ANSIBLE for CLOUD AUTOMATION

Marcos Garcia
Senior Cloud Solutions Architect

FOUR WAYS TO AUTOMATE YOUR CLOUD

With Ansible

Cloud Provider's Native CLI

Ansible Cloud Modules

Cloud Provider's Orchestration Templates

Terraform Module

Case Study: NASA - automating AWS with Ansible

NASA needed to move roughly 65 applications from a traditional hardware based data center to a cloud-based environment for better agility and cost savings. The rapid timeline resulted in **many applications being migrated 'as-is' to a cloud environment.**

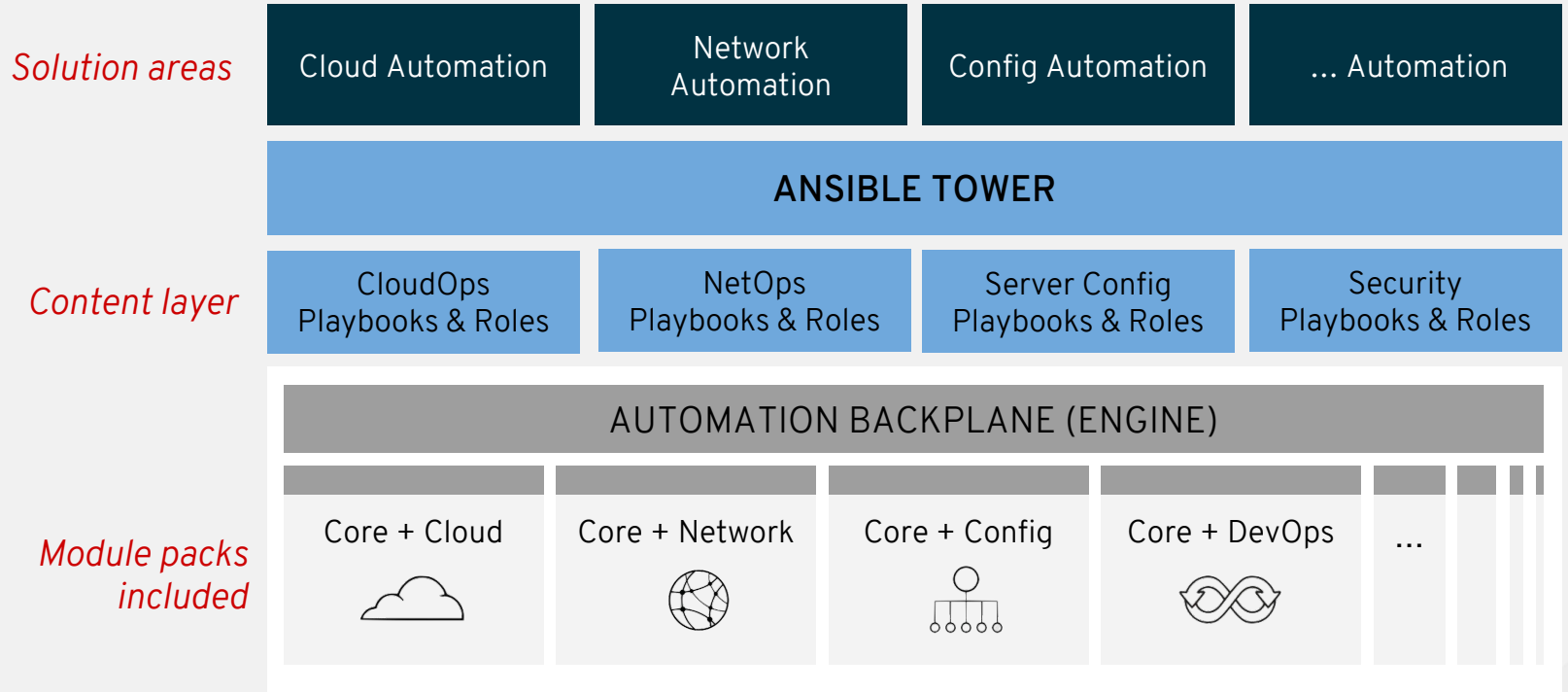
This created an environment **spanning multiple virtual private clouds (VPCs) and AWS accounts** that could not be easily managed. Even simple things, like ensuring every system administrator had access to every server, or simple patching, were **extremely burdensome**

As a result of implementing **Ansible Tower**, NASA is better equipped to manage its AWS environment. Tower allowed NASA to provide **better operations and security** to its clients. It has also **increased efficiency** as a team.

