



# HashiTalks



# 테라폼 대규모 이관, Community to Enterprise

커뮤니티 에디션에서 상용 버전으로 전환



# 박준상

Snr. Solutions Engineer at HashiCorp

[jsp@hashicorp.com](mailto:jsp@hashicorp.com)



— 01 CE to Ent. / Single Workspace

# 단일 Workspace

단일 워크스페이스 이관하기



# 단일 Workspace 이관



<https://developer.hashicorp.com/terraform/cloud-docs/migrate>

1. Terraform 버전 확인
2. cred(사용자 API 토큰), 데이터(상태) 및 코드를 준비
3. 백엔드 구성(hostname, TFE org 및 Workspace name)파일을 생성 후, terraform login 명령어를 통해 API Token을 로컬 환경에 저장.
4. 작업 디렉토리에서 workspace 마이그레이션을 위해 terraform init 실행.  
Workspace 자동 생성
5. Terraform Enterprise내 자동 생성된 Workspace 설정(VCS, 변수, 팀 권한 등) 변경
6. 작업 디렉토리에서 “*terraform plan*” 실행하거나 직접 UI에서 “*queue a plan*” 선택하여 정상 동작 여부 확인

state 파일 백업 과정이 없다.

이관 대상이 적은 경우, 개별 state 파일 이관 후 Workspace 설정 가능.  
대량의 파일 이관 시, 반복 작업 시 오류 발생 가능

## 02 CE to Ent. / Multiple Workspaces

# 대규모 Workspace

다수의 Workspace를 동시에 이관하기



# 복수 개 Workspace 이관



<https://developer.hashicorp.com/terraform/cli/cloud/migrating#multiple-workspaces>

```
terraform {  
  - backend "remote" {  
  + cloud {  
      organization = "my-org"  
  
      workspaces {  
        - prefix = "my-app-"  
        + tags = ["app:mine"]  
      }  
    }  
  }  
}
```

"접두사" 인수를 사용하는 복수 개의  
Workspace를 이관하는 경우

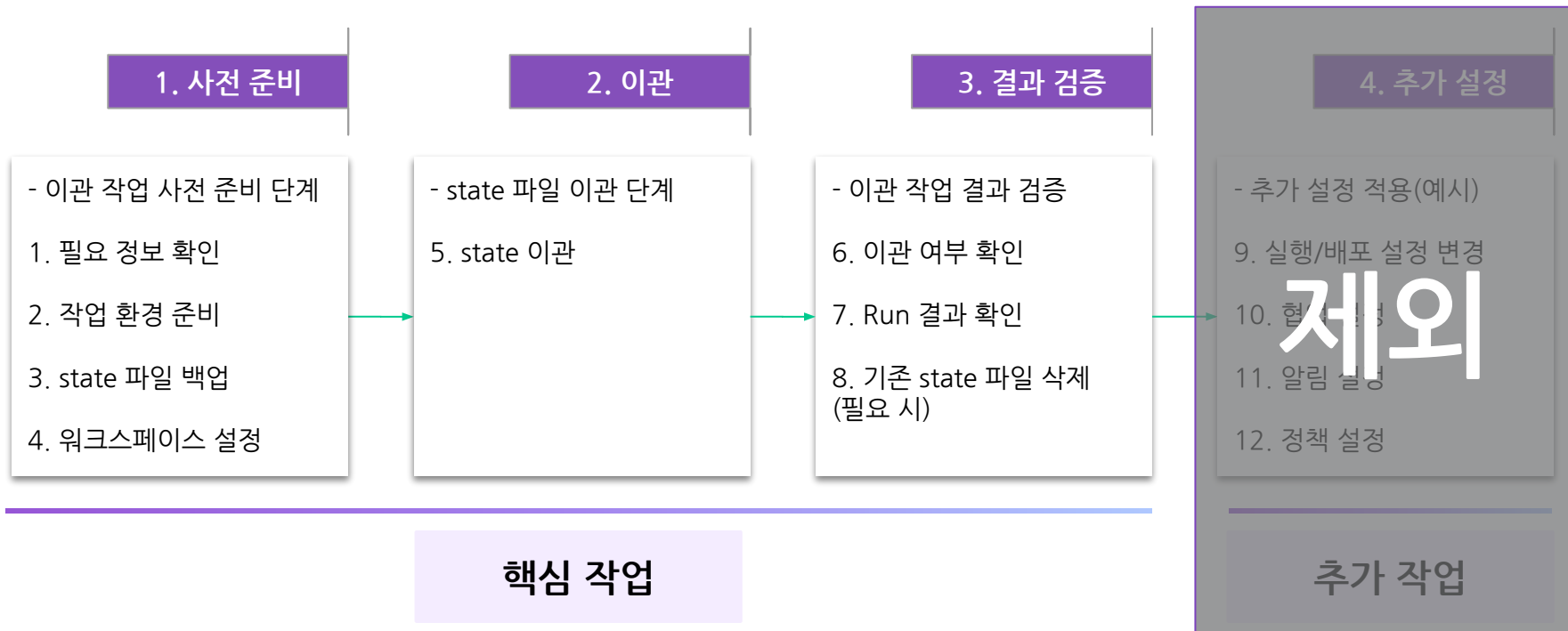
→ backend 설정을 Cloud로, "tag"를 지정.

