Project espresso summary

We had a project to deploy an application called **ESPRESSO SHOP**. which was a microservices application. We deploy on kubenetes clusters. The main goal of the customer when he contacted us was to deploy an online shop where millions of customers can come and purchase coffee and be able to leave reviews. They required high availibility, zero downtime, biweekly deployment and on demand scalability.

So the application was made of **7 services**, one service for the **frontend, 3 API**( the shop products,the shop review 1 and the shop review2, and **3 databases** (postgress, postgress replication, and redis. The 3API were communicating with the postsgress and postgress replication databases and the front end was communicating with the redis database for session catching. When we started designing the application we added an ingress controller so that the traffic of espresso customers get redirected to the appropriate service using an ssl. This provide more security for the application.

Like I said earlier we use kubenetes to deploy this application but we also use **Helm chart** for package management.

The application had 7 services and we deploy **31 ressources files** to accomplish this project.

For the frontend service we had 4 service files: **the deployment file, the service file** to allow pods to properly communicate with each other, an **horizontal pod autoscaller** to be able to scale the application up and down depending of the traffic, and a **pod disruption budget** for more security and avoid *involuntary disruptions* to the application.

All the apis services also had the same 4 ressources each, the deployment, the service file, the hpa and the pdb files deployed.

The 2 postgress databases that are communicatiing with the APIs has 6 ressources deploy.

The deployment file, the **persistant volume, and the persistant volume claim** since they are databases we needed to store datas and mount them to a directory.

So after the pv and the pvc we also deploy the **configmap** for both postgress and postgress replication services to be able to store all the non sensitive datas. We also deploy a pdb for all the databases for more security.

The redis database had 3 ressources file the deployment, the svc and the pdb.

We encounter many problems when deploying the espresso app

-The first problem was a confusion on hyphen and underscore. We realized that when we put the underscore on the deployment name the deployment was not going through and we did put it on the value it works. So we had to use hyphen on the template file and underscore on the value file