Ansible Linux Automation Workshop

Introduction to Ansible for Red Hat Enterprise Linux Automation for System Administrators and Operators
What you will learn

- Intro to Ansible Automation Platform
- How it Works
- Understanding modules, tasks, playbooks
- How to execute Ansible commands
- Using variables and templates
- Automation Controller - where it fits in
- Automation Controller basics
- Major Automation Controller features - RBAC, workflows
Introduction

Topics Covered:

- Why the Ansible Automation Platform?
- What can it do?
Automation happens when one person meets a problem they never want to solve again.
Many organizations share the same challenge

Too many unintegrated, domain-specific tools

Network ops  SecOps  Devs/DevOps  IT ops
Why the Ansible Automation Platform?

Powerful
Orchestrate complex processes at enterprise scale.

Simple
Simplify automation creation and management across multiple domains.

Agentless
Easily integrate with hybrid environments.
Automate the deployment and management of automation

Your entire IT footprint

Do this...

- Orchestrates
- Manage configurations
- Deploy applications
- Provision / deprovision
- Deliver continuously
- Secure and comply

On these...

- Firewalls
- Load balancers
- Applications
- Containers
- Virtualization platforms
- Servers
- Clouds
- Storage
- Network devices
- And more...
Break down silos

Different teams a single platform

Consistent governance

Cloud

Edge

Datacenter

Line of business

Devs/DevOps  IT ops  SecOps  Network ops
What makes a platform?

**Red Hat Ansible Automation Platform**

- **Content creators**
- **Operators**
- **Domain experts**
- **Users**

<table>
<thead>
<tr>
<th>On-premises</th>
<th>Ansible Cloud Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automation controller</td>
<td>Automation hub</td>
</tr>
<tr>
<td>Automation services catalog</td>
<td>Insights for Ansible Automation Platform</td>
</tr>
</tbody>
</table>

**Ansible content domains**

- **Infrastructure**
  - Linux
  - Windows
  - Cloud
  - Network
  - Security

**Ansible command line**

*Fueled by an open source community*
Red Hat named a Leader in The Forrester Wave™

Infrastructure Automation Platforms, Q3 2020

Received highest possible score in the criteria of:

- Deployment functionality
- Product Vision
- Partner Ecosystem
- Supporting products and services
- Community support
- Planned product enhancements

▸ “Ansible continues to grow quickly, particularly among enterprises that are automating networks. The solution excels at providing a variety of deployment options and acting as a service broker to a wide array of other automation tools.”

▸ “Red Hat’s solution is a good fit for customers that want a holistic automation platform that integrates with a wide array of other vendors’ infrastructure.”

Source:

DISCLAIMER: The Forrester Wave™ is copyrighted by Forrester Research, Inc. Forrester and Forrester Wave™ are trademarks of Forrester Research, Inc. The Forrester Wave™ is a graphical representation of Forrester’s call on a market and is plotted using a detailed spreadsheet with exposed scores, weightings, and comments. Forrester does not endorse any vendor, product, or service depicted in the Forrester Wave™.
Section 1
The Ansible Basics
Exercise 1.1

Topics Covered:

- Understanding the Ansible Infrastructure
- Check the prerequisites
Create

The automation lifecycle

Content creators

Domain experts

Build → Ansible content experience

Discover → Red Hat cloud / on-premises

Automation hub

Trust → Ansible content domains

Infrastructure

Linux

Windows

Cloud

Network

Security
---
- name: install and start apache
  hosts: web
  become: yes

  tasks:
  - name: httpd package is present
    yum:
      name: httpd
      state: latest

  - name: latest index.html file is present
    template:
      src: files/index.html
      dest: /var/www/html/

  - name: httpd is started
    service:
      name: httpd
      state: started
What makes up an Ansible playbook?

- Plays
- Modules
- Plugins
Ansible plays

What am I automating?

What are they?
Top level specification for a group of tasks. Will tell that play which hosts it will execute on and control behavior such as fact gathering or privilege level.

