

# Importing resources Lab 1.9

Step through an example of importing an existing resource into Terraform.

## Overview

Another common scenario is importing resources that have been created manually. This is not a fully automated process so this section will guide you through the basics.

In this lab you will

1. Create a storage account resource
2. Check for a clean terraform plan
3. Add a minimal resource block
4. Import the resource
5. Configure until terraform plan is clean again
6. Apply the config with the `--refresh-only` switch

## Starting point

Your files should look similar to this:

- provider.tf
  - terraform {
  - required\_providers {
  - azurerm = {
  - source = "hashicorp/azurerm"
  - version = "~>3.1"
  - }
  - }
  - }
  - 
  - provider "azurerm" {
  - features {}
  - 
  - storage\_use\_azuread = true
  - }
- variables.tf
  - variable "resource\_group\_name" {
  - description = "Name for the resource group"
  - type = string
  - default = "terraform-basics"

```

• }
•
• variable "location" {
•   description = "Azure region"
•   type        = string
•   default     = "West Europe"
• }
•
• variable "container_group_name" {
•   description = "Name of the container group"
•   type        = string
•   default     = "terraform-basics"
• }
•

• main.tf

• locals {
•   uniq = substr(sha1(azurerm_resource_group.basics.id), 0, 8)
• }
•
• resource "azurerm_resource_group" "basics" {
•   name      = var.resource_group_name
•   location = var.location
•
•   lifecycle {
•     ignore_changes = [
•       tags,
•     ]
•   }
• }
•
• resource "azurerm_container_group" "basics" {
•   name                = var.container_group_name
•   location             = azurerm_resource_group.basics.location
•   resource_group_name = azurerm_resource_group.basics.name
•   ip_address_type     = "Public"
•   dns_name_label      = "${var.container_group_name}-${local.uniq}"
•   os_type             = "Linux"
•
•   container {
•     name   = "inspectorgadget"
•     image  = "jelledruyts/inspectorgadget:latest"
•     cpu    = "0.5"
•     memory = "1.0"
•
•     ports {
•       port      = 80
•       protocol = "TCP"
•     }
•   }
• }
•

• outputs.tf

```

- output "ip\_address" {
  - value = azurerm\_container\_group.basics.ip\_address
  - }
  - 
  - output "fqdn" {
  - value = "http://\${azurerm\_container\_group.basics.fqdn}"
  - }
  - terraform.tfvars
  - location = "UK South"
- You may have set a different value for *location*.

## Create the resource

Use the portal to create a storage account in the *terraform-basics* resource group.

1. [Create a storage account in the portal](#) (open in a new tab or window)
  - Basics tab
    - Select the correct subscription
    - Select the *terraform-basics* resource group
    - Create a valid and **unique storage account name**
    - Select the same region as the resource group, e.g. *UK South*
    - Leave the Performance as the default, *Standard*
    - Change redundancy to LRS
  - Advanced tab
    - Disable blob public access
      - Deselect *Allow enabling public access on containers*
    - Enable hierarchical namespace
    - Enable NFS v3
  - In the Networking tab
    - Disable public access and use private access

Ignore the additional steps to specify a virtual network or private endpoint. (The storage account will not be accessed in this lab.)
2. Click on *Review and create*

Check the config matching the requirements and validates.

[Home](#) >

## Create a storage account ...

Basics   Advanced   Networking   Data protection   Encryption   Tags   Review

Replication   Locally-redundant storage (LRS)

### Advanced

Secure transfer	Enabled
Allow storage account key access	Enabled
Allow cross-tenant replication	Disabled
Default to Azure Active Directory authorization in the Azure portal	Disabled
Blob public access	Disabled
Minimum TLS version	Version 1.2
Permitted scope for copy operations (preview)	From any storage account
Enable hierarchical namespace	Enabled
Enable network file system v3	Enabled
Access tier	Hot
Enable SFTP	Disabled
Large file shares	Disabled

### Networking

Network connectivity	Private endpoint
----------------------	------------------

---

[Create](#)   [< Previous](#)   [Next >](#)   [Download a template for automatic deployment](#)

3. Click on *Create*

Deployment should take a few seconds. Navigate to the resource once deployment has succeeded.

