



Ansible Tower Workflow

Creating an Application Service

Brian Dumont, RHCE
Sr Solution Architect

Agenda

Tower Overview

Workflow Concepts

Workflow to deploy an Application Service

Ad-hoc Automation is happening in silos



Developers

Ansible used in silo



Security

DIY scripting automation



Infrastructure

Open source config management tool

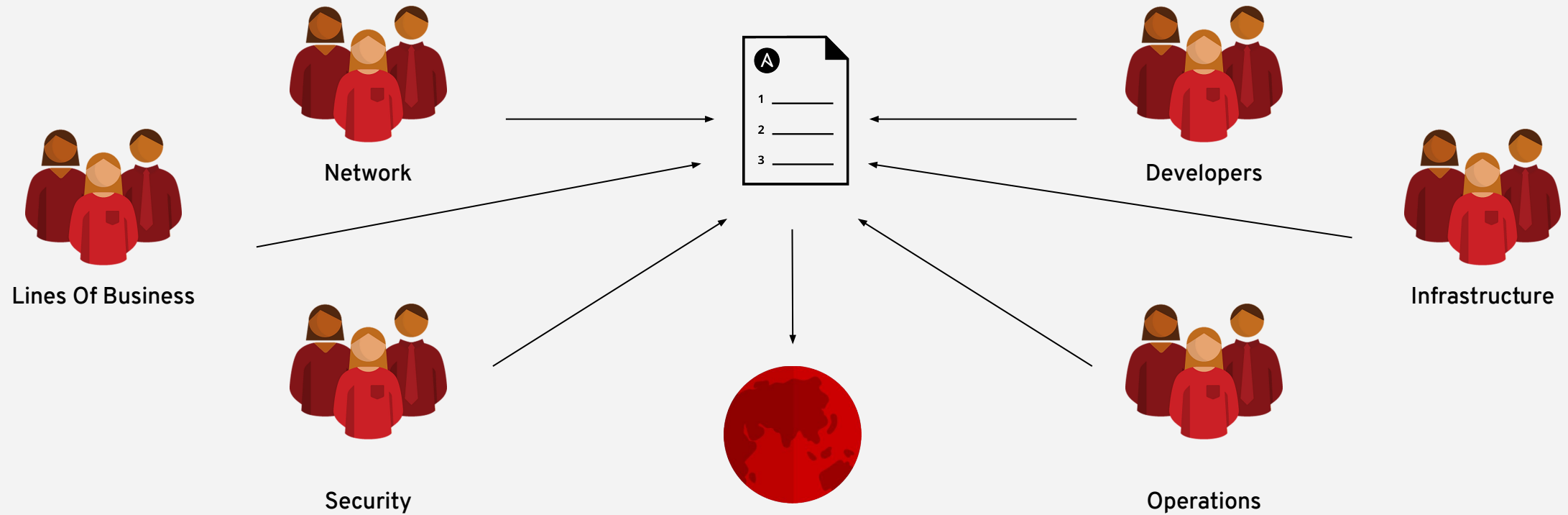


Network

Proprietary vendor supplied automation

Is organic automation enough?

When automation crosses teams, you need an automation platform



Red Hat Ansible Automation Platform



Network

Lines of
business

Security

Operations

Infrastructure

Developers

Engage

Ansible SaaS: Engage users with an automation focused experience

Scale

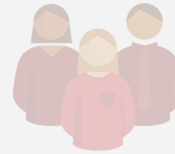
Ansible Tower: Operate & control at scale

