From CLI to Tower

David Federlein, Director of Product Field Engineering
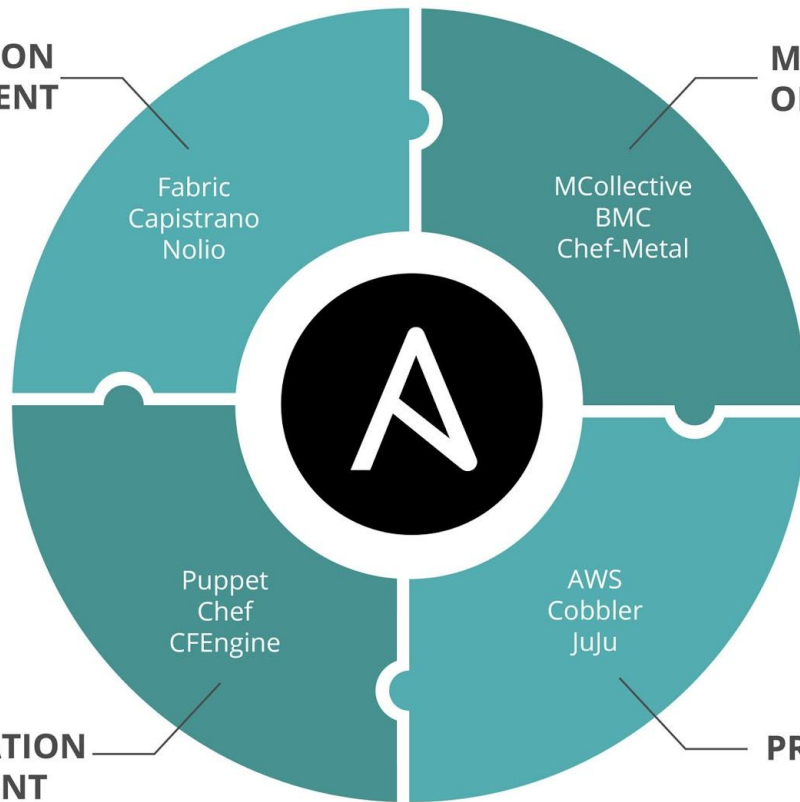
ANSIBLE TOWER
by Red Hat®

TOWER EXPANDS AUTOMATION TO YOUR ENTERPRISE.

CONTROL	**KNOWLEDGE**	**DELEGATION**
Scheduled and centralized jobs	Visibility and compliance	Role-based access and self-service

| **SIMPLE**	**POWERFUL**	**AGENTLESS**
Everyone speaks the same language	Designed for Multi-tier deployments	Predictable, reliable, and secure

AT ANSIBLE'S CORE IS AN **OPEN-SOURCE** AUTOMATION ENGINE.

WHAT IS ANSIBLE TOWER?

Ansible tower is an **enterprise framework** for controlling, securing and managing your Ansible automation – with a **UI and RESTful API.**

- **Role-based access control** keeps environments secure, and teams efficient.

- Non-privileged users can **safely deploy** entire applications with **push-button deployment** access.

