

KodeKloud

The Ultimate DevOps Tools Guide For Teams

By Mumshad Mannambeth




KodeKloud The Ultimate DevOps Tools Guide For Teams

Introduction

When people talk about DevOps, they often talk about different popular tools like Docker, Kubernetes, Ansible, Terraform, Git, Jenkins, Prometheus, Grafana, and the list goes on...Every other month, it seems like there's a brand new tool being announced that's supposedly going to solve all of your scaling problems! The fact is that DevOps is a rapidly evolving field, and it can be challenging to keep up with the latest tools and techniques.

KodeKloud
The Ultimate DevOps Tools Guide For Teams

However, learning about the right DevOps processes for your organization, and how tools can enable those processes, can be a game-changer for your team's productivity and efficiency. By automating repetitive tasks, enhancing communication and collaboration, and providing real-time insights into your operations, these tools can help you deliver software faster and more reliably.

While DevOps is not just about tools, they do play a critical role in helping organizations implement best practices, processes, and automation. The challenge is understanding which tools are right for your mission-critical projects, especially when members of the team may not truly understand how the tools can be implemented. So when organizations decide to get serious about implementing DevOps and making it a top priority for their engineering teams, it can be an overwhelming initiative for everyone involved. The sheer number of tools and technologies to learn alone can cause analysis paralysis.

If this is the type of situation that your organization is in, then don't worry. This is feedback we frequently hear from teams seeking top-quality training. They'll come to us and say: "Management needs our team to shift workloads to Kubernetes and the cloud within 6 months."

The race against time begins, and team leads have to ask the tough question: “do we currently have the needed skills on our existing team? Do we need to bring in some external help which could very well take 3+ months? Or can we upskill and reskill our engineers which could start today?”

If your team needs help closing their skills gap for DevOps, then this is the resource you need to share with them to help them get started. In this guide, we'll break down some of the most popular DevOps tools and technologies, and we’ll provide practical guidance on how they can be used to modernize your development processes, improve efficiency, and ultimately help you build better products.

With that, let’s get started with a story...

A story of why DevOps matters

It all starts with an idea.

You have this idea that you think is going to change the world: you're going to build a website that books tickets to Mars in advance so that people don't have to pay too much or wait in a queue in the future. It sounds like a fun idea, but what do you do to get started?

As any other intelligent developer would do – you start with market research – oops, sorry I got that wrong. As any other intelligent developer would do - you get coding.

You open your favorite editor and start building your project. Hours later... you have the first version of your product ready. It's time to share it with the world. Now how do you do that?

Deploying your new application

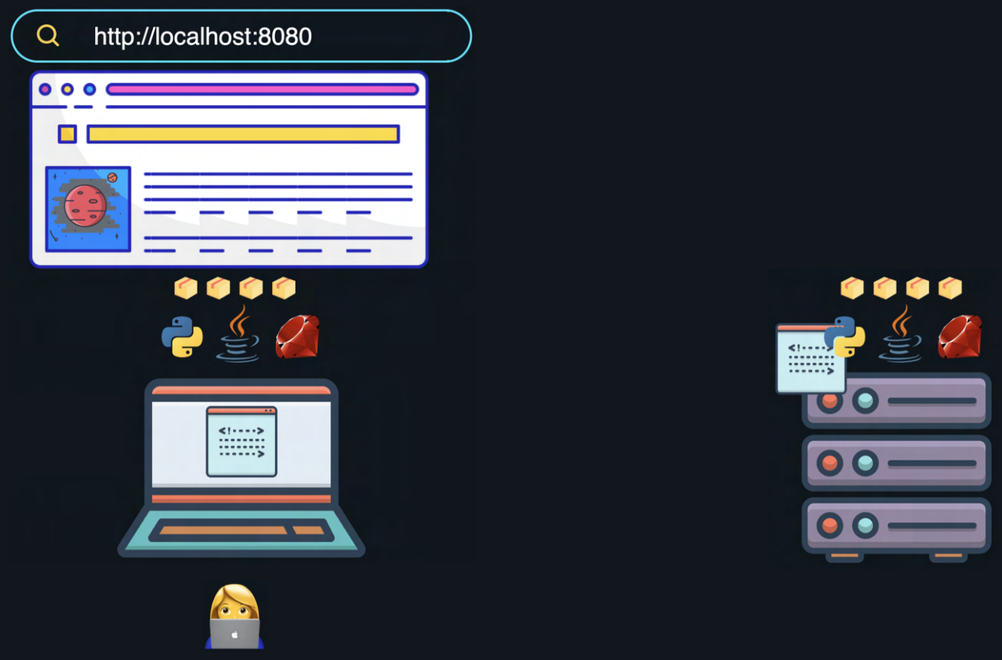
```
>_                    code.py

def book_my_ticket_to_mars():
    # world changing code here

# starts here
if __name__ == '__main__':
    book_my_ticket_to_mars()
```

KodeKloud
The Ultimate DevOps Tools Guide For Teams

This code is still running on your laptop, which means it's only accessible on your local host at **http://localhost:8080** for now. You've got to find somewhere to host it, because even if you find a way to share it with the world from your laptop, when you shut down your laptop, no one would have access to it. You need to host your application on a system that is never turned off.

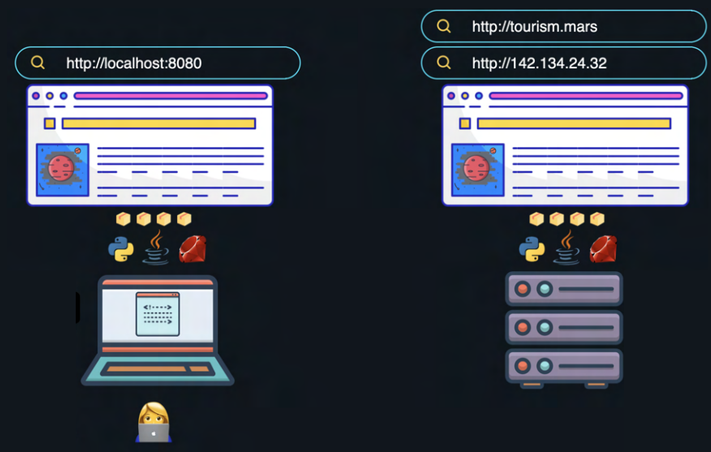
So you identify a server that is either a physical server in a data center that you rent, or a virtual machine (VM) in a data center or in the cloud that you pay for on-demand. You then need to copy the code to that server or VM and run the application. Except you can't just run an application on a system by simply copying code to it. You need to have the system configured first to be able to run it.

For example, if the application was written in Python or Java or any other programming language, you must have one of those programming languages or the runtimes in place on the server too. If the application uses any libraries or packages, then you have to have those exact versions of libraries and packages configured on the server in the exact same way.



Once all of that is set up, you now have your application running on the server. The server has a public IP address that you can use to access the app in your browser. Except, you don't want to have to share an IP address with people because no one would remember that. Instead, you purchase a domain name (like, say, https://tourism.mars) and map that to the IP. Now you have something to share with the world and you're ready for users!

