



Red Hat Training and Certification

DO374 - Instructor Demo Guide

Travis Michette

Version 1.0

Table of Contents

- 1. Developing Playbooks with Ansible Automation Platform 2 1
 - 1.1. Introducing Red Hat Ansible Automation Platform 2 1
 - 1.1.1. Orientation to Red Hat Ansible Automation Platform 2 1
 - 1.1.2. Red Hat Ansible Automation Platform 2 Components 1
 - 1.1.2.1. Ansible Core 1
 - 1.1.2.2. Ansible Content Collections 1
 - 1.1.2.3. Ansible Content Navigator 2
 - 1.1.2.4. Ansible Execution Environments 2
 - 1.1.2.5. Automation Controller 2
 - 1.1.2.6. Ansible Automation Hub 2
 - 1.1.2.7. Hosted Services 2
 - 1.1.3. Red Hat Ansible Automation Platform 2 Architecture 2
 - 1.1.3.1. Developing Playbooks with Ansible Automation Platform 2 2
 - 1.2. Running Playbooks with Automation Content Navigator 2
 - 1.2.1. Introducing Automation Content Navigator 2
 - 1.2.1.1. Improving Portability with Automation Execution Environments 2
 - 1.2.2. Installing Automation Content Navigator 2
 - 1.2.3. Configuring Authentication to Managed Hosts 2
 - 1.2.3.1. Preparing SSH Key-Based Authentication 2
 - 1.2.3.2. Providing Private Keys to the Automation Execution Environment 2
 - 1.2.4. Running Automation Content Navigator 2
 - 1.2.4.1. Running Playbooks 2
 - 1.2.4.2. Reviewing Previous Playbook Runs 2
 - 1.2.4.3. Reading Documentation 2
 - 1.2.4.4. Getting Help 2
 - 1.3. Managing Ansible Project Materials Using Git 2
 - 1.3.1. Defining Infrastructure as Code 3
 - 1.3.2. Introducing Git 3
 - 1.3.3. Describing Initial Git Configuration 3
 - 1.3.4. Starting the Git Workflow 3
 - 1.3.4.1. Examining the Git Log 3
 - 1.3.5. Working with Branches and References 3
 - 1.3.5.1. Creating Branches 3
 - 1.3.5.2. Merging Branches 3
 - 1.3.5.3. Creating Branches from Old Commits 3
 - 1.3.5.4. Pushing Branches to Remote Repositories 3
 - 1.3.6. Structuring Ansible Projects in Git 3
 - 1.3.6.1. Roles and Ansible Content Collections 3
 - 1.3.6.2. Configuring Git to Ignore Files 3
 - 1.4. Implementing Recommended Ansible Practices 3
 - 1.4.1. The Effectiveness of Ansible 3
 - 1.4.2. Keeping Things Simple 3
 - 1.4.2.1. Keeping Your Playbooks Readable 3

