



ESCAPE

European Science Cluster of Astronomy &
Particle physics ESFRI research Infrastructures

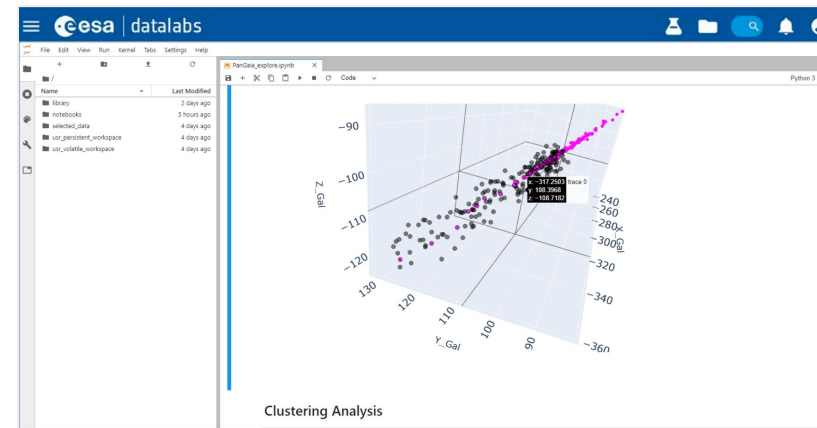
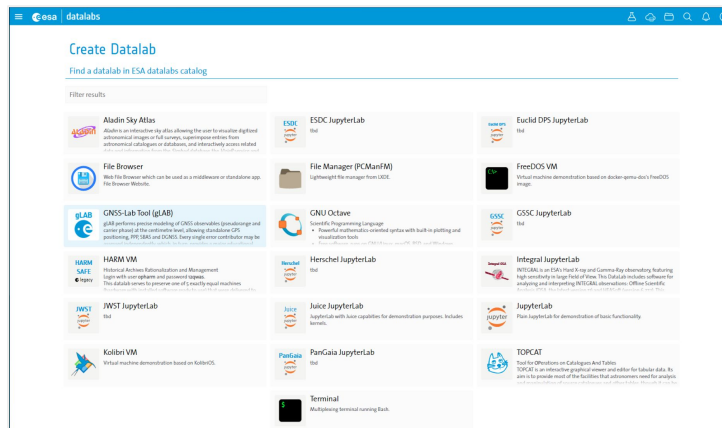
Rosetta: a container-centric science platform for interactive, resource intensive data analysis

Stefano Alberto Russo - INAF OATS



What is a science platform?

“A science platform is an environment designed to offer users a smoother experience when interacting with computing and storage resources”



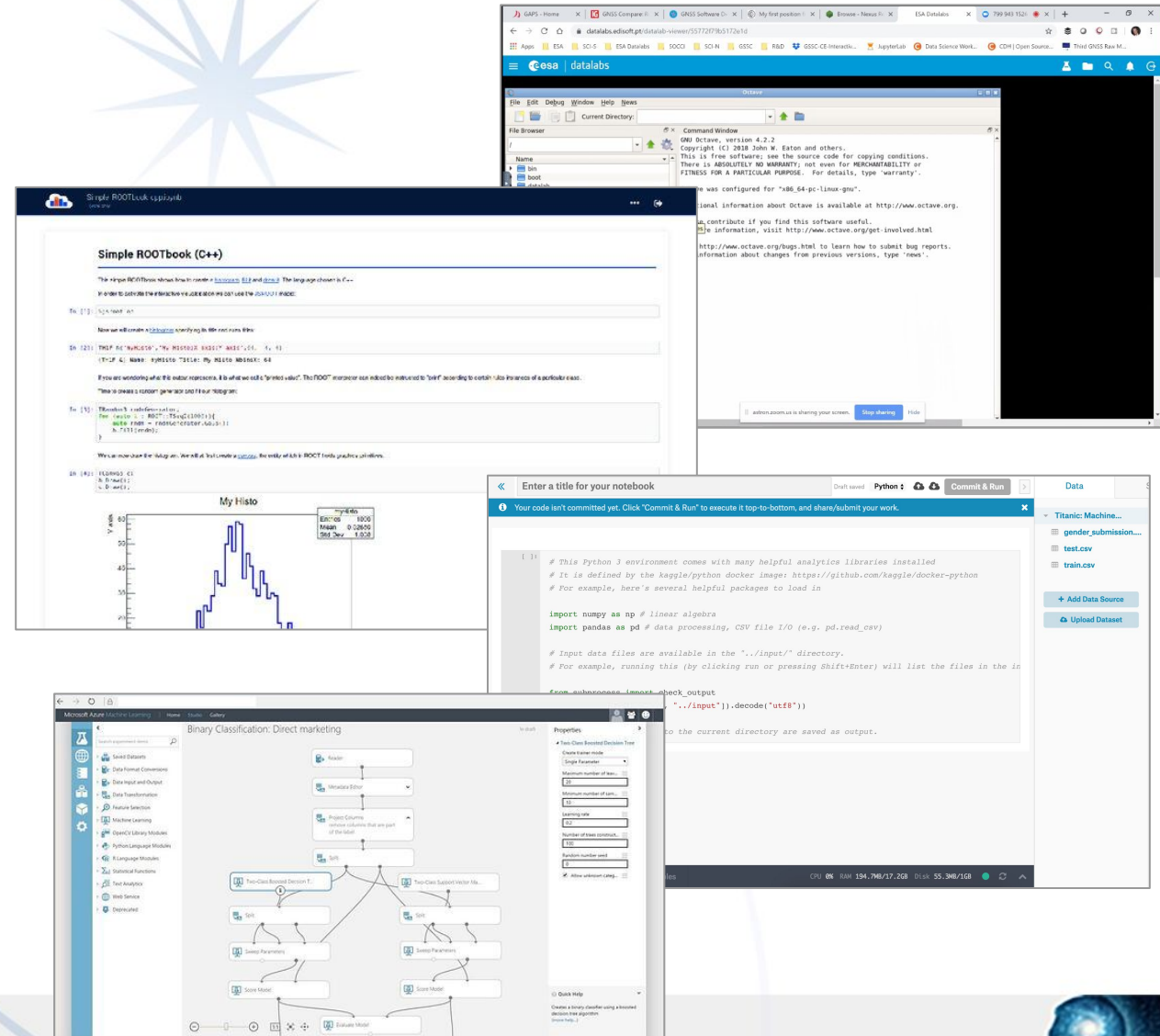
Some examples

Public Research/ Academia:

- CERN SWAN
- ESA Datalabs
- LSST Science Platform
- ...

Private Research / Industry:

- Google Colabs
- Kaggle Kernels
- Azure Machine Learning Studio
- ...



Limitations

There are some strong limitations with the current science platforms:

1. A user cannot run in an environment not supported by the platform
→ from a different Python version to a different Linux distribution.
→ this is a reproducibility issue!!
2. A user is tight to a specific interface, with little options for other ones
→ a web-based notebook (Jupyter or similar) interface makes it impossible to run native GUI applications (common in Astrophysics).
3. Poor HPC support, hard to integrate
→ the focus is usually more on making the data accessible rather than to run resource-intensive analysis



Introducing Rosetta

- A web-based science platform supporting both pre-defined and custom analysis environments
- Executes user tasks in software containers on:
 - local CPUs;
 - standalone computing resources
 - HPC clusters and Cloud systems
- Microservice-oriented approach: each task runs in its own container and expose its interface directly to the user:
 - *Users can choose from different environments and interfaces*
 - *Users can add their own containers*




Introducing Rosetta

- A web-based science platform supporting both pre-defined and custom analysis environments
- Executes user tasks in software containers on:
 - local CPUs;
 - standalone computing resources
 - HPC clusters and Cloud systems
- Microservice-oriented approach: each task runs in its own container and expose its interface directly to the user:
 - *Users can choose from different environments and interfaces*
 - *Users can add their own containers*

**Now integrated
with ESAP!**



Rosetta main page

Rosetta 

A container-centric Science Platform

Welcome to Rosetta!

This is the INAF OATs Rosetta science platform, which gives you simplified access to the [HOTCAT Cluster](#) computing and storage resources. For the accredited users, also Big Beauty is supported.

You can either signup as a generic user or login using your INAF credentials using Open ID Connect (RAP)

After signing up, you are required to copy your Rosetta public SSH key in the `~/.ssh/authorized_keys` the account of the computing resource you want to use. For the HOTCAT Cluster, this means adding it from cluster front-end node: Amonra. You can find your Rosetta public SSH key in your [account](#) section.

After this, you need to add an extra configuration parameter to your account profile specifying your Amonra login. Again from the [account](#) section, click on "Add new..." under the "extra configurations" tab, then choose `computing_user` as configuration type, and enter your Amonra user.

You can then choose the [software container](#) you want and execute it. You can also browse your files on the shared filesystem (`/beegfs`) using the [file manager](#).

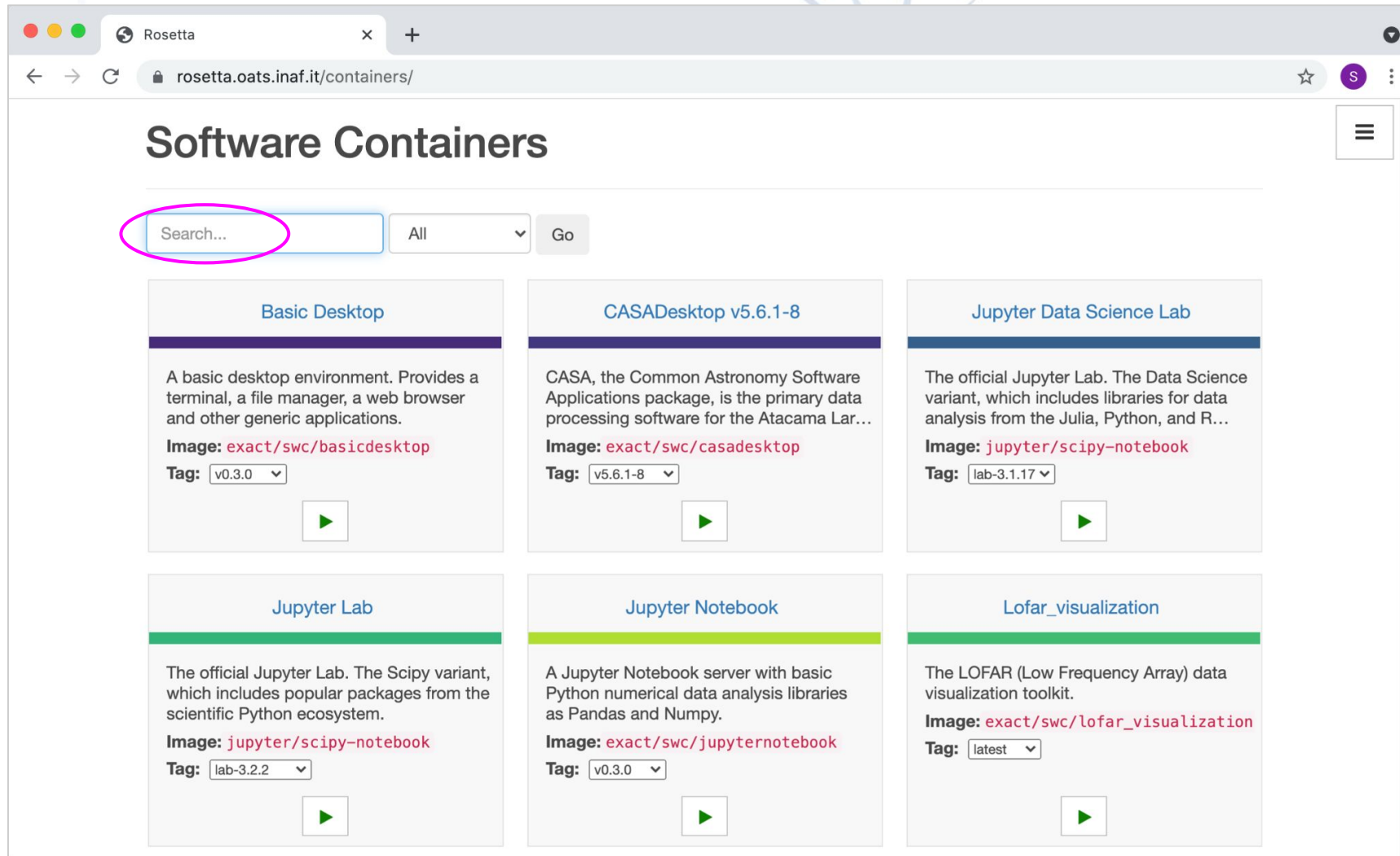
This work was supported by the European Science Cluster of Astronomy and Particle Physics ESFRI Research Infrastructures project funded by the European Union's Horizon 2020 research and innovation programme under Grant Agreement no. 824064.

