



Red Hat Ansible Tower

What's New in version 3.5 – An Overview

UI Enhancements

The screenshot shows the 'TEMPLATES' page in the Ansible Tower UI. At the top, there's a search bar, a 'KEY' button, and a green '+' button. Below the header, there's a table with four rows of templates:

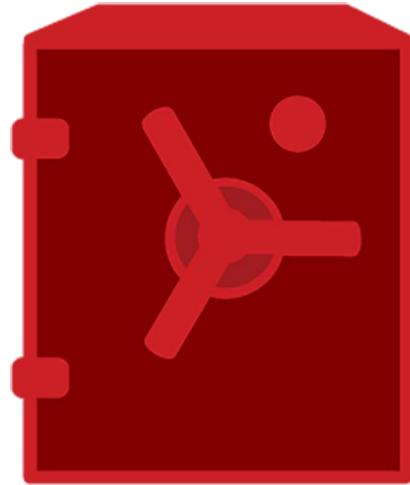
NAME	TYPE	INVENTORY	PROJECT	LAST MODIFIED	LAST RAN
Connect with secrets	Job Template	localhost	Secrets deployment	5/3/2019 1:44:03 PM by admin	4/25/2019
Deploy application	Job Template		App deployment	4/24/2019 4:05:28 PM by admin	5/17/2019 11:29:00 AM
Deploy database	Job Template		Pipelines	4/24/2019 4:05:29 PM by admin	5/17/2019 11:29:03 AM

Each row has a 'CREDENTIALS' section below it. The first row has 'Operations AWS key' and 'Super Secret Credential'. The second row has 'AWS key' and 'Super Secret Credential'. The third row has 'AWS key' and 'Super Secret Credential'. The fourth row has 'AWS key' and 'Super Secret Credential'. To the right of the table, a dropdown menu titled 'SORT BY' is open, showing options like 'Name (Ascending)', 'Name (Descending)', 'Modified (Ascending)', etc.

Extended list display

Expand and collapse lists as needed, including sorting.

Support for External Credential Vaults



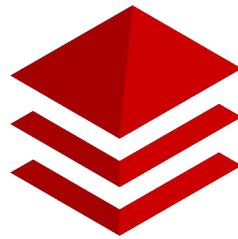
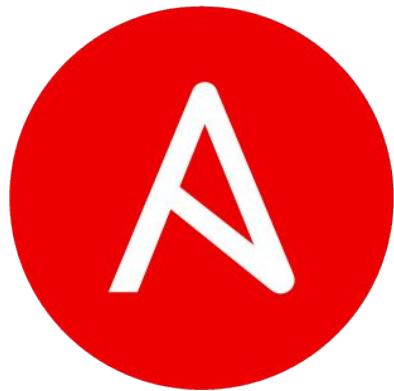
Introducing Credential Plugins

Use credentials from your corporate standard password and key storage directly from Tower.

- HashiCorp Vault
- CyberArk AIM
- CyberArk Conjur
- Microsoft Azure Key Vault



Enabling Advanced Ansible Features



Inventory Plugins

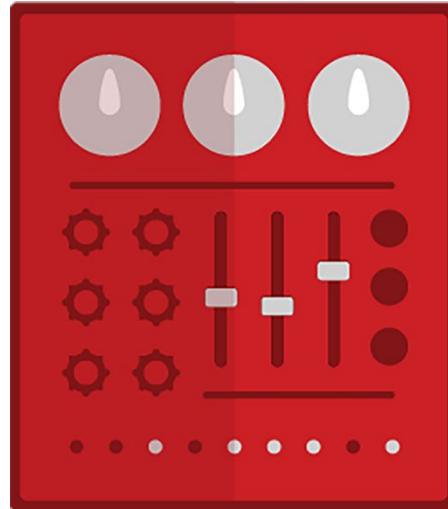
Azure
GCE
OpenStack
Tower



Privilege Escalation Plugins

Handle complex privilege escalation in
your enterprise environment.

Metrics for Monitoring



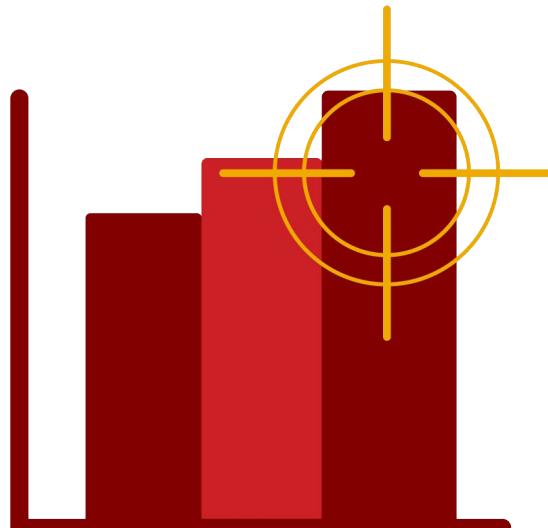
Ansible Tower Health And Stats at a Glance

New, Prometheus-compatible metrics

Available at `/api/v2/metrics`



Enabling Intelligence for Automation



Introducing Automation Insights

You've got questions on your automation.

