



ONAP

OPEN NETWORK AUTOMATION PLATFORM

Microservice Powered Orchestration

Huabing Zhao ZTE, System Engineer, Network Management & Service, OPEN-O Common Service PTL

zhao.huabing@zte.com.cn

Zhaoxing Meng ZTE, NFV&SDN Architect, Network Management & Service, OPEN-O Common Service PTL

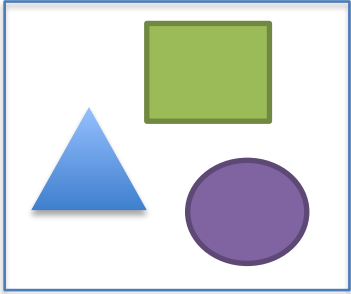
zhaoxing.meng1@zte.com.cn

Agenda

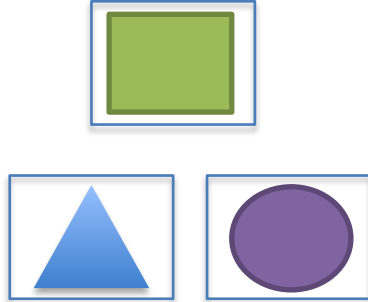
- Why Microservice at OPEN-O
- Challenges of Microservice
- MSB(Microservice Bus) Solution
- What can MSB bring to ONAP

Monolith vs Microservice

Monolith



Microservices

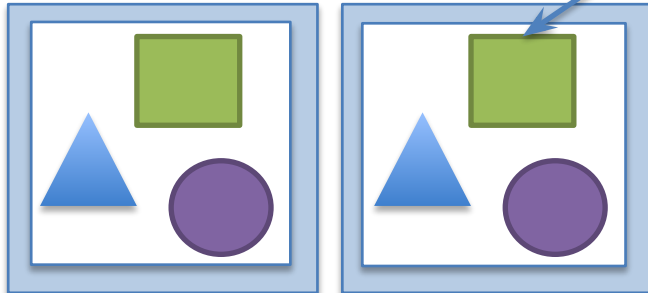


The microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API.

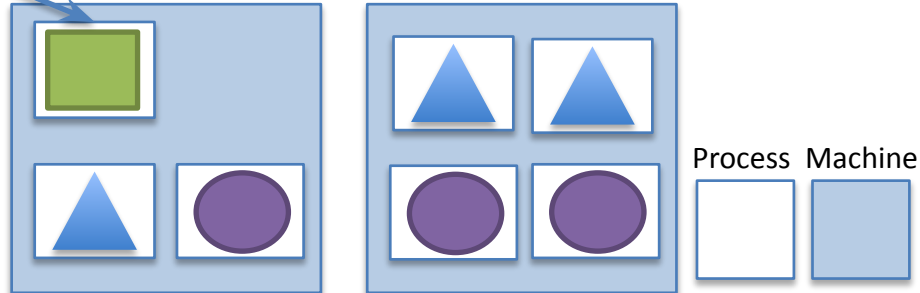
-Martin Fowler

No performance issue here

Scaling Monolith



Scaling Microservices



Why chose Microservice Architecture?



How to integrate existing seed codes in different technical stack?

- We didn't start from scratch
- A dozen of existing seed codes repos
- Ambitious release plan



How to build an OPEN community?

- We have various members and we are expecting more joining in
- Each organization has its own tech Stack

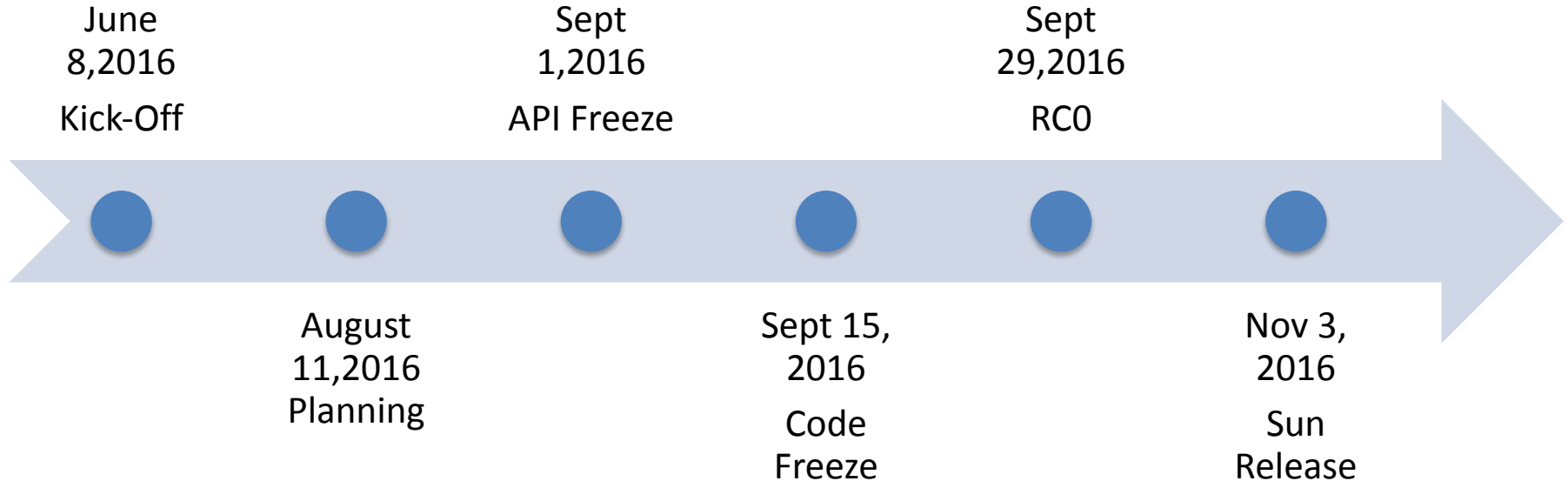


How to make orchestration reliable and scalable?

