Microservice Bus Tutorial

Huabing Zhao, PTL of MSB Project, ZTE

Agenda

- ❑MSB Overview
- ❑Service Registration
- ❑Service Discovery
- ❑Example & Demo
- ❑Suggested Integration Approach
- ❑Future plan

MSB Overview-Introduction

Microservices Bus(MSB) provide a reliable, resilient and scalable communication and governance infrastructure to support Microservice Architecture including including service registration/discovery, external API gateway, internal API gateway, client SDK. It's a pluggable architecture so it can integrate with auth service provider to provide centralized Authentication & Authorization. MSB also provides a service portal to manage the REST APIs.

MSB doesn’t depend on a specific environment. It can work in bare metal, virtual machine or containerized environment.

MSB Overview-Functionalities

Service Registration

- Service Registration
- Service Discovery
- Service Change Notification
- Service Status Change Notification
- Service Healthy Check

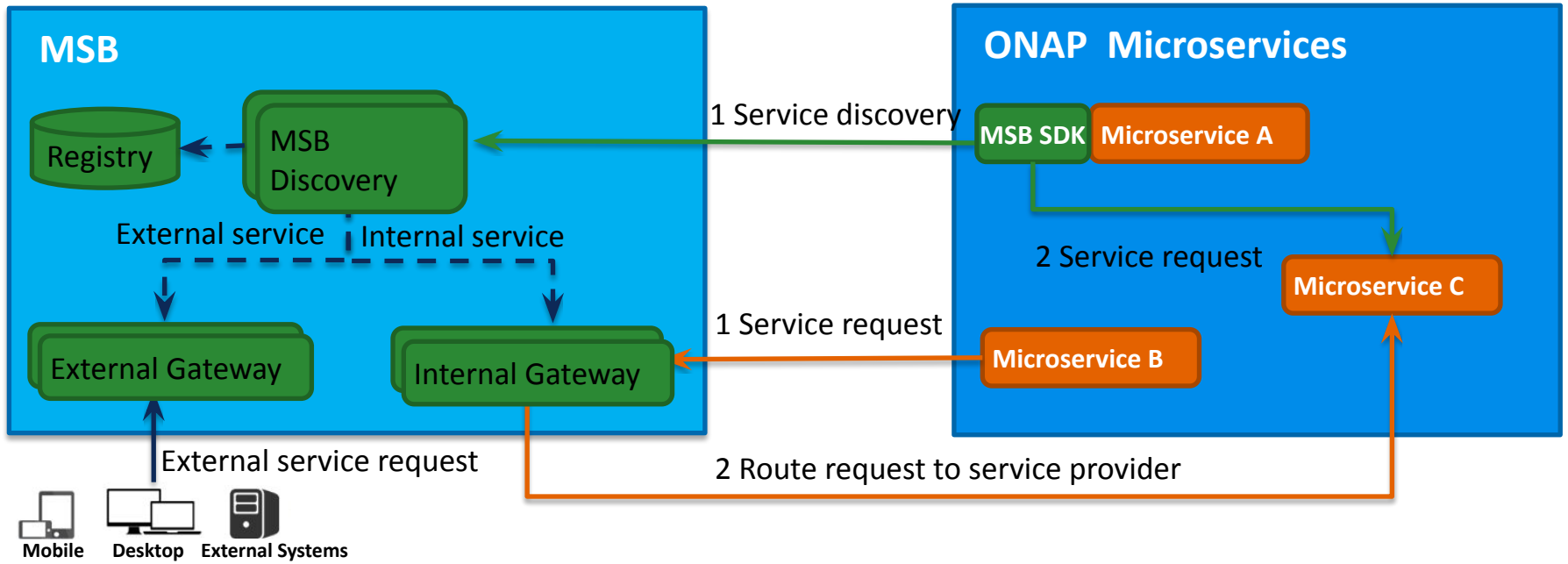
Load Balancing

- TCP/UDP Forwarding
- FTP Forwarding
- HTTP/HTTPS Forwarding
- WEB Socket Forwarding
- Route dynamically update

API Gateway

- Service requests statistics and analysis
- Pluggable Architecture
  - Transformation
  - Flow tagging
  - Rate Limiting
  - Circuit Breaker
  - Authentication
  - Other Plug-in …

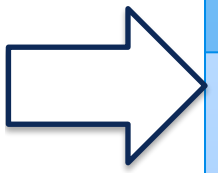
MSB Overview-Components



- Registry
  Service information storage, MSB uses Consul as the service registry.
- MSB Discovery
  Provides REST APIs for service discovery and registration
- API Gateway
  Provide service request routing, load balancing and service governance. It can be deployed as external Gateway or Internal Gateway.
- MSB SDK
  Java SDK for point to point communication

