



**Red Hat**

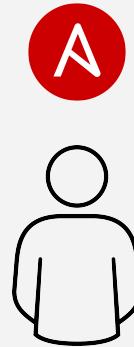
Ansible  
Automation

# Automation for everyone

**J.R. Morgan**  
Solutions Architect, Public Sector SIs

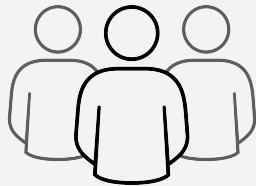


**Red Hat**



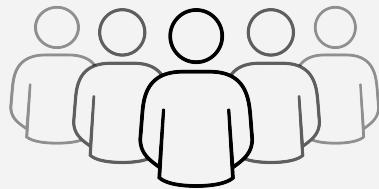
Automation happens when one person meets a  
problem they never want to solve again

A



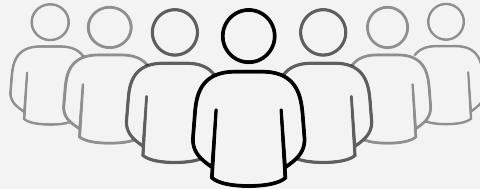
ACCELERATE

A



INTEGRATE

A



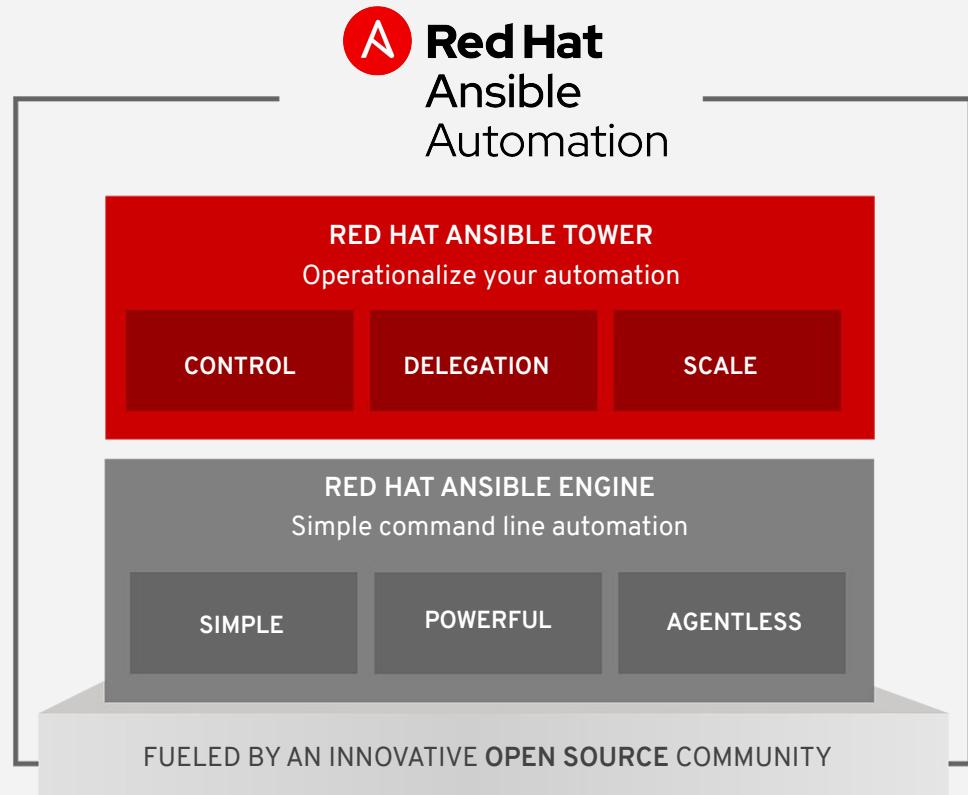
COLLABORATE

# What is Ansible Automation?

Ansible Automation is the enterprise framework for automating across IT operations.

Ansible Engine runs Ansible Playbooks, the automation language that can perfectly describe an IT application infrastructure.

Ansible Tower allows you **scale** IT automation, manage complex deployments and speed productivity.



# Why Ansible?



## Simple

Human readable automation

No special coding skills needed

Tasks executed in order

Usable by every team

**Get productive quickly**



## Powerful

App deployment

Configuration management

Workflow orchestration

Network automation

**Orchestrate the app lifecycle**



## Agentless

Agentless architecture

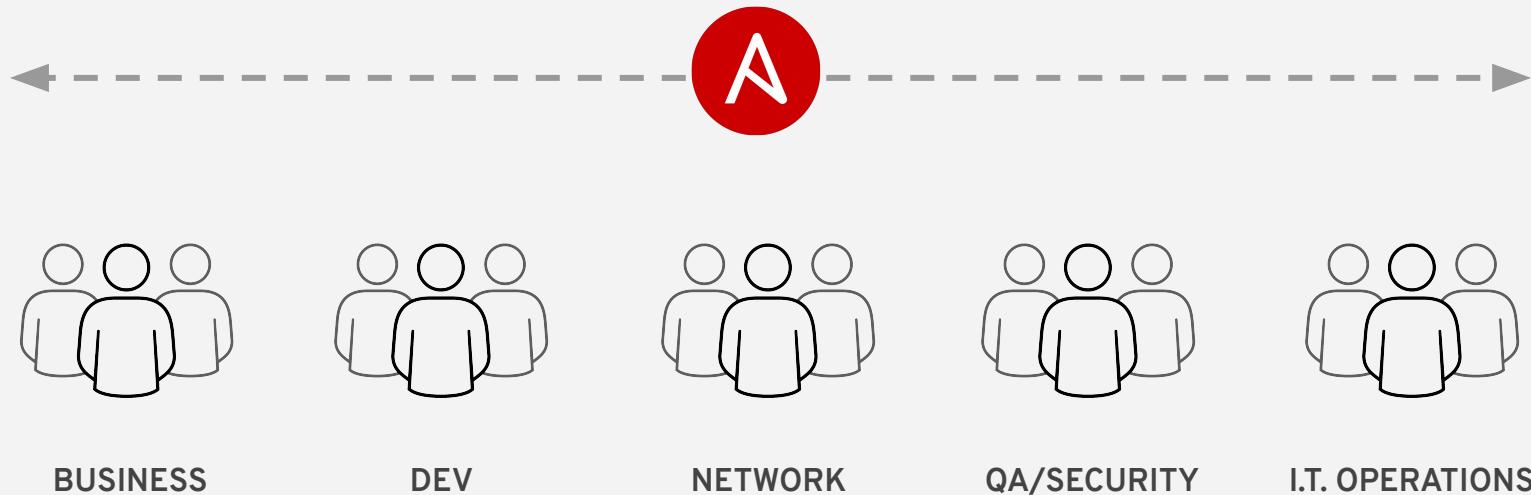
Uses OpenSSH & WinRM

No agents to exploit or update

Get started immediately

**More efficient & more secure**

# Ansible Automation works across teams



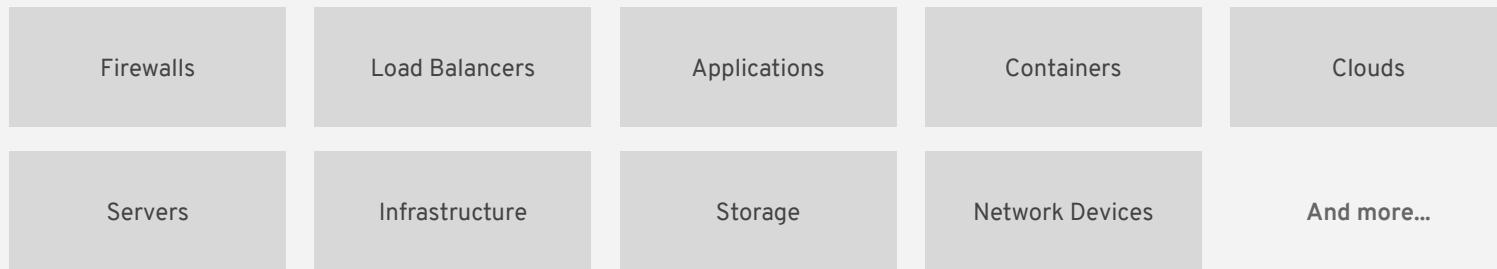
# What can I do using Ansible?

Automate the deployment and management of your entire IT footprint.

Do this...



On these...



# Ansible automates technologies you use

Time to automate is measured in minutes

Cloud	Virt & Container	Windows	Network	Devops	Monitoring
AWS	Docker	ACLs	Arista	Jira	Dynatrace
Azure	VMware	Files	A10	GitHub	Airbrake
Digital Ocean	RHV	Packages	Cumulus	Vagrant	BigPanda
Google	OpenStack	IIS	Bigswitch	Jenkins	Datadog
OpenStack	OpenShift	Regedits	Cisco	Bamboo	LogicMonitor
Rackspace	+more	Shares	Cumulus	Atlassian	Nagios
+more		Services	Dell	Subversion	New Relic
Operating Systems	Storage	Configs	F5	Slack	PagerDuty
		Users	Juniper	Hipchat	Sensu
Rhel And Linux	Netapp	Domains	Palo Alto	+more	StackDriver
Unix	Red Hat Storage	+more	OpenSwitch		Zabbix
Windows	Infinidat		+more		+more
+more	+more				



**Red Hat**  
Ansible  
Engine

# The language of automation

# Red Hat Ansible Engine

## 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

Orchestration plays well with others: ServiceNow, Infoblox, AWS, Terraform, Cisco ACI and more



```
---
```

