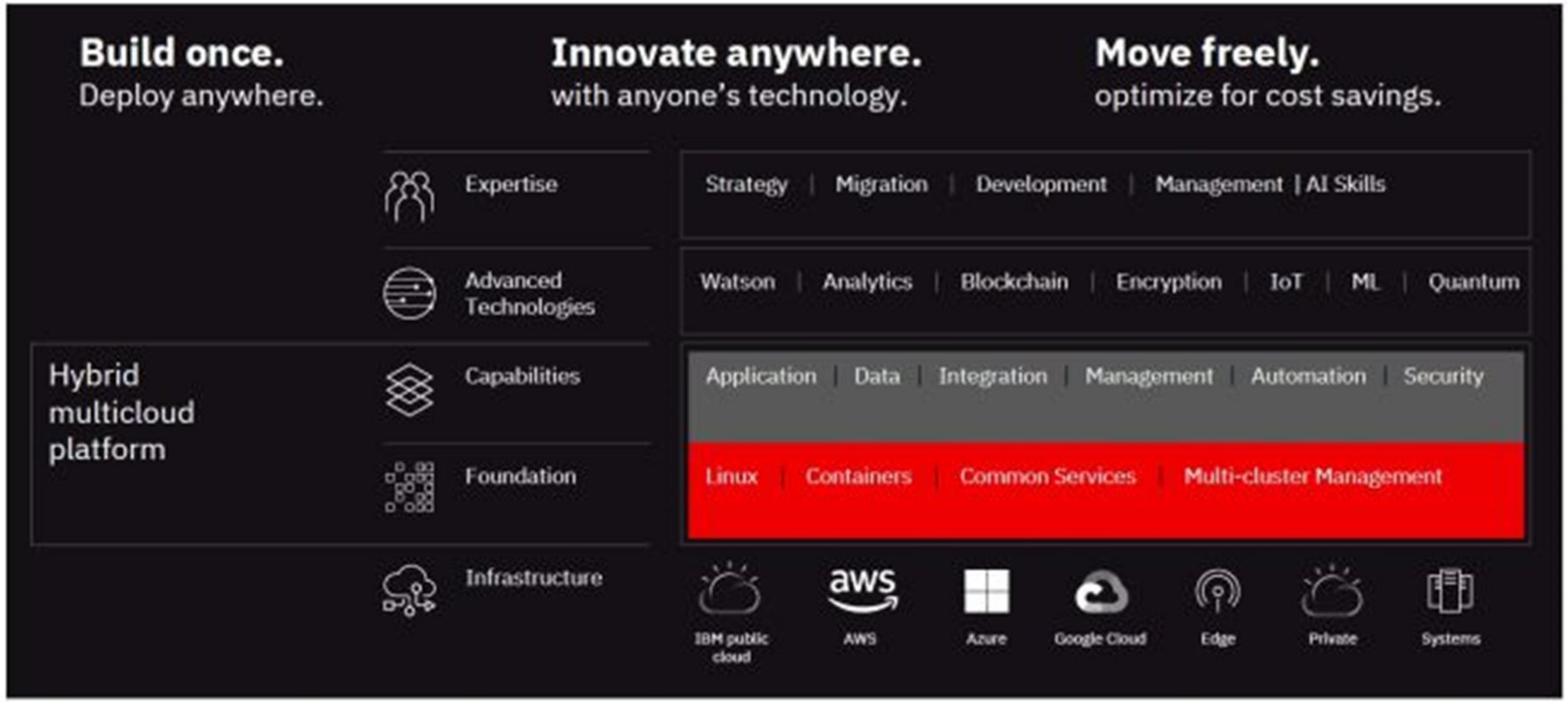
Multi Cloud – What It IS?



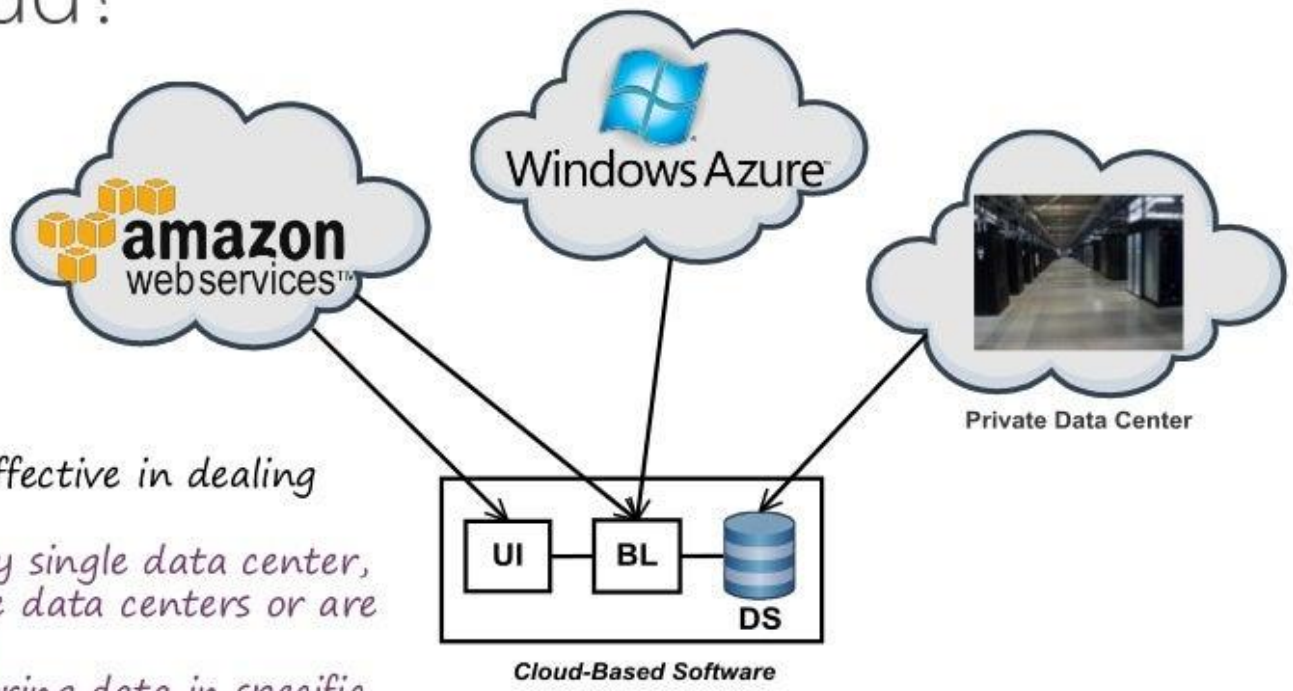
- Multi-cloud environment is capable of processing user demand and distributing work to resources deployed across multiple clouds.
- Multi-cloud denotes the usage of multiple, independent clouds by a client or a service.
- Hybrid deployment when an application is deployed in both on premise infrastructure as well as cloud platforms.
- A Federation is achieved when a set of cloud providers voluntarily interconnect their infrastructures to allow sharing of resources among each other

Why Multi-Cloud

Why Multi-Cloud?

