Kickstart Server - Concept and Working

This document explains the concept and working of a Kickstart server, along with a step-by-step guide covering server setup and automated client OS installation. The provided configuration uses the following details:

Kickstart URL: http://172.16.0.126/ks.cfg
Repository URL: http://172.16.0.126/rocky8/

The Kickstart configuration file (ks.cfg) is created using the file generated by Anaconda during a manual installation of the Rocky Linux operating system and is later customized as required.

1. Concept of Kickstart Server

Kickstart is a method used to automate the installation of Linux operating systems such as Rocky Linux. It allows administrators to predefine installation settings in a configuration file (ks.cfg), enabling the operating system to be installed without any manual intervention. This approach is especially useful when deploying multiple systems with the same configuration.

A Kickstart server hosts the Kickstart configuration file and the operating system installation repository on a network-accessible server. Client machines can boot via PXE (network boot) or installation media and fetch the Kickstart file to perform an unattended installation.

2. How Kickstart Works

1. The administrator prepares a ks.cfg file containing installation preferences such as disk partitioning, package selection, user creation, and network settings.
2. The Kickstart file is stored on a network-accessible server using HTTP, FTP, or NFS.
3. The operating system installation ISO contents are copied to a network-accessible location to act as an installation repository.
4. The client system boots via PXE (network boot) and is instructed to use the Kickstart file for automated installation.
5. The installer reads the ks.cfg file and installs the operating system automatically according to the predefined settings.

3. Step-by-Step Process

Follow these steps to set up and use the Kickstart server:

Step 1: Install and Configure the Web Server

1. Install Apache HTTP server: **sudo dnf install httpd -y**
2. Enable and start Apache: **sudo systemctl enable httpd --now**
3. Ensure firewall allows HTTP:

```
sudo firewall-cmd --add-service=http --permanent
sudo firewall-cmd --reload
```

Step 2: Prepare the Kickstart File

1. After installing Rocky Linux manually, copy the Anaconda-generated Kickstart file:

```
sudo cp /root/anaconda-ks.cfg /var/www/html/ks.cfg
```

2. Modify **ks.cfg** as needed (packages, network settings, etc.).
3. Verify it is accessible in a browser at:
   http://172.16.0.126/ks.cfg

Step 3: Set Up the Installation Repository

1. Mount the Rocky Linux installation ISO: **sudo mount -o loop /Path to the ISO File/Rocky-8.x.iso /mnt**
2. Create & Copy its contents to the web server directory:

=> **sudo mkdir -p /var/www/html/rocky8 && sudo cp -rv /mnt/* /var/www/html/rocky8/**

3. Verify repository is accessible via browser at:
   http://172.16.0.126/rocky8/ [Note:you have to use your kickstart servers ip]

4. You will see the files like **Appstream** and **BaseOs** etc

```
=> AppStream/
   BaseOS/
   EFI/
   images/
   isolinux/
   repodata/
   media.repo
   TRANS.TBL
```

Step 4: Install and Configure DHCP Server

1. Install the DHCP package: **sudo dnf install dhcp-server -y**

2. Edit the DHCP configuration file: **sudo nano /etc/dhcp/dhcpd.conf**

   => Example configuration (adjust network/subnet to your setup):

```
default-lease-time 600;     # Default IP lease time
max-lease-time 7200;        # Maximum IP lease time
authoritative;              # Main DHCP server
log-facility local7;        # Syslog facility for DHCP logs
```

```
# Used to detect BIOS vs UEFI clients
option architecture-type code 93 = unsigned integer 16;

subnet 192.168.0.0 netmask 255.255.255.0 {

    range 192.168.0.100 192.168.0.110;   # IP range for PXE clients

    option routers 192.168.0.1;          # Default gateway
    option subnet-mask 255.255.255.0;
    option domain-name-servers 8.8.8.8, 1.1.1.1;
    option broadcast-address 192.168.0.255;

    next-server 192.168.0.10;            # kickstart server IP

    # Serve bootloader based on client firmware
    if option architecture-type = 00:07 or option architecture-type = 00:09 {
        filename "grubx64.efi";          # UEFI clients
    } else {
        filename "pxelinux.0";           # Legacy BIOS clients
    }
}
```

3. Enable and start DHCP: **sudo systemctl enable dhcpd –now**

   [Note:Enabling the DHCP service causes it to start automatically at boot; running multiple DHCP servers on the same network segment may lead to IP address conflicts, so verify the network environment before enabling it. Alternatively, you can start the service manually when needed using sudo systemctl start dhcpd.]

Step 5: Install and Configure TFTP + PXELINUX Bootloader

In this step, the TFTP server is configured to provide PXE boot files to client machines.
This setup supports both Legacy (BIOS) and UEFI based systems.

