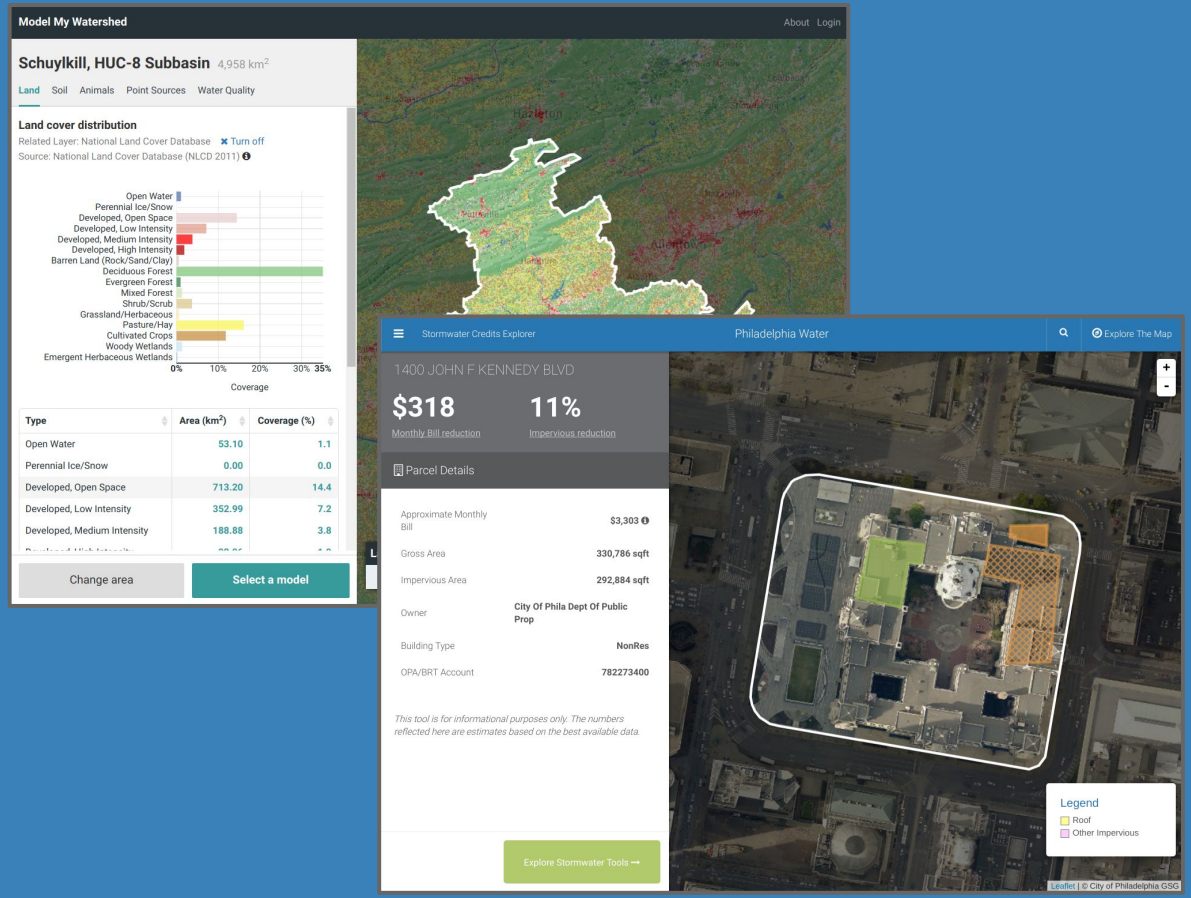
Raster processing on Serverless Architecture

Using python tools on AWS Lambda

Hello.

azavea



What is Serverless Architecture?

What is Serverless Architecture?

