MSB(Microservice Bus) Deep Dive

Huabing Zhao ZTE, System Engineer, Network Management & Service, OPEN-O Common Service PTL
zhao.huabing@zte.com.cn

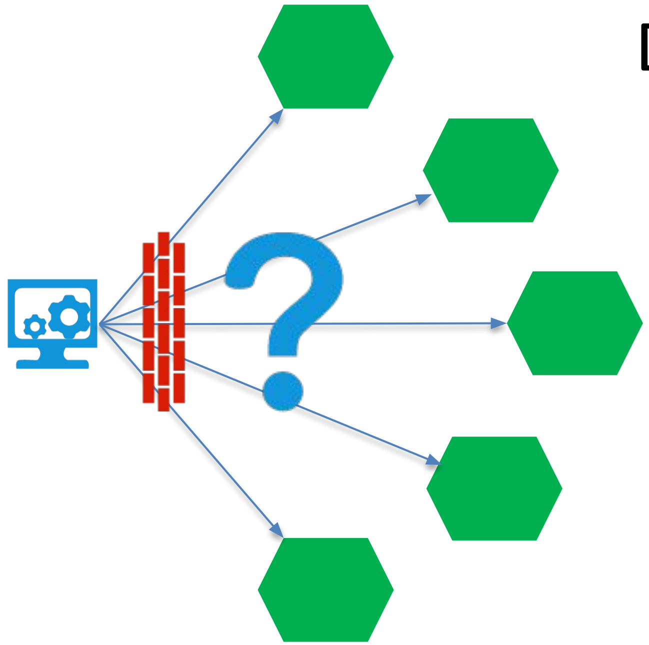
Agenda

- Current Challenges and MSB Solutions
- MSB Architecture & Features
- API & Example

Problem being Solved

- **How do the clients application access the back end services?**
- **How do the client or another service - discover the location of a service instance?**
- **How to enforce centralized authentication and authorization?**

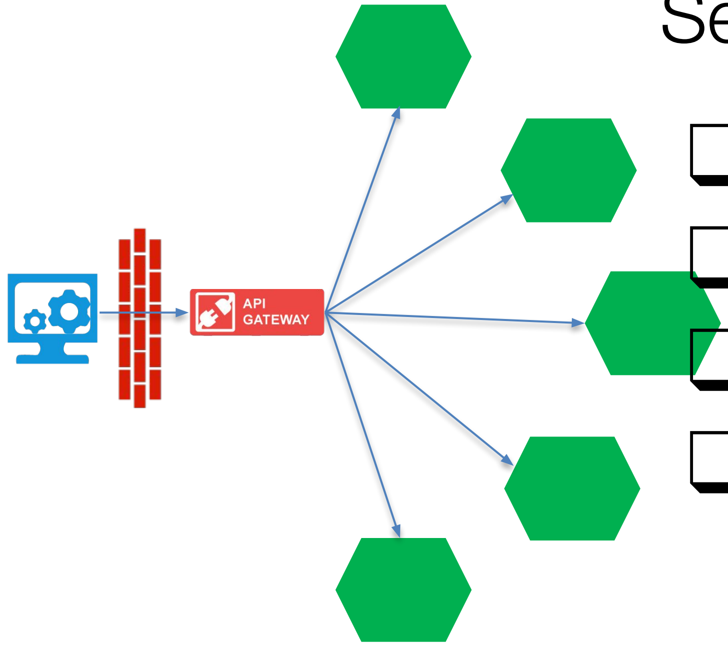
Problem: How do Clients Access Back End Services?



Direct Communication has problems:

- ❑ Add complexity to client codes
- ❑ Nightmare for firewall configuration
- ❑ Coupling of client and individual services
- ❑ Cross-domain issue for web app

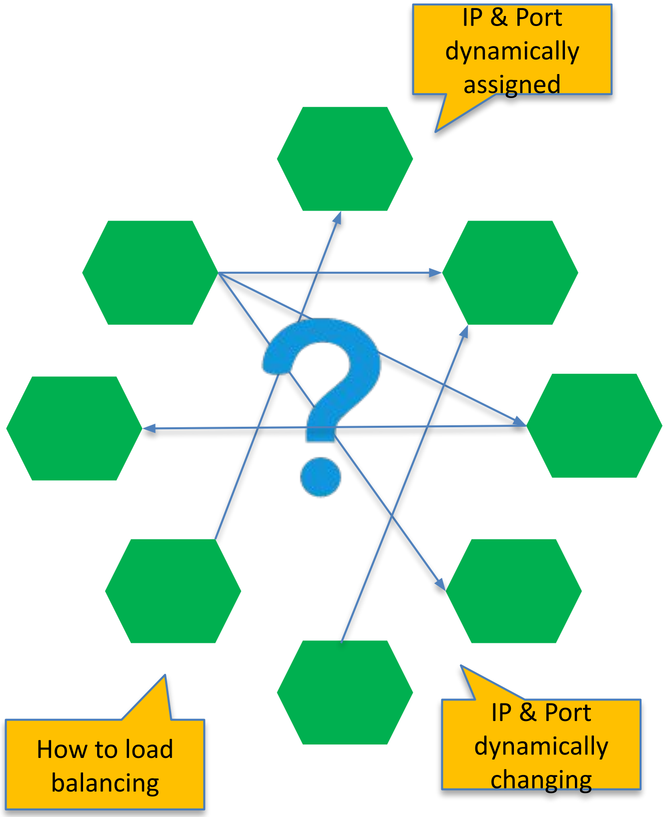
Solution: Service Gateway



Service gateway hides the complexity

- ❑Simplify the client codes.
- ❑Reduce request roundtrips
- ❑Provide API management
- ❑Solve cross-domain issue for web app

Problem: How to find the service?


IP & Port dynamically assigned
How to load balancing
IP & Port dynamically changing

In order to access a service, you need to know the exact endpoint(IP & Port)

"Traditional" application

- ❑Service endpoint doesn't change a lot
- ❑Consumer can get the endpoint from configuration files

Microservice application

- ❑The IP & port is dynamically allocated
- ❑IP & port changes along with the scaling/ updating/ self-healing of service instances

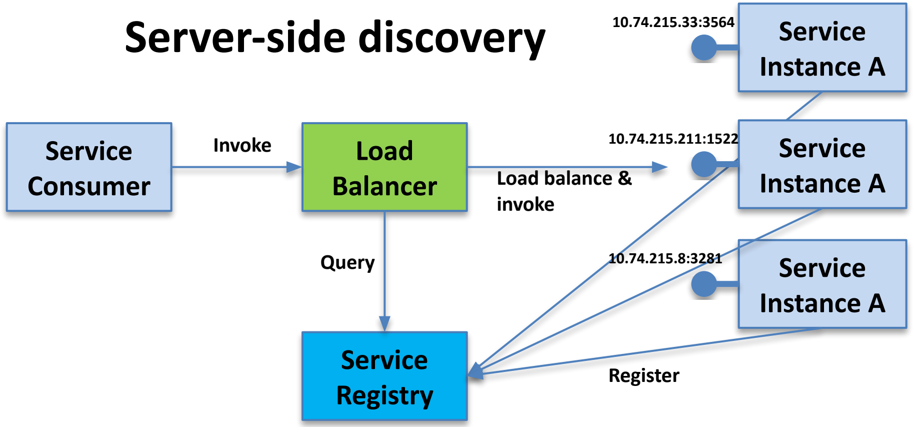
Solution: Service Registration & Discovery

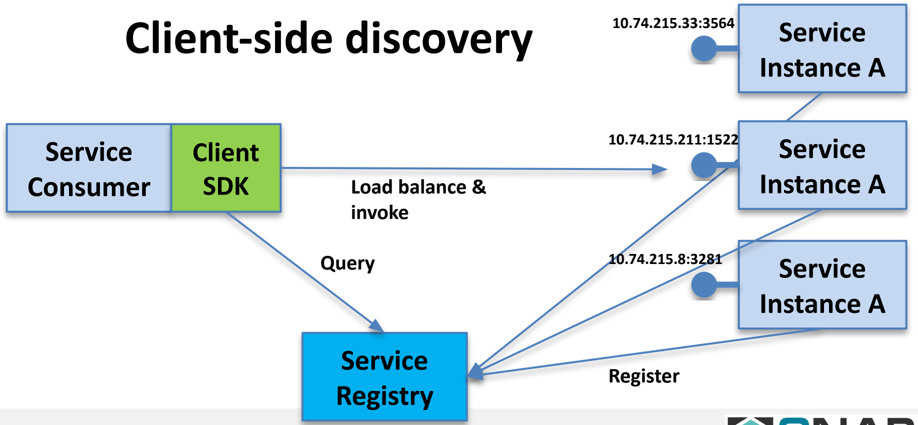
Service Registration:

