Assignment 4

Assumption:

* The current application is monolithic.
* The current application is deployed on-prem.

Current Scenario:

* As the current application is monolithic deploying a newer version or rolling back to the older version will be a big challenge and the cost will also increase as we must provision a new box to deploy the newer version. Test the changes and if everything works fine then delete the older version.
* The following are the drawbacks of the above architecture.
  + Delayed delivery to market.
  + No resource optimization.
  + Scaleup and scale down will require downtime.
  + If we have to change any subproject or module in current architecture, we must build the entire application.

Solution:

* We have to breakdown the monolithic application into smaller microservices based on the requirement.
* Following will be the architecture of microservices deployment.

A screenshot of a computer

Description automatically generated with medium confidence

Benefits of Microservice Architecture:

1. We can implement CI/CD using the above architecture to achieve faster delivery to the market.
2. We will only deploy the microservices for which we have implemented the code changes.
3. The code quality and code reliability will increase significantly using the above approach.
4. There will be no downtime when using the above approach for deployment.

Process:

* For newly created microservices we will create docker images
* To create any new docker image we will implement CI which will make sure the code passes static code analysis and code review process to make sure there are less bugs in production.
* The newly create docker image will be scanned using ECR image can.
* The deployment of the newly created docker image will be done to Kubernetes cluster using Jenkins.
* The pod for the given service will have 2 containers [Application container and sidecar container (filebeat) ]
* The filebeat container will send the logs to Logstash server and the Logstash server will push the logs to Elasticsearch.
* Each microservice will implement opentelemetry-api, opentelemetry-api-trace which will allow the developer team to trace the transaction and pinpoint the error.
* We will also implement elastic-apm to monitor the performance of the application which will help us to find the latency, tps and throughput of the given microservice.
* Prometheus and Grafana will be used to monitor the application and visualize the data.

Tech Stack used:

* Containerization: Docker
* Orchestrator: Kubernetes.
* CI/CD Tools: Jenkins, Maven, Gradle.
* Scripting: Bash, python.
* Logging: Elasticsearch, Logstash, Kibana.
* Monitoring: Prometheus and Grafana
* Tracing: open telemetry, jaeger.
* Code coverage: jacoco
* Static Analysis: SonarQube
* Testing Report: Cucumber.
* Code repository: GitHub, GitLab
* Database: Mongo dB