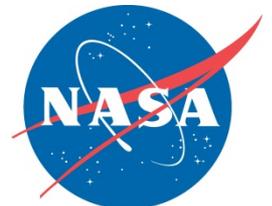


Collaborative Workbench to Accelerate Science Algorithm Development

Manil Maskey*, Rahul Ramachandran, Kwo-Sen Kuo,
Chris Lynnes

Oct 28, 2014
Earth Science Technology Forum

mmaskey@itsc.uah.edu



Outline

- Motivation
- Collaboration
- CWB Core
- CWB-enabled Applications
 - GLM Validation Tool
 - CWB-AES Integration
 - Giovanni Integration
 - NEOS³ Integration
- Conclusions and Plans

Motivation

- Scientists increasingly find themselves working with a more diverse set of data, complex science algorithms, and models.
 - Each new dataset, algorithm, or model - own knowledge
 - Acquiring this knowledge - difficult and time consuming
 - One way - social collaboration (virtually)
- Adopting social networking and collaboration
 - Only a few have been successfully customized to address and impact Earth science research modalities
 - New set of analysis tools to leverage the collaboration infrastructure
 - Adoption: inhibited by the steep learning curve

Approach

- Collaborative Workbench (CWB)
 - Augment a scientist's current computational research environment
 - Provide a science algorithm development environment that seamlessly integrates the researcher's desktop with a cloud infrastructure.
 - Focus on enabling sharing of research artifacts + collaborative development of algorithms
 - Interpretation of data and analysis results as well as their knowledge (in the form of annotations).
 - Accommodate various modes of collaboration, in order for it to be effective.
 - Let scientists focus on science, rather than learning collaboration tools.

-COLLABORATION-
YOU + OTHER + GOAL

**COLLABORATION HAS
DIFFERENT LEVELS**



REALTIME

**You must be
present**

Chat
Skype
Messenger
Meeting



ASYNC

You will be present

Email
Forum
Dropbox

Managing Collaboration

Versioning + Provenance

Ownership

Who created/modified it

Assumptions

Lineage

Quality

Replication Recipe

....

Science Collaborations

- Occur when 2 or more people work together to achieve a common goal, result or project
- Cover all levels of collaboration
- Find ways to share reuse data, programs, workflows, experiments ..
- Require feedback and iteration
 - Members review each others work and make revisions
- Track – who, what, when, how, why, which

Collaboration Systems

- Need to provide a mechanism for:
 - Communication
 - Content management
 - Workflow for revisions

Components of Collaboration Systems

- Data
 - Cloud for scaling
- Software
 - Cloud for sharing, scaling, and executing
- People
 - Different roles, levels of control, groups, ...

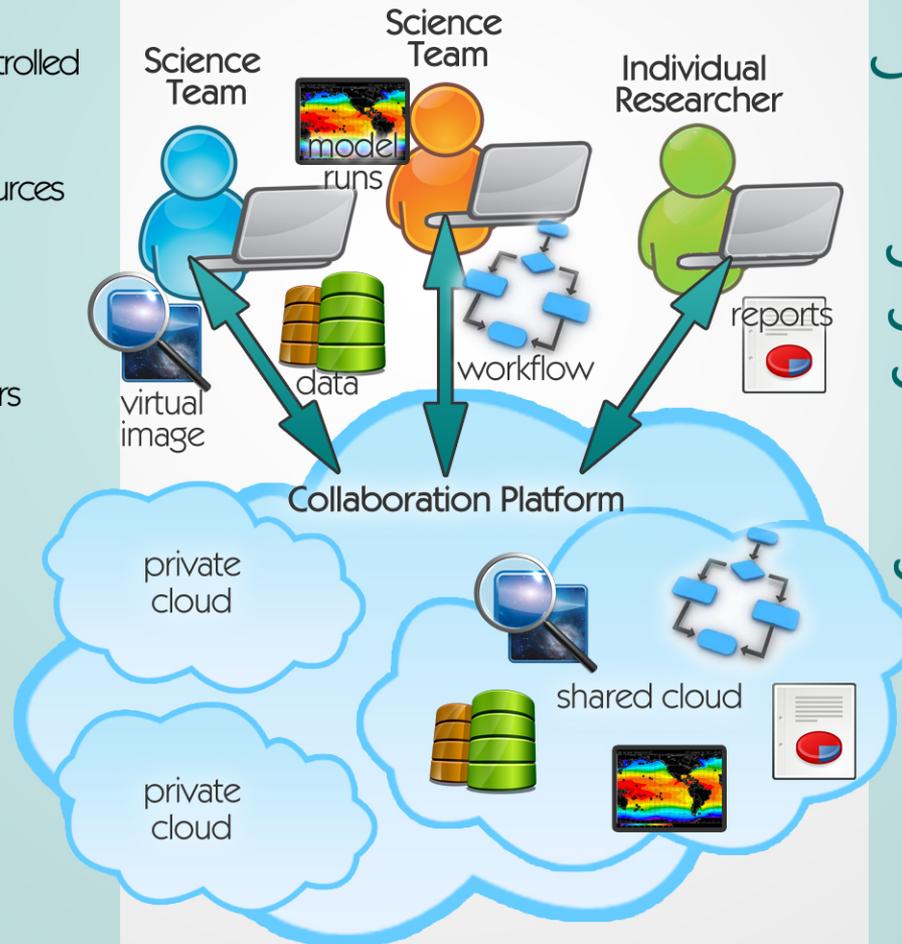
CWB CORE

Collaborative Workbench (CWB) to Accelerate Science Algorithm Development

Sharing Knowledge is at the heart of science, yet it is challenging for researchers to effectively share information and tools

Goals

- An architecture for scalable, controlled collaboration
- Selective sharing of science resources
 - among individuals
 - within science teams
 - with the entire science community.
- Software that fits how researchers currently do scientific analysis to promote adoption



Benefits

- Accelerate science algorithm development by distributed science teams
- Reduce redundancy
- Improve productivity
- Securely share all science artifacts (data, information, workflow, virtual machines)
- Generalizable to support collaborative science algorithm development for other mission and model enterprises

