Traffic Ops

Go Rewrite

Progress

Dewayne Richardson
dewrich@apache.org

Original Initiative

- Mojolicious Perl API Version 1.2 - (2010 - Present)
- Perl to Go with the assumption of API compatibility
- Minimize Deployment footprint (use the same rpms)
- Design philosophy "override" through route mapping to "Strangle Out" the Perl

Traffic Control 2.1

Traffic Ops 2.1

Features TC Summit Fall 2017

- 1.2 Go Proxy Conceived (July 13, 2017)
- API Version 1.2
- Low Level Features
  - Configurable Logging
  - Go Perl Config parser
  - tocookie for backward compatibility to Mojolicious Perl
  - Add header (via wrapper) to Go endpoints
  - CORS headers
  - Unit Tests
- Documentation

Traffic Control 2.2

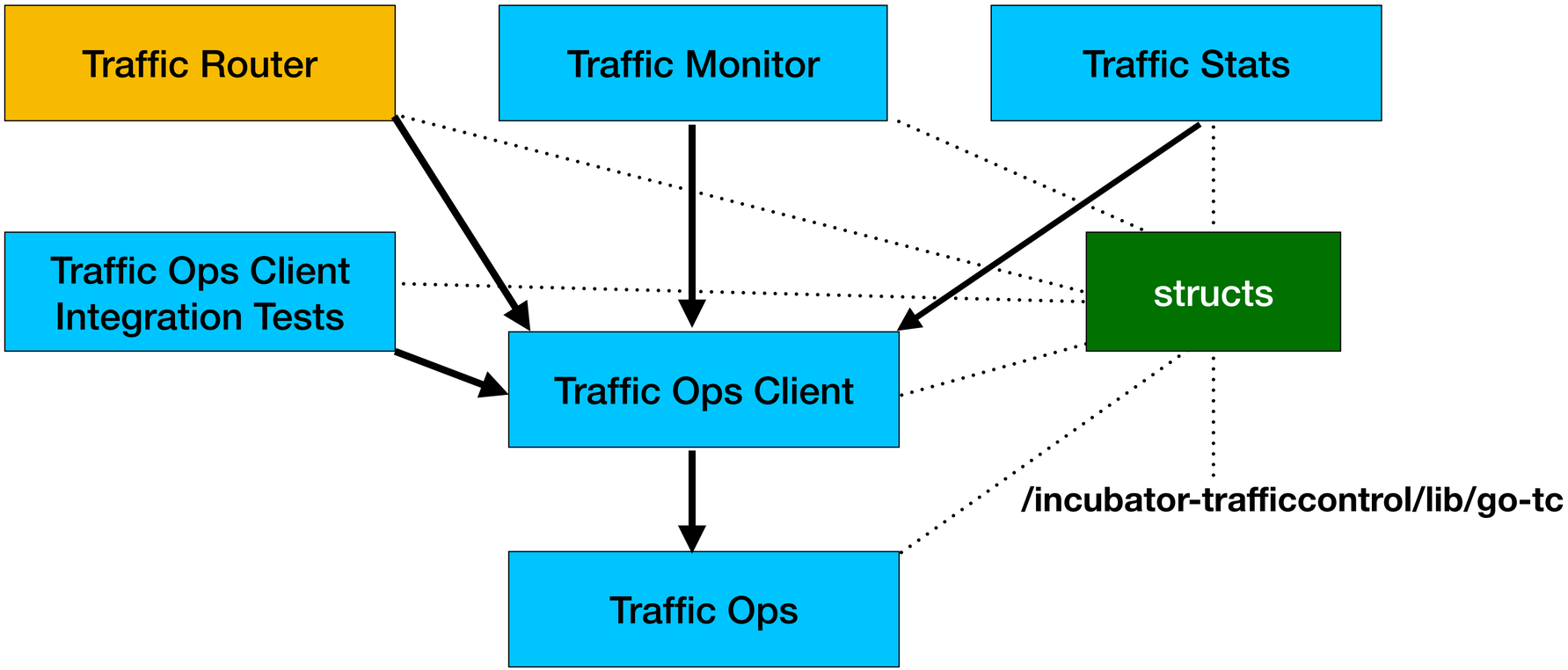
Traffic Ops 2.2

- API Versions 1.2 and 1.3
- CRUD Interfaces framework to standardize Postgres interaction
- Login API - PR
  - Mojolicious Perl (still required)
- Change Log Interface

