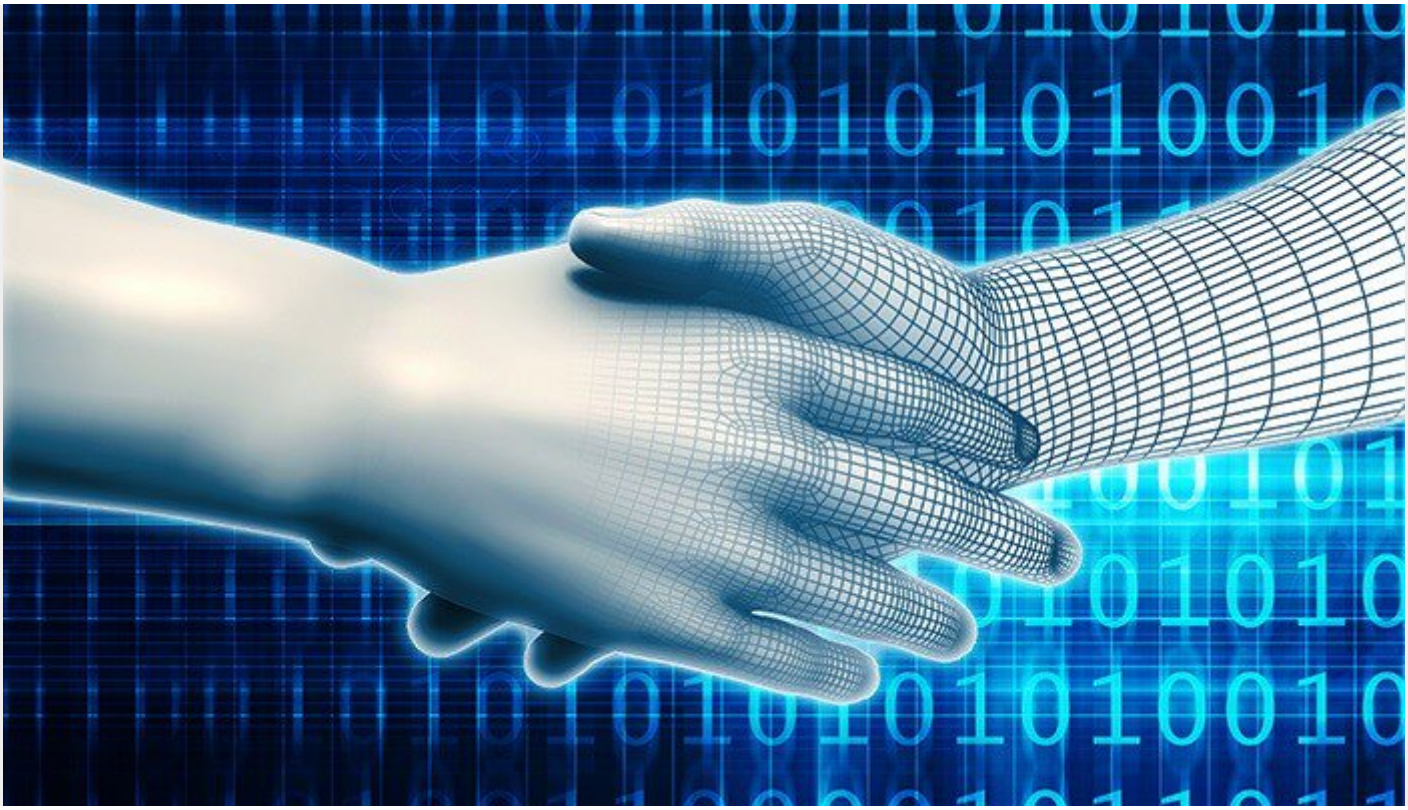
WHAT IS CLOUD NATIVE DEVOPS?



Based on the name, cloud native DevOps may seem to be the practice of running containerized application in the cloud, but this definition is misleading. Instead, cloud native DevOps is a method to structure your teams to take advantage of the automation and scalability that cloud native technologies like containers and Kubernetes offer—so you can increase the velocity of your business.

In this article, we'll take a look at cloud native DevOps.

