GCP

מדריך דיפלוי מלא לפרוייקט

Full-Stack

(spring + MySQL + React)

לאורך כל המדריך, אפשר "לחיות" בלי הטקסים באפור TL;DR

Google **C**loud **P**latform או בקיצור **GCP** ולימים **GC**

הם נותנים $300 לניצול תוך 3 חודשים - אך חייבים להכניס כרטיס אשראי
מדגישים המון שלא יהיה חיוב אוטומטי עד שתפעילו אותו (הודעה תמידית באתר למעלה)

טוב.. זה גוגל.. המון כלים, המון אפשרויות והמון רפרנסים.. בהחלט אוברקיל לצורך הספציפי שלנו!

כדי שלא ללכת לאיבוד, הלכתי על אסטרטגיית "שלב-שלב":
עבודה מול DB בענן.. עבודה מול BACKEND (וגם DB) בענן.. ובסוף גם FRONTEND בענן..

לא בטוח אם קיימת דרך "אלגנטית" יותר.. אבל זה עבד עם האסטרטגיה שבחרתי..
ועל הדרך לומדים דבר או שניים..

מקווה שתאהבו..

מכל ה-

ממשקים, כלים ושירותים

GC_Console *(ממשק ה- WEB הגרפי - האתר של GC)*
- דרישת קדם: חשבון גוגל
- מצריך כניסה עם חשבון גוגל + קישור חשבון זה לחשבון תשלום (כ"א)
- כולל גם Cloud Shell בענן - לפקודות שורה (טקסט) ~ CMD

GC_CLI *(ערכת פיתוח - SDK)*
- דרישת קדם: חשבון (פעיל עם חשבון תשלום) ב- GC_Console
- מצריך התקנת Python runtime
- כולל גם Cloud Shell במחשב שלנו - לפקודות שורה (טקסט) ~ CMD

הוראות התקנה בהמשך

Cloud_Code *(פלאגין/הרחבה ל- VS Code / IntelliJ)*
- דרישת קדם: GC_CLI (יתקין אם צריך)
<< אין תמיכה בשירות הדיפלוי שבחרנו (ראה למטה - App Engine)

הוראות התקנה בסוף (נספח)

שירותי מחשוב (Compute)
* **App Engine** - דיפלוי - פריסה והפעלה בענן (תמיכה במגוון שפות, ניהול וסקיילינג אוטומטיים..) -- *[חומרה / מיקום]*
* **Compute Engine** - מכונה וירטואלית (VM) עם מ"ה - עליה יישב ה-DB למשל -- *[חומרה / מיקום]*
* kubernetes - הנצחה (orchestration) על קונטיינרים (docker למשל)..
...

שירותי אחסון (Storage)
* **Cloud SQL** - מסד נתונים (DB) בענן (כולל תמיכה ב-MySQL) - יישען על Compue Engine (נראה בהמשך)..
* Cloud Storage - כונן (Drive) בענן (שמירת אובייקטים ב- buckets.. למשל bucket לקבצי תמונות, bucket לקבצי וידאו)..
...

מגוון רחב של שירותים נוספים ו- APIs לכל מטרה (ككككللל מטרה)
...

עבור כל פרוייקט שיוצרים, כל SERVICE או API ב-GC שהוא רוצה להשתמש בו, חייבים לתת לפרוייקט הרשאה מפורשת לכך.. הדבר מצריך שלכל שירות כזה יהיה חשבון עם credentials (שם משתמש וסיסמה) שיגדיר הרשאות מסויימות לשירות..
למזלנו, לכל שירות, נוצר אוטומטית חשבון דיפולטי (עם הרשאה מלאה) שנקרא Default Service Account (DSA).
בהמשך @@ נשמור במחשב שלנו את ה-credentials לחשבון זה כ- Application Default Credentials (ADC), וכל אפליקציה (פרוייקט) שנרשה לה, תשתמש בהם.