⇒ 접두사를 사용하지 않는 경우는?



# 수정된 마이그레이션 절차

안전한 데이터 이관 및 대규모 이관을 고려





# 필요정보 확인

## 클라우드 계정 정보, API 토큰 등



### 작업 내용

인프라 배포 및 Terraform Enterprise 사용을 위한 인증 정보, Configuration Template 등 확인

- 인프라 배포 시 사용하는 클라우드 인증 정보
- 테라폼 엔터프라이즈 API 토큰
- 기 사용 중인 테라폼 Configuration template 파일
- 현재 원격 저장소 정보 및 state 파일

[main.tf]

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = ">= 3.68.0"
    }
  }
  required_version = ">= 1.0.7"
}

provider "aws" {
  region = "ap-northeast-2"
}

resource "aws_vpc" "testvpc" {
  cidr_block = "10.0.0.0/16"
}

output "vpcid" {
  value = aws_vpc.testvpc.id
}
```

[remote\_backend.tf]

```
terraform {
  backend "S3" {
    bucket = "tfs3store"
    key   = "tf/oss/remote"
    region = "ap-northeast-2"
  }
}
```

> tree

```
.
├── files
│   └── deploy_app.sh
├── main.tf
├── outputs.tf
├── remote_backend.tf
├── terraform.tfstate
├── terraform.tfvars
└── variables.tf
```

# 작업환경준비

## 작업 디렉토리 내 각종 파일 복사



### 작업 내용

작업 디렉토리 내에 state 파일, Configuration Template을 복사 후, 이관 작업 준비

- 작업 디렉토리 내 Configuration Template 복사  
(hidden 디렉토리 .terraform 포함할 것)  
또는 Configuration Template 복사 후  
terraform init 수행
- Remote Backend 설정 변경:  
S3 → Terraform Enterprise

```
[remote_backend.tf]
terraform {
  backend "S3" {
    bucket = "tfs3store"
    key    = "tf/oss/remote"
    region = "ap-northeast-2"
  }
}

↓

terraform {
  backend "remote" {
    hostname = "app.terraform.io"
    organization = "hashitalks-kr"
    workspaces {
      name = "talks-hashicat-aws"
    }
  }
}
```

# 데이터 백업

## 기존 state 파일 백업



### 작업 내용

리모트 백엔드 설정 파일을 이용, 작업 대상 state 파일 백업

- 별도 백업 디렉토리 내에 remote\_backend 파일 설정
- state 파일 백업 : terraform state pull > {워크스페이스\_이름}/terraform.tfstate
- 위 명령어 기반 스크립트로 복수의 파일을 동시에 백업(다운로드)

[remote\_backend.tf]

```
terraform {  
  backend "s3" {  
    bucket = "tfs3store"  
    key    = "tf/oss/remote"  
    region = "ap-northeast-2"  
  }  
}
```

[down.sh]