We'll help you get answers.

Tower now collects automation information via the Insights Platform, to be delivered via Automation Insights later this year.



Now supporting: Red Hat Enterprise Linux 8
Ansible Tower now deploys on and manages
RHEL 8.

Assorted minor enhancements

- Added per-organization license limits
- Removed 1024-character limit on --limit fields
- Removed the restriction that schedules must be uniquely named
- Allow custom virtual environments to be at an alternate path
- Added Grafana notification type
- Can pass `ipv6_deployment=True` when installing (TECH PREVIEW)
- Additional job details shown
 - custom environment used
 - artifacts stored
 - execution node

Notes for operators

DEPRECATIONS, REMOVALS, AND BEHAVIOR CHANGES

- **Deprecation** of /api/v1 - **will be removed in 3.6.**
- **Deprecation** of Ubuntu as a Tower platform - **will be removed in 3.6.**
- **Deprecation** of several inventory computed fields in the API, and the dashboard API endpoint
See the release notes for details
- **Removal** of deprecated system tracking endpoints
- **Change** to licensing behavior - no feature distinction between
basic/self-support/standard/premium licenses
- **Change** to dashboard job count behavior
Implicit jobs (SCM checkouts, etc.) no longer counted
- **Change** to playbook detection in projects
Playbook names must be valid UTF-8

Notes for operators

DEPRECATIONS, REMOVALS, AND BEHAVIOR CHANGES

- Tower is now a **Python 3** application
 - Using software collections on RHEL 7, using native python on RHEL 8.
 - Ansible uses system python (this means **python 3** on RHEL 8)
 - Python3-incompatible code must be removed from `/etc/tower/conf.d/*.py` before upgrade
 - `/etc/tower/settings.py` will be overwritten on upgrade
- All playbooks and tasks run via **ansible-runner**, including on isolated nodes
- Added an index for job event creation on upgrade
 - Upgrades may take longer on large systems (still measured in minutes, not hours)

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.



[linkedin.com/company/red-hat](https://www.linkedin.com/company/red-hat)



[youtube.com/user/RedHatVideos](https://www.youtube.com/user/RedHatVideos)



[facebook.com/redhatinc](https://www.facebook.com/redhatinc)



twitter.com/RedHat

Section #2 – Best Practices

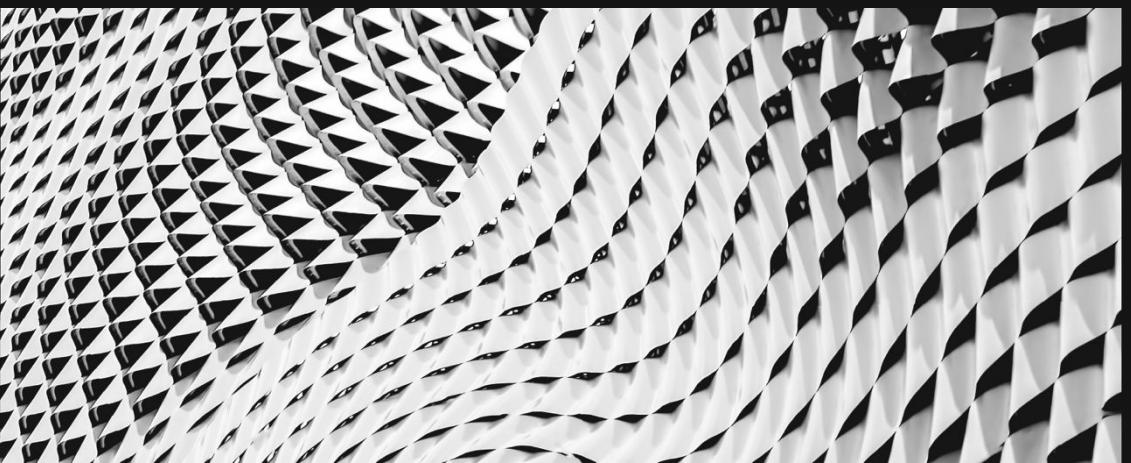
Time-Permitting Overview – Takeaway Slides
Use for Customer (Re)-engagement



Ansible Best Practices

How to write, how to execute, and how to use in real life

How to use



Treat your Ansible content like code

- Version control your Ansible content
- Iterate
 - Start with a basic playbook and static inventory
 - Refactor and modularize later

Do it with style

- Create a style guide for consistency:
 - Tagging
 - Whitespace
 - Naming of Tasks, Plays, Variables, and Roles
 - Directory Layouts
- Enforce the style
- Nice example: openshift-ansible Style Guide
example: <https://goo.gl/JfWBcW>

A large pile of colorful puzzle pieces of various shapes and sizes, scattered across the entire background.

