

Lpi.305-300J.v2026-02-24.q59

試験コード : 305-300J
試験名称 : LPIC-3 Exam 305: Virtualization and Containerization (305-300日本語版)
認証ベンダー : Lpi
無料問題の数 : 59
バージョン : v2026-02-24
ページの閲覧量 : 104
問題集の閲覧量 : 597
<https://www.jpnsiken.com/shiken/Lpi.305-300J.v2026-02-24.q59.html>

質問: 1

コンテナイメージとは何ですか?

- A. 物理サーバー
- B. 軽量仮想マシン
- C. アプリケーションとその依存関係のスナップショット
- D. オペレーティングシステムカーネル

正解: **C** ([コメントを发表する](#))

A container image is a static, immutable package that contains an application along with all its required dependencies, libraries, and configuration files. Containerization documentation defines a container image as a snapshot of an application and its runtime environment, which can be instantiated as a running container.

Container images do not include a full operating system kernel; instead, containers share the host system's kernel. This design makes containers lightweight and fast to deploy compared to traditional virtual machines.

Therefore, the correct answer is C.

質問: 2

libvirt ドメイン定義の <domain> 要素の type 属性で有効な値は次のうちどれですか? (2つお選びください。)

- A. プロシージャ
- B. ネームスペース
- C. kvm
- D. cgroup
- E. lxc

正解: ([正解を表示します](#))

The type attribute of a <domain> element in a libvirt domain definition specifies the hypervisor used for running the domain. The allowed values are driver specific, but include "xen", "kvm", "hvf" (since 8.1.0 and QEMU 2.12), "qemu" and "lxc"1. Therefore, the valid values among the options are C. kvm and E. lxc. KVM stands for Kernel-based Virtual Machine, which is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or

AMD-V)2. LXC stands for Linux Containers, which is an operating system-level virtualization method for running multiple isolated Linux systems (containers) on a single control host3. The other options are not valid values for the type attribute, as they are either not hypervisors or not supported by libvirt. References:<http://libvirt.org/formatdomain.html>
<https://libvirt.org/formatcaps.html>

質問: 3

コンテナベースの仮想化に関して正しいのは次のどれですか? (2つお選びください。)

- A. 各コンテナは独自のオペレーティング システム カーネルを実行します。
- B. 異なるコンテナは、同じオペレーティング システムの異なるディストリビューションを使用する場合があります。
- C. コンテナベースの仮想化は、ホスト システムの CPU からのハードウェア サポートに依存します。
- D. すべてのコンテナはホスト システムのオペレーティング システム カーネル内で実行されます。
- E. カーネル API が欠落しているため、Linux はコンテナベースの仮想化をサポートしていません。

正解: ([正解を表示します](#))

Container-based virtualization is a method of operating system-level virtualization that allows multiple isolated user spaces (containers) to run on the same host system1. Each container shares the same operating system kernel as the host, but has its own file system, libraries, and processes2. Therefore, the statements A and C are false, as containers do not run their own kernels or rely on hardware support from the CPU. The statement E is also false, as Linux does support container-based virtualization through various technologies, such as cgroups, namespaces, LXC, Docker, etc12. The statement B is true, as different containers may use different distributions of the same operating system, such as Debian, Ubuntu, Fedora, etc., as long as they are compatible with the host kernel3. The statement D is also true, as all containers run within the operating system kernel of the host system, which provides isolation and resource management for them12. References:

* 1: Containerization (computing) - Wikipedia.

* 2: What are containers? | Google Cloud.

* 3: What is Container-Based Virtualization? - StackHowTo.

質問: 4

LXD コンテナ構成で有効なネットワーク インターフェイス タイプは次のうちどれですか? 3つお選びください。)

- A. ipsec
- B. マクブラン
- C. ブリッジ済み
- D. 物理的

E. Wi-Fi

正解: ([正解を表示します](#))

LXD supports the following network interface types in an LXD container configuration¹:

- * **macvlan**: Creates a virtual interface on the host with a unique MAC address and attaches it to an existing physical interface. This allows the container to have direct access to the physical network, but prevents communication with the host and other containers on the same host².
- * **bridged**: Connects the container to an existing bridge interface on the host. This allows the container to communicate with the host and other containers on the same bridge, as well as the external network if the bridge is connected to a physical interface³.
- * **physical**: Passes an existing physical interface on the host to the container. This allows the container to have exclusive access to the physical network, but removes the interface from the host⁴.

The other network interface types, **ipsec** and **wifi**, are not valid in an LXD container configuration. **Ipsec** is a protocol for secure communication over IP networks, not a network interface type. **Wifi** is a wireless technology for connecting devices to a network, not a network interface type.

References:

- * About networking - Canonical LXD documentation
- * Macvlan network - Canonical LXD documentation
- * Bridge network - Canonical LXD documentation
- * Physical network - Canonical LXD documentation

質問: 5

Packer はシステム イメージとどのように対話しますか？

- A.** Packer はターゲット イメージ内にインストールする必要があり、準備タスクを実行するためにイメージの最初の起動時に実行されます。
- B.** Packer はイメージ内にクライアントをインストールします。このクライアントは、Packer サーバーから最新のテンプレートを取得してローカルに適用するために、cron 経由で定期的に行う必要があります。
- C.** Packer は、実行中のすべての Packer イメージの Packer デーモンにネットワーク経由で定期的に接続し、テンプレート全体を実行中のインスタンスに再適用します。
- D.** Packer は、イメージのファイル システムを変更するためにイメージをダウンロードして抽出し、変更されたイメージを再パックして、再度アップロードします。
- E.** Packer はソース イメージに基づいてインスタンスを作成し、ネットワーク接続を通じてインスタンスを準備し、結果のインスタンスを新しいシステム イメージとしてバンドルします。

正解: ([正解を表示します](#))

Packer is a tool that automates the creation of identical machine images for multiple platforms from a single source configuration. Packer works by creating an instance based on a source image, which is a pre-existing image that serves as a starting point. Packer then connects to the instance through a network connection, such as SSH or WinRM, and runs various commands and scripts to install and configure software within the instance. Packer then shuts down the

instance and creates a new system image from it, which can be used to launch new instances. Packer supports many platforms, such as AWS, Azure, VMware, Docker, and others. Packer does not install any software or run any daemon within the target image, nor does it periodically connect to the running instances to re-apply the template. Packer also does not modify the source image directly, but creates a new image from the modified instance.

References:

- * Packer by HashiCorp
- * HashiCorp Packer - Build Automated Machine Images
- * Introduction | Packer | HashiCorp Developer

質問: 6

クラウド インスタンスを構成するために使用できる cloud-init データ ソースの種類は何ですか (該当するものをすべて選択してください)

- A. クラウドベースの構成管理ツール
- B. メタデータサービス
- C. QRコード
- D. ユーザーデータスクリプト

正解: ([正解を表示します](#))

Cloud-init is a widely used initialization system for cloud instances, allowing configuration at first boot.

According to cloud-init documentation, the two primary data sources used to configure instances are metadata services and user data scripts.

A metadata service provides instance-specific information such as hostname, instance ID, networking details, and SSH keys. User data scripts allow administrators to supply custom configuration instructions, commonly written as shell scripts or cloud-config YAML.

Cloud-based configuration management tools like Ansible or Puppet are not cloud-init data sources; instead, cloud-init may install or trigger them. QR codes are not supported data sources in cloud-init.

Virtualization and cloud documentation clearly identifies metadata and user data as the supported configuration mechanisms. Therefore, the correct answers are B and D.

質問: 7

次の KVM パラメータのうち、KVM パラメータ `-hdb file.img` と同じものはどれですか?

- A. `-drive bus=ide1,type=slave,image=file.img`
- B. `-drive type=loop,src=file.img,dst=disk:hdb`
- C. `-drive image=file.img,if=ide,device=hdb`
- D. `-drive file=file.img,index=1,media=disk,if=ide`
- E. `-drive bus=hd,busid=b,src=file.img`

正解: ([正解を表示します](#))

In QEMU/KVM, the legacy parameter `-hdb file.img` attaches a disk image as the primary IDE slave device.

According to QEMU documentation, this corresponds to an IDE device with index 1, media type disk, and interface IDE.

The modern equivalent using the -drive syntax is:

`-drive file=file.img,index=1,media=disk,if=ide`

Option D matches this mapping precisely. The other options use invalid or unsupported parameter combinations and do not correctly represent the IDE device mapping.

Therefore, the correct answer is D.

質問: 8

Docker Swarm で Docker Compose ファイルを実行するために Docker スタックを使用する場合、Docker Compose 構成で参照されるイメージはどのようにして Swarm ノードで利用できるようになりますか？

- A. docker スタックはイメージをローカルに構築し、サービスを実行する Swarm ノードのみにコピーします。
- B. Docker スタックはイメージを Swarm マスターに渡し、Swarm マスターはイメージを他のすべての Swarm ノードに配布します。
- C. docker スタックは、レジストリにイメージをアップロードしませんが、レジストリからイメージをプルするように Swarm ノードに指示します。
- D. docker スタックは、イメージをローカル Docker キャッシュから各 Swarm ノードに転送します。
- E. docker スタックは、Swarm のすべてのノード上のイメージのビルド プロセスをトリガーします。

正解: ([正解を表示します](#))

Docker stack is a command that allows users to deploy and manage a stack of services on a Docker Swarm cluster. A stack is a group of interrelated services that share dependencies and can be orchestrated and scaled together. A stack is typically defined by a Compose file, which is a YAML file that describes the services, networks, volumes, and other resources of the stack. To use docker stack to run a Compose file on a Swarm, the user must first create and initialize a Swarm cluster, which is a group of machines (nodes) that are running the Docker Engine and are joined into a single entity. The Swarm cluster has one or more managers, which are responsible for maintaining the cluster state and orchestrating the services, and one or more workers, which are the nodes that run the services.

