CernVM Virtual Machine Documentation

Release 4.5

The CernVM Team

Jan 27, 2021

Contents

CHAPTER 1

What is the CernVM Virtual Machine?

CernVM is a virtual machine image based on CentOS combined with a custom, virtualization-friendly Linux kernel.

CernVM is based on the µCernVM bootloader. Its outstanding feature is that it does not require a hard disk image to be distributed (hence "micro"). Instead it is distributed as a read-only image of ~20MB containing a Linux kernel and the CernVM-FS client. The rest of the operating system is downloaded and cached on demand by CernVM-FS. The virtual machine still requires a hard disk as a persistent cache, but this hard disk is initially empty and can be created instantaneously.

CHAPTER 2

Contents

2.1 Release Notes

2.1.1 CernVM 4.5 [27.01.2021]

CernVM 4.5 is a minor update to CernVM 4. It upgrades the base system to EL 7.9. It also changes the ssh daemon default config in order to allow for public key authenticated login to the VM.

In order to upgrade, run `sudo cernvm-update -a` followed by a reboot.

2.1.2 CernVM 4.4 [17.06.2020]

CernVM 4.4 is a minor update to CernVM 4. It upgrades the base system to EL 7.8 and it updates the certificate chain to fix contextualization with CernVM Online.

Note: if your bootloader is older than version 2020.04-1, you need to run

```
sudo cp /etc/cvmfs/keys/cern.ch/cern-it4.cern.ch.pub /mnt/.rw/aux/keys/cernvm-prod.
↪cern.ch
sudo cernvm-update -k
```

before being able to update with `sudo cernvm-update -a`.

2.1.3 CernVM 4.3 [27.01.2020]

CernVM 4.3 is a minor update to CernVM 4.2. It upgrades the base system to EL 7.7 and provides new versions of the docker and eos packages. Note that in the course of the EOS update, that eosd daemon disappeared. Instead, eos mounts are now autofs managed, in the same way they are on CERN lxplus.

2.1.4 CernVM 4.2 [04.02.2019]

CernVM 4.2 is a minor update to CernVM 4.1, upgrading the base system to EL 7.6.

2.1.5 CernVM 4.1 [12.06.2018]

Installation

Download the latest images from our download page. The image is generic and available in different file formats for different hypervisors. The image needs to be contextualized to become either a graphical development VM or a batch virtual machine for the cloud.

Detailed instructions are available for *VirtualBox*, *KVM*, *Docker*, *OpenStack*, *Amazon EC2*, *Google Compute Engine*, and *Microsoft Azure*.

Contextualization

In most cases, a CernVM needs to be contextualized on first boot. The process of contextualization assigns a profile to the particular CernVM instance. For instance, a CernVM can have the profile of a graphical VM used for development on a laptop; applying another context let the CernVM become a worker node in the cloud.

The *CernVM Launcher* allows for instantiating CernVMs from text-based contexts, such as our public demo contexts.

Please find details on the various contextualiztion options on the *contextualization page*.

Updates and Version Numbers

When booted, CernVM will load the latest available CernVM version and pin itself on this version. This ensures that your environment stays the same unless you explicitly take action. Both the µCernVM bootloader and the CernVM-FS provided operating system can be updated using the `cernvm-update` script. CernVM machines show an update notification in /etc/motd and in the GUI. The support list will be notified when updates are ready and will post specific instructions for each update.

The CernVM 4 strong version number consists of 4 parts: 4.X.Y.Z. Major version 4 indicates an Scientific Linux 7 based CernVM. Minor version X will be changed when there is a significant change in the set of supported features. "Y" is the bugfix version. "Z" is the security hotfix version; changes in "Z" don't change the set of packages but provide security fixes for selected packages.

Next Steps

Once booted and contextualized, you can use ssh to connect to your virtual machine. SSHFS and shared folders provide you an easy means to exchange files between the host and CernVM.

For storing data and analysis results, we recommend not to use the root partition. Instead, attach a second hard drive to the virtual machine or use shared folders. This way, you can move data between virtual machines and the data remains intact even in case the virtual machine ends up in an unsuable state.

Single Sign On

You can get a Kerberos token with `kinit`. With the token, you can login to lxplus and work with subversion repositories without the need to provide a password.

Swap Space

By default, CernVM has no swap space enabled. The following commands creates a 2G swap file

```
sudo fallocate -l 2G /mnt/.rw/swapfile
sudo chmod 0600 /mnt/.rw/swapfile
sudo mkswap /mnt/.rw/swapfile
sudo swapon /mnt/.rw/swapfile
```

If a file /mnt/.rw/swapfile exists, it will picked up automatically on boot as a swap space. In order to activate a swap space through contextualization, add to your amiconfig user data

```
[cernvm]
swap_size=<SIZE>
```

where <SIZE> can be anything understood by `fallocate -l` or it can be `auto`, in which case CernVM uses 2G/core.

Resizing the Root Partition

If you increase your virtual hard drive, you can have CernVM increase your root partition accordingly. To do so, run

```
sudo touch /mnt/.rw/aux/resize
```

and reboot. Resizing the root partition is a delicate operation, please **make a VM snapshot before you proceed**.

Debugging

In case you cannot login (any more) to your virtual machine, even though the machine was properly contextualized, you can boot CernVM in "debug mode". In the early boot menu, select the "Debug" entry. This enables kernel debug messages and pauses the boot process just before the µCernVM bootloader hands over to the operating system. Here, type `reset_root_password` followed by `ENTER` and `Ctrl+D`. Once booted, you can then login as root with password "password".

2.2 How to run on...

This section describes how to instantiate CernVM appliances on various local hypervisors and clouds.

2.2.1 VirtualBox

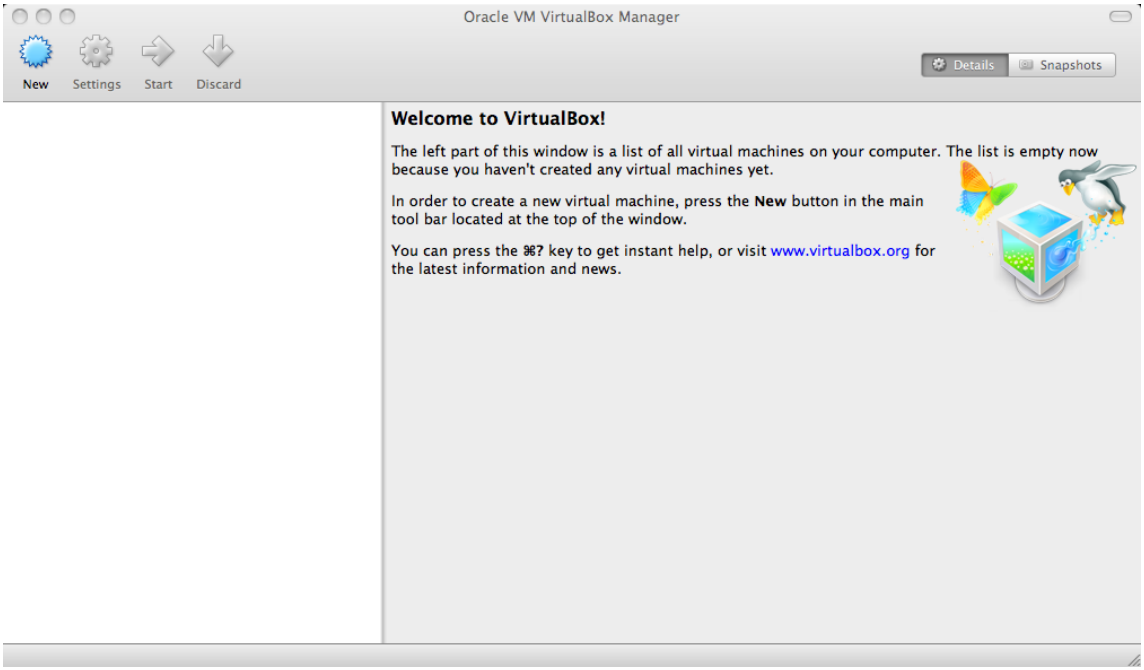
CernVM Launch

The recommended way to start a CernVM with VirtualBox is using the *CernVM Launch* utility.

