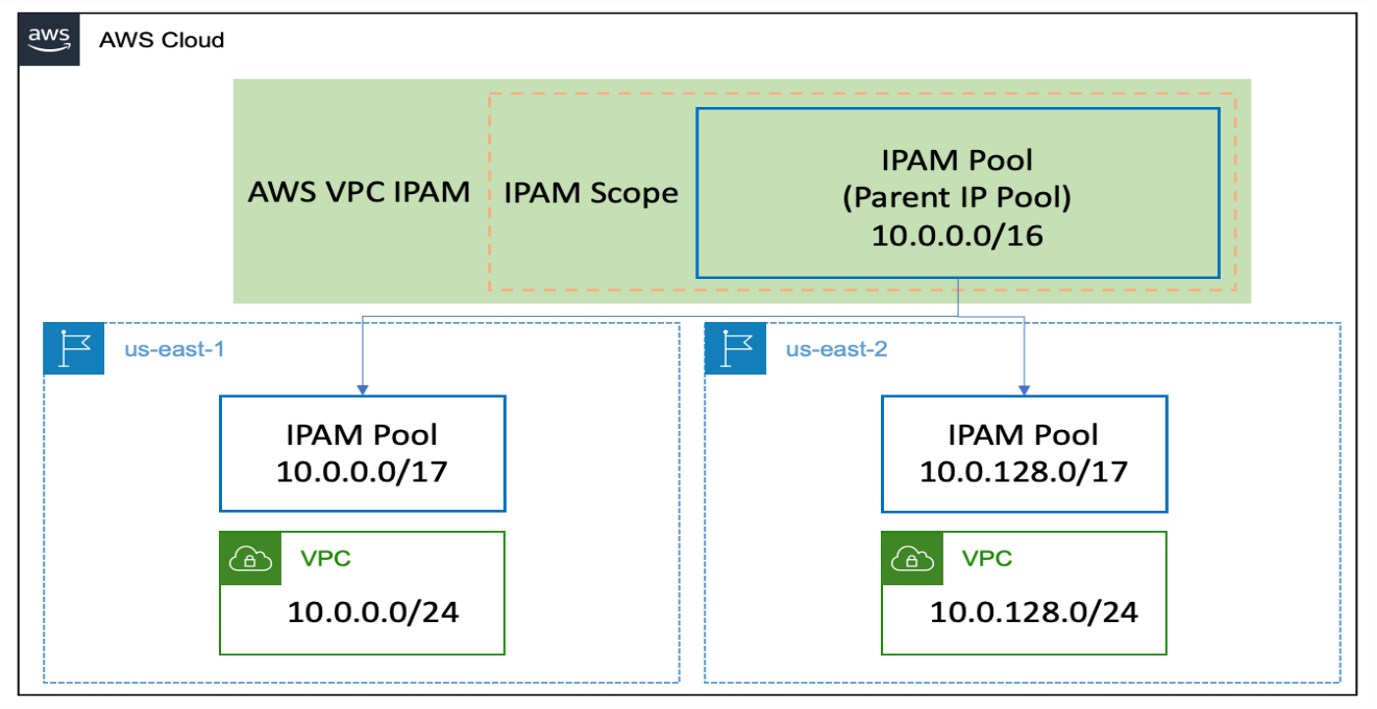
**Managing Cloud Resources with Terraform**

[AWS HashiCorp Terraform]

**Project Description – Scenario**

The intent of this project is to educate the cloud practitioners on the use of Terraform to manage their cloud resources.

This project introduces managing cloud resources with Terraform and Terraform Providers using IaC. This workshop is composed of modules, and each module has a specific learning objective.

