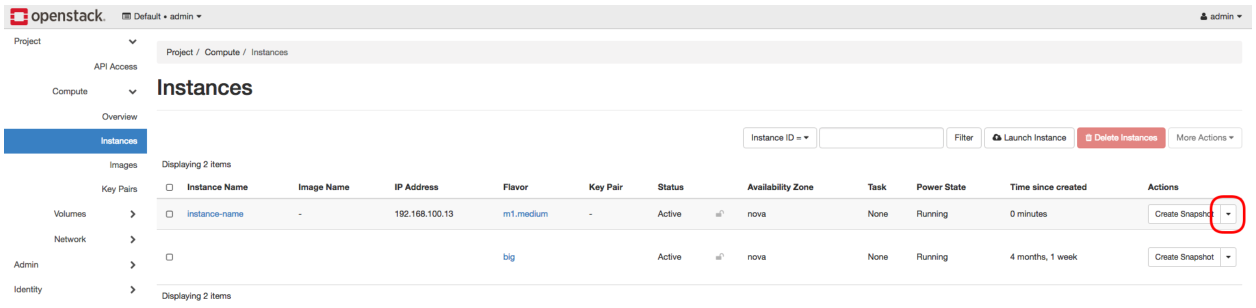
	Network	Subnets Associated	Shared	Admin State	Status
1	int_net	subnet1	Yes	Up	Active

Available 1 — Select at least one network

Click here for filters.

Network	Subnets Associated	Shared	Admin State	Status
ext_net	subnet2	No	Up	Active

- When the Power State become "Running", the virtual machine has been successfully deployed and is ready to be used:



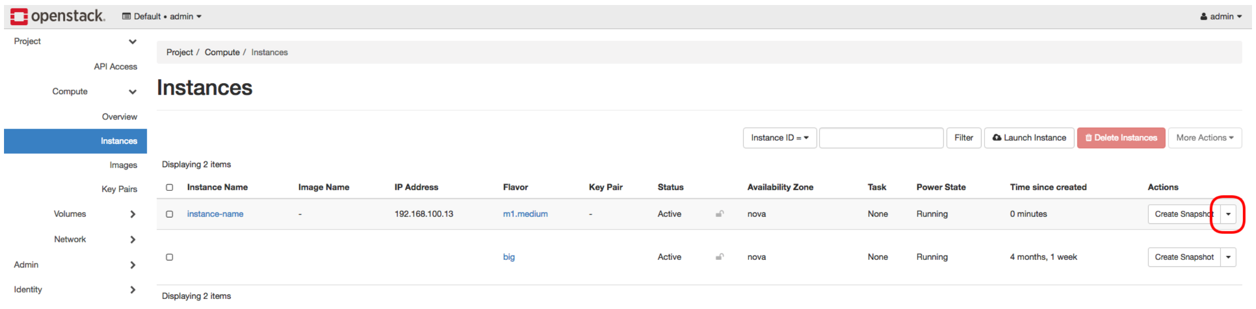
Instances

Displaying 2 items

Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
instance-name	-	192.168.100.13	m1.medium	-	Active	nova	None	Running	0 minutes	Create Snapshot
			big		Active	nova	None	Running	4 months, 1 week	Create Snapshot

Displaying 2 items

- Since the VM is deployed inside private network you will need to associate Floating IP from your network have the access:

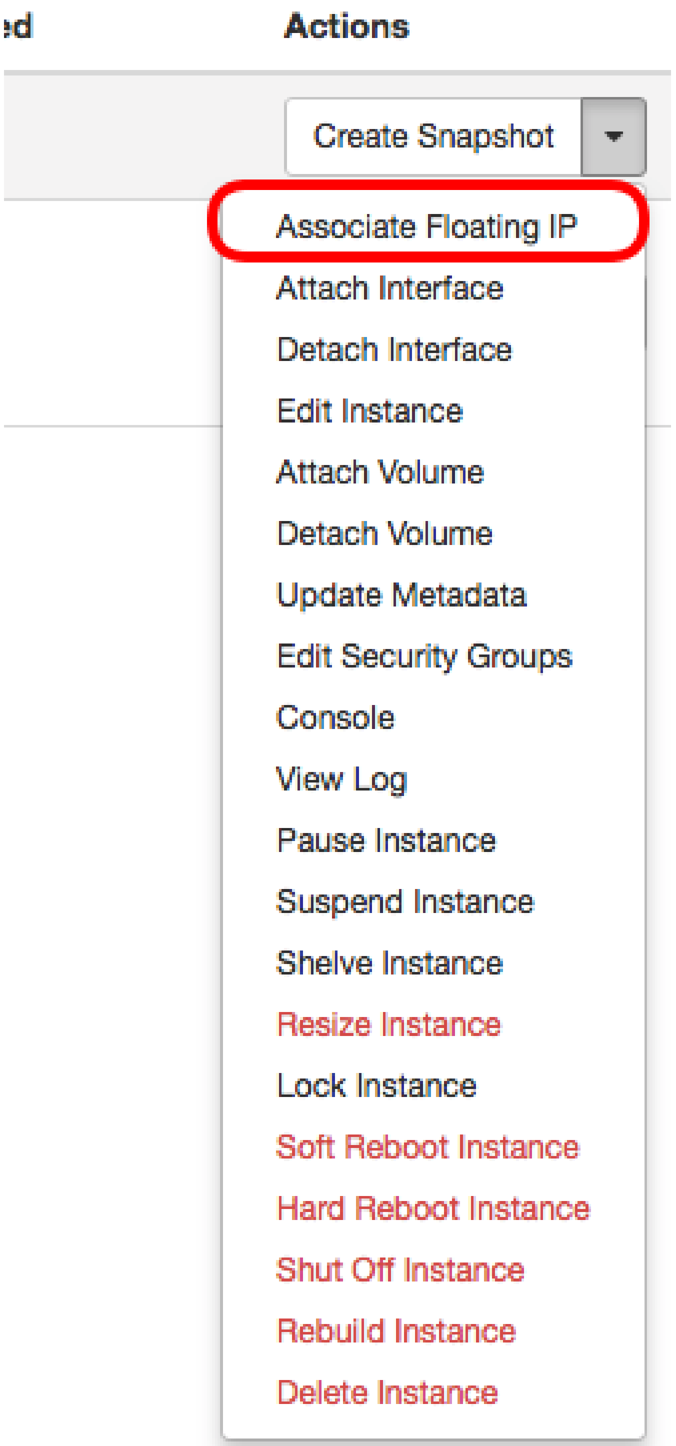


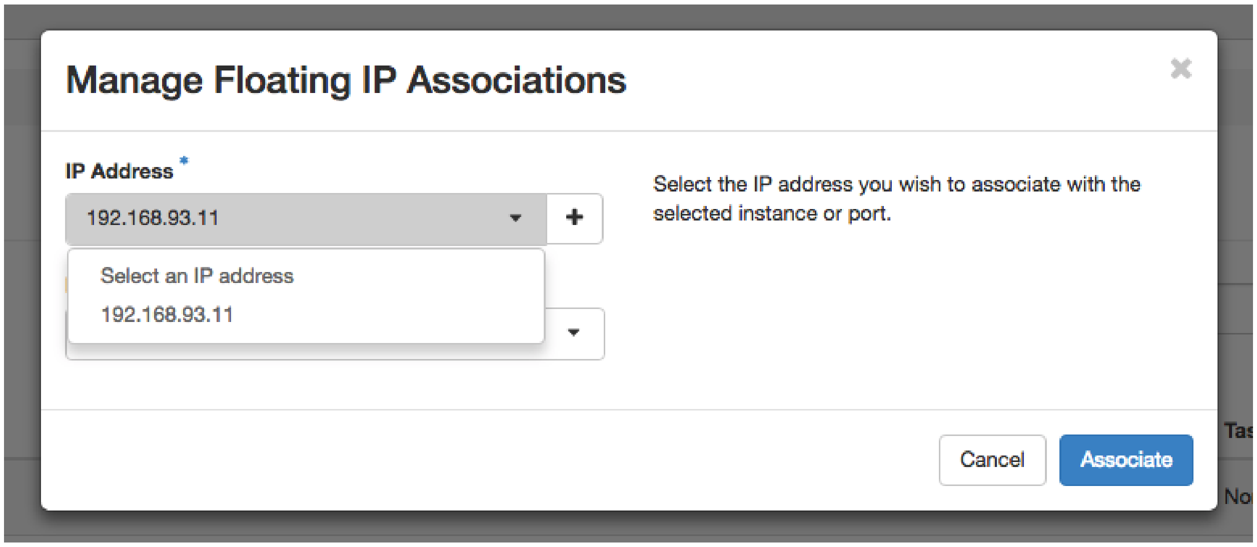
Instances

Displaying 2 items

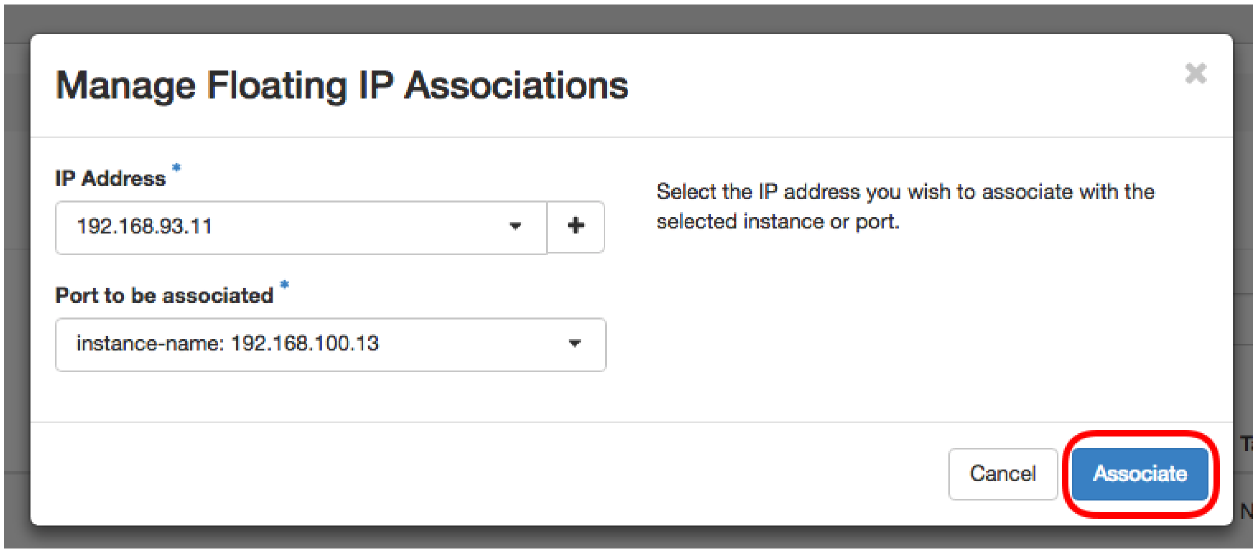
Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
instance-name	-	192.168.100.13	m1.medium	-	Active	nova	None	Running	0 minutes	Create Snapshot
			big		Active	nova	None	Running	4 months, 1 week	Create Snapshot

Displaying 2 items

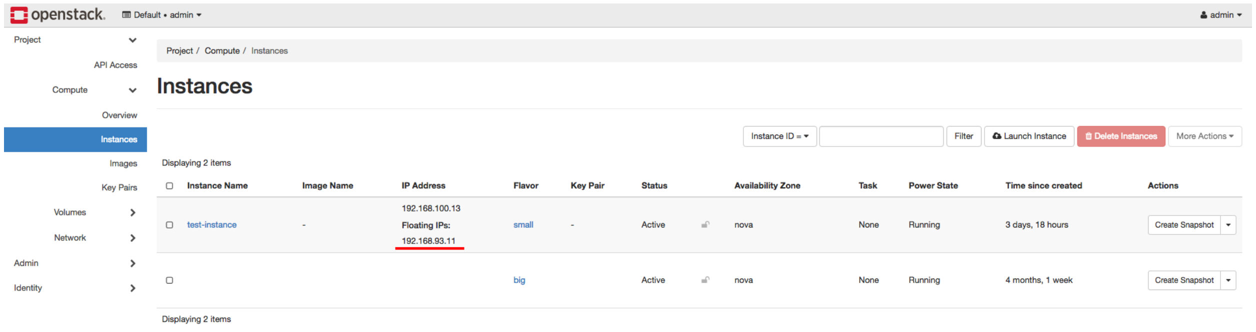

ed
Actions
Create Snapshot
Associate Floating IP
Attach Interface
Detach Interface
Edit Instance
Attach Volume
Detach Volume
Update Metadata
Edit Security Groups
Console
View Log
Pause Instance
Suspend Instance
Shelve Instance
Resize Instance
Lock Instance
Soft Reboot Instance
Hard Reboot Instance
Shut Off Instance
Rebuild Instance
Delete Instance



Select the IP address you wish to associate with the selected instance or port.