Building blocks for playbooks
Multiple plays can exist within an Ansible playbook that execute on different hosts.

```yaml
- name: install and start apache
  hosts: web
  become: yes
```
Ansible modules
The “tools in the toolkit”

What are they?
Parametrized components with internal logic, representing a single step to be done. The modules “do” things in Ansible.

Language
Usually Python, or Powershell for Windows setups. But can be of any language.
Ansible plugins

The “extra bits”

What are they?

Plugins are pieces of code that augment Ansible’s core functionality. Ansible uses a plugin architecture to enable a rich, flexible, and expandable feature set.

Example become plugin:

```yaml
- name: install and start apache
  hosts: web
  become: yes
```

Example filter plugins:

```yaml
{{ some_variable | to_nice_json }}
{{ some_variable | to_nice_yaml }}
```
Ansible Inventory
The systems that a playbook runs against

What are they?
List of systems in your infrastructure that automation is executed against

```
[web]
webserver1.example.com
webserver2.example.com

[db]
dbserver1.example.com

[switches]
leaf01.internal.com
leaf02.internal.com
```
Ansible roles

Reusable automation actions

What are they?

Group your tasks and variables of your automation in a reusable structure. Write roles once, and share them with others who have similar challenges in front of them.

```yaml
---
- name: install and start apache
  hosts: web
  roles:
    - common
    - webservers
```
Collections
Simplified and consistent content delivery

What are they?
Collections are a data structure containing automation content:
- Modules
- Playbooks
- Roles
- Plugins
- Docs
- Tests
```yaml
---
- name: Install NGINX
  hosts: all
  tasks:
    - name: Install NGINX
      include_role:
        name: nginxinc.nginx
      vars:
        nginx_type: plus

    - name: Install NGINX App Protect
      include_role:
        name: nginxinc.nginx_app_protect
      vars:
        nginx_app_protect_setup_license: false
        nginx_app_protect_remove_license: false
        nginx_app_protect_install_signatures: false
```
Why the Red Hat Ansible Automation Platform?

90+
certified platforms
How Ansible Automation Works

Module code is executed locally on the control node.

Module code is copied to the managed node, executed, then removed.

Network Devices / API Endpoints

Local Execution

Remote Execution

Linux / Windows Hosts
Exercise 1.1

- Follow the steps in to access environment
- Use the IP provided to you, the script only has example IP
- Which editor do you use on command line?
  If you don’t know, we have a short intro
Lab Time
Complete exercise 1.1 now in your lab environment
Exercise 1.2

Topics Covered:

- Ansible inventories
- Accessing Ansible docs
- Modules and getting help
Inventory

- Ansible works against multiple systems in an inventory
- Inventory is usually file based
- Can have multiple groups
- Can have variables for each group or even host
Ansible Inventory

The Basics

An example of a static Ansible inventory including systems with IP addresses as well as fully qualified domain name (FQDN)

[myservers]
10.42.0.2
10.42.0.6
10.42.0.7
10.42.0.8
10.42.0.100
host.example.com
[app1srv]
appserver01 ansible_host=10.42.0.2
appserver02 ansible_host=10.42.0.3

[web]
node-[1:30]

[web:vars]
apache_listen_port=8080
apache_root_path=/var/www/mywebdocs/

[all:vars]
ansible_user=kev
ansible_ssh_private_key_file=/home/kev/.ssh/id_rsa
[app1srv]
appserver01 ansible_host=10.42.0.2
appserver02 ansible_host=10.42.0.3

[web]
node-[1:30]

[web:vars]
apache_listen_port=8080
apache_root_path=/var/www/mywebdocs/

[all:vars]
ansible_user=ender
ansible_ssh_private_key_file=/home/ender/.ssh/id_rsa
Accessing the Ansible docs

With the use of the latest command utility ansible-navigator, one can trigger access to all the modules available to them as well as details on specific modules.

