

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?

Incall webhook

Incall webhook will deliver real-time information of the call to other application during calls

Method *	Target URL *
We recommend not putting any query string argument in URL	
Get	https://

How are web hooks configured?

<https://env7xuvnzsrlr.x.pipedream.net>

We will submit a POST request to this URL with a JSON object representing the event data for any events you choose to be notified of.

Select the types of events you wish to be notified of

- Email/Push Sent**
- Email Delivered
- Email/Push Opened
- Email Clicked
- Email Bounced
- User Unsubscribed
- User Created
- User Updated
- User Converted

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.bucketRegionalDomainName`

`aws_s3_bucket.example.hostedZoneId`

`aws_s3_bucket.example.region`

`aws_s3_bucket.example.tagsAll`

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.bucketRegionalDomainName`

`data.aws_s3_bucket.example.hostedZoneId`

`data.aws_s3_bucket.example.region`

`data.aws_s3_bucket.example.websiteEndpoint`

`data.aws_s3_bucket.example.websiteDomain`

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_domain_name
module.example_bucket.s3_bucketRegionalDomainName
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"
```

```
    < config  
    < dev  
        < backend.tfvars  
        < terraform.tfvars  
        > prod  
        > test  
        < terraform.tfvars  
        > lambdas  
        > terraform
```

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"
```

```
✓ config  
✓ dev  
  ✓ backend.tfvars  
  ✓ terraform.tfvars  
  > prod  
  > test  
  ✓ terraform.tfvars  
  > lambdas  
  > terraform
```

