Composable Disaggregated Infrastructure

K. Chilton

Abstract

This research considers Composable Disaggregated Infrastructure (CDI) as a possible solution for eliminating all unnecessary human-machine interaction in the modern data center. The intended audience for this research is executive management.

Keywords: Data Storage, Storage Insight, Composable Disaggregated Infrastructure

1. Background

Composable Disaggregated Infrastructure (CDI) enables dynamic composition of data processing systems from pools of disaggregated Compute, Storage, and Network components. The composition process is model driven, based on declarative templates that define what the composed system should look like in terms of its components and hardware capabilities. IDC describes CDI as "an emerging category of infrastructure systems that make use of high-bandwidth, low-latency interconnects to aggregate compute, storage, and networking fabric resources into shared resource pools that can be available for on-demand allocation." [1].

The disaggregated resource pools are configured from a diverse set of infrastructure that may include generic hardware, such as servers and switches, fixed-purpose components, such as Data Storage systems or GPGPU cards, or hybrid combinations of both, such as a Database Appliance. Some concepts limit CDI to exclude anything except generic hardware, but such a limitation is not optimal.

The infrastructure is configured and connected to expose sets of interfaces for the consumption and administration by CDI. These interfaces include those based on industry standard DTMF Redfish APIs and other standard protocols known by the CDI implementation and found on Converged Fabrics such as Ethernet. In some concepts, the infrastructure is configured to a base level by a separate "infrastructure manager" that prepares hardware for CDI, where other concepts that include more diverse platforms envision CDI also managing the infrastructure.

2. Definitions

CDI introduces several terms to help describe the concept.

Composer software: Provides flexible provisioning, orchestration, and automation of infrastructure resources to compose platforms of the exact elements required for a particular workload. When the workload ends, the Composer software securely makes resources available for other workloads.

Disaggregated hardware: Physical resources meeting compatibility/interoperability criteria are independent from the Composer software, allowing data center operators to choose best-in-class components, which can then scale independently.

Services: APIs and offload capability including individual pieces of business functionality that are independently developed, deployed, and managed as well as headless backends supporting HPDA, HPC/AI, SaaS, Cloud Native apps, and other front-ends that may even include end-user consumable applications. This may also include Function-as-a-Service (FaaS) elements.

Template: The heart of CDI is in its Templates, which describe what is needed to compose a solution, including all optional methods, and how to install, configure, control, and measure the solution against the business requirements. Templates have Standardized content and follow a Standard format, so they can be supplied by the creator of the solution to be deployed, whether hardware or software.

3. Purpose and Scope

3.1 Problem Statement

The major concern of almost all Data Storage and Data Processing system consumers is the costs required to manage these systems. This includes the costs of the efforts to plan, acquire, install, provision, deploy, operate, manage, maintain, license, and support the systems. These costs compel customers to look for alternatives such as Public Cloud and other off-prem solutions. However, the costs of these alternatives are not as low as technology could provide if the human-machine interface for routine tasks was replaced entirely by automation.

Another concern common among customers is the risks associated with the business. These risks include the inability to accurately forecast and respond to growth, the degree of service disruptions, data security concerns including data loss and ransomware, regulatory impact, and urgent changes in business workflows to respond to market changes and opportunities. These risks traditionally involve human factors that if eliminated will significantly reduce them. Automation is the most economical way of preventing failure, mitigating risks, and responding to faults.

A third concern among customers is the inability to correlate costs to value. Public Cloud has provided new insights into the costs associated with each business workflow. On premise, the assets, people, and infrastructure are typically shared between workflows, and the value each of those assets contribute to the workflow is not clear. To make decisions that will reduce costs and increase value, managers need clear asset-to-value data. Not only can these calculations be performed by software monitoring the environment, but automation can be used to control and optimize asset utilization to return maximum value and provide actionable recommendations for decisionmakers.

One final concern within the purposes that define CDI is the need to respond rapidly to changing technologies. A company's ability to quickly respond to changes makes it more competitive, including its ability to realize the benefits of new technologies. CDI automation provides the ability to non-disruptively and rapidly adopt new technology to lower costs and improve business results without a delay or need to bring training and other human needs into the process. This capability is necessarily coupled with a requirement to remove older technologies from the infrastructure that are no longer useful.

CDI automation should be inherently heterogeneous. The purpose of CDI is not to create a larger monolith than the systems it manages, but to free the customer to always select the best-in-class options and solutions that best serve the workflows that support the business. Not only should such a automation work with a great diversity of equipment from practically every vendor, but it should also cooperate with other CDI automations from other sources on the same infrastructure. This ability to use competitive or repetitive automations can be very useful in Cloud and MSP environments, enterprise businesses with non-aligned business units such as those resulting from acquisitions, businesses where a specialized CDI automation performs better for unique workflows, or cases of transition from one automation to another. The cooperation, preferably sharing infrastructure rather than dividing it among the automations, should be parallel without excluding the benefits of some hierarchical capability and needs the support of Standards to define the information interchange content and format to accomplish it.

There scope of what CDI automation ultimately should address is as follows. While it will be necessary to implement CDI in stages, this ultimate scope defines the direction, endpoint, and vision for CDI automation.

3.2 Maintenance and Operations Costs

A majority of OpEx in the data center is personnel cost. Reducing the number of people to support the data center has the highest potential to reduce data center costs. The outsourcing of IT functions to other suppliers, such as STaaS, MSPs, and Public Clouds, does not reduce the total cost of people for large enterprises, but only moves the cost from one line to another on the books. To substantially reduce cost, the number and expertise of personnel required must be reduced. For large data centers, this automation will save more than 40% of the total operational expense.

CDI automation should limit human involvement in Data Storage and Processing to only those things that cannot be automated. Humans must inform the automation of business requirements and goals, perform physical changes and repairs, interact with applications, and receive reports and visualizations to inform decision-making.

A significant problem in many geographies is that there are not enough skilled professionals to fill all the available jobs. The growth of IT demands requires more new IT professionals, but they do not exist. This creates a wage spiral that increases the costs of personnel, as one company lures talent away from another, only to lose their own talent to other companies. The number of qualified workers is a relatively fixed commodity, and the ideal solution is not to require many of them to accomplish business goals. CDI automation providing the technical expertise will make a huge impact to support the rapid growth in data centers in the information age by allowing the skilled workers to spend their time only on the tasks humans must do because automation cannot.