**CODE MUST BE
ORGANIZED**

USE GIT!

Do it with style

```
site.yml                      # master playbook, calling others
webservers.yml                 # playbook for webserver tier
deployonce.yml                 # separate playbook for single-shot tasks
inventories/
    production/              # different stages via inventory
        hosts                  # inventory file for production servers
        group_vars/
        host_vars/
    london/                  # additional, alternative grouping if useful
roles/
    requirements.yml          # includes roles from some other place
    common/                   # base line, company wide configuration
    webtier/
```

Start with one Git repository - but when it grows,
use multiple!

At the beginning: put everything in one Git repository

In the long term:

- One Git repository per role
- Dedicated repositories for completely separated teams / tasks

New to git? Get your cheat sheet here: <https://opensource.com/downloads/cheat-sheet-git>

SO, WHAT DO WE HAVE?



Give inventory nodes human-meaningful names rather than IPs or DNS hostnames.

10.1.2.75

10.1.5.45

10.1.4.5

10.1.0.40



db1 ansible_host=10.1.2.75

db2 ansible_host=10.1.5.45

db3 ansible_host=10.1.4.5

db4 ansible_host=10.1.0.40

w14301.acme.com

w17802.acme.com

w19203.acme.com

w19304.acme.com

web1 ansible_host=w14301.acme.com

web2 ansible_host=w17802.acme.com

web3 ansible_host=w19203.acme.com

web4 ansible_host=w19304.acme.com

Group hosts for easier inventory selection and less conditional tasks -- the more the better.

[db]	[east]	[dev]
db[1:4]	db1	db1
	web1	web1
[web]	db3	
web[1:4]	web3	[testing]
		db3
	[west]	web3
	db2	
	web2	[prod]
	db4	db2
	web4	web2
		db4
		web4

Use dynamic sources where possible. Either as a single source of truth - or let Ansible unify multiple sources.

- Stay in sync automatically
- Reduce human error
- No lag when changes occur
- Let others manage the inventory

VARIABLES

JUST WORDS,
RIGHT?

Proper variable names can make plays more readable and
avoid variable name conflicts

a: 25
data: ab
data2: abc
id: 123

apache_max_keepalive: 25
apache_port: 80
tomcat_port: 8080

Avoid collisions and confusion by adding the role name to a variable as a prefix.

```
apache_max_keepalive: 25  
apache_port: 80  
tomcat_port: 8080
```

Know where your variables are

- Find the appropriate place for your variables based on what, where and when they are set or modified
- Separate logic (tasks) from variables and reduce repetitive patterns
- Do not use every possibility to store variables - settle to a defined scheme and as few places as possible



NO!

```
- name: install telegraf
  yum: name=telegraf-{{ telegraf_version }} state=present update_cache=yes
  notify: restart telegraf

- name: start telegraf
  service: name=telegraf state=started
```

Better, but no

```
- name: install telegraf
  yum: >
    name=telegraf-{{ telegraf_version }}
    state=present
    update_cache=yes
    enablerepo=telegraf
  notify: restart telegraf

- name: start telegraf
  service: name=telegraf state=started
```

Yes!

```
- name: install telegraf
  yum:
    name: "telegraf-{{ telegraf_version }}"
    state: present
    update_cache: yes
    enablerepo: telegraf
    notify: restart telegraf

- name: start telegraf
  service:
    name: telegraf
    state: started
```

Exhibit A

```
- hosts: web
  tasks:
    - yum:
        name: httpd
        state: latest
    - service:
        name: httpd
        state: started
        enabled: yes
```

```
PLAY [web]
*****
TASK [setup]
*****
ok: [web1]

TASK [yum]
*****
ok: [web1]

TASK [service]
*****
ok: [web1]
```

Exhibit B

```
- hosts: web
  name: installs and starts apache

  tasks:
    - name: install apache packages
      yum:
        name: httpd
        state: latest

    - name: starts apache service
      service:
        name: httpd
        state: started
        enabled: yes
```

```
PLAY [install and starts apache]
*****
TASK [setup]
*****
ok: [web1]

TASK [install apache packages]
*****
ok: [web1]

TASK [starts apache service]
*****
ok: [web1]
```

The background of the image is a wall constructed from large, rectangular stone blocks. The stones are of various colors, including shades of brown, tan, green, and blue, and have a rough, textured surface. They are laid in a staggered pattern, creating a sense of depth and strength.

