Terraform Manage Resource Lifecycle

create_before_destroy:

If this meta-argument is not used, by default the existing resources are deleted first and then new resources are created.

```
ubuntu@ip-172-31-85-196: ~/ec2

resource "aws_instance" "demo" {
 count = 2
 provider = aws.aws_east
 ami = var.ami
 instance_type = var.type


 tags = {
   Name = "Demo System"
 }

}
~
~
~
~
~
~
~
~
~
-- INSERT --                                          14,1          All
```

On terraform apply, instances are created. Now we will update the key, so that two new instances are replaced.

```
ubuntu@ip-172-31-85-196: ~/ec2

resource "aws_instance" "demo" {
 count = 2
 provider = aws.aws_east
 ami = var.ami
 instance_type = var.type
 key_name = "23mar"

 tags = {
   Name = "Demo System"
 }

}

~
~
~
~
~
~
~
~
~
"main.tf" 14L, 175C                                    7,9           All
```

On terraform apply the older instances are first deleted and then newer are created

```
ubuntu@ip-172-31-85-196: ~/ec2

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_instance.demo[1]: Destroying... [id=i-0937e1022723a4525]
aws_instance.demo[0]: Destroying... [id=i-085d683983cccacd6]
aws_instance.demo[1]: Still destroying... [id=i-0937e1022723a4525, 10s elapsed]
aws_instance.demo[0]: Still destroying... [id=i-085d683983cccacd6, 10s elapsed]
aws_instance.demo[1]: Still destroying... [id=i-0937e1022723a4525, 20s elapsed]
aws_instance.demo[0]: Still destroying... [id=i-085d683983cccacd6, 20s elapsed]
aws_instance.demo[1]: Still destroying... [id=i-0937e1022723a4525, 30s elapsed]
aws_instance.demo[0]: Still destroying... [id=i-085d683983cccacd6, 30s elapsed]
aws_instance.demo[1]: Destruction complete after 39s
aws_instance.demo[1]: Creating...
aws_instance.demo[0]: Still destroying... [id=i-085d683983cccacd6, 40s elapsed]
aws_instance.demo[1]: Still creating... [10s elapsed]
aws_instance.demo[0]: Destruction complete after 50s
aws_instance.demo[0]: Creating...
aws_instance.demo[1]: Still creating... [20s elapsed]
aws_instance.demo[0]: Still creating... [10s elapsed]
```

Now create_before_destroy is added

```
ubuntu@ip-172-31-85-196: ~/ec2

resource "aws_instance" "demo" {
 count = 2
 provider = aws.aws_east
 ami = var.ami
 instance_type = var.type
 key_name = "second"

 tags = {
   Name = "Demo System"
 }
       lifecycle {
      create_before_destroy = true
   }
}

~
~
~
~
~
~
~
"main.tf" 16L, 234C                                    8,1           All
```

In this case, new resources are created first, and then older are deleted

```
ubuntu@ip-172-31-85-196: ~/ec2

aws_instance.demo[0]: Creating...
aws_instance.demo[1]: Creating...
aws_instance.demo[0]: Still creating... [10s elapsed]
aws_instance.demo[1]: Still creating... [10s elapsed]
aws_instance.demo[0]: Still creating... [20s elapsed]
aws_instance.demo[1]: Still creating... [20s elapsed]
aws_instance.demo[0]: Still creating... [30s elapsed]
aws_instance.demo[1]: Still creating... [30s elapsed]
aws_instance.demo[0]: Still creating... [40s elapsed]
aws_instance.demo[1]: Still creating... [40s elapsed]
aws_instance.demo[0]: Creation complete after 42s [id=i-0948aa3ec098b6f55]
aws_instance.demo[0] (deposed object 2c282a89): Destroying... [id=i-0ccf7fe6fcd5
e7b32]
aws_instance.demo[1]: Creation complete after 42s [id=i-0b2fcb7a37252605c]
aws_instance.demo[1] (deposed object 68201162): Destroying... [id=i-04b402a6a855
3fcda]
aws_instance.demo[0]: Still destroying... [id=i-0ccf7fe6fcd5e7b32, 10s elapsed]
aws_instance.demo[1]: Still destroying... [id=i-04b402a6a8553fcda, 10s elapsed]
aws_instance.demo[0]: Still destroying... [id=i-0ccf7fe6fcd5e7b32, 20s elapsed]
aws_instance.demo[1]: Still destroying... [id=i-04b402a6a8553fcda, 20s elapsed]
aws_instance.demo[0]: Still destroying... [id=i-0ccf7fe6fcd5e7b32, 30s elapsed]
aws_instance.demo[1]: Still destroying... [id=i-04b402a6a8553fcda, 30s elapsed]
```

Ignore_changes:

Here the name tag of ec2 is changed

```
ubuntu@ip-172-31-85-196: ~/ec2

resource "aws_instance" "demo" {
 count = 2
 provider = aws.aws_east
 ami = var.ami
 instance_type = var.type
 key_name = "second"

 tags = {
   Name = "Demo"
 }
       lifecycle {
      create_before_destroy = true
   }
}

~
~
~
~
~
~
~
-- INSERT --                                          11,3          All
```

