

הוראות התקנה - מעבדה 2:

שלב 1 - יצירת דיאגרמה ארכיטקטונית ותחשיב עלויות
שלב 2 - בדיקת המערכת הקיימת ע"י כניסה לאתר ה-Monolithic application
לפי כתובת ה-IP של ה-EC2
בדיקת תקינות שניתן להוסיף/לערוך/למחות רשומות של suppliers

שלב 2.3 - ניתוח אופן הרצת האפליקציה המונוליטית

1. התחברות לשרת

השתמש ב-EC2 Instance Connect כדי להתחבר לשרת בשם **MonolithicAppServer**.

2. בדיקת האפליקציה

הרץ את הפקודה הבאה כדי לבדוק איזו תוכנה מאזינה על פורט 80:

```
sudo lsof -i :80
```

שים לב אילו תהליכים מאזינים על פורט 80 ובאיזו פרוטוקול ופורט רץ תהליך ה-node.

3. בדיקת תהליך node

הרץ את הפקודה הבאה כדי לבדוק את התהליכים שקשורים ל-node:

```
ps -ef | head -1; ps -ef | grep node
```

בדוק איזה משתמש מריץ את תהליך ה-node והאם מזהה התהליך (PID) תואם למה שראית בפקודה הקודמת.

4. ניתוח קוד האפליקציה

עבור לתיקיית קוד האפליקציה:

```
cd ~/resources/codebase_partner
```

```
ls
```

בתיקייה זו קיים הקובץ **index.js** שמכיל את הלוגיקה הבסיסית של האפליקציה.

חיבור למסד הנתונים RDS

1. מציאת כתובת ה-endpoint

מצא את כתובת ה-endpoint של מסד הנתונים מסוג RDS בסביבת הלימוד.

2. בדיקת גישה למסד הנתונים

השתמש בפקודה הבאה כדי לבדוק אם הפורט הסטנדרטי של MySQL פתוח (3306):

```
nmap -Pn <endpoint>
```

3. התחברות ל-MYSQL

התחבר למסד הנתונים באמצעות לקוח MySQL שמוקדן כבר על השרת:

```
mysql -u admin -p
```

כאשר תתבקש, הזן את הסיסמה:

```
lab-password
```

4. בדיקת נתונים במסד הנתונים

מהרשומה `mysql>`, הרץ את הפקודות הבאות כדי לבדוק את התוכן:

```
SHOW DATABASES;
```

```
USE COFFEE;
```

```
SHOW TABLES;
```

```
SELECT * FROM suppliers;
```

5. יציאה

צא מה-MYSQL:

```
EXIT;
```

סגור את חלונית EC2 Instance Connect ואת חלונית האפליקציה "coffee suppliers".

שלב 3.1 יצירת סביבת עבודה
צור סביבת עבודה ב-Cloud9 בשם `MicroservicesIDE`, השתמש בשרת EC2 חדש מסוג `t3.small` ולהריץ את מערכת ההפעלה `Amazon Linux 2`. השרת צריך לתמוך בחיבורים דרך SSH ולהיות מופעל בתוך רשת ה-`LabVPC` בתת-הרשת הציבורית מספר 1 (`Public Subnet1`).

שלב 3.2 העתקת קוד האפליקציה ל-IDE שלך

1. הורדת קובץ ההתחברות (`labsuser.pem`)

מהפאנל `AWS Details` בהוראות המעבדה, הורד את הקובץ `labsuser.pem` למחשב המקומי שלך.

2. העלאת הקובץ ל-Cloud9

העלה את הקובץ שהורדת אל סביבת `AWS Cloud9` שלך.

3. שינוי הרשאות על הקובץ

השתמש בפקודה הבאה כדי להגדיר הרשאות מתאימות לקובץ:

```
chmod 400 ~/environment/labsuser.pem
```

4. יצירת תיקיית זמנית

צור תיקייה זמנית בנתיב הבא:

```
mkdir -p /home/ec2-user/environment/temp
```

5. שליפת כתובת ה-IPv4 הפרטית של השרת

היכנס לקונסול של `Amazon EC2` ומצא את כתובת ה-`Private IPv4` של השרת `MonolithicAppServer`.

6. העתקת קוד המקור באמצעות `scp`

השתמש בפקודה הבאה (התאם את כתובת ה-IP):

```
scp -r -i ~/environment/labsuser.pem  
ubuntu@<כתובת-IP>:/home/ubuntu/resources/codebase_partner/*  
~/environment/temp/
```

7. אימות ההעתקה

בדוק בדפדפן הקבצים של `Cloud9` שהקבצים אכן הועתקו לתיקייה `temp`.

שלב 3.3 - יצירת תקיית עבור Microservices

רשום את הפקודות הבאות ליצירת תקיית

```
mkdir microservices
```

```
cd microservices
```

```
mkdir customer
```

```
mkdir employee
```

והפקודות הבאות להעתקת קבצי המקור לתקיות ה-Microservices ומחיקת התקיייה הזמנית

```
cp -r ~/environment/temp/* ~/environment/microservices/customer/
```

```
mv -f ~/environment/temp/* ~/environment/microservices/employee/
```

```
rmdir ~/environment/temp/
```

שלב 3.4 יצירת Repository ב-CodeCommit בשם microservices והעלאת הקבצים

```
cd ~/environment/microservices
```

```
git init
```

```
git branch -m dev
```

```
git add .
```

```
git commit -m 'two unmodified copies of the application code'
```

```
git remote add origin
```

```
https://git-codecommit.us-east-1.amazonaws.com/v1/repos/microservices
```

```
git push -u origin dev
```

בדוק בCodeCommit שנוצר Repository חדש בשם microservices

שלב 4.1 התאמת הגדרות קבוצת האבטחה של ה-EC2 של Cloud9

בשלב זה תשתמש ב-AWS Cloud9 כסביבת בדיקות.