The Multi-Cloud Deployment is effective in dealing with the following challenges:

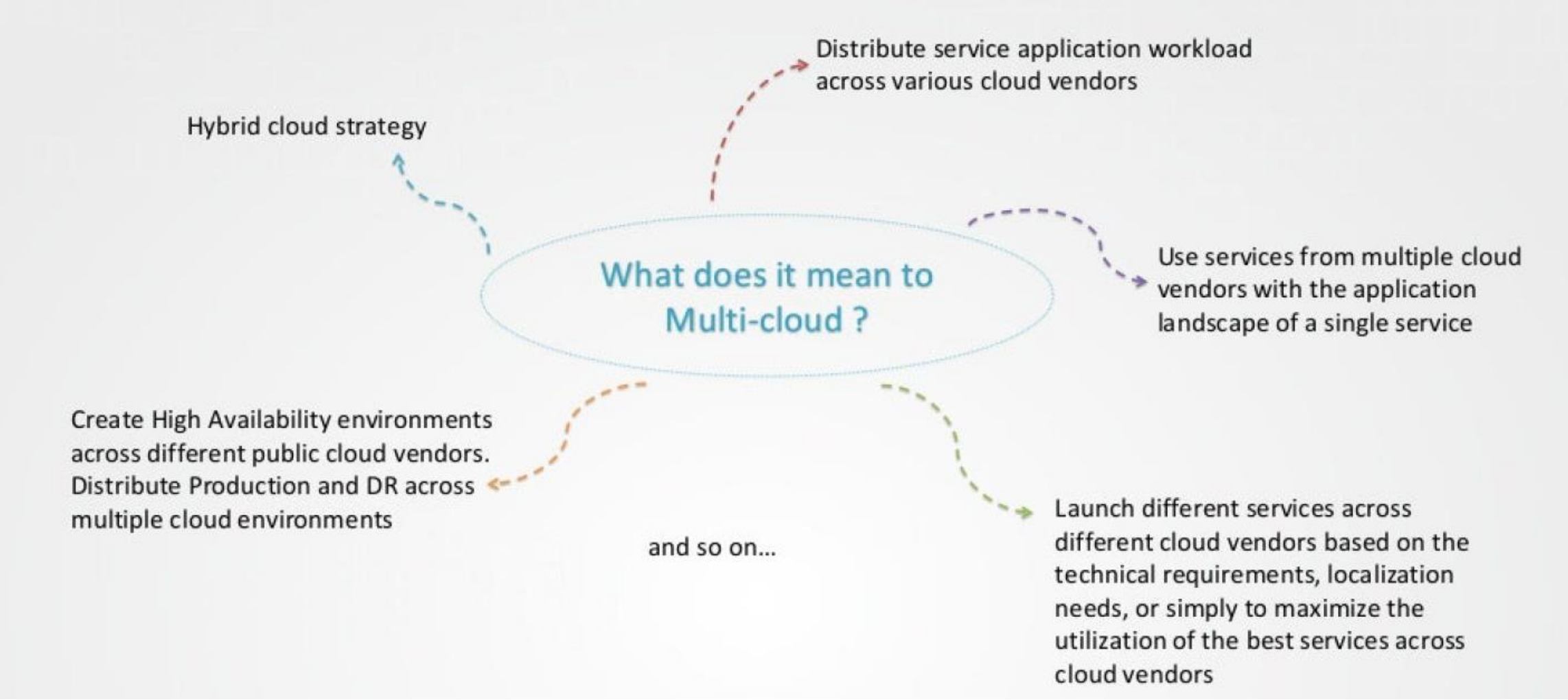
- Users are not clustered near any single data center, but form clusters around multiple data centers or are widely distributed geographically
- Regulations limit options for storing data in specific data centers
- Circumstances require that the public cloud be used in concert with on-premises resources
- Application must be resilient to the loss of a single data center



Why Not Multi-Availability Zones?

- Sometimes AWS fails in all Availability Zones
- Vendor lock
- Complexity of Multi-AZ setup is similar to Multi-Cloud, just shifted
  - Single cloud setup becomes easier (just use 1 AZ)
  - Cross-cloud setup becomes more complicated
  - With the same overall complexity we can get better results
- Better protection – no single point failure

Multi-Cloud – Business Drivers



Multi-Cloud – Operations

- Not a Disaster Recovery – just always running production traffic from multiple independent clouds
- No single point of failure
- Almost instant recovery in case of Cloud outage - just all traffic is served by surviving Cloud
- No “failover/switching back” – when Cloud is restored after outage, we’ll just start sending traffic there
- High complexity/cost, but much better reliability
- Continuously live-tested (monitoring, deployment, real customers)

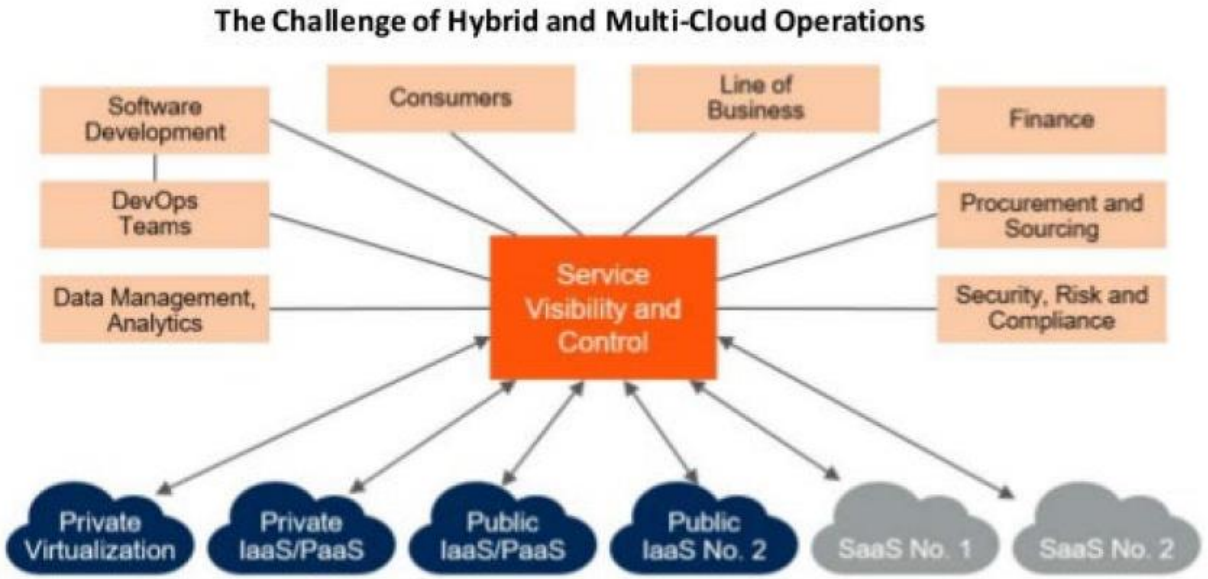
Multi Cloud – Challenges

Transitioning to a Multi-Cloud Environment – Challenges

1. **Provisioning**
   - The infrastructure layer transitions from running dedicated servers at limited scale to a dynamic environment where organizations can easily adjust to increased demand by spinning up thousands of servers and scaling them down when not in use. As architectures and services become more distributed, the sheer volume of compute nodes increases significantly.
2. **Security**
   - The security layer transitions from a fundamentally "high-trust" world enforced by a strong perimeter and firewall to a "zero-trust" environment with no clear or static perimeter. As a result, the foundational assumption for security shifts from being IP-based to using identity-based access to resources. This shift is highly disruptive to traditional security models.
3. **Networking**
   - The networking layer transitions from being heavily dependent on the physical location and IP address of services and applications to using a dynamic registry of services for discovery, segmentation, and composition.
4. **Applications**
   - The runtime layer shifts from deploying artifacts to a static application server to deploying applications with a scheduler atop a pool of infrastructure which is provisioned on-demand. In addition, new applications have become collections of services that are dynamically provisioned, and packaged in multiple ways: from virtual machines to containers.
5. **Data**
   - Multi-cloud architectures offer some potential benefits for databases and data-centric solutions but involve greater complexity, cost and effort than single-cloud architectures.
6. **Governance**
   - The ability to provide strategic direction, track performance, allocate resources, and make adjustments to ensure that organizational objectives are met, without breaching the parameters of risk tolerance or compliance obligations.

