Terraform documentation

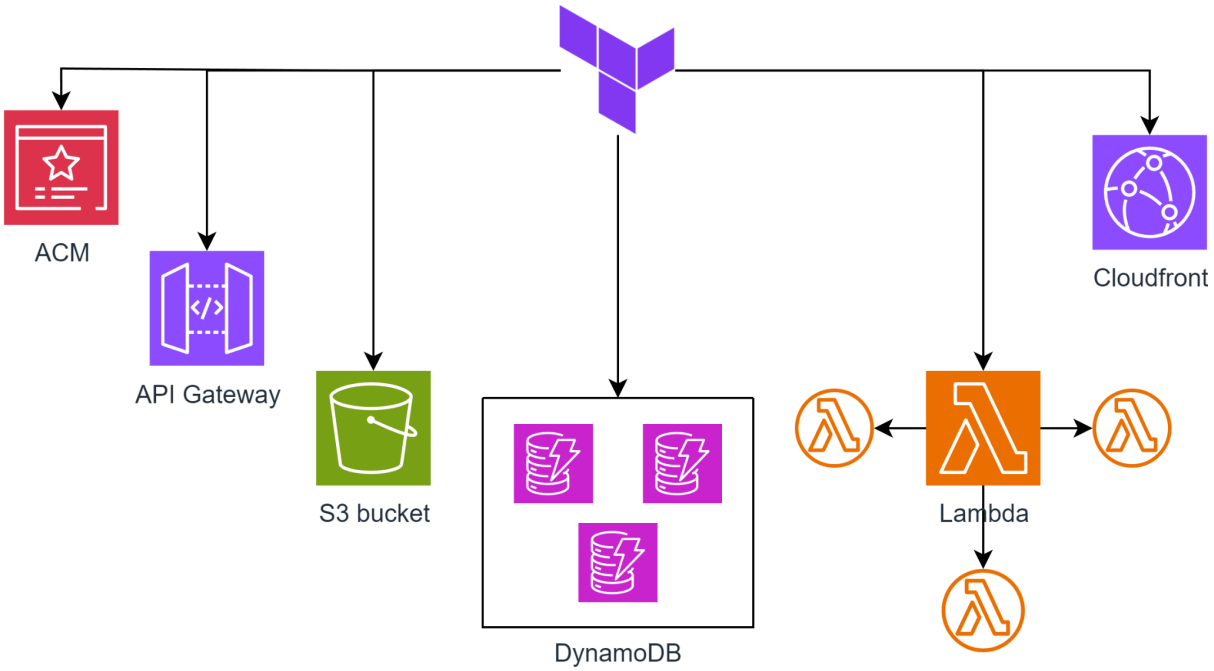


Table of contents

1. Introduction

This document provides a comprehensive guide to the infrastructure configuration for this project, created using Terraform. It will walk you through each part of the code, explaining the structure, purpose, and functionality of each component. Additionally, this documentation outlines the reasoning behind choosing Terraform as the Infrastructure as Code (IaC) tool for this project, highlighting the benefits of automation, repeatability, and cost-efficiency. By the end, you will have a clear understanding of how the infrastructure supports the application, as well as insights into the advantages of using Terraform to streamline deployment and management.

2. Story

Building an application does not end at writing code, it needs a reliable infrastructure to run on. This infrastructure can vary: it might be a personal computer, a dedicated server, a Raspberry Pi, or scalable resources in the cloud. AWS offers a powerful services to create the ideal infrastructure for any project. But when you are developing on AWS, you often find yourself in a repetitive cycle. After setting up resources for development or testing, you might need to tear it all down to save on costs. The next day, you repeat the process, manually recreating everything from scratch in the AWS Console, only to delete it again when you are done.

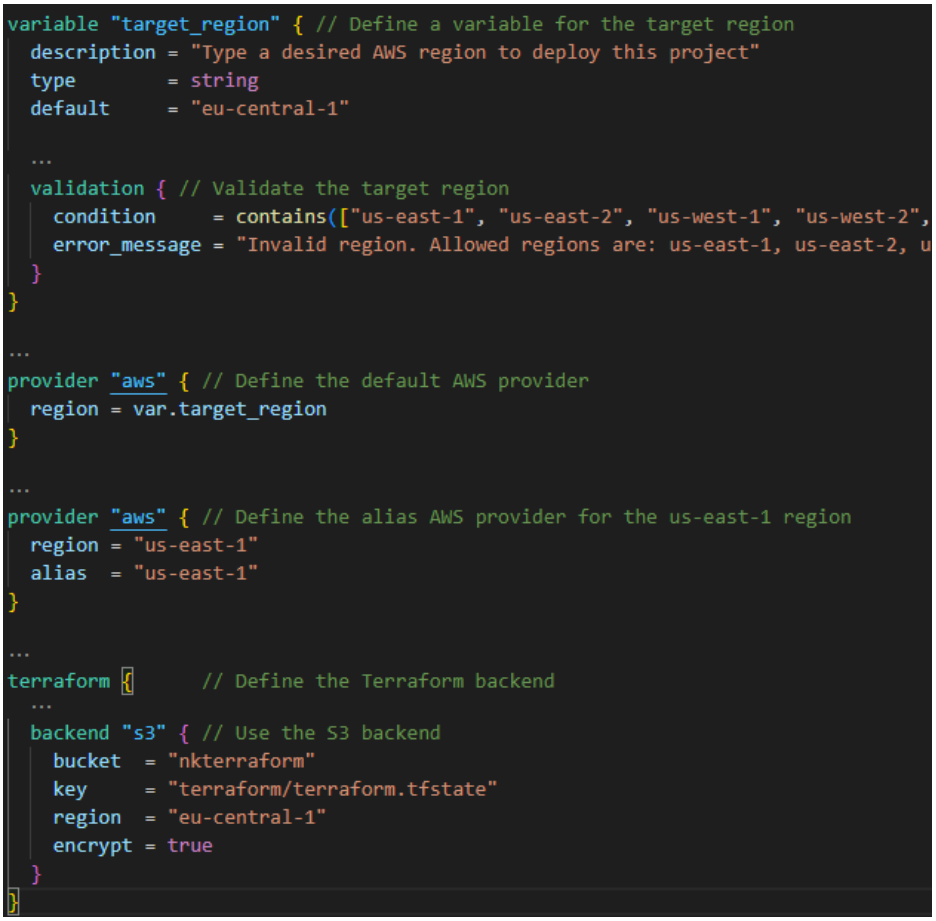
This back-and-forth is time-consuming, error-prone, and limits productivity. Here’s where Infrastructure as Code (IaC) tools like CloudFormation or Terraform come into play. With IaC, you can script your entire infrastructure setup, from provisioning servers to configuring networks and databases, all in a way that is reusable and repeatable. Instead of manually clicking through the AWS Console, you deploy your infrastructure with a single command.

