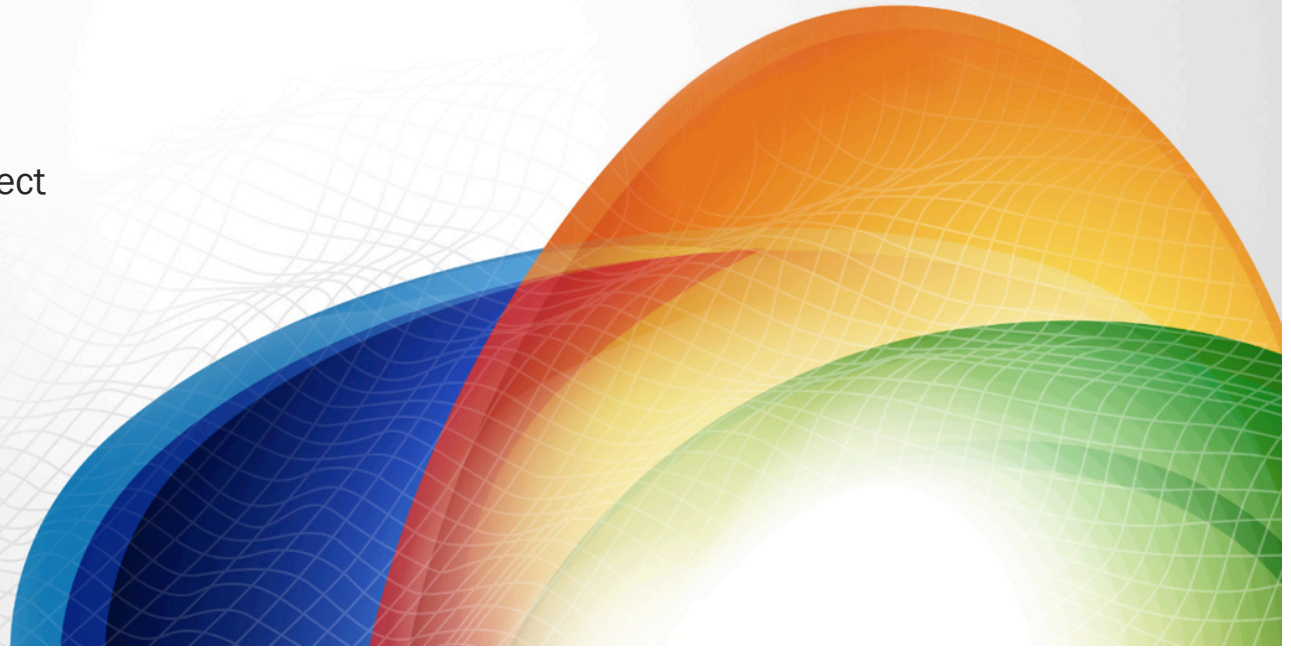
Intro to Automation & Orchestration

Hitesh Patel – Solution Architect



REST Intro

REST Intro

- Based on HTTP and JSON
- Uses HTTP verbs (GET, POST, PUT, PATCH, etc)
- Data is sent using the Javascript Object Notation format
  - {
    “attribute1”:”value1”,
    “attribute2”:[“array”,”of”,”values”],
    “attribute3”: [{ “nested1”:”value1”, “nested2”:”value2” }, {“nested3”:”value3”}]
    }

REST APIs and HTTP Verbs

- The following table summarizes the interpretation of HTTP methods for REST APIs.
- HTTP methods (verbs) are used to create, read, update, and delete (CRUD) resources.
- APIs must use HTTP verbs in a manner described in the table below.
- APIs may implement a subset as required.

URI	POST	GET	PUT	DELETE	PATCH
Collection	Create resources	Get representation of all resources in the collection.	Fully update all resources in a collection.	Delete all resources in a collection.	Partially update all resources in a collection.
Resource	Used for non-idempotent controller resources.	Get a resource's representation.	Fully update the resource if it exists.	Delete a resource.	Partially update a resource.

Response Codes

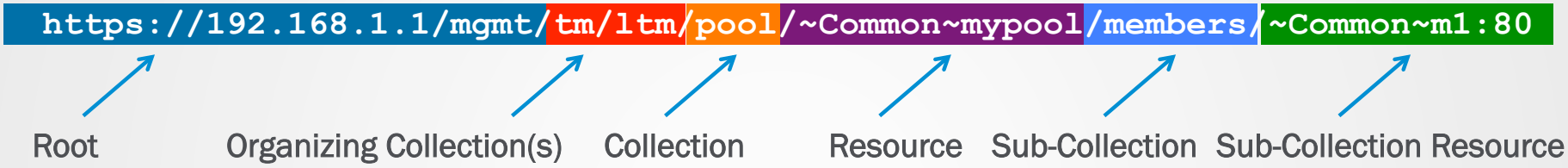
- APIs must make use of HTTP response codes where appropriate.
- The following table describes the required success response codes.