1.4.2.2. Use Existing Modules	3
1.4.2.3. Adhering to a Standard Style	3
1.4.3. Staying Organized	3
1.4.3.1. Following Conventions for Naming Variables.	3
1.4.3.2. Standardizing the Project Structure	3
1.4.3.3. Using Dynamic Inventories.	3
1.4.3.4. Taking Advantage of Groups	4
1.4.3.5. Using Roles and Ansible Content Collections for Reusable Content	4
1.4.3.6. Running Playbooks Centrally	4
1.4.3.7. Building Automation Execution Environments	4
1.4.4. Testing Often	4
1.4.4.1. Testing the Results of Tasks	4
1.4.4.2. Using Block/Rescue to Recover or Rollback	4
1.4.4.3. Developing Playbooks with the Latest Ansible Version	4
1.4.4.4. Using Test Tools	4
2. Managing Content Collections and Execution Environments	5
2.1. Reusing Content from Ansible Content Collections	5
2.1.1. Defining Ansible Content Collections	5
2.1.1.1. Organizing Ansible Content Collections in Namespaces	5
2.1.2. Using Ansible Content Collections	5
2.1.2.1. Accessing Ansible Content Collection Documentation.	5
2.1.2.2. Using Ansible Content Collections in Playbooks	5
2.1.2.3. Finding Ansible Content Collections.	5
2.1.2.4. Using the Built-in Ansible Content Collection	5
2.2. Finding and Installing Ansible Content Collections	5
2.2.1. Sources for Ansible Content Collections	5
2.2.1.1. Finding Collections on Ansible Automation Hub	5
2.2.2. Installing Ansible Content Collections	5
2.2.2.1. Installing Collections from the Command Line	5
2.2.2.2. Installing Collections with a Requirements File	5
2.2.2.3. Listing Installed Collections	5
2.2.3. Configuring Collection Sources	5
2.2.3.1. Installing Collections from Ansible Automation Hub	5
2.2.3.2. Installing Collections from Private Automation Hub	5
2.3. Selecting an Execution Environment	5
2.3.1. Describing Automation Execution Environments	6
2.3.2. Selecting a Supported Automation Execution Environment.	6
2.3.3. Inspecting Automation Execution Environments	6
2.3.4. Using Automation Execution Environments with Ansible Content Navigator.	6
3. Running Playbooks with Automation Controller.	7
3.1. Explaining the Automation Controller Architecture	7
3.1.1. Introduction to Automation Controller.	7
3.1.2. Describing the Architecture of Automation Controller	7
3.1.3. Automation Controller Features	7
3.2. Running Playbooks in Automation Controller	7
3.2.1. Exploring Resources in Automation Controller	7

3.2.2. Creating Credential Resources	7
3.2.2.1. Listing Credentials	7
3.2.2.2. Creating a Machine Credential	7
3.2.2.3. Creating a Source Control Credential	7
3.2.3. Creating Project Resources	7
3.2.4. Creating Inventory Resources	7
3.2.4.1. Manually Creating Groups and Hosts	7
3.2.4.2. Populating Groups and Hosts Using a Project Inventory File	7
3.2.5. Creating Job Template Resources.	7
3.2.6. Launching and Reviewing Jobs	7
4. Working with Ansible Configuration Settings	8
4.1. Examining Ansible Configuration with Automation Content Navigator	8
4.1.1. Inspecting Configuration in Interactive Mode	8
4.1.1.1. Searching for Specific Configuration Parameters	8
4.1.1.2. Accessing Parameter Details	8
4.1.1.3. Inspecting Local Configuration	8
4.1.2. Inspecting Ansible Configuration in Standard Output Mode	8
4.2. Configuring Automation Content Navigator	8
4.2.1. Format of the Settings File	8
4.2.2. Locating the Settings File	8
4.2.2.1. Selecting a Settings File to Use	8
4.2.3. Editing the Settings File	8
4.2.3.1. Setting a Default Automation Execution Environment	8
4.2.3.2. Default to Running in Standard Output Mode	8
4.2.3.3. Disabling Playbook Artifacts.	8
4.2.3.4. Overview of an Example Settings File	8
5. Managing Inventories.	9
5.1. Managing Dynamic Inventories	9
5.1.1. Generating Inventories Dynamically	9
5.1.2. Discussing Inventory Plug-ins	9
5.1.2.1. Using Inventory Plug-ins	9
5.1.3. Developing Inventory Scripts	9
5.1.3.1. Using Inventory Scripts	9
5.1.4. Managing Multiple Inventories	9
5.2. Writing YAML Inventory Files.	9
5.2.1. Discussing Inventory Plug-ins	9
5.2.2. Writing YAML Static Inventory Files.	9
5.2.2.1. Setting Inventory Variables	9
5.2.3. Converting a Static Inventory File in INI Format to YAML	9
5.2.4. Troubleshooting YAML Files	9
5.2.4.1. Protecting a Colon Followed by a Space	9
5.2.4.2. Protecting a Variable that Starts a Value	9
5.2.4.3. Knowing the Difference Between a String and a Boolean or Float	9
5.3. Managing Inventory Variables	9
5.3.1. Describing the Basic Principles of Variables	9
5.3.2. Variable Merging and Precedence.	9

