What does DigitalGlobe do?

• Fly a constellation of satellites that take images of the earth.
• Collect about 3,500,000 square kilometers of imagery everyday
  - Slightly more than the landmass of India
  - 21,875 times the landmass of Lichtenstein
• Downlink ~5-6 TB of new “raw” imagery per day (~2 PB year)
• Create products from that imagery with various forms of image processing and analytics
• Produce between about 40 TB - 100 TB of new products per day
• Manage a platform in the cloud where you can run algorithms against our imagery (Take the processing to the imagery)
DigitalGlobe sets the standard for quality

Accuracy. Currency. Completeness. Consistency. We lead the industry.
24 Hours of Collections
7 Days of Collections
1 Month of Collections
6 Months of Collections
1 Year of Collections
DigitalGlobe Needed A New Architecture

• ‘Old Way’ at DigitalGlobe
  - ~3-6 months to get a new applications with new VMs deployed. All ‘snowflakes’, all F5 configurations were snowflakes, etc. Some puppet used in production.
  - Lead us to do ‘expedient’ (unnatural) things.
    - Bolt on inconsistent functionality onto existing software to avoid a new deployment.
  - Integration through a centralized Database.
    - Fragile, slow to change, hard to use our data in new and different ways.
  - Very project focused.
    - Twist the monolith to meet the needs of each new project
DigitalGlobe Needed A New Architecture

• New Opportunity – WorldView-4
  - Tried to introduce change and structure into old environment (Several attempts didn’t work)
  - Upper management gave us the green light to change.
  - Tight schedule. Need to move at ‘rapid’ pace.
  - Designed and put in a microservice based architecture with flexible/extensible workflows.
  - Scheduled satellite launch Q3 2016
Technical Decision Process

• Knew that a PaaS was key.
  - Didn’t want to ‘just do IaaS’ (a technology 8 years old at the time) as our “huge technology leap”
• Surveyed the available PaaS alternatives
• Created “knockout” criteria
• Down-Select – CF chosen as leading candidate
• Performed tasks to verify knock out criteria met
  - e.g. upgraded PCF release while measuring uptime of platform services
• Migrated DG sample Java app, Ruby app and Python app, and we verified portability across multiple cloud providers
• Built pricing and staffing models
Path to Adoption

1. Prototypes
2. Small Team – Pioneers
3. Learn/fix/adapt
4. Expand to larger team
5. Learn/fix/adapt
6. Expand again
7. Learn/fix/adapt
8. Expand again
Learn/Fix/Adapt Cycles

• Microservices Sprawl Fast
  - Had to quickly implement Eureka to keep track of them
• Configuration Centralization
  - Using Spring Cloud Config Server updated to have a Postgres backend
• API Management is hard
  - Using SoftwareAG’s Centrasite to track services
    - Integrated with our build pipeline
  - Using APIARY to design APIs.
  - Still learning
• Decoupling Code Deploy from Feature Deploy
  - Still learning.
Learn/Fix/Adapt Cycles

• Standards, Standards, Standards are your friend
  - Standard endpoints on every service (/status,/healthcheck)
  - Authentication Standards, Authorization Standards
  - Logging Standards & Centralized Logging
  - RESTful standards....etc. etc.
  - Service urls: service://service-name/api-version/environment/<rest of url>

• Build Pipeline – Right tool for each job
  - Jenkins for building, XL Release for deployments

• Vocabulary
  - Avoid the words ‘Common Services’. Everything should be a common-use service.
Cloud Foundry @ DigitalGlobe

• Current Dev/Test State
  - Open Source Cloud Foundry Running on OpenStack
    - Kudos to ECS Team. Great work helping us here.
  - Custom deployment – best-effort at High-Availability config:
    - 54 VMs in the foundation. 25 are DEAs.
    - Need some tweaking to fully utilize 2 AZ’s correctly.
  - This environment supports: Development, Test
  - Total of >800 services running in the current environment
    - Not all unique. Developers may have to deploy a service (or mock of the service) to their space to do development of test with a common service.
    - ~530 instances are in developers spaces
    - ~270 instances are in various stages of test
    - Interesting fact: All of this is in support of about ~60-80 ‘real’ microservices.
  - DEA VMs are 2 CPU, 16 GB RAM with Cloud Foundry configured for a 3x overcommit on memory.
  - All logs being shipped into an ELK stack via log-drains bound to apps. Looking at firehose integration.
Cloud Foundry @ DigitalGlobe

