Processing Webhooks with Terraform and AWS

Processing Webhooks with Terraform and AWS

Scott Blake

Client Platform Engineer

@MScottBlake

https://github.com/MScottBlake/Presentations

Agenda

- Webhooks!
- How do we harness this power?
- Terraform Basics
- AWS Services
- Putting it all together

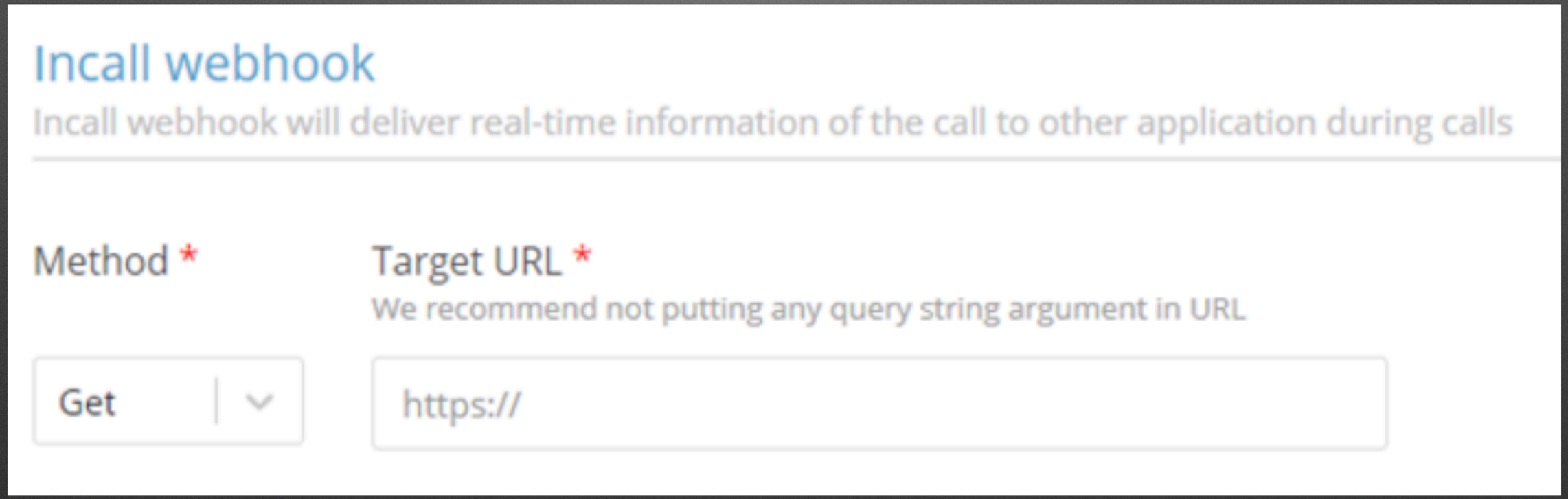
What is a Webhook?

What is a Webhook?

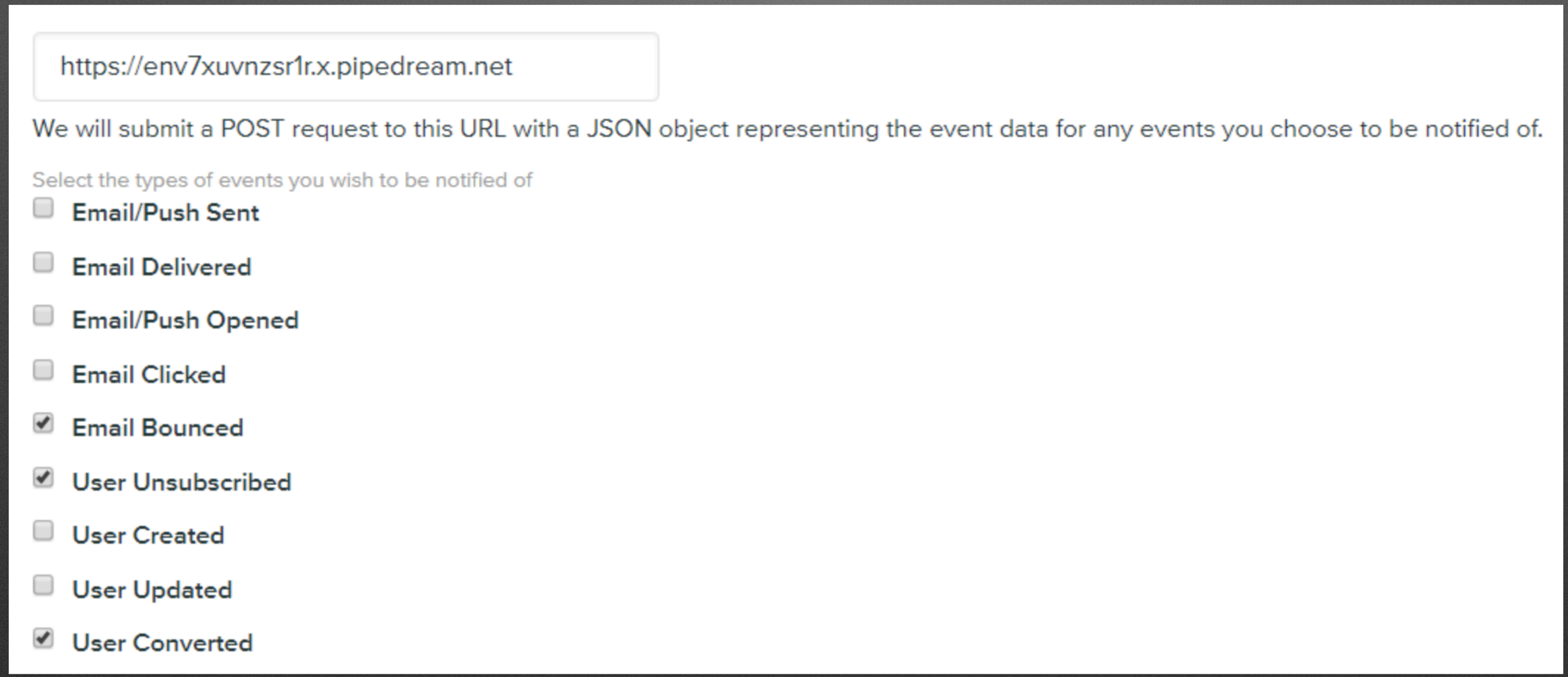
- A web-based notification whenever a specific event occurs

How are web hooks configured?

How are web hooks configured?



How are web hooks configured?



Webhook message contents

Webhook message contents

```
{
  "event": {
    "managementId": integer,
    "type": "string"
  },
  "webhook": {
    "eventTimestamp": epoch,
    "id": integer,
    "name": "string",
    "webhookEvent": "PushSent"
  }
}
```

Source: https://developer.jamf.com/developer-guide/docs/webhooks

Webhook message contents

https://webhook.site

https://github.com/webhooksite/webhook.site

What do I do with this info?

Terraform Basics

Terraform Basics

- Resources
- Data Sources
- Modules
- Input Variables

Resource

Resource

```
resource "aws_s3_bucket" "example" {
    bucket = "my-test-bucket"
}
```

Resource

aws_s3_bucket.example.id

Resource

aws_s3_bucket.example.id
aws_s3_bucket.example.arn
aws_s3_bucket.example.bucket_domain_name
aws_s3_bucket.example.bucket_regional_domain_name
aws_s3_bucket.example.hosted_zone_id
aws_s3_bucket.example.region
aws_s3_bucket.example.tags_all

Source: https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/s3_bucket#attributes-reference

Resource

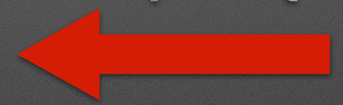
```
resource "aws_s3_bucket" "example" {
    bucket = "my-test-bucket"
}
```

Resource

```
resource "aws_s3_bucket" "example" {
    bucket = "my-test-bucket"
}
resource "aws_s3_bucket_versioning" "example" {
    bucket = aws_s3_bucket.example.id
    versioning_configuration {
        status = "Enabled"
    }
}
```

Resource

```
resource "aws_s3_bucket" "example" {
    bucket = "my-test-bucket"
}
resource "aws_s3_bucket_versioning" "example" {
    bucket = aws_s3_bucket.example.id
    versioning_configuration {
        status = "Enabled"
    }
}
```



Data Source

Data Source

```
data "aws_s3_bucket" "example" {
  bucket = "my-test-bucket"
}
```

Data Source

data.aws_s3_bucket.example.id
data.aws_s3_bucket.example.arn
data.aws_s3_bucket.example.bucket_domain_name
data.aws_s3_bucket.example.bucket_regional_domain_name
data.aws_s3_bucket.example.hosted_zone_id
data.aws_s3_bucket.example.region
data.aws_s3_bucket.example.website_endpoint
data.aws_s3_bucket.example.website_domain

Source: https://registry.terraform.io/providers/hashicorp/aws/latest/docs/data-sources/s3_bucket#attribute-reference

Data Source

```
data "aws_s3_bucket" "example" {
    bucket = "my-test-bucket"
}
resource "aws_s3_bucket_versioning" "example" {
    bucket = data.aws_s3_bucket.example.id
    versioning_configuration {
        status = "Enabled"
    }
}
```

Module

Module

```
module "example_bucket" {
    source = "terraform-aws-modules/s3-bucket/aws"
    version = "3.14.0"

    bucket = "my-test-bucket"
}
```

Module

module.example_bucket.s3_bucket_arn
module.example_bucket.s3_bucket_bucket_domain_name
module.example_bucket.s3_bucket_bucket_regional_domain_name
module.example_bucket.s3_bucket_hosted_zone_id
module.example_bucket.s3_bucket_id
module.example_bucket.s3_bucket_lifecycle_configuration_rules
module.example_bucket.s3_bucket_policy
module.example_bucket.s3_bucket_region
module.example_bucket.s3_bucket_website_domain
module.example_bucket.s3_bucket_website_endpoint

Source: https://registry.terraform.io/modules/terraform-aws-modules/s3-bucket/aws/latest?tab=outputs

Module

```
module "example_bucket" {
    source = "terraform-aws-modules/s3-bucket/aws"
    version = "3.14.0"

    bucket = "my-test-bucket"
}
```

Module

```
module "example_bucket" {
    source = "terraform-aws-modules/s3-bucket/aws"
    version = "3.14.0"

    bucket = "my-test-bucket"
    versioning = {
        enabled = true
    }
}
```

Module

```
module "example_bucket" {
  source = "terraform-aws-modules/s3-bucket/aws"
  version = "3.14.0"

  bucket = "my-test-bucket"
  versioning = {
    enabled = true
  }
  acl = "private"
  logging = {
    target_bucket = module.log_bucket.s3_bucket_id
    target_prefix = "log/"
  }
}
```

Variables

Variables

```
bucket_name = "my-test-bucket"
enable_versioning = true
```

Variables

```
module "example_bucket" {
    source = "terraform-aws-modules/s3-bucket/aws"
    version = "3.14.0"

    bucket = "my-test-bucket"
    versioning = {
        enabled = true
    }
}
```

Variables

```
module "example_bucket" {
    source = "terraform-aws-modules/s3-bucket/aws"
    version = "3.14.0"

    bucket = var.bucket_name
    versioning = {
        enabled = var.enable_versioning
    }
}
```

Variables

aws_s3_bucket.example.id

data.aws_s3_bucket.example.id

module.example_bucket.s3_bucket_id

var.bucketname

Variables

"prefix-${aws_s3_bucket.example.id}-postfix"

"prefix-${data.aws_s3_bucket.example.id}-postfix"

"prefix-${module.example_bucket.s3_bucket_id}-postfix"

"prefix-${var.bucketname}-postfix"

Variables

"${var.subdomain}.${var.domain}"

description = "Holds images for the ${var.subdomain}.${var.domain} website."

Variables

```
"${var.subdomain}.${var.domain}"

description = "Holds images for the ${var.subdomain}.${var.domain} website."

locals {
    full_domain = "${var.subdomain}.${var.domain}"
}

description = "Holds images for the ${local.full_domain} website."
```

Terraform Commands

- terraform init
- terraform validate
- terraform plan
- terraform apply
- terraform destroy

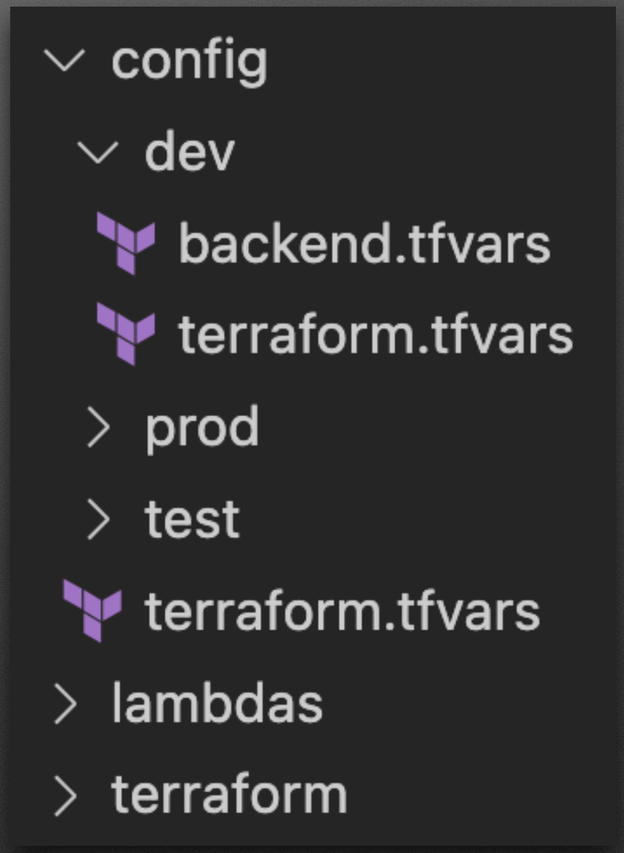
terraform init

terraform init

```
terraform init
  -backend-config="../config/dev/backend.tfvars"
```

terraform init

```
terraform init
  -backend-config="../config/dev/backend.tfvars"
```



terraform init

```
terraform init
  -backend-config="../config/dev/backend.tfvars"
```

config/dev/backend.tfvars:

```
bucket         = "dev-terraform-remote-state"
region         = "us-west-1"
key            = "env:/dev/sample-webhook-handler"
dynamodb_table = "dev-terraform-state-lock"
```

terraform validate

terraform plan

terraform plan

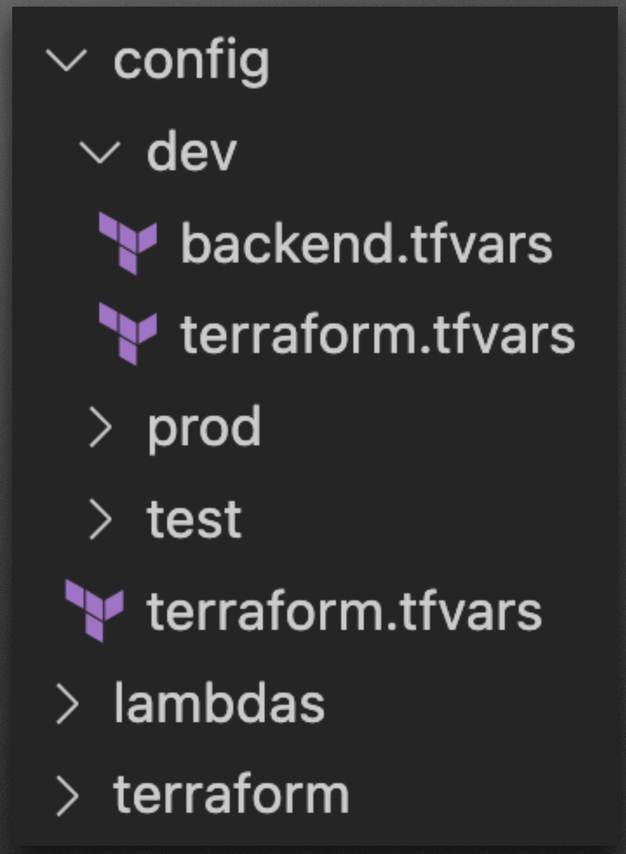
```
terraform plan
  -out plan.out
```

terraform plan

```
terraform plan
  -out plan.out
  -var-file="../config/dev/terraform.tfvars"
  -var-file="../config/terraform.tfvars"
```

terraform plan

```
terraform plan
  -out plan.out
  -var-file="../config/dev/terraform.tfvars"
  -var-file="../config/terraform.tfvars"
```



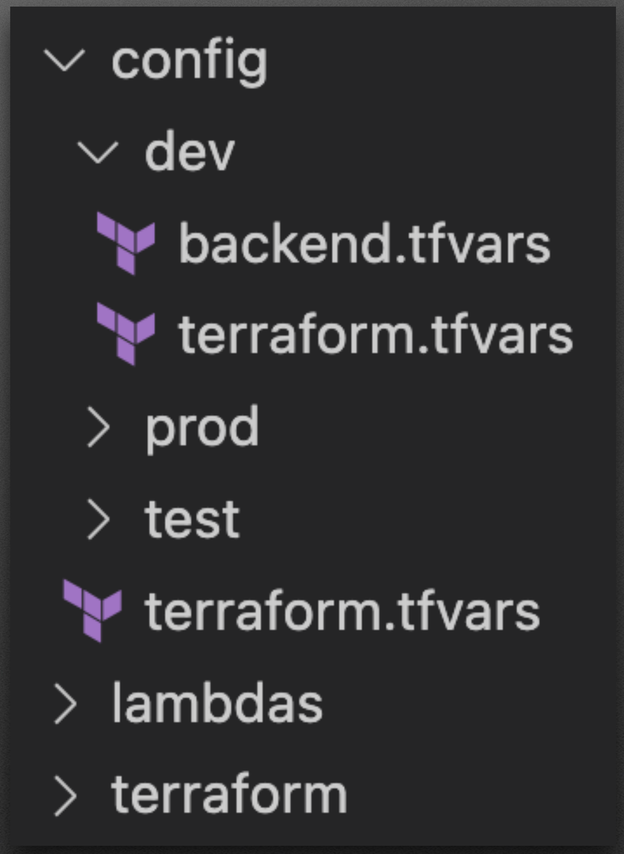
terraform plan

config/dev/terraform.tfvars:

```
aws_account_alias = "sample_account_alias"
domain            = "domain.io"
environment       = "dev"
region            = "us-west-1"
```

config/terraform.tfvars:

```
stack_name = "sample-webhook-handler"
subdomain  = "sample-webhook-handler"
```



terraform apply

terraform apply

```
terraform apply
  plan.out
```

terraform apply

```
terraform apply
  -input=false
  plan.out
```

terraform destroy

terraform destroy

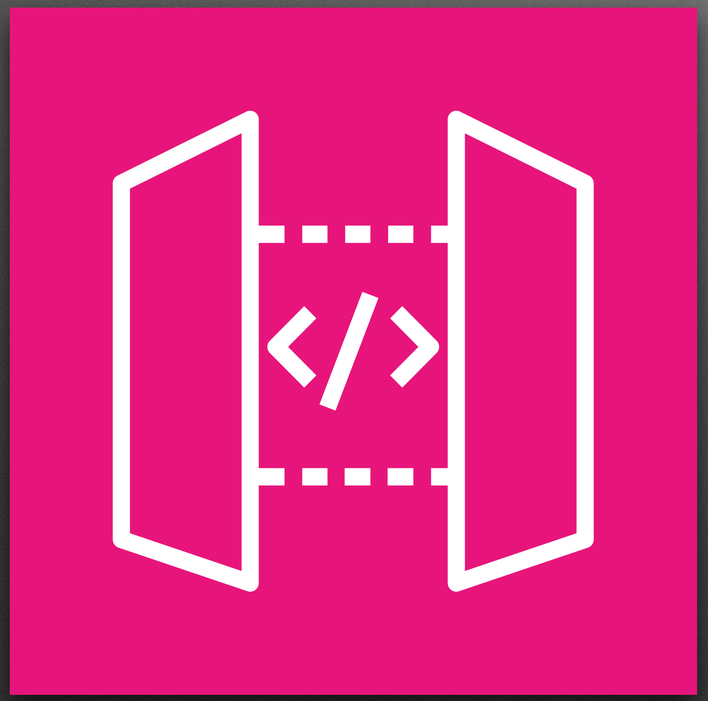
terraform destroy

terraform apply -destroy

terraform plan -destroy

Amazon Web Services (AWS)

