The concept of Infrastructure as Code (IaC) and its benefits in the context of DevOps.

What is Infrastructure as Code?

Infrastructure as code (IaC) is the ability to provision and support your computing infrastructure using code instead of manual processes and settings. Any application environment requires many infrastructure components like operating systems, database connections, and storage. Developers have to regularly set up, update, and maintain the infrastructure to develop, test, and deploy applications.

Manual infrastructure management is time-consuming and prone to error, especially when you manage applications at scale. Infrastructure as code lets you define your infrastructure's desired state without including all the steps to get to that state. It automates infrastructure management so developers can focus on building and improving applications instead of managing environments. Organizations use infrastructure as code to control costs, reduce risks, and respond with speed to new business opportunities.

What are the benefits of infrastructure as code?

Automation is a key goal across any computing environment. Infrastructure as code (IaC) is used for infrastructure automation to create environments. The most common use of IaC is in software development to build, test, and deploy applications.

Traditionally, system administrators used a combination of scripts and manual processes to set up infrastructure environments. The process was complex and time-consuming. Today, you can use IaC to automatically set up your environment within minutes and manage it more efficiently. We give some benefits next.

1. **Easily duplicate an environment**

The same environment can be deployed on a different system in another location using the same IaC, as long as the infrastructure resources are available.

For example, imagine a business’s regional branch has IaC to describe the whole branch’s enterprise environment, including servers, networking, and custom configurations. If the business opened another regional branch, they could use IaC to duplicate the exact same environment and quickly make the branch online and

operational. IaC removes the repetitive manual steps and checklists that were needed in the past.

2. Reduce configuration errors

Manual configuration is error-prone due to human involvement. People make mistakes. Or there could be configuration drift due to changes in one setup (like a developer environment) that was missed in another setup (like a test environment).

In contrast, IaC reduces errors and streamlines error checking. If there are errors due to IaC code updates, you can quickly fix the situation by rolling the codebase to the last known stable configuration files. It's also possible to roll back environments using previous versions of IaC configuration files for other reasons, such as the deployment of older application versions.

3. Iterate on best-practice environments

Source control allows software developers to easily build and branch on environments. For instance, imagine that an application grew to include an optional machine learning module. A developer could branch the application's IaC to initiate, use, and stop a server instance.

How does infrastructure as code work?

Much like software code describes an application and how it works, infrastructure as code (IaC) describes a system architecture and how it works. An infrastructure architecture contains resources such as servers, networking, operating systems, and storage. IaC controls virtualized resources by treating configuration files like source code files. You can use it to manage infrastructure in a codified, repeatable way.

IaC configuration management tools use different language specifications. You can develop IaC similar to application code in Python or Java. You also write the IaC in an integrated development environment (IDE) with built-in error checking. And you can maintain it under source control with commits at each code change. IaC files are included as part of the wider codebase.

Approaches to IaC

There are two different approaches to infrastructure as code.

Declarative

Declarative IaC allows a developer to describe resources and settings that make up the end state of a desired system. The IaC solution then creates this system from the infrastructure code. This makes declarative IaC simple to use, as long as the developer knows which components and settings they need to run their application.

Imperative

Imperative IaC allows a developer to describe all the steps to set up the resources and get to the desired system and running state. While it isn't as simple to write imperative IaC as declarative IaC, the imperative approach becomes necessary in complex infrastructure deployments. This is especially true when the order of events is critical.

What is the role of IaC in DevOps?

DevOps is the process of improving collaboration between software development and IT operations teams. It aims to shorten the application development lifecycle and provide continuous delivery of high-quality software. DevOps teams integrate operations activities with developer tools and code commits, so applications can have extremely rapid release cycles.

A key goal of DevOps is to automate infrastructure tasks across the development process. You can integrate infrastructure as code (IaC) into continuous integration and continuous deployment (CI/CD) pipelines. This way, when software goes through its build and release process, the necessary infrastructure changes can be made in tandem.

DevOps teams use infrastructure as code for many purposes:

- Quickly set up complete environments, from development to production
- Help ensure consistently reproducible configurations between environments
- Integrate seamlessly with cloud providers and efficiently scale infrastructure resources up or down based on demand

IaC provides a common language for both developers and operations. Changes can be reviewed in a transparent manner, which fosters better collaboration in a DevOps environment.