In this project, I used Terraform to fully automate the deployment of my infrastructure. Not only does it save time by streamlining the setup and teardown process, but it also reduces human error and optimizes costs. When I am not developing or testing, Terraform allows me to bring down resources seamlessly. By embracing Infrastructure as Code, I have made my development process more efficient, predictable, and resilient – allowing me to focus on the application itself instead of the overhead of infrastructure management.

3. Design

Here I will explain every code line what it does and why I need it for my project.

- **provider.tf** - to set up an AWS provider and a parameter for the user to fill in the desired region.
    - **target_region variable** - to set up a target AWS region to deploy the infrastructure to users liking. It has a default value of “eu-central-1” and the possibility of using any other major AWS region.
    - **Two AWS providers** - for necessary packages that are required for automatic deployment in the AWS. One is created in eu-central-1 as default and the second one is created as an alias in us-east-1 (for CloudFront and TSL certificates).
    - **Backend in s3** - to store state files in the s3 bucket instead of locally. It enables working in multiple working environments without all-time sharing the state file.



```
variable "target_region" { // Define a variable for the target region
  description = "Type a desired AWS region to deploy this project"
  type        = string
  default     = "eu-central-1"

  ...
  validation { // Validate the target region
    condition     = contains(["us-east-1", "us-east-2", "us-west-1", "us-west-2",
    error_message = "Invalid region. Allowed regions are: us-east-1, us-east-2, u
  }
}

...
provider "aws" { // Define the default AWS provider
  region = var.target_region
}

...
provider "aws" { // Define the alias AWS provider for the us-east-1 region
  region = "us-east-1"
  alias  = "us-east-1"
}

...
terraform {      // Define the Terraform backend
  ...
  backend "s3" { // Use the S3 backend
    bucket  = "nkterraform"
    key     = "terraform/terraform.tfstate"
    region  = "eu-central-1"
    encrypt = true
  }
}
```

First is the target_region variable that you pass to Terraform. Second and third are the AWS providers to set up the whole environment to use AWS resources in specific region. Third is setting state file remotely in the s3 bucket in the eu-central-1 region with encryption.

- **s3.tf** - to set up the S3 bucket as primary storage and configure it for website hosting.
    - **s3b** - the creation of an S3 bucket with the name of my custom domain “nknez.tech” and the force destroy option to avoid problems with s3 bucket deletion.
    - **s3_cors** - to set up a basic CORS setting so everything is easily accessible inside the s3 bucket.
    - **s3b_enable_public_access** - to disable “Block public access” to make sure you can host a static website inside the s3 bucket.
    - **website_s3b** - enable static website mode inside the bucket and pair index.html with index and error.html with error values.
    - **bucket_policy_for_website** - setting up a bucket policy to allow GET requests for objects inside the s3 bucket.

```
resource "aws_s3_bucket" "s3b" {    // Defining the S3 bucket
  bucket        = local.domain_name // My own personal domain for testing
  force_destroy = true
}

You, 5 days ago | 2 authors (You and one other)
resource "aws_s3_bucket_cors_configuration" "s3_cors" { // Defining the CORS configuration for the S3 bucket
  bucket = aws_s3_bucket.s3b.id
  You, 5 days ago | 1 author (You)
  cors_rule {
    allowed_methods = ["GET"]
    allowed_origins = ["https://${local.domain_name}"]
  }      You, 3 months ago • Finished doing the testing enviroment for the A…
}

Norbert Knez, 2 months ago | 2 authors (You and one other)
resource "aws_s3_bucket_public_access_block" "s3b_enable_public_access" { // Disabling public access block
  bucket = aws_s3_bucket.s3b.id

  block_public_acls   = false
  block_public_policy = false
}
```