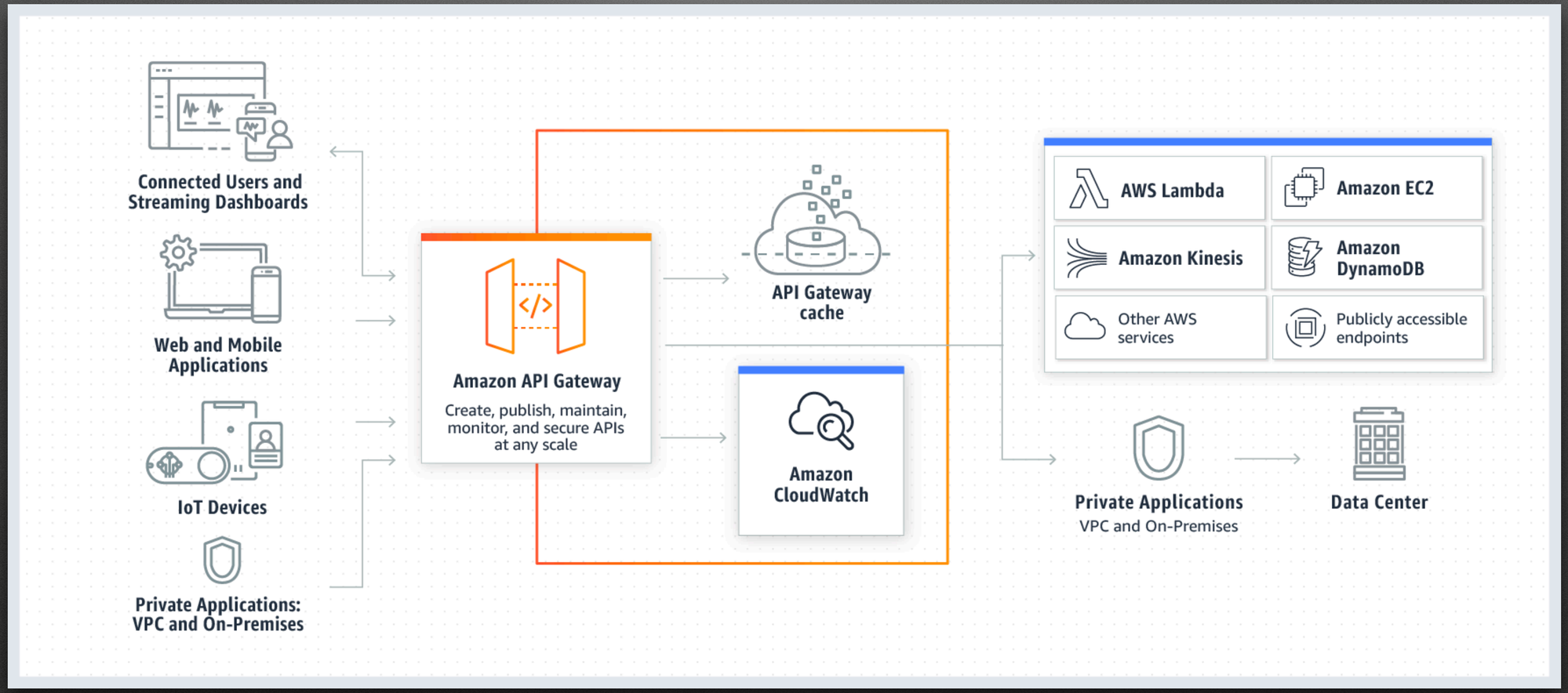
AWS



API Gateway

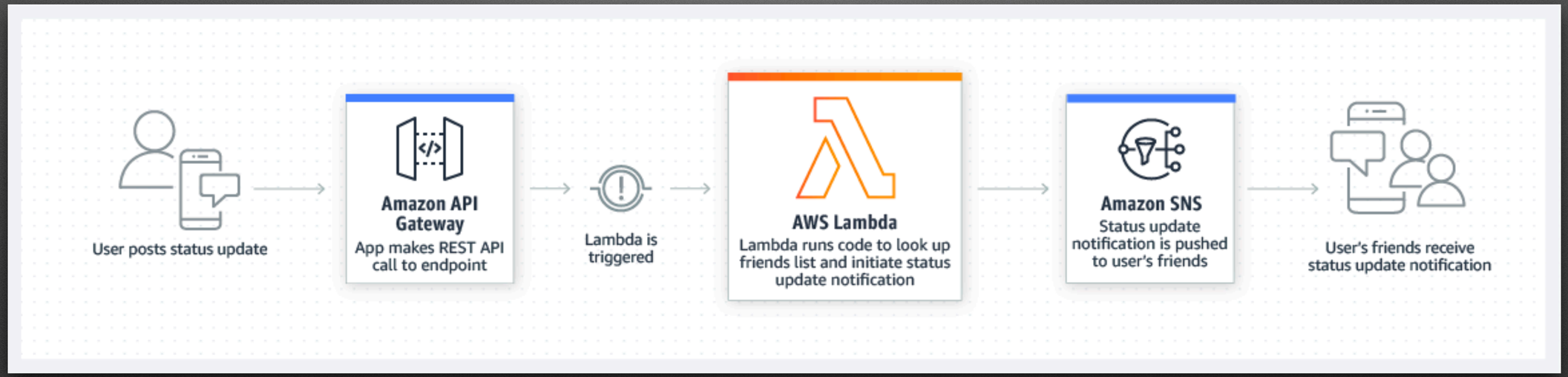
Lambda

API Gateway



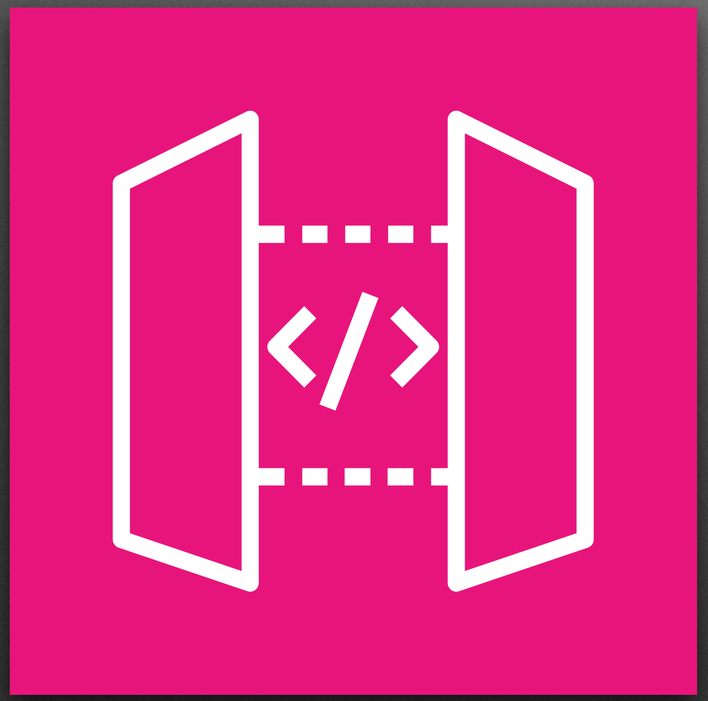
https://aws.amazon.com/api-gateway

Lambda



https://aws.amazon.com/lambda/

AWS



API Gateway

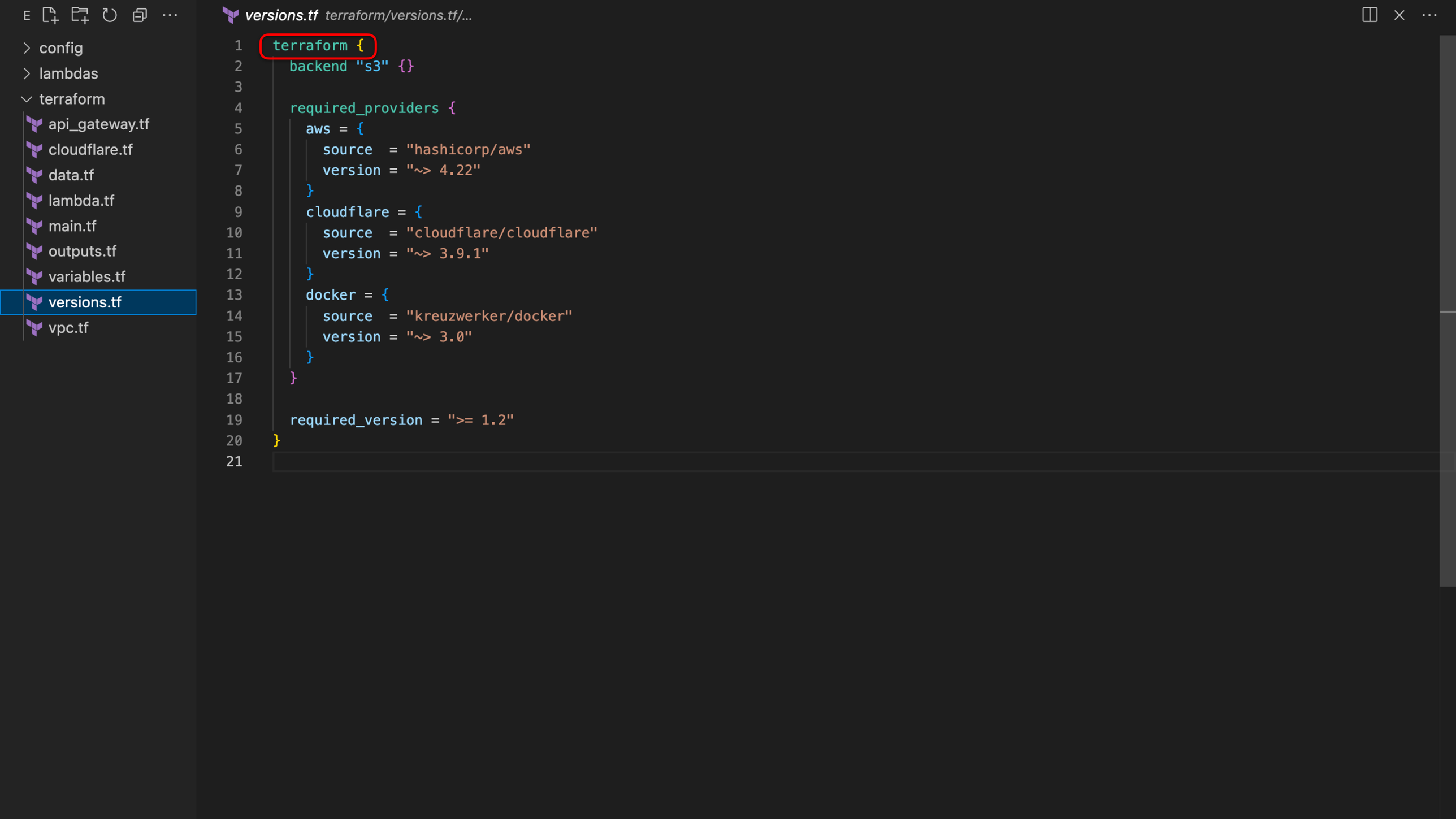
Lambda

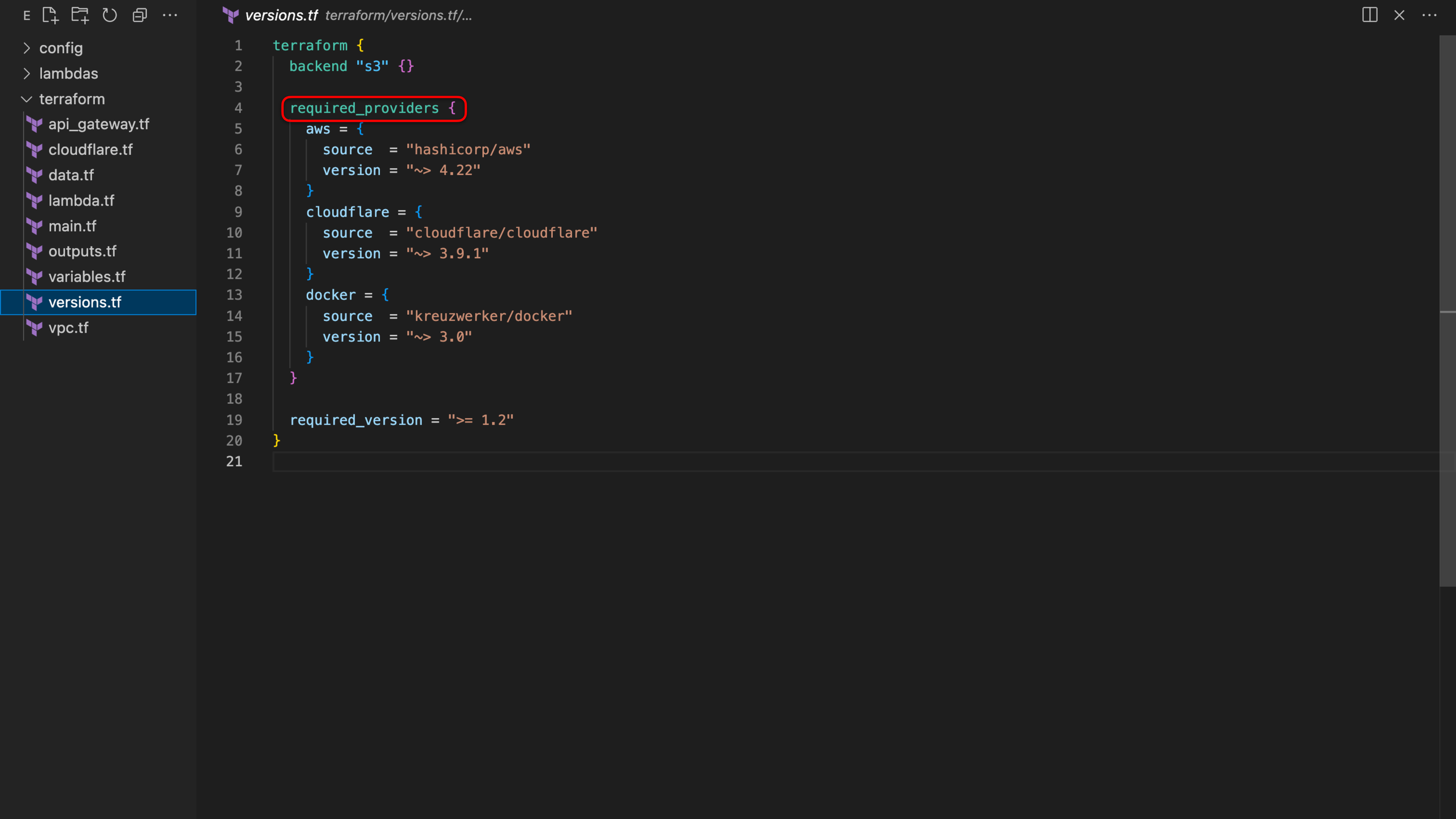
AWS

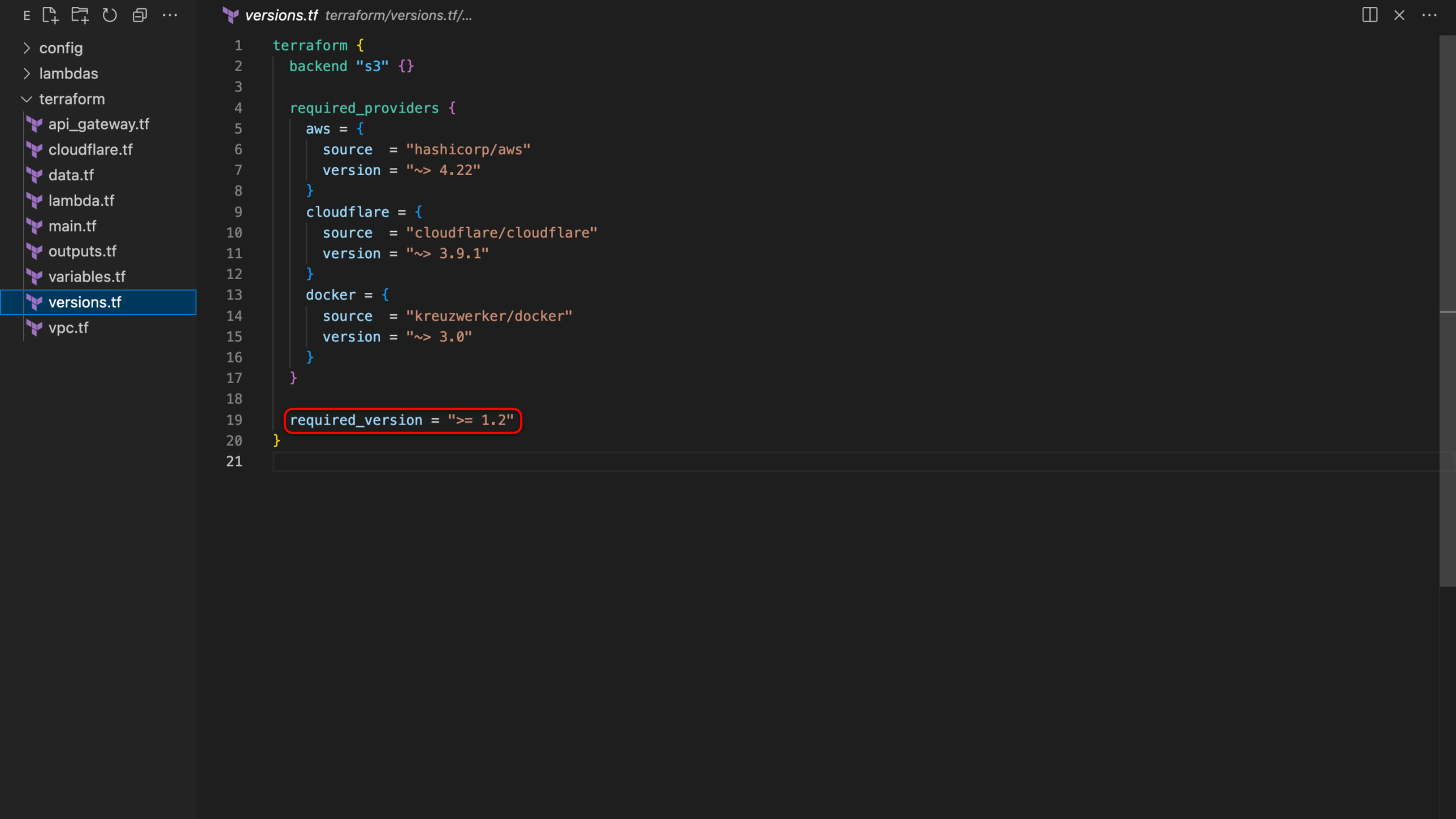
https://registry.terraform.io/modules/terraform-aws-modules/apigateway-v2/aws/latest

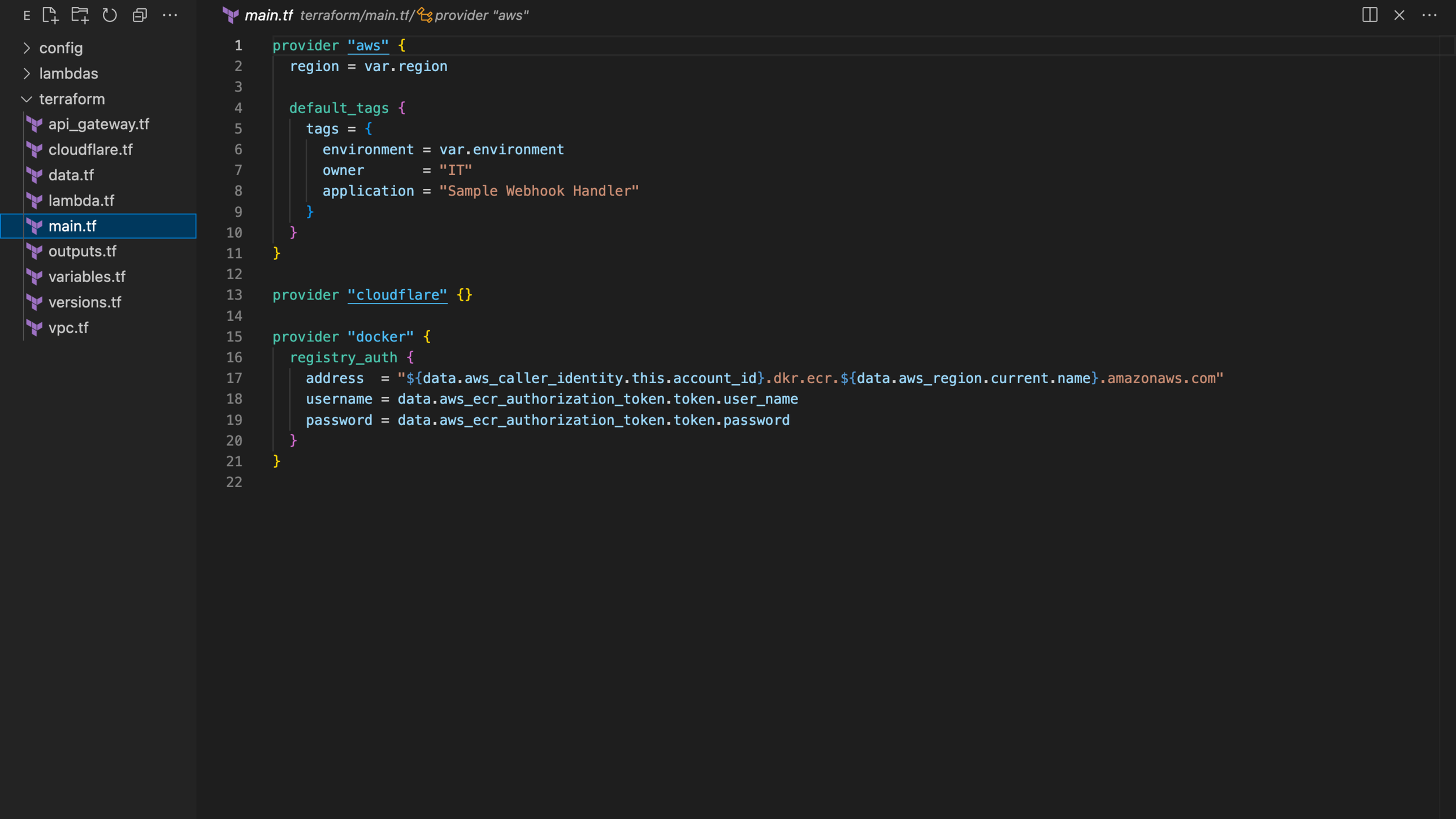
https://registry.terraform.io/modules/terraform-aws-modules/lambda/aws/latest

