AWS Multi-Account, Multi-Region Networking with Terraform

17 November 2020

Agenda

- Who are we?
- Problem Statement
- Tools
- Account and Network Diagram
- Terraform Solution

trility

Who are we?

Eric Gerling, Senior DevOps Architect, Trility Consulting

Background as a full-stack cloud software and systems architect focused on the simplest solution for the problem to be solved. Hands-on technical leadership with cloud architecture capacity, systems administration, full-stack testing and development in the insurance, agribusiness, Internet of Things, financial payments, and cable industries.

Fluent in cloud technologies and working on middle- and lower-tier technologies where security, performance, scalability, and reliability are the most important invisible attributes no one asks for, but everyone expects.

Who are we?

Nathan Levis, Solutions Architect, Trility Consulting

Nathan helps clients identify and craft solutions for clients to simplify, automate, and secure their paths forward. He focuses on 1. Incremental progress across all aspects of the professional environment, 2. Uses agile (little ‘a’, not big ‘A’) methodologies to improve existing and future operations, 3. Leverages strong social and professional skills to encourage team members to evaluate existing and future procedures and behaviors, and 4. Strives to take a pragmatic and open-minded approach to every situation.

AWS Certified Solutions Architect - Associate

trility

Problem Statement

How do I manage multiple AWS accounts with resources spread across multiple regions in a simple, cost effective way? Desired attributes include:

- Infrastructure as Code (Terraform)
- Single Source of Truth for AWS Accounts
- Rapid deployment of new regions
- VPN Access with access to dynamically created VPCs in multiple regions

trility

Problem Statement: Breakout

Infrastructure as Code

- All things in code, all the time, for everything.
- Terraform
  - State File Management
  - IMPORT!!

Rapid Deployment of New Regions

- Active - Active/Passive/Warm
- Regulatory compliance constantly changing
- Security also

Single Source of Truth for AWS Accounts

- Accounts are not static, they live and breath
- Account creation for new projects

VPN Access with access to dynamically created VPCs in multiple regions

- Dynamic Development Environments
- Data Analysts, Developer Access, Production Support

trility

Tools

- Terraform
- CI/CD Orchestration
- AWS Services (Organizations, IAM, CloudTrail, Config, Lambda, and many others)

trility

Tools: Terraform

From the Hashicorp web site:
"Terraform is a tool for building, changing, and versioning infrastructure safely and efficiently."

Cool.

- Execution Plans
- Resource Graphs
- Change Automation
- Automated and fast

What about …

- CloudFormation (AWS)
- Cloud Development Kit (AWS)
- Deployment Manager (GCP)
- ARM Templates (Azure)
- Chef/Puppet/Ansible

All good tools, no question. However, multi-cloud is always part of the conversation. Terraform is a single tool teams can leverage across any cloud.

Tools: CI/CD Orchestration

Traditional Continuous Integration/Continuous Deployment

- Take code from the developer, run tests, scan for security vulnerabilities, run more tests, test it again, deploy it to production, repeat

Why is infrastructure any different?

Wait? Automate infrastructure deployments? Are you crazy?

Common Tools

- Jenkins
- Concourse
- CircleCI
- AWS CodeBuild, CodeDeploy, CodePipeline

Don’t forget! Your orchestration jobs should be in code too, but that’s another topic for another day.

Tools: AWS Services

Account Services

- Organizations
- CloudTrail
- Config
- Identity and Access Management
- S3
- CloudWatch

