ANSIBLE

AUTOMATION FOR NETWORK INFRASTRUCTURE

- **name**: Victor da Costa
  victor:
    role: Senior Specialist Solution Architect EMEA
    network: CCIE #39373
    solution: Ansible by Red Hat
    email: vdacosta@redhat.com
  when: june_05_2018

redhat

Think about it...

Ask yourself the following:

How long does it takes to provision **Network** resources for Physical and Virtual workloads?

What is your average time spent **troubleshooting** configuration **errors**?

How long does it takes to provision new Physical and Virtual Workloads?

What is your average **time** spent with **paperwork** for Change Requests?


MANAGING NETWORKS
HASN’T CHANGED
IN 30 YEARS.

WHY HASN'T NETWORKING CHANGED?

PEOPLE

Domain specific skill sets

Vendor oriented experience

Siloed organizations

Legacy operational practices

PRODUCTS

Infrastructure-focused features

Baroque, CLI-only methodologies

Siloed technologies

Monolithic, proprietary platforms

BIGGEST CHALLENGE FOR ENTERPRISES: CULTURE!

Traditional Network Ops

- Legacy culture
- Risk averse
- Proprietary solutions
- Siloed from others
- “Paper” practices, MOPs
- “Artisanal” networks

Next-Gen Network Ops

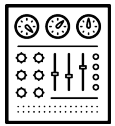
- Community culture
- Risk aware
- Open solutions
- Teams of heroes
- Infrastructure as code
- Virtual prototyping / DevOps

Other Challenges: Complexity, Lack of Agility, OpEX, Anything Manual

THE ROAD TO AUTOMATION

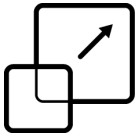
STANDARDIZE
with Red Hat Ansible Engine

- Snapshot State
- Detect Unauthorized Change
- Standardize Existing Configs
- Standardize New Deployments



AUTOMATE
with Red Hat Ansible Engine

- Automate common tasks
- Make changes across any set of network devices
- Validate that changes were successful



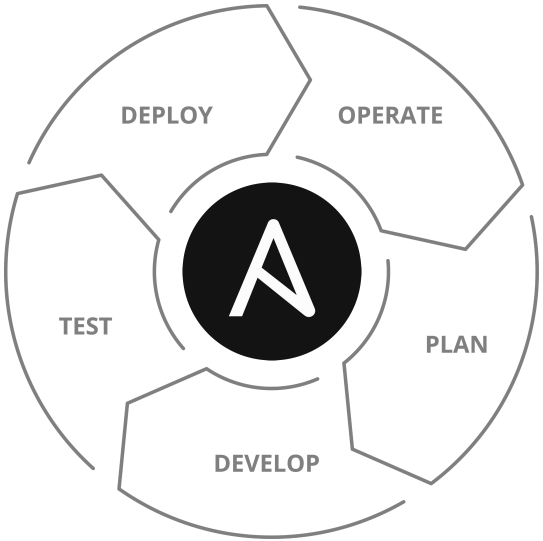
SCALE
with Red Hat Ansible Tower

- Automated deployment from Services Catalogue
- Automated compliance checking & enforcement
- API-Driven Integration with Application Development

Organize → Optimize → SCAAAAAAAAAALE

START SMALL, THINK BIG

3 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

ROI VALUE OF AUTOMATION

GROW REVENUE

- Provides new services and applications
- Adds capacity (new customers, workloads)
- Streamlines and standardizes security and risk mitigation
- Faster time to market

- Reduce lock-in, 40+ networking platforms included
- Do more with less
- Become proactive instead of reactive
- Apply business rules more quickly