By the numbers:

- Updating nasa.gov went from over 1 hour to under 5 minutes
- Patching updates went from a multi-day process to 45 minutes
- Achieving near real-time RAM and disk monitoring (accomplished without agents)
- Provisioning OS Accounts across entire environment in under 10 minutes
- Baselining standard AMIs went from 1 hour of manual configuration to becoming an invisible and seamless background process
- Application stack set up from 1-2 hours to under 10 minutes per stack

Ansible Automation



Ansible 101

WHY ENTERPRISE-WIDE IT AUTOMATION IS ELUSIVE TODAY

1

PEOPLE PROBLEMS

Skills gaps & org charts get in the way

2

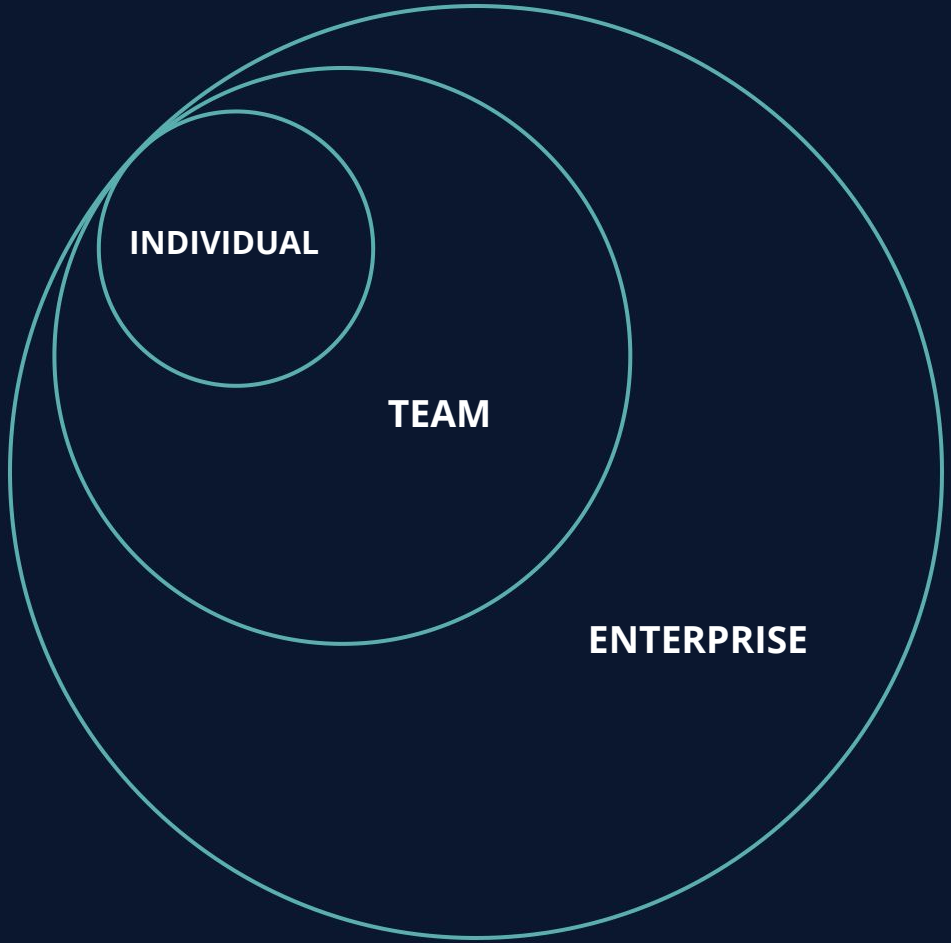
POINT TOOLS

Proliferation of point solutions and vendor-specific tools

3

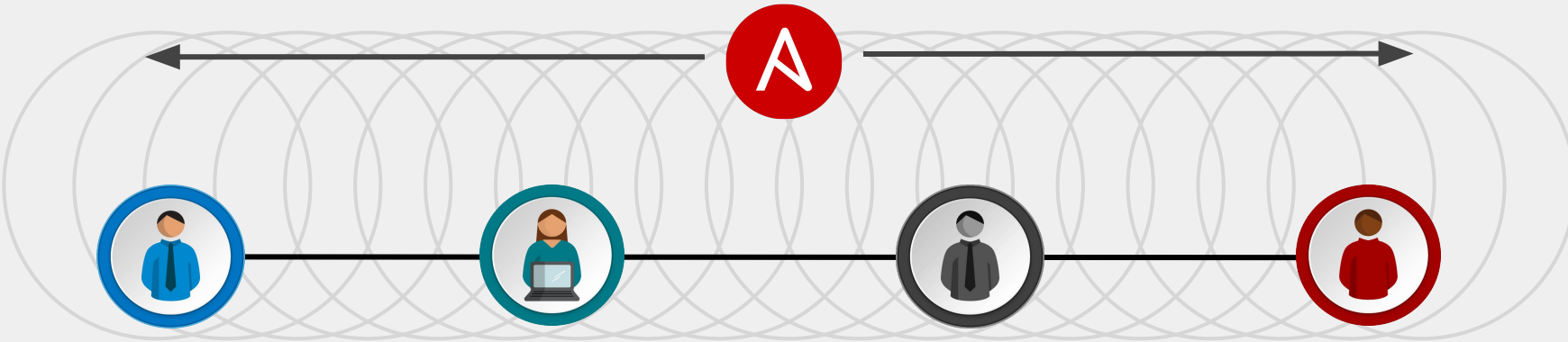
PACE OF INNOVATION

Automation requires integration across domains



An enterprise-wide automation strategy must benefit individuals first.

