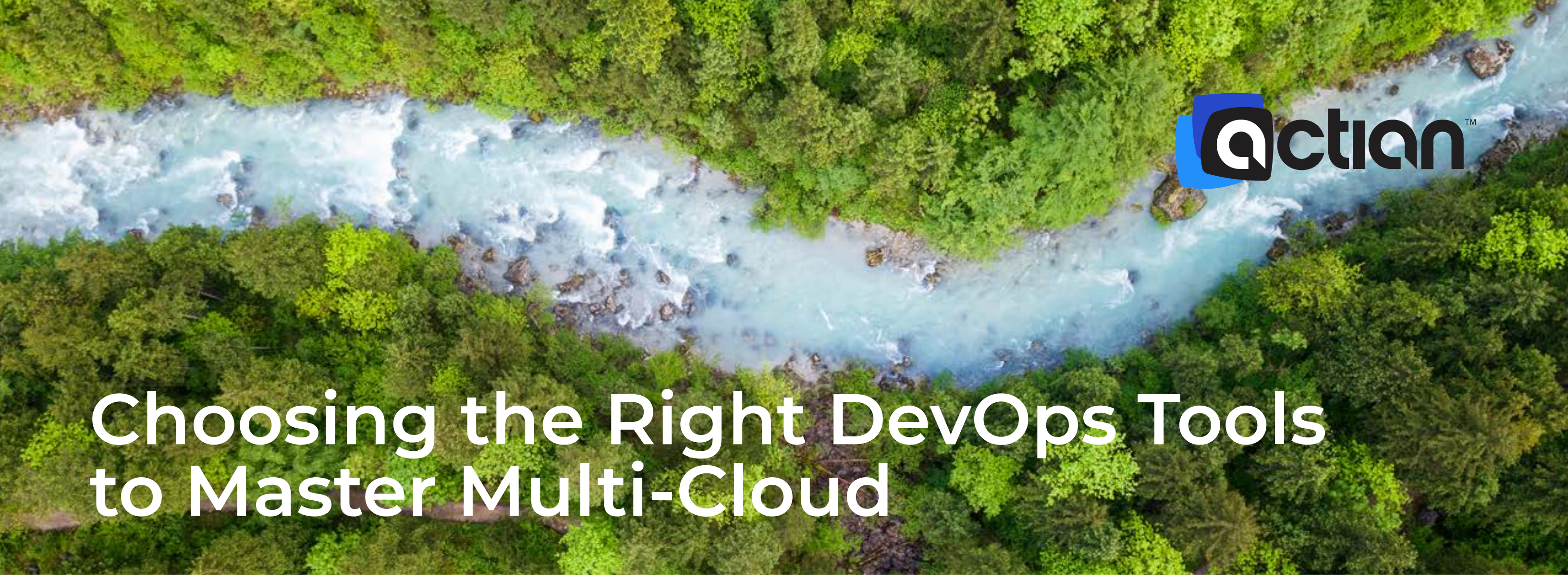


Choosing the Right DevOps Tools to Master Multi-Cloud

By: Oliver Fasterling,
Principal DevOps Engineer, Actian

Need More information?

Visit www.actian.com for product, service, and solution information, papers, success stories and much more.

White Paper

As we embrace the cloud and the concept of delivering a subscription-based solution (PaaS/SaaS) to our customers, we are faced with numerous challenges regarding our ability to bring these solutions to market.

In most cases, software development projects are allotted a considerable amount of resources dedicated to Research and Development. The priorities are usually the same: The need to develop a new product, with great features, top performance, reliability, security, affordability, competitiveness, and all within the context of a compelling use-case, which is meant to attract more customers and ultimately impact our bottom-line. All of this is clear. To deliver a successful SaaS offering, however, – which in-turn results in a continuous revenue stream from satisfied customers, – we must pay close attention to how we will build and support such a platform in a sustainable way.

