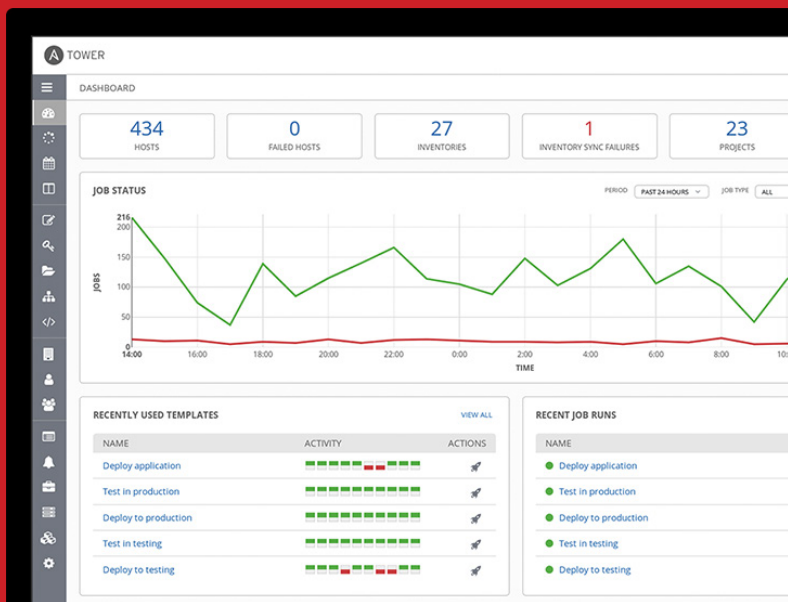
RED HAT ANSIBLE TOWER **IN THE SOFTWARE DEVELOPMENT LIFECYCLE**



"Ansible Tower Enterprise is a critical part of our infastructure. With Tower there is no downtime and we can easily schedule Ansible Playbooks to run when we want."

YVES SOELE
IT OPERATIONS, AMELCO

INTRODUCTION

Red Hat Ansible 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 deploy their infrastructure consistently and repeatedly. NASA uses Ansible to update security vulnerabilities and to patch manage nasa. gov weekly. Enterprises that make money delivering applications via the web find that Ansible Tower excels at removing IT bottlenecks, automating repetitive tasks, and accelerating the delivery of applications to market. For IT operations, rolling updates with zero downtime is a very common orchestration pattern that has been discussed at length, but let's take a look at how Ansible Tower fits earlier into the software development lifecycle and removes bottlenecks in operations, as well as in development and test.

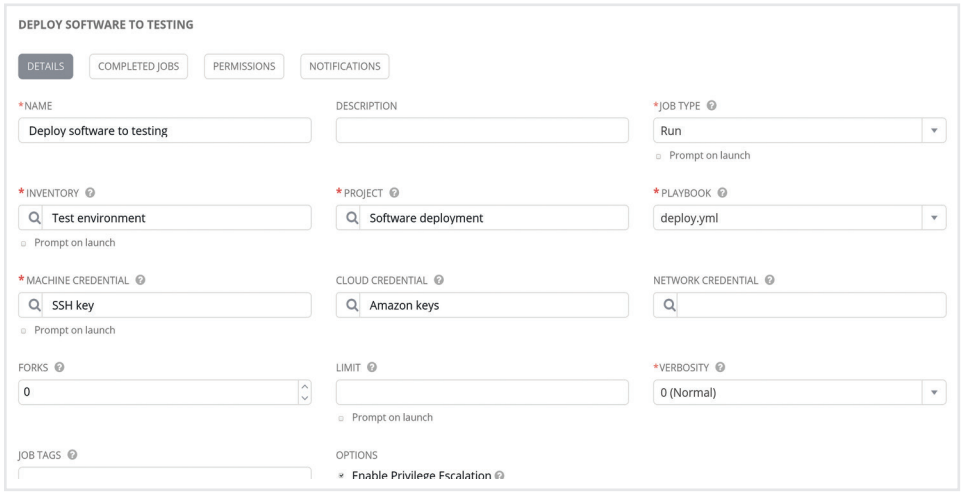
Ansible Tower offers a number of benefits in the SDLC, specifically to:

- Drive consistency between environments
- Enable self-service
- Improve automated testing

Ansible Tower provides a pushbutton interface that templates how a Playbook should be run — deploying consistently and with repeatability.

