APACHE TRAFFIC CONTROL

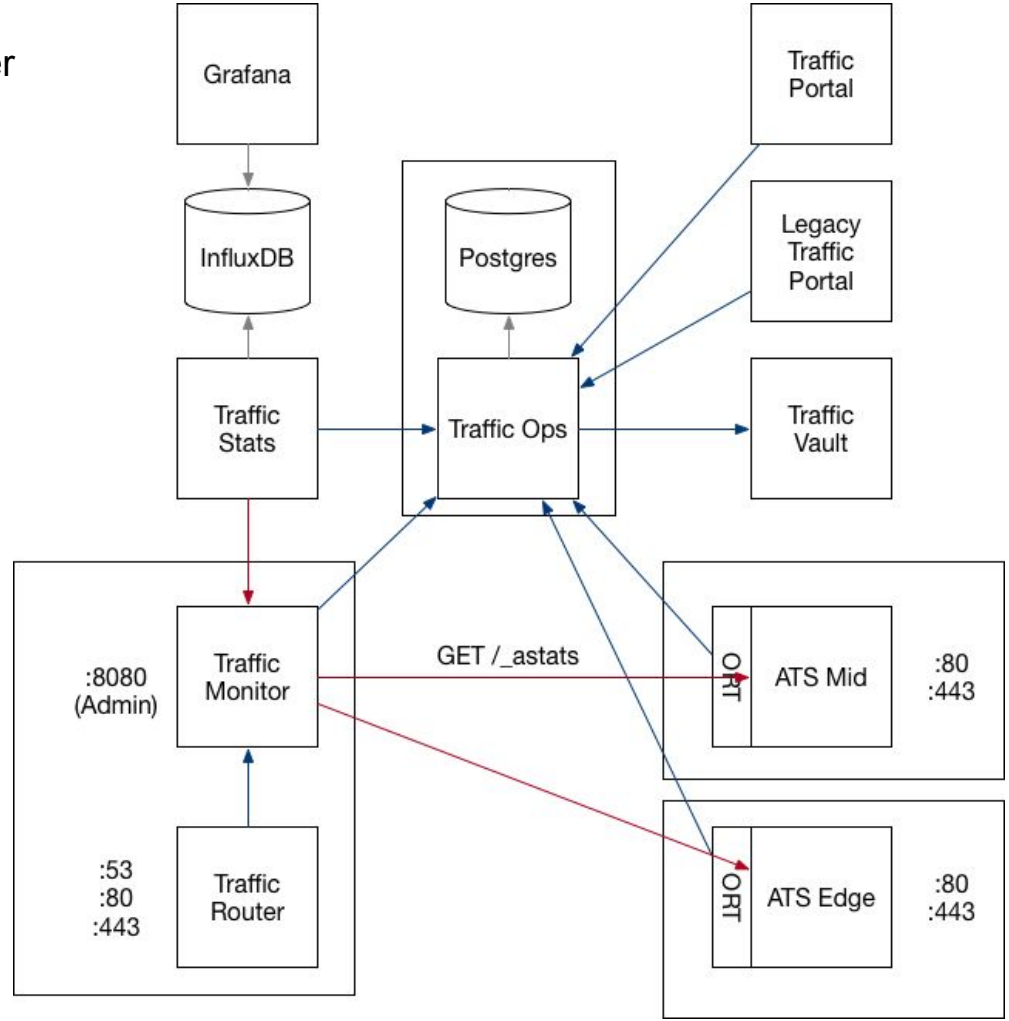
Apache Traffic Control & Lua

Matt Mills
Comcast

Who am I & What is this talk about?

- Sr. Systems Engineer - Comcast CDN - Denver, Colorado
- Somewhere in between Dev and Ops
- How I want to use Lua within Traffic Control
- Why I want to build a generic layer for describing programmatic functionality
- Some tangentially related problems