- Service providers register themselves to the registry when start up
- Update service information when service instances change

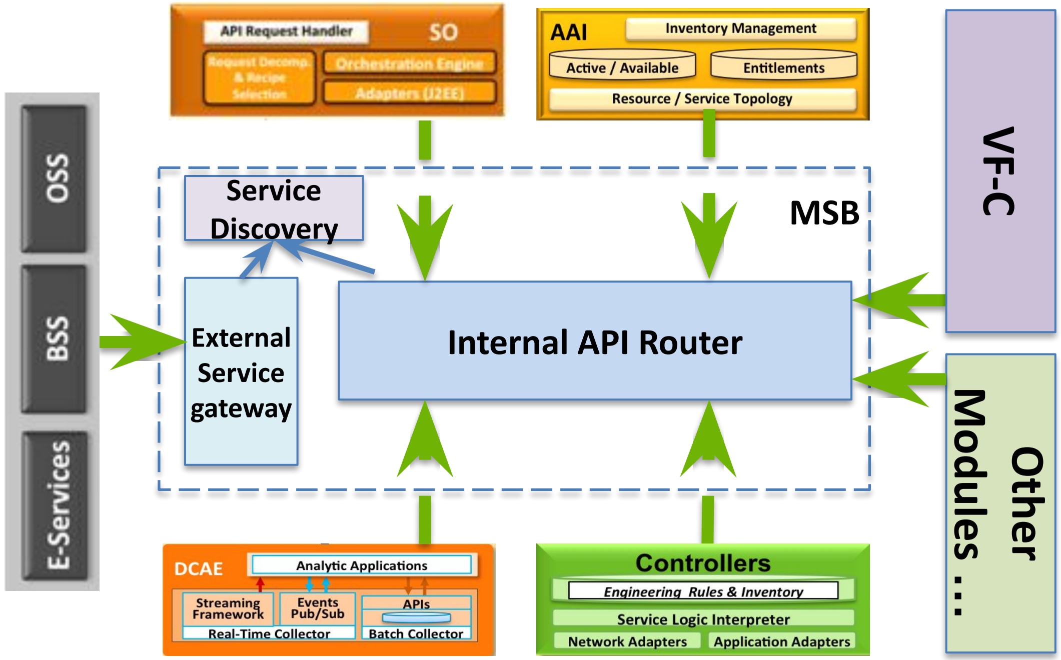
Service Discovery:

- Service consumers query registry to find the locations of service
- Two approaches: Server-side discovery & Client-side discovery





MSB Solution for ONAP: Service Discovery & Routing



MSB handles the service **discovery & routing & LB**

Before:

Using a configuration file, we might have problems on scaling, failover and update

```
"aaiEndpoint": "https://c1.vm1.aai.simpledemo.openecomp.org:8443",
"adaptersCompletemsoprocessEndpoint": "http://mso:8080/CompleteMsoProcess",
"adaptersDbEndpoint": "http://mso:8080/dbadapters/RequestsDbAdapter",
"adaptersSdncEndpoint": "http://mso:8080/adapters/SDNCAdapter",
"adaptersTenantEndpoint": "http://mso:8080/tenants/TenantAdapter",
"workflowSdncadapterCallback": "http://mso:8080/mso/SDNCAdapterCallbackService"
"adaptersNetworkEndpoint": "http://mso:8080/networks/NetworkAdapter",
"adaptersNetworkRestEndpoint": "http://mso:8080/networks/rest/v1/networks",
"adaptersVnfAsyncEndpoint": "http://mso/vnfs/VnfAdapterAsync",
"workflowVnfAdapterDeleteCallback": "http://mso:8080/mso/vnfAdapterNotify",
"workflowVnfAdapterCreateCallback": "http://mso:8080/mso/vnfAdapterNotify",
"adaptersVnfRestEndpoint": "http://mso:8080/vnfs/rest/v1/vnfs",
......
```

After:

```
"apigateway": "https://apigateway.onap.org:80"
```

MSB as the **single entry point**

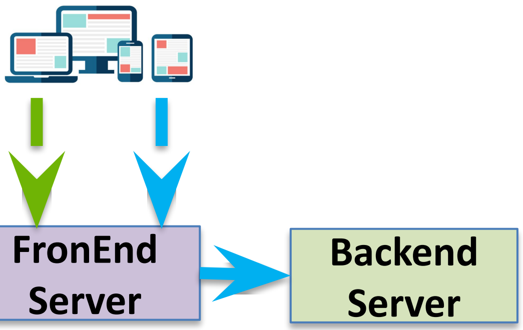
How to call service:

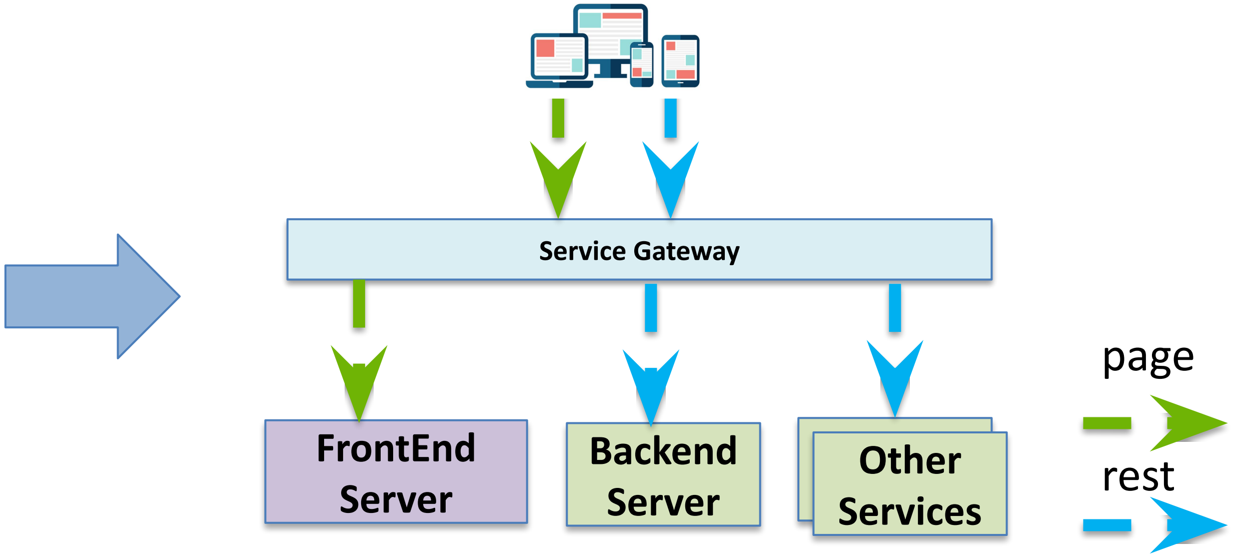
GET https://apigateway.onap.org/api/aai/v8/cloud-infrastructure/cloud-regions/cloud-region/{cloud-owner}/{cloud-region-id}

API gateway routes the request to:

GET https://c1.vm1.aai.simpledemo.openecomp.org:8443/aai/v8/cloud-infrastructure/cloud-regions/cloud-region/{cloud-owner}/{cloud-region-id}

MSB Solution for ONAP: Reverse Proxy





Before:

- ❑The business logic(rest service) forwader must be add to front end server
- ❑Solve the cross-domain issue cause coupling of business logic and UI pages

