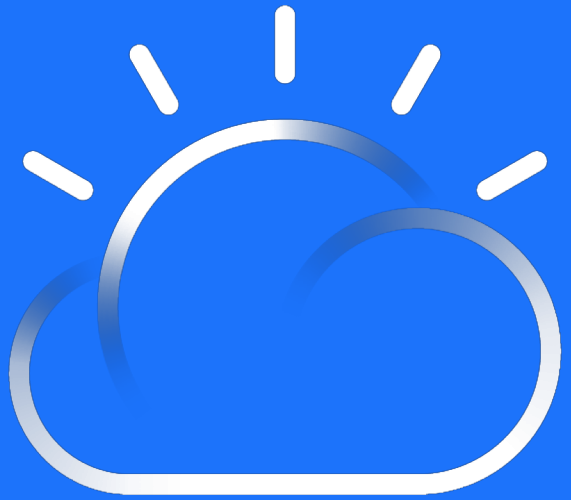
Integration Technical Conference 2019

Scaling Integration

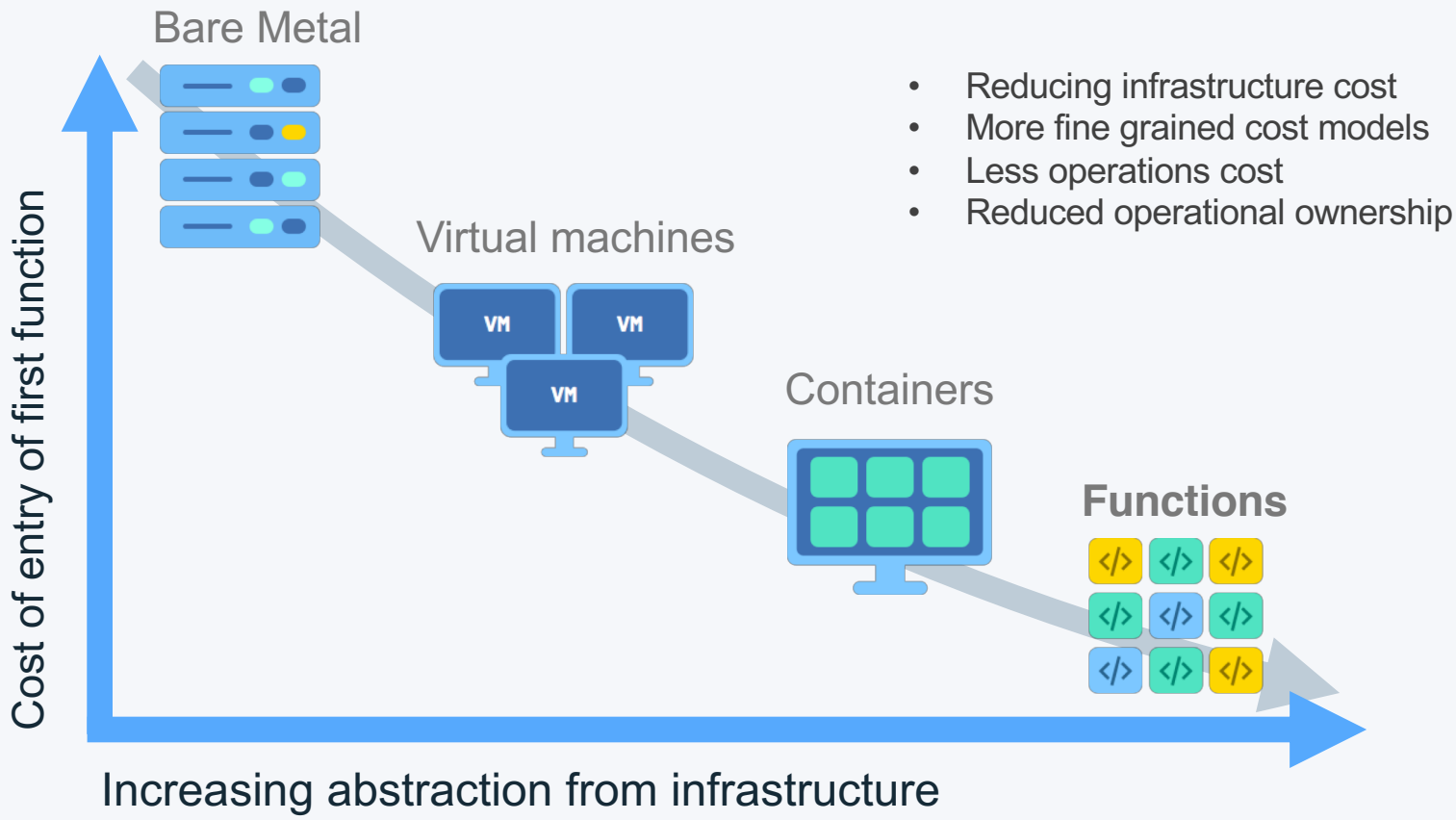
Kim Clark
Integration Architect

Rob Nicholson
Distinguished Engineer – Integration Portfolio



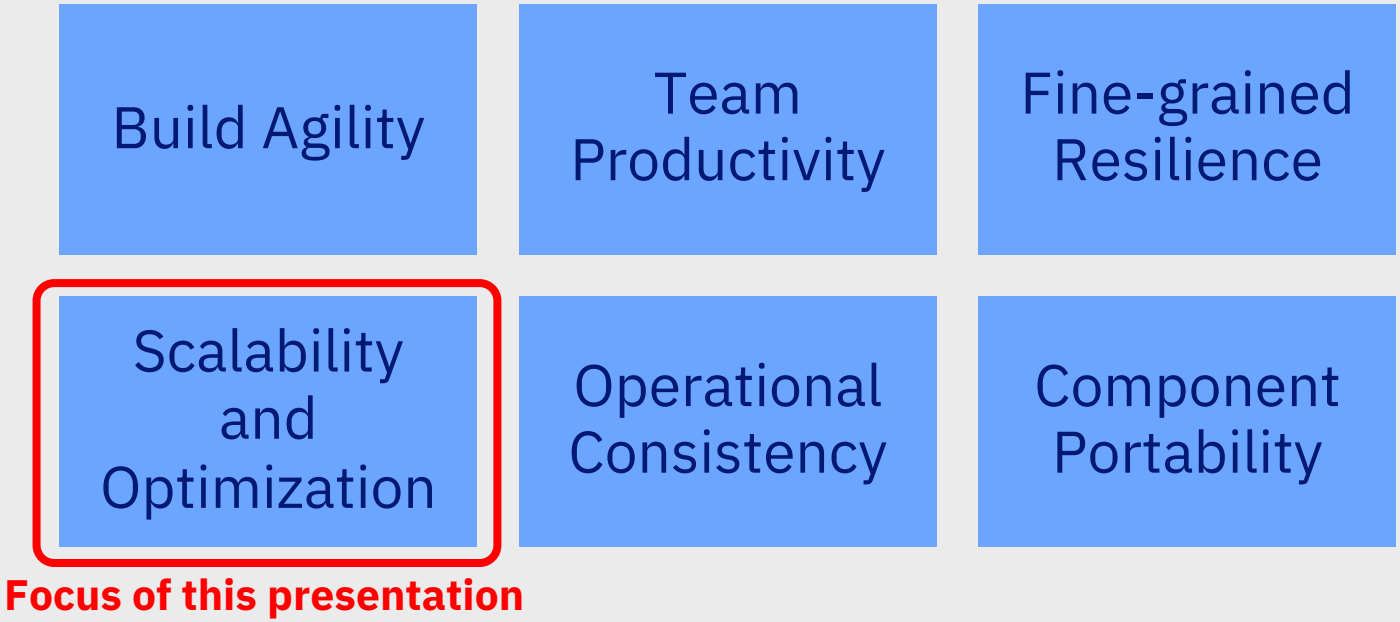
IBM **Cloud**

IBM



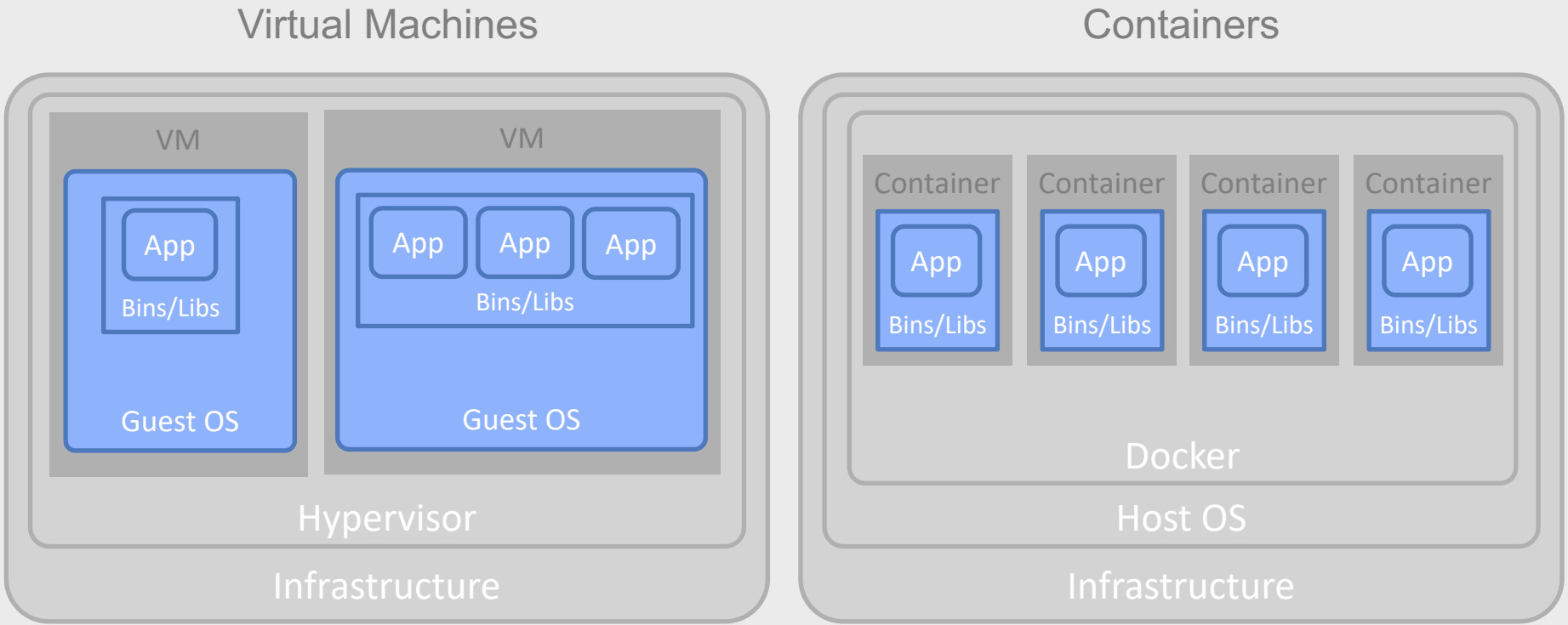
- Reducing infrastructure cost
- More fine grained cost models
- Less operations cost
- Reduced operational ownership

Benefits of a container based strategy



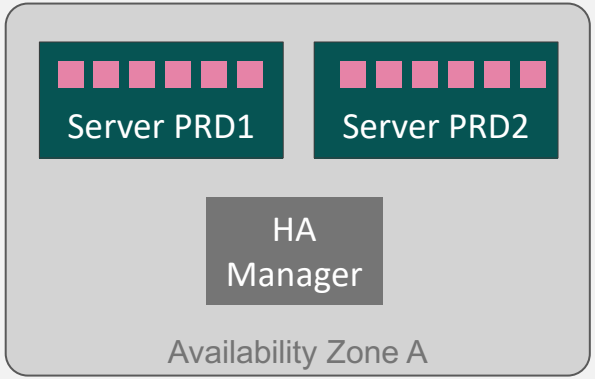
Containerization is more than just a re-platforming exercise. “lift and shift” will not bring the above benefits.
