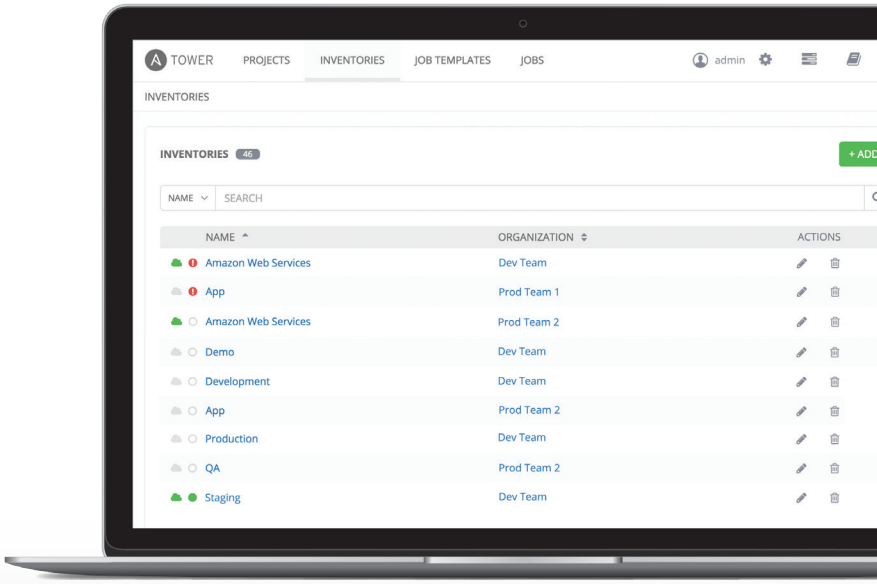
ANSIBLE
by Red Hat

BEGINNER'S GUIDE:

CONTROL
WITH ANSIBLE TOWER



CONTENTS

INTRODUCTION

In this paper, we'll discuss how IT organizations can expand automation to the enterprise and bring new levels of control, security and delegation capabilities to Ansible environments. We'll also go beyond the marketing speak to explain what this actually means in practice. Finally, using examples and tips from our own team, we'll show you how easy it is to adopt a mission control approach to running Ansible in your organization.

IT is driving innovation. If you can't deliver software fast, your organization can't deliver, period. Yet one of the biggest barriers to innovation is complexity. To overcome this obstacle many organizations are looking to automation and DevOps tools and practices. But getting to DevOps and other agile methodologies has always required unique programming skills, until now.

Using the open source Ansible automation engine, organizations are deploying apps faster, managing systems more efficiently and crushing complexity. In doing so, they are building a strong foundation for DevOps and making automation a reality for everyone.

If you're already a user, you'll know this because each day you're experiencing Ansible's simple, powerful and agentless automation (and you learned it quickly). Deploying software became fun again! That's because Ansible loves the repetitive work that your people hate. It doesn't require special coding skills, thereby removing some of the most significant barriers to automation across IT and it gives you the one thing you can't get enough of — time.

THE CHALLENGE OF MAINTAINING CONTROL

Automation for everyone is great, but with extensibility comes challenges, particularly for team-based use.

Consider this scenario: When you first started working with Ansible, your team of users may have been small. Ansible worked perfectly, saving you time through automation. But as more users started adopting Ansible, the landscape changed. Now a variety of users are writing their own Playbooks or trying to configure your organization's entire infrastructure — at one time. It could be anywhere between five to twenty people, maybe more. There's no magic number, but if you have multiple users running disparate or concurrent automation, things can get tricky.

In smaller team environments where everyone is well versed on Ansible, maintaining control over your infrastructure and adhering to best practices in terms of Playbooks, security and delegation is manageable. But many organizations have team-based usage needs that stretch beyond Ansible's command line interface (CLI).

Specifically, organizations need:

Control
Allows delegation of authority to different users or teams and lock down access for particular projects or resources.

Scheduling
Allows you to schedule jobs and set repetition options.

Visibility
Administrators want a real-time view of what Ansible is up to at any time, such as job status updates, Playbook runs, as well as what's working or not in their Ansible environment.

Inventory
A better way to manage and track their entire inventory, even across complex, hybrid virtualized and cloud environments.

System Tracking
Verifies that machines are in compliance and configured exactly as they should be.

