Open in app

Jack Roper

Follow 13 Followers About

Converting information from a spreadsheet into Terraform formatted variable files...part 1!

Jack Roper Aug 11, 2020 · 3 min read

Recently, I noticed a fellow tech guy in the Welsh Azure user group on LinkedIn had asked a question around how information would be passed from the customer to the engineer which would contain all the information necessary to run a Terraform script successfully.

The consensus seemed to be that the consultant would engage with the customer and fill in a spreadsheet which would basically be a list of variables required for the solution, and then pass this to the engineer to transfer into the Terraform variables file.

This approach had several benefits. It would enable all the information to be gathered in one hit, reduce back and forth, and minimise errors. A standard solution could be deployed, potentially in minutes! However the engineer would have to be vigilant to update the spreadsheet with any new variables when new code was added.

This seemed like a sensible way to operate, and when each variable was coupled with a clear description, (which is basically the comment from the variables file), this would enable even a 'non-techy' sales person to gather the required information from the customer.

Taking this a step further, the question arises 'How can i automate transferring the information from the spreadsheet into the variables file quickly and reliably?' Manual input would introduce error, and could be severely time consuming. Consider a file containing 100's of subnets the customer wanted to use in a given solution, this would be a large piece of work in itself to create the file.

Taking a scripted approach to this, the spreadsheet would be formatted in a standard format, and could be read in using PowerShell, and outputted into Terraform formatted variable blocks using a template.

The CSV file would be a list of network names and network subnet addresses formatted as below:

name,value

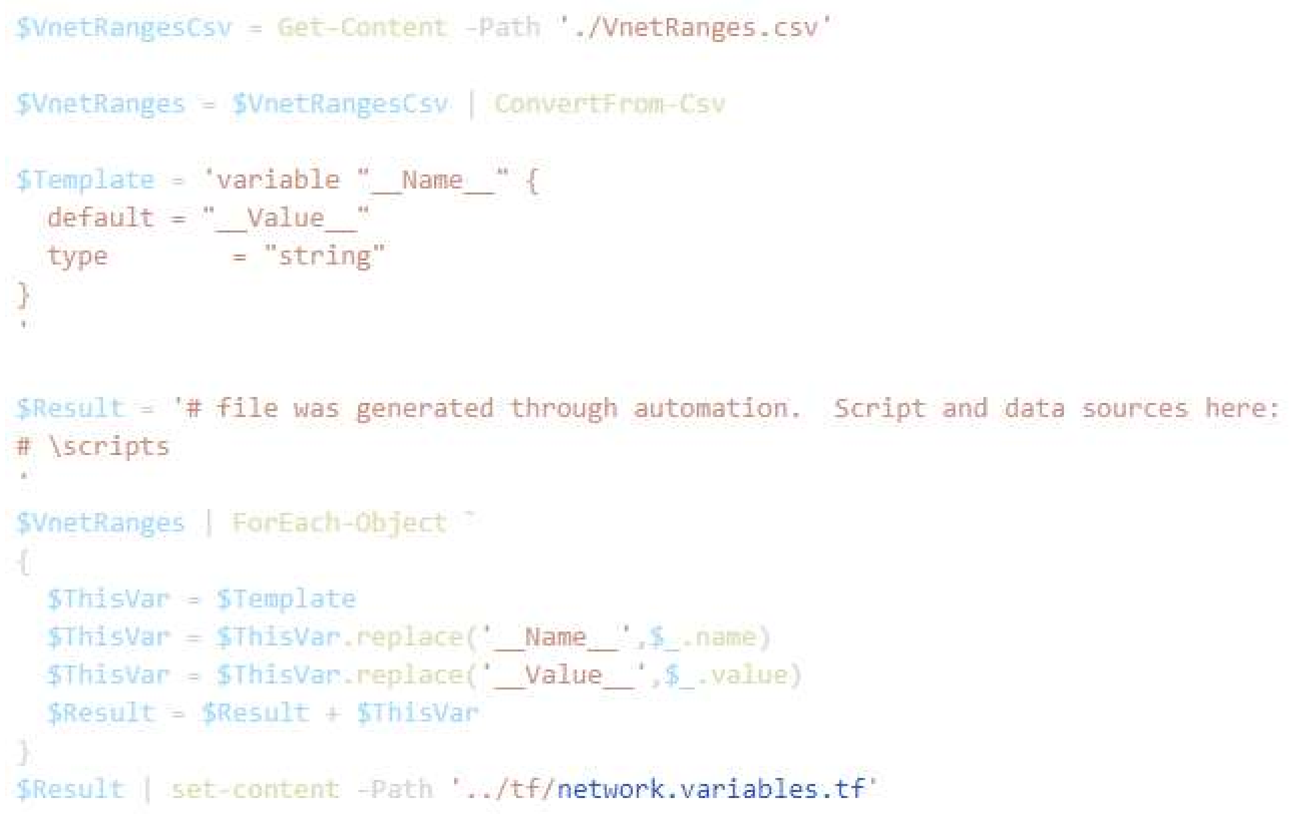
net1,10.21.24.0/21

net2,10.21.24.0/23

net3,10.21.26.0/23

net4,10.21.28.0/23

The PowerShell script takes this file as input and looks like this:



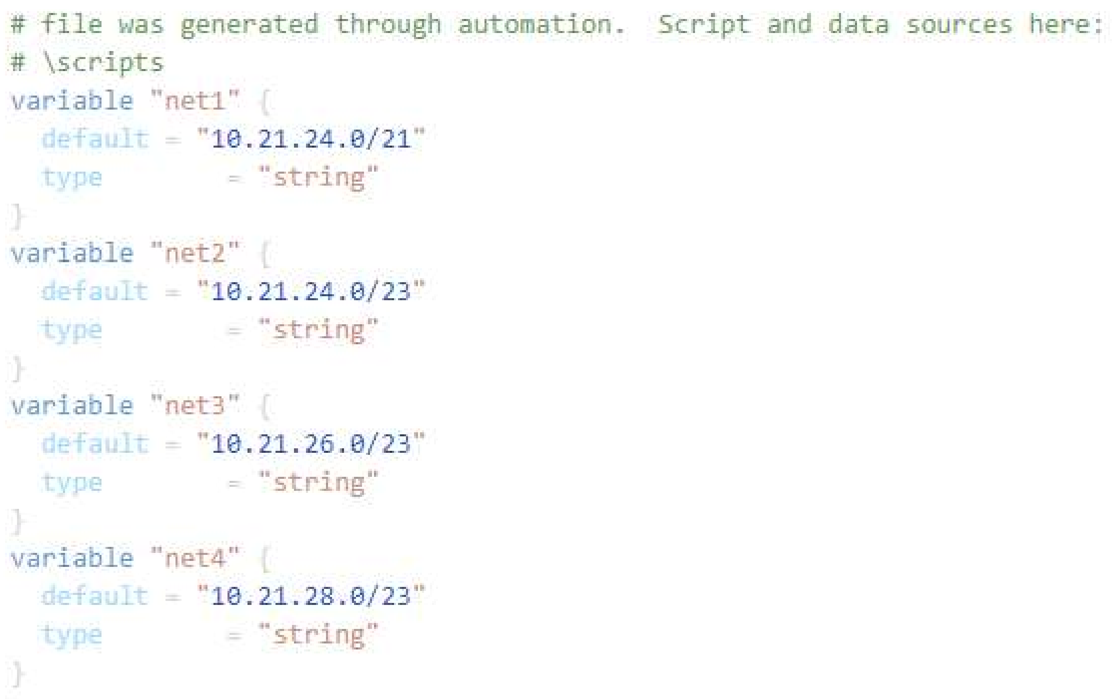
```
$VnetRangesCsv = Get-Content -Path './VnetRanges.csv'

$VnetRanges = $VnetRangesCsv | ConvertFrom-Csv

$Template = 'variable "__Name__" {
  default = "__Value__"
  type        = "string"
}
'

$Result = '# file was generated through automation.  Script and data sources here:
# \scripts
'
$VnetRanges | ForEach-Object `
{
  $ThisVar = $Template
  $ThisVar = $ThisVar.replace('__Name__',$_.name)
  $ThisVar = $ThisVar.replace('__Value__',$_.value)
  $Result = $Result + $ThisVar
}
$Result | set-content -Path '../tf/network.variables.tf'
```

The output file is a file formatted as below, ready to use in Terraform!



```
# file was generated through automation.  Script and data sources here:
# \scripts
variable "net1" {
  default = "10.21.24.0/21"
  type        = "string"
}
variable "net2" {
  default = "10.21.24.0/23"
  type        = "string"
}
variable "net3" {
  default = "10.21.26.0/23"
  type        = "string"
}
variable "net4" {
  default = "10.21.28.0/23"
  type        = "string"
}
```

Taking this a step further, a more complex variable file would not just require variables with the same layout, but multiple types including maps, lists etc. How could this be handled in the same way?

This solution was developed by one of my colleagues using a Python script, which I will run through in a subsequent blog post and hopefully link to the code on Github with his permission in part 2.

As you can see this approach has massive potential for streamlining the workflow and delivery from customer to deployment, eliminating error, increasing speed, and improving results!

Originally published at https://azurelynot.blogspot.com on August 11, 2020.

Powershell Python Terraform Azure