Requires: *Fine-grained deployment, organizational decentralization, pipeline automation, disposable components…*

Difference between virtual machines and containers

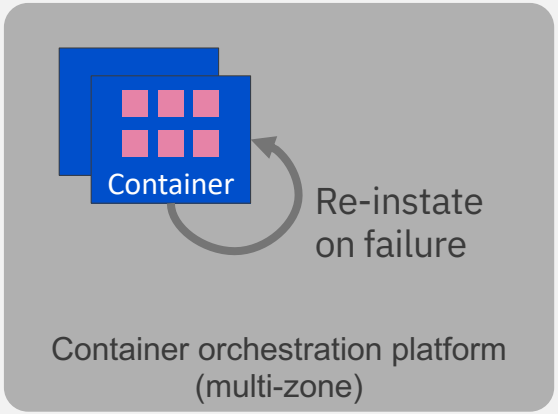


Derived from https://www.docker.com/what-container

What's the container equivalent of the HA/DR topology you have today?



Traditional
(explicit configuration)

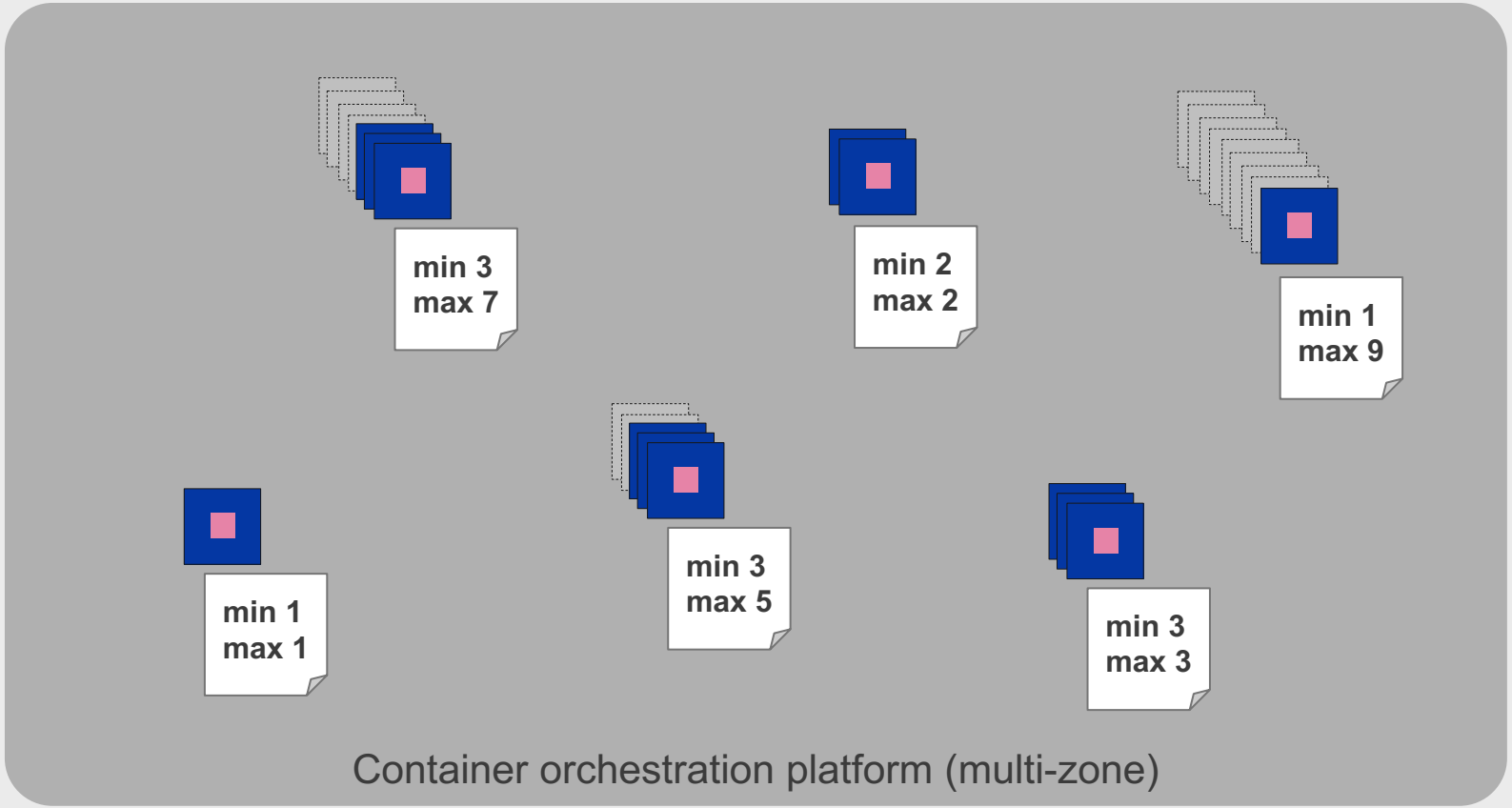


Infrastructure as code

Replication:
minimum 2
maximum 2
Spread across zones
Balance workload evenly

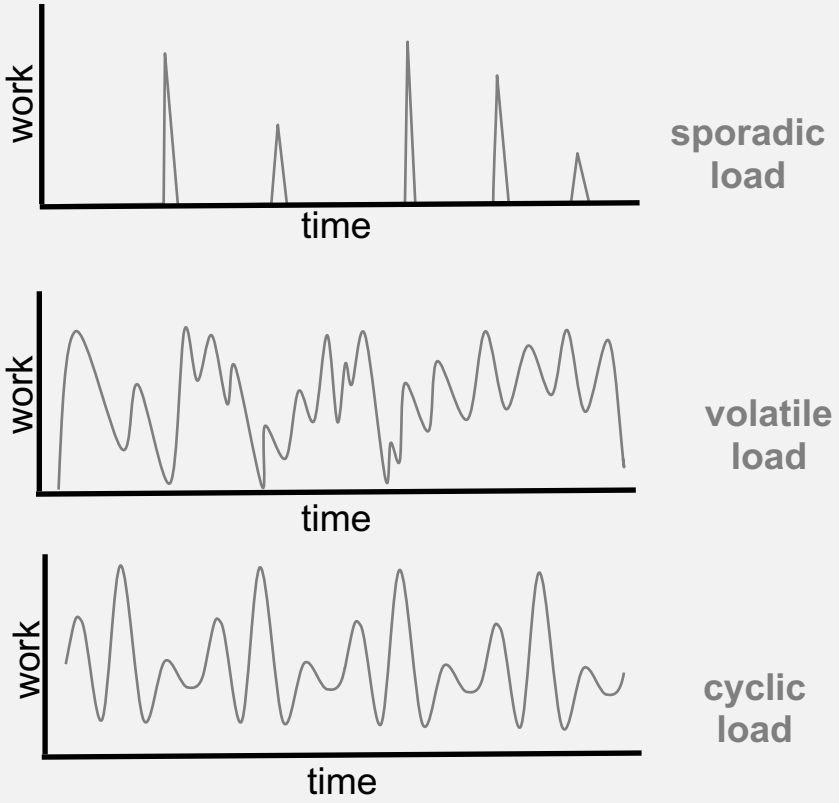
Container platform
(declarative logical configuration)