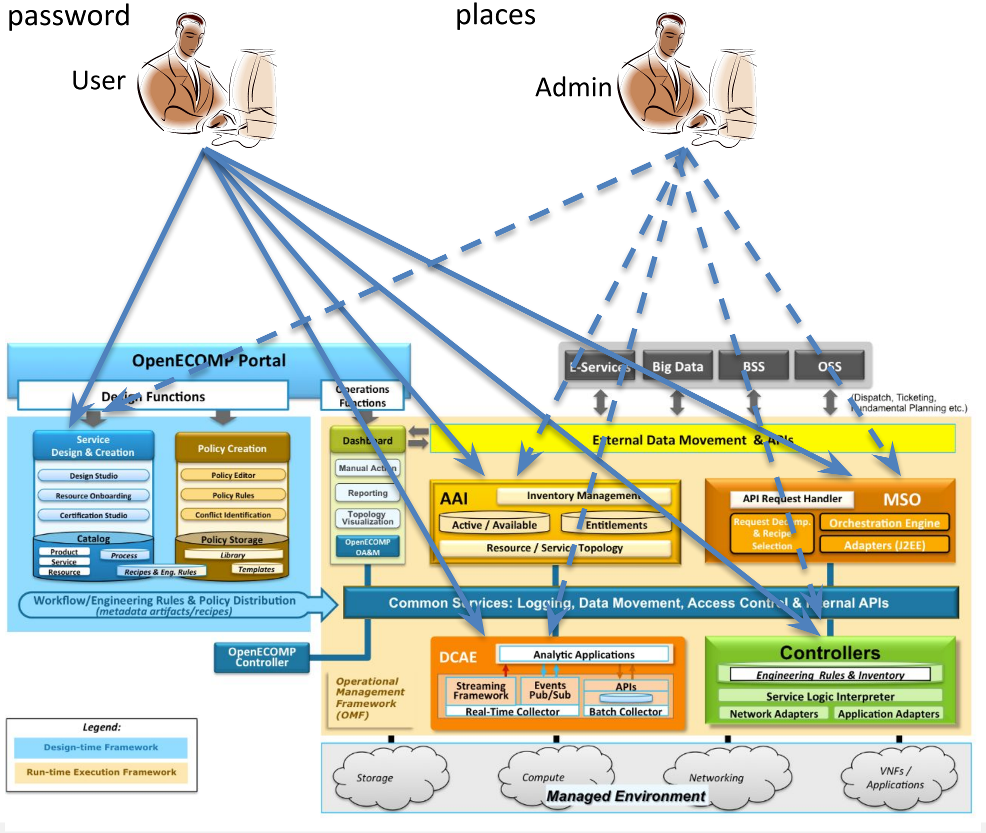
After:

- ❑service gateway to solve cross-domain issue
- ❑Cache for static resources (page, picture)
- ❑Clearer boundary between UI and business logic

Decentralized Authentication & Authorization

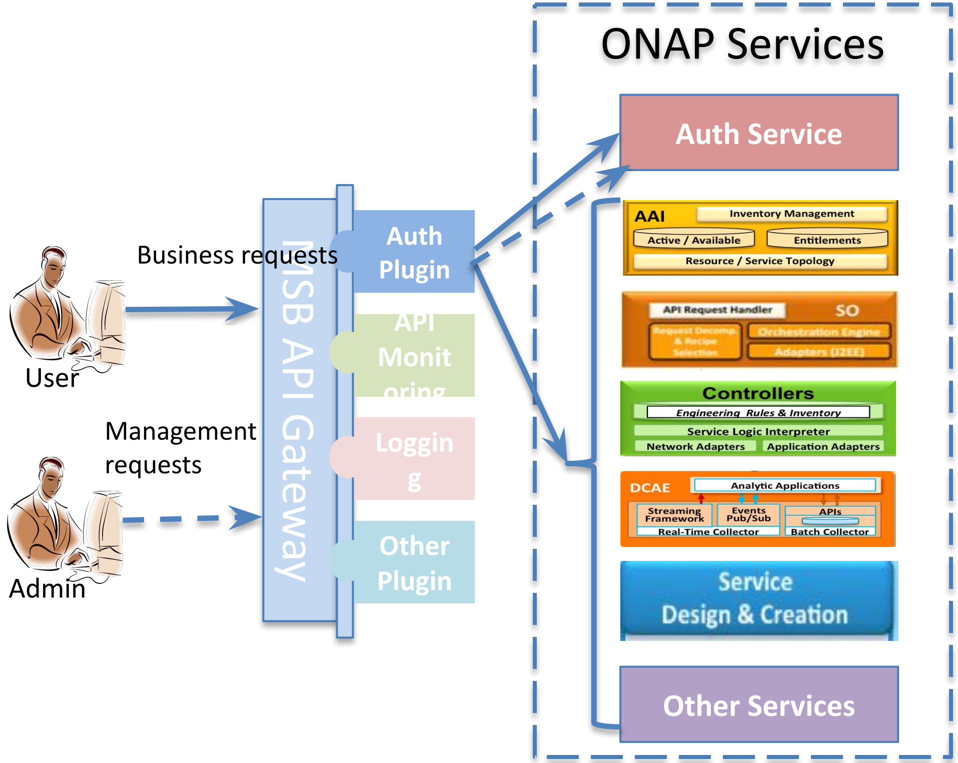
Login with different user and password

Add Users/Roles in different places



- No centralized authentication
- No centralized authorization
- No centralized user management
- There are at least 13 user/password combos that are used by the test automation to perform anything

MSB Solution: Centralized Auth with Plugin(SSO)



Centralized Authentication

1. User send a service request to MSB API Gateway
2. MSB API Gateway auth plugin check the auth token

   2.1 If a valid token exist, MSB API Gateway forward the request to the destination service provider

   2.2 If not, MSB API Gateway forward the request to the Auth Service, and redirect user request to login page

   2.3 Auth service create a token after user login with valid name and password, send the token back to user agent(browser)

Centralized Authorization(Assuming user already login)

1. User send a service request to MSB API Gateway
2. MSB API Gateway auth plugin send the user token and request(Http method + Resource url) to Auth Service to check if user has the permission to access the resource

