Tutorial III

Cómo trabajar con JupyterLab en la plataforma Comunidad

JupyterLab es un entorno de desarrollo interactivo de código abierto basado en la web que integra código, datos y documentación en un único espacio de trabajo flexible. Permite a investigadores y desarrolladores combinar la ejecución de código en vivo, texto narrativo y resultados visuales, garantizando flujos de trabajo transparentes y reproducibles.
Dentro del proyecto COMUNIDAD, JupyterLab sirve como entorno central para:

- Acceso y exploración de datos: la integración con el estándar Catálogo de activos espacio-temporales (STAC) permite a los usuarios consultar, descubrir y acceder a conjuntos de datos de observación de la Tierra (EO) de fuentes distribuidas de forma estandarizada.

- Análisis reproducible: los cuadernos capturan tanto la metodología como los resultados, lo que respalda la verificación y la transferencia de conocimientos entre los socios del proyecto.

- Soporte para múltiples idiomas: Python es el lenguaje de programación predeterminado, pero se puede configurar soporte para otros, como R y Julia, a través de kernels, lo que lo hace adaptable para diversos equipos de investigación.

- Visualización y análisis geoespacial: La compatibilidad con bibliotecas como GeoPandas, rasterio y Leaflet permite el procesamiento y visualización de datos geoespaciales recuperados de los puntos finales de STAC o directamente del repositorio de escenas mapeadas de Sentinel ubicado en la infraestructura de COMUNIDAD.

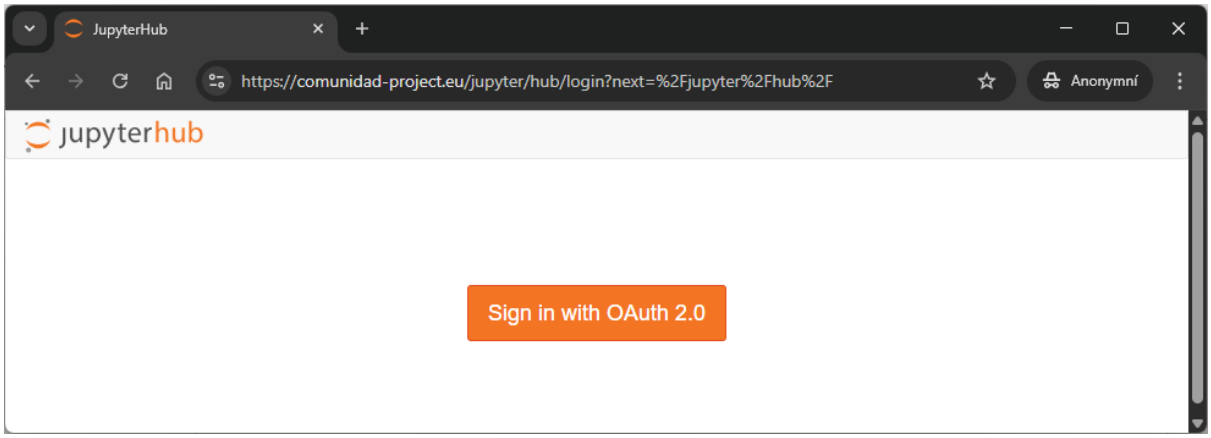
- Extensibilidad: Su diseño modular permite la integración con servicios específicos del proyecto (por ejemplo, MapProxy, API, almacenamiento basado en la nube) y complementos adaptados a los casos de uso de EO e IoT.

- Colaboración: Los cuadernos se pueden compartir, versionar y reutilizar entre instituciones, lo que fomenta la cooperación transfronteriza y la alineación con los principios de datos FAIR (Encontrables, Accesibles, Interoperables, Reutilizables).

Al combinar JupyterLab con el descubrimiento de datos basado en STAC, el proyecto COMUNIDAD garantiza un entorno sólido y fácil de usar para trabajar con Copernicus y otros datos de EO, lo que permite a los usuarios técnicos y a las partes interesadas experimentar, validar y escalar soluciones innovadoras en línea con las prioridades digitales de la UE.

