Feasibility Report
11/11/2022

USGS AirFlow Processing Pipeline

Team ARES

Sponsored By:
Trent Hare

Team Mentor:
Vahid Nikoonejad Fard

Team Members:
Hunter Woodruff (Team Lead)
Quinton Jasper
Chris McCorkle
Isaiah Raspet
Richard McCormick

I. Introduction

With each success that NASA accomplishes, landing rovers on to the surfaces of foreign planets and conducting groundbreaking research, it's easy for the average person to overlook the crucial underlying sciences that support such efforts, especially in the early stages. Questions such as: “How can this rover land safely on planet Y?” or “How will this rover navigate the surface of planet Y once it is landed?” require reliable answers before any technology can set foot in outer space. In fact, NASA is not alone in its effort to answer such questions.

The United States Geological Survey (USGS) Astrogeology Science Center is a government research center based in Flagstaff, Arizona. Their mission is to collect, analyze and publish cartographic and geological data, which helps both researchers and the general public to better understand the surface of not just Earth, but also other celestial bodies within our solar system. The USGS team accomplishes this via multiple pieces of software designed for specific tasks, such as file conversions, data compression, image rendering, etc. For the course of this project, Team Ares will be working with a specific package of USGS software, collectively known as ISIS3.

ISIS3 is a package of more than three-hundred individual applications that each handle unique roles in the image processing workflow. The sponsor for this capstone project, Mr. Trent Hare, has informed the team that, while ISIS3 is a complete product and works for their needs, its fragmentation makes it difficult for users to manage. In order to follow the workflow properly, users have to locate and understand each required application from within a linux terminal. It is also expected that users will need to establish custom workflows, where each may consist of many different combinations of applications based on the user’s needs. From the perspective of a researcher or an enthusiast, the current structure involves too many steps with too many complications that increase the time and energy required to obtain the desired outputs.

Our team has been tasked with assisting in the development of *a workflow management system*. That is, a system that makes the process of organizing, executing and monitoring a workflow more intuitive and accessible to more users. To our sponsor, this would be best accomplished with some sort of graphical user interface. This interface would allow for users to interact with their workflows much like a flow-chart. This *Directed Acyclic Graph* (DAG) would consist of nodes that represent an individual application, and the connections between these nodes show the direction of data input and output.

Before work can begin on this project, the team must conduct some exploratory research to discover what tools and technologies could assist with development of a final product. This report will document this process which will take into consideration the client’s specific requirements, implied solutions, and technical hurdles to overcome in order to *successfully* implement our client’s minimum viable product (MVP) in addition to any possible stretch goals.

Table of Contents

I. Introduction **2**

II. Technological Challenges **4**
- Pipeline software 4
- Graphical User interface 4
- Documentation 4
- Containerization 4

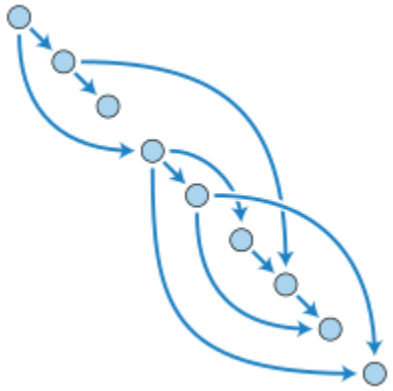
III. Technology Analysis **5**
- Pipeline Software 5
  - Apache Airflow 5
  - Potential Alternatives 5
- Graphical User Interface 6
  - Drag-and-Drop Functionality 6
  - Drag-and-Drop: Pipeline Construction 6
  - Drag-and-Drop: DAG Construction 6
  - GDAL and Leaflet 6
- Documentation 7
  - Issue Tracking 7
  - Documents 7
- Application Containerization 7
  - Docker 7
  - Docker Networks 7
  - Kubernetes 8

IV. Technological Integration **8**
- ISIS3 and Kalasiris 8
- Airflow and Elyra 8
- Codebase 9

V. Conclusion **9**

II. Technological Challenges

For each of the requirements outlined in the previous section, technological challenges will be identified and presented.



Pipeline software

Our main technological challenge is to implement a pipelining software that can integrate into the USGS' ISIS3 planetary imaging software. Specifically, the pipelines created will need to be represented in a Directed Acyclic Graph (DAG - seen left) wherein nodes - Python wrapped ISIS3 commands - can be "dragged and dropped" into the DAGs to create new and fluid pipelines. This way, researchers at the USGS can quickly, effectively, and easily generate images in a much more fluid motion than before

Graphical User interface

For the first technical challenge, as previously specified in the project requirements, our client requires that a functional user interface be accessible to the research team. By using the Airflow software, we can access the native GUI and DAG graph architecture to represent the ISIS3 command pipelines, thus allowing the process of creating/saving these pipelines to be visualized as specified.

