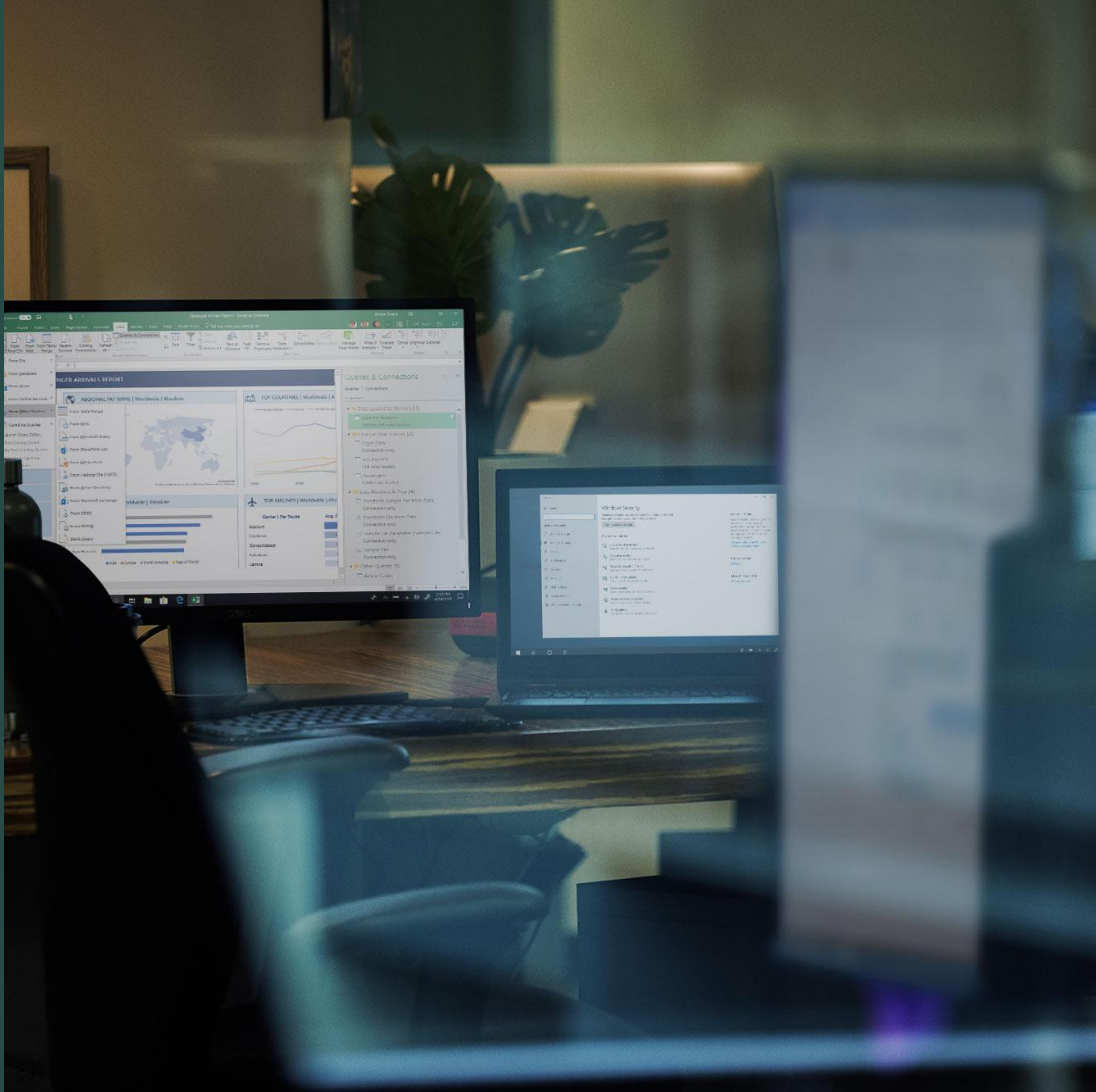
Microsoft

State Management and Provisioners

Subtitle or speaker name



Recap

Bigger picture around IaC

Terraform

- Core Workflow
- Config Options, Expressions, Functions
- Dependency Management and Imports
- Local and Remote Modules
- State Management
  - Remote State Management using Azure Blob Storage
- Niche Topics
  - Workspaces and Provisioners
- DevOps with Terraform

Remote State

Terraform State

Terraform must store state about your managed infrastructure and configuration.

