# Integrating ESAP with the ESCAPE data lake

## ESAP Training Workshop 21-11-22

James Collinson (SKAO)

Yan Grange (ASTRON)

# Overview

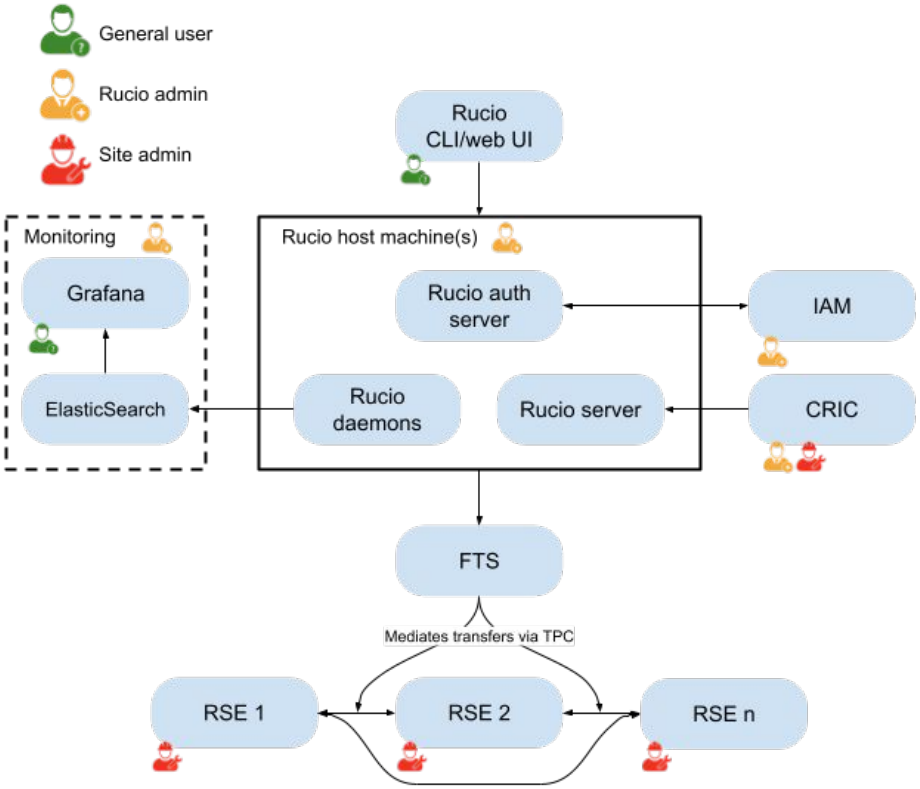
- The ESCAPE Data lake
  - Overview of Rucio
  - Architecture, glossary of basic terminology
  - Interacting with Rucio
  - Authentication
- Querying data from Rucio from ESAP
- Integrating Rucio storage with Jupyter environment

# Rucio - the ESCAPE data lake orchestrator

- Rucio is a large scale scientific distributed data management tool
  - Built by/for the ATLAS experiment but since has been adopted by CMS, Belle II, …
- Rucio architecture: server, daemons, middleware, applications
  - Server exposes an API, handles authentication etc.
  - Daemons handle asynchronous work to keep the system in the requested state, e.g. transfers, deletions…
- "User" inputs declarative statements that create rules constraining how many copies of your data you want and where
- Storage agnostic: Rucio is not responsible for managing storage - as long as there is a communication protocol to talk to it, it can be connected

# Rucio architecture

- Rucio server/auth server/daemons run separately to data transfer tools
- Commands issued to FTS to enact transfers
- RSEs send/receive data between each other directly (Third Party Copy)



