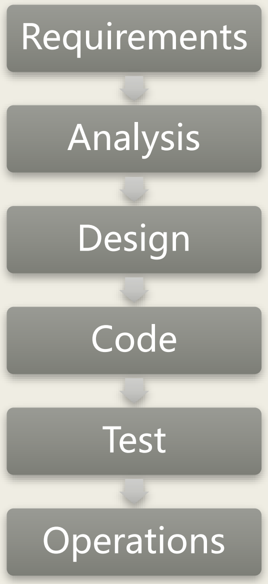
INTRODUCTION TO DEVOPS

Software development methodologies

- In order to develop a piece of software application, the responsible team must follow a set of rules that guarantee an easy and successful development process.
- Throughout software development history, many methodologies came to existence. Examples of which are Waterfall, Prototyping, Spiral, Rapid, Agile among others.
- Choosing one way or another for the next project depends on the type of this project rather than whether or not an approach is "absolutely" better than the other.
- For the sake of this course, we'll have a quick look at Agile development. To give you a better idea, we'll start by showing you a contrasting approach, the Waterfall.

The Waterfall approach

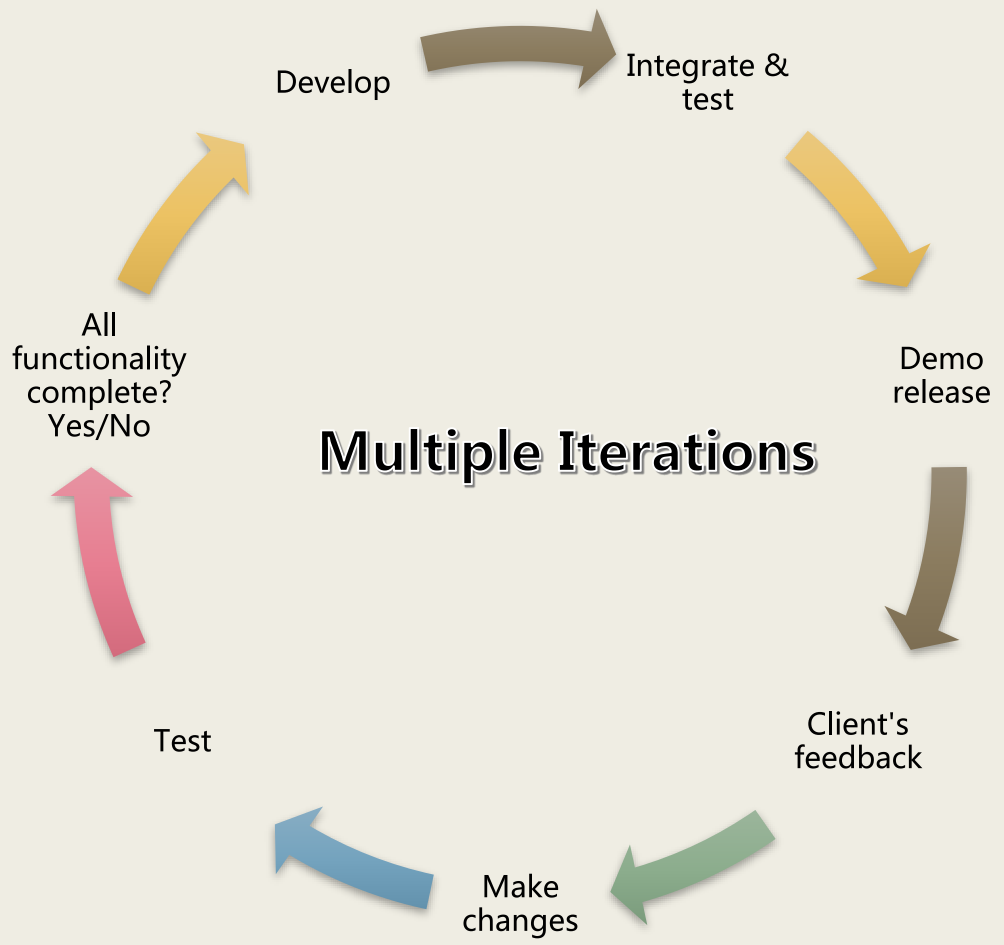
- As the name suggests, the development process in this approach "falls" downwards through a set of stages or phases until it reaches the end point.



