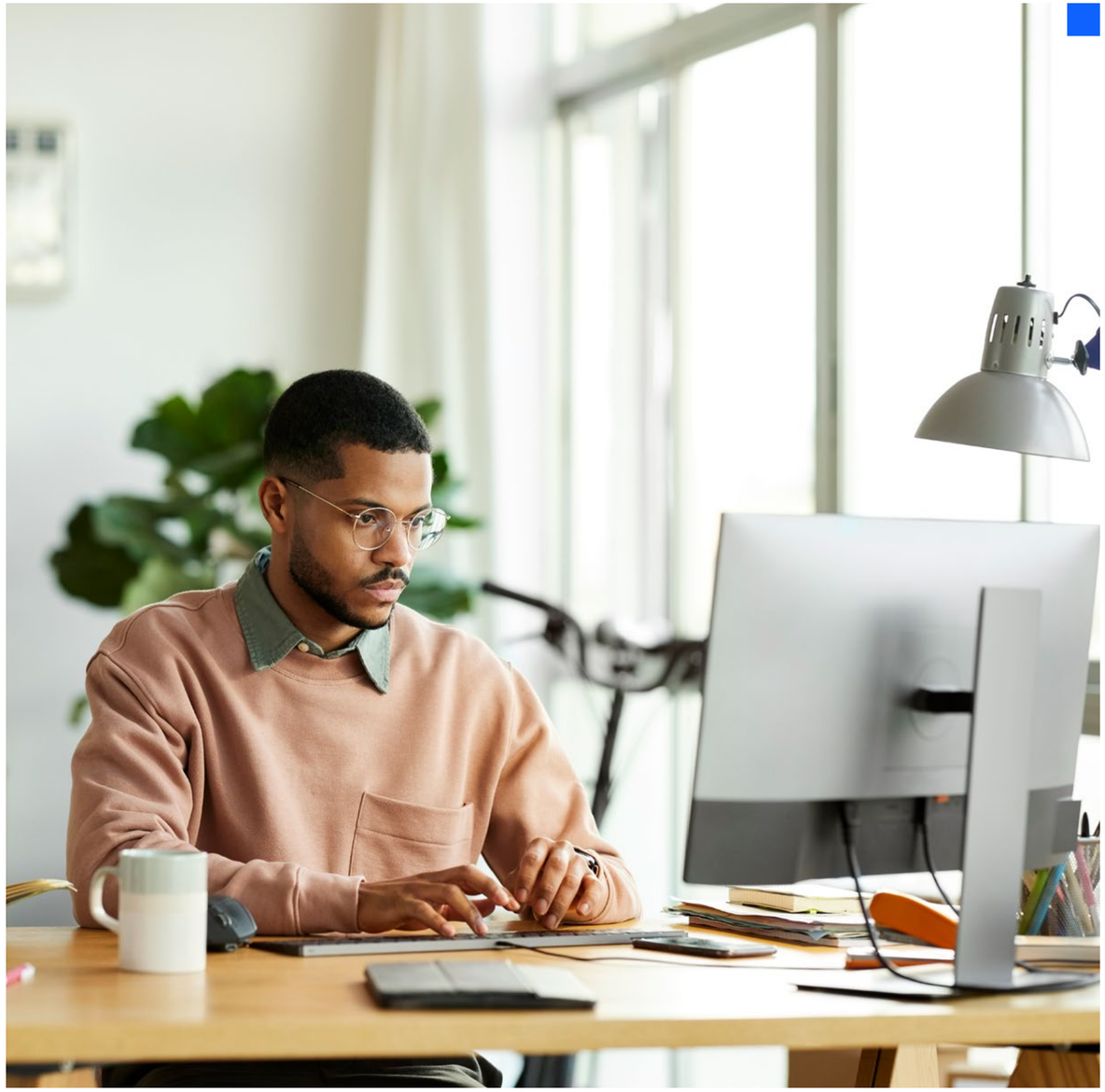
IBM Instana

Application performance management for microservice applications on Kubernetes

How to improve business-critical app performance in a Kubernetes environment



Contents

01 →
Introduction

02 →
Kubernetes basics—
or the ABCs of K8s

03 →
Kubernetes application
monitoring challenges

04 →
Seeing through
the complexity

05 →
Kubernetes monitoring
tools and strategies

06 →
Conclusion

07 →
Is IBM Instana right
for you?

