Glossary: Introduction to Containers with Docker, Kubernetes, and OpenShift

Welcome! This alphabetized glossary contains many of the terms you'll find within this course. This comprehensive glossary also includes additional industry-recognized terms not used in course videos. These terms are important for you to recognize when working in the industry, participating in user groups, and participating in other certificate programs.

Container Basics

Term	**Definition**
Agile	An iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer issues.
Client-server architecture	A distributed application structure that divides jobs or workloads between the providers of a resource or service, known as servers, and the users, known as clients.
Container	A standard unit of software, powered by a containerization engine, that encapsulates the application code, runtime, system tools, system libraries, and settings necessary for programmers to efficiently build, ship, and run applications.
Container Registry	A system used for the storage and distribution of named container images. Its primary functions include storing images and enabling their retrieval.
CI/CD pipelines	A sequence of automated steps designed to streamline the process of delivering new software versions. It enhances the software development life cycle by focusing on automation to improve software delivery efficiency and quality.
Cloud native	A type of application specifically designed for a cloud computing architecture. They operate directly in the cloud and take full advantage of cloud-based software delivery models.
Daemon-less	A container runtime that operates without running any specific background process (daemon) to create and manage objects such as images, containers, networks, and volumes.
DevOps	A set of practices, tools, and a cultural philosophy aimed at automating and integrating processes between software development and IT teams to improve efficiency and collaboration.
Docker	An open container platform that enables the development, deployment, and execution of applications within containers.
Dockerfile	A configuration file with instructions for creating a Docker image. Docker can automatically build images by following the instructions specified in a Dockerfile.
Docker client	The Docker client is the main interface for many Docker users. When you issue commands like docker run, the client sends these instructions to the Docker daemon (dockerd), which executes them using the Docker API. The Docker client is capable of communicating with multiple Docker daemons.
Docker Command Line Interface (CLI)	The Docker client provides a command line interface (CLI) that allows users to execute commands for building, running, and stopping applications through interaction with a Docker daemon.
Docker daemon (dockerd)	The core engine that is responsible for creating and managing Docker objects, such as images, containers, networks, and volumes.
Docker Hub	Docker Hub is a cloud-based registry service that allows you to create, manage, and deliver containerized applications. It provides a centralized platform for storing, sharing, and deploying Docker images. With Docker Hub, you can easily collaborate with your team, automate your workflow, and ensure consistency across your environments.
Docker localhost	Docker offers a host network mode that allows containers to use the host's network stack. This means that a localhost in a container resolves to the physical host, instead of the container itself.
Docker remote host	A Docker Engine operating on a machine, either on the local network or externally, that is accessible for managing Docker containers remotely. This remote host can be accessed via the Docker Engine API by exposing certain ports, enabling external queries.
Docker networks	Docker networks provide a way to isolate container communications, allowing you to create separate networks for different groups of containers. This isolation ensures that containers can only communicate with each other if they are connected to the same network, improving security and reducing the risk of unauthorized access.
Docker plugins	Docker plugins are extensions that add functionality to Docker. For example, a storage plugin connects external storage platforms to your Docker environment, allowing you to keep data even after a container is stopped. Other plugins might provide features like networking, security, or logging.
Docker storage	The methods used to maintain data beyond the lifecycle of a container. It includes techniques like volumes and bind mounts. Volumes are shared directories between the host system and a container, providing a way to persist data. Bind mounts allows a directory on the host machine to be directly mounted into a container as if it were a volume that enables direct access to host files from within the container.
IBM Cloud Container Registry	A fully managed private registry that stores and distributes container images.
Image	A static file that includes all the source code, libraries, and dependencies needed to run an application. Images serve as a templates or blueprints for a container.
Immutability	Refers to the read-only nature of images. Any modifications to an image result in the creation of a new image.
LXC	An OS-level virtualization technology that allows for the creation and management of isolated Linux virtual environments (VEs) on a single host system.
Microservices	A cloud-native architectural approach in which a single application contains many loosely coupled and independently deployable smaller components or services.