- All Ansible automations are **centrally logged,** ensuring **complete auditability and compliance**.
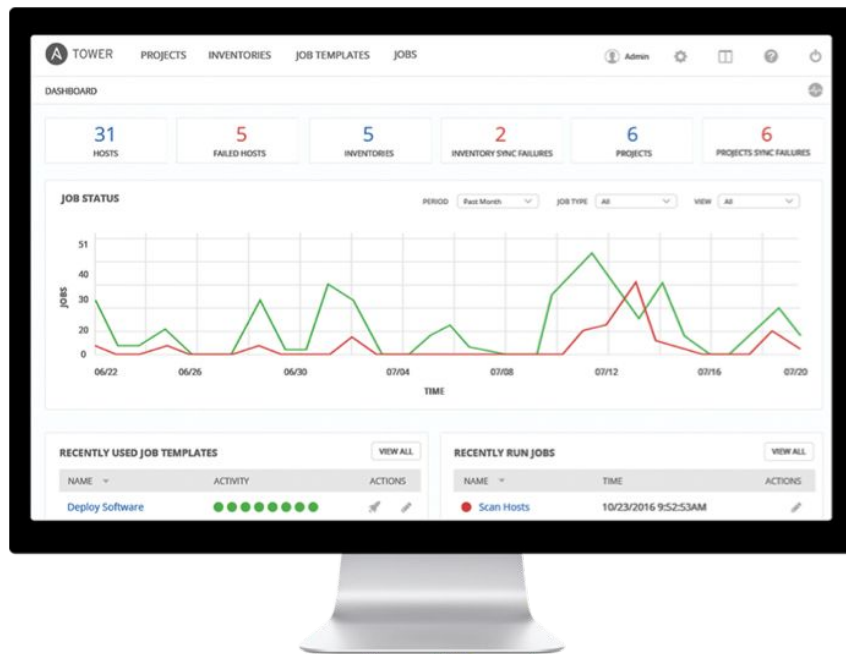
WHAT PROBLEMS DOES EACH SOLVE?

Ansible solves the problem
of automating
and orchestrating

Does not address bigger
picture security/compliance

Tower spotlights security
considerations and
provides predictability

Role-based access control and
secure credential storage

API integrations, accountability and
execution history

Only respects security in place at host user level

No abstraction of remote host or cloud credentials from user

No guarantee of execution parameters or integrity of Playbook as designed by the team

Abstracts security from the user

CONTROL YOUR ANSIBLE DEPLOYMENT

SITUATIONAL AWARENESS IS THE KEY TO DEVOPS

- Dashboard and real-time automation updates

- Integrated RBAC with credential management

- SCM controls security of playbooks

- Job scheduling

- Graphical inventory management

- Built-in notifications to keep teams informed

- Stabilized API to plumb into existing tooling and processes

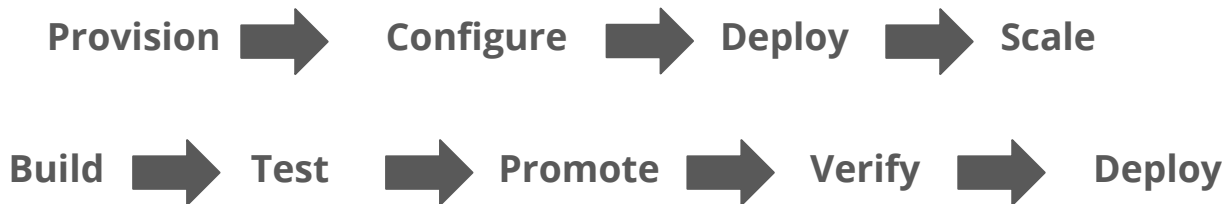- Model entire processes with **new Workflows**

