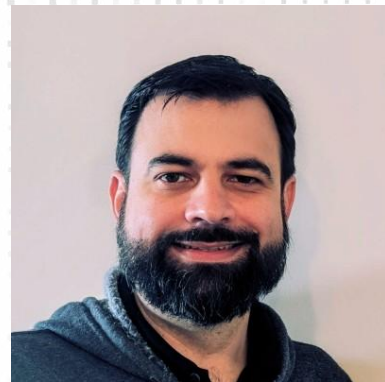


OSS at the Network Edge: Automating DCIM for Accuracy and Speed

Philosophy of Automation, Open Source Software and the Loops
that Bind Us All

Craig Jackson, DevOps Engineer



DevOps Engineer

Redneck Adventurer

Synthesizer Enthusiast

Process / Automation

Studied Horticulture and like growing plants!

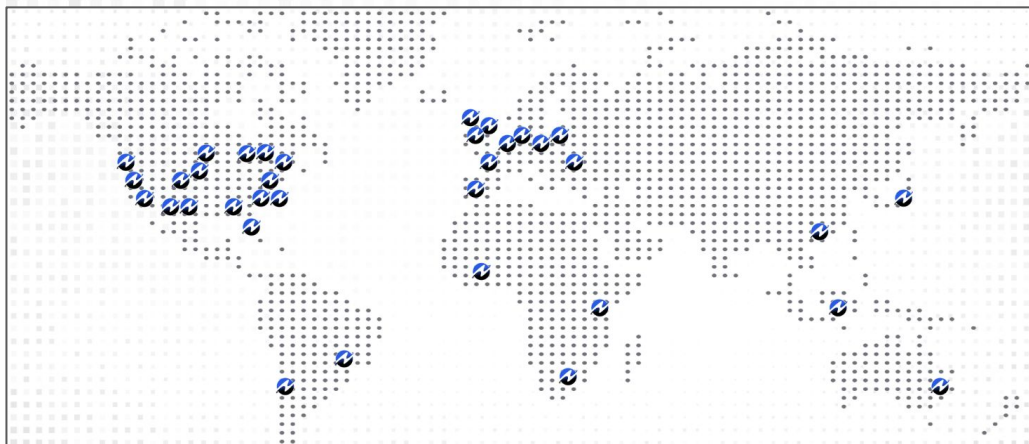
IRC: wicked@EFNet - Ansi art fan

cjackson@netactuate.com



NetActuate

NetActuate is a leading provider of highly available, low latency custom network and infrastructure services that reach every major global market. We built one of the world's largest global networks by number of peers, and it serves as the foundation for our performance BGP anycast platform that powers over 25 billion transactions a day.



- Global Cloud Compute Platform
- Managed Anycast Network
 - H-Root DNS Server
- Devops Consulting
- Helping Companies turn monolith services into global architecture

Presentation Journey

- **Abstraction**
- **Source of Truth**
- **Technical Evolution**
 - Let's make a Terraform Module!
- **Orchestration**
- **Axioms Revisited**

But First, Let's come to terms...

DCIM

Datacenter Infrastructure Management - holds the data and provides views into your infrastructure.

Infrastructure as Code

Declarative Infrastructure – Defining your infrastructure in a configuration file, declaring the desired or defined state. (outsourcing the actual creating of infrastructure to an orchestration loop)

Imperative Infrastructure – Script driven infrastructure defining HOW infrastructure gets created. Chef or scripts containing a series of API calls in order, for example. Still beats a web interface (checklist approach)

Chaos

Unreconciled state

- Where intent and reality drift apart.
- Systems exist in unpredictable states.
- Configuration lives in many places
 - scripts, consoles, human memory.
- Change is manual, reactionary, and impermanent.

You control by doing

Order

Continuous reconciliation

- The system knows what “should be,” and works to make it so.
- State is declarative, observable, and self-healing.
- Humans express intent; controllers enforce reality.

You control by describing

In Every Creation Myth...

"From the primal void of Chaos emerged Gaia – the first hint of Order." — *Hesiod's Theogony*

And the Engineer was born

Every day, we stand between Chaos and Order — between what isn't running, and.... what should be.