4. Click on *JSON View*
5. View the resource ID

## Resource JSON

richeney27182818

Resource ID

/subscriptions/9b7a166a-267f-45a5-b480-7a04cfc1edf6/resourcegroups/terraform-basics/providers/Microsoft.S

```
1  {
2    "sku": {
3      "name": "Standard_LRS",
4      "tier": "Standard"
5    },
6    "kind": "StorageV2",
7    "id": "/subscriptions/9b7a166a-267f-45a5-b480-7a04cfc1edf6/resourceGroup",
8    "name": "richeney27182818",
9    "type": "Microsoft.Storage/storageAccounts",
10   "location": "uksouth",
11   "tags": {},
12   "properties": {
13     "minimumTlsVersion": "TLS1_2",
14     "allowBlobPublicAccess": false,
15     "allowSharedKeyAccess": true,
16     "isHnsEnabled": true,
17     "networkAcls": {
18       "bypass": "AzureServices",
19       "virtualNetworkRules": [],
20       "ipRules": [],
21       "defaultAction": "Deny"
22     },
23     "supportsHttpsTrafficOnly": true,
24     "encryption": {
25       "services": {
26         "file": {
27           "enabled": true,
28           "lastEnabledTime": "2022-10-25T10:39:16.7094271Z"
29         },
30         "blob": {
31           "enabled": true
```

You will need the resource ID for the import command in the next section. However we'll set a variable using the CLIs and use that later in the lab.

#### 6. Set a variable for the resource ID

Use either Bash or PowerShell to set a variable with the storage account's resource ID.

Bash:

```
saId=$(az storage account list --resource-group terraform-basics --query "[0].id" --output tsv)
```

Powershell:

```
$saId = (Get-AzStorageAccount -ResourceGroupName terraform-basics)[0].id
```

This will set the variables to the resource ID of the first storage account found in the resource group.

## Check for no diff



If you get the output above then you can skip the next section and go straight to the [import](#).

## **Refresh state (*only if required*)**



```
}  
  
}
```

```
terraform apply -refresh-only
```

Unless you have made equivalent changes to your configuration, or ignored the relevant attributes using `ignore_changes`, the following plan may include.

factions to undo or respond to these changes.

---

**No changes.** Your infrastructure matches the configuration.

Your configuration already matches the changes detected above. If you'd like to update the Terraform state to match, create and apply a refresh-only plan:

1. Follow the advice and refresh the state file
2. `terraform apply --refresh-only`
3. Rerun `terraform plan` to confirm there is now no diff
4. `terraform plan`

Desired output:

```
azurerm_resource_group.basics: Refreshing state...  
[id=/subscriptions/9b7a166a-267f-45a5-b480-7a04cfc1edf6/resourceGroups/terraform-basics]  
  
azurerm_container_group.basics: Refreshing state...  
[id=/subscriptions/9b7a166a-267f-45a5-b480-7a04cfc1edf6/resourceGroups/terraform-basics/providers/Microsoft.ContainerInstance/containerGroups/terraform-basics]
```

**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.



# Import into state

Now that you have confirmed that there is no diff, you can create the resource block and import.

## 1. Create an empty resource block

Use the [Terraform azurerm docs](#) for [azurerm storage account](#) to get an example block.

```
resource "azurerm_storage_account" "import_example" {
  name                        = "storageaccountname"
  resource_group_name        = azurerm_resource_group.basics.name
  location                   = azurerm_resource_group.basics.location
  account_tier                = "Standard"
  account_replication_type   = "GRS"

  tags = {
    environment = "staging"
  }
}
```

- Copy the example into your main.tf

Note that the block above deviates from the docs page

- resource group references have been updated to *azurerm\_resource\_group.basics*.
- identifier label has been set to *import\_example*. You would usually set the identifier to your preferred name. Please keep it as *import\_example* for this lab.

- Set the name to your storage account's name

Don't worry that the other arguments do not match your created resource yet.

## 2. Import the resource

## 3. terraform import azurerm\_storage\_account.import\_example \$saId

Example output:

```
azurerm_storage_account.import_example: Importing from ID
"/subscriptions/9b7a166a-267f-45a5-b480-
7a04cfc1edf6/resourceGroups/terraform-
basics/providers/Microsoft.Storage/storageAccounts/pmw45665...
azurerm_storage_account.import_example: Import prepared!
Prepared azurerm_storage_account for import
```

```
azurerm_storage_account.import_example: Refreshing state...  
[id=/subscriptions/9b7a166a-267f-45a5-b480-  
7a04cfc1edf6/resourceGroups/terraform-  
basics/providers/Microsoft.Storage/storageAccounts/pmw45665]
```

Import successful!

The resources that were imported are shown above. These resources are now in

your Terraform state and will henceforth be managed by Terraform.