```
#!/bin/bash  
# Terraform state 파일을 다운로드할 디렉토리  
STATE_DIR=( "../aws001" "../aws002")  
# Shell Script를 실행할 디렉토리  
WORKING_DIR="../state_backup"  
# STATE_DIR 별 State 파일 다운로드  
for directory in "${STATE_DIR[@]"; do  
# 해당 디렉토리로 이동  
cd $directory  
# Terraform state 파일 다운로드  
terraform state pull > terraform.tfstate  
cd $WORKING_DIR  
done
```

# 워크스페이스

## 엔터프라이즈 내 워크스페이스 생성 및 설정



### 작업 내용

테라폼 엔터프라이즈 내 Workspace 생성 및 기본 설정. 테라폼 코드로 구성 예정

- 조직-프로젝트-워크스페이스 수준에 맞춰 생성
- AWS 배포를 위한 Credential 설정
- 기본 변수 설정 : Variable 또는 Variables Set
- CLI Driven, Remote Execution으로 설정.
- 현재 사용 중인 테라폼 버전 지정 등

hashicorp-jennewing / Projects & workspaces / hashicat-aws / Variables

hashicat-aws  
ID: ws-58Tc2h8UJsbWwv [🔗](#)

Resources: 0 Terraform version: 1.2.5 Updated: 10 months ago

No workspace description available. [Add workspace description.](#)

[🔗 Unlinked](#) [Actions](#)

### Variables

Terraform uses all [Terraform](#) and [Environment](#) variables for all plans and applies in this workspace. Workspaces using Terraform 0.10.0 or later can also load default values from any [\\*.auto.tfvars](#) files in the configuration. You may want to use the Terraform Cloud Provider or the variables API to add multiple variables at once.

#### Sensitive variables

[Sensitive](#) variables are never shown in the UI or API, and can't be edited. They may appear in Terraform logs if your configuration is designed to output them. To change a sensitive variable, delete and replace it.

#### Workspace variables (0)

Variables defined within a workspace always overwrite variables from variable sets that have the same type and the same key. [Learn more about variable set precedence](#).

Key	Value	Category
There are no variables added.		

Select variable category

☒ Terraform variable  
These variables should match the declarations in your configuration. Click the HCL box to use interpolation or set a non-string value.

☐ Environment variable  
These variables are available in the Terraform runtime environment.

Key:  Value:  ☐ HCL ☐ Sensitive

Description (Optional):

[Add variable](#) [Cancel](#)

## state 이관

## 복수 파일 이관



## 작업 내용

다수의 state 파일을 이관

- Terraform Configuration Template에서 Workspace API 호출
- “terraform\_data”에서 local-exec Provisioner로 API 호출
- 다음 순서로 동작 :  
Workspace 잠금  
→ State 파일 업로드  
→ Workspace 잠금 해제

```
provisioner "local-exec" {
  command = <<EOT
    curl -s \
      --header "Content-Type: application/vnd.api+json" \
      --header "Authorization: Bearer ${var.tfe_token}" \
      --request POST \
      --data '{ "reason": "Locking ${data.tfe_workspace.target_workspace[each.key].name} for state upload" }' \
      ${var.api_url}/workspaces/${data.tfe_workspace.target_workspace[each.key].id}/actions/lock
    echo "${data.tfe_workspace.target_workspace[each.key].name}가 성공적으로 잠금처리되었습니다."
  EOT
}
```

You, 2 months ago • Initial Commit

