RHEL In-place Upgrade Automation Workshop

A comprehensive approach for automating in-place upgrades of Red Hat Enterprise Linux
What you will learn

▸ Key features of the upgrade automation approach
▸ Workshop setup and walkthrough
▸ Generating pre-upgrade reports
▸ Automating recommended remediations
▸ Understand potential application impacts
▸ How upgrade from RHEL 7 and RHEL 8
▸ The importance of snapshot/rollback
▸ Ansible roles available to help you get started
Introduction

Topics Covered:

● Key features of the upgrade automation approach
● Overview of the workshop lab environment
Automation happens when one person meets a problem they never want to solve again.
RHEL In-place Upgrade Automation

Key Features to Succeed at Scale

Automate Everything
Make RHEL upgrades a push-button service easily consumable by Ops and App teams

Snapshot/rollback
Eliminate application impact risk so there is never an excuse not to upgrade

Custom Modules
Automate remediations to deal with your standard tools, agents and middleware

Reporting Dashboard
Visualize pre-upgrade results and track progress of upgrades completed across the estate
Workshop Lab Environment

- The workshop lab environment includes a number of RHEL cloud instances
- One instance is dedicated to hosting AAP and launches playbook and workflow jobs
- These jobs perform in-place upgrades on the remaining RHEL "pet app servers"
- The automation uses LVM for the snapshot/rollback capability
Lab Time - Workshop Lab Environment

Complete exercise 1.1 in your lab environment now
Section 1
Pre-upgrade Analysis
RHEL In-place Upgrade Automation Workflow

Analysis Phase

Upgrade Phase

Commit Phase

Note

The icon indicates workflow steps that are automated by Ansible playbooks.
Launching the Analysis Job Template

- Job templates are used to launch automation jobs from AAP
- We’ll use the Analysis job template to generate pre-upgrade reports for all of the RHEL pet app servers in our lab
Lab Time - Run Pre-upgrade Jobs
Complete exercise 1.2 in your lab environment now
RHEL In-place Upgrade Automation Workflow

Note
The icon indicates workflow steps that are automated by Ansible playbooks.
▸ When inhibitor risk findings are reported, the RHEL upgrade is blocked and can't proceed without first resolving the issue.

▸ Other high risk findings are no big deal as we'll learn at the end of the next exercise.
Lab Time - Review Pre-upgrade Reports
Complete exercise 1.3 in your lab environment now
RHEL In-place Upgrade Automation Workflow

The icon indicates workflow steps that are automated by Ansible playbooks.
Automating Remediation of Inhibitor Findings

- Using an Ansible playbook is the best way to automate remediations for inhibitors identified in the pre-upgrade reports.

- The `Configure sshd` task addresses the "Possible problems with remote login using root account" inhibitor.

- The `Remove pam_pkcs11 module` task will update the Leapp answer file to solve our other inhibitor.

- After running the remediation playbook job, we'll generate a fresh pre-upgrade report and see no more inhibitors.

```yaml
- name: Configure sshd
  ansible.builtin.lineinfile:
    path: "/etc/ssh/sshd_config"
    regex: "^(#)?{{ item.key }}"
    line: "{{ item.key }} {{ item.value }}"
    state: present
  loop:
    - {key: "PermitRootLogin", value: "prohibit-password"}
    - {key: "PasswordAuthentication", value: "no"}
  notify:
    - Restart sshd

- name: Remove pam_pkcs11 module
  ansible.builtin.shell: |
    set -o pipefail
    leapp answer --add --section remove_pam_pkcs11_module_check.confirm=True
  args:
    executable: /bin/bash
```
Lab Time -
Perform Recommended Remediations
Complete exercise 1.4 in your lab environment now
A custom actor can implement pre-upgrade checks specific to your enterprise.

Here we see an inhibitor raised for failed compliance with an imaginary organization's "reboot hygiene" policy.

Continue to the next exercise to learn more about custom actors and other ways to tailor automation to your needs.
Lab Time - Custom Modules
Complete exercise 1.5 in your lab environment now
To demonstrate how in-place upgrades and rollbacks might impact business applications, we’ll install a sample app.