Local State

Remote State

Local state is fragile and not fit for production. (or anything outside local environment)

State File Fundamentals

- State files are required to manage a resource past day 1
- State files can contain secrets
- Local state files cannot be used for a team size of >1
- Continuous Delivery pipelines need access to state files
- State file locking is important
- Granular permissions are not supported with open source Terraform
    - If outputs are required, full state file access is required
- State files contain multi-cloud data that can be mined

State File Scaling and Security

- Remote State must be used
- For Azure: Blob Storage or Terraform Cloud / Enterprise
- Blob Storage structure
  - Single storage account with a single container
  - Single storage account with multiple containers
  - Storage account per subscription, resource group or resource
- Terraform Cloud / Enterprise granular security
  - View runs, but no access to state file
  - Only access to outputs and not full state file
- Use principle of least privilege and automate creation
- Use Zero-Trust: OpenID Connect or Managed Service Identity

terraform backend block - remote state setup for community version

```
# main.tf
terraform {
  backend "azurerm" {
    resource_group_name   = "<rg-name>"
    storage_account_name  = "<sg-acct-name>"
    container_name        = "<container-name>"
    key                   = "terraform.tfstate" # default state file name
  }
}


# Running an init will create a remote backend.
terraform init
```

Partial config with terraform backend

```
# main.tf
terraform {
  backend "azurerm" {
    resource_group_name   = "<rg-name>"
    container_name        = "<container-name>"
    key                   = "terraform.tfstate" # default state file name
  }
}
```

```
# Pass the partial config args during the init
terraform init -backend-config="storage_account_name=<sg-account-name>"
```

