# NOS 125 - Final Exam Instruction Set s Rubric

Your final exam will assess your understanding and practical application of Red Hat Automation with Ansible, focusing on fundamental concepts and techniques. You will be provided with a scenario-based task aimed at evaluating your ability to create basic automation workflows using Ansible.

**Complete this using the Red Hat Lab environment**. Do NOT attempt to use Netlab for this assignment unless you are testing configurations.

## Task Description:

You are a junior system administrator tasked with automating routine server management tasks in a small IT environment. Your goal is to develop an **Ansible playbook** that streamlines the configuration and maintenance of **two Linux servers: servera and serverb**. You will design a basic playbook that accomplishes the specifications given, ensuring that a brief overview and documentation are provided. See 'Submission Guidelines' for additional details.

## Requirements:

Create a functional automation solution using Ansible, ensuring that all necessary files are in place to execute the playbook successfully. Your playbook should target **servera** and **serverb**.

1. **Package Installation and Configuration:**

   Create a task that automates the installation and configuration of **at least three common packages** (e.g., `httpd`, `MySQL`, `MariaDB-Server`). You can choose any packages, but make sure to install at least three.

   - Test the packages on `serverc` using the `dnf install packagename` command before adding them to your playbook targeting `servera` and `serverb`. (i.e., `dnf install packagename`).
   - **Tip**: Use **loops** and/or **variables** to avoid writing separate tasks for each package installation. Aim for a clean, efficient playbook.

2. **Service Management:**

   Add a task to ensure that the installed services are **enabled** and **running** on both servers. Include a task to check the status of each service and verify that the services are running properly after installation.

   - Consider using **loops** or **lists** to handle multiple package installations efficiently, rather than creating separate tasks for each package.

3. **User Management:**

   Create a task to automate the creation and management of user accounts on **both servers**.

   - The task should create **two users** with specific usernames (your choice) and assign each user to one of two predefined groups: `jr_admins` and `sr_admins`. One user should be part of the `jr_admins` group, and the other part of the `sr_admins` group.
   - You may need to create a task that will create these groups on the targeted servers first if you do not do it manually.

4. **Variables:** Ensure that usernames, group names, and package names are **configurable using Ansible variables** for easy reuse in different environments.
5. **Error Handling:**

   Implement one or more error-handling mechanisms to ensure that your playbook handles potential failures gracefully.

   - You could use `ignore_errors`, `failed_when`, or `rescue` blocks to handle errors during execution. Include informative error messages in case something goes wrong during playbook execution.
6. **Documentation:**

   Comment your playbook clearly, explaining each task and variable used. Provide **instructions** for how to execute your playbook and verify that the configuration has been applied successfully.

## Submission Guidelines:

1. Submit your Ansible playbook file along with any necessary configuration files =. Keep in mind that there are many ways to design this, i.e. a loop instead of a dictionary list of packages, or a variable name defined for different packages. Requirement 3 mentions variables specifically and must be a part of your design.

2. Provide a brief overview of your playbook's structure, highlighting key components and design decisions. Take the perspective of how you'd explain your playbook to upper management, which often have little to no understanding of the inner-workings of IT.

3. Include documentation detailing how to execute your playbook and validate the successful completion of tasks on target servers.

4. Use the Table of Contents in your book to see what chapter you can find the information/examples you need to complete your playbook. Be sure to use the slide decks and any other resource that may be helpful in your design.

# Ansible Automation Final Exam Rubric (100 points total)

1. **Package Installation and Configuration (25 Points)**
    a. **15 Points**: Automates the installation and configuration of at least three common packages on both `servera` and `serverb`, ensuring they are installed correctly.
    b. **10 Points**: Uses loops or variables to streamline the process, avoiding repetitive tasks and improving playbook efficiency.

2. **Service Management (25 Points)**
    a. **15 Points**: Ensures that installed services are enabled and running on both servers.
    b. **10 Points**: Includes a task that verifies the status of each service and ensures they are running properly after installation.

3. **User Management (15 Points)**
    a. **10 Points**: Creates at least two users on both servers, assigning them to the predefined groups (`jr_admins` and `sr_admins`).
    b. **5 Points**: Ensures that the user creation tasks are correctly implemented and functional.

4. **Use of Variables (10 Points)**
    a. **10 Points**: Ensures that usernames, group names, and package names are configurable using Ansible variables, making the playbook reusable in different environments.

5. **Error Handling (10 Points)**
    a. **5 Points:** Implements error handling mechanisms (e.g., `ignore_errors`, `failed_when`, `fail`,`rescue` blocks) to manage potential failures during playbook execution.
    b. **5 Points**: Provides informative error messages when errors occur, ensuring the playbook fails gracefully.

6. **Documentation (15 Points)**
    a. **10 Points**: Clear and well-organized comments explaining each task, variable, and logic used in the playbook.
    b. **5 Points**: Provides detailed instructions for executing the playbook and verifying successful configuration.

## Weight Breakdown:

**Execution (50%):**

- Package Installation and Configuration (25%)
- Service Management (25%)

**Supporting Features (25%):**

- User Management (15%)
- Use of Variables (10%)

**Best Practices and Playbook Structure (25%):**

- Error Handling (10%)
- Documentation (15%)