ANSIBLE IS THE UNIVERSAL LANGUAGE

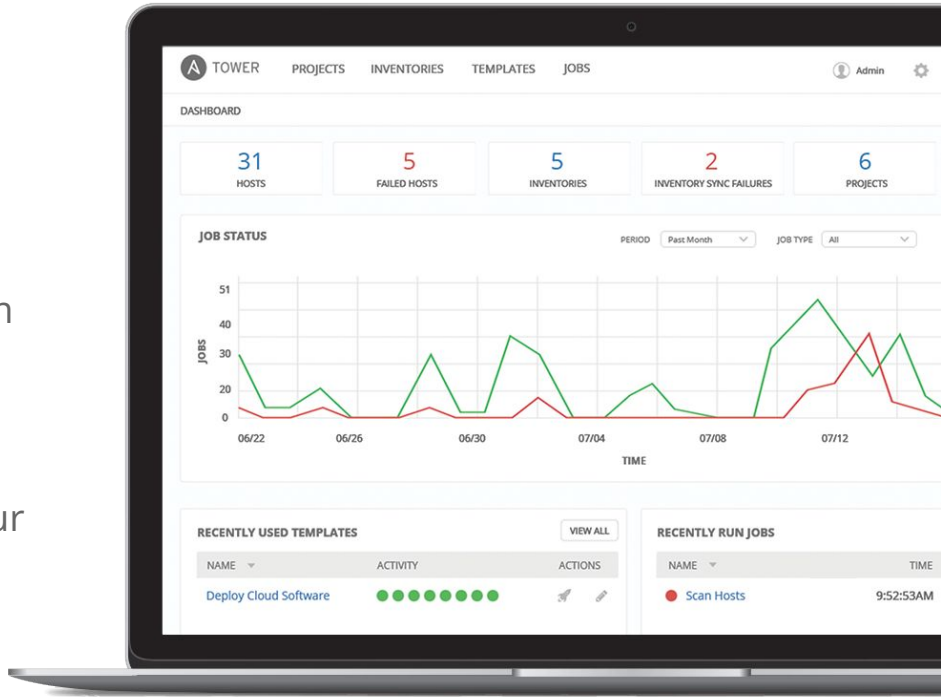


WHAT IS ANSIBLE AUTOMATION?

The Ansible project is an open source community sponsored by Red Hat. It's also a **simple automation language** that perfectly describes IT application environments in **Ansible Playbooks**.

Ansible Engine is a **supported product** built from the Ansible community project.

Ansible Tower is an **enterprise framework** for controlling, securing, managing and extending your Ansible automation (community or engine) with a **UI and RESTful API.**




ANSIBLE
v1 - Set config file to use on boot
1. Write multiple configuration files
• For each environment/region
2. Inspect metadata on boot and use the matching config file
v1 - Set config file to use on boot
1. Write multiple configuration files
• For each environment/region
2. Inspect metadata on boot and use the matching config file
30,000+
Stars on GitHub
1900+
Ansible modules
500,000+
Downloads a month

THE ANSIBLE WAY

CROSS PLATFORM

Agentless support for all major OS variants, physical, virtual, cloud and network devices.

HUMAN READABLE

Perfectly describe and document every aspect of your application environment.

PERFECT DESCRIPTION OF APPLICATION

Every change can be made by Playbooks, ensuring everyone is on the same page.

VERSION CONTROLLED

Playbooks are plain-text. Treat them like code in your existing version control.

DYNAMIC INVENTORIES

Capture all the servers 100% of the time, regardless of infrastructure, location, etc.

ORCHESTRATION PLAYS WELL WITH OTHERS

Every change can be made by Playbooks, ensuring everyone is on the same page.

WHAT CAN I DO WITH ANSIBLE?

Automate the deployment and management of your entire IT footprint.

Do this...

Orchestration	Configuration Management	Application Deployment	Provisioning	Continuous Delivery	Security and Compliance

On these...

Firewalls	Load Balancers	Applications	Containers	Clouds
Servers	Infrastructure	Storage	Network Devices	**And more...**

ANSIBLE SHIPS WITH OVER 1900 MODULES

ANSIBLE

CLOUD

AWS
Azure
CenturyLink
CloudScale
Digital Ocean
Docker
Google
Linode
OpenStack
Rackspace
And more...

