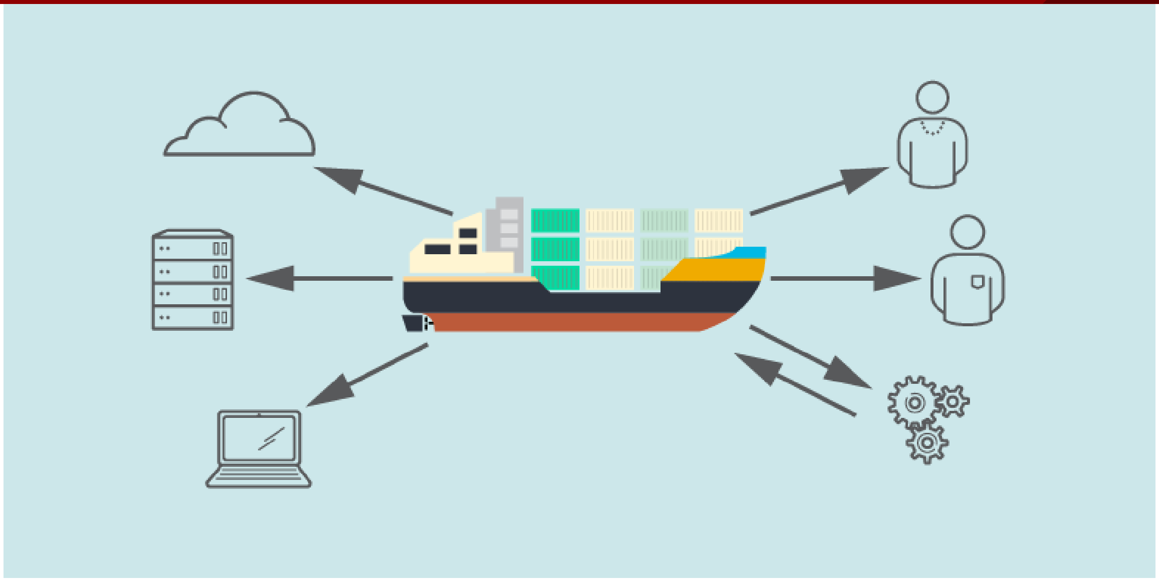
THE ENTERPRISERS PROJECT

Cheat sheet: What's the difference between a pod, a cluster, and a container?

Cheat sheet: What's the difference between a pod, a cluster, and a container?

By Kevin Casey



If you're discussing containers and Kubernetes, here's the breakdown on three crucial terms: Pod vs. cluster vs. container

The cloud-native ecosystem has generated a new jargon, and containerization and orchestration are central to the vocabulary.

We recently shared some plain-English definitions of orchestration and containers.

In both cases, the technologies these terms represent draw on the more universal meanings of the underlying words. A container is sort of like the Tupperware of software: It holds your application or service – and everything it needs to run – inside. (The maritime shipping container is another popular analogy.)

Similarly, if orchestration makes you picture the conductor who leads a musical ensemble, you're on the right path. A container orchestrator makes sure that all of the component pieces of a system "play" in the right place at the right time, and stop when they're no longer needed.

Then there's Kubernetes, the open source orchestration platform and all-around darling of the cloud-native world. It has a language of its own, too: Pods and nodes and clusters and secrets (what are they hiding?!) and kubelet and...well, there's a lot to parse through here.

Can we deduce similar connections to more mundane counterparts? Pod, like, a peapod? It's not totally off-target! (In fact, the Kubernetes documentation references the peapod, as well as a pod of whales, in defining the term.) And yes, a cluster represents a grouping or multiple of things – that's true in the Kubernetes and cloud-native lingo, too.

Definitions: The difference between a pod, a cluster, and a container

No matter the etymology, all of these terms can start blending together, especially if you're just learning the ropes of containers and container orchestration. We're here to help draw some clear lines between three fundamental terms: What's the difference between a pod, a cluster, and a container?

What is a container?

Let's start with the container: Again, the packaging and shipping comparisons exist for a reason: A software container is a means of packing up an application or service and everything required for it to run, regardless of environment, in a single place.