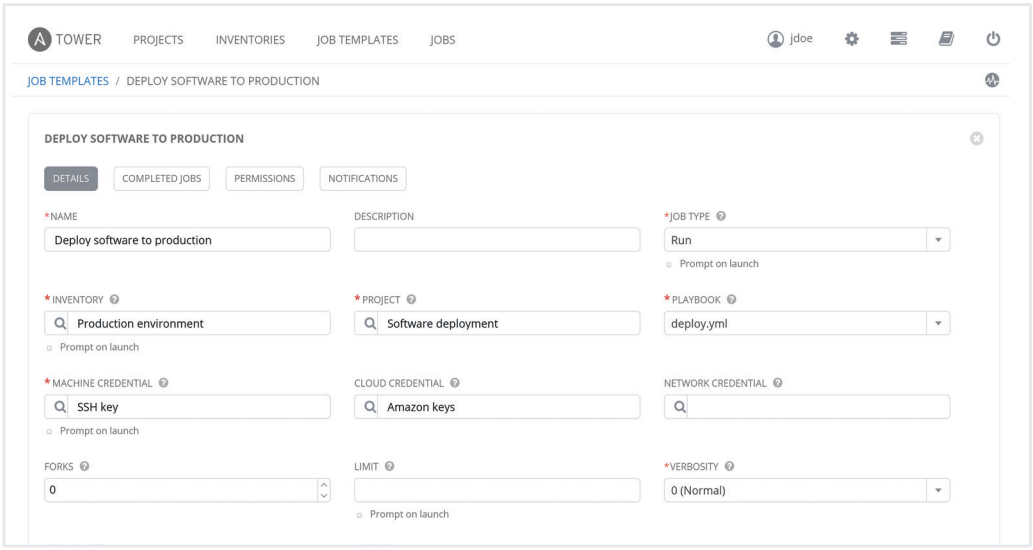
DEVELOPMENT

A typical day in the life of an application developer may be to: check out code, make changes, test locally, check in changes. During the course of this work, the development environment may become infected with small configuration tweaks and hand-crafted software stacks that differ dramatically from the environment that production code is ultimately deployed to. In fact, a common problem in many organizations is that the development environment may not exactly (or even closely) match what's in QA, Stage, and Release. Generally, the further away from production you get, the less and less the environments look the same. (Not just the obvious examples like software library versions, but more subtle items like configuration settings, authentication details, etc.). Inconsistency between code and the software stack that supports the application is a common cause of software errors and resulting production downtime. With Ansible, a Playbook captures the desired state of the stack and even the entire environment. Ansible Tower provides a pushbutton interface that templates how a Playbook should be run — deploying consistently and with repeatability. With Ansible Tower, a Playbook is run from a job template, which specifies the parameters and environment details of how to run the Playbook, with all of the parameters and options you would normally pass via the Ansible command line. There is no concern that the user will mistype an option or parameter to Ansible, potentially deploying to the wrong environment or with the wrong Playbook arguments since the template is triggered via the push of a button (or API.)



Not only do these templates drive consistency between deployments — each time the button is pressed the same result is expected — but they also encourage a tremendous amount of Playbook reuse: Multiple job templates can be created that launch the same identical Playbook with different parameters. This enables the same Playbook to be used between environments or stages of release. The job template can use the same Playbook, but target different inventories (corresponding to different stages of release or different environments), pass different options, even use different credentials.

This drives consistency between releases, even across different stages, by ensuring that the same steps to automate provisioning and deployment are followed between each stage.



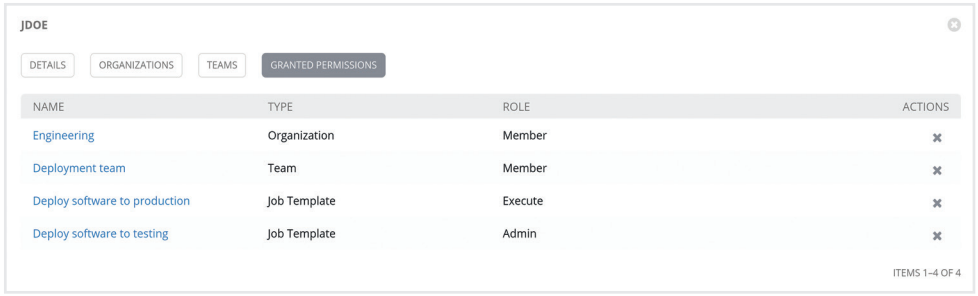
When combined with the role-based access control capabilities of Ansible Tower, Playbooks can be deployed at the push of a button, but in a controlled and easily-audited way. This allows an organization to spread the power of Ansible to any of its users. Even users unfamiliar with Ansible can run Playbooks that others have written. These features are commonly used to enable self-service provisioning of development environments by the developers themselves.