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"
```

```
✓ config  
  ✓ dev  
    ➜ backend.tfvars  
    ➜ terraform.tfvars  
  > prod  
  > test  
    ➜ terraform.tfvars  
  > lambdas  
  > terraform
```

terraform apply

terraform apply

terraform apply
plan.out

terraform apply

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

terraform destroy

terraform destroy

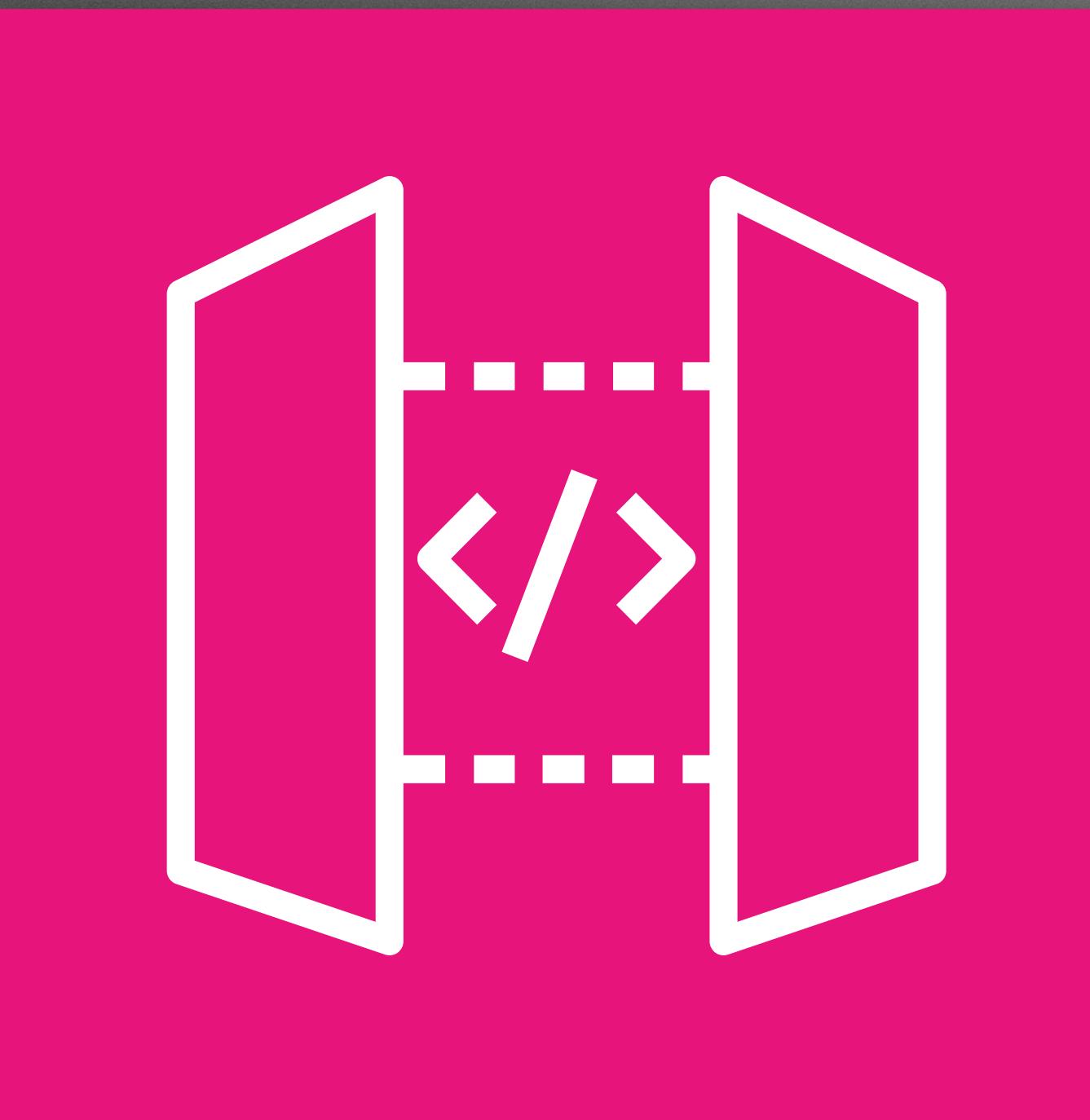
terraform destroy

terraform apply -destroy

terraform plan -destroy

Amazon Web Services (AWS)

AWS

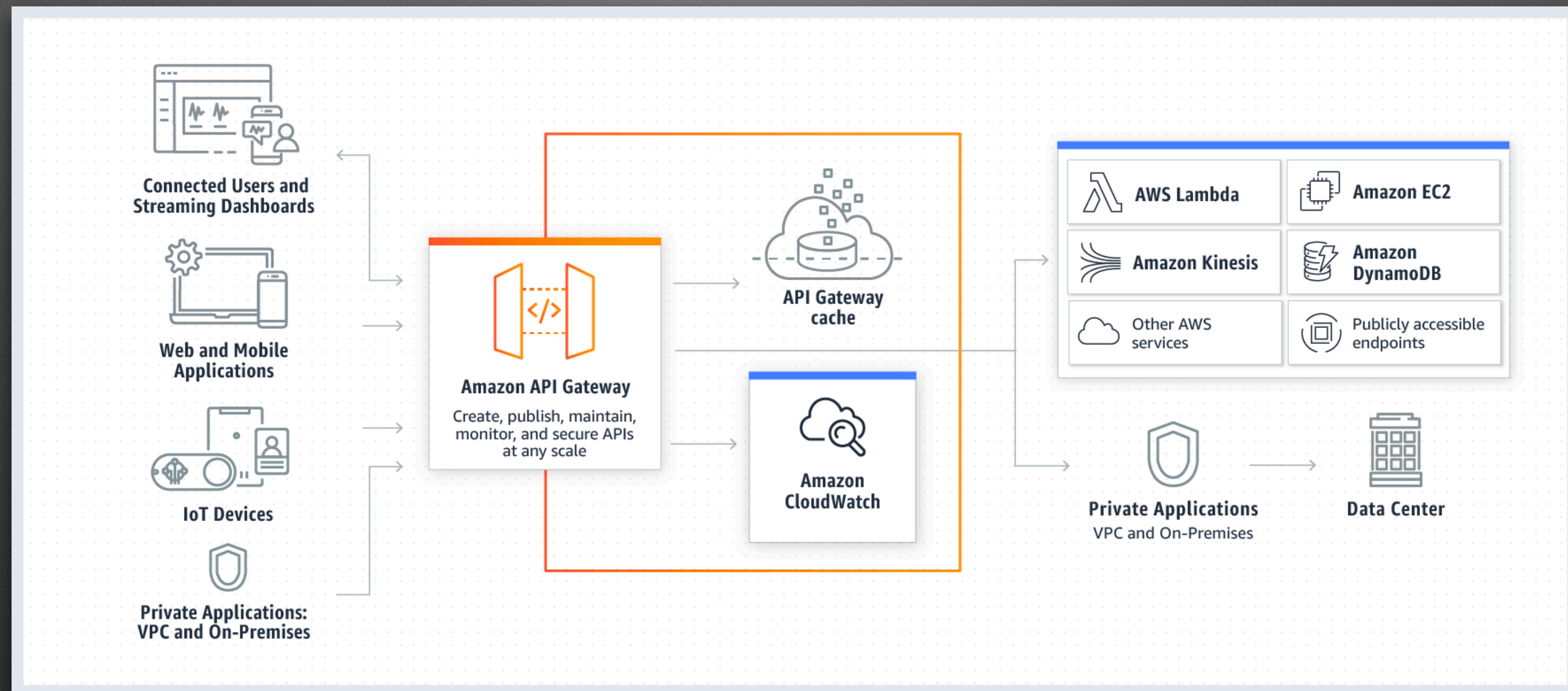


API Gateway



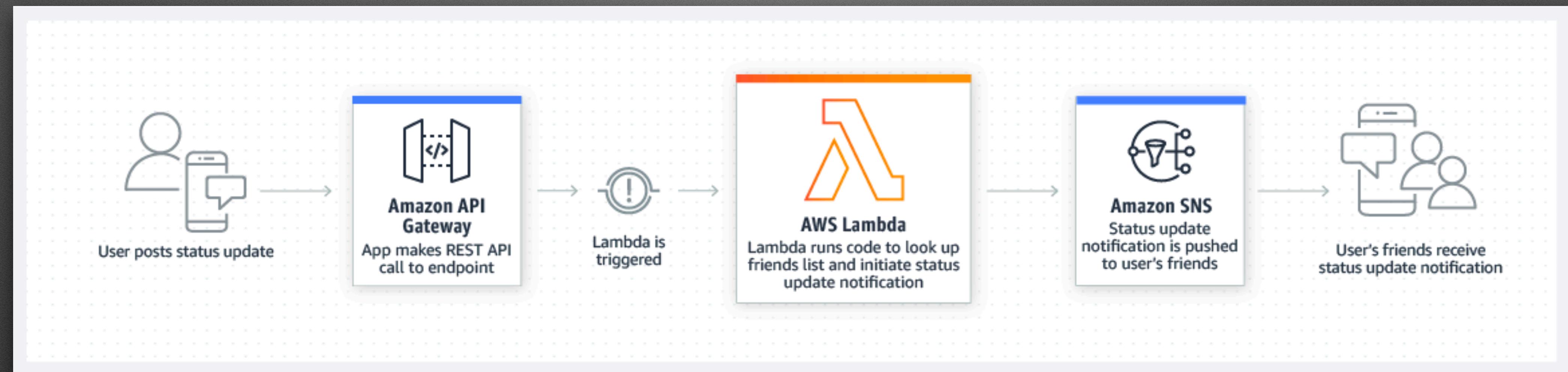
Lambda

API Gateway



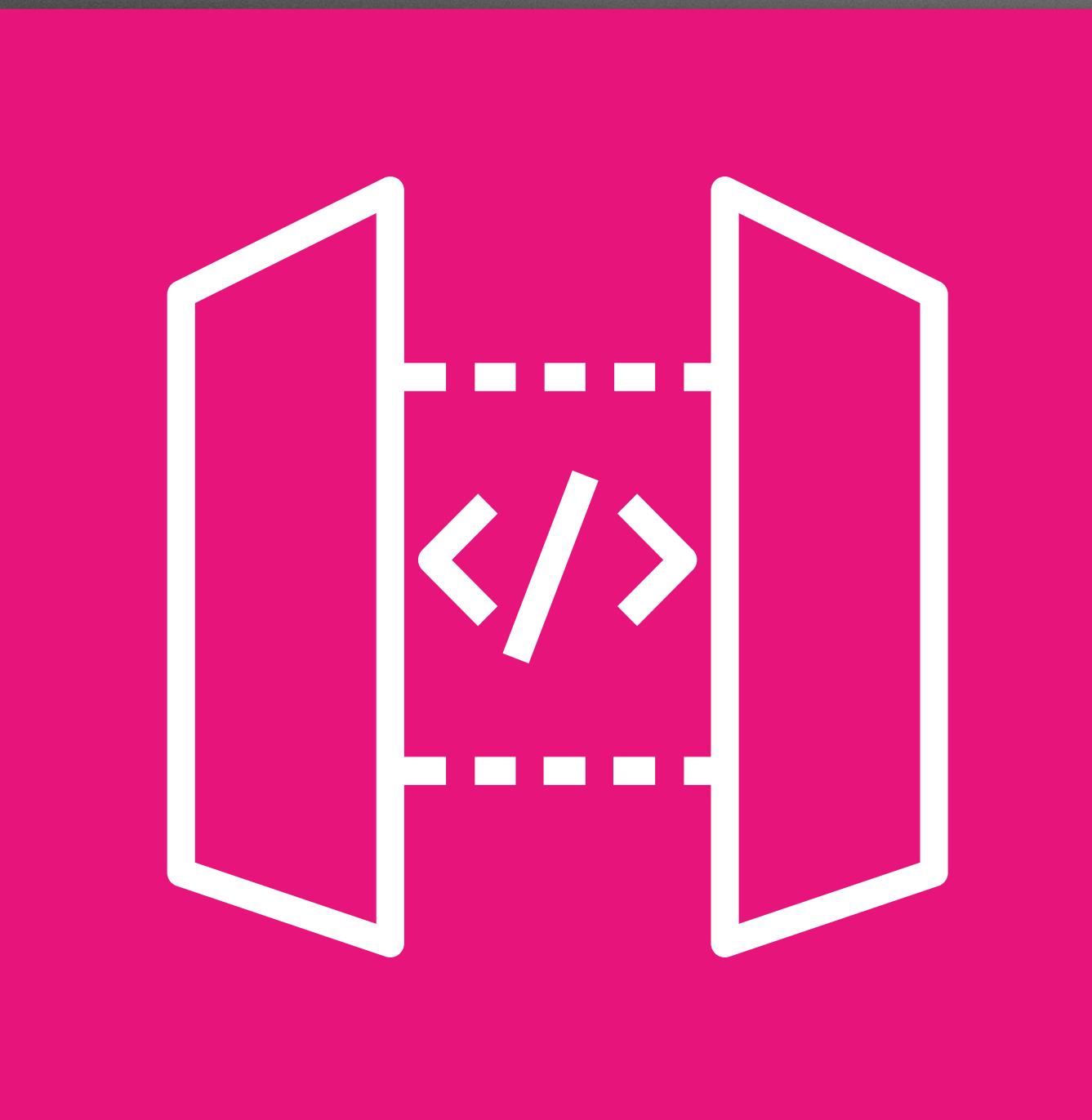
<https://aws.amazon.com/api-gateway>

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/apigateway-v2/aws/latest)

[https://registry.terraform.io/modules/terraform-aws-modules/lambda/aws/
latest](https://registry.terraform.io/modules/terraform-aws-modules/lambda/aws/latest)

Putting it all Together



