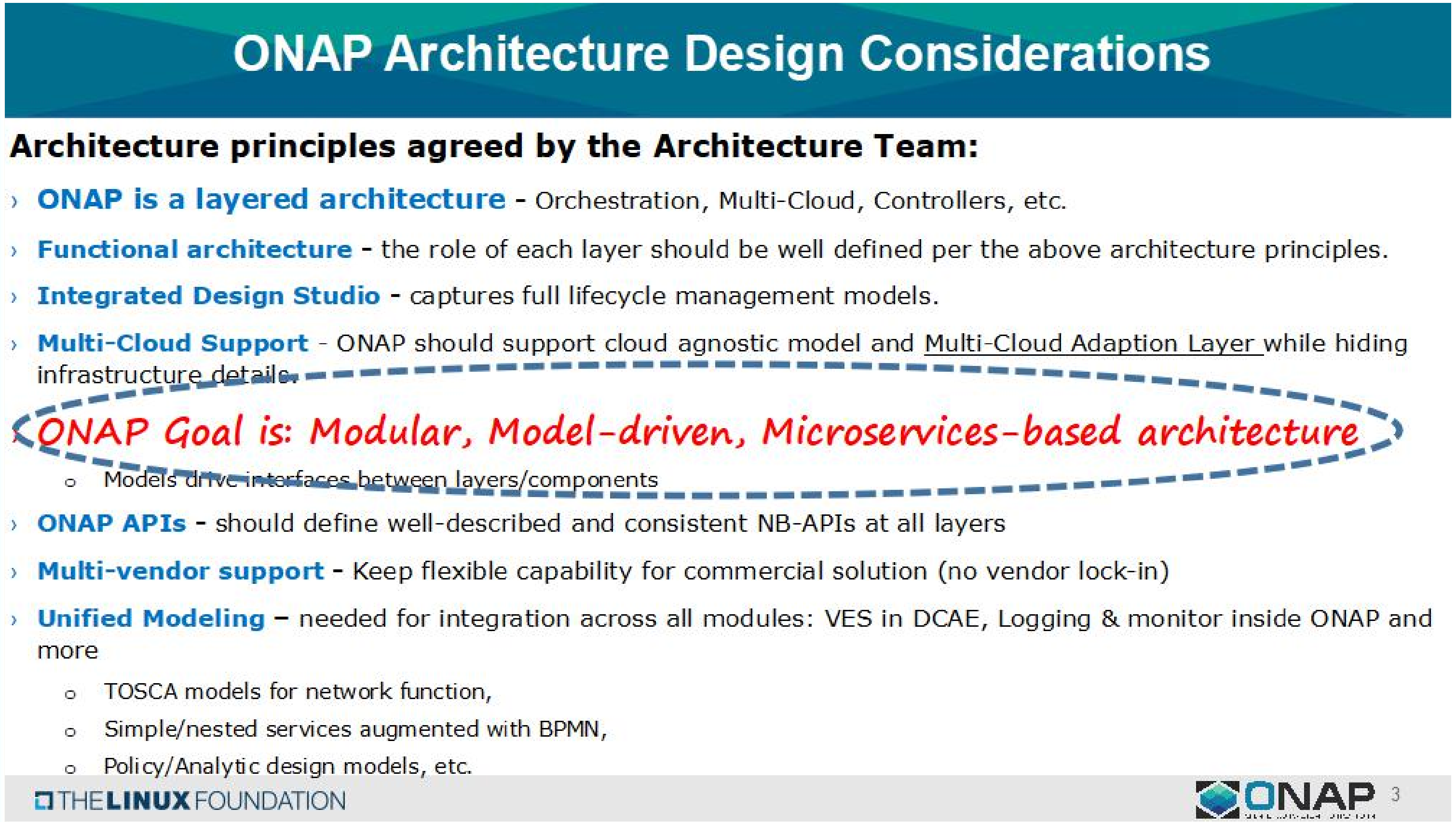
MSB's Plan to Support Microservice-Based ONAP with Istio Service Mesh

