Adding resources Lab1.5

Use the azurerm documentation to add a resource to your configuration.

Overview

In this lab you will

- browse the aka.ms/terraform documentation
- add a resource to create an Azure Container Instance
- plan and apply the change

Starting point

Your files should still look like 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"
}
```

- main.tf

```
resource "azurerm_resource_group" "basics" {
  name     = var.resource_group_name
  location = var.location
}
```

- terraform.tfvars

```
location = "UK South"
```

You may have set a different value for *location*.

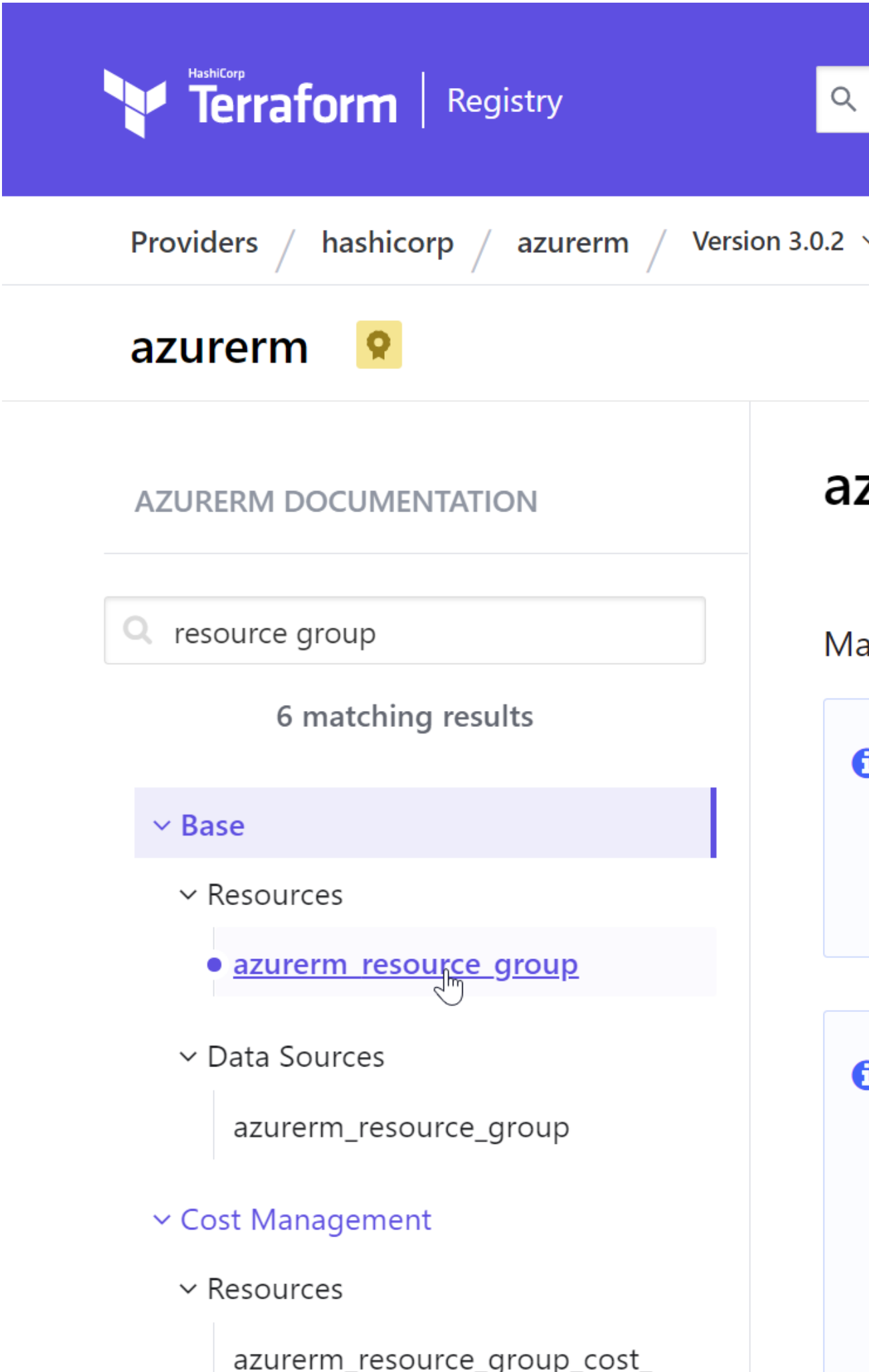
Azure Resource Group

There is a short URL for the azure provider's resource documentation.

1. Browse to aka.ms/terraform in a new tab

   Commit this short URL to memory! You'll be using it often.

   Note the azurerm version drop down at the top of the page. Default is the latest version.


HashiCorp
Terraform
Registry
Providers / hashicorp / azurerm / Version 3.0.2
azurerm
AZURERM DOCUMENTATION
resource group
6 matching results
Base
Resources