```
provisioner "local-exec" {
  command = <<EOT
    curl \
      -s --request POST \
      --header "Authorization: Bearer ${var.tfe_token}" \
      --header "Content-Type: application/vnd.api+json" \
      --data '{
        "data": {
          "type": "state-versions",
          "attributes": {
            "serial": "${each.value["serial"]}",
            "md5": "${md5(file("../${each.key}/terraform.tfstate"))}",
            "lineage": "${each.value["lineage"]}",
            "state": "${base64encode(file("../${each.key}/terraform.tfstate"))}"
          }
        }
      }' \
      ${var.api_url}/workspaces/${data.tfe_workspace.target_workspace[each.key].id}/state-versions
    echo " Statefile을 Workspace ${data.tfe_workspace.target_workspace[each.key].name}로 성공적으로 Upload 하였습니다."
  EOT
}
```

# 이관여부확인

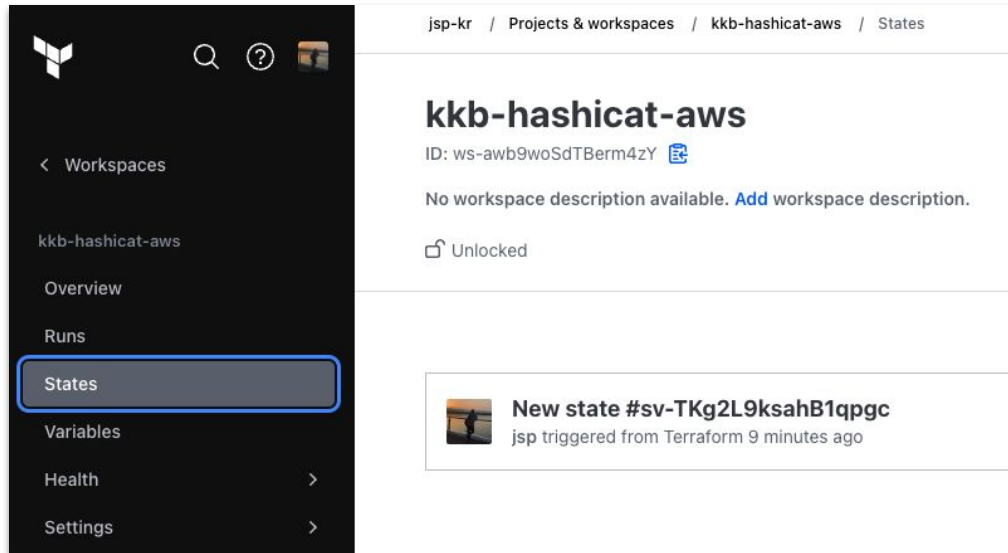
## 대상 워크스페이스 내 state 파일 확인



### 작업내용

이관 대상 Workspace에 접속, state 파일 이관  
정상 여부 확인

- 테라폼 엔터프라이즈 접속 >  
Organization 선택 > (Project 선택) >  
대상 Workspace 선택
- Workspace 메뉴에서 “States” 선택
- 우측화면처럼 state 파일이 존재해야 함



# Run결과확인

## 이관된 인프라 관리 가능 여부 확인



### 작업 내용

엔터프라이즈로 state 파일 이관 후, 기존 형상에 대한 관리 가능 여부 확인

- CLI Driven, Local Execution 으로 설정된 경우, 현재 상태에서 Plan 또는 Run 실행 (1차 검증)
- Workspace 내 메뉴에서 Settings > General에서 Execution Mode를 “Remote”로 설정
- CLI Driven, Remote Execution으로 수정 후 Plan 또는 Run 실행 (2차 검증)

```
$ terraform plan
```

Running apply in the remote backend. Output will stream here. Pressing Ctrl-C will cancel the remote apply if it's still pending. If the apply started it will stop streaming the logs, but will not stop the apply running remotely.

Preparing the remote apply...

To view this run in a browser, visit:

<https://app.terraform.io/app/hashitalks-kr/talks-hashicat-aws/runs/run-6VhMTXp4SdYGHUq5>

Waiting for the plan to start...

```
Terraform v1.5.2
on linux_amd64
Initializing plugins and modules...
tls_private_key.hashicat: Refreshing state... [id=95601697ffff3dc4ba278a098a924b67b2a68177]
data.aws_ami.ubuntu: Reading...
aws_key_pair.hashicat: Refreshing state... [id=talks-ssh-key.pem]
aws_vpc.hashicat: Refreshing state... [id=vpc-050e9f6761230fee9]
data.aws_ami.ubuntu: Read complete after 1s [id=ami-0502b8f5f0ca3ed7d]
aws_internet_gateway.hashicat: Refreshing state... [id=igw-0204e8de43da12d33]
aws_subnet.hashicat: Refreshing state... [id=subnet-0f0180caa1b0e1da1]
aws_security_group.hashicat: Refreshing state... [id=sg-0c966ba7064660c48]
aws_route_table.hashicat: Refreshing state... [id=rtb-012ee453024d179a9]
aws_instance.hashicat: Refreshing state... [id=i-0829f53336583cca6]
aws_route_table_association.hashicat: Refreshing state... [id=rtbassoc-0092c9da751db0f72]
aws_eip.hashicat: Refreshing state... [id=eipalloc-03e026c801212660e]
aws_eip_association.hashicat: Refreshing state... [id=eipassoc-032c2cffffd902a2c]
null_resource.configure-cat-app: Refreshing state... [id=8236399718520921727]
```

**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.

# 기존파일삭제

## 기존 Remote 백엔드, 이관 작업 디렉토리



### 작업 내용

정상적으로 이관된 경우, 기존 State파일과  
작업용 디렉토리 삭제

- 기존 Backend에 저장된 state 파일  
(예:S3)
- 작업을 위해 백업한 state 파일 및  
Terraform Configuration Template

