RED HAT® OPENSHIFT

redhat.

WHITEPAPER

EXECUTIVE GUIDE TO SELECTING A CLOUD-NATIVE DEVELOPMENT PLATFORM

INTRODUCTION

When any organization starts planning for cloud-native applications, it is important to consider the entire time span: from selecting a development platform until an application is truly production-grade and ready for delivery in the cloud. It can be a long journey, with many decisions along the way that can help or hinder progress.

For example, at the beginning of a move to cloud-native development, it is easy for inefficiencies to occur if developers begin selecting tools and frameworks before they know where the application will be deployed. While enterprise developers want choice of runtimes, frameworks, and languages, organizations need standards that address the entire application life cycle in order to reduce operational costs, decrease risks, and meet compliance requirements. Organizations also want to avoid lock-in, whether it is to a single provider of cloud infrastructure or the latest architectural style.

In addition, given the steep learning curve in cloud development, considerations for building robust, scalable, and resilient applications cannot be left until late in the development cycle. A primary driver for moving to the cloud is elastic capacity for dynamic scaling, so resiliency cannot be an afterthought.

For an effective cloud-native application strategy, the entire picture needs to be considered, from development tools and cloud platform to deployment automation and operations. The benefit of a holistic approach to cloud-native development is that it lends itself to a guided path that eliminates unnecessary detours.

UNDERSTANDING CLOUD-NATIVE DEVELOPMENT

Cloud-native development is about responding to change with speed, resiliency, and agility. This response is achieved through more frequent deployments that significantly reduce the lead time to react to change.

Much of the discussion about cloud-native applications goes back to a manifesto called "The Twelve-Factor App," which is a set of principles gleaned from the experience of building and operating Software-as-a-Service (SaaS) applications. The goals are:

- Reducing the developer learning curve.
- Building applications that are well suited for deployment on cloud platforms.
- Maximizing agility through continuous deployment.
- Enabling applications to scale up without requiring significant changes.

Through working with customers, Red Hat has found the following key elements for developing applications that are cloud native:

- **A services-based architecture.** The style can be microservices or any modular, loosely coupled architectural model. Services represent a business activity that is reasonably self-contained and readily understood. The goal is to have services that can easily be updated or replaced. A well defined service is easier to test more thoroughly than a complete application.
- **Containers.** Linux® containers, using Docker images, are a common packaging model and a self-contained execution environment that provides portability as well as isolation. Containers enable the advanced automation that makes cloud platforms appealing.
- **DevOps automation.** A set of collaborative processes and practices aimed at unifying development and operations. The goals are to improve deployment frequency and deliver higher quality releases, resulting in faster time to market, less risk, and better user satisfaction. Continuous integration/continuous delivery (CI/CD) are very closely tied to improvements resulting from DevOps. Instrumentation and monitoring to understand performance, as well as ensure a quality end-user experience, is another key goal.
- **API-based communication.** Interprocess communication only occurs through application programming interfaces (APIs) using clean, contract-based interfaces over the network. This eliminates unintended coupling that restricts change and is a common source of outages. The importance of this becomes clearer in hybrid landscapes where applications no longer reside in the same data center.

UNDERSTANDING MICROSERVICES

Microservices architecture is a compelling architectural style for solving many software development challenges. Applications are broken into a loosely connected set of services that implement specific business functions. The primary goal is more rapid innovation based on the idea that an individual microservice is much easier to understand, improve, test, and deploy than a monolithic application. As the purpose and function of each microservice should be well defined, automated testing and continuous delivery become much more practical to implement.

REQUIREMENTS FOR CLOUD-NATIVE APPLICATIONS

Cloud-native architecture places a number of technical requirements on applications. These include:

- **Deployment needs to be automated.** To speed release frequency, there is a need to incorporate deployment automation so the process can occur by simply clicking a button for approval. An automated build and deployment pipeline is required along with the ability to quickly roll back to a previous release if necessary.
- **Configuration information must be separate from the application.** The configuration details needed for an application to run within a particular environment should be stored in the deployment environment. This allows the same application image to be used in all environments including development, testing, and production. Additionally, secure credential storage is needed as applications will need to securely communicate with data sources and other components over the network.
- **Application services must be located dynamically.** This is required for portability across environments, high availability, and dynamic scaling where load balancing is used.
- **Separate datastores are needed for persistent data.** Processes and containers must be stateless. Processes can crash and be restarted, possibly on different machines within a small number of seconds. Data, session information, and logs must all be kept in external datastores.

Some considerations when beginning cloud-native development are:

- How much experience does your organization have with building and deploying containerized applications to run on cloud platforms? Do developers understand what is required to manage persistent data and configuration separately from applications?
- How long will it take to build a development environment to begin cloud-native development? Will the development environment match the production cloud environment?
- How much time will it take to build and effectively integrate CI/CD into the development and deployment processes?

All of these things must be considered when selecting a cloud-native development platform.