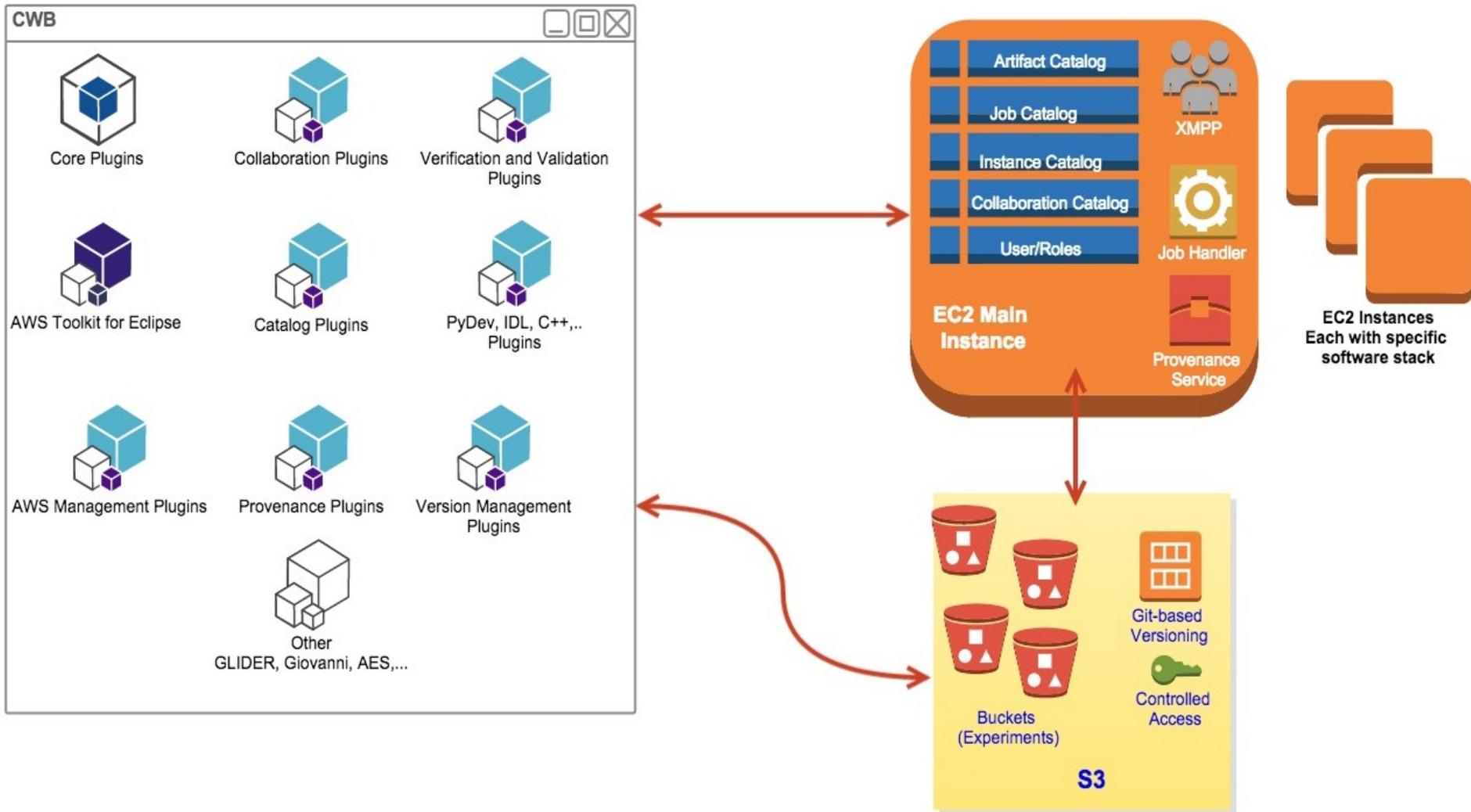
Project Objectives

- Investigate different science collaboration modalities:
 - Shared Resources, Local Computation – Dropbox Model.
 - Shared Resources, Cloud Computation
 - Shared Virtual Machine Instances
- Build **core** components required for an Earth Science Collaboratory
 - Collaborative Workbench (CWB)
 - Cloud-Interfaces
 - Catalogs

Workbench - CWB

- Sophisticated code editing, navigation, and management tools for development and execution.
- Familiarity
- Eclipse platform
 - Can be specialized for particular requirement.
- Plugin architecture

CWB Components



Core Components

- Communication Plugins
 - Shared Code Editing, Chats, Send files
- Catalog for Data/User/Job management
 - Search, Version, Provenance
- Development tools
 - IDL, Python, ADaM, Visual workflow composer
- Cloud job submission
 - Submit job to multiple instances (multiple software stacks, versions)
 - ENVI Service Engine backend
- Cloud Resource Management
- **Share public/individual**
 - Share Local, Share VM

Versioning

Data and source code versioning using git

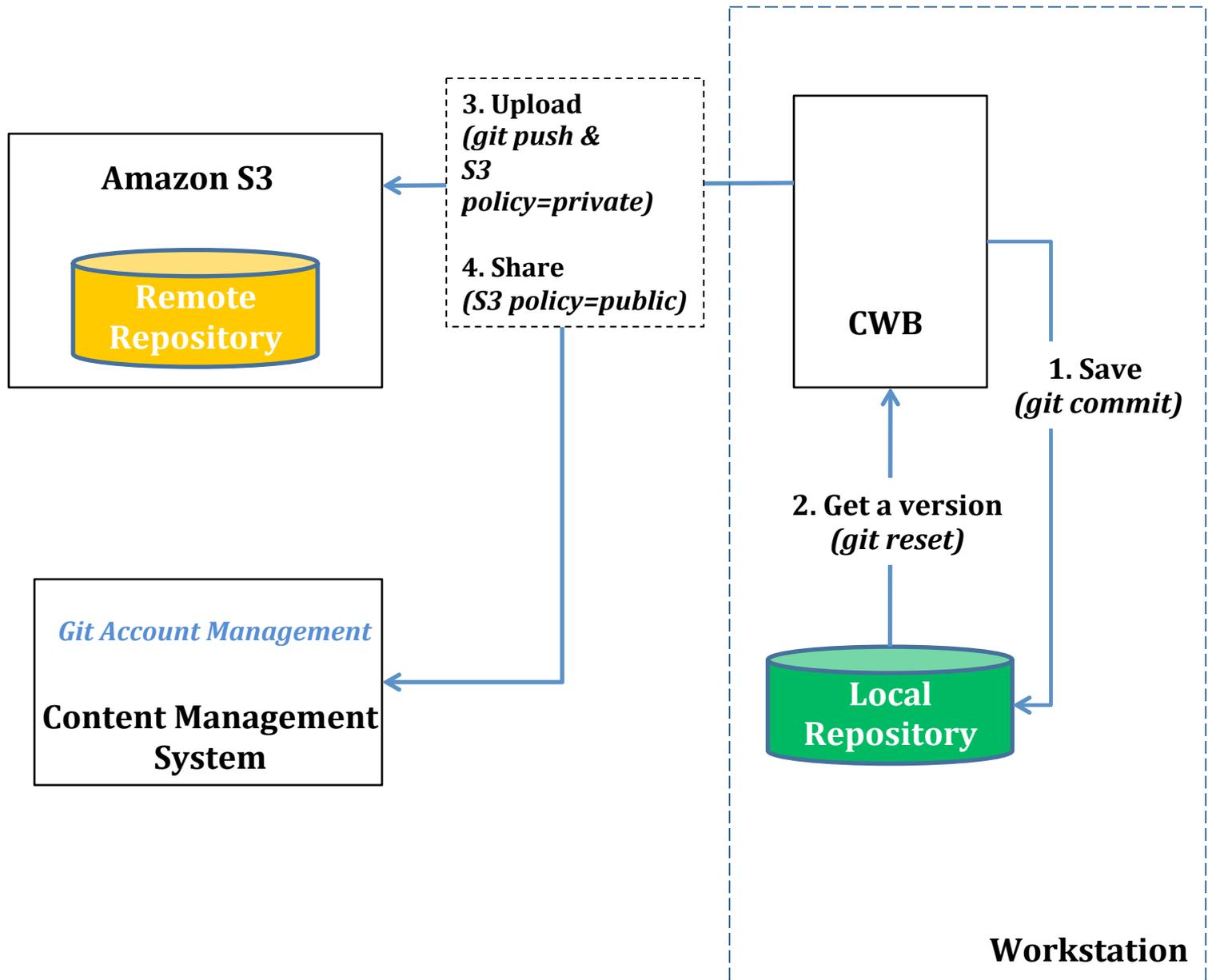
Why git?

- **git** allows distributed versioning
- Integrates with S3
- Files are compressed
- Only change (deltas) stored

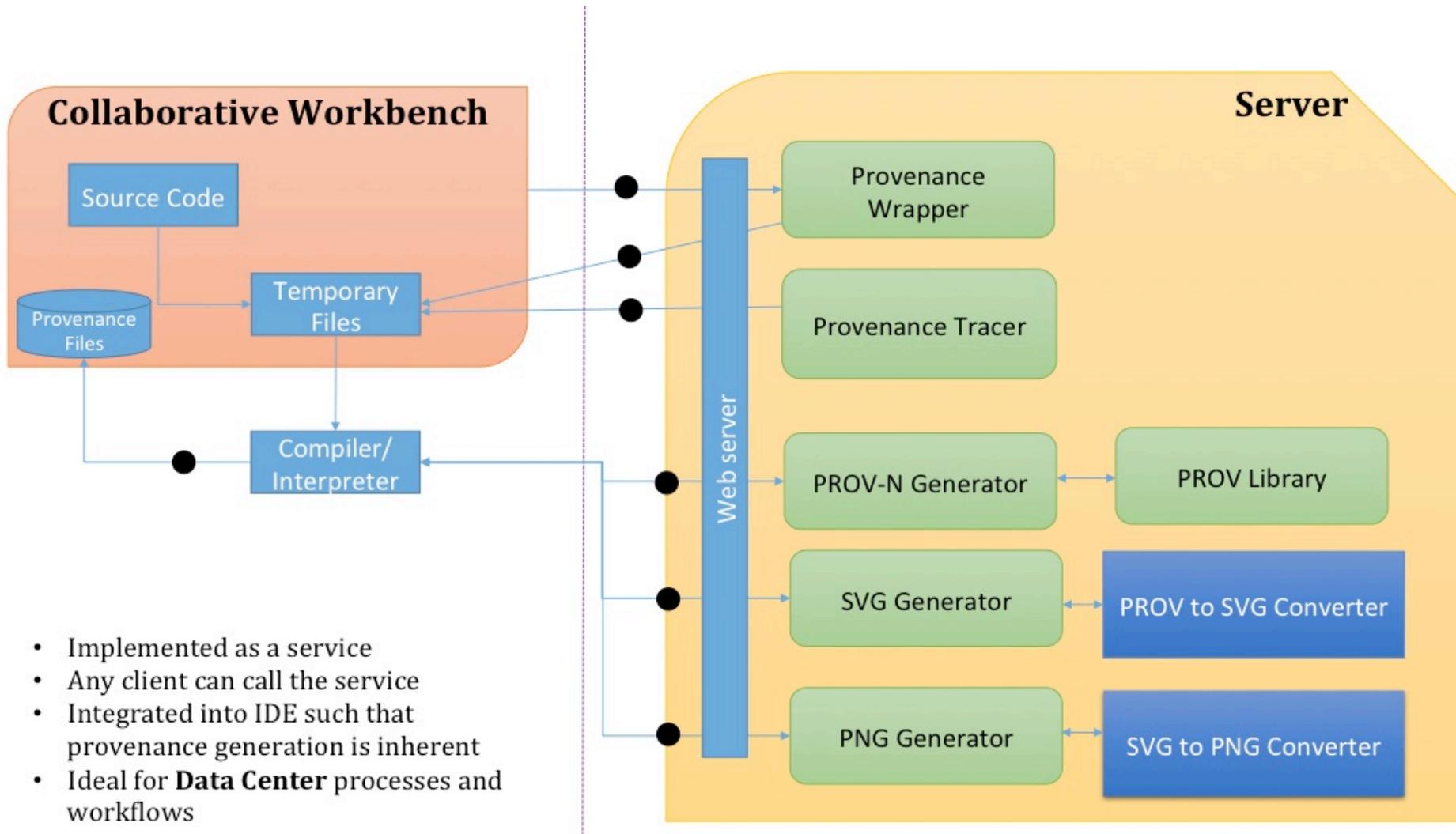
- **Plugins**

- **Hide all the version control nomenclature from scientist and provide versioning implicitly – very important**
- **CWB + Cloud + Git with single sign on**