Create

Ansible Engine: Universal language of automation

Fueled by an open source community

Red Hat Ansible Automation Platform



Network

Lines of
business

Security

Operations

Infrastructure

Developers

Engage

Ansible SaaS: Engage users with an automation focused experience

Scale

Control

Web UI and API

Delegation

Role Based Access Controls

Scale

Scalable Execution Capacity

Create

Ansible Engine: Universal language of automation

Fueled by an open source community



VIEWS

Dashboard

Jobs

Schedules

My View

RESOURCES

Templates

Credentials

Projects

Inventories

Inventory Scripts

ACCESS

Organizations

Users

Teams

ADMINISTRATION

Credential Types

Notifications

Management Jobs

Instance Groups

Applications

Settings

DASHBOARD

32

HOSTS

4

FAILED HOSTS

13

INVENTORIES

1

INVENTORY SYNC
FAILURES

6

PROJECTS

0

PROJECT SYNC FAILURES

JOB STATUS

PERIOD

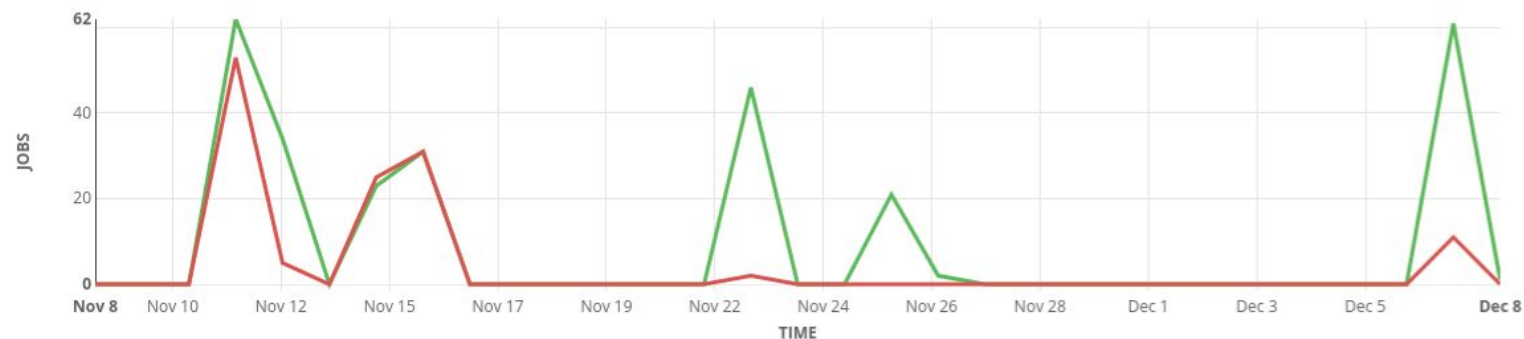
PAST MONTH

JOB TYPE

ALL

VIEW

ALL



RECENTLY USED TEMPLATES

[VIEW ALL](#)

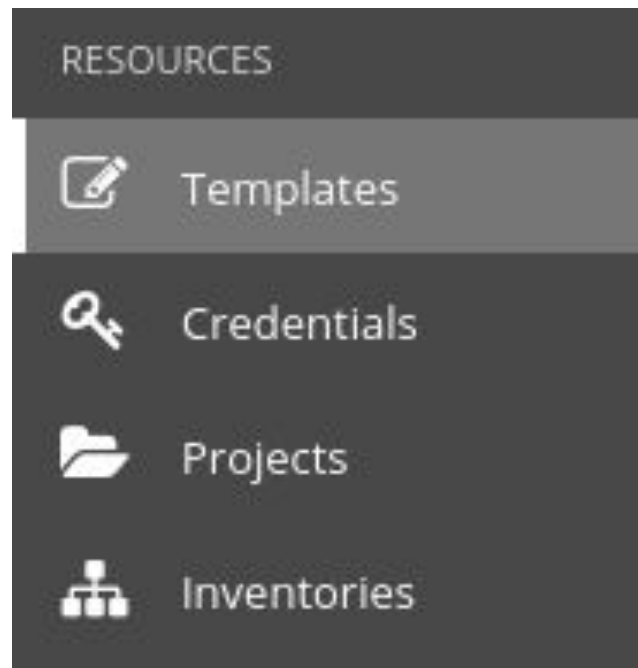
NAME	ACTIVITY	ACTIONS
Lear Demo	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	
Create DB_server Decomm schedule	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	
Install and Configure Apache	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	
Subscribe_RHEL_to_Sat	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	
Create ec2 Instance	<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>	

RECENT JOB RUNS

[VIEW ALL](#)

NAME	TIME
Lear Demo	12/6/2019 11:06:27 PM
Create DB_server Decomm schedule	12/6/2019 11:06:26 PM
Install and Configure Apache	12/6/2019 11:05:44 PM
Subscribe_RHEL_to_Sat	12/6/2019 11:05:18 PM
Create ec2 Instance	12/6/2019 11:04:13 PM

Ansible Tower Resources



Credentials

utilized by Ansible Tower for authentication with various external resources

Projects

Logical collection of Ansible Playbooks, represented in Ansible Tower

Inventories

a collection of hosts (nodes) with associated data and groupings that Ansible Tower can connect to and manage