```
1 terraform {
2     backend "s3" {}
3
4     required_providers {
5         aws = {
6             source  = "hashicorp/aws"
7             version = "~> 4.22"
8         }
9         cloudflare = {
10            source  = "cloudflare/cloudflare"
11            version = "~> 3.9.1"
12        }
13        docker = {
14            source  = "kreuzwerker/docker"
15            version = "~> 3.0"
16        }
17    }
18
19    required_version = ">= 1.2"
20 }
21
```

The screenshot shows a code editor interface with a dark theme. On the left is a sidebar containing a file tree. The file tree includes:

- > config
- > lambdas
- terraform
 - api_gateway.tf
 - cloudflare.tf
 - data.tf
 - lambda.tf
 - main.tf
 - outputs.tf
 - variables.tf
 - versions.tf** (highlighted with a blue background)
 - vpc.tf

The main editor area displays the contents of the `versions.tf` file. The code is as follows:

```
1  terraform {  
2      backend "s3" {}  
3  
4      required_providers {  
5          aws = {  
6              source  = "hashicorp/aws"  
7              version = "~> 4.22"  
8          }  
9          cloudflare = {  
10             source  = "cloudflare/cloudflare"  
11             version = "~> 3.9.1"  
12         }  
13         docker = {  
14             source  = "kreuzwerker/docker"  
15             version = "~> 3.0"  
16         }  
17     }  
18  
19     required_version = ">= 1.2"  
20 }  
21
```

The line `required_providers` is highlighted with a red rounded rectangle.

The screenshot shows a code editor interface with a dark theme. On the left is a sidebar displaying a file tree:

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

The main editor area contains the following Terraform code:

```
1  terraform {  
2      backend "s3" {}  
3  
4      required_providers {  
5          aws = {  
6              source  = "hashicorp/aws"  
7              version = "~> 4.22"  
8          }  
9          cloudflare = {  
10             source  = "cloudflare/cloudflare"  
11             version = "~> 3.9.1"  
12         }  
13         docker = {  
14             source  = "kreuzwerker/docker"  
15             version = "~> 3.0"  
16         }  
17     }  
18  
19     required_version = ">= 1.2"  
20 }  
21
```

The line `required_version = ">= 1.2"` is highlighted with a red rectangular border.

E ⌘ F ⌘ O ⌘ ⌘ ...

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

```
1 provider "aws" {  
2   region = var.region  
3  
4   default_tags {  
5     tags = {  
6       environment = var.environment  
7       owner      = "IT"  
8       application = "Sample Webhook Handler"  
9     }  
10  }  
11}  
12  
13 provider "cloudflare" {}  
14  
15 provider "docker" {  
16   registry_auth {  
17     address  = "${data.aws_caller_identity.this.account_id}.dkr.ecr.${data.aws_region.current.name}.amazonaws.com"  
18     username = data.aws_ecr_authorization_token.token.user_name  
19     password = data.aws_ecr_authorization_token.token.password  
20   }  
21 }  
22
```

E ⌘ F ⌘ O ⌘ ⌘ ...

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

```
1 provider "aws" {
2   region = var.region
3
4   default_tags {
5     tags = {
6       environment = var.environment
7       owner       = "IT"
8       application = "Sample Webhook Handler"
9     }
10  }
11 }
12
13 provider "cloudflare" {}
14
15 provider "docker" {
16   registry_auth {
17     address  = "${data.aws_caller_identity.this.account_id}.dkr.ecr.${data.aws_region.current.name}.amazonaws.com"
18     username = data.aws_ecr_authorization_token.token.user_name
19     password = data.aws_ecr_authorization_token.token.password
20   }
21 }
22
```



> config
> lambdas
∨ terraform
 └ api_gateway.tf
 └ cloudflare.tf
 └ data.tf
 └ lambda.tf
 └ main.tf
 └ outputs.tf
 └ variables.tf **variables.tf**
 └ versions.tf
 └ vpc.tf

