API Connect Architecture (Micro Services)

Ozair Sheikh

Senior Offering Manager – API Connect & Gateways

Important Disclaimers

IBM’s statements regarding its plans, directions and intent are subject to change or withdrawal without notice at IBM’s sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision. The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Important Disclaimers

- **IBM Confidential**. Unless specifically advised otherwise, you should assume that all the information in this presentation (whether given in writing or orally) is IBM Confidential and restrict access to this information in accordance with the confidentiality terms in place between your organization and IBM.
- **Content Authority**. The workshops, sessions and materials have been prepared by IBM or the session speakers and reflect their own views. They are provided for informational purposes only, and are neither intended to, nor shall have the effect of being, legal or other guidance or advice to any participant. While efforts were made to verify the completeness and accuracy of the information contained in this presentation, it is provided AS-IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this presentation or any other materials. Nothing contained in this presentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.
- **Performance**. Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.
- **Customer Examples**. Any customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.
- **Availability**. References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates.

Trademark acknowledgements

- IBM & IBM logo are trademarks of International Business Machines Corporation, registered in many jurisdictions.
- Other company, product and service names may be trademarks, registered marks or service marks of their respective owners. A current list of IBM trademarks is available on the web at "Copyright and trademark information" ibm.com/legal/copytrade.shtml

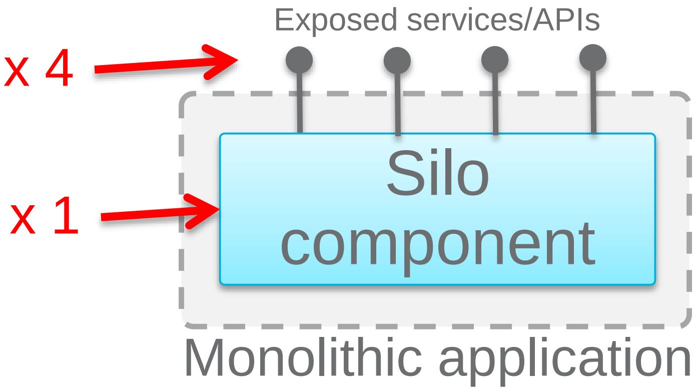
Microservices

Common misconception resulting from the term “microservice”

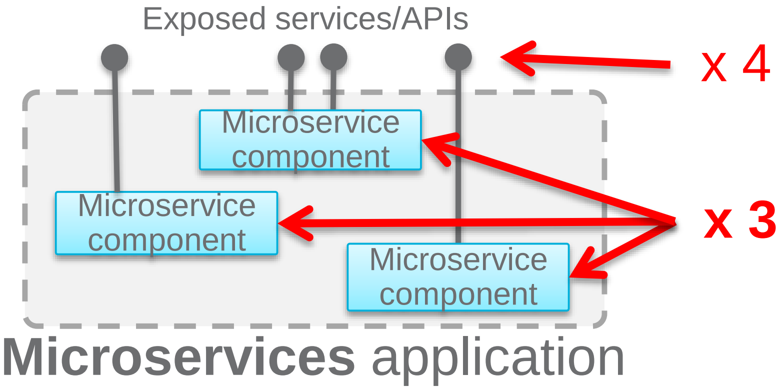
~~Microservices are just more fine grained web services~~

~~APIs are microservices~~

“micro” refers to the granularity of the ***components****,*
not the granularity of the exposed interfaces



Monolithic application



Microservices application

Is “microservices architecture” is really “micro-component architecture”?

Clarification on Microservices vs APIs - short video (4 mins)
http://ibm.biz/MicroservicesVsAPIVideo

Inter-microservice vs. inter-application communication

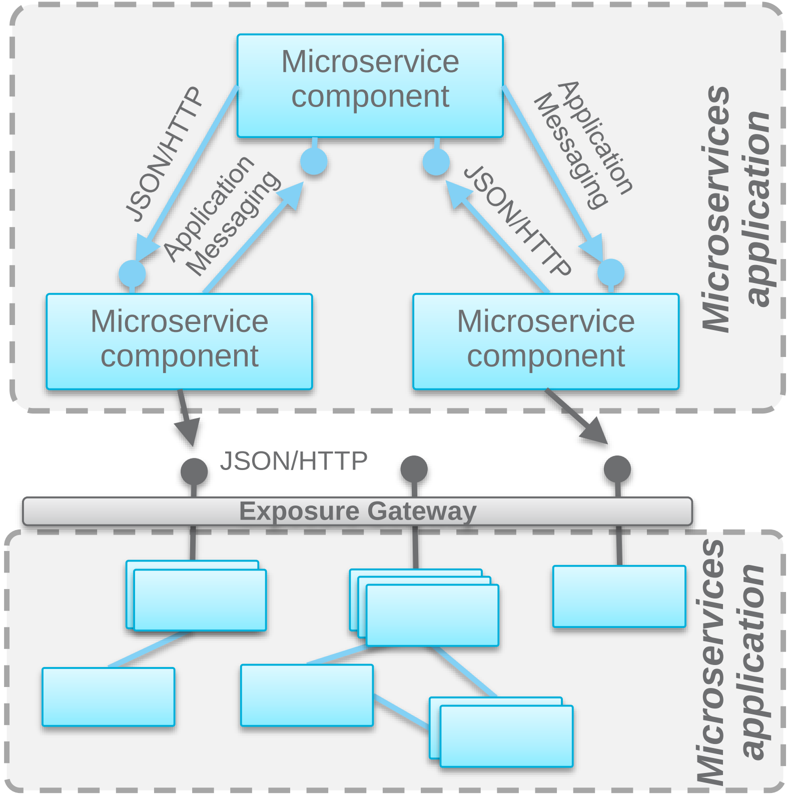
Inter-**microservice** communication

- Lightweight protocols: HTTP, application messaging
- Runtime component registry
- Client-side load balancing and circuit breaker patterns