- The process starts by gathering the requirements from the client. This gets added to a "product requirements document".
- Those requirements get analyzed. Models and business rules are created as a result.
- Developers start to write code for build, test, and integration.
- Regular discovery and debugging of any flaws.
- Finally deployment (installation/migration), support and maintenance of the application is carries out.

- The important point to notice here is that once each step is completed, it is very hard (if not impossible) to return back to it.
- Such an approach becomes useful when the client won't change the scope of the project once it started, and when speed is not as an issue as clearly defining the project.

The Agile approach



- In this model, the project is built through multiple, short iterations, each will take a week or a month.
- Several versions of the end product are built, from the least functional to the most one.
- Any changes, customer feedback, or additional requirements are accepted and adopted in any stage (there can be always a second iteration).
- When the end product reaches the point where no more changes could be made, and no more functionality is needed, it is released to the market.
- Of course Agile development is not the answer to all development problems. However, it may suit most of the modern project requirements.

Infrastructure needs for Agile development

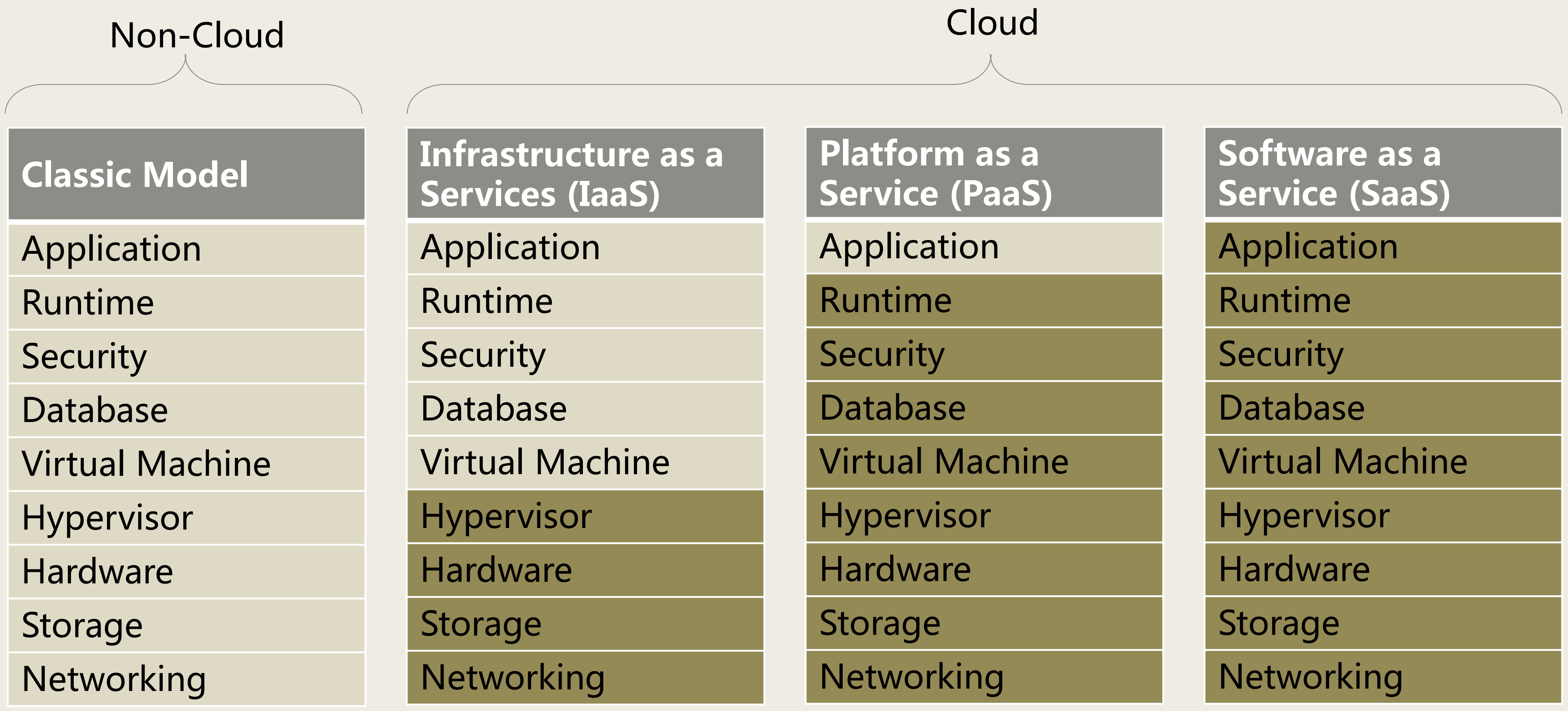
- In Agile methodology, the product is coded in "sprints", each of which takes two to four weeks at most. This required multiple development environments created quickly and easily.
- Long time ago, to create a new environment, the developer would request one hardware server, and the operations team starts availing the required prerequisites (database, webserver, libraries…etc.)
- Few years ago, this was done through virtualization. The ability to make one powerful machine host a number of "virtual" machines each having its own set of resources. This made it much faster to avail more than one machine to the developer and, thus, creating multiple environments in a much quicker way.
- Moreover, there's still the need to avail the required prerequisites, which consumed time and involved other teams who may not always be available for this task.
- A modern approach for this is cloud computing.

