GREN Map

Development Roadmap and Collaboration Model

Version: 1.0

Author: GREN Map Working Group

December 2019

Table of Contents

Project Vision

The goal of the Global Research and Education Network (GREN) Map Working Group (WG) is to guide the development of a continuously up-to-date, dynamic visualization of the GREN to demonstrate the value of R&E Networks to all stakeholders.

To support this effort, CANARIE proposes a high-level model for a distributed system to meet the goals identified by the WG:

1. Dramatically increase the refresh cycle frequency of the network mapping information;
2. Dramatically reduce the effort required to supply, maintain, and collect the information; and
3. Provide a flexible platform for dynamically visualizing the global map.

Call for Participation

The WG accepted the high-level system model proposal above, and implementation is to begin immediately. CANARIE will supply a substantial portion of the required development effort, but additional support from the community is required in order to meet the project's timeline.

The WG is soliciting commitments from all Research and Education Networks (RENs) — including subnational, national, and supranational networks to lend development capacity to collaborate in this initiative.

Commitments of design and development effort may be time-based or linked to the implementation of specific features or components (as described in the Roadmap section). They may vary in time, duration, and skillset, with the following minimums:

- Duration: minimum of one calendar week, totalling a minimum of two full-time employee equivalent (FTE) weeks.
- Skillset: proficiency in source code control, Python, and web applications is required for all work; additional requirements for each specific feature are outlined in the Roadmap section.

Development assistance that does not meet these minimums is welcomed on an ad-hoc basis and will augment the capacity of an active team.

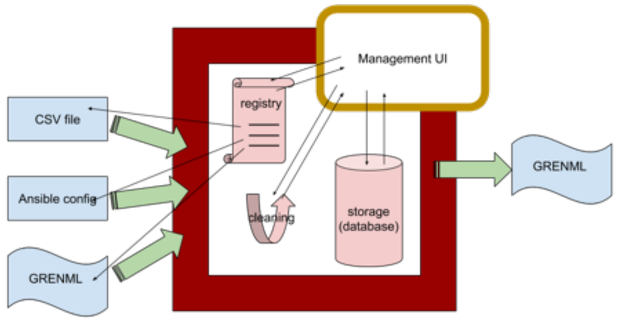
This request for assistance focusses on the earlier phases of the project in the first two quarters of 2020. Commitments for the last two quarters of 2020 are also welcomed; details of that work will be released in Q2 2020 as development progresses.

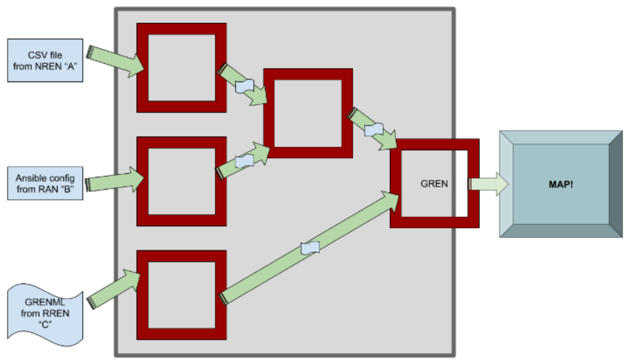
Additional details relevant to this request follow in this document.

System Overview

The system model involves the development of a flexible module, an instance of which would be maintained by each REN participating in the GREN, large or small, at every level. RENs connecting other smaller regional RENs would consolidate all of their data and publish it to be further consolidated. Ultimately, a module at the top of the hierarchy would contain all mapping data for the entire GREN.

Modules would be instantiated by Docker, interconnected via a registry system, communicate via APIs, and interchange data in a format being developed by the WG, based on NML, called GRENML.





A brief overview of the system is available in the accepted proposal, and more detail is provided in a high-level design document. The GREN Map wiki is the archive for all documents related to the project, including these: https://wiki.geant.org/display/GlobalMap/Global+Interactive+Map

Timeline

Development of this system is to begin in January 2020.

In order to spur widespread adoption of this system, the WG intends to present a prototype of the distributed system TNC20 in June 2020, with functioning drafts of the most fundamental components in place and data representing a small diversity of early adopter RENs.

To ensure that this milestone is met, work has been broadly planned out in phases, roughly mapping to calendar 2020 quarters, as follows:

Q1 - Phase 1 — Initial infrastructure

Q2 - Phase 2 — Distributed Database

Milestone: TNC20, June 2020

Q3 - Phase 3 — Visualization

Milestone: Canadian NREN Assembly, October 2020

Q4 - Phase 4 — Data Complexity and Security

Components have been placed in the phases corresponding to when they will be needed according to this timeline. If sufficient development capacity is achieved, earlier completion of later phases is possible.

Collaboration Model

CANARIE will supply a substantial portion of the required development effort and has thus assumed a coordination role for participating teams.

Contributions by developers will follow well-established open source collaboration practices.