VIRT AND CONTAINER

Docker
VMware
RHEV
OpenStack
OpenShift
Atomic
CloudStack
And more...

WINDOWS

ACLs
Files
Commands
Packages
IIS
Regedits
Shell
Shares
Services
DSC
Users
Domains
And more...

NETWORK

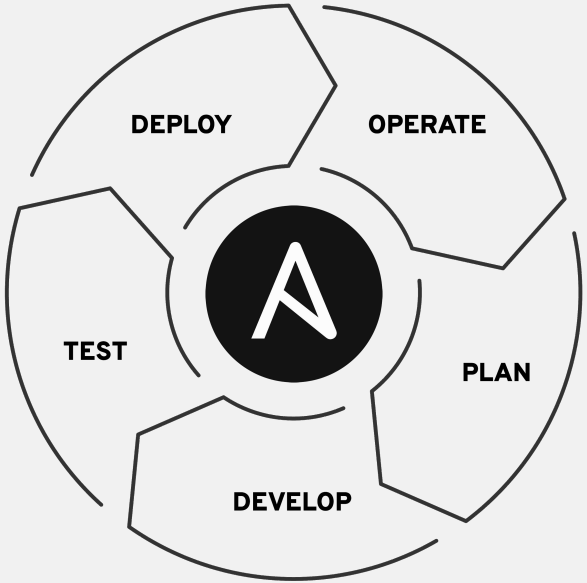
Arista
A10
Cumulus
Big Switch
Cisco
Cumulus
Dell
F5
Juniper
Palo Alto
OpenSwitch
And more...

NOTIFY

HipChat
IRC
Jabber
Email
RocketChat
Sendgrid
Slack
Twilio
And more...

START SMALL, THINK BIG

Three high-level benefits for successful network operations



INFRASTRUCTURE AS YAML

- Automate backup & restores
- Manage “golden” versions of configurations

CONFIGURATION MANAGEMENT

- Changes can be incremental or wholesale
- Make it part of the process: agile, waterfall, etc.

ENSURE AN ONGOING STEADY STATE

- Schedule tasks daily, weekly, or monthly
- Perform regular state checking and validation

ANSIBLE NETWORK

OPERATIONS CENTRIC NETWORK AUTOMATION

- Build and push device configurations
- Automate tactical operations on network devices

APPLICATION CENTRIC NETWORK AUTOMATION

- Automate network devices in support of applications
- Support direct to device and controller based virtualization

CLOUD CENTRIC NETWORK AUTOMATION

- Describe and deploy network connectivity between clouds
- Support public/private and/or public/public clouds

Cloud CLI

Cloud Provider's CLI

AWS

$ yum install **awscli**

$ aws configure

(New!)

$ pip install **aws-shell**

Azure

$ yum install **azure**

$ az login

GCP

$ yum install **google-cloud-sdk**

$ gcloud init

An Example

```
- hosts: localhost
  connection: local

  tasks:
  - name: Describe VPCs
      register: vpcs
      local_action:
      module: command aws ec2 describe-vpcs
  - name: Print VPCs
      debug:
      msg: "{{item}}"
      with_items: "{{vpcs.stdout | from_json}}.Vpcs"
```

Demo: Ansible CLI for EC2

```
$ ansible-playbook clidemo.yml
```

Ansible Cloud Modules

CLOUD INTEGRATIONS

Ansible includes over 300 modules to support cloud infrastructures, including public clouds:

MORE INFO

MORE INFO

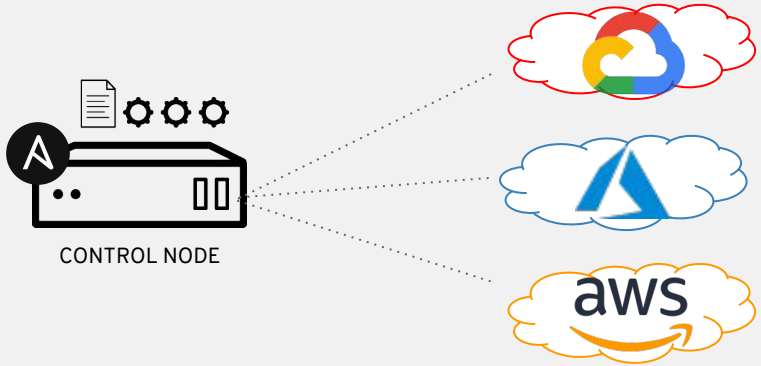
MORE INFO

MORE INFO

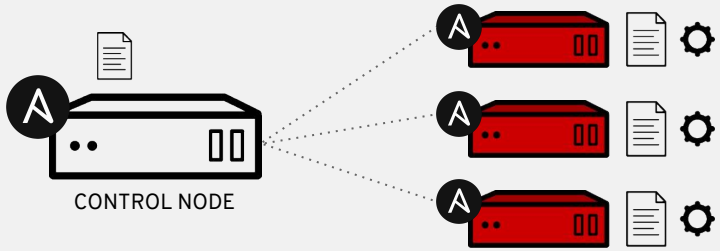
https://www.ansible.com/integrations/cloud

HOW DOES CLOUD AUTOMATION WORK?