POWERFUL
BLOCKS

Blocks can help in organizing code, but also enable rollbacks or output data for critical changes.

```
- block:  
  copy:  
    src: critical.conf  
    dest: /etc/critical/crit.conf  
  service:  
    name: critical  
    state: restarted  
  rescue:  
    command: shutdown -h now
```

How to execute





**PROPER
LAUNCHING**

Ansible provides multiple switches for command line interaction and troubleshooting.

- vvvv
- step
- check
- diff
- start-at-task

Ansible has switches to show you what will be done

Use the power of included options:

- list-tasks
- list-tags
- list-hosts
- syntax-check

If there is a need to launch something without an inventory
- just do it!

- For single tasks - note the comma:

```
ansible all -i neon.qxyz.de, -m service -a  
"name=redhat state=present"
```

- For playbooks - again, note the comma:

```
ansible-playbook -i neon.qxyz.de, site.yml
```

THE RIGHT TOOLS



Don't just start services -- use smoke tests

```
- name: check for proper response
  uri:
    url: http://localhost/myapp
    return_content: yes
  register: result
  until: '"Hello World" in result.content'
  retries: 10
  delay: 1
```

Try to avoid the command module - always seek out a module first

- name: add user
command: useradd appuser
- name: install apache
command: yum install httpd
- name: start apache
shell: |
 service httpd start && chkconfig
httpd on
- name: add user
user:
 name: appuser
 state: present
- name: install apache
yum:
 name: httpd
 state: latest
- name: start apache
service:
 name: httpd
 state: started
 enabled: yes

If managed files are not marked, they might be overwritten accidentally

- Label template output files as being generated by Ansible
- Use the `ansible_managed**` variable with the `comment` filter

```
{{ ansible_managed | comment }}
```

ROLES AND GALAXIES



Roles enable you to encapsulate your operations.

- Like playbooks -- keep roles purpose and function focused
- Store roles each in a dedicated Git repository
- Include roles via `roles/requirements.yml` file, import via `ansible-galaxy` tool
- Limit role dependencies

Get roles from Galaxy, but be careful and adopt them to your needs

- Galaxy provides thousands of roles
- Quality varies drastically
- Take them with a grain of salt
- Pick trusted or well known authors



ACCESS RIGHTS

Root access is harder to track than sudo - use sudo wherever possible

- Ansible can be run as root only
- But login and security reasons often request non-root access
- Use become method - so Ansible scripts are executed via sudo (sudo is easy to track)
- Best: create an Ansible only user
- Don't try to limit sudo rights to certain commands - Ansible does not work that way!

DEBUG YOUR PROBLEM



Check logging on target machine

```
ansible-node sshd[2395]: pam_unix(sshd:session): session  
    opened for user liquidat by (uid=0)  
ansible-node ansible-yum[2399]: Invoked with name=['httpd']  
    list=None install_repoquery=True conf_file=None  
    disable_gpg_check=False state=absent disablerepo=None  
    update_cache=False enablerepo=None exclude=None
```

How to keep the code executed on the target machine

Look into the logging of your target machine

```
$ ANSIBLE_KEEP_REMOTE_FILES=1 ansible target-node -m yum  
-a "name=httpd state=absent"
```

Execute with:

```
$ /bin/sh -c 'sudo -u $SUDO_USER /bin/sh -c  
"/usr/bin/python /home/liquidat/.ansible/tmp/..."'
```