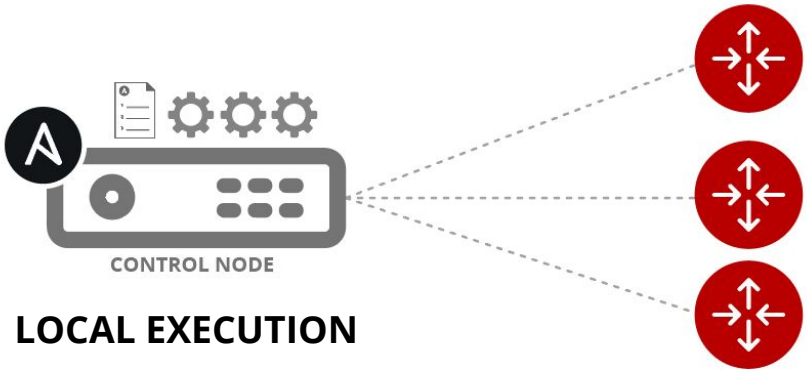
REDUCE COST

Sources: IDC White Paper sponsored by Cisco, , January 2017. Figure 7, p.16.

HOW DOES NETWORK AUTOMATION WORK?

ANSIBLE

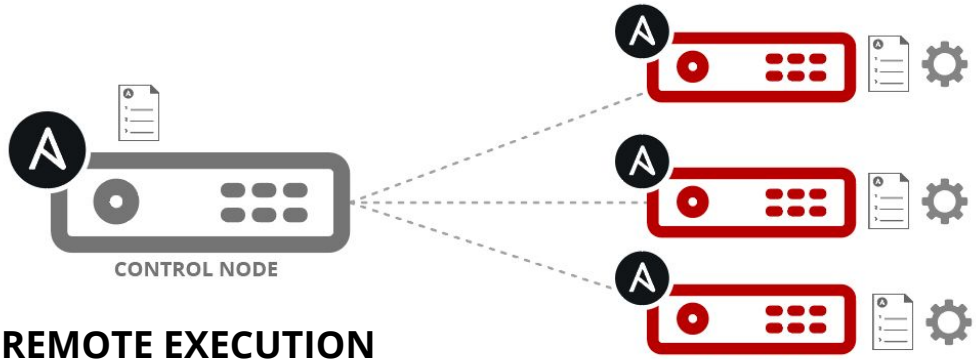
Python code is executed locally on the control node



LOCAL EXECUTION

NETWORKING DEVICES

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



REMOTE EXECUTION

LINUX HOSTS

redhat.

PLAYBOOK EXAMPLE: RHEL

```
---
- name: Configure webservers
  hosts: webservers
  tasks:
    - name: Ensure state of httpd
      yum:
        name: httpd
        state: present

    - name: Ensure state of service
      service:
        name: httpd
        state: started
```

Inventory

Function

```
[network:children]
switches
routers
loadbalancers

[switches]
spine1
spine2

[routers]
rtr1
rtr2
```

Vendor

```
[nxos]
spine1
spine2

[ios]
rtr1

[junos]
rtr2

[panos]
fw01
```

- Use as SoT (Source of Truth).
- Static for ad-hoc activities and small environments.
- Dynamic for wider activities and large/enterprise/multi-site environments.
- Do not keep host_vars/group_vars in the inventory file.
- Groups hosts by function, location, vendor, etc.

PLAYBOOK EXAMPLE: F5

```
---
- name: Configure webservers in loadbalancers
  hosts: loadbalancers
  tasks:
    - name: Ensure node is member of pool
      bigip_pool_member:
        server: "{{ ansible_host }}"
        validate_certs: no
        pool: "http-pool"
        host: "10.1.0.10"
        port: "80"
```

