White paper
Cisco public

Declarative and Resilient IoT Architecture Powered by Kubernetes and Infrastructure as Code

Contents

1. Introduction 3

2. IOT Solution Scalability and Resilience Challenges 4

2.1. Current IoT Solution Architecture 4

2.2. Scalability and Resilience Challenges in IOT Solution Orchestration 5

2.2.1. Challenges in Infrastructure Orchestration 5

2.2.2. Challenges in Application Orchestration 6

3. Solution: Declarative and Resilient IoT Architecture Powered by K8s and IaC 6

3.1. Solution overview 6

3.1.1. IaC and K8s achieve a declarative architecture 6

3.1.1.1. IaC and its declarative feature 6

3.1.1.2. K8s and its declarative feature 8

3.1.2. High-level architecture 9

3.1.3. Declarative orchestration at DC and edge is the key 11

3.2. Solution details 11

3.2.1. Orchestration Engine – IoT Solution Management Center 11

3.2.1.1. Cloud/Data Center Side Infrastructure Orchestrator 11

3.2.1.2. Cloud/Data Center IoT Solution App Orchestrator 13

3.2.1.2.1. K8s for Application Orchestration 13

3.2.1.2.2. IaC Tools to Facilitate the Application Deployment 14

3.2.1.3. Edge Gateway Infrastructure Orchestrator 14

3.2.1.4. Edge Computing App Orchestrator 15

3.2.2. Edge Orchestration Trend Forecasting 15

3.2.2.1. Orchestrate ML Model at the Edge 15

3.2.2.1.1. Usage Scenarios for IoT ML Apps 15

3.2.2.1.2. ML Benefits for IoT Apps 15

3.2.2.1.3. ML at the Edge 16

3.2.2.1.4. Containerization and IaC in ML Lifecycle 16

3.2.2.2. Extend K8s to the Edge 17

3.2.3. IaC and K8s Address the Security Challenges in IoT Solutions 17

4. Summary 18

5. Authors 19

6. Reviewers 19

1. Introduction

A typical Internet of Things (IoT) solution is composed of several tiers of distributed components, including cloud and edge. Machine Learning (ML) is trending to be utilized in multiple places in the structure of the solution hierarchy. It is imperative that these complicated IoT solutions can have quick, easy, and error-free deployments, and it is also necessary that IoT solutions can scale with business size with good resilience in all the tiers of components.

Infrastructure as Code (IaC) enables provisioning and managing IT infrastructure using source code, rather than manual processes or standard documented procedures. Using IaC tools, we can already easily orchestrate cloud or data center infrastructure in an extremely reliable, repeatable, and error-free manner. There is an ongoing trend that IaC technologies will also be utilized in the edge of IoT solutions.

Containerization technologies are getting more and more popular and are being utilized within the edge layer as well. As the edge components have ever-increasing computing power, it's becoming increasingly realistic and practical that edge computing infrastructure can also be orchestrated as a unified platform using orchestration technologies such as Kubernetes (K8s).

Meanwhile, as ML is being extended from cloud to the edge, there are an increasing number of ML algorithms that can run on different types of edge devices. It is essential to figure out the proper ways to manage and orchestrate those ML models onto the edge devices so that scaled ML can help with the analytics at the edge of the IoT solution.

This white paper will bring all the separate elements together to illustrate a vision of how all the technologies of IoT, IaC, K8s, and ML will seamlessly work together to digitally transform our world, from the aspects of solution orchestration and solution operation. The IoT solutions envisioned can greatly leverage IaC and K8s across all solution tiers for seamless infrastructure platform orchestration and application orchestration, in which declarative descriptions can be used to orchestrate needed micro-services including ML model service on top of the unified infrastructure, so that eventually we can easily manage the entire lifecycle of the IoT solution of the appropriate scale in a declarative manner.

2. IOT Solution Scalability and Resilience Challenges

Before presenting the new architecture, it's important to first discuss the current IoT solution architecture and the challenges related to scalability and resilience.

2.1. Current IoT Solution Architecture

The prevalent IoT solution architecture encompasses both the cloud/DC tier and the edge tier, as depicted in Figure 1 below:

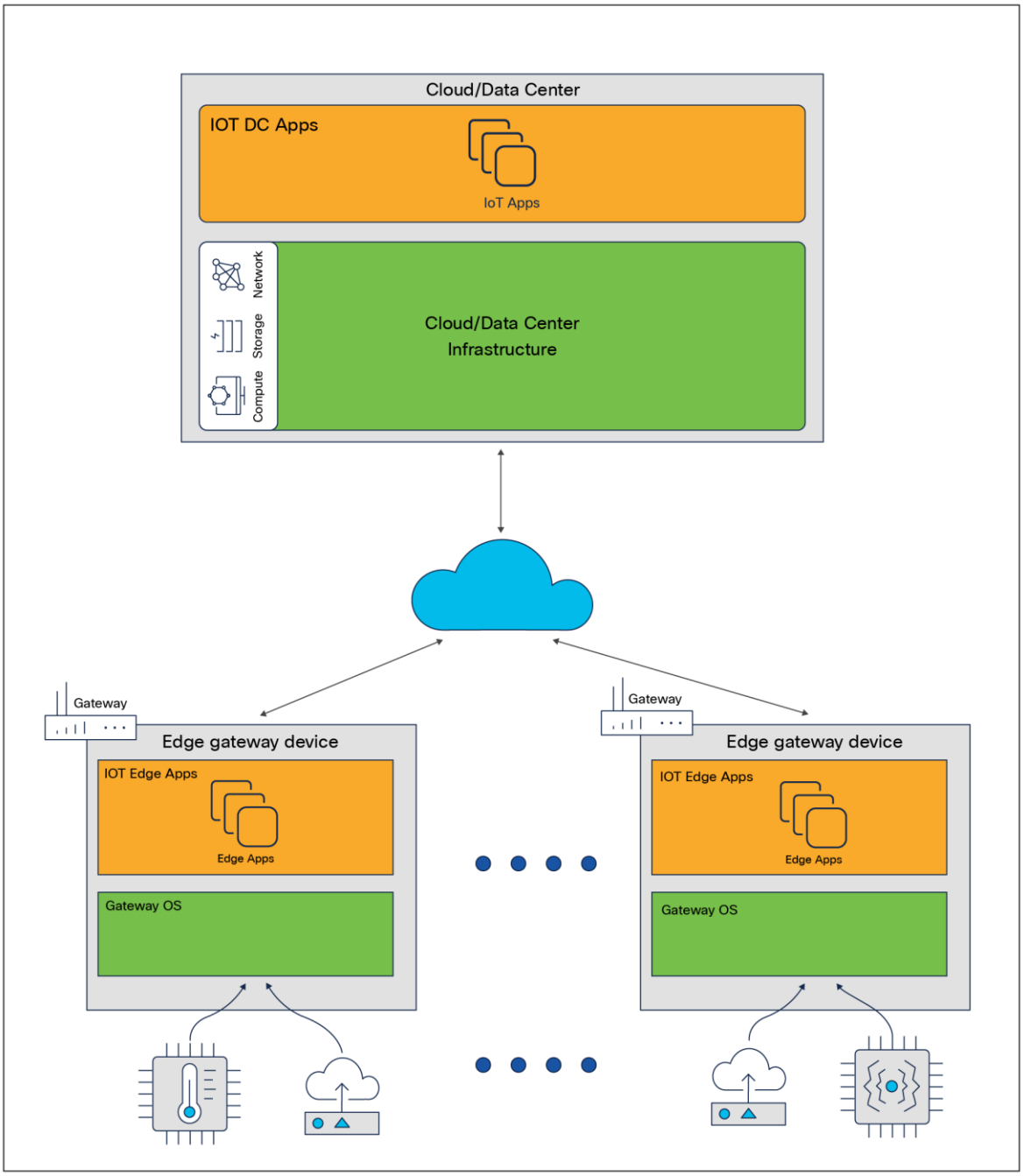

Figure 1.
Common IoT solution architecture

Cloud/DC infrastructure may take the form of bare metal, IaaS (Infrastructure as a Service), or PaaS (platform as a service). Bare metal refers to a physical server that is fully dedicated to a single tenant. IaaS is a type of cloud computing that provides users with virtualized computing resources over the internet. Hypervisors are the main component of IaaS. A hypervisor is a software layer that enables multiple Virtual Machines (VMs) to run on a single physical machine by abstracting the underlying hardware resources. IaaS has appeared before, so we should consider using the acronym. Please check PaaS and SaaS as well. Hypervisors are a critical component that enables customers to provision and manage their own virtual machines in the cloud/DC. Some examples of those hypervisors for IaaS in the cloud/DC include VMware ESXi, Microsoft Hyper-V, and Citrix Hypervisor. PaaS is a cloud computing service that provides users with a platform to develop, run, and manage their own applications without the need to manage the underlying infrastructure. Bare metal provides the most control but requires the most management, while PaaS provides the least control but requires the least management. IaaS falls somewhere in between, providing a balance of control and management.

The edge tier, on the other hand, refers to the computing infrastructure that is located closer to the source of data, such as IoT devices, sensors, and gateways. The edge tier is responsible for processing data in real time, and it can perform simple data filtering and analysis before sending it to the cloud for further processing. The edge layer is typically used for applications that require low latency, such as industrial automation and real-time monitoring of critical infrastructure.

2.2. Scalability and Resilience Challenges in IOT Solution Orchestration

2.2.1. Challenges in Infrastructure Orchestration

Infrastructure orchestration refers to the process of automating the deployment, configuration, and management of infrastructure resources, such as servers, containers, networks, and storage.

In any given IoT solution, managing the solution's lifecycle involves accounting for numerous dynamic factors, with scalability being paramount. Traditionally, once a solution is established, its scalability tends to be predefined, indicating the solution's inherent limitations in infrastructure expansion. Any scaling out typically necessitates manual intervention for infrastructure orchestration.

Infrastructure orchestration involves coordinating and managing multiple components, services, and tools, which can be complex and difficult to manage. This complexity can lead to errors, configuration drift, and other issues that can affect performance and reliability.

Without using the later-mentioned declarative approach, it will lead to some issues in real-world situations. For example, if the business owner needs to expand their business where more IoT devices will be needed to join the domain, they have to implement it manually and it’s difficult for them to ensure the eventual desired state will be reached. A lot of times, they need to ask help from professionals so it would be advantageous if they have a one-stop solution to take care of the infrastructure/application orchestration from the start as well as later when their business scales.

