

QATIP Intermediate

AWS Lab 06

Advanced Terraform Expressions, Functions, and Regular Expressions

Lab Objectives

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

- Implement conditional expressions to control resource creation.
- Leverage string, numeric, and date/time functions to standardize S3 bucket naming, tagging, and configuration.
- Utilize regular expressions (regex) to standardize inconsistent AWS 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 /aws-tf-int/labs/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 an S3 Bucket with Dynamic Naming and Tagging

Scenario

You are tasked with creating a reusable Terraform configuration that:

- Creates an S3 bucket with a name within the range BUCKET1 to BUCKET6.
- Adds a mandatory **Environment** tag that is always uppercase and restricted to **DEV**, **PROD**, or **TEST**.
- Adds a second tag, **UpdatedOn**, with the creation date in the format **YYYY-MM-DD**.
- Uses Terraform variables with default values for the S3 bucket name and environment.
- Implements validation checks to ensure:
 - The S3 bucket name is within the specified range.
 - The environment value is restricted to DEV, PROD, or TEST.

Goal: Define Terraform Configuration for S3 Bucket

Updating the files provided in **/aws-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**

Review the output to ensure that the S3 bucket name, Environment tag, and UpdatedOn tag meet the required constraints.

Run tests with invalid parameters:

```
terraform plan -var="s3_bucket_name=BUCKET8" -var="environment=Plan"
```

Expected error output:

Error: Bucket name must be BUCKET1 through BUCKET6.

Error: Environment must be one of DEV, PROD, or TEST.

Run tests with valid parameters

```
terraform plan -var="s3_bucket_name=BUCKET1" -var="environment=dev"
```

Expected success message:

The environment should be converted to uppercase (DEV).

Hints

- Use `format()` and `upper()` for consistency.
- Use `timestamp()` to capture creation date/time.
- Use `formatdate()` to generate the `UpdatedOn` tag.
- Use `contains()` to validate the S3 bucket name.

Proposed Solution

A proposed solution to this challenge can be found at [/aws-tf-int/labs/solutions/o6/challenge1](#)

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 AWS. Each department has used its own naming convention, leading to inconsistent VM naming such as:

| | |
|--------------------------------------|-----------------------------------|
| <code>`dev-finance.db01`</code> | <code>(stage-project.name)</code> |
| <code>`infra_prod_appserver`</code> | <code>(project_stage_name)</code> |
| <code>`sales-test_db02`</code> | <code>(project-stage_name)</code> |
| <code>`infra_prod_web01`</code> | <code>(project_stage_name)</code> |
| <code>`test.backup.storage03`</code> | <code>(stage.project.name)</code> |

The business has mandated a standardized naming convention for all AWS EC2 instances: **[environment]-[project]-[name]** (e.g., *prod-finance-db01*)

Goal: Load JSON File and Process Data using Regex

Updating the files provided in [/was-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 and projects
- Extract the environment, project, and machine name 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 [/was-tf-int/labs/06/challenge2](#)