CWB versioning and sharing using Git



Auto Provenance for CWB



- Implemented as a service
- Any client can call the service
- Integrated into IDE such that provenance generation is inherent
- Ideal for **Data Center** processes and workflows

CWB: Auto Provenance Generation (Python)

Sharing/Versioning

The screenshot displays the Collaborative Workbench interface. The main window title is "Collaborative Workbench - C:\Users\sshrestha\cwb.workspace\shree - shree [ss0108@uah.edu]". The interface is divided into several panels:

- Search Panel:** Located at the top left, it contains a "Search Workflows" input field and a "Search" button.
- Shared Experiments Panel:** Below the search panel, it lists shared experiments: aogs-kuo, giovanni-experiment, lynnestrial, neos1, and shree-experiment.
- My Experiments Panel:** This panel is expanded to show "Personal Sandbox" for three users: giovanni-experiment, neos1, and shree-experiment. Under "neos1 - Personal Sandbox", a "python-workflow" folder is visible, containing "sample_output.txt" and "test.py".
- Other Experiments Panel:** At the bottom left, it lists other experiments: ajinkya-experiment1, ak-experiment1, akulkarni-ex2, aogs-demo, aogs-kuo, erwalter-trial, lynnestrial, myexp1, rahul-cmac, somali-jet, and xydff.
- Code Editor:** The central pane shows a Python script named "test.py" with the following code:

```
target = open("sample_output.txt", 'w')
for x in range(0, 3):
    target.write("Writing data to a test file" + "\n")
target.close()
```
- Program List Panel:** At the bottom right, it shows a list of programs: gsf_to_glider, ITSC_ModisToGlider, ITSC_ImageToGlider, ITSC_GsfToGlider, ITSC_GliderToArff, ITSC_CvtImageToGif, and ITSC_CvtGifToImage.

The status bar at the bottom left indicates the current file path: "neos1/python-workflow/test.py".

Provenance

The screenshot displays the Collaborative Workbench (CWB) interface. The main workspace is a large, empty area with a central rounded rectangle containing the text "CWB Provenance Demo".

The interface includes several panels:

- Search:** A search bar with the text "Search Workflows" and a "Search" button.
- Shared Experiments:** A list of shared experiments: aogs-kuo, giovanni-experiment, lynnestril, neos1, and shree-experiment.
- My Experiments:** A list of personal sandboxes: akulkarni-ex2 (Personal Sandbox), aogs-kuo (Personal Sandbox), and neos1 (Personal Sandbox). Under akulkarni-ex2, there are two workspaces (w1 and w2). w1 contains folders for input and output, and files test1.py, test2.py, and test3.py. w2 is empty.
- Other Experiments:** A list of other experiments: ajinkya-experiment1, ak-experiment1, aogs-demo, erwalter-trial, giovanni-experiment, lynnestril, myexp1, rahul-cmac, shree-experiment, somali-jet, and xydff.
- Program List:** A list of programs: gsf_to_glider, ITSC_ModisToGlider, ITSC_ImageToGlider, ITSC_GsfToGlider, ITSC_GliderToArff, ITSC_CvtImageToGif, ITSC_CvtGifToImage, and ITSC_CvtArffToImage.

The top of the window shows the title bar with the path "C:\Users\akulkarni\Desktop\CMAC\eclipse-standard-kepler-SR2-win32-x86_64 (1)\runtime-edu.uah.itsc.cmac.rcp.cwb(1) - user1 [user1@abc.com]" and standard window controls. The menu bar includes File, Edit, Workbench, Window, Help, Run, Search, and Workflow Options. A "Quick Access" bar is visible in the top right corner.

***CWB-ENABLED
APPLICATIONS***

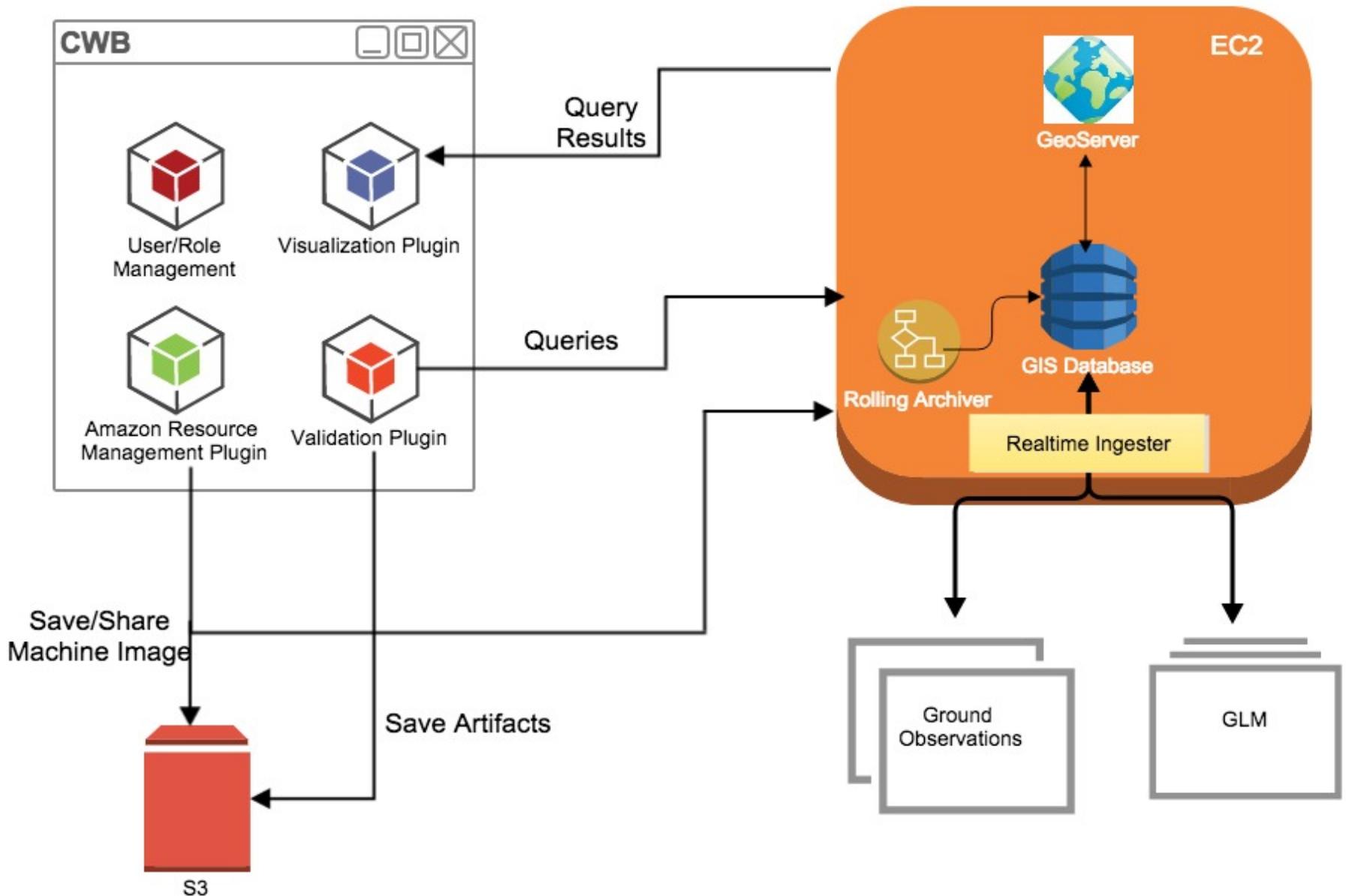
Applications and Integrations

- GLM Verification and Validation
- CWB-AES
- Giovanni upload
 - Upload dataset
 - Run workflow on dataset
- Giovanni
 - Upload
 - GLIDER
- NEOS³
 - netCDF packaging of scattering database
 - Python module: job handling

