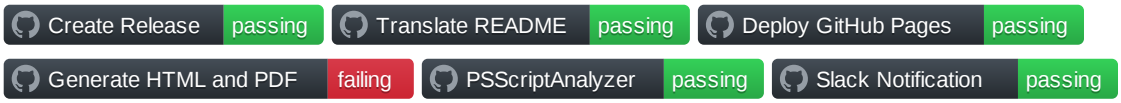
 Light



LEARNING LPIC-3 305-300

Explore the docs »

Web Site - Code Page - Report Bug - Request Feature

Summary

About Project

This project aims to help students or professionals to learn the main concepts of GNULinux and free software

Light

Some GNULinux distributions like Debian and RPM will be covered
Installation and configuration of some packages will also be covered
By doing this you can give the whole community a chance to benefit from your changes.
Access to the source code is a precondition for this.
Use vagrant for up machines and execute labs and practice content in this article.
I have published in folder Vagrant a Vagrantfile with what is necessary
for you to upload an environment for studies

(back to top)

Getting Started

For starting the learning, see the documentation above.

Prerequisites

- Git

Installation

Clone the repo

```
git clone https://github.com/marcossilvestrini/learning-lpic-3-305-300.git
```

Usage

Use this repository for get learning about LPIC2 202-450 exam

(back to top)

Roadmap

- [x] Create repository
- [ ] Create examples about Topic 351
- [ ] Create examples about Topic 352
- [ ] Create examples about Topic 353
- [ ] Upload simulated itexam

Four Essential Freedoms

0.The freedom to run the program as you wish, for any purpose (freedom 0).
1.The freedom to study how the program works, and change it so it does your computing as you wish (freedom 1).
Access to the source code is a precondition for this.
2.The freedom to redistribute copies so you can help others (freedom 2).
3.freedom to distribute copies of your modified versions to others (freedom 3).

Inspect commands

```
type COMMAND
apropos COMMAND
whatis COMMAND --long
whereis COMMAND
COMMAND --help, --h
man COMMAND
```

(back to top)

Topic 351: Full Virtualization

The difference between type-1 and type-2 hypervisors

351.1 Virtualization Concepts and Theory

Weight: 6

Description: Candidates should know and understand the general concepts, theory and terminology of virtualization. This includes Xen, QEMU and libvirt terminology.

Key Knowledge Areas:

- Understand virtualization terminology
- Understand the pros and cons of virtualization
- Understand the various variations of Hypervisors and Virtual Machine Monitors
- Understand the major aspects of migrating physical to virtual machines
- Understand the major aspects of migrating virtual machines between host systems
- Understand the features and implications of virtualization for a virtual machine, such as snapshotting, pausing, cloning and resource limits
- Awareness of oVirt, Proxmox, systemd-machined and VirtualBox
- Awareness of Open vSwitch

351.1 Cited Objects

```
Hypervisor
Hardware Virtual Machine (HVM)
Paravirtualization (PV)
Emulation and Simulation
CPU flags
/proc/cpuinfo
Migration (P2V, V2V)
```

Hypervisors

Type 1 Hypervisor (Bare-Metal Hypervisor)

Definition: Runs directly on the host's physical hardware, providing a base layer to manage VMs without the need for a host operating system.

Characteristics:

- High performance and efficiency.
- Lower latency and overhead.
- Often used in enterprise environments and data centers.

Examples:

- VMware ESXi: A robust and widely used hypervisor in enterprise settings.
- Microsoft Hyper-V: Integrated with Windows Server, offering strong performance and management features.
- Xen: An open-source hypervisor used by many cloud service providers.

- KVM (Kernel-based Virtual Machine): Integrated into the Linux kernel, providing high performance for Linux-based systems.

Type 2 Hypervisor (Hosted Hypervisor)

Definition: Runs on top of a conventional operating system, relying on the host OS for resource management and device support.

Characteristics:

- Easier to set up and use, especially on personal computers.
- More flexible for development, testing, and smaller-scale deployments.
- Typically less efficient than Type 1 hypervisors due to additional overhead from the host OS.

Examples:

- VMware Workstation: A powerful hypervisor for running multiple operating systems on a single desktop.
- Oracle VirtualBox: An open-source hypervisor known for its flexibility and ease of use.
- Parallels Desktop: Designed for Mac users to run Windows and other operating systems alongside macOS.
- QEMU (Quick EMUlator): An open-source emulator and virtualizer, often used in conjunction with KVM.

