Online Bookstore Microservices Project

Project Progress Report - Checkpoint 3

Prepared For: Prof. Pedram Habibi

Prepared By: Group 9

Group Members:

Helly Rajeshbhai Patel Jiyad Mohammed

Arif Shaikh

Nicholas Nwanua Ilechie

Nirajbhai Ranchhodbhai Limbasiya

Date Submitted: July 24th, 2025

"I declare that the attached assessment is wholly my own work in accordance with Seneca Academic Policy. No part of this assignment has been copied manually or electronically from any other source (including web sites) or distributed to other students."

Introduction

This report presents the work completed for Checkpoint 3 of the Service-Oriented Architecture (SOA) course project. We deployed the User and Product Services to a local Kubernetes cluster using Minikube. System monitoring was implemented using Prometheus and Grafana, and a Horizontal Pod Autoscaler (HPA) was configured for the User Service to enhance scalability. Networking between pods was tested to ensure reliable inter-service communication. All resources were organized within the dedicated bookstore namespace, and resource specification issues were resolved to achieve stable operation. This introduction outlines the scope of work and deliverables for Phase 1 of the online bookstore microservices project, including kubectl outputs, monitoring dashboards, and networking validation results.

Kubernetes Deployment

We deployed the User and Product Services in the bookstore namespace using Kubernetes manifests, including Deployment, Service, ConfigMap, and Secret objects.

User Service

The User Service, developed using Node.js and Express.js, provides functionalities for user registration and retrieval through the following endpoints: POST /users/register, GET /users, and GET /users/{id}. The service is deployed with two replicas to enhance availability and resilience. Resource requests and limits have been defined to ensure efficient allocation of CPU and memory resources, preventing starvation and maintaining stability.

An excerpt from the deployment configuration (user-service-deployment.yaml) is shown below:

```
apiVersion: apps/v1
kind: Deployment
metadata:
name: user-service
namespace: bookstore
spec:
replicas: 2
selector:
 matchLabels:
 app: user-service
template:
 metadata:
  app: user-service
 spec:
  containers:
   - name: user-service
   image: user-service:latest
    - containerPort: 3000
    resources:
     cpu: "500m"
     memory: "512Mi"
     requests:
     cpu: "200m"
     memory: "256Mi"
```

Product Service

The Product Service is responsible for managing the book catalog and supports endpoints for catalog retrieval: GET /books and GET /books/{id}. Like the User Service, it is deployed with two replicas and identical resource configurations to ensure consistency, scalability, and stable performance.

Horizontal Pod Autoscaling

A Horizontal Pod Autoscaler (HPA) was configured for the User Service to ensure scalability under variable workloads. The HPA automatically adjusts the number of replicas between 2 and 5 based on a target CPU utilization of 70%, utilizing the Minikube metrics server for real-time resource monitoring.

An excerpt from the deployment configuration (user-service-hpa.yaml) is shown below:

apiVersion: autoscaling/v2

kind: HorizontalPodAutoscaler

metadata:

name: user-service-hpa namespace: bookstore

spec:

scaleTargetRef:

apiVersion: apps/v1 kind: Deployment name: user-service

minReplicas: 2 maxReplicas: 5

metrics:

- type: Resource

resource: name: cpu target:

type: Utilization

averageUtilization: 70

Monitoring

Prometheus was configured to scrape metrics from both the User Service and Product Service at 15-second intervals, with settings managed via a dedicated ConfigMap.

Grafana was integrated for visualization, providing real-time monitoring through custom dashboards. These dashboards are accessible via NodePort services, enabling performance tracking and health assessment of the deployed services.

