#ANSIBLEAUTOMATES

AUTOMATION FOR NETWORK INFRASTRUCTURE

Pier Luigi Quidacciolu
Senior Solution Architect

AGENDA

1 Network Automation Stumbling Blocks

2 How Ansible Network Works

3 Short Live DEMO

4 Use Cases

MANAGING NETWORKS HASN'T CHANGED IN 30 YEARS.

WHY HASN'T NETWORKING CHANGED?

PEOPLE

- Siloed organizations
- Specific skill sets
- Vendor oriented experience

PRODUCTS

- Siloed technologies
- Monolithic, proprietary platforms
- CLI-only methodologies

BIGGEST CHALLENGE FOR ENTERPRISES: CULTURE!

ANSIBLE

Traditional Network Ops

- Legacy culture
- Risk averse
- “Artisanal” networks

Next-Gen Network Ops

- Risk aware
- Infrastructure as code
- Virtual prototyping / DevOps

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

redhat.

THE ROAD TO AUTOMATION

STANDARDIZE
with Red Hat Ansible Engine

- Standardize Existing Configs
- Standardize New Deployments
- Detect and Reclaim Unstructured Configs

AUTOMATE
with Red Hat Ansible Engine

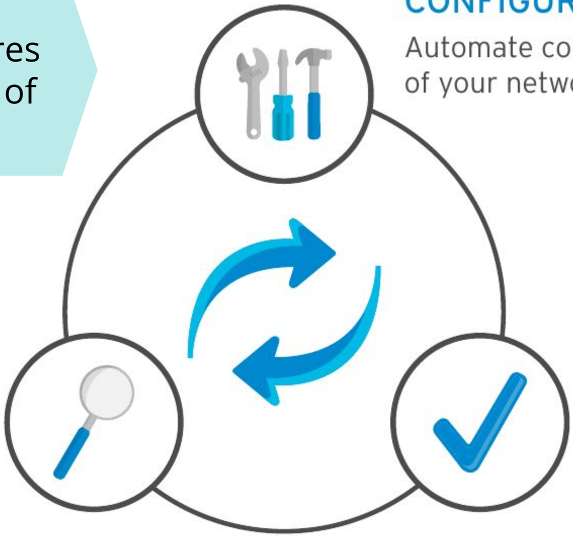
- Automate common tasks
- Make changes across any set of network devices
- Validate that changes were successful

ORCHESTRATE
with Red Hat Ansible Tower

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

Organize → Optimize → Improve

Infrastructure as YAML:
Automate backup & restores
Manage “golden” versions of configurations



CONFIGURE
Automate configuration of your network stack

MONITOR
Continuously check for network configuration drift

VALIDATE
Test and validate your existing network state

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

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

PRODUCTION-GRADE AUTOMATION TECHNOLOGY

SINGLE INTERFACE FOR YOUR ENTIRE NETWORK
Automate everything with support for 50 platforms and 700+ modules.

NETWORK-SPECIFIC ROLES
Simplify network operations with predefined, preinstalled automation.

ROLE-BASED ACCESS CONTROL (RBAC)
Specify access by people, processes, and devices from Ansible Tower.

DYNAMIC INVENTORY CAPABILITIES
Connect to any data source in your network to build an inventory.

WORKFLOWS AND SCHEDULING
Organize tasks and schedule playbooks to run at a specific time.

RESTFUL API
Send and receive messages and instructions from other tools.

VALUE OF AUTOMATION

ANSIBLE

GROW REVENUE

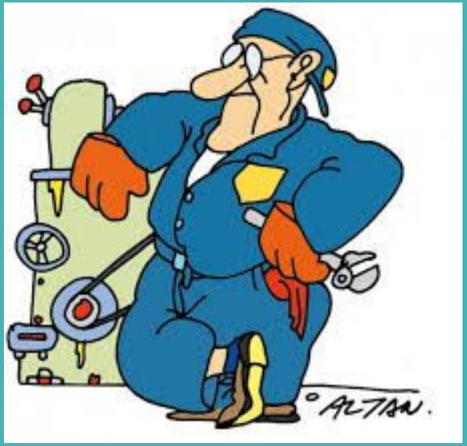
- Faster time to market, with more security and risk mitigation.

