



MORPHEUS

Making Terraform Better with the Morpheus CMP

Integration Overview with
HashiCorp Terraform

Last Updated: March 2020

Copyright © 2020 Morpheus Data, LLC. All Rights Reserved
All third-party product and company names are property of their
respective holders and use does not imply any specific endorsement

TABLE OF CONTENTS

INTRODUCTION.....	3
INFRASTRUCTURE AS CODE (IaC) USING TERRAFORM.....	3
WHAT ARE SOME OF THE GAPS WITH TERRAFORM?.....	4
BRITTLE IN PRIVATE CLOUD ENVIRONMENTS.....	4
MISSING ROLE-BASED ACCESS CONTROLS.....	4
POOR INTEGRATION WITH CONFIGURATION AND ITSM TOOLS.....	4
FOCUSED ON INFRASTRUCTURE, NOT APPS.....	5
NO GRAPHICAL USER INTERFACE (GUI).....	5
MORPHEUS AND TERRAFROM INTEGRATION TYPES.....	6
1.MORPHEUS BLUEPRINTS.....	6
2.TERRAFORM PROVIDER FOR MORPHEUS.....	7
MORPHEUS AND TERRAFRM ARE BETTER TOGETHER.....	7
ENHANCE YOUR OVERALL SECURITY POSTURE.....	7
MORPHEUS CYPHER SERVICE FOR SECURE SECRET MANAGEMENT.....	8
ENABLE EXECUTION HISTORY AND AUDITING.....	8
HOW TO USE TERRAFORM WITH MORPHEUS.....	9
TERRAFORM INSTALLATION.....	9
GITHUB/GIT REPOSITORY SETUP.....	10
CREATING MORPHEUS BLUEPRINTS.....	10
THE TERRAFORM PROVIDER.....	12
SUMMARY.....	12

INTRODUCTION

Independently, Morpheus and Terraform can each provide benefits that help increase efficiency and simplify automation across cloud environments. However, leveraging both Terraform and Morpheus together will inherently provide more value by combining the infrastructure-as-code capabilities of Terraform with the multi-cloud governance and policy engine within Morpheus. In this document, we will cover the additional benefits one would gain by leveraging Morpheus and Terraform in combination. We'll touch on the ability to trigger Terraform files from Morpheus, and how to leverage the Terraform Provider for Morpheus. Before we jump in let's spend a few minutes on the pros and cons of Terraform itself.

INFRASTRUCTURE AS CODE (IaC) USING TERRAFORM

Infrastructure as Code is the process of managing and provisioning infrastructure by leveraging simple to read and write definition files, (rather than physical hardware configuration or interactive configuration tools). The value of using Infrastructure-as-Code includes speed of execution and automated delivery of infrastructure as well as a reduction of risk associated with human error that can in turn lead to downtime. IaC can also be a key enabling best practice in DevOps because Developers become more involved in defining configuration and Ops teams get involved earlier in the development process. Another enabler of DevOps is use of IaC to treat infrastructure resources of immutable so you can quickly create and destroy then re-create a fresh instance as needed.

In the case of Terraform, these definition files let you declaratively define the underlying infrastructure in template files (.tf) and manage the infrastructure's state. Other IaC languages include AWS CloudFormation as well as Azure Resource Manager (ARM) templates.

Terraform is cloud-agnostic and uses HashiCorp Configuration Language (HCL) as its IaC templating language. Terraform has a lot of appeal from software developers and IT infrastructure administrators because creating templates can be done quickly and with relative ease. There is a broad open source community for Terraform plus engagement from vendors managing a network of Terraform connectors for different infrastructure endpoints (VMware, AWS, Azure, etc.).

Terraform can be used to manage the state of infrastructure created with Terraform but unfortunately it will not detect changes in a virtual machine that have occurred as a result of installing applications locally or using a configuration management tool like Chef or Ansible or manual changes that happen once infrastructure is in use.

To sum it up, here are just some of the top reasons why developers and IT Infrastructure administrators like Terraform:

- Its perceived simplicity, like coding your infrastructure in English
- It leverages declarative Infrastructure as Code for speed
- It is cloud agnostic with a broad ecosystem
- It's an immutable approach to infrastructure

WHAT ARE SOME OF THE GAPS WITH TERRAFORM?

BRITTLE IN PRIVATE CLOUD ENVIRONMENTS: Terraform is simple when you want to provision infrastructure using one provider in the public cloud, let's say to AWS for example. Write a Terraform file to deploy as many resources as you would like, and quickly deploy the declared infrastructure. But what happens when you need to interact with multiple providers as you would in a private cloud?