Networking Services

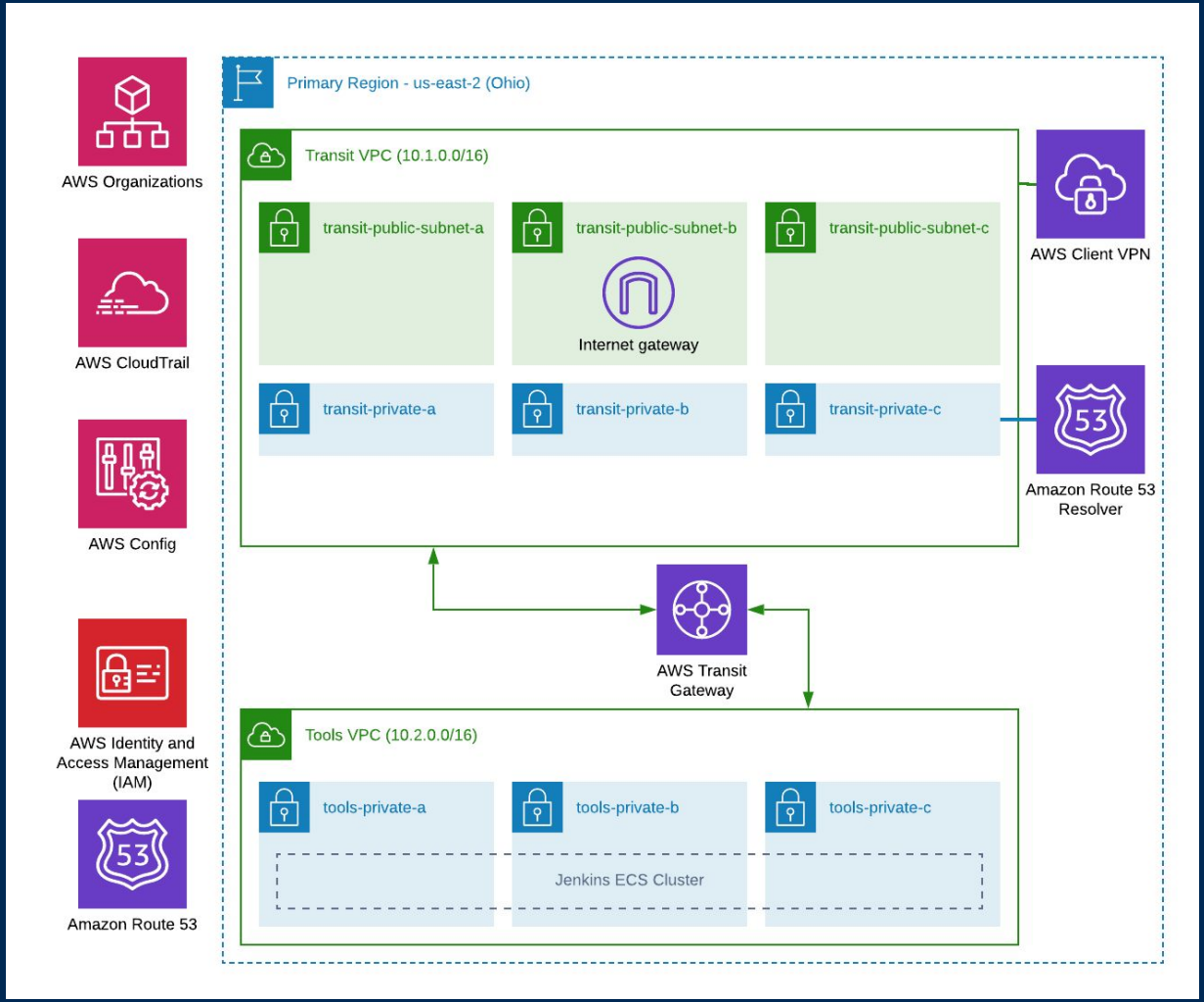
- Transit Gateway
- VPC
- AWS Client VPN
- Route 53
- Route 53 Resolver