KodeKloud
The Ultimate DevOps Tools Guide For Teams



You tweet it out tagging a certain popular individual who's fascinated with Mars, and that certain individual retweets it and off you go! Your website is now famous and you have thousands of users visiting your site and booking their future travel to the red planet.

You start getting a bunch of feature requests, and you need to start implementing the most important ones. How can you do that? Let's take a look.

Development and production environments

All of a sudden, you need to think about having and using 2 separate environments: the development environment, which is going to be your laptop or desktop, and the production environment, which is the server you're hosting the application on.

You start off by writing code on your laptop. This could be using your favorite text editor – like VSCode or PyCharm.

KodeKloud
The Ultimate DevOps Tools Guide For Teams

Depending on the programming language you're using, you may have to 'build' the application before deploying it. Just like how an application you want to run on Windows needs to have a .exe extension, or needs binaries for Linux, some programming languages need to be compiled before they can be served by webservers. Java apps, for example, need to be compiled in the WAR format. Converting code from a text format to a binary or executable format is known as building the code.

There are tools available such as the Python setup tools, or Maven or Gradle for other platforms. You usually have a build script that invokes these tools to build the application.

Once built, the executable is moved to a production environment - in this case, our server – and run in production, and that's referred to as the Deploy stage. This is a simple 3-step deployment process.

There are so many variations to this process and different ways and tools used to do this, but we're keeping it high level for now.

KodeKloud
The Ultimate DevOps Tools Guide For Teams

The complexity continues to increase...since you now have users visiting your website, those users are requesting more features, and so you bring your friends along as additional developers.

All of a sudden, everyone is working from their own development environments but on the same code base. To merge everyone's changes, they all copy their code to a central hub whenever they're ready. This means everyone is stepping on each other's toes, working on the same files at the same time, and creating conflicts that take hours to resolve.

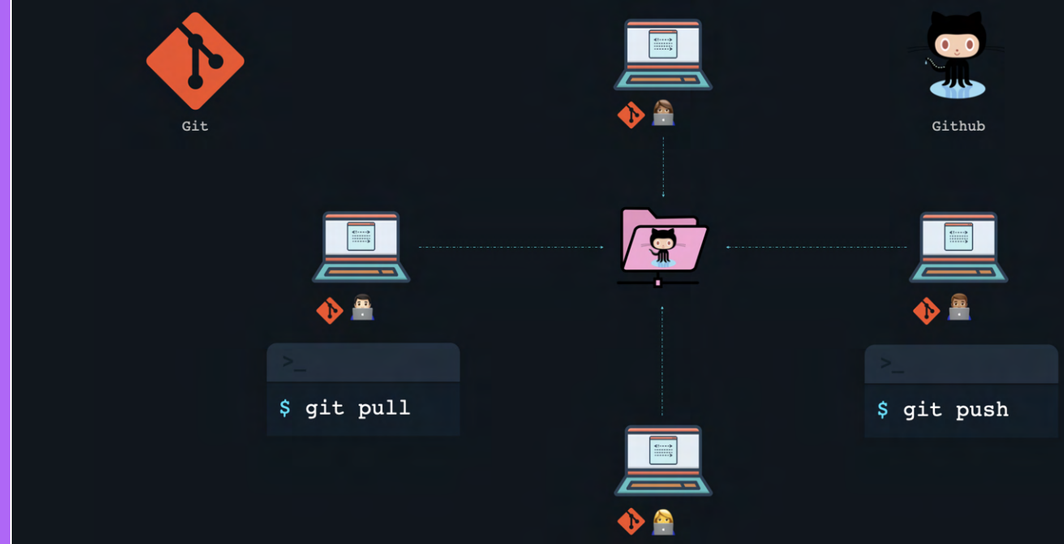
You need a solution that can help collaboration. That solution is Git.

Version control and team collaboration

Git helps developers work on the same application at the same time and collaborate efficiently.

All you need is for everyone on the team to install Git on their machines (which is very easy). Then, they can pull the latest code from the central repo using the git pull command. From there, they can add their own changes and then push them back to everyone using the git push command.

KodeKloud
The Ultimate DevOps Tools Guide For Teams



Once pushed to the central repo, which is a cloud-based platform that serves as a central hub for the codebase and all code changes, the rest of the team can review these changes and make suggestions or merge them with their own code changes.

Git is the underlying technology that makes this possible, and there are multiple services that can act as the cloud-based repository platform. Three of the most popular ones are GitHub, GitLab, and Bitbucket. Using these services, you can configure projects, organizations, and users.

Technology Spotlight

About Git

Git is the most widely used version control system. It's an open source project that was originally developed in 2005 by Linus Torvalds (yup, he created the Linux operating system kernel as well!).

While not the only version control system in existence, one of the main benefits of Git is that it can be distributed. That way, instead of only storing a full version history locally on one person's machine, Git is capable of storing a repository of every developer's working copy of the code and full history of changes. Even if you have 10 developers working remotely around the world, they can all collaborate on a single codebase and sync their changes regularly.

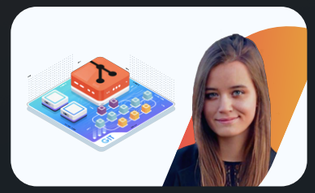
KodeKloud
The Ultimate DevOps Tools Guide For Teams

About GitHub, GitLab, and BitBucket

While Git is the version control system, GitHub, GitLab, and Bitbucket are hosting services for Git repositories. Let's say you want to create open source software for all of the world to enjoy. Instead of hosting your repository (repo) on your local machine, you can upload the repo to GitHub/GitLab/Bitbucket and make it public. Now, others from around the world can access your code and so can search engines. Organizations and individuals can also host private repositories on those platforms if they want, as it prevents them from having to deploy and maintain their own repo hosting.

Each of the three platforms offer something slightly different, but at their core they serve a similar purpose: Git repository hosting.

Want to learn more about Git?



DevOps Programming

GIT for Beginners

7 Lessons

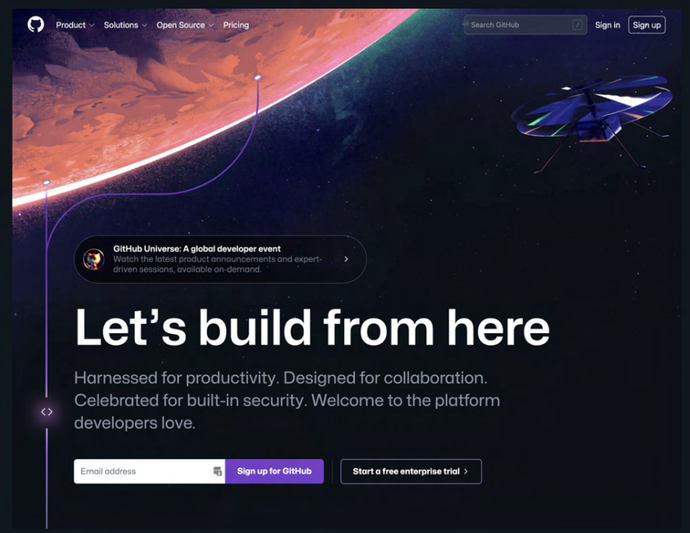
To summarize so far, Git is the command line tool and the underlying technology that enables versioning of code and collaboration between multiple developers. GitHub is the git-based publicly accessible repository of code where you push your code to. It has a web interface where you invite new developers, manage your project, manage issues with your project, add documentation to your code, etc.