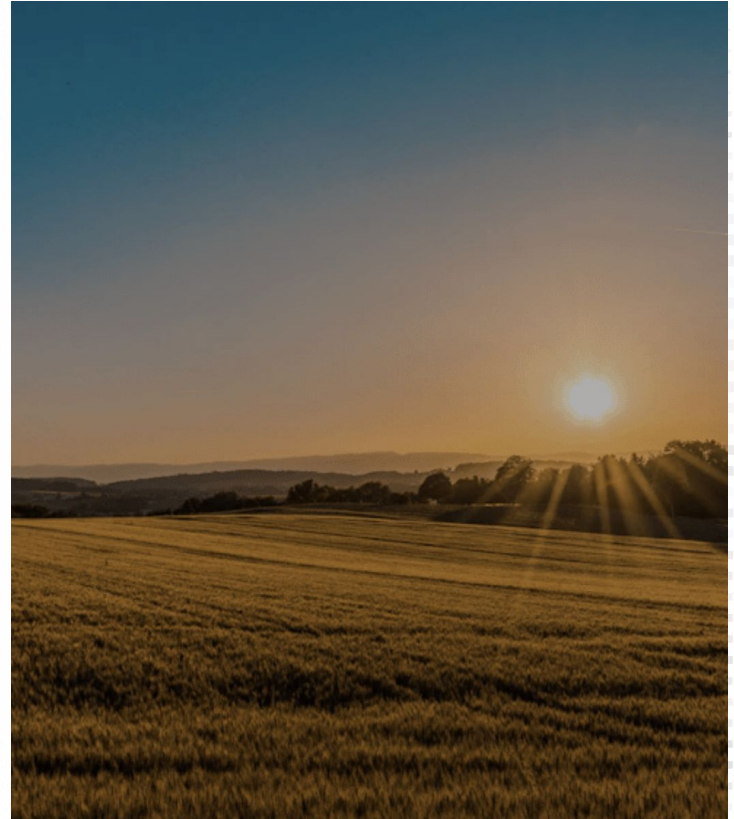
Seasons and Stars: Our Ancient Source of Truth

Ancient societies organized their entire existence around the predictable cycles of nature.

The seasons and stars were their Source of Truth — the fundamental pattern that determined when to plant, harvest, migrate, and celebrate.

These celestial and terrestrial cycles provided a reliable framework that sustained civilization for millennia.

In a world of constant flux, the seasons remain our most ancient and reliable source of truth.



Loop: Moving from Declared to Actual State

Declared State

What should be running — your intent

Actual State

What is running — your reality

The Loop

Continuously detects and corrects the drift between them

The gap between intent and reality narrows with each iteration of the loop.



IAC

Infrastructure as Code: And the systems around them

The reconciliation loop is the heartbeat of self-correcting systems

Chaos



Order

Application Lifecycle Mgmt.



SCM/VCS



Testing



Deployment



Cloud / IaaS / PaaS



Communication & ChatOps



CI



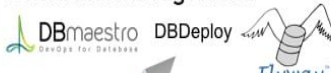
Knowledge Sharing



Build



Database Management



Config Mgmt./Provisioning



Orchestration & Scheduling



Artefact Management



BI / Monitoring / Logging





The Holy Trinity of Hybrid Cloud Automation



NetBox: Multiple Views of Infrastructure

NetBox provides different focused views of your infrastructure:

Organizational Hierarchy:

Sites, racks, and device grouping

Device Relationships:

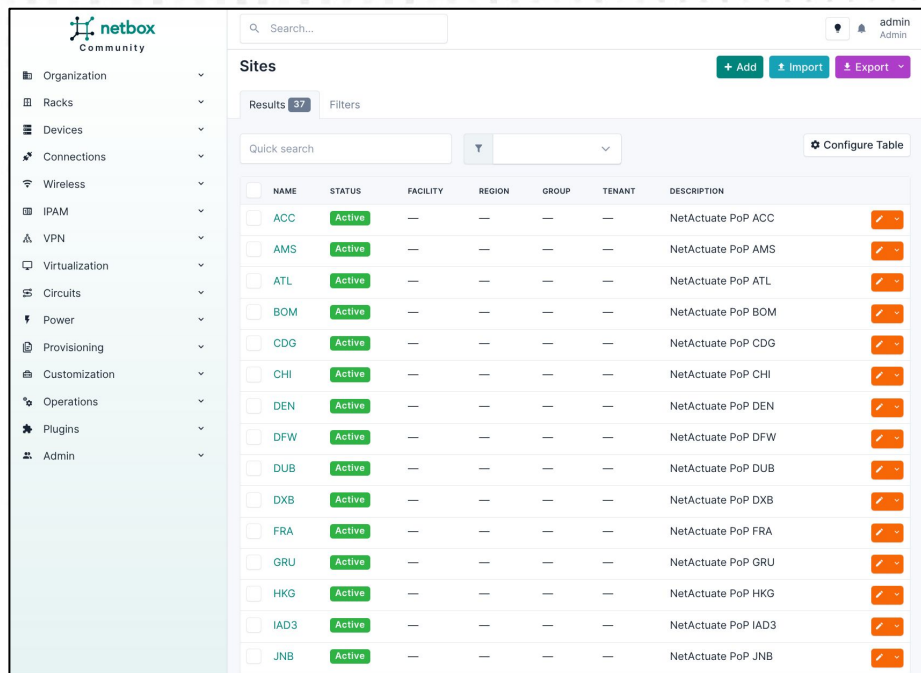
Connections between physical and virtual components

Network Topology:

Circuit paths, IP addressing, and VLAN structures

Operational Context:

Status tracking and change management across regions



	NAME	STATUS	FACILITY	REGION	GROUP	TENANT	DESCRIPTION	
☐	ACC	Active	—	—	—	—	NetActuate PoP ACC	
☐	AMS	Active	—	—	—	—	NetActuate PoP AMS	
☐	ATL	Active	—	—	—	—	NetActuate PoP ATL	
☐	BOM	Active	—	—	—	—	NetActuate PoP BOM	
☐	CDG	Active	—	—	—	—	NetActuate PoP CDG	
☐	CHI	Active	—	—	—	—	NetActuate PoP CHI	
☐	DEN	Active	—	—	—	—	NetActuate PoP DEN	
☐	DFW	Active	—	—	—	—	NetActuate PoP DFW	
☐	DUB	Active	—	—	—	—	NetActuate PoP DUB	
☐	DXB	Active	—	—	—	—	NetActuate PoP DXB	
☐	FRA	Active	—	—	—	—	NetActuate PoP FRA	
☐	GRU	Active	—	—	—	—	NetActuate PoP GRU	
☐	HKG	Active	—	—	—	—	NetActuate PoP HKG	
☐	IAD3	Active	—	—	—	—	NetActuate PoP IAD3	
☐	JNB	Active	—	—	—	—	NetActuate PoP JNB	

Each view becomes a dimension of your Source of Truth, enabling targeted automation workflows

NetBox: Modern Source of Truth

NetBox provides different focused views of your infrastructure:

DCIM as Your Foundation

Centralized repository of infrastructure truth

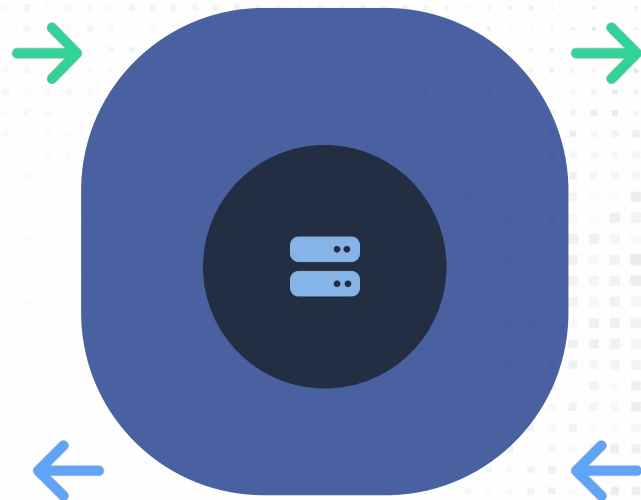
Automation In

Scripts and integrations continuously feed real-world data into NetBox

Automation Out

Use the trusted data to orchestrate further processes, configurations, and workflows

When your source of truth is well-maintained, your entire infrastructure evolves with confidence.



