Nagp Kubernetes and Devops DAR Document

A logo with a black background

Description automatically generated

Nagarro Software Pvt. Ltd.

Vikas Luthra

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision History** | | | |
| Version | Date | Author/Contributor | Comments |
| 1.0 | 31-Jul-2025 | Vikas Luthra | Initial draft |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1 Introduction 5](#_Toc205137704)

[1.1 Objective and scope of document 5](#_Toc205137705)

[2 Requirements at a Glance 5](#_Toc205137706)

[3 Available tools 5](#_Toc205137707)

[3.1 Managed Kubernetes Services 6](#_Toc205137708)

[3.1.1 Amazon EKS (Elastic Kubernetes Service) 6](#_Toc205137709)

[3.1.2 Azure AKS (Azure Kubernetes Service) 6](#_Toc205137710)

[3.1.3 GCP (Google Cloud Platform 6](#_Toc205137711)

[3.2 Red Hat OpenShift 7](#_Toc205137712)

[3.2.1 Features 7](#_Toc205137713)

[3.3 PostgreSQL 7](#_Toc205137714)

[3.3.1 Features 7](#_Toc205137715)

[3.4 MySQL 8](#_Toc205137716)

[3.4.1 Features 8](#_Toc205137717)

[3.5 Oracle Database 8](#_Toc205137718)

[3.5.1 Features 9](#_Toc205137719)

[3.6 Microsoft SQL Server 9](#_Toc205137720)

[3.6.1 Features 9](#_Toc205137721)

[4 Comparison Analysis 10](#_Toc205137722)

[4.1 Openshift vs EKS vs AKS vs Docker Swarm 10](#_Toc205137723)

[4.2 PostgreSQL vs MySQL vs Oracle Database vs SQL Server 11](#_Toc205137724)

[5 Solution overview 13](#_Toc205137725)

[5.1 Architecture 13](#_Toc205137726)

[5.2 Kubernetes Objects 14](#_Toc205137727)

[5.3 Workflow 14](#_Toc205137728)

[5.4 Resilience & Scaling 15](#_Toc205137729)

[5.5 Security 15](#_Toc205137730)

[5.6 Deployment Flow 15](#_Toc205137731)

[5.7 Deliverables 15](#_Toc205137732)

[6 Recommendation 16](#_Toc205137733)

[7 Assumptions 17](#_Toc205137734)

[8 Risks 17](#_Toc205137735)

# Introduction

## Objective and scope of document

This is DAR (Decision Analysis and Resolution) document for evaluation of tools and technologies used for Kubernetes and Devops Assignment. There is brief description of each tool, comparison of the tools of same category and then recommendation of tool which best suits the requirements. The document may touch upon but will not cover detailed implementation methodologies used to achieve the design approach or system components described in the document.

# Requirements at a Glance

The solution needs to design, containerize, and deploy a multi-tier architecture on Kubernetes involving one microservice and one database. The microservice image is needed to be pushed to docker hub. The microservice needs to fetch the data from database and expose it to outside world via an endpoint.

|  |  |  |
| --- | --- | --- |
| **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 |

# Available tools

Following is the list of tool/technologies/frameworks that has been considered while creating the high level design of the application.In following sections there is description of the features of these tools and comparison for choosing the one which best suits application requirement.

**Application deployment:**

* Redhat Openshift
* Managed Kubernetes Services (EKS, AKS)
* Docker Swarm

**Database:**

* PostgreSQL
* MySql
* Oracle database
* Sql Server

**Microservice:**

* Java

## Managed Kubernetes Services

Managed Kubernetes Services are cloud-based solutions provided by major cloud providers to simplify the deployment, management, and scaling of Kubernetes clusters. They handle the underlying infrastructure, allowing users to focus on application development and deployment.