Git

VS

Github

```
>_
$ git pull
```



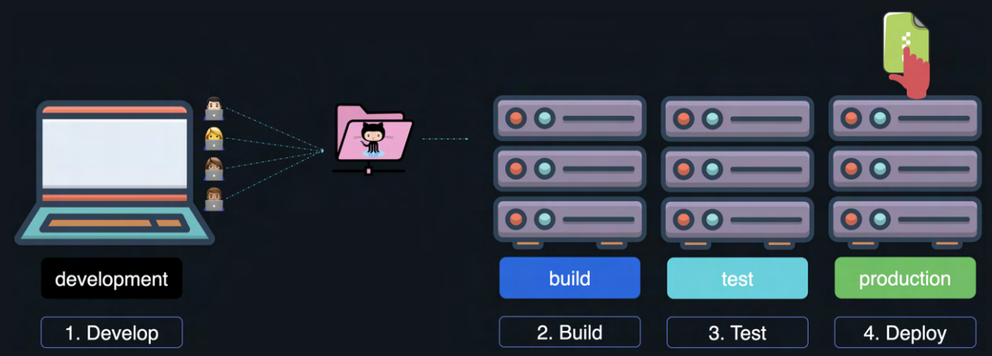
Building a development pipeline

Now with the 4 developers as well as Git & GitHub in place, the development issues are sorted.

KodeKloud
The Ultimate DevOps Tools Guide For Teams

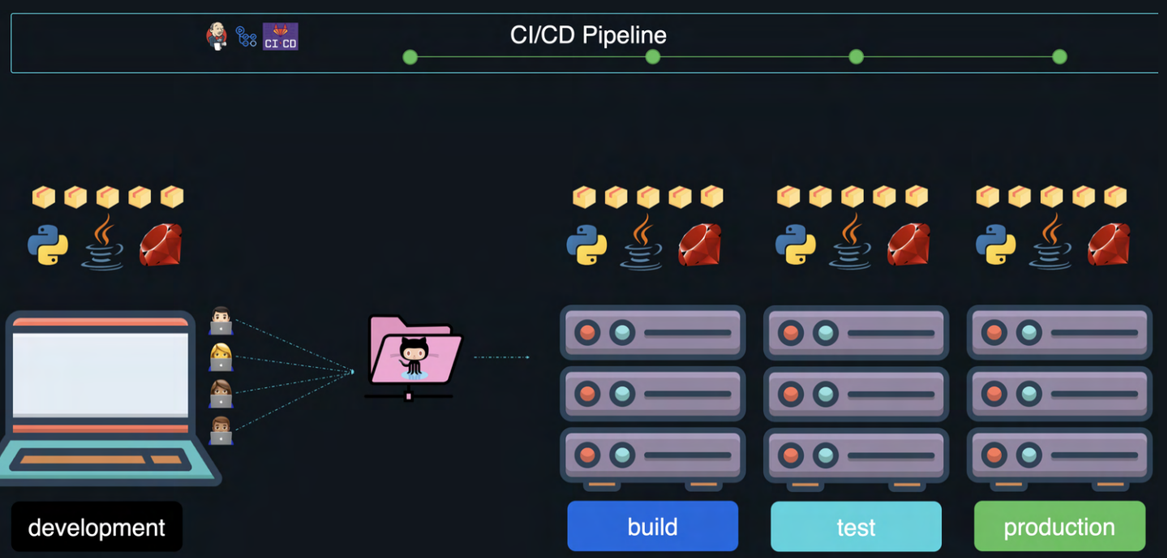
However, we still need to move code manually to the production environment every time something new is ready. The previous workflow involved developing code in your development environment, then building the code to an executable package, and then deploying it to production.

Now that we have multiple developers, the code needs to be built with the changes contributed by all of the developers. So building on the laptop itself no longer works as an individual's laptop may not have all the latest changes. It makes sense to move the build operation to a dedicated build server...that gets the latest version of the code and builds it before moving it to production.



Pushing new builds to production is risky as it might have introduced new bugs or broken something that worked before. So we need to test it in a test environment. We do that by adding a test server in between.

CI/CD pipelines



CI/CD stands for continuous integration and continuous delivery/ deployment and tools like Jenkins, GitHub actions or GitLab CI/CD help you automate these manual tasks and build a pipeline.

KodeKloud
The Ultimate DevOps Tools Guide For Teams

With one of these tools configured, every time code is pushed, it is automatically pulled from the repository to the build server and built, and then the executable is automatically moved to the test server and tested. As long as the test is successful, it's automatically moved to the production server and deployed.

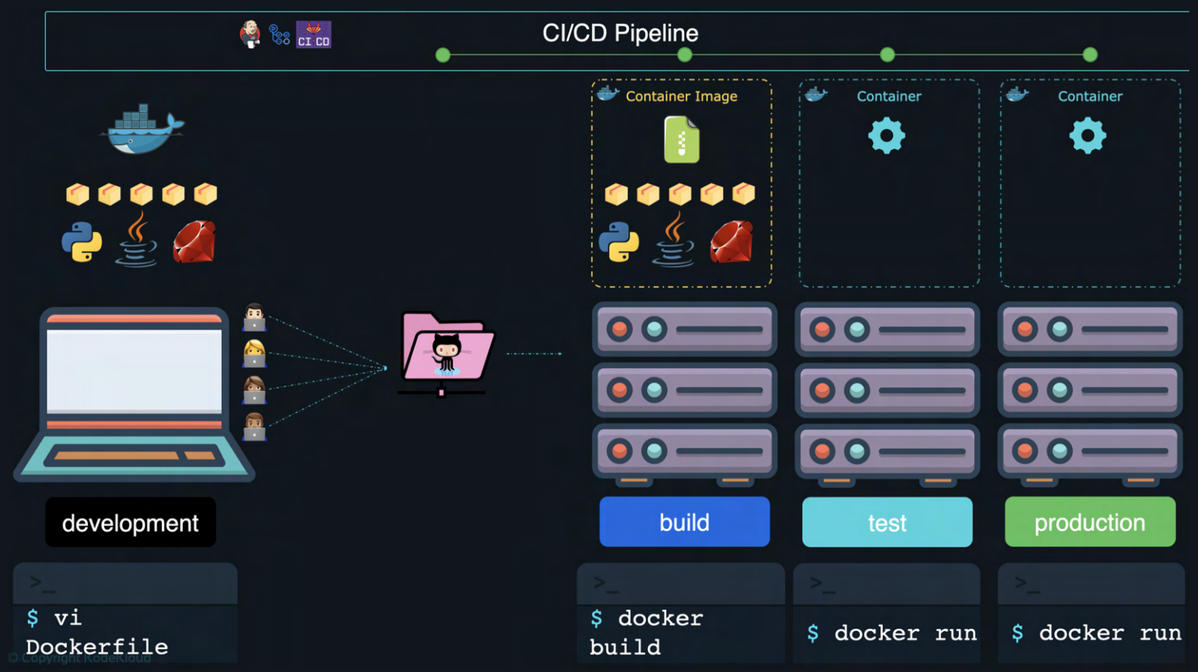
This allows changes, features and bugfixes to move faster through the pipeline and be deployed more often with fewer manual efforts, ultimately enabling you and your team to resolve issues faster, ship features more often and make your users happier.

