Chmurowisko

Szkolenie Terraform: Dzień 2 Dodatkowe zadania

1. Zadanie: Ustawienia diagnostyczne

W tym zadaniu wykorzystaj kod utworzonych zasobów podczas zadań z dnia 2.

Na początku należy utworzyć Log Analytics Workspace.

Następnie należy włączyć wysyłanie danych diagnostycznych z Azure Firewall do Log Analytics Workspace wykorzystując zasób Diagnostic Settings. Przy pomocy data „monitor_diagnostic_categories" możesz pobrać dostępne kategorie metryk i logów dla Azure Firewall.

Przy tworzeniu Diagnostic Settings wykorzystaj blok dynamiczny do zdefiniowania kategorii logów. Blok dynamiczny może korzystać z zmiennej wykorzystaj tutaj mapę obiektów, gdzie zdefiniujesz w pojedynczym obiekcie nazwę kategorii oraz to czy jest ona włączona.

Przydatne linki:
Terraform dokumentacja diagnostic settings
https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/monitor_diagnostic_setting

Terraform dokumentacja data monitor categories
https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/data-sources/monitor_diagnostic_categories

Terraform dokumentacja dynamic blocks
https://www.terraform.io/language/expressions/dynamic-blocks

2. Zadanie: Powołanie Azure Policy

Należy powołać definicje dwóch polityk, a następnie przypisać je do własnej Resource Group'y. W końcu należy przetestować działanie polityk np. poprzez utworzenie testowego zasobu.

Jedną politykę przygotuj tylko z wykorzystaniem kodu terraform, w drugiej polityce wykorzystaj plik .json przy pomocy funkcji file(), jsondecode(), jsonencode().

Należy stworzyć 2 polityki:

- Polityka z efektem Deny, która będzie wymuszać tworzenie zasobów w sprecyzowanych przez nas lokalizacjach np. westeurope i northeurope https://github.com/Azure/azure-policy/blob/master/built-inpolicies/policyDefinitions/General/AllowedLocations_Deny.json
- Polityka z efektem Deny, która będzie zezwalać na utworzenie dozwolonych rozmiarów Maszyn Wirtualnych https://github.com/Azure/azure-policy/blob/master/built-inpolicies/policyDefinitions/Compute/VMSkusAllowed_Deny.json

Zasoby które należy utworzyć (w kolejności):

- Policy definition
- Policy assignment
- Resource testowy

Przydatne linki:
Terraform dokumentacja Policy Definition
https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/policy_definition

Terraform dokumentacja Resource Group Policy Assigment
https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/resource_group_policy_assignment

Terraform dokumentacja funkcji file()
https://www.terraform.io/language/functions/file

Terraform dokumentacja funkcji jsonencode()
https://www.terraform.io/language/functions/jsonencode

Terraform dokumentacja funkcji jsondecode()
https://www.terraform.io/language/functions/jsondecode

3. Zadanie: Provisioners

W tym zadaniu wypróbujesz działanie provisioners w Terraform, poznasz działanie file, local-exec i remote-exec provisioner. Każdy provisioner zdefiniuj w zasobie wirtualnej maszyny z publicznym adresem IP utworzonej w ramach zadania 2 z zjazdu 2.

Provisioner file – Utwórz plik z wiadomością .md, która chcesz przesłać na maszynę wirtualną po utworzeniu i skonfiguruj provisioner tak, by przesłać go na maszynę.

Provisioner remote-exec – Przy jego pomocy spróbuj zainstalować NGINX na maszynie wirtualnej.

Provisioner local-exec – Wykorzystaj go tak, by zapisać w pliku na lokalnej maszynie publiczny adres IP utworzonej maszyny.

Przydatne linki:
Terraform dokumentacja provisioners
https://www.terraform.io/language/resources/provisioners/syntax

Terraform dokumetnacja provisioner file
https://www.terraform.io/language/resources/provisioners/file

Terraform dokumentacja provisioner local-exec
https://www.terraform.io/language/resources/provisioners/local-exec

Terraform dokumentacja provisoner remote-exec
https://www.terraform.io/language/resources/provisioners/remote-exec