• Current Production State
  - PCF Running on OpenStack
  - Installed, up, and running
  - A few services have been snow-flakish deployed and tested
  - Retooling our pipeline. True automated prod deployments are happening for this first time right about now.
Cloud Foundry Wins

- **Development Speed**
  - Once we had ‘patterns’ down, it is now easy to develop a new microservice
  - Template-based bootstrap of new team members and services

- **Ease of Deployment**
  - Self-service Jenkins portal to create a new service
  - Merge to master in GitHub to initiate the CI/CD pipeline (not finalized)
  - Managed deployment to several stops in the pipeline (not finalized)

- **Visibility**
  - A few cf curl commands reveals all that is deployed

- **Auditability**
  - When new services are found, Nimsoft alerts are sent
  - Monitoring automatically creates a new dashboard in Kibana for the discovered service (still a work in progress)

- **Control**
  - Now that we have Visibility, and Auditability, we have control over our services.
Cloud Foundry Wins

• Testing
  - The test groups are ecstatic. Generally very easy to test micro services. Test coverage and ability to isolate and troubleshoot vastly superior to legacy.

• Resiliency
  - OpenStack compute node failures
  - Bosh realized its was missing bits and rebuilt the environment
    - *No users knew*
      - We lost a few VMs outside of CF that affected people, but nobody knew the CF environment was even affected.
  - Has happened several times, with the same results.
  - We’ve captured stats on these failures and they have been highly useful in discussion with Program and Executive management
Cloud Foundry Challenges

• Synchronization across Foundations
  - E.g. UAA, Client IDs, Secrets, Scopes.
  - Best practices around load-balancing across foundations.
  - Best practices around naming of domains across foundations...
    - Still want each foundation individually addressable for some functions.
      - E.g. we want to be able to target a foundation to do a deploy.
      - We want applications load-balanced across foundations though.

• SSL & Domains
  - Since CF only supports a single cert, we have to use a ton of SANs (subject alternative names).
  - Impossible to manage with 100+ developers/testers.
  - HTTPS required in our environment
  - Please fix. We should be able to get a cert for a developers space and just add it.
Cloud Foundry Challenges

• Open Source
  - No out-of-the-box support for an HA environment.
  - Manually keeping up to date. Editing deployments for new features, etc.
  - Fall behind quickly
  - Lack of Graphical Management tools, even just for visibility

• Developer/DevOps access to spaces
  - We have a lot of spaces, and some ephemeral spaces.
  - It is tough to add all the developers as space auditors so they can see the logs from all spaces.
  - Sometimes this is needed with certain types of failures that can't be diagnosed from the ELK stack logs.
  - Want to limit the ability to create a snowflake.

• Tension between microservices and licensing models (All Vendors)
  - The more you do the right thing, the more it costs.
Cultural Challenges

• Change is hard
  - The old ways come back....You have to be vigilant
• You need someone in Executive Management that backs you
  - Not everyone is going to be on-board
• As an Architecture Team – you need to be willing to:
  - Put it on the line (reputation / job)
  - Abandon the ivory tower – Dig in, implement, provide guidance, “Keep pumping gas into the plane”
  - We’ve done all of this (and to a significant degree asked our EA colleagues to “keep the lights on” in the rest of the enterprise)
Future needs

• A small footprint, OEM type installation of CF
• Manage multiple foundations as a single environment
  - E.g. Cross Region type of deployments and management.
• Persistent storage access — NFS mounts available in a container specifically. Woo-hoo! Delivered in Diego now!
CF Org Structure

CF ORG

Infrastructure
- Config Service
- Event Service

Business Domain
- Inventory
- MCS
- MPS

Eureka
ActiveMQ
BMPS
PostgreSQL
We are hiring!

Contact Information

• Mike Waters
  Enterprise Software Architect
  mwaters@digitalglobe.com

• Mike Wierzbinski
  Enterprise Cloud Architect
  wierz@digitalglobe.com

• Steve Wall
  Senior Contractor - ECS
  swall@ecsteam.com