Overview

How to use this deck

Name:
Windows Automation workshop

Purpose:
These additional slides are used in conjunction with the windows automation workshop as provisioned from:
https://github.com/ansible/workshops

Last updated:
Jan 19, 2022

What this deck is for?

What this deck is not for?

Google Slides source link (Red Hat internal):
https://docs.google.com/presentation/d/1RO5CQiCoqLDSEINvT1_1fQrR-oWMINuW-uB0JRvtJzE/edit#slide=id.g10efc4a0549_0_2429

Owner:
Ansible MBU, ansible-pmm-tmm@redhat.com

List of all official Ansible content:
Red Hat Ansible Automation Platform One Stop:
https://redhat.highspot.com/items/5966647572ad8e20778bc270?ifr m=srp.10
Ansible Windows automation workshop

Introduction to automating Microsoft Windows with Ansible Automation Platform 2
What you will learn

- Introduction to Ansible automation
- How Ansible works for Windows automation
- Understanding Ansible modules and playbooks
- Using Ansible controller to scale automation to the enterprise
- Reusing automation with Ansible Roles
Anyone can automate... but an enterprise needs to coordinate and scale
Many organizations share the same challenge

Too many unintegrated, domain-specific tools
Break down silos
Different teams a single platform

Consistent governance
Why the Red Hat® Ansible® Automation Platform?
Why the Red Hat Ansible Automation Platform?

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...

Orchestrate  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 ...
100+
certified platforms
What makes a platform?
What makes a platform?

Combining the universal automation language with cloud services and certified content for automating, deploying, and operating applications, infrastructure and services securely at enterprise scale.

---

**Ansible automation**

Providing scalable, secure implementation for describing, building, and managing the deployment of enterprise IT applications across diverse enterprise architectures.

**Cloud services**

Cloud services that facilitate team collaboration and provide operational analytics for automating heterogeneous, hybrid environments.

**Certified content**

Extends native platform capabilities with certified, supported content designed to expand the automation domain and accelerate adoption for enterprise customers.
Holistic automation for your enterprise

What makes a platform?

Create ➔ Operate ➔ Consume
What makes a platform?

Red Hat
Ansible Automation Platform

Content creators
Operators
Domain experts
Users

Fueled by an open source community
Create
Create

The automation lifecycle

Build

Ansible content experience

Red Hat cloud / on-premises

Automation hub

Discover

Ansible content domains

Infrastructure

Linux

Windows

Cloud

Network

Security

Trust

Content creators

Domain experts
---
- name: start IIS/stop firewall
  hosts: windows-web
  become: yes

  tasks:
  - name: IIS is running
    win_service:
      name: W3Svc
      state: running

  - name: firewall service is stopped/disabled
    win_service:
      name: MpsSvc
      state: stopped
      start_mode: disabled
What makes up an Ansible playbook?
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.

---
- name: start IIS
  hosts: windows-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
Powershell for Windows, python for linux. Can be of any language.

- name: ITS is running
  win_service:
    name: W3Svc
    state: running
**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: start IIS
  hosts: windows-web
  become: yes
```

Example filter plugins:

```yaml
{{ some_variable | to_nice_json }}
{{ some_variable | to_nice_yaml }}
```
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 IIS
  hosts: windows-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
deploy-nginx.yml

---
- name: Install NGINX Plus
  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
Automation Controller
Operate

The automation lifecycle

Operators

Control

On-premises

Automation hub

Ansible Cloud Services

Insights for Ansible Automation Platform

Manage

Automation controller

Deploy

Ansible command line
A playbook run
Where it all starts

- A playbook is interpreted and run against one or multiple hosts - task by task. The order of the tasks defines the execution.
- In each task, the module does the actual work.
Anatomy of Automation Operation

- Workflows
- API & Webhooks
- Credentials
- Centralized Logging & Audit Trail
- Automation controller
Execution of content

Running at the core

- The central execution of automation content is managed and done either via central cluster.
- Can also sync git repositories, takes care of execution environments, collections, credentials, inventory and logging.
- Full audit trail of the execution, including what version of content was executed, what variable values were provided, etc.
Inventories and credentials

How to talk to others

- An inventory is a collection of hosts (nodes) with associated data and groupings that the automation platform can connect to and manage:
  - Nodes
  - Groups
  - Can be static or dynamic
  - Smart inventories possible

- And what usernames and passwords do you use during connection? That is kept in the credentials.
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.
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.
API

Integration of automation into larger workflows

- The API provides programmatic access to the automation via a defined interface.
- Underneath it is still powered by the same bits and pieces which are at the core: workflows, inventories, etc.
- It offers simple integration into other tools like ITSM, SOAR, etc.
Lab Time

Exercise 1 - Configure Automation Controller

This lab is all about exploring the environment and configuring Automation Controller to import project code from source control

Approximate time: 15 mins
Ad-hoc command execution
An ad-hoc command is a single Ansible task to perform quickly, but don’t want to save for later.
Ad-hoc Commands: Common Options

- **-m MODULE_NAME, --module-name=MODULE_NAME**
  Module name to execute the ad-hoc command

- **-a MODULE_ARGS, --args=MODULE_ARGS**
  Module arguments for the ad-hoc command

- **-b, --become**
  Run ad-hoc command with elevated rights such as sudo, the default method

- **-e EXTRA_VARS, --extra-vars=EXTRA_VARS**
  Set additional variables as key=value or YAML/JSON

- **--version**
  Display the version of Ansible

- **--help**
  Display the MAN page for the Ansible tool
# Ad-hoc Commands

# check all my inventory hosts are ready to be managed by Ansible
$ ansible all -m win_ping

# collect and display the discovered facts for the localhost
$ ansible localhost -m setup

# run the uptime command on all hosts in the web group
$ ansible web -m command -a "uptime"
Ad-hoc Commands from Automation Controller

![Ad-hoc Commands from Automation Controller](image-url)
Lab Time
Exercise 2 - Ad-hoc commands

This lab guides you through executing ad-hoc commands from Automation Controller

 средства

Approximate time: 15 mins
Playbooks
Variables

Ansible can work with metadata from various sources and manage their context in the form of variables.

- Command line parameters
- Plays and tasks
- Files
- Inventory
- Discovered facts
- Roles
Facts are bits of information derived from examining a host systems that are stored as variables for later use in a play.

