What exactly is DevOps?

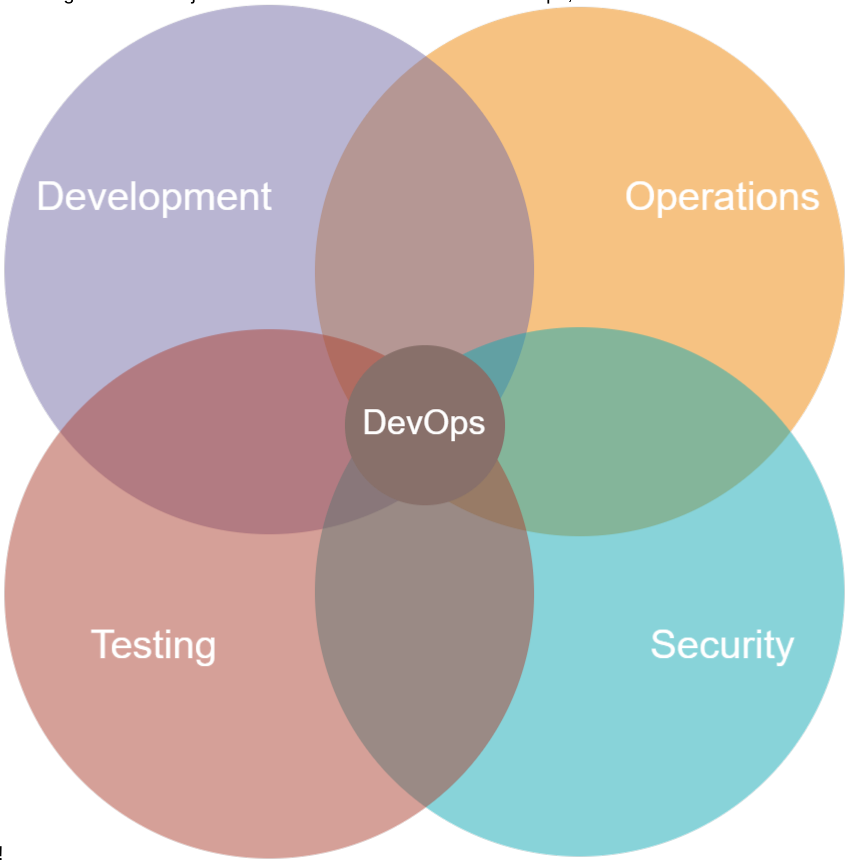
Draft notes

- DevOps Definition
  The simple definition of the intersection
- Traditional Application Release Process
  Here, we explain what is the release process i.e. the process during which an application is coded, tested, packaged, and deployed

  What I would like to include:
  1. Built & packaged = app in some kind of executable form so that it can run
  2. Deploy it: before you have to configure the server in which you plan to deploy the app = install dependencies, deploy the app, configure firewall to allow secure access to the server
  3. This is the initial release process, et c'est pas fini ! While in use, you have to check-in on your application. Is everything running fine? Are there any bugs that you might have missed during development and testing? Can the application handle high user load? is it still accessible? All of these questions are answered by a good observability system constantly running after deployment. Et c'est pas fini !
  4. As the app is being used, you get feedback by users to add improvements: new features, bug fixes, performance optimizations... So you have to update your app, and this new version of the app has to go through the same release process. and this done again and again and again.

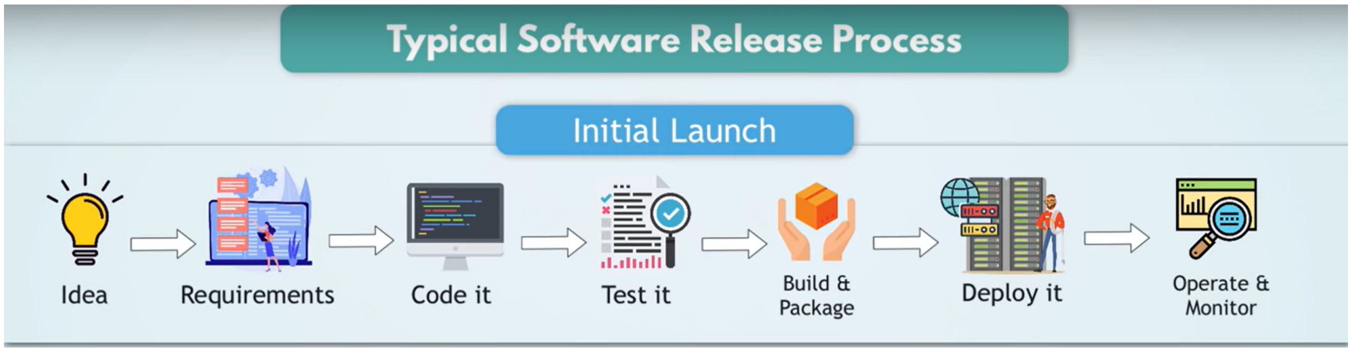
  This the process of release and continuous update (or continuous delivery 😉) that every successful app goes through. DevOps is about making this process of continuous delivery fast and with minimal errors and bugs.