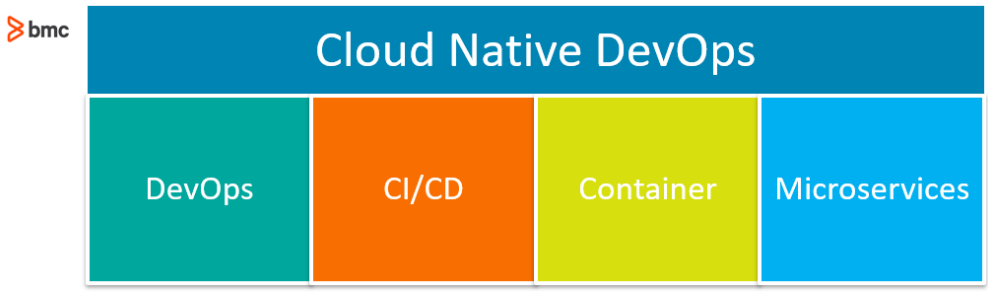
Understanding CNAs and DevOps

To properly define or explain cloud native DevOps, we must first understand cloud native apps (CNAs) and DevOps.

- CNAs are applications built for resiliency, agility, operability, and observability in mind.
- DevOps is a practice of operations and dev engineers working together throughout the entire lifecycle.

Based on both explanations, we can see that nothing is actually related to the cloud; they are simply principles and methodology followed while working on a set of services or applications. Therefore, we can define cloud native DevOps as a set of practices that involves continuous improvement, automation, cross-functional teams, and better alignment with business needs with customer expectations in mind. These principles apply to people, tools, culture, and process not where the

actual application lives (cloud or on-prem).



At its core, Cloud Native DevOps is a way to increase the velocity of your business and a method to structure your teams to take advantage of the automation and scalability that cloud native technologies like containers and Kubernetes offer. By nature, these cloud native technologies are designed to be:

- **Resilient.** Embracing failures instead of trying to prevent them, taking advantage of the dynamic nature of running on a platform.
- **Agile.** Allowing for fast deployments and quick iterations.
- **Operable.** Adding control of application life cycles from inside the application instead of relying on external processes and monitors.
- **Observable**. Providing information to answer questions about the application state.

Changes needed to implement cloud native DevOps

To properly implement cloud native DevOps, changes must happen in three key areas:

1. **Cultural change from silos to proper DevOps.** As mentioned above, it is not necessary to run applications in the cloud in order to be cloud native, but DevOps is a must in order to practice cloud native. The goal of DevOps is to align everyone with the same tools and a common set of priorities.
2. **Organizational change involving buy-in from everyone to work in collaboration to achieve the same goal.** The idea is to encourage faster feedback loop between developers and end users which in turn speeds up application development and provides action items for the business.
3. **Technical change which relates to the way the application is built.** For example, moving from monolith to microservices.

Ways to implement cloud native DevOps

Implementing CNAs is not as straightforward as deploying into the cloud. To be considered cloud native, a CNA needs to meet certain characteristics:

- **Aligning with the microservices patterns.** Monolithic apps should be broken into small services that can be developed independently. As long as each service adheres to a strong contract, it can be iterated on. All these services comprise the application.
- **Using containerization.** Code can be packaged without worrying about the underlying system.
- **Following declarative communication pattern.** CNAs must trust that the network will deliver the message and that it will return either a success or failure. This helps standardize a communication model, moving the functional implementation of how something achieves a desired state away from the application to a remote API or service endpoint.
- **Deploying container orchestration.** Perhaps the biggest orchestration platform out is

Kubernetes and for good reason. The biggest benefit of k8s is the fact that it abstracts away the details of underlying compute, storage and networking resources.
- **Writing code according to 12-factor application principles.** This ensures clean, declarative contracts for cloud platform deployments.
- **Increasing automation in CI/CD pipelines.** Continuous integration and deployment are nothing new to cloud native but the added complexity they bring mean there must be extra automation in place to deal with the complexity of the pipelines.
- **Exposing health check.** This is great for knowing what is going on with the application. The application is telling the platform it is running on what state it is in which in turn make monitoring easier.
- **Collecting telemetry data.** Things like latency, request per minute, etc. are information that is needed to determine whether you are meeting service level objectives (SLO). Telemetry data can and should be alerted on to consider your application cloud native.

Of course, cloud native DevOps is no silver bullet—it's just as important to be aware of the drawbacks as the benefits. Still, for companies looking to speed up automation and customize production to better serve customers, cloud native DevOps may be a useful tool.