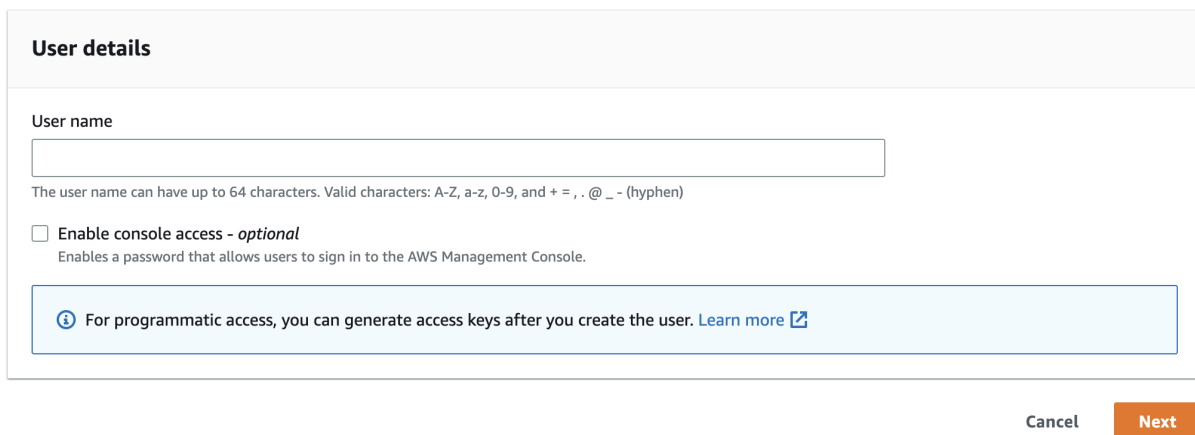# Steps to implement Hands-on Project - Mission 2

## Amazon Web Services

- Access AWS console and go to IAM service

- Under Access management, Click in "Users", then "Add users". Insert the User name **luxxy-covid-testing-system-en-app1** and click in **Next** to create a programmatic user.

### Specify user details

| User details |
| --- |
| User name |
| |
| The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ _ - (hyphen) |
| ☐ Enable console access – *optional* |
| Enables a password that allows users to sign in to the AWS Management Console. |
| ⓘ For programmatic access, you can generate access keys after you create the user. Learn more 🗗 |

Cancel    **Next**

- On Set permissions, Permissions options, click in "Attach policies directly" button.

## Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. Learn more ⬈

**Permissions options**

| ○ **Add user to group** | ○ **Copy permissions** | ⦿ **Attach policies directly** |
|---|---|---|
| Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function. | Copy all group memberships, attached managed policies, and inline policies from an existing user. | Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group. |

**Permissions policies** (1037)         ↻    Create policy ⬈
Choose one or more policies to attach to your new user.

🔍 Filter distributions by text, property or value      ‹ **1** 2 3 4 5 6 7 … 52 › ⚙

| ☐ | Policy name ⬈ ▲ | Type ▽ | Attached entities ▽ |
|---|---|---|---|
| ☐ ⊞ | 📦 AccessAnalyzerServiceRolePolicy | AWS managed | 0 |
| ☐ ⊞ | 📦 AdministratorAccess | AWS managed - job function | 1 |
| ☐ ⊞ | 📦 AdministratorAccess-Amplify | AWS managed | 0 |
| ☐ ⊞ | 📦 AdministratorAccess-AWSElasticBeanstalk | AWS managed | 0 |
| ☐ ⊞ | 📦 AlexaForBusinessDeviceSetup | AWS managed | 0 |
| ☐ ⊞ | 📦 AlexaForBusinessFullAccess | AWS managed | 0 |

- Type **AmazonS3FullAccess** in **Search**.

- Select **AmazonS3FullAccess**

---

**Permissions policies** (1/1037)        ↻    Create policy ⬈
Choose one or more policies to attach to your new user.

🔍 Filter distributions by text, property or value    1 match      ‹ **1** › ⚙

[ AmazonS3FullAccess ✕ ]   **Clear filters**

| ☑ | Policy name ⬈ ▲ | Type ▽ | Attached entities ▽ |
|---|---|---|---|
| ☑ ⊞ | 📦 AmazonS3FullAccess | AWS managed | 2 |

- Click in **Next**

- Review all details and click in Create user

## Review and create

Review your choices. After you create the user, you can view and download the autogenerated password, if enabled.

**User details**

| User name | Console password type | Require password reset |
|---|---|---|
| terraform-en-1 | None | No |

**Permissions summary**

⟨ 1 ⟩

| Name ↗ ▽ | Type ▽ | Used as ▽ |
|---|---|---|
| AmazonS3FullAccess | AWS managed | Permissions policy |

**Tags** - *optional*

Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

No tags associated with the resource.

