RED HAT 3Scale

API Management Workshop

March, 2021

Tommer Amber, Senior Cloud Security Consultant
Red Hat Professional Services, Israel

Agenda

- APIs - Background & Types of Web APIs
- API Management Lifecycle
- 3Scale - Overview
- Day Two Deployment Options
- API Security
- Automation

BACKGROUND

WHAT IS AN API?

Application
Programming
Interface

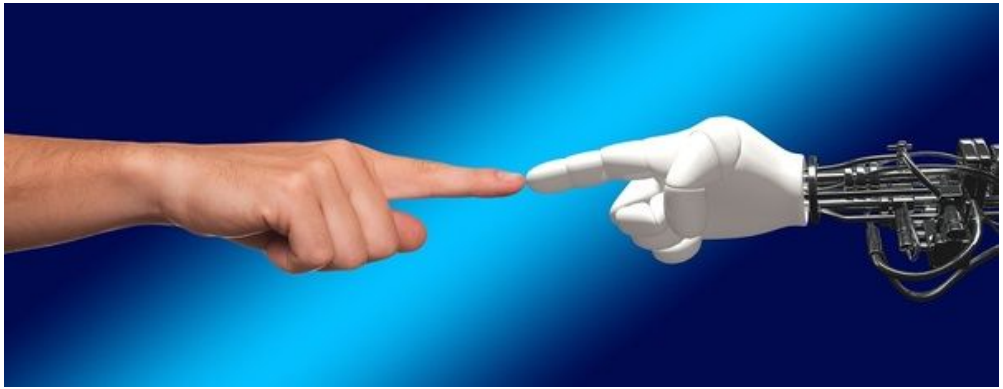
WHAT IS AN API?

"Software component or a system, externally exposed, that defines how other components or systems can use it"

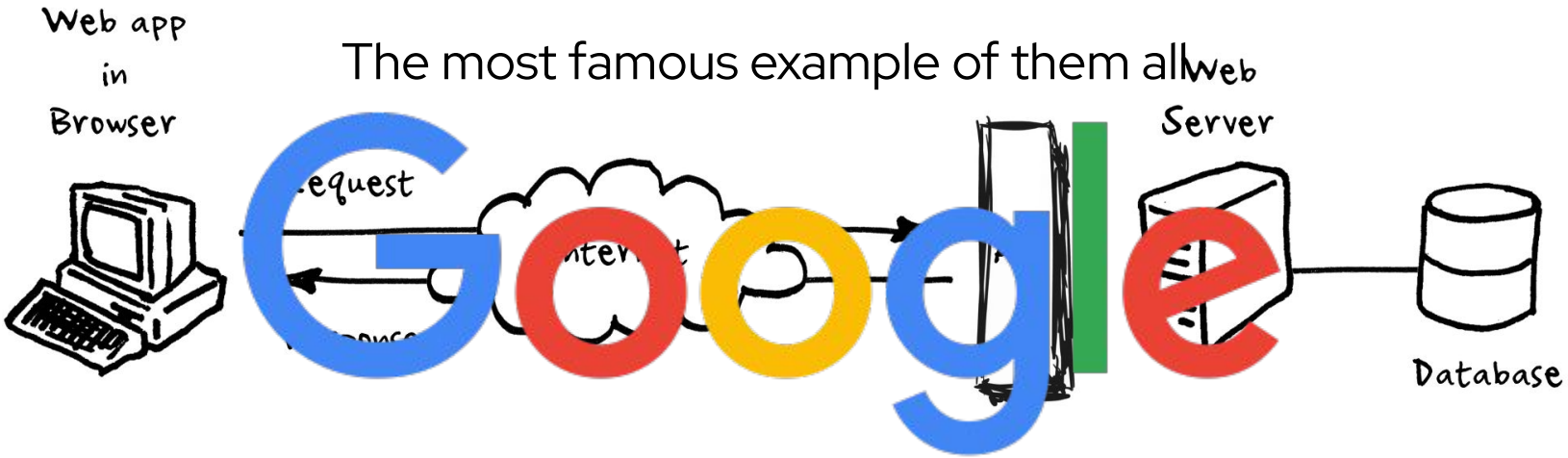
It defines the kinds of calls or requests that can be made, how to make them, the data formats that should be used etc.

WHAT IS AN API?

Machine's API == "I understand requests in the following formats only"



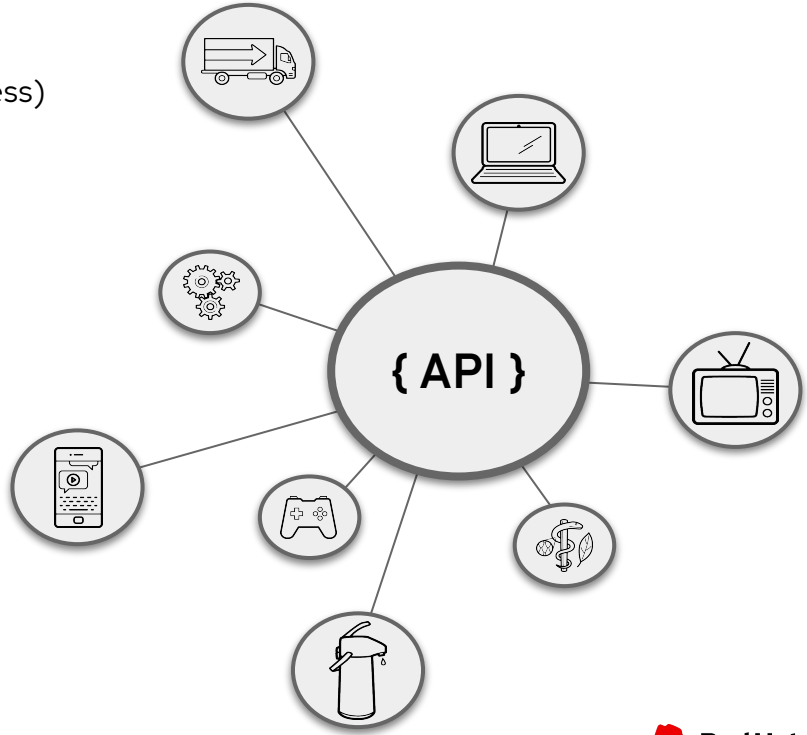
API-exposure architecture - Simplified



Red Hat

APIs Are Everywhere!

- Mission critical to every business (or soon every business)
- Customers and partners depend on it
- Fundamental to company agility and competitiveness
  - (It is basically a standard nowadays)
- Web and mobile depend on APIs
  - And not only them

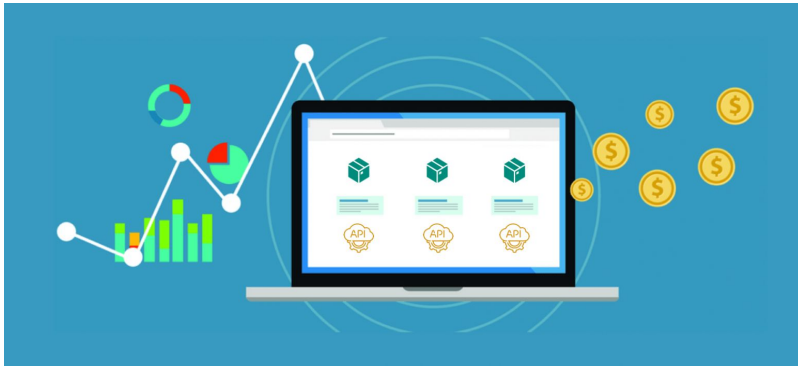


Red Hat

API Economy

API As A Service

- Supplier & Consumer
  - Customers (Apps functionality)
  - Partners (Develop new products)
- Bridge between IT vs. Business
- Forms of consumption
  - C2B - Customer-to-Business (Web, Mobile, Apps)
  - B2B - Business-to-Business (Internally & Externally)
- *API Access Monetization Question - How?*

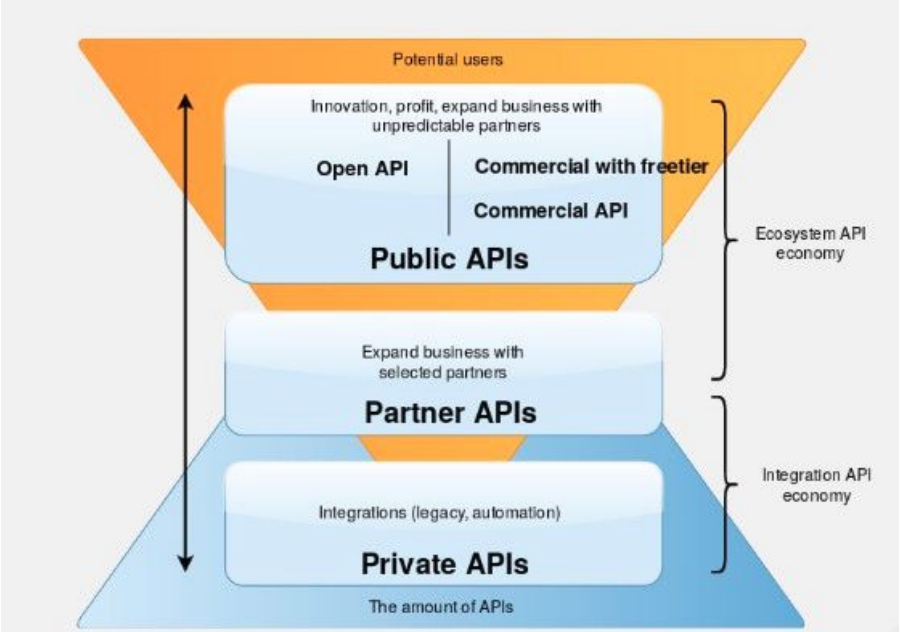


APIs – in depth

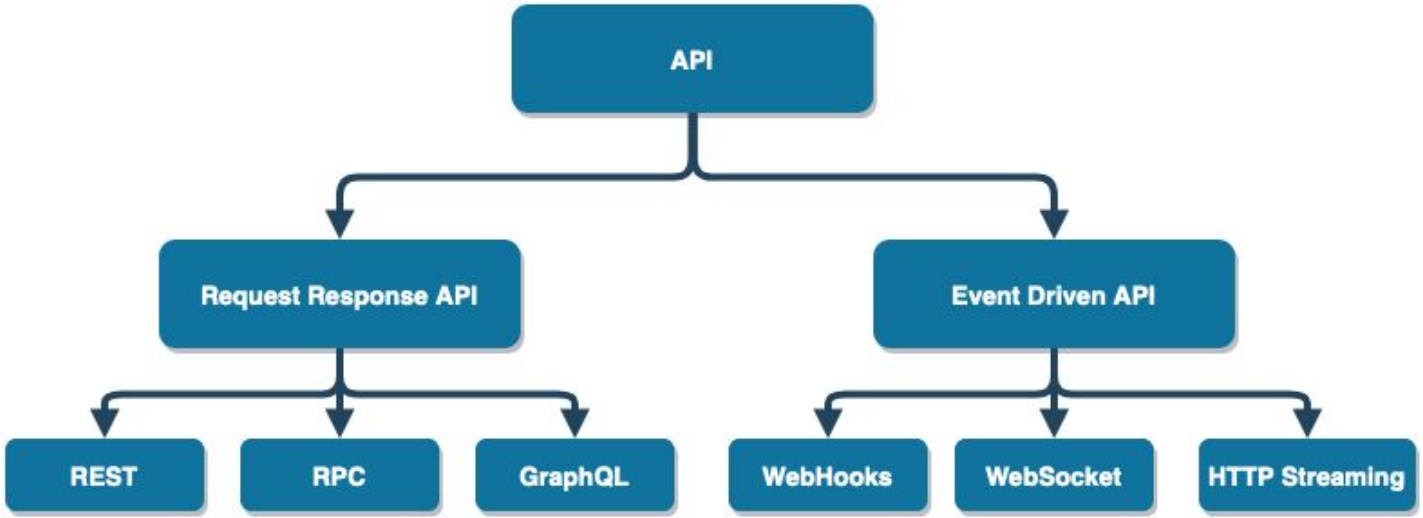
https://rapidapi.com/blog/types-of-apis/

API Categories

- Open APIs
  - I.e. GoogleMaps
- Private APIs
  - Intermediate consumption only
  - Internal DB / Organizational Shared Resource
- Partner APIs
  - Github
  - Cross organizational products' development (e.g. SaaS)



Red Hat



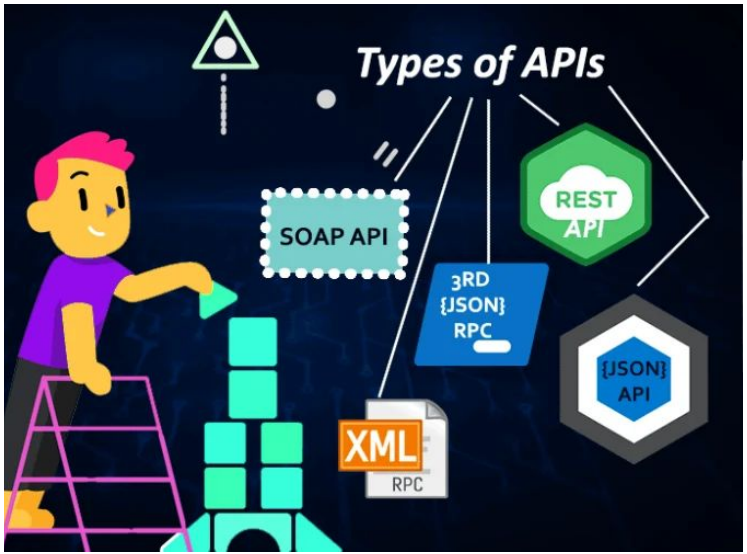
https://laptrinhx.com/comparing-api-architectural-styles-soap-vs-rest-vs-graphql-vs-rpc-1726803065/


API TIMELINE
1991
CORBA
1993
RDA
1998
XML-RPC
1999
SOAP
2000
REST
2005
JSON-RPC
2007
ODATA
2015
GRAPHQL
2016
gRPC

Red Hat

Web (HTTP/S) API Types
(Most Popular)

- RESTful API
  - CURD oriented
  - E.g. Webhook
- SOAP
  - XML based
- RPC
  - Application Specific
  - XML vs. JSON
- gRPC
  - Declarative Docs
- GraphQL
  - Robust use cases
  - one to many query association



Red Hat

RESTful API

HTTP Verb	Operation	URL Example
GET	Read	GET /api/project/1
POST	Create	POST /api/project
DELETE	Delete	DELETE /api/project/1
PUT	Update	PUT /api/project/1

Small Example

RESTful API

- REST = **Re**presentational **S**tate **T**ransfer
  - Interactive applications that use Web services