Traffic Control Refresher



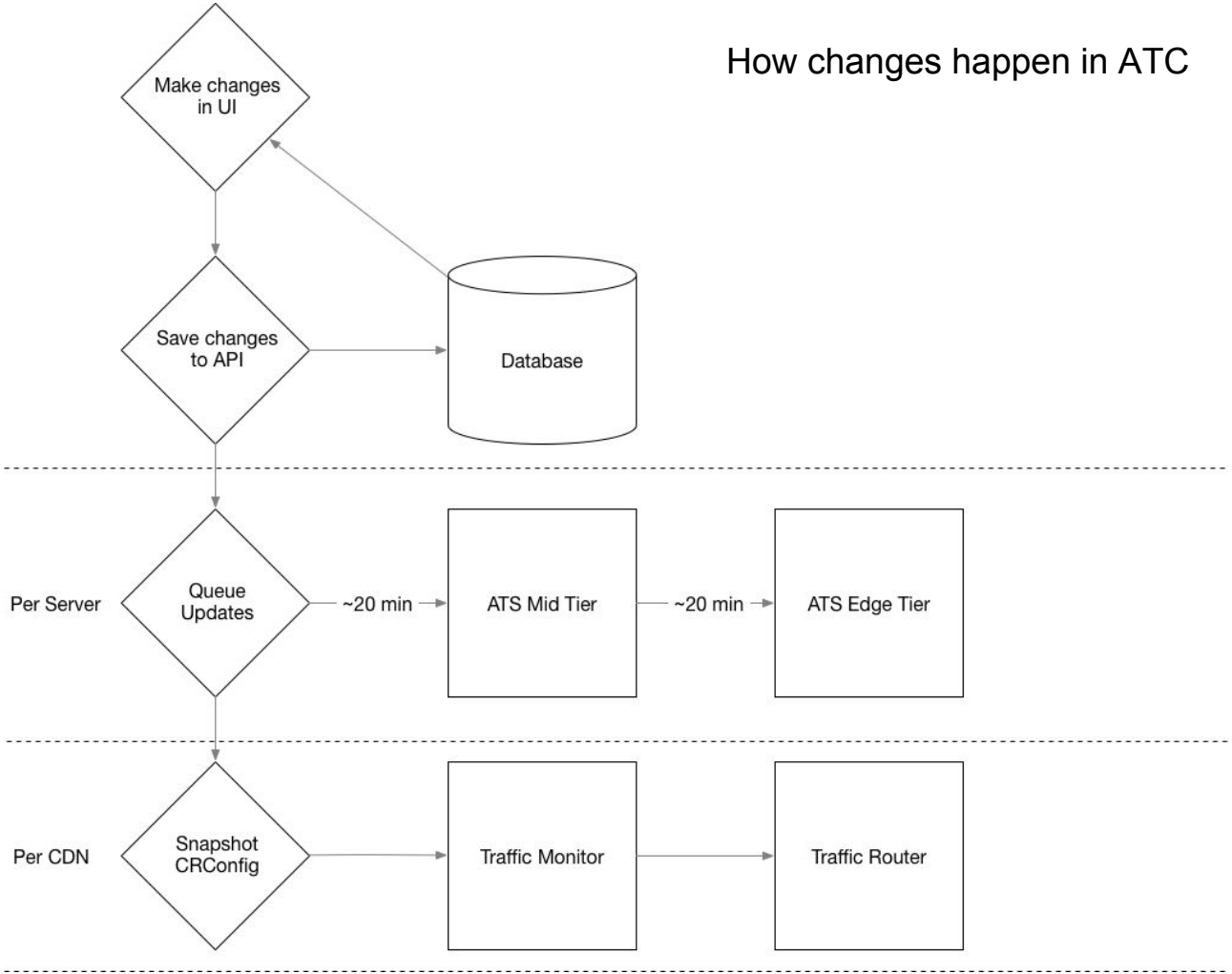
Problem:

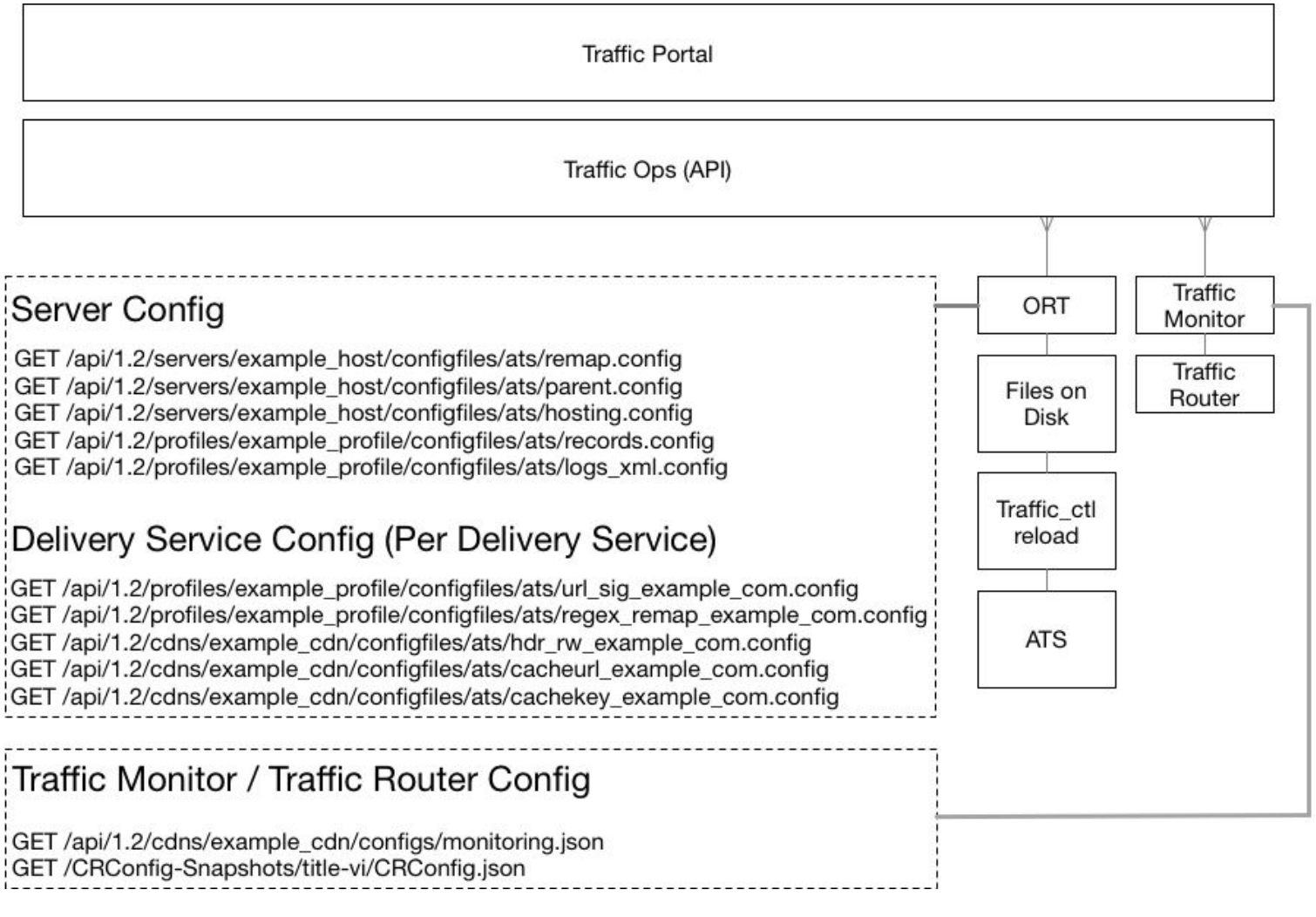
1. UI, database schema, and API all speak ATS
    a. Self service users don’t understand regex or header rewrite or...
    b. Can’t leverage non-ATS caching proxy without a LOT of development
2. Adding new custom delivery service functionality requires touching too many components

Related problems:

1. CDN changes propagate too slowly and can cause a thundering herd.
2. Changes are not atomic (Queued updates & Snapshot CRConfig)
    a. Different users’ changes can collide
    b. User A can push out User B’s changes before User B is ready

How changes happen in ATC




Traffic Portal
Traffic Ops (API)
Server Config
GET /api/1.2/servers/example_host/configfiles/ats/remap.config
GET /api/1.2/servers/example_host/configfiles/ats/parent.config
GET /api/1.2/servers/example_host/configfiles/ats/hosting.config
GET /api/1.2/profiles/example_profile/configfiles/ats/records.config
GET /api/1.2/profiles/example_profile/configfiles/ats/logs_xml.config
Delivery Service Config (Per Delivery Service)
GET /api/1.2/profiles/example_profile/configfiles/ats/url_sig_example_com.config
GET /api/1.2/profiles/example_profile/configfiles/ats/regex_remap_example_com.config
GET /api/1.2/cdns/example_cdn/configfiles/ats/hdr_rw_example_com.config
GET /api/1.2/cdns/example_cdn/configfiles/ats/cacheurl_example_com.config
GET /api/1.2/cdns/example_cdn/configfiles/ats/cachekey_example_com.config
Traffic Monitor / Traffic Router Config
GET /api/1.2/cdns/example_cdn/configs/monitoring.json
GET /CRConfig-Snapshots/title-vi/CRConfig.json
ORT
Files on Disk
Traffic_ctl reload
ATS
Traffic Monitor
Traffic Router

Delivery service configuration

Edge Header Rewrite Rules	set-header Accept-Encoding None [L]
Mid Header Rewrite Rules	
Traffic Router Additional Response Headers	
Traffic Router Log Request Headers	
Regex remap expression	^/([^/]+)/(.*)$ https://$1.example.com/$2
Cache URL expression	
Raw remap text	@plugin=tslua.so @pparam=/opt/trafficserver/etc/trafficserver/test.lua

What does “custom delivery service functionality” mean?