Huabing Zhao, PTL of MSB Project

ONAP Microservices-Based Architecture

Recommendation from Architecture Team



ONAP Architecture Design Considerations

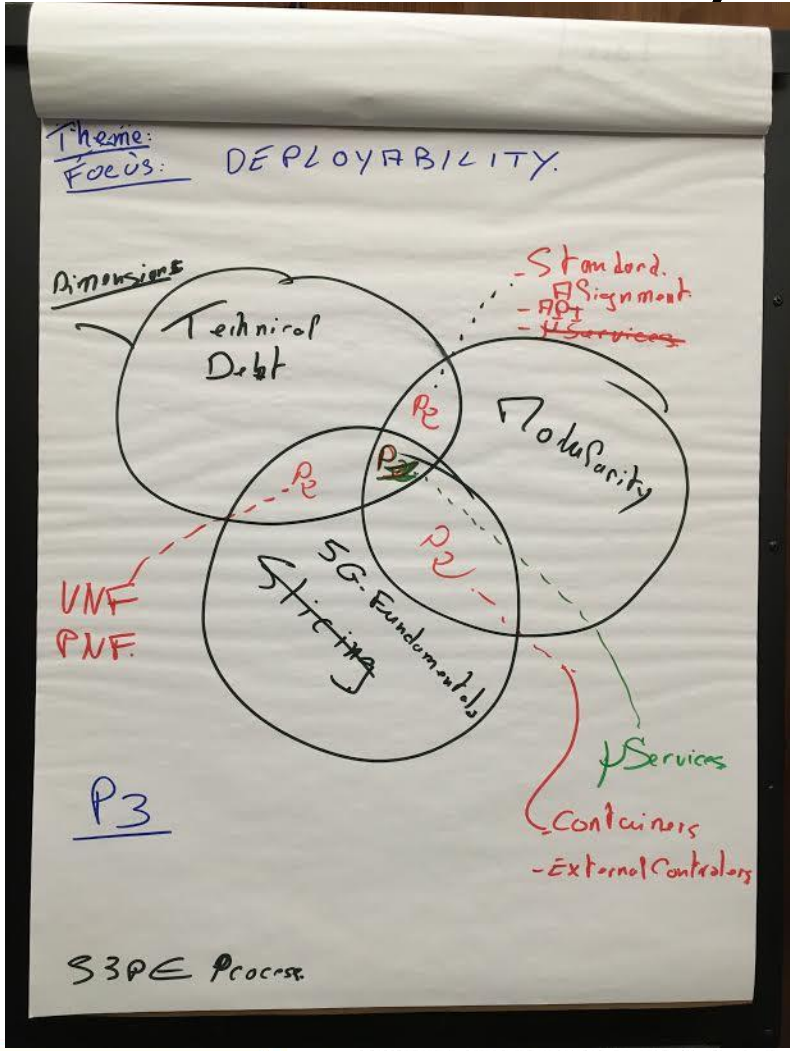
Architecture principles agreed by the Architecture Team:

- **ONAP is a layered architecture** - Orchestration, Multi-Cloud, Controllers, etc.
- **Functional architecture** - the role of each layer should be well defined per the above architecture principles.
- **Integrated Design Studio** - captures full lifecycle management models.
- **Multi-Cloud Support** - ONAP should support cloud agnostic model and Multi-Cloud Adaption Layer while hiding infrastructure details.
- *ONAP Goal is: Modular, Model-driven, Microservices-based architecture*
  - Models drive interfaces between layers/components
- **ONAP APIs** - should define well-described and consistent NB-APIs at all layers
- **Multi-vendor support** - Keep flexible capability for commercial solution (no vendor lock-in)
- **Unified Modeling** – needed for integration across all modules: VES in DCAE, Logging & monitor inside ONAP and more
  - TOSCA models for network function,
  - Simple/nested services augmented with BPMN,
  - Policy/Analytic design models, etc.