- Reduce lock-in, reduce manual tasks and become proactive instead of reactive.

REDUCE COST

redhat.

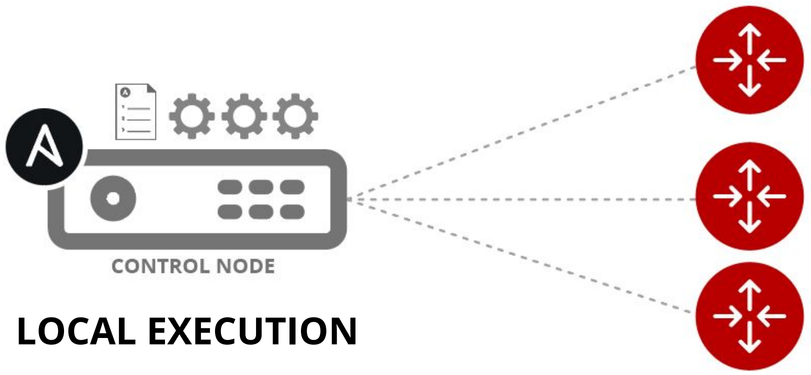
HOW DOES IT WORK?



HOW DOES NETWORK AUTOMATION WORK?

ANSIBLE

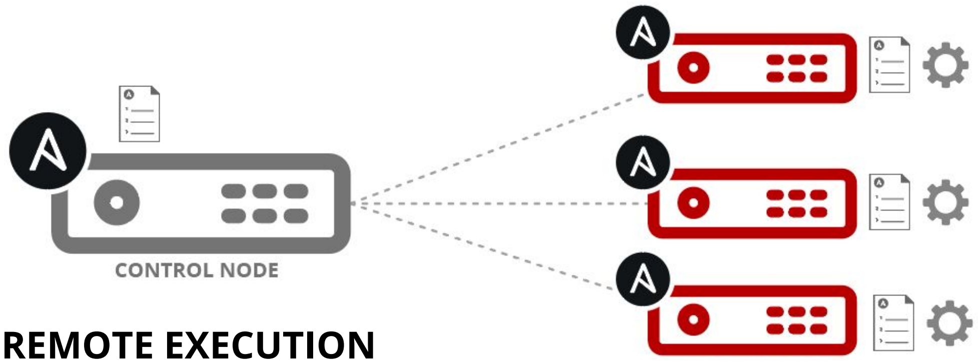
Python code is executed locally on the control node



LOCAL EXECUTION

NETWORKING DEVICES

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



REMOTE EXECUTION

LINUX HOSTS

redhat.

PLAYBOOK EXAMPLE: RHEL

```yaml
---
- 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
```

PLAYBOOK EXAMPLE: F5

```yaml
---
- 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"
```