A formal introduction to ansible-navigator and how it can be used to run playbooks in the following exercise.
Accessing the Ansible docs

Aside from listing a full list of all the modules, you can use ansible-navigator to provide details about a specific module.

In this example, we are getting information about the user module.

```bash
$ ansible-navigator doc user -m stdout

> ANSIBLE.BUILTIN.USER
(/usr/lib/python3.8/site-packages/ansible/modules/user.py)

Manage user accounts and user attributes. For Windows targets, use the [ansible.windows.win_user] module instead.
```
Lab Time

Complete exercise 1.2 now in your lab environment
Exercise 1.3

Topics Covered:

- Playbooks basics
- Running a playbook
- name: install and start apache
  hosts: web
  become: yes

tasks:
- name: httpd package is present
  yum:
    name: httpd
    state: latest

- name: latest index.html file is present
  template:
    src: files/index.html
    dest: /var/www/html/

- name: httpd is started
  service:
    name: httpd
    state: started
- name: install and start apache
  hosts: web
  become: yes

  tasks:
  - name: httpd package is present
    yum:
      name: httpd
      state: latest

  - name: latest index.html file is present
    template:
      src: files/index.html
      dest: /var/www/html/

  - name: httpd is started
    service:
      name: httpd
      state: started
---
- name: install and start apache
  hosts: web
  become: yes
  tasks:
   - name: httpd package is present
     yum:
       name: httpd
       state: latest
   - name: latest index.html file is present
     template:
       src: files/index.html
       dest: /var/www/html/
   - name: httpd is started
     service:
       name: httpd
       state: started
Running Playbooks
The most important colors of Ansible

A task executed as expected, no change was made.

A task executed as expected, making a change

A task failed to execute successfully
Running an Ansible Playbook

Using the latest ansible-navigator command

What is ansible-navigator?

ansible-navigator command line utility and text-based user interface (TUI) for running and developing Ansible automation content.

It replaces the previous command used to run playbooks “ansible-playbook”.

$ ansible-navigator run playbook.yml
How do I use ansible-navigator?

As previously mentioned, it replaces the ansible-playbook command. As such it brings two methods of running playbooks:

- Direct command-line interface
- Text-based User Interface (TUI)

# Direct command-line interface method
$ ansible-navigator run playbook.yml -m stdout

# Text-based User Interface method
$ ansible-navigator run playbook.yml
## ansible-navigator

### Mapping to previous Ansible commands

<table>
<thead>
<tr>
<th>ansible command</th>
<th>ansible-navigator command</th>
</tr>
</thead>
<tbody>
<tr>
<td>ansible-config</td>
<td>ansible-navigator config</td>
</tr>
<tr>
<td>ansible-doc</td>
<td>ansible-navigator doc</td>
</tr>
<tr>
<td>ansible-inventory</td>
<td>ansible-navigator inventory</td>
</tr>
<tr>
<td>ansible-playbook</td>
<td>ansible-navigator run</td>
</tr>
</tbody>
</table>
## ansible-navigator

### Common subcommands

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>CLI Example</th>
<th>Colon command within TUI</th>
</tr>
</thead>
<tbody>
<tr>
<td>collections</td>
<td>Explore available collections</td>
<td>ansible-navigator collections --help</td>
<td>:collections</td>
</tr>
<tr>
<td>config</td>
<td>Explore the current ansible configuration</td>
<td>ansible-navigator config --help</td>
<td>:config</td>
</tr>
<tr>
<td>doc</td>
<td>Review documentation for a module or plugin</td>
<td>ansible-navigator doc --help</td>
<td>:doc</td>
</tr>
<tr>
<td>images</td>
<td>Explore execution environment images</td>
<td>ansible-navigator images --help</td>
<td>:images</td>
</tr>
<tr>
<td>inventory</td>
<td>Explore and inventory</td>
<td>ansible-navigator inventory --help</td>
<td>:inventory</td>
</tr>
<tr>
<td>replay</td>
<td>Explore a previous run using a playbook artifact</td>
<td>ansible-navigator replay --help</td>
<td>:replay</td>
</tr>
<tr>
<td>run</td>
<td>Run a playbook</td>
<td>ansible-navigator run --help</td>
<td>:run</td>
</tr>
<tr>
<td>welcome</td>
<td>Start at the welcome page</td>
<td>ansible-navigator welcome --help</td>
<td>:welcome</td>
</tr>
</tbody>
</table>
Lab Time