- **name:** install and start apache

**hosts:** web

**become:** yes

**vars:**

- http\_port:** 80

**tasks:**

- **name:** httpd package is present

- yum:**

  - name:** httpd
  - state:** latest

- **name:** latest index.html file is present

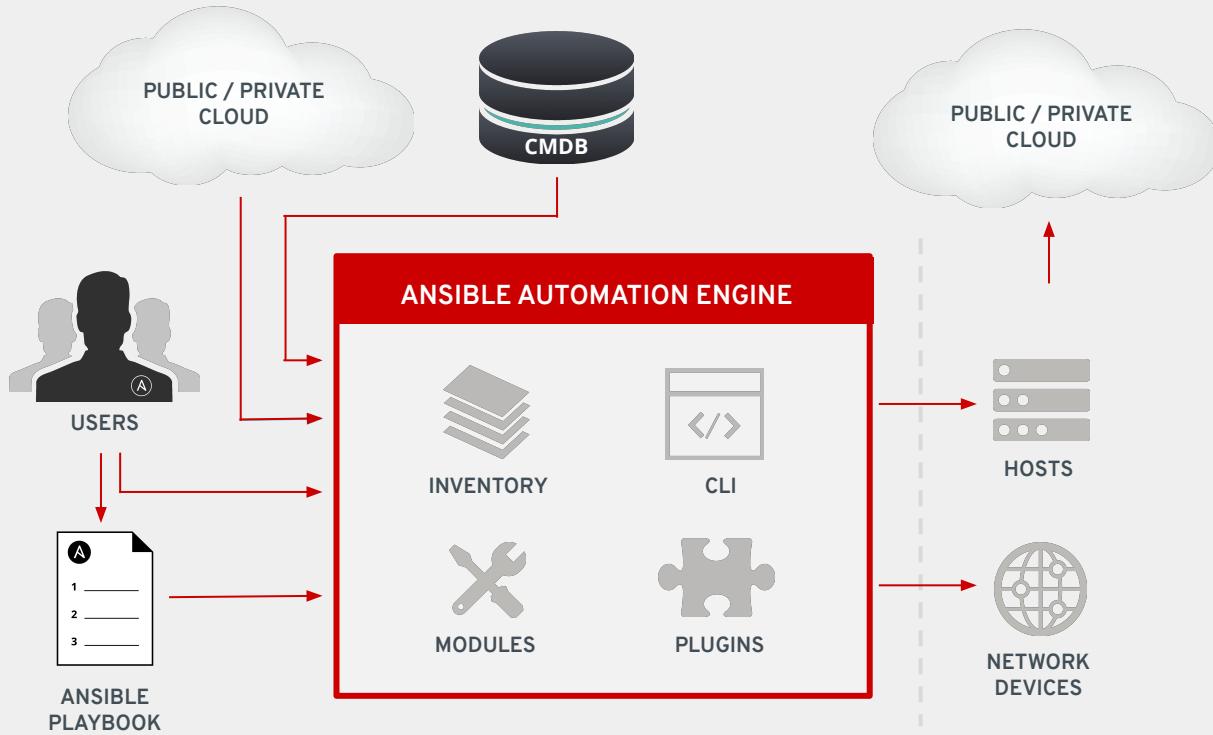
- copy:**

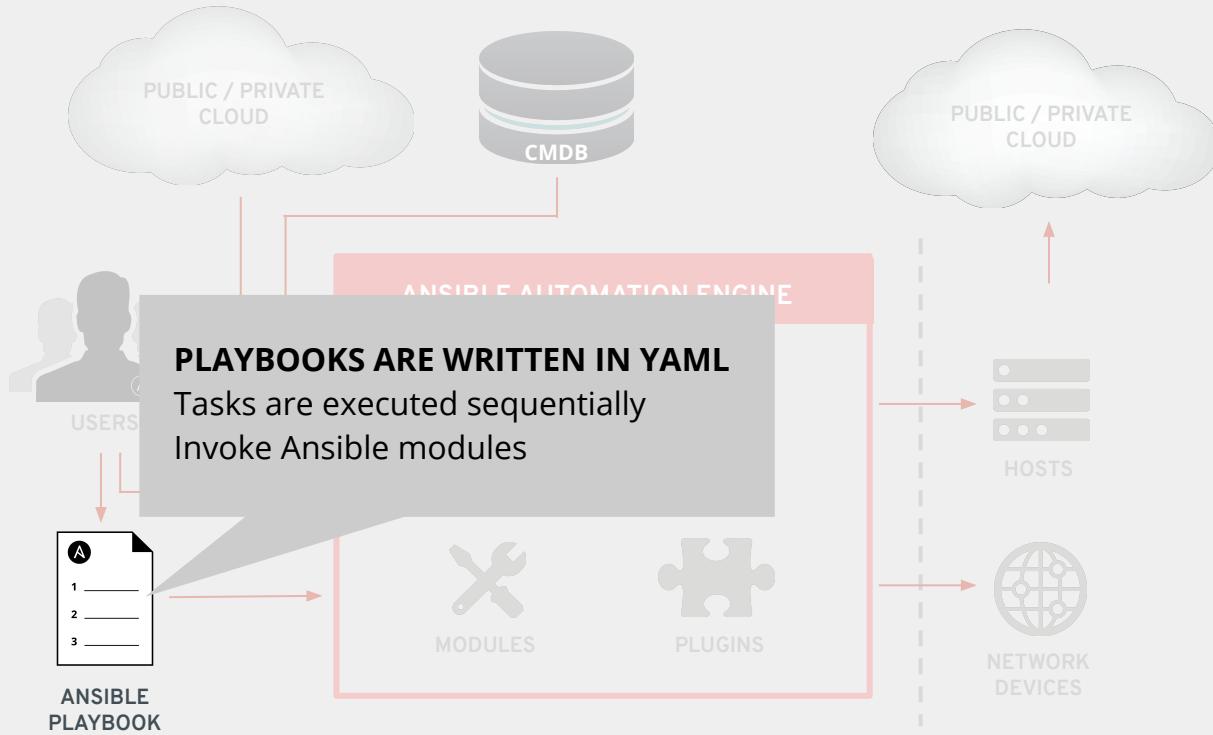
  - src:** files/index.html
  - dest:** /var/www/html/