For example, one provider to deploy a server, another to go grab and IP from an IP Address Management solution (maybe Infoblox), and another to provision a load balancer (maybe F5). This scenario requires you to call multiple Terraform providers and in-turn maintain multiple individual .tf files. What if later you choose to leverage a different IPAM solution, or a different load balancer technology. With this change you have to leverage different Terraform Providers and your previous file(s) are no longer valid. This added complexity and re-work of multiple .tf files introduces fragility and risk. Later we will discuss how the Terraform Provider for Morpheus helps simplify infrastructure and application deployments in private clouds.

MISSING ROLE-BASED ACCESS CONTROL (RBAC): Control and Identity Access Management is vital to properly scoping and governing the activities of self-service consumers, especially in large Enterprise or Service Provider environments. RBAC is available in the paid version of Terraform (Terraform Enterprise and Sentinel). In Morpheus, RBAC is core to the design of the platform and provides a rich set of granular and multi-tenancy access control capabilities.

POOR INTEGRATION WITH CONFIG. MANAGEMENT AND ITSM TOOLS

Terraform does a good job at tracking infrastructure state. But what about application and environment state? Terraform lacks integrations into CM and ITSM solutions, and therefore doesn't address tracking application changes, nor does it integrate into organizations ITIL processes to track and maintain CMDB records, deal with approvals, etc.

Morpheus integrates into all of the popular configuration management tools (Ansible, Puppet, Chef, and SaltStack) and can even enable the well governed mixing and matching of automation task types. Morpheus also integrates deeply into ITSM tools (ServiceNow, Cherwell, and Remedy). This allows Morpheus to be part of the ITIL processes for approvals and CMDB updates, and hence involved in the overall GitOps workflow.

FOCUSED ON INFRASTRUCTURE, NOT APPS: As noted earlier, Terraform is good at deploying many infrastructure resources to a single cloud and managing the infrastructure state, but what about applications? HashiCorp has embraced the notion of “Immutable Infrastructure” but at the end of the day it’s all about the application. If something needs to be updated or fixed, new servers built from a common image (where the updated software is baked in) are provisioned to replace the old ones.

This process can be functional for smaller teams or projects. However, at the Enterprise level a pre-baked application image approach can fall apart. The high number of images and software versions that need to be managed and maintained can become an operations team’s nightmare.

On the other hand, Morpheus defines the application(s) to be provisioned at the logical level of the service provisioning process. All the way down to policy-based automation workflows that need to occur in the provisioning of an environment. This enables Immutable Application Stacks... not just infrastructure.

NO GRAPHICAL USER INTERFACE (GUI): Terraform is effectively a command-line tool. There are different tools out there that provide a bolt-on UI to Terraform however, if you want to use a HashiCorp UI you have to move to Terraform Enterprise which will provide a graphical user interface at a significant cost premium.