An excerpt of the Prometheus configuration (prometheus-config.yaml) is shown below:

```
data:

prometheus.yml: |

global:

scrape_interval: 15s

scrape_configs:

- job_name: 'user-service'

static_configs:

- targets: ['user-service:80']

- job_name: 'product-service'

static_configs:

- targets: ['product-service:80']
```

5. Networking

Inter-service communication was validated using a BusyBox pod, as the node:18-alpine image does not include curl. The following command confirmed that the User Service could be reached from within the cluster:

```
kubectl run -i --tty test-pod --image=busybox --restart=Never -n bookstore -- sh

wget -q -O- http://user-service:80/users

Output:

[{"id": "1", "username": "johndoe", "email": "john@example.com"}, ...]
```

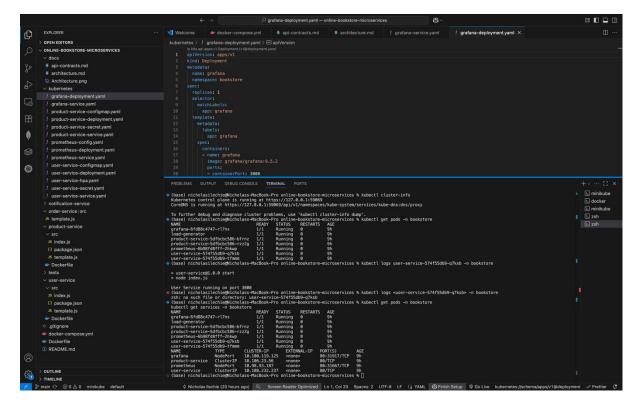
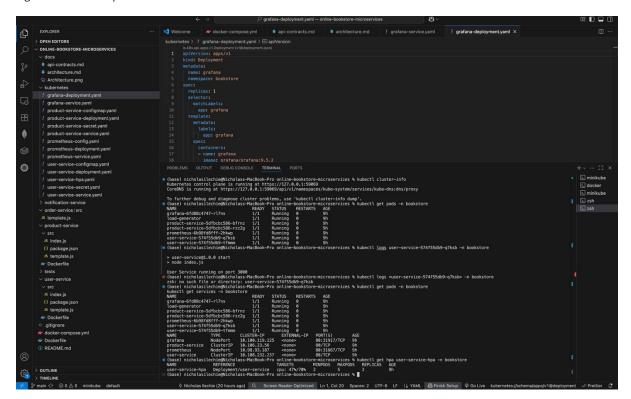


Figure: kubectl Outputs



```
TERMINAL
PROBLEMS
                                    DEBUG CONSOLE
configmap/user-service-config created
deployment.apps/user-service created
horizontalpodautoscaler.autoscaling/user-service-hpa created
secret/user-service-secret created
service/user-service created
(base) nicholasilechie@Nicholass-MacBook-Pro online-bookstore-microservices % kubectl get pods -n bookstore
kubectl get services —n bookstore
NAME
                                                         READY
                                                                      STATUS
                                                                                      RESTARTS
                                                                                                        AGE
                                                                                                        18s
18s
grafana-6fd88c4747-rl7ns
product-service-5dfbcbc586-bfrnz
product-service-5dfbcbc586-rzz2g
                                                         1/1
1/1
1/1
1/1
1/1
1/1
                                                                      Running
                                                                      Running
                                                                      Running
                                                                                                        18s
prometheus-6b98fd8fff-2hkwp
user-service-574f55db9-q7ksb
user-service-574f55db9-tfmmm
                                                                                      0
                                                                                                        18s
                                                                      Running
                                                                      Running
                                                                                                        18s
                                                                                                        18s
                                                                      Running
```

Figure: Kubernetes Cluster

Conclusion

Checkpoint 3 achieved the successful deployment of the User and Product Services to a Kubernetes cluster, complete with monitoring, autoscaling, and networking capabilities. Resource configuration errors were resolved, and all components were organized within the dedicated bookstore namespace to ensure stability and maintainability.

With these foundational services in place, we are now prepared to advance to Phase 2, which will involve the implementation of Order and Notification Services and migration of the deployment to a cloud provider. We welcome constructive feedback to further optimize our implementation.