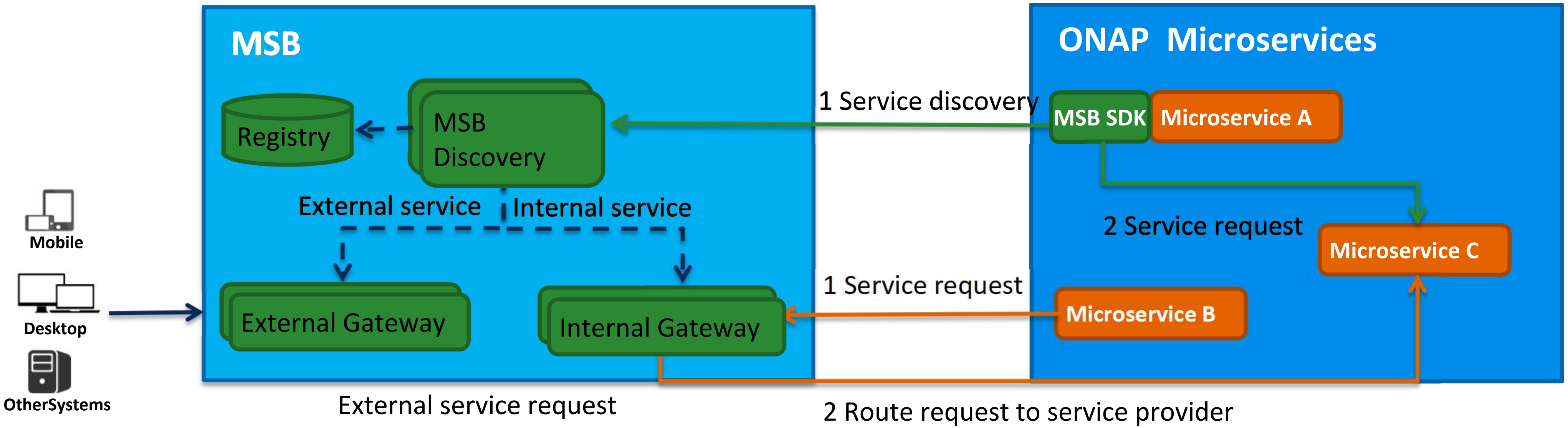
THE LINUX FOUNDATION

ONAP 3

Feedback from Community



MSB Overview-Components



- Registry

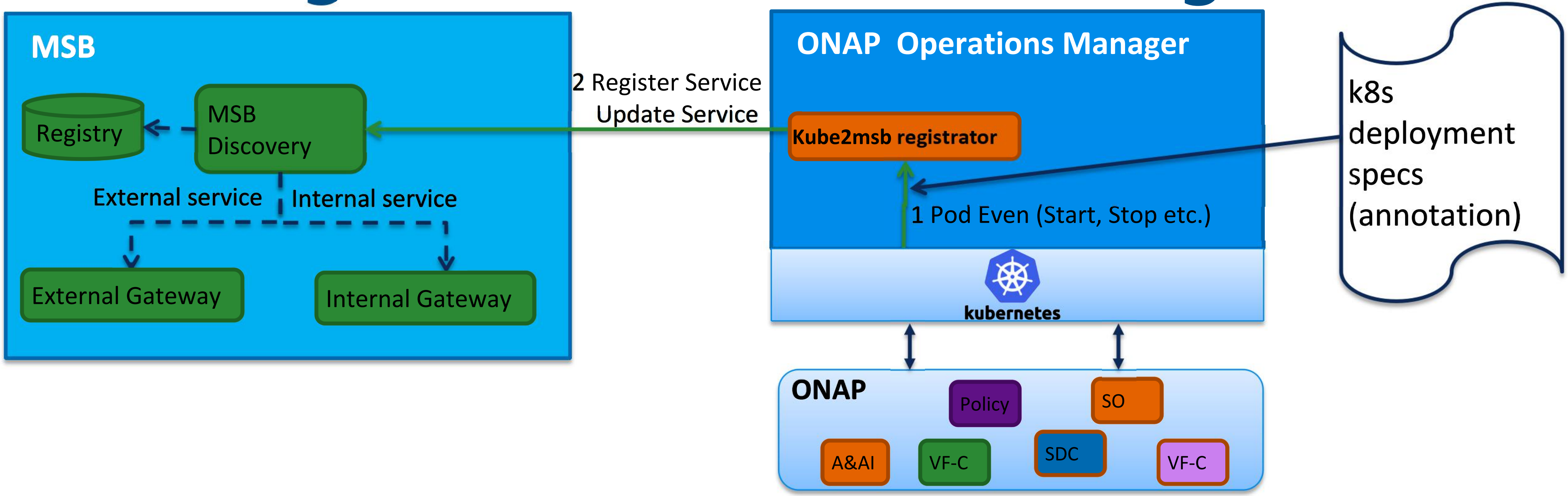
  Service information storage, MSB uses Consul as the service registry.
- MSB Discovery

  Provides REST APIs for service registration and discovery
- 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

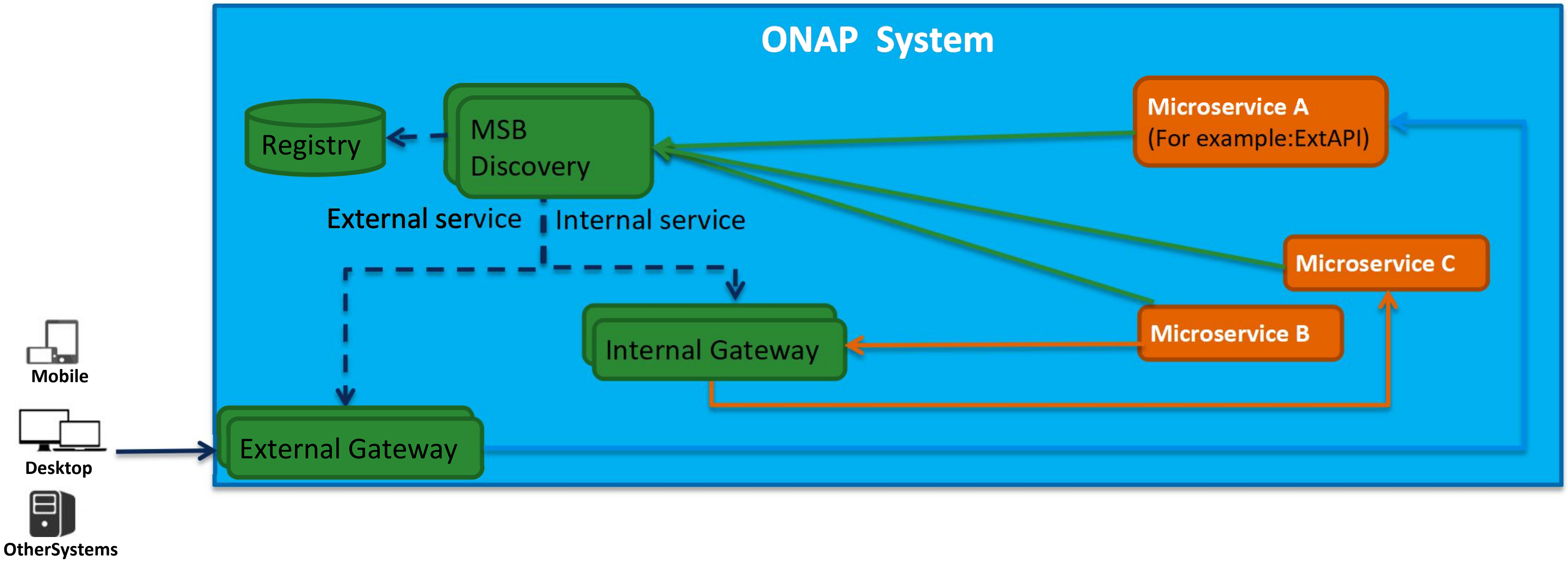
Service Registration-Kube2msb Registrator



