PUBLICIS SAPIENT

BOOKMYMOVIE

RUNBOOK DOCUMENT

Version: 1.0

Effective Date: July-24, 2024

Author: Palash Chanda

Statement of Confidentiality

The information contained herein shall not be disclosed, duplicated, or used in whole or in part for any purpose other than to evaluate the proposal, provided that if a contract is awarded to this offer as a result of, or in connection with, the submission of these information, the recipient shall have the right to duplicate, use or disclose the information to the extent provided in the agreement. This restriction does not limit the right to use information contained in the data if it is obtained from another source without restriction.

Contents

About This Guide

This document is intended for understanding the details of Kubernetes process, Microservice Ports of Book-My-Movie Project.

Audience

This guide is intended for the Book My Movie Development team.

Related Documents

Document title	Owner/Link	Version
High Level Design Document	Architect	V1.0

Document History

Created on	Created by	Changed on	Changed by	Reviewed by	Version
24/07/2024	Palash Chanda	24/07/2024	Palash Chanda	TBD	V1.0
		26/07/2024	Palash Chanda	TBD	V2.0

Microservice Port Mapping

Microservice Server ports

Microservice	Port
bookmymovie-orchestrator	8081
bookmymovie-theater	8082
bookmymovie-cinema	8083
bookmymovie-viewer	8084
bookmymovie-order	8085
bookmymovie-payment	8086
bookmymovie-notification	8087
bookmymovie-utility	8088
bookmymovie-auth	8089

Microservice Datastore Ports

Microservice	Port
bookmymovie-orchestrator	9081
bookmymovie-theater	9082
bookmymovie-cinema	9083
bookmymovie-viewer	9084
bookmymovie-order	9085
bookmymovie-payment	9086
bookmymovie-notification	9087
bookmymovie-utility	9088
bookmymovie-auth	9089

Microservice Deployment

Minukube Deployment

Deployment in Minikube (Kubernetes Cluster)/Docker/Helm. Steps:

- Start Docker Desktop:
  1. Check Docker installation status: `docker –v` OR `docker info`
  2. Check Docker running status: `wls -l -v`

```
Admin@DESKTOP-SGDOOU5 MINGW64 ~
$ wsl -l -v
  NAME              STATE           VERSION
* docker-desktop    Running         2
```

- Install Kubectl: Check status: `kubectl version` OR `kubectl version --client -o json`
- Install Minikube (Kubernetes Cluster).
- Install Istio Daemon inside the Minikube.
- Install/configure Helm.
- Run below Commands:

```
minikube start
minikube docker-env
@FOR /f "tokens=*" %i IN ('minikube -p minikube docker-env --shell cmd') DO @%i
minikube tunnel //Run it in different terminal
kubectl get service istio-ingressgateway -n istio-system
minikube status
kubectl cluster-info
kubectl get nodes -o wide
kubectl create namespace bookmymovie-namespace
kubectl label namespace bookmymovie-namespace istio-injection=enabled
kubectl get namespace -L istio-injection
cd /d F:\Workspace-IntelliJ-POC\bookmymovie-service\bookmymovie-scripts\kubernetes\local
kubectl apply -f gateways.yaml -n bookmymovie-namespace
kubectl get gateway bookmymovie-istio-gateway -n bookmymovie-namespace
```

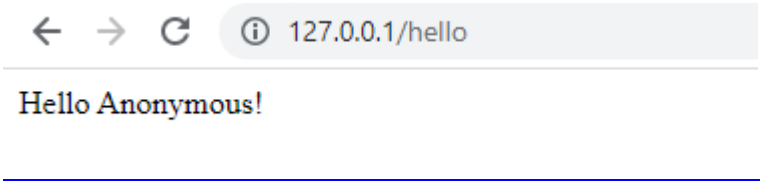
- Build Orchestrator Microservice: build-orchestrator.cmd
- Deploy Orchestrator Microservice: kubernetes-orchestrator.bat
- Validate Orchestrator Microservice:
  Datastore: http://127.0.0.1:80/orchReqTracker
  Smoke: http://127.0.0.1:80/smoke
  Swagger: http://127.0.0.1:80/swagger-ui/index.html
- Similarly Build/Deploy other Microservices: Theater, Cinema, Order, Payment, Viewer: TBD.
- Test Application: If you want to use the host's domain name in the URL (e.g. - http://httpbin.example.com:80/hello), you must configure the requested host properly and DNS resolvable. So here you can access as: http://$INGRESS_HOST:$INGRESS_PORT/hello
  1. Get INGRESS_HOST: `echo INGRESS_HOST=$(kubectl -n istio-system get service istio-ingressgateway -o jsonpath='{.status.loadBalancer.ingress[0].ip}')`

```
iamsabre+SG0313286@AADS5CD127784S MINGW64 ~
$ echo INGRESS_HOST=$(kubectl -n istio-system get service istio-ingressgateway -o jsonpath='{.status.loadBalancer.ingress[0].ip}')
INGRESS_HOST=127.0.0.1
```

  2. Get INGRESS_PORT: `echo INGRESS_PORT=$(kubectl -n istio-system get service istio-ingressgateway -o jsonpath='{.spec.ports[?(@.name=="http2")].port}')`

```
iamsabre+SG0313286@AADS5CD127784S MINGW64 ~
$ echo INGRESS_PORT=$(kubectl -n istio-system get service istio-ingressgateway -o jsonpath='{.spec.ports[?(@.name=="http2")].port}')
INGRESS_PORT=80
```

3. Open API URL in browser: http://127.0.0.1:80/hello



- Notes:
  1. Minikube has a separate Docker Daemon runs inside the Minikube Virtual Machine, which is independent of Docker installed and running on the host

