**Application Architecture and benefits**

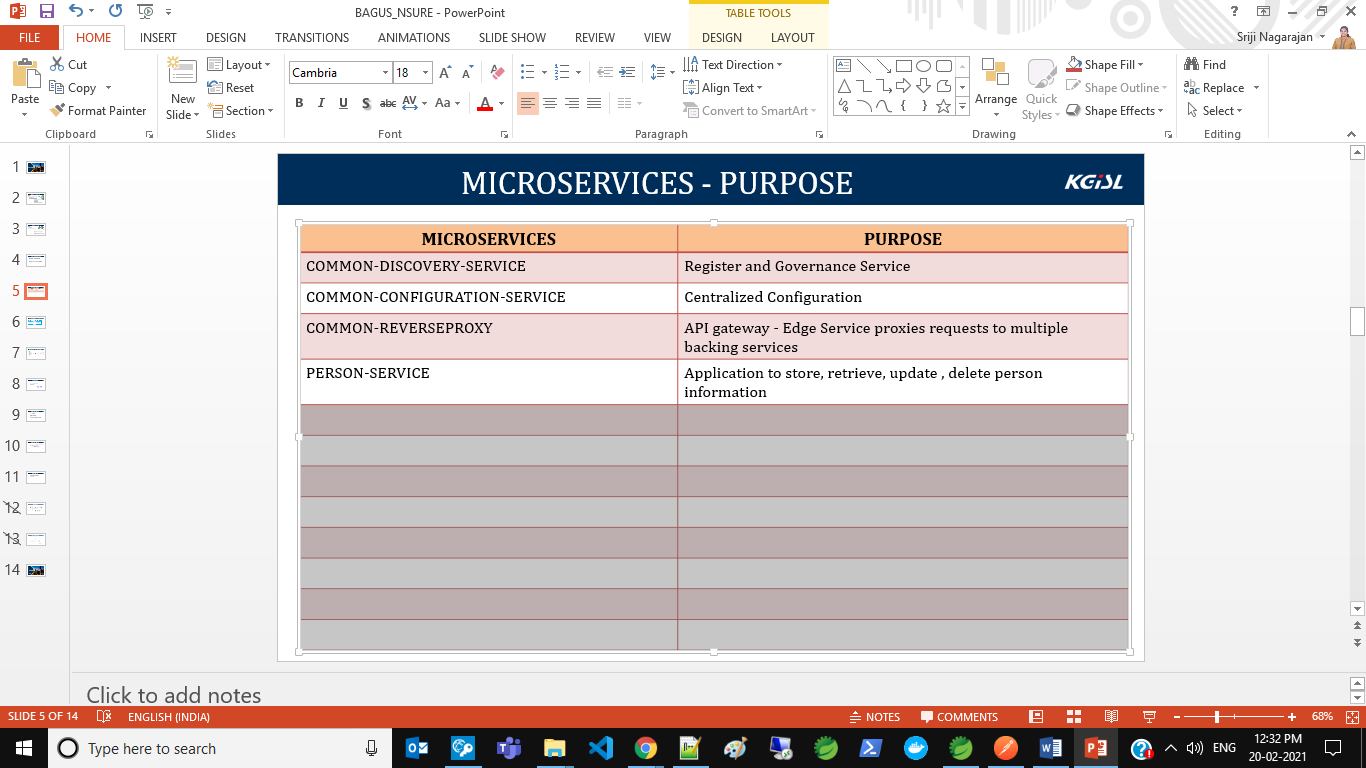
* Microservices are **Independently Executable, Upgradable , Replaceable and Scalable**
* **Domain specific features** – Achieve **Single Responsibility Principle**
* **Light-Weight , Independent, Loosely Coupled** business unit
* Owns **Codebase, Managed, and Developed** by a small team.
* Own their **Database – Decentralized Data Management**
* Choose **Best Technology Stack - Evolutionary Design**
* Test, Release, Deploy, Scale, Integrate, and Maintain Independently – **Own Devops Plan**
* **Smart Endpoints and Dump Pipes -** REST over HTTP or asynchronous messaging
* **Recovery** and **Fault Tolerance – Auto Healing Capability**
* **More Effective , Efficient Deliverables** and **Customer Focused**

