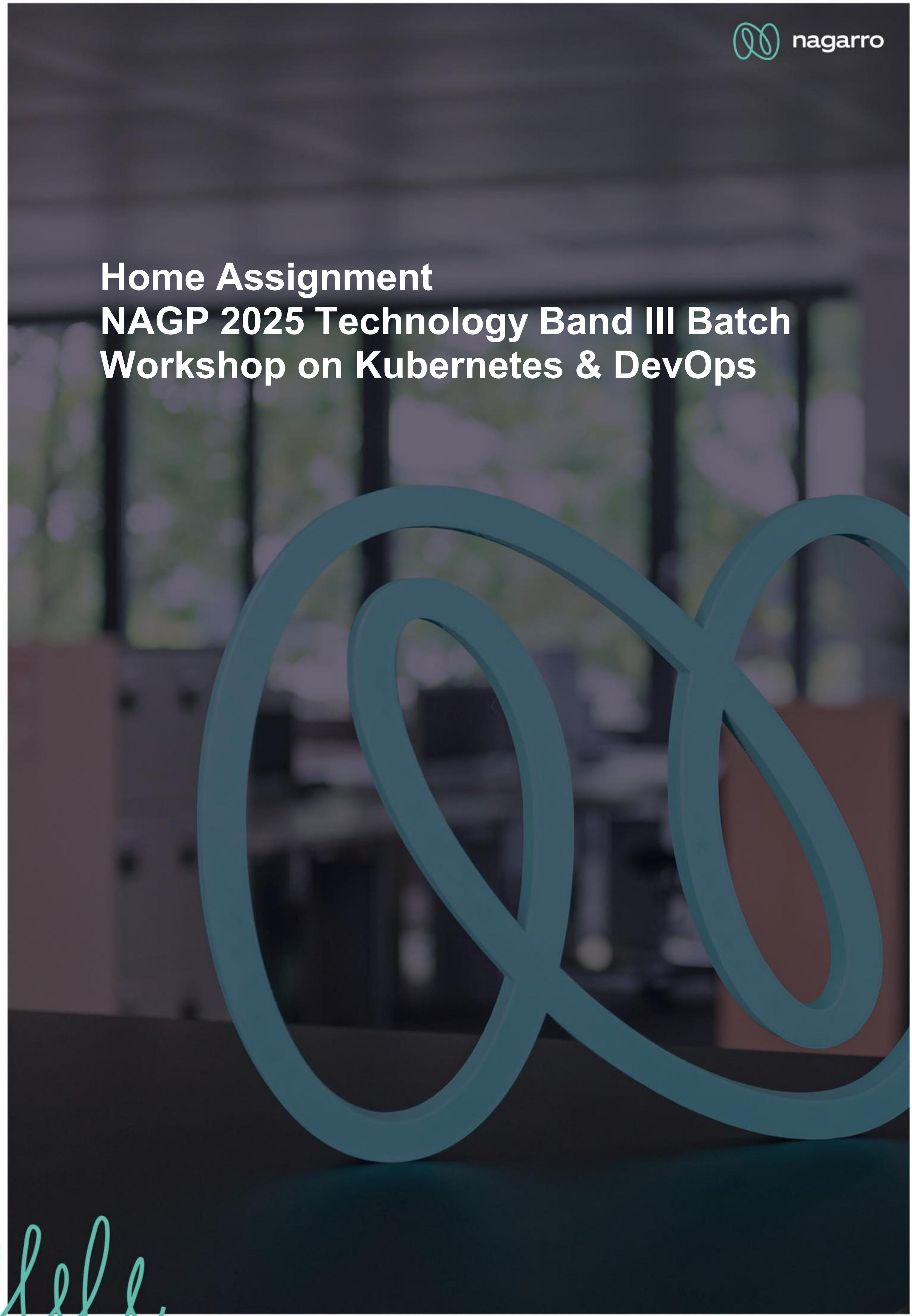

Home Assignment
NAGP 2025 Technology Band III Batch
Workshop on Kubernetes & DevOps

Problem Statement

You are required to design, containerize, and deploy a multi-tier architecture on Kubernetes involving one microservice and one database.

The system should simulate a simple real-world setup where the service tier fetches data from the database tier via an exposed API.

Build and push application docker images to Docker hub

Any Tech Stack can be used for microservice or application.

Service API Tier

- Exposes an API/Application endpoint.
- On API/Application invocation, fetches data from the database tier.
- Can use any standard language/framework of your choice (Node.js, Java, Python, .NET, etc.).
- Should use best practices for connecting to the database (e.g., connection pooling, config separation).

Database Tier

- Must include one table with 5–10 records.

- Should support data persistence.

Kubernetes Requirements

Feature	Service API Tier	Database Tier
Exposed outside the cluster	✓ Yes	✗ No
Number of pods	4	1
Rolling updates support	✓ Yes	✗ No
Persistent storage	✗ No	✓ Yes

Other Requirements

- The database configuration to be provided in Service API tier should be configurable from outside the pod definition file and application code (use Kubernetes ConfigMap).
- The database connection password should not be clearly visible in any Kubernetes YAML files (use Kubernetes Secrets).
- Database data should not be lost if the pod for database is re-deployed.
- Pod IPs should not be used for communication between tiers.
- Expose the Service API Tier externally using **Ingress**

Deliverables

- Source Code for the project. Provide repository URL, don't upload whole source code.
    - Make sure it includes all Kubernetes YAML files used in the assignment.
    - Dockerfile should be present as well.
    - Repository can be GitHub or Gitlab. **DO NOT use your project source code.**
- Also include a README file in code which has:
    - Link for the code repository.
    - Docker hub URL for docker images.
    - URL for Service API tier to view the records from backend tier.
    - Screen recording video showing all the objects deployed in Kubernetes cluster.
        - Show all objects deployed and running.
        - Show an API call retrieving records from database.
        - Kill API microservice pod and show it regenerates.
        - Kill database pod and show it regenerates and keeps old data.
- Prepare a comprehensive documentation that includes the following sections:
    - Requirement Understanding
    - Assumptions
    - Solution Overview
    - Justification for the Resources Utilized

Note:-

- **Do NOT use any client or company project** source code.
- You may **delete your Kubernetes cluster** after the deliveries have been captured to avoid any additional cost.