# \*\*Study Material: Day 10 - Advanced Ansible Tower (AWX) Techniques\*\*

---

## \*\*Table of Contents\*\*

1. \*\*Introduction to Advanced Ansible Tower (AWX) Techniques\*\*

- What are Advanced Techniques?

- Why Use Advanced Techniques?

2. \*\*Workflow Automation\*\*

- Creating Workflow Templates

- Example: Multi-Step Deployment Workflow

3. \*\*Integration with CI/CD Pipelines\*\*

- Integrating with Jenkins

- Example: CI/CD Pipeline with Ansible Tower

4. \*\*Using the REST API\*\*

- Overview of the REST API

- Example: Automating Job Execution via API

5. \*\*Best Practices for Advanced Techniques\*\*

- Organizing Workflows

- Securing API Access

- Monitoring and Logging

6. \*\*Conclusion and Next Steps\*\*

---

## \*\*1. Introduction to Advanced Ansible Tower (AWX) Techniques\*\*

### \*\*What are Advanced Techniques?\*\*

Advanced techniques in Ansible Tower (AWX) include workflow automation, integration with CI/CD pipelines, and using the REST API for automation.

### \*\*Why Use Advanced Techniques?\*\*

- \*\*Efficiency\*\*: Automate complex workflows and reduce manual intervention.

- \*\*Integration\*\*: Seamlessly integrate with existing CI/CD pipelines.

- \*\*Scalability\*\*: Use the REST API to manage and scale automation tasks.

---

## \*\*2. Workflow Automation\*\*

### \*\*Creating Workflow Templates\*\*

Workflow templates allow you to chain multiple job templates together to create complex automation workflows.

### \*\*Example: Multi-Step Deployment Workflow\*\*

1. \*\*Create Job Templates\*\*:

- \*\*Provision Infrastructure\*\*: Job template to provision infrastructure.

- \*\*Deploy Application\*\*: Job template to deploy the application.

- \*\*Run Tests\*\*: Job template to run tests.

2. \*\*Create Workflow Template\*\*:

- \*\*Add Nodes\*\*: Add the job templates as nodes in the workflow.

- \*\*Define Dependencies\*\*: Define dependencies between nodes (e.g., run tests after deployment).

3. \*\*Run Workflow\*\*: Execute the workflow to automate the entire deployment process.

---

## \*\*3. Integration with CI/CD Pipelines\*\*

### \*\*Integrating with Jenkins\*\*

1. \*\*Install Jenkins\*\*: Set up Jenkins on your server.

2. \*\*Install Ansible Plugin\*\*: Install the Ansible plugin in Jenkins.

3. \*\*Create Jenkins Job\*\*: Create a Jenkins job to trigger Ansible Tower job templates.

### \*\*Example: CI/CD Pipeline with Ansible Tower\*\*

1. \*\*Source Code Change\*\*: A developer pushes code to the Git repository.

2. \*\*Jenkins Build\*\*: Jenkins triggers a build and runs unit tests.

3. \*\*Ansible Tower Job\*\*: Jenkins triggers an Ansible Tower job to deploy the application.

4. \*\*Run Tests\*\*: Ansible Tower runs integration tests.

5. \*\*Deploy to Production\*\*: If tests pass, deploy to production.

---

## \*\*4. Using the REST API\*\*

### \*\*Overview of the REST API\*\*

The REST API allows you to programmatically manage Ansible Tower, including creating and running jobs, managing inventories, and more.

### \*\*Example: Automating Job Execution via API\*\*

1. \*\*Get API Token\*\*:

```bash

curl -X POST -u username:password https://tower.example.com/api/v2/tokens/

```

2. \*\*Run Job Template\*\*:

```bash

curl -X POST -H "Authorization: Bearer <token>" -H "Content-Type: application/json" -d '{"extra\_vars": {"var1": "value1"}}' https://tower.example.com/api/v2/job\_templates/<id>/launch/

```

---

## \*\*5. Best Practices for Advanced Techniques\*\*

### \*\*Organizing Workflows\*\*

- \*\*Logical Grouping\*\*: Organize workflows logically (e.g., by environment, application).

- \*\*Naming Conventions\*\*: Use consistent naming conventions for workflows and job templates.