The Spring Pet Clinic Sample Application will be installed to use a third-party JDK runtime and a local MariaDB database.

Will the application lead to more findings on the pre-upgrade report? Will it still function correctly after we perform the RHEL upgrade? Continue to the next exercise to find out!
Lab Time - Deploy a Pet App
Complete exercise 1.6 in your lab environment now
Section 2
RHEL In-place Upgrade
RHEL In-place Upgrade Automation Workflow

Analysis Phase

Upgrade Phase

Commit Phase

Note

The icon indicates workflow steps that are automated by Ansible playbooks.
Launching the Upgrade Workflow Job Template

- A workflow job template in AAP is used to run the snapshot and upgrade playbooks back-to-back in a single job.
- This job will take about 20 minutes to finish the upgrades on all the RHEL hosts in our lab. While we are waiting, we’ll read up to learn more about how Leapp works.
Lab Time - Run OS Upgrade Jobs
Complete exercise 2.1 in your lab environment now
Leapp Framework Overview

- Keep in mind that the Leapp framework is responsible only for upgrading the RHEL OS packages. Additional tasks required for upgrading your standard agents, tools, middleware, etc., need to be included in the upgrade playbooks you develop to deal with the specific requirements of your organization's environment.

- The Leapp framework performs the RHEL in-place upgrade by following a sequence of phases as shown in this flowchart found in the Leapp developer docs. The phases of the RHEL in-place upgrade are implemented in modules known as Leapp actors.

- Phases under the Old System group run under the existing RHEL installed version. The Interim System phases starts when the host reboots to an upgrade environment under which the network and other services are not started. It is at this time that all RHEL packages are upgraded. Once all the packages are upgraded, another reboot brings the host up under the new RHEL major version and the FirstBoot phase starts. This final phase runs a few post-upgrade actors that require network access and then the upgrade is done.

## Different Snapshot Options Compared

<table>
<thead>
<tr>
<th>Snapshot type</th>
<th>Works with</th>
<th>Benefits</th>
<th>Drawbacks</th>
</tr>
</thead>
</table>
| LVM           | • Bare metal  
• On-prem VMs  
• Cloud* | • No external API access required  
• Scope can be just OS or everything | • Free space required in volume group  
• Snapshots can run out of space if not sized correctly  
• Automation must backup and restore /boot separately |
| VMware        | • On-prem VMs (ESX) | • Simple and reliable  
• Scope includes everything | • Doesn’t support bare metal, etc.  
• Using VMware snapshot for over 3 days is discouraged  
• Getting API access can be difficult  
• No free space in datastores because of overcommitment  
• Everything scope might be too much |
| Amazon EBS    | • Amazon EC2 | • Simple and reliable  
• Unlimited storage capacity  
• Scope can be just OS or everything | • Only works on AWS |
| Break Mirror  | • Bare metal | • Alternative to LVM for servers with hardware RAID | • Significant development and testing effort required  
• RAID and Redfish API standards vary across different vendors and hardware models |
| ReaR          | • Bare metal  
• On-prem VMs | • Method of last resort if no snapshot options will work | • Not really a snapshot, but does offer boot ISO full recovery capability |
Lab Time - Let’s Talk About Snapshots
Complete exercise 2.2 in your lab environment now
Checking Upgrade Job Log Output

- After the upgrade playbook job finishes, review the play recap in the log output.
- All hosts should show failed=0 indicating that the upgrades are completed.
- The application teams can now start their validation and acceptance testing.
Checking the RHEL and Kernel Versions

- You can refresh the RHEL Web Console system overview page to show the upgraded RHEL version.