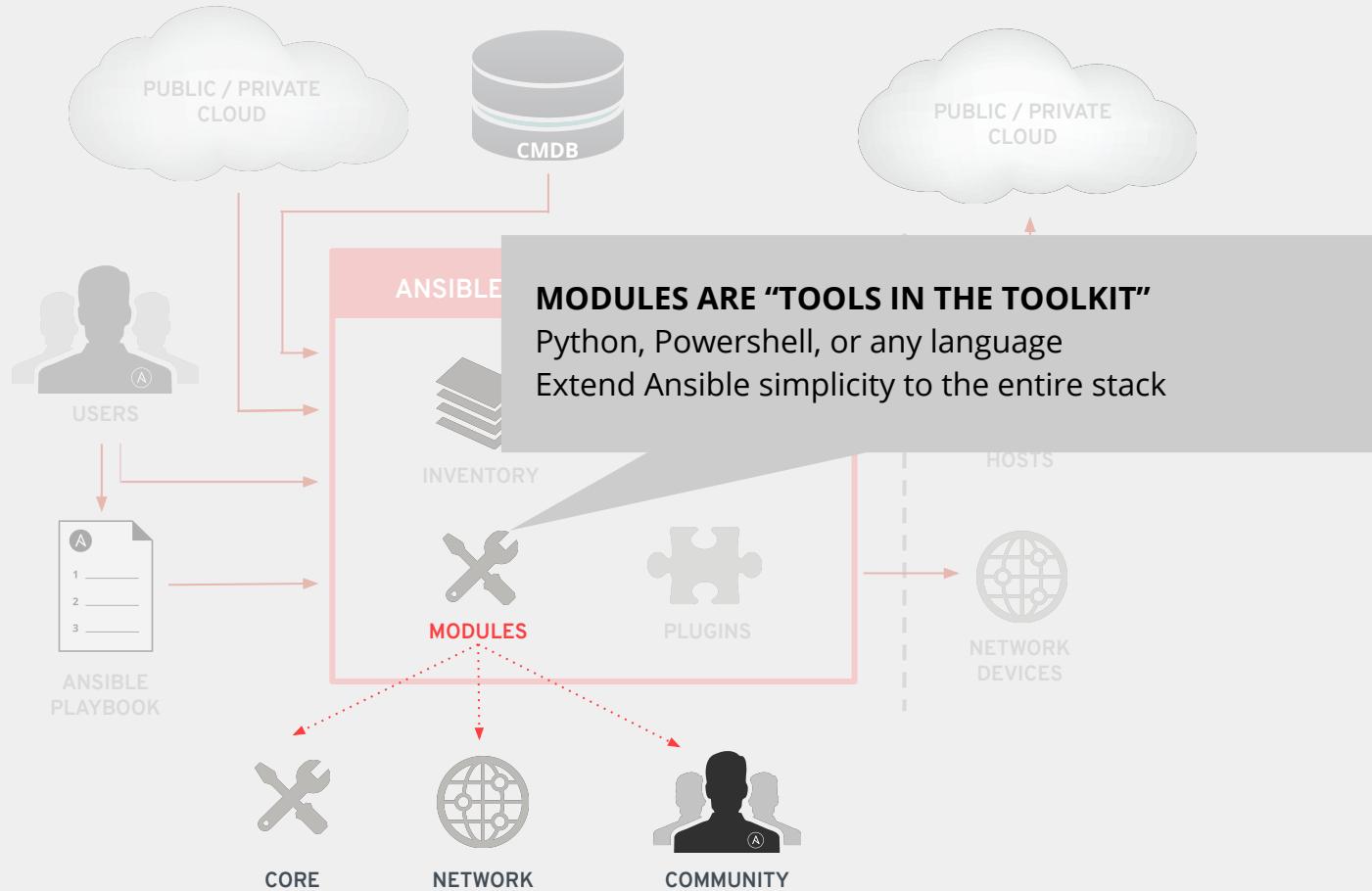
- **name:** httpd is started

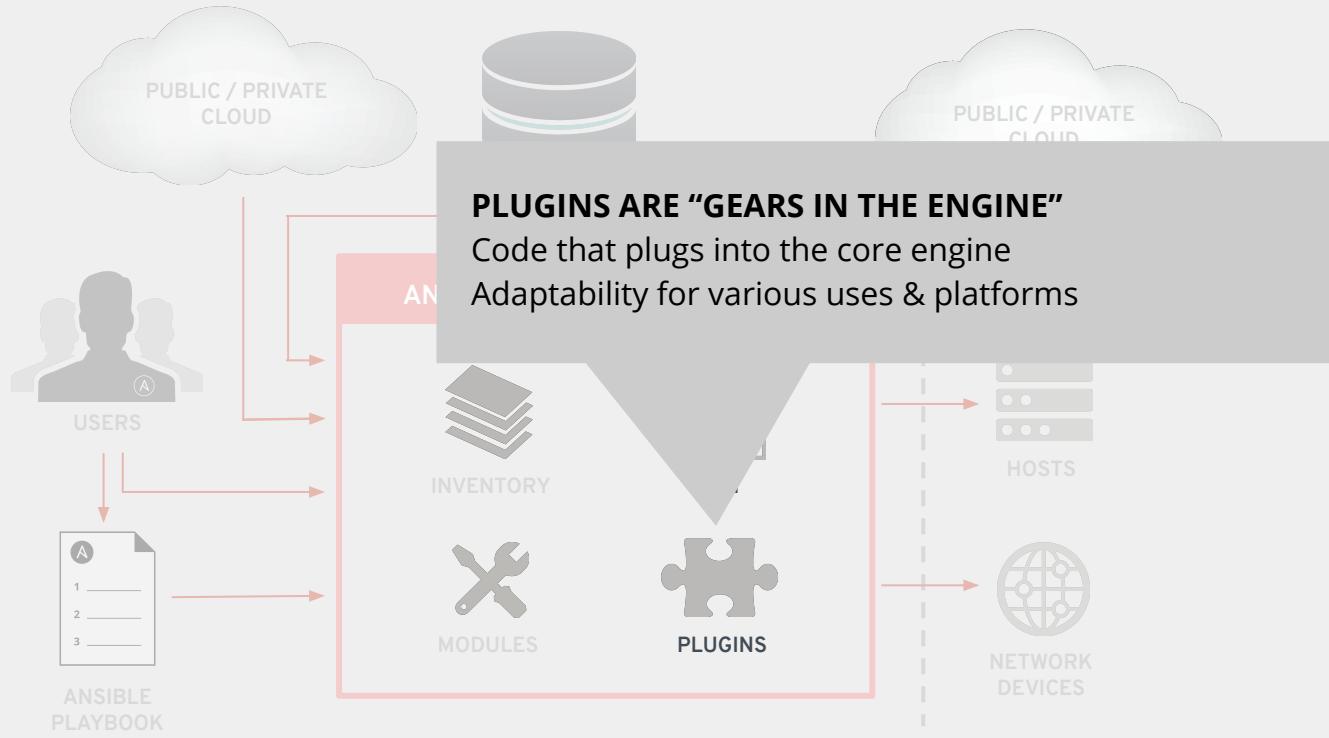
- service:**

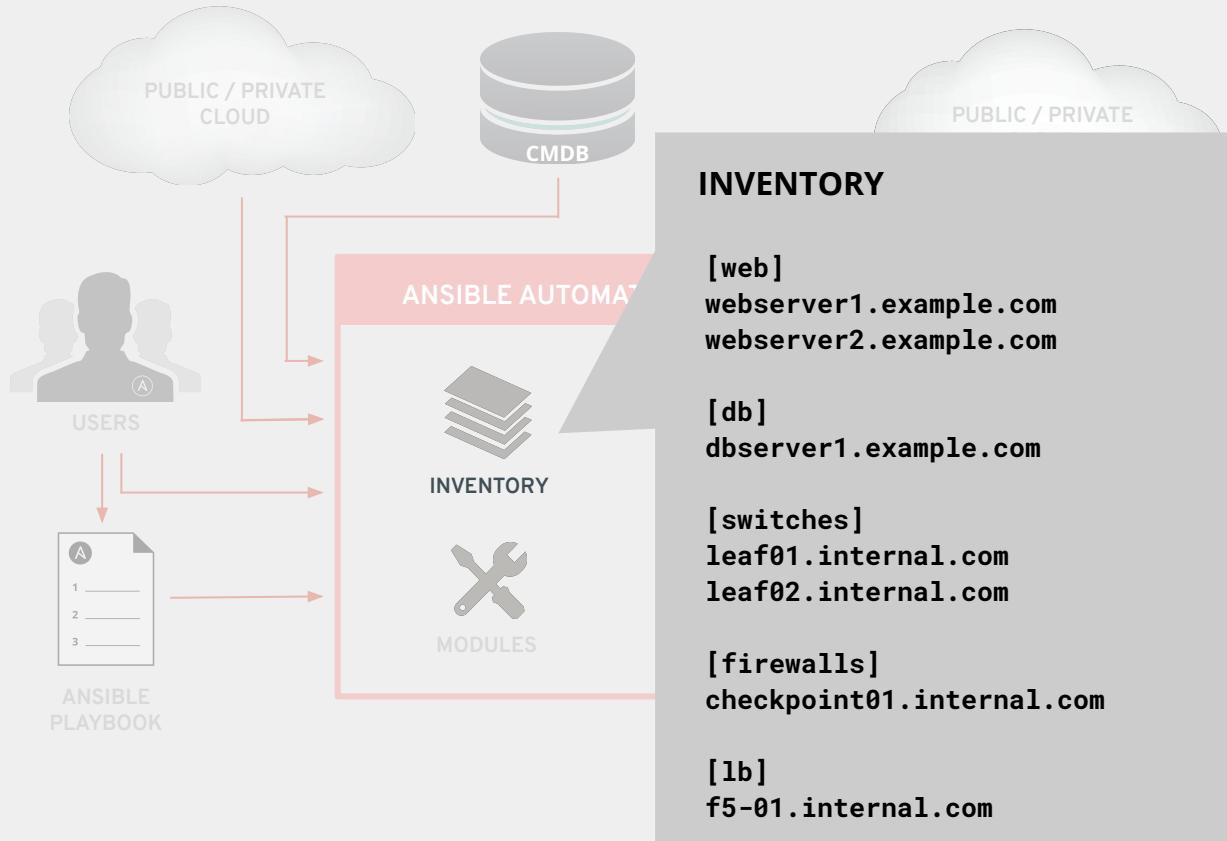
  - name:** httpd
  - state:** started

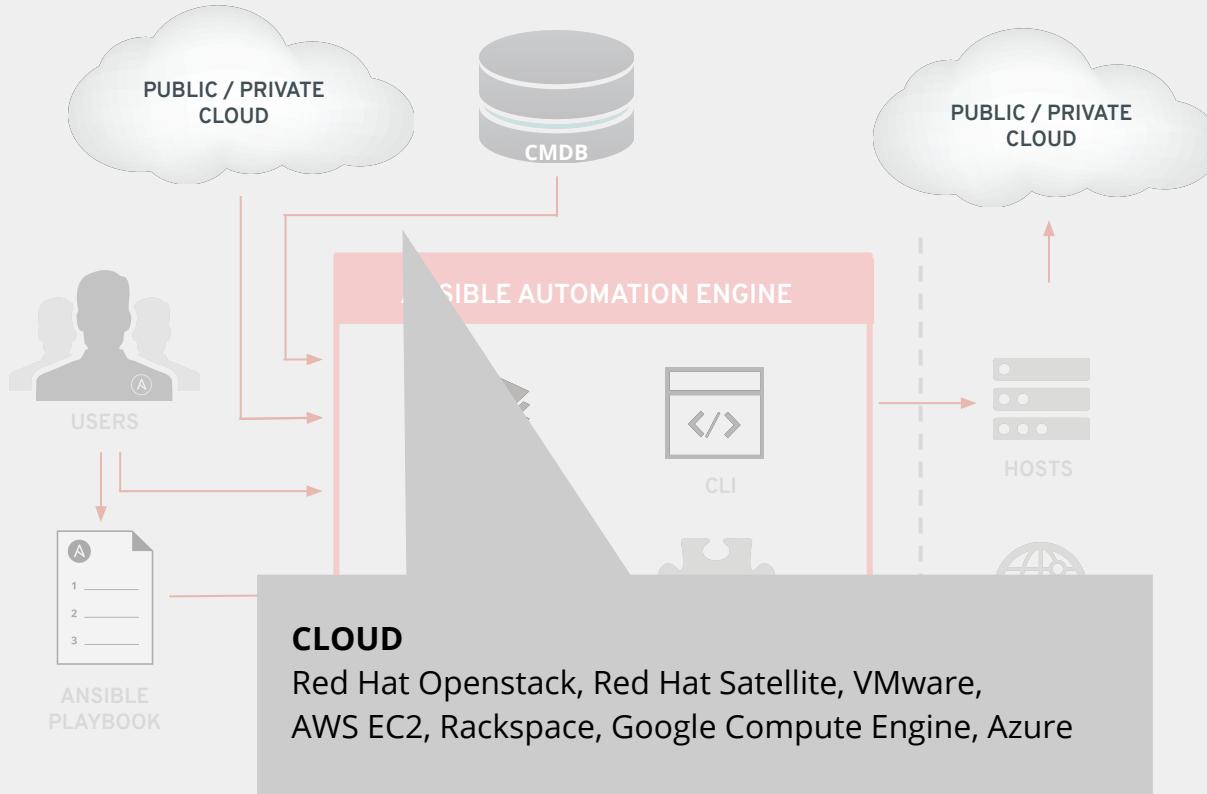


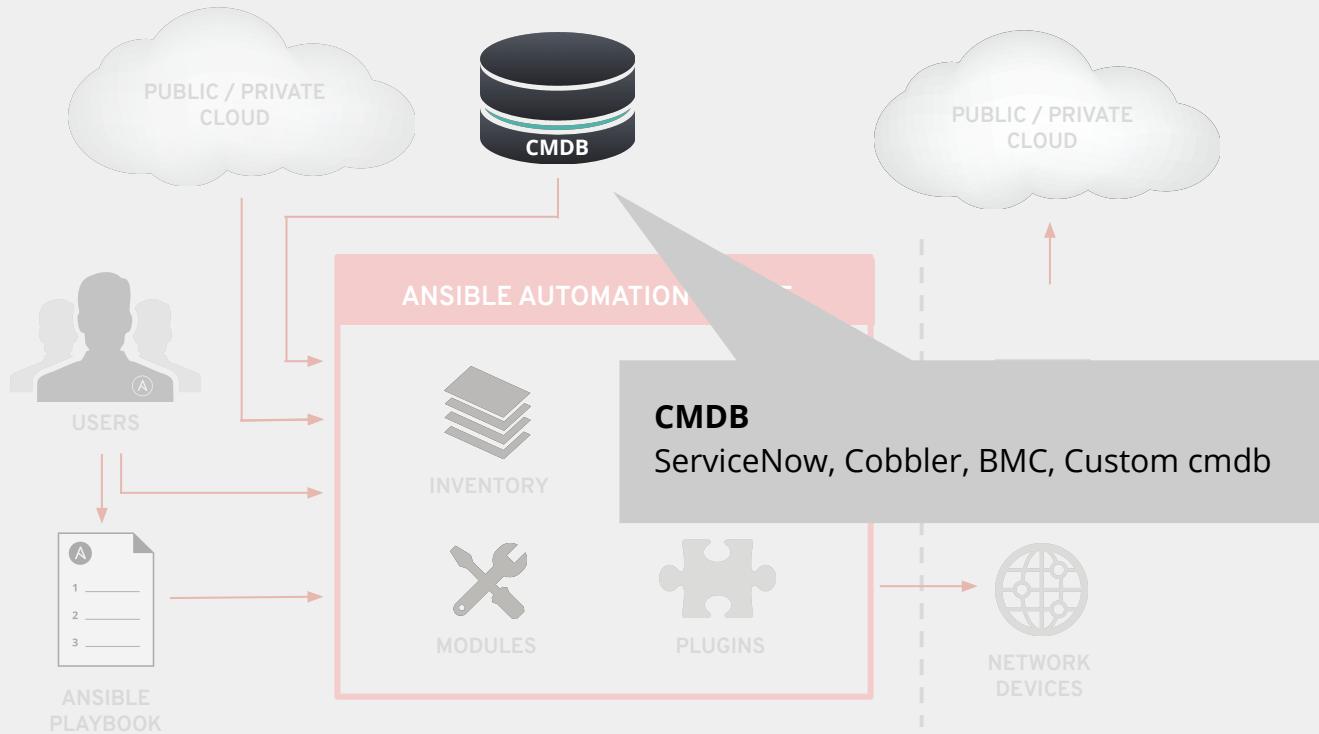


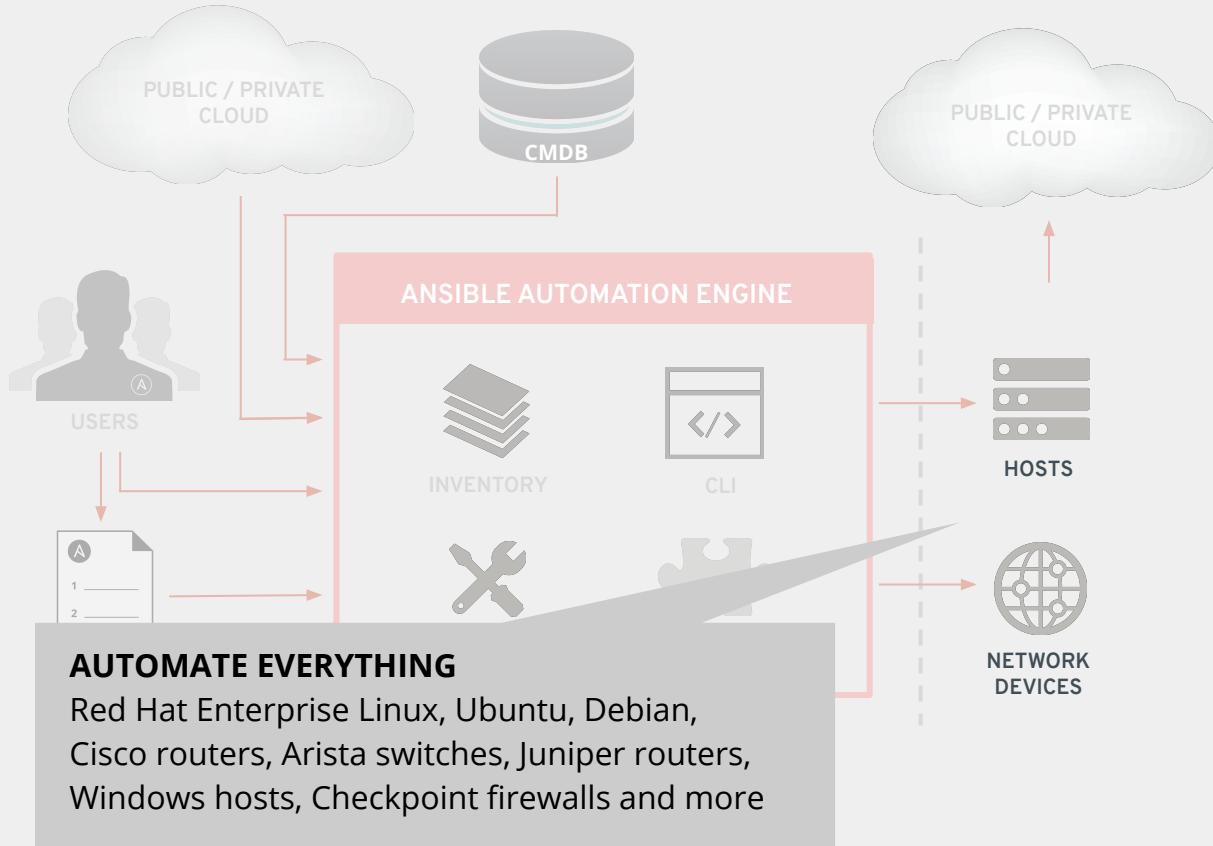












## Playbook examples:

### GITHUB

[github.com/ansible/ansible-examples](https://github.com/ansible/ansible-examples)

