GETTING STARTED

WITH

SYSTEM BOOT PROCESS

➢ SYSTEM BOOT PROCESS:

- An important and powerful aspect of Red Hat Enterprise Linux is the open, user-configurable method it uses for starting the operating system. Users are free to configure many aspects of the boot process, including specifying the programs launched at boot-time.
- Below are the basic stages of the boot process for an x86 system:
    - The system **BIOS / UEFI** checks the system and launches the first stage boot loader on the **MBR / GUID Partition Table (GPT)** of the primary hard disk.
    - The first stage boot loader loads itself into memory and launches the second stage boot loader from the **/boot/ partition**.
    - The second stage boot loader loads the kernel into memory, which in turn loads any necessary modules and mounts the **root partition read-only**.
    - The kernel transfers control of the boot process to the **/sbin/system or init** program.
    - The **/sbin/systemd** program loads all services and user-space tools, and mounts all partitions listed in **/etc/fstab.**
    - The user is presented with a login screen for the freshly booted Linux system.

UNDERSTANDING BOOT PROCESS IN DEPTH:

BIOS / UEFI:

- BIOS and UEFI are two essential firmware interfaces responsible for initializing hardware components, running system diagnostics, and supporting the startup of the operating system on a computer.

BIOS (BASIC INPUT OUTPUT SYSTEM):

- As soon as the computer is switched on, the BIOS assumes command and executes the **Power-On Self-Test (POST)** to ensure the essential hardware components – including the RAM, CPU, and storage devices – are all functioning properly.
- Once the boot device has been identified, the BIOS proceeds to search for **Master Boot Record (MBR)** on the storage device. These contain the crucial initial boot loader code.

MBR (Master Boot Record):

- The MBR is located in the first sector of a storage device (usually the first 512 bytes of a hard drive or SSD). It's in a fixed location, the LBA (Logical Block Address) 0.
- The Master Boot Record (MBR) if of 512 bytes in size. It consists of three components:
  - ✓ The primary boot loader information occupies the initial 446 bytes.
  - ✓ Following that, the partition table information fills the subsequent 64 bytes.
  - ✓ Lastly, the MBR validation check, also known as the magic number, resides in the final 2 bytes.
- MBR has limitations in supporting only four primary partitions or three primary partitions and one extended partition, which can further contain multiple logical partitions. This restricts the number of partitions usable on a disk.
- MBR uses 32-bit addressing, limiting disk sizes to 2 terabytes (TB). Larger disks cannot be fully utilized under MBR due to this limitation.
- The BIOS then dutifully passes the reins to the designated boot loader, such as GRUB / GRUB2 for Linux operating systems.

UEFI (UNIFIED EXTENSIBLE FIRMWARE INTERFACE):

- UEFI is a more modern and versatile replacement for BIOS. It provides more advanced features and capabilities than BIOS.
- Similar to BIOS, UEFI starts with the hardware initialization and system checks. But UEFI supports more modern hardware standards and allows for faster boot times compared to traditional BIOS.
- UEFI introduced Secure Boot, a security feature that verifies the digital signatures of boot loaders and operating system kernels during the boot process. This helps prevent the loading of unauthorized or malicious code during boot time.
- Once the boot device has been identified, the UEFI proceeds to search for **GUID Partition Table (GPT)** on the storage device.
- UEFI requires a specific partition known as the **UEFI System Partition (ESP),** which is a primary component of the GPT scheme.

- UEFI firmware uses information stored in the GPT to locate the UEFI boot loader. The boot loader is stored in the ESP and is specified in the firmware's boot configuration data.
- GPT and UEFI work together to support Secure Boot functionality.
- GPT supports disk sizes larger than 2TB, addressing the limitations of the MBR partitioning scheme.
- It efficiently manages partitions on larger disks and provides scalability for future storage needs.

DIFFERENCES BETWEEN BIOS AND UEFI:

- BIOS uses the Master Boot Record (MBR) method, while UEFI uses the GUID Partition Table (GPT) method.
- UEFI is more flexible and supports larger storage capacities, modern hardware, and faster boot times compared to BIOS.
- UEFI introduces Secure Boot, enhancing system security by verifying the authenticity of bootloader and OS components.

