Lab: Terraform Taint and Replace

The terraform `taint` command allows for you to manually mark a resource for recreation. The `taint` command informs Terraform that a particular resource needs to be rebuilt even if there has been no configuration change for that resource. Terraform represents this by marking the resource as "tainted" in the Terraform state, in which case Terraform will propose to replace it in the next plan you create.

- Task 1: Manually mark a resource to be rebuilt by using the terraform `taint` command.
- Task 2: Observe a `remote-exec` provisoner failing, resulting in Terraform automatically tainting a resource.
- Task 3: Untaint a resource
- Task 4: Use the `-replace` option rather than `taint`

Task 1: Manually mark a resource to be rebuilt by using the terraform `taint` command

There are some situations where we want to recreate a resource without modifying any terraform configuration. An example of this might be to rebuild a server and force it to rerun it's bootsrap process. This can be accomplished by manually "tainting" the resoruce.

Step 1.1 - Create a new Web Server

Let's add a new webserver to our configuration that we can work on. Update your `main.tf` to include the new webserver.

```
# Terraform Resource Block - To Build Web Server in Public Subnet
resource "aws_instance" "web_server" {
  ami                         = data.aws_ami.ubuntu.id
  instance_type               = "t2.micro"
  subnet_id                   = aws_subnet.public_subnets["public_subnet_1
     "].id
  security_groups             = [aws_security_group.vpc-ping.id,
     aws_security_group.ingress-ssh.id, aws_security_group.vpc-web.id]
  associate_public_ip_address = true
  key_name                    = aws_key_pair.generated.key_name
  connection {
    user        = "ubuntu"
    private_key = tls_private_key.generated.private_key_pem
    host        = self.public_ip
  }
```


```
  # Leave the first part of the block unchanged and create our `local-exec
    ` provisioner
  provisioner "local-exec" {
    command = "chmod 600 ${local_file.private_key_pem.filename}"
  }

  provisioner "remote-exec" {
    inline = [
      "sudo rm -rf /tmp",
      "sudo git clone https://github.com/hashicorp/demo-terraform-101 /tmp
        ",
      "sudo sh /tmp/assets/setup-web.sh",
    ]
  }

  tags = {
    Name = "Web EC2 Server"
  }

  lifecycle {
    ignore_changes = [security_groups]
  }

}
```

Execute a `plan` and `apply` to build out the web server.

```
terraform validate
terraform plan
terraform apply
```

Step 1.2 - Reference the Web Server

Now that that server has been built, let's validate it's resource ID. If you're unsure of the correct ID for the resource you want to taint, you can use `terraform state list`:

```
terraform state list
```

```
data.aws_ami.ubuntu
data.aws_ami.ubuntu_16_04
data.aws_ami.windows_2019
data.aws_availability_zones.available
data.aws_region.current
aws_eip.nat_gateway_eip
aws_instance.ubuntu_server
aws_instance.web_server
```


```
aws_internet_gateway.internet_gateway
aws_key_pair.generated
aws_nat_gateway.nat_gateway
aws_route_table.private_route_table
aws_route_table.public_route_table
aws_route_table_association.private["private_subnet_1"]
aws_route_table_association.private["private_subnet_2"]
aws_route_table_association.private["private_subnet_3"]
aws_route_table_association.public["public_subnet_1"]
aws_route_table_association.public["public_subnet_2"]
aws_route_table_association.public["public_subnet_3"]
aws_security_group.ingress-ssh
aws_security_group.vpc-ping
aws_security_group.vpc-web
aws_subnet.private_subnets["private_subnet_1"]
aws_subnet.private_subnets["private_subnet_2"]
aws_subnet.private_subnets["private_subnet_3"]
aws_subnet.public_subnets["public_subnet_1"]
aws_subnet.public_subnets["public_subnet_2"]
aws_subnet.public_subnets["public_subnet_3"]
aws_vpc.vpc
local_file.private_key_pem
tls_private_key.generated
```

Step 1.3 - Mark the webserver to be recreated with the `taint` command

We can the force the resource to be destroyed and recreated with the taint command:

```
terraform taint aws_instance.web_server
Resource instance aws_instance.web_server has been marked as tainted.
```