- OPEN-O is a large, complex software system
- Each component may have different resource requirement
- Each component may have different working load

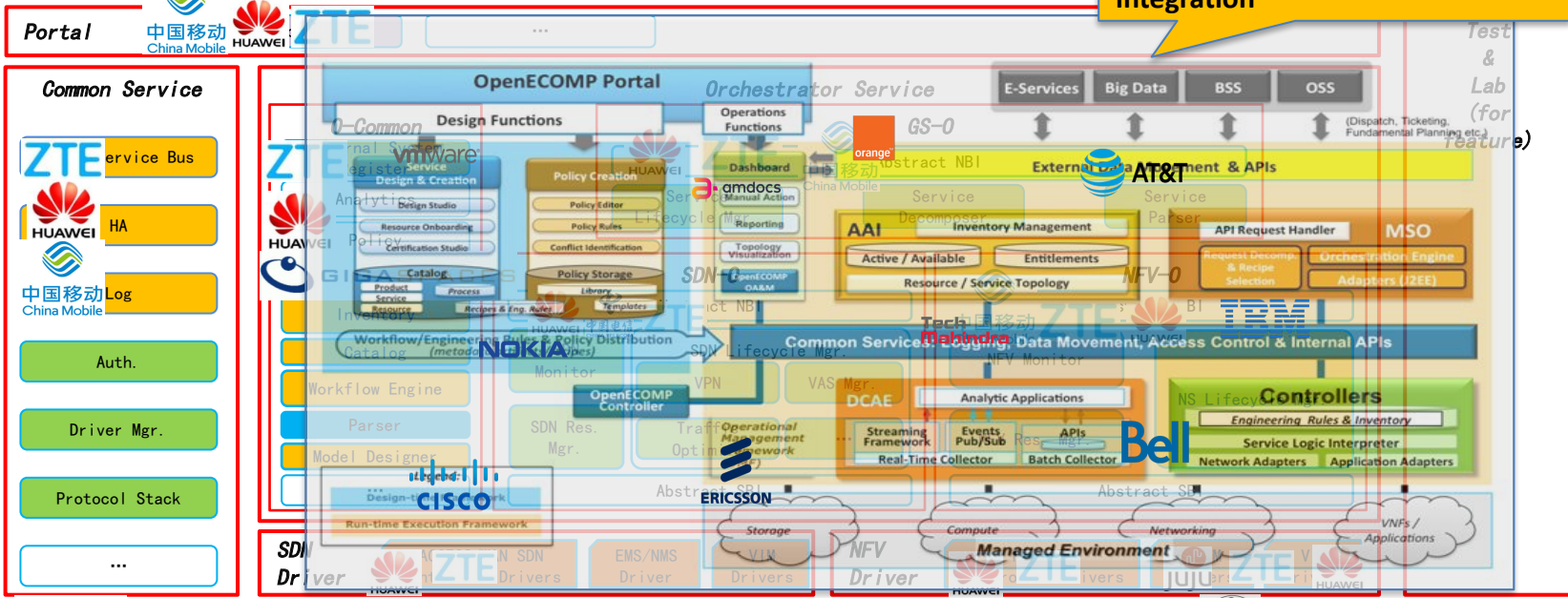
Ambitions Sun Release Plan

We had made an ambitions plan for SUN Release



Challenge of Integration

We get bigger challenge for ONAP integration

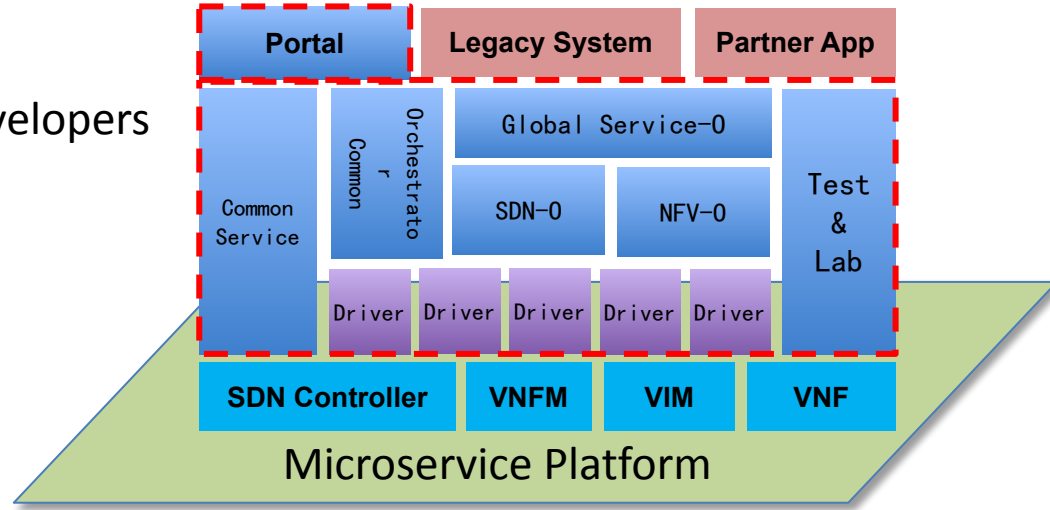
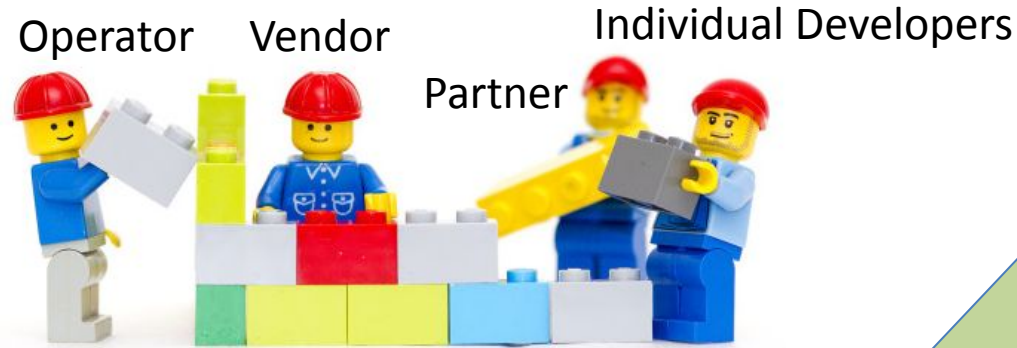


