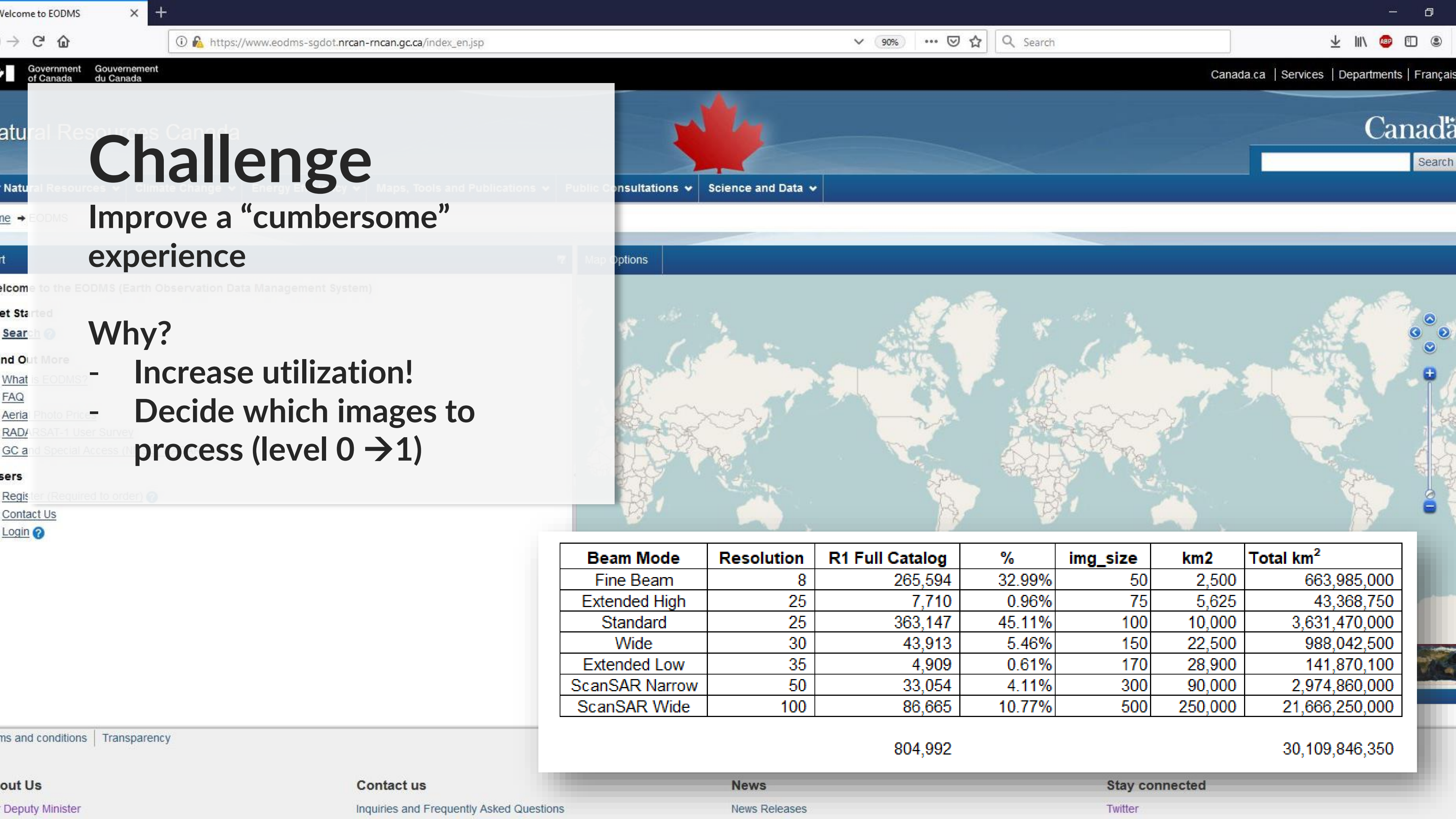
RADARSAT-1 Liberators

Nick Kellett, Adam Simonini, Alan Higginson, Kevin Jones

CSA Challenge #6

https://www.radarsat-1-archive.space

https://github.com/adamsimonini/rs1



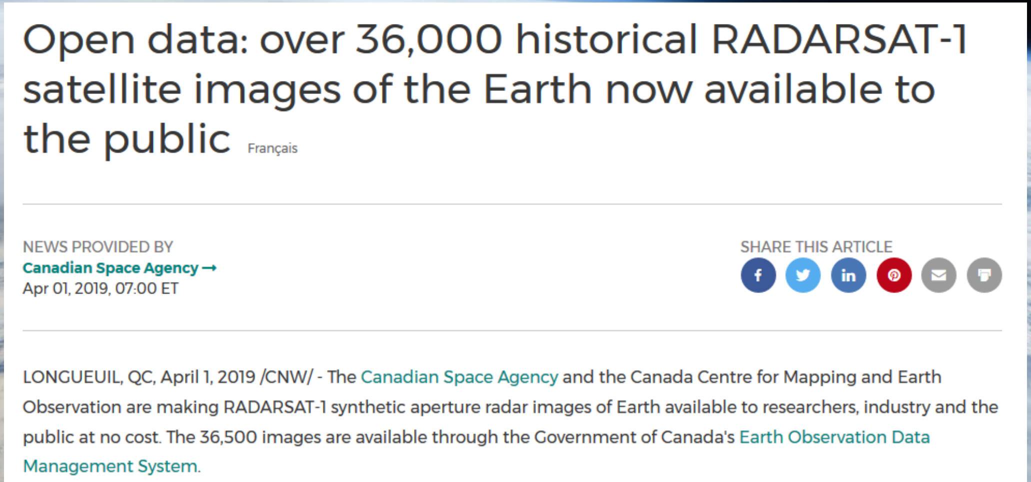
Challenge

Improve a “cumbersome” experience

Why?

- Increase utilization!
- Decide which images to process (level 0 →1)

Beam Mode	Resolution	R1 Full Catalog	%	img_size	km2	Total km^2
Fine Beam	8	265,594	32.99%	50	2,500	663,985,000
Extended High	25	7,710	0.96%	75	5,625	43,368,750
Standard	25	363,147	45.11%	100	10,000	3,631,470,000
Wide	30	43,913	5.46%	150	22,500	988,042,500
Extended Low	35	4,909	0.61%	170	28,900	141,870,100
ScanSAR Narrow	50	33,054	4.11%	300	90,000	2,974,860,000
ScanSAR Wide	100	86,665	10.77%	500	250,000	21,666,250,000
		804,992				30,109,846,350



Open data: over 36,000 historical RADARSAT-1 satellite images of the Earth now available to the public