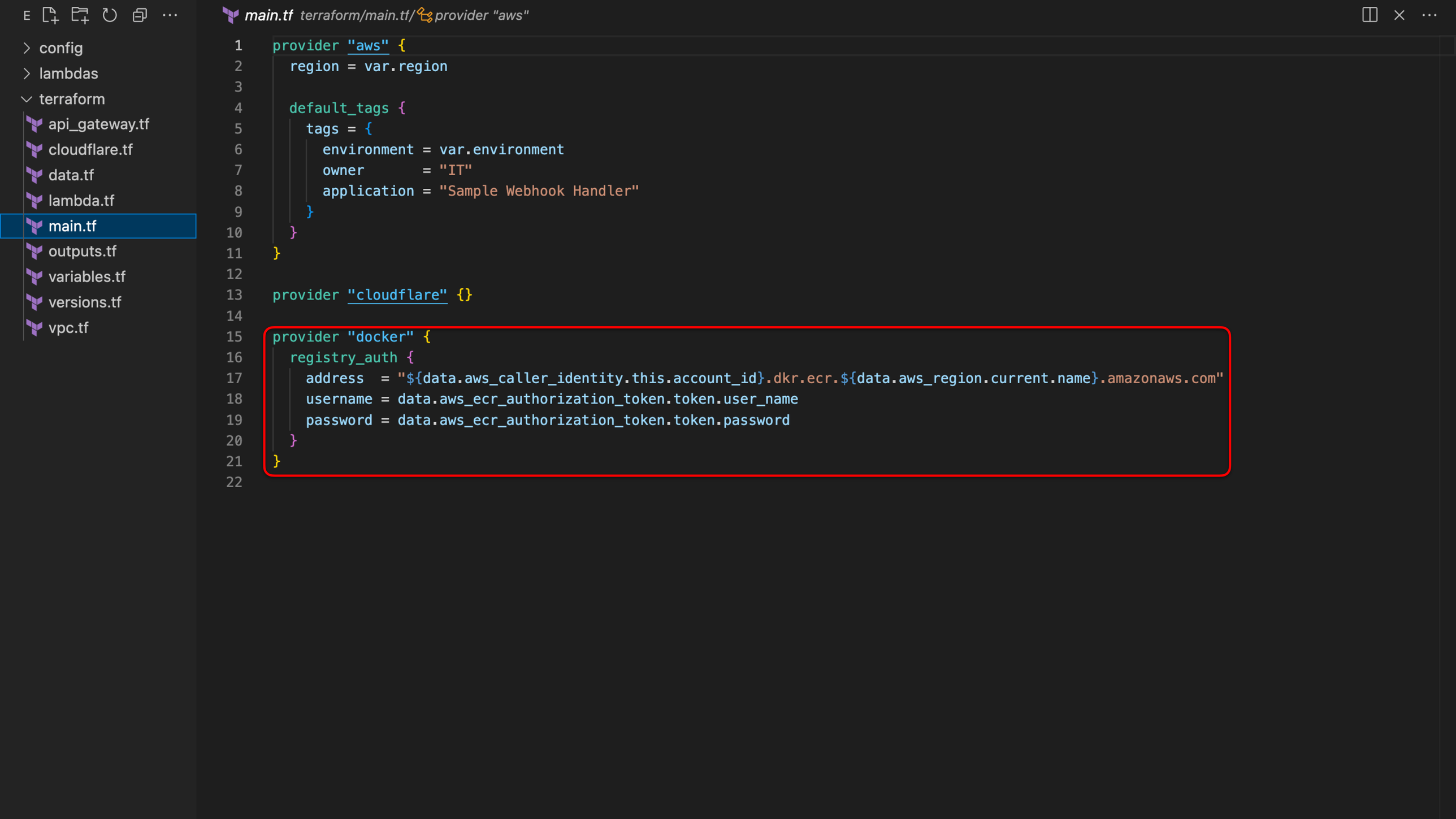
Putting it all Together

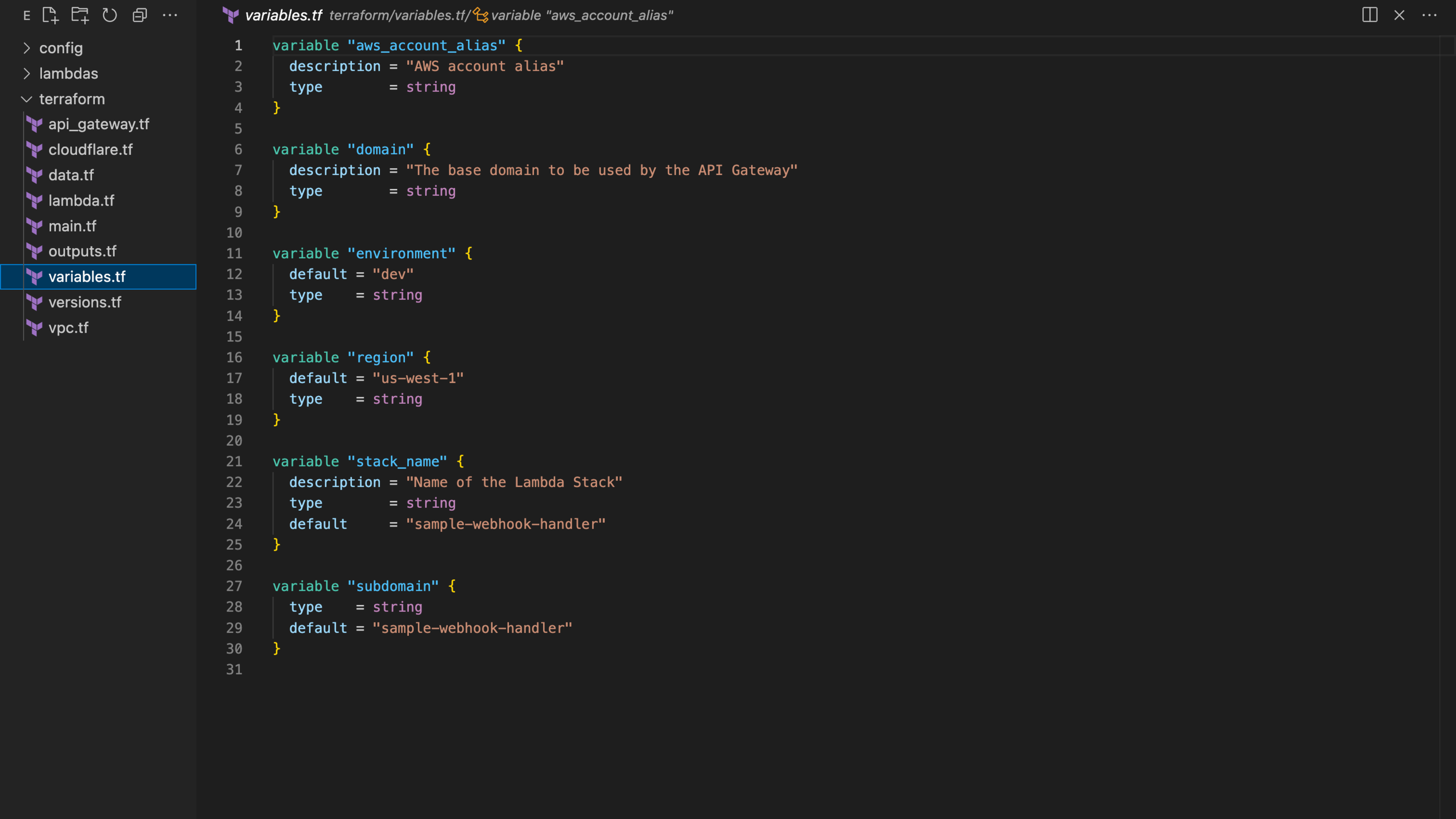

versions.tf terraform/versions.tf/...
config
lambdas
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf
terraform {
backend "s3" {}
required_providers {
aws = {
source = "hashicorp/aws"
version = "~> 4.22"
}
cloudflare = {
source = "cloudflare/cloudflare"
version = "~> 3.9.1"
}
docker = {
source = "kreuzwerker/docker"
version = "~> 3.0"
}
}
required_version = ">= 1.2"
}

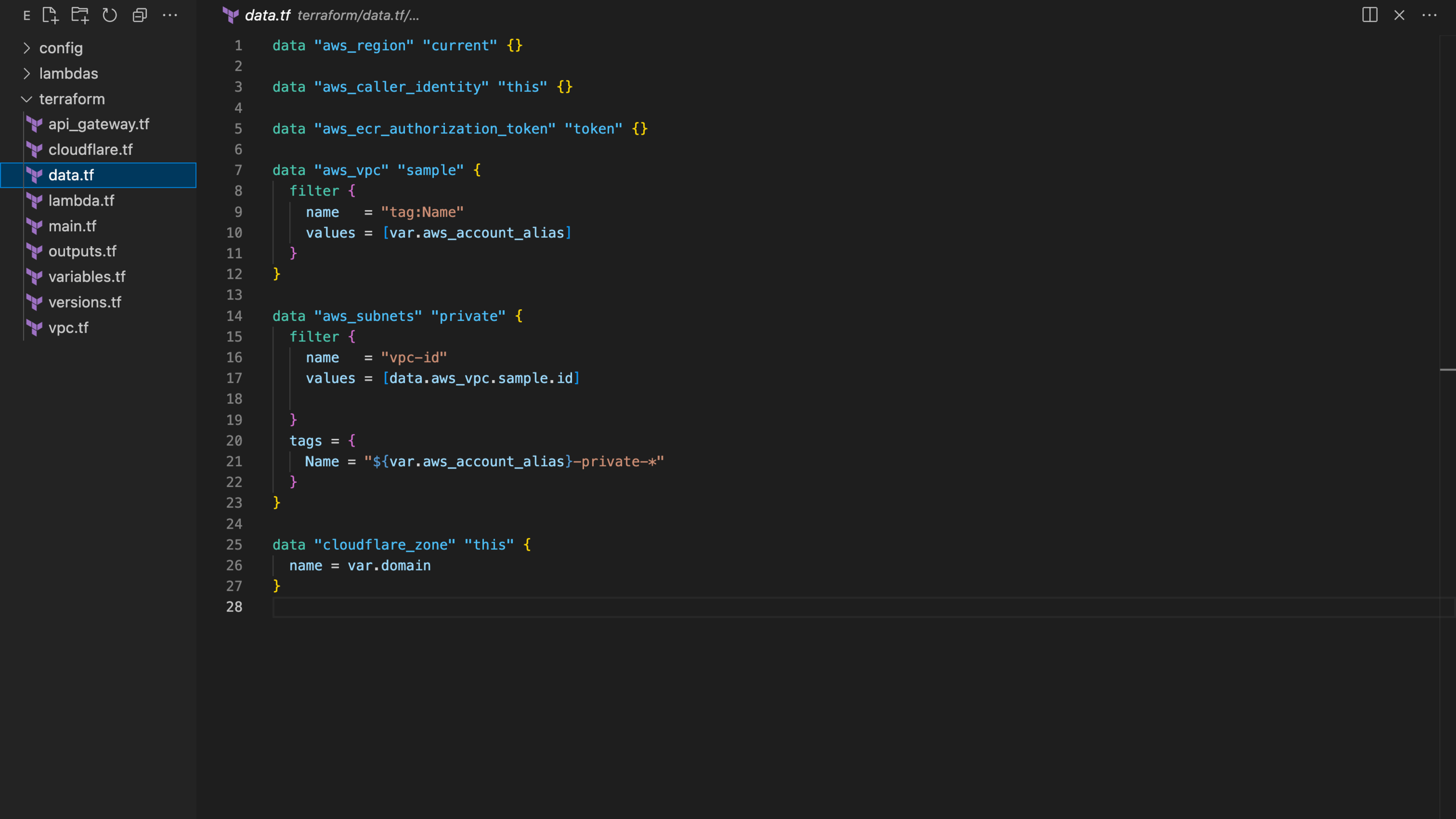

versions.tf terraform/versions.tf/...
config
lambdas
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf
terraform {
backend "s3" {}
required_providers {
aws = {
source = "hashicorp/aws"
version = "~> 4.22"
}
cloudflare = {
source = "cloudflare/cloudflare"
version = "~> 3.9.1"
}
docker = {
source = "kreuzwerker/docker"
version = "~> 3.0"
}
}
required_version = ">= 1.2"
}

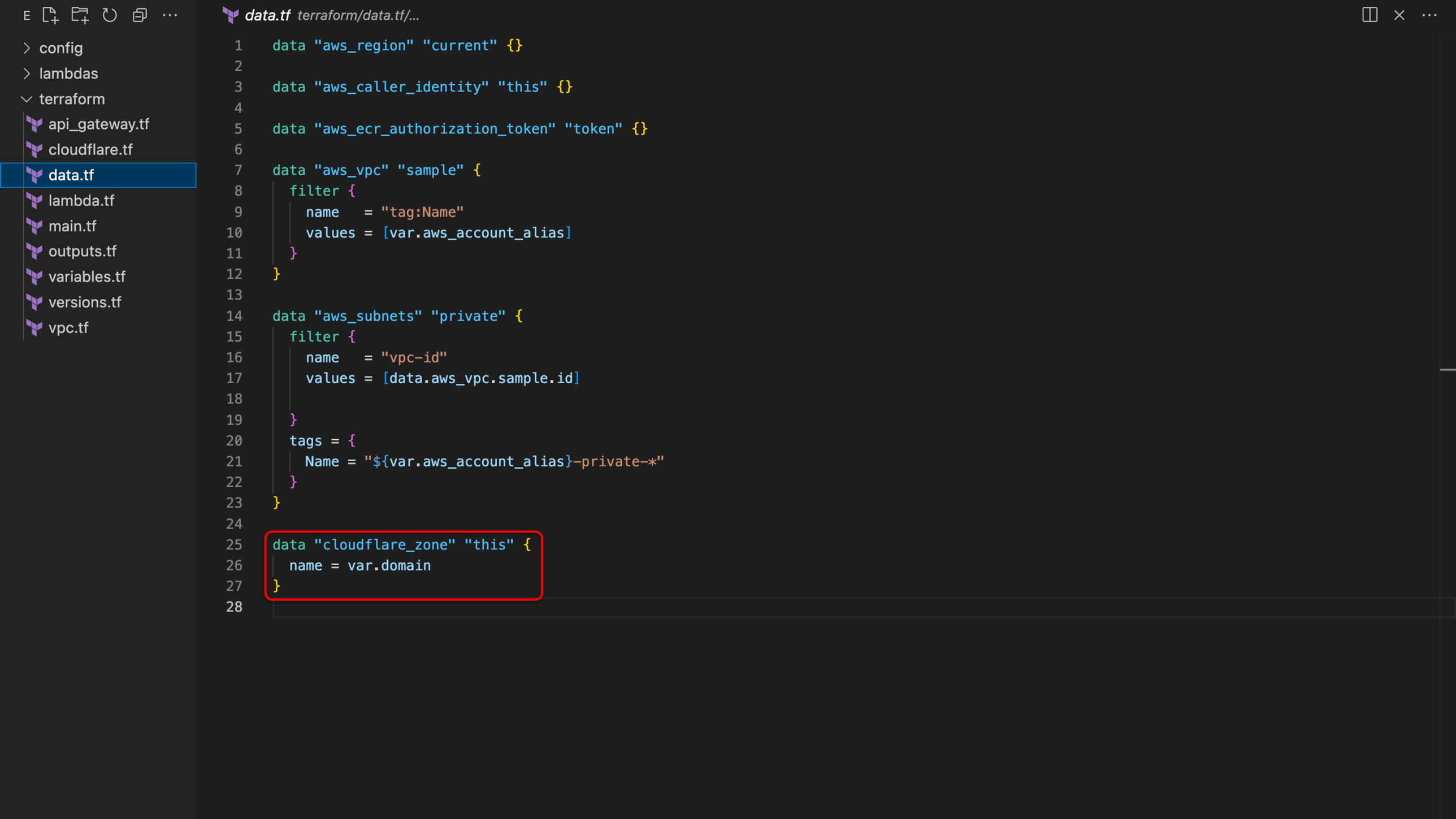

config
lambdas
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf
versions.tf terraform/versions.tf/...
terraform {
backend "s3" {}
required_providers {
aws = {
source = "hashicorp/aws"
version = "~> 4.22"
}
cloudflare = {
source = "cloudflare/cloudflare"
version = "~> 3.9.1"
}
docker = {
source = "kreuzwerker/docker"
version = "~> 3.0"
}
}
required_version = ">= 1.2"
}

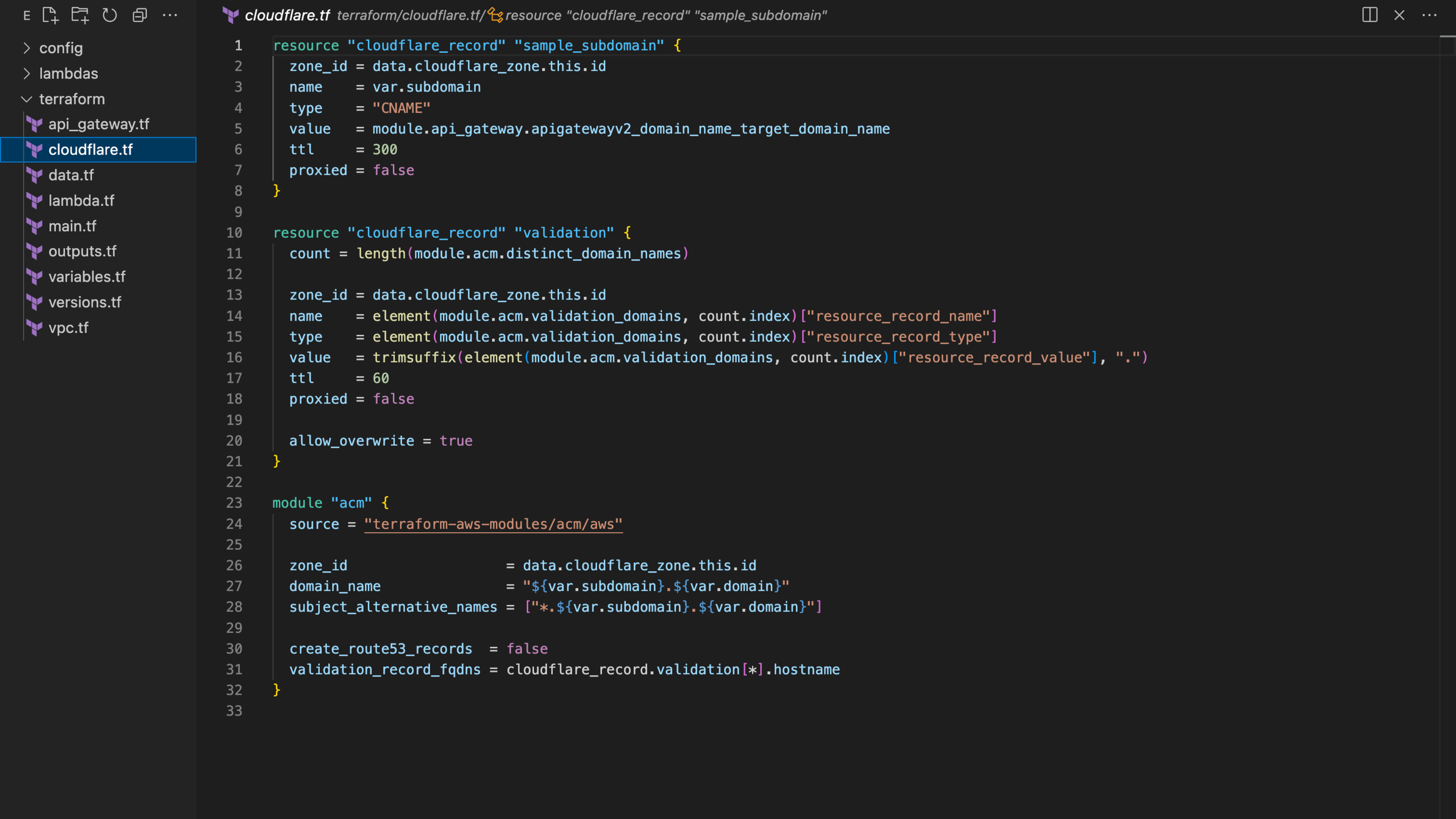

main.tf terraform/main.tf/ provider "aws"
config
lambdas
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf
provider "aws" {
region = var.region
default_tags {
tags = {
environment = var.environment
owner = "IT"
application = "Sample Webhook Handler"
}
}
}
provider "cloudflare" {}
provider "docker" {
registry_auth {
address = "${data.aws_caller_identity.this.account_id}.dkr.ecr.${data.aws_region.current.name}.amazonaws.com"
username = data.aws_ecr_authorization_token.token.user_name
password = data.aws_ecr_authorization_token.token.password
}
}

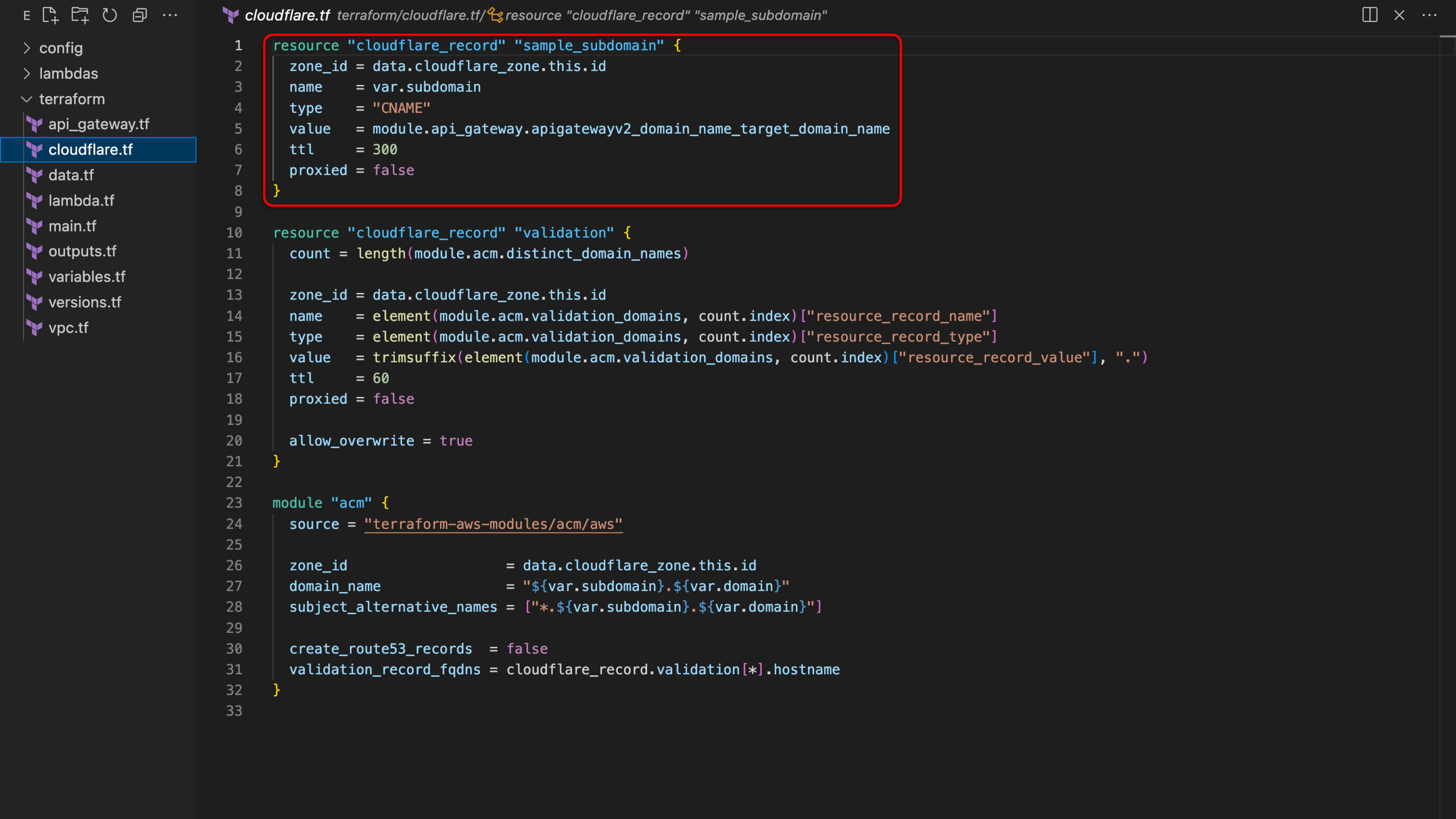

E
config
lambdas
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf
main.tf terraform/main.tf/ provider "aws"
provider "aws" {
region = var.region
default_tags {
tags = {
environment = var.environment
owner = "IT"
application = "Sample Webhook Handler"
}
}
}
provider "cloudflare" {}
provider "docker" {
registry_auth {
address = "${data.aws_caller_identity.this.account_id}.dkr.ecr.${data.aws_region.current.name}.amazonaws.com"
username = data.aws_ecr_authorization_token.token.user_name
password = data.aws_ecr_authorization_token.token.password
}
}