Key Differences Between Type 1 and Type 2 Hypervisors

- Deployment Environment:
  - Type 1 hypervisors are commonly deployed in data centers and enterprise environments due to their direct interaction with hardware and high performance.
  - Type 2 hypervisors are more suitable for personal use, development, testing, and small-scale virtualization tasks.
- Performance:
  - Type 1 hypervisors generally offer better performance and lower latency because they do not rely on a host OS.
  - Type 2 hypervisors may experience some performance degradation due to the overhead of running on top of a host OS.
- Management and Ease of Use:
  - Type 1 hypervisors require more complex setup and management but provide advanced features and scalability for large-scale deployments.
  - Type 2 hypervisors are easier to install and use, making them ideal for individual users and smaller projects.

HVM and Paravirtualization

Hardware-assisted Virtualization (HVM)

Definition: HVM leverages hardware extensions provided by modern CPUs to virtualize hardware, enabling the creation and management of VMs with minimal performance overhead.

Key Characteristics:

- **Hardware Support**: Requires CPU support for virtualization extensions such as Intel VT-x or AMD-V.
- **Full Virtualization:** VMs can run unmodified guest operating systems, as the hypervisor provides a complete emulation of the hardware environment.
- **Performance:** Typically offers near-native performance because of direct execution of guest code on the CPU.
- **Isolation:** Provides strong isolation between VMs since each VM operates as if it has its own dedicated hardware.

Examples: VMware ESXi, Microsoft Hyper-V, KVM (Kernel-based Virtual Machine).

Advantages:

- **Compatibility:** Can run any operating system without modification.
- **Performance:** High performance due to hardware support.
- **Security:** Enhanced isolation and security features provided by hardware.

Disadvantages:

- **Hardware Dependency:** Requires specific hardware features, limiting compatibility with older systems.
- **Complexity:** May involve more complex configuration and management.

Paravirtualization

Definition: Paravirtualization involves modifying the guest operating system to be aware of the virtual environment, allowing it to interact more efficiently with the hypervisor.

Key Characteristics:

- **Guest Modification:** Requires changes to the guest operating system to communicate directly with the hypervisor using hypercalls.
- **Performance:** Can be more efficient than traditional full virtualization because it reduces the overhead associated with emulating hardware.
- **Compatibility:** Limited to operating systems that have been modified for paravirtualization.

Examples: Xen with paravirtualized guests, VMware tools in certain configurations, and some KVM configurations.

Advantages:

- **Efficiency:** Reduces the overhead of virtualizing hardware, potentially offering better performance for certain workloads.
- **Resource Utilization:** More efficient use of system resources due to direct communication between the guest OS and hypervisor.

Disadvantages:

- **Guest OS Modification:** Requires modifications to the guest OS, limiting compatibility to supported operating systems.
- **Complexity:** Requires additional complexity in the guest OS for hypercall implementations.

Key Differences

Guest OS Requirements:

- **HVM:** Can run unmodified guest operating systems.
- **Paravirtualization:** Requires guest operating systems to be modified to work with the hypervisor.

Performance:

- **HVM:** Typically provides near-native performance due to hardware-assisted execution.
- **Paravirtualization:** Can offer efficient performance by reducing the overhead of hardware emulation, but relies on modified guest OS.

Hardware Dependency:

- **HVM:** Requires specific CPU features (Intel VT-x, AMD-V).
- **Paravirtualization:** Does not require specific CPU features but needs modified guest OS.

Isolation:

- **HVM:** Provides strong isolation using hardware features.
- **Paravirtualization:** Relies on software-based isolation, which may not be as robust as hardware-based isolation.

Complexity:

- **HVM:** Generally more straightforward to deploy since it supports unmodified OS.
- **Paravirtualization:** Requires additional setup and modifications to the guest OS, increasing complexity.

Types of Virtualization

Hardware Virtualization (Server Virtualization)

Definition: Abstracts physical hardware to create virtual machines (VMs) that run separate operating systems and applications.
Use Cases: Data centers, cloud computing, server consolidation. **Examples:** VMware ESXi, Microsoft Hyper-V, KVM.

Operating System Virtualization (Containerization)

Definition: Allows multiple isolated user-space instances (containers) to run on a single OS kernel. **Use Cases:** Microservices architecture, development and testing environments. **Examples:** Docker, Kubernetes, LXC.

Network Virtualization

Definition: Combines hardware and software network resources into a single, software-based administrative entity. **Use Cases:** Software-defined networking (SDN), network function virtualization (NFV). **Examples:** VMware NSX, Cisco ACI, OpenStack Neutron.

Storage Virtualization

Definition: Pools physical storage from multiple devices into a single virtual storage unit that can be managed centrally. **Use Cases:** Data management, storage optimization, disaster recovery. **Examples:** IBM SAN Volume Controller, VMware vSAN, NetApp ONTAP.