Documentation

As previously mentioned, this project will require a video of the design architecture being run in a single instance from ISIS3 input to actionable images for the surface of Mars.. To accomplish this, the technologies used will be Apache's Airflow processing pipeline software, which will host our software's architecture and DAG's. Along with a video demonstration, our client specified a report of the design and implementation of our technologies in the product. This will require a technology that can track and house our implementations and improvements as we further the product, for which we are selecting Github.

Containerization

Lastly, our client recognized a possible future-proofing of our product. To have a fully realized product would possibly require cloud features to be translatable to our product, which containerization can offer. This represents a technological challenge in that we currently can only

run on a local machine with zero cloud support, by using technologies such as Docker or Kubernetes we can realize our product in the cloud, which will allow it to be more accessible and easier to use overall.

III. Technology Analysis

With the major requirements and technological challenges for this project now outlined, we move to an analysis of the technologies to be implemented. A concise explanation of each technology and the reasoning behind its selection, as well as possible alternatives, will be provided for each challenge outlined.

Pipeline Software

The central requirement of this project is creating a workflow pipeline which can be customized by the user, along with the ability to offer pre-created processing 'recipes' and for the user to be able to create these recipes themselves. Crucial requirements include the ability to both import and export processing recipes, to be able to add new Directed Acyclic Graphs as necessary. Furthermore, it should ideally have a built-in drag-and-drop functionality for both of the above capabilities.

Apache Airflow

During our research, we identified Apache Airflow as the best candidate for solving this pipeline software challenge. It could be the best choice because the client also indicated that it is one of their preferred solutions to such a challenge. Besides, there has been widespread adoption of Airflow by other similar data scientists in the industry.

In particular, this technology can support any number of Directed Acyclic Graphs, as well as support for adding new Directed Acyclic Graphs if needed. Users can view their pipelines as either graphs or as code. Additionally, users will be able to either select specific DAG's to run individually or run the entire pipeline at once.

Potential Alternatives

There are several viable alternatives to Apache Airflow which we have considered. For instance, Amazon offers the most robust second choice, with several options which may be useful for this project, satisfying other technological challenges outlined in the previous section. However, each of these alternatives runs in the cloud on the AWS platform, which may incur unnecessary costs and complications.

Jenkins Pipeline is another alternative that we have discarded because it is an add-on to an existing data-processing platform, instead of existing as a dedicated pipeline in its own right.

Graphical User Interface

A major requirement of this project is to have an easy to use GUI for users to easily inspect their pipelines, as well as to interact with the system as a whole. Apache Airflow offers this functionality out of the box, which would be convenient for addressing this challenge. Additionally, the client has requested that the GUI be integrated with the existing Leaflet map system to allow users to add their finished product into a visual cartographic display.

Drag-and-Drop Functionality

Our client has strongly indicated that they would prefer having a drag-and-drop functionality as part of this project, specifically in regard to both building pipelines and constructing DAGs. Our research has identified several options for implementing both of these requirements.

Drag-and-Drop: Pipeline Construction

Elyria is an open-source application designed to work with Apache Airflow that allows users to visually construct pipelines with drag-and-drop functionality. The client has indicated that this would be a useful tool for implementing the visual drag and drop feature for pipeline construction. Our research has concluded that this would be the optimal solution to the requirement, as well as being one that could be relatively easily implemented.

Drag-and-Drop: DAG Construction

Chartis is a plug-in for Apache Airflow that allows Airflow to implement Common Workflow Language (CWL). Combined with the stand-alone Rabix application, visual DAG construction can be implemented in Apache Airflow. While technically feasible, implementation of these features may prove time and labor intensive, and, as such, may be considered a stretch goal or unobtainable with current time constraints.

GDAL and Leaflet

Using the above technological solution for pipeline implementation, our finished product will be capable of outputting a single file which may be optimized for cloud storage, retrieval, and display. To enable this we would also need to integrate the GDAL (Geospatial Data Abstraction Library) to export the cloud-optimized file format (called Cloud-Optimized GeoTIFF or COG) and also the planetary enabled map viewer called CartoCosmo (based on the Leaflet map viewer). At this time, we cannot assess the feasibility of directly integrating GDAL

and the CartoCosmo map display into the program pipeline thoroughly examining the existing client application.

Documentation

For the documentation, version control, and issue tracking of this project, our team will use both GitHub for project code and Google Drive for project documents. GitHub is a simple, powerful, and versatile tool that allows our team to share our project with the client while it is in-progress.

Issue Tracking

GitHub offers a comprehensive issue-tracking solution ideal for this project. The client will have access to this feature as well, allowing them to stay informed of progress.

Documents

Our team will be implementing Google Drive to store all deliverable documents.

Application Containerization

Our client has specified in the project requirements that they would ideally like the entire project to be containerized so that it may be easily deployed to cloud platforms in the future. Additionally, our research has identified that in order to run all services required for this project, containers may prove necessary rather than ideal. We have identified Docker and Kubernetes as two technologies that can be used to address this challenge.