5.3.2.1. Determining Command-line Option Precedence	10
5.3.2.2. Determining Role Default Precedence	10
5.3.2.3. Determining Host and Group Variable Precedence	10
5.3.2.4. Determining Play Variable Precedence	10
5.3.2.5. Determining the Precedence of Extra Variables	10
5.3.3. Separating Variables from Inventory	10
5.3.4. Using Special Inventory Variables	10
5.3.4.1. Configuring Human Readable Inventory Host Names	10
5.3.5. Identifying the Current Host Using Variables	10
6. Managing Task Execution	11
6.1. Controlling Privilege Escalation	11
6.1.1. Privilege Escalation Strategies.	11
6.1.1.1. Privilege Escalation by Configuration.	11
6.1.1.2. Defining Privilege Escalation in Plays	11
6.1.1.3. Privilege Escalation in Tasks	11
6.1.1.4. Grouping Privilege Escalation Tasks with Blocks.	11
6.1.1.5. Applying Privilege Escalation in Roles	11
6.1.1.6. Listing Privilege Escalation with Connection Variables	11
6.2. Choosing Privilege Escalation Approaches	11
6.3. Controlling Task Execution	11
6.3.1. Controlling the Order of Execution	11
6.3.1.1. Importing or Including Roles as a Task	11
6.3.1.2. Defining Pre- and Post-tasks	11
6.3.1.3. Reviewing the Order of Execution	11
6.3.2. Listening to Handlers	11
6.3.2.1. Notifying Handlers	11
6.3.3. Controlling the Order of Host Execution.	11
6.4. Running Selected Tasks	11
6.4.1. Tagging Ansible Resources	11
6.4.2. Managing Tagged Resources	12
6.4.2.1. Running Tasks with Specific Tags	12
6.4.2.2. Combining Tags to Run Multiple Tasks	12
6.4.2.3. Skipping Tasks with Specific Tags	12
6.4.2.4. Listing Tags in a Playbook	12
6.4.3. Assigning Special Tags	12
6.5. Optimizing Execution for Speed.	12
6.5.1. Optimizing Playbook Execution	12
6.5.1.1. Optimizing the Infrastructure	12
6.5.1.2. Disabling Fact Gathering	12
6.5.1.3. Reusing Gathered Facts with Fact Caching.	12
6.5.1.4. Limiting Fact Gathering	12
6.5.1.5. Increasing Parallelism	12
6.5.1.6. Avoiding Loops with the Package Manager Modules.	12
6.5.1.7. Efficiently Copying Files to Managed Hosts.	12
6.5.1.8. Using Templates	12
6.5.1.9. Enabling Pipelining.	12

6.5.2. Profiling Playbook Execution with Callback Plug-ins	12
6.5.2.1. Timing Tasks and Roles	12
7. Transforming Data with Filters and Plug-ins	13
7.1. Processing Variables Using Filters	13
7.1.1. Ansible Filters	13
7.1.2. Variable Types	13
7.1.3. Manipulating Lists	13
7.1.3.1. Extracting list elements	13
7.1.3.2. Modifying the Order of List Elements	13
7.1.3.3. Merging Lists	13
7.1.3.4. Operating on Lists as Sets	13
7.1.4. Manipulating Dictionaries	13
7.1.4.1. Joining dictionaries	13
7.1.4.2. Converting Dictionaries	13
7.1.5. Hashing, Encoding, and Manipulating Strings	13
7.1.5.1. Hashing strings and passwords	13
7.1.5.2. Encoding strings	13
7.1.5.3. Formatting Text	13
7.1.5.4. Replacing Text	13
7.1.6. Manipulating JSON Data	13
7.1.6.1. JSON Queries	13
7.1.6.2. Parsing and Encoding Data Structures	13
7.2. Templating External Data using Lookups	13
7.2.1. Lookup Plug-ins	14
7.2.2. Calling Lookup Plug-ins	14
7.2.3. Selecting Lookup Plug-ins	14
7.2.3.1. Reading the Contents of Files	14
7.2.3.2. Applying Data with a Template	14
7.2.3.3. Reading Command Output in the Execution Environment	14
7.2.3.4. Getting Content from a URL	14
7.2.3.5. Getting Information from the Kubernetes API	14
7.2.3.6. Using Custom Lookup Plug-ins	14
7.2.4. Handling Lookup Errors	14
7.3. Implementing Advanced Loops	14
7.3.1. Comparing Loops and Lookup Plug-ins	14
7.3.2. Example Iteration Scenarios	14
7.3.2.1. Iterating over a List of Lists	14
7.3.2.2. Iterating Over Nested Lists	14
7.3.2.3. Iterating Over a Dictionary	14
7.3.2.4. Iterating Over a File Globbing Pattern	14
7.3.2.5. Retrying a Task	14
7.4. Using Filters to Work with Network Addresses	14
7.4.1. Gathering and Processing Networking Information	14
7.4.2. Network Information Filters	14
7.4.2.1. Testing IP Addresses	15
7.4.2.2. Filtering Data	15