```
1  variable "aws_account_alias" {  
2    description = "AWS account alias"  
3    type        = string  
4  }  
5  
6  variable "domain" {  
7    description = "The base domain to be used by the API Gateway"  
8    type        = string  
9  }  
10  
11 variable "environment" {  
12   default = "dev"  
13   type    = string  
14 }  
15  
16 variable "region" {  
17   default = "us-west-1"  
18   type    = string  
19 }  
20  
21 variable "stack_name" {  
22   description = "Name of the Lambda Stack"  
23   type        = string  
24   default     = "sample-webhook-handler"  
25 }  
26  
27 variable "subdomain" {  
28   type    = string  
29   default = "sample-webhook-handler"  
30 }  
31
```

The screenshot shows a code editor interface with a dark theme. On the left is a sidebar containing a file tree. The tree includes a 'config' folder, a 'lambdas' folder, a 'terraform' folder which contains 'api_gateway.tf', 'cloudflare.tf', 'data.tf' (which is selected and highlighted with a blue background), 'lambda.tf', 'main.tf', 'outputs.tf', 'variables.tf', 'versions.tf', and 'vpc.tf'. The main editor area displays a Terraform configuration file named 'data.tf'. The code uses AWS provider syntax to define data sources for regions, caller identities, ECR authorization tokens, VPCs, subnets, and Cloudflare zones.

```
1  data "aws_region" "current" {}
2
3  data "aws_caller_identity" "this" {}
4
5  data "aws_ecr_authorization_token" "token" {}
6
7  data "aws_vpc" "sample" {
8    filter {
9      name   = "tag:Name"
10     values = [var.aws_account_alias]
11   }
12 }
13
14 data "aws_subnets" "private" {
15   filter {
16     name   = "vpc-id"
17     values = [data.aws_vpc.sample.id]
18   }
19 }
20 tags = {
21   Name = "${var.aws_account_alias}-private-*"
22 }
23 }
24
25 data "cloudflare_zone" "this" {
26   name = var.domain
27 }
28
```

A screenshot of a code editor displaying a Terraform configuration file named `data.tf`. The file is located in a directory structure under `terraform`, which also contains `api_gateway.tf`, `cloudflare.tf`, and `vpc.tf`. The `data.tf` file is highlighted with a blue background.

```
1  data "aws_region" "current" {}
2
3  data "aws_caller_identity" "this" {}
4
5  data "aws_ecr_authorization_token" "token" {}
6
7  data "aws_vpc" "sample" {
8    filter {
9      name   = "tag:Name"
10     values = [var.aws_account_alias]
11   }
12 }
13
14 data "aws_subnets" "private" {
15   filter {
16     name   = "vpc-id"
17     values = [data.aws_vpc.sample.id]
18   }
19 }
20 tags = {
21   Name = "${var.aws_account_alias}-private-*"
22 }
23 }
24
25 data "cloudflare_zone" "this" {
26   name = var.domain
27 }
28
```

The code defines several data sources and their filters. It includes `aws_region`, `aws_caller_identity`, `aws_ecr_authorization_token`, `aws_vpc` (with a specific sample configuration), `aws_subnets` (filtering by `vpc-id`), and `cloudflare_zone` (filtering by `name` and setting it to `var.domain`). A red box highlights the `cloudflare_zone` data source block.

The screenshot shows a code editor interface with a dark theme. On the left is a sidebar containing a file tree:

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

The main editor area displays Terraform configuration code for Cloudflare resources. The code includes:

- A Cloudflare CNAME record resource named "sample_subdomain".
- A Cloudflare validation record resource named "validation".
- A module named "acm" which sources from "terraform-aws-modules/acm/aws".

```
1 resource "cloudflare_record" "sample_subdomain" {
2   zone_id = data.cloudflare_zone.this.id
3   name    = var.subdomain
4   type    = "CNAME"
5   value   = module.api_gateway.apigatewayv2_domain_name_target_domain_name
6   ttl     = 300
7   proxied = false
8 }
9
10 resource "cloudflare_record" "validation" {
11   count = length(module.acm.distinct_domain_names)
12
13   zone_id = data.cloudflare_zone.this.id
14   name    = element(module.acm.validation_domains, count.index)["resource_record_name"]
15   type    = element(module.acm.validation_domains, count.index)["resource_record_type"]
16   value   = trimsuffix(element(module.acm.validation_domains, count.index)["resource_record_value"], ".")
17   ttl     = 60
18   proxied = false
19
20   allow_overwrite = true
21 }
22
23 module "acm" {
24   source = "terraform-aws-modules/acm/aws"
25
26   zone_id          = data.cloudflare_zone.this.id
27   domain_name      = "${var.subdomain}.${var.domain}"
28   subject_alternative_names = ["*.${var.subdomain}.${var.domain}"]
29
30   create_route53_records = false
31   validation_record_fqdns = cloudflare_record.validation[*].hostname
32 }
33
```

```
E ⌂ ⌂ ⌂ ⌂ ... 🌐 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

1 resource "cloudflare_record" "sample_subdomain" {
2   zone_id = data.cloudflare_zone.this.id
3   name    = var.subdomain
4   type    = "CNAME"
5   value   = module.api_gateway.apigatewayv2_domain_name_target_domain_name
6   ttl     = 300
7   proxied = false
8 }

9
10 resource "cloudflare_record" "validation" {
11   count = length(module.acm.distinct_domain_names)

12   zone_id = data.cloudflare_zone.this.id
13   name    = element(module.acm.validation_domains, count.index)["resource_record_name"]
14   type    = element(module.acm.validation_domains, count.index)["resource_record_type"]
15   value   = trimsuffix(element(module.acm.validation_domains, count.index)["resource_record_value"], ".")
16   ttl     = 60
17   proxied = false
18
19   allow_overwrite = true
20 }

21 }
22
23 module "acm" {
24   source = "terraform-aws-modules/acm/aws"
25
26   zone_id          = data.cloudflare_zone.this.id
27   domain_name      = "${var.subdomain}.${var.domain}"
28   subject_alternative_names = ["*.${var.subdomain}.${var.domain}"]
29
30   create_route53_records = false
31   validation_record_fqdns = cloudflare_record.validation[*].hostname
32 }

33
```



> config
> lambdas
∨ terraform
 └ api_gateway.tf
 └ cloudflare.tf
 └ data.tf
 └ lambda.tf
 └ main.tf
 └ outputs.tf
 └ variables.tf
 └ versions.tf
 └ vpc.tf