[Add new tag]

You can add up to 50 more tags.

Cancel    Previous    Create user

# Steps to create access key:

- Click on the user you have created.

- Go to Security credentials tab.

- Scroll down and go to Access keys section.

- Click on Create access key

**Access keys (0)**

Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. Learn more ↗

[Create access key]

**No access keys**

As a best practice, avoid using long-term credentials like access keys. Instead, use tools which provide short term credentials. Learn more ↗

[Create access key]

- Select **Command Line Interface (CLI)** and **I understand the above recommendation and want to proceed to create an access key** checkbox.

- Click Next

- Click on Create access key

- Click on Download .csv file

- After download, click Done.

- Now, rename .csv file downloaded to **luxxy-covid-testing-system-en-app1.csv**

## Google Cloud Platform (GCP)

- Navigate to Cloud SQL instance and create a new user **app** with password **welcome123456** on Cloud SQL MySQL database

https://prod-files-secure.s3.us-west-2.amazonaws.com/9a82f1a4-c086-4469-a0ee-c028781a5e80/e42adbaf-eb30-44dd-a3b4-23c0e8346ccf/SQL_User_Creation_EDITED.mp4

- Connect to Google Cloud Shell

- **Download** the mission2 files to Google Cloud Shell using the wget command as shown below

```
cd ~
mkdir mission2_en
cd mission2_en
wget https://tcb-public-events.s3.amazonaws.com/icp/mission2.zip
unzip mission2.zip
```

- Connect to MySQL DB running on Cloud SQL (once it prompts for the password, provide **welcome123456**)

```
mysql --host=<public_ip_cloudsql> --port=3306 -u app -p
```

- Once you're connected to the database instance, create the products table for testing purposes

```
use dbcovidtesting;
source ~/mission2_en/mission2/en/db/create_table.sql
show tables;
exit;
```

- Enable Cloud Build API via Cloud Shell.

```
# Command to enable Cloud Build API

gcloud services enable cloudbuild.googleapis.com
```

**Known issue during this step**

If you see the error below, please follow the steps to fix it:

```
ERROR: (gcloud.builds.submit) INVALID_ARGUMENT: could not resolve source: googleapi: Error
403: 989404026119@cloudbuild.gserviceaccount.com does not have storage.objects.get access
to the Google Cloud Storage object., forbidden

To solve it:

1. Access IAM & Admin;
2. Click on check-box Include Google-provided role grants;
3. Click and select your Cloud Build Service Account

Example: 989404026119@cloudbuild.gserviceaccount.com Cloud Build Service Account

4. On your Cloud Build Service Account, right side, click on Edit principal
5. Click on Add another role
6. Click on Select Role, and filter by Storage Admin or gcs. Select Storage Admin (Full co
ntrol of GCS resources).
7. Click on Save and go to Cloud Shell.
```

- Build the Docker image and push it to Google Container Registry. Please replace the <PROJECT_ID> with your My First Project ID.

```
cd ~/mission2_en/mission2/en/app
gcloud builds submit --tag gcr.io/<PROJECT_ID>/luxxy-covid-testing-system-app-en
```

- Open the Cloud Editor and edit the Kubernetes deployment file (**luxxy-covid-testing-system.yaml**) and update the variables below on **line 33** in **red** with your <PROJECT_ID> on the Google Container Registry path, on **line 42** AWS Bucket name, AWS Keys (open file l**uxxy-covid-testing-system-en-app1.csv** and use **Access key ID** on **line 44** and **Secret access key** on **line 46**)  and Cloud SQL Database Private IP on **line 48**.

```
cd ~/mission2/en/kubernetes
luxxy-covid-testing-system.yaml


        image: gcr.io/<PROJECT_ID>/luxxy-covid-testing-system-app-en:latest
...
        - name: AWS_BUCKET
          value: "luxxy-covid-testing-system-pdf-en-xxxx"
        - name: S3_ACCESS_KEY
          value: "xxxxxxxxxxxxxxxxxxxx"
        - name: S3_SECRET_ACCESS_KEY
          value: "xxxxxxxxxxxxxxxxxxx"
        - name: DB_HOST_NAME
          value: "172.21.0.3"
```

- Connect to the GKE (Google Kubernetes Engine) cluster via Console (follow the video)

- Deploy the application Luxxy in the Cluster

```
cd ~/mission2_en/mission2/en/kubernetes
kubectl apply -f luxxy-covid-testing-system.yaml
```

- Get the Public IP and test the application (<u>CLICK HERE to download COVID-19 Testing result sample</u>). Search for **GKE**, click on **Services & Ingress**, and then on **Endpoints** address:port

- You should see the app up & running! Congrats! 🎉

Guest Name... | Search | New | View Guest Results

# Luxxy - Luxury Hotels & Resorts

COVID-19 Testing Status System

**Add guest result »**