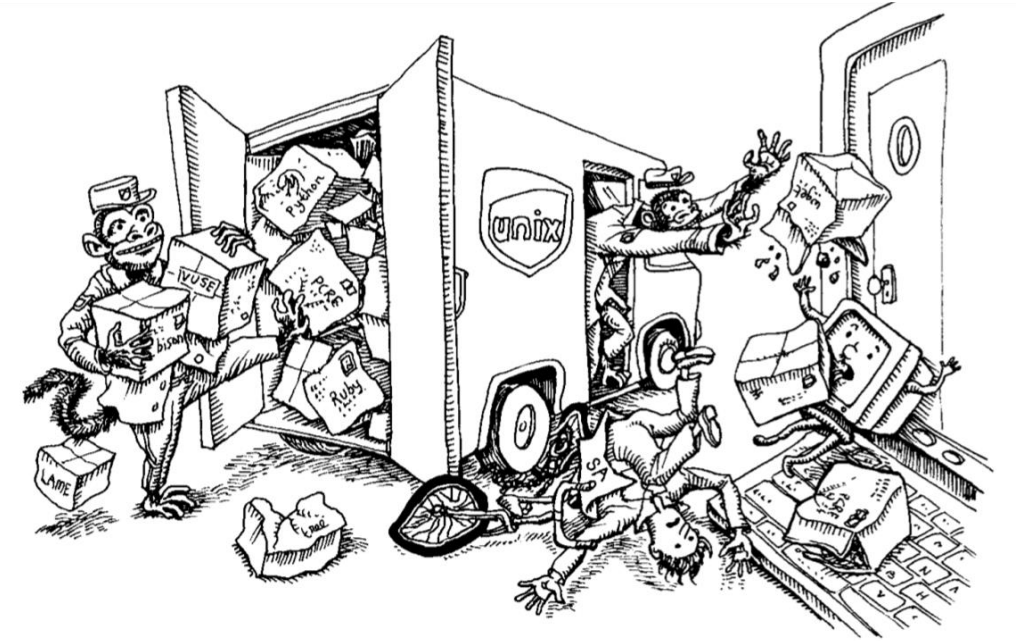
# Software Installation and Management



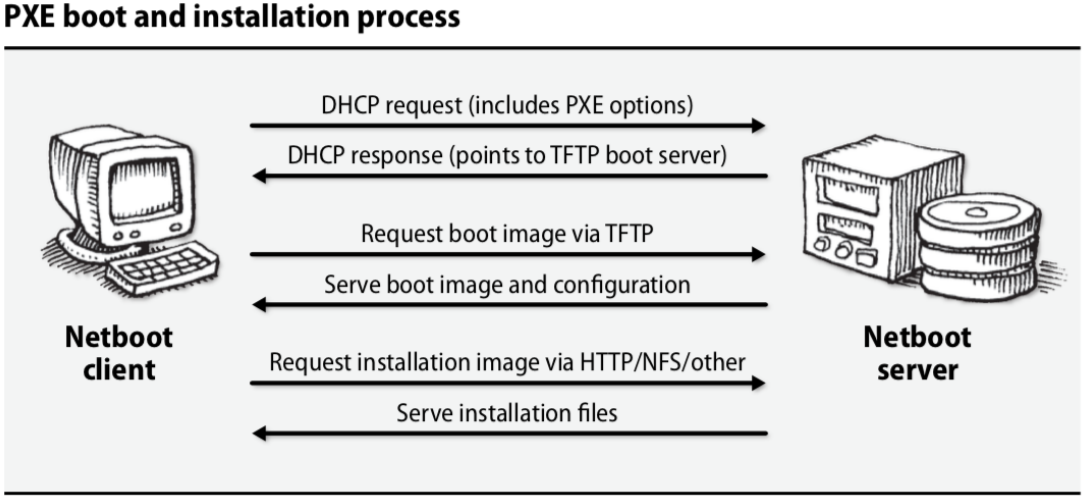
# Software Management at All Levels

- Large parts of sysadmins jobs are doing the following:
  - Automating mass installation of operating systems
  - Maintaining custom OS configurations (“localization”)
  - Keeping systems and applications patched and up to date
  - Tracking software licenses
  - Managing add-on software packages
- With fleets of any non-trivial size these tasks need to be highly automated and (when possible, which is almost always these days) done remotely
- These types of network drive installation typically use DHCP and TFTP to boot the system, then receive the OS over the network through HTTP, NFS, or FTP