Common Service

- ZTE Service Bus
- HUAWEI HA
- 中国移动 China Mobile Log
- Auth.
- Driver Mgr.
- Protocol Stack
- ...



Build an Open Community



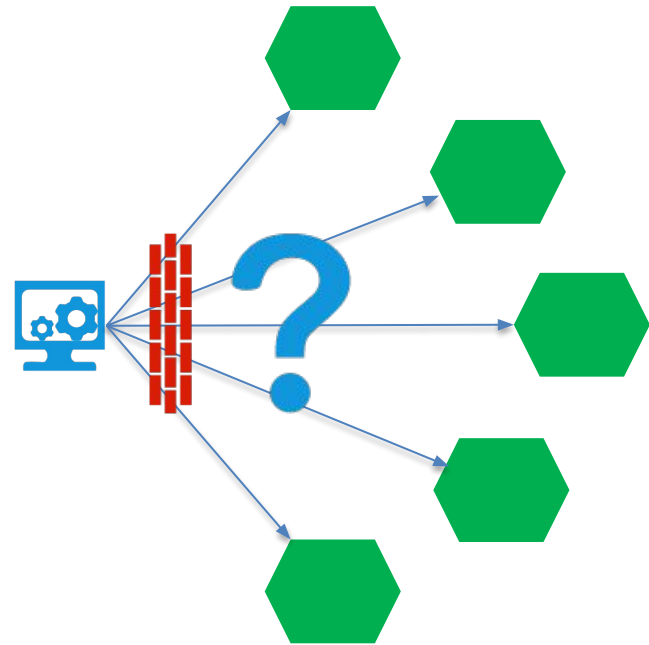
Build an open community so that everyone can enjoy the party

Challenges of Microservice Architecture

Microservice Architecture comes at a price: Complexity

- ? How do the clients application access the back end services?**
- ? How do the client or another service - discover the location of a service instance?**

Direct Client-to-Microservice Communication ?



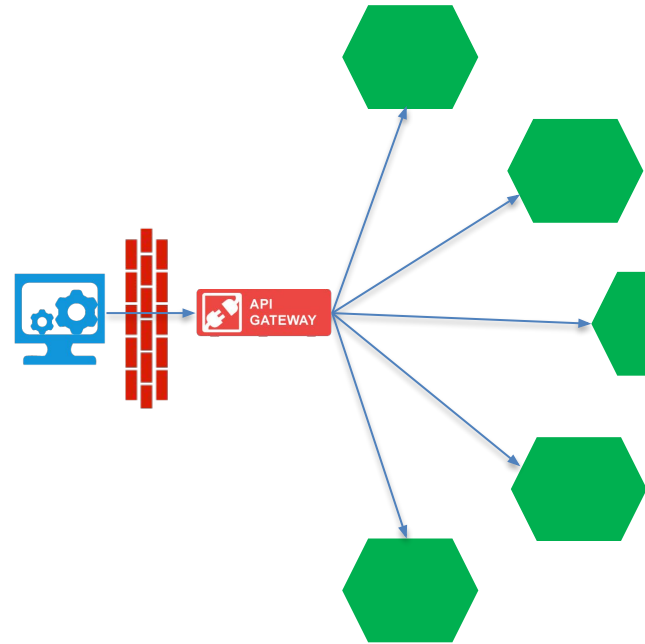
This approach has some problems:

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

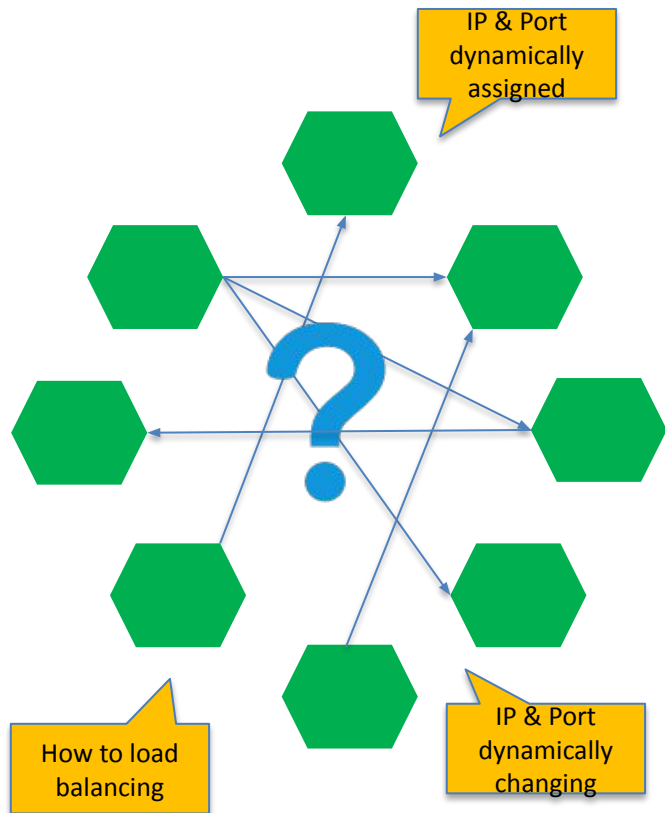
Solution: Service Gateway

Service gateway hides the complexity

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



How to find the service?