   2.1 If user has the permission, MSB API Gateway forward the request to the destination service provider

   2.2 If not, MSB return operation not allowed error to user

Centralized User, Role and Permission Management

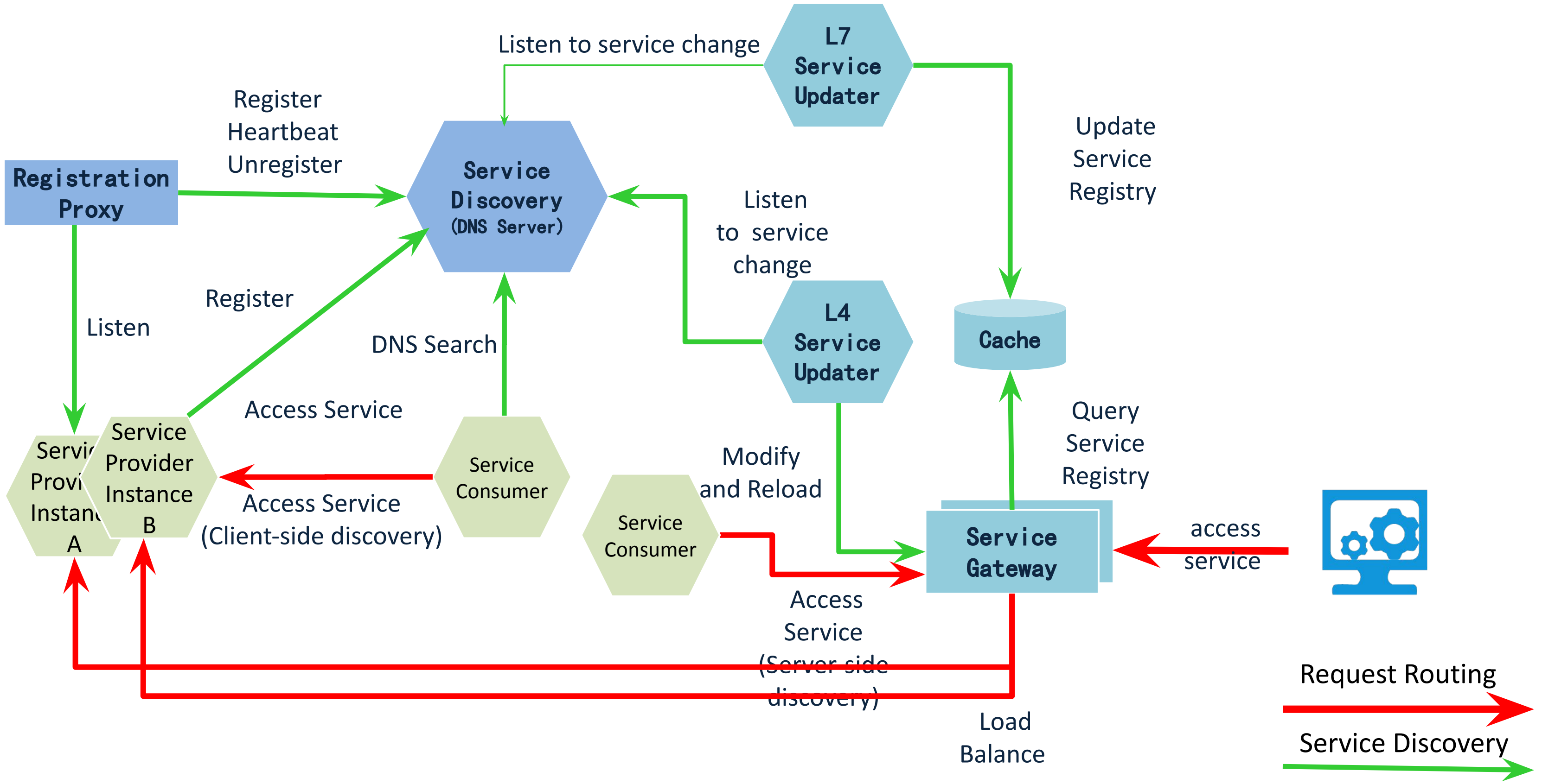
Centralized in the Auth Service

Note: Auth Service is not in the scope of MSB

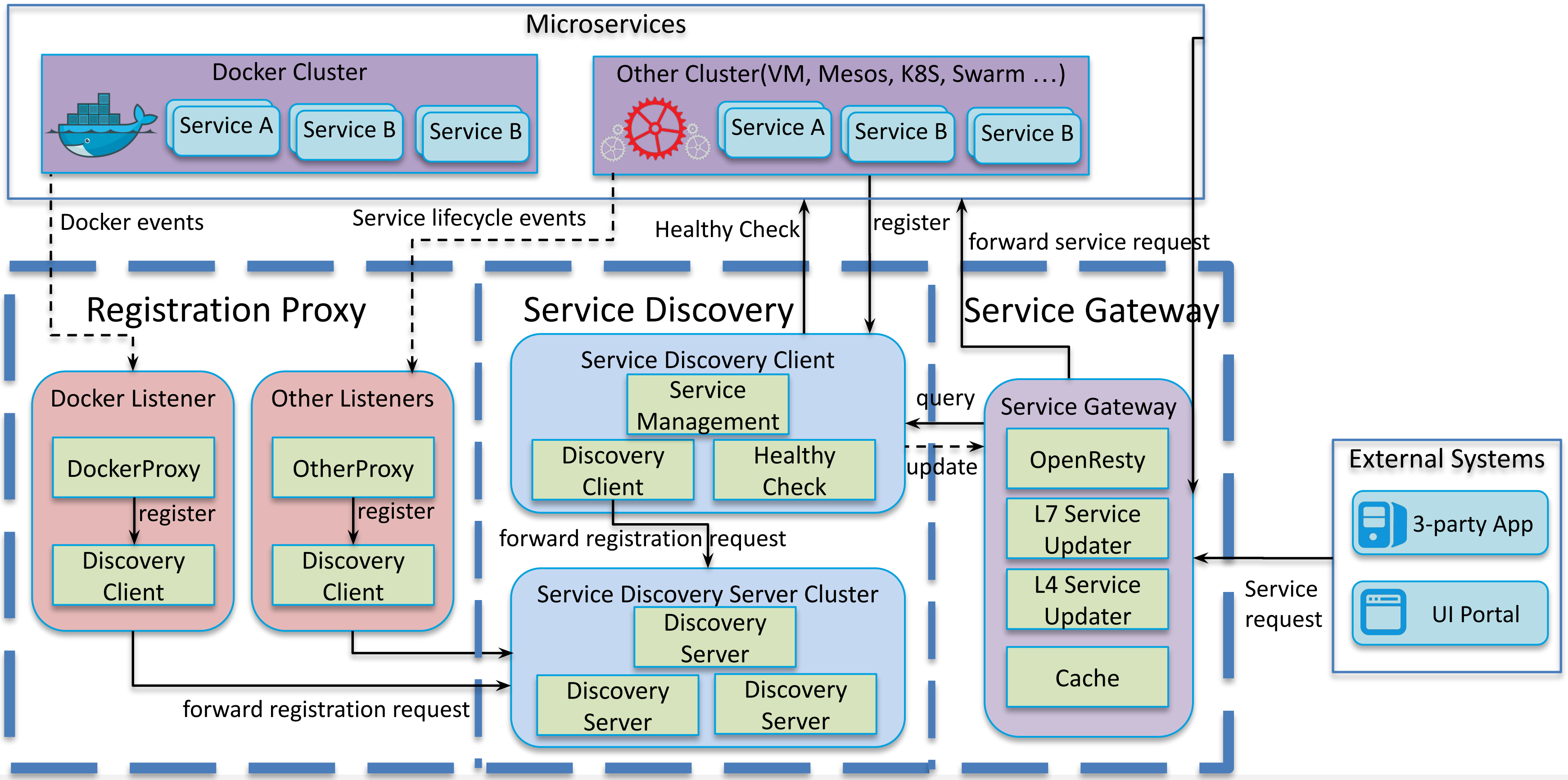
Agenda

- Current Challenges and MSB Solutions
- MSB Architecture & Features
- API & Example

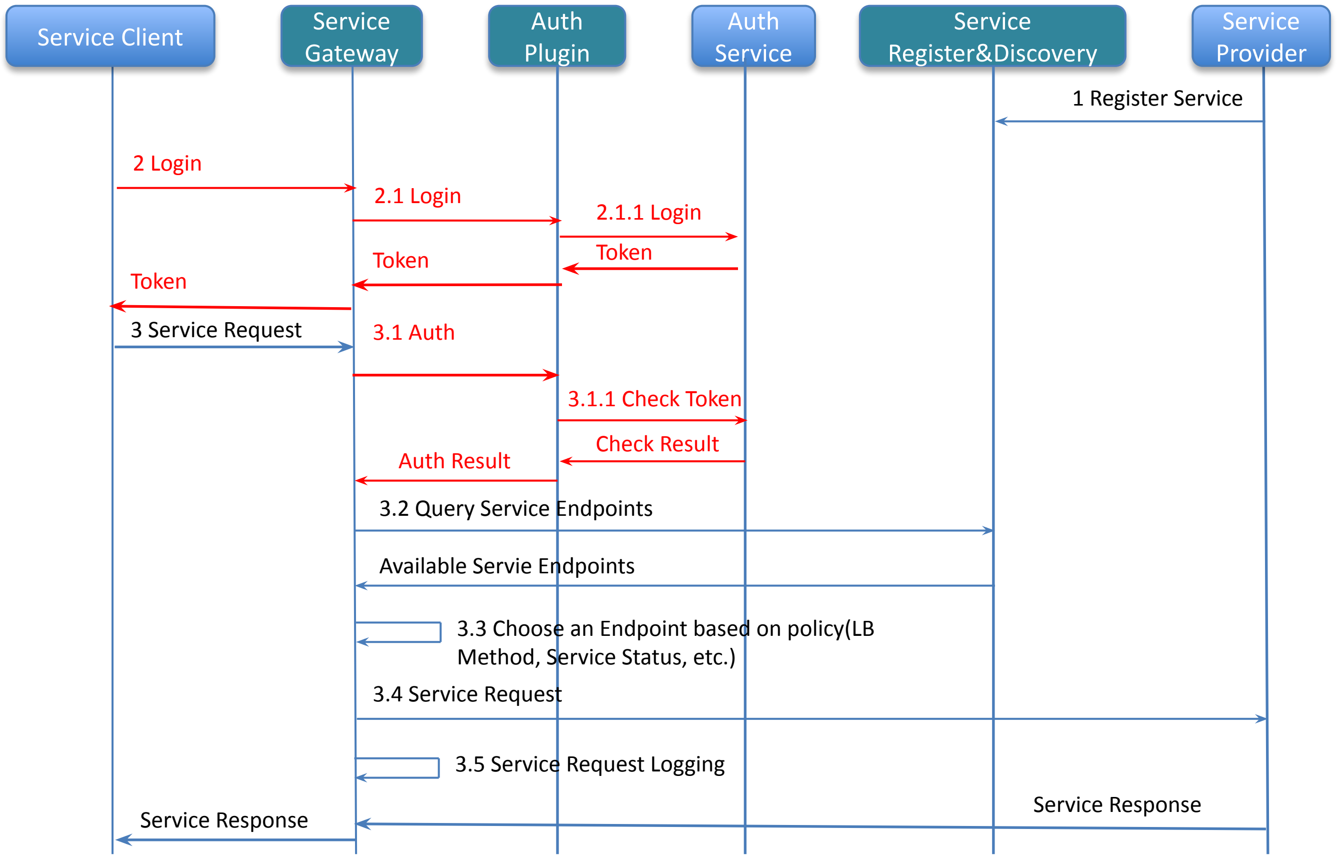
OPEN-O Microservice Solution: High Level Architecture