תפעיל קונטיינרים של Docker כדי לבדוק microservices.
כדי שתוכל לגשת אליהם דרך הדפדפן, עליך לפתוח פורטים מתאימים בגדרות security groups של Cloud9.

- פתח תעבורת TCP נכנסת על הפורטים:

- 8080

- 8081

הפעולה תתבצע דרך הגדרות קבוצת האבטחה של מופע ה-EC2 של Cloud9.

שלב 4.2 שינוי קוד המקור של שירות הלקוח (Customer Microservice)

בקוד השירות של הלקוח יש עדיין פונקציונליות של ניהול ספקים (add/edit/delete) שלא מתאימה ללקוחות.

הסר את הקוד כדי להשאיר רק פעולות קריאה (read-only) כמו בדוגמא.

שינוי בקובץ `supplier.controller.js`

נתיב: `customer/app/controller/supplier.controller.js`

```
const Supplier = require("../models/supplier.model.js");
const {body, validationResult} = require("express-validator");

exports.findAll = (req, res) => {
  Supplier.getAll((err, data) => {
    if (err)
      res.render("500", {message: "The was a problem retrieving the list of suppliers"});
    else res.render("supplier-list-all", {suppliers: data});
  });
};

exports.findOne = (req, res) => {
  Supplier.findById(req.params.id, (err, data) => {
    if (err) {
      if (err.kind === "not_found") {
        res.status(404).send({
          message: `Not found Supplier with id ${req.params.id}.`
        });
      } else {
        res.render("500", {message: `Error retrieving Supplier with id ${req.params.id}`});
      }
    } else res.render("supplier-update", {supplier: data});
  });
};
```

שינוי בקובץ `supplier.model.js`

נתיב: `customer/app/models/supplier.model.js`

- השאר רק את הפונקציות:

`Supplier.getAll` ○

`Supplier.findById` ○

- אל תשכח להשאיר את השורה האחרונה:

```
module.exports = Supplier;
```

שינוי בתפריט הניווט

נתיב: `customer/views/nav.html`

- שורה 3: שנה מ-`Monolithic Coffee suppliers` ל-`Coffee suppliers`

- שורה 7: שנה מ-`Home` ל-`Customer home`

- לאחר שורה 8, הוסף את השורה הבאה:

```
<a class="nav-link" href="/admin/suppliers">Administrator link</a>
```

ניקוי ממשק המשתמש

נתיב: `customer/views/supplier-list-all.html`

- מחק את שורה 32 (כפתור "Add a new supplier")
- מחק את שורות 26 ו-27 (כפתורי עריכה/עדכון של ספקים)

מחיקת קבצים מיותרים

מחק את הקבצים הבאים מתיקיית `customer/views`:

- `supplier-add.html`

- `supplier-form-fields.html`

- `supplier-update.html`

שינוי בקובץ `index.js`

נתיב: customer/index.js

- שורות 27 עד 37 – הפוך אותן להערות ע"י הוספת // בתחילת כל שורה
- שורה 45 – שנה את מספר הפורט ל-8080

```
app.listen(8080, () => {  
  console.log("Server is running on port 8080.");  
});
```

שלב 4.3 יצירת Docker עבור customer microservice

בנתיב **customer/**, צרו קובץ חדש בשם **Dockerfile** עם התוכן הבא:

```
FROM node:11-alpine  
RUN mkdir -p /usr/src/app  
WORKDIR /usr/src/app  
COPY . .  
RUN npm install  
EXPOSE 8080  
CMD ["npm", "run", "start"]
```

בתוך הטרימינל של AWS Cloud9, עברו לתיקיית **customer**:

```
cd ~/environment/microservices/customer
```

```
docker build --tag customer .
```

לאחר מכן, בדקו שה-Image נוצר:

```
docker images
```

```
voqlabs:~/environment/microservices/employee (dev) $ docker images  
REPOSITORY TAG IMAGE ID CREATED SIZE  
employee latest 614abdd0b631 13 seconds ago 82.7MB  
customer latest 1a55d56ea8c7 43 minutes ago 82.7MB  
voqlabs:~/environment/microservices/employee (dev) $ dbEndpoint=$(cat ~/environment/microservices/employee/app/config/config.js | grep 'APP_DB_HOST' | cut -d '"' -f2)  
voqlabs:~/environment/microservices/employee (dev) $ echo $dbEndpoint  
supplierdb.cryyakaqr1.us-east-1.rds.amazonaws.com  
voqlabs:~/environment/microservices/employee (dev) $ docker run -d --name employee_1 -p 8081:8081 -e APP_DB_HOST="$dbEndpoint" employee  
e59aa479ea56  
voqlabs:~/environment/microservices/employee (dev) $ docker ps  
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  
e59aa479ea56 employee "docker-entrypoint.s..." 3 minutes ago Up 3 minutes 8080/tcp, 0.0.0.0:8081->8081/tcp, :::8081->8081/tcp employee_1  
1f620b4a3e16 customer "docker-entrypoint.s..." 38 minutes ago Up 38 minutes 0.0.0.0:8080->8080/tcp, :::8080->8080/tcp customer_1  
voqlabs:~/environment/microservices/employee (dev) $
```

נגדיר משתנה סביבה עם כתובת ה-DB:

```
dbEndpoint=$(cat
~/environment/microservices/customer/app/config/config.js | grep
'APP_DB_HOST' | cut -d ' ' -f2)
echo $dbEndpoint
```

ונריץ את ה-Container:

```
docker run -d --name customer_1 -p 8080:8080 -e APP_DB_HOST="$dbEndpoint"
customer
```

נבדוק אילו קונטיינרים פעילים:

```
docker ps
```

ונבדוק שה-microservice של customer עובד:

ניגש את כתובת ה-IP של Cloud9

```
http://<cloud-9-public-IPv4-address>:8080
```

נוודא ש:

- ניתן להציג את רשימת הספקים (List of suppliers).
- אין כפתורי "Add a new supplier" או כפתורי עריכה.
- הקישור "Administrator link" עדיין לא פעיל (עד שייבנה employee microservice).

