Fit for purpose: Hybrid Cloud Platform Technologies

By IBM Global Industries, Financial Services

March 2023
Executive Summary

For our Financial Services clients, there are other imperatives that we need to incorporate:

- **Financial Metrics:** All firms focus on expense compression and driving profitable revenue growth.

- **Regulation and Compliance:** Every financial services firm needs to understand, implement, test, audit, and comply with regulations. Technology helps automate regulatory challenges.

- **Cyber-Security:** While hybrid technologies incorporate a consistent security model, everything about the platforms in financial services needs to be built around a Zero Trust Security-First architecture.

- **Digital Transformation:** Financial Services firms are reinventing themselves to offer digital services across every line of business. Without new thinking, the transformation journeys are unpredictable at best.

- **Automation:** Firms are full of manual business and technology processes run and monitored by human operations teams. Manual processes create fragmented customer and employee experiences.

- **Risk:** The risk landscape has become ever more complex, both in a regulatory context but also shifting the view of risk from being a necessity to a business differentiator.

- **Disruption:** Just as emergent ecosystems have disrupted entire industries, the same will proceed in financial services. Clients have the choice to create or participate in ecosystems that extend the value of an institution’s services directly into commonly used business platforms, such as ERP and Payments systems.

- **Embracing Data to create AI-infused insight:** Accessing dark pools of siloed data, reinventing data governance, and leveraging self-service data science to build trusted and transparent insights at scale is the goal of every firm.

---

This paper aims to help every level of executive and practitioner understand the definition, scope, and value of hybrid technologies. Digital Transformation and Hybrid Cloud are now ingrained parts of our 2020s vocabulary. In incumbent businesses seeking to modernize in response to disruptive forces like the pandemic or new competitors, digital transformation and hybrid cloud is part of every technology approach. We throw these words around in everyday conversation as if their meaning is self-explanatory. IBM’s competitors have sought to define simple terminology to meet their business objectives, for example, the ability to run workloads in multiple physical locations. One example is the false equivalence of terms like Multicloud and Hybrid interchangeably. Almost every firm consumes or deploys cloud services from more than one provider. That approach is undoubtedly multicloud but is only partially hybrid. To be fully hybrid, a firm must embrace new mindsets, principles, architecture, ways of working, and tools.

At IBM, our broader definition of Hybrid includes the following:

- **Open** – The hybrid cloud runs on open frameworks that are both generic and industry-specific

- **Hybrid** – Most businesses are already in multiple environments on-prem, in clouds, and at the edge. Software layers secure and operate these environments automatically and consistently.

- **Multi-Cloud** – Embracing multiple public clouds typically means embracing cost and complexity. Public clouds inspire us to think of cloud as a mindset and a new way of working rather than a specific location. The new techniques include Agile development, DevOps, and Site Reliability Engineering (SRE).

- **AI** – Clients need to train and deploy AI at scale with trust and transparency. AI is also necessary to observe and run environments.
We assert that a hybrid approach accelerates value creation by adopting new processes and technology frameworks, each of which is considered. As clients review their incumbent estate and plan for their adoption journey, they must consider the specific economics (ROI and Technical Debt) and known pathways to value. Doing so will help avoid a technology strategy built upon shiny objects. Instead, a program will be built on defined outcomes and predictable experiences. Every business needs to make fit-for-purpose decisions about where technology runs. Again, considering fit-for-purpose in functional, non-functional, and regulatory requirements shapes the outcome. Irrespective of a specific platform, clients need foundational platform-agnostic hybrid cloud accelerators like Red Hat OpenShift. No company adopts a hybrid approach by abandoning incumbent investments such as IBM Z, a full participant in the hybrid journey. IBM Z enables multiple modernization approaches with integration to the IBM Cloud Paks and Zero Trust security to accelerate delivering new digital client experiences with all necessary Zero Trust security and controls.

—

Environmental and Social Governance: a central theme for the coming decade, institutions need to show their appropriate stewardship of the earth’s resources, as well as ensure diversity and inclusion in both their staff, vendors, and product mix.

Written by

John J Duigenan
The Primary Value in Cloud Comes from a “Mindset” and a “New Way of Working”

Given that we have concerns about the typical usage of the word “Cloud,” let’s be very specific about its benefit. Early beliefs suggested that the adoption of elastic cloud infrastructure would reduce costs. However, customers of cloud providers saw their planned expenses go up by as much as 300% due to unforeseen billing and their inability to abandon their existing incumbent technology investments. At face value, the cost increase is prohibitive. However, rather than anchor in the infrastructure cost, the calculation must be redirected to the value created from a new mindset, a new way of working, and new tools. The assertion is that new ways of working can radically accelerate and simplify software development, delivery, and operations. IBM’s experience with clients demonstrates that value is accelerated 2.5 times compared to a public cloud-only approach when using a hybrid approach.

Hybrid Technologies are Not Monolithic

Hybrid Technologies are Not Monolithic: Monolithic technologies are rigid and inflexible. Hybrid Technologies are de-coupled. The foundational construct within hybrid technology is the combination of API and Microservice:

— An API is an application programming interface, in other words, the contract definition for how a distinct function is called. Invoking an API call is the primary way developers call a code function. An example of an API call might be to get an account’s balance or initiate a transfer between accounts. The API definition specifies both the processing location of a service and the definition of invocation and response parameter values. A prior way of thinking about API calls was an RPC or remote procedure call.

— A Microservice is the implementation of code in support of an API definition. The microservice can be implemented in any programming language. It can be deployed to any kind of server or platform environment anywhere, providing it supports API calls. Microservices are intended to encapsulate very distinct, discrete, and de-coupled business functions. Whereas monolithic code combines user interface, business processing, and data functions, each of these must be distinct in well-formed microservices. De-coupling is done to promote and enable the re-use of business and data functions across a code base for multiple delivery channels. Microservices are intended to be stateless; in other words, a specific instance of a microservice does not retain stateful session data; this enables multiple instances of a microservice to run at once, enabling scalability. A microservice may call other microservices via API calls.

