18

Developing for Multi-Cloud with DevOps and DevSecOps

The typical reason why most enterprises adopt the cloud is to accelerate application development. Applications are constantly evaluated and changed to add new features. Since everything is codified in the cloud, these new features need to be tested on the infrastructure of the target cloud. The final step in the life cycle of applications is the actual deployment of applications to the cloud and the handover to operations so that developers have their hands free to develop new features again, based on business requirements.

To speed up this process, 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. Consistency is crucial: the source code needs to be under strict version control. That is what CI/CD pipelines are for: continuous integration and continuous delivery and deployment.

We will study the principles of DevOps, how CI/CD pipelines work with push and pull mechanisms, and how pipelines are designed so that they fit multi-cloud environments. Next, we will discuss how we must secure our DevOps processes using the principles of the DevSecOps Maturity Model and the most common security frameworks.

In this chapter, we're going to cover the following main topics:

- Introducing DevOps and CI/CD
- Using push and pull principles in CI
- Designing the multi-cloud pipeline

- Using the DevSecOps Maturity Model
- Automating security best practices using frameworks

Introducing DevOps and CI/CD

Before we get into the principles of DevSecOps, we need to have a good understanding of DevOps. There are a lot of views on DevOps, but this book sticks to the definition and principles as defined by the **DevOps Agile Skills Association** (**DASA**). It defines a DevOps framework based on six principles:

- **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.
- **Create with the end in mind**: How will the application look when it's completely finished?
- **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*.
- **Cross-functional autonomous teams**: Teams need to be able and allowed to make decisions themselves in the development process.
- **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.
- **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.

DevOps has been described in the literature as *culture*, a new way of working. It's a new way of thinking about developing and operating IT systems based on the idea of a feedback loop. Since cloud platforms are code-based, engineers can apply changes to systems relatively easily. Systems are code, and code can be changed, as long as changes are applied in a structured and highly controlled way. That's the purpose of **CI/CD pipelines**.

Continuous Integration (**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) 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. We have to make a note here: continuous delivery and continuous deployment are not the same thing.

Continuous delivery is putting the artifacts, built in the CI process, in environments, typically development, staging, and production, one after the other, with testing and approvals in between. Continuous deployment means code being put into the production environment from CI in a completely automated way without any human intervention.

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.

To learn more, we refer you to the following blog by Martin Fowler: `https://www.martinfowler.com/bliki/ContinuousDelivery.html`

CI/CD enables the DevOps cycle. Combined with CI/CD, all responsibilities, from planning to management, lie with the team, and changes can reach the customer much faster through an automated and robust development process. *Figure 18.1* shows the DevOps cycle with CI/CD:

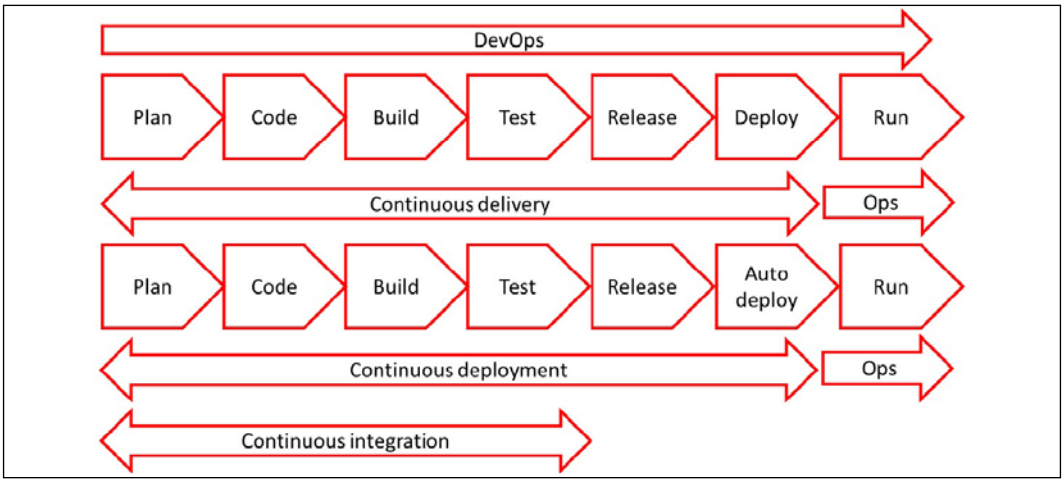

Figure 18.1: DevOps cycle with CI/CD

In the next section, we will study how to get started with 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.

With CI, development teams can change code as often as they want, leading to the continuous improvement of systems. Enterprises will have multiple development teams, working in multi-cloud environments, which makes it necessary to have one way of working. Fully automated processes in CI/CD pipelines can help keep environments consistent. CI/CD and DevOps are, however, not about tools. They're about culture and *sticking to processes*.