- Stateless function as unit of work
- Event driven
- Cost per execution
- No provisioning, limited management
- Auto-scaling, fault tolerant by default

AWS Lambda

AWS Lambda - how it works

- Write a function that handles events
- Implement workflow as usual
- Zip code and dependencies
- Upload and register/update

AWS Lambda - what you get

- A runtime (python, node, java, .NET) on linux
- Compute resources (bundled RAM/CPU)
- Event integrations
- Web UI, CLI , 3rd party tools
- Lots of limitations

AWS Lambda - execution limits

- Up to 1.5GB memory (proportional CPU)
- 512MB `/tmp` space
- 300s timeout
- 6MB limit response/request
- Cap on threads/process/file descriptors

AWS Lambda - deployment

- Code + Dependencies bundled
  - 50mb compressed
  - 250mb uncompressed
- Build dependencies which target AMZ Linux
- Deployment size impacts cold start time

AWS Lambda - costs

- Charged per request + per 100ms
  - $0.20 / 1MM requests
  - $0.00001667 / GB-s
- Perpetual free-tier (per month)
  - First million requests
  - First 400,000 GB-s

When to use

- Latency tolerant
- Stateless
- Traffic is bursty
- Relatively short-running
- Modest resource requirements

Example Use Cases

- ETL
- “Micro-services”
- Raster statistics
- Raster tiling

Python Raster Tools

rasterio & numpy (and friends)

Rasterio

- Idiomatic python API over GDAL
- Reads raster data into numpy arrays
- Raster processing utilities
- Windowed reads
- HTTP/S3 reads

Numpy

- Large, multidimensional arrays (aka, rasters)
- Not spatial
- Typically fast
- Extensive community and ecosystem