When the user runs docker stack deploy with a Compose file, the command parses the file and creates the services as specified. However, docker stack does not build or upload the images referenced in the Compose file to any registry. Instead, it instructs the Swarm nodes to pull the images from a registry, which can be the public Docker Hub or a private registry. The user must ensure that the images are available in the registry before deploying the stack, otherwise the deployment will fail. The user can use docker build and docker push commands to create and upload the images to the registry, or use an automated build service such as Docker Hub or GitHub Actions. The user must also make sure that the image names and tags in the Compose

file match the ones in the registry, and that the Swarm nodes have access to the registry if it is private. By pulling the images from a registry, docker stack ensures that the Swarm nodes have the same and latest version of the images, and that the images are distributed across the cluster in an efficient way.

The other options are not correct. Docker stack does not build the images locally or on the Swarm nodes, nor does it copy or transfer the images to the Swarm nodes. Docker stack also does not pass the images to the Swarm master, as this would create a bottleneck and a single point of failure. Docker stack relies on the registry as the source of truth for the images, and delegates the image pulling to the Swarm nodes. References:

* Deploy a stack to a swarm | Docker Docs1

* docker stack deploy | Docker Docs2

* docker build | Docker Docs3

* docker push | Docker Docs4

質問: 9

Vagrant の Vagrantfile 構成ファイルの主な目的は何ですか？

- A. 仮想マシンのハードウェア要件を定義する
- B. 利用可能なソフトウェアパッケージのリストを提供する
- C. 開発者の名前と連絡先情報を指定する
- D. クラウドプロバイダーの詳細を指定する

正解: ([正解を表示します](#))

The Vagrantfile is the core configuration file used by Vagrant to define and control virtual machine environments. According to Vagrant documentation, its primary purpose is to describe the virtual machine configuration, including hardware-related settings such as memory size, CPU count, networking, synced folders, and provisioning instructions.

The Vagrantfile allows developers to declare VM characteristics in a reproducible and portable way, ensuring consistent development environments across teams. While it can also reference providers and provisioning tools, its fundamental role is defining how the virtual machine should be created and configured.

Options B and C are unrelated to Vagrant's purpose, and option D is too narrow, as Vagrantfile usage is not limited to cloud providers.

Therefore, the correct answer is A.

質問: 10

xl のどのサブコマンドが、現在 Xen ゲスト ドメインに接続されているブロック デバイスを一覧表示しますか？

正解:

block-list

Explanation:

In Xen virtualization, the `xl toolstack` provides commands to manage and inspect guest domains. According to Xen documentation, the `block-list` sub-command is used to list all block devices currently attached to a specific Xen guest domain. This command displays information such as virtual device names, backend devices, and access modes, allowing administrators to verify disk attachments and troubleshoot storage-related issues. It is commonly used after attaching or detaching disks using `block-attach` or `block-detach`. Therefore, the correct and documented sub-command is `block-list`.

質問: 11

cloud-init と互換性のあるオペレーティング システムはどれですか? (該当するものをすべて選択してください)

- A. Windows
- B. macOS
- C. Linux
- D. Android

正解: ([正解を表示します](#))

Cloud-init is an initialization and configuration system designed primarily for Linux-based cloud instances.

According to official cloud-init documentation, it is supported on a wide range of Linux distributions, including Ubuntu, Red Hat Enterprise Linux, CentOS, Debian, and SUSE. Windows systems use a separate project called Cloudbase-Init, which is not cloud-init itself. macOS and Android do not support cloud-init. Therefore, the correct answer is C (Linux).

質問: 12

LXC ではどのような仮想化が実装されていますか?

- A. システムコンテナ
- B. アプリケーションコンテナ
- C. ハードウェアコンテナ
- D. CPU エミュレーション
- E. 準仮想化

正解: ([正解を表示します](#))

LXC implements system containers, which are a type of operating-system-level virtualization. System containers allow running multiple isolated Linux systems on a single Linux control host, using a single Linux kernel. System containers share the same kernel with the host and each other, but have their own file system, libraries, and processes. System containers are different from application containers, which are designed to run a single application or service in an isolated environment. Application containers are usually smaller and more portable than system containers, but also more dependent on the host kernel and libraries. Hardware containers, CPU emulation, and paravirtualization are not related to LXC, as they are different kinds of

virtualization methods that involve hardware abstraction, instruction translation, or modification of the guest operating system. References:

- * 1: LXC - Wikipedia
- * 2: Linux Virtualization : Linux Containers (lxc) - GeeksforGeeks
- * 3: Features - Proxmox Virtual Environment

質問: 13

KVM 仮想化ホストをセットアップする場合、次のコンポーネントのうちどれが必須ではありませんか？

- A. ブリッジユーティリティ
- B. kvm カーネルモジュール
- C. qemu
- D. libvirt
- E. virsh

正解: ([正解を表示します](#))

When configuring a KVM-based virtualization host, several core components are mandatory to enable and manage virtual machines. According to KVM and virtualization documentation, KVM kernel modules are essential because they provide hardware-assisted virtualization support within the Linux kernel. QEMU is required to perform hardware emulation and manage virtual machine execution. Libvirt acts as the virtualization management API, enabling centralized and secure control of virtual machines. Additionally, bridge-utils is commonly required to configure network bridges, allowing virtual machines to communicate with external networks.

However, virsh is not strictly required. Virsh is a command-line utility that interacts with libvirt to manage virtual machines, but it is only a management interface, not a core dependency. Virtual machines can still be created and managed using alternative tools such as virt-manager, Ansible, OpenStack, or custom API-based solutions without virsh being installed.

Virtualization documentation clearly distinguishes between essential backend components (KVM, QEMU, libvirt) and optional management tools (virsh). Therefore, while virsh is widely used and highly recommended for administrative convenience, it is not a mandatory component for a functional KVM virtualization host.

質問: 14

Dockerfile の FROM ステートメントで有効な値は次のうちどれですか？

- A. ubuntu:focal
- B. docker://ubuntu: 焦点
- C. レジストリ:ubuntu:focal
- D. ファイル:/tmp/ubuntu/Dockerfile
- E. http://docker.example.com/images/ubuntu-focal.iso

正解: ([正解を表示します](#))

The FROM statement in a Dockerfile specifies the base image from which the subsequent instructions are executed¹. The value of the FROM statement can be either an image name, an

image name with a tag, or an image ID¹. The image name can be either a repository name or a repository name with a registry prefix². For example, ubuntu is a repository name, and docker.io/ubuntu is a repository name with a registry prefix². The tag is an optional identifier that can be used to specify a particular version or variant of an image¹. For example, ubuntu:focal refers to the image with the focal tag in the ubuntu repository². The image ID is a unique identifier that is automatically generated when an image is built or pulled¹. For example, sha256:9b0dafaadb1cd1d14e4db51bd0f4c0d56b6b551b2982b2b7c637ca143ad605d2 is an image ID³. Therefore, the only valid value in the FROM statement among the given options is ubuntu:focal, which is an image name with a tag. The other options are invalid because:

* docker://ubuntu:focal is not a valid image name format. The docker:// prefix is used to specify a transport protocol, not a registry prefix⁴.

* registry:ubuntu:focal is not a valid image name format. The registry prefix should be a valid hostname or IP address, not a generic term².

* file:/tmp/ubuntu/Dockerfile is not a valid image name format. The file: prefix is used to specify a local file path, not an image name⁵.

* http://docker.example.com/images/ubuntu-focal.iso is not a valid image name format.

The

http:// prefix is used to specify a web URL, not an image name

5.

References:

1: Dockerfile reference | Docker Docs

2: docker - Using FROM statement in dockerfile - Stack Overflow

3: How to get the image id from a docker image - Stack Overflow

4: skopeo - Docker Registry v2 API tool - Linux Man Pages (1)

5: How to build a Docker image from a local Dockerfile? - Stack Overflow

質問: 15

acgroupdirectory 内のどのファイルに、thiscgroup に属するプロセスのリストが含まれていますか?

A. pid

B. メンバー

C. プロシージャ

D. casks

E. 件名

正解: ([正解を表示します](#))

The file procs in a cgroup directory contains the list of processes belonging to this cgroup. Each line in the file shows the PID of a process that is a member of the cgroup. A process can be moved to a cgroup by writing its PID into the cgroup's procs file. For example, to move the process with PID 24982 to the cgroup cg1, the following command can be used: echo 24982 > /sys/fs/cgroup/cg1/procs1. The file procs is different from the file tasks, which lists the threads

belonging to the cgroup. The file procs can be used to move all threads in a thread group at once, while the file tasks can be used to move individual threads². References:

- * Creating and organizing cgroups cgroup2 - GitHub Pages
- * Control Groups - The Linux Kernel documentation

質問: 16

仮想マシンに接続された CD-ROM イメージ ファイルを変更するために使用される virsh サブコマンドはどれですか？

正解:

change-media

Explanation:

In libvirt-based virtualization environments, the `virsh change-media` subcommand is used to change the media attached to removable devices such as CD-ROM drives. According to libvirt documentation, this command allows administrators to insert or eject ISO images from a running virtual machine without shutting it down.

This functionality is commonly used during operating system installations or software updates. The `change-media` command supports both live and persistent changes depending on the options used.

