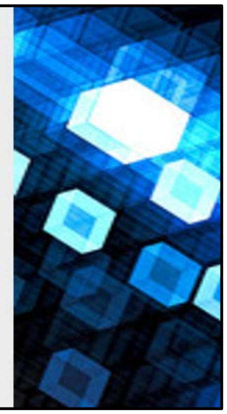
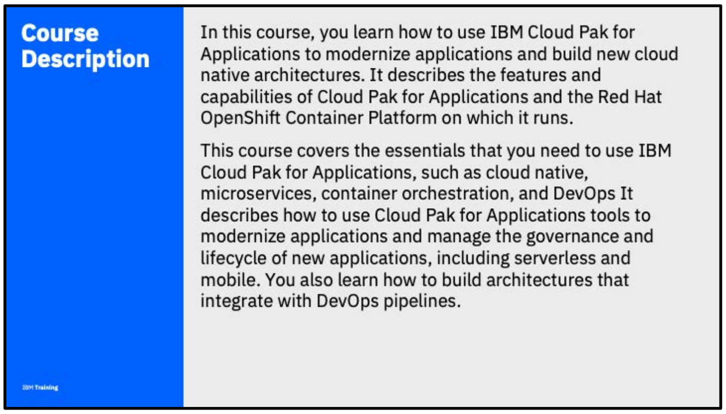
IBM Cloud Pak for Applications Solution Architect Workshop

WD108/ZD108
ERC 1.0



IBM **Training**

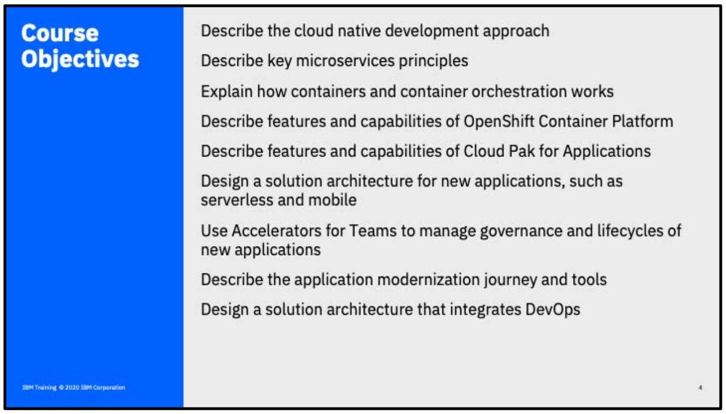


Note: This course and the associated IBM Professional Certification exam is based on IBM Cloud Pak for Applications V4.1, which runs on OpenShift Container Platform V4.2. At the time of publishing this course, the latest version available was IBM Cloud Pak for Applications V4.2, which runs on OpenShift Container Platform V4.4. Any major differences between versions are identified where possible.

Audience & Prerequisites

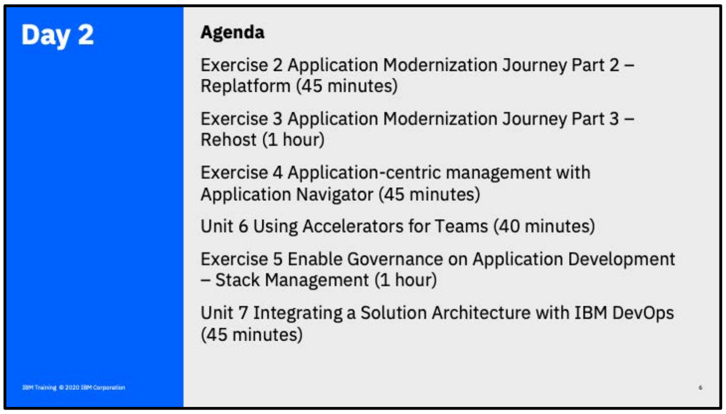
This course is for system architects and designers who want to a build solution architecture by using IBM Cloud Pak for Applications

The ideal candidate for this course has a strong foundation in cloud computing and working knowledge of the Linux operating system


Course Objectives
Describe the cloud native development approach
Describe key microservices principles
Explain how containers and container orchestration works
Describe features and capabilities of OpenShift Container Platform
Describe features and capabilities of Cloud Pak for Applications
Design a solution architecture for new applications, such as serverless and mobile
Use Accelerators for Teams to manage governance and lifecycles of new applications
Describe the application modernization journey and tools
Design a solution architecture that integrates DevOps

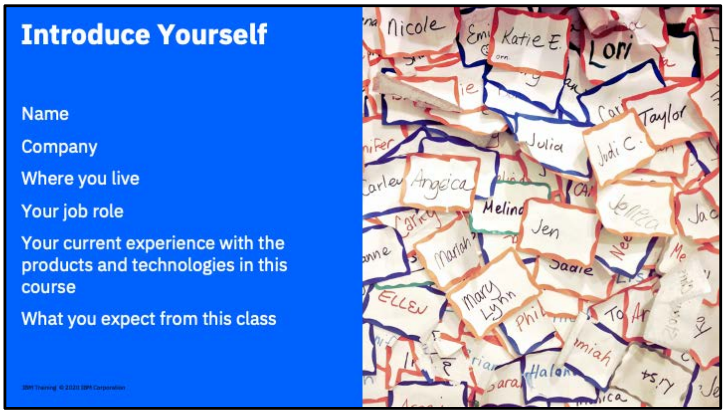
For instructor-led (ILT/ILO) deliveries:

These durations are time estimates for lecture material and hosted lab exercises only and do not include time for discussions, breaks, or supplemental learning materials such as videos, readings, and optional assignments.

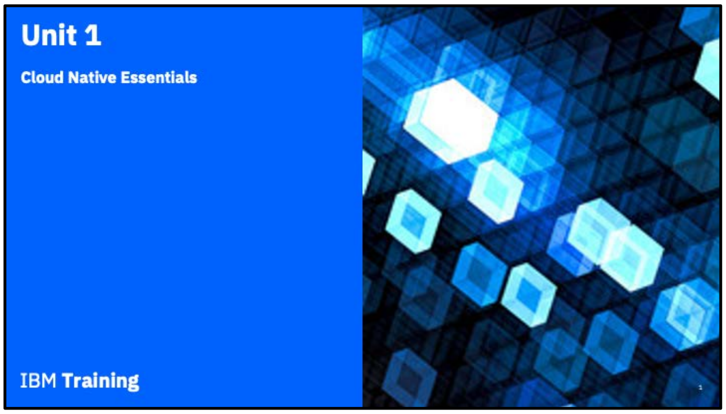


For instructor-led (ILT/ILO) deliveries:

These durations are time estimates for lecture material and hosted lab exercises only and do not include time for discussions, breaks, or supplemental learning materials such as videos, readings, and optional assignments.



For instructor-led (ILT/ILO) deliveries.



Unit 1: Cloud Native Essentials

This unit introduces you to the important concepts you need to know about to use Cloud Pak for Applications. It is not an exhaustive discourse on the topic but summarizes essential knowledge and provides entry points for further study.

Presentation time: about 1 hour

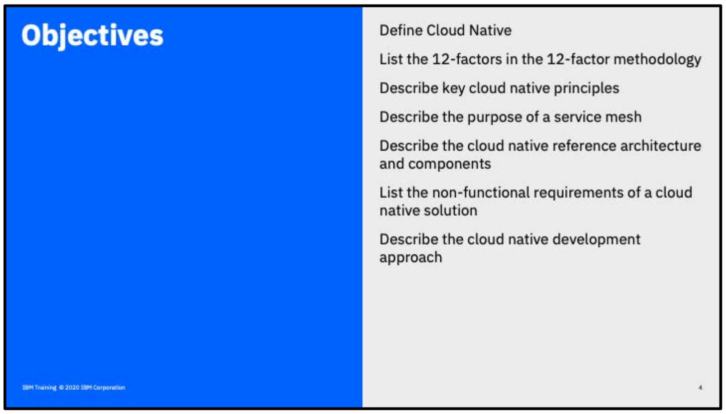
Supplemental material:

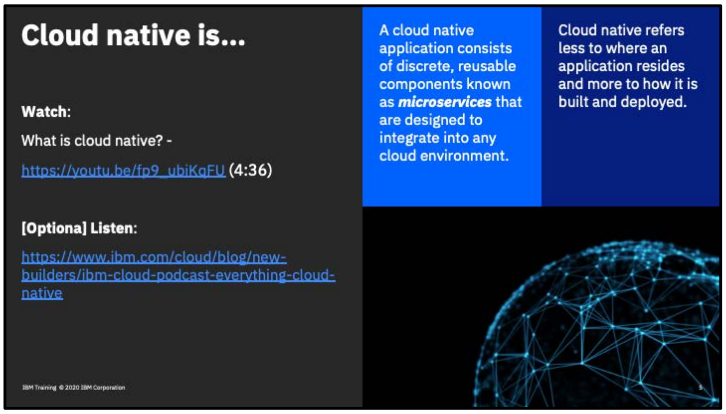
Videos: 1 hour

Readings: 50 minutes

Topics


What is
cloud native?
IBM Training © 2020 IBM Corporation
3


Objectives
Define Cloud Native
List the 12-factors in the 12-factor methodology
Describe key cloud native principles
Describe the purpose of a service mesh
Describe the cloud native reference architecture and components
List the non-functional requirements of a cloud native solution
Describe the cloud native development approach
IBM Training © 2020 IBM Corporation
4



What is cloud native?

Cloud native refers less to where an application resides and more to how it is built and deployed.

A cloud native application consists of discrete, reusable components known as microservices that are designed to integrate into any cloud environment.

These microservices act as building blocks and are often packaged in containers.

Microservices work together as a whole to comprise an application, yet each can be independently scaled, continuously improved, and quickly iterated through automation and orchestration processes.

The flexibility of each microservice adds to the agility and continuous improvement of cloud-native applications.

In the video, "What is Cloud Native?", Andrea Crawford gives an overview of some of the key concepts. (4:36)

You can also dive into the IBM Cloud Podcast, which has a seven-episode miniseries entitled "Everything Cloud Native."

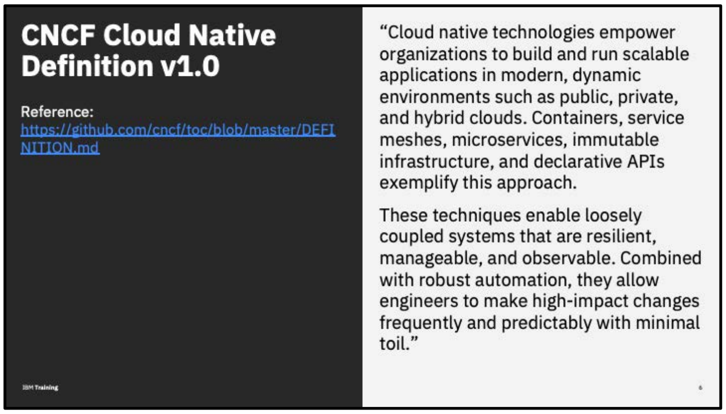
Watch:

What is cloud native? -

https://youtu.be/fp9_ubiKqFU (4:36)

[Optional] Listen:

https://www.ibm.com/cloud/blog/new-builders/ibm-cloud-podcast-everything-cloud-native

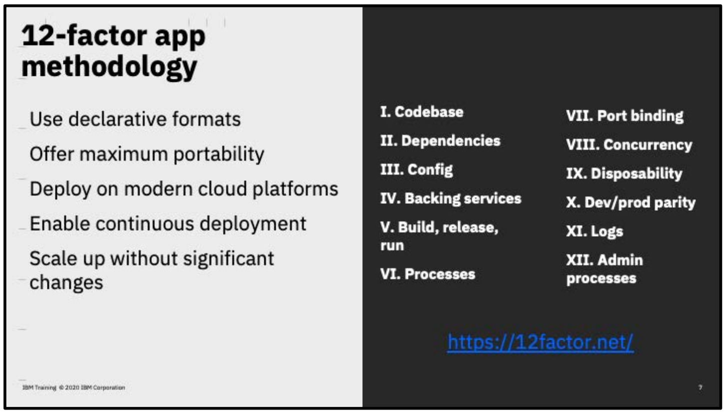


You can find an "official" definition of cloud native at the Cloud Native Computing Foundation web site. According to the CNCF:

Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.

These techniques enable loosely coupled systems that are resilient, manageable, and observable. Combined with robust automation, they allow engineers to make high-impact changes frequently and predictably with minimal toil.

The Cloud Native Computing Foundation seeks to drive adoption of this paradigm by fostering and sustaining an ecosystem of open source, vendor-neutral projects. They democratize state-of-the-art patterns to make these innovations accessible for everyone.



Cloud native development often applies the 12-factor app methodology. It is a methodology for building software-as-a-service apps that:

Use **declarative** formats for setup automation, to minimize time and cost for new developers joining the project;

Have a **clean contract** with the underlying operating system, offering **maximum portability** between execution environments;

Are suitable for **deployment** on modern **cloud platforms**, obviating the need for servers and systems administration;

Minimize divergence between development and production, enabling **continuous deployment** for maximum agility;

And can **scale up** without significant changes to tooling, architecture, or development practices.

The twelve-factor app methodology can be applied to apps that are written in any programming language, and which use any combination of backing services (database, queue, memory cache, and so on).

The 12 factors are:

I. Codebase

One codebase tracked in revision control, with many deploys

II. Dependencies

Explicitly declare and isolate dependencies

III. Config

Store configuration in the environment

IV. Backing services

Treat backing services as attached resources

V. Build, release, run

Strictly separate build and run stages

VI. Processes

Execute the app as one or more stateless processes

VII. Port binding

Export services by using port binding

VIII. Concurrency

Scale out by using the process model

IX. Disposability

Maximize robustness with fast startup and graceful shutdown

X. Dev/prod parity

Keep development, staging, and production environments as similar as possible

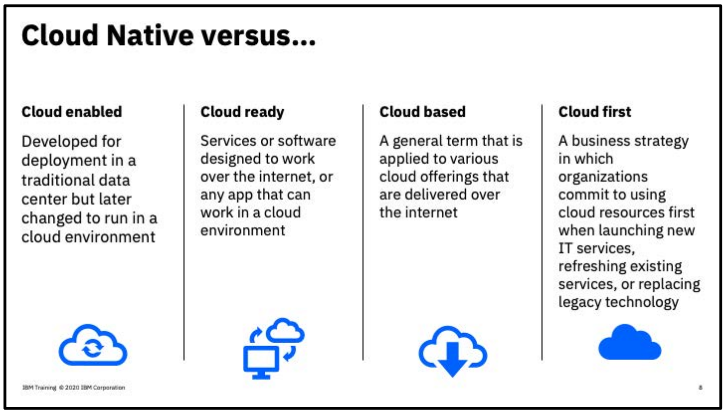
XI. Logs

Treat logs as event streams, and

XII. Admin processes

Run admin or management tasks as one-off processes

You can see a detailed definition of each factor at the 12factor.net web site here - https://12factor.net/



Do not confuse cloud-native with other cloud terminology, such as...

Cloud native vs. Cloud enabled

A cloud enabled application is an application that was developed for deployment in a traditional data center but was later changed so that it also could run in a cloud environment. Cloud native applications, however, are built to operate only in the cloud. Developers design cloud native applications to be scalable, platform agnostic, and comprised of microservices.

Cloud native vs. Cloud ready

In the short history of cloud computing, the meaning of "cloud ready" has shifted several times. Initially, the term applied to services or software designed to work over the internet. Today, the term is used more often to describe an application that works in a cloud environment or a traditional app that has been reconfigured for a cloud environment. The term "cloud native" has a much shorter history and refers to an application developed from the outset to work only in the cloud and take advantage of the characteristics of cloud architecture or an existing app that has been refactored and reconfigured with cloud native principles.

Cloud native vs. Cloud based

A cloud-based service or application is delivered over the internet. It's a general term applied liberally to any number of cloud offerings. Cloud native is a more specific term. Cloud native describes applications designed to work in cloud environments. The term denotes applications that rely on microservices, continuous integration and continuous delivery (CI/CD) and can be used via any cloud platform.

Cloud native vs. Cloud first

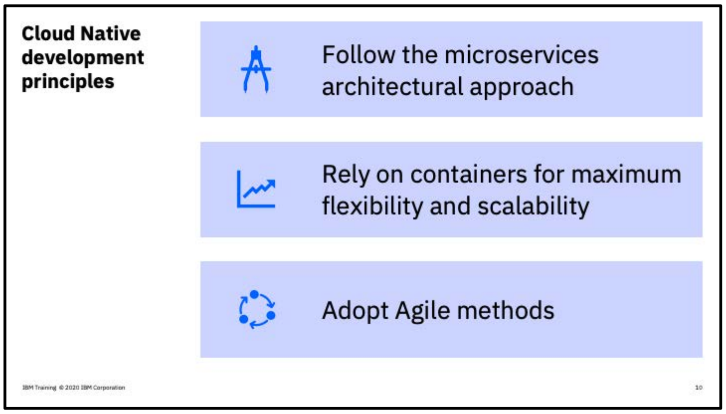
Cloud first describes a business strategy in which organizations commit to using cloud resources first when launching new IT services, refreshing existing services, or replacing legacy technology. Cost savings and operational efficiencies drive this strategy. Cloud native applications pair well with a cloud-first strategy because they use only cloud resources and are designed to take advantage of the beneficial characteristics of cloud architecture.

A microservice is a small application with a small footprint that performs a specific function. Microservices enable an architectural approach where a much larger application is composed of discrete, independently deployed components. The microservices approach to software development can be used in multiple ways but has become closely associated with cloud native application development. Microservices work well with cloud native apps because of their modular structure, compatibility with Agile and DevOps processes, and harmony with container use.

For a full overview of microservices, see "Microservices: A Complete Guide."

Read:

Microservices: A Complete Guide - https://www.ibm.com/cloud/learn/microservices

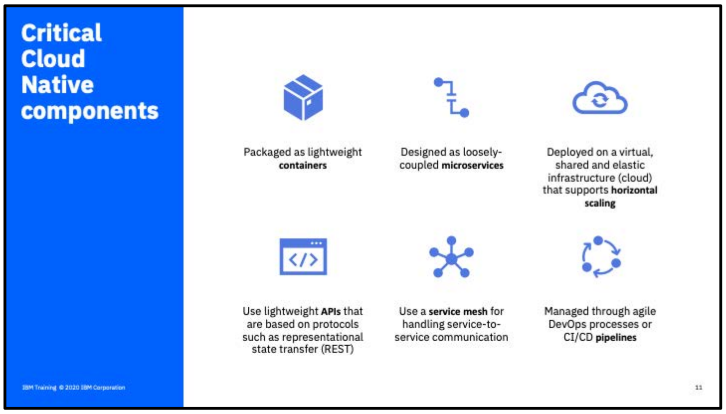


Whether creating a new cloud native application or modernizing an existing application, developers adhere to a consistent set of principles:

They follow the microservices architectural approach by breaking applications down into single-function services. These services are loosely coupled but remain independent, allowing the incremental, automated, and continuous improvement of an application without causing downtime.

They rely on containers for maximum flexibility and scalability: Containers package software with all its code and dependencies in one place, allowing the software to run anywhere. This allows maximum flexibility and portability in a multicloud environment. Containers also allow fast scaling up or down with Kubernetes orchestration policies that are defined by the user.

And, they adopt Agile methods: Agile methods speed the creation and improvement process. Developers can quickly iterate updates based on user feedback, allowing the working application version to match as closely as possible to end-user expectations.



To be considered cloud native, an application must be infrastructure-agnostic and use containers. Containers provide the application with a lightweight runtime, libraries, and dependencies that allow it to run as a stand-alone environment able to move in and out of the cloud—independent of virtual servers or compute instances. Therefore, containers can be portable between different environments.

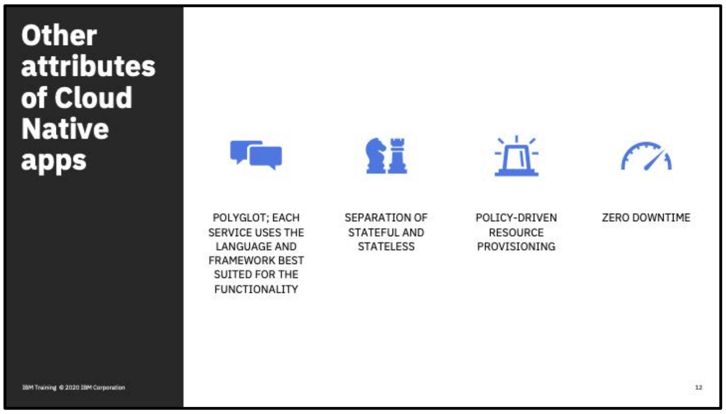
Microservices are are loosely coupled services that allow developers to treat each service independent of the other. With this decoupling, a developer can focus on the core functionality of each service to deliver fine-grained functionality. This approach leads to efficient lifecycle management of the overall application, because each service is maintained independently and with clear ownership.

Cloud-native applications are deployed on a virtual, shared and elastic infrastructure – in other words, cloud. Unlike virtual machines, containers can scale-out and scale-in rapidly. This is done through the container orchestrator, and not by physical servers.

Cloud-native services use lightweight APIs that are based on protocols such as representational state transfer (REST). REST is the lowest common denominator to expose APIs over hypertext transfer protocol (HTTP).

If you’re building a cloud native application, you need a service mesh. Cloud native applications can grow to become hundreds or even thousands of interrelated services. Services that belong to the same application discover each other through the application runtime. They exist independent of other services. A service mesh is a dedicated infrastructure layer for making service-to-service communication safe, fast, and reliable.

Each service of a cloud-native application goes through an independent life cycle, which is managed through an agile DevOps process. Multiple continuous integration/continuous delivery (CI/CD) pipelines might work in tandem to deploy and manage a cloud-native application. DevOps is covered later, too.

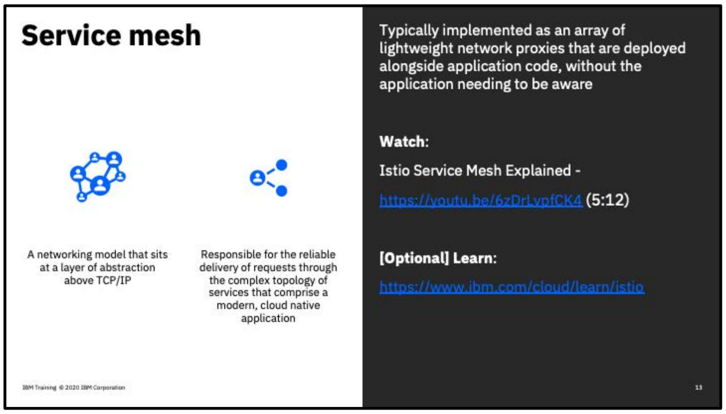


Each service of a cloud-native application is developed by using the language and framework best suited for the functionality. Cloud-native applications are polyglot; in other words, services use a variety of languages, runtimes and frameworks.

Persistence is a factor that must be viewed in context with state, and statelessness and is a consideration for storage. Cloud native apps are architected with a clean separation of stateless and stateful services.

Governance can be applied to cloud-native apps through a defined set of policies. You can use policies to control central processing unit (CPU) and storage quotas, and network policies that allocate resources to services.

Cloud native apps have zero downtime because container orchestrators like Kubernetes can do rolling updates. A rolling update incrementally updates Pod instances with new ones, rather than all at once, so there is effectively no downtime.



A service mesh is a networking model that sits at a layer of abstraction above TCP/IP. It is responsible for the reliable delivery of requests through the complex topology of services that comprise a modern, cloud native application. It is typically implemented as an array of lightweight network proxies that are deployed alongside application code, without the application needing to be aware.

The concept of the service mesh as a separate layer is tied to the rise of the cloud native application. In the cloud native model, a single application might consist of hundreds of services; each service might have thousands of instances; and each of those instances might be in a constantly-changing state as they are dynamically scheduled by an orchestrator like Kubernetes. Not only is service communication in this world incredibly complex, it is a pervasive and fundamental part of runtime behavior. Managing it is vital to ensuring end-to-end performance and reliability.

Watch the video, Istio Service Mesh Explained, to learn more: https://youtu.be/6zDrLvpfCK4 (5:12)

Read more about Istio here - https://www.ibm.com/cloud/learn/Istio

Watch:

Istio Service Mesh Explained -

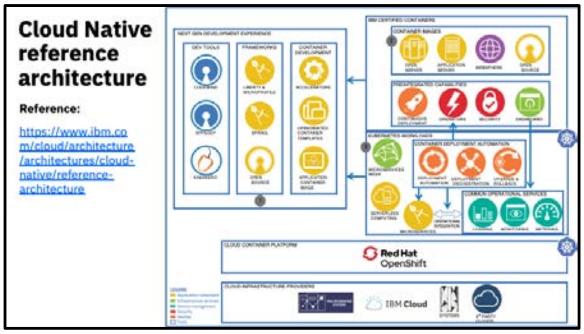
https://youtu.be/6zDrLvpfCK4 (5:12)

[Optional] Learn:

https://www.ibm.com/cloud/learn/istio

Reference:

https://www.cncf.io/blog/2017/04/26/service-mesh-critical-component-cloud-native-stack/



The Cloud-native reference architecture includes a set of technologies to build and run scalable applications in public, private, and hybrid clouds. With techniques that include containers, service meshes, and microservices, you can enable loosely coupled systems that are resilient, manageable, and observable. Combine those techniques with automation to make frequent high-impact changes.

To design a cloud native solution with this architecture:

First,

Developers and architects can build, run, and test solutions locally, by using a set of development tools and frameworks. Architects work with development teams to agree on a framework and runtimes to consolidate applications.

Step 2

Use container images that are certified on Red Hat OpenShift by using best practices. Continuously integrate and deploy by using the provided open source tools.

Step 3

Configure your microservices to use service mesh to make microservices leaner in complexity and have the control, observability, and security shift to the service mesh. Optimize hardware resources by using serverless computing to scale to zero.

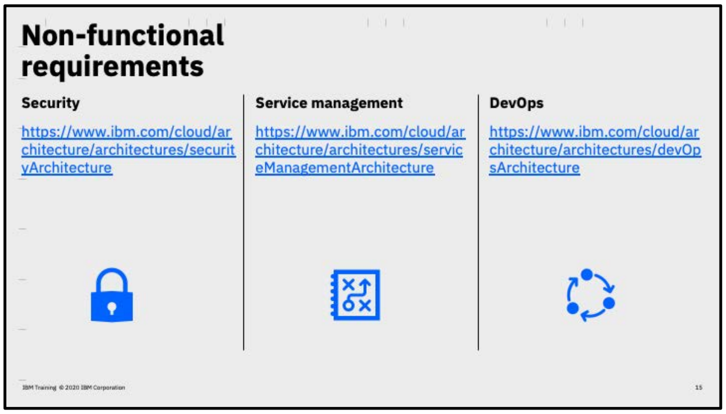
Step 4

The operations team manages applications by instrumenting metrics, tracing, and logs. Dashboards, GitOps, ChatOps, and event management are also used to run workloads in production.

Visit the Cloud Native reference architecture at this link for a closer look.

Reference:

https://www.ibm.com/cloud/architecture/architectures/cloud-native/reference-architecture



Cloud-native architectures must satisfy several nonfunctional requirements: Security, Service Management, and DevOps.

Secure applications require certificate management, microservice-based network policies that control application access, distributed denial-of-service (DDoS) protection or mitigation, and web application firewalls (WAF).

Authentication in Kubernetes can also be provided by a service mesh, such as Istio, through its mutual TLS authentication. That authentication aims to enhance the security of microservices and their communication without requiring service code changes. The service mesh is responsible for these tasks:

- Providing each service with a strong identity that represents its role to enable interoperability across clusters and clouds
- Securing service-to-service communication and user-to-service communication
- Providing a key management system to automate key and certificate generation, distribution, rotation, and revocation

Service management nonfunctional requirements include health checks, logging, metrics, and distributed tracing.

You can configure Istio-enabled applications to collect trace spans by using Zipkin or Jaeger. Regardless of what language, framework, or platform you use to build your application, Istio can enable distributed tracing.

DevOps is not a strict requirement for your applications, but CI/CD pipelines are often cited as pillars of successful software development and DevOps practices. If you are deploying software into production, it's best to use a CI/CD pipeline.

Because continuous deployment is key in delivering cloud-based applications, the Ops part of your DevOps team has much less time to build and apply knowledge to prepare for each deployment.

You can learn more about each architecture at the links shown here:

Security

https://www.ibm.com/cloud/architecture/architectures/securityArchitecture

Service management

https://www.ibm.com/cloud/architecture/architectures/serviceManagementArchitecture

DevOps

https://www.ibm.com/cloud/architecture/architectures/devOpsArchitecture

1. When considering the right cloud architecture for your applications and workloads, you must begin with the unique needs of your business. This can include many factors such as government regulations, security, performance, data residency, service levels, time to market, architecture complexity, skills and preventing vendor lock-in.

2. The right cloud model depends on your workload. You should understand the pluses and minuses of each cloud deployment model and take a methodical approach to determining which workloads to move to which type of cloud for the maximum benefit.

3. Getting started with the latest technologies like AI, machine learning, blockchain or IoT requires an understanding of how they work and what they can do. Proven reference architectures for using them plus a knowledge of development practices such as design thinking, lean startup, agile and continuous delivery position you to design, deliver and validate your ideas quickly.

4. To help you define your cloud strategy, look for experienced partners who can meet you where you are in your cloud journey, help you envision your future, and draw a road map to get there.

5. To innovate with emerging technologies, you need freedom to build to your unique business specifications. An architecture that is built on open standards is the foundation for innovative cloud solutions that are built for the enterprise and scale for production deployments.

6. Not only does your underlying infrastructure need to be built on open standards, but it must deliver key capabilities to accommodate these newer technologies and the data that fuels them, such as:

Solutions that combine bare metal servers with GPUs to accelerate AI and other data-driven workloads, and r**apidly scalable, affordable storage options.** Hybrid and multi-cloud infrastructures form the optimal base for data-driven applications. With much of enterprise data stored in on-premises systems, solutions that enable you to extend those systems to the cloud quickly and with minimal retraining of staff can help you realize the benefits of cloud sooner rather than later. Certain application servers can be run both in on-premises and cloud environments, offering yet another entry point to cloud for your legacy applications.

7. With cloud-native development, you can deliver enterprise-grade products in days or weeks versus the traditional monolithic process where it takes months, and

meet constantly changing business demands by easily updating your apps multiple times a day.

8. Microservices mean speed and quality.

9. Containers technology is a perfect match with a microservices architecture because it supports the smaller, faster narrative.

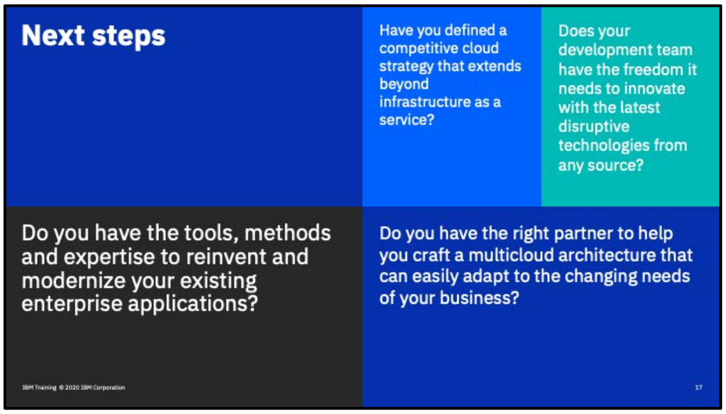
10. You must be able to monitor, manage and scale multiple components in a microservices architecture. Orchestration options range from an open source tool like Kubernetes for container management to Platform as a Service (PaaS), where developers focus on code, while the underlying orchestration technology is built into the platform itself.

11. When DevOps is fully engrained in your culture, your teams and processes perform in concert with your technologies to keep up with dynamic demands. Cloud native changes the relationship between development and operations teams. With so many moving parts, integration is critical. DevOps fosters autonomous, cross-functional teams with approaches such as the agile methodology, which supports an environment of shared responsibility, shared decision-making, trust and collaboration. When your teams adopt a DevOps culture, they can feel more confident that they are working together to deliver the end goal.

12. DevOps relies heavily on automation tools during key parts of the software delivery process to reduce the errors from awkward hand-offs and manual processes that often delay deployment. Automation frees up developer teams to focus on higher-value tasks while positioning them to iterate faster and release high-quality apps more often.

Reference:

https://www.ibm.com/cloud/smartpapers/cloud-native-application-innovation/



Moving to cloud is a bit more complicated than you might have anticipated, but leading companies are finding that it's worth the effort. With the right strategy, the right cloud infrastructure, and the integration of emerging technologies, you can propel the disruption in your industry.

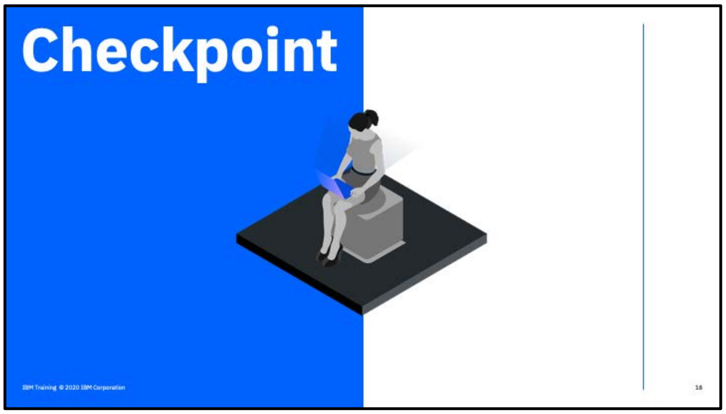
As you think about next steps, consider these questions:

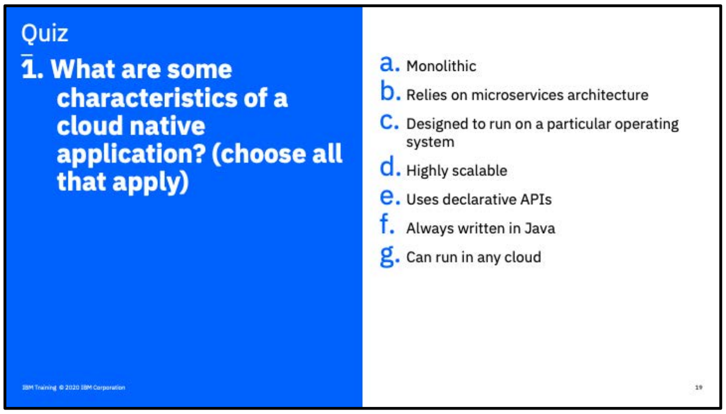
Have you defined a competitive cloud strategy that extends beyond infrastructure as a service?

Does your development team have the freedom it needs to innovate with the latest disruptive technologies from any source?

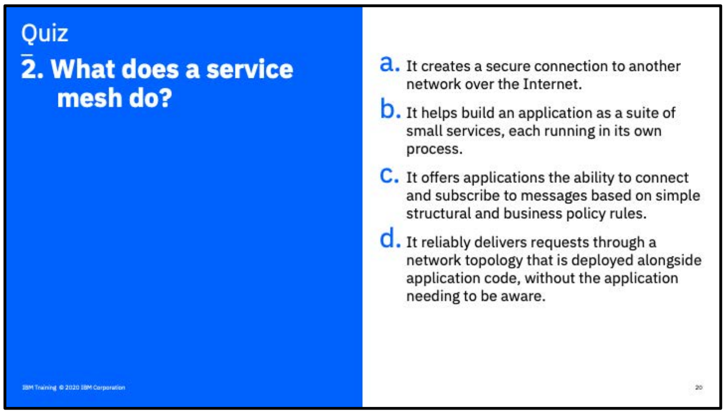
Do you have the tools, methods and expertise to reinvent and modernize your existing enterprise applications?

Do you have the right partner to help you craft a multicloud architecture that can easily adapt to the changing needs of your business?

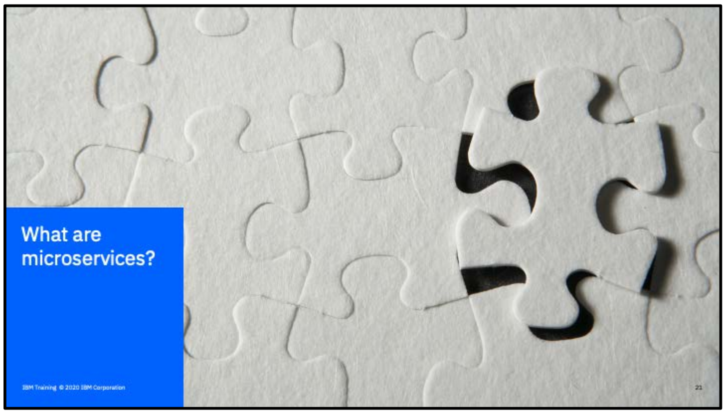

Checkpoint
IBM Training © 2020 IBM Corporation
18

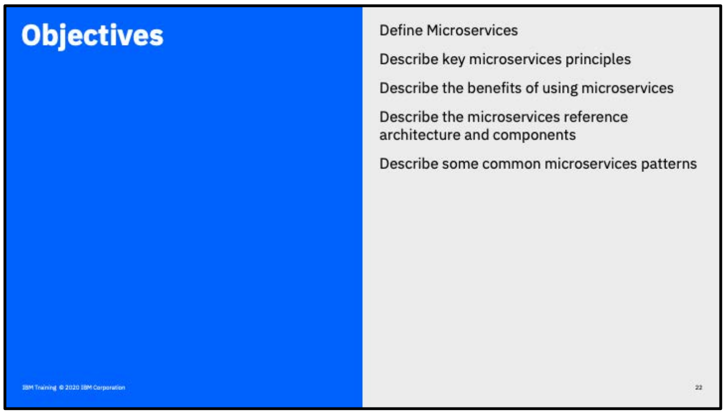


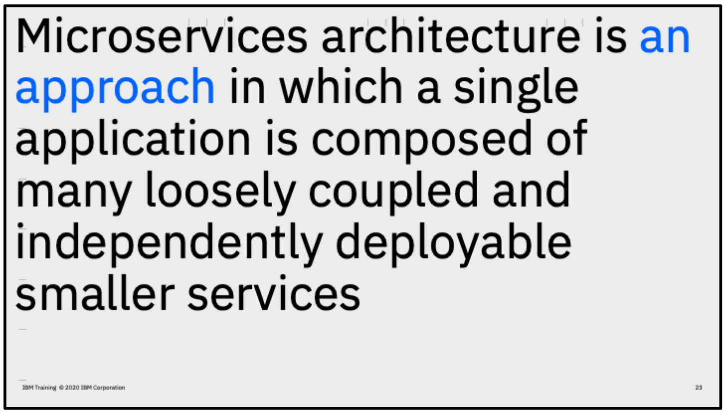
Correct answers: b, d, e, g



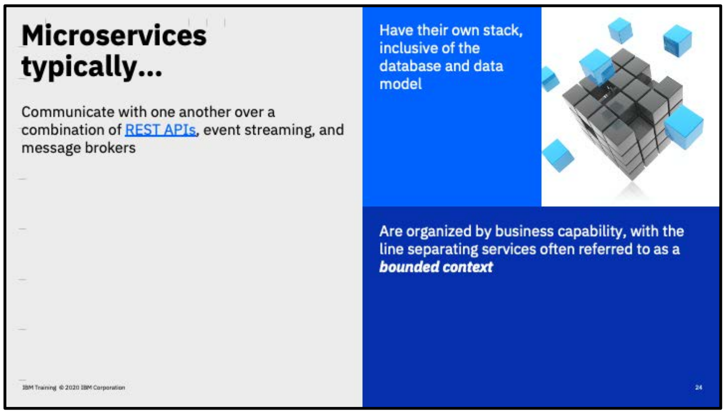
Correct answer: d.


What are
microservices?
IBM Training © 2020 IBM Corporation
21


Objectives
Define Microservices
Describe key microservices principles
Describe the benefits of using microservices
Describe the microservices reference architecture and components
Describe some common microservices patterns
IBM Training © 2020 IBM Corporation
22



Microservices architecture is an approach in which a single application is composed of many loosely coupled and independently deployable smaller services.

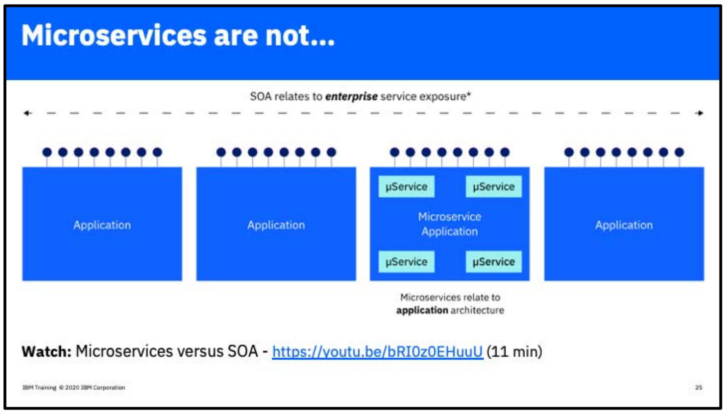


Microservices (or microservices architecture) are a cloud native architectural approach in which a single application is composed of many loosely coupled and independently deployable smaller components, or services. These services typically have their own stack, inclusive of the database and data model; communicate with one another over a combination of REST APIs, event streaming, and message brokers; and are organized by business capability, with the line separating services often referred to as a bounded context.

Each microservice can have its own DevOps pipeline, scale individually, and have its own database where it owns a data model.

References:

https://www.ibm.com/cloud/learn/rest-apis



Microservices can also be understood by what they are *not*. The two comparisons drawn most frequently with microservices architecture are monolithic architecture and [service-oriented architecture (SOA)](https://youtu.be/bRI0z0EHuuU).

While SOA and a microservices architecture seem similar and are often confused, they are in fact completely separate concepts.

The use of SOA and the associated ESB pattern is an enterprise-wide initiative to make the data and functions in systems of record readily available to new applications. Solution designers create reusable, synchronous interfaces such as web services and RESTful APIs to expose the systems of record, so new, innovative applications can be created more quickly by incorporating data from multiple systems in real time.

A microservices architecture, on the other hand, is a way of writing an individual application as a set of smaller (microservice) components in a way that makes the application more agile, scalable and resilient.

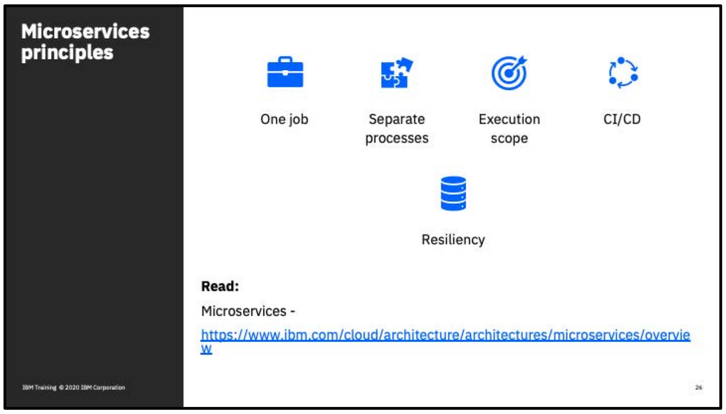
SOA was an enterprise-wide effort to standardize the way all services talk to and integrate with each other, whereas microservices architecture is application-specific.

The biggest difference between the two approaches is how they are deployed. For many years, applications were packaged in a monolithic fashion; that is, a team of developers constructed one large application that did everything for a business need. After the application was built, it was deployed multiple times across a farm of application servers. In contrast, with the microservices architectural style, developers independently build and package several smaller applications that each implement only parts of the whole application.

In summary, SOA is about real-time integration between applications, whereas a microservices architecture is about how the applications are built internally.

Watch:

Microservices versus SOA - https://youtu.be/bRI0z0EHuuU (11 min)



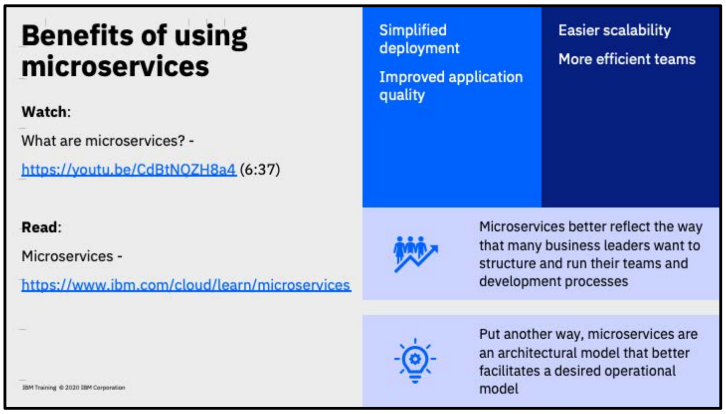
When you develop microservices for an application, keep these principles in mind:

- **One job:** Each microservice must be optimized for a single function. Each service is smaller and simpler to write, maintain, and manage.
- **Separate processes:** Communication between microservices must be conducted through APIs and message brokers. All communication from service to service must be through the service API or must use an explicit communication pattern.
- **Execution scope:** Although microservices can expose themselves through APIs, the focus is not on interfaces, but on the running components.
- **CICD:** Each microservice can be continuously integrated (CI) and continuously delivered (CD). When you build a large application that is composed of many services, you soon realize that different services evolve at different rates. If each service has a unique continuous integration or continuous delivery pipeline, the service can proceed at its own pace. In the monolithic approach, different aspects of the system are all released at the speed of the slowest moving part of the system.
- **Resiliency:** You can apply high availability and clustering decisions to each microservice. When you build large systems, another realization that you have is that when it comes to clustering, one size does not fit all. The monolithic approach of scaling all the services in the monolith at the same level can lead to the overuse or underuse of services. Even worse, when shared resources are monopolized, services might be neglected. In a large system, you can deploy services that do not need to scale to a minimum number of servers to conserve resources. Other services require scaling up to large numbers.

Read:

Microservices -

https://www.ibm.com/cloud/architecture/architectures/microservices/overview



While much of the discussion about microservices revolves around architectural definitions and characteristics, their value can be understood through simple business and organizational benefits:

- Code can be updated more easily. Each microservice is built and aligned around a business function to reduce the complexity of the application change-management process. Because each service is individually changed, tested, and deployed without affecting other services, time to market is accelerated.
- Another benefit is improved application quality. Microservices architecture lends itself to test-driven development, as components can be tested in isolation and combined with a full or virtualized set of microservices. This approach results in overall improvement in application quality.
- Components can be scaled independently of one another, reducing the waste and cost associated with having to scale entire applications, and
- Teams become more efficient. Because microservices are designed to act independently, they are naturally consistent with agile principles that promote end-to-end team ownership. Teams that develop microservices can make technology decisions that are right for the job. They can use different stacks for different components. They can experiment with new technologies, libraries, languages, and frameworks, which yields faster innovation cycles.
- Microservices are likely to be popular with executives, project leaders and developers because they better reflect the way many business leaders want to structure and run their teams and development processes.
- Put another way, microservices are an architectural model that better facilitates a desired operational model.

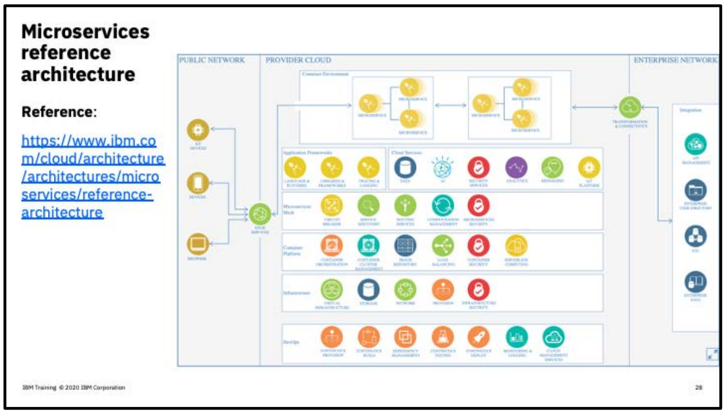
Watch:

https://youtu.be/CdBtNQZH8a4 (6:37)

Read:

Microservices **-**

https://www.ibm.com/cloud/learn/microservices



There are a lot of components in the Microservices reference architecture. The link here opens an interactive diagram where you can click on each component to learn more details.

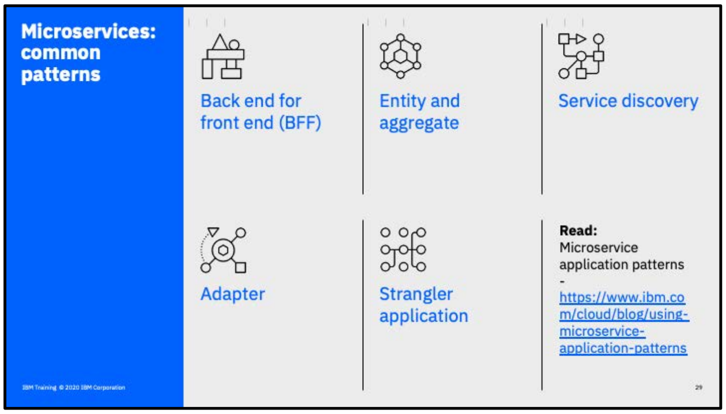
At a high level, a client browser, or device on the public network connects to Edge services, which provide network capability to deliver content through DNS, CDN, firewall, and load balancer.

The request is routed to the container environment, which hosts microservices. Microservices are supported by various application frameworks, cloud services, a service mesh, and so on.

Take some time to explore this interactive diagram at the IBM Architecture Center.

Reference:

https://www.ibm.com/cloud/architecture/architectures/microservices/reference-architecture



Within microservices architectures, there are many common and useful design, communication, and integration patterns that help address some of the more common challenges and opportunities, for example:

Backend-for-frontend (BFF) pattern: This pattern inserts a layer between the user experience and the resources that experience calls on. For example, an app used on a desktop will have different screen size, display, and performance limits than a mobile device. The BFF pattern allows developers to create and support one backend type per user interface using the best options for that interface, rather than trying to support a generic backend that works with any interface but may negatively impact frontend performance.

Entity and aggregate patterns: An entity is an object distinguished by its identity. For example, on an e-commerce site, a Product object might be distinguished by product name, type, and price. An aggregate is a collection of related entities that should be treated as one unit. So, for the e-commerce site, an Order would be a collection (aggregate) of products (entities) ordered by a buyer. These patterns are used to classify data in meaningful ways.

Service discovery patterns: These help applications and services find each other. In a microservices architecture, service instances change dynamically due to scaling, upgrades, service failure, and even service termination. These patterns provide discovery mechanisms to cope with this transience. Load balancing may use service discovery patterns by using health checks and service failures as triggers to rebalance traffic.

Adapter microservices patterns: Think of adapter patterns in the way you think of plug adapters that you use when you travel to another country. The purpose of adapter patterns is to help translate relationships between classes or objects that are otherwise incompatible. An application that relies on third-party APIs might need to use an adapter pattern to ensure the application and the APIs can communicate.

Strangler application pattern: These patterns help manage refactoring a monolithic application into microservices applications. The colorful name refers to how a vine (microservices) slowly and over time overtakes and strangles a tree (a monolithic application).

You can learn more about these patterns in the article link here.

Read:

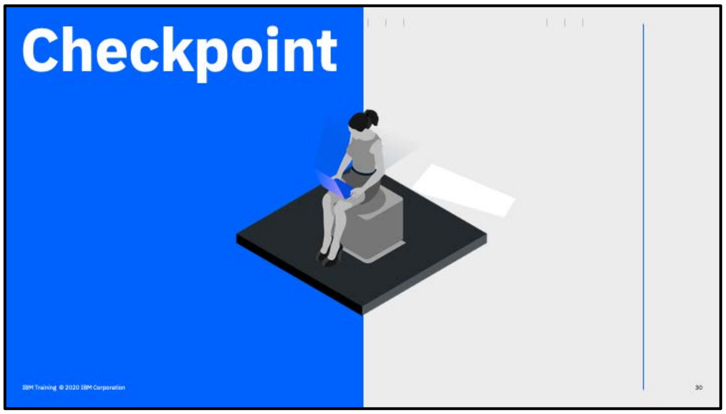
Microservice application patterns -

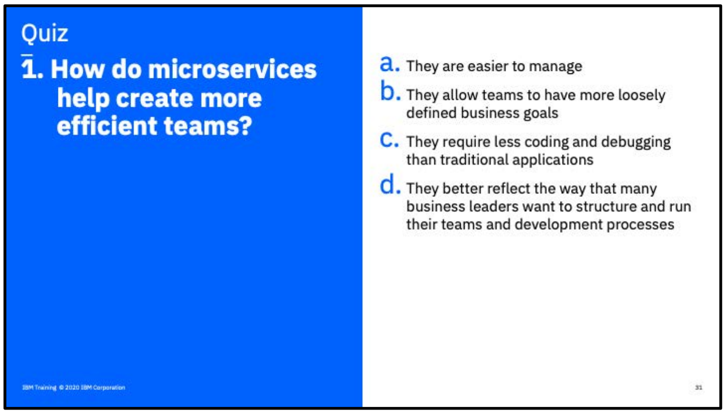
https://www.ibm.com/cloud/blog/using-microservice-application-patterns

The IBM Developer web site also provides more information about other microservices patterns here –

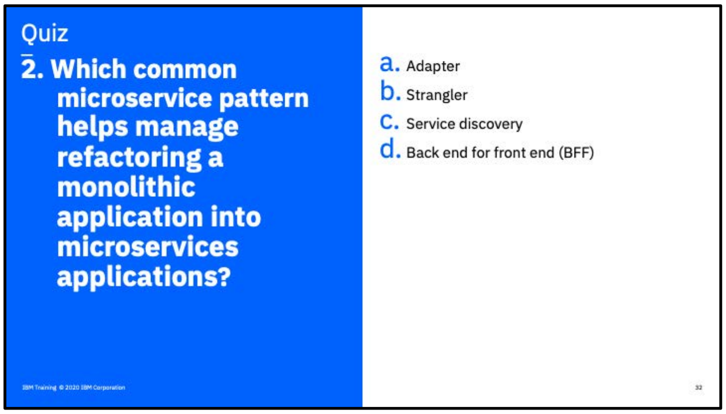
[Optional] **Learn**:

https://developer.ibm.com


Checkpoint
IBM Training © 2020 IBM Corporation
30

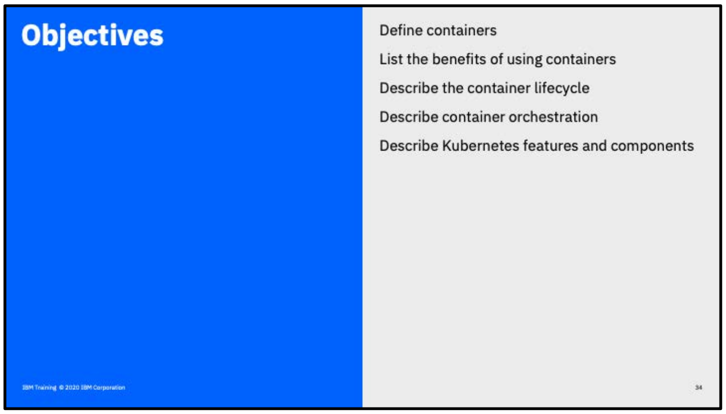


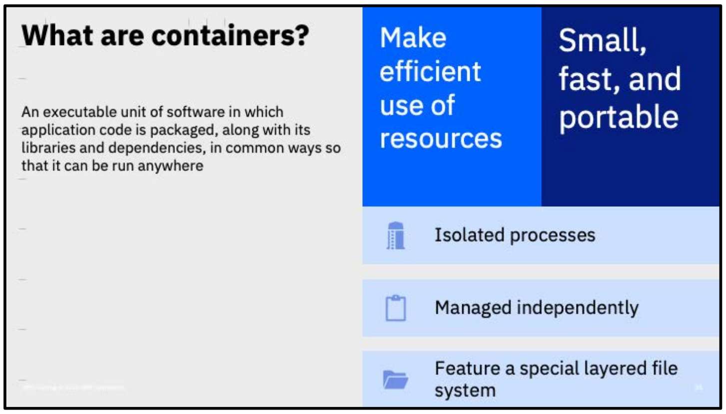
Correct answer: d.