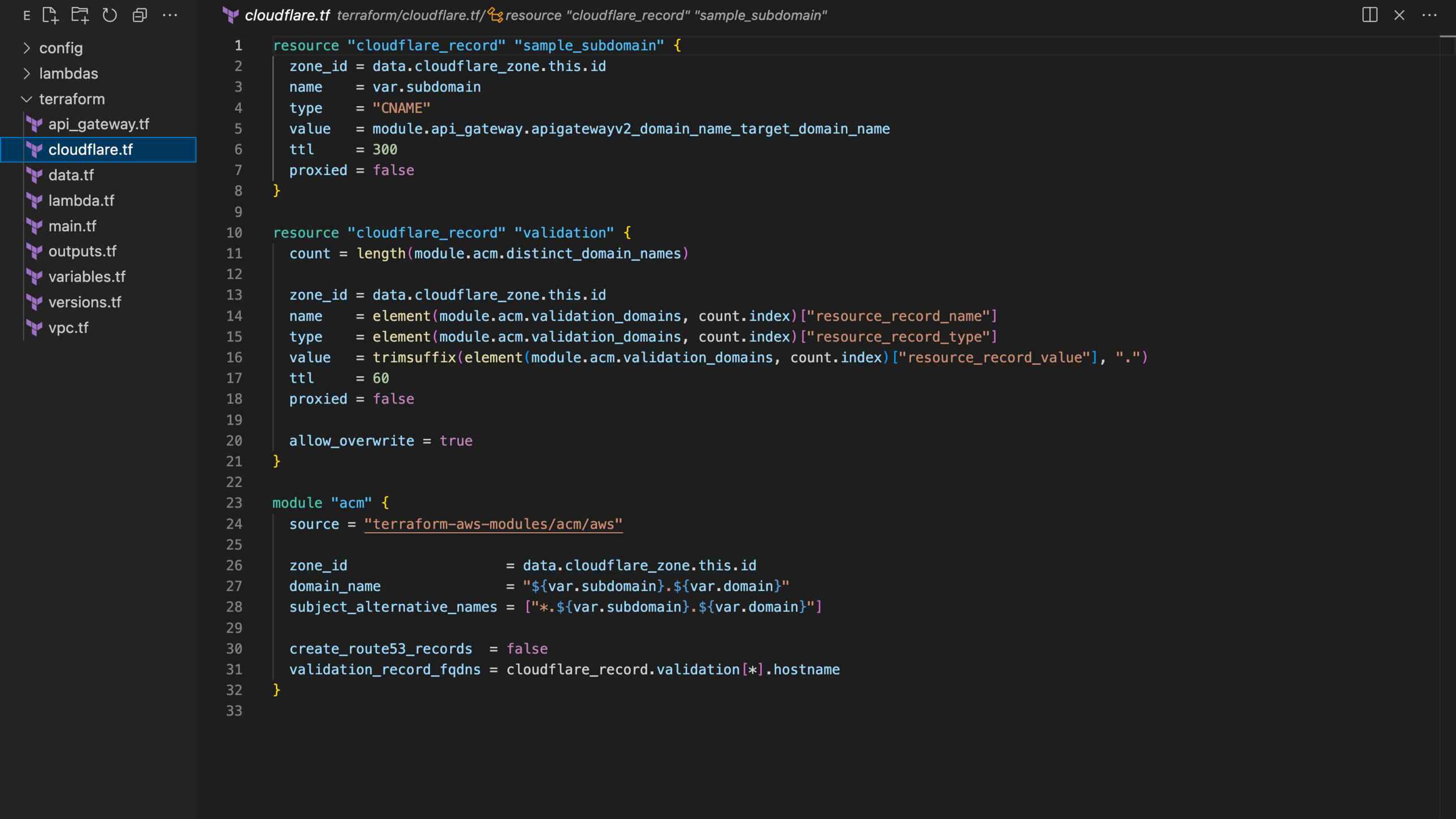

variables.tf terraform/variables.tf/ variable "aws_account_alias"
config
lambdas
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf
variable "aws_account_alias" {
description = "AWS account alias"
type = string
}
variable "domain" {
description = "The base domain to be used by the API Gateway"
type = string
}
variable "environment" {
default = "dev"
type = string
}
variable "region" {
default = "us-west-1"
type = string
}
variable "stack_name" {
description = "Name of the Lambda Stack"
type = string
default = "sample-webhook-handler"
}
variable "subdomain" {
type = string
default = "sample-webhook-handler"
}

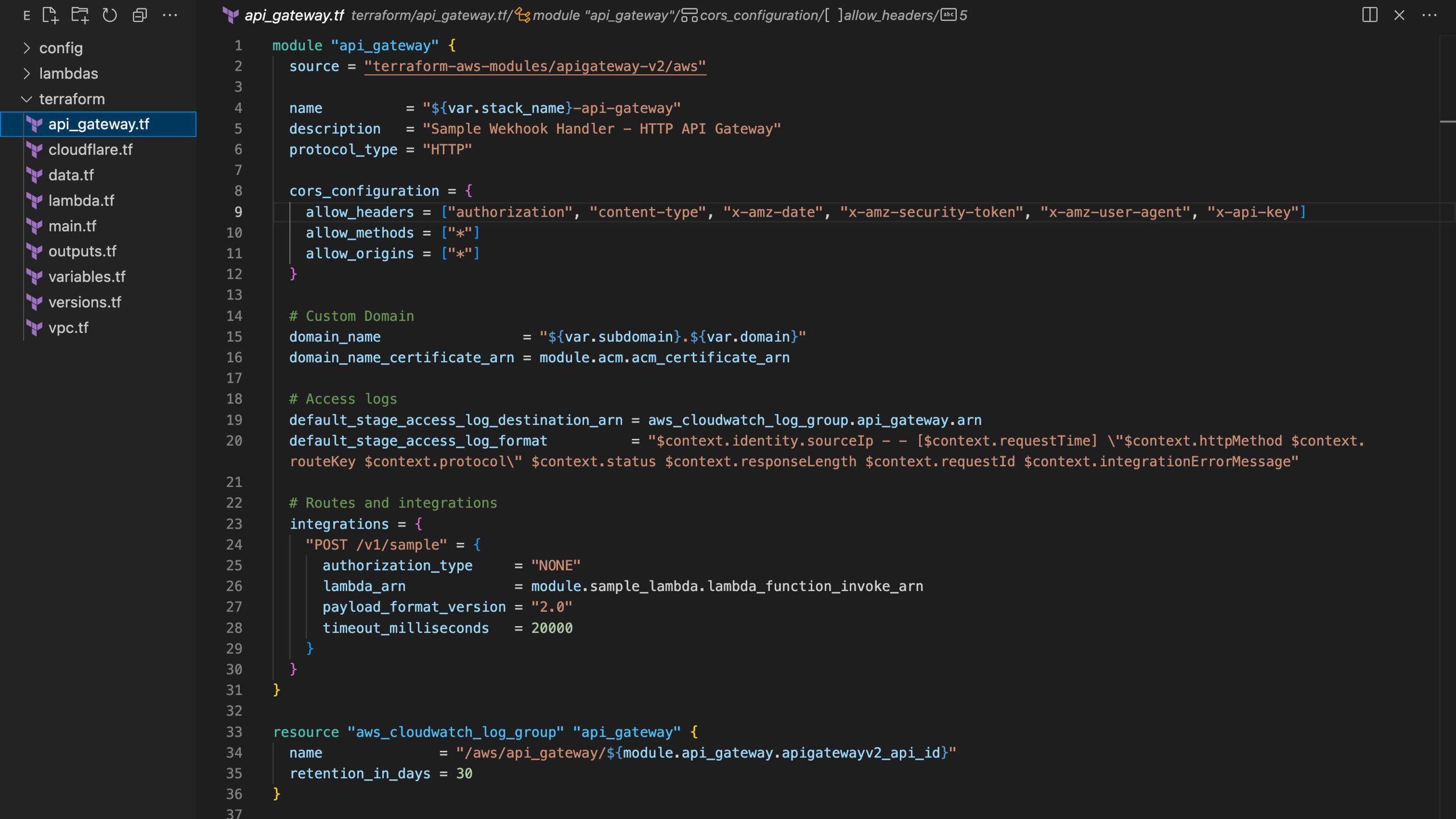

data.tf terraform/data.tf/...
config
lambdas
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf
data "aws_region" "current" {}
data "aws_caller_identity" "this" {}
data "aws_ecr_authorization_token" "token" {}
data "aws_vpc" "sample" {
filter {
name = "tag:Name"
values = [var.aws_account_alias]
}
}
data "aws_subnets" "private" {
filter {
name = "vpc-id"
values = [data.aws_vpc.sample.id]
}
tags = {
Name = "${var.aws_account_alias}-private-*"
}
}
data "cloudflare_zone" "this" {
name = var.domain
}

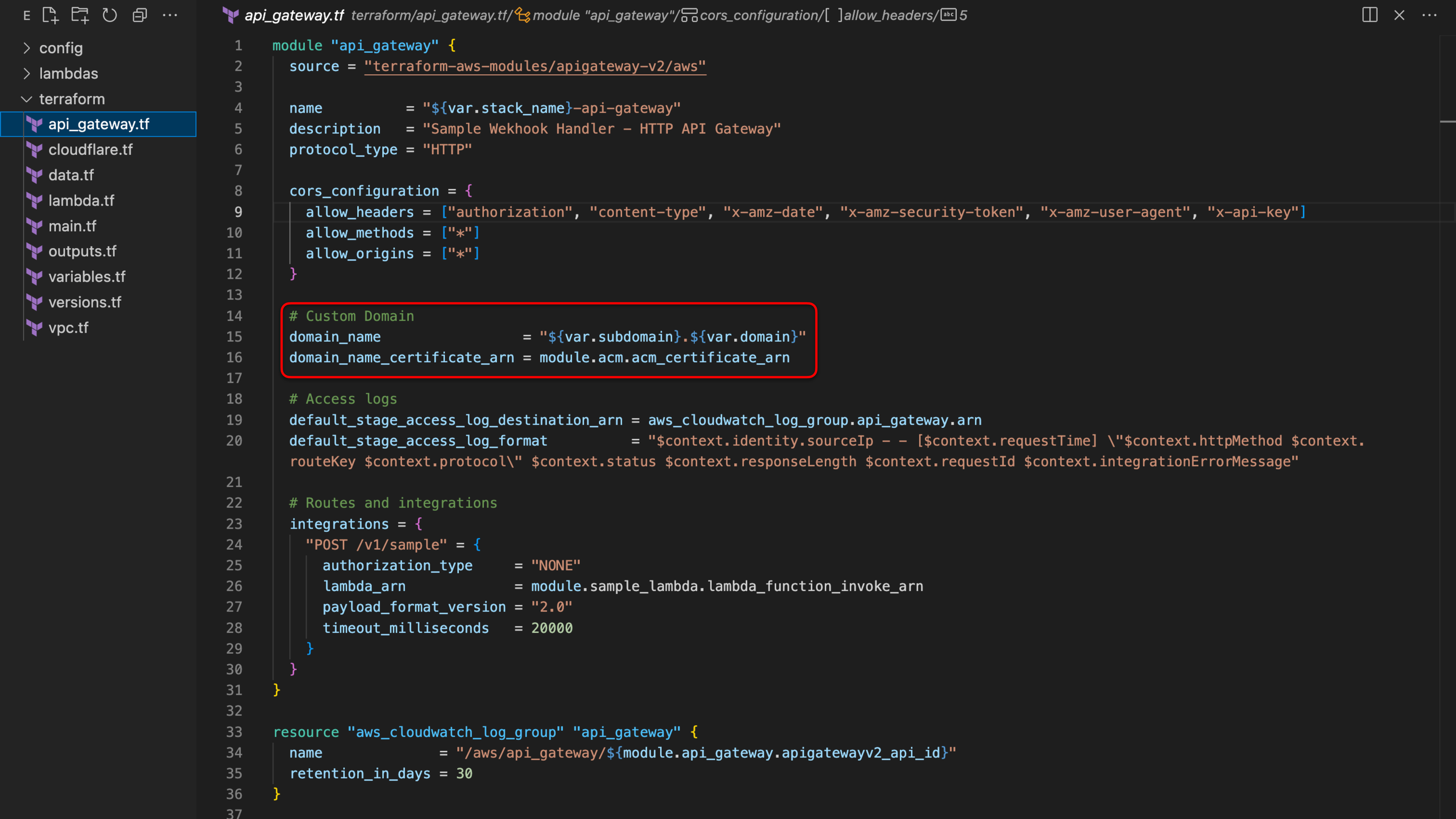

config
lambdas
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf
data.tf terraform/data.tf/...
data "aws_region" "current" {}
data "aws_caller_identity" "this" {}
data "aws_ecr_authorization_token" "token" {}
data "aws_vpc" "sample" {
filter {
name = "tag:Name"
values = [var.aws_account_alias]
}
}
data "aws_subnets" "private" {
filter {
name = "vpc-id"
values = [data.aws_vpc.sample.id]
}
tags = {
Name = "${var.aws_account_alias}-private-*"
}
}
data "cloudflare_zone" "this" {
name = var.domain
}


cloudflare.tf terraform/cloudflare.tf/ resource "cloudflare_record" "sample_subdomain"
config
lambdas
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf
resource "cloudflare_record" "sample_subdomain" {
zone_id = data.cloudflare_zone.this.id
name = var.subdomain
type = "CNAME"
value = module.api_gateway.apigatewayv2_domain_name_target_domain_name
ttl = 300
proxied = false
}
resource "cloudflare_record" "validation" {
count = length(module.acm.distinct_domain_names)
zone_id = data.cloudflare_zone.this.id
name = element(module.acm.validation_domains, count.index)["resource_record_name"]
type = element(module.acm.validation_domains, count.index)["resource_record_type"]
value = trimsuffix(element(module.acm.validation_domains, count.index)["resource_record_value"], ".")
ttl = 60
proxied = false
allow_overwrite = true
}
module "acm" {
source = "terraform-aws-modules/acm/aws"
zone_id = data.cloudflare_zone.this.id
domain_name = "${var.subdomain}.${var.domain}"
subject_alternative_names = ["*.${var.subdomain}.${var.domain}"]
create_route53_records = false
validation_record_fqdns = cloudflare_record.validation[*].hostname
}


cloudflare.tf terraform/cloudflare.tf/ resource "cloudflare_record" "sample_subdomain"
config
lambdas
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf
resource "cloudflare_record" "sample_subdomain" {
zone_id = data.cloudflare_zone.this.id
name = var.subdomain
type = "CNAME"
value = module.api_gateway.apigatewayv2_domain_name_target_domain_name
ttl = 300
proxied = false
}
resource "cloudflare_record" "validation" {
count = length(module.acm.distinct_domain_names)
zone_id = data.cloudflare_zone.this.id
name = element(module.acm.validation_domains, count.index)["resource_record_name"]
type = element(module.acm.validation_domains, count.index)["resource_record_type"]
value = trimsuffix(element(module.acm.validation_domains, count.index)["resource_record_value"], ".")
ttl = 60
proxied = false
allow_overwrite = true
}
module "acm" {
source = "terraform-aws-modules/acm/aws"
zone_id = data.cloudflare_zone.this.id
domain_name = "${var.subdomain}.${var.domain}"
subject_alternative_names = ["*.${var.subdomain}.${var.domain}"]
create_route53_records = false
validation_record_fqdns = cloudflare_record.validation[*].hostname
}


cloudflare.tf terraform/cloudflare.tf/ resource "cloudflare_record" "sample_subdomain"
config
lambdas
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf
resource "cloudflare_record" "sample_subdomain" {
zone_id = data.cloudflare_zone.this.id
name = var.subdomain
type = "CNAME"
value = module.api_gateway.apigatewayv2_domain_name_target_domain_name
ttl = 300
proxied = false
}
resource "cloudflare_record" "validation" {
count = length(module.acm.distinct_domain_names)
zone_id = data.cloudflare_zone.this.id
name = element(module.acm.validation_domains, count.index)["resource_record_name"]
type = element(module.acm.validation_domains, count.index)["resource_record_type"]
value = trimsuffix(element(module.acm.validation_domains, count.index)["resource_record_value"], ".")
ttl = 60
proxied = false
allow_overwrite = true
}
module "acm" {
source = "terraform-aws-modules/acm/aws"
zone_id = data.cloudflare_zone.this.id
domain_name = "${var.subdomain}.${var.domain}"
subject_alternative_names = ["*.${var.subdomain}.${var.domain}"]
create_route53_records = false
validation_record_fqdns = cloudflare_record.validation[*].hostname
}


config
lambdas
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf
api_gateway.tf terraform/api_gateway.tf/ module "api_gateway"/ cors_configuration/[ ]allow_headers/ 5
module "api_gateway" {
source = "terraform-aws-modules/apigateway-v2/aws"
name = "${var.stack_name}-api-gateway"
description = "Sample Wekhook Handler - HTTP API Gateway"
protocol_type = "HTTP"
cors_configuration = {
allow_headers = ["authorization", "content-type", "x-amz-date", "x-amz-security-token", "x-amz-user-agent", "x-api-key"]
allow_methods = ["*"]
allow_origins = ["*"]
}
# Custom Domain
domain_name = "${var.subdomain}.${var.domain}"
domain_name_certificate_arn = module.acm.acm_certificate_arn
# Access logs
default_stage_access_log_destination_arn = aws_cloudwatch_log_group.api_gateway.arn
default_stage_access_log_format = "$context.identity.sourceIp - - [$context.requestTime] \"$context.httpMethod $context.
routeKey $context.protocol\" $context.status $context.responseLength $context.requestId $context.integrationErrorMessage"
# Routes and integrations
integrations = {
"POST /v1/sample" = {
authorization_type = "NONE"
lambda_arn = module.sample_lambda.lambda_function_invoke_arn
payload_format_version = "2.0"
timeout_milliseconds = 20000
}
}
}
resource "aws_cloudwatch_log_group" "api_gateway" {
name = "/aws/api_gateway/${module.api_gateway.apigatewayv2_api_id}"
retention_in_days = 30
}

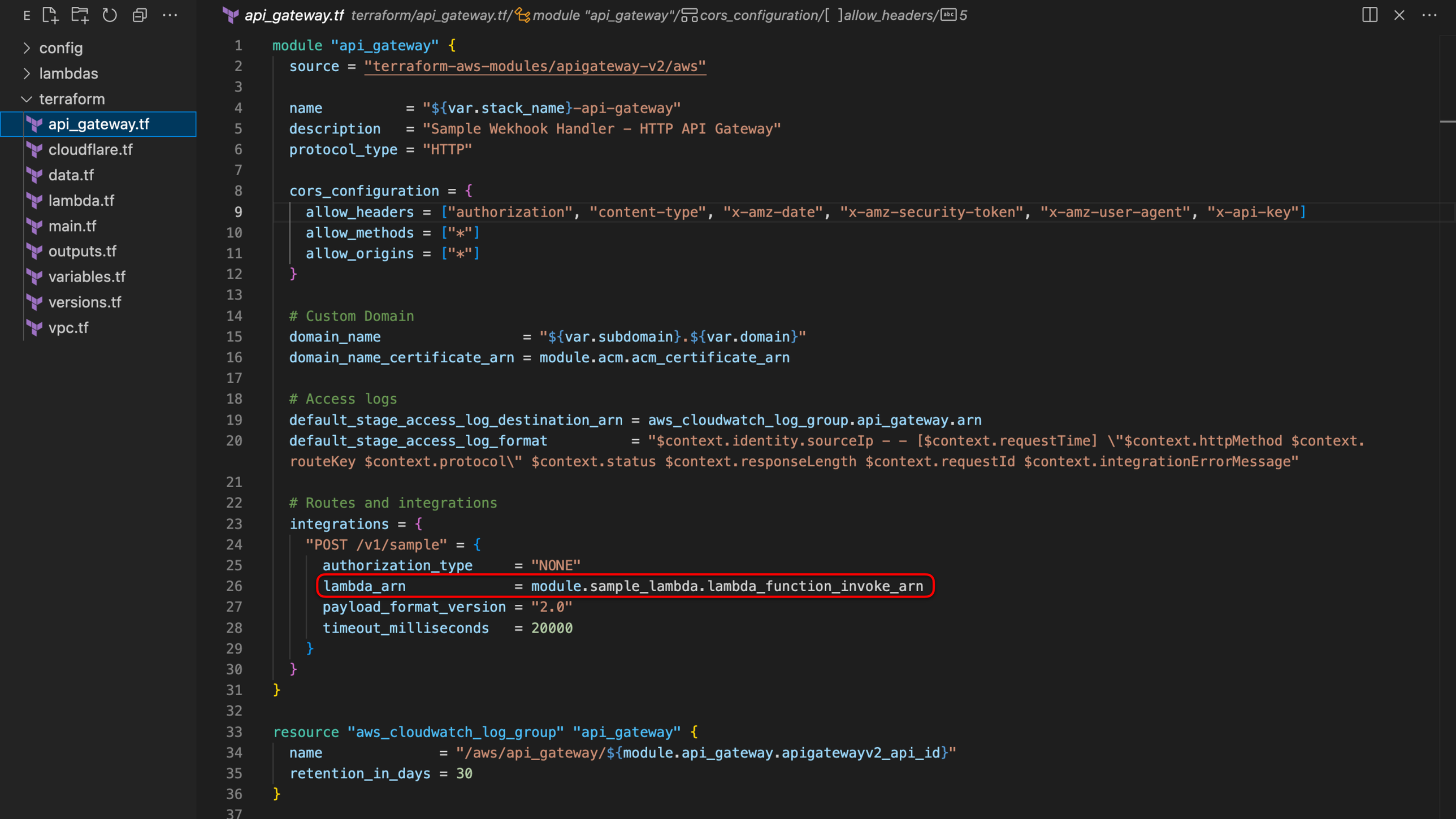

config
lambdas
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf
api_gateway.tf terraform/api_gateway.tf/ module "api_gateway"/ cors_configuration/[ ]allow_headers/ 5
module "api_gateway" {
source = "terraform-aws-modules/apigateway-v2/aws"
name = "${var.stack_name}-api-gateway"
description = "Sample Wekhook Handler - HTTP API Gateway"
protocol_type = "HTTP"
cors_configuration = {
allow_headers = ["authorization", "content-type", "x-amz-date", "x-amz-security-token", "x-amz-user-agent", "x-api-key"]
allow_methods = ["*"]
allow_origins = ["*"]
}
# Custom Domain
domain_name = "${var.subdomain}.${var.domain}"
domain_name_certificate_arn = module.acm.acm_certificate_arn
# Access logs
default_stage_access_log_destination_arn = aws_cloudwatch_log_group.api_gateway.arn
default_stage_access_log_format = "$context.identity.sourceIp - - [$context.requestTime] \"$context.httpMethod $context.
routeKey $context.protocol\" $context.status $context.responseLength $context.requestId $context.integrationErrorMessage"
# Routes and integrations
integrations = {
"POST /v1/sample" = {
authorization_type = "NONE"
lambda_arn = module.sample_lambda.lambda_function_invoke_arn
payload_format_version = "2.0"
timeout_milliseconds = 20000
}
}
}
resource "aws_cloudwatch_log_group" "api_gateway" {
name = "/aws/api_gateway/${module.api_gateway.apigatewayv2_api_id}"
retention_in_days = 30
}

