ST. LOUIS

Regional Data Alliance

CONNECTING DATA FOR COMMUNITY CHANGE

An Open Stack for Open Data

Branson Fox – Lead Data Engineer, St. Louis Regional Data Alliance

About the Regional Data Alliance (RDA)

- Housed at the Community Innovation and Action Center of the University of Missouri, St. Louis
- Convening Data Partners in the Region from:
  - Government
  - Non-Profits
  - Community Organizations
  - Healthcare Institutions
- Goal of Making data more accessible to everyone

General Technical Challenges

- Aggregating and Combining Data from a Variety of Sources
  - A Robust ETL Process
  - Rigorous Error Handling
  - Infinite Room for Expansion
- Storing and Retrieving these Data in a Scalable Manner
  - Well Defined Schemas
  - Stateless, Scalable APIs
- Presentation and Accessibility of Data to All Types of Stakeholders
  - APIs for Developers
  - Dashboards for Specific Topics
  - Common Data Extracts for Everyone

Goals of the Stack

- Open
  - 100% Open Source
  - All Code Hosted In Public Git Repositories (**github.com/stlrda**)
- Agile
  - Full Continuous Deployment (5 Minutes from Push to Production)
  - Intuitive Tools for Collaboration
- Simple
  - Less (Code) is More
  - Modern Languages/Frameworks
  - Bug Squashing and Security Advantage
- Fast/Lightweight/Scalable/Portable
- Affordable
  - Don’t Spend more $$ Than You Have To

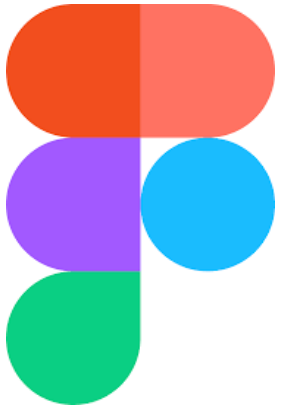
Behind the Stack

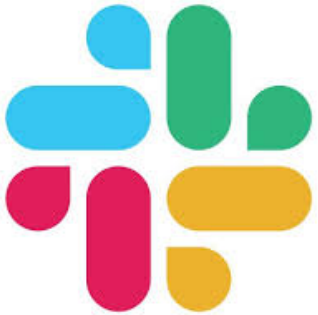
- Languages:
  - Python (Airflow, FastAPI)
  - SQL (Postgres, +Airflow, +FastAPI)
  - Javascript (React, Vanilla)
  - Docker (Compose)
  - Git (GitHub)
- Tools
  - Airflow for ETL Jobs
  - FastAPI for REST APIs
  - Javascript/React for Client Apps

Behind the Stack

- Language shouldn’t (doesn’t) matter.
  - **R** ETL Process for Crime Data
  - **R** Shiny Crime Dashboard
  - **Node** ETL Process for Vacancy Data

- The Stuff You Don’t Deploy
  - Project Management via GitHub Kanban Boards, Wikis
  - Diagramming, Planning in Miro
  - Front-End Mockups in Figma
  - Slack for Communication





The Stack

- AWS – EC2 VMs, S3 Buckets, RDS Databases, Route53 Domains
  - Nothing complicated, No Vendor Lock-In
- Everything is a container.
  - Airflow for ETL
  - FastAPI for REST APIs
  - React Apps in Front
  - Caddy to Route, Proxy and Secure all Deployments

Airflow

- From airbnb to the Apache Foundation, Written in Python
- “Cron on Steroids”
- Define DAGs entirely in Python, or even SQL or Docker
- Vertically Scalable with Parallel Execution, Horizontally Scalable with Celery or Kubernetes
- Robust Error Notification, Accessible Logs
- Web-Based Secure UI

FastAPI

- The Latest Contender in Python REST API Frameworks
  - Faster Than Django, Flask
- Supports Asynchronous Request Handling
  - Async Python Functions AND Async SQL Queries
- Automatic Documentation with Both ReDoc and Swagger
- Full Compliance with OpenApi Specification (OpenAPI.json)
  - We plan to build the data commons as an extension of this schema
- Ridiculously Easy Queries, Type Checking, and Data Structuring
  - Seriously, we've written APIs in minutes