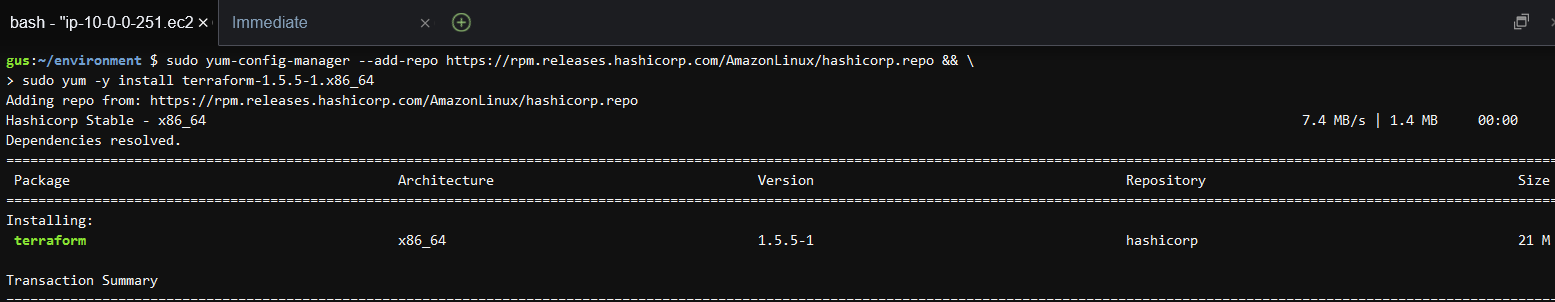
**Architecture - Overview**

**Provisioned Cloud9 environment**

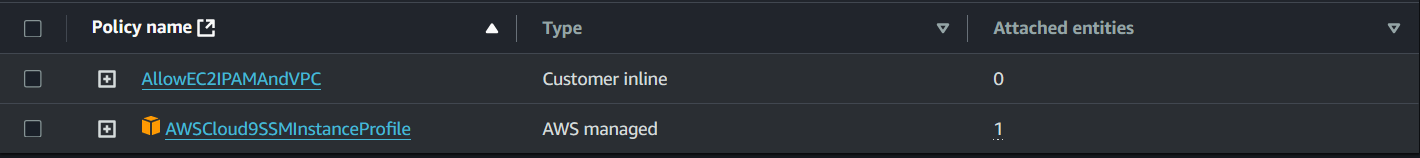
**A screenshot of a computer

Description automatically generated**

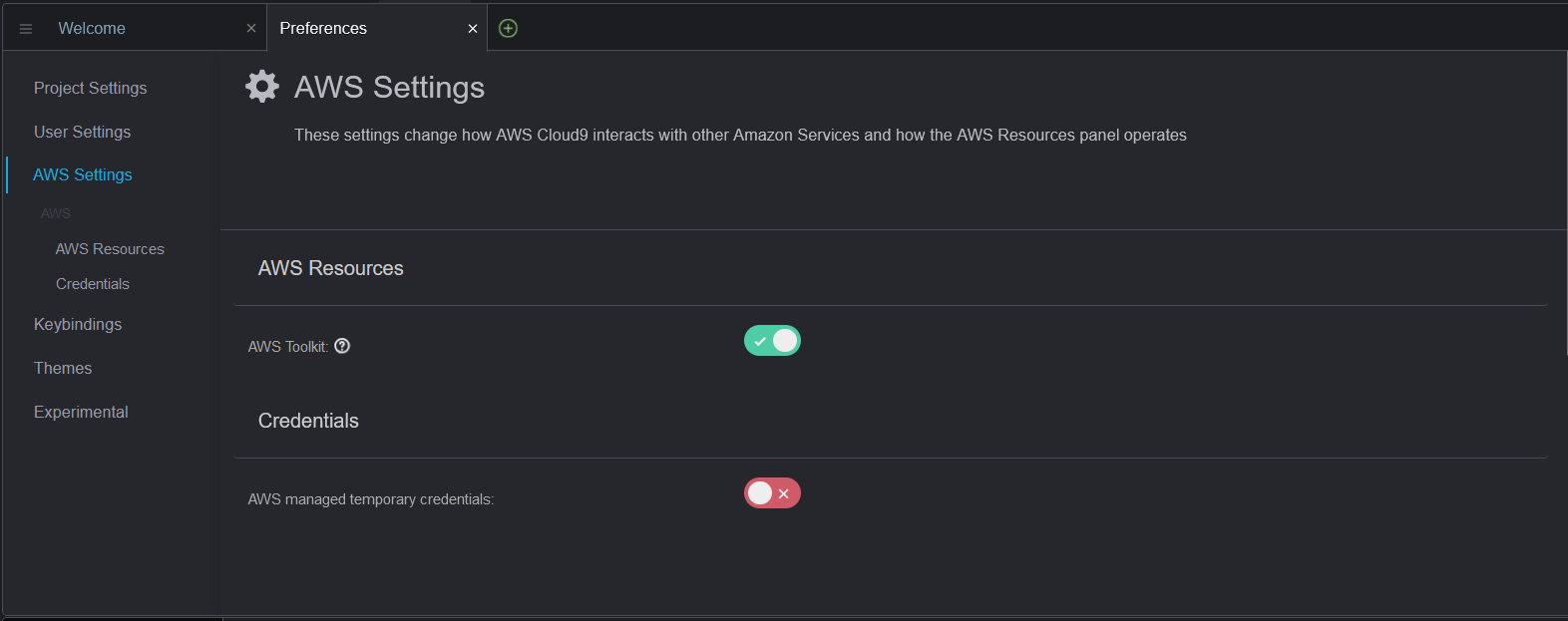
**Install terraform**

****

Create the IAM Role – Edit the trust policy & create an inline policy for the IAM role – privileges for Cloud9

****

For this role to take effect, it is necessary to disable Cloud9's normal management of your credentials.