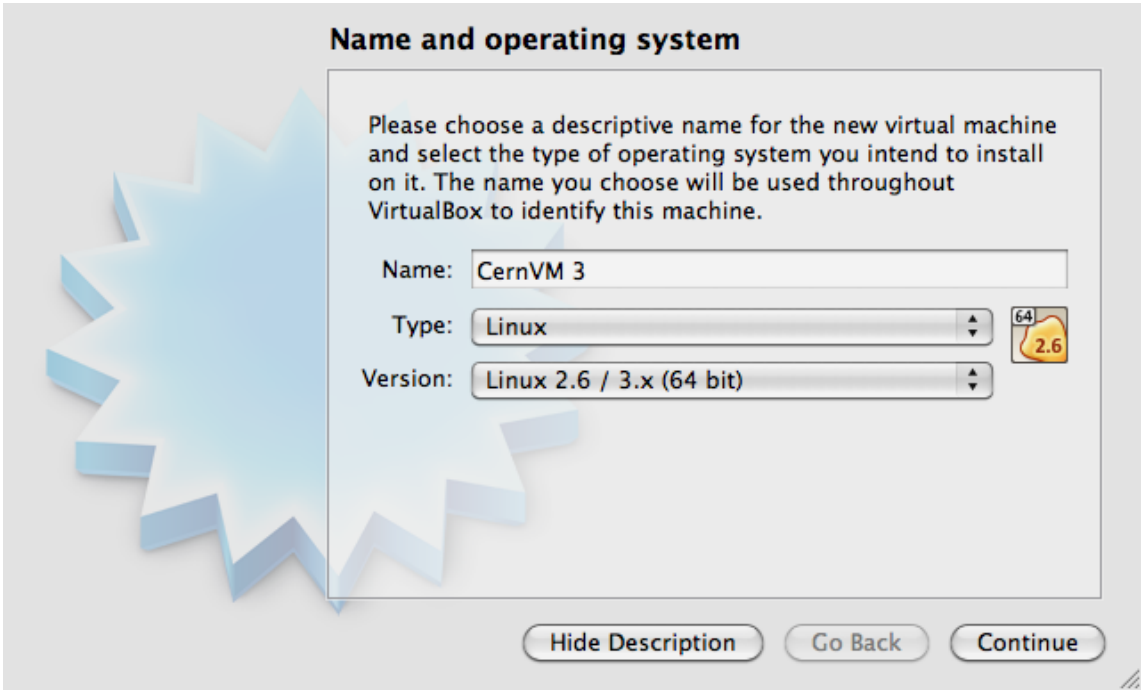
Manual Instantiation

To configure your VirtualBox for running CernVM do the following steps:

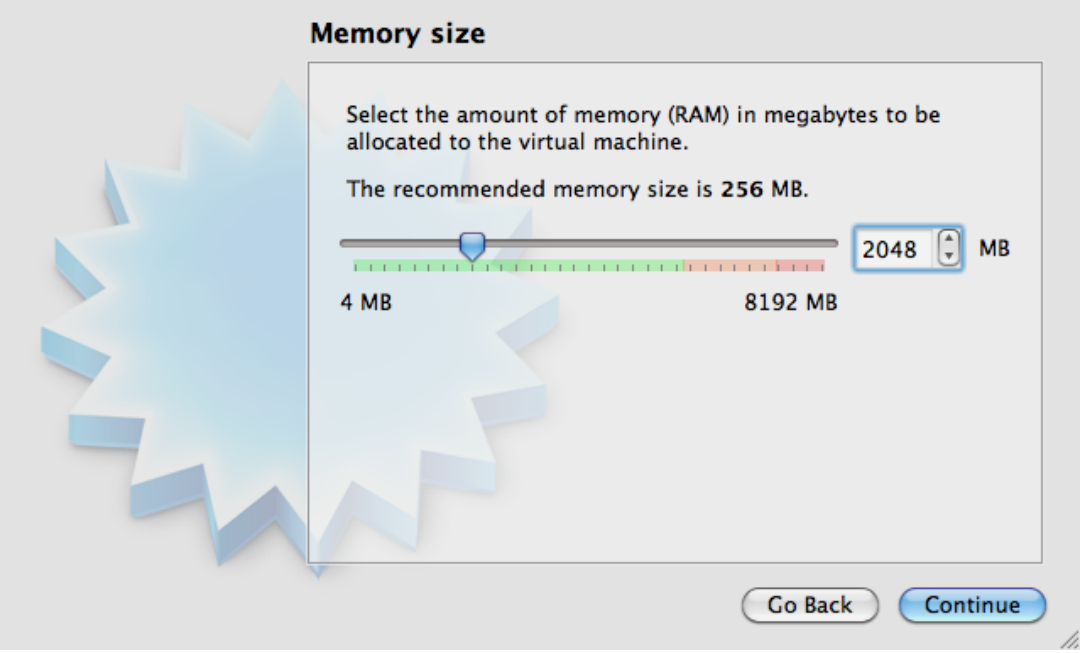
- Start VirtualBox



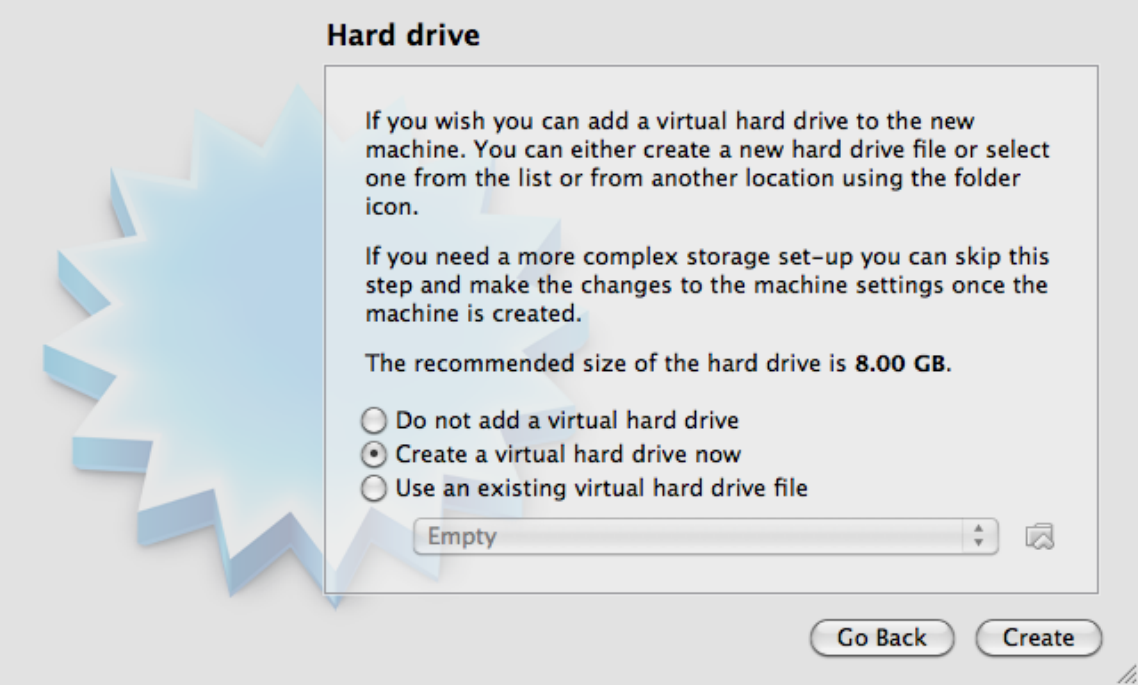
- Click the "**New**" button and a "Create New Virtual Machine" dialogue will appear



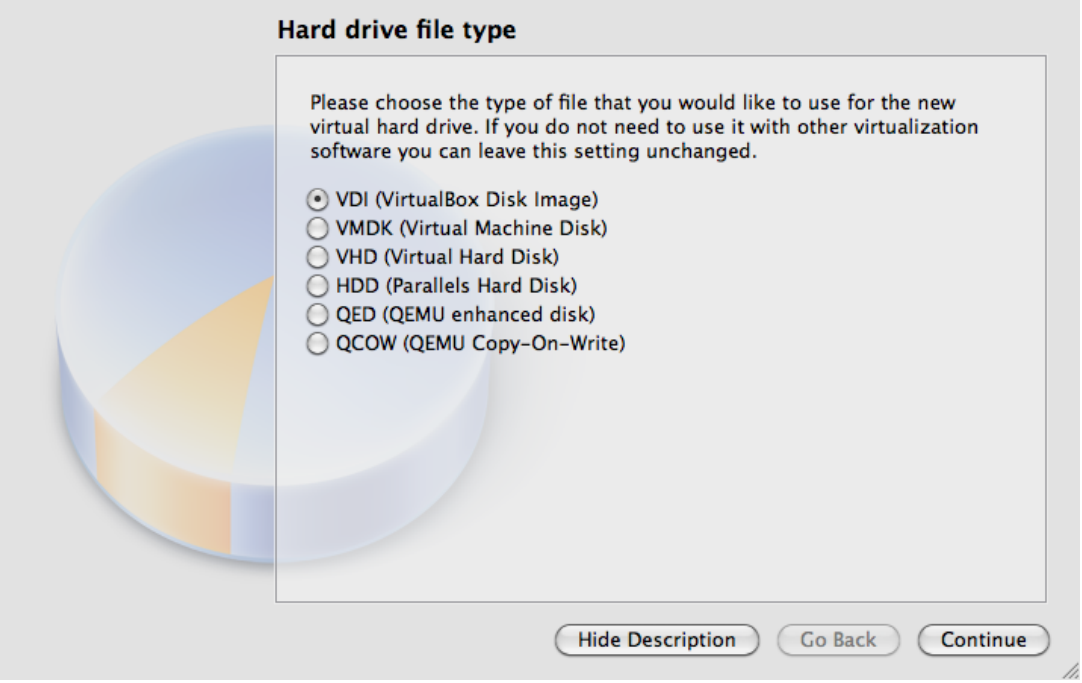
- Enter the name of the virtual machine (e.g. "CernVM 4") and define the OS Type:
  - As the "**Type**" chose Linux
  - As the "**Version**" chose "Linux 2.6 / 3.x / 4.x (64 bit)"
  - Click "**Continue**"