Data Infrastructure Management:
The single Source of Truth

From Terraform to OpenTofu

Infrastructure as Code Evolution

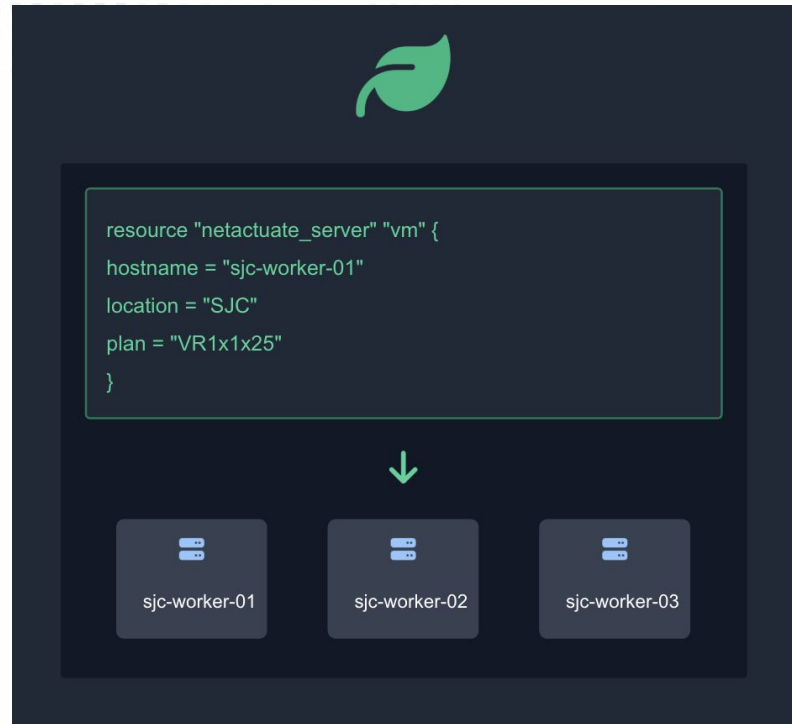
Terraform pioneered infrastructure as code, while OpenTofu maintains compatibility and continues the powerful declarative configuration approach.

Open Tofu Technical Benefits

- Familiar syntax and workflow
- Drop-in Replacement for existing code
- Strong ecosystem provider compatibility

Declarative Orchestration

Both tools let you describe what you want and handle the how — making infrastructure deployment consistent, repeatable, and reliable.



Infrastructure as code transforms complexity into clarity, regardless of scale.

Infrastructure: As Code.

Organized ... Chaos

*If you touch it twice, you've
failed to plan once.*

```
basic - main.tf
1 provider "netactuate" {
2   # Optional: API key can be set here, rather than as an environment variable
3   //api_key = "NETACTUATE_API_KEY"
4 }
5
6 # Define SSH Key to use for server login
7 resource "netactuate_sshkey" "sshkey" {
8   name = "default_key"
9   key  = "ssh-ed25519 REDACTED_SSH_KEY user@email.test"
10 }
11
12 # Define locations, OS, and instance sizing
13 resource "netactuate_server" "server" {
14   hostname = "terraform.example.com"
15   plan     = "VR1x1x25"
16   location = "SJC"
17   image    = "Ubuntu 22.04 (20221110)"
18   ssh_key_id = netactuate_sshkey.sshkey.id
19   package_billing_contract_id = PROVIDED_CODE
20 }
```

This approach worked but it
didn't scale with growing
infrastructure needs.



