state drift with the AWS provider in Terraform:

Let's say you have a Terraform configuration that provisions an AWS EC2 instance with the following code:

```hcl
provider "aws" {
  region = "us-west-2"
}

resource "aws_instance" "example" {
  ami           = "ami-0c94855ba95c71c99"
  instance_type = "t2.micro"
  tags = {
    Name = "example-instance"
  }
}
```

After running `terraform apply`, Terraform provisions the EC2 instance and creates a state file to track the resources it manages. The state file stores important information about the resources, such as their IDs, attributes, and dependencies.

Now, let's say someone manually modifies the EC2 instance outside of Terraform. For example, they change the instance type from `t2.micro` to `t3.micro` directly in the AWS Management Console.

If you run `terraform plan` again without applying the changes, Terraform will detect a state drift. The plan will show that there is a discrepancy between the expected state (as defined in your Terraform code) and the actual state of the EC2 instance.

```
# aws_instance.example will be updated in-place
~ resource "aws_instance" "example" {
```

```
    id           = "i-0a7e8b9d7c41592e6"
  ~ instance_type  = "t2.micro" -> "t3.micro"
    tags         = {
      "Name" = "example-instance"
    }
}
```

In this case, the computed value of the instance type has changed externally, and Terraform detects it as a drift. Terraform recognizes that the `instance_type` attribute in the state file doesn't match the expected value defined in your code.

To resolve this state drift, you have a few options:

1. **Apply Changes**: If the changes made outside of Terraform are intentional and align with your desired state, you can run `terraform apply` to update the state file and bring it back in sync with the actual infrastructure.

2. **Discard Changes**: If the changes made outside of Terraform were not intentional or should not be applied, you can discard the changes by resetting the state or importing the resource back into Terraform. This will overwrite the state file with the correct values from your Terraform code.

It's essential to regularly review and update your Terraform configuration to account for any changes made outside of Terraform. By doing so, you can ensure that the expected state defined in your code aligns with the actual state of your AWS resources and avoid state drift.