### LAMP + HAProxy + NAGIOS

[github.com/ansible/ansible-examples/tree/master/lamp\\_haproxy](https://github.com/ansible/ansible-examples/tree/master/lamp_haproxy)

### WINDOWS

[github.com/ansible/ansible-examples/tree/master/windows](https://github.com/ansible/ansible-examples/tree/master/windows)

### SECURITY COMPLIANCE

[github.com/ansible/ansible-lockdown](https://github.com/ansible/ansible-lockdown)

### NETWORK AUTOMATION

[ansible.com/linklight](https://ansible.com/linklight)

[github.com/network-automation](https://github.com/network-automation)

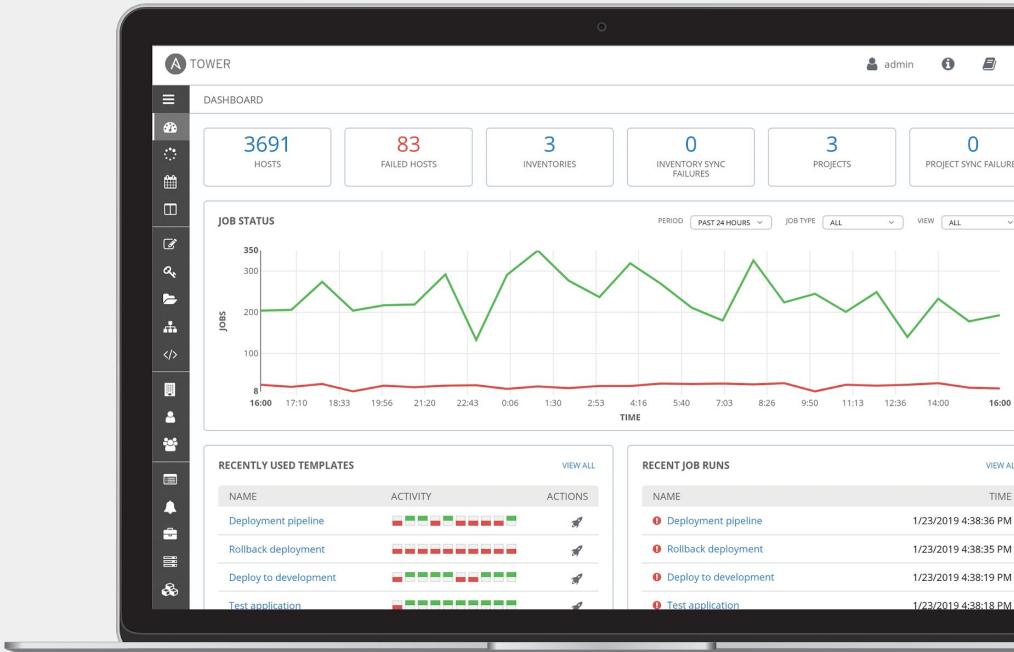


# Automation across the enterprise

# What is Ansible Tower?

Ansible Tower is a UI and RESTful API allowing you to scale IT automation, manage complex deployments and speed productivity.

- Role-based access control
- Deploy entire applications with push-button deployment access
- All automations are centrally logged
- Powerful workflows match your IT processes



# Red Hat Ansible Tower

## RBAC

Allow restricting playbook access to authorized users. One team can use playbooks in check mode (read-only) while others have full administrative abilities.

## Push button

An intuitive user interface experience makes it easy for novice users to execute playbooks you allow them access to.

## RESTful API

With an API first mentality every feature and function of Tower can be API driven. Allow seamless integration with other tools like ServiceNow and Infoblox.

## Workflows

Ansible Tower's multi-playbook workflows chain any number of playbooks, regardless of whether they use different inventories, run as different users, run at once or utilize different credentials.

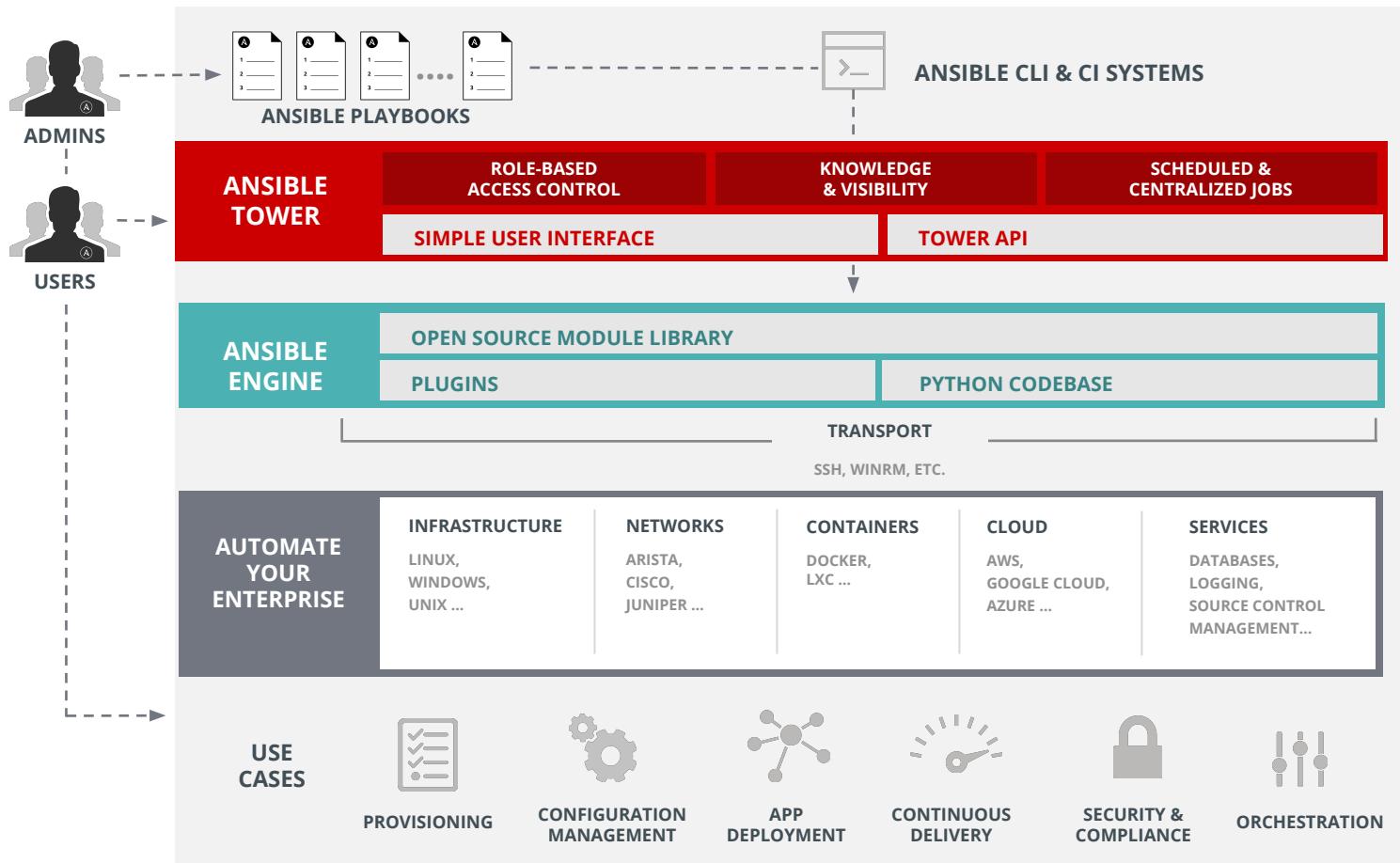
## Enterprise integrations

Integrate with enterprise authentication like TACACS+, RADIUS, Azure AD. Setup token authentication with OAuth 2. Setup notifications with PagerDuty, Slack and Twilio.

## Centralized logging

All automation activity is securely logged. Who ran it, how they customized it, what it did, where it happened - all securely stored and viewable later, or exported through Ansible Tower's API.







# Red Hat Ansible Tower

FEATURE OVERVIEW:

## Control

The dashboard provides a high-level overview of the Ansible Tower environment. Key metrics include 3691 hosts, 83 failed hosts, 3 inventories, 0 inventory sync failures, 3 projects, and 0 project sync failures. A prominent feature is the 'JOB STATUS' chart, which tracks the number of jobs over time, showing a fluctuating pattern between 100 and 350 jobs per hour. Below the chart, a list of recently used templates is displayed, each with activity indicators and an 'Actions' button. The 'RECENT JOB RUNS' section lists completed jobs with their names, times, and status icons.

NAME	ACTIVITY	ACTIONS
Deployment pipeline	<span style="color:red;">■</span> <span style="color:green;">■</span>	<a href="#">View</a>
Rollback deployment	<span style="color:red;">■■■■■</span>	<a href="#">View</a>
Deploy to development	<span style="color:green;">■■■■■</span>	<a href="#">View</a>
Test application	<span style="color:red;">■■■■■</span>	<a href="#">View</a>
Deploy database	<span style="color:green;">■■■■■</span>	<a href="#">View</a>

NAME	TIME
Deployment pipeline	1/23/2019 4:38:36 PM
Rollback deployment	1/23/2019 4:38:35 PM
Deploy to development	1/23/2019 4:38:19 PM
Test application	1/23/2019 4:38:18 PM
Deploy database	1/23/2019 4:38:05 PM



# ANSIBLE TOWER FEATURES: YOUR ANSIBLE DASHBOARD

A TOWER

admin

DASHBOARD

3062 HOSTS | 1 FAILED HOSTS | 7 INVENTORIES | 0 INVENTORY SYNC FAILURES | 2 PROJECTS | 0 PROJECT SYNC FAILURES

JOB STATUS

JOBS

TIME

PAST MONTH | ALL | ALL

RECENTLY USED TEMPLATES

VIEW ALL

NAME	ACTIVITY	ACTIONS
Deploy application	███████ ██████████	🔗
Deploy database	██████████ ████████	🔗
Deploy webserver	███████ ██████████	🔗
Deploy ticketing application	██████████ ████████	🔗
Rollback workflow	██████████ ████████	🔗

RECENT JOB RUNS

VIEW ALL

NAME	TIME
Deploy application	1/8/2019 9:42:22 PM
Deploy database	1/8/2019 9:42:20 PM
Deploy webserver	1/8/2019 9:41:06 PM
Deploy webserver	1/8/2019 9:40:45 PM
Deploy application	1/8/2019 9:40:20 PM

# ANSIBLE TOWER FEATURES: JOB STATUS UPDATE

The screenshot shows the Ansible Tower interface for a job named "184 - BACKUP NETWORK CONFIG".