Typically, developers manually construct a development environment that starts out only loosely coupled to the production software stack and deviates further and further over time. Alternatively, developers are forced to request a new development environment from operations teams, usually incurring delays that proliferate throughout the software development lifecycle and lead to late deployments, missed deadlines, and lost business.

Tower enables developers to check out and provision their own development environment, built from scratch and 100% consistent with the software stack used in test, stage, and production.

Tower enables developers to check out and provision their own development environment, built from scratch and 100% consistent with the software stack used in test, stage, and production — because it is built with the same Playbook. Developers log into Ansible Tower and, under RBAC, see only the job templates that they have permission to access and deploy.

The developer presses a few buttons in Ansible Tower, goes and gets a cup of coffee, and then returns to a newly 100% provisioned environment.



Developers do not require any knowledge of Ansible or the Playbook. They can provide new parameters and options, specifying perhaps the release tag of the application to be deployed or fine tuning the deployment to fit within the constraints of the environment. The Playbook handles provisioning the new servers (e.g. by calling out to VMware or Amazon Web Services), deploying and configuring the software stack, and the actual application under development. The developer presses a few buttons in Ansible Tower, goes and gets a cup of coffee, and then returns to a newly 100% provisioned environment. Tear down and re-provisioning is a snap, so a new environment for pristine development can be deployed at any time.

QA

A typical day for QA might look like this: Program a CI tool to automatically build the codebase periodically, deploy to a test environment, run tests, report results. At most shops, some of this work is generally automated, usually building the latest code and the deploying the latest code artifacts into test. Deployment of the application stack may not be automated (or easily refreshed between test cycles), leading to the same consistency problem as in development,or worse, because test environments and artifacts are potentially reused between test cycles.

Here again, the reuse of Playbooks between environments through the Ansible Tower job template will promote consistency between stages of release. Additionally, the same Ansible Tower self-service capability that can be provided to developers can also be provided to testers. QA may write their own Playbooks for automated testing and Ansible Tower can be used to delegate responsibility to QA staff who are not Ansible experts or aren't comfortable running Ansible on the command line.

In addition to self-service, it is very common to use Ansible Tower in conjunction with a Continuous Integration system in QA to automate not only application deployment in test, but provisioning of the application stack to refresh the test environment between tests. This way, Ansible Tower provides a complete RESTful API that powers the Ansible Tower user interface and makes it possible to programmatically invoke Ansible Tower to do any function that can be performed via point and click in the UI. In this way, a CI tool like Jenkins can be connected to Tower to launch job templates for automated testing. Jenkins is configured to call the Ansible Tower API to launch a specific job template that invokes an Ansible Playbook to provision a new QA environment, deploy and configure the appropriate software stack, the correct version of the application, and the test suite. Automated testing can be performed and the environment decommissioned, all within one or more Playbook runs. Just as in development, this ensures that the test environment is consistent with all release stages — that what is being tested is the same as what was developed and what is being deployed into production. Test environments can quickly be provisioned and deployed 100% from scratch, improving your ability to test and to speed up the application to production delivery. When used in conjunction with a CI tool like Jenkins, the combination is more powerful than with a CI tool alone: Jenkins can kick off Ansible Playbooks to provision environments and deploy the latest code as part of any Jenkins CI job, but under the framework of role-based access control, inventory management, and aggregated job output that Ansible Tower provides.

As you adopt Ansible Tower into Operations, consider also the benefits of Ansible and Ansible Tower throughout your software development lifecycle. Apply automation to eliminate bottlenecks through every stage of release and accelerate the delivery of your applications to market.

RED HAT ANSIBLE TOWER

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. Red Hat Ansible Tower is a commercial offering that helps teams manage complex multi-tier deployments by adding control, knowledge, and delegation to Ansible-powered environments.