****

Attach the role to cloud9 environment ec2 instance

**A screenshot of a computer

Description automatically generated**

**Implement IPAM**

This makes it easier to plan, track, and monitor IP addresses for AWS workloads.

I will start creating the architecture by its individual components. The sequence of deployments are as follows:

1. Deploy IPAM
2. Deploy IPAM Root Pool
3. Deploy IPAM Child Pools
4. Deploy a VPC in the us-east-1 region
5. Deploy a VPC in the us-east-2 region

**A screenshot of a computer

Description automatically generated**

**Create the IPAM resource**

Use Terraform to create AWS resources

A black rectangular object with a black line

Description automatically generated

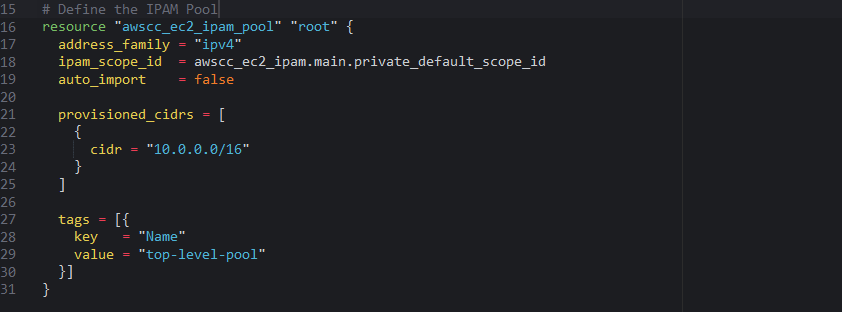
A screenshot of a computer

Description automatically generated

A screenshot of a computer

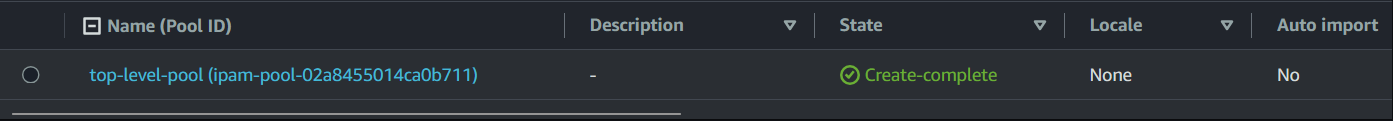
Description automatically generated

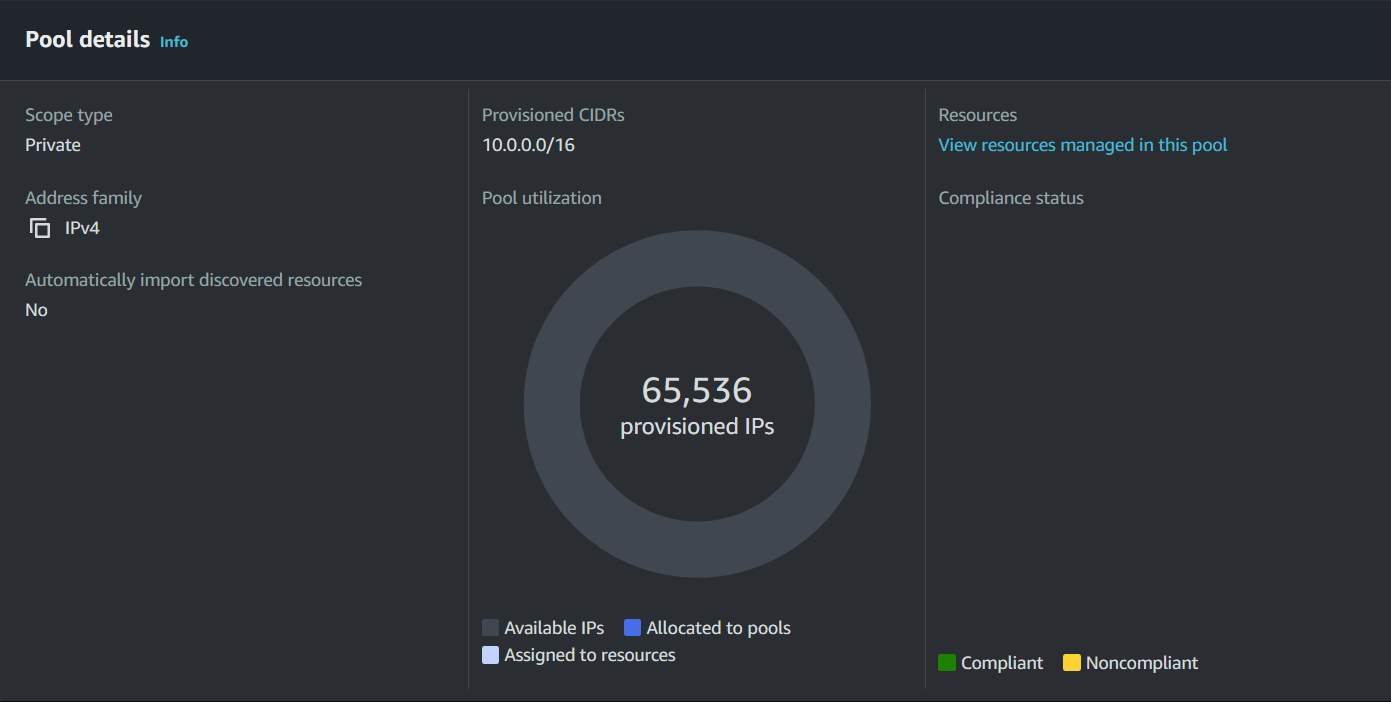
**Create the Root IPAM Pool resource**

****

**Create the IPAM Pool**

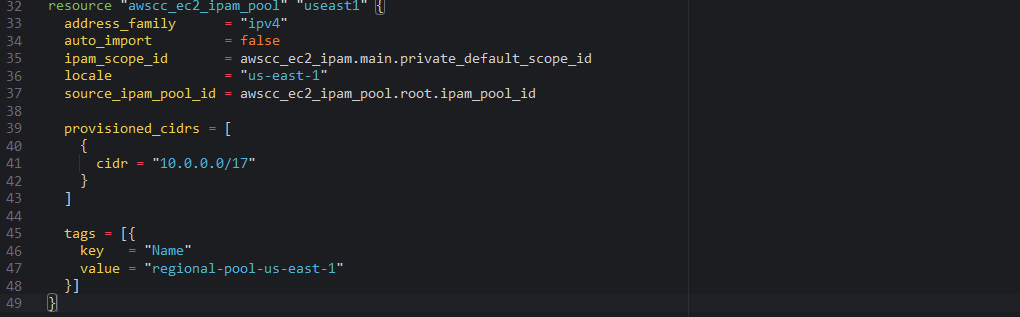
This pool references an attribute from the IPAM resource