INVENTORY: vyos_inventory

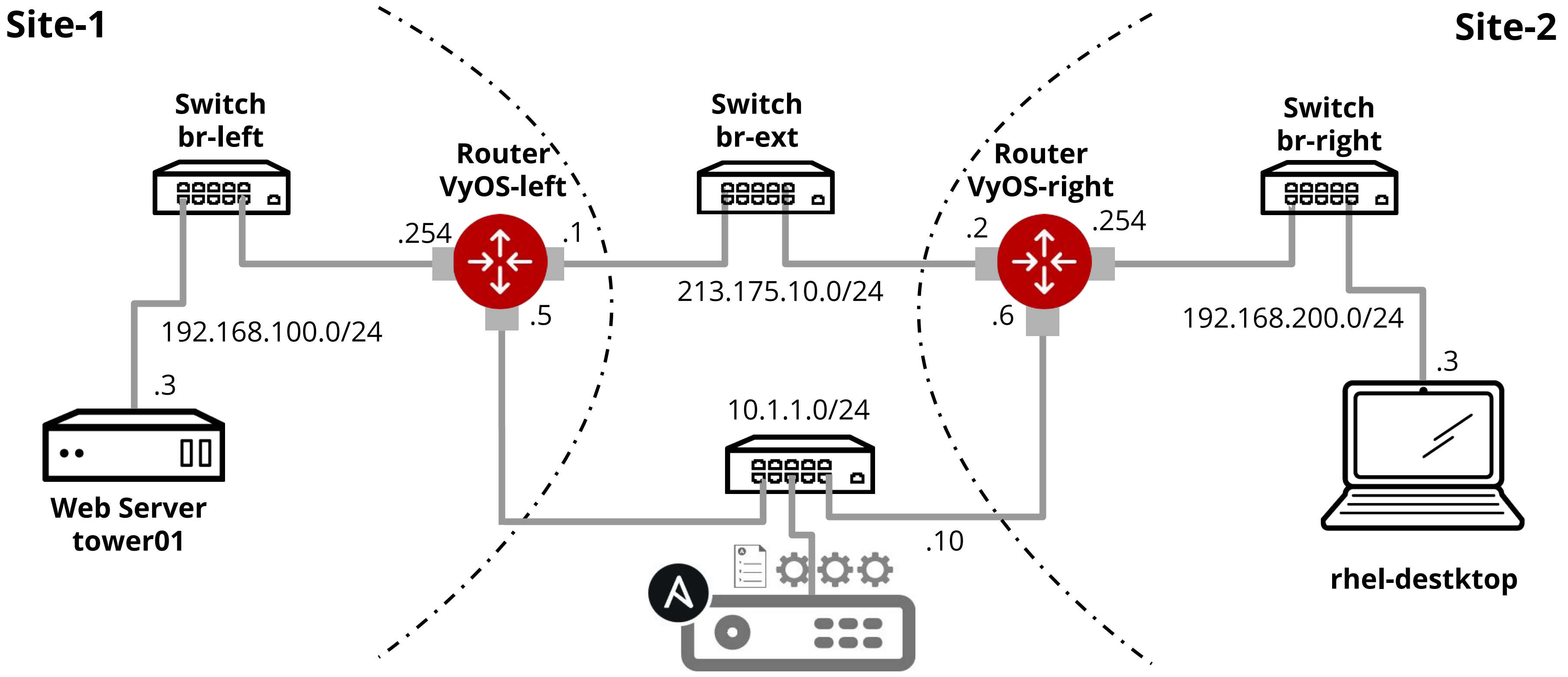
```
[leaves]
leaf01 ansible_host=10.1.1.5
leaf02 ansible_host=10.1.1.6

[leaves:vars]
vyos_connection: network_cli
ansible_network_os=vyos
ansible_user=vyos

[spines]
spine01 ansible_host=10.16.10.13
spine02 ansible_host=10.16.10.14

[spines:vars]
ansible_network_os=vyos
ansible_user=my_vyos_user
```

```
[network:children]
leaves
spines

[servers]
server01
ansible_host=10.16.10.15
server02
ansible_host=10.16.10.16

[datacenter:children]
leaves
spines
servers
```

DEMO: VyOS – IpSEC configuration playbook

```
- name: Network IPSEC configuration on VyOS
  connection: network_cli
  gather_facts: false
  hosts: all
  vars:
    vpn_ipsec_to_edit: "vpn ipsec site-to-site peer {{ peer_ip }}"
  tasks:
  - name: configure ipsec
    vyos_config:
      lines:
        - set system host-name {{ inventory_hostname }}
        - set vpn ipsec ipsec-interfaces interface {{ ipsec_if }}
        # IKE configuration
        - set vpn ipsec ike-group {{ ike_grp }} proposal 1
        - set vpn ipsec ike-group {{ ike_grp }} proposal 1 encryption {{ encr_type }}
        - set vpn ipsec ike-group {{ ike_grp }} proposal 1 hash {{ sha_type }}
        - set vpn ipsec ike-group {{ ike_grp }} lifetime {{ ike_lifetime }}
        # ESP configuration
        - set vpn ipsec esp-group {{ esp_grp }} proposal 1
        - set vpn ipsec esp-group {{ esp_grp }} proposal 1 encryption {{ encr_type }}
        - set vpn ipsec esp-group {{ esp_grp }} proposal 1 hash {{ sha_type }}
        - set vpn ipsec esp-group {{ esp_grp }} lifetime {{ esp_lifetime }}
        # Define connection
        - set {{ vpn_ipsec_to_edit }} authentication mode pre-shared-secret
        - set {{ vpn_ipsec_to_edit }} authentication pre-shared-secret {{ ipsec_secret }}
        - set {{ vpn_ipsec_to_edit }} default-esp-group {{ esp_grp }}
        - set {{ vpn_ipsec_to_edit }} ike-group {{ ike_grp }}
        - set {{ vpn_ipsec_to_edit }} local-address {{ local_ip }}
        - set {{ vpn_ipsec_to_edit }} tunnel 1 local prefix {{ local_tunnel_prefix }}
        - set {{ vpn_ipsec_to_edit }} tunnel 1 remote prefix {{ remote_tunnel_prefix }}
      save: yes
```