2.2.2. Challenges in Application Orchestration

Scalability is also a critical consideration in any application deployment, and orchestration is an essential part of ensuring that an application can scale effectively. Here are some challenges in scalability if we use the traditional way to orchestrate applications:

- **Limited automation:** Without a powerful orchestration platform, it can be challenging to automate scaling activities. This can lead to manual intervention, which can be time consuming and error-prone.
- **Complexity:** Building a scalable infrastructure without a powerful orchestration platform can be complex and challenging. This can be particularly true when dealing with large and complex applications.
- **Limited portability:** Without a widely adopted orchestration platform, it can be challenging to move applications and infrastructure between different environments.

With regard to the business, the following drawbacks may occur:

1. All the applications have to be deployed manually to each and every device, making the process repetitive, time consuming, and error-prone.
2. All the applications cannot be orchestrated from one single orchestrator, causing them to be not easily manageable or monitorable.
3. The applications have to be scaled manually, which is less efficient and consistent than the automated method.

3. Solution: Declarative and Resilient IoT Architecture Powered by K8s and IaC

3.1. Solution overview

3.1.1. IaC and K8s achieve a declarative architecture

Before we introduce the proposed architecture, we will first cover the two key technologies used in this paper (i.e., IaC and K8s).

3.1.1.1. IaC and its declarative feature

IaC is an approach of infrastructure management that involves defining infrastructure using code. The goal of IaC is to automate the provisioning, configuration, and management of infrastructure resources, such as servers, networks, storage, and applications, in a repeatable and scalable manner.

With IaC, infrastructure is treated as code, just like any other software, and version controlled using tools like Git. This allows teams to collaborate on infrastructure development and make changes more easily, with less risk of errors or inconsistencies. IaC also enables organizations to deploy infrastructure more rapidly and reliably and to scale resources up or down as needed. Popular tools for implementing IaC include Terraform and Ansible.

The following table shows the differences between the IaC approach and manual approach in terms of infrastructure provisioning.

Table 1. Comparison between IaC and the manual approach

Criteria	Infrastructure as Code	Manual Approach
Speed and Efficiency	Can provision infrastructure quickly and efficiently using code templates and automation.	Can be slower due to manual processes, especially for large-scale deployments.
Consistency	Ensures consistency across all environments, preventing configuration drift and reducing errors.	Can be error-prone due to manual processes, leading to configuration drift and inconsistent environments.
Version Control	Enables version control and change tracking for infrastructure, making it easier to roll back changes and maintain a history of modifications.	Does not have inherent version control, making it harder to track changes and roll back modifications.
Scalability	Can scale infrastructure quickly and easily by changing the code templates.	Can be more difficult to scale manually, especially for large-scale deployments.
Collaboration	Enables collaboration and sharing of infrastructure code templates among team members, improving efficiency and productivity.	Can be challenging to collaborate and share infrastructure among team members, leading to silos and reduced productivity.
Reliability	Can improve reliability by automating the provisioning process and reducing the risk of human error.	Can be less reliable due to the potential for human error during the manual provisioning process.
Cost	Can potentially reduce costs by automating infrastructure provisioning and reducing the need for manual processes.	Can be more expensive due to the need for manual processes and potential for errors.

A feature of IaC is that it can enable defining and managing IT infrastructure in a declarative way. A declarative approach involves defining the desired state of the infrastructure in a configuration file, rather than an imperative approach where we specify the sequence of steps required to create it.

You can specify the desired state of your infrastructure in a configuration file, and then use an IaC tool like Terraform or Ansible to create or update the actual infrastructure based on that configuration. The IaC tool reads the configuration file and compares it to the current state of the infrastructure. If there are differences, the tool makes the necessary changes to bring the infrastructure into the desired state.

The following table shows the comparison between the declarative approach and the imperative approach in IaC.

Table 2. Comparison between declarative approach and imperative approach

Criteria	Declarative Approach	Imperative Approach
Description	Describes the desired state of infrastructure	Describes the steps needed to create infrastructure
Configuration	Uses a configuration file (e.g., YAML, JSON) to specify the desired state	Focuses on how to achieve the desired state of infrastructure
Automation	Highly automated	Less automated
Idempotence	Highly idempotent	Less idempotent
Control	High level of control	Less control
Simplicity	Easy to understand and maintain	Can be complex and harder to maintain
Scalability	Easily scalable	Scalability can be more challenging
Error-proneness	Less error-prone	More error-prone
Reusability	Highly reusable	Less reusable

Among all the criteria, since idempotency is a strong requirement for IoT solutions, the declarative approach is preferred over imperative. Idempotency is a property or characteristic of a system, operation, or function where applying the operation multiple times produces the same result as applying it once. In other words, if you perform an idempotent operation multiple times, the outcome remains unchanged after the first execution.

Imagine attempting to establish three ML containers using an imperative method, but encountering an issue that prevents success. Once the issue is addressed and you proceed to create the containers anew, neglecting to remove the initial, unsuccessful containers could result in a total of six containers instead of the intended three.

Overall, the declarative approach is a key concept in IaC and can help one create more consistent, reliable, and scalable infrastructure, which is very helpful in orchestrating the infrastructure of an IoT solution spanning across its whole lifecycle.

