



---

# SYSTEM PROVISIONING AND CONFIGURATION MANAGEMENT

## LAB FILE

**NAME:** Kartikay Bisaria

**SAP ID:** 500096303

**BATCH:** B3

**SUBMITTED TO:** Dr. Hitesh Kumar Sharma

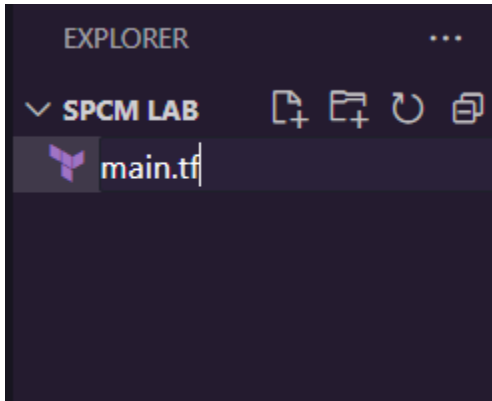
**SEMESTER:** VI

**ENROLLMENT NO.:** R2142211032

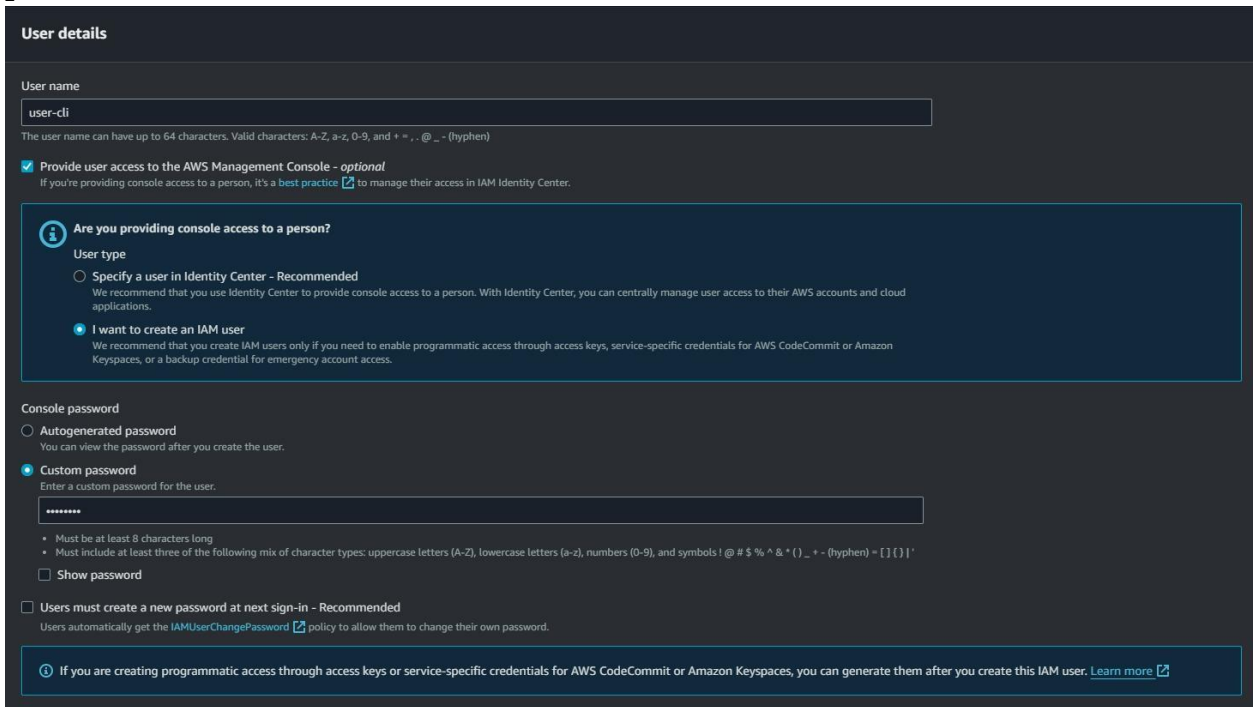
## EXPERIMENT 2:

### Terraform AWS Provider and IAM user Settings

1. Create a new folder for your terraform configuration.
2. Add a file named 'main.tf'.



3. Now make a new IAM account in your AWS console. Also set the custom password.

A screenshot of the AWS IAM console 'User details' page. The page is titled 'User details'. Under 'User name', there is a text input field containing 'user-cli'. Below this, there is a checkbox labeled 'Provide user access to the AWS Management Console - optional' which is checked. A blue box contains the question 'Are you providing console access to a person?' with two options: 'Specify a user in Identity Center - Recommended' (unselected) and 'I want to create an IAM user' (selected). Below this, under 'Console password', there are two options: 'Autogenerated password' (unselected) and 'Custom password' (selected). A text input field for the custom password is shown with masked characters. At the bottom, there is a checkbox for 'Users must create a new password at next sign-in - Recommended' which is unselected. A blue box at the very bottom contains an information icon and text about generating access keys or service-specific credentials.

4. Set appropriate permissions.

## 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

Policy name

Type

Attached entities

<input type="checkbox"/>	AccessAnalyzerServiceRolePolicy	AWS managed	0
<input checked="" type="checkbox"/>	AdministratorAccess	AWS managed - job function	1

- Select create Access Key and note down the access key and secret key.

## 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

- Add the following content.

```
Instance.tf  main.tf  X
E: > Terraform-aws-demo > main.tf > terraform

1  terraform {
2      required_providers {
3          aws = {
4              source = "hashicorp/aws"
5              version = "5.32.1"
6          }
7      }
8  }
9  provider "aws" {
10     region = "ap-south-1"
11     access_key = "AKIAZW6RGWG6KYNVNEET"
12     secret_key = "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"
13 }
```

7. Run 'terraform init' command to initialise the working directory.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS

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.
PS E:\Terraform-aws-demo>
```