Therefore, the correct and documented answer is `change-media`.

有効的な**305-300J**問題集はJPNTTest.com提供され、**305-300J**試験に合格することに役に立ちます！JPNTTest.comは今最新**305-300J**試験問題集を提供します。JPNTTest.com 305-300J試験問題集はもう更新されました。ここで**305-300J**問題集のテストエンジンを手に入れます。最新版のアクセス、<https://www.jpntest.com/shiken/305-300J-mondaishu> **125問、30%ディスカウント**、特別な割引コード: **JPNshiken**」

質問: 17

完全仮想化のリソース管理に関して正しいのは次のどれですか？(2つお選びください。)

- A. ハイジャバイザは、プロセス数などのゲスト オペレーティング システムの内部要素に対してきめ細かい制限を提供する場合があります。
- B. ハイパーバイザーは、仮想マシンのリソースを制限する定義された容量のハードウェアを各仮想マシンに提供します。
- C. 完全仮想化では仮想マシンに制限を設けることができず、常に先着順でホスト システムのリソースが割り当てられます。
- D. 仮想マシン内で作成されたすべてのプロセスは、CPU と I/O の使用量に合わせてホスト システムで透過的かつ均等にスケジュールされます。
- E. 割り当てられたハードウェア リソースを使用して、たとえば任意の数のネットワーク ソケットを作成するかどうかは、仮想マシン次第です。

正解: **B,E** ([コメントを发表する](#))

Resource management for full virtualization is the process of allocating and controlling the physical resources of the host system to the virtual machines running on it. The hypervisor is the software layer that performs this task, by providing each virtual machine with a virtual hardware of a defined capacity that limits the resources of the virtual machine. For example, the hypervisor can specify how many virtual CPUs, how much memory, and how much disk space each virtual machine can use. The hypervisor can also enforce resource isolation and prioritization among the virtual machines, to ensure that they do not interfere with each other or consume more resources than they are allowed to. The hypervisor cannot provide fine-grained limits to internal elements of the guest operating system, such as the number of processes, because the hypervisor does not have access to the internal state of the guest operating system. The guest operating system is responsible for managing its own resources within the virtual hardware provided by the hypervisor. For example, the guest operating system can create an arbitrary amount of network sockets, as long as it does not exceed the network bandwidth allocated by the hypervisor. Full virtualization can pose limits to virtual machines, and does not always assign the host system's resources in a first-come-first-serve manner. The hypervisor can use various resource management techniques, such as reservation, limit, share, weight, and quota, to allocate and control the resources of the virtual machines. The hypervisor can also use resource scheduling algorithms, such as round-robin, fair-share, or priority-based, to distribute the resources among the virtual machines according to their needs and preferences. All processes created within the virtual machines are not transparently and equally scheduled in the host system for CPU and I/O usage. The hypervisor can use different scheduling policies, such as proportional-share, co-scheduling, or gang scheduling, to schedule the virtual CPUs of the virtual machines on the physical CPUs of the host system. The hypervisor can also use different I/O scheduling algorithms, such as deadline, anticipatory, or completely fair queuing, to schedule the I/O requests of the virtual machines on the physical I/O devices of the host system. The hypervisor can also use different resource accounting and monitoring mechanisms, such as cgroups, perf, or sar, to measure and report the resource consumption and performance of the virtual machines. References:

* Oracle VM VirtualBox: Features Overview

* Resource Management as an Enabling Technology for Virtualization - Oracle

* Introduction to virtualization and resource management in IaaS | Cloud Native Computing Foundation

質問: **18**

準仮想化マシンではなく完全仮想化マシンを作成するには、Xen ゲスト ドメインの構成ファイルのオプション ビルダーでどの値を設定する必要がありますか？

正解:

builder = "hvm"

Explanation:

In Xen virtualization, guest domains can be created using either paravirtualization (PV) or hardware-assisted full virtualization, also known as HVM (Hardware Virtual Machine). According to official Xen documentation, the builder option in a Xen guest domain configuration file determines the type of virtualization used when creating a virtual machine.

To create a fully virtualized (HVM) guest, the configuration file must explicitly set the value of the builder option to "hvm":

```
builder = "hvm"
```

This instructs the Xen toolstack to use hardware-assisted virtualization features provided by the CPU, such as Intel VT-x or AMD-V, allowing unmodified guest operating systems to run. Fully virtualized guests behave like physical machines and can run operating systems that are unaware of Xen.

If the builder option is omitted or set to the default paravirtualized builder, Xen expects a PV-capable kernel, which requires guest operating system modifications. HVM guests, by contrast, rely on device emulation and hardware virtualization support.

Virtualization documentation clearly distinguishes between PV and HVM guests and identifies

```
builder =
```

"hvm" as the required configuration for full virtualization. This setting is fundamental when deploying operating systems such as Windows or standard Linux distributions without Xen-specific kernels.

Therefore, the correct and documented value is `builder = "hvm"`.

質問: 19

次のタスクのうち、ハイパーバイザーの責任の一部となるのはどれですか? (2つお選びください。)

- A. 新しい仮想マシン Quest オペレーティング システムのインストール中にファイル システムを作成します。
- B. 仮想マシン間のプロセス間通信を容易にするために、仮想マシン内で実行されているプロセスにホスト全体で一意的 PID を提供します。
- C. 仮想マシンのリソースをホスト システムのリソースにマッピングします。
- D. 仮想マシン内で実行されているネットワーク サービスへの認証を管理します。
- E. 仮想マシンを隔離し、他の仮想マシンのリソースへの不正アクセスを防止します。

正解: ([正解を表示します](#))

A hypervisor is a software that creates and runs virtual machines (VMs) by separating the operating system and resources from the physical hardware. One of the main tasks of a hypervisor is to map the resources of VMs to the resources of the host system, such as CPU, memory, disk, and network. This allows the hypervisor to allocate and manage the resources among multiple VMs and ensure that they run efficiently and independently¹²³. Another important task of a hypervisor is to isolate the VMs and prevent unauthorized access to resources of other VMs. This ensures the security and privacy of the VMs and their data, as well

as the stability and performance of the host system. The hypervisor can use various techniques to isolate the VMs, such as virtual LANs, firewalls, encryption, and access control¹⁴⁵.

The other tasks listed are not part of a hypervisor's responsibility, but rather of the guest operating system or the application running inside the VM. A hypervisor does not create filesystems during the installation of new VMs, as this is done by the installer of the guest operating system⁶. A hypervisor does not provide host-wide unique PIDs to the processes running inside the VMs, as this is done by the kernel of the guest operating system⁷. A

hypervisor does not manage authentication to network services running inside a VM, as this is done by the network service itself or by a directory service such as LDAP or Active Directory⁸.

References: 1 (search for "What is a hypervisor?"), 2 (search for "How does a hypervisor work?"), 3 (search for "The hypervisor gives each virtual machine the resources that have been allocated"), 4 (search for "Benefits of hypervisors"), 5 (search for "Isolate the virtual machines and prevent unauthorized access"), 6 (search for "Create filesystems during the installation of new virtual machine guest operating systems"), 7 (search for "Provide host-wide unique PIDs to the processes running inside the virtual machines"), 8 (search for "Manage authentication to network services running inside a virtual machine").

質問: 20

Packer を使用してマシン イメージを作成する場合、ベース イメージの一般的なソースは何ですか (該当するものをすべて選択してください)。

- A. 物理サーバー
- B. 信頼できないソースからの画像
- C. 組織によって作成されたカスタムイメージ
- D. 公式オペレーティング システム イメージ

正解: ([正解を表示します](#))

Packer is an image automation tool used to create identical machine images for multiple platforms from a single configuration. According to virtualization and containerization documentation, Packer typically uses trusted and standardized base images as the starting point for image creation. The most common sources include official operating system images provided by vendors such as Red Hat, Ubuntu, or cloud service providers, and custom images created internally by an organization.

Using official images ensures that the base system is secure, properly maintained, and compliant with vendor standards. Organizations often build custom base images to enforce internal security policies, hardening standards, and preinstalled software, which Packer then extends into environment-specific images.

Physical servers are not considered common or practical base sources for Packer, as Packer operates on machine image artifacts rather than cloning live hardware systems. Images from untrusted sources are explicitly discouraged in virtualization documentation due to security and compliance risks.

Therefore, documentation clearly supports custom organizational images and official operating system images as the correct and recommended base sources for Packer image creation.

質問: 21

Linux で仮想マシンのディスク イメージを作成および管理するために使用されるユーティリティはどれですか？

- A. ディスクユーティリティ
- B. VirtualBox マネージャー
- C. Libvirt ディスクマネージャー
- D. qemu-img

正解: **D** ([コメントを发表する](#))

In Linux-based virtualization environments, `qemu-img` is the standard utility used to create, convert, inspect, and manage virtual machine disk images. Virtualization and containerization documentation clearly identifies `qemu-img` as a core component of the QEMU/KVM virtualization stack. It supports multiple disk image formats such as RAW, QCOW2, VMDK, and VHD, making it highly versatile across different hypervisors and platforms.

The `qemu-img` tool allows administrators to perform critical disk operations, including creating new disk images, resizing existing images, converting between formats, and checking disk integrity. For example, it is commonly used to create QCOW2 images, which support advanced features like copy-on-write, snapshots, and compression.

Other options listed are incorrect because `Disk Utility` is a general-purpose storage management tool, `VirtualBox Manager` is specific to Oracle VirtualBox, and `Libvirt Disk Manager` is not a recognized standalone utility. Libvirt relies on `qemu-img` internally for disk image handling. Official virtualization notes consistently reference `qemu-img` as the primary disk image management tool in Linux virtualization environments, especially when working with KVM and libvirt-based systems.