Containers enable fine-grained deployment...and discrete scaling policies with near-complete abstraction from physical resources



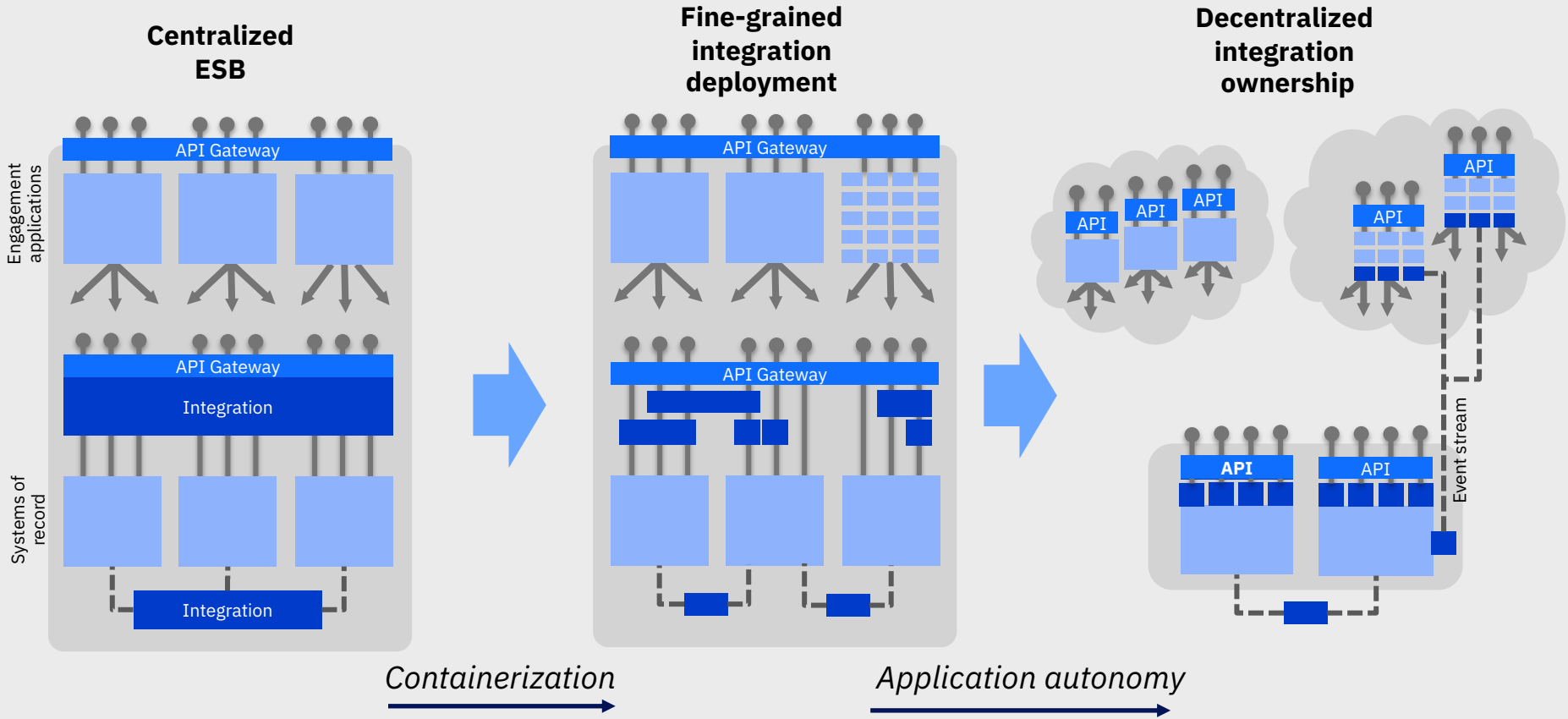
Use cases benefiting from elastic scaling

- Typical use cases in production
  - Rarely used functions
  - Functions with high workload variation
  - Elastic parallelization of batch
- Use cases beyond production
  - Prototyping
  - Performance testing



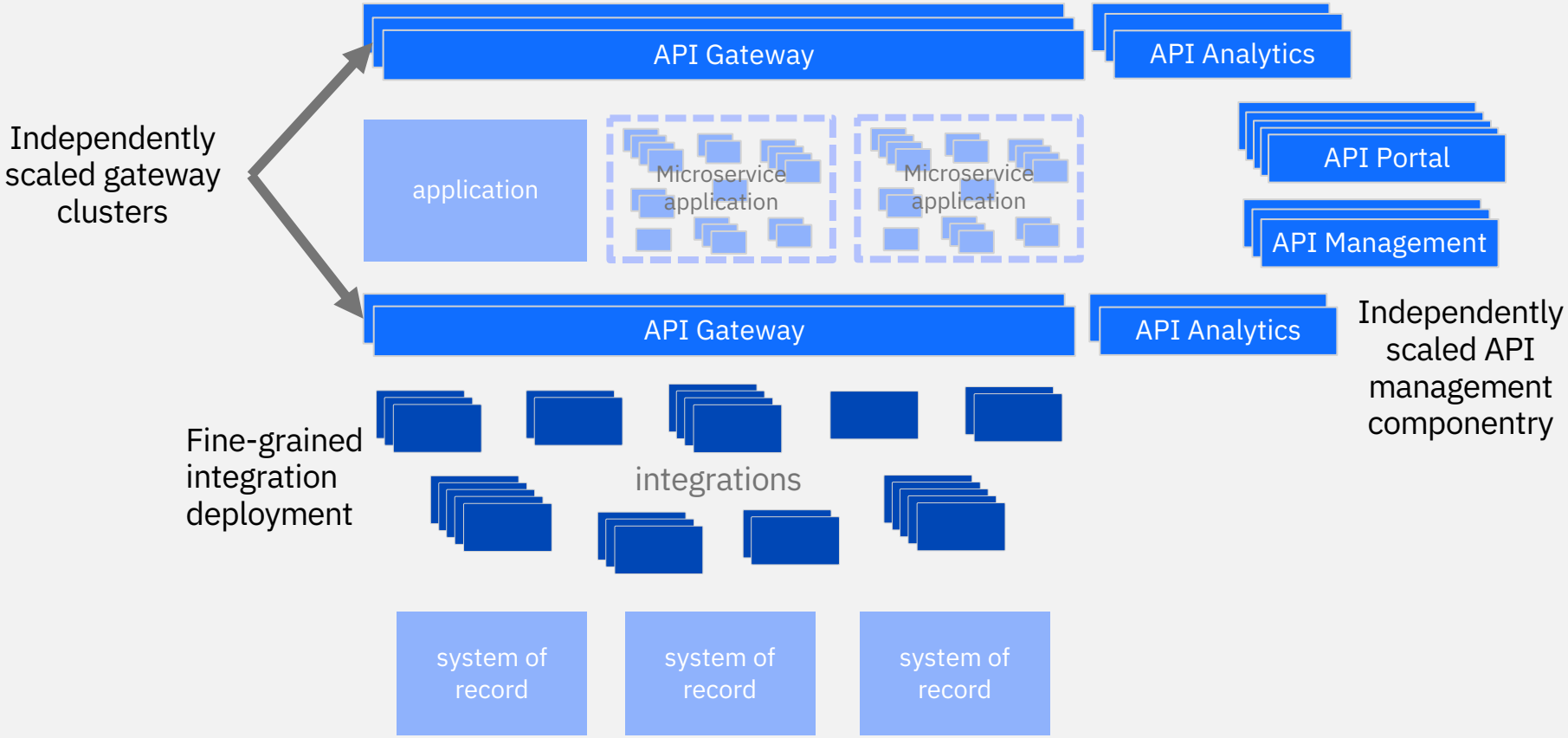
Almost all use cases have variable load at the individual function level