Namespace	A Linux kernel feature that isolates and virtualizes system resources, ensuring that processes within a namespace can only interact with resources or other processes within the same namespace. Namespaces are fundamental to Docker's isolation model and exist for various resources, including networking, storage, and process management.
Operating System Virtualization	An OS-level technology that enables the kernel to support multiple isolated user space instances, often referred to as containers, zones, virtual private servers, partitions, virtual environments, virtual kernels, or jails.
Private Registry	A Private Registry is a registry that restricts access to images, allowing only authorized users to view and use them. This provides an additional layer of security and control over your containerized applications, ensuring that sensitive images are not exposed to unauthorized users.
REST API	A RESTful API conforms to the constraints of REST architectural style and allows for interaction with RESTful web services.
Registry	A hosted service that stores repositories of Docker images and interacts with clients through the Registry API.
Repository	A collection of Docker images, which can be shared by uploading to a registry server. Each image within the repository can be labeled with tags to differentiate versions or configurations.
Server Virtualization	The process of dividing a physical server into multiple unique and isolated virtual servers by using a software application. Each virtual server operates independently, capable of running its own operating system.
Serverless	A cloud-native development methodology that enables developers to create and deploy applications without needing to manage servers.
Tag	A label that differentiates Docker images within a repository.

Kubernetes Basics

Term	Definition
Automated bin packing	A technique that increases resource utilization and cost savings using a mix of critical and best-effort workloads.
Batch execution	Handles batch and continuous integration workloads.If configured, it automatically replaces failed containers, ensuring continuous operation.
Cloud Controller Manager	A part of the Kubernetes control plane that handles cloud-specific control logic, integrating Kubernetes with various cloud provider services.
Cluster	A set of worker machines, known as nodes, that run and manage containerized applications in a coordinated manner.
Container Orchestration	The process of automating the lifecycle management of containerized applications.
Container Runtime	The software that executes and manages containers, ensuring they run as intended.
Control Loop	A continuous, non-terminating loop that maintains the desired state of a system by monitoring and adjusting as needed.
Control plane	The core component of container orchestration that provides the API and interfaces for defining, deploying, and managing the lifecycle of containers.
Controller	Control loops in Kubernetes that watch the state of the cluster and make or request changes as necessary.
Data (Worker) Plane	The layer that provides the essential resources like CPU, memory, network bandwidth, and storage needed for containers to operate and interact within a network.
DaemonSet	Within a Kubernetes cluster, a DaemonSet serves as a controller object that guarantees the scheduled execution of a Pod specification across a designated set of worker nodes. This construct is particularly well-suited for deploying essential system daemons, such as log collection agents or monitoring tools, that require ubiquitous presence on all cluster nodes.
Declarative Management	A Kubernetes approach where the desired state of the system is defined, and Kubernetes continuously monitors and adjusts to ensure the current state aligns with this predefined desired state.
Deployment	A Kubernetes resource that manages updates and scaling for both Pods and their underlying ReplicaSets, facilitating the automated rollout and rollback of application versions.
Designed for extensibility	Is the Kubernetes ability to add features to cluster without modifying source code.
Docker Swarm	Automates deployment of containerized applications, designed specifically to work with Docker Engine and tools.
Ecosystem	The array of services, support, and tools that are broadly available and compatible with Kubernetes.
etcd	A key-value store that is used as a reliable and consistent data store for distributed systems.
Eviction	The process of terminating and removing one or more Pods from a Node.
Imperative commands	Commands that directly create, update, and delete live objects.
Imperative Management	An approach that defines steps and actions to reach a desired state.
Ingress	Ingress is a Kubernetes component that manages external access to services in a cluster. It provides a single entry point for incoming traffic, allowing you to route requests to multiple services within the cluster.
IPv4/IPv6 dual stack	The IPv4/IPv6 dual stack feature assigns both IPv4 and IPv6 addresses to Pods and Services. This allows your cluster to support both IPv4 and IPv6 networks, ensuring compatibility with a wide range of environments.