Correct answer: b.


Containers and container orchestration
IBM Training © 2020 IBM Corporation

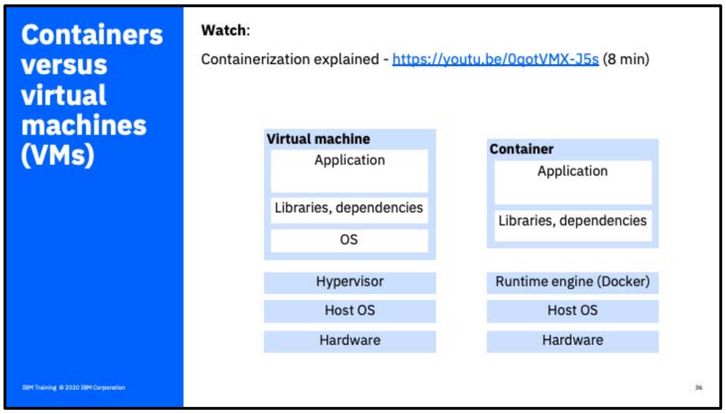

Objectives
Define containers
List the benefits of using containers
Describe the container lifecycle
Describe container orchestration
Describe Kubernetes features and components



Containers are an executable unit of software in which application code is packaged, along with its libraries and dependencies, in common ways so that it can be run anywhere, whether it be on desktop, traditional IT, or the cloud.

To do this, containers take advantage of a form of operating system (OS) virtualization in the Linux kernel namespaces and cgroups primitives are leveraged to both isolate processes and control the amount of CPU, memory, and disk that those processes have access to.

Containers are small, fast, and portable because unlike a virtual machine, containers do not need to include a guest OS in every instance and can leverage the features and resources of the host OS.



In traditional virtualization, a hypervisor virtualizes physical hardware. The result is that each virtual machine contains a guest OS, a virtual copy of the hardware that the OS requires to run, and an application and its associated libraries and dependencies.

Instead of virtualizing the underlying hardware, containers virtualize the operating system so that each individual container holds *only* the application and its libraries and dependencies. Docker is the most popular tool for creating and running Linux containers.

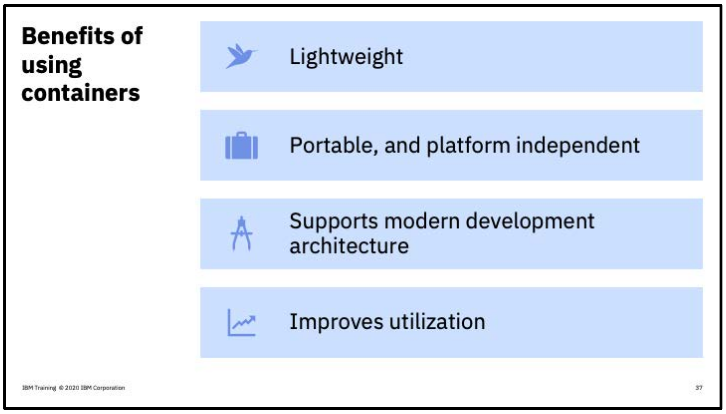
Watch:

Containerization explained - https://youtu.be/0qotVMX-J5s (8 min)

Reference:

Containers vs. VMs: What's the difference? -

https://www.ibm.com/blogs/cloud-computing/2018/10/31/containers-vs-vms-difference/



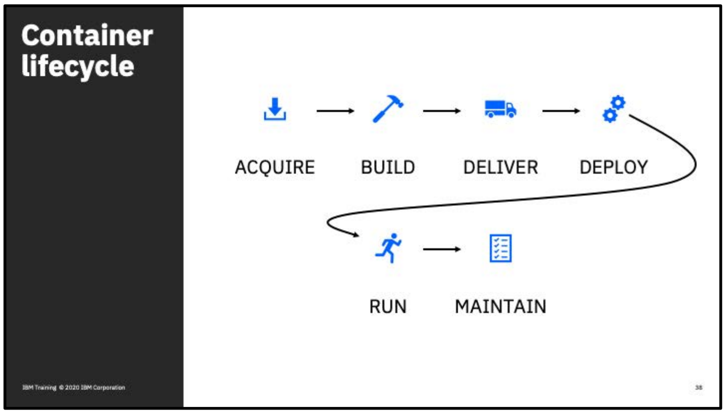
The primary advantage of containers, especially compared to a VM, is providing a level of abstraction that makes them lightweight and portable.

Lightweight: Containers share the machine OS kernel, eliminating the need for a full OS instance per application and making container files small and easy on resources. Their smaller size, especially compared to virtual machines, means they can spin up quickly and better support cloud-native applications that scale horizontally.

Portable and platform independent: Containers carry all their dependencies with them, meaning that software can be written once and then run without needing to be re-configured across laptops, cloud, and on-premises computing environments.

Supports modern development and architecture: Due to a combination of their deployment portability/consistency across platforms and their small size, containers are an ideal fit for modern development and application patterns—such as DevOps, serverless, and microservices—that are built are regular code deployments in small increments.

Improves utilization: Like VMs before them, containers enable developers and operators to improve CPU and memory utilization of physical machines. Where containers go even further is that because they also enable microservice architectures, application components can be deployed and scaled more granularly, an attractive alternative to having to scale up an entire monolithic application because a single component is struggling with load.



An effective container strategy focuses on the entire container lifecycle. Developers follow the container lifecycle through these six stages, in relation to Docker:

1. **Acquire**: The lifecycle starts with acquiring capabilities or content. One aspect of Docker is this idea of layered images. You can derive an application from a base image that is derived from another base image. As a software vendor, this allows you to deliver capabilities to users as an image, and they can extend that image in a very consistent and maintainable way. Identify the content that you want to start with and then decide where to store it, and how to access it.

2. **Build**: The second step in the lifecycle is the build. Here, you decide how to build the application and run it through the delivery pipeline.

3. **Deliver:** With the delivery step, you deliver that build into the production systems. You can automate the entire delivery pipeline across your application. You might also include a step to scan for vulnerabilities at this stage.

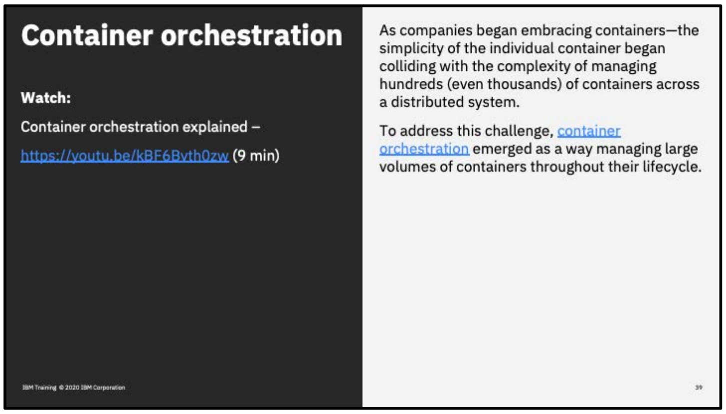
4. **Deploy:** The deploy step involves the actual process of deploying the application into production and addressing the need for updates. If you need to do continuous delivery around an application, you need a system that can orchestrate the replacement of old versions with new ones. Then, you can do canary tests to make sure that the new version behaves as expected before rolling traffic onto that new application.

5. **Run:** The run step sets the management system and the runtime environment around your container. In this step, you determine how to scale the application and how to recover from failures. You also determine how to connect the container to other services and applications in your environment.

6. **Maintain:** Finally, the maintenance step gives visibility into the application. This is how you detect failures, debug, make updates, and then connect back to the beginning to the beginning of the cycle.

Reference:

https://www.ibm.com/blogs/cloud-computing/2016/02/08/the-6-steps-of-the-container-lifecycle/



As companies began embracing containers—the simplicity of the individual container began colliding with the complexity of managing hundreds (even thousands) of containers across a distributed system.

To address this challenge, [container orchestration](https://www.ibm.com/cloud/learn/container-orchestration) emerged as a way of managing large volumes of containers throughout their lifecycle.

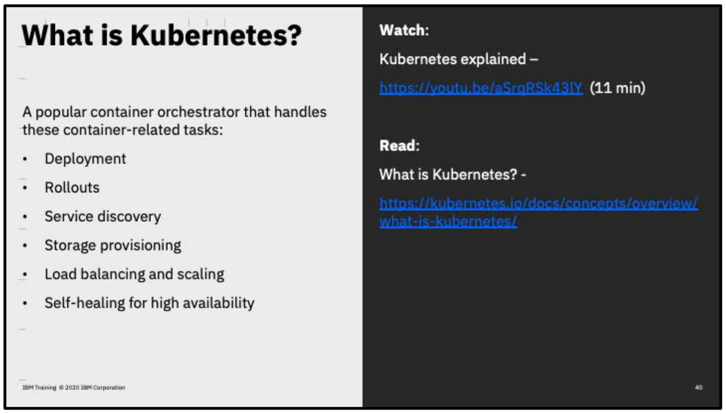
Watch:

Container orchestration explained –

https://youtu.be/kBF6Bvth0zw (9 min)

Reference:

https://www.ibm.com/cloud/blog/container-orchestration-explained



Kubernetes is a container orchestrator. It schedules and automates these and other container-related tasks:

Deployment: Kubernetes deploys a specified number of containers to a host and keeps them running in a desired state.

Rollouts: A rollout is a change to a deployment. Kubernetes lets you initiate, pause, resume, or roll back changes.

Service discovery: Kubernetes can automatically expose a container to the internet or to other containers using a DNS name or IP address.

Storage provisioning: You can set Kubernetes to mount persistent local or cloud storage for your containers as needed.

Load balancing and scaling: When traffic to a container spikes, Kubernetes can employ load balancing and scaling to distribute it across the network to maintain stability, and...

Self-healing for high availability: When a container fails, Kubernetes can restart or replace it automatically; it can also take down containers that don't meet your health-check requirements.

Developers chose (and continue to choose) Kubernetes for its breadth of functionality, its vast and growing ecosystem of open source supporting tools, and its support and portability across the leading cloud providers (some of whom now offer fully managed Kubernetes services).

Watch:

Kubernetes explained –

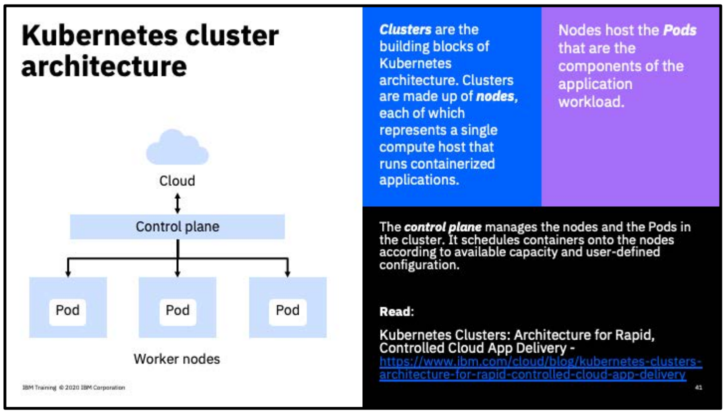
https://youtu.be/aSrqRSk43lY (11 min)

Read:

https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

Reference:

https://www.ibm.com/cloud/learn/kubernetes



When you deploy Kubernetes, you get a cluster.

Clusters are the building blocks of Kubernetes architecture. The clusters are made up of *nodes*, each of which represents a single compute host (virtual or physical machine). Every cluster has at least one node.

The ***control plane*** manages the nodes and the Pods in the cluster. It schedules containers onto the nodes according to available capacity and user-defined configuration. In production environments, the control plane usually runs across multiple computers and a cluster usually runs multiple nodes, providing fault-tolerance and high availability.

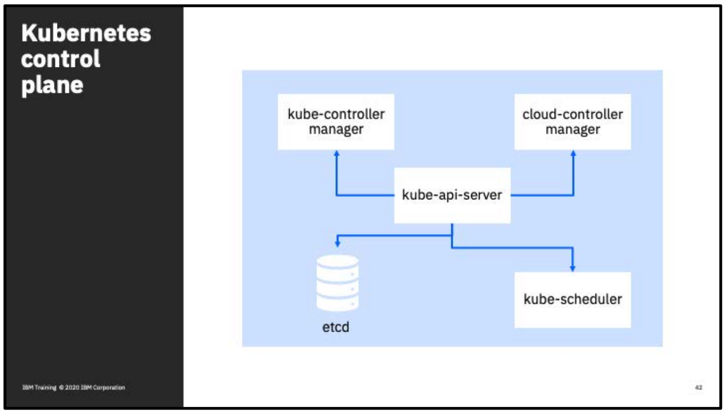
Read:

Kubernetes Clusters: Architecture for Rapid, Controlled Cloud App Delivery -

https://www.ibm.com/cloud/blog/kubernetes-clusters-architecture-for-rapid-controlled-cloud-app-delivery

Reference:

https://kubernetes.io/docs/concepts/overview/components/



The control plane components make global decisions about the cluster (for example, scheduling), and detect and respond to cluster events (for example, starting up a new pod to meet deployment requirements).

Control plane components can run on any machine in the cluster, but typically, run on a dedicated host.

The kube-apiserver is the front end of the control plane and exposes the Kubernetes API. It is designed to scale horizontally—that is, it scales by deploying more instances. You can run several instances of kube-apiserver and balance traffic between those instances.

Etcd is typically used as the backing store for cluster data.

The kube-scheduler considers available resources and determines where in the cluster to place new pods.

The kube-controller manager runs several controller processes that are compiled into a single binary. These include:

A node controller that detects and responds when a node goes down.

A replication controller that is responsible for maintaining the correct number of pods for every replication controller object in the system.

An endpoints controller that populates the Endpoints object (that is, it connects Services & Pods), and so on.

The cloud-controller-manager embeds cloud-specific control logic that links your cluster into your cloud provider's API and separates the components that interact with that cloud platform from components that just interact with your cluster.

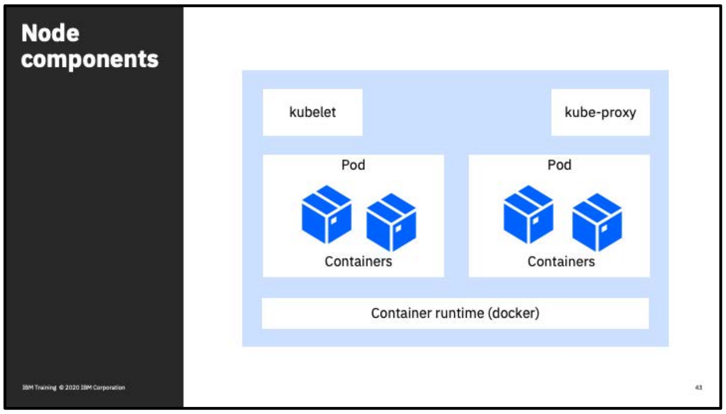
It only runs controllers that are specific to your cloud provider. If you are running Kubernetes on your own premises, or in a learning environment inside your own PC, the cluster does not have a cloud controller manager.

As with the kube-controller-manager, the cloud-controller-manager combines several logically independent control loops into a single binary that runs as a single process.

Other controllers can have cloud provider dependencies, such as the node controller, or route controller.

Reference:

https://kubernetes.io/docs/concepts/overview/components/



Node components run on every node, maintain running pods and provide the Kubernetes runtime environment. *Pods* are groups of containers that share the same compute resources and the same network. Each Pod has its own IP address and shares a PID namespace, network, and host name.

Pods are also the unit of scalability in Kubernetes: If a container in a pod is getting more traffic than it can handle, Kubernetes replicates the pod to other nodes in the cluster. For this reason, it’s a good practice to keep pods compact so that they hold only containers that must share resources.

The kubelet is an agent that runs on each node in the cluster. It ensures that containers are running in a Pod.

The kubelet takes a set of PodSpecs that are provided through various mechanisms and ensures that the containers described in those PodSpecs are running and healthy. The kubelet doesn’t manage containers which were not created by Kubernetes.

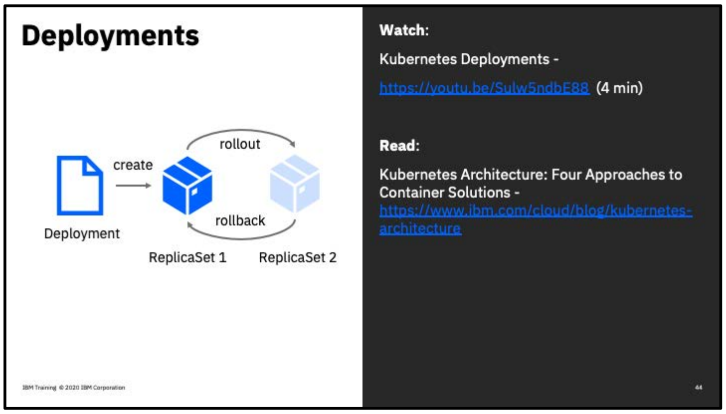
kube-proxy is a network proxy that runs on each node and implements part of the Kubernetes Service concept.

It maintains network rules on nodes. These network rules allow network communication to Pods from network sessions inside or outside of the cluster. kube-proxy uses the operating system packet filtering layer if there is one and it's available. Otherwise, kube-proxy forwards the traffic itself.

The container runtime is the software that is responsible for running containers, in this case, Docker.

Reference:

https://kubernetes.io/docs/concepts/overview/components/



The *deployment* controls the creation and state of the containerized application and keeps it running. It specifies how many replicas of a pod should run on the cluster. If a pod fails, the deployment creates a new one.

You can define a *deployment* to create a ReplicaSet. A *ReplicaSet* is a set of Pod templates that describes a set of Pod replicas. It uses a template that describes what each Pod must contain. The ReplicaSet ensures that a specified number of Pod replicas are running at any time.

You can also revise or remove deployments and adopt all their resources with new deployments.

When you revise a deployment, a ReplicaSet is created that describes the state that you want. During a rollout, the deployment controller changes the actual state to the state that you want at a controlled rate. Each deployment revision can also be rolled back. Deployments can also be scaled.

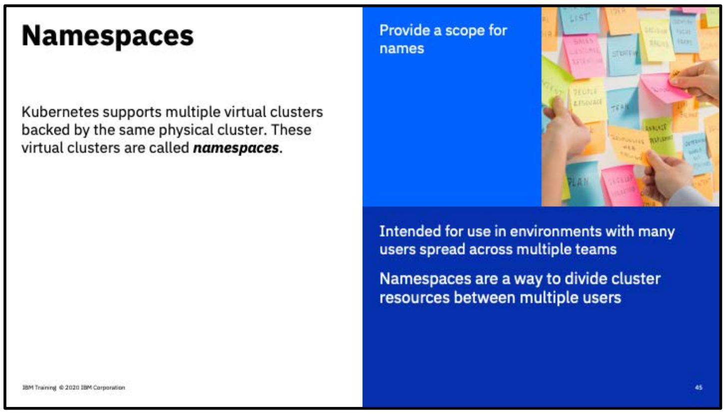
Watch:

Kubernetes Deployments -

https://youtu.be/Sulw5ndbE88 (4 min)

Read:

Kubernetes Architecture: Four Approaches to Container Solutions -
https://www.ibm.com/cloud/blog/kubernetes-architecture



Namespaces are intended for use in environments with many users spread across multiple teams, or projects. For clusters with a few to tens of users, you probably do not need to create namespaces. Start using namespaces when you need the features they provide.

Namespaces provide a scope for names.

Kubernetes supports multiple virtual clusters backed by the same physical cluster. These virtual clusters are called namespaces.

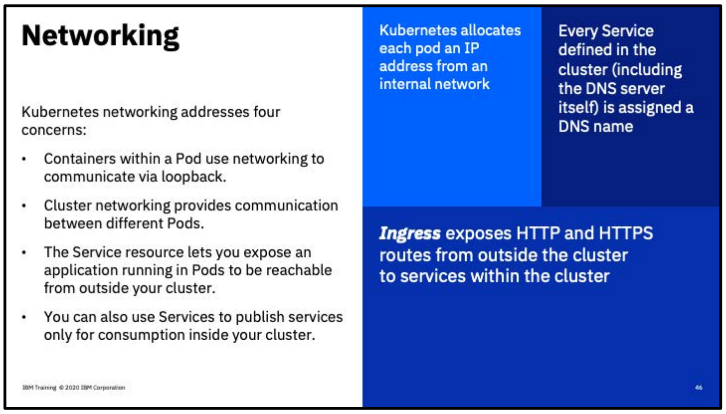
Names of resources must be unique within a namespace, but not across all namespaces. Namespaces can not be nested inside one another and each Kubernetes resource can only belong to one namespace.

Namespaces are a way to divide cluster resources between multiple users (via resource quota).

It is not necessary to use multiple namespaces just to separate slightly different resources, such as different versions of the same software: instead, you can use labels to distinguish resources within the same namespace.

Reference:

https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces



Kubernetes networking addresses four concerns:

- Containers within a Pod use networking to communicate via loopback.
- Cluster networking provides communication between different Pods.
- The Service resource lets you expose an application running in Pods to be reachable from outside your cluster.
- You can also use Services to publish services only for consumption inside your cluster.

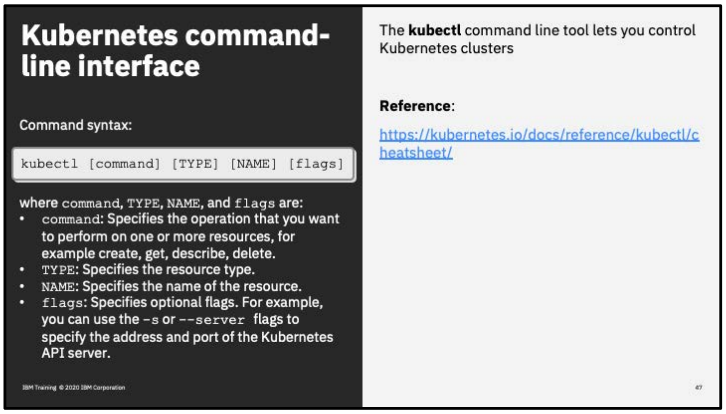
Kubernetes ensures that Pods can network with each other and allocates each Pod an IP address from an internal network. This ensures all containers within the Pod behave as if they were on the same host. Giving each Pod its own IP address means that Pods can be treated like physical hosts or virtual machines in terms of port allocation, networking, naming, service discovery, load balancing, application configuration, and migration.

Kubernetes DNS schedules a DNS Pod and Service on the cluster and configures the kubelets to tell individual containers to use the DNS Service's IP to resolve DNS names. Every Service defined in the cluster (including the DNS server itself) is assigned a DNS name. By default, a client Pod's DNS search list includes the Pod's own namespace and the cluster's default domain.

Ingress exposes HTTP and HTTPS routes from outside the cluster to services within the cluster. A service is an abstract way to expose an application running on a set of Pods as a network service.

Reference:

https://kubernetes.io/docs/concepts/services-networking/

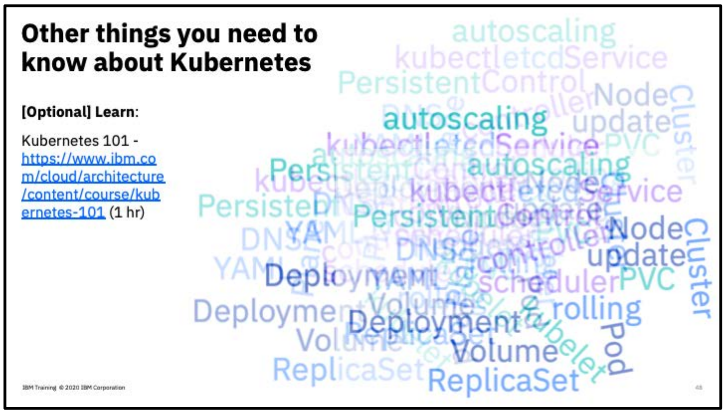


The kube-c-t-l command line tool lets you control Kubernetes clusters. “ctl” stands for control. [There are a few pronunciations that I have come across for kubectl, such as “kube control”, “kube cuddle”, kub-ec-tl – you get the idea.]

This overview covers command syntax, describes the command operations, and provides common examples. For details about each command, including all the supported flags and subcommands, see the kubectl reference documentation.

Reference:

https://kubernetes.io/docs/reference/kubectl/cheatsheet/



Kubernetes is a big topic. Some other things that you need to know about, for example, are:

YAML, which is a human-readable serialization language that you can use to directly manipulate Kubernetes resources.

Services - A *Service* is a collection of Pods that arre exposed as an endpoint. The Service propagates state and networking information to all worker nodes.

DNS and name discovery –

Common kubectl commands –

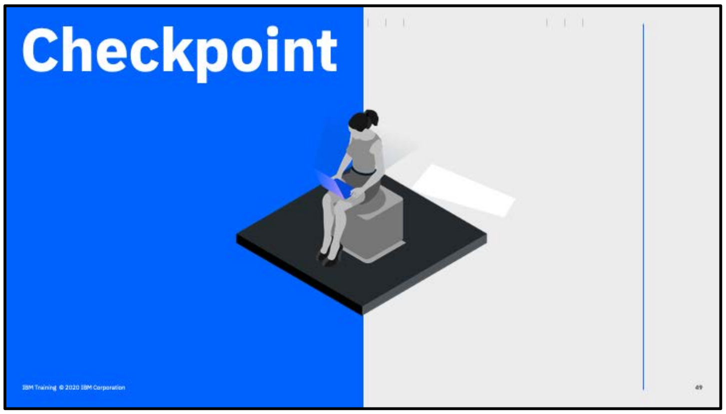
Persistent Volumes, Persistent Volume Claims, and

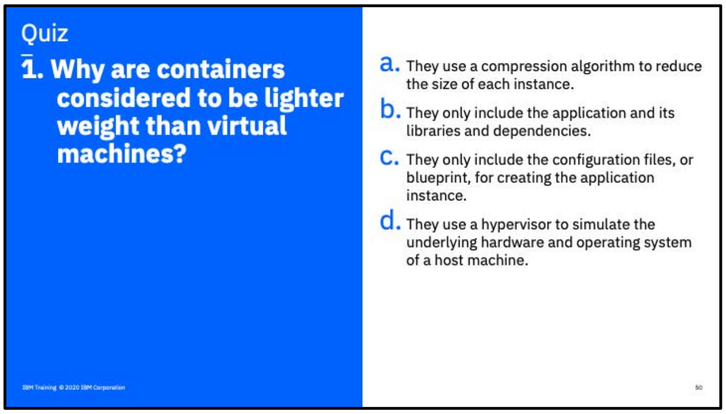
How it all connects and works together.

These topics are all covered in the course -

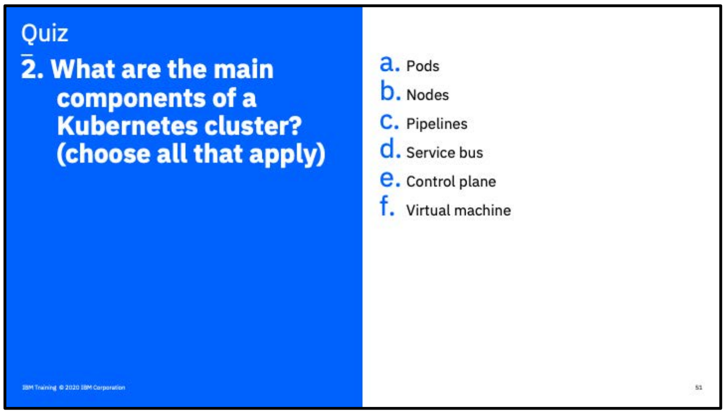
Kubernetes 101 - https://www.ibm.com/cloud/architecture/content/course/kubernetes-101 (1 hour)

It's important to have a good understanding of how Kubernetes works before moving on in your cloud journey, so take some time to check it out.


Checkpoint



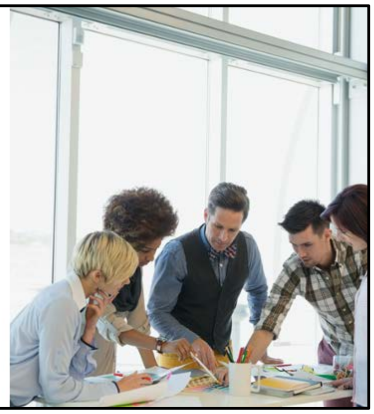
Correct answer: b.

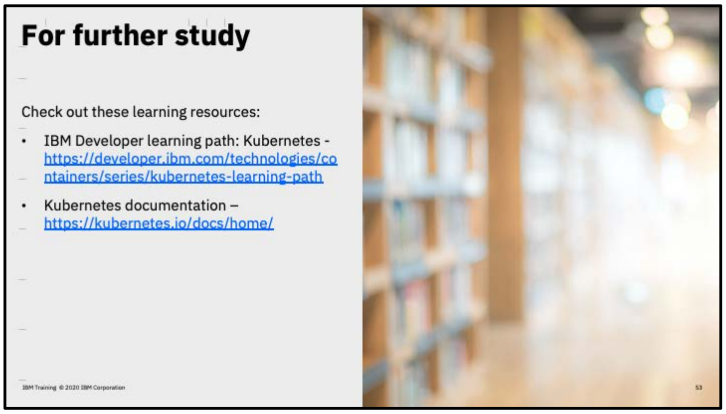


Correct answers: a, b, e.

Discussion prompt:

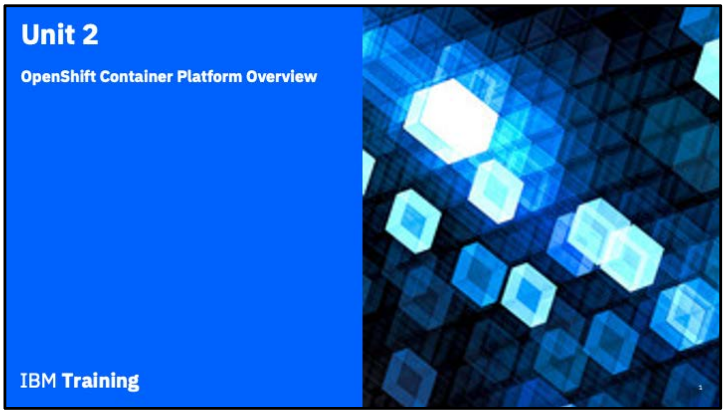
What are the main challenges to your organization adopting a cloud native approach?





Check out these learning resources:

- IBM Developer learning path: Kubernetes - https://developer.ibm.com/technologies/containers/series/kubernetes-learning-path
- Kubernetes documentation – https://kubernetes.io/docs/home/



Unit 2 OpenShift Container Platform Overview

This unit introduces the Red Hat OpenShift Container Platform. It describes what you need to know to get started with using it with Cloud Pak for Applications and provides resources for further study.

Presentation time: about 1 hour

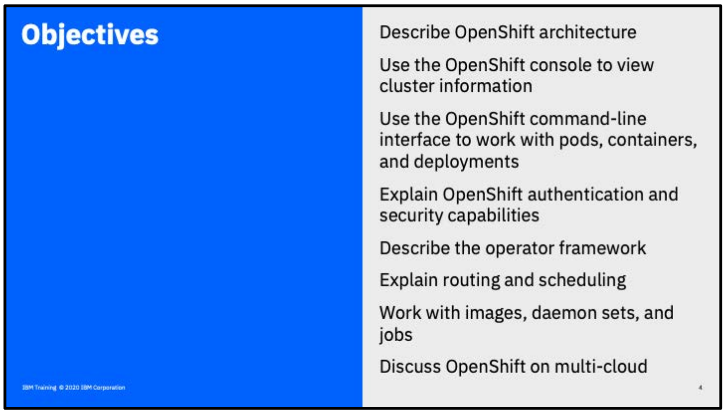
Supplemental material:

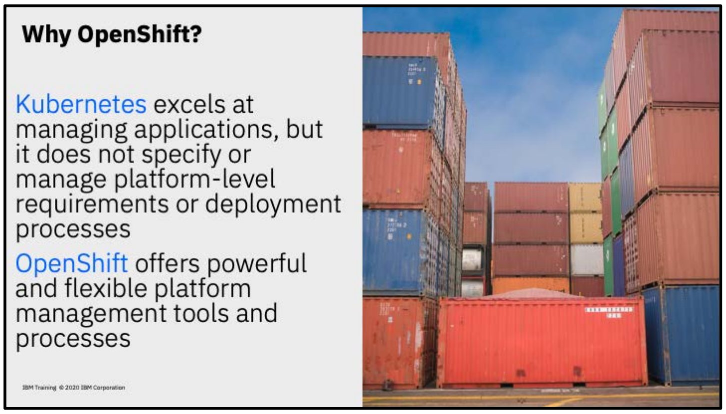
Videos: 36 minutes

Readings: 2 hours

Topics

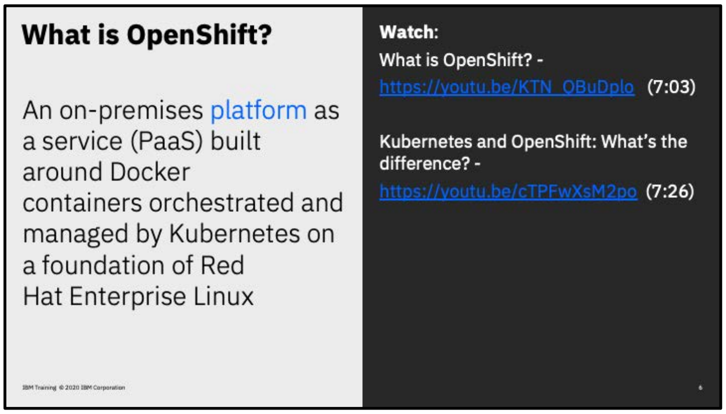

OpenShift
Architecture
and
capabilities
IBM Training © 2020 IBM Corporation


Objectives
Describe OpenShift architecture
Use the OpenShift console to view cluster information
Use the OpenShift command-line interface to work with pods, containers, and deployments
Explain OpenShift authentication and security capabilities
Describe the operator framework
Explain routing and scheduling
Work with images, daemon sets, and jobs
Discuss OpenShift on multi-cloud



Although Kubernetes excels at managing applications, it does not specify or manage platform-level requirements or deployment processes. OpenShift Container Platform offers powerful and flexible platform management tools and processes, and these are important benefits.

Note: This course and the associated IBM Professional Certification exam is based on IBM Cloud Pak for Applications V4.1, which runs on OpenShift Container Platform V4.2. At the time of publishing this course, the latest version available was IBM Cloud Pak for Applications V4.2, which runs on OpenShift Container Platform V4.4. Any major differences between versions are identified where possible.



OpenShift is an on-premises **platform** as a service built around **Docker containers** orchestrated and managed by Kubernetes on a foundation of **Red Hat** Enterprise Linux.

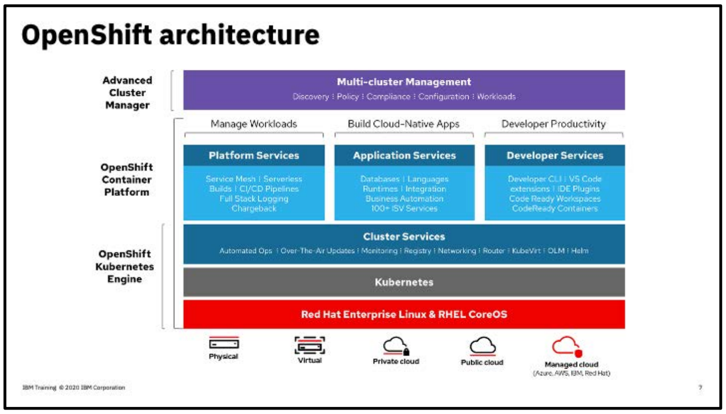
Watch:

What is OpenShift? -

https://youtu.be/KTN_QBuDplo (7:03)

Kubernetes and OpenShift: What's the difference? -

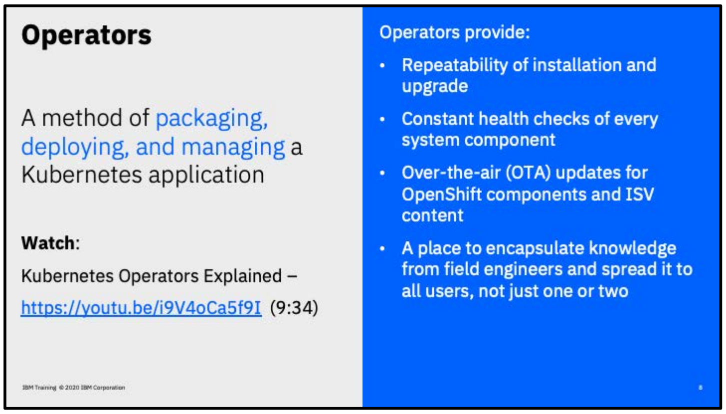
https://youtu.be/cTPFwXsM2po (7:26)



OpenShift Container Platform provides enterprise-ready enhancements to Kubernetes, including the following enhancements:

- Hybrid cloud deployments. You can deploy OpenShift Container Platform clusters to variety of public cloud platforms or in your data center.
- Integrated Red Hat technology. Major components in OpenShift Container Platform come from Red Hat Enterprise Linux and related Red Hat technologies. OpenShift Container Platform benefits from the intense testing and certification initiatives for Red Hat's enterprise quality software.
- Open source development model. Development is completed in the open, and the source code is available from public software repositories. This open collaboration fosters rapid innovation and development.

OpenShift Container Platform uses Red Hat Enterprise Linux CoreOS (RHCOS), a container-oriented operating system that combines some of the best features and functions of the CoreOS and Red Hat Atomic Host operating systems.



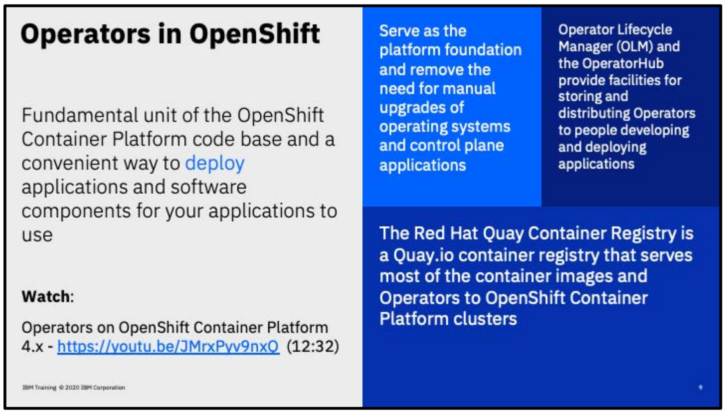
Operators are a method of packaging, deploying, and managing a Kubernetes application.

A Kubernetes application is an app that is both deployed on Kubernetes and managed using the Kubernetes APIs and kubectl or oc tooling. To be able to make the most of Kubernetes, you require a set of cohesive APIs to extend in order to service and manage your apps that run on Kubernetes. Think of Operators as the runtime that manages this type of app on Kubernetes.

Watch:

Kubernetes Operators Explained –

https://youtu.be/i9V4oCa5f9I (9:34)



Operators are both the fundamental unit of the OpenShift Container Platform code base and a convenient way to deploy applications and software components for your applications to use. In OpenShift Container Platform, Operators serve as the platform foundation and remove the need for manual upgrades of operating systems and control plane applications. OpenShift Container Platform Operators such as the Cluster Version Operator and Machine Config Operator allow simplified, cluster-wide management of those critical components.

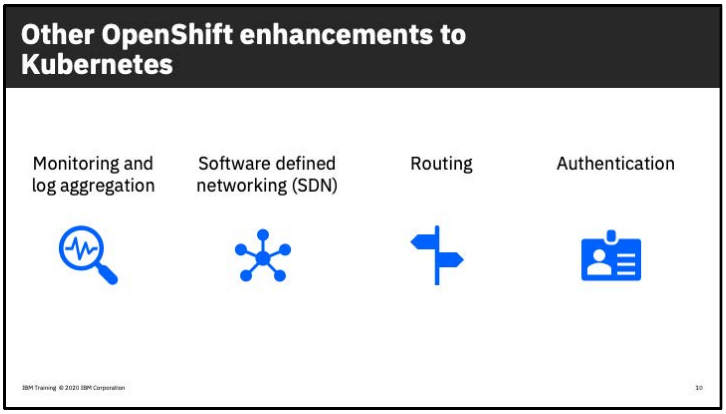
Operator Lifecycle Manager (OLM) and the OperatorHub provide facilities for storing and distributing Operators to people developing and deploying applications.

The Red Hat Quay Container Registry is a Quay.io container registry that serves most of the container images and Operators to OpenShift Container Platform clusters. Quay.io is a public registry version of Red Hat Quay that stores millions of images and tags.

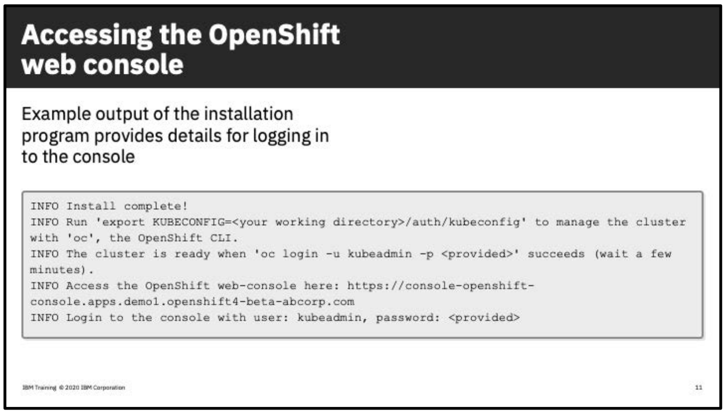
Watch:

Operators on OpenShift Container Platform 4.x

https://youtu.be/JMrxPyv9nxQ (12:32)



Other enhancements to Kubernetes in OpenShift Container Platform include improvements in software defined networking (SDN), authentication, log aggregation, monitoring, and routing. OpenShift Container Platform also offers a comprehensive web console and the custom OpenShift CLI (oc) interface.



The OpenShift Container Platform web console is a user interface accessible from a web browser. You can use the web console to visualize, browse, and manage the contents of projects.

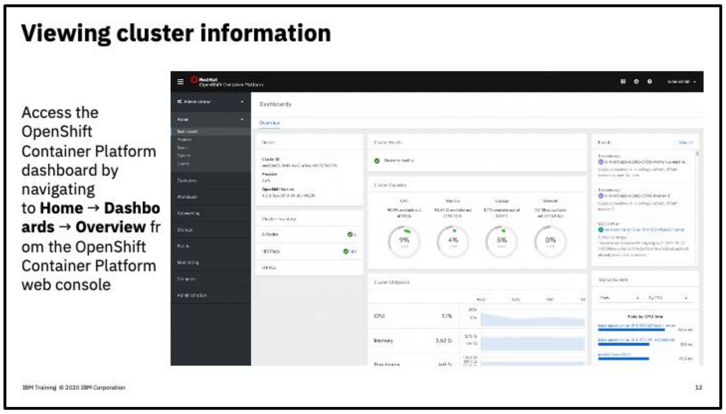
The web console runs as a pod on the control plane. The static assets required to run the web console are served by the pod.

OpenShift Container Platform creates a cluster administrator, kubeadmin, after the installation process completes.

This user has the cluster-admin role automatically applied and is treated as the root user for the cluster. The password is dynamically generated and unique to your OpenShift Container Platform environment. After installation completes, the password is provided in the installation program's output, as shown in the example here.

Reference:

https://docs.openshift.com/container-platform/4.2/web_console/web-console.html



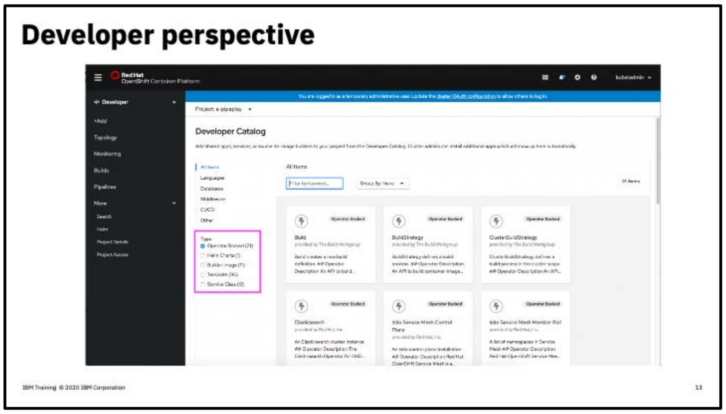
Access the OpenShift Container Platform dashboard, which captures high-level information about the cluster, by navigating to **Home** → **Dashboards** → **Overview** from the OpenShift Container Platform web console.

The OpenShift Container Platform dashboard provides various cluster information, captured in individual dashboard cards.

You can also configure and customize the console in various ways.

Reference:

https://docs.openshift.com/container-platform/4.2/web_console/web-console.html



The OpenShift Container Platform web console provides two perspectives; the **Administrator** perspective and the **Developer** perspective.

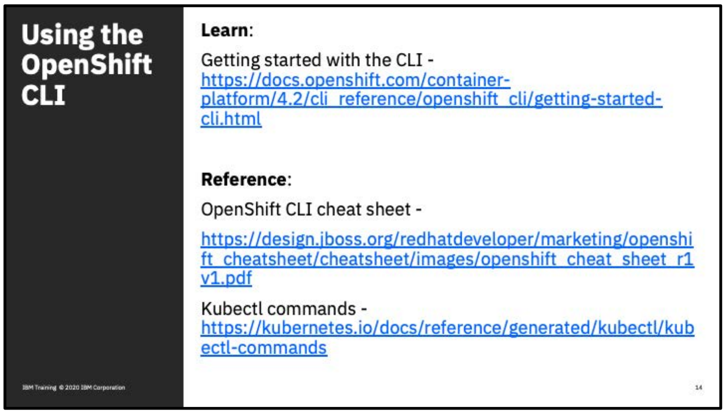
The **Developer** perspective provides workflows specific to developer use cases, such as the ability to:

- Create and deploy applications on OpenShift Container Platform by importing existing codebases, images, and dockerfiles.
- Visually interact with applications, components, and services associated with them within a project and monitor their deployment and build status.
- Group components within an application and connect the components within and across applications.
- Integrate serverless capabilities (Technology Preview).
- Create workspaces to edit your application code using Eclipse Che.

Reference:

https://docs.openshift.com/container-platform/4.2/web_console/odc-about-developer-perspective.html

[You can also use the command-line interface to perform most developer and administrative tasks.]



The link here provides an introduction to the OpenShift command-line interface.

This cheat sheet is a guide to basic and frequently used commands.

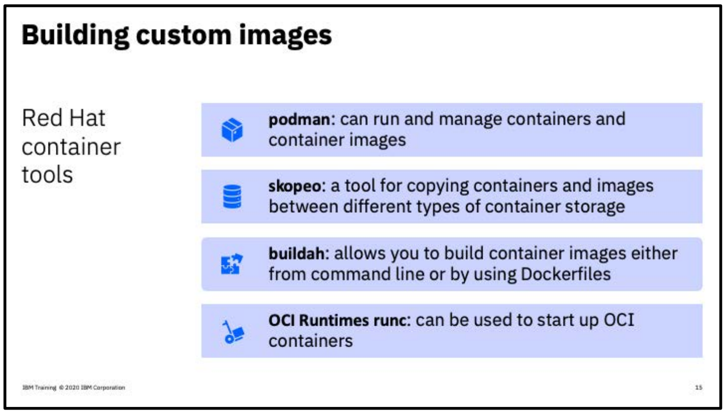
You can also use the Kubernetes CLI, kubectl, to work directly with Kubernetes resources, and you can use docker or podman commands to work directly with images, registries, and containers.

Reference:

OpenShift CLI cheat sheet -

https://design.jboss.org/redhatdeveloper/marketing/openshift_cheatsheet/cheatsheet/images/openshift_cheat_sheet_r1v1.pdf

Kubectl commands - https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands



Red Hat Enterprise Linux offers a set of container tools to work directly with Linux containers and container images that requires no container engine or docker commands or services. These tools include:

podman: The podman command can run and manage containers and container images. It supports the same features and command options you find in the dockercommand, with the main differences being that podman doesn't require the docker service or any other active container engine for the command to work. Also, podmanstores its data in the same directory structure used by Buildah, Skopeo, and CRI-O, which will allow podman to eventually work with containers being actively managed by CRI-O in OpenShift.

Podman has a lot of advanced features, such as the support for running containers in Pods. It fully integrates with systemd, including the ability to generate unit files from containers and run systemd within a container. Podman also offers User Namespace support, including running containers without requiring root.

skopeo: The skopeo command is a tool for copying containers and images between different types of container storage. It can copy containers from one container registry to another. It can copy images to and from a host, as well as to other container environments and registries. Skopeo can inspect images from container image registries, get images and image layers, and use signatures to create and verify images.

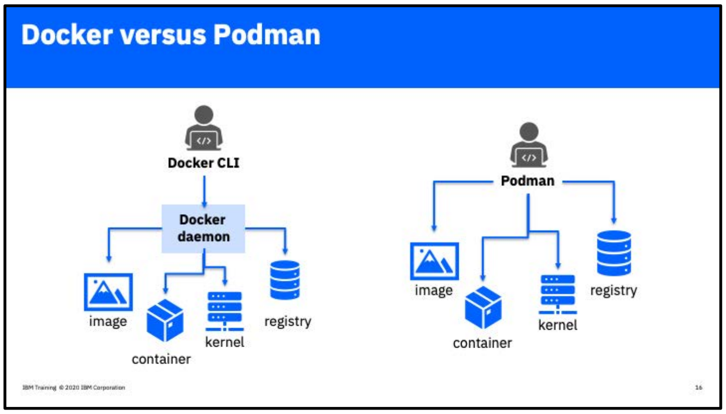
buildah: The buildah command allows you to build container images either from command line or using Dockerfiles. These images can then be pushed to any container registry and can be used by any container engine, including Podman, CRI-O, and Docker. The buildah command can be used as a separate command but is incorporated into other tools as well. For example the podman build command used buildah code to build container images. Buildah is also often used to securely build containers while running inside of a locked down container by a tool like Podman, OpenShift/Kubernetes or Docker.

OCI Runtimes:

runc: The runc command can be used to start up OCI containers.

Reference:

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux_atomic_host/7/html/managing_containers/finding_running_and_building_containers_with_podman_skopeo_and_buildah



How Docker works is it runs a daemon process to service all of your Docker commands and does all the work with registries, images, containers, and the kernel. The Docker command-line interface (CLI) asks the daemon to do this on your behalf.

The Podman approach is simply to directly interact with the image registry, with the container and image storage, and with the Linux kernel through the runC container runtime process (which is.not a daemon).

You install Podman instead of Docker. You do not need to start or manage a daemon process like the Docker daemon.

The commands you are familiar with in Docker work the same for Podman.

Podman stores its containers and images in a different place than Docker.

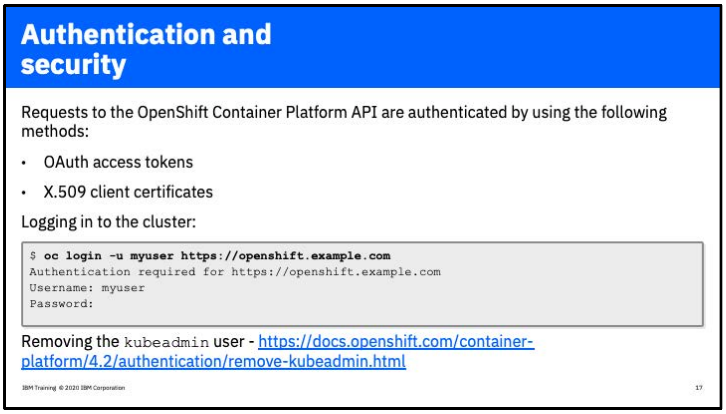
Podman and Docker images are compatible.

Podman does more than Docker for Kubernetes environments.

Reference:

https://developers.redhat.com/blog/2019/02/21/podman-and-buildah-for-docker-users/

[The next few slides go through some important concepts and tasks that you must be familiar with to work with Cloud Pak for Applications on OpenShift, but it is not an exhaustive list. Check out the links to these topics in the documentation for more details.]



For users to interact with OpenShift Container Platform, they must first authenticate to the cluster. The authentication layer identifies the user associated with requests to the OpenShift Container Platform API. The authorization layer then uses information about the requesting user to determine if the request is allowed.

Requests to the OpenShift Container Platform API are authenticated by using the following methods:

- OAuth access tokens
- X.509 client certificates

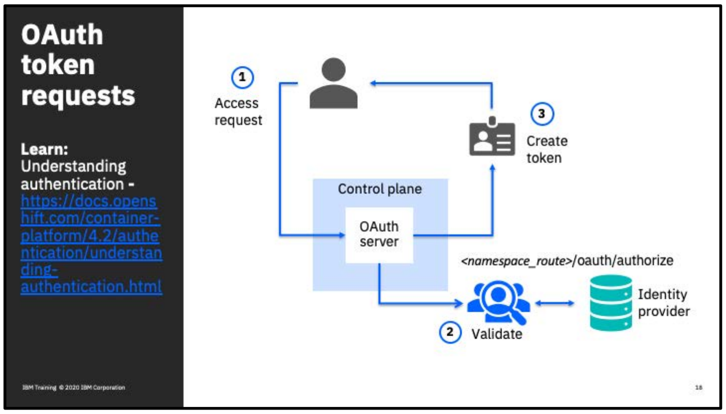
Here is an example of logging in with the OpenShift command line interface.

As mentioned previously, OpenShift Container Platform creates a cluster administrator, kubeadmin, after the installation process completes. After you define an identity provider and create a new cluster-admin user, you can remove the kubeadmin to improve cluster security. You must have configured at least one identity provider and added the cluster-admin role to the user.

Read:

Removing the `kubeadmin` user -

https://docs.openshift.com/container-platform/4.2/authentication/remove-kubeadmin.html



The OpenShift Container Platform control plane includes a built-in OAuth server. Users obtain OAuth access tokens to authenticate themselves to the API.

When a person requests a new OAuth token, the OAuth server uses the configured identity provider to determine the identity of the person making the request.

It then determines what user that identity maps to, creates an access token for that user, and returns the token for use.

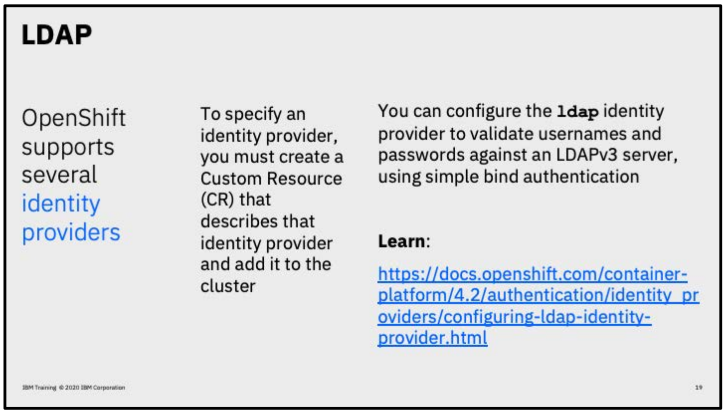
Every request for an OAuth token must specify the OAuth client that will receive and use the token. The following OAuth clients are automatically created when starting the OpenShift Container Platform API:

- openshift-browser-client: Requests tokens at *<namespace_route>*/oauth/token/request with a user-agent that can handle interactive logins.
- openshift-challenging-client: Requests tokens with a user-agent that can handle WWW-Authenticate challenges.

All requests for OAuth tokens involve a request to *<namespace_route>*/oauth/authorize. Most authentication integrations place an authenticating proxy in front of this endpoint, or configure OpenShift Container Platform to validate credentials against a backing identity provider. Requests to *<namespace_route>*/oauth/authorize can come from user-agents that cannot display interactive login pages, such as the CLI. Therefore, OpenShift Container Platform supports authenticating using a WWW-Authenticate challenge in addition to interactive login flows.

Reference:

https://docs.openshift.com/container-platform/4.2/authentication/understanding-authentication.html



You can configure OAuth to specify an identity provider after you install your cluster. To specify an identity provider, you must create a Custom Resource (CR) that describes that identity provider and add it to the cluster. OpenShift supports several identity providers. A common one is LDAP.