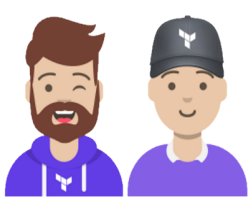
Step 1.4 - Recreate the Web Server

Now we we execute a `terraform plan` and `terraform apply` we will notice that terraform has marked the resource to be recreated. This is because the resource has been tainted.

```
terraform plan
```

```
  # aws_instance.web_server is tainted, so must be replaced
-/+ resource "aws_instance" "web_server" {

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

```
terraform apply
```

```
Plan: 1 to add, 0 to change, 1 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_instance.web_server: Destroying... [id=i-0c06f6a2b91486226]
aws_instance.web_server: Still destroying... [id=i-0c06f6a2b91486226, 10s
   elapsed]
aws_instance.web_server: Still destroying... [id=i-0c06f6a2b91486226, 20s
   elapsed]
aws_instance.web_server: Still destroying... [id=i-0c06f6a2b91486226, 30s
   elapsed]
aws_instance.web_server: Still destroying... [id=i-0c06f6a2b91486226, 40s
   elapsed]
aws_instance.web_server: Destruction complete after 41s
aws_instance.web_server: Creating...
aws_instance.web_server: Still creating... [10s elapsed]
aws_instance.web_server: Still creating... [20s elapsed]
aws_instance.web_server: Still creating... [30s elapsed]
aws_instance.web_server: Provisioning with 'local-exec'...
aws_instance.web_server (local-exec): Executing: ["/bin/sh" "-c" "chmod
   600 MyAWSKey.pem"]
aws_instance.web_server: Provisioning with 'remote-exec'...
aws_instance.web_server (remote-exec): Connecting to remote host via SSH
   ...
aws_instance.web_server (remote-exec):   Host: 18.236.86.172
aws_instance.web_server (remote-exec):   User: ubuntu
aws_instance.web_server (remote-exec):   Password: false
aws_instance.web_server (remote-exec):   Private key: true
aws_instance.web_server (remote-exec):   Certificate: false
aws_instance.web_server (remote-exec):   SSH Agent: false
aws_instance.web_server (remote-exec):   Checking Host Key: false
aws_instance.web_server (remote-exec):   Target Platform: unix
...
aws_instance.web_server: Creation complete after 57s [id=i-00
   e3dafde418bdf73]
```


Task 2: Observe a `remote-exec` provisoner failing, resulting in Terraform automatically tainting a resource

In addition to being able to `taint` a resource manually, Terraform will automatically taint a resource if it is configured with a creation-time provisioner and that provisioner fails. A tainted resource will be planned for destruction and recreation upon the next terraform apply. Terraform does this because a failed provisioner can leave a resource in a semi-configured state. Because Terraform cannot reason about what the provisioner does, the only way to ensure proper creation of a resource is to recreate it.

Modify the `remote-exec` provisioner within the `aws_instance.web_server` resource block to intentionally include a bad command.

```
resource "aws_instance" "web_server" {
  ami              = data.aws_ami.ubuntu.id
  instance_type    = "t2.micro"
  subnet_id        = aws_subnet.public_subnets["public_subnet_1"].id
  security_groups = [aws_security_group.vpc-ping.id, aws_security_group.
     ingress-ssh.id, aws_security_group.vpc-web.id]
  key_name         = aws_key_pair.generated.key_name
  connection {
    user        = "ubuntu"
    private_key = tls_private_key.generated.private_key_pem
    host        = self.public_ip
  }
  associate_public_ip_address = true
  tags = {
    Name = "Web EC2 Server"
  }

  provisioner "local-exec" {
    command = "chmod 600 ${local_file.private_key_pem.filename}"
  }

  provisioner "remote-exec" {
    inline = [
      "exit 2",
      "git clone https://github.com/hashicorp/demo-terraform-101",
      "cp -a demo-terraform-101/. /tmp/",
      "sudo sh /tmp/assets/setup-web.sh",
    ]
  }
}
```

Manaully `taint` the resource and run a `terraform apply` and observe that the `remote-exec` provisioner fails.

```
terraform taint aws_instance.web_server
terraform apply
```

```
| Error: remote-exec provisioner error
|
|   with aws_instance.web_server,
|   on main.tf line 169, in resource "aws_instance" "web_server":
|  169:   provisioner "remote-exec" {
|
| error executing "/tmp/terraform_1714387465.sh": Process exited with
   status 2
```

This time Terraform automatically tainted the resource. You can see resource has been marked as `tainted` by looking at the resource using the `terraform show` command.