# OS Installation Management

# Preboot eXecution Environment (PXE)

- Miniature OS that sits on network card (ROM) for PXE enabled machines
- Coordinates between the network card and the BIOS to allow for boot images to be requested through the network and installed automatically
- It then allows you to choose (or automate) an OS install from a server



# Kickstart for Red Hat and CentOS

- Regardless of the boot locations, an installation can be automated
    - The user does not to sit and select the installation options
- The configurations can be provided in a file
- The file can be placed in a removable media, or on the network
- At the boot time, the configuration can be provided using boot options

# Kickstart for Red Hat and CentOS

- Runs based on a preconfiguration file with three basic sections:
  - Command section, which specifies options such as language, keyboard, and time zone as well as source distribution (http here), root password, and hard drive partitions
  - Packages section, where packages and sets of packages that should be installed after the OS are listed
  - Pre/Post commands section, which allows you to specify arbitrary shell commands which should run `%pre` or `%post` installation

```
text
lang en_US                      # lang is used during the installation...
langsupport en_US               #   ...and langsupport at run time
keyboard us                     # Use an American keyboard
timezone --utc America/EST      # --utc means hardware clock is on GMT
mouse
rootpw --iscrypted $6$NaCl$X5jRlREy9DqNTCXjHp075/
reboot                          # Reboot after installation. Always wise.
bootloader --location=mbr       # Install default boot loader in the MBR
install                         # Install a new system, don't upgrade
url --url http://installserver/redhat
clearpart --all --initlabel     # Clear all existing partitions
part / --fstype ext3 --size 4096
part swap --size 1024
part /var --fstype ext3 -size 1 --grow
network --bootproto dhcp
auth --useshadow --enablemd5
firewall --disabled
xconfig --defaultdesktop=GNOME --startxonboot --resolution 1280x1024
    --depth 24
```

# Netbooting with Cobbler

- Cobbler is an open source netbooting service (written by Michael DeHaan)
- It bundles DHCP, DNS and TFTP services together and helps manage OS images for physical and virtual machines
- Cobbler supports templates through the use of snippets which allow you to create a semi-standard template file which then references snippets for differentiation
- Snippets allow you to manage machine variance easier such as time zone differences for geographically disparate machines, web vs. user configuration, and other situations where small variances occur in a manageable way
- Cobbler can also work with a variety of hypervisors to manage OS installations for virtualized systems (VMs and containers)

# Software Packages

# Packages

- Software used to be distributed as compressed archives that then needed to be uncompressed (and usually installed if it was software source)
- Packaging systems have generally replaced this system, and include all the files needed to run a piece of software for a particular machine and OS combination
    - Source files, precompiled binaries, dependency information, and configuration file templates
- Packages can run scripts which means they can add new users and groups, runs sanity checks, and customizes environmental settings
- You can create custom packages for local applications or third party applications that aren’t normally installed, or even your own localizations

# Packages

- Packaging system typically don’t overwrite custom configurations, instead they create a backup when upgrading software or make a template configuration under a different name for admins to modify
    - This doesn’t always work as intended so it’s important to test package upgrades before rolling them out to the full fleet
- Dependency awareness means packages will try and install all the software they depend on before trying to install themselves
    - This also can be incorrect, and lead to “dependency hell” where you cannot install a package because of version incompatibilities
- Dependencies can also be utilize to create package groups, which are really just dependency lists that when installed install other sets of software

# Low Level Package Management

- The command `rpm` installs, verifies, and queries the status of packages while `rpmbuild` actually does the building of them from source
- Primary flags to know for `rpm` are
  - `-i` (install)
  - `-U` (upgrade)
  - `-e` (erase)
  - `-q` (query)
- The `rpm` system is aware of dependencies, but will not automatically handle the upgrading of necessary dependencies unless all are selected for upgrade at the same time (`rpm` does understand wildcard * character though)