Evolution to **agile integration architecture** – detail view

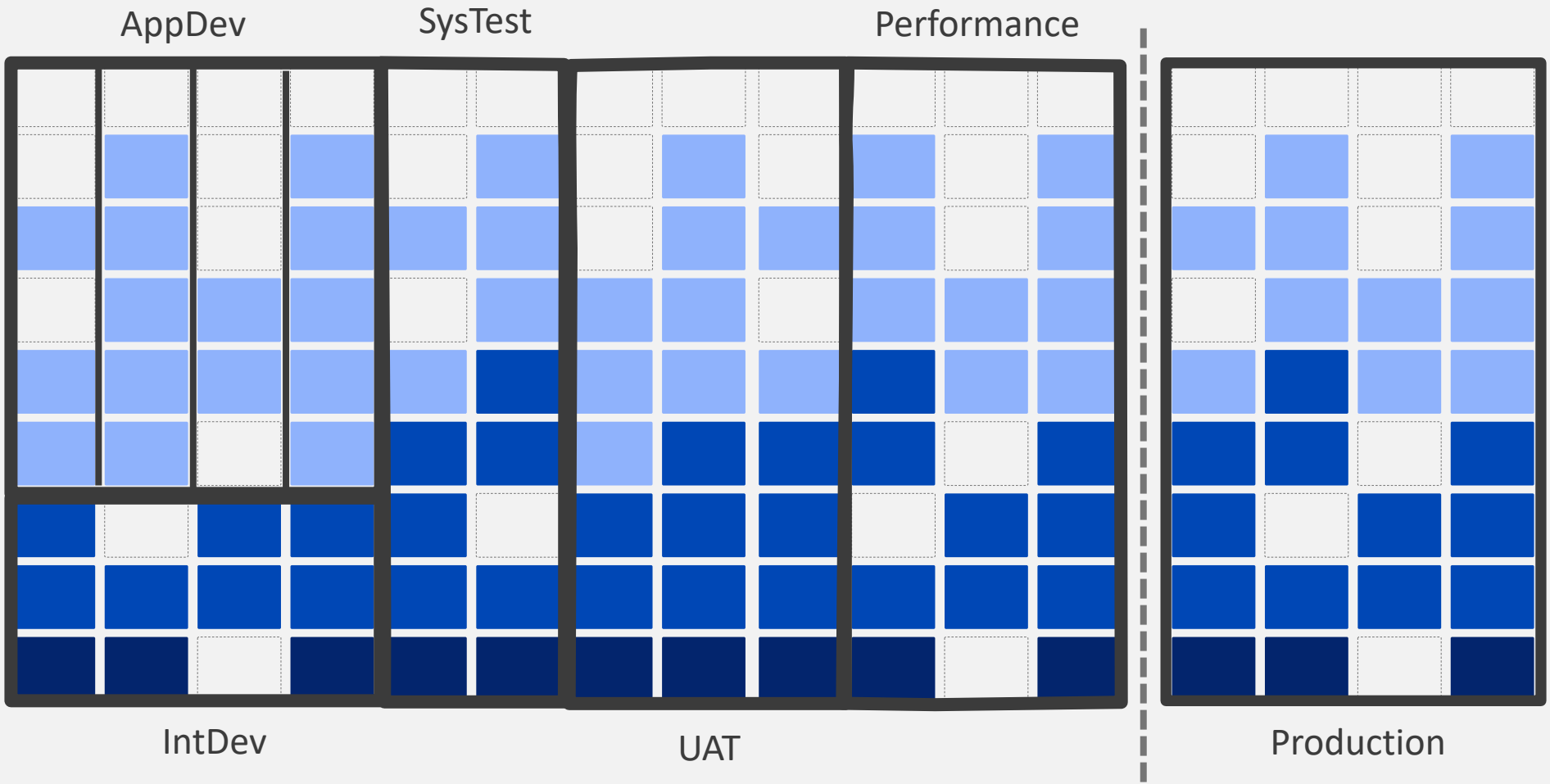


Webinar http://ibm.biz/AgileIntegArchWebinar
eBooklet https://www.ibm.com/cloud/agile-integration-architecture

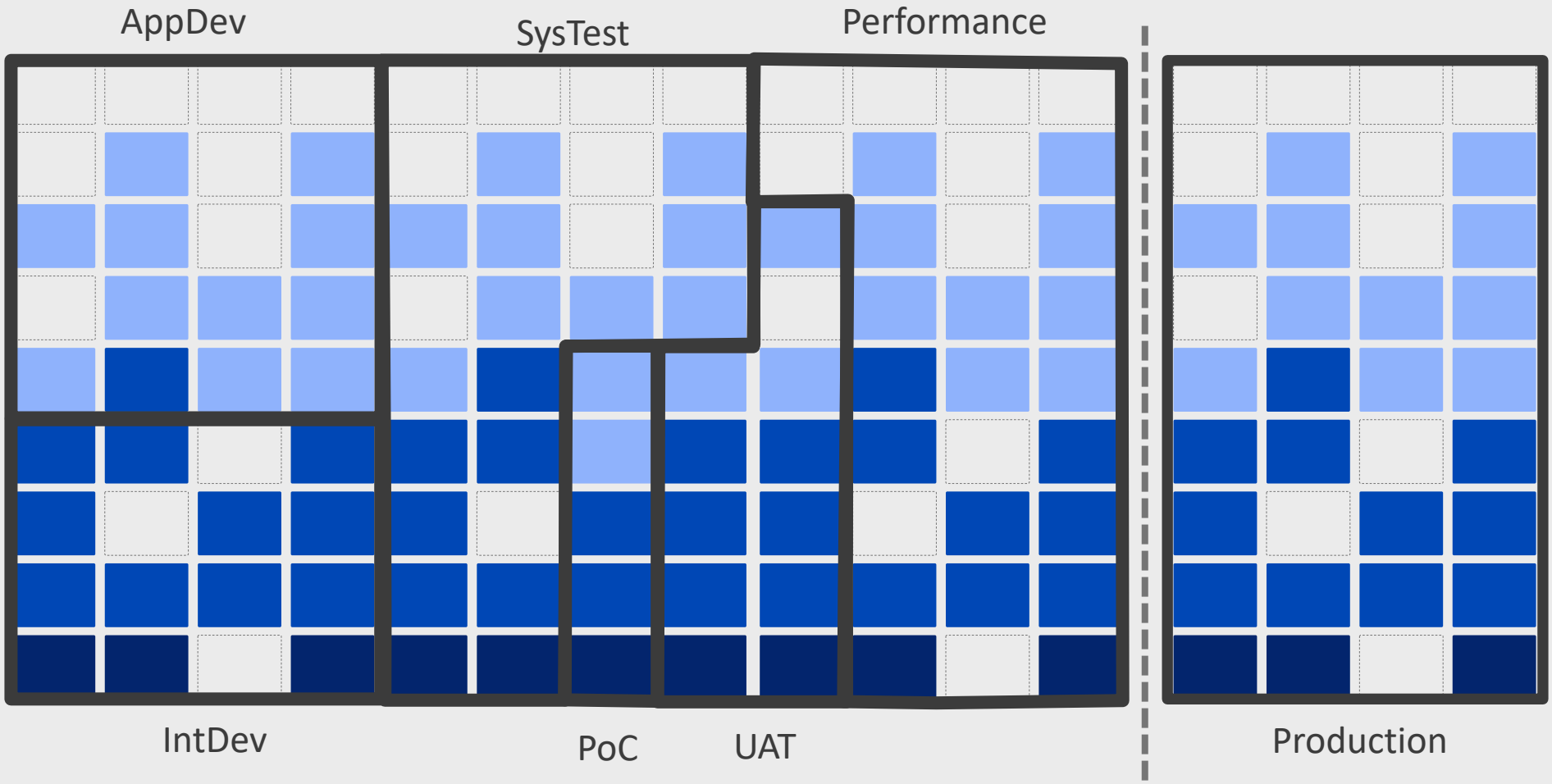
Capabilities scale in different ways



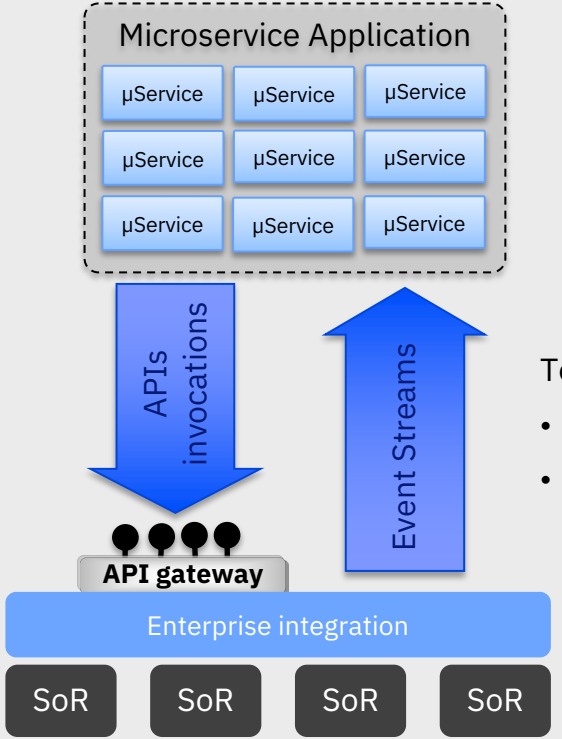
Scaling across environments



Scaling across environments



Creating truly independent digital applications requires asynchronous communications as well as APIs



