DUMPSBOSS.

LPIC-3: Virtualization and Containerization

LPI 305-300

Version Demo

Total Demo Questions: 10

Total Premium Questions: 103

Buy Premium PDF

https://dumpsboss.co

support@dumpsboss.co

support@dumpsboss.co
dumpsboss.co

QUESTION NO: 1

Which of the following directives is used in the configuration file of a Xen guest domain in order to define network interfaces?

A. vif

B. eth

C. vnet

D. vbr

E. net

ANSWER: A

Explanation:

In a Xen domain (guest) configuration file (xl/xm style), network interfaces are defined using the `vif` directive. This directive specifies one or more virtual interfaces and their parameters, typically as a list of strings, e.g. `vif = [ 'bridge=xenbr0,mac=00:16:3e:xx:xx:xx' ]`. Xen then creates a corresponding virtual interface (vif) and connects it to the chosen backend (often a Linux bridge or Open vSwitch) on the host. This is the standard and documented way to attach networking to a Xen guest via its domain config.

The other options are not valid Xen domain configuration directives: `eth` is a common Linux interface naming prefix but not a Xen config key; `vnet` and `vbr` are not standard xl/xm directives (you might see “xenbr0” as a bridge name, but that’s a value used inside `vif`, not a directive itself); and `net` is not the directive used in Xen domain configs for interface definitions.

References: Xen xl.cfg(5) manual, Xen Project Network Configuration Examples.

QUESTION NO: 2

Which of the following mechanisms are used by LXC and Docker to create containers? (Choose three.)

A. Linux Capabilities

B. Kernel Namespaces

C. Control Groups

D. POSIXACLs

E. File System Permissions

ANSWER: A B C

Explanation:

LXC and Docker both rely on core Linux kernel isolation and resource-governance features to implement containers. The primary mechanisms are **kernel namespaces** (option B) and **control groups (cgroups)** (option C). Namespaces provide isolation of global system resources so a container can have its own process tree (PID namespace), networking stack (net namespace), mount points (mount namespace), hostname (UTS), IPC objects, and user/group mappings (user namespace). Cgroups provide accounting and limits for CPU, memory, I/O, and PIDs, which is essential for preventing one container from exhausting host resources.

Linux capabilities (option A) are also commonly used by container runtimes to reduce the privileges of processes inside containers (dropping capabilities like CAP_SYS_ADMIN, etc.). While capabilities are not the only security mechanism involved, they are a standard part of how Docker and LXC implement least privilege and are widely referenced in container security models.

POSIX ACLs (option D) and generic **file system permissions** (option E) are normal Unix authorization features, but they are not the defining kernel mechanisms used to create container isolation. They may be used inside containers like any Linux system, yet they don't provide the container boundary itself (that's namespaces/cgroups/capabilities).

References: man7: namespaces(7), Docker Engine security

QUESTION NO: 3

When KVM is launched with the parameter -boot order=n, which of the following devices will be searched for a bootable operating system?

A. All floppy, CDROM, and hard disk drives in that order.

B. No devices are searched and the user is prompted to choose the boot device.

C. All network interfaces attached to the VM.

D. All hard disks and no other devices.

E. No devices are searched in order to support directly booting a Linux kernel.

ANSWER: C

Explanation:

In QEMU/KVM, the `-boot order=` parameter defines the BIOS/firmware boot device priority using single-letter codes. The letter `n` specifically means "boot from network" (PXE). Therefore, when KVM is started with `-boot order=n`, the guest firmware will attempt to boot via the VM's network boot ROM, trying the available virtual NICs for a PXE-capable boot path.

Option C is correct because it matches this behavior: the VM will search the attached network interface(s) for a bootable OS via network boot.

Option A is wrong because that describes a multi-device boot order like `order=fdc` (floppy, disk, CD-ROM) rather than `n` alone. Option B is wrong because `-boot` does not inherently "prompt the user"; it sets a deterministic order (interactive boot menus are a separate firmware feature). Option D is wrong because hard disks correspond to `c`, not `n`. Option E is wrong because direct kernel boot is done with options like `-kernel`/`-initrd`/`-append`, not by setting `order=n`.

References: QEMU System Emulator invocation (boot options), qemu-system-x86_64 man page.

QUESTION NO: 4

Which of the following statements are true regarding OCFS2? (Choose TWO correct answers.)

A. OCFS2 is an integral part of Pacemaker and relies on Pacemaker services for functionality.

B. To avoid the need for shared storage, OCFS2 can replicate the content of its filesystems via the network.

C. When using OCFS2 with additional cluster software, OCFS2 must be integrated into the overall cluster manager to ensure consistent cluster behaviour.