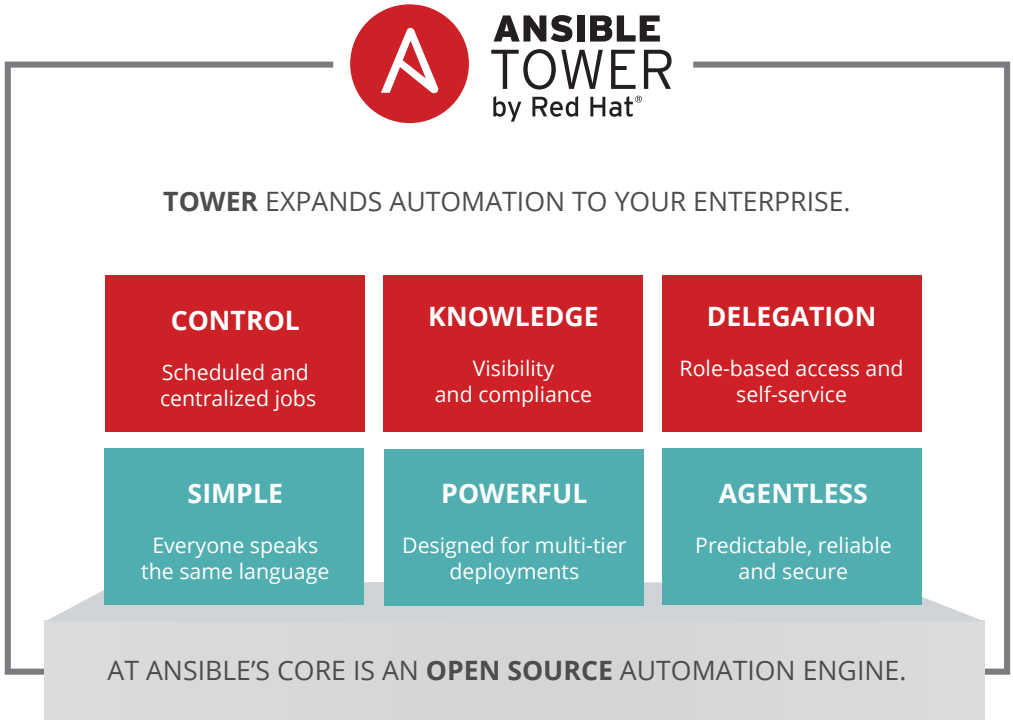
Enterprise Integration
Integrates Ansible into an existing environment and enterprise tool set.

Self-Service IT
Provides the flexibility to free up time and delegate automation jobs to others.

A BETTER WAY TO RUN ANSIBLE

Ansible Tower by Red Hat checks off many of these items. Tower is an enterprise framework for controlling, securing and managing your Ansible automation with a UI and RESTful API. Tower is the best way to run Ansible in your organization because it layers control, knowledge and delegation on top of Ansible's simple, powerful automation engine.

As mission control for Ansible, Tower centralizes and controls your Ansible infrastructure with a visual dashboard that provides a heads-up NOC-style display of everything going on in your Ansible environment, role-based access control, job scheduling and graphical inventory management. Because it centralizes Ansible runs, Tower also makes it easier to integrate Ansible into other systems or workfows required for things like continuous integration and continuous delivery (CI/CD) or DevOps processes.



TOWER AND INTEGRATION IN A LARGE ENTERPRISE

Tower is particularly useful for enterprise or team-based Ansible usage because it streamlines and centralizes Ansible IT automation initiatives — many of which can be run by delegates within your organization, without any Ansible expertise.

Tower is used in a variety of different ways, from traditional configuration management, to custom application deployment, to the orchestration of zero-downtime rolling updates. Companies like Amelco use Ansible to deploy their infrastructure consistently and repeatedly. NASA uses Tower to update security vulnerabilities and to patch and manage nasa.gov weekly. Enterprises that make money delivering applications via the web find that Tower excels at removing IT bottlenecks, automating repetitive tasks and accelerating the delivery of applications to market.

3 WAYS TO TAKE CONTROL OF YOUR INFRASTRUCTURE WITH TOWER

When layered with the powerful open source Ansible automation engine that you've come to depend on, Tower provides many powerful tools to make your automation life easier, adds utility to Ansible and helps you take control of your Ansible environment. But how does that work in practice?