Menu

- Software
- Computing
- Storages
- Tasks
- Account



The software “catalogue”



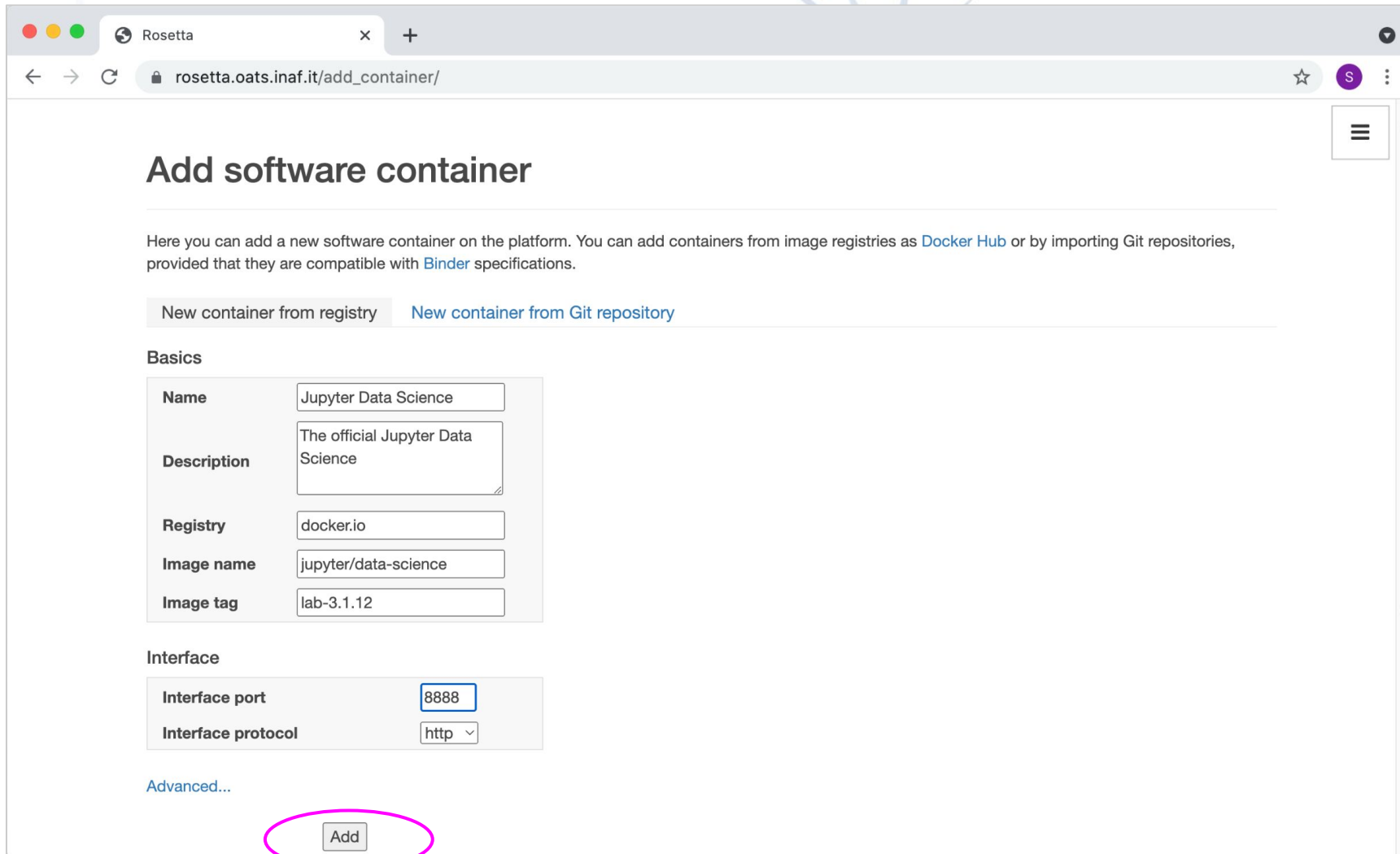
The screenshot shows a web browser window with the URL `rosetta.oats.inaf.it/containers/`. The page title is "Software Containers". At the top, there is a search bar (highlighted with a pink oval) containing the text "Search...", a dropdown menu set to "All", and a "Go" button. Below the search bar, there are six container cards arranged in a 2x3 grid:

- Basic Desktop**: A basic desktop environment. Provides a terminal, a file manager, a web browser and other generic applications. **Image:** `exact/swc/basicdesktop` **Tag:** `v0.3.0`
- CASADesktop v5.6.1-8**: CASA, the Common Astronomy Software Applications package, is the primary data processing software for the Atacama Lar... **Image:** `exact/swc/casadesktop` **Tag:** `v5.6.1-8`
- Jupyter Data Science Lab**: The official Jupyter Lab. The Data Science variant, which includes libraries for data analysis from the Julia, Python, and R... **Image:** `jupyter/scipy-notebook` **Tag:** `lab-3.1.17`
- Jupyter Lab**: The official Jupyter Lab. The Scipy variant, which includes popular packages from the scientific Python ecosystem. **Image:** `jupyter/scipy-notebook` **Tag:** `lab-3.2.2`
- Jupyter Notebook**: A Jupyter Notebook server with basic Python numerical data analysis libraries as Pandas and Numpy. **Image:** `exact/swc/jupyternotebook` **Tag:** `v0.3.0`
- Lofar_visualization**: The LOFAR (Low Frequency Array) data visualization toolkit. **Image:** `exact/swc/lofar_visualization` **Tag:** `latest`

Each card includes a description, the image name, the tag, and a green play button icon.



Adding new software [from container registries]



The screenshot shows a web browser window with the URL `rosetta.oats.inaf.it/add_container/`. The page title is "Add software container". Below the title, there is a paragraph explaining that users can add containers from image registries like Docker Hub or by importing Git repositories, provided they are compatible with Binder specifications. There are two tabs: "New container from registry" (selected) and "New container from Git repository".

Basics

Name	<input type="text" value="Jupyter Data Science"/>
Description	<input type="text" value="The official Jupyter Data Science"/>
Registry	<input type="text" value="docker.io"/>
Image name	<input type="text" value="jupyter/data-science"/>
Image tag	<input type="text" value="lab-3.1.12"/>

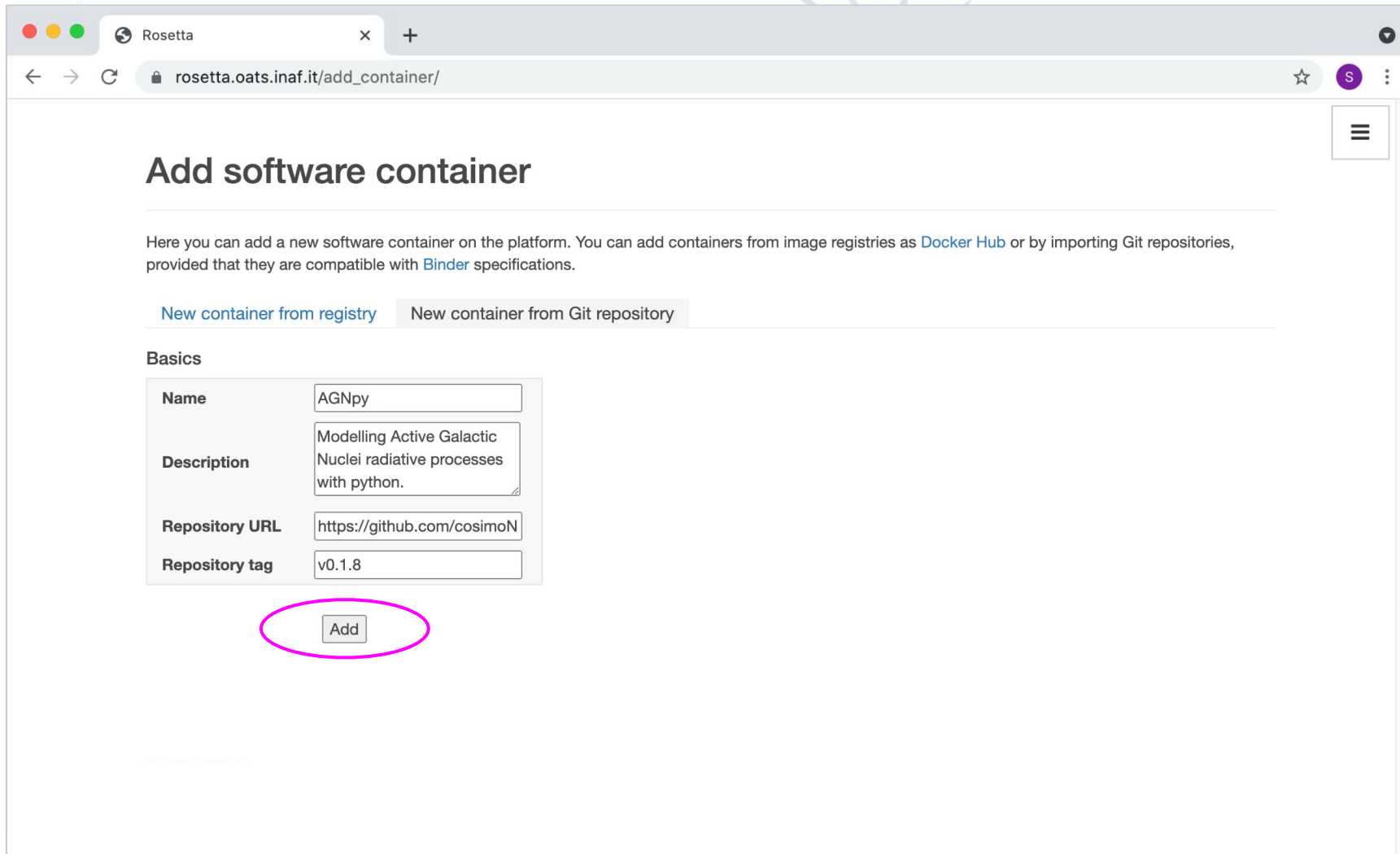
Interface

Interface port	<input type="text" value="8888"/>
Interface protocol	<input type="text" value="http"/>

[Advanced...](#)



Adding new software [Binder compatibility]



The screenshot shows a web browser window with the URL `rosetta.oats.inaf.it/add_container/`. The page title is "Add software container". Below the title, there is a paragraph explaining that users can add containers from image registries like Docker Hub or by importing Git repositories. Two tabs are visible: "New container from registry" (selected) and "New container from Git repository". Under the "Basics" section, there are four input fields: "Name" (AGNpy), "Description" (Modelling Active Galactic Nuclei radiative processes with python.), "Repository URL" (https://github.com/cosimoN), and "Repository tag" (v0.1.8). A pink oval highlights the "Add" button at the bottom of the form.

Add software container

Here you can add a new software container on the platform. You can add containers from image registries as [Docker Hub](#) or by importing Git repositories, provided that they are compatible with [Binder](#) specifications.

