Debugging Terraform IAM User Error

Problem:
When applying the Terraform plan, Terraform attempted to create an IAM User named *TestUser*, but AWS returned an error:

```
Error: creating IAM User (TestUser): operation error IAM: CreateUser, https response error
StatusCode: 409, EntityAlreadyExists: User with name TestUser already exists.
```

Diagnosis:
The IAM user *TestUser* already existed in AWS, but Terraform did not know about it yet because it was not in Terraform's state. As a result, Terraform tried to create a new user with the same name, which caused the *EntityAlreadyExists* error.

Fix:
1. Verified that *TestUser* existed in AWS using the AWS CLI:
```
aws iam get-user --user-name TestUser
```

2. Imported the existing IAM user into Terraform state:
```
terraform import aws_iam_user.test_user TestUser
```

3. Re-ran terraform plan and saw that Terraform now recognized the user and only needed to update it in place.

4. Applied the plan again, which succeeded and brought Terraform state in sync with AWS.

Why This Fix Worked:
- Terraform tracks resources in its *state file*. If a resource exists in AWS but not in the state, Terraform assumes it must create it.
- By importing the IAM user, Terraform learned that *TestUser* already exists, so instead of creating it, it switched to managing it.
- After the import, Terraform only made necessary updates (like setting *force_destroy = true*) and added the group membership.
- Final terraform plan showed **No changes**, confirming AWS and Terraform state were in sync.

Lessons Learned:
- Always check if a resource already exists in AWS before creating it with Terraform.
- Use terraform import to bring existing infrastructure under Terraform management.
- Re-run terraform plan after importing to confirm the state matches reality.
- When Terraform says "No changes," it means your infrastructure is fully in sync with your code.