Service Registration-Information Model

```
{
  "serviceName": "catalog",
  "version": "v1",
  "url": "/api/catalog/v1",
  "protocol": "REST",
  "visualRange": "1",
  "lb_policy":"ip_hash",
  "nodes": [
    {
      "ip": "10.74.55.66",
      "port": "6666",
      "ttl": 0
    },
    {
      "ip": "10.74.56.36",
      "port": "8988",
      "ttl": 0
    }
  ]
}
```



Attribute	Description
serviceName	Service Name
version	Service Version
url	the actual URL of the service to be registered
protocol	supported protocols: 'REST', 'UI', 'HTTP','TCP'
visualRange	Visibility of the service. External(can be accessed by external systems):0 Internal(can only be accessed by ONAP microservices):1
path	The customized publish path of this service. If path parameter is specified when registering the service, the service will be published to api gateway under this path. Otherwise, the service will be published to api gateway using a fixed format: api/{serviceName}/{version}. The customized publish path should only be used for back-compatible.
lb_policy	Load balancing method, Currently two LB methods are supported, round-robin and ip-hash.
enable_ssl	True if the registered service is based on https. False if the registered service is based on http.
nodes	ip: the ip of theservice instance node port: the port of the service instance node ttl: time to live, this parameter is reserved for later use

Service Registration–RESTFul API

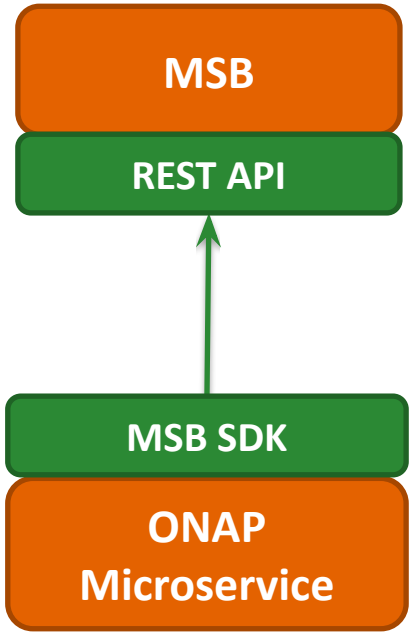
http method: POST
url: http://{msb_ip}:{msb_port}/api/microservices/v1/services

Example:

```
curl -X POST \
    -H "Content-Type: application/json" \
    -d '{"serviceName": "test", "version": "v1", "url": "/","protocol": "REST", "lb_policy":"round-robin","nodes": [
    {"ip": "127.0.0.1","port": "9090"}]}' \
    "http://127.0.0.1:10081/api/microservices/v1/services”
```