```bash
$ ansible localhost -m setup
localhost | success >> {
  "ansible_facts": {
    "ansible_default_ipv4": {
      "address": "192.168.1.37",
      "alias": "wlan0",
      "gateway": "192.168.1.1",
      "interface": "wlan0",
      "macaddress": "c4:85:08:3b:a9:16",
      "mtu": 1500,
      "netmask": "255.255.255.0",
      "network": "192.168.1.0",
      "type": "ether"
    }
  }
}
```
Variable Precedence

The order in which the same variable from different sources will override each other.

1. command line values (eg “-u user”)
2. role defaults [1]
3. inventory file or script group vars [2]
4. inventory group_vars/all [3]
5. playbook group_vars/all [3]
6. inventory group_vars/* [3]
7. playbook group_vars/* [3]
8. inventory file or script host vars [2]
9. inventory host_vars/* [3]
10. playbook host_vars/* [3]
11. host facts / cached set_facts [4]
12. play vars
13. play vars_prompt
14. play vars_files
15. role vars (defined in role/vars/main.yml)
16. block vars (only for tasks in block)
17. task vars (only for the task)
18. include vars
19. set_facts / registered vars
20. role (and include_role) params
21. include params
22. extra vars (always win precedence)
Tasks

Tasks are the application of a module to perform a specific unit of work.

- **win_file**: A directory should exist
- **win_package**: A package should be installed
- **win_service**: A service should be running
- **win_template**: Render a configuration file from a template
- **win_get_url**: Fetch an archive file from a URL
- **win_copy**: Copy a file from your repository or a remote source
tasks:
- name: Ensure IIS Server is present
  win_feature:
    name: Web-Server
    state: present

- name: Ensure latest index.html file is present
  win_copy:
    src: files/index.html
    dest: c:www

- name: Restart IIS
  win_service:
    name: IIS Admin Service
    state: restarted
Handlers are special tasks that run at the end of a play if notified by another task when a change occurs.

*If a package gets installed or updated, notify a service restart task that it needs to run.*
Handler Tasks

tasks:
- name: Ensure IIS Server is present
  win_feature:
    name: Web-Server
    state: present
  notify: Restart IIS

- name: Ensure latest index.html file is present
  win_copy:
    src: files/index.html
    dest: c:\www\

handlers:
- name: Restart IIS
  win_service:
    name: IIS Admin Service
    state: restarted
Plays and playbooks

Plays are ordered sets of tasks to execute against host selections from your inventory. A playbook is a file containing one or more plays.
Plays and playbooks

---

- name: Ensure IIS is installed and started
  hosts: web
  become: yes
  vars:
    service_name: IIS Admin Service
  tasks:
  - name: Ensure IIS Server is present
    win_feature:
      name: Web-Server
      state: present
  - name: Ensure latest index.html file is present
    win_copy:
      src: files/index.html
      dest: c:\www\
  - name: Ensure IIS is started
    win_service:
      name: "{{ service_name }}"
      state: started
Meaningful names

---

- name: Ensure IIS is installed and started
  hosts: web
  become: yes
  vars:
    service_name: IIS Admin Service

  tasks:
  - name: Ensure IIS Server is present
    win_feature:
      name: Web-Server
      state: present

  - name: Ensure latest index.html file is present
    win_copy:
      src: files/index.html
      dest: c:\www\

  - name: Ensure IIS is started
    win_service:
      name: "{{ service_name }}"
      state: started
- **name**: Ensure IIS is installed and started
  ```yaml
  hosts: web
  become: yes
  vars:
    service_name: IIS Admin Service
  ```

  **tasks:**
  - **name**: Ensure IIS Server is present
    ```yaml
    win_feature:
      name: Web-Server
      state: present
    ```
  - **name**: Ensure latest index.html file is present
    ```yaml
    win_copy:
      src: files/index.html
      dest: c:\www\
    ```
  - **name**: Ensure IIS is started
    ```yaml
    win_service:
      name: "{{ service_name }}"
      state: started
    ```
Privilege escalation

---

- **name**: Ensure IIS is installed and started
  
  **hosts**: web
  
  **become**: yes

  **vars**:
  
  - **service_name**: IIS Admin Service

- **tasks**:
  
  - **name**: Ensure IIS Server is present
    
    **win_feature**:
    
    - **name**: Web-Server
    
    - **state**: present

  - **name**: Ensure latest index.html file is present
    
    **win_copy**:
    
    - **src**: files/index.html
    
    - **dest**: c:\www\

  - **name**: Ensure IIS is started
    
    **win_service**:
    
    - **name**: "{{ service_name }}"
    
    - **state**: started
- name: Ensure IIS is installed and started
  hosts: web