Debugging tasks can clutter the output, apply some housekeeping

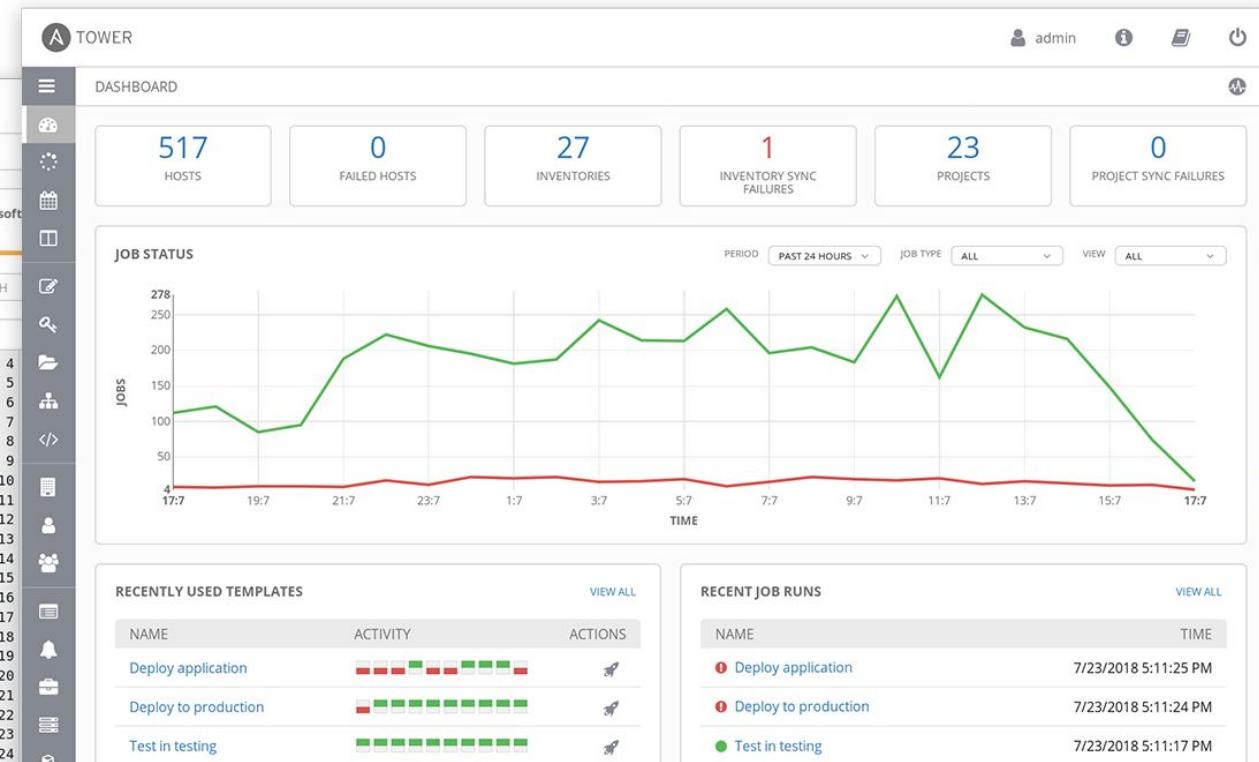
- name: Output debug message
debug:
 msg: "This always displays"

- name: Output debug message
debug:
 msg: "This only displays with ansible-playbook -vvv"
 verbosity: 2

How to use in real life

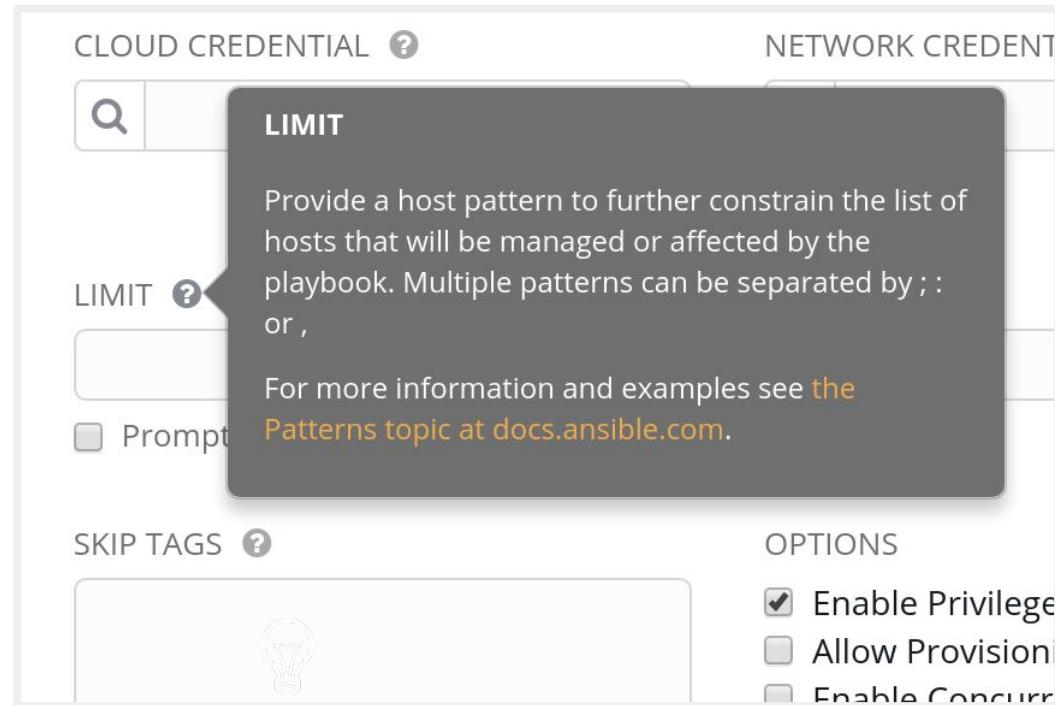


Simple: Use Tower.



