UNIT – 5

Developing for Multi-Cloud with DevOps and DevSecOps

Introducing DevOps and CI/CD

Every organizations work in DevOps cycles, using release cycles for applications with continuous development and the possibility to test, debug, and deploy code multiple times per week, or even per day, so that these applications are constantly improved.

That is what CI/CD pipelines are for: **Continuous Integration and Continuous Delivery and Deployment.**

DevOps Agile Skills Association (DASA) defines the DevOps framework based on six principles

1. Customer-centric action
2. Create with the end in mind
3. Cross-functional autonomous teams
4. Continuous improvement
5. Automate as much as possible

Introducing DevOps and CI/CD

DevOps Agile Skills Association (DASA) defines the DevOps framework based on six principles

1. Customer-centric action: Develop an application with the customer in mind: what do they need and what does the customer expect in terms of functionality? This is also the goal of another concept, domain-driven design, which contains good practices for designing.

2. Create with the end in mind: How will the application look when it's completely finished?

3. End-to-end responsibility: Teams need to be motivated and enabled to take responsibility from the start to the finish of the application life cycle. This results in mottos such as you build it, you run it and you break it, you fix it. One more to add is you destroy it, you rebuild it better.

4. Cross-functional autonomous teams: Teams need to be able and allowed to make decisions themselves in the development process.

DevOps Agile Skills Association (DASA) defines the DevOps framework based on six principles

5. Continuous improvement: This must be the goal—to constantly improve the application. But DevOps applies to more than just the application: it's also about the processes, the people, and the tools. DevOps, at its core, is a culture, a mindset.

6. Automate as much as possible: The only way to really gain speed in delivery and deployment is by automating as much as possible. Automation also limits the occurrence of failures, such as misconfigurations.

