Jack Roper

Follow 13 Followers About

The ‘best’ way to format and structure Terraform code?

 Jack Roper Aug 27, 2020 · 3 min read

Over the past year I’ve been delving deeper into Terraform, using it in anger to deploy resources on Azure and AWS, working on multiple projects. A few weeks into my journey, i was wondering...just how should this stuff be laid out, formatted and structured? There is no clear answer of course as with everything in code and tech, but there are some ‘best practice’ guidelines.

This article doesn’t touch on all best practices around Terraform (e.g. for those familiar with Terraform, multiple workspaces, terraform remote state, using terraform cloud etc.), but focuses on configuration file layout and structure.

Use the ‘terraform fmt’ command

This command when run against your terraform files formats everything into a standard format and style, providing consistency across files and projects. More info here: https://www.terraform.io/docs/commands/fmt.html

Standard file naming and structure across projects and modules

I was involved recently in creating a standard customer landing zone in Azure, a concept in which a standard set of resources could be deployed into an Azure subscription, giving the customer a base to run workloads in Azure. Microsofts Cloud Adoption Framework (CAF) programme has lots more detail on this. Microsoft also provide a docker container set up for Terraform development of landing zones with a base configuration, called 'CAF Rover'. These base configs can them be tweaked to your needs.

The moral of the story here, is that the way Microsoft had laid out their files and modules, and named those files and modules, were consistent and structured, so I took these to be best practice guidelines for future projects I've been working on.

As a minimum, each Terraform project should be split into the following set of files:

main.tf

The starting point for anyone looking at your code. This includes the module calls, and any base resources / local declarations.

provider.tf

Defines the providers used, e.g. Azurerm, and the terraform remote state backend.

variables.tf

Defines the variables needed

output.tf

Defines all the outputs.

README.md

A clear description of the project!

Splitting projects into the above files makes reading and understanding how the Terraform code works much more palatable than one monolithic file. For creating a few

simple resources this might be overkill, but when projects grow, this is a much more scaleable approach. Local modules should follow the same breakdown.

Use modules!

Use builtin modules from the terraform provider registry wherever possible, rather than writing code for individual resources or writing your own modules.

Using 1 level of nesting is recommended, any more than this makes code harder to follow.

Hashicorp call a flat style of using modules ‘ module composition’, because it takes multiple composable building-block modules and assembles them together to produce a larger system.

Interestingly the Microsoft CAF example mentioned previously in this article included multiple levels of nesting, which certainly makes things harder to unpick when trying to understand how outputs and inputs are being passed around between modules. It looks like this was done because of the architecture of their solution; in that they were taking a building block approach.

Use comments!

Obvs! This will differ between authors, I like to add a description of the purpose of the file at the top as a comment, and inline comments on top of each block of code describing what it is doing and why.

Any more?

Following these 4 guidelines will mean you could has consistent formatting, structure and is easy for someone else to understand if you are working in a team.

Disagree with the guidance above or have any ideas I’ve missed? It would be great to here from you! Thanks in advance and happy Terraforming!

Originally published at https://azurelynot.blogspot.com on August 27, 2020.

Terraform Azure AWS Certification

About Help Legal

Get the Medium app