- Tower was developed with Ansible in mind
- Extends the limits of Ansible to meet enterprise needs:
Scalability, API, RBAC, audits, etc.

Tower has inbuilt help



- Tower provides in-program help via questionmark bubbles
- Can include examples or links to further docs



BRANCHES, ANYONE?

Tower can import a repository multiple times with different branches

- Use feature or staging branches in your Git
- Import them all separately, address them separately
- Useful for testing of new features but also to move changes through stages

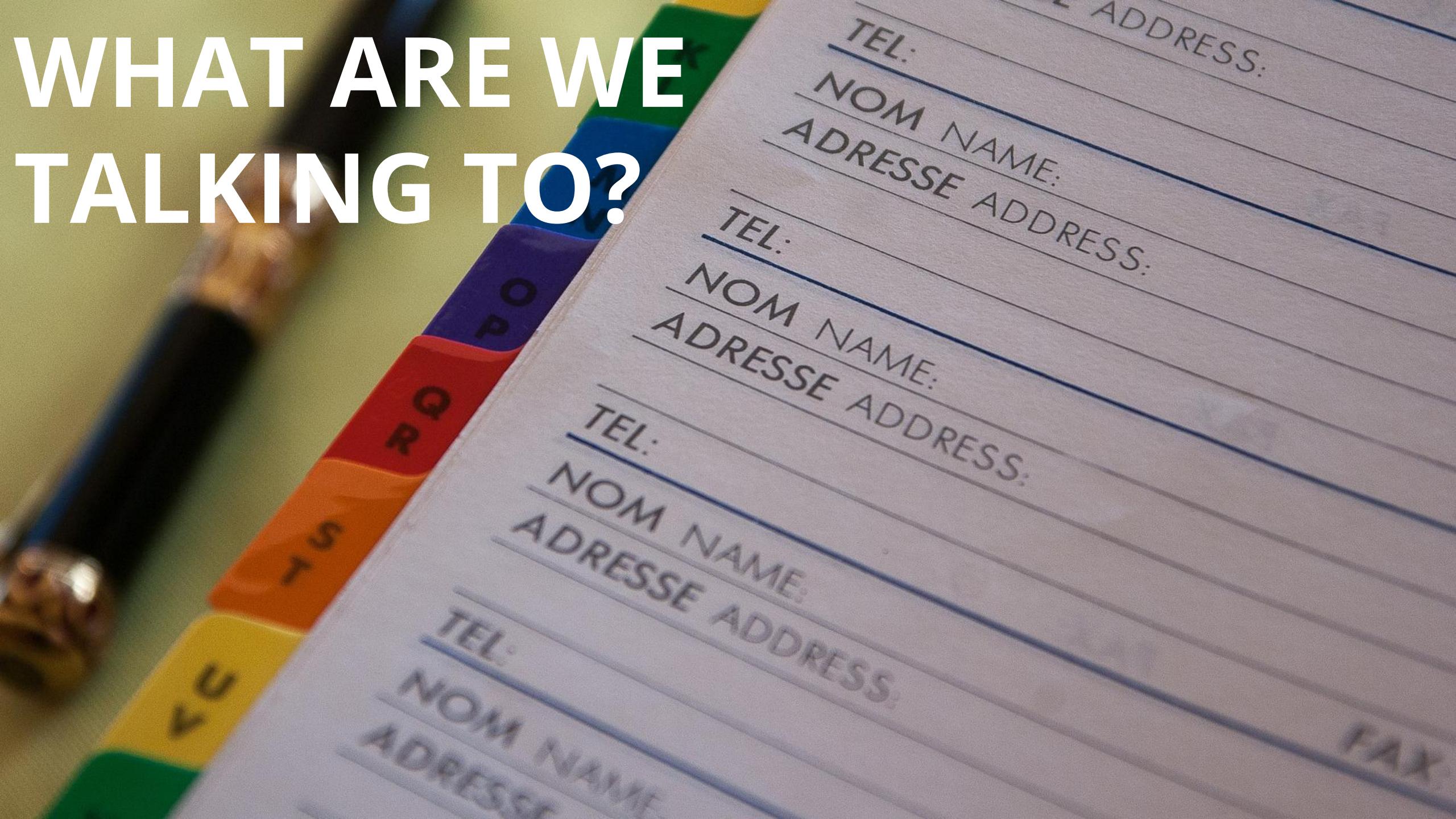


MANY, MANY ROLES

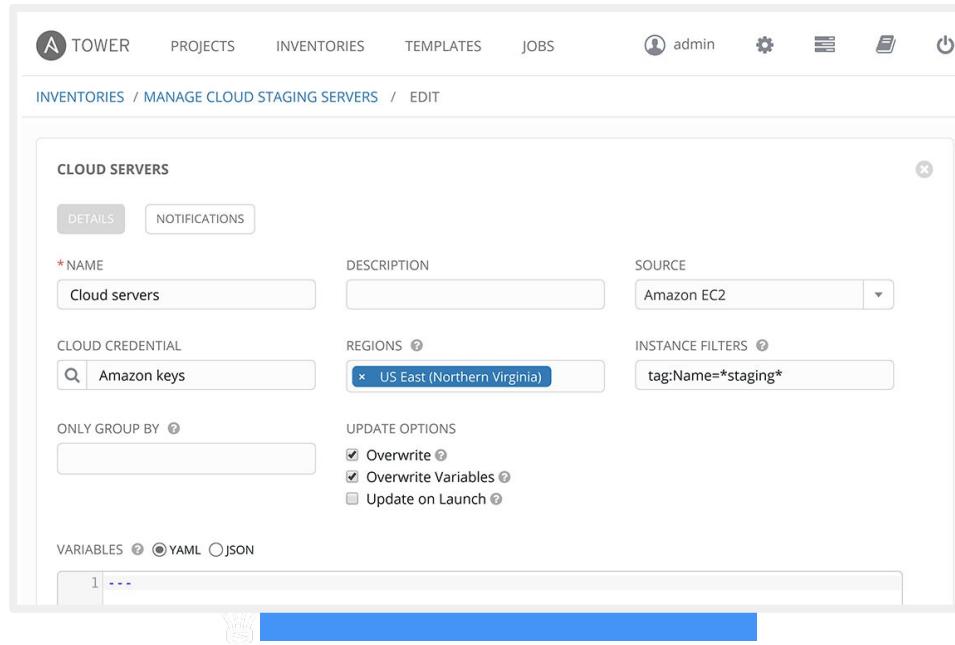
Tower automatically imports Roles during Project update

- Do not copy roles into your playbook repository, just create a roles/requirements.yml
- Tower will automatically import the roles during Project installation
- Mix roles from various sources
- Fix version in roles/requirements.yml to have auditable environment!

WHAT ARE WE TALKING TO?