Hybrid Technologies are Built on Open Standards

Technology organizations have evolved to embrace code, APIs, and frameworks based upon open standards. There is a strong belief that open standards serve multiple interest parties and foster innovation through open collaboration. The following open technologies are common elements of hybrid technologies:
**API Standards**: Clients can adopt standards-based APIs to avoid the burden of architecting and implementing a custom API framework. Using a standards-based API assists interoperability when interfacing with multiple related components. Examples of open API standards include those offered by the Banking Industry Architecture Network or BIAN (https://www.bian.org), an organization comprised of multiple financial institutions and their technology providers. BIAN aims to provide API and data-model coverage across a universe of banking-specific functions. Another example of open standards in financial services is the “Payment Service Directive” related to the European Commission law opening all banks’ balance and payment functions via API calls. All European banks were required to implement these APIs.

**Containers**: A container packages the executable functions for a microservice. In the past, a typical packaging vehicle was the virtual machine image. Virtual machine images contain ALL required executables and configuration assets, including operating system, client frameworks for security and monitoring, middleware, database, and application runtimes. While virtual machines provide efficiency, running them consumes excessive system resources given the duplication of assets needed for each environment. In contrast, a lightweight container image contains only the code assets and libraries required to run a microservice. A container runs within a host operating system and therefore does not need to package operating system services. Each running container is isolated from others by design. This is partially to ensure that one container cannot attack another. Once built, the configuration of a container can be immutable; in other words, the configuration cannot be changed. This is a valuable characteristic as immutability prevents attacks based upon the ability to manipulate a configuration. Containers are packed with a deployment descriptor file that describes the configuration of a microservice packaged within a container. Container images will standardize to a format defined by the Open Container Initiative. Container technologies and standards are platform-neutral.

**Container Hosting and Orchestration**: A hypervisor runs multiple machine images simultaneously in virtualized environments. A container host is the nearest equivalent of a hypervisor that runs numerous container images. At its most basic function, a container host supports the deployment of container images and their starting and stopping. The container host routes the network traffic related to API calls to each container image. It monitors the utilization and performance of each container and can scale up or down additional container images vertically within a host or horizontally across multiple hosts. Multiple container hosts are known as a cluster. Every public cloud provider provides a proprietary container service, often compatible with the Kubernetes open standard. Additionally, Hybrid Cloud Technology providers like IBM and Red Hat provide the OpenShift Cloud Platform that provides a fully compliant Kubernetes container hosting and orchestration capability that can be deployed and operated consistently in any cloud provider location any data center. Red Hat’s Open Cloud Platform, OpenShift, incorporates Kubernetes container hosting and orchestration.

**A Service Mesh**: provides secure routing between microservices within a container host and between multiple hosts in a cluster. The mesh layer acts as an API traffic router. As API request traffic flows across the mesh, it can be logged or captured for analysis and observability. Istio is an open standard for the service mesh. Red Hat’s Open Cloud Platform, OpenShift, can integrate a service mesh.

**Multicloud API Networking Accelerators**: When containers are dispersed across multiple locations, edges, or clouds, connecting them safely can be challenging. Frameworks like Skupper have emerged to simplify interconnects, providing a virtualized, secure, and highly available API network across many locations. The benefits of a framework like Skupper include no changes to application code or configuration, inherent encryption and security protections, no need for VPNs or exposing ports on the internet. From a networking perspective, Skupper provides dynamic load balancing based upon service capability, cost and locality-based traffic forwarding, and redundant routing to protect network outages.

**A Container Repository**: simply a library of available container images, ready for deployment. The image is compliant with the Open Container Initiative: a digital library of first-party in-house container images and their configuration, or those which are open or proprietary sourced from third-party suppliers. Enterprises typically use private repositories that run under their control and protection. Red Hat’s Open Cloud Platform, OpenShift, incorporates a container repository. Frameworks like Skupper are beneficial in safely
exposing APIs on IBM zSystems to a broader platform ecosystem.

— A Container Scanner: akin to a virus scanner, a container scanner searches and identifies vulnerabilities in a container image and produces a report that enables a user to determine whether a container image is safe to deploy. Multiple vendors offer container scanning capabilities, broadly adopted in regulated industries. Some firms have a policy that a container image may not be deployed to a container repository or a container host until container scanning indicates that a container is safe.

— A Source Code Library: most developers are very familiar with source code control systems that provide a library where code and configuration artifacts can be stored, versioned, and retrieved. Any project with multiple developers working on the same code artifacts needs a library and control mechanisms. Just like in a book library, artifacts such as files are checked out and checked back in by registered members under the supervision of a librarian. Unlike a book library, code artifacts are typically changed between check-out and check-in. A person performing the librarian’s role can approve or decline changes made upon check-in, preventing bad changes from entering a codebase. The library enables extracts of any codebase version for review, compilation, building, and testing. The predominant open standard for the source code library is Git, initially created by Linux creator Linus Torvalds. While git is a genuinely open standard, multiple companies package it commercially.

— DevOps Pipeline: combines and integrates development and operational processes. Before DevOps, organizations separated development, build, test, release, and operations functions. Each SDLC step was distinct, discreet, and unintegrated from the previous or next steps. The predominant analogy was “throwing code over a wall.” This fire and forget software delivery method was fraught with skills transfer and knowledge issues and worsened the absence of automated hand-offs. DevOps address each of these issues head-on. DevOps works on the grounding principle that every step and hand-off of a development and release process is integrated and automated. The integration is provided by describing every build or configuration step in a machine-executable model. When infrastructure steps are built into a DevOps pipeline, this is known as “Infrastructure as Code,” as the executable model provisions the required infrastructure. Having a machine-executable model can be validated for compliance with security policies. Hence DevOps can be referred to DevSecOps or DevSecComplianceOps. The term pipeline is that the build, package, and release steps use multiple underlying tools such as compilers, linkers, packagers, and checkers. Each tool in the chain (toolchain) has a distinct function but operates in a coordinated and integrated way with its partners. There is a close relationship between the Source Code Library and a DevOps Pipeline. Committed changes to source code or configuration assets automatically invoke a DevOps pipeline to build and deliver code through an automated release process. This level of integration automation is referred to as “CI/CD” or Continuous Integration and Continuous Delivery. If changing only the code for a single microservice, the only deployment to be made following a release approval step will be the re-deployment of the container for the changed microservice. While a DevOps pipeline is often fully automated, regulated firms will use approval steps in a pipeline to ensure that essential scrutiny is given to code changes and release processes. Jenkins and Tekton are popular DevOps pipeline tools. Red Hat’s Open Cloud Platform, OpenShift, incorporates a DevOps pipeline.