```
1 resource "cloudflare_record" "sample_subdomain" {
2   zone_id = data.cloudflare_zone.this.id
3   name    = var.subdomain
4   type    = "CNAME"
5   value   = module.api_gateway.apigatewayv2_domain_name_target_domain_name
6   ttl     = 300
7   proxied = false
8 }
9
10 resource "cloudflare_record" "validation" {
11   count = length(module.acm.distinct_domain_names)
12
13   zone_id = data.cloudflare_zone.this.id
14   name    = element(module.acm.validation_domains, count.index)["resource_record_name"]
15   type    = element(module.acm.validation_domains, count.index)["resource_record_type"]
16   value   = trimsuffix(element(module.acm.validation_domains, count.index)["resource_record_value"], ".")
17   ttl     = 60
18   proxied = false
19
20   allow_overwrite = true
21 }
22
23 module "acm" {
24   source = "terraform-aws-modules/acm/aws"
25
26   zone_id          = data.cloudflare_zone.this.id
27   domain_name      = "${var.subdomain}.${var.domain}"
28   subject_alternative_names = ["*.${var.subdomain}.${var.domain}"]
29
30   create_route53_records = false
31   validation_record_fqdns = cloudflare_record.validation[*].hostname
32 }
33
```

```
E ⌂ ⌃ ⌁ ⌂ ... api_gateway.tf terraform/api_gateway.tf/ module "api_gateway"/ cors_configuration/[ ]allow_headers/[abc]5
> config
> lambdas
< terraform
  api_gateway.tf
  cloudflare.tf
  data.tf
  lambda.tf
  main.tf
  outputs.tf
  variables.tf
  versions.tf
  vpc.tf

1   module "api_gateway" {
2     source = "terraform-aws-modules/apigateway-v2/aws"
3
4     name          = "${var.stack_name}-api-gateway"
5     description   = "Sample Wekhook Handler - HTTP API Gateway"
6     protocol_type = "HTTP"
7
8     cors_configuration = {
9       allow_headers = ["authorization", "content-type", "x-amz-date", "x-amz-security-token", "x-amz-user-agent", "x-api-key"]
10      allow_methods = ["*"]
11      allow_origins = ["*"]
12    }
13
14    # Custom Domain
15    domain_name           = "${var.subdomain}.${var.domain}"
16    domain_name_certificate_arn = module.acm.acm_certificate_arn
17
18    # Access logs
19    default_stage_access_log_destination_arn = aws_cloudwatch_log_group.api_gateway.arn
20    default_stage_access_log_format          = "$context.identity.sourceIp -- [$context.requestTime] \"$context.httpMethod $context.routeKey $context.protocol\" $context.status $context.responseLength $context.requestId $context.integrationErrorMessage"
21
22    # Routes and integrations
23    integrations = {
24      "POST /v1/sample" = {
25        authorization_type    = "NONE"
26        lambda_arn            = module.sample_lambda.lambda_function_invoke_arn
27        payload_format_version = "2.0"
28        timeout_milliseconds  = 20000
29      }
30    }
31  }
32
33  resource "aws_cloudwatch_log_group" "api_gateway" {
34    name          = "/aws/api_gateway/${module.api_gateway.apigatewayv2_api_id}"
35    retention_in_days = 30
36  }
37
```

```
E ⌂ ⌃ ⌁ ⌂ ... api_gateway.tf terraform/api_gateway.tf/ module "api_gateway"/ cors_configuration/[ ]allow_headers/[abc]5
> config
> lambdas
< terraform
  api_gateway.tf
  cloudflare.tf
  data.tf
  lambda.tf
  main.tf
  outputs.tf
  variables.tf
  versions.tf
  vpc.tf

1   module "api_gateway" {
2     source = "terraform-aws-modules/apigateway-v2/aws"
3
4     name          = "${var.stack_name}-api-gateway"
5     description   = "Sample Wekhook Handler - HTTP API Gateway"
6     protocol_type = "HTTP"
7
8     cors_configuration = {
9       allow_headers = ["authorization", "content-type", "x-amz-date", "x-amz-security-token", "x-amz-user-agent", "x-api-key"]
10      allow_methods = ["*"]
11      allow_origins = ["*"]
12    }
13
14    # Custom Domain
15    domain_name           = "${var.subdomain}.${var.domain}"
16    domain_name_certificate_arn = module.acm.acm_certificate_arn
17
18    # Access logs
19    default_stage_access_log_destination_arn = aws_cloudwatch_log_group.api_gateway.arn
20    default_stage_access_log_format          = "$context.identity.sourceIp -- [$context.requestTime] \"$context.httpMethod $context.routeKey $context.protocol\" $context.status $context.responseLength $context.requestId $context.integrationErrorMessage"
21
22    # Routes and integrations
23    integrations = {
24      "POST /v1/sample" = {
25        authorization_type    = "NONE"
26        lambda_arn            = module.sample_lambda.lambda_function_invoke_arn
27        payload_format_version = "2.0"
28        timeout_milliseconds  = 20000
29      }
30    }
31  }
32
33  resource "aws_cloudwatch_log_group" "api_gateway" {
34    name          = "/aws/api_gateway/${module.api_gateway.apigatewayv2_api_id}"
35    retention_in_days = 30
36  }
37
```

```
E ⌂ ⌃ ⌁ ⌂ ... api_gateway.tf terraform/api_gateway.tf/ module "api_gateway"/ cors_configuration/[ ]allow_headers/[abc]5
> config
> lambdas
< terraform
  api_gateway.tf
  cloudflare.tf
  data.tf
  lambda.tf
  main.tf
  outputs.tf
  variables.tf
  versions.tf
  vpc.tf

1   module "api_gateway" {
2     source = "terraform-aws-modules/apigateway-v2/aws"
3
4     name          = "${var.stack_name}-api-gateway"
5     description   = "Sample Wekhook Handler - HTTP API Gateway"
6     protocol_type = "HTTP"
7
8     cors_configuration = {
9       allow_headers = ["authorization", "content-type", "x-amz-date", "x-amz-security-token", "x-amz-user-agent", "x-api-key"]
10      allow_methods = ["*"]
11      allow_origins = ["*"]
12    }
13
14    # Custom Domain
15    domain_name           = "${var.subdomain}.${var.domain}"
16    domain_name_certificate_arn = module.acm.acm_certificate_arn
17
18    # Access logs
19    default_stage_access_log_destination_arn = aws_cloudwatch_log_group.api_gateway.arn
20    default_stage_access_log_format          = "$context.identity.sourceIp -- [$context.requestTime] \"$context.httpMethod $context.routeKey $context.protocol\" $context.status $context.responseLength $context.requestId $context.integrationErrorMessage"
21
22    # Routes and integrations
23    integrations = {
24      "POST /v1/sample" = {
25        authorization_type      = "NONE"
26        lambda_arn              = module.sample_lambda.lambda_function_invoke_arn
27        payload_format_version = "2.0"
28        timeout_milliseconds   = 20000
29      }
30    }
31  }
32
33  resource "aws_cloudwatch_log_group" "api_gateway" {
34    name          = "/aws/api_gateway/${module.api_gateway.apigatewayv2_api_id}"
35    retention_in_days = 30
36  }
37
```

