Lab: Execute changes to infrastructure with Terraform `terraform apply`

The `terraform apply` command executes the actions proposed in a Terraform plan.

- Task 1: Apply a Terraform Plan
- Task 2: Auto Approve Execution of an Apply
- Task 3: Execute a saved Terraform Plan

Task 1: Apply a Terraform Plan

The most straightforward way to use terraform apply is to run it without any arguments at all, in which case it will automatically create a new execution plan (as if you had run terraform plan) and then prompt you to approve that plan, before taking the indicated actions.

Make an update the `Environment` tag of the `vpc` resource

```
resource "aws_vpc" "vpc" {

  cidr_block = var.vpc_cidr

  tags = {
    Name        = var.vpc_name
    Environment = "stage"
    Terraform   = "true"
  }
}
```

```
terraform apply
```

Review the new execution plan and approve it.

```
Terraform will perform the following actions:
  # aws_vpc.vpc will be updated in-place
  ~ resource "aws_vpc" "vpc" {
        id                               = "vpc-08b492b03641cb916"
      ~ tags                             = {
          ~ "Environment" = "demo_environment" -> "stage"
            # (2 unchanged elements hidden)
        }
      ~ tags_all                         = {
          ~ "Environment" = "demo_environment" -> "stage"
            # (4 unchanged elements hidden)
        }
        # (14 unchanged attributes hidden)
    }

Plan: 0 to add, 1 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_vpc.vpc: Modifying... [id=vpc-08b492b03641cb916]
aws_vpc.vpc: Modifications complete after 1s [id=vpc-08b492b03641cb916]

Apply complete! Resources: 0 added, 1 changed, 0 destroyed.
```

Task 2: Auto Approve Execution of an Apply

Make another update the `Environment` tag of the `vpc` resource

```
resource "aws_vpc" "vpc" {

  cidr_block = var.vpc_cidr

  tags = {
    Name        = var.vpc_name
    Environment = "QA"
    Terraform   = "true"
  }
}
```

```
terraform apply -auto-approve
```

```
Terraform will perform the following actions:

  # aws_vpc.vpc will be updated in-place
  ~ resource "aws_vpc" "vpc" {
        id                               = "vpc-08b492b03641cb916"
      ~ tags                             = {
          ~ "Environment" = "stage" -> "QA"
            # (2 unchanged elements hidden)
        }
      ~ tags_all                         = {
          ~ "Environment" = "stage" -> "QA"
```

```
            # (4 unchanged elements hidden)
        }
        # (14 unchanged attributes hidden)
    }

Plan: 0 to add, 1 to change, 0 to destroy.
aws_vpc.vpc: Modifying... [id=vpc-08b492b03641cb916]
aws_vpc.vpc: Modifications complete after 1s [id=vpc-08b492b03641cb916]

Apply complete! Resources: 0 added, 1 changed, 0 destroyed.
```

Note that there is no prompt for approval of the plan with this execution. This is ideal for automated pipelines and workflows but should be used with caution.

Task 3: Execute a saved Terraform Plan

Make another update the `Environment` tag of the `vpc` resource

```
resource "aws_vpc" "vpc" {

  cidr_block = var.vpc_cidr

  tags = {
    Name        = var.vpc_name
    Environment = "test-dev"
    Terraform   = "true"
  }
}
```

```
terraform plan -out=myplan
```

```
terraform apply myplan
```

```
aws_vpc.vpc: Modifying... [id=vpc-08b492b03641cb916]
aws_vpc.vpc: Modifications complete after 2s [id=vpc-08b492b03641cb916]

Apply complete! Resources: 0 added, 1 changed, 0 destroyed.
```