— Site Reliability Engineering: Site reliability engineering (SRE) is a software engineering approach to IT operations. SRE teams use software frameworks like Ansible as a tool to manage systems, solve problems, and automate operational tasks. SRE takes the tasks that operations teams have historically done, often manually. Instead, it gives them to engineers or ops teams who use software and automation to solve problems and manage production systems. SRE is a valuable practice when creating scalable and highly reliable software systems. It helps you manage large systems through code, which is more scalable and sustainable for sysadmins managing thousands or thousands of machines. SRE teams are responsible for how code is deployed, configured, and monitored, as well as the availability, latency, change management, emergency response, and capacity management of services in production. Site reliability engineering helps teams to determine what new features can be launched and when by using service-level agreements (SLAs) to define the required reliability of the system through service-level indicators (SLI) and service-level objectives (SLO). With SRE, 100% reliability is not expected; failure is planned for and accepted. Recovery from failures and outages is automated.
— **Hybrid Databases and Stores:** database technologies continue to evolve. Relational databases are powerful but not best suited to every use case. For example, databases that support non-structured data are prevalent and are based upon NoSQL (Not Only SQL) principles. Popular NoSQL vendors include MongoDB and Cassandra. Graph databases represent entity-based relationships for specific analytical purposes, and Time-Series databases are popular for capturing and analyzing time-based data streams. Regardless of the database style, databases must participate in a hybrid ecosystem regarding their operations and how they can be called from APIs and microservices. Operationally, many databases are now offered as container images that can be deployed to a container host. The IBM Cloud Pak for Data offers multiple databases and connectors.

— **Agile Development Methods:** Conventional development was done using a familiar waterfall technique. The waterfall is mostly a burdensome serial process and can be rigid. While there are clear advantages to good discipline for software development, the limitations can outweigh the benefits. Agile fosters rapid design processes and delivers minimum viable capabilities. Agile techniques include recommendations for project size, work-effort structure, team size and structure, team communications, and project management.

### Existing Investments Must Participate

Few organizations can start a new development program that disregards incumbent technology choices or ongoing financial commitments. Generally, only start-ups have the short-lived luxury of a clean sheet. A mature organization has multi-year investments in technology platforms and physical or human assets such as data centers and operations teams. Data centers often represent billions of dollars of investment amortized over a facility’s twenty to thirty-year life. Hardware and software platforms are typically amortized between three and five years. While these factors seem obvious, many development programs aspire to create value or savings. Counter-intuitively, ROI calculations do not include incumbent costs. New developments increase the expense run-rate and only deliver run-rate savings when existing investments are eventually decommissioned. Financially responsible firms will seek to leverage existing investments within a newly established hybrid ecosystem.
Moving forward with addressing existing technical debt is challenging. Clients typically consider the following approaches when planning for application modernization. Each needs to be viewed through the lenses of financial and risk measures, predominantly geared towards assessing ROI and expense. Remember that transition is not just about making an application current today but how to best position it to remain current. This is why we recommend that most assets be transitioned using Iterative Modernization.

**Lift and Shift**
_Easy to Start. However, value creation is sub-optimized due to a primary focus on infrastructure._

Using modernized infrastructure without modernizing an application is the essence of lift and shift. Depending upon the technology platform, it may be possible to lift and shift an application into a virtualized environment or a small number of container images. Code typically runs ‘as-is’ on the new infrastructure.

Lifting and shifting potentially eliminate the platform benefits of the original hosting environment, such as availability, integrity, reliability, throughput, and security. Doing so can result in reduced service levels and increased outages. A chipset change between existing and new platforms can also result in incompatibility issues and the need to recompile aged code.

Lifting and shifting do not introduce new business capabilities, so it is simply a play to reduce the run rate cost of existing platforms and operations teams. In many cases, it is not successful in doing so.

**Rip and Replace with New**
_Requires Upfront Capital Expenditure. Customization creates risk._

Clients seduced by the agility of their emerging disruptive competitors often choose to build/buy a completely new system. However, this is usually done by ignoring or miscalculating an existing asset’s value. Determining what is ‘sunk cost’ and an
'underutilized asset' is challenging to evaluate. Appropriate time should be invested before making this choice to ensure the optimal ROI.

While it may be simple to buy a new platform, customization often sinks such projects. Most financial services firms demand extensive customization to cookie-cutter platforms. Customization is often seen as the final 5%, but anyone who has experienced a construction project knows that the last five percent of work consumes disproportionate cost and time. Customization is instrumental in introducing delivery or execution risk.

Iterative Modernization
Balancing investment with Value Creation. Preferred by most clients.

Clients looking to balance risk, ROI, and expenditure choose this approach. Iterative modernization involves introducing impactful gradual changes to existing code and systems. This aligns well with the overall business strategy embracing modernization as an ongoing investment.

In its most basic form, iterative modernization takes three styles that may be used separately or in combination:

- **Modify existing code:** depending upon the knowledge and understanding of existing code, this can be akin to open-heart surgery. It involves manipulating current code in its existing programming language and assumes that skills are readily available. A growing number of automated analysis and migration tools are becoming available to assist with code conversion and refactoring. IBM and others have applied AI-based machine learning to understand existing COBOL helping understand business logic, dependencies, and conversion to Java or an intermediate programming language.

- **On-platform modernization accelerators:** The introduction of new middleware can simplify interfacing applications.

- **The Extension of new code alongside existing code:** New code is created in current programming languages and is interfaced with existing core functions via APIs (RESTful or programmatic), messaging middleware, or database capture/synchronization. Providing the new code leverages effectively uses open source and containers, it can run in any destination chosen by the client.
ROI and Technical Debt represent seriously inconvenient truths, especially when ranked lower than technical criteria. Technologists enjoy pursuing new technologies and approaches without validating their ability to reduce operating expenses, simplify a business, or drive new revenue and profit. Financial considerations need to be at the heart of all technology strategies, including the hybrid approach.

IBM’s experience with clients has demonstrated a lift of 2.5 x in financial benefits from a hybrid platform approach over the economic levers of a public cloud-only deployment. Every business will need to consider ROI through industry and business business-specific lenses.

Adopting a modern computing approach provides demonstrable ROI improvements. Fundamentally running a hybrid or modernized environment ensures that enterprises can minimize the investment necessary to continue to benefit from existing assets while simultaneously competing on a level playing field with new entrants who are leveraging the latest innovations in technology.