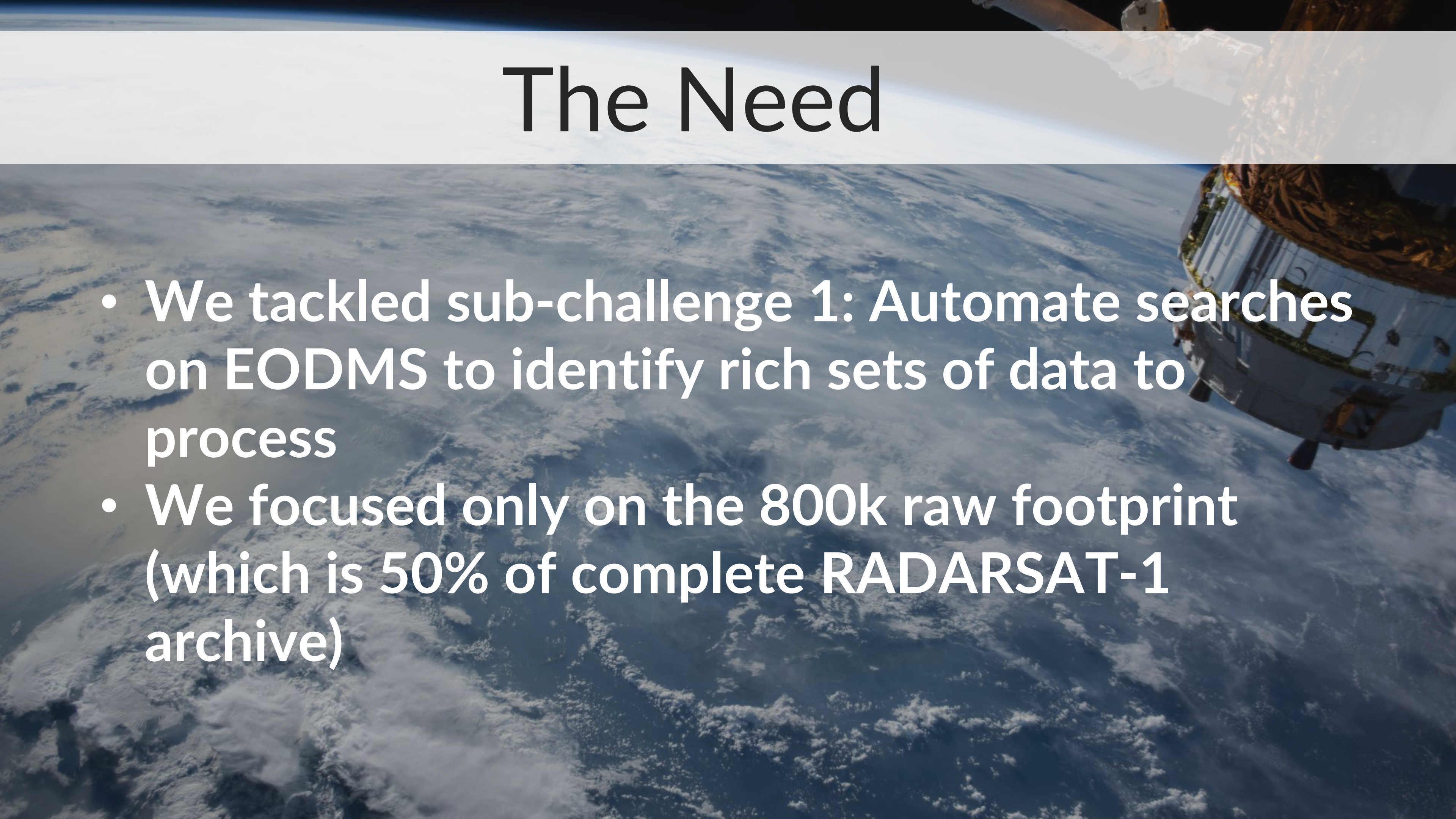
Français

NEWS PROVIDED BY
Canadian Space Agency →
Apr 01, 2019, 07:00 ET

SHARE THIS ARTICLE

LONGUEUIL, QC, April 1, 2019 /CNW/ - The Canadian Space Agency and the Canada Centre for Mapping and Earth Observation are making RADARSAT-1 synthetic aperture radar images of Earth available to researchers, industry and the public at no cost. The 36,500 images are available through the Government of Canada's Earth Observation Data Management System.

- April 1st, 2019: CSA opened >36k RADARSAT-1 satellite images to public
- Downloadable through NRCan's Earth Observation Data Management System(EODMS)
- Only processed images can be used for analysis. RADARSAT-1 acquired a significant amount of data; 36k images corresponds to a very small portion of the entire archive.
- To avoid processing costs, help identify and prioritize meaningful data for processing and public release.