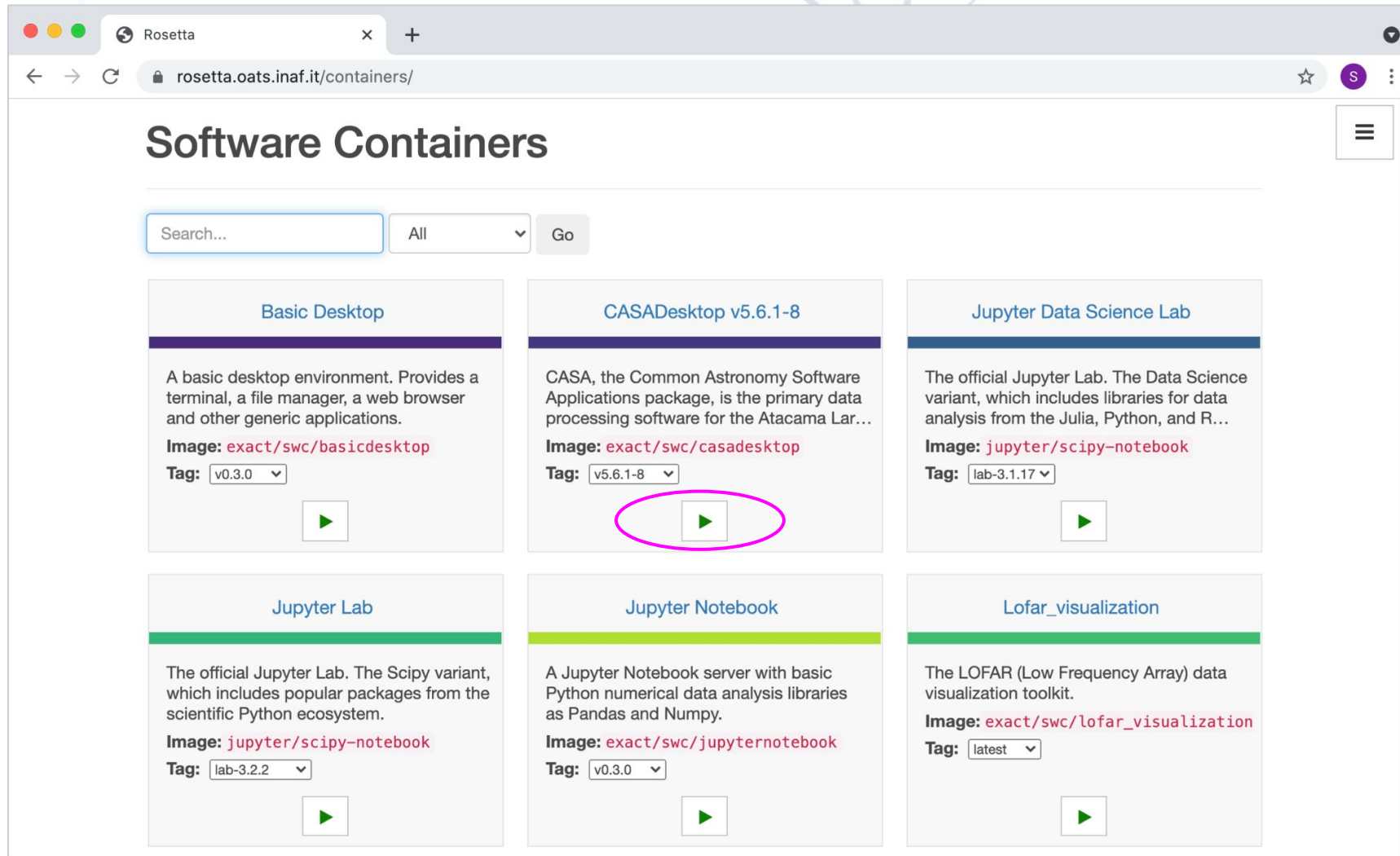
[New container from registry](#) [New container from Git repository](#)

Basics

Name	<input type="text" value="AGNpy"/>
Description	<input type="text" value="Modelling Active Galactic Nuclei radiative processes with python."/>
Repository URL	<input type="text" value="https://github.com/cosimoN"/>
Repository tag	<input type="text" value="v0.1.8"/>



Creating a new task

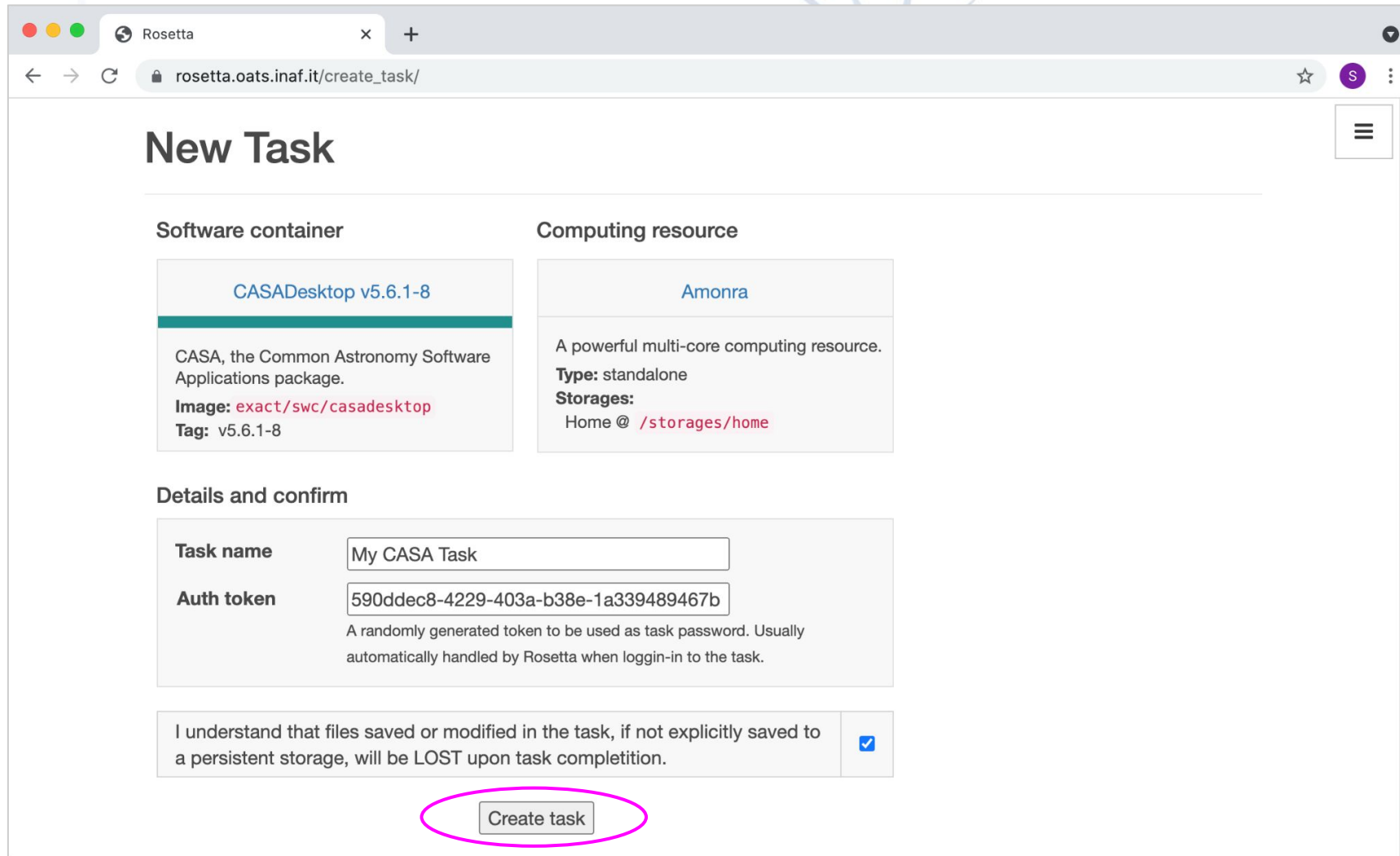


The screenshot shows a web browser window with the URL `rosetta.oats.inaf.it/containers/`. The page title is "Software Containers". At the top, there is a search bar and a filter dropdown set to "All". Below this, there are six container cards arranged in a 2x3 grid:

- Basic Desktop**: A basic desktop environment. Provides a terminal, a file manager, a web browser and other generic applications. Image: `exact/swc/basicdesktop`. Tag: `v0.3.0`.
- CASADesktop v5.6.1-8**: CASA, the Common Astronomy Software Applications package, is the primary data processing software for the Atacama Lar... Image: `exact/swc/casadesktop`. Tag: `v5.6.1-8`. The play button icon is circled in pink.
- Jupyter Data Science Lab**: The official Jupyter Lab. The Data Science variant, which includes libraries for data analysis from the Julia, Python, and R... Image: `jupyter/scipy-notebook`. Tag: `lab-3.1.17`.
- Jupyter Lab**: The official Jupyter Lab. The Scipy variant, which includes popular packages from the scientific Python ecosystem. Image: `jupyter/scipy-notebook`. Tag: `lab-3.2.2`.
- Jupyter Notebook**: A Jupyter Notebook server with basic Python numerical data analysis libraries as Pandas and Numpy. Image: `exact/swc/jupyternotebook`. Tag: `v0.3.0`.
- Lofar_visualization**: The LOFAR (Low Frequency Array) data visualization toolkit. Image: `exact/swc/lofar_visualization`. Tag: `latest`.



Creating a new task



The screenshot shows a web browser window with the URL `rosetta.oats.inaf.it/create_task/`. The page title is "New Task". It features two columns of options: "Software container" and "Computing resource".

Software container: CASADesktop v5.6.1-8. Description: CASA, the Common Astronomy Software Applications package. Image: `exact/swc/casadesktop`. Tag: `v5.6.1-8`.

Computing resource: Amonra. Description: A powerful multi-core computing resource. Type: standalone. Storages: Home @ `/storages/home`.

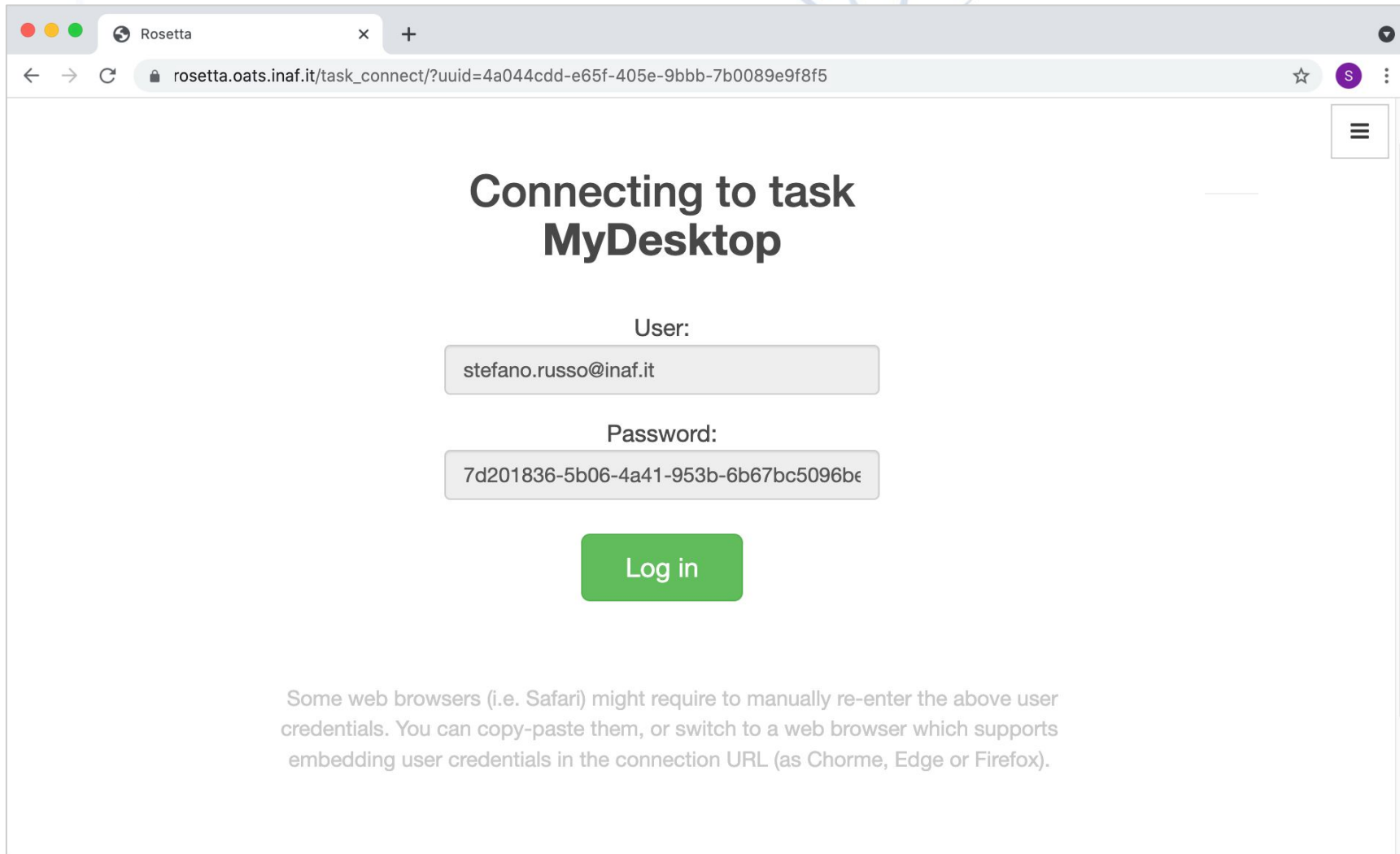
Details and confirm:

- Task name:
- Auth token:
A randomly generated token to be used as task password. Usually automatically handled by Rosetta when loggin-in to the task.