1. **Step 1.1:** Install and Start TFTP Service
   Step 1.2: Install the required packages for TFTP and PXE boot: **dnf install tftp-server syslinux -y**
   Step 1.3: Enable and start the TFTP service: **systemctl enable tftp --now**

2. Configure TFTP Root Directory
   Step 2.1: Create the TFTP root directory: **mkdir -p /var/lib/tftpboot**

3. **PXE Bootloader Configuration for Legacy (BIOS) Clients**
   Legacy BIOS systems use PXELINUX as the PXE bootloader.
   Step 3.1: Copy PXELINUX bootloader and required support files:
   cp /usr/share/syslinux/pxelinux.0 /var/lib/tftpboot/
   cp /usr/share/syslinux/menu.c32 /var/lib/tftpboot/
   cp /usr/share/syslinux/ldlinux.c32 /var/lib/tftpboot/
   cp /usr/share/syslinux/libutil.c32 /var/lib/tftpboot/
   cp /usr/share/syslinux/libcom32.c32 /var/lib/tftpboot/
   OR
   You can use following single command :
   => **cp /usr/share/syslinux/{pxelinux.0,menu.c32,ldlinux.c32,libutil.c32,libcom32.c32} /var/lib/tftpboot/**
   Step 3.2: Create the PXELINUX configuration directory: **mkdir -p /var/lib/tftpboot/pxelinux.cfg**

4. Create the default PXE menu configuration file: **nano /var/lib/tftpboot/pxelinux.cfg/default**
   Step 4.1: Example PXELINUX configuration (Legacy BIOS):
   =>
```
DEFAULT menu.c32
PROMPT 0
TIMEOUT 30
MENU TITLE PXE Boot Menu (Legacy BIOS)

LABEL rocky8
MENU LABEL Install Rocky Linux 8 (Kickstart)
KERNEL vmlinuz
APPEND initrd=initrd.img inst.ks=http://172.16.0.126/ks.cfg
inst.repo=http://172.16.0.126/rocky8
```

   The above configuration file is used only by **Legacy** BIOS clients.

5. PXE Bootloader Configuration for UEFI Clients
   UEFI systems use **GRUB2** as the PXE bootloader.
   Step 5.1: Copy the GRUB EFI bootloader: **cp /boot/efi/EFI/rocky/grubx64.efi /var/lib/tftpboot/**
   Step 5.2: Create the GRUB configuration file: **nano /var/lib/tftpboot/grub.cfg**
   =>
```
# GRUB configuration for PXE booting Rocky Linux 8
# This menu entry boots the installer using PXE and Kickstart

set timeout=5        # Time (in seconds) before default entry is selected
set default=0        # Default menu entry index

menuentry 'Install Rocky Linux 8 (PXE + Kickstart)' {

    # Kernel image for Rocky Linux installer
    linuxefi rocky8/vmlinuz \
    ip=dhcp \
    inst.repo=http://172.16.0.126/rocky8 \
    inst.ks=http://172.16.0.126/ks.cfg

    # Initial RAM disk for installer
    initrdefi rocky8/initrd.img
}
```

The above configuration file is used only by UEFI clients.

6. Common Boot Files for Both Legacy and UEFI
   Both Legacy BIOS and UEFI clients use the same kernel and initrd files.
   Step 6.1: Mount the Rocky Linux ISO: **mount -o loop /path/to/Rocky-8.x.iso /mnt**
   [If the ISO is already mounted in **Step 3**, skip this step and copy the files below.]
   Step 6.2: Copy **kernel** and **initrd** files to the TFTP directory:
   cp /mnt/images/pxeboot/vmlinuz /var/lib/tftpboot/
   cp /mnt/images/pxeboot/initrd.img /var/lib/tftpboot/
   OR
   You can use following single command :
   sudo cp /mnt/images/pxeboot/{vmlinuz,initrd.img} /var/lib/tftpboot/

Step 6: Copy Kernel and Initrd for PXE Boot

1. Mount Rocky ISO again if not already mounted:**sudo mount -o loop /Path/to/Rocky-8.x.iso /mnt** (If you have already done this then please skip this step no need to repeat)
2. Copy kernel and initrd files to TFTP directory:
   sudo cp /mnt/images/pxeboot/vmlinuz /var/lib/tftpboot/
   sudo cp /mnt/images/pxeboot/initrd.img /var/lib/tftpboot/

Step 7: Firewall Rules for PXE Services

1. Allow TFTP and DHCP in firewall:

=> **sudo firewall-cmd --add-service=dhcp --permanent**
sudo firewall-cmd --add-service=tftp --permanent
sudo firewall-cmd –reload

Step 8: Test PXE Boot

1. Boot a client machine over network (enable **PXE Boot** in BIOS/UEFI).
2. It should get an IP from DHCP, fetch PXELINUX from TFTP, load the boot menu, and then automatically install using:
   - Kickstart file → `http://172.16.0.126/ks.cfg/`
   - Repository → `http://172.16.0.126/rocky8/`

Step 9: Boot the Client Machine

1. Boot the client system using PXE (Network Boot).
2. The client receives an IP address and boot information from the DHCP server.
3. The PXE boot menu is displayed automatically.
4. When the Rocky Linux option is selected (or after timeout), the installer starts.
5. Kernel boot parameters such as:
   inst.ks=http://172.16.0.126/ks.cfg
   inst.repo=http://172.16.0.126/rocky8
   are already provided via PXE configuration.
6. The system installs automatically without any manual intervention.