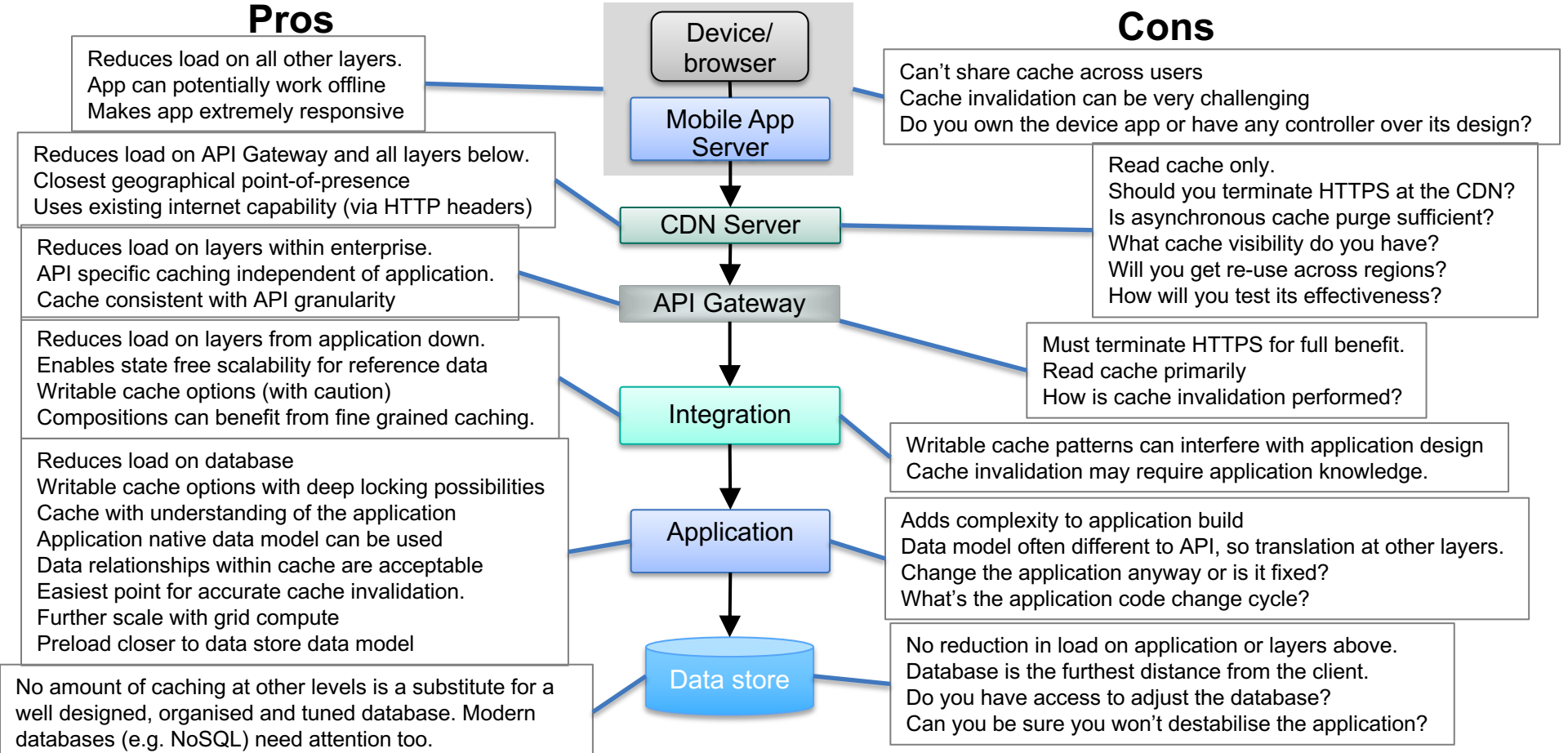
Truly independent, decoupled microservice components enable

- ***Agility***: Innovate rapidly without affecting other components
- ***Scalability***: Scale only what you need, and only when you need to
- ***Resilience*:** Fail fast, return fast, without affecting other components

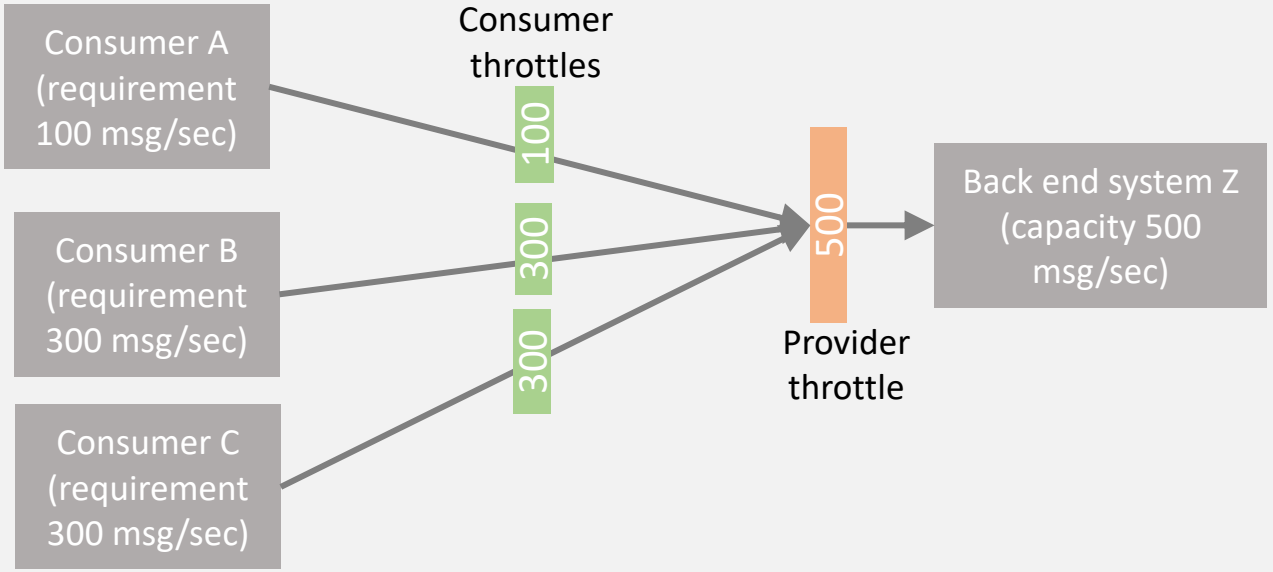
To provide those benefits they need to be independent of the systems of record

- ***APIs***: Are simplest to use, but create a real-time dependency
- ***Event streams:*** *Enable microservices to build decoupled views of the data*

Caching – but in which layer?



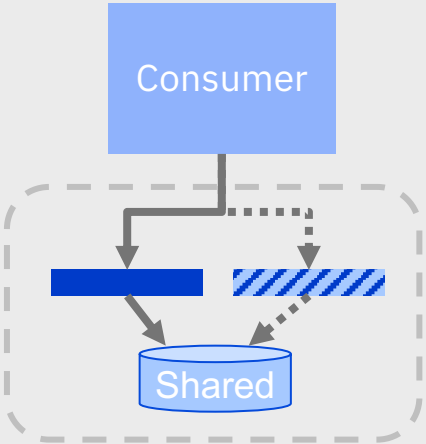
Managing throughput effectively requires multiple throttling points



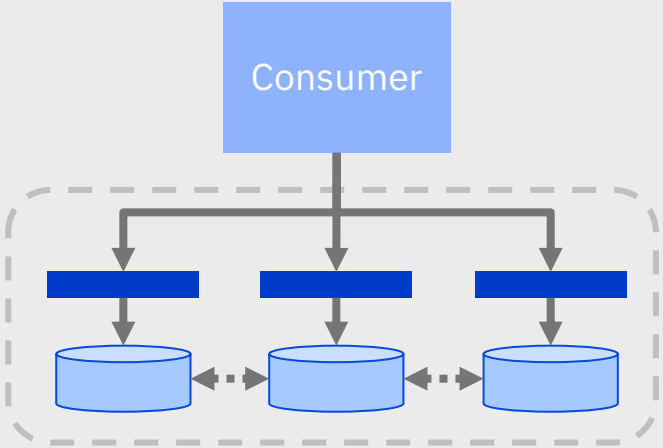
- No throttling – back end system easily overloaded.
- Provider only throttling – Back end system protected, but consumers can still take more than they are contracted for.
- Consumer only throttling – Enables prioritisation of consumers. Aggregate of consumers must be limited to back end capacity or risk overloading.
- Consumer and provider throttling – Consumers forced to behave to SLA, and back end system protected.

https://developer.ibm.com/articles/mw-1705-phillips

Simplified scaling and high availability



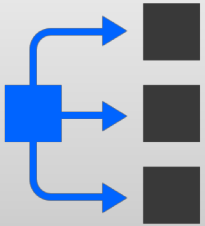
Active/passive with shared data
One master available at a time
Failover requires detection and warm up
Dependent on shared persistence



Active/active with duplicated data
Horizontal scaling
Instant failover
Replicated local persistence

Events
(notifications)

Stream History

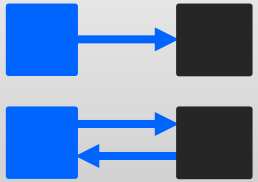


Scalable Consumption

Immutable Data

Messaging
(commands)

Transient Data Persistence



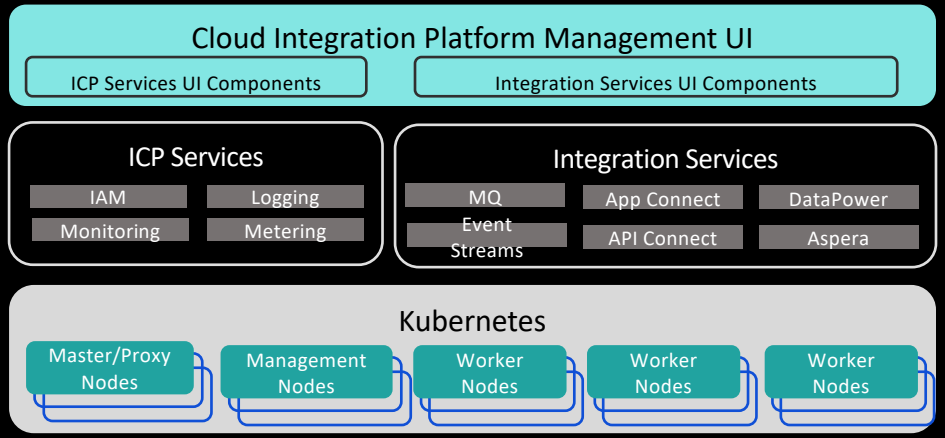
Source / Target Request / Reply

Targeted Reliable Delivery

https://developer.ibm.com/messaging/2018/05/18/comparing-messaging-event-streaming-use-cases

Cloud Integration Platform Architecture

- Cloud Integration Platform (CIP) deploys and manages instances of **Integration Services** running on a Kubernetes Infrastructure
- Instances of Integration services are deployed individually as needed to satisfy each use-case.
- Services can be deployed in **Highly Available** topologies across multiple Kubernetes worker nodes or non HA on a single worker.