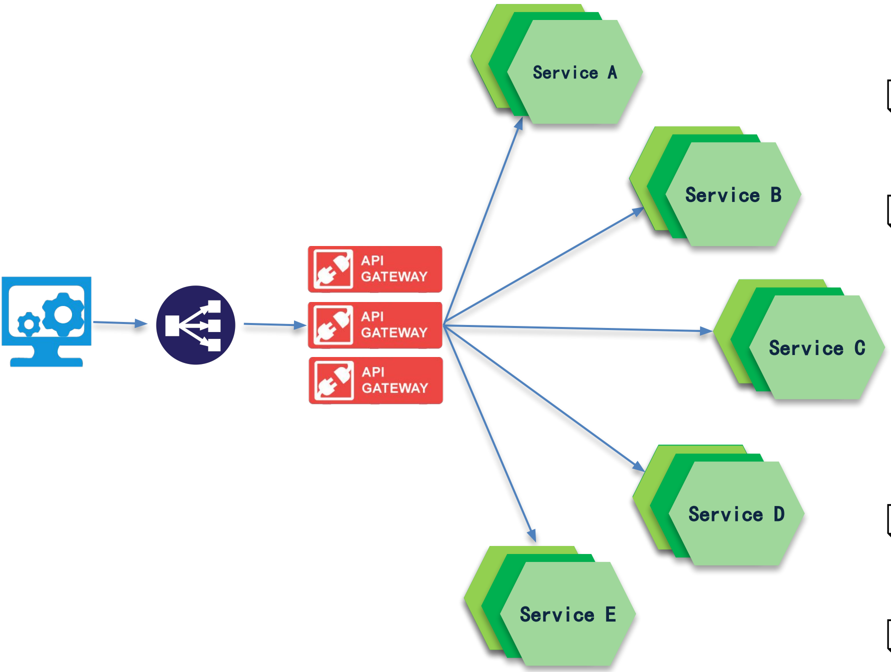
OPEN-O Microservice Solution : MSB Components



Service Request Sequence Diagram



MSB Features-High Availability



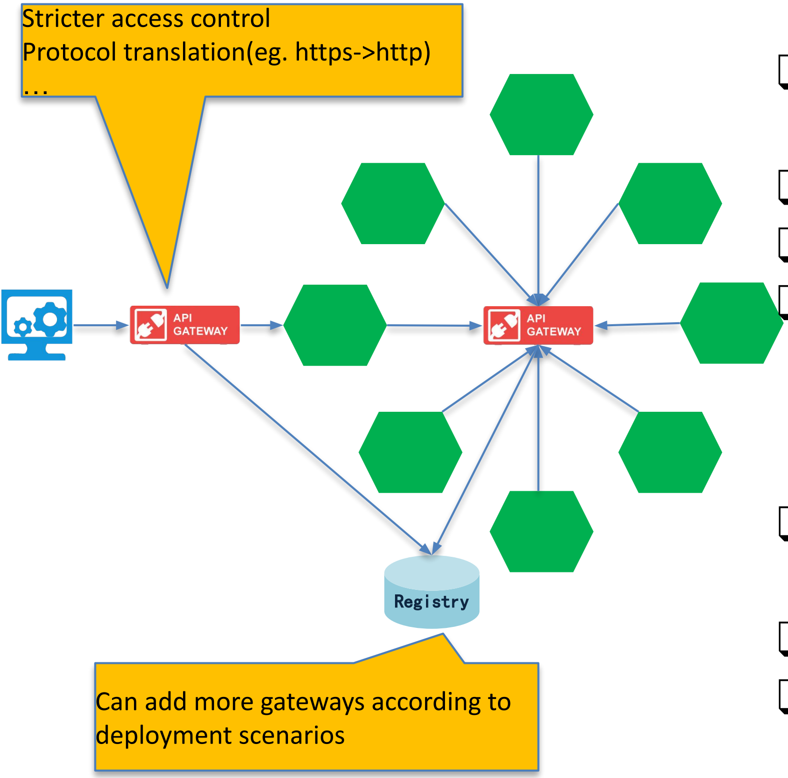
Access Layer

- ❑Load balancer(DNS Server/LVS etc.) in the front end
- ❑Service gateway cluster to avoid SPOF of service gateway

Service Layer

- ❑Service gateway as the load balancer for services
- ❑Deploy multiple service instances to avoid SPOF of service

MSB Features-Separated gateway for External and Internal Routing


Stricter access control
Protocol translation(eg. https->http)
…
API GATEWAY
API GATEWAY
Registry
Can add more gateways according to deployment scenarios

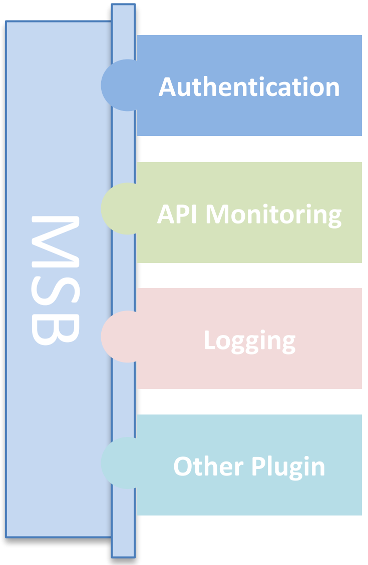
External service gateway

- ❑Expose the services(Rest API, UI pages, etc.)which need to be accessed by external systems
- ❑Solve the cross-domain issue for web app
- ❑Stricter access control
- ❑Adaption between external API and internal service

Internal API gateway (router)

