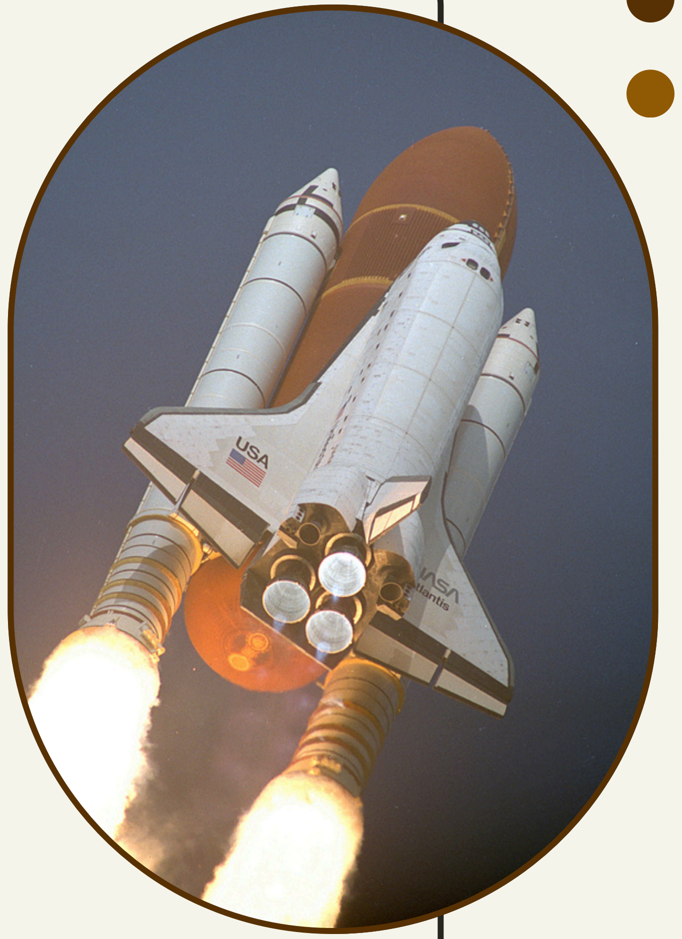
Empowering Open Science with Scalable Interactive Computing Environments in India

Kriyanshi Shah
Space Applications Centre, ISRO

ABOUT ME

I am Kriyanshi Shah. I work as a software engineer at Space Applications Centre, ISRO.

- I am a Software Engineer with experience in building large-scale backend systems and developer tools.
- I specialized in designing platforms for satellite data research and scientific data processing pipelines.
- My main focus at SAC is on applying Free and Open Source Software (FOSS) to enable reproducible, collaborative, and scalable scientific research.
- I am Experienced in Kubernetes, JupyterHub, Docker, Golang, and distributed computing systems for scientific workflows.



HAVE YOU EVER WORKED WITH SATELLITE DATA BEFORE?

H5 FILES, SHORT FOR HIERARCHICAL DATA FORMAT VERSION 5, ARE A WIDELY USED FILE FORMAT DESIGNED TO STORE AND ORGANIZE LARGE AMOUNTS OF COMPLEX DATA, PARTICULARLY IN SCIENTIFIC, ENGINEERING, AND REMOTE SENSING APPLICATIONS.

The Challenge in Scientific Research

OFTEN WORKING AND MAKING SENSE OF SATELLITE DATA IS EASIER RATHER THAN SPENDING SLEEPLESS NIGHTS WONDERING WHY DIDN'T THAT LIBRARY INSTALL ITSELF CORRECTLY.

Our Observation

Our researchers and scientists work on air gapped environments and often installing and maintaining all the python libraries are a headache. Sometimes different environments for different needs and that too on machines without internet. Managing all this with writing meaningful code is a bit too much.

Diverse scientific domains require customized computational environments.

Traditional setups are often siloed, non-reproducible, and lack scalability.

There's a pressing need for open, collaborative, and scalable solutions to advance scientific research.

SO TO SUM IT UP

PROBLEM VS. SOLUTION

- Researchers face complex, fragmented FOSS workflows.
- Onboarding new users to scientific computing tools is time-consuming.
- Converting notebooks into shareable applications requires extra coding effort.
- Limited infrastructure and support slows down adoption of Open Science practices.