Multi Cloud – Challenges

The Challenge of Multi-Cloud Operations



Objective:

- To simplify access to the multiple environments and to maintain visibility across them without impacting the agility of lines of business and developers.
- Organizations need to apply consistent policies, minimize cost and monitor activities across all the services.

Cloud-Enabled Applications

Cloud-Enabled Application Types

1. Web-based application: periodic traffic patterns, requires reliable, scalable, secure, and high performance infrastructure
2. Content delivery network: requires low latency, high availability, durability, access control, and millions of views
3. Batch system: highly variable usage patterns that have usage peaks followed by significant periods of underutilization
4. Fault tolerant system: services and infrastructure to build reliable, fault-tolerant, and highly available systems in the cloud
5. Big data system: involve huge data sets collected from scientific equipment, measurement devices, or other compute jobs
6. Advertising system: internet advertising services need to serve targeted advertising in a small period of time
7. Disaster recovery application: duplicating infrastructure to ensure the availability of spare capacity in disaster
8. File sharing system: stateless client-server architecture in which web services are viewed as resources and have URLs
9. Media sharing and social network: users wants to share their photos and videos on social networking sites
10. Online game: peak usage periods, multiple players, and high volumes of write operations
11. Log file analysis application: log data can be an important source of knowledge for companies having e-commerce systems
12. Financial system: on demand provisioning combined with low latency access to on-premise data sources
13. E-Commerce system: large product catalogs, global customer base, checkout items securely and product recommendations
14. Time-series processing system: data arrive as a succession of regular measurements
15. HPC application: require high bandwidth, low latency networking, and very high compute capabilities.

Multi-Cloud - Key Considerations

- Availability. applications must be architected to guarantee maximum availability.
- Management. runtime information that enable administrators to monitor the system and support on-the-fly changes.
- Scalability. Cloud environments enable applications to scale out to meet bursts in demand, and scale in when demand decreases.
- Resiliency. Cloud environments provide the ability for a system to gracefully handle and recover from failures.

Migration Patterns – Re-host

MP1: Re-host

Definition: An application (component) is re-hosted as-is on cloud platform(s)

Problem: Resource constraints limit scalability, Need to improve the SLO, Single point of failure, Reduce cost of ownership, Modernization strategies

Solution: Re-host on cloud environments, make use of elastic resources, multiple cloud deployment for failover and scalability.

Benefits: Improved Backup and Failover, Coarse-grained scalability at application level, Simple coarse-grained re-deployment.

Risks: Existing architecture constrains portability, deployment time and cost, scalability, integration may introduce greater complexity.

Migration Patterns – Cloudfication

MP2: Cloudification

Definition: An application (component) is hosted on-premise as-is but use public cloud services for extending capabilities instead of on-premise components.

Problem: Need to improve reusability, extensibility, Avoid reinventing the wheel by consuming existing publicly accessible cloud services

Solution: Extend the on-premise application by integrating with existing public cloud services.

Benefits: Improved time to market.

Risks: Integration may introduce greater complexity.

Migration Patterns – Relocation & Optimization

MP3: Relocation and Optimization

Definition: A component re-hosted (or relocated) on a cloud platform is optimized but without evolution in the application architecture.

Problem: The performance of an application needs to be enhanced without the significant effort of architecture change, and without incurring capital expenditure for on-premise hardware.

Solution: Leverage cloud platform services (e.g., IaaS, PaaS services) to improve throughput by leveraging Queue, Database partitioning & sharding, NoSQL, Cache

Benefits: As component re-hosting in cloud and optimized performance.

Risks: The type of application requests changes over time for example proportion of read only calls reduces, Cloud provider does not provide the necessary services to wrap the optimizations around the application without re-architecting.

Migration Patterns – Multi-Cloud Relocation

MP4: Multi-Cloud Relocation

Definition: A component re-hosted (or relocated) on a cloud platform is enhanced by using the environmental services of the other cloud platforms.

Problem: The availability of an application needs to be enhanced without the significant effort of architecture change, and without incurring capital expenditure for on-premise hardware.

Solution: Leverage cloud platform environment services to improve availability, e.g., live migration from existing platform to the target platform in case of service outage

Benefits: As component re-hosting in multiple cloud platforms and improve availability and avoid vendor lock-in.

Risks: Cloud providers does not provide the necessary services to enable application to run in multiple cloud platforms without re-architecting or rewriting the code.

Migration Patterns – Refactor

MP5: Multi-Cloud Refactor

Definition: An on-premise application is re-architected for deployment on cloud platform to provide better QoS.

Problem: Coarse-grained applications are not agile enough to respond to requirement changes or variations in workload, and cannot take full advantage of the SLO improvements that can be offered by cloud platforms.

Solution: The application is re-architected as a set of fine-grained components, The deployment of high-usage components can be optimized independently of low-usage ones, Parallel design for better throughput to multiple cloud platforms, Components designed as independent integrity units to reduce dependencies and enable replacement.

Benefits: Optimal scalability and performance, wider range of multi-cloud deployment options, agility to respond to business and IT change.

Risks: on-premise application is modernized in isolation, and not as part of a portfolio that ensures consistent architecture, Modernization is performed primarily for technical reasons resulting in continued sub-optimal response to business change, Component architecture is only determined bottom-up from existing APIs, Transaction and data integrity approaches may need to be re-evaluated because of multi-cloud environment.

Migration Patterns – Refactor with On-premise

MP7: Refactor with on-premise Adaptation

Definition: A re-architected application is deployed partially on a cloud environment and partially to its current on-premise platform.

Problem: All components of the re-architected application may not be suitable for deployment on cloud. For example, due to sensitivity of data, lack of cloud capability to support current feature of application, license restrictions, or to support a gradual migration plan. However, as they are not co-located, some mechanisms are required to integrate the components.

Solution: A component adapter (e.g., on-premise façade) is adopted to provide integration of the on-premise components with re-hosted cloud-based components.

Benefits: Sensitive data remains isolated and in-house

Risks: Integration is dependent on specific cloud platform and may cause vendor lock-in.

Migration Patterns – Refactor with Cloud Adoption

MP8: Refactor with Cloud Adaptation

Definition: An interface is implemented to provide loose-coupled access to components re-hosted on cloud platform.

Problem: Components re-hosted as-is lacks appropriate service interfaces for integration

Solution: Build a façade, hosted in the cloud platform

Benefits: Loose coupling, platform independent interoperability

Risks : Lack of suitable API on legacy application, Existing application process may not compliant with message-based interaction as a common style in the cloud. Façade is not provided as part of a well-formed service architecture.

Migration Patterns – Multi-Cloud Rebinding

MP10: Multi-Cloud Rebinding

Definition: A re-architected application is deployed partially on multiple cloud environments and enables the application to continue to function using secondary deployment when there is a failure with the primary platform.

Problem: A natural disaster, such as a hurricane, or a failure such as a software bug or configuration error that impacts cloud services may cause a failure to a cloud platform and stop that from functioning.

Solution: The same architecture for resilient systems that route users to the closest data center can be used to account for failover scenarios. In particular, they can be configured to monitor the health of the services to which they are directing users, and if any service is unavailable, traffic will be routed to a healthy instance. An on-premise component adapter (e.g., service bus or elastic load balancer) is adopted to provide integration of the components in different cloud platforms.

Benefits: As unhealthy services become healthy again, traffic can be delivered, returning system responsiveness to maximum levels.

Risks: This scheme does not guarantee instant or seamless failover. There will be downtime.