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

Service Communication-API Gateway



- External API Gateway sits at the edge of ONAP system and to expose the services to the outside world.
- To push the service to External API Gateway, just set the visualRange of the registered service to 0
- API Gatway provide reliable communication and some policies such as Retry, LB, Circult Breaker, Rate Limiting, URL rewriting, etc.
- API Gateway is an open platform which can be extended by plugins

registration ←

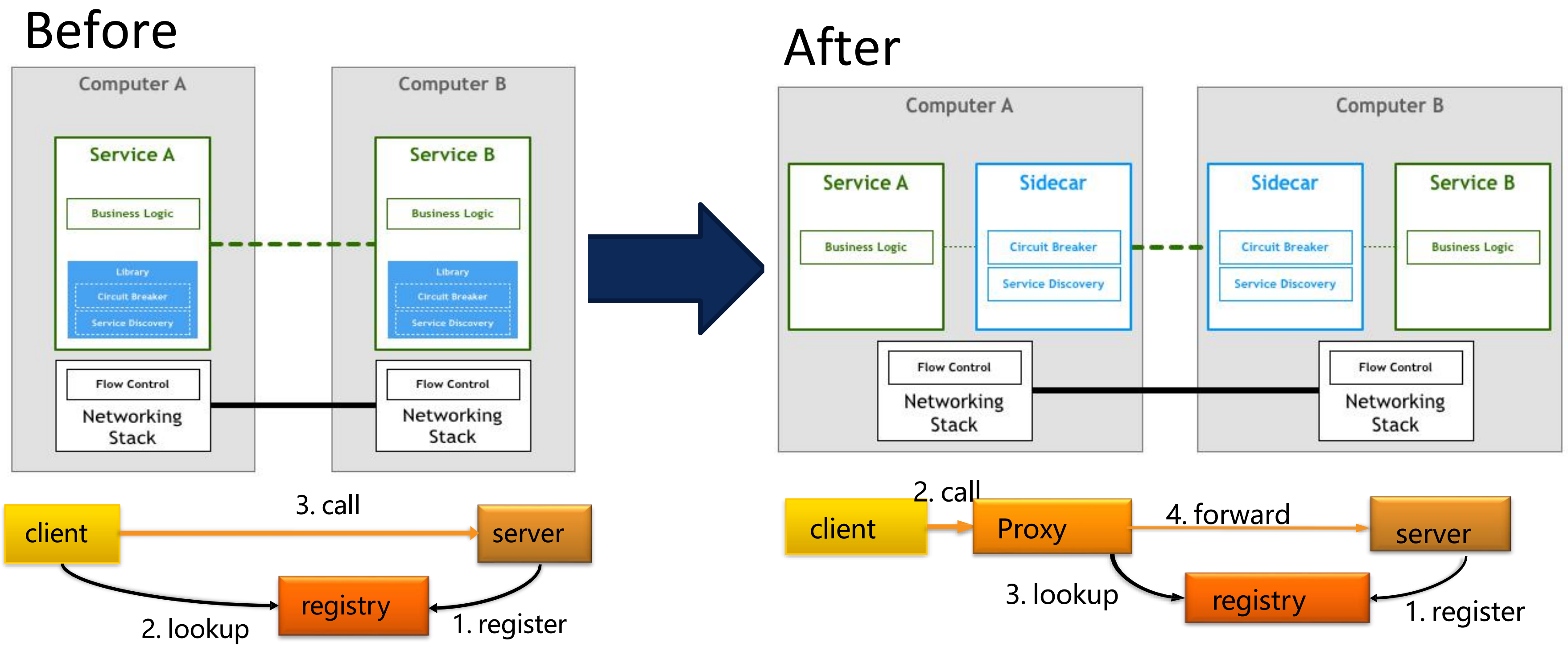
internal request ←

External request ←

Service Information ← - - -

Service Mesh

A service mesh is a dedicated infrastructure layer for handling service-to-service communication. It's responsible for the reliable delivery of requests through the complex topology of services that comprise a modern, cloud native application. In practice, the service mesh is typically implemented as an array of lightweight network proxies that are deployed alongside application code, without the application needing to be aware. -- **Willian Morgan**