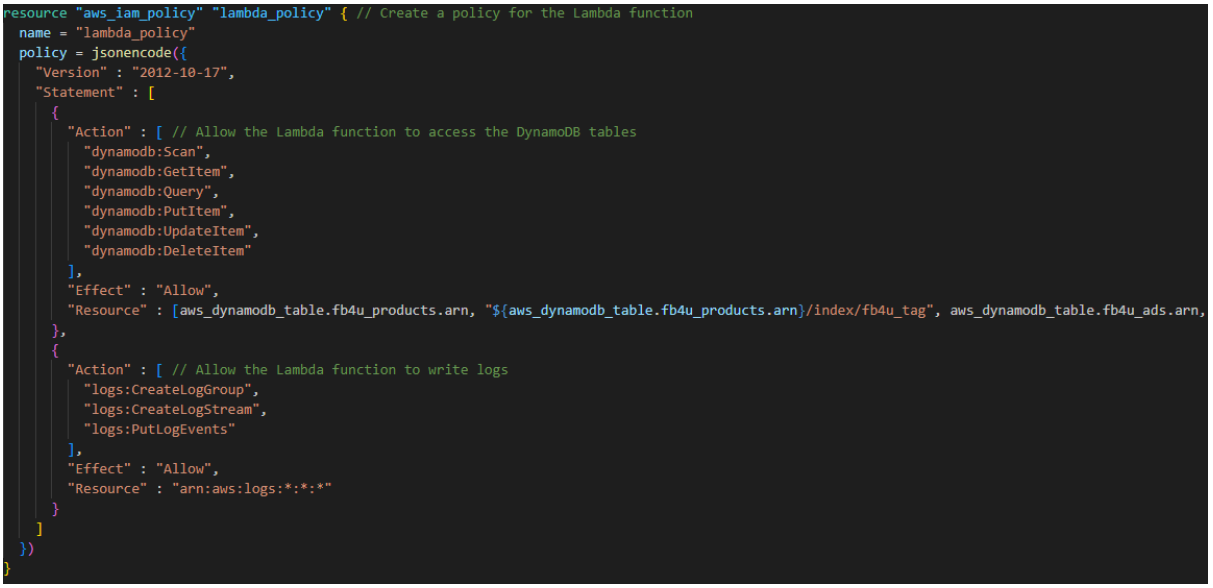
First is the creation of the s3 bucket. I have enabled force deletion to ease up cleaning the environment. Second is CORS settings to allow only GET requests from my domain. Third is enabling public access (as it is disabled by default) to enable my website to be accessible from the Internet.

```
resource "aws_s3_bucket_website_configuration" "website_s3b" { // Defining the website settings for S3 bucket
  bucket = aws_s3_bucket.s3b.id

  You, 3 months ago | 1 author (You)
  index_document {
    suffix = "index.html"
  }

  You, 3 months ago | 1 author (You)
  error_document {
    key = "error.html"
  }
}
You, 5 days ago | 2 authors (You and one other)
resource "aws_s3_bucket_policy" "bucket_policy_for_website" { // Setting the bucket policy
  bucket = aws_s3_bucket.s3b.id

  policy = jsonencode({
    Version = "2012-10-17",
    Statement = [
      {
        Effect    = "Allow",
        Principal = "*",
        Action    = "s3:GetObject",
        Resource  = "${aws_s3_bucket.s3b.arn}/*"
      }
    ]
  })

  depends_on = [aws_s3_bucket_public_access_block.s3b_enable_public_access]
}
```

The first is to enable static website hosting on the s3 bucket. The second is allowing to GET objects from the s3 bucket.

- **lambdas.tf** - to create lambdas function for the serverless backend functionality and configure the service to work properly.
    - **lambda_assume_role** - simple permission to allow lambda to assume a role to get necessary permissions.
    - **lambda_policy** - custom iam policy that allows lambda function access DynamoDB and do specific actions on it.
    - **lambda_policy_attachment** - small resource to connect policy with the role.
    - **list_products, list_home_page, store_user_data, get_user_data, add_to_cart, load_cart, clear_cart -** lambda functions with specific settings like python runtime, specific zip file with the python code, IAM role, name of the function inside the python and timeout of the lambda.

Permission settings to allow Lambda to assume a role.



Permission for the Lambda to do various actions on the DynamoDB. Necessary for the proper backend functionality.

```
Norbert Knez, 2 months ago | 2 authors (You and one other)
resource "aws_iam_role_policy_attachment" "lambda_policy_attachment" { // Attach the policy to the role
  role       = aws_iam_role.iam_for_lambda.name
  policy_arn = aws_iam_policy.lambda_policy.arn
}

Norbert Knez, 2 months ago | 2 authors (You and one other)
resource "aws_iam_role" "iam_for_lambda" { // Create a role for the Lambda function
  name               = "role_for_lambda"
  assume_role_policy = data.aws_iam_policy_document.lambda_assume_role.json
}
```