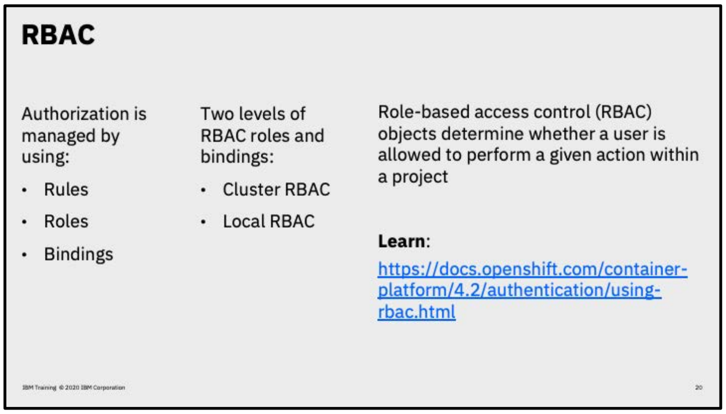
You can configure the ldap identity provider to validate user names and passwords against an LDAPv3 server, using simple bind authentication.

Learn:

Details here - https://docs.openshift.com/container-platform/4.2/authentication/identity_providers/configuring-ldap-identity-provider.html

Reference:

https://docs.openshift.com/container-platform/4.2/authentication/understanding-identity-provider.html



Role-based access control (RBAC) objects determine whether a user is allowed to perform a given action within a project.

Cluster administrators can use the cluster roles and bindings to control who has various access levels to the OpenShift Container Platform platform itself and all projects.

Developers can use local roles and bindings to control who has access to their projects. Note that authorization is a separate step from authentication, which is more about determining the identity of who is taking the action.

A cluster role binding is a binding that exists at the cluster level. A role binding exists at the project level. The cluster role *view* must be bound to a user using a local role binding for that user to view the project. Create local roles only if a cluster role does not provide the set of permissions needed for a situation.

This two-level hierarchy allows reuse across multiple projects through the cluster roles while allowing customization inside of individual projects through local roles.

During evaluation, both the cluster role bindings and the local role bindings are used. For example:

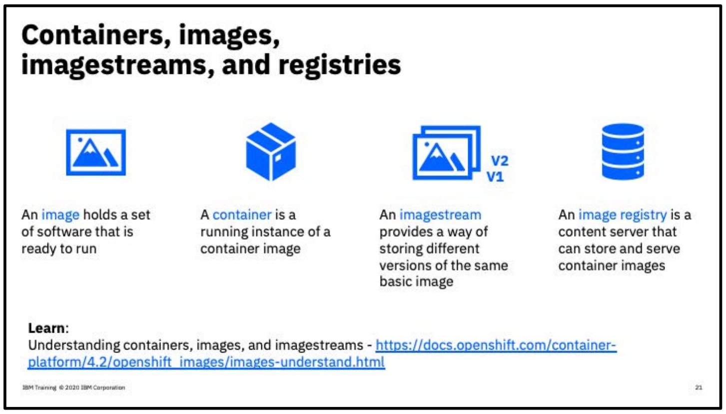
1. Cluster-wide "allow" rules are checked.
2. Locally-bound "allow" rules are checked.
3. Deny by default.

Learn:

https://docs.openshift.com/container-platform/4.2/authentication/using-rbac.html

Reference:

https://kubernetes.io/docs/reference/access-authn-authz/rbac/



Containers, images, and imagestreams are important concepts to understand when you set out to create and manage containerized software. An image holds a set of software that is ready to run, while a container is a running instance of a container image. An imagestream provides a way of storing different versions of the same basic image. Those different versions are represented by different tags on the same image name.

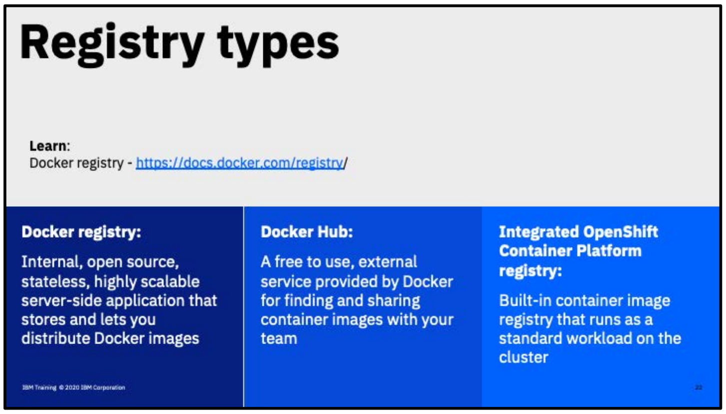
An **image repository** is a collection of related container images and tags identifying them.

A registry contains a collection of one or more image repositories, which contain one or more tagged images. Red Hat provides a registry at registry.redhat.io for subscribers. OpenShift Container Platform can also supply its own internal registry for managing custom container images.

An image tag is a label applied to a container image in a repository that distinguishes a specific image from other images in an imagestream. Typically, the tag represents a version number of some sort.

Learn:

Understanding containers, images, and imagestreams - https://docs.openshift.com/container-platform/4.2/openshift_images/images-understand.html



The Docker Registry is a stateless, highly scalable server-side application that stores and lets you distribute Docker images. The Registry is open-source, under the permissive Apache license.

You should use the Registry if you want to:

- tightly control where your images are being stored
- fully own your images distribution pipeline
- integrate image storage and distribution tightly into your in-house development workflow

While it’s highly recommended to secure your registry using a TLS certificate issued by a known CA, you can choose to use self-signed certificates, or use your registry over an unencrypted HTTP connection. Either of these choices involves security trade-offs and additional configuration steps.

OpenShift Container Platform provides a built-in container image registry that runs as a standard workload on the cluster. The registry is typically used as a publication target for images built on the cluster as well as a source of images for workloads running on the cluster. When a new image is pushed to the registry, the cluster is notified of the new image and other components can react to and consume the updated image.

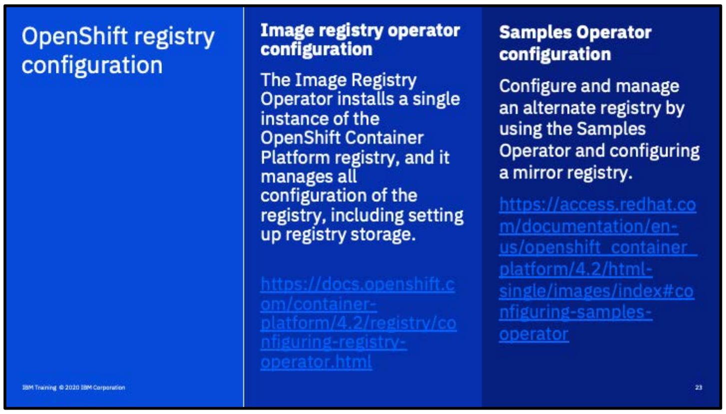
If you are looking for a zero maintenance, ready-to-go solution, consider using the Docker Hub, which provides a free-to-use, hosted Registry, plus additional features.

Learn:

Docker registry - https://docs.docker.com/registry/

Reference:

https://docs.docker.com/docker-hub/



The Image Registry Operator installs a single instance of the OpenShift Container Platform registry, and it manages all configuration of the registry, including setting up registry storage.

Or, you can configure and manage an alternate registry by using the Samples Operator and configuring a mirror registry.

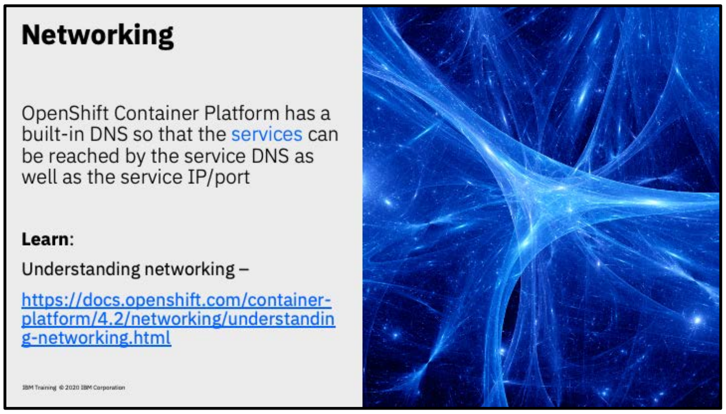
Learn:

Configuring the registry operator -

https://docs.openshift.com/container-platform/4.2/registry/configuring-registry-operator.html

Configuring the sample operator -

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.2/html-single/images/index#configuring-samples-operator



OpenShift Container Platform DNS

If you are running multiple services, such as front-end and back-end services for use with multiple Pods, environment variables are created for user names, service IPs, and more so the front-end Pods can communicate with the back-end services. If the service is deleted and recreated, a new IP address can be assigned to the service, and requires the front-end Pods to be recreated to pick up the updated values for the service IP environment variable. Additionally, the back-end service must be created before any of the front-end Pods to ensure that the service IP is generated properly, and that it can be provided to the front-end Pods as an environment variable.

For this reason, OpenShift Container Platform has a built-in DNS so that the services can be reached by the service DNS as well as the service IP/port.

The DNS Operator deploys and manages CoreDNS to provide a name resolution service to pods, enabling DNS-based Kubernetes Service discovery in OpenShift.

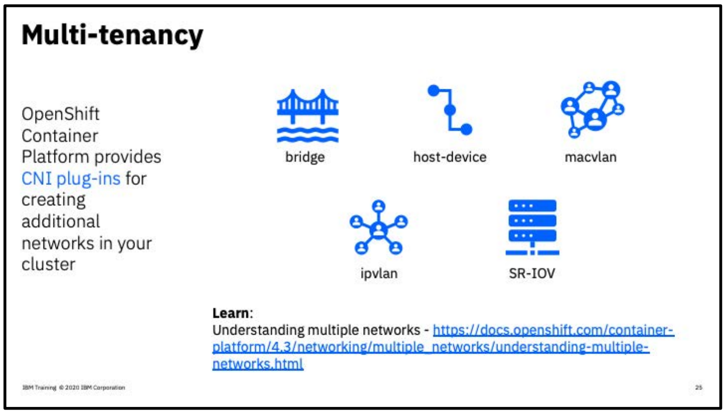
Learn:

Understanding networking –

https://docs.openshift.com/container-platform/4.2/networking/understanding-networking.html

Reference:

https://docs.openshift.com/container-platform/4.3/networking/dns-operator.html



In Kubernetes, container networking is delegated to networking plug-ins that implement the Container Network Interface (CNI).

OpenShift Container Platform uses the Multus CNI plug-in to allow chaining of CNI plug-ins. During cluster installation, you configure your *default* Pod network. The default network handles all ordinary network traffic for the cluster. You can define an *additional network* based on the available CNI plug-ins and attach one or more of these networks to your Pods. You can define more than one additional network for your cluster, depending on your needs. This gives you flexibility when you configure Pods that deliver network functionality, such as switching or routing.

You can use an additional network in situations where network isolation is needed, including data plane and control plane separation. Isolating network traffic is useful for performance and security reasons.

OpenShift Container Platform provides the following CNI plug-ins for creating additional networks in your cluster:

bridge: Creating a bridge-based additional network allows Pods on the same host to communicate with each other and the host.

host-device: Creating a host-device additional network allows Pods access to a physical Ethernet network device on the host system.

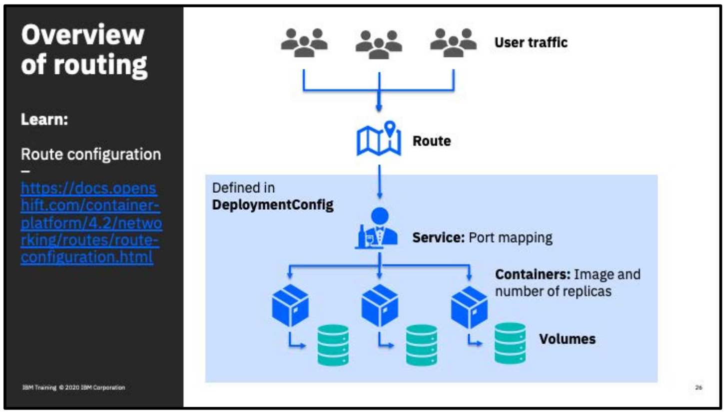
macvlan: Creating a macvlan-based additional network allows Pods on a host to communicate with other hosts and Pods on those hosts by using a physical network interface. Each Pod that is attached to a macvlan-based additional network is provided a unique MAC address.

ipvlan: Creating an ipvlan-based additional network allows Pods on a host to communicate with other hosts and Pods on those hosts, similar to a macvlan-based additional network. Unlike a macvlan-based additional network, each Pod shares the same MAC address as the parent physical network interface.

SR-IOV: Creating a SR-IOV based additional network allows Pods to attach to a virtual function (VF) interface on SR-IOV capable hardware on the host system.

Learn:

Understanding multiple networks - https://docs.openshift.com/container-platform/4.3/networking/multiple_networks/understanding-multiple-networks.html



You can expose a service to create a route for your application, configure route timeouts and configure secure routes by using a custom certificate.

Learn:
Route configuration –
https://docs.openshift.com/container-platform/4.2/networking/routes/route-configuration.html

OpenShift Container Platform provides the following methods for communicating from outside the cluster with services running in the cluster.

The methods are recommended, in order or preference:

- If you have HTTP/HTTPS, use an Ingress Controller.
- If you have a TLS-encrypted protocol other than HTTPS. For example, for TLS with the SNI header, use an Ingress Controller.
- Otherwise, use a Load Balancer, an External IP, or a NodePort.

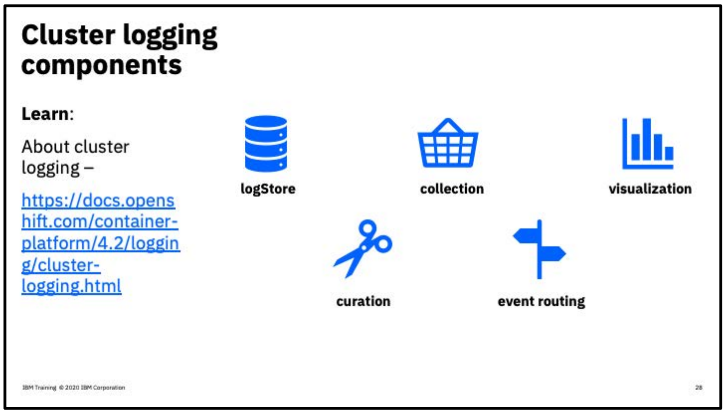
Learn:

Configuring ingress cluster traffic –

https://docs.openshift.com/container-platform/4.2/networking/configuring_ingress_cluster_traffic/overview-traffic.html

Reference:

https://kubernetes.io/docs/concepts/services-networking/ingress/



As a cluster administrator, you can deploy cluster logging to aggregate all the logs from your OpenShift Container Platform cluster, such as node system logs, application container logs, and so forth.

You can install cluster logging by deploying the Elasticsearch and Cluster Logging Operators. The Elasticsearch Operator creates and manages the Elasticsearch cluster used by cluster logging. The Cluster Logging Operator creates and manages the components of the logging stack.

You can view OpenShift Container Platform cluster logs in the CLI or OpenShift Container Platform web console.

The cluster logging components are based upon Elasticsearch, Fluentd, and Kibana (EFK). The collector, Fluentd, is deployed to each node in the OpenShift Container Platform cluster. It collects all node and container logs and writes them to Elasticsearch (ES). Kibana is the centralized, web UI where users and administrators can create rich visualizations and dashboards with the aggregated data.

There are currently 5 different types of cluster logging components:

logStore - This is where the logs will be stored. The current implementation is Elasticsearch.

collection - This is the component that collects logs from the node, formats them, and stores them in the logStore. The current implementation is Fluentd.

visualization - This is the UI component used to view logs, graphs, charts, and so forth. The current implementation is Kibana.

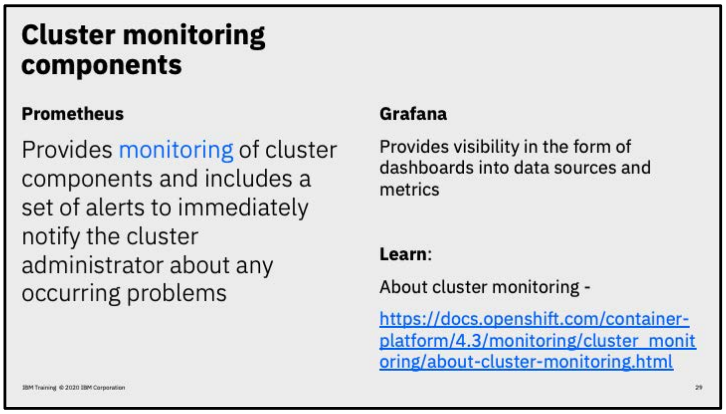
curation - This is the component that trims logs by age. The current implementation is Curator.

event routing - This is the component forwards OpenShift Container Platform events to cluster logging. The current implementation is Event Router.

Learn:

About cluster logging –

https://docs.openshift.com/container-platform/4.2/logging/cluster-logging.html

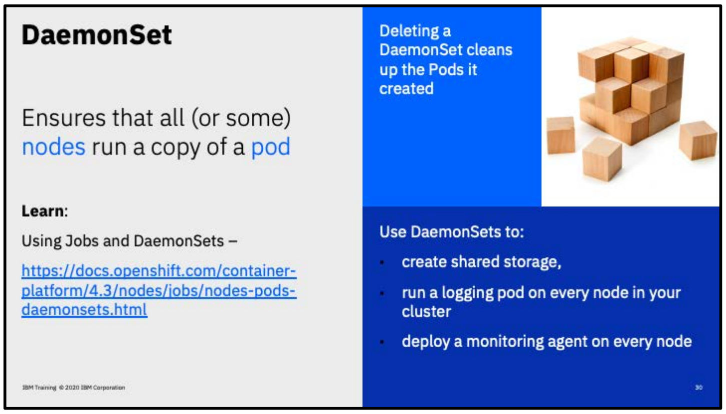


OpenShift Container Platform includes a pre-configured, pre-installed, and self-updating monitoring stack that is based on the Prometheus open source project and its wider eco-system. It provides monitoring of cluster components and includes a set of alerts to immediately notify the cluster administrator about any occurring problems and a set of Grafana dashboards. The cluster monitoring stack is only supported for monitoring OpenShift Container Platform clusters.

Learn:

About cluster monitoring -

https://docs.openshift.com/container-platform/4.3/monitoring/cluster_monitoring/about-cluster-monitoring.html



You can create and use DaemonSets to run replicas of a pod on specific or all nodes in an OpenShift Container Platform cluster.

A DaemonSet ensures that all (or some) nodes run a copy of a pod. As nodes are added to the cluster, pods are added to the cluster. As nodes are removed from the cluster, those pods are removed through garbage collection. Deleting a DaemonSet cleans up the Pods it created.

You can use daemonsets to create shared storage, run a logging pod on every node in your cluster, or deploy a monitoring agent on every node.

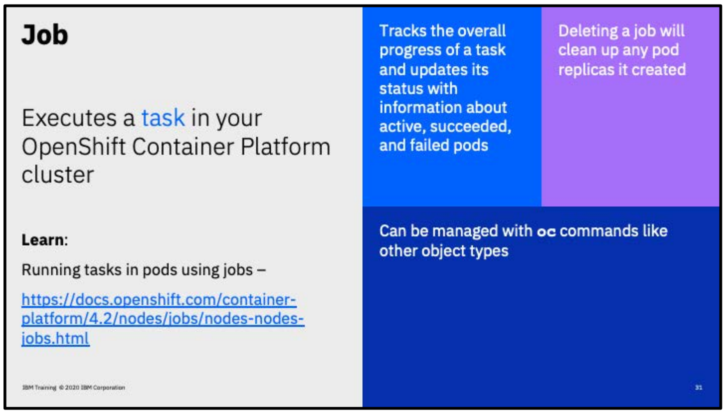
Learn:

Using Jobs and DaemonSets –

https://docs.openshift.com/container-platform/4.3/nodes/jobs/nodes-pods-daemonsets.html

Reference:

https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/



A *job* executes a task in your OpenShift Container Platform cluster.

A job tracks the overall progress of a task and updates its status with information about active, succeeded, and failed pods. Deleting a job will clean up any pod replicas it created. Jobs are part of the Kubernetes API, which can be managed with oc commands like other object types.

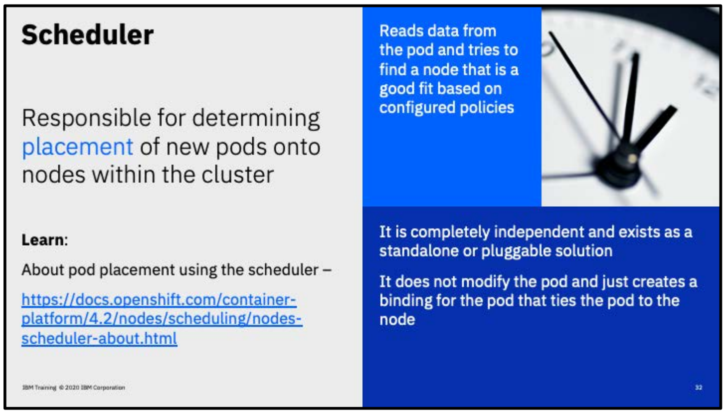
Learn:

Running tasks in pods using jobs –

https://docs.openshift.com/container-platform/4.2/nodes/jobs/nodes-nodes-jobs.html

Reference:

https://kubernetes.io/docs/concepts/workloads/controllers/jobs-run-to-completion/



Pod scheduling is an internal process that determines placement of new pods onto nodes within the cluster.

OpenShift Container Platform comes with a default scheduler that serves the needs of most users. The default scheduler uses both inherent and customization tools to determine the best fit for a pod.

In situations where you might want more control over where new pods are placed, the OpenShift Container Platform advanced scheduling features allow you to configure a pod so that the pod is required or has a preference to run on a particular node or alongside a specific pod by using pod affinity and anti-affinity rules.

One of the important use cases for scheduling within OpenShift Container Platform is to support flexible affinity and anti-affinity policies.

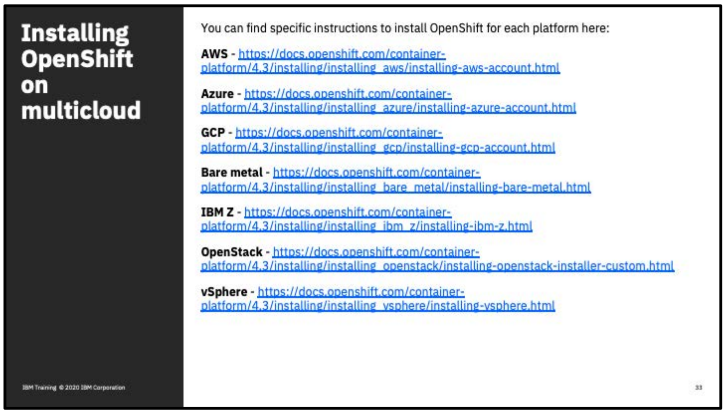
Learn:

About pod placement using the scheduler –

https://docs.openshift.com/container-platform/4.2/nodes/scheduling/nodes-scheduler-about.html

Reference:

https://kubernetes.io/docs/concepts/scheduling/kube-scheduler/



You can find specific instructions to install OpenShift for each platform here:

AWS - https://docs.openshift.com/container-platform/4.3/installing/installing_aws/installing-aws-account.html

Azure - https://docs.openshift.com/container-platform/4.3/installing/installing_azure/installing-azure-account.html

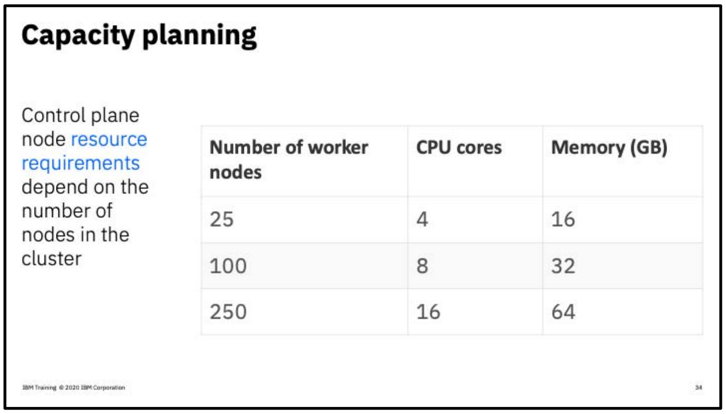
GCP - https://docs.openshift.com/container-platform/4.3/installing/installing_gcp/installing-gcp-account.html

Bare metal - https://docs.openshift.com/container-platform/4.3/installing/installing_bare_metal/installing-bare-metal.html

IBM Z - https://docs.openshift.com/container-platform/4.3/installing/installing_ibm_z/installing-ibm-z.html

OpenStack - https://docs.openshift.com/container-platform/4.3/installing/installing_openstack/installing-openstack-installer-custom.html

vSphere - https://docs.openshift.com/container-platform/4.3/installing/installing_vsphere/installing-vsphere.html

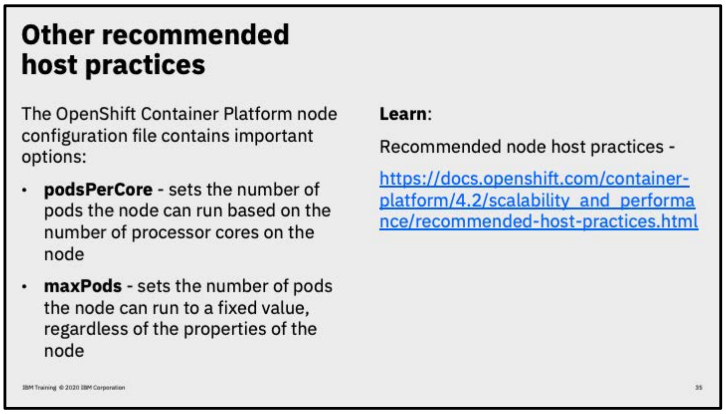


The control plane node resource requirements depend on the number of nodes in the cluster. The following control plane node size recommendations are based on the results of control plane density focused testing.

Because you cannot modify the control plane node size in a running OpenShift Container Platform cluster, you must estimate your total node count and use the suggested size during installation.

Reference:

https://docs.openshift.com/container-platform/4.2/scalability_and_performance/recommended-host-practices.html#master-node-sizing



The OpenShift Container Platform node configuration file contains important options. For example, two parameters control the maximum number of pods that can be scheduled to a node: podsPerCore and maxPods.

When both options are in use, the lower of the two values limits the number of pods on a node. Exceeding these values can result in:

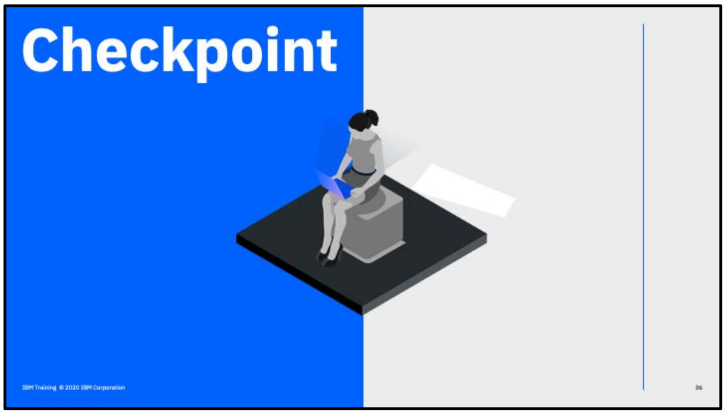
- Increased CPU utilization.
- Slow pod scheduling.
- Potential out-of-memory scenarios, depending on the amount of memory in the node.
- Exhausting the pool of IP addresses.
- Resource overcommitting, leading to poor user application performance.

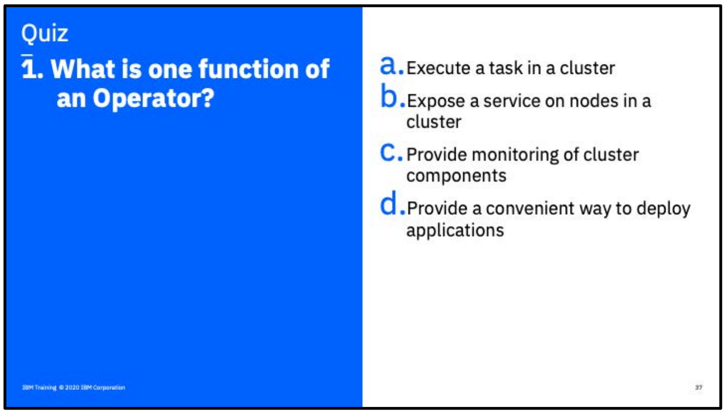
[There are other settings that you must be aware of, detailed in the documentation, here.]

Learn:

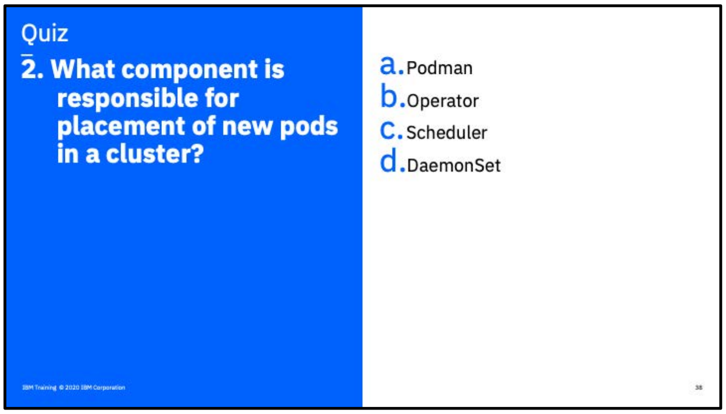
Recommended node host practices -

https://docs.openshift.com/container-platform/4.3/scalability_and_performance/recommended-host-practices.html

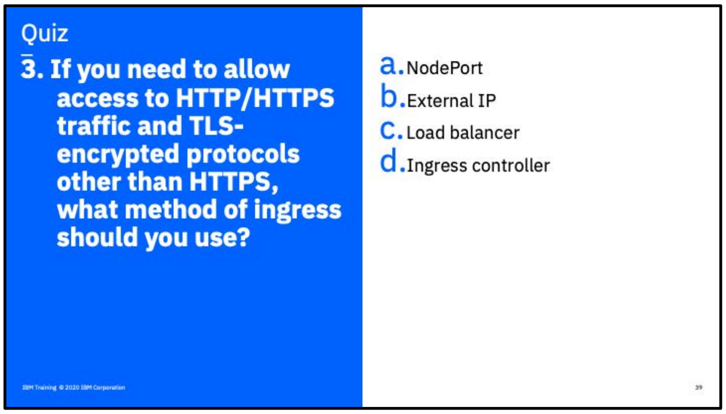

Checkpoint



Correct answer: d.



Correct answer: c.



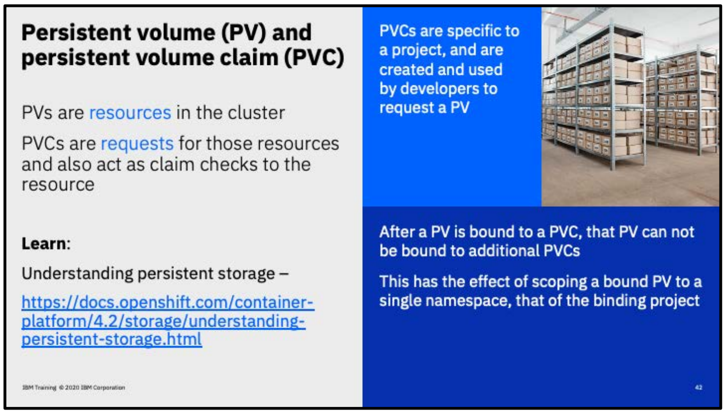
Correct answer: d.


High availability, disaster recovery, backup, and storage

Objectives

- Discuss storage considerations
- Explain OpenShift high availability
- Explain OpenShift disaster recovery
- Discuss backup and restore strategies

41



Managing storage is a distinct problem from managing compute resources. OpenShift Container Platform uses the Kubernetes persistent volume (PV) framework to allow cluster administrators to provision persistent storage for a cluster. Developers can use persistent volume claims (PVCs) to request PV resources without having specific knowledge of the underlying storage infrastructure.

PVCs are specific to a project and are created and used by developers as a means to use a PV. PV resources on their own are not scoped to any single project; they can be shared across the entire OpenShift Container Platform cluster and claimed from any project. After a PV is bound to a PVC, that PV can not be bound to additional PVCs. This has the effect of scoping a bound PV to a single namespace, that of the binding project.

Many storage types are available for use as persistent volumes in OpenShift Container Platform. While all of them can be statically provisioned by an administrator, some types of storage are created dynamically using the built-in provider and plug-in APIs.

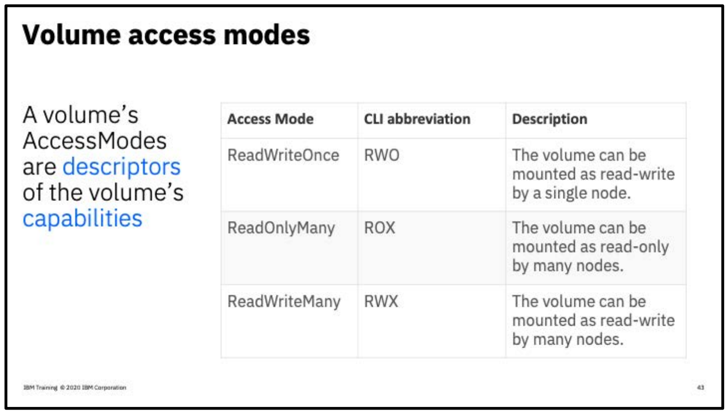
Learn:

Understanding persistent storage –

https://docs.openshift.com/container-platform/4.2/storage/understanding-persistent-storage.html

Reference:

https://kubernetes.io/docs/concepts/storage/persistent-volumes/



Access Mode	CLI abbreviation	Description
ReadWriteOnce	RWO	The volume can be mounted as read-write by a single node.
ReadOnlyMany	ROX	The volume can be mounted as read-only by many nodes.
ReadWriteMany	RWX	The volume can be mounted as read-write by many nodes.

A PersistentVolume can be mounted on a host in any way supported by the resource provider. Providers have different capabilities and each PV's access modes are set to the specific modes supported by that particular volume. For example, NFS can support multiple read-write clients, but a specific NFS PV might be exported on the server as read-only. Each PV gets its own set of access modes describing that specific PV's capabilities.

Claims are matched to volumes with similar access modes. The only two matching criteria are access modes and size.

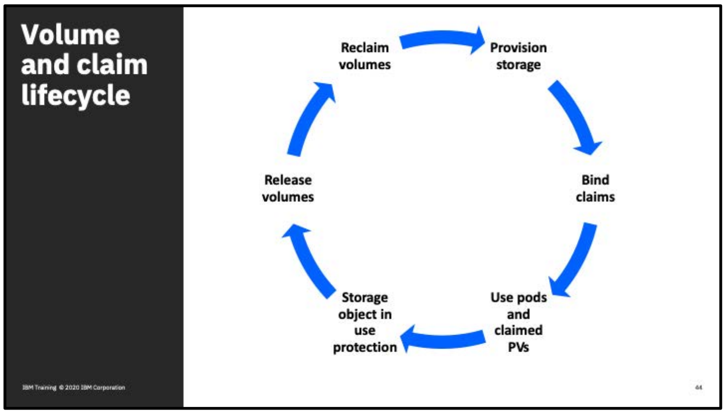
A volume's AccessModes are descriptors of the volume's capabilities.

They are not enforced constraints.

The storage provider is responsible for runtime errors resulting from invalid use of the resource.

Reference:

https://docs.openshift.com/container-platform/4.2/storage/understanding-persistent-storage.html



The interaction between PVs and PVCs have the following lifecycle.

Provision storage: In response to requests from a developer defined in a PVC, a cluster administrator configures one or more dynamic provisioners that provision storage and a matching PV. Alternatively, a cluster administrator can create a number of PVs in advance that carry the details of the real storage that is available for use. PVs exist in the API and are available for use.

Bind claims: When you create a PVC, you request a specific amount of storage, specify the required access mode, and create a storage class to describe and classify the storage. The control loop in the master watches for new PVCs and binds the new PVC to an appropriate PV. If an appropriate PV does not exist, a provisioner for the storage class creates one. The size of all PVs might exceed your PVC size. This is especially true with manually provisioned PVs. To minimize the excess, OpenShift Container Platform binds to the smallest PV that matches all other criteria. Claims remain unbound indefinitely if a matching volume does not exist or can not be created with any available provisioner servicing a storage class. Claims are bound as matching volumes become available. For example, a cluster with many manually provisioned 50Gi volumes would not match a PVC requesting 100Gi. The PVC can be bound when a 100Gi PV is added to the cluster.

Use Pods and claimed PVs: Pods use claims as volumes. The cluster inspects the claim to find the bound volume and mounts that volume for a Pod. For those volumes that support multiple access modes, you must specify which mode applies when you use the claim as a volume in a Pod. Once you have a claim and that claim is bound, the bound PV belongs to you for as long as you need it. You can schedule Pods and access claimed PVs by including persistentVolumeClaim in the Pod’s volumes block.

Storage Object in Use Protection: The Storage Object in Use Protection feature ensures that PVCs in active use by a Pod and PVs that are bound to PVCs are not removed from the system, as this can result in data loss. Storage Object in Use Protection is enabled by default. A PVC is in active use by a Pod when a Pod object exists that uses the PVC. If a user deletes a PVC that is in active use by a Pod, the PVC is not removed immediately. PVC removal is postponed until the PVC is no longer actively used by any Pods. Also, if a cluster admin deletes a PV that is bound to a PVC, the PV is not removed immediately. PV removal is postponed until the PV is no longer bound to a PVC.

Release volumes: When you are finished with a volume, you can delete the PVC object from the API, which allows reclamation of the resource. The volume is considered released when the claim is deleted, but it is not yet available for another claim. The previous claimant’s data remains on the volume and must be handled according to policy.

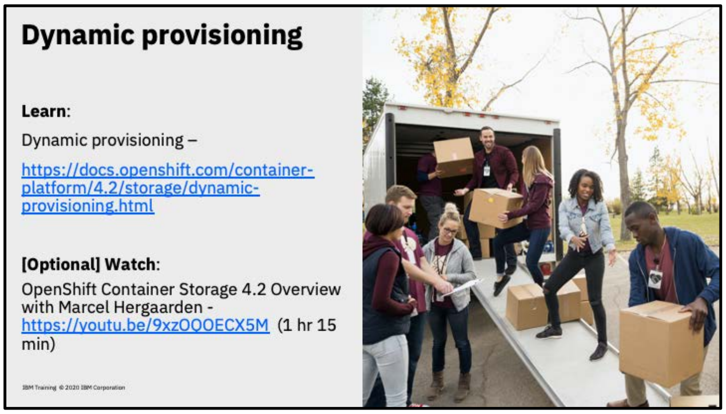
Reclaim volumes: The reclaim policy of a PersistentVolume tells the cluster what to do with the volume after it is released. Volumes reclaim policy can either be Retain, Recycle, or Delete.

- Retain reclaim policy allows manual reclamation of the resource for those volume plug-ins that support it.
- Recycle reclaim policy recycles the volume back into the pool of unbound persistent volumes once it is released from its claim.
- The Recycle reclaim policy is deprecated in OpenShift Container Platform 4. Dynamic provisioning is recommended for equivalent and better functionality.
- Delete reclaim policy deletes both the PersistentVolume object from OpenShift Container Platform and the associated storage asset in external infrastructure, such as AWS EBS or VMware vSphere.

Dynamically provisioned volumes are always deleted.

Reference:

https://docs.openshift.com/container-platform/4.2/storage/understanding-persistent-storage.html



The StorageClass resource object describes and classifies storage that can be requested, as well as provides a means for passing parameters for dynamically provisioned storage on demand. StorageClass objects can also serve as a management mechanism for controlling different levels of storage and access to the storage. Cluster Administrators (cluster-admin) or Storage Administrators (storage-admin) define and create the StorageClass objects that users can request without needing any intimate knowledge about the underlying storage volume sources.

The OpenShift Container Platform persistent volume framework enables this functionality and allows administrators to provision a cluster with persistent storage. The framework also gives users a way to request those resources without having any knowledge of the underlying infrastructure.

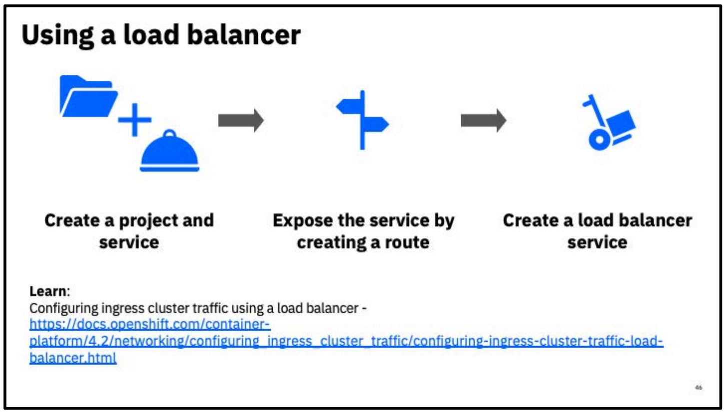
Learn:

Dynamic provisioning -

https://docs.openshift.com/container-platform/4.2/storage/dynamic-provisioning.html

[Optional] Watch:

OpenShift Commons Briefing OpenShift Container Storage 4.2 Overview with Marcel Hergaarden Red Hat - https://youtu.be/9xzOQOECX5M (1 hour and 15 minutes)



OpenShift Container Platform provides methods for communicating from outside the cluster with services running in the cluster. This method uses a load balancer. If you do not need a specific external IP address, you can configure a load balancer service to allow external access to an OpenShift Container Platform cluster. A load balancer service allocates a unique IP. The load balancer has a single edge router IP, which can be a virtual IP (VIP), but is still a single machine for initial load balancing.

Using a load balancer involves:

Creating a project and service

Exposing the service by creating a route

Creating a load balancer service

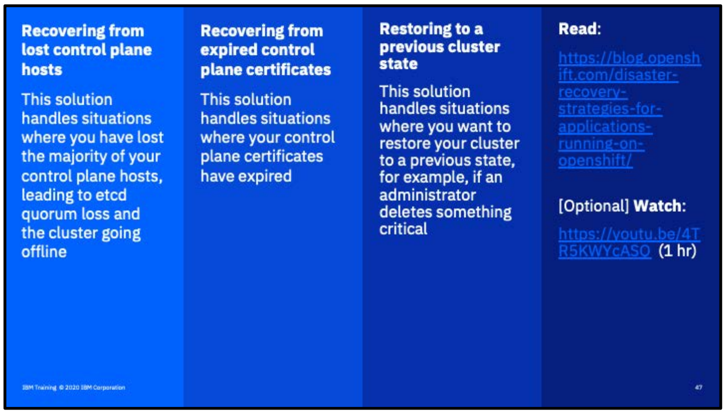
Learn:

Configuring ingress cluster traffic using a load balancer -

https://docs.openshift.com/container-platform/4.2/networking/configuring_ingress_cluster_traffic/configuring-ingress-cluster-traffic-load-balancer.html

Reference:

https://kubernetes.io/docs/tasks/access-application-cluster/create-external-load-balancer/



You might need to follow one or more of the following procedures in order to return your cluster to a working state.

Recovering from lost control plane hosts

This solution handles situations where you have lost the majority of your control plane hosts, leading to etcd quorum loss and the cluster going offline. As long as you have taken an etcd backup and have at least one remaining healthy control plane host, you can follow this procedure to recover your cluster.

If you have a majority of your control plane hosts still available and have an etcd quorum, then follow the procedure to replace a single failed control plane host.

Restoring to a previous cluster state

This solution handles situations where you want to restore your cluster to a previous state, for example, if an administrator deletes something critical. As long as you have taken an etcd backup, you can follow this procedure to restore your cluster to a previous state.

Recovering from expired control plane certificates

This solution handles situations where your control plane certificates have expired. For example, if you shut down your cluster before the first certificate rotation, which occurs 24 hours after installation, your certificates will not be rotated and will expire. You can follow this procedure to recover from expired control plane certificates.

Read:

Disaster Recovery Strategies for Applications Running on OpenShift -

https://blog.openshift.com/disaster-recovery-strategies-for-applications-running-on-openshift/

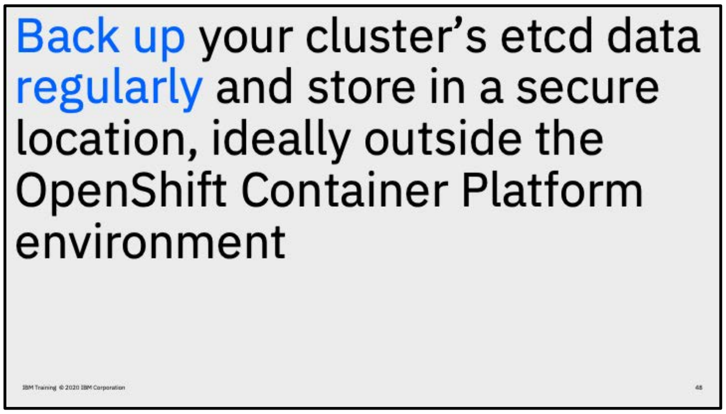
Watch:

5 Key Traits of Effective Disaster Recovery on Kubernetes -

https://youtu.be/4TR5KWYcASQ (1 hour)

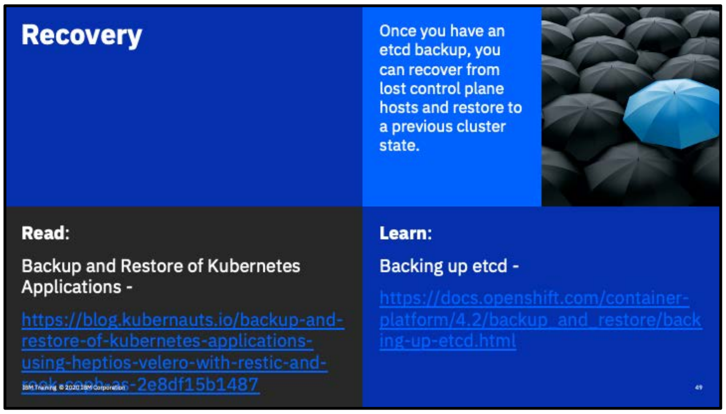
Reference:

https://docs.openshift.com/container- platform/4.2/backup_and_restore/disaster_recovery/about-disaster-recovery.html



etcd is the key-value store for OpenShift Container Platform, which persists the state of all resource objects

Back up your cluster's etcd data regularly and store in a secure location ideally outside the OpenShift Container Platform environment. Do not take an etcd backup before the first certificate rotation completes, which occurs 24 hours after installation, otherwise the backup will contain expired certificates. It is also recommended to take etcd backups during non-peak usage hours, as it is a blocking action.



Once you have an etcd backup, you can recover from lost control plane hosts and restore to a previous cluster state.

You can perform the etcd data backup process on any control plane host that has connectivity to the etcd cluster, where the proper certificates are provided.

Read:

Backup and Restore of Kubernetes Applications -

https://blog.kubernauts.io/backup-and-restore-of-kubernetes-applications-using-heptios-velero-with-restic-and-rook-ceph-as-2e8df15b1487

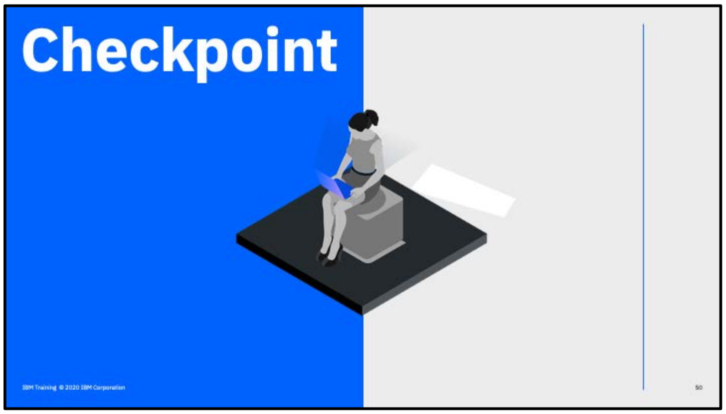
Learn:

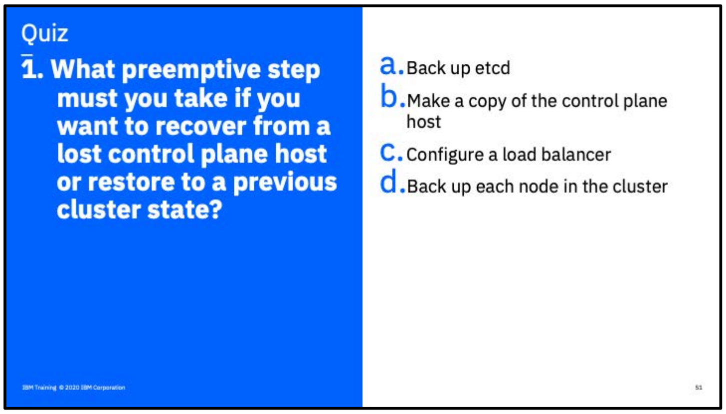
Backing up etcd -

https://docs.openshift.com/container-platform/4.2/backup_and_restore/backing-up-etcd.html

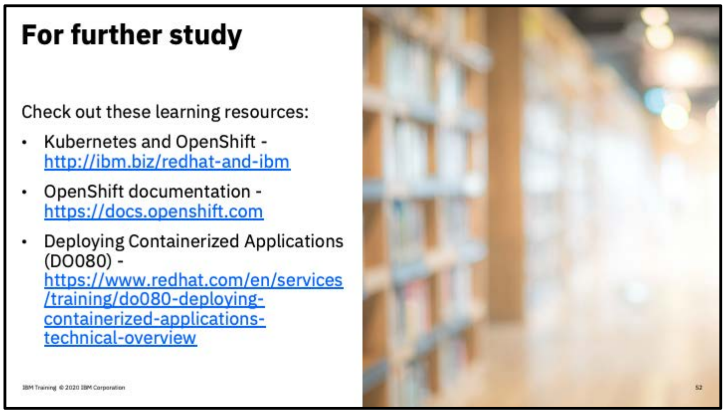
Reference:

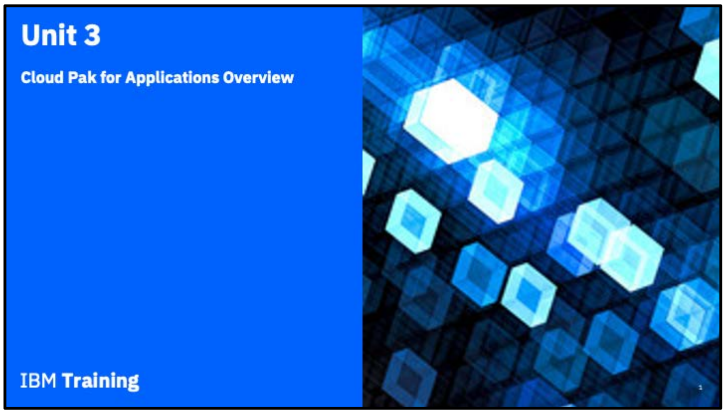
https://docs.openshift.com/container-platform/4.2/backup_and_restore/replacing-failed-master.html


Checkpoint



Correct answer: a.


For further study
Check out these learning resources:
• Kubernetes and OpenShift - http://ibm.biz/redhat-and-ibm
• OpenShift documentation - https://docs.openshift.com
• Deploying Containerized Applications (DO080) - https://www.redhat.com/en/services/training/do080-deploying-containerized-applications-technical-overview
IBM Training © 2020 IBM Corporation
52



Unit 3 Cloud Pak for Applications Overview

This unit introduces the Cloud Pak for Applications. It highlights the components, features and benefits, and describes some use cases for Cloud Pak for Applications.

Presentation time: about 40 minutes

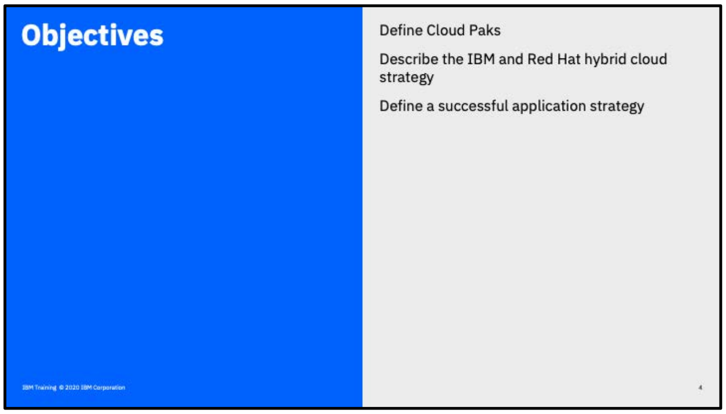
Supplemental material:

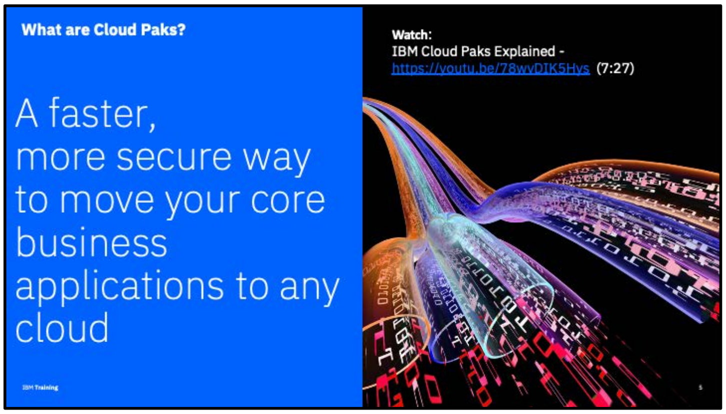
Videos: 23 minutes

Readings: 35 minutes

Topics


What are
Cloud Paks?
IBM Training © 2020 IBM Corporation
3

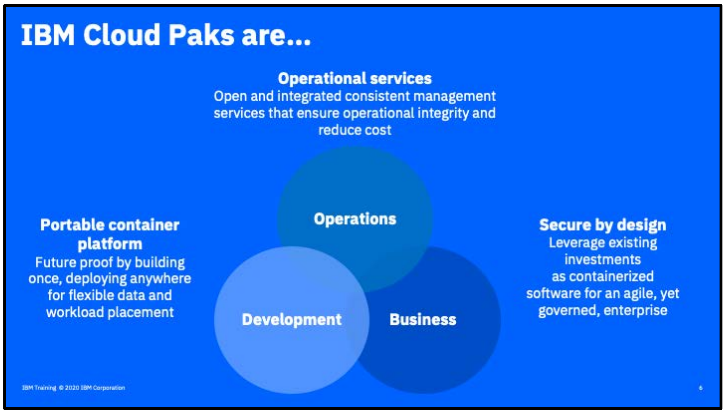

Objectives
Define Cloud Paks
Describe the IBM and Red Hat hybrid cloud strategy
Define a successful application strategy



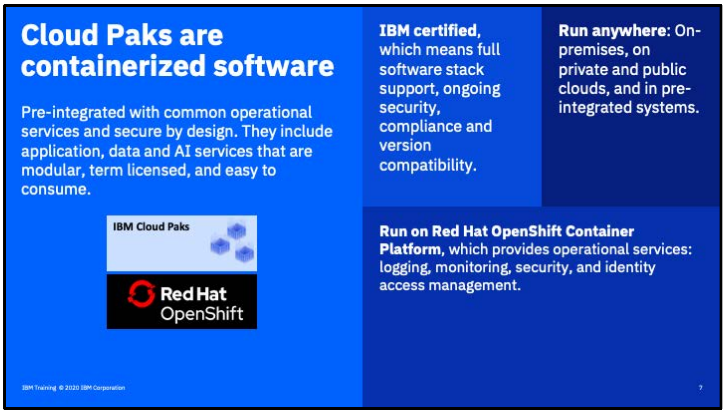
Cloud Paks are a faster more secure way to move your core business applications to any cloud.