Complete exercise 1.3 now in your lab environment
Exercise 1.4

Topics Covered:

- Working with variables
- What are facts?
---
- name: variable playbook test
  hosts: localhost

  vars:
  var_one: awesome
  var_two: ansible is
  var_three: "{{ var_two }} {{ var_one }}"

  tasks:
  - name: print out var_three
    debug:
      msg: "{{ var_three }}"
---
- name: variable playbook test
  hosts: localhost

  vars:
  var_one: awesome
  var_two: ansible is
  var_three: "{{ var_two }} {{ var_one }}"

  tasks:
  - name: print out var_three
    debug:
      msg: "{{ var_three }}"

ansible is awesome
Ansible Facts

- Just like variables, really...
- ... but: coming from the host itself!
- Check them out with the setup module

```yaml
tasks:
- name: Collect all facts of host
  setup:
    gather_subset:
    - 'all'
```
---
- name: facts playbook
  hosts: localhost

  tasks:
  - name: Collect all facts of host
    setup:
      gather_subset:
      - 'all'

$ ansible-navigator run playbook.yml
## Ansible Navigator TUI

### Red Hat Ansible Platform technical deck: Create

#### PLAY NAME

<table>
<thead>
<tr>
<th>PLAY NAME</th>
<th>OK</th>
<th>CHANGED</th>
<th>UNREACHABLE</th>
<th>FAILED</th>
<th>SKIPPED</th>
<th>IGNORED</th>
<th>IN PROGRESS</th>
<th>TASK COUNT</th>
<th>PROGRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>facts playbook</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>COMPLETE</td>
</tr>
</tbody>
</table>

#### RESULT

<table>
<thead>
<tr>
<th>RESULT</th>
<th>HOST</th>
<th>NUMBER</th>
<th>CHANGED</th>
<th>TASK</th>
<th>TASK ACTION</th>
<th>DURATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 OK</td>
<td>localhost</td>
<td>0</td>
<td>False</td>
<td>Gathering Facts</td>
<td>gather_facts</td>
<td>1s</td>
</tr>
<tr>
<td>1 OK</td>
<td>localhost</td>
<td>1</td>
<td>False</td>
<td>Collect all facts of host</td>
<td>setup</td>
<td>1s</td>
</tr>
</tbody>
</table>

#### PLAY [facts playbook:1]

```
PLAY [facts playbook:1]
*******************************************************************************************************************
TASK [Collect all facts of host]
***************************************************************************************************************
OK: [localhost].

ansible_facts:
  ansible_all_ipv4_addresses:
    - 10.0.2.100
  ansible_all_ipv6_addresses:
    - fe80::1caa:f0ff:fe15:23c4
```
Lab Time
Complete exercise 1.4 now in your lab environment
Exercise 1.5

Topics Covered:

● Conditionals
● Handlers
● Loops
Conditionals via VARS

Example of using a variable labeled my_mood and using it as a conditional on a particular task.

```yaml
vars:
  my_mood: happy

tasks:
  - name: task, based on my_mood var
debuge:
  msg: "Yay! I am {{ my_mood }}!"
when: my_mood == "happy"
```
---
- name: variable playbook test
  hosts: localhost

  vars:
    my_mood: happy

  tasks:
    - name: task, based on my_mood var
      debug:
        msg: "Yay! I am {{ my_mood }}!"
      when: my_mood == "happy"

    - name: task, based on my_mood var
      debug:
        msg: "Ask at your own risk. I’m {{ my_mood }}!"
      when: my_mood == "grumpy"