- Scalable cloud-native platform built with Kubernetes + JupyterHub.
- Preconfigured Docker environments for different scientific domains (e.g., planetary science, meteorology).
- Notebook → Web App conversion via Mercury integration inside JupyterLab.
- REST APIs + JupyterLab extensions for intuitive app management (create/update/delete).
- Lower barrier to entry: researchers focus on science, not infrastructure.

IDEA BEHIND IT

So we crafted something for our lovely team mates initially and then we thought maybe this can be implemented everywhere. Cause they say that,

“build something you want.”

WE DID SOMETHING:

We built interactive computing environments using JupyterHub and Kubernetes, that offers scalable, secure, and domain-customizable computing environments tailored for scientific disciplines such as planetary science and meteorology. Customized Docker containers preloaded with scientific libraries allow researchers to spin up user-ready workspaces with minimal technical overhead.

OUR SOLUTION

Built on Kubernetes + JupyterHub for scalable and secure user environments. Customized Docker containers preloaded with scientific libraries (planetary science, meteorology, etc.). Integrated Jupyter Server Proxy + Mercury to convert notebooks into shareable web applications. Added REST APIs & JupyterLab extensions for managing Mercury apps (create/update/delete) via UI.

1. **Cloud-Native Architecture**
2. **Enhanced User Experience**
3. **Domain-Specific Environments**
4. **Seamless Research-to-App Workflow**

OVERVIEW

User Access

Researchers access JupyterHub via a web interface

Containerization

Each user session runs in a kubernetes pod tailored to specific scientific needs.

Interactivity

Custom build of Mercury integrates with JupyterLab, allowing users to convert notebooks into shareable web apps

Scalability

Kubernetes manages container deployment, scaling, and resource allocation.

OPEN SOURCE TOOLS USED

1. Jupyterhub
2. KubeSpawner
3. Kubernetes
4. Jupyter Server Proxy
5. Docker
6. JupyterLab
7. Custom mljar/mercury build
8. Custom JupyterLab Extension