Static definition, manual provisioning

for_each: Server Patterns

From Static to Dynamic

For_each creates dependencies that become hard to track as they get nested.

```
terraform-provider-netactuate - main.tf
1 provider "netactuate" {
2   # Optional: API key can be set here, rather than as an environment variable
3   //api_key = "NETACTUATE_API_KEY"
4 }
5
6 # Define SSH Key to use for server login
7 resource "netactuate_sshkey" "sshkey" {
8   name = "default key"
9   key  = "ssh-ed25519 SHORT_SSH_KEY_REDACTED_email@email.test"
10 }
11
12 # Define locations, OS, and instance sizing ( fqdn = { location name , OS_image , VM sizing } )
13 locals {
14   server_settings = {
15     "server-1.terraform.test" = { location = "LGA", image = "Ubuntu 22.04 (20221110)", plan = "VR1x1x25" },
16     "server-2.terraform.test" = { location = "AMS", image = "Ubuntu 22.04 (20221110)", plan = "VR2x2x25" },
17     "server-3.terraform.test" = { location = "LAX", image = "Ubuntu 22.04 (20221110)", plan = "VR1x1x25" }
18   }
19 }
20
21 # Define server creation values
22 resource "netactuate_server" "map" {
23   for_each = local.server_settings
24   hostname = each.key
25   plan     = each.value.plan
26   location = each.value.location
27   image    = each.value.image
28   ssh_key_id = netactuate_sshkey.sshkey.id
29   package_billing_contract_id = PROVIDED_CODE
30 }
```

Now every new server in any location is one line away

"But beware of unearned knowledge..." - C. G. Jung

Modules: Cultural Memory

Modules are just nested loops that you can abstract into.

- A “PoP” is now an abstract terraform concept that contains several Netbox resources that contains NetActuate VM’s that will populate all the inherited fields.

"Who is watching the watchers" - Juvenal's Satires

```
netbox - main.tf

69 #####
70 # Per-PoP module
71 #####
72 module "pop" {
73     for_each      = var.pops
74     source        = "../modules/pop"
75     cluster_type_id = netbox_cluster_type.hypervisor.id
76     vm_tags       = [netbox_tag.anycast.name]
77     pop_code      = each.key
78     location_id   = each.value.location_id
79     plan          = each.value.plan
80     image_id      = var.image_id
81     hostname_prefix = each.value.hostname_prefix
82
83     contract_id = var.contract_id
84     asn         = var.bgp_asn
85     local_as_id = netbox_asn.local.id
86     remote_as_id = netbox_asn.peer.id
87     bgp_ipv4_prefix = var.bgp_ipv4_prefix
88
89     bgp_group_id = var.bgp_group_id
90     bgp_enable_ipv6 = var.bgp_enable_ipv6
91
92     device_role_id = netbox_device_role.anycast_worker.id
93     device_type_id = netbox_device_type.anycast_worker.id
94
95     # NEW: key id for server creation
96     ssh_key_id = var.ssh_key_id
97
98     depends_on = [
99         netbox_custom_field.na_server_id,
100         netbox_custom_field.na_contract_id,
```

Modules: Cultural Memory



Reduce Reuse Recycle

- Try and build your modules to be used in multiple projects.
- If you build a DNS service on NetActuate, build your module so that you can also build another service on us just by copy and pasting that module in to your new project.

Take Care of Your Future Self

```
cjackson@NA-Craig:~/infra/demos/netbox/terraform$ tree | gro
├── custom_fields.tf
├── hosts.tftpl
├── inventory.tf
├── main.tf
├── modules
│   └── pop
│       ├── main.tf
│       ├── netactuate
│       │   ├── main.tf
│       │   ├── outputs.tf
│       │   ├── variables.tf
│       │   └── versions.tf
│       ├── netbox
│       │   ├── main.tf
│       │   ├── main.tf.bak
│       │   ├── outputs.tf
│       │   ├── variables.tf
│       │   └── versions.tf
│       ├── variables.tf
│       └── versions.tf
├── outputs.tf
├── providers.tf
├── terraform.tfstate
├── terraform.tfvars
├── variables.tf
└── versions.tf
```

