<< To blog overview (http://container-solutions.com/blog/)

Mar 25 2016

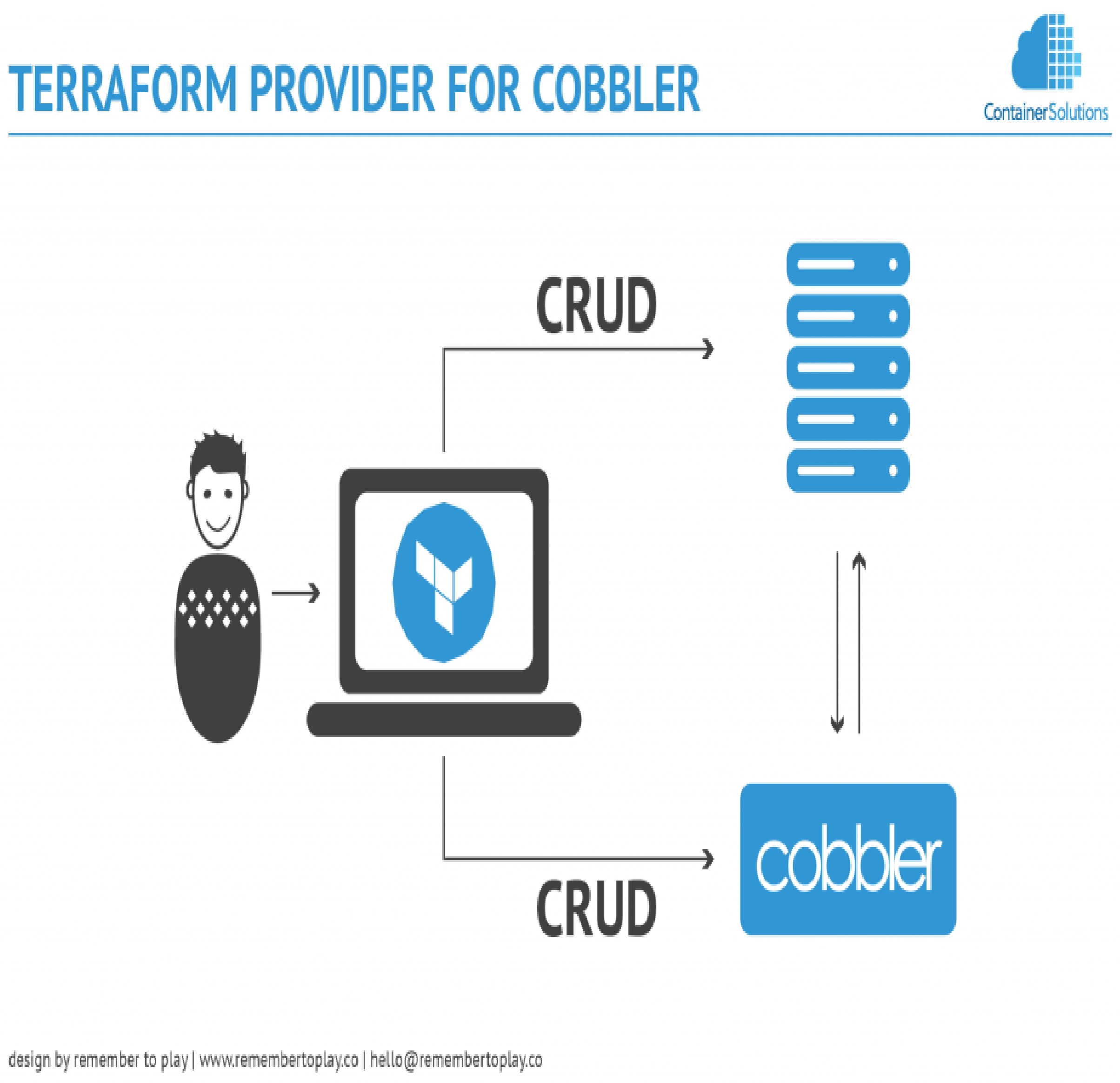
Share:
(https://twitter.com/intent/tweet?text=Terraform provider for Cobbler&url=http://container-solutions.com/terraform-provider-for-cobbler/)
(https://www.facebook.com/sharer/sharer.php?u=http://container-solutions.com/terraform-provider-for-cobbler/)
(mailto:?subject=Check this out&body=This: http://container-solutions.com/terraform-provider-for-cobbler/)

Terraform provider for Cobbler

by Thijs Schnitger (http://container-solutions.com/author/thijs/) 0 (http://container-solutions.com/terraform-provider-for-cobbler#comments) Submit

At Container Solutions we constantly push the boundaries of the tools we work with. While exploring programmable infrastructure we combine the available tools in new ways. Sometimes it works and sometimes it doesn't, but nevertheless we like to contribute things we learned back to the community.