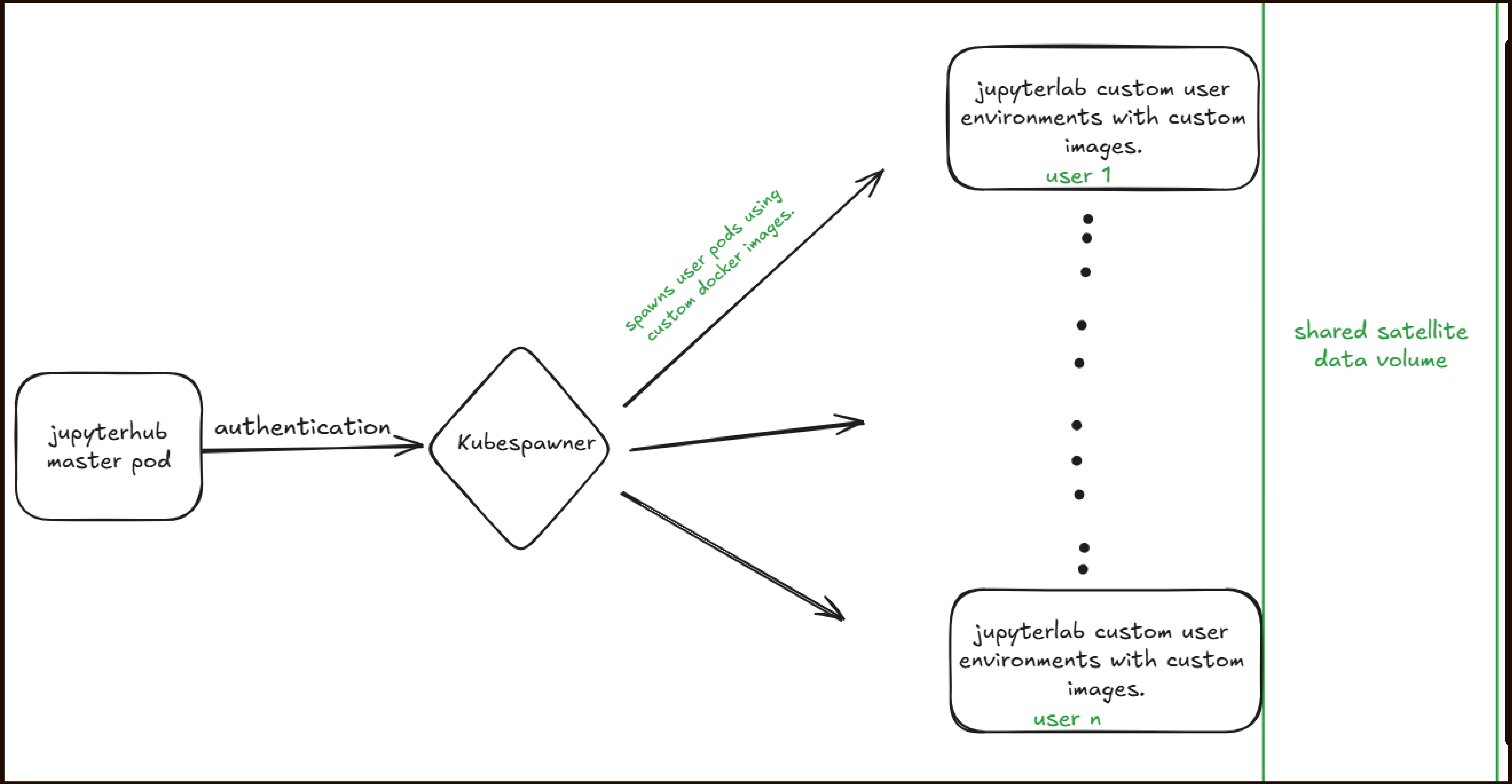
WHY THEY ARE USED

JupyterHub	Provides a centralized web interface for researchers to log in and access computing environments.
KubeSpawner	Spawns user pods on Kubernetes after authentication, ensuring isolation and scalability
Kubernetes	Orchestrates container deployment, scaling, and resource allocation for multiple users.
Docker	Packages domain-specific scientific environments (e.g., planetary science, meteorology) with preconfigured libraries.

JupyterLab	Interactive computing environment where researchers write, analyze, and visualize scientific workflows.
Jupyter Server Proxy	Enables secure access to additional services within the JupyterHub environment through proxied URLs
Mercury (custom build)	Converts jupyter notebooks into webapps. Our custom build supports it inside from jupyterhub user pods.
JupyterLab Extensions	Custom extensions developed to let users create, update, and delete Mercury web apps directly from the JupyterLab interface.

ARCHITECTURE

Here's the architecture we crafted to create the platform.



USER FLOW

Login & Authentication

- A researcher logs into the platform via the JupyterHub web interface.
- Authentication ensures secure, role-based access.

Pod Creation

- KubeSpawner launches a dedicated Kubernetes pod for the user.
- Each pod is based on a custom Docker image tailored for specific scientific domains.

Data Access

- The user pod is mounted with a shared satellite data volume, giving access to large-scale datasets for analysis.

Interactive Environment

- Inside the pod, the researcher works with JupyterLab to write, run, and visualize scientific workflows.

App Creation & Sharing

- Using Mercury integration, researchers can convert notebooks into interactive web apps.
- Web apps can then be shared with peers for collaboration and reproducibility.

TAILORED ENVIRONMENTS

Planetary Science

Includes libraries including cartopy, rasterio, xarray, zarr, geopandas, netcdf4, GDAL, astropy, skyfield, sunpy, Pyproj, Shapely, D3.

Meteorology

Pre-installed with climate modeling tools, data visualization libraries like matplotlib, cartopy, geoplotlib, geoviews, hyplot, plotly, bokeh, seaborn, Folium, Altair, pygal.

MY LEARNINGS

1. Installing all the python libraries on air gapped machines
2. Creating custom jupyterlab extensions
3. Creating custom docker images

ENHANCING USER EXPERIENCE

1. **Mercury Integration**

   Users can convert notebooks into web apps without leaving JupyterLab

2. **RESTful APIs**

   Added for programmatic control over app deployment.

3. **JupyterLab Extensions**

   Custom extensions to manage Mercury apps via GUI.

FUTURE DIRECTIONS:

1. **Broader Adoption**

   Plans to extend the platform to more scientific domains.

2. **Long-term Support**

   Strategies for maintaining and updating the platform.

Links to the tools that are used:

1. https://github.com/kriyanshii/mercury(custom build)
2. https://github.com/jupyterhub/jupyter-server-proxy
3. https://github.com/jupyterhub/kubespawner
4. https://github.com/mljar/mercury
5. microk8s
6. docker
7. https://github.com/jupyterhub/jupyterhub
8. https://github.com/jupyterlab/jupyterlab
9. https://github.com/jupyterlab/extension-template

THANK YOU!

Kriyanshi Shah
http://github.com/kriyanshii
http://twitter.com/kriyanshii