Watch:

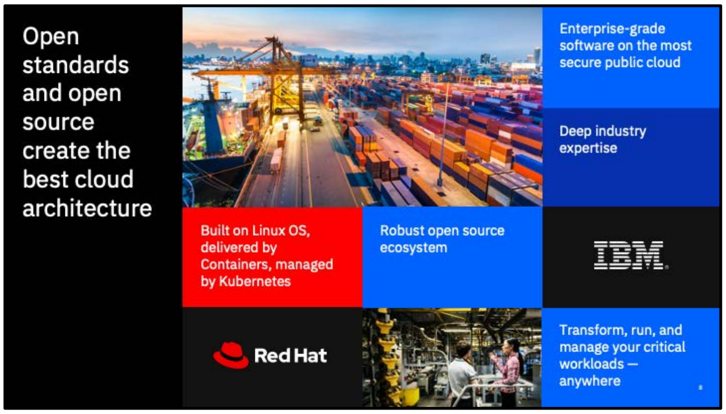
IBM Cloud Paks Explained - https://youtu.be/78wvDIK5Hys (7:27)



IBM Cloud Paks provide a portable container platform that you can deploy anywhere. They provide operational services to ensure consistent management and operational integrity at a reduced cost. They leverage existing investments as containerized software for a secure, agile enterprise. And, the portable container platform is essentially future proof because you can deploy anywhere.



Cloud Paks are containerized software that is pre-integrated with common operational services and is secure by design. These services include application, data and AI services that are modular, term licensed and easy to consume. Cloud Paks are IBM certified, meaning that they are fully supported with ongoing security, compliance and version compatibility. As mentioned before, you can run Cloud Paks anywhere – on premises, on private and public clouds, and in pre-integrated systems. Cloud Paks run on Red Hat OpenShift Container Platform, which provides logging, monitoring, security, and identity access management.



IBM and Red Hat

A significant part of IBM's hybrid cloud strategy is the acquisition of Red Hat.

IBM cloud strategy was focused on hybrid outcomes for many years – with solutions that are supported in a variety of ways across private and public clouds.

IBM was heading in a container-based direction, specifically kubernetes, for some time. **Containers** are the practical means to achieve multi-cloud portable workloads and **kubernetes** is the practical means to achieve multi-cloud common management and orchestration approaches for those workloads. **OpenShift, running on Red Hat Enterprise Linux, is a concrete way to provide a common container management platform wherever you choose to run your apps.**

IBM and Red Hat together contribute to almost every part of the Kubernetes platform and have made the greatest number of contributions after Google over the lifetime of the technology. [1]

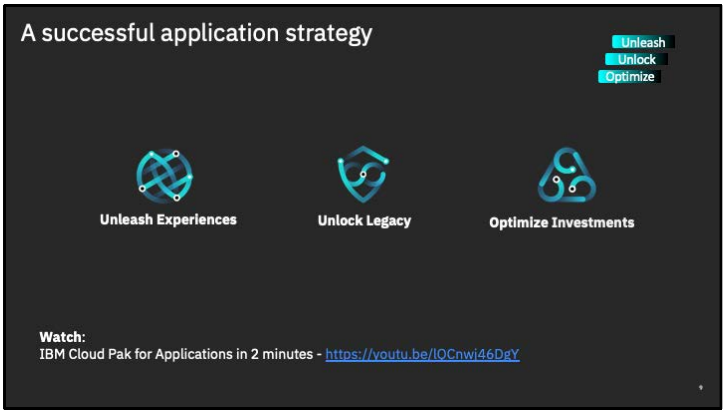
Building the core platform capabilities in the open was critical to the breadth of the ecosystem and the pace of innovation achieved, but it is the integration of this innovation into a coherent, secure, supported offering - OpenShift Container Platform – that is valuable for customers and IBM.

IBM uses this technology itself to provide a common approach to patch, secure and support the technologies that are built on the platform.

IBM delivers solutions on this foundation with a prescriptive and automated approach.

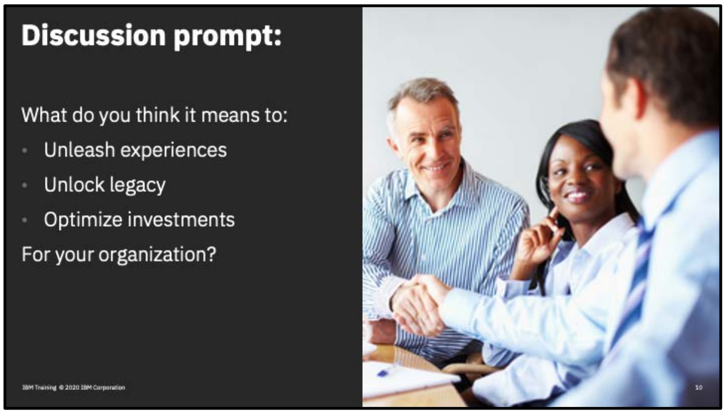
[1] https://k8s.devstats.cncf.io/d/9/companies-table?orgId=1&var-period_name=Last%20decade&var-metric=contributions

Kubernetes v1.0 released in 2015 and contributed to the CNCF

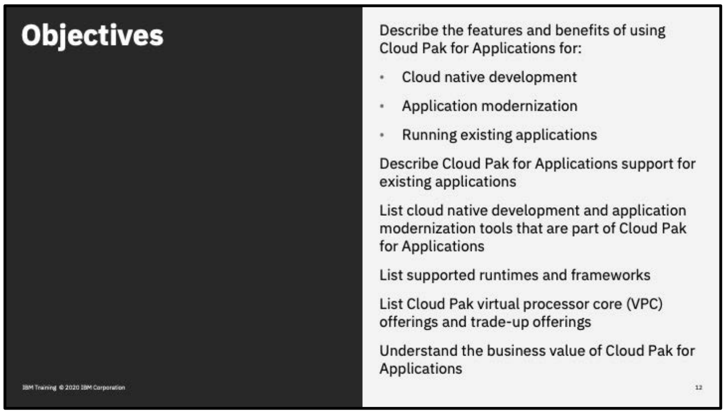


A successful application strategy lets you unleash experiences, unlock legacy assets, and optimize investments.

Watch:
IBM Cloud Pak for Applications in 2 minutes - **-**
https://youtu.be/lQCnwj46DgY


Discussion prompt:
What do you think it means to:
Unleash experiences
Unlock legacy
Optimize investments
For your organization?


Cloud Pak for Applications features and benefits
IBM Training © 2020 IBM Corporation


Objectives
Describe the features and benefits of using Cloud Pak for Applications for:
Cloud native development
Application modernization
Running existing applications
Describe Cloud Pak for Applications support for existing applications
List cloud native development and application modernization tools that are part of Cloud Pak for Applications
List supported runtimes and frameworks
List Cloud Pak virtual processor core (VPC) offerings and trade-up offerings
Understand the business value of Cloud Pak for Applications
IBM Training © 2020 IBM Corporation
12

Cloud Pak for Applications provides what you need to run existing applications, modernize and leverage existing investments, and build new cloud-native solutions.

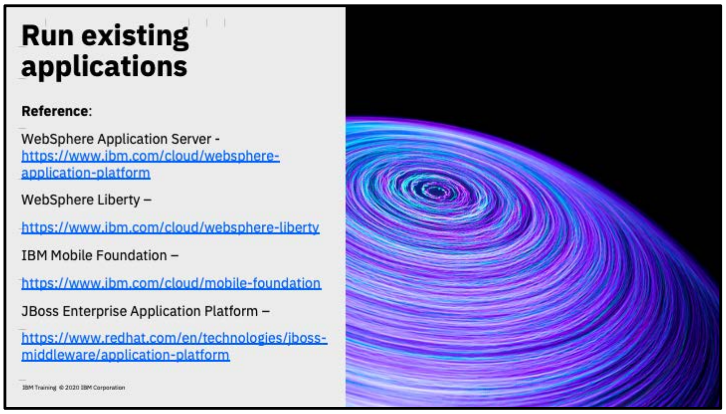
To run existing applications, Cloud Pak for Applications includes WebSphere Application Server Base and Network Deployment, Liberty Core, and Mobile Foundation. It also includes Jboss Enterprise Application Platform.

Cloud Pak for Applications includes a suite of modernization tools, including Transformation Advisor, Application Navigator, and WebSphere Migration Toolkit. In V4.2, a new tool, Mono2Micro, is included as a tech preview or open beta. Enterprise developer tools and extensions for local IDEs are also included.

To build new cloud native applications, Cloud Pak for Applications provides a complete platform that includes Accelerators for Teams with enterprise governance, Red Hat CodeReady Workspaces, several enterprise runtimes – all running on Red Hat OpenShift Container Platform.

Perpetual and term licensing options are available with no functional restrictions to OpenShift.

These components are described in more detail in this course.



For existing applications that just need to keep on working without change, Cloud Pak for Applications provides the familiar WebSphere family of products to run Traditional WebSphere and Liberty.

WebSphere® Application Server is a scalable, secure and reliable Java EE runtime. It supports microservices and standards-based programming models, enterprise-level security, integrated management, and administrative tooling. It can provide a faster, easier path to cloud with Cloud Pak for Applications.

WebSphere Liberty's speed and flexibility increases developer productivity 25 percent. Simplified admininstration and low overhead reduce infrastructure costs 30 percent.

IBM Mobile Foundation is a platform to build and deploy the next generation of digital apps, including apps for mobile, wearables, conversation, web and PWAs. With IBM Mobile Foundation, developers get containerized mobile backend services that cover comprehensive security, application lifecycle management, push notifications, feature toggle, offline sync and backend integration. The platform also ships a low-code studio, private app store and rich SDKs for widely used mobile frameworks for both native and hybrid developers.

JBoss EAP also delivers enterprise-grade security, performance, and scalability in any environment.

[You can learn more by checking out these resources.]

Reference:

WebSphere Application Server - https://www.ibm.com/cloud/websphere-application-platform

WebSphere Liberty –

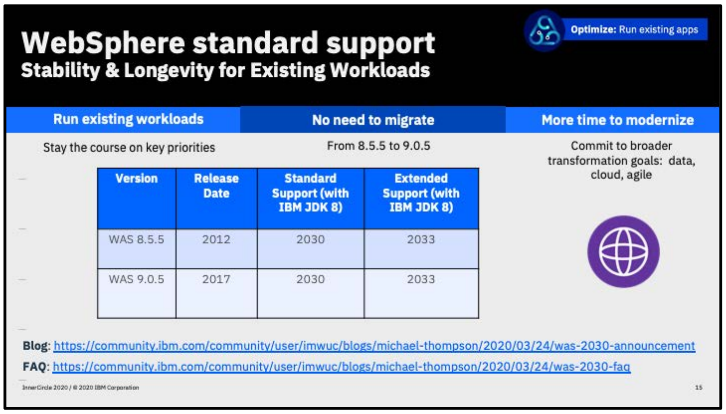
https://www.ibm.com/cloud/websphere-liberty

IBM Mobile Foundation –

https://www.ibm.com/cloud/mobile-foundation

JBoss Enterprise Application Platform –

https://www.redhat.com/en/technologies/jboss-middleware/application-platform



IBM will provide standard support for WebSphere Application Server versions 8.5.5 and 9.0.5, with IBM JDK 8, through at least 2030, for WAS ND, WAS (Base) and WAS Family Edition. Nearly all WebSphere clients are exploring or have already begun a digital transformation journey. This transformation includes building new cloud native applications as well as modernizing existing assets. This timeline is intended to provide ample time and opportunity for any organization to determine the appropriate path to cloud according to their needs.

Note: This support statement is for traditional WAS only. Liberty has its own roadmap and support lifecycle: https://www.ibm.com/support/pages/node/869798

Blog: https://community.ibm.com/community/user/imwuc/blogs/michael-thompson/2020/03/24/was-2030-announcement

FAQ: https://community.ibm.com/community/user/imwuc/blogs/michael-thompson/2020/03/24/was-2030-faq

Transformation Advisor evaluates on-premises Java EE applications and messaging infrastructure, and then provides recommendations, detailed reports, and generates artifacts for automated deployment to containers.

It introspects traditional WebSphere, IBM MQ, IIB/ACE, WebLogic, JBoss and Tomcat deployments to determine their complexity and cloud-readiness.

In Cloud Pak for Applications V4.2, Transformation Advisor:

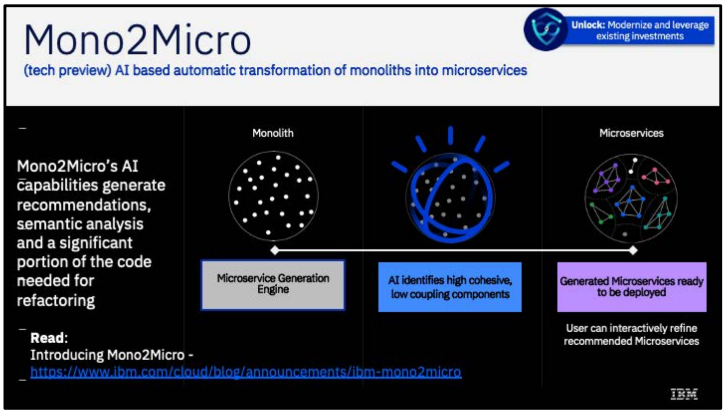
Analyzes WebSphere deployments on z/OS

Supports tWAS Base containers

IBM Application Navigator is a tool that extends the Kubernetes® console to provide the display, inspection, understanding, and navigation of applications composed of Kubernetes resources and existing middleware, such as WebSphere Application Server. Application Navigator extends the open source Kubernetes Application Navigator (kAppNav) by providing integration to WebSphere Application Server Network Deployment cells and Liberty collectives, which enables visibility to existing and containerized applications.

Watch:

Simplify Modernization with IBM Cloud Pak for Applications - https://youtu.be/IM7S8BqF4U8 (2:46)



Mono2Micro is a tech preview that has AI capabilities to generate recommendations, semantic analysis, and a significant portion of the code needed for refactoring applications into microservices.

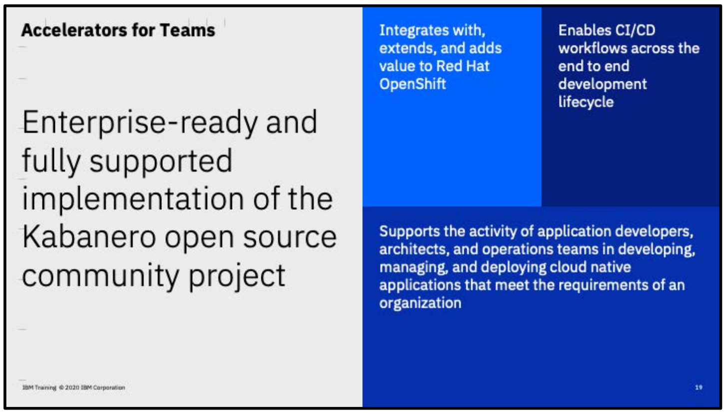
It is now available in Cloud Pak for Applications V4.2 as an open beta.

Read:

Introducing Mon2Micro -

https://www.ibm.com/cloud/blog/announcements/ibm-mono2micro

This course focuses on Accelerators for Teams, and how they are used to support cloud native application development, deployment, and management, with enterprise governance and security in mind. Red Hat CodeReady workspaces also provides developers with a means to quickly get started locally with OpenShift, but that is not going to be the emphasis of this course. This course covers what you need to know about Accelerators for Teams, enterprise runtimes, and Red Hat OpenShift Container Platform to build a cloud native solution architecture.



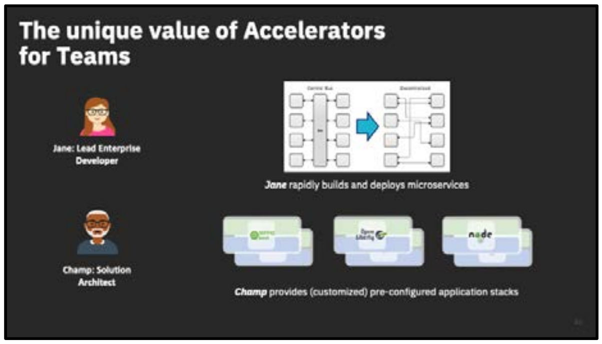
Accelerators for Teams is the commercial enterprise-ready and fully supported implementation of the Kabanero open source community project. Accelerators for Teams integrates with, extends, and adds value to Red Hat OpenShift.

Kabanero offers open source technologies in a microservices-based framework that simplifies development, build, and deployment of applications for both Kubernetes and Knative (serverless).

Accelerators for Teams supports the activity of application developers, architects, and operations teams in developing, managing, and deploying cloud native applications that meet the requirements of an organization. It enables Continuous Integration and Continuous Deployment (CI/CD) workflows across the end to end development lifecycle.

With Accelerators for Teams, application developers use application stacks to simplify the development and optimization of containerized cloud native applications.

Also supporting the developer experience is a set of managed pipelines that provide consistent, managed, and governed CI/CD processes that react to repository events that occur during code development. These components cover the disciplines of serving applications, service mesh, build, and deployment.



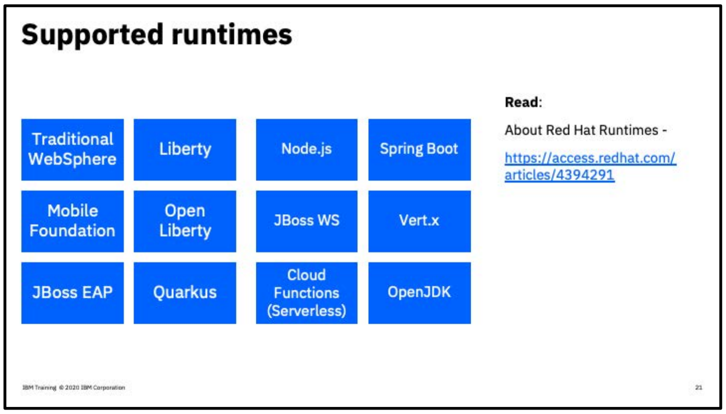
Some of the unique benefits of Accelerators for Teams are:

- **It empowers teams to start quicker:** IT operations and architects can provide curated, pre-configured development and runtime stacks to the development team, and

- **It promotes operational efficiency:** Maintenance, security controls, and compliance can be returned to the IT operations team with updates being dynamically pushed to the environments that are used by the development team.

In this scenario, Jane, the lead enterprise developer can rapidly build and deploy microservices by using the pre-configured application stacks that Champ, the solution architect, provides. These stacks can be customized for a broad range of runtimes.

[Accelerators for Teams is described in more detail later in the course.]

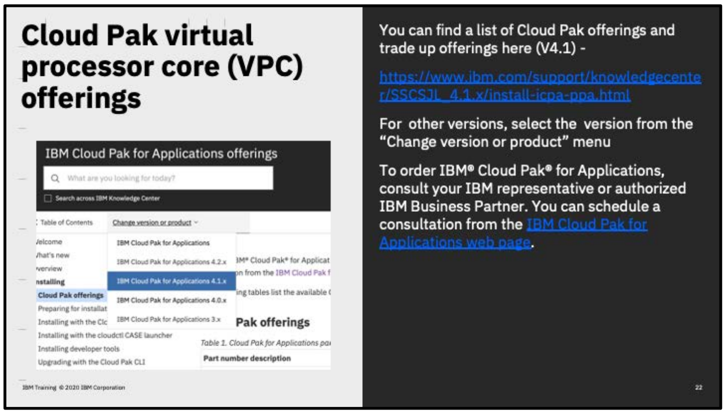


Cloud Pak for Applications supports several popular runtimes, listed here. For the latest information about supported Red Hat runtimes, read the article here.

Read:

About Red Hat Runtimes **-**

https://access.redhat.com/articles/4394291

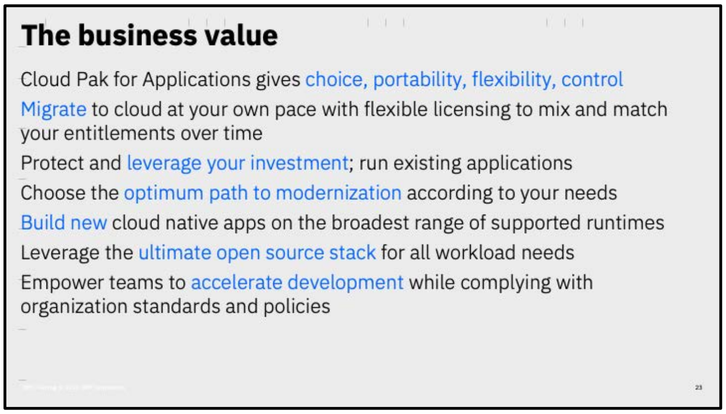


You can find a list of Cloud Pak offerings and trade up offerings here -

https://www.ibm.com/support/knowledgecenter/SSCSJL_4.1.x/install-icpa-ppa.html

You can also find information on other versions of the product by selecting the version from the "Change version or product" menu.

To order IBM® Cloud Pak® for Applications, consult your IBM representative or authorized IBM Business Partner. You can schedule a consultation from the IBM Cloud Pak for Applications web page.



In summary,

Cloud Pak for Applications gives choice, portability, flexibility, control

Migrate to cloud at your own pace with flexible licensing to mix and match your entitlements over time

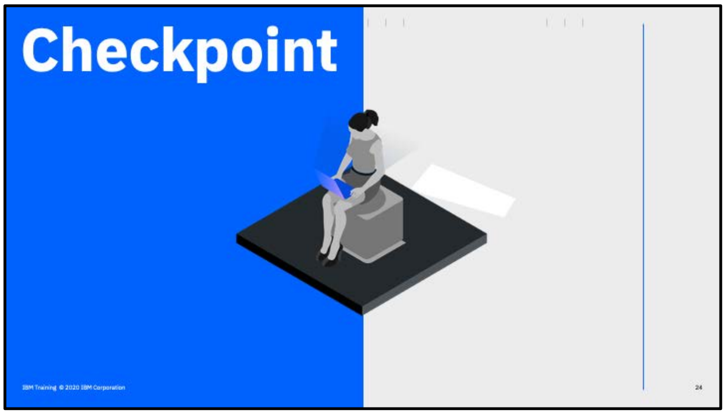
Protect and leverage your investment; run existing applications

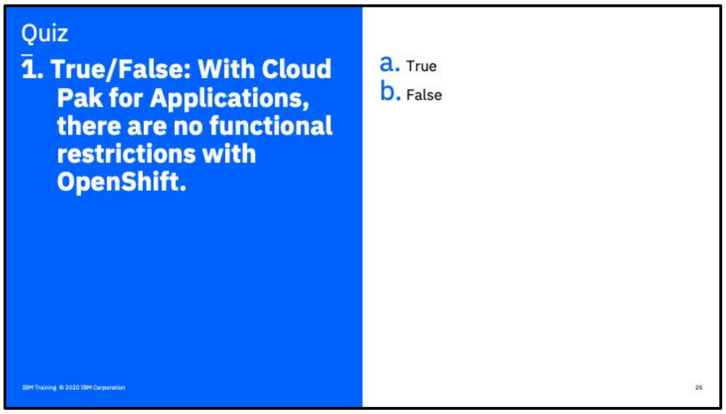
Choose the optimum path to modernization according to your needs

Build new cloud native apps on the broadest range of supported runtimes

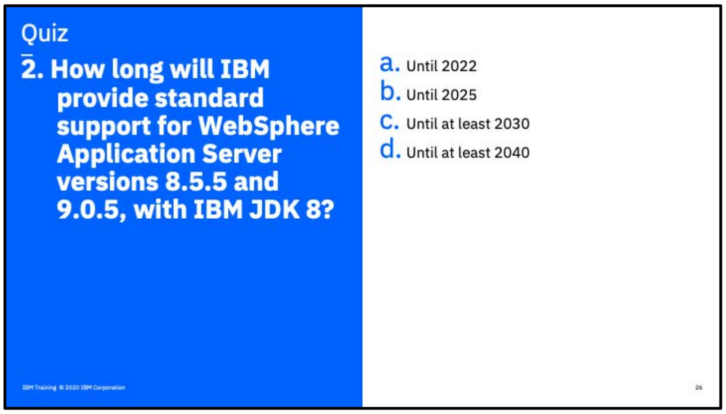
Leverage the ultimate open source stack for all workload needs

Empower teams to accelerate development while complying with organization standards and policies

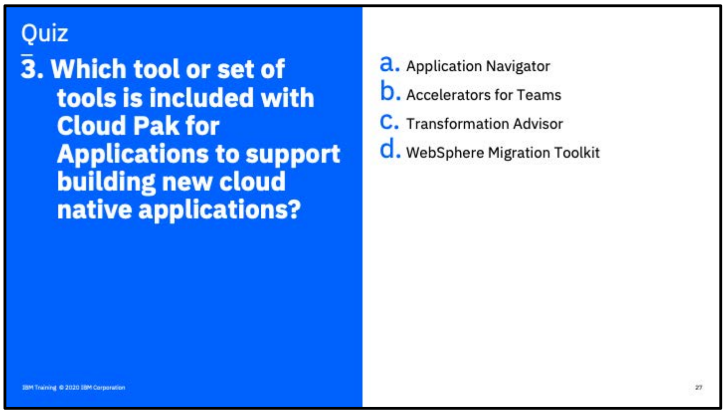

Checkpoint
IBM Training © 2020 IBM Corporation
24



Correct answer: a.

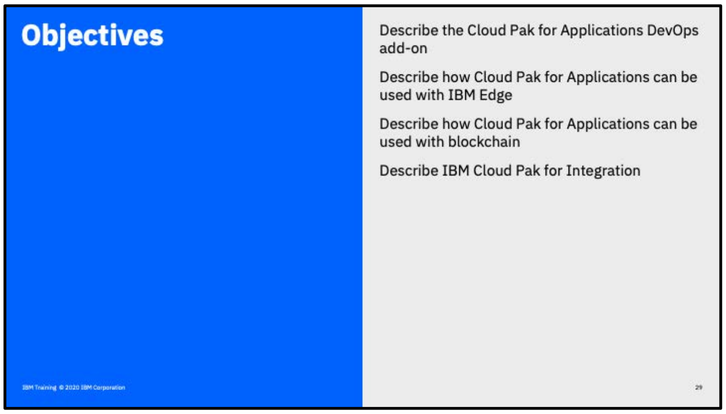


Correct answer: c.

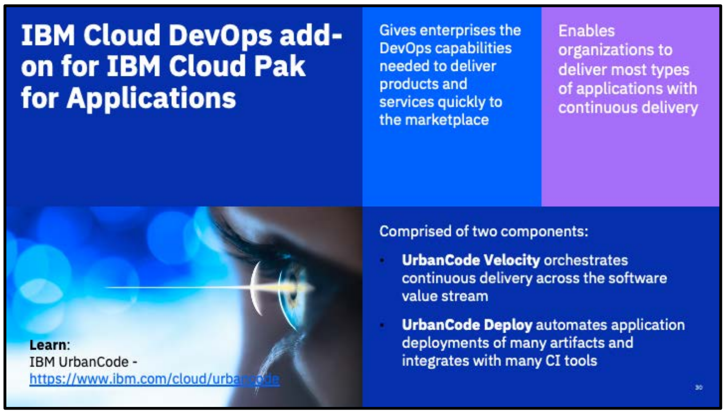


Correct answer: b.


Cloud Pak for Applications integration scenarios
IBM Training © 2020 IBM Corporation



[This section describes other offerings that IBM Cloud Pak for Applications can be integrated or used adjacent with.]



IBM Cloud DevOps add-on for IBM Cloud Pak for Applications gives enterprises the capability to deliver products and services quickly to the marketplace by using continuous delivery.

Cloud DevOps for IBM Cloud Pak for Applications V3.0 is comprised of two components:

UrbanCode® Velocity V1.4 orchestrates continuous delivery across the software value stream. It helps you visualize, orchestrate, and optimize your continuous delivery value stream. Development teams, release managers, and transformation executives now have one tool to consolidate data and drive action across many agile continuous integration and continuous delivery systems. UrbanCode Velocity V1.4 is designed for companies that are looking to move beyond automation to drive culture change and fully realize the benefits of their DevOps transformation investments.

Reference:

https://www.ibm.com/support/knowledgecenter/SSCKX6_1.4.x/com.ibm.uvelocity.doc/ucv_version_welcome.html

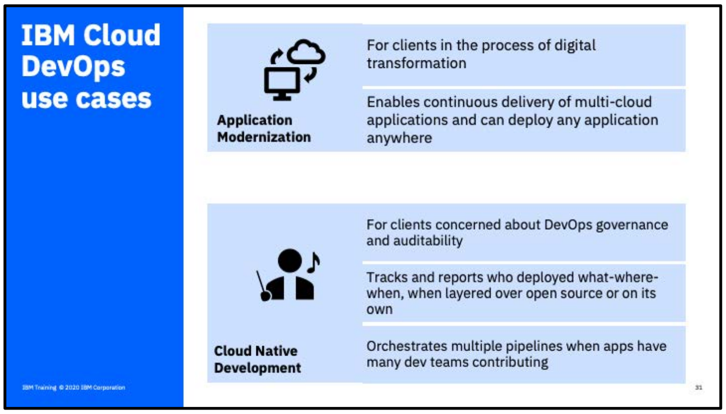
UrbanCode Deploy V7.0.4 automates application deployments of many artifacts and integrates with many CI tools. It can be the single deployment automation and orchestration tool for an enterprise. With UrbanCode Deploy V7.0.4, you can deploy any application anywhere, including distributed platforms, IBM® z/OS®, private and public clouds, and container platforms Kubernetes and Red Hat OpenShift. Run UrbanCode Deploy itself in a container on IBM Cloud Private, IBM Cloud Public, or Red Hat OpenShift for easy use and administration.

Reference:

https://www.ibm.com/support/knowledgecenter/SS4GSP_7.0.4/com.ibm.udeploy.doc/topics/c_node_overview.html

Announcement:

https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?infotype=an&subtype=ca&appname=gpateam&supplier=897&letternum=ENUS220-051#abstrx

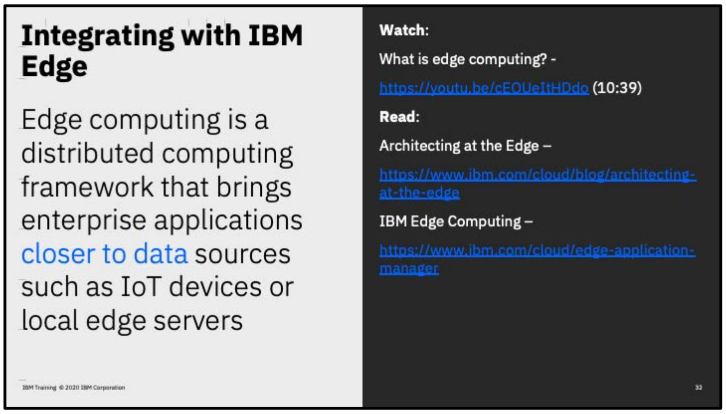


IBM Cloud Pak for Applications DevOps Add-On V3.0

The Cloud DevOps offering is a complement to Cloud Pak for Applications for two major use cases:

Cloud Native: For clients concerned about DevOps governance and auditability, Cloud DevOps tracks and reports who deployed what-where-when when layered over open source or on its own. Orchestrates multiple pipelines when apps have many dev teams contributing.

Modernization: As clients modernize applications, there will be a mix of new and existing apps as they innovate and use their existing investments. IBM Cloud DevOps enables continuous delivery of these multi-cloud applications and can deploy any application anywhere: from tWAS on VMs, to z Systems, to containers on Kubernetes, and to public/private clouds, and integration with other deployment tools such as Tekton and Jenkins.



Edge computing is a distributed computing framework that brings enterprise applications closer to data sources such as IoT devices or local edge servers. This proximity to data at its source can deliver strong business benefits: faster insights, improved response times and better bandwidth availability.

[Check out these resources to learn more.]

Watch:

What is edge computing? -

https://youtu.be/cEOUeItHDdo (10:39)

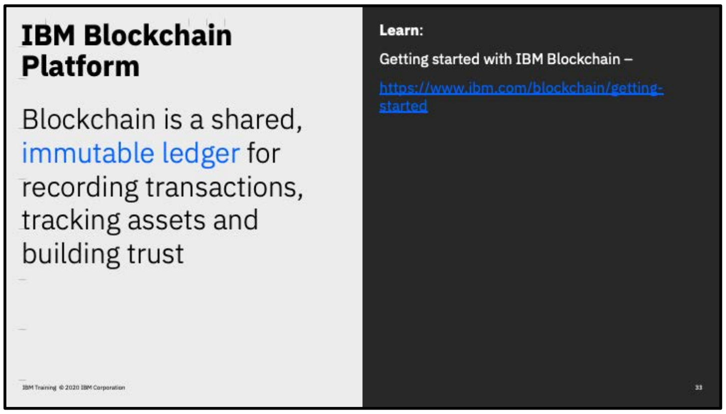
Read:

Architecting at the Edge –

https://www.ibm.com/cloud/blog/architecting-at-the-edge

IBM Edge Computing –

https://www.ibm.com/cloud/edge-application-manager



Blockchain is a shared, immutable ledger for recording transactions, tracking assets and building trust. If you are new to Blockchain, you can learn more at this link.

Learn:

Getting started with IBM Blockchain –

https://www.ibm.com/blockchain/getting-started

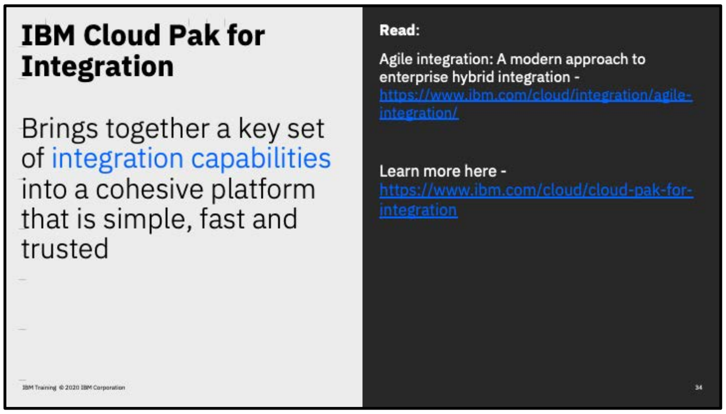
The IBM® Blockchain Platform continues to grow its presence in multicloud and is now available to run on Kubernetes or OpenShift Container Platform. This includes environments outside the IBM and Red Hat ecosystem like AWS, Azure, and Google Cloud. This offering is ideal if you want to deploy your components, store your data, or run your workloads on your own infrastructure or across public and private clouds for security, risk mitigation, preference, or compliance reasons.

The IBM® Blockchain Platform 2.5 enables a consortium of organizations to easily build and join a blockchain network *on-premises*, or on any private, public, or hybrid multicloud that uses Kubernetes. Customers can deploy their nodes on the cloud platform of their choice and connect to any IBM Blockchain Platform network, whether it is deployed on your own Kubernetes cluster or with the IBM Blockchain Platform for IBM Cloud. The IBM Blockchain Platform 2.5 leverages Hyperledger Fabric v1.4.7 and v2.x and supports deployment on multiple Kubernetes distributions.

Reference:

IBM Blockchain Platform –

https://www.ibm.com/blockchain/platform



Cloud Paks can be used together to further expand enterprise capabilities.

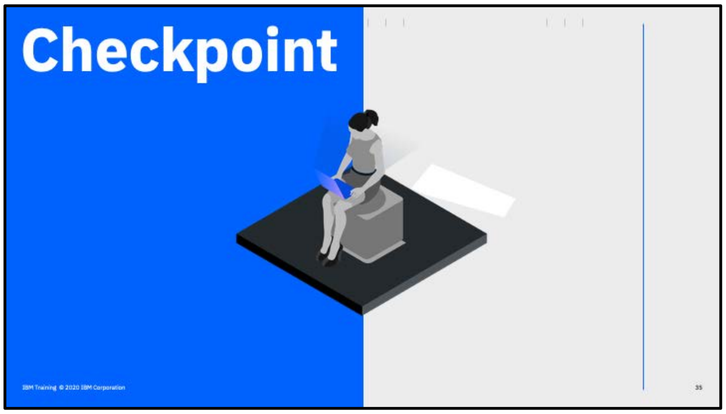
IBM Cloud Pak for Integration brings together a key set of integration capabilities into a cohesive platform that is simple, fast and trusted. It allows you to easily build powerful integrations and APIs in minutes, provides performance and scalability, and offers end-to-end capabilities with enterprise-grade security. It offers these capabilities:

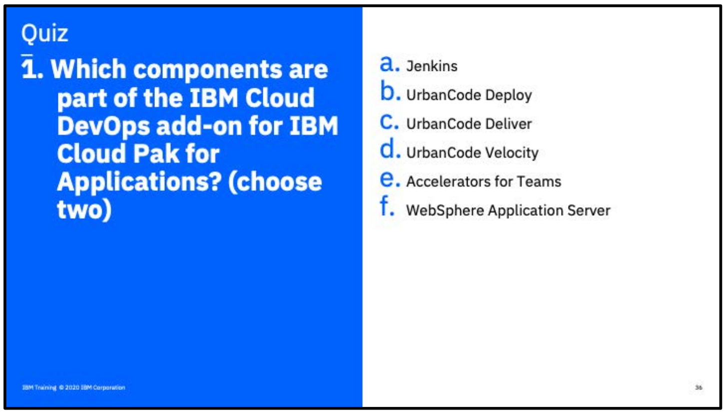
- Application and data integration: Connect applications and data sources on premises or in the cloud to coordinate the exchange of business information so that data is available when and where it's needed.
- API lifecycle management: Expose and manage business services as reusable APIs for select developer communities both internal and external to your organization. You can adopt an API strategy to accelerate how effectively they can share their unique data and services assets to fuel new applications and new business opportunities.
- Enterprise messaging: Ensure real-time information is available from anywhere at any time by providing reliable message delivery without message loss, duplication or complex recovery in the event of system or network issue.
- Event streaming: Analyze data associated with an event and respond to it in real time. Your business can tap into unused data, take advantage of real-time data insights and create responsive customer experiences.
- High-speed data transfer: Move huge amounts of data between on premises and cloud or cloud-to-cloud rapidly and predictably with enhanced levels of security to facilitate fast adoption of cloud platforms when data sets are very large.
- Secure gateway: Extend connectivity and integration beyond the enterprise with edge capabilities that protect APIs, the data they move, and the systems behind them.

Read:

Agile integration: A modern approach to enterprise hybrid integration - https://www.ibm.com/cloud/integration/agile-integration/

Learn more here - https://www.ibm.com/cloud/cloud-pak-for-integration

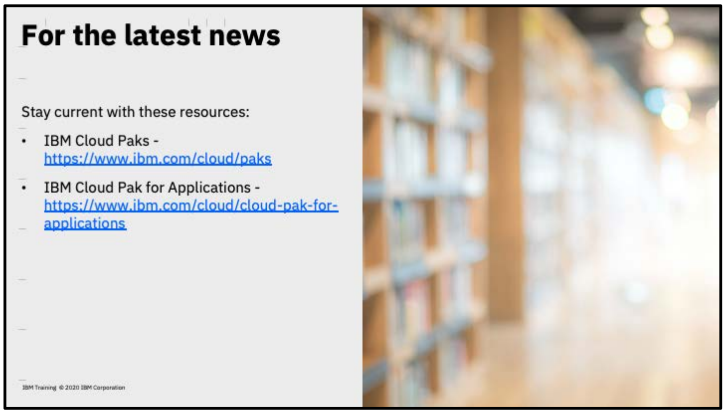

Checkpoint
IBM Training © 2020 IBM Corporation
35



Correct answer: b. and d.

Discussion prompt:

What other technologies would you consider including with Cloud Pak for Applications as part of your solution architecture?


For the latest news
Stay current with these resources:
• IBM Cloud Paks - https://www.ibm.com/cloud/paks
• IBM Cloud Pak for Applications - https://www.ibm.com/cloud/cloud-pak-for-applications
IBM Training © 2020 IBM Corporation

Unit 4: Building a cloud native solution architecture

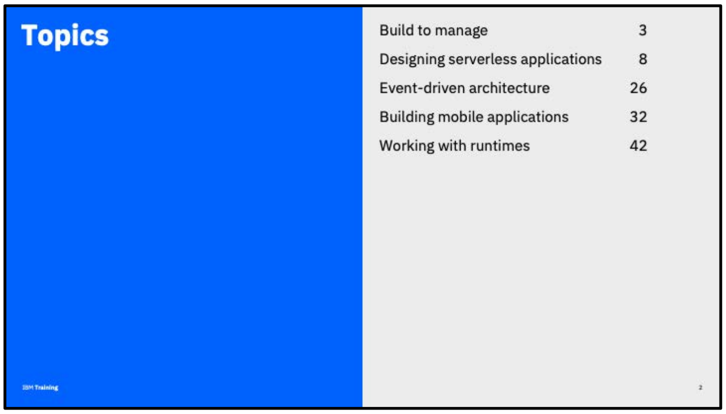
This unit discusses several options and considerations for building cloud native applications and describes the Cloud Pak for Applications components that support building a cloud native architecture.

Presentation time: about 40 minutes

Supplemental material:

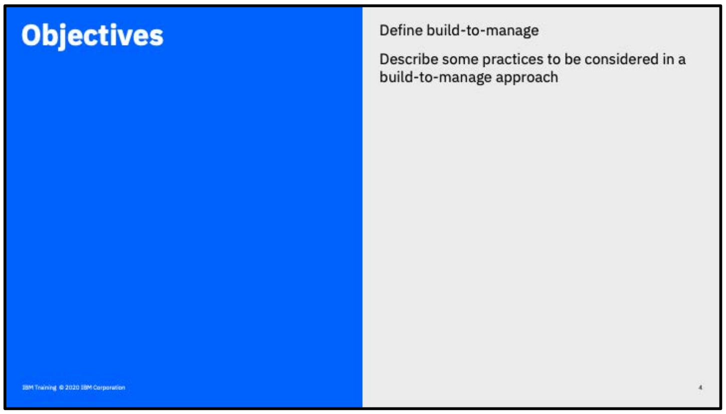
Videos: 11 minutes

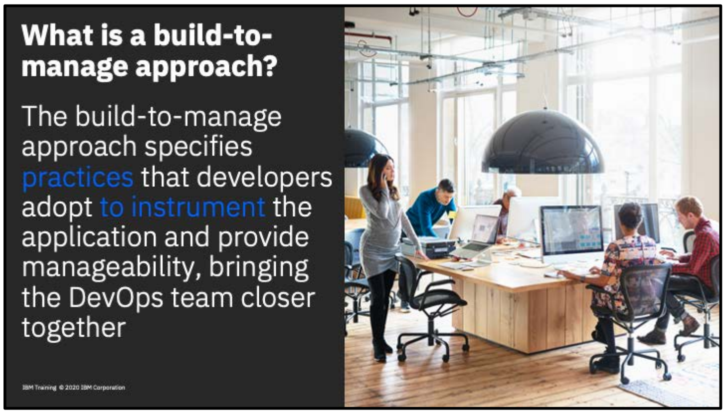
Readings: 1 hour and 15 minutes



This unit covers a range of related topics, from build to manage, to working with runtimes. It does not cover every possible aspect of building a cloud native architecture but focuses on the particular technologies that Cloud Pak for Applications is intended to support.

Build to
manage
IBM Training © 2020 IBM Corporation


Objectives
Define build-to-manage
Describe some practices to be considered in a build-to-manage approach



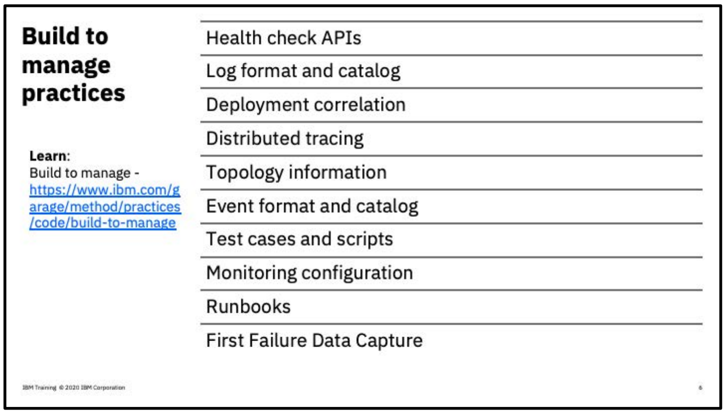
Traditionally, a development team was measured on how fast it updated features and released them into production. The operations team was measured on availability, which resulted in resistance to change. It's easy to see that these goals are diametrically opposed. In the traditional world, your team had time to build knowledge. Applications were infrequently updated, and after they were deployed, the lifetime of the application spanned years.

As you adopt practices that increase velocity and the speed of change, operations can become a bottleneck, leading to long release times or increased operational risk. To address this problem, create DevOps teams with a broad set of skills and common goals. All the team members are empowered to use their unique skills to drive the team towards overall success. The knowledge of your skilled operations team members helps your developers create more robust software.

Because continuous deployment is key in delivering cloud-based applications, the Ops part of your DevOps team has much less time to build and apply knowledge to prepare for each deployment. To address this reality, you need a different approach to operational management: build to manage.

The build-to-manage approach specifies practices that developers adopt to instrument the application and provide manageability, bringing the DevOps team closer together.

As development and operations come closer together, new practices arise to ease operations for cloud-based applications.



Build to manage specifies a set of practices that developers can adopt to instrument the application and provide manageability aspects as part of the release. When you implement a build-to-manage approach, consider these practices:

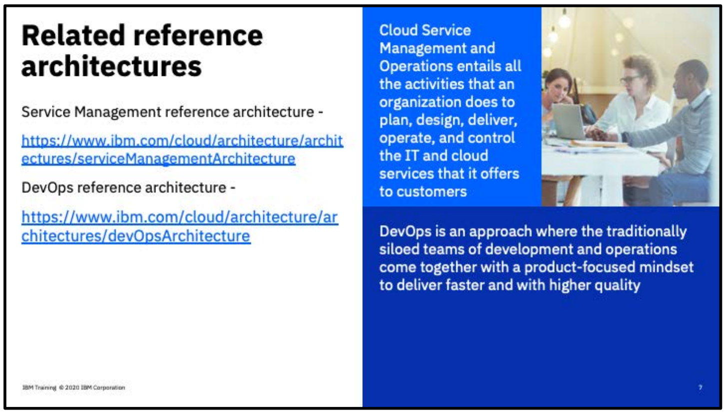
- Health check APIs
- Log format and catalog
- Deployment correlation
- Distributed tracing
- Topology information
- Event format and catalog
- Test cases and scripts
- Monitoring configuration
- Runbooks
- First Failure Data Capture

By adopting those practices, your organization achieves a more mature operational level and faster velocity. Your DevOps team comes closer together as it works toward the common goal of quickly releasing robust functions that meet the required functional, availability, performance, and security objectives.

Learn:

Build to manage -

https://www.ibm.com/garage/method/practices/code/build-to-manage



Cloud Service Management and Operations entails all the activities that an organization does to plan, design, deliver, operate, and control the IT and cloud services that it offers to customers.

Service management includes the operational aspects of your applications and services. After an application is pushed to production, it must be managed. Applications are monitored to ensure availability and performance according to service level agreements (SLAs) or service level objectives (SLOs).

DevOps is an approach where the traditionally siloed teams of development and operations come together with a product-focused mindset to deliver faster and with higher quality. Over time, other stakeholders, such as quality assurance (testing), security, and even users can come together to collaborate to continuously deliver software. By following this approach, your business can seize market opportunities and reduce the time to include customer feedback in its products.

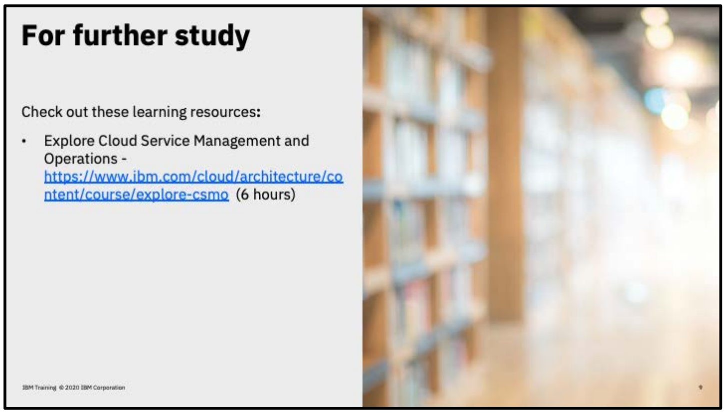
DevOps is a people, process, and technology transformation. Increasing velocity involves automation, removing organizational and behavioral impedance, and optimizing software delivery. Velocity must be balanced with the appropriate amount of risk mitigation, such as testing to identify defects earlier.

[DevOps is covered in more detail later in the course.]

Discussion prompt:

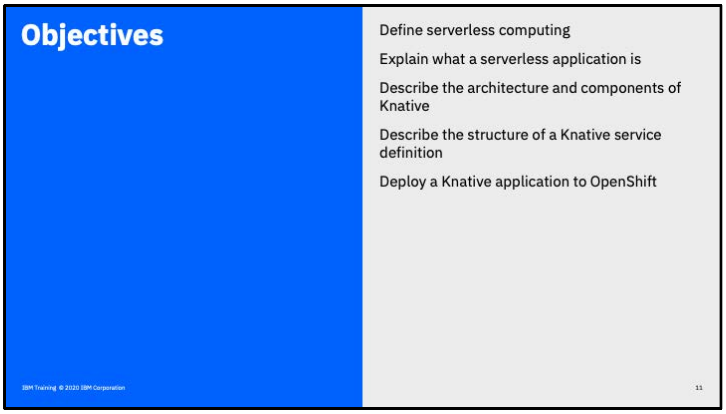
What changes would be necessary for your organization to adopt a build-to-manage approach?

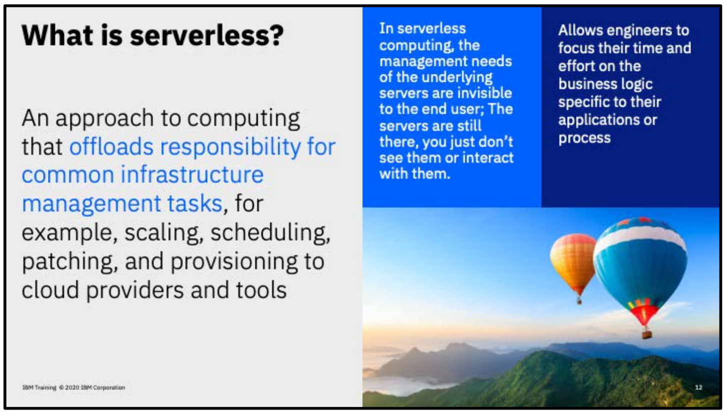
IBM Training © 2020 IBM Corporation



To learn more about adopting a build to manage approach, see these related architectures for Service Management and DevOps, and optionally, take the Explore Cloud Service Management and Operations course.


Designing serverless applications
IBM Training © 2020 IBM Corporation
10


Objectives
Define serverless computing
Explain what a serverless application is
Describe the architecture and components of Knative
Describe the structure of a Knative service definition
Deploy a Knative application to OpenShift
IBM Training © 2020 IBM Corporation
11



Serverless is an approach to computing that offloads responsibility for common infrastructure management tasks (like, scaling, scheduling, patching, provisioning, and so on) to cloud providers and tools, allowing engineers to focus their time and effort on the business logic specific to their applications or process.

The most useful way to define and understand serverless is focusing on the handful of core attributes that distinguish serverless computing from other compute models, namely:

Serverless computing runs code only on-demand on a per-request basis, scaling transparently with the number of requests being served.

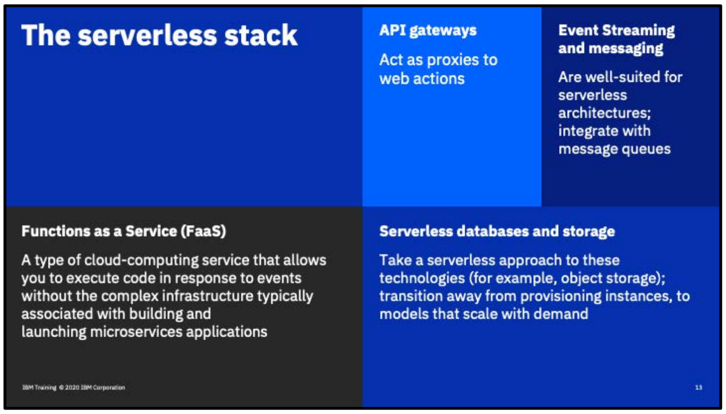
Serverless computing enables end users to pay only for resources being used, never paying for idle capacity.

Serverless is fundamentally about spending more time on code, less on infrastructure.

In serverless computing, the management needs of the underlying servers are invisible to the end user; The servers are still there, you just don't see them or interact with them.

Reference:

https://www.ibm.com/cloud/learn/serverless



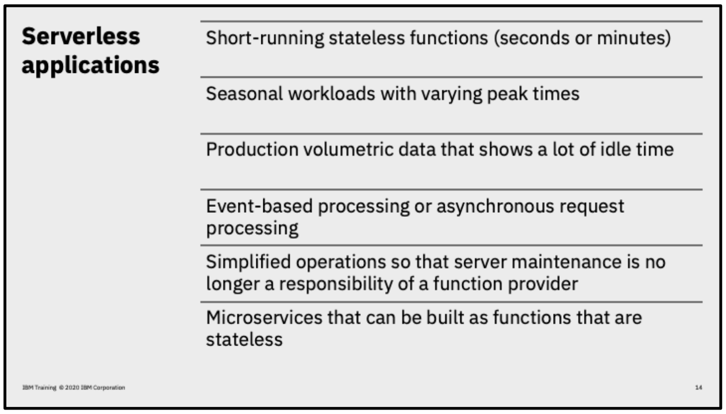
Defining serverless as a set of common attributes, instead of an explicit technology, makes it easier to understand how the serverless approach can manifest in other core areas of the stack.

Functions as a Service (FaaS): FaaS is widely understood as the originating technology in the serverless category. It represents the core compute/processing engine in serverless and sits in the center of most serverless architectures.

Serverless databases and storage: Databases and storage are the foundation of the data layer. A “serverless” approach to these technologies (with object storage being the prime example within the storage category) involves transitioning away from provisioning “instances” with defined capacity, connection, and query limits and moving toward models that scale linearly with demand, in both infrastructure and pricing.

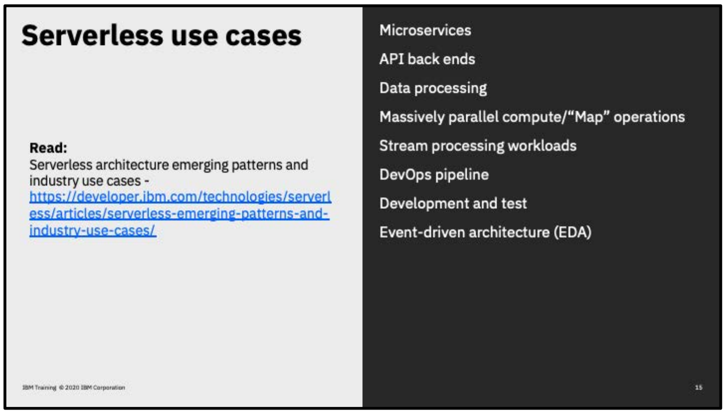
Event streaming and messaging: Serverless architectures are well-suited for event-driven and stream-processing workloads, which involve integrating with message queues, most notably Apache Kafka.

API gateways: API gateways act as proxies to web actions and provide HTTP method routing, client ID and secrets, rate limits, CORS, viewing API usage, viewing response logs, and API sharing policies.



Serverless applications often share these characteristics:

- Short-running stateless functions (seconds or minutes)
- Seasonal workloads with varying peak times
- Production volumetric data that shows a lot of idle time
- Event-based processing or asynchronous request processing
- Needs to simplify operations so that server maintenance is no longer a responsibility of a function provider organization (for example, the ability to automatic scaling without the need to configure the underlying PaaS)
- Microservices that can be built as functions that are stateless



The most common use case of serverless today is supporting [microservices architectures](). While microservices can also be built and operated using either VMs or containers, serverless gained significant momentum given its attributes around small bits of code that do one thing, inherent and automatic scaling, rapid provisioning, and a pricing model that does not charge for idle capacity.