    OR
  - Non-Interactive automation mechanisms

- PreDetermined calls (GET, POST, UPDATE, DELETE)
  - payloads lighter

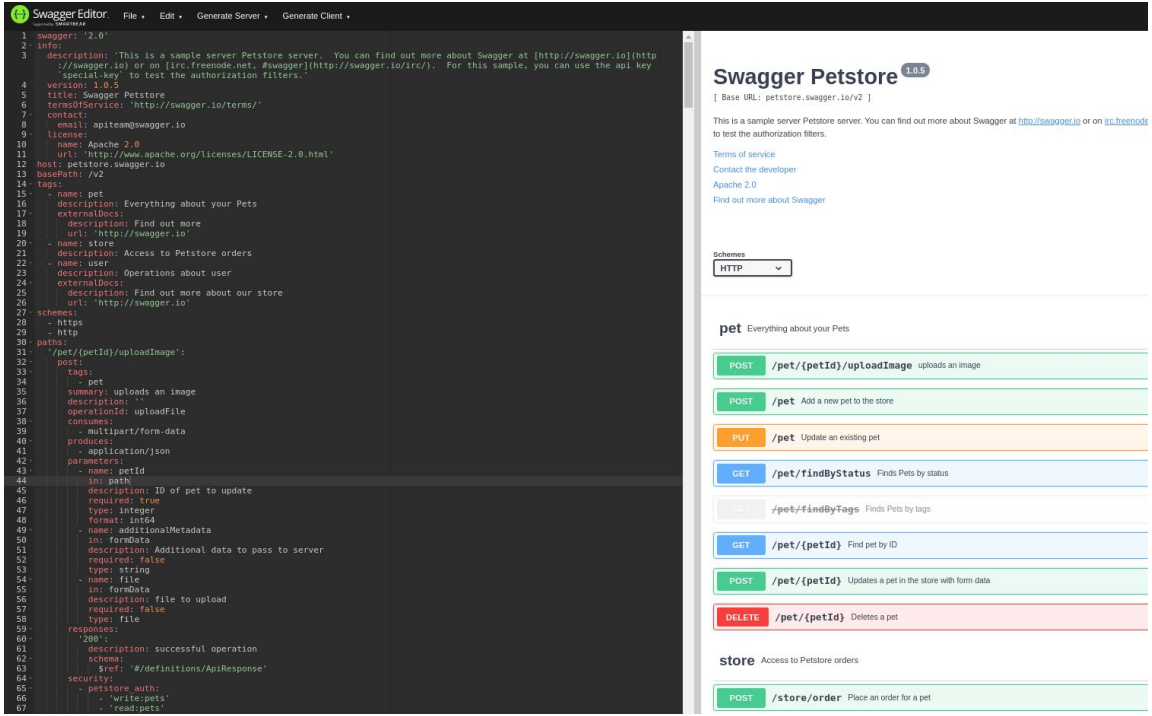
- REST APIs uses different data formats
  - plain text (raw) , HTML, XML, and JSON sending payload

- Stateless

Swagger Editor

API Specification Visualization

- Swagger editor
- File ⇒ import url ⇒ https://petstore.swagger.io/v2/swagger.json

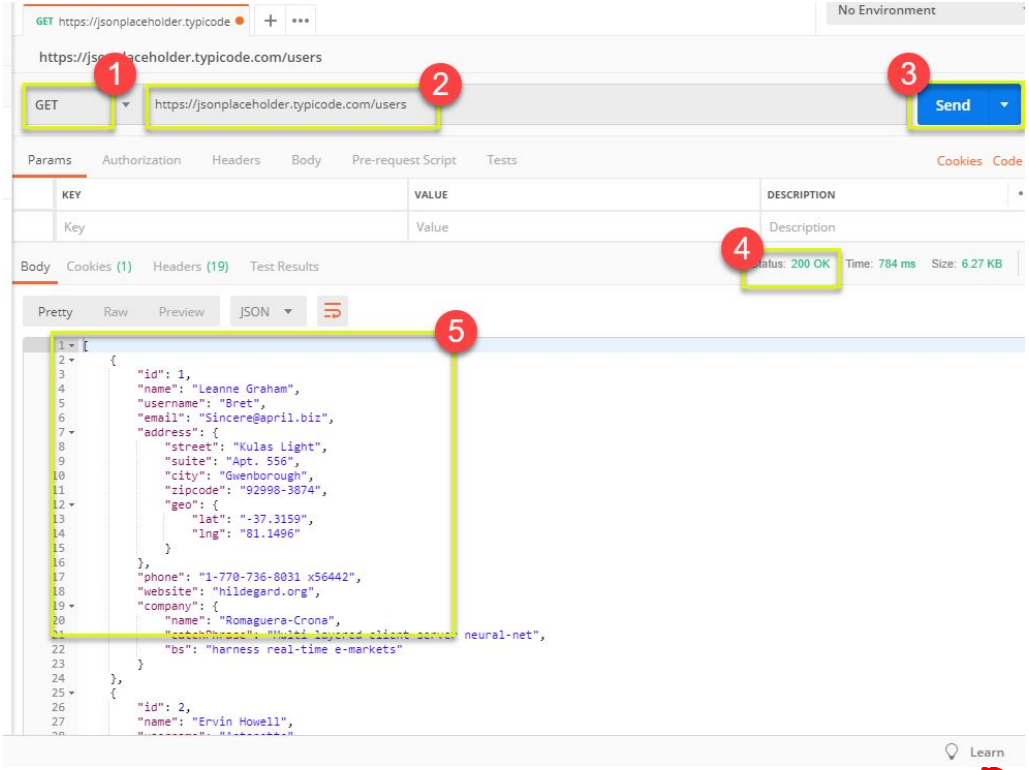


Red Hat

Postman

Testing Tool

- https://www.postman.com/



Red Hat

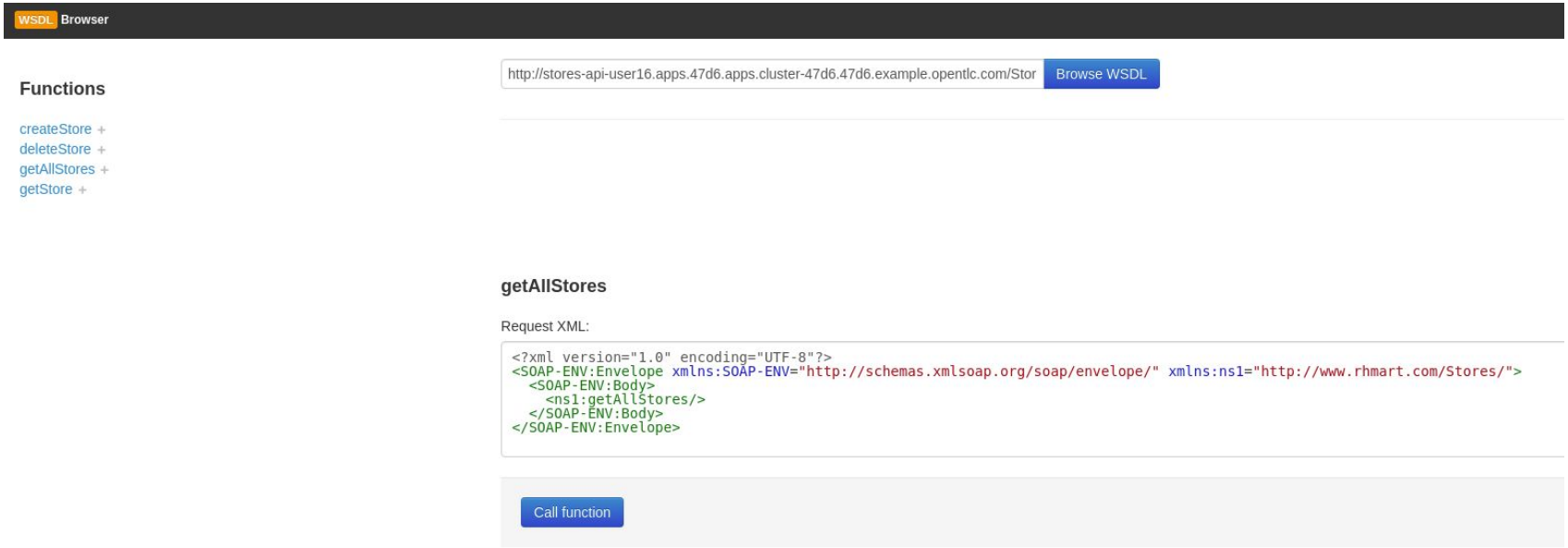
SOAP – Simple Object Access Protocol

- An established web API protocol
  - Can also be implemented for TCP, SMTP etc.
- (HTTP) Header, XML Body
  - Compatible with schema
- WSDL – Web Services Definition Language
  - Dictionary of APIs accessible
  - Swagger Equivalent

SOAP Code Example

```
@WebService(endpointInterface="com.redhat.service.Stores")
public class StoresWS implements Stores {

        @Inject
        StoreDao storeDAO;

        @Override
        public String createStore(Store store) {
                store = new Store(store.getStoreName(),store.getStoreLat(),store.getStoreLong());
                storeDAO.createStore(store);
                return "Store ID:" + store.getStoreID() + " CREATED";
        }

        @Override
        public String deleteStore(int storeID) {
                storeDAO.deleteStore(storeID);
                return "Store ID: " + storeID + " DELETED";
        }

        @Override
        public Store getStore(int storeID) {
                return storeDAO.getStoreById(storeID);
        }

        @Override
        public StoresType getAllStores() {
                StoresType st = new StoresType();
                st.store = storeDAO.getAll();
                return st;
        }

}
```

WSDL Browser



Red Hat

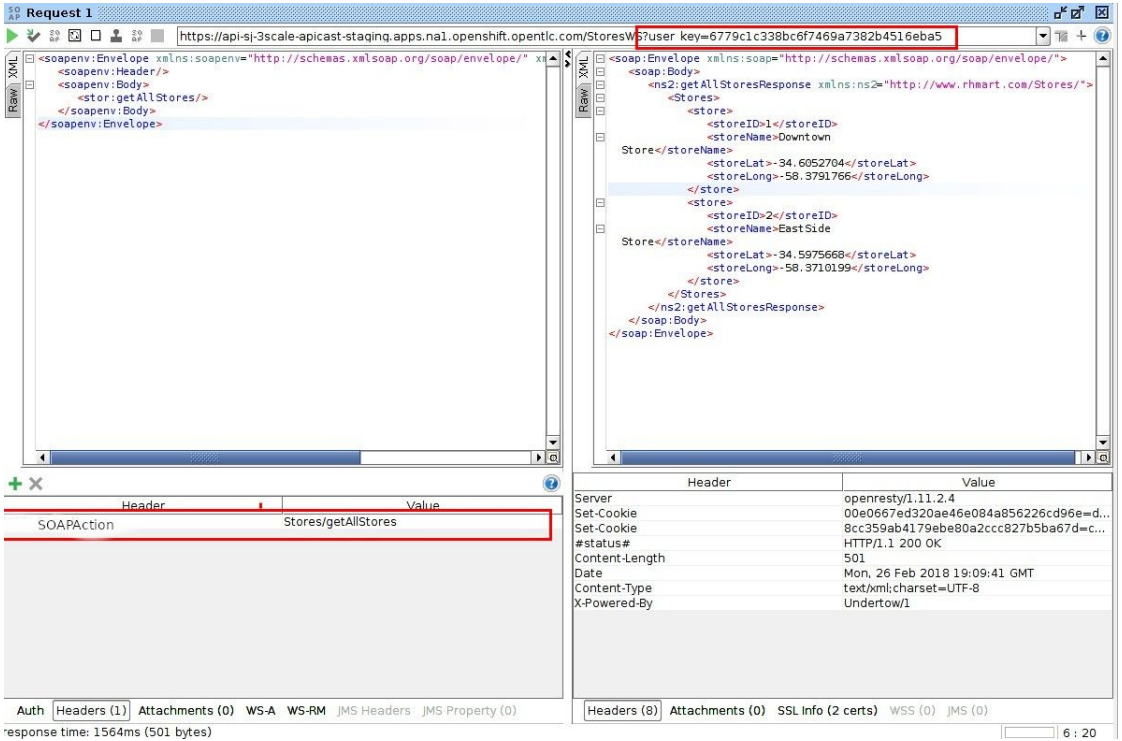
getAllStores

Result:

```
<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <ns2:getAllStoresResponse
            xmlns:ns2="http://www.rhmart.com/Stores/">
            <Stores>
                <store>
                    <storeID>1</storeID>
                    <storeName>Downtown
  Store</storeName>
                    <storeLat>-34.6052704</storeLat>
                    <storeLong>-58.3791766</storeLong>
                </store>
                <store>
                    <storeID>2</storeID>
                    <storeName>EastSide
  Store</storeName>
                    <storeLat>-34.5975668</storeLat>
                    <storeLong>-58.3710199</storeLong>
                </store>
            </Stores>
        </ns2:getAllStoresResponse>
    </soap:Body>
</soap:Envelope>
```

SOAP UI

- Send SOAP requests
  - left - request
  - right - response
- Better visibility of XML
- Add header parameters



Red Hat

RPC

Remote Procedure Call

- Specifications
  1. Internal API's parameters awareness
  2. Query info transferred in the request's data
  3. Not relying solely on verbs (GET/POST, etc.)

- Biggest Con of RPC

  The sender need to be aware of the exact "serialization" of the payload and every information that relates to the specific method

One typo and you're out

RPC vs. REST

URL Query:

```
// Request
curl -X PUT https://localhost:8080/api/users?name=Piers
```

URL path:

```
// Request
curl -X PUT https://localhost:8080/api/users/Stephen
```

HTTP body:

```
// Request
curl -X PUT https://localhost:8080/api/users { "name": "Josh" }
```

This can be a bit overwhelming, and sometimes we're not sure which one to use; but JSON-RPC is here to help us sort this out since it offers a simpler way:

```
// Request
curl -X POST –data '[{
  "jsonrpc": "2.0",
  "method": "addUser",
  "params": { "name": "Angad" },
  "id": 1
}]' https://localhost:8080/api
```

RPC Batch Actions

- RPC Pro
  1. Batch actions
  2. Less packets => Lower overhead => Increase performance

Batched action

```
// Request
curl -X POST–data '[{
  "jsonrpc": "2.0",
  “method”: “addUser”,
  “params”: { “name”: “Zalan” },
  “id”: 1
}, {
  "jsonrpc": "2.0",
  “method”: “addUser”,
  “params”: { “name”: “Dan” },
  “id”: 1
}, {
  "jsonrpc": "2.0",
  “method”: “addUser”,
  “params”: { “name”: “Edgars” }
}]’ https://localhost:8080/api
```

gRPC

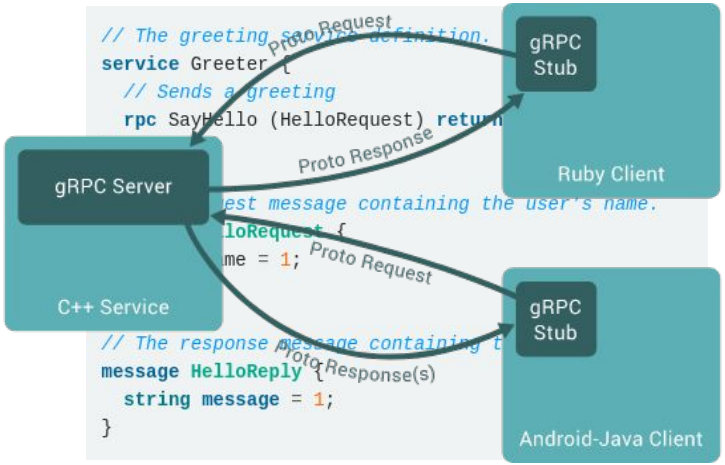
- Definitions all over!