### \*\*Securing API Access\*\*

- \*\*API Tokens\*\*: Use API tokens for authentication and limit their scope.

- \*\*Role-Based Access\*\*: Restrict API access based on roles.

### \*\*Monitoring and Logging\*\*

- \*\*Monitoring\*\*: Use monitoring tools to track the performance of workflows.

- \*\*Logging\*\*: Enable logging to capture detailed information about job executions.

---

## \*\*6. Conclusion and Next Steps\*\*

### \*\*What We Learned\*\*

- \*\*Workflow Automation\*\*: Creating and managing workflow templates.

- \*\*CI/CD Integration\*\*: Integrating Ansible Tower with Jenkins for CI/CD pipelines.

- \*\*REST API\*\*: Using the REST API to automate job execution.

- \*\*Best Practices\*\*: Organizing workflows, securing API access, and monitoring.

### \*\*Next Steps\*\*

- \*\*Day 11\*\*: Using Ansible for configuration management and deployment.

- \*\*Day 12\*\*: Advanced playbook techniques, including using `set\_fact` and `register`.

---

## \*\*Images and Tables\*\*

### \*\*Example Workflow Template\*\*

![Workflow Template](https://www.example.com/workflow-template.png)

### \*\*Example CI/CD Pipeline\*\*

```yaml

- name: CI/CD Pipeline with Ansible Tower

hosts: localhost

tasks:

- name: Trigger Jenkins Build

uri:

url: "http://jenkins.example.com/job/build/buildWithParameters"

method: POST

user: "jenkins\_user"

password: "jenkins\_password"

body: "{}"

status\_code: 201

- name: Trigger Ansible Tower Job

uri:

url: "https://tower.example.com/api/v2/job\_templates/1/launch/"

method: POST

headers:

Authorization: "Bearer <token>"

body: "{}"

status\_code: 201

```

### \*\*Example REST API Call\*\*

```bash

curl -X POST -H "Authorization: Bearer <token>" -H "Content-Type: application/json" -d '{"extra\_vars": {"var1": "value1"}}' https://tower.example.com/api/v2/job\_templates/<id>/launch/

```

---

## \*\*References\*\*

- [Ansible Tower Documentation](https://docs.ansible.com/ansible-tower/latest/html/)

- [AWX GitHub Repository](https://github.com/ansible/awx)

- [Ansible Tower REST API Guide](https://docs.ansible.com/ansible-tower/latest/html/towerapi/index.html)

---

This study material provides a comprehensive guide to advanced Ansible Tower (AWX) techniques, including workflow automation, CI/CD integration, and using the REST API. It is designed to help you understand the core concepts and apply them in real-world scenarios.  
  
  
**Study Material: Day 10 - Advanced Ansible Tower (AWX) Techniques**

**Table of Contents**

1. **Introduction to Advanced Ansible Tower (AWX) Techniques**
   * What are Advanced Techniques?
   * Why Use Advanced Techniques?
2. **Workflow Automation**
   * Creating Workflow Templates
   * Example: Multi-Step Deployment Workflow
3. **Integration with CI/CD Pipelines**
   * Integrating with Jenkins
   * Example: CI/CD Pipeline with Ansible Tower
4. **Using the REST API**
   * Overview of the REST API
   * Example: Automating Job Execution via API
5. **Best Practices for Advanced Techniques**
   * Organizing Workflows
   * Securing API Access
   * Monitoring and Logging
6. **Conclusion and Next Steps**

**1. Introduction to Advanced Ansible Tower (AWX) Techniques**

**What are Advanced Techniques?**

Advanced techniques in Ansible Tower (AWX) include workflow automation, integration with CI/CD pipelines, and using the REST API for automation.

**Why Use Advanced Techniques?**

* **Efficiency**: Automate complex workflows and reduce manual intervention.
* **Integration**: Seamlessly integrate with existing CI/CD pipelines.
* **Scalability**: Use the REST API to manage and scale automation tasks.

**2. Workflow Automation**

**Creating Workflow Templates**

Workflow templates allow you to chain multiple job templates together to create complex automation workflows.