Automation and algorithms have proven their ability to continuously optimize better, faster than humans. The optimal solution for a large set of workloads is difficult for humans to accomplish and moving from configuration state to state non-disruptively is risky and challenging when humans must orchestrate the changes. Automation can achieve higher utilizations without sacrificing the ability to handle changes in the workloads or the environment. This higher utilization reduces infrastructure cost and should make on-premises the lowest cost option vis-à-vis Public Cloud or other outsourced option for larger customers.

3.3 Resiliency

Data loss, security, and unavailability of services are major concerns for every customer. The recent Ransomware awareness is just one aspect that Data Protection must address. Ensuring that the infrastructure and data are protected cannot be provided by humans, since over 90% of all attacks on data centers thus far investigated have been traced to humans either intentionally or accidentally creating the breech. [2]

Edge poses even more challenges for Data Protection and Service Availability. The ability to quickly respond to one of thousands of lights-out Edge sites is practically and financially impossible for customers.

Mistakes account for 30-80% of all downtime events, depending on how downtime and its cause is classified.

Approximately 1/3 of faults in a data center have hardware failure at their origin. Another 1/3 of faults are related solely to mistakes made by human operators performing change management. The final 1/3 are attributed to software and system bugs, incompatibilities, or other defects discovered during deployment or other change management actions. However, in almost all cases, the manifestation of these faults into service unavailability is the product of faulty design or actions of human actors. Most customers agree that more than 80% of downtime events could have been avoided if a human had not made a bad decision somewhere along the way. CDI automation is designed to move from operational state to operational state non-disruptively and to inherently recover from faults by returning to the original state, thus minimizing downtime to only extreme cases where the constaints placed on the environment by the business leave no fully operational solution.

3.4 Asset-to-value

Customers need proof that their infrastructure and software investments are returning benefits. Monitoring, visualization, and reporting support the needs of the customers to know that their operations are satisfying all of the business goals. CDI automation must optimize utilization to increase asset value and increase efficiency and provide customers with clarity about which assets are returning benefits and which need to be retired or replaced.

Systems today are not being utilized for maximum benefit. This is partially the result of managing latency and partially to plan for future growth, but mainly poor utilization is tolerated in order to avoid making changes that have a risk of creating problems. The core problem is twofold. First, the measures of success for business workflows do not map well to the measures and SLAs that IT decisionmakers are given to manage the assets. Second, the complexity of the assets and all the workflows is more than humans can manage. Automation has proven the ability to manage utilization much more successfully than humans, but need to be applied holistically across the entire stack to effectively manage the assets to satisfy end-user expectations.

The customers also need visibility to attach asset cost to business workflows. IT must chargeback to business units and processes accurately based on each unit's consumption of services. This is usually how IT receives budget and justifies expenses. With a clear picture of the cost of business workflows, companies can better optimize their business operations and increase profitability. This information must be accurate and provable. Often, this is a labor intensive process where IT managers retrieve information in spreadsheets, make decisions on the data, and generate suitable reports. In an ideal CDI automation, workflows are tagged and can be assigned attributes which inform the automation about governance, priority, value, KPIs, sensitivity, cost constraints, and other business success factors. Using these attributes, CDI can ensure business goals for workflows and provide accurate, automated statistics of asset consumption and derived cost. Coupled with business value for the workflows, the return on investment and asset-to-value are measurable, reportable, and provable.

Asset-to-value is just one business metric that holistic data center management automation can provide. CDI automation can reduce management overhead, save costs, and increase profitability for the whole business of the customer.

3.5 Technological Readiness

Technology is advancing faster than people can learn. For many years, the wide diversity of hardware and platform software in the industry was reduced to a few options. Now, the limitations of those options is leading to a return to diversity in hardware, software, and architecture to support the need for growth in the information age.

Infrastructure Diversity is becoming the new normal with not only x86 CPUs, but ARM, Risc-V, GPUs, TPUs, and other processing elements finding a home in the infrastructure. Networks are taking on a more diverse set of traffic and include the internet and many wireless options with different security and management considerations. Storage is no longer Block or File, but Object, Key/value, and other new emerging paradigms. Platforms are diversifying, with many new suppliers providing Container options and OS variations. New Database Management Systems are appearing, with geographic dispersion and new protocols to replace SQL. Integrated systems with many capabilities under one service-oriented cover are becoming common again. All of these changes are met with new workloads to support use cases like AI, Analytics, and IoT. This sort of challenge in times past would open new areas for specialization, but without time and people to fill these roles, current IT staff is overwhelmed to understand and respond to it all.

New technologies also drive complexity. There is no longer just one way to connect things, one way to deploy things, or one way to solve a business need. For instance, a business analytics workflow could be solved using a cluster of x86 in Linux Containers and server-based Storage, or maybe ARM is a better choice, or are GPUs and All-flash

Storage worth the investment, or should be offloaded to a Apache Spark based appliance for NDP? Instead of Spark, is Apache Storm a better option, or Snowflake, or several other choices? The ability of IT staff to work with their customers and provide knowledgeable service and to manage the wide diversity of options available in the most profitable manner possible is diminishing. It is not possible for customers to grow expertise to handle complexity growth. It does not scale. Technology must be provided to manage technology.

Customers, like most people, are not ready for change. New technologies are embraced in the mind, but are resisted when it might negatively impact the job. If we are to take advantage of new IT technologies rapidly for our busness concerns, we ideally will make it transparent to those who those whose jobs depend on supporting and delivering those technologies. CDI automation does that, but moving the expertise for the new technology to the technology creators, automating its installation, deployment, operation, and maintenance, and automatically fitting in with all of the other business of IT.

3.6 Agility

Agility includes flexibility, scalability, and decreasing Time-to-Value. Agility is the ability to respond quickly to changes. Agility is a key component to business success. An agile IT operation can increase IT value and reduces costs.

Time-to-market is a longstanding metric for businesses to optimize. It had been repeatedly reported that in a competitive business a delay of six months for product release will sacrifice half of the revenue that could be obtained from that product. Minimal time for customers to realize the value of differentiated capabilities results in a competitive advantage. IT customers provide increased value by rapidly delivering solutions in response to business needs. IT agility is key not just for IT, but for their whole business.

Nothing can be perfectly planned. Agility is not the fulfilment of projects according to schedule, but the ability to fulfil requests that are not part of the plan. Unplanned change is disruptive and the ability of IT to accommodate unplanned change is one measure of the quality and health of the IT operation. Businesses must respond quickly to changing market conditions and seize opportunity, so IT must also be agile to support the business.