Introducing DevOps and CI/CD

DevOps cycle with CI/CD

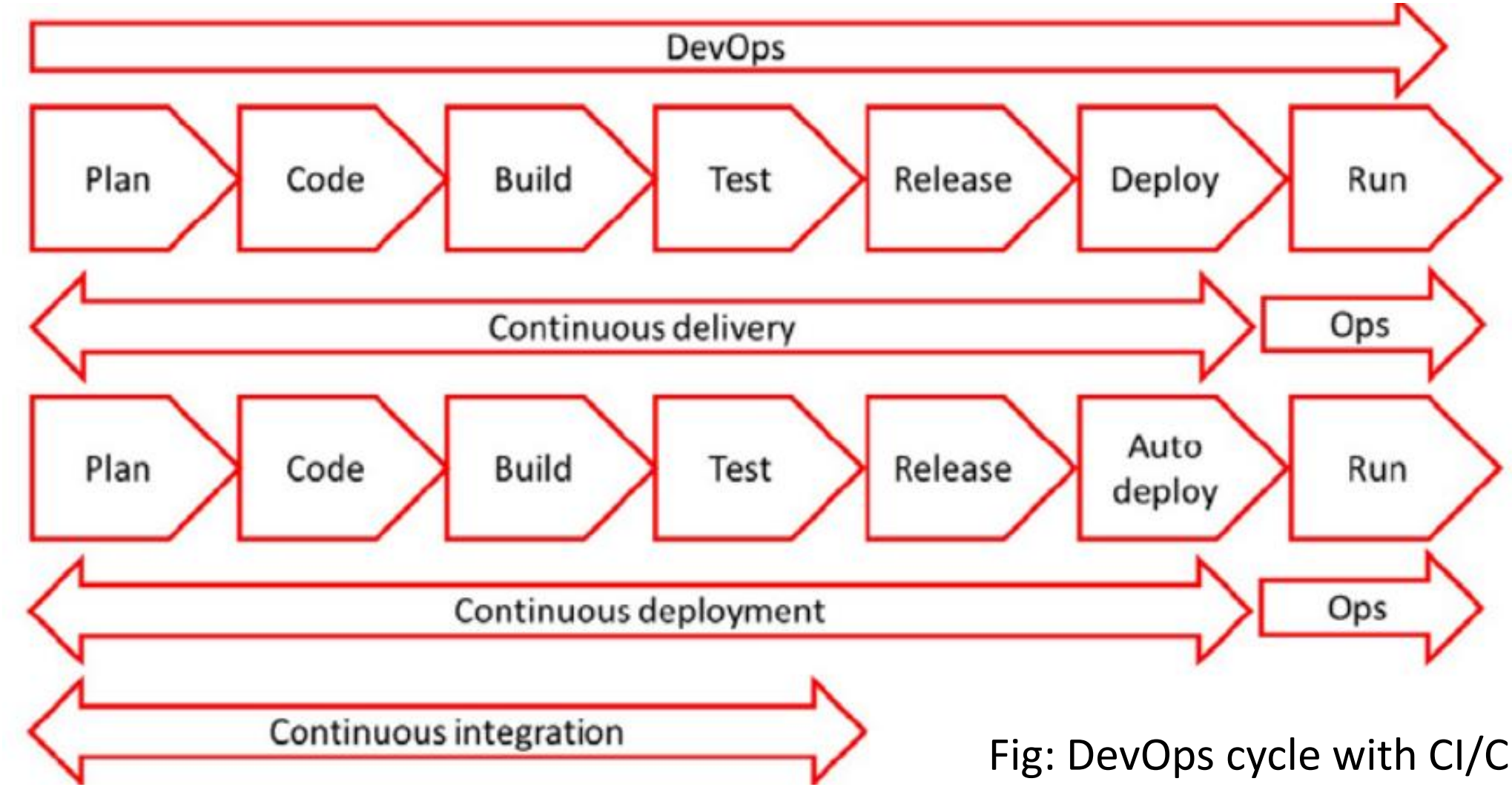


Fig: DevOps cycle with CI/CD

Introducing DevOps and CI/CD

DevOps cycle with CI/CD

- **Continuous Integration (CI)**
  - CI is built on the principle of a shared repository, where code is frequently updated and shared across teams that work in the cloud environment.
  - CI allows developers to work together on the same code at the same time.
  - The changes in the code are directly integrated and ready to be fully tested in different test environments.

- **Continuous Delivery and Deployment (CD)**
  - CD focuses on the automated transfer of software to test environments.
  - The ultimate goal of CD is to bring software to production in a fully automated way.
  - Various tests are performed automatically. After deployment, developers immediately receive feedback on the functionality of the code.

DevOps cycle with CI/CD

We have to make a note here: continuous delivery and continuous deployment are not the same thing.

The following are some key points to differentiate both practices:

- Continuous delivery usually takes time from development to production; weeks or days in the best case.
- The kinds of changes applied under continuous deployment can go to production several times per day

Introducing DevOps and CI/CD

Getting started with CI/CD

- CI/CD is widely adopted by enterprises, but a lot of projects fail. This section explains how enterprises can successfully implement CI/CD and how they can avoid pitfalls.
- The major takeaway should be that an implementation starts with consistency. That counts for cloud implementations as well as for CI/CD.

To get to a successful implementation of DevOps, an organization is advised to follow these steps:

1. One of the key principles in DevOps is autonomous teams that take end-to-end responsibility.
2. Choose the CI/CD system. Decide on the CI/CD system and ensure all teams work with that system. Again, it's about consistency.
3. It's advised to perform a proof of concept. **Generic Test Agreement** (GTA) describes what and how tests must be executed before systems are pushed to production.
4. Automate as much as possible. This means that enterprises will have to adopt working in code, including **Infrastructure as Code (IaC)**

