Running your own Vector Tiles Stack

Alex Rolek & Gautam Dey - Scoutred
Jeffrey Johnson - Terranodo LLC

As the use of Vector Tiles has exploded in the past few years, more and more use cases have emerged that require setting up an ‘end to end’ stack for preparing, serving and consuming vector tiles in a web application. While many projects may be able to simply use basemap tiles from a CDN provided by a service provider, many other use cases call for serving vector tiles from existing and often private datasets or databases. This workshop will walk the participant through the process of preparing data for serving as vector tiles, loading and optimizing the database and configuring and scaling Tegola (http://tegola.io) as a vector tile server for serving and caching tiles conforming to the Mapbox Vector Tile (MVT) specification. With the MVT web service configured, the participants will style their data using Maputnik, a mapbpox-gl based visual style editor and prepare a simple web application with OpenLayers3 or mapboxgl-js using this style as well as providing user interface widgets to inspect the data contained in the tiles. Participants will leave this session prepared to setup and maintain an ‘end to end’ scalable vector tiles stack on their own infrastructure for use in web mapping applications.

Participants should have basic web development skills, bring their own laptop with PostgreSQL + PostGIS installed and their own data for use in the workshop. Advanced participants will use Docker to manage their stack for development and/or deploy it to a remote container host.

Advanced developers will be led through the process of building a tegola binary from its Go language source code and a general overview of the codebase. Advanced Javascript developers will be led through an overview of the client side components and how to setup a development environment for working with these libraries and tools. Additionally, the workshop will go over the process of loading data from OpenStreetmap pbf files into PostGIS using imposm3 and configuring this database for use with Tegola.

Outline

- Vector Tiles ecosystem overview
    - The MVT Specification
    - Mapbox Studio
    - Mapzen
    - OpenMapTiles
    - GeoServer
    - Others
- The Vector Tiles Stack
    - Imposm3
    - PostGIS

    - Tegola
    - Maputnik
    - OpenLayers3
    - Mapboxgl-js
- Preparing data for serving vector tiles
    - Projections
    - Simplification
    - Zoom Level Filtering
    - Attribute types and filtering
    - Using Imposm3 to import OSM pbf to PostGIS
    - Loading data using ogr2ogr
- Setting up Tegola
    - Using a release binary
    - Building from Source
    - Using docker-compose
    - Configuration toml
        - Layers
        - Maps
        - Filters
        - Capabilities
        - Hstore fields
    - Maximizing/optimizing resource usage
    - Horizontal Scaling
    - Setting up a Cache
- Using Maputnik to create mapboxgl json styles
    - Configuring your vector tile service
    - Adding layers to your map
    - Point Symbology
    - Line Symbology
    - Polygon Symbology
    - Using filters
    - Configuring Zoom levels
- Setting up a web application
    - Setting up basic application in OL3
    - Setting up basic application in Mapboxgl-js
    - Adding your data service
    - Configuring the style
    - Adding a feature inspect widget
- The vector tile stack in production
    - Deploying your application
    - Optimizing PostGIS tables
    - Caching and CDN
    - Scaling tegola