Common Open Source Practices in Developing Cloud-Native Applications

Introduction

As many enterprises are trying to meet challenging business realities in the most cost-efficient way, they look for innovations and successes by becoming Cloud-Native. Open source technologies, as adapted to the Cloud-Native production environment, can be used to fulfill the demands created in the technology market. However, developing a good open source strategy that can build successful Cloud-Native applications is not an easy task.

This eBook aims to provide an general overview of open source technologies used in the era of Cloud-Native applications, which include containers, orchestration, and microservices. Further, it is dedicated to exploring some popular open source projects used in developing Cloud-Native applications and their characteristics.

Table of Contents

Why Cloud-Native?

Modern Application Requirements

Digital experiences have become an instrumental part of the many activities that we engage in today. We want applications or services to be available at any time, be perpetually upgraded with new savvy features, and provide personalized experiences. These increasingly demanding user expectations translate into business realities that empower enterprises to build applications that are always available, evolving with frequent releases, and easily scalable.

These needs have led to the emergence of Cloud-Native software and a revolutionized process to develop them. By migrating applications to the cloud environment, businesses can take advantage of the cloud infrastructures that provide elasticity, scalability, and on-demand access to a shared pool of computing resources.

Advantages of a Cloud-Native Approach

The trend for businesses today is to become Cloud-Native, meaning developers are writing code, testing and deploying it, and operating the applications all in the cloud. This is because Cloud-Native technologies enable enterprises to build applications that are more open, portable, scalable, and flexible than ever before. Moreover, the term Cloud-Native has become a buzzword that describes modern applications that utilize cloud, microservices, containers, and orchestration, often based on open source software.

Going Cloud-Native has become the current digital transformation trend, as companies that adopt these technologies are gaining competitive advantages over traditional organizations. For example, the Cloud-Native container technology can break down barriers in terms of environments and realize cross-domain collaboration between R&D and operation and maintenance, which improves delivery efficiency and productivity.

Cloud-Native technology can also shorten the time businesses take to update or release new versions, and therefore create more time for innovation. The feature of agile application development significantly improves delivery speed and reduces trial and error costs, allowing firms to cope with users' needs and enhance user experience effectively.

The Cloud-Native way utilizes the cloud's full potential and helps companies build flexible, reliable, and loosely coupled application systems. Each component that goes into building Cloud-Native applications involves different technologies and requires developers to make optimal decisions based on their technical and business situations.

A Cloud-Native, Open Source Approach

Open source has become the de-facto way of developing reliable and secure software. As we can see cloud adoption becomes increasingly mainstream, a Cloud-Native, open source approach will be the new normal of application development within many enterprises.

As expected, developers build most applications by leveraging open source frameworks or pulling in libraries that have been tested and proven in many production use cases. By using open source software, companies not only avoid building everything from scratch, but they also save time, money, and effort while creating more business innovation and value.

This is because the collaborative nature of open source enables a large group of developers to instill the best of their wisdom in the open source products that are reliable and available for the users to constantly contribute and improve. The open source community also provides a space for free exchange of knowledge and expertises, cultivating many excellent open source solutions for Cloud-Native transformation. Moreover, the open source software is at the forefront of creating innovations and supporting the Cloud-Native transformation, which will be further discussed next.

Cloud-Native Technological Innovations

With the rise of digital transformation, Cloud-Native technologies push the traditional, distributed approach to a new level by introducing innovations such as containers, orchestration, and microservices, which are the major components of the Cloud-Native system.

Cloud-Native technologies have the potential to create a truly open and flexible distributed system that can support enterprises to meet market requirements in runtime, scaling, etc.

Container Technology

Containers can be explained as lightweight virtual machines that wrap applications around and are infrastructure-independent. This means containers can work on and be moved between environments without affecting the applications they host.

Containers are also lightweight, standalone runtime environments that eliminate inconsistencies by wrapping software up in a minimal filesystem, containing only what it needs to run: code, runtime, system tools, and system libraries. Container-based applications can be deployed quickly and consistently whether in a private data center or the public cloud.

