TEXT TO SPEECH

Go to windows PowerShell check if you have python installed using following command

>>> python –version

Check if you have pip3 installed or install it

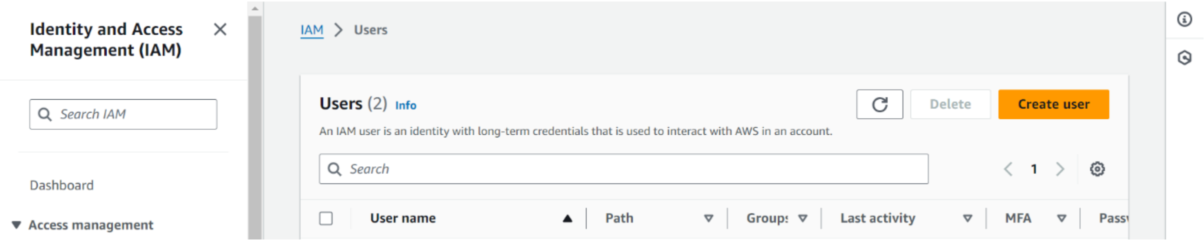
>>> pip3 --version

Next we will install the SDK we need for this project that is Boto3

>>> pip3 install boto3

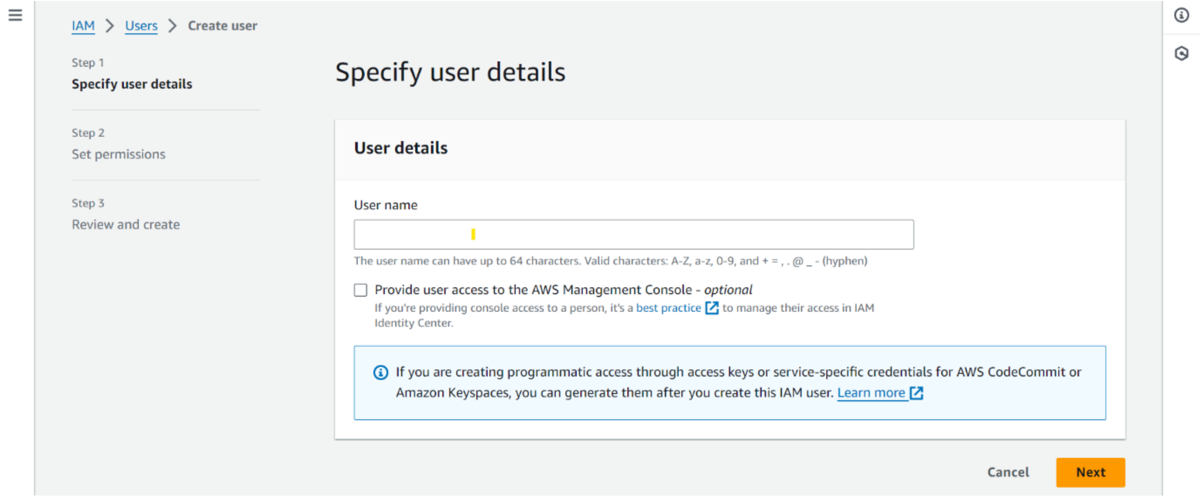
Install the AWS cli for windows or any operating system from the following link - https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

Now we have the python and SDK installed we should configure the cli with AWS to access the functionalities of it. For this here we create a IAM user. Go to the management console of AWS search for IAM and click on create user. We can also provide the root user information.



An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

Name your user and click on next. (Don’t select the option to provide the access to the console as we don’t need to have we will do it using the cli)

