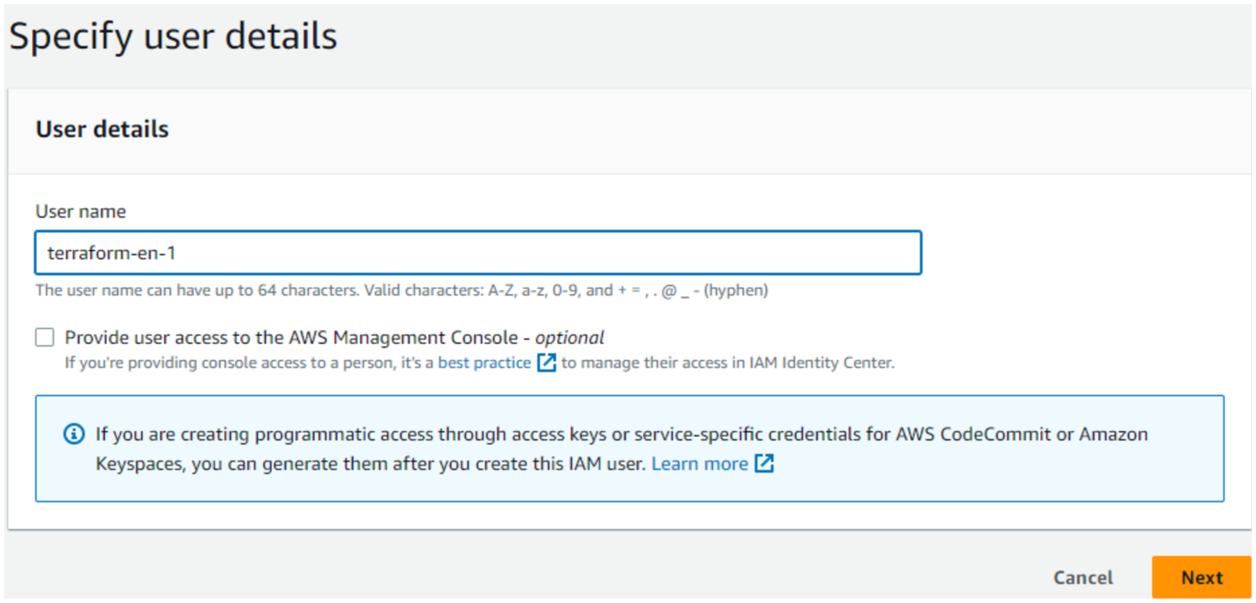
Steps to implement Hands-on Project - Mission 1

Amazon Web Services

- Access AWS console and go to IAM service
- Under Access management, Click in "Users", then "Add users". Insert the User name **terraform-en-1** and click in **Next** to create a programmatic user.