Several essential factors in an ROI computation influenced by technical debt present challenges because the value is considered ‘soft’ or hard to assign. Two examples immediately come to mind:

- **Availability of Skills:** The availability of technical skills is a pressing challenge. The inability to attract and hire or procure appropriate skills can inhibit any planned transformation program. Skills may be necessary for two key reasons. First is the ability to understand code as it is currently implemented. On occasion, the only way to understand a codebase is via human investigation and expertise. Second, if a codebase needs to be updated or extended, experienced hands-on skills are likely required. The expense here does not relate to staffing resources. Instead, it relates to both costs of being unable to execute development activities and the dollar impact of missed business opportunities.

- **Cyber-Security Risk:** Aging code is prone to cyber-security issues, one of the existential topics for regulated firms. Using appropriate
technical skills, vulnerabilities in aging code need to be identified and mitigated, assuming that workarounds exist. If a codebase uses a proprietary library identified as 1) end of life and 2) containing a risk, a client may have no choice but to remove a dependency by replacing a library with an equivalent. While the cost of replacement activities is measurable, the potential cost of an incident is less easy to quantify. The case and ROI for making changes are compelling when quantified.

Because hybrid technology enables clients to make their investments fungible, interoperable, and substitutable, they can always make changes as technology evolves. Better ROI is possible both in the short term and over an extended period.

While this topic is critical, a very comprehensive view of technical debt and ROI is beyond the scope of this document.

04
Fit for Purpose Platform Decisions ↩

The assessment of whether a workload is deployed to a fit-for-purpose platform can be a very subjective topic. In this case, subjectivity is most often a factor of understanding and comfort. A very comfortable person in one technology domain may not understand why other technologies could be more appropriate, relevant, or simple.

An objective way to think about Fit-for-Purpose is to consider non-functional requirements (NFR), often known by architects as the ‘ilities.’ For a comprehensive overview of non-functional requirements, see https://en.wikipedia.org/wiki/Non-functional_requirement where 66 types of NFR are listed. See the attached graphic.
The specific significance of an NFR is that it is a requirement, not a platform characteristic. Each platform implements NFRs differently, if at all. Some platforms support NFRs inherently, thus simplifying a potential development burden. If a platform does not implement an NFR, but that NFR must still be supported, a developer needs to find platform-independent mechanisms, such as proprietary, open, or home-grown software techniques. Non-functional requirements are often determined as necessary through regulation. Where regulated, implementing an NFR is mandatory. One specific example is transactional integrity and consistency. Real-time consistency is currently not available in public cloud database services. Their loosely coupled architecture does not support real-time consistency. They offer “eventual consistency,” meaning that transactional updates may take time to replicate across nodes. A client must decide whether eventual consistency is sufficient for financial transactional data.

Linked to NFRs are specific functional requirements or workload characteristics. For example, a transactional workload such as a bank’s transactional ledger is particular. Still, it must also be delivered to support specific non-function requirements such as security, reliability, availability, recoverability, throughput, scale, integrity, consistency, and serviceability. On an enterprise server platform such as IBM zSystems, these non-functional requirements are inherently and fully supported by the Operating System and middleware. A commodity platform may not inherently implement all non-functional requirements, resulting in either a compromise to de-prioritize some NFRS or allocating additional cost to implement workaround approaches.

Data gravity, sovereignty, and transactional gravity are essential considerations in fit-for-purpose architecture and most often become considerations when ignored or misunderstood. Gravity implies that data retrieval and transaction updates are typically optimal when co-located. For example, if most of a firm’s core transactional data is on a server in Texas, processing that data in a cloud data center five hundred miles away will result in multiple side effects. The first side-effect will be latency due to network bandwidth considerations. Latency could render an application useless due to performance or throughput considerations. Secondly, data egress charges from a cloud provider create surprise bills. When data processing and database are co-located.
and readily accessible, data sovereignty, locality, and gravity considerations minimize latency, cost, and the number of copies of data in the enterprise. In other words, the platform(s) that host the largest bodies of transactions will draw business apps that need to extend or integrate closer to them. In the case of existing investments, most new business services need to integrate with existing business services – and, in many cases, within the same transactional scope. This is another critical consideration for workload placement.

The final consideration is whether a technology platform is a partial or complete participant in a hybrid cloud ecosystem. At its minimum, a platform needs to offer:

- Modern and legacy programming languages
- APIs and Microservice integration
- Containers and Container Orchestration
- A Standard visual development environment, source code library, and DevOps toolchain
- Ready access to inherent data services.

IBM’s perspective is that each of these is necessary, and each of our platforms, including IBM zSystems, POWER, and IBM Cloud, offer these capabilities consistently.

A hybrid cloud developer can access and gain value from the capabilities of IBM’s platforms without inherent underlying knowledge of each hardware or operating system platform. IBM can deliver these capabilities because of the availability of Red Hat and IBM’s hybrid cloud accelerator offerings.

For clients that decide that they want to adopt Hybrid Technologies, there may still be uncertainty about the need to move to the cloud. Specifically, some financial services clients have no intention of running workloads or storing data in a public or private service offered by a cloud provider. However, those same clients want the benefit of a hybrid cloud, specifically the acceleration of delivery and the adoption of standards-based software and frameworks on fit-for-purpose technology platforms.

Clients should not need to choose whether their environment is exclusively public, private, or on-premises. They will likely choose to run their workloads in multiple locations achieving all combined platform benefits. The point is not to choose only one way. Instead, the point is to adopt a technology platform that can be continuously evolved and optimized with business outcomes as the primary driver.
The plethora of technologies, frameworks, and standards in a hybrid ecosystem might suggest that it is complicated, challenging to learn, and hard to get started. When developers and operations staff need to be productive, complexity gets in the way and becomes an inhibitor to business acceleration. Another dimension to complexity is introduced when using multiple cloud providers, especially as a client locks into more and more proprietary services. Doing so creates a need to have specific framework skills in each cloud provider’s overarching frameworks and knowledge of their particular services.

Red Hat OpenShift and the Red Hat Open Cloud Platform deliver a consistent developer and operational experience regardless of where technology is imagined, developed, and delivered. OpenShift is often thought of primarily as the market-leading Kubernetes offering. The Red Hat OpenShift experience can be delivered everywhere, with product offerings for customer-managed environments or options to buy as a service from IBM, Red Hat, and cloud providers. The use of OpenShift is not limited to distributed x86 platforms. Its use on IBM POWER and IBM zSystems is proliferating.