One example of our efforts concerns Terraform (https://www.terraform.io/). We have done extensive research into Terraform and we have been using it with different platforms, some more exotic than others. When we were applying Terraform to provision bare-metal servers, we found that we were lacking a few networking services in the environment where we were provisioning these servers. When a machine starts up in an unconfigured state, it will try to boot using BOOTP. So you need services like DHCP, a PXE server and eventually a DNS server. Cobbler (https://cobbler.github.io/) is a system that bundles these services, so we decided to use that. We quickly found that to handle systems in Cobbler through Terraform, the proper way for Terraform to interface with Cobbler would be through a Provider. No such provider existed yet so we decided to write our own.


TERRAFORM PROVIDER FOR COBBLER
ContainerSolutions
CRUD
CRUD
cobbler
design by remember to play | www.remembertoplay.co | hello@remembertoplay.co

(http://container-solutions.com/content/uploads/2016/03/16.03.24_terraformcobbler-01-1.png)

Structure of a Provider

My mate Carlos (https://twitter.com/mongrelion) already started out on a series detailing how to write a provider (http://container-solutions.com/write-terraform-provider-part-1/) so I won't repeat that stuff here. Instead I'll explain a bit about the specifics of dealing with Cobbler. From Carlos' post it's clear that we need to define a `Provider` with three items: a `Schema` which is a collection of parameters for configuring the Provider, a `ResourcesMap` that lists the resources that will be configurable using this Provider and a `ConfigureFunc` that details how to set up the connection. For each of the resources we wil need a `Schema` as well, plus methods describing how to `Create` , `Read` , `Update` and `Delete` these resources. I could paste some snippets of the code here but you might as well head over to Github to check out the full source code (https://github.com/ContainerSolutions/terraform-provider-cobbler).

Provider

The provider (https://github.com/ContainerSolutions/terraform-provider-cobbler/blob/master/provider.go) is fairly straightforward, it has a `Schema` that contains three strings: a url, a username and a password. For now we have three resources defined in the `ResourcesMap` : a system, a kickstart file and a snippet.
The `ConfigureFunc` for the provider returns an HTTP client that talks to the Cobbler server. We abstracted the code that actually interfaces with Cobbler into a separate client library (https://github.com/ContainerSolutions/cobblerclient), to make the Provider code cleaner. We just pass the values for the url, username and password that we get from the .tf configuration file provided by the user, as arguments to a new `Client` object.

System Resource

Next we define the System Resource (https://github.com/ContainerSolutions/terraform-provider-cobbler/blob/master/resource_cobbler_system.go). This has a slightly more elaborate `Schema` which contains a map of network interfaces. However, most fields in the `Schema` are just strings so it's not really special either. The interesting stuff here is in the `Create` method. Cobbler expects that you perform a series of requests to create a resource (a `System` in this case) and modify it's properties, and then after you're done you send a sync (https://cobbler.github.io/manuals/2.6.0/3/2/2_-_Sync.html) request which commits your changes. This involves updating the configuration files and restarting the various services under Cobblers control, e.g. the DHCP server. The good thing is that this encourages you to set up a bunch of changes and commit them all at once. In fact, if you call the `sync` after each change, Cobbler will try to restart the services repeatedly and it breaks very quickly.
The unfortunate thing is that Terraform doesn't accomodate this behaviour, i.e. there is no way to run a post-execution hook or something. So we decided to call the `sync` in a goroutine. The first `Create` thread will call the goroutine and wait for a signal from the goroutine over a channel. We set a timeout when creating the resource, and perform the `sync` only after that timeout has passed. Each call to the `Create` method will reset the timeout to one second in the future, and so the timeout will pass one second after the last `Create` has been executed. The channel is used to make sure the calling thread will wait on the goroutine, else the Terraform run would end before the `sync` actually had been called. The timeout of one second is a bit arbitrary, but it seems to do the job.