**Job Details:**

- Status: Successful
- Started: 1/8/2019 9:50:26 PM
- Finished: 1/8/2019 9:51:02 PM
- Job Template: BACKUP NETWORK CONFIG
- Job Type: Run
- Launched By: admin
- Inventory: Workshop Inventory
- Project: Workshop Project
- Revision: 23a23b8
- Playbook: network\_backup.yml
- Credential: Workshop Credential
- Instance Group: tower

**Job Log:**

PLAYS: 2 | TASKS: 9 | HOSTS: 1 | ELAPSED: 00:00:36

```
SEARCH
```

```
21 ok: [rtr3 -> 35.183.122.35]
22
23 TASK [CREATE TIMESTAMP DIRECTORY ON ansible] **** 21:50:53
*****
24 changed: [rtr3 -> 35.183.122.35]
25
26 TASK [TRANSFER FILE FROM THIS ANSIBLE HOST TO ansible] **** 21:50:54
*****
27 skipping: [ansible]
28 changed: [rtr2 -> 35.183.122.35]
29 changed: [rtr4 -> 35.183.122.35]
30 changed: [rtr3 -> 35.183.122.35]
31 changed: [rtr1 -> 35.183.122.35]
32
33 PLAY [BACKUP ROUTER CONFIGURATIONS] **** 21:50:56
*****
34
35 TASK [FIND BACKUPS] **** 21:50:56
*****
36 ok: [localhost -> 35.183.122.35]
37
38 TASK [CREATE RESTORE JOB TEMPLATE] **** 21:50:56
*****
39 changed: [localhost]
```



# ANSIBLE TOWER FEATURES: ACTIVITY STREAM

The screenshot shows the Ansible Tower interface with the "ACTIVITY STREAM" page selected. The left sidebar contains various navigation icons. The main area has a header "ACTIVITY STREAM" with a search bar, key filter, and a dropdown set to "All Activity". Below is a table with columns: TIME, INITIATED BY, EVENT, and ACTIONS. The table lists the following activity entries:

TIME	INITIATED BY	EVENT	ACTIONS
1/7/2019 1:43:31 PM	admin	created notification_template Email Results	🔍
1/7/2019 1:43:10 PM	admin	updated notification_template Failure Messages	🔍
1/7/2019 1:43:10 PM	admin	created notification_template Failure Messages	🔍
1/7/2019 1:42:46 PM	admin	updated notification_template Prod Ops Complete	🔍
1/7/2019 1:42:46 PM	admin	created notification_template Prod Ops Complete	🔍
1/7/2019 1:37:08 PM	admin	associated workflow_job_template_node to workflow_job_template_node	🔍
1/7/2019 1:37:08 PM	admin	disassociated workflow_job_template_node from workflow_job_template_node	🔍

# ANSIBLE TOWER FEATURES: MANAGE AND TRACK YOUR INVENTORY

The screenshot shows the Ansible Tower web interface. The top navigation bar includes the 'TOWER' logo, user 'admin', and various system icons. The main title is 'INVENTORIES / Durham / SOURCES / Cloud dev servers'. The central modal window is titled 'Cloud dev servers' and contains the following fields:

- DETAILS** tab is selected.
- \* NAME**: Cloud dev servers
- DESCRIPTION**: sync to AWS development us-ea
- \* SOURCE**: Amazon EC2
- SOURCE DETAILS** section:
  - CREDENTIAL**: AWS dev keys
  - REGIONS**: US East (Ohio)
  - INSTANCE FILTERS**: tag:Name=\*development\*
- ONLY GROUP BY**: (empty field)
- VERBOSITY**: 1 (INFO)
- UPDATE OPTIONS**:
  - Overwrite
  - Overwrite Variables
  - Update on Launch
- SOURCE VARIABLES** section:
  - YAML** tab is selected.
  - Content area shows: 1 ---

At the bottom right of the modal are 'CANCEL' and 'SAVE' buttons.

# ANSIBLE TOWER FEATURES: SCHEDULE JOBS

The screenshot shows the Ansible Tower interface for creating a new schedule. The top navigation bar includes the 'TOWER' logo, user 'admin', and various icons for help, logs, and system control. The current page is 'TEMPLATES / BACKUP NETWORK CONFIG / SCHEDULES / CREATE SCHEDULE'. On the left, a sidebar lists icons for Playbooks, Host Groups, Inventory, Tags, Variables, Roles, Scripts, and Artifacts. The main form is titled 'Daily Network Backup' and contains the following fields:

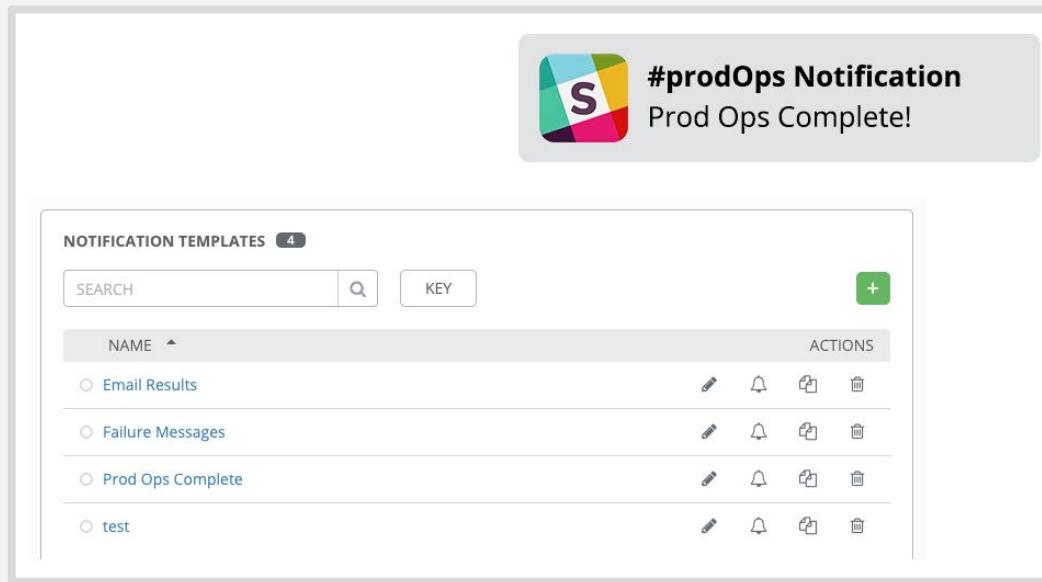
- \* NAME:** Daily Network Backup
- \* START DATE:** 1/09/2019
- \* START TIME (HH24:MM:SS):** 02:00:00
- \* LOCAL TIME ZONE:** America/New\_York
- \* REPEAT FREQUENCY:** Day
- FREQUENCY DETAILS:**
  - \* EVERY:** 1 DAYS
  - \* END:** Never
- SCHEDULE DESCRIPTION:** every day
- OCCURRENCES (Limited to first 10):** 01-09-2019 02:00:00, 01-10-2019 02:00:00, 01-11-2019 02:00:00, 01-12-2019 02:00:00
- DATE FORMAT:** LOCAL TIME ZONE (radio button selected), UTC (radio button unselected)

# ANSIBLE TOWER FEATURES: EXTERNAL LOGGING

The screenshot shows the Ansible Tower web interface with a focus on the Activity Stream. The top navigation bar includes links for TOWER, PROJECTS, INVENTORIES, TEMPLATES, JOBS, and user ADMIN. To the right of the user icon are icons for settings, a checklist, a document, and a power button. Below the navigation is a header bar with 'ACTIVITY STREAM' and a blue heart icon. The main content area is titled 'ACTIVITY STREAM | ALL ACTIVITY'. It features a search bar with 'INITIATED BY' dropdown, a 'SEARCH' input field, and a magnifying glass icon. A 'REFRESH' button and a dropdown menu set to 'All Activity' are also present. A table lists three events:

TIME	INITIATED BY	EVENT	ACTIONS
10/3/2016 5:00:52 PM	admin	created schedule Daily remediation	
10/3/2016 4:51:45 PM	admin	deleted schedule Hourly scan	
10/3/2016 4:51:13 PM	admin	created schedule Hourly scan	

# ANSIBLE TOWER FEATURES: INTEGRATED NOTIFICATIONS



The screenshot shows the Ansible Tower interface for managing notification templates. At the top, there is a preview box displaying a colorful icon and the text "#prodOps Notification Prod Ops Complete!". Below this, the main section is titled "NOTIFICATION TEMPLATES 4". It features a search bar, a key button, and a green plus sign button for creating new templates. The table lists four templates:

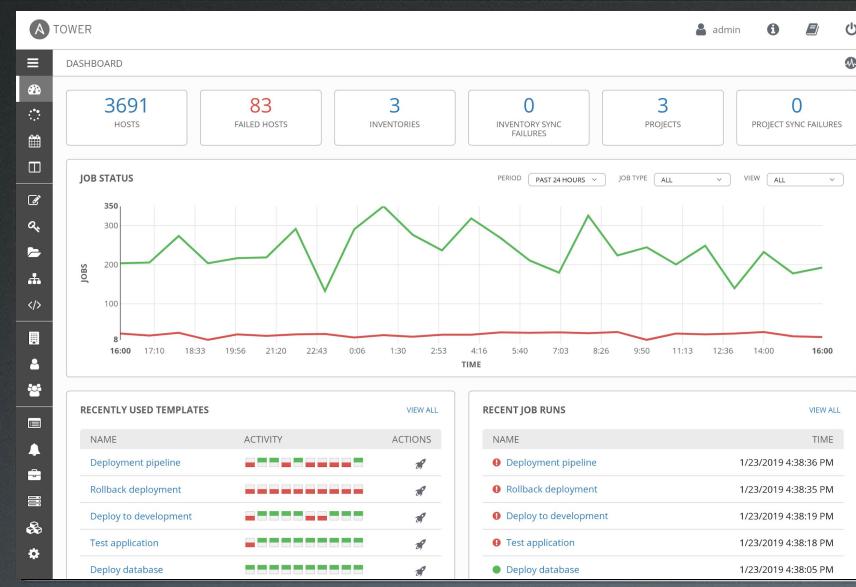
NAME	ACTIONS
Email Results	
Failure Messages	
Prod Ops Complete	
test	