PLAYBOOK EXAMPLE: F5

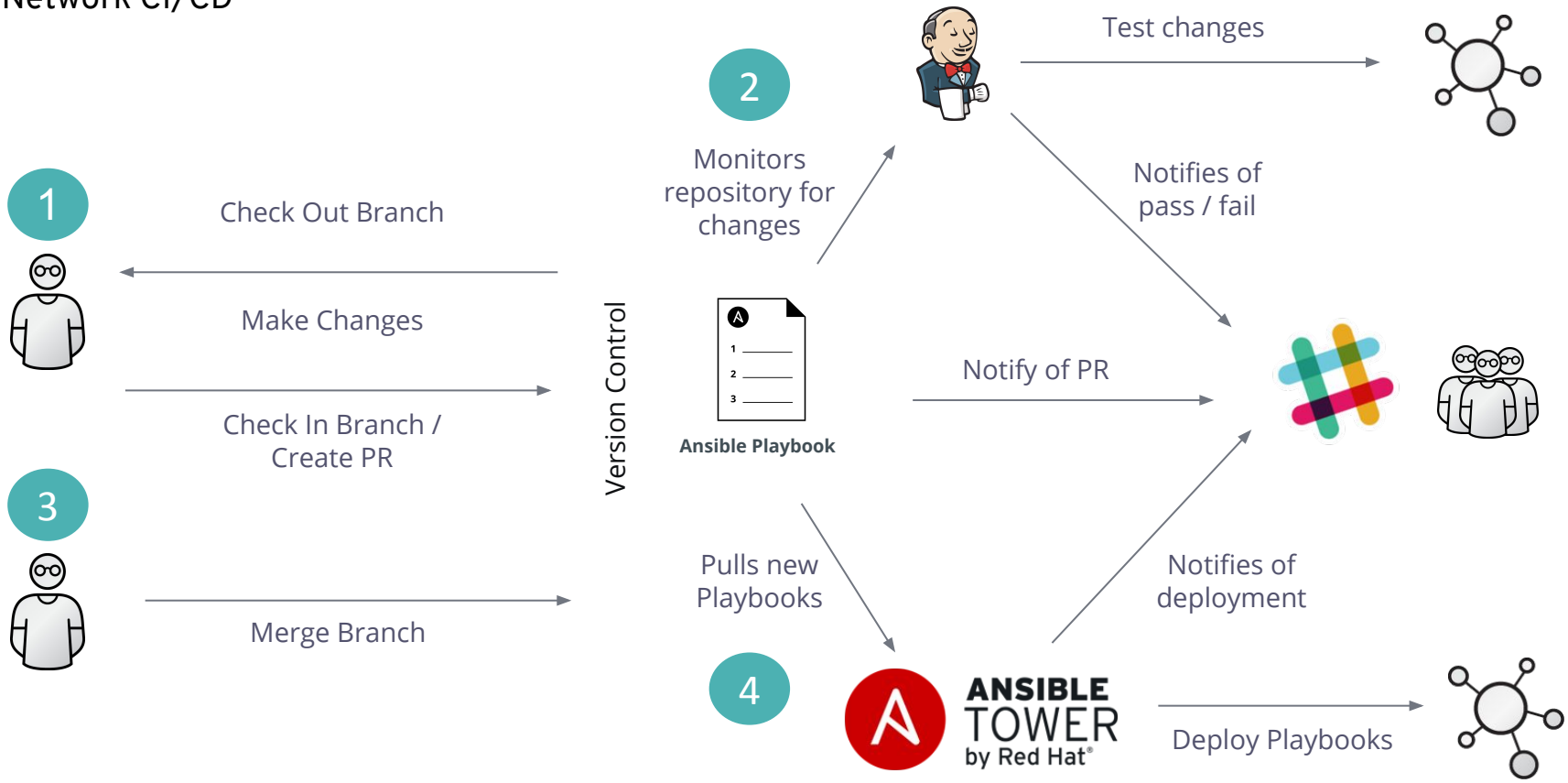
```
- name: Configure PAN security rules
  hosts: panos
  connection: local
  gather_facts: False

  tasks:
    - name: Create security rules
      panos_security_rule:
        ip_address: "{{ ansible_host }}"
        password: "{{ network_password }}"
        rule_name: "Incoming SSH"
        service: "service-ssh"
        description: "Allow Incoming SSH traffic"
        source_zone:
            - untrust
        destination_zone: "trust"
        action: "allow"
```