質問: 22

libvirt ドメイン定義の `<os>` 要素の `firmware` 属性で有効な値は次のうちどれですか？ (2 つ選択してください)。

- A. scsi
- B. virtio
- C. efi
- D. bios
- E. pcie

正解: ([正解を表示します](#))

The `firmware` attribute of the `<os>` element in a libvirt domain definition specifies the type of firmware used to boot the virtual machine. The valid values for this attribute are `efi` and `bios`, which correspond to the Extensible Firmware Interface (EFI) and the Basic Input/Output System (BIOS) respectively. EFI is a newer standard that supports more features and security than BIOS, such as Secure Boot and faster boot times. BIOS is an older standard that is widely compatible and supported by most hypervisors and operating systems. The other values, `scsi`,

virtio, and pcie, are not related to firmware, but to different types of devices or drivers that can be used in a virtual machine. References: 1 (search for firmware enum)

質問: 23

ユーザー ネットワークで KVM を使用する場合、次の制限のうちどれが当てはまりますか？

- A. 仮想マシンは、ユーザーネットワークを使用してホストシステムや他の仮想マシンと通信できます。
- B. 仮想マシンは他の仮想マシンとのみ通信でき、ホストシステムとは通信できません。
- C. 仮想マシンはICMPを使用できません
- D. 仮想マシンは IPv6 のみの使用に制限され、IPv4 は使用できません。

正解: [C \(コメントを发表する\)](#)

KVM User Networking (SLIRP) provides basic network connectivity without requiring root privileges or bridge configuration. According to QEMU/KVM documentation, one key limitation is that ICMP (ping) is not supported.

User Networking allows outbound TCP/UDP connections but restricts inbound connections and ICMP traffic.

Therefore, C is the correct answer.

質問: 24

VirtualBox に関して正しいのは次のどれですか？

- A. これは、最初の通常のオペレーティング システムが起動する前に起動される特別なカーネルとして設計されたハイパーバイザーです。
- B. ゲスト オペレーティング システムとして Linux のみをサポートし、仮想マシン内で Windows を実行することはできません。
- C. 仮想マシンのディスク イメージを仮想化ホストのブロック デバイスにローカルに保存できないため、専用の共有ストレージが必要です。
- D. 仮想マシンを管理するためのグラフィカル ユーザー インターフェイスとコマンド ライン ツールの両方を提供します。
- E. Linux でのみ使用でき、現在実行中の Linux カーネルのソース コードが使用可能である必要があります。

正解: [\(正解を表示します\)](#)

VirtualBox is a hosted hypervisor, which means it runs as an application on top of an existing operating system, not as a special kernel that is booted before the first regular operating system starts¹. VirtualBox supports a large number of guest operating systems, including Windows, Linux, Solaris, OS/2, and OpenBSD¹. VirtualBox does not require dedicated shared storage, as it can store virtual machine disk images locally on block devices of the virtualization host, or on network shares, or on iSCSI targets¹. VirtualBox provides both a graphical user interface (GUI) and command line tools (VBoxManage) to administer virtual machines¹. VirtualBox is available for Windows, Linux, macOS, and Solaris hosts¹, and does not require the source code of the currently running Linux kernel to be available. References:

質問: 25

Docker アーキテクチャに必須のコンポーネントは次のどれですか (該当するものをすべて選択してください)。

- A. Dockerクライアント
- B. Docker Compose
- C. Dockerデーモン
- D. Dockerレジストリ

正解: ([正解を表示します](#))

Docker architecture consists of several core components that enable container creation and management.

According to Docker documentation, the Docker Client, Docker Daemon, and Docker Registry are essential components.

The Docker Client is the user-facing command-line interface that sends commands to the daemon. The Docker Daemon (dockerd) performs the heavy lifting, such as building images, running containers, and managing networks and storage. The Docker Registry stores container images, enabling image distribution and reuse.

Docker Compose is an optional orchestration tool used to define and run multi-container applications, but it is not an essential architectural component.

Therefore, the correct answers are A, C, and D.

質問: 26

コンテナに関連付けられていないすべてのボリュームを削除するコマンドは次のどれですか?

- A. Docker ボリュームのクリーンアップ
- B. docker volume orphan -d
- C. Docker ボリューム プルーン
- D. Docker ボリュームのバキューム
- E. Docker ボリュームのガベージコレクション

正解: ([正解を表示します](#))

The command that deletes all volumes which are not associated with a container is docker volume prune. This command removes all unused local volumes, which are those that are not referenced by any containers. By default, it only removes anonymous volumes, which are those that are not given a specific name when they are created. To remove both unused anonymous and named volumes, the --all or -a flag can be added to the command. The command will prompt for confirmation before deleting the volumes, unless the --force or -f flag is used to bypass the prompt. The command will also show the total reclaimed space after deleting the volumes¹².

The other commands listed in the question are not valid or do not have the same functionality as docker volume prune. They are either made up, misspelled, or have a different purpose. These commands are:

- * docker volume cleanup: This command does not exist in Docker. There is no cleanup subcommand for docker volume.
- * docker volume orphan -d: This command does not exist in Docker. There is no orphan subcommand for docker volume, and the -d flag is not a valid option for any docker volume command.
- * docker volume vacuum: This command does not exist in Docker. There is no vacuum subcommand for docker volume.
- * docker volume garbage-collect: This command does not exist in Docker. There is no garbage-collect subcommand for docker volume.

:

docker volume prune | Docker Docs

How to Remove all Docker Volumes - YallaLabs.

質問: 27

Docker Hub とは何ですか?

- A. Linuxディストリビューション
- B. コンテナイメージレジストリ
- C. コンテナオーケストレーションプラットフォーム
- D. コンテナランタイム

正解: ([正解を表示します](#))

Docker Hub is a cloud-based container image registry used to store, manage, and distribute Docker container images. According to container documentation, it hosts both official images and user-contributed images.

Docker Hub is not a Linux distribution, orchestration platform, or runtime. Therefore, the correct answer is B.

質問: 28

Vagrant のデフォルトのプロバイダーは何ですか?

- A. lxc
- B. ハイパーブ
- C. 仮想ボックス
- D. vmware_workstation
- E. ドッカー

正解: C ([コメントを发表する](#))

Vagrant is a tool that allows users to create and configure lightweight, reproducible, and portable development environments. Vagrant supports multiple providers, which are the backends that Vagrant uses to create and manage the virtual machines. By default, VirtualBox is the default provider for Vagrant.

VirtualBox is still the most accessible platform to use Vagrant: it is free, cross-platform, and has been supported by Vagrant for years. With VirtualBox as the default provider, it provides the lowest friction for new users to get started with Vagrant. However, users can also use other

providers, such as VMware, Hyper- V, Docker, or LXC, depending on their preferences and needs. To use another provider, users must install it as a Vagrant plugin and specify it when running Vagrant commands. Users can also change the default provider by setting the VAGRANT_DEFAULT_PROVIDER environmental variable. References:

- * Default Provider - Providers | Vagrant | HashiCorp Developer1
- * Providers | Vagrant | HashiCorp Developer2
- * How To Set Default Vagrant Provider to Virtualbox3

質問: 29

インフラストラクチャ・アズ・コード (IaC) アプローチで知られているクラウド管理ツールはどれですか (該当するものをすべて選択してください)。

- A. 人形
- B. テラフォーム
- C. AWS クラウドフォーメーション
- D. アンシブル

正解: ([正解を表示します](#))

Infrastructure as Code (IaC) is an approach where infrastructure is defined and managed using machine-readable configuration files. According to DevOps and cloud documentation, Puppet, Terraform, AWS CloudFormation, and Ansible all support IaC principles. Terraform and AWS CloudFormation are declarative IaC tools used to provision cloud infrastructure. Puppet and Ansible are configuration management and automation tools that also enable infrastructure definition through code.

All listed tools are widely recognized in IaC workflows, making A, B, C, and D correct.

質問: 30

ハードウェア仮想化拡張機能を使用して QEMU 仮想マシンを起動するコマンドは次のどれですか?

- A. `qvirt -create -drive file=debian.img -cdrom debian.iso -m 1024 -boot d -driver hvm`
- B. `vm -kvm -drive file=debian.img -cdrom debian.iso -m 1024 -boot d`
- C. `qemu-hw -create -drive file=debian.img -cdrom debian.iso -m 1024 -boot d`
- D. `qemu -accel kvm -drive file=debian.img -cdrom debian.iso -m 1024 -boot d`
- E. `qvm start -vmx -drive file=debian.img -cdrom debian.iso -m 1024 -boot d`

正解: ([正解を表示します](#))

The correct command to boot a QEMU virtual machine using hardware virtualization extensions is `qemu - accel kvm -drive file=debian.img -cdrom debian.iso -m 1024 -boot d`. This command uses the `-accel` option to specify the hardware accelerator to use, which in this case is `kvm`. KVM is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V)¹. The `-drive` option specifies the disk image file to use, which in this case is `debian.img`. The `-cdrom` option specifies the ISO image file to use as a CD-ROM, which in this case is `debian.iso`. The `-m` option specifies the amount of memory to allocate to the virtual

machine, which in this case is 1024 MB. The -boot option specifies the boot order, which in this case is d, meaning to boot from the CD-ROM first. References:

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/virtualization_deployment_and_administration_guide/sect-troubleshooting-enabling_intel_vt_x_and_amd_v_virtualization_hardware_extensions_in_bios
<https://fedoraproject.org/wiki/Virtualization>

質問: 31

Intel CPU アーキテクチャで KVM を使用するためにロードする必要があるカーネル モジュールの名前は何ですか?