Use dynamic & smart inventories



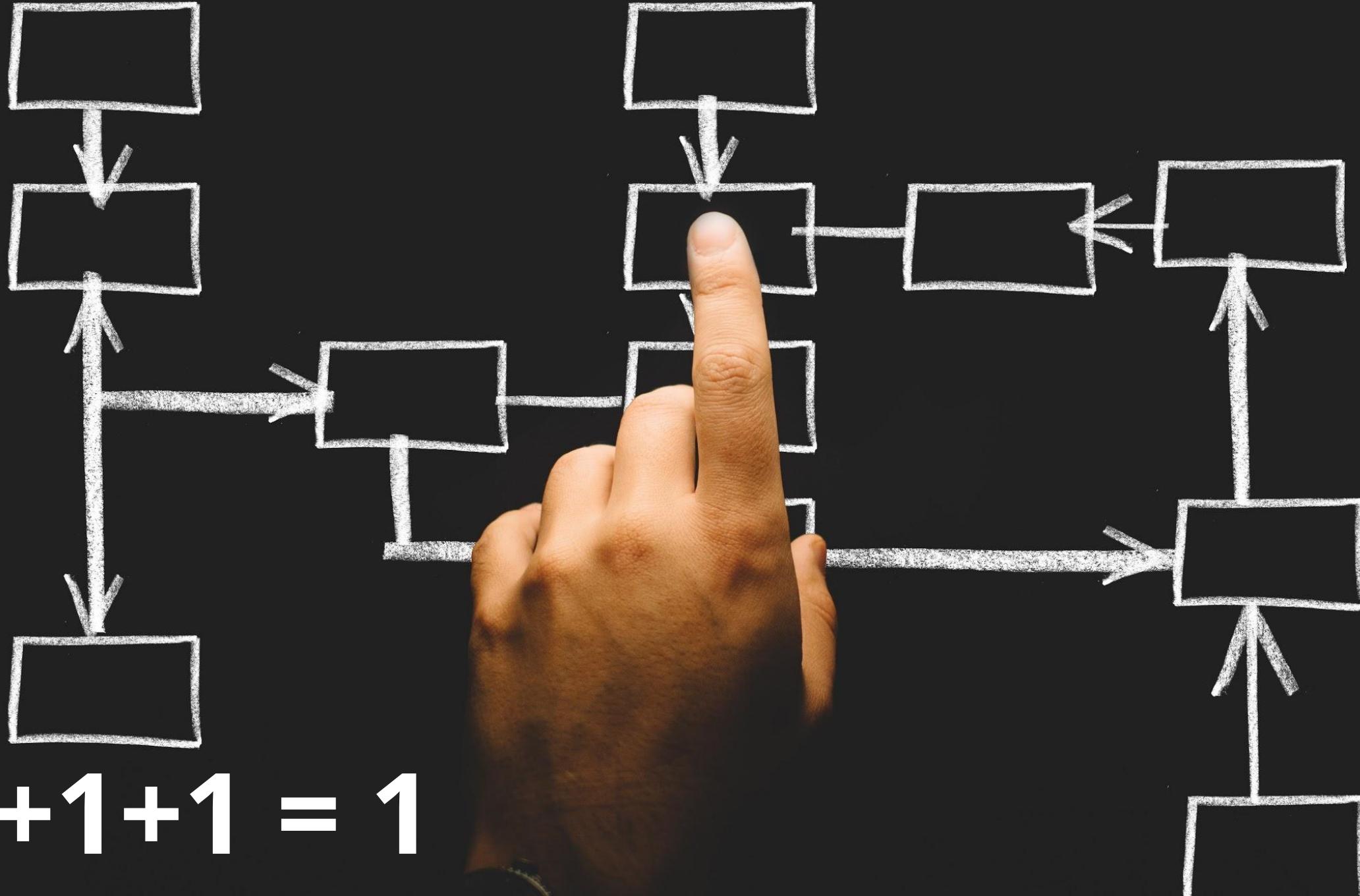
- Combine multiple inventory types
- Let Tower take care of syncing and caching
- Use smart inventories to group nodes



DOING GOOD JOBS

Tower job templates provide multiple options - use them wisely

- Keep jobs simple, focussed - as playbooks or roles
- Add labels to them to better filter
- For idempotent jobs, create “check” templates as well - and let them run over night
- Combine with notifications - and get feedback when a “check” failed



$$1+1+1 = 1$$

Multiple playbooks can be combined into one workflow

- Simple jobs, complex workflows
- React to problems via workflow
- Combine playbooks of different teams, different repositories
- Re-sync inventories during the play

WHO How WHAT
WHEN WHERE WHY

DO ASK PROPER QUESTIONS

Use surveys to get variable values

* PROMPT

DESCRIPTION

* ANSWER VARIABLE NAME ?

* ANSWER TYPE

MINIMUM LENGTH

 MAXIMUM LENGTH

DEFAULT ANSWER

- Use good, meaningful variable names
- Provide a default choice
- Multiple choice > free text
- If answer not required - do you really need it at all?



A POWERFUL TEAM

Tower provides tenants, teams, and users - use them for separation

- Provide automation to others without exposing credentials
- Let others only see what they really need
- Use personal view instead of full Tower interface



ONE KEY TO RULE
THEM ALL ...

Tower credentials should only be used by Tower - not by others

- Set up a separate user and password/key for Tower
- That way, automation can easily be identified on target machines
- The key/password can be ridiculously ~~complicated~~ secure
- Store key/password in a safe for emergencies

NOTIFY YOURSELF!



MF'D BY
SOUTHERN GEMINI
TAYLORSVILLE, MS

39163

Tower can send notifications if a job succeeds, fails or always - as mail, IRC, web hook, and so on

- Let Tower notify you and your team if something breaks
- Send mails/web-hooks automatically to a ticket systems and monitoring if there is a serious problem



LOGS, ANYONE?

Send all logs from Tower to central logging

- Splunk, Loggly, ELK, REST
- Send results from Ansible runs - but also from Tower changes



**ALWAYS KEEP
THE LIGHTS ON**

Tower can be easily set up HA - and for restricted networks,
deploy isolated nodes

- Make Tower HA - it is easy! (Well, except the DB part maybe....)
- For distant or restricted networks, use isolated nodes

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.



[linkedin.com/company/red-hat](https://www.linkedin.com/company/red-hat)



[youtube.com/user/RedHatVideos](https://www.youtube.com/user/RedHatVideos)



[facebook.com/redhatinc](https://www.facebook.com/redhatinc)



twitter.com/RedHat