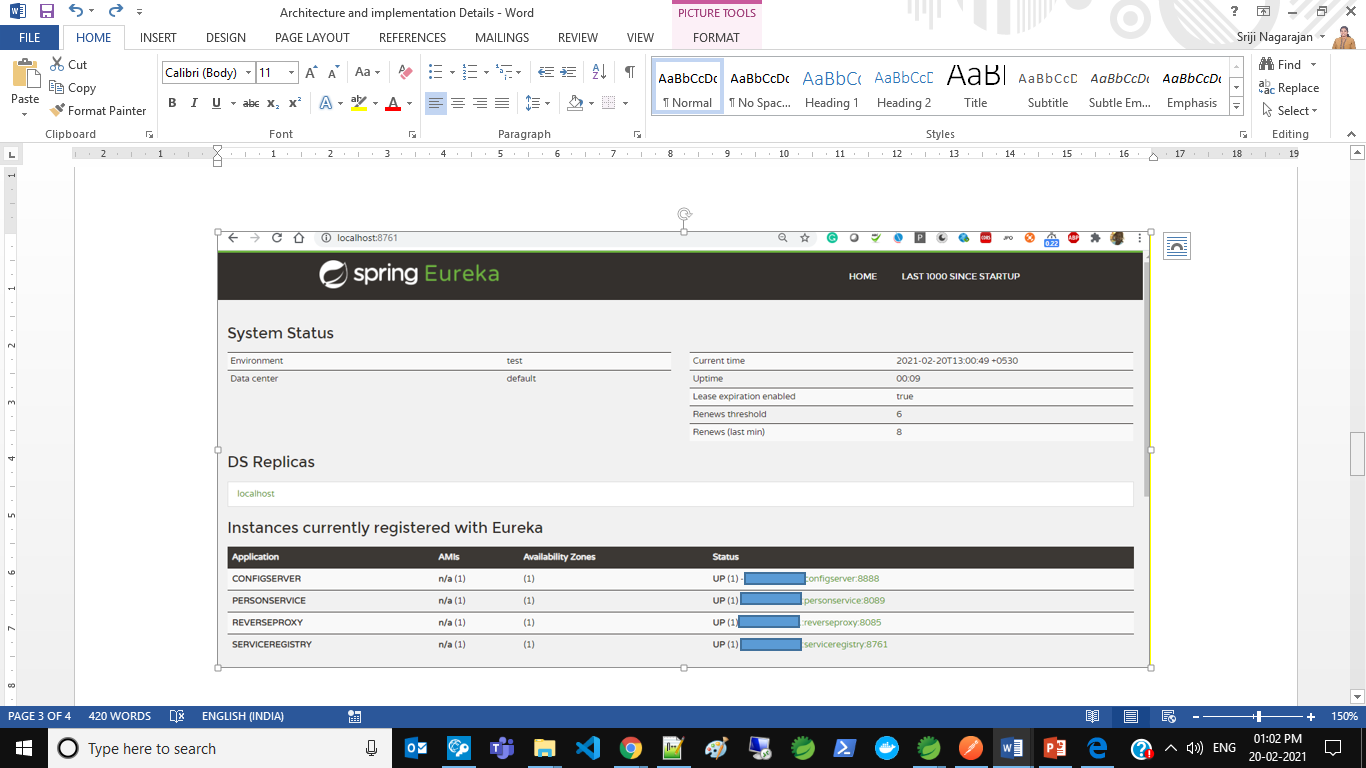
**Key Features of Application**

* Distributed/versioned configuration
* Service registration and discovery
* Routing
* Service-to-service calls
* Load balancing -
* Circuit Breakers
* 12 Factor Implementation
* Docker File – High Availability, Resilient
* Kubernetes Cluster – Clustering Framework
* Swagger – API Specification
* Auto Scaling , Multiple Instance (Replica) - Each Services
* chaos monkey - Testing

**Implementation via 2 Ways**

* Running as independent Spring boot application (Refer ReadMe File in PERSON-SERVICE)
* Running via gateway service (Netflix Zulu**)**
  + Order Of Execution
    - COMMON-DISCOVERY-SERVICE
    - COMMON-CONFIGURATION-SERVICE
    - COMMON-APIGATEWAY
    - PERSON-SERVICE





**PURPOSE OF DISCOVERYSERVICE**

--------------------------------------------

Eureka server acts as the registry to which other services register themselves.

When one micro service needs to communicate with another, it can query Eureka to find the active instance of that microservices.

**PURPOSE OF CONFIGURATION-SERVICE**

-----------------------------------------------------

Externalize properties or resource files out of server where the values of those resources vary during runtime – usually different configurations that will differ in each environment.

