What are Terraform graphs?

The Terraform graph command generates a visual representation of the dependency relationships between resources in your Terraform configuration or execution plan, helping you to understand the structure and dependencies within your infrastructure. These graphs can also be useful for identifying potential issues, planning changes, and debugging. Each resource is represented as a node, and the arrows between nodes represent the dependencies between resources.

Graphs are generated in the DOT output format, a text-based graph description language, which can be processed by GraphViz software to create visual diagrams (e.g., PNG, SVG).

The graph command does not create or apply any infrastructure changes; it's purely for visualization. For large or complex graphs, breaking them down into smaller sections or using interactive visualizations (see later section on other available tools) can improve readability.

How to use the Terraform graph command?

To generate a Terraform graph type:

```
terraform graph [options]
```

The options available with this command:

- -type= — Specify the type of graph to generate, which can be set to one of the following:
- plan — Graph based on the current configuration.
- plan-refresh-only — Graph-based on a refresh plan only.
- plan-destroy — Graph-based on a plan for destroying resources.
- apply — Graph-based on a saved execution plan.
- -draw-cycles — Include cycles in the graph with colored edges (useful for identifying potential issues).
- -plan=tfplan — Render a graph based on specified plan file instead of configuration files in the current working directory.

Terraform graph examples

My Terraform configuration file main.tf contains the following to create an nginx docker image:

```
terraform {
 required_providers {
  docker = {
   source  = "kreuzwerker/docker"
   version = "2.23.1"
  }
 }
}
```

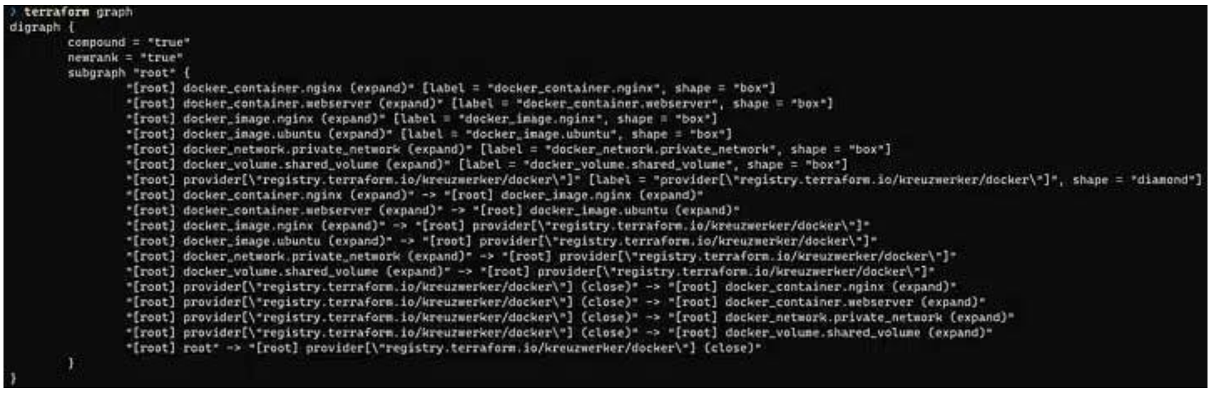
```
provider "docker" {
  host = "tcp://localhost:2375"
}

resource "docker_image" "ubuntu" {
  name = "ubuntu:latest"
}
resource "docker_container" "webserver" {
  image             = docker_image.ubuntu.latest
  name              = "terraform-docker-test"
  must_run          = true
  publish_all_ports = true
  command = [
    "tail",
    "-f",
    "/dev/null"
  ]
}
resource "docker_image" "nginx" {
  name         = "nginx:latest"
  keep_locally = false
}
resource "docker_container" "nginx" {
  image = docker_image.nginx.latest
  name  = "nginx-test"
  ports {
    internal = 80
    external = 8000
  }
}
resource "docker_network" "private_network" {
  name = "my_network"
}
resource "docker_volume" "shared_volume" {
  name = "shared_volume"
}
```

To visualize the dependencies, I simply run:

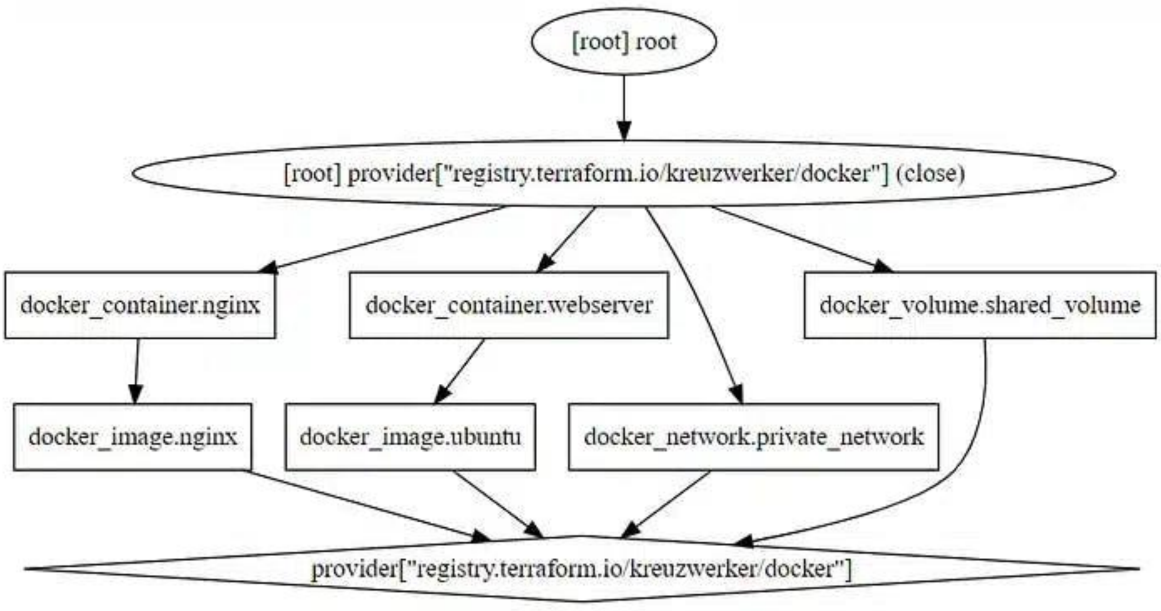
```
terraform graph
```

This provides me with the code in DOT output format:



```
> terraform graph
digraph {
        compound = "true"
        newrank = "true"
        subgraph "root" {
                "[root] docker_container.nginx (expand)" [label = "docker_container.nginx", shape = "box"]
                "[root] docker_container.webserver (expand)" [label = "docker_container.webserver", shape = "box"]
                "[root] docker_image.nginx (expand)" [label = "docker_image.nginx", shape = "box"]
                "[root] docker_image.ubuntu (expand)" [label = "docker_image.ubuntu", shape = "box"]
                "[root] docker_network.private_network (expand)" [label = "docker_network.private_network", shape = "box"]
                "[root] docker_volume.shared_volume (expand)" [label = "docker_volume.shared_volume", shape = "box"]
                "[root] provider[\"registry.terraform.io/kreuzwerker/docker\"]" [label = "provider[\"registry.terraform.io/kreuzwerker/docker\"]", shape = "diamond"]
                "[root] docker_container.nginx (expand)" -> "[root] docker_image.nginx (expand)"
                "[root] docker_container.webserver (expand)" -> "[root] docker_image.ubuntu (expand)"
                "[root] docker_image.nginx (expand)" -> "[root] provider[\"registry.terraform.io/kreuzwerker/docker\"]"
                "[root] docker_image.ubuntu (expand)" -> "[root] provider[\"registry.terraform.io/kreuzwerker/docker\"]"
                "[root] docker_network.private_network (expand)" -> "[root] provider[\"registry.terraform.io/kreuzwerker/docker\"]"
                "[root] docker_volume.shared_volume (expand)" -> "[root] provider[\"registry.terraform.io/kreuzwerker/docker\"]"
                "[root] provider[\"registry.terraform.io/kreuzwerker/docker\"] (close)" -> "[root] docker_container.nginx (expand)"
                "[root] provider[\"registry.terraform.io/kreuzwerker/docker\"] (close)" -> "[root] docker_container.webserver (expand)"
                "[root] provider[\"registry.terraform.io/kreuzwerker/docker\"] (close)" -> "[root] docker_network.private_network (expand)"
                "[root] provider[\"registry.terraform.io/kreuzwerker/docker\"] (close)" -> "[root] docker_volume.shared_volume (expand)"
                "[root] root" -> "[root] provider[\"registry.terraform.io/kreuzwerker/docker\"] (close)"
        }
}
```