Separation of Concers：

- The remote communication logic is moved into the proxy and deployed as "sidecar"
- Service developers only need to take care of the business logic

THE LINUX FOUNDATION

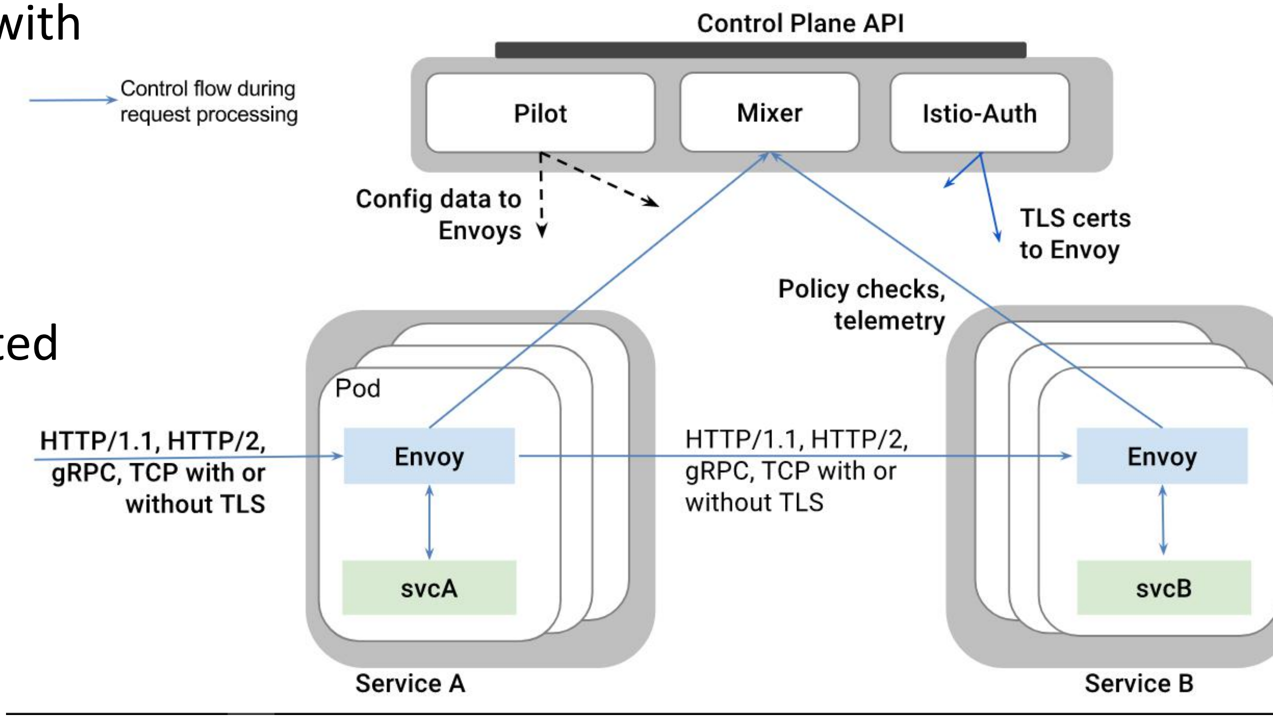
Istio service mesh

Istio is a open source service mesh project.

Istio features:

- **Stability and Reliability**: Reliable communication with retries and circuit breaker
- **Security** : Secured communication with TLS
- **Performance**: Latency aware load balancing with warm cache
- **Observability**: Metrics measurement and distributed tracing without instrumenting application
- **Manageability**: Routing rule and rate limiting enforcement
- **Testability**: Fault injection to test resilience of the services