I understand that files saved or modified in the task, if not explicitly saved to a persistent storage, will be LOST upon task completion.



Creating a new task

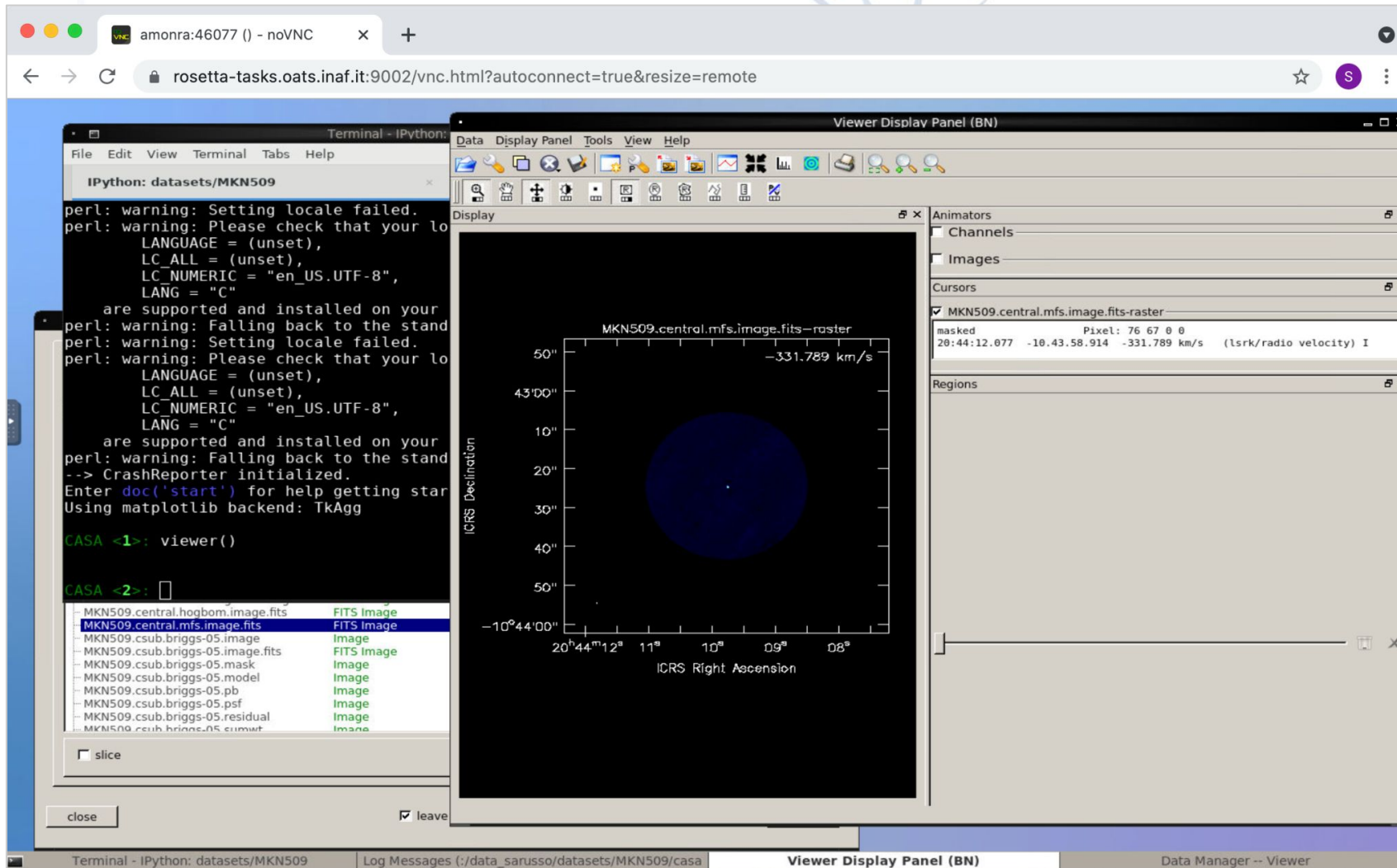


The screenshot shows a web browser window with the following elements:

- Browser tab: Rosetta
- Address bar: `rosetta.oats.inaf.it/task_connect/?uuid=4a044cdd-e65f-405e-9bbb-7b0089e9f8f5`
- Page title: Connecting to task MyDesktop
- User field: `stefano.russo@inaf.it`
- Password field: `7d201836-5b06-4a41-953b-6b67bc5096bc`
- Log in button: A green button with the text "Log in".
- Footer text: "Some web browsers (i.e. Safari) might require to manually re-enter the above user credentials. You can copy-paste them, or switch to a web browser which supports embedding user credentials in the connection URL (as Chrome, Edge or Firefox)."



A remote desktop task



The screenshot shows a remote desktop environment with the following components:

- Terminal - IPython:**

```

IPython: datasets/MKN509
perl: warning: Setting locale failed.
perl: warning: Please check that your locale
LANGUAGE = (unset),
LC_ALL = (unset),
LC_NUMERIC = "en_US.UTF-8",
LANG = "C"
are supported and installed on your
perl: warning: Falling back to the stand
perl: warning: Setting locale failed.
perl: warning: Please check that your lo
LANGUAGE = (unset),
LC_ALL = (unset),
LC_NUMERIC = "en_US.UTF-8",
LANG = "C"
are supported and installed on your
perl: warning: Falling back to the stand
--> CrashReporter initialized.
Enter doc('start') for help getting star
Using matplotlib backend: TkAgg

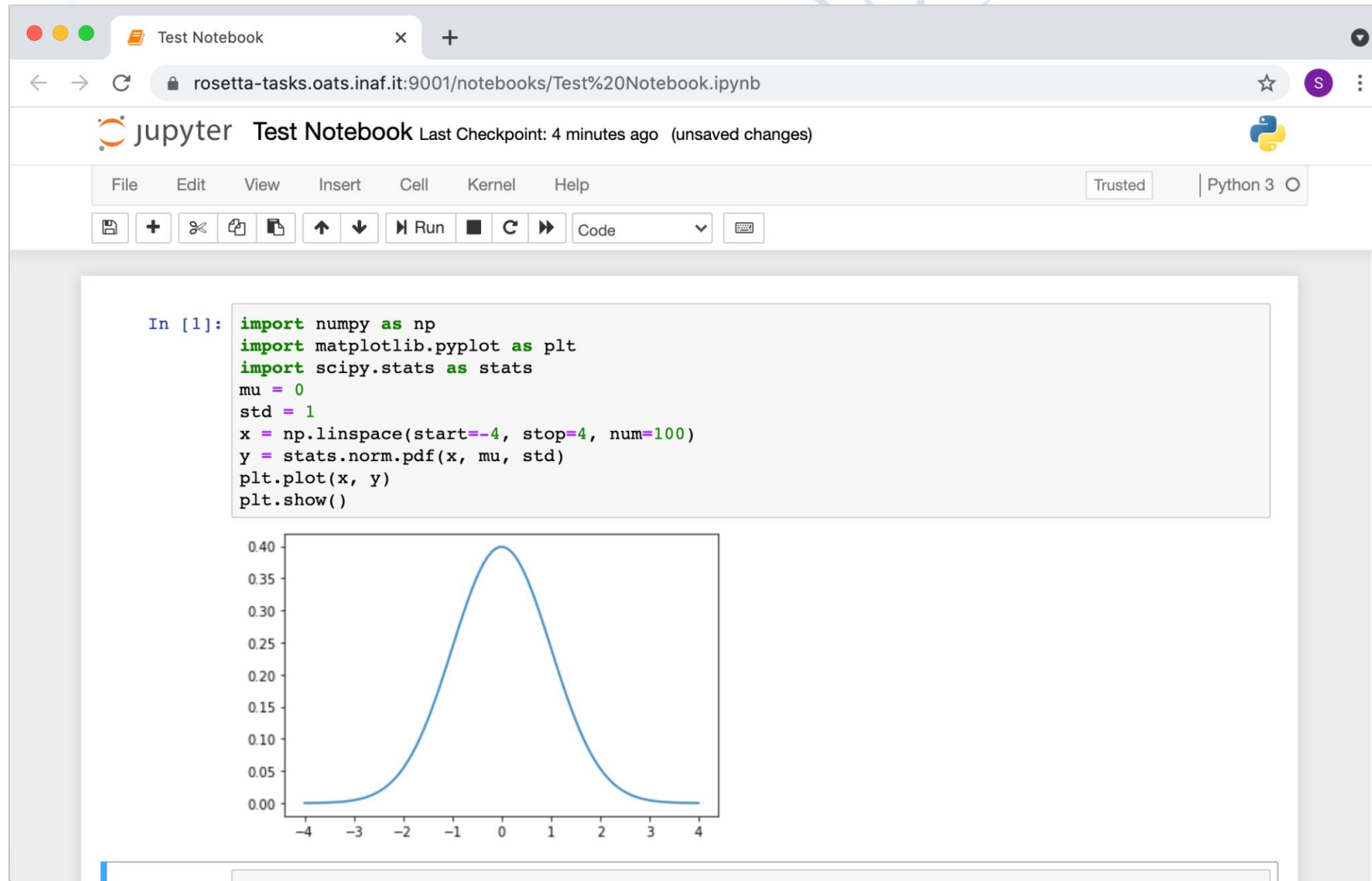
CASA <1>: viewer()

CASA <2>:
-MKN509.central.hogbom.image.fits      FITS Image
MKN509.central.mfs.image.fits        FITS Image
MKN509.csib.briggs-05.image          image
-MKN509.csib.briggs-05.image.fits    FITS Image
-MKN509.csib.briggs-05.mask          image
-MKN509.csib.briggs-05.model         image
-MKN509.csib.briggs-05.pb           image
-MKN509.csib.briggs-05.psf          image
-MKN509.csib.briggs-05.residual     image
-MKN509.csib.briggs-05.sumwt        image

```
- Viewer Display Panel (BN):**
 - Display: MKN509.central.mfs.image.fits-raster
 - ICRS Right Ascension: 20^h44^m12^s to 08^s
 - ICRS Declination: 43°00' to -10°44'00"
 - Velocity scale: -331.789 km/s
 - Animators: Channels, Images
 - Cursors: MKN509.central.mfs.image.fits-raster
 - masked Pixel: 76 67 0 0
 - 20:44:12.077 -10.43:58.914 -331.789 km/s (lsrk/radio velocity) I
 - Regions: (empty)



A Jupyter Notebook task



The screenshot shows a web browser window with a Jupyter Notebook titled "Test Notebook". The browser address bar shows the URL "rosetta-tasks.oats.inaf.it:9001/notebooks/Test%20Notebook.ipynb". The notebook interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Help), a "Trusted" status indicator, and "Python 3" as the selected kernel. Below the menu is a toolbar with icons for saving, adding, deleting, and running cells. The main content area displays a code cell with the following Python code:

```
In [1]: import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
mu = 0
std = 1
x = np.linspace(start=-4, stop=4, num=100)
y = stats.norm.pdf(x, mu, std)
plt.plot(x, y)
plt.show()
```

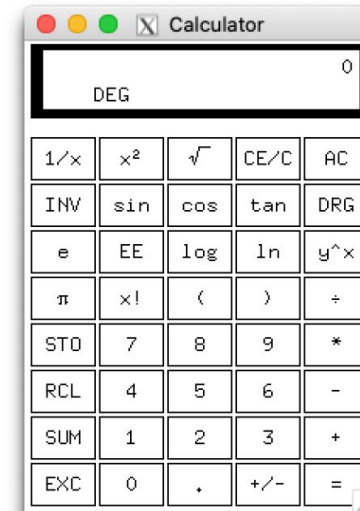
The output of the code cell is a plot of a normal distribution curve. The x-axis ranges from -4 to 4, and the y-axis ranges from 0.00 to 0.40. The curve is centered at 0 and reaches a maximum height of approximately 0.40.