Support in-url range requests

- Customer had legacy clients that couldn’t support Range: header
- But also clients that DO support and use Range header
- So they put it in the URL (/path/to/file/range/100-200)
- Need to parse in-URL range request when present, and convert to normal Range header at the edge
- 11 lines of Lua

Manipulate URLs in flight to “capture” sessions

- Customer wanted our CDN to use another CDN as upstream
- But they can’t change the URLs that they use for playback (streaming video)
- Also, they didn’t want us to talk to the other CDN about it
- Need to manipulate API responses as well as HLS m3u8 manifests
- Replace existing URLs with our CDN edge URL
- 9 Lines of Lua

Speed test (mod_hotair/generator at the edge)

- Generate hot air (bytes) at the edge, no upstream request
- Variable size (1 byte to 1 gigabyte)
- Should respond with 200 OK to POST (upload speed test)
- Upstream requests to other URLs should still work
- Preferably doesn’t crash server or leak memory
- 57 lines of Lua

AWS S3 v4 Signing

- Before the existing ATS s3_auth plugin was updated to support v4 we had a customer requirement to support upstream auth against AWS S3 v4 signing
- Didn’t need to support full feature set, just sign GET requests
- LUA OpenSSL binding library
- Now we can do HMAC SHA256 in Lua!
- 109 Lines of Lua

Simpler stuff

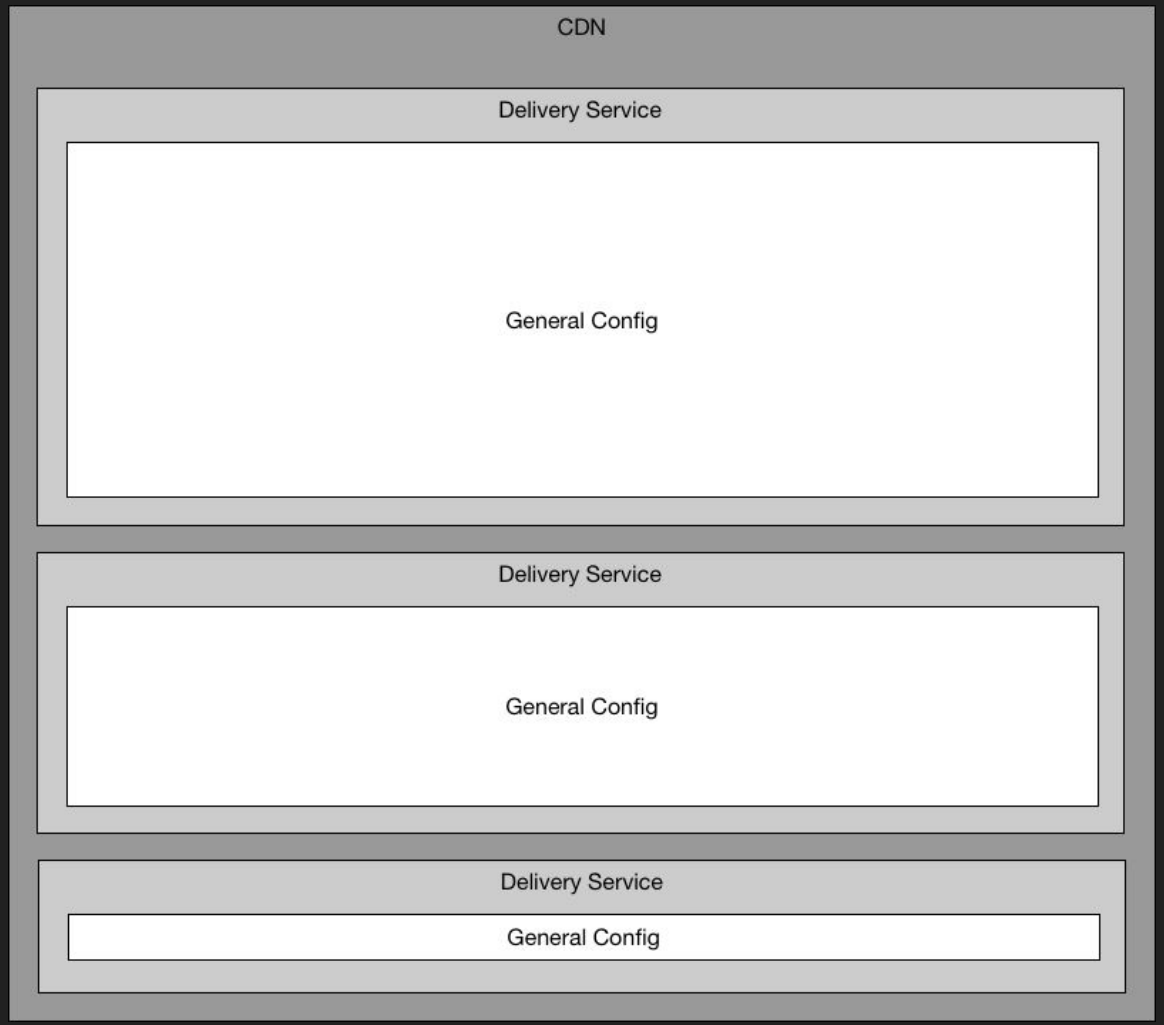
- Add/Remove/Modify headers
- Manipulate parent or origin URL / URI / Query Params / Scheme
- Manipulate parent or origin hostname
- Enabling debug logging
- Config overrideables per delivery service
- A lot of the existing functionality of:
  - Header_rewrite
  - Cache URL / Cache Key
  - Regex remap