4. List the identifiers
5. terraform state list

Expected output:

```
azurerm_container_group.basics  
azurerm_resource_group.basics  
azurerm_storage_account.import_example
```

6. Show the imported config
7. terraform state show azurerm\_storage\_account.import\_example

Example output:

```
# azurerm_storage_account.import_example:  
  
resource "azurerm_storage_account" "import_example" {  
    access_tier                = "Hot"  
    account_kind               = "StorageV2"  
    account_replication_type   = "LRS"  
    account_tier               = "Standard"  
    allow_nested_items_to_be_public = false  
    cross_tenant_replication_enabled = false  
    enable_https_traffic_only    = true  
}
```

```
id = "/subscriptions/9b7a166a-267f-45a5-b480-7a04cfc1edf6/resourceGroups/terraform-basics/providers/Microsoft.Storage/storageAccounts/pmw45665"

infrastructure_encryption_enabled = false

is_hns_enabled = true

location = "uksouth"

min_tls_version = "TLS1_2"

name = "pmw45665"

nfsv3_enabled = true

primary_access_key = (sensitive value)

primary_blob_connection_string = (sensitive value)

primary_blob_endpoint =
"https://pmw45665.blob.core.windows.net/"

primary_blob_host = "pmw45665.blob.core.windows.net"

primary_connection_string = (sensitive value)

primary_dfs_endpoint =
"https://pmw45665.dfs.core.windows.net/"

primary_dfs_host = "pmw45665.dfs.core.windows.net"

primary_file_endpoint =
"https://pmw45665.file.core.windows.net/"

primary_file_host = "pmw45665.file.core.windows.net"

primary_location = "uksouth"

primary_queue_endpoint =
"https://pmw45665.queue.core.windows.net/"

primary_queue_host = "pmw45665.queue.core.windows.net"

primary_table_endpoint =
"https://pmw45665.table.core.windows.net/"

primary_table_host = "pmw45665.table.core.windows.net"

primary_web_endpoint =
"https://xxxxxxx.z33.web.core.windows.net/"

primary_web_host = "
xxxxxxx.z33.web.core.windows.net"

public_network_access_enabled = false

queue_encryption_key_type = "Service"
```

```
resource_group_name      = "terraform-basics"
secondary_access_key      = (sensitive value)
secondary_connection_string = (sensitive value)
shared_access_key_enabled = true
table_encryption_key_type = "Service"
tags                     = {}
```

```
blob_properties {
  change_feed_enabled      = false
  last_access_time_enabled = false
  versioning_enabled       = false
```

```
  delete_retention_policy {
    days = 7
  }
}
```

```
network_rules {
  bypass      = [
    "AzureServices",
  ]
  default_action = "Deny"
  ip_rules      = []
  virtual_network_subnet_ids = []
}
```

```
queue_properties {
```

```
  hour_metrics {
```

```
        enabled          = true

        include_apis     = true

        retention_policy_days = 7

        version          = "1.0"
    }

    logging {

        delete            = false

        read              = false

        retention_policy_days = 0

        version           = "1.0"

        write             = false
    }

    minute_metrics {

        enabled           = false

        include_apis      = false

        retention_policy_days = 0

        version           = "1.0"
    }
}

share_properties {

    retention_policy {

        days = 7
    }
}
```

```
    timeouts {}  
  }  
}
```

## Check the diff

OK, so the state looks good, but run a terraform plan and you'll see we have more to do.

1. Run a plan
2. terraform plan

You should see extensive output showing what Terraform plans to change.  
Example output:

```
azurerm_resource_group.basics: Refreshing state...  
[id=/subscriptions/9b7a166a-267f-45a5-b480-7a04cfc1edf6/resourceGroups/terraform-basics]  
  
azurerm_container_group.basics: Refreshing state...  
[id=/subscriptions/9b7a166a-267f-45a5-b480-7a04cfc1edf6/resourceGroups/terraform-basics/providers/Microsoft.ContainerInstance/containerGroups/terraform-basics]  
  
azurerm_storage_account.import_example: Refreshing state...  
[id=/subscriptions/9b7a166a-267f-45a5-b480-7a04cfc1edf6/resourceGroups/terraform-basics/providers/Microsoft.Storage/storageAccounts/pmw45665]
```

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:

**-/+** destroy and then create replacement

Terraform will perform the following actions:

```
# azurerm_storage_account.import_example must be replaced  
  
-/+ resource "azurerm_storage_account" "import_example" {  
    ~ access_tier                = "Hot" -> (known after apply)  
    ~ account_replication_type   = "LRS" -> "GRS"  
    ~ allow_nested_items_to_be_public = false -> true
```

```

~ cross_tenant_replication_enabled = false -> true

~ id = "/subscriptions/9b7a166a-267f-45a5-b480-7a04cfc1edf6/resourceGroups/terraform-basics/providers/Microsoft.Storage/storageAccounts/pmw45665 -> (known after apply)

~ is_hns_enabled = true -> false # forces replacement

+ large_file_share_enabled = (known after apply)

name = "pmw45665

~ nfsv3_enabled = true -> false # forces replacement

~ primary_access_key = (sensitive value)

~ primary_blob_connection_string = (sensitive value)

~ primary_blob_endpoint = "https://pmw45665.blob.core.windows.net/" -> (known after apply)

~ primary_blob_host = "pmw45665.blob.core.windows.net" -> (known after apply)

~ primary_connection_string = (sensitive value)

~ primary_dfs_endpoint = "https://pmw45665.dfs.core.windows.net/" -> (known after apply)

~ primary_dfs_host = "pmw45665.dfs.core.windows.net" -> (known after apply)

~ primary_file_endpoint = "https://pmw45665.file.core.windows.net/" -> (known after apply)

~ primary_file_host = "pmw45665.file.core.windows.net" -> (known after apply)

~ primary_location = "uksouth" -> (known after apply)

~ primary_queue_endpoint = "https://pmw45665.queue.core.windows.net/" -> (known after apply)

~ primary_queue_host = "pmw45665.queue.core.windows.net" -> (known after apply)

~ primary_table_endpoint = "https://pmw45665.table.core.windows.net/" -> (known after apply)

~ primary_table_host = "pmw45665.table.core.windows.net" -> (known after apply)

~ primary_web_endpoint = "https://pmw45665.z33.web.core.windows.net/" -> (known after apply)

```

```

~ primary_web_host =
"pmw45665.z33.web.core.windows.net" -> (known after apply)

~ public_network_access_enabled = false -> true

~ secondary_access_key = (sensitive value)

+ secondary_blob_connection_string = (sensitive value)

+ secondary_blob_endpoint = (known after apply)

+ secondary_blob_host = (known after apply)

~ secondary_connection_string = (sensitive value)

+ secondary_dfs_endpoint = (known after apply)

+ secondary_dfs_host = (known after apply)

+ secondary_file_endpoint = (known after apply)

+ secondary_file_host = (known after apply)

+ secondary_location = (known after apply)

+ secondary_queue_endpoint = (known after apply)

+ secondary_queue_host = (known after apply)

+ secondary_table_endpoint = (known after apply)

+ secondary_table_host = (known after apply)

+ secondary_web_endpoint = (known after apply)

+ secondary_web_host = (known after apply)

~ tags = {
  + "environment" = "staging"
}

# (9 unchanged attributes hidden)

~ blob_properties {
  ~ change_feed_enabled = false -> (known after apply)
  + default_service_version = (known after apply)
  ~ last_access_time_enabled = false -> (known after apply)
  ~ versioning_enabled = false -> (known after apply)
}

```



```
+ container_delete_retention_policy {  
    + days = (known after apply)  
}  
  
+ cors_rule {  
    + allowed_headers    = (known after apply)  
    + allowed_methods    = (known after apply)  
    + allowed_origins    = (known after apply)  
    + exposed_headers    = (known after apply)  
    + max_age_in_seconds = (known after apply)  
}  
  
~ delete_retention_policy {  
    ~ days = 7 -> (known after apply)  
}  
  
}  
  
~ network_rules {  
    ~ bypass = [  
        - "AzureServices",  
    ] -> (known after apply)  
    ~ default_action = "Deny" -> (known after apply)  
    ~ ip_rules       = [] -> (known after apply)  
    ~ virtual_network_subnet_ids = [] -> (known after apply)  
  
    + private_link_access {  
        + endpoint_resource_id = (known after apply)  
        + endpoint_tenant_id   = (known after apply)  
    }  
}
```

```
}
```

```
~ queue_properties {
```

```
  + cors_rule {
```

```
    + allowed_headers    = (known after apply)
```

```
    + allowed_methods    = (known after apply)
```

```
    + allowed_origins    = (known after apply)
```

```
    + exposed_headers    = (known after apply)
```

```
    + max_age_in_seconds = (known after apply)
```

```
  }
```

```
~ hour_metrics {
```

```
  ~ enabled                = true -> (known after apply)
```

```
  ~ include_apis           = true -> (known after apply)
```

```
  ~ retention_policy_days = 7 -> (known after apply)
```

```
  ~ version                 = "1.0" -> (known after apply)
```

```
}
```

```
~ logging {
```

```
  ~ delete                = false -> (known after apply)
```

```
  ~ read                  = false -> (known after apply)
```

```
  ~ retention_policy_days = 0 -> (known after apply)
```

```
  ~ version                 = "1.0" -> (known after apply)
```

```
  ~ write                  = false -> (known after apply)
```

```
}
```

```
~ minute_metrics {
```

```
  ~ enabled                = false -> (known after apply)
```

```
  ~ include_apis           = false -> (known after apply)
```

```

    ~ retention_policy_days = 0 -> (known after apply)

    ~ version                = "1.0" -> (known after apply)
  }
}

+ routing {
  + choice                  = (known after apply)
  + publish_internet_endpoints = (known after apply)
  + publish_microsoft_endpoints = (known after apply)
}

~ share_properties {
  + cors_rule {
    + allowed_headers    = (known after apply)
    + allowed_methods    = (known after apply)
    + allowed_origins    = (known after apply)
    + exposed_headers    = (known after apply)
    + max_age_in_seconds = (known after apply)
  }

  ~ retention_policy {
    ~ days = 7 -> (known after apply)
  }

  + smb {
    + authentication_types          = (known after apply)
    + channel_encryption_type       = (known after apply)
    + kerberos_ticket_encryption_type = (known after apply)
    + versions                     = (known after apply)
  }
}