Templates

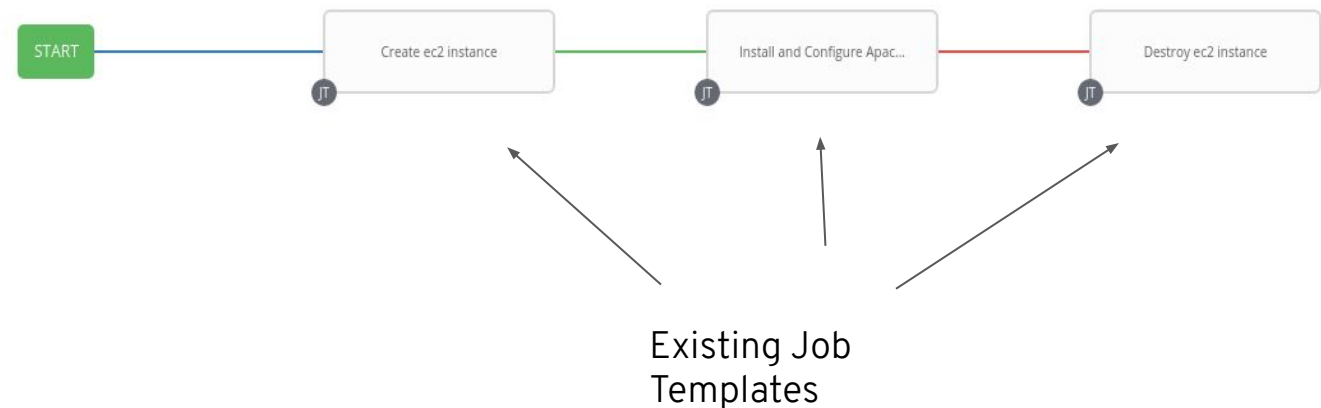
allow Ansible Playbooks to be controlled, delegated and scaled for an organization

Workflow Concepts

Workflows

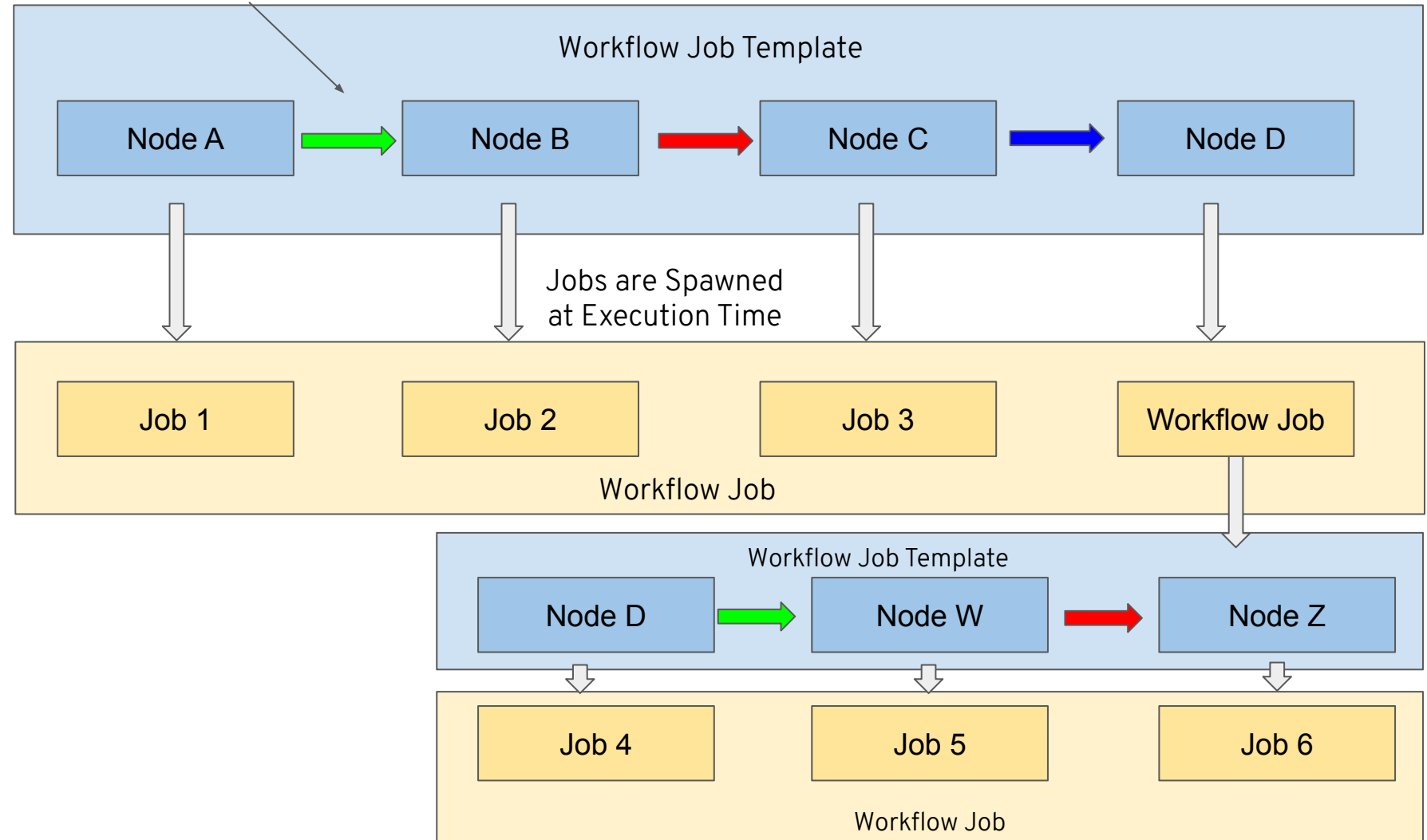
Links together a sequence of disparate resources that accomplishes the task of tracking the full set of jobs that were part of the release process as a single unit. These resources may include:

- job templates
- workflow templates
- project syncs
- inventory source syncs



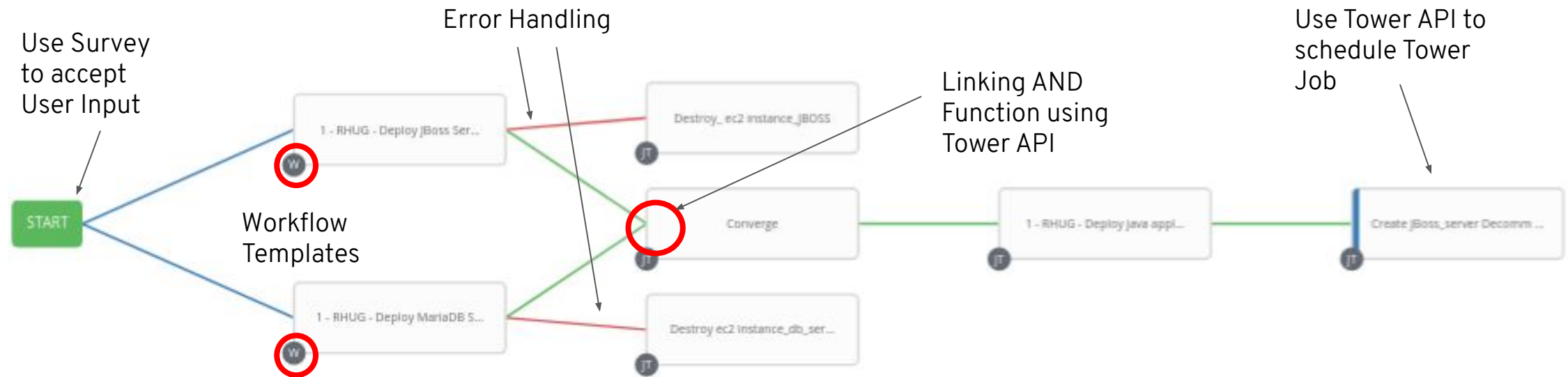
Conditional
Execution

1. Workflow tracks a series of jobs as a single execution unit
2. Workflow Job Templated consists of a series of nodes
3. Nodes can be:
 - a. Templates,
 - b. Inventory Syncs
 - c. Project Syncs
 - d. Workflow Job Template