(パス情報なしで、モジュール サフィックスの有無にかかわらず、モジュール名のみを指定します。) 解決策:

kvm-intel.ko - または - kvm-intel - または - kvm_intel.ko - または - kvm_intel
与えられた解決策が正しいかどうかを判断します。

A. 正解

B. 偽

正解: ([正解を表示します](#))

Kernel-based Virtual Machine (KVM) relies on hardware-assisted virtualization features provided by modern CPUs. On Intel CPU architectures, this support is enabled through the `kvm_intel` kernel module. Official KVM and Linux virtualization documentation clearly states that the required module for Intel processors is named `kvm_intel`, with the optional `.ko` suffix when referring to the kernel object file.

The provided solution lists multiple acceptable representations of the module name, including `kvm_intel` and `kvm_intel.ko`, both of which are valid and correct. Linux kernel module naming conventions allow the module to be referenced with or without the `.ko` suffix when loading it using tools such as `modprobe` or `lsmod`.

Although variants using a hyphen (`kvm-intel`) are not the canonical kernel module name, the solution explicitly includes the correct and documented module name. Therefore, the solution correctly identifies the required kernel module for enabling KVM on Intel CPUs.

Virtualization documentation emphasizes that KVM functionality requires both the generic `kvm` module and the CPU-specific module (`kvm_intel` for Intel or `kvm_amd` for AMD). Hence, the determination that the solution is correct aligns with verified documentation.

有効的な**305-300J**問題集はJPNTTest.com提供され、**305-300J**試験に合格することに役に立ちます！JPNTTest.comは今最新**305-300J**試験問題集を提供します。JPNTTest.com 305-300J試験問題集はもう更新されました。ここで**305-300J**問題集のテストエンジンを手に入れます。最新版のアクセス、<https://www.jpntest.com/shiken/305-300J-mondaishu> **125問、30%ディスカウント**、特別な割引コード: **JPNshiken**」

質問: 32

ハードウェア仮想化拡張機能で QEMU を使用するためにロードする必要がある Linux カーネルモジュールに関して正しいのはどれですか？

- A. 仮想マシンのコンソールがホスト システムの物理コンソールに接続される場合にのみ、ホスト システムのカーネルにロードする必要があります。
- B. ホスト システムのファイル システムからファイルやディレクトリにアクセスする各仮想マシンのカーネルにロードする必要があります。
- C. ホスト システムの CPU の視覚化拡張機能を使用するには、ホスト システムのカーネルにロードする必要があります。
- D. QEMU ベアメタル ハイパーバイザーと対話するため、最初の仮想マシンのカーネルにロードする必要があります、追加の仮想マシンの起動をトリガーするために必要です
- E. QEMU で必要な Para 仮想化を提供するには、各仮想マシンのカーネルにロードする必要があります。

正解: [\(正解を表示します\)](#)

The Linux kernel module that must be loaded in order to use QEMU with hardware virtualization extensions is KVM (Kernel-based Virtual Machine). KVM is a full virtualization solution that allows a user space program (such as QEMU) to utilize the hardware virtualization features of various processors (such as Intel VT or AMD-V). KVM consists of a loadable kernel module, `kvm.ko`, that provides the core virtualization infrastructure and a processor specific module, `kvm-intel.ko` or `kvm-amd.ko`. KVM must be loaded into the kernel of the host system in order to use the virtualization extensions of the host system's CPU. This enables QEMU to run multiple virtual machines with unmodified Linux or Windows images, each with private virtualized hardware. KVM is integrated with QEMU, so there is no need to load it into the kernel of each virtual machine or the first virtual machine. KVM also does not require paravirtualization, which is a technique that modifies the guest operating system to communicate directly with the hypervisor, bypassing the emulation layer. References:

- * [Features/KVM - QEMU](#)
- * [Kernel-based Virtual Machine](#)
- * [KVM virtualization on Red Hat Enterprise Linux 8 \(2023\)](#)

質問: 33

オプション `dom0_mem` を使用して Xen Domain-0 に割り当てられるメモリ量を制限するには、このオプションをどこで指定する必要がありますか？

- A. ブートローダー構成内 (Xen 起動時)。
- B. Xen のグローバル構成ファイルのいずれか。
- C. Domain-0 カーネルの構築時の `.config` ファイル内。
- D. Xen の起動時の構成ファイル `/etc/xen/Domain-0.cfg` 内。
- E. Xen のビルド時の Makefile 内。

正解: A ([コメントを發表する](#))

The option `dom0_mem` is used to set the initial and maximum memory size of the Domain-0, which is the privileged domain that starts first and manages the unprivileged domains (DomU) in Xen. The option `dom0_mem` must be specified in the bootloader configuration, such as GRUB or GRUB2, when Xen is booted. This ensures that the Domain-0 kernel can allocate memory for storing memory metadata and network related parameters based on the boot time amount of memory. If the option `dom0_mem` is not specified in the bootloader configuration, the Domain-0 will use all the available memory on the host system by default, which may cause performance and security issues. References:

- * Managing Xen Dom0's CPU and Memory
- * Xen Project Best Practices
- * Dom0 Memory - Where It Has Not Gone

質問: 34

QEMU 仮想マシンの CPU に関して正しいのは次のうちどれですか? (2つお選びください。)

- A. QEMU 仮想マシンの CPU アーキテクチャは、ホスト システムのアーキテクチャから独立しています。
- B. 各 QEMU 仮想マシンは、1 つのコアを持つ 1 つの CPU のみを搭載できます。
- C. QEMU 仮想マシンごとに、専用の物理 CPU コアを 1 つ予約する必要があります。
- D. QEMU は仮想 CPU の概念を使用して、仮想マシンを物理 CPU にマッピングします。
- E. QEMU 仮想マシンは、SMP システムを実行するために複数の仮想 CPU をサポートします。

正解: ([正解を表示します](#))

The CPU architecture of a QEMU virtual machine is independent of the host system's architecture. QEMU can emulate many CPU architectures, including x86, ARM, Alpha, and SPARC, regardless of the host system's architecture¹. This allows QEMU to run guest operating systems that are not compatible with the host system's hardware. Therefore, option A is correct. QEMU virtual machines support multiple virtual CPUs in order to run SMP systems. QEMU uses the concept of virtual CPUs (vCPUs) to map the virtual machines to physical CPUs. Each vCPU is a thread that runs on a physical CPU core. QEMU allows the user to specify the number of vCPUs and the CPU model for each virtual machine. QEMU can run SMP systems with multiple vCPUs, as well as single-processor systems with one vCPU². Therefore, option E is also correct. The other options are incorrect because they do not describe the CPU of a QEMU virtual machine. Option B is wrong because QEMU virtual machines can have more than one CPU with more than one core. Option C is wrong because QEMU does not require a dedicated physical CPU core for each virtual machine. QEMU can share the physical CPU cores among multiple virtual machines, depending on the load and the scheduling policy. Option D is wrong because QEMU does not use the term CPU, but vCPU, to refer to the virtual machines' processors.

References:

- * QEMU vs VirtualBox: What's the difference? - LinuxConfig.org
- * QEMU / KVM CPU model configuration - QEMU documentation
- * Introduction - QEMU documentation

質問: 35

次のうち仮想化の利点はどれですか？

- A. 上記のすべて
- B. セキュリティの強化
- C. リソース使用率の低下
- D. 簡素化されたハードウェア管理

正解: ([正解を表示します](#))

One of the core benefits of virtualization is simplified hardware management. Virtualization abstracts physical hardware into software-defined resources, making it easier to manage, allocate, migrate, and maintain systems.

Option C is incorrect because virtualization increases, not decreases, resource utilization. Since one option is incorrect, "All of the above" cannot be correct. While improved security is also a benefit, the most accurate single answer in this context is simplified hardware management. Therefore, the correct answer is D.

質問: 36

AMD CPU のハードウェア仮想化機能を示す CPU フラグはどれですか？

- A. HVM
- B. VIRT
- C. SVM
- D. PVM
- E. VMX

正解: ([正解を表示します](#))

The CPU flag that indicates the hardware virtualization capability on an AMD CPU is SVM. SVM stands for Secure Virtual Machine, and it is a feature of AMD processors that enables the CPU to run virtual machines with hardware assistance. SVM is also known as AMD-V, which is AMD's brand name for its virtualization technology. SVM allows the CPU to support a hypervisor, which is a software layer that creates and manages virtual machines. A hypervisor can run multiple virtual machines on a single physical machine, each with its own operating system and applications. SVM improves the performance and security of virtual machines by allowing the CPU to directly execute privileged instructions and handle memory access, instead of relying on software emulation or binary translation. SVM also provides nested virtualization, which is the ability to run a virtual machine inside another virtual machine. To use SVM, the CPU must support it and the BIOS must enable it. The user can check if the CPU supports SVM by looking for the svm flag in the /proc/cpuinfo file or by using the lscpu command. The user can also use the virt-host-validate command to verify if the CPU and the BIOS are properly configured for hardware virtualization¹²³. References:

* How to check if CPU supports hardware virtualization (VT technology)¹

* Processor support - KVM³

* How to Enable Virtualization in BIOS for Intel and AMD4

質問: 37

Xen 3.x で xl コマンドの前身として使用されていたコマンドはどれですか？

正解:

xm

Explanation:

In Xen 3.x, the primary command-line tool used to manage virtual machines was xm. According to Xen virtualization documentation, xm was part of the original Xen management stack and provided commands to create, start, stop, migrate, and monitor Xen domains.