Attaching policy to the IAM role and IAM role to the assumption policy for lambda.

```
resource "aws_lambda_function" "list_products" { // Create a Lambda function for the kits listing
  function_name = "list_products"
  filename      = "../lambdas/lambda_load_products.zip" // Set the filename to the Lambda function zip file
  role          = aws_iam_role.iam_for_lambda.arn
  runtime       = local.lambda_runtime                 // Set the runtime to Python 3.12
  handler       = "lambda_load_products.lambda_handler" // Set the handler to lambda_handler
  timeout       = local.lambda_timeout                 // Set the timeout to 30 seconds
  memory_size   = local.lambda_memory_size
}

You, 5 days ago | 2 authors (Norbert Knez and one other)
resource "aws_lambda_function" "list_home_page" { // Create a Lambda function for the home page
  function_name = "list_home_page"
  filename      = "../lambdas/lambda_home_page.zip" // Set the filename to the Lambda function zip file
  role          = aws_iam_role.iam_for_lambda.arn
  runtime       = local.lambda_runtime              // Set the runtime to Python 3.12
  handler       = "lambda_home_page.lambda_handler" // Set the handler to lambda_handler
  timeout       = local.lambda_timeout              // Set the timeout to 30 seconds
  memory_size   = local.lambda_memory_size
}

You, 5 days ago | 2 authors (You and one other)
resource "aws_lambda_function" "store_user_data" { // Create a Lambda function for the home page
  function_name = "store_user_data"
  filename      = "../lambdas/lambda_store_user_data.zip" // Set the filename to the Lambda function zip file
  role          = aws_iam_role.iam_for_lambda.arn
  runtime       = local.lambda_runtime                    // Set the runtime to Python 3.12
  handler       = "lambda_store_user_data.lambda_handler" // Set the handler to lambda_handler
  timeout       = local.lambda_timeout                    // Set the timeout to 30 seconds
  memory_size   = local.lambda_memory_size
}
```

```
resource "aws_lambda_function" "get_user_data" { // Create a Lambda function for the home page
  function_name = "get_user_data"
  filename      = "../lambdas/lambda_get_user_data.zip" // Set the filename to the Lambda function zip file
  role          = aws_iam_role.iam_for_lambda.arn
  runtime       = local.lambda_runtime                 // Set the runtime to Python 3.12
  handler       = "lambda_get_user_data.lambda_handler" // Set the handler to lambda_handler
  timeout       = local.lambda_timeout                 // Set the timeout to 30 seconds
  memory_size   = local.lambda_memory_size
}

You, 5 days ago | 2 authors (You and one other)
resource "aws_lambda_function" "add_to_cart" { // Create a Lambda function for the home page
  function_name = "add_to_cart"
  filename      = "../lambdas/lambda_add_to_cart.zip" // Set the filename to the Lambda function zip file
  role          = aws_iam_role.iam_for_lambda.arn
  runtime       = local.lambda_runtime                // Set the runtime to Python 3.12
  handler       = "lambda_add_to_cart.lambda_handler" // Set the handler to lambda_handler
  timeout       = local.lambda_timeout                // Set the timeout to 30 seconds
  memory_size   = local.lambda_memory_size
}
You, 5 days ago | 2 authors (You and one other)
resource "aws_lambda_function" "load_cart" { // Create a Lambda function for the home page
  function_name = "load_cart"
  filename      = "../lambdas/lambda_load_cart.zip" // Set the filename to the Lambda function zip file
  role          = aws_iam_role.iam_for_lambda.arn
  runtime       = local.lambda_runtime             // Set the runtime to Python 3.12
  handler       = "lambda_load_cart.lambda_handler" // Set the handler to lambda_handler
  timeout       = local.lambda_timeout
  memory_size   = local.lambda_memory_size
}
```

```
resource "aws_lambda_function" "clear_cart" { // Create a Lambda function for the home page
  function_name = "clear_cart"
  filename      = "../lambdas/lambda_clear_cart.zip" // Set the filename to the Lambda function zip file
  role          = aws_iam_role.iam_for_lambda.arn
  runtime       = local.lambda_runtime               // Set the runtime to Python 3.12
  handler       = "lambda_clear_cart.lambda_handler" // Set the handler to lambda_handler
  timeout       = local.lambda_timeout               // Set the timeout to 30 seconds
  memory_size   = local.lambda_memory_size
}       You, 3 months ago • Adding testing lambda function, index.html and …
```

Lambda functions were created for the proper serverless backend functionality. Each one requires a runtime, a role with permissions and the code with proper handler set-up.

- **dynamodb.tf** - to create NOSQL tables in dynamodb service for data storage. This will provide high available serverless database solution for the project.
    - **fb4u_products, fb4u_ads, fb4u_users, fb4u_cart** - NOSQL dynamodb tables configured with their own primary key and in pay_per_request billing mode.
    - **Additionally, fb4u_products** has a **global secondary index** to give an additional way to query the table by a different key.

```
resource "aws_dynamodb_table" "fb4u_products" { // Create a DynamoDB table for the products
  name         = "fb4u_products"
  billing_mode = local.dynamodb_billing_mode // Set the billing mode to pay per request
  hash_key     = "product_id"              // Set the hash key (primary key) to product_id

  Norbert Knez, 2 months ago | 2 authors (You and one other)
  attribute {
    name = "product_id"
    type = "S"
  }

  Norbert Knez, 2 months ago | 2 authors (Norbert Knez and one other)
  attribute { // Additional tag attribute for the global secondary index
    name = "tag"
    type = "S"
  }

  Norbert Knez, 2 months ago | 2 authors (You and one other)
  global_secondary_index { // Create a global secondary index for the tag attribute to allow querying by tag
    name            = "fb4u_tag"
    hash_key        = "tag"
    projection_type = "ALL"
  }
}
```

Creation of fb4u_products. The resource contains the product_id primary key, the billing mode to pay by request, the second attribute to enable GSI creation and the Global Secondary Index itself.