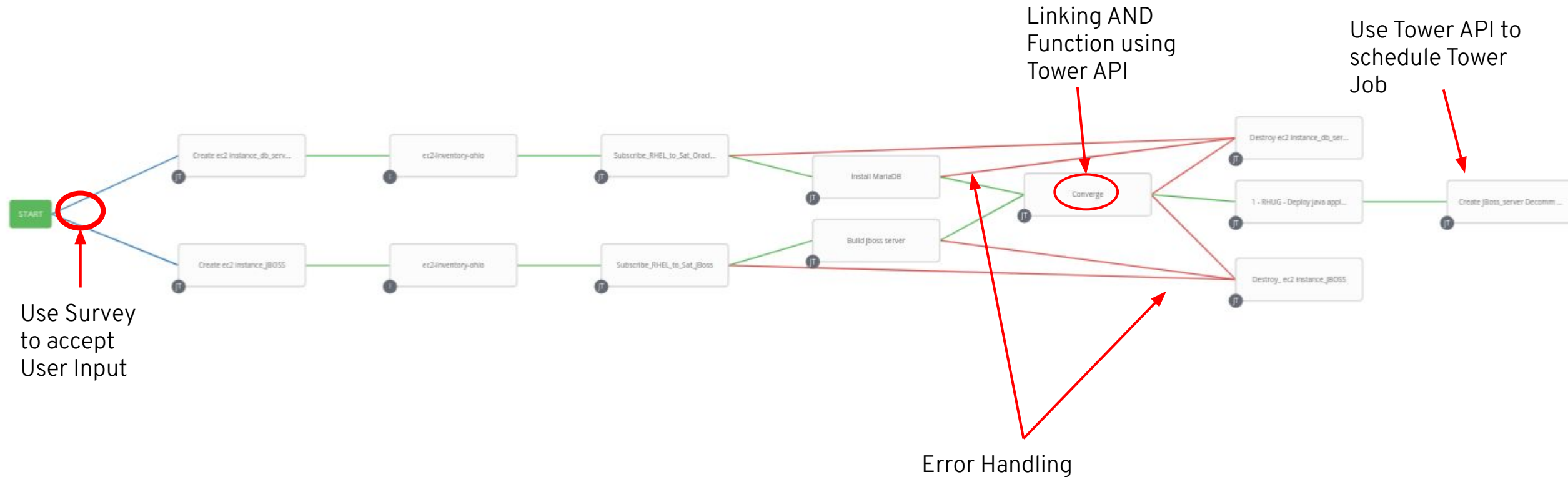


A Workflow to deploy a service

Deploy Java Application with MariaDB Service



Deploy Java Application with MariaDB Service



Surveys

Matches to variable
in playbook or in
workflow

Governance

1 - RHUG - Deploy Application Service | SURVEY

EDIT SURVEY PROMPT

* PROMPT

How Many Maria DB Servers

DESCRIPTION

* ANSWER VARIABLE NAME

db_server_count

* ANSWER TYPE

Integer

MINIMUM

1

MAXIMUM

2

DEFAULT ANSWER

1

☒ REQUIRED

CLEAR

UPDATE

PREVIEW

* HOW MANY MARIA DB SERVERS

1

* WHAT VERSION OF RHEL FOR MARIADB SERVERS

rhel8

* HOW MANY JBOSS SERVERS

1

* WHAT VERSION OF RHEL FOR JBOSS SERVERS

rhel7

* HOW MANY DAYS WOULD YOU LIKE THIS SERVICE

5

DELETE SURVEY

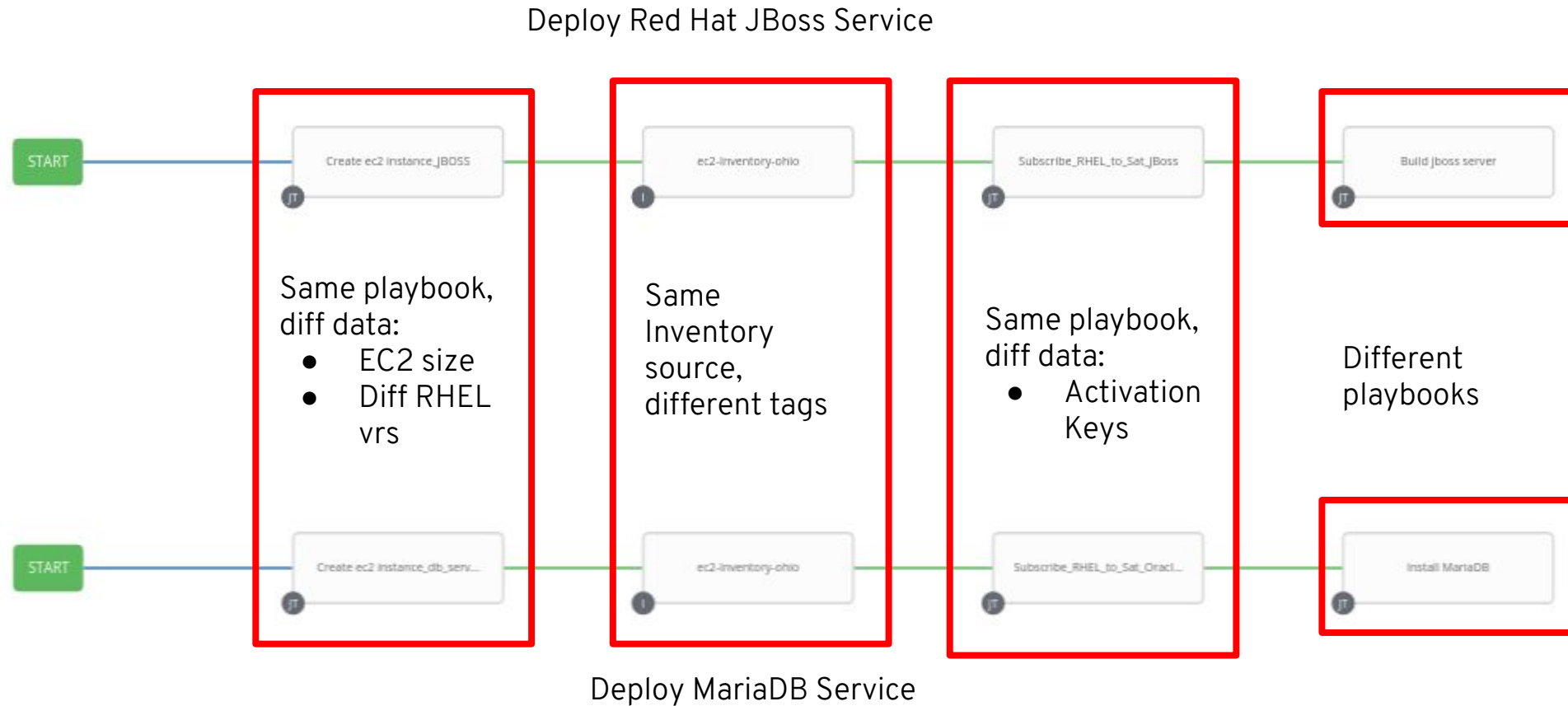
CANCEL

SAVE

16

Red Hat

Two Independent Workflows



Linking works as OR Function by Default