Job	A task with a defined start and end, commonly used for executing finite or batch operations within a Kubernetes cluster.
Kubectl	The command-line interface tool for interacting with the Kubernetes control plane allows you to manage clusters and their resources.
Kubelet	The primary "node agent" in Kubernetes operates on each node, managing containers, executing tasks, and facilitating communication with the control plane.
Kubernetes	The foremost open-source platform standard for container orchestration.
Kubernetes API	The Kubernetes API is a RESTful interface that serves Kubernetes functionality and stores the cluster state.It offers a programmatic method for interacting with the cluster, enabling task automation, the creation of custom tools, and integration with other systems.
Kubernetes API Server	The component in Kubernetes that validates and configures data for API objects in Kubernetes.
Kubernetes Controller Manager	This crucial component operates the various controller processes that continuously observe and regulate the Kubernetes cluster, ensuring that its actual state aligns with the desired configuration and settings.
Kubernetes Cloud Controller Manager	A Kubernetes component that embeds cloud-specific control logic in Kubernetes.
Kubernetes Proxy	A network proxy service within Kubernetes that handles network traffic rules, facilitating and managing communication between Pods and services within the cluster.
kube-scheduler	The Kubernetes component responsible for selecting the appropriate node on which newly created Pods will run.
Label Selector	In Kubernetes, a Label Selector functions as a filtering mechanism employed to identify resources based on assigned labels. These labels, key-value pairs attached to entities like Pods or Services, provide supplementary metadata for organizational purposes. By leveraging Label Selectors, users can precisely choose specific resources within the cluster, simplifying administration and resource management tasks.
Labels	Labels provide a way to attach metadata with Kubernetes objects such as Pods or Services, which simplifies the process of putting objects into user-defined groups based on characteristics.These are a pair of key values, the strings, and places that listen for string or integer values. Labels allow us to tag and group resources, which in turn makes managing and organizing your cluster much easier.
Load balancing	The practice of distributing network traffic evenly across multiple Pods to optimize performance and ensure availability within a Kubernetes cluster.
Marathon	An Apache Mesos framework for scaling container infrastructure.
Namespace	A feature in Kubernetes that supports isolation of resources within a single cluster.
Node	A worker machine within a Kubernetes cluster responsible for running containerized applications, managed by the control plane.
Nomad	A cluster management and scheduling tool supporting Docker and other apps.
Object	Represents an entity in the Kubernetes system.
Persistence	The characteristic of an object in Kubernetes to exist until it is explicitly modified or removed, ensuring continuity and stability of resources.
Preemption	Preemption is a mechanism that helps a pending Pod find a suitable Node by evicting low-priority Pods. This ensures that high-priority Pods are scheduled quickly, even if it means terminating lower-priority Pods.
Pod	The fundamental execution unit in Kubernetes, representing one or more containers running together on a cluster. A Pod provides a shared environment and resources for its containers, allowing them to work together closely.
Proxy	An intermediary server for remote services.
ReplicaSet	A Kubernetes resource designed to ensure that a specified number of identical Pods are running at all times, providing redundancy and scaling capabilities for applications.
Self-healing	The capability of Kubernetes to automatically restart, replace, reschedule, and kill failing containers.
Service	A network abstraction that exposes a group of Pods as a single, consistent network endpoint. It provides a stable IP address and load-balancing capabilities, simplifying access and management of the underlying application.
Service Discovery	The process of discovering and connecting to Pods using their IP addresses or a single DNS name.
StatefulSet	A StatefulSet is a Kubernetes component that manages the deployment and scaling of Pods, ensuring ordering and uniqueness. It is used to deploy stateful applications, such as databases, that require a specific ordering and uniqueness of Pods.
Storage	Storage refers to the mechanisms used to provide persistent and temporary storage for Pods. This includes Persistent Volumes, StorageClasses, and Volume Claims, which allow you to manage storage resources and provide persistent storage for your applications.
Storage Orchestration	Storage Orchestration is a mechanism that automatically mounts storage systems for Pods. It provides a way to manage storage resources and provide persistent storage for your applications, making it easier to deploy and manage stateful applications.
Workload	An application running on Kubernetes.