Given its unique combination of attributes and benefits, serverless architectures are well-suited for use cases around data and event processing, IoT, microservices, and mobile backends.

API backends

Any action (or function) in a serverless platform can be turned into a HTTP endpoint ready to be consumed by web clients. When enabled for web, these actions are called web actions. Once you have web actions, you can assemble them into a full-featured API with an API Gateway that brings additional security, OAuth support, rate limiting, and custom domain support.

Data processing

Serverless is well-suited to working with structured text, audio, image, and video data.

Massively parallel compute/"Map" operations

Any kind of parallel task is very well-suited to be run on a serverless runtime. Each task results in one action invocation.

Stream processing workloads

Combining managed Apache Kafka with FaaS and database/storage offers a powerful foundation for real-time buildouts of data pipelines and streaming apps. These architectures are ideally suited for working with all sorts of data stream ingestions (for validation, cleansing, enrichment, transformation

In addition, each of these use cases is a fast-emerging pattern for using serverless architecture in application design:

The **DevOps pipeline** is emerging as one of the top candidates for adoption of serverless computing (for example, functions that address operational issues by taking corrective actions in response to an operational event).

Development and test environments using serverless functions made significant innovations toward optimal consumption of resources. Examples include functions that bring down development and test resources when no users are logged in and that bring relevant resources up when a user logs in (based on user profile and preferences).

EDA: Applications act on events triggered by internal and external services or sources. Tasks are scheduled according to a specific time or event, such as trigger fraud analytics in response to a suspicious activity. An anomaly detected from sensor input triggers performing analytics. A database change triggers running application logic (for example, to enable change data capture on select datasets). Business events cause analytics to be performed and images to be processed. For example, as soon as an image is generated, medical imaging analytics generate thumbnails for an image upload.

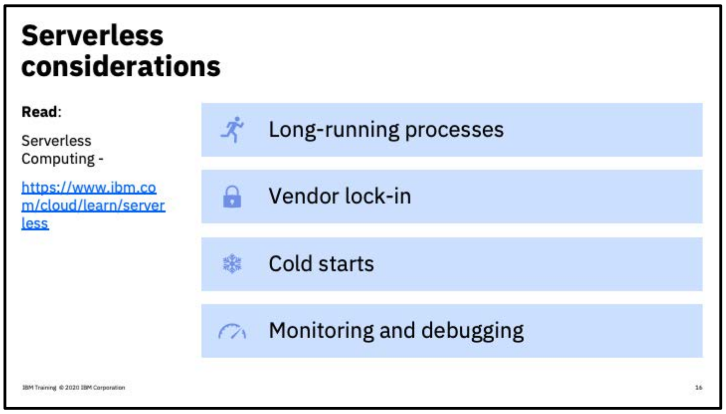
Read:

Serverless architecture emerging patterns and industry use cases -

https://developer.ibm.com/technologies/serverless/articles/serverless-emerging-patterns-and-industry-use-cases/

Reference:

https://www.ibm.com/cloud/learn/serverless



Serverless computing is not appropriate for every application. There are some challenges and limitations to consider, such as:

Long-running processes

A lack of cost savings with long-running processes. serverless workloads are designed to scale up and down in response to workload, so there is a significant advantage for fluctuating workloads. But for workloads characterized by long-running processes, these same cost advantages are no longer realized and managing a traditional server environment might be simpler and more cost-effective.

Vendor lock-in

Serverless architectures are designed to take advantage of an ecosystem of managed cloud services and, in terms of architectural models, go the furthest to decouple a workload from something more portable, like a VM or a container. For some companies, deeply integrating with the native managed services of cloud providers is where much of the value of cloud can be found; for other organizations, these patterns represent material lock-in risks that need to be mitigated.

Cold starts

Because serverless architectures forgo long-running processes in favor of scaling up and down to zero, they also sometimes need to start up from zero to serve a new request. For certain applications, this delay isn't much of an impact, but for something like a low-latency financial application, this delay is not acceptable.

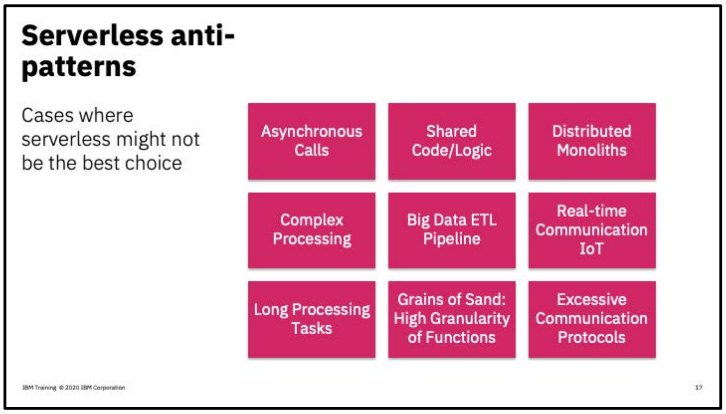
Monitoring and debugging

Monitoring and debugging operational tasks are challenging in any distributed system, and the move to both microservices and serverless architectures can only exacerbated the complexity associated with managing these environments carefully.

Read:

Serverless Computing -

https://www.ibm.com/cloud/learn/serverless

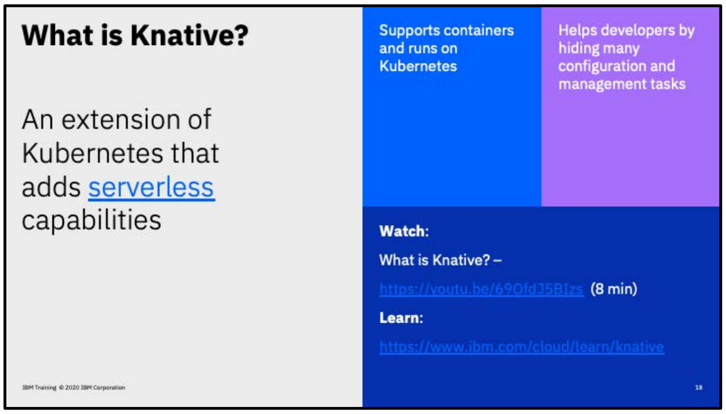


Some other “bad” ideas include:

- Asynchronous Calls – With serverless, you only pay for resources consumed, but processing serverless asynchronous requests can involve wait times – you still have to pay for that. So this is a pattern to avoid.
- Shared Code/Logic – can lead to too-close coupling between services. This practice crosses the isolation barrier, reduces the effectiveness of your serverless architecture and hampers its scalability.
- Distributed Monoliths – happen when a change to one function often requires change to all the functions. In this case, once again, you lose the benefits of loose coupling.
- For complex processing like in the case of video processing you, run into limitations of the computing power for a single function.
- Big Data extract, transform, load (ETL) Pipeline – with big data payloads, data is pulled in from a queue, database or other data source. The more data transmission you have, the more risk for data breaches.
- Real-time Communication Internet of Things (IoT) – most IoT use cases are highly sensitive to latency, so this is also a pattern to avoid.
- Long Processing Tasks – you want to avoid for reasons already mentioned.
- Grains of Sand: High Granularity of Functions – results from over-engineering, and can lead to problems with monitoring, managing, and security.
- Excessive Communication Protocols – result in complex systems. The goal is to find the right balance between asynchronous and synchronous protocols.

Reference:

https://www.simform.com/serverless-antipatterns/



Knative is an extension of Kubernetes that hides many of the management and configuration tasks from you and adds [serverless](#) capabilities.

Google, in close collaboration with IBM, Red Hat, Pivotal, and SAP, developed Knative as an open source platform. It supports [containers](#) and runs on Kubernetes.

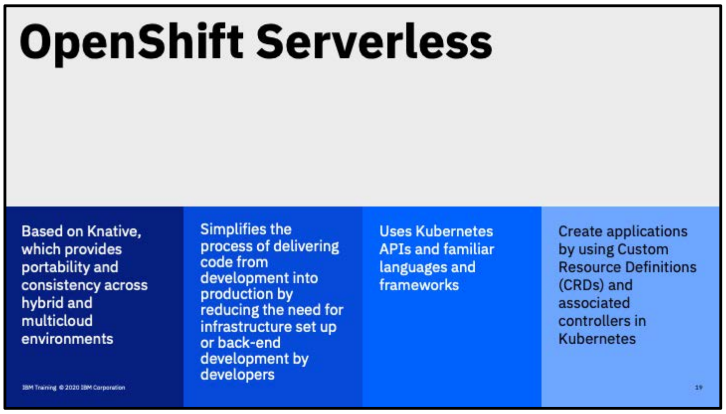
Watch:

What is Knative? –

https://youtu.be/69OfdJ5BIzs (8 min)

Learn:

https://www.ibm.com/cloud/learn/knative

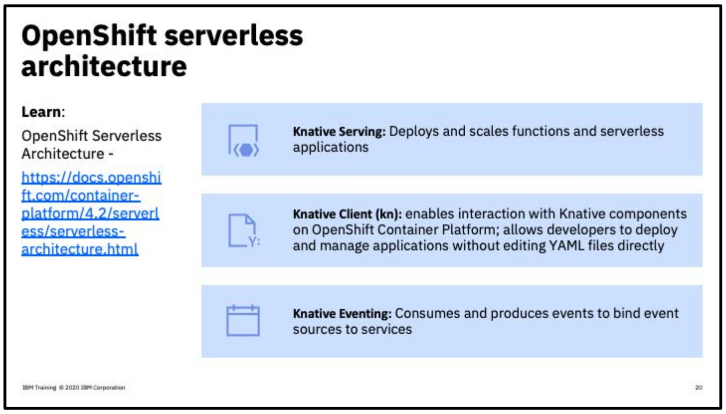


OpenShift Serverless simplifies the process of delivering code from development into production by reducing the need for infrastructure set up or back-end development by developers.

Developers on OpenShift Serverless can use the provided Kubernetes-native APIs, as well as familiar languages and frameworks, to deploy applications and container workloads.

OpenShift Serverless is based on the open source Knative project, which provides portability and consistency across hybrid and multi-cloud environments by enabling an enterprise-grade serverless platform.

You create applications by using Custom Resource Definitions (CRDs) and associated controllers in Kubernetes and package them as OCI compliant Linux containers that can be run anywhere.



Knative Serving on OpenShift Container Platform builds on Kubernetes and Istio to support deploying and serving serverless applications.

It deploys and scales functions by creating a set of Kubernetes Custom Resource Definitions (CRDs) that are used to define and control the behavior of serverless workloads on an OpenShift Container Platform cluster.

These CRDs can be used as building blocks to address complex use cases, such as rapid deployment of serverless containers, automatic scaling of Pods, routing and network programming for Istio components, or viewing point-in-time snapshots of deployed code and configurations.

Knative Serving: Deploy and scale functions and serverless applications. Knative Serving can be installed on OpenShift Container Platform using the OpenShift Serverless Operator.

The Knative Client (kn) extends the functionality of the oc or kubectl tools to enable interaction with Knative components on OpenShift Container Platform. kn allows developers to deploy and manage applications without editing YAML files directly.

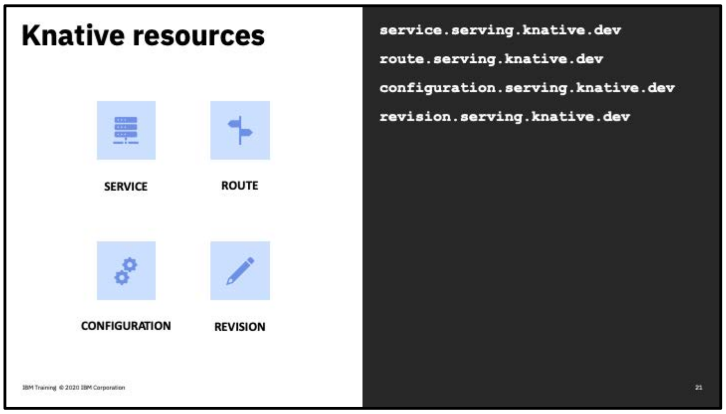
A developer preview version of Knative Eventing is available for use with OpenShift Serverless. However, this is not included in the OpenShift Serverless Operator and is not currently supported as part of this Technology Preview. For more information about Knative Eventing, including installation instructions and samples, see the Knative Eventing on OpenShift Container Platform documentation - https://openshift-knative.github.io/docs/docs/index.html

Knative Eventing: Consume and produce events to bind event sources to services. Knative Eventing can be installed on OpenShift Container Platform using the Knative Eventing Operator.

Learn:

OpenShift Serverless Architecture -

https://docs.openshift.com/container-platform/4.2/serverless/serverless-architecture.html



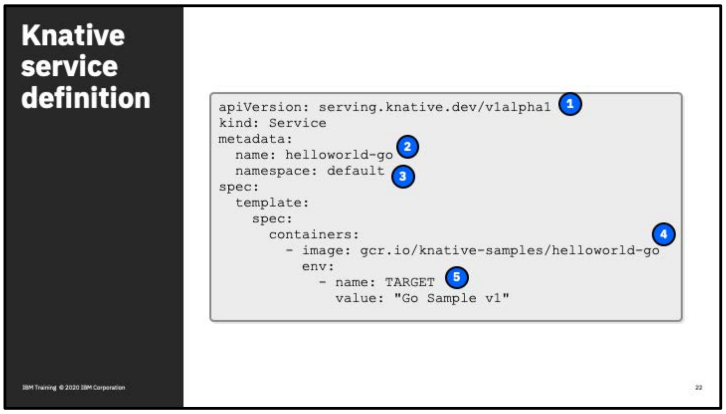
These components are the resources that Knative Serving requires to be configured and run correctly.

Knative service resource: The service.serving.knative.dev resource automatically manages the whole lifecycle of a serverless workload on a cluster. It controls the creation of other objects to ensure that an app has a route, a configuration, and a new revision for each update of the service. Services can be defined to always route traffic to the latest revision or to a pinned revision.

Knative route resource: The route.serving.knative.dev resource maps a network endpoint to one or more Knative revisions. You can manage the traffic in several ways, including fractional traffic and named routes.

Knative configuration resource:The configuration.serving.knative.dev resource maintains the required state for your deployment. Modifying a configuration creates a new revision.

Knative revision resource: The revision.serving.knative.dev resource is a point-in-time snapshot of the code and configuration for each modification made to the workload. Revisions are immutable objects and can be retained for as long as needed. Cluster administrators can modify the revision.serving.knative.dev resource to enable automatic scaling of Pods in your OpenShift Container Platform cluster.



Knative services are Kubernetes services that you create to deploy a serverless application. Youy define each Knative service by a route and a configuration that is contained in a .yaml file.

To create a service, you must create the service.yaml file.

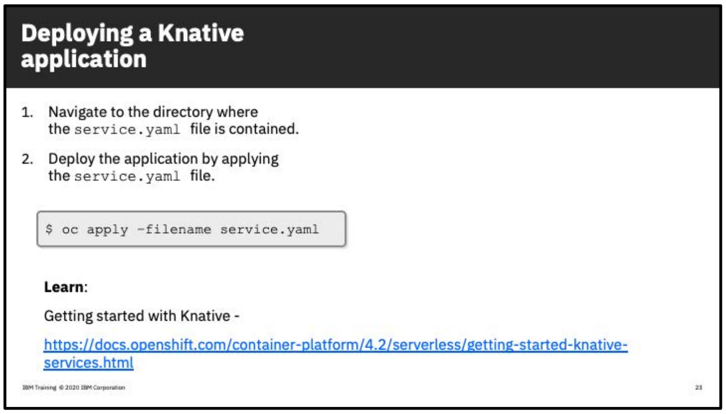
This example will create a simple golang application called helloworld-go and allows you to specify the image for that application.

Here, you can see

1. Current version of Knative
2. The name of the application
3. The namespace the application will use
4. The URL to the image of the application
5. The environment variable printed out by the sample application

Reference:

https://docs.openshift.com/container-platform/4.2/serverless/getting-started-knative-services.html



To deploy a serverless application, you must apply the service.yaml file.

Navigate to the directory where the service.yaml file is contained.

Deploy the application by applying the service.yaml file.

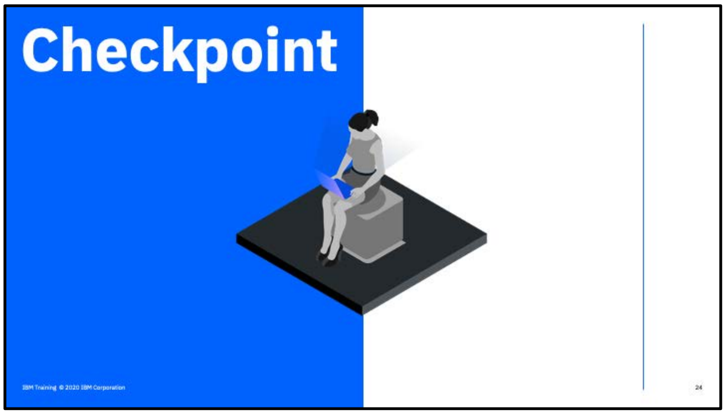
$ oc apply --filename service.yaml

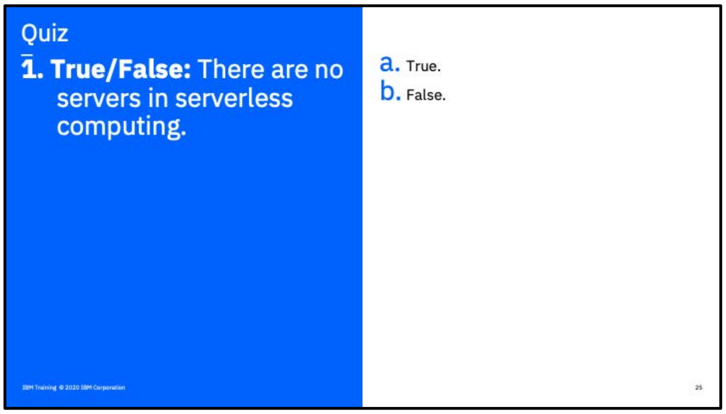
Now that service has been created and the application has been deployed, Knative will create a new immutable revision for this version of the application.

Knative will also perform network programming to create a route, ingress, service, and load balancer for your application, and will automatically scale your pods up and down based on traffic, including inactive pods.

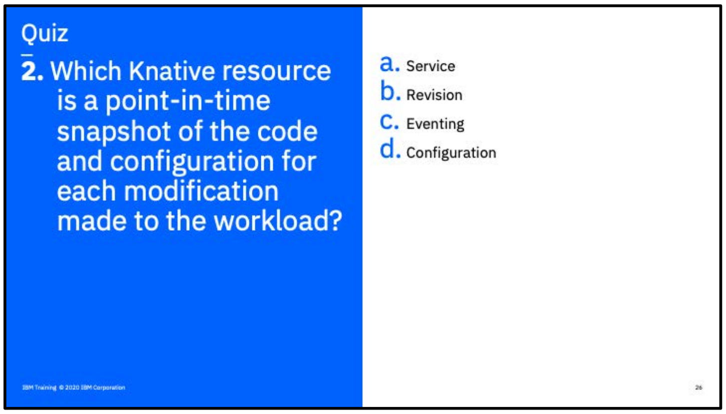
Learn:

Getting started with Knative - https://docs.openshift.com/container-platform/4.2/serverless/getting-started-knative-services.html

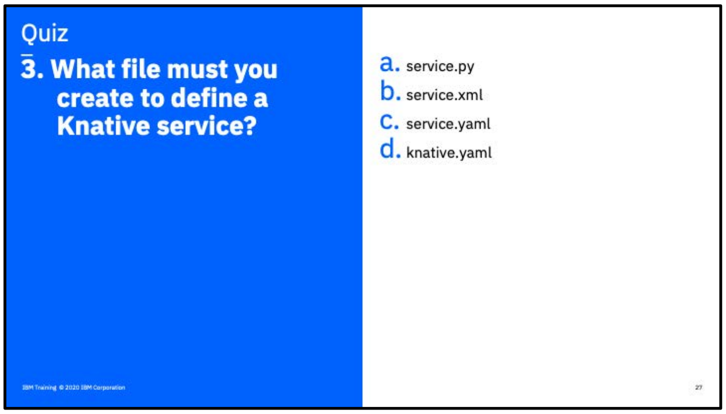

Checkpoint
IBM Training © 2020 IBM Corporation
24



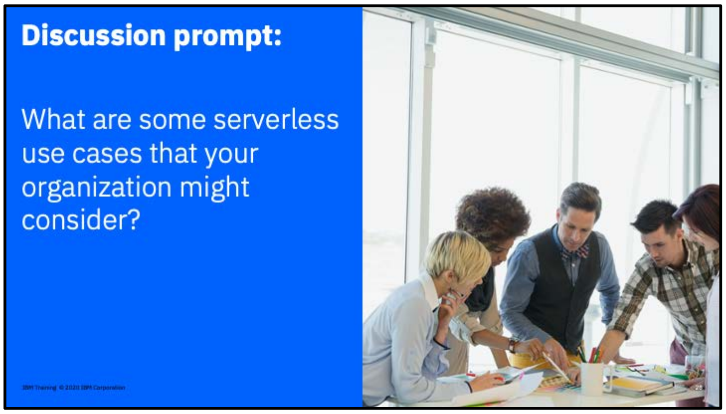
Correct answer: b.



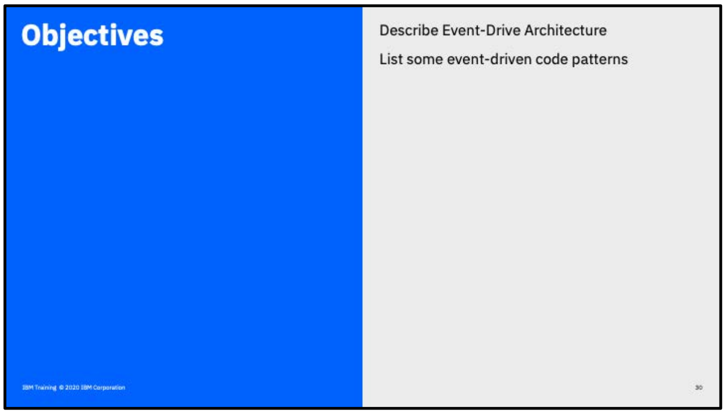
Correct answer: b

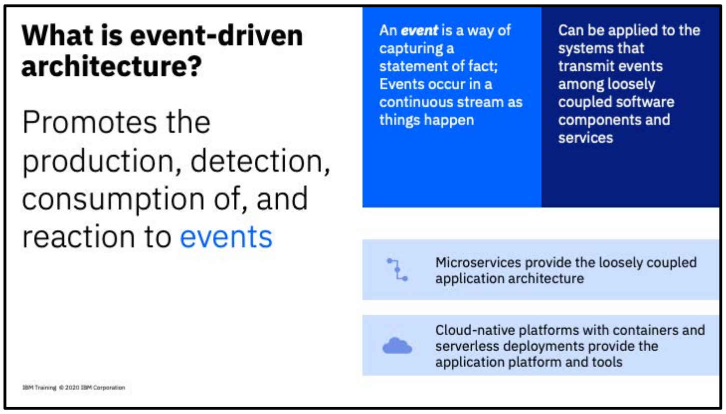


Correct answer: c.


Discussion prompt:
What are some serverless use cases that your organization might consider?


Event-driven
architecture
IBM Training © 2020 IBM Corporation
29


Objectives
Describe Event-Drive Architecture
List some event-driven code patterns
IBM Training © 2020 IBM Corporation
30



Event-driven architecture (EDA) is an architecture pattern that promotes the production, detection, consumption of, and reaction to events. This architectural pattern can be applied to the systems that transmit events among loosely coupled software components and services.

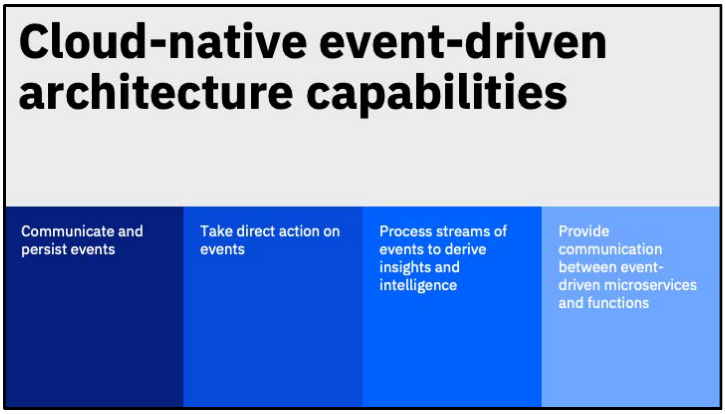
An *event* is a way of capturing a statement of fact. Events occur in a continuous stream as things happen. By taking advantage of this continuous stream, applications can react and reason about the future based on what happened in the past.

For enterprise IT teams, embracing event-driven development is foundational to the next generation of digital business applications. IT teams must design, develop, deploy, and operate event-driven solutions in cloud-native styles.

Event-driven architectures and reactive programming models aren't new concepts. However, when you move to cloud-native architectures with microservices, container-based workloads, and serverless computing, you can revisit event-driven approaches in the cloud-native context. Think of event-driven architectures as extending the resilience, agility, and scalable characteristics of cloud-native architectures to also be reactive and responsive.

Two aspects of a cloud-native architecture are essential to developing an event-driven architecture:

- Microservices provide the loosely coupled application architecture that enables deployment in highly distributed patterns for resilience, agility, and scale.
- Cloud-native platforms with containers and serverless deployments provide the application platform and tools that realize the resilience, agility, and scalability of the microservices architectures.

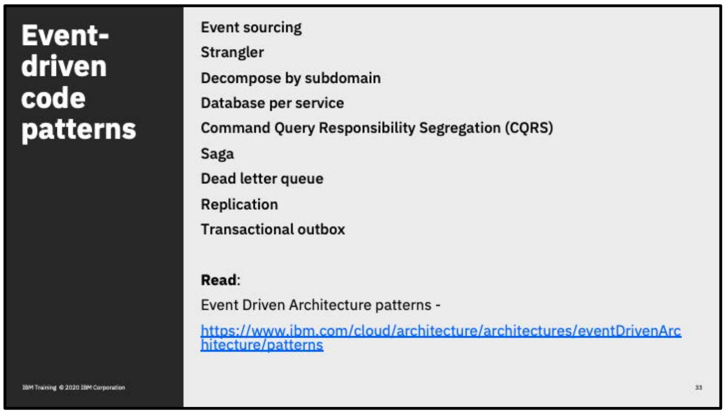


A cloud-native event-driven architecture must support at least these capabilities:

- Communicate and persist events
- Take direct action on events
- Process streams of events to derive insights and intelligence, and
- Provide communication between event-driven microservices and functions

Reference:

https://www.ibm.com/cloud/architecture/architectures/eventDrivenArchitecture/reference-architecture

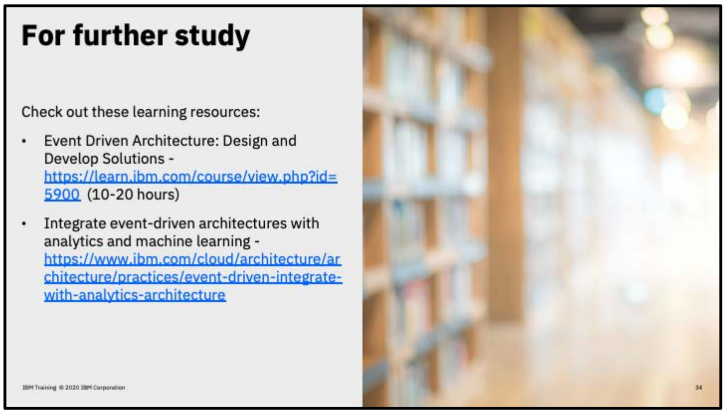


When you adopt messaging (publish/subscribe) as a microservice communication approach, you need to implement at least some of the following event-driven patterns.

Some of these patterns might look familiar because they are also common microservice patterns, such as the Strangler pattern. You can see a description of each one listed here in the IBM Cloud architecture center.

Reference:

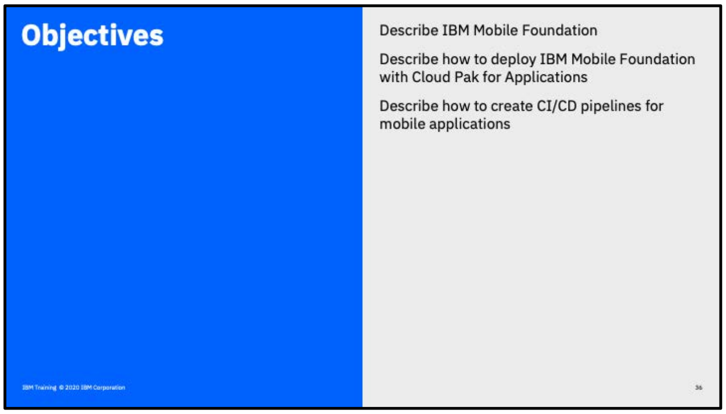
https://www.ibm.com/cloud/architecture/architectures/eventDrivenArchitecture/patterns

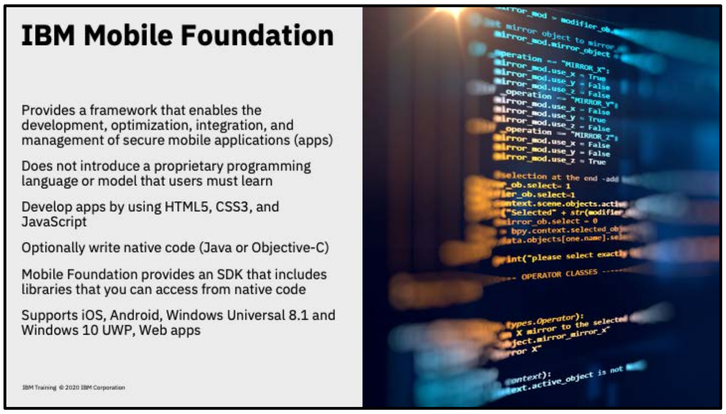


In this self-paced course, you learn about the EDA Reference Architecture, different patterns, its common use-cases and tools that are available for developing and running event-driven applications. Depending on your role and need, the duration is estimated between 10 to 20 hours.

This article explains how to integrate data science, artificial intelligence, and machine learning into an event-driven solution.


Building
mobile
applications
IBM Training © 2020 IBM Corporation
35


Objectives
Describe IBM Mobile Foundation
Describe how to deploy IBM Mobile Foundation with Cloud Pak for Applications
Describe how to create CI/CD pipelines for mobile applications



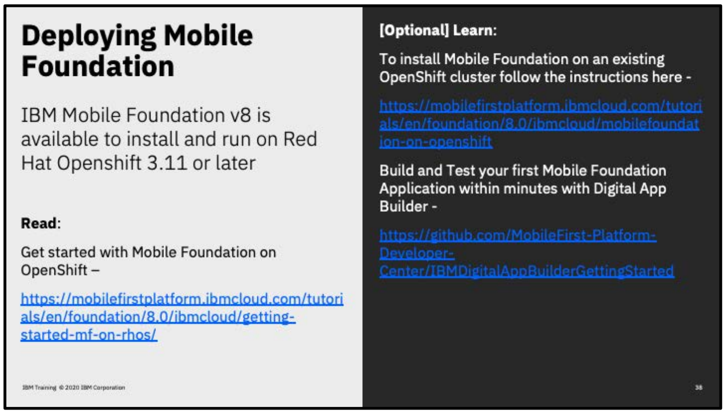
IBM Mobile Foundation provides a framework that enables the development, optimization, integration, and management of secure mobile applications (apps). Mobile Foundation does not introduce a proprietary programming language or model that users must learn.

You can develop apps by using HTML5, CSS3, and JavaScript. You can optionally write native code (Java or Objective-C). Mobile Foundation provides an SDK that includes libraries that you can access from native code.

With IBM Mobile Foundation, you can use capabilities such as development, testing, back-end connections, push notifications, offline mode, update, security, analytics, monitoring, and application publishing.

The Mobile Foundation SDKs support the following platforms:

- iOS
- Android
- Windows Universal 8.1 and Windows 10 UWP
- Web apps



IBM Mobile Foundation v8 is available to install and run on Red Hat Openshift 3.11 or later.

To install Mobile Foundation on an existing OpenShift cluster follow the instructions here - https://mobilefirstplatform.ibmcloud.com/tutorials/en/foundation/8.0/ibmcloud/mobilefoundation-on-openshift.

Developing Applications

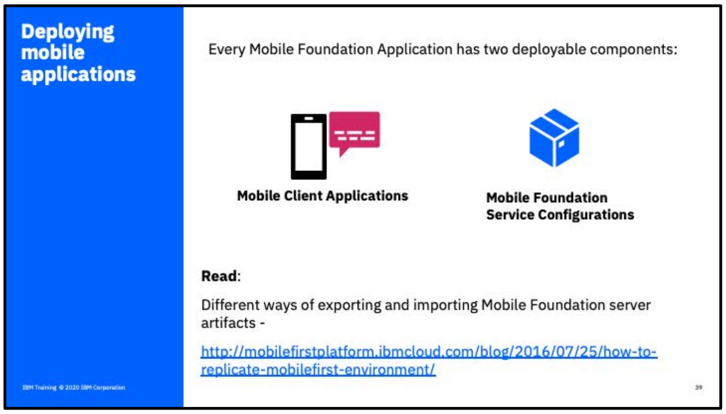
You can quickly and easily develop Mobile Applications that use Mobile Foundation Lifecycle Management, Security, Engagement and Analytics by using the IBM Digital App Builder (DAB) Tool. This tool also provides mobile application accelerators for secure connectivity to back end microservices.

Build and Test your first Mobile Foundation Application within minutes - get started with IBM Digital App Builder

Read:

Get Started with Mobile Foundation on OpenShift -

https://mobilefirstplatform.ibmcloud.com/tutorials/en/foundation/8.0/ibmcloud/getting-started-mf-on-rhos/



Every Mobile Foundation Application has two deployable components:

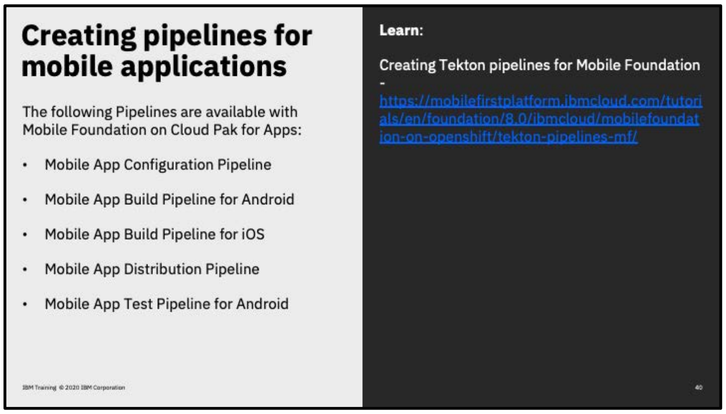
- Mobile Client Applications which can be deployed to the Mobile Foundation App Center or any other public App Store
- Mobile Foundation Service Configurations for Application Lifecycle, Security, Push Notifications, LiveUpdate. These configurations can be exported from Mobile Foundation development enviroment and imported into a Mobile Foundation staging or production environment.

Refer to the following documentation for more information related to exporting and importing Mobile Foundation Service configurations across deployments.

Read:

Different ways of exporting and importing Mobile Foundation server artifacts **-**

http://mobilefirstplatform.ibmcloud.com/blog/2016/07/25/how-to-replicate-mobilefirst-environment/

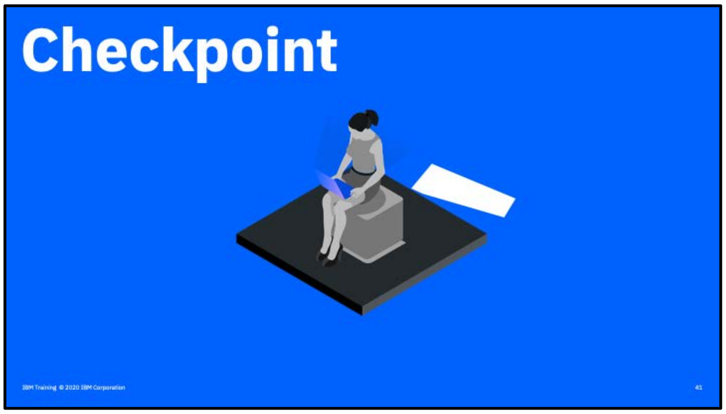


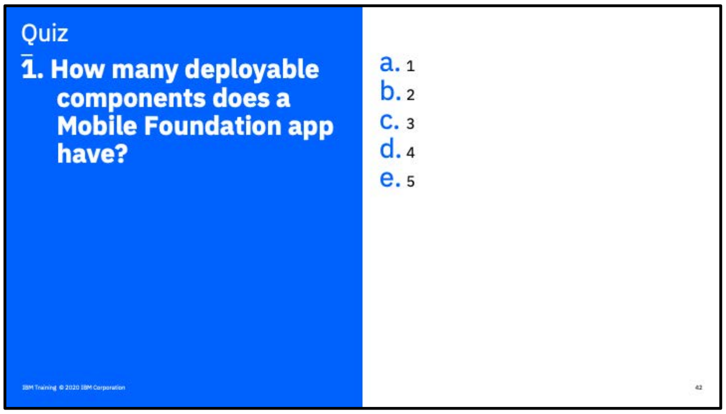
Mobile Foundation in Cloud Pak for Apps is bundled with OpenShift pipelines for automating common DevOps tasks related to Mobile Foundation.

Following Tekton Pipelines are available with Mobile Foundation on Cloud Pak for Apps:

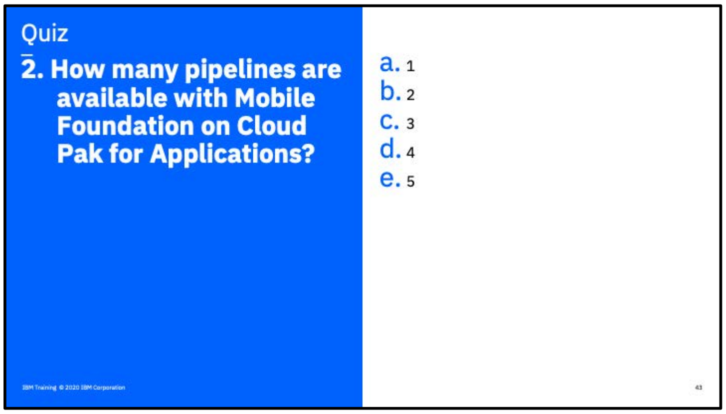
- Mobile App Configuration Pipeline (*mobile-app-registration-pipeline*)
- Mobile App Build Pipeline for Android (*mobile-app-build-android-pipeline*)
- Mobile App Build Pipeline for iOS (*mobile-app-build-ios-pipeline*)
- Mobile App Distribution Pipeline (*application-center-deploy*)
- Mobile App Test Pipeline for Android (*mobile-app-test-android-pipeline*)

Learn: https://mobilefirstplatform.ibmcloud.com/tutorials/en/foundation/8.0/ibmcloud/mobilefoun dation-on-openshift/tekton-pipelines-mf/

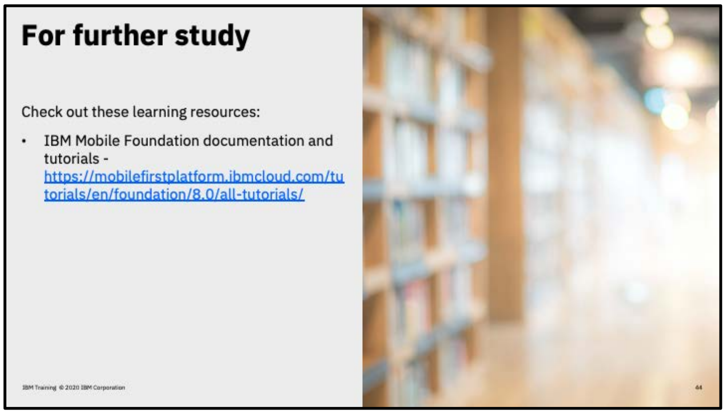

Checkpoint
IBM Training © 2020 IBM Corporation
41



Correct answer: b.



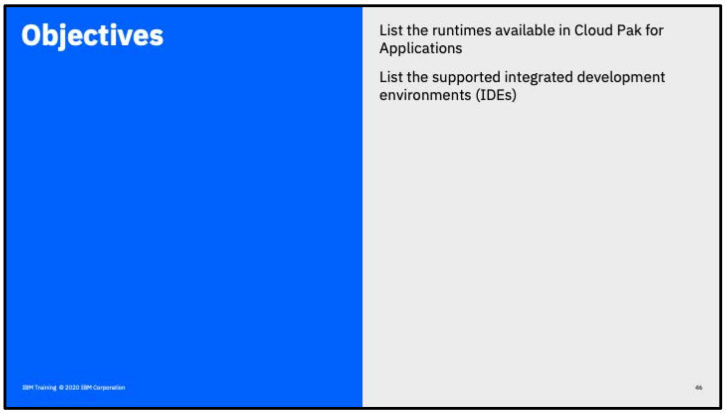
Correct answer: e.



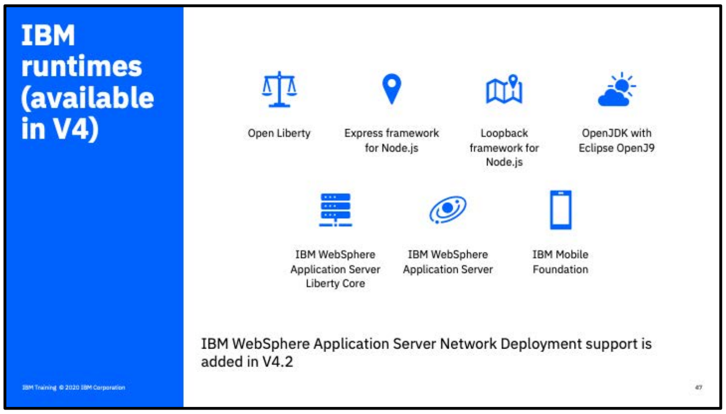
Check out these learning resources:

- IBM Mobile Foundation documentation and tutorials - https://mobilefirstplatform.ibmcloud.com/tutorials/en/foundation/8.0/all-tutorials/


Working with
runtimes
IBM Training © 2020 IBM Corporation
45



Describe the features and benefits of using Cloud Pak for Applications for cloud native development, application modernization, running existing applications, and delivering applications on multiple clouds



IBM® Cloud Pak for Applications V4 delivers Red Hat OpenShift Container Platform 4.2, accompanied by one of the most robust sets of application run times. It supports application workloads wherever they are currently running and provides the destination for modernization and new cloud-native workloads as you drive digital transformation and innovation projects.

IBM Cloud Pak for Applications provides support for the following bundled programs when deployed in containers onto Red Hat OpenShift Container Platform:

Red Hat OpenShift Container Platform 4.2

Red Hat Runtimes

IBM WebSphere® Application Server Liberty Core

IBM WebSphere Applications Server

IBM Mobile Foundation

Cloud Pak for Applications also provides support for run times from IBM, including:

Open Liberty

Express® framework for Node.js

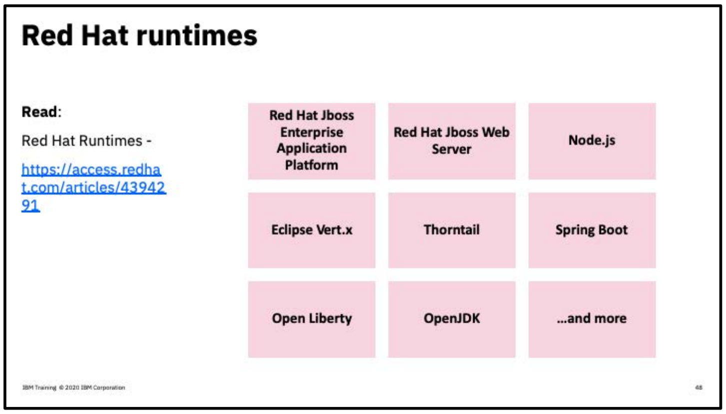
Loopback framework for Node.js

OpenJDK with Eclipse OpenJ9

IBM WebSphere Application Server Network Deployment is added in V4.2.

REFERENCE:

https://www-01.ibm.com/common/ssi/ShowDoc.wss?docURL=/common/ssi/rep_ca/4/897/ENUS219-574/index.html&lang=en&request_local&request_loca

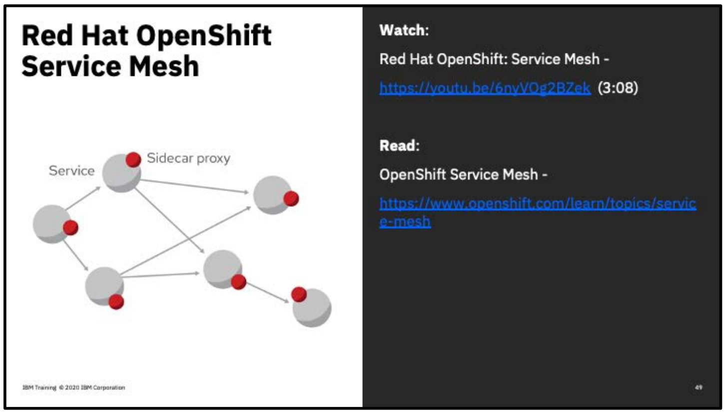


Customers of IBM's Cloud Pak for Applications product have access to the full portfolio of Red Hat's Runtimes and associated components. This article details the entitlements and resources available to IBM customers –

Read:

Read Hat Runtimes -

https://access.redhat.com/articles/4394291



Red Hat OpenShift Container Platform 4.2 also introduces support for Red Hat OpenShift Service Mesh, which enables clients to benefit from an end-to-end developer-focused experience, and support for disconnected installations, which allows clients to get the latest version of the Red Hat OpenShift Container Platform in environments not accessible through the internet, or with strict image testing policies.

OpenShift Service Mesh includes Istio. Istio is an open-source project for integrating and managing traffic flow across services. It works in concert with an underlying cluster manager (like Kubernetes). Centralized components, sidecar proxies, and node agents work together to create the data and control planes over a distributed application.

Watch:

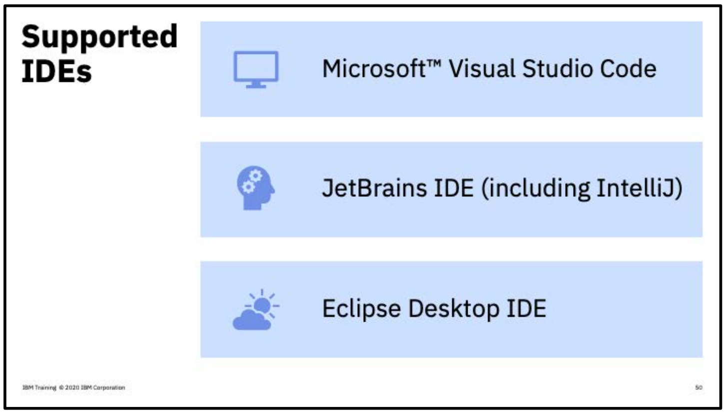
Red Hat OpenShift: Service Mesh -

https://youtu.be/6nyVOg2BZek (3:08)

Read:

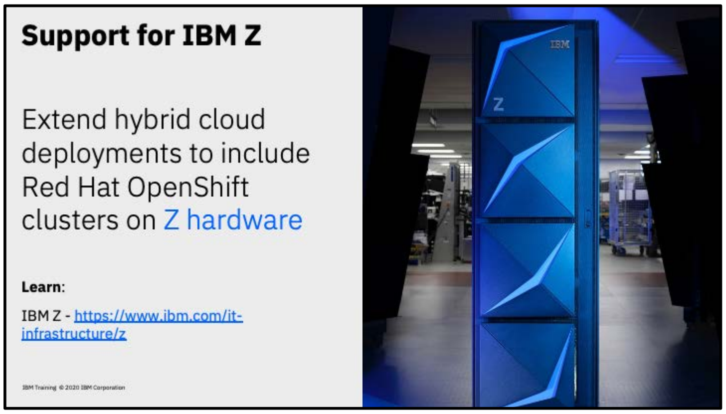
OpenShift Service Mesh -

https://www.openshift.com/learn/topics/service-mesh



Cloud Pak for Applications V4.0 delivers Red Hat OpenShift Container Platform 4.2 as the foundation to hybrid multicloud deployments. Red Hat OpenShift Container Platform 4.2 contains the following updates to help developers include these client tools:

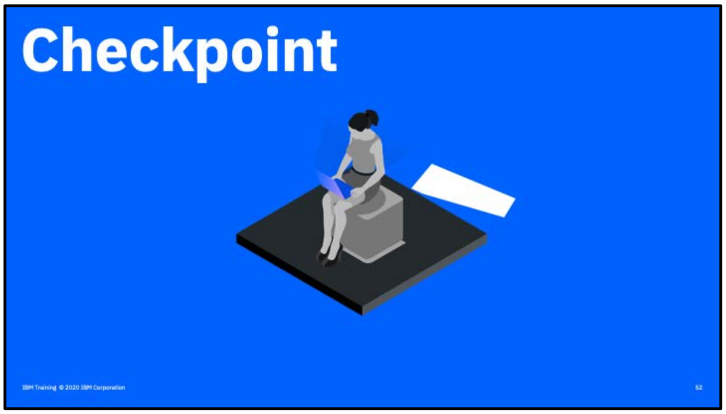
- Web Console with a developer perspective, so developers can focus coding instead of operations
- Red Hat OpenShift Connector for Microsoft™ Visual Studio Code, JetBrains IDE (including IntelliJ), and Eclipse Desktop IDE, making it easier to plug into existing developer pipelines, and
- Red Hat OpenShift Deployment Extension for Microsoft Azure DevOps

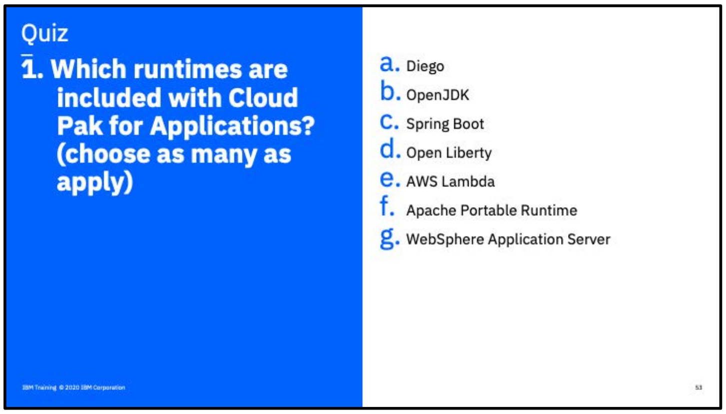


Additionally, Cloud Pak for Applications V4 extends support for Red Hat OpenShift Container Platform 4.2 onto the IBM Z platform. Clients can extend their hybrid cloud deployments to include Red Hat OpenShift clusters on Z hardware, taking advantage of the container orchestration platform and tools to bring a consistent development experience for the development of cloud-native workloads. Support for OpenShift on IBM Z in this release of IBM Cloud Pak for Applications is limited to the container platform only. IBM run times continue to provide support for IBM Z, including container deployments where appropriate.

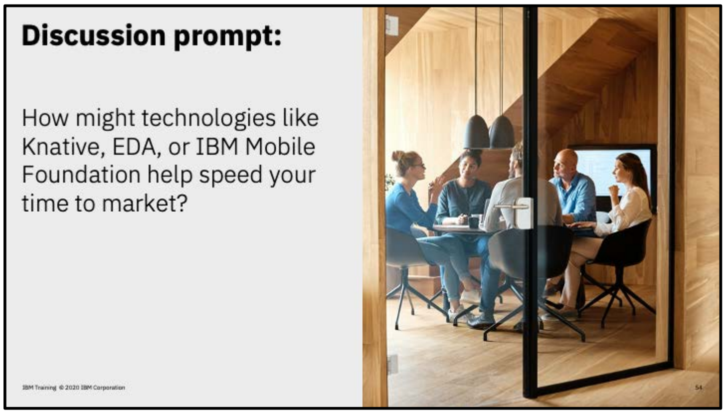
Learn:

IBM Z - https://www.ibm.com/it-infrastructure/z

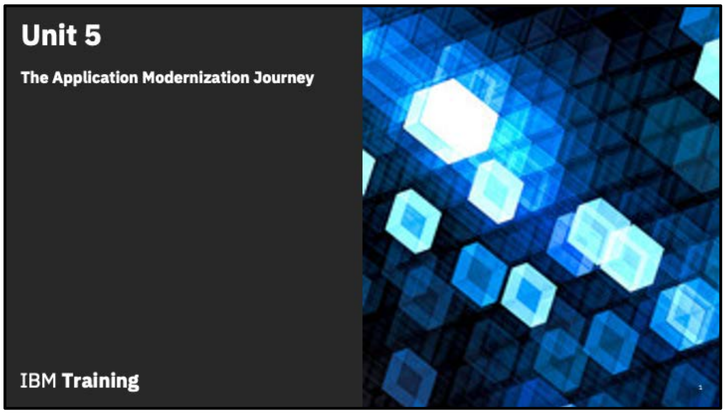

Checkpoint
IBM Training © 2020 IBM Corporation
52



Correct answer: b, c, d, g.



Choose one or more of these topics to discuss.



Unit 5: The Application Modernization Journey

This unit describes various strategies and techniques for modernizing your existing applications by using Cloud Pak for Applications.

Presentation time: about 1 hour

Supplemental material:

Videos: 1 hour and 30 minutes

Readings: 1 hour and 30 minutes

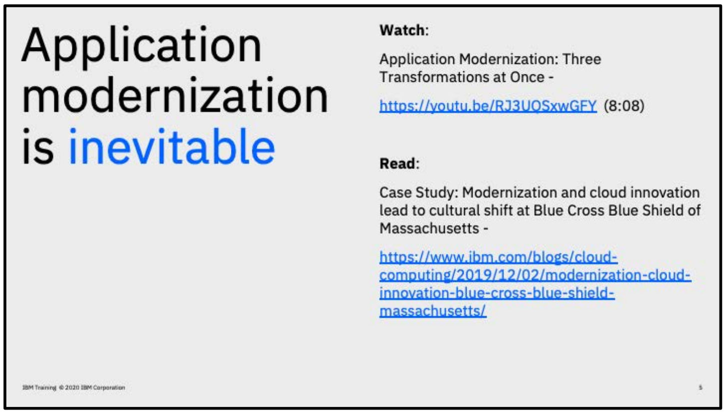
Topics

 2


What is application modernization?
IBM Training © 2020 IBM Corporation
3

Objectives

- Define application modernization
- List the benefits of modernizing applications
- Describe how to assess the application portfolio
- Describe some application modernization strategies



Whether you're moving workloads in stages or working through a complete transformation, you'll have applications that you need to move to the cloud.

Application modernization is a necessary part of cloud-centric business transformation. Business pressures demand faster time to market. IT economics require greater workload mobility to allow migrations to the cloud, improving operational efficiency and reducing cost.