7.4.2.3. Manipulating IP Addresses.	15
7.4.2.4. Reformatting or Calculating Network Information.	15
8. Coordinating Rolling Updates	16
8.1. Delegating Tasks and Facts	16
8.1.1. Delegating Tasks	16
8.1.1.1. Delegating to localhost	16
8.1.2. Delegating Facts	16
8.2. Configuring Parallelism	16
8.2.1. Configure Parallelism in Ansible Using Forks	16
8.2.2. Running Batches of Hosts Through the Entire Play.	16
8.3. Managing Rolling Updates.	16
8.3.1. Overview	16
8.3.2. Controlling Batch Size	16
8.3.2.1. Setting a Fixed Batch Size	16
8.3.2.2. Setting Batch Size as a Percentage.	16
8.3.2.3. Setting Batch Sizes to Change During the Play.	16
8.3.3. Aborting the Play	16
8.3.3.1. Specifying Failure Tolerance	16
8.3.4. Running a Task Once	16
9. Creating Content Collections and Execution Environments	17
9.1. Writing Ansible Content Collections.	17
9.1.1. Developing Ansible Content Collections	17
9.1.1.1. Selecting a Namespace for Collections	17
9.1.1.2. Creating Collection Skeletons	17
9.1.1.3. Adding Content to Collections	17
9.1.1.4. Updating Collection Metadata	17
9.1.1.5. Declaring Collection Dependencies	17
9.1.1.6. Building Collections	17
9.1.1.7. Validating and Testing Collections.	17
9.1.2. Publishing Collections	17
9.2. Building a Custom Execution Environment	17
9.2.1. Deciding When to Create a Custom Automation Execution Environment	17
9.2.2. Preparing for a New Automation Execution Environment	17
9.2.2.1. Declaring the Ansible Content Collections to Install.	17
9.2.2.2. Declaring Python Packages	17
9.2.2.3. Declaring RPM Packages	17
9.2.3. Building a New Automation Execution Environment	17
9.2.3.1. Interacting with the Build Process	17
9.3. Validating a Custom Execution Environment.	17
9.3.1. Testing Automation Execution Environments Locally	18
9.3.1.1. Running a Test Playbook	18
9.3.1.2. Providing Authentication Credentials	18
9.3.2. Sharing an Automation Execution Environment from Private Automation Hub	18
9.4. Using Custom Content Collections and Execution Environments in Automation Controller	18
9.4.1. Using Custom Collections with Existing Execution Environments	18
9.4.1.1. Preparing Ansible Projects for Automation Controller	18

9.4.1.2. Storing Authentication Credentials for Collections	18
9.4.2. Using Custom Automation Execution Environments with Automation Controller	18
9.4.2.1. Storing Container Registry Credentials	18
9.4.2.2. Configuring Automation Execution Environments	18
9.4.2.3. Configuring the Default Automation Execution Environment for a Project.	18
9.4.2.4. Specifying an Automation Execution Environment in a Template	18

1. Developing Playbooks with Ansible Automation Platform 2

1.1. Introducing Red Hat Ansible Automation Platform 2

Describing the architecture of Red Hat Ansible Automation Platform 2 (AAP2) and new features for Ansible development.