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

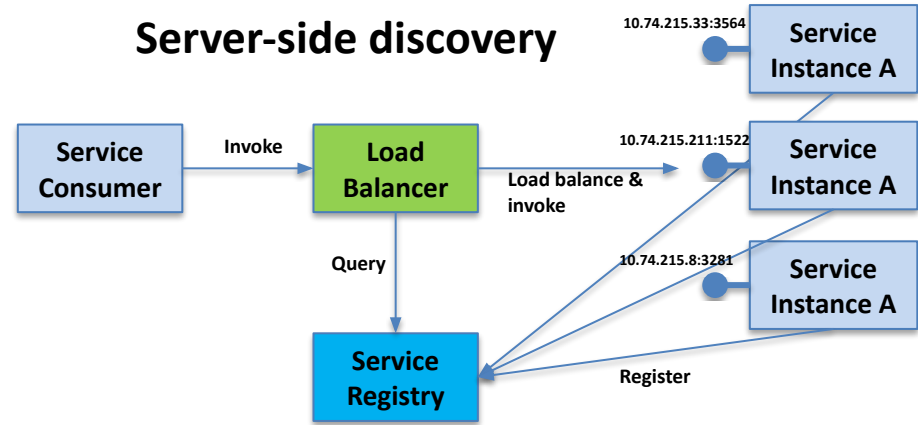
Service Registration:

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

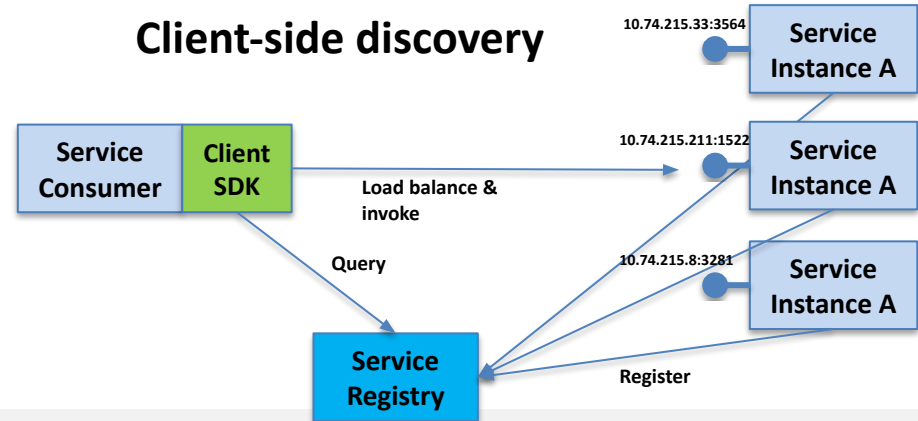
Service Discovery:

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

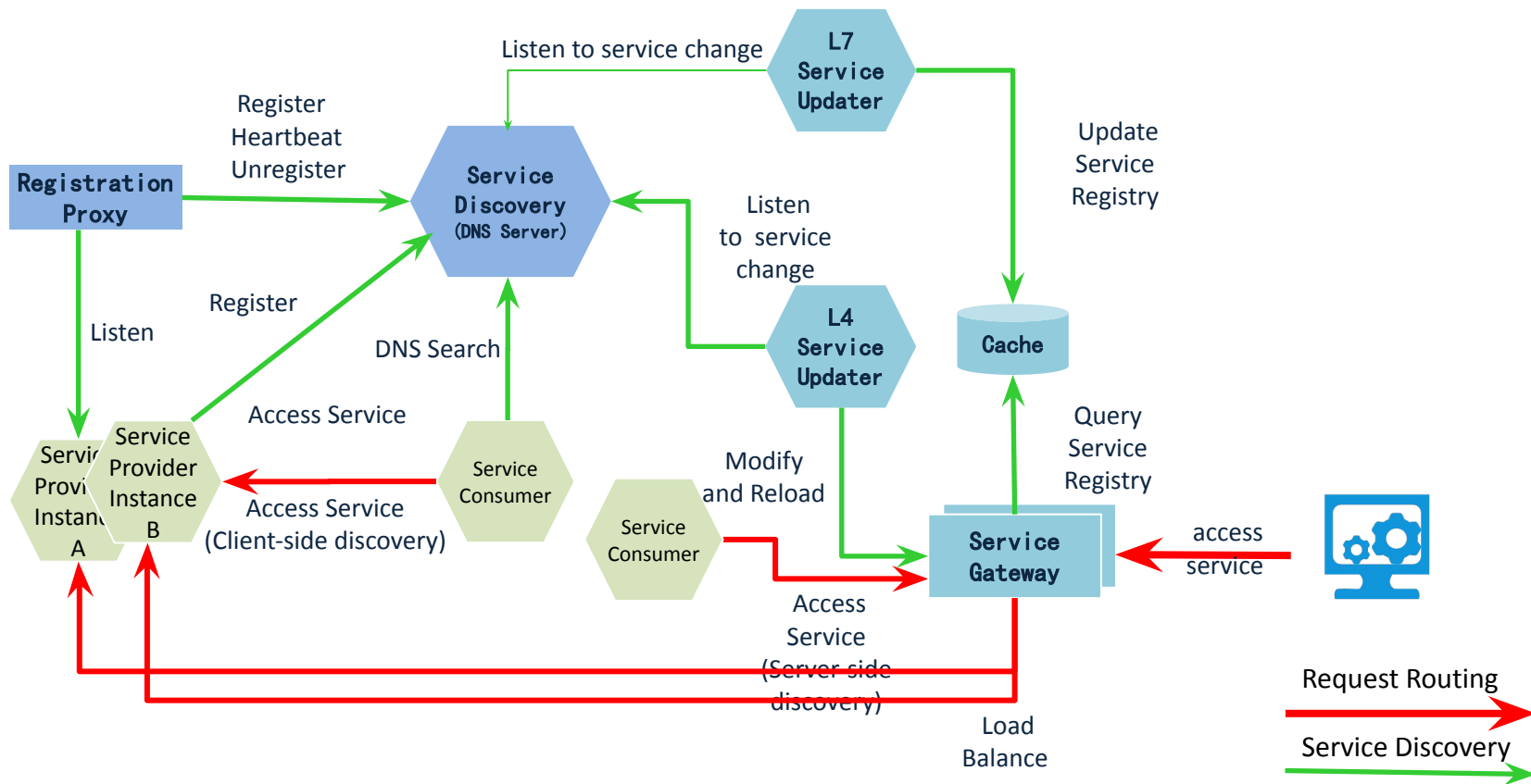
Server-side discovery



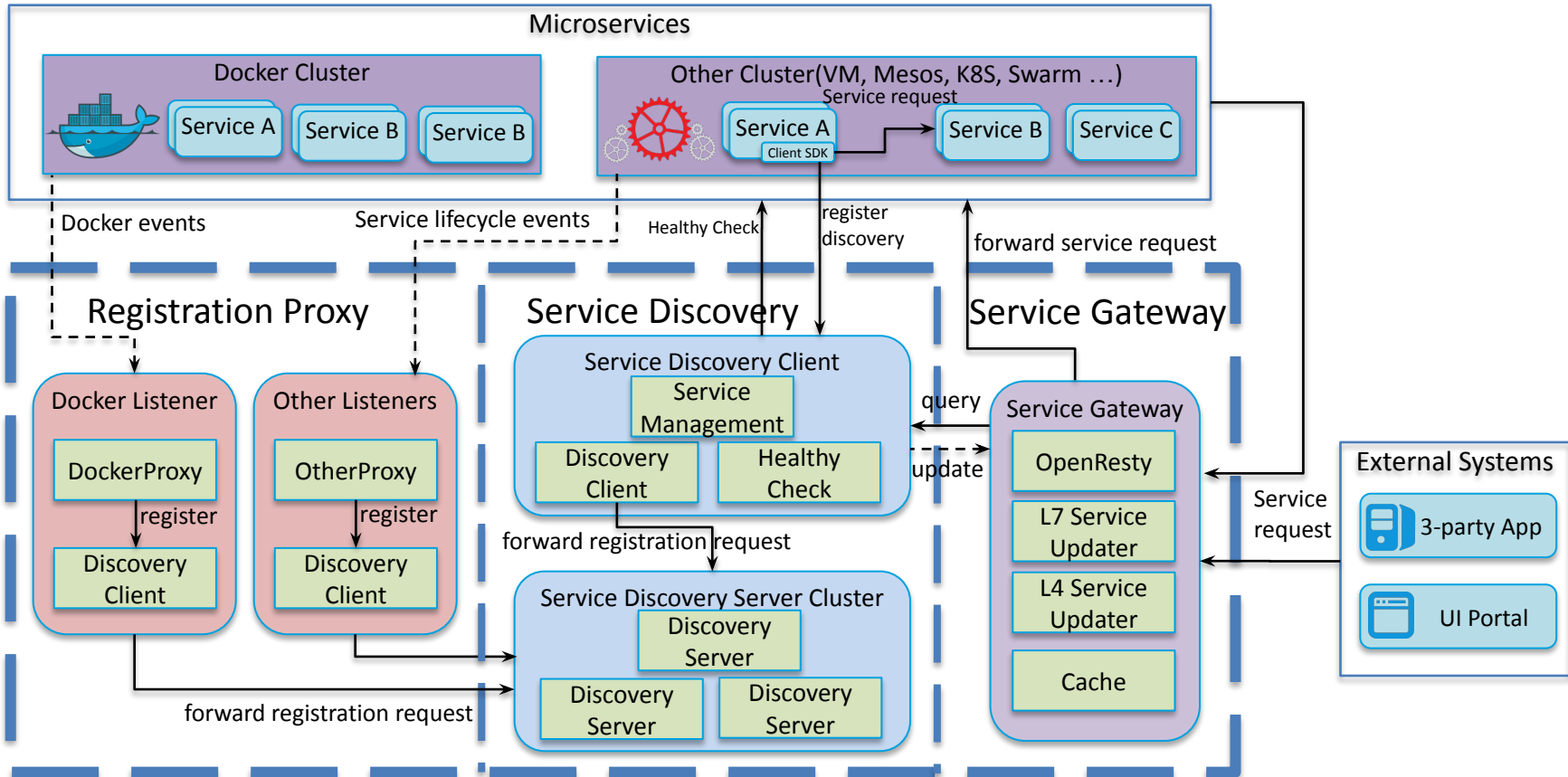
Client-side discovery



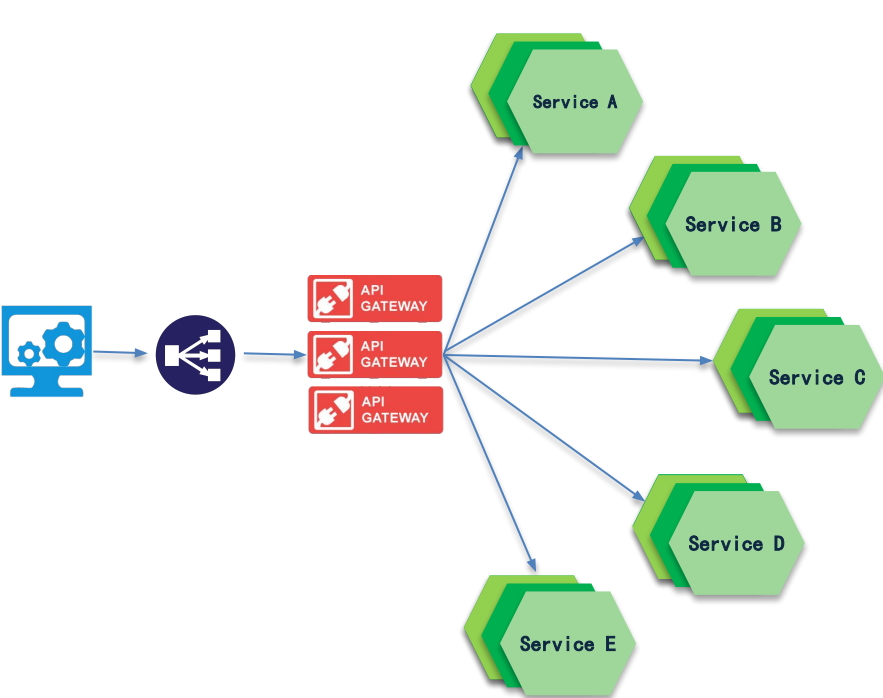
OPEN-O Microservice Solution: High Level Architecture



OPEN-O Microservice Solution : MSB Components