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

1. Implementing an effective CI/CD pipeline begins with all stakeholders implementing DevOps processes. One of the key principles in DevOps is autonomous teams that take end-to-end responsibility. It's imperative that the teams are given the authority to make decisions and act on them. Typically, DevOps teams are agile, working in short sprints of 2 to a maximum of 4 weeks. If that time is wasted on getting approval for every single detail in the development process, the team will never get to finish anything in time.
2. Choose the CI/CD system. There are a lot of tools on the market that facilitate CI/CD. Jenkins is a popular one, but a lot of companies that use Azure choose to work in Azure DevOps. Next, GitHub Actions has gained a lot of popularity. Involve the people who have to work with the system daily and enable them to take a *test drive* with the application. Then, make a decision 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. An important element of CI/CD is the automation of testing, so the first step is to create an automated process pipeline. Enterprises often already have quality and test plans, possibly laid down in a **Generic Test Agreement (GTA)**. This describes what and how tests must be executed before systems are pushed to production. This is a good starting point, but in DevOps, organizations work with a **Definition of Done (DoD)**:

- The DoD describes the conditions and the acceptance criteria a system must meet before it's deployed to production. The DoD is the standard of quality for the end product, application, or IT system that needs to be delivered. In DevOps, teams work with user stories. An example of a user story is: "as a responsible business owner for an online retail store, I want to have multiple payment methods so that more customers can buy our products online." This sets requirements for the development of applications and systems. The DoD is met when the user story is fulfilled, meaning that unit testing is done, the code has been reviewed, acceptance criteria are signed off, and all functional and technical tests have passed.
- *Figure 18.2* 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:

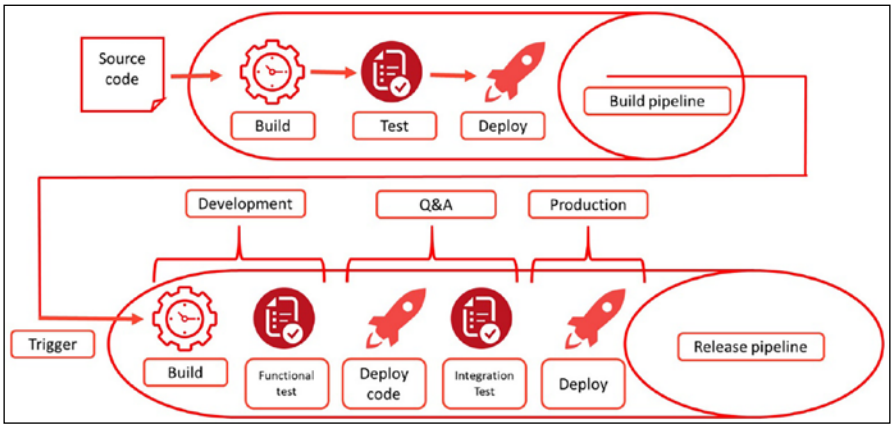

Figure 18.2: Conceptual diagram of a build and release pipeline

4. Automate as much as possible, as one of the principles of DevOps states. This means that enterprises will have to adopt working in code, including **Infrastructure as Code** (**IaC**). In CI/CD, teams work from one repository, and this means that the application code and the infrastructure code are in the same repository, so that all teams can access both whenever they need to.

If all these steps are followed, organizations are able to start working in DevOps teams using CI/CD. In the next sections, CI/CD is explained in more detail, starting with version control, and then discussing the functionality of commits and push and pull mechanisms in the pipeline.

Working under version control

By working from one code repository with different teams, version control becomes crucial in CI/CD. Git and Subversion are popular version control systems that enable teams to organize their files that form the source code, test scripts, deployment scripts, and configuration scripts used for applications and infrastructure components. Everything is code, which means that systems consist of a number of code packages: the code for the VM, the code for how the VM should be configured based on policies, and the application code itself. A version control system also enables teams to retrieve the historical state of systems, in case a deployment fails or systems are compromised and need to be rebuilt.

Version control systems keep track of changes to files that are stored in the repository. In DevOps, these changes are commonly referred to as commits, something that we'll discuss further in the next section, *Using push and pull principles in CI/CD*. A **commit** comprises the code change itself, along with metadata on who made the change and the rationale behind the code change. This ensures that code is kept consistent and, with that, repeatable and predictable. It also means that teams are forced to document everything in the repository and bring it under version control.

This list contains many of the items that need to be under version control:

- Application code
- API scripts and references (what is the API for?)
- IaC components such as components such as VMs, network devices, storage, images for operating systems, DNS files, and firewall configuration rules
- Infrastructure configuration packages
- Cloud configuration templates, such as AWS CloudFormation, **Desired State Configuration** (**DSC**) in Azure, and Terraform files
- Code definitions for containers, such as Docker files
- Container orchestration files, such as Kubernetes and Docker Swarm files
- Test scripts

Once companies have implemented this, they need to maintain it. This is not a one-time exercise. Teams should confirm that version control is applied to application code, systems configuration, and automation scripts that are used in the CI/CD pipeline. Only if this is applied and used in a consistent way will enterprises be able to deploy new applications and systems rapidly, yet securely and reliably.

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. Next, we will see two ways of pushing new code to that main branch in the following sections.