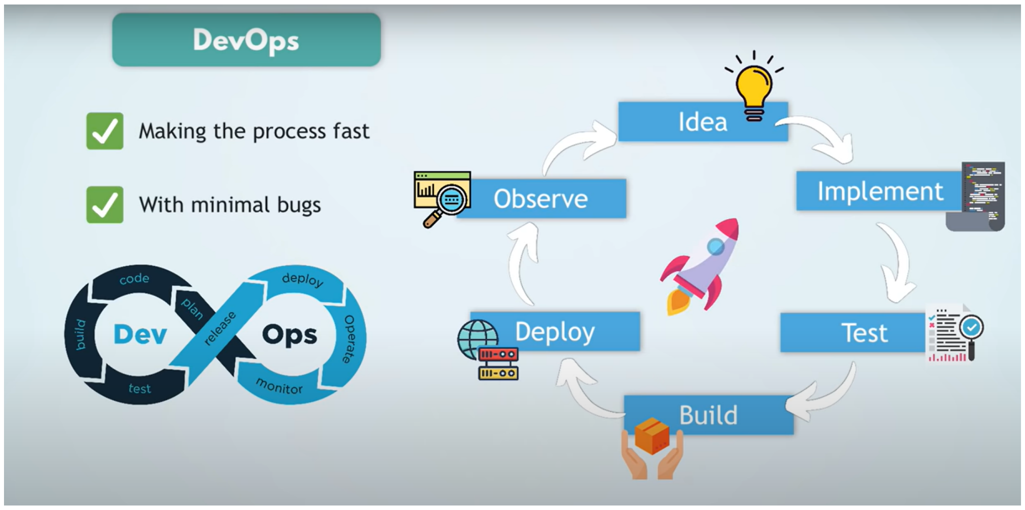
- What were the challenges that teams had to face during this release process, and that DevOps tries to solve
  During this continuous integration continuous delivery process, many problems and roadblocks arise making it slow, painful and well... not really 'continuous...':
  1. Miscommunication: No clearly defined process of handover of the tested app from Devs to operations. This often results in operations receiving a product that is not deployment ready, or poorly documented on the part needed for deployment, causing the app getting shipped back to devs and so on and so forth. Extending the time to market by weeks or even months.
  2. Conflict of interests: Devs want to push out new features fast, while operations want to make sure that these new features won't break anything. Operations naturally slow down the delivery because they have to make sure the app will continue to be user ready.
  3. Security: Same as problem two, but now the show-slower is the security team.
  4. Application testing: separate teams for testers and Devs. Some of these tests are done manually Slowing again the process
  5. Manual work in general: Manually deploying the app, Manually preparing the deployment environment, Manually configuring user access & permissions Slow, more error-prone, hard to trace (who executed what when) more difficult to recover and replicate exact infrastructure state...
  DevOps is here to remove all these roadblocks making it more than just the intersection between Dev and Ops, but also the intersection with all



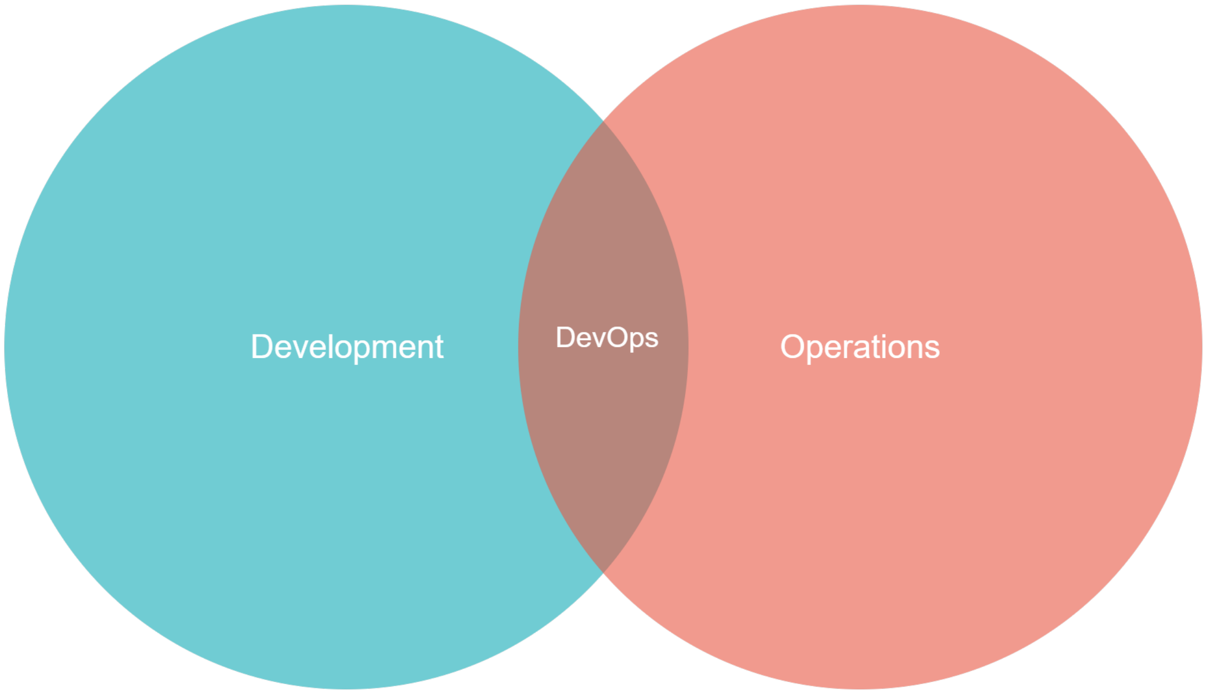
  the teams involved during the release process!
  - DevOps in Practice - Become a DevOps Engineer
    1. Learn how the developers work, meaning understanding the git workflow they're using, how the application is configured to talk to other services or dbs, as well as concepts of automated testing.
    2. The app needs to be deployed on infrastructure made of servers (either on premise or in the cloud). As DevOps engineer, you may be responsible for preparing the infrastructure to run the application. Since these servers are usually Linux servers, you need to know Linux