```
> tree /hashicat-aws
./hashicat-aws001
├── files
│   └── deploy_app.sh
├── main.tf
├── outputs.tf
├── remote_backend.tf
├── terraform.tfstate
├── terraform.tfvars
└── variables.tf
```

2 directories, 7 files

```
> tree /state_backup
./state_backup
├── README.md
└── down.sh
```

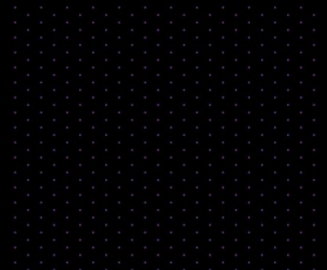
1 directory, 2 files



03 CE to Ent. / Demo

# 시연

대규모 Workspace 마이그레이션



# 시연 순서



## 작업환경준비

이관 작업을 수행할  
디렉토리 구성 및  
Terraform  
Configuration  
Template 다운로드

## State파일백업

원격 저장소(S3)에서  
작업 대상 디렉토리로  
state file 다운로드

## Workspace생성

Terraform Cloud 내  
대상 Org에 이관할 워크  
스페이스를 생성

## State 이관

백업받은 State 파일을  
대상 Workspace로 이관

## 이관여부확인

이관된 State 파일을  
사용하여 인프라 관리  
가능 여부 확인

## #2. State 파일 백업



### State 파일 백업

원격 저장소(S3)에서  
작업 대상 디렉토리로  
state file 다운로드

```
./01_ent001
├── files
│   ├── deploy_app.sh
│   ├── main.tf
│   ├── outputs.tf
│   ├── remote_backend.tf
│   ├── terraform.tfvars
│   └── variables.tf
├── ./01_hc02
│   ├── files
│   │   ├── deploy_app.sh
│   │   ├── main.tf
│   │   ├── outputs.tf
│   │   ├── remote_backend.tf
│   │   ├── terraform.tfvars
│   │   └── variables.tf
│   └── ./01_tkdemo3
│       ├── files
│       │   ├── deploy_app.sh
│       │   ├── main.tf
│       │   ├── outputs.tf
│       │   ├── remote_backend.tf
│       │   ├── terraform.tfvars
│       │   └── variables.tf
│       └── ./01_weba001
│           ├── files
│           │   ├── deploy_app.sh
│           │   ├── main.tf
│           │   ├── outputs.tf
│           │   ├── remote_backend.tf
│           │   ├── terraform.tfvars
│           │   └── variables.tf
```

8 directories, 24 files

```
> tree
.
├── README.md
└── down.sh

1 directory, 2 files
> sh down.sh
===== 작업 Directory: ../01_ent001 =====
state file download
-rw-r--r-- 1 jsp staff 22306 Aug 29 15:02 terraform.tfstate
state file download 완료

===== 작업 Directory: ../01_hc02 =====
state file download
-rw-r--r-- 1 jsp staff 22321 Aug 29 15:02 terraform.tfstate
state file download 완료

===== 작업 Directory: ../01_tkdemo3 =====
state file download
-rw-r--r-- 1 jsp staff 22461 Aug 29 15:02 terraform.tfstate
state file download 완료

===== 작업 Directory: ../01_weba001 =====
state file download
-rw-r--r-- 1 jsp staff 22489 Aug 29 15:02 terraform.tfstate
state file download 완료
```