Additionally, containers can support the microservices approach and agile development processes. The microservices model divides application functionality into separate, self-contained services, which containers can help to achieve. As you deploy each service of a microservices application into a different container, each piece can be maintained, updated, swapped out, and modified independently of the others.

In this way, containers enable application portability, faster software delivery cycles, and efficient system resources.

Orchestration Technology

In practice, you will need to monitor and control containers. As your application grows, the number of containers you need to manage and control can become out of hand and complexity follows. That's when container or cloud orchestrators comes in. They are the tools that can help you monitor and manage container clustering and scheduling in the cloud more efficiently.

Cloud orchestrators function like a datacenter operating system administrator that manages cloud resources for containerized applications across different environments. They essentially take care of deploying your containers, scheduling, scaling, networking, and all the things related to sysadmins in an automated, programmable way. These benefits mean that developers no longer need to worry about where their applications run. Instead, they can focus on the business logic by simply deploying on orchestration platforms, such as Kubernetes, to take care of the rest.

Microservices Technology

Microservices describe an architecture in which applications are broken down into their smallest components or services. Microservices is a decomposition technique that aims to split complex systems into multiple independent, narrowly focused services. These services also have their isolated business logic and data store. The central idea behind a microservices architecture is that applications are simpler to build, modify, and maintain these smaller services.

These smaller pieces are loosely coupled, which provides the capability to run at scale with interchangeable pieces that snap into the application architecture. Additionally, the built-in space for extensibility in a microservices architecture has enabled developers to rapidly push out new features and updates or roll them back should they not work as planned.

Additionally, microservices enable developers from different teams to simultaneously work on modules, while enhancing the overall system's performance and uptime without the risk of affecting the other teams.

A successful Cloud-Native application is built on a well-designed paradigm of microservices. However, deciding what those microservices should be, where the boundaries are, and how the different services should interact is a difficult task.

As a common practice, developers will resort to a suitable microservices framework to avoid building everything from top to bottom, thus saving time and cost.

Putting the Technologies Together

Consider that the microservices approach is the recipe that developers follow to design their applications in a distributed manner according to the principles of scalability, flexibility, and resilience. The lightweight containers then wrap around the microservices-based applications and run in environments different from the development stages, thus accelerating application deployment cycles and increasing network agility.

Next, a container orchestrator, acting as the system administrator, directs and manages the collection of containers, ensuring the applications are run smoothly without interruption or downtime and can automatically scale up when needed. The orchestrator can then start the containers with the new version, wait until they become healthy, and then shut down the old ones if necessary.

Together, the three innovations discussed above establish the technological foundation for building Cloud-Native applications that are automatable, flexible, resilient and scalable.

Cloud-Native Tools Overview

Now that we understand the benefits that come with containers, container orchestration, and microservices, we will go into the best practices or options in these technological fields. In addition to introducing the options offered in the market, we will explore the characteristics of different projects that have made them optimal for Cloud-Native applications.

Containers

While Docker is not the only container platform out there, it indeed has risen to massive prominence over the past few years. What makes Docker so popular? As an open source project, Docker has an appeal in the technology market, which is seeing open source as the de facto way for software production. Docker enables developers to easily pack, ship, and run any applications as a lightweight, portable, self-sufficient container, and a collection of containers take up less space, handle more applications, and use fewer system resources.

Additionally, Docker has revolutionized application virtualization and been cited as a leader in the enterprise container platform category. As a software platform for building applications based on containers, Docker stands out from previous approaches because it has made containers easier and safer to deploy and use.

Docker also brings cloud-like flexibility to infrastructures capable of running containers. Docker's container image tools allow developers to build libraries of images, compose applications from multiple images, and launch those containers and applications on local or remote infrastructure. Further, Docker has helped popularize the technology and drive the trend of containerization in software development. Docker's key open source component, libcontainer is a product from its partnerships with other container powers, including Canonical, Google, Red Hat, and Parallels, bringing wider recognition and standardization to containers in the industry.

