13.2.2 Virtual Machine Concepts: Part 2

Click one of the buttons to take you to that part of the video.

Virtual Machine Concepts: Part 2 0:00-0:12

In this video, we're going to talk about virtual machines. Specifically, we're going to cover bootstrapping techniques, virtual machine disk storage, and virtual machine management tools.

Bootstrapping 0:13-1:09

One challenge with creating templates is that all virtual machines created using the same template will have the same settings, including the same hostname and security identifiers, and sometimes even the same IP address. This problem can be solved with bootstrapping. Bootstrapping is the automated process of provisioning a virtual machine with unique settings and configurations. Bootstrapping customizes a virtual machine during installation (or, more specifically, when the virtual machine first boots) without requiring user input.

As a best practice, you should prepare your template by removing SSH host keys, machine identifiers, and any other configurations that need to be unique when you use the template to build a new VM. Then you can use any number of bootstrapping solutions. Two popular bootstrapping technologies that automate the provisioning of new Linux virtual machines are Cloud-init and a combination of Anaconda and Kickstart.

Cloud-init is implemented by installing the cloud-init application on your Linux template.

Cloud-init 1:04-2:29

You then customize the configuration by editing the /etc/cloud/cloud.cfg file. This file controls the cloud-init modules that are run. These modules fall under three configuration stages: the init stage, the config stage, and the final stage. If you don't want a module to run, you delete the line from the cloud.cfg file.

Once you're satisfied with your template, you can clone it to create a new VM.

There's one more step that you perform before starting up the cloned VM. This step is unique to the hypervisor. As the new VM boots, cloud-init takes over. By various means, it retrieves metadata that's used to set the hostname, set the default locale, and generate SSH host keys. Some hypervisors supply this metadata by creating and mounting a cloud-init optical drive. The important idea here is that a good hypervisor provides a user interface to configure this metadata for each new VM.

Finally, cloud-init is used by many cloud providers. If you require a Linux platform, your provider may have a cloud-init dashboard, where you use menu items to supply configurations such as a hostname and the default locale and add user data. When you click Launch, the proper virtual machine template is cloned, and the cloud-init agent customizes the VM according to your specifications. In this way, you can build a custom Linux machine in just a few minutes.

Anaconda and Kickstart 2:30-3:00

Anaconda is an installation program that's used by Fedora, Red Hat Enterprise Linux, and other distributions. It identifies the computer's hardware, creates a file system, and provides a user interface that guides the installation process. Anaconda installations can be scripted with kickstart for unattended installations.

Using kickstart, a Linux administrator creates a single file containing the answers to all the questions that would be asked during a typical installation. This file can be kept on a server and read by individual computers during the Linux installation.

Kickstart File 3:01-3:14

The kickstart file is a simple text file that contains keywords arranged in sections. The sections can be in any order and are not required. Here's a list of the kickstart sections. This list can give you an idea of the types of configurations you can automate with kickstart.

Virtual Machine Disk Storage 3:15-3:37

One of the configurations that can be made during virtual machine creation as well as during bootstrapping is adding virtual storage. A virtual disk is a file or set of files maintained by the host hypervisor that appears as a physical disk to the guest operating system. This makes the virtual machine's disk portable. The disk files can be moved between locations even if the location is on another host hypervisor.

Thick Provisioning 3:38-6:56

When you create a virtual disk, the space for the disk is allocated by the host hypervisor. You can choose to allocate or provision this space in one of two ways.

If you choose thick provisioning, the complete amount of storage capacity is pre-allocated on the hypervisor's physical storage device. For example, if the disk size is to be 10 GB, the full 10 GB plus any overhead is set aside for the virtual disk right from the start. This physical disk space will be unavailable for use by any other virtual machine.

Many administrators choose to use thin provisioning. A thin provisioned virtual disk consumes only the space that it needs initially, and then grows according to demand. The benefits of thin provisioning are that the disk is provisioned quickly and storage space is saved. However, the down side is that overprovisioning disks – assigning virtual disks more space than is physically available – causes the virtual machine to fail if the physical storage space becomes full. Fortunately, most hypervisors will inform you if a physical disk starts to fill up.

Another virtual disk setting offered by hypervisors controls what happens when the virtual machine is shut down. Disk volumes in persistent mode behave like conventional disk drives. All data that's written to the virtual disk is permanent so that it's available when the virtual machine starts again.

You can also configure a virtual disk to be non-persistent. Any data written to the disk is lost when the virtual machine is powered off. This is helpful if you want your virtual machine to always start in the same state for software testing or for doing demonstrations. You keep your virtual disk in persistent mode until your virtual machine is customized properly. Then you shut down the virtual machine and switch the virtual disk to non-persistent mode. Afterwards, every time the virtual machine is shut down, the virtual disk reverts to the state it was in when non-persistent mode was set.

There are three popular toolsets used by Linux-based hypervisors to manage virtual machines: libvirt, virsh, and vmm.

Strictly speaking, libvirt is an open-source application programming interface (API) that's used for creating, monitoring, migrating, starting, and stopping virtual machines. If you're a software developer, you can use the functions and routines provided by the libvirt API to interface with a hypervisor, be it Xen, KVM, or QEMU, to create and control virtual machines. As a Linux administrator, you most likely won't be involved in creating virtualization applications, but you should be aware that many virtualization applications require libvirt and a hypervisor to work properly.

While libvirt's main role is as an API, it also provides a command line tool named virsh for controlling virtualization. The virsh tool has many arguments that are, essentially, commands in and of themselves. For example, the virtsh list command displays the virtual machines running on the hypervisor. The virsh start command starts a virtual machine. And the virsh shutdown command cleanly shuts down a virtual machine.

Virtual Machine Manager is a popular graphical tool for managing virtual machines on a Linux host. It's also known as virt-manager since the graphical interface can be started using the virt-manager command. It also comes with command line tools like virt-install, which is used to provision operating sytems into VMs; virt-clone, which clones existing VMs; and virt-convert, which converts OVF VMs to run with libvert.

Summary 6:57-7:05

So, let's review what we discussed. In this video, we talked about bootstrapping techniques and virtual machine disk storage. We finished by describing a three virtual machine management tools: libvert, virsh and virt-manager.