MSB Features-High Availability



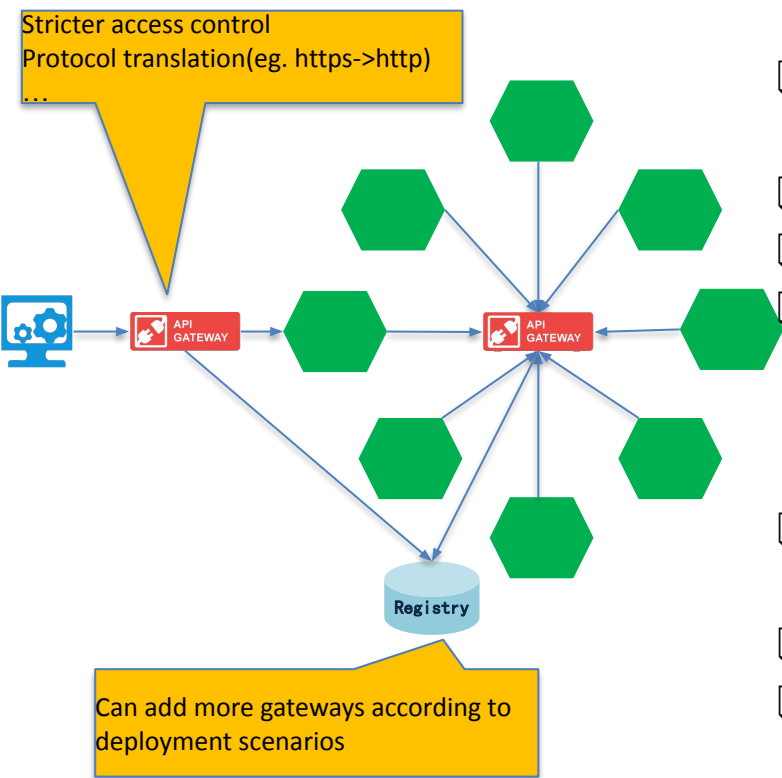
Access Layer

- ❑ 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



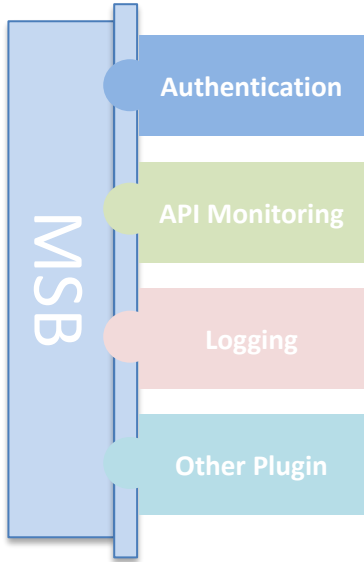
External service gateway

- ❑ 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


MicroService Bus

[Service Export](#)

API Service


IUI Service

[+ Add API Service](#)