01

Introduction

Kubernetes deserves the attention it's received in recent years for good reason. As organizations transition to managed container environments with microservice application stacks, Kubernetes has become the go-to container orchestration solution.

Despite its strengths, Kubernetes alone doesn't address all the management challenges you face, particularly regarding performance. Effectively managing orchestrated microservice applications and achieving operational excellence requires that you have a comprehensive understanding of what Kubernetes can and cannot do, as well as the capabilities your DevOps teams are looking for.

This ebook provides an in-depth examination of Kubernetes, covering its management capabilities and its limitations. The book looks at modern DevOps processes, including performance management tooling for continuous delivery of business services to achieve exceptional operational performance. It also provides a detailed analysis of how to successfully operate and manage the performance of microservice applications running on Kubernetes.

02

Kubernetes basics—or the ABCs of K8s

Kubernetes, sometimes abbreviated K8s, is a container orchestration tool for microservice application deployment. It originated as an infrastructure orchestration tool built by Google to help manage container deployment in its hyperscale environment. Google ultimately released K8s as an open-source solution through the Cloud Native Computing Foundation (CNCF).

Orchestration includes these basic Kubernetes features:

- Automated container deployment, freeing administrators of manually starting them
- Instance management, or balancing the number of instances of a given container running concurrently to meet application demand
- Domain name system (DNS) management regarding microservice and container load balancing and clustering to help manage scaling due to increased request loads
- Container distribution management across host servers spreading the application load evenly across the host infrastructure to maximize application availability

Notice there is a critical aspect of operational management missing: application performance management (APM). The whole discipline of application performance visibility and management is not part of the Kubernetes platform.

Why Kubernetes is important

Remember, the goal of DevOps is speed. Orchestration facilitates fast and easy changes to production environments so business applications can rapidly evolve.

The message is clear: speeding up your application delivery cycles can add huge value to your business.

Automating container orchestration can be a great complement to agile development methods and the microservice architecture. Modern continuous integration and continuous deployment (CI/CD automates the testing and delivery stages of development. Then containers and Kubernetes make it much easier to get your code into production and manage resources.

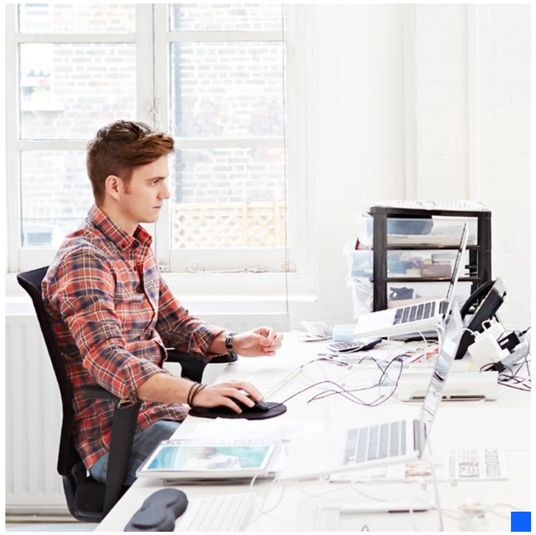
Kubernetes distributions
Many cloud providers have their own versions of Kubernetes, called "distributions." They have unique enterprise capabilities added to the open source Kubernetes version, which can provide a few distinct advantages:

- Organizations concerned about enterprise readiness get a fully tested and supported version of K8s.
- Additional enterprise functionality is included. For example, the Red Hat® OpenShift® K8s distribution adds security features and builds in automation.

For most enterprise use cases, it can be faster and easier to use a cloud provider's Kubernetes distribution than to set up the open-source version. A wide variety of Kubernetes distributions are available that can run on local infrastructure or as a hosted service in the cloud.

Note: *There are many distributions available that can be found in Kubernetes online documentation.*

Kubernetes application monitoring challenges



Container management is not application performance management

Now that we've discussed what Kubernetes does, let's explain what it doesn't do. Remember, Kubernetes orchestrates containers that are part of an application. However, it does not measure application performance or availability of highly distributed applications. Similarly, it doesn't consider performance when managing infrastructure.