3.1.1.2. K8s and its declarative feature

K8s is a powerful open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications. High availability and resilience are key features of K8s, which are crucial characteristics that we need while choosing a container orchestration framework. We envision that gradually K8s will become the mainstream option for service orchestration in the IoT solution stack and play a crucial role for scalable and resilient architecture.

- **Scalability:** K8s makes it easy to scale containerized applications by automatically creating and scaling replicas of containers based on application demand.
- **High availability:** K8s provides high availability for applications by automatically detecting and replacing failed containers or nodes.
- **Self-healing:** K8s provides self-healing capabilities by automatically restarting failed containers, rescheduling containers on different nodes, and replacing nodes if necessary.
- **Extensibility:** K8s is highly extensible and provides a rich set of APIs and plugins for integrating with third-party tools and services.

Overall, K8s offers a robust set of features and capabilities that help ensure that IoT applications are highly available and resilient to failures.

Declarative orchestration is a key feature of K8s and allows you to describe the desired state of your application and infrastructure, rather than the specific steps required to achieve that state. Here's how you can achieve declarative orchestration using K8s:

- **Define the desired state:** Start by defining the desired state of your application and infrastructure in a declarative way. This could include the number of replicas, the desired pod configuration, and any other relevant configuration parameters.
- **Create manifests:** Use K8s manifests to describe the desired state of your application and infrastructure. These manifests can be written in YAML or JSON and define the desired state of your resources, such as deployments, services, and pods.
- **Apply the manifests:** Use the "kubectl apply" command to apply your manifests to your K8s cluster. This will create, update, or delete resources as required to achieve the desired state.
- **Monitor and manage the state:** Once your resources are created, K8s will monitor the state of your application and infrastructure and make any necessary changes to maintain the desired state. You can use the K8s API to query the state of your resources and manage them as needed.
- **Use version control:** Store your K8s manifests in a version control system, such as Git, to track changes over time and enable rollbacks if necessary.

Declarative orchestration using K8s is a powerful approach that allows you to manage your application and infrastructure at a higher level of abstraction. By defining the desired state of your resources, you can reduce the complexity of managing your infrastructure and ensure that your application remains in a consistent and reliable state.

3.1.2. High-level architecture

Deploying and scaling an IoT solution can be challenging, but we will be introducing a declarative and resilient IoT solution powered by K8s and IaC that can help ensure quick, easy, and error-free deployments while also allowing for scalability and resilience across all tiers of the solution.

With the solution, the user can declare the business requirement in a declarative fashion, where non-technical users edit scaling factors on the web GUI and experienced users can use some domain-specific language to define it. While the infrastructure or application in the cloud or datacenter is scaled, the edge infrastructure or application will be scaled proportionately.

Figure 2 below illustrates the high-level architecture of the solution.