 **extsys**
version:v1
[edit](#) [delete](#) [status](#)


 **gvnfmdriver**
version:v1
[edit](#) [delete](#) [status](#)


 **inventory**
version:v1
[edit](#) [delete](#) [status](#)

 **microservices**
version:v1
[edit](#)

 **multivim**
version:v1
[edit](#) [delete](#) [status](#)


 **multivim-kilo**
version:v1
[edit](#) [delete](#) [status](#)

 **nslcm**
version:v1
[edit](#) [delete](#) [status](#)

 **test**
version:v1
[edit](#) [delete](#) [status](#)


 **tosca**
version:v1
[edit](#) [delete](#) [status](#)

 **vnflcm**
version:v1
[edit](#) [delete](#) [status](#)

 **vnfmgr**
version:v1
[edit](#) [delete](#) [status](#)

 **vnfres**
version:v1
[edit](#) [delete](#) [status](#)

 **wso2bpel**
version:v1
[edit](#) [delete](#) [status](#)

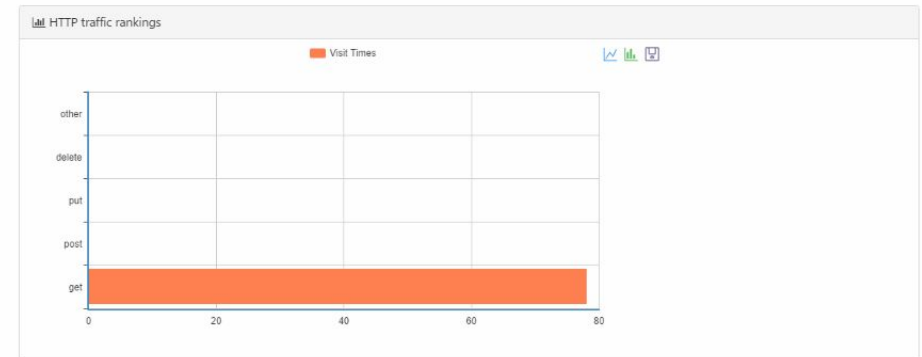
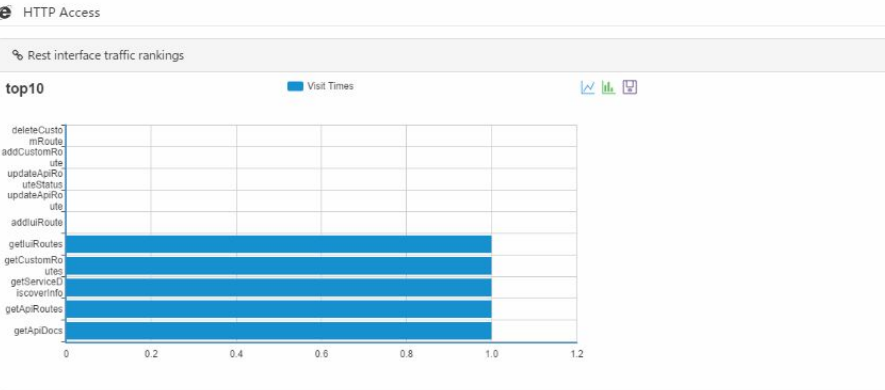
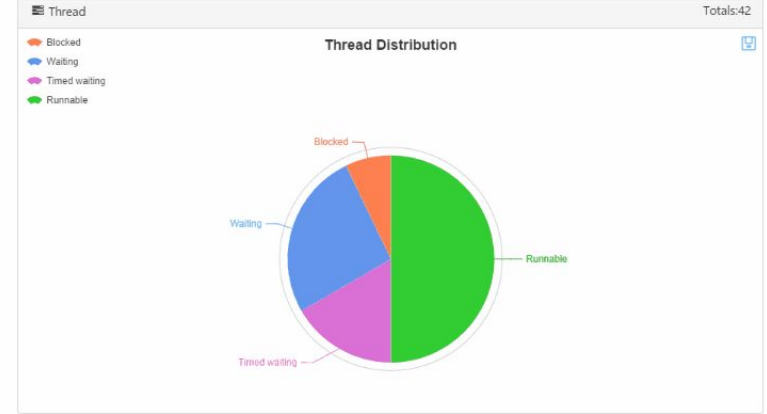
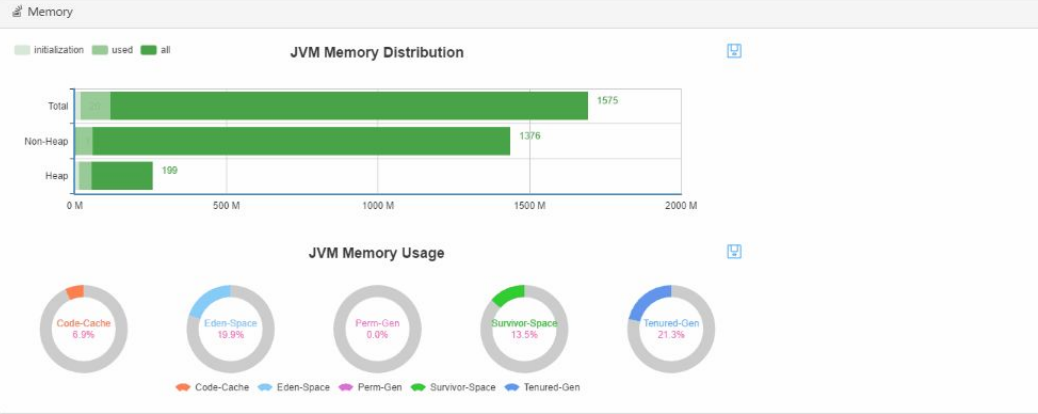
 **ztevmanagerdriver**
version:v1
[edit](#) [delete](#) [status](#)

[catalog-Service Detail](#)

[Rest Interface](#)

[Metrics](#)

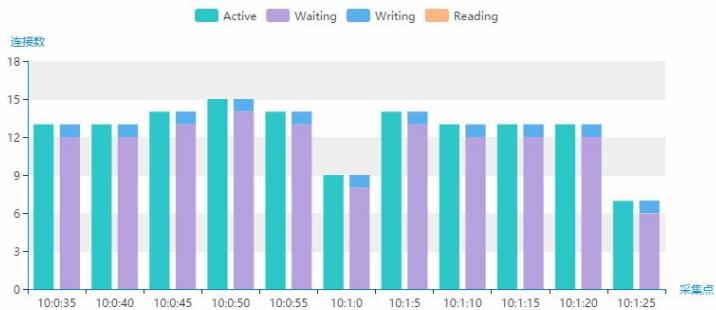
MSB Features-Service Healthy Monitoring



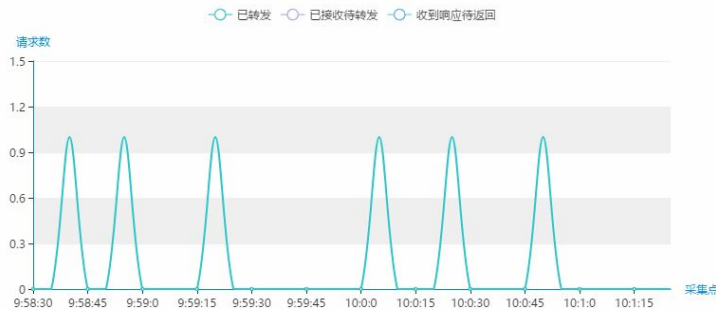
MSB Features-API Monitoring

Apigateway服务监控

当前连接数统计



正在处理请求数



历史访问次数统计



How MSB may fit into ONAP (Service Discovery & Routing)