Inter-**application** communication

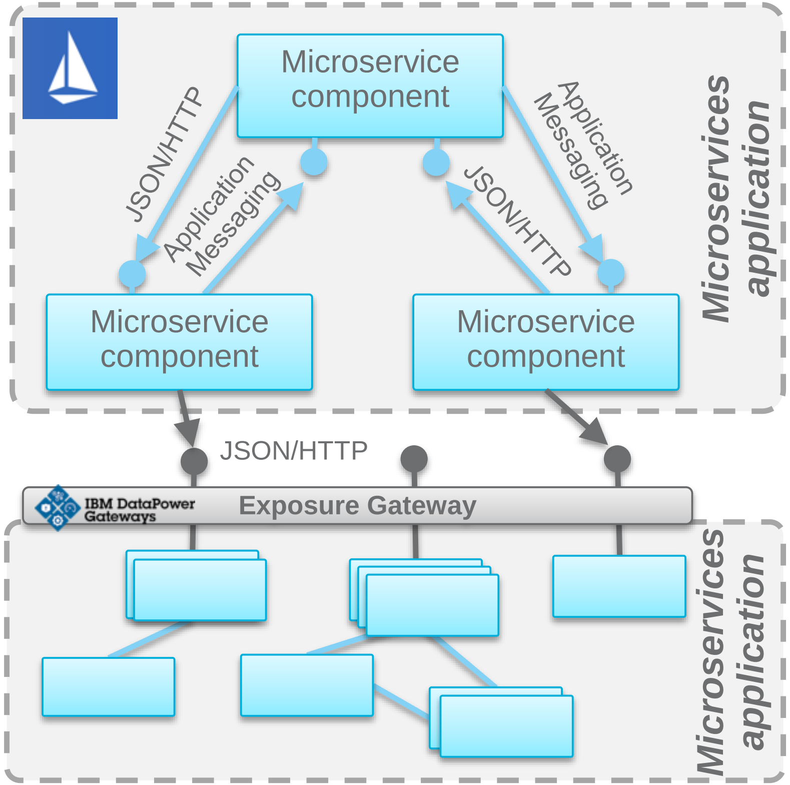
- Enterprise protocols: Managed API gateways, enterprise messaging
- Design time developer portals
- Gateway load balancing and throttling

JSON/HTTP RESTful communication styles may be present in both types of communication, but their implementation may be radically different.