Step 10: Completion

Once installation is complete, the system will reboot into the newly installed OS with all settings applied as per the Kickstart file.

4. Kickstart Installation Workflow (Behind the Scenes)

1. **Client Boots via PXE (Network Boot)**
   - Student powers on the client machine → BIOS/UEFI is set to **PXE Boot**.
   - Client broadcasts **DHCP Discover** to request an IP.
2. **DHCP Server Responds**
   - Kickstart server's **DHCP service** replies:
     1. Assigns an IP address (from DHCP range).
     2. Informs client about: **TFTP server IP** (next-server).
     3. Detect client architecture:
        i) Legacy (BIOS) : sends pxelinux.0
        ii)UEFI : sends grubx64.efi / bootx64.efi

👉 *This is where /etc/dhcp/dhcpd.conf is used.*

3. **Client Downloads Bootloader from TFTP**
   - Client contacts the **TFTP server** and downloads:
     1) If client is on legacy it will download following files :
        i) pxelinux.0 (the PXE bootloader)
        ii) ldlinux.c32,menu.c32,etc. (boot menu helpers)
        iii) pxelinux.cfg/default (tells what OS options are available)

👉 *This is where /var/lib/tftpboot/pxelinux.cfg/default is used.*

4. **PXE Boot Menu Loads**
   - Client displays the PXE boot menu.
   - If Rocky Linux option is chosen (or auto after timeout):

- Downloads **kernel** (vmlinuz).
- Downloads **initrd** (initrd.img).

👉 *These files come from /mnt/images/pxeboot/ copied to /var/lib/tftpboot/.*

5. **Kernel + Initrd Start the Installer (Anaconda)**

5.1. Now the Rocky Linux installer (**Anaconda**) starts.
5.2. It looks at the **boot parameters** (from PXE config):

5.2.1. inst.ks=http://172.16.0.126/ks.cfg → use this Kickstart file.

5.2.2. inst.repo=http://172.16.0.126/rocky8 → use this repo for packages.

5.3. HTTP server on Kickstart server provides:
5.4. i) Kickstart file (ks.cfg)
5.5. ii) Rocky Linux installation repository

👉 *This is where ks.cfg and the web repo (/var/www/html/rocky8/) are used.*

6. **Kickstart File Executes**

6.1. Anaconda fetches the Kickstart file (ks.cfg).
6.2. It reads instructions such as:
6.2.1. Disk partitioning.
6.2.2. Root password.
6.2.3. Package selection (e.g., minimal/server).
6.2.4. Post-install scripts (like creating users).

👉 *This is where /var/www/html/ks.cfg is used.*

7. **Package Installation from Repo**
   - Installer downloads required RPMs from the repo served over **HTTP**.
   - BaseOS/ and AppStream/ directories are used.

👉 *This is where /var/www/html/rocky8/ files are used.*

8. **Post-Install Configuration**
   - Any post-install scripts in Kickstart (%post section) run.
   - Example: configure hostname, enable services, custom setup.

9. **Installation Completes → System Reboots**
- After reboot, the client has a fresh Rocky Linux installation.
- All steps were automated without manual input.

5. Kickstart PXE Installation Workflow (In Short Steps)

1. Client boots via PXE and sends a DHCP Discover request.
2. DHCP server assigns an IP address and provides TFTP server details along with the bootloader filename
   (pxelinux.0 for Legacy BIOS clients, grubx64.efi for UEFI clients).

3. Client contacts the TFTP server and downloads the bootloader and PXE configuration files
   (pxelinux.cfg/default for Legacy clients, grub.cfg for UEFI clients).
4. PXE boot menu is displayed and the Rocky Linux option is selected automatically or manually.
5. Kernel (vmlinuz) and initrd (initrd.img) are loaded from the TFTP server.
6. Rocky Linux installer (Anaconda) starts.
7. Anaconda reads the kernel boot parameters and fetches the Kickstart file (ks.cfg) from the HTTP server.
8. Installer downloads required packages from the HTTP repository (BaseOS and AppStream).
9. Kickstart file executes, performing disk partitioning, package installation, user creation, and post-install scripts.
10. Installation completes and the system reboots into the newly installed operating system.