However, minimizing OpenShift’s role to Kubernetes is too limiting. OpenShift also delivers a much broader developer experience that includes an ecosystem of integrated and open developer frameworks such as a security model, a DevOps toolchain, a container repository. Due to its complexity, security is often an afterthought. However, OpenShift reduces operational risk by integrating security directly into an automated DevOps using built-in policy templates that enforce a configuration’s adherence to security and compliance policies. Doing so protects application workloads at runtime.

In addition, Red Hat OpenShift is IBM’s hosting platform for the IBM Cloud Paks. Cloud Paks enable developers to accelerate application development. OpenShift allows developers and ops teams to host the Cloud Paks wherever data and processing are.

A side effect of workload portability is insulating and hedging technology risk. In brief, we could think about two pre-eminent risks, assuring resilience and facilitating workload portability. A consistent hybrid operating environment running across multiple locations enables business continuity when one site fails. Recently, this has been top of mind for enterprises who placed workloads with cloud providers that took significant outages. Organizations that could not automatically pivot to other workload processing locations went dark, unable to perform standard business functions or transactions. Additionally, legislation is emerging in Europe to ensure that firms do not deeply lock into a cloud provider. In an event where the situation, e.g., a regulatory mandate or a client decision, determines a need to move workload, the hybrid technology platform enables a client to remove workload and re-host it from one cloud provider to another location, either on-prem, a hosting center, or an alternate cloud provider.
There is no question that IBM’s clients for the mainframe need it to be a dynamic, vibrant, and innovative hybrid technology platform. Despite constant propaganda from IBM’s competitors, clients need IBM zSystems’ extreme reliability, security, throughput, integrity, and availability for their mission-critical core product-processing transactional systems.

While IBM zSystems’ hardware and operating systems are vibrant and highly contemporary, many third-party application platforms are not current. Because mainframe systems run predictably for decades, independent software vendors and clients have consistently under-invested in their application platforms, resulting in technical debt.

Two operating systems, z/OS and Linux, are full participants in a hybrid ecosystem for the IBM zSystems environment. As clients build new digital experiences for consumers that access IBM zSystems data and applications, they can build once and run them anywhere. They can tap into the Red Hat OpenShift platform and use containers to modernize with lower risk and lower cost without sacrificing resiliency and security. The availability of containerization on IBM zSystems enables the co-location of cloud-native applications on the same system hosting enterprise data, reducing the cores needed by up to 3.6 times compared to using remotely connected x86.

IBM and our partners provide technologies to help modernize the application platforms on the mainframe to embrace a hybrid future.

There’s an obvious reason for the mainframe to be an intense source of value for clients and equally a target for IBM’s competitors: the most crucial...
product transaction data for large enterprises typically sits in mainframe databases driving data gravity considerations. Competitors primarily target moving data and processing from the mainframe but often do so at a reduced quality of service and increased risk. Projects typically focus on replicating or extracting mainframe data via ETL and data movement. Such an approach creates increased cost from mainframe utilization, increased cost from mainframe data, and results in multiple ‘shadow’ copies of mainframe data away from the transactional core.

In comparison, IBM’s offerings provide the following platform capabilities that seamlessly integrate IBM zSystems as a first-class participant in a hybrid cloud ecosystem:

- The IBM zSystems Digital Integration Hub provides secured virtualized and cached access to mainframe data via SQL, ODBC, and JDBC.

- z/OS Connect exposes mainframe transactions through API calls

- IBM z/OS Container Extensions (zCX) enable a Docker container to host a microservice that co-mingles both Linux and Z/OS functionality

- Red Hat OpenShift provides Kubernetes capability in both Linux and Z/OS

- IBM zSystems’ developer tooling and programming language coverage provide complete equivalence to distributed platforms.

- DevOps platform components such as Git and Jenkins are readily available on IBM zSystems. An IBM zSystems build and deployment target can be completely transparent to a developer.

- Analytical tooling from IBM and our partners helps use AI to analyze existing COBOL and Assembler codes. The analysis stage is essential before remediation.

Finally, IBM added support to spin-up on-demand a z/OS dev or test environment within the IBM Cloud in minutes. This option provides clients with new agility to accelerate the development process.
Modernization Approaches for Software Platforms on IBM zSystems

We previously discussed Lift & Shift, Iterate & Extend, and Buy New as modernization approaches. Each of these can apply to application platforms on IBM zSystems.

Iterating and extending is the primary modernization approach adopted by enterprises running IBM zSystems. This approach has six variants.

**Expose mainframe data to new analytic or business functions:** the center of gravity for enterprise data in most enterprises is the mainframe. Hybrid analytics and AI platforms such as the IBM Cloud Pak for Data can access data sources via SQL or JDBC connectors. Mainframe data was traditionally considered to be locked away and hard to access. One approach to work around a perceived lock is to extract data and move it. However, we'd argue that a better way is secured virtualized access to mainframe data via the z/Digital Integration Hub.

**Exposing existing transactions as microservices via APIs:** z/OS Connect exposes existing mainframe transactions via API calls. This is the simplest way to extend current code into the hybrid technology platform alongside virtualized data access. Invoking an API from another microservice executes the transaction mapped to the API call. The transactional response is returned in a JSON format.

**Adding new Code in a current code base without modernization:** this is a viable approach for clients with a great understanding of their existing core platform. Even with code analysis, adding new capabilities can be highly complex for those who don't have great insight into their current state. Changing code without modernization does not typically de-couple monolithic user interface and data functions.

**De-Couple Monolithic Code and Containerize:** The most compelling approach to minimizing technical debt and creating future value is to refactor existing code for containerization. Analytic tools are essential to understanding how to demarcate functional and transactional boundaries. Automated migration assistants are also valuable. The containerization approach does not mandate a change of programming language from COBOL, although doing so might be desirable.

**Extending new business functions through language interoperability:** The ability for a Java application to programatically interface with COBOL and PLI assets within the same unit of work. This enables new business services to be easily added in Java, sitting right alongside COBOL and PLI business services, all managed by IBM zSystems.

