More Examples

Deploy Dependent resources

Hands-on:

> Let's use terraform to create an EC2 Machine and security group Refer to 01-Getting-Started/03-Dependent-Resources

- Key takeaways:
  - Look at the generated dependency graph
  - Order of resources in main.tf file has no impact

> Let's ssh into the ec2 instance

```
ssh -i /path/my-key-pair.pem ec2-user@13.232.199.95
```

> If you get this error - "Permissions 0666 for '/Users/kapilm/salesforce-keypair.pem' are too open.". Run following command:

```
 chmod 400 ~/salesforce-keypair.pem
```

> Let's edit the main.tf file to change port to 8080. Run terraform apply again Try ssh again

Using Data Sources

- *Data sources* allow data to be fetched or computed for use elsewhere in Terraform configuration.
- *Data sources* differs based on the provider you are using.
- Local only *Data Sources* are used for reading files that are locally stored

Hands-on:

> Let's use data source to look up AMI for the latest Ubuntu machine Refer to 01-Getting-Started/04-Using-Data-Source

- Key takeaways:
  - You can look up AWS (provider) resources using data source
  - The data source gathers the data before generating a plan
  - Similarly, you can look up other resources like Avalability Zones etc using the Data Source

How to use Loops

- Let's say we want to create 3 IAM users using Terraform. How'd we do that?
- We can write below code thrice in `main.tf`

```
resource "aws_iam_user" "example" {
    name = "Dravid"
}
```

- A better approach is to use `count`

```
resource "aws_iam_user" "example" {
    count = 3
    name  = "Dravid"
}
```

- This will create 3 users but all having same name.
- How about we use `index`

```
resource "aws_iam_user" "example" {
    count = 3
    name  = "Dravid.${count.index}"
}
```

- This will give us 3 users suffixed with 0, 1 and 2.
- The better approach will be using a List variable

```
variable "user_names" {
    description = "Create IAM users with these names"
    type        = "list"
    default     = ["Dhoni", "Sachin", "Dravid"]
}

resource "aws_iam_user" "example" {
    count = "${length(var.user_names)}"
    name  = "${element(var.user_names, count.index)}"
}
```

- How to find out the output vars of resources created using `count` ?

```
output "Dhoni_arn" {
    value = "${aws_iam_user.example.0.arn}"
}

output "all_arns" {
    value = ["${aws_iam_user.example.*.arn}"]
}
```

How to use Conditions (If/Else)

- Example - Create alert only when it's Prod env.

```
variable "env_name" {
    description = "Environment for which the resources are being created"
```

```
}

resource "aws_cloudwatch_metric_alarm" "example" {
    count = "${var.env_name == 'prod'" ? 1 : 0}"

    alarm_name = "cpu alert"
    ...
}

...
```

- If count is 0, then the resource will not be created.
- Another example to show if/else - Give Full Access if needed, else give read-only access

```
variable "give_dravid_full_ec2_access" {
    description = "If true, Dravid gets full EC2 access"
}

resource "aws_iam_user_policy_attachment" "dravid_access" {
    count = "${var.give_dravid_full_ec2_access}"

    user = "Dravid"
    policy_arn = "${aws_iam_policy.ec2_full_access.arn}"
}

resource "aws_iam_user_policy_attachment" "dravid_access" {
    count = "${1 - var.give_dravid_full_ec2_access}"

    user = "Dravid"
    policy_arn = "${aws_iam_policy.ec2_read_only_access.arn}"
}
```

- Interpolation syntax allows basic maths calculations
- `count` is a meta parameter.
- There is another useful meta parameter called `lifecycle`

Hands-on:

> Let's see how the lifecycle meta parameter is useful Refer to 01-Getting-Started/05-Using-Lifecycle
> Let's first create the resources - ec2 machine and security group without lifecycle meta parameter Now, let's change the ami and see how Terraform behaves. Let's uncomment the lifecycle block to understand the behaviour and change the ami again

- Key takeaways:
  - The dependent resources also need to have the same lifecycle meta parameter
  - This flag is used to ensure the replacement of a resource is created before the original instance is destroyed.
  - This will help in avoiding downtime
  - Another use case can be modifying DNS names