What is Containers as a Service?

Learn what Containers as a Service are, when to use them, and how they enable faster application delivery.

BY IAN BUCHANAN

Browse topics

Continuous Delivery Principles

Continuous Delivery Pipeline 101

What is Continuous Integration

Software testing for continuous delivery

What Is Continuous Deployment?

Und
Microser
Architec

Overvie

3 Secrets to Buildin Microse

What are contain

What is Kubern

Contair vs Virtual Machin

• Contair as a Service

Bitbucke CI/CD tutorials

Continuc Delivery articles

Containers as a Service (CaaS) is a cloud-based service that allows software developers and IT departments to upload, organize, run, scale, and manage containers by using container-based virtualization.

A container is a package of software that includes all dependencies: code, runtime, configuration, and system libraries so that it can run on any host system. CaaS enables software teams to rapidly deploy and scale containerized applications to high availability cloud infrastructures. CaaS differs from Platform as a Service (PaaS) since it relies on the use of containers. PaaS is concerned with explicit 'language stack' deployments like Ruby on Rails, or Node.js, whereas CaaS can deploy multiple stacks per container.

What is CaaS?

CaaS is essentially automated hosting and deployment of containerized software packages. Without CaaS, software development teams need to deploy, manage, and monitor the underlying infrastructure that containers run on. This infrastructure is a collection of cloud machines and network routing systems that requires dedicated DevOps resources to oversee and manage.

CaaS enables development teams to think at the higher order container level instead of mucking around with lower infrastructure management. This brings development teams better clarity to the end product and allows for more agile development and higher value delivered to the customer.

CaaS versus PaaS

Platform as a service (PaaS) is concerned with and limited to code stack level infrastructure. When using PaaS, a project has no control over the underlying operating system. Container runtimes offer configuration and virtualization of the operating system, allowing for advanced customization and control. Containers can be critical to the development of highly customized and specialized software. Yet for more generic and standard software, PaaS is often the best choice.

PaaS is a cloud hosting paradigm that focuses on application-level code deployment. PaaS providers offer automated hosted environments that focus on higher level application infrastructure dependencies like language runtimes and databases. The "Platform" in PaaS is usually associated with a code language ecosystem, or a "stack". Some examples of popular PaaS "stacks" are Ruby on Rails, Node.js, .NET, and Java Spring MVC.

PaaS is typically better suited for monolithic application deployments, since it generally focuses on a single stack per deployment. CaaS can be a better fit for for micro services since each container deployed to the CaaS can have its own encapsulated operating system and language stack. PaaS still suffers from the "works on my machine" problem. There can be subtle differences between the development environment and the production environment of a PaaS system. A container system's primary goal is to avoid and ensure consistent behavior across underlying deployment environments.

Other Cloud Services

IaaS

Infrastructure as a Service (IaaS) is the foundation layer of cloud computing and enables teams to reserve and

provision remote computational resources. All other cloud, “as a service” paradigms depend on IaaS. Using IaaS, developers can provision and request access to a cloud computer instance from their hosting provider. This cloud computer instance can then be remotely accessed and configured to install custom software on.

SaaS

Software as a Service (SaaS) is a term for describing a business model class of hosted cloud product offerings. SaaS companies generally offer subscription- based billing models for access to hosted cloud software. This differs from traditional, unit- priced deliverable software business models. SaaS companies are built on other as a service infrastructure tools like PaaS and IaaS.

FaaS

Functions as a Service (FaaS) is the forefront of cloud computing offerings and is also called “Serverless”. FaaS enables developers to directly upload code functions and execute them without configuring or managing any underlying system infrastructure or dependencies. This enables teams to focus directly on their relevant business domain concerns and forego distraction with infrastructure management.

The Benefits of CaaS

Containers and CaaS make it much easier to deploy and compose distributed systems or micro service architectures. During development, a set of containers can manage different responsibilities or different code language ecosystems. The network protocol relationship between containers can be defined and committed for deployment to other environments. The promise of CaaS is that these defined and committed container architectures can be quickly deployed to cloud hosting.

To expand on this idea let’s explore an example. Imagine a hypothetical software system that is organized in a microservice architecture, where the services system are structured by business domain ownership. The domains of the services might be: payments, authentication, and shopping cart. Each one of these services has its own code base and are containerized. Using CaaS, these service containers can be instantly deployed to a live system.

Deploying containerized applications to a CaaS platform enables transparency into the performance of a system through tools like log aggregation and monitoring. CaaS also includes built in functionality for auto scaling and orchestration management. It enables teams to rapidly build high visibility and high availability distributed systems. In addition, CaaS increases team development velocity by enabling rapid deployments. Using containers insures a consistent deployment target while CaaS can lower engineering operating costs by reducing the DevOps resources needed to manage a deployment.

Summary

CaaS is a powerful modern hosting paradigm that requires familiarity with containers to utilize. CaaS can be extremely beneficial to highly agile software development teams. It can be a great aide in establishing continuous deployment on a project. You need not look far for a good CaaS, since most modern cloud hosting providers offer CaaS solutions at competitive prices.

Integration

Software testing for continuous delivery

What Is Continuous Deployment?

Microservices and Microservices Architecture

- Overview
- 3 Secrets to Building Microservices
- What are containers?
- What is Kubernetes?
- Containers vs Virtual Machines
- **Containers as a Service**

IAN BUCHANAN

While Ian has broad and deep experience with both Java and .NET, he's best known as a champion of agile methods in large enterprises. He's currently focused on the emerging DevOps culture and the tools for enabling better continuous integration, continuous delivery, and data analysis. During his career, he's successfully managed enterprise software development tools in all phases of their lifecycle. He has driven organization-wide process improvement with results of greater productivity, higher quality, and improved customer satisfaction. He has built multi-national teams that value self-

Bitbucket CI/CD tutorials

Continuous Delivery articles

direction and self-organization. When not speaking or coding, you can find Ian indulging his passions in parsers, meta-programming, and domain-specific languages. Follow Ian at @devpartisan.

ARTICLE

Containers

Learn what containers are, how they work, and the different container platforms available.

Read this article →

TUTORIAL

Containers vs Virtual Machines

Learn the key differences between containers and virtual machines, as well as popular providers for both.

Try this tutorial →

CI/CD Topics

Continuous Delivery
Continuous Integration
Continuous Deployment
Pipelines
Software Testing
Microservices
Tutorials

Sign up for more CI/CD articles and tutorials.

Email

email@example.com

Subscribe

Up Next
Bitbucket CI/CD tutorials →