```
> tree ./01_*
./01_ent001
├── files
│   ├── deploy_app.sh
│   ├── main.tf
│   ├── outputs.tf
│   ├── remote_backend.tf
│   ├── terraform.tfstate
│   ├── terraform.tfvars
│   └── variables.tf
├── ./01_hc02
│   ├── files
│   │   ├── deploy_app.sh
│   │   ├── main.tf
│   │   ├── outputs.tf
│   │   ├── remote_backend.tf
│   │   ├── terraform.tfstate
│   │   ├── terraform.tfvars
│   │   └── variables.tf
│   └── ./01_tkdemo3
│       ├── files
│       │   ├── deploy_app.sh
│       │   ├── main.tf
│       │   ├── outputs.tf
│       │   ├── remote_backend.tf
│       │   ├── terraform.tfstate
│       │   ├── terraform.tfvars
│       │   └── variables.tf
│       └── ./01_weba001
│           ├── files
│           │   ├── deploy_app.sh
│           │   ├── main.tf
│           │   ├── outputs.tf
│           │   ├── remote_backend.tf
│           │   ├── terraform.tfstate
│           │   ├── terraform.tfvars
│           │   └── variables.tf
```

8 directories, 28 files

# #3. Workspace 생성



## 워크스페이스 생성

Terraform Cloud 내  
대상 Org에 이관할 워크  
스페이스를 생성

```
Changes to Outputs:
+ ws_name = [
+   "01_ent001",
+   "01_hc02",
+   "01_tkdemo3",
+   "01_weba001",
+ ]

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

tfe_workspace.test[2]: Creating...
tfe_workspace.test[3]: Creating...
tfe_workspace.test[1]: Creating...
tfe_workspace.test[0]: Creating...
tfe_workspace.test[1]: Creation complete after 1s [id=ws-id3dhzjnsfJNLJ9Y]
tfe_workspace.test[3]: Creation complete after 2s [id=ws-y1i4HDN3E5txn2ES]
tfe_workspace.test[0]: Creation complete after 2s [id=ws-uhUpC5T7ZX8vkPpG]
tfe_workspace.test[2]: Creation complete after 2s [id=ws-LKtDsKvvvj92sm74]

Apply complete! Resources: 4 added, 0 changed, 0 destroyed.

Outputs:
ws_name = [
  "01_ent001",
  "01_hc02",
  "01_tkdemo3",
  "01_weba001",
]
```

Workspaces in hashitalks				
<div>0 0 0 0 0</div>				
Workspace Name ↓				
01_ent001	2023	demo	hashitalks	migrated
01_hc02	2023	demo	hashitalks	migrated
01_tkdemo3	2023	demo	hashitalks	migrated
01_weba001	2023	demo	hashitalks	migrated

## #4. State 이관



### State 파일 이관

백업받은 State 파일을  
대상 Workspace로 이관

```
> sh get_state_info.sh
```

출력이 완료되었습니다. 결과는 output.txt에 저장되었습니다.

```
> cat output.txt
```

```
"01_ent001" = {ws_name = "01_ent001",serial=8, lineage="3eb06845-a608-02ce-fde2-a8d650b21fd0" }  
"01_hc02" = {ws_name = "01_hc02",serial=7, lineage="93b83b15-03a6-2bad-e0f3-200c0c551c06" }  
"01_tkdemo3" = {ws_name = "01_tkdemo3",serial=2, lineage="e7387148-b2e3-9c7f-3c17-1cfb041f76fc" }  
"01_weba001" = {ws_name = "01_weba001",serial=2, lineage="467c2701-f9a2-313a-1486-5b2eabbf0cee" }
```

```
state_info = {  
  "01_ent001" = {ws_name = "01_ent001",serial=8, lineage="3eb06845-a608-02ce-fde2-a8d650b21fd0" }  
  "01_hc02" = {ws_name = "01_hc02",serial=7, lineage="93b83b15-03a6-2bad-e0f3-200c0c551c06" }  
  "01_tkdemo3" = {ws_name = "01_tkdemo3",serial=2, lineage="e7387148-b2e3-9c7f-3c17-1cfb041f76fc" }  
  "01_weba001" = {ws_name = "01_weba001",serial=2, lineage="467c2701-f9a2-313a-1486-5b2eabbf0cee" }  
}
```

# #5. 이관 여부 확인



## 이관 여부 확인

이관된 State 파일을  
사용하여 인프라 관리  
가능 여부 확인

01\_ent001  
ID: ws-uHupC5T7ZXBvKpPpG  
No workspace description available. [Add workspace description.](#)  
Unlocked

Resources 13  
Outputs 2  
Current as of the most recent state version.

NAME	PROVIDER	TYPE	MODULE	CREATED
ubuntu	hashicorp/aws	data.aws_ami	root	Aug 29 2023
hashicat	hashicorp/aws	aws_eip	root	Aug 29 2023
hashicat	hashicorp/aws	aws_eip_asso...	root	Aug 29 2023
hashicat	hashicorp/aws	aws_instance	root	
hashicat	hashicorp/aws	aws_internet...	root	
hashicat	hashicorp/aws	aws_key_pair	root	