  become: yes
  vars:
    service_name: IIS Admin Service

tasks:
- name: Ensure IIS Server is present
  win_feature:
    name: Web-Server
    state: present

- name: Ensure latest index.html file is present
  win_copy:
    src: files/index.html
    dest: c:\www\

- name: Ensure IIS is started
  win_service:
    name: "{{ service_name }}"
    state: started
Tasks

---
- name: Ensure IIS is installed and started
  hosts: web
  become: yes
  vars:
    service_name: IIS Admin Service

tasks:
- name: Ensure IIS Server is present
  win_feature:
    name: Web-Server
    state: present

- name: Ensure latest index.html file is present
  win_copy:
    src: files/index.html
    dest: c:\www\

- name: Ensure IIS is started
  win_service:
    name: "{{ service_name }}"
    state: started
Lab Time

Exercise 3 - Intro to playbooks

In this lab you’ll author your first playbook

Exercise 4 - Configure a job template

This lab guides you through creating a job template from an existing project

Approximate time: 25 mins
Advanced playbooks
Doing more with playbooks

Here are some more essential playbook features that you can apply:

- Templates
- Loops
- Conditionals
- Tags
- Blocks
Ansible embeds the **Jinja2 templating engine** that can be used to dynamically:

- Set and modify play variables
- Conditional logic
- Generate files such as configurations from variables
Loops can do one task on multiple things, such as create a lot of users, install a lot of packages, or repeat a polling step until a certain result is reached.

- name: Ensure IIS Server is present
  win_feature:
    name: "{{ item }}"
    state: present
  loop:
    - Web-Server
    - NET-Framework-Core
Ansible supports the conditional execution of a task based on the run-time evaluation of variable, fact, or previous task result.

- name: Ensure IIS Server is present
  win_feature:
    name: Web-Server
    state: present
  when: ansible_os_family == "Windows"
Doing more with playbooks: **Tags**

Tags are useful to be able to run a subset of a playbook on-demand.

- name: Ensure IIS Server is present
  win_feature:
    name: "{{ item }}"
    state: present
  with_items:
    - Web-Server
    - NET-Framework-Core
  tags:
    - packages

- name: Copy web.config template to Server
  win_template:
    src: templates/web.config.j2
    dest: C:\inetpub\wwwroot\web.config
  tags:
    - configuration
Doing more with playbooks: Blocks

Blocks cut down on repetitive task directives, allow for logical grouping of tasks and even in play error handling.

- block:
  - name: Ensure IIS Server is present
    win_feature:
      name: "{{ item }}"
      state: present
    with_items:
      - Web-Server

- name: Copy web.config template to Server
  win_template:
    src: templates/web.config.j2
    dest: C:\inetpub\wwwroot\web.config

when: ansible_os_family == "Windows"
Lab Time
Exercise 5 - More advanced playbook

This lab expands on the existing playbook

Approximate time: 15 mins
Sharing automation
Roles are a packages of closely related Ansible content that can be shared more easily than plays alone.

- Improves readability and maintainability of complex plays
- Eases sharing, reuse and standardization of automation processes
- Enables Ansible content to exist independently of playbooks, projects -- even organizations
- Provides functional conveniences such as file path resolution and default values
## Project with Embedded Roles Example

<table>
<thead>
<tr>
<th>site.yml</th>
</tr>
</thead>
<tbody>
<tr>
<td>roles/</td>
</tr>
<tr>
<td>common/</td>
</tr>
<tr>
<td>files/</td>
</tr>
<tr>
<td>templates/</td>
</tr>
<tr>
<td>tasks/</td>
</tr>
<tr>
<td>handlers/</td>
</tr>
<tr>
<td>vars/</td>
</tr>
<tr>
<td>defaults/</td>
</tr>
<tr>
<td>meta/</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>iis/</th>
</tr>
</thead>
<tbody>
<tr>
<td>files/</td>
</tr>
<tr>
<td>templates/</td>
</tr>
<tr>
<td>tasks/</td>
</tr>
<tr>
<td>handlers/</td>
</tr>
<tr>
<td>vars/</td>
</tr>
<tr>
<td>defaults/</td>
</tr>
<tr>
<td>meta/</td>
</tr>
</tbody>
</table>
Roles

Project with Embedded Roles Example

```yaml
# site.yml
---
- name: Execute common and iis role
  hosts: web
  roles:
    - common
    - iis
```
Roles

http://galaxy.ansible.com

Ansible Galaxy is a hub for finding, reusing and sharing Ansible content.

Jump-start your automation project with content contributed and reviewed by the Ansible community.
Lab Time
Exercise 6 - Ansible roles

In this lab you will convert your existing automation into roles that can be reused as a part of larger automated workflows

 вли Approximate time: 15 mins
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

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