Terraform Configuration Guidance for 2023 Direct Connect Architecture Migration

This page is being retained for historical purposes, but is no longer maintained. All relevant Direct Connect information about the current (2023 and after) Direct Connect architecture has been migrated to primary customer Direct Connect documentation, Cornell AWS Direct Connect.

Introduction

Part of the 2023 Cornell AWS Direct Connect Architecture Migration process creates new tags on Cornell AWS VPCs that use Direct Connect. Those tags, prefixed by "cit:", can cause Terraform to hiccup if you use Terraform to manage AWS network resources.

Please don't allow Terraform to delete the tags prefixed by "cit:", or the "Cost Center" tag! They are important for the migration to the v2 Direct Connect architecture. If you (or Terraform) delete those tags, they will be recreated before the migration proceeds. Deleting the "Cost Center" tag on TGW attachments will result in customers paying for TGW attachment costs instead of CIT.

This is what it looks like when Terraform finds those tags, and makes a plan to delete them:

```
# terraform plan
aws_vpc.blank-vpc: Refreshing state... [id=vpc-cde7e0a8]
...
aws_route_table_association.v2-private-1: Refreshing state... [id=rtbassoc-08f9e7ea923cc8454]

Terraform used the selected providers to generate the following execution plan. Resource actions are
indicated with the following symbols:
  ~ update in-place

Terraform will perform the following actions:

  # aws_subnet.example will be updated in-place
  ~ resource "aws_subnet" "example" {
        id                                              = "subnet-0d705338215b4d08b"
      ~ tags                                            = {
          - "cit:dc-arch-migration-description" = "No change." -> null
          - "cit:dc-arch-migration-target"      = "no" -> null
          - "cit:dc-arch-version"               = "v1" -> null
          - "cit:subnet-type"                   = "public" -> null
            # (1 unchanged element hidden)
        }
      ~ tags_all                                        = {
          - "cit:dc-arch-migration-description" = "No change." -> null
          - "cit:dc-arch-migration-target"      = "no" -> null
          - "cit:dc-arch-version"               = "v1" -> null
          - "cit:subnet-type"                   = "public" -> null
            # (1 unchanged element hidden)
        }
        # (14 unchanged attributes hidden)
    }

Plan: 0 to add, 1 to change, 0 to destroy.
```

Tell Terraform to Ignore the Tags

There are two options to tell Terraform to ignore the "cit:" tags, depending on the AWS provider version you are using:

- If using AWS provider version >= 2.60.0, you can configure a global **`ignore_tags`** setting in the provider configuration. This is by far the simplest approach.
- If using an earlier provider version, you will need to a **`lifecycle`** stanza to the all the affected resources and setting the **`ignore_changes`** attribute.

Since the "Cost Center" tag is added only to a new Transit Gateway Attachment that will be added to VPCs, you probably won't need to include it in configurations shown below, unless you plan to import the new TGW Attachment resources into your own Terraform configuration.

`ignore_tags` in AWS Provider Configuration

This option can be used for any AWS provider version >= 2.60.0

```
provider "aws" {
  # ... potentially other configuration ...

  ignore_tags {
    key_prefixes = ["cit:"]
  }
}
```

`ignore_changes` in **`lifecycle`** stanza for Each Resource

Terraform Versions >= 0.12.3

Recent Terraform Versions (>= v0.13)

```
resource "aws_subnet" "example" {
  cidr_block        = "10.92.117.128/25"
  vpc_id            = aws_vpc.example.id

  ...

  tags = {
    Name = "example-subnet"
  }

  lifecycle {
    ignore_changes = [
      tags["cit:dc-arch-migration-description"],
      tags["cit:dc-arch-migration-target"],
      tags["cit:dc-arch-version"],
      tags["cit:dc-vgw"],
      tags["cit:subnet-type"],
      tags["cit:tgw-attachment-target"],
      tags["cit:tgw-attachment-guidance"],
          tags["Cost Center"],
     ]
  }
}
```

Terraform Version 0.12.0 through 0.12.2

You will need to upgrade Terraform to at least version 0.12.3 and then use the configuration above.

Terraform Versions 0.11.x

Terraform v0.11.x

```
resource "aws_subnet" "example" {
  cidr_block        = "10.92.117.128/25"
  vpc_id            = aws_vpc.example.id

  ...

  tags = {
    Name = "example-subnet"
  }

  lifecycle {
    ignore_changes = [
          "tags.%",
      "tags.cit:dc-arch-migration-description",
      "tags.cit:dc-arch-migration-target",
      "tags.cit:dc-arch-version",
      "tags.cit:dc-vgw",
      "tags.cit:subnet-type",
      "tags.cit:tgw-attachment-target",
      "tags.cit:tgw-attachment-guidance",
      "tags.Cost Center",
     ]
  }
}
```

Last Ditch Options

If your Terraform version or AWS provider version doesn't support (or behave as expected) with the options above, you should be able to, at least, tell Terraform to ignore all changes to tags, as shown below:

```
  lifecycle {
    ignore_changes = [ tags ]
  }
```

Or...

```
  lifecycle {
    ignore_changes = [ "tags" ]
  }
```

Terraform Configuration Templates for Added Resources

This guide does not describe how to import pre-existing resources into Terraform. See Terraform documentation/tutorial for how-to and concepts: Import Terraform Configuration

Generally, you will take these steps:

1. Add the configuration below to your Terraform configuration.
2. Edit/customize the added configuration to match reality.
3. Import the resources into your Terraform state. (See comments in the **.tf** files below for specific import commands.)
4. Continue to edit/customer the configuration below until a **terraform plan** doesn't result in Terraform wanting to make any changes.

Utility Subnet Resources

This Terraform configuration will need minor edits to correspond to the resources in your VPC.

Terraform configuration	utility-resources.tf

Route Table Resources

This Terraform configuration is a super basic template of the v2 Route Table resources added to customer VPCs. You will generally want to take one of two approaches:

- Start with the template below and copy/paste/edit to get to a Terraform configuration that matches reality.
  - OR -
- Copy the pre-existing Terraform configuration for your route tables and add elements from the Terraform template below, to get to a Terraform configuration that matches reality.

Terraform template	route-tables.tf

Other Resources

Transit Gateway Attachment

We recommend that you **do not** import the Transit Gateway Attachment resource directly into your Terraform configuration. The utility-resources.tf template provides a Terraform **data** source that you can use in your configuration when you need to reference the TGW ID or the TGW Attachment ID elsewhere in your Terraform.

Secondary VPC CIDR Block

Each customer VPC had secondary a CIDR block added to it. The utility subnets were created from this CIDR. We **do not** recommend adding a **aws_vpc_ipv4_cidr_block_association** resource to your Terraform configuration in order to manage this secondary CIDR block into your Terraform configuration. The reason behind this is that these secondary CIDRs were allocated using Amazon VPC IP Address Manager (IPAM) and customers do not have privileges to these IPAM resources.

Virtual Private Gateway

Virtual Private Gateways (VGW) will be detached from VPCs on 19 Jan 2023 by CIT. During the week of 23 Jan 2023, VGWs will be deleted entirely. Your Terraform configuration will need to be updated accordingly.

References

- Cornell
  - 2023 Cornell AWS Direct Connect Architecture Migration
- Terraform
  - The lifecycle Meta-Argument
- AWS
  - What is IPAM?