Diagrams

- Master Account
- Sub, or Child, Accounts

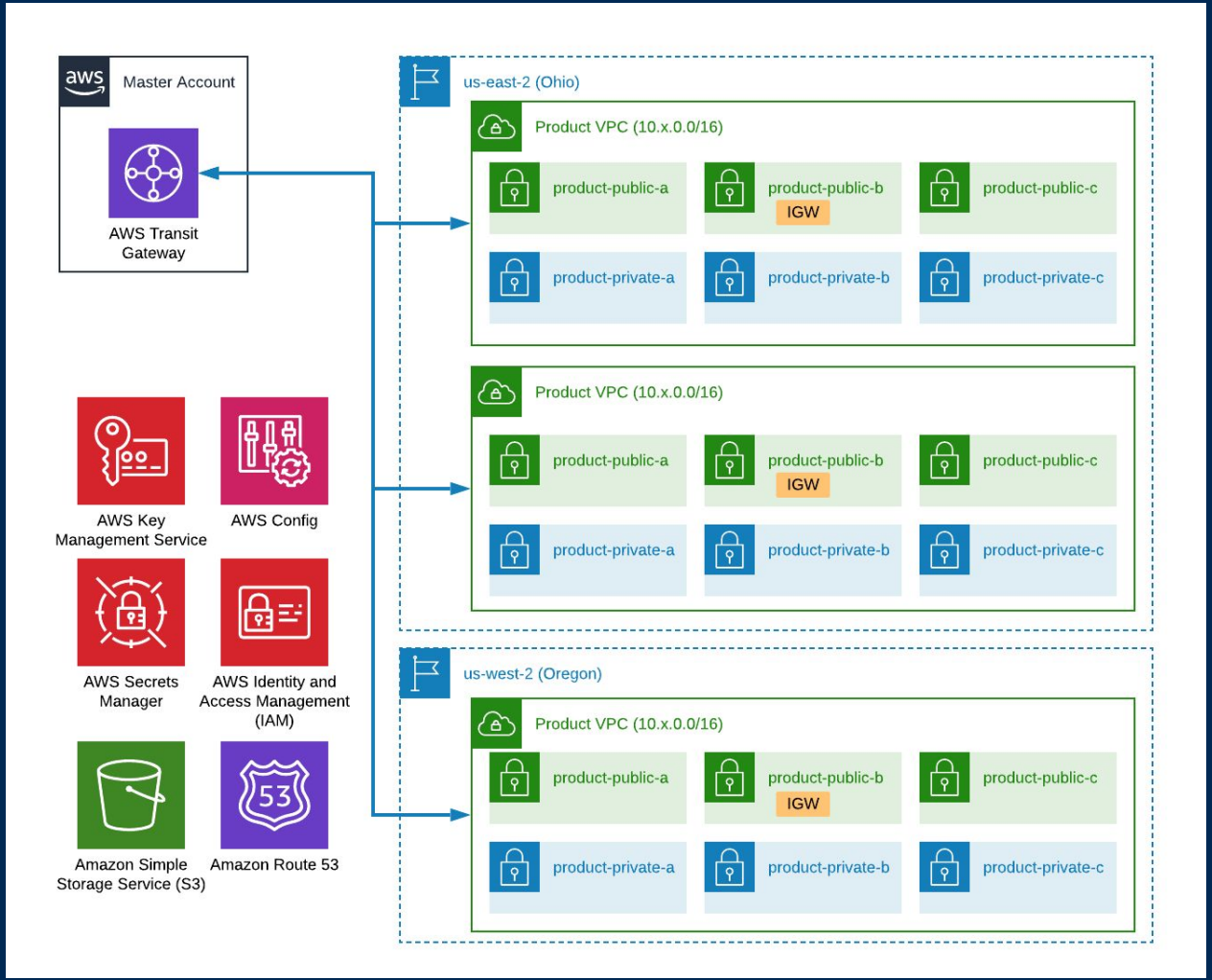
trility

Master AWS Account Diagram



trility

AWS Sub-Account Diagram



trility

Putting It All Together

Time to dive in and start building.