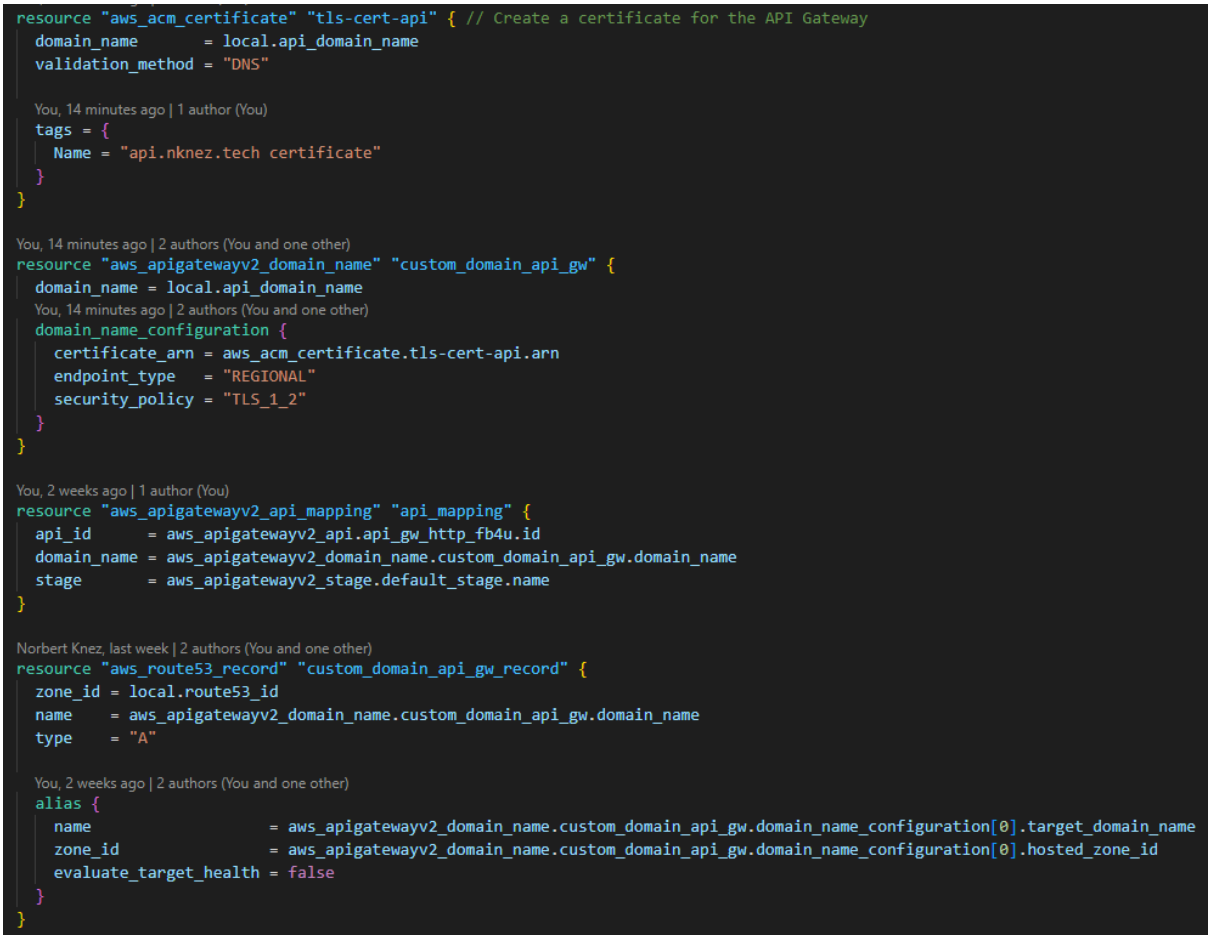
```
resource "aws_dynamodb_table" "fb4u_ads" { // Create a DynamoDB table for the ads
  name         = "fb4u_ads"
  billing_mode = local.dynamodb_billing_mode // Set the billing mode to pay per request
  hash_key     = "ad_id"                   // Set the hash key (primary key) to ad_id

  Norbert Knez, 2 months ago | 1 author (Norbert Knez)
  attribute {
    name = "ad_id"
    type = "S"
  }
}

You, 2 minutes ago | 1 author (You)
resource "aws_dynamodb_table" "fb4u_users" { // Create a DynamoDB table for the users
  name         = "fb4u_users"
  billing_mode = local.dynamodb_billing_mode // Set the billing mode to pay per request
  hash_key     = "user_id"                 // Set the hash key (primary key) to user_id

  You, 3 weeks ago | 1 author (You)
  attribute {
    name = "user_id"
    type = "S"
  }
      You, last week • terraform fmt
}

You, 2 minutes ago | 1 author (You)
resource "aws_dynamodb_table" "fb4u_cart" { // Create a DynamoDB table for the cart
  name         = "fb4u_cart"
  billing_mode = local.dynamodb_billing_mode // Set the billing mode to pay per request
  hash_key     = "user_id"                 // Set the hash key (primary key) to user_id so that each user has a unique cart
  You, 3 weeks ago | 1 author (You)
  attribute {
    name = "user_id"
    type = "S"
  }

}
```

Creation of the rest of the tables. Steps identical to the fb4u_products table but without the GSI and with different primary key.

- **api_gateway.tf** - to create HTTP API with a custom domain and lambda integrations and permissions with the specific routes.
    - **api_gw_http_fb4u** - the creation of API gateway in HTTP mode. It has an additional CORS configuration to allow only specific methods and headers.
    - **tls-cert-api** - TLS certificate creation for the custom API domain.
    - **custom_domain_api_gw** - settings custom domain name for the API.
    - **api_mapping** - connecting api gateway with the custom domain name.
    - **custom_domain_api_gw_record** - creating a custom A record in the Route 53 zone for the proper DNS functionality.
    - **load_products_integration, home_ads_integration, store_user_data_integration, get_user_data_integration, add_to_cart_integration, load_cart_integration, clear_cart_integration** - integrating specific lambda function with the api gateway.
    - **route_products, route_home, storeUserData_route, getUserData_route, addToCart_route, loadCart_route, clearCart_route** - exposing specific routes for GET or POST requests and integrating them with lambda functions.
    - **products_api_gateway_permission, home_api_gateway_permission, store_user_data_permission, get_user_data_permission, add_to_cart_permission, load_cart_permission, clear_cart_permission** - allowing API gateway to execute specific lambda functions.
    - **default_stage** - setting up a production stage for API gateway.

```
resource "aws_apigatewayv2_api" "api_gw_http_fb4u" { // Create an API Gateway
  name          = "api-gateway-http-fb4u"
  protocol_type = "HTTP" // The protocol used by the API Gateway

  You, 10 seconds ago | 1 author (You)
  cors_configuration {
    allow_origins = ["*"] // Allow the origin of the request
    allow_methods = ["GET", "OPTIONS", "POST"]
    allow_headers = ["*"]
  }

  depends_on = [aws_s3_bucket_website_configuration.website_s3b]
}
```