Modules: Early Chaos = Future Order

Abstract Early and Relax Later

- Try and build your modules to be used in multiple projects.
- If you build a DNS service on NetActuate, build your module so that you can also build another service on us just by copy and pasting that module in to your new project.



netbox - terraform.tfvars

```
8 # PoPs (plan defaults to VR1x1x25, image_id fixed at 5787)
9 pops = {
10 #####
11 # START (current small US footprint)
12 #####
13 RDU = { location_id = 236, worker_count = 2 } # RDU - Raleigh, NC
14 IAD3 = { location_id = 245 } # IAD3 - Ashburn, VA
15 CHI = { location_id = 31 } # CHI - Chicago, IL
16 DEN = { location_id = 122 } # DEN - Denver, CO
17 PDX = { location_id = 571 } # PDX - Portland, OR
18 SEA = { location_id = 124 } # SEA - Seattle, WA
```

It's 10:00 PM Do you know where your service is?

Modules: Combining NetBox and NetActuate

Automated PoP registration in your Source of Truth



NetBox + NetActuate Integration

"Don't Panic" - The Hitchhiker's Guide to the Galaxy - Douglas Adams

```
netbox - main.tf
1 #####
2 # NetBox: Site + Cluster for this PoP
3 #####
4
5 resource "netbox_site" "this" {
6     name          = var.pop_code
7     slug          = lower(var.pop_code)
8     description   = "NetActuate PoP ${var.pop_code}"
9     status        = "active"
10 }
11
12 resource "netbox_cluster" "this" {
13     name          = "${var.pop_code}-cluster"
14     cluster_type_id = var.cluster_type_id
15     site_id       = netbox_site.this.id
16 }
17
18 resource "netbox_device" "this" {
19     name          = netactuate_server.vm.hostname
20     device_type_id = var.device_type_id
21     role_id       = var.device_role_id
22     site_id       = netbox_site.this.id
23     status        = "active"
24     tags          = var.vm_tags
25 }
26
27 #####
28 # NetActuate: Build one VM in this PoP
29 #####
```

Orchestrating from NetBox

Using Your Source of Truth to Drive Automation

NetBox provides structured data for infrastructure state

Custom fields extend tracking capabilities

API enables dynamic inventory extraction

*When your source of truth is reliable, your
automation becomes trustworthy*

NetBox Dynamic Inventory

```
plugin: netbox.netbox.nb_inventory
query_filters:
  - site: "LAX"
  - status: "active"
group_by:
  - device_roles
```

“” AWX Workflows

- Automated provisioning from NetBox data
- Event-driven workflows trigger on changes
- Role-based access control for operations teams

The Terrable Stack - Terraform into Ansible

Real-time orchestration of services from infrastructure defined in NetBox

Ansible tasks execute against nodes discovered from NetBox, Configuring and starting all necessary services on the edge.

```
TASK [Node lookup] *****
ok: [anycast-ingress-den.worker.demo -> localhost]

TASK [Initialize node configuration] *****
skipping: [anycast-ingress-den.worker.demo]

TASK [Append node configuration] *****
ok: [anycast-ingress-den.worker.demo]

TASK [Set SSH IP] *****
ok: [anycast-ingress-den.worker.demo]

TASK [Update node vars] *****
changed: [anycast-ingress-den.worker.demo -> localhost]

PLAY [Setup BGP on nodes] *****

TASK [Gathering Facts] *****
ok: [anycast-ingress-den.worker.demo]

TASK [ssh : authorized_key] *****
changed: [anycast-ingress-den.worker.demo]

TASK [rc.local : include_vars] *****
ok: [anycast-ingress-den.worker.demo]

TASK [rc.local : Create rc.local] *****
changed: [anycast-ingress-den.worker.demo]

TASK [rc.local : Set permissions for rc.local] *****
[ ]
```