Scalability for Integration services.

Integration Services run on the **worker nodes**.

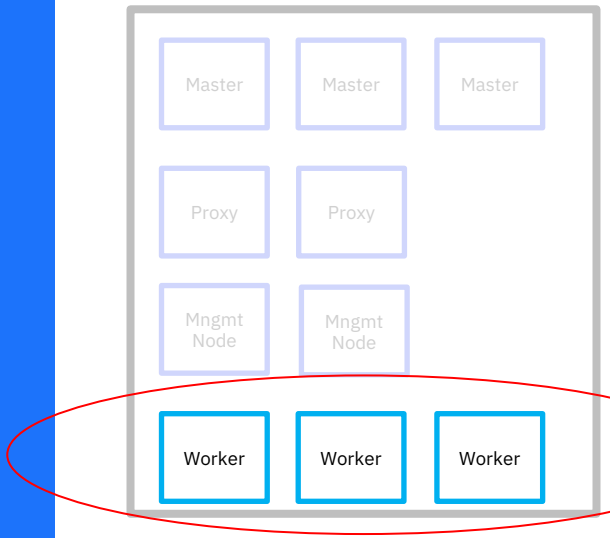
The Cloud Integration **SolutionPak** is composed from the **CloudPaks** making from the component products.

IBM cloudpaks are developed by the product development teams to embody best practice for deploying the product onto kubernetes as a secure, **scalable** Highly Available deployment.

Typically, all you need to do is scale the number of instances of each service and the CPU and memory it is allotted.

Kubernetes takes care of:

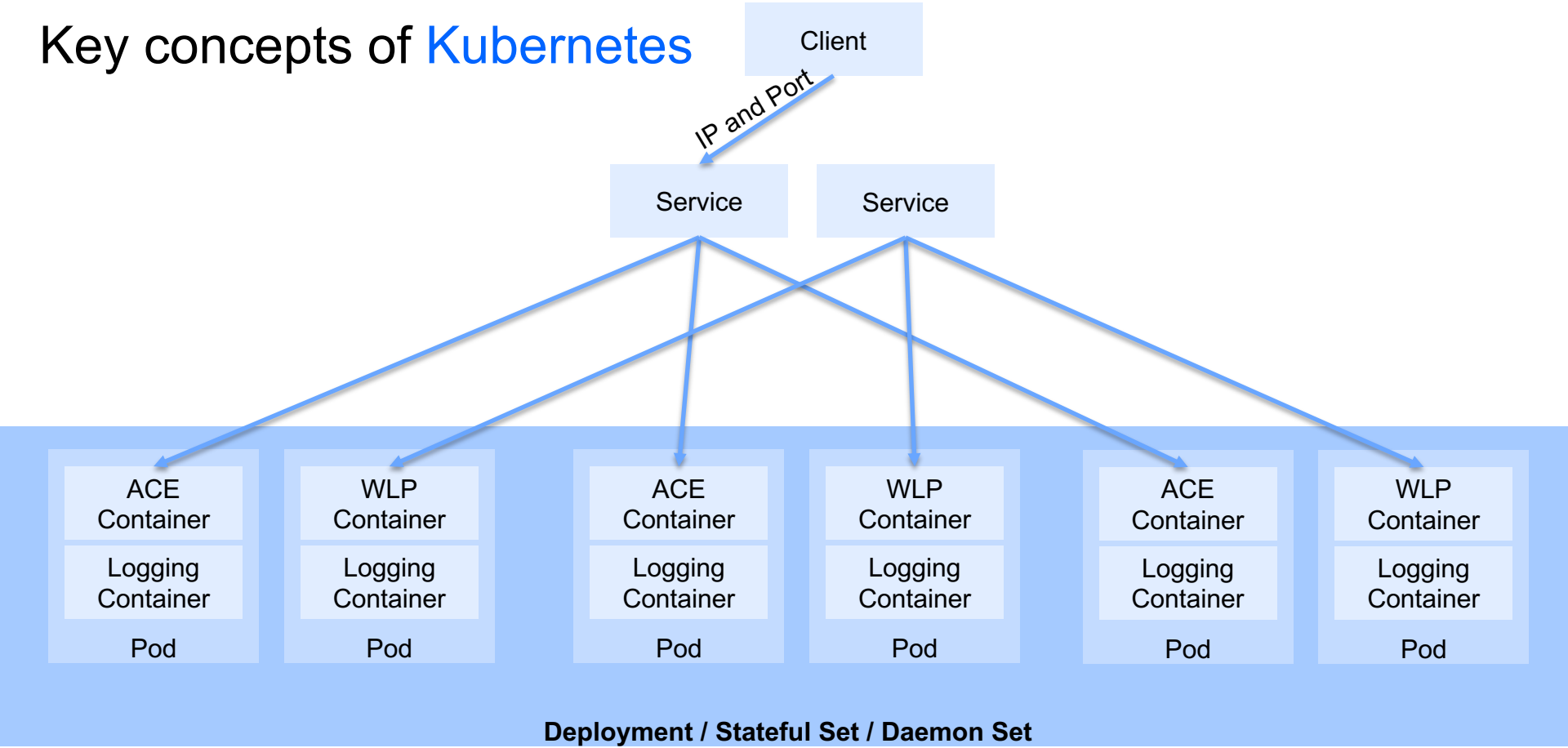
- Running appropriate numbers of instances of the solutionpak, spread across the available workers.
- Mixing together the workloads on the available worker nodes taking account the resources they need.
- Restarting workloads that fail and recovering from failed nodes by scheduling workloads onto alternative nodes.



Cloud Integration Platform HA Cluster

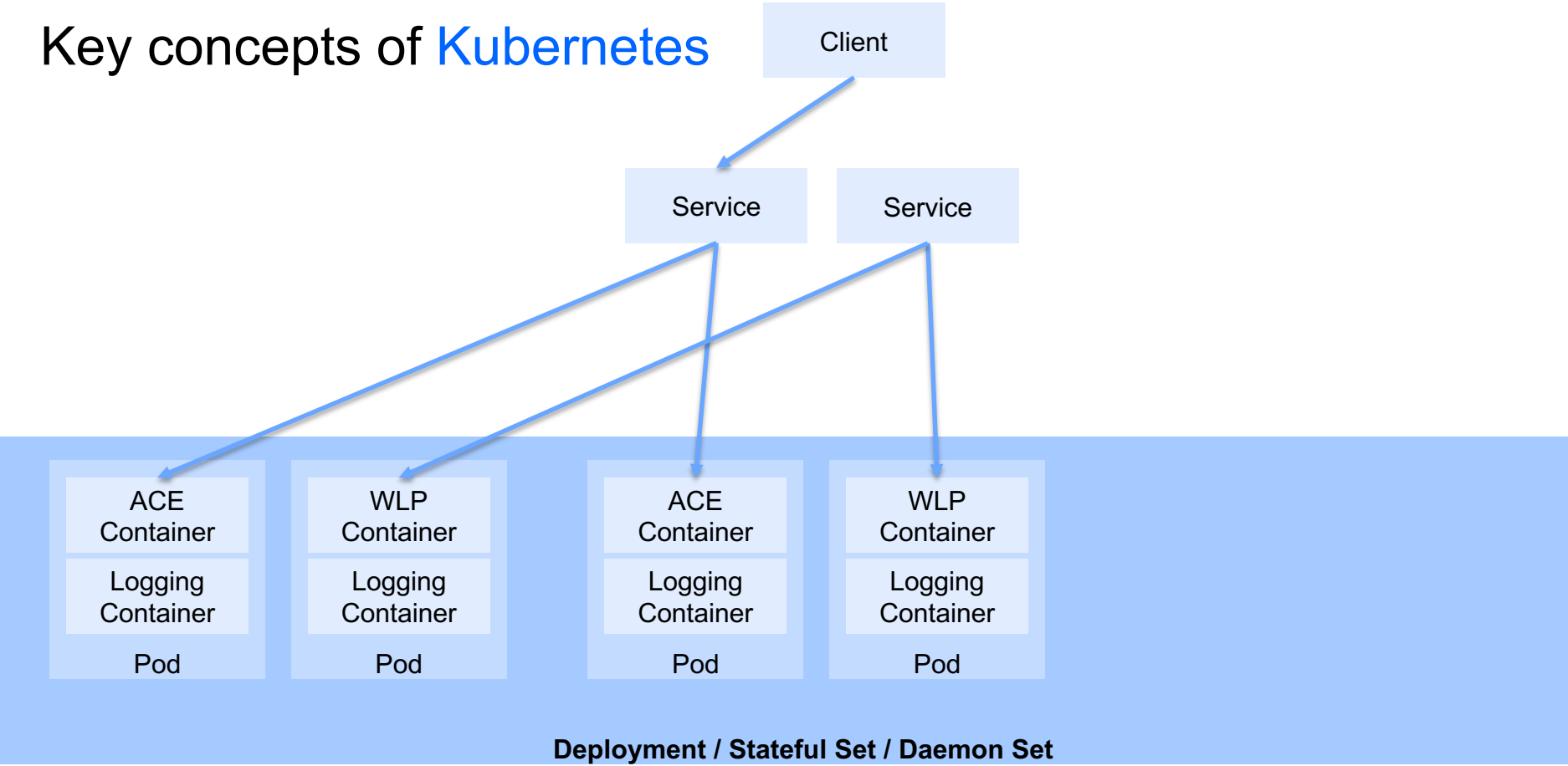
* Some of the component products require 2 worker nodes for active/standby syle HA and others require 3 or more workers for quorum style deployment.
** In larger clusters the nodes should be spread between 3 ore more availability zones.