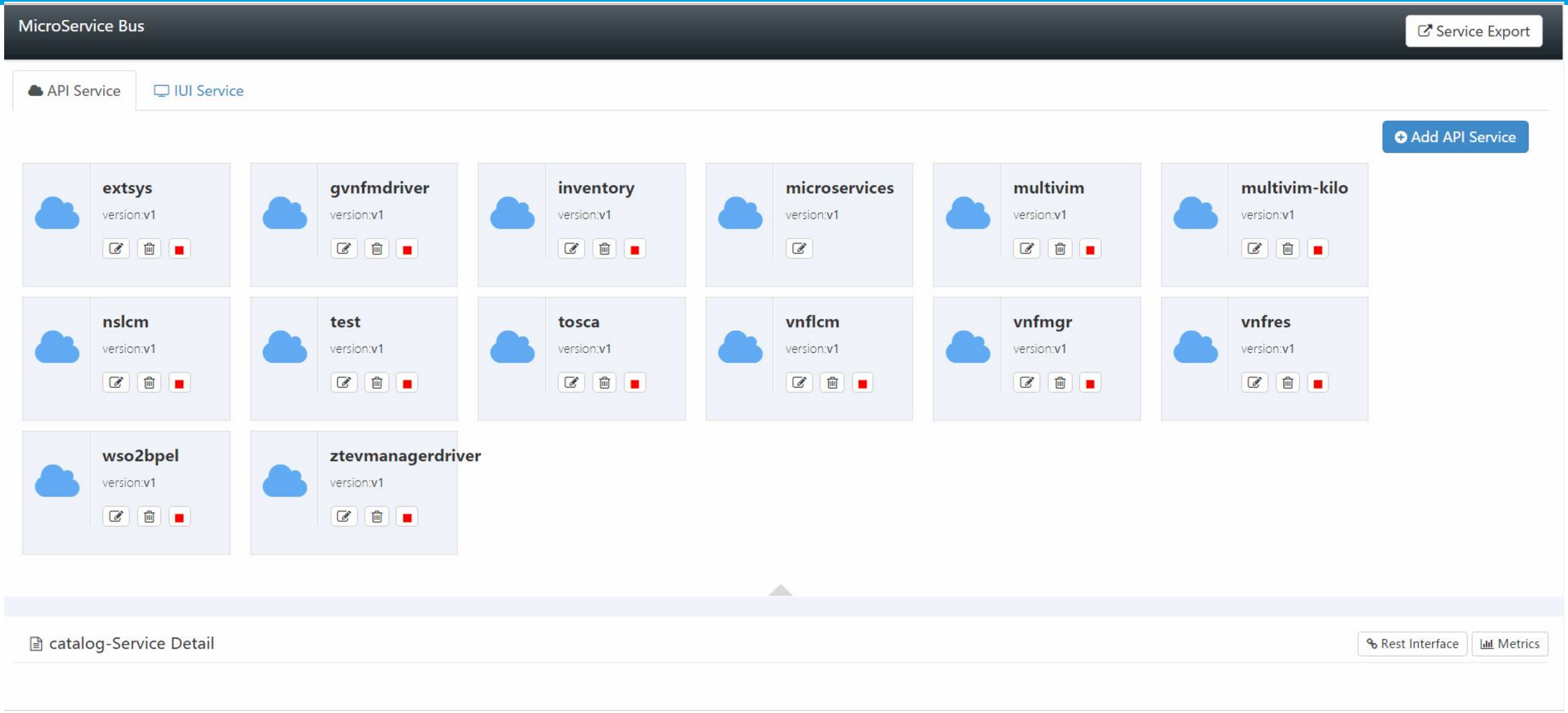
- ❑Routing and load balancing of the API calls within the system
- ❑Less control in trusted zone
- ❑Light weight communication protocol

MSB Features-Extendability



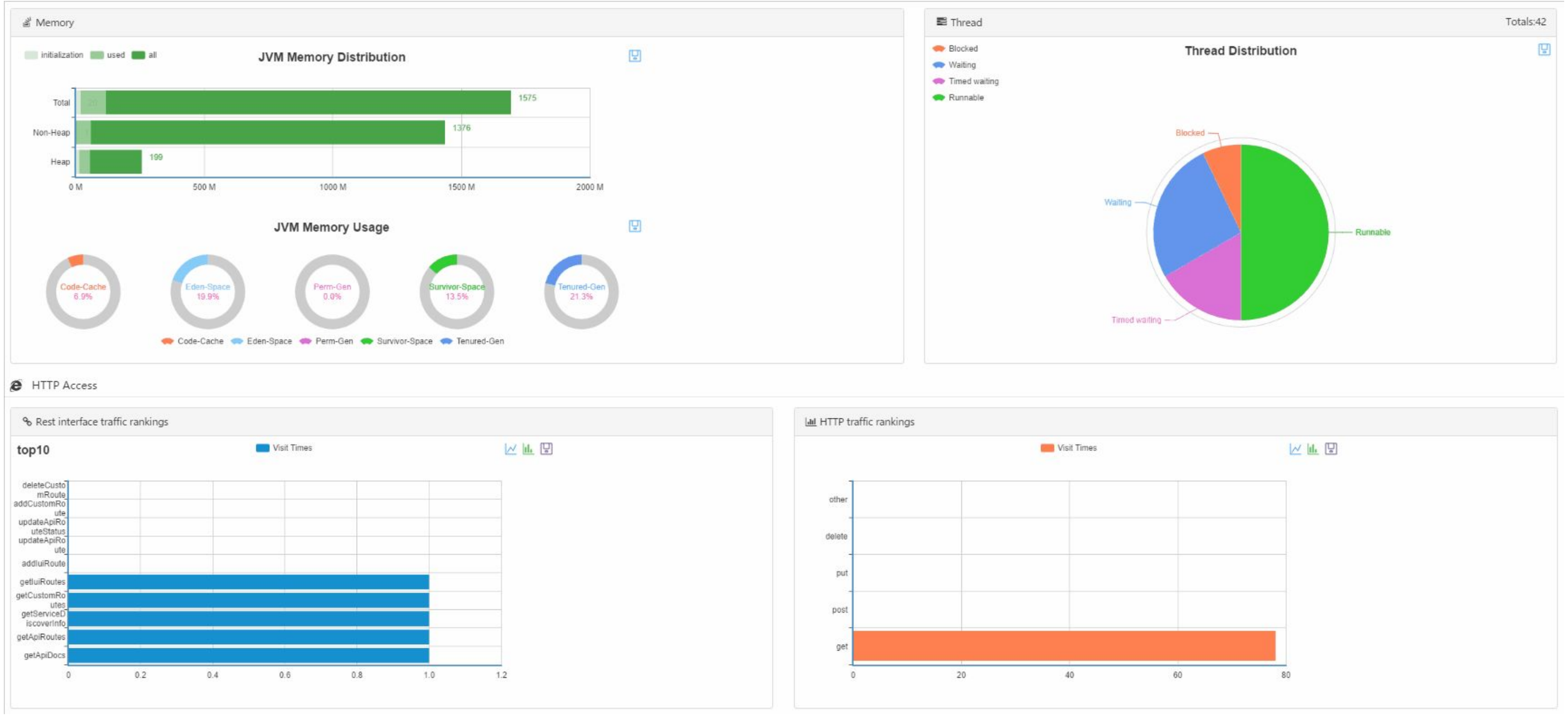
- Extendable architecture for adding functionality
  - Auth: add auth to APIs, integrated with Openstack keystone
  - Driver routing: add driver specify routing logic for devices
  - Logging: API calling logging
  - Service health monitoring
  - ACL,API Analytics,Transformations
  - Anything: new functionality can be added on demand by plugins