**Extending new business function alongside an existing codebase:** The extend approach bridges existing code and new code. Several mechanisms bridge existing and extended code, such as the SQL or event-driven capabilities of the zSystems Digital Integration Hub, a messaging approach such as MQ, or maybe a data replication approach via the capture of changes to a database (change data capture). Given data gravity considerations, it will likely be optimal to host an extended function on the same IBM zSystems server, a Linux or OpenShift environment. Extending is typically done because an objective analysis of existing code demonstrates its re-factoring to be complex. There are considerations about new data created by an extended function. For simplicity, it is recommended that, where possible, new tables and fields are added to existing databases. However, this may not always be possible.
As already explored, the center of gravity for data and processing is a compelling reason enough to co-locate the code supporting new digital client experiences on IBM zSystems.

Organizations have avoided doing modernization for its own sake, believing that the ROI is not compelling. Given this, the catalyst for undertaking a modernization initiative is motivated by a combination of delivering what needs to get delivered, for example, a new digital experience, and how it needs to be delivered, the how being non-functional requirements and fit-for-purpose decisions.

While a detailed examination of use-cases is a topic for another occasion, two patterns outlined here:

**Exposing new digital channels that leverage IBM zSystems databases and transactions**

One intense focus area for accelerating digital transformation has been the innovation of client experience. The consumerization of IT is a ubiquitous theme as customers demand a customer experience that is aligned with their favorite mobile apps or in-store experiences. Simple examples center on anticipating a customer’s likely need, understanding their relationship history, and removing friction from every part of their customer journey. In many industries, digital customer experiences are reliant on information in a transactional core system of record. The public cloud provider’s recommendation has been to remove data from the mainframe in an attempt to shift the center of gravity. This has resulted in data movement charges, multiple copies of data, security risks, and data inconsistency. The reason for wanting to move the data was also based on a perception that it is hard to access. By leveraging new zSystems capabilities like the zSystems Digital Integration Hub, developers can securely access curated and virtualized views of raw transactional data through standard SQL APIs or as injected events into an event-driven architecture. From new digital channels, developers can write back to mainframe transactions through APIs or SQL. This is a terrific compromise as data on the mainframe is no longer locked from developers, and security and controls have not been abandoned.

**Delivering AI inference inside an existing transaction on IBM zSystems**

AI-infused decision-making has become prevalent in almost every business process. Firms are using AI for three reasons:

- Understand context and intent through natural language processing
- Create insight to identify a business opportunity or optimize an operation
- Detect risk or behavior associated with fraud, money laundering, or other crimes.

However, until recently, executing a machine learning model was typically not done within an on-platform transactional scope. AI was executed in one of two ways. First, it was offloaded to an off-platform service, often hosted in a public cloud. Doing so could introduce a second of latency into a credit card authorization which would break the transactional SLAs. Second, AI was performed after the fact, after the closure of a transaction, resulting in uncaught activity that might incur a financial liability.

The IBM Telum processor on the IBM Z16,
announced in April 2022, embeds AI capabilities directly onto the mainframe processor. This can enable customers to leverage the results of AI inference to better control the outcome of transactions before they complete.

For example, leveraging AI for risk mitigation in Clearing & Settlement applications to predict which trades or transactions have high risk exposures and to propose solutions for a more efficient settlement process. A more expedited remediation of questionable transactions can help clients prevent costly consequences and negative business impact.

For instance, an international bank uses AI on IBM zSystems as part of their credit card authorization process instead of using an off-platform inference solution. As a result, the bank can detect fraud during its credit card transaction authorization processing. For the future, this client is looking to attain sub-millisecond response times, exploiting complex deep learning AI models, while maintaining the critical scale and throughput needed to score up to 100,000 transactions per second, nearly a 10X increase over what they can achieve today. The client wants consistent and reliable inference response times, with low millisecond latency to examine every transaction for fraud.

AI-OPS. Hybrid cloud architectures are de-coupled and can run in multiple places. Arguably, monitoring hybrid application deployments becomes too complicated for a human operator, given the number of sites that code runs and the number of interfaces between code modules. AI-infused monitoring can keep track of instrumentation and health data emitted by each hybrid environment component. AI can understand related and unrelated events patterns and correlate them to identify non-obvious patterns and relationships. AI fuels observability, in other words, the ability to gain a deep understanding of code execution across multiple processing locations, understand bottlenecks and errors. AI can suggest how to remediate code issues. IBM’s Instana offering provides hybrid native application observability and performance management. Additionally, clients wish to optimize their resource consumption to ensure they pay for only the external services needed to support business objectives. IBM’s Turbonomic offering uses AI to recommend and execute configuration changes that support business objectives to grow or compress environments optimizing external spending with a cloud provider.

IBM’s definition of the hybrid stack includes AI. Why? While it may seem counter-intuitive or overstated, there are two clear reasons.

Firstly, IBM asserts that all applications will benefit from AI-infused decisioning and automation capabilities. Being able to adopt these techniques by invoking microservices are remote or co-located with data and other processing. For clients, the most critical decision is choosing where AI services are trained and executed. Machine Learning training often relies on highly sensitive data that cannot leave a client’s production environment. Knowing that a cloud provider requires data movement into their environment to complete model training activities becomes a non-starter. Secondly, a client may not wish to incur the latency of calling off-platform to an external ML model execution service during a low-latency fraud check. AI capabilities need to be available in hybrid building blocks as every new application will consume AI services.

Second, next-generation monitoring and operational automation systems require AI-infused operations or AI-OPS. Hybrid cloud architectures are de-coupled and can run in multiple places. Arguably, monitoring hybrid application deployments becomes too complicated for a human operator, given the number of sites that code runs and the number of interfaces between code modules. AI-infused monitoring can keep track of instrumentation and health data emitted by each hybrid environment component. AI can understand related and unrelated events patterns and correlate them to identify non-obvious patterns and relationships. AI fuels observability, in other words, the ability to gain a deep understanding of code execution across multiple processing locations, understand bottlenecks and errors. AI can suggest how to remediate code issues. IBM’s Instana offering provides hybrid native application observability and performance management. Additionally, clients wish to optimize their resource consumption to ensure they pay for only the external services needed to support business objectives. IBM’s Turbonomic offering uses AI to recommend and execute configuration changes that support business objectives to grow or compress environments optimizing external spending with a cloud provider.
A key consideration for clients modernizing their workloads and leveraging the public cloud is extending their security perimeter following zero-trust principles to encompass the hybrid cloud paradigm while not compromising regulatory or security requirements. The following section discusses the critical aspects of designing a secure cloud environment and shows how IBM Cloud provides a set of capabilities that helps accelerate the transformation. The following table highlights the core design principles that underlie implementing a secure hybrid cloud environment.