"A container by definition is a package with the program to execute and all its dependencies, such as the code, runtime, system libraries, et cetera, [all] bound together in a box," says Raghu Kishore Vempati, a Kubernetes practitioner and director of technology, research, and innovation at Altran.

> “
> A software container is a means of packing up an application or service and everything required for it to run, regardless of environment, in a single place.

It's also useful to have a high-level understanding of the relationship between containers and orchestration. The latter wouldn't exist without the former: Running containerized applications, especially in production, is what created the need for orchestration in the first place. There's not much need for a full-scale Kubernetes deployment if you're not running containerized applications.

"When Docker containers were first popularized, they mainly ran on a single computer – a developer's laptop," Portworx CTO Gou Rao told us recently. "But when it became clear that containers could be used instead of VMs to run applications, they started to run across many computers, and thus was born the need to manage many containers."

How does Kubernetes work with containers?

Enter container orchestration tools like Kubernetes. Rao notes that Kubernetes enables you to automate, manage, and schedule applications defined by individual containers – a necessary operational lever when you consider the possibility, if not likelihood (especially in a microservices architecture), that you might be running tens, hundreds, or even thousands of ephemeral containers as part of a complete application(s).

Doing that work manually isn't advisable for many (if any) teams; orchestration is what makes running and scaling containerized applications sustainable.

While containerization led to container orchestration, running Kubernetes now means some organizations begin running more containerized workloads because they have the infrastructure in place. What's the overall benefit? Time savings – leading to faster time to market of products and services – is one benefit that many executives seek. “A Kubernetes platform lets an enterprise take advantage of numerous cloud providers and grow as rapidly as you may need, without having to re-architect your infrastructure. This saves the need for significant deployment cycles and drastically improves your ability to provide new services as quickly as possible,” Ernest Jones, vice president, North America sales, partners & alliances for Red Hat, recently noted. Workload portability and security also top the list of benefits enterprises want from choosing Kubernetes.

What is a pod?

Where do pods and clusters come in? Let’s tackle pods first: They’re essentially a wrapper or housing for your individual containers when deploying them in Kubernetes. Containers for your containers, in a sense.

“A pod is a logical wrapper entity for a container to execute on a K8s cluster,” Vempati says. “Think about each pod as a transparent wrapper that would provide a slot for the container.”

Pods are the smallest deployable units in Kubernetes. As the official documentation puts it: “A pod (as in a pod of whales or pea pod) is a group of one or more containers, with shared storage/network resources, and a specification for how to run the containers.” So, in the simplest terms possible, a pod is the mechanism for how a container actually gets turned “on” in Kubernetes.

What is a cluster?

A cluster is central to the basic architecture of Kubernetes: If you’re running Kubernetes, you’re running at least one cluster. (Literally, there’s no such thing as a Kubernetes deployment without a cluster.) This cluster is sort of like a central nervous system for your application(s). Or, as Vempati explains it, it’s kind of like a motherboard or circuit board powering your applications: “A cluster is a board that provides the circuitry to run all the pods (which have the container instances in them) in an orchestrated manner as defined by the users,” Vempati says.

So there’s a symbiotic relationship between these terms:

Container > Pod > Cluster

Vempati walks through the progression of this relationship:

- “A container runs logically in a pod (though it also uses a container runtime);
- A group of pods, related or unrelated, run on a cluster. A pod is a unit of replication on a cluster;
- A cluster can contain many pods, related or unrelated [and] grouped under the tight logical borders called namespaces.”

There's another key concept, the node, which exists between the pod and cluster in this relationship. In Kubernetes, nodes are essentially the machines, whether physical or virtual, that host the pods. (Check out our article on Kubernetes architecture for beginners for more.)

This relationship also works in reverse, in the sense that there's not much point in running a Kubernetes cluster without containers or the pods that house them. They're distinct things, yet they each depend on each other.

More on containers and kubernetes:

- Kubernetes: 5 realities IT pros wish the CIO knew
- Getting started with Kubernetes: 5 misunderstandings, explained
- 3 reasons to use an enterprise Kubernetes platform
- O'Reilly: Kubernetes Operators: Automating the Container Orchestration Platform