And more

- Shapely: vector operations
- Pyproj: reprojection
- Pillow: image generation

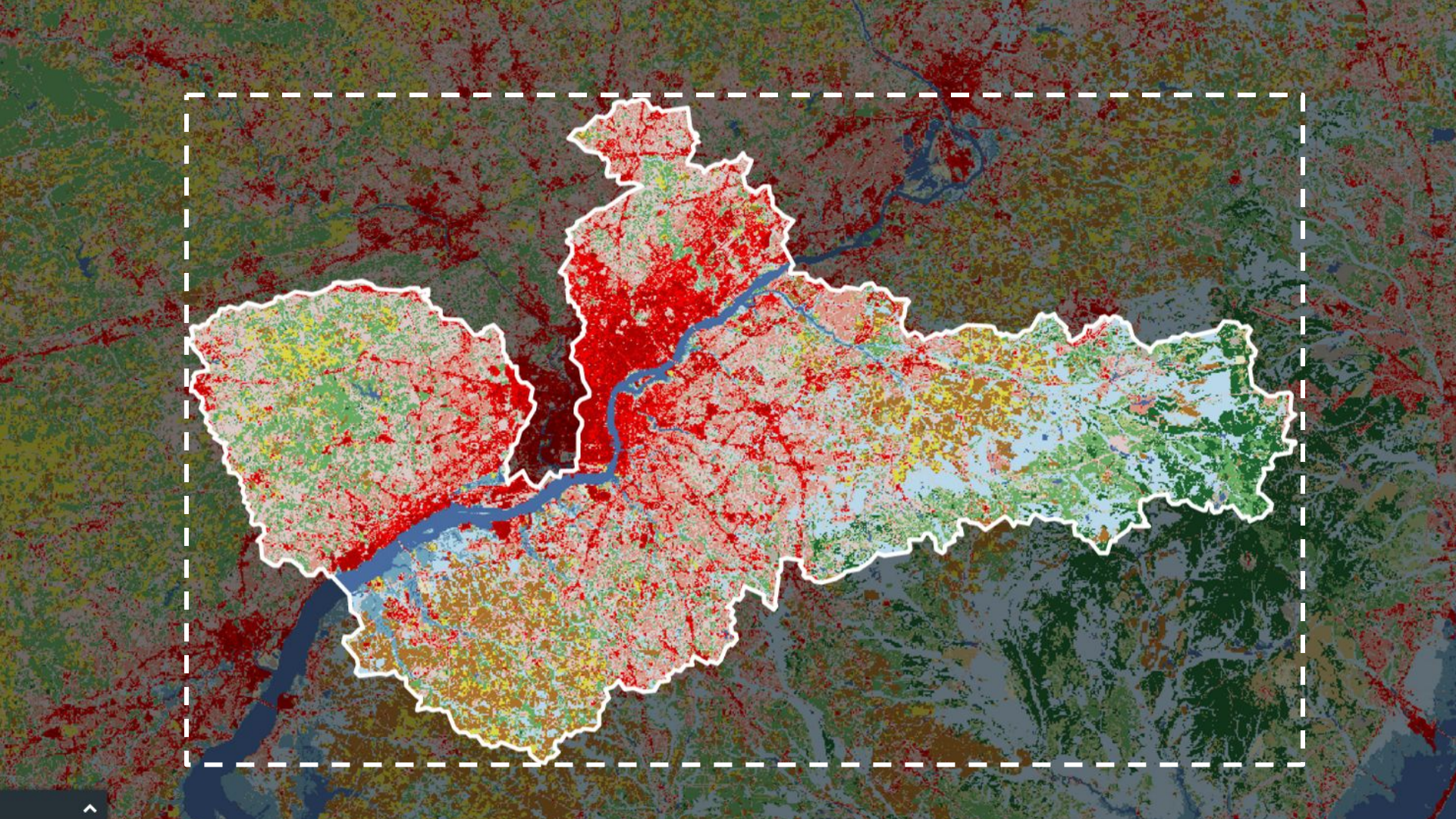
How it’s done

Optimize your rasters

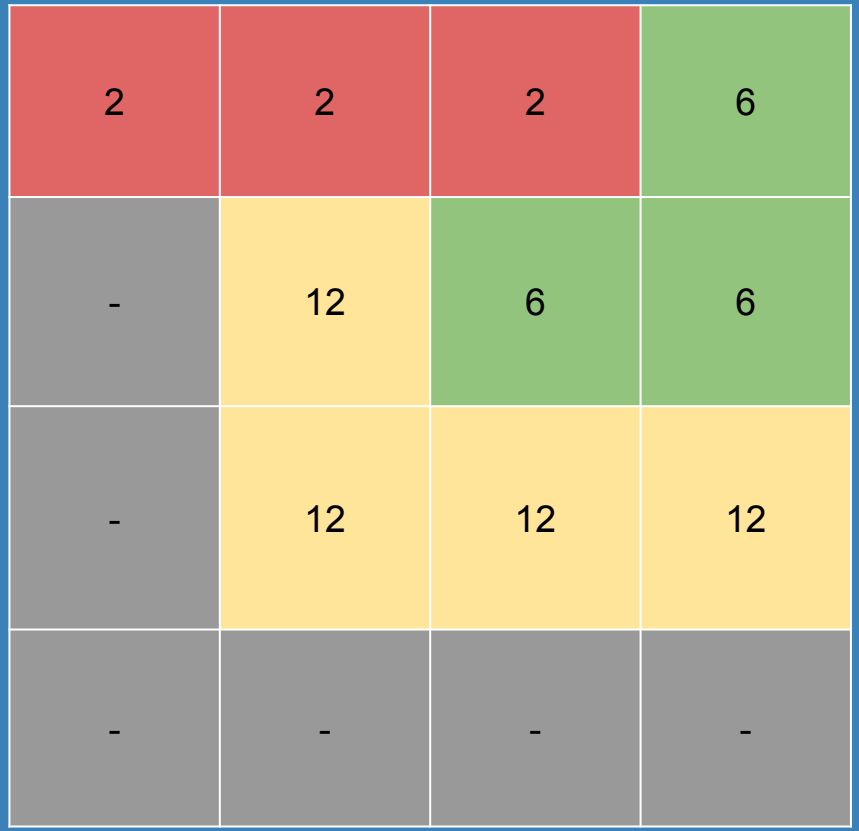
- Target S3
- Compress
- Internal tiles and and overviews
- Use VRTs (with /vsis3)