Use Tower API to fail this template if one of the preceding nodes failed

Converge

DETAILS PERMISSIONS NOTIFICATIONS COMPLETED JOBS SCHEDULES ADD SURVEY

* NAME

DESCRIPTION

* JOB TYPE ☐ PROMPT ON LAUNCH

* INVENTORY ☐ PROMPT ON LAUNCH

* PROJECT

* PLAYBOOK

CREDENTIALS ☐ PROMPT ON LAUNCH

FORKS

LIMIT

* VERBOSITY ☐ PROMPT ON LAUNCH

JOB TAGS ☐ PROMPT ON LAUNCH

SKIP TAGS

LABELS

INSTANCE GROUPS

JOB SLICING

TIMEOUT

SHOW CHANGES ☐ ☐ PROMPT ON LAUNCH

OPTIONS

- ☐ ENABLE PRIVILEGE ESCALATION
- ☐ ENABLE PROVISIONING CALLBACKS
- ☐ ENABLE WEBHOOK
- ☐ ENABLE CONCURRENT JOBS
- ☐ ENABLE FACT CACHE

Lookup JobID

- job_id: "{{ lookup('env', 'JOB_ID') }}"

Get Workflow node id for the job

- url: "{{ tower_base_url
}}/workflow_job_nodes/?job_id={{ job_id }}"

Get parent workflow nodes for this workflow node

- url: "{{ tower_base_url
}}/workflow_job_nodes/?success_nodes={{
result.json.results[0].id }}"