Introducing DevOps and CI/CD

Conceptual diagram of a build and release pipeline

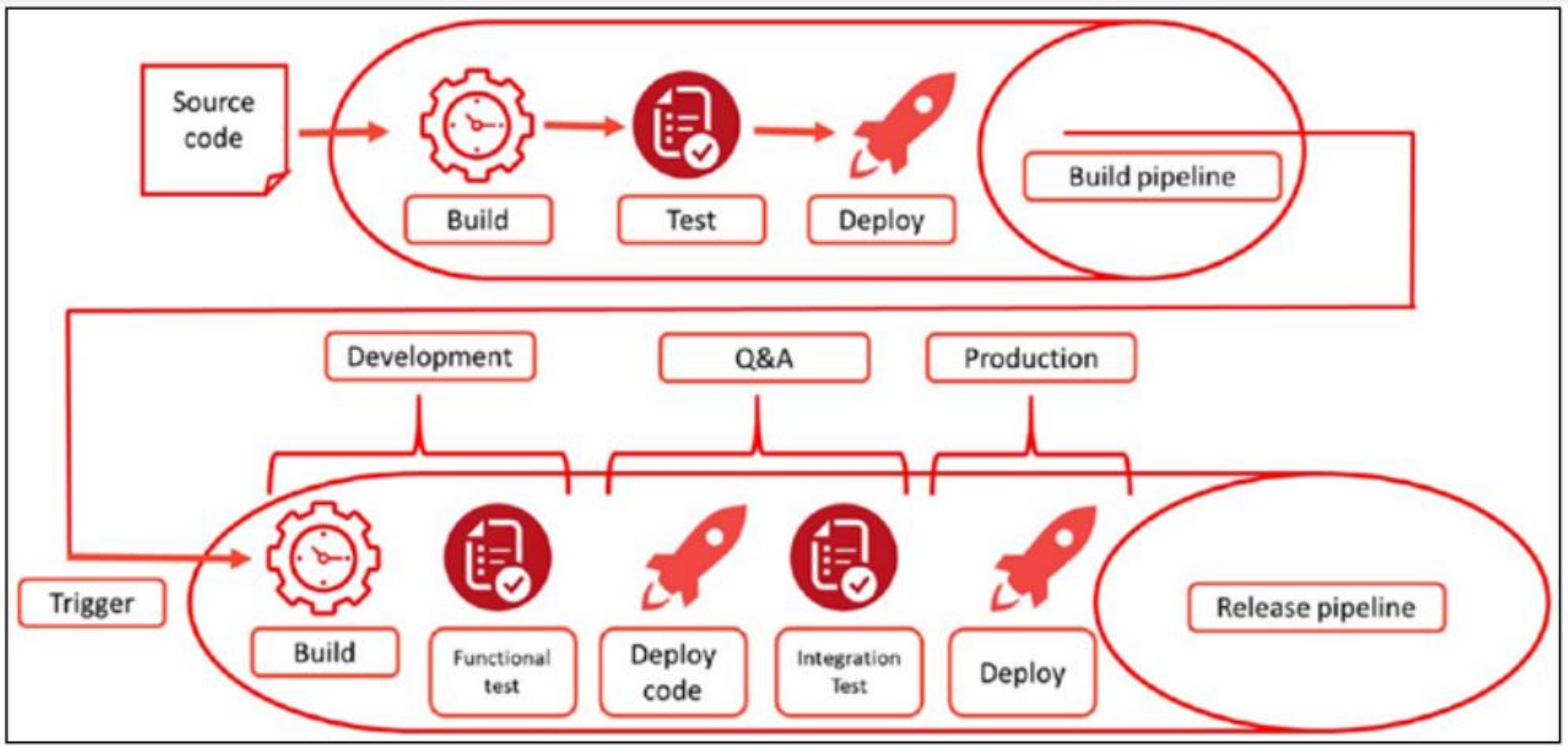


Figure: Conceptual diagram of a build and release pipeline

Conceptual diagram of a build and release pipeline

The Figure shows the concept of implementing a build and release pipeline with various test stages.

- The code is developed in the build pipeline and then sent to a release pipeline where the code is configured and released for production.
- During the release stages, the full build is tested in a test or **Quality and Assurance (Q&A)** Assurance environment.
- In Q&A, the build is accepted and released for deployment into production

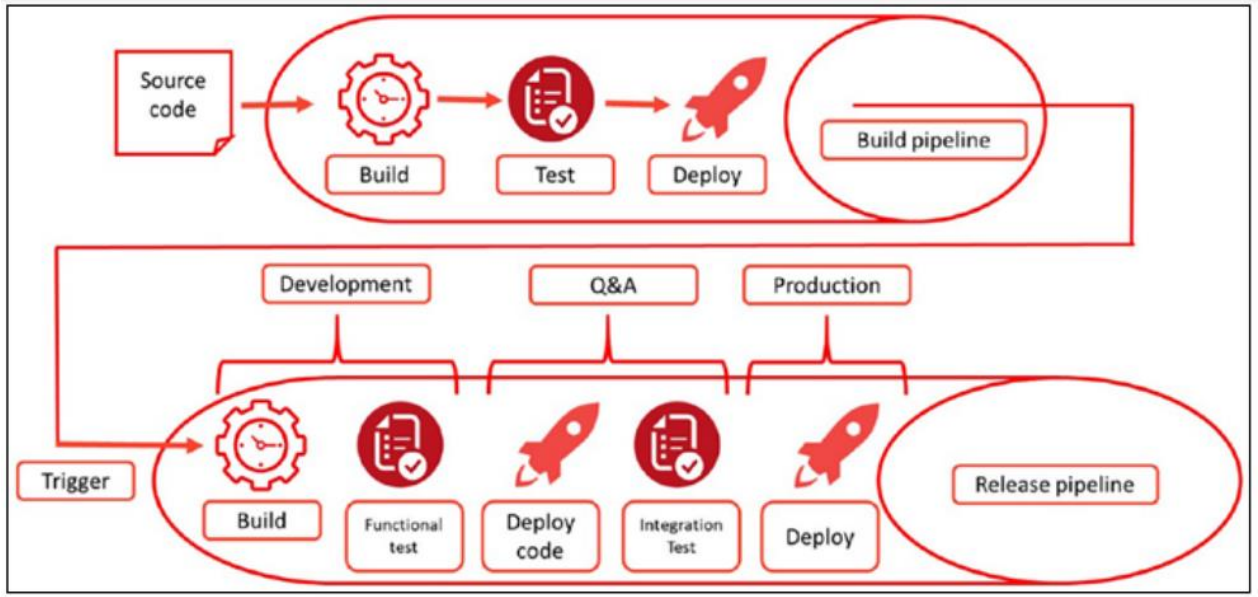


Figure: Conceptual diagram of a build and release pipeline

Conceptual diagram of a build and release pipeline

The Figure shows the concept of implementing a build and release pipeline with various test stages.