A few questions arise: How will we release new features and deliver patches? How do we respond to unexpected outages? How can we provide a resilient bridge between our developers and the operators who will be responsible for running the show? What are our available resources, both from a head-count and budget perspective? The answers reside in having a strong DevOps practice and culture, and choosing the right tools for the task at hand.

Back in the day, we hosted everything "on-premise." We purchased hardware, installed it, configured it and relied on the IT department to keep the machines up and running. Times are evolving and we are now faced with the undeniable fact that on-premise solutions are becoming less and less attractive to our customers. Instead, they favor cloud-based solutions, to which they can have immediate access and without having to worry about hardware upkeep. Now, it's important to recognize on-premise solutions will not disappear entirely. There are still customers who won't fully embrace the cloud, for a variety of reasons. Overall, however, on-premise solutions have become products for a niche market. This article will focus on the trend regarding cloud adoption and development of cloud-based solutions.

We have a consensus that embracing the cloud is the future. That in itself presents new questions, however, and ultimately a variety of challenges to consider:

- Which cloud platform should we pick?
  - AWS?
  - Azure?
  - GCP?
  - Others?
- Which compute/virtualization platform should we use?
  - Standalone virtual instances, e.g., EC2, Azure VMs
  - VMWare on cloud
  - Containers, e.g., Docker/Kubernetes
  - Serverless/FaaS (Functions-as-a-Service), e.g., AWS Lambda, Azure Functions
- Which cloud computing service (XaaS) should we use? What are the differences between them?
  - IaaS (Infrastructure-as-a-Service)
  - PaaS (Platform-as-a-Service)
  - SaaS (Software-as-a-Service)
- What is our Continuous Integration and Continuous Delivery (CI/CD) strategy?
  - How often will we build and release new patches and major features?
    - Weekly, monthly or Agile (Sprint-based release schedule)?
  - Which tools should we use for integration? Which tools for delivery? An all-inclusive tool that combines everything?
  - How will we provision our infrastructure in a sustainable way?
    - CloudFormation templates/Azure Resource Manager
    - Terraform/Ansible/Chef/PowerShell
  - Which type of third-party applications are we allowed to consume?
    - Free/Open source/community-based support
    - Closed source/paid support

From a holistic viewpoint, any XaaS offering must consider all of the above items – clearly a non-inclusive list – to be successful. To help answering some of the questions above, we must leverage the right DevOps tools for the right task. First, however, we must understand the meaning of DevOps, an often-misrepresented term within the corporate culture.

What is DevOps?

Among the various definitions of "DevOps," the one which I consider to be the most accurate is:

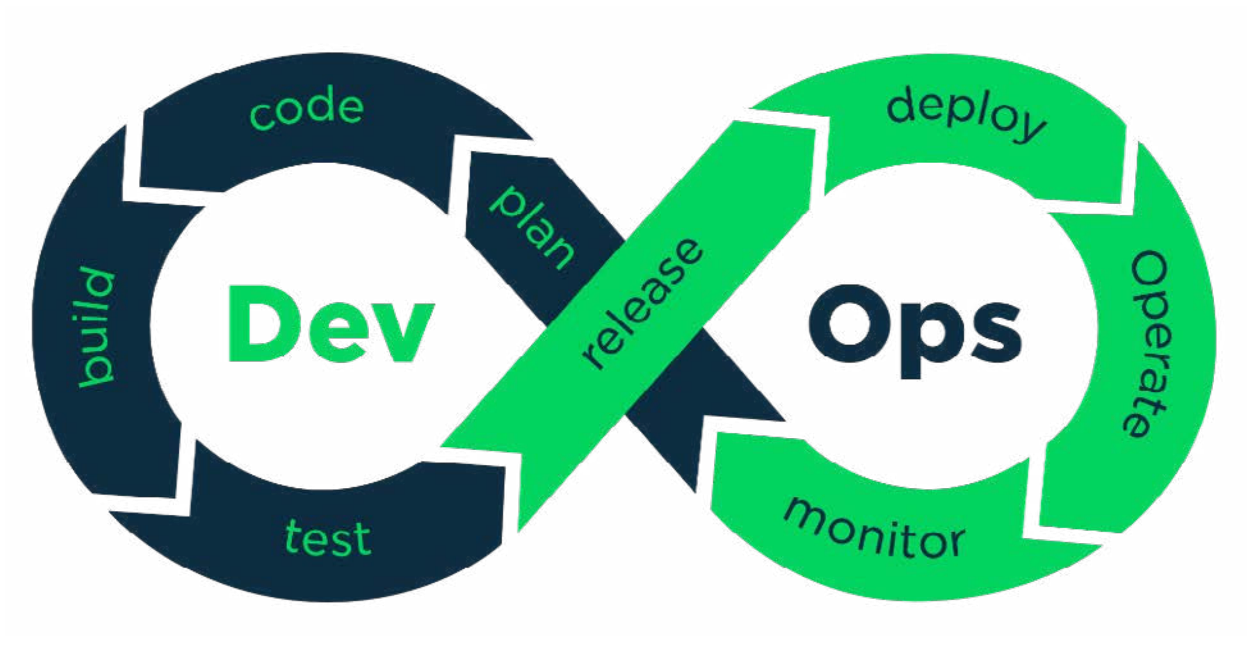
"DevOps is the practice of operations and development engineers participating together in the entire service lifecycle, from design through the development process to production support."

Ernest Mueller, rev. 12 Jan 2019,
https://theagileadmin.com/what-is-devops/

The definition is simple and to the point. It brings people together from two distinct disciplines, development and operations, into one common practice.

A more thorough definition of DevOps is one which incorporates the 8 practices of SDLC (Software Delivery Life Cycle) in a continuous, infinite loop (a.k.a. "Moebius Loop"):

1) Planning, 2) Coding, 3) Building, 4) Testing, 5) Releasing, 6) Deploying, 7) Operating and 8) Monitoring



Source: https://cdn-images-1.medium.com/max/2600/1*EBXc9eJ1YRFLtkNI_djaAw.png

Throughout my journey in the DevOps space, I have often seen a misunderstood application of DevOps practices within the corporate structure. Many software projects are allotted a team of "developers" on one side and "operators" on the other. These teams are often part of two separate reporting structures, which makes the division even more noticeable.

This is a major hindrance to an effective software development strategy and an exact contradiction of the DevOps philosophy. The true DevOps discipline was created from the notion that two distinct teams couldn't work well together because of the inherent discrepancies that existed in skills and responsibilities. Indeed, DevOps is a hybrid solution, which incorporates a healthy mix of both "developer" and "operator" mindsets, with the purpose of bringing teams together with the ultimate goal of delivering software with greater speed and quality.

A DevOps resource should be part of a common corporate reporting structure, which includes both development and operations. In summary, DevOps resources must be present throughout the entire software development life cycle, interchanging roles as developers, architects, testers and operators, and ready to wear multiple hats at any given time. Corporate leadership must grasp this basic concept to allocate resources effectively.

Separation of Duties

While it is important to recognize the need for a streamlined and unified DevOps practice, we also cannot ignore the risks associated with access to customer data and a lack of a proper change management process. Letting one person (or a team of people) have too much access to the available resources is unacceptable, and clearly a poor implementation of the proper checks and balances that are required to be compliant with external regulatory bodies. Having proper strategy involving “Separation of Duties” (SoD) is essential to provide our customers with an enterprise-grade and secure solution.

The challenge, however, is not having a DevOps team and SoD policy in place, but how to make both concepts work together in a seamless fashion. The article reference on the right provides an honest approach about how to overcome this challenge.

