To understand the concepts of desired state and current state in the context of an EC2 instance in Terraform, let's consider an example.

Suppose you have defined an EC2 instance resource in your Terraform configuration file (`main.tf`):

```hcl
resource "aws_instance" "example" {
  ami           = "ami-0c94855ba95c71c99"
  instance_type = "t2.micro"
  subnet_id     = "subnet-12345678"
  key_name      = "my-key-pair"

  tags = {
    Name        = "ExampleInstance"
    Environment = "Production"
  }
}
```

In this example, you have specified the attributes of the EC2 instance, such as the Amazon Machine Image (AMI), instance type, subnet ID, and key pair. The configuration represents the desired state of the EC2 instance.

When you run `terraform apply` to apply this configuration, Terraform will compare the desired state (specified in your configuration) with the current state of the infrastructure.

- Desired State: The desired state represents the configuration you have specified in your Terraform code. It defines how you want your infrastructure to be provisioned and managed. In our example, the desired state includes attributes like the AMI ID, instance type, subnet ID, and key pair.

- Current State: The current state represents the actual state of the infrastructure resources as recorded by Terraform. It is stored in the Terraform state file (`terraform.tfstate`). Initially, when you run `terraform apply` for the first time, the current state will match the desired state because no

resources have been provisioned yet. However, as you make changes to the infrastructure or update the configuration, the current state may differ from the desired state.

For example, let's say you ran `terraform apply` and the EC2 instance was successfully created. Now, the current state reflects the provisioned resources, including the actual values of attributes like the instance ID, IP address, and other details.

If you make changes to the configuration, such as modifying the instance type or AMI, and run `terraform apply` again, Terraform will compare the desired state (based on the updated configuration) with the current state. It will determine the necessary actions to bring the infrastructure to the desired state, such as updating the instance type or creating a new instance with the updated AMI. Terraform will make the required changes to match the desired state.

The state file (`terraform.tfstate`) is crucial for Terraform to keep track of the current state of resources and manage them accordingly. It allows Terraform to know what actions to take when you run `terraform apply`, `terraform destroy`, or other Terraform commands.

Overall, the desired state represents your configuration, while the current state represents the actual state of the provisioned resources. Terraform uses the current state as a reference to apply changes and maintain the desired state of your infrastructure.