E
config
lambdas
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf

```hcl
module "api_gateway" {
  source = "terraform-aws-modules/apigateway-v2/aws"

  name          = "${var.stack_name}-api-gateway"
  description   = "Sample Wekhook Handler - HTTP API Gateway"
  protocol_type = "HTTP"

  cors_configuration = {
    allow_headers = ["authorization", "content-type", "x-amz-date", "x-amz-security-token", "x-amz-user-agent", "x-api-key"]
    allow_methods = ["*"]
    allow_origins = ["*"]
  }

  # Custom Domain
  domain_name                 = "${var.subdomain}.${var.domain}"
  domain_name_certificate_arn = module.acm.acm_certificate_arn

  # Access logs
  default_stage_access_log_destination_arn = aws_cloudwatch_log_group.api_gateway.arn
  default_stage_access_log_format          = "$context.identity.sourceIp - - [$context.requestTime] \"$context.httpMethod $context.
  routeKey $context.protocol\" $context.status $context.responseLength $context.requestId $context.integrationErrorMessage"

  # Routes and integrations
  integrations = {
    "POST /v1/sample" = {
      authorization_type     = "NONE"
      lambda_arn             = module.sample_lambda.lambda_function_invoke_arn
      payload_format_version = "2.0"
      timeout_milliseconds   = 20000
    }
  }
}

resource "aws_cloudwatch_log_group" "api_gateway" {
  name              = "/aws/api_gateway/${module.api_gateway.apigatewayv2_api_id}"
  retention_in_days = 30
}
```

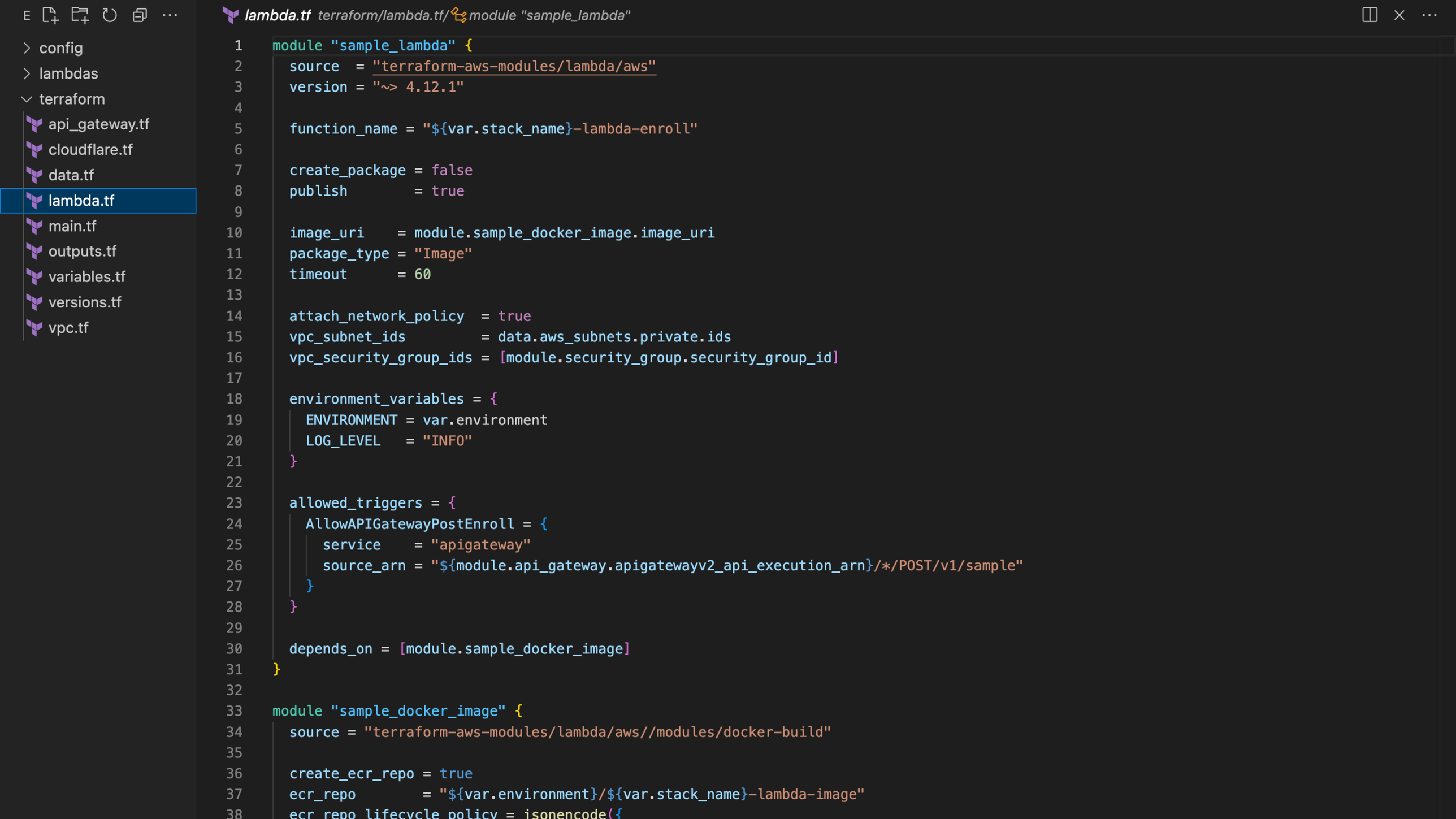
- config
- lambdas
- terraform
  - api_gateway.tf
  - cloudflare.tf
  - data.tf
  - lambda.tf
  - main.tf
  - outputs.tf
  - variables.tf
  - versions.tf
  - vpc.tf


```hcl
module "api_gateway" {
  source = "terraform-aws-modules/apigateway-v2/aws"

  name          = "${var.stack_name}-api-gateway"
  description   = "Sample Wekhook Handler - HTTP API Gateway"
  protocol_type = "HTTP"

  cors_configuration = {
    allow_headers = ["authorization", "content-type", "x-amz-date", "x-amz-security-token", "x-amz-user-agent", "x-api-key"]
    allow_methods = ["*"]
    allow_origins = ["*"]
  }

  # Custom Domain
  domain_name                 = "${var.subdomain}.${var.domain}"
  domain_name_certificate_arn = module.acm.acm_certificate_arn

  # Access logs
  default_stage_access_log_destination_arn = aws_cloudwatch_log_group.api_gateway.arn
  default_stage_access_log_format          = "$context.identity.sourceIp - - [$context.requestTime] \"$context.httpMethod $context.
  routeKey $context.protocol\" $context.status $context.responseLength $context.requestId $context.integrationErrorMessage"

  # Routes and integrations
  integrations = {
    "POST /v1/sample" = {
      authorization_type     = "NONE"
      lambda_arn             = module.sample_lambda.lambda_function_invoke_arn
      payload_format_version = "2.0"
      timeout_milliseconds   = 20000
    }
  }
}

resource "aws_cloudwatch_log_group" "api_gateway" {
  name              = "/aws/api_gateway/${module.api_gateway.apigatewayv2_api_id}"
  retention_in_days = 30
}
```


config
lambdas
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf
api_gateway.tf terraform/api_gateway.tf/ module "api_gateway"/ cors_configuration/[ ]allow_headers/ 5
module "api_gateway" {
source = "terraform-aws-modules/apigateway-v2/aws"
name = "${var.stack_name}-api-gateway"
description = "Sample Wekhook Handler - HTTP API Gateway"
protocol_type = "HTTP"
cors_configuration = {
allow_headers = ["authorization", "content-type", "x-amz-date", "x-amz-security-token", "x-amz-user-agent", "x-api-key"]
allow_methods = ["*"]
allow_origins = ["*"]
}
# Custom Domain
domain_name = "${var.subdomain}.${var.domain}"
domain_name_certificate_arn = module.acm.acm_certificate_arn
# Access logs
default_stage_access_log_destination_arn = aws_cloudwatch_log_group.api_gateway.arn
default_stage_access_log_format = "$context.identity.sourceIp - - [$context.requestTime] \"$context.httpMethod $context.
routeKey $context.protocol\" $context.status $context.responseLength $context.requestId $context.integrationErrorMessage"
# Routes and integrations
integrations = {
"POST /v1/sample" = {
authorization_type = "NONE"
lambda_arn = module.sample_lambda.lambda_function_invoke_arn
payload_format_version = "2.0"
timeout_milliseconds = 20000
}
}
}
resource "aws_cloudwatch_log_group" "api_gateway" {
name = "/aws/api_gateway/${module.api_gateway.apigatewayv2_api_id}"
retention_in_days = 30
}

- config
- lambdas
- terraform
  - api_gateway.tf
  - cloudflare.tf
  - data.tf
  - lambda.tf
  - main.tf
  - outputs.tf
  - variables.tf
  - versions.tf
  - vpc.tf


```hcl
module "api_gateway" {
  source = "terraform-aws-modules/apigateway-v2/aws"

  name          = "${var.stack_name}-api-gateway"
  description   = "Sample Wekhook Handler - HTTP API Gateway"
  protocol_type = "HTTP"

  cors_configuration = {
    allow_headers = ["authorization", "content-type", "x-amz-date", "x-amz-security-token", "x-amz-user-agent", "x-api-key"]
    allow_methods = ["*"]
    allow_origins = ["*"]
  }

  # Custom Domain
  domain_name                 = "${var.subdomain}.${var.domain}"
  domain_name_certificate_arn = module.acm.acm_certificate_arn

  # Access logs
  default_stage_access_log_destination_arn = aws_cloudwatch_log_group.api_gateway.arn
  default_stage_access_log_format          = "$context.identity.sourceIp - - [$context.requestTime] \"$context.httpMethod $context.
  routeKey $context.protocol\" $context.status $context.responseLength $context.requestId $context.integrationErrorMessage"

  # Routes and integrations
  integrations = {
    "POST /v1/sample" = {
      authorization_type     = "NONE"
      lambda_arn             = module.sample_lambda.lambda_function_invoke_arn
      payload_format_version = "2.0"
      timeout_milliseconds   = 20000
    }
  }
}

resource "aws_cloudwatch_log_group" "api_gateway" {
  name              = "/aws/api_gateway/${module.api_gateway.apigatewayv2_api_id}"
  retention_in_days = 30
}
```

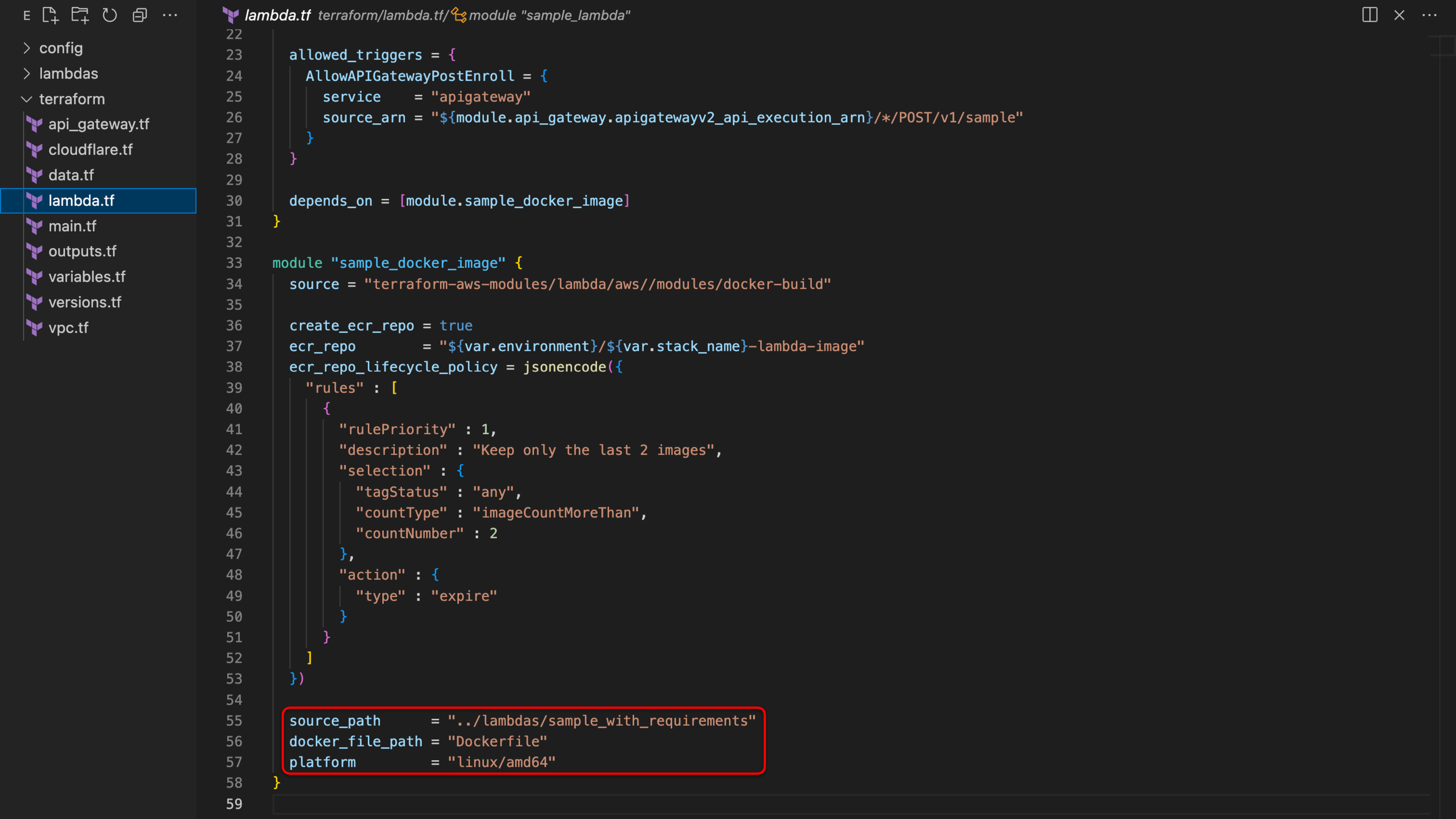

lambda.tf terraform/lambda.tf/ module "sample_lambda"
config
lambdas
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf
module "sample_lambda" {
source = "terraform-aws-modules/lambda/aws"
version = "~> 4.12.1"
function_name = "${var.stack_name}-lambda-enroll"
create_package = false
publish = true
image_uri = module.sample_docker_image.image_uri
package_type = "Image"
timeout = 60
attach_network_policy = true
vpc_subnet_ids = data.aws_subnets.private.ids
vpc_security_group_ids = [module.security_group.security_group_id]
environment_variables = {
ENVIRONMENT = var.environment
LOG_LEVEL = "INFO"
}
allowed_triggers = {
AllowAPIGatewayPostEnroll = {
service = "apigateway"
source_arn = "${module.api_gateway.apigatewayv2_api_execution_arn}/*/POST/v1/sample"
}
}
depends_on = [module.sample_docker_image]
}
module "sample_docker_image" {
source = "terraform-aws-modules/lambda/aws//modules/docker-build"
create_ecr_repo = true
ecr_repo = "${var.environment}/${var.stack_name}-lambda-image"
ecr_repo_lifecycle_policy = jsonencode({

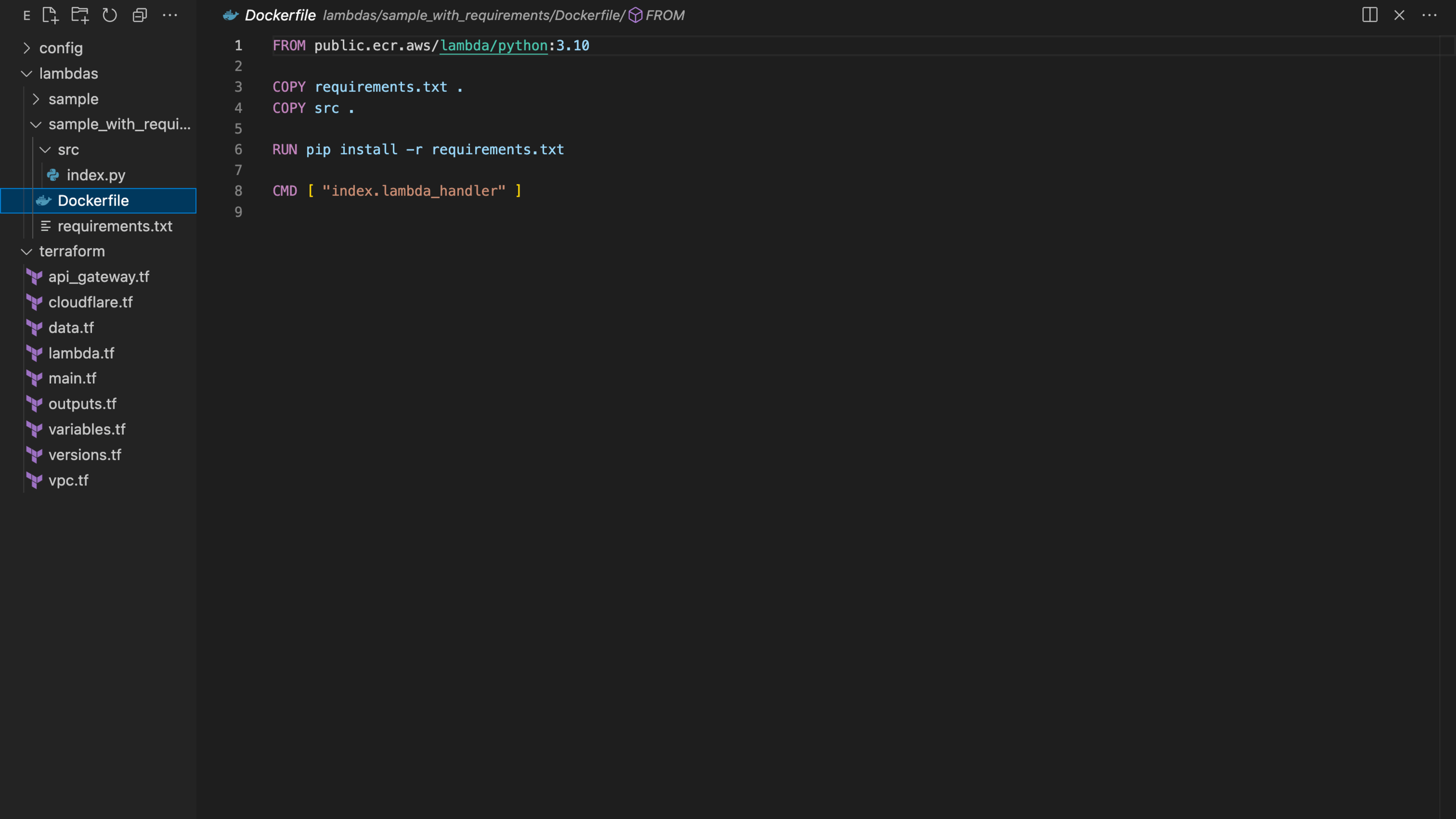
- config
- lambdas
- terraform
  - api_gateway.tf
  - cloudflare.tf
  - data.tf
  - lambda.tf
  - main.tf
  - outputs.tf
  - variables.tf
  - versions.tf
  - vpc.tf