# Rucio terminology

| | |
|---|---|
| **Rucio Storage Element** (RSE) | A logical abstraction of a storage system e.g. supported protocols, hostnames, ports, prefixes, ... |
| **Data Identifier** (DID) | Unique addressable unit for files, **datasets** or **containers**. Follows pattern of <**scope**>:<**name**>. **DIDs** are unique within the full namespace. |
| **Scope** | A partitioning of the full namespace. |
| **Name** | Self explanatory. **Names** are unique within a **scope**. |
| **Dataset** | A collection of dids. |
| **Container** | A collection of **datasets** or, recursively, **containers**. |
| **Rule** | The medium by which the user affects the system state and by which the state is constantly evaluated for compliance, e.g. "*make two copies of data, one on disk and one on tape*" |
| **Replica** | The physical result of a replication **rule**, i.e. bytes on storage |

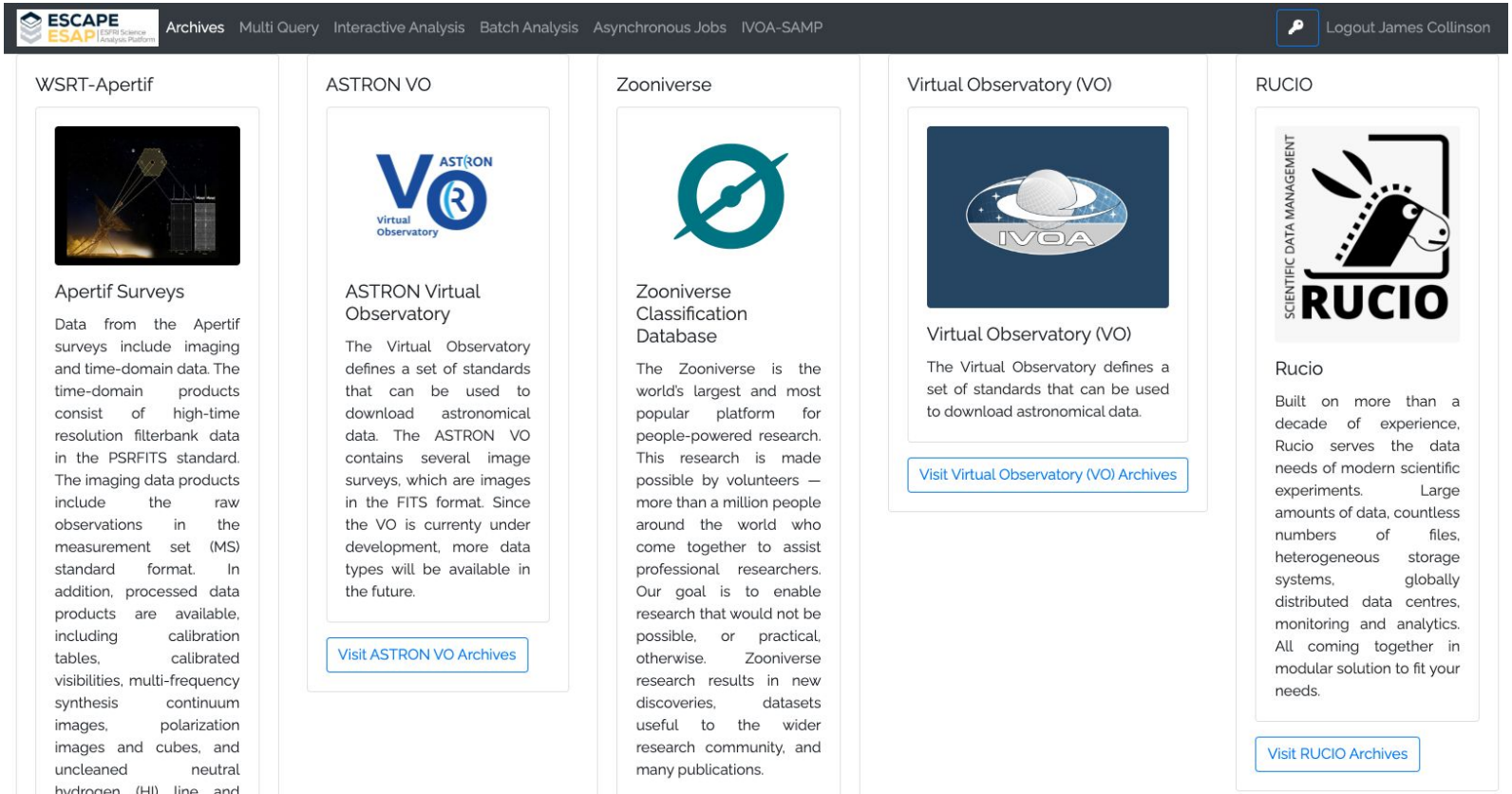
# Interacting with Rucio

- **REST API**
  - Directly call Rucio server via HTTP/REST
  - (https://rucio.cern.ch/documentation/html/rest_api_doc.html)
  - e.g. `$ curl --request GET --url http://<SERVER_URL>/rules/{RULE_ID} --header 'X-Rucio-Auth-Token: <TOKEN>'`