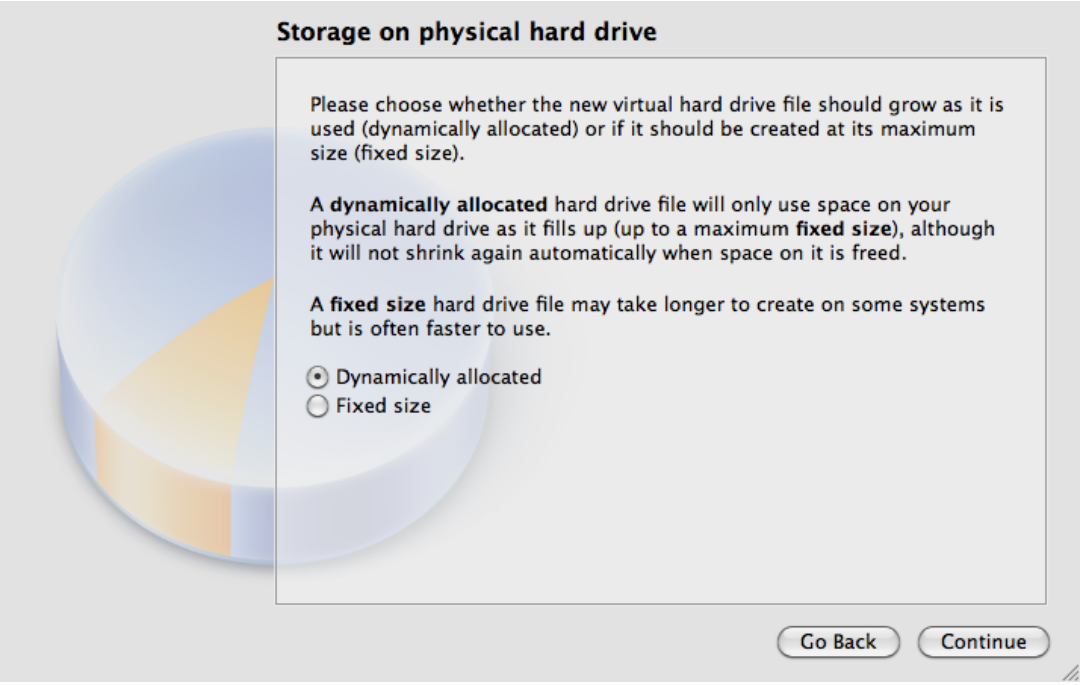
- Define the amount of memory which you want to make available to the Virtual Machine (e.g. 2048 MB) and click “”**Continue**”. Choose at least 1G of memory and leave at least 1G of memory free for your host.



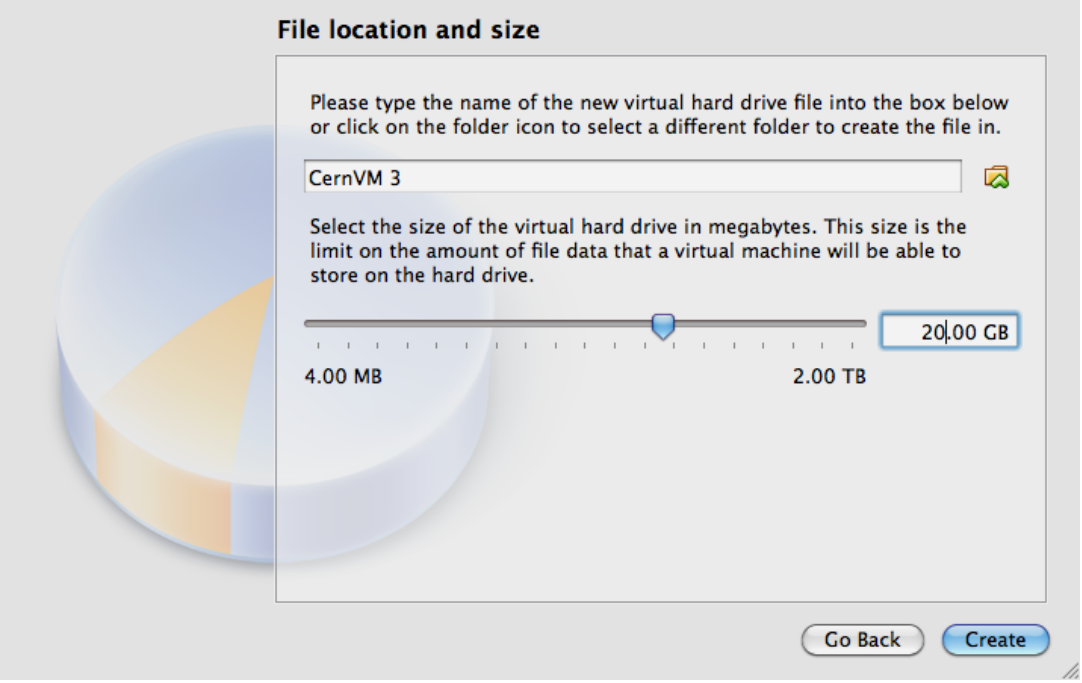
- Configure the Virtual Machine hard drive.
  - Check the “**Create a virtual hard drive now**” option
  - As a hard drive file type, select “**VDI (VirtualBox Disk Image)**”



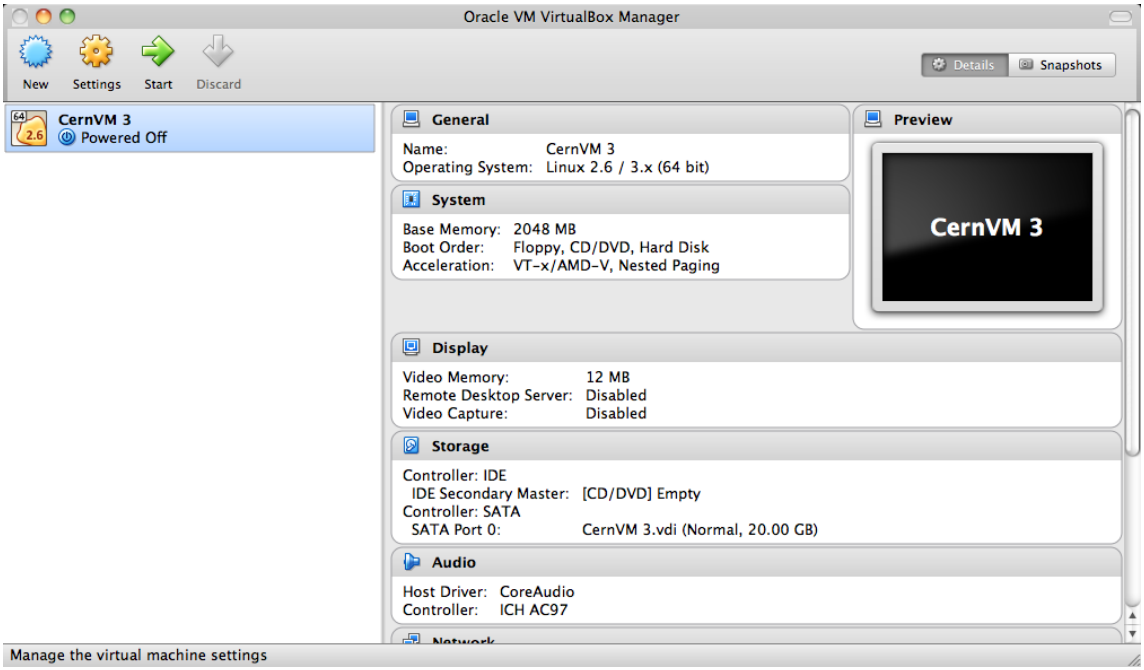
– For storage on physical hard drive, select "**Dynamically allocated**"