  Solves RPC biggest con

- 3Scale 2.8+ Supports gRPC

- Scalable, Fast, Works across multiple languages and platforms

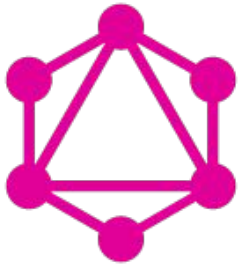
- https://grpc.io/docs/what-is-grpc/



Red Hat

GraphQL

- The best explanation is an example - https://graphql.org/
- One - to - Many API Query tool
- Graphql vs. REST - Medium Blog
- **Currently not supported by 3Scale (Future Feature)**



Get Your Hands Dirty

Requirements:

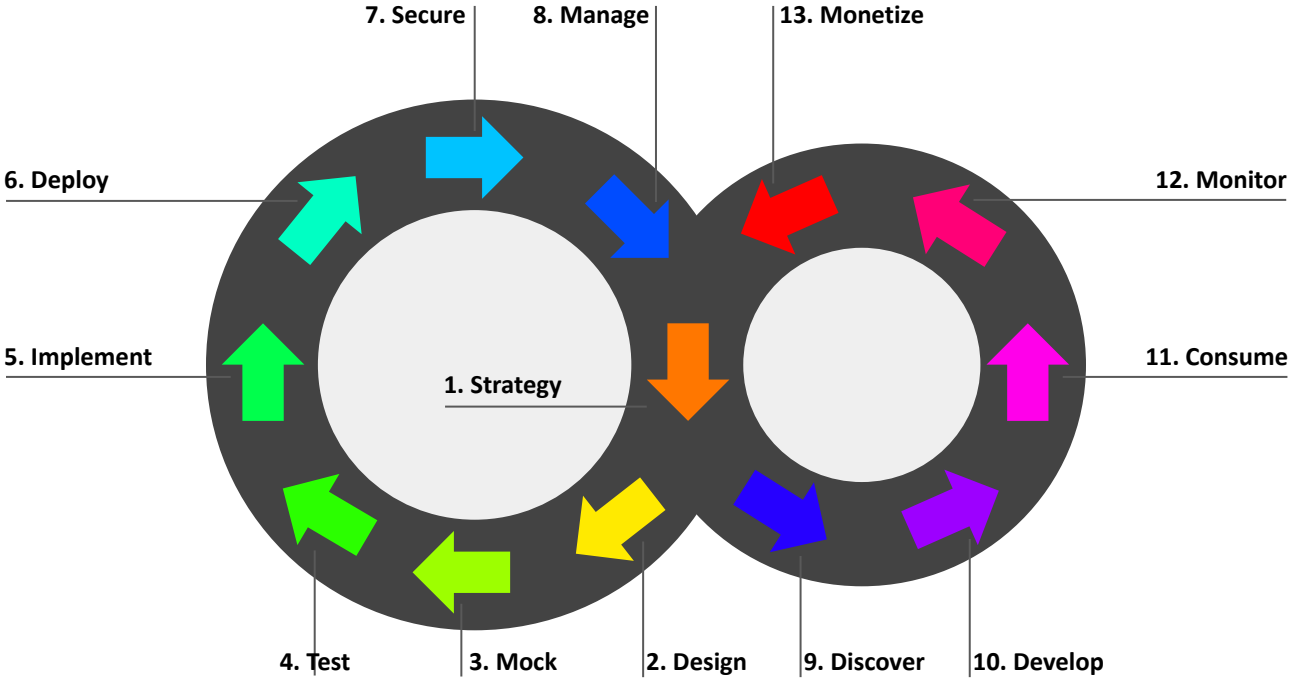
OCP client (oc), python, jq

Optionally use an IDE (such as VS Code) to view files

Exercise 1

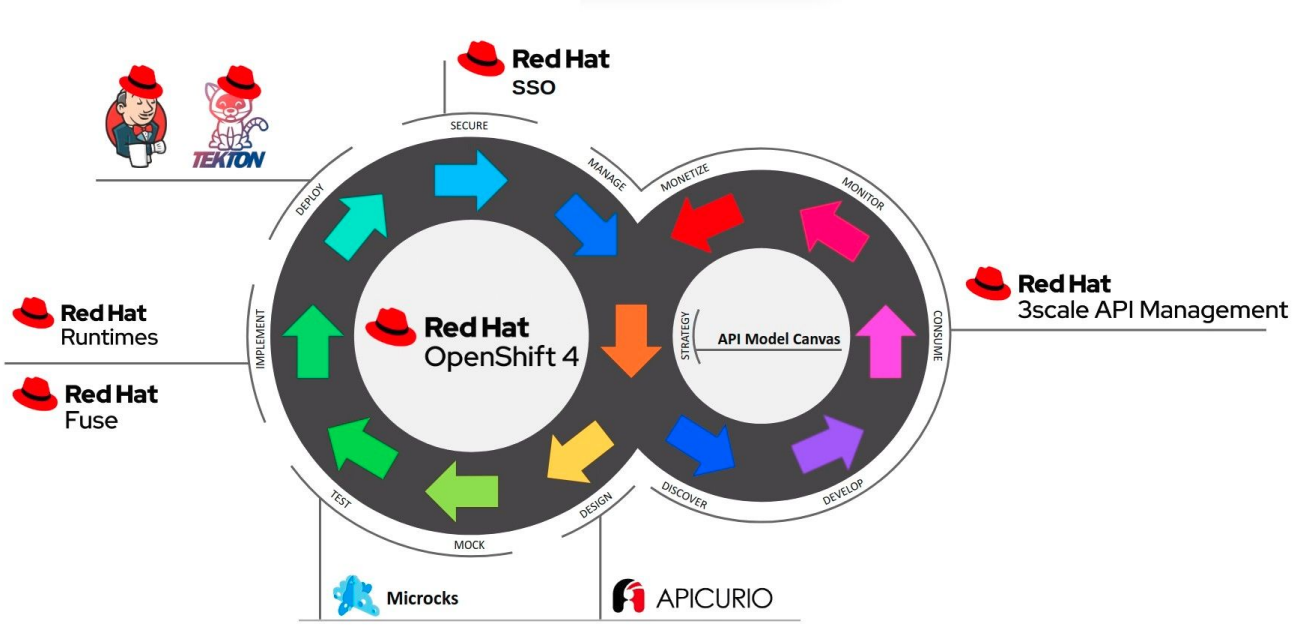
RED HAT AGILE INTEGRATION

FULL API LIFECYCLE MANAGEMENT



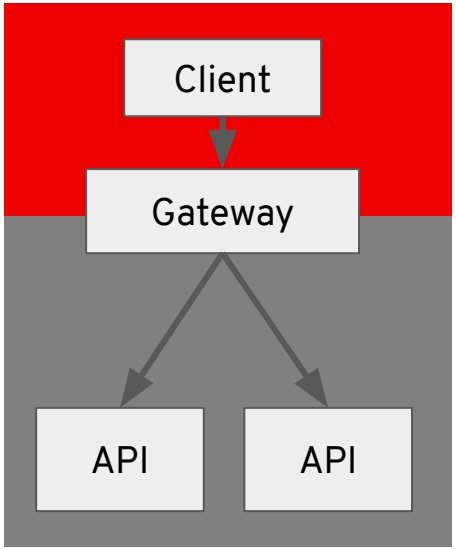
Red Hat

API LIFECYCLE AND STACK – OCP 4.x



Red Hat

Most Common Implementation for API GW



- APIs as a digital access point for your business
- “North-South” service architecture pattern
- Requires traditional API management capabilities
- APIs As A Product

Red Hat 3Scale Overview
&
Basic Architecture

RED HAT 3SCALE API MANAGEMENT

100% Open Source

Modular

No single point of failure

Quick time-to-value

Highly scalable

Support Multi Tenancy

Support Remote-Implementation Options

My Medium Articles

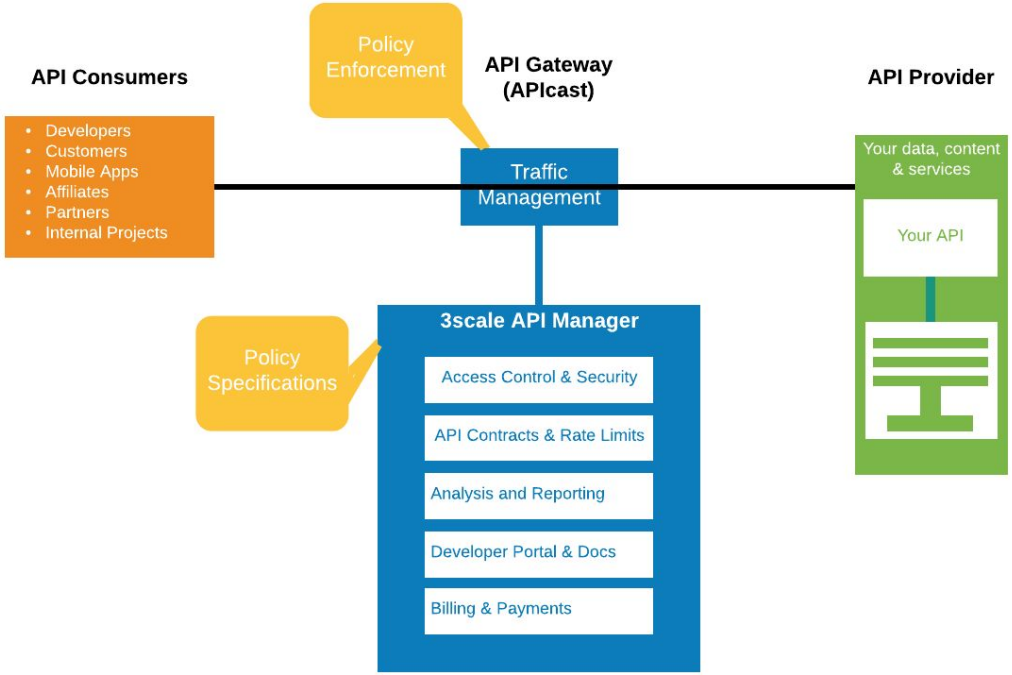
API Management Overview

Installation Guide

Red Hat

API Management Main Disciplines

Terminology

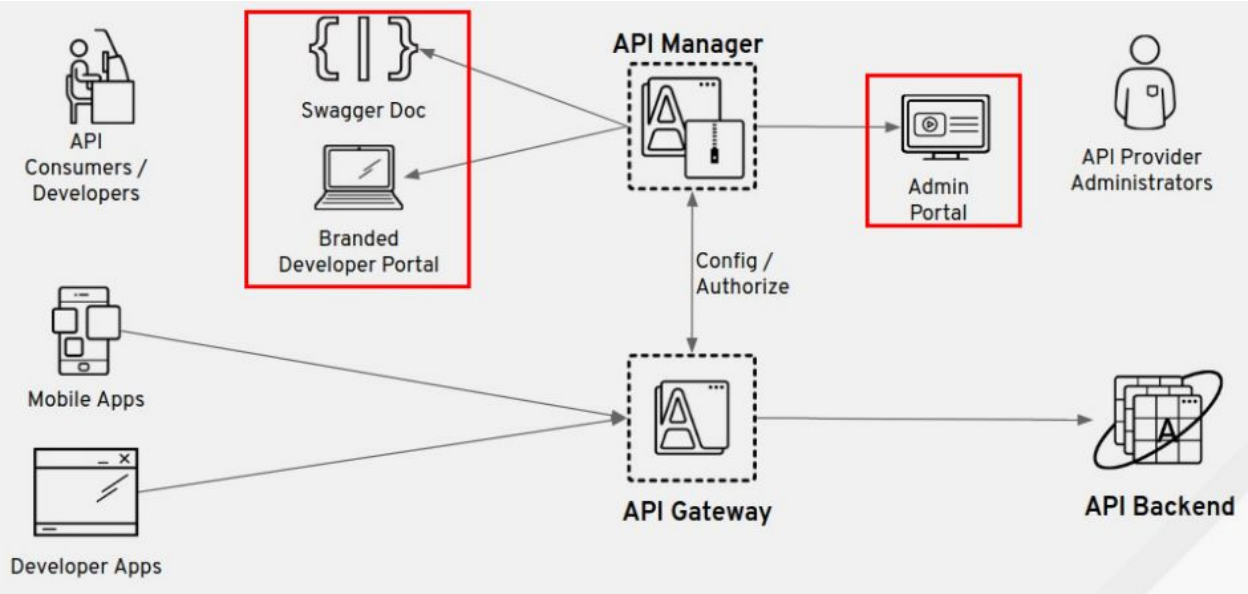


Red Hat

Multiple Usage Types

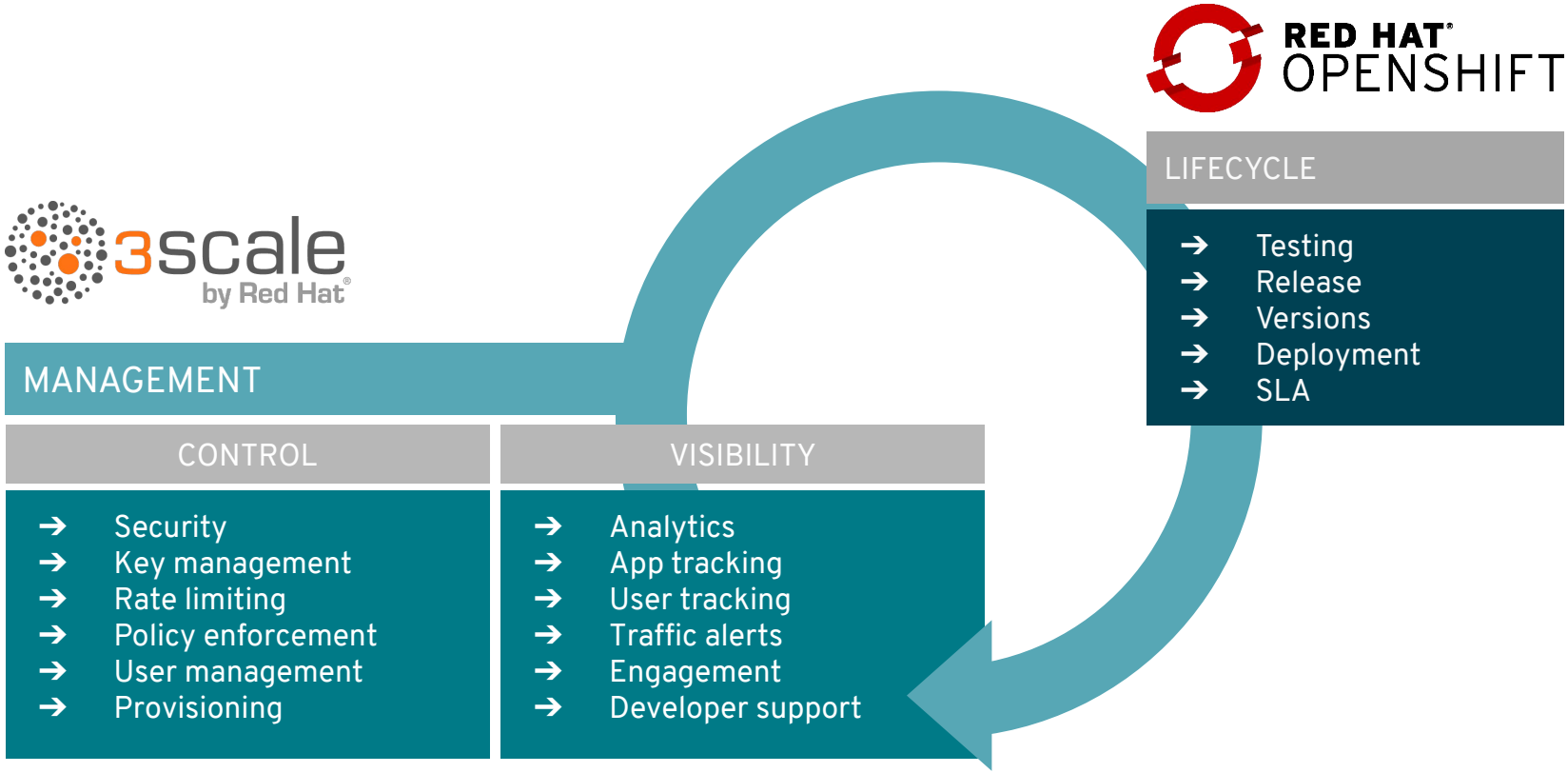
- Internal Developers
  - Subscription Plans
- 3Scale Admins
  - Product Management, Deployment & Scaling
- App Client - Mobile/Browser
  - Account Plans
- External Companies - Developers
  - Customizable Developer Portal
  - Monetization
  - Also relevant for integration between internal services (East-West)