“DevOps and Segregation of Duties”
by Jeehad Jebeile, 19 Nov 2018
https://medium.com/@jeehad.jebeile/devops-and-segregation-of-duties-9c1a1bea022e

The article describes that certain industries, such as Finance or Healthcare, – which are highly regulated, – require a strict implementation of SoD. It is possible, however, to achieve compliance without creating a major impact to an integrated DevOps practice.

One size does not fit all

There is a novel idea regarding the ability to design an entire solution stack, which can be portable and reusable with any cloud provider. We refer to this as designing a solution with a “cloud-agnostic” mindset. The idea seems great, but it is actually not possible to be truly agnostic. If we pick a particular cloud provider, then we are inherently forced to use that provider’s service ecosystem.

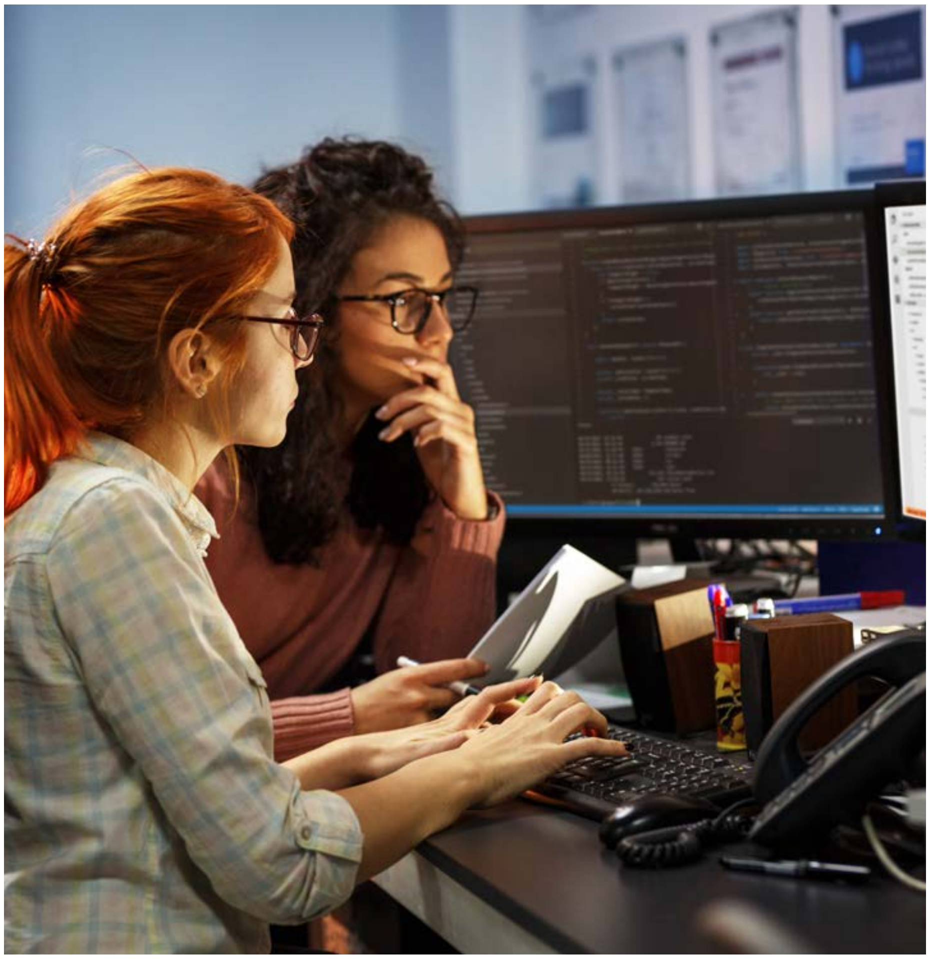
Yes, we can containerize our application, which can give us a significant amount of freedom and portability. However, putting the different pieces together (compute, network, storage, logging, security, etc.), requires the solution to use specific services only available from the current provider. If we have a directive to build our product on a specific provider, then it will be undoubtedly a custom-solution for that provider. In addition, if we need to replicate that effort on a second or third provider, then we’ll take as much of the portability as we can, – such as containerized applications –, but the rest will require to be re-done to match that new provider.