- The code is developed in the build pipeline and then sent to a release pipeline where the code is configured and released for production.
- During the release stages, the full build is tested in a test or **Quality and Assurance (Q&A)** Assurance environment.
- In Q&A, the build is accepted and released for deployment into production

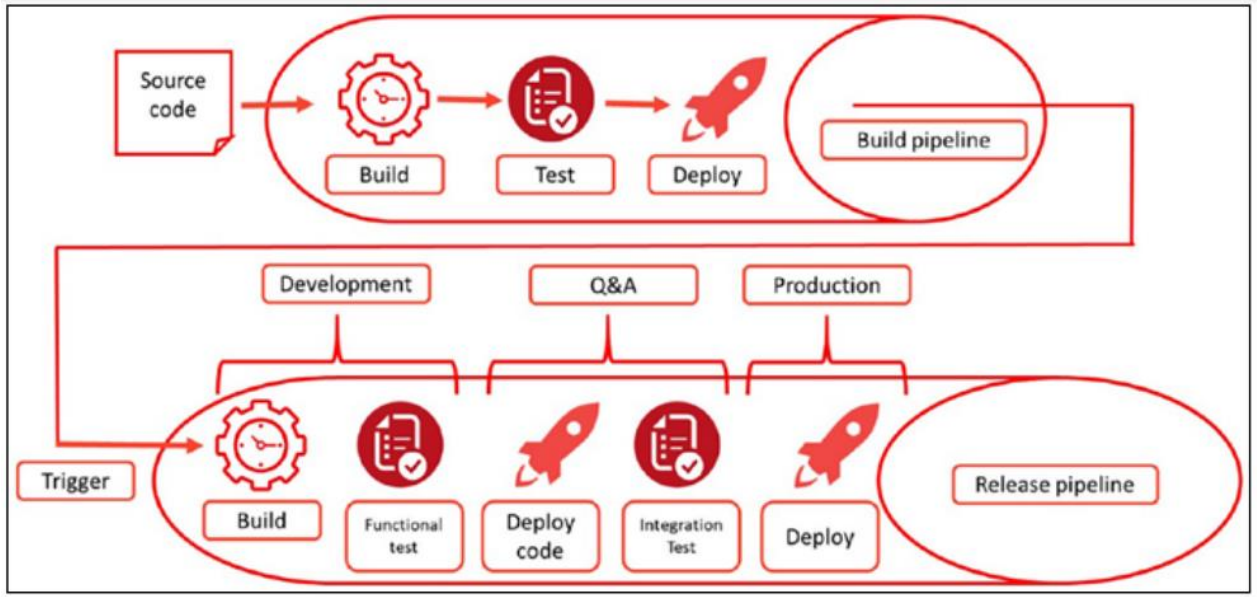


Figure: Conceptual diagram of a build and release pipeline

Using push and pull principles in CI

CI/CD pipelines work with branches, although other terms can be used for this. The main branch is sometimes referred to as a mainline or, when teams work in GCP, as a trunk.

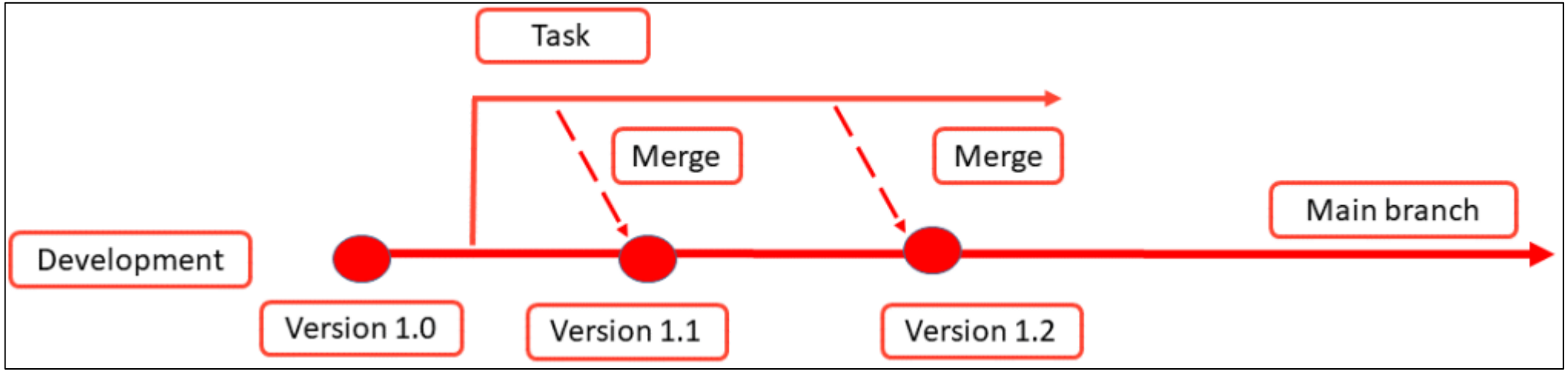
The most important principle to remember is that a development team has one main branch or mainline