- Rucio CLI client
  - Containers available (https://hub.docker.com/r/rucio/rucio-clients)
  - e.g. `$ rucio rule-info <RULE_ID>`
- Rucio Python clients
  - PIP installable (https://pypi.org/project/rucio-clients/)
  - e.g. `>>> RuleClient.get_replication_rule(<RULE_ID>)`

# Data Lake Authentication

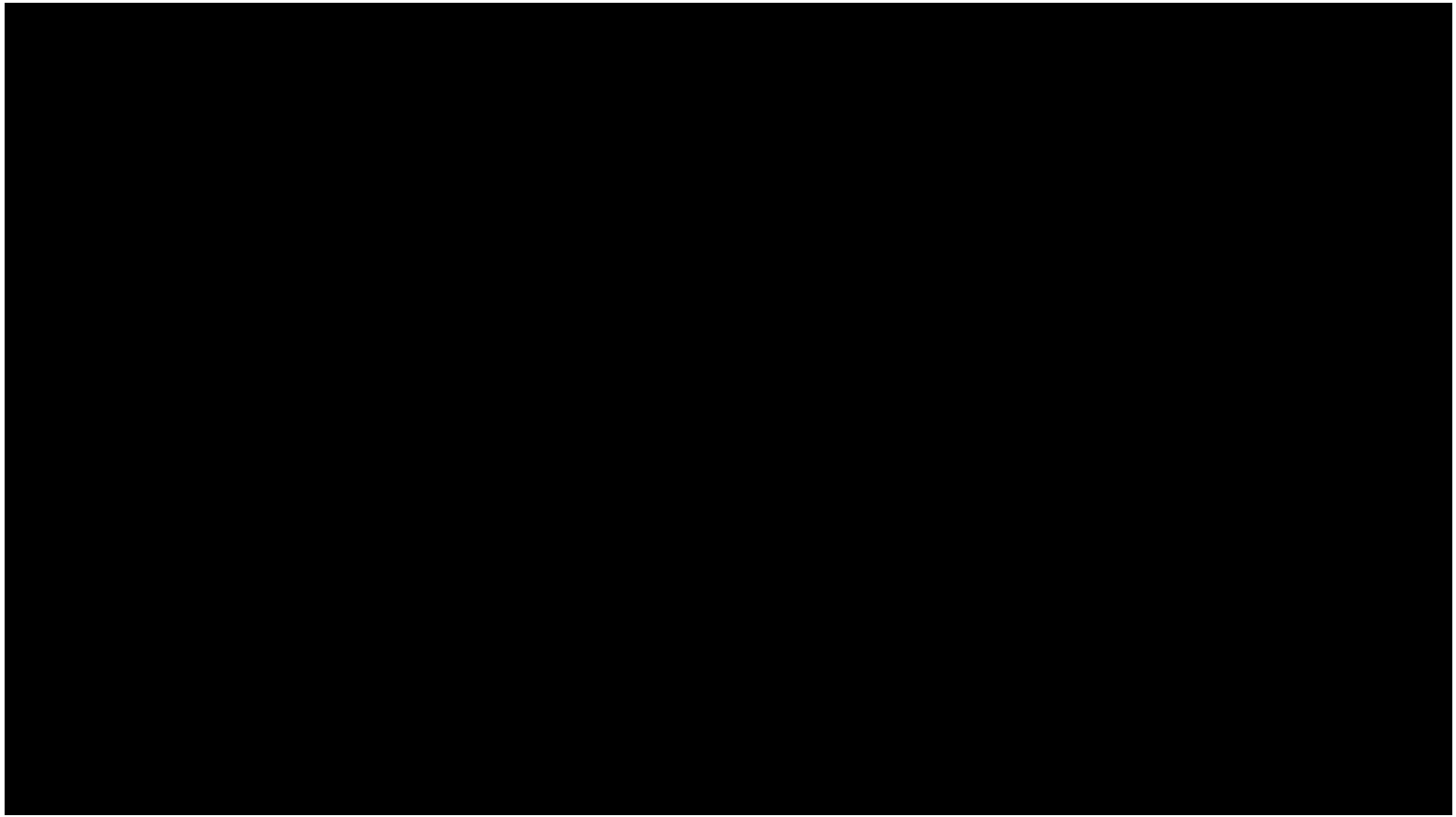
- Three modes of auth:
  - X.509 (legacy)
  - User/pass (deprecated)
  - **OIDC token**
    - Built on OAuth2.0
- Rucio server authenticates with Indigo IAM service, gets token
- Token is exchanged for FTS/storage site operations