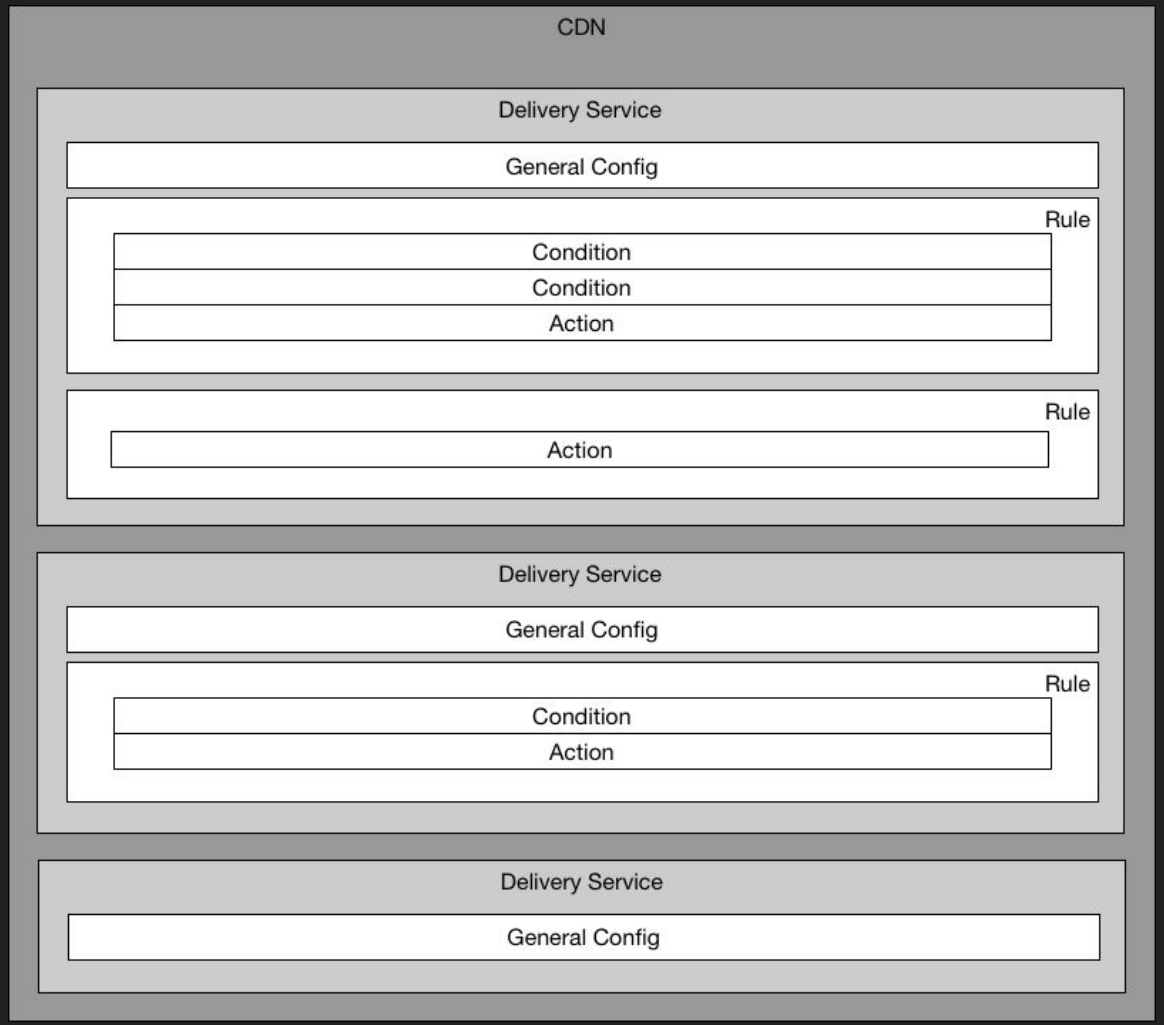
What do I want to do?

