Hi Rakshit,

Thanks for following up. Your answers were very helpful and gave me some good ideas. :) I will outline these ideas below, but please understand that this is **NOT OFFICIAL** best practices, and is **NOT**formal advice.   
  
Here's an approach I was considering:  
- Create a git repo for all your Consul configuration information. This repo would be separate from the terraform code for provisioning the actual clusters. In the repo, you can do several layers of abstractions, based on folder structure, Terraform, and some CI system. My recommendation hinges on the [Consul Terraform Provider](https://www.terraform.io/docs/providers/consul/index.html)  
/

- modules/

   |-> serviceA/

      |-> [main.tf](http://main.tf/)       #This would include the [service definition](https://www.terraform.io/docs/providers/consul/r/service.html) & healthchecks, [intentions](https://www.terraform.io/docs/providers/consul/r/intention.html), and [ACLs](https://www.terraform.io/docs/providers/consul/r/acl_token.html)

      |-> [outputs.tf](http://outputs.tf/)

      |-> [variables.tf](http://variables.tf/)

   |-> shared/

      |-> .....             #Shared modules used across environments

- envs/

   |-> dev/

      |-> [main.tf](http://main.tf/)       #This would be `shared` + `serviceA` module + `serviceB` +...+n

      |-> ...

   |-> uat/

      |-> [main.tf](http://main.tf/)

      |-> ...

   |-> prod/

      |-> ...

This assumes that the actual setting up of your Consul cluster is done separately (helm), and that your CI system has permission to connect to the cluster as a member, or can send tokens with curl requests to the remote cluster. When your modules are deployed correctly in dev, you can then promote the module to higher environments by updating the {env}/[main.tf](http://main.tf/) to include that module and run the deployment. My major call out for this would be that you'd need to capture the tokens generated by your CI system if you were to try manually interacting with the cluster, but [tokens are exposed as part of the provider](https://www.terraform.io/docs/providers/consul/r/acl_token.html).

 A variant on this would be to use some other type of config management, or script a lot of this manually/use a MakeFile/ansible. Assuming "modules" becomes "services", you could use a tool like Make to allow you to create custom tasks that iterate through the folder structure and deploy the needed configurations by reading a manifest file in your {env} folder, and running defined tasks.

One of the things I haven't assessed yet is the implications of using Terraform against k8s/openshift. We recently released a [Terraform Operator](https://github.com/hashicorp/terraform-k8s) to help with provisioning inside of Kubernetes environments. Because this is an alpha feature, a lot is in flux.

Now, I would like to ask for your patience. As this will eventually be a formal blog, it has to go through various levels of review and approval before being released. Again, this is not formal advice until published, and there are likely gotchas/callouts/limits that I haven't considered yet. This work needs to be prioritized, and will take time to implement, test and polish. That said, I hope that this provides a good context for things to consider in implementing your own structure of promoting configs.  
  
If you have any feedback on this initial idea, I'd love to hear it!

Best,

Jono Sosulska

**Digital Developer Advocate || Community**

[**jono@hashicorp.com**](mailto:jono@hashicorp.com)**|| @Jono**