Conversely, it’s just as important not to fall into the trap of using every service that is made available from a particular provider. A great example is Amazon ECS vs. Kubernetes, where both offer containerization orchestration, but as evidence has shown, Kubernetes is revealed as being a superior choice due to its portability and ease-of-use across different cloud platforms.

Containerization vs. PaaS vs. FaaS

The concept of creating reusable containers, such as Docker, is undoubtedly one of the most important elements when designing a cloud solution. Containers are without a question a "must-have" in our architecture. Containers create portability and portability is good. We must understand, however, there are limitations with containers. Containers are great for bundling an application, which can be difficult to assemble in more than one platform. We cannot, however, containerize everything. Storage must be separate and isolated. Networking cannot be fully containerized, as we have other services that exist outside of the container infrastructure. Also, we often have the need to leverage separate PaaS and FaaS services for minor workloads, such as databases (RDS/SQL Azure) and serverless (AWS Lambda/Azure Functions), where it would be counterproductive to create our own container for such small tasks.

In summary, for closed-source applications, such as our own portfolio of Actian databases, containerization makes perfect sense. Clearly, however, not everything else may apply. We should not dismiss the use of readily available PaaS offerings because we fear the notion of becoming "vendor-locked." Often, the pursuit of containerizing everything for the sake of portability in a multi-cloud stance is counterproductive and overkill, and likely requires a significant amount of time and energy before such portability can be achieved, if it can be achieved at all.