DEMO ENVIRONMENT



DEMO: VyOS – IpSEC Inventory

```
[endpoints]
vyos-left  ansible_host=10.1.1.5 local_ip=213.175.10.1 peer_ip=213.175.10.2 ipsec_if=eth0
        local_tunnel_prefix=192.168.100.0/24 remote_tunnel_prefix=192.168.200.0/24
vyos-right ansible_host=10.1.1.6 local_ip=213.175.10.2 peer_ip=213.175.10.1 ipsec_if=eth0
        local_tunnel_prefix=192.168.200.0/24 remote_tunnel_prefix=192.168.100.0/24

[endpoints:vars]
ansible_network_os=vyos
ansible_user=vyos
encr_type=aes256
sha_type=sha1
ike_lifetime=3600
esp_lifetime=1800
ipsec_secret=roadshow
ike_grp=IKE-1W
esp_grp=ESP-1W
```

DEMO

IPSEC Tunnel

On VyOS devices

USE CASES

Ansible Automation

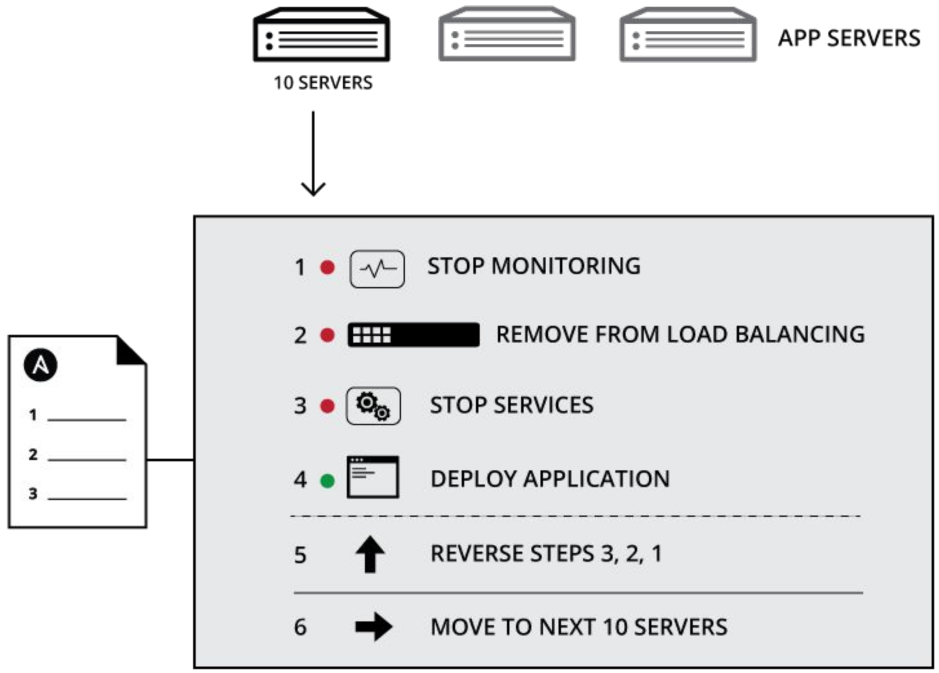
USE CASES

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:

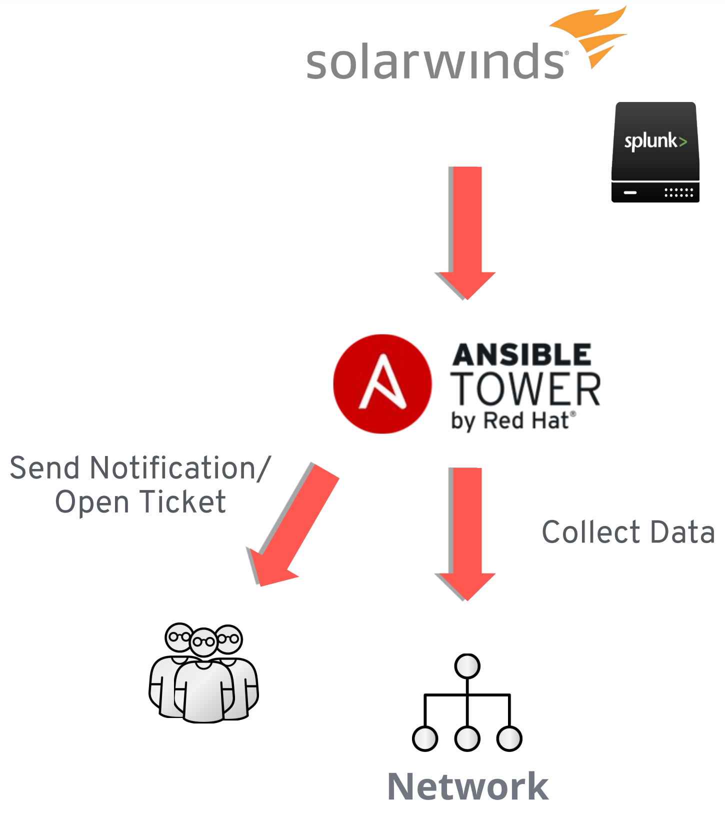
- Provisioning
- App Deployment
- Configuration Management
- Multi-tier Orchestration



USE CASES

Tier 1 Support Automation

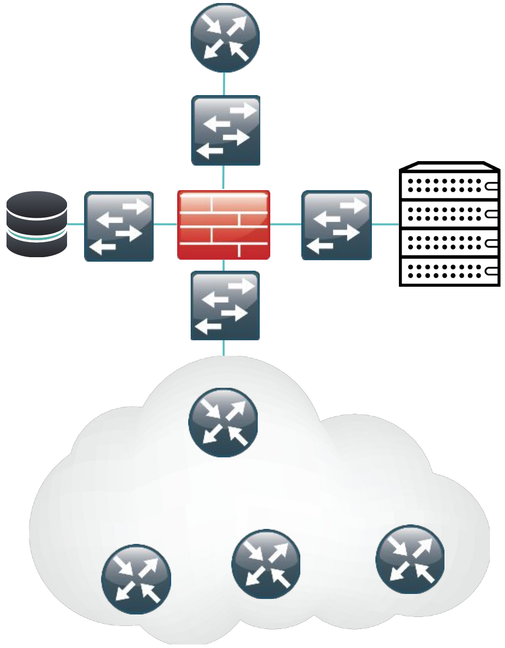
1. Monitoring/Logging Platform detects event and calls the Ansible Tower API
2. Ansible Tower runs a playbook to collect event-specific information
3. Ansible Tower runs a playbook to open a support ticket and/or notify Tier 2 support



USE CASES

Automating Troubleshooting

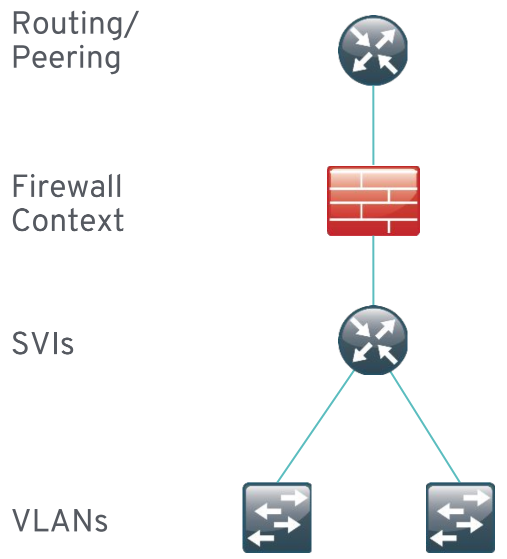
```
collect:
  ios_router:
    - show ip ospf neighbors....
    - show bgp summary....
    - show ip ospf route....
    - show ip bgp route....
  nxos_switch:
    - show ip arp....
    - show mac address-table....
  bigip:
    - ....
  junos:
    - ....
  linux:
    - ....
```



USE CASES

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.