Remote State using Azure Storage

Lab 7

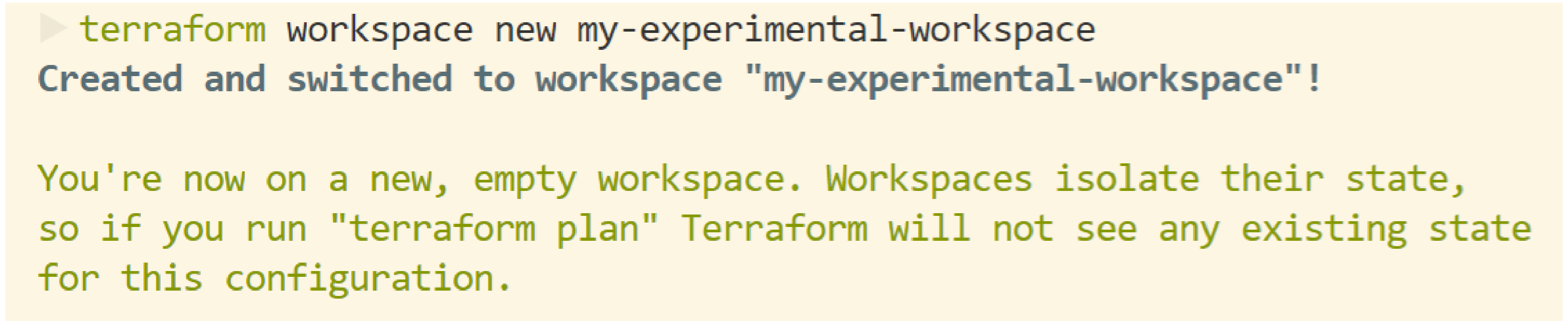
Local Workspaces

Terraform Workspaces

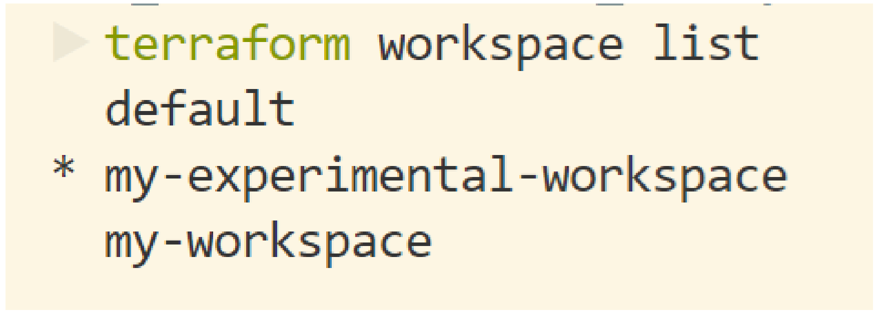
- Provides a way to separate out **.tfstate** files by their own space/directory
- Similar to feature branches in version-control
- Provides isolation during plan, apply and destroy
- Initially, terraform creates just one workspace called "default"
- Remote backends such as AzureRM support having multiple workspaces

Workspaces does not warrant complete isolation use-cases. See https://www.terraform.io/docs/state/workspaces.html#when-to-use-multiple-workspaces

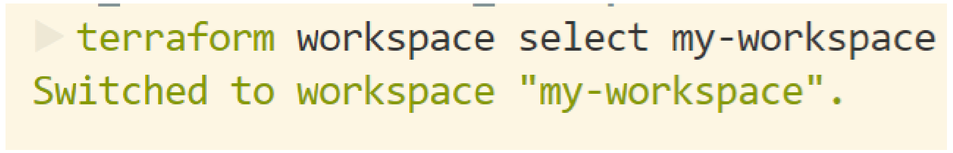
Workspace Management



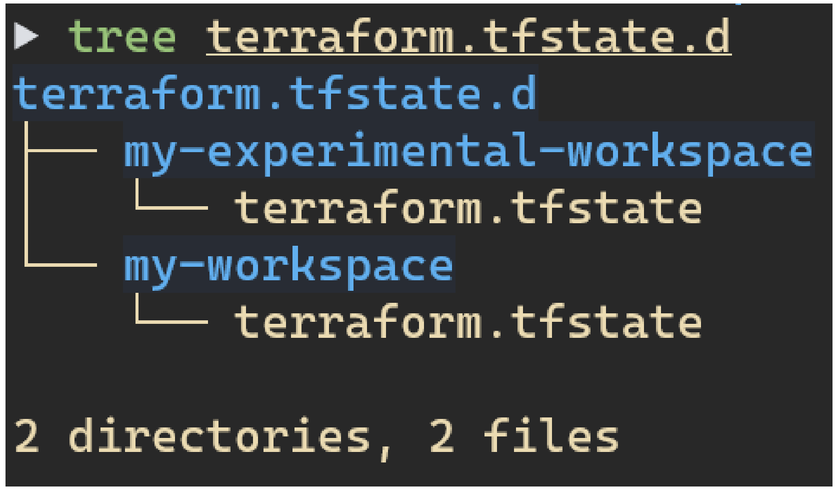
```
▶ terraform workspace new my-experimental-workspace
Created and switched to workspace "my-experimental-workspace"!

You're now on a new, empty workspace. Workspaces isolate their state,
so if you run "terraform plan" Terraform will not see any existing state
for this configuration.
```



```
▶ terraform workspace list
  default
* my-experimental-workspace
  my-workspace
```



```
▶ terraform workspace select my-workspace
Switched to workspace "my-workspace".
```



```
▶ tree terraform.tfstate.d
terraform.tfstate.d
├── my-experimental-workspace
│   └── terraform.tfstate
└── my-workspace
    └── terraform.tfstate

2 directories, 2 files
```

Provisioners

Provisioners are a Last Resort

Can be used to model specific actions on the local machine or on a remote machine in order to prepare servers or other infrastructure objects for service. (can also be used with *terraform_data)*

Introduced in terraform as a measure of pragmatism, knowing that there will always be certain behaviours that can't be directly represented in Terraform's declarative model.