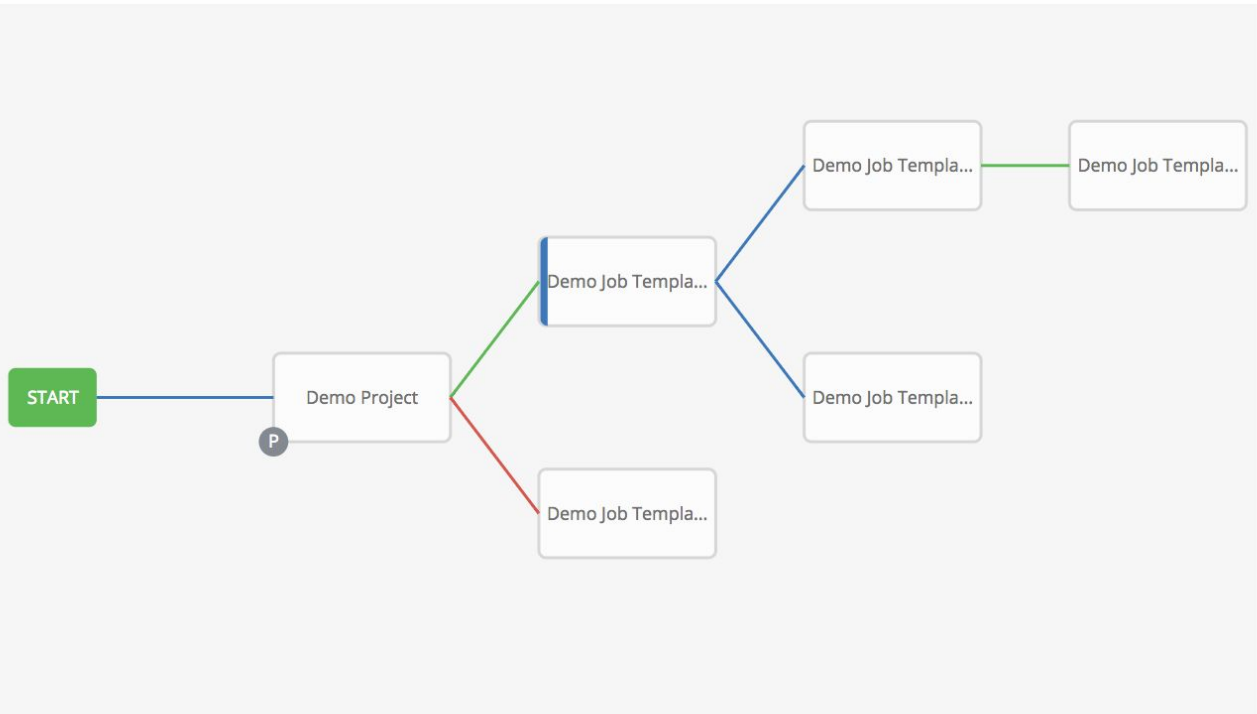## MIX AND RE-USE AUTOMATIONS WITHOUT WRITING A PLAYBOOK

- Combine any number of Playbooks into a Workflow

- Delegate access just like any other Tower automation

- Launchable with customizable parameters

- Easily build in-app workflows

Provision ➡ **Configure** ➡ **Deploy** ➡ **Scale**

Build ➡ **Test** ➡ **Promote** ➡ **Verify** ➡ **Deploy**

EDIT WORKFLOW

KEY: ▬ On Success ▬ On Fail ▬ Always Ⓟ Project Sync Ⓘ Inventory Sync   TOTAL TEMPLATES 6 ⚙

START

Demo Project

Demo Job Templa...

Demo Job Templa...

Demo Job Templa...

Demo Job Templa...

Demo Job Templa...

DEMO JOB TEMPLATE

| JOBS | PROJECT SYNC | INVENTORY SYNC |

SEARCH 🔍   KEY

NAME ▲
⦿ Demo Job Template   INFO

ITEMS 1 - 1 OF 1

*TYPE
⦿ On Success
◯ On Failure

CANCEL   SELECT

CLOSE   SAVE

DELEGATE ANSIBLE TO ANYONE

EMPOWER YOUR TEAMS INSIDE AND OUTSIDE OF OPERATIONS

- Connect to your LDAP, AD, SAML and other directories
- Full role-based access control engine
- Store credentials for use without exposure
- Enable users to automate without previous Ansible knowledge
- Find relevant information more quickly with **new Smart Search**
- Simple surveys configure automation at run-time
- REST API allows integration into your existing processes and tools
- Add capacity with **new Tower Clusters**

INTRODUCING SMART SEARCH

KEY:VALUE SEARCHING IN NEARLY ANY LIST OF OBJECTS

- Drill down to just the items you want to see
- Rapidly find just the entries you need
- Works across Activity Stream, Projects, Users, Teams, Inventories, and more.

JOBS

SEARCH   🔍   **KEY**

EXAMPLES: `name:foo`   `organization.name:Default`   `id:>10`

FIELDS: id, created, modified, name, description, launch_type, status, failed, started, finished, elapsed, job_explanation, execution_node

RELATED FIELDS: unified_job_template, system_job_template, notifications, cancel, created_by, workflow_job_template, workflow_nodes, labels, relaunch, inventory, project, credential, stdout, source_workflow_job, job_host_summaries, job_events, job_template, start, project_update

ADDITIONAL INFORMATION: For additional information on advanced search search syntax please see the Ansible Tower documentation.