Managing Applications with Kubernetes

Term	**Definition**
Cluster Autoscaler	Also known as CA. An API resource that autoscales the cluster itself by increasing and decreasing the number of available nodes that pods can run on.

Config Map	A Kubernetes object that allows to store configuration data as key-value pairs. It provides a way to manage and inject configuration settings into containers, decoupling configuration from application code.
Horizontal Pod Autoscaler	An API resource in Kubernetes that automatically adjusts the number of Pod replicas based on specified metrics like CPU utilization or custom metrics.
IBM Cloud catalog	A collection of various services provided by IBM, ranging from visual recognition and natural language processing to creating chatbots.
Linguistic Analysis	The process of detecting the tone in a given text.
Persistent Volume	Within the Kubernetes API, a Persistent Volume acts as an abstraction for persistent storage. PVs offer flexible and pluggable storage options, existing independently of any Pod's lifecycle. This ensures enduring storage for applications running within the cluster.
Persistent Volume Claim	A Kubernetes API object representing storage that persists independently of any Pod's lifecycle, providing durable storage for applications within the cluster.
Rolling Updates	A strategy that provides a way to roll out application changes in an automated and controlled fashion throughout your pods. Rolling updates work with pod templates such as deployments. Rolling updates allow for rollback if something goes wrong.
Secrets	Secrets are objects that store sensitive information, such as passwords, OAuth tokens, and ssh keys. They provide a secure way to manage sensitive data, making it easier to deploy and manage.
Service binding	The process of establishing connections between applications and external or backing services, such as REST APIs, databases, and event buses.
Tone Analyzer Service	An IBM Cloud Service that uses linguistic analysis to detect tone in a given text.
Vertical Pod Autoscaler (VPA)	Also known as VPA. An API resource that adjusts the CPU and memory resources allocated to an existing Pod, allowing a service to scale vertically within a cluster.
Volume	A directory containing data, accessible to multiple containers in a Pod.
Volume Mount	Mounting of the declared volume into a container in the same Pod.
Volume Plugin	Facilitates the integration of storage into a Pod within Kubernetes, enabling containers within the Pod to access and use persistent storage resources.

OpenShift Basics

Term	Definition
A/B testing	A strategy often used to test new features in front-end applications. It involves evaluating two versions, A and B, to determine which performs better in a controlled environment.
Build	The process of transforming inputs into a resultant object.
BuildConfig	An OpenShift-specific object that outlines the steps and strategy for executing a build. The BuildConfig serves as a blueprint for the build process, defining how input sources are utilized to generate the desired output.
Canary Deployments	A deployment strategy where a new version of an application is gradually rolled out to a subset of users or traffic, allowing real-world testing before a full deployment.
Circuit breaking	A method to prevent errors in one microservice from cascading to other microservices.
Configuration Change	A trigger that causes a new build to run when a new BuildConfig resource is created.
Control Plane	The control plane takes the desired configuration and its view of the services and dynamically programs and updates the proxy servers as the environment changes.
Custom build strategy	In OpenShift, a build strategy that enables users to define and use their own custom builder image.
Custom builder images	Regular Docker images that contain the logic needed to transform the inputs into the expected output.
CRDs (Custom Resource Definitions)	Custom code that defines a resource to add to your Kubernetes API server without building a complete custom server.
Custom controllers	Custom controllers manage Custom Resource Definitions (CRDs) by continuously reconciling their current state with the desired state specified in the CRD's definition, ensuring alignment and performing necessary actions to achieve consistency and functionality as intended.
Data plane	The layer in a network architecture that handles communication between services. If a service mesh is absent, the network cannot identify the type of traffic that flows, the source, and the destination and make any necessary decisions.
Enforceability (Control)	Istio enforces policies across the system to ensure fair resource distribution among consumers, helping to manage and regulate the use of services.
Envoy proxy	A proxy used by Istio to intercept and manage all network traffic. Envoy provides features such as load balancing, observability, and security based on its configuration.
Human operators	Individuals who manage systems by deploying services, identifying issues, and implementing solutions to ensure smooth operations.
Image Change	A trigger to rebuild a containerized application when a new or updated version of an image is available. For example, if an application is based on a Node.js image, it will be rebuilt automatically when updates like security patches are released for Node.js.
ImageStream	In OpenShift, an abstraction used to reference container images. Each ImageStream contains metadata such as image IDs or digests that uniquely identify them. ImageStreams themselves do not store image data but serve as pointers to specific image digests.

