canarie

GREN Map: High-Level Design Proposal

Ryan Davies, Software Developer | Oct 1, 2019

Objective

> Design a system to gather, store, and consolidate data representing the GREN network
  - …with a minimum of repetitive, onerous work
  - …for the purpose of visualizing it

Challenges

> Core problem: distributed data from hundreds of sources

- highly-variable source formats, e.g.
  - highly-curated databases
  - Ansible configurations
  - spreadsheets
  - human memory
  - etc.
- highly-variable source data definitions, classifications, identification, availability
- multiple sources may report shared network infrastructure, resulting in duplication
- data expiry
- data not published

> Work to collect and collate is onerous, and repeated by many actors many times.

Peer Projects

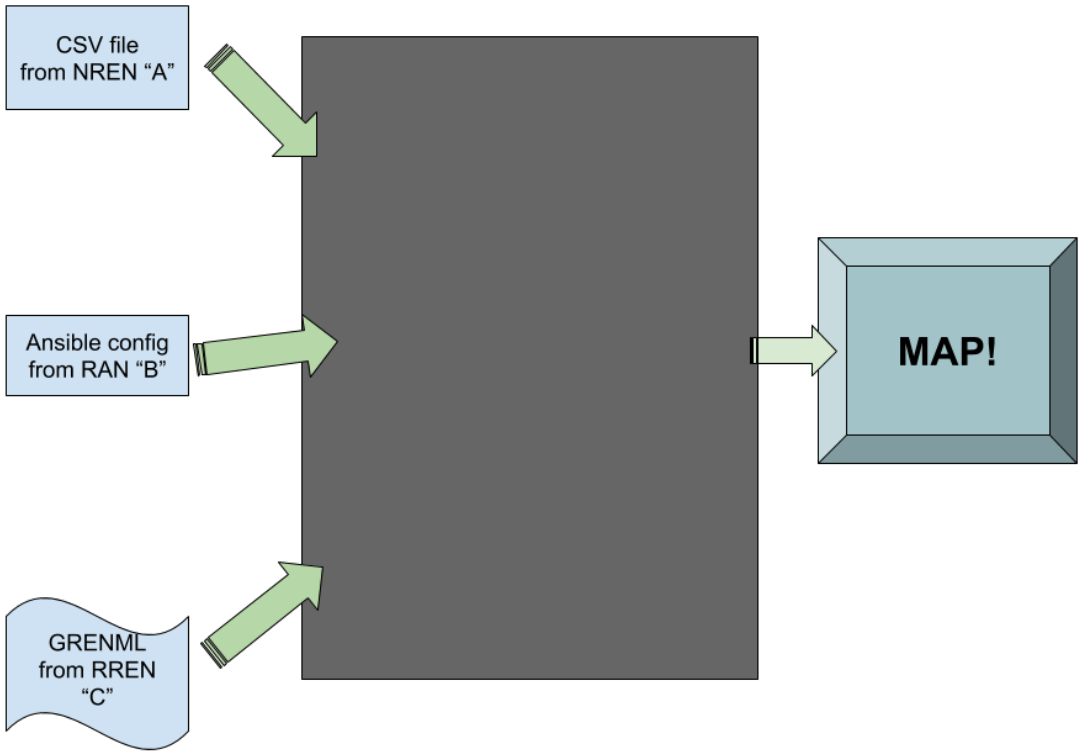
- There is no current implementation that captures the entire GREN in a dynamic visualization.
- Investigate parallel projects
  - Collaborate and harmonize efforts
  - However, the GREN map project has unique needs, as identified by our requirements gathering exercise

Proposed Solution

- Federate data collection and curation via a flexible module that:
  - ingests from various sources into a common schema
  - cleans & integrates each import with other sources, if applicable
  - allows manual curation after ingestion
  - exports to a common format (GRENML)
  - registers remote sources, if applicable, and polls routinely for fresh data ("pull" paradigm)
  - logs all activity, for monitoring, debugging, and security
- Once the above has been done, a reference visualization may be built upon any level of data in the hierarchy (most notably the top level).