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

Before:

```
"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**

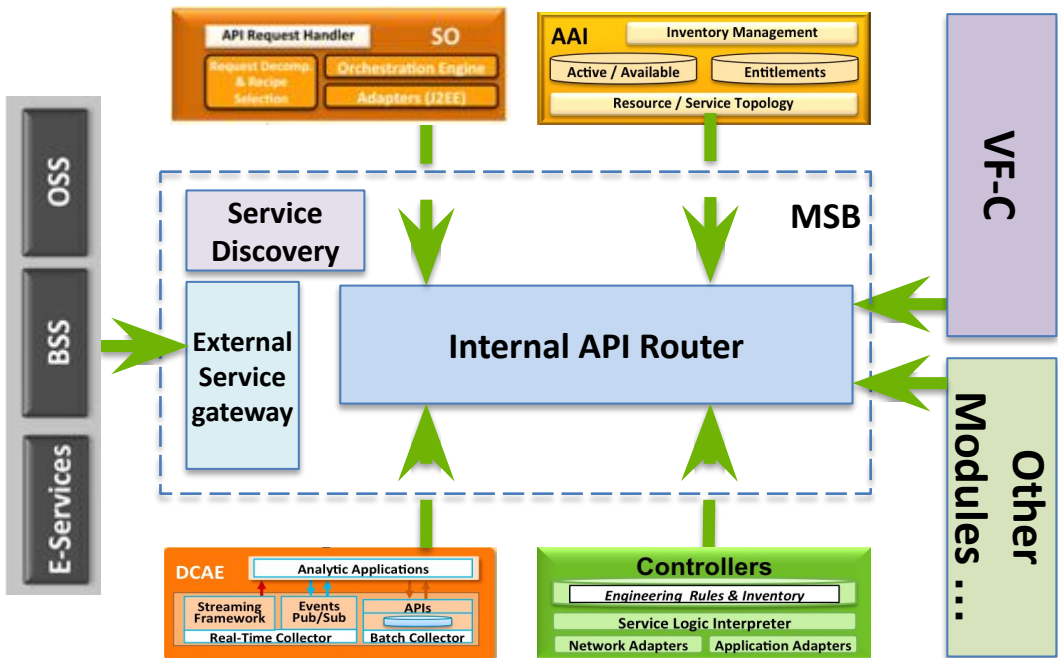
How to call service:

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

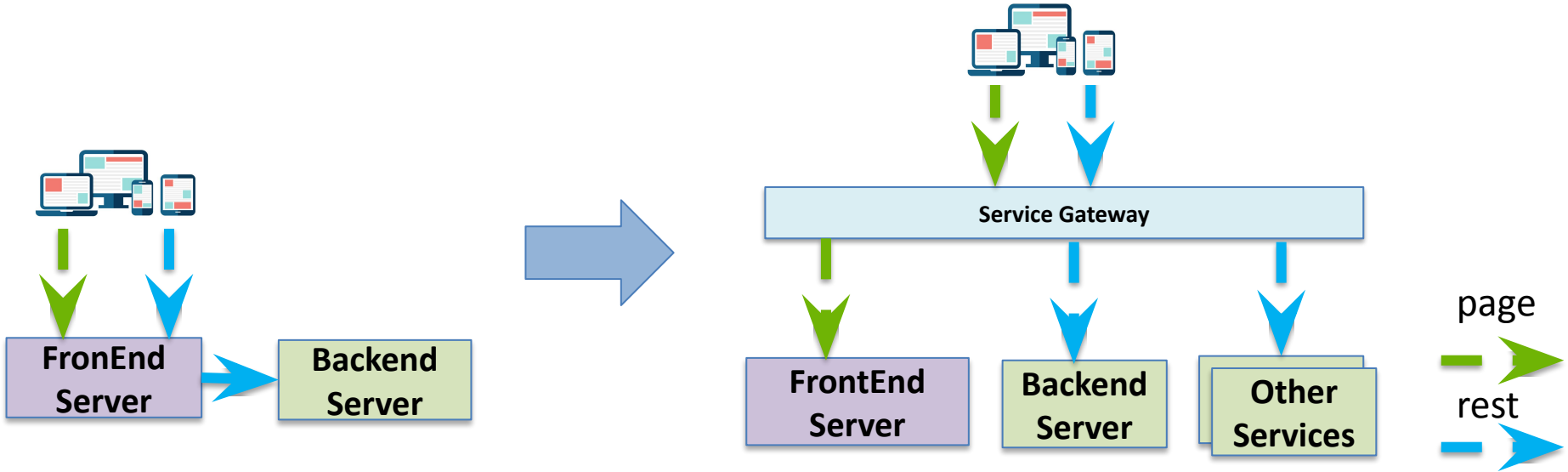
API gateway routes the request to:

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

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



How MSB may fit into ONAP(reverse proxy for web app)



Before:

- The business logic(rest service) forwarder 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



Thank You

www.onap.org