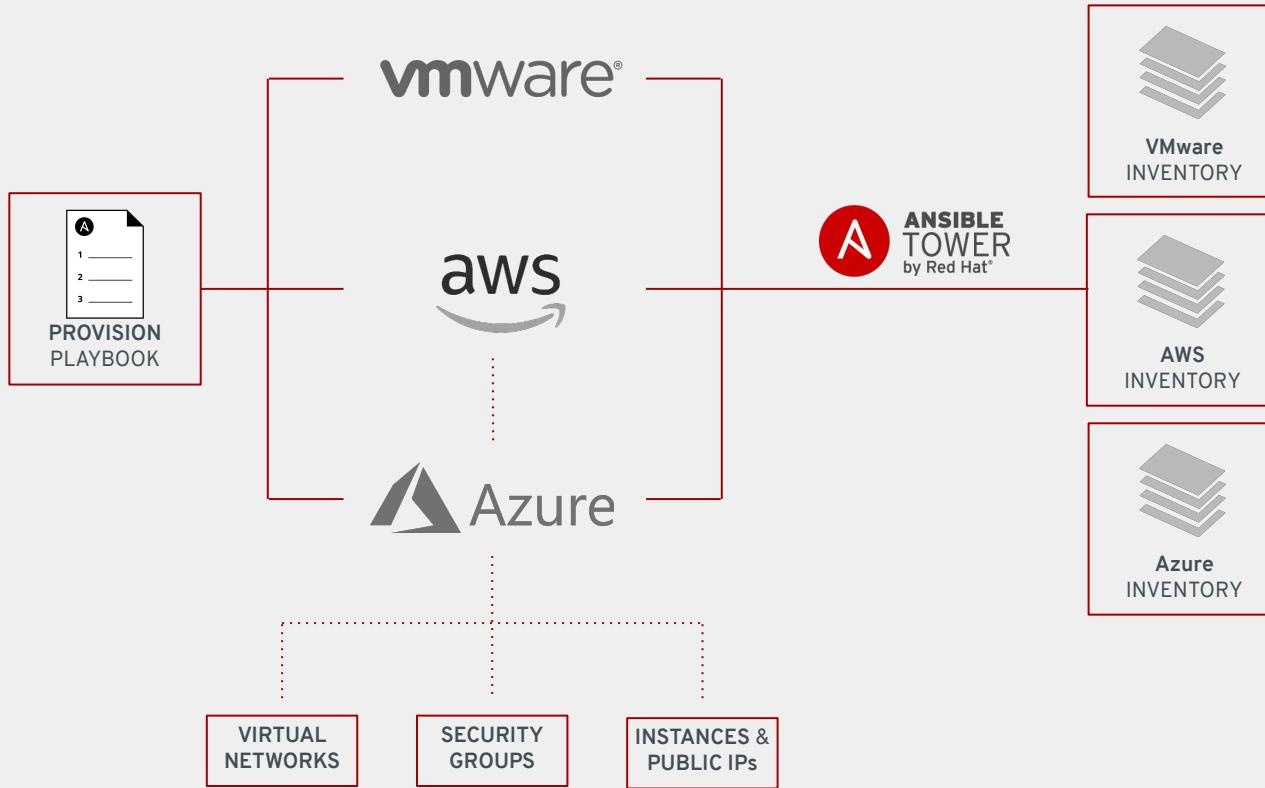
# Red Hat Ansible Tower

## Provision + STIG + Multicloud

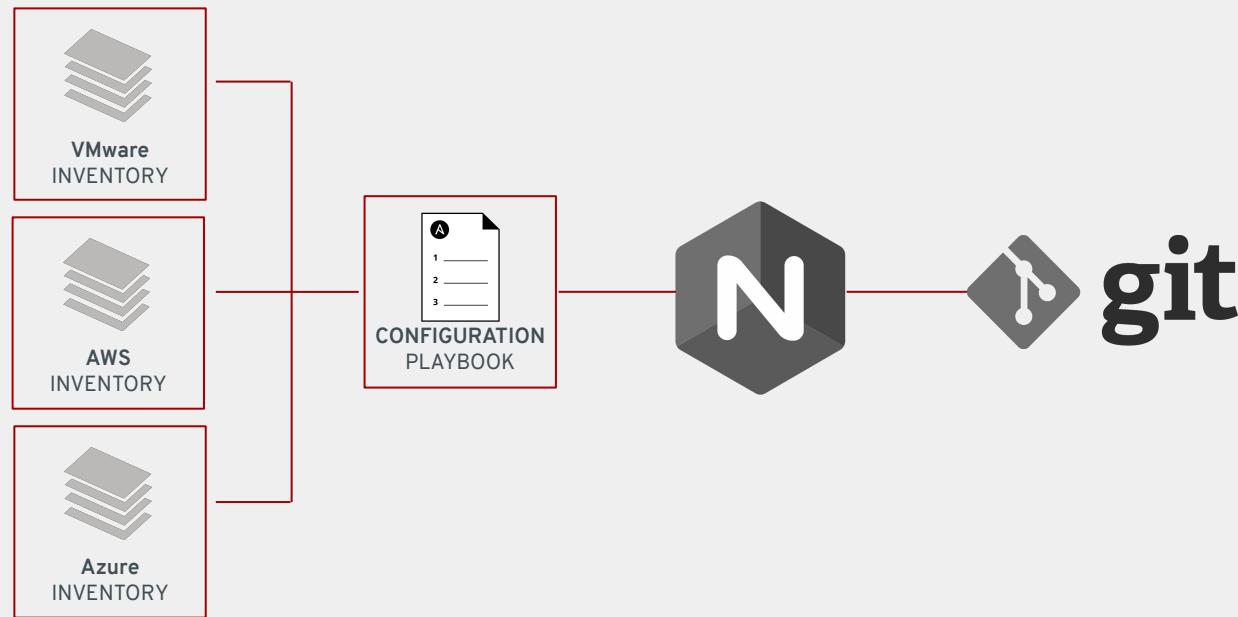
# Demo Overview



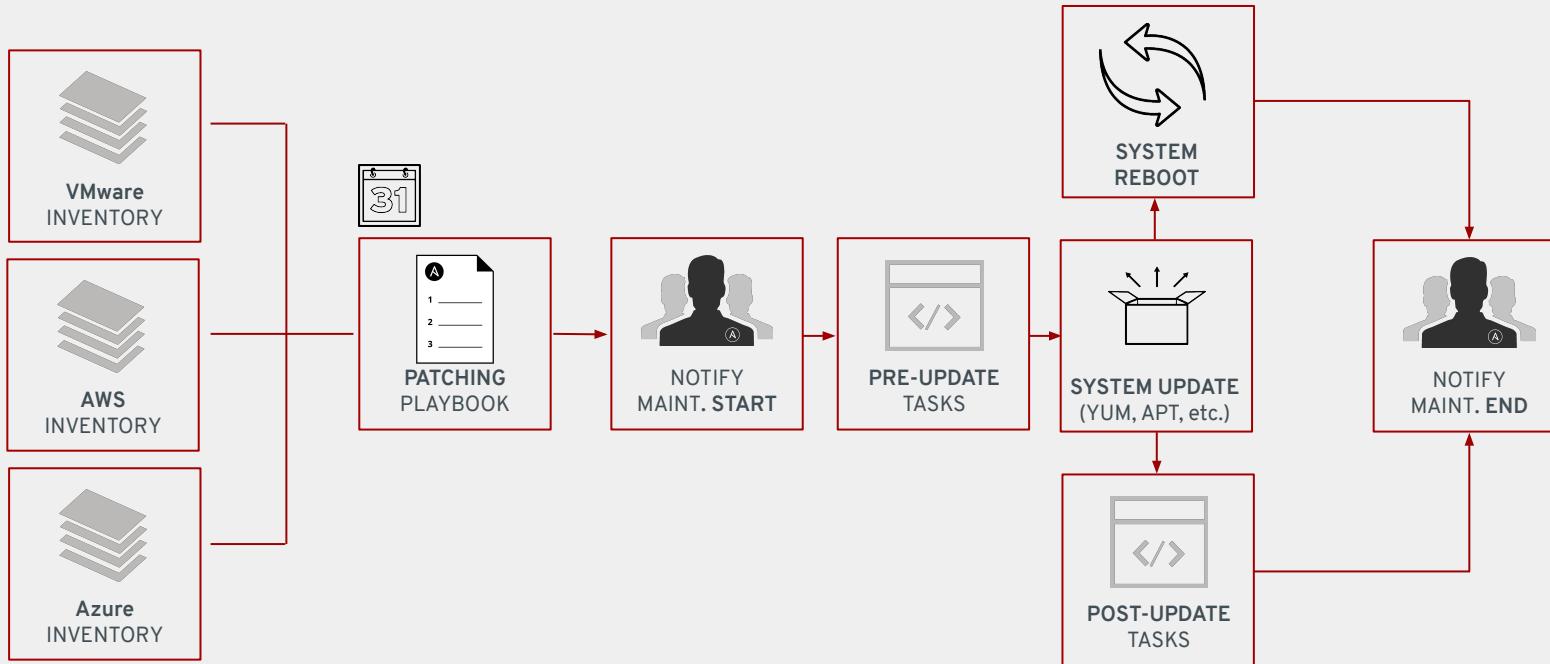
# ANSIBLE TOWER: PROVISIONING



## ANSIBLE TOWER: CONFIG MANAGEMENT + APPLICATION DEPLOYMENT



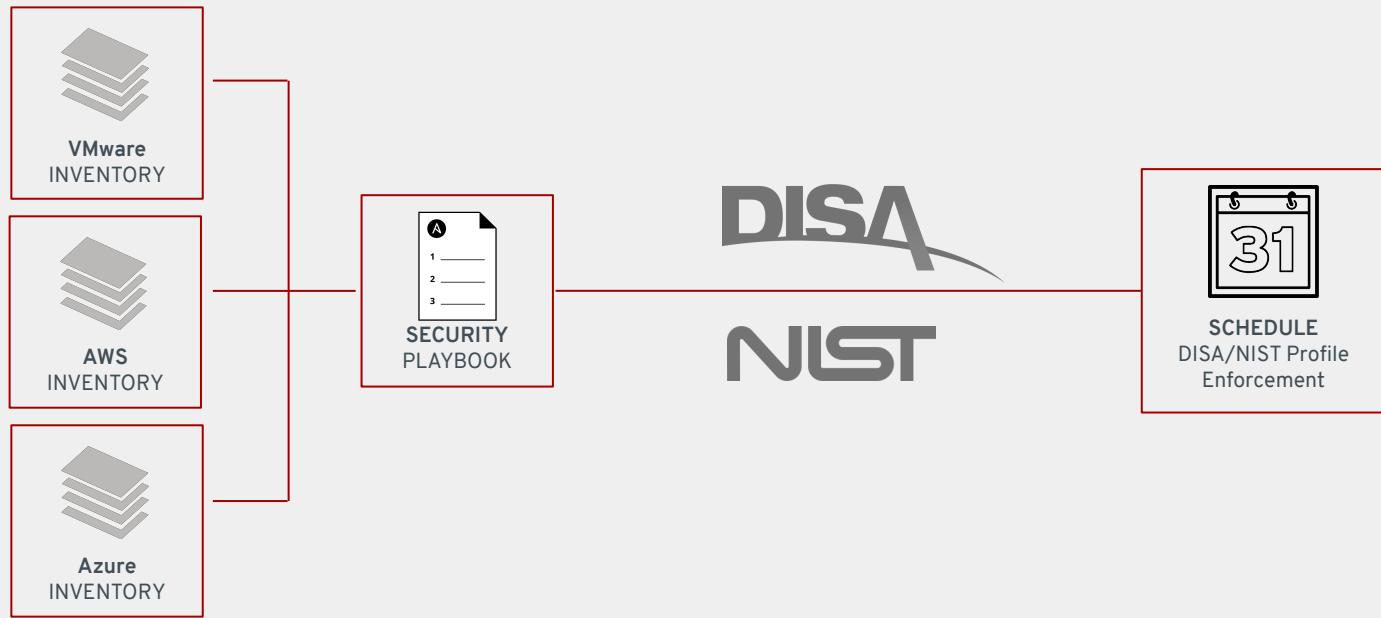
# ANSIBLE TOWER: PATCHING + REMEDIATION



## ANSIBLE TOWER: PATCHING + REMEDIATION

- Compliments, rather than replaces, Red Hat Satellite
  - No content lifecycle management
  - No distribution + synchronization of content
- Ansible helps:
  - Notify system owners of patching maintenance
  - Execute pre- or post-update automation ensuring safe svc shutdown/startup
  - Automate reboots when required by updates (e.g. kernel)
  - Apply + enforce system security profiles (e.g. NIST, STIG)

# ANSIBLE TOWER: SECURITY

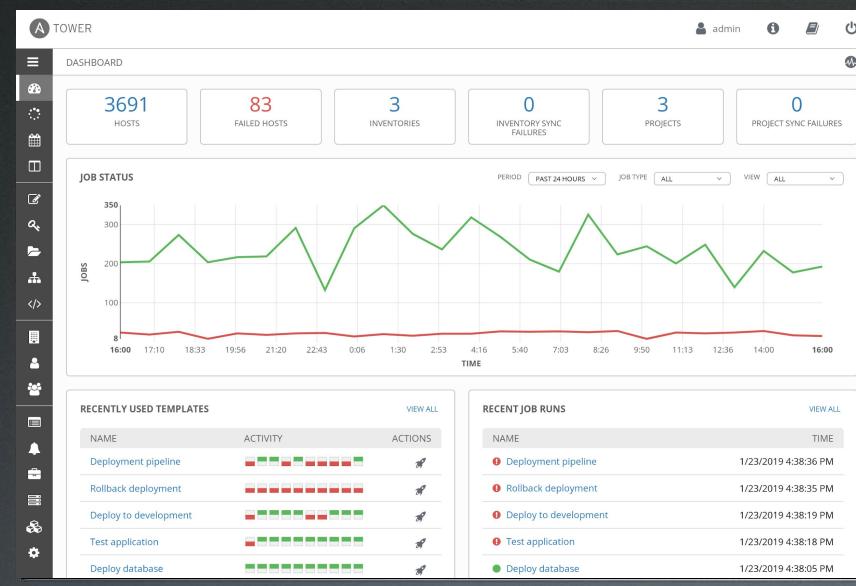




# Red Hat Ansible Tower

## FEATURE OVERVIEW:

# Delegation



# ANSIBLE TOWER FEATURES: ROLE BASED ACCESS CONTROL

USERS

The screenshot shows the 'USERS' section of the Ansible Tower interface. At the top, there is a search bar labeled 'SEARCH' with a magnifying glass icon and a 'KEY' button. Below the search bar is a table header with columns: 'USERNAME' (sorted), 'FIRST NAME', 'LAST NAME', and 'ACTIONS'. The table lists 10 users:

USERNAME	FIRST NAME	LAST NAME	ACTIONS
admin			
amadrid	Bonzo	Madrid	
awiggin	Andrew	Wiggin	
ccarby	Carn	Carby	
dmeeker	Dink	Meeker	
fmolo	Fly	Molo	
hgraff	Hyrum	Graff	
mrackham	Mazer	Rackham	
ndelphiki	Nikolai	Delphiki	
parkanian	Petra	Arkanian	

At the bottom right of the table, it says 'ITEMS 1 - 10'.