נשמור את השינויים ב-CodeCommit
בצעו commit ו-push של השינויים:

```
git add .
git commit -m "Refactor customer microservice for Docker deployment"
git push
```

שלב 4.4 – שינוי קוד המקור של employee microservice:

שינויים בקוד:

1. `employee/app/controller/supplier.controller.js`

מצאו את כל השורות עם `redirect` והוסיפו `admin/` לנתיב.

להרצת פקודת חיפוש מהירה:

```
cd ~/environment/microservices/employee
```

```
grep -n 'redirect' app/controller/supplier.controller.js
```

2. `employee/index.js`

הוסיפו `admin/` לכל `app.get` ו-`app.post`.

שנו את פורט ההאזנה (ב-שורה 45) ל-8081 כדי למנוע התנגשויות עם השירות של הלקוחות.

מצאו את השורות עם:

```
grep -n 'app.get\|app.post' index.js
```

שימו לב:

ב-שורה 22 הנתיב צריך להסתיים ב `admin/`

3. `employee/views/supplier-add.html` ו-`supplier-update.html`

הוסיפו `admin/` בתחילת הכתובות בתוך `action=` של הטפסים.

```
*/grep -n 'action' views
```

4. `employee/views/supplier-list-all.html` ו-`home.html`

עדכנו את הקישורים (`href`) – הוסיפו `admin/` לכתובות.

חיפוש קישורים:

```
grep -n 'href' views/supplier-list-all.html views/home.html
```

5. `employee/views/header.html`

שנו את כותרת הדף ל-`Manage coffee suppliers`.

6. `employee/views/nav.html`

שורה 3: שנו מ-`"Monolithic Coffee suppliers"` ל-`"Manage coffee suppliers"`.

שורה 7: החליפו את הקישור ל:

```
<a class="nav-link" href="/admin/suppliers">Administrator  
home</a>
```

שורה חדשה (אחרי שורה 8): הוסיפו קישור לעמוד הלקוחות:

```
<a class="nav-link" href="/">Customer home</a>
```

לאחר כל העריכות – ודאו ששמרתם את הקבצים.

שלב 4.5 - צור Dockerfile והרץ Container לבדיקת microservice
העתק את ה-Dockerfile משירות הלקוחות (customer) לתיקיית העובדים (/employee).

ערוך את שורת EXPOSE בקובץ החדש ושנה את הפורט ל-8081.
בנה את הimage:

```
cd ~/environment/microservices/employee  
docker build -t employee .
```

הרץ את הקונטיינר:

```
docker run -d --name employee_1 -p 8081:8081 -e  
DBHOST=<your-db-endpoint> employee
```

פתחו דפדפן עם כתובת ואמתו ש-Employee microservice עובד:

<http://<Cloud9-IP>:8081/admin/suppliers>

כולל רשימת ספקים, כפתורים להוספה ועריכה!

בדקו את הפונקציונליות

- הוספת ספק חדש – ודא שהוא מופיע לאחר ההוספה.
- עריכת ספק קיים – ודא שהשינויים נשמרים.
- מחיקת ספק – ערוך ספק ובחר באופציית "Delete this supplier", ואמת שהוא הוסר מהרשימה.

שלב 4.6 - שינוי הפורט ל-8080 והכנת image לפריסה ב-ECS

ערוך את הקבצים הבאים:

- `employee/index.js`: שנה את הפורט מ-8081 ל-8080.
- `employee/Dockerfile`: שנה את שורת EXPOSE ל-8080.

2. מחק את הקונטיינר הישן:

```
docker rm -f employee_1
```

3. בנה מחדש את התמונה:

```
docker build -t employee .
```

אין צורך להריץ קונטיינר חדש. זה הכנה לפריסה ב-ECS.

שלב 4.7 - שליחת השינויים ל-CodeCommit

סקור שינויים

- לחץ על סמל ה-Source Control ב-AWS Cloud9.
- עיין ברשימת הקבצים ששוננו (כמו `index.js`) והשווה לגרסאות קודמות.

בצע `commit` ו-`push` ל-CodeCommit

באמצעות ממשק Git של Cloud9 או בטרמינל:

```
git add .
git commit -m "Updated employee microservice to support /admin
path and changed port to 8080"
git push
```

שלב 5: יצירת מאגרי ECR, ECS Cluster, הגדרות משימה וקבצי AppSpec

שלב 5.1: יצירת מאגרי ECR והעלאת תמונות Docker

התחברות ל-ECR דרך Docker:

```
account_id=$(aws sts get-caller-identity | grep Account | cut  
-d ' ' -f4)  
aws ecr get-login-password --region us-east-1 | docker login  
--username AWS --password-stdin  
$account_id.dkr.ecr.us-east-1.amazonaws.com
```

יצירת מאגרי ECR נפרדים:

- customer ○
- employee ○

עדכון הרשאות לכל Cluster:

```
{  
  "Version": "2008-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Principal": "*",  
      "Action": "ecr:*"  
    }  
  ]  
}
```

תיוג Images

```
docker tag customer:latest  
$account_id.dkr.ecr.us-east-1.amazonaws.com/customer:latest  
docker tag employee:latest  
$account_id.dkr.ecr.us-east-1.amazonaws.com/employee:latest
```

דחיפת התמונות ל-ECR:

```
docker push  
$account_id.dkr.ecr.us-east-1.amazonaws.com/customer:latest
```

```
docker push
```

```
$account_id.dkr.ecr.us-east-1.amazonaws.com/employee:latest
```

שלב 5.2 - יצירת ECS

- צור ECS cluster בשם `microservices-serverlesscluster` מסוג `Fargate`.
- הגדר אותו לשימוש ב-`LabVPC`, תת-רשת `PublicSubnet1` ו-`PublicSubnet2` בלבד.
- עקוב אחר תהליך ההקמה ב-`CloudFormation` עד שהסטטוס הוא `CREATE_COMPLETE`.

