Local 지정

Local resource

```
$ vim main.tf

provider "local" {
}

# Infra resource
resource "local_file" "foo" {
    # path.module - String inter pdation terraform 변수
    # ${path.module} : string interpolation 문법
    # 파일이 위치한 디렉토리 경로, main.tf
    filename = "${path.module}/foo.txt"
    content = "Hello World!"
}
```

Terraform init

```
$ terraform init
Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 4.0"...
- Installing hashicorp/aws v4.45.0...
- Installed hashicorp/aws v4.45.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

$ ls -al
drwxr-xr-x  3 ubuntu ubuntu 4096 Jan 21 09:40 .terraform
-rw-r--r--  1 ubuntu ubuntu 1078 Jan 21 09:40 .terraform.lock.hcl
-rw-r--r--  1 ubuntu ubuntu  124 Jan 21 09:40 main.tf
```

Terraform 디렉토리 확인

```
# 프로바이드의 데이터를 다운받아서 저장
$ tree .terraform/
.terraform
└── providers
    └── registry.terraform.io
        └── hashicorp
            └── local
                └── 2.2.3
                    └── linux_amd64
                        └── terraform-provider-local_v2.2.3_x5

# 협업에 필요한 데이터 저장
$ cat .terraform.locl.hcl
# This file is maintained automatically by "terraform init".
# Manual edits may be lost in future updates.

provider "registry.terraform.io/hashicorp/local" {
  version = "2.2.3"
  hashes = [
    "h1:aWp5iSUxBGgPv1UnV5yag9Pb0N+U1I0sZb38AXBFO8A=",
    "zh:04f0978bb3e052707b8e82e46780c371ac1c66b689b4a23bbc2f58865ab7d5c0",
    "zh:6484f1b3e9e3771eb7cc8e8bab8b35f939a55d550b3f4fb2ab141a24269ee6aa",
    "zh:78a56d59a013cb0f7eb1c92815d6eb5cf07f8b5f0ae20b96d049e73db915b238",
    "zh:78d5eefdd9e494defcb3c68d282b8f96630502cac21d1ea161f53cfe9bb483b3"
    ...
  ]
}
```

Terraform plan

```
$ terraform plan
Terraform used the selected providers to generate the following
execution plan. Resource actions are indicated with the following
symbols:
  + create

Terraform will perform the following actions:

  # local_file.foo will be created
  + resource "local_file" "foo" {
      + content              = "Hello World!"
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "./foo.txt"
      + id                   = (known after apply)
    }

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

Terraform apply

```
$ terraform apply
Terraform used the selected providers to generate the following
execution plan. Resource actions are indicated with the following
symbols:
  + create

Terraform will perform the following actions:

  # local_file.foo will be created
  + resource "local_file" "foo" {
      + content              = "Hello World!"
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "./foo.txt"
      + id                   = (known after apply)
    }

Plan: 1 to add, 0 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 # yes 입력

local_file.foo: Creating...
local_file.foo: Creation complete after 0s [id=2ef7bde608ce5404e97d5f042f95f89f1c232871]

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

$ ls -al
drwxr-xr-x  3 ubuntu ubuntu 4096 Jan 21 09:48 .
drwxr-xr-x 15 ubuntu ubuntu 4096 Jan 21 09:49 ..
drwxr-xr-x  3 ubuntu ubuntu 4096 Jan 21 09:40 .terraform
-rw-r--r--  1 ubuntu ubuntu 1078 Jan 21 09:40 .terraform.lock.hcl
-rwxr-xr-x  1 ubuntu ubuntu   12 Jan 21 09:48 foo.txt
-rw-r--r--  1 ubuntu ubuntu  124 Jan 21 09:40 main.tf
-rw-r--r--  1 ubuntu ubuntu  829 Jan 21 09:48 terraform.tfstate
```

출력파일 확인

```
$ cat foo.txt
Hello World!
```

상태 파일

```
$ cat terraform.tfstate
{
  "version": 4,
  "terraform_version": "1.1.4",
  "serial": 1,
  "lineage": "e43e603f-6a95-f4bd-6609-7dd20fae085f",
  "outputs": {},
  "resources": [
    {
      "mode": "managed",
      "type": "local_file",
      "name": "foo",
      "provider": "provider[\"registry.terraform.io/hashicorp/local\"]",
      "instances": [
        {
          "schema_version": 0,
          "attributes": {
            "content": "Hello World!",
            "content_base64": null,
            "directory_permission": "0777",
            "file_permission": "0777",
            "filename": "./foo.txt",
            "id": "2ef7bde608ce5404e97d5f042f95f89f1c232871",
            "sensitive_content": null,
            "source": null
          },
          "sensitive_attributes": [],
          "private": "bnVsbA=="
        }
      ]
    }
  ]
}
```

출력 파일 지정

```
$ vim main.tf
provider "local" {
}
resource "local_file" "foo" {
  # 출력 파일 지정
  filename = "${path.module}/foo.txt"
  content = "Hello World!"
}
# 입력 데이터파일 지정
data "local_file" "bar" {
  filename = "${path.module}/bar.txt"
}
```

입력 파일 생성

```
$ vim bar.txt
Hello DevOps!

$ cat bar.txt
Hello DevOps!
```

실행

```
$ terraform apply
local_file.foo: Refreshing state... [id=2ef7bde608ce5404e97d5f042f95f89f1c232871]

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.
```

파일 내용 출력 지정

```
$ vim main.tf

provider "local" {
}
resource "local_file" "foo" {
  # 출력 파일 지정
  filename = "${path.module}/foo.txt"
  content = "Hello World!"
}
# 입력 데이터파일 지정
data "local_file" "bar" {
  filename = "${path.module}/bar.txt"
}

# 데이터 출력 지정
output "file_bar" {
    value = data.local_file.bar
}
```

실행

```
$ terraform apply

local_file.foo: Refreshing state... [id=2ef7bde608ce5404e97d5f042f95f89f1c232871]
data.local_file.bar: Reading...
data.local_file.bar: Read complete after 0s [id=5097daa995194730034d8d1949cfafa2659cb2bd]

Changes to Outputs:
  + file_bar = {
      + content        = <<-EOT
             DevOps!
        EOT
      + content_base64 = "IERldk9wcyEK"
      + filename       = "./bar.txt"
      + id             = "5097daa995194730034d8d1949cfafa2659cb2bd"
    }
```

```
You can apply this plan to save these new output values to the Terraform state, without changing any
real infrastructure.

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

Outputs:

file_bar = {
  "content" = <<-EOT
   DevOps!

  EOT
  "content_base64" = "IERldk9wcyEK"
  "filename" = "./bar.txt"
  "id" = "5097daa995194730034d8d1949cfafa2659cb2bd"
}
```

완성된 main.tf

```
provider "local" {
}

resource "local_file" "foo" {
    filename = "${path.module}/foo.txt"
    content = "Hello World!"
}

data "local_file" "bar" {
  filename = "${path.module}/bar.txt"
}

output "file_bar" {
  value = data.local_file.bar
}
```