Creating API Gateway resource. HTTP protocol is used with custom cors configuration. It allows only GET, OPTIONS and POST methods.



```
resource "aws_acm_certificate" "tls-cert-api" { // Create a certificate for the API Gateway
  domain_name       = local.api_domain_name
  validation_method = "DNS"

  You, 14 minutes ago | 1 author (You)
  tags = {
    Name = "api.nknez.tech certificate"
  }
}

You, 14 minutes ago | 2 authors (You and one other)
resource "aws_apigatewayv2_domain_name" "custom_domain_api_gw" {
  domain_name = local.api_domain_name
  You, 14 minutes ago | 2 authors (You and one other)
  domain_name_configuration {
    certificate_arn = aws_acm_certificate.tls-cert-api.arn
    endpoint_type   = "REGIONAL"
    security_policy = "TLS_1_2"
  }
}

You, 2 weeks ago | 1 author (You)
resource "aws_apigatewayv2_api_mapping" "api_mapping" {
  api_id      = aws_apigatewayv2_api.api_gw_http_fb4u.id
  domain_name = aws_apigatewayv2_domain_name.custom_domain_api_gw.domain_name
  stage       = aws_apigatewayv2_stage.default_stage.name
}

Norbert Knez, last week | 2 authors (You and one other)
resource "aws_route53_record" "custom_domain_api_gw_record" {
  zone_id = local.route53_id
  name    = aws_apigatewayv2_domain_name.custom_domain_api_gw.domain_name
  type    = "A"

  You, 2 weeks ago | 2 authors (You and one other)
  alias {
    name                   = aws_apigatewayv2_domain_name.custom_domain_api_gw.domain_name_configuration[0].target_domain_name
    zone_id                = aws_apigatewayv2_domain_name.custom_domain_api_gw.domain_name_configuration[0].hosted_zone_id
    evaluate_target_health = false
  }
}
```

First one is creation of TLS certificate for the custom domain for the API. Second is creation of custom domain for the API with TLS certificated attached. Third is mapping necessary to bind domain name with the API. Fourth is custom A record in the route 53 for the custom domain functionality.

```
resource "aws_apigatewayv2_integration" "load_products_integration" { // Create an integration for the kits listing
  api_id           = aws_apigatewayv2_api.api_gw_http_fb4u.id
  integration_type = "AWS_PROXY"
  integration_uri  = aws_lambda_function.list_products.invoke_arn
}

You, 3 weeks ago | 1 author (You)
resource "aws_apigatewayv2_integration" "home_ads_integration" { // Create an integration for the home page
  api_id           = aws_apigatewayv2_api.api_gw_http_fb4u.id
  integration_type = "AWS_PROXY"
  integration_uri  = aws_lambda_function.list_home_page.invoke_arn
}

You, 3 weeks ago | 1 author (You)
resource "aws_apigatewayv2_integration" "store_user_data_integration" { // Create an integration for the home page
  api_id           = aws_apigatewayv2_api.api_gw_http_fb4u.id
  integration_type = "AWS_PROXY"
  integration_uri  = aws_lambda_function.store_user_data.invoke_arn
}

You, 3 weeks ago | 2 authors (Norbert Knez and one other)
resource "aws_apigatewayv2_integration" "get_user_data_integration" { // Create an integration for the home page
  api_id           = aws_apigatewayv2_api.api_gw_http_fb4u.id
  integration_type = "AWS_PROXY"
  integration_uri  = aws_lambda_function.get_user_data.invoke_arn
}

You, 3 weeks ago | 1 author (You)
resource "aws_apigatewayv2_integration" "add_to_cart_integration" { // Create an integration for the home page
  api_id           = aws_apigatewayv2_api.api_gw_http_fb4u.id
  integration_type = "AWS_PROXY"
  integration_uri  = aws_lambda_function.add_to_cart.invoke_arn
}

You, 3 weeks ago | 1 author (You)
resource "aws_apigatewayv2_integration" "load_cart_integration" { // Create an integration for the home page
  api_id           = aws_apigatewayv2_api.api_gw_http_fb4u.id
  integration_type = "AWS_PROXY"
  integration_uri  = aws_lambda_function.load_cart.invoke_arn
}

You, 3 weeks ago | 1 author (You)
resource "aws_apigatewayv2_integration" "clear_cart_integration" { // Create an integration for the home page
  api_id           = aws_apigatewayv2_api.api_gw_http_fb4u.id
  integration_type = "AWS_PROXY"
  integration_uri  = aws_lambda_function.clear_cart.invoke_arn
}
```

Integrations between Lambdas and API Gateway. Necessary for proper backend functionality.

```
resource "aws_apigatewayv2_route" "route_products" { // Create a route for the product l
  api_id    = aws_apigatewayv2_api.api_gw_http_fb4u.id
  route_key = "GET /loadProducts"
  target    = "integrations/${aws_apigatewayv2_integration.load_products_integration.id}"
}

Norbert Knez, 2 months ago | 1 author (Norbert Knez)
resource "aws_apigatewayv2_route" "route_home" { // Create a route for the home page
  api_id    = aws_apigatewayv2_api.api_gw_http_fb4u.id
  route_key = "GET /"
  target    = "integrations/${aws_apigatewayv2_integration.home_ads_integration.id}"
}

You, 3 weeks ago | 1 author (You)
resource "aws_apigatewayv2_route" "storeUserData_route" {
  api_id    = aws_apigatewayv2_api.api_gw_http_fb4u.id
  route_key = "POST /storeUserData"
  target    = "integrations/${aws_apigatewayv2_integration.store_user_data_integration.i
}
You, 3 weeks ago | 1 author (You)
resource "aws_apigatewayv2_route" "getUserData_route" {
  api_id    = aws_apigatewayv2_api.api_gw_http_fb4u.id
  route_key = "GET /getUserData"
  target    = "integrations/${aws_apigatewayv2_integration.get_user_data_integration.id}"
}

You, 3 weeks ago | 1 author (You)
resource "aws_apigatewayv2_route" "addToCart_route" {
  api_id    = aws_apigatewayv2_api.api_gw_http_fb4u.id
  route_key = "POST /addToCart"
  target    = "integrations/${aws_apigatewayv2_integration.add_to_cart_integration.id}"
}

You, 3 weeks ago | 1 author (You)
resource "aws_apigatewayv2_route" "loadCart_route" {
  api_id    = aws_apigatewayv2_api.api_gw_http_fb4u.id
  route_key = "GET /loadCart"
  target    = "integrations/${aws_apigatewayv2_integration.load_cart_integration.id}"
}

You, 3 weeks ago | 1 author (You)
resource "aws_apigatewayv2_route" "clearCart_route" {
  api_id    = aws_apigatewayv2_api.api_gw_http_fb4u.id
  route_key = "POST /clearCart"
  target    = "integrations/${aws_apigatewayv2_integration.clear_cart_integration.id}"
}
```

