Research Document TuneTurtle

Nieben van Sint Annaland
05-22-2024

Project Description

TuneTurtle is going to be the next Spotify. It's going to have most of Spotify's features such as playlists, liked songs, skipping songs, looping songs, changing volume, recommendations, and as many other features as possible. It's going to be a microservice structured project that is going to be hosted on the cloud.

Problem Description

The project is going to be hosted on the cloud. This is something that I've never worked with, and is quite hard for me to grasp still. That is why research on this topic is important, so that I can get familiar with how the cloud works, so that in the future I can implement the hosting of the project on the cloud without an issue. The difference between all the different cloud providers and the different options they all provide is very overwhelming for someone that has never touched the cloud before. I expect the research to take about 2 weeks in total.

Research Question

Main: What are the key steps and considerations involved in deploying a microservice-based project on cloud platforms, and how do these processes contribute to the overall functionality and scalability of the application?

Sub:

1. What are the primary factors to consider when selecting a cloud provider for hosting microservices?
2. How do containerization technologies such as Docker contribute to the ease of deploying microservices on cloud platforms?
3. How can automated deployment pipelines and continuous integration/continuous deployment (CI/CD) processes streamline the deployment of microservices on the cloud?
4. How do monitoring and logging solutions help in maintaining the health and performance of microservices deployed on the cloud?

Research Methods

During the research of this project I'm going to be utilizing research methods from the **DOT Framework** to make sure that I research my questions properly. Using the

DOT framework is going to allow me to do research with a balanced perspective. I'm mainly going to be using these methods:

- **Community Research:** To look into if others have done this before and use that information.
- **Literature Study:** To find general information online.
- **Available Product Analysis:** Find out if what I've done has already been done by others in full.
- **Best Good and Bad Practices:** To look into what the best ways to tackle a problem is, and what not to do.
- **Prototyping:** To build something experimental with the information I've learnt to make sure that I fully grasp what I've researched.

Answers to research questions

What are the primary factors to consider when selecting a cloud provider for hosting microservices?

There are multiple important factors to consider when selecting a cloud provider. The most important primary factors are **pricing**, **serverless or server vs container**, **system for automatically scaling**, the **region** of the datacenters, what kind of **container orchestration** they provide, integration with popular **CI/CD** tools such as GitHub, **ease of use,** and a proper **learning curve**.

These are the most important factors while considering a cloud provider, there are of course more factors at play, depending on what project you're trying to build and the specific needs of that project.

How do containerization technologies such as Docker contribute to the ease of deploying microservices on cloud platforms?

Containerization technologies like Docker significantly contribute to the ease of deploying microservices on cloud platforms by providing a standardized and efficient way to package, distribute, and run applications. Some of the most important reasons why Docke rmakes it easier are:

- **Environment Consistency:** Docker makes sure all your dependencies and packages are installed no matter what kind of container you have

- **Isolation:** Docker makes sure all containers run separately from each other so there can't be any conflicts between them
- **Immutable Infrastructure:** Because Docker promotes the use of images, containers can be predicted and are immutable, meaning they can't be changed during deployment, because they're built from an image
- **Quick Provisioning:** Docker makes it really easy to start / stop / scale more containers during deployment for ease of use
- **Orchestration tools:** Docker has integrates support for orchestration tools like Kubernetes, which makes it really easy to build a huge application off of docker.
- **CI/CD Integration:** Docker has integrated support for CI/CD tools like GitHub which makes it really easy to automate deployments
- **Version Control:** Docker has integrated version control which allows you to rollback your setup if an mistake is made. This is really useful for developers working in bigger scale applications
- **Security Features:** Docker includes various security features, such as image signing, secure defaults, and isolation mechanisms, which help in securing containerized applications.

How can automated deployment pipelines and continuous integration/continuous deployment (CI/CD) processes streamline the deployment of microservices on the cloud?

Automated deployment pipelines and CI/CD's processes streamline the deployment of microservices on the cloud by automating the build, test, and deployment stages, ensuring faster, more reliable, and more consistent releases. Having your work automatically uploaded, tested, quality checked and published to a cloud provider can really save a lot of time for most developers, especially when working in a larger scale application. Using a proper CI/CD pipeline also gives you infinite configurability. If you're working with multiple microservices, you can setup your CI/CD pipeline to do something different for every single one, even in the same repository. Having this configurability and automation for everything really streamlines and improves the efficiency of a developer. A proper CI/CD pipeline also makes sure that the right steps are taken every single time a change is made, this ensures no possible obvious errors.

How do monitoring and logging solutions help in maintaining the health and performance of microservices deployed on the cloud?

Monitoring and logging solutions are crucial for maintaining the health and performance of microservices deployed on the cloud. They provide visibility into the system’s operations, help identify and diagnose issues, and ensure that services run smoothly. Having proper monitoring tools gives you real time insight into what’s going on in your application. It gives you the possibility to have a dashboard showcasing exactly the information you need, which is also something that you can configure to your needs. With a monitoring tool you also have insight in whether or not you’re scaling too much and if you’re able to scale down which in some cases can really save a lot of money. It also gives you immediate insight in when something goes wrong, which would otherwise sometimes take a long time to be noticed. A proper monitoring tool can also be used by end-users instead of just the developers of the application, since it’s usually rather easy to use, which can be really helpful for clients if you’re building something for them.

Main question:

What are the key steps and considerations involved in deploying a microservice-based project on cloud platforms, and how do these processes contribute to the overall functionality and scalability of the application?

Deploying a microservice-based project on cloud platforms involves a series of key steps and considerations that ensure the functionality, scalability, and reliability of the application. The most important key steps are picking the right cloud provider based on the project you’re doing, and what technology you need, such as integrated support with Kubernetes, pricing, need for support, etc. Another massive consideration is whether or not you want to use containerization technology such as Docker. Usually, if you want to streamline your process and automate stuff properly, an application that uses microservices is going to want to use some form of containerization technology. Using a CI/CD pipeline is also important to greatly improve the efficiency of the developers while creating the project, since everything besides the code is done for them, the deployment, the code checking, testing, etc.