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**. These commits are done several times per day, or at least as soon as possible. Working in this way ensures that releases can be done very frequently, as opposed to working in code forks that result in separate or feature branches, which are described in the second method. *Figure 18.3* shows how direct pushes to the main branch work:

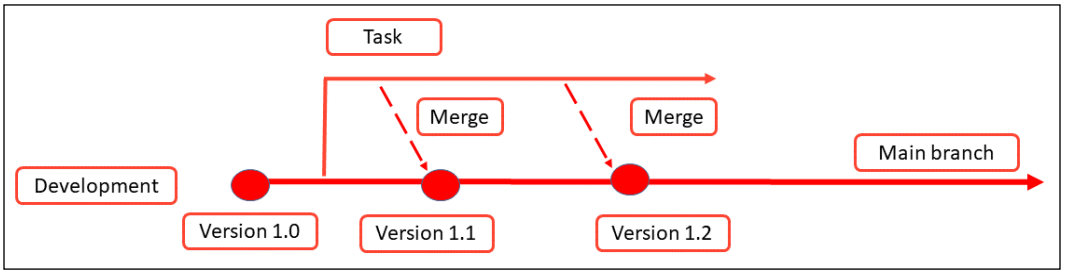

Figure 18.3: Developers merging code directly to the main branch

The idea behind CI is that companies get rid of long, complex integrations. Developers work in small batches of the code that they frequently commit to the main. 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. This is DevOps—the developers are responsible for the build, the commit, and the result: you break it, you fix it. Automated tests that are executed after the code commit are crucial to ensure that systems keep running without failures.

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**. In GCP, this is not trunk-based development, or better said: this is referred to as feature-driven development.

This method is often used for major developments, creating new features. 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. The downside is that this can lead to complex integrations. This can't be done on a frequent basis as intensive testing is required before the merging takes place. *Figure 18.4* shows how feature branches operate in a workflow:

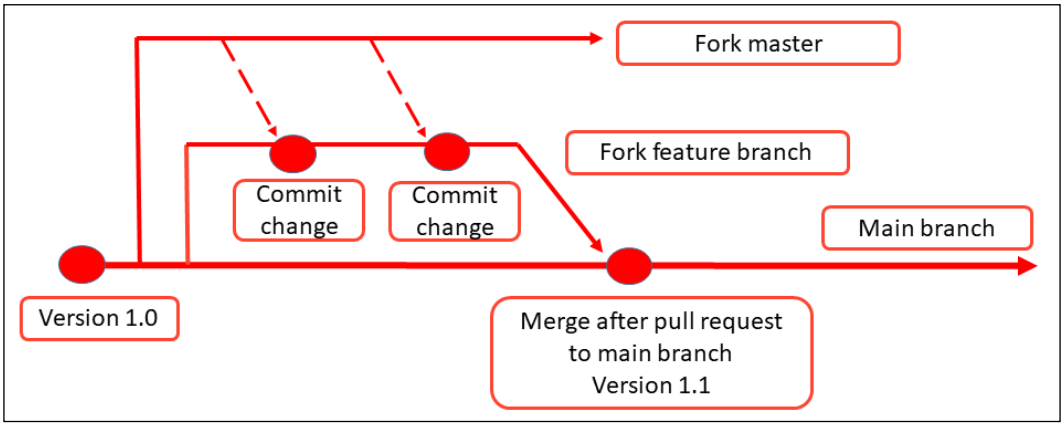

Figure 18.4: Developers working in a feature branch before merging to the main branch

In both methods, code gets pushed to the repository in, for example, GitHub. As soon as a developer has committed their code, they execute a pull request. This is the stage where the new, changed code is reviewed before the changes are actually merged into the main branch.

Best practices for working with CI/CD

There are a few best practices to remember when working with CI/CD. One of the biggest pitfalls is that code reviews can often be too extensive, meaning that developers have to get approval from different stakeholders before they can push the code to production. This will cost a lot of time and effectively slow down the DevOps process. Companies that adopt DevOps should have two principles implemented:

- The **four-eyes principle**: Have code reviewed while it's being written by working in developer pairs, where the second developer reviews the code of the first developer. This is also referred to as **pair programming**. Peer review is another method: here, the authors of the code each review at least one other developer's work, typically at the end of the development process.

- Running automated test scripts is most important. These scripts must be executed before code is actually committed to the main branch to make sure that systems keep functioning after the code commit.

By following these principles, we are already applying the first principles of DevSecOps. But it's not enough: we must really integrate security into DevOps. To do that, we can work with the DevSecOps Maturity Model. We will study this in the next section.

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. The position of security in the DevOps cycle is shown in the following diagram:

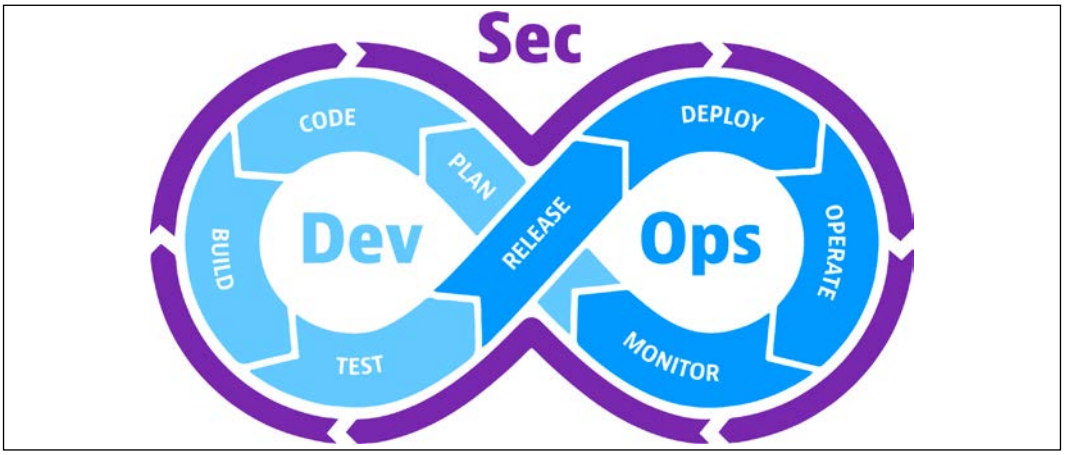

Figure 18.5: The DevSecOps cycle

The **DevSecOps Maturity Model** of the **Open Web Application Security Project** (**OWASP**) is a framework that helps organizations assess and improve their software development and delivery practices. The model aims to integrate security practices into the DevOps process that we described in the previous sections. By using this model, businesses can improve the security of their software products and reduce the risk of data breaches and cyber-attacks.