****

****

**Provisioning two child pools that will manage CIDRS in us-east-1 & us-east-2 regions.**

Appending resource content to the main.tf file us-east-1

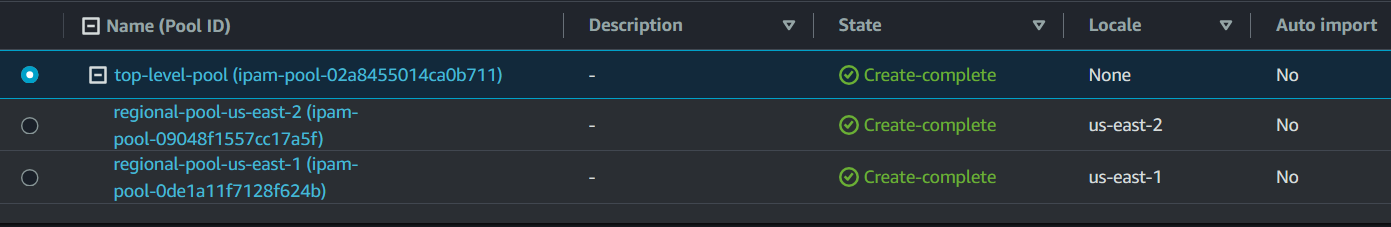


Appending resource content to the main.tf file us-east-1

A screen shot of a computer

Description automatically generated

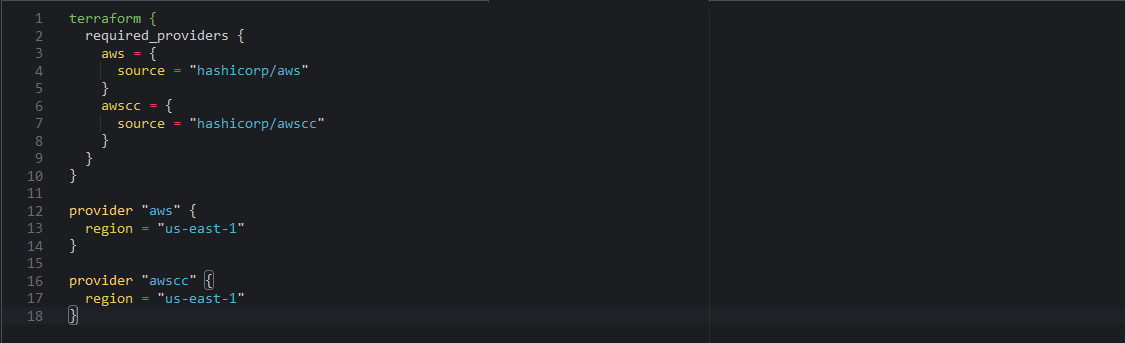
**Pools**

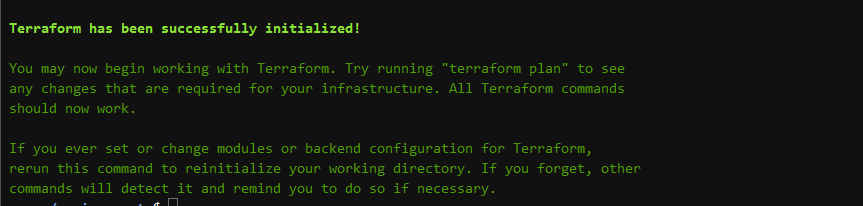
****

**Using multiple providers to reference resources between providers**

I will use the aws provider to create VPCs. Both VPCs will retrieve their CIDR allocation from the respective region's IPAM pool.

Appending additional content to the provider.tf file





**Create a VPC using the AWS provider**

In this section I will create a vpc resource in us-east-1 using the AWS provider and assign CIDR ranges to it using the IPAM Pools.

This will read attributes from resources created in a different provider in a new resource definition.

**Define the VPC Resource**

Appending vpc content to the main.tf file **[us-east-1]**

set the ipv4\_ipam\_pool\_id attribute to that of the IPAM pool awscc\_ec2\_ipam\_pool.useast1.id [created earlier]

A computer screen shot of a computer screen

Description automatically generated

**A screenshot of a computer

Description automatically generated**

**Adding provider configurations**

Appending content to the provider.tf file **[us-east-2**]

A black rectangular object with a white border

Description automatically generated

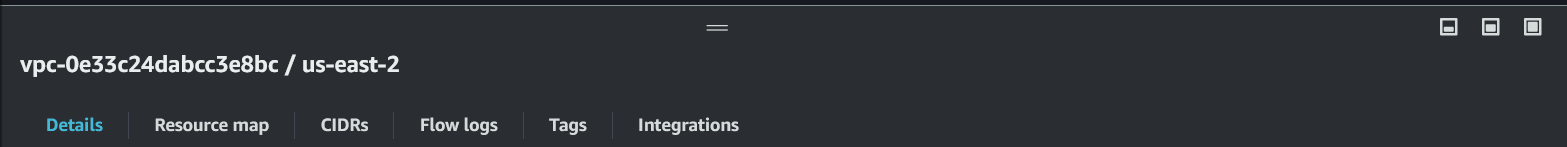
**Define the VPC Resource**

Appending vpc content to the main.tf file **[us-east-2]**

set the ipv4\_ipam\_pool\_id attribute to that of the IPAM pool awscc\_ec2\_ipam\_pool.useast2.id [created earlier]

A screenshot of a computer

Description automatically generated

****

**Shared Tags for Providers FYI**

You can assign metadata to your AWS resources using tags.

Tags can help you manage, identify, organize, search for, and filter resources. It is common practice to have a defined set of standard or required tags that are applied to all resources.

**Provider differences**

* The AWS and awscc providers have different interfaces for defining tags.
* The awscc provider supports the tags argument, which is an attribute list of tag specifications
* The AWS provider supports the tags argument, which is a map of tags to apply to the resource.

**Done!**