הנה מדריך מלא ומפורט לביצוע הפריסה של הפרויקט לניטור רמות הים ב-AWS, הכולל את כל השלבים הנדרשים לפריסת ה-Backend וה-Frontend, הגדרת משתני סביבה, ובדיקות ומוניטורינג.

---

# מדריך פריסה מלא לפרויקט ניטור רמות הים

מדריך זה מספק הנחיות שלב אחר שלב לפריסת הפרויקט ב-AWS.

## פריסת ה-Backend

### 1. אריזת תיקיית `shared` כ-AWS Lambda Layer

- צור תיקייה בשם `python` בתוך תיקייה חדשה (למשל, `shared\_layer`).

- העתק את כל הקבצים מתיקיית `shared` לתוך `shared\_layer/python`.

- דחוס את התיקייה לקובץ ZIP:

```bash

cd shared\_layer

zip -r ../shared\_layer.zip .

```

- העלה את ה-Layer ל-AWS דרך ה-Lambda Console:

- עבור ל-"Layers" > "Create layer".

- הזן שם (למשל, `shared-layer`), העלה את ה-ZIP, ובחר Runtime (למשל, Python 3.9).

- שמור את ה-ARN של ה-Layer לשימוש מאוחר יותר.

### 2. פריסת פונקציות Lambda

עבור כל פונקציה (למשל, `get\_stations`):

- ארוז את הקוד:

- ודא שיש `main.py` ו-`requirements.txt` בתיקייה.

- התקן תלויות:

```bash

pip install -r requirements.txt -t ./package

```

- דחוס לקובץ ZIP:

```bash

cd backend/lambdas/get\_stations

zip -r ../../../get\_stations.zip .

```

- צור את הפונקציה ב-AWS באמצעות AWS CLI:

```bash

aws lambda create-function --function-name get\_stations --runtime python3.9 --handler main.handler --zip-file fileb://get\_stations.zip --role arn:aws:iam::YOUR\_ACCOUNT:role/lambda-role --layers arn:aws:lambda:us-east-1:YOUR\_ACCOUNT:layer:shared-layer:1

```

- חזור על התהליך עבור כל פונקציה נוספת.

### 3. הגדרת API Gateway

- צור REST API חדש ב-API Gateway Console.

- הוסף Resources ו-Methods (למשל, `/stations` עם GET שמפנה ל-`get\_stations`).

- הפעל CORS עבור כל Method.

- פרוס את ה-API וקבל את ה-Invoke URL.

### 4. הגדרת Cognito Authorizer

- צור User Pool ב-Cognito Console עם App Client.

- הוסף Authorizer ב-API Gateway והחל אותו על Methods הדורשים אימות.

## פריסת ה-Frontend

### 1. בניית ה-React App

- נווט ל-`frontend/` ובצע:

```bash

npm run build

```

### 2. העלאת תיקיית `build` ל-S3

- צור S3 Bucket והפעל "Static website hosting".

- העלה את הקבצים:

```bash

aws s3 sync ./build/ s3://sea-level-frontend-bucket

```

### 3. הגדרת CloudFront

- צור CloudFront Distribution עם ה-Bucket כ-Origin.

- הגדר `index.html` כ-Default Root Object.

- שמור את ה-Domain Name של CloudFront.

## משתני סביבה

### 1. Backend: הגדרת `DB\_URI`

- ב-Lambda Console, הוסף משתנה סביבה `DB\_URI` לכל פונקציה עם ה-URI של ה-DB.

### 2. Frontend: עדכון `API\_BASE\_URL`

- ערוך את `App.js` עם ה-Invoke URL של ה-API Gateway:

```javascript

const API\_BASE\_URL = 'https://YOUR\_API\_ID.execute-api.us-east-1.amazonaws.com/prod';

```

- בצע build והעלה מחדש ל-S3.

## בדיקות ומוניטורינג

### 1. בדיקות עם Postman

- קבל טוקן JWT מ-Cognito.

- בדוק כל endpoint עם הטוקן ב-Postman.

### 2. הגדרת CloudWatch

- בדוק Logs ב-CloudWatch עבור כל פונקציה.

- הגדר Alarms לשגיאות או זמני תגובה איטיים.

### 3. שיפור ביצועים עם ElastiCache

- צור Redis Cluster ב-ElastiCache.