Kubernetes effectively adds a layer of abstraction between the running application (containers) and the actual compute infrastructure. On its own, Kubernetes makes decisions about where containers run and can move them around abruptly. Visibility of exactly how your technical stack is deployed and how service requests are flowing across the microservices is not easily available with Kubernetes, nor is performance data—request rate, errors and duration or latency—of services a native part of Kubernetes. Production monitoring of application performance and health is not part of the Kubernetes specification.

Let's look at other aspects of orchestrated containerized application environments that can further complicate monitoring.

Any microservice application, by definition, can create a trio of issues:

- Exponentially more individual components
- Constant change in the infrastructure and applications (the application stack)
- Dynamic application components

A Kubernetes environment tends to include many more moving parts than there would be in a traditional application stack. And that means more complexity.

With the addition of containers and then orchestration with Kubernetes, each of these management challenges can become more difficult. Every time there is a decoupling of physical deployment from the application functionality, it’s often more difficult to monitor application performance and solve problems. Instead of host servers connected with a physical network, Kubernetes uses a cluster of nodes and virtualizes the network, which can be distributed across a mix of on-premises and cloud-based infrastructure, or even multiple clouds.

With so many different pieces of infrastructure and middleware, along with the polyglot of application languages used to create the microservices, it’s difficult for monitoring tools to distinguish the disparate needs and behaviors of all these critical components in the application stack. For example, collecting and interpreting monitoring data from any one platform is likely different from all other platforms. What do you do when you face Python, Java, PHP, .NET, application proxies, 4 different databases and a multitude of middleware?

Decoupling microservices from physical infrastructure

Kubernetes takes control of running the containers that make up the microservices of your application, completely automating their lifecycle management and abstracting the hardware.

Kubernetes will run the requested workloads on any available host or node, using software-defined networks to ensure that those workloads are reachable and load balanced. Compute resources—memory and CPU—are also abstracted with each workload that has a configured limit for those resources. Because containers are ephemeral, any long-term storage is provisioned by persistent volume claims provided by various storage drivers.

The already deep level of abstraction may be further compounded by Kubernetes nodes running on external cloud computing services such as Amazon Elastic Compute Cloud (Amazon EC2), Google Compute Engine (GCE) or Microsoft Azure.

The high level of disconnect between the application code and the hardware it's running on tends to make traditional infrastructure monitoring less critical. This process however elevates the importance of understanding how the microservices and overarching applications are performing and if they are meeting their desired service level agreements (SLAs). An understanding of the overall health of the Kubernetes backplane is also helpful to ensure the highest levels of service for your application.

Service mapping: a new layer of abstraction

One of the main reasons to use an orchestrator such as Kubernetes is to automate the deployment of containers and establish communications between them. However, orchestrators on their own can't necessarily guarantee that microservices can communicate and integrate with each other effectively. To do that, it's helpful to directly monitor the services and their interactions.

Direct monitoring is challenging because Kubernetes doesn't offer a way to automatically map or visualize relationships between microservices.

Admins would typically have to determine which microservices are actually running, where within the cluster they exist, which services depend on other ones, and how requests are flowing between services—either manually or from other data sources.

When updating software or performing regression tests, the faster that the application owners, DevOps and developers can see the impact from the update, the sooner they can move on to the next step of the project—and know whether or not their applications are behaving appropriately.

Root cause ambiguity

APM tools mostly exist because middleware-based business applications—first using Java and .NET, then using service-oriented architecture (SOA) principles, and even microservices and containers—can make it difficult to monitor application performance, trace user requests, identify problems and fix them when they occur.

The more complex the application environment, the more difficult it is for DevOps teams to get the performance visibility and component dependencies needed to effectively manage application performance.

In a Kubernetes environment, determining the root cause of a problem based on surface-level symptoms can be even more difficult. That's because the relationships between different components of the environment tend to be harder to map—and they continuously change. For example, a problem in a Kubernetes application might be caused by an issue with physical infrastructure, but it could also be due to a configuration mistake or coding problem. Or perhaps the problem lies within the virtual network that allows microservices to communicate with each other.

Of course, when the problem lies within the application code, deep code-level visibility can help debug actual code issues. This kind of deeper visibility might even recognize when bad parameters or other inputs are causing application problems. Ultimately, there could be myriad of root causes for the issue, ranging from configuration problems in Kubernetes to an issue with data flows between containers to a physical hardware failure.