The centralized properties for all the service are served from COMMON-CONFIG from the below path

https://github.com/SrijiVijay/COMMON\_CONFIG.git

**PURPOSE OF GATEWAY-SERVICE**

---------------------------------------------

Zuul is (of course) our gatekeeper to the outside world, not allowing any unauthorized external requests pass through. Zulu also provides a well-known entry point to the micro services in the system landscape. Using dynamically allocated ports is convenient to avoid port conflicts and to minimize administration but it makes it of course harder for any given service consumer. Zuul uses Ribbon to look up available services and routes the external request to an appropriate service instance.

**PURPOSE OF PERSON-SERVICE**

---------------------------------------------

Simple RESTful API which provides a service for storing, updating, retrieving and deleting Person entities.

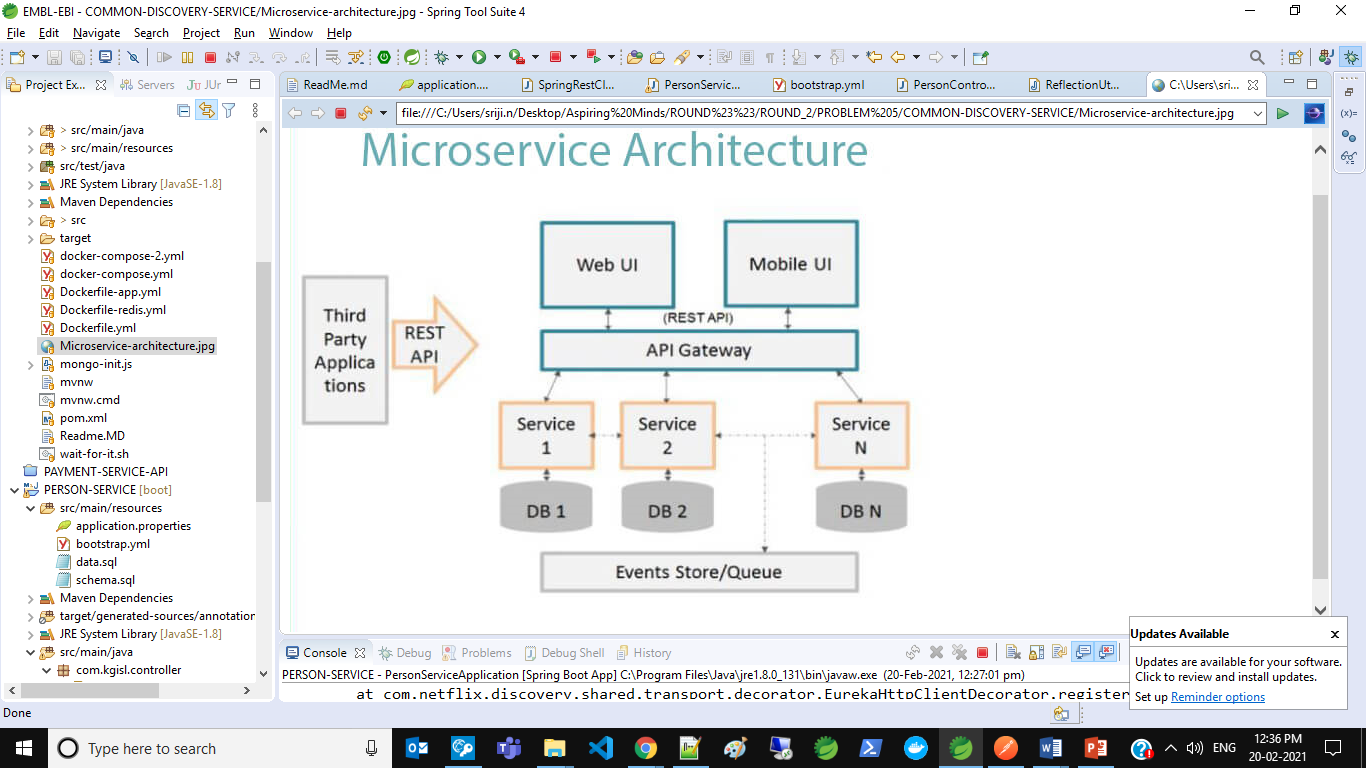
**URL to Run the Application is mentioned in README.md file**

**FAULT TOLERANCE & LOADBALANCING**

-----------------------------------------------------

* Hystrix is a circuit-breaker. Hystrix lets you define a fall-back method that gets invoked if your network calls to another microservice fails.
* It reverts back to normal behaviour once the service is available again.
* Ribbon is added for load balancing

**HIGHLEVEL ARCHITECTURE**



**Cloud Compatible Architecture**

* Application is added with docker for containerization on below pattern

