# TrendLine Application: Kubernetes Architecture Summary

# Overview

The **TrendLine** e-commerce platform leverages **Kubernetes** to manage the deployment, scaling, and maintenance of its microservices-based architecture. The application is deployed in three isolated environments—**development**, **staging**, and **production**—each hosted in its own Kubernetes namespace.

This separation ensures clear workflows for development, testing, and release management while enabling proper resource allocation and access control.

# Architecture Components

1. Frontend

The frontend is a web-based user interface developed using React. It is packaged as a container image and deployed as a **Kubernetes Deployment**.

* Stateless and horizontally scalable.
* Exposes port 3001 and communicates with the backend via services.
* Deployed with resource constraints and configured differently in each environment.

1. Backend

The backend is a RESTful API built with ASP.NET Core, responsible for business logic, user authentication, product management, and order processing.

* Deployed as a **Deployment** in Kubernetes.
* Communicates with the PostgreSQL database and Solr search engine.
* Environment-specific variables like connection strings are managed via Kubernetes ConfigMaps and Secrets.
* Exposes port 5000.

1. Database (PostgreSQL)

PostgreSQL is deployed as a **StatefulSet**, allowing stable network identities and persistent storage.

* Uses **PersistentVolumeClaims (PVCs)** to ensure data durability.
* Each pod maintains its own disk, enabling recovery after restarts.
* Exposes port 5432.
* One replica is used in dev for simplicity, while for staging and prod, three replicas each are deployed.

1. Search Engine (Apache Solr)

Apache Solr is used for indexing and full-text product search.

* Deployed as a **StatefulSet** with a dedicated PersistentVolume.
* Communicates with the backend API via internal service discovery.
* Exposes port 8983.

1. Monitoring (Prometheus & Grafana)

Observability is achieved through **Prometheus** for metrics collection and **Grafana** for dashboarding.

* Both are deployed as **Deployments**.
* Grafana connects to Prometheus via a Kubernetes service.

# Kubernetes Features

* **Deployments** are used for stateless components like frontend and backend, allowing rolling updates and scaling.
* **StatefulSets** are used for PostgreSQL and Solr to manage stable storage and identity.
* Each component is exposed via a **NodePort service** for external communication outside of the cluster. Only the databases are exposed via **ClusterIP** for internal communication only.
* **Horizontal Pod Autoscaling (HPA)** is enabled in **staging** and **production** environments. HPA’s automatically scale pods based on **CPU or memory usage** thresholds for example: backend service in production can scale from 3 to 10 pods based on load.
* **Vertical Pod Autoscaling (VPA)** is deployed in **recommendation mode** for staging and production to suggest optimal CPU/memory for pods. This helps refine resource requests over time based on usage patterns.
* **Resource Quotas** are also deployed: **Development** is limited to 1 CPU and 4GB RAM to avoid resource exhaustion. **Staging** has more than double the resources of development. **Production** has **no resource limits** to support full scaling.
* **Persistent Volumes** are used with PVCs to store data for PostgreSQL and Solr. Volumes are retained even if pods are deleted, ensuring data persistence.