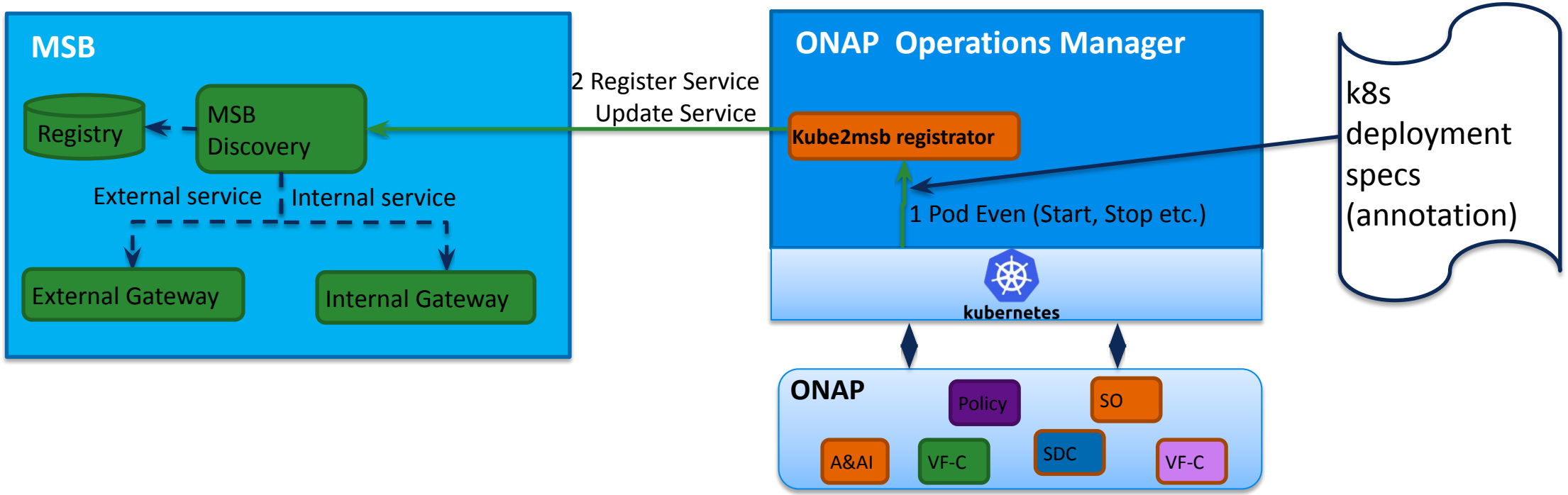
Service Registration–MSB SDK

Microservices can use MSB SDK to register themselves to MSB.




```java
public void registerMsb() throws Exception {

  //For real use case, MSB IP and Port should come from configuration
  //file instead of hard code here
  String MSB_IP="127.0.0.1";
  int MSB_Port=10081;

  MicroServiceInfo msinfo = new MicroServiceInfo();

  msinfo.setServiceName("animals");
  msinfo.setVersion("v1");
  msinfo.setUrl("/api/rpc/v1");
  msinfo.setProtocol("REST");
  msinfo.setVisualRange("1");

  Set<Node> nodes = new HashSet<>();
  Node node1 = new Node();
  node1.setIp(InetAddress.getLocalHost().getHostAddress());
  node1.setPort("9090");
  nodes.add(node1);
  msinfo.setNodes(nodes);

  MSBServiceClient msbClient = new MSBServiceClient(MSB_IP, MSB_Port);
  msbClient.registerMicroServiceInfo(msinfo, false);
}
```

Service Registration-Kube2msb Registrator



Kube2msb registrator can register service endpoints for the microservices deployed by OOM

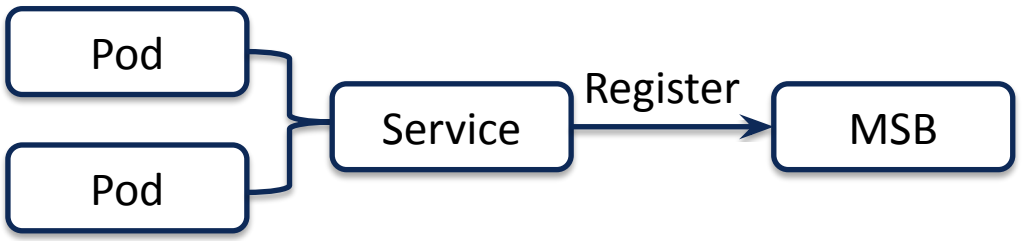
- OOM(Kubernetes) deploy/start/stop ONAP components.
- Registrator watches the kubernetes pod event .
- Registrator registers service endpoint info to MSB. It also updates the service info to MSB when ONAP components are stopped/restarted/scaled by OOM

Kube2msb Registrator-Service configuration

Use Kubernetes annotations to attach service endpoint metadata to objects.
Service endpoint can be defined at Pod level or Service level
Pod level: leverage the LB capabilities of MSB to distribute requests to multiple pods
Service level: MSB send the request to service(Cluster IP), K8s dispatch the request to the backend Pod