# High Level Package Management

- Metapackage management systems add several features that traditional package managers do not normally perform:
  - Simplify the task of locating and downloading packages
  - Automate the process of updating or upgrading systems
  - Manage the dependencies between multiple packages
- In order for software to be distributed widely through a package metapackage manager it must have developer side tools to describe the software, its installation, its dependencies, and all other necessary extra steps
- It also has a mechanism for describing where the packages can be acquired from to make querying a much simpler task, which can be utilized by sysadmins to separate internally approved packages from general ones

# Package Repositories

- Web of FTP server maintained by Linux distributors (or others), default configurations typically point to distributors own repository
- Some common and useful terms for packages:
  - **Release**: a self-consistent snapshot of all the packages in the distribution
  - **Component**: a subset of software within a release. Distributors partition their releases differently but common ways are core vs. extra software and open-source vs. restricted
  - **Architecture**: the class of hardware that the package is meant for (x86_63, x86_32, etc.)
  - **Package**: architecture-specific independently versioned software, a collection of which makes up a release

# Package Managers

- The two major package management systems are the Advanced Package Tool (`apt`) and the Yellowdog Updater, Modified (`yum`)
- Some important command that are (more or less) shared between the two managers:
  - `apt/yum install <package-name>`: installs the <package-name> assuming it exists in one of the registered repositories
  - `apt update`: updates the cached list of packages (including new versions) from the remote repositories
  - `apt upgrade/yum update`: updates all packages previously installed to the newest version

# Repository Configuration

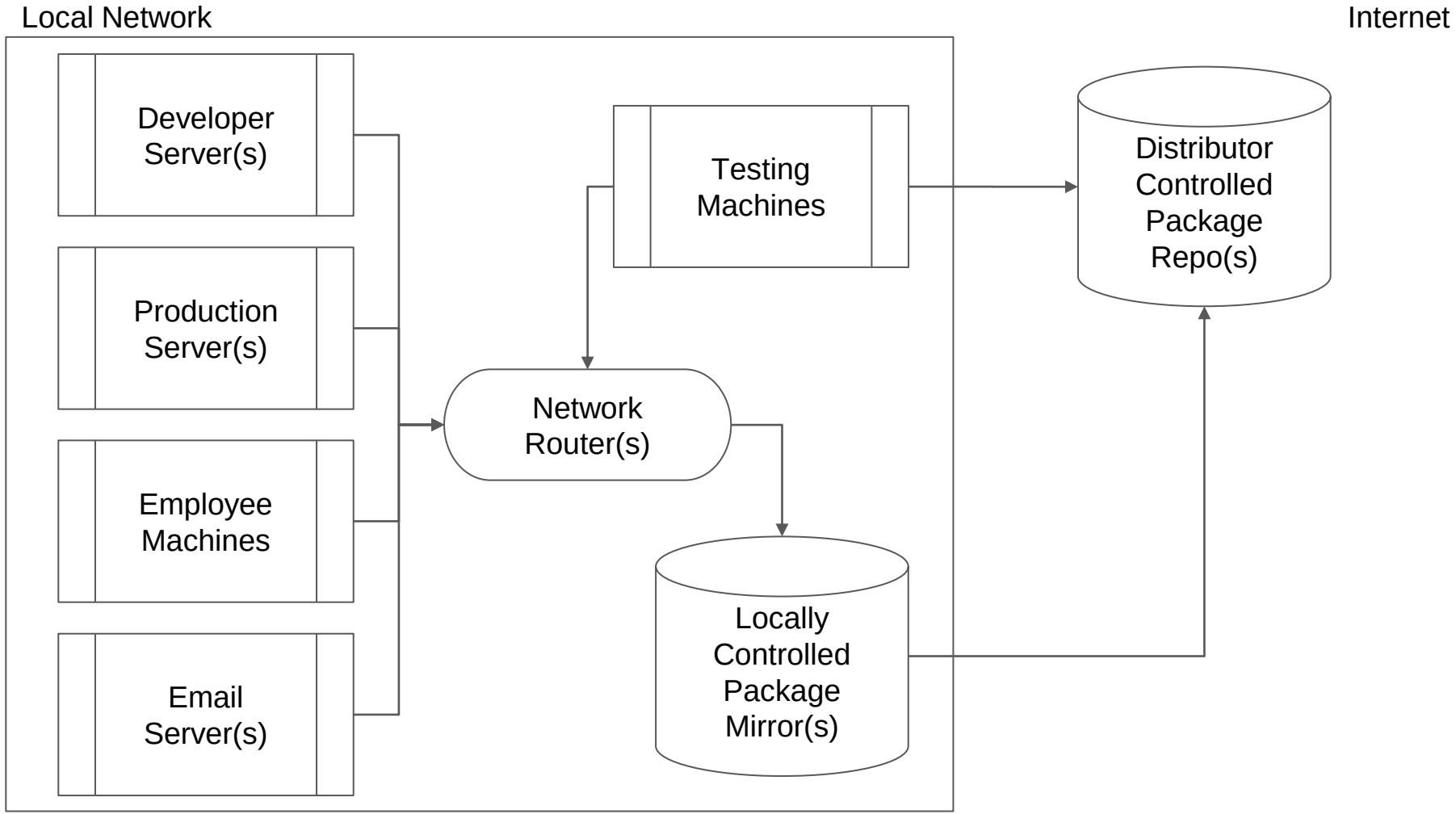
- The `apt` repositories are configured at `/etc/apt/sources.list`
  - Formatted: `type mirror-uri distribution [components]`
  - Good practice to change the `mirror-uri` to the closest mirror
  - Replace `mirror-uri` when using a local mirror rather than “remote”
  - Typically you add additional “nice to have” software repos to the list
- The `yum` repositories are configured at `/etc/yum.conf` and `/etc/yum.repos.d/*.repo`
  - Additional repositories should be added as `*.repo` files
  - Recommended organization is one `*.repo` file per “nice to have” repo
  - Modify `baseurl` for repositories (there is also `yum-config-manager`)

