git init

git add .

git commit -m “Initial Commit”

Tyler Schmitz

COSC 484

Sean Donnelly

April 22, 2019

PaaS With MicroServices

Dear Jane Doe,

Here is the summary for the recent talk I attended on Platform as a Service (PaaS) using microservices.

Key Terms:

Platform as a Service – This is a cloud computing model in which a third party provides hardware and software tools to the client over the internet. The third party hosts the hardware and software itself and allows users to interact with its systems without having to develop its own code or supply its own hardware.

Microservices – This is a way to design code. Instead of having one large application, the parts of the application are broken into many smaller pieces. This reduces coupling and increases interoperability. This also increases maintainable and testable code and are independently deployable. The independent microservices then interact with each through some common protocol.

Horizontally Scaling – In essence, horizontal scaling is using more parts to achieve a goal compared to vertical scaling which increases the power of each part. Ex. Horizonal will have 30 nodes with 1gb of ram process work compared to vertical where you would have 3 nodes with 10gbs of ram do the work.

MTTD – Mean Time to Detect

MTTR – Mean Time to Repair

git add .

git commit -m “Added Key Terms”

Summary:

In the talk the speaker explained the structure of the Netflix services and how they are evolving and why. The speaker began by explaining how Netflix is a large company with over 100 million users and having maintainable and scalable code is very important. In previous years there was a more simplistic monolithic approach which worked reasonably well for the time but needed to change to keep up with the demand. To accomplish this, they decided to go with a microservices based approach. This allowed individuals to create their own API’s to fit their needs. By using API’s that were more tailored to their product it reduced the amount of time and lowered the learning curve for these developers. This methodology was enabled through three strategies: Standardizing Components, using a Preassembled Platform, and Automation and tooling.

Standardizing Components was an important part of the process by providing a set standard many different problems. By setting limited ways to interact with the system Netflix can ensure that there is a higher concentration of knowledge on a smaller area of technology. This helps with more consistent systems that are easier to create, diagnose, and fix.

Providing a Preassembled Platform promotes a prompt and proper setup that precludes the need to prepare and program standard pieces (That sentence was preposterous). By providing a platform for the client to use that is ready, the client does not need worry about the many different factors that could go into setting up the service. Instead of worrying about things like documentation, versions, initialization, and configuration, the client can now move on to working on the actual product itself. It also helps ensure that important pieces are not left out due to accidental oversight or assuming a piece is not important. For example, by making alerts about issues part of the system automatically, this can reduce MTTD and MTTR.

The last strategy that was used was Automation and Tooling. To help lower the difficulty of setting up the environment and easing the testing process Netflix developed a CLI. This sets up the environment with all the tools the developer needs like npm, git, node, docker or whatever else you need.

git add .

git commit -m “Wrote Summary of Talk”

This style of setting up the system could be very effective at our company. Because many of the developers are coming from a Java background many of them may be unfamiliar with many different aspects and pieces that could be used inside a Node or JavaScript product. By standardizing we could ensure that they all learn an use similar methodology to create products. By giving them a preassembled platform it will reduce the amount of time it takes them to start working and ensure they don’t make mistakes in the setup process. Lastly, providing tools for them to use would streamline the process because they would not need to create the environment.

git add .

git commit -m “Added How Technology was applicable”

I think a key technology that should be investigated fully for our use is Docker. This would help with CI/CD development where it helps to make a smoother deployment and integration among different microservices. This increases the flexibility of our applications allowing them to interact separately together helping prevent coupling and entanglement allowing for more growth.

Further Reading:

Platform as a Service: <https://searchcloudcomputing.techtarget.com/definition/Platform-as-a-Service-PaaS>

Microservices: <https://www.youtube.com/watch?v=SouNISAnXlo>

git add .

git commit -m “Midterm Complete”

git tag “Midterm Completed”

git push