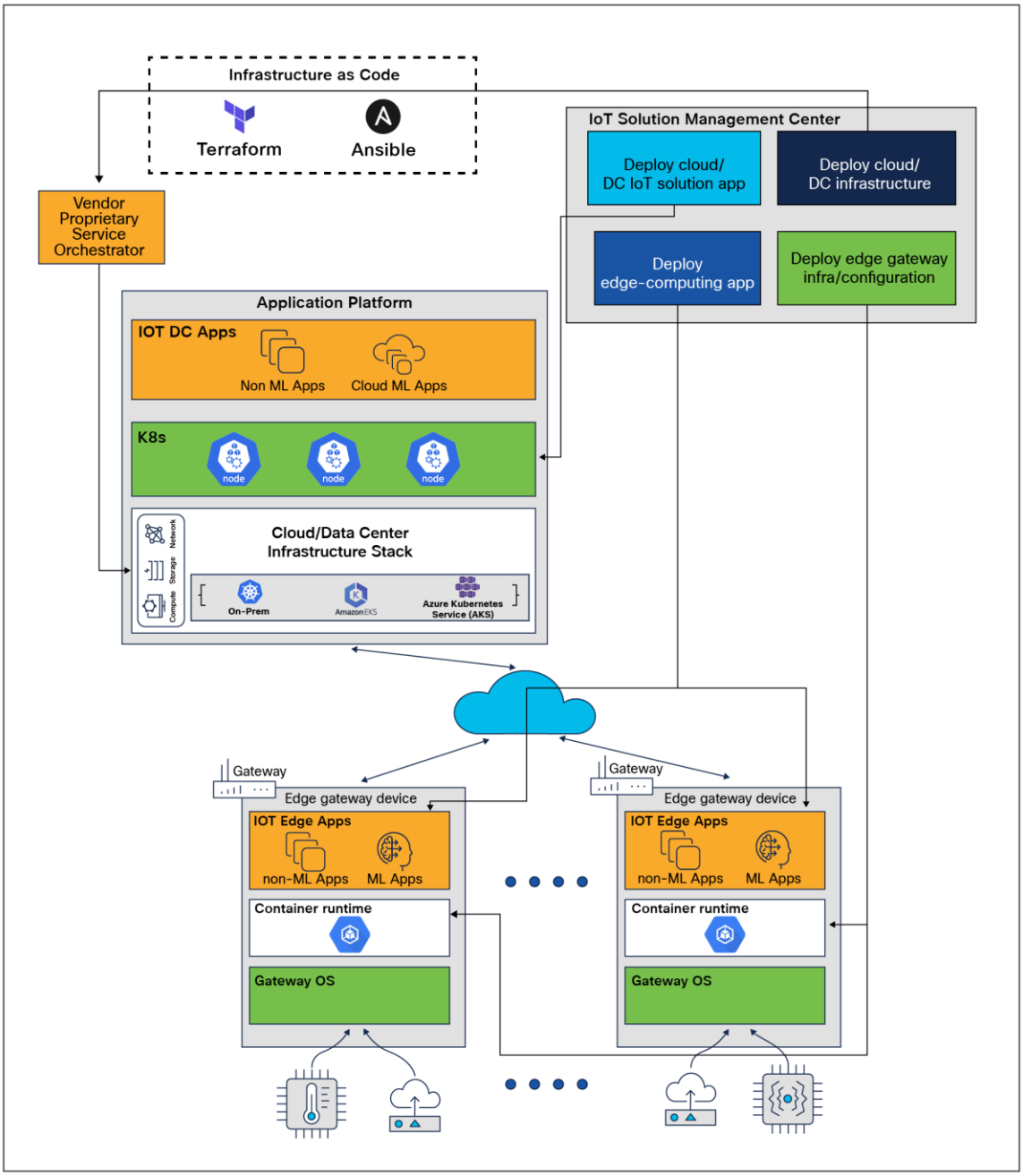

Figure 2.
Architectural design of the solution

This solution introduces an IoT Solution Management Center to facilitate the infrastructure provisioning of the cloud/DC layer and the edge layer. This can help reduce the risk of human errors or downtime during deployments and can help ensure that the solution is resilient in response to business change.

The solution also introduces the Application Platform, which refers to the centralized computing infrastructure that is located in data centers/cloud and provides data storage, processing, and analysis capabilities over the network. In an IoT deployment, the application platform is responsible for collecting data from the edge devices, storing it, and performing complex analytics on it to derive meaningful insights.

The solution also leverages containerization technologies such as Docker or K8s to package and deploy applications and services on the cloud/DC tier, which can help ensure that deployments are consistent and repeatable and can simplify the process of scaling up or down as needed. Since K8s functions as the application orchestration engine, the consolidated cloud/DC layer can now be referred to as the Application Platform.

ML is also becoming an increasingly important component in these solutions. ML can be used to extract insights from the vast amount of data generated by IoT devices, improving their functionality and efficiency and enabling new applications and use cases.

By leveraging these strategies, IoT solutions can be deployed quickly, easily, and with a high degree of reliability and resilience across all layers of the solution.

3.1.3. Declarative orchestration at DC and edge is the key

A declarative approach in infrastructure and application provisioning provides greater consistency, automation, scalability, reusability, version control, and ease of understanding, which leads to more reliable and efficient infrastructure and application deployment.

IaC is leveraged in two areas in the solution. One is to use the web UI or IaC code template to define the business solution to be deployed. The other is to deploy all the components/application with a declarative approach.

With declarative orchestration, it brings the following benefits:

1. **Provisioning and expanding infrastructure made easy:** Whenever the user wants to initially provision or expand the infrastructure he or she just needs to define the template for it and the system will ensure the desired state will be achieved based on what is defined in the template, making it easy and repeatable.
2. **Facilitating the deployment of the underlying software components:** Under the hood, the infrastructure of the solution is deployed/expanded with IaC so that the existing large library of IaC can be utilized to deploy K8s clusters and Docker runtime environment, which is robust and less prone to human errors and allows a faster time to market.
3. **Orchestrating applications in a declarative manner:** The application orchestration also leverages declarative orchestration to orchestrate the workloads so that it's easy to ensure the targeted deployment is implemented successfully.

3.2. Solution details

3.2.1. Orchestration Engine – IoT Solution Management Center

The IoT Cloud Management Center provides cloud-based services that can be used to manage IoT cloud/DC/edge components and the data they generate. This platform includes the following four orchestrators.

3.2.1.1. Cloud/Data Center Side Infrastructure Orchestrator

IaC can handle bare metal, IaaS, and PaaS in terms of infrastructure provisioning.

In the case of bare metal, IaC tools such as Ansible can be leveraged to deploy the essential software and K8s clusters on the bare metal machine.

For example, the following are the general steps to use Ansible to deploy a K8s cluster on bare metal machines.

1. **Set up your inventory:** Create an inventory file that defines the bare metal machines you want to use for your K8s cluster.
2. **Install Ansible:** Ensure that you have Ansible installed on your control machine from which you will run the deployment.
3. **Create Ansible playbooks:** Ansible playbooks are YAML files that define the tasks and configurations for your deployment. Create the necessary playbooks to install and configure the K8s cluster components on the bare metal machines.
    - **Playbook 1:** Prepare the target machines by installing required dependencies and configuring network settings.
    - **Playbook 2:** Install and configure the K8s control plane components (e.g., etcd, kube-apiserver, kube-controller-manager, kube-scheduler).
    - **Playbook 3:** Install and configure the K8s worker node components (e.g., kubelet, kube-proxy).
4. **Define Ansible roles:** Organize your playbooks into reusable roles. Roles allow you to group related tasks, making your playbooks more modular and maintainable.
5. **Configure variables:** Create variable files to define the necessary configuration parameters for your K8s cluster, such as network settings, authentication details, and cluster-specific options. Use these variables in your playbooks and roles.
6. **Write task templates:** In your roles, define tasks using Jinja2 templates to generate dynamic configuration files specific to each target machine. For example, you can use templates to generate the K8s configuration files (kubeconfig) for the control plane and worker nodes.
7. **Run the Ansible playbooks:** Execute your Ansible playbooks using the ansible-playbook command. Specify the inventory file, playbooks, and any required variables. Ansible will connect to the bare metal machines and execute the defined tasks, installing and configuring the K8s cluster components.