Complete Diagram



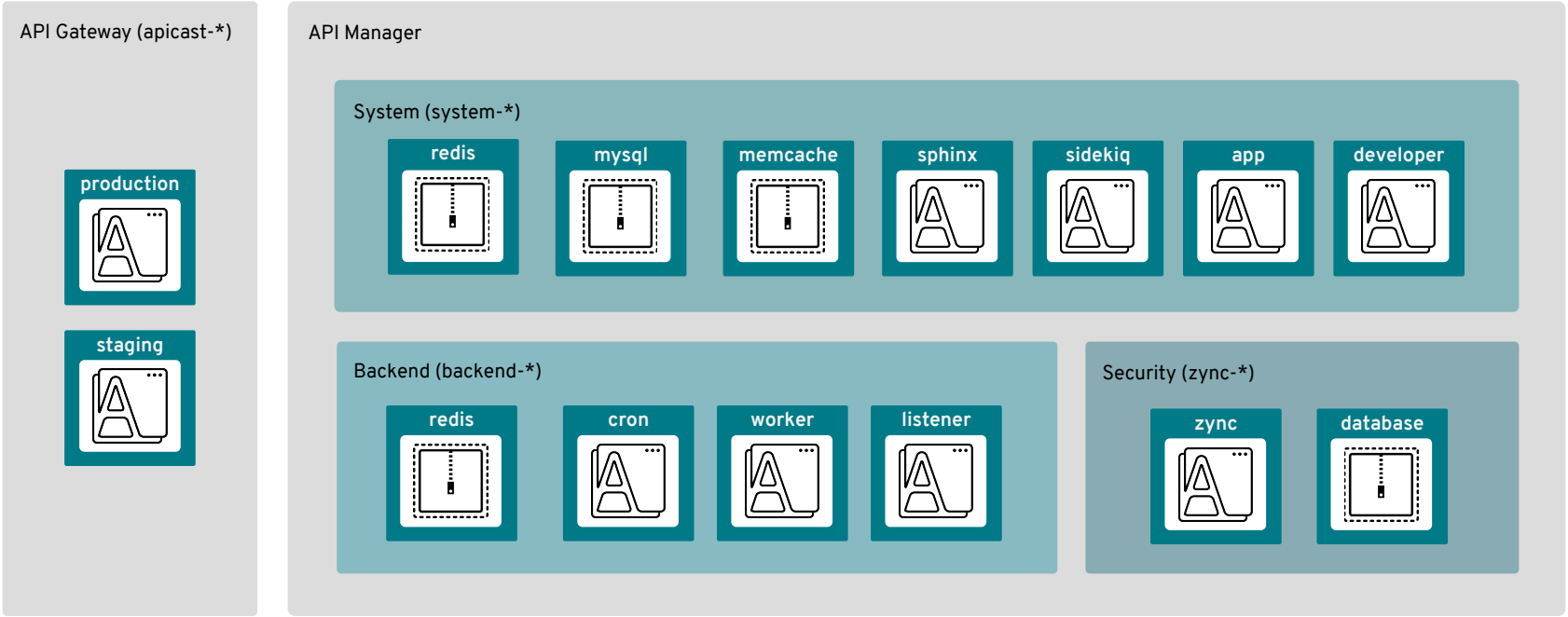
Red Hat

API MANAGEMENT vs. LIFECYCLE



Red Hat

3Scale Components



[Supported Configuration]

Red Hat

RED HAT 3SCALE API MANAGEMENT

- Storage Requirements
  - 3 or 4 Persistent Volumes
  - 3 RWO PV for Redis and MySQL/Postgres persistence (always needed)
  - 1 RWX PV (amp.yml template) **MUST** be group writable: chmod g+w

    or
  - AWS S3 Bucket for CMS assets (amp-s3.yml template)
- 10GB storage should be plenty to get started, 30GB for Production
- PVs should be writeable by OpenShift (Permissions)
  - Owned by root group
  - Proper selinux permissions: chcon -Rt svirt_sandbox_file_t
- Supported Configuration - Update Periodically

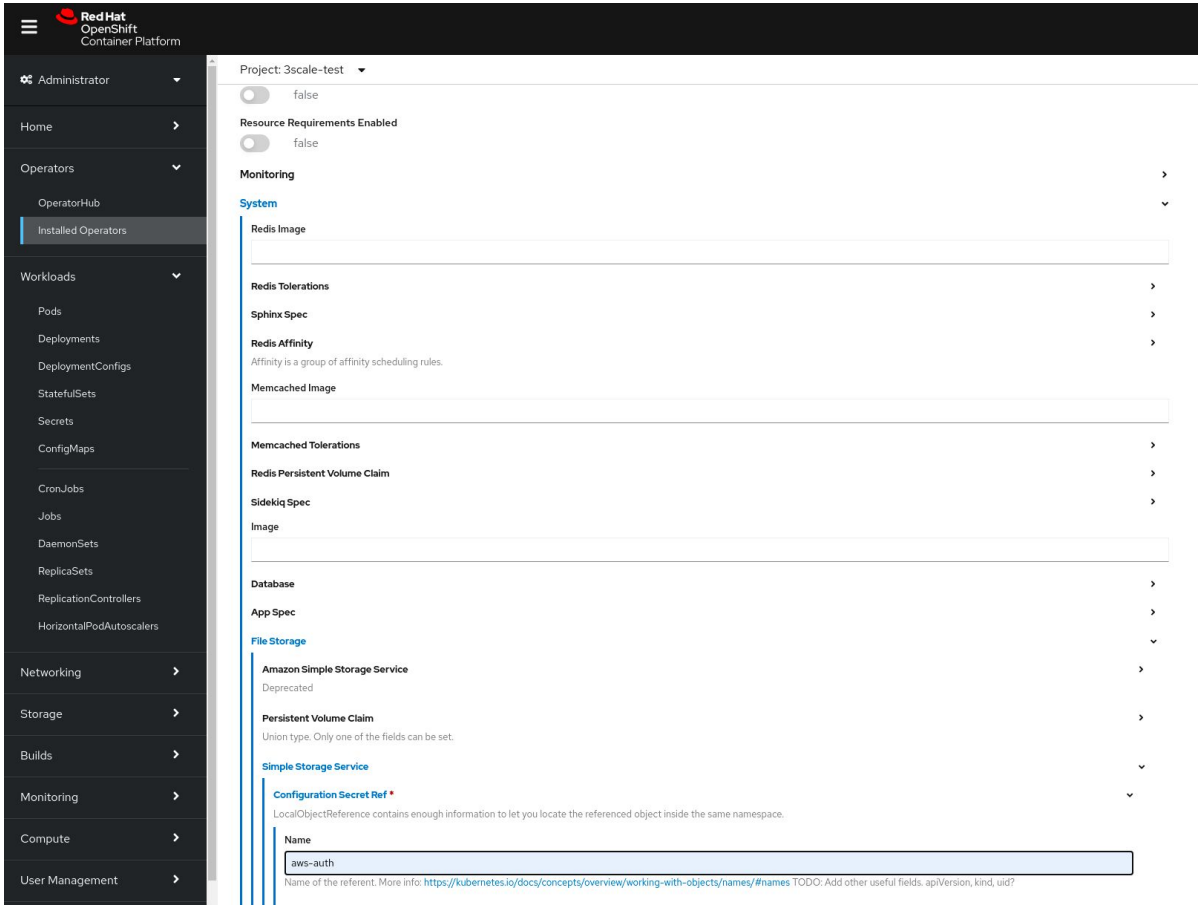
Deploy 3Scale with S3 using Operator

- Link to Documentation

- The RHPDS lab environment runs on AWS managed, which means that it has credentials for the aws bucket for the internal registry, and I used them to deploy 3scale on the same storage
  - oc describe configs.imageregistry.operator.openshift.io cluster | grep “Bucket:”
  - oc describe configs.imageregistry.operator.openshift.io cluster | grep "Region:"
  - oc extract secret/installer-cloud-credentials -n openshift-image-registry --to=--

- Note! You won’t see a PVC ‘system-storage’, unlike with the other option (NFS)

```
kind: Secret
metadata:
  creationTimestamp: null
  name: aws-auth
stringData:
  AWS_ACCESS_KEY_ID: AKIA4TELPBRUEPWZQW55
  AWS_SECRET_ACCESS_KEY: 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
  AWS_BUCKET: cluster-3003-f6bqs-image-registry-eu-west-1-sqpcmucqiymddjgiql
  AWS_REGION: eu-west-1
type: Opaque
```


Red Hat OpenShift Container Platform
Administrator
Home
Operators
OperatorHub
Installed Operators
Workloads
Pods
Deployments
DeploymentConfigs
StatefulSets
Secrets
ConfigMaps
CronJobs
Jobs
DaemonSets
ReplicaSets
ReplicationControllers
HorizontalPodAutoscalers
Networking
Storage
Builds
Monitoring
Compute
User Management
Project: 3scale-test
false
Resource Requirements Enabled
false
Monitoring
System
Redis Image
Redis Tolerations
Sphinx Spec
Redis Affinity
Affinity is a group of affinity scheduling rules.
Memcached Image
Memcached Tolerations
Redis Persistent Volume Claim
Sidekiq Spec
Image
Database
App Spec
File Storage
Amazon Simple Storage Service
Deprecated
Persistent Volume Claim
Union type. Only one of the fields can be set.
Simple Storage Service
Configuration Secret Ref *
LocalObjectReference contains enough information to let you locate the referenced object inside the same namespace.
Name
aws-auth
Name of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names TODO: Add other useful fields. apiVersion, kind, uid?

Red Hat

APIcast (API Gateway)

NGINX - Web Proxy

Lua - Built-in & Custom Policies

OpenResty - Modules to support in various protocols

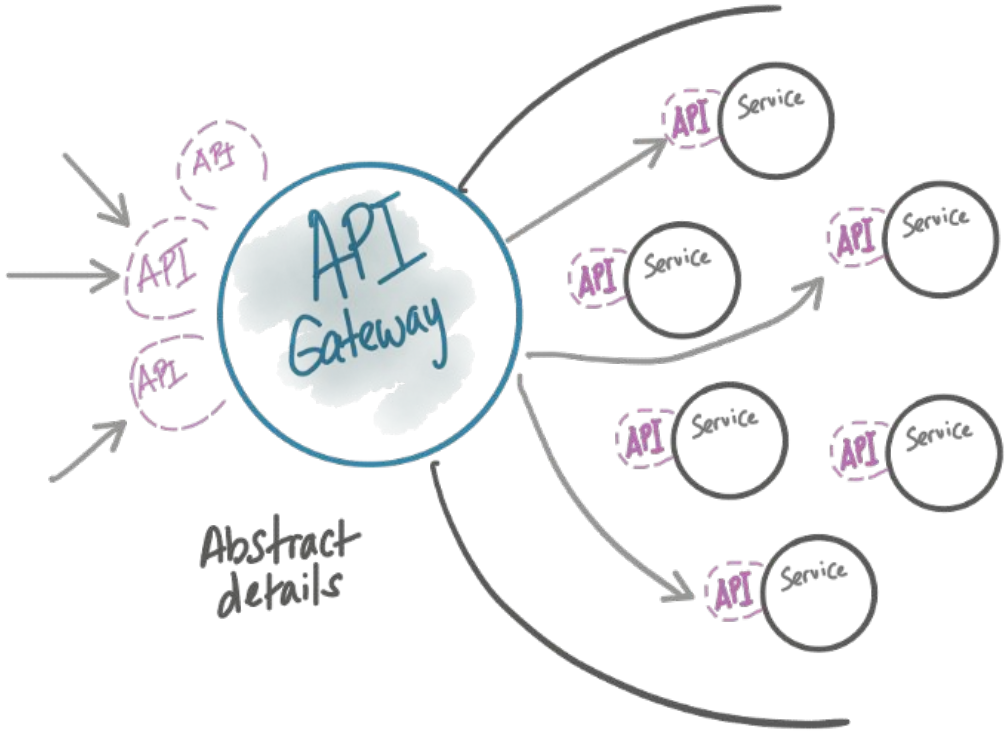
Supported API Protocols

- REST API (inc. HTTP 2)
- APICast Policies
  - Simple SOAP (Can be extended with RH Fuse)
  - gRPC
  - WebSocket
- Graphql, other RPC protocols - Not supported
- Other protocols (Tcp etc.) - built-in/custom policies, RH-Fuse

3Scale Technical Glossary

- Backend - Internal Service
- Product - Externally exposed Service via APICast
- Method mapping - permit access to specific endpoint-functions
- Metrics - Counting hits to endpoint-functions

Products & Backends association



Red Hat

API Products & API Backends

API Product
public/managed

Business layer

- Public facing (“facade”) → URL structures
- Can use multiple API backends
- Sign-up → API creds → AuthN
- App plans = limits and pricing rules
- (Global) metrics/mapping rules
- APIcast policies

Gateway

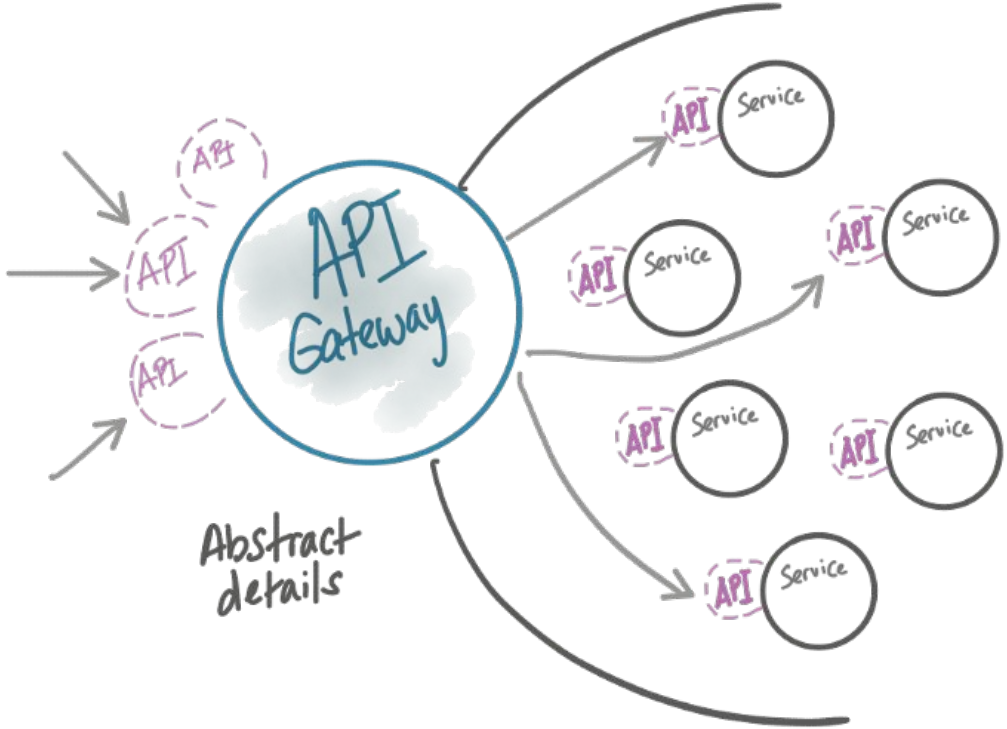
Path-based routing

API Backend
private

Technical layer

- Private base URL
- Can be used by multiple API products
- (Local) metrics/mapping rules (by method)

Take another look, Just for the sake of it



Red Hat

Mapping Rules 101

Basic: GET /page

Wildcard: GET /frontpage/{version}

Exact Match: GET /wow$

Passing values: From URL: GET /something?example_value={value} Or From Body: POST/PUT/DELETE /something?another_example_value={value}

Note! More than one mapping rule can match the request path, but if none matches, the request is discarded with an HTTP 404 status code.