Desktop Virtualization

Definition: Allows a desktop operating system to run on a virtual machine hosted on a server. **Use Cases:** Virtual desktop infrastructure (VDI), remote work solutions. **Examples:** Citrix Virtual Apps and Desktops, VMware Horizon, Microsoft Remote Desktop Services.

Application Virtualization

Definition: Separates applications from the underlying hardware and operating system, allowing them to run in isolated environments. **Use Cases:** Simplified application deployment, compatibility testing. **Examples:** VMware ThinApp, Microsoft App-V, Citrix XenApp.

Data Virtualization

Definition: Integrates data from various sources without physically consolidating it, providing a unified view for analysis and reporting. **Use Cases:** Business intelligence, real-time data integration. **Examples:** Denodo, Red Hat JBoss Data Virtualization, IBM InfoSphere.

Benefits of Virtualization

Resource Efficiency: Better utilization of physical resources. **Cost Savings:** Reduced hardware and operational costs. **Scalability:** Easy to scale up or down according to demand. **Flexibility:** Supports a variety of workloads and applications. **Disaster Recovery:** Simplified backup and recovery processes. **Isolation:** Improved security through isolation of environments.

351.2 Xen

Weight: 3

Description: Candidates should be able to install, configure, maintain, migrate and troubleshoot Xen installations. The focus is on Xen version 4.x.

Key Knowledge Areas:

- Understand architecture of Xen, including networking and storage
- Basic configuration of Xen nodes and domains
- Basic management of Xen nodes and domains
- Basic troubleshooting of Xen installations
- Awareness of XAPI
- Awareness of XenStore
- Awareness of Xen Boot Parameters
- Awareness of the xm utility

351.2 Cited Objects

```
Domain0 (Dom0), DomainU (DomU)
PV-DomU, HVM-DomU
/etc/xen/
xl
xl.cfg
xl.conf
xentop
```

351.2 Important Commands

foo

```
foo
```

351.3 QEMU

Weight: 4

Description: Candidates should be able to install, configure, maintain, migrate and troubleshoot QEMU installations.

Key Knowledge Areas:

- Understand the architecture of QEMU, including KVM, networking and storage
- Start QEMU instances from the command line
- Manage snapshots using the QEMU monitor
- Install the QEMU Guest Agent and VirtIO device drivers
- Troubleshoot QEMU installations, including networking and storage
- Awareness of important QEMU configuration parameters

351.3 Cited Objects

```
Kernel modules: kvm, kvm-intel and kvm-amd
/dev/kvm
QEMU monitor
qemu
qemu-system-x86_64
ip
brctl
tunctl
```

351.3 Important Commands

ip

```
# list links
ip link show
```

(back to sub Topic 351.3)

(back to Topic 351)

(back to top)

351.4 Libvirt Virtual Machine Management

Light

Weight: 9

Description: Candidates should be able to manage virtualization hosts and virtual machines (‘libvirt domains’) using libvirt and related tools.

Key Knowledge Areas:

- Understand the architecture of libvirt
- Manage libvirt connections and nodes
- Create and manage QEMU and Xen domains, including snapshots
- Manage and analyze resource consumption of domains
- Create and manage storage pools and volumes
- Create and manage virtual networks
- Migrate domains between nodes
- Understand how libvirt interacts with Xen and QEMU
- Understand how libvirt interacts with network services such as dnsmasq and radvd
- Understand libvirt XML configuration files
- Awareness of virtlogd and virtlockd

351.4 Cited Objects

```
libvirtd
/etc/libvirt/
virsh (including relevant subcommands)
```

351.4 Important Commands

foo

```
foo
```

(back to sub Topic 351.4)

(back to Topic 351)

(back to top)

351.5 Virtual Machine Disk Image Management

Weight: 3

Description: Candidates should be able to manage virtual machines disk images. This inc converting disk images between various formats and hypervisors and accessing data stored within an image.

Key Knowledge Areas:

- Understand features of various virtual disk image formats, such as raw images, qcow2 and VMDK
- Manage virtual machine disk images using qemu-img
- Mount partitions and access files contained in virtual machine disk images using libguestfish
- Copy physical disk content to a virtual machine disk image
- Migrate disk content between various virtual machine disk image formats
- Awareness of Open Virtualization Format (OVF)

351.5 Cited Objects

```
qemu-img
guestfish (including relevant subcommands)
guestmount
guestumount
virt-cat
virt-copy-in
virt-copy-out
virt-diff
virt-inspector
virt-filesystems
virt-rescue
virt-df
virt-resize
virt-sparsify
virt-p2v
virt-p2v-make-disk
virt-v2v
virt-sysprep
```

351.5 Important Commands

foo

```
foo
```

(back to sub Topic 351.5)

(back to Topic 351)

(back to top)

 Light

Topic 352: Container Virtualization

352.1 Container Virtualization Concepts

Weight: 7

Description: Candidates should understand the concept of container virtualization. This includes understanding the Linux components used to implement container virtualization as well as using standard Linux tools to troubleshoot these components.