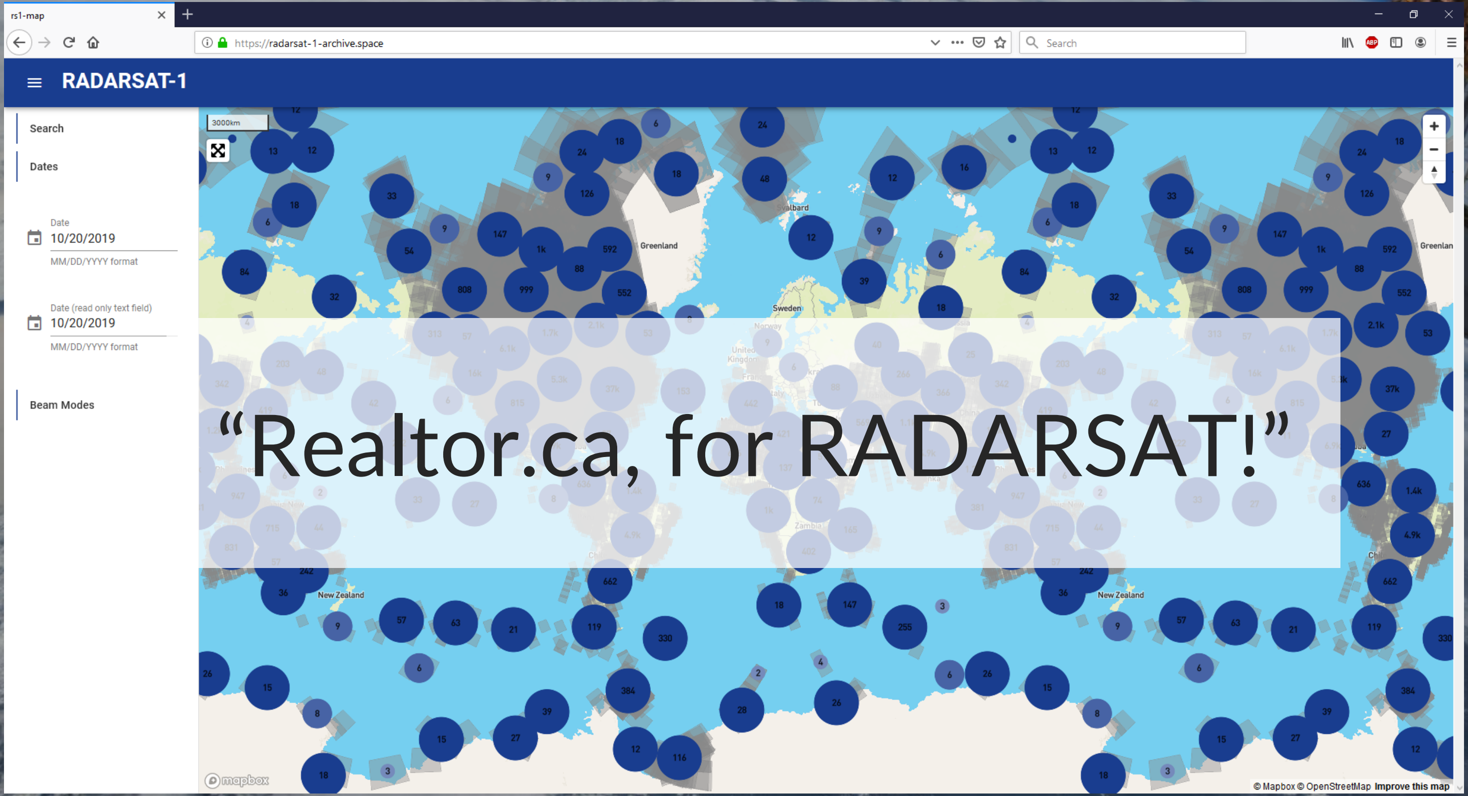


The Need

- We tackled sub-challenge 1: Automate searches on EODMS to identify rich sets of data to process
- We focused only on the 800k raw footprint (which is 50% of complete RADARSAT-1 archive)

Multi-Temporal Analysis

The Solution



rs1-map

https://radarsat-1-archive.space

RADARSAT-1

Search

Dates

Date
10/20/2019
MM/DD/YYYY format

Date (read only text field)
10/20/2019
MM/DD/YYYY format

Beam Modes

“Realtor.ca, for RADARSAT!”

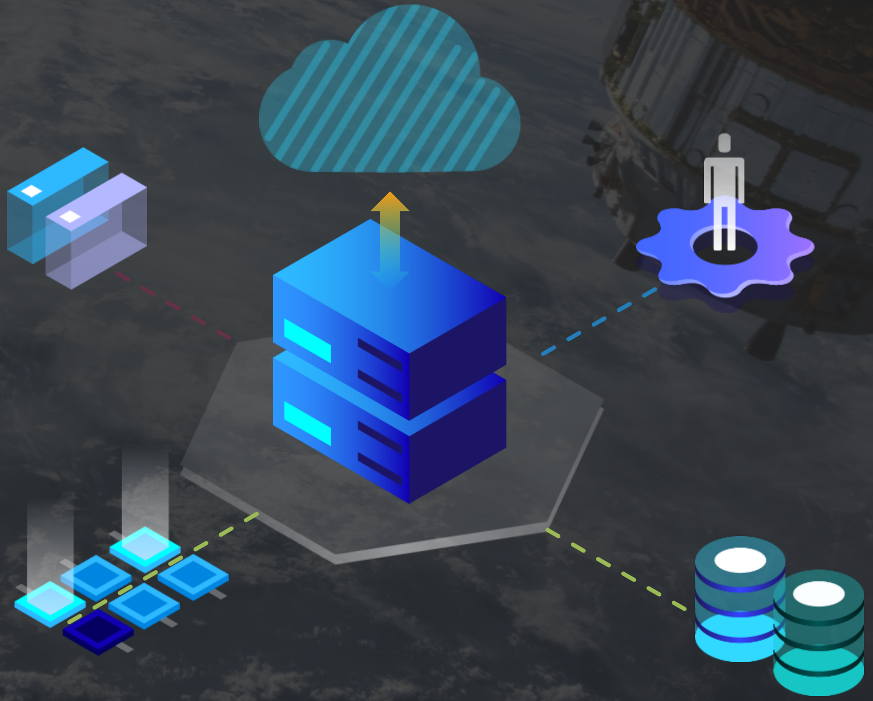
https://www.radarsat-1-archive.space

LIVE DEMO

https://www.radarsat-1-archive.space

What we built

- Established build architecture
- Selected system components
- Data Extraction
- Build/test/deploy (54 code commits!)

