JETEISH PRATAP SINGH
JSING162

SER 516 Assignment 4 GitHub Actions

Q1. What do the authors mean by "infrastructure is fluid"? In what ways can it be fluid?

The term fluidity here means that the infrastructure here in the context of cloud native applications is changing. They are changing in the sense that the resource like computational requirements, storage, network, database or anything else far and between is not static and might change based on updating requirements or market forces. The authors here mean that cloud native apps are designed with this possibility of changes in consideration. In lean also we defer commitment meaning we make final or critical decisions at the last moment, here this can be linked with infrastructure as these "critical" decisions here pertain that of cloud native applications as these applications scale and grow throughout their lifecycle and moved and replaced at any time as required.

Containers are one of the "tools" used in development of these cloud native applications. These containers are responsible for ensuring uniformity of these services that are running on cloud. They are run and decomposed as needed meaning it is not static that they are always running and all the services that were running initially are running live in real time and are dependent on the use case and requirements. In the same sense the coding environments change and modify, might happen accordingly to the development scenario or it may be causing the org might be shifting their tech stack or tools used. Also, when an org might want to roll out updates and give out new features that might completely revamp their service/ application as a whole. This rolling out of features in itself is very dynamic in nature and makes the whole concept of cloud applications very fluid.

Q2. What do the authors mean by "failure is constant"? In what ways is failure constant?

In the paper the author has stated that "applications developed for execution on single PC, mainframe assume underlying OS and hardware as rock solid and when these are ported to cloud, they fail". This means that when assuming the requirements would come back to bite you as this would not only mean that you are assuming the specifications to be rigid, but this would ensure that the application would sure shot fail when hosting on cloud as the requirements are very specific and rigid. Having said that, the author has also stated that as we try to scale the application something is bound to break this is guaranteed by the law of large numbers. The cloud applications are very complex, and fluid (discussed above), this is due to the fact that they are almost entirely distributed in nature and multiple components and modules. This raises the expectation of failure, and we can assume that our system will fail.

The containers we discussed are run and decomposed multiple times, this means that there is also some chances where they might fail and crash. When rolling out new features and bug fixes/ updates bugs might be introduced in one or more components that might affect all the linked components and affect the application and cause failure.

Q3. Which aspects of Figure 1 are true microservices and which are appliances as we discussed when doing microservices?

According to the paper, specifically “An Application Walkthrough”, which highlights and illustrated the application is designed/ built there are 3 types of microservices in figure 1 “simple microservice cloud-native application”. These are namely:

1. **Data monitors**: The services from M1 - M6 are dubbed as data monitor microservices. These skim the data(news sites, RSS feeds in this example application in the figure).
2. **Analysis services**: These are the services from A1 – A5. These hold the actual, as the name suggests analysis methods depending on the application needs and requirements.
3. **Data managing services**: Services through D1 – D3 push documents into AWS queue services and simply responsible for managing the access to the databases.

According to Fowler microservices are, “small services that are capable of running in its own process and communicating with lightweight mechanisms”. This means that all the microservices identified are scalable and completely modular. Meaning their numbers can be scaled up and down and can be replaced without any disruption.

As for the appliances these are:

1. AWS Simple queueing service
2. AWS DynamoDB
3. Azure CosmoDB

These are not microservices but are instead appliances that are supporting the cloud application. They provide infrastructure to the application in the give example(fig 1).

Q4. Identify the challenges in implementing an observability solution for this hypothetical application. You do not have to design a whole solution architecture, I’m asking you to consider what will be difficult to do when considering the 3 pillars of observability (so yes, structure your answer here in terms of the 3 pillars)

1. There are many microservices in the example. This means there might be different ways in which logs are formatted/ logged, or what is the limit as to which detail logs are put in. This means when logs are aggregated it would pose a challenge based on these concerns standardizing them might pose an issue.
2. When gathering services metrics for various reasons we might encounter challenges of handling generalizing metric conventions and naming throughout the project and different appliances that are utilized in development. We are also using different appliances that might pose a problem as they might have their own bespoke way of gathering metrics and data which might have conflicted or incompatible naming/ metric conventions.
3. We might require custom tracing methods as we are using multiple cloud native appliances that might or might not have compatible tracing routes/methods that persist across services and/or programming languages. Hence it might pose a challenge to set up trace pipeline (channel) that retains context through AWS ⇔ Azure.