

K8S OpenShift מבחן – Terrorists API

פרטי המבחן

משך זמן: 4 שעות

ניקוד כולל: 100 נקודות

הגשה: קישור לרפוזיטורי GitHub שמוגש למטלה הרלוונטית במודל

תרחיש

כמה חודשים עברו אחרי שיוסף התחיל לעבוד ביחד עם המוסד כדי לאתר מחבלים ימח שימם וצה"ל ביחד עם חברת רפאל פיתחו מל"ט חדש וייחודי שיועד להתאבד על מטרה ספציפית ולחסל את המטרה. הבעיה היא שיש רק מספר מוגבל של מל"טים שאפשר לפתח בחודש לכן נוצר דרישה למצוא את המחבלים הכי מסוכנים שעליהם יופעלו המל"טים.

אתם בונים מיקרו-שירות לקליטת נתונים של מחבלים עבור יחידות המודיעין של צה"ל. השירות מקבל קובץ CSV המכיל רשומות של מחבלים, מעבד את הנתונים באמצעות pandas, מדרג אותם לפי רמת מסוכנות, מחזיר את 5 המחבלים המסוכנים ביותר, ושומר את התוצאות ב-MongoDB. כל המערכת חייבת להיות ניתנת לפריסה ב-OpenShift עם 3 רפליקות של MongoDB ו-3 רפליקות של ה-API.

חומרי עזר מותרים:

אין להשתמש בקוד עבר שכתבתם או חיפוש בגוגל אלא במקורות הבאים:

[W3 Schools](#)

[GeeksforGeeks](#)

[Real Python](#)

[Stack Overflow](#) (שימוש במנוע חיפוש פנימי ולא באמצעות גוגל)

official docs:

[Docker Docs](#)

[Kubernetes Documentation](#)

[OpenShift Container Platform | 4.20 | Red Hat Documentation](#)

YAML בדיקת מבנה - [The YAML Validator](#)

[pandas 2.3.3 documentation](#)

[Tutorial - User Guide - FastAPI](#)

- HTTP Post request with a csv file:
[How to write a FAST API function taking .csv file and making some preprocessing in pandas - Stack Overflow](#)

[Models - Pydantic Validation](#)

חלק 1: דרישות אפליקציית FastAPI

1.1 הגדרת Endpoint

יש ליצור אפליקציית FastAPI עם endpoint אחד בלבד:

Endpoint:

POST /top-threats

פורמט הבקשה:

- Content-Type: `multipart/form-data`
- שם השדה: `file`
- סוג הקובץ: CSV

1.2 תהליך העיבוד (Processing Pipeline)

יש לממש את השלבים הבאים בדיוק לפי הסדר:

- קריאת קובץ ה-CSV לתוך DataFrame של pandas
- מיון לפי `danger_rate` בסדר יורד
- בחירת 5 השורות העליונות
- ולידציה והמרה באמצעות מודלי Pydantic כך שיישארו רק השדות:
 - `name` (מחרוזת)
 - `location` (מחרוזת)
 - `danger_rate` (מספר שלם, 1–10)

1.3 פורמט התגובה

יש להחזיר JSON במבנה המדויק הבא:

```
{
  "count": 5,
  "top": [
    {"name": "Mohammad Abu Maniac", "location": "Kuan Yunes", "danger_rate": 10},
    {"name": "Hassan Balboah", "location": "Gaza City", "danger_rate": 9},
    {"name": "Ahmed Aiben Imashlo", "location": "Rafah", "danger_rate": 8},
    {"name": "Sharif Hammudi", "location": "Sagaiyah", "danger_rate": 7},
    {"name": "Zoran Mamdani", "location": "Lebanon City", "danger_rate": 6}
  ]
}
```

חשוב:

- אם לאחר העיבוד קיימות פחות מ-5 שורות תקינות – יש להחזיר את מספר הרשומות הקיים
- יש לעדכן את השדה `count` בהתאם למספר הרשומות שהוחזר בפועל

1.4 שמירה ב־MongoDB

לאחר יצירת התגובה:

1. יש להכניס (`Insert`) את כל הרשומות מתוך המערך `top` ל-MongoDB
2. שם האוסף (`Collection`): `top_threats`
3. מבנה המסמך:

```
{
  "name": "Alpha",
  "location": "Metropolis",
  "danger_rate": 10
}
```

1.5 משתני סביבה

אפליקציית FastAPI חייבת לקרוא את משתני הסביבה הבאים:

משתנה	ערך לדוגמה	תיאור
MONGO_HOST	mongo-0.mongo	שם ה- <code>hostname</code> של הפוד הראשון של MongoDB
MONGO_PORT	27017	פורט MongoDB
MONGO_USERNAME	admin	שם משתמש <code>root</code> של MongoDB
MONGO_PASSWORD	secretpass	סיסמת <code>root</code> של MongoDB
MONGO_DB	threat_db	שם בסיס הנתונים
MONGO_AUTH_SOURCE	admin	בסיס הנתונים לאימות

1.6 טיפול בשגיאות

יש לממש טיפול נכון בשגיאות עם קודי HTTP מתאימים:

הודעת תגובה	קוד HTTP	מצב שגיאה
<code>{"detail": "No file provided"}</code>	400	לא סופק קובץ
<code>{"detail": "Invalid CSV file"}</code>	400	הקובץ אינו CSV / שגיאת פרסור
<code>detail": "Database"</code> <code>{"unavailable</code>	503	MongoDB אינו נגיש
פירוט שגיאת ולידציה של Pydantic	422	פורמט נתונים שגוי

1.7 רשימת טכנולוגית

- גרסת Python: 3.11 ומעלה
 - ספריות נדרשות:
 - `fastapi`
 - `pandas`
 - `pydantic`
 - `pymongo`
 - `python-multipart` (להעלאת קבצים)
 - `uvicorn` (שרת ASGI)
-

חלק 2: דרישות Docker

2.1 קבצים נדרשים

שורש הרפוזיטורי חייב לכלול:

- `Dockerfile`
 - `dockerignore.`
 - `gitignore.`
-

2.2 דרישות (חובה) Dockerfile

דרישות חובה:

- Image בסיס: `python:3.11-slim` או דומה
 - מיקום קוד האפליקציה: `app/`
 - פורט חשוף: `8000`
 - פקודת הפעלה: הרצת `uvicorn` על `0.0.0.0:8000`
 - העתקת `requirements.txt` תחילה, התקנת תלויות, ולאחר מכן העתקת שאר קוד האפליקציה
-

2.3 קובץ .dockerignore

חייב להחריג לפחות:

- `__pycache__/`
 - `*.pyc`
 - `.git/`
 - `k8s/`
 - `.gitignore`
 - `README.md`
 - `screenshots/`
-

חלק 3: דרישות פריסה ב-OpenShift

כל קבצי ה-YAML של Kubernetes חייבים להיות פרוסים על Cluster של OpenShift.

MongoDB StatefulSet 3.1

קובץ: `k8s/mongo-statefulset.yaml`

הגדרות בסיסיות:

מאפיין	ערך
שם	<code>mongo</code>
מספר רפליקות	<code>3</code>
שם השירות	<code>mongo</code>
Image	<code>docker.io/library/mongo:7</code>
פורט קונטיינר	<code>27017</code>

:Labels

מפתח	ערך
<code>app</code>	<code>mongo</code>
<code>tier</code>	<code>database</code>

משתני סביבה:

שם משתנה	ערך
<code>MONGO_INITDB_ROOT_USERNAME</code>	<code>admin</code>
<code>MONGO_INITDB_ROOT_PASSWORD</code>	<code>secretpass</code>
<code>MONGO_INITDB_DATABASE</code>	<code>threat_db</code>

:Persistent Volume Claim (volumeClaimTemplates

מאפיין	ערך
שם ה-Claim	mongo-data
Access Mode	ReadWriteOnce
בקשת אחסון	1Gi

:Volume Mount

מאפיין	ערך
שם Volume	mongo-data
נתיב Mount	data/db/

3.2 שירות MongoDB

קובץ: k8s/mongo-service.yaml

הגדרות:

מאפיין	ערך
שם	mongo
סוג	ClusterIP
ClusterIP	(None (headless
פורט	27017
Target Port	27017

:Labels

מפתח	ערך
app	mongo
tier	database

:Selector

מפתח	ערך
app	mongo
tier	database

3.3 פריסת API

קובץ: `k8s/api-deployment.yaml`

הגדרות בסיסיות:

מאפיין	ערך
שם	threat-api
רפליקות	3
Image	your-dockerhub-username>/threat-api:v1>
פורט קונטיינר	8000

:Labels

מפתח	ערך
app	threat-api
tier	backend

משתני סביבה:

ערך	שם משתנה
mongo-0.mongo	MONGO_HOST
27017	MONGO_PORT
admin	MONGO_USERNAME
secretpass	MONGO_PASSWORD
threat_db	MONGO_DB
admin	MONGO_AUTH_SOURCE