Python code is executed locally on the control node **(localhost)**



Python code is copied to the managed node, executed, then removed



Cloud Provider's Python Modules

AWS

https://docs.ansible.com/ansible/2.6/scenario_guides/guide_aws.html

$ pip install **boto3 boto**

> ~/.aws/credentials

OR

> Environment Variables

OR

> Module parameters as Variables in Vault

Azure

https://docs.ansible.com/ansible/2.6/scenario_guides/guide_azure.html

$ pip install **ansible[azure]**

> ~/.azure/credentials

OR

> Environment Variables

OR

> Module parameters as Variables in Vault

GCP

https://docs.ansible.com/ansible/2.6/scenario_guides/guide_gce.html

$ pip install request **google-auth**

> **service_account**_file: file.json

OR

> Environment Variables

OR

> Module parameters as Variables in Vault

An Example

```
- name: Ensure the keypair exists
  ec2_key:
    name: "{{keypair_name}}"
    key_material: "{{ lookup('file', keypair_path) }}"
    region: "{{region}}"

- name: Launch the CentOS AMI
  ec2:
    key_name: "{{keypair_name}}"
    image: "{{ami_id}}"
    region: "{{region}}"
    instance_type: "{{size}}"
    assign_public_ip: yes
    vpc_subnet_id: "{{vpc_subnet_id}}"
  register: myec2

- name: Refresh EC2 facts for that instance
  ec2_instance_facts:
    instance_ids: "{{myec2.instances[0]['id']}}"
```

Demo: Ansible basic EC2

$ ansible-playbook clouddemo.yml

Showcase: Ansible advanced Azure

https://github.com/hornjason/ansible-ocp-azure

https://blog.openshift.com/openshift-container-platform-reference-architecture-implementation-guides/

Ansible and Terraform

Option 1: Ansible calls Terraform

To re-use well-known TF templates from other teams, running in prod with current state

```
- terraform:
    project_path: '{{ project_dir }}'
    state: present
```

Also, Ansible can use Terraform State as a Dynamic Inventory

https://docs.ansible.com/ansible/latest/modules/terraform_module.html#terraform-module

Option 2: Terraform calls Ansible

To provision Instances using Ansible Roles, standardized by IT and used on-premises

```
resource "aws_instance" "jenkins_master" {
    ami = "ami-f95ef58a"
    instance_type = "t2.small"
    subnet_id = "${aws_subnet.jenkins.id}"
    security_group_ids = ["${aws_security_group.jenkins_master.id}"]
    associate_public_ip_address = true
    key_name = "deployer-key"

    # This is where we configure the instance with ansible-playbook
    provisioner "local-exec" {
        command = "sleep 120; ANSIBLE_HOST_KEY_CHECKING=False \
ansible-playbook -u clouduser --private-key ./deployer.pem -i
'${aws_instance.jenkins_master.public_ip},' master.yml"
    }
}
```


https://nicholasbering.ca/tools/2018/01/08/introducing-terraform-provider-ansible/

Demo: Ansible calls Terraform

```
$ ansible-playbook tfdemo.yml
```

Cloud-Specific Orchestration

SILOED AUTOMATION

AWS CloudFormation

```
Resources:
  outsidesite2scarter:
    Type: AWS::EC2::Subnet
    Properties:
      CidrBlock: 10.2.1.0/24
      AvailabilityZone: us-east-1a
      VpcId:
        Ref: site2scarter
      Tags:
       - Key: Name
         Value: outside.site2.scarter
```

Azure Resource Manager Template

```
resources:
- type: Microsoft.Network/virtualNetworks/subnets
   name: "site2.scarter/outside"
   apiVersion: '2017-06-01'
   properties:
     addressPrefix: "10.2.1.0/24"
```

SILOED AUTOMATION

AWS CloudFormation

```
Resources:
  outsidesite2scarter:
    Type: AWS::EC2::Subnet
    Properties:
      CidrBlock: 10.2.1.0/24
      AvailabilityZone: us-east-1a
      VpcId:
        Ref: site2scarter
      Tags:
       - Key: Name
         Value: outside.site2.scarter
```

Azure Resource Manager Template

```
resources:
- type: Microsoft.Network/virtualNetworks/subnets
   name: "site2.scarter/outside"
   apiVersion: '2017-06-01'
   properties:
     addressPrefix: "10.2.1.0/24"
```