שלב 5.3 - יצירת מאגר CodeCommit

- צור מאגר בשם `deployment` שישמש לאחסון קבצי הגדרה.
- ב-`Cloud9`, צור תיקייה בשם `deployment` והפעל בה `Git` עם ענף `dev`.

שלב 5.4 - הגדרות משימה (task definitions)

:customer

צור קובץ `taskdef-customer.json` עם התוכן הבא:

```
{
  "containerDefinitions": [
    {
      "name": "customer",
      "image": "<IMAGE1_NAME>",
      "environment": [
        {
          "name": "APP_DB_HOST",
          "value": "<RDS-ENDPOINT>"
        }
      ],
      ...
    }
  ]
}
```

```

    }
  ],
  "requiresCompatibilities": ["FARGATE"],
  ...
  "executionRoleArn":
"arn:aws:iam::<ACCOUNT-ID>:role/PipelineRole",
  "family": "customer-microservice"
}

```

- החלף **<ACCOUNT-ID>** במספר חשבון AWS שלך.

- החלף **<RDS-ENDPOINT>** בכתובת של RDS שלך.

- הערה: לאחר ההרצה הראשונה, מחליפים את **"image": "customer"** ל-**"<IMAGE1_NAME>"**.

:employee

- שכפל את הקובץ וערוך את שלוש הפעמים שבהן מופיע המילה **customer** ל-**employee**.

- הרשם כל אחד באמצעות הפקודה:

```

aws ecs register-task-definition --cli-input-json
file:///home/ec2-user/environment/deployment/taskdef-employee.
json

```

שלב 5.5 - קובצי AppSpec

:customer

צור קובץ **appspect-customer.yaml** עם התוכן:

```

version: 0.0
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: <TASK_DEFINITION>
        LoadBalancerInfo:

```

```
ContainerName: "customer"  
ContainerPort: 8080
```

:employee

- שכפל ושנה את `ContainerName` ל-`employee`.

שלב 5.6 - עדכון הקבצים ב-CodeCommit

ערוך את שורת ה-`image` בכל קובץ JSON ל-"`<IMAGE1_NAME>`".

שמור את כל 4 הקבצים:

- `taskdef-customer.json`
- `taskdef-employee.json`
- `appspec-customer.yaml`
- `appspec-employee.yaml`

דחוף את הקבצים ל-CodeCommit:

```
git add .  
git commit -m "Added ECS task defs and AppSpec files"  
git push origin dev
```

שלב 6 - יצירת Target Groups ו-Application Load Balancer

שלב 6.1 - יצירת ארבע Target Groups

בגלל שהולכים להשתמש באסטרטגיית פריסה מסוג Blue/Green, נדרשות שתי קבוצות יעד לכל מיקרו-שירות. המשמעות:

- Blue: הסביבה הנוכחית שפועלת בפרודקשן
 - Green: הגרסה החדשה שעוברת בדיקות לפני החלפה
- הגדרות שיש להחיל עבור כל Target Group:

קבוצת יעד ראשונה למיקרו-שירות **customer**:

- שם: **customer-tg-one**
- סוג יעד: IP addresses
- פרוטוקול: HTTP
- פורט: 8080
- VPC: LabVPC
- נתיב לבדיקה (Health Check): /
- לא לרשום מטרות (targets) בשלב זה

קבוצת יעד שנייה ל-**customer**:

- זהה להגדרות הקודמות, רק עם שם: **customer-tg-two**

קבוצת יעד ראשונה ל-**employee**:

- שם: **employee-tg-one**
- נתיב לבדיקה (Health Check): **/admin/suppliers**
- שאר ההגדרות כמו בקבוצות הקודמות

קבוצת יעד שנייה ל-**employee**:

- כמו הקודמת, עם שם: **employee-tg-two**

שלב 6.2 - יצירת security group עבור microservices

Create security group [Info](#)

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To create a new security group, complete the fields below.

Basic details

Security group name [Info](#)

Name cannot be edited after creation.

Description [Info](#)

VPC [Info](#)

Inbound rules [Info](#)

This security group has no inbound rules.

[Add rule](#)

Outbound rules [Info](#)

Type Info	Protocol Info	Port range Info	Destination Info	Description - optional Info	
HTTP	TCP	80	Anywhere...		Delete
Custom TCP	TCP	8080	Anywhere...		Delete

[Add rule](#)

Rules with destination of 0.0.0.0/0 or ::/0 allow your instances to send traffic to any IPv4 or IPv6 address. We recommend setting security group rules to be more restrictive and to only allow traffic to specific known IP addresses.

יצירת Application Load Balancer

Listeners and routing [Info](#)

A listener is a process that checks for connection requests using the port and protocol you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets.

▼ Listener HTTP:80

[Remove](#)

Protocol: HTTP
Port: 80
1-65535

Default action [Info](#)

Forward to: customer-tg-two
Target type: IP, IPv4
HTTP

[Create target group](#)

Listener tags - optional

Consider adding tags to your listener. Tags enable you to categorize your AWS resources so you can more easily manage them.

[Add listener tag](#)

You can add up to 50 more tags.

▼ Listener HTTP:8080

[Remove](#)

Protocol: HTTP
Port: 8080
1-65535

Default action [Info](#)

Forward to: customer-tg-one
Target type: IP, IPv4
HTTP

[Create target group](#)

Listener tags - optional

Consider adding tags to your listener. Tags enable you to categorize your AWS resources so you can more easily manage them.