Key Knowledge Areas:

- Understand the concepts of system and application container
- Understand and analyze kernel namespaces
- Understand and analyze control groups
- Understand and analyze capabilities
- Understand the role of seccomp, SELinux and AppArmor for container virtualization
- Understand how LXC and Docker leverage namespaces, cgroups, capabilities, seccomp and MAC
- Understand the principle of runc
- Understand the principle of CRI-O and containerd
- Awareness of the OCI runtime and image specifications
- Awareness of the Kubernetes Container Runtime Interface (CRI)
- Awareness of podman, buildah and skopeo
- Awareness of other container virtualization approaches in Linux and other free operating systems, such as rkt, OpenVZ, systemd-nspawn or BSD Jails

352.1 Cited Objects

```
nsenter
unshare
ip (including relevant subcommands)
capsh
/sys/fs/cgroups
/proc/[0-9]+/ns
/proc/[0-9]+/status
```

352.1 Important Commands

foo

(back to sub topic 352.1)

(back to topic 352)

(back to top)

352.2 LXC

Weight: 6

Description: Candidates should be able to use system containers using LXC and LXD. The version of LXC covered is 3.0 or higher.

Key Knowledge Areas:

- Understand the architecture of LXC and LXD
- Manage LXC containers based on existing images using LXD, including networking and storage
- Configure LXC container properties
- Limit LXC container resource usage
- Use LXD profiles
- Understand LXC images
- Awareness of traditional LXC tools

352.2 Cited Objects

```
lxd
lxc (including relevant subcommands)
```

352.2 Important Commands

foo

```
foo
```

(back to sub topic 352.2)

(back to topic 352)

(back to top)

 Light

352.3 Docker

Weight: 9

Description: Candidate should be able to manage Docker nodes and Docker containers. This include understand the architecture of Docker as well as understanding how Docker interacts with the node's Linux system.

Key Knowledge Areas:

- Understand the architecture and components of Docker
- Manage Docker containers by using images from a Docker registry
- Understand and manage images and volumes for Docker containers
- Understand and manage logging for Docker containers
- Understand and manage networking for Docker
- Use Dockerfiles to create container images
- Run a Docker registry using the registry Docker image

352.3 Cited Objects

```
dockerd
/etc/docker/daemon.json
/var/lib/docker/
docker
Dockerfile
```

352.3 Important Commands

docker

```
# Examples of docker
```

(back to sub topic 352.3)

(back to topic 352)

(back to top)

352.4 Container Orchestration Platforms

Weight: 3

Description: Candidates should understand the importance of container orchestration and key concepts Docker Swarm and Kubernetes provide to implement container orchestration.

Key Knowledge Areas:

- Understand the relevance of container orchestration
- Understand the key concepts of Docker Compose and Docker Swarm
- Understand the key concepts of Kubernetes and Helm
- Awareness of OpenShift, Rancher and Mesosphere DC/OS

(back to sub topic 352.4)

(back to topic 352)

(back to top)

Topic 353: VM Deployment and Provisioning

353.1 Cloud Management Tools

Weight: 2

Description: Candidates should understand common offerings in public clouds and have basic feature knowledge of commonly available cloud management tools.

Key Knowledge Areas:

- Understand common offerings in public clouds
- Basic feature knowledge of OpenStack
- Basic feature knowledge of Terraform
- Awareness of CloudStack, Eucalyptus and OpenNebula

353.1 Cited Objects

```
IaaS, PaaS, SaaS
OpenStack
Terraform
```

353.1 Important Commands

foo

```
# examples
```

(back to sub topic 353.1)

(back to topic 353)

(back to top)

353.2 Packer

Weight: 2

Description: Candidates should be able to use Packer to create system images. This includes running Packer in various public and private cloud environments as well as building container images for LXC/LXD.