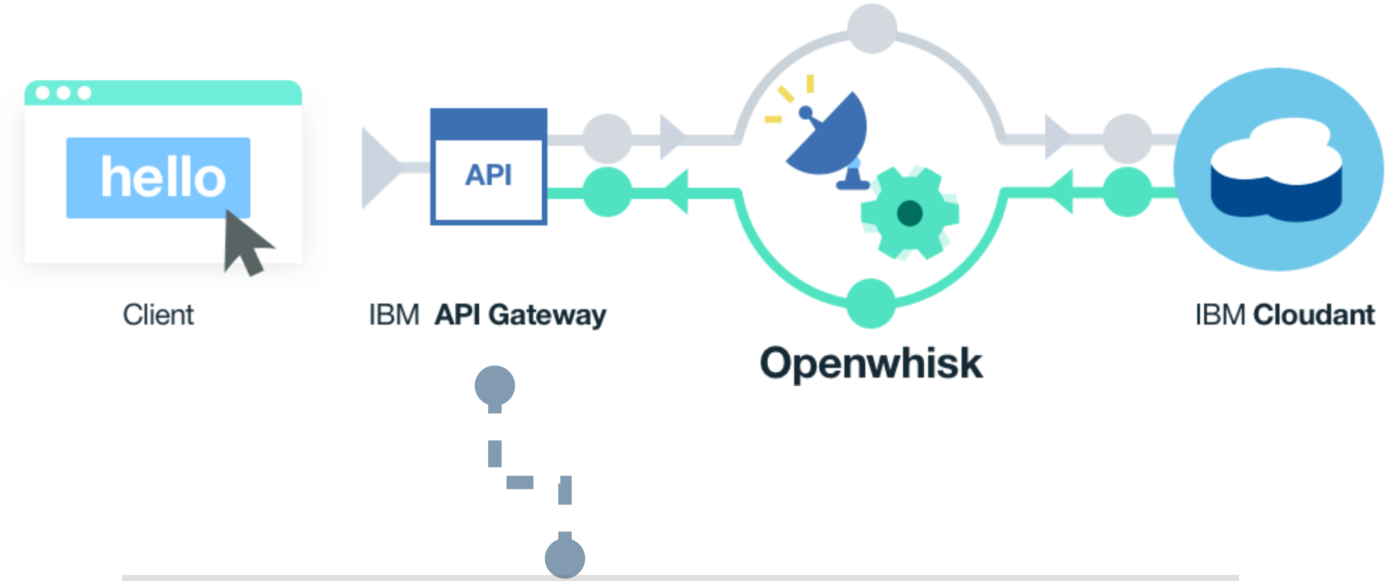


ISTIO – microservices management

- Inter-service proxy between microservices (ie polygot runtimes)
  - Non-intrusive to App (ie transparently intercepted and proxied)
- Inject app-smarts (ie rate limit, security, etc …) in the network without making app changes
- Pluggable decision maker called **mixer** optimized for inter-service communication for enforcing



Microservices Example: Securing and Controlling your Serverless Action…



API Gateway takes care of:
Throttling (Rate limiting)
Security (API Key, Secret, OAUTH, CORS)
Map Endpoints to Actions & Define them
Branding

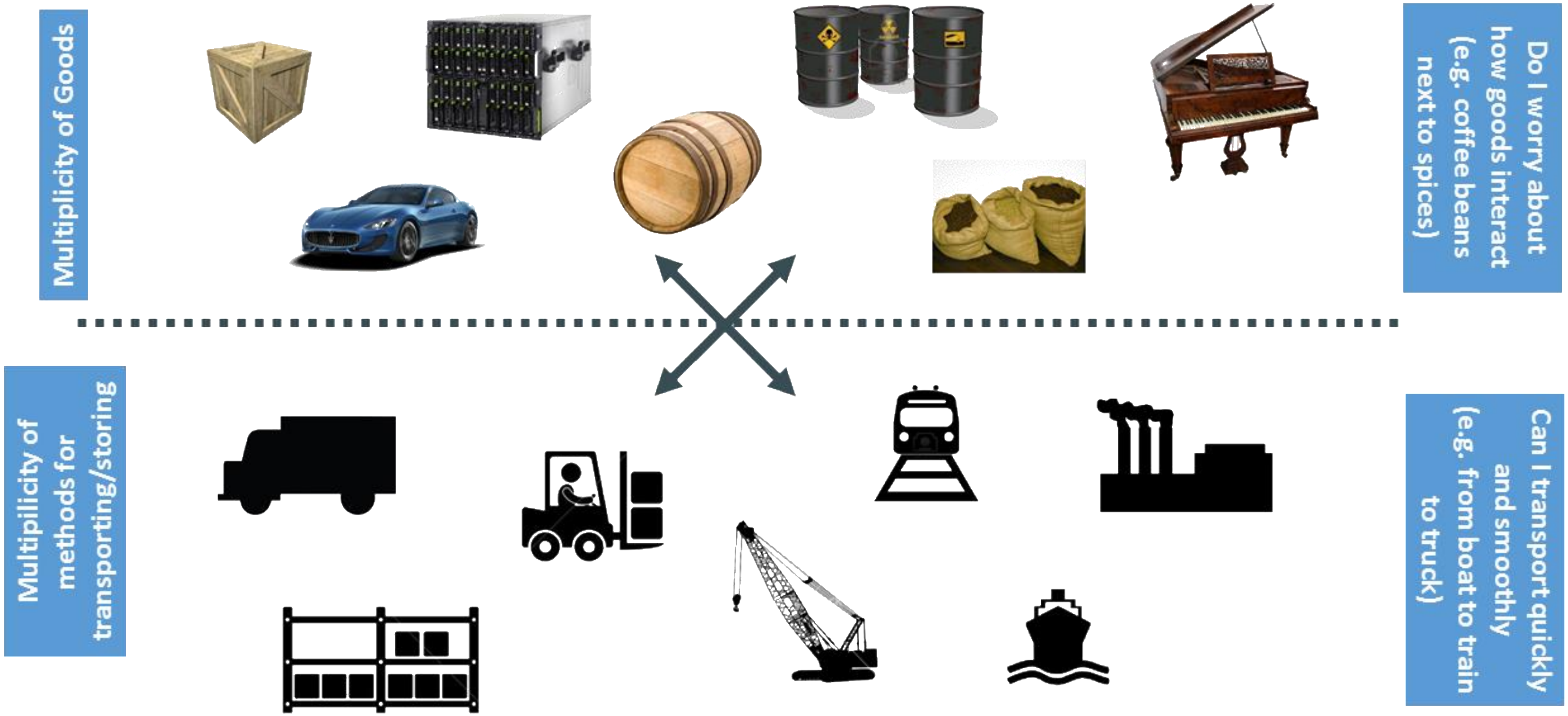
Define API Endpoints (URLs)
(optionally by using custom domains)

Define Actions:
getCostumer
createCostumer
deleteCostumer

Map Endpoints to Actions
Get
Post
Delete

Docker

What are containers?