Here are some of the ways that Ansible and Tower go beyond just running automated Playbooks to help you take control your infrastructure. We've also included best practices and sample Playbooks that you can put to work immediately.

1. AUTOMATE CONFIGURATION MANAGEMENT

Centralizing configuration file management and deployment is a common use case for Ansible. It's also how many power users are first introduced to the Ansible automation platform. In fact, Ansible is the simplest solution for configuration management available. It's designed to be minimal in nature, consistent, secure and highly reliable, with an extremely low learning curve for administrators, developers and IT managers.

One of the key reasons for this is that Ansible configurations are simple data descriptions of your infrastructure (both human-readable and machine-parsable) that ensure everyone on your team will be able to understand the meaning of each configuration task. New team members can quickly dive in and make an impact. Existing team members can get work done faster — freeing up cycles to attend to more critical and strategic work instead of configuration management.

Tower offers a number of features that brings new levels of control to your configurations including automated configuration, provisioning callbacks, job scheduling and continuous remediation.

AUTOMATED CONFIGURATION

Ansible Playbooks can be run on any machine at any time to apply configuration, but add Ansible Tower to the equation and you can also ensure that every machine launched in the environment is properly configured, automatically.

Whether you want to apply available updates or have a more detailed configuration that defines your infrastructure, Tower gives you complete, automated control.

Check out these examples of configuration Playbooks that you can use within your own Ansible environment.

PROVISIONING CALLBACKS

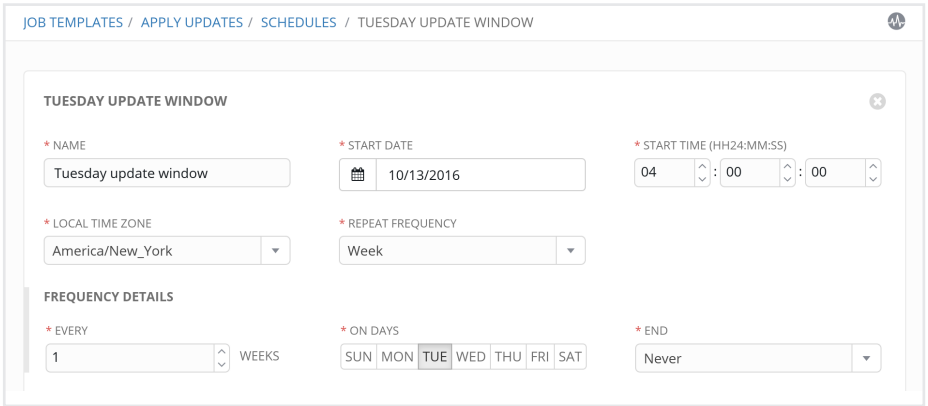
Tower also lets you automatically configure a system after it has been provisioned by another system (such as AWS auto-scaling or an OS provisioning system like Kickstart or Preseed) or for invoking a job programmatically without using the Tower user interface directly. Using Tower's provisioning callbacks feature, any Tower Playbook can be triggered to run on a machine via Tower's RESTful API, rather than waiting for a user to launch a job to manage the host from the Tower console.

To set up a provisioning callback and view a sample EC2 provisioning Playbook, check out these tips.

JOB SCHEDULING

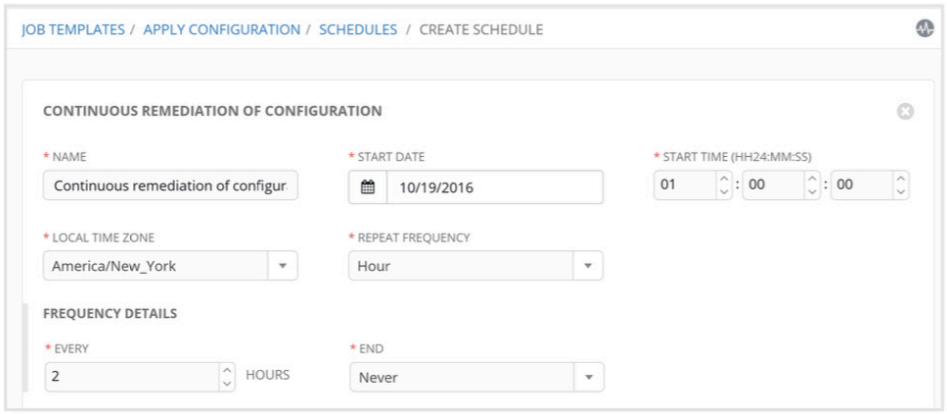
Move beyond manual scripts and ad hoc practices with a consistent, reliable and secure way to manage your environment. Playbook runs, cloud inventory updates and source control updates can all be scheduled inside Tower and schedules may be set to occur once or repeat (such as during maintenance windows). In the case of many management requests, the built- in queuing system will ensure jobs are run efficiently.