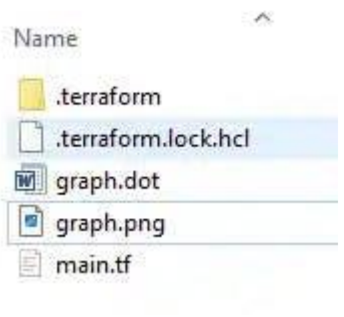
I paste this code into the online graph viewer Webgraphviz, which generates my visual representation!


[root] root
[root] provider["registry.terraform.io/kreuzwerker/docker"] (close)
docker_container.nginx
docker_container.webserver
docker_volume.shared_volume
docker_image.nginx
docker_image.ubuntu
docker_network.private_network
provider["registry.terraform.io/kreuzwerker/docker"]

If you have installed Graphviz, you can also generate the dot output file directly on the command line and then use the dot command to generate the PNG file:

```
terraform graph > graph.dot


dot -Tpng graph.dot -o graph.png
```


Name
.terraform
.terraform.lock.hcl
graph.dot
graph.png
main.tf

Other Terraform visualization tools

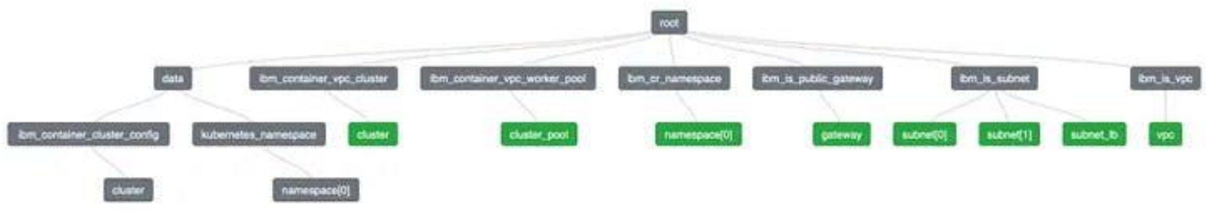
Visualizing complex graphs can be challenging, for more interactive or alternative visualizations, you can consider tools like Blast Radius, Inframap, Rover, or Terraform Visual. These tools offer different features.

1. Terraform Visual

Terraform Visual is a free and open-source tool specifically designed to visualize your Terraform plan interactively. To try it out, simply generate your plan file using the commands below and upload it here.

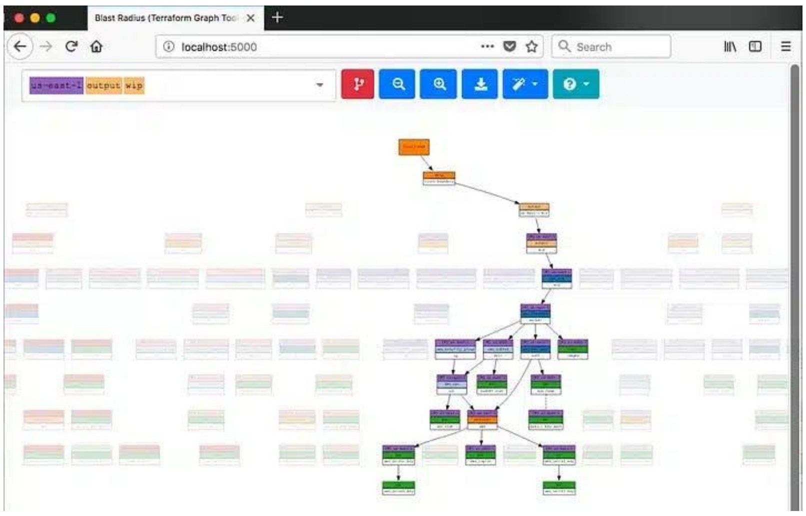
```
$ terraform plan -out=plan.out

$ terraform show -json plan.out > plan.json
```