1.1.1. Orientation to Red Hat Ansible Automation Platform 2

New evolution of Ansible Platform providing customization with Ansible Execution Environments (EEs), Ansible Navigator, and a redesign of Ansible Tower which has now become Ansible Controller. Ansible Automation Platform now also provides Ansible Automation Hub which is a private Ansible Galaxy as well as a container registry service for Ansible EEs.

1.1.2. Red Hat Ansible Automation Platform 2 Components

1.1.2.1. Ansible Core

The Ansible Core package is provided by **ansible-core** and is version Ansible Core 2.11 in AAP2.0. This package provides the **ansible** command as well as the built-in modules allowing administrators to run playbooks with the **ansible-playbook** command. The **ansible-core** package only contains a minimal set of modules (**ansible.builtin**) collection and all other modules have been moved to Ansible collections.



*The **ansible** Package*

It is still possible to install the package called **ansible**. This will install Ansible 2.9 which is AAP1.2. This version of Ansible will support collections, but is not the full AAP2.0 version of Ansible.

1.1.2.2. Ansible Content Collections

Ansible content and modules have now been re-organized into what is referred to as Ansible Content Collections (**Content Collections**) in order to support the growth and rapid development of modules and packages. This separation allows modules, roles, plug-in to be separated from the **Ansible Core** for a simpler management style.

This separation provides the following

- Developers can easily upgrade and deploy new version of their modules without depending on Ansible
- Only needed modules can be present on the Ansible system or in the execution environment
- New modules and content doesn't need to wait for a new version of Ansible to be deployed



ansible.builtin

The **ansible.builtin** collection is a special collection that will always be part of Ansible Core. However, this has a limited number of modules. Things like the **Firealld** module have now been moved as part of the **POSIX** Ansible Collection.



Collection Mapping

Ansible mapping of content collections: https://github.com/ansible/ansible/blob/devel/lib/ansible/config/ansible_builtint_runtime.yml