azurerm_resource_group
Data Sources
azurerm_resource_group
Cost Management
Resources
azurerm_resource_group_cost_

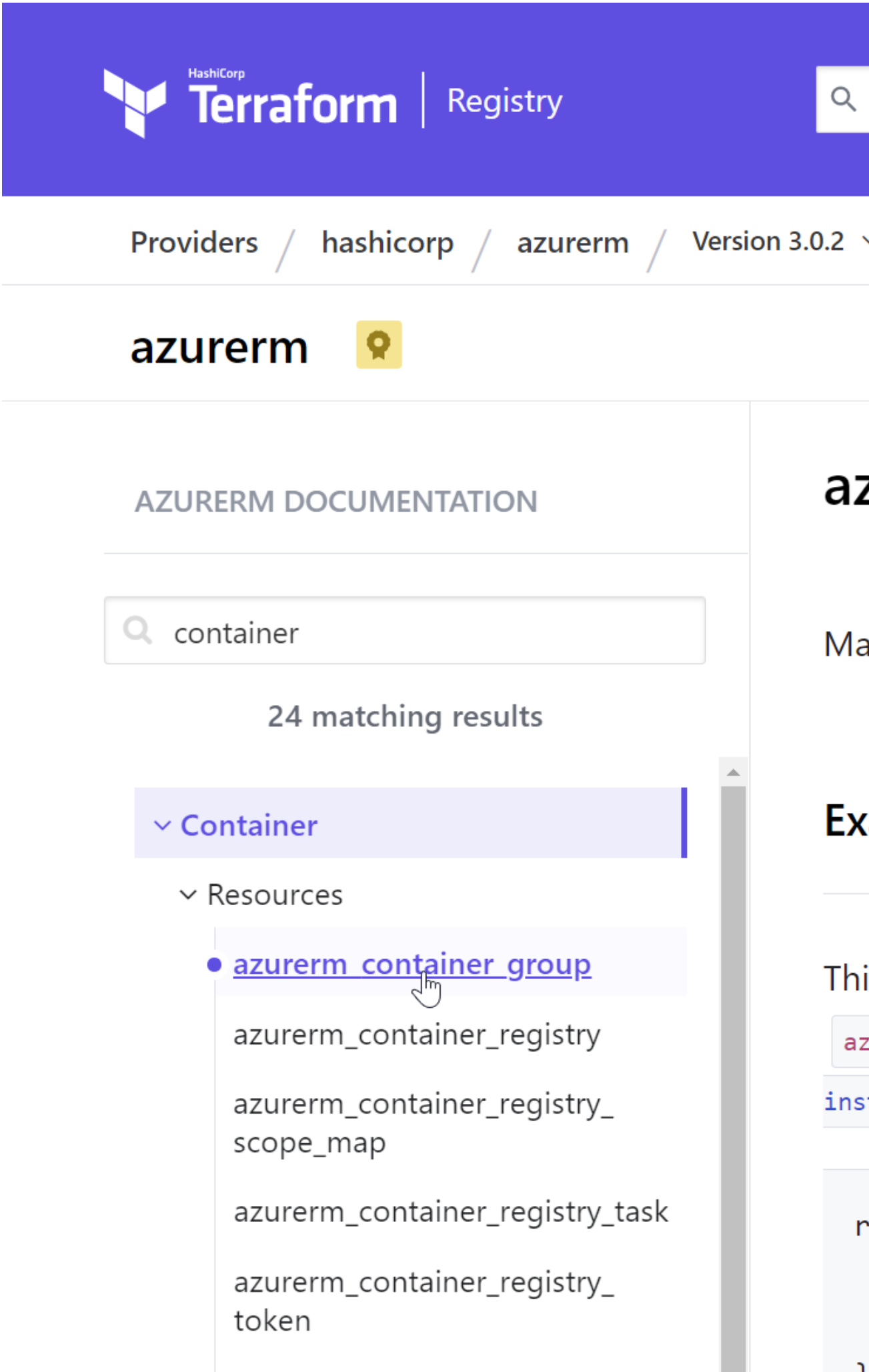
This is the documentation page for the resource group you've already created.

Each resource page shows an example, plus the arguments (*name*, *location*, *tags*) and the exported attributes (all arguments plus *id*).

Azure Container Instance

In the next section you will add a new azurerm_container_group resource to create an Azure Container Instance. Look at the documentation page for the resource.

1. Filter to "container" on the aka.ms/terraform page
2. Select the **azurerm_container_group** resource


HashiCorp
Terraform
Registry
Providers / hashicorp / azurerm / Version 3.0.2
azurerm
AZURERM DOCUMENTATION
container
24 matching results
Container
Resources
azurerm_container_group
azurerm_container_registry
azurerm_container_registry_scope_map
azurerm_container_registry_task
azurerm_container_registry_token

This is a more complex resource with a larger number of arguments and attributes.

Here is the Example Usage from the page:

```
resource "azurerm_resource_group" "example" {
  name     = "example-resources"
  location = "West Europe"
}

resource "azurerm_container_group" "example" {
  name                = "example-continst"
  location            = azurerm_resource_group.example.location
  resource_group_name = azurerm_resource_group.example.name
  ip_address_type     = "public"
  dns_name_label      = "aci-label"
  os_type             = "Linux"

  container {
    name   = "hello-world"
    image  = "mcr.microsoft.com/azuredocs/aci-helloworld:latest"
    cpu    = "0.5"
    memory = "1.5"

    ports {
      port     = 443
      protocol = "TCP"
    }
  }

  container {
    name   = "sidecar"
    image  = "mcr.microsoft.com/azuredocs/aci-tutorial-sidecar"
    cpu    = "0.5"
    memory = "1.5"
  }

  tags = {
    environment = "testing"
  }
}
```

Note that as well as top level arguments (name, location, os_type etc.) that there are also blocks. This example has container blocks. Some blocks support multiples. (The example container group contains two containers.) There can be blocks within blocks, such as the ports block within the first container block.

The documentation shows which arguments are required and optional within each block type, and detail on the permitted values.

In the example it uses `azurerm_resource_group.example.location` as the location value, rather than `"UK South"` or `var.location`.

This creates an implicit dependency between azurerm_container_group.example and azurerm_resource_group.example.

These become linked nodes in the graph that Terraform generates to understand the order of operations, parallelism, dependencies etc.

You can also define explicit dependencies using the **`depends_on = []`** meta-argument.

Challenge

This would use the "bare" format, i.e. `var.container_group_name`

- The DNS label should concatenate the *prefix* and the *container_group_name*

  The interpolation format is `"${var.prefix}-${var.container_group_name}"`. Interpolation allows us to generate more complex expressions. Terraform knows to evaluate the variables surrounded with `${}` to get their values.

- Use the Inspector Gadget image for the container

  - name: `inspectorgadget`
  - image: `"jelledruyts/inspectorgadget:latest"`
    The Inspector Gadget image should be the only container in the container group.

  - Reduce the memory requirement to 1GB

  - Set the container port to 80 (HTTP)

- No tags

Don't forget to save all of your files.

2. Open a new tab in the browser and go to the site

Request

URL	http://terra
HTTP Method	GET
Is HTTPS	False
Request ID	0HMGNRA
Client Certificate Serial Number	
Local IP Address	::ffff:192.16
Local Port	80
Remote IP Address	::ffff:10.92.
Remote Port	61877

HTTP Headers

Accept	text/html,appl q=0.9
Accept-Encoding	gzip, deflate
Accept-Language	en-GB,en;q=0.
Connection	keep-alive

If your screen is similar to the one above then you have been successful! If not then check if your config differs from the one shown at the start of the next lab. (Note that the value of your prefix should be different.)

Summary

You successfully updated the config and added a resource.

Being able to navigate the documentation is a key skill when you are adding and modifying resources. You will also need to have a good working understanding of Azure, how the resources work and relate to each other.

If you find the documentation is a little vague on an argument and its possible values then check the documentation for the Azure Resource Manager REST APIs as they can sometimes add insight where the resources closely match the properties found in those REST API calls.

In the next lab we will use locals and add an output.