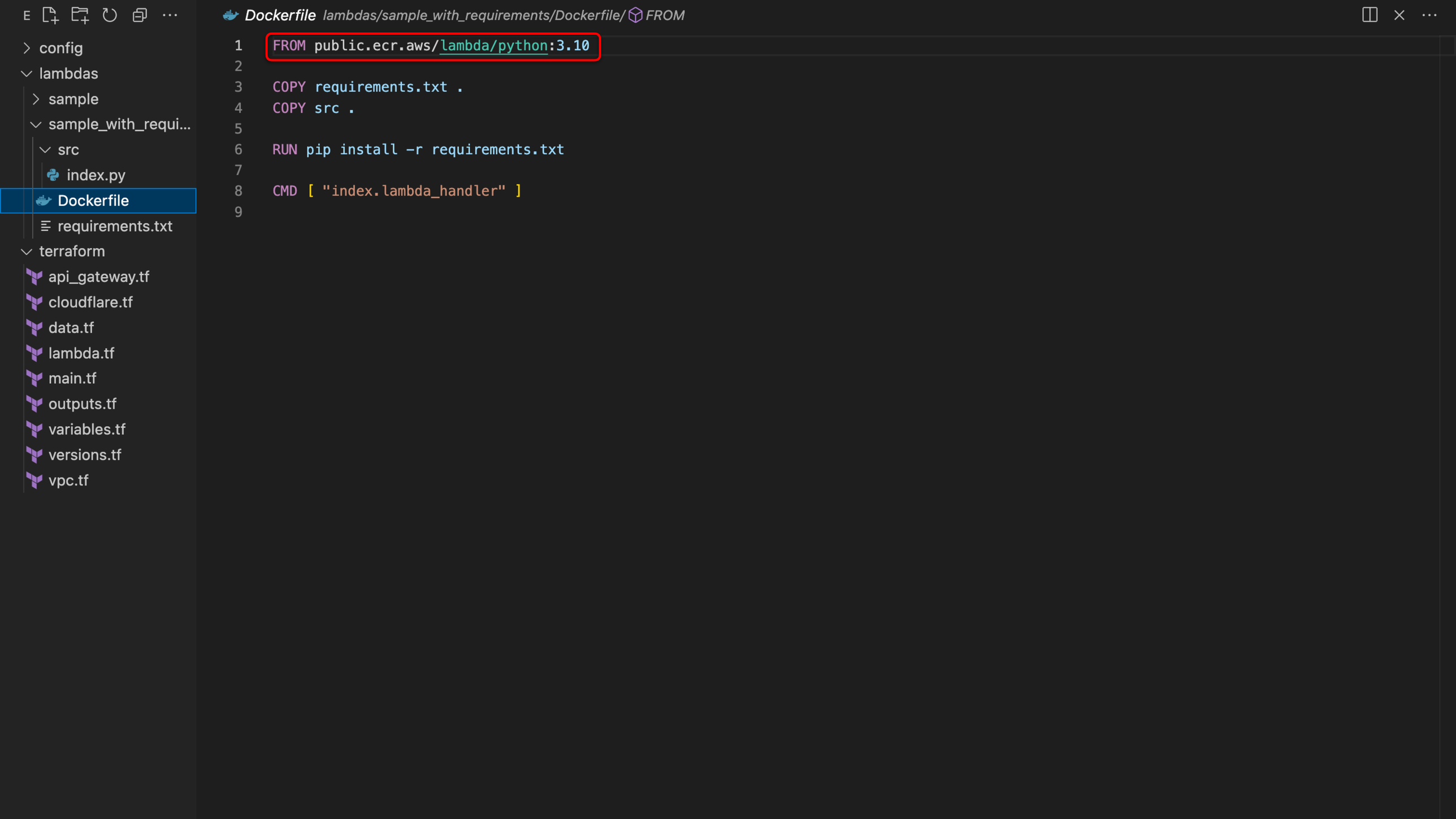

```
    allowed_triggers = {
      AllowAPIGatewayPostEnroll = {
        service    = "apigateway"
        source_arn = "${module.api_gateway.apigatewayv2_api_execution_arn}/*/POST/v1/sample"
      }
    }

    depends_on = [module.sample_docker_image]
  }

  module "sample_docker_image" {
    source = "terraform-aws-modules/lambda/aws//modules/docker-build"

    create_ecr_repo = true
    ecr_repo        = "${var.environment}/${var.stack_name}-lambda-image"
    ecr_repo_lifecycle_policy = jsonencode({
      "rules" : [
        {
          "rulePriority" : 1,
          "description" : "Keep only the last 2 images",
          "selection" : {
            "tagStatus" : "any",
            "countType" : "imageCountMoreThan",
            "countNumber" : 2
          },
          "action" : {
            "type" : "expire"
          }
        }
      ]
    })

    source_path      = "../lambdas/sample_with_requirements"
    docker_file_path = "Dockerfile"
    platform         = "linux/amd64"
  }
```

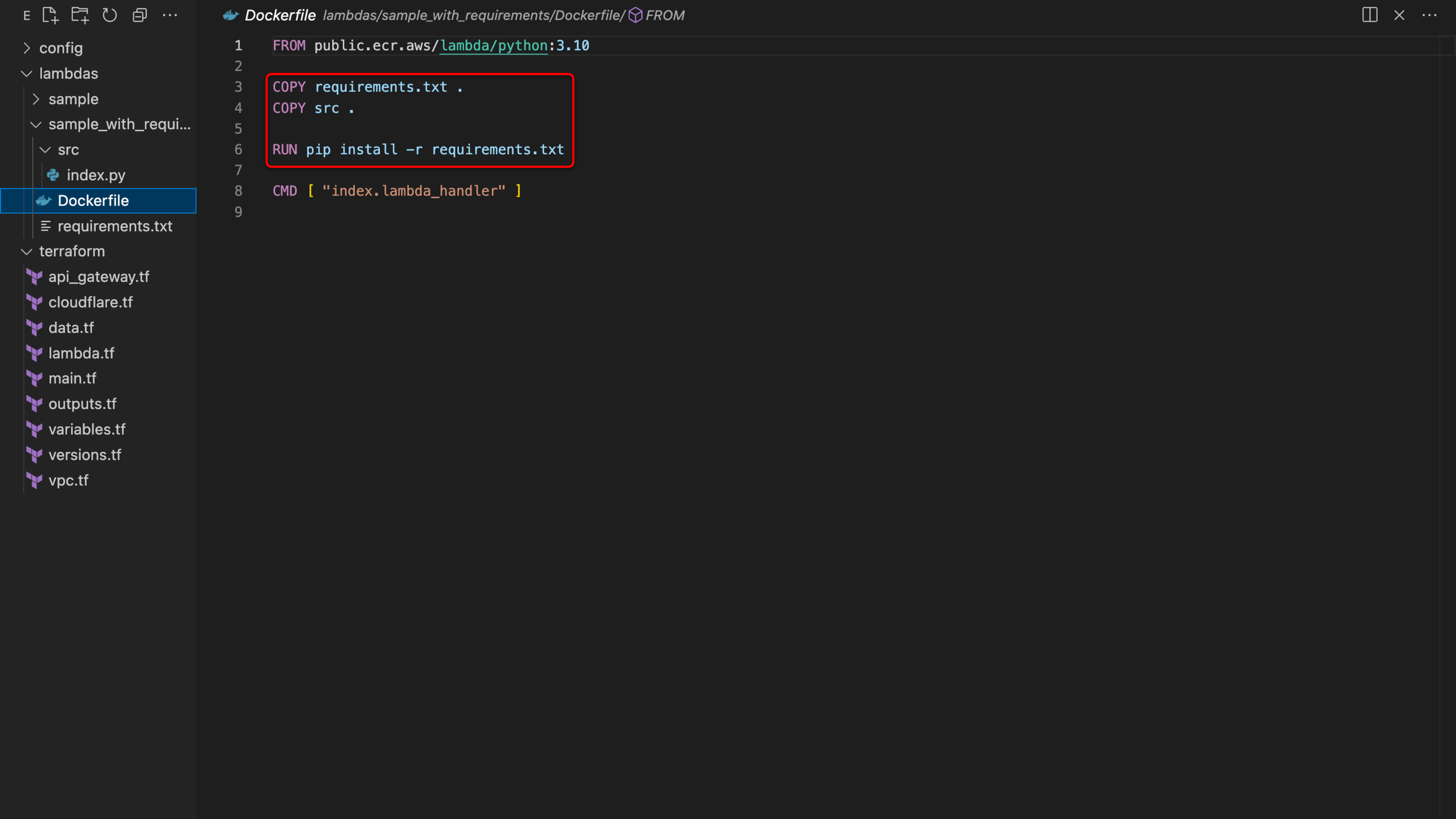
- config
- lambdas
- terraform
  - api_gateway.tf
  - cloudflare.tf
  - data.tf
  - lambda.tf
  - main.tf
  - outputs.tf
  - variables.tf
  - versions.tf
  - vpc.tf

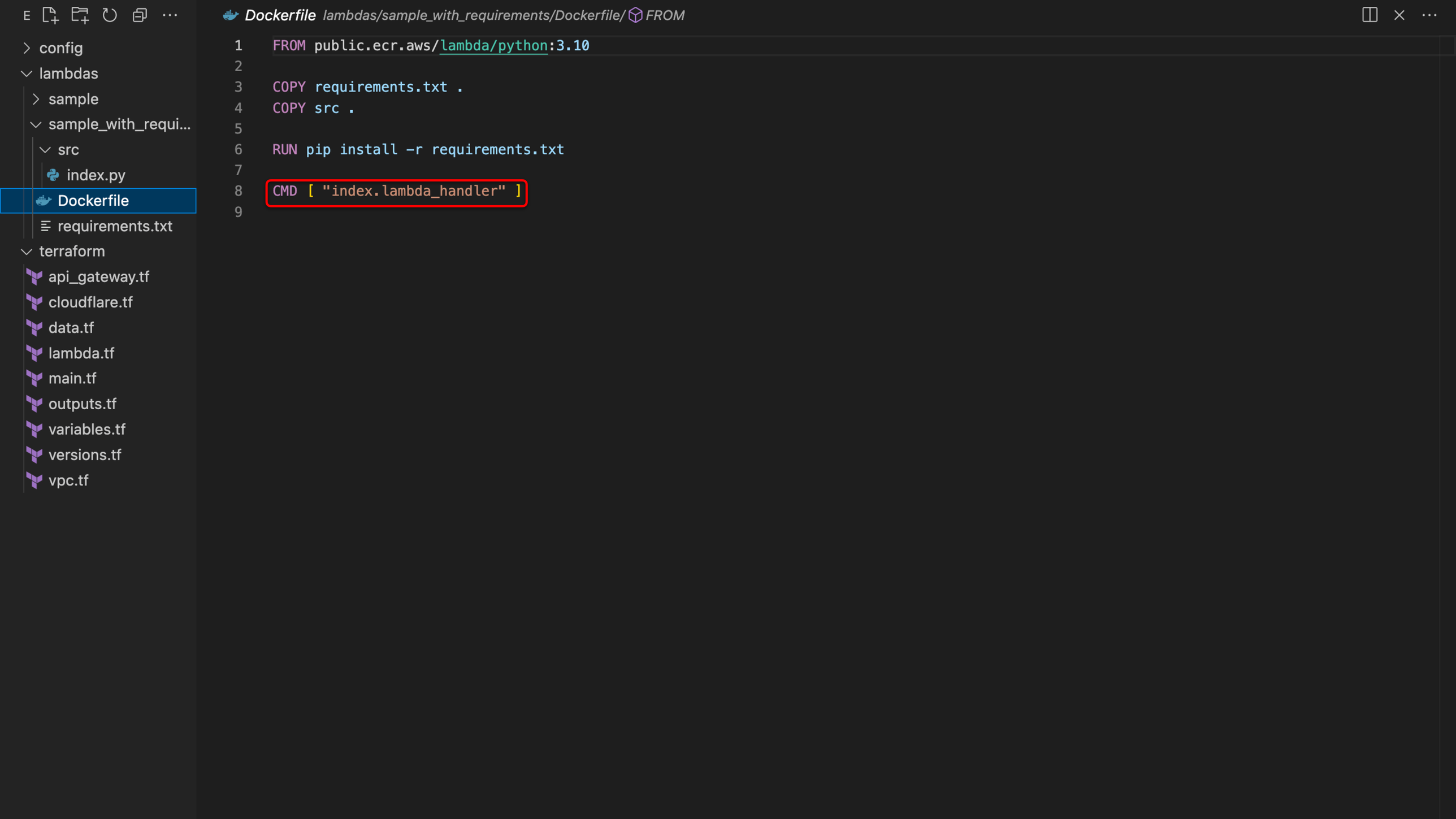

```
22
23    allowed_triggers = {
24      AllowAPIGatewayPostEnroll = {
25        service    = "apigateway"
26        source_arn = "${module.api_gateway.apigatewayv2_api_execution_arn}/*/POST/v1/sample"
27      }
28    }
29
30    depends_on = [module.sample_docker_image]
31  }
32
33  module "sample_docker_image" {
34    source = "terraform-aws-modules/lambda/aws//modules/docker-build"
35
36    create_ecr_repo = true
37    ecr_repo        = "${var.environment}/${var.stack_name}-lambda-image"
38    ecr_repo_lifecycle_policy = jsonencode({
39      "rules" : [
40        {
41          "rulePriority" : 1,
42          "description" : "Keep only the last 2 images",
43          "selection" : {
44            "tagStatus" : "any",
45            "countType" : "imageCountMoreThan",
46            "countNumber" : 2
47          },
48          "action" : {
49            "type" : "expire"
50          }
51        }
52      ]
53    })
54
55    source_path      = "../lambdas/sample_with_requirements"
56    docker_file_path = "Dockerfile"
57    platform         = "linux/amd64"
58  }
59
```

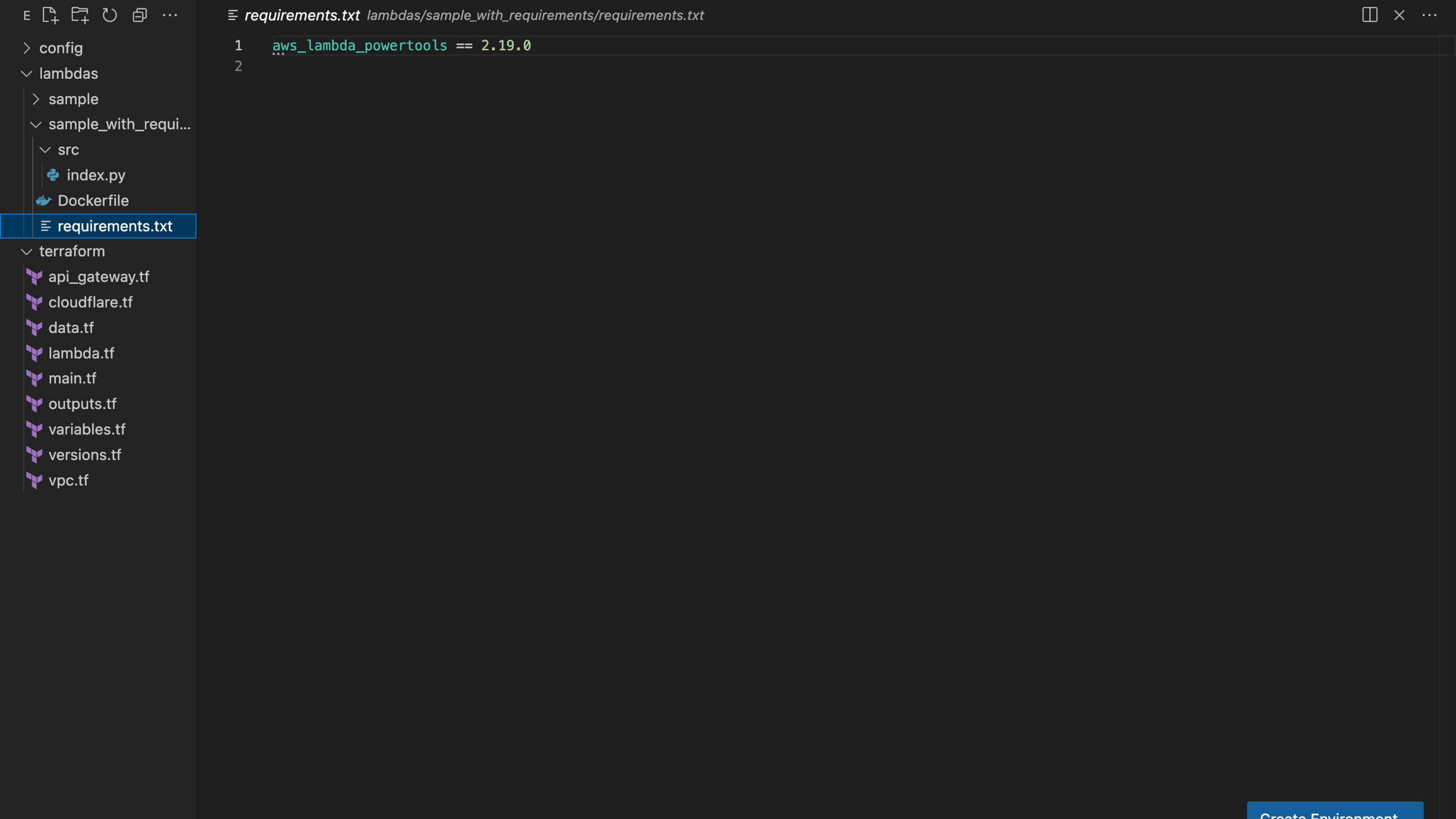

config
lambdas
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf
lambda.tf terraform/lambda.tf/ module "sample_lambda"
allowed_triggers = {
AllowAPIGatewayPostEnroll = {
service = "apigateway"
source_arn = "${module.api_gateway.apigatewayv2_api_execution_arn}/*/POST/v1/sample"
}
}
depends_on = [module.sample_docker_image]
}
module "sample_docker_image" {
source = "terraform-aws-modules/lambda/aws//modules/docker-build"
create_ecr_repo = true
ecr_repo = "${var.environment}/${var.stack_name}-lambda-image"
ecr_repo_lifecycle_policy = jsonencode({
"rules" : [
{
"rulePriority" : 1,
"description" : "Keep only the last 2 images",
"selection" : {
"tagStatus" : "any",
"countType" : "imageCountMoreThan",
"countNumber" : 2
},
"action" : {
"type" : "expire"
}
}
]
})
source_path = "../lambdas/sample_with_requirements"
docker_file_path = "Dockerfile"
platform = "linux/amd64"
}

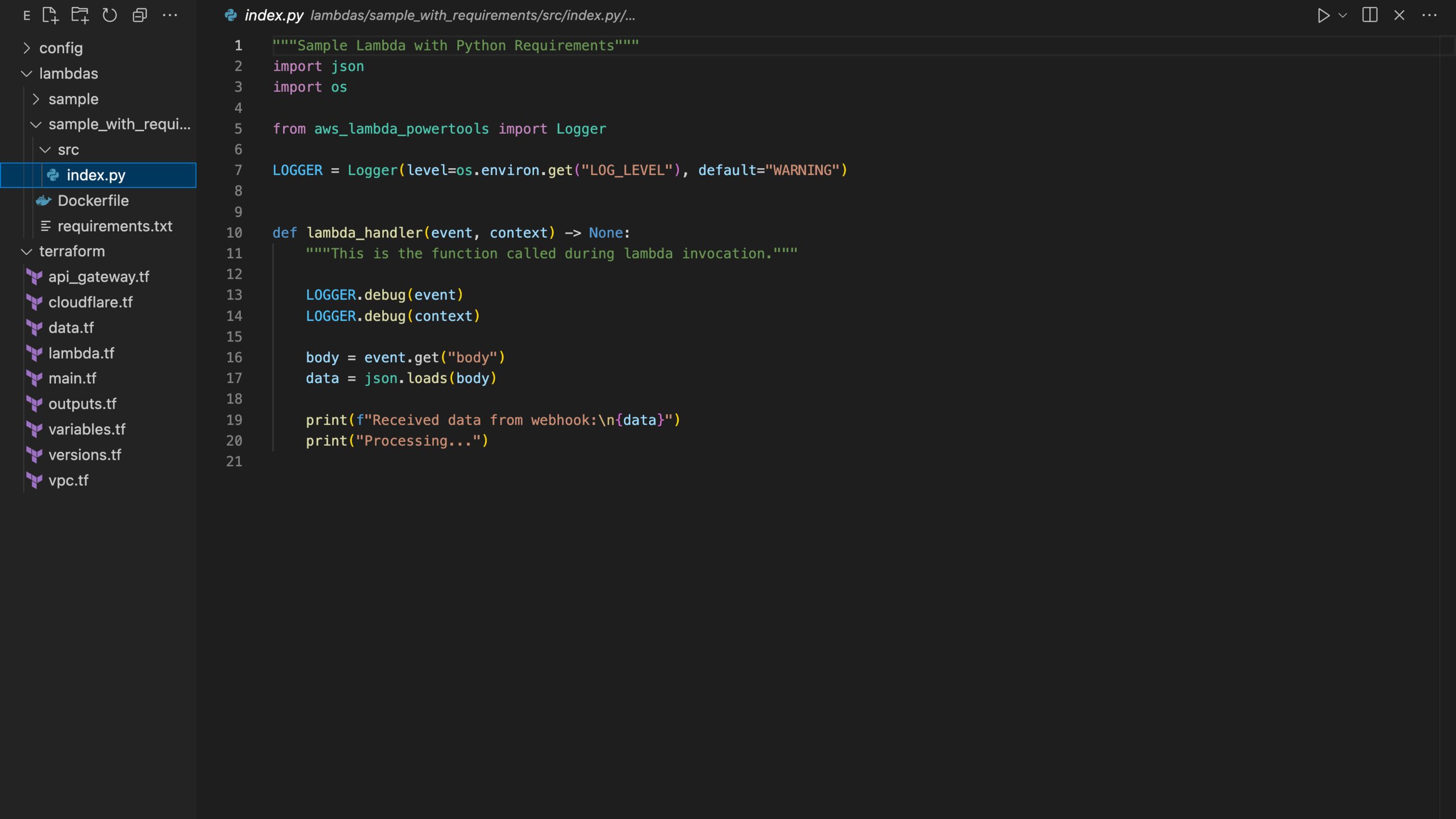

Dockerfile lambdas/sample_with_requirements/Dockerfile/ FROM
config
lambdas
sample
sample_with_requi...
src
index.py
Dockerfile
requirements.txt
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf
1 FROM public.ecr.aws/lambda/python:3.10
2
3 COPY requirements.txt .
4 COPY src .
5
6 RUN pip install -r requirements.txt
7
8 CMD ["index.lambda_handler"]
9