1.1.2.3. Ansible Content Navigator

1.1.2.4. Ansible Execution Environments

1.1.2.5. Automation Controller

1.1.2.6. Ansible Automation Hub

1.1.2.7. Hosted Services

1.1.3. Red Hat Ansible Automation Platform 2 Architecture

1.1.3.1. Developing Playbooks with Ansible Automation Platform 2

1.2. Running Playbooks with Automation Content Navigator

Section Info Here

1.2.1. Introducing Automation Content Navigator

1.2.1.1. Improving Portability with Automation Execution Environments

1.2.2. Installing Automation Content Navigator

1.2.3. Configuring Authentication to Managed Hosts

1.2.3.1. Preparing SSH Key-Based Authentication

1.2.3.2. Providing Private Keys to the Automation Execution Environment

1.2.4. Running Automation Content Navigator

1.2.4.1. Running Playbooks

1.2.4.2. Reviewing Previous Playbook Runs

1.2.4.3. Reading Documentation

1.2.4.4. Getting Help

1.3. Managing Ansible Project Materials Using Git

Section Info Here

1.3.1. Defining Infrastructure as Code

1.3.2. Introducing Git

1.3.3. Describing Initial Git Configuration

1.3.4. Starting the Git Workflow

1.3.4.1. Examining the Git Log

1.3.5. Working with Branches and References

1.3.5.1. Creating Branches

1.3.5.2. Merging Branches

1.3.5.3. Creating Branches from Old Commits

1.3.5.4. Pushing Branches to Remote Repositories

1.3.6. Structuring Ansible Projects in Git

1.3.6.1. Roles and Ansible Content Collections

1.3.6.2. Configuring Git to Ignore Files

1.4. Implementing Recommended Ansible Practices

Section Info Here

1.4.1. The Effectiveness of Ansible

1.4.2. Keeping Things Simple

1.4.2.1. Keeping Your Playbooks Readable

1.4.2.2. Use Existing Modules

1.4.2.3. Adhering to a Standard Style

1.4.3. Staying Organized

1.4.3.1. Following Conventions for Naming Variables

1.4.3.2. Standardizing the Project Structure

1.4.3.3. Using Dynamic Inventories

1.4.3.4. Taking Advantage of Groups

1.4.3.5. Using Roles and Ansible Content Collections for Reusable Content

1.4.3.6. Running Playbooks Centrally

1.4.3.7. Building Automation Execution Environments

1.4.4. Testing Often

1.4.4.1. Testing the Results of Tasks

1.4.4.2. Using Block/Rescue to Recover or Rollback

1.4.4.3. Developing Playbooks with the Latest Ansible Version

1.4.4.4. Using Test Tools

2. Managing Content Collections and Execution Environments

2.1. Reusing Content from Ansible Content Collections

Section Info Here

2.1.1. Defining Ansible Content Collections

2.1.1.1. Organizing Ansible Content Collections in Namespaces

2.1.2. Using Ansible Content Collections

2.1.2.1. Accessing Ansible Content Collection Documentation

2.1.2.2. Using Ansible Content Collections in Playbooks

2.1.2.3. Finding Ansible Content Collections

2.1.2.4. Using the Built-in Ansible Content Collection

2.2. Finding and Installing Ansible Content Collections

Section Info Here

2.2.1. Sources for Ansible Content Collections

2.2.1.1. Finding Collections on Ansible Automation Hub

2.2.2. Installing Ansible Content Collections

2.2.2.1. Installing Collections from the Command Line

2.2.2.2. Installing Collections with a Requirements File

2.2.2.3. Listing Installed Collections

2.2.3. Configuring Collection Sources

2.2.3.1. Installing Collections from Ansible Automation Hub

2.2.3.2. Installing Collections from Private Automation Hub

2.3. Selecting an Execution Environment

Section Info Here

2.3.1. Describing Automation Execution Environments

2.3.2. Selecting a Supported Automation Execution Environment

2.3.3. Inspecting Automation Execution Environments

2.3.4. Using Automation Execution Environments with Ansible Content Navigator

3. Running Playbooks with Automation Controller

3.1. Explaining the Automation Controller Architecture

Section Info Here

3.1.1. Introduction to Automation Controller

3.1.2. Describing the Architecture of Automation Controller

3.1.3. Automation Controller Features

3.2. Running Playbooks in Automation Controller

Section Info Here

3.2.1. Exploring Resources in Automation Controller

3.2.2. Creating Credential Resources

3.2.2.1. Listing Credentials

3.2.2.2. Creating a Machine Credential

3.2.2.3. Creating a Source Control Credential

3.2.3. Creating Project Resources