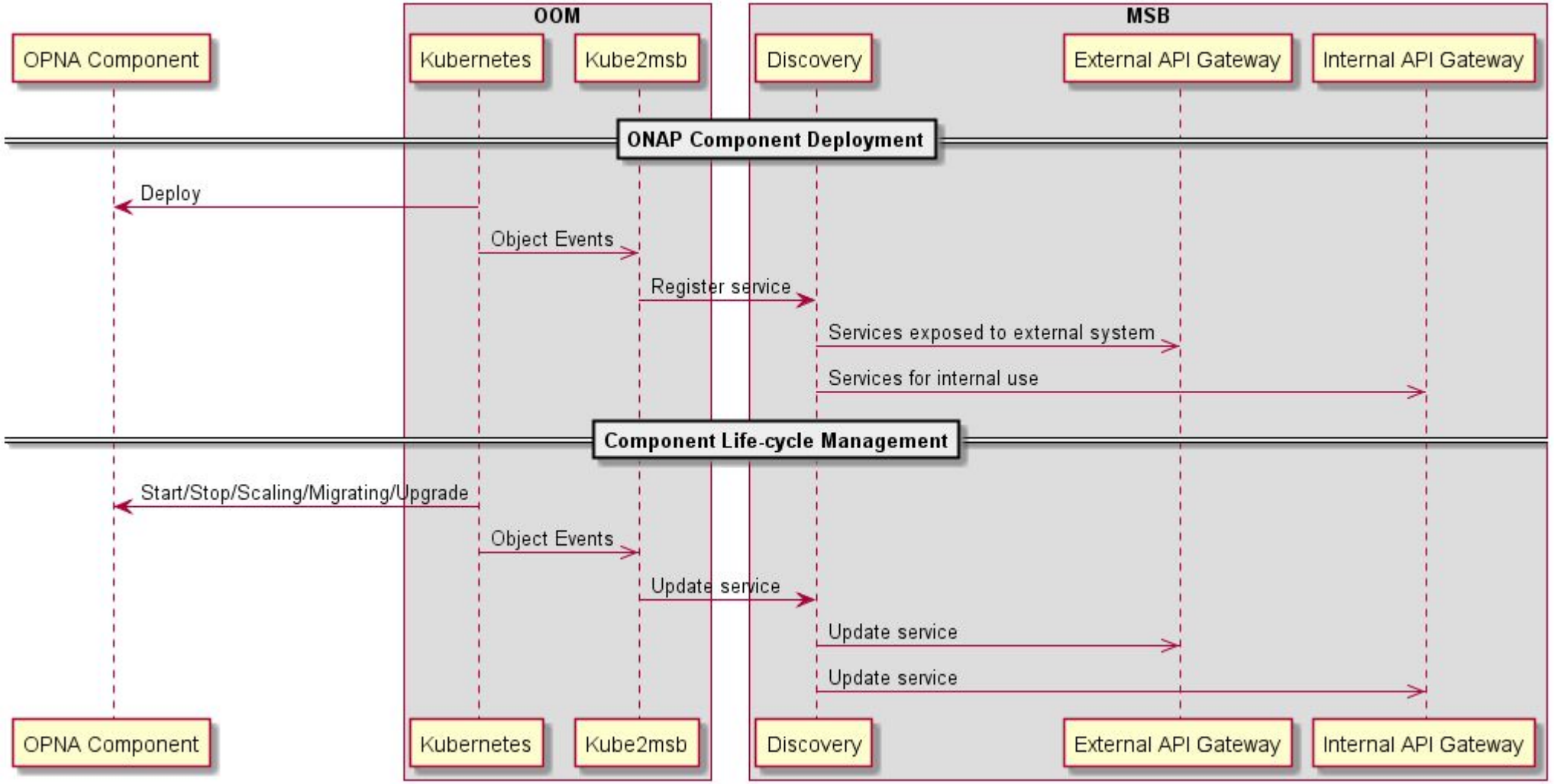
```
apiVersion: v1
kind: Service
metadata:
  name: aai-service
  annotations:
    msb.onap.org/service-info: '[
      {
          "serviceName": "aai-cloudInfrastructure",
          "version": "v1",
          "url": "/cloud-infrastructure",
          "protocol": "REST",
          "lb_policy": "round-robin",
          "visualRange":"1",
          "enable_ssl":"False"
      },
```