MSB Features-Service API Portal

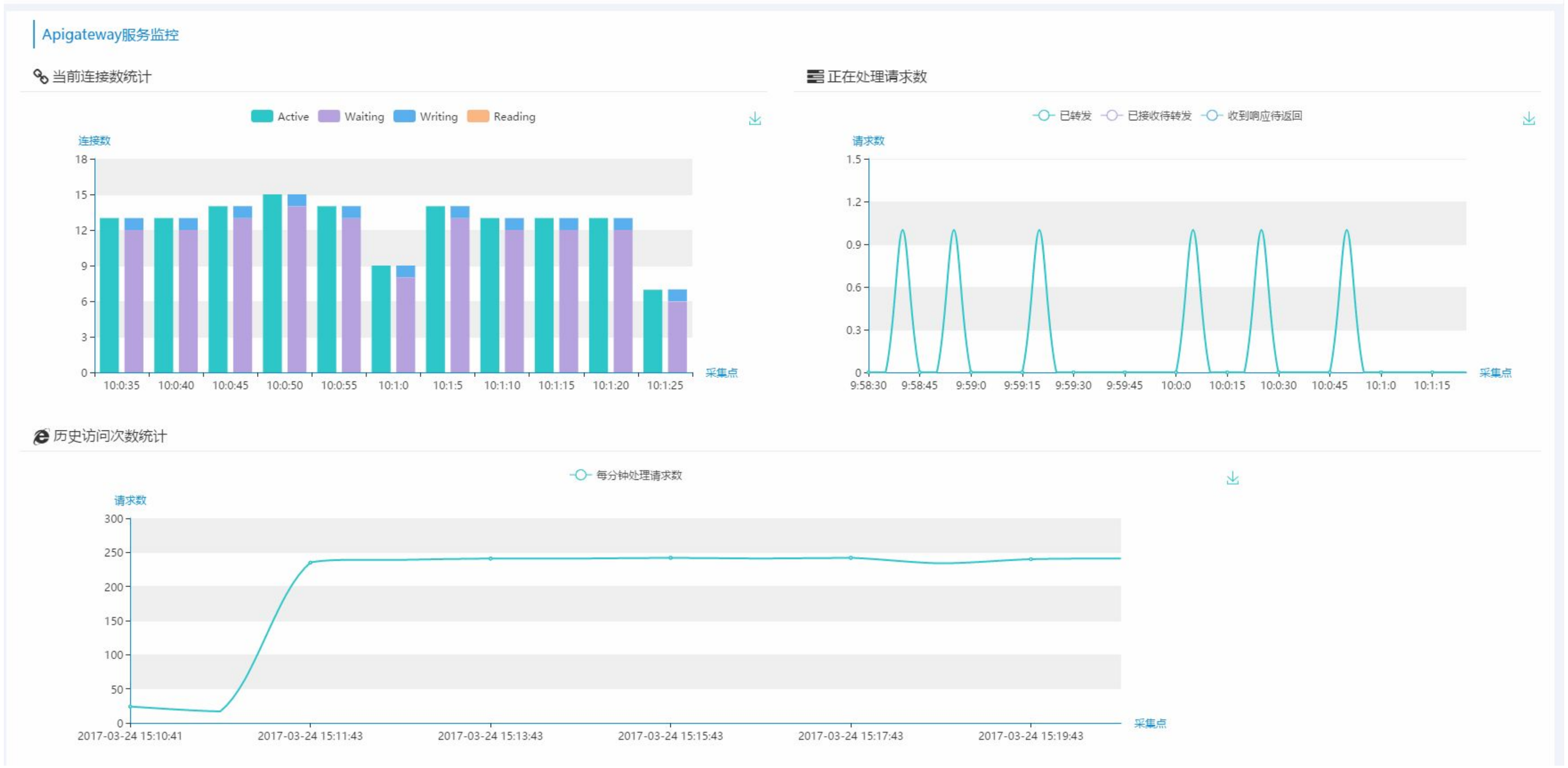


extsys version:v1
gvnfmdriver version:v1
inventory version:v1
microservices version:v1
multivim version:v1
multivim-kilo version:v1
nslcm version:v1
test version:v1
tosca version:v1
vnflcm version:v1
vnfmgr version:v1
vnfres version:v1
wso2bpel version:v1
ztevmanagerdriver version:v1

MSB Features-Service Healthy Monitoring



MSB Features-API Monitoring



Agenda

- Current Challenges and MSB Solutions
- MSB Architecture & Features
- API & Example

Quick Example

- Start MSB using docker

```
sudo docker run -p 80:80 -d --name msb openoint/common-services-msb
```

- Register service

```
curl -X POST \
-H "Content-Type: application/json" \
-d '{"serviceName": "weather", "version": "v1", "url": "/openoapi/weatherexample", "protocol":
    "REST", "nodes": [ {"ip": "10.0.2.15","port": "9090", "ttl": 0}]}' \
"http://127.0.0.1:80/openoapi/microservices/v1/services"
```

- Make request

```
curl -i -X GET \
http://127.0.0.1/openoapi/weather/v1/Middletown
```

MSB Resource Address Specification

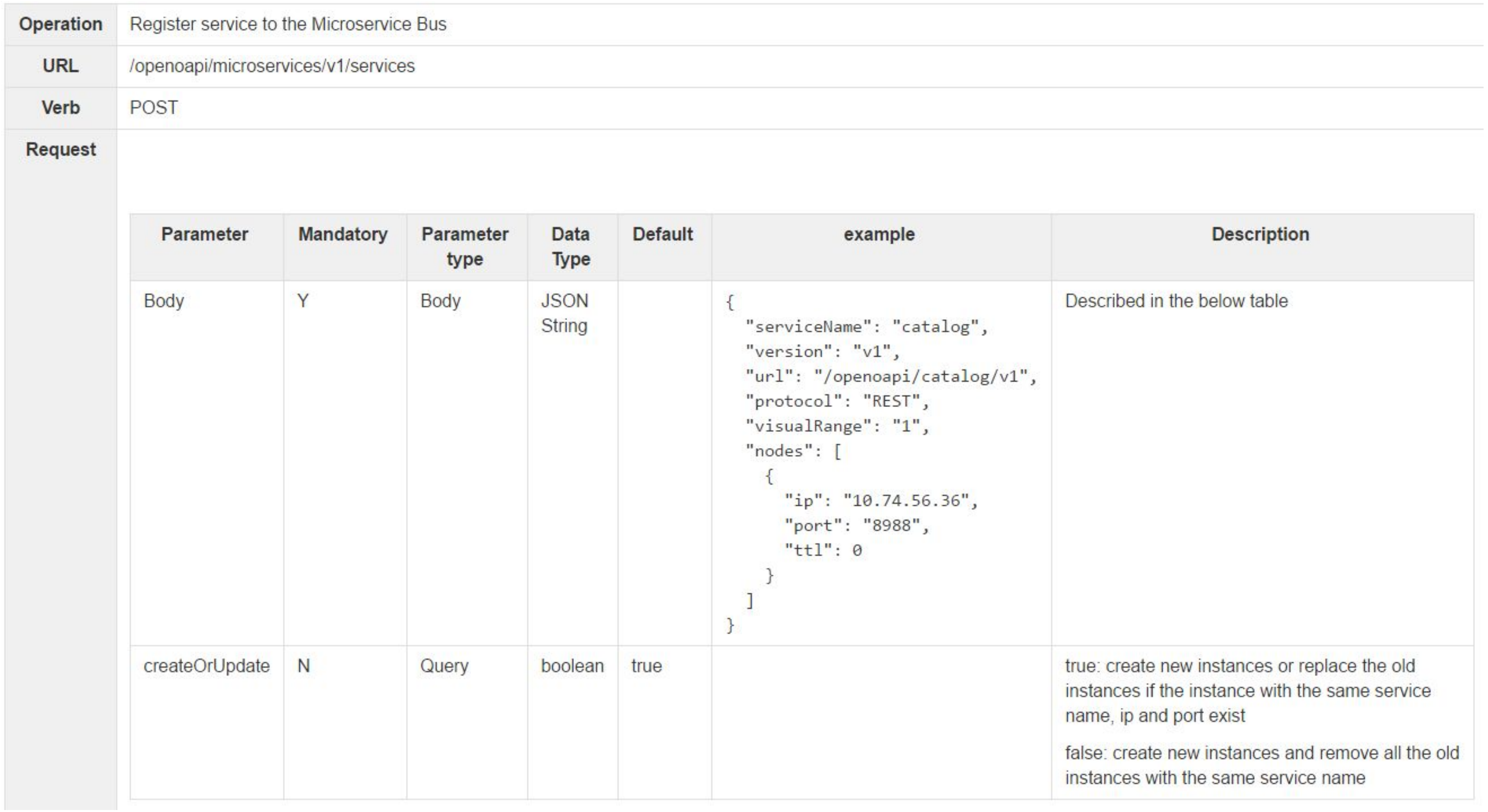
Service type	Type	Query String
API Service Specification	[host]:[port]/openoapi/[ServiceName]/[ServicesVersion]/[PathInfo]	queryparam1=xxx, queryparam2=xxx
Content Service Specification	[host]:[port]/openoui/[PathInfo]	None

Openoapi and openoui could be modified to api and ui

Attribute	Type	Description
ServiceName	String	A unique name for the service. For GSO, SDNO and NFVO, service name should include the project name as well as the microservice name to ensure uniqueness, example: 'sdno-l3vpnService' For O-Common and Common-Tosca, the project name is not necessary in the service name, example: 'catalog’
ServicesVersion	String	The version of service, the version should begin with ‘v’, plus a number or major version number period minor version number
PathInfo	String	Path information for the resource

Example :
log API Service http://127.0.0.1/openoapi/log/v1/syslogs?id=101&filter=admin&count=50
UI Service http://127.0.0.1/openoui/log/index.html

Service Registration API



Operation	Register service to the Microservice Bus
URL	/openoapi/microservices/v1/services
Verb	POST
Request	