Alternatively
---
- name: variable playbook test
  hosts: localhost

  tasks:
  - name: Install apache
    apt:
      name: apache2
      state: latest
    when: ansible_distribution == 'Debian' or ansible_distribution == 'Ubuntu'

  - name: Install httpd
    yum:
      name: httpd
      state: latest
    when: ansible_distribution == 'RedHat'
---
- name: variable playbook test
  hosts: localhost

  tasks:
  - name: Ensure httpd package is present
    yum:
      name: httpd
      state: latest
      register: http_results

  - name: Restart httpd
    service:
      name: httpd
      state: restart
      when: httpd_results.changed
---

- name: variable playbook test
  hosts: localhost

  tasks:
  - name: Ensure httpd package is present
    yum:
      name: httpd
      state: latest
    notify: restart_httpd

  handlers:
  - name: restart_httpd
    service:
      name: httpd
      state: restart
tasks:
- name: Ensure httpd package is present
  yum:
    name: httpd
    state: latest
    notify: restart httpd

- name: Standardized index.html file
  copy:
    content: "This is my index.html file for {{ ansible_host }}"
    dest: /var/www/html/index.html
    notify: restart httpd

If either task notifies a **changed** result, the handler will be notified ONCE.

**handler runs once**
tasks:
- name: Ensure httpd package is present
  yum:
    name: httpd
    state: latest
  notify: restart_httpd

- name: Standardized index.html file
  copy:
    content: "This is my index.html file for {{ ansible_host }}"
    dest: /var/www/html/index.html
  notify: restart_httpd

If both of these tasks notifies of a changed result, the handler will be notified ONCE.
tasks:
- name: Ensure httpd package is present
  yum:
    name: httpd
    state: latest
  notify: restart httpd

- name: Standardized index.html file
  copy:
    content: "This is my index.html file for {{ ansible_host }}"
    dest: /var/www/html/index.html
  notify: restart httpd

TASK [Ensure httpd package is present] ******************************************************
ok: [web2] unchanged
ok: [web1] unchanged

TASK [Standardized index.html file] ******************************************************
ok: [web2] unchanged
ok: [web1] unchanged

PLAY RECAP **********************************************************************************
web2 : ok=2 changed=0 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0
web1 : ok=2 changed=0 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

If neither task notifies a changed result, the handler does not run.
Ansible Variables & Loops

---
- name: Ensure users
  hosts: node1
  become: yes

  tasks:
  - name: Ensure user is present
    user:
      name: dev_user
      state: present

  - name: Ensure user is present
    user:
      name: qa_user
      state: present

  - name: Ensure user is present
    user:
      name: prod_user
      state: present
---
- name: Ensure users
  hosts: node1
  become: yes
  tasks:
  - name: Ensure user is present
    user:
      name: "{{item}}"
      state: present
    loop:
      - dev_user
      - qa_user
      - prod_user
Lab Time

Complete exercise 1.5 now in your lab environment
Exercise 1.6

Topics Covered:

• Templates
- name: Ensure apache is installed and started
  hosts: web
  become: yes
  vars:
    http_port: 80
    http_docroot: /var/www/mysite.com

  tasks:
    - name: Verify correct config file is present
      template:
        src: templates/httpd.conf.j2
        dest: /etc/httpd/conf/httpd.conf
- name: Ensure apache is installed and started
  hosts: web
  become: yes
  vars:
    http_port: 80
    http_docroot: /var/www/mysite.com

  tasks:
  - name: Verify correct config file is present
    template:
      src: templates/httpd.conf.j2
      dest: /etc/httpd/conf/httpd.conf

      ## Excerpt from httpd.conf.j2
      # Change this to Listen on specific IP addresses as shown below to
      # prevent Apache from glomming onto all bound IP addresses.
      #
      # Listen 80    ## original line
      Listen {{ http_port }}

      # DocumentRoot: The directory out of which you will serve your
      # documents.
      # DocumentRoot "/var/www/html"
      DocumentRoot {{ http_docroot }}
Lab Time