TEAMS

The screenshot shows the 'TEAMS' section of the Ansible Tower interface. At the top, there is a search bar labeled 'SEARCH' with a magnifying glass icon and a 'KEY' button. Below the search bar is a table header with columns: 'NAME' (sorted) and 'ACTIONS'. The table lists 5 teams:

NAME	ACTIONS
Cloud Automation Team	
Development Engineering	
Network Administrative Team	
Network Operations Team	
Site Reliability Engineering	

At the bottom right of the table, it says 'ITEMS 1 - 5'.

# ANSIBLE TOWER FEATURES: ROLE BASED ACCESS CONTROL

The screenshot displays the Ansible Tower interface, specifically the 'PERMISSIONS' tab for a 'BACKUP NETWORK CONFIG' template.

**Permissions Tab:**

- USER:** admin (ADMIN, SYSTEM ADMINISTRATOR), amadrid (SYSTEM AUDITOR), awiggin (SYSTEM ADMINISTRATOR), dmeeker (EXECUTE)
- TEAM ROLES:** None assigned.

**ITEMS 1 - 4**

**TEMPLATES [23]:**

**BACKUP NETWORK CONFIG Job Template:**

- ACTIVITY:** [Activity status icons]
- INVENTORY:** Workshop Inventory
- PROJECT:** Workshop Project
- CREDENTIALS:** Workshop Credential
- LAST MODIFIED:** 1/8/2019 9:51:02 PM by admin
- LAST RAN:** 1/8/2019 9:51:02 PM

**Actions:** Rocket icon, Copy icon, Delete icon

# ANSIBLE TOWER FEATURES: SELF-SERVICE I.T.

LAUNCH JOB | DEPLOY SOFTWARE

**\* ENTER NUMBER OF SERVICE INSTANCES.**

2

**\* PLEASE SELECT THE SERVICE OWNER.**

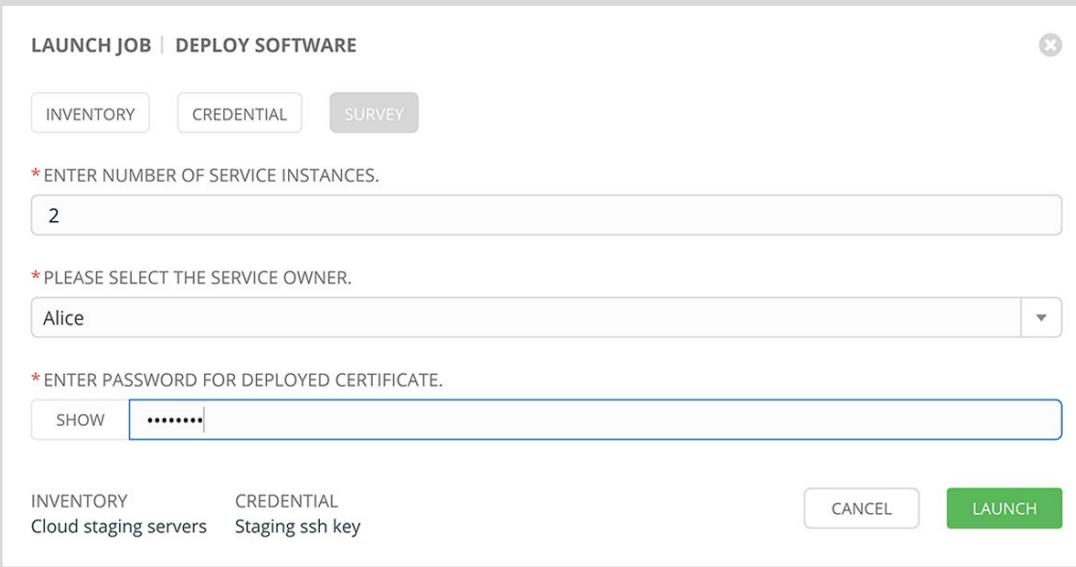
Alice

**\* ENTER PASSWORD FOR DEPLOYED CERTIFICATE.**

SHOW

INVENTORY CREDENTIAL  
Cloud staging servers Staging ssh key

CANCEL LAUNCH



# ANSIBLE TOWER FEATURES: REMOTE COMMAND EXECUTION

The screenshot shows the Ansible Tower interface. The top navigation bar includes the 'TOWER' logo, user 'admin', and various icons for help, refresh, and system status. The main navigation on the left lists 'INVENTORIES / Durham / RUN COMMAND'. The central panel is titled 'EXECUTE COMMAND' and contains the following fields:

- \* MODULE**: yum
- ARGUMENTS**: name=nginx state=restarted
- LIMIT**: rhel1:rhel10
- \* MACHINE CREDENTIAL**: Workshop Credential
- \* VERBOSITY**: 0 (Normal)
- FORKS**: DEFAULT
- SHOW CHANGES**: OFF
- ENABLE PRIVILEGE ESCALATION**
- EXTRA VARIABLES**: YAML (selected), JSON
- YAML content: 1 ---



# Red Hat Ansible Tower

FEATURE OVERVIEW:

## Scale

The screenshot shows the Red Hat Ansible Tower dashboard. At the top, there's a header with the Red Hat logo and a user menu for 'admin'. Below the header, the dashboard has several sections: a summary bar with counts for hosts (3691), failed hosts (83), inventories (3), inventory sync failures (0), projects (3), and project sync failures (0); a 'JOB STATUS' chart showing the number of jobs over time; a 'RECENTLY USED TEMPLATES' table listing five templates with activity indicators and actions; and a 'RECENT JOB RUNS' table listing six job runs with their names, times, and status icons.

NAME	ACTIVITY	ACTIONS
Deployment pipeline	<span style="color:red;">■</span> <span style="color:green;">■</span>	<a href="#">View</a>
Rollback deployment	<span style="color:red;">■</span>	<a href="#">View</a>
Deploy to development	<span style="color:green;">■</span> <span style="color:green;">■</span> <span style="color:red;">■</span> <span style="color:green;">■</span>	<a href="#">View</a>
Test application	<span style="color:red;">■</span> <span style="color:green;">■</span>	<a href="#">View</a>
Deploy database	<span style="color:green;">■</span>	<a href="#">View</a>

NAME	TIME
Deployment pipeline	1/23/2019 4:38:36 PM
Rollback deployment	1/23/2019 4:38:35 PM
Deploy to development	1/23/2019 4:38:19 PM
Test application	1/23/2019 4:38:18 PM
Deploy database	1/23/2019 4:38:05 PM

# ANSIBLE TOWER FEATURES: CREATE AUTOMATION WORKFLOWS

A TOWER

JOBS / 137 - Deploy ticketing application

DETAILS

STATUS Running

STARTED 1/8/2019 4:55:21 PM

FINISHED Not Finished

INVENTORY rtr1

TEMPLATE Deploy ticketing application

LAUNCHED BY admin

EXTRA VARIABLES [YAML](#) [JSON](#) EXPAND

1 ---

Deploy ticketing application

TOTAL NODES 7 ELAPSED 00:00:38

The screenshot shows the Ansible Tower interface. On the left, a sidebar contains various icons for navigation. The main area displays a job details page for "137 - Deploy ticketing application". The "DETAILS" section shows the job status as "Running", started on 1/8/2019 at 4:55:21 PM, and not yet finished. It lists the inventory as "rtr1", the template as "Deploy ticketing application", and it was launched by "admin". Below this, there's an "EXTRA VARIABLES" section with a YAML tab selected, showing a single entry: "1 ---". To the right, the "Deploy ticketing application" workflow is visualized. A blue square node on the left branches into three blue arrows pointing to three white rectangular boxes: "Deploy application", "Deploy webserver", and "Deploy database". From each of these boxes, a green arrow points to a central white box labeled "Run tests". From "Run tests", two green arrows point to a white box labeled "Update CMDB" and one red arrow points to a white box labeled "Rollback environment". The top right of the workflow area shows "TOTAL NODES 7" and "ELAPSED 00:00:38".

# ANSIBLE TOWER FEATURES: SCALE OUT CLUSTERING

The screenshot shows the Ansible Tower 'INSTANCE GROUPS' page. At the top, there is a search bar, a 'KEY' button, and a green '+' button. Below the header, the title 'INSTANCE GROUPS' is displayed with a count of 4. The page lists four instance groups:

INSTANCE GROUP	INSTANCES	RUNNING JOBS	TOTAL JOBS	USED CAPACITY
dev	3	9	89	61.8%
prod	4	6	26	27.3%
test	3	6	44	55.8%
tower	8	0	33	43.6%

At the bottom right of the main content area, it says 'ITEMS 1 - 4'.



**RED HAT®**  
**ANSIBLE®**  
Automation

USE CASE:  
**LINUX AUTOMATION**

# LINUX AUTOMATION

**150+**  
Linux Modules

## AUTOMATE EVERYTHING LINUX

Red Hat Enterprise Linux, BSD,  
Debian, Ubuntu and many more!

