Cloud Surprises in Moving NASA EOSDIS Applications into Amazon Web Services

Brett McLaughlin, NASA EED2 Program

The material is based upon work supported by the National Aeronautics and Space Administration under Contract Number NNG15HZ39C
1. NASA is not a Heroku or PaaS world

- *We thought* NGAP would primarily be “*Heroku for Earth Science*”
  - Hosting for web applications
  - Limited application profiles
  - Ease of Use
“The needs of the many…”

• We were asked for a lot of different things
  – Hosting for web applications all the things
  – Limited Broad application profiles
  – Ease of Use (??)

• As the de facto cloud platform, NGAP had to evolve to meet these broader needs

* https://pixabay.com/p-1458869
NGAP as a PaaS

- ECC (Code testing, tracking, deployment)
  - App Source Code
  - NGAP Base AMI (Secure)
  - NGAP Builder (Creates “slug” from ECC-hosted codebases)
  - NGAP-compliant AMI (Application)
  - NGAP-compliant AMI (Application)
  - NGAP Services (Monitoring, Logging, Security, Autoscaling, Billing, etc.)
  - OCIO GP-MCE (AWS Reseller)

- ESDIS “blessed” component
AWS and WOS: Benchmarking at the Edges

1. Only Data Cannon will pull from NSG and will push the data to Ingest node(s).

2. All Ingest nodes will write to local storage and update Database for serving.

3. ASF decides where to serve the user, depending upon benchmarking, user's network, etc.

*DC refers to a Datacenter in lower 48 that is not yet defined.*
The evolution from PaaS to (more) IaaS

• **NGAP 0.1**: 100% PaaS
• **NGAP 1.0**: 80% PaaS
• **NGAP 1.1**: 60% PaaS
• **NGAP Sandbox**: <50% PaaS
2. Managed Services are the shizzle

• *We thought* NGAP would primarily be a “hosting platform”
  – Reduce hardware buys
  – Provide operational support for apps

• AWS *does* lots of cool stuff
  – Supplies resources (instances, networks, etc.)
  – Monitors and keeps those resources running
Prefer Services Over Custom Code

- But AWS also *provides* some cool stuff™
  - Lambda
  - Step Functions
  - AWS Batch
  - API Gateway
  - Something(s) since I wrote this presentation
Ingest/Processing Components without Step Functions

- DynamoDB
- State Storage
- Dispatcher
- Event Data Storage
- Scheduler
- Discover
  - Sync
  - Step 1
  - Step 2
  - Step 3
- Error Behaviors
- S3
- Granules Table
- CMR
- SIPS
- Dispatch Triggers
- Execution Engine
- Collection-Specific Behavior
- External Data Sources/Stores

Ingest/Processing Components using Step Functions

- Scheduler
- Step Functions
- Discover
  - Sync
  - Step 1
  - Step 2
  - Step 3
- S3
- Granules Table
- CMR
- SIPS
3. EOSDIS Applications are as Sophisticated as AWS allows

- We thought that most of the applications we’d support would be “web applications”
  – Think Rails + database + S3
  – Think buildpacks
  – Think well-constrained technical problems
Managed Services Drive Innovation

• Instances to Containers
  – Greater segregation of functionality
  – Movement toward services over monoliths

• Software on an Instance to AWS Service
  – ElasticSearch to AWS ElasticSearch
  – RabbitMQ to AWS SQS
  – Etc.

• And bigger changes… (more on that later)
GIBS to GIBS in the Cloud

Diagram showing the process flow from GIBS to GIBS in the Cloud, including stages such as Install, Inventory, CM, Subscription Service, SigEvent Server, and OAS Console.
4. NGAP is not as easy as AWS to operate

- *We thought* that NGAP would basically offer the ease of operation that AWS offers to a typical application
  - Low-effort monitoring
  - Low-effort logging
  - Low-latency response times from operations
5. AWS uses an open-ended spending model

- We thought we’d just turn on Amazon’s billing controls and be A-OK.
  - Set spending limits
  - Produce granular billing reports
  - Limit egress at predetermined thresholds
Amazon provides the information and empowers the user

- Amazon wants to inform but not limit
  - AWS is happy to email you
  - AWS is happy to let you know what you’re spending
  - AWS (reasonably) cannot force action, because “the action” is not standard
Egress (in particular) is a big deal

• When data leaves your application, service, data store, etc. …
  – …and goes to another region
  – …and goes outside of AWS

• Egress is expensive
  – Rack Rates: $0.08/GB after first 150TB
  – In other words, a significant portion of total monthly cloud-associated costs
Cost isn’t even the biggest issue

• A huge bill is bad…
  …but jail is worse.

• The Anti-Deficiency Act (ADA) disallows unbounded costs

• We need a means of absolutely limiting egress costs

6. Favor Re-architecture over “just getting into the cloud”

• *We thought* that many applications would simply move their architecture to NGAP and (more or less) call it a day
Case Study: ASF

Developers & SAs

- Configure system
- Setup system tasks
- Deploy “application”

VM

- OS pkg
- OS pkg
- OS pkg
- cron
- code
- code
“Direct” Forklift onto NGAP

NGAP Orchestration

Cloud Formation

Bamboo

Configure system

manages

Setup system tasks

Deploy “application”

NGAP VM

OS pkg

OS pkg

OS pkg

cron

code

code

NASA
But… it turns out…

“We wish we’d re-architected.” – ASF

Why?

• Managed Services
• Natural Inflection Point
• Opportunity for Innovation
ASF, Rearchitected
Ingest, Rearchitected

* Created with https://cloudcraft.co/
One of the great beauties of architecture is that each time, it is like life starting over again.

-Renzo Piano
Cloud Surprises in Moving NASA EOSDIS Applications into Amazon Web Services
This material is based upon work supported by the National Aeronautics and Space Administration under Contract Number NNG15HZ39C.

Raytheon