Local Execution

Theater Microservice:
gcloud beta emulators datastore start --project=bookmymovie-db --host-port=0.0.0.0:9082 --data-dir=F:\Program-Files\Firestore-Emulator-Data-Dir
Datastore ping: http://localhost:9082/
Datastore:
http://localhost:8082/city
http://localhost:8082/theater
http://localhost:8082/screen
http://localhost:8082/seat
Smoke: http://localhost:8082/smoke/
Swagger: http://localhost:8082/swagger-ui/index.html

Movie Microservice:
gcloud beta emulators datastore start --project=bookmymovie-db --host-port=0.0.0.0:9083 --data-dir=F:\Program-Files\Firestore-Emulator-Data-Dir
Datastore ping: http://localhost:9083/
Datastore:
http://localhost:8083/movie
http://localhost:8083/movieshow
Smoke: http://localhost:8083/smoke/
Swagger: http://localhost:8083/swagger-ui/index.html

Viewer Microservice:
gcloud beta emulators datastore start --project=bookmymovie-db --host-port=0.0.0.0:9084 --data-dir=F:\Program-Files\Firestore-Emulator-Data-Dir
Datastore ping: http://localhost:9084/
Smoke: http://localhost:8084/smoke/
Datastore: http://localhost:8084/viewer
Swagger: http://localhost:8084/swagger-ui/index.html

Orchestrator Microservice:
gcloud beta emulators datastore start --project=bookmymovie-db --host-port=0.0.0.0:9081 --data-dir=F:\Program-Files\Firestore-Emulator-Data-Dir
Datastore ping: http://localhost:9081/
Datastore: http://localhost:8081/orchReqTracker
Smoke: http://localhost:8081/smoke
Swagger: http://localhost:8081/swagger-ui/index.html

Order Microservice:
gcloud beta emulators datastore start --project=bookmymovie-db --host-port=0.0.0.0:9085 --data-dir=F:\Program-Files\Firestore-Emulator-Data-Dir
Datastore ping: http://localhost:9085/
Datastore: http://localhost:8085/order
Smoke: http://localhost:8085/smoke/
Swagger: http://localhost:8085/swagger-ui/index.html

Payment Microservice:
gcloud beta emulators datastore start --project=bookmymovie-db --host-port=0.0.0.0:9086 --data-dir=F:\Program-Files\Firestore-Emulator-Data-Dir
Datastore ping: http://localhost:9086/
Datastore: http://localhost:8086/payment
Smoke: http://localhost:8086/smoke/
Swagger: http://localhost:8086/swagger-ui/index.html

Notification Microservice:
gcloud beta emulators datastore start --project=bookmymovie-db --host-port=0.0.0.0:9087 --data-dir=F:\Program-Files\Firestore-Emulator-Data-Dir
Datastore ping: http://localhost:9087/
Smoke: http://localhost:8087/smoke /
Swagger: http://localhost:8087/swagger-ui/index.html

Utility Microservice:
gcloud beta emulators datastore start --project=bookmymovie-db --host-port=0.0.0.0:9088 --data-dir=F:\Program-Files\Firestore-Emulator-Data-Dir
Datastore ping: http://localhost:9088/
Smoke: http://localhost:8088/smoke/
Swagger: http://localhost:8088/swagger-ui/index.html

Auth Service:
gcloud beta emulators datastore start --project=bookmymovie-db --host-port=0.0.0.0:9089 --data-dir=F:\Program-Files\Firestore-Emulator-Data-Dir
Datastore ping: http://localhost:9089/
Smoke: http://localhost:8089/smoke/
Swagger: http://localhost:8089/swagger-ui/index.html

Metrics Validation

Micrometer Metrics validation:

- Check Metrics: http://localhost:7081/actuator/metrics


```
{
  "names": [
    "bmm.orch.http.booking.errcount",
    "bmm.orch.http.orchtoorder.errcount",
    "jvm.buffer.count",
    "jvm.buffer.memory.used",
    "jvm.buffer.total.capacity",
    "jvm.classes.loaded",
    "jvm.classes.unloaded",
    "jvm.gc.live.data.size",
    "jvm.gc.max.data.size",
    "jvm.gc.memory.allocated",
    "jvm.gc.memory.promoted",
    "jvm.gc.pause",
    "jvm.memory.committed",
    "jvm.memory.max",
    "jvm.memory.used",
    "jvm.threads.daemon",
    "jvm.threads.live",
    "jvm.threads.peak",
    "jvm.threads.started",
    "jvm.threads.states",
    "logback.events",
    "process.cpu.time",
    "process.cpu.usage",
    "process.start.time",
    "process.uptime",
    "system.cpu.count",
    "system.cpu.usage",
    "tomcat.sessions.active.current",
    "tomcat.sessions.active.max",
    "tomcat.sessions.alive.max",
    "tomcat.sessions.created",
    "tomcat.sessions.expired",
    "tomcat.sessions.rejected"
  ]
}
```


- Check execution: http://localhost:7081/actuator/metrics/bmm.orch.http.orchtoorder.errcount


```
{
  "name": "bmm.orch.http.orchtoorder.errcount",
  "description": "Counts Exception while sending request from Orchestrator to Order",
  "baseUnit": null,
  "measurements": [
    {
      "statistic": "COUNT",
      "value": 1
    }
  ],
  "availableTags": [
    {
      "tag": "environment",
      "values": [
        "LOCAL"
      ]
    }
  ]
}
```

Monorepo

Github Monorepo: https://github.com/palashrc/bookmymovie-service