GLM Verification and Validation

- Validation of the Geostationary Lightning Mapper (GLM) instrument, scheduled to launch on GOES-R in 2016.
- Close collaboration with LIS SCF.
- Uses real time flash rate datasets, such as Earth Networks Total Lightning Network, the National Lightning Detection Network (NLDN) and Vaisala's Global Lightning Dataset (GLD360) that have been ingested into a GIS database
- Displays thematic layers of each dataset
- Performs real time analysis of discrepancies between GLM and ground based sensors; display interactive statistics, histograms..
- Subsets spatially and temporally
- Saves anomaly as image/video
- Uses CWB features to share artifacts
- Controlled access

GLM Verification and Validation Architecture



GLM Verification and Validation

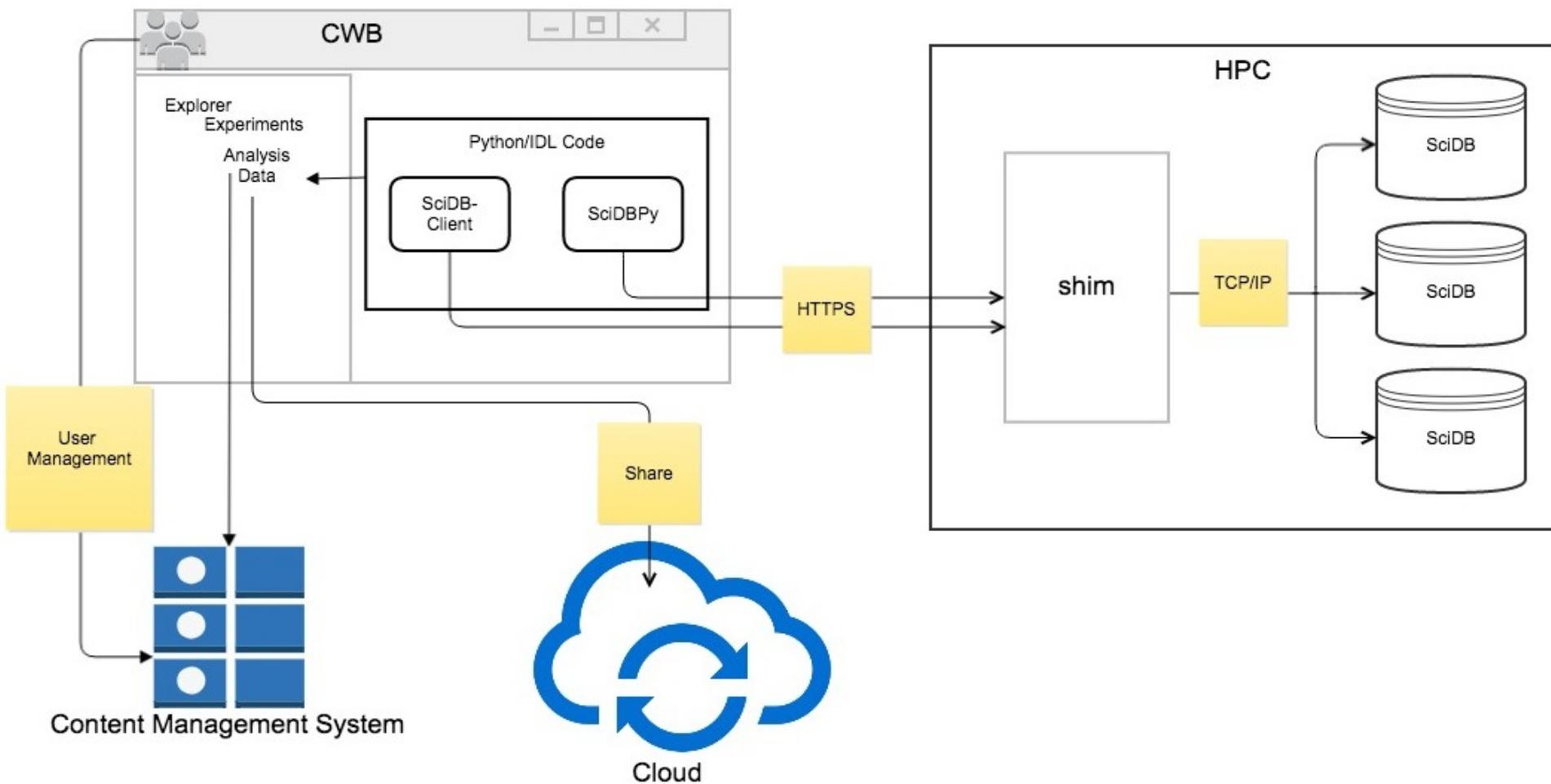


CWB-AES Integration

Automated Event Service (AES)

- AIST, PI: Tom Clune
- Observing and studying events
- System being developed to methodically mine custom defined events

CWB-AES Integration





Starting CWB

**Authentication and
authorization
Handled by CWB**

**Point to the appropriate
HPD End Point**

Search

Search Workflows

- Shared Experiments
- aogs-kuo
 - giovanni-experiment
 - lynnestrial
 - neos1
 - shree-experiment

- My Experiments
- somali-jet - Personal Sandbox
 - somalijetdetection
 - Somali_F8_1990
 - AES_Client.py
 - AES_OnsetDate.py

MyExperiment on Somali Jet Detection

- Other Experiments
- ajinkya-experiment1
 - ak-experiment1
 - akulkarni-ex2
 - aogs-demo
 - aogs-kuo
 - erwalter-trial
 - giovanni-experiment
 - lynnestrial
 - myexp1
 - neos1
 - rahul-cmar
- 0 items selected

Experiment AES_Client AES_OnsetDate

```
#!/usr/bin/python
#####
# AES_Somali.py
#####

import os
import httplib
from AES_Parameters import AES_Parameters
from AES_GeoArray import AES_GeoArray
from AES_WebClient import AES_WebClient
from AES_Contour import AES_Contour
from AES_PlotLayout import AES_PlotLayout

# Python requires true type fonts to draw text, specify path to ttf file here
#glTypeface = "/usr/share/fonts/truetype/freefont/FreeSans.ttf"
#glTypeface = "/usr/share/fonts/liberation/LiberationSans-Regular.ttf"
glTypeface = "/Library/Fonts/Courier New Bold.ttf"

#####
# Parameters for Somali Jet region and data set
#####

glSomaliParams = AES_Parameters()
glSomaliParams.mDataSet = "nil"
glSomaliParams.mStartYear = 1996
glSomaliParams.mStartMonth = 5
glSomaliParams.mStartDay = 1
glSomaliParams.mEndYear = 1996
glSomaliParams.mEndMonth = 5
glSomaliParams.mEndDay = 1
```

Import AES Library and use All the functions to write my own Script to query data, and execute optimized detection algorithms

Program List Contacts EC2 AMI View EC2 Instance View Console

No consoles to display at this time.

The image shows a code editor window with two tabs: 'AES_Client' and 'AES_OnsetDate'. The code in the 'AES_Client' tab is as follows:

```

# Create an http connection
client = AES_WebClient()
#client.Connect("/Users/jrushing/Projects/NASA/Connect/tremor3.txt")
client.Connect()

# Generate a subset
subsetId = GenerateSubset(client, sat, year, month, day)
if (subsetId is None):
    return False

# Generate a subset
projectId = client.ProjectQuery(subsetId, "wind")
if (projectId is None):
    return False
print "SomaliJet - Generated Project Array " + projectId

# Get the data array for the subset
npArray = client.GetSlice(projectId, 0)
if (npArray is None):
    print "Error: Could not get data for " + projectId
    return False

# Convert to AES geo array
dataArray = AES_GeoArray()
dataArray.InitFromNumpyArray(npArray)

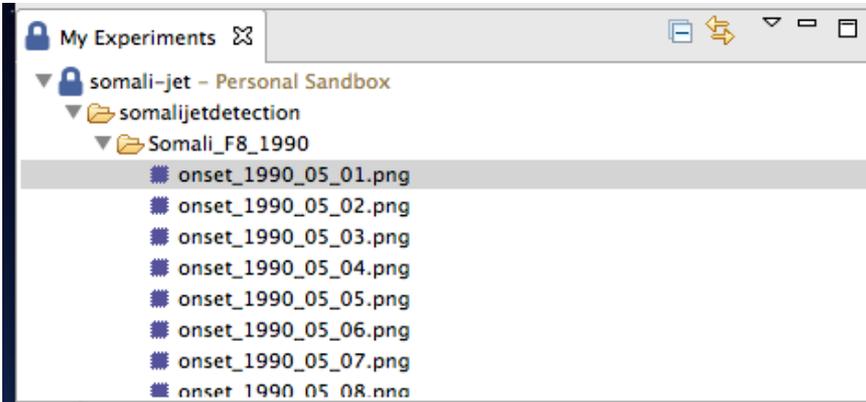
# Get month string
mstr = "May"
if (month == 6):
    mstr = "June"

