Integrating with Network Infrastructure Automation (NIA)

Avi Technical Reference (v20.1)

Integrating with Network Infrastructure Automation (NIA)

view online

Overview

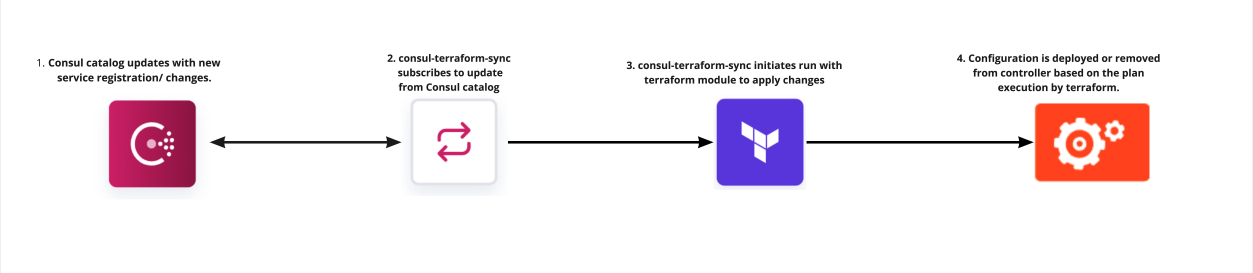
This guide explains the pool module for Network Infrastructure Automation (NIA).

Note: This Terraform module is designed to be used only with consul-terraform-sync.

This module supports scale up and scale down pool and pool members(servers) based on the service(s) configuration in Consul catalog.

Consul-Terraform-Sync

The `consul-terraform-sync` runs as a daemon that enables a publisher-subscriber paradigm between Consul and the Controller to support Network Infrastructure Automation (NIA).



High level architecture diagram of Network Infrastructure Automation (NIA) for VMware NSX Advanced Load Balancer (Avi).

The `consul-terraform-sync` subscribe to updates from the Consul catalog and executes one or more automation tasks with appropriate value of service variables based on those updates. The `consul-terraform-sync` leverages Terraform as the underlying automation tool and utilizes the Terraform provider ecosystem to drive relevant change to the network infrastructure.

Each task consists of a runbook automation written as a compatible Terraform module using resources and data sources for the underlying network infrastructure provider.

You can install consul-terraform-sync using Installing Consul-Terraform-Sync link.

Prerequisites

Requirements

Name	Version
Terraform	>= 0.13
consul-terraform-sync	>= 0.1.0
consul	>= 1.7

Providers

Name	Version
vmware/avi	>= 20.1.4

Usage

In order to use this module, you will need to install consul-terraform-sync, create a task with this Terraform module as a source within the task, and run consul-terraform-sync.

You can subscribe to the services in the consul catalog and define the Terraform module which will be executed when there are any updates to the subscribed services using a task.

Note: It is recommended to have the Consul-Terraform-Sync Configuration Guide link for reference.

1. Download the `consul-terraform-sync` on a node which is highly available (preferably, a node running a consul client)
2. Add `consul-terraform-sync` to the path on that node
3. Check the installation using the following CLI code:

```
wget https://releases.hashicorp.com/consul-terraform-sync/
<consul-terraform-sync-version>
 /consul-terraform-sync_
 <consul-terraform-sync-version>
  _linux_amd64.zip unzip consul-terraform-sync_
  <consul-terraform-sync-version>
   _linux_amd64.zip sudo mv consul-terraform-sync /usr/local/bin consul-terraform-sync --version
  </consul-terraform-sync-version>
 </consul-terraform-sync-version>
</consul-terraform-sync-version>
```

4. Create a config file `tasks.hcl` for `consul-terraform-sync`. For instance,

```
log_level = <log_level> # eg. "info"

consul {
  address = "<consul agent address>" # eg. "1.1.1.1:8500"
}

driver "terraform" {
  log = true
  required_providers {
    avi = {
    source =  "vmware/avi"
    version = "20.1.4"
    }
  }
```

```
}

terraform_provider "avi" {
  avi_username    = "admin"
  avi_tenant      = "admin"
  avi_password    = "{{ env \"avi_password\" }}"
  avi_controller  = "<avi_controller_address>"
  avi_version     = "<api_version>"
  avi_api_timeout = 50
}

task {
  name = <name of the task (has to be unique)> # eg. "avi-pool-counting"
  description = <description of the task> # eg. "Automatically Scale/Configure AVI Pool Servers"
  source = "vmware/modules/nia/pool" # to be updated
  providers = ["avi"]
  services = ["<list of services you want to subscribe to>"] # eg. ["web", "counting"]
  variable_files = ["<list of files that have user variables for this module (please input full path)>"] # eg.
  enabled = true
}
```

5. Start `consul-terraform-sync` using the following CLI code:

```
$ consul-terraform-sync -config-file=tasks.hcl
```

The `consul-terraform-sync` will create pool for each task and subscribed services from consul catalog will become the servers of pool.