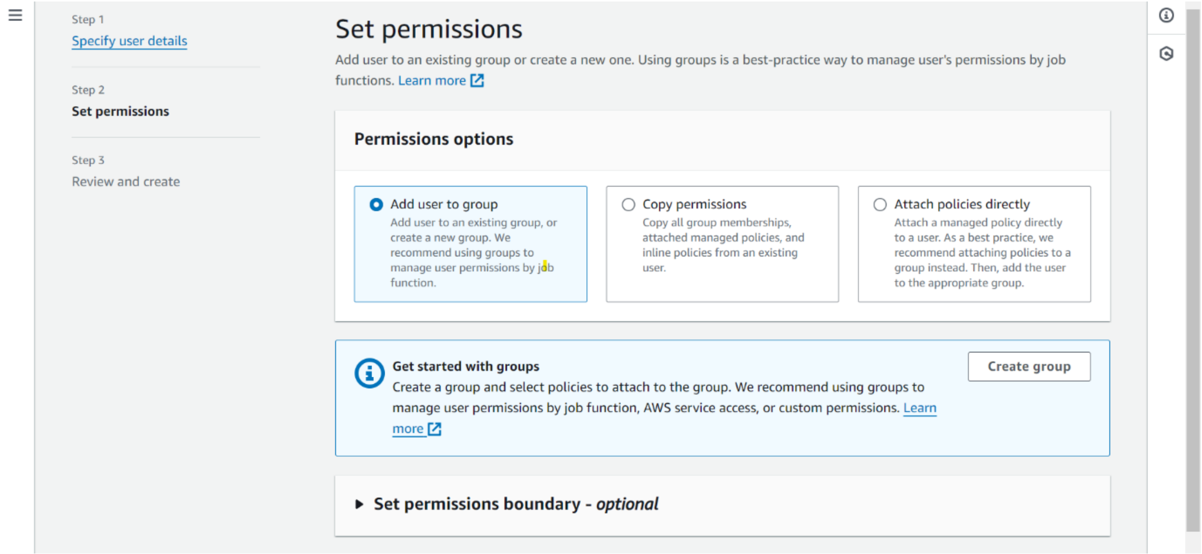


The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ _ - (hyphen)

□ 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

We will set the permissions, we can also add it to an existing group of user. But here we will add the policies directly. Click on 'Attach policies directly'.



Step 1
Specify user details

Step 2
Set permissions

Step 3
Review and create

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.

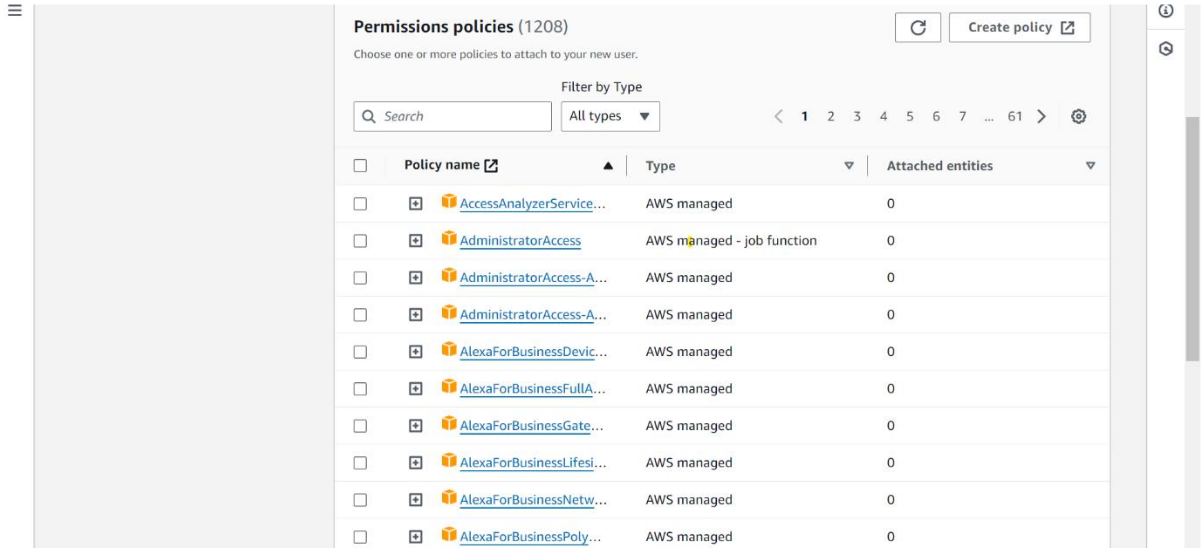
○ Copy permissions
Copy all group memberships, attached managed policies, and inline policies from an existing user.

○ 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.

Get started with groups
Create a group and select policies to attach to the group. We recommend using groups to manage user permissions by job function, AWS service access, or custom permissions. Learn more

▶ **Set permissions boundary - *optional***

A dropdown list will appear from which we can add the policies we want. For this we only need to access the AWS Polly.