```

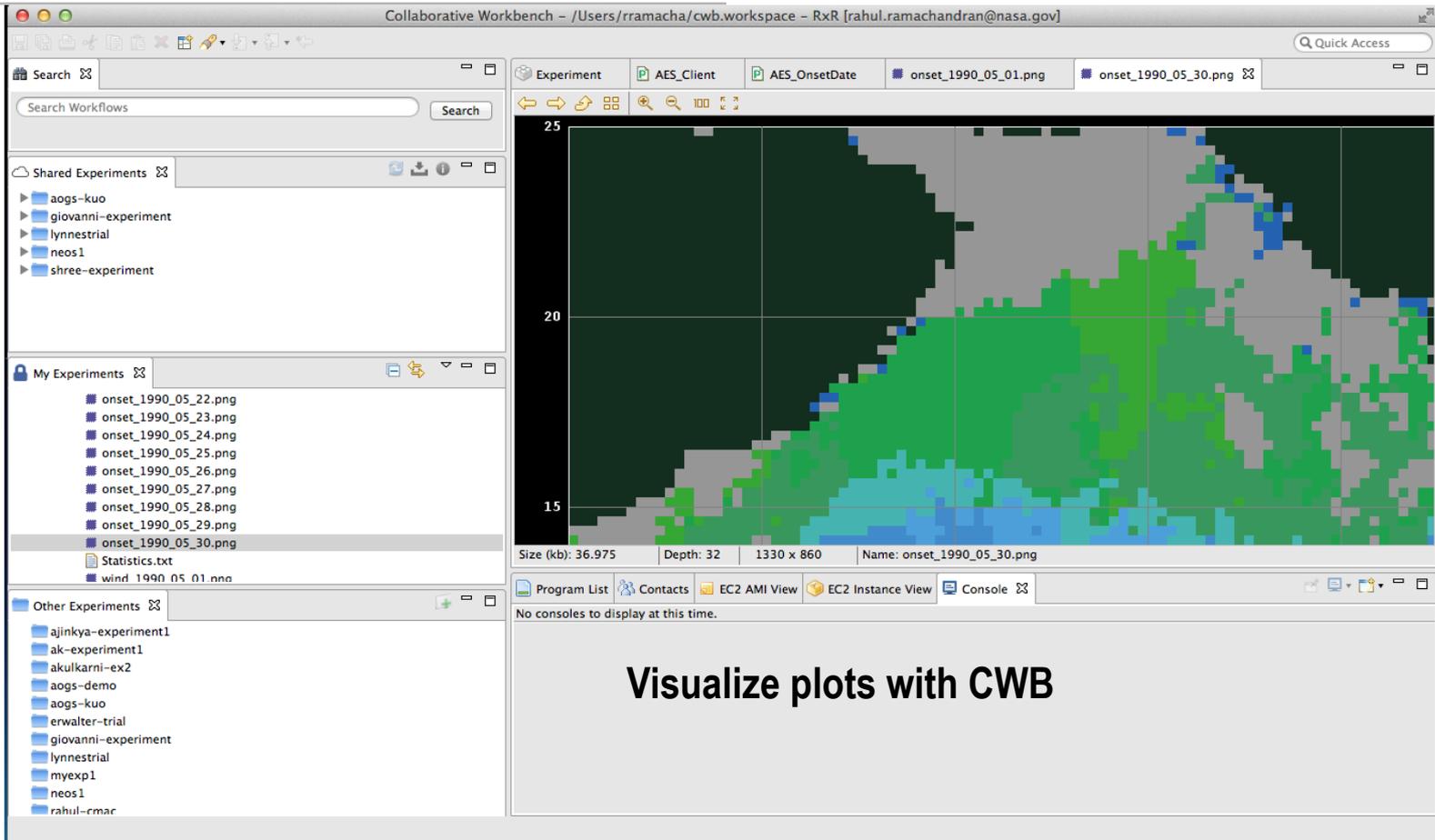
The text "Example methods" is overlaid on the code. A blue arrow points from the bottom of the code editor to the first bullet point below. On the right side, a context menu is open, showing options like 'Undo', 'Save', 'Run As', and 'Add Experiment'. The 'Run As' option is highlighted, and a sub-menu is visible with '1 Python Run' and '2 Python unit-test' options.

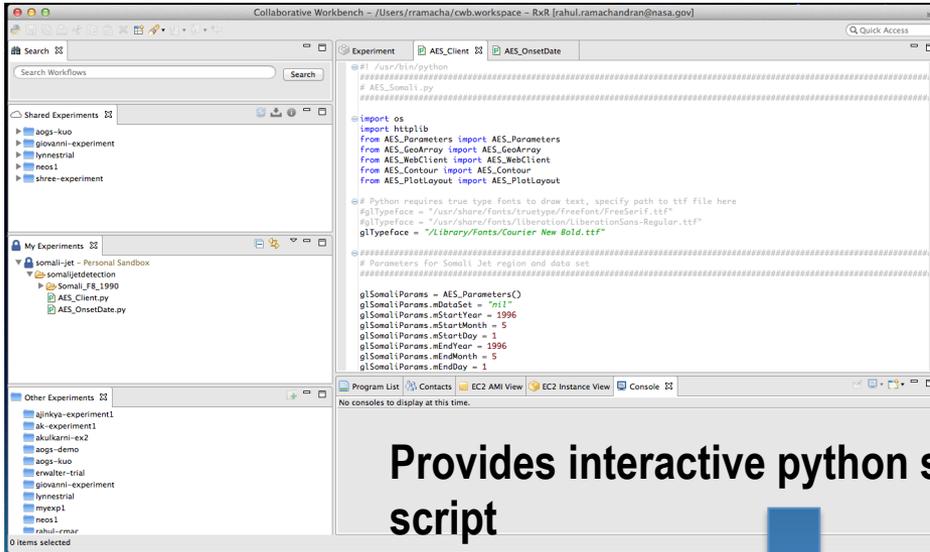
- Execute optimized data parallel queries on a remote HPC
- Integrate local data processing and analysis seamlessly

Run my script from CWB

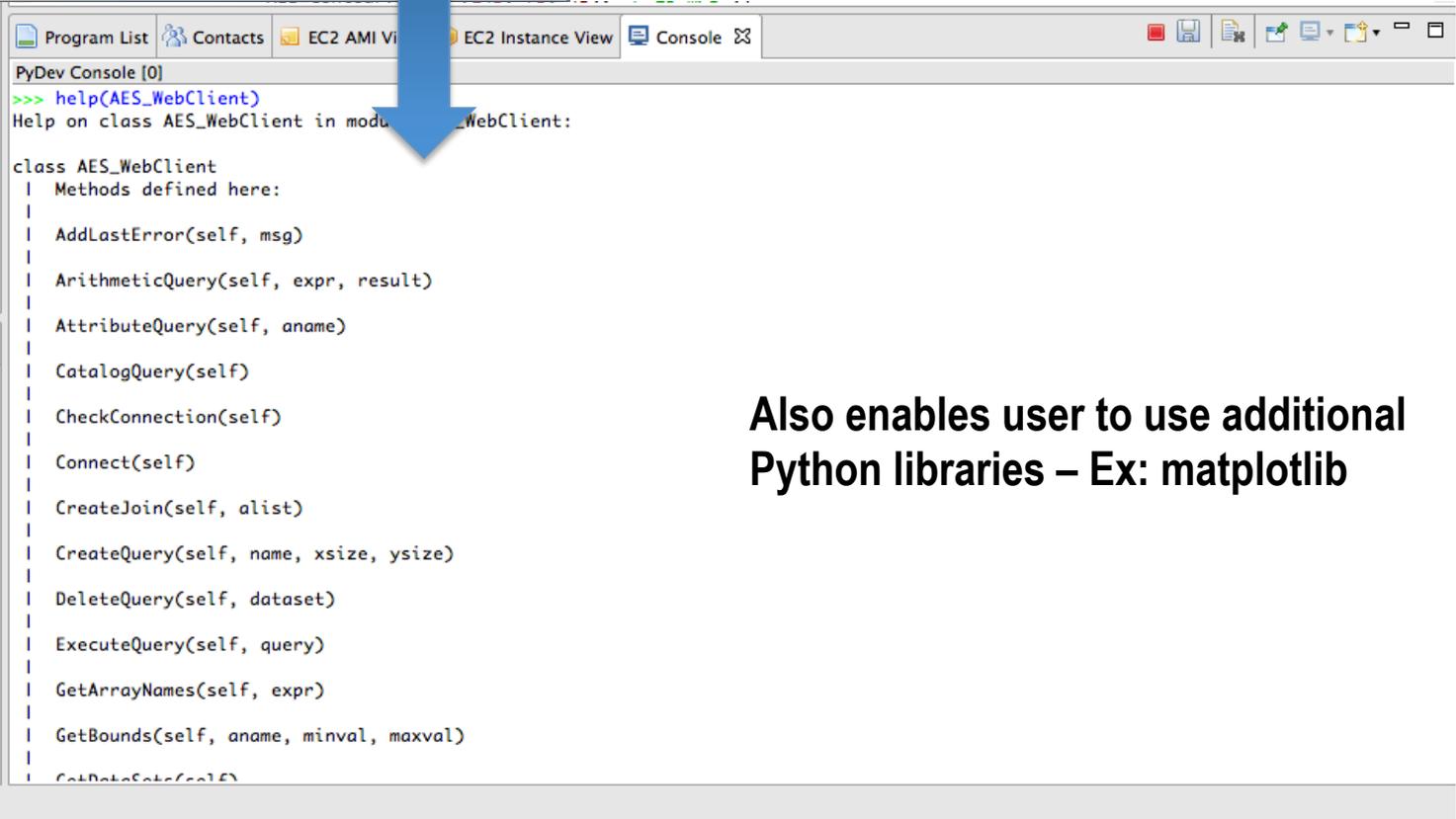
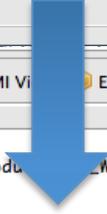


