**What terraform graph Does**

* **Dependency Visualization**  
  terraform graph outputs a DOT language representation of the resource dependency graph. Nodes represent resources, variables, and data sources; edges show the “depends on” relationships.
* **Why It’s Useful**
  + Quickly understand complex inter-resource dependencies
  + Spot unintended implicit dependencies
  + Communicate architecture in diagrams

**2. Prerequisites**

1. **Terraform CLI** installed (v0.12+)
2. **Graphviz** (for rendering the DOT output into PNG/SVG/etc.)

**# Windows Installer method**:  
Go to https://graphviz.org/download/ and grab the Windows installer.  
Run it and leave the “Add Graphviz to the system PATH” option checked.

 **Chocolatey** (if you have it):

choco install graphviz

 **WinGet** (Windows 10/11):

winget install Graphviz.Graphviz

dot -V

# macOS (Homebrew)

brew install graphviz

# Ubuntu/Debian

sudo apt-get update && sudo apt-get install graphviz

Create Key on Linux:

ssh-keygen -t rsa

**3. Sample AWS EC2 Configuration**

# main.tf

provider "aws" {

region = "us-east-1"

}

resource "aws\_key\_pair" "kp" {

key\_name = "example-key"

public\_key = file("~/.ssh/id\_rsa.pub")

}

resource "aws\_security\_group" "sg" {

name = "allow\_ssh"

description = "Allow SSH inbound"

ingress {

from\_port = 22

to\_port = 22

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

}

resource "aws\_instance" "web" {

ami = "ami-0c94855ba95c71c99"

instance\_type = "t2.micro"

key\_name = aws\_key\_pair.kp.key\_name

vpc\_security\_group\_ids = [aws\_security\_group.sg.id]

root\_block\_device {

volume\_size = 8

delete\_on\_termination = true

}

tags = {

Name = "example-web"

}

}

**4. Generating the Graph**

1. **Initialize**

terraform init

1. **Generate DOT Output**

terraform graph > graph.dot

1. **Render an Image**

**Linux/Mac**

dot -Tpng graph.dot -o graph.png

WIndows

dot -Tpng .\graph.dot -o .\graph.png

This produces graph.png, where you’ll see nodes for aws\_key\_pair.kp, aws\_security\_group.sg, and aws\_instance.web, with arrows showing how web depends on kp and sg.

**5. Interpreting the Graph**

* Each **ellipse** (or box) is a resource/data/variable.
* **Arrows** point **from dependency → dependent**.  
  e.g.

aws\_key\_pair.kp -> aws\_instance.web

aws\_security\_group.sg -> aws\_instance.web

* **Filtering**
  + To focus on top‐level modules only:

terraform graph -module-depth=1 > graph.dot

* + To highlight cycles (rare in Terraform):

terraform graph -draw-cycles > graph.dot

**6. Tips & Tricks**

* **Interactive Visualization**  
  Load the DOT file into an online viewer like [Viz.js](https://viz-js.com/) for quick inspection without installing Graphviz.
* **Embedding in Docs**  
  After generating a PNG/SVG, embed it in your architecture docs or README to keep diagrams in sync with your code.
* **Automate in CI**  
  Add a step in your CI pipeline to regenerate and publish documentation diagrams whenever your Terraform code changes.

With this approach, you’ll gain clear, up-to-date visuals of your AWS EC2 (and any other) Terraform configuration’s dependency graph—making complex infrastructure much easier to reason about.