Response Code	Applicable Verbs	Notes
200 OK	• All	Return on most positive responses including DELETE.
201 Created	• POST	HTTP Location header contains link to newly created resource
202 Accepted	• POST • PUT • PATCH • DELETE	Return when a request will take a long time; server should return a Location header for client to get state updates.
404	• GET	The resource does not exist
500	• All	Check /var/log/restjavad.0.log

REST API Organization

Type	Description
Organizing Collection	Objects are not configurable, rather they contains other Collections or Resources
Collection	Objects are not configurable, however, a **Collection** contains Resources of the same type
Resource	A fully configurable object that supports create, update, refresh, delete, load, exists (CURDLE) operations.
Sub-Collection	A collection that is attached to a particular Resource. Must be accessed through the 'parent' Resource.
Sub-Collection Resource	Same as a Resource except it must be accessed via the Subcollection

Anatomy of a REST URI



NOTE: Resource names map '~' to '/' (e.g. ~Common~mypool is really /Common/mypool)

How the REST API is Implemented on TMOS

- REST API attributes are derived from TMSH schema
- You can ‘follow’ the API by mapping to tmsh commands in most cases
- Generally, if the attribute/option is available in TMSH it’s available in REST

REST Query Parameters

- ?$filter=<partition>
  - Allows an OData search string to filter objects
  - ?$filter eq MyPartition
- ?$select=attribute1,attribute2
  - Select specific attributes to return in the reponse
- ?expandSubcollections=true
  - Automatically expand any Sub-Collections in a resource
- See REST API guide for more

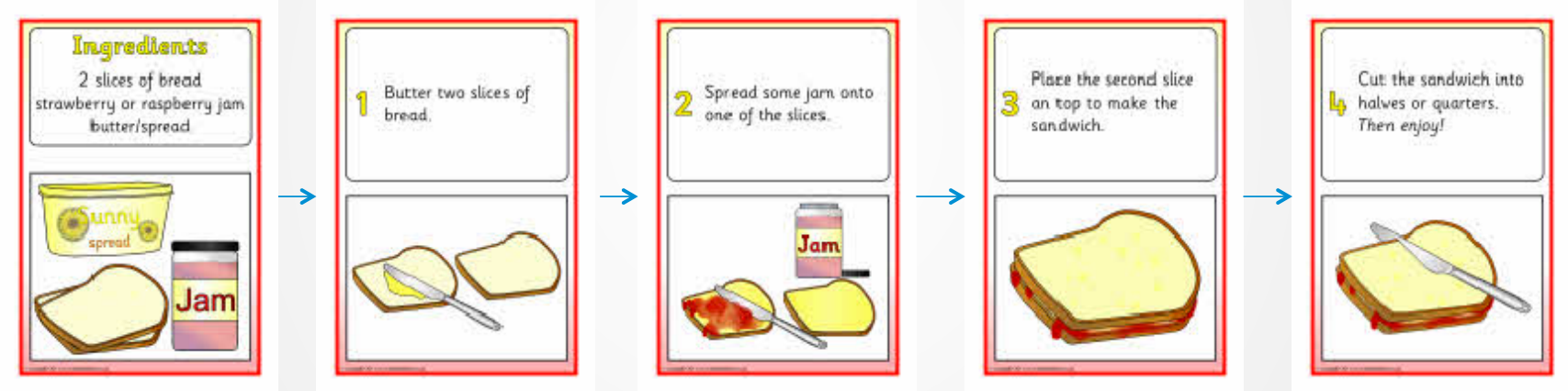
REST Chicken and the Egg

- Problem: I want to create something but can't figure out how REST wants it represented
- Solution: Add 'example' to the end of a REST collection URI and you'll get a template
  - E.g.: GET /mgmt/tm/ltm/pool/example

Automation & Orchestration Concepts

Concepts - Imperative

- Imperative – What we've done for years (scripting, iRules, etc.) Imperative methodology implies that you define the flow of an operation implicitly. It also implies that domain-specific knowledge is required to interact with the system.



- What domain-specific knowledge is required to make this sandwich?

Concepts - Declarative

- Declarative – What we're evolving to. Declarative methodology implies that you define the desired outcome and depend on underlying mechanisms to deliver that outcome. This methodology tries to reduce or eliminate the need for domain specific knowledge.