Results saved in MyExperiment folder





Provides interactive python shell to try methods while writing the script

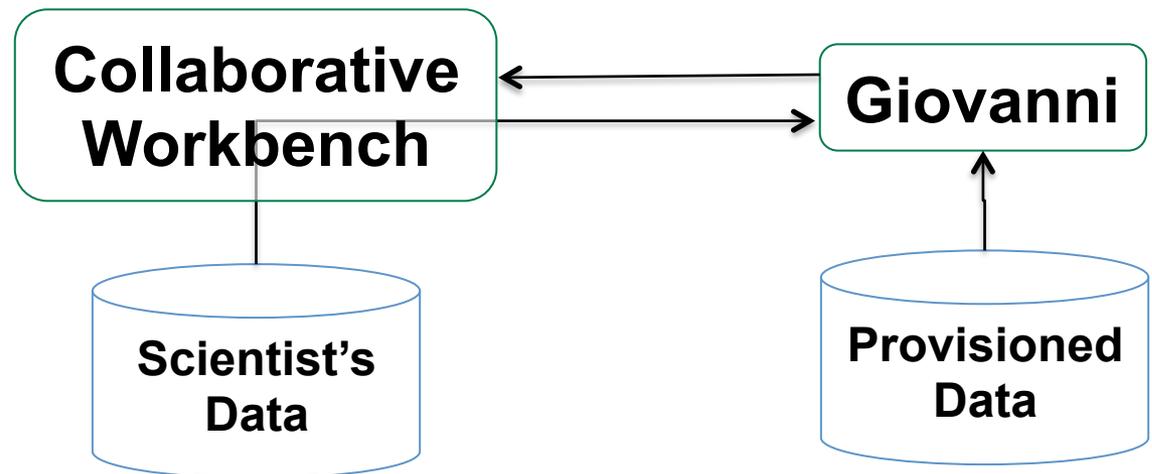


Also enables user to use additional Python libraries – Ex: matplotlib

CWB-GIOVANNI

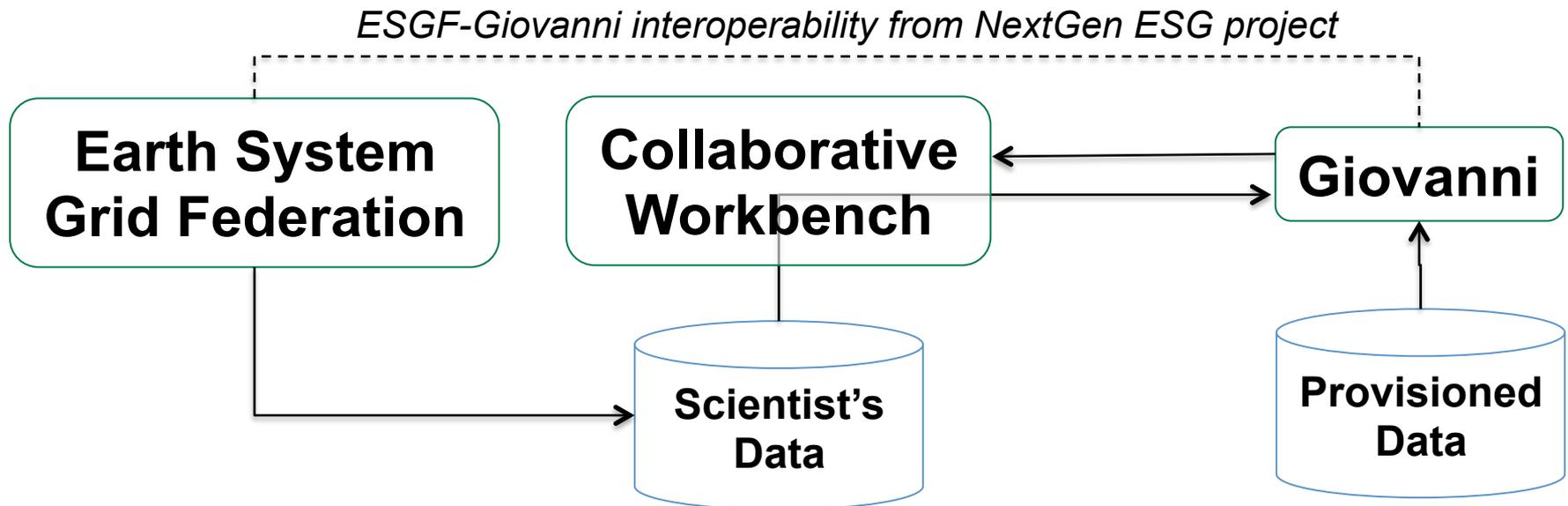
Giovanni Integration

- Giovanni
 - Goddard Interactive Online Visualization ANd aNalysis Infrastructure
 - Popular online visualization and analysis tool
 - Exploration and comparison of remote sensing data +model output
- Integration with CWB
 - CWB python to execute a Giovanni workflow
 - CWB drop-in to upload scientist's own data
 - Giovanni modifications to support workflows with scientist's own data
 - User-Supplied Data Use Cases
 - Upload "my" data to Giovanni for comparison with data already in Giovanni (completed)
 - Upload my data to Giovanni to allow other users to interact with it (90% completed)



Giovanni Integration++

- Earth System Grid Federation
 - NextGen ESG project, PI = C. Mattmann
 - Main Goal: publish DAAC data to ESGF
 - Spinoff: ESGF data turns out to be “Giovanni-friendly”
- CWB “Integration” with ESGF
 - Acquire data from ESGF
 - Upload ESGF data via CWB/Giovanni User-Supplied Data Uploader
 - Run ESGF data in Giovanni workflows



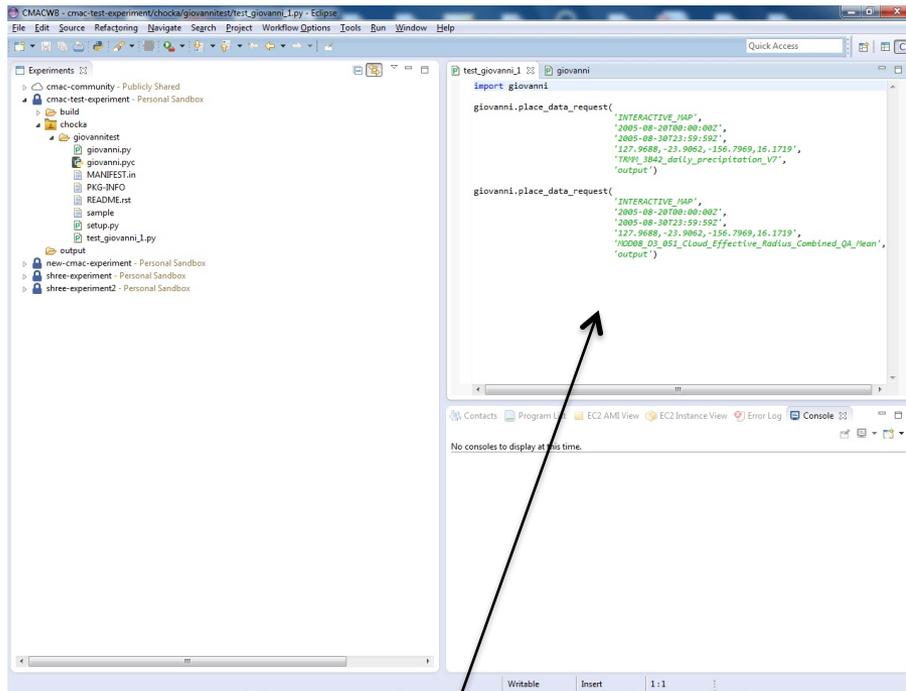
GLIDER Integration

- GLIDER – Globally Leveraged Integrated Data Explorer for Research
 - Desktop tool to easily visualize, analyze and mine satellite imagery
 - Allows image enhancements and pattern recognition algorithms to be applied on the satellite imagery
 - Visualization with **NASA World Wind** 3D tool
 - RCP application consisting of Eclipse plugins
 - Integration to CWB included dropping the GLIDER plugins into CWB dropins folder

CWB Giovanni-GLIDER plugin: Use Case

- Demonstrate that researchers can use CWB to download data using python code to their workspace and apply GLIDER plugin for image view of the data and use World Wind plugin to visualize and compare.
- Steps:
 - Access Giovanni service via Python Module – Download data to personal space in CWB
 - Convert the result data to GLIDER format
 - Use GLIDER (Eclipse Plugin) to visualize and enhance the data
 - Display the enhanced image on the NASA World Wind globe (Eclipse Plugin)
 - Overlay multiple images on the globe to see the correlation.