Watch:

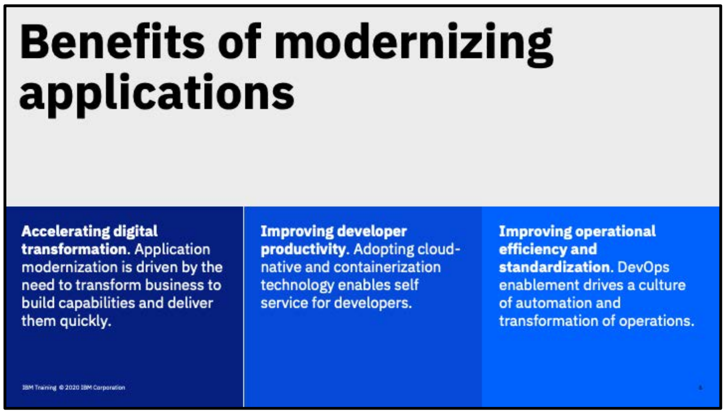
Application Modernization: Three Transformations at Once -

https://youtu.be/RJ3UQSxwGFY (8:08)

Read:

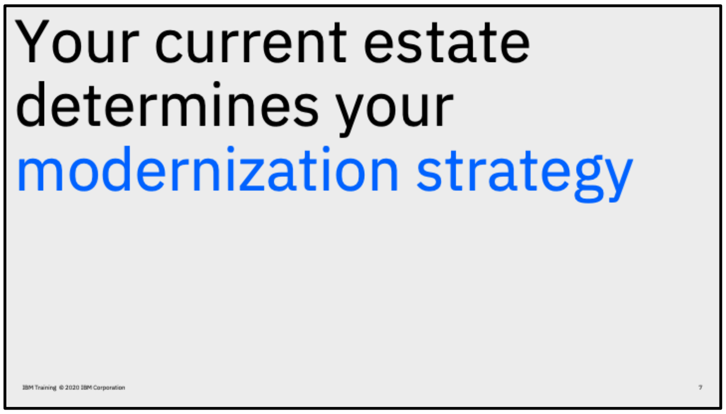
Case Study: Modernization and cloud innovation lead to cultural shift at Blue Cross Blue Shield of Massachusetts -

https://www.ibm.com/blogs/cloud-computing/2019/12/02/modernization-cloud-innovation-blue-cross-blue-shield-massachusetts/



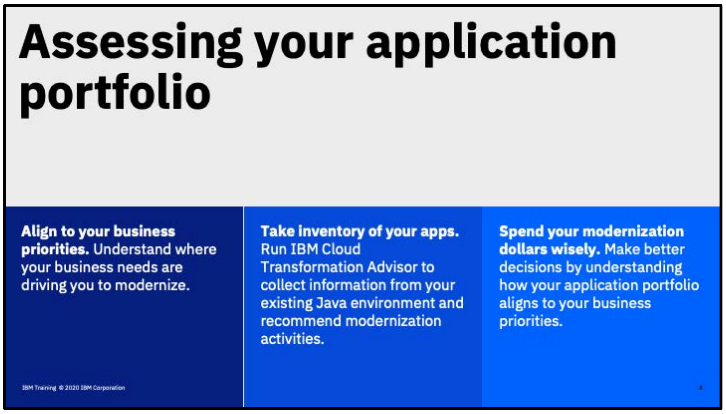
Modernizing your applications provides a few immediate benefits:

- Accelerating digital transformation. Application modernization is driven by the need to transform business to build capabilities and deliver them quickly.
- Improving developer productivity. Adopting cloud-native and containerization technology enables self service for developers.
- Improving operational efficiency and standardization. DevOps enablement drives a culture of automation and transformation of operations.



So, what does *application modernization* really mean? To start your modernization journey, you must understand the approaches that are right for your inventory and your modernization goals and choose the approach that best fits your needs.

The success of a digital transformation depends on the ability of applications to build innovative capabilities and deliver them quickly and to accelerate developer productivity and the adoption of new cloud-native technologies. Containers, Kubernetes, and microservices deliver speed and simplicity and are being adopted rapidly. However, your current estate determines your modernization strategy. Rewriting your entire application estate probably is not feasible, so you must devise a modernization strategy.



Before tackling the job of modernizing your applications, you need to understand your application inventory and how it aligns with business priorities. This will help you determine the best technical path to modernization and evaluate the effort required.

Align to your business priorities. Understand where your business needs are driving you to modernize. Classify your application modernization portfolio by choosing an optimal combination of modernization techniques and effort required to meet your business goals.

Take inventory of your apps. Run IBM Cloud Transformation Advisor to collect information from your existing Java environment and recommend modernization activities.

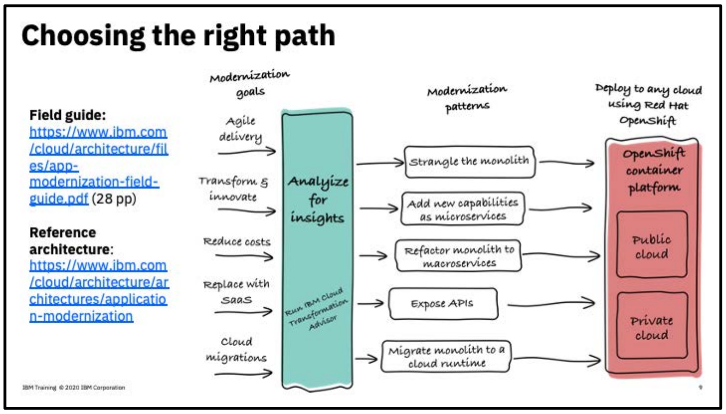
Spend your modernization dollars wisely. Make better decisions by understanding how your application portfolio aligns to your business priorities.

[Optional] Tutorial:

Cloud Pak Experience: Assess cloud readiness

https://www.ibm.com/cloud/paks/experiences/cloud-pak-for-applications/assess

To complete the IBM Cloud Pak Experiences, you must be using at least Firefox version 54 or Chrome version 59.03.



Several factors determine an application's path through the modernization phases and where it eventually finishes. Among those factors is the number of code changes that are required to move an application to a modern, cloud-ready application server, and the business need for future investment in the application. First, analyze your current application estate and prioritize your modernization goals. The outcome of the analysis can help you select appropriate modernization patterns.

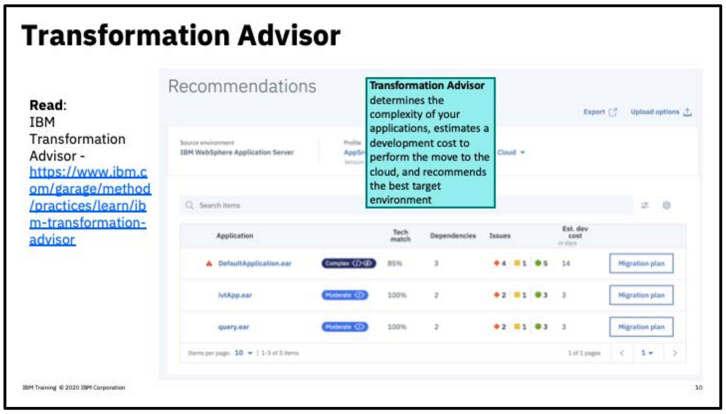
Review the Application modernization field guide and reference architecture to understand which people, tools, and steps you might need on your journey.

Field guide:

https://www.ibm.com/cloud/architecture/files/app-modernization-field-guide.pdf (28 pp)

Reference architecture:

https://www.ibm.com/cloud/architecture/architectures/application-modernization



You can use [IBM Cloud™ Transformation Advisor](https://www.ibm.com/garage/method/practices/learn/ibm-transformation-advisor) to analyze and categorize your applications. It provides guidance on the right target application server and the amount of effort that is required to start the modernization journey. [And, we'll talk more about it later, but first, let's look at some application modernization strategies.]

You can learn more about TA here –

Read:

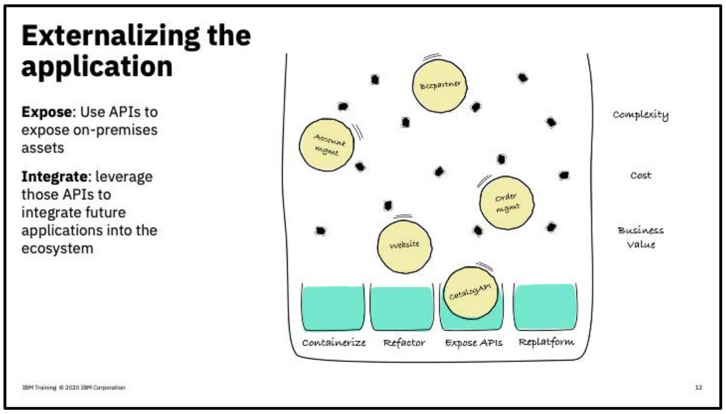
IBM Transformation Advisor **-**

https://www.ibm.com/garage/method/practices/learn/ibm-transformation-advisor

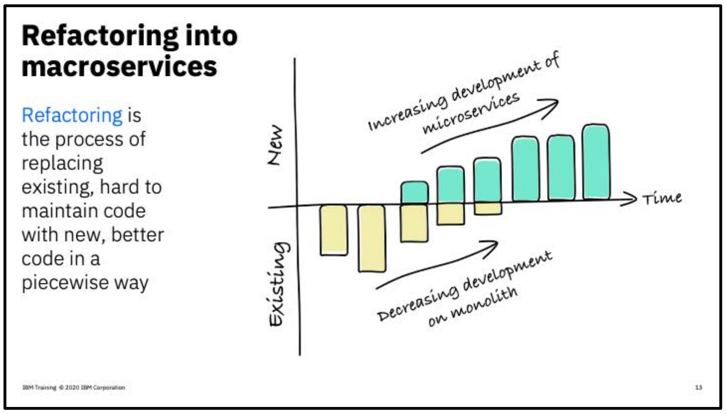
Application migration to the cloud is often approached by using a "lift and shift" approach. However, because of the rapid and broad adoption of container orchestration on Kubernetes platforms across virtually all cloud providers, containerization is usually the better way to move applications to the cloud. At the highest level, *containerize* means to get the applications that you have today to run in a container.

Wrapping an application in a container image is a good first step toward modernization, but many applications are not optimized for containers. Load balancing, application state handling, and monitoring are different in containerized applications. As a result, you might need to rewrite portions of your applications. Likewise, performance tuning and DevOps processes must be aligned to containers.

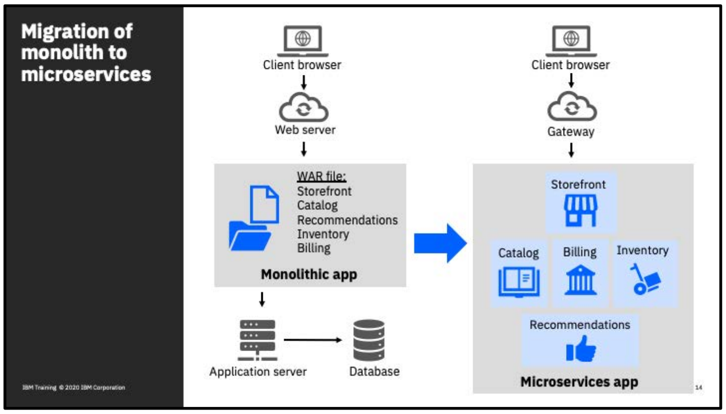
By using the information from Transformation Advisor, you can decide at this phase whether the application can move to a modern, cloud-ready runtime such as IBM®WebSphere® Liberty. Such a move can require code changes and investment from the development team. You can also decide whether the application can move to a container that is based on traditional WebSphere without code changes.



Some existing applications are best exposed as APIs. Typically, this involves creating a REST interface that allows easy access from any endpoint. APIs can then be reused for building new capabilities to augment the existing application. You can leverage those APIs to integrate future applications into the ecosystem, without requiring changes to other applications. IBM Cloud Pak for Integration provides capabilities that you might consider leveraging alongside Cloud Pak for Applications.



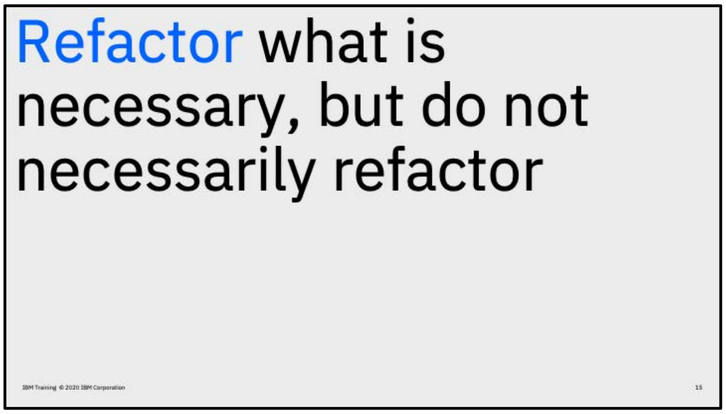
Refactoring is the process of replacing existing, hard to maintain code with new, better code in a piecewise way. Refactoring to a more scalable architecture should be done incrementally. It’s important to provide real business value at each step in the refactoring process.



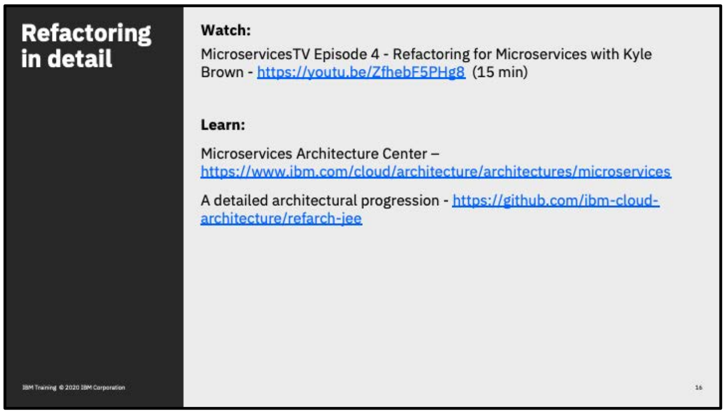
In the monolithic application, the code is on a single server. When you update a single component, you must deploy other components at the same time, so you need a fully equivalent server to satisfy high-availability requirements through a blue-green deployment*.

In the refactored microservices application, deployments of updates are simplified because business services operate independently on separate cloud-based compute infrastructure.

***Blue-green deployment** is a technique that reduces downtime and risk by running two identical production environments called **Blue** and **Green**. At any time, only one of the environments is live, with the live environment serving all production traffic.



But not all applications might need to be refactored. There are other approaches to app modernization, as described previously.

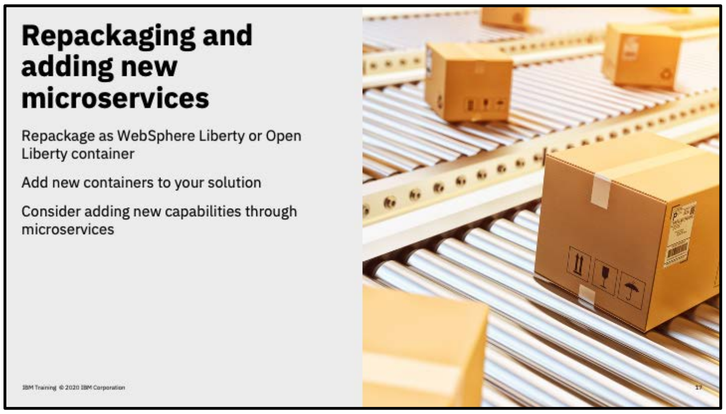


Watch:

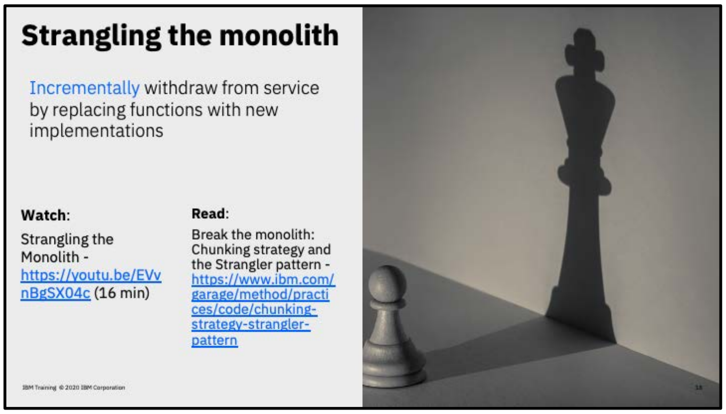
MicroservicesTV Episode 4 - Refactoring for Microservices with Kyle Brown

- https://youtu.be/ZfhebF5PHg8 (15 min)

For modernizing and refactoring a monolithic application, see the microservices architecture on the IBM Cloud Architecture Center, and view a detailed architectural progression here at this URL - https://github.com/ibm-cloud-architecture/refarch-jee



Another approach is to repackage your monolith as a WebSphere Liberty or Open Liberty container. Then add new containers to your solution as you separate business functions into new microservices. Also, Consider adding new capabilities through microservices. Enriching an application with new business function is a great way to realize the value from application modernization for cloud. Your application can consume a wide variety of innovative services that are available in the cloud already, such as AI, industry, and domain services.



Strangling the monolith means to incrementally withdraw it from service by replacing functions with new implementations.

Watch:

MicroservicesTV Episode 16 - Strangling the Monolith

- https://youtu.be/EVvnBgSX04c (16 min)

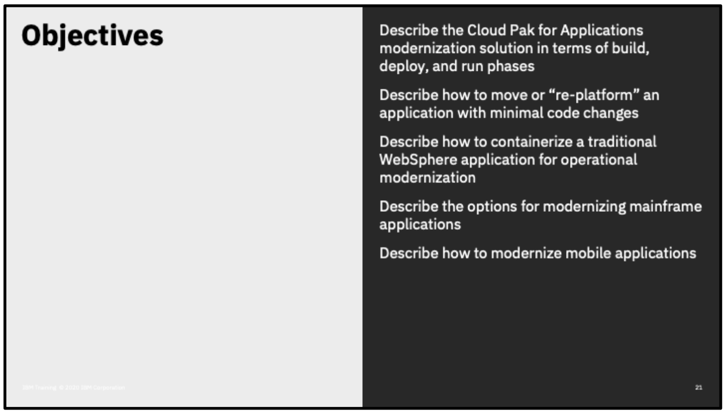
Read:

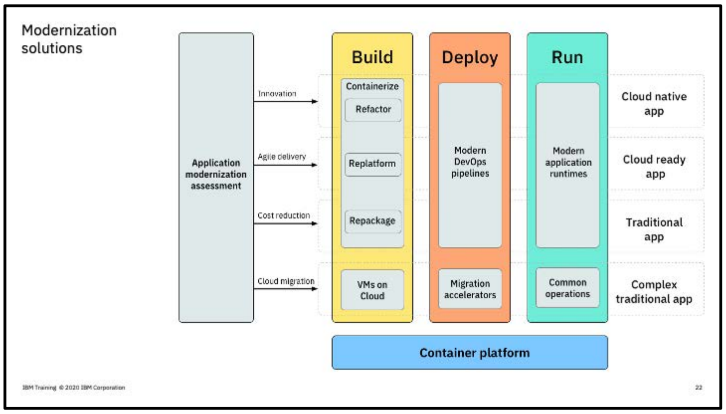
Break the monolith: Chunking strategy and the Strangler pattern

- https://www.ibm.com/garage/method/practices/code/chunking-strategy-strangler-pattern

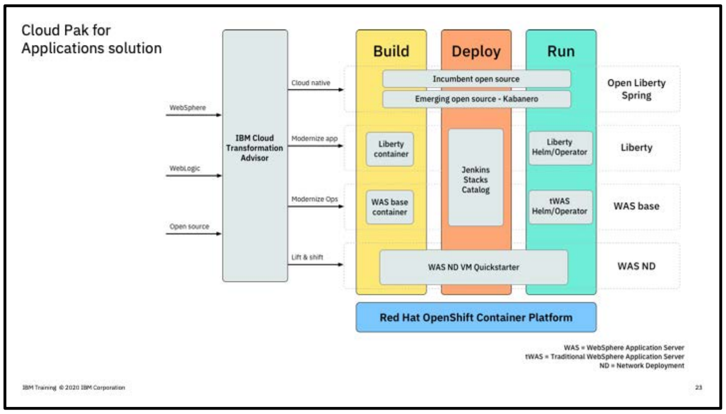

Discussion prompt:
Why is application modernization inevitable?
Which modernization strategy would work best for your organization?


Cloud Pak for
Applications as
modernization
solution
IBM Training © 2020 IBM Corporation


Objectives
Describe the Cloud Pak for Applications modernization solution in terms of build, deploy, and run phases
Describe how to move or “re-platform” an application with minimal code changes
Describe how to containerize a traditional WebSphere application for operational modernization
Describe the options for modernizing mainframe applications
Describe how to modernize mobile applications



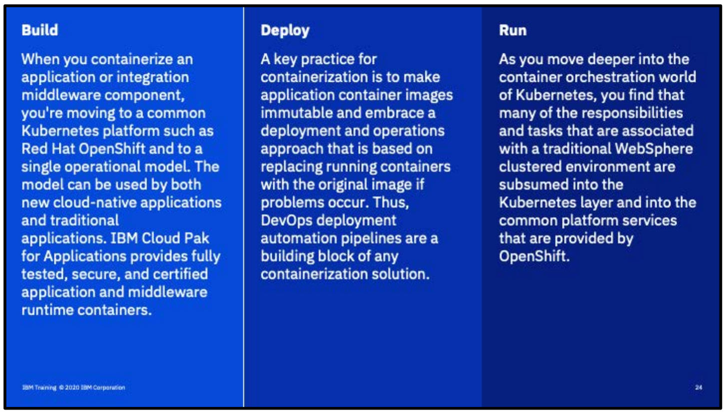
So, you have several options for modernization, as summarized in this diagram. IBM introduced experiences on modernization approaches, such as the incremental lift-and-shift to cloud-native microservices. With these strategies, you can make the most of your investments without rewriting your entire estate. And, with a cloud-native microservices approach, you can capitalize on the scalability and flexibility that is inherent to cloud.



So, where does IBM Cloud Pak for Applications come into play? It provides a full-stack cloud solution. It gives you an open environment where you can quickly build cloud-native applications, modernize or extend your applications, and deploy middleware consistently across clouds.

As shown in the diagram, you can customize your modernization strategy with a flexible deployment model and modernization toolkit. You can also gradually move traditional applications to the cloud in a cost-effective way that makes sense for your business. As you modernize your applications, you can use insights to make decisions about what to re-platform, repackage, and refactor.

You can develop cloud-native applications with containers, common services, developer tools of your choice, and integrated DevOps and use an optimized set of frameworks and runtimes for cloud-native and traditional applications. The solution is an end-to-end application modernization experience that includes Build, Deploy, and Run phases.



The solution is an end-to-end application modernization experience that includes Build, Deploy, and Run phases:

Build.

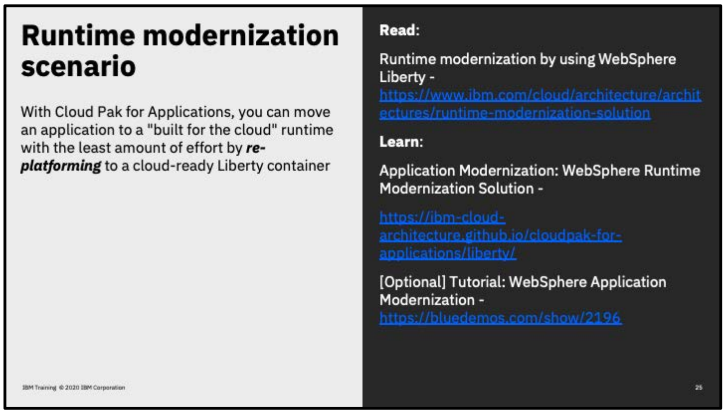
When you containerize an application or integration middleware component, you're moving to a common Kubernetes platform such as Red Hat OpenShift and to a single operational model. The model can be used by both new cloud-native applications and traditional applications. IBM Cloud Pak for Applications provides fully tested, secure, and certified application and middleware runtime containers.

Deploy.

A key practice for containerization is to make application container images immutable and embrace a deployment and operations approach that is based on replacing running containers with the original image if problems occur. Thus, DevOps deployment automation pipelines are a building block of any containerization solution.

Run.

As you move deeper into the container orchestration world of Kubernetes, you find that many of the responsibilities and tasks that are associated with a traditional WebSphere clustered environment are subsumed into the Kubernetes layer and into the common platform services that are provided by OpenShift.



Runtime modernization moves an application to a "built for the cloud" runtime with the least amount of effort.

Application containerization outcomes are greatly improved when containerization is accompanied by re-platforming the application to a modern container-native runtime that is most suitable for cloud environments, such as WebSphere Liberty. Such modern application runtimes allow application containers to be more agile in terms of smaller runtime footprint and faster startup time. They also provide DevOps-oriented server configuration by using simple human-readable files and prebuilt integration points with common operational services such as logging, metrics, metering, health, and security. A few code changes are typically required to move applications from traditional WebSphere to Liberty.

For an end-to-end scenario that moves an application to the WebSphere Liberty runtime and runs it in Red Hat OpenShift, see Runtime modernization by using WebSphere Liberty - https://www.ibm.com/cloud/architecture/architectures/runtime-modernization-solution

IBM® WebSphere®Liberty is a fast and dynamic Java™ application server that is built on the open source Open Liberty project It is a combination of IBM technology and open source software. It has fast startup times (less than 2 seconds), no server restarts to pick up changes, and a simple XML configuration.

However, WebSphere Liberty doesn't support all the traditional Java EE and WebSphere proprietary functions. You might need to change some code to move an application to the new runtime. You also need to move the application configuration from traditional WebSphere to WebSphere Liberty's XML configuration files.

This path gets the application onto a cloud-ready runtime container that is easy to use and portable. However, the application is mostly unchanged and is not modernized to a newer architecture such as microservices.

[You can learn more about this scenario here]

Learn:

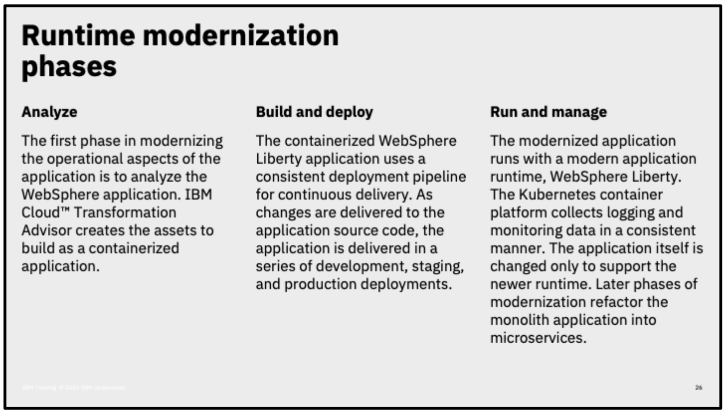
Application Modernization: WebSphere Runtime Modernization Solution -

https://ibm-cloud-architecture.github.io/cloudpak-for-applications/liberty/

[Hands-on demo here]

[Optional] Tutorial:

WebSphere Application Modernization -https://bluedemos.com/show/2196



Analyze

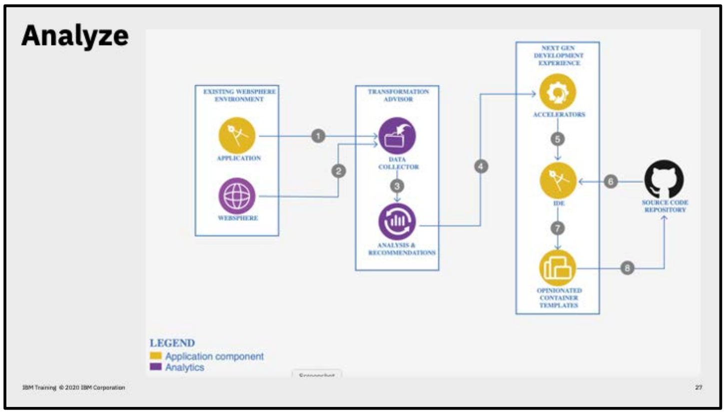
The first phase in modernizing the operational aspects of the application is to analyze the WebSphere application. IBM Cloud™ Transformation Advisor creates the assets to build as a containerized application.

Build and deploy

The containerized WebSphere Liberty application uses a consistent deployment pipeline for continuous delivery. As changes are delivered to the application source code, the application is delivered in a series of development, staging, and production deployments.

Run and manage

The modernized application runs with a modern application runtime, WebSphere Liberty. The Kubernetes container platform collects logging and monitoring data in a consistent manner. The application itself is changed only to support the newer runtime. Later phases of modernization refactor the monolith application into microservices.



Here is a workflow of the Analyze phase.

Step 1

The Data Collector in IBM Transformation Advisor scans the WebSphere Java application. The Data Collector scans the compiled code for known changes in Java, Java EE, and WebSphere APIs that might require code changes to move to newer versions.

Step 2

The Data Collector scans the WebSphere Application Server runtime. The Data Collector collects information about the Java EE resources such as data sources and messaging definitions that need to be moved with the application code to the new runtime.

Step 3

The Analysis and Recommendations engine interprets the results from the Data Collector and presents them.

Step 4

In this scenario, the application requires minimal code changes to move to the WebSphere Liberty runtime. The team decides to modernize the runtime. Transformation Advisor creates a Dockerfile and a WebSphere Liberty server.xml configuration file and identifies the source code changes that are needed.

Step 5

The developer imports the Dockerfile and WebSphere Liberty server.xml configuration file into the IDE.

Step 6

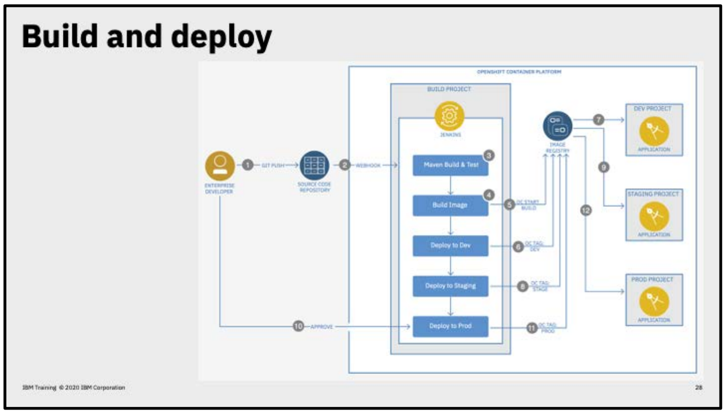
The developer imports the application source code into the IDE and makes the required changes to the source code.

Step 7

The application and the runtime configuration are loaded into an IBM-provided opinionated container template for local testing.

Step 8

When local testing is complete, the updated application source code and the new runtime configuration files are loaded into git and are ready for deployment to Red Hat OpenShift.



Here are the steps in the Build and Deploy phase.

Step 1

The developer commits the application source code and deployment artifacts into git.

Step 2

A webhook automatically triggers the Jenkins pipeline in the Red Hat OpenShift build project.

Step 3

The pipeline checks out the Jenkins pipeline defintion file, application source code, and deployment artifacts from git and uses Maven to build and test the application.

Step 4

The 'oc start build' command is used to build an immutable image for the application. The image contains the configured application server and the newly compiled and tested application EAR file.

Step 5

The immutable image is added to the ImageStream of the build project and pushed to the image registry in the Red Hat OpenShift cluster.

Step 6

The immutable image is tagged for the dev project.

Step 7

The deployment that is running in the dev project is restarted by using the new immutable image.

Step 8

The immutable image is tagged for the stage project.

Step 9

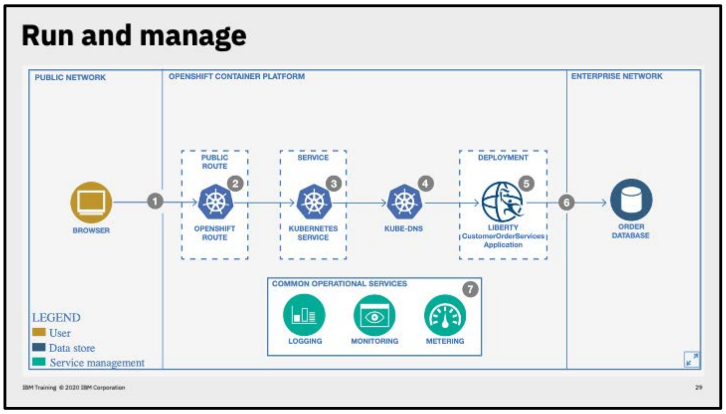
The deployment that is running in the stage project is restarted by using the new immutable image.

Step 10

The developer is prompted to approve the deployment to production.

Step 11

The immutable image is tagged for the prod project.



Here are the steps in the Run and Manage phase.

Step 1

The Customer Order Services application is a web-based application that is accessed by using a browser.

Step 2

A Red Hat OpenShift route is the public entry point into the container platform.

Step 3

The route passes the request from the browser to a Kubernetes service. The service is an abstraction that defines a logical set of Kubernetes Pods and a policy by which to access them.

Step 4

The service uses the Kubernetes DNS to find the application's deployment (a set of Kubernetes Pods).

Step 5

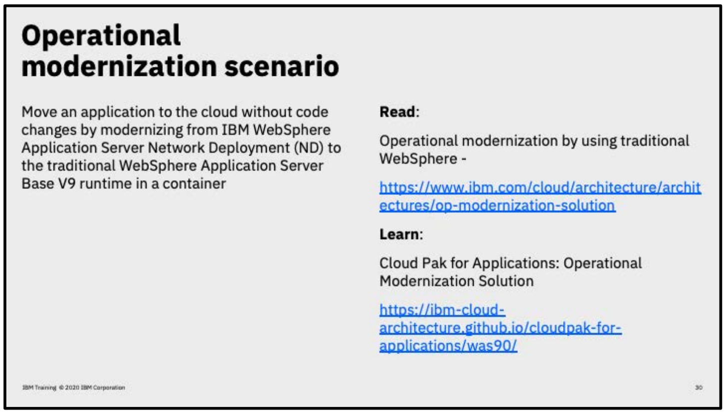
The immutable WebSphere Liberty Docker image for the Customer Order Services application is used to create a container in a Kubernetes Deployment. The Customer Order Services application processes the request from the browser.

Step 6

The Customer Order Services application uses a Db2 database that remains in the enterprise network.

Step 7

By running in the Red Hat OpenShift Container Platform, the application benefits from the common operational services, such as unified logging, monitoring, and usage metering.



Containerizing a traditional WebSphere application for operational modernization

Refactoring enterprise Java applications to WebSphere Liberty and Open Liberty containers is your most strategic option. However, if there is little business need for future investment in the WebSphere application and Transformation Advisor suggests that you need to change the code to move the application to WebSphere Liberty, you can use the container for traditional IBM WebSphere Application Server.

By repackaging traditional WebSphere or WebSphere Network Deployment (ND) applications into a container, you can modernize traditional WebSphere ND operations and replace manual administration of WebSphere ND cells with automated DevOps pipelines that automate applications deployment and configuration. You can reuse WebSphere automation scripting. If no such automation exists, you can use the configuration migration tools that are provided by IBM Cloud Pak for Applications to capture WebSphere environment configurations. Those tools create a separate DevOps pipeline for each application in the cell that is moved to a traditional WebSphere container.

For an end-to-end scenario that moves an application to the traditional WebSphere container and runs it in Red Hat OpenShift, see [Operational modernization by using traditional WebSphere](https://www.ibm.com/cloud/architecture/architectures/op-modernization-solution) - https://www.ibm.com/cloud/architecture/architectures/op-modernization-solution

Operational modernization gives an operations team the opportunity to embrace the practices for modern operations without putting change requirements on the development team. By modernizing from IBM® WebSphere® Application Server Network Deployment (ND) to the traditional WebSphere Application Server Base V9 runtime in a container, you can move an application to the cloud without code changes.

The container runtime provides the scaling, routing, clustering, high availability, and continuous availability that was previously required by WebSphere ND. As a result, your operations team can run cloud-native and older applications in the same environment with the same standardized logging, monitoring, and security frameworks.

While traditional WebSphere isn't a "built for the cloud" runtime like WebSphere Liberty, it can still run in containers and reap the benefits of consistency and reliability that containers provide. The use of containers also helps to improve DevOps and speed to market.

This type of modernization doesn't require code changes and can be driven by the operations team. This path gets the application into a container with the least amount of effort but doesn't modernize the application or the runtime.

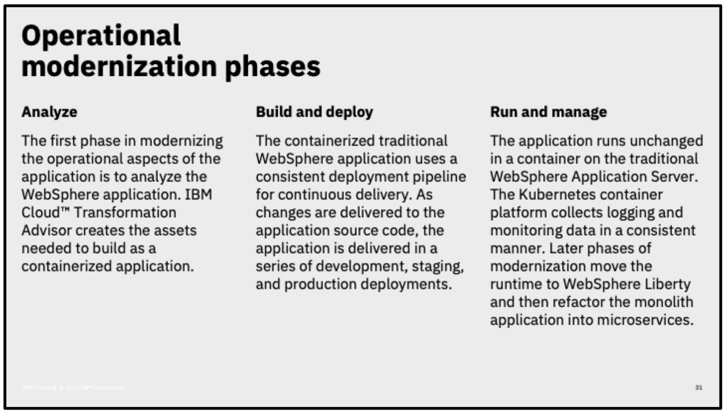
Read:

Operational modernization by using traditional WebSphere -

https://www.ibm.com/cloud/architecture/architectures/op-modernization-solution

Learn:

Cloud Pak for Applications: Operational Modernization Solution https://ibm-cloud-architecture.github.io/cloudpak-for-applications/was90/



Analyze

The first phase in modernizing the operational aspects of the application is to analyze the WebSphere application. IBM Cloud™ Transformation Advisor creates the assets needed to build as a containerized application.

Build and deploy

The containerized traditional WebSphere application uses a consistent deployment pipeline for continuous delivery. As changes are delivered to the application source code, the application is delivered in a series of development, staging, and production deployments.

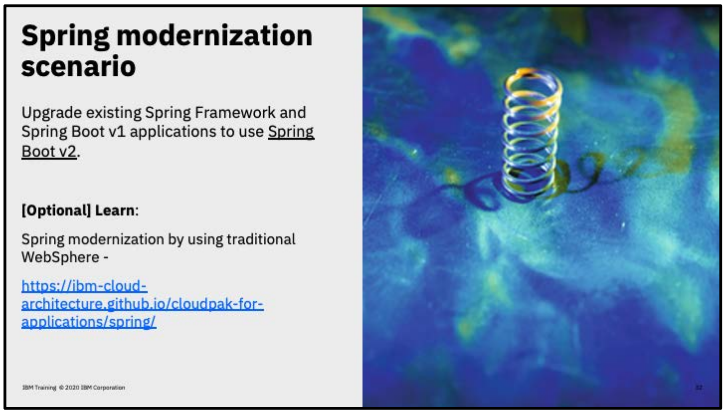
Run and manage

The application runs unchanged in a container on the traditional WebSphere Application Server. The Kubernetes container platform collects logging and monitoring data in a consistent manner. Later phases of modernization move the runtime to WebSphere Liberty and then refactor the monolith application into microservices.

The steps in the architecture diagram for the operational scenario are mostly the same as for the runtime scenario, except that in the analyze phase:

The team decides to use the traditional WebSphere runtime instead. This path gets the application in to a container with the least amount of effort but doesn't modernize the application or the runtime. IBM Cloud Transformation Advisor creates a Dockerfile for the application as an accelerator.

The Dockerfile configures an opinionated container template that is augmented with application-specific configuration files and the application EAR file for local testing.

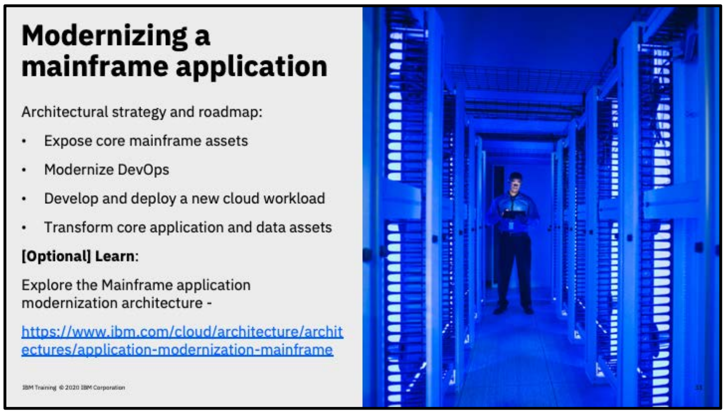


Spring modernization describes the process of upgrading existing Spring Framework and Spring Boot v1 applications to use Spring Boot v2. With the Open Liberty project, IBM provides support for Spring Boot and optimized the runtime and Docker images specifically for Spring Boot.

[Optional] Learn:

Spring modernization by using traditional WebSphere -

https://ibm-cloud-architecture.github.io/cloudpak-for-applications/spring/

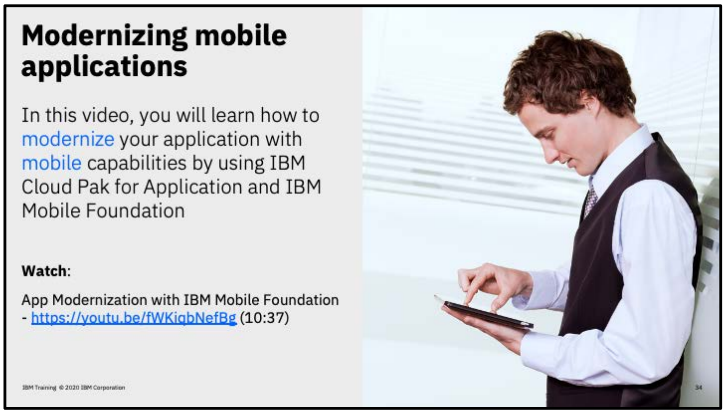


Organizations rely on core applications that maintain system-of-record (SOR) data on the mainframe can also modernize. You can develop hybrid-cloud solutions, build cloud-native applications, modernize to a DevOps approach, and incrementally transform your core mainframe assets.

[Optional] Learn:

Explore the Mainframe application modernization architecture -

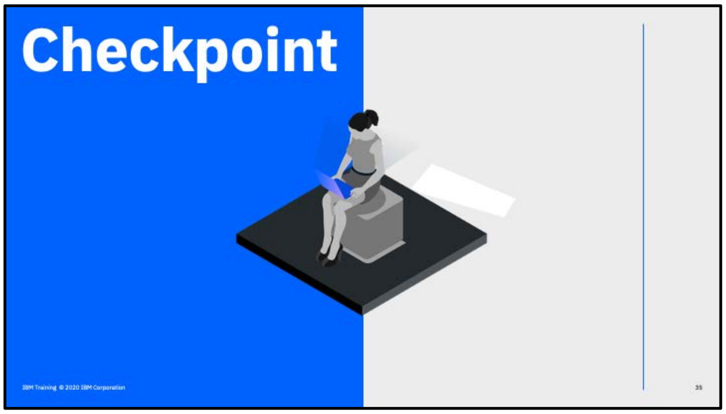
https://www.ibm.com/cloud/architecture/architectures/application-modernization-mainframe

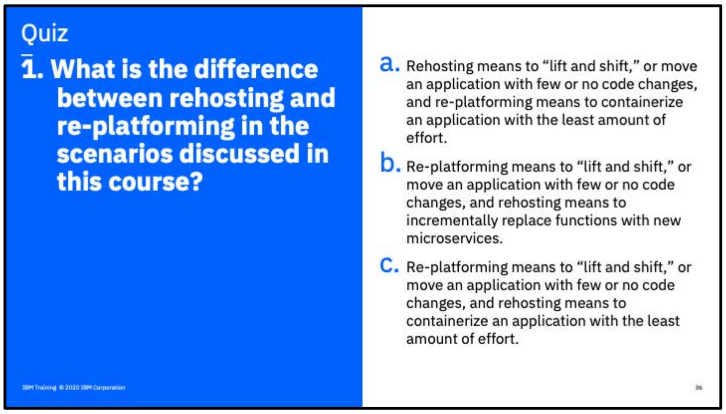


In this video you will learn how to modernize your application with mobile capabilities by using IBM Cloud Pak for Application and IBM Mobile Foundation.

Watch:

App Modernization with IBM Mobile Foundation - https://youtu.be/fWKiqbNefBg (10:37)


Checkpoint
IBM Training © 2020 IBM Corporation
35



Correct answer: a.


Discussion prompt:
What value does Cloud Pak for Applications provide in each phase of the application modernization journey: build, deploy, and run?
What is the value of migrating existing applications to run in Cloud Pak for Applications?
IBM Training © 2020 IBM Corporation

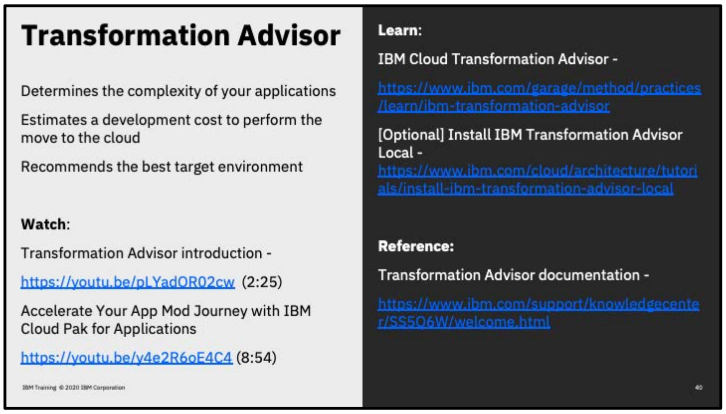

Application
modernization
tools
IBM Training © 2020 IBM Corporation

Objectives

Describe how to use Transformation Advisor to assess application cloud readiness

Describe how to use Application Navigator to visualize, inspect, and interact with deployed resources in your environment

Describe how to use WebSphere Application Migration Toolkit to evaluate and move applications to Liberty



The process of modernizing applications and moving to the cloud can be a large undertaking. Careful planning is needed to prepare business inventory and infrastructure and to determine the best path forward for each application. If you don't yet have a cloud platform but are ready to begin the assessment for your modernization journey, installing IBM Cloud™ Transformation Advisor locally can help you get started. Transformation Advisor is also included as a web application with IBM Cloud Pak for Applications.

Transformation Advisor can help you with your transformation journey in these areas:

Assessment: Applications evaluated based on business needs, High-level inventory of application content and structure, Potential issues and migration complexity assessments

Strategic Planning: Identification of key applications that require replacement or upgrade, Estimated effort for resolving migration issues, Recommendations for a modernization path

Execution: Migration bundle automatically generated to assist with containerization and deployment, Files that are included are tailored to your applications, Dependencies identified to help you to complete the bundle

TA Provides:
Analysis Report - Potential issues, severity, possible solutions and estimate of resolution effort
Technology Report - Details on IBM platform support for technologies used in the app
Inventory Report - High-level inventory of application content and structure
Customized application configuration and deployment files

Watch:
Transformation Advisor introduction – https://youtu.be/pLYadOR02cw (2:25)

Accelerate Your App Mod Journey with IBM Cloud Pak for Applications - https://youtu.be/y4e2R6oE4C4 (8:54)

Learn:

IBM Cloud Transformation Advisor -

https://www.ibm.com/garage/method/practices/learn/ibm-transformation-advisor

[Optional] Tutorial:

Install IBM Transformation Advisor Local (1 hour) - https://www.ibm.com/cloud/architecture/tutorials/install-ibm-transformation-advisor-local

Reference:

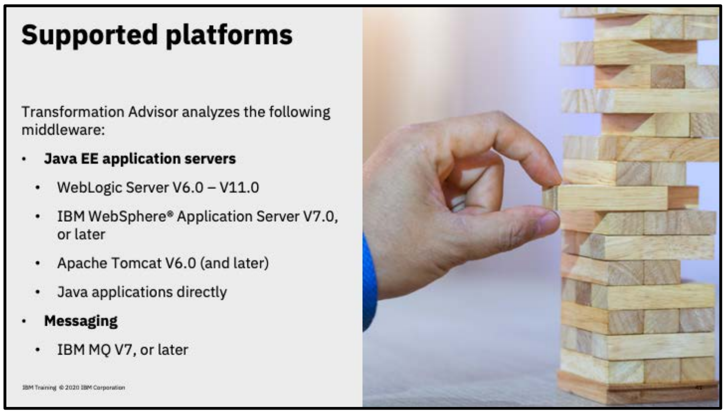
https://www.ibm.com/support/knowledgecenter/SS5Q6W/welcome.html

[As you review these resources, answer these questions:

What are the capabilities and limitations of Transformation Advisor?

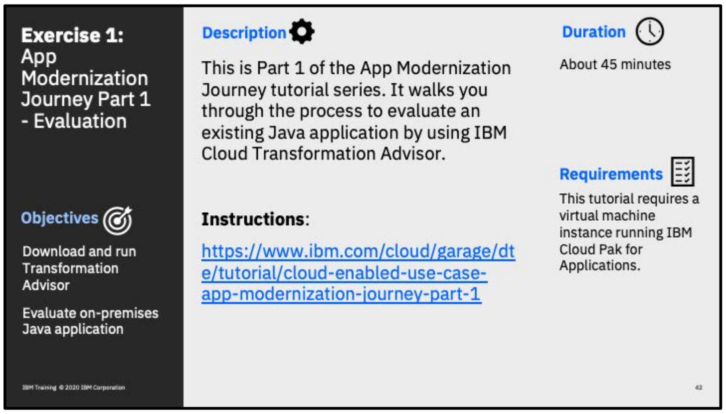
How can you use it to assess applications for cloud readiness?

How can you deploy it within the Cloud Pak for Applications framework?]



Transformation Advisor analyzes the following middleware:

- **Java EE application servers**
  - WebLogic Server V6.0 – V11.0
  - IBM WebSphere® Application Server V7.0, or later
  - Apache Tomcat V6.0 (and later)
  - Java applications directly
- **Messaging**
  - IBM MQ V7, or later



App Modernization Journey Part 1 – Evaluation

This is Part 1 of the App Modernization Journey tutorial series. It will walk you through the process to evaluate an existing Java application by using IBM Cloud Transformation Advisor. (45 minutes)

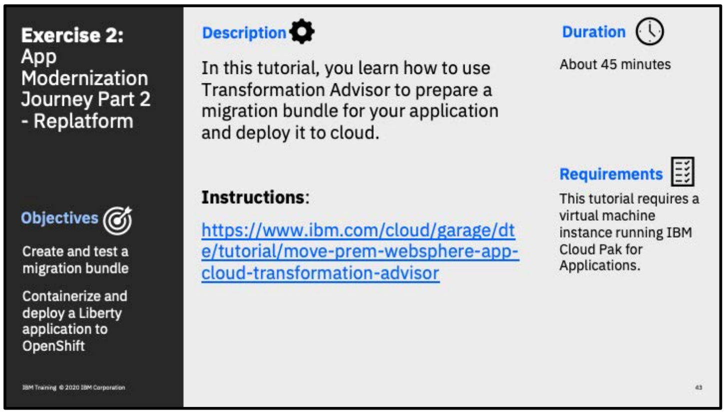
Instructions:

https://www.ibm.com/cloud/garage/dte/tutorial/cloud-enabled-use-case-app-modernization-journey-part-1

This tutorial requires a virtual machine instance running IBM Cloud Pak for Applications.

For classroom and instructor-led online deliveries, the training provider can provide instructions on how to access the remote lab environment for this course.

For self-paced virtual classroom deliveries, instructions for how to access your lab environment are provided in the "Start here" or "Welcome" section of the course.



Cloud Enabled Use Case: App Modernization Journey Part 2 - Replatform
In this tutorial, you learn how to use Transformation Advisor to prepare a migration bundle for your application and deploy it to cloud. (45 minutes)

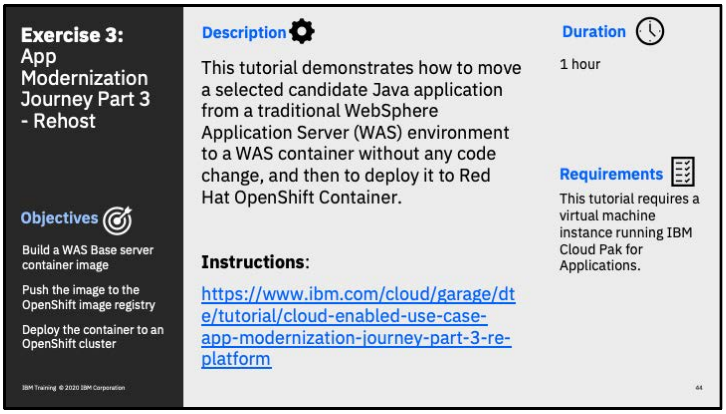
Instructions:

https://www.ibm.com/cloud/garage/dte/tutorial/move-prem-websphere-app-cloud-transformation-advisor

This tutorial requires a virtual machine instance running IBM Cloud Pak for Applications.

For classroom and instructor-led online deliveries, the training provider can provide instructions on how to access the remote lab environment for this course.

For self-paced virtual classroom deliveries, instructions for how to access your lab environment are provided in the "Start here" or "Welcome" section of the course.



This tutorial demonstrates how to move a selected candidate Java application from a traditional WebSphere Application Server (WAS) by environment to a WAS container without any code change, and then to deploy it to Red Hat OpenShift Container.

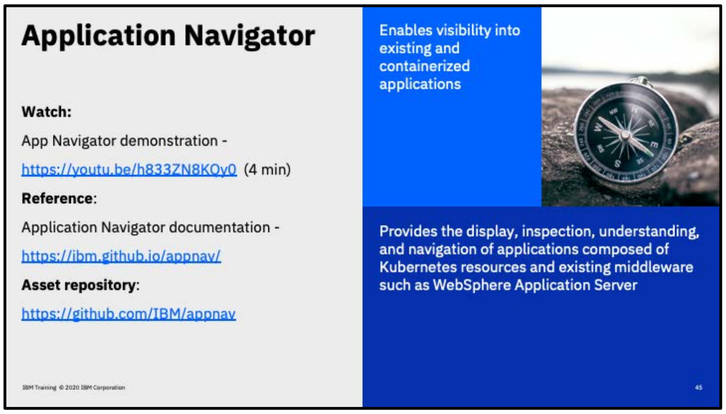
This tutorial showcases the re-hosting process. Re-host uses Lift and Shift approach to move an existing application to the same server environment in the cloud. You learn how to move a selected candidate Java application from a traditional **WebSphere Application Server** (**tWAS**) environment to a **tWAS container** without any code change, and to deploy it to a **RHOCP** environment.

Instructions:

https://www.ibm.com/cloud/garage/dte/tutorial/cloud-enabled-use-case-app-modernization-journey-part-3-re-platform

For classroom and instructor-led online deliveries, the training provider can provide instructions on how to access the remote lab environment for this course.

For self-paced virtual classroom deliveries, instructions for how to access your lab environment are provided in the “Start here” or “Welcome” section of the course.



IBM Application Navigator is a tool that extends the Kubernetes® console to provide the display, inspection, understanding, and navigation of applications composed of Kubernetes resources and existing middleware, such as WebSphere Application Server. Application Navigator extends the open source Kubernetes Application Navigator (kAppNav) by providing integration to WebSphere Application Server Network Deployment cells and Liberty collectives, which enables visibility to existing and containerized applications.

Watch:

App Navigator demonstration -

https://youtu.be/h833ZN8KQy0 (4 min)

[https://mediacenter.ibm.com/media/Application+Navigator+Introduction/0_zifamkjp#]

Reference:

Application Navigator documentation -

https://ibm.github.io/appnav/

Asset repository:

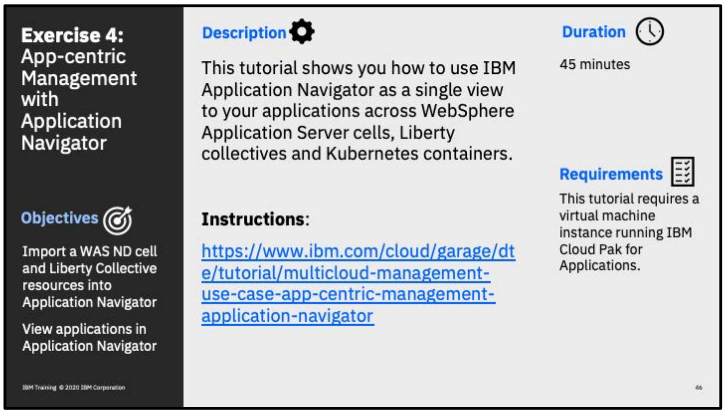
https://github.com/IBM/appnav

[As you watch the video, and review the documentation, answer these questions:

What are the use cases of Application Navigator in the context of application modernization?

How does Application Navigator fit into the OpenShift Container Platform?

What are the integration points between the Application Navigator and Cloud Pak for Applications consoles?]



App Centric Management with Navigator

This hands-on tutorial shows you how to use IBM Application Navigator as a single view to see your applications across WebSphere Application Server cells, Liberty collectives and Kubernetes containers.