### Amazon EKS (Elastic Kubernetes Service)

Features

* **Fully Managed:** AWS handles the Kubernetes control plane, including upgrades and patching.
* **Integration with AWS Services:** Seamless integration with other AWS services like IAM, VPC, CloudWatch, and RDS.
* **Scalability:** Supports auto-scaling of clusters and nodes.
* **Security:** Integrated with AWS security services such as IAM for role-based access and Secrets Manager for secure storage of credentials.
* **Network Management:** VPC integration and support for AWS PrivateLink and Service Discovery.
* **High Availability**: Multi-AZ deployments for high availability and fault tolerance.

### Azure AKS (Azure Kubernetes Service)

Features

* **Fully Managed:** Azure manages the Kubernetes control plane, including updates and maintenance.
* **Integration with Azure Services:** Integration with Azure Active Directory (AD), Azure Monitor, and Azure Container Registry.
* **Scaling:** Supports automatic scaling of nodes and clusters, with integration for Azure Autoscale.
* **Security:** Integration with Azure security services like Azure AD for authentication and Azure Security Center for security management.
* **Networking:** VNet integration and support for Azure Load Balancer and Application Gateway.
* **Developer Tools**: Supports Azure DevOps and GitHub Actions for CI/CD.

### GCP (Google Cloud Platform

Features

* **Compute Engine:** Virtual Machines (VMs) on demand
* **Google Kubernetes Engine (GKE):** Managed Kubernetes cluster service.
* **Cloud Storage:**  Object storage for any data type. Supports multi-region, versioning, lifecycle rules, and fine-grained access control.
* **Cloud Functions:** Serverless execution of code in response to events.
* **BigQuery:** Serverless, highly scalable data warehouse.

## Red Hat OpenShift

Red Hat OpenShift is a comprehensive Kubernetes-based container platform that provides a robust environment for building, deploying, and managing containerized applications. It is designed to support enterprise-scale workloads with added features and capabilities that enhance Kubernetes.

### Features

* **Kubernetes-based:** Built on top of Kubernetes, offering advanced container orchestration.
* **Integrated CI/CD**: Built-in Jenkins integration and OpenShift Pipelines for continuous integration and delivery.
* **Contract-based:** Interface and policies are strictly described by an interface specification.
* **Developer Tools:** Provides developer-friendly tools like OpenShift Console, CLI, and Developer Sandbox.
* **Security:** Includes security features such as Role-Based Access Control (RBAC), Security Context Constraints (SCC), and integrated authentication.
* **Multi-cloud Support**: Can be deployed on various public and private clouds, including AWS, Azure, and Google Cloud.
* **Service Catalog**: Access to a marketplace of pre-configured services and applications.
* **Operator Framework:** Automates application lifecycle management using Operators.
* **Networking:** Advanced networking features, including Service Mesh and Network Policies.
* **Logging and Monitoring:** Integrated logging and monitoring tools (e.g., Elasticsearch, Fluentd, and Kibana (EFK) stack).

## PostgreSQL

PostgreSQL is a powerful, open source object-relational database system. It has more than 15 years of active development and a proven architecture that has earned it a strong reputation for reliability, data integrity, and correctness. PostgreSQL runs on all major operating systems, including Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), and Windows.

### Features

* User-defined types
* Table inheritance
* Sophisticated locking mechanism
* [Foreign key referential integrity](http://www.postgresqltutorial.com/postgresql-foreign-key/)
* [Views](http://www.postgresqltutorial.com/postgresql-views/), rules, [subquery](http://www.postgresqltutorial.com/postgresql-subquery/)
* Nested transactions (savepoints)
* **Multi-version concurrency control (MVCC):** PostgreSQL manages [concurrency](https://en.wikipedia.org/wiki/Concurrency_control) through a system known as [multiversion concurrency control](https://en.wikipedia.org/wiki/Multiversion_concurrency_control) (MVCC), which gives each transaction a "snapshot" of the database, allowing changes to be made without being visible to other transactions until the changes are committed. This largely eliminates the need for read locks, and ensures the database maintains the [ACID](https://en.wikipedia.org/wiki/ACID) (atomicity, consistency, isolation, durability) principles in an efficient manner.
* **Asynchronous replication:** PostgreSQL includes built-in binary replication based on shipping the changes ([write-ahead logs](https://en.wikipedia.org/wiki/Write-ahead_logging)) to replica nodes asynchronously, with the ability to run read-only queries against these replicated nodes. This allows splitting read traffic among multiple nodes efficiently.