USE CASES

Ansible for Automation

Network CI/CD

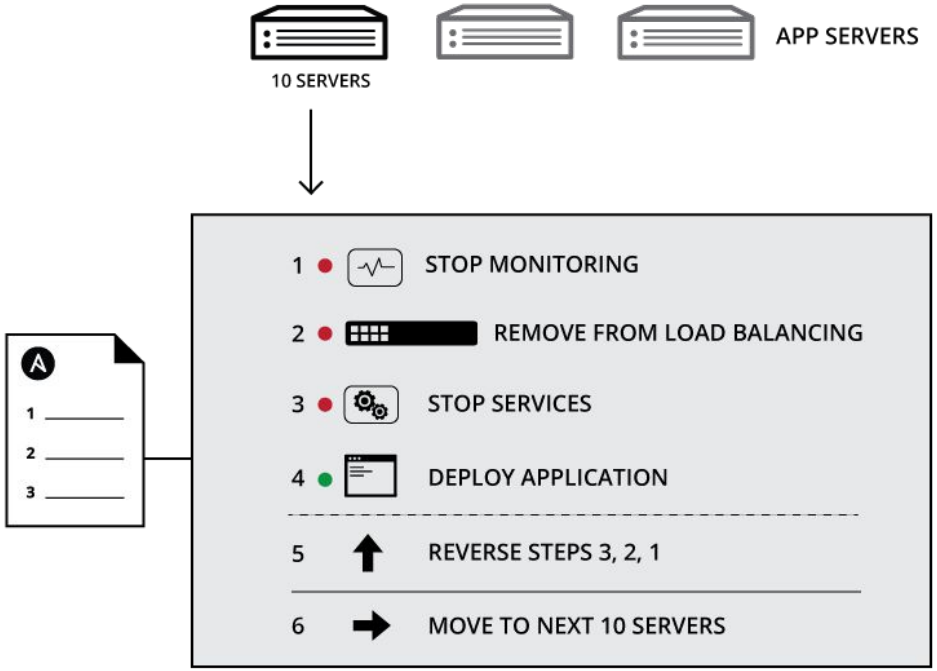


End-to-End Automation

Your applications and systems **are more than just collections of configurations.** They're a finely tuned and **ordered list** of tasks and processes that result in **your working application.**

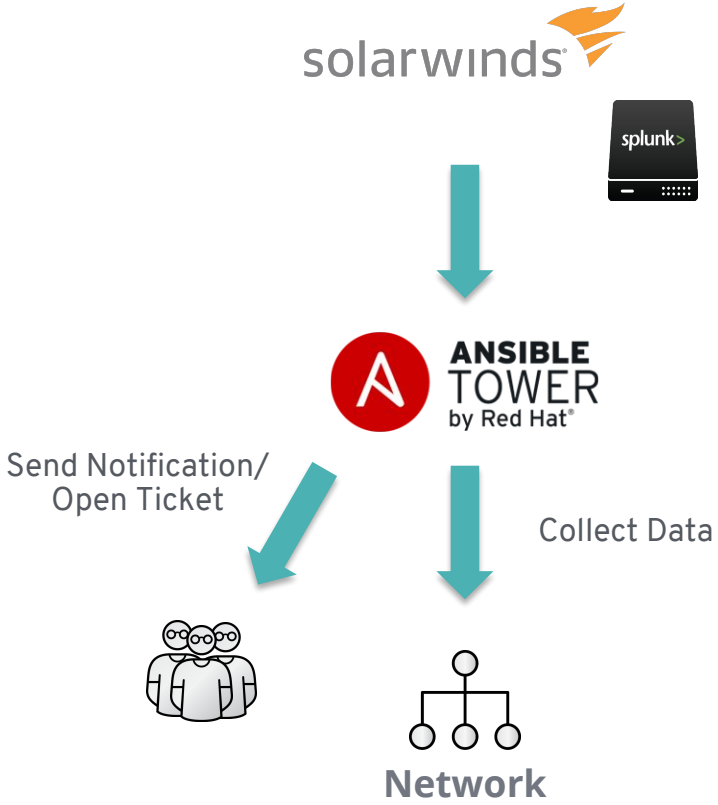
You can do it all with Ansible:

- Provisioning
- App Deployment
- Configuration Management
- Multi-tier Orchestration