Multiplicity of Goods

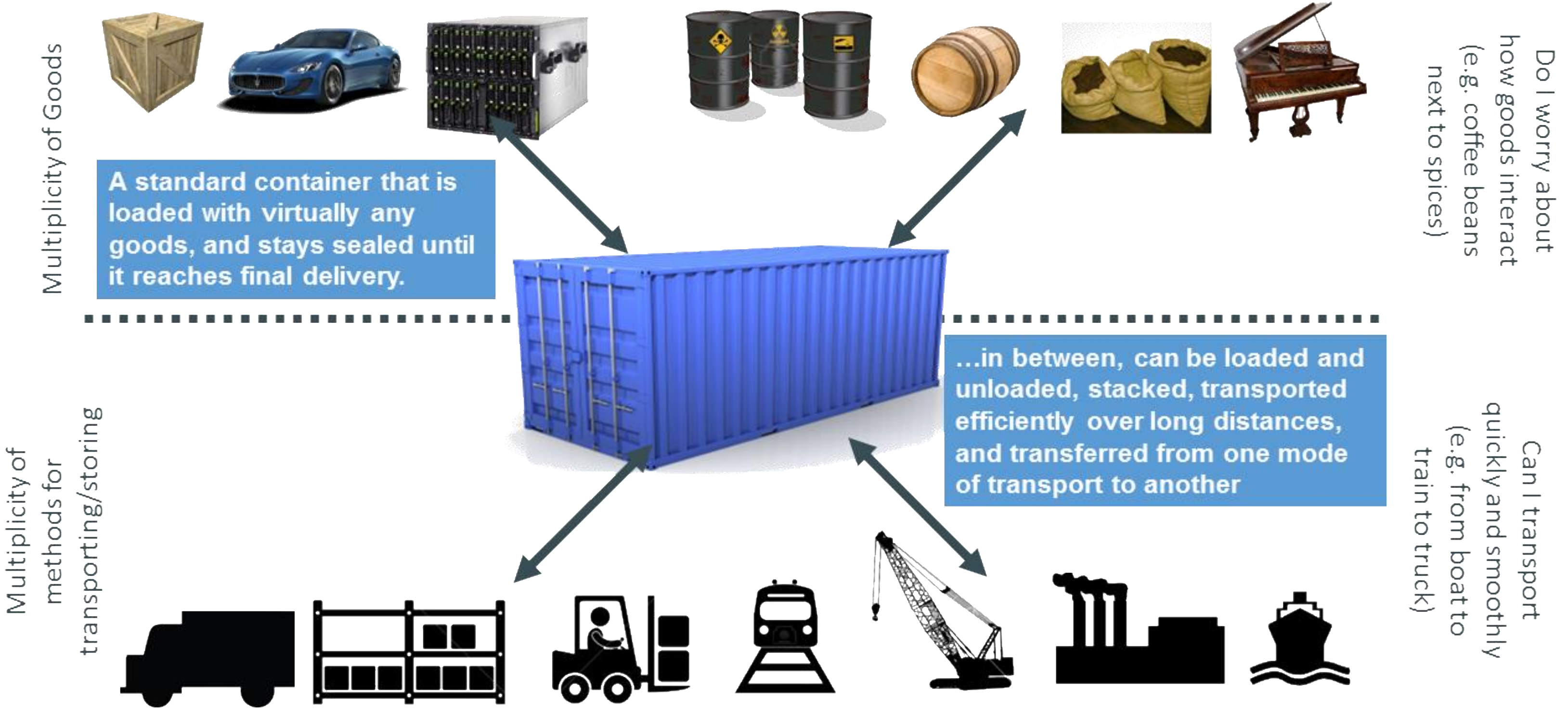
Do I worry about how goods interact (e.g. coffee beans next to spices)

Multipilicity of methods for transporting/storing

Can I transport quickly and smoothly (e.g. from boat to train to truck)

Source: http://slidedeck.io/pointful/docker-intro

What are containers?



Multiplicity of Goods

A standard container that is loaded with virtually any goods, and stays sealed until it reaches final delivery.

Do I worry about how goods interact (e.g. coffee beans next to spices)

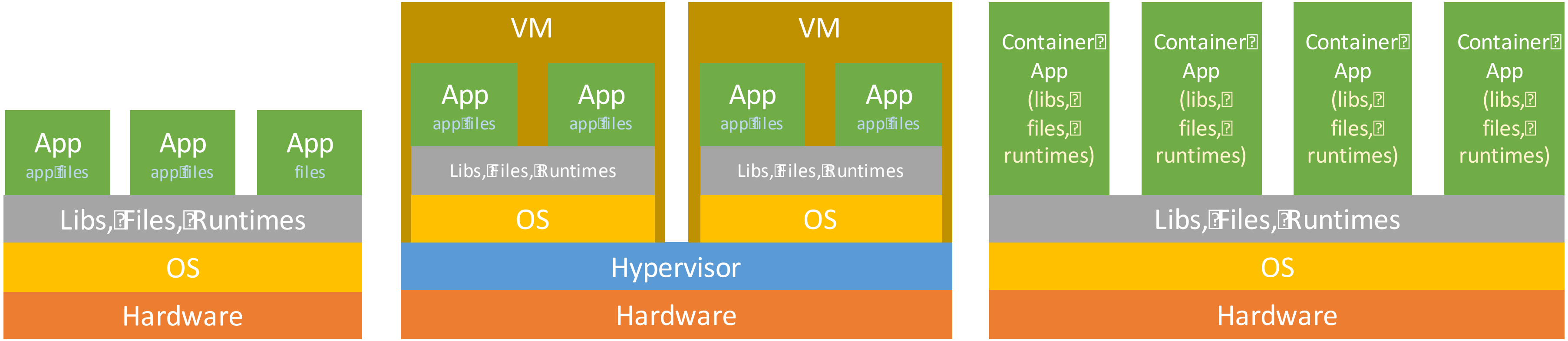
…in between, can be loaded and unloaded, stacked, transported efficiently over long distances, and transferred from one mode of transport to another