למה?
א. כל שירות כזה אמור לעלות כסף (אחרי ה-$300 מתנה) וכל דבר מתומחר, וזה מחובר לכרטיס אשראי.
ב. בדרך זו מתאפשרת האופציה לחשבונות נוספים עם הרשאות מוגבלות לאנשים נוספים איתם משתפים את ניהול הפרוייקט למשל לאחד תהיה הרשאה רק ל- Cloud SQL, ולשני רק ל- App Engine..

באופן כללי ##, בהרצת פקודה מסויימת, אם נשכח משהו, תהיה הודעת "שגיאה" כזאת שתנחה מה לעשות: להפעיל שירות עזר / לאשר לפרוייקט שימוש בחשבון הדיפולטי..

ב-GC_Console (האתר של GC), קיים בר עליון קבוע (כחול) ובו:

* **תיבת חיפוש** - לחיפוש כל דבר לרבות השירות אליו נרצה לנווט
* **תיבת בחירה** - לבחירת "פרוייקט פעיל" אליו יתייחסו הפעולות/הפקודות שלנו
(בהמשך נראה איך עושים זאת במחשב שלנו)

מ ת ח י ל י ם

ב- GC_Console (נכון מבלבל אבל לא אזכיר יותר שזה האתר),
דרך **תיבת הבחירה** >> **new project**
נותנים לו שם (למשל my project back) - להשאיר NO ORGANIZATION >> לעשות **create**
לשמור בצד ***PROJECT_ID*** שייגזר משם הפרוייקט שבחרנו (ניתן לשנות בזמן יצירה כל עוד "ייחודי בעולם")
(את שם הפרוייקט נוכל לשנות בהמשך, ה-ID לא) **$$**

נוודא שהפרוייקט שלנו מסומן ב- **תיבת הבחירה**
ונחפש ב- **תיבת החיפוש** את APP ENGINE
לעשות **create** ולבחור REGION (בחרתי EUROPE-WEST6).
>> הפעלנו את שירות הדיפלוי לפרוייקט זה

נוודא שהפרוייקט שלנו מסומן ב- **תיבת הבחירה**
ונחפש ב- **תיבת החיפוש** את SQL
לעשות **create instance** ולבחור MySQL
יבקש שנפעיל Compute Engine (ה-DB ישב על VM) - ראה הערה **##** מקודם
נבחר **enable**

המשך קונפיגורציות ל- instance בעמ' הבא . .

. . המשך קונפיגורציות ל- instance מעמ' קודם

להרחיב תצוגת אפשרויות (חץ למטה - V) :

instance name (למשל mysql-instance)

root pass

MySQL version 8

development

לבחור REGION (בחרתי EUROPE-WEST6)
(ME-WEST1_ISRAEL - קיים כאן אבל לא ב-APP ENGINE, אז רציתי שיהיו באותו מקום)

SINGLE ZONE
Machine Type: Lightweight - 1 vCPU, 3.75GB
Storage: SSD, 10Gb
UNCHECKED "Enable automatic storage increase"

הלכתי על הקונפיגורציות "כמעט" הכי נמוכות (איטיות) ואני כמעט בטוח שהן טובות בהרבה מאלה שהיו בהירוקו (לא שזה היה מפורט שם)..

ללחוץ ADD NETWORK

ליצירת רשת עם המחשב שלנו שנוכל לגשת ל- DB שבענן. למשל כשנריץ את ה-backend מהמחשב שלנו בזמן פיתוח (השלב הבא) או בכלל דרך ה- WORKBENCH (תכף)..

name (למשל mysql-instance-network)

ב- network נכניס את ה-ip שלנו שמתקבל בביקור באתר:

What's my IP

סביר להניח שאין לנו IP קבוע (מספק האינטרנט) ואם אחרי תקופה זה משתנה - אז מעדכנים ב- NETWORK.