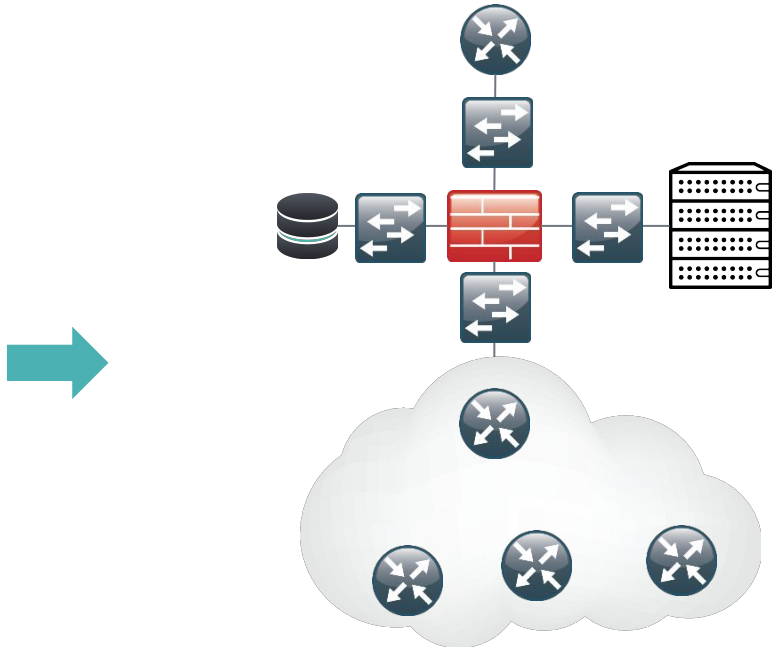
Tier 1 Support Automation

1. Monitoring/Logging Platform detects event and calls the Ansible Tower API
2. Ansible Tower runs a playbook to collect event-specific information
3. Ansible Tower runs a playbook to open a support ticket and/or notify Tier 2 support



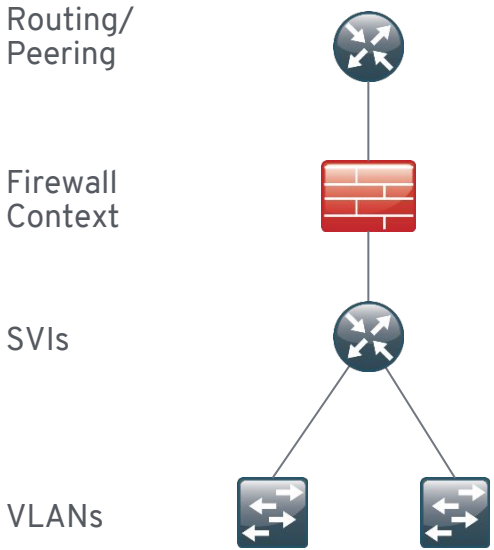
Automating Troubleshooting

```
collect:
  ios_router:
    - show ip ospf neighbors....
    - show bgp summary....
    - show ip ospf route....
    - show ip bgp route....
  nxos_switch:
    - show ip arp....
    - show mac address-table....
  bigip:
    - ....
  junos:
    - ....
  linux:
    - ....
```



Automating Complex Tasks

1. Automate the deployment of the individual components as a workflow.
2. Make that workflow available to operators.
3. Force changes to workflow to maintain compliance
4. Run that workflow on a regular bases to detect any deviation from the original deployment.

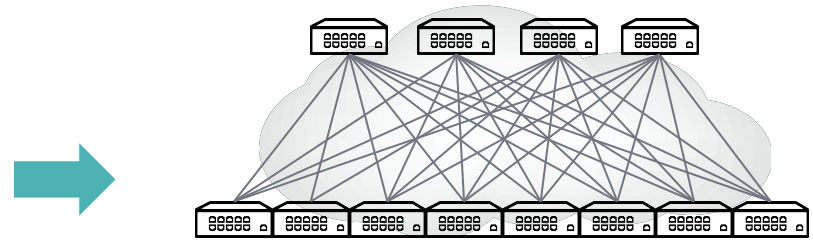


Data Center Fabric Deployment

```
interfaces:
  vtep:
    name: nve1
    source_interface: loopback0
    host_reachability: yes

  control:
    name: loopback0
    address: "{{ control_plane_address }}"

  fabric:
    Ethernet1/1-4:
      name: Ethernet1/1-4
```