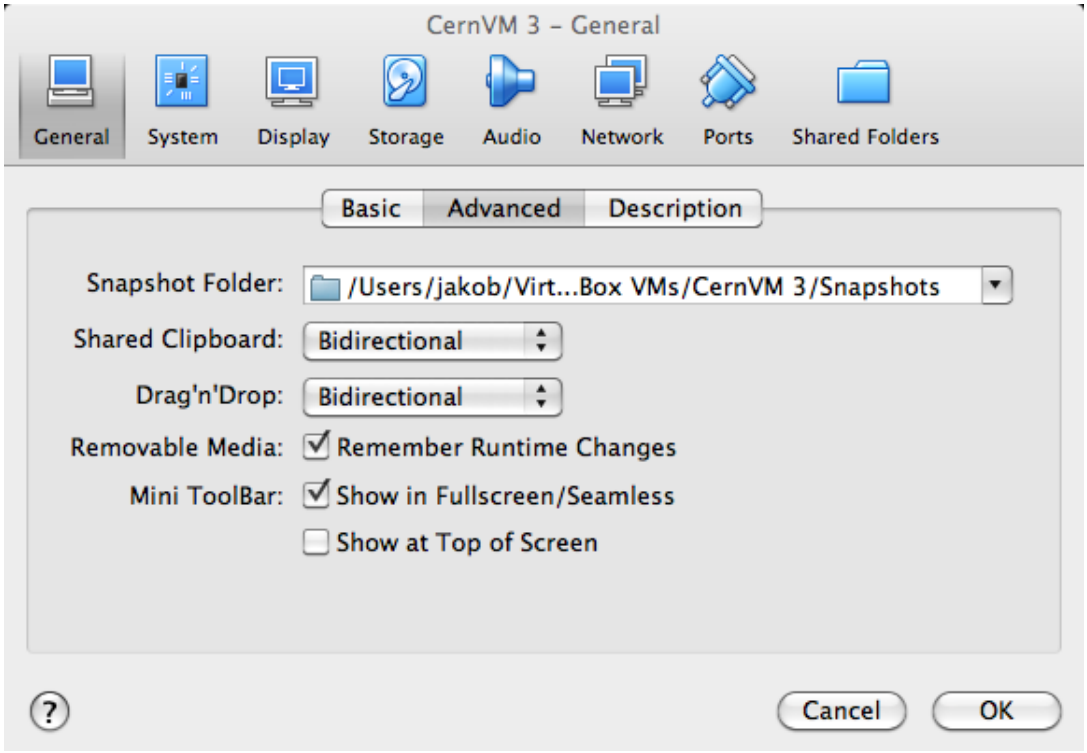
– Under file location and size, select the folder and file name of the virtual machine hard disk image and the maximum size. Select at least 20G for the size.



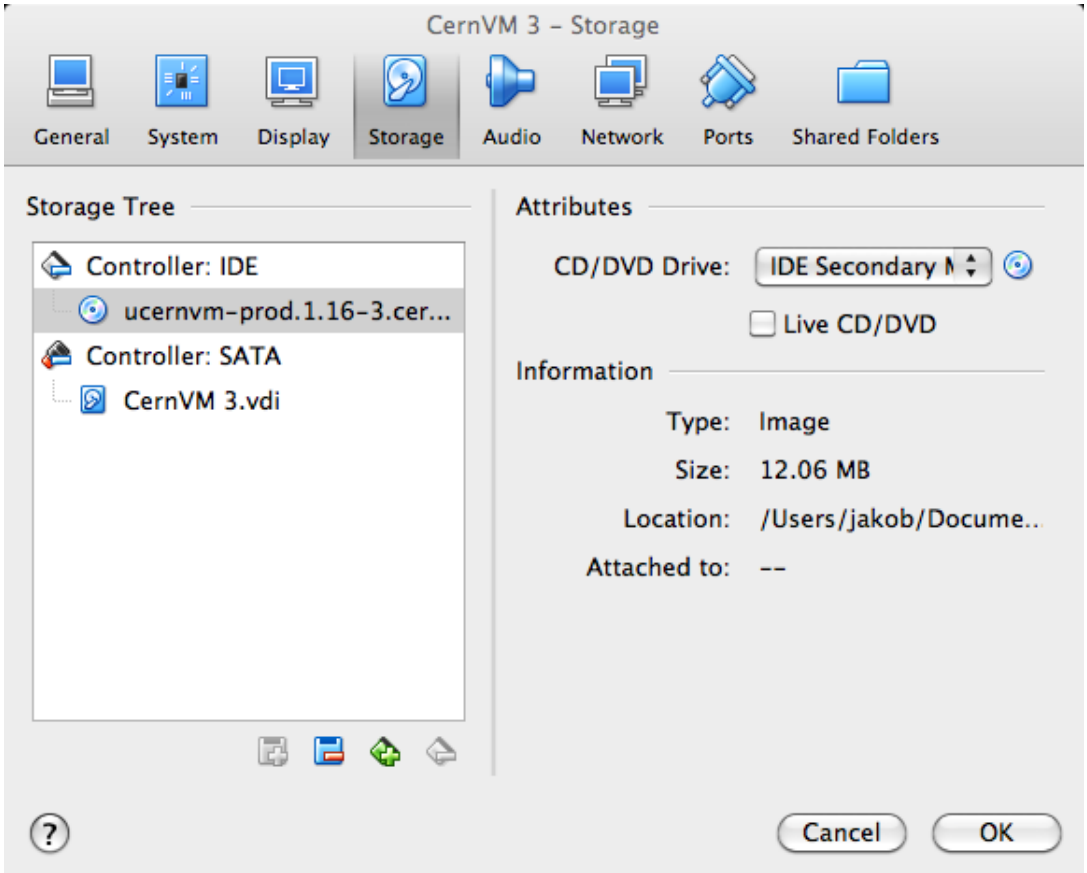
- Your new virtual machine will appear in the pane on the left.



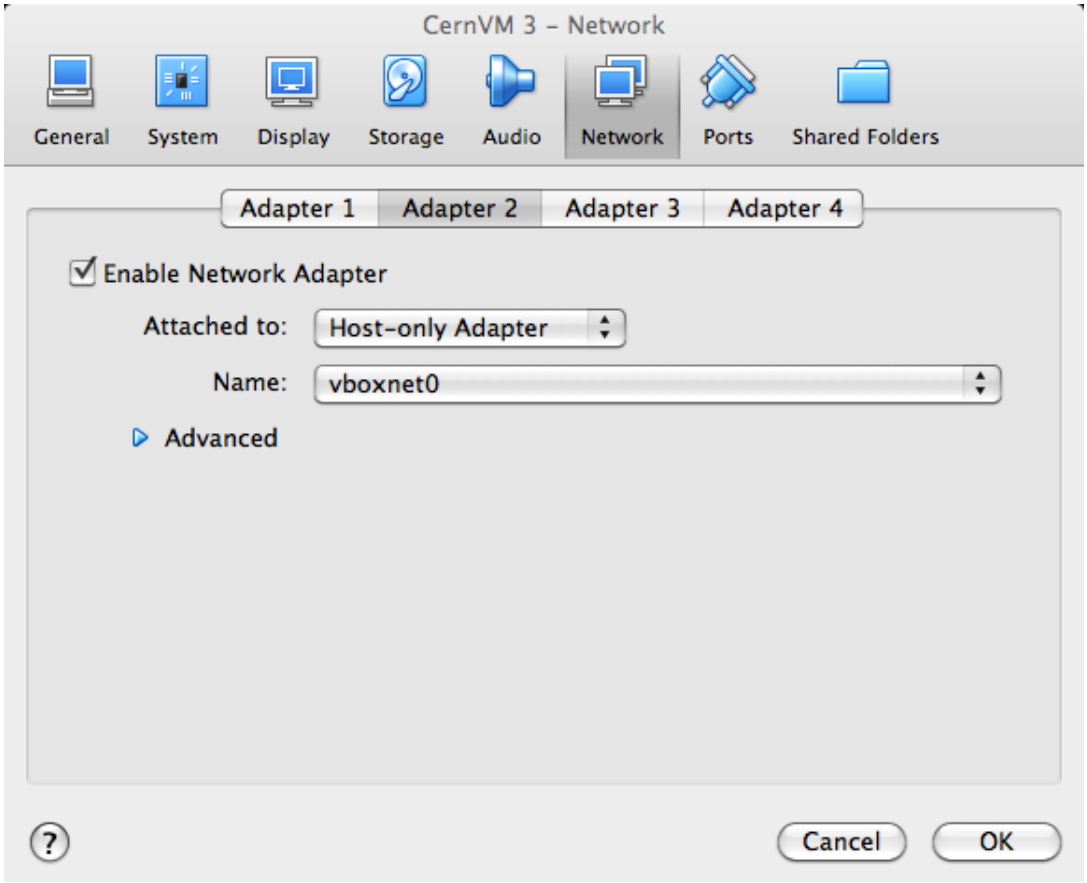
- Click on "**Settings**"
  - Under "General", select the "Advanced" tab and choose "**Bidirectional**" for "Shared Clipboard" and "Drag and Drop"



– Under “Storage”, click on the empty CD drive entry on the left. Then click on the little CD icon on the top right. Select the CernVM ISO image that you downloaded previously. The CernVM image should appear as CD entry in the left pane.