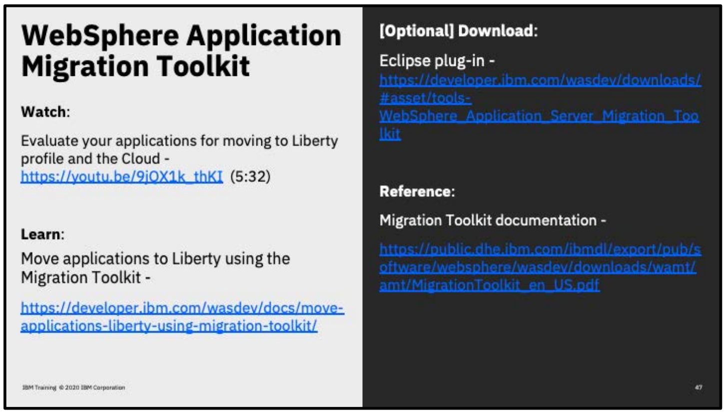
Instructions:

https://www.ibm.com/cloud/garage/dte/tutorial/multicloud-management-use-case-app-centric-management-application-navigator

This tutorial requires a virtual machine instance running IBM Cloud Pakfor Applications.

For classroom and instructor-led online deliveries, the training provider can provide instructions on how to access the remote lab environment for this course.

For self-paced virtual classroom deliveries, instructions for how to access your lab environment are provided in the "Start here" or "Welcome" section of the course.



[As you review these resources, answer this question:

How does the Migration Toolkit fit into the application modernization journey?]

Watch:

Evaluate your applications for moving to Liberty profile and the Cloud -

https://youtu.be/9jQX1k_thKI (5:32)

Learn:

Move applications to Liberty by using the Migration Toolkit -

https://developer.ibm.com/wasdev/docs/move-applications-liberty-using-migration-toolkit/

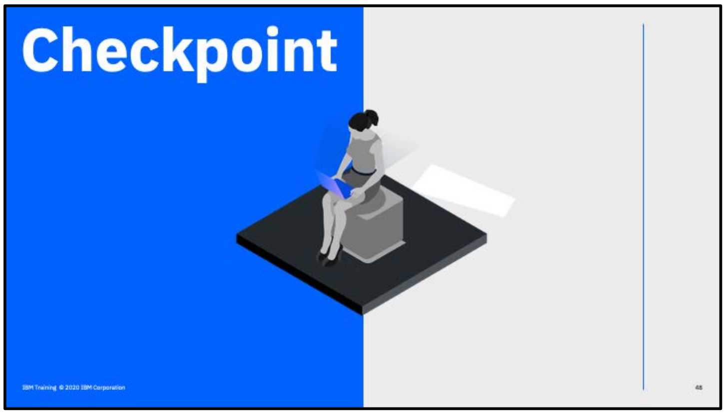
If you use Eclipse, use the WebSphere® Application Migration Toolkit Eclipse plug-in to help accelerate the process of modifying the application code. You can download it from here -

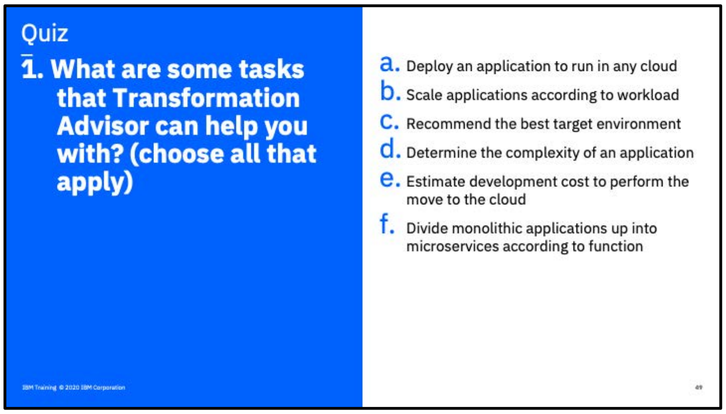
[Optional] Download:

https://developer.ibm.com/wasdev/downloads/#asset/tools-WebSphere_Application_Server_Migration_Toolkit

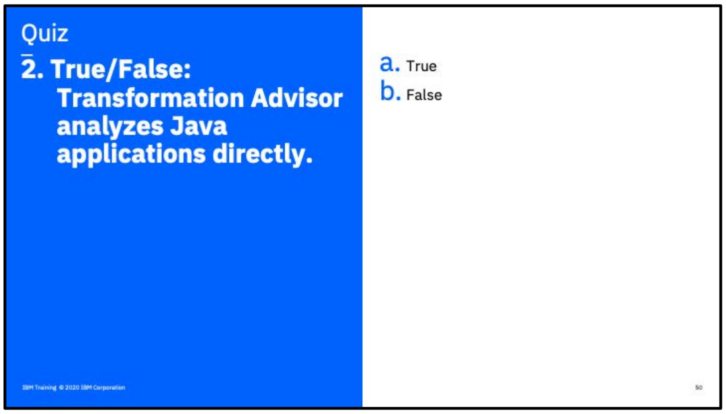
Reference:

https://public.dhe.ibm.com/ibmdl/export/pub/software/websphere/wasdev/downloads/wamt/amt/MigrationToolkit_en_US.pdf

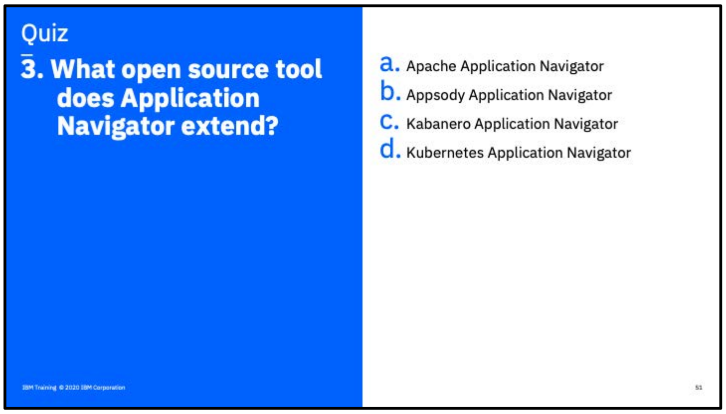

Checkpoint
IBM Training © 2020 IBM Corporation
48



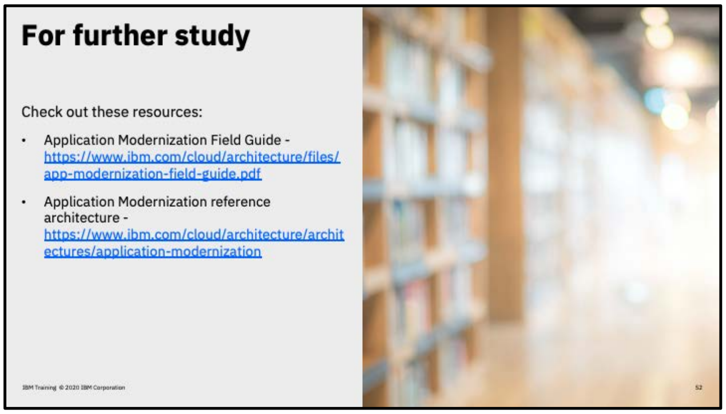
Correct answers: c, d, e

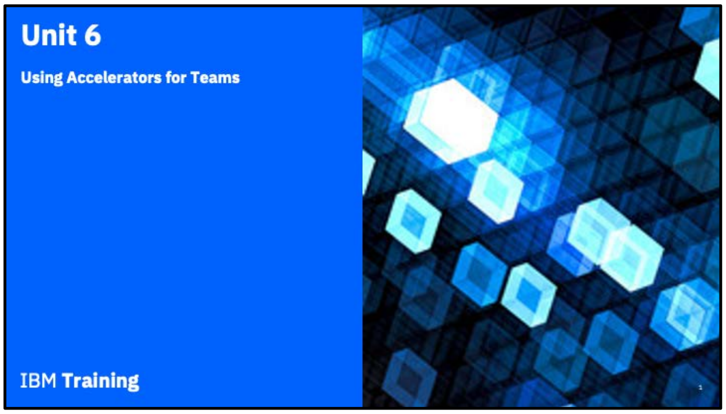


Correct answer: a.



Correct answer: d.


For further study
Check out these resources:
• Application Modernization Field Guide - https://www.ibm.com/cloud/architecture/files/app-modernization-field-guide.pdf
• Application Modernization reference architecture - https://www.ibm.com/cloud/architecture/architectures/application-modernization



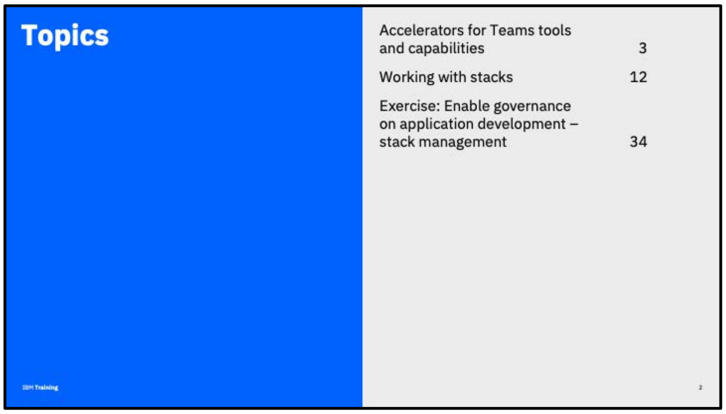
Unit 6 Using Accelerators for Teams

This unit describes the Accelerators for Teams component in Cloud Pak for Applications in more detail and explains how solution architects can use it to enable governance over application development.

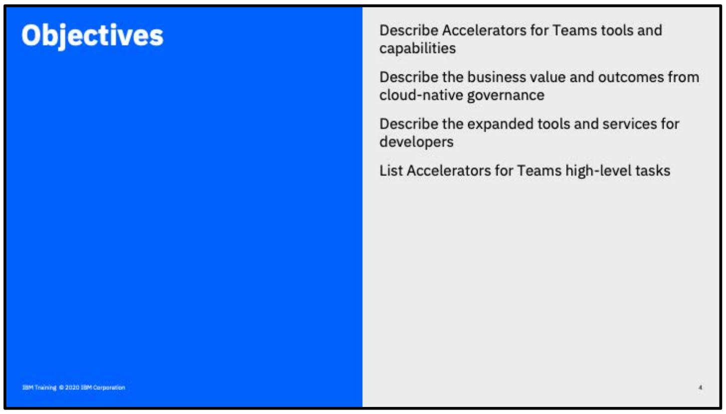
Presentation time: about 40 minutes

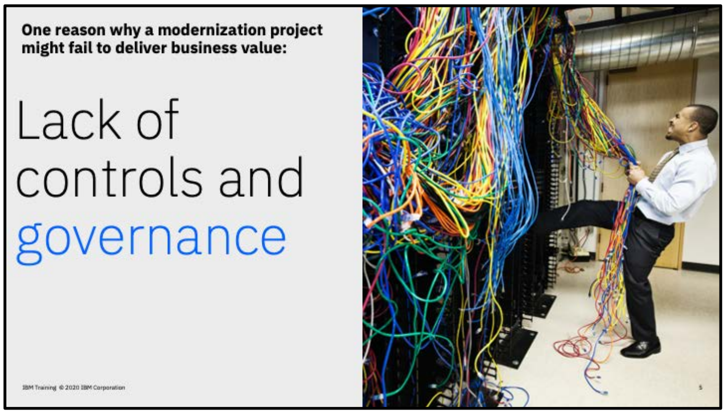
Supplemental material:

Readings: 1 hour and 10 minutes


Topics
Accelerators for Teams tools and capabilities 3
Working with stacks 12
Exercise: Enable governance on application development – stack management 34
IBM Training
2


Accelerators
for Teams
tools and
capabilities
IBM Training © 2020 IBM Corporation


Objectives
Describe Accelerators for Teams tools and capabilities
Describe the business value and outcomes from cloud-native governance
Describe the expanded tools and services for developers
List Accelerators for Teams high-level tasks



When siloed development and project teams work outside the oversight of a governed environment, duplication of effort and resources drive up costs. The transfer and use of massive amounts of data can introduce security and compliance threats. Faulty data used to train artificial intelligence (AI) models can put projects, reputations and businesses at risk.

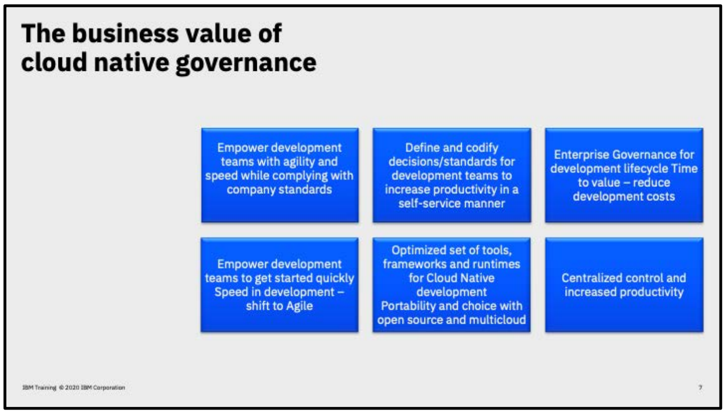
Reference:

https://www.ibm.com/blogs/cloud-computing/2019/01/30/modernization-projects-business-value/

Instead of thinking about environments as two distinct worlds in your transformation journey — "the old world" and the shiny "new world"— to be managed separately, it's critical to manage existing assets and new assets as one environment to propel business forward.

Reference:

https://www.ibm.com/blogs/cloud-computing/2019/01/30/modernization-projects-business-value/



The business value of cloud native governance with Accelerators for Teams can be characterized as such:

- Empower development teams with agility and speed while complying with company standards
  Define and codify decisions/standards for development teams to increase productivity in a self-service manner
- Enterprise Governance for development lifecycle Time to value – reduce development costs
- Empower development teams to get started quickly Speed in development – shift to Agile
- Optimized set of tools, frameworks and runtimes for Cloud Native development
  Portability and choice with open source and multicloud
- Centralized control and increased productivity

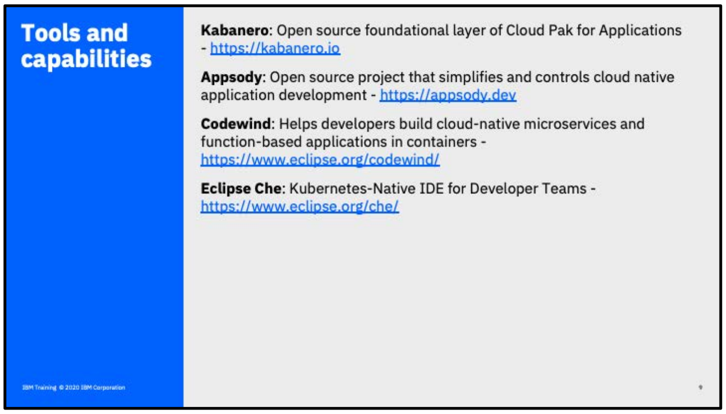
The challenge of integrating new tools and frameworks into a platform of some kind can be a distraction from your primary task of solving business problems. While Kubernetes has become a popular platform for developers, it's not a complete development platform by itself.

Accelerators for Teams supports the activity of application developers, architects, and operations teams in developing, managing, and deploying cloud native applications that meet the requirements of an organization. It brings together the tools and services that are needed to enable CI/CD workflows across the development cycle.

Reference:

https://developer.ibm.com/blogs/cloud-native-development-grows-up/

https://developer.ibm.com/technologies/containers/blogs/app-modernization-ibm-cloud-pak-applications/



These tools include:

Kabanero the open source foundational layer of Cloud Pak for Applications. Kabanero itself is made up of accepted, best-in-class cloud technologies which are all open source. Kabanero uses OKD (the upstream project for Red Hat OpenShift) to give you a robust, proven foundation on Kubernetes. One of the special ingredients in Kabanero is Appsody which uses technology stacks and templates to create a disciplined and consistent approach to developing apps within an enterprise organization.

Appsody is an open source project that simplifies and controls cloud-native application development. Appsody's primary component is a stack, which builds a pre-configured Docker image that developers can immediately use to create applications in a cloud environment. Appsody allows stack builders to decide which parts of the users' resulting application images are fixed (a set of technology choices and configurations defined by the stack image) and which parts stack users can modify or extend (as templates).
The Eclipse Codewind project helps developers build cloud-native microservices and function-based applications in containers. It provides extensions that understand Appsody stacks, so you can start building using the pre-defined stacks and benefit from Codewind's automated container build right from within the IDE. A performance dashboard in Eclipse Codewind helps you immediately begin to understand the performance characteristics of the new app or service early in the development cycle.

Eclipse Che: makes Kubernetes development accessible for developer teams, providing one-click developer workspaces and eliminating local environment configuration for your entire team. Che brings your Kubernetes application into your development environment and provides an in-browser IDE, allowing you to code, build, test and run applications exactly as they run on production from any machine.

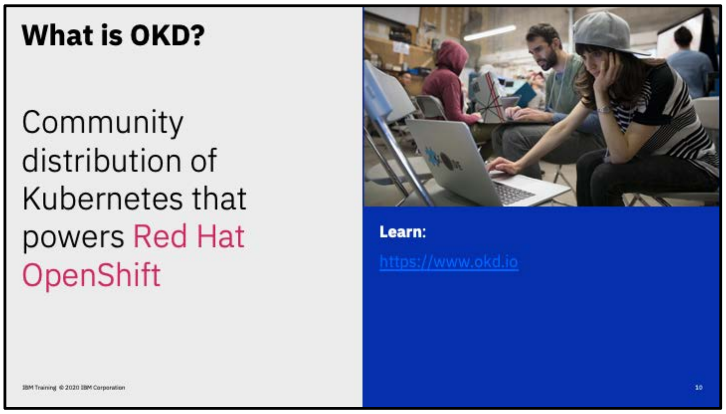
Reference:

https://kabanero.io

https://appsody.dev

https://www.eclipse.org/codewind/

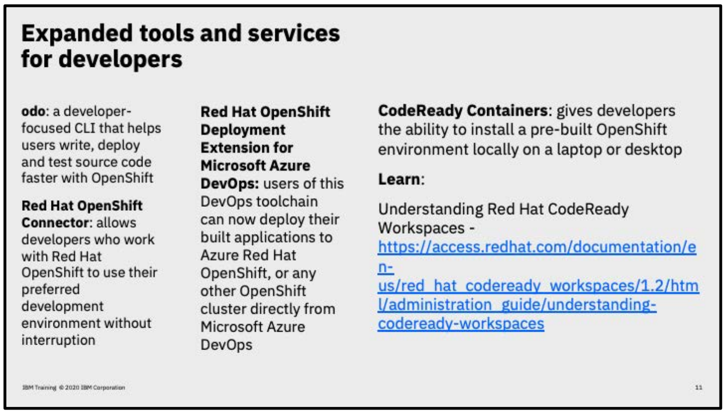
https://www.eclipse.org/che/



OKD is the upstream community project for Red Hat OpenShift. Built around a core of OCI container packaging and Kubernetes container cluster management, OKD is also augmented by application lifecycle management functionality and DevOps tooling.

Reference:

https://www.okd.io



There are several other tools and services available to developers, to help integrate the developer experience with Accelerators for teams, such as:

Red Hat CodeReady Containers

Red Hat CodeReady Containers gives developers the ability to install a pre-built OpenShift environment locally on a laptop or desktop. CodeReady Containers enables local development for OpenShift and helps developers get started with OpenShift quickly and easily.

odo: A Developer-focused Command Line Interface

odo is a developer-focused CLI that helps users write, deploy and test source code faster with OpenShift. Using a few CLI commands and a “git push” style interaction, developers can turn their source code into a running container on OpenShift.

In addition to working with source code changes, odo allows developers to manage other aspects of their deployed source code, such as creating a url for the application, linking a deployed application component to other application components deployed on OpenShift, viewing logs of deployed applications and more. odo helps developers focus on the source code they are writing for applications, rather than all the details of deploying that application component on Kubernetes.

Red Hat OpenShift Connector for Microsoft Visual Studio Code, JetBrains IDE (including IntelliJ) and Eclipse Desktop IDE

The Red Hat OpenShift Connector allows developers who work with Red Hat OpenShift to use their preferred development environment without interruption. The extension provides a quick, simple way for developers to work their “inner loop” process of coding, building and testing directly, using their IDE.

Red Hat OpenShift Deployment Extension for Microsoft Azure DevOps: users of this DevOps toolchain can now deploy their built applications to Azure Red Hat OpenShift, or any other OpenShift cluster directly from Microsoft Azure DevOps

Learn:

Understanding Red Hat CodeReady Workspaces - https://access.redhat.com/documentation/en-us/red_hat_codeready_workspaces/1.2/html/administration_guide/understanding-codeready-workspaces

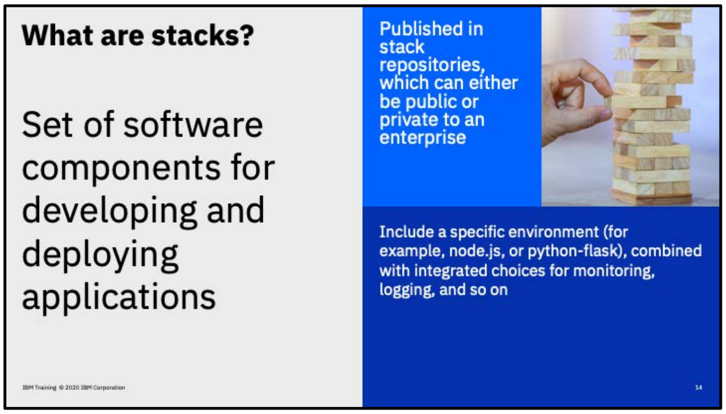
Reference:

https://www.openshift.com/blog/openshift-4-2-expanded-tools-and-services-for-developers


Working with
Stacks
IBM Training © 2020 IBM Corporation

Objectives

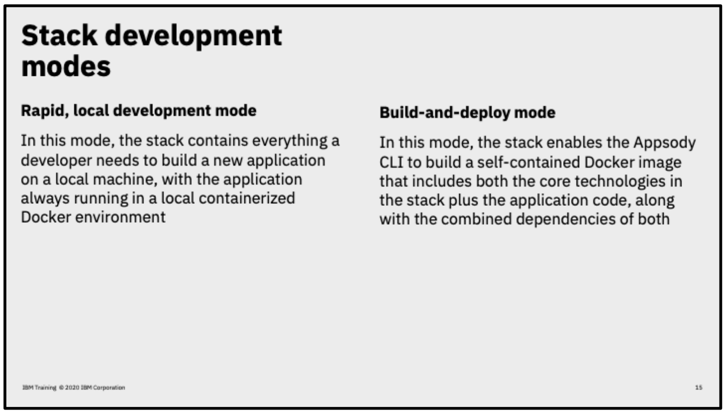
- Set up application stacks
- Manage stacks with the CLI
- Customize stacks
- Work with custom resource definitions



A stack represents a pre-configured set of technologies aimed at simplifying the building of a certain type of cloud native application. This might include a specific environment (for example, node.js, or python-flask), combined with integrated choices for monitoring, logging etc. Stacks are published in stack repositories, which can either be public or private to an enterprise. Developers can then use the Appsody CLI to pull in the appropriate stack for the application they are building. Kabanero contains all the tools for using and contributing to public stack repositories, and a set of curated stacks suitable for the enterprise.

Developers use stacks to simplify building applications that require a specific set of technologies or development patterns. While there are numerous publicly available stacks to choose from, many enterprises want to build their own set of stacks that uphold their specific requirements and standards for how they want their developers to build cloud native applications.

An application stack is created by an **application architect** as a logical set of software components for developing and deploying applications. Although the application architect builds the stacks, all the software components that comprise them should be agreed upon by operations, enterprise, and developer advocates. An application stack is realized as components (such as tools and container images) used at coding time by application developers. Application stack configuration is done via standard Kubernetes operator (CRD) configuration.



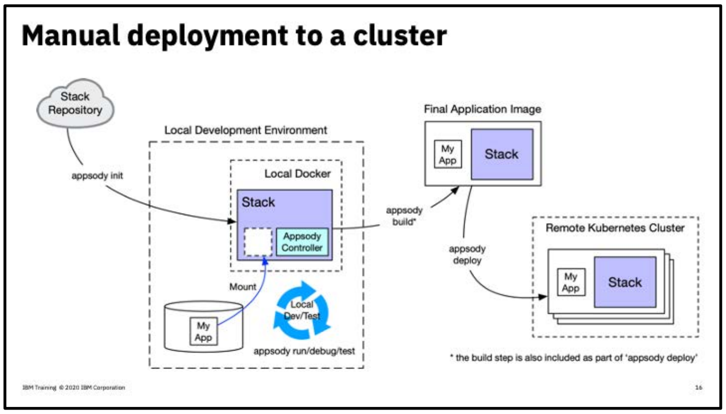
A stack is designed to support the developer in either a rapid, local development mode or a build-and-deploy mode.

Rapid, local development mode

In this mode, the stack contains everything a developer needs to build a new application on a local machine, with the application always running in a local containerized Docker environment.

Build-and-deploy mode

In this mode, the stack enables the Appsody CLI to build a self-contained Docker image that includes both the core technologies in the stack plus the application code, along with the combined dependencies of both.

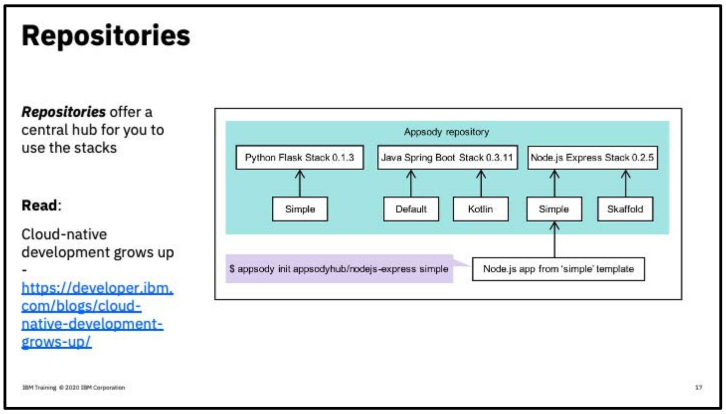


This image shows the flow of how a developer uses Accelerators for Teams to pull down and modify a stack, build it and then deploy it to a remote Kubernetes cluster.

This flow shows the *manual* deployment to a Kubernetes cluster. In more production-orientated environments, GitOps might trigger the build and deploy steps, and Pipelines would drive the deployment.

Reference:

https://developer.ibm.com/blogs/cloud-native-development-grows-up/



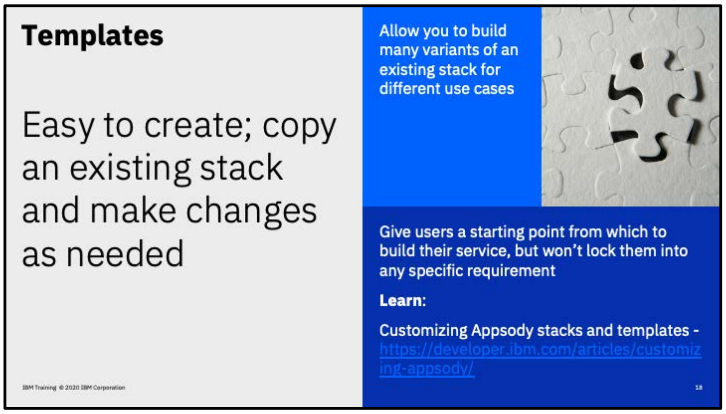
Application stacks are stored in Git repositories. How you organize the application stacks that you intend to use is entirely up to you and might depend on several local requirements. For example, using a different repository for each stack gives you the flexibility to apply version control at the individual stack level. The important part to understand is that each repository you create results in a unique configuration file that a developer might use for accessing any stacks that are stored there.

Reference:

Cloud-native development grows up -

https://developer.ibm.com/blogs/cloud-native-development-grows-up/

The hierarchy of objects is made up of repositories at the top level, followed by stacks, which include a curated image full of technologies, and templates at the lowest level. Repositories offer a central hub for you to use the stacks.
You can use the Appsody CLI to add or remove repositories and initialize new projects based on stacks.



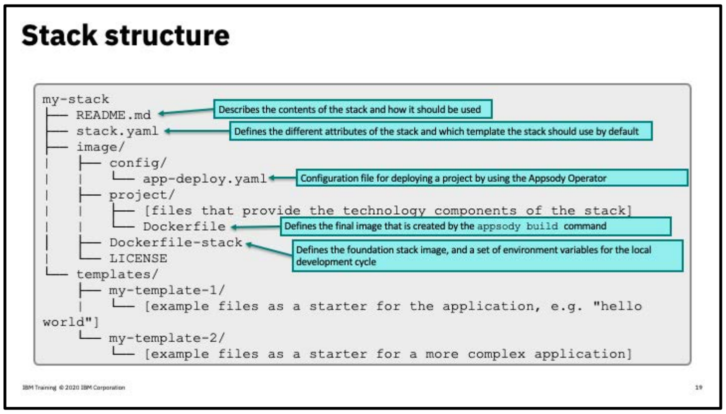
If your stack doesn't require a new function, you can work from an existing template, or create a new one to meet your needs. Templates can provide scaffolding code that is suitable for starting a microservice for a specific task, for example, creating a REST API backend for a web or mobile application.

Templates:

- are easy to create. You simply copy an existing stack and tweak it.
- allow you to build many variants of an existing stack for different use cases.
- give users a starting point from which to build their service but won't lock them into any specific requirement.

Learn:

Customizing Appsody stacks and templates -https://developer.ibm.com/articles/customizing-appsody/



Because a single stack can enable both rapid, local development and build-and-deploy modes, all stacks follow a standard structure. This example represents the source structure of a stack.

As a *stack architect*, you must create the above structure, build it into an actual stack image ready for use by an *application developer* who bases their new application on your stack. Part of your role as a stack architect is to include one or more sample applications (*templates*) to help the application developer get started.

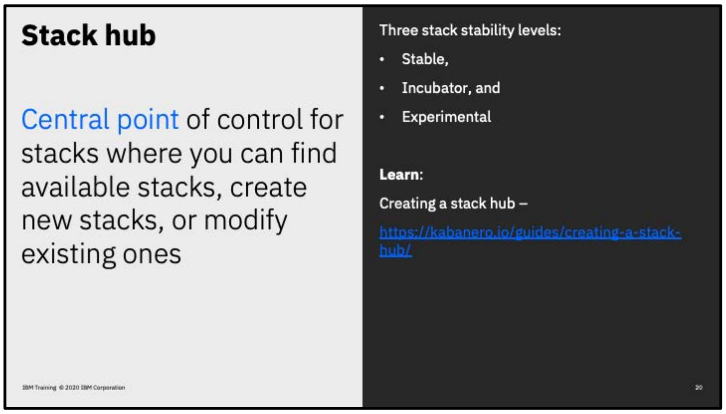
Therefore, when you build a stack, this structure is processed and generates a Docker image for the stack, along with tar files of each of the templates, which can then all be stored and referenced in a local or public repository. The Appsody CLI can access the repository to use the stack to initiate local development.

Reference:

https://appsody.dev/docs/stacks/stacks-overview/

https://github.com/appsody/stacks

https://developer.ibm.com/tutorials/create-appsody-stack/



The Hub is the central point of control for stacks where you can find available stacks, create new stacks, or modify existing ones. There are three **stack stability levels** - stable, incubator, and experimental. Stable stacks are production-ready. You can use the Hub content in the public repository or clone it to provide a private Hub that's based on your requirements. By making changes to the Stacks in the Hub, you can deploy updates to any application that's been built on them, simply by restarting the application. You can also create individual stacks outside of the Hub, that can be supported by the Appsody CLI.

Learn:

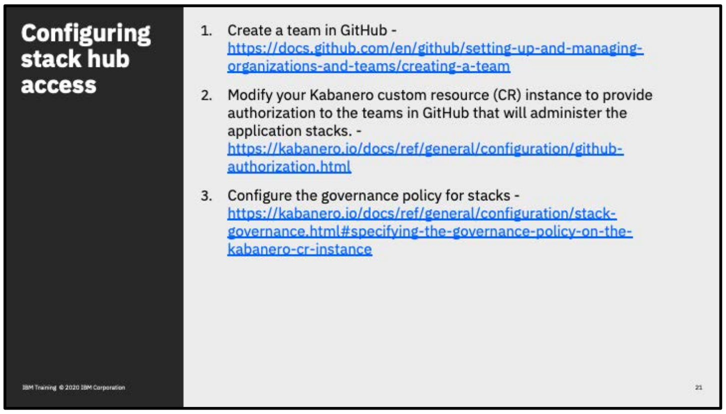
Creating a stack hub –

https://kabanero.io/guides/creating-a-stack-hub/

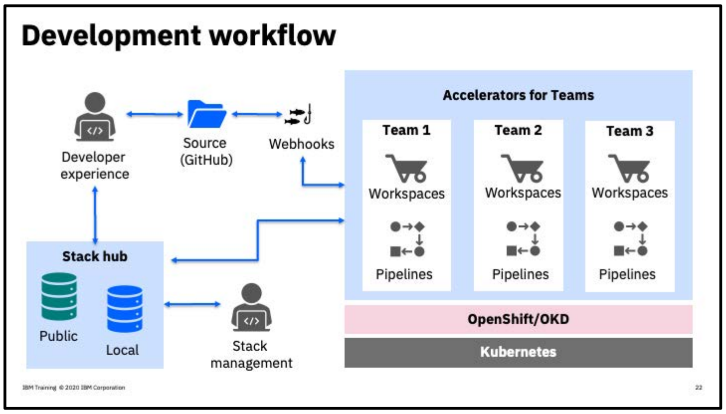
Reference:

https://appsody.dev/docs

https://kabanero.io/guides/creating-a-stack-hub/#understanding-the-configuration-format



1. Create a team in GitHub - https://docs.github.com/en/github/setting-up-and-managing-organizations-and-teams/creating-a-team
2. Modify your Kabanero custom resource (CR) instance to provide authorization to the teams in GitHub that will administer the application stacks. - https://kabanero.io/docs/ref/general/configuration/github-authorization.html
3. Configure the governance policy for stacks - https://kabanero.io/docs/ref/general/configuration/stack-governance.html#specifying-the-governance-policy-on-the-kabanero-cr-instance



The Accelerators for Teams architecture supports the activity of application developers, application architects, and operations teams in continuously delivering applications to Kubernetes environments. The architecture also covers non-functional requirements, such as supporting the implementation of operational policies for application management and security.

This image illustrates the main building blocks of the architecture.

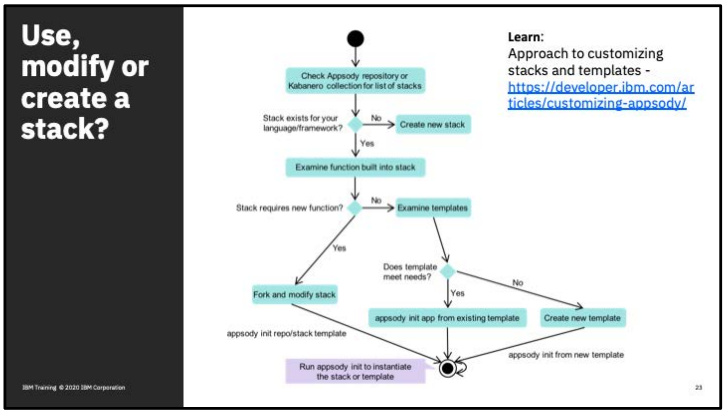
Developer experience: includes tools such as Appsody and Codewind.

Accelerators for Teams sits on the opposite side of the developer experience, as a set of managed pipelines that provide consistent, managed, and governed CI/CD processes that react to repository events that occur during code development.

With these components deployed to a Kubernetes cluster, webhooks are used to connect the source code repositories (such as GitHub) to pipelines that extract the source code, reassemble it according to the structure and runtime specifications of the underlying application stack, execute all the build and verification steps, and then make the final application container pods available in the cluster.

Reference:

https://kabanero.io/docs/ref/general/overview/architecture-overview.html



Depending on your needs, you will either:

- Select an existing stack and base your application on one of its templates
- Modify or extend a stack
- Create an entirely new stack

The question is: how do you decide which option is best for your particular use case? Here is a decision tree to help you make that decision.

Creating or modifying stacks

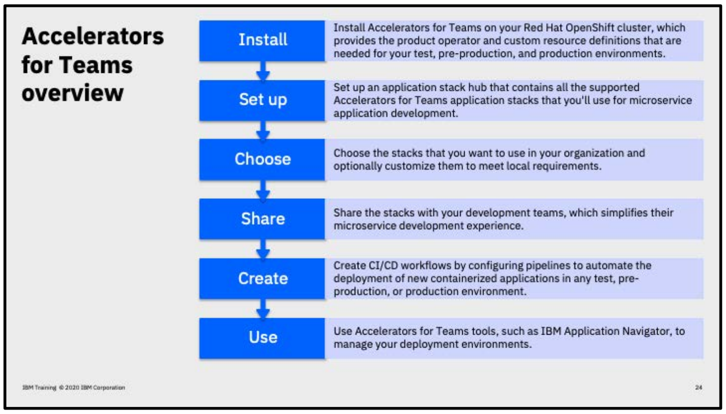
Stacks might contain code that is always part of the application and cannot be modified, while templates contain code that users can modify. So, for example, suppose you decide that everyone in an organization that uses a private repository must have a set of dependencies and libraries in their microservice. In this scenario, you would want to build a set of customized stacks in a private repository.

Other examples of when you would want to create or modify a stack (or multiple stacks) include:

- Your organization needs to create a library of base images that come pre-built with certain common features (health checks, telemetry, security, logging, etc) that should be part of every microservice. These features are controlled by the Dockerfile that is stored in the Appsody repository. You can update the Dockerfile at any time, and the Appsody CLI will pull the latest available version during build time.
- When a user initializes an instance of an Appsody template, the Dockerfile is not downloaded to the user's sandbox as part of the initialization. The appsody buildstep downloads it from the repository each time, compiling the user's code and adding it to the appropriate path in the base image.

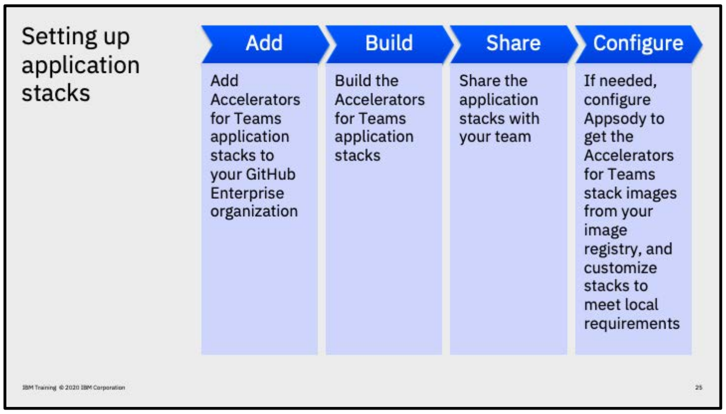
Learn:

Approach to customizing stacks and templates -https://developer.ibm.com/articles/customizing-appsody/



The high-level activities can be summarized in the following workflows:

- Install Accelerators for Teams on your Red Hat OpenShift cluster, which provides the product operator and custom resource definitions that are needed for your test, pre-production, and production environments.
- Set up an application stack hub that contains all the supported Accelerators for Teams application stacks that you'll use for microservice application development.
- Choose the stacks that you want to use in your organization and optionally customize them to meet local requirements.
- Share the stacks with your development teams, which simplifies their microservice development experience.
- Create Continuous Integration / Continuous Deployment (CI/CD) workflows by configuring pipelines to automate the deployment of new containerized applications in any test, pre-production, or production environment.
- Use Accelerators for Teams tools, such as IBM Application Navigator, to manage your deployment environments.



You can use the Cloud Pak installer to add Accelerators for Teams application stacks to your GitHub Enterprise organization and then build your Accelerators for Teams application stack hub.

An advantage of working with stacks in your GitHub Enterprise instance is that you can customize the stacks for your needs and share them with your team members.

It is not required that you set up application stacks in a GitHub Enterprise organization, but it is recommended. Before you can set up stacks, [Accelerators for Teams must be installed] and your GitHub Enterprise instance must be reachable by the OpenShift Container Platform (OCP) cluster.

The following steps describe how to set up stacks and share them with your team:

Add Accelerators for Teams application stacks to your GitHub Enterprise organization.

Build the Accelerators for Teams application stacks.

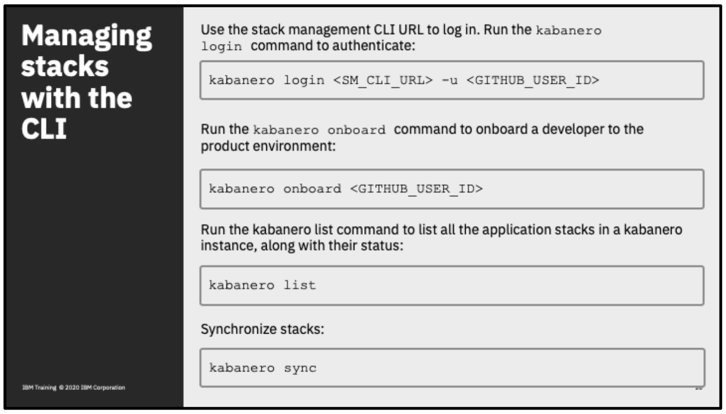
Share the application stacks with your team.

If needed, configure Appsody to get the Accelerators for Teams stack images from your image registry.

If you want to customize the stacks to meet local requirements, you can learn how to do this in the Customizing application stacks guide

Reference

https://www.ibm.com/support/knowledgecenter/SSCSJL_4.1.x/settingup.html

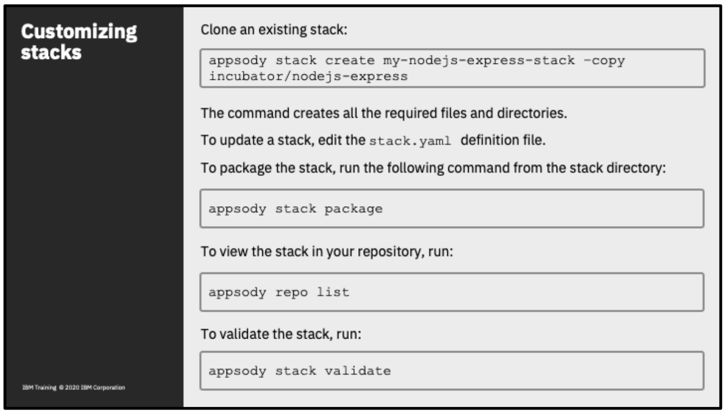


You can use the stack management command-line interface to define and manage the application stacks. Here are some examples of some common commands.

[For a complete list of commands and options, see the documentation.]

Reference:

https://www.ibm.com/support/knowledgecenter/SSCSJL_4.1.x/docs/ref/general/reference/kabanero-cli.html



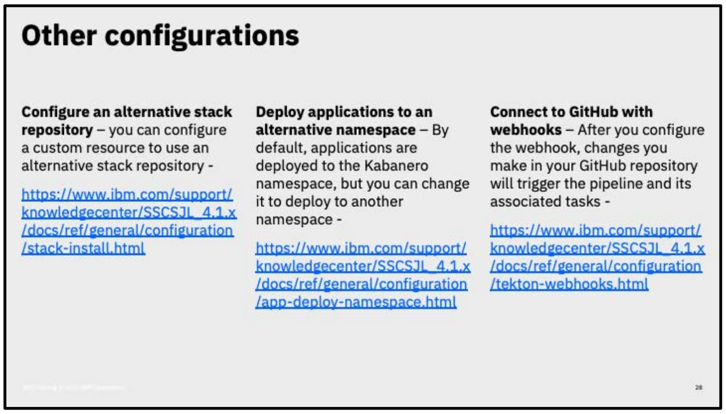
The examples here use an application stack for Javascript development with the Node.js Express framework and illustrate how it can be customized by using the CLI. You can clone an existing stack that you want to use as a base by using the command shown here. The command creates the stack with all the required files and directories. You can add or delete them as needed. Then, you can update the stack by editing the stack.yaml file.

After you configure a stack to suit your requirements, you must package it by running the stack package command before it can be used for application development. Then, you can run repo list to see it in your repository. To test the stack, the first step is to run the stack validate command. The validation process steps through several test operations that check the structure of your application stack before packaging and initializing a project. The project is then run and tested against any generic tests that are defined in the stack. Finally, a production image is generated for deployment.

[For a detailed description of how to customize stacks, see the documentation]

Reference:

https://www.ibm.com/support/knowledgecenter/SSCSJL_4.1.x/guides/guide-working-with-stacks/README.html



The installation process sets up Accelerators for Teams on your Red Hat OpenShift cluster with a default configuration. However, you might want to modify the configuration in some way to suit your local deployment. For example, you might want to configure your custom resource instance to use an alternative namespace or connect the product to GitHub with webhooks. You can:

Configure an alternative stack repository - you can configure a custom resource to use an alternative stack repository - https://www.ibm.com/support/knowledgecenter/SSCSJL_4.1.x/docs/ref/general/configuration/stack-install.html

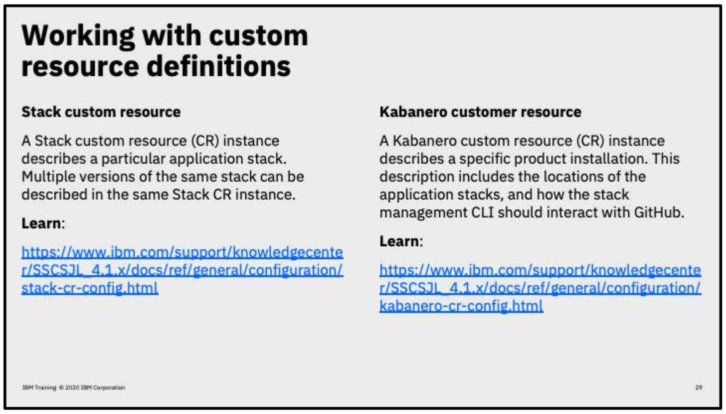
Deploy applications to an alternative namespace - By default, applications are deployed to the Kabanero namespace, but you can change it to deploy to another namespace -

https://www.ibm.com/support/knowledgecenter/SSCSJL_4.1.x/docs/ref/general/configuration/app-deploy-namespace.html

Connect to GitHub with webhooks - A webhook is an outbound HTTP request that helps you create a relationship between your GitHub repository and a particular URL, in this case a pipeline . After you configure the webhook, changes you make in your GitHub repository will trigger the pipeline and its associated tasks. - https://www.ibm.com/support/knowledgecenter/SSCSJL_4.1.x/docs/ref/general/configuration/tekton-webhooks.html

Reference:

https://www.ibm.com/support/knowledgecenter/SSCSJL_4.1.x/using-configuring.html



A custom resource (CR) instance describes a specific installation. This description includes the locations of the application stacks, and how the stack management CLI should interact with GitHub. There are two types of custom resources:

Stack custom resource

A Stack custom resource (CR) instance describes a specific application stack. Multiple versions of the same stack can be described in the same Stack custom resource instance.

Learn:

https://www.ibm.com/support/knowledgecenter/SSCSJL_4.1.x/docs/ref/general/configuration/stack-cr-config.html

Kabanero customer resource

A Kabanero custom resource (CR) instance describes a particular product installation. This description includes the locations of the application stacks, and how the stack management CLI should interact with GitHub.

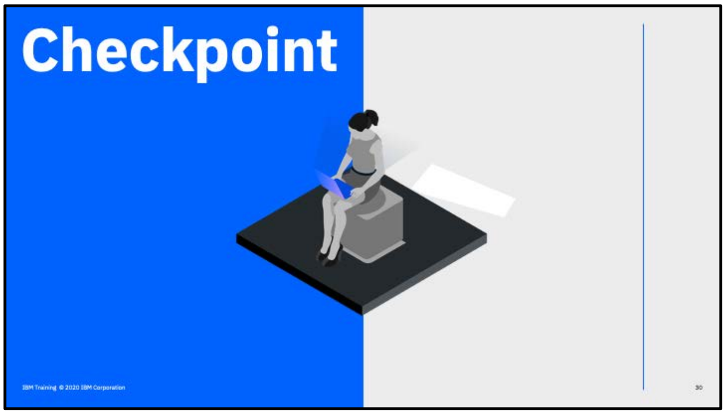
Learn:

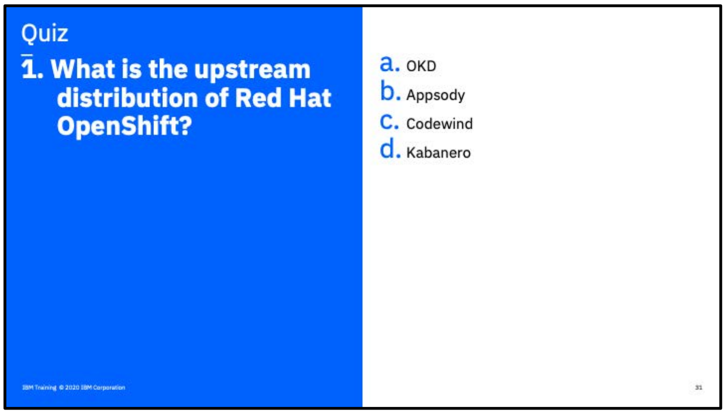
https://www.ibm.com/support/knowledgecenter/SSCSJL_4.1.x/docs/ref/general/configuration/kabanero-cr-config.html

Reference:

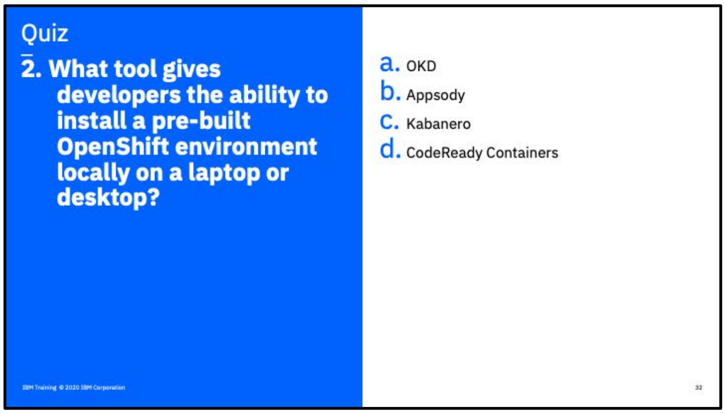
https://www.ibm.com/support/knowledgecenter/SSCSJL_4.1.x/using-reference.html

https://kabanero.io/docs/ref/general/configuration/kabanero-cr-config.html

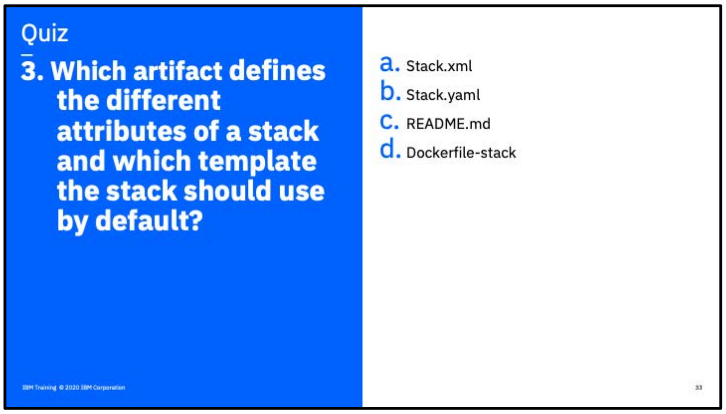

Checkpoint



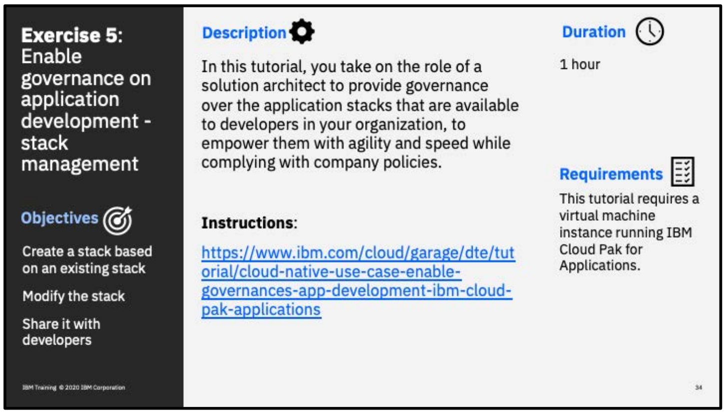
Correct answer: a.



Correct answer: d.



Correct answer: b.



Enable Governance on App Development with IBM Cloud Pak for Applications Stack Management

In this tutorial, you take on the role/persona of a solution architect to provide governance over the application stacks that are available to developers in your organization, to empower them with agility and speed while complying with company policies. (1 hour)

Instructions:

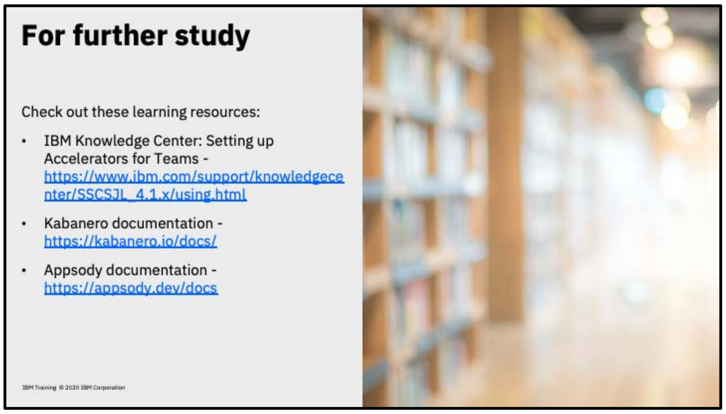
https://www.ibm.com/cloud/garage/dte/tutorial/cloud-native-use-case-enable-governances-app-development-ibm-cloud-pak-applications

https://ibm.ent.box.com/s/kvqhu85aiawyuhanius0vv35y6qihmlh

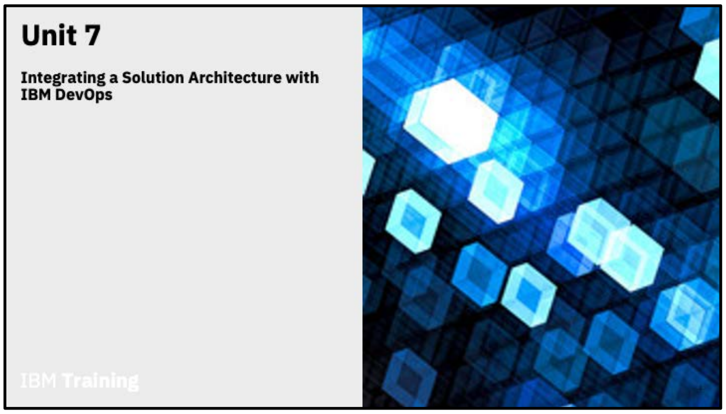
This tutorial requires a virtual machine instance running IBM Cloud Pak for Applications.

For classroom and instructor-led online deliveries, the training provider can provide instructions on how to access the remote lab environment for this course.

For self-paced virtual classroom deliveries, instructions for how to access your lab environment are provided in the "Start here" or "Welcome" section of the course.



- IBM Knowledge Center: Setting up Accelerators for Teams - https://www.ibm.com/support/knowledgecenter/SSCSJL_4.1.x/using.html
- Kabanero documentation - https://kabanero.io/docs/
- Appsody documentation - https://appsody.dev/docs



Unit 7 Integrating a Solution Architecture with IBM DevOps

This unit describes the features and capabilities of IBM DevOps and discusses how to use those capabilities in a solution architecture.