Mapping Rules

Product-level Mapping Rules

- Take precedence – i.e. go on top of all MRs
- Always evaluated, no matter to which Backend the traffic will be routed

Backend-level Mapping Rules

- Evaluated after Product-level MRs
- Evaluated only if the traffic is being routed to the same Backend the Mapping Rule belongs to
- Same set of MRs exists in all Products using the Backend
- The path of the Backend in a given Product is automatically and transparently prepended to each MR of the Backend in that Product

Methods & Metrics

Product-level Methods & Metrics

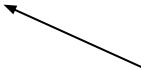
- “Hits” metric counts hits mapped to “Hits” itself and to its methods + the hits mapped to all Backend-level “Hits” metrics and their methods.

Backend-level Method & Metrics

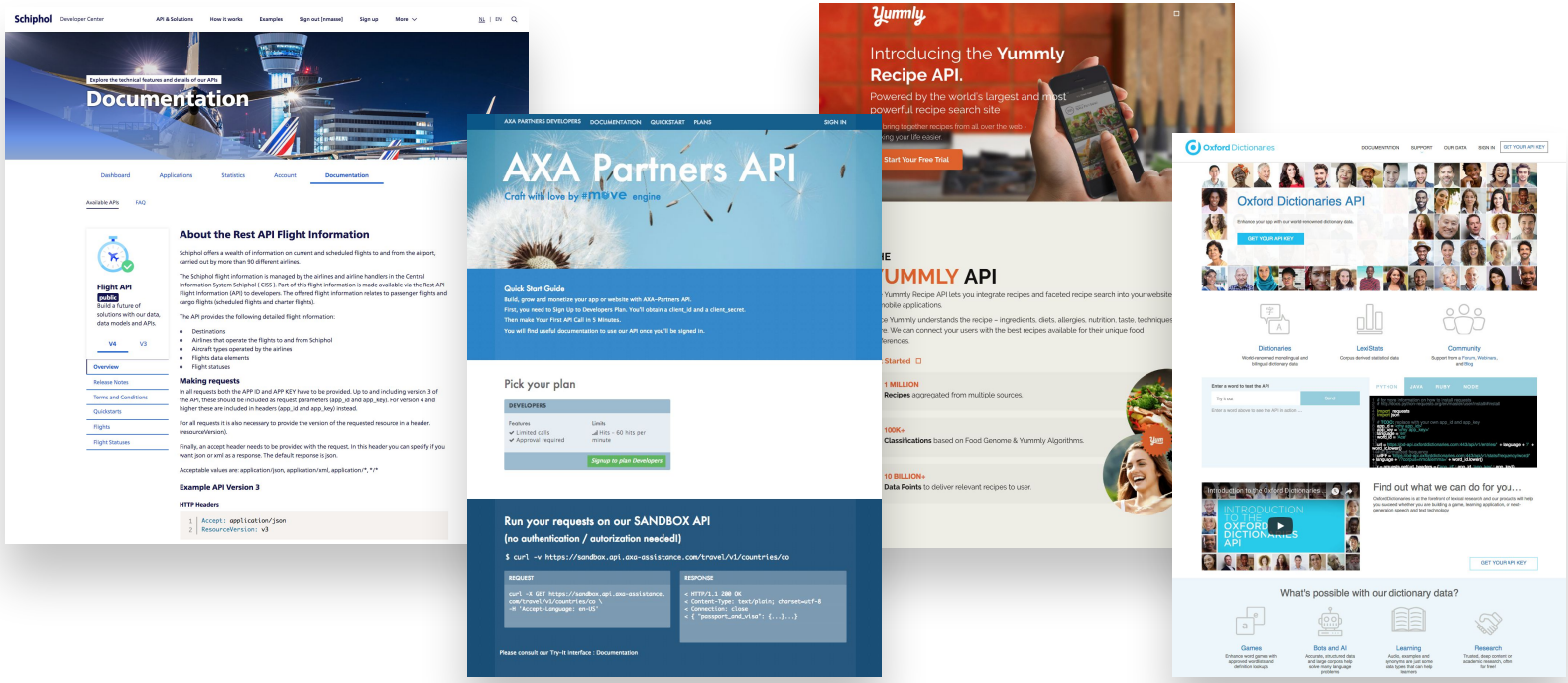
- Registered in Apisonator as if they belonged to each Product using the Backend
- Automatically and transparently get the ID of the Backend appended to the `system_name` of the metric
- Limits and Pricing rules can be set upon Backend-level metrics in the Application Plans at the Product level

Complete Example - Expose APIs via 3Scale

Starting with the backend and then associating them with the products



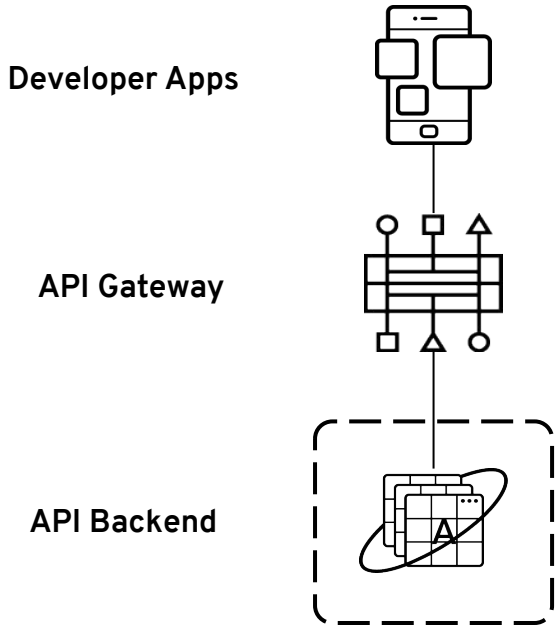
DEVELOPER & PARTNER PORTAL



Red Hat

Most Basic & Common Scenario

HTTP/S Web Services Endpoints



Relevant Use-Cases

- Service is already a HTTP REST / Simple SOAP Endpoint
- Needs to implement access control and security
- Direct routing to service implementation

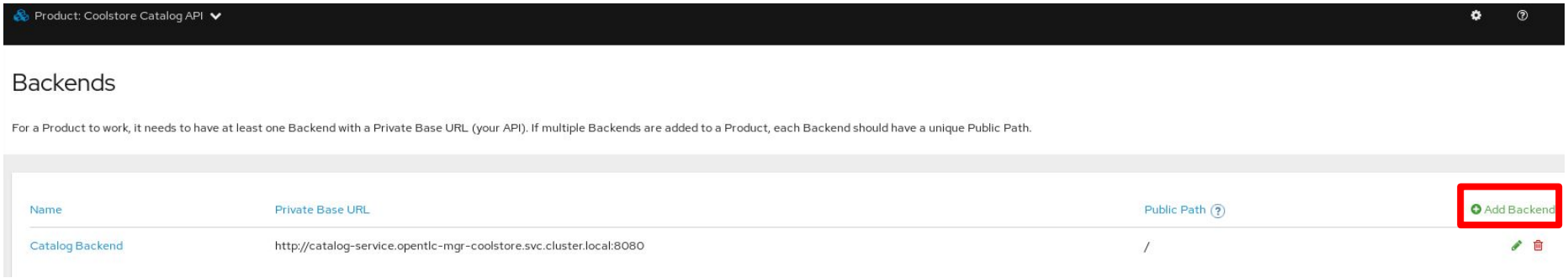
Red Hat

Give it a try

Exercise #2

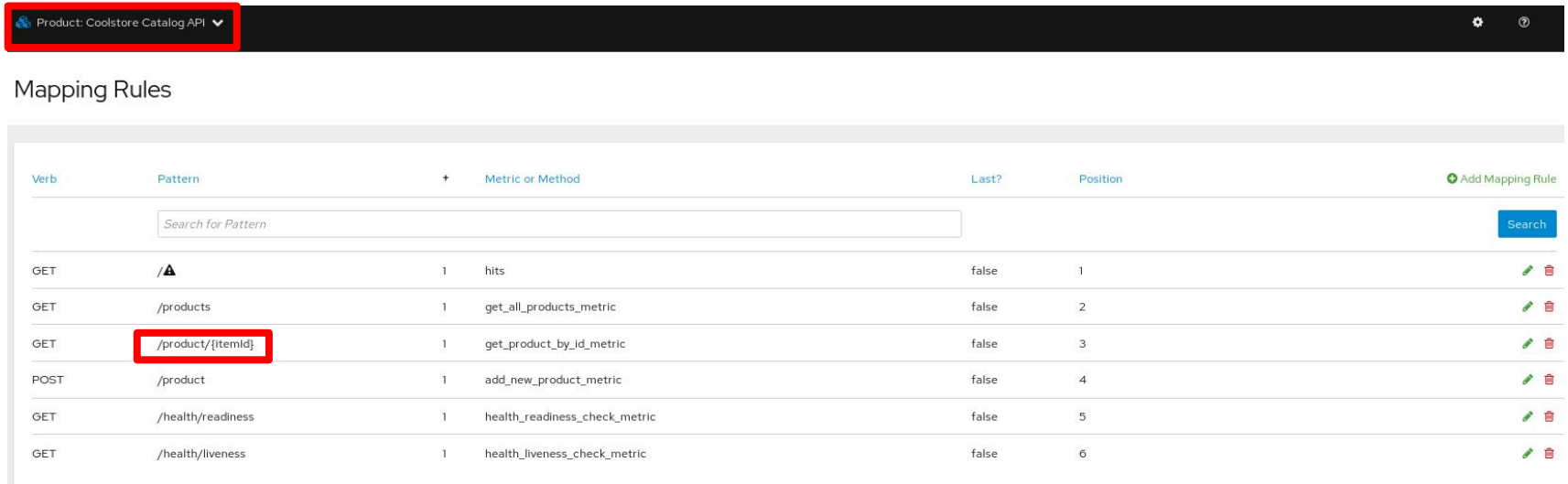
Features Features Features

Multiple Backends per-APICast



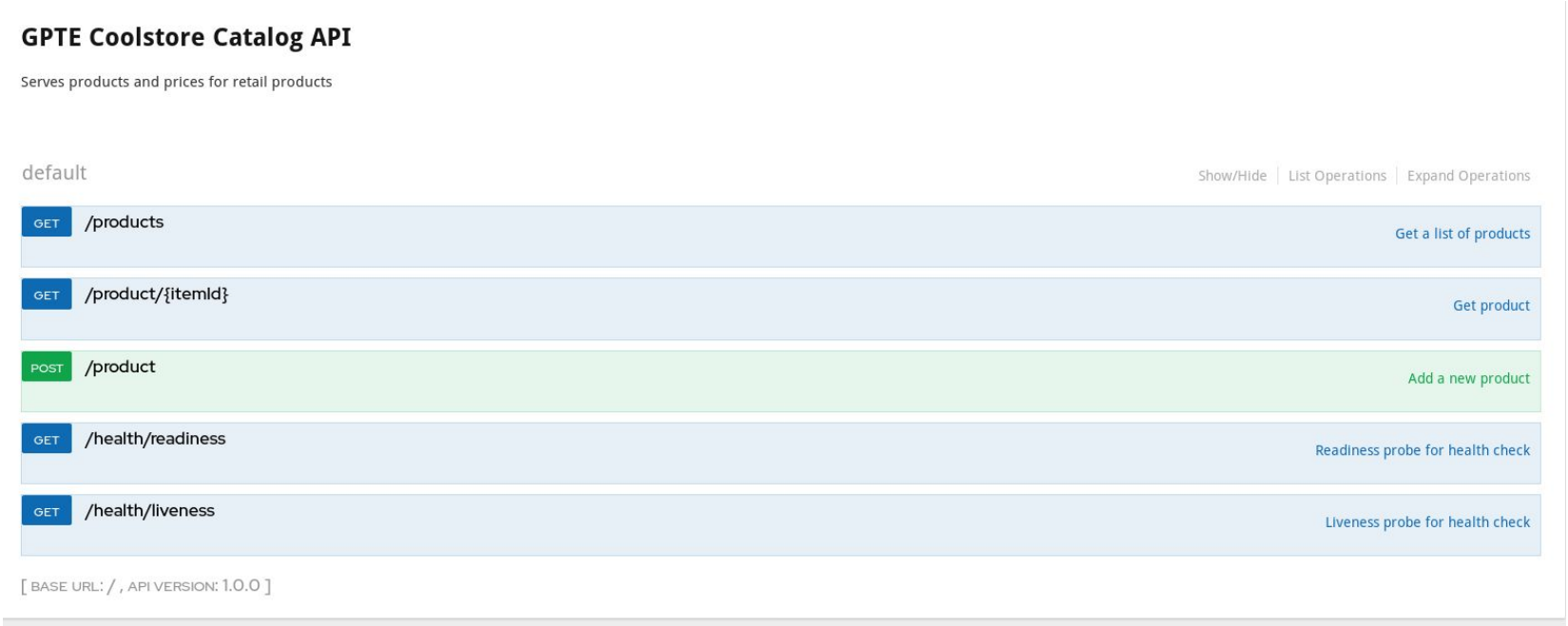
Red Hat

Mapping Rules & Wildcards



Red Hat

ActiveDocs



Red Hat

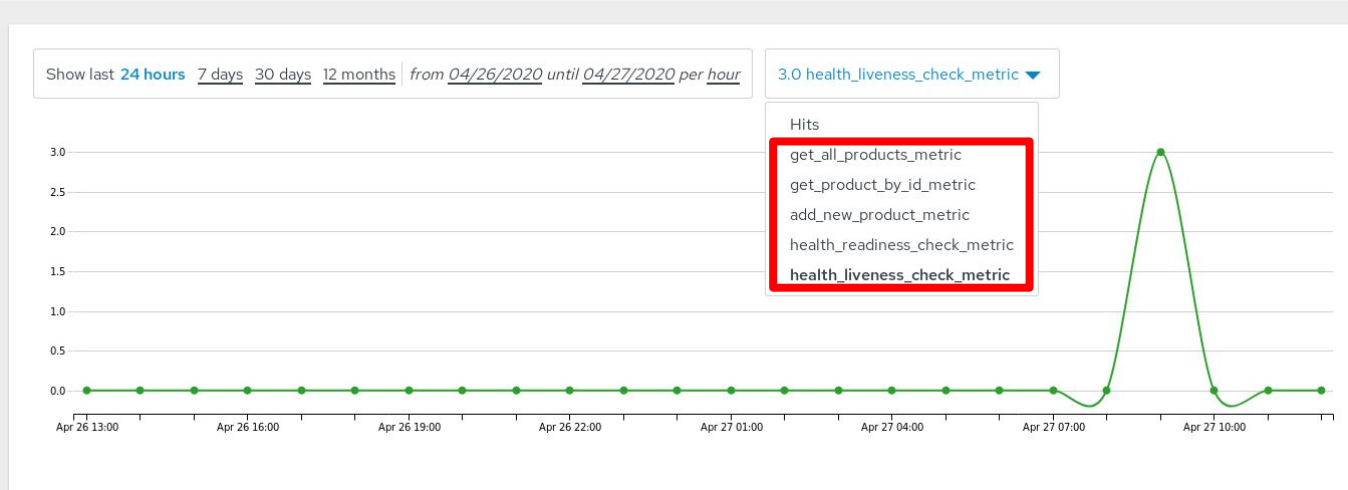
3Scale Service Discovery

In-Cluster Services

```
...

apiVersion: v1
kind: Service
metadata:
  annotations:
    description: The data virtualization services.
    discovery.3scale.net/port: "8080"
    discovery.3scale.net/scheme: http
    openshift.io/generated-by: OpenShiftNewApp
  creationTimestamp: 2019-02-05T13:13:34Z
  labels:
    app: stock-api
    application: stock-api
    discovery.3scale.net: "true"
    template: stock-api
    xpaas: 1.4.0
  name: stock-api

...
```