Concepts – Source of Truth(iness)

- Source of Truth is defined as a system or object that contains the authoritative representation of a service
- Changes for a service should propagate (push) from the source of truth to sub-ordinate systems
  - Out-of-band changes must be handled very carefully (or be totally avoided)
- To address real-world challenges I prefer to call this ‘Source-of-Truthiness’:
  - In an automation tool chain it’s possible to create layers of ‘truth’ as long sub-ordinate systems ‘filter’ the service definition upstream.
  - What the northbound system doesn’t know won’t hurt it
  - Changes should NEVER have to flow from sub-ordinate devices upstream!
- iApp strict updates are an example of strict Source-of-Truth enforcement
- iWorkflow service catalog is an example of a Source-of-Truthiness

Concepts – F5 Automation Anatomy

Concepts – Multi-Tenancy

- BIG-IP is Multi-Tenant inherently
  - Route-Domains
  - Partitions
  - What about route-domain 0?
- iWorkflow implements a Provider/Tenant model for multi-tenancy
- All automation code should be multi-tenant aware by nature
  - It's not that hard… always use a partition for object names
  - Always use a route domain on all L3 addresses (including 0)

Concepts - RBAC

- Declarative Methodology – RBAC is not necessarily required if a strict service abstraction model is implemented. The tenant service interface should inherently implement security controls as part of the service definition

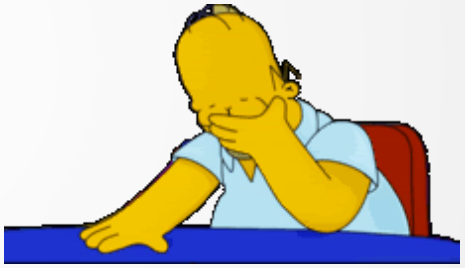
- Imperative Methodology – RBAC is necessary in most environments, however, it should be implemented upstream from BIG-IP (i.e. iWorkflow) and an API aggregator/proxy

Lessons from the road...

JSON Formatting

- JSON is a strictly defined syntax. As a result it is very unforgiving of syntactical errors.
- When defining a JSON object the last element in the object should NOT have a comma:

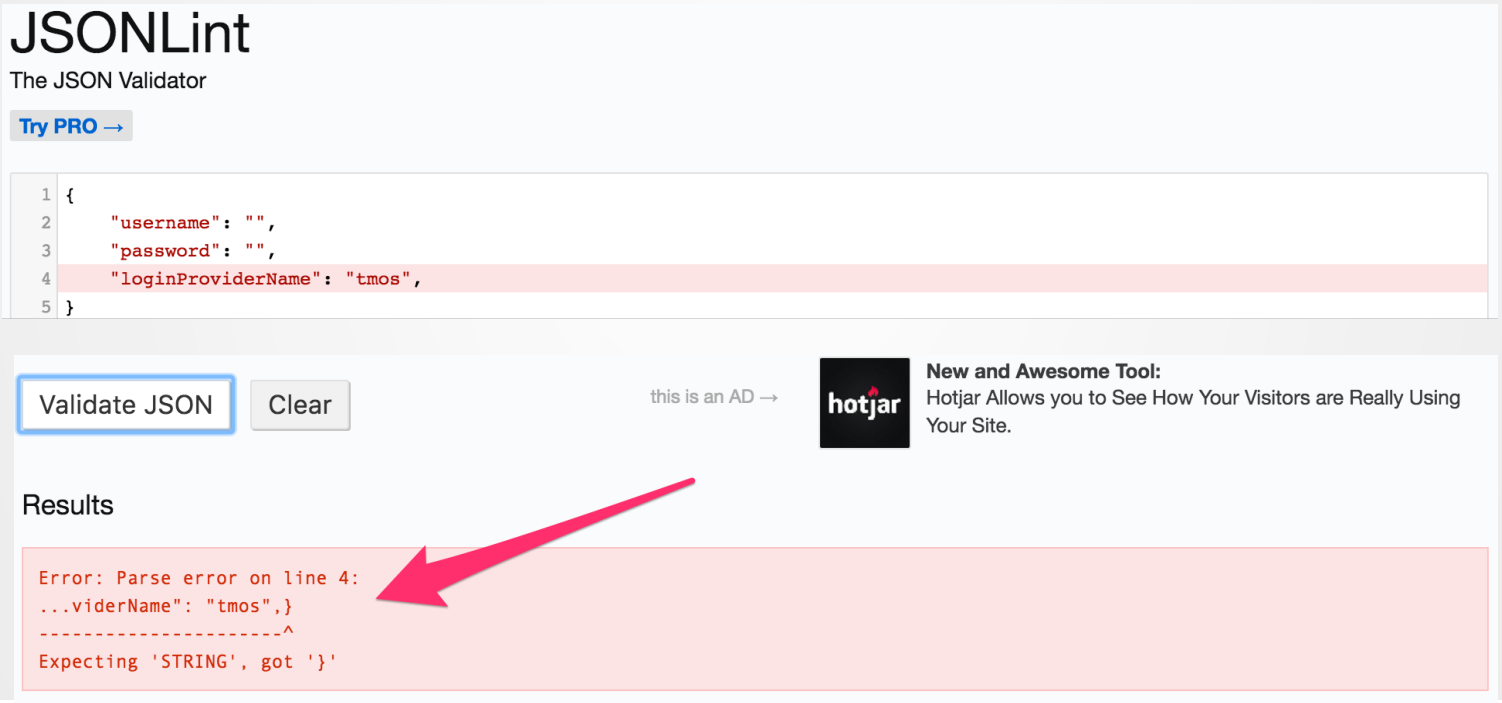
```
1  {
2      "username":"",
3      "password":"",
4      "loginProviderName":"tmos",
5  }
```



```
1  {
2    "code": 400,
3    "message": "com.google.gson.stream.MalformedJsonException: Expected name at line 5 column 2 path $.loginProviderName",
4    "originalRequestBody": "{\n    \"username\":\"\",\n    \"password\":\"\",\n    \"loginProviderName\":\"tmos\",\n}",
5    "referer": "192.168.255.1",
6    "restOperationId": 297725,
7    "kind": ":resterrorresponse"
8  }
```