When we run terraform plan it shows 2 need to be changed

```
ubuntu@ip-172-31-85-196: ~/ec2
  ~ resource "aws_instance" "demo" {
        id                                   = "i-0b2fcb7a37252605c"
      ~ tags                                 = {
          ~ "Name" = "Demo System" -> "Demo"
        }
      ~ tags_all                             = {
          ~ "Name" = "Demo System" -> "Demo"
        }
        # (28 unchanged attributes hidden)





        # (5 unchanged blocks hidden)
    }

Plan: 0 to add, 2 to change, 0 to destroy.

------------------------------------------------------------------------

Note: You didn't use the -out option to save this plan, so Terraform can't
guarantee to take exactly these actions if you run "terraform apply" now.
ubuntu@ip-172-31-85-196:~/ec2$
```

Now we add ignore_changes

```
ubuntu@ip-172-31-85-196: ~/ec2

resource "aws_instance" "demo" {
 count = 2
 provider = aws.aws_east
 ami = var.ami
 instance_type = var.type
 key_name = "second"

 tags = {
   Name = "Demo"
 }
       lifecycle {
      create_before_destroy = true
          ignore_changes = [
      tags,
    ]
   }
}

~
~
~
~
-- INSERT --                                                18,2          All
```

And now if we do terraform apply the change in name tag is ignored

```
ubuntu@ip-172-31-85-196: ~/ec2
ubuntu@ip-172-31-85-196:~/ec2$ terraform apply
aws_instance.demo[1]: Refreshing state... [id=i-0b2fcb7a37252605c]
aws_instance.demo[0]: Refreshing state... [id=i-0948aa3ec098b6f55]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and
found no differences, so no changes are needed.

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

Outputs:

instance_id = [
  "i-0948aa3ec098b6f55",
  "i-0b2fcb7a37252605c",
]
ubuntu@ip-172-31-85-196:~/ec2$
```

Prevent_destroy:

We have added a different subnet for ec2

```
ubuntu@ip-172-31-85-196: ~/ec2

resource "aws_instance" "demo" {
 count = 2
 provider = aws.aws_east
 ami = var.ami
 instance_type = var.type
 key_name = "second"
 subnet_id = "subnet-01f714c965da21c97"

 tags = {
   Name = "Demo"
 }
       lifecycle {
      create_before_destroy = true
          ignore_changes = [
      tags,
    ]
   }
}

~
~
~
"main.tf" 20L, 314C                                    8,39          All
```

Now on terraform apply, previous resources will get deleted and new will be added

```
ubuntu@ip-172-31-85-196: ~/ec2
          ~ tags                  = {} -> (known after apply)
          ~ throughput            = 0 -> (known after apply)
          ~ volume_id             = "vol-0ef4e75b438035c66" -> (known after appl
y)
          ~ volume_size           = 8 -> (known after apply)
          ~ volume_type           = "gp2" -> (known after apply)
        }
    }

Plan: 2 to add, 0 to change, 2 to destroy.

Changes to Outputs:
  ~ instance_id = [
      - "i-0948aa3ec098b6f55",
      - "i-0b2fcb7a37252605c",
      + (known after apply),
      + (known after apply),
    ]

------------------------------------------------------------------------

Note: You didn't use the -out option to save this plan, so Terraform can't
guarantee to take exactly these actions if you run "terraform apply" now.
ubuntu@ip-172-31-85-196:~/ec2$
```

Prevent_destroy is added to lifecycle

```
ubuntu@ip-172-31-85-196: ~/ec2

resource "aws_instance" "demo" {
 count = 2
 provider = aws.aws_east
 ami = var.ami
 instance_type = var.type
 key_name = "second"
 subnet_id = "subnet-01f714c965da21c97"

 tags = {
   Name = "Demo"
 }
       lifecycle {
      create_before_destroy = true
      prevent_destroy = true
          ignore_changes = [
      tags,
    ]
   }
}

~
~
:wq
```

Now terraform apply will throw error because we have added prevent_destroy argument

```
ubuntu@ip-172-31-85-196: ~/ec2
aws_instance.demo[0]: Refreshing state... [id=i-0948aa3ec098b6f55]

| Error: Instance cannot be destroyed
|
|   on main.tf line 2:
|    2: resource "aws instance" "demo" {
|
| Resource aws_instance.demo[1] has lifecycle.prevent_destroy set, but the plan
| calls for this resource to be destroyed. To avoid this error and continue
| with the plan, either disable lifecycle.prevent_destroy or reduce the scope
| of the plan using the -target flag.


| Error: Instance cannot be destroyed
|
|   on main.tf line 2:
|    2: resource "aws instance" "demo" {
|
| Resource aws_instance.demo[0] has lifecycle.prevent_destroy set, but the plan
| calls for this resource to be destroyed. To avoid this error and continue
| with the plan, either disable lifecycle.prevent_destroy or reduce the scope
| of the plan using the -target flag.

ubuntu@ip-172-31-85-196:~/ec2$
```