- If you prefer the shell prompt, try these commands:
  - `cat /etc/redhat-release`
  - `uname -r`
Lab Time - Check if the Upgrades Worked
Complete exercise 2.3 in your lab environment now
RHEL In-place Upgrade Automation Workflow

Analysis Phase

Start
- Run Pre-upgrade Analysis
- Review Report
- Looks OK?
  - NO: Apply Recommended Remediation
  - YES: Create Snapshot

Upgrade Phase
- Run In-place Upgrade
- App Team Validations
  - Looks OK?
    - NO: Rollback
    - YES: Can Fix?
      - NO: Delete Snapshot
      - YES: Done

Commit Phase

Note: The icon indicates workflow steps that are automated by Ansible playbooks.
Lab Time - How is the Pet App Doing?
Complete exercise 2.4 in your lab environment now
Section 3
Rolling Back
RHEL In-place Upgrade Automation Workflow

Analysis Phase

Upgrade Phase

Commit Phase

Note

The icon indicates workflow steps that are automated by Ansible playbooks.
Simulate a Failed Upgrade or Application Impact

- To help demonstrate the effectiveness of rolling back, we’re going to intentionally mess up one of our upgraded hosts.

- What if the RHEL upgrade had caused our temurin-17-jdk 3rd-party JDK runtime package to be removed because of unresolvable dependencies? Our pet app requires the JDK runtime to function. Without it, our application will be broken. We can simulate this by manually removing the package like this:

  ```
  sudo dnf -y remove temurin-17-jdk
  ```
Lab Time - Trash the Instance
Complete exercise 3.1 in your lab environment now
Rolling Back the Upgrade

- The rollback playbook uses the snapshots that were automatically created by the upgrade workflow job and reverts an instance back to its previous state.

- This job completed in just under 3 minutes!

- After rolling back, we see the RHEL and kernel versions are the same as before we upgraded.
Lab Time - Run Rollback Job
Complete exercise 3.2 in your lab environment now
Is Everything Working Again?

- After rolling back, repeat the observations we made on our host after the upgrade with the expectation that everything is back as it was before the upgrade.
- Look for any app data you added or modified after the upgrade and you will find that all those changes are preserved.
- What does this tell us about the snapshot scope implemented by our rollback playbook?
- In the next exercise, we will assess our state after rolling back and consider next steps.
Lab Time - Check if Upgrade Undone
Complete exercise 3.3 in your lab environment now
Congratulations!

- You have reached the end of the workshop. You are now armed with the knowledge needed to start developing an automation solution to help your organization manage RHEL upgrades at scale.

- The workshop lab environment is now yours to play with. Dream up your own ideas for additional learning and experimentation. Remember you can upgrade and roll back as often as you like. Rinse and repeat!
Lab Time - Rinse and Repeat
Complete exercise 3.4 in your lab environment now
Checkout the Code

- All of the Ansible roles and playbooks used in this workshop are maintained in open source repositories.
- Take some time to review the code and get engaged with the communities supporting these resources.

- [github.com/redhat-cop/infra.leapp](https://github.com/redhat-cop/infra.leapp)
  The `infra.leapp` collection provides the Ansible role that generates the pre-upgrade reports and another that is used to perform the RHEL upgrades. This collection uses the Leapp framework for upgrades from RHEL 7 and later, but also supports upgrading from RHEL 6 using the older Red Hat Upgrade Tool. The collection is published on Ansible Galaxy [here](https://galaxy.ansible.com/redhat-cop/infra.leapp) and also available from Ansible Automation Hub validated content [here](https://github.com/redhat-cop/infra.leapp). If you are planning to do RHEL in-place upgrades for your organization, these roles will help you quickly roll out proof-of-concept automation and start upgrading.

- [github.com/swapdisk/infra.lvm_snapshots](https://github.com/swapdisk/infra.lvm_snapshots)
  Here you will find work in progress on a new Ansible role for managing snapshot sets using LVM. The collection also includes roles that can be used to shrink LVM logical volumes to free up volume group space for snapshots and another that makes it possible to increase the size of /boot partitions.

- [github.com/oamg/leapp-supplements](https://github.com/oamg/leapp-supplements)
  Leapp Supplements is a repository of example Leapp custom actors. The CheckRebootHygiene actor that was demonstrated in the optional Custom Pre-upgrade Checks exercise is maintained here. There is also a Makefile and RPM spec file that can be used to build packages for installing your Leapp custom actors.

- [github.com/redhat-partner-tech/leapp-project](https://github.com/redhat-partner-tech/leapp-project)
  This is where you will find all of the AAP job templates and Ansible playbooks included in the workshop. You can also explore the infrastructure as code (IaC) magic that is used to provision the workshop 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