GRUB / GRUB2:

- **GRUB stands for Grand Unified Boot Loader.** It's an old boot loader in the Linux world, responsible for managing the boot process of a computer.
- RHEL7 is distributed with v2 of the **GNU Grand Unified Bootloader (GRUB 2),** which allows the user to select an operating system or kernel to be loaded at system boot time. GRUB 2 also allows the user to pass arguments to the kernel.
- GRUB 2 reads its configuration from the **/boot/grub2/grub.cfg** file on traditional BIOS-based machines and from the **/boot/efi/EFI/redhat/grub.cfg** file on UEFI machines.

INITRD (INITIAL RAM DISK) IMAGE:

- The **initrd (initial RAM disk)** and **initramfs (initial RAM File System)** are different methods we can use to load a temporary root file system to the RAM for successful booting.
- Initramfs/initrd is used as the first root filesystem that your machine has access to.

    - Traditionally, initrd was used, but modern systems often use initramfs (a more flexible successor). Initramfs is a cpio archive that is uncompressed into a RAM disk at boot time.
    - It's more versatile, allowing for a more modular approach to including essential files and drivers.

KERNEL:

- The kernel is the core of the operating system, managing hardware resources, providing abstractions, and controlling interactions between hardware and software.
- After the initrd image completes its tasks, the kernel takes control. It initializes the system hardware, mounts the root file system, and begins the user-space initialization process.
- The kernel uses information provided by the initrd to mount the actual root file system (for example, ext4, XFS) specified in the boot parameters.

RootFS:

  - The Root File System (rootfs) is a critical component in the booting process of an operating system. It is the top-level directory hierarchy of the file system and contains essential system files and directories.
  - In the context of the booting process, the root file system is the initial file system that the operating system kernel mounts during the boot sequence.

INIT / SYSTEMD PROCESS:

- It is the first process that runs in the system with process ID of 1.
- After the kernel has loaded and initialized, it hands over control to the system/init process.
- Traditional Unix systems used the init process with different runlevels, where each runlevel represented a different system state. But modern Linux systems, including those based on Red Hat Enterprise Linux (RHEL), have transitioned to using systemd, which serves as a replacement for init and introduces a more flexible and efficient approach to managing system initialization.

- systemd reads its configuration from unit files located in directories such as /etc/systemd/system and /usr/lib/systemd/system.
- System daemons, also known as background processes or services
- During the boot process, the init or systemd process is responsible for starting system daemons.
- Following examples of various **systemd unit types:**

Service	**:** Controls and manages individual system services.
Target	**:** Represents a group of units that define system states.
Device	**:** Manages hardware devices and their availability.
Mount	**:** Handles file system mounting.
Timer	**:** Schedules tasks to run at specific intervals.

RUNLEVELS / TARGETS:

- When a system starts, systemd activates the default.target symbolic link, which points to the true target unit.
- Following are the target groups:
  - **poweroff.target (runlevel 0)** **:** Poweroff or Shutdown the system.
  - **rescue.target (runlevel 1)** **:** launches a rescue shell session.
  - **multi-user.target (runlevel 2,3,4):** Non-graphical / multi-user system
  - **graphical.target (runlevel 5)** **:** Graphical multi-user interface
  - **reboot.target (runlevel 6)** **:** Reboots the system.

- Once system daemons are initialized, the system is ready to handle user interactions. That means the user is presented with a login screen for the freshly booted Linux system.

CHANGING THE DEFAULT TARGET TO BOOT INTO:

- Each target represents a certain level of functionality and is used for grouping other units.
- When you set a default target unit, the current target remains unchanged until the next reboot.

→ **The current default target unit systemd uses to start the system:**

```
#systemctl get-default
```

→ **List the currently loaded targets:**

```
#systemctl list-units --type target
```

→ **Configure the system to use a different target unit by default:**

```
#systemctl set-default <name>.target

#systemctl set-default multi-user.target
```

→ **Verify the default target unit:**

```
#systemctl get-default
```

→ **Apply the changes by rebooting:**

```
#reboot
```

CHANGING THE CURRENT TARGET:

- On a running system, you can change the target unit in the current boot without reboot. If you switch to a different target, **systemd** starts all services and their dependencies that this target requires, and stops all services that the new target does not enable.
- Isolating a different target affects only the current boot.

→ **Determine the current target:**

```
#systemctl get-default
```

→ **Change to a different target unit in the current boot:**

```
#systemctl isolate <name>.target

#systemctl isolate multi-user.target
```