Register at service level



Register at pod level

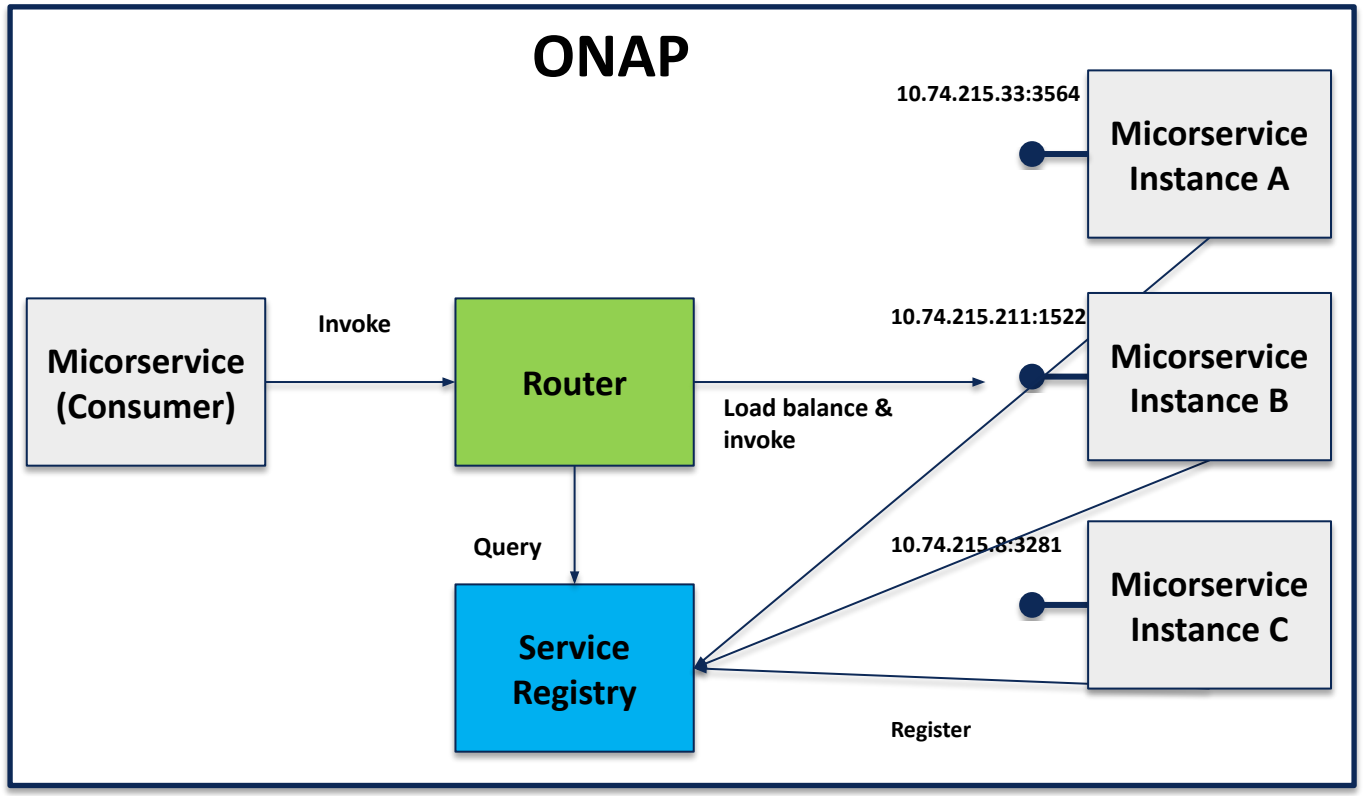
Kube2msb Registrator-flow chart



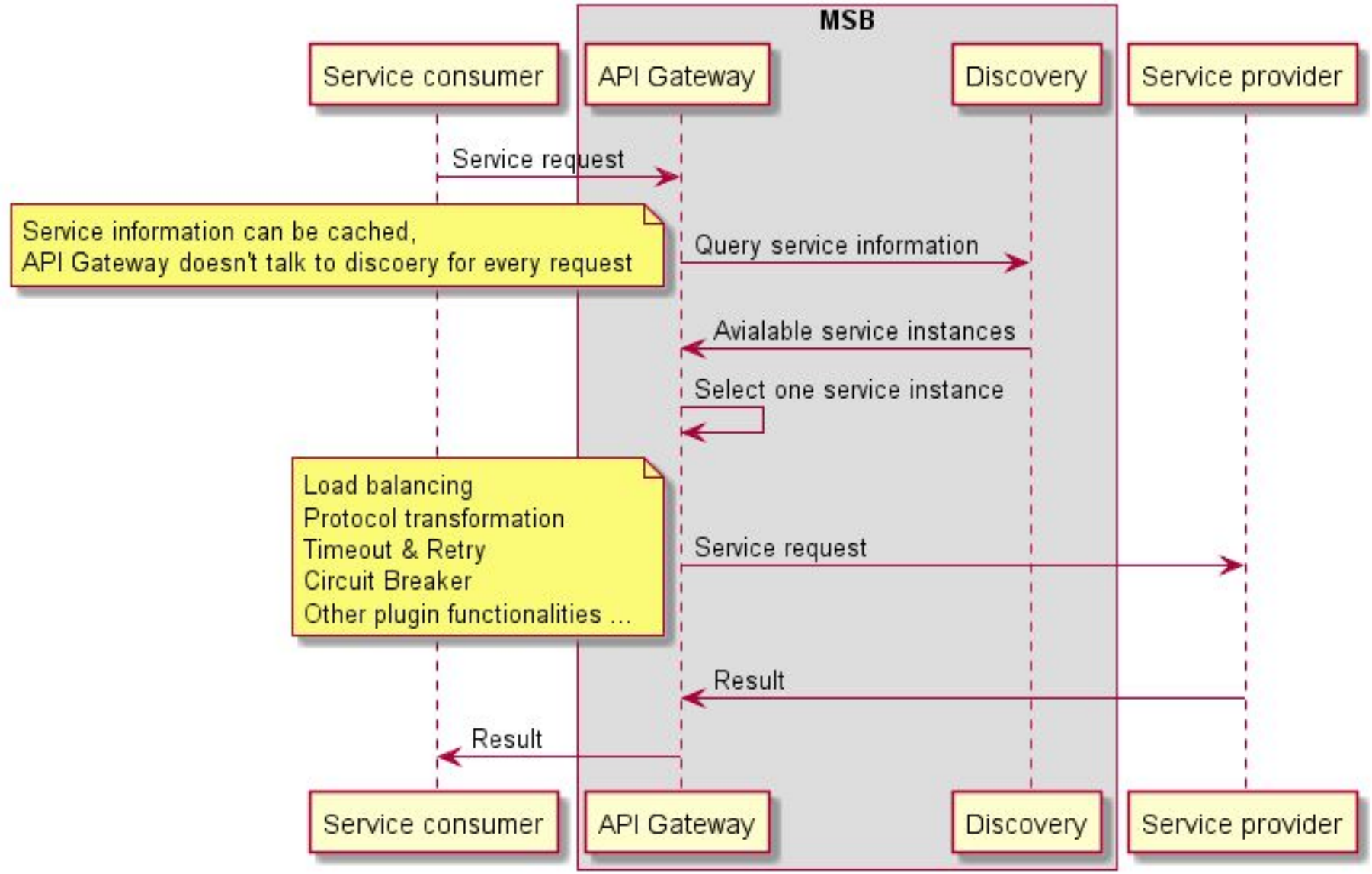
Service Discovery–Server Side Discovery

- Compared to client-side discovery, the client code is simpler since it does not have to deal with discovery. Instead, a client simply makes a request to the router.
- One more network hop is required than when using client-side discovery