D. OCFS2 can be used without any additional software as it contains its own cluster manager, O2CB.

E. In addition to filesystems, OCFS2 can handle other cluster services such as IP addresses and server daemons.

ANSWER: C D

Explanation:

OCFS2 (Oracle Cluster File System 2) is a shared-disk cluster filesystem: multiple nodes mount the same block device concurrently, and OCFS2 coordinates access using a cluster stack. A key point is that OCFS2 ships with its own cluster infrastructure called O2CB (the “OCFS2 Cluster Base”), which provides node membership/heartbeat and the distributed lock manager needed by the filesystem. Therefore, OCFS2 can be deployed without Pacemaker/Corosync if you only need clustered filesystem semantics, making option D true.

In real HA deployments, you often run OCFS2 alongside a cluster manager (e.g., Pacemaker/Corosync) that controls resources and fencing. In that case, you must ensure OCFS2 is properly integrated into the overall cluster behavior (ordering, membership expectations, fencing) so that mounts and lockspace are consistent with the cluster’s view of node health—this makes option C true.

Option A is false because OCFS2 is not an integral part of Pacemaker and does not rely on Pacemaker to function. Option B is false because OCFS2 does not replicate data over the network; it requires shared storage. Option E is false because OCFS2 is a filesystem, not a general cluster resource manager for IPs/daemons.

References: Oracle Linux: About OCFS2, man7.org: o2cb(5)

QUESTION NO: 5

Which of the following are true regarding the CPU of a QEMU virtual machine? (Choose two.)

A. The CPU architecture of a QEMU virtual machine is independent of the host system's architecture.

B. Each QEMU virtual machine can only have one CPU with one core.

C. For each QEMU virtual machine, one dedicated physical CPU core must be reserved.

D. QEMU uses the concept of virtual CPUs to map the virtual machines to physical CPUs.

E. QEMU virtual machines support multiple virtual CPUs in order to run SMP systems.

ANSWER: D E

Explanation:

Correct answers are D and E. QEMU models guest CPU execution using *virtual CPUs* (vCPUs): each vCPU is a thread from the host's perspective and the host scheduler maps those threads onto available physical CPUs. This is exactly the mechanism that lets multiple guests share the same physical cores without requiring strict one-to-one reservations. In addition, QEMU supports SMP guests by exposing multiple vCPUs to the guest OS (for example via `-smp`), allowing the guest to run as a multiprocessor system.

Option A is not generally true: without full system emulation, QEMU typically relies on KVM hardware virtualization, which requires the guest CPU architecture to match the host (e.g., x86_64 on x86_64). While QEMU *can* emulate different architectures, that is slower and not "independent" in the general sense implied here. Option B is false because QEMU can present multiple CPUs/cores to a VM. Option C is false because vCPUs do not require dedicated physical cores; overcommit is common, and pinning/reservation is optional tuning rather than a requirement.

References: QEMU System Emulator invocation (QEMU docs), KVM FAQ (linux-kvm.org)

QUESTION NO: 6

While setting up a load-balanced HTTP service, Linux Virtual Server was configured with the commands:

ipvsadm -A -t 198.51.100.2:80 -s rr

ipvsadm -a -t 198.51.100.2:80 -r 192.168.10.1:80 -m ipvsadm -a -t 198.51.100.2:80 -r 192.168.10.2:80 -m

and all backend servers are using the LVS host as the default router. Which additional command has to be issued on the LVS host in order to correctly handle incoming HTTP traffic?

A. echo "1" > /proc/sys/net/ipv4/ip_forward

B. iptables -t nat -Z

C. ipvsadmin -L -n

D. ipmasq -A 192.168.10.1 -A 192.168.10.2 -p tcp --dport www -j DEMASQ

E. netstat -r

ANSWER: A

Explanation:

The given `ipvsadm` configuration uses `-m`, which means LVS-NAT (masquerading). In LVS-NAT, the director (LVS host) must forward packets between the public VIP side and the private real-server network. Because the real servers use the LVS host as their default gateway, return traffic will come back to the director and must be routed/forwarded out to the client. Therefore, IPv4 forwarding must be enabled on the LVS host (either via `/proc` or persistently via `sysctl`), otherwise packets will be received but not forwarded and the service will fail.

Option A enables kernel IP forwarding immediately and is the required missing step for an LVS-NAT director. Option B only zeroes NAT table counters and does not change forwarding behavior. Option C (also misspelled as `ipvsadmin`) merely lists IPVS rules and is not a configuration change. Option D refers to old ipmasq tooling/targets and is not how LVS-NAT is set up with modern kernels and IPVS. Option E shows the routing table but does not enable forwarding.

References: Linux kernel IPVS sysctl documentation, ipvsadm(8) manual.

QUESTION NO: 7

Which of the following are tools or services that manage a Linux Virtual Server (LVS) setup? (Choose TWO correct answers.)

A. keepalived

B. ldirectord

C. lvproxy

D. roundrobind

E. vserverd

ANSWER: A B

Explanation:

In an LVS (Linux Virtual Server) environment, the actual load-balancing datapath is implemented in the Linux kernel via IPVS (IP Virtual Server), but administrators typically rely on user-space tools/services to configure IPVS rules and to provide high availability for the “director” node. **keepalived** is a common choice because it can both manage VRRP for failover (via the keepalived daemon) and configure IPVS virtual servers/real servers, making it a standard LVS management component. **ldirectord** (from the Linux-HA/Piranha ecosystem) is another well-known LVS management tool: it monitors real servers with health checks and dynamically updates the IPVS table accordingly, which is exactly “managing an LVS setup.”