– Under “Network”, click on the “**Adapter 2**” tab and enable the adapter as a “**Host-only Adapter**”

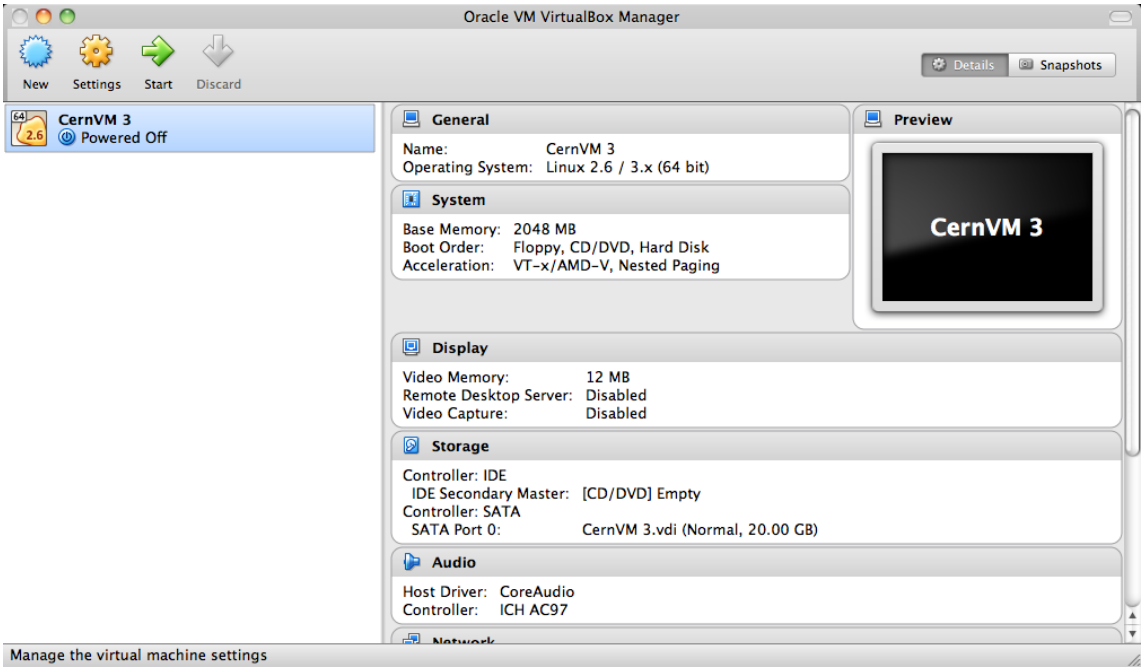


- Under “Graphics”, select “**VBoxSVGA**” as a “Graphics Controller”
- Close the settings dialog by clicking “**OK**”.

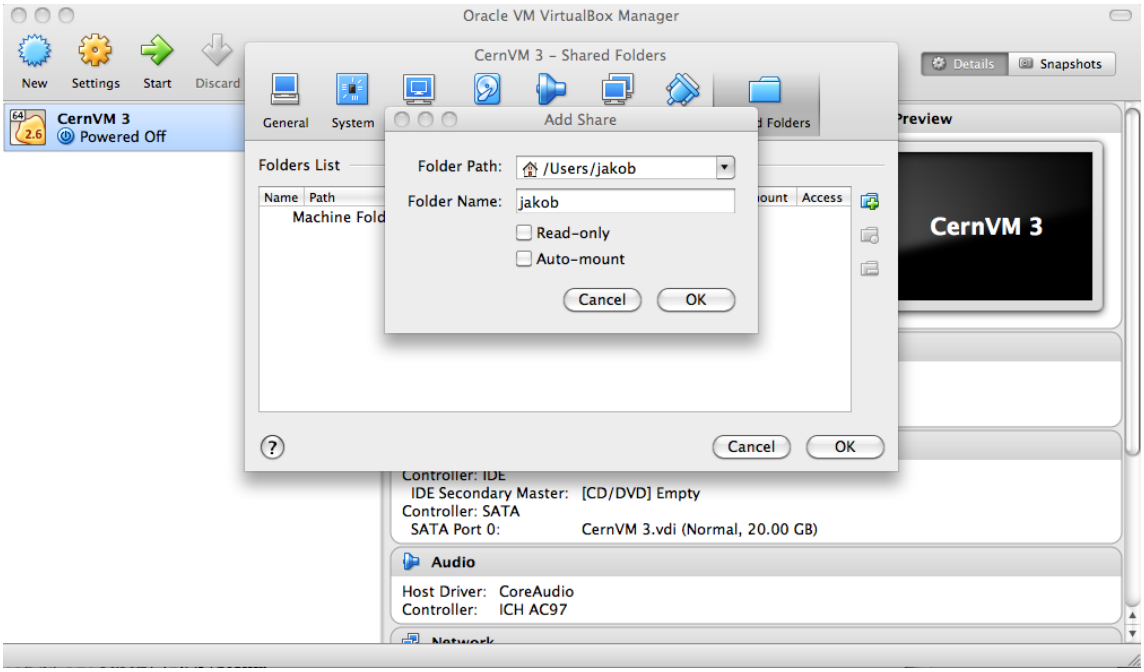
Now you can Start the virtual machine by double clicking your virtual machine entry in the left pane. Once booted, you need to pair the virtual machine with a context defined in CernVM Online.

Shared Folders

- To configure created Virtual Machine, click on “**Settings**”.



- Add the directories under "Shared Folders". Do *not* select the "Auto-mount" option. Inside the virtual machine, shared folders are automatically mounted under /mnt/shared.



2.2.2 KVM

CernVM images for KVM (Kernel-based Virtual Machine) are intended to be used by experienced users, or by system administrators. For creating and controlling virtual machines with KVM we recommend using libvirt (in particular `virsh` command line utility) and virt-manager graphical tool.

Prerequisies

- Make sure that your host supports kvm (the output of the command should not be empty)

```
egrep '(vmx|svm)' --color=always /proc/cpuinfo
```

- Make sure that you have kvm, bridge-utils, libvirt, and virt-manager packages installed. The output of the following commands should not be empty.

```
lsmod | egrep 'kvm(_intel|_amd)'
brctl --version
virsh --version
virt-manager --version # (only for graphical interface)
```

- Make sure that the permissions of /dev/kvm file are set to 0660 (or more open). The file should be owned by "root", and the group ownership should be set to "kvm"

```
> ls -al /dev/kvm
crw-rw---- 1 root kvm 10, 232 Oct 19 14:49 /dev/kvm
```

- Make sure that KVM network is properly set up for using NAT

```
> virsh net-dumpxml default
<network>
  <name>default</name>
  <uuid>a6ae5d2a-8ab5-45a9-94b2-62c29eb9e4f4< /uuid>
  <forward mode='nat'/>
  <bridge name='virbr0' stp='on' forwardDelay='0' />
  <ip address='192.168.122.1' netmask='255.255.255.0'>
    <dhcp>
      <range start='192.168.122.2' end='192.168.122.254' />
    </dhcp>
  </ip>
</network>
```

Creating a Virtual Machine

Download CernVM ISO image from CernVM Downloads page. CernVM requires an empty hard drive as a persistent CernVM-FS cache. Create a sparse hard disk image file with dd:

```
dd if=/dev/zero of=cernvm-hd.img bs=1 count=0 seek=20G
```

Create a virtual machine definition file for `virsh` (libvirt guest domain management interface), which should contain the following:

- Virtual machine name
- Memory size (in MB)
- Number of virtual CPUs
- Type of architecture ("x86_64") and boot device ("cdrom")
- Hard drive and CD-ROM definition
- Network interface definition
- Graphical device definition

Example virtual machine definition file looks like this:

```
<domain type='kvm'>
  <name>CernVM</name>
  <memory>2097152</memory>
  <vcpu>1</vcpu>
  <os>
    <type arch='x86_64'>hvm</type>
```

(continues on next page)

(continued from previous page)

```
    <boot dev='cdrom' />
  </os>
  <features>
    <acpi />
    <apic />
    <pae />
  </features>
    <devices>
      <disk type='file' device='cdrom'>
        <source file='/data/test/cernvm4-micro-2020.04-1.iso' />
        <target dev="hdc" bus="ide" />
        <readonly />
      </disk>
      <disk type='file' device='disk'>
        <source file='/home/user/cernvm-hd.img' />
        <target dev='vda' bus="virtio" />
      </disk>
      <interface type='network'>
        <source network='default'/>
        <model type='virtio'/>
      </interface>

      <graphics type='vnc' listen='0.0.0.0' port='6019'/>
    </devices>
</domain>
```

The virtual machine is created with the following command

```
virsh create vm-definition.xml
```

Virtual machines can be listed, started, stopped, and removed with:

```
> virsh list
 Id Name                 State
----------------------------------
 5  CernVM               running

> virsh shutdown CernVM
Domain CernVM is being shutdown

> virsh start CernVM
Domain CernVM started

> virsh destroy CernVM
Domain CernVM destroyed
```

2.2.3 CernVM as a Docker Container

The CernVM docker container resembles the µCernVM idea in docker. It consists mainly of a busybox and the parrot sandboxing tool. The rest of the operating system is loaded on demand. Note that newer versions of docker prevent the use of the `ptrace()` call, which is required for parrot. This needs to be explicitly allowed by the `--security-opt seccomp:unconfined` parameter to `docker run ...`

Alternatively, it is possible to bind mount the cvmfs operating system repository into the docker container, and then the container will automatically use this instead of parrot.