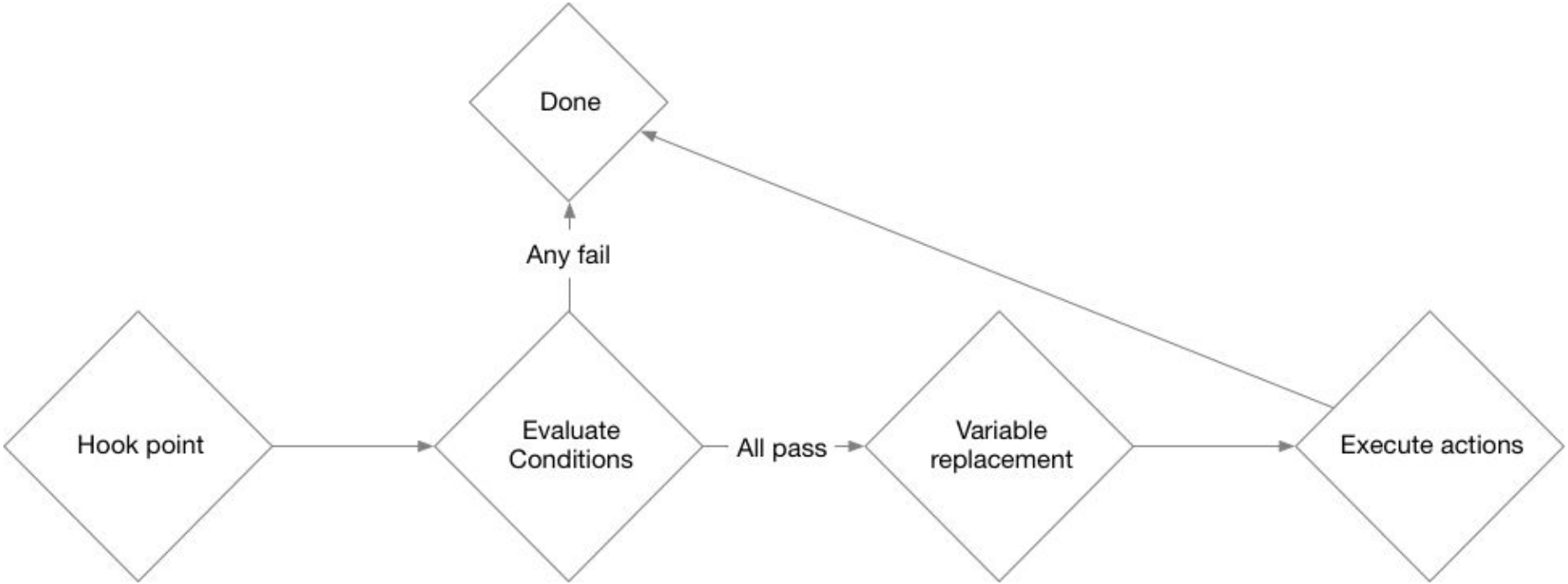
LUA all the things!

What do I want to do (cont’d)

- Create a structured way to describe programmatic functionality of a Delivery Service (rules)
- Create a Lua plugin for ATS that interprets & executes that structure, but also allows custom expansion
- Replace existing ATS specific database & API elements in Traffic Control
- Add a web UI for manipulating these rules easily to existing Traffic Portal
- Add an administrative UI for managing the defined components of a rule (conditions and actions) as well as adding custom functionality and defining who can access it


CDN
Delivery Service
General Config
Delivery Service
General Config
Delivery Service
General Config


CDN
Delivery Service
General Config
Rule
Condition
Condition
Action
Rule
Action
Delivery Service
General Config
Rule
Condition
Action
Delivery Service
General Config


Done
Any fail
Hook point
Evaluate Conditions
All pass
Variable replacement
Execute actions

Condition

Type	Target	Operator	Value	Arguments (type specific)
is_internal_request uri url header incoming_port scheme	client_request server_request client_response server_response	== != regex store starts_with ends_with contains	User defined	User defined