Multiplicity of methods for transporting/storing

Can I transport quickly and smoothly (e.g. from boat to train to truck)

Source: http://slidedeck.io/pointful/docker-intro

Background and History



Early days in the Data Center: A dedicated OS for the hardware and several apps deployed in that OS sharing the libraries and runtimes

Virtualization in the Data Centers: A hypervisor enables sharing of the hardware resources across multiple virtual machines with their own definition of CPU, memory, storage and dedicated OS

Modern Data Centers: Each container defines its own sandbox where it loads its required versions of libraries, distinct file system isolation (through layers) and application components and runtimes.

How is IBM involved?

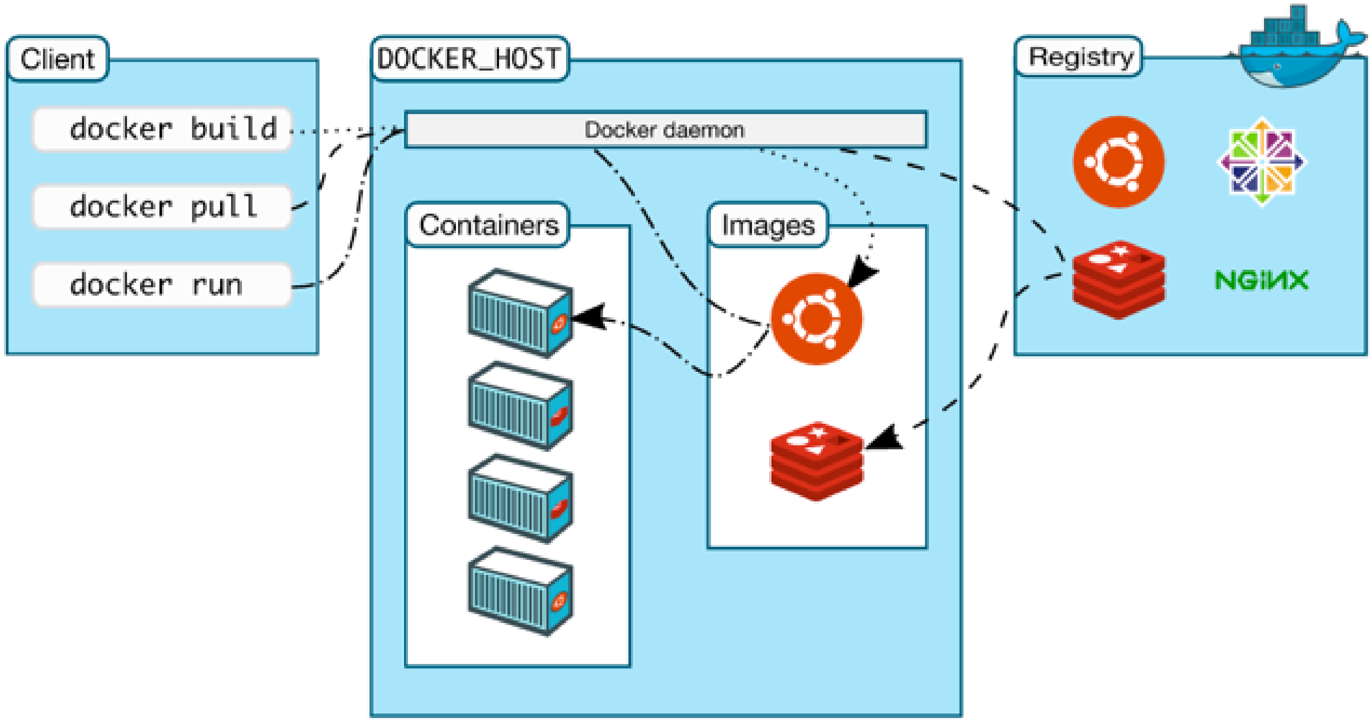
IBM CONTAINERS

- Fully managed **hosted runtime**
- **Integrated** logging, monitoring, load balancing, registry and auto-scaling
- Advanced **security** features
- Built using **Docker** technology

OPEN CONTAINER INITIATIVE

docker

What is Docker?