Istio Architecture



Control Plane API

Control flow during request processing

Pilot

Mixer

Istio-Auth

Config data to Envoys

TLS certs to Envoy

Policy checks, telemetry

Pod

HTTP/1.1, HTTP/2, gRPC, TCP with or without TLS

Envoy

svcA

HTTP/1.1, HTTP/2, gRPC, TCP with or without TLS

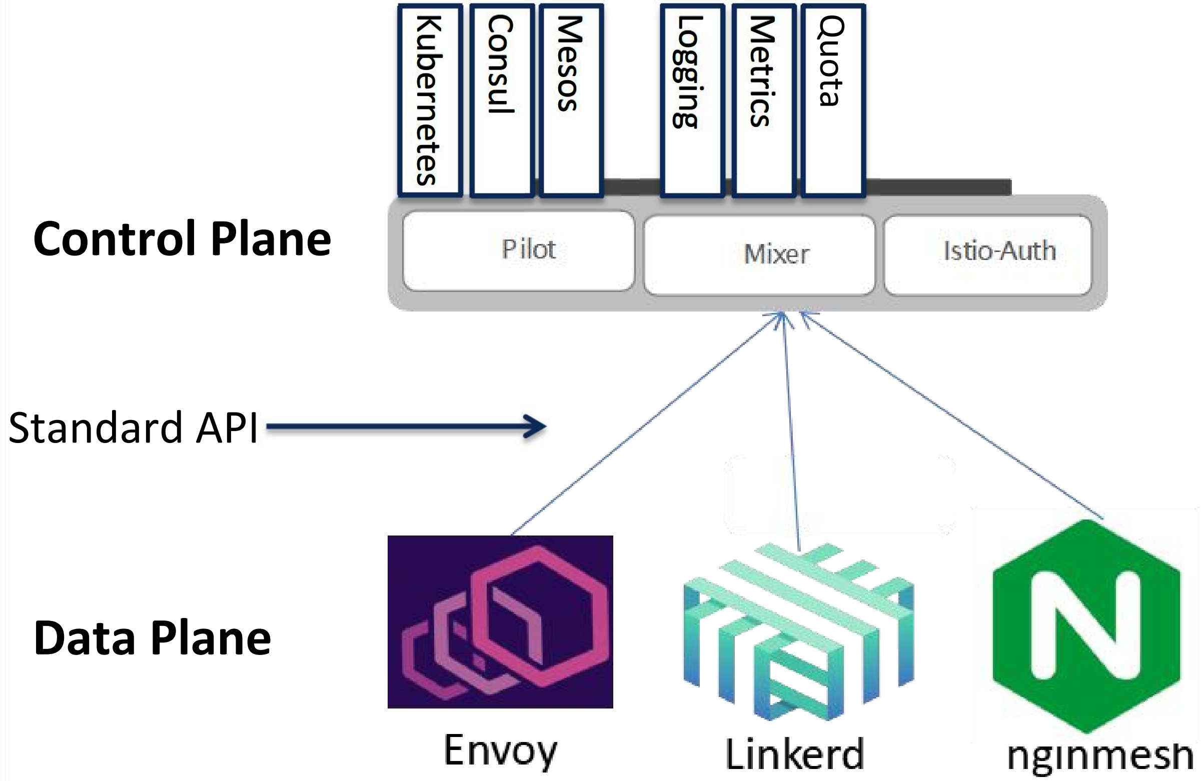
Envoy

svcB

Service A

Service B

Why Istio?



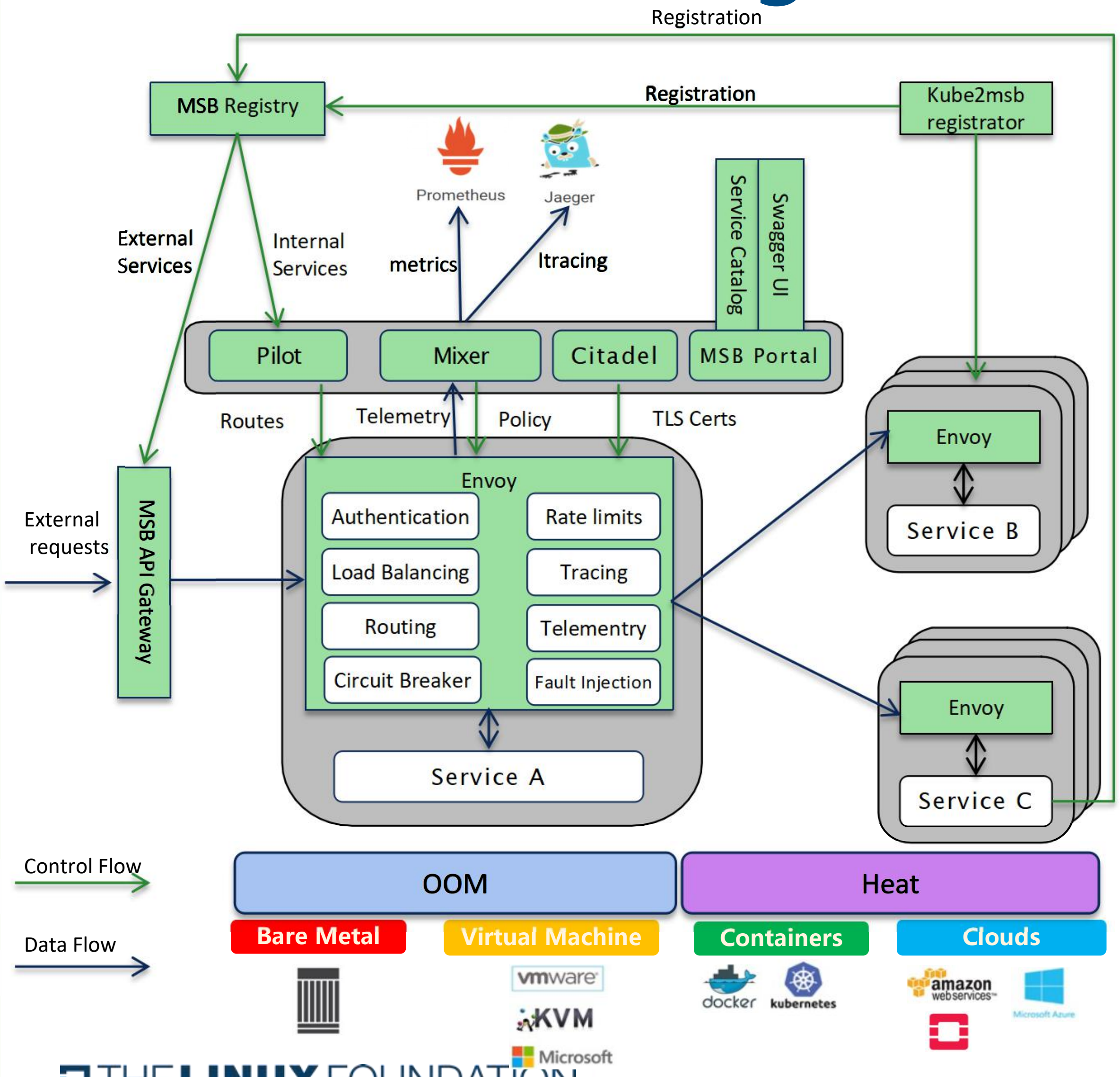
The main advantage of Istio is introducing a centralized **Control Plane** to manage the distributed sidecars across the mesh, Control plane is a set of centralized management utilities including:

- **Pilot**: routing tables, service discovery, and load balancing pools
- **Mixer**: Policy enforcement and telemetry collection
- **Citadel**: TLS mutual service authentication and fine-grained RBAC

The Istio control plane is highly extendable by design.

- Multiple adapters can be plugged into Pilot to populate the services: Kubernetes, Consul, Mesos...
- Different backends can be connected to Mixer without modification at the application side: Prometheus, Heapster,AWS CloudWatch...
- Standard API between Pilot and data plane for service discovery, LB pool and routing tables which decouples the sidecar implementation and Pilot: Envoy, Linkerd, Nginmesh are all support Istio now and can work along with Istio as sidecar

ONAP Istio Integration



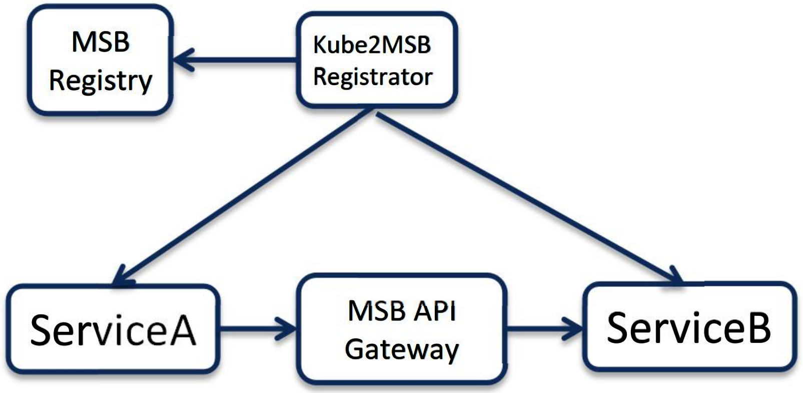
- Integrate Istio with ONAP to provide a reliable, secure and flexible service communication layer(service discovery/retries/circuit breaker/route rule/policy)
- Integrate with CNCF projects
  - ➢ jaeger to provide distributed tracing
  - ➢ prometheus and grafana for metrics collection and display
- Add MSB Portal to control plane to provide service Catalog ,swagger UI of Restful API, service mesh configuration ,etc
- Leverage Istio to achieve close loop operation for ONAP system itself - long time goal

Principles:

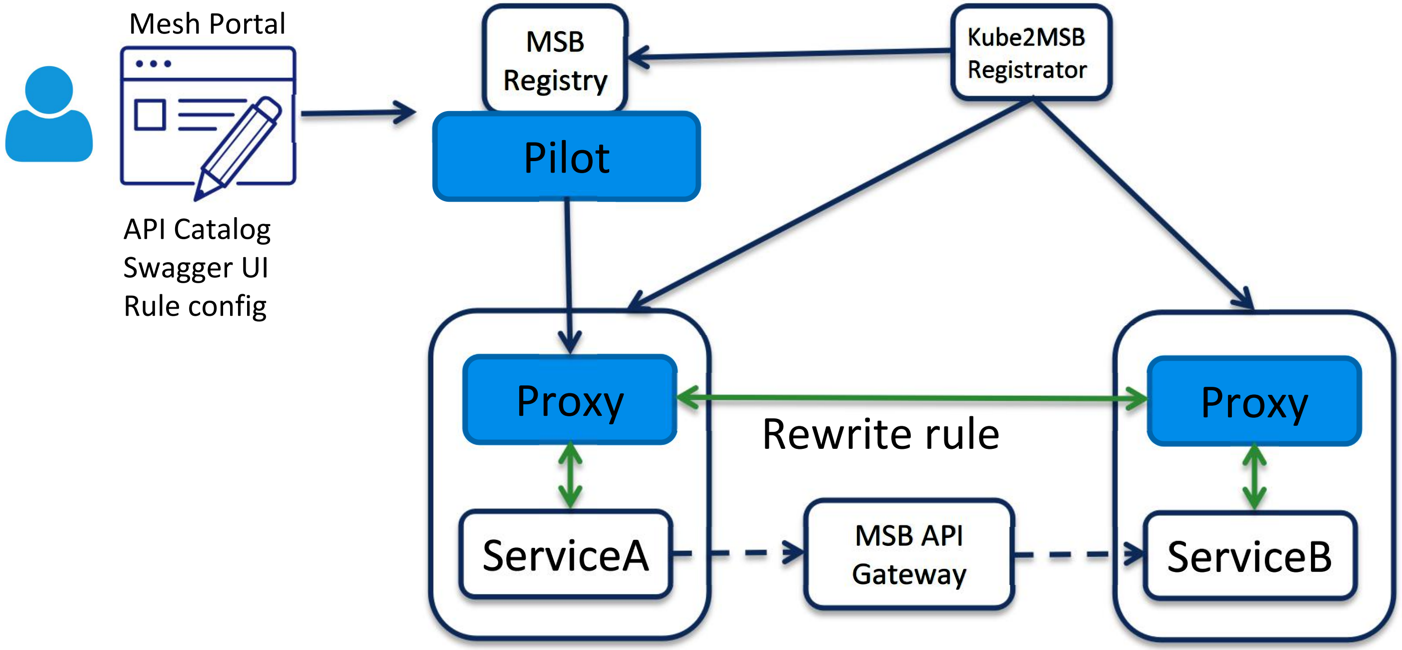
- Minimize the impacts to ONAP projects
- Start from a few Microservices
- Keep it compatible with existing inter-services communication approaches

Support Either Deployment With or Without Service Mesh

ONAP without Service Mesh



ONAP with Service Mesh



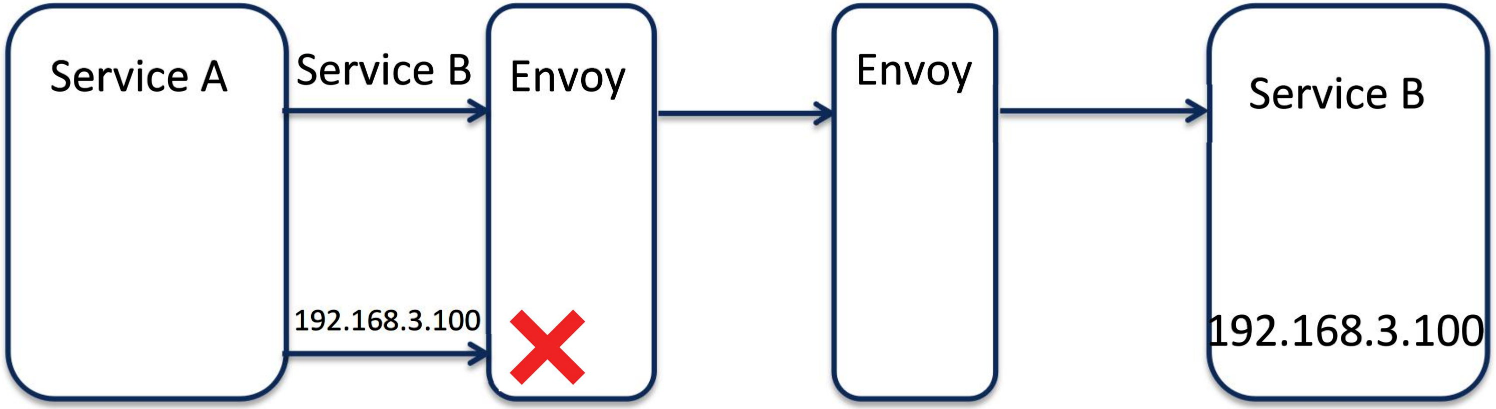
It looks exactly the same from the microservie point of view

- ONAP can work **with or without service mesh**
- **Compatible** with current inter-service communication approaches
- Be **transparent** to microservices, no modification at the application codes
- Support **OOM and Heat** deployment with/without service mesh for transition
- Give ONAP users the **flexibility** to choose based on their own requirement and scenario

Some Considerations

Dataplane sidecar takes over all the traffics, however,

- ONAP modules may not adapt service mesh approach at once, need to allow services inside and outside of the mesh to access each other
- Some traffics may not need to go through sidecar, for example, plain UDP traffics



Solution:

- Modify the IP Table rules to bypass some specific traffics

Current Progress

- Istio installation with bookinfo sample application in ONAP lab-done
- Integrate MSB Discovery to Pilot via Consul plugin-done
- An agent to watch the MSB Discovery and put route rules to Pilot-80%
- User UI for Istio routing rules - 50%

Next

- Start with an ONAP moudle to deploy sidecar and test Istio functionalities
- Istio installation and sidecar auto injection -work with OOM
- Addone installation (Jaeger, Promeheus)-work with OOM
- ……