USING THE C4 MODEL TO DOCUMENT

‘BOOKS STORE SYSTEM’ (It is required)

Level 1: Context Diagram for Books Store System

- Books Store System allows users to interact with book records
- This diagram is more than enough when trying to convey three details we need to need to our stockholders about Books Store System
    - It has two different types of users: Public and Authorized
    - Depends on an external Authorization System for authorization purposes
    - Depends on an external Publisher System for giving details about books published by them

Level 2: Container Diagram for Books Store System

- This diagram zooms in to mention more technical details about the actual system, it explicitly indicates the technologies we use and depending on the technology being used it calls out the concrete containers in charge of handling that work
    - Search Web API
        - Developed with Go
        - Allows only authorized users searching books records via HTTPs handlers
        - Authorized by external Authorization System.
        - Use ElasticSearch as the Search Database for searching read-only records
    - Admin Web API:
        - Developed with Go
        - Allows only authorized users administering books details via HTTP handlers
        - Authorized by external Authorization System.

    - Use PostgreSQL as the Read/Write Relational Database for administering records (Read data from and write data to Read/Write Relational Database)
    - Publishes Events to external Books Kafka container
- Public Web API: Allows Public users to get books details
    - It reads data from Read/Write Relational Database
    - It reads/writes data to Reader Cache database
- Book Kafka System
    - Use Apache Kafka 3.0
    - Handles book-related domain events
- ElasticSearch Events Consumer
    - Developed with Go
    - Listening to Kafka domain events and write publisher to Search Database for updating
- Search Database: Stores searchable books details
    - ElasticSearch
- Read/Write Relational Database: Stores books details
    - PostgreSQL
- Reader Cache: Caches books details
    - Memcached
- Publisher Recurrent Updater:
    - Listening to external events coming from Publisher System
    - Updates the Read/Write Relational Database with detail from Publisher system
    - Kafka
    - It uses the Admin Web API for updating that data

Level 3: Component Diagram for Admin Web API

- It's a service which name is [service.Book], allow administering books details
- It authorizes books detail by using Authorization Service, named [service.Authorizer]. Authorization Service allows authorizing users by using external Authorization System

- It publishes books-related events to an Events Publisher, named [service.EventsPublisher]
- Events Publisher publishes books-related domain events to external Books Kafka container
- Read from and write data to Read/Write Relational Database

Deployment Diagram – Microservices Deployment ‘*BOOKS STORE SYSTEM’ On AWS (It is optional)*

- **Books store system** will be deployed in AWS EKS
- The architecture is divided into two parts
  - Public internet : It’s just a normal internet and users are on public internet. Users make APIs calls
  - Private infrastructure on AWS
    - There is a VPC that has three subnets: 2 private subnets (named: priv-net-a and priv-net-b) and one public subnet (named: pub-net)
    - In Public Subnet
      - Deploying AWS API Gateway as a gateway to the privates subnets and to access APIs
    - In Private Subnets
      - There is a EC2 (named EC2-a) worker node in private subnet: priv-net-a
      - There is a EC2 (named EC2-b) worker node in private subnet: priv-net-b
      - Search Web API, Admin Web API and Public Web API will be deployed on EC2-a as microservices
      - Use AWS RDS for deploying PostgreSQL Read/Write Relational Database. AWS RDS is in priv-net-b
      - Deploy ElasticSearch Search Database on AWS OpenSearch. And AWS OpenSearch is in priv-net-b
      - ElasticSearch Events Consumer will be deployed on EC2-b as a micro-service

        - Use AWS ElastiCache for Reader Cache. It’s in priv-sub-b
        - Deploying Book Kafka System on EC2-b
    - Authorization System, Publisher System are external system so we will not documented it. Their workloads are considering in AWS cloud as well

Simple Dynamic Diagram – Workflow CI/CD for deploying

‘BOOKS STORE SYSTEM’ using AWS services (It is optional)

- The workflow includes the following steps:
    1. Developer commit and push changes to a source code repository.
    2. A webhook on the code repository triggers a CodePipeline build in the AWS Cloud.
    3. CodePipeline downloads the source code and starts the build process.
    4. CodeBuild downloads the necessary source files and starts running commands to build and tag a local Docker container image.
    5. CodeBuild pushes the container image to Amazon ECR. The container image is tagged with a unique label derived from the repository commit hash.
    6. CodeBuild deploys image on Amazon EKS