CWB Giovanni Python Module



Access Giovanni via Python module with PyDev

Data:

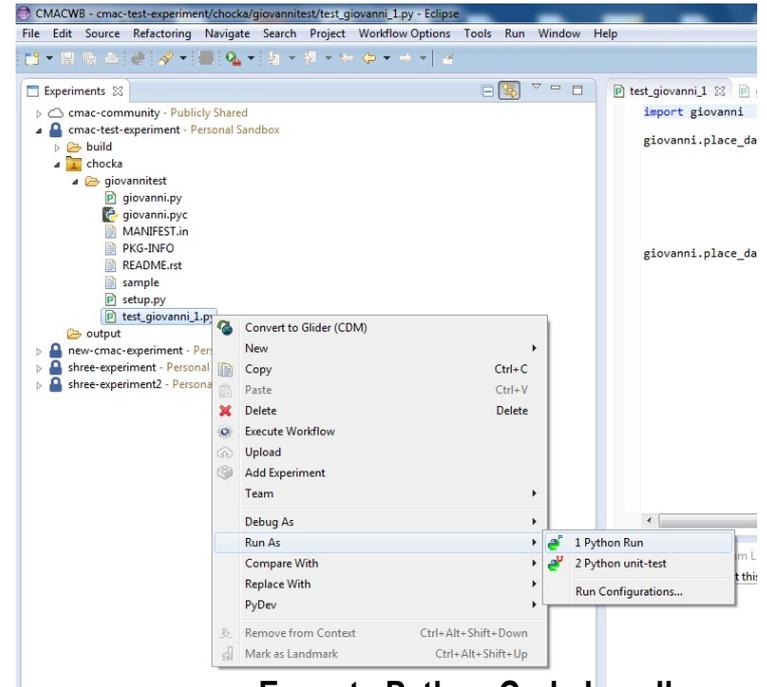
TRMM 3B42 Daily Rainfall Estimate V7 0.25 deg.
Cloud Effective Radius Combined QA Mean

Temporal Bounds:

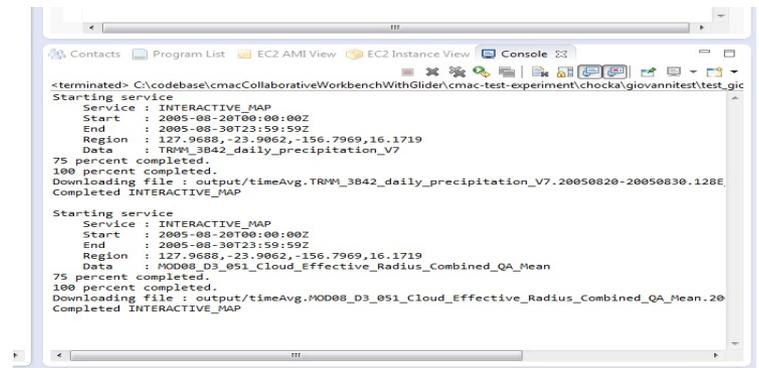
2005-08-20 to 2005-08-30

Spatial Bounds:

127.9688, -23.9062, -156.7969, 16.1719

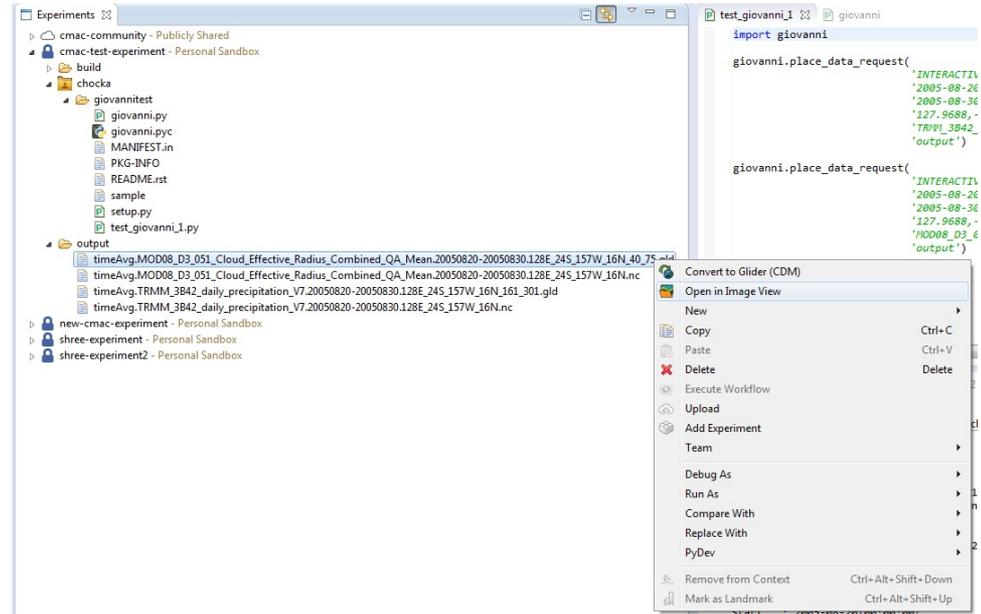
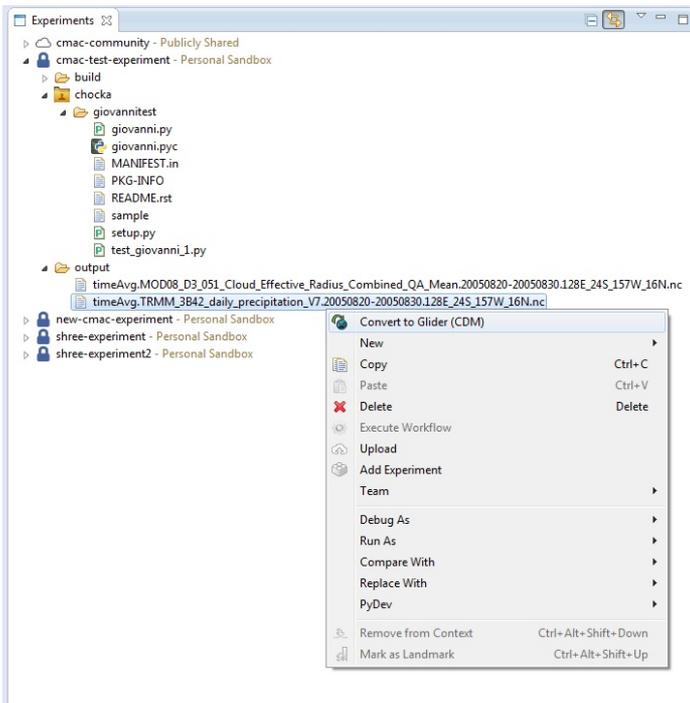


