QATIP Intermediate
Azure Lab 06
Advanced Terraform Expressions, Functions, and Regular Expressions

Contents

Lab Objectives

This multi-challenge lab integrates expressions, dynamic blocks, Terraform functions, and regular expressions to create flexible and reusable infrastructure configurations in Azure. By the end of this lab, you will:

- Implement conditional expressions to control resource creation.
- Leverage string, numeric, and date/time functions to standardize resource naming, tagging, and configuration.

- Utilize regular expressions (regex) to standardize inconsistent resource names.
- Validate outputs using terraform console.

Before you begin

- Ensure you have completed Lab0 before attempting this lab.
- In the IDE terminal pane, enter the following command...

  cd c:\azure-tf-int\lab\06
- There is a sub-directory for each of the following challenges. Navigate to the appropriate folder as you attempt each challenge. Ensure all file changes and commands are executed in the appropriate challenge sub-directory

Challenge 1: Creating a Resource Group with Dynamic Naming and Tagging

Scenario

You are tasked to create a reusable configuration that..

- Creates a resource group with a name within the range RG1 to RG6.
- Adds an **Environment** tag that is always upper-case and restricted to DEV, PROD, or TEST.
- Add a second tag, **UpdatedOn** with the date of resource group creation/update in the format YYYY-MM-DD.
- terraform.tfvars is to be used with default values for the resource group name and environment.
- Validation check should ensure that the resource group name proposed is within the specified range “RG1” through to “RG6” and that the environment is “DEV”, “PROD” or “TEST”.

Goal: Define Terraform Configuration for Resource Group

Updating the files provided in **c:\azure-tf-int\labs\06\challenge1**, attempt to create a deployment that complies with the given scenario.

Validation Checks

To validate your configuration, run **terraform plan** operations and review the output to ensure the resource group name, Environment and UpdatedOn tags are compliant with the requirements specified.

Run tests with invalid parameters by updating terraform.tfvars with “RG8” as the resource group name and “plan” as the environment

This should result in error messages being output

Run tests with valid parameters by updating terraform.tfvars with “RG1” as the resource group name and “dev” as the environment

This should return a success message with the environment being converted to uppercase

Hints

- Use format() and upper() for consistency.
- Use timestamp() to capture creation date and time
- Use formatdate() to generate the creation date tag.
- Use contains() to validate the resource group name

Proposed Solution

A proposed solution to this challenge can be found at c:\azure-tf-in\labs\solutions\06\challenge1

Post Challenge Clean-up

Use **terraform destroy** to remove any resource created

Challenge 2: Standardizing Resource Names Using Regular Expressions

Scenario

You’ve been assigned to the Cloud Infrastructure team at a company migrating virtual machines (VMs) from multiple departments to Azure. Each department has used its own naming convention, leading to inconsistent VM naming such as:

`dev-finance.db01` (stage-project.name)

`infra_prod_appserver` (project_stage_name)

`sales-test_db02` (project-stage_name)

`infra_prod_web01` (project_stage_name)

`test.backup.storage03` (stage.project.name)

The business has mandated a standardized naming convention for all cloud hosted VMs: **[environment]-[project]-[name]** *(e.g., prod-finance-db01)*

Goal: Load JSON File and Process Data using Regex

Updating the files provided in **c:\azure-tf-int\labs\06\challenge2**, attempt to create a deployment that complies with the given scenario

Use Terraform functions to process a JSON file containing inconsistent virtual machine names. Reconstruct the names in accordance with the mandated naming convention using Regex.

- Load the JSON file **resource-names.json** using jsondecode()
- Construct regex patterns dynamically based on known stages (dev, prod and test), and projects (finance, sales, infra and backup)
- Extract the environment, project, and machine name from the json file using regex()
- Generate new names in accordance with the naming convention
- Generate screen output listing the original names and the converted names:

```
+ "dev-finance.db01"      = "dev-finance-db01"
+ infra_prod_appserver    = "prod-infra-appserver"
+ infra_prod_web01        = "prod-infra-web01"
+ sales-test_db02         = "test-sales-db02"
+ "test.backup.storage03" = "test-backup-storage03"
```

- There is no requirement to create these machines at this stage

Hints

- Use jsondecode(file("./resource-names.json")) to read the JSON file

- Use join("|", {stages/projects}) logic to create a regex pattern for stages and projects to find matches against
- Use try(regex(pattern, name), "default_value") to avoid errors on missing matches

Solution

A proposed solution for this challenge can be found at c:\azure-tf-int\labs\solutions\06\challenge2