Now with Git, GitHub/GitLab, and CI/CD pipelines in place, we have enabled our team to make changes to our application and get them to production servers seamlessly. However, it's still not all that seamless, is it? Remember the dependencies and libraries we talked about earlier? The ones that are required for the application to run on any system? These dependencies need to be configured the exact same way on the servers.

This means that if a new package is required, it needs to be manually installed and configured on all the servers that this code will run on. At the moment, this is still a manual task. If you miss out configuring one of these packages with the right version and in the right way, it will lead to the software not working and it will result in unexpected outcomes on different systems.

That's where containers come in.

Containers



KodeKloud
The Ultimate DevOps Tools Guide For Teams

Containers help package the application and its dependencies into an image that can be run on any system without worrying about dependencies.

Now, during the build stage, you build a container image with the application and its dependencies packaged into it. All other systems can simply run a container from that image without worrying about installing and configuring the libraries and dependencies.

One technology that enables working with containers is Docker. With Docker, the developer can create a Dockerfile that specifies what the dependencies are and that Dockerfile can be used during the build to build an image. That image can then be run on any server using a simple **docker run** command.

The major functionality of a container is that it enables isolation between processes. Each container is an isolated instance and this allows us to run multiple containers that all have their own separate instance of the application on the same server.

Technology Spotlight

About containers

Container technology allows developers to package and run applications in a more portable and efficient way than with just virtual machines. In traditional computing, applications are installed and run on a physical or virtual machine, which can lead to issues with dependencies and conflicts between different environments. Containers solve this problem by isolating and packaging the application and its dependencies in a lightweight, standalone environment, allowing it to run consistently across different systems. That means one developer can use Windows, while another uses Linux, while yet another uses MacOS — regardless of what the production environment is running.

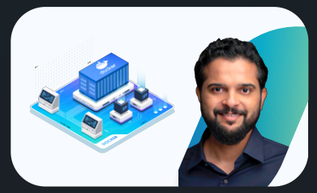
Containers achieve this by using a shared operating system kernel, which means they are more lightweight than virtual machines (VMs), and can be started up quickly and easily. Instead of waiting minutes for an application to be able to serve requests, as is the case with VMs, containers can be up and running in seconds. They're also significantly smaller. A VM could be gigabytes large, while containers could be a few megabytes in size.

KodeKloud
The Ultimate DevOps Tools Guide For Teams

About Docker

Docker is a platform for developing, shipping, and running applications using container technology. Docker provides an entire ecosystem and community of third-party products and services. As we just learned, containers are isolated units that can be used to run applications. These are very lightweight and they can be rebuilt, replaced, destroyed, and moved around easily. This means that they're particularly useful for developing, shipping, and deploying applications from start to finish, regardless of the environments and software being used. Containers package up code and all of its dependencies so that an application can run successfully from one environment to another. Docker can be used to implement all of that by helping manage our containers for us, and by giving us access to a community full of container images we can use and customize.

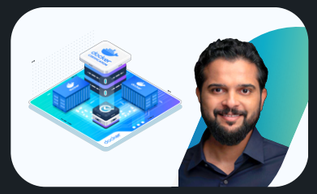
Want to learn more about Docker?