As opposed to bare metal deployment, in the case of IaaS, there is an extra step of VM provisioning where IaC tools such as Terraform can provision VMs on the hypervisor such as KVM, Hyper-V, or ESXi.

For example, to use Terraform to provision VMs on the ESXi hypervisor, you can follow these general steps:

1. **Install Terraform:** Ensure that you have Terraform installed on your local machine from which you will run the provisioning.
2. **Create a Terraform configuration:** Create a new file with a .tf extension, which will contain the configuration for provisioning VMs on ESXi.
3. **Configure the vSphere provider:** In your Terraform configuration file, specify the vSphere provider and configure the necessary settings to connect to your ESXi hypervisor. This includes providing the vSphere server address, username, password, and optionally the data center and cluster where the VMs will be created.
4. **Define the VM resources:** Define the VM resources you want to provision using Terraform. This includes specifying details like the VM name, guest operating system, resource allocation (CPU, memory), network configuration, and any additional customization required.
5. **Initialize and validate Terraform:** Run the “terraform init” command in your project directory to initialize the Terraform environment. This command will download the necessary provider plugins

based on your configuration. Additionally, you can run the “terraform validate” command to validate the syntax and structure of your Terraform configuration.

6. **Plan and apply the changes:** Run the “terraform plan” command to see a preview of the changes that Terraform will make. This step is optional but recommended to review the proposed infrastructure changes. If the plan looks satisfactory, execute the “terraform apply” command to start the provisioning process. Terraform will create the VMs on the ESXi hypervisor based on your defined configuration.

PaaS on the other hand will make things much simpler by only requiring the workloads to be deployed on the managed K8s cluster. For example, to deploy an AKS cluster on Azure using Azure Resource Manager (ARM) and IaC, the steps below can be followed:

1. **Create an Azure Resource Group:** First, create a resource group in Azure that will contain all the resources for your AKS cluster.
2. **Create an ARM template:** Use an ARM template to describe the infrastructure required for your AKS cluster. This template will define the AKS cluster's nodes, virtual networks, and other resources required.
3. **Create an Azure Container Registry:** You will need an Azure Container Registry (ACR) to store your container images.
4. **Configure Azure DevOps:** Use Azure DevOps to automate your infrastructure deployment process.
5. **Create a build pipeline:** Set up a build pipeline in Azure DevOps to build and push your Docker images to the ACR.
6. **Create a release pipeline:** Create a release pipeline in Azure DevOps to deploy the ARM template and configure the AKS cluster.
7. **Deploy the AKS cluster:** Run the release pipeline to deploy the AKS cluster, and it should automatically configure the K8s nodes and set up networking and security.

3.2.1.2. Cloud/Data Center IoT Solution App Orchestrator

3.2.1.2.1. K8s for Application Orchestration

K8s facilitates application orchestration in several ways:

- **Deployment Management:** K8s enables users to manage and automate the deployment of containerized applications. Users can define the desired state of their application in a YAML file and K8s takes care of the rest.
- **Scaling:** K8s allows users to scale their application horizontally by adding or removing replicas of a container. K8s also supports vertical scaling by changing the resources allocated to each container.
- **Service Discovery and Load Balancing:** K8s provides built-in service discovery and load balancing capabilities. This enables users to easily expose their application to the outside world and ensure that traffic is distributed evenly across all available containers.
- **Self-healing:** K8s ensures the availability of applications by automatically detecting and recovering from failures. K8s constantly monitors the state of the application and takes corrective actions when necessary.

- **Configuration Management:** K8s enables users to manage the configuration of their application using ConfigMaps and Secrets. This makes it easy to manage application configuration across multiple environments.

3.2.1.2.2. IaC Tools to Facilitate the Application Deployment

There are quite a few popular tools that already have community support and can be leveraged. For example, Terraform, as a popular IaC tool, can suit the purpose quite well. To deploy a K8s application using Terraform, you typically follow these steps:

1. **Define your infrastructure/application requirements:** You need to define the resources required for your application such as containers, pods, services and deployments.
2. **Write the Terraform code:** You write Terraform code that describes the resources you want to create.
3. **Initialize the Terraform environment:** You run the "terraform init" command to initialize the Terraform environment and download any required provider plugins.
4. **Plan the infrastructure:** You run the "terraform plan" command to preview the changes that Terraform will make to your infrastructure.
5. **Create the K8s resources:** You run the "terraform apply" command to create the K8s resources defined in the Terraform code. Terraform will create the resources in the order specified in the code.
6. **Verify the application deployment:** After the K8s resources have been created, you can run automation tests against the infrastructure/resources to see if they are as expected.