Exposing routes for each lambda function.

```
resource "aws_lambda_permission" "products_api_gateway_permission" { // Create a permission for the kits listing
  statement_id  = "AllowExecutionFromAPIGateway"
  action        = "lambda:InvokeFunction"
  function_name = aws_lambda_function.list_products.function_name
  principal     = "apigateway.amazonaws.com"
  source_arn    = "${aws_apigatewayv2_api.api_gw_http_fb4u.execution_arn}/*"
}

You, 3 weeks ago | 1 author (You)
resource "aws_lambda_permission" "home_api_gateway_permission" { // Create a permission for the home page
  statement_id  = "AllowExecutionFromAPIGateway"
  action        = "lambda:InvokeFunction"
  function_name = aws_lambda_function.list_home_page.function_name
  principal     = "apigateway.amazonaws.com"
  source_arn    = "${aws_apigatewayv2_api.api_gw_http_fb4u.execution_arn}/*"
}

You, 3 weeks ago | 1 author (You)
resource "aws_lambda_permission" "store_user_data_permission" { // Create a permission for the storing user data
  statement_id  = "AllowExecutionFromAPIGateway"
  action        = "lambda:InvokeFunction"
  function_name = aws_lambda_function.store_user_data.function_name
  principal     = "apigateway.amazonaws.com"
  source_arn    = "${aws_apigatewayv2_api.api_gw_http_fb4u.execution_arn}/*"
}
You, 3 weeks ago | 1 author (You)
resource "aws_lambda_permission" "get_user_data_permission" { // Create a permission for the storing user data
  statement_id  = "AllowExecutionFromAPIGateway"
  action        = "lambda:InvokeFunction"
  function_name = aws_lambda_function.get_user_data.function_name
  principal     = "apigateway.amazonaws.com"
  source_arn    = "${aws_apigatewayv2_api.api_gw_http_fb4u.execution_arn}/*"
}

You, 3 weeks ago | 2 authors (Norbert Knez and one other)
resource "aws_lambda_permission" "add_to_cart_permission" { // Create a permission for the storing user data
  statement_id  = "AllowExecutionFromAPIGateway"
  action        = "lambda:InvokeFunction"
  function_name = aws_lambda_function.add_to_cart.function_name
  principal     = "apigateway.amazonaws.com"
  source_arn    = "${aws_apigatewayv2_api.api_gw_http_fb4u.execution_arn}/*"
}
```

```
resource "aws_lambda_permission" "load_cart_permission" { // Create a permission for the storing user data
  statement_id  = "AllowExecutionFromAPIGateway"
  action        = "lambda:InvokeFunction"
  function_name = aws_lambda_function.load_cart.function_name
  principal     = "apigateway.amazonaws.com"
  source_arn    = "${aws_apigatewayv2_api.api_gw_http_fb4u.execution_arn}/*"
}

You, 3 weeks ago | 1 author (You)
resource "aws_lambda_permission" "clear_cart_permission" { // Create a permission for the storing user data
  statement_id  = "AllowExecutionFromAPIGateway"
  action        = "lambda:InvokeFunction"
  function_name = aws_lambda_function.clear_cart.function_name
  principal     = "apigateway.amazonaws.com"
  source_arn    = "${aws_apigatewayv2_api.api_gw_http_fb4u.execution_arn}/*"
}
```

Permissions necessary to allow API Gateway to execute lambda functions.

```
resource "aws_apigatewayv2_stage" "default_stage" { // Create a stage for the API Gateway
  api_id      = aws_apigatewayv2_api.api_gw_http_fb4u.id
  name        = "$default"
  auto_deploy = true
}
```

Creation of default stage where API Gateway will be primarily deployed.

- **cloudfront.tf** - to create a CDN to cache website assets.
    - **s3_distribution** - CloudFront distribution configured with specific settings like:
        - The target deployment region is us-east-1 (as this is the default for the global services within AWS)
        - Uses my custom domain name as an alias.
        - Setting up the origin to the s3 bucket website URL
        - Enable only HTTP connection to the CloudFront (required for s3 website origins)
        - Setup cache behaviour to redirect HTTP connections to HTTPS, allow only GET and HEAD methods and use a custom cache policy.
        - Disables any geographical restrictions.
        - Links with the TLS certificate for the proper HTTPS functionality.
    - **tls-cert** - custom certificate for the my custom domain name. Used by CloudFront.
    - **record_for_cloudfront** - creation of A record in my Route53 hosted zone to link my domain to CloudFront URL.