Migration Patterns – Replacement

MP12: Replacement

Definition: Individual capabilities in a re-architected solution are re-provisioned rather than re-engineered.

Problem: Some of existing components provided by the current application are not the best alternative to meet business requirements.

Solution: Analyze the requirements and identify a set of capabilities that can be replaced by existing cloud services. The provisioning of each capability is assessed by considering current systems analysis on the existing application to identify which of these capabilities could be supported by the re-architected system, alternative cloud services that provide a benefit over the re-engineering of the current capability are identified and replace existing components.

Benefits: The solution is improved though best-in-class cloud services, Re-engineering costs and effort are saved.

Risks: Cloud services presume specific communication protocol that make the replacement a challenging tasks.

Migration Patterns – Application Modernization

MP15: Multi-application Modernization

Definition: on-premise applications are re-architected as a portfolio and deployed on cloud environment.

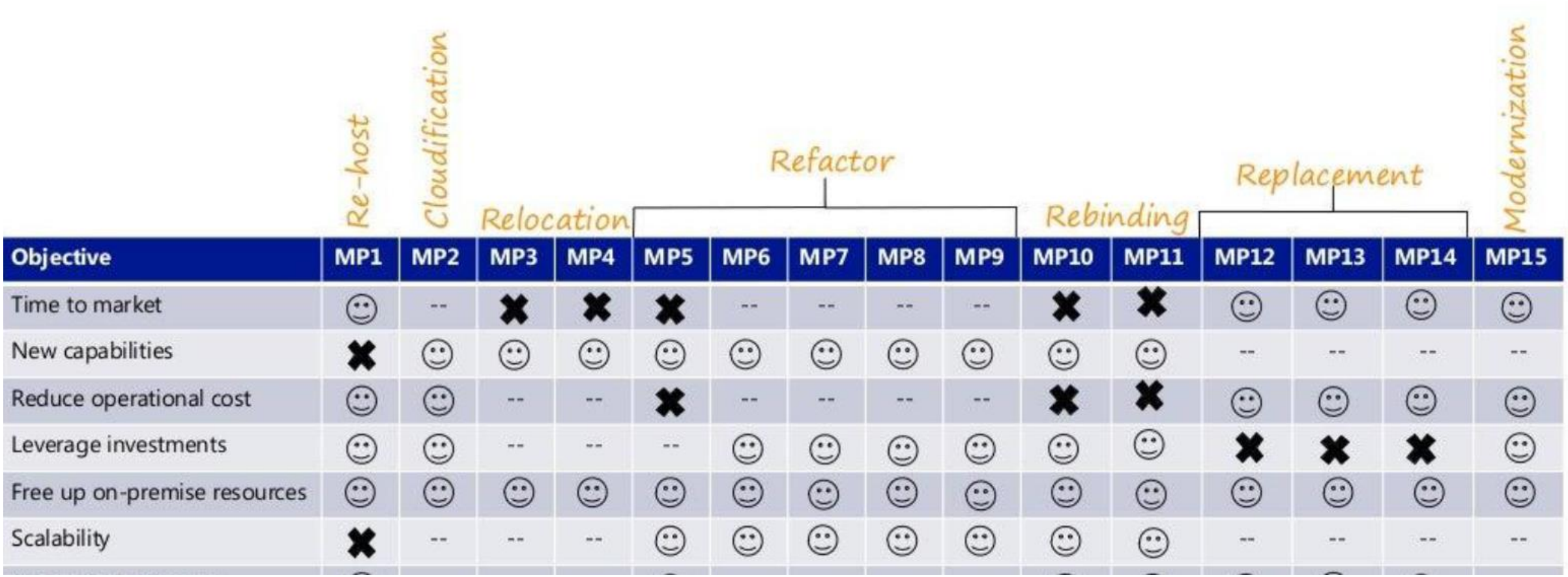
Problem: The re-architecting of on-premise applications in isolation does not remove inconsistencies in data or duplicated functionalities, nor reduce the cost of their combined operation or maintenance.

Solution: Current applications are analyzed as a portfolio to identify opportunities for consolidation and sharing. The separation of the service architecture and the solution architecture enables the identification of components (capabilities) that are shared by more than one solution.

Benefits: Consistent information and rules in shared components, Reduced operation and maintenance costs for shared components, Foundation for more agile delivery of subsequent new applications.

Risks: Lack of business commitment to shared capabilities.

Migration Patterns – Selection Criteria



	Re-host	Cloudification	Relocation	Refactor					Rebinding	Replacement				Modernization
Objective	**MP1**	**MP2**	**MP3**	**MP4**	**MP5**	**MP6**	**MP7**	**MP8**	**MP9**	**MP10**	**MP11**	**MP12**	**MP13**	**MP14**	**MP15**
Time to market	☺	--	✖	✖	✖	--	--	--	--	✖	✖	☺	☺	☺	☺
New capabilities	✖	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	--	--	--	--
Reduce operational cost	☺	☺	--	--	✖	--	--	--	--	✖	✖	☺	☺	☺	☺
Leverage investments	☺	☺	--	--	--	☺	☺	☺	☺	☺	☺	✖	✖	✖	☺
Free up on-premise resources	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺	☺
Scalability	✖	--	--	--	☺	☺	☺	☺	☺	☺	☺	--	--	--	--

Use Case - Localization

Use Case – Disaster Recovery

Improved DR reliability

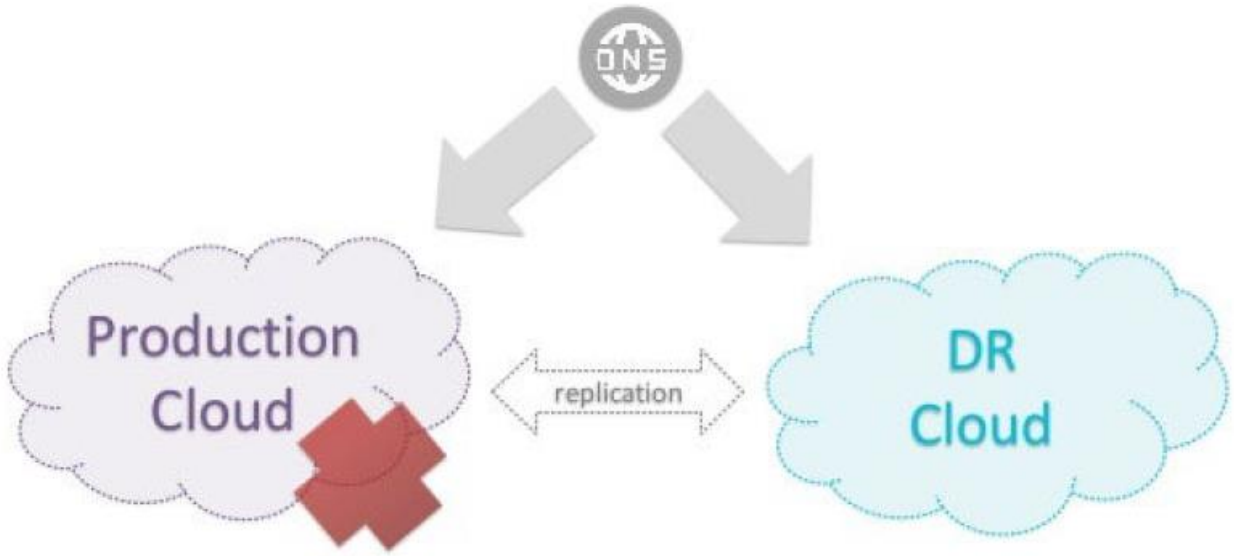
Disaster Recovery not bound to a single provider

Data replication across two clouds

Movement out of a cloud provider for this application becomes easier in the future from both an application- and data perspective

Impact of vendor-specific service issues minimized

Production can be moved to the new environment in case issues are being faced with one cloud vendor



- Vendor Neutral infrastructure services
- Replication latency
- Managing multiple clouds

Use Case – Hybrid Cloud

Cloud burst for peak traffic

Burst private infrastructure to public cloud

Distribute critical and non-critical work loads

- Keep Business critical services on the private cloud
- Move non-critical workloads (e.g. Test and Dev) to public cloud
- Choose the environment as per application requirement
- Optimize Cost

Establish cloud interconnect

Connect the two environments using VPN, Lease Line or via the internet.