PostgreSQL is the first database management system that implements multi-version concurrency control (MVCC) feature, even before Oracle. PostgreSQL is designed to be extensible. In PostgreSQL, you can define your own data types, index types, functional languages, etc. Many companies have built products and solutions using PostgreSQL. Some featured companies are Apple, Fujitsu, Red Hat, Cisco, Juniper Network, etc.

## MySQL

MySQL is an open-source relational database management system (RDBMS) known for its speed, reliability, and ease of use. It is widely used in web applications and is a key component of the LAMP stack (Linux, Apache, MySQL, PHP/Perl/Python). MySQL is maintained by Oracle Corporation and has a large community of users and developers.

### Features

* **High Speed**: MySQL is designed for high-speed data retrieval and processing, with optimizations for both read and write operations.
* **Indexing**: Supports various types of indexing to improve query performance, including primary, unique, and full-text indexes.
* **Replication**: MySQL offers several replication options, including master-slave replication and master-master replication, to enhance scalability and fault tolerance.
* **Backup and Recovery**: Includes tools for data backup and recovery, such as mysqldump and MySQL Enterprise Backup.
* **Authentication**: Supports various authentication plugins and mechanisms, including native password and LDAP authentication.

## Oracle Database

Oracle is the one of the largest vendor of RDBMS (relational database management system) in the IT market. It is called as an Oracle database, Oracle DB or Oracle marketed by Oracle. Oracle Database is being used by many companies in the IT industry for transaction processing, business analytics, business intelligence application purpose. Oracle database is developed in 1977 by Lawrence Ellison which is built around a relational database in which data can be accessed by users through an application or query language called SQL (structured query language). Oracle corporation first commercialized oracle RDBMS in 1979. Oracle database is available in different editions such as Enterprise edition, Standard edition, express edition and oracle lite. Oracle database runs on the major platforms like Windows, UNIX, Linux, and MacOS. The biggest rival for Oracle database is Microsoft SQL server.

### Features

* Active Data Guard
* Advanced Analytics
* Advanced Compression
* Advanced Security
* Database In-Memory
* Database Vault
* TimesTen Application-Tier Database Cache
* Label Security
* Management Packs
* Multitenant
* Online Analytical Processing (OLAP)
* Partitioning
* Real Application Clusters
* Real Application Clusters One Node
* Real Application Testing
* Spatial and Graph

## Microsoft SQL Server

Microsoft SQL Server is a relational database management system (RDBMS) developed by Microsoft. It is designed for enterprise-level data management, providing a comprehensive platform for managing and analyzing large volumes of data. SQL Server is widely used in businesses for various applications, including data warehousing, online transaction processing (OLTP), and business intelligence (BI).

### Features

* **High Performance:** SQL Server offers high performance with advanced indexing, query optimization, and in-memory processing technologies such as Columnstore indexes and in-memory OLTP.
* **Scalability:** Supports horizontal and vertical scaling, allowing it to handle large volumes of data and high transaction loads. It includes features like partitioning and parallel processing to manage scalability effectively.
* **Always On Availability Groups**: Provides high availability and disaster recovery with features like failover clustering and automatic failover to ensure continuous database operations.
* **Database Mirroring and Log Shipping**: Supports database mirroring and log shipping for backup and disaster recovery.
* **Comprehensive Data Types**: Supports a wide range of data types, including numeric, date/time, string, XML, JSON, and spatial data types.
* **Authentication and Authorization**: Supports Windows Authentication and SQL Server Authentication with granular role-based access control (RBAC).
* **T-SQL**: Uses Transact-SQL (T-SQL) as its query language, extending standard SQL with additional procedural programming features.

