THE
ENTERPRISERS
PROJECT

How to explain orchestration in plain English

Orchestration helps teams run containerized workloads at scale—while making hybrid cloud strategy more flexible. Here's how to explain container orchestration and orchestration platforms like Kubernetes to anyone in plain terms.

Orchestration might sound more musical than technical, but it's an important IT concept. Actually, teams running containers in production (or planning to soon) commonly find that "important" is an understatement. It's a must for running containerized workloads at any significant scale. It's also one of the key cloud-native tools that make flexible infrastructure strategies such as hybrid cloud manageable.

"Container orchestration makes complex, growing infrastructures much easier to manage by allowing both the application and the team to scale easily," says Zack Shoylev, senior developer advocate at Netdata.

What exactly is orchestration, though? How do you define it succinctly—and how do you explain it to others, including non-technical people? Just as the image of stacks of shipping containers is often used as an analogy for software containers, so too does the musical metaphor work here—think of the many instruments that play, off and on, in concert (quite literally) with one another to form a composition.

"Container orchestration makes complex, growing infrastructures much easier to manage."

"A container orchestrator is to containers as a conductor is to an orchestra," says Dave Egts, Chief Technologist, North America Public Sector, Red Hat. "In the same way a conductor would say how many trumpets are needed, which ones play first trumpet, and how loud each should play, a container orchestrator would say how many web server front end containers are needed, what they serve, and how many resources are to be dedicated to each one."

A conductor waving their baton can be a useful analogy, especially for non-technical people, and one which we'll flesh out even more below.

Let's start with some plain-terms definitions of orchestration that you can use with your team and elsewhere in your organization.

What is container orchestration?

Consider these five definitions:

"Whether we're talking about a single computer or a datacenter full of them, if every software component just selfishly did its own thing there would be chaos. Linux and other operating systems provide the foundation for corralling all this activity. Container orchestration builds on Linux to provide an additional level of coordination that combines individual containers into a cohesive whole."

Gordon Haff, technology evangelist, Red Hat

"Container orchestration is the next step up from using a single container to run a service. It's the management of multiple containers working together as part of an application infrastructure, using the same infrastructure-as-code approach used for containers. While containers deal almost entirely with how a single service is configured and deployed, container orchestration is about how multiple containers will work together by declaratively describing an immutable infrastructure configuration for containers (deployment and configuration), container networking (open ports, network configuration, load balancing, service discovery), and scaling rules (resource allocation, horizontal scaling, failover)."

Zack Shoylev, senior developer advocate, Netdata

"Container orchestration refers to the tools and platforms used to automate, manage, and schedule applications defined by individual containers. Container orchestration tools like Kubernetes help you run and manage all your containers in production and solve problems that may not have been present in the development stage on a single machine—for instance, updating applications composed of multiple containers running in production without downtime or even recovering from a complete data center outage. Container orchestration can be used in any environment where you use containers. It can help you to deploy the same application across different environments without needing to redesign it."

Gou Rao, CTO at Portworx

"Think about your laptop; you have a lot of applications. Unfortunately, [unlike with a user and their laptop], there is not [usually] one engineer watching one machine, i.e. the engineer-to-server ratio. They typically manage thousands of machines if not tens of thousands. So if a program stops running or something needs to get updated, containers make this easy. Our expectation as consumers is for rapid changes and if someone had to go reinstall software on 10,000 machines it would take forever. Thus the age of the container orchestrator. Part resource manager and part scheduler equal an orchestrator. A resource manager ensures that what you ask for is what you get. A scheduler is like your manager in a retail store—it makes sure there is enough staff (e.g enough nodes) and knows what to do if someone calls in sick (minimums and maximums)."

Ravi Lachhman, evangelist at Harness

"Container orchestration is the coordination of the where and how a containerized process runs in an environment."

Jonathan Katz, VP of platform engineering at Crunchy Data—(winner of our brevity prize.)

Why do we need container orchestration?

Containers are pretty nifty—and some will say very necessary—evolution in how software gets packaged, deployed, and operated. Once developers began embracing them, however, it became clear there was a need for tools that could keep everything running smoothly, especially if you wanted to run them somewhere other than a local box.

“When Docker containers were first popularized, they mainly ran on a single computer—a developer’s laptop,” says Rao from Portworx. “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.”

This can quickly turn into an operational behemoth, even for the largest of teams, and especially when the word “scale” is a mainstay in an organization’s vocabulary.

Shoylev from Netdata notes that scaling might not be an urgent need for a legacy application with a stable user base, but “containers and orchestration are indispensable tools for growing online applications.”

“The application is allowed to scale horizontally because multiple containers can be easily started or stopped, even for the same type of service,” Shoylev continues. “The container management framework would take care of the details, as long as the application is properly architected and configured. The team can also scale, because the configuration and orchestration of the application infrastructure is stored in configuration files that allow collaboration using common software development techniques, like source control, pull requests, and issue tracking.”

What's the difference between orchestration and automation?