As Xen evolved, the xm toolstack was deprecated and replaced by the xl toolstack. The xl command interfaces directly with the Xen hypervisor and offers better performance, improved maintainability, and reduced complexity. Modern Xen installations no longer support xm, but it remains historically significant as the predecessor to xl.

Therefore, the correct answer is xm.

質問: 38

virsh 内のどのコマンドが現在のホストで使用可能な仮想マシンを一覧表示しますか？

A. list-all

B. list

C. list-vm

D. show

E. view

正解: ([正解を表示します](#))

The virsh command used to list all virtual machines, including both running and stopped domains, is virsh list --all. The option list-all directly corresponds to this functionality.

The plain list command only shows running domains. The other options are not valid virsh commands.

Therefore, the correct answer is A.

質問: 39

xl ツール スタックのグローバル構成ファイルと同じものは何ですか？

正解:

/etc/xen/xl.conf

Explanation:

The xl toolstack is the modern command-line management interface used for administering the Xen hypervisor

, replacing the older xm toolstack. According to official Xen virtualization documentation, the global configuration file for the xl toolstack is /etc/xen/xl.conf.

This configuration file defines default behaviors and policies for the xl management commands. It allows administrators to configure global settings such as default CPU scheduling behavior,

memory handling, device model settings, and other operational parameters that affect how virtual machines (domains) are managed on a Xen host. These settings apply system-wide and influence the behavior of all xl commands unless overridden by per-domain configuration files. The file /etc/xen/xl.conf is read by the xl toolstack at runtime and is part of the standard Xen configuration directory structure. Individual virtual machine definitions are typically stored as separate configuration files in

/etc/xen/, but xl.conf serves as the central, global configuration reference.

Virtualization notes emphasize that proper configuration of xl.conf is important for performance tuning, stability, and compliance in Xen-based environments. While many default installations function correctly without modifying this file, enterprise deployments often customize it to align with workload requirements and hardware capabilities.

Therefore, the correct and documented global configuration file for the xl toolstack is /etc/xen/xl.conf.

質問: 40

次の出力 of xl list に関して正しいのは次のどれですか。

Name	ID	Mem	VCPUs	State	Time (s)
Domain-0	0	1024	1	r-----	498.7
Debian	2	1024	1	--p---	783.5
Ubuntu	6	1024	1	-b----	313.6
CentOS	7	2048	2	r-----	455.1

- A. Debian と Ubuntu の両方で、実行を開始するには xl コマンドが必要です。
- B. ID 2 のドメインはパラ仮想化を使用します。
- C. CentOS は、CPU 時間を最も多く消費したドメインです。
- D. Ubuntu はアイドル状態であるか、I/O を待機しています。
- E. Ubuntu の状態を実行状態にするには、xl コマンドを使用する必要があります。

正解: [\(正解を表示します\)](#)

The output of xl list shows the state of the domains. The domain with ID 6, Ubuntu, has a state of "b-". This means that the domain is blocked, which means it is idle or waiting for I/O.

<https://xenbits.xen.org/docs/unstable/man/xl.1.html>

質問: 41

次のコマンドを使用してデータ コンテナをセットアップした後:

```
docker create -v /data --name datastore debian /bin/true
```

/datavolume をデータストア コンテナと共有する追加の新しいコンテナはどのように起動されますか?

- A. docker run --share-with datastore --name service debian bash
- B. docker run -v datastore:/data --name service debian bash

- C. `docker run --volumes-from datastore --name service debian bash`
- D. `docker run -v /data --name service debian bash`
- E. `docker run --volume-backend datastore -v /data --name service debian bash`

正解: ([正解を表示します](#))

The correct way to start a new container that shares the `/data` volume with the `datastore` container is to use the

`--volumes-from` flag. This flag mounts all the defined volumes from the referenced containers. In this case, the `datastore` container has a volume named `/data`, which is mounted in the `service` container at the same path.

The other options are incorrect because they either use invalid flags, such as `--share-with` or `--volume-backend`, or they create new volumes instead of sharing the existing one, such as `-v datastore:/data` or `-v /data`.

References:

- * Docker Docs - Volumes
- * Stack Overflow - How to map volume paths using Docker's `--volumes-from`?
- * Docker Docs - `docker run`

質問: 42

QEMU は主に何に使用されますか？

- A. 仮想マシンの管理
- B. 仮想化環境の監視
- C. 仮想ディスクイメージの作成
- D. 仮想マシンを実行するためのハードウェアのエミュレーション

正解: ([正解を表示します](#))

QEMU is primarily used for hardware emulation, enabling virtual machines to run by emulating CPU, memory, storage, and peripheral devices. Virtualization documentation describes QEMU as a generic and open-source machine emulator and virtualizer. When used alone, QEMU can fully emulate hardware, allowing operating systems compiled for one architecture to run on another.

In modern Linux virtualization environments, QEMU is most commonly paired with KVM (Kernel-based Virtual Machine). In this configuration, KVM provides CPU virtualization using hardware extensions, while QEMU supplies device emulation and virtual hardware components. This division of responsibility allows near-native performance while maintaining flexibility.

Although QEMU includes utilities such as `qemu-img` for disk image creation, that is not its primary purpose.

VM management is handled by tools like `libvirt`, `virsh`, and `virt-manager`, and monitoring is performed by separate observability tools.

Virtualization notes consistently emphasize that QEMU's core function is hardware emulation, making option D the correct answer.

質問: 43

完全仮想化について正しい記述はどれですか? (正しい回答を 3 つ選択してください)

- A. 完全仮想化は準仮想化よりも高速です
- B. 完全な仮想化ではゲストOSカーネルを変更する必要はありません
- C. 完全な仮想化はCPUエミュレーションを通じて機能します
- D. 完全仮想化は、エミュレートされたデバイスドライバを通じて優れたI/Oパフォーマンスを実現します。
- E. 完全な仮想化には、エミュレーションのための時間とシステムリソースが必要です。

正解: [\(正解を表示します\)](#)

Full virtualization allows unmodified guest operating systems to run in virtual machines, making statement B correct. The hypervisor presents a complete hardware abstraction to the guest OS. Historically and conceptually, full virtualization relies on CPU emulation or hardware-assisted virtualization, making statement C correct. Even with hardware extensions, some degree of emulation is involved.

Full virtualization also requires additional system resources and processing time to emulate hardware and manage isolation, making statement E correct.

Statement A is incorrect because paravirtualization often offers better performance. Statement D is incorrect because emulated I/O devices generally perform worse than paravirtualized drivers. Therefore, the correct answers are B, C, and E.

質問: 44

次のうち有効な KVM パラメータはどれですか? (正しい回答を 3 つ選択してください。)

- A. drive file=imap://user:password@mailserver:/INBOX/Wufnc6MjYp@mailgate
- B. -drive file=ssh://user@host/tmp/file.img
- C. -drive file=iscsi://user%password@fileservers/iqn.2001-04.com.example/1
- D. -drive

file=http://user:password@fileservers/pub/linux.iso,media=cdrom,readonly

- E. -drive file=rsync://user%password@fileservers:/tmp/file.iso,media=cdrom,readonly

正解: **B,C,D** ([コメントを发表する](#))

KVM uses QEMU as its userspace component, and QEMU supports a variety of network-based block device backends through the -drive file= parameter. According to QEMU and KVM documentation, valid protocols include SSH, iSCSI, and HTTP/HTTPS.

Option B is valid because QEMU supports accessing disk images securely over SSH.

Option C is valid as iSCSI is a supported storage backend for QEMU block devices.

Option D is valid because QEMU can boot or attach ISO images over HTTP, commonly used for CD-ROM media in installation scenarios.

Option A is invalid because IMAP is not a supported QEMU block protocol.

Option E is invalid because rsync is not supported as a block device transport.

Therefore, the correct answers are B, C, and D.

質問: 45

Vagrant はどの仮想化プラットフォームで使用できますか? (該当するものをすべて選択してください)

- A. ドッカー
- B. VMware
- C. バーチャルボックス
- D. ハウズ

正解: ([正解を表示します](#))

Vagrant supports multiple providers, allowing it to work with different virtualization and container platforms.

According to official Vagrant documentation, supported providers include VirtualBox, VMware, and Docker.

VirtualBox is the default and most commonly used provider. VMware is supported through official plugins.

Docker is also supported as a provider, enabling lightweight container-based development environments.

Option D ("Haws") is an apparent typographical error and does not correspond to a valid virtualization platform. While Vagrant can integrate with AWS via community or third-party plugins, the option as written is invalid.

Therefore, the correct answers are A, B, and C.

質問: 46

コマンド `vzctl _____ 105 /usr/bin/apt-get install wget` では、OpenVZ コンテナ 105 に `wget` をインストールするために、`vzctl` のどのサブコマンドが不足していますか?

正解:

`exec`

Explanation:

In OpenVZ virtualization, the `vzctl` command is used to manage containers. According to OpenVZ documentation, the `exec` subcommand is required to execute a command inside a running container.

The correct command syntax is:

`vzctl exec 105 /usr/bin/apt-get install wget`

The `exec` subcommand runs the specified command in the context of the container identified by its container ID (CTID), in this case 105. Without `exec`, the command would not be executed inside the container environment.

Therefore, the missing and correct subcommand is `exec`.