SO MANY WORDS… but only a few things matter

TODAY…

2010 Era silos

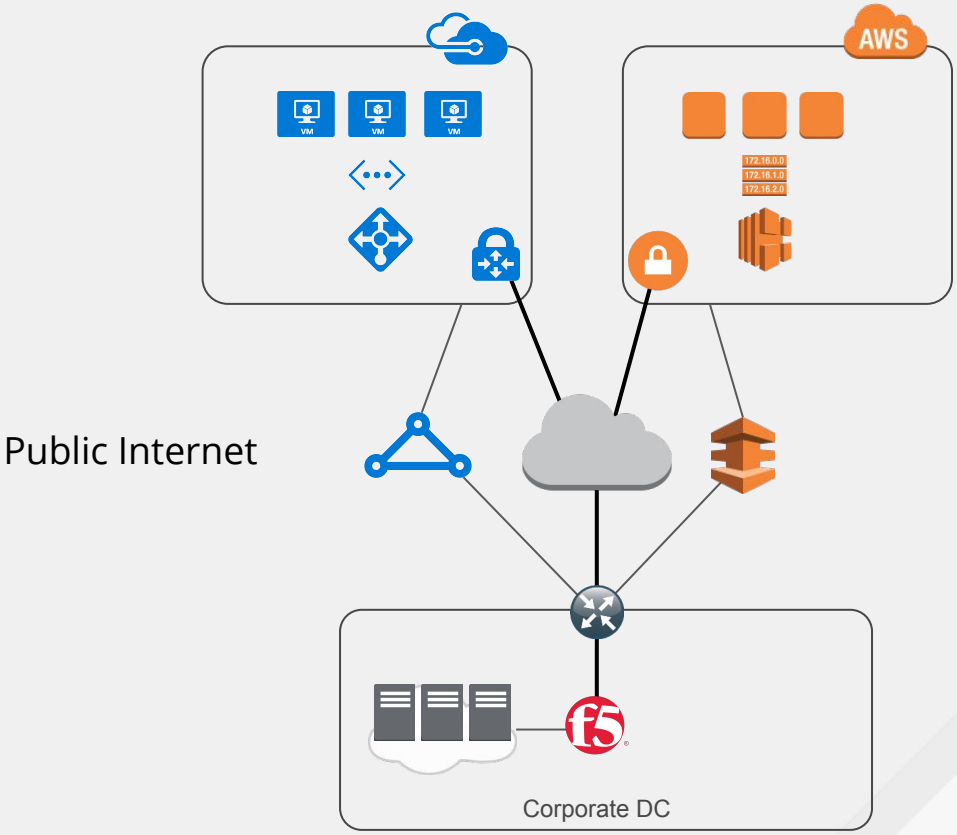
Multiple VPN Options:

- AWS Virtual Private Gateway
- Azure VPN Gateway
- GCE Cloud VPN

Multiple peering options:

- AWS Direct Connect
- Azure ExpressRoute
- GCE Dedicated Interconnect

They cannot even agree on the icons!



DATA MODELS

Better Living Through Abstraction

```
vpc_list:
- name: site2.scarter
  cidr: 10.2.0.0/16
  networks:
    - name: mgmt.site2.scarter
      cidr: 10.2.0.0/24
    - name: outside.site2.scarter
      cidr: 10.2.1.0/24
    - name: inside.site2.scarter
      cidr: 10.2.2.0/24
```

Azure Resource Manager Template

```
resources:
- type: Microsoft.Network/virtualNetworks/subnets
   name: "site2.scarter/outside"
   apiVersion: '2017-06-01'
   properties:
     addressPrefix: "10.2.1.0/24"
```

AWS CloudFormation

```
Resources:
  outsidesite2scarter:
    Type: AWS::EC2::Subnet
    Properties:
      CidrBlock: 10.2.1.0/24
      AvailabilityZone: us-east-1a
      VpcId:
        Ref: site2scarter
       Tags:
       - Key: Name
         Value: outside.site2scarter
```