FABRIC

Firewall/Load Balancer Updates

```
fw_rules:
    - { rule: "public", src_ip: 0.0.0.0/0, dst_ip: 192.133.160.23/32, dst_port: 32400, proto: tcp, action: allow, comment: app1 }
    - { rule: "public", src_ip: 0.0.0.0/0, dst_ip: 192.133.160.23/32, dst_port: 1900, proto: udp, action: allow, comment: app2  }
    - { rule: "public", src_ip: 0.0.0.0/0, dst_ip: 192.133.160.23/32, dst_port: 3005, proto: tcp, action: allow, comment: app3  }
    - { rule: "public", src_ip: 0.0.0.0/0, dst_ip: 192.133.160.23/32, dst_port: 5353, proto: udp, action: allow, comment: app4  }
```

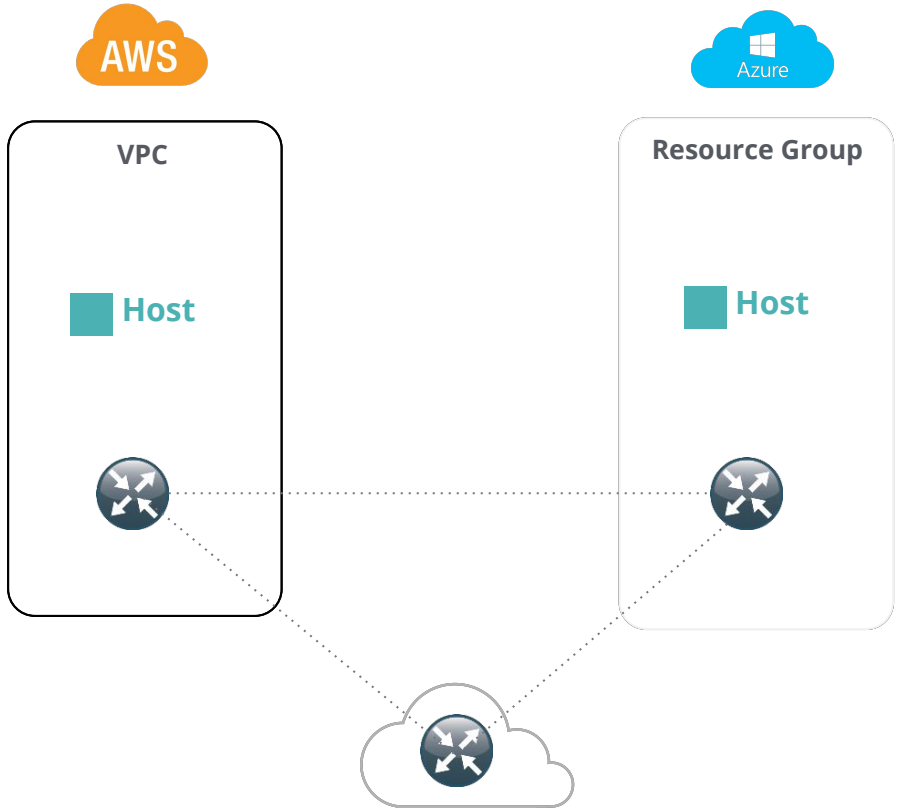
Automate and abstract ACL insertion

```
- name: Insert ASA ACL
     asa_config:
       lines:
         - "access-list {{ item.rule }} extended {{ item.ac
| ipaddr('network') }}{{ item.dst_ip | ipaddr('network') }}{
       provider: "{{ cli }}"
     with_items: "{{ fw_rules }}"
```

```
- name: Create security rules
     panos_security_rule:
       operation: "{{ item.action | default (omit) }}"
       rule_name: "{{ item.comment | default (omit) }}"
       service: "{{ item.dst_port | default (omit) }}"
       description: "{{ item.description | default (omit) }}"
       source_zone: "{{ item.rule | default (omit) }}"
       destination_zone: "{{ item.destination_zone | default (omit) }}"
       action: "{{ item.action | default ('allow') }}"
       commit: "{{ item.comment | default (omit) }}"
```

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.



Workflow Automation

1. Customer makes request from the service catalog
2. Request goes through approval process
3. Service catalog calls Tower API to fulfill request
4. Ansible Tower updates ticket

#ANSIBLEAUTOMATES

redhat. | ANSIBLE