```

```
    }  
  }  
  
  - timeouts {}  
}
```

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

Ouch! OK, let's work through this.

### 3. Identify the required config updates

The good news is that it is pretty quick to deconstruct the output and work out what is important, and experience helps.

First of all, ignore those lines that include (known after apply). The only ones that we need to pay attention to are those which have changes, deletes or adds where the target state is shown as a specific value such as a literal string or boolean.

The output below has been manually truncated to help you to focus on what is important. Update the config files with the correct arguments and you should eventually get to a clean plan with no diff.

Terraform will perform the following actions:

```
# azurerm_storage_account.import_example must be replaced  
-/+ resource "azurerm_storage_account" "import_example" {  
    ~ account_replication_type      = "LRS" -> "GRS"  
    ~ allow_nested_items_to_be_public = false -> true  
    ~ cross_tenant_replication_enabled = false -> true
```

```

    ~ is_hns_enabled          = true -> false # forces
replacement
    ~ nfsv3_enabled          = true -> false # forces
replacement
    ~ public_network_access_enabled = false -> true

    ~ tags                    = {
      + "environment" = "staging"
    }
  }
}

```

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

That is a more manageable set. Let's get to work.

## Update the config files

1. Update the replication type

The plan included:

```

    ~ account_replication_type = "LRS" -> "GRS"

```

Update account\_replication\_type string value to *LRS*.

```

account_replication_type = "LRS"

```

2. Add the public blob access boolean

The plan included:

```

    ~ allow_nested_items_to_be_public = false -> true

```

Add the `allow_blob_public_access` argument and set the boolean value to false. (Default is true.)

```
allow_nested_items_to_be_public = false
```

### 3. Check on progress

Run a diff.

```
terraform plan
```

You should see that those two changes are no longer planned. Making progress!

### 4. Update the main.tf to match

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.

References:

- the [azurerm\\_storage\\_account](#) documentation page
- the `terraform state show azurerm_storage_account.import_example` output

Check on your progress by periodically saving the file and rerunning `terraform plan`. (You may notice that `terraform plan` also validates the files first.)

If you get stuck then a working config is shown at the start of the next lab.

## 5. Check for no diff

Confirm that *terraform plan* is clean

```
terraform plan
```

Example output:

```
azurerm_resource_group.basics: Refreshing state...
[id=/subscriptions/9b7a166a-267f-45a5-b480-7a04cfc1edf6/resourceGroups/terraform-basics]

azurerm_container_group.basics: Refreshing state...
[id=/subscriptions/9b7a166a-267f-45a5-b480-7a04cfc1edf6/resourceGroups/terraform-basics/providers/Microsoft.ContainerInstance/containerGroups/terraform-basics]

azurerm_storage_account.import_example: Refreshing state...
[id=/subscriptions/9b7a166a-267f-45a5-b480-7a04cfc1edf6/resourceGroups/terraform-basics/providers/Microsoft.Storage/storageAccounts/pmw45665]
```

**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.

## 6. Format the files

Check that the formatting is as it should be.

```
terraform fmt
```

Example output:

```
main.tf
```

## Summary

Importing resources is a little messy, but is a useful skill to have as a Terraform admin.

It can be a useful way to add in the config for complex high value resources. For example, the documentation for Azure Application Gateway is difficult to decipher given the range of options and possible configuration. You may find it simpler to provision the resource using the portal and then import the config.

The good news is that Microsoft employees have released a preview of Azure Terraform (aztfy) as per this [blog post](#).

In the next lab we will destroy the config and tidy up.