Orchestration

There are three major cloud container orchestration programs in the technology market, which are Docker Swarm, Kubernetes, and Mesosphere. While all three of them exist and can serve as alternatives for one another today, Kubernetes has become the most dominant cloud orchestration program since 2017.

Originally open sourced by Google and now maintained by the Cloud-Native Computing Foundation, Kubernetes serves as an infrastructure platform that manages all infrastructures on-premise and in the cloud and containerized workloads and services.

Kubernetes can also automate many traditional sysadmin tasks like upgrading servers, installing security patches, configuring networks, and running backups, making these less of a concern for developers in the Cloud-Native world. There are some features such as load balancing and autoscaling (support demand spikes) built into the Kubernetes core, and you can incorporate other features on the Kubernetes platform by using add-ons, extensions, and third-party tools that use the Kubernetes API.

The support for Kubernetes is becoming universal. Major platforms such as Docker and Amazon Web Services (AWS) have moved to support Kubernetes. Moreover, organizations often find that Kubernetes and Docker’s pairing is the best option when moving large containerized workloads into production.

However, Kubernetes typically comes with a steeper learning curve for most users. It isn’t a silver bullet for all things microservices and containers when building Cloud-Native applications.

Microservices

When building microservices platforms, organizations should choose a microservices framework that can optimize their business value.

Service governance and multiple language support are the major criteria to select an optimal microservices framework for enterprises. This is because the modular nature of microservices means that multiple teams would collaborate to create a single microservices application, so it is possible that they would use different programming languages in developing services of a single application.

Given the enormous size of Cloud-Native applications, multiple language support is indispensable in a microservice framework during the development stage. Service governance is also an important feature because the failure to implement proper governance mechanisms can result in an unmanageable and unstable architecture.

Microservices Framework

With this in mind, let’s consider a few popular options for microservices frameworks. Unlike Docker and Kubernetes which have won prominence in their respective areas, there are several methods to achieve microservices with different features to consider.

- **Single language framework with service governance**

  Spring Cloud is one such example and only supports Java.

- **Single language, no service governance framework, but with RPC (Remote Procedure Call)**

  Examples include gRPC and Thrift.

- **Service mesh: multi-language framework with service governance through the sidecar pattern**

  Examples include Istio.

- **Multi-language/Governance framework**

  TARS is an open source full-fledged microservices framework that fulfills both criteria.

The first two categories represented by gRPC and Spring Cloud have limitations when applied to actual scenarios and development environments. Because each programming language can have advantages in terms of performance and portability, introducing a multi-language microservices framework can optimize the technology stacks used by developers and increase scalability in addition to allowing cross-functional teams to collaborate. Thus, multilingual frameworks such as TARS and Istio are preferred especially in the Cloud-Native era.

Service Mesh

It is important to note that service mesh such as Istio can be viewed as a derivative of microservices architecture. Istio, as a dedicated infrastructure layer built into the application, focuses on managing all service-to-service communications between different parts. Istio uses "sidecar" proxies (Figure 1) that are deployed alongside each service through which all traffic is transparently routed.

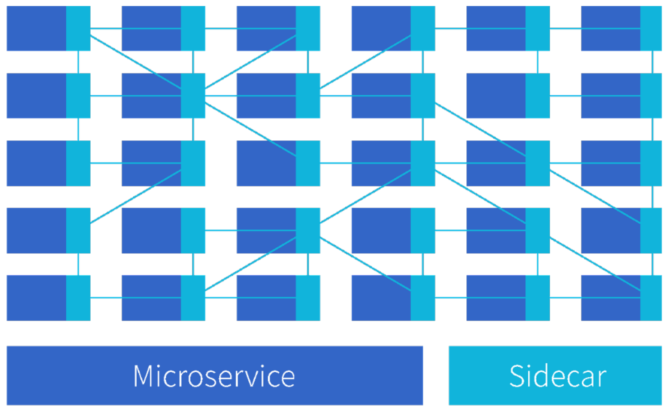

Figure 1: Service Mesh Pattern

Service mesh reduces the complexity associated with a microservice architecture and provides a lot of the functionalities such as load balancing, service discovery, traffic management and routing, etc.