להכין קפה ☕ בזמן שזה יוצר את ה- instance

אחרי היצירה, אנחנו בתפריט של SQL - אז בתפריט הצידי לבחור DATABASES

add database - ולבחור רק שם (שם ה- "schema" כפי שאנחנו רגילים)

למרות שנוכל להשתמש באותו instance ל- databases (סכמות/פרוייקטים) נוספים,
לא נהוג לעשות זאת - הטרנד הוא "אינסטנס אחד לכל מיקרוסרביס"

לשמור בצד ***DATABASE*** , ***PUBLIC_IP_ADDRESS*** , ***CONNECTION_NAME*** , ***ROOT_PASS***

👍 כבר ניתן לבדוק את החיבור ב- WORKBENCH

להשתמש ב- ***PUBLIC_IP_ADDRESS*** ו- ***ROOT_PASS*** ששמרנו מקודם
(משתמש: root)

ה-WORKBENCH מרגיש כמעט מקומי.. בבדיקה שלי, העלות השבועית הייתה אפסית..

נרצה להתקין את ה- **GC SDK** (GC_CLI)
ה-Python runtime שצריך, יותקן אוטומטית אם לא יימצא (מלבד בדגמי MAC ללא מעבד אינטל).. אז..

אפליסטים - מהלינק הבא - לבחור התקנה מתאימה (או לקרוא על התקנת פייתון):
[לא צריך לקרוא הכל - המידע הרלוונטי מרוכז בהתחלה]
GC SDK INSTALLER - MAC

כל אלה שצוחקים 🙃 להוריד ולהתקין מהלינק הבא:
GC SDK INSTALLER - WINDOWS

כאמור - אם יופיע python3 כחלק מההתקנה - להשאיר מסומן
בסיום, בחלון ה-Cloud Shell שנפתח, יתן אפשרות configure the gcloud CLI - ללכת על זה
(אם בטעות דילגתם אז להריץ את הפקודה **gcloud init**)
מעלה דפדפן להזדהות מול GC.. בחירת הפרוייקט הפעיל (לפי ***PROJECT ID*** ששמרנו בצד מקודם) אליו יתייחסו הפקודות בהמשך.. (גם אופציה לבחירת REGION דיפולטי ל- Compute Engine..)
הנתונים שקבענו נשמרים עד שנשנה אחד מהם (למשל נשנה פרוייקט פעיל בהמשך) או עד שנפעיל את תהליך ה-init מחדש (סביר שלא).

.

ניתן להריץ פקודה **gcloud components list** שתציג קומפוננטות מותקנות..
נרצה להוסיף להתקנה עוד קומפוננטה של App Engine עם תמיכה ב- JAVA, אז נריץ את הפקודה:
gcloud components install app-engine-java
זה מתקין גם לפייתון בלית ברירה

נריץ את הפקודה:
gcloud auth application-default login
גם מעלה דפדפן להזדהות ואישור פעולה מול GC ושומר את פרטי הכניסה באיזשהו נתיב שהאפליקציות יוכלו לקחת משם כ-ADC. (ראה הערה**@@** מקודם)
אם משום מה לא תרצו שיהיה שמור במחשב אז מוחקים עם הפקודה:
gcloud auth application-default revoke
(יש אפשרות נוספת להגדיר environment variable GOOGLE_APPLICATION_CREDENTIALS
שיצביע לקובץ עם פרטי הכניסה)

זה הזמן לוודא שהפרוייקט שלנו עובד (מקומית) נכון..
ללא קשר למה שעושים כאן - תעשו פוש ל- GitHub.

בתוך <project> (לא בדפנדנסיז!)