Limitations of the CernVM Docker Container

The CernVM docker container is a runtime environment only. It can be used to start arbitrary commands "dockerized" in CernVM. Due to its internal mechanis, it cannot be used, however, as a base image to create derived Docker containers, e.g. with a `Dockerfile`.

Instead you can wrap the setup commands that would be part of the `Dockerfile` into a script and pass this script as parameter to the `/init` command line (see below). The script can be bind mounted into the container with the `-v` option, like

```
docker run --security-opt seccomp:unconfined -v /path/to/script:/bootstrap ... \
  /init /bootstrap/script.sh
```

Importing and Running the Container

In order to import the image, ensure that the docker service is running and execute

```
cat <CernVM Docker tarball> | docker import - my_cernvm
```

In order to start an interactive shell, run

```
docker run --security-opt seccomp:unconfined -it my_cernvm /init
```

The initial command always needs to be `/init`, but any other command can be appended, for instance

```
docker run --security-opt seccomp:unconfined -it my_cernvm /init ls -lah
```

In case CernVM-FS is mounted on the docker host, it is possible to help the container and bind mount the operating system repository like

```
docker run -v /cvmfs/cernvm-prod.cern.ch:/cvmfs/cernvm-prod.cern.ch ...
```

In this case, there is no Parrot environment. Every repository that should be available in the docker container needs to be mapped with another `-v ...` parameter. **Note**: the cernvm-prod.cern.ch repository (or other OS hosting cvmfs repositores) should be mounted with the `CVMFS_CLAIM_OWNERSHIP=no` option. You can create a file `/etc/cvmfs/config.d/cernvm-prod.cern.ch.local` and add the configuration parameter. This will ensure that sudo works in your docker container.

The image can be further contextualized by environment variables. To turn on more verbose output:

```
docker run --security-opt seccomp:unconfined -e CERNVM_DEBUG=1 -e PARROT_OPTIONS="-
↪d cvmfs" -it ...
```

To use another operating system provided by CernVM-FS:

```
docker run --security-opt seccomp:unconfined -e CERNVM_ROOT=/cvmfs/cernvm-sl7.cern.
↪ch/cvm4 -it ...
```

or

```
docker run --security-opt seccomp:unconfined -e CERNVM_ROOT=/cvmfs/cernvm-slc5.
↪cern.ch/cvm3 -it ...
```

or

```
docker run --security-opt seccomp:unconfined -e CERNVM_ROOT=/cvmfs/cernvm-slc4.
↪cern.ch/cvm3 -it ...
```

Standard LHC cvmfs repositories are present by default, other repositories can be added with

```
docker run --security-opt seccomp:unconfined -e PARROT_CVMFS_REPO=" \
  <REPONAME>:url=<STRATUM1-URL>,pubkey=/UCVM/keys/<KEYNAME> \
  <REPONAME>: ..."
```

The corresponding public key needs to be stored in the container under /UCVM/keys first.

2.2.4 OpenStack

Publicly Available Images at CERN

The CERN OpenStack interface provides publicly available CernVM images for the x86_64 architecture. The CernVM 4 images are CC7 compatible. The image name indicates the "bootloader version". The bootloader contains the Linux kernel and a CernVM-FS client. The actual operating system is loaded from a CernVM-FS repository.

To start a new CernVM instance,

- Log on to lxplus-cloud.cern.ch
- Check the available CernVM images from `openstack image list`
- Check the available virtual machine flavors from `openstack flavor list`
- Start a new instance like

```
openstack server create --flavor cvm.medium --image "CernVM 4 - Bootloader␣
↪v2020.04-1 [2020-04-01]" ...
```

  Add `--property cern-services=false` to speed up VM creation

CernVM images can be *contextualized with cloud-init and amiconfig*, general information about the image can be found in the *release notes*.

Manually Uploading Images (outside CERN)

To be completed.

2.2.5 Amazon EC2

To run instances on Amazon EC2, the CernVM image must be uploaded first to Amazon S3 ("instance storage" instant types) or to Amazon EBS (EBS backed instance types). Note that you need to provision the image in the same Amazon region where you intend to run your instances. Use `ec2-describe-regions` for a list of available regions.

Preparation

In order to avoid passing credentials and region to each and every command, export the following variables:

```
export AWS_ACCESS_KEY=<ACCESS KEY>
export AWS_SECRET_KEY=<SECRET KEY>
export EC2_URL=https://ec2.<REGION>.amazonaws.com
```

If you want to use Amazon's "enhanced networking" capabilities or if you have a recent account with AWS without support for "EC2 Classic Mode", you need to first create a virtual network ("Virtual Private Cloud (VPC)"). There are many options to configure such a virtual network. Here, we'll create a simple private network with a NAT to the Internet. You can also use the Amazon Web Console to create the VPC.

```
ec2-create-vpc 10.1.0.0/16 --tenancy default
  --> <VPC ID>
ec2-create-subnet -c <VPC ID> -i 10.1.0.0/16
  --> <SUBNET ID>  # needed for ec2-run-instances
ec2-create-route-table <VPC ID>
  --> <ROUTE TABLE ID>
ec2-associate-route-table <ROUTE TABLE ID> -s <SUBNET ID>
ec2-create-internet-gateway
  --> <GATEWAY ID>
ec2-attach-internet-gateway <GATEWAY ID> -c <VPC ID>
ec2-create-route <ROUTE TABLE ID> -r 0.0.0.0/0 -g <GATEWAY ID>
ec2-create-group cernvm-firewall -c <VPC ID> -d "default inbound/outbound port␣
↪openings"
  --> <SECURITY GROUP ID>  # required for ec2-run-instances
# Unrestricted inbound access:
ec2-authorize <SECURITY GROUP ID> --protocol all --cidr 0.0.0.0/0
# Or: ssh only inbound access:
ec2-authorize <SECURITY GROUP ID> --protocol tcp --port-range 22 --cidr 0.0.0.0/0
ec2-create-keypair key-cernvm-<REGION>  # required for ec2-run-instances
```

Copy the "BEGIN RSA" / "END RSA" block from the last command into a file `key-cernvm-<REGION>.pem` and run `chmod 0600 key-cernvm-<REGION>.pem`. As a further prerequisite, you need to have an S3 storage bucket in your target region, which you can create through the Amazon Web Console.

Using Images from EBS for "EBS Backed" Instance Types

The following steps are necessary to prepare the EBS volume snapshots and the image. First import the CernVM "Raw (HVM)" image for Amazon from the CernVM download page into a minimally sized (1G) EBS volume:

```
ec2-import-volume -o $AWS_ACCESS_KEY -w $AWS_SECRET_KEY -f raw -s 1 \
  -z <AVAILABILITY ZONE> --bucket <S3 BUCKET> <CERNVM IMAGE>.hvm
```

The zones for the `-z` parameter can be listed with `ec2-describe-availability-zones`. Use `ec2-describe-conversion-tasks` to get the import task id and to check when the import task finished. Once finished, remove the intermediate image manifest in the S3 bucket with

```
ec2-delete-disk-image -t <IMPORT TASK ID>
```

Use `ec2-describe-volumes` to get the volume id of the imported volume and create a snapshot with

```
ec2-create-snapshot <IMAGE VOLUME ID>
  --> <IMAGE SNAPSHOT ID>
```

In addition to the image volume, create a scratch volume (e.g. with 25G) and a scratch snapshot using

```
ec2-create-volume -s 25 -z <AVAILABILITY ZONE>
  --> <SCRATCH VOLUME ID>
ec2-create-snapshot <SCRATCH VOLUME ID>
  --> <SCRATCH SNAPSHOT ID>
```

Register an EBS backed image with

```
ec2-register -n <NAME> -a x86_64 -d <DESCRIPTION> -snapshot <IMAGE SNAPSHOT ID< \
  -b /dev/sdb=<SCRATCH SNAPSHOT ID> --virtualization-type hvm --sriov simple
  --> <AMI ID>
```

Start instances for the new image with

```
ec2-run-instances <AMI ID> -n <NUMBER OF INSTANCES> -k key-cernvm-<REGION> \
  -s <SUBNET ID> --group <SECGROUP ID> -t <INSTANCE TYPE> -f <USER DATA FILE> \
  # optionally: --associate-public-ip-address true --ebs-optimized
```