You may get this question as you explain orchestration. Let's take the case of a continuously humming machine with multiple interacting parts. "In its simplest sense, *automation* is about each individual part performing the same repetitive steps over and over again," says E.G. Nadhan, Chief Architect and Strategist, North America (Commercial), for Red Hat. Think of a roller coaster that starts and takes a specific route up, down, and sideways and comes to a halt, he suggests. The roller coaster follows the same routine at predetermined intervals.

"Now, let us add some variables to this machinery where it has to adjust and function differently depending upon other factors, like the weather, the size of the workloads, the age bracket of the roller coaster riders, etc. Orchestration involves the dynamic adjustments to the environment that includes the introduction of additional parts, decommissioning of existing parts, increasing the frequency at which they execute, changing their operating thresholds and schedules. Orchestration ensures that the routines adjust themselves to these variances."

What's the difference between orchestration and Kubernetes?

Depending on whom you ask, the answer these days is likely to be: Nothing.

But a more specific answer: Kubernetes is a popular open source platform for container orchestration. It's not the only option for orchestration, but it has become the default choice.

"There are many options for container orchestration including Kubernetes and its various distributions," Rao says. "There are pros and cons with each, but Kubernetes' rising popularity and strong community support make it the clear king of the container orchestrators."

How to explain orchestration to non-techies

IT leaders and their teams might need to explain orchestration to others who have varying degrees of technical know-how, or perhaps very little at all. If you're making the case with the CFO's office during budget season, for example, it can help to have some useful comparisons at the ready if they need a reference point outside of software and infrastructure.

"If you aren't in IT, you may not have heard of Kubernetes, but it's being rapidly adopted in the tech world to automate all the backend functionality necessary to run large-scale consumer-facing apps like Uber and Lyft as well as business applications like banking or insurance," Rao says. "All these apps run on servers in the cloud and when one of those servers goes offline for any reason—a power failure, data center outage, etc.—a business can't let that result in downtime for their customer. All of these backend components need to be 'orchestrated' or managed in concert to keep apps online."

The "classical" comparison: A musical orchestra

Yes, as we noted earlier, the music metaphor works.

"A container orchestrator works similarly to how a conductor directs an orchestra: the conductor will cue a section of a band to start or stop playing and give directions such as how loud or quickly to play," says Katz from Crunchy Data. "Likewise, a container orchestrator will tell containerized processes on what nodes to execute and give guidance on what resources they are allowed to consume."

Rao extends to conductor-orchestration analogy: "Kubernetes coordinates, adjusts, and directs all the different sounds, patterns, and musicians that make up an application sitting on a server. Only if all the components are working together, does the music sound beautiful. Otherwise, you just have noise."

The auto manufacturing analogy

One of the other key concepts of container orchestration is that it enables teams to standardize processes and even language across heterogeneous, distributed environments and applications. Consider a manufacturing example:

"If your application is a car, then container orchestration is the factory design document for that car," Shoylev says. "It allows engineers to set up factory processes using clear, detailed, and descriptive documents that once reviewed and approved would allow factory workers to produce and test the exact same car again and again."

The well-known app

Lachhman likes to start with the "well-know app" reference point when describing containers and orchestration to someone with less technical know-how.

"Depending on how non-technical the folks are—e.g. my wise but not technically inclined mom – I always tend to start with an application or even a website: For example, 'Hey, you know Facebook or CNN.'"

Lachhman then points out that such apps and sites need to run behind-the-scenes on another computer to be able to properly display on your own devices, such as a laptop or phone. But just like the engineer-to-server ratio, this isn't a 1:1 deal.

"It is inefficient to have one computer to one end user—imagine if Facebook had a billion computers – so there are ways to subdivide and have more density," Lachhman says. "Hello, containers."

The hotel example

Housing provides another way for non-technical people to wrap their heads around the what and why of containers and orchestration. Lachhman compares the single-family home to a physical server: "It's designed for longevity and some level of uniqueness. If something is wrong with your home, you fix it."

If VMs were a way of densifying physical environments, then the apartment building is the next step in the analogy, Lachhman says.

Now think about a hotel and its purpose: For starters, the density typically increases even more, and most hotels are usually built for short-term use by their tenants.

"Containers are like hotel rooms," Lachhman says. If something's wrong with your room, you can ask for another one. Regardless, you're probably not staying there long.

But from the hotel operator's standpoint, there's a problem: "With such density, you need help filling all the rooms to make the hotel profitable," Lachhman says. "Imagine having a realtor representing a single hotel room – that would be overkill. This is the job of the container orchestrator."

Revisiting Lachhman's earlier definition, container orchestrators are part resource manager and part scheduler. So let's close by forwarding that to the hotel operator.

In terms of the resource manager, Lachhman equates that to matching you with the best possible hotel room – if you need two queen beds or one king bed, that happens automatically.

Meanwhile, "the scheduler might be making sure your hotel room always has a bottle of water and a certain number of pillows and if you run out of water or have more guests, more bottles of water appear," Lachhman says. "Having the ability to react quickly to make sure the hotel is humming is the point of a container orchestrator, the most prolific being Kubernetes."