The complexity and risk of changes in IT operations make change management a slow process for people to accomplish. Many steps and signoffs attempt to reduce risk and improve the chances of success. Typical change projects take six weeks just to develop a plan and gain approval from business and technical decisionmakers at major businesses. The problem though is not the technology, but that people will also be implementing the change and the likelihood that they will make a mistake is the risk.

IT agility must be left to automation. The constraints placed on the automation will be the limit to the agility the automation can achieve. CDI automation is designed to provide the most agile response to changes possible. Change is what CDI manages, whether deployment of new workloads and infrastructure, changes to existing operations, or removal of hardware, software, or data.

Heterogeneity is another aspect of agility. The recent supply chain issues have highlighted the impact of single-sourced solutions for products and services. IT operations have had to turn to alternative suppliers to satisfy the requirements of their businesses. The ability to take any tool available and accomplish the job is crucial to agility when resources are scarce. Customers will not soon forget the current environment and will be more forceful to ensure diversity in infrastructure in the future. CDI automation has a great opportunity to handle a wide variety of alternative methods to accomplish the same business goals with whatever resources are available at the time the solution is needed.

4. Market Competitive Landscape

Four classifications to current offerings and futures called “CDI” by vendors:

VM Administrator Automations via Single pane

This set of offerings attempts to define CDI automation as existing automation offerings that unify management across VMware hypervisors to provide a single pane of control and automation. Two vendors who have used this method are HPE with their Synergy offering and Dell.

Specialized Hardware

Some companies have called their specialized hardware as providing a component of CDI. Each of these is a vendor-specific solution that creates a hardware lock-in for the customer. One offering in this category is Liqid’s Matrix which is a PCIe-based, rack-level system limited to a maximum of 255 devices. AWS Nitro also fits into this category, since it uses in-house HBAs to accelerate, offload, and manage the network.

Distributed Operating System

Distributed OSs have been university projects for decades. They have yet to see commercial success because other solutions provide better practical results. One example of a company attempting to provide such a solution is LegoOS, which trades CPU performance for resiliency. While the improvement in application resiliency is modest (less than 50%) the resulting decrease in performance is expensive.

Distributed Hypervisor

A middle ground between Distributed OS and hardware solutions is the Distributed Hypervisor. It has mostly the same problems that a Distributed OS has, some of which are worsened by the larger granularity of the information to manage. Analysts and customers see little benefit in these hypervisors than what they can get with VMware. Companies in this category take different approaches to implementing their products. GiantVM uses KVM and QEMU to create the virtualization, where TidalScale modifies the kernel itself.

Summary of Market Offerings

This definitions for CDI provided by vendors limits it to only solving the problem of application deployment and management on servers and does not address the wider automation of the whole data center and its components. They are not addressing the basic data and workflow models and are ignoring the business aspects of IT operations. None of the vendors appear to be learning from what is suggested by academic research. These vendors are not considering all use cases, applications, workflows, and systems present in a typical data center. These vendors have not provided for diverse resources and emerging technologies in the data center, such as TPUs, FPGAs, and Computational Storage. As such, there are no CDI automation solution in the market nor hinted to by major IT vendors. This does not mean that Dell and HPE are not working on it, because they have good reason to do so to support APEX and GreenLake. Public Cloud providers also have reason to provide CDI, but as of now, information about such a direction has not been released.

5. Differentiating Opportunities

Certain aspects of data center operations can inform a more practical solution than envisioned by the academics. The typical vendors and many people involved in IT technology have a perspective that begins from either a CPU or an OS or an application. A better, customer-centric view involves considering the customer's business customers, the customer's operational environment, and the customer's need to manage and get value from data. This moves the problem from a neat theoretical structure to a complicated practical reality.

Data Centricity

The first important aspect to address in creating a differentiated offering is to take a data-centric approach to the architecture. Data is valuable, but moving it is disruptive and copying it is expensive. CDI automation must consider the placement of data itself as core to the operational strategy. Data is the center of the data center. All other operations are peripheral to where the data is. If the data originates at the Edge, moving it from the Edge to another place is a strategic decision that the automation must make, just as it would when moving data to and from Public Clouds. Some vendors and most academic ideas for CDI are CPU-centric and solving "big memory" and parallel computing problems. In data centers, most problems are data problems. Locality, ingress, egress, resiliency, accessibility, protection, and security for the data drives the CDI automation decisions of how to compose the infrastructure to support a workflow, not just performance. The constraints and requirements are business centric, but the optimizations are data centric.

This means that the Composer needs to be Data Aware. It needs to keep track of the locations, performance, access, and security aspects of the data. Its preference is to move services to the data, not data to the service, and only moves data if necessary to satisfy hard SLAs or as part of a high-level, long-term strategic operation to balance and optimize the whole solution of the environment.

The Composer also need to be Content Aware. The format, user mapping, and relationships between data sets and content help it to prepare for future changes and resist the tendency of dependencies to sprawl or data gravity to pull other data sets in one direction. Content awareness also informs analytics and visualization of data, security, and value. The content also provides for data-structure-informed operations and balancing, as well as improving prediction of workload behaviors on diverse hardware.

The Composer should be Value Aware. Data varies in its importance to the business. Select portions of data need higher levels of security, protection, or performance. Knowing the value of the data can lead to further self-management of SLAs and lifecycle decisions. It can identify opportunities for the business, remove human-machine issues, and reduce cost.

Business Driven

The Composer is constrained by business decisions. In an ideal world, data can be protected better with thousands of copies than with just two, and everything wants the best possible performance. For the business though, this is not practical, and costs must be managed. Therefore, there exist constraints on operations related to costs, performance, priorities, compliance, security, and other factors. These factors cannot be fully guessed by the Composer, although if not assigned by the customer can default to some pre-determined values based on the constraints from similar customers.

The customer's business is composed of processes. These business processes are expressed in workflows that may be continuous, periodic, or ad hoc. Each workflow has a value to the business. Workflows have priority relative to other workflows. These are known by the business and are codified and expressed to the IT operations. For example, IT commonly provides service levels in its catalog to the business units, such as Gold, Silver, and Bronze. These levels are simplified groupings to express differing performance, protection, and costs. The business matches workflows to these SLA levels, and IT operations is expected to satisfy those agreements. CDI automation can support a much finer, more specific set of requirements, allowing for a more optimized cost structure and more control over the experience by the end user. This avoids over provisioning, where for example the business may require high protection of data but not high performance.