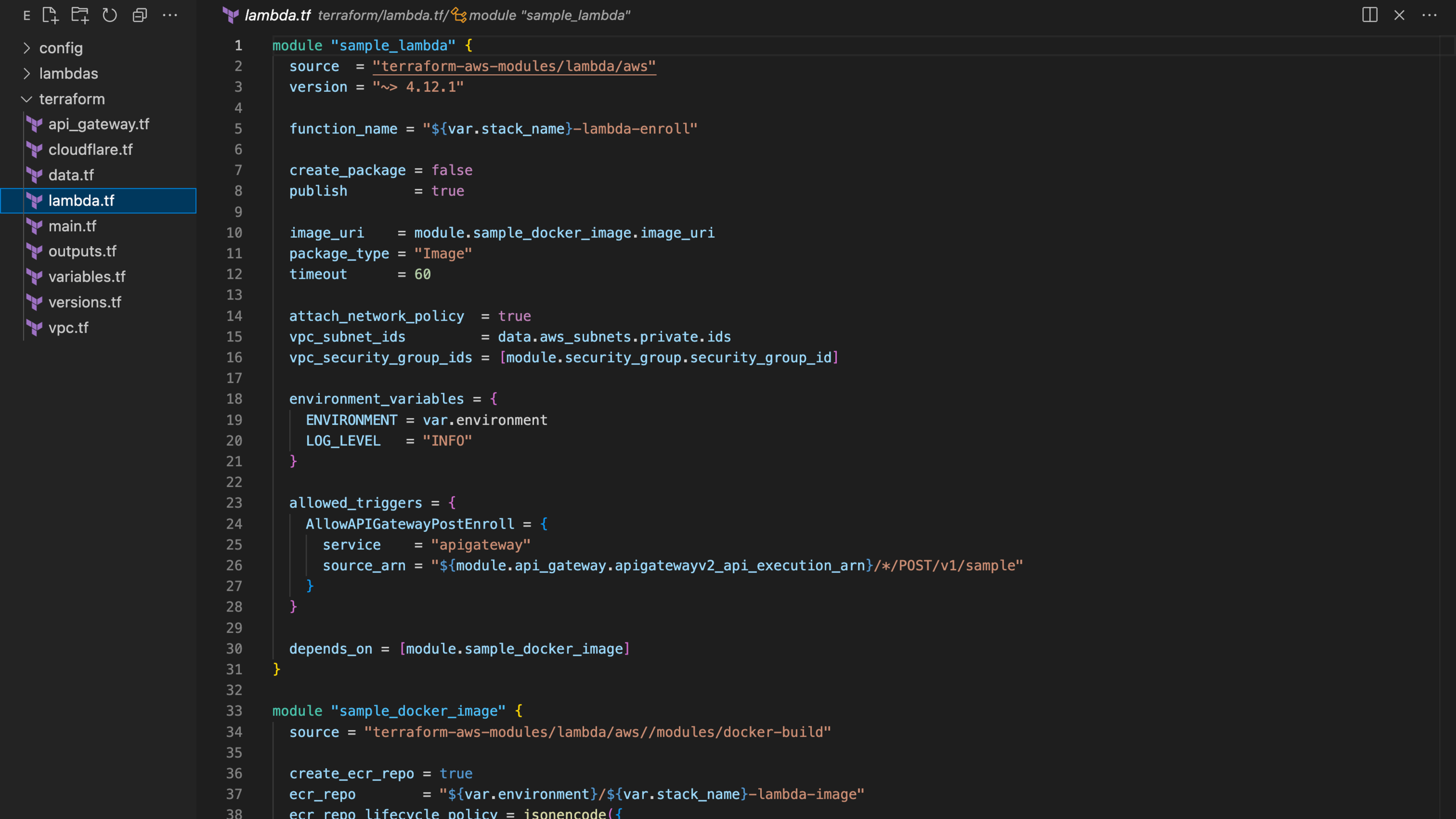

Dockerfile lambdas/sample_with_requirements/Dockerfile/ FROM
config
lambdas
sample
sample_with_requi...
src
index.py
Dockerfile
requirements.txt
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf
1 FROM public.ecr.aws/lambda/python:3.10
2
3 COPY requirements.txt .
4 COPY src .
5
6 RUN pip install -r requirements.txt
7
8 CMD ["index.lambda_handler"]
9

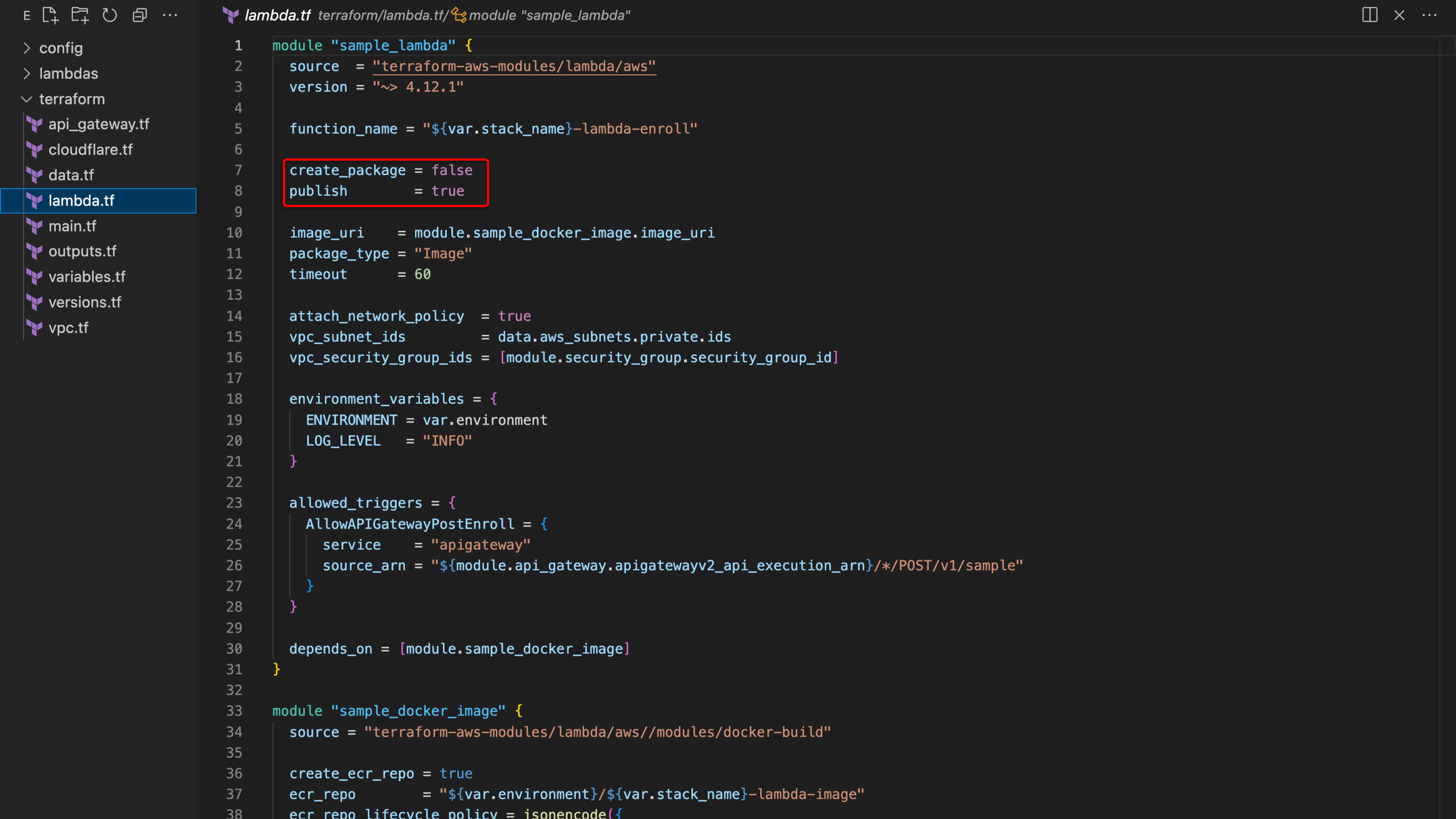

Dockerfile lambdas/sample_with_requirements/Dockerfile/ FROM
config
lambdas
sample
sample_with_requi...
src
index.py
Dockerfile
requirements.txt
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf
1 FROM public.ecr.aws/lambda/python:3.10
2
3 COPY requirements.txt .
4 COPY src .
5
6 RUN pip install -r requirements.txt
7
8 CMD ["index.lambda_handler"]
9

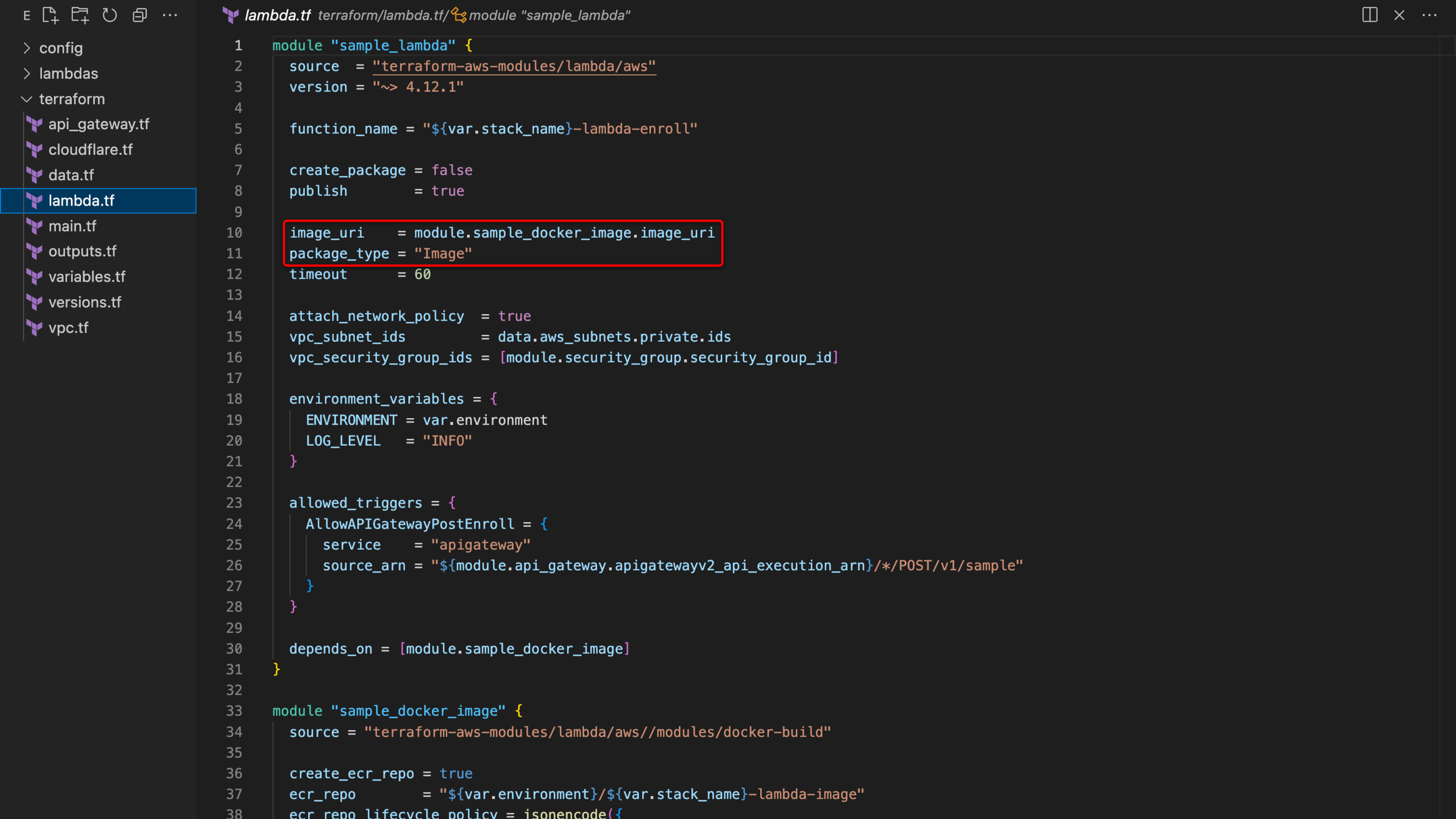

Dockerfile lambdas/sample_with_requirements/Dockerfile/ FROM
config
lambdas
sample
sample_with_requi...
src
index.py
Dockerfile
requirements.txt
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf
1 FROM public.ecr.aws/lambda/python:3.10
2
3 COPY requirements.txt .
4 COPY src .
5
6 RUN pip install -r requirements.txt
7
8 CMD ["index.lambda_handler"]
9

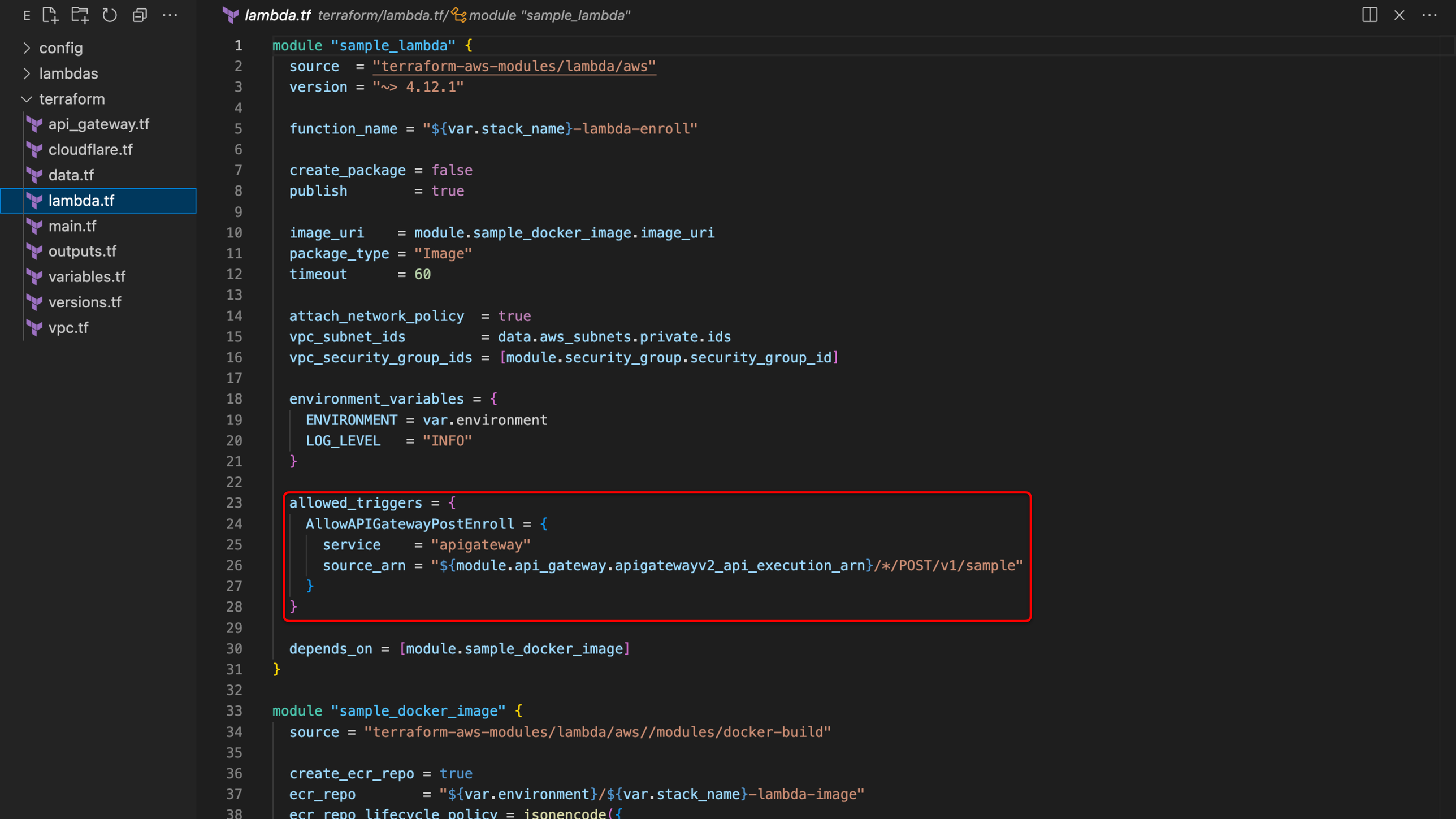

requirements.txt lambdas/sample_with_requirements/requirements.txt
1 aws_lambda_powertools == 2.19.0
2
config
lambdas
sample
sample_with_requi...
src
index.py
Dockerfile
requirements.txt
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf

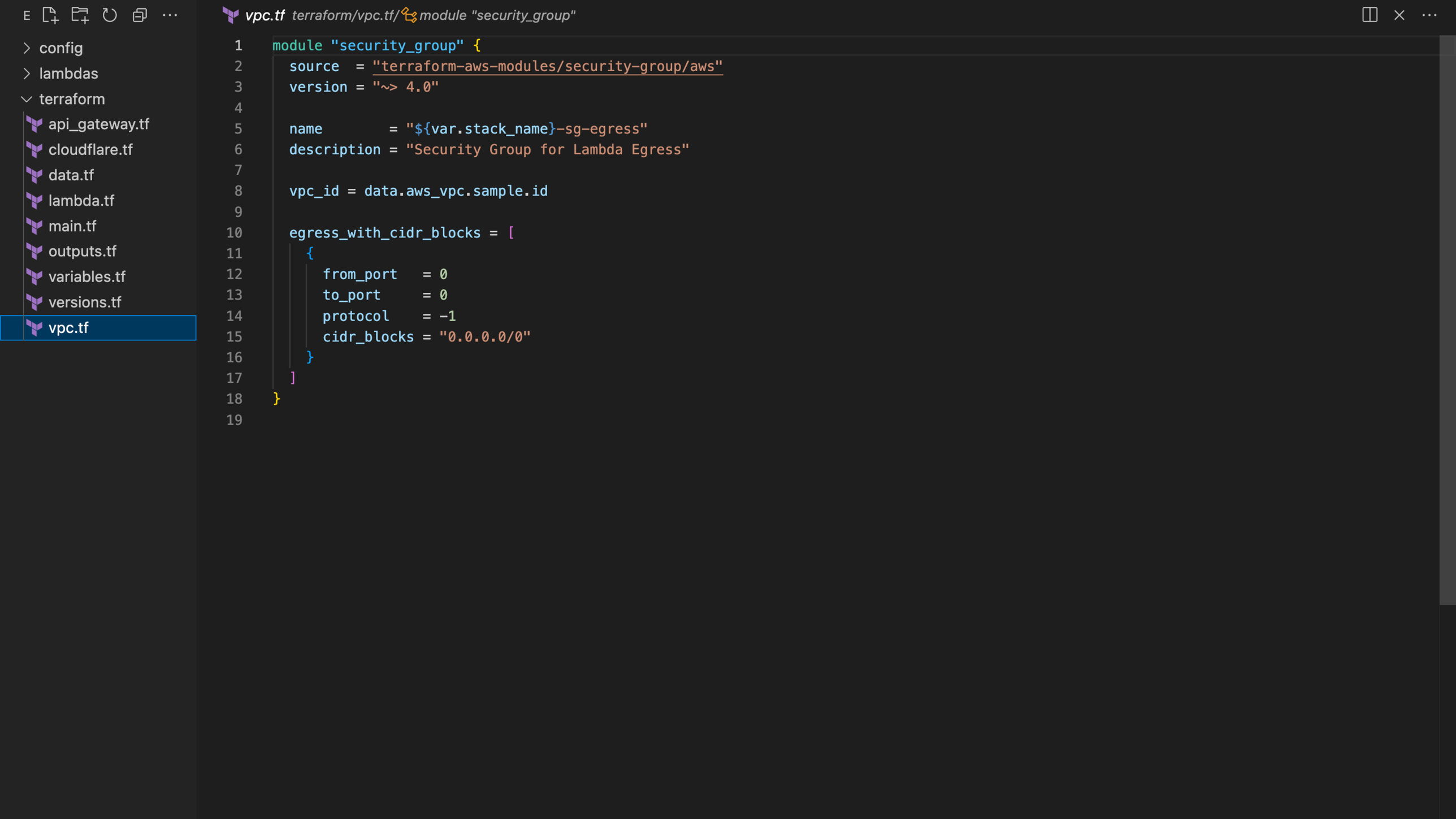

index.py lambdas/sample_with_requirements/src/index.py/...
config
lambdas
sample
sample_with_requi...
src
index.py
Dockerfile
requirements.txt
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf
"""Sample Lambda with Python Requirements"""
import json
import os
from aws_lambda_powertools import Logger
LOGGER = Logger(level=os.environ.get("LOG_LEVEL"), default="WARNING")
def lambda_handler(event, context) -> None:
"""This is the function called during lambda invocation."""
LOGGER.debug(event)
LOGGER.debug(context)
body = event.get("body")
data = json.loads(body)
print(f"Received data from webhook:\n{data}")
print("Processing...")

