CLOUD PROCESSING OF PDS ARCHIVAL PRODUCTS WITH AMAZON WEB SERVICES, KUBERNETES, AND ELASTICSEARCH. Kevin M. Grimes II, Rishi Verma, James Michael McAuley, Tariq Soliman, Anil Natha, Zachary M. Taylor, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA (kevin.m.grimes@jpl.nasa.gov).

Introduction: As cameras and other spacecraft instruments improve, the data they produce becomes richer in quality and, inevitably, larger in volume. Processing these data in a timely fashion via traditional means becomes sluggish, inefficient, and expensive. In order to address challenges posed by next-generation missions taking place in our solar system, a reconsideration of processes used to process these data is required.

The Planetary Data System (PDS) Cartography and Imaging Sciences Node (IMG) retains hundreds of terabytes of data, collected from dozens of missions and spacecraft over as many years. Among the responsibilities of IMG is to make the data not only accessible by the public, but also searchable. By leveraging PDS Engineering Node’s (ENG) [1] software, Amazon’s [2] cloud offering AWS [3], and the open-source container orchestration platform Kubernetes [4], IMG has made strides to provide a rich search experience of its data for the community.

Architecture: IMG follows the microservices pattern for its backend architecture, which enables “rapid, frequent and reliable delivery of large, complex applications” [5] in the form of small, individual services. These services may be developed independently from one another. In IMG’s case, they do not run on traditional servers, but instead on AWS’s EKS [6] service, which provides Kubernetes clusters using Docker [7] as a service. Communication between these services is achieved via APIs, and common data is stored in various databases, including Elasticsearch [8] and DynamoDB [9].

Data Access. Previously, IMG has allowed users to download its data holdings via HTTPS directly from our servers. However, IMG does not intend to continue holding all of its data on-premises; instead, it is exploring gradually moving its holdings to Amazon’s Simple Storage Service (S3). Of course, with movement to the cloud comes a variety of concerns, including cost. A requirement of the new system is that IMG be able to regulate users who abuse our system and, consequently, rack up a large bill, have their download rates be curtailed. This is enabled by the Data Access API, a lightweight application interface inspired by TEA [12]. The Data Access Application Programming Interface (API) abstracts away the physical location of files, allowing them to be stored in various different S3 buckets and actual servers. Additionally, access to these files is controlled by tokens, which the IMG API manages. Finally, users who abuse the system are curtailed, and risk having their tokens revoked altogether.

IMG API. Central to the architecture is the IMG API, a lightweight application interface responsible for interpreting general user requests, authenticating the user who made them, determining the user’s access via role authorization, and routing the request to the target backend service. By forcing all requests to the system (either from tools internal to IMG, or from outside entities) to go through this central location, we can easily revoke access to entities who abuse our services. For this component, IMG uses API Gateway [10], another AWS offering that interprets OpenAPI 3.0 [11] specifications and allows routing to other services IMG maintains in the cloud. Additionally, API Gateway offers token-based authentication, which IMG uses.
Containers can operate in their own contexts without worrying about the larger context. In IMG’s case, this involves running our development environment on hardware at JPL, but our production environment out of AWS.

Scaling and upgrading our services with no downtime is another requirement of the system. Kubernetes enables this with the help of a few additional technologies: Flagger [18], FluxCD [19], and Istio [20]. With the help of these tools, “canary” rollouts [21] are enabled which slowly redirect traffic to new, upgraded versions of services. If the system detects that a significant number of requests are failing, it cancels the rollout and automatically redirects traffic to the previous instance. If there do not appear to be any problems with the upgraded instance, however, all traffic is redirected to the new instance and the old instance is removed.

Future work: Due to the rapidly changing nature of the technologies described above, IMG is constantly learning new and improved design patterns and technologies. Additionally, IMG hopes to interface with PDS Engineering Node’s PDS API [22]—a centralized API that routes traffic to different nodes.

References:
https://www.elastic.co/enterprise-search. [9]
https://swagger.io/specification/
https://pds-engineering.jpl.nasa.gov/ development/pds4/ 5.0.0/ingest/harvest/
https://fluxcd.io. [20] [21]