# Comparison Analysis

In this section we have done comparison of the tools/technologies described above.

## Openshift vs EKS vs AKS vs Docker Swarm

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Feature / Tool | OpenShift | EKS (Amazon Elastic Kubernetes Service) | AKS (Azure Kubernetes Service) | GKE (Google Cloud) | Comments |
| Architecture | 9 | 9 | 9 | 10 | GKE is Kubernetes-native and tightly integrated with Google infrastructure and AI tooling. |
| Scalability | 8 | 10 | 10 | 10 | GKE offers auto-pilot mode and advanced autoscaling (both nodes and pods). |
| Ease of Use | 7 | 8 | 8 | 9 | GKE is known for simplicity in cluster setup and intuitive UI/CLI. |
| Management Tools | 8 | 9 | 9 | 10 | GKE Dashboard, Cloud Console, and full integration with Cloud Monitoring make it strong here. |
| Deployment Options | 8 | 9 | 9 | 10 | GKE supports Autopilot, hybrid, edge (Anthos), and multi-region easily. |
| Security | 8 | 9 | 9 | 10 | GKE integrates with Binary Authorization, IAM, Shielded Nodes, and Workload Identity. |
| Integration with Cloud Services | 7 | 10 | 10 | 10 | GKE works seamlessly with BigQuery, Cloud Build, Pub/Sub, Vertex AI, etc. |
| Support and Community | 8 | 9 | 9 | 9 | GKE benefits from Google support and a strong developer community. |
| Cost | 7 | 8 | 8 | 9 | GKE Autopilot minimizes management cost and offers efficient resource-based pricing. |
| Total Score | **70** | **81** | **81** | **97** |  |

## PostgreSQL vs MySQL vs Oracle Database vs SQL Server

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Feature / Tool | PostgreSQL | MySQL | Oracle | SQL Server | Comments |
| Performance | 9 | 7 | 10 | 9 | Oracle excels in performance with advanced optimization features. SQL Server also offers high performance for various workloads. PostgreSQL provides high performance with advanced features, while MySQL is typically optimized for read-heavy workloads. |
| Community Support | 8 | 8 | 9 | 9 | PostgreSQL and MySQL benefit from strong community support, with PostgreSQL also having robust commercial options. Oracle and SQL Server offer excellent commercial support with extensive resources for enterprise environments. |
| Cost | 9 | 8 | 5 | 6 | PostgreSQL is free and open-source, making it cost-effective. MySQL offers a free community edition but has commercial options as well. Oracle and SQL Server are commercial products with significant licensing costs, though they offer powerful enterprise features. |
| Management | 7 | 7 | 9 | 9 | PostgreSQL and MySQL have good management tools but may require more manual configuration compared to Oracle and SQL Server, which offer comprehensive, enterprise-level management solutions. |
| Cloud Integration | 8 | 8 | 9 | 8 | PostgreSQL and MySQL have broad cloud support across major providers. Oracle has deep integration with its own cloud and strong support on other clouds. SQL Server integrates well with Azure and AWS but is more tightly coupled with Microsoft’s ecosystem. |
| Scalability | 8 | 8 | 10 | 9 | Oracle excels in scalability, handling large volumes and complex queries efficiently. SQL Server also offers high scalability, while PostgreSQL and MySQL are scalable but may require more effort for very large datasets. |
| Availability | 8 | 8 | 10 | 9 | Oracle provides the most robust availability features with RAC and Data Guard. SQL Server also has strong availability options. PostgreSQL and MySQL offer good availability but may need additional configurations and extensions. |
| Security | 9 | 7 | 10 | 9 | Oracle offers the most comprehensive security features. SQL Server also provides robust security options. PostgreSQL has strong security features, while MySQL’s security is good but often requires commercial editions for advanced features. |
| Total Score | **77** | **71** | **80** | **77** |  |