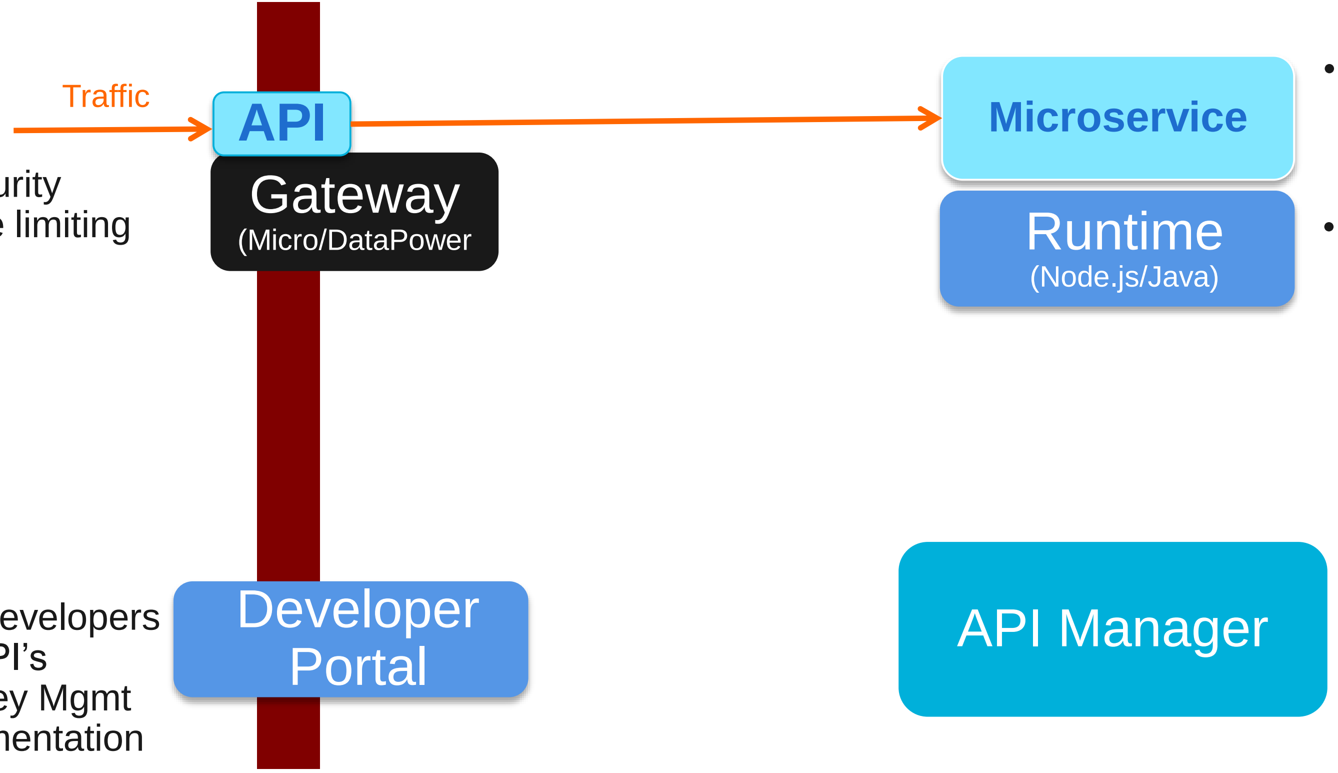
Source: https://docs.docker.com/engine/understanding-docker/#/what-is-dockers-architecture

Why Containers?

- Portable runtime environment for your applications
- No need to worry about missing dependencies, packages, and other concerns during deployments
- Each app is run in it’s own isolated container, and as a result you can run different versions of libraries for each app
- You can automate testing, integration, packaging of your application
- Consistent and repeatable process for app lifecycle
- No more inconsistencies between dev, test, production
- Speeds up continuous deployment and continuous integration

API Connect

API Connect Concepts



Traffic

API

Gateway
(Micro/DataPower

- Security
- Rate limiting

Microservice

- The “brains” of the API. Does all the work.

Runtime
(Node.js/Java)

- Handles HA for the microservice

Developer Portal

- How developers find API’s
- API Key Mgmt
- Documentation

API Manager

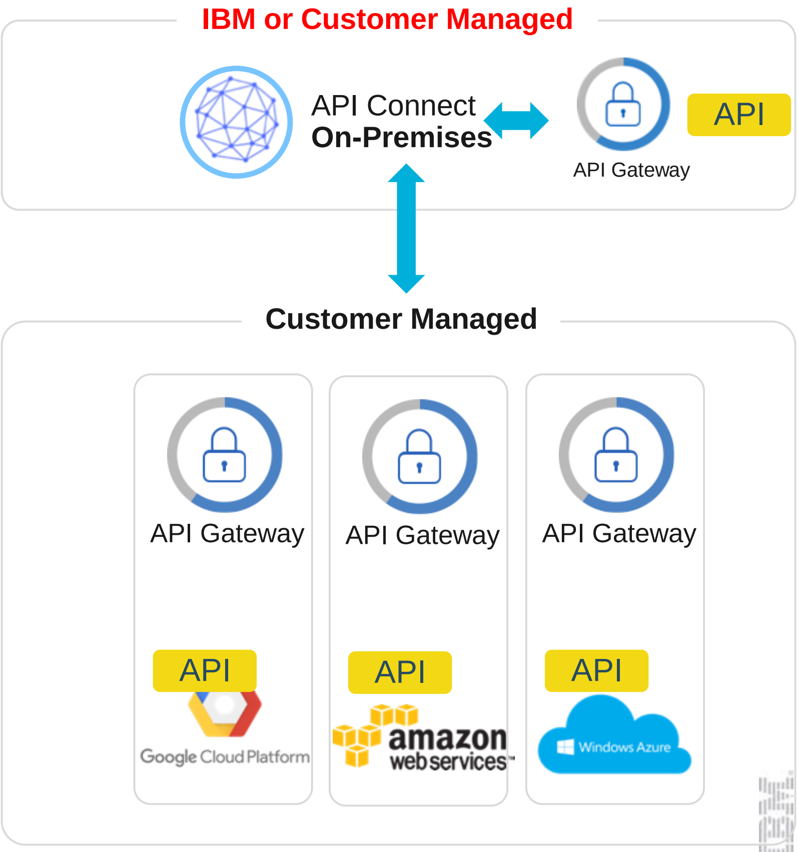
- How API owners manage API’s
- Access control
- Permissions
- Monitoring
- Analytics