The `consul-terraform-sync` is now subscribed to the Consul catalog. Any updates to the services identified in the task will result in updating the servers config of pool on the Avi Controller.

Configuring Task Parameters for Consul Terraform Sync.

For more details on variable for task {} block in task.hcl, refer to Inputs section in README guide.

How does consul-terraform-sync work?

There are 2 aspects of `consul-terraform-sync`, namely, updates from Consul catalog and managing the entire Terraform workflow.

Updates from Consul Catalog

In the backend, `consul-terraform-sync` creates a blocking API query session with the Consul agent identified in the config to get updates from the Consul catalog, named `consul-terraform-sync`.

The `consul-terraform-sync` will get an update for the services in the consul catalog when any of the following service attributes are created, updated or deleted. These updates include service creation and deletion as well.

- Service ID
- Service Name
- Service Address

- Service Port
- Service Meta
- Service Tags
- Service Namespace
- Service Health Status
- Node ID
- Node Address
- Node Datacenter
- Node Tagged Addresses
- Node Meta

Managing the entire Terraform Workflow

If a task and is defined, one or more services are associated with the task, provider is declared in the task and a Terraform module is specified using the source field of the task, the following sequence of events will occur:

- `consul-terraform-sync` will install the required version of Terraform.
- `consul-terraform-sync` will install the required version of the Terraform provider defined in the config file and declared in the task.
- A new directory nia-tasks with a sub-directory corresponding to each task will be created. This is the reason for having strict guidelines around naming. Each sub-directory corresponds to a separate Terraform workspace. Within each sub-directory corresponding a task, `consul-terraform-sync` will template a `main.tf`, `variables.tf`, `terraform.tfvars` and `terraform.tfvars.tmpl`.

main.tf

This files contains declaration for the required Terraform and provider versions based on the task definition.

In addition, this file has the module (identified by the source field in the task) declaration with the input variables Consul K/V is used as the backend state for this Terraform workspace.

Example of generated `main.tf` by `consul-terraform-sync`:

```
# This file is generated by Consul Terraform Sync.
  #
  # The HCL blocks, arguments, variables, and values are derived from the
  # operator configuration for Sync. Any manual changes to this file
  # may not be preserved and could be overwritten by a subsequent update.
  #
  # Task: avi-svc-web
  # Description: Automatically Scale AVI Service Redirection Destinations

  terraform {
    required_version = ">= 0.13.0, < 0.15"
    required_providers {
      avi = {
        source  = "vmware/avi"
        version = "20.1.4"
      }
    }
```

```
    backend "consul" {
      address = "x.x.x.x:8500"
      gzip    = true
      path    = "consul-terraform-sync/terraform"
    }
  }
  provider "avi" {
    avi_api_timeout = var.avi.avi_api_timeout
    avi_controller  = var.avi.avi_controller
    avi_password    = var.avi.avi_password
    avi_tenant      = var.avi.avi_tenant
    avi_username    = var.avi.avi_username
    avi_version     = var.avi.avi_version
  }

  module "avi-svc-web" {
    source   = "vmware/modules/nia/pool"
    services = var.services

    avi_controller = var.avi_controller
    avi_password   = var.avi_password
    avi_username   = var.avi_username
    avi_version    = var.avi_version
    lb_algorithm   = var.lb_algorithm
    pool_name      = var.pool_name
  }
```

variables.tf

This is `variables.tf` file defined in the module.

Example of generated `variables.tf` by `consul-terraform-sync`:

```
# This file is generated by Consul Terraform Sync.
#
# The HCL blocks, arguments, variables, and values are derived from the
# operator configuration for Sync. Any manual changes to this file
# may not be preserved and could be overwritten by a subsequent update.
#
# Task: avi-svc-web
# Description: Automatically Scale AVI Service Redirection Destinations

# Service definition protocol v0
variable "services" {
 description = "Consul services monitored by Consul Terraform Sync"
 type = map(
   object({
     id        = string
     name      = string
     kind      = string
```

```
      address   = string
      port      = number
      meta      = map(string)
      tags      = list(string)
      namespace = string
      status    = string

      node                  = string
      node_id               = string
      node_address          = string
      node_datacenter       = string
      node_tagged_addresses = map(string)
      node_meta             = map(string)

      cts_user_defined_meta = map(string)
    })
 )
}
variable "avi" {
 default     = null
 description = "Configuration object for avi"
 sensitive   = true
 type = object({
   alias           = string
   avi_api_timeout = number
   avi_controller  = string
   avi_password    = string
   avi_tenant      = string
   avi_username    = string
   avi_version     = string
 })
}
```