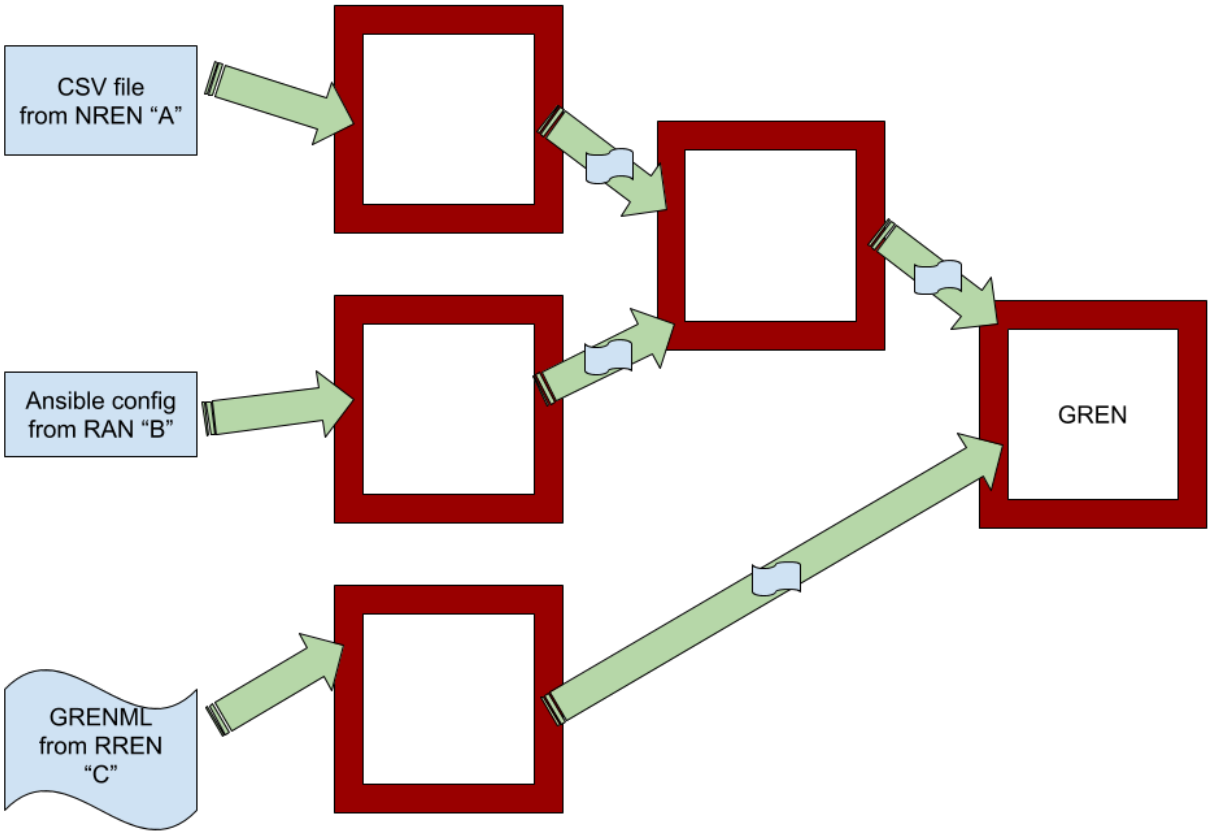
High Level

> Input … “magic” … output

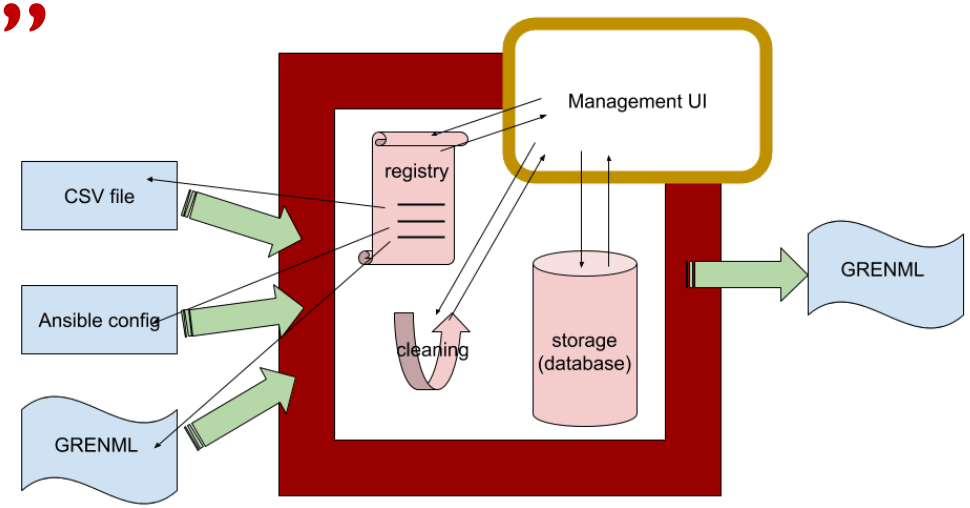


Actually…

> This magic black box would be a set of magic boxes, each maintained by a REN.