Complete exercise 1.6 now in your lab environment
Exercise 1.7

Topics Covered:

- What are roles?
- How they look like
- Galaxy
Role Structure

- **Defaults**: default variables with lowest precedence (e.g. port)
- **Handlers**: contains all handlers
- **Meta**: role metadata including dependencies to other roles
- **Tasks**: plays or tasks
  Tip: It’s common to include tasks in main.yml with “when” (e.g. OS == xyz)
- **Templates**: templates to deploy
- **Tests**: place for playbook tests
- **Vars**: variables (e.g. override port)
Ansible Galaxy

Sharing
Content

Community

Roles, and
more
Lab Time
Complete exercise 1.7 now in your lab environment
Exercise 1.8

Topics Covered:

- A bonus lab - try it on your own, and when time permits
Lab Time

Complete exercise 1.8 now in your lab environment
Section 2
Automation Controller
Exercise 2.1

Topics Covered:

- Introduction to Automation Controller
What makes a platform?

**Red Hat Ansible Automation Platform**

- **Content creators**
- **Operators**
- **Domain experts**
- **Users**

<table>
<thead>
<tr>
<th>On-premises</th>
<th>Ansible Cloud Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automation controller</td>
<td>Automation hub</td>
</tr>
<tr>
<td>Automation services catalog</td>
<td>Insights for Ansible Automation Platform</td>
</tr>
</tbody>
</table>

**Ansible content domains**

- **Infrastructure**
  - Linux
  - Windows
- **Cloud**
- **Network**
- **Security**

**Ansible command line**

Fueled by an open source community
**Automation controller**

- **Push button**
  An intuitive user interface experience makes it easy for novice users to execute playbooks you allow them access to.

- **RESTful API**
  With an API first mentality every feature and function of controller can be API driven. Allow seamless integration with other tools like ServiceNow and Infoblox.

- **RBAC**
  Allow restricting playbook access to authorized users. One team can use playbooks in check mode (read-only) while others have full administrative abilities.

- **Enterprise integrations**

- **Centralized logging**
  All automation activity is securely logged. Who ran it, how they customized it, what it did, where it happened - all securely stored and viewable later, or exported through Automation controllers API.

- **Workflows**
  Automation controller’s multi-playbook workflows chain any number of playbooks, regardless of whether they use different inventories, run as different users, run at once or utilize different credentials.
What is Ansible Automation Controller?

Ansible Automation Controller is a UI and RESTful API allowing you to scale IT automation, manage complex deployments and speed productivity.

- Role-based access control
- Deploy entire applications with push-button deployment access
- All automations are centrally logged
- Powerful workflows match your IT processes
Anatomy of an Automation Job

Playbook

Git / Subversion

Project
Anatomy of an Automation Job

- Playbook
- Git / Subversion
- Project
- Credential

Brands: CyberArk, HashiCorp Vault, Red Hat
Anatomy of an Automation Job

- **Playbook**
- **Git / Subversion**
- **Project**
- **Credential**
- **Inventory**
Anatomy of an Automation Job

- Playbook
- Git / Subversion
- Credential
- Inventory
- Automation controller

Tools:
- CyberArk
- HashiCorp Vault
- DevOps42
- netbox
- Infoblox
- GitHub

Controller:
- Red Hat
Lab Time

Complete exercise 2.1 now in your lab environment
Exercise 2.2

Topics Covered:

- Inventories
- Credentials
Inventory

Inventory is a collection of hosts (nodes) with associated data and groupings that Automation Controller can connect to and manage.

- Hosts (nodes)
- Groups
- Inventory-specific data (variables)
- Static or dynamic sources
Credentials

Credentials are utilized by Automation Controller for authentication with various external resources:

- Connecting to remote machines to run jobs
- Syncing with inventory sources
- Importing project content from version control systems
- Connecting to and managing network devices

Centralized management of various credentials allows end users to leverage a secret without ever exposing that secret to them.
Lab Time
Complete exercise 2.2 now in your lab environment
Exercise 2.3

Topics Covered:

- Projects
- Job Templates
A project is a logical collection of Ansible Playbooks, represented in Ansible Automation Controller.