3.2.4. Creating Inventory Resources

3.2.4.1. Manually Creating Groups and Hosts

3.2.4.2. Populating Groups and Hosts Using a Project Inventory File

3.2.5. Creating Job Template Resources

3.2.6. Launching and Reviewing Jobs

4. Working with Ansible Configuration Settings

4.1. Examining Ansible Configuration with Automation Content Navigator

Section Info Here

4.1.1. Inspecting Configuration in Interactive Mode

4.1.1.1. Searching for Specific Configuration Parameters

4.1.1.2. Accessing Parameter Details

4.1.1.3. Inspecting Local Configuration

4.1.2. Inspecting Ansible Configuration in Standard Output Mode

4.2. Configuring Automation Content Navigator

Section Info Here

4.2.1. Format of the Settings File

4.2.2. Locating the Settings File

4.2.2.1. Selecting a Settings File to Use

4.2.3. Editing the Settings File

4.2.3.1. Setting a Default Automation Execution Environment

4.2.3.2. Default to Running in Standard Output Mode

4.2.3.3. Disabling Playbook Artifacts

4.2.3.4. Overview of an Example Settings File

5. Managing Inventories

5.1. Managing Dynamic Inventories

Section Info Here

5.1.1. Generating Inventories Dynamically

5.1.2. Discussing Inventory Plug-ins

5.1.2.1. Using Inventory Plug-ins

5.1.3. Developing Inventory Scripts

5.1.3.1. Using Inventory Scripts

5.1.4. Managing Multiple Inventories

5.2. Writing YAML Inventory Files

Section Info Here

5.2.1. Discussing Inventory Plug-ins

5.2.2. Writing YAML Static Inventory Files

5.2.2.1. Setting Inventory Variables

5.2.3. Converting a Static Inventory File in INI Format to YAML

5.2.4. Troubleshooting YAML Files

5.2.4.1. Protecting a Colon Followed by a Space

5.2.4.2. Protecting a Variable that Starts a Value

5.2.4.3. Knowing the Difference Between a String and a Boolean or Float

5.3. Managing Inventory Variables

Section Info Here

5.3.1. Describing the Basic Principles of Variables

5.3.2. Variable Merging and Precedence

5.3.2.1. Determining Command-line Option Precedence

5.3.2.2. Determining Role Default Precedence

5.3.2.3. Determining Host and Group Variable Precedence

5.3.2.4. Determining Play Variable Precedence

5.3.2.5. Determining the Precedence of Extra Variables

5.3.3. Separating Variables from Inventory

5.3.4. Using Special Inventory Variables

5.3.4.1. Configuring Human Readable Inventory Host Names

5.3.5. Identifying the Current Host Using Variables

6. Managing Task Execution

6.1. Controlling Privilege Escalation

Section Info Here

6.1.1. Privilege Escalation Strategies

6.1.1.1. Privilege Escalation by Configuration

6.1.1.2. Defining Privilege Escalation in Plays

6.1.1.3. Privilege Escalation in Tasks

6.1.1.4. Grouping Privilege Escalation Tasks with Blocks

6.1.1.5. Applying Privilege Escalation in Roles

6.1.1.6. Listing Privilege Escalation with Connection Variables

6.2. Choosing Privilege Escalation Approaches

6.3. Controlling Task Execution

Section Info Here

6.3.1. Controlling the Order of Execution

6.3.1.1. Importing or Including Roles as a Task

6.3.1.2. Defining Pre- and Post-tasks

6.3.1.3. Reviewing the Order of Execution

6.3.2. Listening to Handlers

6.3.2.1. Notifying Handlers

6.3.3. Controlling the Order of Host Execution

6.4. Running Selected Tasks

Section Info Here

6.4.1. Tagging Ansible Resources

6.4.2. Managing Tagged Resources

6.4.2.1. Running Tasks with Specific Tags

6.4.2.2. Combining Tags to Run Multiple Tasks

6.4.2.3. Skipping Tasks with Specific Tags

6.4.2.4. Listing Tags in a Playbook

6.4.3. Assigning Special Tags

6.5. Optimizing Execution for Speed

Section Info Here

6.5.1. Optimizing Playbook Execution

6.5.1.1. Optimizing the Infrastructure

6.5.1.2. Disabling Fact Gathering

6.5.1.3. Reusing Gathered Facts with Fact Caching

6.5.1.4. Limiting Fact Gathering

6.5.1.5. Increasing Parallelism

6.5.1.6. Avoiding Loops with the Package Manager Modules

6.5.1.7. Efficiently Copying Files to Managed Hosts

6.5.1.8. Using Templates

6.5.1.9. Enabling Pipelining

6.5.2. Profiling Playbook Execution with Callback Plug-ins

6.5.2.1. Timing Tasks and Roles

7. Transforming Data with Filters and Plug-ins

7.1. Processing Variables Using Filters

Section Info Here