## 

# Solution overview

## Architecture

We will design a multi-tier Kubernetes-based system with:

* Service API Tier (Microservice)
  + Built using Spring Boot (Java) (or Node.js if preferred).
  + Exposes a GET REST API (/users) to fetch records.
  + Exposes a POST REST API (/users) to add records
  + Connects to the database tier via JDBC (Spring Data) with connection pooling.
  + Configurable database parameters (url, host, etc.) via Kubernetes ConfigMap and Secrets
  + Database password stored securely in Kubernetes Secrets.
  + Scaled to 4 pods with a Deployment for rolling updates.
  + Exposed externally using Kubernetes Ingress.
* Database Tier (PostgreSQL)
  + A PostgreSQL instance running as a StatefulSet (or Deployment if simplified).
  + Contains 1 table.
  + Uses a PersistentVolumeClaim to ensure data is retained across pod restarts.
  + Not exposed outside the cluster (ClusterIP service).
  + Database credentials stored in Secrets.

## Kubernetes Objects

* ConfigMap: Stores database host, port, database name, and other non-sensitive configs.
* Secret: Stores database username & password.
* PersistentVolume (PV) & PersistentVolumeClaim (PVC): Ensures database data persistence.
* Deployment (API Tier): Runs 4 replicas of the service with rolling updates enabled.
* Deployment/StatefulSet (Database Tier): Runs 1 replica of PostgreSQL with persistent storage.
* Service (ClusterIP): Used for API-to-DB communication.
* Service (Ingress): Exposes API to the outside world.
* Ingress: Routes external traffic to the microservice.

## Workflow

1. User calls API endpoint (http://<ingress-url>/users).
2. Microservice connects to PostgreSQL using credentials from Secrets & host info from ConfigMap.
3. PostgreSQL fetches data from the table and returns it to the service.
4. Service responds with JSON data.
5. Pods:
   * API pods are managed by a Deployment for high availability & rolling updates.
   * Database pod uses PVC to retain data across restarts.

## Resilience & Scaling

* API Tier:
  + 4 replicas (ensures zero downtime during updates and failures).
  + Auto-restart on failure.
* Database Tier:
  + 1 replica with a persistent volume to prevent data loss.

## Security

* No hard-coded passwords: Stored in Secrets.
* Configuration separation: Database credentials & connection info kept outside the application code in ConfigMaps.
* Pod-to-pod communication: Done via ClusterIP service names, not pod IPs.

## Deployment Flow

1. Build & push the API Docker image to Docker Hub.
2. Create Kubernetes manifests for API & Database (YAML files).
3. Apply ConfigMaps, Secrets, PV/PVC, Deployments, and Services.
4. Deploy Ingress to expose the API externally.
5. Verify functionality:
   * Fetch records from API.
   * Kill pods (API & DB) to verify auto-recovery and persistent data.

## Deliverables

* GitHub/GitLab Repository: Contains:
  + Application source code.
  + Dockerfiles.
  + Kubernetes YAML manifests.
  + README with instructions & URLs.
* Docker Hub: Contains pushed images.
* Screen Recording: Shows:
  + All Kubernetes objects running.
  + API call fetching DB records.
  + Pods regenerating after deletion.
  + Database pod restart without losing data.

# Recommendation

On the basis of the comaprisons mentioned above, recommended tools for the system are as follows:

* We’ll be using Microservices architecture to develop our microservices.
* We would recommend GCP cloud deployment options for microservices as GKE is the most feature-complete, scalable, and cloud-integrated platform.
* We would be recommending PostgreSQL over other databases due to various factors related to features, integration, and operational simplicity.

Some of the frameworks/tools that we are using but we have not described here, because there is no good alternate available for comparison are as follows:

* Spring boot framework for developing microservices.

# Assumptions

Assumption made during this analysis

1. The database isn’t huge and need no complex operations to be supported.
2. The application needs to be deployed on a single environment.

# Risks

Vulnerabilities of the open-source tools are publicly known.