有効的な**305-300J**問題集はJPNTest.com提供され、**305-300J**試験に合格することに役に立ちます！JPNTest.comは今最新**305-300J**試験問題集を提供します。JPNTest.com 305-300J試験問題集はもう更新されました。ここで**305-300J**問題集のテストエンジンを手に入れます。最新

質問: 47

同じ Linux ベースのホスト システム上で実行されている複数の仮想マシン間でデータを共有するにはどうすればよいですか？

- A. 同じホスト システム上のすべての仮想マシンが同じファイル システムを使用するため、データをファイル システムに書き込むことによって行われます。
- B. 他の仮想マシンのファイル システムを /dev/virt-disks/remote/ からマウントする。
- C. 1 つの仮想マシンに RAM ディスクをセットアップし、その UUID を使用して他の VM にマウントします。
- D. ネットワーク ファイル システムまたはファイル転送プロトコルを使用します。
- E. 同じ仮想ハードディスクをすべての仮想マシンに接続し、その仮想マシン上で EXT4 共有拡張機能をアクティブ化します。

正解: ([正解を表示します](#))

The correct way to share data between several virtual machines running on the same Linux-based host system is by using a network file system or file transfer protocol. A network file system (NFS) is a distributed file system protocol that allows a user on a client computer to access files over a network in a manner similar to how local storage is accessed¹. A file transfer protocol (FTP) is a standard network protocol used for the transfer of computer files between a client and server on a computer network². Both methods allow data to be shared between virtual machines regardless of their underlying file systems or virtualization technologies. The other options are incorrect because they either do not work or are not feasible. Option A is wrong because each virtual machine has its own file system that is not directly accessible by other virtual machines. Option B is wrong because there is no such device as /dev/virt-disks/remote/ that can be used to mount other virtual machines' file systems. Option C is wrong because a ramdisk is a volatile storage device that is not suitable for sharing data between virtual machines. Option E is wrong because attaching the same virtual hard disk to multiple virtual machines can cause data corruption and conflicts, and EXT4 does not have any sharing extensions that can prevent this. References: <https://kb.vmware.com/s/article/1012706>
<https://bing.com/search?q=data+sharing+between+virtual+machines>

質問: 48

準仮想化デバイス ドライバーはどこにインストールされますか？

- A. ゲストOS内
- B. 準仮想化には特別なドライバーは必要ありません
- C. ハイパーバイザーにコンパイルされる
- D. ホストOS内

正解: A ([コメントを发表する](#))

Paravirtualization is a virtualization technique where the guest operating system is aware that it is running in a virtualized environment and cooperates directly with the hypervisor. According to Xen and virtualization documentation, this cooperation is achieved through paravirtualized device drivers, which must be installed inside the guest operating system.

These specialized drivers replace traditional hardware drivers and communicate directly with the hypervisor using optimized interfaces. This reduces the overhead of hardware emulation and improves performance, particularly for disk and network I/O. Because of this design, the guest OS must either be modified or explicitly include support for paravirtualized drivers.

Option B is incorrect because paravirtualization explicitly requires special guest-side drivers.

Option C is incorrect because drivers are not compiled into the hypervisor; they reside in the guest. Option D is also incorrect because host OS drivers do not provide paravirtualized functionality to guests.

Virtualization notes consistently state that paravirtualized drivers are installed and run within the guest OS, making option A the correct answer.

質問: 49

Dockerfile 内の次のステートメントは、Hello World を出力するコンテナにつながりますか? (2つお選びください。)

- A. ENTRYPOINT "echo Hello World"
- B. ENTRYPOINT ["echo hello world"]
- C. ENTRYPOINT ["echo", "hello", "world"]
- D. ENTRYPOINT echo Hello World
- E. ENTRYPOINT "echo", "Hello", "World"

正解: ([正解を表示します](#))

The ENTRYPOINT instruction in a Dockerfile specifies the default command to run when a container is started from the image. The ENTRYPOINT instruction can be written in two forms: exec form and shell form. The exec form uses a JSON array to specify the command and its arguments, such as ["executable",

"param1", "param2"]. The shell form uses a single string to specify the command and its arguments, such as

"executable param1 param2". The shell form is converted to the exec form by adding /bin/sh -c to the beginning of the command. Therefore, the following statements in a Dockerfile are equivalent and will lead to a container that outputs hello world:

```
ENTRYPOINT [ "echo hello world" ] ENTRYPOINT [ "/bin/sh", "-c", "echo hello world" ]
```

```
ENTRYPOINT
```

```
"echo hello world" ENTRYPOINT [ "echo", "hello", "world" ] ENTRYPOINT [ "/bin/sh", "-c", "echo", "hello", "world" ] ENTRYPOINT "echo hello world"
```

The other statements in the question are invalid or incorrect. The statement A. ENTRYPOINT "echo Hello World" is invalid because it uses double quotes to enclose the entire command, which is not allowed in the shell form. The statement D. ENTRYPOINT echo Hello World is

incorrect because it does not use quotes to enclose the command, which is required in the shell form. The statement E. ENTRYPOINT "echo", "Hello", "World" is invalid because it uses double quotes to separate the command and its arguments, which is not allowed in the exec form. References:

- * Dockerfile reference | Docker Docs
- * Using the Dockerfile ENTRYPOINT and CMD Instructions - ATA Learning
- * Difference Between run, cmd and entrypoint in a Dockerfile

質問: 50

IaaS クラウドで、オペレーティング システムとソフトウェアを使用して新しいコンピューティング インスタンスをプロビジョニングする一般的な方法は何ですか？

- A. 新しい各インスタンスは Linux ディストリビューションのインストール メディアに接続され、SSH 経由でログインすることによってインストーラーへのアクセスを提供します。
- B. 新しいインスタンスはそれぞれ、オペレーティング システム、ソフトウェア、および特定の目的のデフォルト構成を含むイメージ ファイルに基づいて作成されます。
- C. 新しいインスタンスはそれぞれ、現在実行中の別のインスタンスのクローンであり、元のインスタンスのすべてのソフトウェア、データ、状態が含まれます。
- D. 新しい各インスタンスは、プロビジョニングを開始したコンピューターに VPN 経由で接続され、そのコンピューターから PXE ブートを試行します。
- E. 各新しいインスタンスには、管理者がターゲット オペレーティング システムを展開するベースとして、仮想 CD から実行される最小限のライブ システムが含まれています。

正解: [\(正解を表示します\)](#)

In an IaaS cloud, the most common method for provisioning new computing instances is to use an image file that contains a pre-installed operating system and software. This image file is also known as a machine image, a virtual appliance, or a template. The image file can be customized for a specific purpose, such as a web server, a database server, or a development environment. The image file can be stored in a repository or a library that is accessible by the cloud provider or the user. When a new instance is requested, the cloud provider copies the image file to a virtual disk and attaches it to the instance. The instance then boots from the virtual disk and runs the operating system and software from the image file. This method is faster and more efficient than installing the operating system and software from scratch for each new instance. It also ensures consistency and reliability across multiple instances that use the same image file. References:

- * LPI Virtualization and Containerization Exam Objectives, Topic 305.1: Virtualization Concepts and Theory, Objective: Describe the concept of machine images and templates
- * LPI Virtualization and Containerization Study Guide, Chapter 1: Virtualization Concepts and Theory, Section: Machine Images and Templates
- * LPI LPIC-3 305 Certification Sample Questions and Practice Exam, Question 10: In an IaaS cloud, what is a common method for provisioning new computing instances with an operating system and software?

質問: 51

libvirt とは何ですか？

- A. 仮想化管理 API。
- B. Linuxディストリビューション
- C. 仮想化技術
- D. ハードウェア仮想化プラットフォーム

正解: **A** ([コメントを發表する](#))

Libvirt is a virtualization management API, toolkit, and set of command-line utilities designed to manage virtualization platforms in a consistent and secure manner. According to official virtualization and containerization documentation, libvirt provides a common interface for interacting with multiple hypervisors such as KVM, QEMU, Xen, VMware ESXi, and Hyper-V, abstracting their underlying differences.

Libvirt itself is not a virtualization technology or hypervisor; rather, it acts as a management layer between administrators or orchestration tools and the hypervisor. It enables tasks such as creating, starting, stopping, pausing, migrating, and monitoring virtual machines. Tools like virsh and virt-manager rely on libvirt to perform virtualization operations.

One of libvirt's key strengths is its focus on security and isolation. It integrates with Linux security frameworks such as SELinux and AppArmor, enforcing mandatory access controls on virtual machines.

Libvirt also supports remote management using secure protocols, making it suitable for enterprise and cloud environments.

In virtualization documentation, libvirt is described as a foundational component for Infrastructure as Code and automation, commonly used with tools like Ansible, OpenStack, and Kubernetes (via KVM). Its standardized API simplifies virtualization management while ensuring portability, scalability, and compliance with best practices.

質問: 52

仮想化の利点は次のどれですか？

- A. ハードウェア使用率の向上
- B. ソフトウェアの互換性の問題の軽減
- C. セキュリティの強化
- D. 簡素化されたバックアップとリカバリ

正解: **A,C,D** ([コメントを發表する](#))

Virtualization provides several well-documented benefits in modern IT environments. One major advantage is improved hardware utilization(A), as multiple virtual machines can run on a single physical system, maximizing resource usage. Virtualization also enhances security(C) through isolation between workloads, limiting the impact of failures or security breaches.

Another key benefit is simplified backup and recovery(D). Virtual machines are encapsulated into files, making it easier to snapshot, back up, clone, and restore entire systems. This significantly improves disaster recovery and testing workflows.