By allowing the end user, the business, to make the decisions based on business goals, the operation of the IT shop is vastly simplified, even though the operation can support a more complex set of requirements. Agility is increased, costs are reduced, and business owners can manage workflow costs directly.

This is just the beginning of the opportunities that are available to satisfy business objectives using CDI automation. Like Public Cloud, CDI automation increases customer satisfaction by instantaneously providing solutions to business needs, allowing IT professionals to work on strategic technology investments to lower costs and application development to increase IT and business value.

Two-Part Approach

There are two approaches to management. One is an "imperative" model that describes the "how" to accomplish something, usually step-by-step, and is the traditional model used by customer scripts. Most Ansible scripts are also written as imperative. The other approach is a "declarative" model which only identifies the desired end state and not how to accomplish it. HashiCorp's Terraform is declarative to the user. Both declarative and imperative models have advantages and disadvantages – each has its own benefits and problems.

The declarative model is how the IT operations work from a big picture view. The business defines what the business wants, and that is the desired end state. How this is accomplished is details the IT operations staff provide. With CDI automation, though, the end user requirements are added to the existing requirements to define the end state, and the automation generates and executes the imperative processes necessary to move from one state to the next. So, CDI automation includes two management models in one: externally it provides a declarative model, internally it is constantly developing and executing imperative scripts to move the operation from state to state to state.

Optimized CDI Clusters

Years ago, businesses simplified hardware acquisition and installation by providing Converged Infrastructure (CI), where all the components needed were installed, configured, and tested prior to delivery and ordering was accomplished through a single SKU. This simplification evolved to Hyperconverged Infrastructure (HCI) where not only was the hardware preinstalled, but also the hypervisor and platform software to manage the unit. HCI is still popular, but it has problems as well. CDI provides the possibility of HCI convenience without the scalability, vendor lock-in, and performance limitations of the typical HCI product.

An optimized CDI cluster contains Compute, Network, Storage, and other components, but is managed by CDI automation that can be external to the CDI appliance. CDI is not limited to specific hardware configurations, so a single CDI automation will manage all CDI clusters, other systems in the infrastructure, and external resources such as Public Cloud. Where HCI units were difficult to expand or interoperate, CDI logically decomposes systems into their assets and services and then composes them into the environments needed to support the total set of work at hand. CDI automation makes HCI obsolete – CDI can make data centers from bare metal boxes and manage everything as one system.

CDI clusters can be built to optimize classes of workloads. For instance, a CDI cluster could be GPU-heavy, so the Composer will move workloads most benefitting from GPUs

to those clusters, freeing other resources in the infrastructure for other work. Edge systems are another example of a CDI cluster opportunity where the particular hardware needed to perform tasks at the edge can be pre-build and managed centrally according to global requirements.

CDI clusters reinvigorates the on-prem hardware business, where vendors can create hardware managed by CDI automation to give customers the best value for their money for their specific needs. Because CDI responds to workflow attributes, translating business workflows and goals automatically to platform operations, specialized Storage appliances where specific data is needed will reduce costs if built to optimize specific workloads. For instance, the need to support high-performance HPDA, HPC, AI training, and deep analytics workloads can benefit from CDI appliances with FPGAs, NN engines, and NDP capability close to the data. Inference engines built into Storage enhance real-time decision support, MSS, and AI to support cataloging for Analytics. Storage can also provide services specific for Metaverse, Database, CDM, or Video. CDI automation should embrace and enable Computational Storage, NDP, and embedded Database Engines so Storage system can deliver more value.

CDI appliances can also be optimized for external XaaS management, providing a "Cloud-in-a-can", which give SMBs an alternative to Public Cloud and a growth path to support their growing business.

True Lights Out

CDI automation makes the operation self-healing, self-updating, self-migrating, cost optimizing, and green. It obsoletes many as-a-service offerings for large customers, like APEX and GreenLake, because the cost savings of CDI and the minimal management that remains is much lower than paying a third party to do work that is already being done in-house.

CDI automation is inherently built for low-skill partner management and for simple, brainless Customer Replaceable Units. This reduces support costs for customer and vendors. Partners require less training, and most hardware service operations can be performed by customers themselves, allowing access to small businesses without the need to scale vendor-supplied high-touch service capability.

CDI automation leverages Public Infrastructure and Cloud for cloud bursting and resiliency. Public Cloud is a fallback part of the customer strategy, not their primary deployment target. CDI automation leads to real re-patriotization of workloads because data center costs will be lower than Public Clouds can provide and still make a profit.

6. ONAP for Carrier

The TMforum created standards to allow carriers to collaborate on solutions for AI, support Autonomous Operations, enable plug-and-play IT through Cloud Native technology, move the carrier business beyond just connectivity services, improve customer experience and trust, and enhance the ability of the carrier business to find and grow talent. Several of these themes overlap with the more general goals of CDI automation and the efforts of those in the TMforum can be leveraged to inform CDI and support its development. One outcome from the TMforum efforts is ONAP.

ONAP, the Open Network Automation Platform, is a comprehensive platform for orchestration, management, and automation of network and edge computing services for network operators, cloud providers, and enterprises. It is Open-Source with a steering committee featuring people from Deutsche Telekom, Orange, Huawei, China Telecom, AT&T, China Mobile, Ericsson, F5, Bell Canada, Wind River, and Nokia. ONAP is a founding member of Linux Foundation Networking (LFN), which aims to increase collaboration and operational excellence across networking projects.

ONAP provides the ability to dynamically introduce full-service lifecycle orchestration (design, provisioning, and operation) and service APIs for new services and technologies without the need for new platform software releases or without affecting operations for the existing services. It is design to provide scalability and distribution to support many services and large networks. It has metadata- and policy-driven architecture to ensure flexible and automated ways in which capabilities are used and delivered and will enable sourcing best-in-class components.

ONAP provides instant starting features for CDI. For example, ONAP defines a clear KPI monitoring and telemetry concept and code that can be used in the control loop of CDI automation.

ONAP envisions a component that can manage the infrastructure but has yet to implement it. Early versions of CDI automation can interface with ONAP and the Composer can create the resource pools for ONAP to deploy. CDI virtualizes and further optimizes ONAP for Carriers and can ensure continued steady state operations by managing the rest

of the Enterprise with ONAP as one of the solutions it manages.

In this way, ONAP provides an instant customer for CDI automation, with Carrier customers around the world ready to consume offerings designed to work with it.