Containers DevOps

Docker Training Course for the Absolute Beginner

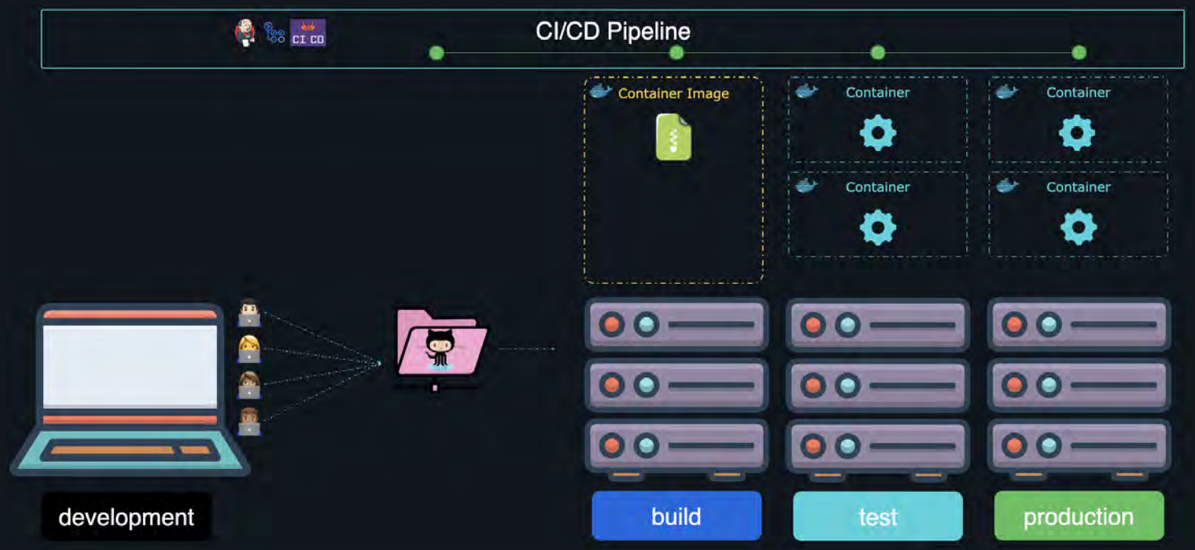
11 Lessons



Containers DevOps Certification

Docker Certified Associate Exam Course

13 Lessons

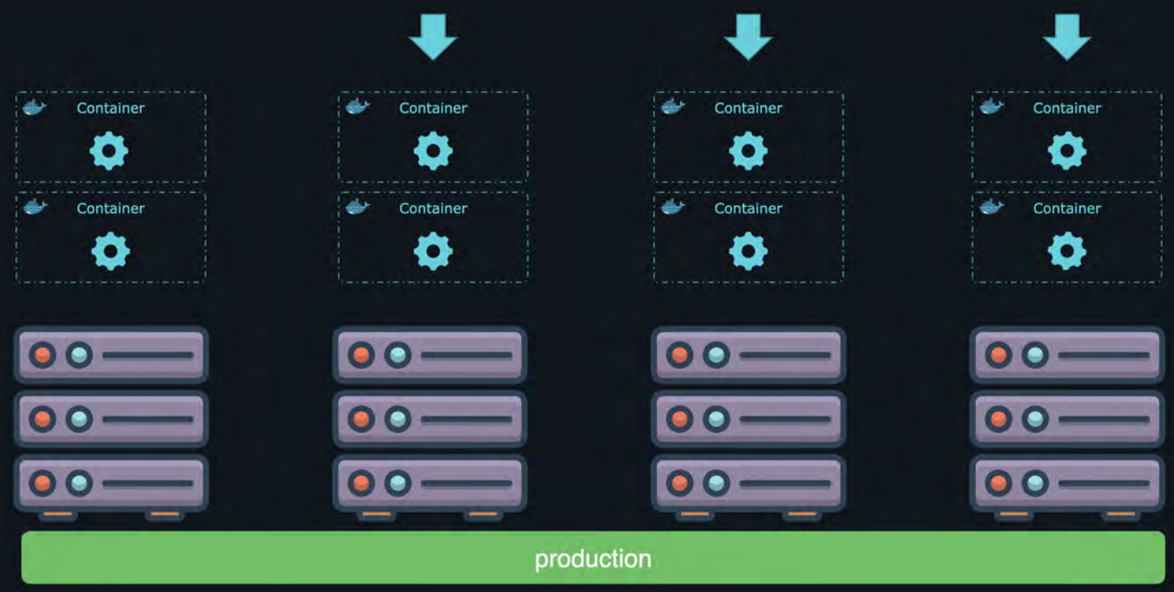


Now let's focus on the production server. Currently, we have one server, but with our number of users increasing, we want to be able to add more servers and run our application on all of them as demand increases. We can do that using containers, but how do you do that the right way so that containers are spun up when the user load increases and destroyed when load reduces? How do you ensure if a container fails that it's automatically brought back up? That's where container orchestration platforms come into play.

KodeKloud
The Ultimate DevOps Tools Guide For Teams

Container orchestration

Kubernetes is a container orchestration platform that helps declare how containers should be deployed and ensures that it always runs in the same way. It can help auto-scale containers as well as the underlying infrastructure based on the current load, and it helps manage resources on the underlying servers to ensure optimal resource utilization.



Technology Spotlight

About Kubernetes

While container technology has many benefits, it does also have its own challenges. For one, organizations can oftentimes have hundreds if not thousands of containers running either on-prem or in the cloud. On top of that, some containers can live for mere seconds, while others are meant to remain operational for a long time. Without some sort of tool, it would be impossible to properly manage all of these containers. Kubernetes was designed to help manage and orchestrate containerized applications across a cluster of nodes. Google initially designed it to help manage its very own containers at scale, which goes to show you how scalable Kubernetes (K8s) can be. Overall, it aims to provide a higher degree of automation, reliability, and scalability for organizations running containers in production.

KodeKloud
The Ultimate DevOps Tools Guide For Teams

Want to learn more about Kubernetes?

Container Orchestration · Containers · Kubernetes · DevOps

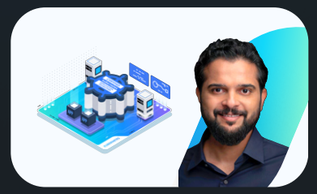
Kubernetes for the Absolute Beginners — Hands-on Tutorial

11 Lessons

Container Orchestration · Containers · DevOps

CKA Certification Course — Certified Kubernetes Administrator

15 Lessons



Containers · Kubernetes · DevOps · Certification

Certified Kubernetes Application Developer (CKAD)

13 Lessons

And more...

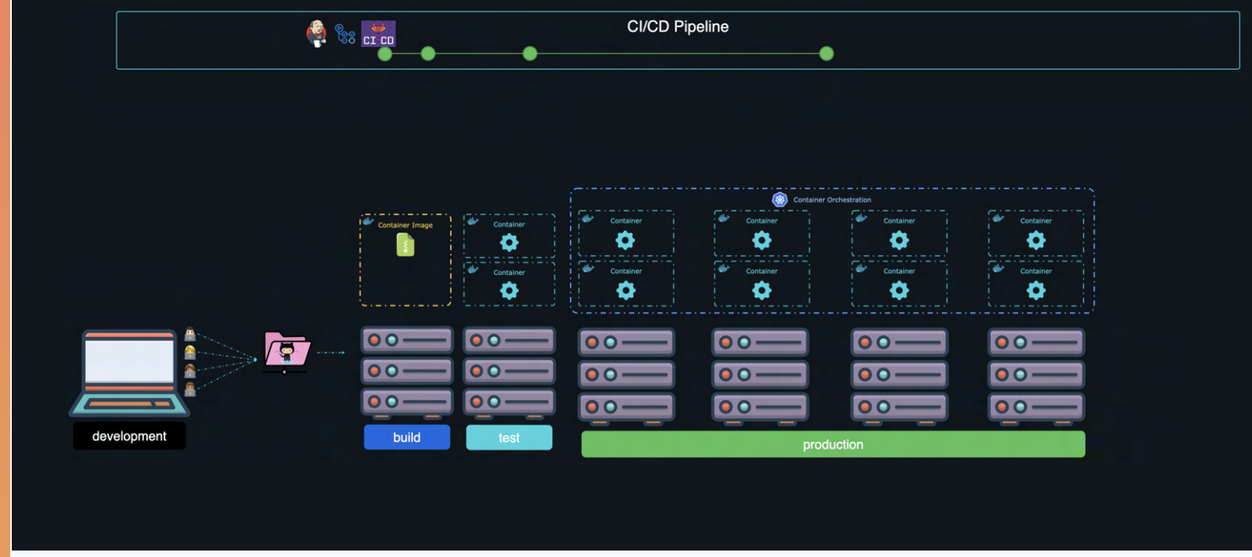
Managing infrastructure

By now, we have developers pushing code to a central GitHub repository, the CI/CD pipeline then pulls the code to the build server, and then uses Docker images to test the application in the test environment. Finally, we deploy to the production environment using containers orchestrated by Kubernetes.

The underlying infrastructure is still a big challenge (ie: the cloud instances, databases, etc...). Every time a new server is to be provisioned, it needs to be setup in the exact same way as the others in the cluster. It needs to have the right resources assigned to it, the right version of the operating system, storage attached to it, software that needs to be pre-configured on it — such as the docker runtime or the necessary Kubernetes packages — and all of this needs to have the exact same configuration every time.