Select the IP address you wish to associate with the selected instance or port.



Displaying 2 items

	Instance Name	Image Name	IP Address	Flavor	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
☐	test-instance	-	192.168.100.13 Floating IPs: 192.168.93.11	small	-	Active	nova	None	Running	3 days, 18 hours	Create Snapshot
☐				big		Active	nova	None	Running	4 months, 1 week	Create Snapshot

Displaying 2 items

- Now using any SSH client connect to the instance through VPN using the Floating IP address

Docker machine deployment

Official docker-machine documentation: https://docs.docker.com/machine/overview/

1. Instalation

Guide: https://docs.docker.com/machine/install-machine/

2. Configuration

In order to use the OpenStack integration you need to export OpenStack Authentication credentials.

For the future use, create an executable file which will export environmental variables automatically. For example you can call file "openstackrc" and the content of the file be:

```
# VM CONFIG
export OS_SSH_USER=ubuntu
export OS_IMAGE_NAME=Ubuntu1604
export OS_FLAVOR_NAME=m1.medium
export OS_FLOATINGIP_POOL=ext_net
export OS_SECURITY_GROUPS=default
export OS_NETWORK_NAME=int_net

# AUTH
export OS_DOMAIN_NAME=default
export OS_USERNAME=<OPENSTACK_USER>
export OS_PASSWORD=<OPENSTACK_PASS>
export OS_TENANT_NAME=geral
export OS_AUTH_URL=http://192.168.93.215:5000/v3
```

OS_SSH_USER Default ssh user, usually it is ubuntu (if operating system is ubuntu)

OS_IMAGE_NAME OS image to be used during virtual machine deployment

OS_FLAVOR_NAME Virtual machine specification (vCPU, RAM, storage, . . .)

Flavor	vCPU	Root Disk	RAM
m1.tiny	1	0	0.5GB
m1.smaller	1	0	1GB
m1.small	1	10GB	2GB
m1.medium	2	10GB	3GB
m1.large	4	10GB	8GB
m1.xlarge	8	10GB	8GB
ska1.full	46	10GB	450GB

OS_FLOATINGIP_POOL Floating IP external network pool is the "ext_net"

OS_SECURITY_GROUPS Security groups, default is "default"

OS_NETWORK_NAME Private network, default is "int_net"

OS_DOMAIN_NAME OpenStack domain region, default is "default"

OS_USERNAME OpenStack username

OS_PASSWORD OpenStack password

OS_TENANT_NAME OpenStack project name, default is "geral"

OS_AUTH_URL OpenStack Auth URL, default is "http://192.168.93.215:5000/v3"

3. Usage

Complete documentation about docker-machine CLI commands can be found here: https://docs.docker.com/machine/reference/

3.1 Run the enviromental variable file

```
$ . openstackrc
```

3.2 Create docker-machine

Create a machine. Requires the –driver flag to indicate which provider (OpenStack) the machine should be created on, and an argument to indicate the name of the created machine.

```
$ docker-machine create --driver=openstack MACHINE-NAME

Creating CA: /root/.docker/machine/certs/ca.pem
Creating client certificate: /root/.docker/machine/certs/cert.pem
Running pre-create checks...
Creating machine...
(MACHINE-NAME) Creating machine...
Waiting for machine to be running, this may take a few minutes...
Detecting operating system of created instance...
Waiting for SSH to be available...
Detecting the provisioner...
Provisioning with ubuntu(systemd)...
Installing Docker...
Copying certs to the local machine directory...
Copying certs to the remote machine...
Setting Docker configuration on the remote daemon...
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on this virtual␣
↪machine, run: docker-machine env MACHINE-NAME
```

In this step docker-machine will create VM inside OpenStack. As soon as the ssh connection to VM is available the Docker service will be installed.

