**What are the benefits of DevOps, what are the problems that it solves, what is Dev vs Ops?**

In a non-DevOps workplace then there are normally two teams; Development – which is a team focused on producing new features for existing products, and the Operations team – which is a team focused on the maintaining the currently deployed systems. Often this could cause conflicts between the two teams on a product for two reasons. The first is that once the Dev team has finished a task it will often be sent to the Ops team who aren’t familiar with this new code and will have to learn what it does and how it integrates with the whole system solely from the documentation given. This will cause large delays in getting the new features out to the customer, as well as making the code a lot harder to maintain for the Ops team. The second reason is that often the new features from the Dev team can create problems with the rest of the project, causing yet more work for the Ops team to integrate the new feature.

In a DevOps workplace these problems are solved along with several others. The Developer and Operations teams in a DevOps workplace become one team as the name suggests. This instantly removes any conflicts that there would be between the two teams as outlined above. There are also other benefits of having a DevOps workplace such as teams knowing what each bit of the code does, more frequent updates being passed on to the customer, and faster recovery times when something does happen to go wrong. Not only are there benefits for the end product being delivered, there are also benefits for the people delivering the solution too. DevOps teams are in general happier, more productive teams than non-DevOps teams, and they also have a larger amount of development opportunities too.

The first tool that we will look at that is commonly used in the DevOps workplace is Gradle. Created in 2007 as follow on from the XML based Ant and Maven from Apache, Gradle has become one of the most popular tools available and is even chosen by google as an official build tool for Android Studio. Gradle was originally designed with large, multi-project, builds in mind and has support for incremental builds by keeping track of the build tree, avoiding re-executing older parts that are not required again making it up to 100 times faster than Maven. Gradle is mainly focused on Java, Groovy, and Scala development, but that doesn’t mean that there is no support for other languages, in fact it is very much the opposite with support for C++ and Python among others.

The second tool that we will be looking at is Git. Developed by Linus Torvalds and originally released in 2005 for Linux, Git is one of the most widely used tools in the software development world today. Git is a version control system allowing one or many users to keep track of many versions of the same files, easily allowing the user to roll back to a previous version if need be. What makes Git so popular though is the integration with tools such as GitHub. This allows many users in a team to host work in the same place, seeing all updates to each part of the project and allowing rollbacks at the same time.

The final tool that we will be looking at is Kubernetes. Kubernetes is an open source system released in 2015 by Google engineers looking for a solution to manage containers at a large scale by grouping your containers into logical units. Kubernetes allows the automation of the management of large numbers of containers and the distribution and scheduling of containers across the whole cluster. IF for example a worker node goes down then Kubernetes will redistribute the container to where it is needed.