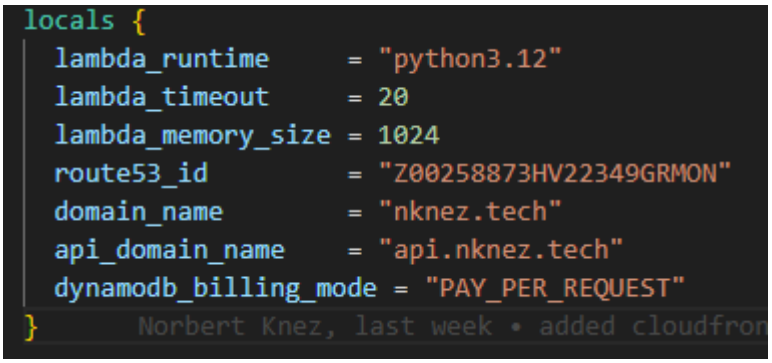
```
resource "aws_cloudfront_distribution" "s3_distribution" { // Define the CloudFront distribution for the S3 bucket
  provider = aws.us-east-1 // Use the alias AWS provider for the us-east-1 region

  aliases = [local.domain_name] // Define the domain name for the CloudFront distribution

  ...
  origin { // Define the origin for the CloudFront distribution
    domain_name = "nknez.tech.s3-website.eu-central-1.amazonaws.com"
    origin_id   = "S3-Website-nknez.tech"

    ...
    custom_origin_config {
      http_port              = 80
      https_port             = 443
      origin_protocol_policy = "http-only"
      origin_ssl_protocols   = ["TLSv1.2"]
    }
  }

  enabled             = true
  is_ipv6_enabled     = false
  comment             = "S3 bucket website distribution"
  default_root_object = "index.html"

  ...
  default_cache_behavior { // Define the default cache behavior for the CloudFront distribution
    target_origin_id       = "S3-Website-nknez.tech"
    viewer_protocol_policy = "redirect-to-https"
    allowed_methods        = ["GET", "HEAD"]
    cached_methods         = ["GET", "HEAD"]
    cache_policy_id        = "658327ea-f89d-4fab-a63d-7e88639e58f6"
  }

  ...
  restrictions { // Define the restrictions for the CloudFront distribution
    ...
    geo_restriction {
      restriction_type = "none"
    }
  }

  ...
  viewer_certificate { // Define the viewer certificate for the CloudFront distribution
    acm_certificate_arn      = aws_acm_certificate.tls-cert.arn
    ssl_support_method       = "sni-only"
    minimum_protocol_version = "TLSv1.2_2021"
  }
```

CloudFront distribution creation. You can see all the explained the settings.

```
resource "aws_acm_certificate" "tls-cert" { // Define the ACM certificate for the CloudFront distribution       You, 4
  provider = aws.us-east-1

  domain_name       = local.domain_name
  validation_method = "DNS"

  You, 2 weeks ago | 1 author (You)
  tags = {
    Name = "nknez.tech certificate"
  }
}

You, 2 minutes ago | 2 authors (Norbert Knez and one other)
resource "aws_route53_record" "record_for_cloudfront" { // Define the Route 53 record for the CloudFront distribution
  zone_id = local.route53_id
  name    = local.domain_name
  type    = "A"

  Norbert Knez, last week | 1 author (Norbert Knez)
  alias {
    name                   = aws_cloudfront_distribution.s3_distribution.domain_name
    zone_id                = aws_cloudfront_distribution.s3_distribution.hosted_zone_id
    evaluate_target_health = false
  }
}
```

Creation of the TLS certificate for the CloudFront and creating an A record in the Route 53 hosted zone.

- **locals.tf** - file created to hold local variables to improve code manageability.
    - **lambda_runtime** - setting up programming language for the lambdas.
    - **lambda_timeout** - time after the lambda function will stop functioning if the operation is long.
    - **lambda_memory_size** - size in MB of the memory for the Lambda. It automatically increases the amount of vCPU according to the memory size.
    - **route53_id** - ID of my Route53 hosted zone (for now hosted zone is deployed manually, subject to change)
    - **domain_name** - my custom domain name for the website URL.
    - **api_domain_name** - my custom domain name for the API URL.
    - **dynamodb_billing_mode** - setting up billing mode of dynamodb to pay by request.

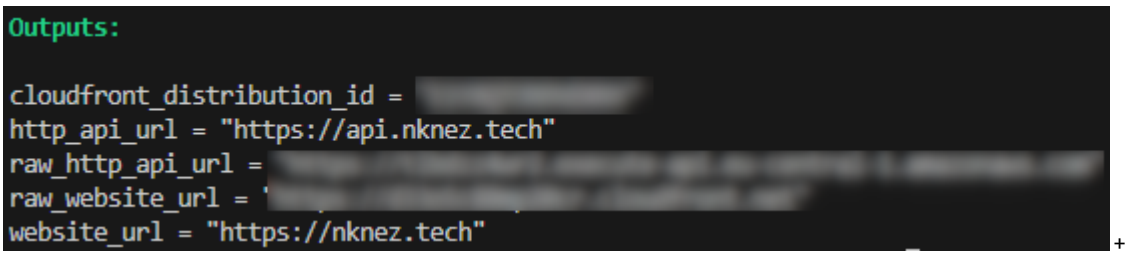


Very small files to enhance my manageability of the code.

- **outputs.tf** - file to store all important outputs that will be displayed after terraform deployment
    - **website_url** - URL of my website (nknez.tech).
    - **raw_website_url** - URL of the CloudFront distribution.
    - **http_api_url** - URL of my API (api.nknez.tech).
    - **raw_http_api_url** - raw URL of my API.
    - **cloudfront_distribution_id** - ID of my CloudFront distribution (necessary for CI/CD automation)

```
output "website_url" { // Output of the URL of the website
  value = "https://${local.domain_name}"
}

You, 6 days ago | 1 author (You)
output "raw_website_url" { // Output of the raw URL of the website
  value = "https://${aws_cloudfront_distribution.s3_distribution.domain_name}"
}

You, 6 days ago | 1 author (You)
output "http_api_url" { // Output the URL of the API Gateway
  value = "https://${local.api_domain_name}"
}

You, 6 days ago | 1 author (You)
output "raw_http_api_url" { // Output the raw URL of the API Gateway
  value = aws_apigatewayv2_api.api_gw_http_fb4u.api_endpoint
}

You, 6 days ago | 1 author (You)
output "cloudfront_distribution_id" { // Output the ID of the CloudFront distribution
  value = aws_cloudfront_distribution.s3_distribution.id
}
```

Small file with only described outputs defined.



This is how it looks like in the command line (blurred for security reasons).

4. Conclusion

This documentation has outlined the design, components, and automation steps used to create a serverless infrastructure for this project. By leveraging Terraform, I was able to streamline deployment, reduce manual effort, and optimize costs, especially by removing resources when not in use. This approach not only saves time but also minimizes the potential for human error, ensuring a more reliable and repeatable setup. Infrastructure as Code with Terraform provides flexibility and scalability, allowing rapid modifications as the project evolves, making it an essential tool for modern cloud infrastructure management.