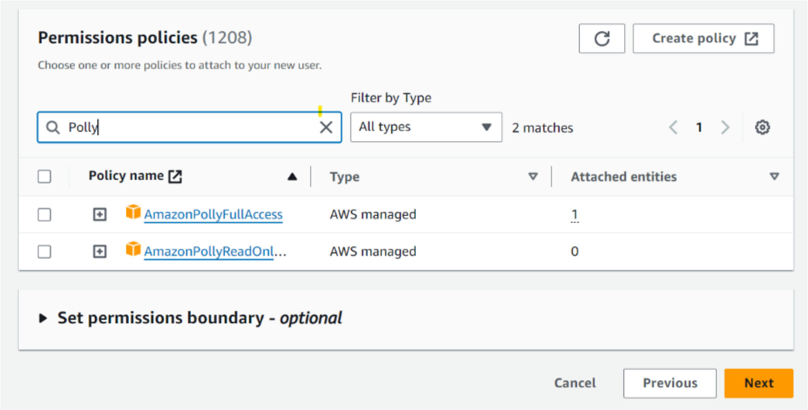


Permissions policies (1208)

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

□	Policy name	Type	Attached entities
□	⊞ AccessAnalyzerService...	AWS managed	0
□	⊞ AdministratorAccess	AWS managed - job function	0
□	⊞ AdministratorAccess-A...	AWS managed	0
□	⊞ AdministratorAccess-A...	AWS managed	0
□	⊞ AlexaForBusinessDevic...	AWS managed	0
□	⊞ AlexaForBusinessFullA...	AWS managed	0
□	⊞ AlexaForBusinessGate...	AWS managed	0
□	⊞ AlexaForBusinessLifesi...	AWS managed	0
□	⊞ AlexaForBusinessNetw...	AWS managed	0
□	⊞ AlexaForBusinessPoly...	AWS managed	0

So search Polly and add the ‘AmazonPollyFullAccess’. Click on next.



Permissions policies (1208)

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

Search: Polly

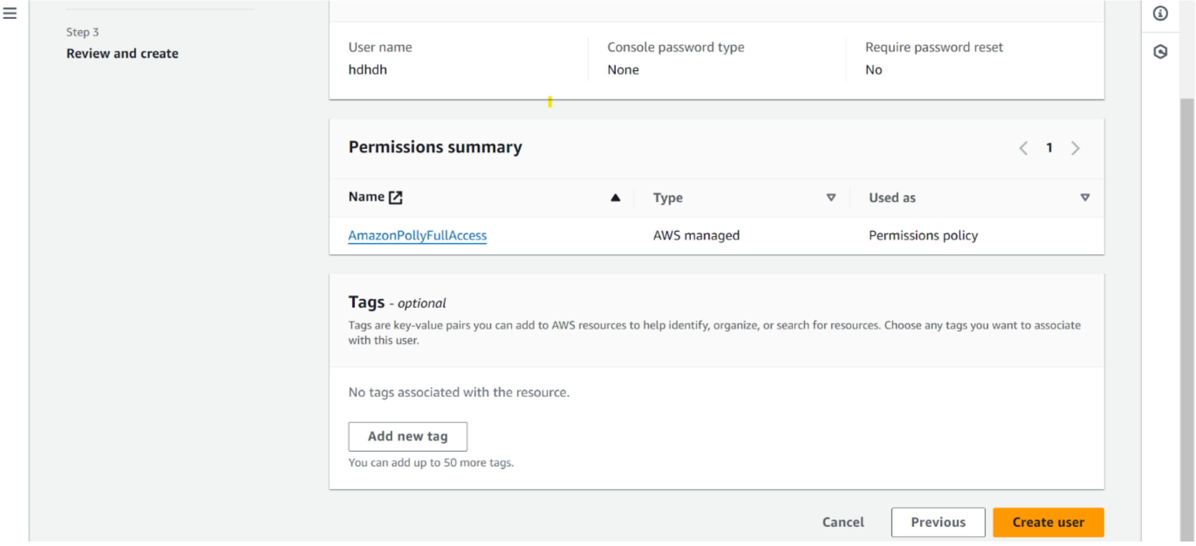
Filter by Type: All types

2 matches

□	Policy name	Type	Attached entities
□	AmazonPollyFullAccess	AWS managed	1
□	AmazonPollyReadOnl...	AWS managed	0

▸ **Set permissions boundary - *optional***

Now review the configurations and create user.