Manage from a single console (ideally)

Be able to manage both environments from a single console

The public cloud starts acting as an extension of the private cloud environment.

- Applications must support cloud bursting
- Services must be interoperable
- Single management pane

Use Case – Specific Services(DNS, CDN…)

DNS

You can easily use a third party DNS service outside your cloud provider

Content Delivery Network (CDN)

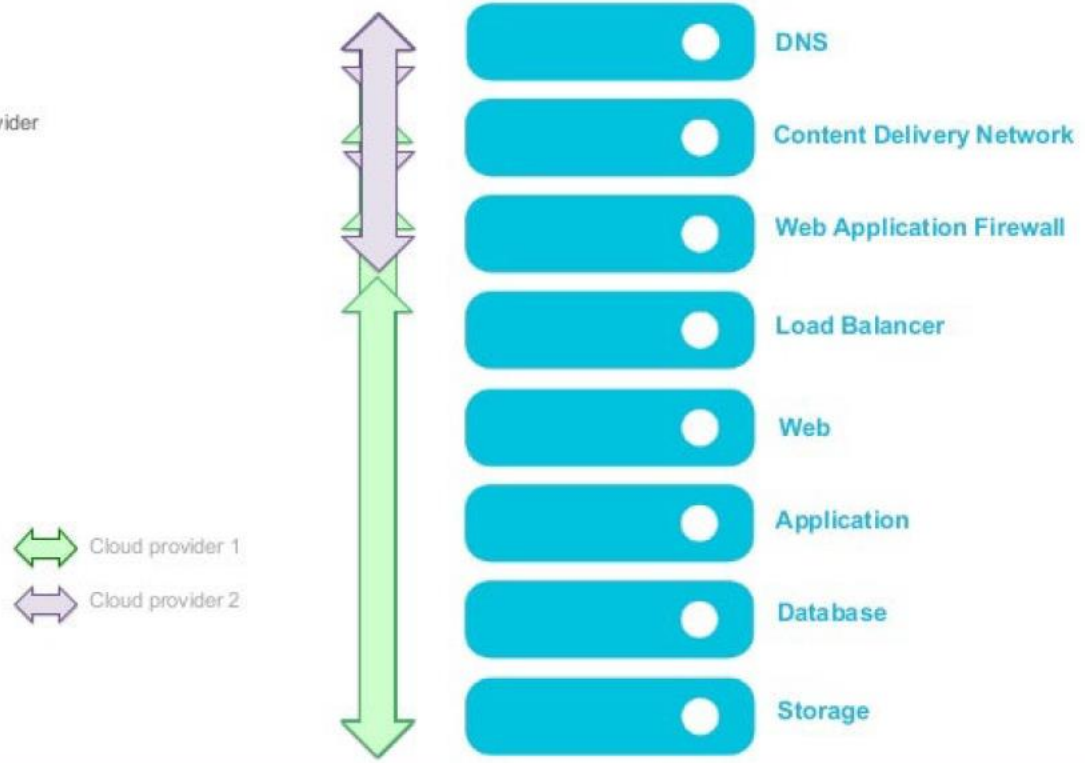
It is not necessary that you use the CDN from within your cloud provider. If you find a CDN provider with better coverage in your target geography you can, in most cases, use that with your existing application.

Web Application Firewall (WAF)

If your cloud provider does not have WAF, or if you find better services outside, in most cases they can be clubbed with an external application

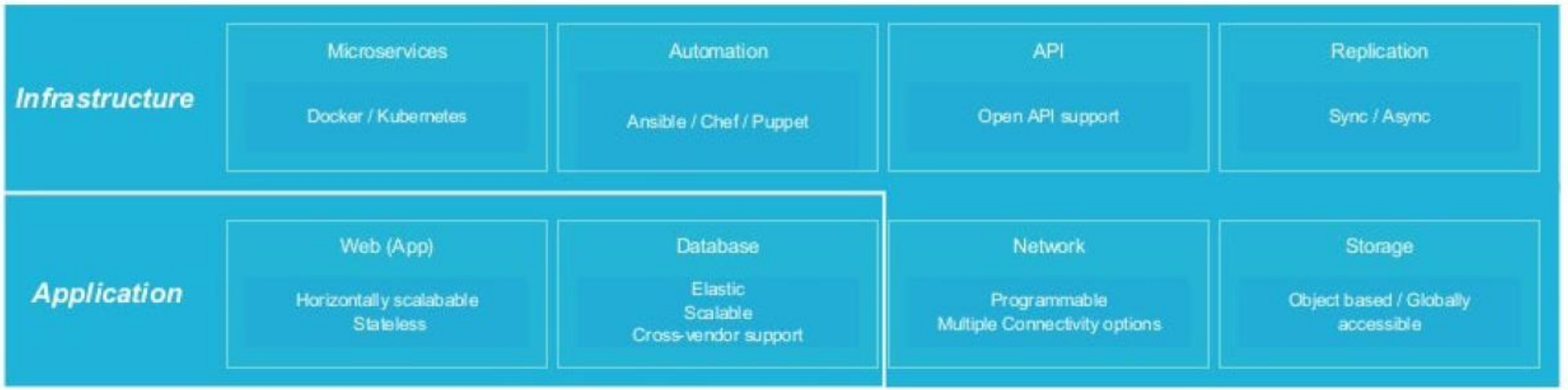
Getting adventurous

Usually replacing edge services with external ones is not a problem. If you want to get really adventurous, throw a third provider in the mix. Be weary of the challenges involved though



- Inter-provider latency and support
- Manual changes across two environments
- Too much segmentation

Use Case - Distributed Workloads



Multiple Architectures

- Distribute one or more service layers with data / storage replication if needed
- Create disparate clusters across different clouds with scheduled or continuous data synchronization
- Have a single entry point to the application and build a “virtual” private cloud across multiple cloud providers