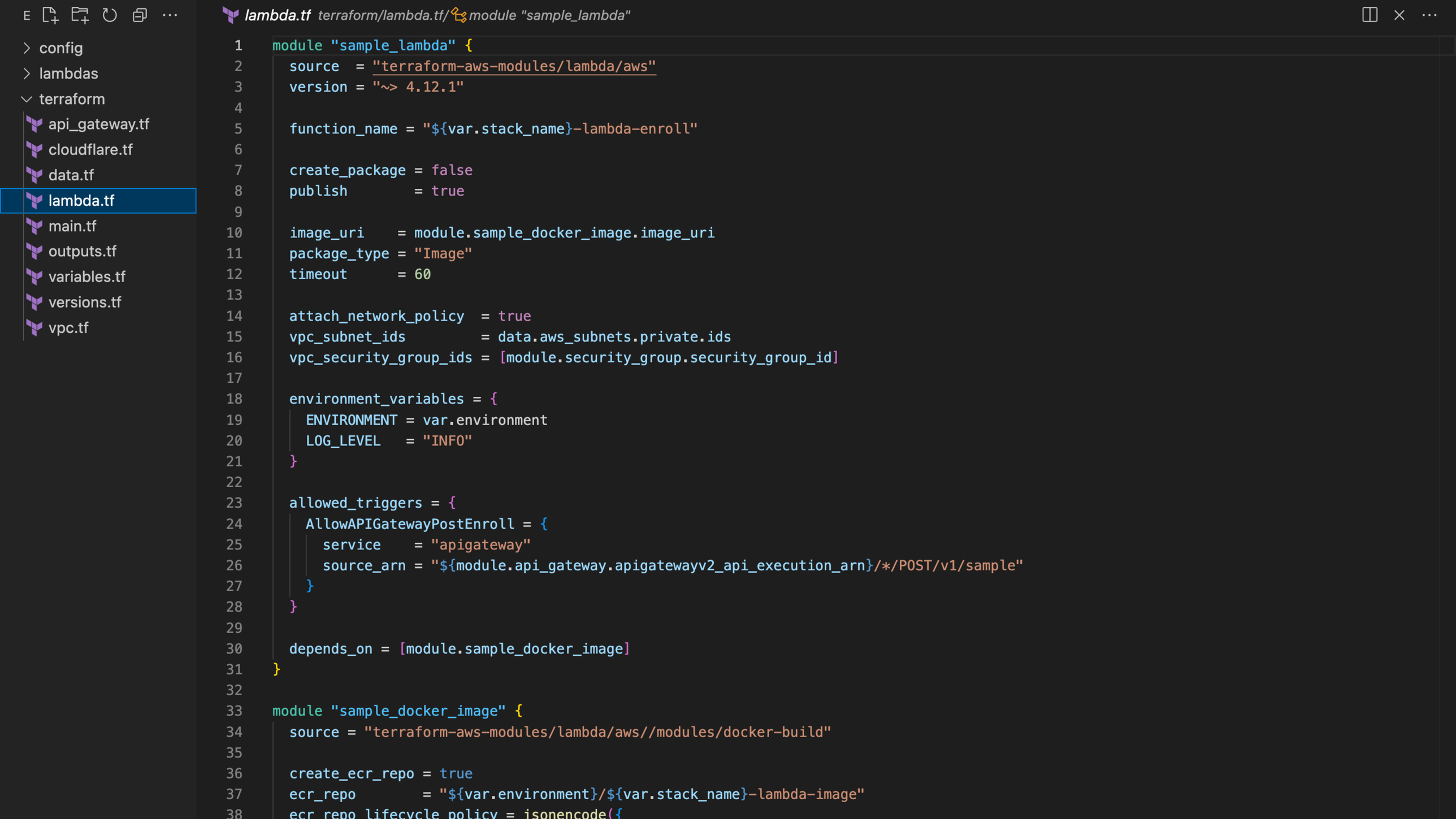

lambda.tf terraform/lambda.tf/ module "sample_lambda"
config
lambdas
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf
module "sample_lambda" {
source = "terraform-aws-modules/lambda/aws"
version = "~> 4.12.1"
function_name = "${var.stack_name}-lambda-enroll"
create_package = false
publish = true
image_uri = module.sample_docker_image.image_uri
package_type = "Image"
timeout = 60
attach_network_policy = true
vpc_subnet_ids = data.aws_subnets.private.ids
vpc_security_group_ids = [module.security_group.security_group_id]
environment_variables = {
ENVIRONMENT = var.environment
LOG_LEVEL = "INFO"
}
allowed_triggers = {
AllowAPIGatewayPostEnroll = {
service = "apigateway"
source_arn = "${module.api_gateway.apigatewayv2_api_execution_arn}/*/POST/v1/sample"
}
}
depends_on = [module.sample_docker_image]
}
module "sample_docker_image" {
source = "terraform-aws-modules/lambda/aws//modules/docker-build"
create_ecr_repo = true
ecr_repo = "${var.environment}/${var.stack_name}-lambda-image"
ecr_repo_lifecycle_policy = jsonencode({

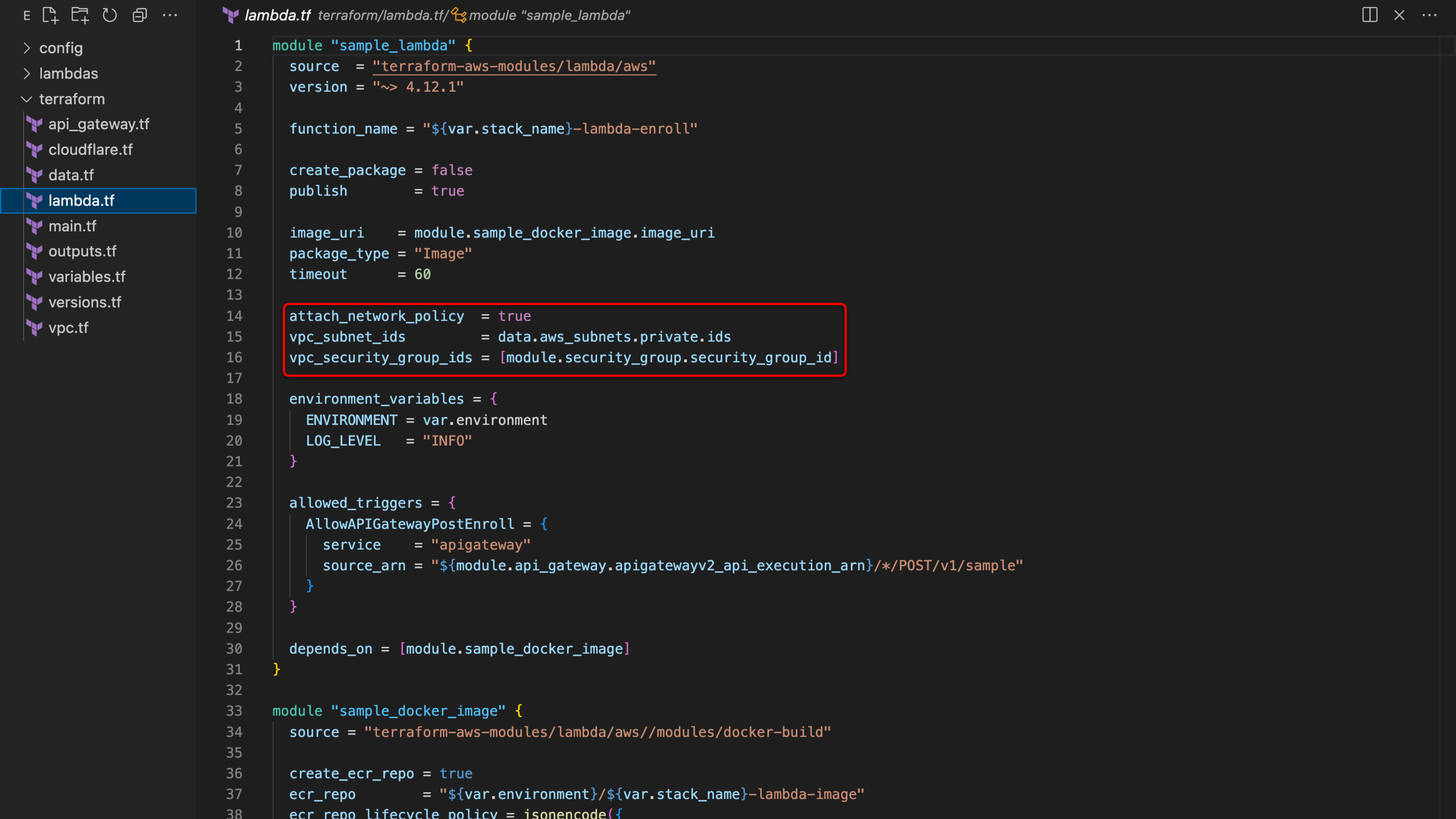

config
lambdas
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf
lambda.tf terraform/lambda.tf/ module "sample_lambda"
module "sample_lambda" {
source = "terraform-aws-modules/lambda/aws"
version = "~> 4.12.1"
function_name = "${var.stack_name}-lambda-enroll"
create_package = false
publish = true
image_uri = module.sample_docker_image.image_uri
package_type = "Image"
timeout = 60
attach_network_policy = true
vpc_subnet_ids = data.aws_subnets.private.ids
vpc_security_group_ids = [module.security_group.security_group_id]
environment_variables = {
ENVIRONMENT = var.environment
LOG_LEVEL = "INFO"
}
allowed_triggers = {
AllowAPIGatewayPostEnroll = {
service = "apigateway"
source_arn = "${module.api_gateway.apigatewayv2_api_execution_arn}/*/POST/v1/sample"
}
}
depends_on = [module.sample_docker_image]
}
module "sample_docker_image" {
source = "terraform-aws-modules/lambda/aws//modules/docker-build"
create_ecr_repo = true
ecr_repo = "${var.environment}/${var.stack_name}-lambda-image"
ecr_repo_lifecycle_policy = jsonencode({

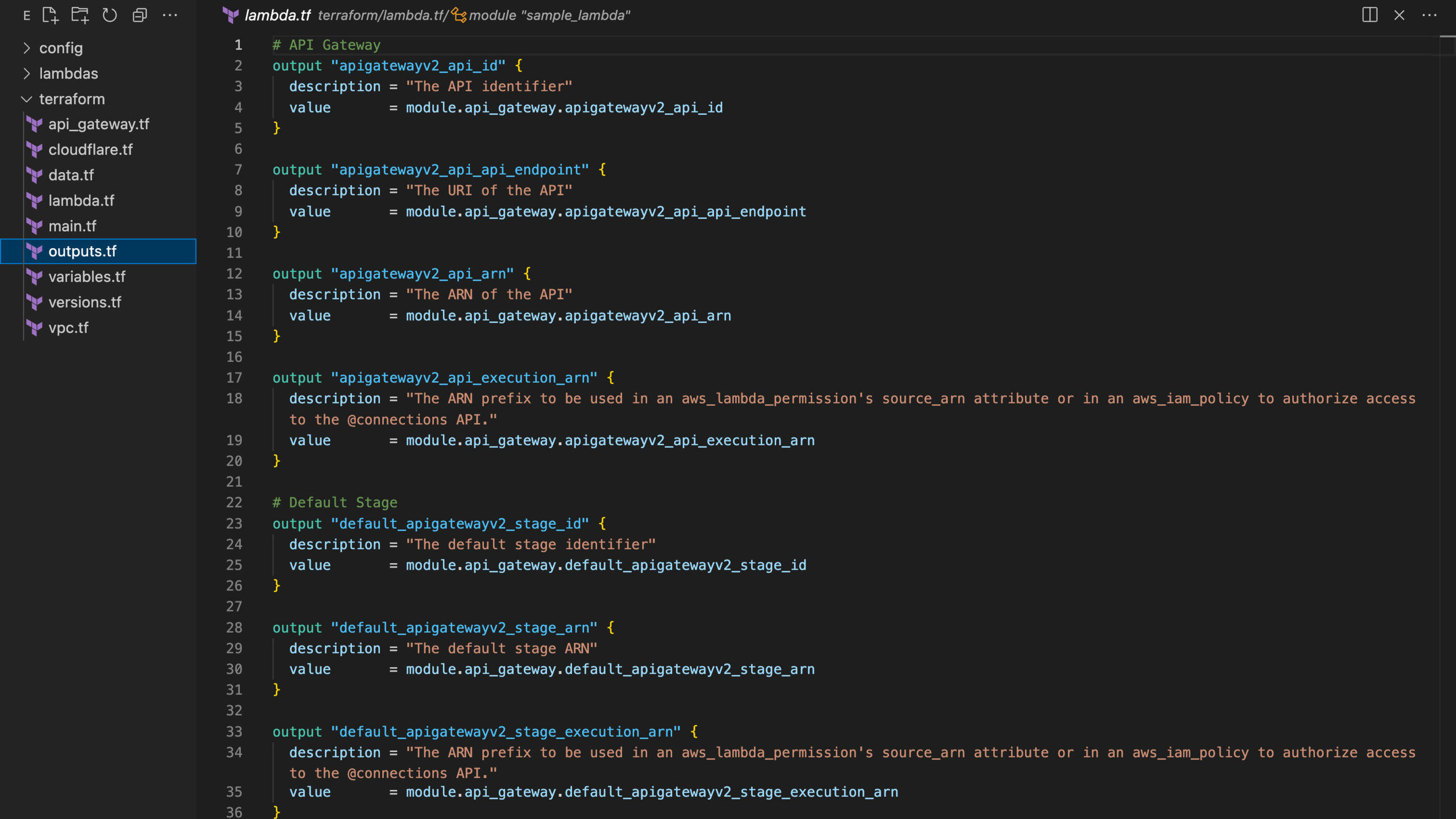

lambda.tf terraform/lambda.tf/ module "sample_lambda"
config
lambdas
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf
module "sample_lambda" {
source = "terraform-aws-modules/lambda/aws"
version = "~> 4.12.1"
function_name = "${var.stack_name}-lambda-enroll"
create_package = false
publish = true
image_uri = module.sample_docker_image.image_uri
package_type = "Image"
timeout = 60
attach_network_policy = true
vpc_subnet_ids = data.aws_subnets.private.ids
vpc_security_group_ids = [module.security_group.security_group_id]
environment_variables = {
ENVIRONMENT = var.environment
LOG_LEVEL = "INFO"
}
allowed_triggers = {
AllowAPIGatewayPostEnroll = {
service = "apigateway"
source_arn = "${module.api_gateway.apigatewayv2_api_execution_arn}/*/POST/v1/sample"
}
}
depends_on = [module.sample_docker_image]
}
module "sample_docker_image" {
source = "terraform-aws-modules/lambda/aws//modules/docker-build"
create_ecr_repo = true
ecr_repo = "${var.environment}/${var.stack_name}-lambda-image"
ecr_repo_lifecycle_policy = jsonencode({

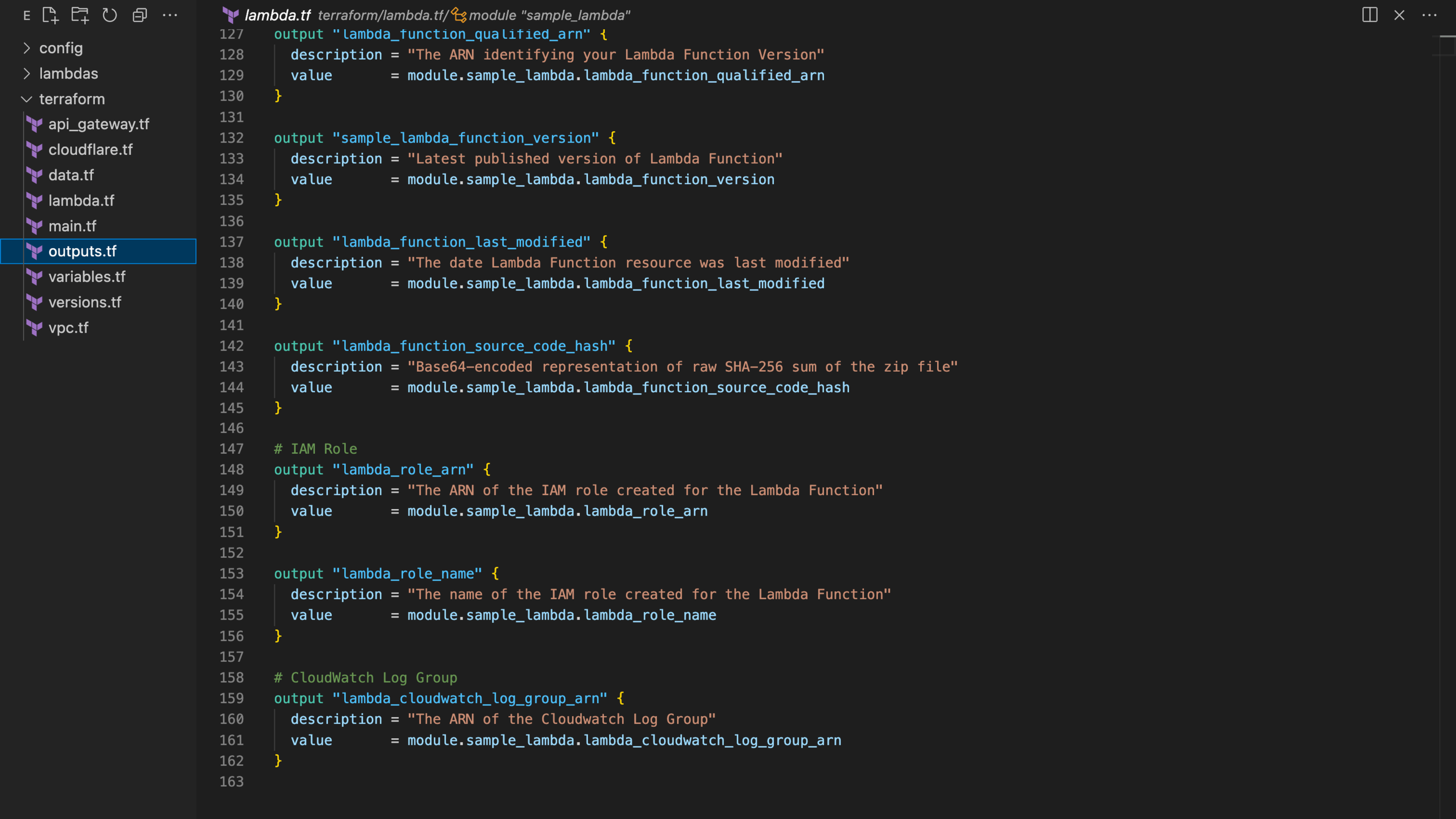

lambda.tf terraform/lambda.tf/ module "sample_lambda"
config
lambdas
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf
module "sample_lambda" {
source = "terraform-aws-modules/lambda/aws"
version = "~> 4.12.1"
function_name = "${var.stack_name}-lambda-enroll"
create_package = false
publish = true
image_uri = module.sample_docker_image.image_uri
package_type = "Image"
timeout = 60
attach_network_policy = true
vpc_subnet_ids = data.aws_subnets.private.ids
vpc_security_group_ids = [module.security_group.security_group_id]
environment_variables = {
ENVIRONMENT = var.environment
LOG_LEVEL = "INFO"
}
allowed_triggers = {
AllowAPIGatewayPostEnroll = {
service = "apigateway"
source_arn = "${module.api_gateway.apigatewayv2_api_execution_arn}/*/POST/v1/sample"
}
}
depends_on = [module.sample_docker_image]
}
module "sample_docker_image" {
source = "terraform-aws-modules/lambda/aws//modules/docker-build"
create_ecr_repo = true
ecr_repo = "${var.environment}/${var.stack_name}-lambda-image"
ecr_repo_lifecycle_policy = jsonencode({


vpc.tf terraform/vpc.tf/ module "security_group"
config
lambdas
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf
module "security_group" {
source = "terraform-aws-modules/security-group/aws"
version = "~> 4.0"
name = "${var.stack_name}-sg-egress"
description = "Security Group for Lambda Egress"
vpc_id = data.aws_vpc.sample.id
egress_with_cidr_blocks = [
{
from_port = 0
to_port = 0
protocol = -1
cidr_blocks = "0.0.0.0/0"
}
]
}


lambda.tf terraform/lambda.tf/ module "sample_lambda"
config
lambdas
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf
module "sample_lambda" {
source = "terraform-aws-modules/lambda/aws"
version = "~> 4.12.1"
function_name = "${var.stack_name}-lambda-enroll"
create_package = false
publish = true
image_uri = module.sample_docker_image.image_uri
package_type = "Image"
timeout = 60
attach_network_policy = true
vpc_subnet_ids = data.aws_subnets.private.ids
vpc_security_group_ids = [module.security_group.security_group_id]
environment_variables = {
ENVIRONMENT = var.environment
LOG_LEVEL = "INFO"
}
allowed_triggers = {
AllowAPIGatewayPostEnroll = {
service = "apigateway"
source_arn = "${module.api_gateway.apigatewayv2_api_execution_arn}/*/POST/v1/sample"
}
}
depends_on = [module.sample_docker_image]
}
module "sample_docker_image" {
source = "terraform-aws-modules/lambda/aws//modules/docker-build"
create_ecr_repo = true
ecr_repo = "${var.environment}/${var.stack_name}-lambda-image"
ecr_repo_lifecycle_policy = jsonencode({


lambda.tf terraform/lambda.tf/ module "sample_lambda"
config
lambdas
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf
module "sample_lambda" {
source = "terraform-aws-modules/lambda/aws"
version = "~> 4.12.1"
function_name = "${var.stack_name}-lambda-enroll"
create_package = false
publish = true
image_uri = module.sample_docker_image.image_uri
package_type = "Image"
timeout = 60
attach_network_policy = true
vpc_subnet_ids = data.aws_subnets.private.ids
vpc_security_group_ids = [module.security_group.security_group_id]
environment_variables = {
ENVIRONMENT = var.environment
LOG_LEVEL = "INFO"
}
allowed_triggers = {
AllowAPIGatewayPostEnroll = {
service = "apigateway"
source_arn = "${module.api_gateway.apigatewayv2_api_execution_arn}/*/POST/v1/sample"
}
}
depends_on = [module.sample_docker_image]
}
module "sample_docker_image" {
source = "terraform-aws-modules/lambda/aws//modules/docker-build"
create_ecr_repo = true
ecr_repo = "${var.environment}/${var.stack_name}-lambda-image"
ecr_repo_lifecycle_policy = jsonencode({


lambda.tf terraform/lambda.tf/ module "sample_lambda"
config
lambdas
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf
# API Gateway
output "apigatewayv2_api_id" {
description = "The API identifier"
value = module.api_gateway.apigatewayv2_api_id
}
output "apigatewayv2_api_api_endpoint" {
description = "The URI of the API"
value = module.api_gateway.apigatewayv2_api_api_endpoint
}
output "apigatewayv2_api_arn" {
description = "The ARN of the API"
value = module.api_gateway.apigatewayv2_api_arn
}
output "apigatewayv2_api_execution_arn" {
description = "The ARN prefix to be used in an aws_lambda_permission's source_arn attribute or in an aws_iam_policy to authorize access to the @connections API."
value = module.api_gateway.apigatewayv2_api_execution_arn
}
# Default Stage
output "default_apigatewayv2_stage_id" {
description = "The default stage identifier"
value = module.api_gateway.default_apigatewayv2_stage_id
}
output "default_apigatewayv2_stage_arn" {
description = "The default stage ARN"
value = module.api_gateway.default_apigatewayv2_stage_arn
}
output "default_apigatewayv2_stage_execution_arn" {
description = "The ARN prefix to be used in an aws_lambda_permission's source_arn attribute or in an aws_iam_policy to authorize access to the @connections API."
value = module.api_gateway.default_apigatewayv2_stage_execution_arn
}


config
lambdas
terraform
api_gateway.tf
cloudflare.tf
data.tf
lambda.tf
main.tf
outputs.tf
variables.tf
versions.tf
vpc.tf
lambda.tf terraform/lambda.tf/ module "sample_lambda"
output "lambda_function_qualified_arn" {
description = "The ARN identifying your Lambda Function Version"
value = module.sample_lambda.lambda_function_qualified_arn
}
output "sample_lambda_function_version" {
description = "Latest published version of Lambda Function"
value = module.sample_lambda.lambda_function_version
}
output "lambda_function_last_modified" {
description = "The date Lambda Function resource was last modified"
value = module.sample_lambda.lambda_function_last_modified
}
output "lambda_function_source_code_hash" {
description = "Base64-encoded representation of raw SHA-256 sum of the zip file"
value = module.sample_lambda.lambda_function_source_code_hash
}
# IAM Role
output "lambda_role_arn" {
description = "The ARN of the IAM role created for the Lambda Function"
value = module.sample_lambda.lambda_role_arn
}
output "lambda_role_name" {
description = "The name of the IAM role created for the Lambda Function"
value = module.sample_lambda.lambda_role_name
}
# CloudWatch Log Group
output "lambda_cloudwatch_log_group_arn" {
description = "The ARN of the Cloudwatch Log Group"
value = module.sample_lambda.lambda_cloudwatch_log_group_arn
}

Questions?

https://github.com/MScottBlake/Presentations

https://bit.ly/psumac2023-111

https://bit.ly/psumac2023-111