Key concepts of Kubernetes



Replica Total of 3

Key concepts of Kubernetes

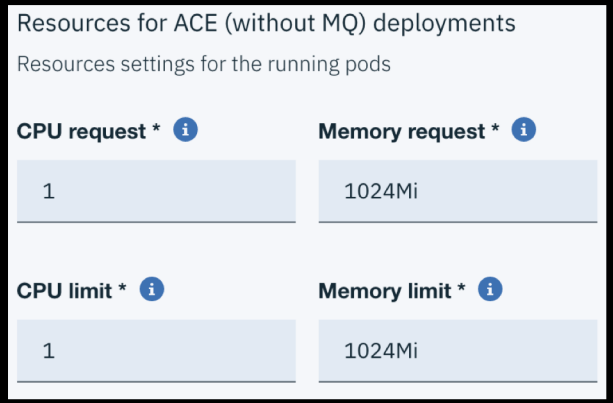


Scaling options

- **Horizontal scaling** – change the number of instances
  - Typical scaling approach used in Kubernetes.
- **Vertical Scaling** – Change CPU / Memory limits.
  - Requests
    - Requests.cpu
    - Requests.memory
    - Used at scheduling time. Pods are scheduled to nodes that have ‘unrequested’ resource.
      - Pod will not be scheduled to a node that is ‘full’
  - Limits
    - Limits.cpu
    - Limits.memory
    - CPU– affects CPU allocated to pod **when node is busy**
    - Memory – Pods that exceed their memory limit likely to be terminated

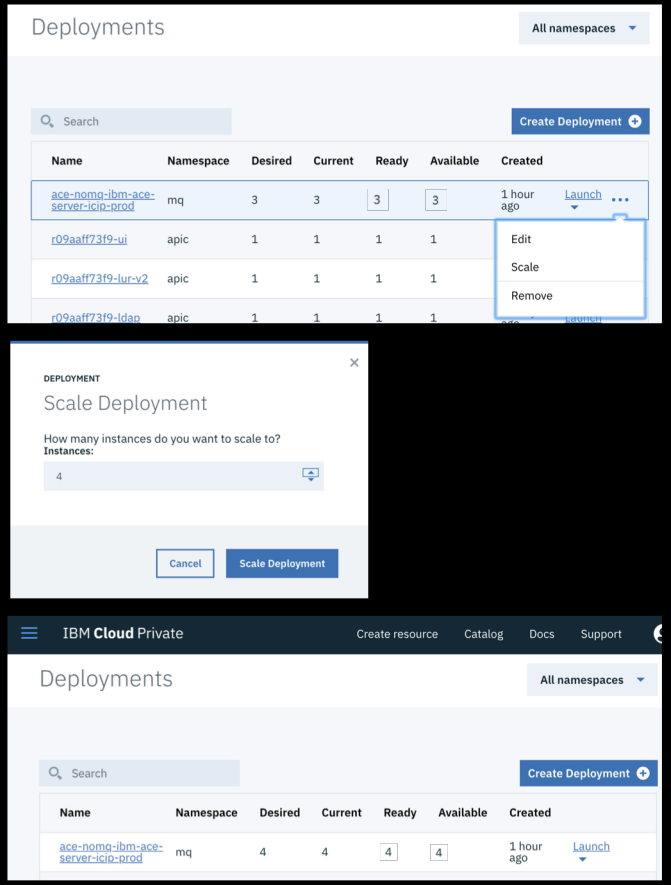
Setting scale at deployment time

- Each of the helm charts includes replica count, CPU and memory limits for their subsystems.



Scaling manually after deployment

- Deployments and replica sets can be scaled manually.
- Kubernetes creates new pods and schedules them where resources are available.

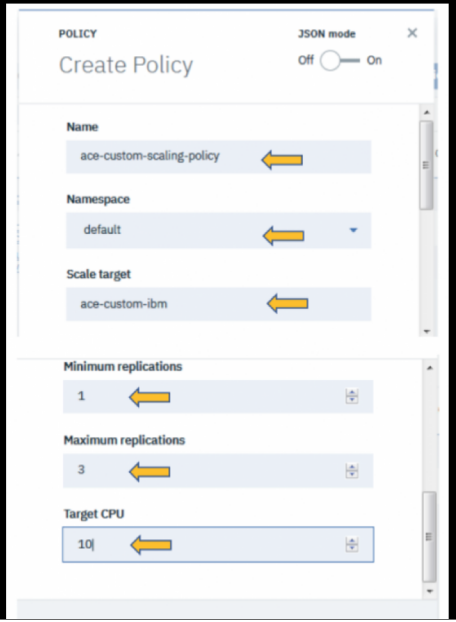


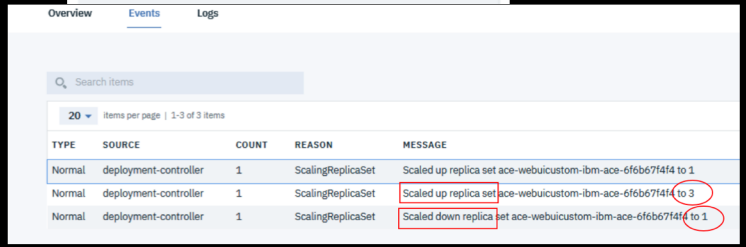
Scaling automatically based on CPU Load.

- Kubernetes scaling policies can be set based on CPU Load.
- Watch CPU load on pods and scale services up and down between defined min and max.

- Today this is based on CPU Load.
- ICP has experimental capability to based this on other metrics from the monitoring service/promethius

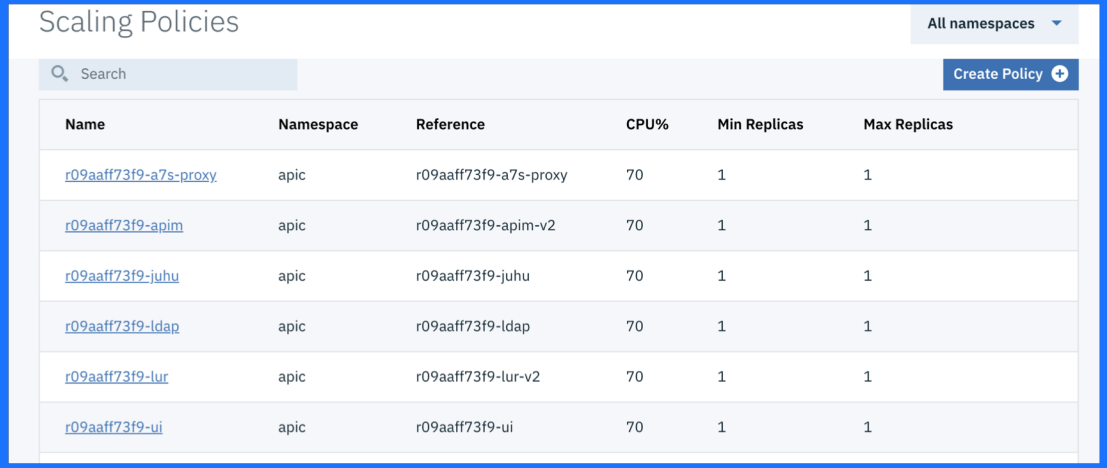
Reference:
https://developer.ibm.com/integration/blog/2018/07/27/customizing-docker-images-helm-charts-iibace-deployment-icp-assigning-scaling-policies-pods/