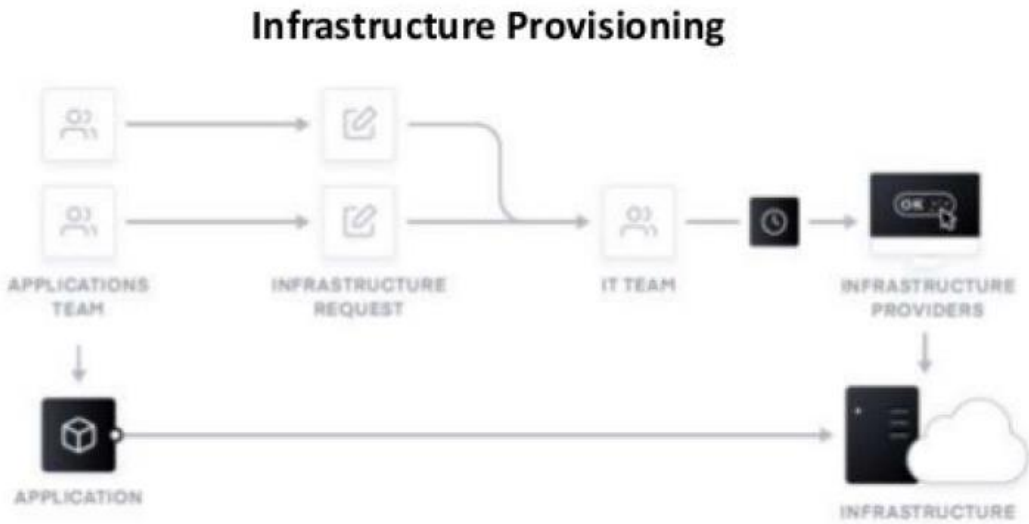
Multi Cloud – Challenges

Transitioning to a Multi-Cloud Environment – Challenges

1. **Provisioning**
   - The infrastructure layer transitions from running dedicated servers at limited scale to a dynamic environment where organizations can easily adjust to increased demand by spinning up thousands of servers and scaling them down when not in use. As architectures and services become more distributed, the sheer volume of compute nodes increases significantly.
2. **Security**
   - The security layer transitions from a fundamentally "high-trust" world enforced by a strong perimeter and firewall to a "zero-trust" environment with no clear or static perimeter. As a result, the foundational assumption for security shifts from being IP-based to using identity-based access to resources. This shift is highly disruptive to traditional security models.
3. **Networking**
   - The networking layer transitions from being heavily dependent on the physical location and IP address of services and applications to using a dynamic registry of services for discovery, segmentation, and composition.
4. **Applications**
   - The runtime layer shifts from deploying artifacts to a static application server to deploying applications with a scheduler atop a pool of infrastructure which is provisioned on-demand. In addition, new applications have become collections of services that are dynamically provisioned, and packaged in multiple ways: from virtual machines to containers.
5. **Data**
   - Multi-cloud architectures offer some potential benefits for databases and data-centric solutions but involve greater complexity, cost and effort than single-cloud architectures.
6. **Governance**
   - The ability to provide strategic direction, track performance, allocate resources, and make adjustments to ensure that organizational objectives are met, without breaching the parameters of risk tolerance or compliance obligations.

Multi Cloud – Infrastructure Provisioning

Multi-Cloud Infrastructure Provisioning



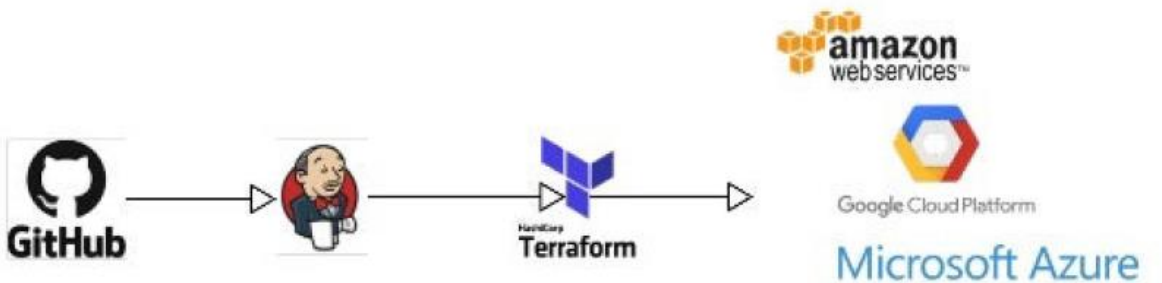
IT teams should start by implementing reproducible infrastructure as code practices, and then layering compliance and governance workflows to ensure appropriate controls.

- **Reproducible infrastructure as code**
  - Infrastructure Provisioning is to enable the delivery of reproducible **infrastructure as code**, providing DevOps teams a way to plan and provision resources inside CI/CD workflows using familiar tools throughout.
- **Compliance and management.**
  - There is need to enforce policies on the type of infrastructure created, how it is used, and which teams get to use it.

Multi Cloud – Infrastructure Provisioning

Building Applications – Infrastructure as Code

	Cloud-Native	Cloud-Agnostic
Source Code Repositories	AWS CodeCommit Azure DevOps Repos Google Cloud Source Repositories	GitHub GitLab Bitbucket
CI/CD Pipelines	AWS CodePipeline Azure DevOps Pipelines Google Cloud Build	Jenkins GitHub Actions CircleCI
IaC Templates	AWS CloudFormation Azure Resource Manager (ARM) Templates Google Cloud Deployment Manager	Terraform Pulumi serverless framework



Multi Cloud – Security

Multi-Cloud Security

Dynamic cloud infrastructure means a shift from zero-trust networks across multiple clouds without a clear network perimeter.

The modern "zero trust" approach requires that applications be explicitly authenticated and authorized to fetch secrets and perform sensitive operations, and be tightly audited.

Secrets management

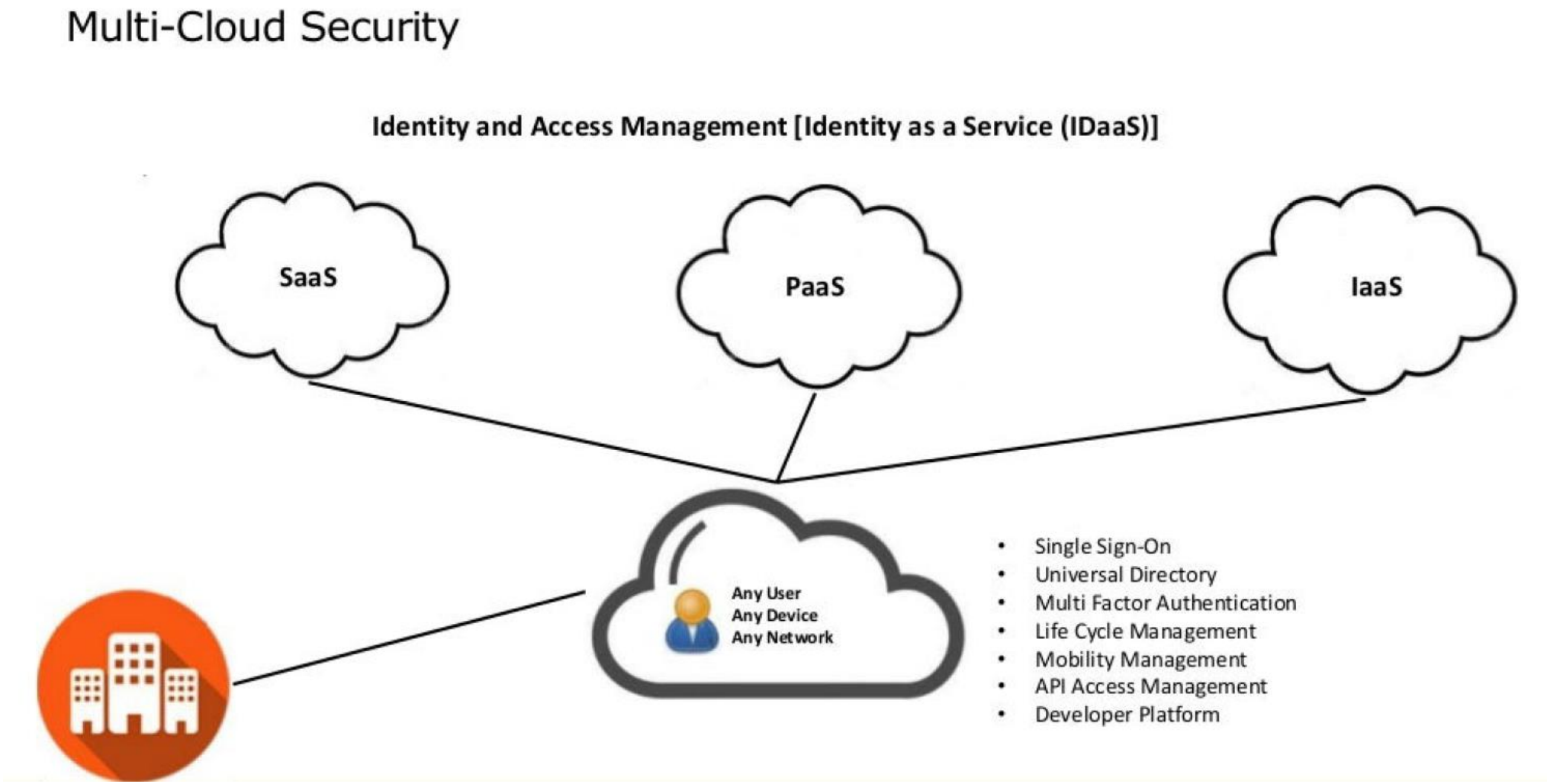
Secrets management is the central storage, access control, and distribution of dynamic secrets. Instead of depending on static IP addresses, integrating with identity-based access systems such as AWS IAM and Azure AD to authenticate and access services and resources is crucial.