USE CASES

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

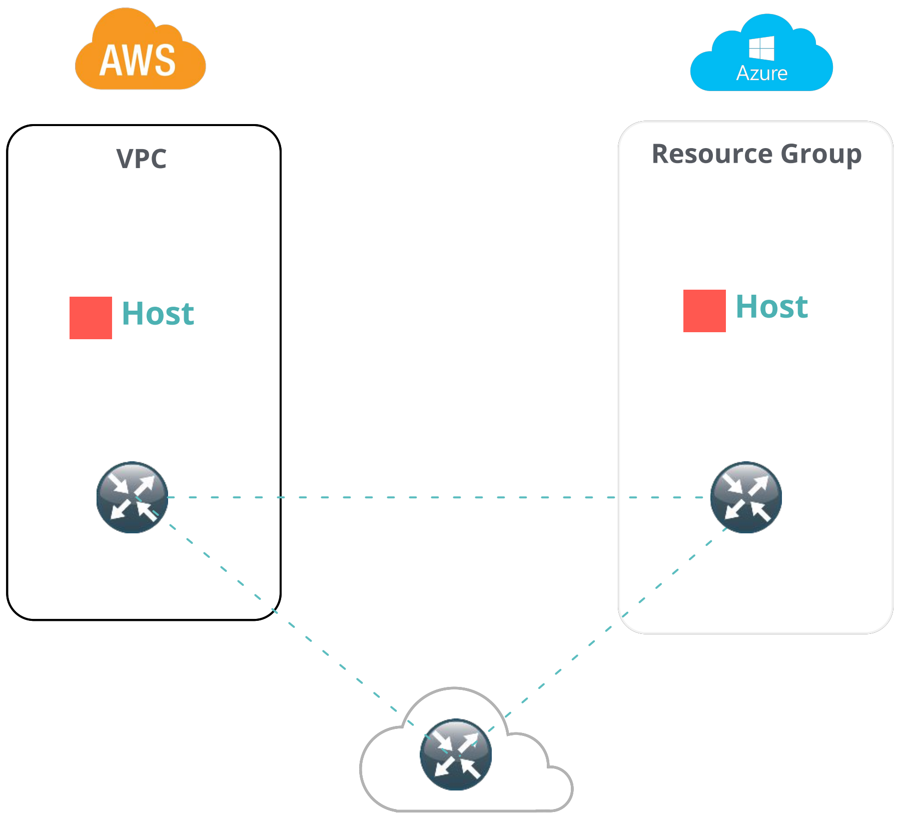
```
- name: Insert ASA ACL
     asa_config:
       lines:
         - "access-list {{ item.rule }} extended {{ item.ac
item.src_ip | ipaddr('network') }}{{ item.dst_ip | ipaddr('n
}}"
       provider: "{{ cli }}"
     with_items: "{{ fw_rules }}"
```

```
- 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) }}"
```

USE CASES

Hybrid Cloud

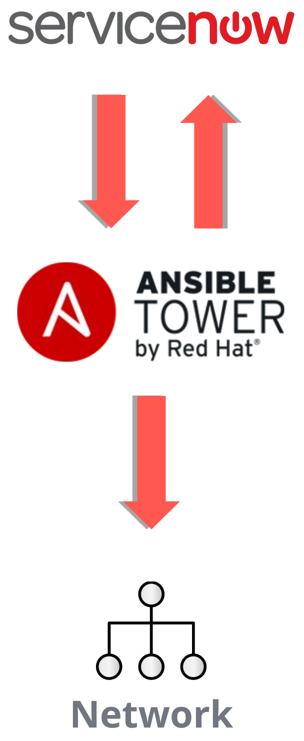
1. Automate the creation of the VPC and network components.
2. Deploy the same routers, load-balancers, and firewalls that you use on-site.
3. Automate the entire network in a uniform way.



USE CASES

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



IT'S EASY TO GET STARTED.

1

Create playbooks that read or check information only.

2

Build simple jobs to replace tedious and unpopular tasks.

3

Apply your team's current knowledge to automation.

ANSIBLE

AUTOMATION EVERYWHERE

redhat. | ANSIBLE