A service mesh enables developers to have observability of the communications layers and to gain full control of all microservices communication logic. However, the infrastructure layer introduced by a service mesh increases the complexity of the architecture and its maintenance.

TARS: A Microservices Ecosystem

The drawbacks of gRPC, Spring Cloud, and Istio, as discussed previously, perhaps have cultivated the rise of TARS, which aims to address common problems when building microservices.

A mature, full-fledged enterprise solution for microservice maintenance, development and operation, TARS has been used by companies for more than 12 years, and it was open sourced in 2018. TARS is an open source cross-language RPC framework based on name service and Tars protocols, as well as an integrated administration platform, which implements hosting-service via a flexible schedule.

Additionally, TARS enables users to execute procedures remotely and supports multiple languages including C++, Java, Node.js, PHP and Python. It is also a rapid build system and automatic code generation that target agile development.

TARS' service governance not only involves the commonly needed functions such as service registration, discovery, load balance, and fault tolerance, but also provides special governance capabilities to face massive access and system overload. Figure 2 is an illustration of the TARS microservices ecosystem.

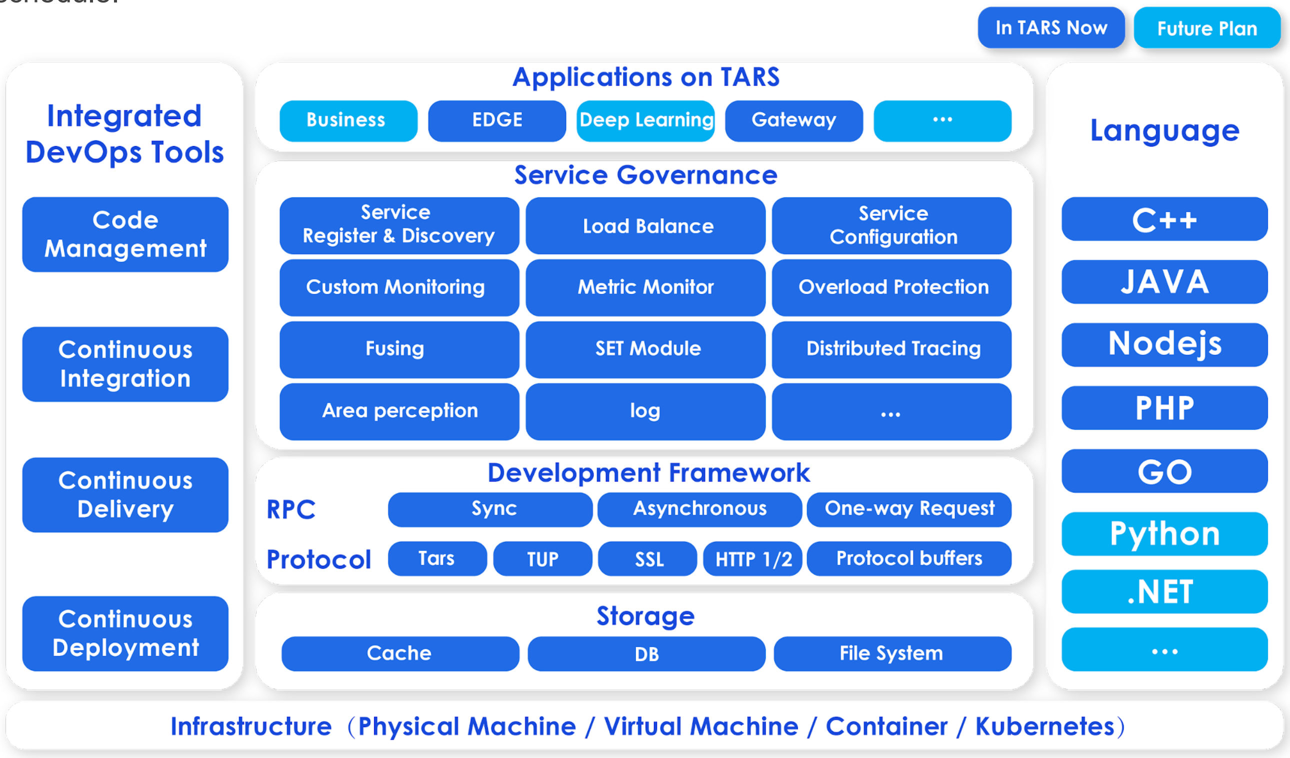

Figure 2: TARS Microservices Ecosystem

A Closer Look at Cloud-Native Practices

To demonstrate the Cloud-Native application of the aforementioned tools, we will also present a discussion about deploying microservices with Kubernetes.