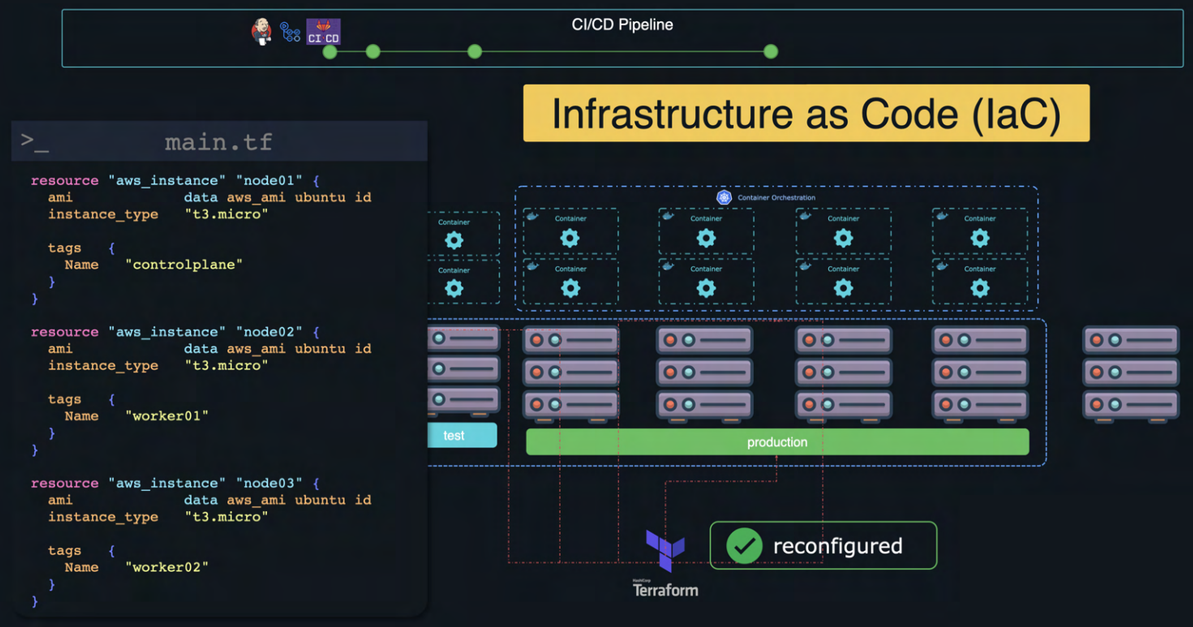
KodeKloud
The Ultimate DevOps Tools Guide For Teams

This is going to be one big challenge if you have to click through the cloud platform's GUI each time a server needs to be provisioned. This can take a lot of time and can lead to human errors in misconfiguring infrastructure resulting in having to rebuild the entire server. To automate the provisioning of these servers, we can use Infrastructure as Code (IaC) tools like Terraform.



Terraform automates the provisioning and configuration of servers irrespective of what cloud platforms they are on. It ensures that the servers configured are always in the same state by preventing drift. For example, if someone manually changes a configuration on these servers and not through Terraform, it can change it back to make sure the state defined is preserved.

The state is defined by configuring a Terraform manifest file that looks like this:



This is a snippet from a Terraform manifest file that has the list of servers and their required configuration. This is why it's called Infrastructure as Code — we are describing the infrastructure using actual lines of code. All of the infrastructure configurations including the virtual machines, storage buckets, VPC etc... are now stored in the form of code and are stored in source code repositories. We can track our infrastructure through version control just like we would with any other code.

If any infrastructure changes are required, we can make changes to our code (the manifest files) and then run the command terraform apply. It will automatically inform you of the changes it's going to make, and then take care of the rest for you. Almost like magic.

KodeKloud
The Ultimate DevOps Tools Guide For Teams

Technology Spotlight

About Terraform

Terraform is one of the leading tools for Infrastructure as Code (IaC), and IaC has helped transform the way that we deploy infrastructure. Say, for example, that you need to deploy resources in the AWS cloud: you need EC2 instances, RDS databases, S3 buckets, security resources and configurations, and an entire VPC with subnets to house many of these resources. You don't yet know about IaC, and so you log into the AWS console and manually create and configure every single one of those resources.

One day in the future, you're asked to replicate that exact environment in a separate AWS account to use as a staging environment. You have to go back through every single resource to see how you configured them. Oh, and while you're doing that, you realize that someone deleted an important service, but you can't remember how it was configured because you forgot to document it. What a nightmare! Instead, you come across KodeKloud and you learn about IaC and Terraform, and you realize that you should be building and managing all of your infrastructure with a tool like Terraform.

Terraform lets you create those environments in the cloud using code. Actual lines of text that you then store in files and keep track of in your Git repos. You describe what resources you need, how you need them configured, and Terraform will automatically provision them for you. Need to make a quick update to an EC2 instance? No problem, write those updates in Terraform, have them peer reviewed in the Git repo, and deploy those changes. That's the power of IaC technology like Terraform. There are alternatives to Terraform, by the way. Tools like Amazon CloudFormation, Azure Resource Manager (ARM), Pulumi, and more.

KodeKloud
The Ultimate DevOps Tools Guide For Teams

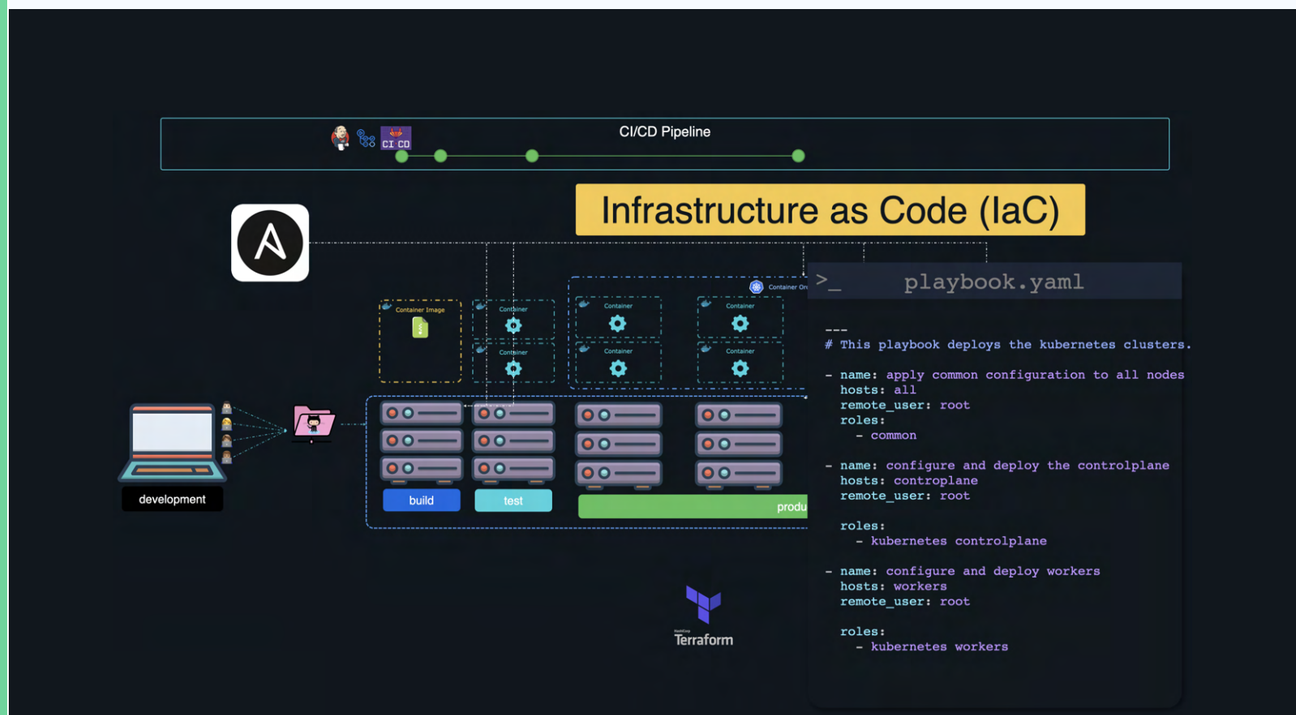
Server automation

Once the servers are provisioned using an IaC tool, the configuration of those servers can be automated using a tool like Ansible.