Parameter	Mandatory	Parameter type	Data Type	Default	example	Description

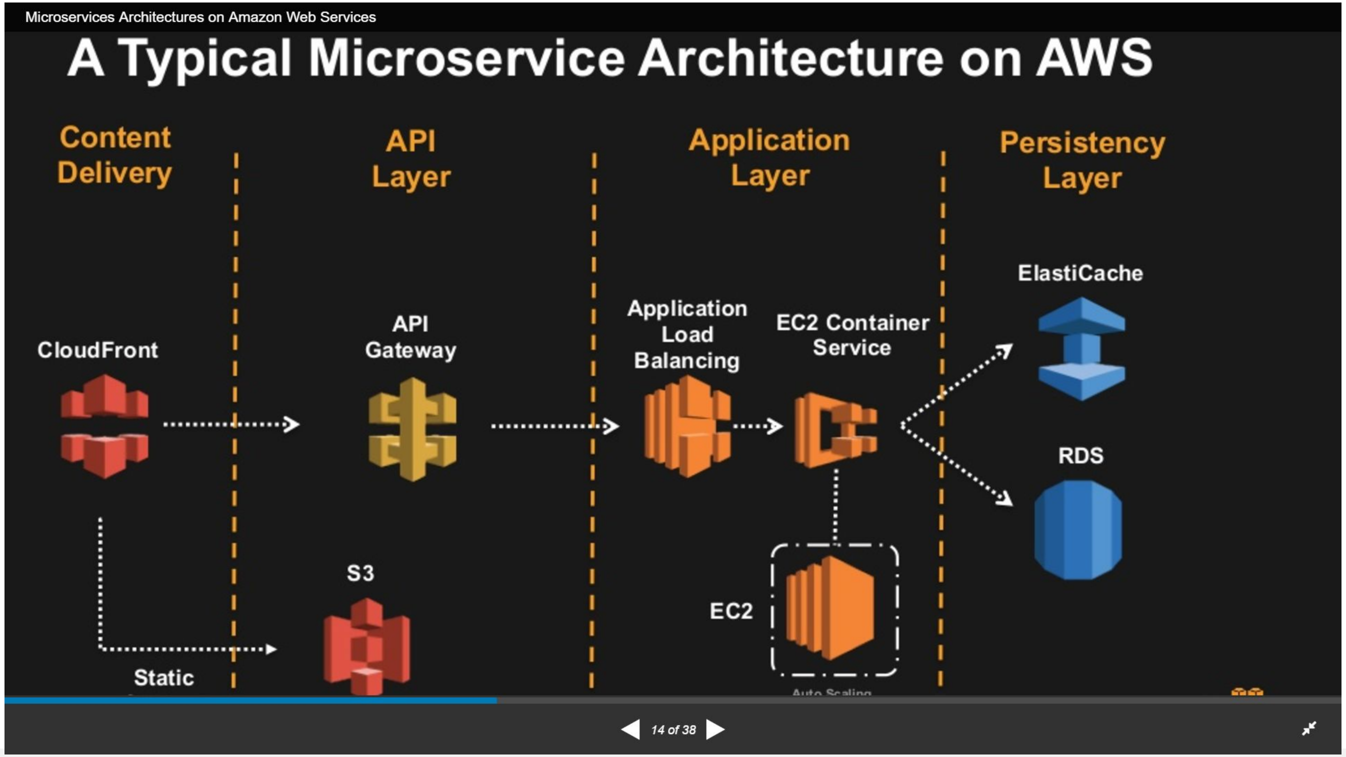
Body	Y	Body	JSON String		`{`
					`  "serviceName": "catalog",`
					`  "version": "v1",`
					`  "url": "/openoapi/catalog/v1",`
					`  "protocol": "REST",`
					`  "visualRange": "1",`
					`  "nodes": [`
					`    {`
					`      "ip": "10.74.56.36",`
					`      "port": "8988",`
					`      "ttl": 0`
					`    }`
					`  ]`
					`}`
						Described in the below table

createOrUpdate	N	Query	boolean	true		true: create new instances or replace the old instances if the instance with the same service name, ip and port exist false: create new instances and remove all the old instances with the same service name

AWS Microservice Architecture Reference1

2016.9: Microservices Architectures on Amazon Web Services

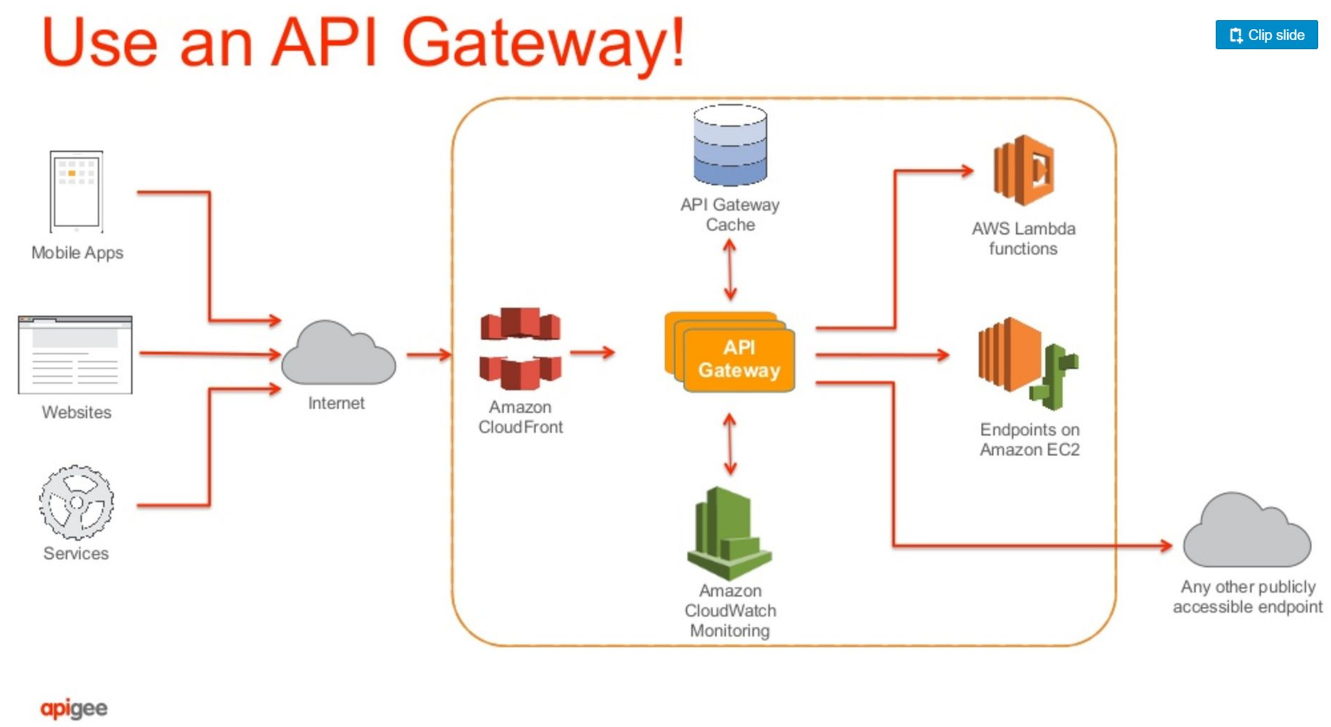
Adam Lynch – Snr. Technical Account Manager refer link



A Typical Microservice Architectur e on AWS S3 CloudFront EC2 Application Load Balancing Static Content Content Delivery API Layer Application Layer Persistency Layer API Gateway EC2 Cont Service Auto Scaling Group DynamoDB

AWS Microservice Architecture Reference2

I Love APIs 2015: Microservices at Amazon
Chris Munns, Amazon – AWS Solution Architect refer link



Establishing a pattern for services and clients It's important that the organization isn't reinventing the wheel on every new service: • How are clients going to communicate? • What cross service authorization requirements are there? • How do services prevent abuse? • How do you quickly build clients against a service? • How do services handle discovery of others services and resources?
Use an API Gateway! Internet Mobile Apps Websites Services API Gateway AWS Lambda functions API Gateway Cache Endpoints on Amazon EC2 Any other publicly accessible endpoint Amazon CloudWatch Monitoring Amazon CloudFront

ONAP
OPEN NETWORK AUTOMATION PLATFORM

Thank You

www.onap.org