[Add listener tag](#)

You can add up to 50 more tags.

[Add listener](#)

שלב 7.1 יצירת ECS Customer microservice

צור קובץ חדש `create-customer-microservice-tg-two.json` בסביבת ה-Cloud9

בתוך תיקיית deployment והעתק את התוכן הבא:

```
{
  "taskDefinition": "customer-microservice:REVISION-NUMBER",
  "cluster": "microservices-serverlesscluster",
  "loadBalancers": [
    {
      "targetGroupArn": "MICROSERVICE-TG-TWO-ARN",
      "containerName": "customer",
      "containerPort": 8080
    }
  ],
  "desiredCount": 1,
  "launchType": "FARGATE",
  "schedulingStrategy": "REPLICA",
  "deploymentController": {
    "type": "CODE_DEPLOY"
  },
  "networkConfiguration": {
    "awsvpcConfiguration": {
      "subnets": [
        "PUBLIC-SUBNET-1-ID",
        "PUBLIC-SUBNET-2-ID"
      ],
      "securityGroups": [
        "SECURITY-GROUP-ID"
      ],
      "assignPublicIp": "ENABLED"
    }
  }
}
```

משימה 7.2: יצירת שירות Amazon ECS עבור employee microservice

שלבי ביצוע:

1. שכפול קובץ JSON

○ העתק את הקובץ ששימש ליצירת שירות הלקוח (למשל: `create-customer-microservice-tg-two.json`)

○ שמור אותו בשם: `create-employee-microservice-tg-two.json` באותה תיקייה.

2. עריכת הקובץ המשוכפל:

- שורה 2: שנה את "customer-microservice" ל-"employee-microservice" כמו כן, עדכן את מספר הגרסה (revision number).

- שורה 6: הדבק את ARN של קבוצת היעד employee-tg-two ! אל תבצע רק החלפת טקסט אוטומטית – ה-ARN שונה לגמרי.

- שורה 7: שנה את "customer" ל-"employee"

3. שמירת הקובץ:

- שמור את השינויים שביצעת בקובץ.

4. הרצת פקודת CLI מתאימה:

- הרץ את פקודת ה-AWS CLI ליצירת השירות ב-ECS תוך שימוש בקובץ ה-JSON שערכת.

שלב 8.1 יצירת CodeDeploy Application

Developer Tools > CodeDeploy > Applications > Create application

Create application

Application configuration

Application name
Enter an application name

100 character limit

Compute platform
Choose a compute platform

Amazon ECS ▼

Tags

Add tag

Cancel

Create application

יצירת Deployment group עבור customer

Create deployment group

Application

Application
microservices
Compute type
Amazon ECS

Deployment group name

Enter a deployment group name

microservices-customer

100 character limit

Service role

Enter a service role

Enter a service role with CodeDeploy permissions that grants AWS CodeDeploy access to your target instances.

Q arn:aws:iam::215251297864:role/DeployRole



Environment configuration

Choose an ECS cluster name

microservices-serverlesscluster

Choose an ECS service name

customer-microservice

Load balancers

Choose a load balancer

microservicesLB

Production listener port

HTTP: 80

Test listener port - optional

A test listener is required if you want to test your replacement version before traffic reroutes to it

HTTP: 8080

Target group 1 name

customer-tg-two

Target group 2 name

customer-tg-one

Deployment settings

Traffic rerouting

Choose whether traffic reroutes to the replacement environment immediately or waits for you to start the rerouting process

☒ Reroute traffic immediately

☐ Specify when to reroute traffic

Deployment configuration

Choose from a list of default and custom deployment configurations. A deployment configuration is a set of rules that determines how fast an application is deployed and the success or failure conditions for a deployment.

CodeDeployDefault.ECSAllAtOnce

or

Create deployment configuration

Original revision termination

Specify how long CodeDeploy waits before it terminates the original task set. After termination starts, you cannot rollback manually or automatically

Days

0

Hours

0

Minutes

5

באופן דומה גם עבור Employee.

Application

Application
microservices
Compute type
Amazon ECS

Deployment group name

Enter a deployment group name

microservices-employee

100 character limit

Service role

Enter a service role

Enter a service role with CodeDeploy permissions that grants AWS CodeDeploy access to your target instances.

Q am:aws:iam::215251297864:role/DeployRole



Environment configuration

Choose an ECS cluster name

microservices-serverlesscluster

Choose an ECS service name

employee-microservice

Load balancers

Choose a load balancer

microservicesLB

Production listener port

HTTP: 80

Test listener port - *optional*

A test listener is required if you want to test your replacement version before traffic reroutes to it

HTTP: 8080

Target group 1 name

employee-tg-two

Target group 2 name

employee-tg-one

Deployment settings

Traffic rerouting

Choose whether traffic reroutes to the replacement environment immediately or waits for you to start the rerouting process

☒ Reroute traffic immediately

☐ Specify when to reroute traffic

Deployment configuration

Choose from a list of default and custom deployment configurations. A deployment configuration is a set of rules that determines how fast an application is deployed and the success or failure conditions for a deployment.

CodeDeployDefault.ECSAllAtOnce



or

Create deployment configuration

Original revision termination

Specify how long CodeDeploy waits before it terminates the original task set. After termination starts, you cannot rollback manually or automatically

Days

0

Hours

0

Minutes

5

שלב 8.2 יצירת CodePipeline

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 1
Choose creation option [Info](#)
Step 1 of 7

Step 2
Choose pipeline settings

Step 3
Add source stage

Step 4
Add build stage

Step 5
Add test stage

Step 6
Add deploy stage

Step 7
Review

Category

☐ Deployment ☐ Continuous Integration ☐ Automation

☒ Build custom pipeline

Cancel **Next**

Choose pipeline settings [Info](#)
Step 2 of 7

Pipeline settings

Pipeline name
Enter the pipeline name. You cannot edit the pipeline name after it is created.