# Demo: Querying Rucio data lake from ESAP




Credit: Hugh Dickinson, Yan Grange, Nico Vermaas, Zheng Meyer, Klaas Kliffen

# Demo: Querying Rucio data lake from ESAP



# JupyterHub/BinderHub

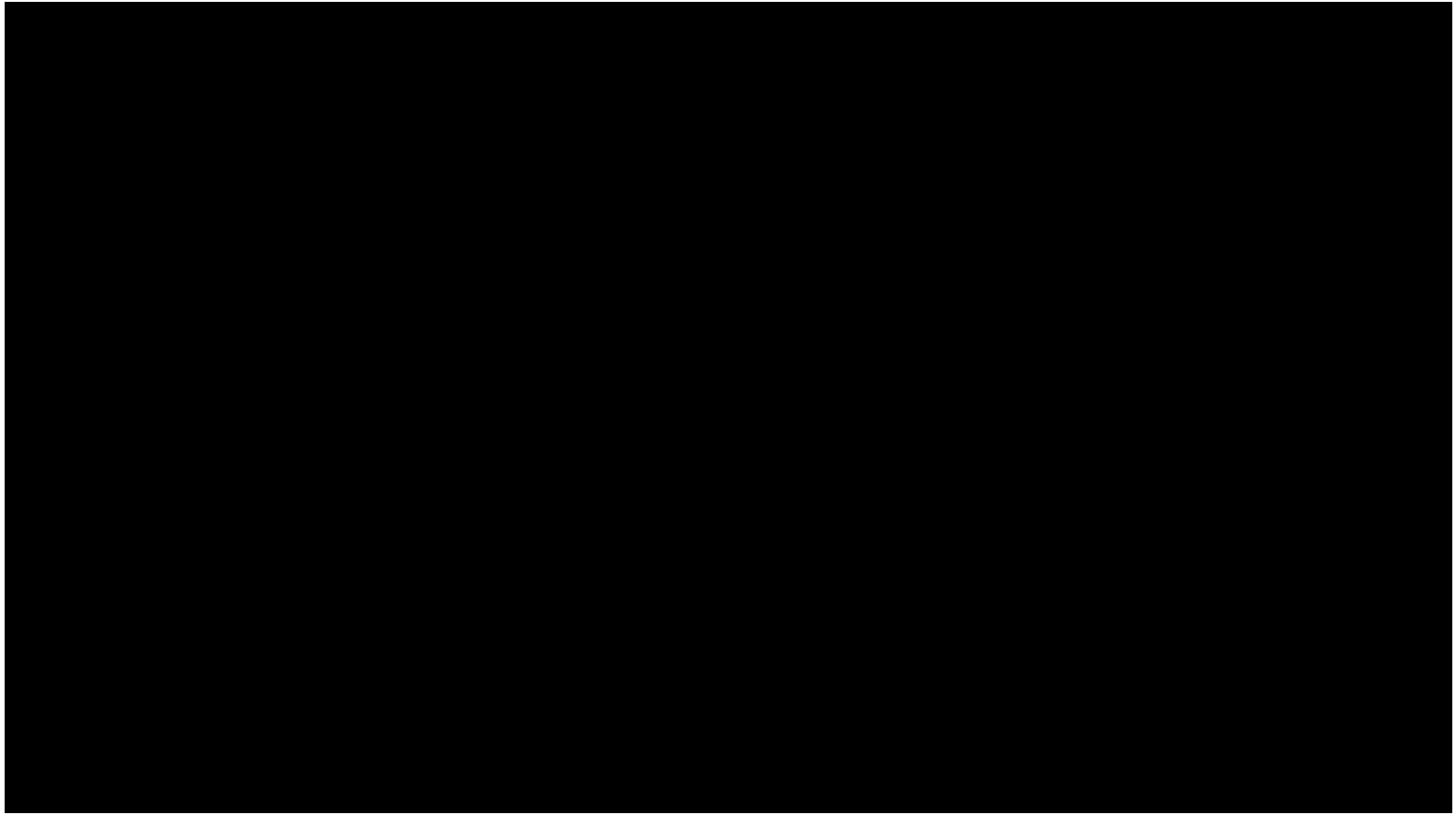
- JupyterHub - web application which can dynamically spawn Jupyter Notebook server apps
- These allow a user to write code (usually Julia/Python/R), run it, and view/share the output
- A number of instances have been deployed at institutes - provide IDA resources to ESAP
- Binder is a tool which can generate Jupyter Notebook environments from Binder-ready code repos
- BinderHub is the combination of Binder running with a JupyterHub

# Bringing compute to the data

- JupyterHub is a good way to securely provision compute to users
- Integrating with storage requires some configuration
- Persistent storage can be configured which allows for per-user scratch spaces and shared volumes
- Configuring with Rucio has been demonstrated initially by CERN*, through the Rucio-JupyterLab extension - this is colloquially known as the 'Data Lake as a Service' (DLaaS)