Scheduling can enable periodic remediation, continuous deployment or even schedule nightly backups. It is easy to configure a schedule. When editing a job template, simply add them under the Schedules expander. You can also navigate to the list of job templates, click the schedule icon and then click "+" to add a new schedule. The job will apply updates automatically on a schedule. If you ever need to pause or stop the schedule, you can.



CONTINUOUS REMEDIATION

Applying a configuration at machine boot is rarely the end of your configuration management duties. Changes invariably follow. Operating system updates, application changes or local changes made by system administrators can all contribute to configuration drift. Hence, the concept of continuous remediation. Continuous remediation automatically applies your configuration on a regular basis to mitigate drift away from its baseline. Ansible makes continuous remediation efficient, but Tower's job scheduling makes it easy.



You can schedule remediation to run as often as is convenient. Once the configuration remediation has run, it's time to interpret the results. While important that your configuration is consistently applied, any persistent configuration resets could indicate a problem. To determine next steps, you need to know what changes you made. A key benefit of Ansible is that it only makes a change if it has to; otherwise the task is reported as "ok." This is often referred to as desired state configuration or idempotency. Combine this with Tower's auditing and logging of all Ansible runs and this makes finding these cases of configuration drift simple. Learn how.

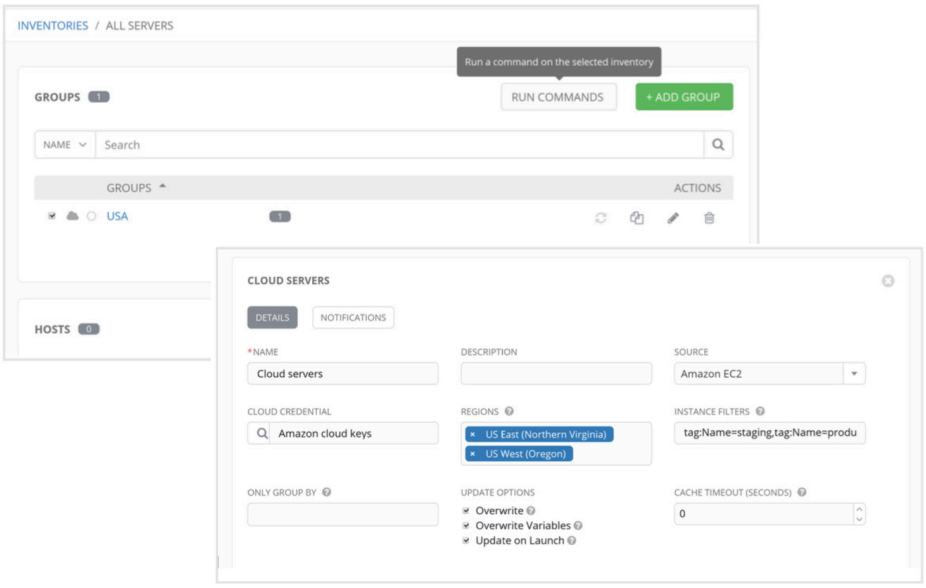
2. MANAGE AND TRACK YOUR ENTIRE INVENTORY

In the last section, we talked about how Tower makes it easy to control the way your infrastructure is configured via configuration definition and continuous remediation. But controlling the configuration of your infrastructure is just one step. You also need a single source of truth for your inventory so that it can be properly controlled within Tower.

Tower provides a number of features that let you easily define and manage your inventory, whether it's AWS, Rackspace, OpenStack, Google Compute Engine, Azure or VMware clouds. Tower not only helps you keep your cloud inventory in sync, its powerful provisioning callbacks allow nodes to request configuration on-demand, enabling autoscaling.