ID ⇕	NAME ⇕	TYPE ⇕	FINISHED ▾	LABELS	ACTIONS
● 31	Cleanup Job Details	Management Job	2/19/2017 2:13:53 PM		🔍 🗑
● 20	Sample Workflow	Workflow Job	2/17/2017 12:25:12 PM		🔍 🚀 🗑
● 28	Demo Job Template Ⓦ	Playbook Run	2/17/2017 12:25:11 PM		🔍 🚀 🗑
● 29	Demo Project	SCM Update	2/17/2017 12:25:03 PM		🔍 🚀 🗑
● 25	Demo Job Template Ⓦ	Playbook Run	2/17/2017 12:24:57 PM		🔍 🚀 🗑
● 26	Demo Project	SCM Update	2/17/2017 12:24:50 PM		🔍 🚀 🗑
● 22	Demo Job Template Ⓦ	Playbook Run	2/17/2017 12:24:46 PM		🔍 🚀 🗑
● 23	Demo Project	SCM Update	2/17/2017 12:24:39 PM		🔍 🚀 🗑
● 21	Demo Project Ⓦ	SCM Update	2/17/2017 12:24:34 PM		🔍 🚀 🗑

ADD TOWER CAPACITY AND REDUNDANCY WITH EASE

- Add new Tower nodes to scale out Tower job capacity
- Tower node fails? No problem
- Individual Tower jobs will run on any node with available capacity
  - Jobs are not spanned across multiple Tower nodes
- Cluster stays in sync with in-Tower configuration

ENSURE ENVIRONMENT CONSISTENCY

- All automation securely logged in Tower
- Use Tower's activity stream for auditing
- Notifications automatically alert the channel of your choice
- **New Enterprise Logging support** automatically pushes results to external aggregators

ANALYZE YOUR AUTOMATION RESULTS

- Log all Tower activity to central enterprise logging

- Cross-reference automation with events and application logs

- Use Tower's API to perform remediation if needed

- Support for:

  - Elastic

  - Splunk

  - Sumologic

  - Loggly

  - Custom (Via WebHook/RESTful API)

SHARING IS CARING

Modular approaches help teams respect each others' standards of operations

SCM is key to spreading Ansible best practices and helping cross-train newcomers to the Orchestration table

Central repository, source control and process equals accountability for Playbook maintenance and development

Write a role, not a Playbook

Groups are not hierarchies they are Venn diagrams
- "Type" groups can overlap with "location" groups
- Bare groups (not groups of groups) should be specific as possible
- Groups of groups should not have overlapping variables to bare groups

Group variables can overstep each other in unpredictable ways

Variable precedence is key to knowing where to put your variables

ANSIBLE

Multiple Inventories may be needed
- Network automation may not need to have app/cache/db hosts in their inventory
- Multiple Cloud Dynamic Sources will overwrite each other

Things that "just work" for CLI may need consideration in Tower
- Execution isolation means config files at ~/.* need to be placed for AWX user in Tower
- PRoot isolates Tower runs to project/Playbook directory (can't write to /tmp locally, etc.)

SCM is to your advantage:
- Playbook projects for different teams/orgs can utilize forks, branch tags
- Roles don't need one monolithic repository

redhat.

ANSIBLE

ansible.com/resources
Whitepapers can help shape your use case

galaxy.ansible.com
Galaxy can help give examples of roles that do things similar to your goals

docs.ansible.com
Examples are comprehensive both in the Core Playbook section and the module list

ansible.com/community
Even if you pay for support, community help is an invaluable tool

redhat.

MORE INFORMATION

https://www.ansible.com/resources
https://docs.ansible.com/
https://www.ansible.com/tower
https://www.ansible.com/tower-editions