The DevSecOps Maturity Model can be found at `https://owasp.org/www-project-devsecops-maturity-model/`.

The OWASP **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. These five levels represent where companies stand in terms of the maturity of their DevSecOps implementation:

- **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.

Now, let's look at how businesses can use this maturity model to improve their DevSecOps practices:

1. **Assess current security practices**: The first step in using the DevSecOps maturity model is to assess the current security practices of the organization. During the assessment, we identify the current level of security integration into the DevOps process and consider the gaps and weaknesses in the process.
2. **Set goals for security integration**: The goals should be **Specific, Measurable, Achievable, Relevant, and Time-Constrained (SMART)**. For example, the organization may set a goal to automate security testing in every stage of the development process by the end of a specified period.
3. **Create a roadmap for improvement**: The roadmap should include specific actions and milestones that need to be achieved to reach the desired level of maturity. The roadmap should also include timelines and responsibilities for each action item.

4. **Implement security automation tools**: We must automate security testing in every stage of the development process. This can be achieved by implementing security automation tools, such as **Static Application Security Testing (SAST)** and **Dynamic Application Security Testing (DAST)** tools. These tools can identify vulnerabilities and weaknesses in the software code and provide developers with feedback on how to fix them. With this, we reach Level 3 of the DSOMM.
5. **Integrate security into the DevOps process**: Security aspects must be considered in the design, development, testing, deployment, and operations phases of the process. This is driven by culture: the mindsets of all of those involved have to be focused on developing and deploying software that is secure. To achieve this, the organization needs to create a culture of security, where everyone in the organization is responsible for the security of the software. This is the target of Level 4 of DSOMM.

We have been talking about software in this section, but DevSecOps also includes the underlying infrastructure in the clouds that we use to run our software. In major clouds such as Azure, AWS, and GCP, infrastructure is provided through a variety of services, including virtual machines, containers, and serverless computing, as we have learned throughout this book. These infrastructure services can be configured and managed using IaC tools such as AWS CloudFormation, **Azure Resource Manager (ARM)**, and Google Cloud Deployment Manager. IaC tools allow infrastructure to be treated as code, enabling developers and operations teams to version, test, and automate infrastructure changes just like they would with software changes.

DevSecOps teams can leverage the infrastructure services and IaC tools provided by the cloud providers to implement security controls throughout the software development life cycle. For example, they can use **Identity and Access Management (IAM)** policies to control access to resources, implement network security controls to protect against attacks, and use logging and monitoring to detect and respond to security incidents. For that reason, cloud infrastructure must also be included in the DevSecOps practices.

One of the first topics that we must address in security and DevSecOps is observability: we need to be able to see what is happening in our pipelines and detect issues in a timely manner. That's the topic of the next section.

Manage traceability and auditability

DevSecOps starts with observability in order to enable the management of traceability and auditability. This is becoming increasingly relevant in today's complex cloud-native environments where companies execute multiple releases of their software per month, week, or even day. And in the context of multi-cloud models and workflows, they might release software across various clouds and use services from different providers.

Observability is essential for maintaining the security and stability of modern software systems. By prioritizing traceability and auditability, organizations can achieve a higher level of observability, enabling them to identify potential security threats and respond to them quickly and effectively. This will help them to reduce the risk of security incidents, improve the reliability and performance of software systems, and ensure compliance with regulatory requirements.

At its core, observability refers to the ability to collect and analyze data from across the entire software stack, including infrastructure, applications, and user interactions. This can be in one cloud, but also across stacks that are deployed in multi-cloud settings. Here, we certainly need to know what service is hosted where and how services are interconnected.

Two critical components of observability are traceability and auditability. Traceability refers to the ability to track and trace events and activities across the entire software system. This means that DevSecOps teams must be able to follow the flow of data and code through the system, identifying potential points of vulnerability and ensuring that security controls are in place at each stage of the process.

Auditability is closely related to traceability, but it focuses more on the ability to review and verify the actions taken by various stakeholders in the software development and deployment process. This includes developers, security analysts, operations teams, and other stakeholders who play a role in ensuring the security and stability of the software system. By maintaining a complete and accurate audit trail of all actions taken, organizations can ensure accountability and transparency while also facilitating more effective incident response and forensic analysis in the event of a security incident.