```
terraform state show aws_instance.web_server
```

```
# aws_instance.web_server: (tainted)
resource "aws_instance" "web_server" {
```

You will notice that because the `remote-exec` provisioner failed, Terraform automatically marked it as `tainted`. This informs Terraform that this resource needs to be rebuilt upon the next terraform apply.

Untaint a resource

You can also `untaint` a resource by using the `terraform untaint` command.

```
terraform untaint aws_instance.web_server
```

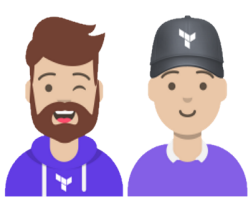
```
Resource instance aws_instance.web_server has been successfully untainted.
```

This informs Terraform that this resource does not need to be rebuilt upon the next terraform apply.

Task 4: Use the `-replace` option rather than `taint`

As of Terraform v0.15.2 and later the `taint` command is deprecated, because there are better alternatives available.

If your intent is to force replacement of a particular object even though there are no configuration changes that would require it, it is recommended to use the `-replace` option with terraform apply in place of the depcreated `taint` command.



Remove the offending error from the `remote-exec` provisioner and rebuild the web server resource using the `terraform apply -replace` command.

```
provisioner "remote-exec" {
    inline = [
      "git clone https://github.com/hashicorp/demo-terraform-101",
      "cp -a demo-terraform-101/. /tmp/",
      "sudo sh /tmp/assets/setup-web.sh",
    ]
  }
```

```
terraform apply -replace="aws_instance.web_server"
```

```
Plan: 1 to add, 0 to change, 1 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_instance.web_server: Destroying... [id=i-00f52f68eec42518b]
aws_instance.web_server: Still destroying... [id=i-00f52f68eec42518b, 10s
    elapsed]
aws_instance.web_server: Still destroying... [id=i-00f52f68eec42518b, 20s
    elapsed]
aws_instance.web_server: Still destroying... [id=i-00f52f68eec42518b, 30s
    elapsed]
aws_instance.web_server: Destruction complete after 31s
aws_instance.web_server: Creating...
aws_instance.web_server: Still creating... [10s elapsed]
aws_instance.web_server: Still creating... [20s elapsed]
aws_instance.web_server: Still creating... [30s elapsed]
aws_instance.web_server: Still creating... [40s elapsed]
aws_instance.web_server: Provisioning with 'local-exec'...
aws_instance.web_server (local-exec): Executing: ["/bin/sh" "-c" "chmod
    600 MyAWSKey.pem"]
aws_instance.web_server: Provisioning with 'remote-exec'...
aws_instance.web_server (remote-exec): Connecting to remote host via SSH
    ...
aws_instance.web_server (remote-exec):   Host: 34.214.67.54
aws_instance.web_server (remote-exec):   User: ubuntu
aws_instance.web_server (remote-exec):   Password: false
aws_instance.web_server (remote-exec):   Private key: true
aws_instance.web_server (remote-exec):   Certificate: false
aws_instance.web_server (remote-exec):   SSH Agent: false
aws_instance.web_server (remote-exec):   Checking Host Key: false
aws_instance.web_server (remote-exec):   Target Platform: unix
```


```
aws_instance.web_server (remote-exec): Connected!
aws_instance.web_server (remote-exec): Cloning into 'demo-terraform-101'
    ...
aws_instance.web_server: Still creating... [50s elapsed]
aws_instance.web_server (remote-exec): remote: Enumerating objects: 417,
    done.
aws_instance.web_server (remote-exec): Receiving objects:   0% (1/417)
aws_instance.web_server (remote-exec): Receiving objects:   1% (5/417)
aws_instance.web_server (remote-exec): Receiving objects:   2% (9/417)
...
aws_instance.web_server (remote-exec): Resolving deltas: 100% (142/142),
    done.
aws_instance.web_server (remote-exec): cp: preserving times for '/tmp/.':
    Operation not permitted
aws_instance.web_server (remote-exec): Created symlink /etc/systemd/system
    /multi-user.target.wants/webapp.service -> /lib/systemd/system/webapp.
    service.
aws_instance.web_server: Creation complete after 53s [id=i-03
    db5c21e61154d36]
```

Using the `-replace` command to rebuild a resource is the recommended approach moving forward.