Presentation time: about 45 minutes

Supplemental material:

Videos: 41 minutes

Readings: 1 hour and 25 minutes

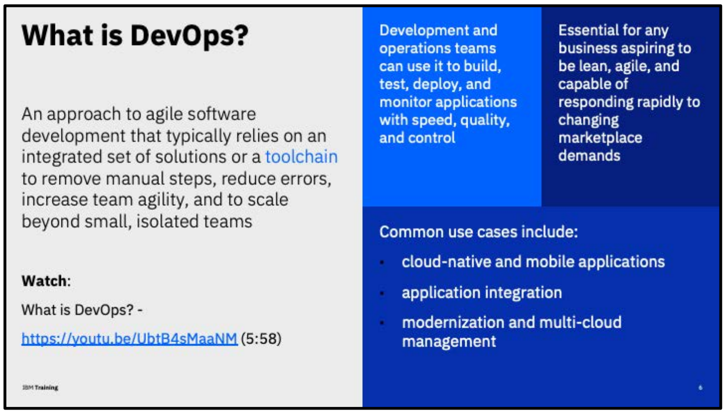
Topics


DevOps
essentials
IBM Training © 2020 IBM Corporation

Objectives

- Define DevOps
- List DevOps principles and methodologies
- Describe the IBM DevOps reference architecture
- Describe OpenShift Pipelines
- Define some basic pipeline concepts
- Discuss rolling updates and rollbacks

Microservices both enable and require DevOps



With modern businesses moving at the speed of cloud, DevOps has become an increasingly common approach to software delivery that development and operations teams use to build, test, deploy, and monitor applications with speed, quality, and control.

DevOps is essential for any business aspiring to be lean, agile, and capable of responding rapidly to changing marketplace demands. It is an approach on the journey to lean and agile software delivery that promotes closer collaboration between lines of business, development, and IT operations while removing barriers between your stakeholders, and your customers.

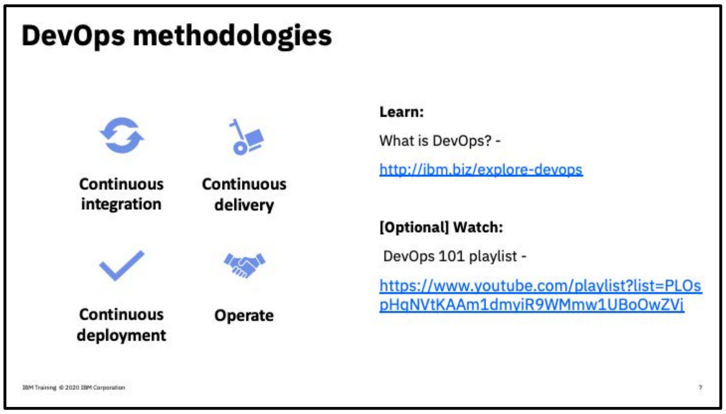
To be essential to customers, all stakeholders in the delivery process need to collaborate. Development teams need to design, develop, deliver and run the software as quickly and reliably as possible. Operations teams need to identify and resolve problems as soon as possible by monitoring, predicting failure, managing the environment and fixing issues. Combining this common approach across Dev and Ops with the ability to monitor and analyze bottlenecks and optimize as quickly as possible gives you DevOps—a collaborative approach across business, development, and operation stakeholders to deliver and run reliable software as soon as possible.

Andrea Crawford, Distinguished Engineer and CTO of DevOps, gives a deeper look into the evolution of DevOps and its underlying goals in her blog post, "What is DevOps?" and the following video:

Watch:

What is DevOps? -

https://youtu.be/UbtB4sMaaNM (5:58)



DevOps methodologies include the following:

Continuous integration, which is where coding, building, integrating, and testing take place.

Continuous delivery, which includes continuous integration, but mainly focuses on product releases.

Continuous deployment, which focuses on automating releases of projects as soon as possible.

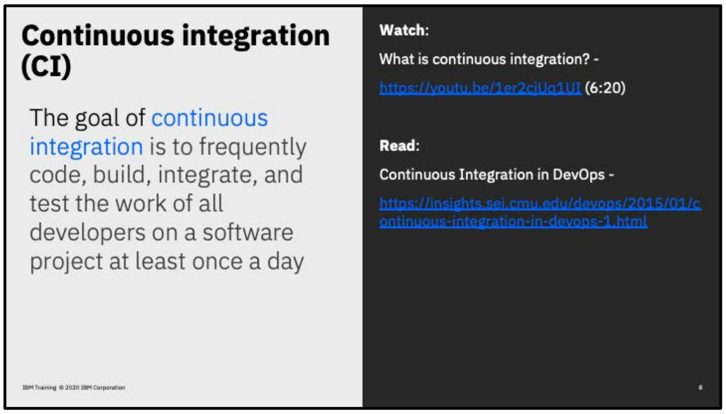
Operate for conducting the development operations of configuration management and continuous monitoring.

Learn:

What is DevOps? **-**

http://ibm.biz/explore-devops

For a deeper dive into some of these DevOps methodogies, see "DevOps 101" series of videos. (https://www.youtube.com/playlist?list=PLOspHqNVtKAAm1dmyiR9WMmw1UBoOwZVj)



Within the rapidly evolving field of product and application lifecycles, continuous integration maintains a simple objective—to frequently code, build, integrate, and test the work of all developers on a software project at least once a day. As a cornerstone of DevOps, continuous integration keeps the code of an individual developer from drifting too far afield from the work of the development group. This technique continually merges source code updates from all developers on a team into a continuous integration server.

If a failure occurs, the development team can refocus and fix it before making any code changes. While this may seem disruptive, in practice it focuses the development team on a single stability metric: a working automated build of the software, according to the Carnegie Mellon University Software Engineering Institute.

Watch:

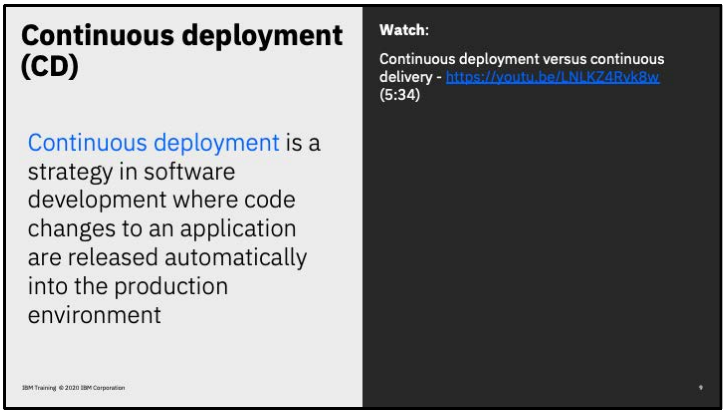
What is continuous integration? -

https://youtu.be/1er2cjUq1UI (6:20)

Read:

Continuous Integration in DevOps -

https://insights.sei.cmu.edu/devops/2015/01/continuous-integration-in-devops-1.html



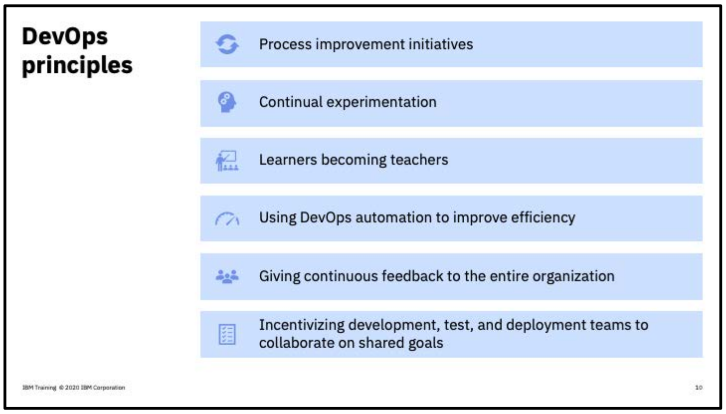
Continuous deployment is a strategy in software development where code changes to an application are released automatically into the production environment. This automation is driven by a series of predefined tests. Once new updates pass those tests, the system pushes the updates directly to the software's users.

Continuous deployment offers several benefits for enterprises looking to scale their applications and IT portfolio. First, it speeds time to market by eliminating the lag between coding and customer value—typically days, weeks, or even months.

In order to achieve this, regression tests must be automated, thereby eliminating expensive manual regression testing. The systems that organizations put in place to manage large bundles of production change—including release planning and approval meetings—can also be eliminated for most changes.

Watch:

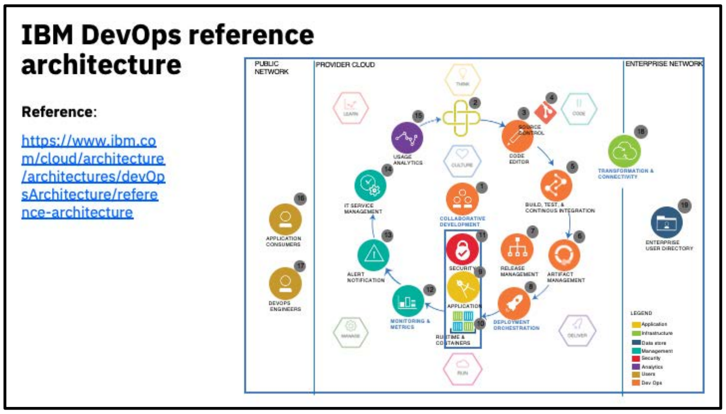
Continuous deployment versus continuous delivery - https://youtu.be/LNLKZ4Rvk8w (5:34)



At the heart of DevOps principles, you will find the idea of collaborative learning and collaborative relationships between development and operations. They focus on increasing the pace of planned work for higher deployment rates, while also upgrading the reliability, stability, resilience, and security of the production environment. To establish an organization based on DevOps principles, you need to emphasize this holistic, whole-system approach across not just the development and operations departments but also every surrounding department and support organization within the company. In return, your whole system should be used to shape your organizational goals.

DevOps principles include:

- Process improvement initiatives to truncate feedback loops to continuously implement needed bug fixes and vulnerability remediation earlier and more cost effectively
- Continual experimentation that encourages risk-taking and learning from success and failure, so continuous attempts will lead to future success and mastery
- Learners becoming teachers and passing along their acquired knowledge to their colleagues
- Using DevOps automation to improve efficiency
- Giving continuous feedback to the entire organization
- Incentivizing development, test, and deployment teams to collaborate on shared goals



The DevOps architecture includes the best of Enterprise Design Thinking, lean startup, agile development, DevOps, and cloud to help enterprise organizations accelerate all phases of the application design, development, and delivery lifecycle. There are a lot of components in the DevOps reference architecture, and several steps in the lifecycle.

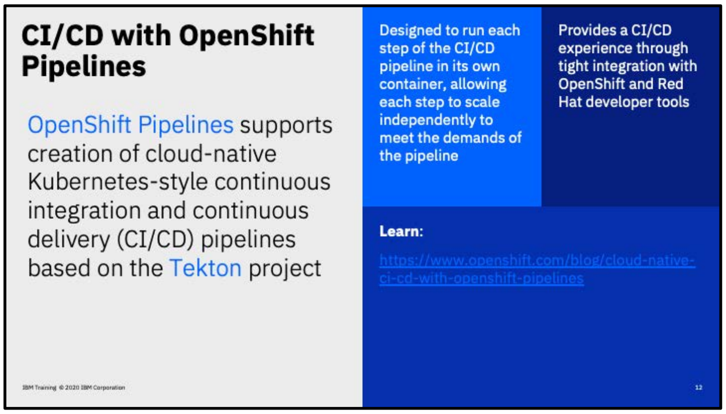
[The link here opens an interactive diagram where you can click on each component to learn more details.]

Some of the key components:

- Collaborative development where Team members and stakeholders continually communicate plans, tasks, issues, and feedback.
- Source control Repository for sharing, storing source code, and versioning code drops, such as GitHub.
- Release management to Help plan, execute, and track a complex release through every stage of the delivery lifecycle.
- Monitoring and metrics that Capture real time and historic application resource and performance data to optimize operation and diagnose problems.
- Alert notifications that Notify the right people on the team or systems when issues occur, and
- IT service management, which Manages the process for responding to operations incidents and delivers the changes to fix any incidents.

Reference:

https://www.ibm.com/cloud/architecture/architectures/devOpsArchitecture/reference-architecture

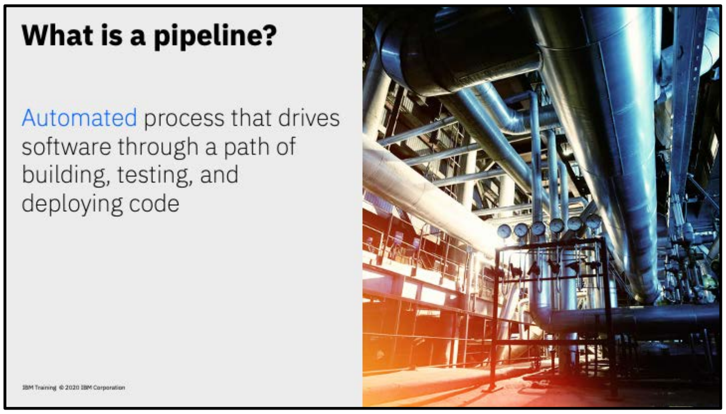


OpenShift Pipelines supports creation of cloud-native Kubernetes-style continuous integration and continuous delivery (CI/CD) pipelines based on the [Tekton](https://tekton.dev) project.

Tekton is an open source project that provides a framework to create cloud-native CI/CD pipelines quickly. As a Kubernetes-native framework, Tekton makes it easier to deploy across multiple cloud providers or hybrid environments. By leveraging the Custom Resource Definitions (CRDs) in Kubernetes, Tekton uses the Kubernetes control plane to run pipeline tasks. By using standard industry specifications, Tekton will work well with existing CI/CD tools such as Jenkins, Jenkins X, Skaffold, and Knative.

Learn:

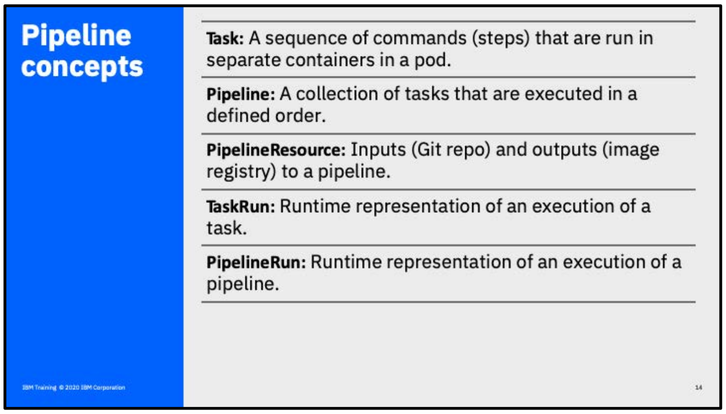
https://www.openshift.com/blog/cloud-native-ci-cd-with-openshift-pipelines



A pipeline in software development is an automated process that drives software through a path of building, testing, and deploying code. By automating the process, the objective is to minimize human error and maintain a consistent process for how software is deployed. Tools that are included in the pipeline could include compiling code, unit tests, code analysis, security, and installer creation. For containerized environments, this pipeline would also include packaging the code into a container to be deployed across the hybrid cloud. A pipeline is critical in supporting continuous integration and continuous deployment (CI/CD) processes.

Reference:

https://www.openshift.com/learn/topics/pipelines



Tekton defines a set of Kubernetes custom resources (CRD) as standard constructs for creating CI/CD pipelines. Here is a brief introduction to these custom resources:

Task: A sequence of commands (steps) that are run in separate containers in a pod.

Pipeline: A collection of tasks that are executed in a defined order.

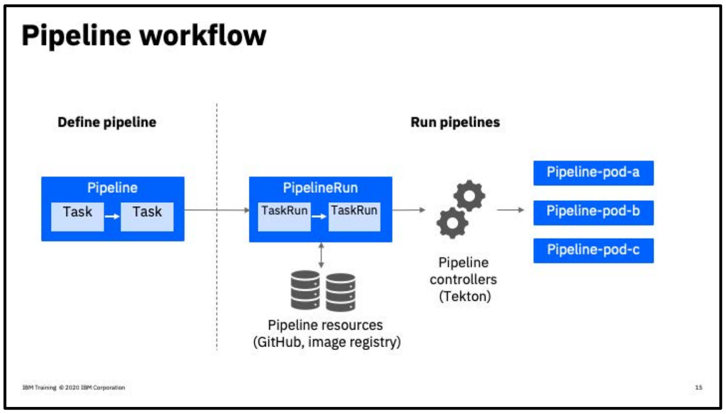
PipelineResource: Inputs (e.g. git repo) and outputs (e.g. image registry) to a pipeline.

TaskRun: Runtime representation of an execution of a task.

PipelineRun: Runtime representation of an execution of a pipeline.

Reference:

https://www.openshift.com/blog/cloud-native-ci-cd-with-openshift-pipelines



This diagram shows how the custom resources would typically interact, from defining the pipeline to running it.

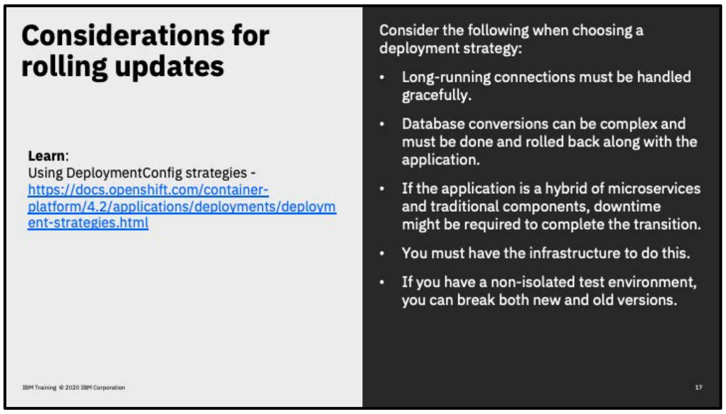
Reference:

https://www.openshift.com/blog/cloud-native-ci-cd-with-openshift-pipelines

In this lightboard video, Sai Vennam and Matt Perrins with IBM Cloud walk through a scenario of taking an existing application and migrating it over to use a cloud-native approach in order to take advantage of increased scalability and higher-level services. These two cloud native experts also demonstrate how to best use DevOps principles to manage the building, testing, and deployment of the application's lifecycle.

Watch:

Cloud native DevOps explained (11 min) - https://youtu.be/FzERTm_j2wE



A *deployment strategy* is a way to change or upgrade an application. The aim is to make the change without downtime in a way that the user barely notices the improvements.

Many deployment strategies are supported through the DeploymentConfig, and some additional strategies are supported through router features.

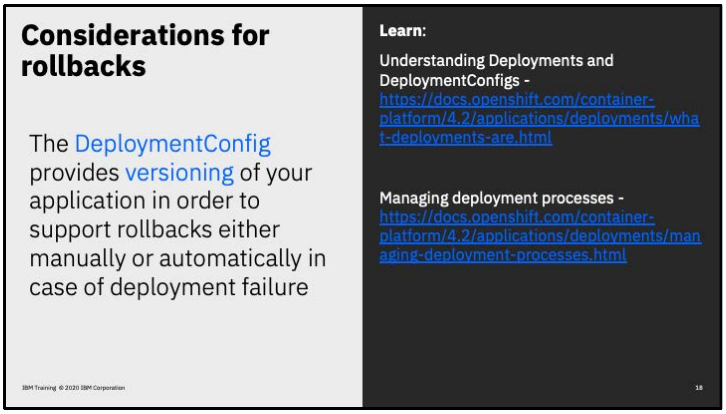
Consider the following when choosing a deployment strategy:

- Long-running connections must be handled gracefully.
- Database conversions can be complex and must be done and rolled back along with the application.
- If the application is a hybrid of microservices and traditional components, downtime might be required to complete the transition.
- You must have the infrastructure to do this.
- If you have a non-isolated test environment, you can break both new and old versions.

[You can learn more about it here.]

Learn:

Using DeploymentConfig strategies - https://docs.openshift.com/container-platform/4.2/applications/deployments/deployment-strategies.html



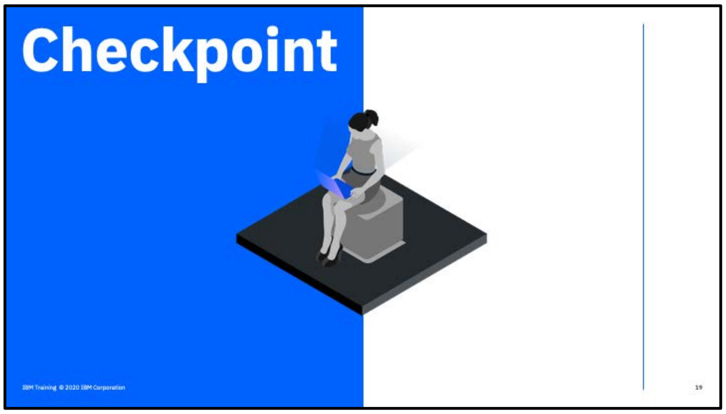
The DeploymentConfig provides versioning of your application in order to support rollbacks either manually or automatically in case of deployment failure.

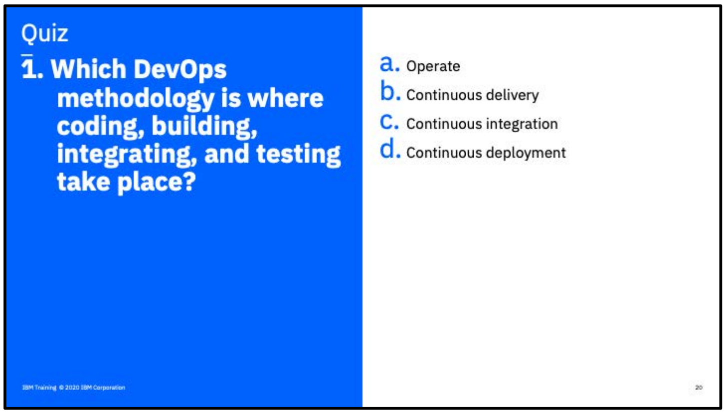
Currently, Deployments do not support automatically rolling back to the last successfully deployed ReplicaSet in case of a failure.

Learn:

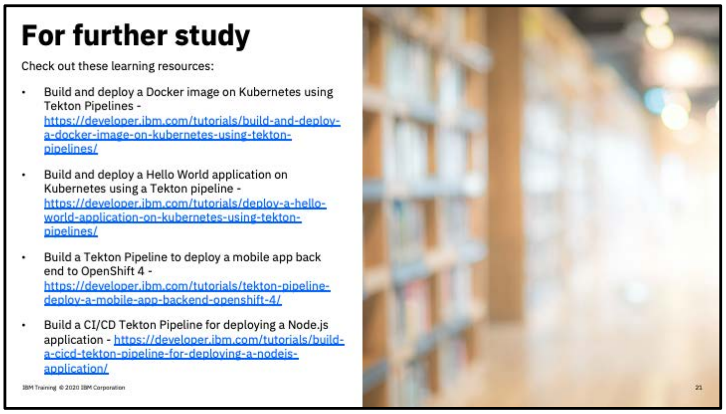
Understanding Deployments and DeploymentConfigs - https://docs.openshift.com/container-platform/4.2/applications/deployments/what-deployments-are.html

Managing deployment processes - https://docs.openshift.com/container-platform/4.2/applications/deployments/managing-deployment-processes.html


Checkpoint
IBM Training © 2020 IBM Corporation
19



Correct answer: c



Check out these learning resources:

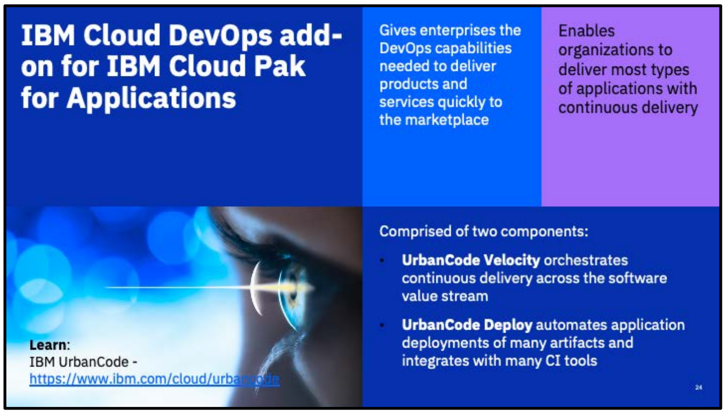
- Build and deploy a Docker image on Kubernetes using Tekton Pipelines - https://developer.ibm.com/tutorials/build-and-deploy-a-docker-image-on-kubernetes-using-tekton-pipelines/
- Build and deploy a Hello World application on Kubernetes using a Tekton pipeline - https://developer.ibm.com/tutorials/deploy-a-hello-world-application-on-kubernetes-using-tekton-pipelines/
- Build a Tekton Pipeline to deploy a mobile app back end to OpenShift 4 - https://developer.ibm.com/tutorials/tekton-pipeline-deploy-a-mobile-app-backend-openshift-4/
- Build a CI/CD Tekton Pipeline for deploying a Node.js application - https://developer.ibm.com/tutorials/build-a-cicd-tekton-pipeline-for-deploying-a-nodejs-application/


IBM Cloud
DevOps
IBM Training © 2020 IBM Corporation
22

Objectives

- Describe IBM Cloud DevOps features and benefits
- Describe IBM Cloud DevOps CI/CD capabilities
- Describe the main functions of UrbanCode Deploy
- Describe the main functions of UrbanCode Velocity
- Describe some IBM Cloud DevOps use cases
- Discuss other DevOps tools
- Explain how to install the IBM Cloud DevOps add-on for IBM Cloud Pak for Applications

23



[Review]

IBM Cloud DevOps add-on for IBM Cloud Pak for Applications gives enterprises the capability to deliver products and services quickly to the marketplace by using continuous delivery.

Cloud DevOps for IBM Cloud Pak for Applications V3.0 is comprised of two components:

UrbanCode® Velocity V1.4 orchestrates continuous delivery across the software value stream.

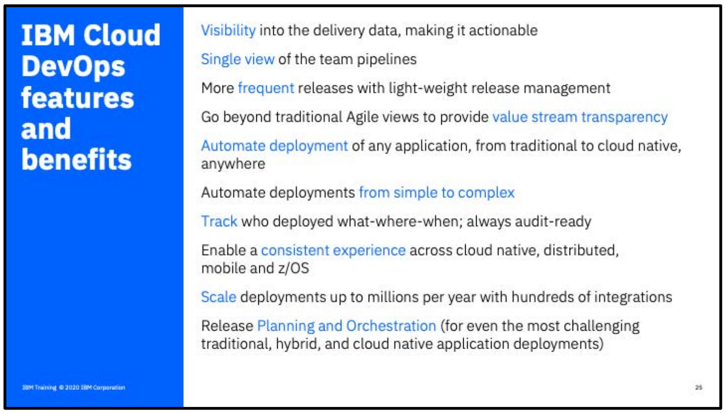
UrbanCode Deploy V7.0.4 automates application deployments of many artifacts and integrates with many CI tools.

Announcement:

https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?infotype=an&subtype=ca&appname=gpateam&supplier=897&letternum=ENUS220-051#abstrx

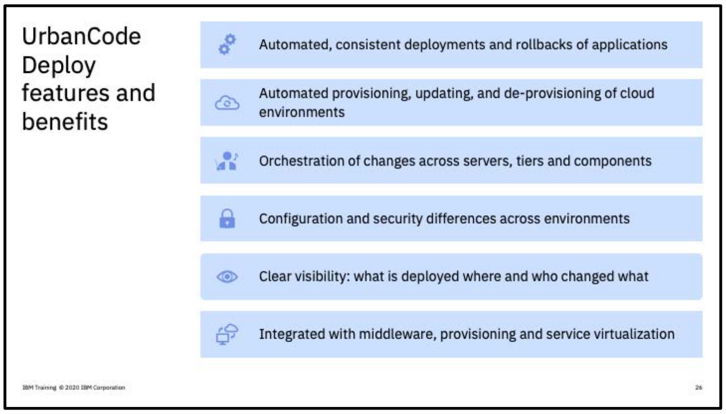
Learn:

IBM UrbanCode - https://www.ibm.com/cloud/urbancode



IBM Cloud DevOps features and benefits include:

- Visibility into the delivery data, making it actionable
- Single view of the team pipelines
- More frequent releases with light-weight release management
- Go beyond traditional Agile views to provide value stream transparency
- Automate deployment of any application, from traditional to cloud native, anywhere
- Automate deployments from simple to complex
- Track who deployed what-where-when; always audit-ready
- Enable a consistent experience across cloud native, distributed, mobile and z/OS
- Scale deployments up to millions per year with hundreds of integrations
- Release Planning and Orchestration (for even the most challenging traditional, hybrid, and cloud native application deployments)



UrbanCode Deploy is a tool for automating application deployments through your environments. It is designed to facilitate rapid feedback and continuous delivery in agile development while providing the audit trails, versioning, and approvals needed in production.

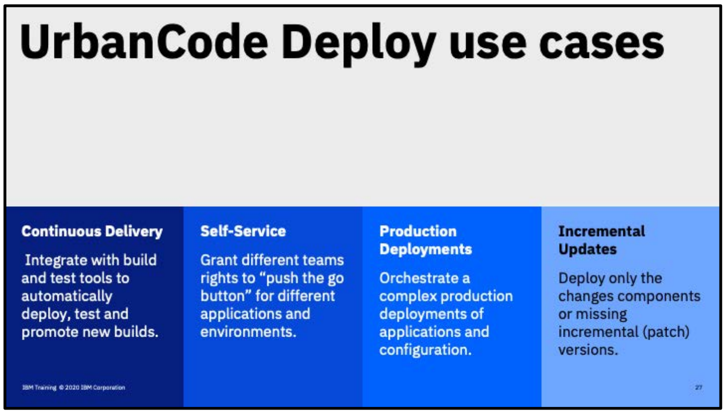
Benefits include:

- Automated, consistent deployments and rollbacks of applications
  Automated provisioning, updating, and de-provisioning of cloud environments
- Orchestration of changes across servers, tiers and components
  Configuration and security differences across environments
- Clear visibility: what is deployed where and who changed what
- Integrated with middleware, provisioning and service virtualization

Reference:

https://www.ibm.com/support/knowledgecenter/SS4GSP_7.0.4/com.ibm.udeploy.doc/topics/c_node_overview.html

https://www.urbancode.com/product/deploy/



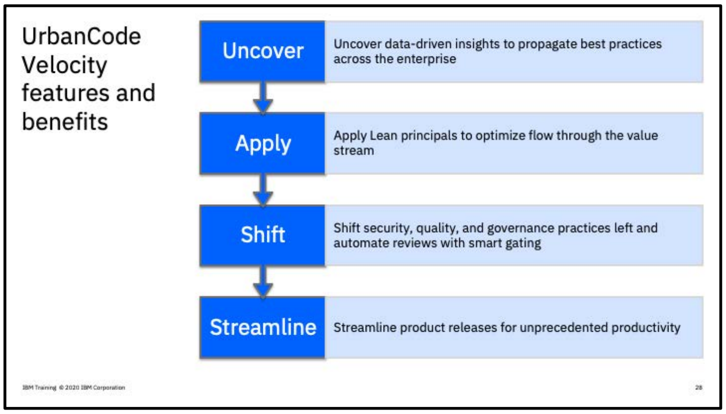
UrbanCode Deploy use cases:

Continuous Delivery: Integrate with build and test tools to automatically deploy, test and promote new builds.

Self-Service: Grant different teams rights to “push the go button” for different applications and environments.

Production Deployments: Orchestrate a complex production deployments of applications and configuration.

Incremental Updates: Deploy only the changes components or missing incremental (patch) versions.



UrbanCode Velocity is a Value Stream Management solution that helps you understand your DevOps practices, implement changes, review change impact, and automate release processes. All these tools are managed from a single user interface.

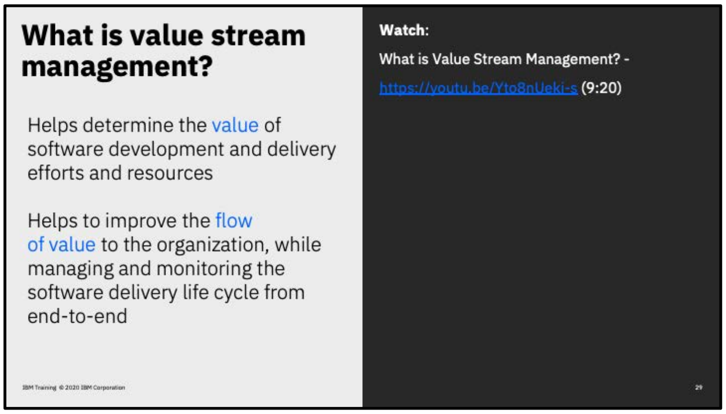
With UrbanCode Velocity:

- Uncover data-driven insights to propagate best practices across the enterprise
- Apply Lean principals to optimize flow through the value stream
- Shift security, quality, and governance practices left and automate reviews with smart gating
- Streamline product releases for unprecedented productivity

Reference:

https://www.ibm.com/support/knowledgecenter/SSCKX6_1.4.x/com.ibm.uvelocity.doc/ucv_version_welcome.html

https://www.urbancode.com/product/urbancode-velocity/



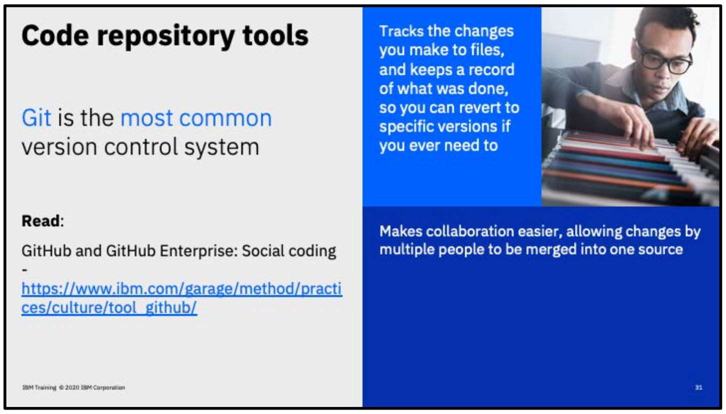
Value stream management helps determine the value of software development and delivery efforts and resources. It also helps to improve the flow of value to the organization, while managing and monitoring the software delivery life cycle from end-to-end.

Watch:

What is Value Stream Management? -

https://youtu.be/Yto8nUeki-s (9:20)

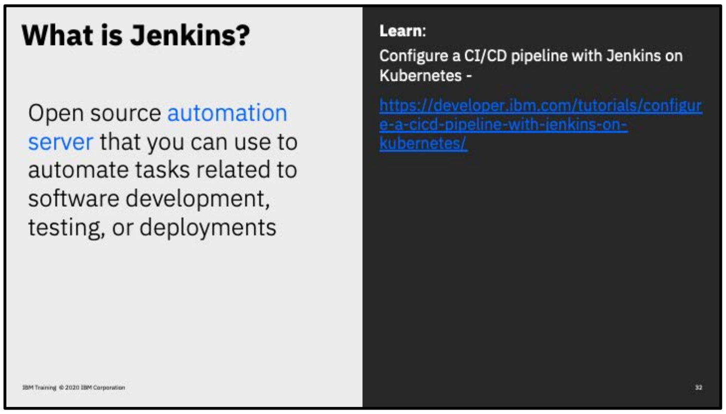
You might use other DevOps tools in your solution architecture, and some popular ones are listed here. This course does not cover all these options, but a few examples are described next.



Git is the most common version control system. It tracks the changes you make to files, and keeps a record of what was done, so you can revert to specific versions if you ever need to. You can have local and remote copies of projects. It makes collaboration easier, allowing changes by multiple people to be merged into one source. For those reasons, it is often named as the source code repository in the examples illustrated in this course.

Read:

GitHub and GitHub Enterprise: Social coding - https://www.ibm.com/garage/method/practices/culture/tool_github/

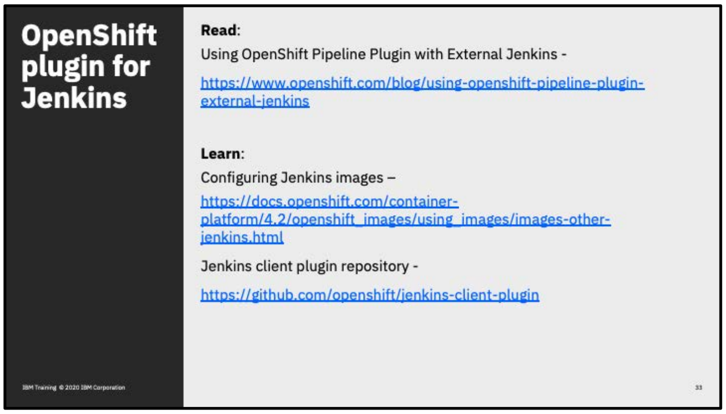


Jenkins is an open source automation server that you can use to automate tasks related to software development, testing, or deployments. This tutorial shows how to configure a CI/CD pipeline with Jenkins.

Learn:

Configure a CI/CD pipeline with Jenkins on Kubernetes -

https://developer.ibm.com/tutorials/configure-a-cicd-pipeline-with-jenkins-on-kubernetes/



OpenShift provides a Jenkins Container as the CI/CD tool to run on an OpenShift cluster. You can use this tool to set up your CI/CD pipelines in order to deploy applications to an OpenShift cluster or elsewhere. However, some use cases require Jenkins to run outside OpenShift and still deploy applications to an OpenShift cluster. For example, some organizations have invested in CI/CD infrastructure that they want to reuse for workloads deployed to OpenShift. In that case, read this blog about how to use an external Jenkins to deploy applications to OpenShift.

Read:

Using OpenShift Pipeline Plugin with External Jenkins -

https://www.openshift.com/blog/using-openshift-pipeline-plugin-external-jenkins

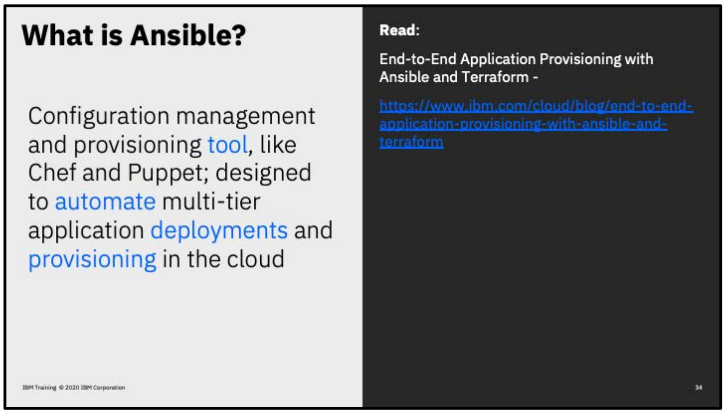
Learn:

Configuring Jenkins images –

https://docs.openshift.com/container-platform/4.2/openshift_images/using_images/images-other-jenkins.html

Jenkins client plugin repository -

https://github.com/openshift/jenkins-client-plugin



Ansible is a config management and provisioning tool, similar to Chef and Puppet. It is designed to automate multi-tier app deployments and provisioning in the cloud. Written in Python, Ansible uses YAML syntax to describe automation tasks. This makes Ansible easy to learn and use.

Read:

End-to-End Application Provisioning with Ansible and Terraform -

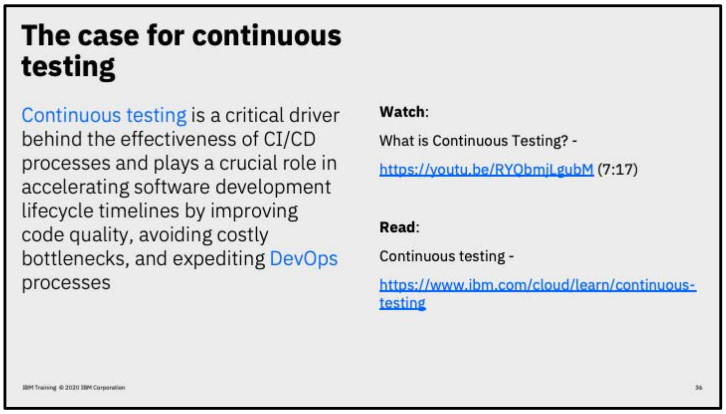
https://www.ibm.com/cloud/blog/end-to-end-application-provisioning-with-ansible-and-terraform

*Source is either Maurits Van Rees or Martin Aspeli and Philipp Von Weitershausen

Reference:

https://zope.maurits.vanrees.org/weblog/archive/2007/10/untested-code-is-broken-code

https://www.slideshare.net/wooda/philipp-von-weitershausen-untested-code-is-broken-code



Continuous testing is the process of incorporating automated feedback at different stages of the software development life cycle (SDLC) in support of better speed and efficiency when managing deployments.

Continuous testing is a critical driver behind the effectiveness of CI/CD (continuous integration/continuous delivery) processes and plays a crucial role in accelerating SDLC timelines by improving code quality, avoiding costly bottlenecks, and expediting DevOps processes.

In a DevOps environment, continuous testing is performed automatically throughout the software development life cycle (SDLC) and works hand in hand with continuous integration to automatically validate any new code integrated into the application.

Watch:

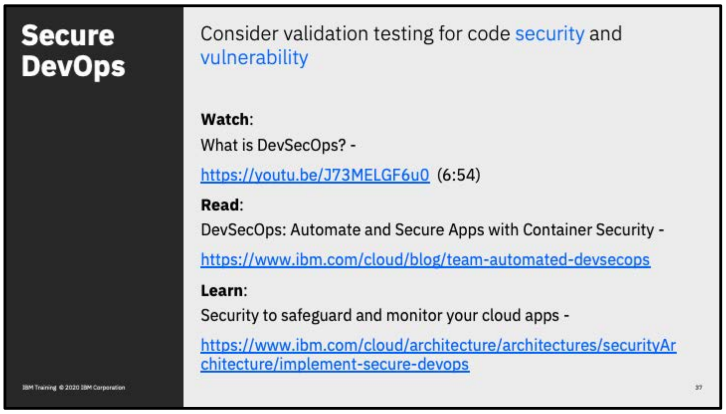
What is Continuous Testing? -

https://youtu.be/RYQbmjLgubM (7:17)

Read:

Continuous testing -

https://www.ibm.com/cloud/learn/continuous-testing



Consider validation testing for code security and vulnerability.

Security testing validates the correctness of design, coding, and integration of software, systems and services. Security testing activities can happen at any of multiple points in the DevOps phases of software and services.

Select the tool or tools and techniques to validate the successful completion of secure coding, configuration, and integration activities.

Vulnerability and incident management is a practice that is shared between development and operations. Vulnerability management refers to detecting and managing vulnerabilities in deployed software and services.

Operational systems need to be scanned and tested regularly to ensure that the integrity of the software, system, or service remains intact. In addition, development personnel must monitor security feeds for vendors and other public sources to learn about newly discovered vulnerabilities in components that are relevant to their software, systems, and services.

Once a vulnerability is detected, the development team must analyze the notification in a timely manner, and resolve, distribute, and deploy fixes for the security defects and vulnerabilities. User notification of vulnerabilities may be required by an organization's policy, subscriber contract, or legal obligation.

Watch:

What is DevSecOps? -

https://youtu.be/J73MELGF6u0 (6:54)

Read:

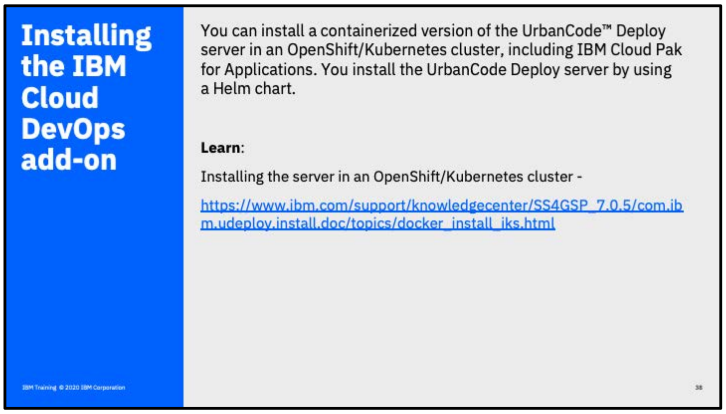
DevSecOps: Automate and Secure Apps with Container Security -

https://www.ibm.com/cloud/blog/team-automated-devsecops

Learn:

Security to safeguard and monitor your cloud apps -

https://www.ibm.com/cloud/architecture/architectures/securityArchitecture/implement-secure-devops

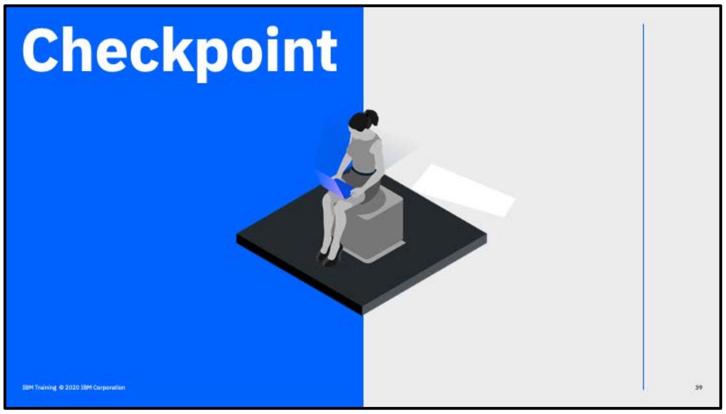


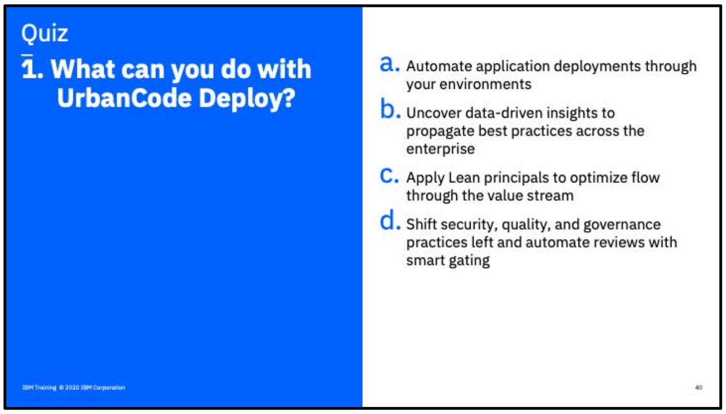
You can install a containerized version of the UrbanCode™ Deploy server in an OpenShift/Kubernetes cluster, including IBM Cloud Pak for Applications. You install the UrbanCode Deploy server by using a Helm chart.

Learn:

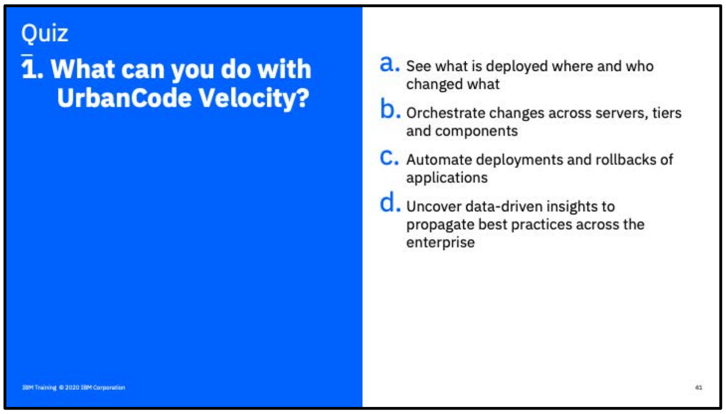
Installing the server in an OpenShift/Kubernetes cluster -

https://www.ibm.com/support/knowledgecenter/SS4GSP_7.0.5/com.ibm.udeploy.install.doc/topics/docker_install_iks.html

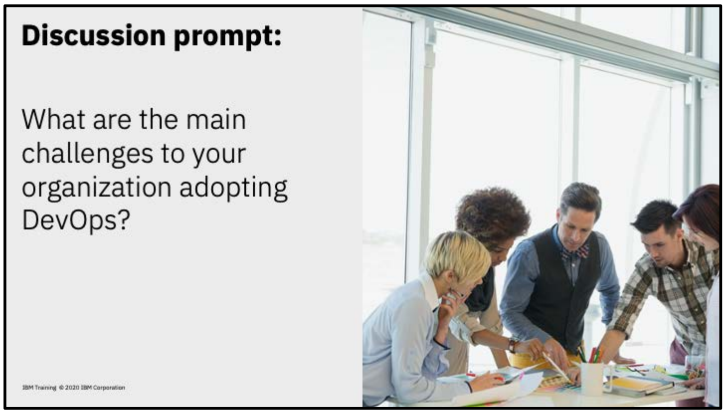

Checkpoint

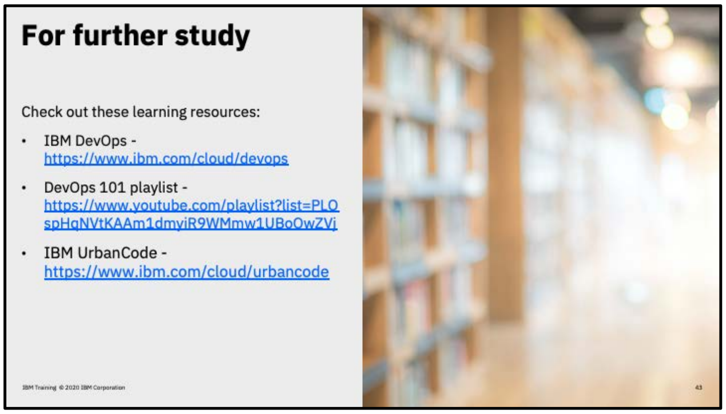


Correct answer: a



Correct answer: d


Discussion prompt:
What are the main challenges to your organization adopting DevOps?



Check out these learning resources:

- IBM DevOps - https://www.ibm.com/cloud/devops
- DevOps 101 playlist - https://www.youtube.com/playlist?list=PLOspHqNVtKAAm1dmyiR9WMmw1UBoOwZVj
- IBM UrbanCode - https://www.ibm.com/cloud/urbancode


Course Summary
IBM Cloud Pak for Applications Solution Architect Workshop
WD108/ZD108
ERC 1.0
IBM Training

Course Objectives

Describe the cloud native development approach

Describe key microservices principles

Explain how containers and container orchestration works

Describe features and capabilities of OpenShift Container Platform

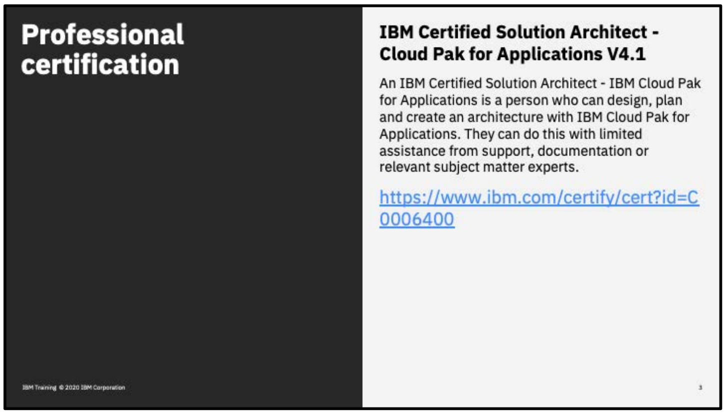
Describe features and capabilities of Cloud Pak for Applications

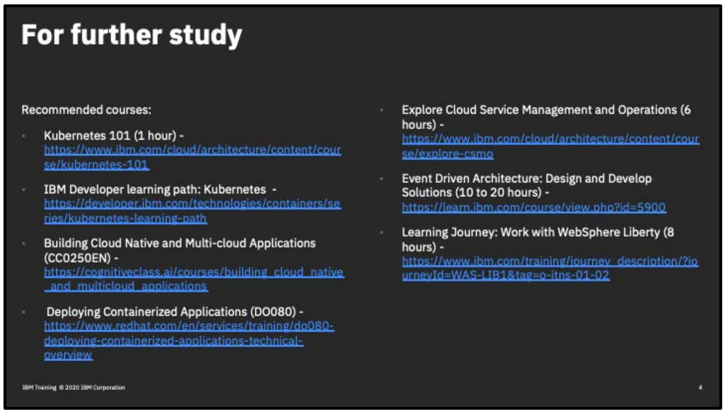
Design a solution architecture for new applications, such as serverless and mobile

Use Accelerators for Teams to manage governance and lifecycles of new applications

Describe the application modernization journey and tools

Design a solution architecture that integrates DevOps


Professional certification
IBM Certified Solution Architect - Cloud Pak for Applications V4.1
An IBM Certified Solution Architect - IBM Cloud Pak for Applications is a person who can design, plan and create an architecture with IBM Cloud Pak for Applications. They can do this with limited assistance from support, documentation or relevant subject matter experts.
https://www.ibm.com/certify/cert?id=C0006400
IBM Training © 2020 IBM Corporation
3



This learning path provides some tutorials to help you get started with Kubernetes - https://developer.ibm.com/technologies/containers/series/kubernetes-learning-path

Resources

IBM Cloud Paks - https://www.ibm.com/cloud/paks

IBM Cloud Pak Playbook - https://ocp42.cloudpak8s.io/playbook/

IBM Cloud Pak for Applications Knowledge Center - https://www.ibm.com/support/knowledgecenter/SSCSJL/welcome.html

Application Modernization Field Guide - https://www.ibm.com/cloud/architecture/files/app-modernization-field-guide.pdf

Application Modernization Reference Architecture - https://www.ibm.com/cloud/architecture/architectures/application-modernization

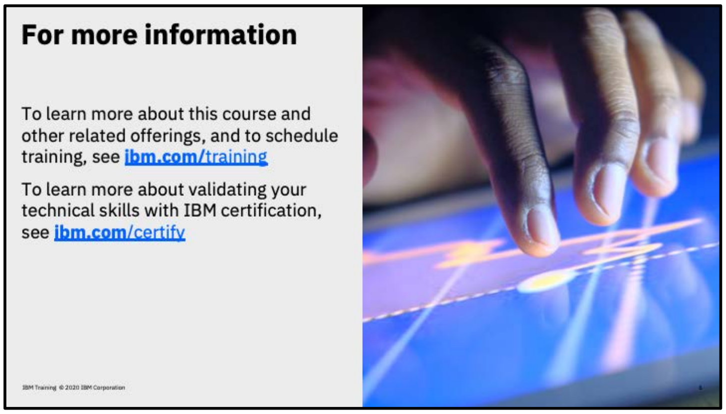
IBM Mobile Foundation documentation and tutorials - https://mobilefirstplatform.ibmcloud.com/tutorials/en/foundation/8.0/all-tutorials/

Kubernetes documentation - https://kubernetes.io/docs/home/

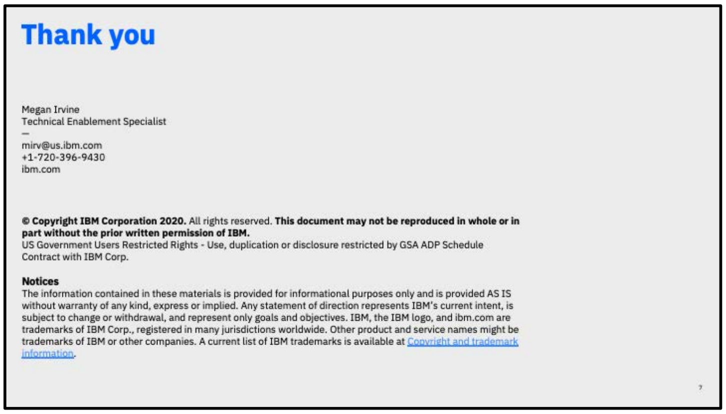
OpenShift documentation - https://docs.openshift.com/

Kabanero documentation - https://kabanero.io/docs/

Appsody documentation - https://appsody.dev/docs



This learning path provides some tutorials to help you get started with Kubernetes - https://developer.ibm.com/technologies/containers/series/kubernetes-learning-path



Thank you
Megan Irvine
Technical Enablement Specialist
—
mirv@us.ibm.com
+1-720-396-9430
ibm.com
© Copyright IBM Corporation 2020. All rights reserved. This document may not be reproduced in whole or in part without the prior written permission of IBM.
US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
Notices
The information contained in these materials is provided for informational purposes only and is provided AS IS without warranty of any kind, express or implied. Any statement of direction represents IBM's current intent, is subject to change or withdrawal, and represent only goals and objectives. IBM, the IBM logo, and ibm.com are trademarks of IBM Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available at Copyright and trademark information.
7