Hybrid Deployment – APIC with Co-located Gateways

- Deploy API management, analytics and socialization developer portal **on-premises** for enhanced corporate control

- **Co-locate** DataPower API Gateway with backends on-premises and/or public clouds for greater flexibility and performance

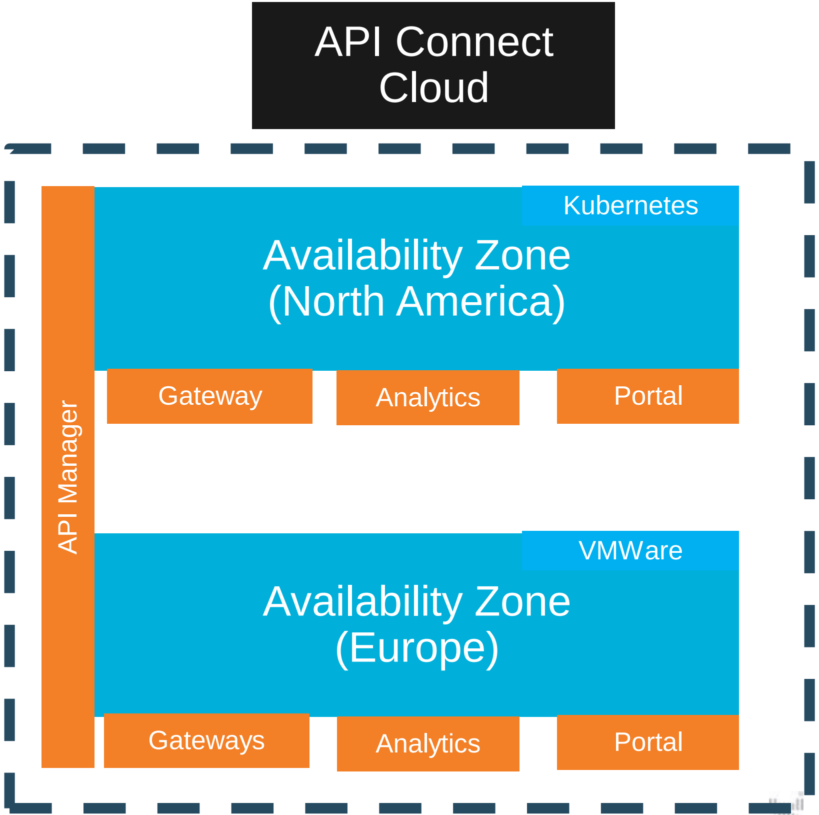
- Speed deployment of co-located API Gateway and reduce risks with **DataPower support for Docker**



Roadmap

API Connect Cloud (1 of 2)

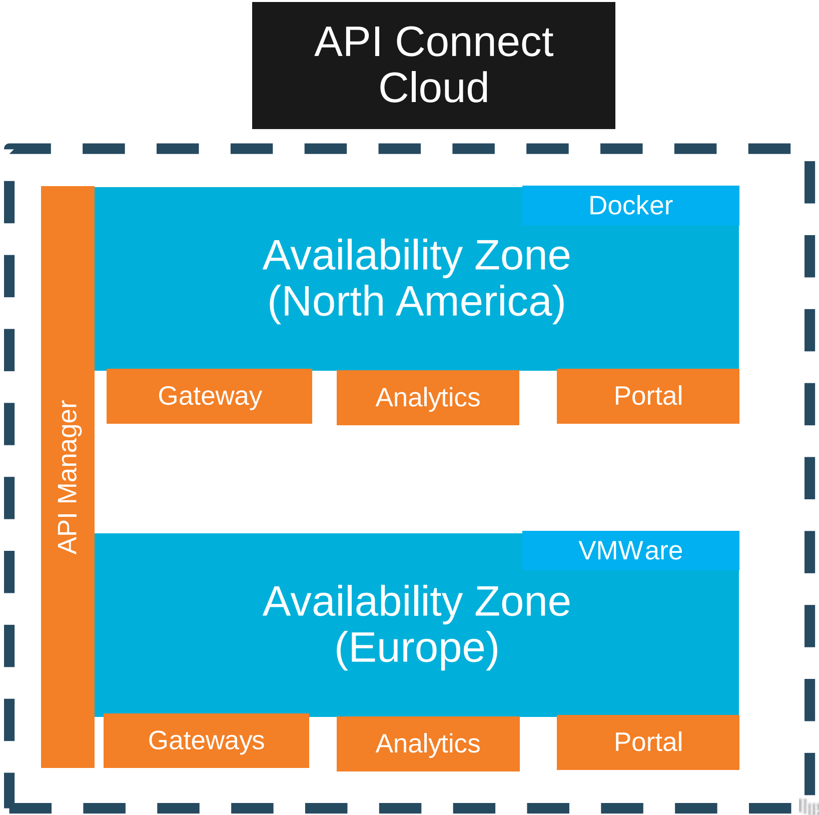
- Single API Connect Cloud consists of **four core services**:
  - API Manager
  - Gateways
  - Analytics
  - Portal
- API Manager is a **single cluster** across multiple availability zones (AZ)
- Multiple Gateway/Analytics/Portal clusters are available within an AZ
- Gateways can bind to a specific Analytics service when co-located