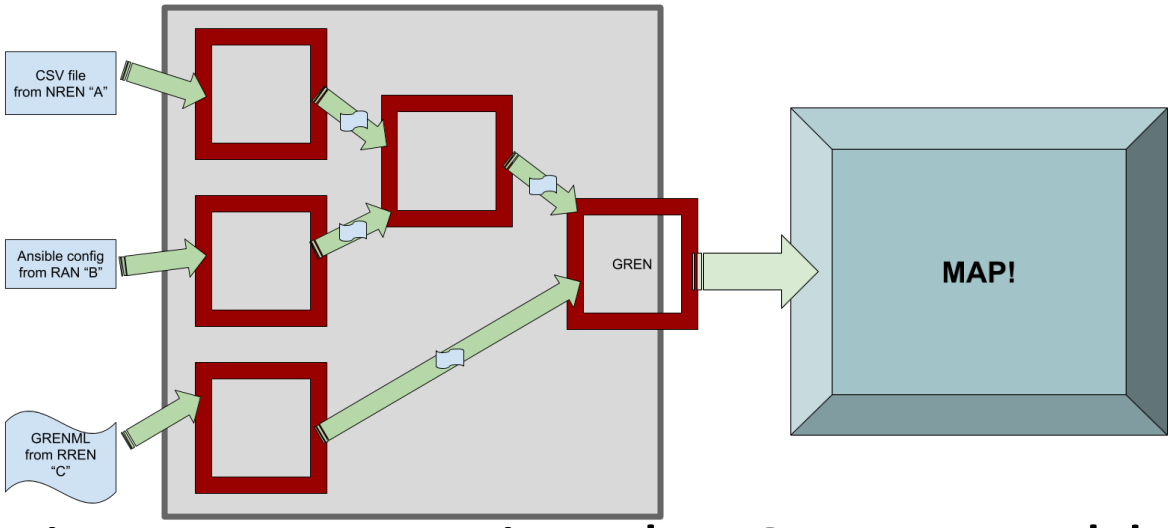
The “Flexible Module”



> Core features:
  - pull from and understand a variety of sources, including but not limited to spreadsheets/CSV, Ansible, and most notably, GRENML
  - mechanism to resolve conflicts and missing information in the set of sources
    - generally human-led resolution, remembered by the system
  - store the consolidated data
  - allow manual curation of the data
  - publish this data to GRENML on demand
  - publish this data directly for visualization
> Additional features such as authentication & authorization, fault tolerance, etc.
> Likely a set of Docker containers.

Hierarchical Federated Distribution

> Every REN publishing data would be expected to have an instance of the module.
  - RENs without the capacity to maintain their own instances may:
    - rely on others, likely their “parents” in the hierarchy, to spin up modules on their behalf, or
    - rely on their “parents” to simply publish their data (not preferred)



> A consortium representing the GREN would maintain the “top” level instance module.
  - This would provide the visualization of the entire GREN.

Module Extended Features

- Data sources listed in a registry
  - Various types of files, and remote GRENML sources
  - For remote sources, the remote source would have the ability to manage their own entry
- "Pull" vs. "push" paradigm
  - Pull seemed to be more in line with expectations of system function.
- Data cleaning & integration
  - This is an important core problem
  - Deduplication, various representations, ownership, missing data, conflicts
  - Implement a set of rules, managed by a human
    - "when you see this, do that"
- Manual data curation would have a "preview" function

Participation Request

> Support from the GREN community is solicited to assist in the creation various components of this module

> CANARIE will identify work packages in a microservice paradigm on the following topics:

- Data ingestion
  - GRENML
  - CSV
  - Ansible
- GRENML export
- DB: staging & published
- Management:
  - API + server
  - UI
- Data source registry
  - Tree visualization
- Data source polling manager
- Cleaning & integration
  - Registry of rules
  - CRUD UI for rule management
- Authentication & authorization
  - On all API endpoints
  - Client for polling manager
- Visualization API + server
- Logging + visualization

Work Packages

> Each work package would:
  - be scoped between 1 FTE week and 3 FTE months
  - include a set of design requirements, plus:
    - existing infrastructure & documentation
    - example input/output where appropriate
  - indicate a required skillset for development, likely including:
    - Docker
    - Python
    - XML
    - Database ORMs
    - Javascript (React) + HTML
    - REST APIs

> Regular communication between teams actively developing work packages and CANARIE as project co-ordinator would be encouraged.

canarie

canarie.ca | @canarie_inc