Terraform provides a declarative language for defining infrastructure resources, which makes it easy to manage your IaC. This enables you to version control your infrastructure and maintain a consistent and reproducible environment for your K8s apps.

3.2.1.3. Edge Gateway Infrastructure Orchestrator

IoT Solution Management Center can use IaC to provision and configure edge infrastructure by doing the following:

- **Automate infrastructure deployment:** IoT cloud providers can use IaC tools to automate the deployment of edge infrastructure resources. This may involve using tools such as Terraform to deploy infrastructure resources automatically based on the configuration code that has been written. This may involve deploying container runtime such as LXC, Docker, etc.
- **Configure infrastructure resources:** IoT cloud providers can use IaC tools to define the edge infrastructure resources that need to be provisioned and configured. This may include pushing network configurations, security policies, and other resources to the edge.
- **Monitor and manage infrastructure:** IoT cloud providers can use IaC tools to monitor and manage the edge infrastructure.

By using IaC, IoT cloud providers can streamline the process of provisioning and configuring edge infrastructure. IaC enables infrastructure to be deployed and managed in a consistent and repeatable manner, reducing the risk of configuration errors and increasing the speed of deployment. This approach can help organizations to create a scalable and reliable IoT edge infrastructure that can be easily managed and monitored from the cloud/DC.

3.2.1.4. Edge Computing App Orchestrator

By encapsulating each app into a container, we can achieve better isolation and security, more efficient resource utilization, and easier deployment and management. Containerization also allows for more flexibility in choosing the hardware and operating system for the edge devices, as the container image can be built once and deployed to different environments.

There are also lots of technologies in place already to do container orchestration in a declarative manner. For example, the popular tool Terraform also has a built-in provider for Docker that allows you to manage Docker images, containers and networks using Terraform configuration files. Terraform makes it easy to provision and manage Docker, allowing you to automate and standardize your container deployments and ensure consistency across different deployments.

Here are the general steps to provision IoT app Docker containers with Terraform:

- Install and configure the Docker provider
- Define your Docker container resources
- Apply your Terraform configuration
- Monitor your Docker containers

Besides open-source tools such as Terraform, proprietary technologies are also introduced by a lot of IoT solution vendors, which can achieve a similar purpose of app orchestration on the edge, the listing of which is not this paper's focus. But the overall trend is that IoT edge application's orchestration is going in the direction of declarative, in order to utilize all of declarative's advantages.

In summary, using IaC at the IoT edge can help organizations streamline their infrastructure management processes, improve agility, and reduce operational costs.

3.2.2. Edge Orchestration Trend Forecasting

3.2.2.1. Orchestrate ML Model at the Edge

3.2.2.1.1. Usage Scenarios for IoT ML Apps

There are many IoT applications that run on the DC/cloud. Here are a few examples:

- **Smart manufacturing:** Smart manufacturing systems gather data from sensors and machines on the factory floor and use cloud computing to analyze and optimize processes. This helps companies improve efficiency and reduce costs.
- **Smart transportation:** IoT sensors and cloud computing can help optimize traffic flow, monitor vehicle performance, and track shipments in real time.

3.2.2.1.2. ML Benefits for IoT Apps

IoT apps can leverage ML, which is used to extract insights from IoT data to improve the functionality and efficiency of IoT devices and systems. The benefits of using ML on IoT includes:

- **Predictive maintenance:** IoT devices can generate real-time data about their performance and condition. By analyzing this data using ML, we can identify patterns that indicate when a device is likely to fail or require maintenance. This allows us to perform maintenance proactively, reducing downtime and increasing efficiency.
- **Energy management:** ML can be used to analyze energy consumption patterns of IoT devices and identify ways to optimize energy usage. This can result in significant cost savings and improved sustainability.
- **Personalization:** IoT devices can generate data about user behavior and preferences. By analyzing this data using ML, personalized experiences that meet individual needs and preferences can be created.
- **Security:** IoT devices are vulnerable to cyberattacks, and ML can be used to detect and prevent such attacks. By analyzing network traffic and device behavior, ML algorithms can detect abnormal activity and take appropriate action.
- **Optimization:** ML algorithms can be used to optimize various aspects of IoT systems, such as routing, scheduling, and resource allocation. This can result in improved efficiency, reduced costs, and better overall performance.

3.2.2.1.3. ML at the Edge

Using ML at the IoT edge has become increasingly popular due to its ability to process data in real time, reduce latency, and improve overall system performance.

Edge devices typically have limited resources in terms of processing power, memory, and energy consumption, making it challenging to run ML algorithms directly on them. However, there are several techniques that can be used to overcome these challenges. For example, edge devices can use lightweight ML algorithms that have a smaller memory footprint and require less processing power. Another approach is to use techniques such as transfer learning, which enables edge devices to use pretrained models rather than training them from scratch.