MICROSERVICES CONSIDERATIONS

Microservices and cloud-native applications are closely related. In fact, microservices are difficult to implement successfully without a cloud platform, containers, and DevOps automation. A primary driver for microservices is the agility that is gained from using smaller application components that can be released more frequently with lower risk. However, what was once a single application could become a dozen microservices with independent release cycles. Here are some considerations when moving to microservices:

- How long will it take to provision new development, test, and production environments for each new microservice?
- Is there enough build and deployment automation to support the number of releases that will be required?
- How will applications discover where microservices are running? Is there the flexibility to relocate microservices as needed for availability or scalability?
- Can microservices be released without incurring downtime that will impact end users?
- How difficult will it be to diagnose problems? How difficult will it be to trace events across different services and possibly hybrid environments?
- Will problems with one service be well contained or will those problems cascade into multiple failures?
- Given the distributed nature of microservices, additional software infrastructure components are required to perform functions such as service discovery, client-side load balancing, persistent state management, distributed tracing, and resiliency.
- Will developers need to implement those services or will they find components off the internet?
- How much of a maintenance burden will there be if these components are now part of every deployed microservice? How will the components get updated to address bugs and security vulnerabilities?
- Are these commodity services that should be built into cloud runtimes or platforms?

For developers to start building microservices, one of the first questions that comes up is what sort of runtimes, frameworks, and languages are needed. Some considerations when answering this question are:

- Do developers need to learn new frameworks to get started?
- Can the organization's existing Java™ EE code and expertise be taken advantage of when moving to microservices?
- Can new development styles, such as reactive programming, be used? Does the platform offer choices needed to meet new demands, such as the event-driven workloads created by mobile and the Internet of Things (IoT)?
- In order to use the best tool for the job, is it reasonable to have different microservices implemented using different runtimes, frameworks, or languages? Will there be differences in the way they are configured, deployed, or secured?
- How long will it take before new microservices can be tested in an actual public cloud environment?

Microservices are very appealing. However, microservices and distributed application architectures can be difficult to master. It is important to select a platform that can abstract away complexity and simplify development.

MODERNIZING EXISTING MONOLITHIC APPLICATIONS

While it might be attractive to developers, it's not always practical or cost-effective to re-write existing monolithic applications into microservices. However, these applications still need to be maintained and have a backlog of improvements that need to be implemented. Reducing the backlog can be difficult due to lengthy release cycles. Re-hosting these applications to a cloud platform can make it easier to implement CI/CD, along with rolling releases, such as blue/green or canary deployments. This allows developers to deliver more frequent releases with less risk. The cloud-hosted application can become a "fast-moving monolith," enabling developers to address the backlog of improvements.

Re-hosting an application in the cloud can also be an effective first step toward implementing microservices. The flexibility of the cloud makes it easier to deploy new microservices in containers alongside the existing application. Developers can then implement microservices that migrate functionality out of the monolithic application.

Given the benefits of lifting and shifting existing applications to the cloud, here are some considerations when choosing a cloud-native development platform:

- Does the development platform have options for modernizing existing applications as well as new greenfield development?
- Can the cloud platform support applications that are not yet fully cloud native?
- What options exist for migrating existing Java EE applications?
- Does the development platform support speeding up the delivery of monolith style applications?

Application servers can be a viable runtime environment for cloud-native applications, but the role of the traditional application server changes when moving to a cloud native platform. Traditionally, application servers provided a runtime and deployment environment, as well as management services for a central domain running on a cluster of systems. However, many of the included operational tools, such as the administration console, are no longer necessary and are counterproductive. Cloud platforms are appealing because they include automated management capabilities that enable dynamic scaling and continuous delivery. Packaging a traditional application server with its management components in a container can prevent developers from taking advantage of those capabilities. Some things to consider:

- Does the platform include an option for a Java EE runtime that is integrated with the cloud platform's management capabilities?
- Are there tools to help migrate applications that are running on a legacy application server to a modern, cloud-based Java EE environment?
- For applications that do not use all of the capabilities of Java EE, such as web applications, are there alternatives that might be a better fit for the cloud?

With the right platform and tools, organizations can gain more value from existing applications while also migrating to a cloud-native and microservices model.

SUPPORTING ON PREMISES AND MULTIPLE CLOUDS

Most IT organizations prefer not to be limited to a single provider of public cloud infrastructure. Additionally, many organizations believe that even years from now, at least half of their applications will still be running on-site. Since most organizations have these two requirements, some considerations are:

- Will applications need to change in order to use cloud infrastructure from different providers? What about deployment, operations, and monitoring? Will those be different on each cloud platform?
- How can the differences between public cloud infrastructure and on-site systems be minimized? Is the cloud platform used in public clouds also available to run on-site or on the organization's preferred Infrastructure-as-a-Service (IaaS)?

THE LARGER PICTURE

When evaluating a cloud-native development platform, it is also worth considering how the choice will impact other areas:

- Are development tools available that work with the cloud development and deployment platform out of the box? If not, how much time will developers spend trying to configure and integrate their tools?
- Does the platform provide developers a prescriptive or guided approach to increase productivity?
- Are a suite of cloud-based middleware services for messaging, data storage, and business process or rules management available, and ready to run on the cloud platform?
- Are pre-built third-party containers available to integrate with applications?
- Are training and consulting services available? Do the available in-house resources have experience with application modernization?