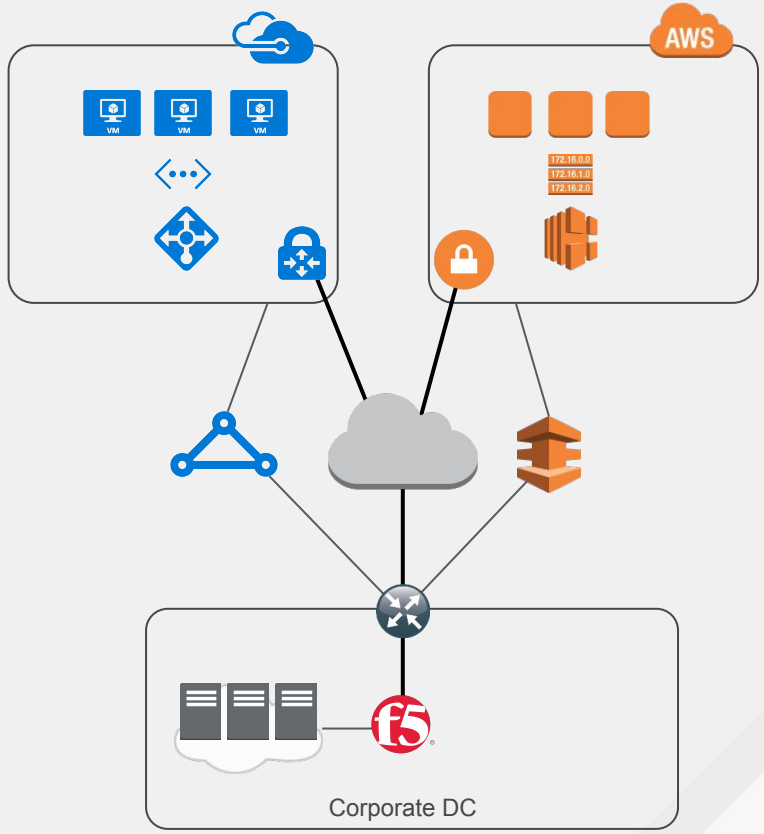
MODEL-DRIVEN INFRASTRUCTURE

```
cloud_acls:
  rtr-acl:
    - { src_ip: 0.0.0.0/0, proto: all }
vpc_list:
  - name: "{{ cloud_name }}"
    provider: "{{ cloud_provider }}"
    region: "{{ cloud_region }}"
    project: "{{ cloud_name }}"
    cidr: "{{ cloud_cidr }}"
    networks:
      - name: "outside.{{ cloud_name }}"
        cidr: "{{ cloud_cidr | ipsubnet(24, 1) }}"
        az: "{{ cloud_region }}a"
      - name: "inside.{{ cloud_name }}"
        cidr: "{{ cloud_cidr | ipsubnet(24, 2) }}"
        az: "{{ cloud_region }}a"
        routes:
          - cidr: "0.0.0.0/0"
            instance: "rtr1.{{ cloud_name }}"
            if_index: 2
    instances:
      - name: "rtr1.{{ cloud_name }}"
        size: medium
        image: csr-byol
        interfaces:
          - name: GigabitEthernet1
```