Optimize your reads

- **Windowed reads**

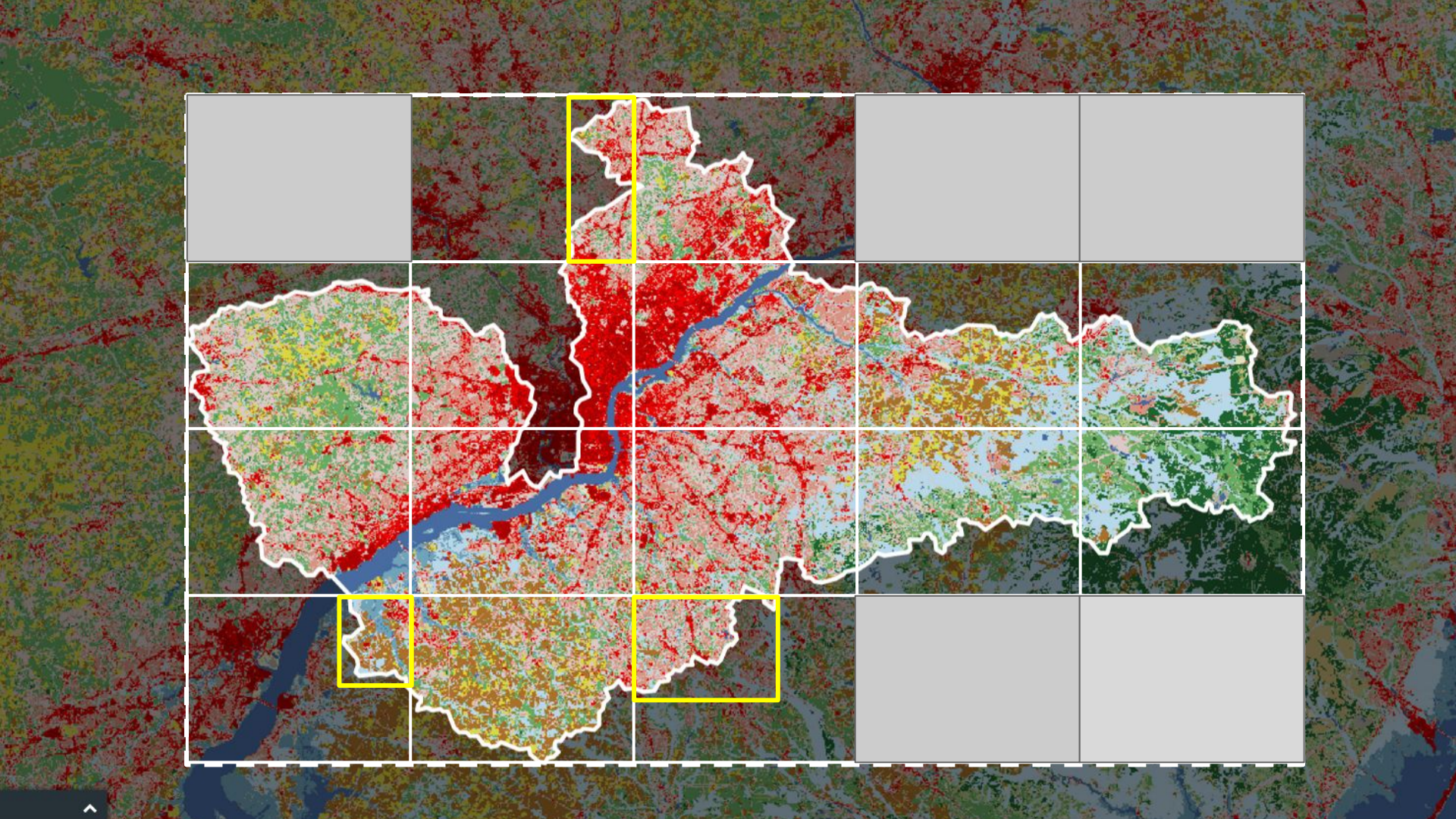


becomes...