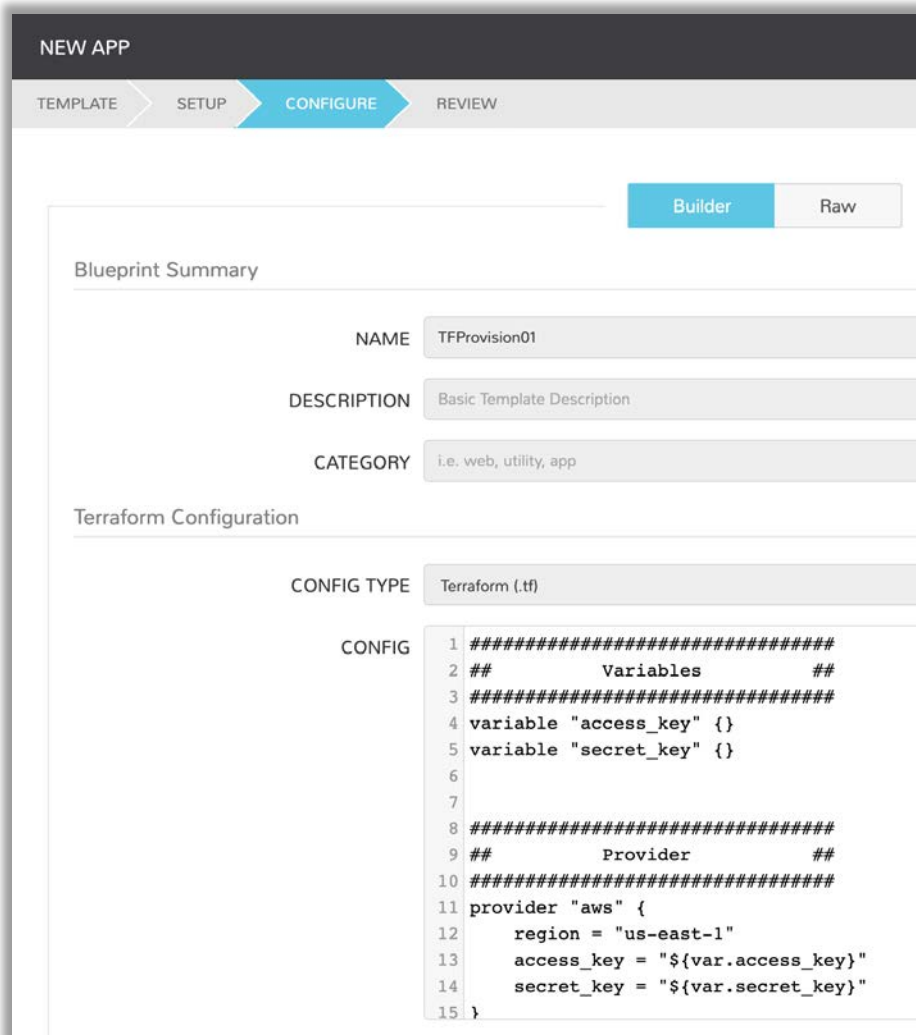
Morpheus is an API first platform built for developers, so it certainly checks all the Dev boxes as far as full fidelity API/CLI. However, it also appeals to IT Operations and other users that may be more comfortable in a feature-rich GUI.

MORPHEUS AND TERRAFROM INTEGRATION TYPES

Presently, Morpheus and Terraform work together in two ways based on who is calling what.

1.Morpheus Blueprints

First you can create blueprints in Morpheus that include Terraform files. Here's a simple example of a Morpheus application blueprint (includes a .tf file) that will deploy infrastructure to AWS. This is the dominant use case for most customers as it enables the operations team to manage all of the surrounding integration points and then expose Terraform templates as application service types. In this model you can leverage Morpheus Cypher to manage secrets/keys.



2. Terraform Provider for Morpheus


The second way to leverage Morpheus which could be more appealing to developers is via the Terraform Provider. The new Terraform provider for the Morpheus Data appliance can be found on GitHub.

<https://github.com/gomorpheus/terraform-provider-morpheus>

There's a distinct advantage to leveraging the Morpheus provider for on premises infrastructure, because it can become a single provider to offload the responsibility of provisioning and orchestrating heterogeneous infrastructure and managing application state. Morpheus integrates out of the box with 80+ tools and 20+ clouds. By writing Terraform files to call Morpheus you can let the CMP do all the heavy lifting and reduce the brittleness of IaC deployments.

Terraform Provider for Morpheus

- Website: <https://www.morpheusdata.com/>
- Docs: [Morpheus Documentation](#)
- Support: [Morpheus Support](#)



MORPHEUS

This is the Terraform provider for the Morpheus data appliance. It interfaces with the [Morpheus API](#) using the `morpheus-go-sdk` client. Like all [Terraform Providers](#), it is written in Go.

This is being developed in conjunction with [morpheus-go-sdk](#).

BETA This library is actively under development and is only available as a prototype. A fully featured version will be available in the near future.

There are a number of other benefits that Morpheus brings to the equation. Let's touch on some of them.