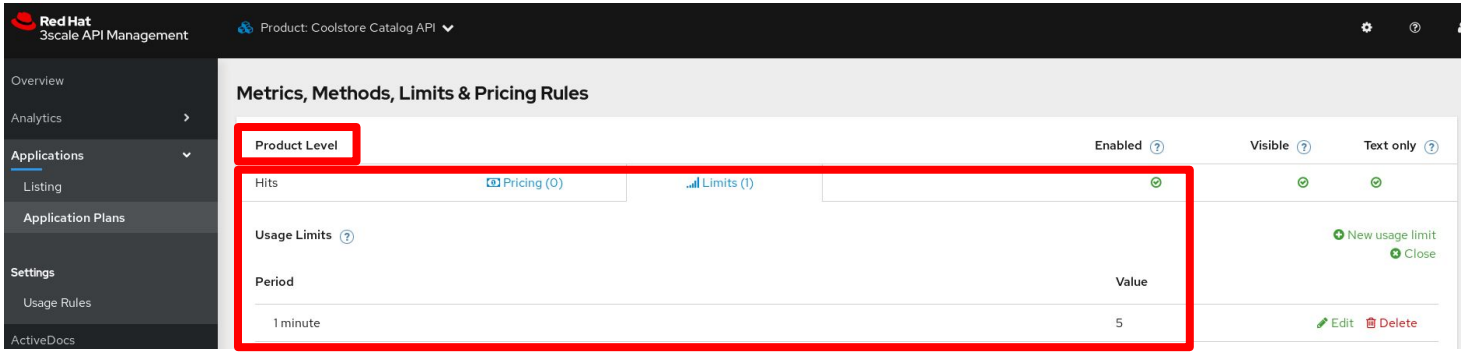
Monitor Usage Per Method

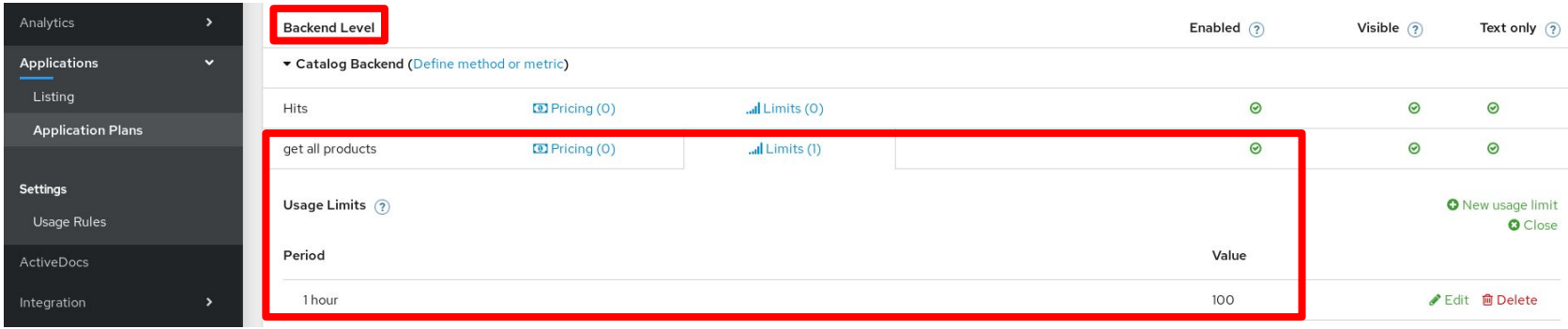
Usage



Red Hat

Rate Limit entire API vs. Rate Limit specific method





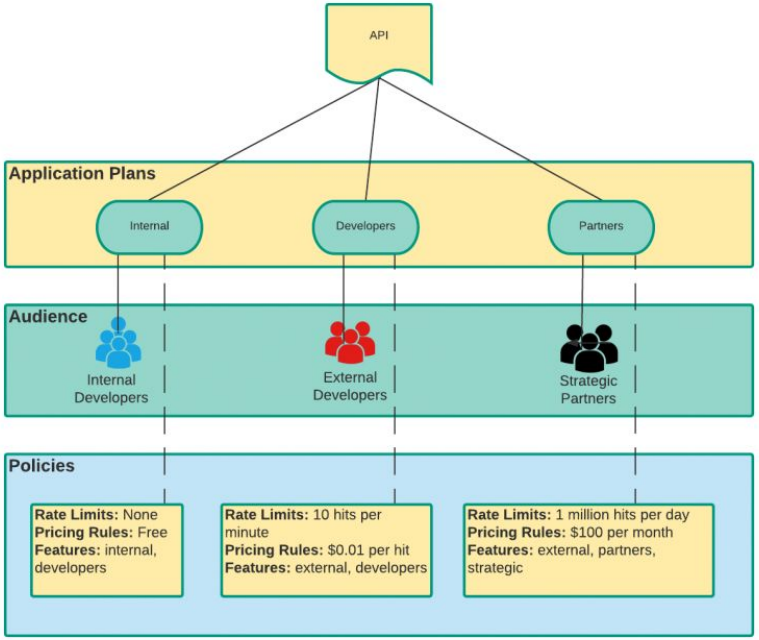
Red Hat

Prevent Access to Specific Method

Name	Applications	State
Basic	1	published
Premium	2	published

Backend Level			Enabled	Visible	Text only
▾ **Catalog Backend** (Define method or metric)					
Hits	Pricing (0)	Limits (0)	✓	✓	✓
get all products	Pricing (0)	Limits (0)	✓	✓	✓
get product by id	Pricing (0)	Limits (0)	✓	✓	✓
add new product	Pricing (0)	Limits (1)	✗	✗	✗

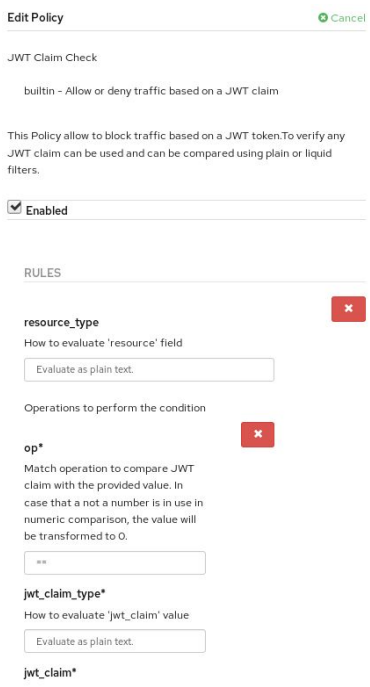
Policy Separation by Group Association



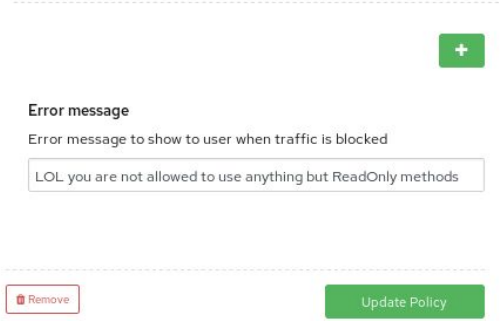
Red Hat

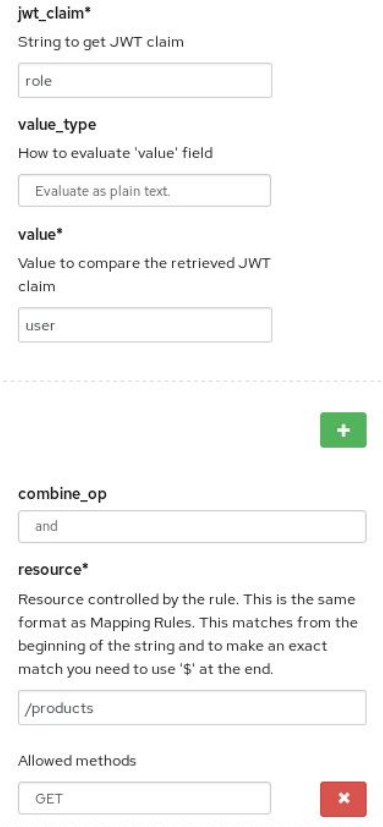
Policies - Examples

JWT fields Validation



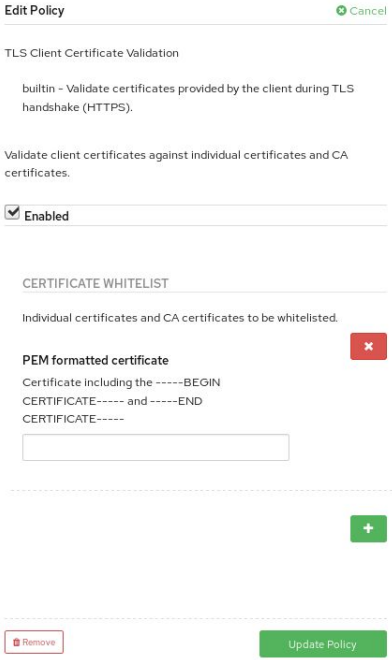
[Complete Example Reference]





Policies - Examples

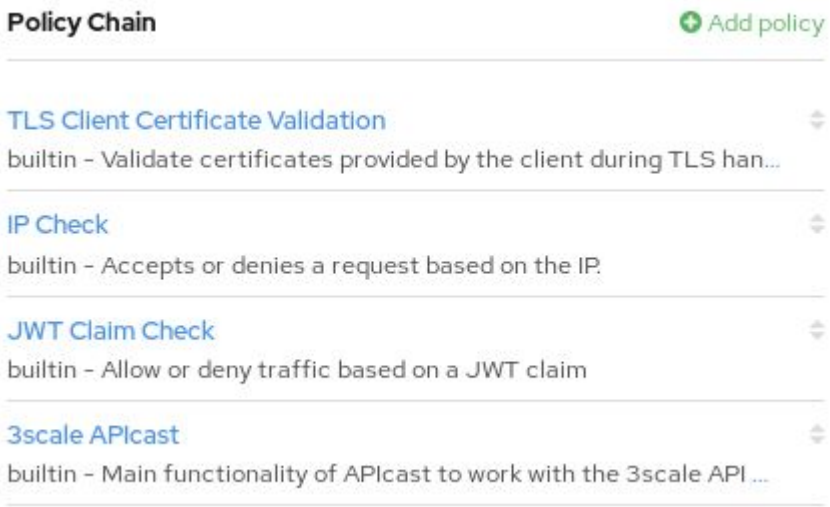
TLS Certification Validation



Red Hat

Policies - Examples

Multiple policies & Policy Chain order



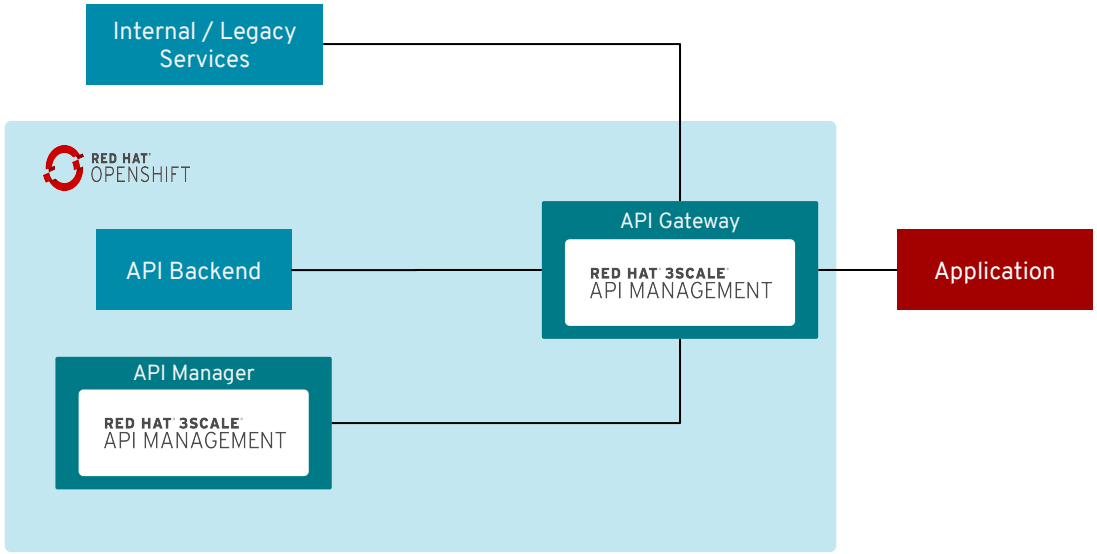
Red Hat

Do it yourself

Exercise #3

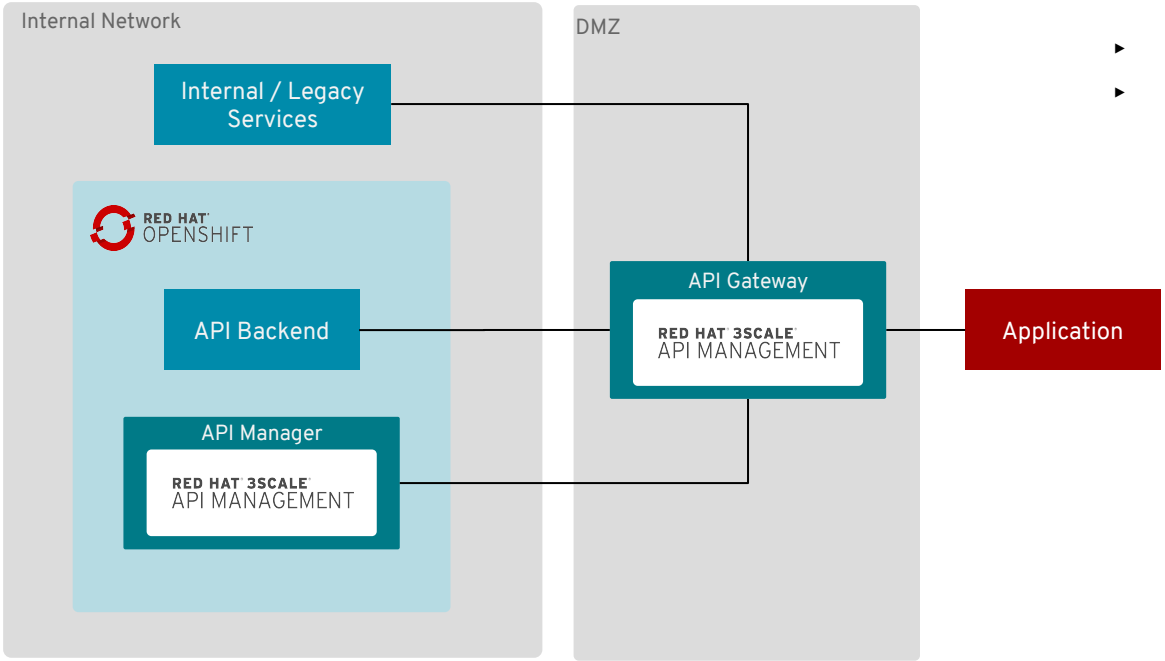
Day 2 Deployment Options

APICast Internal deployment



Red Hat

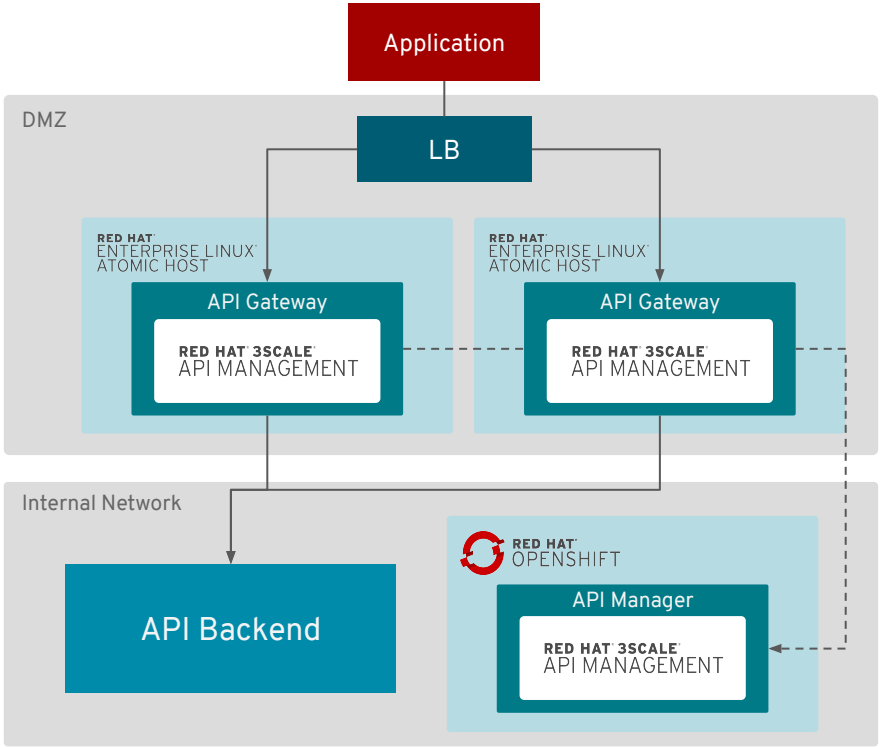
APICast external deployment



- AMP - Must Run on OCP
- APICast - Can run on every Dockerized environment outside of OCP
  (Bare Metal / Standard VMs for example)