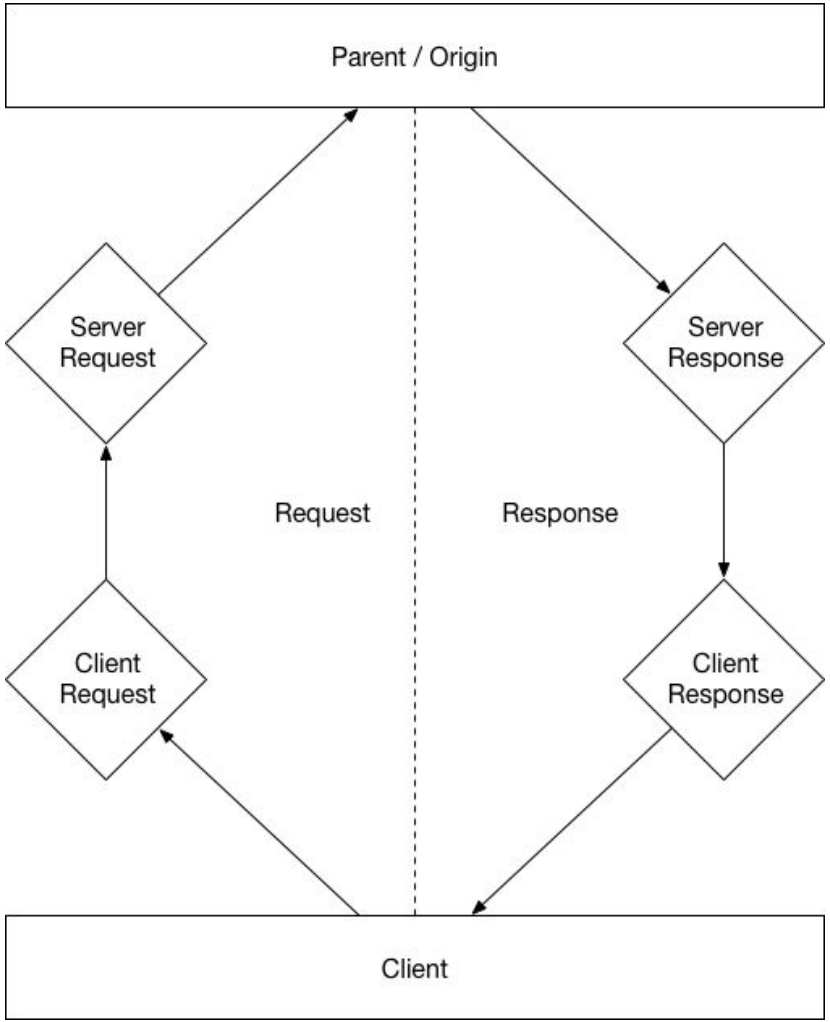
---

url	client_request	store	variable1	

Action

Type	Target	Arguments (type specific)
set_header remove_header set_response set_uri set_uri_args set_uri_params set_url_host set_url_scheme set_status custom_thing_here	client_request server_request client_response server_response	User defined

Hook points


Parent / Origin
Server Request
Server Response
Request
Response
Client Request
Client Response
Client

Why?