MORPHEUS AND TERRAFORM ARE BETTER TOGETHER

Enhance your overall security posture with Morpheus RBAC

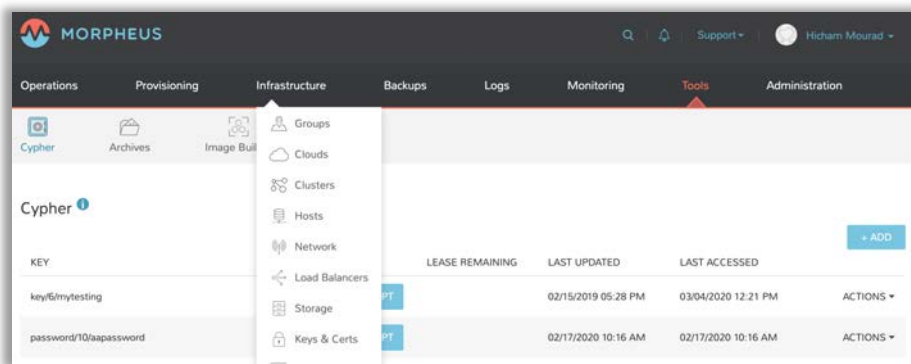
As noted earlier, Terraform is limited when it comes to governance policy and RBAC. If you have access to Terraform, you can plan/apply .tf files but there is no way for IT teams to control access to those .tf files or audit the result of execution. By integrating Terraform with Morpheus, you can take advantage of the fine-grained governance and RBAC available in Morpheus to extend those capabilities when providing access to Terraform files. Morpheus Governance and Control goes well beyond just Terraform. Because Morpheus is a Multi-Cloud Provisioning, Automation, and Orchestration platform, Morpheus takes that governance to Private and Public Clouds alike, as well as any additional integrated technologies,

like configuration management, ITSM, logging, backup systems. You can find all the integrations documented here:

https://docs.morpheusdata.com/en/latest/integration_guides/integration_guides.html#integration-guide

Morpheus Cypher Service for secure secret management

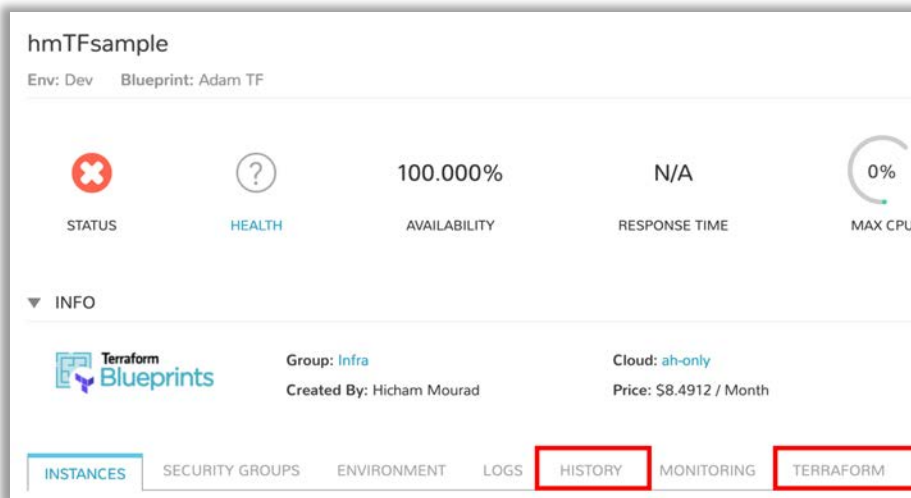
Morpheus has a built-in service called Cypher. It's a secure Key/Value store. Cypher allows the storage of secret data (like credentials or variable sets) in a highly encrypted way for future retrieval. When leveraging Morpheus and Terraform, these variables can be seamlessly passed to Terraform. Secrets and variables only have to be maintained and managed in one place.



Enable Execution History and Auditing of all Terraform actions

From Morpheus whether you provision a Terraform blueprint or make any changes and plan/apply again, the output is available in the History tab of that instance.

That instance will also have a Terraform tab to provide additional details.



The screenshot shows a web interface with a navigation bar at the top containing tabs for INSTANCES, SECURITY GROUPS, ENVIRONMENT, LOGS, MONITORING, and TERRAFORM. The TERRAFORM tab is selected. Below the navigation bar, there are two sections: 'State' and 'Plan'. The 'State' section contains a list of Terraform state entries, each with a line number and a key-value pair. The 'Plan' section contains a list of messages, also with line numbers, indicating that no changes are needed.

```
1 data.vsphere_datacenter.dc:
2   id = datacenter-2
3   name = labs-denver
4 data.vsphere_datastore.datastore:
5   id = datastore-204
6   datacenter_id = datacenter-2
7   name = labs-demo-gnap-240
8 data.vsphere_network.network:
9   id = network-51
10  datacenter_id = datacenter-2
11  name = VM Network
12  type = Network
13 data.vsphere_resource_pool.pool:
14  id = resgroup-158
15  datacenter_id = datacenter-2
16  name = labs-den-demo-cluster/Resources
17 data.vsphere_virtual_machine.template:
18  id = 421f5ca5-8f19-d108-b9f0-c977ae44176b
19  alternate_guest_name =
20  datacenter_id = datacenter-2
21  disks.# = 1
```

```
1
2 No changes. Infrastructure is up-to-date.
3
4 This means that Terraform did not detect any differences between your
5 configuration and real physical resources that exist. As a result, no
6 actions need to be performed.
```