THE IMPORTANCE OF SELECTING THE RIGHT CLOUD DEVELOPMENT PLATFORM

It is natural for developers who are starting with cloud development to solve one problem at a time by following tutorials, and selecting software components from the Internet. First they need to learn how to assemble enough of a runtime environment to build application code into a container. Then they need to decide what other software components are necessary to build complete applications. This process can be slow and have a number of drawbacks:

- The learning curve and the amount of integration required can result in a loss of developer productivity that is difficult to explain to the business. Any new developers hired will need to learn the stack that has been created in-house.
- Any selected software components will need to be maintained to ensure they are free of known bugs and security vulnerabilities. For open source components, there is also the question of whether they are appropriate for enterprise use.
- Many organizations report that developers can wind up writing code for functions like configuration and deployment that do not provide any direct business value. Worse, many of these functions are tied into the choice of a deployment platform and are commodity services that should be part of the deployment platform.

Perhaps the largest problem with assembling a collection of components is that it does not lend itself to opportunities that reduce the overall learning curve for development and operations. The alternative is to select a cloud development platform that is designed to handle all phases, from selecting a runtime to beginning development, all the way through to production deployment.

Moving to cloud-native applications is a process that does not occur overnight. For many organizations, it is a process that improves with experience. Given all that cloud-native development entails, it is clear that an end-to-end approach, covering development to deployment, is more likely to be successful. Red Hat offers such a platform with Red Hat® OpenShift Application Runtimes and Red Hat OpenShift.

RED HAT OPENSHIFT APPLICATION RUNTIMES AND RED HAT OPENSHIFT

Red Hat OpenShift Application Runtimes is designed to simplify cloud-native application development. A curated set of integrated runtimes and frameworks provide a guided path with a prescriptive experience that can be used to jump-start development. A completely streamlined development and deployment platform is possible because OpenShift Application Runtimes is based on, and optimized for, Red Hat OpenShift, a container application platform that is designed for hybrid clouds.

Red Hat OpenShift provides a self-service platform for developers and operations to build and run containerized applications. Through OpenShift, an environment for a new microservice or application can be provisioned in minutes. Significantly reduced deployment cycles with much lower risk are possible with OpenShift. Powerful, automated CI/CD build and deployment pipelines are available with a few clicks. Together, Red Hat OpenShift Application Runtimes and Red Hat OpenShift, with Red Hat development tools and Red Hat Consulting, can help organizations move to cloud-native applications with less time and risk.

RED HAT® OPENSHIFT

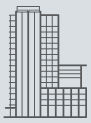
redhat.

WHITEPAPER Executive guide to selecting a cloud-native development platform

The runtimes in OpenShift Application Runtimes are selected to give developers the right tool for the job. For microservices development, developers have the choice of using Java EE to take advantage of existing expertise, the newer Java microprofile standard that has evolved to meet the needs of microservices, or an event-driven framework to build reactive microservices that scale for high-concurrency, low-latency workloads. A Node.js runtime is provided for JavaScript backend services that are frequently used with mobile and web applications. For migrating existing applications, a runtime based on Red Hat JBoss® Enterprise Application Platform, a Java EE 7 certified application server based on a modern, modular, cloud-ready architecture, is provided. All of the runtimes are tested and verified by Red Hat.

To get developers started quickly, Launch, a free, cloud-based, SaaS tool running on Red Hat OpenShift Online, is available through the Red Hat Developers website. After choosing a sample application and a runtime, developers get a complete code base that is ready to build and run in the cloud on OpenShift Online. As an educational tool, the same sample applications are available for all runtimes, which enables developers to easily compare the merits of the available architectural styles.

To learn more see: https://www.redhat.com/en/technologies/cloud-computing/openshift/application-runtimes



ABOUT RED HAT

Red Hat is the world's leading provider of open source software solutions, using a community-powered approach to provide reliable and high-performing cloud, Linux, middleware, storage, and virtualization technologies. Red Hat also offers award-winning support, training, and consulting services. As a connective hub in a global network of enterprises, partners, and open source communities, Red Hat helps create relevant, innovative technologies that liberate resources for growth and prepare customers for the future of IT.

facebook.com/redhatinc
@redhatnews
linkedin.com/company/red-hat

redhat.com
F10657_V1_0118_KVM

NORTH AMERICA
1 888 REDHAT1

EUROPE, MIDDLE EAST, AND AFRICA
00800 7334 2835
europe@redhat.com

ASIA PACIFIC
+65 6490 4200
apac@redhat.com

LATIN AMERICA
+54 11 4329 7300
info-latam@redhat.com

Copyright © 2018 Red Hat, Inc. Red Hat, Red Hat Enterprise Linux, the Shadowman logo, and JBoss are trademarks of Red Hat, Inc., registered in the U.S. and other countries. Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries. Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle America, Inc. in the U.S. and other countries.