**Example: Multi-Step Deployment Workflow**

1. **Create Job Templates**:
   * **Provision Infrastructure**: Job template to provision infrastructure.
   * **Deploy Application**: Job template to deploy the application.
   * **Run Tests**: Job template to run tests.
2. **Create Workflow Template**:
   * **Add Nodes**: Add the job templates as nodes in the workflow.
   * **Define Dependencies**: Define dependencies between nodes (e.g., run tests after deployment).
3. **Run Workflow**: Execute the workflow to automate the entire deployment process.

**3. Integration with CI/CD Pipelines**

**Integrating with Jenkins**

1. **Install Jenkins**: Set up Jenkins on your server.
2. **Install Ansible Plugin**: Install the Ansible plugin in Jenkins.
3. **Create Jenkins Job**: Create a Jenkins job to trigger Ansible Tower job templates.

**Example: CI/CD Pipeline with Ansible Tower**

1. **Source Code Change**: A developer pushes code to the Git repository.
2. **Jenkins Build**: Jenkins triggers a build and runs unit tests.
3. **Ansible Tower Job**: Jenkins triggers an Ansible Tower job to deploy the application.
4. **Run Tests**: Ansible Tower runs integration tests.
5. **Deploy to Production**: If tests pass, deploy to production.

**4. Using the REST API**

**Overview of the REST API**

The REST API allows you to programmatically manage Ansible Tower, including creating and running jobs, managing inventories, and more.

**Example: Automating Job Execution via API**

1. **Get API Token**:
2. curl -X POST -u username:password https://tower.example.com/api/v2/tokens/
3. **Run Job Template**:
4. curl -X POST -H "Authorization: Bearer <token>" -H "Content-Type: application/json" -d '{"extra\_vars": {"var1": "value1"}}' https://tower.example.com/api/v2/job\_templates/<id>/launch/

**5. Best Practices for Advanced Techniques**

**Organizing Workflows**

* **Logical Grouping**: Organize workflows logically (e.g., by environment, application).
* **Naming Conventions**: Use consistent naming conventions for workflows and job templates.

**Securing API Access**

* **API Tokens**: Use API tokens for authentication and limit their scope.
* **Role-Based Access**: Restrict API access based on roles.

**Monitoring and Logging**

* **Monitoring**: Use monitoring tools to track the performance of workflows.
* **Logging**: Enable logging to capture detailed information about job executions.

**6. Conclusion and Next Steps**

**What We Learned**

* **Workflow Automation**: Creating and managing workflow templates.
* **CI/CD Integration**: Integrating Ansible Tower with Jenkins for CI/CD pipelines.
* **REST API**: Using the REST API to automate job execution.
* **Best Practices**: Organizing workflows, securing API access, and monitoring.

**Next Steps**

* **Day 11**: Using Ansible for configuration management and deployment.
* **Day 12**: Advanced playbook techniques, including using set\_fact and register.

**Images and Tables**

**Example Workflow Template**

**Example CI/CD Pipeline**

- name: CI/CD Pipeline with Ansible Tower

hosts: localhost

tasks:

- name: Trigger Jenkins Build

uri:

url: "http://jenkins.example.com/job/build/buildWithParameters"

method: POST

user: "jenkins\_user"

password: "jenkins\_password"

body: "{}"

status\_code: 201

- name: Trigger Ansible Tower Job

uri:

url: "https://tower.example.com/api/v2/job\_templates/1/launch/"

method: POST

headers:

Authorization: "Bearer <token>"

body: "{}"

status\_code: 201

**Example REST API Call**

curl -X POST -H "Authorization: Bearer <token>" -H "Content-Type: application/json" -d '{"extra\_vars": {"var1": "value1"}}' https://tower.example.com/api/v2/job\_templates/<id>/launch/

**References**

* [Ansible Tower Documentation](https://docs.ansible.com/ansible-tower/latest/html/)
* [AWX GitHub Repository](https://github.com/ansible/awx)
* [Ansible Tower REST API Guide](https://docs.ansible.com/ansible-tower/latest/html/towerapi/index.html)

This study material provides a comprehensive guide to advanced Ansible Tower (AWX) techniques, including workflow automation, CI/CD integration, and using the REST API. It is designed to help you understand the core concepts and apply them in real-world scenarios.