To do

We haven't implemented the `Read` and `Update` methods yet, they just return nil for now because we thought they were not important for our immediate use case. While diving further into Terraform provider internals we came to understand the need for a `Read` method, which is to sync the current state of the infrastructure (Cobbler) with Terraforms internal state as reflected in the state file. So this is definitely on our list of things to implement.

Another thing we need to change is the way we handle logging in to Cobbler. We currently login at the beginning of the `Create` and `Delete` methods, which causes new logins for every resource created or deleted. We should move the login to the `ProviderFunc` , so the login happens only once and the token that is returned will be reused throughout the Terraform run.

Also we currently don't implement the `sync` method after deleting resources, which we really should because the DHCP server also needs to be notified of resources that have been removed.

Other Resources

The code for the kickstart (https://github.com/ContainerSolutions/terraform-provider-cobbler/blob/master/resource_cobbler_kickstart_file.go) and snippet (https://github.com/ContainerSolutions/terraform-provider-cobbler/blob/master/resource_cobbler_snippet.go) resources is quite simple really. They just take a path to a textfile on the local system and post that to Cobbler using the specified name.

Wrapping up

As you can see there's still a lot missing but basically it works. We can create a system in Cobbler using Terraform and have our bare-metal server boot over the network having it's boot image served by Cobbler. The exciting thing is that our efforts did not go unnoticed by the Hashicorp folks, who kindly proposed (https://github.com/ContainerSolutions/terraform-provider-cobbler/issues/1) to merge the code into the Terraform tree. We're currently in the process of getting our PR (https://github.com/hashicorp/terraform/pull/4271) approved. As you can probably guess we're very pleased to get a chance to give something back to the community, which after all is one of Arnold Schwarzenegger's 6 rules of success (https://www.youtube.com/watch?v=EyhOmBPtGNM).

Bio **Latest Posts**

Thijs Schnitger

Senior Engineer at Container Solutions (http://container-solutions.com)

Thijs is a Systems Engineer specialized in all things Linux. At Container Solutions he mainly focuses on programmable infrastructure, when he's not busy lending a hand. His years of experience with development, delivery and operations make him an allround application debugger and problem solver.

(http://twitter.com/thijsschnitger)

(https://nl.linkedin.com/in/thijsschnitger)

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

Post Comment

Related Posts

Write your own Terraform provider: Part 1

(http://container-solutions.com/write-terraform-provider-part-1/)

READ MORE (HTTP://CONTAINER-SOLUTIONS.COM/WRITE-TERRAFORM-PROVIDER-PART-1/)

:tp://container-solutions.com/write-terraform-provider-part-1/)

User Access Manager application on Mantl

(http://container-solutions.com/user-access-manager-application-on-mantl/)

READ MORE (HTTP://CONTAINER-SOLUTIONS.COM/USER-ACCESS-MANAGER-APPLICATION-ON-MANTL/)

:tp://container-solutions.com/user-access-manager-application-on-mantl/)

Cobbler in a Docker Container

(http://container-solutions.com/cobbler-in-a-docker-container/)

READ MORE (HTTP://CONTAINER-SOLUTIONS.COM/COBBLER-IN-A-DOCKER-CONTAINER/)

:tp://container-solutions.com/cobbler-in-a-docker-container/)

:tp://container-solutions.com/docker-swarm-azure-container-services/)

Docker Swarm with Azure Container Services

(http://container-solutions.com/docker-swarm-azure-container-services/)

READ MORE (HTTP://CONTAINER-SOLUTIONS.COM/DOCKER-SWARM-AZURE-CONTAINER-SERVICES/)

:tp://container-solutions.com/terraforming-a-nomad-cluster/)

Terraforming a Nomad cluster

(http://container-solutions.com/terraforming-a-nomad-cluster/)

READ MORE (HTTP://CONTAINER-SOLUTIONS.COM/TERRAFORMING-A-NOMAD-CLUSTER/)

:tp://container-solutions.com/we-are-cohosting-the-gopher-gala-2016/)