Modules

- Terraform provides modules which allow us to abstract away re-usable parts, which we can configure once, and use everywhere
- Modules are self-contained packages which can be shared across teams for different projects.
- Every directory is a Terraform Module
- Using modules in terraform is similar to using resources except we use module clause for modules instead of resource clause.

```
module "moduleName" {
  source  = "module/path"
}
```

- Source parameter is a required field for modules. This specifies from where to download module configuration. We can download modules from multiple resources i.e. local path, terraform registry, GitHub, HTTP URLs, s3 etc
- There is a public terraform registry which contains many battle tested modules which can be used by anyone for faster development.

> https://registry.terraform.io/

Local

Example Creating all network resources like VPC, Subnet etc

- Creating a module

```
resource "aws_vpc" "vpc" {
  cidr_block = "${var.cidr_vpc}"
  enable_dns_support   = true
  enable_dns_hostnames = true
  tags {
    "Environment" = "${var.environment_tag}"
  }
}

resource "aws_internet_gateway" "gateway" {
  vpc_id = "${aws_vpc.vpc.id}"
  tags {
    "Environment" = "${var.environment_tag}"
  }
}

resource "aws_subnet" "subnet" {
  vpc_id = "${aws_vpc.vpc.id}"
  cidr_block = "${var.cidr_subnet}"
  map_public_ip_on_launch = "true"
  availability_zone = "${var.availability_zone}"
  tags {
    "Environment" = "${var.environment_tag}"
  }
```

```
}

resource "aws_route_table" "table" {
  vpc_id = "${aws_vpc.vpc.id}"

  route {
      cidr_block = "0.0.0.0/0"
      gateway_id = "${aws_internet_gateway.igw.id}"
  }

  tags {
    "Environment" = "${var.environment_tag}"
  }
}

resource "aws_route_table_association" "table_assoc" {
  subnet_id      = "${aws_subnet.subnet_public.id}"
  route_table_id = "${aws_route_table.rtb_public.id}"
}

resource "aws_security_group" "sg_22" {
  name = "sg_22"
  vpc_id = "${aws_vpc.vpc.id}"

  # SSH access from the VPC
  ingress {
      from_port   = 22
      to_port     = 22
      protocol    = "tcp"
      cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags {
    "Environment" = "${var.environment_tag}"
  }
}
```

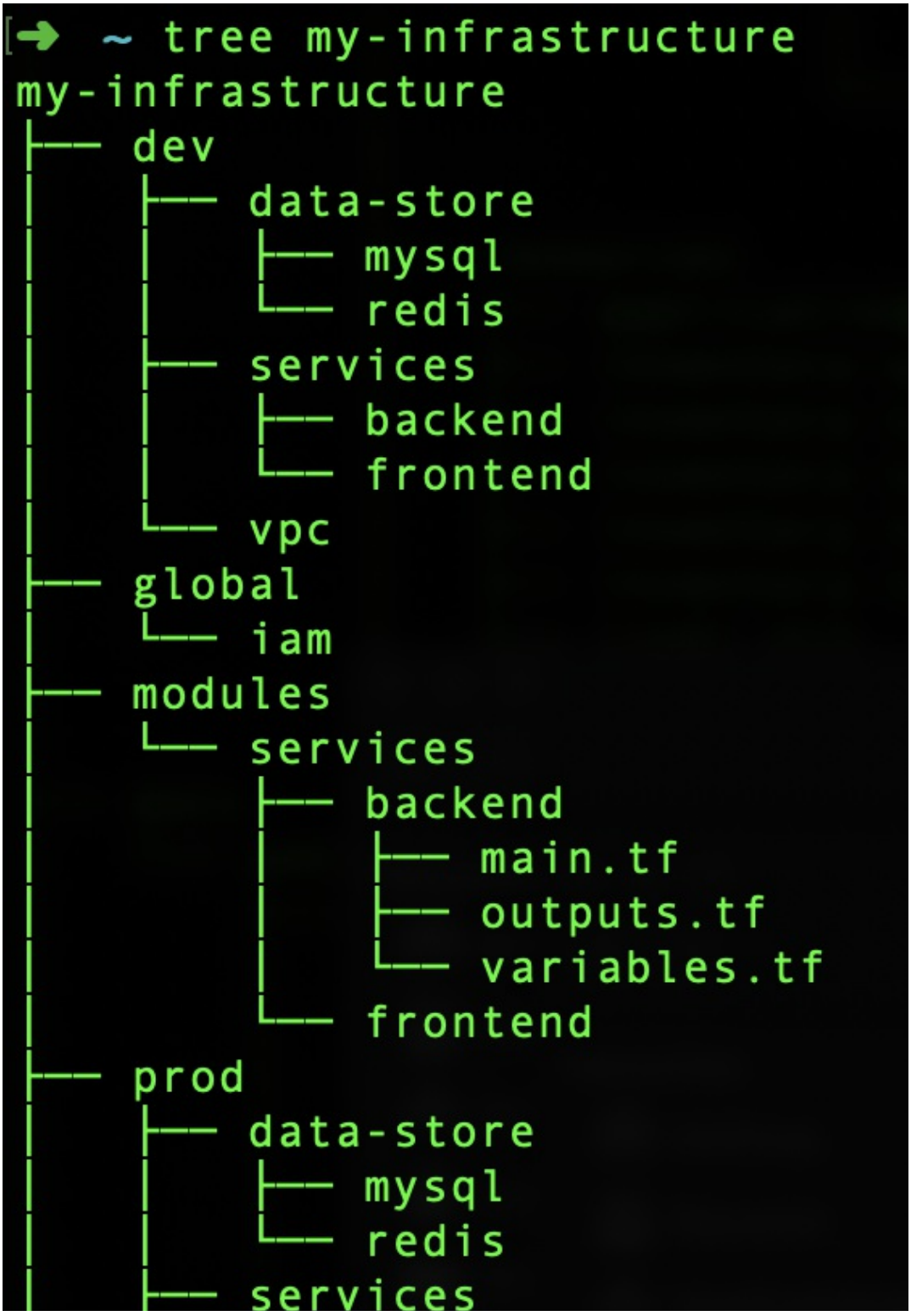
- Using the Module in Dev/Test/Prod setups