While Terraform is more of an infrastructure provisioning tool, Ansible is an automation tool that helps configure this infrastructure once provisioned. There are many areas that both Terraform and Ansible overlap – both of these tools can provision infrastructure and automate software configuration on them, but each has its own benefits.

KodeKloud
The Ultimate DevOps Tools Guide For Teams

While Terraform is used mostly for provisioning and de-provisioning infrastructure, Ansible is used mostly for post-configuration activities such as installing software and configuration. For example, Terraform can spin up an EC2 instance in AWS, and then Ansible can create an admin user, a webserver user, a sysadmin user, etc... all with correct permissions. It can then install that webserver software (like Nginx or Apache), and the list goes on.



Similar to Terraform, Ansible also uses code to configure servers. That code is setup in what's called Ansible Playbooks. This code also gets pushed to a source code repository in GitHub.

Technology Spotlight

About Ansible

Ansible is somewhat similar to Terraform, and the two are often compared. Ansible is an IT automation tool and it can be used to configure systems, deploy software, and orchestrate IT tasks like deployments and updates. As we explained in the story scenario, Terraform is often used with Ansible to provision infrastructure, and then Ansible is responsible for configuring the infrastructure. While Terraform might help spin up an EC2 instance, Ansible will help make sure that the correct software is installed on that instance. Need a webserver like nginx running on the instance? Ansible will install that. Need to push updates to all of your active servers? Ansible can make sure the updates successfully install, and roll back if not.

KodeKloud
The Ultimate DevOps Tools Guide For Teams

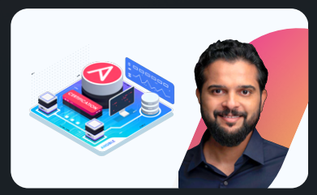
Ansible is also great at keeping resources in specific states, which means that if a SysAdmin were to try and make a manual modification to one of those resources — like update a configuration file — Ansible would detect this change and revert back, preventing drift. It also uses code with YAML files, which means that its files can be checked into a Git repo, just like with Terraform files.

Want to learn more about Ansible?

Containers DevOps

Learn Ansible Basics — Beginners Course

5 Lessons



Containers DevOps Certification

Ansible Advanced Course

9 Lessons

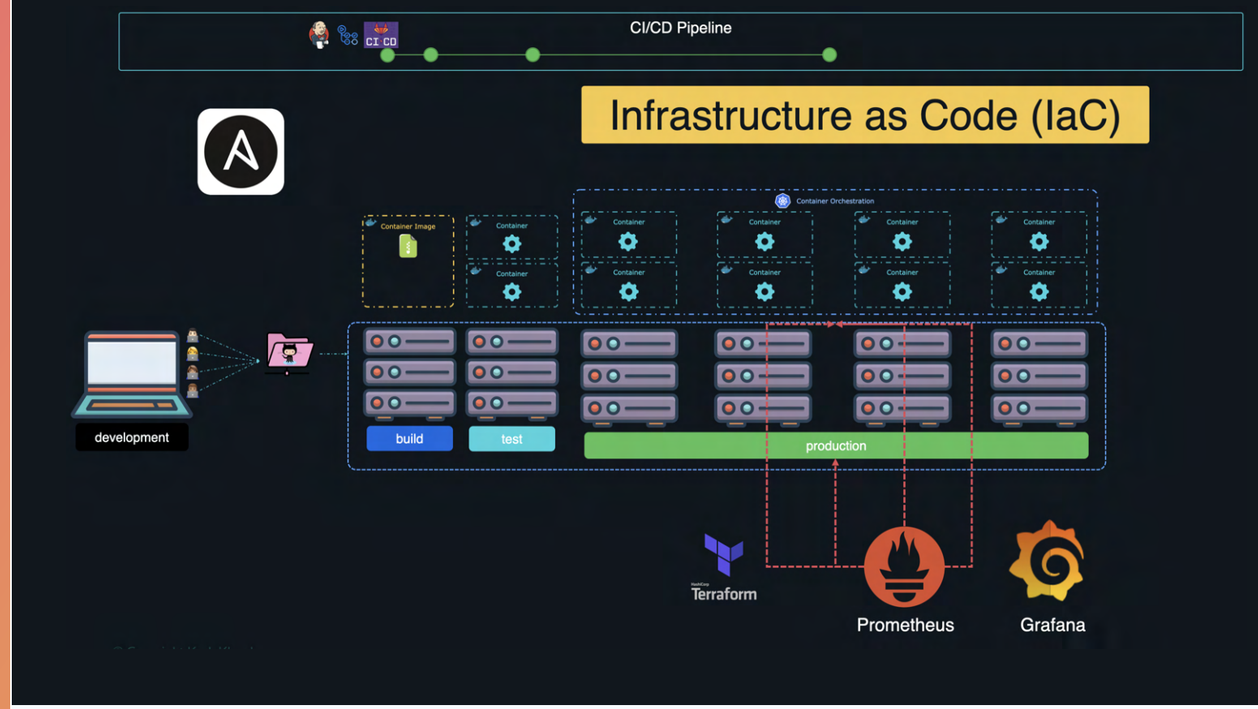
Operating and Monitoring

Provisioning is not all, we now need to maintain everything as well. We want to be able to monitor servers and apps to take preventive measures.

We want to be able to see the CPU utilization on these servers, the memory consumption, monitor the processes, identify what processes are causing higher consumption, etc. That's where tools like Prometheus come in. Prometheus collects metrics from the different servers and containers. and stores them centrally.

Not only do we want to collect metrics, but we also want to be able to visualize them graphically. That's where tools like Grafana come in.

KodeKloud
The Ultimate DevOps Tools Guide For Teams



Grafana helps make sense of the data collected by Prometheus by visualizing it through charts and graphs.

Technology Spotlight

About Prometheus

Prometheus is a tool used for monitoring and alerting. It's helpful for keeping an eye on the health and performance of servers, containers, and applications.

KodeKloud
The Ultimate DevOps Tools Guide For Teams

Prometheus works by regularly collecting data about the system or application it is monitoring, such as CPU usage, memory consumption, network traffic, and other metrics. This data is then stored in a time-series database, allowing Prometheus to analyze and visualize the data over time. It also has a built-in alerting system that can notify when certain thresholds or conditions are met, such as when CPU usage exceeds a certain level or when a specific error occurs.

Not only is it a great option for monitoring static servers, but it's also great at monitoring highly dynamic micro-services that may be short-lived.

Want to learn more about Prometheus?

