GraalVM™

מה זה GraalVM

- GraalVM היא סביבת ריצה מתקדמת מבוססת OpenJDK.
- תומכת בהרצת יישומי Java ובשפות נוספות (JavaScript, Python, Ruby ועוד).
- כוללת שני מצבי ריצה עיקריים:
- JIT (Just-In-Time) עם Graal Compiler
- AOT (Ahead-Of-Time) באמצעות Native Image

JIT (Just-In-Time) עם Graal Compiler

מה זה JIT (Just-In-Time)?

- בתהליך JIT, קוד הבייטקוד – הקוד הביניים שנוצר מתרגום הקוד המקורי בשפת Java – מתורגם לקוד מכונה (כלומר, קוד שהמעבד יכול להריץ ישירות) בזמן הריצה של היישום.
- התרגום מתבצע "בזמן אמת": הקוד מתורגם רק כאשר הוא נדרש, מה שמאפשר לשפר את ביצועי היישום תוך כדי הריצה.

היתרונות ב־GraalVM JIT לעומת OpenJDK:

- קומפיילר Graal מבצע אופטימיזציות מתקדמות יותר (למשל, ניתוח גרפי של הקוד).
- שימוש טוב יותר בזיכרון – Graal Compiler יעיל יותר מה־C2 הישן של HotSpot.
- ביצועים גבוהים יותר באפליקציות ארוכות טווח – לאחר "התחממות" הקוד, הוא רץ מהר יותר.
- תומך בטכנולוגיות מודרניות: שילוב עם Truffle, הרצת שפות נוספות, התאמה לעולם הענן

AOT – Ahead-Of-Time באמצעות Native Image

– בתהליך זה, קוד הבייטקוד מתורגם מראש לקוד מכונה לפני שהיישום יורץ.

איך Native Image עובד?

- **שלב 1: קוד Java רגיל**

כותבים את הקוד כרגיל ב-Java, ומקמפלים אותו עם javac -> קבצי .class.

- **שלב 2: יצירת Native Image**

קובץ הרצה עצמאי שמכיל את כל האפליקציה והספריות שלה
מרצים את הפקודה: native-image MyApp

- **שלב 3: GraalVM מנתח את הקוד**

  - הוא אוסף את כל המחלקות, התלויות והספריות.
  - מזהה אילו חלקים דרושים בריצה.
  - מבצע **tree shaking** – מסיר קוד לא רלוונטי.

- **שלב 4: בניית קובץ הרצה**

  - GraalVM יוצר קובץ בינארי עצמאי (כמו .exe או ELF ב־Linux).
  - לא צריך JVM, ולא צריך קובצי .class – הכל כלול בתוך קובץ אחד!

AOT – Ahead-Of-Time באמצעות Native Image

- **מה היתרונות?**
  - **זמן אתחול מהיר:** הקובץ המוגמר עולה מהר מאוד.
  - **צריכת זיכרון נמוכה:** אין צורך להריץ JVM שלם, מה שמקטין את השימוש בזיכרון.
- **מה החסרונות?**
  - תהליך הקומפילציה מראש יכול לקחת זמן רב יותר.
  - חלק מהתכונות הדינמיות של Java מוגבלות או דורשות תצורה מיוחדת.

Truffle & Polyglot – עבודה עם שפות רבות בסביבת GraalVM

- Truffle היא מסגרת ב־GraalVM לבניית interpreters לשפות תכנות שונות.
- לכל שפה (JavaScript, Python, Ruby ועוד) יש interpreter ייעודי שנכתב במיוחד עבור GraalVM באמצעות Truffle.
- interpreters אינם המנועים המקוריים של השפות (לדוג' CPython או V8) אלא נכתבו מחדש.

Polyglot - עבודה מרובת שפות:

- מאפשר הרצת קוד ממספר שפות באותו תהליך ובאותה סביבת ריצה.
- ניתן לגשת ממקור קוד אחד (לדוג' Java) לקוד בשפה אחרת (Python, JS וכו').
- אפשר גם להריץ קוד Java מתוך Python או JavaScript (דו־כיווני).
- כל השפות מנוהלות ביעילות דרך מנגנון משותף ומאפשרות שיתוף נתונים קל.

למה זה שימושי?

- מאפשר לצוותים לעבוד עם שפות שונות באותה מערכת.
- מנצל את היתרונות של כל שפה.
- חוסך צורך בתקשורת בין־תהליכית (Inter-Process Communication) כי הכול רץ בתוך אותו תהליך.

למה GRAALVM בעיקר מתאים ?

במערכות ענן מודרניות, זמן תגובה מהיר, צריכת זיכרון נמוכה ואבטחה הדוקה הם תנאים בסיסיים להפעלה יעילה.
GraalVM נותן מענה מדויק לדרישות אלו:

- קובץ הרצה עצמאי (Native Image) שאינו תלוי ב־JVM
- זמן עלייה של השירות תוך מילישניות בלבד
- צריכת זיכרון מינימלית

דוגמאות

MICROSERVISES ו GRAALVM

מערכת המבוססת מיקרו סרוויסים היא מערכת מחולקת להרבה שירותי קטנים (לוגיקה , משתמשים, תשלומים וכו)
כל שירות יכול לרוץ ולהתעדכן בנפרד.
בנוסף , כל שירות צריך לעלות מהר ולהגיב מיידית

בזכות Graalvm :

- אפשר להריץ יותר שירותים במקביל (בזכות החיסכון בזיכרון)
- אפשר לשכפל שירותים במהירות (בגלל העלייה המיידית)
- קונספט של scaling מתבצע בקלות ובעלות נמוכה יותר

המשמעות: פחות קוד = פחות זיכרון = פחות RAM = יותר שירותים על אותו שרת

חסרונות:

בנייה ארוכה – יצירת קובץ הרצה (Native Image) לוקחת זמן, ולרוב נמשכת כמה דקות. כל שינוי בקוד מחייב בנייה מחדש – מה שמאט תהליכי פיתוח תכופים.

קוד דינמי פחות נתמך – אם יש חלקים בקוד שפועלים רק בזמן הריצה (כמו טעינת רכיבים לפי שם), צריך להצהיר עליהם מראש. אחרת הם לא ייכללו בקובץ – ועלולות להתרחש שגיאות.

דרישות מערכת נוספות – לדוגמה, ב־Windows יש צורך בהתקנת כלי פיתוח נוספים (כמו C++ Build Tools).

לסיכום

GraalVM משנה את הדרך שבה אנו מריצים קוד בסביבות ענן:
הוא מציע הרצה מהירה, יעילה ובטוחה – בדיוק מה שנדרש בעולמות של Microservices, Serverless ו־Kubernetes.

הרצה תוך מילישניות – ללא JVM
שימוש בזיכרון נמוך
קובץ הרצה עצמאי, נייד וקל לפריסה
הפחתת סיכוני אבטחה – רק הקוד הדרוש נכלל
מתאים במיוחד ל־Cloud Native ול־Auto Scaling

בשורה התחתונה:
אם המערכת שלך צריכה לעלות מהר, לפעול ביעילות ולתמוך בפריסה דינמית – GraalVM הוא פתרון ששווה לבחון ברצינות.