IBM Cloud has been designed with the exacting demands of the world’s largest and most complex organizations in mind. It relies on the cryptographic technologies shown to be most effective in the financial industry. Data that a client store on IBM Cloud belongs only to that client and can only be accessed by them. Clients can bring and keep their encryption key that no one else can see—not even IBM—and can build and run core business applications and workloads with single-dashboard visibility and multiplatform portability.

A key differentiator that IBM Cloud provides for Financial Services is the IBM Cloud Framework for Financial Services, a purpose-built framework with specific controls that address the unique risks of the financial services industry. The Framework was

<table>
<thead>
<tr>
<th>Defense in Depth</th>
<th>Cloud infrastructure will provide multiple, redundant layers of security safeguards to prevent compromise of the service from a single point of attack.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restricted Privileged Rights</td>
<td>No individual should be given enough privileges to misuse a system on their own and should be granted the minimum required authorizations to perform their activities.</td>
</tr>
<tr>
<td>Safeguard Data</td>
<td>Data is a valuable asset that needs to be protected from unauthorized disclosure, modification and destruction.</td>
</tr>
<tr>
<td>Continuous Controls Assurance</td>
<td>Security controls must be configured securely by default through automation and checked continuously for compliance to provide continued controls assurance.</td>
</tr>
<tr>
<td>Detection and Response</td>
<td>Enable traceability through logging, monitoring, alerting, and collection of audit information in real time.</td>
</tr>
<tr>
<td>Service Resilience</td>
<td>Availability of services and data are critical to operation of business applications, and they will incorporate multiple levels of resilience to maintain cloud services even after multiple component failures.</td>
</tr>
<tr>
<td>Secure by Design and SW Integrity</td>
<td>Follow secure development/operations processes and ensure software integrity through automation.</td>
</tr>
</tbody>
</table>
designed in collaboration with leading banks and with Promontory, an IBM company and a global leader in financial services regulatory compliance advisory services. The Framework enables an approach that delivers industry-specific risk-centric controls at the intersection of business and technology. It provides a standard set of controls spanning financial institutions, ISV / SaaS providers, and IBM Cloud Services (see Figure 2 below).

The IBM Framework enables the IBM Cloud for Financial Services to stay current as regulations evolve. This helps clients minimize the burden of regulatory obligations. Through the work with Promontory, the IBM Framework is intended to consider the regulatory requirements of financial institutions from over 75 financial services regulators in 24 countries, including nations in Europe, the Americas, and Asia-Pacific. The focus is on those requirements targeted directly at cloud usage and those regulations applicable more broadly to third-party risk management, cybersecurity, and data privacy. Regulations from various regulatory authorities, including those listed in Figure 3, inform and influence the framework and associated controls.

End-to-end data encryption with extensive control: IBM offers the industry’s strongest, commercially available, state-of-the-art cryptographic technology with IBM Cloud Hyper Protect Crypto Services. This service provides the unique “keep your own key” (KYOK) capability, based on FIPS 140-2 Level 4 certification, giving clients the ability to retain control of their own encryption keys and the hardware-security modules (HSMs) that protect them. IBM requires that ISV and SaaS providers agree to encrypt data at rest using KYOK. Unauthorized parties, including IBM Cloud for Financial Services personnel, will never have access to customer encryption keys. Whenever a customer application encrypts data with those keys, no other parties will have access to that customer’s data.

Workload-centric security by default: Each workload requires various access and security rules. IBM enables organizations to define and enforce such guidelines by way of integrated container security and DevSecOps for cloud-native applications with Red Hat OpenShift® as a service. Continuous security and compliance: IBM Cloud Security and Compliance Center provides a unified experience to view and manage security and compliance postures. Security, compliance, and operations teams can quickly identify security risks and vulnerabilities, govern cloud resource configurations, and centrally manage compliance with their organization’s and regulatory guidelines.

Multi-Zone Regions (MZR) leverage the underlying capabilities of IBM Cloud for Financial Services to enhance business resiliency and disaster recovery. MZRs comprise multiple, high-speed, low-latency,
interconnected Availability Zones independent from each other to ensure that single-failure events impact only a single Availability Zone. They enable financial institutions to locate workloads in specific geographies to fit their needs.

**Isolation and segmentation** provide compute isolation and network segmentation capabilities—meaning workloads can be deployed and managed with private-cloud-level security within a public cloud model. Compute isolation offers dedicated servers for cloud-native and VMware workloads, mitigating concerns around shared compute. With software-defined networking constructs, workloads and applications can be deployed within segmented network Availability Zones and secure connectivity across hybrid deployments.

**Prescriptive control implementation** reflects the targeted security and compliance requirements that should be met. For a standardized set of controls to be consistently implemented, assessed, monitored, and remediated, the control requirement must be prescriptive to minimize misinterpretations or incorrect implementations. Prescriptive controls help achieve the consistent and repeatable implementation of those controls across diverse technical environments.

**Logging and auditing rules** require that SaaS and ISV providers log all actions taken through the cloud portal, API, or command-line interface to be recorded in detail using IBM Cloud Activity Tracker. This provides standard logging of activity on systems and services and full session recording of precisely what actions operators take. This information is centrally stored and analyzed. The logging process is auditable to enable tracing of all steps, including logging both successful and unsuccessful events and giving role-based protection at all intervention points. The access logs are stored along with timestamps to assist in analysis and forensics.

**IBM Cloud Security and Compliance Center:** To aid in preventing compliance drift from the IBM Framework, IBM provides leading-edge tools which help client cloud administrators and application developers mitigate risk and manage compliance. These administrators and developers can automate manual processes with these tools. Enterprise security and compliance policies can be expressed in terms of controls and a demonstrable set of implementation goals. These elements can be composed into a collection of profiles, including NIST 800-53, applicable to financial institution workloads and applications.