HOW TO USE TERRAFORM WITH MORPHEUS

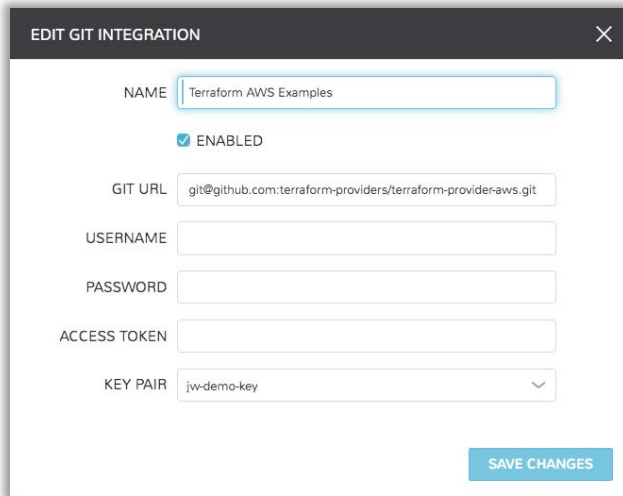
Let's go ahead and discuss the setup required for this integration, and then the interaction between Terraform and Morpheus. As mentioned earlier there are two ways to integrate and leverage Terraform with Morpheus. By creating Application blueprints that leverage Terraform files, and the second is via the Terraform Provider for Morpheus. In order to use Morpheus application blueprints with Terraform there is some initial installation and setup required.

Terraform Installation

The first step in taking advantage of this integration is to install Terraform on the Morpheus appliance. The great news is that Morpheus will automatically install Terraform locally upon the first Terraform application provisioning action. You can also manually install and configure Terraform on the Morpheus appliance if you wish. More details on the manual installation are available here: https://docs.morpheusdata.com/en/latest/integration_guides/Automation/terraform.html

GitHub/Git Repository Setup

To use .tf files from a Git repo, GitHub integration needs to be configured in Administration - Integrations. If one is not configured .tf or .tf.json files can be manually added to the Terraform application blueprints.



EDIT GIT INTEGRATION

NAME

ENABLED

GIT URL

USERNAME

PASSWORD

ACCESS TOKEN

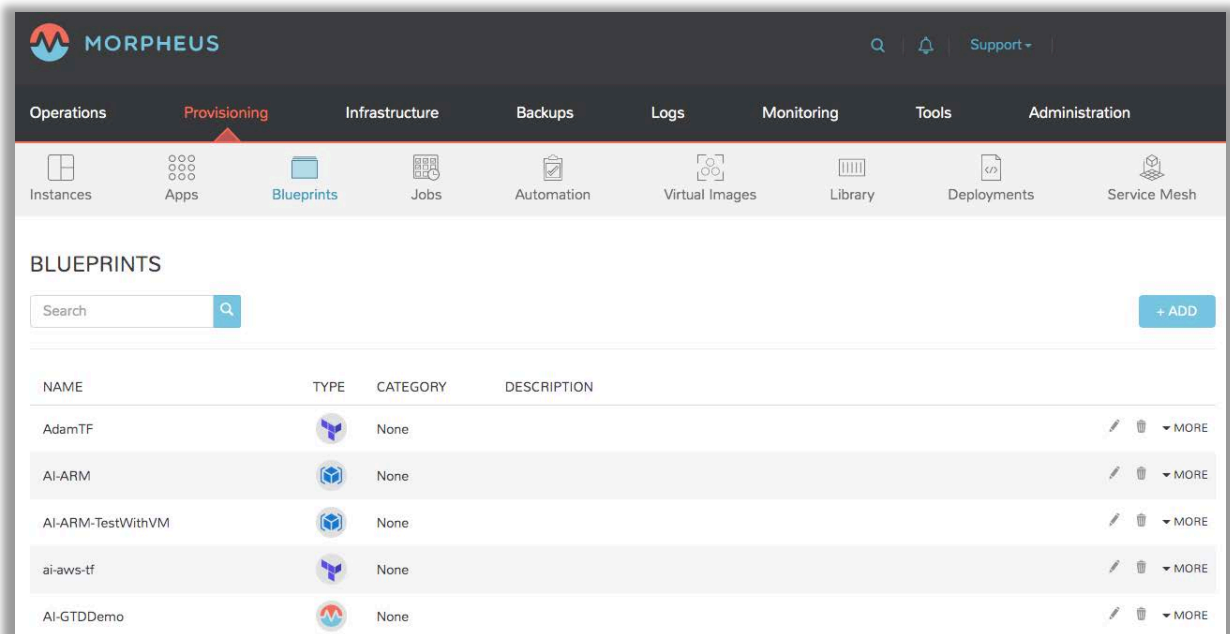
KEY PAIR

[SAVE CHANGES](#)

Now that the setup is complete, we are able to leverage Terraform.

