XMM-Newton SAS Virtual Development Environments

Aitor Ibarra & José Marcos on behalf of XMM-Newton SAS Team
Overview

1. XMM-Newton Science Analysis System (SAS) Introduction
2. SAS building process
   1. Infrastructure-as-Code
3. SAS code maintenance
   1. Platform-as-Code
4. SAS Virtual Development Environments
5. SAS on cloud infrastructures: Datalabs
6. What next...
The XMM-Newton Scientific Analysis System (SAS) is a freely distributed suite of programs for dealing with data from all XMM-Newton instruments. SAS is a collection of tasks (C/C++ & Fortran-[77,90]), scripts (perl & python) and libraries, specifically designed to reduce and analyze data collected by the XMM-Newton instruments. SAS is able to convert the XMM-Newton data from L0.5 (raw) to L3 (science products)

- React quickly to new developments in calibration
- Applies calibrations to raw data
- Optimally screen/filter the data
SAS development process

- SAS code → high dependency on the C/C++/F77/F90 compiler → introduces high dependency on the Operating System
- Third-party libraries makes the building process a bit more complex.
  - HEASOFT → used in harness tests → recommendation build from source code
  - Perl → evolves with time → version dependency
  - TexLive....
- Daily builds based on ftp and tgz
  - Geographically Distributed Code Development.
  - Continuous Integration System BEFORE the Concept was invented!!!
- Central Code Repository.
- SAS developers authenticated via gpg key-ring.
- Too complex to maintain this infrastructure and knowledge transfer.
SAS development process: (Dockerized SAS builder)

- Hardware virtualization
  - The first step to reduce costs.
  - But still too manual process.
- Infrastructure as Code
  - New paradigm to encapsulate and automatize the building process.
- Dockerization of the SAS building process:
  - Not easy because of the complexity
- Jenkins as SAS builder orchestrator

Infrastructure as Code:
- Dockerfiles
- Bitbucket code repository
- Nexus repository to store third-party libraries versions and SAS docker images

ENVIRONMENT
From configuring build environments manually:

To Docker containers with same tools and configurations:

- HEASoft
- CFITSIO
- PGPLOT
- WCSTools
- TEX Live
- GCC
SAS code maintenance:

• SAS development takes place on developer’s infrastructure
  • Coordination with IT team → Different projects – Different needs.
  • Infrastructure set-up effort is high.
  • Switching between Operating Systems (Linux/Mac OS) to maintain the code is not easy.
  • Switching between Operating Systems to work (Windows/Linux) tedious and costly.
  • Big organizations tendencies make this process even more complex → ESA365.

• SAS Virtual Development Environment
  • Docker deployment → Platform-as-Code (PaC)
  • Easy to integrate in Cloud Platforms.
  • Easy to integrate with common code repositories (Gitlab – Bitbucket)

• IaC + PaC
  • Centralized setup and maintenance.
  • Less HW dependency.
  • Simpler backup environment.
  • Security Improvement.
  • Access to many environments per user.
SAS Virtual Development Environments

Where are we now?

• Working on a stable prototype for SAS developers:
  • Users account management not easy….
  • User Authentication via ESA cosmos account (the same as the XMM-Newton Archive!!)
    • Still working how to set-up accessibility to different tools are repositories.
  • Working on SAS Bitbucket migration.
  • Writing documentation for SAS developers.

• Accessible via:
  • Local Visual Studio Code (testing other IDEs)
SAS virtual development environments

SAS Virtual Development Environment

SAS Technology Stack

Next Steps…
**SAS Virtual Development Environments**

**Steps to set-up SAS VDE**

1. Authenticate, Download and Run SAS Docker builder Image from ESA Nexus repository.
2. Download or have access to CCF repository in your host machine.

```bash
#> docker pull scidockreg.esac.esa.int:61900/xmm/sasbuilder/ubuntu-22.04/gcc-11.3.0:1.0.0-3

#> mkdir /pathToCCF/ccf

#> rsync -v -a --delete --delete-after --force --include='*.CCF' --exclude='*' sasdev-xmm.esac.esa.int::XMM_CCF /pathToCCF/
```
SAS Virtual Development Environments

Steps to set-up SAS VDE

3.- Run de SAS builder image
4.- Build SAS

#> ./docker_run_it.sh -o ubuntu-22.04 -g gcc-11.3.0

#> sasbuilder/bin/buildsas "" "11.3.0" devtrack dev
SAS Virtual Development Environments

Steps to set-up SAS VDE

5.- Start coding!!!

Select an option to open a Remote Window

Connect to Host...
Connect Current Window to Host...
Open Container Configuration File
New Dev Container...
Attach to Running Container...
Clone Repository in Container Volume...
Open Folder in Container...