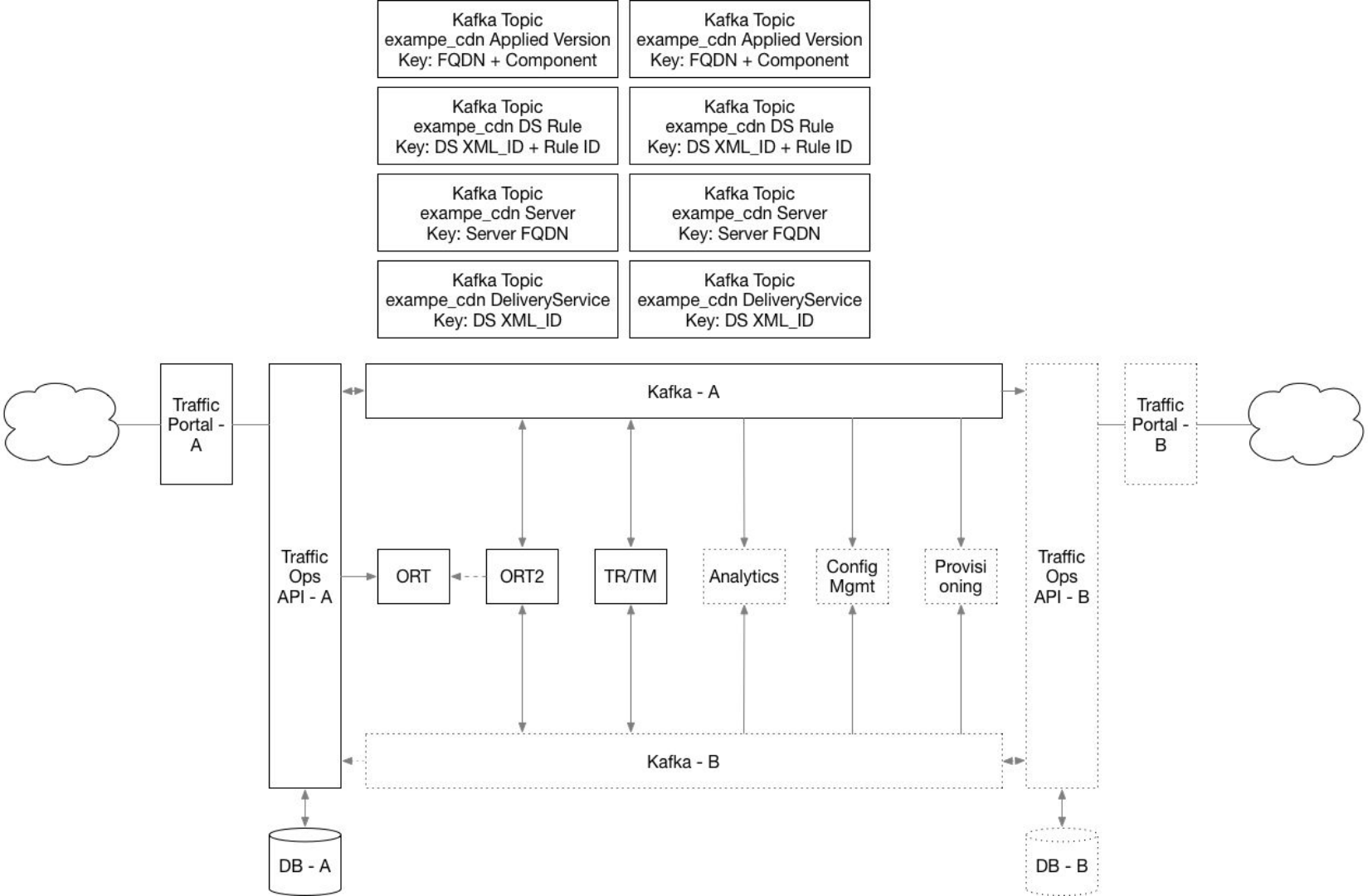
- Moves most ATS specific functionality out of the core of Traffic Control
  - Simpler to upgrade ATS
- Rules functionality can be re-implemented or translated to existing config language on other caches (nginx, varnish, homebrew_cache.exe)
- Allows self-service users (of a CDN) to access a lot more functionality
- Can build simple and “advanced” rules UIs for easier use
- Adding new custom (lua based) functionality is as simple as:
  - Write Lua code & check into CI/CD system (should run tests & deploy)
  - Add newly defined condition or action to admin UI, configure permission levels
  - Users with appropriate permissions can now configure that rule on their delivery services
  - Probably not quite that simple in reality
- Provides better isolation between delivery services

Examples from proof of concept

(This probably isn’t how the final result will look)

```
{
    name = 'test_set_header-client_request',
    hook_point = '',
    conditions = {
        {
            type = 'uri',
            target = 'client_request',
            args = {},
            operator = '==',
            value = '/test_set_header-client_request',
        },
    },
    actions = {
        {
            type = 'set_header',
            target = 'client_request',
            args = {name='X-test-header', value='This is only a test.'},
        },
    }
},
```

```
{
    name = 'test_set_response',
    hook_point = '',
    conditions =
    {
    },
    actions =
    {
       {
          type = 'set_response',
          target = '',
          args = {response_code=418, body="I'm a teapot"},
       },
    }
 },
```


Kafka Topic
exampe_cdn Applied Version
Key: FQDN + Component
Kafka Topic
exampe_cdn Applied Version
Key: FQDN + Component
Kafka Topic
exampe_cdn DS Rule
Key: DS XML_ID + Rule ID
Kafka Topic
exampe_cdn DS Rule
Key: DS XML_ID + Rule ID
Kafka Topic
exampe_cdn Server
Key: Server FQDN
Kafka Topic
exampe_cdn Server
Key: Server FQDN
Kafka Topic
exampe_cdn DeliveryService
Key: DS XML_ID
Kafka Topic
exampe_cdn DeliveryService
Key: DS XML_ID
Traffic Portal - A
Traffic Ops API - A
Kafka - A
Traffic Portal - B
ORT
ORT2
TR/TM
Analytics
Config Mgmt
Provisioning
Traffic Ops API - B
Kafka - B
DB - A
DB - B