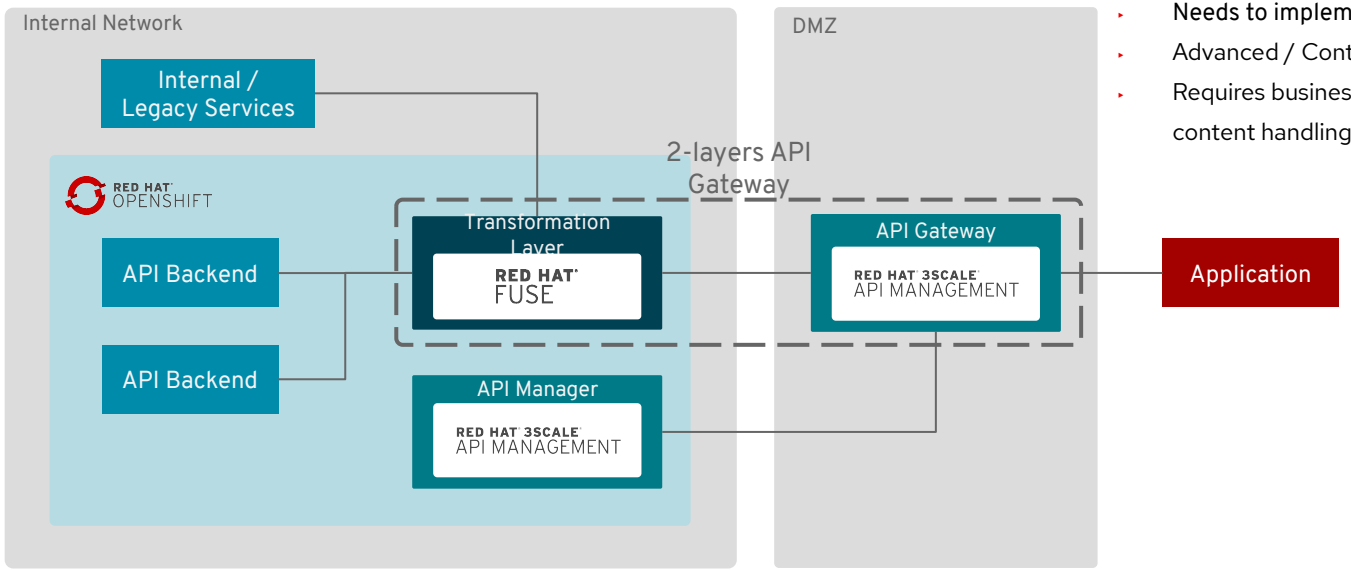
Red Hat

APICast External & High Availability



Red Hat

Fuse Transformation Layer + 3Scale API Gateway

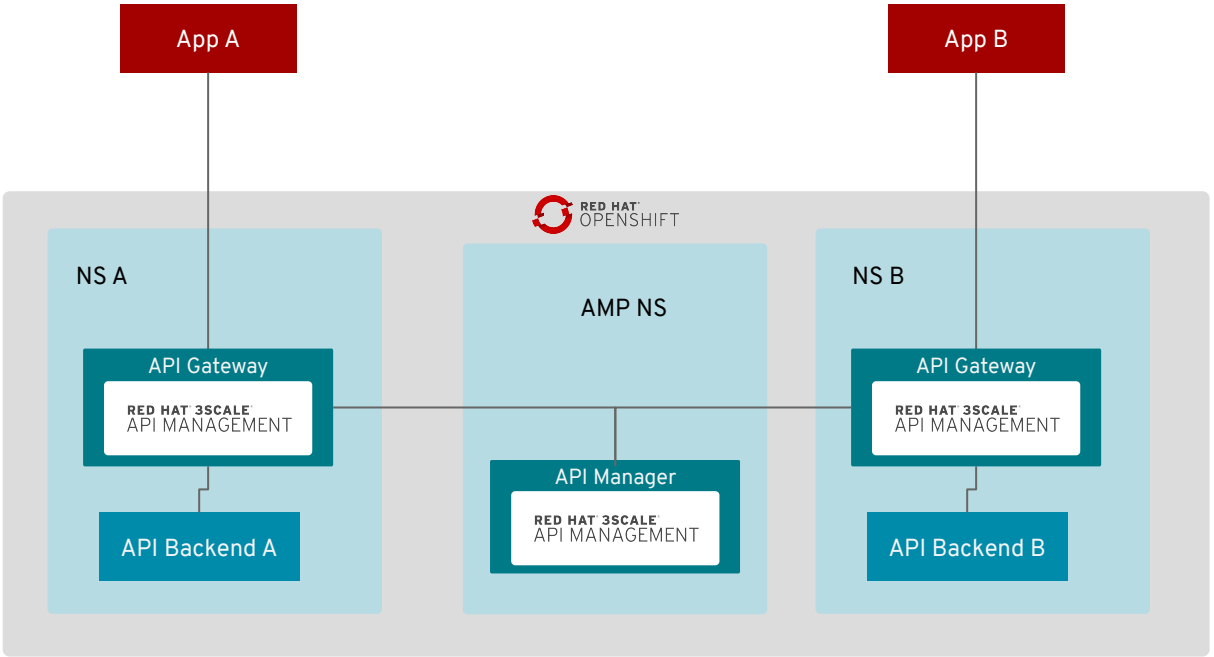


Relevant Cases

- Service is Complex SOAP Endpoint
- Non-HTTP protocols: TCP, Messaging
- Needs to implement access control and security
- Advanced / Content-based Routing
- Requires business-based transformations / content handling

Red Hat

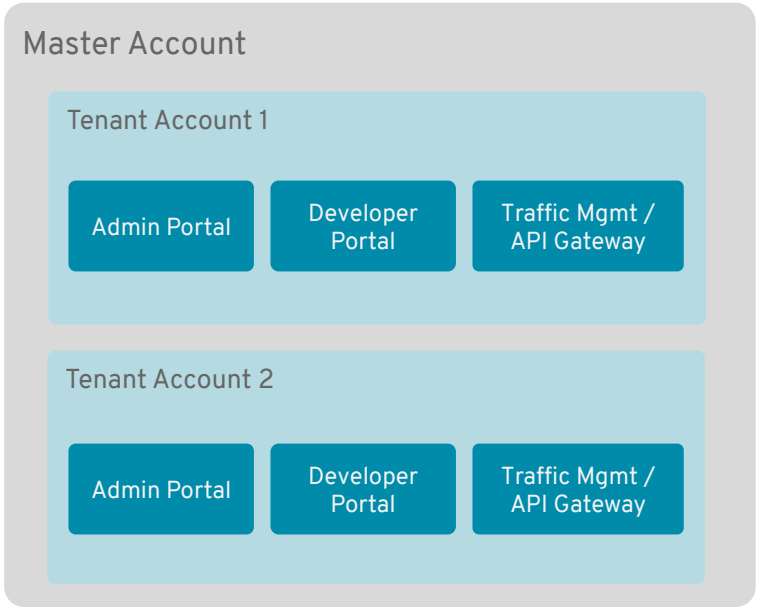
APICast Self-Managed
APICast Dedicated Per-App, Without risking AMP



Medium Article Guide

Red Hat

Multi-tenant



Master Admin

- Manage Tenants
- Impersonate Tenants

Tenant Admin

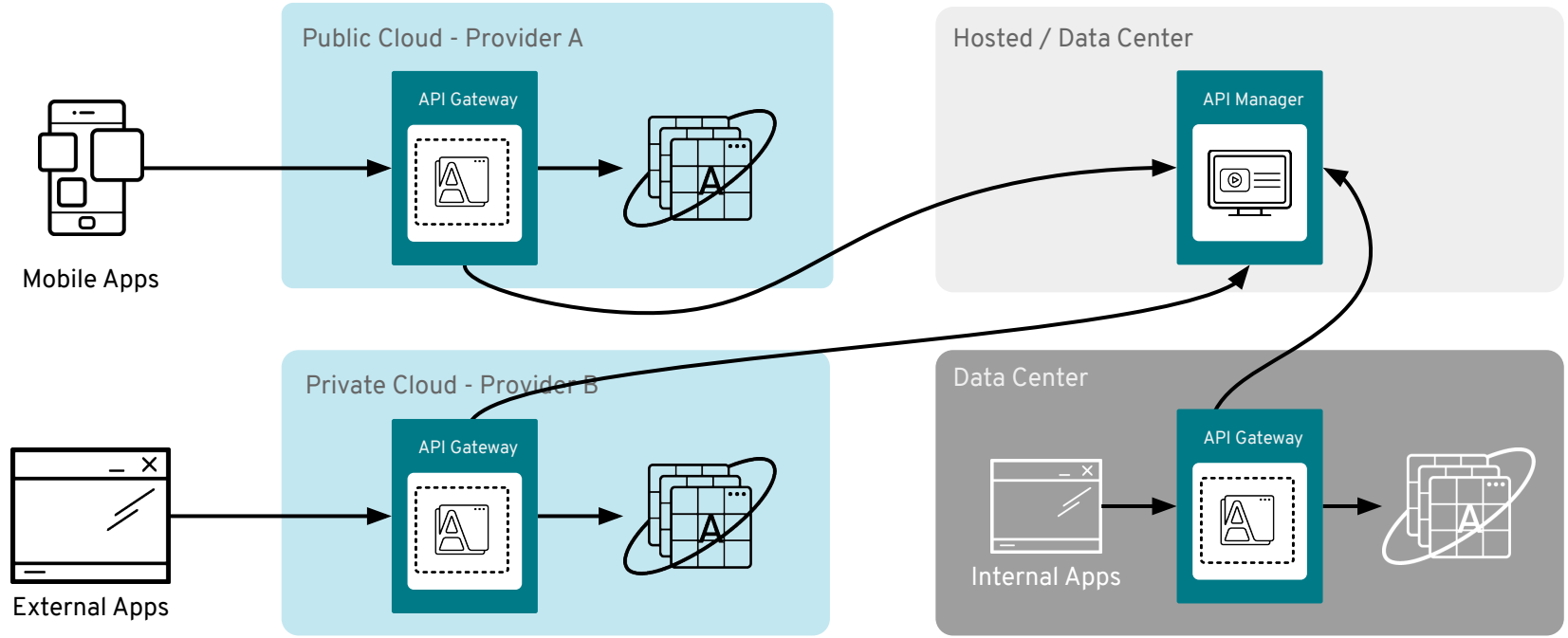
- Manage tenant admins / users
- Access APIs and Admin Portal

Member

- Access given services / sections

Installation Guide - My Medium Article

Hybrid Deployment



Red Hat

API SECURITY

Main Security Features

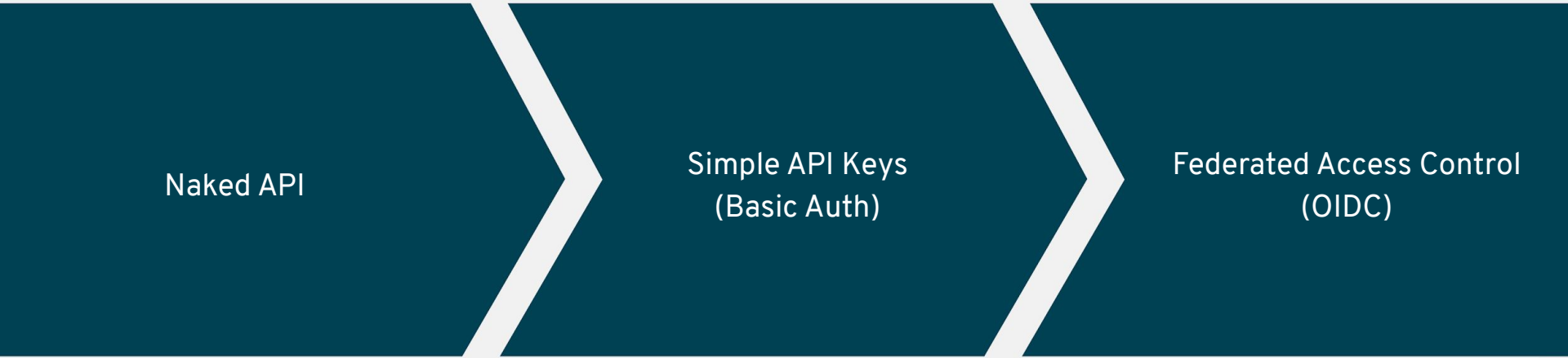
- **Authentication**
  - OpenID Connect / Basic Auth
- **Authorization** (RBAC)
  - Application Planes & Access Policies
- **Method mapping**
- **Built-in & Custom Policies**
  - Drop unwelcome requests by policies rules

Main Security Features

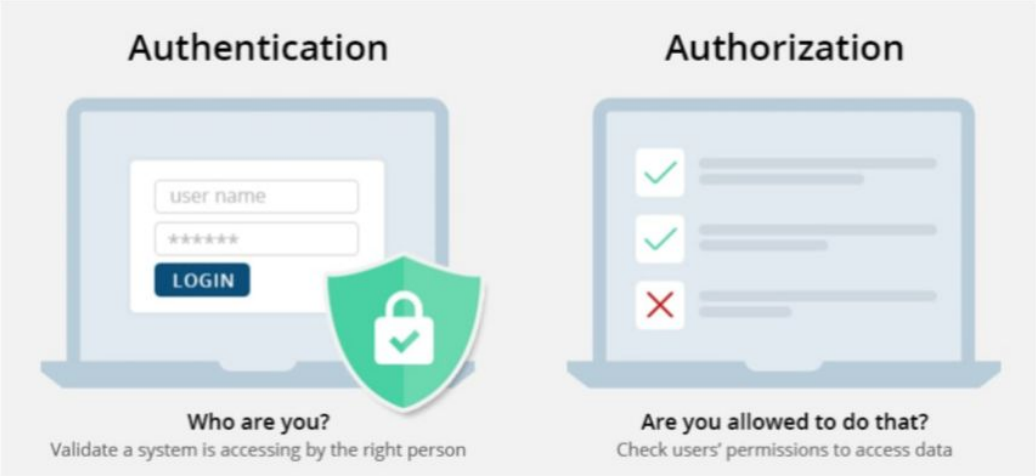
- **Generate Alerts** for suspicious traffic
- **Rate Limiting**
  - Metrics
- **Advanced Security - Collaboration with other security products**
  - SSO
  - Service Mesh
  - K-WAF

API SECURITY

Evolution of API Security



Auth vs. AuthZ



Red Hat

AUTHENTICATION

API Key: a shared secret used to authenticate a client application. Cannot easily be renewed.