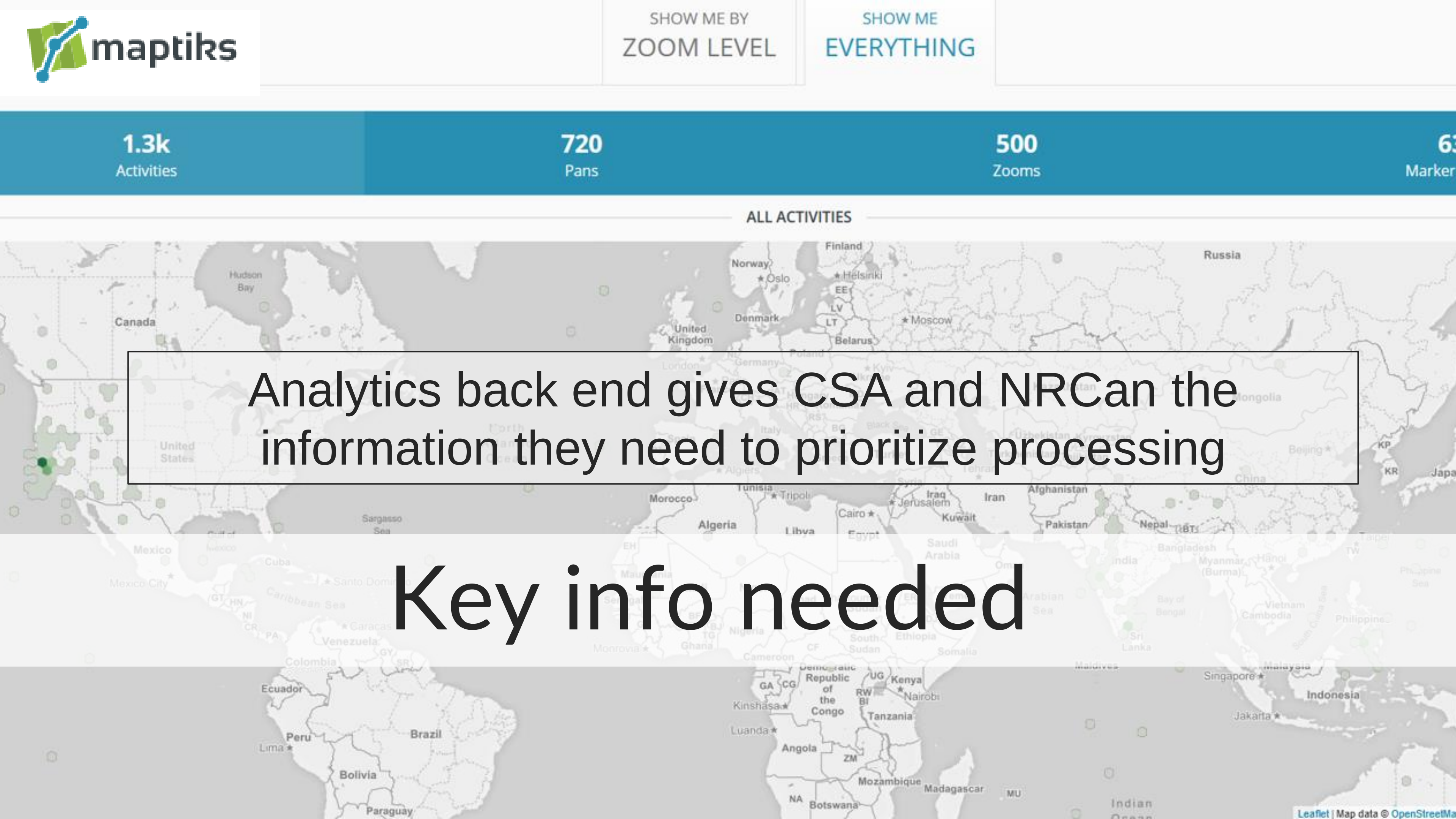


What didn’t work

- We couldn’t connect to EODMS due to CORS (security) – manual downloads and conversions of files = painful!!!
- Using Elastic Search database as the data store – we didn’t have time to configure it for Geojson format so went with PostgreSQL db instead
- We tried a Katalon script to automate extract the data but it didn’t work right away (would benefit from more support/documentation)

Next Version!