JSON Formatting – When you get stuck

- http//jsonlint.com -- Verify Syntax, Pretty Print Ugly JSON Responses



Useful Logs

- tail –f /var/log/restjavad.0.log | python –m json.tool (BIG-IP & iWorkflow)
- /var/log/ltm
- /var/log/audit

L4-7 Service Delivery with SDN/Cloud Infrastructure

Cisco/VMware/OpenStack Provides:

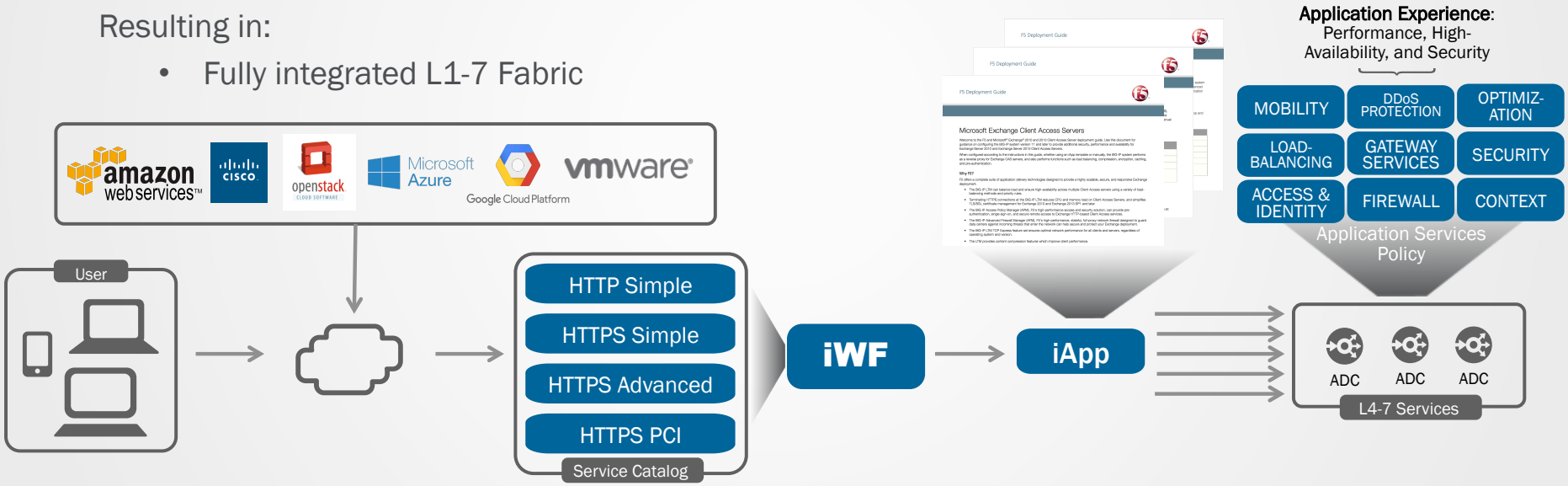
- L1-3 Network Fabric

F5 Provides:

- L4-7 **Services** Fabric

Resulting in:

- Fully integrated L1-7 Fabric



Solutions for an application world.