Specify user details

User details

User name

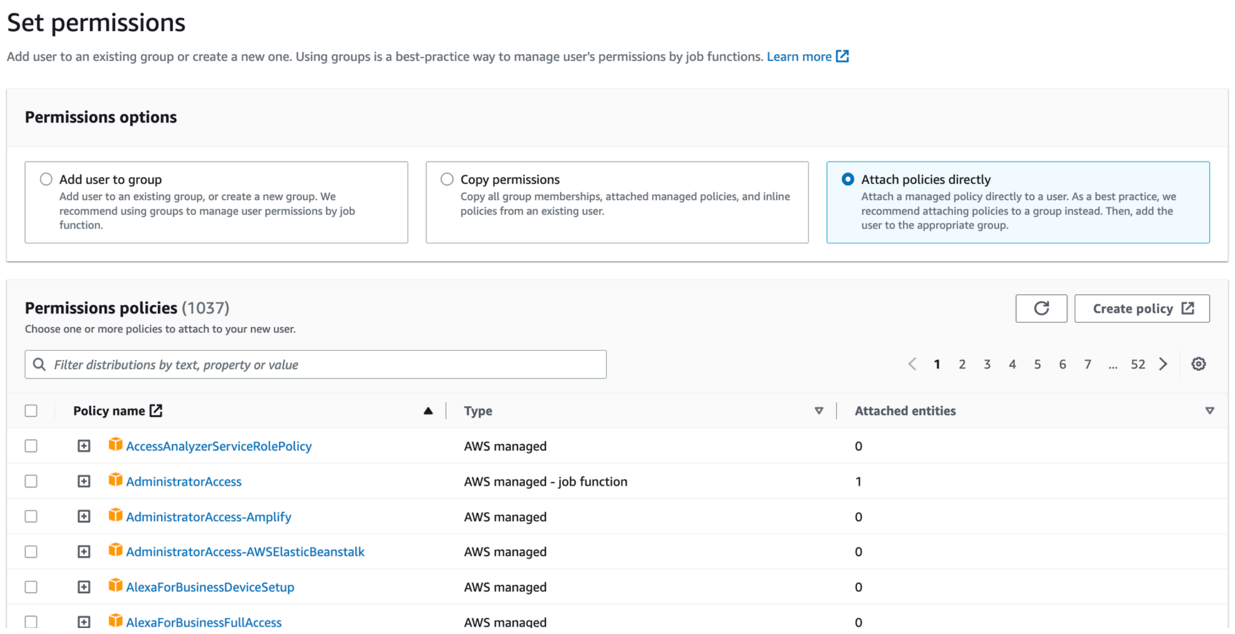
terraform-en-1

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ _ - (hyphen)

☐ Provide user access to the AWS Management Console - *optional*
If you're providing console access to a person, it's a best practice to manage their access in IAM Identity Center.

If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. Learn more

- 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 — Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.
- ○ Copy permissions — Copy all group memberships, attached managed policies, and inline policies from an existing user.
- ● Attach policies directly — 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.

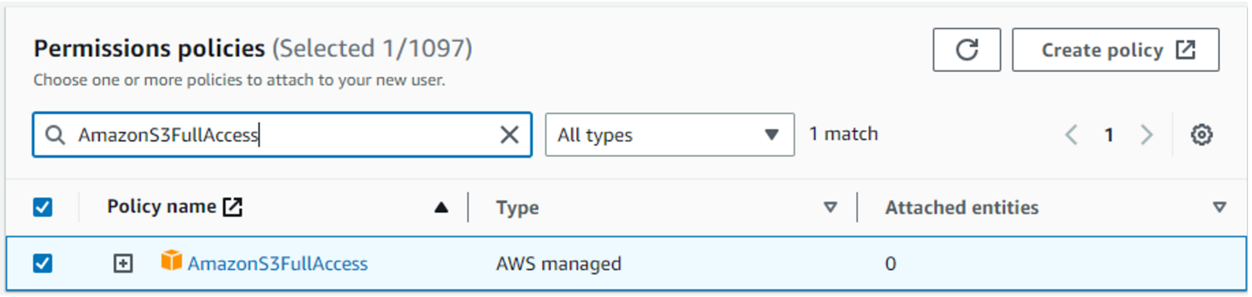
Permissions policies (1037)

Choose one or more policies to attach to your new user.

Filter distributions by text, property or value

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**

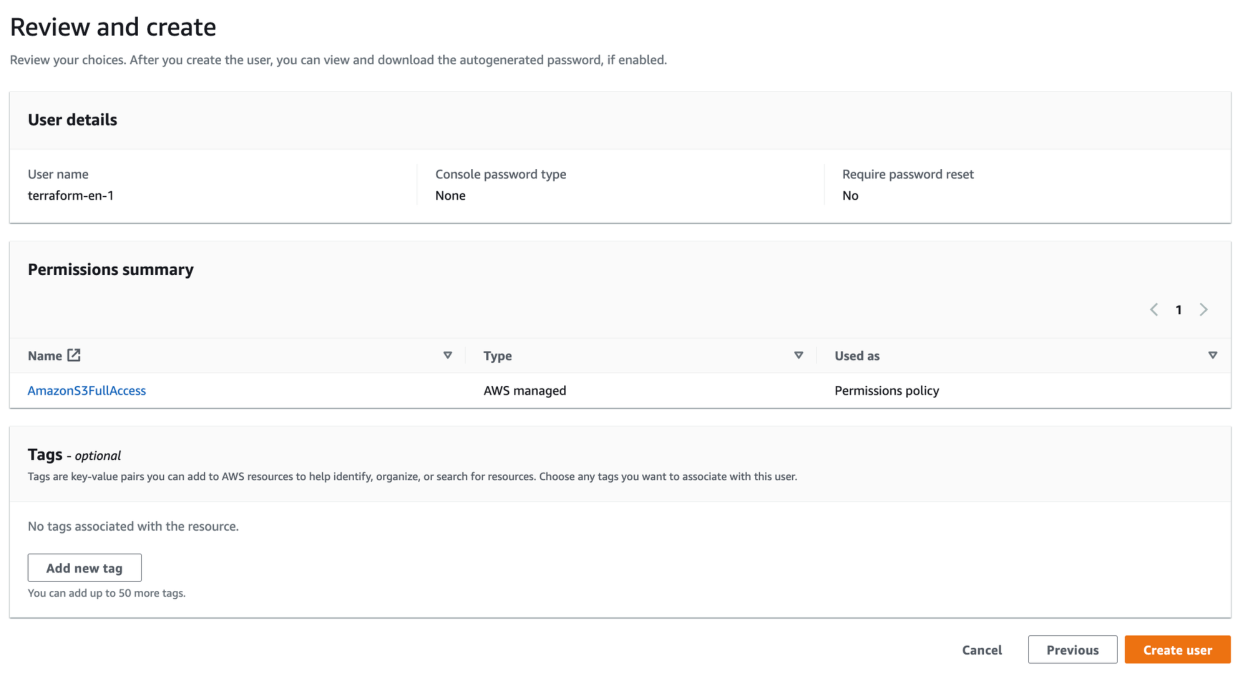

Permissions policies (Selected 1/1097)

Choose one or more policies to attach to your new user.