To achieve this level of observability, we must invest in the right tools and processes. This includes leveraging automation tools to collect and analyze data from across the software stack, the cloud infrastructure that we have deployed and manage with IaC, implementing robust logging and monitoring capabilities, and integrating security and compliance checks into every stage of the software development and deployment process. In multi-cloud systems, we therefore need "agnostic" tools that can work with various clouds and software stacks. Popular examples of such tools are:

- **Prometheus**: An open-source monitoring system that can be used to collect and query metrics from multiple sources, including different cloud providers. It supports a variety of data sources and can be integrated with many different tools.
- **Grafana**: Like Prometheus, Grafana is open source. It helps to visualize and analyze processes to create dashboards and alerts based on data from multiple sources, including various cloud stacks.
- **Fluentd**: Fluentd is an open-source data collector that can be used to collect, process, and forward log data from multiple sources, including different cloud providers. It supports a variety of outputs and can be configured to integrate with many different tools, making it a flexible choice for multi-cloud observability.
- **Jaeger**: Jaeger is an open-source distributed tracing system that can be used to trace requests across multiple services and cloud providers. It supports various tracing protocols and can be integrated with many different tools, making it a good choice for multi-cloud observability.
- **OpenTelemetry**: OpenTelemetry is a vendor-neutral observability framework that provides a standard way to collect, process, and export telemetry data from multiple sources, including different cloud providers. It supports various programming languages and can be integrated with many different tools, making it a popular choice for multi-cloud observability.

Choosing the right tools is one thing, but what do we monitor against? The answer to that question is: we implement security guardrails and guidelines to be secure and compliant. We can use security frameworks to help us in setting the appropriate levels when securing our platforms. This is the topic of the final section of this chapter.

Automating security best practices using frameworks

The hardest part in getting security to the appropriate level in organizations is to define when the organization is compliant, and environments are "secure enough"—if such a thing exists. The problem with security in any IT environment is that just like cloud technology itself, the tactics, techniques, and processes used to attack environments are also evolving fast. Hackers will constantly find new ways to compromise environments. That's why every team member in a DevOps team must be fully aware of security risks. Every choice that a team makes comes with a consequence that must be thought through in terms of security. Are we introducing a vulnerability or other risk by developing and deploying software or by using a specific cloud service? What do we need to do to protect the data, application, underlying infrastructure, connectivity, and ultimately, the user?

Frameworks such as OWASP, CIS, and MITRE ATT&CK can help in defining the level of risk and determining what security measures must be applied. The next step is to automate this. Automating security practices is an essential aspect of modern DevSecOps practices. By automating security practices, developers and operations teams can ensure that security controls are implemented consistently and effectively across their software and infrastructure.

Let's have a more detailed look at the most commonly used frameworks—OWASP, the CIS Controls, and MITRE ATT&CK:

- The OWASP Top 10 is a list of the most critical security risks to web applications. To automate security practices based on the OWASP Top 10, organizations can use automated tools to scan their applications for vulnerabilities, such as SQL injection or **Cross-Site Scripting (XSS)**. These tools can be integrated into the CI/CD pipeline to automatically test code changes for vulnerabilities before they are deployed. Additionally, organizations can use automated code analysis tools to identify vulnerabilities in code as it is being written.
- The CIS Controls are a set of guidelines for securing information systems. To automate security practices based on the CIS Controls, organizations can use configuration management tools such as Chef, Puppet, or Ansible to enforce security policies on infrastructure. These tools can be used to automatically configure security settings on servers, networks, and applications, ensuring that security controls are consistent across the organization. CIS has issued controls for each separate cloud, including Azure, AWS, GCP, Alibaba Cloud, and OCI.
- Lastly, the MITRE ATT&CK framework is a knowledge base of **Tactics, Techniques, and Processes (TTPs)**. To automate security practices based on the MITRE ATT&CK framework, organizations can use tools that monitor network and system activity for signs of attack. For example, they can use **Security Information and Event Management (SIEM)** tools to detect suspicious activity and use automated incident response tools to respond to security incidents as they occur. Using the framework to define what specific vulnerabilities are in an environment is not an easy task. Some adjacent tools might be handy when working with MITRE ATT&CK. One example is DeTTECT, originally invented for the Dutch bank Rabobank, one of the biggest financial organizations in the Netherlands. It assists in using ATT&CK to score and compare data log source quality, visibility coverage, detection coverage, and threat actor behavior.

19

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. AIOps tooling will help in discovering issues, thereby providing advice for optimization. The best part is that good AIOps tools do this cross-platform since they operate from the perspective of the application and even the business chain.

This chapter is an introduction to the concept of AIOps. The components of AIOps will be discussed, including data analytics, automation, and Machine Learning (ML). After completing this chapter, you will have a good understanding of how AIOps can help in optimizing cloud environments and how enterprises can get started with implementing AIOps.

We will also discuss a new concept in cloud operations, which is focusing on sustainability. With the enormous growth of the cloud, we have to be aware that the cloud uses data centers all over the world. These have an impact on our environment. GreenOps makes us conscious of the usage of the cloud in the most environmentally friendly way.

In this chapter, we're going to cover the following main topics:

- Understanding the concept of AIOps
- Optimizing cloud environments using AIOps
- Exploring AIOps tools for multi-cloud
- Introducing GreenOps

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.

Especially in the world of multi-cloud, where enterprises have systems in various clouds and still on-premises, gaining visibility over the full landscape is not easy. How would an engineer tell that the bad performance of a website that hosts its frontend in a specific cloud is caused by a bad query in a database that runs from a data lake in a different cloud?