Example: Curl http://msb_ip:msb_port/api/sdc/v1/catalog/resources



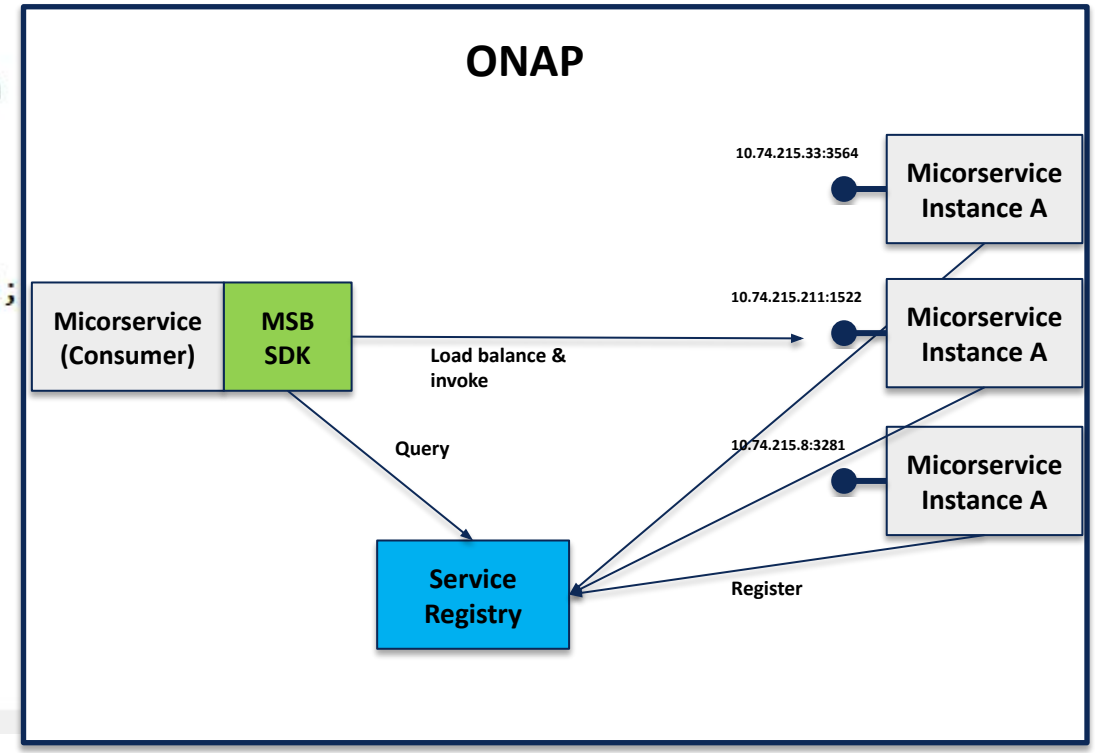
Service Discovery-Server Side Discovery



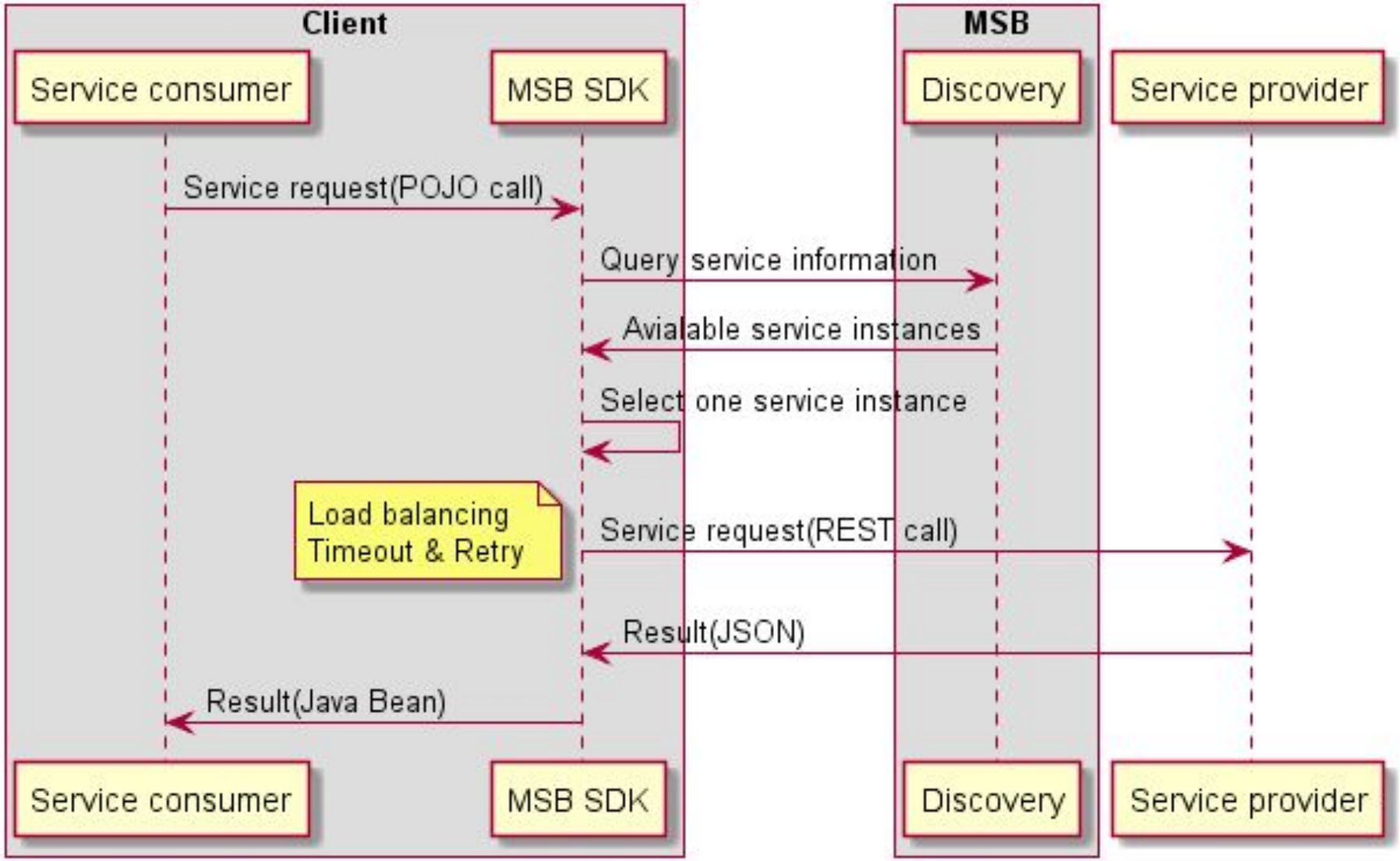
Service Discovery-Client Side Discovery

Microservices can use MSB SDK to discovery and access other microservices within ONAP.

```java
public static void main(String[] args) throws IOException {
  //For real use case, MSB IP and Port should come from configuration
  //file instead of hard code here
  String MSB_IP="127.0.0.1";
  int MSB_Port=10081;

  MSBServiceClient msbClient = new MSBServiceClient(MSB_IP, MSB_Port);

  RestServiceCreater restServiceCreater =
      new RestServiceCreater(msbClient);

  AnimalServiceClient implProxy =
      restServiceCreater.createService(AnimalServiceClient.class);

  Animal animal = implProxy.queryAnimal("panda").execute().body();
  System.out.println("animal:" + animal);
}
```



Service Discovery-Client Side Discovery



Example & Demo-Without OOM

- Start MSB services

1. Run the Consul dockers.
sudo docker run -d --net=host --name msb_consul consul:0.9.3

2. Run the MSB dockers.
Login the ONAP docker registry first: docker login -u docker -p docker nexus3.onap.org:10001

sudo docker run -d --net=host --name msb_discovery nexus3.onap.org:10001/onap/msb/msb_discovery
sudo docker run -d --net=host -e "ROUTE_LABELS=visualRange:1" --name msb_internal_apigateway nexus3.onap.org:10001/onap/msb/msb_apigateway

- Explore the MSB portal.

http://127.0.0.1/msb

- Register and test your REST service with MSB via curl

https://wiki.onap.org/display/DW/MSB+Test+Environment+Setup

Example & Demo-Within OOM

- **Precondition**

Have kubernetes cluster, kubectl and helm installed.
Login the ONAP docker registry first: docker login -u docker -p docker nexus3.onap.org:10001

- **Download oom from ONAP gerrit**

git clone https://gerrit.onap.org/r/oom

- **Install MSB and Kube2MSB registrator**

cd ~/oom/kubernets/config
./createConfig.sh -n onap
cd ~/oom/kubernetes/oneclick/
../createAll.bash -a msb -n onap
./createAll.bash -a kube2msb -n onap