3.4 שירות API

קובץ: k8s/api-service.yaml

הגדרות:

ערך	מאפיין
threat-api-svc	שם
NodePort	סוג
8000	פורט
8000	Target Port
30080	Node Port

:Labels

ערך	מפתח
threat-api	app
backend	tier

:Selector

מפתח	ערך
app	threat-a
	pi
tier	backend

חלק 4: מבנה הרפוזיטורי

הרפוזיטורי חייב להיות במבנה הבא בדיוק:

- main.py
- models.py
- db.py
- requirements.txt
- k8s/
 - mongo-statefulset.yaml
 - mongo-service.yaml
 - api-deployment.yaml
 - api-service.yaml
- screenshots/
 - Pods.png
 - openshift-ui.png
 - api-request.png
- commands.txt
- Dockerfile
- .dockerignore
- .gitignore
- README.md

חלק 5: דרישות קובץ Commands

יש ליצור קובץ בשם `commands.txt` שיופיע בענף הראשי של הרפוזיטורי המכיל את כל הפקודות הדרושות לבנייה ולפריסה של האפליקציה. הפקודות חייבות לעבוד ברצף כאשר מריצים אותן לפי הסדר.

חשוב:

אין לכלול פקודות `git`. ופקודות של מערכת ההפעלה (כגון יצירת תיקיות וכו) יש לכלול רק פקודות `Docker` ו-`OpenShift` (`oc`).

חלק 6: תהליך עבודה עם Git

6.1 Branches נדרשים

הרפוזיטורי חייב להכיל:

1. `main` – פתרון סופי עובד
2. `feature/app` – קוד אפליקציית FastAPI
3. `feature/k8s` – קבצי Kubernetes

6.2 תהליך פיתוח

1. יצירת רפוזיטורי ו-`commit` ראשוני על `main`
 2. יצירת `branch feature/app`
 - פיתוח אפליקציית FastAPI
 - `commits` תכופים עם הודעות משמעותיות
 - מיזוג ל-`main` בסיום
 3. יצירת `branch feature/k8s`
 - פיתוח manifests של Kubernetes
 - `commits` תכופים
 - מיזוג ל-`main` בסיום
-

חלק 7: דרישות צילומי מסך

יש לכלול 3 צילומי מסך בתיקית `screenshots` המאשרים שהאפליקציה רצה ב-OpenShift:

7.1 צילום 1: סטטוס פודים (`pods.png`)

צילום פלט הפקודה:

```
oc get pods
```

חייב להציג:

- כל פודי mongo-0, mongo-1, mongo-2 (MongoDB) במצב Running
 - כל פודי ה-API במצב Running
 - כל Init Jobs הסתיימו בהצלחה
-

7.2 צילום 2: ממשק (`openshift-ui.png`) OpenShift

צילום ממסך ה-Web Console של OpenShift.

חייב להציג:

- תצוגת Topology או Workloads
 - Deployments ו-StatefulSets גלויים
 - אינדיקציה ברורה שהמשאבים רצים
-

7.3 צילום 3: בקשת (`api-request.png`) API

צילום של בקשת HTTP מוצלחת ל-Route של OpenShift.

חייב להציג:

- URL מלא של ה-Route
- שיטת הבקשה (POST ל-`top-threats`)
- תגובה עם 5 הרשומות בפורמט JSON
- קוד סטטוס HTTP 200

כלים מותרים:

curl, Postman, REST Client, או כל כלי בדיקות API.

חלק 8: דרישות README.md

קובץ README.md חייב לכלול:

Threat Analysis Microservice

**שם הסטודנט: [שם מלא]

**תעודת זהות / מזהה סטודנט: [ID]

**קורס / כיתה: [שם הקורס]

הוראות הגשה

1. יצירת רפוזיטורי GitHub בשם `terrorists-api-k8s`
 2. דחיפת הפתרון המלא ל-`main` branch
 3. וידוא שכל ה-`branches` נדחפו לרימוט (`main`, `feature/app`, `feature/k8s`)
 4. הגשת קישור לרפוזיטורי דרך מערכת הקורס
-

הערות חשובות

- יש לבדוק היטב לפני ההגשה – פתרון שלא עובד יגרור הורדת נקודות משמעותית
 - יש לבצע **commits** תכופים עם הודעות משמעותיות
 - יש לוודא שכל קבצי ה-YAML תקינים ומעוצבים נכון
 - יש לוודא שכל משתני הסביבה מוגדרים נכון הן בטבלאות והן בפריסות
-

בהצלחה!