To put it simply, tracing problems in a Kubernetes environment back to their root cause might not be feasible in many cases—unless you use tools that can automatically parse through the complex web of data and dependencies that compose your cluster and your microservice application's structure.

Seeing through the complexity

Although managing the performance and availability of Kubernetes applications might be challenging and intimidating, it's not impossible. The right APM tool that recognizes the unique challenges of monitoring a Kubernetes environment can help you manage your environment in a way that maximizes uptime and optimizes performance. Getting the best of both worlds helps you combine the benefits of K8s with the goal of DevOps excellence.

Let's look at key types of visibility for supporting applications running in a Kubernetes environment.

Application service identification and mapping
As discussed earlier, Kubernetes injects a new level of application abstraction which can make it difficult to evaluate individual services or the interdependencies between all the deployed services. The deeper your APM tools can see past Kubernetes and the container system, the easier it is to identify application services and their dependencies.

A Kubernetes service is an abstraction which defines a logical set of pods and a policy that allows you to access them.

There can be multiple application services within a pod.

Microservice relationships
If you know how your Kubernetes services map to application services, the microservices they are built upon, and their physical infrastructure, then you can determine how the infrastructure impacts the services' availability and performance. Kubernetes doesn't necessarily make it easy to get all this information. It probably requires you to run multiple kubectl commands—using the command line tool for communicating with a Kubernetes cluster's control plane—to manually build a mapping at a single point in time. That can be quite a distraction when a production issue arises.

Application request mapping and tracing
The microservices that comprise an application constantly send and receive requests from each other. It can be much easier to monitor and manage microservice application performance if your APM tool can detect all the services and the interdependencies between them and visualize the dynamic relationships—that is, map them—in real time.

Troubleshooting can be easier if the tool can also extract individual traces from each specific application request, especially across all the distributed microservices it touches.

Deployment failures
If Kubernetes fails to deploy a pod as expected, it could be helpful if you knew when, why and how it happened. However, it's probably more important to understand if your application functionality has been negatively impacted by this deployment failure. Is your application slower and handling less workload, or is it throwing errors because a critical service is unavailable?

The event stream can show where the deployment failed, but because you can't see the performance of your application using kubectl commands, you would have to find an alternative way to answer the above question, most likely with an APM tool that understands Kubernetes.

Performance regressions
If your application is responding slowly, you probably want to identify the issue and trace it to its root cause quickly. Unfortunately, Kubernetes doesn't natively include performance monitoring, so there aren't really any kubectl commands you can run to understand microservice or application performance.

When troubleshooting microservice applications running on Kubernetes, it's helpful if your APM tool can correlate metrics up and down the full application stack: infrastructure, application code, Kubernetes system information and the trace data between the services.

Infrastructure metrics such as central processing unit (CPU), memory, disk input and output (I/O), network I/O and so on can be good KPIs to reference while troubleshooting performance issues. However, they tend to be only a part of the information required to fully determine root cause. There might also be issues with the application code or Kubernetes configuration issues that are causing resource contention. For example, it can be common to over-allocate CPU and memory resources on Kubernetes nodes through improper configuration.

Performance optimization opportunities
In agile development environments, developers often push new code into production on a daily basis. How do they know their code is delivering good response time and not consuming too many resources?

The more responsive and flexible your APM tool is regarding changes, the more likely it is to help you automatically and immediately recognize when new code has been deployed as well as any infrastructure or architectural changes. It's even better if it can also help developers analyze their code with line-of-code visibility or production profiling.

For these use cases, it can help to have granular visibility into user requests, host resources (K8s nodes) and workload patterns. Also helpful are robust analytics that can include all the data you're collecting, especially when working with data beyond the scope of what Kubernetes natively captures.

Should you use Prometheus?
Prometheus has become one of the standard performance monitoring methods for Kubernetes, but it may also have some critical functional gaps. Let's begin by discussing what Prometheus does well.

Prometheus is an open source time series metrics monitoring and alerting tool. It is typically used to monitor KPIs such as rates, counters and gauges from infrastructure and application services. You can use Prometheus to monitor request response times. But this method often requires you to modify your source code to add the Prometheus application programming interface (API) calls. Prometheus can help you understand overall response times and request rates, but the approach can also lack the level of detail required to troubleshoot or optimize application performance.