3.2.2.1.4. Containerization and IaC in ML Lifecycle

Training and distributing ML models between IoT cloud apps and IoT edge apps involve a set of techniques and tools that allow for seamless communication and coordination between the different components of the system. Here are some steps to train and distribute ML models between IoT cloud apps and IoT edge apps:

1. **Data Collection and Preparation:** Collect and preprocess the data that will be used to train the ML model. This can be done on the edge device itself or on the cloud server, depending on the size and complexity of the data.
2. **Model Training:** Train the ML model on the collected data.
3. **Model Distribution:** Distribute the trained model to the edge device with the help of the IoT Solution Management Center.
4. **Model Deployment:** Deploy the distributed model on the edge device and integrate it with the IoT edge app. This can be done using a containerization technology like Docker or K8s. Running ML applications in Docker containers can provide a level of flexibility, consistency, and scalability that can be difficult to achieve with other approaches. With the declarative approach of IaC, ML containers can easily be deployed and scaled.
5. **Model Monitoring:** Monitor the performance of the deployed model and collect feedback data to improve its accuracy and efficiency. This can be done using tools like Prometheus or Grafana.

6. **Model Update:** Update the deployed model based on the feedback data collected. This involves repeating the training and optimization steps on the updated data.

To summarize, containerization and IaC can be used together to facilitate the lifecycle of the ML model.

3.2.2.2. Extend K8s to the Edge

In the near future, K8s can be extended from the cloud/DC to the edge, which will allow K8s clusters to manage edge computing workloads. In that scenario, two main components will be involved: the cloud-side component, which runs in the cloud-based K8s cluster, and the edge-side component, which runs on the edge devices. The cloud-side component provides the control plane for managing the edge-side component and for deploying and scaling edge applications. The edge-side component provides the data plane for running edge applications and for communicating with the cloud-side component.

Extending K8s to the edge helps organizations bring the power of cloud-native technology to the IoT edge, enabling fast decision-making, improved reliability and greater scalability while reducing costs and increasing efficiency.

IaC can be implemented one step further by employing linked scaling, which means if the cloud/DC infrastructure/app is scaled, the edge infrastructure/app will be scaled proportionately. Non-technical users can edit scaling factors on the web GUI so that both cloud/DC and edge will scale proportionately. For advanced users, they could directly edit a template file such as a Terraform YAML file to obtain more granular control.

3.2.3. IaC and K8s Address the Security Challenges in IoT Solutions

IaC and K8s can help address security challenges in IoT solutions by providing standardized and secure infrastructure management. Here's how IaC and K8s contribute to IoT security:

1. **Standardized and Consistent Infrastructure:** IaC allows you to define and provision infrastructure in a standardized and repeatable manner. This ensures that IoT devices, gateways and supporting infrastructure are consistently configured with secure settings. By eliminating manual configuration and enforcing security best practices through infrastructure code, IaC reduces the risk of misconfigurations and vulnerabilities.
2. **Automated Security Configuration:** IaC tools provide the capability to automate security configurations for IoT infrastructure components. You can include security-hardening scripts, firewall rules, access controls and encryption settings as part of the infrastructure code. By automating these security configurations, IaC helps ensure that security measures are consistently applied across all IoT devices and supporting systems.
3. **Secure Deployment and Updates:** K8s facilitates secure deployment and updates of IoT applications and services. By leveraging containerization, you can package IoT application components into isolated containers, ensuring separation and security. K8s provides features like rolling updates and version management, allowing you to deploy security patches and updates without downtime or disruption to the IoT solution.
4. **Isolation and Resource Allocation:** K8s allows you to isolate IoT applications and services in separate namespaces or clusters. This isolation enhances security by minimizing the impact of potential vulnerabilities or attacks. K8s also provides resource allocation and quota management capabilities, enabling you to enforce resource limits and prevent resource abuse or denial-of-service attacks.

4. Summary

In summary, the proposed declarative and resilient IoT architecture powered by K8s and IaC can provide several business benefits, including:

1. **Improved reliability and uptime:** K8s can help to ensure that IoT architectures are highly available and resilient to failures. This can improve reliability and reduce downtime, which can have a significant impact on business operations.
2. **Increased scalability:** K8s enables the scaling of applications and infrastructure components horizontally and vertically, which can help IoT architectures to handle large numbers of devices and data processing. This can enable businesses to scale their IoT applications as needed to meet changing demands.
3. **Faster time to market:** IaC provides automation for infrastructure deployment, configuration, and management. This can help to speed up the time to market for IoT applications and services, enabling businesses to bring new products and services to market faster.
4. **Simplified management:** K8s and IaC provide a simplified approach to managing IoT infrastructure and applications. This can reduce the complexity of managing IoT architectures, making it easier for businesses to manage their IoT deployments at scale.
5. **ML model lifecycle management:** K8s simplifies the distribution and lifecycle management of models, allowing data scientists and developers to focus on building and improving the models themselves rather than worrying about the underlying infrastructure. IaC provides a systematic and automated approach to manage the distribution and lifecycle of ML models. By leveraging K8s and IaC, organizations can benefit from efficient scaling, automated management, high availability, and consistent deployment of ML models.

5. Authors

Shawn Wu
Cisco Certified DevNet Specialist in DevOps, Cisco Customer Experience

Terence Zhang
Delivery Manager, Cisco Customer Experience

6. Reviewers

Hazim Dahir
Distinguished Engineer

Mike Reshetar
Principal Architect

John Garrett
Principal Architect

Pix Xu
Engineer

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV Amsterdam,
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at **https://www.cisco.com/go/offices**.

Printed in USA C11-3866044-00 11/23