Roadmap

API Connect Cloud (2 of 2)

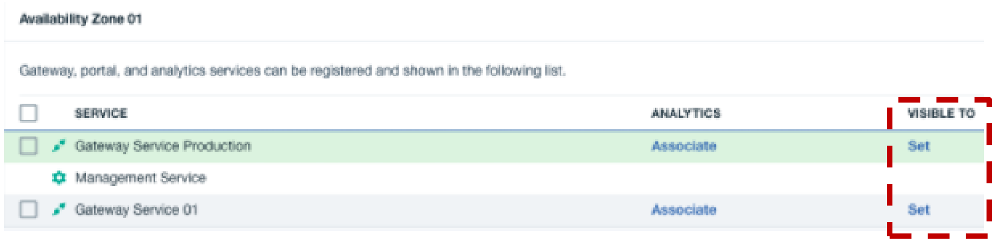
- Each **Availability Zone defines a deployment target** (Docker or VMWare)
- Docker:
  - Initial delivery focuses on deploying services using **Kubernetes**.
  - AZ **discovers** Services (gateway/analytics/portal) from the Kubernetes platform
- VMWare
  - Packaged as OVA file
  - Admin **manually** Register services (gateway/analytics/portal)



Roadmap

API Connect Cloud services constrained to provider organizations

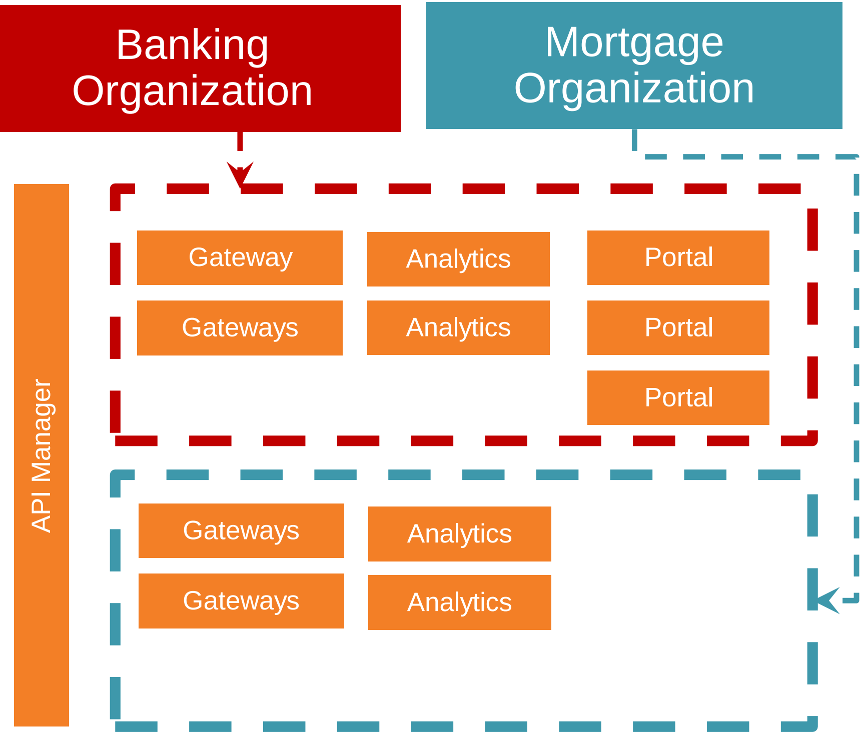
- Cloud administrators can restrict access to services (Gateway/ Analytics / Portal) to specific provider organizations



Availability Zone 01

Gateway, portal, and analytics services can be registered and shown in the following list.

□	SERVICE	ANALYTICS	VISIBLE TO
□	Gateway Service Production	Associate	Set
	Management Service		
□	Gateway Service 01	Associate	Set



Banking Organization
Mortgage Organization
API Manager
Gateway
Analytics
Portal
Gateways
Analytics
Portal
Portal
Gateways
Analytics
Gateways
Analytics