7. Example CDI Automation System

CDI automation can be implemented as a two-level automation. The "physical" level discovers and manages bare hardware. At this level CDI identifies, installs, tests, classifies, and catalogs hardware components in the inventory and presents components for CDI to use to compose solutions. The "logical" level uses these components to install, deploy, configure, and manage the systems, services, and applications that provide the solution. The physical and logical levels are constantly active, with the physical transforming hardware into needed components and the logical transitioning the operating environment from state to state. Should hardware need to be reinstalled to better optimize the solutions, the logical level will move operations off the re-installable hardware and the physical level will reinstall, test, and make the hardware available for logical layer use as a new component.

An example of the major functions, interfaces, and information flow of the first level of the CDI automation is shown in Figure 1.

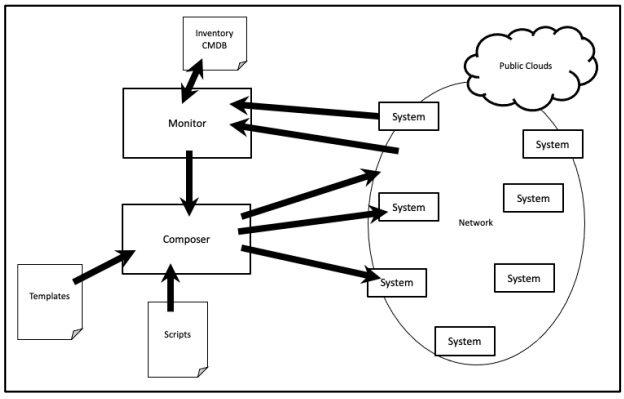

Figure 1: Major physical-level CDI elements

At the physical level, the Monitor task continuously collects information from the environment. Hardware appearing on the network or changes to the network are detected. The Monitor uses the CMDB to inform the messages to the Composer about the state of the hardware in view and alerts the Composer of changes to the environment.

The Composer task receives the operating state information from the Monitor task and uses this information to determine if the current state is optimal or if change management is required to satisfy the goals and requirements of the business.

The Composer gains its knowledge about the hardware from Templates and uses Scripts (such as Ansible) to set up the hardware and install agents. It can bring up machines and network from bare hardware to running and configured applications, as well as update systems and software, automate change management, and orchestrate data movement/migration.

With the Composer and Monitor, CDI can autonomously bring up hardware into a platform ready to install services and compose those services into applications. For example, if the Monitor reported new hardware, the Composer may then perform a physical-level installation of a platform on that hardware to make it ready for later use by the logical layer. The installation of the platform would be detectable by the Monitor who would close the loop by informing the Composer of the completion of the install and any test results.

While this example shows one Monitor and one Composer, it is possible that there is a plurality of either or both. This may be done for resiliency of these components of the CDI system, to improve parallel performance, for geographic considerations, or to support vendor-specific elements.

An example of the major functions, interfaces, and information flow of the second level of the CDI automation is shown in Figure 2.

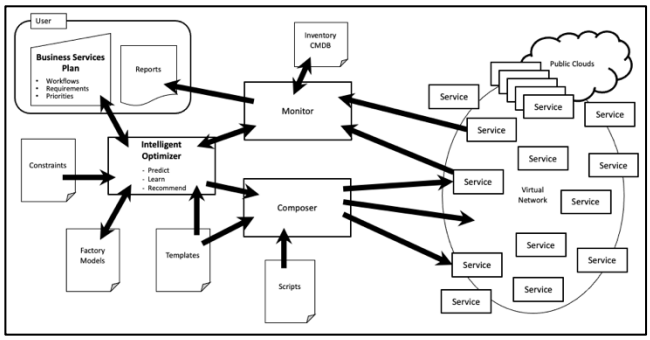

Figure 2: Major logical-layer CDI elements

In the logical layer, the Monitor sees the agents installed by the Composer, interacts with the Inventory (CMDB) and is available to provide reports on status for the Customer. The Monitor identifies any KPIs, faults, changes, and requirements discrepancies. It also is aware of data content and its location, protections, and security. It also provides alerts, alarms, and call home capabilities.

The Intelligent Optimizer collects the business workflows, priorities, requirements, and the needs the business has for information – CDI is end-customer-driven to automate and remove human interaction as much as possible. The Optimizer receives the current information collected by the Monitor and, using algorithms and AI, informs the Composer of changes that need to be transacted to meet business requirements, improve customer experience, and reduce cost. It also will provide customers with recommendations to better satisfy customer priorities, including hardware acquisition, interfaces, licenses, and topology. The Optimizer reports can also include suggestions for improvements to the supplied requirements based on the Data itself, available capabilities, Factory-supplied models applied to the environment, or new technologies that will better meet customer business goals. Some of these suggestions will recommend additional security if sensitive data is detected, additional DR protection for valuable data, or additional performance options to shorten the time to Analytics results.

The Intelligent Optimizer also uses Factory-supplied models and base training as well as the Templates to inform its training and calculations. It can create or supplement Templates based on machine learning, collect recommendations from external applications and services (such as Spark), and interface to other knowledge sources via APIs. The Intelligent Optimizer adds in-field experience to improve replication, security, service and data placement, alternative configuration strategies, and best utilization of specialized diverse hardware and refines configurations, topologies, and low-level SLAs to optimize agility, resiliency, performance, and cost.

The Optimizer will continuously create a long-term strategy that can be reviewed and steered by the Customer through application of additional requirements. The Optimizer also can be supplied with Constraints which come from other sources, such as the vendors, to limit its behavior particularly for specific industries or geographies, such as ensuring GDPR and/or FR for the EU market. Constraints also provide defaults for situations where AI models and customer requirements do not set bounds for parameters.

Ideally, there is one Optimizer. The Optimizer does not need to respond to inputs within milliseconds and can be moved and restarted as needed. The Optimizer state is held entirely within Templates and not in memory to simplify recovery. While the Optimizer can run on a cluster with many threads to improve throughput, it should avoid becoming split-brained.

Based on input from the Optimizer, the Composer will install and manage applications according to supplied SLAs using the Templates and Scripts as needed. Any changes are detected by the Monitor and the Composer can react to these changes to bring any exception back to meeting SLAs. Any hardware failures will be recovered by following predefined mitigations or refitting the Optimizer's last solution to the remaining hardware, and the Optimizer will begin work to re-establish any lost resiliency provisions based on the current state reported by the Monitor.

8. Example Application Deployment

In this example, a DevOps customer will request a MySQL server to be made available with a set of requirements that may include resiliency, security, capacity, and performance. The requirements that are available to set are defined by the CDI Template for MySQL.

