OPEN-SOURCE PLANETARY DATA PROCESSING ENVIRONMENTS BASED ON JUPYTERHUB AND DOCKER CONTAINERS.

Giacomo Nodjoumi [1], Carlos Brandt [1], Angelo Pio Rossi [1]
[1]Jacobs University Bremen, Campus Ring 1, Bremen, Germany (g.nodjoumi@jacobs-university.de)

Introduction: Planetary data acquired by planetary orbiters are commonly available in the form of low end and higher-end product groups where the former includes raw data that has not been processed and the latter contains all the data derived from the processing of the raw data, including calibrated and map-projected products. Planetary data archives such as NASA PDS Geosciences Node Orbital Data Explorer (ODE) [1], ESA (European Space Agency) Planetary Science Archive (PSA) [2], Mars System of Information (MarSI) [3], Multi-purpose Advanced Tool for Instruments of the Solar System Exploration (MATISSE) [4] are moving towards to provide access not only to the above-mentioned data but also to Analysis Ready Data (ARD) [5]–[7] and interactive web-based analysis tools [8]. However, most of the higher-end products are still processed through widely used open-source tools such as the Integrated Software for Imagers and Spectrometers (ISIS) developed by U.S. Geological Survey (USGS) Astrogeology Research Program [9]–[11] and the NASA Ames Stereo Pipeline (ASP) [12]. Despite being very well documented and maintained, both of those tools require the end-user to perform several technical tasks that may require moderate knowledge in UNIX-based systems and package installation, to get both software working especially if additional libraries and packages are required. Moreover, these installations are more specific for single users and less compatible with working groups or large-scale deployments.

In this work, we present a novel approach for deploying ISIS-ASP jupyter-based environments, customized with several commonly used python libraries and easily scalable from small to large working groups.

The solution presented here requires low deployments and maintenance efforts to system administrators and almost no efforts to end-users, providing a ready to use environment based on jupyter hub/lab.

Components: the solution presented here is based on a combination of a Docker ecosystem [13], Jupyter Hub [14] and customizable combined installation of ISIS-ASP-GISPY that also includes a collection of commonly used python packages (GISPY) that users can further implements to accommodate their needs.

All components are built as docker containers, in which there is the main server (Jupyter Hub) that upon login requests through GitHub/GitLab (or custom secure sign-in validation) spawn an instance of the ISIS-ASP-GISPY container and assign solely to the logged user, automatically adding the ISIS_DATA folder, a persistent user space storage and a shared folder containing pre-prepared Jupyter notebooks for planetary data processing. The structure is presented in **Error!**

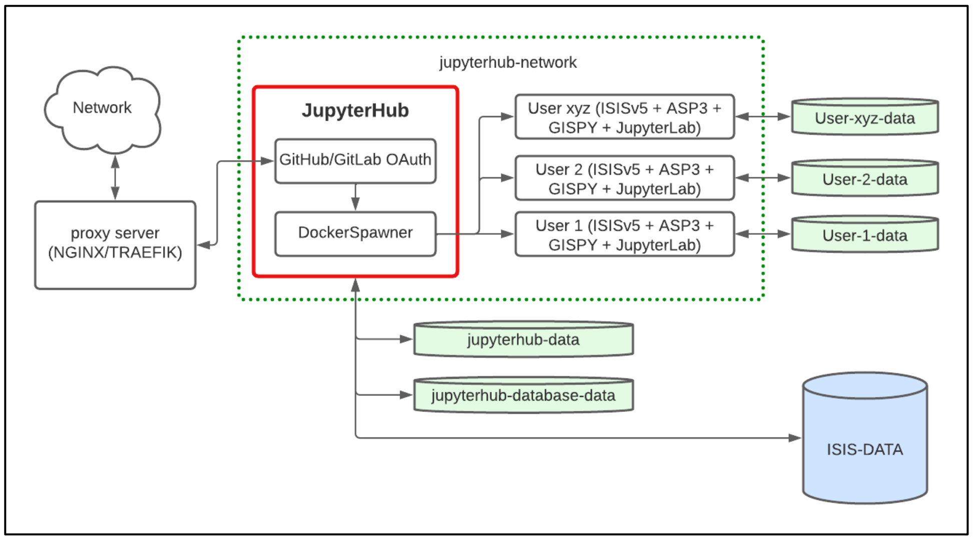
jupyterhub-network
Network
proxy server (NGINX/TRAEFIK)
JupyterHub
GitHub/GitLab OAuth
DockerSpawner
User xyz (ISISv5 + ASP3 + GISPY + JupyterLab)
User 2 (ISISv5 + ASP3 + GISPY + JupyterLab)
User 1 (ISISv5 + ASP3 + GISPY + JupyterLab)
User-xyz-data
User-2-data
User-1-data
jupyterhub-data
jupyterhub-database-data
ISIS-DATA

Figure 1. Schematics of the proposed solution with additional reverse proxy to grant domain association and provide access from outside the local area network where the toolset is deployed.