- Automate downloading and converting of EODMS images
- Schedule Geo Meta DB intake process to find newly extracted files
- Add more filters (beam mode only working filter currently)
- We recommend a “whole” CSV file of 800k records that the public can use
- Export button for data
- Link to available image file for download from EODMS
- Filter on raw vs processed data
- Satellite/Street view toggle
- Image count on screen
- Get image IDs and prioritized list of specific images based on user interactions



maptiks

SHOW ME BY ZOOM LEVEL

SHOW ME EVERYTHING

1.3k Activities

720 Pans

500 Zooms

63 Marker

ALL ACTIVITIES

Analytics back end gives CSA and NRCan the information they need to prioritize processing

Key info needed

RADARSAT-1 Liberators

Nick Kellett, Adam Simonini, Alan Higginson, Kevin Jones

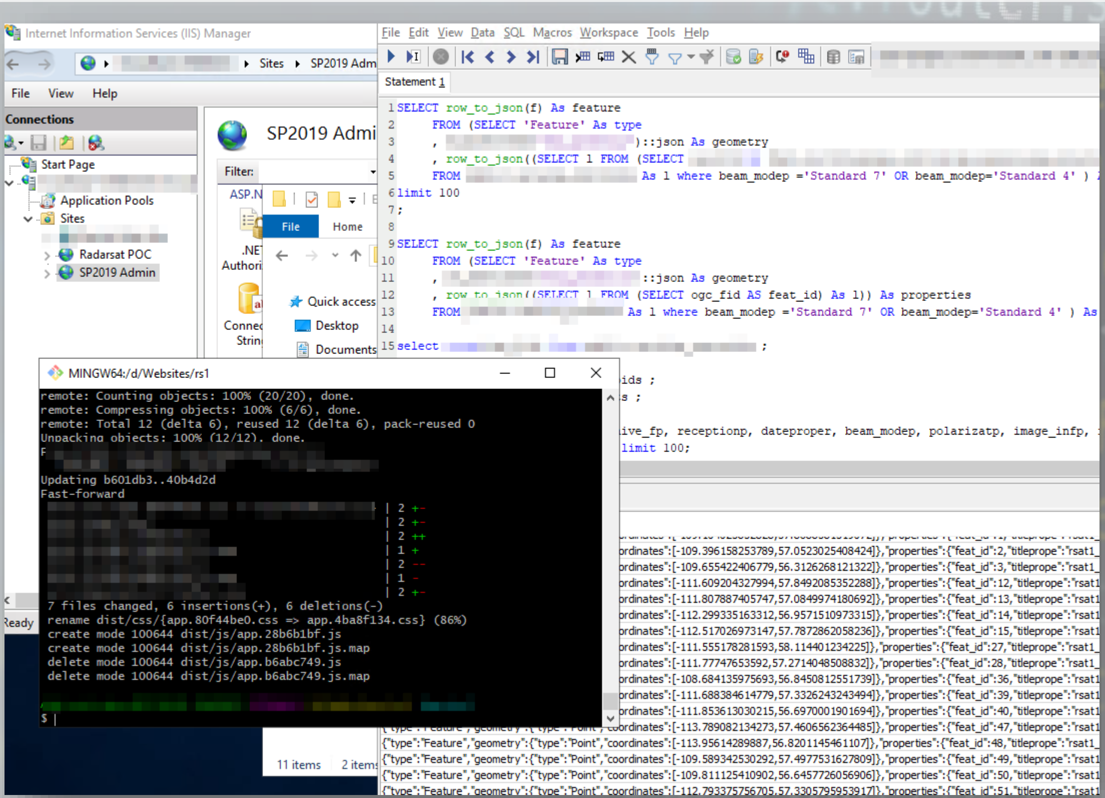
Questions?