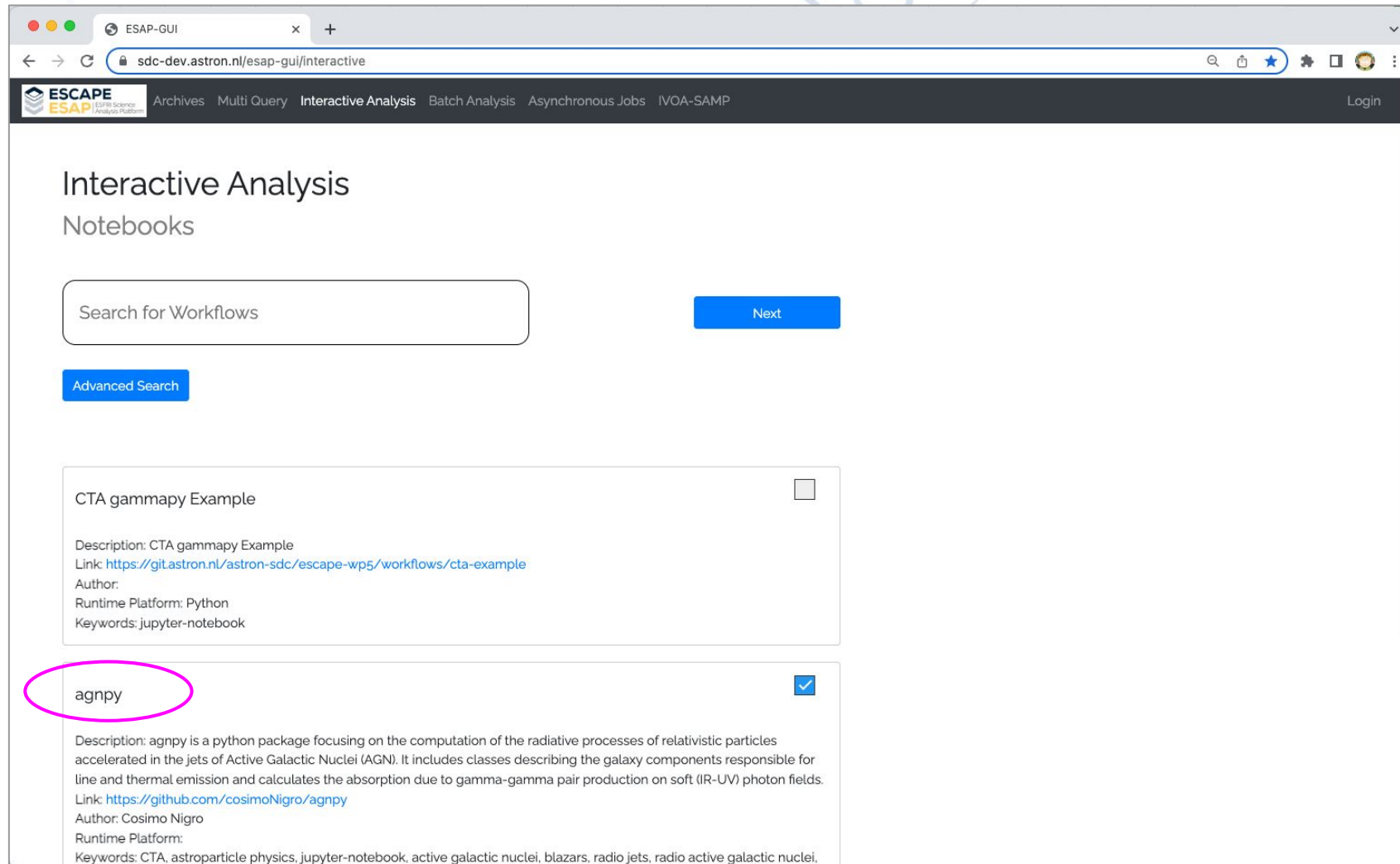
A SSH server task

```
ste — ssh -X -p7000 metauser@rosetta-tasks.oats.inaf.it — 116x36
ste@Stes-MacBookAir:~ $ ssh -X -p7000 metauser@rosetta-tasks.oats.inaf.it
metauser@rosetta-tasks.oats.inaf.it's password:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

metauser@task-0d378dfd:~$ pwd
/home/metauser
metauser@task-0d378dfd:~$ ls -alh
total 44K
drwxr-xr-x 6 metauser metauser 4.0K Nov 17 23:55 .
drwxrwxrwx 1 root      root      4.0K Nov 17 23:53 ..
-rw----- 1 metauser metauser   50 Nov 17 23:55 .Xauthority
-rw----- 1 metauser metauser  123 Nov 17 23:55 .bash_history
-rw-r--r-- 1 metauser metauser  220 Apr  4  2018 .bash_logout
-rw-r--r-- 1 metauser metauser  3.8K Nov 17 23:54 .bashrc
drwx----- 2 metauser metauser  4.0K Nov 17 23:53 .cache
-rw-r--r-- 1 metauser metauser    0 Nov 17 23:53 .initialized
drwxrwxr-x 3 metauser metauser  4.0K Nov 17 23:54 .local
drwxr-xr-x 2 metauser metauser  4.0K Nov  4 19:58 .logs
-rw-r--r-- 1 metauser metauser   807 Apr  4  2018 .profile
drwxr-xr-x 2 metauser metauser  4.0K Nov 17 23:53 custom_ssh
metauser@task-0d378dfd:~$ xcalc
█
```



ESAP Integration (1)



ESAP-GUI

sdc-dev.astron.nl/esap-gui/interactive

ESCAPE ESAP
ESFRI Science Analysis Platform

Archives Multi Query Interactive Analysis Batch Analysis Asynchronous Jobs IVOA-SAMP Login

Interactive Analysis Notebooks

Search for Workflows Next

Advanced Search

CTA gammapy Example

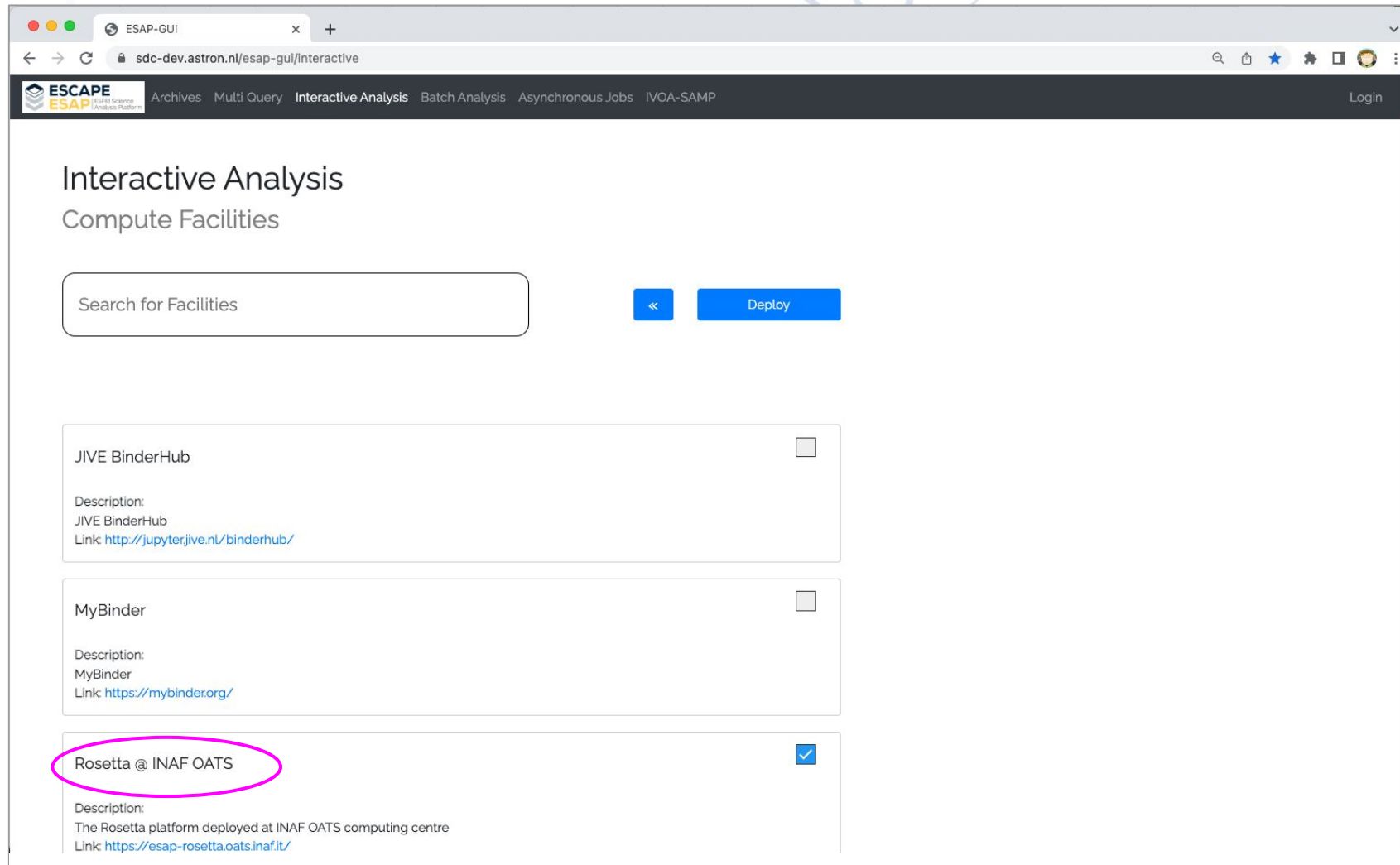
Description: CTA gammapy Example
Link: <https://git.astron.nl/astron-sdc/escape-wp5/workflows/cta-example>
Author:
Runtime Platform: Python
Keywords: jupyter-notebook

agnpy

Description: agnpy is a python package focusing on the computation of the radiative processes of relativistic particles accelerated in the jets of Active Galactic Nuclei (AGN). It includes classes describing the galaxy components responsible for line and thermal emission and calculates the absorption due to gamma-gamma pair production on soft (IR-UV) photon fields.
Link: <https://github.com/cosimoNigro/agnpy>
Author: Cosimo Nigro
Runtime Platform:
Keywords: CTA, astroparticle physics, jupyter-notebook, active galactic nuclei, blazars, radio jets, radio active galactic nuclei.