- עדכן את קוד ה-Lambda לשימוש ב-Cache עבור קריאות תכופות.

---

# מדריך פריסה מלא לפרויקט ניטור רמות הים

## פריסת ה-Backend

### 1. אריזת תיקיית `shared` כ-AWS Lambda Layer

- צור תיקייה בשם `python` בתוך תיקייה חדשה (למשל, `shared\_layer`).

- העתק את כל הקבצים מתיקיית `shared` לתוך `shared\_layer/python`.

- דחוס את התיקייה לקובץ ZIP:

```bash

cd shared\_layer

zip -r ../shared\_layer.zip .

```

- העלה את ה-Layer ל-AWS דרך ה-Lambda Console:

- עבור ל-"Layers" > "Create layer".

- הזן שם (למשל, `shared-layer`), העלה את ה-ZIP, ובחר Runtime (למשל, Python 3.9).

- שמור את ה-ARN של ה-Layer לשימוש מאוחר יותר.

### 2. פריסת פונקציות Lambda

עבור כל פונקציה (למשל, `get\_stations`):

- ארוז את הקוד:

- ודא שיש `main.py` ו-`requirements.txt` בתיקייה.

- התקן תלויות:

```bash

pip install -r requirements.txt -t ./package

```

- דחוס לקובץ ZIP:

```bash

cd backend/lambdas/get\_stations

zip -r ../../../get\_stations.zip .

```

- צור את הפונקציה ב-AWS באמצעות AWS CLI:

```bash

aws lambda create-function --function-name get\_stations --runtime python3.9 --handler main.handler --zip-file fileb://get\_stations.zip --role arn:aws:iam::YOUR\_ACCOUNT:role/lambda-role --layers arn:aws:lambda:us-east-1:YOUR\_ACCOUNT:layer:shared-layer:1

```

- חזור על התהליך עבור כל פונקציה נוספת.

### 3. הגדרת API Gateway

- צור REST API חדש ב-API Gateway Console.

- הוסף Resources ו-Methods (למשל, `/stations` עם GET שמפנה ל-`get\_stations`).

- הפעל CORS עבור כל Method.

- פרוס את ה-API וקבל את ה-Invoke URL.

### 4. הגדרת Cognito Authorizer

- צור User Pool ב-Cognito Console עם App Client.

- הוסף Authorizer ב-API Gateway והחל אותו על Methods הדורשים אימות.

## פריסת ה-Frontend

### 1. בניית ה-React App

- נווט ל-`frontend/` ובצע:

```bash

npm run build

```

### 2. העלאת תיקיית `build` ל-S3

- צור S3 Bucket והפעל "Static website hosting".

- העלה את הקבצים:

```bash

aws s3 sync ./build/ s3://sea-level-frontend-bucket

```

### 3. הגדרת CloudFront

- צור CloudFront Distribution עם ה-Bucket כ-Origin.

- הגדר `index.html` כ-Default Root Object.

- שמור את ה-Domain Name של CloudFront.

## משתני סביבה

### 1. Backend: הגדרת `DB\_URI`

- ב-Lambda Console, הוסף משתנה סביבה `DB\_URI` לכל פונקציה עם ה-URI של ה-DB.

### 2. Frontend: עדכון `API\_BASE\_URL`

- ערוך את `App.js` עם ה-Invoke URL של ה-API Gateway:

```javascript

const API\_BASE\_URL = 'https://YOUR\_API\_ID.execute-api.us-east-1.amazonaws.com/prod';

```

- בצע build והעלה מחדש ל-S3.

## בדיקות ומוניטורינג

### 1. בדיקות עם Postman

- קבל טוקן JWT מ-Cognito.

- בדוק כל endpoint עם הטוקן ב-Postman.

### 2. הגדרת CloudWatch

- בדוק Logs ב-CloudWatch עבור כל פונקציה.

- הגדר Alarms לשגיאות או זמני תגובה איטיים.

### 3. שיפור ביצועים עם ElastiCache

- צור Redis Cluster ב-ElastiCache.

- עדכן את קוד ה-Lambda לשימוש ב-Cache עבור קריאות תכופות.

---

זהו המדריך המלא לפריסת הפרויקט. הוא עונה על כל הדרישות ומספק מערכת מודרנית, גמישה וסקיילבילית לניטור רמות הים.