two ways of pushing new code to that main branch

1. Pushing the code directly to the main branch
2. Pushing code to forks of the main

Pushing the code directly to the main branch

- In this method, the developers work directly in the main code; they change small pieces of the code and merge these directly back into the main branch.
- Pushing code back to the main branch is called a **commit**.
- The big advantage is that developers immediately see whether the change is done correctly, with the possibility to revert the change without having a huge impact on the main as a whole.
- the developers are responsible for the build, the commit, and the result

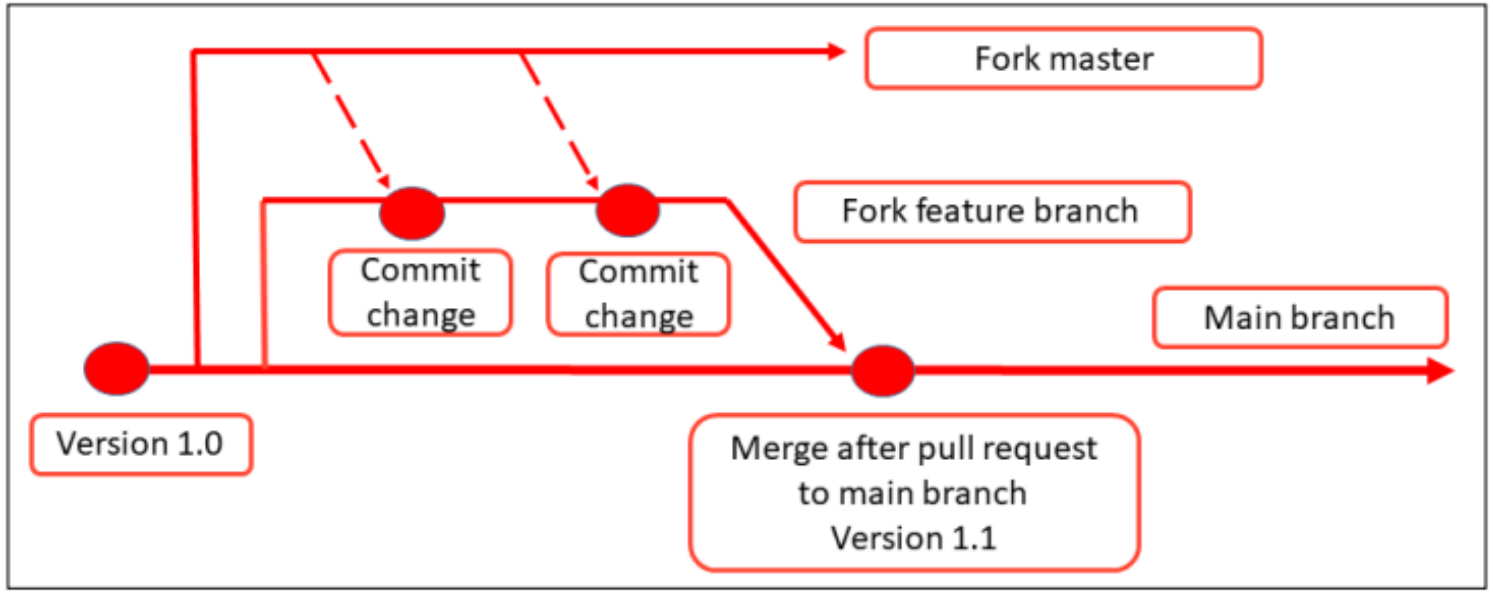


Developers merging code directly to the main branch

2. Pushing code to forks of the main

In this method, teams copy code from the main and create a separate or feature branch.