Traffic Ops 2.2

- Tenancy - scope control of Delivery Services and Users
- CHANGELOG.MD (https://github.com/apache/incubator-trafficcontrol/blob/master/CHANGELOG.md)
- APIs are moving Semantic Versioning - Major.Minor.Patch
  - Consensus still needed

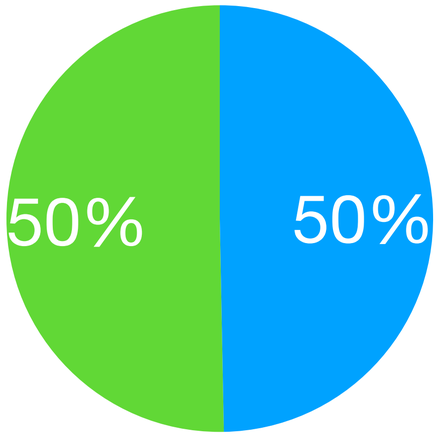
Traffic Ops Go
Components
(master)



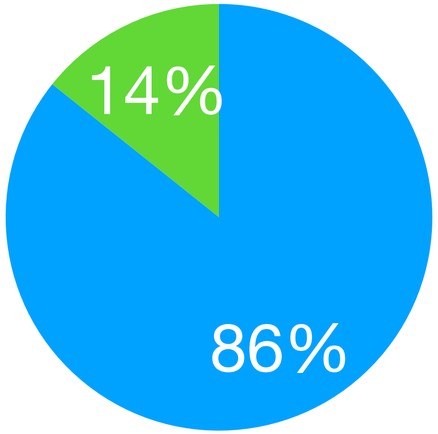
Traffic Ops APIs

by Language for TC Consumers

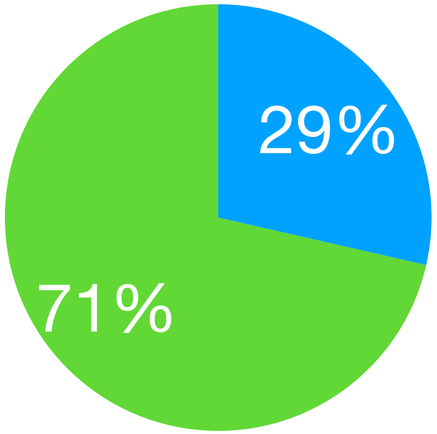
Traffic Portal



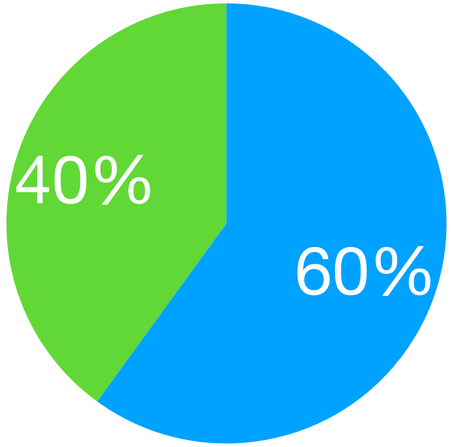
Traffic Router



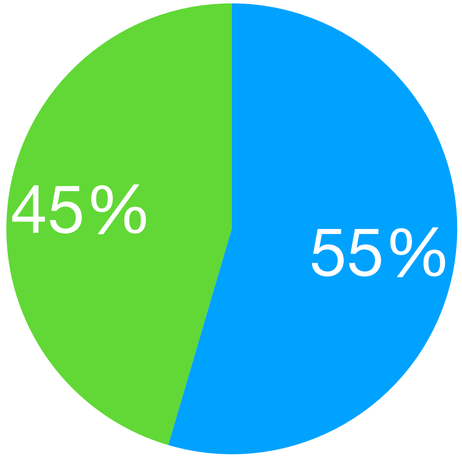
Traffic Monitor



Traffic Stats



ORT



Traffic Control 2.3

Traffic Ops 2.3

- Collapse Traffic Ops Migrations
  - (remove JvD's slow migration 20170205101432_cdn_table_domain_name.go)
- Fix camelCase Consistency
  - secondaryParentCachegroupId vs secondaryParentCacheGroupId

Traffic Ops 2.3

- Move away from “xmlId” (deliveryservice “name”) and “xmppId” (hash_id)
- Correct Foreign Key naming conventions
- Foreign Key ID's would be 'cdnId' vs. 'cdn' (database alignment changes would come later)

Lessons Learned

- Go inexperience, discovering better ways for struct organization
- Use of Anonymous Structs
    - (https://play.golang.org/p/ZbcTwLF_e2H)
- Go's strong typing breaks the loosely typed Perl 1.2 API
- nil struct field pointers
    - (https://github.com/apache/incubator-trafficcontrol/blob/master/lib/go-tc/servers.go#L82)

Legacy Problems

- Inconsistent Mojo Routes patterns make it a challenge to build consistent frameworks
- Existing route patterns prevent proper matching to allow pass through to Perl
  - For Example:
    - /cdns vs /cdns/capacity
- Move existing Traffic Control API Consumers away from the legacy 1.2 API

Traffic Ops Roadmap

Traffic Ops API Roadmap

- The need for Traffic Ops API 2.0