מנהל מספרי גרסאות של spring-cloud-gcp

```
<!-- Add Spring Cloud GCP Dependency BOM -->
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>com.google.cloud</groupId>
        <artifactId>spring-cloud-gcp-dependencies</artifactId>
        <version>2.0.3</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
<!-- ******************* -->
```

בדפנדנסיז (מחליפים MySQL "הרגיל" בזה של GCP)

```
<!-- Add CloudSQL Starter for MySQL -->
  <dependency>
    <groupId>com.google.cloud</groupId>
    <artifactId>spring-cloud-gcp-starter-sql-mysql</artifactId>
  </dependency>

  <!-- MySQL -->
  <!--<dependency>-->
  <!--    <groupId>mysql</groupId>-->
  <!--    <artifactId>mysql-connector-java</artifactId>-->
  <!--    <scope>runtime</scope>-->
  <!--</dependency>-->
<!-- ******************* -->
```

בתוך <build> <plugins>

שירות הדיפלוי הנבחר App Engine

```
<!--GCP app-engine maven plugin-->
  <plugin>
    <groupId>com.google.cloud.tools</groupId>
    <artifactId>appengine-maven-plugin</artifactId>
    <version>2.4.4</version>
  </plugin>
<!-- ******************* -->
```

application.properties

לא נגעתי

```
spring.jpa.generate-ddl=true
spring.jpa.properties.hibernate.enable_lazy_load_no_trans=true
```

הוחלף

```
#spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5InnoDBDialect
```

הערת **CLR** למטה

```
#spring.jpa.hibernate.ddl-auto=create-drop
#spring.jpa.hibernate.ddl-auto=create
spring.jpa.hibernate.ddl-auto=update
```

בוטל

```
#spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```

MySQL מקומי - בוטל - הוחלף בזה של GCP למטה

```
#spring.datasource.url=jdbc:mysql://localhost:3306/DATABASE?serverTimezone=UTC&createDatabaseIfNotExist=true
#spring.datasource.username=root
#spring.datasource.password=1234
```

השארתי את ה- URL למרות ש- Cloud SQL starter מגדיר אותו אוטומטית (עובד בלי שורה זו)

```
spring.datasource.url=jdbc:mysql://PUBLIC_IP_ADDRESS:3306/DATABASE?serverTimezone=UTC&createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=ROOT_PASS
spring.cloud.gcp.sql.database-name=DATABASE
spring.cloud.gcp.sql.instance-connection-name=CONNECTION_NAME
spring.profiles.active=mysql
```

עוד של GCP שלא השתמשתי

```
#database=mysql
#spring.datasource.initialization-mode=always
#spring.datasource.hikari.maximum-pool-size=10
```

כבר ניתן להריץ את פרוייקט ה-backend (ב-intellij) והוא יעבוד מול ה- DB בענן

בהרצה ראשונה, תתקבל "שגיאה" (ראה הערה **##** מקודם)
שצריך להפעיל שירות "Cloud SQL Admin API" עם לינק להפעלתו - לוחצים ומפעילים ומריצים שוב.

לגבי **CLR** לאתחול DB:
במקום אסטרטגיית create-drop (שיוצרת בעליית השרת ומוחקת בנפילתו), ובהנחה שהפלנו את השרת וכרגע ה-DB ריק, נרים את השרת פעם אחת עם create בלבד ואז נעביר את ה- CLR ל-off ונעבור סופית ל- update

שוב, מרגיש כמעט מקומי..

בדיפלוי,Aapp Engine משתמש בערכי דיפולט, שכוללים runtime java 17. נרצה להנחות אותו לשנות את זה לגרסה 11

עם/בלי קשר ל-$300 מתנה, בפרוייקטים שלנו לא דאגנו להתעלמות מבקשות חדשות בזמן המתנה לתשובת פרומיס..
גם לא לסימן המתנה/טעינה.. מה גם שזה נורא מתסכל ללחוץ כפתור ולחכות 10-20שניות לפעולה אטומית שמביאה "חתול"

הדיפולט קובע רמת מעבד/זיכרון הכי בסיסית שנקראת F1
F2 משפר את התוצאה הסופית באופן ניכר!
[לא הספקתי לראות עלויות לעומת F1]
אם מכל סיבה תחליטו להיות בדיפולט F1 פשוט תמחקו (או תתעדו) את השורה השניה

ניצור קובץ src/main/**appengine/app.yaml** שיכיל:

```
runtime: java11
instance_class: F2
```

לצורך הבילד נתקין את APACHE MAVEN .. מורידים מהלניק הבא:
apache-maven-3.8.6-bin.zip
מחלצים ל- c:\ program files
מוסיפים את הנתיב לתיקיה C:\Program Files\apache-maven-3.8.6\bin ל- Path..
בדומה למה שעשינו ב-JAVA בזמנו (חיפוש SYSTEM ENVIRONMENT ב- WINDOWS)..
בסיום בודקים ע"י **mvn -v** ב- cmd - אמור להציג נתיב ל-MAVEN ו- JAVA..

CMD מתוך התיקיה בה נמצא ה- POM.XML - מריצים את הפקודה:
gcloud app deploy

היא עושה כמה פקודות ברקע ומשתמשת ב- maven וב- app engine
הפקודה היחידה שהרגשתי שהיא נותנת עבודה בשבילי

בוחרים REGION (בחרתי EUROPE-WEST6).
כבר לפני האישור הסופי (וגם בסיום התהליך) - מציג את ה- **URL** של האפליקציה בענן אחרי ההעלאה:

https://***PROJECT_ID*** .oa.r.appspot.com

ה- .oa.r
יכול להיות שונה אצלכם

ה-**URL** יחליף בפרונט את ה- http://localhost:8080 שבתוך Globals (בינתיים ב- Development)

ה- frontend כבר יעבוד מול ה- backend (וה-DB) בענן

עשינו שינוי? ← פקודת הדיפלוי שוב..
ואז ב-GC_Console (נו, אתר GC) ב- App Engine ב- Versions ניתן למחוק את הדיפלוי הישן.

על מנת לוודא/לבחור את הפרוייקט כפרוייקט הפעיל,
נריץ (ב-CMD) את הפקודה/ות:
gcloud projects list
gcloud config get-value project
gcloud config set project *PROJECT_ID*

FRONTEND

ב-GC_CONSOLE ניצור פרוייקט חדש (למשל my project front)
ונפעיל עבורו את App Engine (ראה $$ מקודם)

ב-Globals נעדכן את ה-urls ב- Production (כמו שעשינו בשלב הקודם ב- Development)

* ההרחבה ל- VS Code שויתרנו עליה, היתה מאפשרת דיפלוי לשירות אחר Cloud Run
* אם לנצל את הפקודה gcloud app deploy שתעשה גם בילד, אז צריך לשנות ב- package.json
(ובחזרה אם נרצה להמשיך לפתח ולהשתמש ב-npm start)
אז הלכתי על פתרון נוח יותר:

root directory > **app.yaml**

```
runtime: nodejs16
instance_class: F2
handlers:
  # Serve all static files with url ending with a file extension
  - url: /(.*\..+)$
    static_files: build/\1
    upload: build/(.*\..+)$
  # Catch all handler to index.html
  - url: /.*
    static_files: build/index.html
    upload: build/index.html
```

לגבי F2 - אותו הסבר כמו ב- BACKEND (עמ' קודם)

בנוסף להגדרת סביבת ה-runtime, מנחים את APP ENGINE לקחת את תיקיית הבילד לדיפלוי

קובי: "גאון אני לא"

נתקלתם בקונפליקט של YUP (אם השתמשתם כמובן)? הנה הפיתרון:

root directory > **.npmrc**

```
force=true
legacy-peer-deps=true
```

תודה רבה לשקד ולאידיל

בטרמינל של VS Code :

נבחר את פרוייקט הפרונט כפרוייקט הפעיל