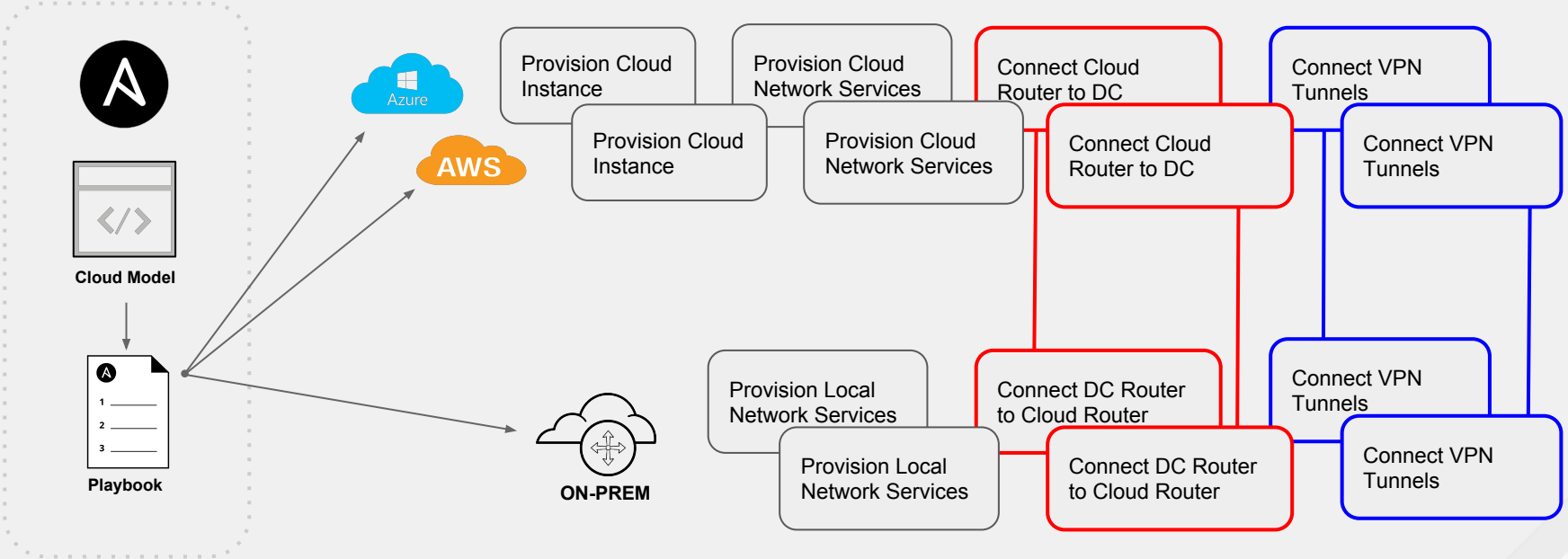
THIS

THIS

THAT

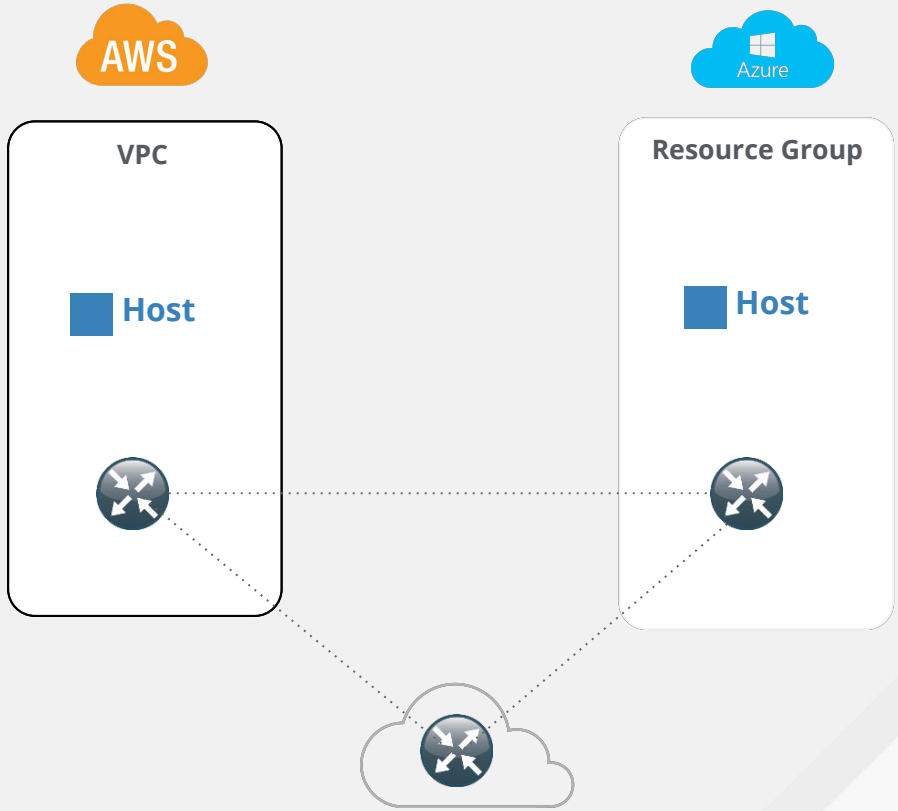


ONBOARDING CLOUD NATIVE



Hybrid Cloud

1. Automate the creation of the VPC and network components.
2. Deploy the same routers, load-balancers, and firewalls that you use on-site.
3. Automate the entire network in a uniform way.



Demo Builder (Cloud Networking)

```
$ cd ansible-cloudbuilder-playbook
$ ansible-playbook build-aws-csr-spoke.yml
$ ansible -m ping -i  inventory/cloudbuilder/csr-lab1.yml  all
```

Bonus: Packer and Docker

When to use Packer vs bare Docker (no k8s)

Packer pros:

- Cloud-aware builds and cloud agnostic
- Works with legacy Virtualization too
- Runs older Linux versions
- Runs any software
- Good ol' Golden Image (ITSM process)
- Builds to Docker too

Packer cons:

- It's just a tool for IaaS
- Need to store binaries for each target
- Yet another tool!

Docker pros:

- Vast collection in docker registry
- Immutable infra
- Path to kubernetes and microservices
- Better portability
- Immediate rollbacks, dependency isolation

Docker cons:

- Needs a running daemon, non-root UID
- Layers, layers, layers!
- Need to modernize (CGroups, SELinux)
- Lacks systemd integration
- No standard process manager

redhat.

THANK YOU


plus.google.com/+RedHat


facebook.com/redhatinc


linkedin.com/company/red-hat


twitter.com/RedHatNews


youtube.com/user/RedHatVideos


red.ht/red-hat-shares

MULTI-SITE/CLOUD DEMO

Scenario: Provision new cloud capacity using template and add to corporate SD-WAN

1. Provision the new Cloud node
2. Configure remote router
   a. Set Hostname, DNS, Banners, etc.
   b. Harden router
   c. Configure Interfaces
   d. Backup
3. Add remote router to VPN
   a. Checkpoint State
   b. Create IPSEC VPN
   c. Configure BGP
   d. Check connectivity
   e. Rollback on failure