ESAP Integration (2)

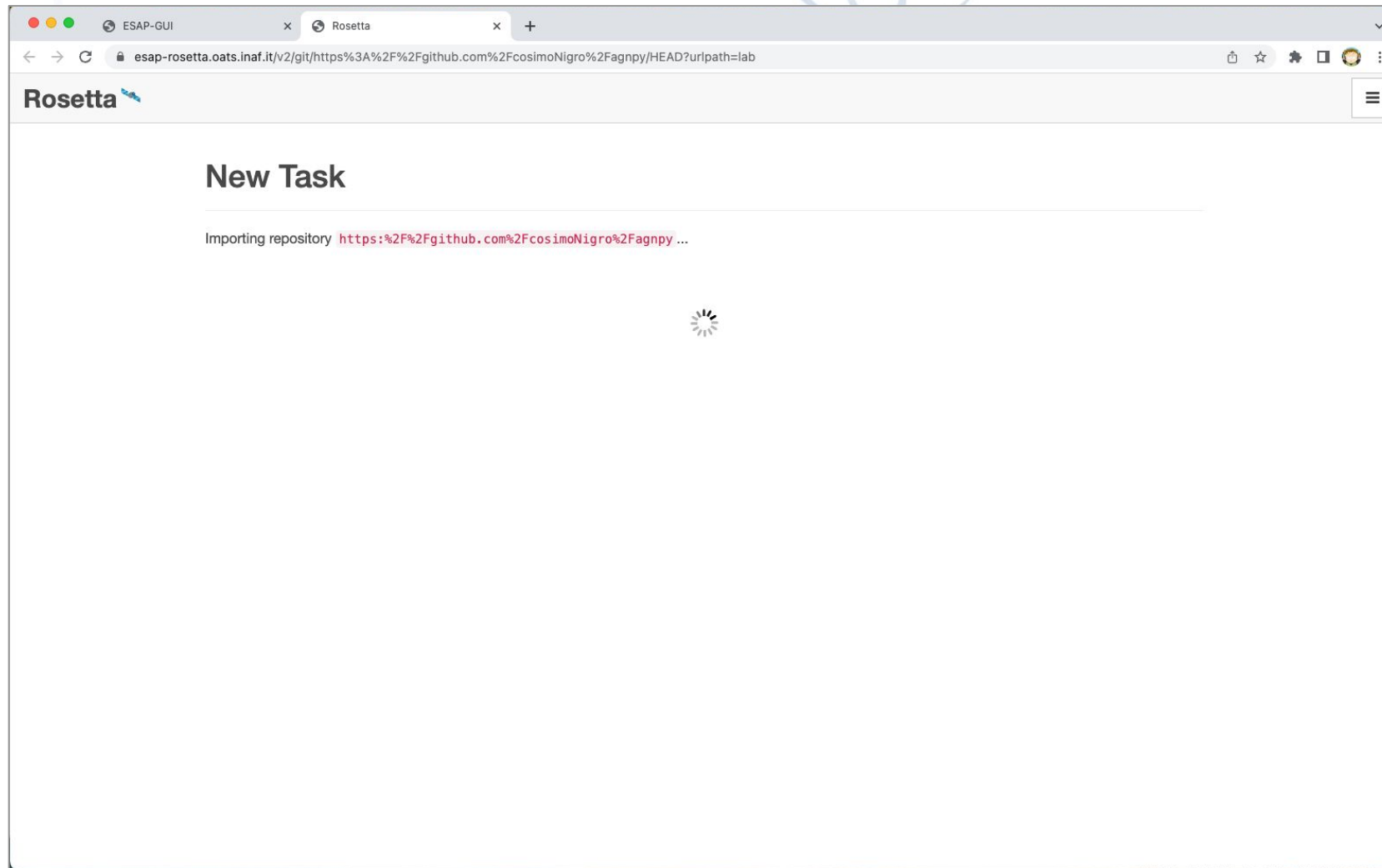


The screenshot shows a web browser window with the URL `sdsc-dev.astron.nl/esap-gui/interactive`. The page title is "Interactive Analysis" and the sub-section is "Compute Facilities". There is a search input field labeled "Search for Facilities" and a "Deploy" button. Below the search field, there is a list of three compute facilities:

- JIVE BinderHub** (checkbox unchecked)
Description: JIVE BinderHub
Link: <http://jupyterjive.nl/binderhub/>
- MyBinder** (checkbox unchecked)
Description: MyBinder
Link: <https://mybinder.org/>
- Rosetta @ INAF OATS** (checkbox checked, circled in pink)
Description: The Rosetta platform deployed at INAF OATS computing centre
Link: <https://esap-rosetta.oats.inaf.it/>



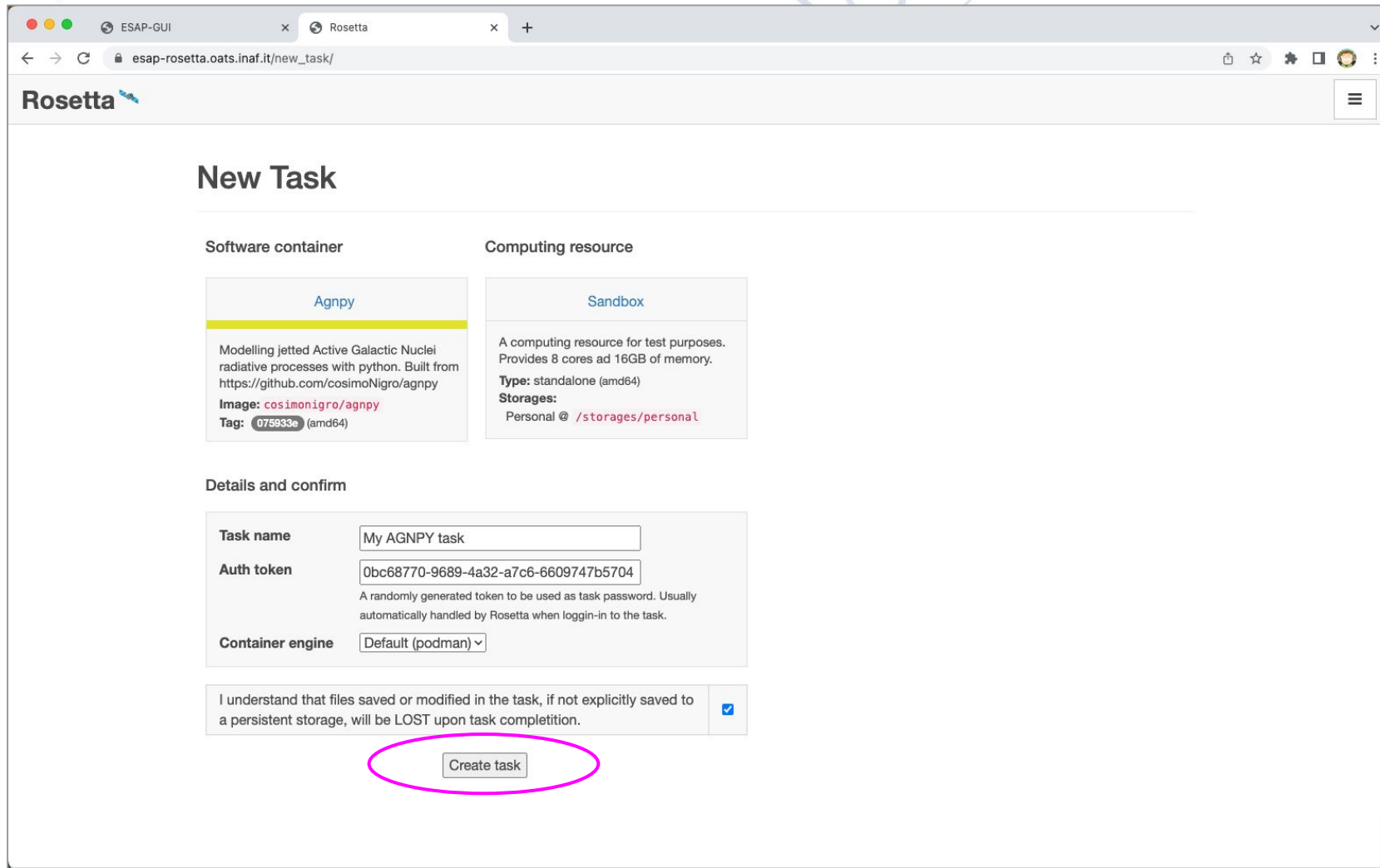
ESAP Integration (3)



The screenshot shows a web browser window with two tabs: 'ESAP-GUI' and 'Rosetta'. The address bar shows the URL: `esap-rosetta.oats.inaf.it/v2/git/https%3A%2F%2Fgithub.com%2FcosimoNigro%2Fagnpy/HEAD?urlpath=lab`. The page title is 'Rosetta'. The main content area displays 'New Task' followed by a horizontal line and the text 'Importing repository <https://github.com/cosimoNigro/agnpy> ...'. A loading spinner is centered below the text.



ESAP Integration (4)



The screenshot shows a web browser window with the URL `esap-rosetta.oats.inaf.it/new_task/`. The page title is "Rosetta". The main heading is "New Task".

There are two columns of options:

- Software container:** "Agnpy" is selected. Description: "Modelling jetted Active Galactic Nuclei radiative processes with python. Built from <https://github.com/cosimoNigro/agnpy>. Image: `cosimonigro/agnpy`. Tag: `075933e` (amd64)".
- Computing resource:** "Sandbox" is selected. Description: "A computing resource for test purposes. Provides 8 cores ad 16GB of memory. Type: standalone (amd64). Storages: Personal @ `/storages/personal`".

Details and confirm

Task name:

Auth token:
A randomly generated token to be used as task password. Usually automatically handled by Rosetta when login-in to the task.

Container engine:

I understand that files saved or modified in the task, if not explicitly saved to a persistent storage, will be LOST upon task completion.



ESAP Integration (5)

The screenshot shows a web browser window with the URL `esap-rosetta.oats.inaf.it/tasks/`. The page title is "Rosetta". Under the heading "Tasks", there is a card for "My AGNPY task". The card displays the following information: "Software: Agnpy", "Computing: Sandbox", and "Status: running". At the bottom of the card, there are three buttons: "Stop", "Connect", and "logs". The "Connect" button is highlighted with a pink circle. Below the task card, there is a link that says "New task...".



ESAP Integration (6)

The screenshot shows a Jupyter Notebook window titled "check_tau_dt" with the following content:

```
R_dt= 1.565247584249853e+19 cm
```

Warning: /home/cosimo/software/miniconda3/lib/python3.7/site-packages/astropy/units/quantity.py:477: RuntimeWarning: invalid value encountered in sqrt
result = super().__array_ufunc__(function, method, *arrays, **kwargs)

```
In [4]: # the reference curve is multiplied by a factor 2 because this is missing in the emissivity definition of Finke
make_comparison_plot(
    nu_ref,
    2 * tau_ref,
    tau_agnp,
    "Figure 14, Finke (2016)",
    "agnpy",
    "Absorption on Ring Dust Torus",
    "./test_dt.png",
    "tau",
)
```

Figure: Absorption on Ring Dust Torus

The plot shows the absorption coefficient τ_{ν} on the y-axis (log scale from 10^{-2} to 10^2) versus frequency ν in Hz on the x-axis (log scale from 10^{27} to 10^{28}). Two data series are plotted: "Figure 14, Finke (2016)" (blue circles) and "agnpy" (orange dashed line). The "agnpy" curve is shifted to the right relative to the "Figure 14, Finke (2016)" curve. A legend indicates the ratio $1 - \tau_{\nu, \text{agnpy}} / \tau_{\nu, \text{reference}}$ is shown in orange.

```
In [16]: # back of envelope calculations, threshold
eps=dt.epsilon_dt * mec2 # energy of DT photons
#print(eps.to("eV"))
x0=np.sqrt(R_dt**2 + x**2)
```



Rosetta architecture

Rosetta architecture is based on two main ingredients:

- 1) Microservices
- 2) Software containers



Rosetta architecture: microservices

Microservices are independent and self-contained units that perform a given and well-defined task, using a well defined *interface*.

→ From just summing two numbers to running a neural network.

Microservices are completely decoupled from the underlying infrastructure and from each other. Encapsulation is maximum.

In Rosetta, each user task is framed as a microservice



Rosetta architecture: microservices

Examples of interfaces:

- REST APIs
- HTTP
- SSH
- RPC
- etc.



Rosetta architecture: containers

Software containers are lightweight, standalone, executable packages of software that includes everything needed to run an application:

→ code, runtime, system tools, system libraries and settings.

They are a solution to the problem of how to get software to run reliably when moved from one computing environment to another.