A web application project – The traditional way

- A quick example goes as follows: you are a web developer building a web application for a customer. You need a web server equipped with all your requisites (Apache, MySQL database, PHP 5.5). In a traditional model you (or whoever the responsible) will have to do the following:
  - Assuming that the host machine is already set up (and purchased), create a new virtual machine with the required resources, install the operating system, configure the necessary security and create the appropriate user accounts, install Apache, MySQL, and PHP 5.5, then start deploying your application. Should you need a new environment, you will have to repeat the previous steps, or ask the responsible team to do so. Moreover, if you need more CPU and/or memory to run a particular test for a specific period of time, this will be a lengthy process that will incur a downtime.

A web application project – the cloud-computing way

- In a cloud computing environment, you don’t have even purchase the hardware on which you will work. All what you have to do is setup an account on the cloud computing vendors (like Amazon AWS, Google Apps…etc.) and choose the model that best suits you: IaaS (Infrastructure as a Service), PaaS (Platform as a Service), or SaaS (Software as a Service).
- Setting up the machine is as easy as making a few mouse clicks and it takes minutes instead of hours or days in the traditional model.
- You can also host your own cloud environment.
- In all cases development speed is increased, and this nicely fits the Agile model.



Non-Cloud	Cloud		
Classic Model	**Infrastructure as a Services (IaaS)**	**Platform as a Service (PaaS)**	**Software as a Service (SaaS)**
Application	Application	Application	Application
Runtime	Runtime	Runtime	Runtime
Security	Security	Security	Security
Database	Database	Database	Database
Virtual Machine	Virtual Machine	Virtual Machine	Virtual Machine
Hypervisor	Hypervisor	Hypervisor	Hypervisor
Hardware	Hardware	Hardware	Hardware
Storage	Storage	Storage	Storage
Networking	Networking	Networking	Networking

Where does DevOps fit in?

- DevOps is a combination between "Development" and "Operations"
- "Development" implies software developers
- "Operations" implies other IT departments like system administration, quality assurance and so on.
- It complements cloud computing and the Agile development model
- It aims at facilitating and, thus, shortening the application life cycle by automating the delivery and infrastructure changes.
- Before DevOps, communication between functionally-separated organizations was more difficult and time consuming.

DevOps misconceptions

- With the emergence of Agile development, cloud computing then DevOps, more and more emphasis is placed on the developers job. Now the development team does their job in addition to the operations', which may further imply that system administration and operations staff are losing their jobs. This is entirely untrue.
- As a matter of fact, DevOps was initiated and is supported by the operations team in the first place. They realized that their existing practices cannot keep pace with the current, fast-moving climate. More and more parts of operations need to be automated. Additionally, some of the operational tasks need to be carried out by developers.
- The outcome of this collaboration is a person who has both operation and development skills. That is a DevOps engineer.

Tools of the trade

- DevOps is not a tool in itself; it is rather a set of tools that may be categorized as one or more of the following
  - Continuous Integration (CI): like Jenkins
  - Containerization: like Docker, Chroot, and VMware ThinApp
  - Version Control: like Git
  - Testing: like Selenium
  - Packaging: like dpkg, rpm…etc.
  - Configuration management: like Puppet and Chef