https://www.radarsat-1-archive.space

https://github.com/adamsimonini/rs1

BACKUP SLIDES

Architecture



```sql
SELECT row_to_json(f) As feature
     FROM (SELECT 'Feature' As type
     , [redacted])::json As geometry
     , row_to_json((SELECT l FROM (SELECT [redacted]
     FROM [redacted] As l where beam_modep ='Standard 7' OR beam_modep='Standard 4' )
limit 100
;

SELECT row_to_json(f) As feature
     FROM (SELECT 'Feature' As type
     , [redacted]::json As geometry
     , row_to_json((SELECT l FROM (SELECT ogc_fid AS feat_id) As l)) As properties
     FROM [redacted] As l where beam_modep ='Standard 7' OR beam_modep='Standard 4' ) As

select [redacted] ;

ids ;
s ;

ive_fp, receptionp, dateproper, beam_modep, polarizatp, image_infp,
limit 100;
```

```
remote: Counting objects: 100% (20/20), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 12 (delta 6), reused 12 (delta 6), pack-reused 0
Unpacking objects: 100% (12/12), done.
F[redacted]

Updating b601db3..40b4d2d
Fast-forward
 [redacted]                                         | 2 +-
                                                    | 2 +-
                                                    | 2 ++
                                                    | 1 +
                                                    | 2 --
                                                    | 1 -
                                                    | 2 +-
 7 files changed, 6 insertions(+), 6 deletions(-)
 rename dist/css/{app.80f44be0.css => app.4ba8f134.css} (86%)
 create mode 100644 dist/js/app.28b6b1bf.js
 create mode 100644 dist/js/app.28b6b1bf.js.map
 delete mode 100644 dist/js/app.b6abc749.js
 delete mode 100644 dist/js/app.b6abc749.js.map

[redacted]
$ |
```

```
ordinates":[-109.396158253789,57.0523025408424]},"properties":{"feat_id":2,"titleprope":"rsat1_
ordinates":[-109.655422406779,56.3126268121322]},"properties":{"feat_id":3,"titleprope":"rsat1_
ordinates":[-111.609204327994,57.8492085352288]},"properties":{"feat_id":12,"titleprope":"rsat1
ordinates":[-111.807887405747,57.0849974180692]},"properties":{"feat_id":13,"titleprope":"rsat1
ordinates":[-112.299335163312,56.9571510973315]},"properties":{"feat_id":14,"titleprope":"rsat1
ordinates":[-112.517026973147,57.7872862058236]},"properties":{"feat_id":15,"titleprope":"rsat1
ordinates":[-111.55178281593,58.114401234225]},"properties":{"feat_id":27,"titleprope":"rsat1_
ordinates":[-111.77747653592,57.2714048508832]},"properties":{"feat_id":28,"titleprope":"rsat1_
ordinates":[-108.684135975693,56.8450812551739]},"properties":{"feat_id":36,"titleprope":"rsat1
ordinates":[-111.688384614779,57.3326243243494]},"properties":{"feat_id":39,"titleprope":"rsat1
ordinates":[-111.853613030215,56.6970001901694]},"properties":{"feat_id":40,"titleprope":"rsat1
ordinates":[-113.789082134273,57.4606562364485]},"properties":{"feat_id":47,"titleprope":"rsat1
{"type":"Feature","geometry":{"type":"Point","coordinates":[-113.95614289887,56.8201145461107]},"properties":{"feat_id":48,"titleprope":"rsat1_
{"type":"Feature","geometry":{"type":"Point","coordinates":[-109.589342530292,57.4977531627809]},"properties":{"feat_id":49,"titleprope":"rsat1
{"type":"Feature","geometry":{"type":"Point","coordinates":[-109.811125410902,56.6457726056906]},"properties":{"feat_id":50,"titleprope":"rsat1
{"type":"Feature","geometry":{"type":"Point","coordinates":[-112.793375756705,57.3305795953917]},"properties":{"feat_id":51,"titleprope":"rsat1
```

System Components

- **Front End UX**
  - MapBox, Vue.js, Beautify for UI, served by node.js backend. Talks to:
- **GEOJSON REST API**
  - Custom REST api using C# Web.API. Talks to:
- **GEOSPATIAL METADATA DATABASE**
  - PostgreSQL with PostGIS. Contains GeoJSON metadata. Loaded from:
- **GEOJSON EXTRACTS FROM EODMS**
  - Hand-extracted from HMI, converted to GEOJSON using QGIS. Loaded into GEO METADATA DB by:
- **GEO META DB ETL UTILITY**
  - Custom Powershell creating command line scripts to automate GEOJSON file extracts using Ogr2Ogr utility.