From source of truth to provisioned service

A Lot of Systems Integrate!

```
terraform-provider-netactuate - main.tf

34 provider "grafana" {
35     url = "http://grafana.example.com/"
36     auth = var.grafana_auth
37 }
38
39 // Optional (On-premise, not supported in Grafana Cloud): Create an organization
40 resource "grafana_organization" "my_org" {
41     name = "my_org"
42 }
43
44 // Create resources (optional: within the organization)
45 resource "grafana_folder" "my_folder" {
46     org_id = grafana_organization.my_org.org_id
47     title = "Test Folder"
48 }
49
```

```
terraform-provider-netactuate - main.tf

61 # Configure the Icinga2 provider
62 provider "icinga2" {
63     api_url = "https://192.168.33.5:5665/v1"
64     api_user = "root"
65     api_password = "icinga"
66     insecure_skip_tls_verify = true
67 }
68
69 # Configure a host
70 resource "icinga2_host" "web-server" {
71     # ...
72 }
```



 Slow is Smooth; Smooth is Fast

 Don't Touch It Twice

 We Don't Log In Anymore

What Does This Philosophy Enable?

If you Maintain a Source of Truth:

- Consistency
- Reliability
- Observability
- Automate Documentation

If you Automate configurations:

- Program RBAC Controls
 - Hire more effectively for specific smaller roles
- Upgrades are consistent and predictable
- Automate Monitoring and Observability

If you Automate Infrastructure:

- Scale Quickly
- React Quickly

.... And all of this is version controlled.

..... you can roll back infrastructure.

Enjoy the fruits of your labor

Closing: The Great Loop of Truth



"Everything Flows" – Heraclitus



Thank You

Where to go from here?

Github

<http://www.github.com/netactuate/>

Terraform Module

Ansible Playbooks



anycast  com

Anycast.com

Resource Hub for BGP
Anycast & Networking

<https://anycast.com/>

The Serial Port

From Dial up to DevOps

Booth #49

<https://www.youtube.com/@theserialport>



END

Two Column Example

- Vestibulum non lorem justo. Nulla et euismod elit. Morbi magna ex, condimentum eget turpis id, mollis sagittis libero.
- Morbi magna ex, condimentum eget turpis id, mollis sagittis libero.
- Morbi magna ex, condimentum eget turpis id, mollis sagittis libero.
- Morbi magna ex, condimentum eget turpis id, mollis sagittis libero.
- Morbi magna ex, condimentum eget turpis id, mollis sagittis libero.
- Morbi magna ex, condimentum eget turpis id, mollis sagittis libero.

Three Column Example

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum non lorem justo. Nulla et euismod elit. Morbi magna ex, condimentum eget turpis id, mollis sagittis libero.