The other options are not standard LVS management tools. **lvproxy** and **roundrobind** are not recognized, commonly used LVS/IPVS management utilities in mainstream Linux HA/LVS stacks. **vserverd** relates to Linux-VServer (OS-level virtualization) rather than LVS load balancing, so it does not manage IPVS/LVS configurations.

References: keepalived project, Linux Virtual Server (LVS) software/tools

QUESTION NO: 8

Within the graphical output of a KVM virtual machine, which key sequence switches to the KVM monitor of the VM?

A. Ctrl-Alt-1

B. Ctrl-Alt-0

C. Ctrl-Alt-4

D. Ctrl-Alt-2

E. Ctrl-Alt-3

ANSWER: D

Explanation:

In a typical QEMU/KVM setup using the graphical console (for example via SDL/GTK display), QEMU provides multiple “virtual consoles” that you can switch between with Ctrl-Alt-number. The QEMU monitor (the interactive management

console where you can run commands like `info`, `stop`, `cont`, etc.) is conventionally on virtual console 2, so the key sequence to switch to it from the guest's graphical output is **Ctrl-Alt-2**. This is a long-standing QEMU convention: Ctrl-Alt-1 returns you to the guest display, while Ctrl-Alt-2 switches to the monitor.

That's why option D is correct. Option A (Ctrl-Alt-1) is commonly used to switch back to the guest display, not to the monitor. Options B, C, and E (Ctrl-Alt-0/4/3) do not correspond to the default monitor console in standard QEMU/KVM key mappings; they may map to other virtual consoles if configured, but they are not the default monitor switch.

References: QEMU Monitor documentation, QEMU invocation and console switching.

QUESTION NO: 9

Which of the following values would be valid in the FROM statement in a Dockerfile?

A. ubuntu:focal

B. docker://ubuntu: focal

C. registry:ubuntu:focal

D. file:/tmp/ubuntu/Dockerfile

E. http://docker.example.com/images/ubuntu-focal.iso

ANSWER: A

Explanation:

In a Dockerfile, the `FROM` instruction must reference a base image using Docker's image reference format: `[registryhost[:port]/]repository[:tag]` (or with an optional `@digest`). A very common valid example is `ubuntu:focal`, which means "use the `ubuntu` repository and the `focal` tag" (by default from Docker Hub's library namespace). That matches option A, so A is correct.

Option B is not valid Dockerfile syntax: `docker://` is a transport prefix used by other tools (e.g., Skopeo/containers-image) but not by Dockerfile `FROM`. It also contains an invalid space in the tag (`ubuntu: focal`). Option C is malformed because `registry:ubuntu:focal` uses multiple colons; a registry host must be separated with `/` (e.g., `registry.example.com/ubuntu:focal`). Option D is invalid because `FROM` cannot point to a local file URL or a Dockerfile path; it must be an image reference. Option E is invalid because `FROM` cannot fetch an ISO or arbitrary HTTP URL; it pulls images from registries using image references.

References: Dockerfile reference: FROM, Docker image naming and tags.

QUESTION NO: 10

How can data in a computing instance in an IaaS cloud be permanently saved and accessed even after the recreation of the computing instance? (Choose TWO correct answers.)

A. By saving the data to the memory of the computing instance using tmpfs.

B. By saving the data anywhere in the computing instance's file system.

C. By saving the data to /cloud/persistent/ which is provided in all common IaaS clouds.

D. By saving the data to object stores provided by a separate service in the cloud.

E. By saving the data on persistent block devices that must be explicitly connected to the computing instance.

ANSWER: D E

Explanation:

In an IaaS cloud, the compute instance is typically treated as ephemeral: if you delete/recreate it (or it is rebuilt from an image), any data stored on the instance's local/root disk is usually lost unless that disk is backed by a separately managed persistent service. To keep data across instance recreation, you generally use storage services that exist independently of the VM lifecycle.

Object storage (option D) is designed exactly for durable, long-lived data and is accessed over the network via APIs (e.g., S3/Swift). Because it's a separate service, the data remains even if the VM is destroyed and later recreated.

Persistent block storage volumes (option E) are also independent resources. You create a volume, attach it to an instance, write data to it, then detach and reattach it to a new instance later. This is the standard pattern in OpenStack Cinder, AWS EBS, etc.

Option A is wrong because tmpfs is RAM-backed and disappears on reboot/termination. Option B is wrong because "anywhere in the instance filesystem" usually means the ephemeral root disk. Option C is wrong because there is no universal /cloud/persistent path across IaaS providers; persistence is provided via services/volumes, not a standardized directory.

References: OpenStack Cinder documentation, Amazon S3 User Guide