Rosetta uses them to wrap the user task microservices



Rosetta architecture: task orchestration

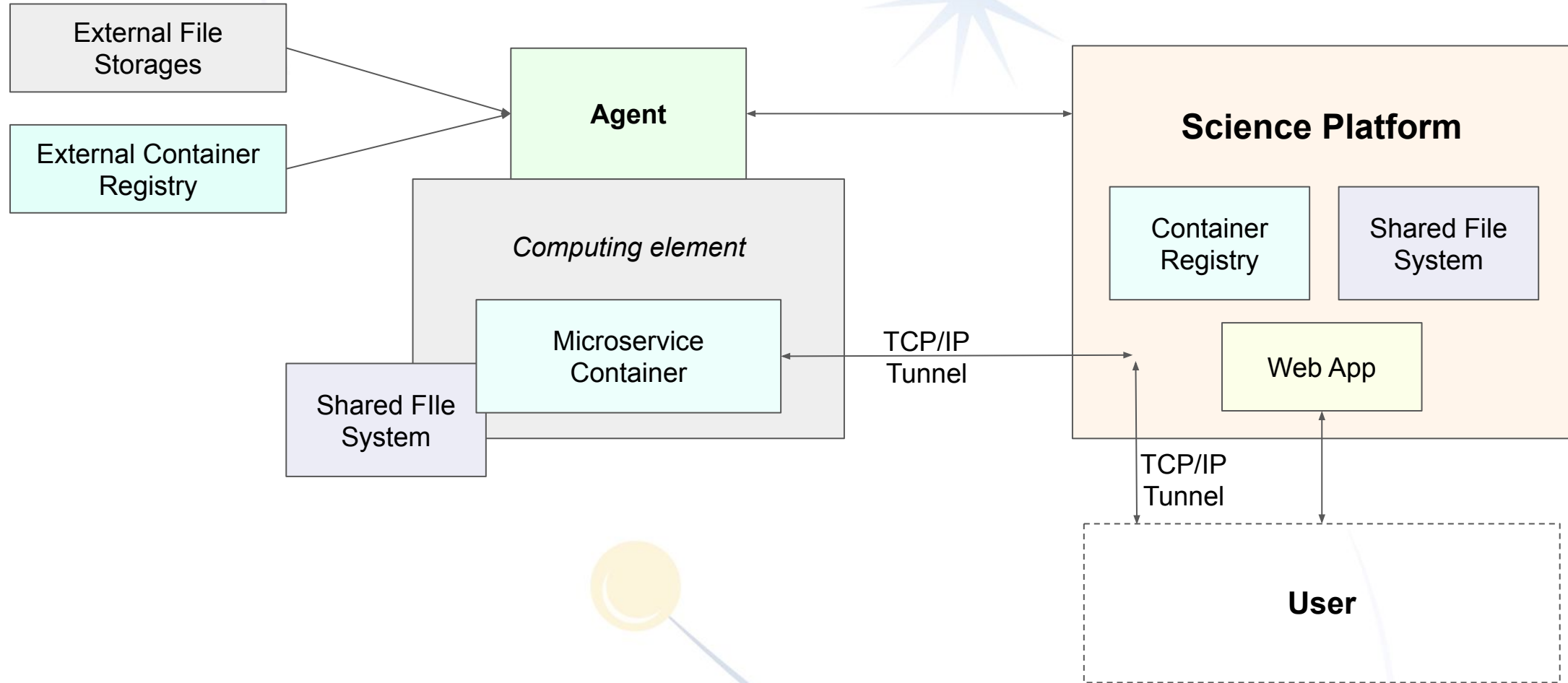
Task orchestration include:

- sending the task for execution
- setting up the all the tunneling required to reach its interfaces.

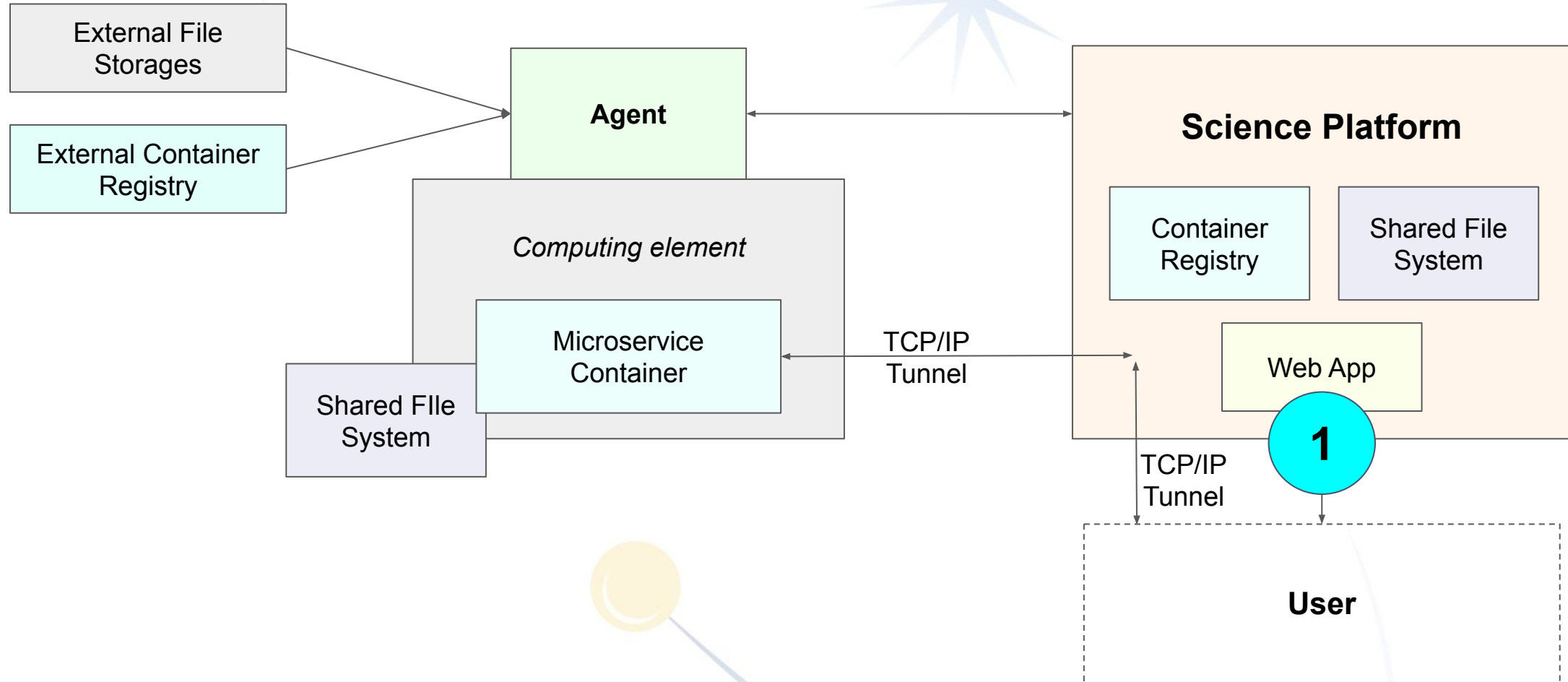
It can rely on the WMS or use an *agent*.



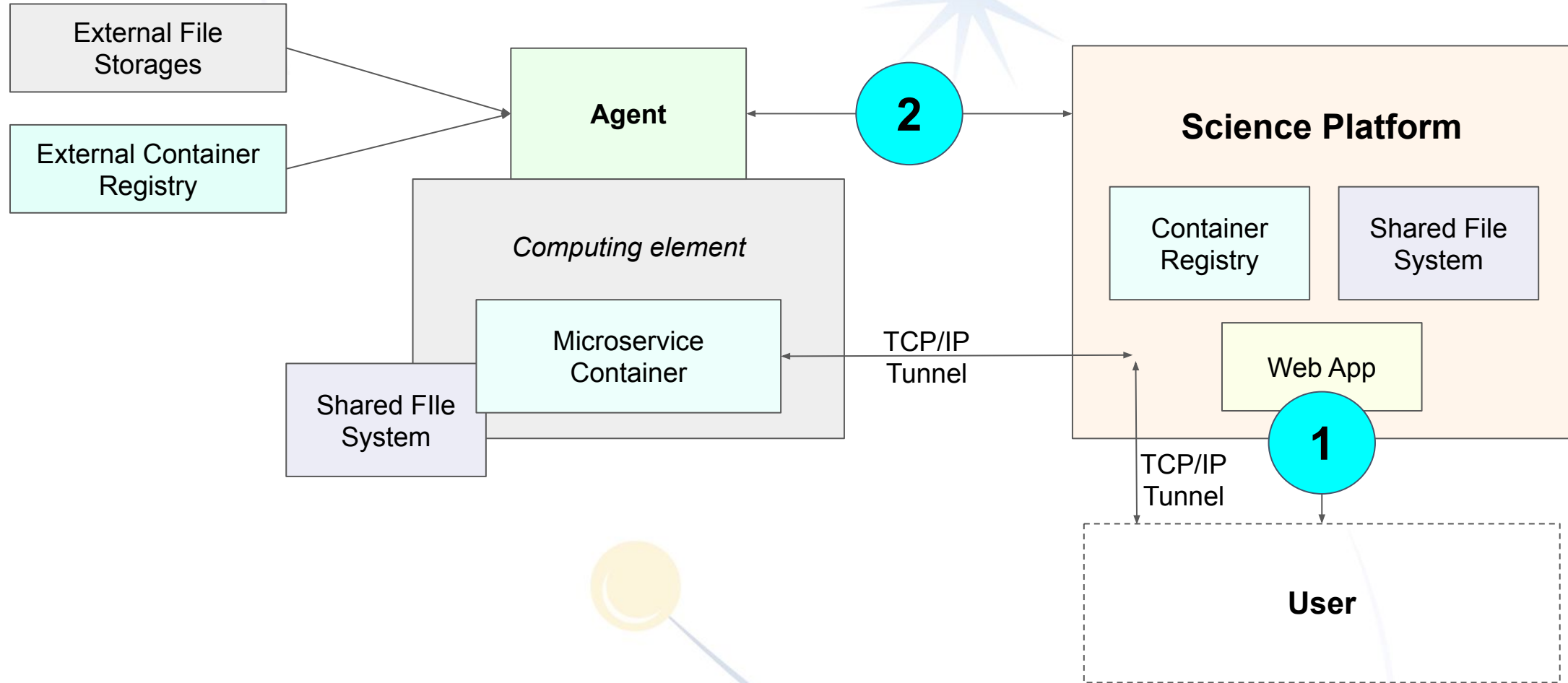
Rosetta architecture: task orchestration



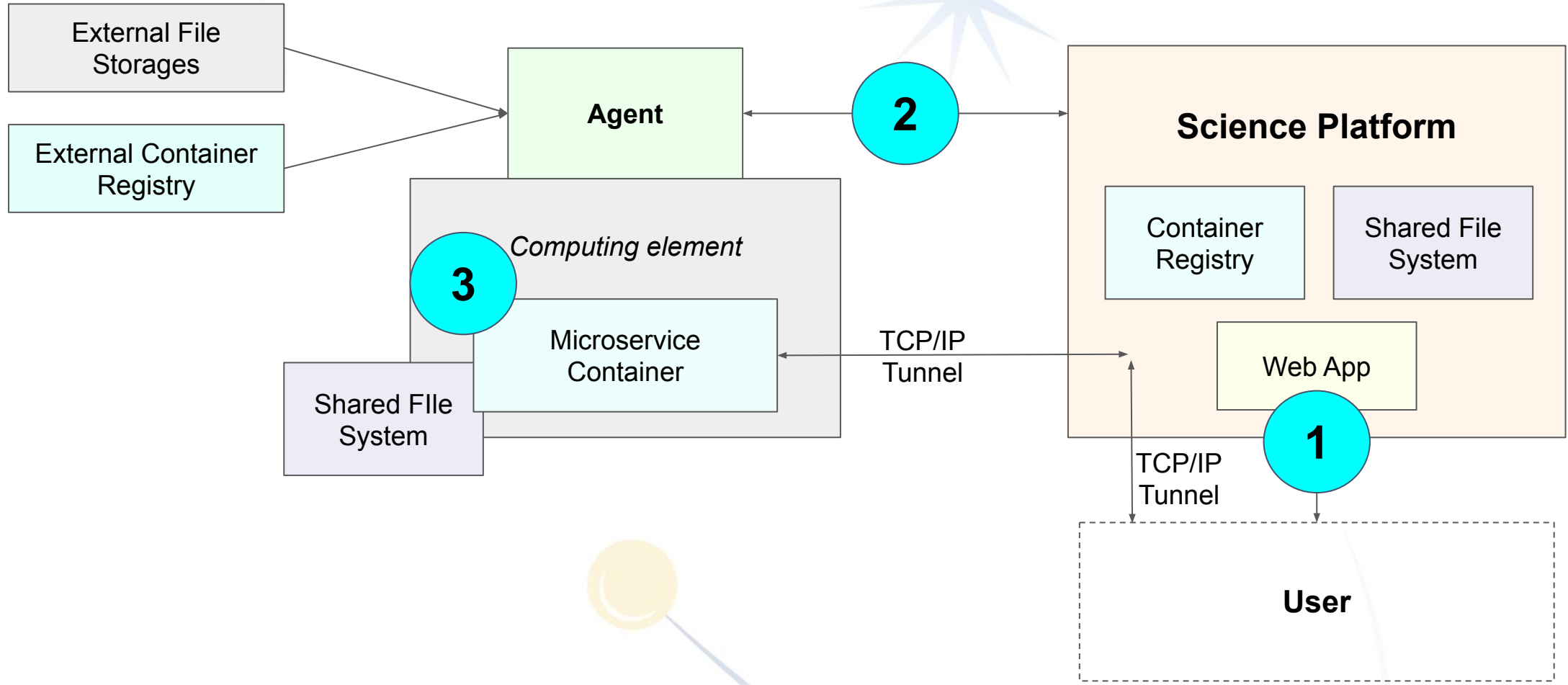
Rosetta architecture: task orchestration