Step 3

Review and create

User name	Console password type	Require password reset
hdhdh	None	No

Permissions summary

Name	Type	Used as
AmazonPollyFullAccess	AWS managed	Permissions policy

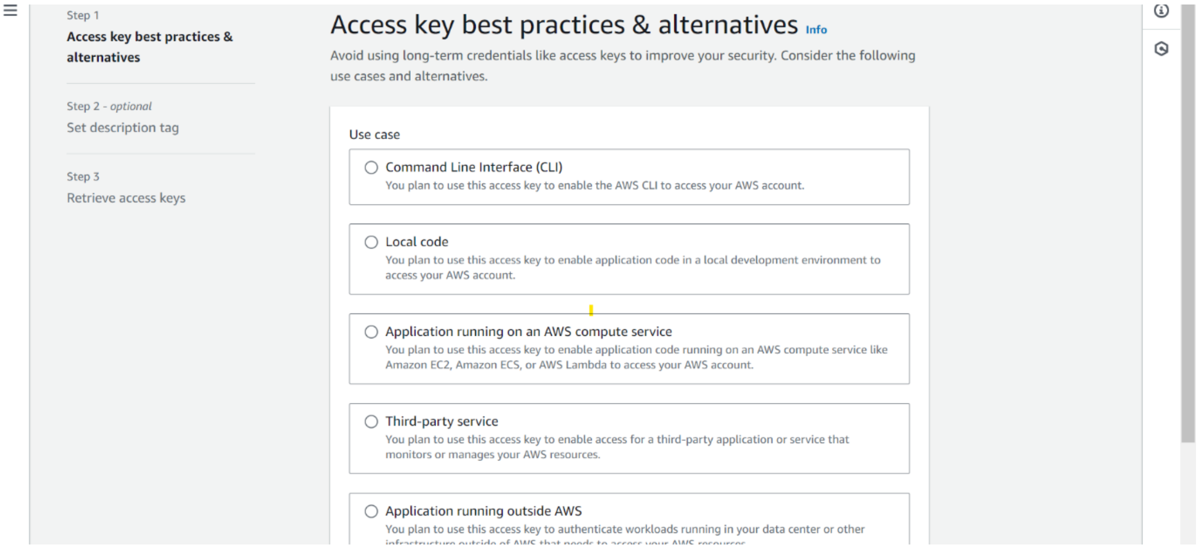
Tags - *optional*

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.

After the user is created click on it and search for ‘Create access key’ option. Following window will appear.



Step 1
Access key best practices & alternatives

Step 2 - *optional*
Set description tag

Step 3
Retrieve access keys

Access key best practices & alternatives Info

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

Use case

○ Command Line Interface (CLI)
You plan to use this access key to enable the AWS CLI to access your AWS account.

○ Local code
You plan to use this access key to enable application code in a local development environment to access your AWS account.

○ Application running on an AWS compute service
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.

○ Third-party service
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.

○ Application running outside AWS
You plan to use this access key to authenticate workloads running in your data center or other

Choose the first option to use for CLI. Click on next. And click on create access key and the access key will be created. Download the csv file and store it safe. Here we will need the ‘Access key’ and the ‘Secret access key’.

Open the PowerShell again and check if the AWS CLI is installed.

```
>>> pip install awscli
```

Now we will configure our AWS IAM user. Type the following

```
>>> aws configure –profile (iam user name)
```

This will create a new profile for our IAM user.

Next it will ask you for the AWS access key id, copy and paste the access key which we created just now. Press Enter.

Next do the same for the AWS secret access key. Press Enter.

Next provide the region name. Press Enter.

Next the output format, write ‘json’. Press Enter.

Now we have completed the configurations. Hence the system would know which AWS account to act upon.

Type the following to see the user in your system on PowerShell

```
>>> cd .aws

>>> dir

>>> type config
```

Here we can see the profile name which we have created and use it in the python code .

Now we will write the code In python for text to speech copy it from the 'my.py' file.

Note: Don't forget to add your profile name and your region in the following line of code

```
>> aws_mag_con = boto3.session.Session(profile_name='my-polly')
   client = aws_mag_con.client(service_name='polly', region_name='ap-south-1')
```

After this run your code enter the text and click on read. Your system media player will open up the text will be read-a-loud.