basics and be comfortable using the cli: knowing how to install tools via the cli, understanding Linux file system, knowing how to ssh into a server, basics of how to administer a server etc...
3. Basics of networking and security to configure firewalls to secure the application, but also to open some ports to make the application accessible from outside, as well as understand how IP addresses, ports and DNS works. /!\ You only need the basics, sys admins, network and security engineers are professions of their own! Know just enough to be able to be able to prepare the server to run your app.
4. Containers: Containers are becoming the defacto software packaging model: you will probably be running your app as a container in your server
With step 1 to 4, you are able to automate the tests of an app. You are able to package into a Docker container and you are also able to prepare the infrastructure on which it will be running. How to get the app from dev teams to the servers so that it is indeed available for end users?
HERE IS WHERE THE MAIN RESPONSIBILITIES OF A DEVOPS ENGINEER COME IN How to do this deployment in a continuous, fast and automated way.
5. Once the app (or the new patch, feature, bug fix) is tested, we need to package the app as an artifact (a jar file, a zip, a tarball etc...) so that we can deploy it. That's where build tools & Package Manager tools come in (Some of the examples are Maven and Gradle for java app, npm for javascript etc...) you need to understand how this process of packaging applications work.
6. Then you will probably be building a docker images from you application (a docker image where you'll put your app artifact, and installed all dependencies). You want to keep trace of you docker images which are essentially the different versions of your app: to do that you'll push them to an artifact repository, so you'll have to learn how to manage that as well (dockerhub for instance).
7. Of course, you don't want to do this manually, you want instead one pipeline that does all of this in sequential steps. So you need build automation: One of the most popular build automation tools is Jenkins (other honorable mentions are gitlab and github actions).
8. Of course you need to connect this pipeline with the git repository to get the code, so this is part of continuous integration: code changes from the code repo get continuously tested. You want to deploy that new change to the server after its tested built and packages which is part of the continuous deployment process.
9. There could be other steps to the pipeline such as sending teams notifications about the pipeline state, or handling build failures etc...
10. Let's go back to the infrastructure for a minute: nowadays, most of companies use cloud infrastructure instead of on premise. You'll need to learn for at least one cloud provider, how to configure servers on it (basically steps two through four)
11. Container Orchestration: Kubernetes.
12. Monitoring: Now that you have these 1000s of containers running in k8s on 100s of servers, how do you track performance of your individual applications or whether everything runs successfully? You need to setup monitoring for your running application, the underlying k8s cluster and the servers on which the cluster is running. You need to know a monitoring tool such as Prometheus or Nagios for example.
13. Fiouh, we build our production environment where code changes are continuously integrated and deployed to the infrastructure in the cloud. Mais c'est pas fini ! We need that same environment multiple times (one for testing and another one for development!). Automating the process of creating the infrastructure as well as configuring it to run and deploy your app can be done using two types of infrastructure as code tools: Infrastructure provisioning tool like terraform, and configuration management tool like Ansible, Chef or Puppet.
14. Finally a scripting language for automating various tasks (Python, Bash, or GO)





In simple terms, DevOps is a field born of the intersection between development and Operations. But what are the exact boundaries of DevOps? Which part of Dev is not DevOps and which part of Ops is not DevOps?



Tradition

Development and operations are two main components in the whole application release process. So let's look at this delivering process from the beginning.

Whenever we are developing an application, we always have one goal: delivering the application to the end users.