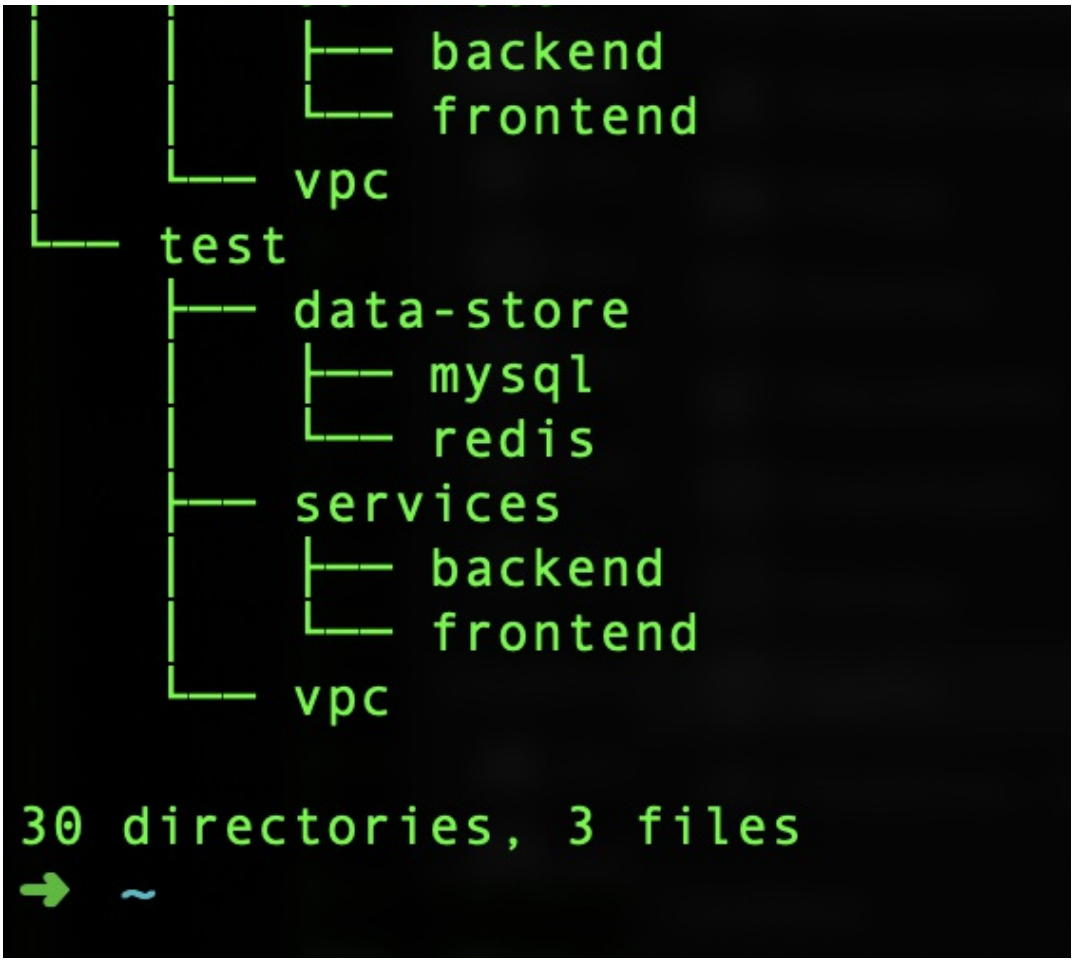
```
module "networkModule" {
  source = "./module/network"
}


resource "aws_instance" "testInstance" {
  ami           = "${var.instance_ami}"
  instance_type = "${var.instance_type}"
  subnet_id = "${module.networkModule.public_subnet_id}"
  vpc_security_group_ids = ["${module.networkModule.sg_22_id}"]
  tags {
    "Environment" = "${var.environment_tag}"
 }
}
```

Hands-on:

> Let's create our own module and use it Refer to 05-Modules/01-Local-Module Run terraform get once you have made any changes to the module

- Key takeaways:
  - Terraform initialises the module as well during terraform init
  - Where should the provider definition go?
  - Note the output.tf of the root module
- Now, this is how our directory structure should look like
- 



External

- Let's browse the Terraform Registry.
- We'll use the following module for HandsOn. https://registry.terraform.io/modules/terraform-aws-modules/security-group/aws/2.16.0

Hands-on:

> Let's use an existing module from registry Refer to 05-Modules/02-Module-Registry

- Key takeaways:
  - Terraform Registry provides public modules for usage which has been tested extensively.
  - It has different versions specific to a corresponding Terraform version
  - The source code is open source and available for fork - https://github.com/terraform-aws-modules/terraform-aws-security-group/tree/v2.16.0
  - The documentation specifies all input and output properties.