- Foundation
- State Files
- Workspaces
- Providers
- Modules
- Code

Putting It All Together: Foundation

Terraform

- CLI vs Terraform Cloud vs TFE
- Pin the Versions?

```
terraform {
  required_version = "0.12.12"

  required_providers {
    aws = "~> 2.3"
  }
}
```

Source Code Management

- Terraform Modules
  - Single Repository
  - Multiple Repositories
- Core Terraform Code
  - Master Account
  - Sub Account
  - Transit VPC
  - Client VPN
  - DNS Zones

trility

Putting It All Together: State Files

State Files

- The Crown Jewels
- Lots of options, do not ever use local storage. Ever.
- Security also.
- For AWS, can't go wrong with S3
  - Versioning
  - Logging
  - Bucket Policy
- Number of State Files

Terraform Backend Configuration

```
terraform {
  backend "s3" {
    bucket  = "account-backend"
    key     = "master_account"
    region  = "us-east-2"
    encrypt = true
  }
}
```

Import

- Seriously - import

trility

Putting It All Together: Workspaces

Terraform Workspaces

- Collections of Infrastructure
  - Maximize Code Reuse

Terraform Workspaces

- Local
- Data sources
- Variables

```
locals {
  vpc_name = terraform.workspace == "default" ? "transit" : "transit-${terraform.workspace}"
  transit_gw_name = terraform.workspace == "default" ? "tgw" : "tgw-${terraform.workspace}"

  tags = {
    Owner     = var.owner
    Workspace = terraform.workspace
  }
}

data "aws_caller_identity" "current" {}

data "aws_iam_account_alias" "current" {}
```

Putting It All Together: Providers

Terraform Providers

- Manages API Interactions
- Continuously growing and changing
- Can’t find one, write one

```
provider "aws" {
  region = "us-east-2"
}

provider "aws" {
  alias  = "useast1"
  region = "us-east-1"
}

provider "aws" {
  alias  = "uswest2"
  region = "us-west-2"
}
```

```
provider "aws" {
  region = var.region

  assume_role {
    role_arn = "${var.roles[terraform.workspace]}"
  }
}

provider "aws" {
  alias  = "useast1"
  region = "us-east-1"

  assume_role {
    role_arn = "${var.roles[terraform.workspace]}"
  }
}

provider "kafka" {
  bootstrap_servers = ["localhost:9092"]
}

provider "google" {
  project = "newco-app"
  region  = "us-central1"
}
```

Putting It All Together: Modules

iam_account_alias Module

```
variable "alias" {
  description = "AWS Account Alias"
  type        = string
}

resource "aws_iam_account_alias" "alias" {
  account_alias = var.alias
}
```

Module Reference

```
module "account_alias" {
  alias  = var.account_alias
  source = "git::https://gitserver/terraform-modules.git//iam_account_alias?ref=v1"
}
```

trility

Putting It All Together: Modules

s3_bucket Multi-purpose Module

```
resource "aws_s3_bucket" "bucket" {
  bucket        = var.name
  acl           = var.acl
  region        = var.region
  force_destroy = var.force_destroy
  tags          = "${merge(map("Name", var.name), var.tags)}"

  versioning {
    enabled = var.object_versioning
  }

  server_side_encryption_configuration {
    rule {
      apply_server_side_encryption_by_default {
        sse_algorithm     = var.sse_algorithm
        kms_master_key_id = var.kms_key_arn
      }
    }
  }

  dynamic "logging" {
    for_each = var.logging_bucket == "" ? [] : [var.logging_bucket]

    content {
      target_bucket = var.logging_bucket
      target_prefix = "${var.name}/"
    }
  }
```

```
  dynamic "lifecycle_rule" {
    for_each = var.lifecycle_ttl == null ? [] : var.lifecycle_ttl

    content {
      enabled = true
      prefix  = lifecycle_rule.value.prefix
      tags    = lifecycle_rule.value.tags
      expiration {
        days = lifecycle_rule.value.expiration_days
      }
    }
  }

  lifecycle {
    ignore_changes = [
      region,
    ]
  }
}

resource "aws_s3_bucket_public_access_block" "public_block" {
  count = var.s3_website == true || var.s3_301_redirect == true ? 0 : 1
  bucket = aws_s3_bucket.bucket.id
  block_public_acls       = true
  block_public_policy     = true
  ignore_public_acls      = true
  restrict_public_buckets = true
}
```

trility

Code

Questions?

Eric Gerling
Senior DevOps Architect
Trility Consulting
eric@trility.io

Nathan Levis
Solutions Architect
Trility Consulting
nathan.levis@trility.io

trility