Select the container to attach VS Code

/sasbuilderUbuntu-22.04_gcc-11.3.0 scidockreg.esac.esa.int:61900/xmm/sasbuilder/ubuntu-2...
SAS Virtual Development Environments

Steps to set-up SAS VDE

5.- Start coding!!!
SAS building process based on dockers

**Goal**
- Avoid hardware dependencies
- To have a bit more agile Continuous Integration System
- Get ready to move the integration system to a cloud platform

---

**Diagram**

- **SAS build**
- **Unit tests**
- **Functional tests**
- **E2E Scientific validation and regression testing**
- **E2E Scientific Validation And regression testing**
- **Quality metrics**
- **Vulnerability scan**
- **Functional tests**
Bringing Your Questions to the Data

There is a new paradigm, opening completely new opportunities for discovery—a data-intensive approach to science. In many domains, we have entered what could be called the golden age of surveys, with several large-scale projects, spanning decades, between finished, ongoing, and planned activities. ESA is...

Registration is subject to moderation

Personal and team workspace

Data Volumes

Pipelines

Datalabs
SAS in Datalabs

Data Volume Catalog

Domain
- Space Science (1)

 xmm

**XMM-Newton CCFs**
Data Volume for XMM-Newton Calibration CCFs repository. Data volume made available by XMM-Newton mission.
SAS evolution along the years: DL4SAS + Threads

- To help users to analyse XMM-Newton data, a set of Data Analysis Threads are provided to the community.

With the infrastructure of Python introduced in SAS, three experimental threads have been released under Jupiter Notebooks. These threads are not intended to be complete but to serve the purpose of illustrating how to use the Python interface to run SAS from a Jupiter Notebook.

**COMMON THREADS**

- Starting the SAS
  - SAS start-up

**IMAGE Gallery**

- All in one gas from raw data (ODF) to science products
  - Analysis chain for point-like sources: _xmmextractor_
  - Timing analysis with _XMMOS_

**EPIC RELATED THREADS**

- All in one gas from raw data (ODF) to science products
  - Analysis chain for point-like sources: _xmmextractor_
  - Step-by-Step

**JUPYTER NOTEBOOK THREADS**

- SAS start-up and event list manipulation
  - How to reprocess ODFs to generate calibrated and concatenated EPIC event lists
    - Python Notebook
  - How to filter EPIC event lists for flaring particle background
    - Python Notebook

**ESA DataLab**

- Filter files by name
  - _EPIC-bkgfltering_singleevt.fits_
  - _reprocessing.fits_
  - _SAS_image_viewer.fits_

**Bring your questions to the data**

There is a new paradigm, opening completely new opportunities for discovery—a paradigm shift in science. In many domains, we have entered a phase where we can start asking questions that were not possible before. This paradigm shift is particularly relevant in the field of astrophysics, where the XMM-Newton and Chandra observatories have revolutionized our understanding of the universe.

With the advent of new detectors and observing strategies, we have seen the development of powerful data analysis tools. One of these tools is SAS, which has evolved from a simple data reduction system into a comprehensive suite of applications for the analysis and visualization of astronomical data.

The evolution of SAS has been marked by the introduction of new features and capabilities, driven by the needs of the scientific community. One such feature is the integration of Python, which allows users to leverage the power of this popular programming language for data analysis within the SAS environment.

The provision of Data Analysis Threads, particularly through the use of Jupyter Notebooks, is a testament to this evolution. These threads are not only educational tools but also practical demonstrations of how to use the Python interface to run SAS for specific tasks, such as processing event lists or analyzing point-like sources.

In conclusion, the evolution of SAS has been characterized by innovation, adaptability, and a commitment to meeting the needs of astronomers around the world. As we continue to push the boundaries of what we can learn from the universe, SAS and its associated tools will remain at the forefront of astronomical research, providing the necessary tools to explore the mysteries of the cosmos.
SAS on cloud infrastructures: Datalabs/SciServer

- SAS is a docker can be used in any cloud platform.
- Scientific cloud platforms (Datalabs and SciServer) uses Jupyter Lab as user interface.
- SAS can be adapted to these platforms as long as we provide to user the basic functionalities to work with XMM-Newton data:
  - Image visualization
  - Light-curve visualization
  - Interactivity with these two functionalities
- Currently working on how to add interactivity in SAS Jupyter Lab environments using:
  - jpyjs9
  - lcviz
Future Work

- Improve the SAS DevOps infrastructure
  - Fully automatic SAS building deployment indifferent cloud environments

- Improve/automatize SAS Virtual Development Environments

- Improve SAS & Datalabs (cloud) usage
  - Improve the SAS python infrastructure to help users to create their own scripts.
  - Interactivity
    - Source a background regions
    - Good Time Interval selection

- More things to come…