AIOps requires highly sophisticated systems, comprising the following components:

- **Data analytics:** The system gathers data from various sources containing log files, system metrics, monitoring data, and also data from systems outside the actual IT environment, such as posts on forums and social media. A sudden high number of incidents logged into the systems of the service desk may also be a source. AIOps systems will aggregate the data, look for trends and patterns, and compare these to known models. This way, AIOps is able to determine issues quickly and accurately.
- **ML:** AIOps uses algorithms. In the beginning, it will have a baseline that represents the normal behavior of systems, applications, and users. Applications and the usage of data and systems might change over time. AIOps will constantly evaluate these new patterns and learn from them, teaching itself what the new normal behavior is and what events will create alerts. From the algorithm, AIOps will prioritize events and alerts and start remediating actions.
- **Automation**: This is the heart of AIOps. If the system detects issues, unexpected changes, or abnormalities in behavior, it will prioritize and start remediation. It can only do that when the system is highly automated. From the analytics output and the algorithm, AIOps systems can determine what the best solution is to solve an issue. If a system runs out of memory because of peak usage, it can automatically increase the size of memory. Some AIOps systems may even be capable of predicting the peak usage and already start increasing the memory before the actual usage occurs, without any human intervention. Be aware that cloud engineers will have to allow this automated scaling in the cloud systems themselves.

- **Visualization**: Although AIOps is fully automated and self-learning, engineers will want to have visibility of the system and its actions. For this, AIOps offers real-time dashboards and extensive ways of creating reports that will help in improving the architecture of systems. That's the only thing AIOps will not do: it will not change the architecture. Enterprises will still need cloud architects for that. The next section discusses how AIOps can help in improving cloud environments.

AIOps is a good extension of DevOps, where enterprises automate the delivery, deployment, and operations of systems. With AIOps, they can automate operations.

Also, AIOps evolves, for instance, by including security into the concept. A good article about AISecOps can be found at `https://www.forbes.com/sites/forbestechcouncil/2022/02/16/coining-the-term-ai-secops-why-your-business-should-consider-aiops-for-cybersecurity/?sh=57406c1f5449`.

Is there something after AIOps? Yes, it's called **NoOps**, or **No IT Operations**, where all operational activities are fully automated. The idea here is that teams can completely concentrate on development. All daily management routines on IT systems are taken over by automated systems, such as system updates, bug fixing, scaling, and security operations. But there is a trade-off here: although it's called NoOps, engineers and some form of governance are still needed to set up the systems and implement the operation's baseline.

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. AIOps can optimize this scaling since it knows what thresholds are required to do this, whereas the cloud provider requires engineers to define and hardcode it.

Since the system is learning, it can help in predicting when and what resources are needed. The following diagram shows the evaluation of operations, from descriptive to prescriptive. Most monitoring tools are descriptive, whereas AIOps is predictive:

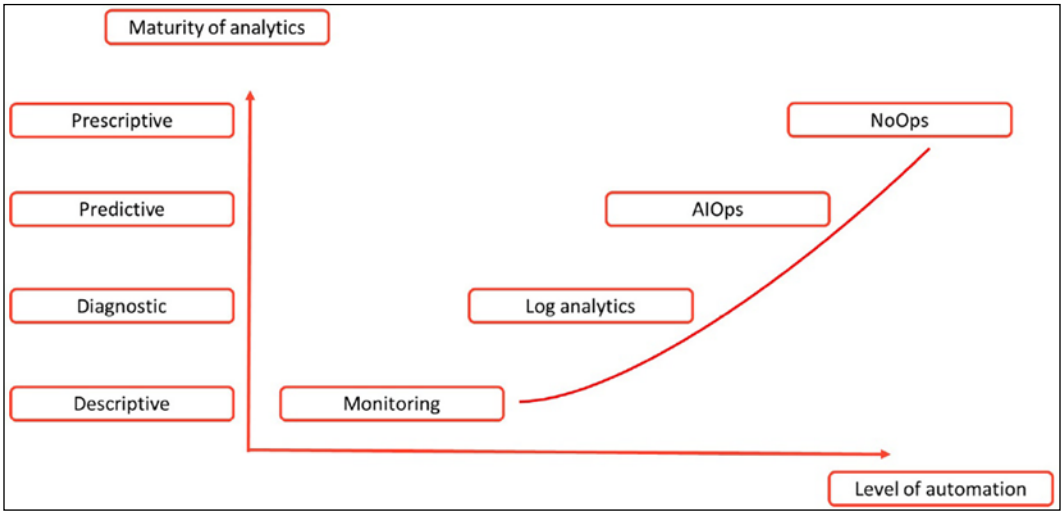

Figure 19.1: Evolution of monitoring to AIOps

Monitoring simply registers what's happening. With log analytics, companies can set a diagnosis of events and take remediation actions based on the outcomes of these analyses This is all reactive, whereas AIOps is proactive and predictive. By analyzing data, it can predict the impact of changes. The last step is systems that are prescriptive, being able to tell what should happen and already preparing systems for events, fully automated. Some very sophisticated AIOps systems can already do that.

So, we can use AIOps to help us manage and optimize the complex and dynamic nature of these environments. But how do these systems do that? AIOps will use ML algorithms to analyze data from cloud infrastructure, applications, and logs to identify patterns and anomalies that could lead to issues. This enables IT teams to predict and prevent problems before they occur, reducing downtime and improving availability.

Next, with AIOps, we can **automate** the response to common issues in cloud environments, such as scaling resources up or down based on demand, restarting failed services, or reconfiguring network settings. This reduces the need for manual intervention, saving time and reducing the risk of human error.