```
E ⌂ ⌃ ⌁ ⌂ ... api_gateway.tf terraform/api_gateway.tf/ module "api_gateway"/ cors_configuration/[ ]allow_headers/[abc]5
> config
> lambdas
< terraform
  api_gateway.tf
  cloudflare.tf
  data.tf
  lambda.tf
  main.tf
  outputs.tf
  variables.tf
  versions.tf
  vpc.tf

1   module "api_gateway" {
2     source = "terraform-aws-modules/apigateway-v2/aws"
3
4     name          = "${var.stack_name}-api-gateway"
5     description   = "Sample Wekhook Handler - HTTP API Gateway"
6     protocol_type = "HTTP"
7
8     cors_configuration = {
9       allow_headers = ["authorization", "content-type", "x-amz-date", "x-amz-security-token", "x-amz-user-agent", "x-api-key"]
10      allow_methods = ["*"]
11      allow_origins = ["*"]
12    }
13
14    # Custom Domain
15    domain_name           = "${var.subdomain}.${var.domain}"
16    domain_name_certificate_arn = module.acm.acm_certificate_arn
17
18    # Access logs
19    default_stage_access_log_destination_arn = aws_cloudwatch_log_group.api_gateway.arn
20    default_stage_access_log_format          = "$context.identity.sourceIp -- [$context.requestTime] \"$context.httpMethod $context.routeKey $context.protocol\" $context.status $context.responseLength $context.requestId $context.integrationErrorMessage"
21
22    # Routes and integrations
23    integrations = {
24      "POST /v1/sample" = {
25        authorization_type    = "NONE"
26        lambda_arn            = module.sample_lambda.lambda_function_invoke_arn
27        payload_format_version = "2.0"
28        timeout_milliseconds  = 20000
29      }
30    }
31  }
32
33  resource "aws_cloudwatch_log_group" "api_gateway" {
34    name          = "/aws/api_gateway/${module.api_gateway.apigatewayv2_api_id}"
35    retention_in_days = 30
36  }
37
```

```
E ⌂ ⌃ ⌁ ⌂ ... api_gateway.tf terraform/api_gateway.tf/ module "api_gateway"/ cors_configuration/[ ]allow_headers/[abc]5
> config
> lambdas
< terraform
  api_gateway.tf
  cloudflare.tf
  data.tf
  lambda.tf
  main.tf
  outputs.tf
  variables.tf
  versions.tf
  vpc.tf

1   module "api_gateway" {
2     source = "terraform-aws-modules/apigateway-v2/aws"
3
4     name          = "${var.stack_name}-api-gateway"
5     description   = "Sample Wekhook Handler - HTTP API Gateway"
6     protocol_type = "HTTP"
7
8     cors_configuration = {
9       allow_headers = ["authorization", "content-type", "x-amz-date", "x-amz-security-token", "x-amz-user-agent", "x-api-key"]
10      allow_methods = ["*"]
11      allow_origins = ["*"]
12    }
13
14  # Custom Domain
15  domain_name           = "${var.subdomain}.${var.domain}"
16  domain_name_certificate_arn = module.acm.acm_certificate_arn
17
18  # Access logs
19  default_stage_access_log_destination_arn = aws_cloudwatch_log_group.api_gateway.arn
20  default_stage_access_log_format          = "$context.identity.sourceIp -- [$context.requestTime] \"$context.httpMethod $context.routeKey $context.protocol\" $context.status $context.responseLength $context.requestId $context.integrationErrorMessage"
21
22  # Routes and integrations
23  integrations = {
24    "POST /v1/sample" = {
25      authorization_type      = "NONE"
26      lambda_arn              = module.sample_lambda.lambda_function_invoke_arn
27      payload_format_version = "2.0"
28      timeout_milliseconds   = 20000
29    }
30  }
31 }
32
33 resource "aws_cloudwatch_log_group" "api_gateway" {
34   name          = "/aws/api_gateway/${module.api_gateway.apigatewayv2_api_id}"
35   retention_in_days = 30
36 }
37
```

```
E ⌂ ⌃ ⌁ ⌂ ... api_gateway.tf terraform/api_gateway.tf/ module "api_gateway"/ cors_configuration/[ ]allow_headers/[abc]5
> config
> lambdas
< terraform
  api_gateway.tf
  cloudflare.tf
  data.tf
  lambda.tf
  main.tf
  outputs.tf
  variables.tf
  versions.tf
  vpc.tf

1   module "api_gateway" {
2     source = "terraform-aws-modules/apigateway-v2/aws"
3
4     name          = "${var.stack_name}-api-gateway"
5     description   = "Sample Wekhook Handler - HTTP API Gateway"
6     protocol_type = "HTTP"
7
8     cors_configuration = {
9       allow_headers = ["authorization", "content-type", "x-amz-date", "x-amz-security-token", "x-amz-user-agent", "x-api-key"]
10      allow_methods = ["*"]
11      allow_origins = ["*"]
12    }
13
14    # Custom Domain
15    domain_name           = "${var.subdomain}.${var.domain}"
16    domain_name_certificate_arn = module.acm.acm_certificate_arn
17
18    # Access logs
19    default_stage_access_log_destination_arn = aws_cloudwatch_log_group.api_gateway.arn
20    default_stage_access_log_format          = "$context.identity.sourceIp -- [$context.requestTime] \"$context.httpMethod $context.routeKey $context.protocol\" $context.status $context.responseLength $context.requestId $context.integrationErrorMessage"
21
22    # Routes and integrations
23    integrations = {
24      "POST /v1/sample" = {
25        authorization_type    = "NONE"
26        lambda_arn            = module.sample_lambda.lambda_function_invoke_arn
27        payload_format_version = "2.0"
28        timeout_milliseconds  = 20000
29      }
30    }
31  }
32
33  resource "aws_cloudwatch_log_group" "api_gateway" {
34    name          = "/aws/api_gateway/${module.api_gateway.apigatewayv2_api_id}"
35    retention_in_days = 30
36  }
37
```

The screenshot shows a code editor interface with a dark theme. On the left is a sidebar containing a file tree:

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

The main area displays the content of the selected file, `lambda.tf`. The code defines a module named "sample_lambda" which uses the `aws_lambda` provider from the `terraform-aws-modules/lambda/aws` repository. The module configuration includes:

- `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]`

Below this, another module definition for `sample_docker_image` is shown, which uses the `aws_lambda_docker_build` provider.