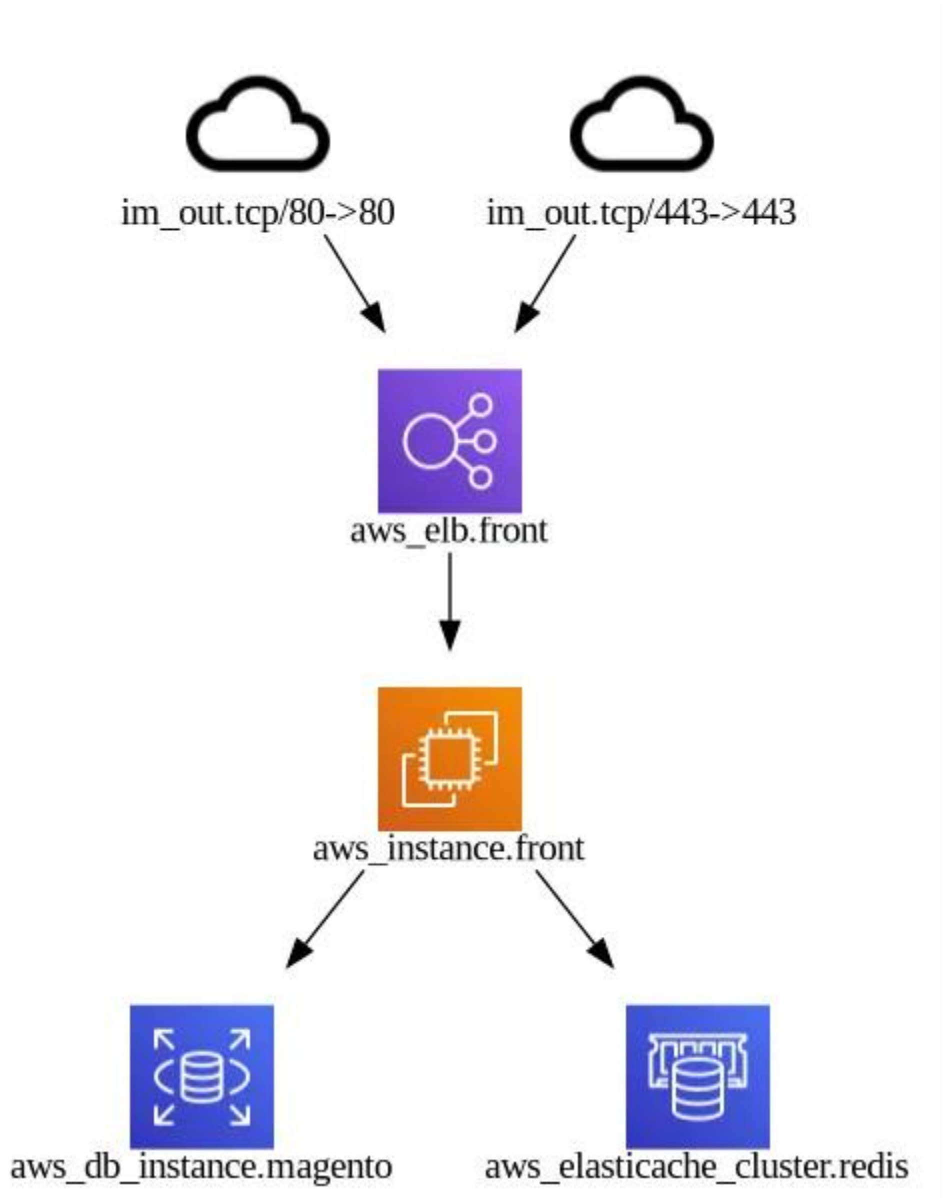
2. Blast radius

Blast radius is another great free option for Terraform visualization.



3. Inframap

Inframap reads your tfstate or HCL to generate a graph specific for each provider, showing only the resources that are most important/relevant.



4. WebGraphviz and GraphvizOnline

Lastly, if you cannot make changes to your local system, online services like WebGraphviz and GraphvizOnline can render DOT files without installing software.