Before the customer even begins to use the system, CDI has already started to work with the environment. It already has templates for the existing hardware, required services, and MySQL. The hardware has been discovered and at least some systems have been brought to a base configuration. CDI is already in closed loop with a current state, which may include factory defaults for new components.

The Customer creates a request for MySQL

While this example of installing MySQL is a typical DevOps operation that a developer would use, business units might be more likely to invoke a business workflow that might include installing a new MySQL server along with other applications. The flow is similar, but a more complicated workflow may have more components that can be installed in parallel.

This request for MySQL inherently brings a set of requirements, including performance, cost, security, and reliability. The request and any workflow is tagged with all of the attributes and user information. The Optimizer may receive this request and the attributes from a Factory-supplied web portal or API.

The Optimizer predicts, based on the current state and feedback from the Monitor, a proper deployment that includes this new MySQL instance using the Templates. It forms declarative code to describe the entire operation under all the affected Composers' control.

In this example, the Optimizer could use a Template that describes adding a Container, a Storage Volume, and a MySQL K8S package. It will specify the number of ports for

the container and MySQL and their performance goals. It will set the capacity, protection, and performance SLA of the storage volume as seen by the container, as well as required security and replication attributes to the current declarative code that describes the current operation.

The Optimizer then pushes the new declarative code to the Composer(s), receives a ticket, and continues to track that ticket. If the ticket is rejected or fails, the Optimizer will retry with another solution. If all solutions fail, a recommendation is generated, and the Customer request fails.

The Composer receives the declarative code

After receiving the new state description, the Composer creates a To-Be Plan (valid state to valid state) from the existing state using the Templates. This To-Be Plan is selected to include the minimal number of changes (cost-based analysis) to implement the new solution, as well as rollbacks to the current state and any required test scripts to verify the correctness of the new installations.

If a To-Be solution is found, the Composer supplies a ticket back to the requestor, otherwise it rejects the request. The rejection of the request tells the Optimizer that its solution could not be implemented given the current state.

After forming a valid To-Be plan, the Composer begins to execute it, including use of supplied Scripts, with any failure triggering automatic fallback and ticket failure. Plan steps are logged, reportable, and inspectable, and may be concurrent with the execution of non-intersecting plans

MySQL is now installed and ready for use, the ticket is marked completed.

The Monitor is configured by the Composer

The Monitor will attach to, subscribe to, or start polling the new KPIs and logs, depending on the changes that occur. The new KPIs are made available for subscription by the Monitor. The changes are noted in the CMDB, if needed, and real-time infrastructure and utilization reports are updated.

The Monitor will update real-time reports and dashboards to reflect the current state. Cost accounting, history, debug, and other information will note the changes and begin recording.

The Optimizer sees the completed installation

Once the Monitor reports the new state and the Optimizer recognizes the requested changes, it subscribes to the new KPIs with the Monitor(s). It includes the new MySQL instance in steady-state continuous optimization and tracks the status from the Monitors to ensure that it, as well as everything else, is working optimally.

The Optimizer then informs the customer, according to the method of request, that the service/application is now available and within required parameters.

9. Details of Major CDI Elements

9.1 Monitor

The Monitor can work with several KPI sources. The sources may include existing APIs for resources that report data such as performance or health, such as Storage systems. Agents running in platforms can collect data from the platform and may be specialized to tap into particular application metrics. KPI information may also be provided by SDKs built into applications/services and vendors can create Open-Source code for developers to incorporate in their in-house applications to inform the CDI automation. It is also possible for there to be a market for third-party specialized monitors, such as host-based heartbeat checkers, log scrapers, IO sniffers, OS loadable libraries and kernel modules. ONAP has monitor agents that could interface directly to CDI Monitors.

There may be multiple Monitors in an enterprise to support geographic dispersion, Disaster Recovery, or 3rd party IT providers such as MSPs. Multiple Monitors may help with security partitioning. Having multiple Monitors in one CDI may be necessary to support multi-vendors toolsets, particularly to support online migration from one CDI vendor to another. Multiple Monitors may help separate client customers as an option for multi-tenancy. Depending on implementation, multiple Monitors or even hierarchical Monitors may be a better performance choice than trying to scale a single Monitor.

Discovery

The Monitor finds hardware, systems, platforms, and applications in the environment. A simple method of discover on TCP/IP is to have the Monitor provide DHCP, which limits the effort needed to scan the network for unregistered hardware. The Monitor will also find agents using standard PnP methods. The Monitor matches discovered inventory with the known Inventory (CMDB) and maintains status and history in the CMDB. A CMDB is provided as part of the CDI package, but CDI should be able to work with other customer supplied CMDBs and standard DBMS systems used for providing a CMDB. Having CMDB

responsibility allows CDI to directly provide updates to current and historical infrastructure knowledge management for the Customer including location, identity, contracts, licensing, ownership, users, relationships, operational status, hardware revisions, software versions, investment and operating costs, value, usage, utilization, connectivity, performance, faults, and service activity. Accesses to the CDI CMDB will pass through to the customer CMDB and any fields missing from the customer supplied CMDB will be supplemented by the CDI CMDB. The information from the Monitor can be used to provides resource and topology visualizations for clients and is required for the Optimizer to make informed decisions regarding asset usage and create quality recommendations.

Collects Status

The Monitor receives KPI messages from the environment. This can also include information from Data Center utilities, such as power consumption, costs, and temperature. The Monitor collates logs, alerts, and alarms, and creates its own logs, alerts, and alarms including call home notifications and details for vendor support systems. The Monitor can identify gross faults, exceptions, and requirements violations, and can work with interfaces to Intelligent Debug resources, like DME, so the information can be used for other service processes to effect repair.

The Monitor provides updates and operational statistics on resource usage and behavior for clients. While supplemental tools can create customizable reports for customers from the information provided by the Monitor via its APIs and the CMDB, the Monitor also should directly provide comprehensive real-time reporting and dashboards on operations and usage.

9.2 Intelligent Optimizer

Analysis can include optimizations for agility, resiliency, performance, and cost for replication, security, service and data placement, alternative configuration strategies, and best utilization of specialized diverse hardware.

It will provide recommendations/reports the customer needs to better satisfy priorities, current and trending. Recommendation reports may include hardware acquisition, interfaces, licenses, and physical topology changes.