API Key Pair: an identifier + a shared secret used to authenticate a client application. The identifier remains the same during the whole lifetime of the application, the secret can easily be renewed to ensure higher security.

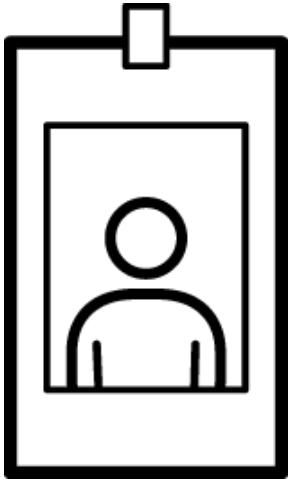
OpenID Connect: a standard protocol to authenticate the client application and the end-user connected on this application. Currently the highest level of security.

OPENID CONNECT - Auth + AuthZ

- Built on top of the OAuth 2.0 protocol
- Stateless, JWT token
- Allows clients to verify the identity of an end user and obtains basic profile information
- RESTful HTTP API, using JSON as a data format
- My Medium Article - 3Scale & RHSSO (OpenID Connect Implementation)

OPENID CONNECT

- Provides identity information to the application from the Authority Server
- Base64 encoded - easy to work with.



Name: John Doe

Type: Employee

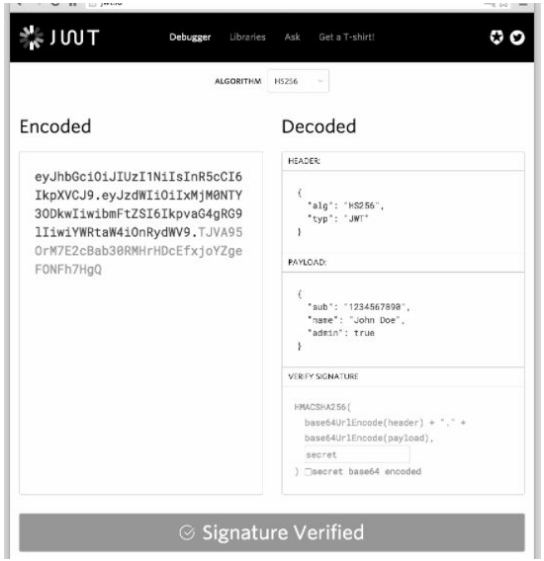
Issued by: Company

Expiration Date:

02-06-2019

JWT (“JOT”) to the Rescue

- Signed by algo and verified by only correct key
- Contains user identity in form of claims (Private, public, reserved)
- For OIDC purpose, SSO is widely adopted in consumer/enterprise apps
- Eliminates the need to look up against a central access control list

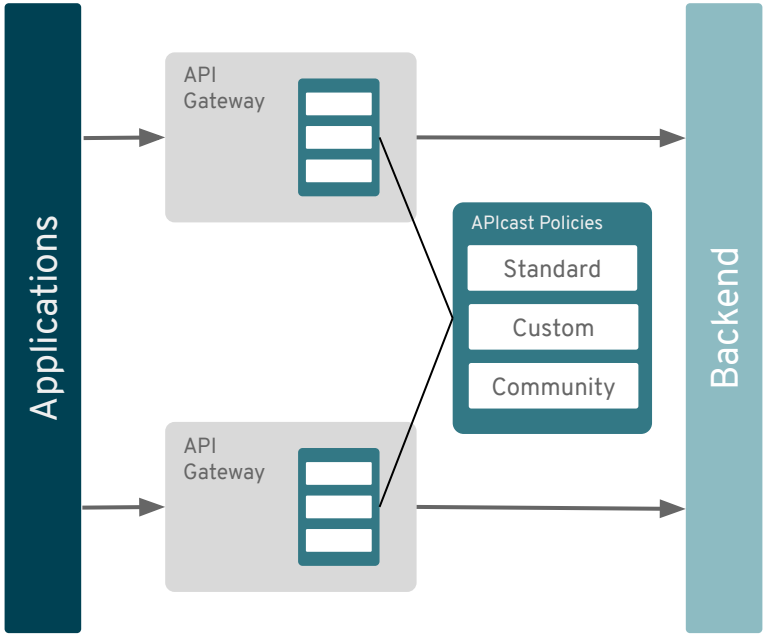


Red Hat

3SCALE API GATEWAY POLICIES

Modular Policy Architecture Benefits

- Descriptive configuration, not code
- Add gateway logic with new policies for any phase of the request cycle
- Better extensibility
- Improved maintainability
- Leverage community contributions
- OOTB policies configurable from UI

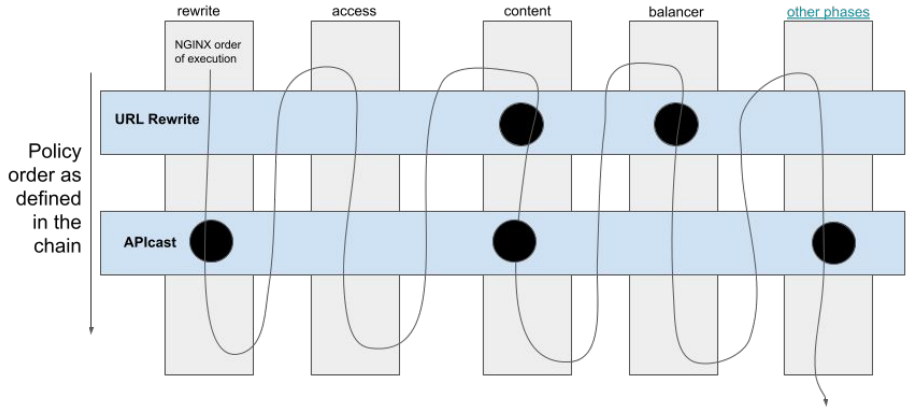


Red Hat

Built-in Policies

- **OAuth Token Introspection**: Executes OAuth 2.0 token introspection for every API call
- **Upstream**: Allows modify the upstream URL of the request based on its path
- **Edge Limiting**: Algorithm-based rate limiting, allows global and per service caching
- **Url Rewriting:** Allows modification of a path request
- **SOAP**: Adds support for small subset of SOAP
- **CORS**: Enable Cross-Origin Resource Sharing
- **HTTP Headers**: Allows control of HTTP request and response headers
- **Echo**: Prints the request back to the client (status code optional)
- **Auth Caching**: Control 3scale authorization cache
- **Anonymous Access**: Provides default credentials for unauthenticated requests
- **Logging**: Enables / disables access logs per service
- **Referrer:** Sends the contents of the Referer HTTP header to backend so it can be validated
- **Metrics**: Enable backend metrics
- **Batcher**: Caches auth from backend and reports
- **IP Check**: Accepts or denies a request based on the IP

Policy Chaining



Notes:
Policies can define code for execution in any required phase.
Order of execution is as per the arrow, with processing happening at each phase:

- APIcast's operation in *rewrite* phase will be run BEFORE URL Rewrite operation in *content* phase
- APIcast's operation in *content* phase will be run AFTER URL Rewrite operation in *content* phase.

Custom Policies - in depth

Link 1

DROP Unwelcome Requests - Examples

Gateway Operations

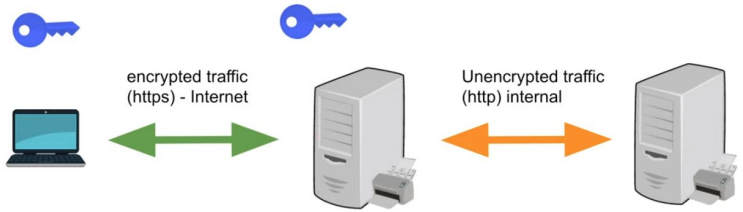
- Checks the timestamp for ‘expired’ token.
- Checks the client_id is still valid
- Performs a check on the signature of the JWT using RH SSO public key

DROP Unwelcome Requests - Examples

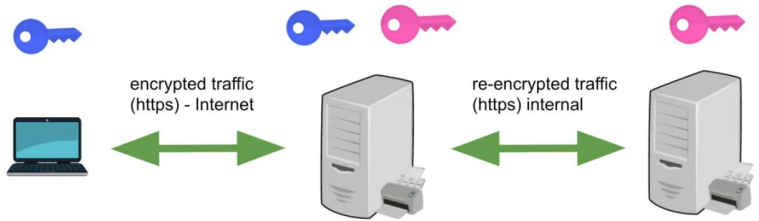
TLS Strategies (3Scale 2.7+)

- Passthrough
- TLS Termination
- TLS Forward (Re-Encrypt)

TLS Termination Proxy



TLS Forward Proxy

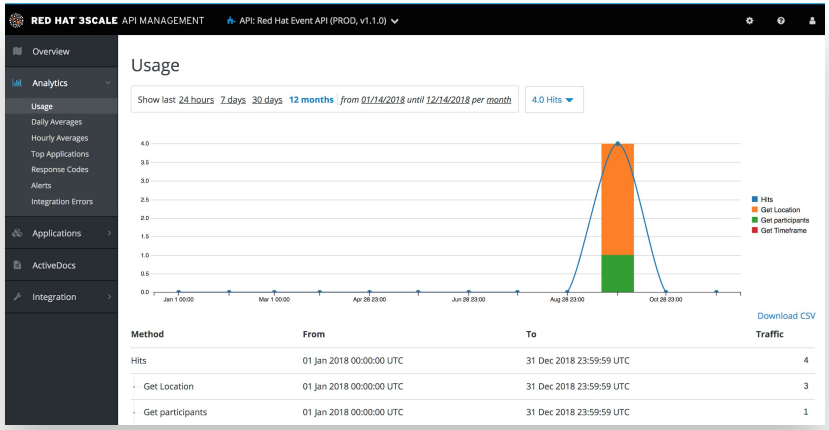


Red Hat

ANALYTICS & Alerts

Measure the success of your APIs. Take actions based on numbers

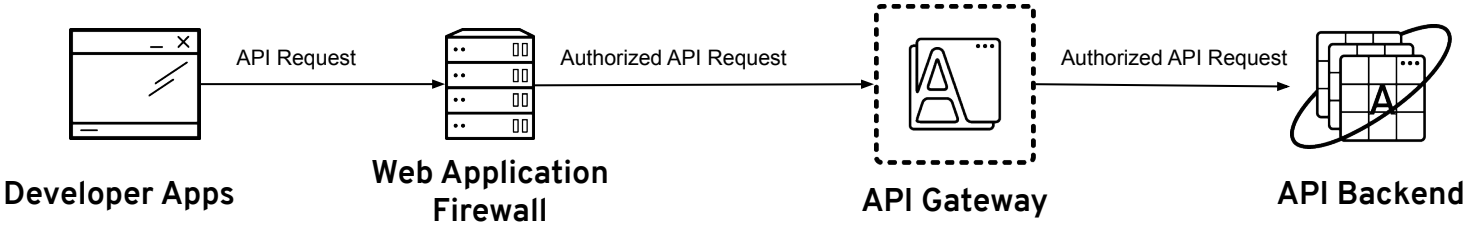
- Define and present tailored metrics
- Drill down up to the API Method level
- Implement business metrics
- Export data in CSV format



Red Hat

ADVANCED SECURITY

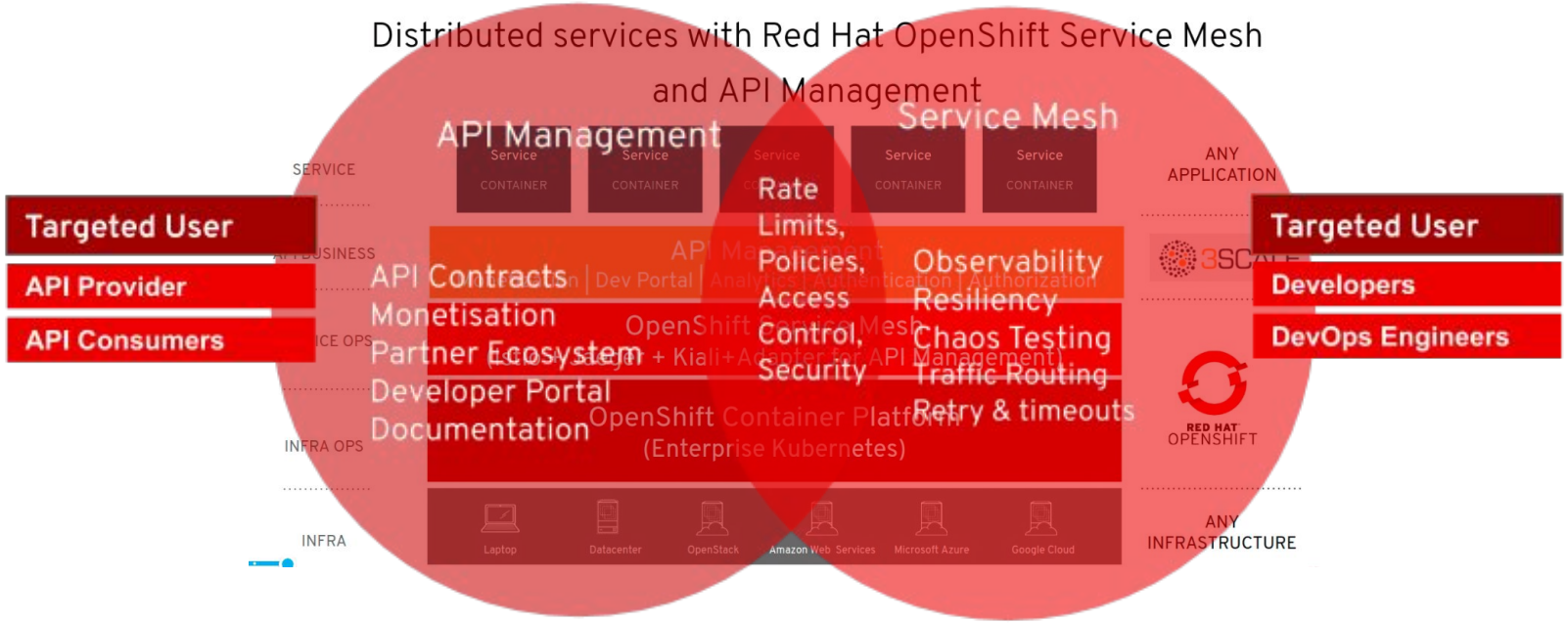
A Web Application Firewall can be used to enhance the security of your APIs



Red Hat

ADVANCED SECURITY

API Management vs. Service Mesh



Red Hat

Automation

Automation Options

- Pipelines
  - 3Scale Management API
- 3Scale Toolbox
  - Easier to implement
- **New!** YAML-oriented management via 3Scale Operator & GitOps (v2.9-2.10)
  - 3Scale-Operator Github Project & RH 3Scale Official documentation
    - **Github complete example**
    - APIManager CRD Reference
    - Backend CRD Reference
    - Product CRD Reference
    - Tenant CRD Reference
    - Video Demo

External Access to 3Scale (API Calls || Operator)

Medium Article - Example Demo

```
apiVersion: v1
kind: Secret
metadata:
  name: main-three-scale-tenant #admin
type: Opaque
stringData:
  adminURL: https://3scale-admin.example.com:443
    token: "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
```

```
apiVersion: v1
kind: Secret
metadata:
  name: mytenant
type: Opaque
stringData:
  adminURL: https://my-3scale-tenant.example.com:443
    token: "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
```

3Scale toolbox

- 3scale toolbox is a set of tools to help you manage your 3scale product.
- Official Project Git Repository
- Official Red Hat documentation
- Can be installed on laptop or run on podman/docker container

3Scale Management API

Why work hard when you can automate?

Just do the Exercise - it covers everything

(requires jq)

Any Questions?

tamber@redhat.com
https://www.linkedin.com/in/tommeramber/
https://medium.com/@tamber

Red Hat

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.


linkedin.com/company/red-hat


youtube.com/user/RedHatVideos


facebook.com/redhatinc


twitter.com/RedHat