One of the most common areas where AIOps shows value is **root cause analysis**. When issues occur in cloud environments, AIOps can help identify the root cause by analyzing data from multiple sources and correlating events. This will help to quickly diagnose and resolve problems, reducing downtime and improving service levels. Automation and root cause analysis are ways to continuously optimize our cloud environments. Tools will continuously monitor cloud environments and recommend optimizations based on usage patterns and performance metrics. This includes optimizing resource allocation, identifying opportunities to reduce costs, and recommending improvements to application performance.

Enterprises are discovering AIOps because it helps them in optimizing their IT infrastructure. But how do companies start with AIOps? The following guidelines are recommended to successfully implement an AIOps strategy:

- **AIOps systems are learning systems:** Enterprises will have to learn how to work with and interpret analysis from these systems as well to get the best out of it. So, don't try to get the entire IT environment under AIOps in one go, but start with a small pilot and iterate from there.
- **Data is essential in AIOps:** This should not only be data that comes from IT systems but also business data. After all, the great benefit of AIOps is that it can take actions that are based on business data. If AIOps knows that certain products sell better at specific times of the year—which is business-driven data—it can take actions to optimize IT systems for that peak period. Also, if it turns out that systems are not used as expected, AIOps will be able to analyze the usage and correlate it with other events. In that way, AIOps can be a fantastic source for the business in becoming a truly data-driven organization. Businesses, therefore, absolutely need to be involved in the implementation of AIOps.
- **Most important in a successful implementation is to standardize**: Throughout this book, it has been stressed that multi-cloud environments need to be implemented in a consistent way, meaning that infrastructure must be defined and configured as code so that it can be deployed in a consistent manner to various cloud platforms. The code must be centrally managed from one repository, as much as possible. This will ensure that AIOps systems will learn quickly how systems look and how they should behave so that anomalies can be detected quickly.

AIOps can also help in testing against real-life scenarios and take much more into consideration in terms of testing. As such, AIOps is a great extension to DevOps and CI/CD pipelines executing frequent releases to applications. AIOps will know which systems will be impacted when changes are applied to a certain system, and also vice versa: which systems will respond to changes in terms of performance and stability. These can be systems that are hosted in different clouds or platforms; they can be part of the application chain.

This problem of the coexistence of applications and systems that disproportionately consume resources is referred to as **noisy neighbor**. AIOps will identify the neighbors, warn them of upcoming changes, and even take proactive measures to avoid the applications and systems from running into trouble. This goes beyond the unit and integration tests that are triggered by a CI/CD pipeline.

Today's multi-cloud environments are complex, with servers and services running in various clouds. Systems are connected over network backbones of different cloud platforms, routing data over the enterprise's gateways, yet continuously checking whether users and systems are still compliant with applied security frameworks. There's a good chance something may be missed when distributing applications across these environments.

AIOps can be used to improve the overall architecture. Architects will have much better insight into the environment and all the connections between applications and systems; this includes not only servers but also network and security devices. Next, AIOps will help in the distribution of applications across platforms and the scaling of infrastructure without impacting the neighbors, even if the neighbors are sitting on a different platform.

Exploring AIOps tools for multi-cloud

AIOps helps enterprises in becoming data-driven organizations. From the first chapter of this book, the message has been that IT—and IT architecture—is driven by business decisions. But business itself is driven by data: how fast does a market develop, where are the customers, what are the demands of these customers, and how can IT prepare for these demands? The agility to adapt to market changes is key in IT, and that's exactly what cloud environments are for—that is, cloud systems can adapt quickly to changes. It becomes even faster when data drives the changes directly, without human interference. Data drives every decision.

That's the promise of AIOps. An organization that adopts the principle of becoming a data-driven enterprise must have access to vast amounts of data from a lot of different sources, inside and outside IT. It needs to embrace automation. But above all, it needs to trust and rely on sophisticated technology with data analytics, AI, and ML. That's a true paradigm shift for a lot of companies. It will only succeed when it's done in small steps. The good news is that companies already have a lot of business and IT data available, which they can feed into AI and ML algorithms. So, they can get started, but first, they will need to select a platform or a tool.

Market analysts expect that the use of AIOps will grow from being worth around 26 billion USD in 2020 to well over 600 billion USD in 2030. This explains why a lot of leading IT companies are investing heavily in AIOps, including big names such as IBM, VMwire, and ServiceNow. Also, cloud providers themselves invest heavily in AIOps capabilities, for instance, in Azure Monitor, which already provides deep insights into the behavior of resources and applications in Azure.

AWS offers CloudWatch and X-Ray. The latter is a distributed tracing service that provides end-to-end 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 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. It can also use ML algorithms to detect anomalies and perform root cause analysis.

In OCI, we would look at **Oracle Management Cloud**, a suite of management and monitoring services for OCI. It includes features such as log analytics, application performance monitoring, and infrastructure monitoring, and can be used to automate operational tasks and improve application performance and reliability. It also includes machine learning algorithms to detect anomalies and perform root cause analysis.

How does an enterprise choose the right tool? When an enterprise is working in multi-cloud, it needs an AIOps solution that can handle multi-cloud. These are AIOps platforms that have APIs for the major cloud providers and can integrate with the monitoring solutions of these providers and the third-party tools that enterprises have in the cloud environments. An example of such a platform is Splunk Enterprise, which collects, correlates, and analyzes data from IT infrastructure, applications, and security systems.

