# LightVM

**Lighter (and Safer) VMs than Containers**

Download (/projects/lightvm/download/)

We are HIRING!

(/jobs/)

Overview (/projects/lightvm/) Papers & Talks (/projects/lightvm/papers/) **Getting Started (/projects/lightvm/download/)** Team (/projects/lightvm/team/)

## Getting Started

LightVM is a virtualization solution based on Xen that is optimized to offer fast boot-times regardless of the number of active VMs. This is achieved by replacing Xenstore with a new distributed solution called NoXS (no Xenstore) which provides a shared page for each device containing all the information needed for device initialization.

LightVM uses the new Chaos toolstack which currently implements operations such as instantiation, saving, restoring and migration. Chaos can also be used in Xen environments based on Xenstore, making it a streamlined alternative for the `xl` toolstack. VMs management is assisted by the XenDevD daemon for proper device initialization.

Currently, LightVM relies on Linux kernel for dom0, while for unprivileged domains one can use both Linux and Mini-OS based guests.

LightVM environment consists of multiple components, each of them having its own repository. The Xen repository provides the Xen hypervisor, on top of which the virtualized domains will be running, and the libraries needed by Chaos toolstack. The toolstack uses `libxc` library for interacting with the hypervisor on domain creation, shutdown and inspection, and for migrating domains data to remote hosts. If deployed in a Xenstore-based environment, the toolstack will also need the `libxenstore` library for communicating with the `xenstored` daemon.

The Linux repository provides the changes needed for building both dom0 and domU domains. Inside dom0, Chaos will make use of the `/dev/xen/noxs_backend` device when requesting to the backend drivers the creation of devices configured for the target guest domains. With NoXS, besides the backend drivers for network and block devices, the `sysctl` backend driver is in charge with providing all the system-wise information and events (such as shutdown events) needed by the guest domains. On the guest side, the `sysctl` front driver will receive the information and will trigger the actions implied by the events.

The XenDevD repository provides the source code for the XenDevD daemon and the libraries needed for communication with Chaos. XenDevD daemon listens for udev events in order to carry out the userland operations for devices initialization (e.g. adding vifs to bridges). Some devices may require userland operations before their creation, in which case Chaos will initiate the requests directly to the daemon and will wait for these operations to complete. For example, block devices that are file based will need to be mounted using loop devices; Chaos will provide the name of the file being mounted and the daemon will reply with the loop device path on success.

The Chaos repository contains the source code for toolstack: the tool used in domain creation, shutdown and inspection, the daemon used for receiving migrating domains and the daemon used in split instantiation. Similarly to `xl` using `libxl` library for most of its functionality, the common functionality of Chaos tools is provided by the `libh2` library.

Besides Linux domU domains, LightVM also supports Mini-OS based applications. The Mini-apps repository provides a set of applications examples that can be used to demonstrate the functionality and performance of NoXS based environments.

## Xen

- Repo: https://github.com/sysml/xen (https://github.com/sysml/xen)
- Branch: noxs-4.8.1 based on Xen 4.8.1
- Branch: noxs-4.8.0 based on Xen 4.8.0
- Build and installation steps are the same ones used for upstream Xen. Be sure to provide a custom installation path before building if a different location is desired.

```
$ ./configure --prefix=<my Xen distribution directory>
$ make dist-xen
$ make dist-tools
```

## Linux

- Repo: https://github.com/sysml/linux (https://github.com/sysml/linux)
- Branch: noxs
- Build: Add CONFIG_XEN_NOXS=y in the config file in addition to using the Xen config flags for building Linux domains.

Prepare the userspace headers which will be used by the Chaos toolstack:

```
$ make headers_install INSTALL_HDR_PATH=<my Linux headers>
```

## XenDevD

- Repo: https://github.com/sysml/xendevd (https://github.com/sysml/xendevd)
- Branch: noxs
- Build: Before running make command, update the Makefile to refer to the headers and libraries installed in the previously configured Xen distribution directory:

```
-CFLAGS   += -Iinc -Wall -g -O3
-LDFLAGS  += -lxenstore
+CFLAGS   += -Iinc -Wall -g -O3 -I<Xen source tree>/<my Xen distribution directory>/in
clude
+LDFLAGS  += -lxenstore -L<Xen source tree>/<my Xen distribution directory>/lib
```

## Chaos

- Repo: https://github.com/sysml/chaos (https://github.com/sysml/chaos)
- Branch: master
- Build: Before building, configure the variables in the config.in file to refer to the previously configured environment paths. For build, simply run the make command. NoXS can be enabled by using the CONFIG_H2_XEN_NOXS flag:

```
 $ make CONFIG_H2_XEN_NOXS=y
```

## Mini-OS

- Repo: https://github.com/sysml/mini-os (https://github.com/sysml/mini-os)
- Branch: `noxs`
- Build: Enable NoXS by setting `CONFIG_NOXS` flag:

```
 $ make CONFIG_NOXS=y
```

## Minipython

- Repo: https://github.com/sysml/minipython (https://github.com/sysml/minipython)

## Mini-Apps

- Repo: https://github.com/sysml/mini-apps (https://github.com/sysml/mini-apps)
- Branch: `noxs`
- Build requirements: the Mini-OS applications need the Newlib and Lwip libraries provided by the Mini-OS toolchain.
- Build: In the target application directory:

```
 $ cd daytime
 $ make XEN_ROOT=... MINIOS_ROOT=... NEWLIB_ROOT=... LWIP_ROOT=... <Mini-OS specific
flags>
```

## Mini-OS toolchain

- Repo: https://github.com/sysml/toolchain (https://github.com/sysml/toolchain)
- For build details, follow the indications in the toolchain README