You can manage Ansible Playbooks and playbook directories by placing them in a source code management system supported by Automation controller, including Git and Subversion.
Job Templates

Everything in Automation controller revolves around the concept of a **Job Template**. Job Templates allow Ansible Playbooks to be controlled, delegated and scaled for an organization.

Job templates also encourage the reuse of Ansible Playbook content and collaboration between teams.

A **Job Template** requires:

- An **Inventory** to run the job against
- A **Credential** to login to devices.
- A **Project** which contains Ansible Playbooks
Expanding on Job Templates

Job Templates can be found and created by clicking the **Templates** button under the **Resources** section on the left menu.
Executing an existing Job Template

Job Templates can be launched by clicking the **rocketship button** for the corresponding Job Template.
Creating a new Job Template (1/2)

New Job Templates can be created by clicking the Add button.
Creating a new Job Template (2/2)

This New Job Template window is where the inventory, project and credential are assigned. The red asterisk * means the field is required.
Lab Time
Complete exercise 2.3 now in your lab environment
Exercise 2.4

Topics Covered:

- Surveys
Controller surveys allow you to configure how a job runs via a series of questions, making it simple to customize your jobs in a user-friendly way.

An Ansible Controller survey is a simple question-and-answer form that allows users to customize their job runs. Combine that with Controller’s role-based access control, and you can build simple, easy self-service for your users.
Creating a Survey (1/2)

Once a Job Template is saved, the Survey menu will have an **Add Button**

Click the button to open the Add Survey window.
The Add Survey window allows the Job Template to prompt users for one or more questions. The answers provided become variables for use in the Ansible Playbook.
Using a Survey

When launching a job, the user will now be prompted with the Survey. The user can be required to fill out the Survey before the Job Template will execute.
Lab Time
Complete exercise 2.4 now in your lab environment
Exercise 2.5

Topics Covered:

- Role based access control
Role-based access control

How to manage access

- Role-based access control system:
  Users can be grouped in teams, and roles can be assigned to the teams.

- Rights to edit or use can be assigned across all objects.

- All backed by enterprise authentication if needed.
User Management

- An **organization** is a logical collection of users, teams, projects, inventories and more. All entities belong to an organization.

- A **user** is an account to access Ansible Automation Controller and its services given the permissions granted to it.

- **Teams** provide a means to implement role-based access control schemes and delegate responsibilities across organizations.
Lab Time
Complete exercise 2.5 now in your lab environment
Exercise 2.6

Topics Covered:

● Workflows
Workflows

Combine automation to create something bigger

- Workflows enable the creation of powerful holistic automation, chaining together multiple pieces of automation and events.
- Simple logic inside these workflows can trigger automation depending on the success or failure of previous steps.
Adding a New Template

- To add a new **Workflow** click on the **Add** button.
  This time select the **Add workflow template**
Creating the Workflow

- Fill out the required parameters and click **Save**. As soon as the Workflow Template is saved the Workflow Visualizer will open.
Workflow Visualizer

- The Workflow Visualizer will start as a blank canvas.
- Click the green Start button to start building the workflow.
Visualizing a Workflow
Workflows can branch out, or converge in.

Blue indicates this Job Template will always run

Green indicates this Job Template will only be run if the previous Job Template is successful

Red indicates this Job Template will only be run if the previous Job Template fails
Lab Time

Complete exercise 2.6 now in your lab environment
Exercise 2.7

Topics Covered:

● Wrap-up
Lab Time
Complete exercise 2.7 now in your lab environment
Where to go next

Learn more
- Workshops
- Documents
- Youtube
- Twitter

Get started
- Evals
- cloud.redhat.com

Get serious
- Red Hat Automation Adoption Journey
- Red Hat Training
- Red Hat Consulting
Thank you