```
> terraform plan
Acquiring state lock. This may take a few moments...
tls_private_key.hashicat: Refreshing state... [id=9eefc0ebcc535f2b8fa015e67feb53b7b0127bbb]
aws_key_pair.hashicat: Refreshing state... [id=hc02-ssh-key.pem]
data.aws_ami.ubuntu: Reading...
aws_vpc.hashicat: Refreshing state... [id=vpc-079ec24f39f3ec7a2]
data.aws_ami.ubuntu: Read complete after 0s [id=ami-039c6c8b7212df747]
aws_internet_gateway.hashicat: Refreshing state... [id=igw-0c4c1993beb352a80]
aws_subnet.hashicat: Refreshing state... [id=subnet-0335c2dfcf1df4f4a]
aws_security_group.hashicat: Refreshing state... [id=sg-09b8db2a9853af74e]
aws_route_table.hashicat: Refreshing state... [id=rtb-00731806e1e008848]
aws_instance.hashicat: Refreshing state... [id=i-015f902e3afdf72ef]
aws_route_table_association.hashicat: Refreshing state... [id=rtbassoc-08b86130ffac28e91]
aws_eip.hashicat: Refreshing state... [id=eipalloc-0675addbaa047bf6c]
aws_eip_association.hashicat: Refreshing state... [id=eipassoc-0566ebf612bc13af1]
null_resource.configure-cat-app: Refreshing state... [id=8631640802391143542]

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.
```

# Enterprise 사용을 위한 추가 작업



## Enterprise 기능 연계

- VCS 연계
- 실행 방법
- RBAC
- 알림 설정 등

### Version Control

#### Not Connected

Connecting the workspace to Version Control will

[Connect to version control](#)

### Execution Mode

If you change the execution mode any in progress runs will be discarded.

#### ☐ Remote

Your plans and applies occur on Terraform Cloud's infrastructure. You and your team have the ability to review and collaborate on runs within the app.

#### ☒ Local

Your plans and applies occur on machines you control. Terraform Cloud is only used to store and synchronize state.

#### ☐ Agent

Terraform Cloud will manage the plans and applies your agents execute.

### Team Access

#### Heads up

Teams with [project-level](#) or [organization-level](#) permissions

### Create a Notification

Notifications allow you to send messages to other applications based on run and workspace events.

#### Destination



#### Webhook

POST messages to any URL



#### Email

Send messages to users via email



#### Slack

Send messages to a Slack channel



#### Microsoft Teams

Send messages to a Microsoft Teams channel

# Lessons Learned



## 백업!

State 파일 관련  
작업을 할 때는 반드시  
**백업**을 하자!

- 원격 저장소 사용 시 State 파일 관리에 어려움을 겪었다면, Terraform Cloud를 사용해보자!

## 자동화!

가능하면 코드  
기반으로 **표준화** 후,  
수작업을 최소화하자.

- 표준화를 통한 코드 재사용 증대
- Enterprise 도입 시 손쉬운 이관 절차 확인

## Cloud!

프로바이더, 모듈  
그리고 State 파일을  
**최소화**하자!

- Terraform Cloud를 사용, 작업 환경마다 생성되는 .terraform 디렉토리와 state 파일을 Cloud로!



# 관련 자료



## State 마이그레이션 관련

- <https://developer.hashicorp.com/terraform/cli/cloud/migrating>
- <https://developer.hashicorp.com/terraform/tutorials/cli/cloud-migrate>
- <https://developer.hashicorp.com/terraform/cloud-docs/migrate>

## Demo Repo

- <https://github.com/jsp-hashicorp/hashitalks2023>



# HashiTalks

[hugs@hashicorp.com](mailto:hugs@hashicorp.com) | [learn.hashicorp.com](https://learn.hashicorp.com) | [discuss.hashicorp.com](https://discuss.hashicorp.com)



# Thank You

[hugs@hashicorp.com](mailto:hugs@hashicorp.com) | [learn.hashicorp.com](https://learn.hashicorp.com) | [discuss.hashicorp.com](https://discuss.hashicorp.com)