System Infrastructure

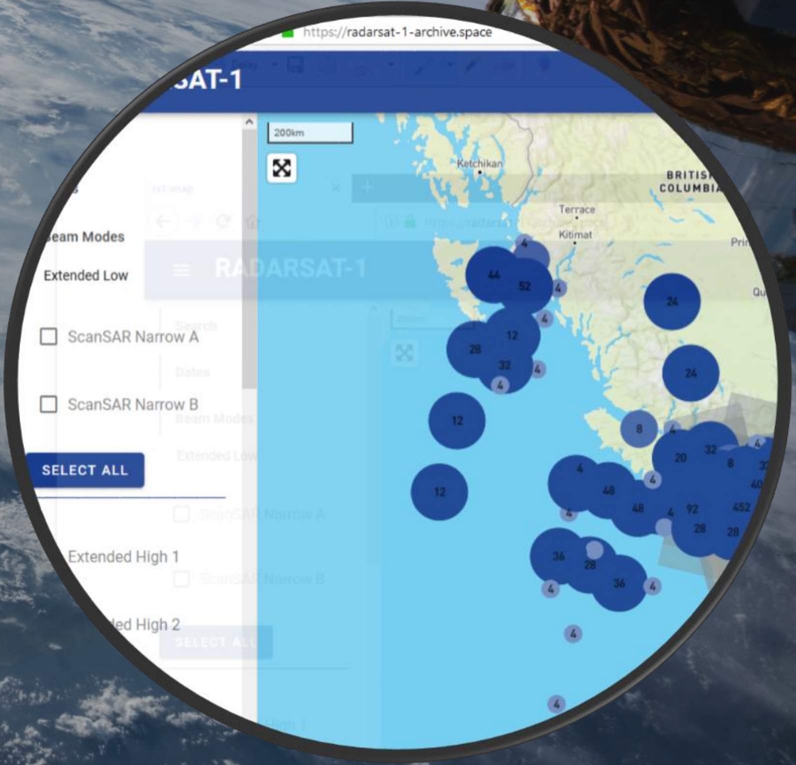
- **Web/Application Server**
  - Amazon EC2 small instance, 1 CPU, 2GB RAM
- **Database Service**
  - Local PostgreSQL db service on web server. Vnext would use hosted DB.
- **Local file store**
  - Metadata and geojson extracts don’t take up much space (images are not downloaded). Think MB not GB.

Data Extraction Process

- Manual visits to EODMS HMI to extract 5k at a time
- Export to CSV, Export to SHP
- QGIS convert SHP to GEOJSON to centroids
- Ran calculate centroid in QGIS to spit out extents
- Then import centroids and extents to PostGIS db via ogr2ogr
- REST API in front of PostGIS serves up the files to MapBox UX

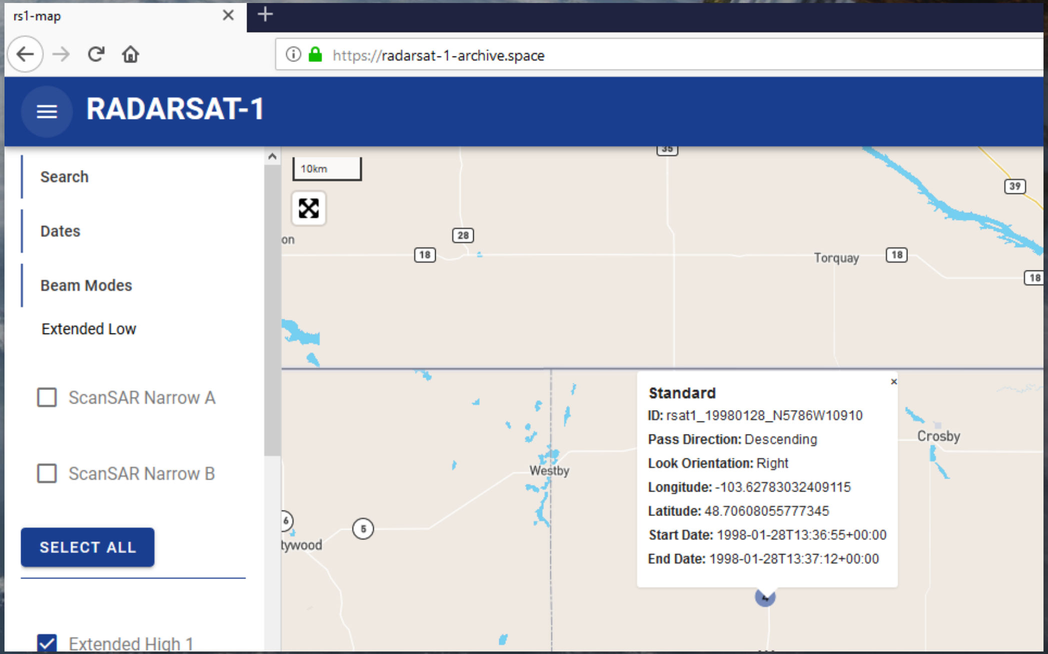
The user experience

- “UX-first” solution design approach
  - We wanted something fast and intuitive
  - Focus on user’s ability to explore data– therefore allow seeing all records
  - Records are extents and points
  - “Steps” and colours indicate quantify of records



Filters and Metadata

- Use filters to reduce the displayed records
- Zooming and panning also removes records from display
- Clicking on points shows property card pop-up
- Queries are made to our local middle-ware geo metadata database rather than EODMS = fast



Standard
ID: rsat1_19980128_N5786W10910
Pass Direction: Descending
Look Orientation: Right
Longitude: -103.62783032409115
Latitude: 48.70608055777345
Start Date: 1998-01-28T13:36:55+00:00
End Date: 1998-01-28T13:37:12+00:00