3.3 Set docker-machine environment

Set environment variables to dictate that docker should run a command against a particular machine.

```
$ docker-machine env MACHINE-NAME

export DOCKER_TLS_VERIFY="1"
export DOCKER_HOST="tcp://192.168.93.23:2376"
export DOCKER_CERT_PATH="/root/.docker/machine/machines/MACHINE-NAME"
export DOCKER_MACHINE_NAME="MACHINE-NAME"
# Run this command to configure your shell:
# eval $(docker-machine env MACHINE-NAME)
```

3.4 Configure shell to use your docker-machine

After this, when you execute "docker" command it will be executed remotely

```
$ eval $(docker-machine env MACHINE-NAME)
```

Now if you run "docker-machine ls" you see that your machine is active and ready to use.

```
$ docker-machine ls

NAME            ACTIVE    DRIVER       STATE     URL                            SWARM  ␣
↪DOCKER      ERRORS
MACHINE-NAME    *         openstack    Running   tcp://192.168.93.23:2376               v18.
↪09.0
```

3.5 Use "docker" command to remotely deploy docker containers

```
$ docker run -d -p 80:80 nginx

Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
a5a6f2f73cd8: Pull complete
67da5fbcb7a0: Pull complete
e82455fa5628: Pull complete
Digest: sha256:98b06873ea9c87d5df1bb75b650926cfbcc4c53f675dfabb158830af0b115f99
Status: Downloaded newer image for nginx:latest
889a1ab275ba072980fe4fd3ec58094513cf41330c3698b226c239ba490a24a6
```

3.6 Remove docker-machine

Remove a machine. This removes the local reference and deletes it on the cloud provider or virtualization management platform.

```
$ docker-machine rm MACHINE-NAME (-f if need force)
```

3.7 Docker-machine IP

Get the IP address of one or more machines

```
$ docker-machine ip MACHINE-NAME

192.168.93.23
```

3.8 Docker-machine list

List currently deployed docker-machines

```
$ docker-machine ls

NAME            ACTIVE   DRIVER      STATE     URL                           SWARM  ␣
↪DOCKER     ERRORS
MACHINE-NAME    *        openstack   Running   tcp://192.168.93.23:2376             v18.
↪09.0
```

3.9 Docker-machine upgrade

Upgrade a machine to the latest version of Docker. How this upgrade happens depends on the underlying distribution used on the created instance.

```
$ docker-machine upgrade MACHINE-NAME

Waiting for SSH to be available...
Detecting the provisioner...
Upgrading docker...
Restarting docker...
```

3.10 Docker-machine stop

Stops running docker-machine

```
$ docker-machine stop MACHINE-NAME

Stopping "MACHINE-NAME"...
Machine "MACHINE-NAME" was stopped.
```

3.11 Docker-machine restart

Restarts docker-machine

```
$ docker-machine restart MACHINE-NAME

Restarting "MACHINE-NAME"...
Waiting for SSH to be available...
Detecting the provisioner...
Restarted machines may have new IP addresses. You may need to re-run the `docker-
↪machine env` command.
```

3.12 Docker-machine start

Starts docker-machine

```
$ docker-machine start MACHINE-NAME

Starting "MACHINE-NAME"...
Machine "MACHINE-NAME" was started.
Waiting for SSH to be available...
```

(continues on next page)

(continued from previous page)

```
Detecting the provisioner...
Started machines may have new IP addresses. You may need to re-run the `docker-
↪machine env` command.
```

3.13 Docker-machine ssh

Log into or run a command on a machine using SSH.

```
$ docker-machine ssh MACHINE-NAME

Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-116-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:     https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage

Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud

153 packages can be updated.
81 updates are security updates.

New release '18.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


ubuntu@MACHINE-NAME:~$
```

6.1.5 Access to the bare metal

In this stage, this option is very restrictive and only in a well-justified situation is allowed.

CHAPTER 7

Developer Services

- *AIT cluster*

CHAPTER 8

Commitment to opensource

As defined in SKA software standard, SKA software development is committed to opensource, and an open source licensing model is always preferred within SKA software development.

Todo:

- Committment to opensource
- Exceptions