INVENTORY — THE BASICS

If you've used Ansible, you know about the basics of inventory. A static Ansible inventory is just an INI-style file that describes your hosts and groups, with the option to apply variables to your hosts and groups. You can view an example of static inventory here. As you can see from the image below, you can easily enter the same sort of inventory into Ansible as well. You also have the flexibility to easily enter inventory using Tower's RESTful API. Tower supports multiple inventories making it easy to create dev, test and production inventories that are similar. Refer to these examples of how to create inventories via the RESTful API.



DYNAMIC INVENTORY

As seen above, Tower can be a source of truth for your inventory. However, most environments have a highly dynamic inventory as machines are provisioned and retired and complex sets of groups, facts and variables for those machines can come from a variety of sources — a cloud provider, a provisioning system or a configuration management database.

Ansible and Tower work with these sources through the concept of dynamic inventory. Consider this example: If you're using AWS as an inventory source, you'd create a group for your AWS hosts and configure the inventory to use Amazon EC2 as an inventory source. This inventory can be filtered in a variety of ways — region, image tags or any other piece of Amazon metadata. Once this inventory group is created, you can update this inventory on demand, on a schedule, or even automatically whenever you run a Playbook that references the inventory. And, as always, setting up dynamic inventory is available via the API as well.

CUSTOM DYNAMIC INVENTORY

Not only does Tower come with inventory scripts for the all the major public and private cloud providers, such as Amazon, Microsoft Azure, OpenStack and more, but it's easy to add your own dynamic inventory as well. Under Tower's Setup menu, there is an item for "Inventory Scripts," which allows you to upload custom inventory scripts.

ADVANCED CUSTOM DYNAMIC INVENTORY

Going a step further, if you have inventory stored alongside your Playbooks in source control, you can update it in lockstep with your Playbooks without having to manually sync it to Tower. View examples of dynamic, custom and advanced custom dynamic inventory scripts.

3. SIMPLIFY DAY-TO-DAY MANAGEMENT TASKS

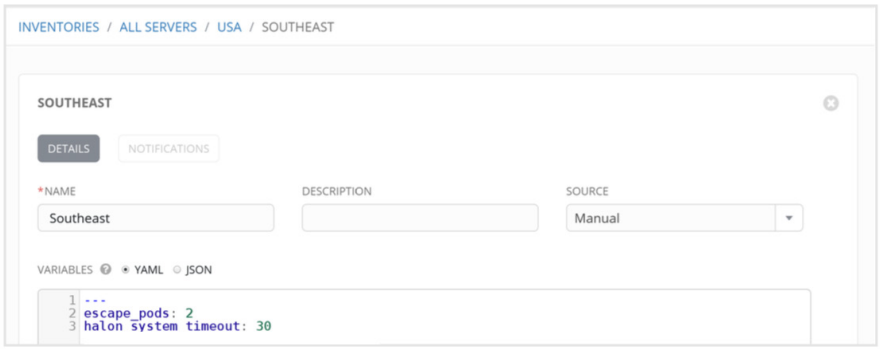
Now that you've created your configuration management and continuous remediation work flows and set up your inventory source of truth, you might think you're done controlling your systems. But day-to-day system management will almost certainly put extra demands on your time. Sometimes you need to restart a service, reboot a machine or perform a one-off patch. With Tower, you can take control of these everyday tasks and simplify them with ease.

ROLE-BASED ACCESS CONTROL AND AUDITING

Take back control over who does what within your Ansible environment. Tower makes delegating specific authority to different teams or explicit users a reality. Keep some projects private. Allow some users to edit inventory and others to run Playbooks against only certain systems — either in check (dry run) or live mode. Allow certain users to use credentials without exposing the credentials to them. Regardless of what you do, Tower records the history of operations and who made them — including objects edited and jobs launched.

CONTROL "JUST-IN-TIME" MANAGEMENT

Often times you just need to do a simple task on a few hosts, whether it's add a single user, update a single security vulnerability or restart a misbehaving service. Tower includes remote command execution. Any task that you can describe as a single Ansible play can be run on a host or group of hosts in your inventory, so you can get up and managing your systems quickly and easily. Plus, it is all backed by Tower's role-based access control engine and detailed audit logging, removing any questions regarding who has done what towhat machines.