Reference source not found..

Deployment: the deployment is straight-forward, upon an initial basic configuration of the file containing all the environmental variables such as the ISIS_DATA folder or the port at which the Jupyter Hub will be deployed, the installer script will automatically take care of all the necessary tasks including the creations of the docker volumes, the docker network and the building of the final docker images. Finally, a docker-compose file can be used to start up the server.

Once the Jupyter Hub is running, users have only to connect to the localhost using the configured port, or to the associated domain url, if configured. Then a prompt will ask to login using the configured service and automatically the Hub will create a user volume, deploy the ISIS-ASP-GISPY image and redirect the user to its own jupyter lab instance.

Customizations: All the components are deeply customizable both from system administrator and user perspective, granting high compatibility with different host systems and network configurations.

The basic configuration for sign-in authentication relies on GitHub application combined with a user list file to allow access only to users whose user GitHub id is found in the user list. Switching to GitLab an organization-based access can be configured.

The default docker image is based on ISIS version 5.0.1 since it is the latest version with confirmed compatibility with the ASP version 3.0, this image also contains several python packages and can also be further implemented with additional packages such as PlanetaryPy [15], all server through a jupyter lab environment.

To have access to all ISIS users have two choices, the first is to launch a terminal within their Jupyter Lab instance or directly use a Jupyter Notebook by taking advantage of the Kalasiris [16]preinstalled python package. ASP can only be used through the terminal or by writing python functions.

All packages and docker images can be customized at any moment by editing their corresponding dockerfiles and re-running the installer.

Availability: The latest pre-release of this work can be found both on Zenodo [17] and GitHub [18].

Conclusion: the work presented here has the two main objectives, the first one is to simplify the deployment and maintenance of planetary data processing environments based on ISIS, ASP, and python while the second one is to grant the access to such powerful software to anyone without installation and configurations overheads, especially in the cases where multiple instances are requested, such as workgroups, classrooms, and workshops.

Further works will implement the possibility to the users to reconnect and recover the state of their notebook even after a disconnection. This feature is critical for long processing but is not yet implemented due to Jupyter Hub and Lab complex architecture.

Funding: This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 101004214 and No 871149.

References:

[1] PDS Geoscience Nodes (2022) PDS Geosciences Node Orbital Data Explorer (ODE)
[2] ESA (European Space Agency) (2022) Planetary Science Archive (PSA UI)
[3] Quantin-Nataf C. et al. (2018) MarsSI: Martian surface data processing information system.
[4] Zinzi A. et al (2021) MATISSE 2.0: The SSDC Webtool for Integrated Planetary Science Analysis
[5] Fergason R. L. et al. (2021) Analysis Ready Data Available Through the SpatioTemporal Asset Catalog (STAC)
[6] Hare T.M. et al. (2021) BUILDING A LUNAR SPATIAL DATA INFRASTRUCTURE (SDI)
[7] Laura J. R. et al. (2021) Knowledge Inventory of Foundational Data Products in Planetary
[8] Fraiser C. Keszthelyi L. (2021) PLANETARY IMAGE EDITOR I/O-WEB SERVICE
[9] Gaddis L. et al. (1997) An Overview of the Integrated Software for Imaging Spectrometers (ISIS)
[10] Anderson J. A. et al (2004) Modernization of the Integrated Software for Imagers and Spectrometers
[11] Laura J. R. (2022) Integrated Software for Imagers and Spectrometers.
[12] Annex A. M. et al (2021) ASAP-Stereo, Ames Stereo Automated Pipeline.
[13] Merkel D. (2014) Docker: lightweight linux containers for consistent development and deployment
[14] Kluyver T. et al. (2016) Jupyter Notebooks – a publishing format for reproducible computational workflows
[15] Aye K. M. et al. (2021) The PlanetaryPy Project
[16] R. A. Beyer, ‘Kalasiris, a Python Library for Calling ISIS Programs’, in 51st Annual Lunar and Planetary Science Conference, Mar. 2020, p. 2441.
[17] G. Nodjoumi and C. H. Brandt, europlanet-gmap/docker-isis3: First Release. Zenodo, 2022. doi: 10.5281/zenodo.6396321.
[18] G. Nodjoumi and C. H. Brandt, ‘docker-isis3’, GitHub. https://github.com/europlanet-gmap/docker-isis3