FACULTY OF INFORMATION COMMUNICATION TECHNOLOGY

Department of Computer Science

Rethabile Moahloli

Student No: 213159526

Email: moahloretha@gmail.com

Submitted in fulfilment of the requirements for the subject

ADH401T

Tshwane University of Technology

SUPERVISOR (s): Mr Litsietsi Montsi

EMAIL: adh401tut@gmail.com

Question 2:

Digibank Continuous Delivery Pipeline Design

When designing a continuous delivery pipeline for Digibank, the main focus is to improve the efficiency, quality, and speed to market within the software development world.

I would like to really focus on breaking down the different stages of the process. This way you can get the basics and hit the ground running with a delivery pipeline in your organization. The delivery pipeline can be broken down into a few major stages, as mentioned below.

1. Source code control (for the purpose of management)
2. Build automation
3. Unit test automation
4. Deployment automation
5. Monitoring

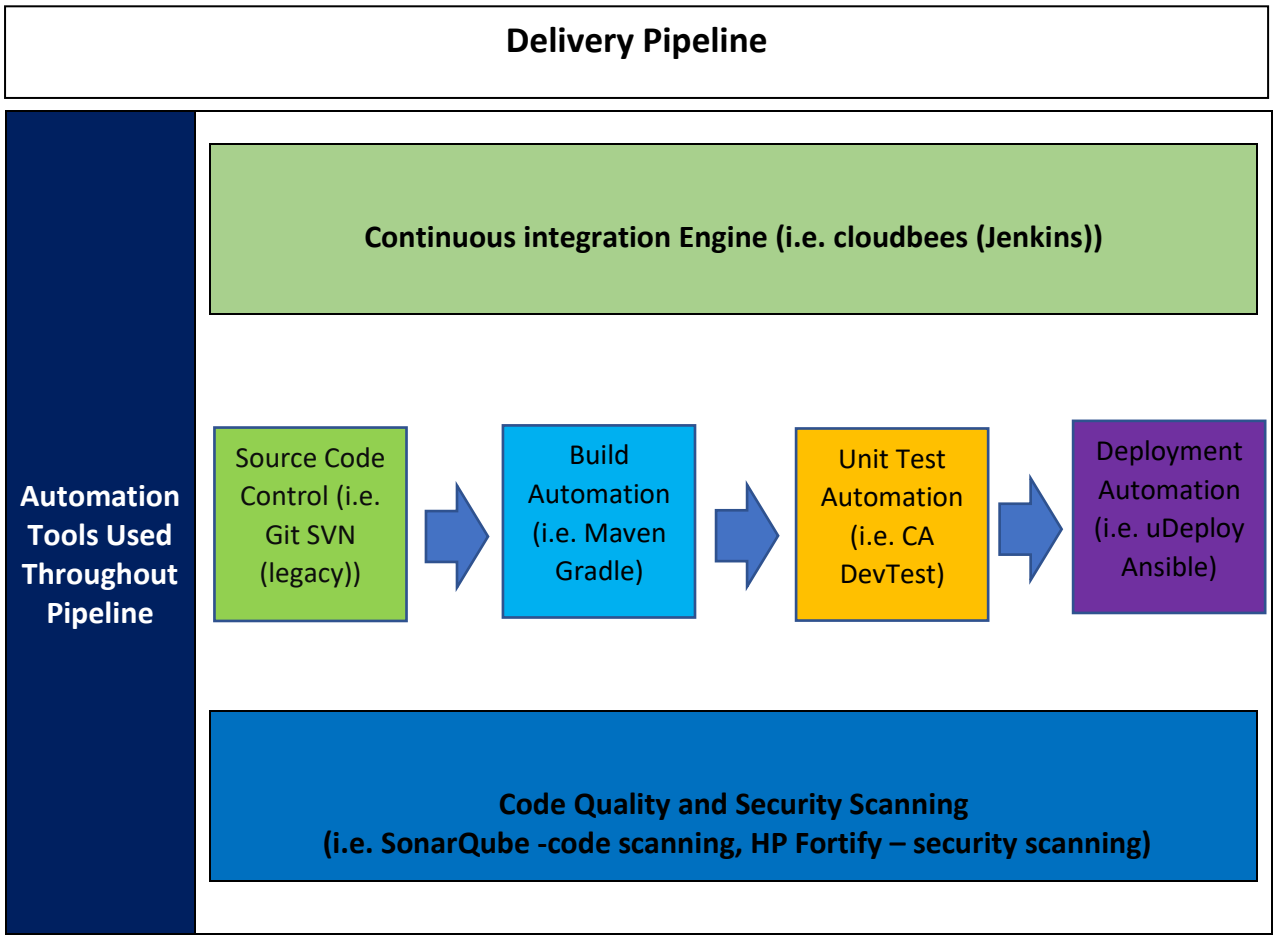

Figure 1: Continuous Integration Delivery Pipeline Example

The Stages explained

1. **Source code control (for the purpose of management)**

   Source code management, or source code control, is certainly not a new topic. This has been around for decades and has evolved over time. The basics here are that your organization stores its code in a source code control system or repository so that it can be tracked, maintained, versioned, and audited. You do not want the developers storing the code on their laptops or virtual machines and trust that will suffice for managing the code.

   Possible Tools

   - Git is probably the most widely used SCM system out there. It is an open source system.
   - Subversion (SVN) has been around for quite some time. It is also an open source system. It is still heavily used across many organizations out there, but there has been more of a push towards Git. If you are just starting out, I would highly recommend using Git.

