

RIDDHIMA RAI

500094024

BATCH 2

LAB 2

Terraform AWS provide and IAM user setting

Creating a new IAM user for CLI.

STEP 1: Create a new use by going in services > IAM > create a new user.

The screenshot shows the 'Create user' console in the AWS IAM service. The 'User details' section is active, showing the 'User name' field with the value 'user-cli'. Below this, there is a checkbox for 'Provide user access to the AWS Management Console - optional', which is checked. A message states: 'If you're providing console access to a person, it's a [best practice](#) to manage their access in IAM Identity Center.' Below this is a section titled 'Are you providing console access to a person?' with an information icon. It contains two radio button options: 'Specify a user in Identity Center - Recommended' (which is selected) and 'I want to create an IAM user'. The 'I want to create an IAM user' option has a note: 'We recommend that you create IAM users only if you need to enable programmatic access through access keys, service-specific credentials for AWS CodeCommit or Amazon Keyspaces, or a backup credential for emergency account access.' Below this is the 'Console password' section with two radio button options: 'Autogenerated password' and 'Custom password'. The 'Custom password' option is selected, and there is a text input field for the password. Below the input field, there are two bullet points: 'Must be at least 8 characters long' and 'Must include at least three of the following mix of character types: uppercase letters (A-Z), lowercase letters (a-z), numbers (0-9), and symbols ! @ # \$ % ^ & * () _ + - (hyphen) = [] { } | ' '. There is also a 'Show password' checkbox. Below the password section is a checkbox for 'Users must create a new password at next sign-in - Recommended', which is unchecked. A note states: 'Users automatically get the [IAMUserChangePassword](#) policy to allow them to change their own password.' At the bottom, there is a message with an information icon: '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](#)'.

STEP 2: Give the administration access to the user.

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 (1/1171)

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

Filter by Type: All types

< 1 2 3 4 5 6 7 ... 59 >

	Policy name	Type	Attached entities
<input type="checkbox"/>	AccessAnalyzerServiceRolePolicy	AWS managed	0
<input checked="" type="checkbox"/>	AdministratorAccess	AWS managed - job function	1

STEP 3: Generate the access and security key under the security credentials , select the CLI as use case and give confirmation then click on next.

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 infrastructure outside of AWS that needs to access your AWS resources.

☐ Other
Your use case is not listed here.

Alternatives recommended

- Use [AWS CloudShell](#), a browser-based CLI, to run commands. [Learn more](#)
- Use the [AWS CLI V2](#) and enable authentication through a user in IAM Identity Center. [Learn more](#)

Confirmation



☒ I understand the above recommendation and want to proceed to create an access key.

Cancel **Next**

STEP 4: Save the access and security key under the security credentials which will be used later to connect with terraform.

Retrieve access keys Info

Access key
If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
 AKIA3I3L5ZPMTAXRCNR	 ***** Show

Configuring terraform

STEP 1: Create a new Directory.

 Terraform	17-01-2024 10:11	File folder
---	------------------	-------------

STEP 2: Create terraform configuration file (main.tf)

```
main.tf > provider "aws"
1 terraform {
2     required_providers {
3         aws = {
4             source = "hashicorp/aws"
5             version = "5.32.1"
6         }
7     }
8 }
9
10 provider "aws" {
11     region = "ap-south-1"
12     access_key = "AKIA3I3L5ZPM4X2ZCSFU"
13     secret_key = "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"
14 }
```

STEP 3: Initialize terraform using 'terraform init' command.

```
PS D:\Terraform> terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "5.32.1"...
- Installing hashicorp/aws v5.32.1...
- Installed hashicorp/aws v5.32.1 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```