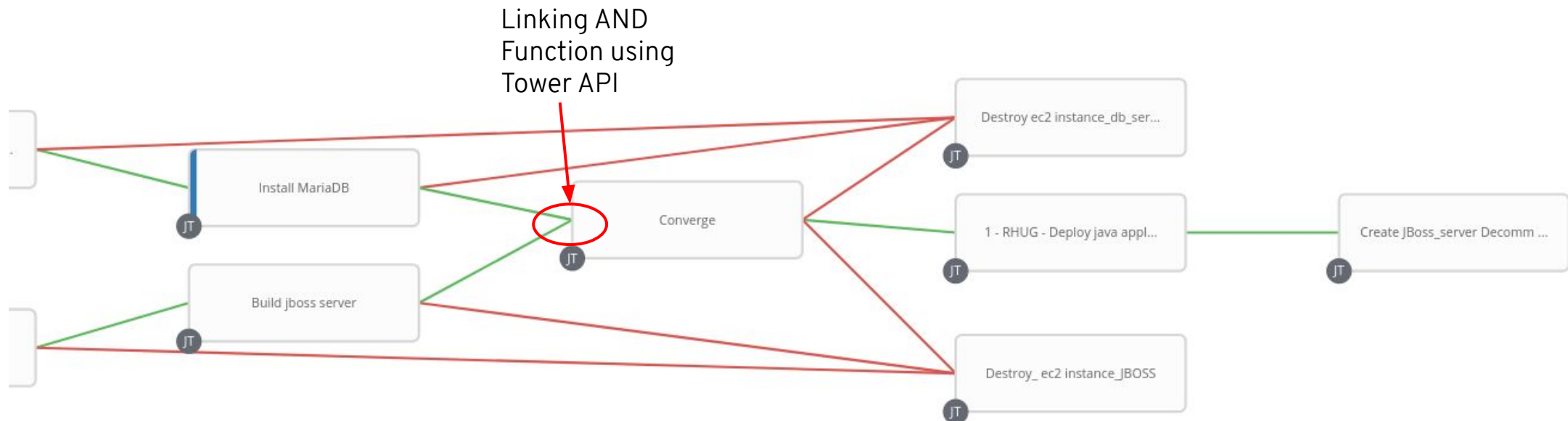
Fail this playbook if a parent node failed

- fail:
msg: "Parent workflow node {{ item }} failed"
when: "item.summary_fields.job.status == 'failed'"
loop: "{{ result.json.results }}"

Deploy Java Application with MariaDB Service

Using Tower API

CONFIDENTIAL Designator



Linking works as OR Function by Default

CONFIDENTIAL Designator

Use Tower API to fail this template if one of the preceding nodes failed

Get Job_id and
Tower_url

```
hosts: localhost
gather_facts: false
vars:
  this_playbook_should_fail: false
  job_id: "{{ lookup('env', 'JOB_ID') }}"
  tower_base_url: "https://{{ lookup('env', 'TOWER_HOST') }}/api/v2"
  tower_username: "{{ lookup('env', 'TOWER_USERNAME') }}"
  tower_password: "{{ lookup('env', 'TOWER_PASSWORD') }}"
  tower_verify_ssl: false
```

```
- name: "Get Workflow node id for this job"
  uri:
    url: "{{ tower_base_url }}/workflow_job_nodes/?job_id={{ job_id }}"
    validate_certs: "{{ tower_verify_ssl }}"
    force_basic_auth: true
    user: "{{ tower_username }}"
    password: "{{ tower_password }}"
  register: result
```

Determine WF
node id for this
job

Get a List of preceding workflow nodes for this node

```
- name: "Get parent workflow nodes for this workflow node"
  uri:
    url: "{{ tower_base_url }}/workflow_job_nodes/?success_nodes={{ result.json.results[0].id }}"
    validate_certs: "{{ tower_verify_ssl }}"
    force_basic_auth: true
  user: "{{ tower_username }}"
  password: "{{ tower_password }}"
  register: result
```