Execute Python Code Locally



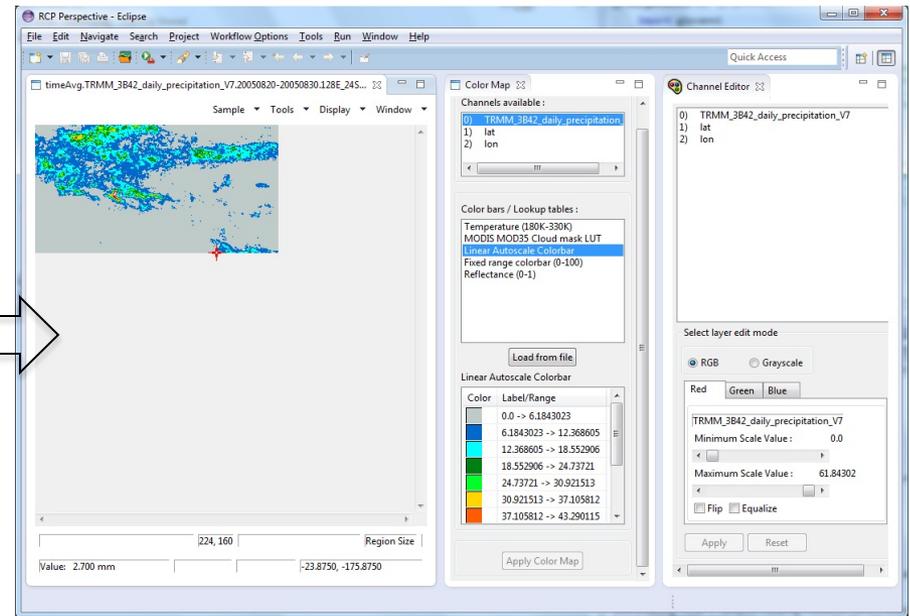
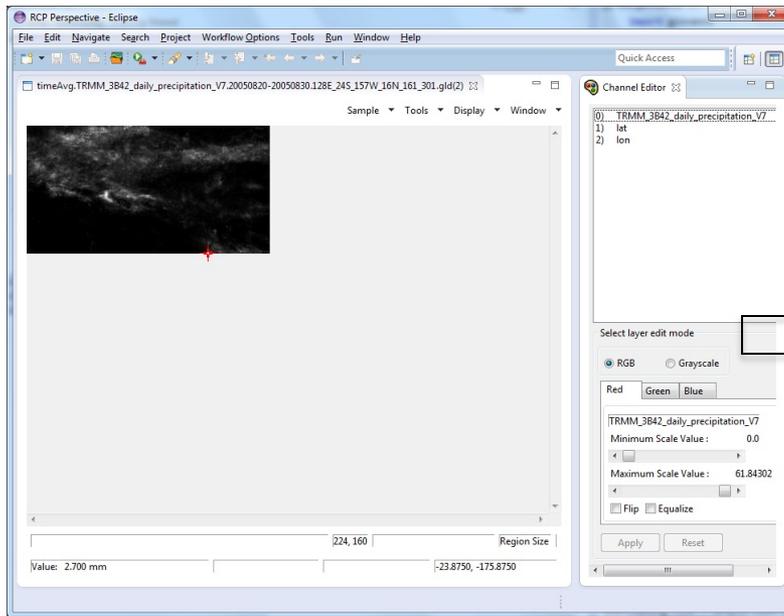
Check the log for completion

CWB GLIDER Plugin

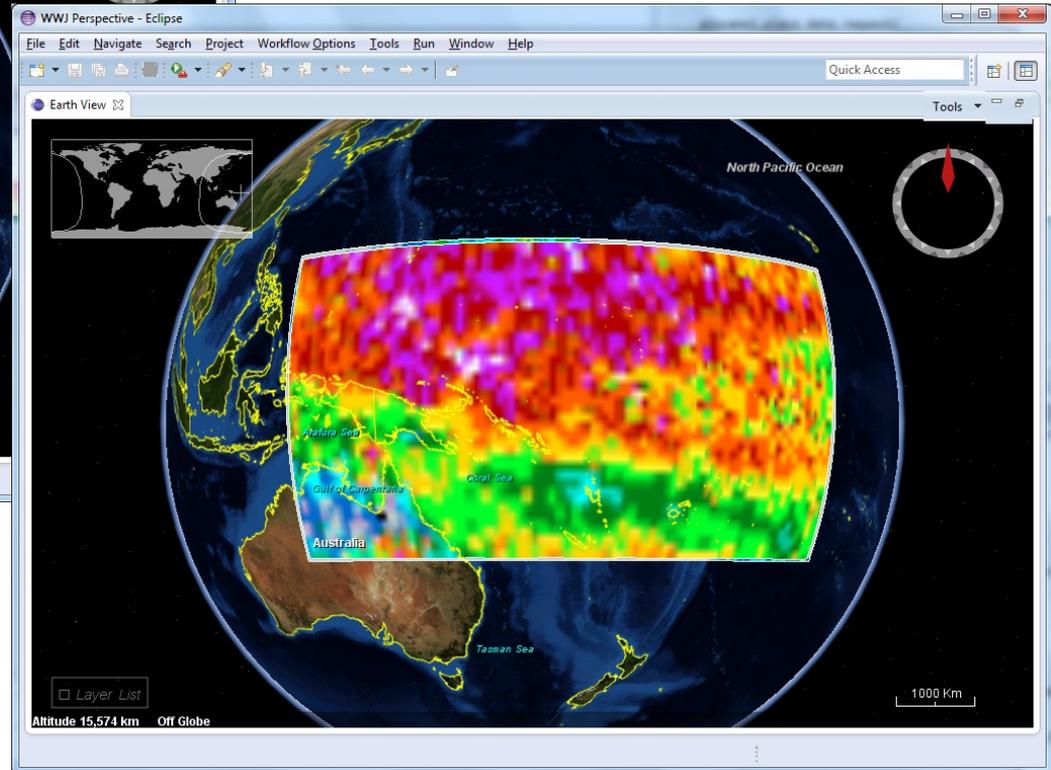
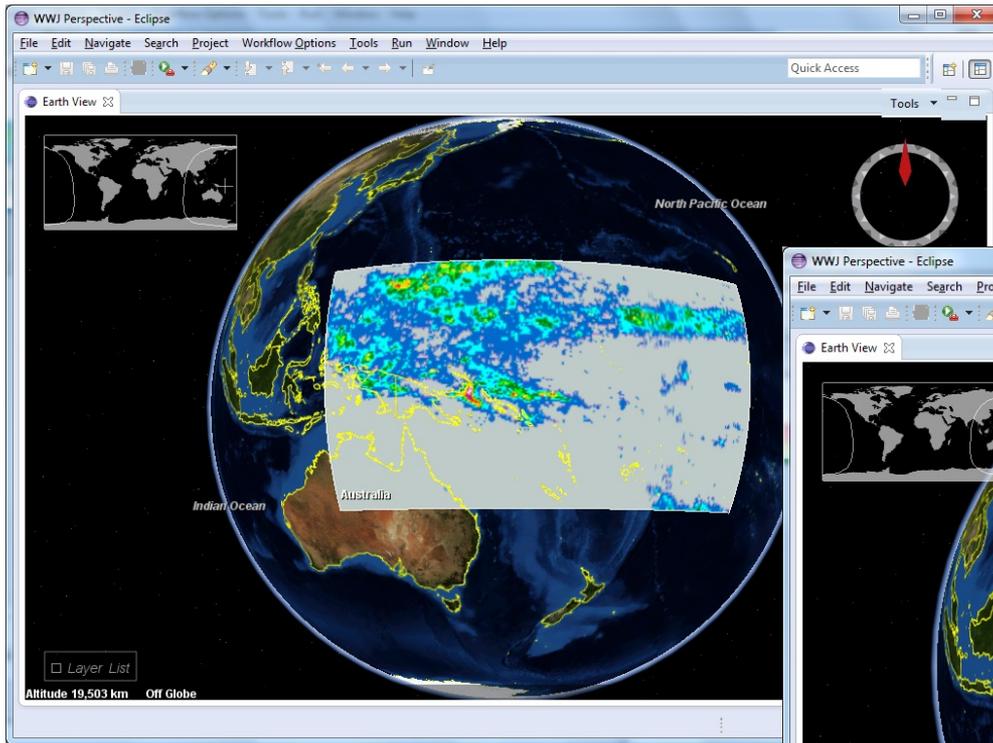


GLIDER functionalities readily available via contextual menus

CWB GLIDER Plugin – Image Enhancement



CWB GLIDER – NASA World Wind Plugin



- Layer images
- Visualize with context

- NEOS³: NASA Earth Observing System Simulation Suite
 - AIST: PI Simone Tanelli
 - Web-based integrated simulator for Earth remote sensing applications.
 - Equipped with start-of-the-art modules to enable the realistic simulation of satellite observables.
 - Providing an advanced, sophisticated, and user-friendly simulator package to be used by both scientists for research-oriented applications and by system engineers for an instrument design purpose.
 - Accessible via a web interface and capable of distributing computationally intensive tasks to remote servers such as those at the NASA Advanced Supercomputing (NAS) Division.

CWB-NEOS³ integration

- Creating NEOS³ Jobs using templates
- Polling NEOS³ for completion
- Retrieving outputs to CWB
- Demo

Lessons Learned

**“To complicate is easy, to simplify is hard.
To complicate, just add,
Everyone is able to complicate.
Few are able to simplify.”**

-Bruno Munari

Conclusion and Plans

- A framework for Science Collaboration
 - Augments existing tools
 - Reduced learning curve
 - Scalable
 - Extendable (Plugin Architecture)
 - Reusability for customizations
- More simplifications
- Integration with other tools based on plugin architecture

**Thanks to NASA CMAC and ESTO for
their support in our efforts to develop
this collaboration framework**