Docker

Docker is a free, partially open-source product that allows users to containerize applications for rapid and consistent deployment. Additionally, Docker enables multiple services to run concurrently without interfering with each other. This may prove optimal for implementation of Apache Airflow, Kalysiris, and ISIS3 running on the same machine, improving the quality of both the user experience and lowering resource costs.

Docker Networks

To coordinate multiple concurrent running containers, Docker implements a feature called Docker Networks. This allows direct routes to be established between multiple containers, allowing the containers to interact with each other without the need for inbound or outbound traffic. Depending on the final scope of our project, using this technology may become unfeasible.

Kubernetes

Depending on the scale, complexity, and quantity of Docker containers, a Kubernetes cluster may be implemented to more easily allow deployment and maintenance.

IV. Technological Integration

This section describes the macro solution for this project, as defined by the logical combination of the solutions given for each of the Technological Challenges listed above. As such, there will be areas where the possible integration strategies differ due to the stretch goals. These splits will be noted prior to the explanation of the integration of that subsystem into the system as a whole. The default assumption will be that all stretch goals are met, and the best integration strategy is in place for the final product.

ISIS3 and Kalasiris

U.S. Geological Survey Researchers already use Kalasiris to wrap ISIS3 routines using Python calls to more easily run the various individual applications. If features that would facilitate a better integration of Kalasiris are required, we will create a fork of the current Kalasiris build and add features as needed. In short, the system already used by the U.S. Geological Survey will be the same we build upon to streamline the process further.

Airflow and Elyra

Elyra will be used to satisfy the drag and drop capabilities that are not present within Apache Airflow itself. Elyra's integration is relatively straightforward. U.S. Geological Survey Researchers will create the required processing pipelines within Elyra and then export the auto-generated DAGs into Apache Airflow for the running of the pipeline. As we are currently looking at running all of this in a local host environment, a lot of thought is going into how Apache will be integrated. At present, the best strategy we have found is to run Apache in one docker container and Kalasiris in another. This will allow for the visualization of the pipeline running provided by Apache Airflow to be seen at all times and for Kalasiris and ISIS3 to run in parallel.

The next best integration strategy involves the loss of the visual component, which would entail not integrating Docker. After the DAGs have been called, to ensure the file structure is correct for Kalasiris, Apache would have to be terminated, and ISIS3 would run through the generated DAGs without visualization. Finally,The last two stretch goals will be handled within Airflow itself. As Airflow has the ability to output the metadata files and visual outputs to the web service hosted by U.S. Geological Survey.

Codebase

Lastly, we will be using three GitHub repositories. The ISIS3 repository, the Kalasiris repository, and the team Ares repository. The first two hold all processes and the wrapping capabilities, as stated earlier. The latter will keep all code written by us for use in the project, any required documentation written to inform on the use of the system, and the built-in task manager to assign and mark completed tasks as development progresses.

V. Conclusion

The goal of this project is to wrap a suite of exo-geology and exo-geography modules known as ISIS3 in an intuitive system of module creation and execution. The most important aspect of this project is to make modular execution of Isis intuitive and effective.

This project requires a vast amount of integration between systems which comes with several challenges. The types of challenges include piecing all of the requirements together in the most expedient way possible while still allowing for intuitive use by a user that is resistant to misuse and errors. There are a variety of technologies that will make this possible including Apache Airflow and Docker. The majority of this project involves combining various technologies in a way that will help those with less technical aptitude use the software with ease and little training.

In overview, we plan to accomplish our goal of creating a program that is able to generate pipelines for Isis modules that are able to be saved through the use of a few key pieces of software. Kalasiris is a wrapper of Isis in python that allows for module calls, which will be very useful to tie in Apache Airflow, which allows for the creation of specific pipelines. These main two components will work for most of the functionality we will need, but we will also be using a few more Python libraries to allow for easier use by the user.

We have quite a good handle on the software that we will be using, and our ability to complete the project in the time allotted. The technologies we will be using are well documented and we will be continuing that trend, leaving our project well documented and modular enough to be able to extend the functionality of the product we end up with, as well as thorough enough to address any issues that may appear on the user’s end.

References & Additional Info

Project RFP
https://www.ceias.nau.edu/cs/CS_Capstone/Projects/F22/Hare_T-AirFlow-ISIS-FY23_v04.pdf

Team Website
https://www.ceias.nau.edu/capstone/projects/CS/2022/TeamAres_F22/

Team GitHub
https://github.com/Team-ARES-Airflow-Processing-Pipeline/

Jenkins Pipeline
https://www.jenkins.io/doc/bookapipeline/

Elyra
https://github.com/elyra-ai/examples/tree/main/pipelines/run-generic-pipelines-on-apache-airflow

Rabix Composer
https://github.com/rabix/composer

Chartis Plug-In
https://github.com/trejas/chartis

Rabix + Chartis Visual DAG Editor Demonstration
https://airflowsummit.org/sessions/demo-visual-dag-editor/