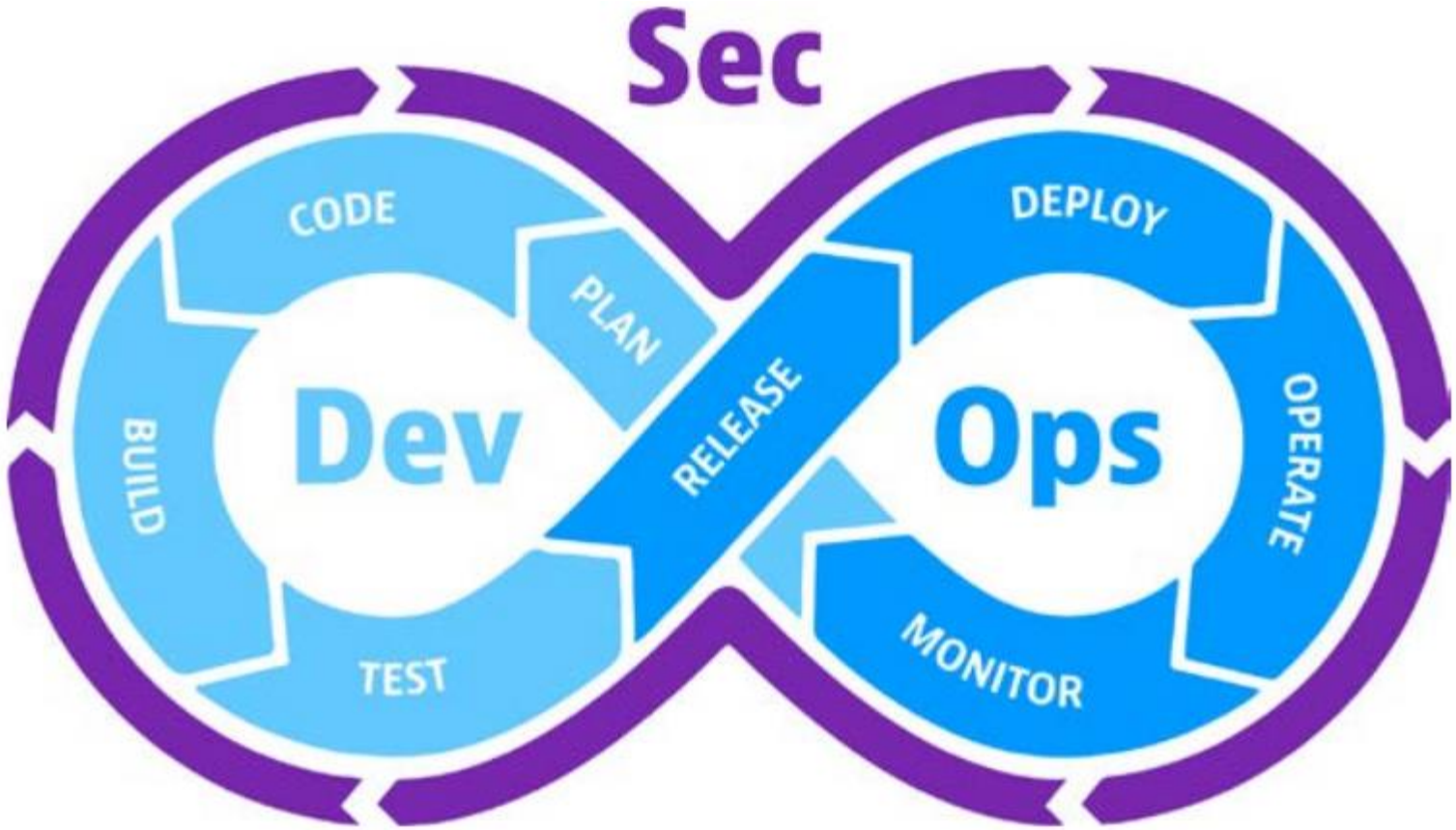
- This is also referred to as **forking**: developers create a feature branch by taking a copy from the source code on the main branch.
- They do their development on this forked code



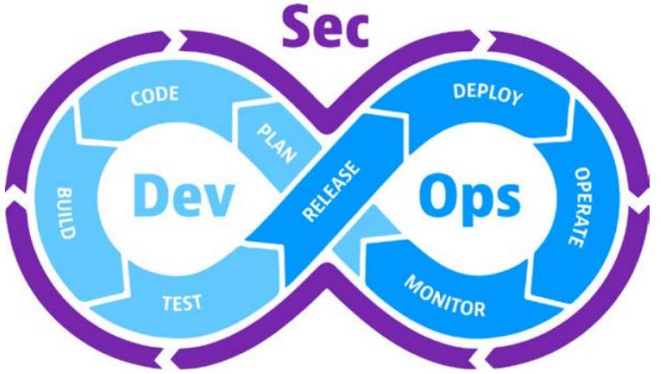
- Developers can work in isolation on the forked code, and when they're done, commit the code back to the main branch, merging the new features or builds with it.
- This can't be done on a frequent basis as intensive testing is required before the merging takes place

Using the DevSecOps Maturity Model

- Security is not a sauce that we put on top of products when they are finished.
- Security policies have to be applied from the first moment of development, all the way up to deployment to production. That's where DevSecOps comes in.



Using the DevSecOps Maturity Model



DevSecOps Maturity Model (DSOMM) is divided into five levels, each representing a different level of maturity in terms of security integration into the DevOps process

Level 0—No security culture: There is no formalized security program or processes, and security is not a consideration in the software development process.

Level 1—Siloed security: Security is taken into account but is limited to a separate team or department. Security is not integrated into the software development process, and there is no collaboration between the security and development teams.