Certification Kubernetes Monitoring

Prometheus Certified Associate (PCA)

11 Lessons

Technology Spotlight

About Grafana

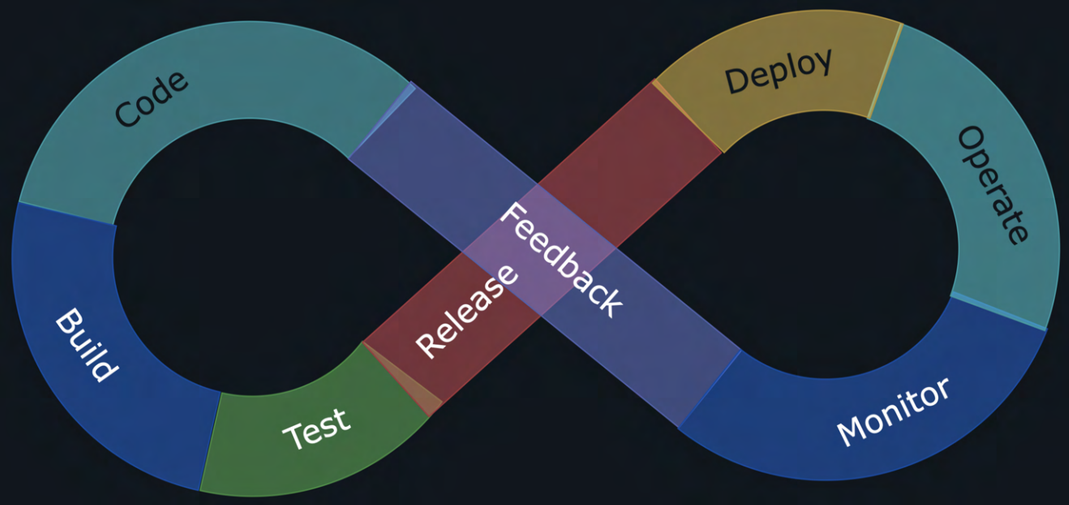
Grafana is a tool that can also be used for monitoring and visualization. It works by connecting to various data sources, such as databases, APIs, or other monitoring tools like Prometheus, and displaying the data in visually appealing and easy-to-understand dashboards. These dashboards can display data in the form of charts, graphs, tables, and other visualizations, making it easier for users to identify trends and patterns.

KodeKloud
The Ultimate DevOps Tools Guide For Teams

Grafana is especially helpful for teams that need to monitor and analyze large amounts of data from multiple sources. With its powerful querying language and flexible data visualization options, it allows users to gain a comprehensive understanding of system and application performance, identify potential issues, and make data-driven decisions.

Overall, Prometheus and Grafana can often be used together to provide monitoring, alerting, and visualization. This is a vital part of DevOps, because it helps increase application performance, optimize operations, and troubleshoot issues (ideally even before they become critical).

Tying it all together



Everything that we just walked through helps us go from a simple idea, to building it, deploying it and getting it out to our users quickly. It doesn't stop there. We can now get feedback from our users, review that feedback, brainstorm, come up with new features and implement those ideas. This process is now a breeze compared to the alternative.

KodeKloud
The Ultimate DevOps Tools Guide For Teams

Any code pushed will now go through the pipeline we have defined. It will automatically get built and tested so that it can be ready to deploy multiple times a day instead of once a month.

Once deployed, our changes are monitored and we can collect more feedback from our users. This cycle repeats indefinitely.

Fundamentally, that's what DevOps is all about. DevOps is a combination of people, processes and tools that work together in going from idea to execution and delivering high-quality software consistently.

Implementing these tools and processes, however, is easier said than done. If you're reading this, it's probably because your organization is thinking about or is in the midst of shifting to using modern DevOps practices and tools. That means many of the engineers on your team(s) are going to have to upskill and reskill in order to be able to do that. That's what KodeKloud specializes in and can help you with. If any part of what you just read in this Ebook could help your organization, reach out to our team below for a free demo!

KodeKloud

About KodeKloud For Business

Get a Free Demo

From $25/user
/month

Take your team to the next level with KodeKloud's immersive DevOps training. Access in-depth courses, hands-on labs, cloud playgrounds, and business-only features.

60+
Courses

400+
Hands-on Labs

50+
DevOps & Cloud Playgrounds

Get your team certified.
Need your team certified to show clients, leadership, or partners proficiency? We'll not only get them certified quickly, but we'll help them build practical skills in the process.

Build knowledge and practical skills.
Help your team tackle whatever DevOps projects you or your clients throw at them with interactive, hands-on, and practical learning.

Upskill and reskill to keep up with tech innovation.
Need to deploy new DevOps tools in your tech stack, but don't have the internal expertise? Upskill and reskill your team to tackle today's and tomorrow's most innovative tech

When you have access to a platform like KodeKloud, upskilling and reskilling your existing engineers is a no-brainer, because we can help your team close their skills gap effectively, which saves you significant time and money. See exactly how we've helped other organizations close their DevOps and Cloud skills gaps by booking a free demo.

Get a Free Demo

KodeKloud

Success Stories

With over 1,000,000 learners, we've helped everyone from individuals to Fortune 500 companies. See what they have to say about our training and platform:

Swetha D.

I am happy to share that I'm now a Certified Kubernetes Administrator (CKA). This was a challenging certification due to its hands-on nature. Thanks to Mumshad Mannambeth and the KodeKloud team for the well-structured course and great labs.

LinkedIn

Ramkumar Nagarajan

I'm happy to share that I've been certified as a Certified Kubernetes Security Specialist (CKS).This also marks the successful completion of the Kubernetes triplet CK[A, AD, S].CKS is more challenging and time-bound exam compared to its peer CKA.Thanks to Mumshad Mannambeth for an excellent course on KodeKloud.

LinkedIn

Saurabh Atirek

I am happy to share that I have successfully cleared CKAD certification. I would like to thank Mumshad Mannambeth for the amazing course and hands on labs at KodeKloud which helped me learn the basics of #kubernetes.

LinkedIn

Shereen Abdelbaky
Student KodeKloud

I studied KodeKloud for the last three years. I learned a lot from KodeKloud. Actually, these are amusing courses and developed me to become a good DevOps Engineer. Now I am working as a DevOps Engineer in an organization. And I want to keep in touch with KodeKloud to learn more. I wait for new courses to learn something new. Thanks to the whole KodeKloud team for launching such great courses.

Aneek Bera
Student KodeKloud

KodeKloud's exemplary course structure and practice tests build a strong bridge between understanding the concepts and their implementation. It's detailed, vast, and in-depth. Most amazingly the complicated concepts have been crushed into finer bits for easy digestion. A must platform for anyone wanting to self-pace and learn stuff. Highly recommended for all age groups.

Gokul Ravichandran
Student KodeKloud

I was quite confused about how to move from Linux to DevOps, don't know where to start that's when I found KodeKloud, spent 6 months on their beginner courses for DevOps (Their Roadmap to DevOps was an eye-opener for me), Everything changed since then Now I am working as a DevOps Engineer. My payscale was raised by 112% because of them. It would have been a long road if not for KodeKloud. Forever Grateful!

Is your team next?