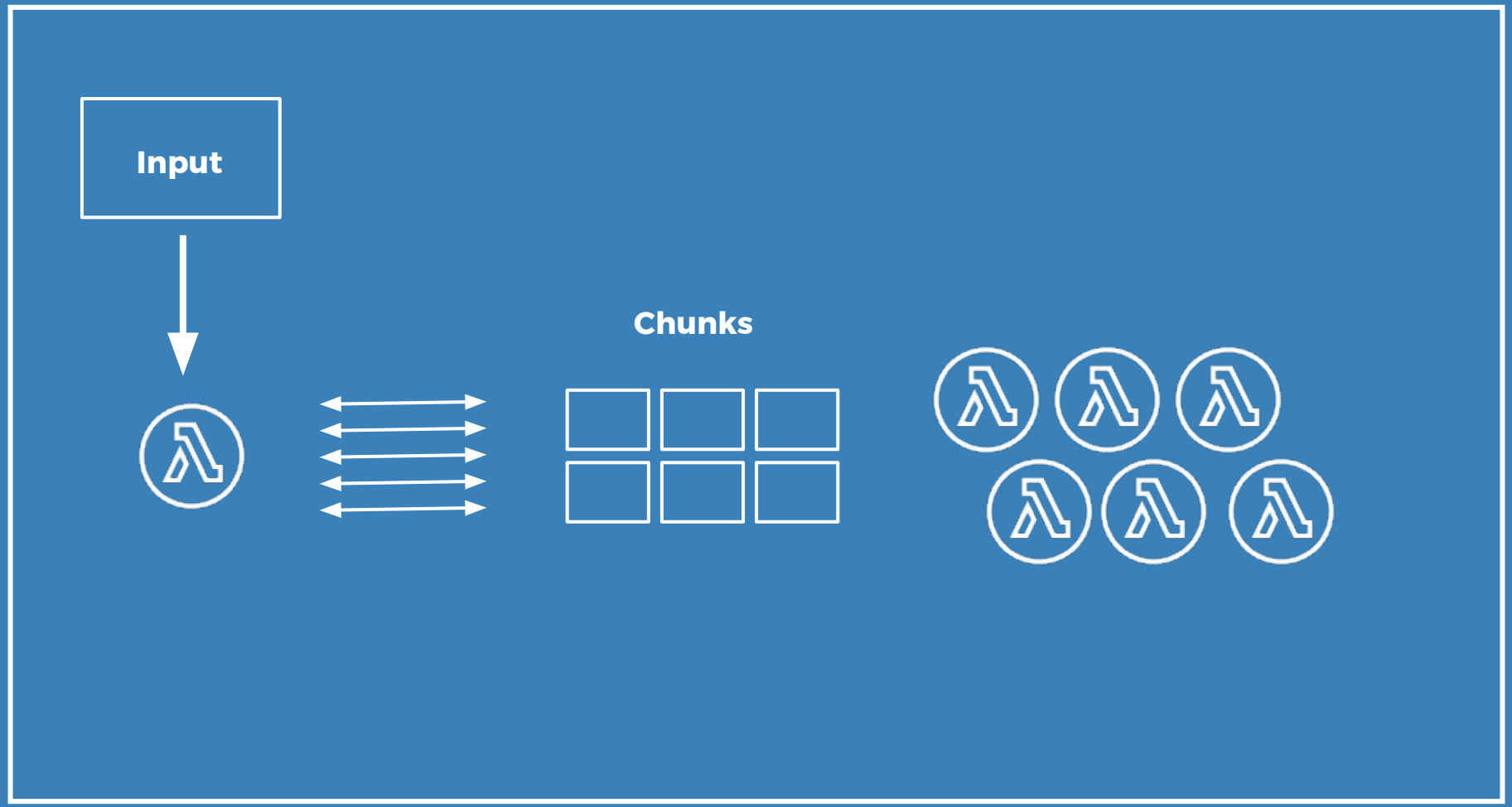
Optimize your reads

- Windowed reads
- Chunk windows



Optimize your reads

- Windowed reads
- Chunk windows
- Parallelize


Input
Chunks

Optimize your reads

- Windowed reads
- Chunk windows
- Parallelize
- Decimated reads + overviews for tiles

Serverless Framework

- NodeJS based CLI
- YAML config file to represent resources
- Don’t use it to package your deps

Dependencies

- Use a virtualenv
- Build in an Amzn Linux Container (if needed)
- Create custom zip
  - Symlink duplicated C libraries
  - High Compression

Demos!

http://raster.surge.sh/

Summary

- Use Lambda for geoprocessing
- Also, don’t use it
- Don’t abandon good engineering practices

Thanks!

Matt McFarland
mmcfarland@azavea.com

https://github.com/mmcfarland/foss4g-lambda-demo