Built-in Provisioners

Local-exec

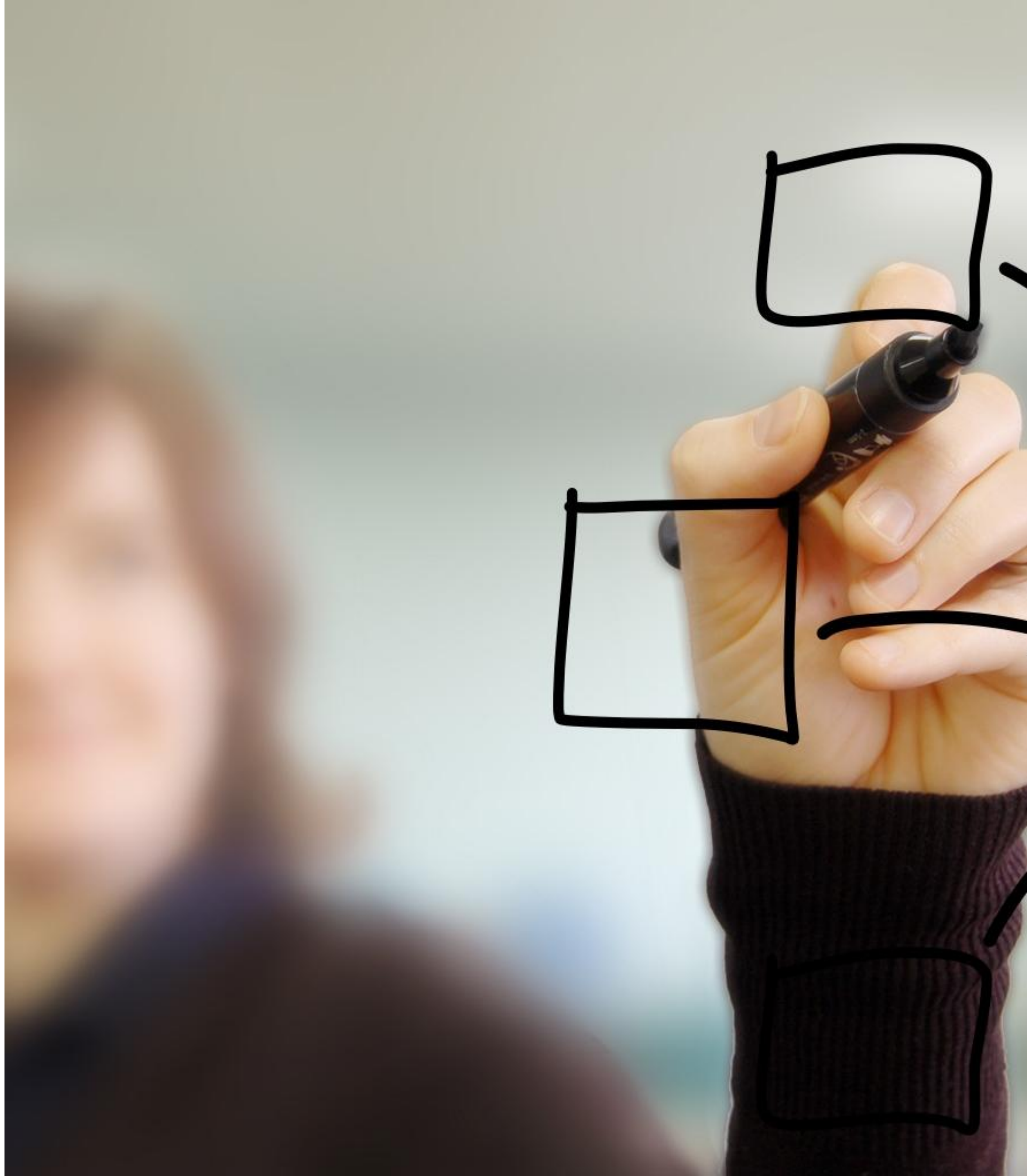
Remote-exec

Use Provisioners as a last resort as they add complexity and uncertainty.

Ask

Discuss

Comment



Microsoft