Encryption as a service

Additionally, enterprises need to encrypt application data at rest and in transit. This requires Encryption-as-a-service to provide a consistent API for key management and cryptography.

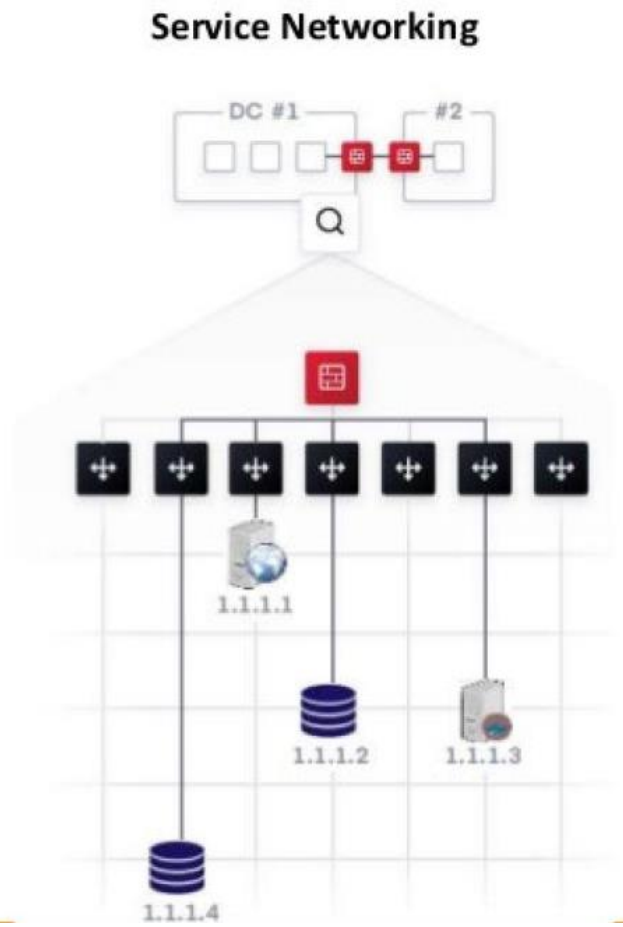
This requires developers to perform a single integration and then protect data across multiple environments.

Multi Cloud – Security



Multi Cloud – Networking

Multi-Cloud Service Networking



Networking services should be able to provide a service registry and service discovery capabilities.

Having a common registry provides a "map" of what services are running, where they are, and their current health status. The registry can be queried programmatically to enable service discovery or drive network automation of API gateways, load balancers, firewalls, and other critical middleware components.

Service Registry & Discovery

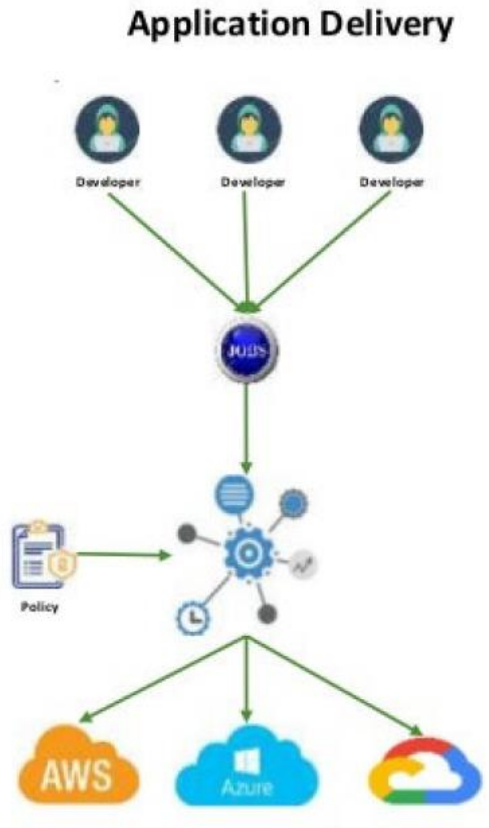
For networking in the cloud it is beneficial to have a common service registry. This would integrate health checks and provide DNS and API interfaces to enable any service to discover and be discovered by other services.

Service Mesh

The two main goals of a service mesh are to allow insight into previously invisible service communications layers and to gain full control of all microservices communication logic, like dynamic service discovery, load balancing, timeouts, fallbacks, retries, circuit breaking, distributed tracing, and security policy enforcement between services. The insights are provided by traffic audit and tracing features.

Multi Cloud – Application Delivery

Multi-Cloud Application Delivery



New apps are increasingly distributed while legacy apps also need to be managed more flexibly. A flexible orchestrator is required to deploy and manage legacy and modern applications, for all types of workloads: from long running services, to short lived batch, to system agents.

Mixed Workload Organization

Many new workloads are developed with container packaging with the intent to deploy to Kubernetes or other container management platforms.