```
λ lambda.tf terraform/lambda.tf/terraform module "sample_lambda"
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
```

```
λ lambda.tf terraform/lambda.tf/terraform module "sample_lambda"
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
```

```
λ lambda.tf terraform/lambda.tf/terraform module "sample_lambda"
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
```

E ⌘ ⌘ ⌘ ⌘ ⌘ ⌘ ⌘ ⌘ ⌘ ⌘ ⌘

Dockerfile lambdas/sample_with_requirements/Dockerfile/ FROM

□ × ⌘

> config
└ lambdas
 > sample
 └ sample_with_requi...
 └ src
 index.py
 Dockerfile
 requirements.txt

```
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
```

The screenshot shows a code editor interface with a dark theme. On the left is a file tree:

- > 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

The Dockerfile content is as follows:

```
FROM public.ecr.aws/lambda/python:3.10
COPY requirements.txt .
COPY src .
RUN pip install -r requirements.txt
CMD [ "index.lambda_handler" ]
```

Line 1, 'FROM public.ecr.aws/lambda/python:3.10', is highlighted with a red border.



> config
✓ lambdas
> sample
✓ sample_with_requi...
✓ src

index.py

Dockerfile

☰ requirements.txt

✓ terraform

Y api_gateway.tf
Y cloudflare.tf
Y data.tf
Y lambda.tf
Y main.tf
Y outputs.tf
Y variables.tf
Y versions.tf
Y 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
```

E ⌘ ⌘ ⌘ ⌘ ⌘ ⌘ ⌘ ⌘ ⌘ ⌘ ⌘ ⌘ ⌘

Dockerfile lambdas/sample_with_requirements/Dockerfile/ FROM

□ X ⌘

> 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
```

The screenshot shows a code editor interface with a dark theme. On the left is a file tree, and on the right is a code editor pane.

File Tree:

- > 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

≡ requirements.txt lambdas/sample_with_requirements/requirements.txt

```
1 aws_lambda_powertools == 2.19.0
2
```

The screenshot shows a code editor interface with a sidebar and a main editor area.

File Explorer (Left):

- > 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

Main Editor Area (Right):

```
1 """Sample Lambda with Python Requirements"""
2 import json
3 import os
4
5 from aws_lambda_powertools import Logger
6
7 LOGGER = Logger(level=os.environ.get("LOG_LEVEL"), default="WARNING")
8
9
10 def lambda_handler(event, context) -> None:
11     """This is the function called during lambda invocation."""
12
13     LOGGER.debug(event)
14     LOGGER.debug(context)
15
16     body = event.get("body")
17     data = json.loads(body)
18
19     print(f"Received data from webhook:\n{data}")
20     print("Processing...")
21
```

E ⌘ ⌘ ⌘ ⌘ ... λ lambda.tf terraform/lambda.tf/ module "sample_lambda"

> config

> lambdas

✗ terraform

 └── api_gateway.tf

 └── cloudflare.tf

 └── data.tf

 └── **λ lambda.tf**

 └── main.tf

 └── outputs.tf

 └── variables.tf

 └── versions.tf

 └── vpc.tf

```
1 module "sample_lambda" {
2   source  = "terraform-aws-modules/lambda/aws"
3   version = "~> 4.12.1"
4
5   function_name = "${var.stack_name}-lambda-enroll"
6
7   create_package = false
8   publish        = true
9
10  image_uri      = module.sample_docker_image.image_uri
11  package_type   = "Image"
12  timeout        = 60
13
14  attach_network_policy = true
15  vpc_subnet_ids       = data.aws_subnets.private.ids
16  vpc_security_group_ids = [module.security_group.security_group_id]
17
18  environment_variables = {
19    ENVIRONMENT = var.environment
20    LOG_LEVEL   = "INFO"
21  }
22
23  allowed_triggers = {
24    AllowAPIGatewayPostEnroll = {
25      service      = "apigateway"
26      source_arn  = "${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 = isonencode{
```

The screenshot shows a code editor interface with a dark theme. On the left is a sidebar with a tree view of files:

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

The main pane displays the contents of the `lambda.tf` file. The code defines a Terraform module named `sample_lambda` which uses the `aws_lambda` provider from the `terraform-aws-modules/lambda/aws` repository. The module creates a Lambda function with the following configuration:

- `function_name = "${var.stack_name}-lambda-enroll"`
- `create_package = false` (highlighted with a red box)
- `publish = true`
- `image_uri = module.sample_docker_image.image_uri`
- `package_type = "Image"`
- `timeout = 60`
- `attach_network_policy = true`
- `vpc_subnet_ids = data.aws_subnets.private.ids`
- `vpc_security_group_ids = [module.security_group.security_group_id]`
- `environment_variables = { ENVIRONMENT = var.environment LOG_LEVEL = "INFO" }`
- `allowed_triggers = { AllowAPIGatewayPostEnroll = { service = "apigateway" source_arn = "${module.api_gateway.apigatewayv2_api_execution_arn}/*/POST/v1/sample" } }`
- `depends_on = [module.sample_docker_image]`

Below this, another module definition for `sample_docker_image` is shown, which uses the `aws_lambda_docker_build` provider from the same repository.

The screenshot shows a code editor interface with a dark theme. On the left is a sidebar with a tree view of files:

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

The main area displays Terraform code for a Lambda function. The code defines a module named "sample_lambda" which sources from "terraform-aws-modules/lambda/aws". It specifies a function name, creates a package (set to false), publishes it (set to true), and defines several configuration parameters. A red box highlights the "image_uri" and "package_type" settings. The code also defines environment variables, allowed triggers (an API Gateway post-enroll trigger), and dependencies on another module named "sample_docker_image". The "sample_docker_image" module is defined at the bottom of the file.