No more than 100 characters

Execution mode [Info](#)
Choose the execution mode for your pipeline. This determines how the pipeline is run.

☐ Superseded

☒ Queued

☐ Parallel

Service role

☐ New service role
Create a service role in your account

☒ Existing service role
Choose an existing service role from your account

Role ARN

Add source stage Info

Step 3 of 7

Source

Source provider
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.

AWS CodeCommit

Repository name
Choose a repository that you have already created where you have pushed your source code.

deployment

Branch name
Choose a branch of the repository

dev

☒ Create EventBridge rule to automatically detect source changes
If disabled, follow AWS documentation to create an EventBridge rule for your source. [Learn more](#)

Output artifact format
Choose the output artifact format.

☒ **CodePipeline default**
AWS CodePipeline uses the default zip format for artifacts in the pipeline. Does not include Git metadata about the repository.

☐ **Full clone**
AWS CodePipeline passes metadata about the repository that allows subsequent actions to do a full Git clone. Only supported for AWS CodeBuild actions. [Learn more](#)

☒ Enable automatic retry on stage failure

Cancel Previous Next

עושים skip ל-build ו-test

נכנסים ל-Pipeline שנוצר

Introducing the new pipeline experience
We've redesigned the pipeline view to streamline the monitoring and debugging experience. Let us know what you think. Or go back to the old experience. Don't show again

Developer Tools > CodePipeline > Pipelines > update-customer-microservice

update-customer-microservice Edit Stop execution Create trigger Clone pipeline Release change

Pipeline Executions Triggers Settings Tags **Stage**

Source Succeeded

Source
[AWS CodeCommit](#)

✓ Succeeded - 1 minutes ago
a64aaa1f

a64aaa1f Source: step 5 updates

מוסיפים ל-Source

Edit: Source

Cancel Done

+ Add action group

Source AWS CodeCommit	Image Amazon ECR	+ Add action
--	---------------------	--------------

+ Add action group

Automated stage configuration: Enable automatic retry on stage failure Retry mode: Retry failed stage

Edit action

Action name

Choose a name for your action

Image

No more than 100 characters

Action provider

Amazon ECR

Repository name

Choose an Amazon ECR repository as the source location.

customer

Image tag - optional

Choose the image tag that triggers your pipeline when a change occurs in the image repository.

Variable namespace - optional

Choose a namespace for the output variables from this action. You must choose a namespace if you want to use the variables this action produces in your configuration. [Learn more](#)

Output artifacts

Choose a name for the output of this action.

image-customer

No more than 100 characters

Cancel

Done

עורכים את ה-Deploy שנכשל

Developer Tools > CodePipeline > Pipelines > update-customer-microservice

update-customer-microservice

Edit

Stages

Pipeline

Executions

Triggers

Settings

Tags

Stage

Deploy Failed

Deploy
Amazon ECS (Blue/Green)

Failed - 5 minutes ago

a64aaa1f Source: step 5 updates

Edit: Deploy

Edit stage

Conditions

Entry: Not configured

Success: Not configured

Failure: Not configured

View details

Deploy

Amazon ECS (Blue/Green)

Automated stage configuration:

Enable automatic rollback on stage failure

Input artifacts

Choose an input artifact for this action. [Learn more](#)

SourceArtifact Defined by: Source

image-customer Defined by: Image

No more than 100 characters

AWS CodeDeploy application name

Choose one of your existing applications, or create a new one in AWS CodeDeploy.

microservices

Create application

AWS CodeDeploy deployment group

Choose one of your existing deployment groups, or create a new one in AWS CodeDeploy.

microservices-customer

Amazon ECS task definition

Choose the input artifact where your Amazon ECS task definition file is stored. If other than the default file path, specify the path and filename of your task definition file.

SourceArtifact

taskdef-customer.json

The default path is taskdef.json.

AWS CodeDeploy AppSpec file

Choose the input artifact where your AWS CodeDeploy AppSpec file is stored. If other than the default file path, specify the path and filename of your AppSpec file.

SourceArtifact

appspec-customer.yaml

Dynamically update task definition image - optional

You can provide an input artifact and a placeholder name for the container definition image that will be used to dynamically update a task definition. You can specify multiple input artifacts and placeholders.

Input artifact with image details

image-customer

Placeholder text in the task definition

IMAGE1_NAME

Remove

Add

Variable namespace - optional

חשוב לשמור את השינויים

Save pipeline changes



Saving your changes cannot be undone. If the pipeline is running when you save your changes, that execution will not complete.



Source action changed

We will update the following resources to detect changes for your updated pipeline.

Change type

Details

Add

Pipeline update-employee-microservice as a target to Amazon CloudWatch Events rule - codepipeline

☐ No resource updates needed for this source action change

Cancel

Save

שלב 8.4 חזור על שלב 8.3 וצור Pipeline עבור employee שלב 8.5 אחרי שמירת השינויים בדוק את המערכת

Developer Tools

CodeDeploy

Source • CodeCommit

Artifacts • CodeArtifact

Build • CodeBuild

Deploy • CodeDeploy

Getting started

Deployments

Deployment

Applications

Deployment configurations

On-premises instances

Pipeline • CodePipeline

Settings

Go to resource

Feedback

Developer Tools > CodeDeploy > Deployments > d-XLZL3JNZC

d-XLZL3JNZC

Copy deployment

Retry deployment

Step 1:

Deploying replacement task set

Completed

Succeeded

100%

Step 2:

Test traffic route setup

Completed

Succeeded

100%

Step 3:

Rerouting production traffic to replacement task set

100% traffic shifted

Succeeded

100%

Step 4:

Wait

Wait completed

Succeeded

100%

Step 5:

Terminate original task set

Completed

Succeeded

100%

Traffic shifting progress

Original

0.0%

Original task set not serving traffic

Replacement

100.0%

Replacement task set serving traffic

Deployment details

Amazon Elastic Container Service > Clusters > microservices-serverlesscluster > Services

Amazon Elastic Container Service

Clusters

- Namespaces
- Task definitions
- Account settings

Install AWS Copilot

Amazon ECR

- Repositories

AWS Batch

Documentation

Discover products

Subscriptions

Tell us what you think

microservices-serverlesscluster

Last updated May 25, 2025 at 20:22 (UTC+3:00) [Update cluster](#) [Delete cluster](#)

Cluster overview

ARN arn:aws:ecs:us-east-1:215251297864:cluster/microservices-serverlesscluster	Status Active	CloudWatch monitoring Default	Registered container instances -
Services Draining -	Active 2	Tasks Pending -	Running 2

[Services](#) [Tasks](#) [Infrastructure](#) [Metrics](#) [Scheduled tasks](#) [Configuration](#) [Tags](#)

Services (2) [Info](#)

Filter services by value

Filter launch type: Any launch type

Filter service type: Any service type

<input type="checkbox"/>	Service name	ARN	Status	Service type	Created at	Deployments and tasks	Last deployment	Task definition
<input type="checkbox"/>	customer-microservice	arn:aws:ecs:us-east-1:215251297864:task-definition/customer-microservice	Active	REPLICA	3 hours ago	1/1 Tasks running	Succeeded	View customer-microservice
<input type="checkbox"/>	employee-microservice	arn:aws:ecs:us-east-1:215251297864:task-definition/employee-microservice	Active	REPLICA	3 hours ago	1/1 Tasks running	In progress	View employee-microservice

EC2 > Target groups

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

Dedicated Hosts

Capacity Reservations

Images

- AMIs
- AMI Catalog

Elastic Block Store

- Volumes
- Snapshots
- Lifecycle Manager

Target groups (4)

Filter target groups

<input type="checkbox"/>	Name	ARN	Port	Protocol	Target type	Load balancer	VPC ID
<input type="checkbox"/>	customer-tg-one	arn:aws:elasticloadbalancing:us-east-1:215251297864:target-group/customer-tg-one	8080	HTTP	IP	microservicesLB	vpc-02fcd3ba18b811d2f
<input type="checkbox"/>	customer-tg-two	arn:aws:elasticloadbalancing:us-east-1:215251297864:target-group/customer-tg-two	8080	HTTP	IP	None associated	vpc-02fcd3ba18b811d2f
<input type="checkbox"/>	employee-tg-one	arn:aws:elasticloadbalancing:us-east-1:215251297864:target-group/employee-tg-one	8080	HTTP	IP	microservicesLB	vpc-02fcd3ba18b811d2f
<input type="checkbox"/>	employee-tg-two	arn:aws:elasticloadbalancing:us-east-1:215251297864:target-group/employee-tg-two	8080	HTTP	IP	None associated	vpc-02fcd3ba18b811d2f

EC2 > Load balancers > microservicesLB

Application Load Balancers now support public IPv4 IP Address Management (IPAM). You can get started with this feature by configuring IP pools in the Network mapping section. [Edit IP pools](#)

microservicesLB

Details

Load balancer type Application	Status Active	VPC vpc-02fcd3ba18b811d2f	Load balancer IP address type IPv4
Scheme Internet-facing	Hosted zone Z355XDOTRQ7X7K	Availability Zones subnet-02c6d20c244757bf7 (us-east-1b (use1-az6)) subnet-080c6d43d1bf8e81a (us-east-1a (use1-az4))	Date created May 25, 2025, 16:59 (UTC+03:00)
Load balancer ARN arn:aws:elasticloadbalancing:us-east-1:215251297864:loadbalancer/app/microservicesLB/1f5ff2623d1e35e8		DNS name info microserviceslb-368664912.us-east-1.elb.amazonaws.com (A Record)	

[Listeners and rules](#) [Network mapping](#) [Resource map](#) [Security](#) [Monitoring](#) [Integrations](#) [Attributes](#) [Capacity](#) [Tags](#)

Listeners and rules (2) [Info](#)

A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.

Filter listeners

<input type="checkbox"/>	Protocol:Port	Default action	Rules	ARN	Security policy	Default SSL/TLS certificate	mTLS	Trust store
<input type="checkbox"/>	HTTP:8080	Forward to target group customer-tg-one (100%) Target group stickiness: Off	2 rules	ARN	Not applicable	Not applicable	Not applicable	Not applicable
<input type="checkbox"/>	HTTP:80	Forward to target group customer-tg-one (100%) Target group stickiness: Off	2 rules	ARN	Not applicable	Not applicable	Not applicable	Not applicable

שני החוקים מצביעים ל-customer-tg-one

שלב 9.1 עורכים את ההגדרות של HTTP:80

The screenshot shows the AWS Management Console interface for the 'microservicesLB' load balancer. The 'HTTP:80 listener' configuration is displayed. A green banner at the top indicates 'Successfully updated rule on listener HTTP:80.' The 'Details' section shows the Protocol as HTTP:80, the Load balancer as microservicesLB, and the Listener ARN. The 'Rules' section shows a table with two rules. The first rule is selected, showing its details: Name tag, Priority 1, Conditions (IF) Path Pattern is /admin/*, and Actions (Then) Forward to target group (employee-tg-two, 1 (100%), Target group stickiness: Off). The second rule is the default rule, showing its details: Name tag, Priority Last (default), Conditions (IF) If no other rule applies, and Actions (Then) Forward to target group (customer-tg-two, 1 (100%), Target group stickiness: Off).

