Below is a post-install configuration script that prepares the deployed VM image for bare-metal use, ensuring that it is generalized and ready to be used on different hardware. Additionally, I'll explain how the same Kickstart file can be modified slightly to deploy to another VM instead of bare metal.

Post-Install Configuration for VM to Bare-Metal Preparation

This post-installation script (%post) will run after the image has been deployed to the bare-metal system. It ensures the system is generalized and ready for use across different hardware configurations.

bash

%post --interpreter=/bin/bash

# Clear the machine ID (so a new one is generated on first boot)

truncate -s 0 /etc/machine-id

rm -f /var/lib/dbus/machine-id

# Clean up cloud-init (if installed)

if [ -x /usr/bin/cloud-init ]; then

cloud-init clean

rm -rf /var/lib/cloud/\*

fi

# Remove persistent udev rules for network interfaces (so they are regenerated based on actual hardware)

rm -f /etc/udev/rules.d/70-persistent-net.rules

# Remove SSH host keys (so new ones are generated on first boot)

rm -f /etc/ssh/ssh\_host\_\*

# Relabel filesystems for SELinux

touch /.autorelabel

# Log the completion of the preparation process

echo "Bare-metal preparation completed on $(hostname)" > /root/prep\_baremetal.log

%end

Explanation of the Post-Install Script

Machine ID: The machine ID is cleared so that the system generates a new one when it boots on different hardware.

Cloud-Init Cleanup: If cloud-init is installed, it’s cleaned up to remove any previous instance data, making the system ready for a new environment.

Udev Rules: Persistent udev rules for network interfaces are removed to ensure the system correctly identifies network interfaces based on the actual hardware of the bare-metal system.

SSH Host Keys: SSH host keys are removed so that new ones are generated on the first boot, ensuring security.

SELinux Relabeling: The file systems are marked for relabeling to ensure SELinux operates correctly on the new hardware.

Logging: A log entry is created to record the completion of the preparation process.

Using the Same Kickstart File for Another VM

To use the same Kickstart file to deploy the image to another VM instead of a bare-metal system, you can follow these steps:

Remove Disk Size Check: Since VMs typically have predefined disk sizes, you might not need the dynamic partitioning based on disk size. You can simplify the partitioning section or keep it if your VM sizes vary significantly.

Modify the Image Deployment Section: If you’re deploying the image to another VM, you can adjust the %pre section that handles the image download and deployment. For example, instead of downloading the image, you could use a local path within the VM or another network location.

Here's a simplified version for VM deployment:

bash

%pre

# Use the raw image created in KVM to populate the root partition

dd if=/path/to/your/image.raw of=/dev/VolGroup00/root bs=4M status=progress

%end

Adjust Networking: Ensure that the network configuration in the Kickstart file aligns with the VM environment, such as using network --bootproto=dhcp or static IP settings as required.

Reuse the Post-Install Script: The post-installation script can remain largely the same, as it helps generalize the system regardless of whether it's deployed on bare-metal or as another VM.

Summary

Generalization: The post-install script ensures the system is generalized and prepared for use on different hardware or as another VM.

Flexibility: The same Kickstart file can be used to deploy to both bare-metal and VM environments, with slight modifications to the partitioning and image deployment sections.

Consistency: This approach allows you to maintain a consistent deployment process across different environments.

You can now deploy your pre-configured KVM image to both bare-metal systems and other VMs using this Kickstart file, with the post-install script ensuring that each deployment is ready for its specific environment.