Level 2—Integrated security: Security is integrated into the software development process and there is collaboration between security and development teams. However, the security process is not yet automated, and security testing is not performed in every stage of the development process.

Level 3—Continuous security: Security is integrated into every stage of the software development process. Security testing is automated, and security checks are performed continuously throughout the development process.

Level 4—Continuous improvement: The DevOps process is continuously monitored and improved to ensure that security is always a top priority. The organization has a culture of security, and security is considered as part of every decision and action taken

Introducing AIOps and GreenOps in Multi-Cloud

AIOps stands for **Artificial Intelligence for Operations**, but what does it really mean? AIOps is still a rather new concept but can help to optimize your multi-cloud platform.
It analyzes the health and behavior of workloads end to end—that is, right from the application's code all the way down to the underlying infrastructure

Introducing AIOps and GreenOps in Multi-Cloud

Understanding the concept of AIOps

- AIOps combines analytics of big data and ML to automatically investigate and remediate incidents that occur in the IT environment.
- AIOps systems learn how to correlate incidents across the various components in the environment by continuously analyzing all logging sources and the performance of assets within the entire IT landscape of an enterprise.
- They learn what the dependencies are inside and outside of IT systems

AIOps require highly sophisticated systems, comprising the following components:

- Data Analytics
- Machine Learning
- Automation
- Visualization

Optimizing cloud environments using AIOps

The two major benefits of AIOps are,

- first, the speed and accuracy in detecting anomalies and responding to them without human intervention.
- Second, AIOps can be used for capacity optimization. Most cloud providers offer some form of scale-out/-up mechanism driven by metrics, already available natively within the platform.