Edit listener

Edit the protocol, port or default actions of your Application Load Balancer (ALB) listener.

► Load balancer details: microservicesLB

Listener details

A listener checks for connection requests using the protocol and port that you configure. The default action and any additional rules that you create determine how the Application Load Balancer routes requests to its registered targets.

Listener ARN
arn:aws:elasticloadbalancing:us-east-1:1215251297864:listener/app/microservicesLB/1f5f2623d1e35e8/c95b046128034157

Listener configuration

The listener will be identified by the protocol and port.

Protocol
Used for connections from clients to the load balancer.
HTTP

Port
The port on which the load balancer is listening for connections.
80
1-65535

Default actions

The default action is used if no other rules apply. Choose the default action for traffic on this listener.

Routing actions

☒ Forward to target groups ☐ Redirect to URL ☐ Return fixed response

Forward to target group
Choose a target group and specify routing weight or [Create target group](#).

Target group
customer-tg-two
Target type: IP, IPv4

Weight
1
0-999

Percent
100%

להצביע חזרה ל- employee-tg-two ו-customer-tg-two
ובנוסף נוסיף עוד תנאי ל-employee המשמש רק למשתמש המחובר דרך כתובת IP
ספציפית עבור שני הפורטים 8080 & HTTP:80

Edit rule

Define the rule and then review it in the context of the other rules on this listener.

► Listener details: HTTP:8080

► Name and tags

Conditions

Requests reaching the listener

Path (1)
if Path is /admin/*

Actions

These actions will be performed if the rule is matched.

Action types

Routing actions

Add condition

Rule condition types
Route traffic based on the condition type of each request. Each rule can include one of each of the following conditions: host-header, path, http-request-method and source-ip. Each rule can include one or more of each of the following conditions: http-header and query-string.

Source IP
Source IP
Specify from where the data or request is coming. If your client is behind a proxy, enter the IP address of the proxy.

is 79.177.132.253/32

Source IP must be a CIDR notation. Valid CIDRs are IPv4 (Example: 10.24.34.0/23) and IPv6 (Example: 2001:db8::/32).

[Add new value](#)

You can add up to 5 more condition values for this rule.

[Cancel](#) [Confirm](#)

Set rule priority [Info](#)

Each rule has a priority. The default rule is evaluated last. You can change the priority of a non-default rule at any time. You can't change the priority of the default rule.

► Listener details: HTTP:8080

Listener rules (2) [Info](#) [Rule limits](#) [Reset priorities](#) [Add gap between priorities](#)

Traffic received by the listener is routed according to the default action and any additional rules. Rules are evaluated in priority order from the lowest value to the highest value.

Name tag	Priority	Conditions (If)	Actions (Then)	ARN	Tags
-	1	<ul style="list-style-type: none"> Path Pattern is /admin/*, AND Source IP is 79.177.132.253/32 	Forward to target group <ul style="list-style-type: none"> employee-tg-one [3]: 1 (100%) Target group stickiness: Off 	ARN	0 tags
Default	Last (default)	If no other rule applies	Forward to target group <ul style="list-style-type: none"> customer-tg-one [3]: 1 (100%) Target group stickiness: Off 	ARN	0 tags

מבצעים את השינוי בקובץ `microservices\employee\view\nav.html` בשורה 1

`navbar-dark bg-dark to navbar-light bg-light`

שלב 9.2 - מעדכנים את ה-Docker Image

```

=> exporting to image
=> exporting layers
=> writing image sha256:699e911fberf248e944440a2516033659dbba0f51a1cf3975f80975bf60ac183
=> naming to docker.io/library/employee
voclabs:~/environment/microservices/employee (dev) $ dbEndpoint=$(cat ~/environment/microservices/employee/app/config/config.js | grep 'APP_DB_HOST' | cut -d '"' -f2)
voclabs:~/environment/microservices/employee (dev) $ echo $dbEndpoint
suppliferdb.cryakacqj1.us-east-1.rds.amazonaws.com
voclabs:~/environment/microservices/employee (dev) $ account_id=$(aws sts get-caller-identity |grep Account|cut -d '"' -f4)
voclabs:~/environment/microservices/employee (dev) $ echo $account_id
215251297864
voclabs:~/environment/microservices/employee (dev) $ docker tag employee:latest $account_id.dkr.ecr.us-east-1.amazonaws.com/employee:latest
voclabs:~/environment/microservices/employee (dev) $ aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin $account_id.dkr.ecr.us-east-1.amazonaws.com
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
voclabs:~/environment/microservices/employee (dev) $ docker push $account_id.dkr.ecr.us-east-1.amazonaws.com/employee:latest
The push refers to repository [215251297864.dkr.ecr.us-east-1.amazonaws.com/employee]
78c0b56f7027: Pushed
949374de4ba6: Pushed
5f70bf18a086: Layer already exists
6e59513952c1: Layer already exists
d81d715330b7: Layer already exists
1dc7f3bb09a4: Layer already exists
d0ce6b728084: Layer already exists
f1b5933fe4b5: Layer already exists
tag invalid: The image tag 'latest' already exists in the 'employee' repository and cannot be overwritten because the tag is immutable.

```

אם יש שגיאה של `tag invalid` אז צריך לשנות את ה-Repository tag ל- `Mutable` דרך ECR

שלב 9.3 - בודקים שהמערכת התעדכנה דרך CodeDeploy אחרי הוספת השינויים

שלב 9.4 - בדיקה עם מכשיר נוסף מרשת שונה שניתן לגשת רק למערכת של Customer

שלב 9.5 - הרחבת ה-Customers למכונות נוספות לעמידה בעומס בעזרת הפקודה

```
aws ecs update-service --cluster microservices-serverlesscluster --service customer-microservice --desired-count 3
```

ב-Cloud9

סיימנו!