The Kubernetes distribution natively supports Prometheus. When the Prometheus Helm package is installed, you'll find several dashboards preconfigured for the purpose of basic health checks. You'll also find a few predefined alerts configured on your cluster.

Time series metrics

Flexible API

High cardinality

Monitoring and alerting

Ultimately, Prometheus can be a good stand-alone metrics tool. But that's not necessarily the same thing as being able to meet the performance management challenges associated with running business-critical microservice workloads on Kubernetes. Here are some drawbacks which could make it difficult for Prometheus to solve a number of critical use cases:

- No distributed tracing capability
- No correlation between service infrastructure and host
- No correlation between Kubernetes resources, request response times and infrastructure metrics
- No built-in analytics interface, roll-ups or aggregates
- No automatic root cause analysis
- No automated alerting
- Management and administrative costs

Kubernetes monitoring tools and strategies

It can be easier and more effective to achieve the above elements in a Kubernetes application with an APM tool that includes features beyond traditional monitoring tools.

For Kubernetes, it can be much easier to manage and deal with issues if your tool does more than just collect monitoring data and detect anomalies. Here are some key suggested capabilities to look for that could help an APM tool ensure the performance of Kubernetes-based applications.

Root cause analysis within the application, containers and orchestration
When problems occur, it's helpful if your monitoring solution can identify the root cause of performance issues automatically, no matter where in the technology stack the problem occurs.

Outside the Kubernetes environment, your monitoring solution should not only detect problems within Kubernetes but also trace problems to their exact cause to help you quickly solve them.

Given the extreme complexity of a Kubernetes-based application and the usual lack of visibility into that environment, identifying the root causes of availability or performance issues can be challenging to accomplish manually.

Integrated service and infrastructure mapping
To help fill gaps in dependencies between application services and components, it's helpful if your monitoring tool can detect and map services automatically.

Even better, it should also understand the relationships and dependencies between those services, helping create more accurate maps. This accuracy can help users identify problems between two disparate services.

Dynamic baselining
With environment architectures and configurations changing constantly, it's easier for your team to stay up to date if your APM tool also makes sense of highly dynamic monitoring data to distinguish true anomalies from normal changes.

Remediation guidance
When something goes wrong in your enormously complex Kubernetes environment, you need to resolve the problem quickly. That can be difficult for human admins to do without the help or guidance from an APM tool, due to large amounts of data and fast-changing variables. It's nearly impossible for humans to wade through the data and formulate an incident response plan on their own.

06

Conclusion

Although Kubernetes is rapidly expanding throughout the DevOps world as a standard infrastructure platform, it doesn't always include the application performance visibility teams need. Kubernetes introduces a new layer of abstraction into the data center, often creating observability challenges that make it more difficult to manage application availability. How would you deliver the needed performance SLAs demanded by your business in a Kubernetes environment?

To effectively manage business-critical applications on Kubernetes, look for an APM tool that includes these key capabilities:

- Full-stack visibility—including infrastructure, code, microservices, request traces, middleware, containers and Kubernetes—of all technology layers
- Continuous discovery of the full application stack to automatically adjust to changes in the environment
- Dependency mapping and correlation between the layers of technology
- Automatic root cause determination and assistance for DevOps teams to troubleshoot application issues

07

Is IBM Instana right for you?

IBM Instana® is an enterprise observability platform that includes automated application performance monitoring capabilities. It’s designed for businesses operating complex, modern, cloud-native applications no matter where they reside—on premises, in public and private clouds, on mobile devices or in an IBM zSystems™ environment.