AmazonS3FullAccess | All types | 1 match

Policy name	Type	Attached entities
✓ AmazonS3FullAccess	AWS managed	0

- Click in **Next**
- Review all details, then click **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

Name	Type	Used as
AmazonS3FullAccess	AWS managed	Permissions policy

Tags - *optional*

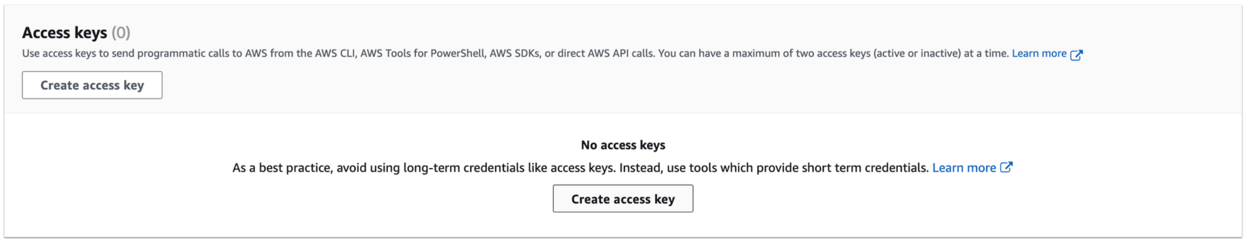
Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

No tags associated with the resource.

You can add up to 50 more tags.

[NEW] AWS has recently changed the way to download the key. Follow the new steps:

- Click on the user you have created.
- After this, click on **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

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

- 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 is finished, click on **Done**

▼ [TIP] Access key best practices

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

- After download, click **Done**.
- Now, rename .csv file downloaded to **accessKeys.csv**

Google Cloud Platform (GCP)

- CLICK HERE to download the mission1.zip hands-on files.
- Access GCP Console and open Cloud Shell
- Upload **accessKeys.csv** and **mission1.zip** hands-on file to GCP Cloud Shell
- Check if upload has been successfully completed using the command **ls -la**
- Hands-on files preparation

```
mkdir mission1_en mv mission1.zip mission1_en cd mission1_en unzip
mission1.zip mv ~/accessKeys.csv mission1/en cd mission1/en chmod +x *.sh
```

- Run the following commands to prepare AWS and GCP environment. Authorize when asked.

```
./aws_set_credentials.sh accessKeys.csv gcloud config set project
<project_id>
```

- Execute the command below

```
./gcp_set_project.sh
```

- Enable the Container Registry API, Kubernetes Engine API and the Cloud SQL API

```
gcloud services enable containerregistry.googleapis.com gcloud services
enable container.googleapis.com gcloud services enable
sqladmin.googleapis.com
```

IMPORTANT (DO NOT SKIP):

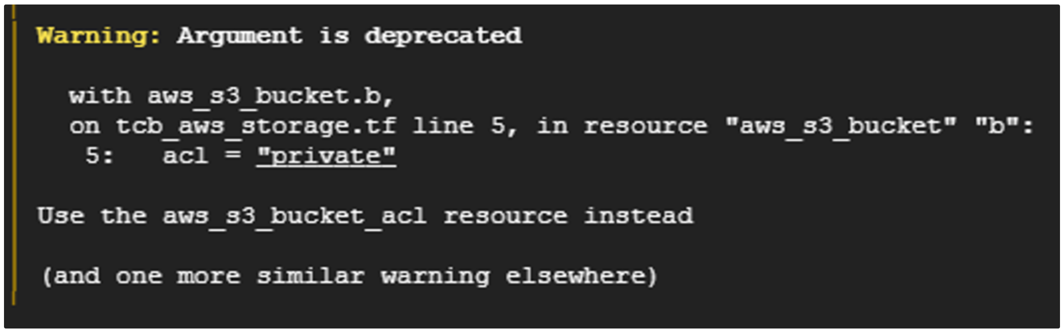
- **Before executing the Terraform commands, open the Google Editor and update the file tcb_aws_storage.tf replacing the bucket name with an unique name (AWS requires unique bucket names).**
    - Open the **tcb_aws_storage.tf** using Google Editor
    - On **line 4** of the file **tcb_aws_storage.tf**:
        - Replace **xxxx** with your name initials, using **5 letters** plus **5 random numbers**:
          Example: **luxxy-covid-testing-system-pdf-en-jerod29292**