Example of generated `variables.module.tf` by `consul-terraform-sync`:

```
# This file is generated by Consul Terraform Sync.
#
# The HCL blocks, arguments, variables, and values are derived from the
# operator configuration for Sync. Any manual changes to this file
# may not be preserved and could be overwritten by a subsequent update.
#
# Task: avi-svc-web
# Description: Automatically Scale AVI Service Redirection Destinations

variable "avi_controller" {
  default = null
  type    = string
}
```



```
variable "avi_password" {
  default = null
  type    = string
}

variable "avi_username" {
  default = null
  type    = string
}

variable "avi_version" {
  default = null
  type    = string
}

variable "lb_algorithm" {
  default = null
  type    = string
}

variable "pool_name" {
  default = null
  type    = string
}
```

terraform.tfvars

This is the most important file generated by `consul-terraform-sync`.

This variables file is generated with the most updated values from Consul catalog for all the services identified in the task.

`consul-terraform-sync` updates this file with the latest values when the corresponding service gets updated in Consul catalog.

Example of generated `terraform.tfvars` by consul-terraform-sync:

```
# This file is generated by Consul Terraform Sync.
#
# The HCL blocks, arguments, variables, and values are derived from the
# operator configuration for Sync. Any manual changes to this file
# may not be preserved and could be overwritten by a subsequent update.
#
# Task: avi-svc-web
# Description: Automatically Scale AVI Service Redirection Destinations

services = {
 "web.avi-dev.dc1" : {
   id      = "web"
   name    = "web"
   kind    = ""
```

```
    address = "x.x.x.x"
    port    = 80
    meta = {
      enabled = "false"
      ratio   = "5"
    }
    tags            = ["rails"]
    namespace       = null
    status          = "passing"
    node            = "ys-dev"
    node_id         = "<id>"
    node_address    = "x.x.x.x"
    node_datacenter = "dc1"
    node_tagged_addresses = {
      lan      = "x.x.x.x"
      lan_ipv4 = "x.x.x.x"
      wan      = "x.x.x.x"
      wan_ipv4 = "x.x.x.x"
    }
    node_meta = {
      consul-network-segment = ""
    }
    cts_user_defined_meta = {}
  },
  "web.mayank-dev.dc1" : {
    id              = "web"
    name            = "web"
    kind            = ""
    address         = "x.x.x.x"
    port            = 80
    meta            = {}
    tags            = ["rails"]
    namespace       = null
    status          = "passing"
    node            = "mayank-dev"
    node_id         = "<id>"
    node_address    = "x.x.x.x"
    node_datacenter = "dc1"
    node_tagged_addresses = {
      lan      = "x.x.x.x"
      lan_ipv4 = "x.x.x.x"
      wan      = "x.x.x.x"
      wan_ipv4 = "x.x.x.x"
    }
    node_meta = {
      consul-network-segment = ""
    }
    cts_user_defined_meta = {}
  },
  "web.ys-avi-dev-blr.dc1" : {
    id              = "web"
```

```
    name            = "web"
    kind            = ""
    address         = "x.x.x.x"
    port            = 89
    meta            = {}
    tags            = ["rails"]
    namespace       = null
    status          = "passing"
    node            = "ys-dev"
    node_id         = "<id>"
    node_address    = "x.x.x.x"
    node_datacenter = "dc1"
    node_tagged_addresses = {
      lan      = "x.x.x.x"
      lan_ipv4 = "x.x.x.x"
      wan      = "x.x.x.x"
      wan_ipv4 = "x.x.x.x"
    }
    node_meta = {
      consul-network-segment = ""
    }
    cts_user_defined_meta = {}
  }
}
```

Note: Network Infrastructure Automation (NIA) compatible modules are built to utilize the above service variables.

The `consul-terraform-sync` manages the entire Terraform workflow for all the individual workspaces corresponding to the defined tasks based on the updates from the services declared in those tasks.

Note: In summary, `consul-terraform-sync` triggers a Terraform workflow based on updates it detects from Consul catalog.