- **Install AAI for testing**

./createAll.bash -a aai-n onap

- **Open the MSB IAG portal in the browser**

You are able to see the registered AAI services at http://${Node_IP}:30080/msb

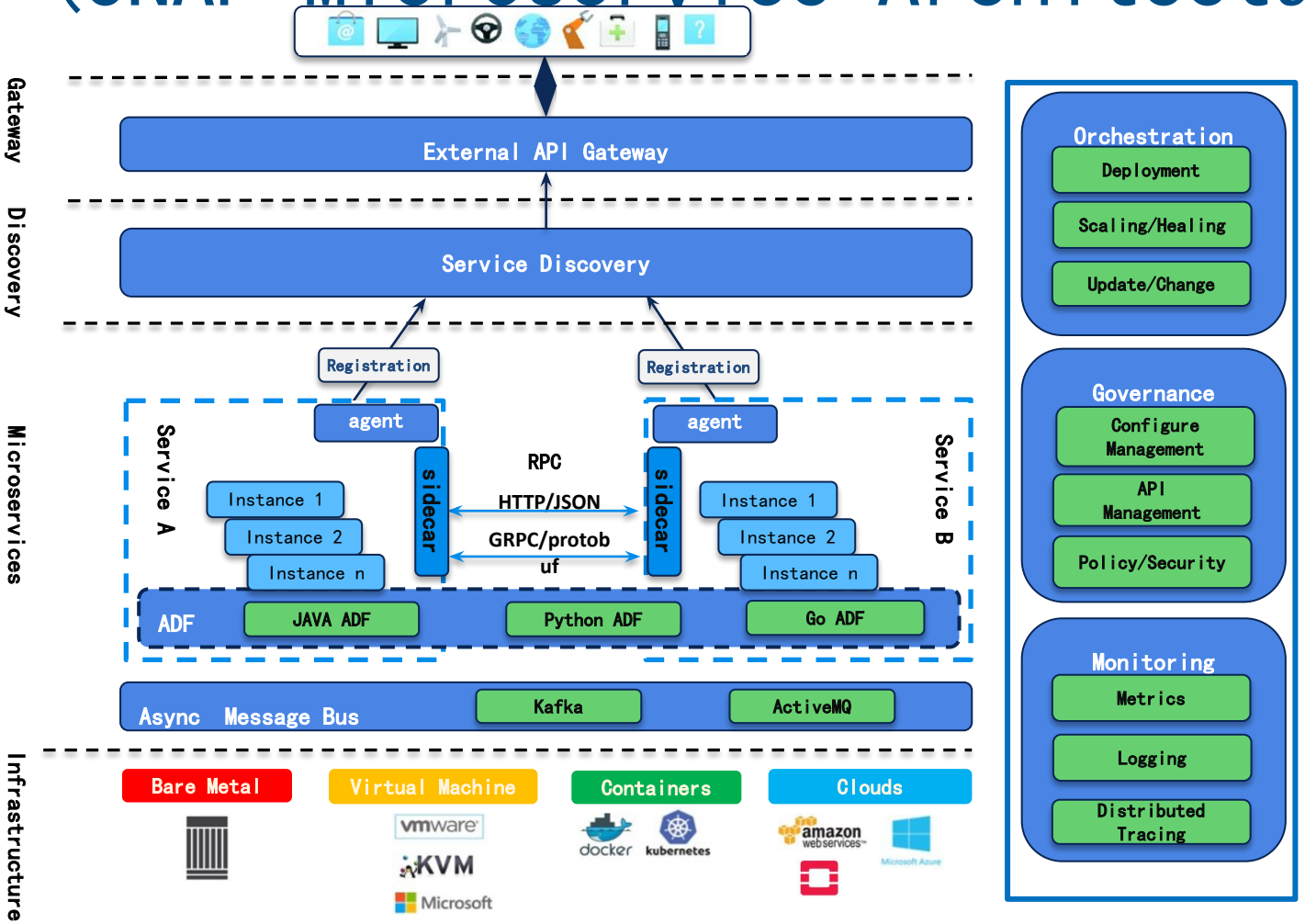
Suggested Integration approach-minimum impact to existing codes

- Automatically MSB registration by OOM Kube2MSB
- Access services via MSB Internal API Gateway
- Follows the standard URI structure

http://[host]:[port]/api/{service name}]/v{version number}/{resource}

https://wiki.onap.org/display/DW/RESTful+API+Design+Specification

The way going forward–OMSA (ONAP Microservice Architecture)



OMSA is the vision of ONAP Microservice Architecture to support carrier-grade requirements of ONAP microservices, which includes service registration/discovery, service communication, API gateway, service orchestration, service governance and service monitoring, etc.

Next step: Investgate Istio service mesh and integrate Istio into OMSA when it's production ready.

Note: this diagram is a functional view of OMSA, which is not mapped to specific projects