- Bulk CRUDs using Array Json structures
- Consistency will allow for Go scaffold generation
- Top Down Design of the API with Swagger for consistency

Traffic Ops API Roadmap

- Top Down Design of the API with Swagger for consistency
- Phase out the TO clients in favor of TO client generation when needed from Swagger Tooling
- Explore Swagger for Server Side Stub generation
- pyswagger (Python), go-swagger (Go)

Traffic Ops API Roadmap

- swagger-codegen - Available Languages (86)
  - android, aspnet5, aspnetcore, async-scala, bash, cwiki, csharp, cpprest, dart, elixir, flash, python-flask, go, groovy, java, jaxrs, jaxrs-cxf-client, jaxrs-cxf, jaxrs-resteasy, jaxrs-resteasy-eap, jaxrs-spec, jaxrs-cxf-cdi, inflector, javascript, javascript-closure-angular, jmeter, nancyfx, nodejs-server, objc, perl, php, python, qt5cpp, ruby, scala, scalatra, finch, silex-PHP, sinatra, rails5, slim, spring, dynamic-html, html, html2, swagger, swagger-yaml, swift, swift3, tizen, typescript-angular2, typescript-angular, typescript-node, typescript-fetch, akka-scala, CsharpDotNet2, clojure, haskell, lumen, go-server, erlang-server, undertow, msf4j, ze-ph

Traffic Ops API Roadmap

- Hide the Traffic Ops database entirely behind the API
  - Remove dependency on "Join" tables through the API (for example “/profile_parameters”)
- Atomic API's
  - Nested relationships within API JSON objects

Traffic Ops API Roadmap

- Extensions through Proxying
- Consider Postgres Views to potentially allow for TO database versioning
- Refactor the “types” table
  - separate tables by “use_in_table” column to allow for type safety needs with Go (Go has no Generics)
  - reduces the use of constraints to help with data integrity

API Resources

https://cwiki.apache.org/confluence/display/TC/API+Guidelines