Deploying Microservices with Kubernetes

Using a microservices framework that supports multiple programming languages and solves service governance problems would be the most ideal for organizations based on common best practices.

In the Cloud-Native context, Kubernetes have arguably become the gold standard for container-based DevOps + microservices + containers. The integration of Kubernetes has become a general direction for many microservice frameworks to improve their functions.

Hence, developers need to consider many factors before choosing an option to manage different functionalities related to microservices in Kubernetes, such as service-to-service communication, which is a challenge.

In order to introduce some essential features that developers should consider, we will illustrate the characteristics of using Istio with Kubernetes and K8STARS, a Kubernetes solution by TARS.

Istio

Service mesh is an approach that essentially takes the logic that is governing service-to-service communication out of individual services and abstracts it to a layer of infrastructure. Istio is composed of a control plane and data plane. The control plane contains the basic components ensuring that correct interaction between the other components. The data plane is responsible for all the communication between services in a microservice system using sidecar proxies.

Istio deploys a sidecar proxy that sits alongside a microservice and routes requests to and from other proxies.

As a whole, these proxies form a mesh network that can manage communication between microservices. With Istio, development and operations can be better equipped to conduct the migration from monolithic applications to Cloud-Native apps.

Istio is very well integrated into Kubernetes, both in standalone Kubernetes environments as well as managed Kubernetes offerings from major cloud providers. Istio uses an extended version of the envoy proxy and deploys it alongside each microservice pod in Kubernetes environments.

With Istio, you can enforce policies consistently across multiple protocols and runtimes with minimal application changes. When using Istio with Kubernetes (or infrastructure) network policies, the benefits include the ability to secure pod-to-pod or service-to-service communication at the network and application layers.

Istio Characteristics

- **Traffic management:** Istio provides traffic routing and rules configuration, allowing you to control the flow of traffic and API calls between services.

- **Security:** Istio has the underlying communication channel and authentication, authorization, and encryption of service communication at scale.

- **Observability:** Istio provides insights into your service mesh deployment with Istio's tracing, monitoring, and logging features. It lets you see how service activity impacts performance upstream and downstream. There are also custom dashboards that provide visibility into the performance of all your services.

TARS

Unlike Istio, TARS is a full-fledge, mature microservices development solution that supports agile development, multiple programming languages, and many more. It has released a solution to support Kubernetes, integrating both technologies to build an option, **K8STARS**, to develop Cloud-Native applications.

TARS Characteristics

- **Multiple programming languages**, including C++, Golang, Java, Node.js, PHP and Python, can be combined with various CI/CD tools.
- **Services can be deployed on different environments**, such as physical machines, virtual machines, containers, Kubernetes, and data can be stored in Cache, database or file system.
- **Supports multiple protocols**, such as self-developed TARS protocol and TUP protocol, as well as SSL, HTTP, PB, etc., commonly used in the industry. In addition, you can customize the protocol.
- **RPC**, synchronous, asynchronous and one-way methods can be used in RPC calls.
- **Supports many service management functions**, including but not limited to service registration/discovery, load balancing, custom monitoring, logging, overload protection, fuse mechanism, IDC SET, etc.
- **Supports a wide range of applications**, such as deep learning, edge computing and API gateways.

K8STARS

K8STARS is a native Kubernetes solution for TARS services. It retains the naming service, high-performance RPC and service management functions of TARS, and integrates the resource scheduling capabilities of Kubernetes, making TARS more Cloud-Native suitable. In the concept of microservices, using containers as infrastructure can achieve rapid deployment and rapid iteration.