ImageStream Tag	An identifier within an ImageStream that points to a specific image in a registry. It represents a named reference to a particular version or variant of an image.
Istio	A popular, platform-independent service mesh that works with Kubernetes. Istio controls traffic and API calls between services, conducts tests, and simplifies network service management.
Man-in-the-middle attacks	The cyber-attack in which the attacker secretly intercepts and relays messages between two parties who believe they are communicating directly with each other.
Observability	The ability to monitor and understand traffic flow within a service mesh, trace call flows and dependencies, and view metrics such as latency and errors.
OpenShift	A hybrid cloud, enterprise Kubernetes platform that enables developers to build, deploy, and manage containerized applications efficiently.
OpenShift CI/CD process	A continuous integration and deployment process that automatically merges code changes, builds, tests, and deploys new versions to different environments.
Operators	Software extensions in Kubernetes that automate complex tasks and act as custom controllers to extend the Kubernetes API.
Operator Framework	A suite of tools and capabilities designed to enhance the development, testing, delivery, and updating of Operators in Kubernetes environments.
OperatorHub	A web console in OpenShift that allows cluster administrators to discover and install Operators. It offers a variety of Operators, including Red Hat Certified, community, and custom-defined Operators.
Operator Lifecycle Manager (OLM)	A component designed to handle the lifecycle management tasks such as installation, upgrades, and role-based access control (RBAC) for Operators within Kubernetes or OpenShift clusters.
Operator maturity model	A framework that defines stages of maturity for Operators, ranging from basic installation to fully automated operations (Auto Pilot).
Operator Pattern	A system design that links a Controller to one or more custom resources.
Operator Registry	A system that stores Custom Resource Definitions (CRDs), Cluster Service Versions (CSVs), and metadata about Operators and channels. It runs in Kubernetes or OpenShift clusters to provide the Operator catalog data to OLM.
Operator SDK	A toolkit that helps authors build, test, and package their Operators without requiring knowledge of Kubernetes API complexities.
postCommit	A section in a build configuration that defines an optional hook
Retries	A mechanism used in microservices to automatically reattempt a failed request to another service.
runPolicy	A field within a BuildConfig object that dictates how builds are executed. It controls whether builds run sequentially (default: Serial) or concurrently.
Service Broker	Provides a short-running process that cannot perform the consecutive day's operations such as upgrades, failover, or scaling.
Service Mesh	A layer dedicated to ensuring the security and reliability of service-to-service communication.It provides traffic management to control the flow of traffic between services, security to encrypt traffic between services, and observability of service behavior; so, you can troubleshoot and optimize applications.
Software operators	Try to capture the knowledge of human operators and automate the same processes.
Source-to-Image (S2i)	A tool used for building reproducible container images. It facilitates the process of creating ready-to-run container images by injecting application source code directly into a base container image.
Source strategy	The source strategy indicates how builds are executed, whether through handling source code directly (Source), utilizing containers (Docker), or custom configurations, optimizing deployment and environment management accordingly. Each approach offers unique benefits tailored to project needs.
Source type	The type of input that determines the primary input like a Git repository, an inline Dockerfile, or binary payloads.
Webhook	A trigger that sends a request to an OpenShift Container Platform API endpoint. Webhooks automate development flows so that builds can occur automatically as new code is developed.