Optimizing cloud environments using AIOps

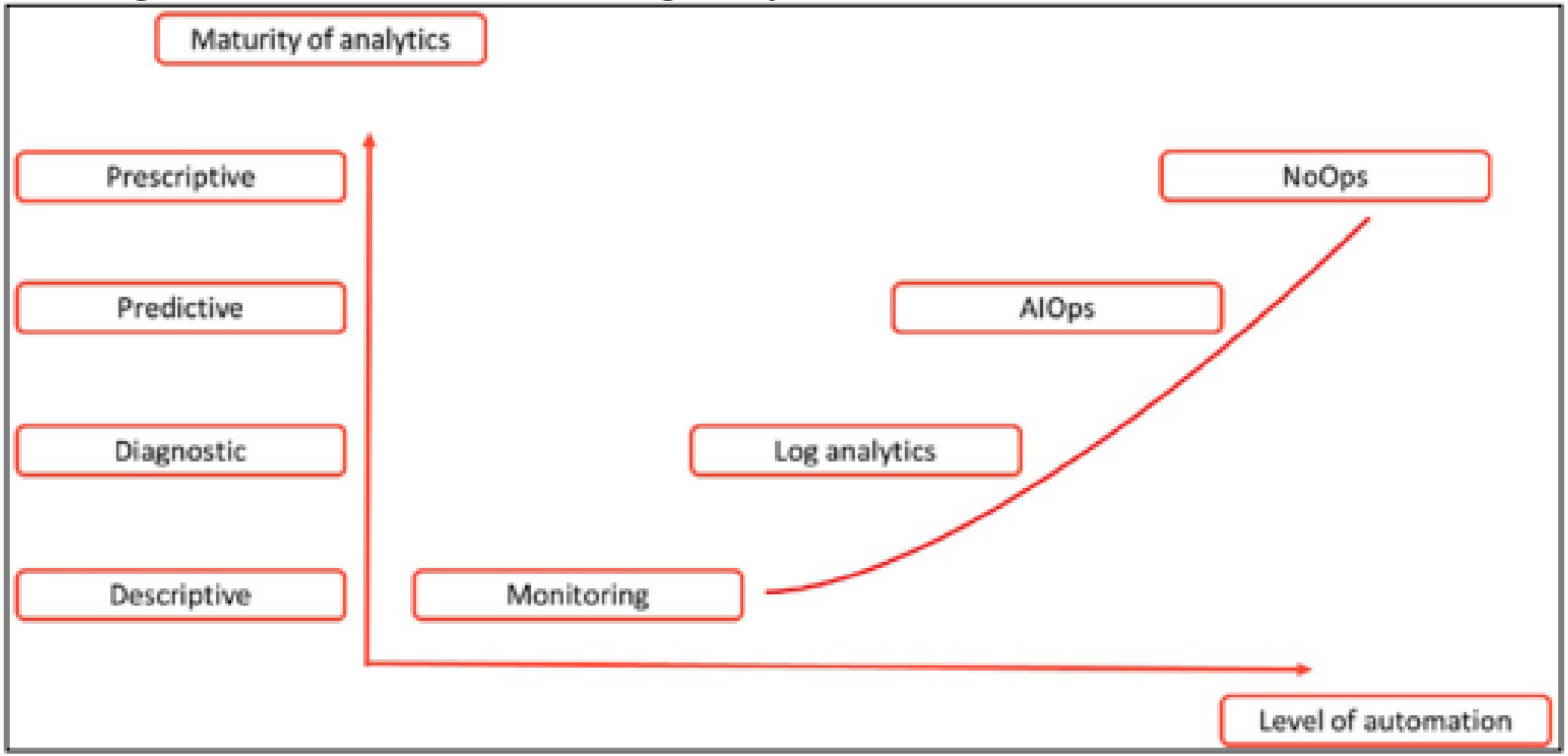


Figure Evolution of monitoring to AIOps

Optimizing cloud environments using AIOps

The following guidelines are recommended to successfully implement an AIOps strategy:

1. AIOps systems are learning systems
2. Data is essential in AIOps
3. Most important in a successful implementation is to standardize

Exploring AIOps tools for multi-cloud

AWS offers CloudWatch and X-Ray.

The latter is a distributed tracing service that provides end-toend visibility into application performance and behavior and allows users to visualize and analyze the flow of requests and responses across distributed systems.

X-Ray can identify performance bottlenecks and errors in complex microservices environments.

In Google Cloud Platform Services (GCP), we could identify Cloud Operations—formerly **Stackdriver—**as an AIOps-enabled tool, since it collects and analyzes metrics, logs, and traces from GCP resources and applications, and provides visibility into operational health and performance

In Oracle Cloud Infrastructure (OCI) , we would look at Oracle Management Cloud, a suite of management and monitoring services for OCI

Some examples of more cloud-agnostic AIOps tools are Dynatrace, Splunk, Cisco AppDynamics, Moogsoft, and IBM Cloud Pak for Watson.

Introducing AIOps and GreenOps in Multi-Cloud

Exploring AIOps tools for multi-cloud

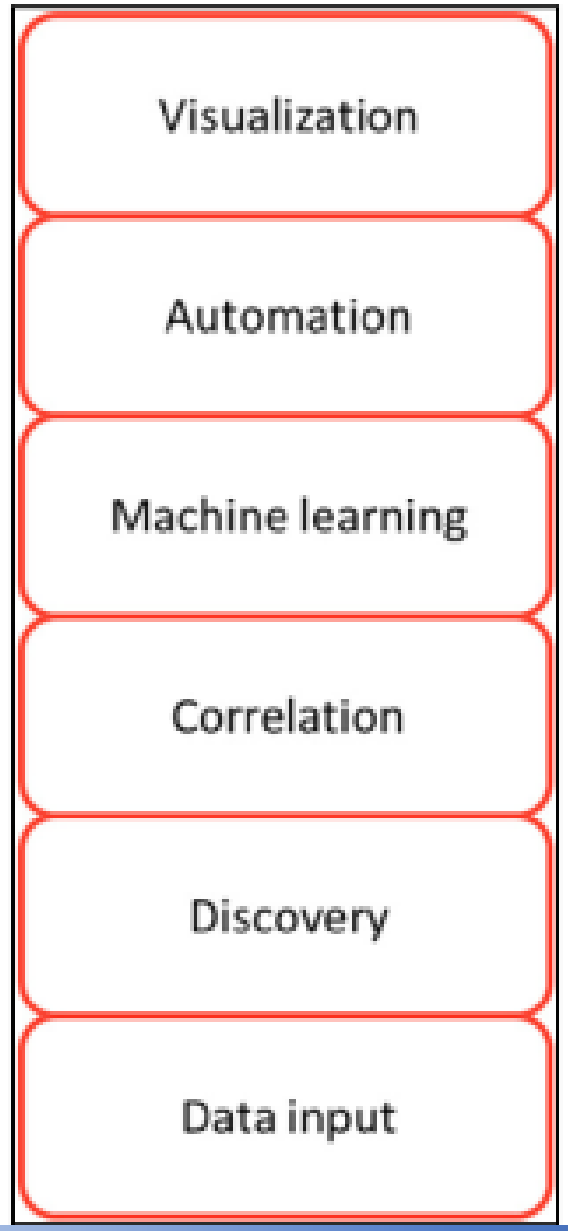


Figure: Layers of AIOps

Introducing GreenOps

The concept of GreenOps is becoming increasingly relevant in today's world, where concerns about climate change and sustainability are growing.

What do we mean by GreenOps?

Before we dive into that, it's relevant to notice that GreenOps and AIOps are actually quite intensively related. Both concepts make use of AI, as we will learn

we can define GreenOps as the practice of using cloud technology to optimize the environmental sustainability of IT operations, helping organizations to reduce their carbon footprint and operate in a more environmentally friendly manner.