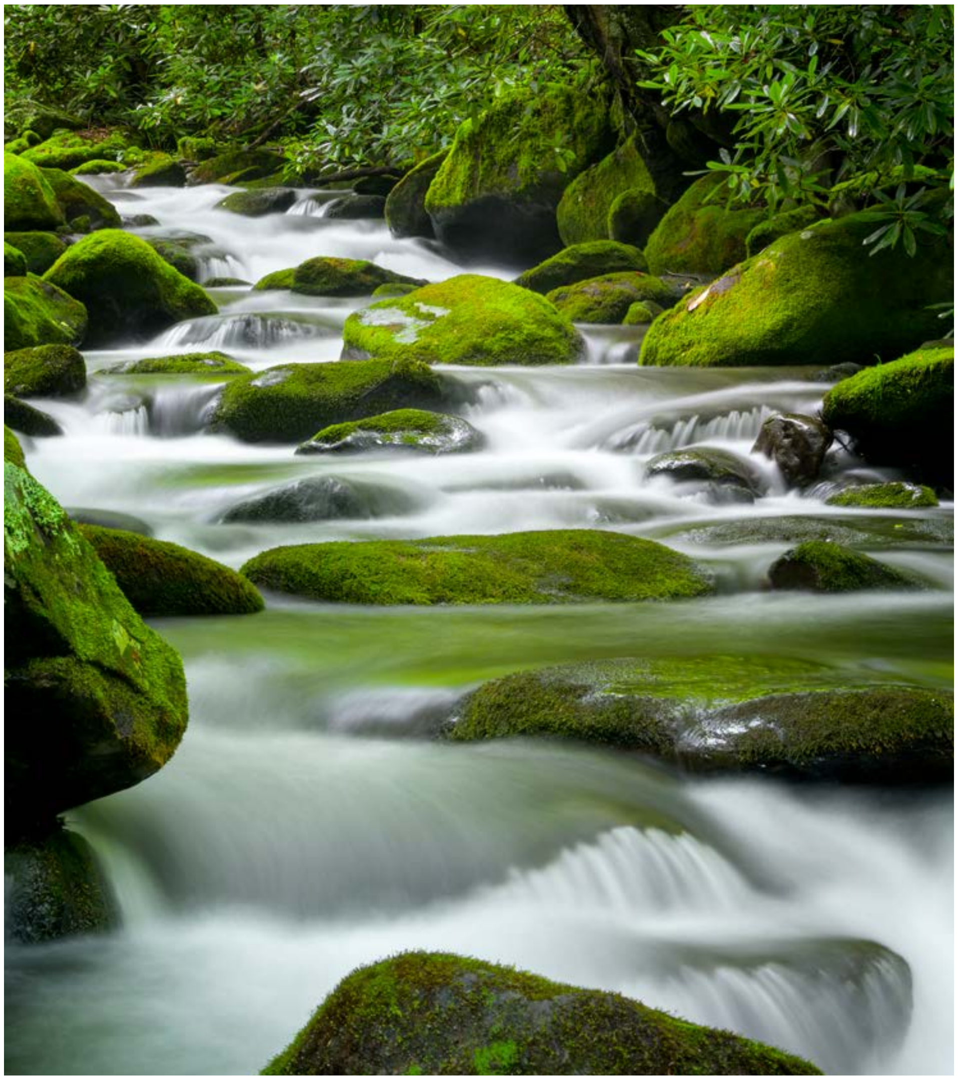
Developing custom CI/CD tools

The concept of developing a "cloud-agnostic" solution is also challenging when we deal with our continuous integration and continuous delivery strategy. Very often, our build and deployment tools make use of specific API calls, which will only work for a specific cloud provider. Some of these tools may only have plugins for one cloud provider, but not for another.
In such cases, we are forced to use a specific toolset for a specific provider.
In addition, how can we achieve a true cloud-agnostic, multi-cloud CI/CD strategy? This is a very good question and with no easy answers. There are third-party solutions available that advertise all-inclusive, multi-cloud, CI/CD capabilities. They are either too difficult to use with limited support (open source), or prohibitively expensive and becomes a non-starter. Alas, the research for the "right" solution may involve a combination of readily available apps and plugins (e.g., Jenkins), while also the employment of custom-made scripts, which are specifically customized for the cloud platform in question.

Open-Source/Cloud-Source tools

There is always a debate about whether we should use open-source tools because of their perceived “free” value. This perception is very short-sighted. Yes, they are free to use and, in most cases, the accompanying licensing is a non-issue. In many cases, however, it costs a considerable amount of time and effort to utilize an open-source solution that works effectively. The learning curve is often steep, and we have very limited human resources (DevOps engineers) available at our disposal. In our DevOps practice, moreover, we are faced with a never-ending backlog of tasks and automation features we hope to produce. Yes, we strive to leverage open-source tools to the best of our ability.

Surely, we will grab those tools the DevOps community have proven to be clear leaders. There are instances, however, when the ROI generated from developing custom tooling with open-source technology becomes drastically diminished, to the point where it may often result in a complete wasted effort. In such instances, we must be open-minded and realize it is OK to pay a monthly fee to a third-party provider if it means we can bring our solution to market at a fraction of the time compared to having it developed in-house. Sometimes, we must accept we cannot do everything ourselves. There are expert players available who have already conquered many of the obstacles we are facing. Why not leverage that knowledge and put our resources to use on more pressing matters?



Conclusion

If we are to deliver an enterprise-grade solution to our customers, then we must take a step back and consider how all of the moving parts fit together. We must come to terms with the right definition of DevOps, and consequently create a robust and unified DevOps practice, while keeping the right checks and balances with an effective SoD strategy. Yes, let’s leverage containerization and reusable technology where appropriate. Let’s not attempt to design a single big black box, however, that can be plugged and expected to run everywhere, as it will never work. Let’s use readily available services, such as PaaS and FaaS, which can be immediately consumed and have negligible cost, even if they are specific to a given cloud provider. Yes, let’s leverage popular open-source tools that match our particular needs. Let’s also come to terms with our limited human resource count and our ability to produce results in a timely fashion. We may often need help from a third-party that can resolve a major hurdle in our development process and free valuable resources, so they can be spent on more important matters.