- Run the following commands to finish provision infrastructure steps

```
cd ~/mission1_en/mission1/en/terraform/ terraform init terraform plan
terraform apply Type Yes and go ahead.
```

Attention: The Cloud SQL database may take **15 to 25 minutes** to create, always check the **CloudShell** and click **Reconnect** when the session expires (the session expires after **5 minutes** of inactivity by default)

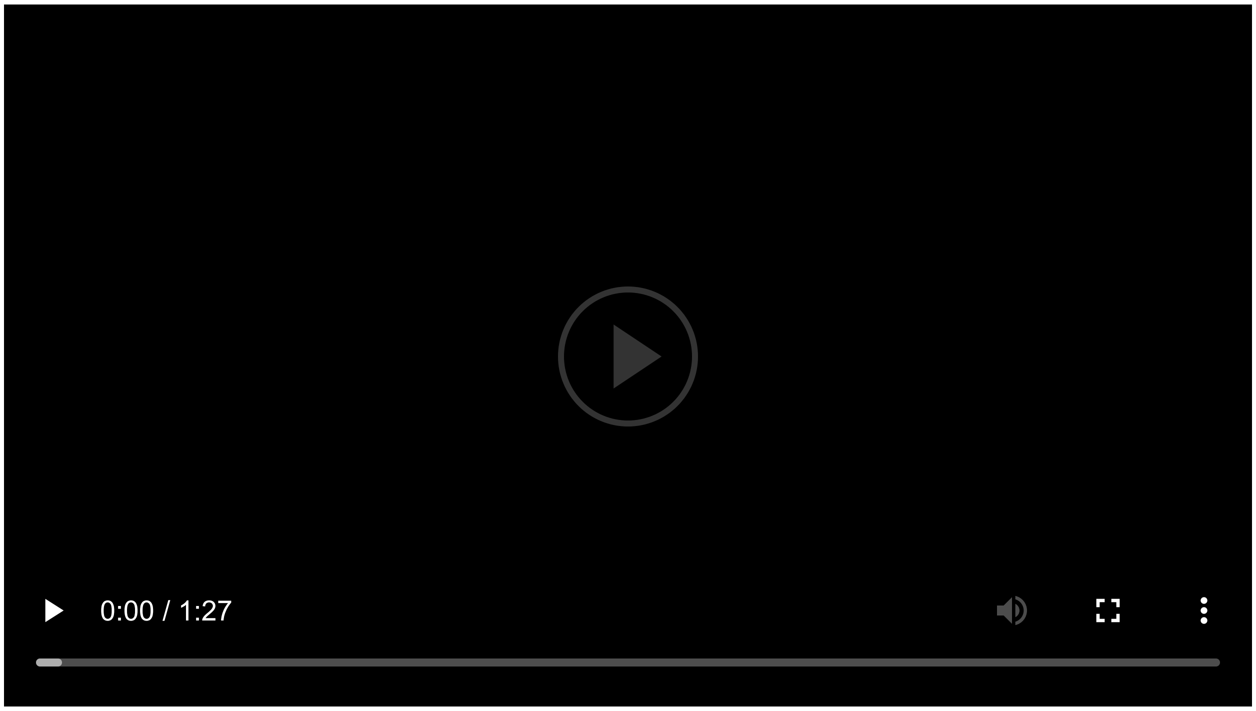
- The **warning** message at the end of **terraform apply** command execution is not a problem, please go ahead:



```
Warning: Argument is deprecated

  with aws_s3_bucket.b,
  on tcb_aws_storage.tf line 5, in resource "aws_s3_bucket" "b":
   5:   acl = "private"

Use the aws_s3_bucket_acl resource instead

(and one more similar warning elsewhere)
```

💡 After finished, access the link to Compare GKE Autopilot and Standard.

SQL Network Configuration



- Once the Cloud SQL instance is provisioned, access the Cloud SQL service
- Click on your Cloud SQL instance.
- On the left side, under Primary Instance, click on **Connections**.
- Go to **Networking** tab.

- Under **Instance IP assignment**, select Private IP to enable.
  - Under **Associated networking**, select "Default"
  - Click **Set up Connection**
  - Click on **Enable API**, to enable Service Networking API (if asked).
  - Select **Use an automatically allocated IP range in your network**.
  - Click **Continue**
  - Click **Create Connection** and **wait a minutes until conclude.** You will see the message: "*Private services access connection for network default has been successfully created.*"