```
lambda.tf terraform/lambda.tf/terraform module "sample_lambda"

1 module "sample_lambda" {
2   source  = "terraform-aws-modules/lambda/aws"
3   version = "~> 4.12.1"
4
5   function_name = "${var.stack_name}-lambda-enroll"
6
7   create_package = false
8   publish        = true
9
10  image_uri      = module.sample_docker_image.image_uri
11  package_type   = "Image"
12  timeout        = 60
13
14  attach_network_policy = true
15  vpc_subnet_ids     = data.aws_subnets.private.ids
16  vpc_security_group_ids = [module.security_group.security_group_id]
17
18  environment_variables = {
19    ENVIRONMENT = var.environment
20    LOG_LEVEL   = "INFO"
21  }
22
23  allowed_triggers = {
24    AllowAPIGatewayPostEnroll = {
25      service    = "apigateway"
26      source_arn = "${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 = isonencode{
```

```
lambda.tf terraform/lambda.tf/terraform module "sample_lambda"

1 module "sample_lambda" {
2   source  = "terraform-aws-modules/lambda/aws"
3   version = "~> 4.12.1"
4
5   function_name = "${var.stack_name}-lambda-enroll"
6
7   create_package = false
8   publish        = true
9
10  image_uri      = module.sample_docker_image.image_uri
11  package_type   = "Image"
12  timeout        = 60
13
14  attach_network_policy = true
15  vpc_subnet_ids     = data.aws_subnets.private.ids
16  vpc_security_group_ids = [module.security_group.security_group_id]
17
18  environment_variables = {
19    ENVIRONMENT = var.environment
20    LOG_LEVEL   = "INFO"
21  }
22
23  allowed_triggers = {
24    AllowAPIGatewayPostEnroll = {
25      service      = "apigateway"
26      source_arn   = "${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 = isonencode{
```

vpc.tf terraform/vpc.tf/ module "security_group"

```
1 module "security_group" {
2   source  = "terraform-aws-modules/security-group/aws"
3   version = "~> 4.0"
4
5   name      = "${var.stack_name}-sg-egress"
6   description = "Security Group for Lambda Egress"
7
8   vpc_id = data.aws_vpc.sample.id
9
10  egress_with_cidr_blocks = [
11    {
12      from_port    = 0
13      to_port     = 0
14      protocol    = -1
15      cidr_blocks = "0.0.0.0/0"
16    }
17  ]
18 }
19
```

E ⌂ ⌂ ⌂ ⌂ ⌂ ⌂ ⌂ ⌂ ⌂

lambda.tf terraform/lambda.tf/ module "sample_lambda"

□ X ...

> config
> lambdas
✓ terraform
 └ api_gateway.tf
 └ cloudflare.tf
 └ data.tf
 └ lambda.tf **selected**
 └ main.tf
 └ outputs.tf
 └ variables.tf
 └ versions.tf
 └ vpc.tf

```
1 module "sample_lambda" {
2   source  = "terraform-aws-modules/lambda/aws"
3   version = "~> 4.12.1"
4
5   function_name = "${var.stack_name}-lambda-enroll"
6
7   create_package = false
8   publish        = true
9
10  image_uri     = module.sample_docker_image.image_uri
11  package_type = "Image"
12  timeout       = 60
13
14  attach_network_policy = true
15  vpc_subnet_ids     = data.aws_subnets.private.ids
16  vpc_security_group_ids = [module.security_group.security_group_id]
17
18  environment_variables = {
19    ENVIRONMENT = var.environment
20    LOG_LEVEL   = "INFO"
21  }
22
23  allowed_triggers = {
24    AllowAPIGatewayPostEnroll = {
25      service    = "apigateway"
26      source_arn = "${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 = isonencode{
```

The screenshot shows a code editor interface with a dark theme. On the left is a file tree sidebar containing files: config, lambdas, terraform, api_gateway.tf, cloudflare.tf, data.tf, lambda.tf (selected), main.tf, outputs.tf, variables.tf, versions.tf, and vpc.tf. The main editor area displays Terraform code for a Lambda module named "sample_lambda". A red box highlights the network configuration block from line 14 to line 44.

```
lambda.tf  terraform/lambda.tf/module "sample_lambda"

1 module "sample_lambda" {
2   source  = "terraform-aws-modules/lambda/aws"
3   version = "~> 4.12.1"
4
5   function_name = "${var.stack_name}-lambda-enroll"
6
7   create_package = false
8   publish        = true
9
10  image_uri     = module.sample_docker_image.image_uri
11  package_type  = "Image"
12  timeout       = 60
13
14  attach_network_policy  = true
15  vpc_subnet_ids      = data.aws_subnets.private.ids
16  vpc_security_group_ids = [module.security_group.security_group_id]
17
18  environment_variables = {
19    ENVIRONMENT = var.environment
20    LOG_LEVEL   = "INFO"
21  }
22
23  allowed_triggers = {
24    AllowAPIGatewayPostEnroll = {
25      service    = "apigateway"
26      source_arn = "${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 = isonencode{
```

```
E ⌂ ⌂ ⌂ ⌂ ⌂ ... λ 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_endpoint" {
  description = "The URI of the API"
  value       = module.api_gateway.apigatewayv2_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
}
```

```
λ lambda.tf terraform/lambda.tf/ module "sample_lambda"
127   output "lambda_function_qualified_arn" {
128     description = "The ARN identifying your Lambda Function Version"
129     value       = module.sample_lambda.lambda_function_qualified_arn
130   }
131
132   output "sample_lambda_function_version" {
133     description = "Latest published version of Lambda Function"
134     value       = module.sample_lambda.lambda_function_version
135   }
136
137   output "lambda_function_last_modified" {
138     description = "The date Lambda Function resource was last modified"
139     value       = module.sample_lambda.lambda_function_last_modified
140   }
141
142   output "lambda_function_source_code_hash" {
143     description = "Base64-encoded representation of raw SHA-256 sum of the zip file"
144     value       = module.sample_lambda.lambda_function_source_code_hash
145   }
146
147 # IAM Role
148 output "lambda_role_arn" {
149   description = "The ARN of the IAM role created for the Lambda Function"
150   value       = module.sample_lambda.lambda_role_arn
151 }
152
153 output "lambda_role_name" {
154   description = "The name of the IAM role created for the Lambda Function"
155   value       = module.sample_lambda.lambda_role_name
156 }
157
158 # CloudWatch Log Group
159 output "lambda_cloudwatch_log_group_arn" {
160   description = "The ARN of the Cloudwatch Log Group"
161   value       = module.sample_lambda.lambda_cloudwatch_log_group_arn
162 }
163
```

Questions?



<https://github.com/MScottBlake/Presentations>

<https://bit.ly/psumac2023-111>



<https://bit.ly/psumac2023-111>