All development shall be fully tested at multiple levels, unless explicitly specified. This includes unit tests of non-trivial functions and functional integration tests of complex features. Full documentation of the implementation approach, component behaviour and interaction, and system administration, is expected.

The detail in each of the work packages has been deliberately kept at a high level for this stage. The aim is to maximize the opportunity for an Agile approach within the framework and constraints of a diverse, distributed effort.

Roadmap

A high-level list of core components has been outlined in the High-Level Design document for this project. More information about each item may be found in that document. Estimates of the expected level of effort required for each of these components and features is outlined in this section, prioritized, and grouped into phases as described above.

Note: "FTE week" refers to the ideal capacity of a single developer for one week, not including overhead. Examples: one developer working half-time for two weeks = 1 FTE week; three developers working full-time for two weeks = 6 FTE weeks.

Phase One: Infrastructure

This phase is expected to be completed during Q1, calendar 2020.

Expected outcomes:

- A solid container-based development platform with major fundamental pieces in place
- The ability to manually maintain a database of network elements

1.a Docker Infrastructure

Implement a three-tier set of Docker containers (in a Docker Compose) to provide a base system for database (Postgresql), server processing (Python 3, Django), and a web server (Nginx).

Required skills: Docker

Estimated time: 4 FTE weeks

1.b Database

Capture all elements and attributes supported by GRENML in the Django ORM (Object Relational Model).

Required skills: Python, Django, Postgresql

Estimated time: 4 FTE weeks

1.c Data & Metadata Management

Implement the Django Admin UI to manage the ORM as defined above. Configure basic authentication and authorization, adjust it to sensible needs, and apply an appropriate skin/theme.

Required skills: Python, Django, Web

Estimated time: 4 FTE weeks

Phase Two: Distributed Database

This phase is expected to be completed during Q2, calendar 2020, in time for a demonstration at TNC20.

Expected outcomes:

- A set of modules, each able to import and export data, arranged in a hierarchical distributed database, with one of the modules containing consolidated data for the whole network. No automatic data manipulation is expected at this stage.
- A very basic visualization of this consolidated data shall be prepared based on this system for the purpose of demonstrating the concept and progress towards it at TNC20.

2.a GRENML Library

A prototype Python library to import and export GRENML (an extension of NML, which is XML) to/from Python objects representing all elements and attributes supported by GRENML. Objects and fields should map 1:1 to the ORM defined above, but this library should not be related to the ORM for dependency reduction when this library is used outside this module.

Required skills: Python, XML, thorough testing

Estimated time: 24 FTE weeks

2.b Data Input/Output

Implement HTTP endpoints to import and export GRENML to/from the database ORM, via the GRENML library.

Required skills: Python, Django, HTTP, Docker (optional), thorough testing

Estimated time: 6 FTE weeks

2.c CSV to GRENML Conversion

Standalone Python script to use the GRENML Python library to convert CSV data into GRENML.

Required skills: Python, XML, Docker (optional)

Estimated time: 4 FTE weeks

2.d Hierarchy Source Registry

To enable the hierarchical distribution of data, the Maintainer of each module instance will maintain a list of the modules “below” containing data to be consolidated.

Required skills: Python, Django, HTTP

Estimated time: 2 FTE weeks

2.e Hierarchy Source Polling

To enable the hierarchical distribution of data, each module instance will consult its list of the modules “below” and consolidate their data with its own, via GRENML export and import features. This process should be schedulable on an individual and recurring basis, and run on demand.

Required skills: Python, HTTP, Docker (optional)

Estimated time: 6 FTE weeks

2.f Basic Visualisation Prototype

Simple representation of nodes and edges from an instance of the module (likely consolidating other modules), using a web-based map rendering library such as Leaflet, to support a demonstration at TNC20.

Required skills: web, Django, Python

Estimated time: 4 FTE weeks

Phase Three: Visualization

Feature-rich dynamic representation of consolidated network mapping data in an embeddable web component.

The details of the items in this section have not yet been determined. It is expected to be undertaken in Q3, calendar 2020.

Features for implementation may include:

- API & client
- Data subset querying
- Embeddable map renderer
- Basic map display navigation: zoom, pan, click, etc.
- Configurable embedding details
- Layering, based on various attributes of map elements
- Zoom-level-based automatic layer filtering
- Flexible branding

Required skills: web, Django, Python, thorough testing

Phase Four: Data Complexity & Security

Scalability, resilience, and security enhancements to the system to support a widespread adoption.

The details of the items in this section have not yet been determined. It is expected to be undertaken in Q4, calendar 2020 and beyond.

Features for implementation may include:

- Maintainer authentication & authorization
- Data authentication
- Data deduplication
- Data conflict resolution
- Data change management
- Staging & publication
- Source tree visualization & error detection
- Fault tolerance & resilience
- Additional source conversion (e.g. Ansible)

Required skills: Python, Django, Docker, Ansible, XML, web, security, thorough testing