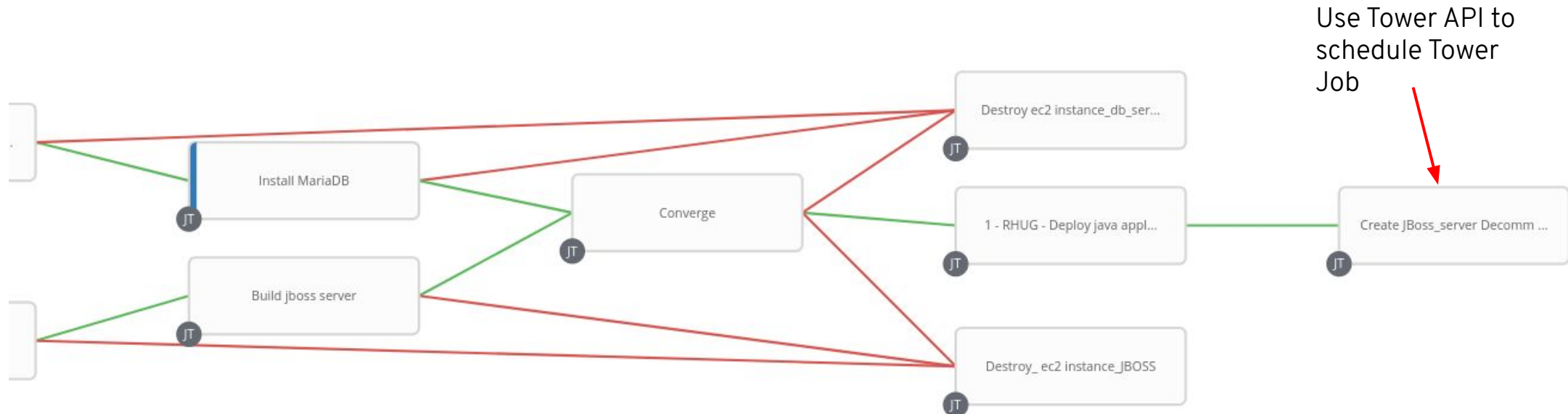
```
- name: "Fail this playbook if a parent node failed"
  fail:
    msg: "Parent workflow node {{ item }} failed"
    when: "item.summary_fields.job.status == 'failed'"
    loop: "{{ result.json.results }}"
```

Loop through the results
and fail if any parent job
failed

Deploy Java Application with MariaDB Service

Using Tower API

CONFIDENTIAL Designator



Schedule Decommission Job

Extracts from Tower API Reference Guide

POST**/api/v2/job_templates/{id}/schedules/** Create a Schedule for a Job Template

Make a POST request to this resource with the following schedule fields to create a new schedule associated with this job template.

- **rrule**: A value representing the schedules iCal recurrence rule. (string, required)

Name	Description
------	-------------

id * required

string
(path)

data
(body)

```
{
  "extra_data": "{\"var1\": \"foo\"}",
  "name": "test sch",
  "rrule": "DTSTART:20151117T050000Z RRULE:FREQ=DAILY;INTERVAL=1;COUNT=1"
}
```

Parameter content type

application/json

Calculate required variable and create variable

```
--
- name:
  hosts: all
  vars:
    decom_interval: 8

  tasks:
    - name: set fact_decom
      set_fact:
        decom_time: "{{ ( ansible_date_time.epoch | int ) + ( 86400 * decom_interval ) }}"

    - name: format time
      set_fact:
        adj_time: "{{ '%Y%m%dT%H%M%S' | strftime( decom_time ) }}"


    - name: print decom_time
      debug:
        var=adj_time

    - name: create variable
      set_fact:
        rrule_var: "DTSTART;TZID=America/New_York:{{ adj_time }} RRULE:FREQ=DAILY;INTERVAL=1;COUNT=1"
```


Use Template to Create Payload

Template to create payload file
(jboss_schedule.json.j2)

```
{  
  "name": "jboss_server_decomm",  
  "rrule": "{{ rrule_var }}"  
}
```



```
- name: create some file  
  local_action: template  
    src=jboss_schedule.json.j2  
    dest=schedule.json  
    mode=0664
```

Post to API with Payload

```
tasks:
- name: schedule decomm - DB
  uri:
    url: https://ec2-18-191-70-179.us-east-2.compute.amazonaws.com/api/v2/job_templates/26/schedules/
    method: POST
    body_format: json
    body: "{{ lookup('file','schedule.json') }}"
    return_content: yes
    validate_certs: false
    user: admin
    status_code: 201
    password: r3dh4t1!
    force_basic_auth: yes
  register: schedules
```

Demo Time

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



youtube.com/user/RedHatVideos



facebook.com/redhatinc



twitter.com/RedHat