It is called “Intelligent” to imply that it involves AI technology to include reinforcement learning and the ability to leverage information from a variety of other sources, such as a vendor model and data, open customer-driven pool of shared experience, and 3rd parties.

Collects Information

The Optimizer Supports collection from multiple Monitors for Edge, Cloud and other 3rd party suppliers, local and remote data centers, and DR bunkers. Wherever a Monitor is running, it should be known to the Optimizer and its collected statistics included in the decision-making.

The Optimizer also collects requirement information from its customer portals, including “what if” scenarios the customer wants to model but not actually deploy. The Optimizer collects input via its APIs from end customers or third-party tools, such as Terraform.

The Optimizer reads the Templates, Constraints, and applies Factory Models/Data. The Optimizer uses the Template format and Template Data Store to create and update version-controlled Templates to add persistence to the temporal information it generates, such as the current operational state, as well as information collected from the end customer, the vendors, and other 3rd party information sources for resilience and to speed recovery and upgrades. Using the Templates for persistence allows the Optimizer to be Cloud Native and removes barriers to migrate to an Optimizer from another vendor.

The Optimizer also uses information from the CDI CMDB, such as cost and physical connectivity. The Optimizer also creates auditable logs of all information exchanged.

Performs Analysis

The Optimizer compares current and historical metrics against requirements, performs reductions of exceptions, and creates a failover strategy for each implementation. If the Templates describe multiple rollback methods, all invalid ones are excluded that do not lead to the current state. It performs Intelligent (including AI) analysis and solving for requirements, improvements to customer experience, and cost reductions.

The result of the analysis is an ordered set of possible implementations that meet the requirements and fit the available resources, sorted according to priorities, to data considerations, and to least deviation from the current declarative state. Implementations recently rejected by the Composer are marked as invalid and the results are saved as a Template.

The Optimizer also creates recommendation information during Analysis that identifies better solutions that will not be selected. These better solutions may include impediments such as having too large of a distance from the current state, hardware that is not available or already fully utilized, customer requirements that are expensive or seemingly over-constraining, licensing or service impact, security or protection risks, or require physical topology changes. The recommendations are collected and if appear frequently enough and are deemed either important enough or simple enough are included in recommendation reports to the customer. If the analysis fails to find any solution to meet end-customer change requests, a report is generated with the top recommendations that will.

Pushes change management to Composer(s)

The desired solution, which is a set of declarative code in Template format, identifies all required services along with any parameters to be used in the configuration to meet requirements. The Template lists the full set of KPI metrics that will be needed ensure that the resulting operation matches the requirements, any SLAs that must be enforced by resources and services, and any limitations, exclusion, and disallowed options. These exclusions may include prohibiting the use of hardware that does not appear healthy enough to be used, implementations that analysis has scored too low, and implementations that are known problematic from prior attempts.

It may be necessary for the Optimizer to sequence states from the current through intermediate states to the desired state if the deviation from the current state to the desired state involves overlapping end-customer requests or multiple Composers. In such an event, the Optimizer will prioritize the transitions and issue them in sequence, saving unfulfilled dependent request Templates for re-evaluation on the next event.

The Optimizer pushes the next state to one or more Composers via the Composers' APIs and identifies their respective Template containing the declarative code for the next state. The Optimizer then receives a transaction tag, a ticket, from each Composer to track completion of the request if the Composer can transform the declarative code into an actionable change management plan. The Optimizer uses the CMDB to track all outstanding tickets.

9.3 Composer

There may be multiple Composers in a data center that do not share hardware. This allows Composers to be separated by geography, for Disaster Recovery purposes, or to support 3rd party providers. Some complex systems may present themselves with their own Composers, such as HCI systems or CDI appliances. Separation of Composers may also be necessary for Security or Failover partitioning, to support multiple client customers in a multi-tenant arrangement, or due to lifecycle or migration operations.

The Composers create version-controlled, auditable logs of all information exchanged, plans, and steps taken.

Workflow Automation

The Composer is the most famous part of CDI and is mainly responsible for the installation of Hardware, Platforms, and Applications. It operates based on the inventory of CI items recorded in the CMDB and the To-Be plan it creates from the Templates. It installs bare metal and monitor agents for Operating Systems and Virtual Machines.

If provided in the Template for the hardware, the Composer runs verifications and tests of that hardware. Hardware is managed through interfaces such as the standard DTMF Redfish APIs and servers can be PXE booted into provided base platforms for that hardware, such as hypervisors, OS with Container, or as an application specific appliance. The Composer can load the KPI agents into standard platforms to support the Monitor.

The Composer initializes network switches from defaults into secure NFV configurations to match the plan. KPI agents installed by the Composer across the infrastructure will verify the topology and its capabilities through the Monitor.

The Composer installs applications and services from Kubernetes, Git, filesystems, or other repositories according to the plan. If provided in the Template for the application or service, the Composer runs post-install verification of installed software elements. The installation will usually involve the execution of Scripts (e.g. Ansible, Puppet, Salt) that include the imperative code needed to fulfill the state transitions inferred from the declarative code.

The Composer also tears down and re-installs hardware as needed to create the components needed to fulfill the requested operating state. It decommissions hardware after migration of platforms completes, allowing for obsolete hardware to be removed. The Composer can power down hardware to meet cost and green objectives and wake them as needed, if these methods are described in the applicable hardware Template.

Orchestration

The Composer drives configurations and resource assignment (configuration and change management). It translates declarative code from the Templates into imperative code for execution. It does this by creating a "To-Be plan" which describes how to move the environment from the current state to the desired state with the fewest changes. The plan includes intermediate states, each of which is a valid, stable operating point, and lists the operations and parameters needed to make each step in the transition. It structures the data repository topology according to attributes and content to minimizing data movement and access bisection. The Composer can create code and scripts needed to be execute via external engines, such as Ansible, Juju, Uyuni, Salt, from Scripts in the CDI library and other algorithms.

The Composer sequences change management to minimize effort and avoid collisions or disruption. Each change is a minimum distance move from state to state. The Composer provisions and configures storage and network connections, including SLAs, QoS, and Traffic Management, configures applications and platforms according to Templates and Optimizer parameters, and configures the KPI agents.

The result moves the environment from the current state to new state, according to processes described in the Template and Scripts. If any part of the To-Be plan fails, the Composer executes predetermined rollback plans as needed. Without a rollback plan, the Composer will re-establish the current state in the environment and fail the change request.

Composer actions automate migration since the Template for the next state can exclude hardware and software options. The Composer will move the environment to its next state automatically, and if the hardware describes any method for secure erasure or power down, it would be deployed if not excluded by the Optimizer.