Creating Morpheus Blueprints

Morpheus has a powerful blueprinting engine that uses its own native YAML/JSON syntax or can integrate with third party blueprint frameworks such as AWS CloudFormation, ARM, Kubernetes HELM, and of course Terraform.



MORPHEUS

Operations Provisioning Infrastructure Backups Logs Monitoring Tools Administration

Instances Apps Blueprints Jobs Automation Virtual Images Library Deployments Service Mesh

BLUEPRINTS

Search [+ ADD](#)

NAME	TYPE	CATEGORY	DESCRIPTION
AdamTF		None	/ 🗑 ▼ MORE
AI-ARM		None	/ 🗑 ▼ MORE
AI-ARM-TestWithVM		None	/ 🗑 ▼ MORE
ai-aws-tf		None	/ 🗑 ▼ MORE
AI-GTDDemo		None	/ 🗑 ▼ MORE

Use this engine to create a new application blueprint of type Terraform with either manual .tf entry or a connected Git repository.

The screenshot shows the 'NEW BLUEPRINT' interface. At the top, there are two tabs: 'NAME' (active) and 'CONFIGURE'. Below the tabs is a 'Blueprint Summary' section. The 'NAME' field contains the text 'Blueprint Name'. The 'TYPE' field has a dropdown menu open, showing several options: 'Morpheus', 'Terraform' (which is selected with a checkmark), 'ARM', 'CloudFormation', 'Kubernetes', and 'Helm'.

The screenshot shows the 'EDIT BLUEPRINT' interface. At the top, there are two tabs: 'Builder' (active) and 'Raw'. Below the tabs is a 'Blueprint Summary' section with fields for 'NAME' (ai-aws-tf), 'DESCRIPTION' (Basic Template Description), 'CATEGORY' (i.e. web, utility, app), and 'IMAGE' (with a 'Browse' button). Below this is a 'Terraform Configuration' section. The 'CONFIG TYPE' dropdown is set to 'Terraform (.tf)'. The 'CONFIG' field contains a Terraform configuration snippet:

```
1 #####
2 ##           Variables           ##
3 #####
4 variable "aws_access_key" {}
5 variable "aws_secret_key" {}
6
7
8 provider "aws" {
9     region = "us-west-2"
10    access_key = "${var.aws_access_key}"
11    secret_key = "${var.aws_secret_key}"
12 }
13
14 resource "aws_vpc" "my_vpc" {
15    cidr_block = "10.0.0.0/26"
16 }
```

Below the configuration field are 'TFVAR SECRET' (set to 'Select') and 'OPTIONS' (i.e. -var 'foo=bar') fields.

Once the blueprints are created then they can be consumed via the self-service provisioning GUI, via command line tool, or via the API.

The Terraform Provider

Leverage the Terraform Provider for Morpheus as mentioned earlier in this document. Check back often for more details and updates on the provider.

<https://github.com/gomorpheus/terraform-provider-morpheus>

SUMMARY

We've discussed what Terraform is, and we've touched on the gaps that it has especially in a heterogenous private cloud, and when provisioning multi-cloud infrastructure and applications. We've noted the benefits of using Morpheus and Terraform together, and how Morpheus helps close the Terraform gaps. We've also highlighted how easy it is to setup this integration to take advantage of these benefits. We also discussed the challenges that Terraform has with private cloud, and by far the best way to leverage Terraform to provision to private cloud is by using the Terraform Provider for Morpheus. You only write to one provider, and Morpheus takes care of provisioning and state management of all the private cloud components, solutions, and constructs. Let Morpheus be the arbiter of all things private cloud, and multi-cloud! One final point I'd like to make is that any organization looking to procure Terraform Enterprise can save costs by leveraging Morpheus and open source Terraform.

To get started with Morpheus today, please visit www.morpheusdata.com

Request a demo of Morpheus at www.morpheusdata.com/demo/