Using Images from S3 for "Instance Store" Instance Types

Use the following commands to upload an image for use with "Instance Store" image types:

```
ec2-bundle-image -u <AWS ACCOUNT NUMBER> -c <AWS CERTIFICATE FILE> -k <AWS PRIVATE␣
↪KEY FILE> \
  -i <CERNVM IMAGE>.hvm --arch x86_64
ec2-upload-bundle -a $AWS_ACCESS_KEY -s $AWS_SECRET_KEY \
  -m /tmp/<CERNVM IMAGE>.hvm.manifest.xml -b <S3 BUCKET> --region <REGION>
ec2-register <S3 BUCKET>/<CERNVM IMAGE>.hvm.manifest.xml -a x86_64 -d <DESCRIPTION>
↪ \
  --virtualization-type hvm --sriov simple
  --> <AMI ID>
```

Start instances for the new image with

```
ec2-run-instances <AMI ID> -n <NUMBER OF INSTANCES> -k key-cernvm-<REGION> \
  -s <SUBNET ID> --group <SECGROUP ID> -t <INSTANCE TYPE> -f <USER DATA FILE>  \
  # optionally: --associate-public-ip-address true
```

Enhanced Networking

CernVM contains the default Xen network driver, as well as the "Intel Virtual Function (VF)" adapter and the Amazon "Elastic Network Adapter (ENA)". With the `--sriov simple` parameter to the `ec2-register` command, the Intel VF adapter is automatically used if provided by the instance type. For ENA, the `aws` command line utility is required (e.g. `sudo pip install aws` in CernVM). Amazon provides instructions on how to enable the "enaSupport" attribute on an instance.

Whether or not ENA / Intel VF drivers are used can be tested with `ethtool -i eth0`. If it says "vif" for the driver, it's the standard Xen driver.

2.2.6 Google Compute Engine

The following steps upload the image and start an instance on GCE:

- Login to GCE and switch to a project with

```
gcloud auth login
gcloud config set project <PROJECT NAME>
```

- If you haven't already done so, upload the GCE .tar.gz image to Google cloud storage with

```
gsutil cp <GCE IMAGE> gs://<BUCKET>/<GCE IMAGE>
gcloud compute images create <IMAGE NAME> --source-uri gs:///<BUCKET>/<GCE␣
↪IMAGE>
```

- If you haven't already done so, create an ssh key pair to login to the VMs in your project

```
ssh-keygen -f <KEYPAIR>
gcloud compute project-info add-metadata --metadata "sshKeys=root:$(cat
↪<KEYPAIR>.pub)"
```

- Start an instance with

```
gcloud compute instances create <INSTANCE NAME> \
  --image <IMAGE NAME> --metadata-from-file user-data=<FILE NAME>
```

2.2.7 Microsoft Azure

You can use the `azure` utility in CernVM to upload images to the Azure cloud storage and to control virtual machines.

Setting Azure Credentials

In order to establish your account credentials, use

```
azure account download
azure account import <CREDENTIALS FILE>
```

and follow the instructions of the utility.

Uploading the CernVM Image

For Azure, get the CernVM image in VHD format from the download page. If you haven't done before, create a *storage account* with

```
azure storage account create <STORAGE ACCOUNT>
```

Otherwise, set the storage account with

```
azure storage account set <STORAGE ACCOUNT>
```

Retrieve the storage *connection string* and set it in your environemnt. The `<ACCOUNT KEY>` refers to the last part of the connection string following `AccountKey=`.

```
azure storage account connectionstring show <STORAGE ACCOUNT>
export AZURE_STORAGE_CONNECTION_STRING="<CONNECTION STRING>"
export AZURE_STORAGE_ACCESS_KEY="<ACCESS KEY>"
```

If you haven't done so, create a *container* in your storage account with

```
azure storage container create <CONTAINER>
```

Upload and create the image (you can pick `<IMAGE NAME>`) with

```
azure vm disk upload <CERNVM IMAGE> \
  https://<STORAGE ACCOUNT>.blob.core.windows.net/<CONTAINER>/<IMAGE NAME>.vhd \
  $AZURE_STORAGE_ACCESS_KEY
azure vm image create --os linux --blob-url \
  https://<STORAGE ACCOUNT>.blob.core.windows.net/<CONTAINER>/<IMAGE NAME>.vhd \
  <IMAGE NAME>
```

Creating Virtual Machines

For Azure VMs, the ssh credentials are extraced from an X.509 certificate. In order to create valid ssh credentials, use

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout <KEY NAME>.key -out
↪<CERT NAME>.pem
chmod 600 <KEY NAME>.key
```

You can also create credentials from an existing SSH key with

```
openssl req -x509 -key ~/.ssh/id_rsa -nodes -days 365 -newkey rsa:2048 -out <CERT␣
↪NAME>.pem
```

These procedure is described in more detail in the Azure Online Help.

Virtual machine creation requires a user name and password, even if ssh credentials are provided. We recommend to use `azure` for `<USER NAME>` and a throw-away password, for instance `"@Aa0$(cat /dev/urandom | tr -cd [:alnum:] | head -c24)"`. Create the virtual machine with

```
azure vm create <INSTANCE NAME> <IMAGE NAME> --ssh --ssh-cert <CERT NAME>.pem \
  --custom-data "./user-data" <USER NAME> <PASSWORD>
```

For ssh login, you can retrieve the public IP address of the image with

```
azure vm show <INSTANCE NAME>
```

For help on creating the user-data file, see our *contextualization page*.

2.3 CernVM Launch

Note: CernVM Launch does currently not work with VirtualBox 6.1 on Windows. We are working on a fix. Please use VirtualBox 6.0 for the time being.

The `cernvm-launch` utility is a single binary for Windows, Linux, and Mac that creates, lists, and destroys (interactive) CernVMs on VirtualBox. It works similar to the vagrant and docker command line utilities. It is meant to be used for interactive (graphical) CernVM instances on a local workstation or laptop.

2.3.1 Installation

As a one time preparation you need to:

- Install VirtualBox.
- Download the CernVM-Launch binary for your operating system and put it in your `PATH` environment (see below).

Once installed, you can manage CernVM instances using different "*context*" ASCII files, which you can store on your computer.

The installation process depends on the platform. For instructions how to install VirtualBox, please visit the official VirtualBox page.

Linux/Mac Installation

Run the following commands in your terminal:

```
# Download
if [ "x$(uname -s)" = "xLinux" ]; then
  curl -o cernvm-launch https://ecsft.cern.ch/dist/cernvm/launch/bin/Linux/cernvm-
↪launch
else
  curl -o cernvm-launch https://ecsft.cern.ch/dist/cernvm/launch/bin/Mac/cernvm-
↪launch
fi
chmod +x cernvm-launch  # make it executable
# As root user, place it in a directory from the $PATH environment
sudo cp cernvm-launch /usr/local/bin
```

You can pick a different directory from your `$PATH` environment. Use `echo $PATH` to see all possible directories.

Windows

Download the cernvm-launch executable. Open a Windows prompt as an Administrator

- Click the Start icon
- Type `cmd`
- Right-click on "cmd.exe" and click "Run as administrator"

Go to the directory where you have the downloaded binary, e.g.

```
cd C:\Users\sftnight\Downloads
```

From there, copy the binary in a directory, which is a default path for executable, e.g.

```
copy cernvm-launch.exe C:\Windows
```

You can see your path directories with `echo %PATH%`.

2.4 Contextualization

2.4.1 Introduction

A context is a small (up to 16kB), usually human-readable snippet that is used to apply a role to a virtual machine. A context allows to have a singe virtual machine image that can back many different virtual machine instances in so far as the instance can adapt to various cloud infrastructures and use cases depending on the context. In the process of contextualization, the cloud infrastructure makes the context available to the virtual machine and the virtual machine interprets the context. On contextualization, the virtual machine can, for instance, start certain services, create users, or set configuration parameters.

For contextualization, we distinguish between so called meta-data and user-data. The meta-data is provided by the cloud infrastructure and is not modifiable by the user. For example, meta-data includes the instance ID as issued by the cloud infrastructure and the virtual machine's assigned network parameters. The user-data is provided by the user on creation of the virtual machine.

Meta-data and user-data are typically accessible through an HTTP server on a private IP address such as 169.254.169.254. The cloud infrastructure shields user-data from different VMs so that it can be used to deliver credentials to a virtual machine.

In CernVM, there are three entities that interpret the user-data. Each of them typically reads "what it understands" while ignoring the rest. The µCernVM bootloader interprets a very limited set of key-value pairs that are used to initialize the virtual hardware and to select the CernVM-FS operating system repository. In a later boot phase, amiconfig and cloud-init are used to contextualize the virtual machine. The amiconfig system was initially provided by rPath but it is now maintained by us. It provides very simple, key-value based contextualization that is processed by a number of amiconfig plugins. The cloud-init system is maintained by Redhat and Ubuntu and provides a more sophisticated but also slightly more complex contextualization mechanism.

2.4.2 Contextualization of the µCernVM Boot Loader

The µCernVM bootloader can process EC2, Openstack, and vSphere user data. Within the user data everything is ignored expect a block of the form