Key Knowledge Areas:

- Understand the functionality and features of Packer
- Create and maintain template files
- Build images from template files using different builders

353.2 Cited Objects

```
packer
```

353.2 Important Commands

packer

```
# examples
```

(back to sub topic 353.2)

(back to topic 353)

(back to top)

353.3 cloud-init

Weight: 3

Description: Candidates should able to use cloud-init to configure virtual machines created from standardized images. This includes adjusting virtual machines to match their available hardware resources, specifically, disk space and volumes.
Additionally, candidates should be able to configure instances to allow secure SSH logins and install a specific set of software packages.
Furthermore, candidates should be able to create new system images with cloud-init support.

Key Knowledge Areas:

- Understanding the features and concepts of cloud-init, including user-data, initializing and configuring cloud-init
- Use cloud-init to create, resize and mount file systems, configure user accounts, including login credentials such as SSH keys and install software packages from the distribution's repository
- Integrate cloud-init into system images
- Use config drive datasource for testing

353.3 Cited Objects

```
cloud-init
user-data
/var/lib/cloud/
```

353.3 Important Commands

foo

```
# examples
```

(back to sub topic 353.3)

(back to topic 353)

(back to top)

353.4 Vagrant

Weight: 3

Description: Candidate should be able to use Vagrant to manage virtual machines, including provisioning of the virtual machine.

Key Knowledge Areas:

- Understand Vagrant architecture and concepts, including storage and networking
- Retrieve and use boxes from Atlas
- Create and run Vagrantfiles
- Access Vagrant virtual machines
- Share and synchronize folder between a Vagrant virtual machine and the host system
- Understand Vagrant provisioning, i.e. File and Shell provisioners
- Understand multi-machine setup

353.4 Cited Objects

```
vagrant
Vagrantfile
```

353.4 Important Commands

vagrant

```
# examples
```

(back to sub topic 353.4)

(back to topic 353)

(back to top)

Contributing

Contributions are what make the open source community such an amazing place to learn, inspire, and create. Any contributions you make are **greatly appreciated**.

If you have a suggestion that would make this better, please fork the repo and create a pull request. You can also simply open an issue with the tag "enhancement". Don't forget to give the project a star! Thanks again!

1. Fork the Project
2. Create your Feature Branch (`git checkout -b feature/AmazingFeature`)
3. Commit your Changes (`git commit -m 'Add some AmazingFeature'`)
4. Push to the Branch (`git push origin feature/AmazingFeature`)
5. Open a Pull Request

License

- This project is licensed under the MIT License * see the LICENSE.md file for details

Contact

Marcos Silvestrini - marcos.silvestrini@gmail.com

Follow @mrsilvestrini

Project Link: https://github.com/marcossilvestrini/learning-lpic-3-305-300

(back to top)

Acknowledgments

- Richard Stallman's
- GNU/Linux FAQ by Richard Stallman
- GNU
- GNU Operating System
- GCC Compiler
- GNU Tar
- GNU Make
- GNU Emacs
- GNU Packages
- GNU/Linux Collection
- GNU Grub Bootloader
- GNU Hurd
- Kernel
- Linux Kernel Man Pages
- Linux Standard Base
- Filesystem Hierarchy Standard
- File Hierarchy Structure
- FSF
- Free Software Directory
- Free Software
- Copyleft
- GPL

Light

- GNU Lesser General Public License
- BSD
- Open Source Initiative
- Creative Commons
- License LTS
- Debian Free Software Guidelines
- X11 Org
- Wayland
- GNU GNOME
- GNOME
- XFCE
- KDE Plasma
- Harmony
- xRDP
- NTP
- Bourne Again Shell
- Shebang
- Environment Variables
- GNU Globbing
- Globbing
- Quoting
- Regular Expressions
- List Linux Distribution
- Distro Watch
- Comparison Linux Distributions
- Download Packages
- Install Packages
- Guide Install Packages
- Bugzila
- Command Not Found
- DistroTest
- Bash RC Generator
- Explainshell
- Vim Tutorial
- Linux Shell Scripting Tutorial
- Github Badges
- Commands Examples

- Compile Your Kernel
- Bind
- Bind Logging
- List of DNS record types
- List of DNS record types
- W3Techs
- Apache
- Apache Directives
- HTTP Status Codes
- Strong Ciphers for Apache, nginx and Lighttpd
- SSL Tutorials
- SSL Config Mozilla
- Virtualization Definitions
  - Red Hat
  - AWS
  - IBM
  - OpenSource.com
- KVM(Kernel Virtual Machines)
- KVM Management Tools
- Wiki XenProject
- LPI Blog: Xen Virtualization and Cloud Computing #01: Introduction
- Openstack Docs
  - RedHat
- LPIC-3 305-300 Objectives
- LPIC-3 305-300 Wiki
- LPIC-3 305-300 Learning Material
- LPIC-3 305-300 Simulated Exam By ITexams

(back to top)