Continuous Deployment with GitHub Actions + Docker

- GitHub Actions builds Docker Images on Push to Any Branch
- Images Stored in GitHub's Image Repository
- Server-side Docker-Compose includes the watchtower service, checks for updates to the image on the remote repository, and seamlessly updates deployment
- With appropriately sized image builds, we consistently achieve 3-5 minutes push to production.
- 100% Free Since our Repos are Public

Caddy

Caddy®

THE ULTIMATE SERVER

- A Modern Webserver, Written in Go
- The Easiest TLS Certificate Management Ever
- Concise Config Files
  - Do in 10 Lines what nginx or apache do in 100
- Used as a reverse proxy and static file server
- Option to use as a robust load balancer
- If you need higher throughput/more security:
  - consider Bunkerized-Nginx

React

- From Facebook, React is the de-facto front-end framework these days
- Every app is its own container.
  - Same Build/Deploy Process
  - Quick Changes to Individual Apps
  - Independent Scalability for Each App

Case Study 1: Crime Data

- Saint Louis Metropolitan Police Department Puts out Monthly Crime Reports, but they are messy
- Chris Prener at SLU writes a function in the Compstatr R package to scrape and reconcile data
- While at the Institute for Public Health at WUSTL, I Docker-ize this scraper, build an API with R Plumber, and build a dashboard using R Shiny
- Fast forward a year, how do I integrate this to the new infrastructure?

Case Study 1: Crime Data

- Airflow Supports Execution of a Docker Container – Moved the scraper Docker service to an Airflow DAG
- Data migrates from a file on disk (ouch) to a small Postgres Instance
- Re-write API in FastAPI within an evening, Dockerize and Deploy
- Proxy Dockerized Shiny App with Caddy at **apps.stldata.org**
- Redirect Old Sub-Domain to new URL

apps.stldata.org/crime

Case Study 2: Regional Data Exchange

- Regional Data is Fragmented
- Each County has their own portal, labeling of data
- We aggregate data from several local counties and organizations, and label it in a predictable, standardized way
- We release these aggregated data with an instance of DKAN

rdx.stldata.org

Case Study 3: Vacancy Data

- The City of St. Louis Doesn’t Know How Many Properties are Vacant
- In 2017, a methodology and dataset are created to define likely Vacancy. It has some limitations and isn’t updated
- Between 2018-2020 Jon Leek makes progress on the Regional-Entity Database (RedB)
- In Q4 2020, Walker Hamilton creates a city API for aggregated parcel data
- So how do we get to a monthly updated dataset of vacancy in the City of St. Louis?

Case Study 3: Vacancy Data

- RedB Completed Q3 2020 – Weekly Airflow DAGs Extract Parcel Data from MS Access Databases (Dump with mdbtools, store in S3, integrate with SQL scripts)
- Q4 2020 – City Parcel API is Scraped Monthly via Airflow
- Q4 2020 – Dave Menninger and Cam Barnes writing a Node Process to generate Vacancy likeliness from Parcel Data
- Q1 2021 – Node Process Dockerized and Integrated to Airflow
- Monthly Vacancy Dataset Generated

Coming Q1 2020

So What's Next?

- Q1 2021 – Anticipated Release of Data Commons
  - Using the OpenAPI Specification, create Query-able Tables and Basic Visualization within the browser (Swagger with better UX)
- 2021 – Community Information Exchange (CIE)
  - Connection of Community Resources and Service Organization
- 2020 – Beyond
  - Public Health Infrastructure (COVID, STI's, Crime & Violence)
  - Seeking Big Ideas – What does the region need to overcome its data challenges? What do you as a user of data want to see?

Reach out!

You have cool data ideas, want to add some commits to your GitHub profile, or just want to talk tech:

bransonfox@umsl.edu