7.1.1. Ansible Filters

7.1.2. Variable Types

7.1.3. Manipulating Lists

7.1.3.1. Extracting list elements

7.1.3.2. Modifying the Order of List Elements

7.1.3.3. Merging Lists

7.1.3.4. Operating on Lists as Sets

7.1.4. Manipulating Dictionaries

7.1.4.1. Joining dictionaries

7.1.4.2. Converting Dictionaries

7.1.5. Hashing, Encoding, and Manipulating Strings

7.1.5.1. Hashing strings and passwords

7.1.5.2. Encoding strings

7.1.5.3. Formatting Text

7.1.5.4. Replacing Text

7.1.6. Manipulating JSON Data

7.1.6.1. JSON Queries

7.1.6.2. Parsing and Encoding Data Structures

7.2. Templating External Data using Lookups

Section Info Here

7.2.1. Lookup Plug-ins

7.2.2. Calling Lookup Plug-ins

7.2.3. Selecting Lookup Plug-ins

7.2.3.1. Reading the Contents of Files

7.2.3.2. Applying Data with a Template

7.2.3.3. Reading Command Output in the Execution Environment

7.2.3.4. Getting Content from a URL

7.2.3.5. Getting Information from the Kubernetes API

7.2.3.6. Using Custom Lookup Plug-ins

7.2.4. Handling Lookup Errors

7.3. Implementing Advanced Loops

Section Info Here

7.3.1. Comparing Loops and Lookup Plug-ins

7.3.2. Example Iteration Scenarios

7.3.2.1. Iterating over a List of Lists

7.3.2.2. Iterating Over Nested Lists

7.3.2.3. Iterating Over a Dictionary

7.3.2.4. Iterating Over a File Globbing Pattern

7.3.2.5. Retrying a Task

7.4. Using Filters to Work with Network Addresses

Section Info Here

7.4.1. Gathering and Processing Networking Information

7.4.2. Network Information Filters

7.4.2.1. Testing IP Addresses

7.4.2.2. Filtering Data

7.4.2.3. Manipulating IP Addresses

7.4.2.4. Reformatting or Calculating Network Information

8. Coordinating Rolling Updates

8.1. Delegating Tasks and Facts

Section Info Here

8.1.1. Delegating Tasks

8.1.1.1. Delegating to localhost

8.1.2. Delegating Facts

8.2. Configuring Parallelism

Section Info Here

8.2.1. Configure Parallelism in Ansible Using Forks

8.2.2. Running Batches of Hosts Through the Entire Play

8.3. Managing Rolling Updates

Section Info Here

8.3.1. Overview

8.3.2. Controlling Batch Size

8.3.2.1. Setting a Fixed Batch Size

8.3.2.2. Setting Batch Size as a Percentage

8.3.2.3. Setting Batch Sizes to Change During the Play

8.3.3. Aborting the Play

8.3.3.1. Specifying Failure Tolerance

8.3.4. Running a Task Once

9. Creating Content Collections and Execution Environments

9.1. Writing Ansible Content Collections

Section Info Here

9.1.1. Developing Ansible Content Collections

9.1.1.1. Selecting a Namespace for Collections

9.1.1.2. Creating Collection Skeletons

9.1.1.3. Adding Content to Collections

9.1.1.4. Updating Collection Metadata

9.1.1.5. Declaring Collection Dependencies

9.1.1.6. Building Collections

9.1.1.7. Validating and Testing Collections

9.1.2. Publishing Collections

9.2. Building a Custom Execution Environment

Section Info Here

9.2.1. Deciding When to Create a Custom Automation Execution Environment

9.2.2. Preparing for a New Automation Execution Environment

9.2.2.1. Declaring the Ansible Content Collections to Install

9.2.2.2. Declaring Python Packages

9.2.2.3. Declaring RPM Packages

9.2.3. Building a New Automation Execution Environment

9.2.3.1. Interacting with the Build Process

9.3. Validating a Custom Execution Environment

Section Info Here

9.3.1. Testing Automation Execution Environments Locally

9.3.1.1. Running a Test Playbook

9.3.1.2. Providing Authentication Credentials

9.3.2. Sharing an Automation Execution Environment from Private Automation Hub

9.4. Using Custom Content Collections and Execution Environments in Automation Controller

Section Info Here

9.4.1. Using Custom Collections with Existing Execution Environments

9.4.1.1. Preparing Ansible Projects for Automation Controller

9.4.1.2. Storing Authentication Credentials for Collections

9.4.2. Using Custom Automation Execution Environments with Automation Controller

9.4.2.1. Storing Container Registry Credentials

9.4.2.2. Configuring Automation Execution Environments

9.4.2.3. Configuring the Default Automation Execution Environment for a Project

9.4.2.4. Specifying an Automation Execution Environment in a Template