# Package Automation

- Neither apt nor yum have built-in scheduling, but you can utilize the manager and cron or systemd timers to accomplish automated updating
  - `apt update && apt upgrade -y`
  - `yum update -y`
- It is a **bad idea** to automatically update packages from distributor repositories since changes to the system may affect other software
- This is especially true about development or production servers which are running custom written code on top of generic services
- Rather than updating automatically from a distributor repository, use a locally controlled mirror repository which is only updated after (extensively) testing new software with your software and services

# Local Repository Mirrors

- Create your own FTP or HTTP(S) server which contains a subset of the available packages (usually based on version)
- Use `apt-mirror` (non-standard, available on GitHub) to create an `apt` repository or re-create the directory structure used for a `yum` repository
- Make the necessary directories available on your local network either through any web server or through an FTP server
- This allows for faster fleet updates since you update the mirror over the internet but the fleet over the local network
- You can test package updates before updating them on the mirror, meaning package upgrade automation is less likely to break systems


Local Network
Internet
Developer Server(s)
Production Server(s)
Employee Machines
Email Server(s)
Testing Machines
Network Router(s)
Locally Controlled Package Mirror(s)
Distributor Controlled Package Repo(s)

# Update Tips

- Updates that cause changes to “user visible” software should be communicated to users in advance and you should solicit feedback about the possible change to see how it will affect users
- Don’t roll out software updates en masse if you have a large fleet of machines, instead use either a “canary” deployment or a “hockey stick” deployment (depending on the number of machines)
- Try and make small updates regularly rather than saving up a bunch of updates and patches for a single update, regular and gradual updates tend to have fewer issues than large and rare updates
- Unless you enjoy being frustrated over the weekend, don’t roll out software updates on Fridays

Questions?

# Additional Resources

Configuring PXE Boot For CentOS 6.5 Using FTP, HTTP, And Kickstart

RedHat Yum Cheat Sheet