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

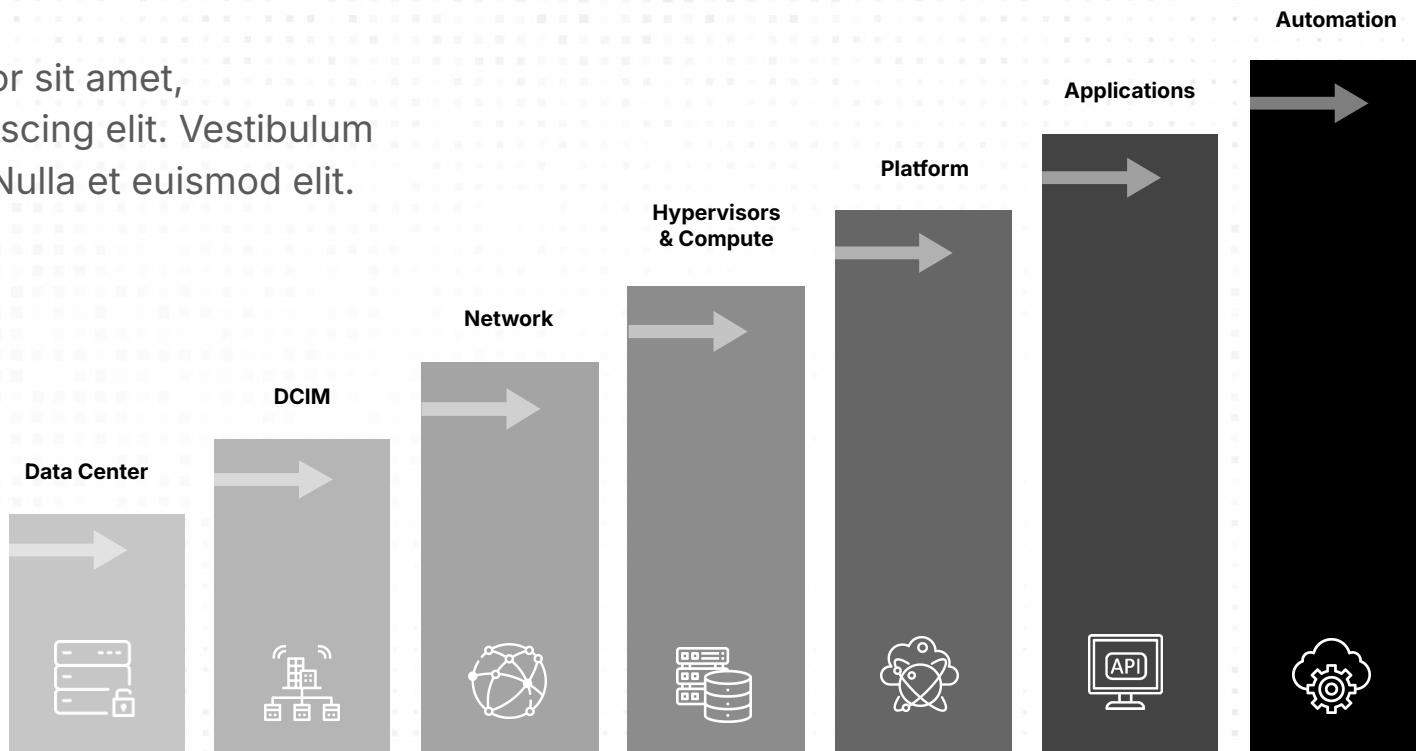
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum non lorem justo. Nulla et euismod elit. Morbi magna ex, condimentum eget turpis id, mollis sagittis libero.

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum non lorem justo. Nulla et euismod elit. Morbi magna ex, condimentum eget turpis id, mollis sagittis libero.

Graph Example

Lorem ipsum dolor sit amet,
consectetur adipiscing elit. Vestibulum
non lorem justo. Nulla et euismod elit.

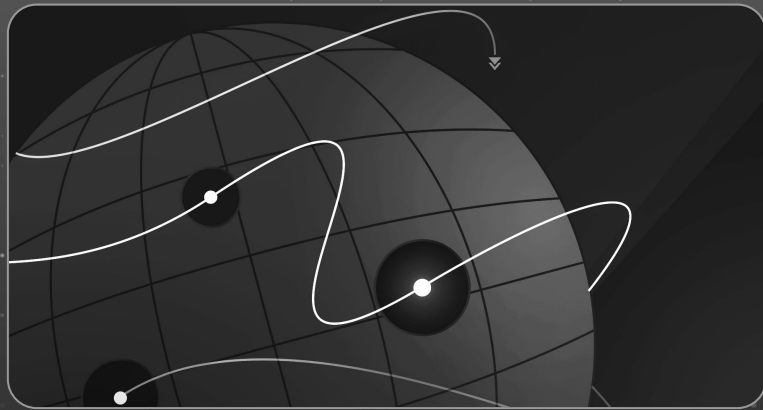


Feature Example

Lorem ipsum dolor sit amet,
consectetur adipiscing elit.

Vestibulum non lorem justo. Nulla
et euismod elit. Morbi magna ex,
condimentum eget turpis id, mollis
sagittis libero. Morbi et rutrum
nulla, sit amet luctus diam. Nam
imperdiet erat turpis, quis faucibus
est maximus non.

Feature Title



Lorem ipsum dolor sit amet, consectetur adipiscing
elit. Vestibulum non lorem justo. Nulla et euismod
elit. Morbi magna ex, condimentum eget turpis id,
mollis sagittis libero.

Breaker Slide

Our Global Footprint



Thank You.

25350 Magic Mountain Pkwy, Ste 300, Valencia, CA 91355

+1 919 381 5400 (p) netactuate.com info@netactuate.com