```
[ucernvm-begin]
key1=value1
key2=value2
...
[ucernvm-end]
```

The following key-value pairs are recognized:

- **resize_rootfs**: Can be **on** or **off**. When turned on, use all of the hard disk as root partition instead of the first 20G
- **cvmfs_http_proxy**: HTTP proxy in CernVM-FS notation
- **cvmfs_pac_urls**: WPAD proxy autoconfig URLs separated by ';'
- **cvmfs_server**: List of Stratum 1 servers, e.g. cvmfs-stratum-one.cern.ch,another.com
- **cvmfs_tag**: The snapshot name, useful for the long-term data preservation
- **cernvm_inject**: Base64 encoded .tar.gz ball, which gets extracted in the root tree
- **useglideinWMS**: Can be **on** or **off**, defaults to on. When turned off, *glideinWMS* auto detection gets disabled

2.4.3 Contextualization with amiconfig

The amiconfig contextualization executes on boot time, parses user data and looks for python style configuration blocks. If a match is found the corresponding plugin will process the options and execute configuration steps if needed. By default, enabled rootsshkeys is the only enabled plugins (others can be enabled in the configuration file).

Default plugins:

```
rootshkeys              - allow injection of root ssh keys
```

Available plugins

```
cernvm                  - configure various CernVM options
condor                  - setup Condor batch system
disablesshpasswdauth    - if activated, it will disable ssh authentication with␣
↪password
dnsupdate               - update DNS server with current host IO
ganglia                 - configure gmond (ganglia monitoring)
hostname                - set hostname
noip                    - register IP address with NOIP dynamic DNS service
nss                     - /etc/nsswithch.conf configuration
puppet                  - set parameters for puppet configuration management
squid                   - configure squid for use with CernVM-FS
```

Common amiconfig options:

```
[amiconfig]
plugins = <list of plugins to enable>
disabed_plugins = <list of plugins to disable>
```

Specific plugin options:

```
[cernvm]
# list of ',' seperated organisations/experiments (lowercase)
organisations = <list>
# list of ',' seperated repositories (lowercase)
repositories = <list>
# list of ',' separated user accounts to create <user:group:[password]>
users = <list>
# CernVM user shell </bin/bash|/bin/tcsh>
shell = <shell>
# CVMFS HTTP proxy
proxy = http://<host>:<port>;DIRECT
------------------------------------------------------------
# url from where to retrieve initial CernVM configuration
config_url = <url>
```

(continues on next page)

(continued from previous page)

```
# list of ',' separated scripts to be executed as given user: <user>:/path/to/
↪script.sh
contextualization_command = <list>
# list of ',' seperated services to start
services = <list>
# extra environment variables to define
environment = VAR1=<value>,VAR2=<value>
```

```
[condor]
# host name
hostname = <FQDN>
# master host name
condor_master = <FQDN>
# shared secret key
condor_secret = <string>
#------------------------
# collector name
collector_name = <string>
# condor user
condor_user = <string>
# condor group
condor_group = <string>
# condor directory
condor_dir = <path>
# condor admin
condor_admin = <path>
highport = 9700
lowport = 9600
uid_domain =  filesystem_domain =  allow_write = *.$uid_domain extra_vars = use_
↪ips =
```

Contextualization scripts

If the user data string starts with a line starting with `#!`, it will be interpreted as a bash script and executed. The same user data string may as well contain amiconfig contextualization options but they must be placed after the configuration script which must end with an exit statement. The interpreter can be `/bin/sh` or `/bin/sh.before` or `/bin/sh.after` depending on when the script is to be executed, before or after amiconfig contextualization. A script for the `/bin/sh` interpreter is executed after amiconfig contextualization.

2.4.4 Contextualization with cloud-init

As an alternative to amiconfig, CernVM supports cloud-init contextualization.

2.4.5 Mixing user-data for µCernVM, amiconfig, and cloud-init

The user-data for cloud-init and for amiconfig can be mixed. The cloud-init syntax supports user data divided into multiple MIME parts. One of these MIME parts can contain amiconfig or µCernVM formatted user-data. All contextualization agents (µCernVM, amiconfig, cloud-init) parse the user data and each one interprets what it understands.

The following example illustrates how to mix amiconfig and cloud-init. We have an amiconfig context amiconfig-user-data that starts a catalog server for use with Makeflow:

```
[amiconfig]
plugins = workqueue
[workqueue]
```

We also have a cloud-init context cloud-init-user-data that creates an interactive user "cloudy" with the password "password"

```
users:
  - name: cloudy
    lock-passwd: false
    passwd: $6$XYWYJCb.$OYPPN5AohCixcG3IqcmXK7.yJ/wr.TwEu23gaVqZZpfdgtFo8X/
↪Z3u0NbBkXa4tuwu3OhCxBD/XtcSUbcvXBn1
```

The following helper script creates our combined user data with multiple MIME parts:

```
amiconfig-mime cloud-init-user-data:cloud-config amiconfig-user-data:amiconfig-
↪user-data > mixed-user-data
```

In the same way, the µCernVM contextualization block can be another MIME part in a mixed context with MIME type ucernvm.

2.4.6 glideinWMS User Data

By default, CernVM will automatically detect user data from glideinWMS and, if detected, activate the glideinWMS VM agent. CernVM recognizes user data that consists of no more than two lines and that contains the pattern `...#### -cluster 0123 -subcluster 4567####...` as glideinWMS user data. It will automatically extract the CernVM-FS proxy configuration (proxy and pac URLs) from the user data. In order to disable autodetection, set `useglideinWMS=false` in the µCernVM contextualization.

2.4.7 Extra Contextualization

In addition to the normal user data, we have experimental support for "extra user data", which might be a last resort where the normal user data is occupied by the infrastructure. For instance, glideinWMS seems to exclusively specify user data, making it necessary to modify the image for additional contextualization. Extra user data are injected in the image under /cernvm/extra-user-data and they are internally appended to the normal user data. This does not yet work with cloud-init though; only with amiconfig and the µCernVM bootloader.

2.4.8 Applying User Data

CernVM supports applying contextualization information at boot time using one of the following mechanisms:

- User-Data text snippet: almost all of the private or public cloud infrastructures provide a mechanism of passing arbitrary data to the instance at the creation time. A good example is Amazon's Instance Metadata for EC2.
- CD-ROM: the user data are stored to CD-ROM ISO images that are attached to the virtual machine.

Both mechanisms eventually pass a string of ini-like data to the instance.

Preparing a User-Data CD-ROM Image

If it is not possible pass the user-data by the cloud infrastructure, you can use the mechanism of a contextualization CD-ROM image. This image must contain at least one file called `context.sh` and this file must have at least the following two lines:

```
EC2_USER_DATA="<user-data>"
ONE_CONTEXT_PATH="/var/lib/amiconfig"
```

Where `<user-data>` is the base64-encoded user data text snippet created as decribed earlier.

To create the CD-ROM image (for example user-data.iso) you can then use the `mkisofs` utility:

```
mkdir iso-tmp
echo 'EC2_USER_DATA=123abc...' >> iso-tmp/context.sh
echo 'ONE_CONTEXT_PATH="/var/lib/amiconfig"' >> iso-tmp/context.sh
mkisofs -o user-data.iso iso-tmp
```

You must then mount this CD-ROM image to you virtual machine before you boot it. This is done differently on every hypervisor, so check your hypervisor configuration for more information.

2.5 Micro-CernVM Bootloader

To be completed

2.6 CernVM Image Signatures

The HDD and ISO images of the CernVM bootloader are signed by the cvm-sign02.cern.ch host certificate. At the end of the images, there is a 64kB signature block (`tail -c $((64*1024)) <image>`) which is ignored by the hypervisors. The 64kB block is divided into two zero-padded 32kB blocks. The first block contains a JSON object with meta-data about the image, the second block contains the signature JSON object. The signature is on the image plus the first 32kB block.

The signature JSON object in the last 32kB signature block contains the base64 encoded strings "certificate" and "signature". There is also a "howto-verify" list of strings containing a few hints how to verify the signature with openssl.

In order to verify the image signature, you can use the following steps

```
tail -c $((64*1024)) <IMAGE>
# write JSON contents of "certificate" to encoded_certificate using text editor
# write JSON contents of "signature" to encoded_signature using text editor
base64 -d encoded_certificate > certificate
base64 -d encoded_signature > signature
openssl verify -CApath <X509_CERT_DIR> certificate
openssl x509 -in certificate -subject -noout # make sure output matches DN of cvm-
↪sign02.cern.ch certificate
openssl x509 -in certificate -pubkey -noout > pubkey
head -c -32768 <IMAGE> > signed_image
openssl dgst -sha256 -verify pubkey -signature signature signed_image
```

CHAPTER 3

Contact and Authors

Visit our website on cernvm.cern.ch.

Authors of this documentation:

- Jakob Blomer