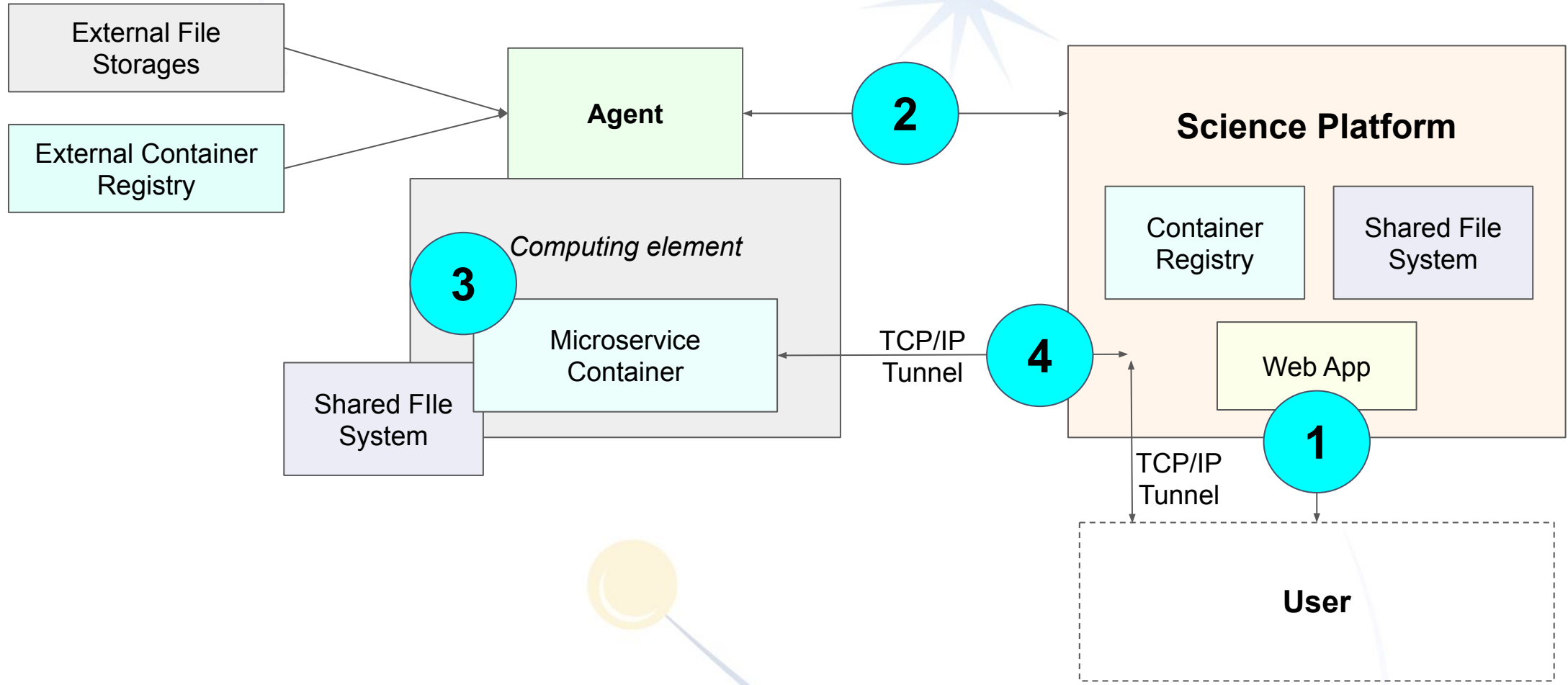
Rosetta architecture: task orchestration



Rosetta architecture: task orchestration

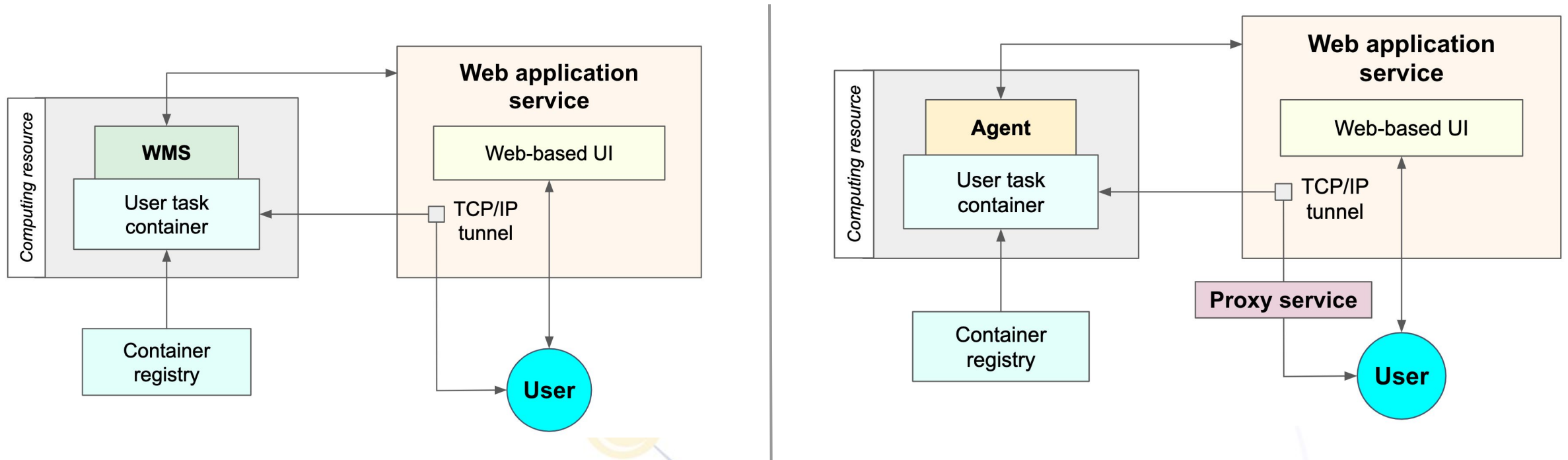


Rosetta architecture: task orchestration

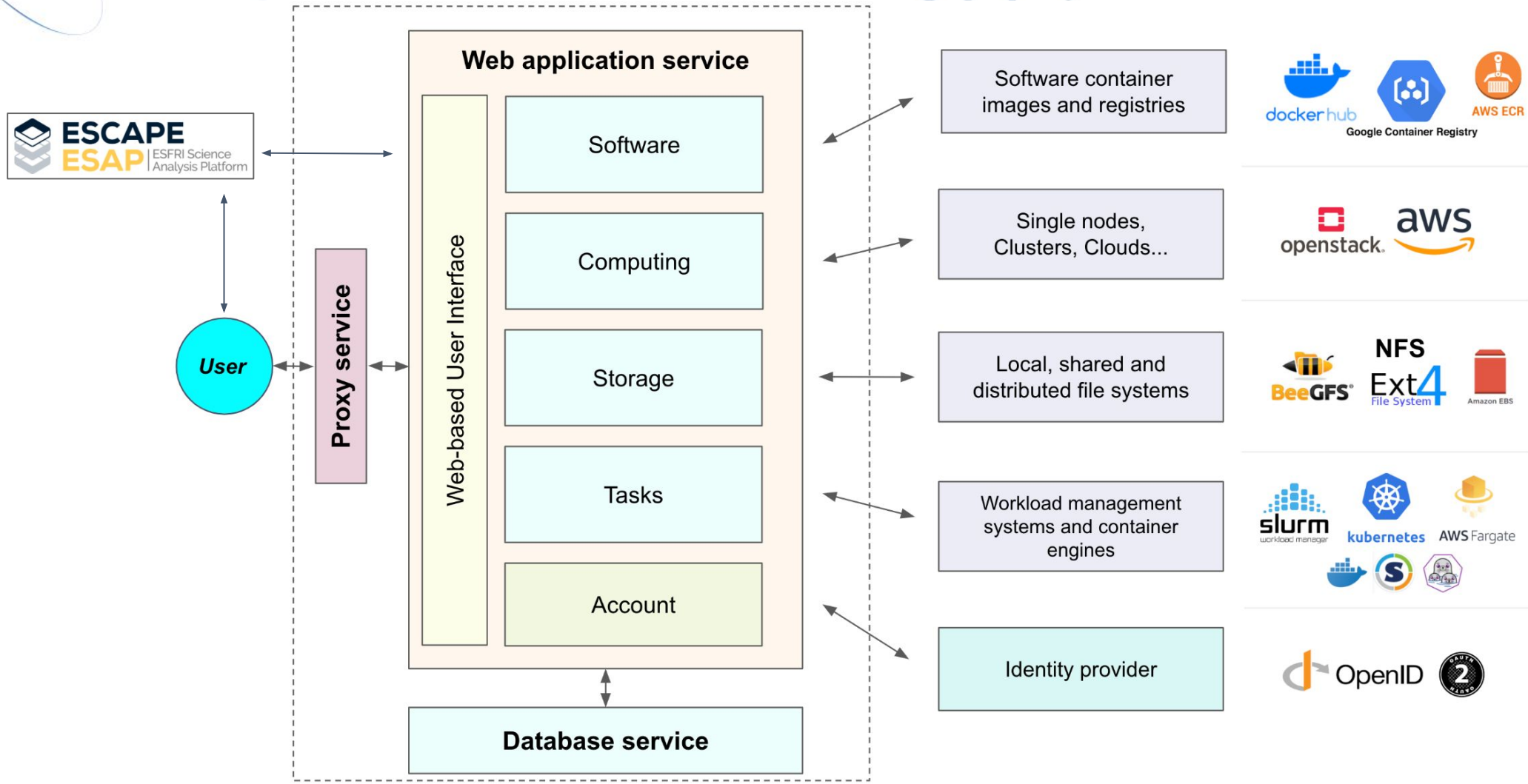


Rosetta architecture: task orchestration

Using a WMS vs Agent and a direct tunnel vs a proxy service



Rosetta: an umbrella for several technologies



Wrapping up

- **Rosetta** is a **Science Platform** which allows to run:
 - **well established software** (*pre-loaded on the platform*)
 - **custom software** and/or dependencies (*added by the users*)
→ *both as: Jupyter Notebooks, GUI applications, command line tools etc.*
- Supports **HPC, Cloud** and **data intensive** systems
- Easy to integrate with existing systems and other platforms (as **ESAP**)
- **Containers** together with **microservice**-level isolation make it **safe** to let users add their own software and provide a **robust architecture**
- Huge improvements in terms of **time-to-setup** and **reproducibility**



Wrapping up

- <https://doi.org/10.1016/j.ascom.2022.100648>
- <https://arxiv.org/abs/2209.02003>
- <https://www.ict.inaf.it/gitlab/exact/Rosetta>



Astronomy and Computing
Volume 41, October 2022, 100648



Full length article
Rosetta: A container-centric science platform for resource-intensive, interactive data analysis

S.A. Russo ^{a, c, ✉}, S. Bertocco ^a, C. Gheller ^b, G. Taffoni ^a

Show more ▾

+ Add to Mendeley 🔗 Share 🗨 Cite

<https://doi.org/10.1016/j.ascom.2022.100648> [Get rights and content](#)

Abstract

Rosetta is a science platform for resource-intensive, interactive data analysis which runs user tasks as software containers. It is built on top of a novel architecture based on framing user tasks as microservices – independent and self-contained units – which allows to fully support custom and user-defined software packages, libraries and environments. These include complete remote desktop and GUI applications, besides common analysis environments as the Jupyter Notebooks. Rosetta relies on Open Container Initiative containers, which allow for safe, effective and reproducible code execution; can use a number of container engines and runtimes; and seamlessly supports several workload management systems, thus enabling containerized workloads on a wide range of computing resources. Although developed in the astronomy and astrophysics space, Rosetta can virtually support any science and technology domain where resource-intensive, interactive data analysis is required.





ESCAPE

European Science Cluster of Astronomy &
Particle physics ESFRI research Infrastructures

THANKS!

Stefano Alberto Russo - INAF OATS