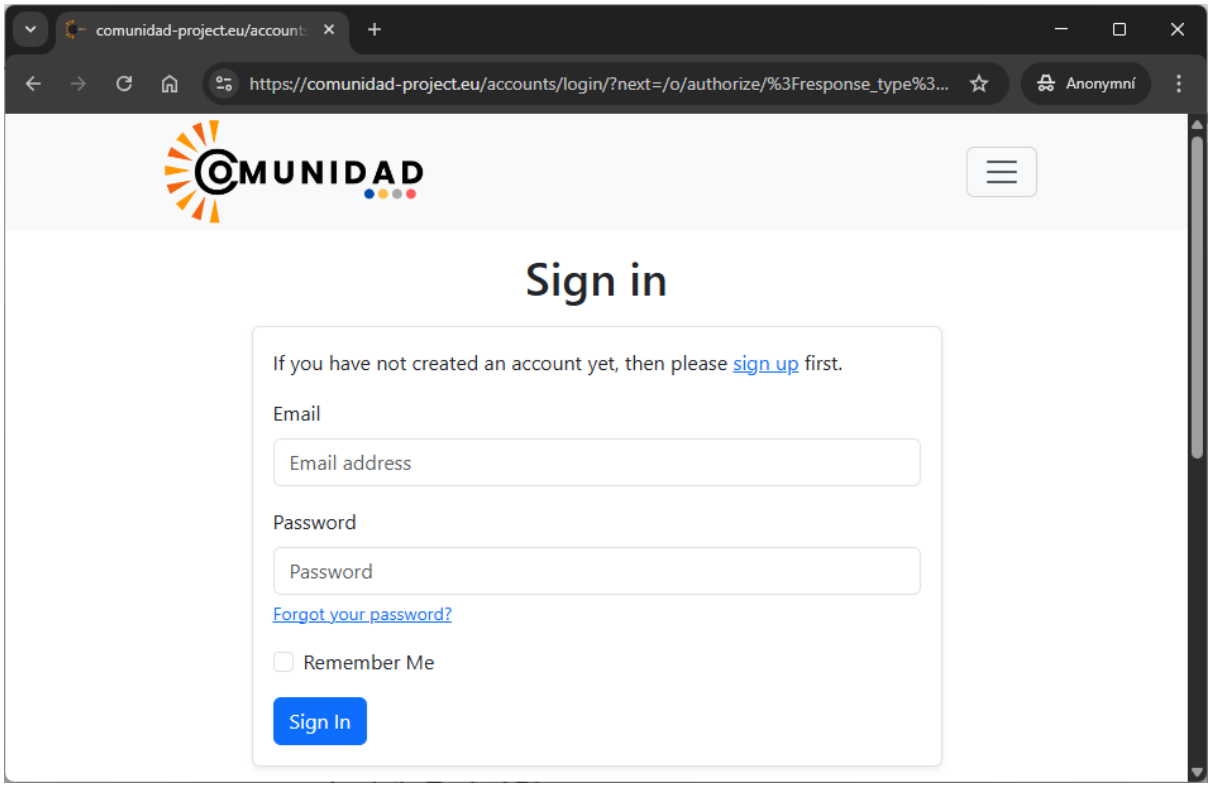
Cómo trabajar con JupyterLab

Acceda a la instancia JupyterLab de la Plataforma COMUNIDAD en https://comunidad-project.eu/jupyter



Inicie sesión utilizando el Portal Web de la plataforma COMUNIDAD y autorice a JupyterLab a acceder a su cuenta del Portal.
Si ya ha iniciado sesión en la página web de la Comunidad, solo tiene que autorizar a Jupyter para que acceda a su cuenta. Si aún no lo ha hecho, se le solicitará que lo haga antes de autorizar.

Una vez que se inicia el servidor Jupyter, puedes empezar a crear un nuevo cuaderno. Voy a seleccionar código basado en Python 3, le asignaré un nombre y luego podremos empezar a codificar.



Puedes comenzar a usar Jupyter, consulta la documentación en https://jupyterlab.readthedocs.io/es/stable/getting_started/overview.html

Utilice los puntos finales STAC de la API EO para acceder a los datos de Copernicus disponibles a través de la Plataforma COMUNIDAD o acceda al repositorio subyacente de escenas Copernicus Sentinel directamente usando el nombre de la escena seleccionada, por ejemplo:

```
importar sistema operativo

# Ruta a su archivo GeoTIFF local
ruta = os.path.expanduser ('~/compartido')
tiff_path = os.path.join (ruta,
      'Caldas/TCI/S2A_MSIL2A_20230103T152641_N0510_R025_T18NVL_20240807T094
726.SAFE_TCI.tif')
# imprime la ruta del archivo TIFF
ruta_tiff
# usa el ráster para cálculos posteriores
…
```

Se puede acceder a la lista de escenas disponibles para cada ubicación a través de la API de EO, por ejemplo: https://comunidad-project.eu/eo-api/scenes?location=Caldas&api_key=eo-api-key-dev

Las escenas también se pueden filtrar por fechas, por ejemplo: https://comunidad-project.eu/eo-api/scenes?location=Caldas&start_date=20240101&end_date=20240130&api_key=eo-api-key-dev

El siguiente ejemplo demuestra el acceso a archivos ráster (.TIFF) y vectoriales (Shapefile) en el servidor, reduce la resolución del ráster y lo renderiza en la ventana de Jupyter mediante el paquete de Python matplotlib. También imprime los metadatos básicos del ráster en la consola.

```
importar sistema operativo
importar numpy como np
importar pandas como pd
importar rasterio
de rasterio.enums importar Remuestreo
desde rasterio.mask importar máscara
importar geopandas como gpd
importar matplotlib.pyplot como plt
desde rasterio.transform importar array_bounds

# Ráster
ruta = os.path.expanduser('~/shared')
tiff_path = os.path.join(path, 'Uso del suelo_2021_Cuenca del río
Aysén_2021.tif')

# SHP (ruta absoluta)
cuencas_path = '/opt/eo-copernicus-api-
dock/work/locations/jupyter/cuencas_rio_aisen/Cuencas_Rio_Aisen.shp'

imprimir("TIFF:", ruta_tiff)
print("SHP :", cuencas_path)

con rasterio.open(tiff_path) como origen:
raster_crs = src.crs
nodata = src.nodata

# reducir la resolución para una representación rápida
max_dim = max(origen.ancho, origen.alto)
escala = max_dim // 1000 si max_dim > 1000 de lo contrario 1
out_h = src.height // escala
out_w = src.width // escala

datos = src.read(
1,
forma_de_salida=(altura_de_salida, anchura_de_salida),
remuestreo=Remuestreo.más cercano
)
```

```
# extensión para matplotlib
izquierda, abajo, derecha, arriba = array_bounds(out_h, out_w,
src.transform)

plt.figura()
plt.imshow(datos, extensión=(izquierda, derecha, abajo, arriba))
plt.title("Ráster de uso del suelo (vista previa)")
plt.xlabel("X")
plt.ylabel("Y")
plt.show()

print("CRS ráster:", raster_crs)
print("Raster sin datos:", sin datos)
```