**IBM Framework for financial services audit:** Security and risk executives (CISOs and CROs) and the managers of financial institutions using the IBM Cloud for Financial Services can gain efficiency in their internal and regulatory audits with the IBM Framework for Financial Services audit report. An independent third-party assessor performs periodic, rigorous assessments of the IBM Cloud for Financial Services against the IBM Framework. These assessments are more extensive than typical system and organization controls (SOC) 2 audits. They benefit clients by offering more visibility and transparency into the control effectiveness.
No discussion about IBM’s hybrid cloud capabilities is complete without considering how to accelerate application development and operations with the IBM Cloud Paks. Most applications consume higher-level capabilities to insulate a development process from becoming overly custom and increase the application delivery rate and pace. All Cloud Paks are designed to be deployed on Red Hat OpenShift in any location. Some are offered on a fully managed Software as a Service basis hosted in multiple enterprise cloud providers, including AWS, Microsoft Azure, and IBM Cloud.

Cloud Paks play an essential role in accelerating application development to support a client’s use-cases. They sit between the Hybrid Cloud Platform running on OpenShift and the client’s application-specific microservice containers. Cloud Paks share a common operational framework, and their business functions can be invoked via API calls.

Cloud Pak® for Data
Accelerates the end-to-end data lifecycle, including databases, data governance, ETL, reporting, data virtualization, natural language processing, virtual agents, and trustworthy AI.

Cloud Pak® for Business Automation

Cloud Pak® for Network Automation
Automate networks to deliver zero-touch operations.

Cloud Pak® for Security
Integrates security events to create predictive incident insights and assist with response and mitigation.

Cloud Pak® for Integration
Offers cloud-native messaging middleware, message brokering, connectivity and protocol brokering, event streaming, data integration, API management, and a security gateway.

Cloud Pak® for Watson AIOps
Offers observability with Instana, automation, and insight with Watson AIOps, optimization with Turbonomic.
Regulated industries are given specific regulations about how and where they run workloads. For example, governments assert that data, and in some cases, processing and operational staffing must be performed inside a geographic boundary. This is resulting in the emergence of issues of “sovereignty.” These requirements become burdensome for firms as they require:

- Partitioning of a data environment with locality-specific security and access controls
- The availability or absence of a trusted cloud provider within a geographic boundary
- Can data be processed remotely from its storage? Can information move across geographic boundaries?
- In-country infrastructure for both storage and processing
- Local staffing for technology and potentially business operations

Hybrid accelerators such as IBM Cloud Satellite alleviate these concerns by providing an integrated and automated hybrid environment that can be deployed on an as-a-service basis anywhere, in any cloud, or in any hosting environment.

IBM Cloud Satellite is a fundamental hybrid enabler providing managed distributed cloud offering based on the same open-source technologies as the IBM public cloud. Satellite distributes and manages cloud services anywhere needed — on-premises, in other vendor clouds, or at edge sites. IBM Cloud serves as the base for Satellite distributions; it abstracts complexities away from the individual locations while it provides a single, secured view of the public cloud where distributed services are observed and managed. Satellite enables users to manage cloud services and applications across public and private environments, including other vendor clouds. Satellite’s flexibility in integrating resources across environments extends to infrastructure, as-a-service operations, secure connectivity, and application lifecycle management.

A Satellite location comprises hosts — essentially, Red Hat OpenShift clusters on Linux hosts deployed on VMs or bare metal servers. Unlike other distributed cloud vendors, Satellite lets users pick the infrastructure they want to leverage, including:

- Already existing infrastructure in an on-premises data center or vendor colocation
- Already existing infrastructure in a private or public cloud environment
- In IBM Cloud or another cloud vendor
- Infrastructure as a service set up and fully managed by IBM
- An appliance that builds and runs IBM Cloud Satellite automatically

For example, let’s say a team has a Satellite location in an on-premises data center in Phoenix where applications are running in containers on a fully managed Red Hat OpenShift cluster. These application workloads are entirely self-contained and can be restricted to serve only serve local requests.
The ops team that operates these clusters uses a monitoring and control dashboard hosted on the nearest IBM Cloud data center – in this case, Dallas. From a dashboard, the engineer can spin up new OpenShift clusters in the Phoenix satellite location, deploy apps into the existing clusters, integrate services like logging and monitoring, and run the playbook of day two operations tasks. In short, Satellite extends services available on IBM Cloud into a company’s private data center. The apps in the Phoenix data center are self-contained, so they run without needing to communicate back to the IBM Cloud data center in Dallas. At the same time, the ops team consistently views and manages core cloud application services in IBM Cloud.

What if you need to establish another location? Like many IT teams in large organizations, let’s assume your team uses multiple cloud providers. A pool of infrastructure set up in a public cloud provider’s data center in Chicago is added as a Satellite location using the IBM Cloud portal. Once configured, the portal is used to create and control resources in the Satellite location. A Satellite location is similar to a self-contained IBM Cloud region running IBM Cloud services.

As a result, Satellite consistently delivers the same user experience everywhere. Satellite reduces the complexity of running workloads on multiple clouds and integrates them. Since IBM Site Reliability Engineering (SRE) takes care of the lifecycle for services, including updates and patches, developers get relief from tedious and repetitive tasks. They remain focused on more quickly achieving primary business objectives.
In Conclusion

Hybrid Cloud platforms are here to stay and help businesses accelerate time to value and customer satisfaction. They are grounded in well-formed architectural principles that are intentionally open and resistant to lock-in. Innovation and standards will inevitably continue to evolve. However, clients shouldn’t wait to get started. IBM brings four accelerating capabilities to support our client’s Hybrid journeys:

- Our most comprehensive open hybrid capabilities include the Red Hat Open Cloud Platform with OpenShift, IBM Cloud Paks, and server platforms like IBM zSystems and POWER. Our media run on every fit-for-purpose platform, including our client premises, hosting centers, Edge environments, and any Cloud provider.

- A breakthrough innovation method with the IBM Client Engineering Team and IBM Elite Team focused on offering an agile breakthrough approach rooted in Design Thinking, Agile Software Development leveraging IBM Technology capabilities.

- IBM Consulting with thousands of hybrid cloud modernization and modernization engagement leveraging Hybrid Technologies and AI.

- IBM’s expanding ecosystem of software vendors, application development, infrastructure, and consulting partners.

Please speak to your IBM Representative to get started or visit https://www.ibm.com/cloud/hybrid.

About the Author

John J Duigenan is the Global Industry Leader, Financial Services, IBM Technology. Additionally, he is an IBM Distinguished Engineer. He can be contacted at either John.Duigenan@us.ibm.com or https://www.linkedin.com/in/duigenan