```
gcloud projects list
gcloud config get-value project
gcloud config set project PROJECT_ID
```

npm run build

gcloud app deploy

בהרצה ראשונה, יכולה לתתקבל "שגיאה" (ראה הערה **##** מקודם)
שצריך להפעיל שירות "Cloud Build" עם לינק להפעלתו - לוחצים ומפעילים* ומריצים שוב.
*אם כבר מופעל (והתקבלה ה-"שגיאה") << לעשות DISABLE ואז ENABLE (יש תקלה כזאת)

כבר לפני האישור הסופי (וגם בסיום התהליך) - מציג את ה- **URL** של האפליקציה בענן אחרי ההעלאה:

https://*PROJECT_ID* .oa.r.appspot.com

ה- .oa.r.
יכול להיות שונה אצלכם

CORS ORIGIN

בזמן פיתוח, ב-BACKEND, אישרנו גישה ל-ORIGIN שלו מה-ORIGIN הזר של ה-FRONTEND שאז זה היה פורט 3000 המקומי של ריאקט.. אז צריך לעדכן שהפרונט כרגע נמצא בענן..

ב-BACKEND ב- filters.CORSFilter :
מחליפים את http://localhost:3000
ב- **URL** של הפרונט שהתקבל בשלב הקודם

נ.ב.
ללא קשר, דפדפן במצב INCOGNITO
יכול לגרום לשגיאות CORS

CMD מתוך התיקיה בה נמצא ה- POM.XML :

לקבוע את ה-BACKEND כפרוייקט הפעיל ורק אז:
gcloud app deploy

הכל בענן - נכנסים ל-URL של ה-frontend כמובן

משאיר לכם את ההחלטה לגבי כדאיות שנמוך הקונפיגורציות (במיוחד אם תרצו להשאיר את זה אחרי תקופת הניסיון)
בכל מקרה ב-BILLING ניתן לקבוע BUDGETS AND ALERTS להגבלת סכום והתראות (פר פרוייקט).
יש לעקוב אחרי עלויות ולא לשכוח שכנראה תעלו יותר מפרוייקט אחד!

עידוד? אם תרצו להעלות עוד פרוייקטים (חסר לכם שלא אחרי דבר כזה), תצטרכו לבצע רק כמה שורות מכל זה..

נספח בעמ' הבא . .

נספח

Cloud_Code *(פלאגין/הרחבה ל- VS Code / IntelliJ)*
- דרישת קדם: GC_CLI (יתקין אם צריך)
<< אין תמיכה בשירות הדיפלוי שבחרנו (App Engine)

Cloud Code (IntelliJ)
no support for APP ENGINE when JAVA11 & IntelliJ community version
Windows: File > Settings > Plugins
Mac OS: IntelliJ IDEA > Preferences > Plugins
search for "Cloud Code" & install it..

Cloud Code (VS Code)

File > Preferences > Extentions
search for "Cloud Code" & install it..

בתפריט הסמלים השמאלי יופיע אייקון חדש, וההרחבה שלו (התפריט השמאלי הרחב) עמוסה
בתפריט הרחב למעלה - סמל שלושת הנקודות (...)
> להוריד סימנים מלבד Cloud APIs ו-Help and Feedback
ב- **Help and Feedback** רואים את שם ה- logged-in GCP user שנשמר משלב CLI init מקודם,
בסטטוס בר של VSCode (הבר הכחול שנמצא בתחתית החלון) מופיע שם "הפרוייקט הפעיל" (של GC),
ובעזרת סמל שני החצים ההפוכים, נוכל לשנות את הבחירה - נשנה לפרוייקט הפרונט עליו עובדים עכשיו.
ב- **Cloud APIs** נבחר ב- APP ENGINE
נראה מספר הודעות (ראה הערה **##** מקודם)
* מבקש הפעלת App Engine Admin API - נלחץ ונאשר
* התקנת Client Library (של APP ENGINE)
מעתיקים פקודת npm מלשונית NodeJS לטרמינל של VSCode
או כבר לוחצים run in terminal במקום copy to clipboard

ב ה צ ל ח ה