Option B is not universally true. While virtualization can help standardize environments, it does not inherently reduce software compatibility issues; in some cases, it may even introduce additional layers of complexity.

Virtualization documentation consistently highlights hardware efficiency, security isolation, and operational flexibility as core benefits, making A, C, and D the correct answers.

質問: 53

自動スケーリングや負荷分散によく関連付けられるコンテナ オーケストレーション プラットフォームはどれですか。

- A. Docker Compose
- B. Kubernetes
- C. Amazon ECS
- D. LXC

正解: **B** ([コメントを发表する](#))

Kubernetes is the container orchestration platform most strongly associated with automated scaling and load balancing. According to container orchestration documentation, Kubernetes provides built-in features such as Horizontal Pod Autoscaling, service-based load balancing, self-healing, and declarative deployment models.

Docker Compose and LXC are not orchestration platforms at scale, and while Amazon ECS supports scaling, Kubernetes is the industry-standard platform most commonly referenced for these capabilities.

Therefore, the correct answer is B.

質問: 54

次のコマンドのうち、ディスク イメージ vm1-snap.img と vm1.img の間のすべての違いをリストするコマンドはどれですか？

- A. virt-delta -a vm1-snap.img -A vm1.img
- B. virt-cp-in -a vm1-snap.img -A vm1.img
- C. virt-cmp -a vm1-snap.img -A vm1.img
- D. virt-history -a vm1-snap.img -A vm1.img
- E. virt-diff -a vm1-snap.img -A vm1.img

正解: **E** ([コメントを发表する](#))

The virt-diff command-line tool can be used to list the differences between files in two virtual machines or disk images. The output shows the changes to a virtual machine's disk images after it has been running. The command can also be used to show the difference between overlays¹. To specify two guests, you have to use the -a or -d option for the first guest, and the -A or -D option for the second guest. For example: virt-diff -a old.img -A new.img¹. Therefore, the correct command to list all differences between the disk images vm1-snap.img and vm1.img is: virt-diff -a vm1-snap.img -A vm1.img. The other commands are not related to finding differences between disk images. virt-delta is a tool to create delta disks from two disk images². virt-cp-in is a tool to copy files and directories into a virtual machine disk image³. virt-cmp is a tool to compare two

files or directories in a virtual machine disk image⁴. virt-history is a tool to show the history of a virtual machine disk image⁵. References:

- * 21.13. virt-diff: Listing the Differences between Virtual Machine Files ...
- * 21.14. virt-delta: Creating Delta Disks from Two Disk Images ...
- * 21.6. virt-cp-in: Copying Files and Directories into a Virtual Machine Disk Image ...
- * 21.7. virt-cmp: Comparing Two Files or Directories in a Virtual Machine Disk Image ...
- * 21.8. virt-history: Showing the History of a Virtual Machine Disk Image ...

質問: 55

パッカーとは何ですか？

- A. コンテナオーケストレーションツール
- B. 画像構築ツール
- C. 構成管理ツール
- D. 仮想化プラットフォーム

正解: [\(正解を表示します\)](#)

Packer is an image automation and building tool developed by HashiCorp. According to virtualization and DevOps documentation, Packer is used to create machine images for multiple platforms from a single configuration file.

Packer is not an orchestration platform, configuration management tool, or hypervisor. Its primary role is to ensure consistent, repeatable, and automated image creation, making B the correct answer.

質問: 56

Vagrant と Docker によって提供される機能はどれですか？ 3つお選びください。）

- A. どちらもホスト ファイル システムからゲストヘディレクトリを共有できます。
- B. どちらも、デフォルトでは仮想マシンではなくコンテナとしてシステム イメージを起動します。
- C. どちらも必要な基本イメージをダウンロードできます。
- D. どちらもベース イメージに変更を適用できます。
- E. どちらも、デフォルトではコンテナではなく仮想マシンとしてシステム イメージを起動します。

正解: [\(正解を表示します\)](#)

* Both Vagrant and Docker can share directories from the host file system to a guest. This allows the guest to access files and folders from the host without copying them. Vagrant uses the config.vm.

synced_folder option in the Vagrantfile to specify the shared folders¹. Docker uses the -v or --volume flag in the docker run command to mount a host directory as a data volume in the container².

* Both Vagrant and Docker can download required base images. Base images are the starting point for creating a guest environment. Vagrant uses the config.vm.box option in the Vagrantfile

to specify the base image to use¹. Docker uses the FROM instruction in the Dockerfile to specify the base image to use². Both Vagrant and Docker can download base images from public repositories or local sources.

* Both Vagrant and Docker can apply changes to a base image. Changes are modifications or additions to the base image that customize the guest environment. Vagrant uses provisioners to run scripts or commands on the guest after it is booted¹. Docker uses instructions in the Dockerfile to execute commands on the base image and create a new image². Both Vagrant and Docker can save the changes to a new image or discard them after the guest is destroyed.

* Vagrant and Docker differ in how they start system images. Vagrant starts system images as virtual machines by default, using a provider such as VirtualBox, VMware, or Hyper-V¹. Docker starts system images as containers by default, using the native containerization functionality on macOS, Linux, and Windows². Containers are generally more lightweight and faster than virtual machines, but less secure and flexible. References: 1: Vagrant vs. Docker | Vagrant | HashiCorp Developer 2: Vagrant vs Docker: Which Is Right for You? (Could Be Both) - Kinsta Web Development Tools

質問: 57

コンテナ仮想化のコンテキストにおける機能の目的は何ですか？

- A. 潜在的に危険なシステム コールを、コンテナ仮想化によって提供されるエミュレーション層にマッピングします。
- B. コンテナが消費できるディスク容量を制限します。
- C. 複数のコンテナに存在するファイルをキャッシュするためのメモリ重複排除を有効にします。
- D. 通常のユーザーが昇格された権限でコンテナを起動できるようにします。
- E. プロセスがコンテナを侵害する可能性のあるアクションを実行しないようにします。

正解: E ([コメントを发表する](#))

Capabilities are a way of implementing fine-grained access control in Linux. They are a set of flags that define the privileges that a process can have. By default, a process inherits the capabilities of its parent, but some capabilities can be dropped or added by the process itself or by the kernel. In the context of container virtualization, capabilities are used to prevent processes from performing actions that might infringe the container, such as accessing the host's devices, mounting filesystems, changing the system time, or killing other processes. Capabilities allow containers to run with a reduced set of privileges, enhancing the security and isolation of the container environment. For example, Docker uses a default set of capabilities that are granted to the processes running inside a container, and allows users to add or drop capabilities as needed¹².

References:

* Capabilities | Docker Documentation¹

* Linux Capabilities: Making Them Work in Containers²

質問: 58

次のコマンドのうち、実行中の LXC コンテナでコマンドを実行するのはどれですか？

- A. lxc-attach
- B. lxc-batch
- C. lxc-run
- D. lxc-enter
- E. lxc-eval

正解: ([正解を表示します](#))

The command `lxc-attach` is used to execute a command in a running LXC container. It allows the user to start a process inside the container and attach to its standard input, output, and error streams¹. For example, the command `lxc-attach -n mycontainer -- ls -lh /home` will list all the files and directories in the `/home` directory of the container named `mycontainer`¹. The other options are not valid LXC commands. The command `lxc-batch` does not exist. The command `lxc-run` is an alias for `lxc-start`, which is used to start a container, not to execute a command in it². The command `lxc-enter` is also an alias for `lxc-attach`, but it is deprecated and should not be used³. The command `lxc-eval` is also not a valid LXC command. References:

* 1: Executing a command inside a running LXC - Unix & Linux Stack Exchange.

* 2: `lxc-start`: start a container. - SysTutorials.

* 3: `lxc-attach`: start a process inside a running container. - SysTutorials.

質問: 59

次のコマンドを使用して新しい Docker ネットワークを作成した後、次の手順を実行します。

```
docker network create --driver Bridge isolation_nw
```

コンテナをネットワークに接続するには、`docker create` にどのパラメータを追加する必要がありますか？

- A. `--eth0=isolated_nw`
- B. `--alias=isolated_nw`
- C. `--ethernet=isolated_nw`
- D. `--network=isolated_nw`
- E. `--attach=isolated_nw`

正解: ([正解を表示します](#))

To attach a container to a network when creating it, the `--network` flag must be used with the name of the network as the argument. The `--network` flag specifies the network mode for the container. By default, the network mode is `bridge`, which means the container is connected to the default bridge network. However, if a custom network is created, such as `isolation_nw` in this case, the container must be explicitly attached to it using the `--network` flag. For example, to create a container named `web1` and attach it to the `isolation_nw` network, the command would be:

```
docker create --name web1 --network isolation_nw nginx
```

The other options are not valid parameters for `docker create`. The `--eth0`, `--ethernet`, and `--attach` flags do not exist. The `--alias` flag is used to specify an additional network alias for the container on a user-defined network, but it does not attach the container to the network. References:

- * docker network create | Docker Documentation1
- * docker create | Docker Documentation
- * Networking overview | Docker Docs2

有効的な**305-300J**問題集はJPNTTest.com提供され、**305-300J**試験に合格することに役に立ちます！JPNTTest.comは今最新**305-300J**試験問題集を提供します。JPNTTest.com 305-300J試験問題集はもう更新されました。ここで**305-300J**問題集のテストエンジンを手に入れます。最新版のアクセス、<https://www.jpntest.com/shiken/305-300J-mondaishu> **125問、30%ディスカ**
ウント、特別な割引コード: **JPNshiken**」