2. **Build automation**

   Once a source code management system is in place and actively being used by your development team, the team will need to be able to compile and build their code. This is probably the first step in the whole chain of Continuous Integration events. This is what gets the ball rolling. The code needs to build cleanly before you can even think about deploying out to your environments for testing and production.

   Possible Tools

   - Gradle is an open source build automation system. Pretty widely used by top companies like Netflix, Google, and LinkedIn.
   - Maven is another open source build automation system.

3. **Unit test automation**

   Developers unit test their code to ensure that the functionality they are building works as expected. In an ideal world, the development team should be saving these unit tests, so that they can be reused and also put into a regression test bed.

   Possible Tools

   There are multiple tools out there for helping developer's unit test their code. Many of these tools are open source and can be used freely.

   - JUnit is an open source unit test framework. This is pretty widely used in the industry.
   - CA DevTest allows for the automation of unit testing, as well as a few other bells and whistles, like service virtualization.

4. **Deployment automation**

   For the last stage in the process, delivery teams need to deploy their code and applications out to various test environments and, of course, production. To reduce errors and overhead in the deployment process, while increasing speed to market, this step can be automated through a variety of tools and methods.

   Possible Tools

   - IBM Urbancode uDeploy allows you to model a process and orchestrate your deployment. This process can then be repeated across all your environments, and of course tweaked for each environment as needed.
   - Ansible is an open source IT automation tool. It can be used for everything from configuration management to product installation to application deployments. This tool is rapidly gaining acceptance and momentum in the DevOps community.

5. **Monitoring**

   Monitoring the processes throughout.

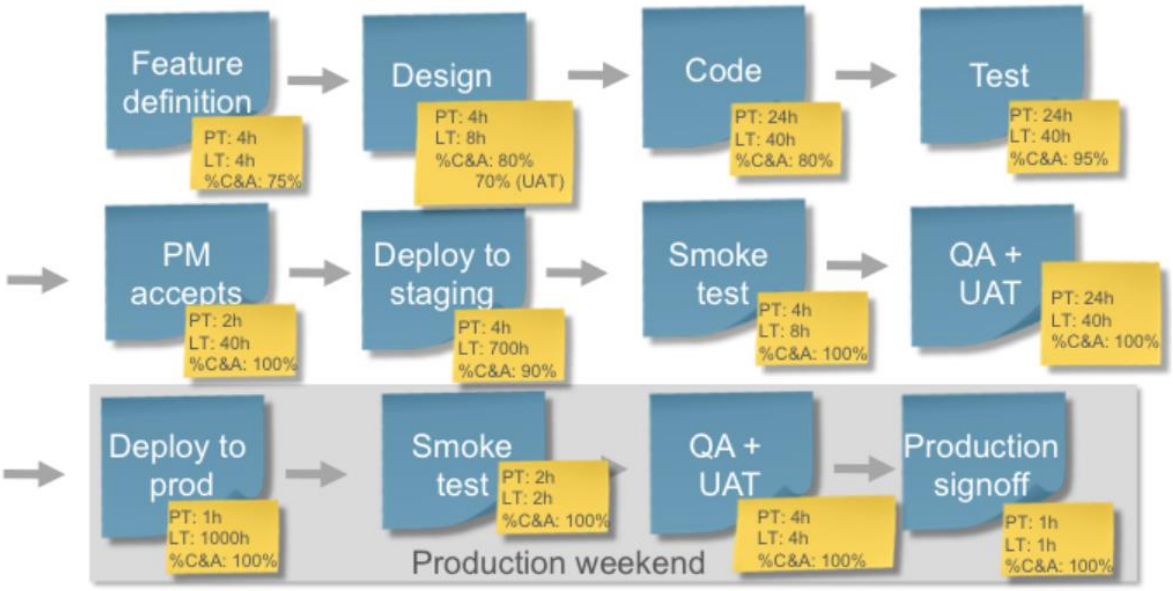

Figure 2: Pipeline Metrics

The continuous dimensions of the pipelines, which consists of the following:

- Continuous Exploration (EE)
- Continuous Integration (CI)
- Continuous Deployment (CD)

- Release on Demand

Continuous delivery is enabled through the deployment pipeline. The purpose of the deployment pipeline has three components: visibility, feedback, and continually deploy.

- **Visibility** – All aspects of the delivery system including building, deploying, testing, and releasing are visible to every member of the team to promote collaboration.
- **Feedback** – Team members learn of problems as soon as possible when they occur so that they are able to fix them as quickly as possible.
- **Continually deploy** – Through a fully automated process, you can deploy and release any version of the software to any environment.

The benefits will consists of the following:

- This capability helps the company stay a step ahead of the competition.
- Building the Right Product: Frequent releases let the application development teams obtain user feedback more quickly. This lets them work on only the useful features. If they find that a feature isn't useful, they spend no further effort on it. This helps them build the right product.
- Improved Productivity and Efficiency: Significant time savings for developers, testers, operations engineers, etc. through automation.
- Reliable Releases: The risks associated with a release have significantly decreased, and the release process has become more reliable. The deployment process and scripts are tested repeatedly before deployment to production. So, most errors in the deployment process and scripts have already been discovered. With more frequent releases, the number of code changes in each release decreases. This makes finding and fixing any problems that do occur easier, reducing the time in which they have an impact.
- Improved Product Quality: The number of open bugs and production incidents has decreased significantly.
- Improved Customer Satisfaction: A higher level of customer satisfaction is achieved.

Conclusion

The continuous delivery pipeline will help the Digibank to solve previous challenges and problems which will be fixed more quickly, will provide better and faster delivery of new features, will bring reduction in exposure to risk and improved software development process and practice at a costs of a better team morale.