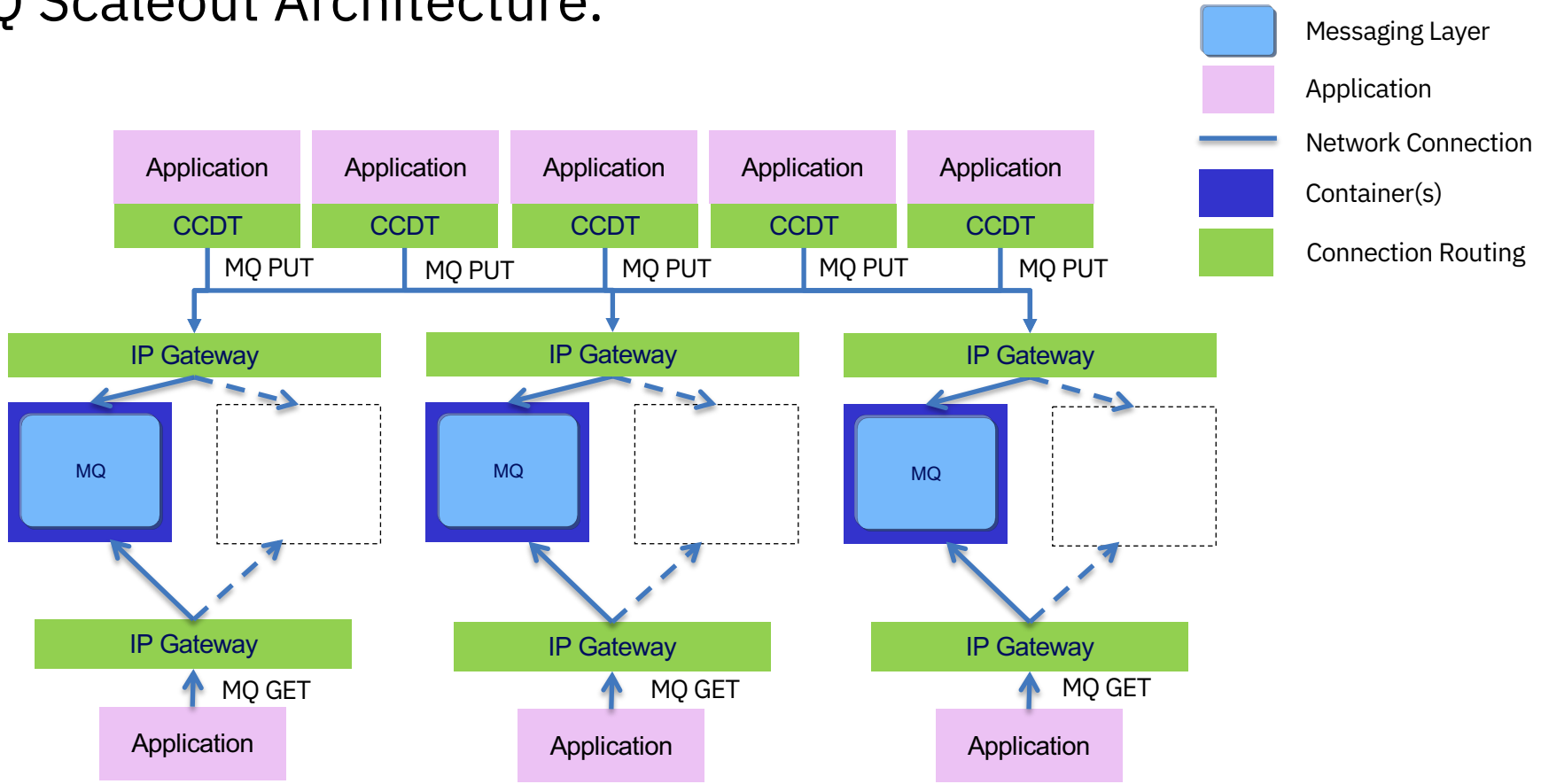
API Connect scaling policies.

- API connect creates scaling policies as part of a standard deployment.
- These can be adjusted to suit the organization needs.



Name	Namespace	Reference	CPU%	Min Replicas	Max Replicas
r09aaff73f9-a7s-proxy	apic	r09aaff73f9-a7s-proxy	70	1	1
r09aaff73f9-apim	apic	r09aaff73f9-apim-v2	70	1	1
r09aaff73f9-juhu	apic	r09aaff73f9-juhu	70	1	1
r09aaff73f9-ldap	apic	r09aaff73f9-ldap	70	1	1
r09aaff73f9-lur	apic	r09aaff73f9-lur-v2	70	1	1
r09aaff73f9-ui	apic	r09aaff73f9-ui	70	1	1

MQ Scaleout Architecture.

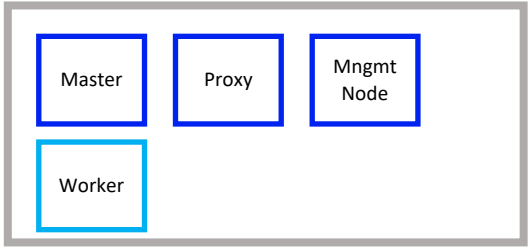


Container Platform Scalability

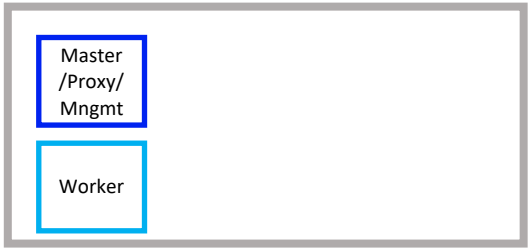
- Cloud Integration Platform is based on ICP Foundation.
- ICP Foundation is functionally identical to ICP Cloud Native edition. The only difference is licensing.
- All HA, scaling sizing and deployment information provided for ICP Cloud Native edition applies to Cloud Integration Platform.
- This presentation provides a summary. For more detail consult ICP Cloud Native edition documentation, articles and education. See the links page at the end of this presentation.

Scaling Kubernetes

- A kubernetes **Node** is a VM or bare metal machine which is part of a Kubernetes cluster.
- **Worker Nodes** run Integration Services. Enough CPU and memory are required to run all the scaled out Pods in your workload.
- **Master Nodes** run the services that control the cluster including the *etcd* database that stores the current state of the cluster.
- The load on the master nodes increases with the number of Nodes, Pods and other objects deployed. Master nodes must be scaled accordingly.
  - For very large clusters the etcd responsibility can be offloaded to searate etcd nodes
- **Proxy Nodes** transmit external requests to the services created inside your cluster. For large clusters use multiple proxy nodes and load balance across them with DNS or external load balancer.
- **Management Nodes** host management services such as monitoring, metering, and logging. As the number of worker nodes increases, management loads will scale proportionately.



Cloud Integration Platform
Cluster Node types



Cloud Integration Platform
Theoretical minimum cluster

Container Platform Scalability

	Node type	Number of nodes	CPU	Memory (GB)
Small Cluster	Master	3	16	32
	Management	2	8	16
	Proxy	2	4	16
	Worker	3+ (Max:20)	8	32
Medium Cluster	Master	3	16	32
	Management	2	16	32
	Proxy	3	4	16
	Worker	5+ (Max:70)	8	32
Large Cluster	Master node	3	16	128
	Management node	2	16	128
	Proxy node	3	4	16
	Worker node	500	8	32

Note that Master, management and proxy nodes are typically scaled UP before scaling out.

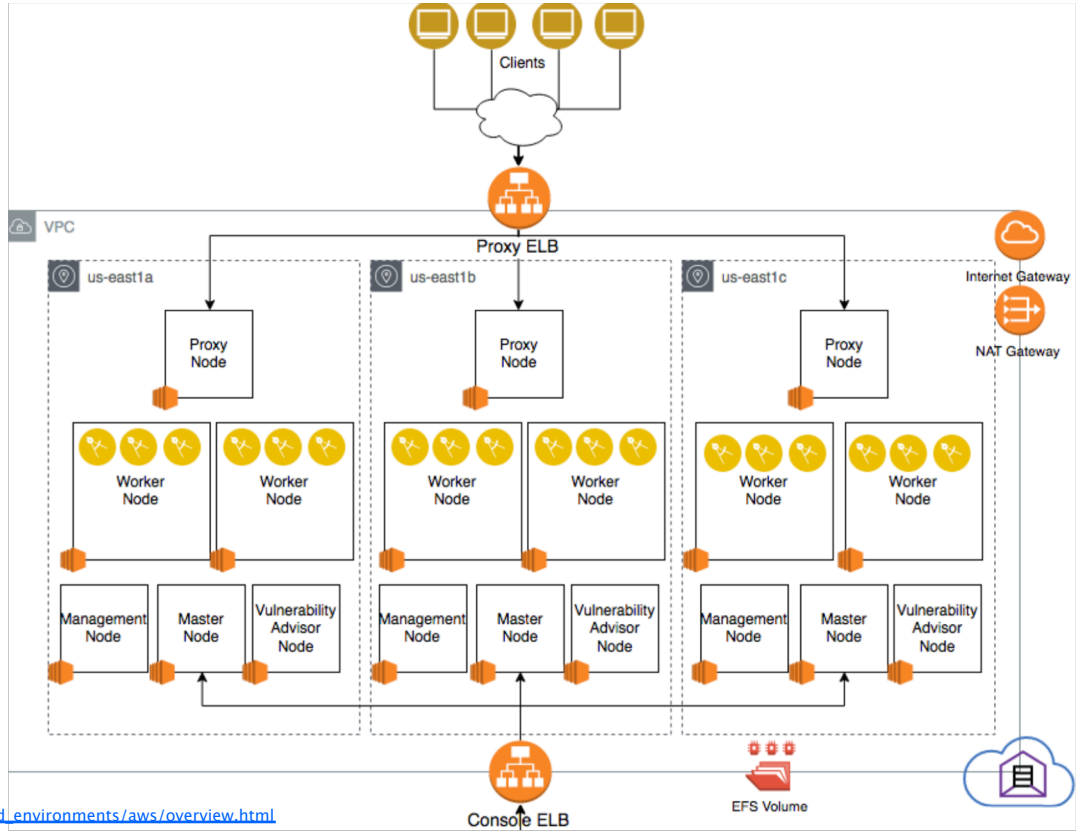
http://ibm.biz/icpcapacityplan

Example – CIP deployed in AWS

Leverage available zone
- Master/mgmt/va across available zone
- User application across available zone

AWS ALB/NLB
- Load balancer for management control plane
- Load balancer for user application
- Security group to control network access

EBS as persistent storage



Article: http://ibm.biz/icponaws
Documentation: https://www.ibm.com/support/knowledgecenter/en/SSBS6K_3.1.2/supported_environments/aws/overview.html
links to a quickstart: https://aws.amazon.com/quickstart/architecture/ibm-cloud-private/

Thank You