Below is a comprehensive Kickstart file that includes both dynamic partitioning and the deployment of a pre-created image from KVM onto bare-metal systems. This file is designed to be flexible, adapting to different hardware configurations while ensuring that the necessary partitions are created according to best practices.

Kickstart File for Bare-Metal Deployment with Dynamic Partitioning

# System language

lang en\_US.UTF-8

# Keyboard layouts

keyboard us

# System timezone

timezone America/New\_York --isUtc

# Root password (replace with your encrypted password)

rootpw --iscrypted $6$somehashedpassword

# Network information

network --bootproto=dhcp --device=eth0

# System authorization information

auth --useshadow --passalgo=sha512

# SELinux configuration

selinux --enforcing

# Firewall configuration

firewall --enabled --ssh

# Do not configure X Window System

skipx

# Reboot after installation

reboot

# Installation source (adjust the URL to match your environment)

url --url="http://192.168.1.10/rocky"

# Clear all partitions and create a new disk label

zerombr

clearpart --all --initlabel

# Dynamic partitioning script

%pre

# Check disk size and dynamically adjust partition sizes

disk\_size=$(lsblk -b -d -n -o SIZE /dev/sda)

# Create a basic partition scheme

echo 'clearpart --all --initlabel' > /tmp/partitioning.ks

echo 'part /boot/efi --fstype="efi" --size=512 --ondisk=sda' >> /tmp/partitioning.ks

if [ "$disk\_size" -gt 100000000000 ]; then

# Large disk (greater than 100GB)

echo 'part /boot --fstype="ext4" --size=1024 --ondisk=sda' >> /tmp/partitioning.ks

echo 'part pv.01 --size=1 --grow --ondisk=sda' >> /tmp/partitioning.ks

else

# Smaller disk (less than 100GB)

echo 'part /boot --fstype="ext4" --size=512 --ondisk=sda' >> /tmp/partitioning.ks

echo 'part pv.01 --size=1 --grow --ondisk=sda' >> /tmp/partitioning.ks

fi

# Create Volume Group

echo 'volgroup VolGroup00 pv.01' >> /tmp/partitioning.ks

# Logical Volumes (adjust sizes based on available disk space)

echo 'logvol / --fstype="ext4" --size=10240 --name=root --vgname=VolGroup00' >> /tmp/partitioning.ks

echo 'logvol swap --fstype="swap" --size=2048 --name=swap --vgname=VolGroup00' >> /tmp/partitioning.ks

echo 'logvol /tmp --fstype="ext4" --size=2048 --name=tmp --vgname=VolGroup00' >> /tmp/partitioning.ks

echo 'logvol /sto --fstype="ext4" --size=8192 --name=sto --vgname=VolGroup00' >> /tmp/partitioning.ks

echo 'logvol /mnt/data --fstype="ext4" --size=8192 --name=data --vgname=VolGroup00' >> /tmp/partitioning.ks

echo 'logvol /home --fstype="ext4" --size=4096 --name=home --vgname=VolGroup00' >> /tmp/partitioning.ks

echo 'logvol /var --fstype="ext4" --size=4096 --name=var --vgname=VolGroup00' >> /tmp/partitioning.ks

echo 'logvol /var/tmp --fstype="ext4" --size=2048 --name=var\_tmp --vgname=VolGroup00' >> /tmp/partitioning.ks

echo 'logvol /var/log --fstype="ext4" --size=4096 --name=var\_log --vgname=VolGroup00' >> /tmp/partitioning.ks

echo 'logvol /var/log/audit --fstype="ext4" --size=2048 --name=var\_log\_audit --vgname=VolGroup00' >> /tmp/partitioning.ks

%end

# Include the dynamically generated partitioning scheme

%include /tmp/partitioning.ks

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

%pre

# Download the VM image and write it to disk

curl -o /tmp/image.raw http://192.168.1.10/images/rocky9.4-baremetal-template.raw

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

%end

# Post installation script (optional customization)

%post

# Ensure proper file labeling for SELinux

touch /.autorelabel

echo "Deployment completed on $(hostname)" > /root/deployment.log

%end

# Bootloader configuration

bootloader --location=mbr --append="rhgb quiet"

**Explanation of the Kickstart File**

1. **Basic Configuration**:
   * **Language, timezone, and network** settings are defined at the beginning.
   * **Root password**: The password is encrypted and set securely.
   * **Firewall**: SSH access is enabled.
2. **Dynamic Partitioning**:
   * The %pre section dynamically creates the partitioning scheme based on the detected disk size.
   * It creates a volume group (VolGroup00) and logical volumes (/, swap, /tmp, etc.) with sizes adjusted according to the available disk space.
3. **Image Deployment**:
   * Another %pre section downloads the raw image that was created in KVM and writes it directly to the root logical volume (/dev/VolGroup00/root).
   * This replaces the root partition with the pre-configured image, essentially deploying the VM to the bare-metal system.
4. **Post-Installation Customization**:
   * The %post section handles any additional configuration, such as ensuring proper SELinux labeling and logging deployment details.
5. **Bootloader Configuration**:
   * The bootloader is configured to install on the MBR, with standard boot options.

**How This Kickstart File Works**

* **Flexibility**: The Kickstart file is designed to handle different hardware configurations by dynamically adjusting partition sizes based on the available disk space.
* **Automation**: The pre-created image from KVM is deployed automatically onto the bare-metal system, reducing the need for manual configuration.
* **Compliance**: The partitioning scheme is designed to meet best practices, including a separate /tmp partition as required by DISA STIG.

**Deployment Process**

1. **PXE Boot**: The bare-metal system boots via PXE and starts the Kickstart process.
2. **Partitioning**: The %pre script dynamically partitions the disk according to the available space.
3. **Image Deployment**: The raw image is downloaded and written to the root partition.
4. **Customization**: The %post script handles any final setup tasks.
5. **Reboot**: The system reboots, and the deployment is complete.

This approach ensures that you can efficiently deploy a pre-configured KVM image to multiple bare-metal systems with different hardware configurations, all while adhering to security and best practice guidelines.