SYSTEM TRACKING

Tower's system tracking brings a new level of visibility to your infrastructure so you can see exactly what is happening on your systems, comparing it to both the prior state of the system and to other systems in your cluster, which helps you to ensure compliance. The rich and extensible store of data available in system tracking is accessible via Tower's RESTful API, enabling you to feed it into other tools and systems.

AUTOMATED SAFETY ENFORCEMENT

Ansible easily takes care of instances where machines are out of specification. Perhaps someone made manual changes or the software is misbehaving, whatever the cause it requires investigation. Ansible's flexible nature and Tower's block support allows for the logical grouping of tasks and in-play error handling. Simply schedule a Playbook and Tower will automatically refresh systems that are significantficantly out of spec, including calling back into Tower to apply the basic configuration once new instances are spun up.

BEST PRACTICES FOR CONTROL WITH TOWER

With the layer of control that Tower wraps around your Ansible environment comes responsibility. To ensure you are making the most of Ansible and Tower it's helpful to follow a few best practices. To understand this further, it's important to understand the nature of Ansible automation.

Ansible defines:

Infrastructure in terms of Playbooks

Configurations in terms of machine readable data/code

This serves to eliminate the manual step-based process of configuring machines and replaces it with a build process that represents your infrastructure and applications programmatically.

In an Ansible environment where infrastructure as code is used to manage machines, it follows that you should also treat your infrastructure as if it's code, i.e. apply the same best practices to ensure that your configurations and Playbooks are properly tested before they are deployed live into production environments.

1. USE SOURCE CONTROL

While Tower supports Playbooks stored directly on the Tower server, a better way is to store your Playbooks, Roles and any associated details in source control. This ensures you'll have an audit trail describing when and why you changed the rules that automate your infrastructure.

Plus, it allows for easy sharing of Playbooks with other parts of your infrastructure or team — you can solve a problem once, automate it with Ansible and then share it with Tower. While Tower does allow you to manually upload Playbooks, we highly recommend you use source control.

2. TEST

Just like a code environment, always test configurations and Playbooks before you push them to production. In addition, build a dev environment so you can test your Playbooks before you send them live. By incorporating a degree of testing into your deployment workflow, there will be fewer surprises when code hits production and, in many cases, tests can be leveraged in production to prevent failed updates from migrating across an entire installation.

Since it's push-based, it's also very easy to run the steps on the localhost or testing servers. Ansible lets you insert as many checks and balances into your upgrade workflow as you would like to have. Learn more about how you can best integrate testing with Ansible Playbooks.

IN SUMMARY

Between bringing control to your configuration tasks, managing your inventory and running on-demand remote commands for day-to-day management, Ansible Tower by Red Hat makes it possible to automate most processes and help enterprise teams centralize and simplify their Ansible IT automation initiatives.

Easy-to-use, agentless and with a single view of your entire Ansible environment, Tower lets you watch your systems configure in real-time, with role-based access control and audit- friendly logs of everything that's taking place.

Tower also brings unparalleled self-service to the Ansible automation engine, so you can spread the power of Ansible throughout your organization. For instance, developers or QA departments can provision their own dev and test environments. Customer service agents can provision a new demo environment. Or junior admins can run simple jobs — like changing passwords — all at the press of a button. With Tower a culture of success comes as standard — overcome complexity, eliminate repetitive tasks and errors, be more productive and improve job collaboration and satisfaction.

ABOUT ANSIBLE

Ansible, an open source community project sponsored by Red Hat, is the simplest way to automate IT. Ansible is the only automation language that can be used across entire IT teams – from systems and network administrators to developers and managers. Ansible by Red Hat provides enterprise-ready solutions to automate your entire application lifecycle – from servers to clouds to containers and everything in between. Ansible Tower by Red Hat is a commercial offering that helps teams manage complex multi-tier deployments by adding control, knowledge, and delegation to Ansible-powered environments.

SOLVE IT. AUTOMATE IT. SHARE IT.

ansible.com/tower

WATCH. LEARN. SHARE.

ansible.com/resources

COLLABORATE. DISCUSS. CONTRIBUTE.

ansible.com/community

redhat.