9.4 Templates

The information in the Template is the key to CDI and what makes it different than IaaC alone. Template formats must be standardized and supported across all vendors, or CDI will not succeed. The Template should use standard encodings like JSON, XML. Template formats can allow for encryption, so vendors can hide "secret sauce" about management of complex systems.

Hardware Template

Templates to describe generic hardware are simplest Template to create. It includes a description of the management access, e.g. Redfish, Swordfish, PMCI, REST, or SNMP. The *CIM Infrastructure Specification* should meet the requirements for many hardware templates (SMI-S, WEBM). Component capabilities for reliability, availability, security, and inbuilt services must be included in the Template. Performance metrics for hardware often include many dependencies, which can be described in the Template to improve prediction.

Hardware Templates will usually have an accompanying set of Scripts. The Scripts provide imperative code to describe how the hardware can move from state to state. For example, server hardware can move from powered off to powered on. A Script can be provided containing the necessary code to perform a Wake-On-LAN (WOL) function for hardware that supports the function. The Template for that hardware will identify the capability and the function that will accomplish it. Scripts for hardware servers will also show how to identify their models and revisions, configure them for DHCP and PXE, learn and upgrade their firmware versions, and discover their capabilities and the resources are installed.

Data Storage hardware will have Templates that identify Data Services and Storage capabilities. A Storage system requires a set of Scripts to describe the way Storage functions are to be accomplished, such as provisioning, configuration, establishing DR relationships, and taking snapshots.

Network hardware will have Templates and Scripts to describe the way to discover topology and setup a virtualized network.

Platform and Service Templates

Templates are diverse and more complex for platforms and services. Like hardware, Platforms and Services may have Scripts to describe their installation and configuration.

Platform installation may be provided by the hardware Template on which the Platform runs in the form of a pre-built image. Pre-built images can be installers themselves, which automatically detect the hardware on which they are running and self-configure the Platform as part of the installation. In other cases, a Script can describe how to install and configure a Platform from another hardware state. After platform installation, monitoring agents will most likely need to be installed, and Scripts will describe how to accomplish this for each Platform available.

There likely will be several ways to build any service on any number of possible platforms. Services can be built from microservices or combined to create macroservices. Performance prediction, management, and metrics for services are also extensive. Services will install similar to applications but will have specific references in other Templates that rely on these services to describe their usage.

Templates for applications may be monolithic, rely on services, or even built out of services. Cloud Native applications that are supplied in pre-built Container or VM format are much easier to manage, and their Templates can reference shared Platform Scripts to support installation. Other applications will require complex Scripts to handle the various options for installation. Application KPIs are usually fixed and may be difficult to monitor without developer support or specialized agents made specifically to derive metrics from the application. Performance (including reliability, security) prediction must be described according to all dependencies, such as IO performance, which makes managing Application behavior not only the most important and meaningful CDI benefit, but also the most complex.

CDI success will begin if/only when the Template includes sufficient information to accomplish the tasks

10. Conclusions and Recommendations

CDI promises benefits that the IT customers desires. It is also invaluable to Cloud operators, MSPs, and vendors with as-a-service offerings. Before the Covid situation, CDI was seen as the correct path to solving the most severe problems in the industry. However, like many technologies, the idea precedes our capability to achieve it. Using the Gartner Hype Cycle as a model, CDI largely has fallen into the "Trough of Disillusionment" as those most enthusiastic for it have surmised the difficulties involved to achieve it. However, CDI remains the best proposal to reduce costs in IT environments, and the answer to IT management challenges will eventually be CDI automation or something very similar.

The path toward CDI automation must be taken in pieces. Some companies already may have set forth on the journey by limiting their CDI story to what they have or that is within their scope of business. Other companies, like HashiCorp, have attempted to jump into a part of the technology, which in their case is the declarative code aspect, and have been rather successful because the problem is so large. For companies that produce all of the IT stack, CDI can be used to expand the market and accelerate growth, as long as it adheres to Standards and promises to not create a lock in, even if its initial implementation only works with the company's own products.

An effective strategy will be to create products that leverage the technologies that will be needed along the CDI development journey. Leveraging the ONAP KPI monitors to create powerful CMDB with real-time dashboards and visualizations of detailed operational behaviors will lead to Monitors that are ready for Optimizer use. A tool that uses AI to create actionable recommendations from requirements and the Monitors is a product on its own but creates technologies that the Optimizer will use to form declarative code for automation. A simple Composer that can recognize a current installation, develop a To-Be plan, and automate change management, even without an Intelligent Optimizer to maintain and manage SLAs and enforce requirements, can save planning and execution time for IT personnel, and reduce risk. CDI can be built up in parts, with each part being a product or a tool which enhances sales of other products and services.

Another effective starting point is to provide unified support for the ONAP effort. ONAP is critical to the core carrier business and completing the CDI portion of it helps retain or attract that market. As carriers continue to evolve into more than just connectivity providers, their relevance in IT grows. Carriers have more presence and opportunity than current Public Cloud providers did to deliver IT services when Public Cloud began, and carriers can easily eclipse the current market leaders over the next several years. ONAP, CDI, and cost-effective self-driving IT will level the field, making carriers and large enterprises the most relevant IT forces.

CDI cannot be accomplished by companies where internal silos prevent them from utilizing their diverse capabilities over the whole stack. IT is moving away from specialization by Storage, Network, and Server layers and toward specialization by business process, application, and deliverable. Infrastructure is sliced vertically, following the lessons learned from the Cloud. Form must follow function, and successful organizational structure and behaviors must match customer ideals and needs.

Leadership in CDI begins by establishing details of interfaces, Templates, CMDB, and other core information assets of CDI automation. These details will lead to Standards and the ability to have vendors provide the information needed to manage the ecosystem, or the leaders in CDI will use existing Standards to create the necessary Scripts and Templates.

Until CDI automation or something like it is complete, the winner of the data plane will be whoever owns the control plan. HPE's GreenLake, EMC's APEX, Public Cloud, and as-a-service providers are positioning themselves to lock-in customers by managing their infrastructure. CDI automation puts control back in the customer hands so they can have choice.

References

[1] "IDC Forecasts Composable/Disaggregated Infrastructure (CDI) Market to Reach $3.4 Billion in 2022." From idc.com, 2018-08-31.

[2] Data from IDC, with the comment that in the remaining almost 10% the investigation only could not prove the employee was responsible.