In essence, all of these tools work in layers. The layers are depicted in the following diagram:

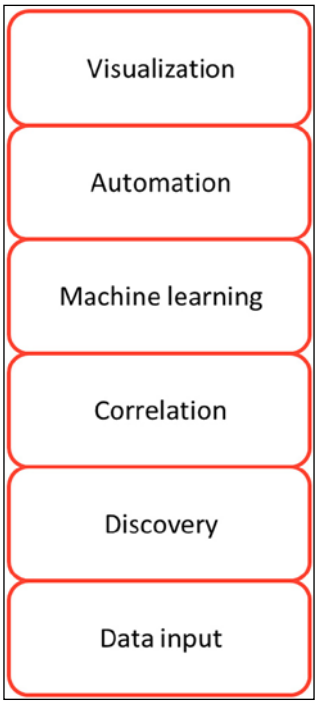

Figure 19.2: Layers of AIOps

Most AIOps systems combine a set of tools in the different layers into an AIOps platform that can handle the various aspects of AIOps.

The market for AIOps is thus rapidly growing. Some examples of more cloud-agnostic AIOps tools are Dynatrace, Splunk, Cisco AppDynamics, Moogsoft, and IBM Cloud Pak for Watson. All these platforms use AI and automation to monitor and analyze cloud environments, including containerized applications and microservices. They provide real-time insights into performance and user experience and can often automatically remediate issues by scaling resources and re-configuring settings.

These are just a few examples. Keep in mind that each tool has its own unique features and capabilities, and the best tool for a particular organization will depend on its specific needs and requirements.

Key in all these solutions is that they auto-discover any changes in environments in real time and can predict the impact on any other component in the IT environment before events actually occur, as well as from changes that are planned from CI/CD pipelines.

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.

In short, 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. The cloud offers a number of benefits when it comes to achieving sustainability goals. One of the most significant advantages of the cloud is that it allows organizations to use shared resources, which can reduce the overall energy consumption of IT operations. By using virtualized infrastructure and shared resources, organizations can improve the efficiency of their IT operations and reduce the number of physical servers they need to run.

Another advantage of the cloud is that it allows organizations to scale their operations up or down quickly and easily, depending on their needs. This can be especially useful for organizations that experience fluctuations in demand, as they can adjust cloud resources in real time to match the level of demand. By avoiding the need to maintain excess capacity, organizations can reduce their energy consumption and save money on their energy bills.

But how does this compare to AIOps? The answer to that question is that GreenOps also involves automation and machine learning to optimize IT operations and, with that, reduce energy consumption. By automating repetitive tasks and using machine learning algorithms to identify areas for optimization, organizations can improve the efficiency of their cloud environments. For example, machine learning algorithms can be used to optimize data center cooling, which can reduce energy consumption and improve the efficiency of IT operations.

Throughout this book, we have discussed the public clouds: Azure, AWS, GCP, OCI, and sometimes Alibaba Cloud. These are all so-called hyperscalers, meaning that they operate at a large scale, serving thousands of customers around the globe. These customers host their environments in the data centers of these providers. How will customers be able to influence and monitor their energy consumption and environmental footprint? Many cloud providers offer energy monitoring and reporting tools that allow organizations to track their energy usage and identify areas for improvement. They can also use cloud-based analytics tools to optimize their energy consumption and reduce their carbon footprint.

Measures to reduce the use of (heavy) resources, implementing automation and efficient automated scaling, will show directly in this energy monitoring. By optimizing IT operations and reducing energy consumption, we can reduce the carbon footprint, save money on energy bills, and—most important of all—help our world to become more sustainable.

Summary

AIOps was a new kid on the block but, since 2020, it has emerged as an almost essential platform for managing complex cloud environments. AIOps systems help organizations in detecting changes and anomalies in their IT environments and already predicting what impact these events might have on other components within their environments. AIOps systems can even predict this from planned changes coming from DevOps systems such as CI/CD pipelines. To be able to do that, AIOps makes use of big data analysis: it has access to a lot of different data sources, inside and outside IT environments. This data is analyzed and fed into algorithms: this is where **AI** comes in, and ML. AIOps systems learn so that they can actually predict future events.

AIOps are complex systems that require vast investments from vendors, and thus from companies that want to start working with AIOps. However, most organizations want to become more and more data-driven, meaning that data is driving all decisions. This makes a company more agile and faster in responding to market changes.

In the last section, we briefly discussed a new concept that becomes increasingly important in using cloud technology: GreenOps. We have to become more aware of the environmental impact of using cloud technology and the data centers of public cloud providers. GreenOps will help us to reduce our carbon footprint and operate our business in the cloud in a more environmentally friendly manner.

After completing this chapter, you should have a good understanding of the benefits as well as the complexity of AIOps. You should also be able to name a few of the market leaders in the field of AIOps. At the end of the day, it's all about being able to respond quickly to changes, but with minimum risk and, preferably, with a low carbon footprint.

Questions

1. AIOps correlates data from a lot of different systems, including IT systems that are not directly in the delivery chain of an application but might be impacted by changes to that chain. What are these systems called in terms of AIOps definitions?
2. Name at least two vendors of cloud-agnostic AIOps systems, recognized as such by market analysts.