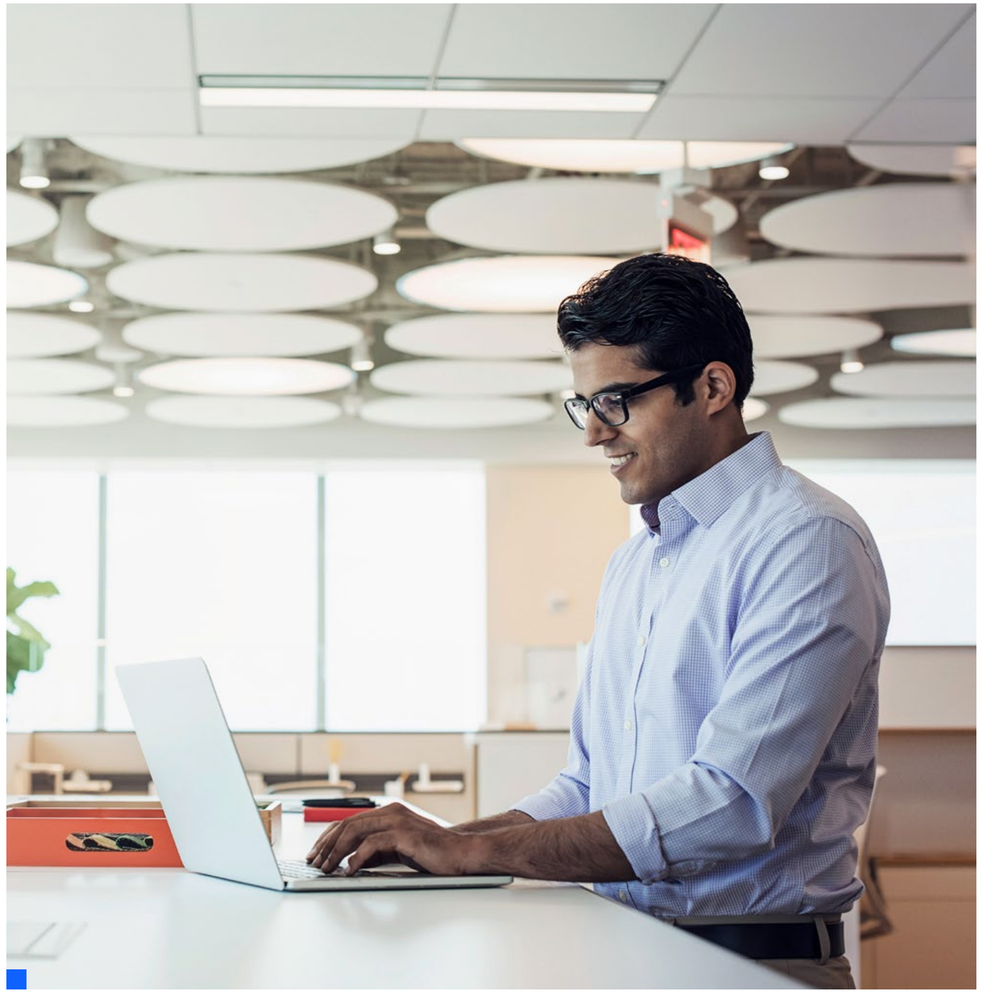
IBM Instana helps you control modern hybrid applications with AI-powered discovery of deep contextual dependencies inside hybrid applications. IBM Instana also provides visibility into development pipelines to help enable closed-loop DevOps automation.

These capabilities provide actionable feedback needed for clients as they optimize application performance, enable innovation and mitigate risk. These features help DevOps increase efficiency and add value to software delivery pipelines so they can meet their service and business-level objectives.

See the power of IBM Instana for yourself. Sign up today for a free 14-day trial of the full version of the product. No credit card required.

IBM Instana free trial →

Explore IBM Instana →



© Copyright IBM Corporation 2023

IBM Corporation
New Orchard Road
Armonk, NY 10504

Produced in the United States of America
May 2023

IBM, the IBM logo, IBM Instana, and zSystems are trademarks or registered trademarks of International Business Machines Corporation, in the United States and/or other countries. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on ibm.com/trademark.

Red Hat and OpenShift are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

Microsoft is a trademark of Microsoft Corporation in the United States, other countries, or both.

This document is current as of the initial date of publication and may be changed by IBM at any time. Not all offerings are available in every country in which IBM operates.

It is the user’s responsibility to evaluate and verify the operation of any other products or programs with IBM products and programs. THE INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT ANY WARRANTY, EXPRESS OR IMPLIED, INCLUDING WITHOUT ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND ANY WARRANTY OR CONDITION OF NON-INFRINGEMENT. IBM products are warranted according to the terms and conditions of the agreements under which they are provided.