K8STARS Characteristics

- **Built-in development capability of TARS.**
- **Automatic registration and configuration deletion of name service for TARS.**
- **Support smooth migration of existing TARS services to K8S and other container platforms.**
- **Non-intrusive design, no coupling with the operating environment.**

Because Kubernetes supports automatic scheduling, better version management can be achieved based on Docker images. TARS supports gateway traffic in the naming service part, which can quickly shield problems when a single point of failure is found, and configuration management is more suitable for use in a production environment.

Cloud-Native Innovations based on Open Source

Note that all of the common practices and tools discussed above have risen to prominence in Cloud-Native applications as open source projects, which are maintained by open source developers and advocates.

Although these development tools might not be the ultimate solution to everything, they are rapidly updating and upgrading through the open source approach.

It is the collective effort of the wider open source community of many extremely talented developers who exchange ideas and feedback on using these technologies that make open source software continuously meet market demands and build the most business value.

Conclusion: Cloud-Native Innovations from an Open Source Perspective

Open Source Collaboration Enables Innovations

The emergence of Cloud-Native applications has successfully helped enterprises cope with the ever-evolving user expectations. As the foundation for the Cloud-Native ecosystem, technological innovations and practices, such as **Docker**, **Kubernetes**, **Istio**, **TARS**, etc., have come a long way to become mature and popularized. Most of the popular technologies mentioned here are open source projects, and it is clear that the open source model of software production has become the standard of the industry.

Open source communities have provided a platform for enterprise-level technologies to thrive and improve via the collaboration of developers from different backgrounds, which can infuse new perspectives into the software and caters to different production demands.

Business Advantages of Open Source

Moreover, open source software displays several distinct business advantages, including cost reduction, speed to market, collaborative market advantages, code improvements, increased efficiency, and innovation jumpstarts. Thus, it is expected to see open source technologies as the driving force in Cloud-Native implementation and innovation.

Open source software has also proven to be the building block for many business successes. This is mostly due to the collaborative nature of open source projects. It is a group of dedicated developers in the global open source community that create credibility and confidence in open source software. Open source is where enterprises can find business competitive advantages in the technology market.

An Open Source, Cloud-Native Future

The development of Cloud-Native cannot happen without open source technology. As Cloud-Native migration becomes more prevalent, more businesses will be incorporating the open source software, and enterprises around the world will continue to place great value on the open source community.

We have explored some well-known Cloud-Native technologies and the open source tools available for Cloud-Native transformation. As you navigate through your Cloud-Native journey, we hope this eBook has helped you grasp Cloud-Native applications better and how the open source model can shape the future of technology.

References

What is Docker?
https://www.docker.com/why-docker

What is Kubernetes?
https://kubernetes.io/docs/home/

What is Istio?
https://istio.io/latest/docs/concepts/

What is TARS?
https://github.com/TarsCloud/TarsDocs_en

**What is K8STARS:
the Kubernetes native solution for TARS services?**
https://github.com/TarsCloud/K8STARS

**Why open source software
matters to your enterprise?**
https://www.linuxfoundation.org/tools/todo-group-why-open-source-matters-to-your-enterprise/

About the TARS Foundation

The TARS Foundation is a nonprofit, open source microservice foundation under the Linux Foundation umbrella to support the rapid growth of contributions and membership for a community focused on building an open microservices platform. It focuses on open source technology that helps businesses embrace microservices architecture as they innovate into new areas and scale their applications. It continues to work on addressing the problems that may occur in using microservices and wishes to accommodate a variety of bottom-up content to build a better microservice ecosystem.

Follow Us

https://tarscloud.org

https://github.com/TarsCloud

https://twitter.com/TarsCloud

Official Account: TarsCloud

tars@tarscloud.org

FREE
Microservices Training

Building Microservice Platforms with TARS

Our training course is available on edX **for FREE.**

Scan the QR code or click on the link and

Enroll Now!

https://www.edx.org/course/building-microservice-platforms-with-tars