ONLY REQUIREMENTS:  
Python 2 (2.6 or later)  
or Python 3 (3.5 or later)

[ansible.com/get-started](http://ansible.com/get-started)



# AUTOMATION FOR EVERYONE: SYSTEM ADMINISTRATORS

```
---  
- name: upgrade rhel packages  
  hosts: rhel  
  
  tasks:  
    - name: upgrade all packages  
      yum:  
        name: '*'  
        state: latest
```

# AUTOMATION FOR EVERYONE: SYSTEM ADMINISTRATORS

```
---  
- name: reboot rhel hosts  
  hosts: rhel  
  
tasks:  
  - name: reboot the machine  
    reboot:
```

# AUTOMATION FOR EVERYONE: SYSTEM ADMINISTRATORS

```
---
```

```
- name: check services on rhel hosts
  hosts: rhel
  become: yes

  tasks:
    - name: ensure nginx is started
      service:
        name: nginx
        state: started
```



RED HAT®  
ANSIBLE®  
Automation

USE CASE:  
**NETWORK AUTOMATION**

# ANSIBLE NETWORK AUTOMATION

**50**

Network  
Platforms

**700+**

Network  
Modules

**12\***

Galaxy  
Network Roles

[ansible.com/for/networks](http://ansible.com/for/networks)  
[galaxy.ansible.com/ansible-network](http://galaxy.ansible.com/ansible-network)

# WHY AUTOMATE YOUR NETWORK?

## PLAN AND PROTOTYPE VIRTUALLY

Use tasks as reusable building blocks

## USE YOUR CURRENT DEVELOPMENT PRACTICES

Agile, DevOps, Waterfall

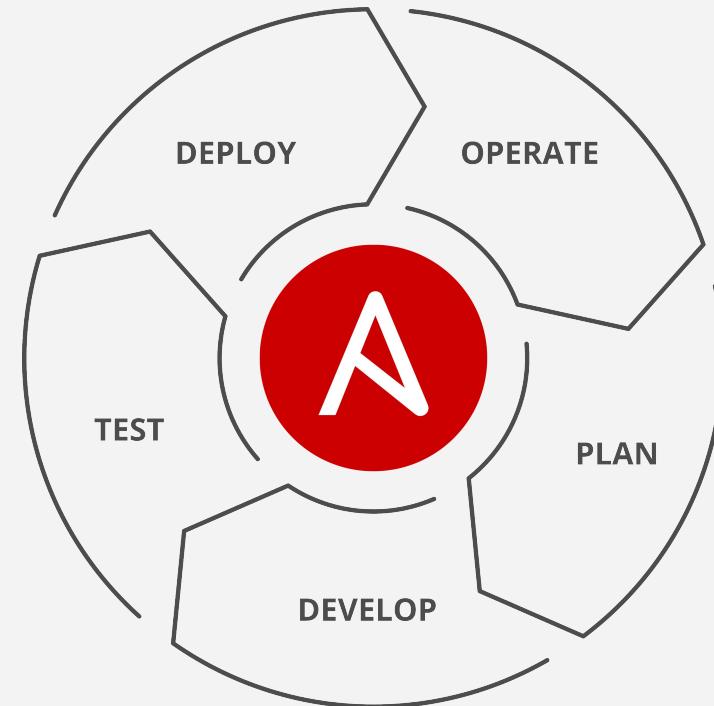
## GO BEYOND THE “PING” TEST

Integrate with formal testing platforms

## BE CONFIDENT DURING DEPLOYMENT

Validate changes were successful

## ENSURE AN ON-GOING STEADY-STATE



# AUTOMATION FOR EVERYONE: NETWORK ENGINEERS

```
---
```

- **hosts**: cisco
- gather\_facts**: false
- connection**: network\_cli
  
- tasks**:
- **name**: show command for cisco
- cli\_command**:  
      **command**: show ip int br
- register**: result
  
- **name**: display result to terminal window
- debug**:  
      **var**: result.stdout\_lines

# AUTOMATION FOR EVERYONE: PLAYBOOK RESULTS

```
[student3@ansible network_setup]$ ansible-playbook example.yml

PLAY [cisco] ****
TASK [show command for cisco] ****
ok: [rtr2]
ok: [rtr1]

TASK [display result to terminal window] ****
ok: [rtr1] => {
  "result.stdout_lines": [
    "Interface          IP-Address      OK? Method Status          Protocol",
    "GigabitEthernet1  172.16.22.120  YES DHCP   up           up  ",
    "VirtualPortGroup0 192.168.35.101 YES TFTP   up           up"
  ]
}
ok: [rtr2] => {
  "result.stdout_lines": [
    "Interface          IP-Address      OK? Method Status          Protocol",
    "GigabitEthernet1  172.17.1.107  YES DHCP   up           up  ",
    "VirtualPortGroup0 192.168.35.101 YES TFTP   up           up"
  ]
}

PLAY RECAP ****
rtr1                  : ok=2    changed=0    unreachable=0    failed=0    skipped=0
rtr2                  : ok=2    changed=0    unreachable=0    failed=0    skipped=0

[student3@ansible network_setup]$ ]
```



# AUTOMATION FOR EVERYONE: NETWORK ENGINEERS

```
---
```

- **hosts**: juniper
- gather\_facts**: false
- connection**: network\_cli
  
- tasks**:
- **name**: show command for juniper
- cli\_command**:  
      **command**: show interfaces terse em1
- register**: result
  
- **name**: display result to terminal window
- debug**:  
      **var**: result.stdout\_lines

# AUTOMATION FOR EVERYONE: PLAYBOOK RESULTS

```
[student3@ansible network_setup]$ ansible-playbook junos-example.yml

PLAY [juniper] *****

TASK [show command for juniper] *****
ok: [rtr3]
ok: [rtr4]

TASK [display result to terminal window] *****
ok: [rtr3] => {
    "result.stdout_lines": [
        "Interface      Admin Link Proto  Local          Remote",
        "em1           up   up",      inet     10.0.0.4/8      ",
        "em1.0         up   up",      inet     128.0.0.1/2      ",
        "              ",          inet     128.0.0.4/2      ",
        "              ",          inet6    fe80::5254:ff:fe12:bdfe/64",
        "              ",          inet6    fec0::a:0:0:4/64",
        "              ",          tnp     0x4"
    ]
}
ok: [rtr4] => {
    "result.stdout_lines": [
        "Interface      Admin Link Proto  Local          Remote",
        "em1           up   up",      inet     10.0.0.4/8      ",
        "em1.0         up   up",      inet     128.0.0.1/2      ",
        "              ",          inet     128.0.0.4/2      ",
        "              ",          inet6    fe80::5254:ff:fe12:bdfe/64",
        "              ",          inet6    fec0::a:0:0:4/64",
        "              ",          tnp     0x4"
    ]
}

PLAY RECAP *****
rtr3                  : ok=2    changed=0    unreachable=0    failed=0    skipped=0
rtr4                  : ok=2    changed=0    unreachable=0    failed=0    skipped=0

[student3@ansible network_setup]$
```





RED HAT®  
ANSIBLE®  
Automation

USE CASE:  
**WINDOWS AUTOMATION**

# WINDOWS AUTOMATION

**90+**

Windows  
Modules

**1,300+**

Powershell DSC  
resources

[ansible.com/windows](http://ansible.com/windows)

# AUTOMATION FOR EVERYONE: WINDOWS ADMINS

```
---
```

```
- name: windows playbook

hosts: new_servers

tasks:
  - name: ensure local admin account exists
    win_user:
      name: localadmin
      password: '{{ local_admin_password }}'
      groups: Administrators
```

# AUTOMATION FOR EVERYONE: WINDOWS ADMINS

```
---
```

```
- name: windows playbook

hosts: windows_machines

tasks:
  - name: ensure common tools are installed
    win_chocolatey:
      name: '{{ item }}'
    loop: ['sysinternals', 'googlechrome']
```

# AUTOMATION FOR EVERYONE: WINDOWS ADMINS

```
---
```

```
- name: update and reboot
  hosts: windows_servers
  tasks:
    - name: ensure common OS updates are current
      win_updates:
        register: update_result

    - name: reboot and wait for host if updates change require it
      win_reboot:
        when: update_result.reboot_required
```

# AUTOMATION FOR EVERYONE: WINDOWS ADMINS

```
---
```

- **name:** update domain and reboot
  - hosts:** windows\_servers
  - tasks:**
    - **name:** ensure domain membership
      - win\_domain\_membership:**
        - dns\_domain\_name:** contoso.corp
        - domain\_admin\_user:** '{{ domain\_admin\_username }}'
        - domain\_admin\_password:** '{{ domain\_admin\_password }}'
        - state:** domain
      - register:** domain\_result
    - **name:** reboot and wait for host if domain change require it
      - win\_reboot:**
        - when:** domain\_result.reboot\_required



# Red Hat Ansible Automation

USE CASE:

## Cloud automation

# CLOUD AUTOMATION

**800+**

Cloud  
Modules

**30+**

Cloud Platforms

[ansible.com/cloud](http://ansible.com/cloud)

# PLAYBOOK EXAMPLE: AWS

```
---
```

```
- name: aws playbook
```

```
  hosts: localhost
```

```
  connection: local
```

```
  tasks:
```

```
    - name: create AWS VPC ansible-vpc
```

```
      ec2_vpc_net:
```

```
        name: "ansible-vpc"
```

```
        cidr_block: "192.168.0.0/24"
```

```
      tags:
```

```
        demo: the demo vpc
```

```
      register: create_vpc
```

## PLAYBOOK EXAMPLE: AZURE

```
---
```

```
- name: azure playbook
```

```
  hosts: localhost
```

```
  connection: local
```

```
  tasks:
```

```
    - name: create virtual network
```

```
      azure_rm_virtualnetwork:
```

```
        resource_group: myResourceGroup
```

```
        name: myVnet
```

```
        address_prefixes: "10.0.0.0/16"
```

# PLAYBOOK EXAMPLE: RED HAT OPENSTACK

```
---
```

```
- name: openstack playbook
  hosts: localhost
  connection: local

  tasks:
    - name: launch an instance
      os_server:
        name: vml
        cloud: mordred
        region_name: ams01
        image: Red Hat Enterprise Linux 7.4
        flavor_ram: 4096
```



# Red Hat Ansible Automation

USE CASE:

## Security Automation

# What is it?

**Ansible Security Automation** is a supported set of Ansible modules, roles and playbooks designed to unify the security response to cyberattacks in a new way - by orchestrating the activity of multiple classes of security solutions that wouldn't normally integrate with each other.

# What does it do?

Through Ansible Security Automation, IT organizations can address multiple popular use cases:

- For **detection and triage of suspicious activities**, for example, Ansible can automatically enable logging or increase the log verbosity across enterprise firewalls and IDS to enrich the alerts received by a SIEM for an easier triage.
- For **threat hunting**, for example, Ansible can automatically create new IDS rules to investigate the origin of a firewall rule violation, and whitelist those IP addresses recognized as non threats.
- For **incident response**, for example, Ansible can automatically validate a threat by verifying an IDS rule, trigger a remediation from the SIEM solution, and create new enterprise firewall rules to blacklist the source of an attack.

At launch, Red Hat's Ansible security automation platform provides support for:

- **Check Point** – Next Generation Firewall (NGFW);
- **Splunk** – Splunk Security Enterprise (SE);
- **Snort**

# Who is it for?

Ansible Security Automation extends the Ansible agentless, modular and easy to use enterprise automation platform to support the following industry constituencies:

- **End-user organizations' security teams** in charge of Security Operations Centres (SOCs)
- **Managed security service providers (MSSPs)** responsible for the governance of thousands of enterprise security solutions across their whole customer base
- **Security ISVs** offering security orchestration and automation (SOAR) solutions currently using custom-made automation frameworks

# AUTOMATION FOR EVERYONE: SECURITY OPERATIONS

```
---
```

```
- name: checkpoint playbook
  hosts: checkpoint
  connection: httpapi

  tasks:
    - name: create access rule
      checkpoint_access_rule:
        layer: Network
        name: "Drop attacker"
        position: top
        source: attacker
        destination: Any
        action: Drop
```

# AUTOMATION FOR EVERYONE: SECURITY OPERATIONS

```
---
```

```
- name: checkpoint playbook
  hosts: checkpoint
  connection: httpapi

  tasks:
    - name: delete access rule
      checkpoint_access_rule:
        layer: Network
        name: "Drop attacker"
        state: absent
```

# NEXT STEPS

## GET STARTED

[ansible.com/get-started](https://ansible.com/get-started)

[ansible.com/tower-trial](https://ansible.com/tower-trial)

---

## WORKSHOPS & TRAINING

[ansible.com/workshops](https://ansible.com/workshops)

[Red Hat Training](#)

## JOIN THE COMMUNITY

[ansible.com/community](https://ansible.com/community)

---

## SHARE YOUR STORY

[Follow us @Ansible](#)

[Friend us on Facebook](#)