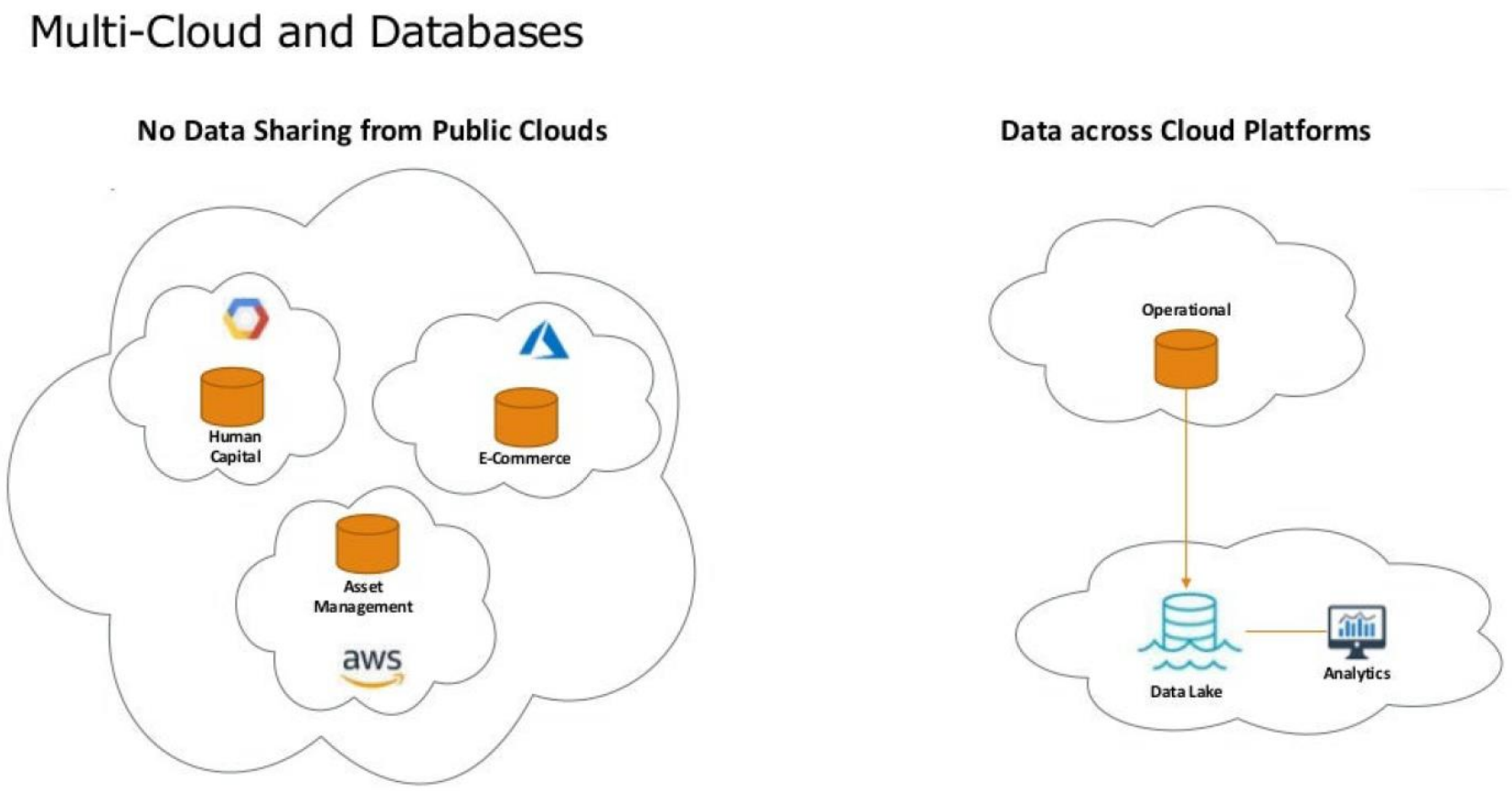
High Performance Compute

Schedule applications with low latency across very large clusters. This is critical for customers with large batch jobs, as is common with High Performance Computing (HPC) workloads.

Multi-Data Workload Orchestration

As teams roll out global applications in multiple data centers, or across cloud boundaries, provide for orchestration and scheduling for these applications, supported by the infrastructure, security, and networking resources and policies to ensure the applications are successfully deployed.

Multi Cloud – Databases

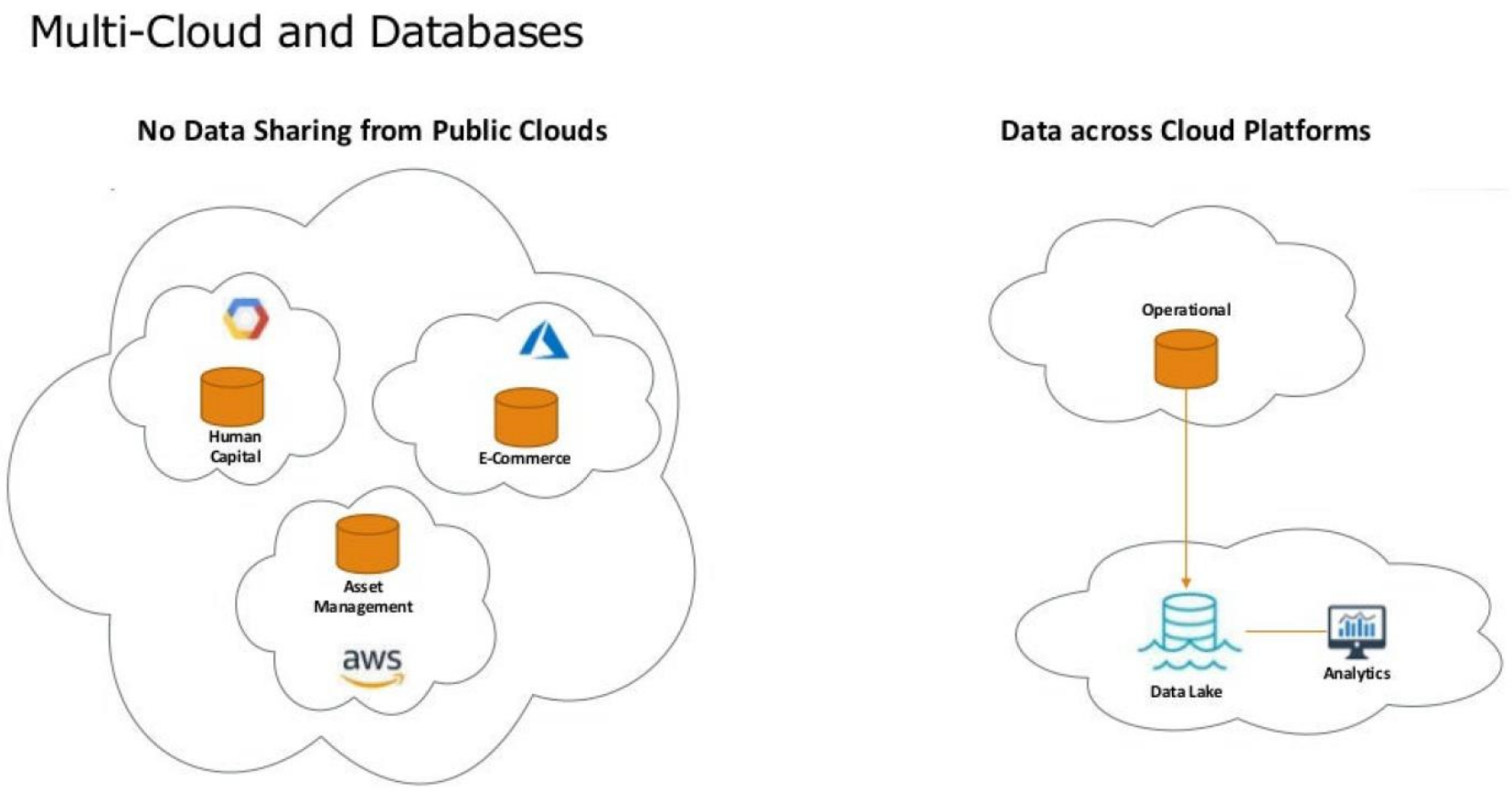


Multi Cloud – Databases

Databases

Cloud-Native (PaaS)	Cloud-native (PaaS) Cloud-Agnostic (protocol)	Cloud-Agnostic (IaaS)
Amazon DynamoDB GCP Firestore		
	Amazon RDS for **PostgreSQL** Amazon Aurora for **PostgreSQL** Azure Database for **PostgreSQL** GCP Cloud SQL for **PostgreSQL**	**PostgreSQL**
	Amazon ElastiCache for **Redis** Azure Cache for **Redis** GCP Memorystore for **Redis** **Redis** Enterprise Cloud	**Redis**
	Amazon DocumentDB (with **MongoDB** compatibility) Azure Cosmos DB's API for **MongoDB** **MongoDB** Atlas	**MongoDB**
	Amazon Keyspaces (for Apache **Cassandra**) Azure Cosmos DB **Cassandra** API	Apache **Cassandra**
	Amazon Neptune Azure Cosmos DB **Gremlin** API	Apache TinkerPop **Gremlin**

Multi Cloud – Databases



Multi Cloud – Architecture Domains

Summary – Architecture Domains

1. Applications
2. Databases
3. Networking
4. Security
5. Infrastructure Provisioning