*Credit Muhammad Aditya Hilmy, Rizart Dona, Riccardo Di Maria

# Data lake as a Service



# DLaaS Requirements

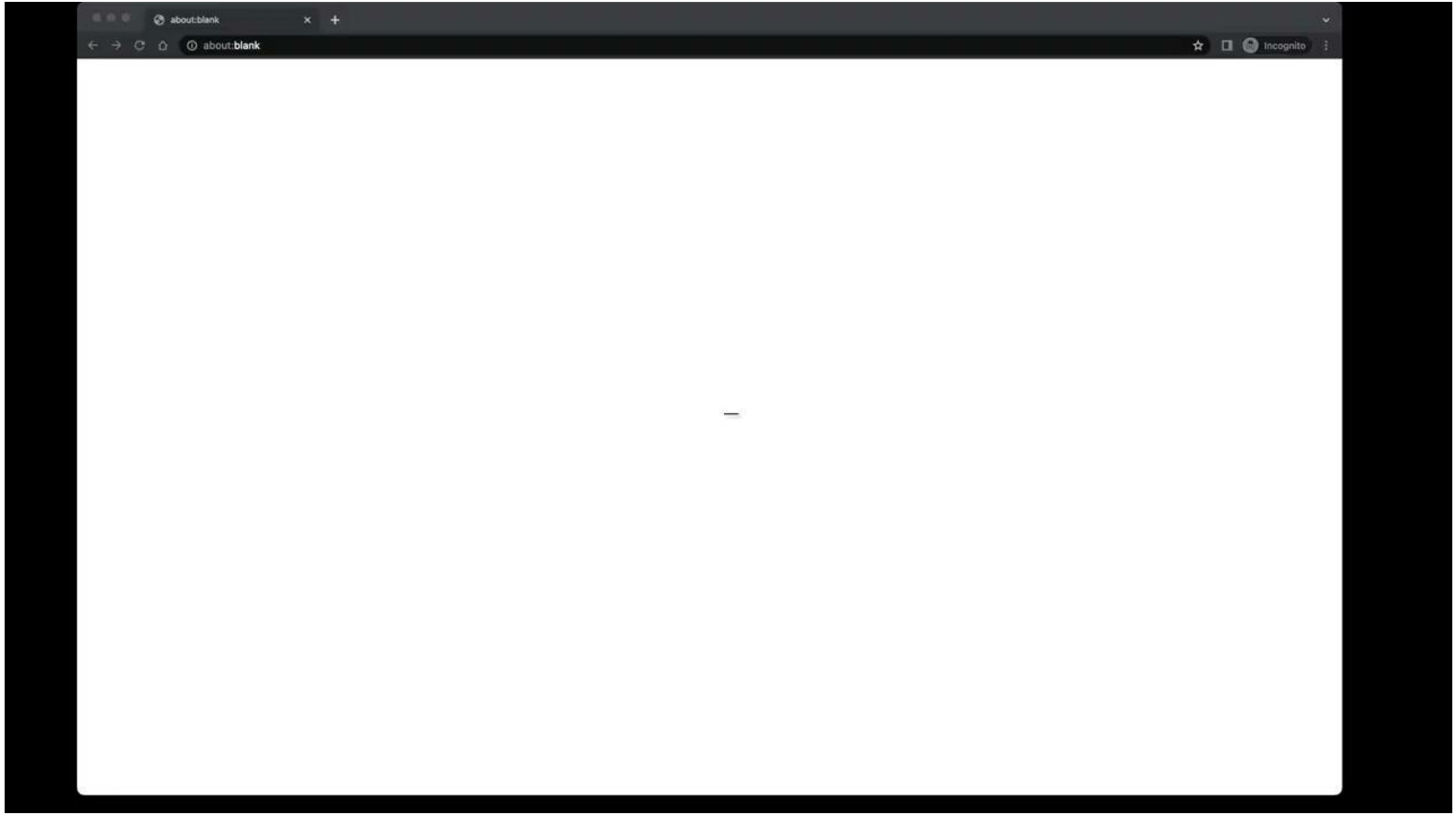
- RSE colocated with JupyterHub
- Jupyter Notebook environments have read-access to RSE volume
- Prerequisites:
  - JupyterHub
  - Colocated RSE
  - Rucio server

  } OIDC token authentication
- Custom JupyterHub and Notebook server images
- ConfigMaps to prep Rucio client/setup token refresh
- Detailed setup/config guide available on request

# DLaaS-ESAP Shopping Basket integration

- ESAP data from queries can be put in a “shopping basket”
  - Using a python client, the references to the data (e.g. URLs to images, or Rucio DIDs) can be obtained
- Shopping basket works with “connectors”
  - Minimal connector defines how to select data from a specific source
  - Also connectors can contain custom data access functionality

# Demo: DLaaS-ESAP Shopping Basket integration



# Summary and Next Steps

- Rucio integrated with ESAP for data discovery
- Data lake as a Service brings computation to data managed by Rucio

## Future

- Improved query mechanism
  - Metadata over DIDs?
  - Improved IVOA metadata integration in development
- More streamlined authentication mechanism
- Tighter integration with interactive analysis

# Questions?