Garden Linux


Garden Linux

Garden Linux is a Debian GNU/Linux derivate that aims to provide small, auditable Linux images for most cloud providers (e.g. AWS, Azure, GCP etc.) and bare-metal machines. Garden Linux is the best Linux for Gardener nodes. Garden Linux provides great possibilities for customizing that is made by a highly customizable feature set to fit your needs.

Table of Content

Features

- Easy to use build system
- Repeatable and auditable builds
- Small footprint
- Purely systemd based (network, fstab etc.)
- Initramfs is dracut generated
- Running latest LTS Kernel

- MIT license
- Security
  - Fully immutable image(s) *(optional)*
  - OpenSSL 3.0 *(default)*
  - CIS Framework *(optional)*
- Testing
  - Unit tests (Created image testing)
  - Integration tests (Image integration tests in all supported platforms)
  - License violations (Testing for any license violations)
  - Outdated software versions (Testing for outdated software)
- Supporting major platforms out-of-the-box
  - Major cloud providers AWS, Azure, Google, Alicloud
  - Major virtualizer VMware, OpenStack, KVM
  - Bare-metal systems

Quick Start

The entire build runs in a *privileged* Podman/Docker container that orchestrates all further actions. If not explicitly skipped, unit tests will be performed. Extended capabilities are at least needed for loop back support. Currently `AMD64` and `ARM64` architectures are supported.

By default, Garden Linux uses Podman as container runtime (`Docker` is optionally supported) for building Garden Linux images (Garden Linux artifacts however will have Docker in them to maintain compatibility with older Kubernetes versions). If - for whatever reason - you want or need to use Docker instead, you can set the environment variable `GARDENLINUX_BUILD_CRE=docker` before invoking the build.

Build Requirements

System:

- RAM: 2+ GiB (use '--lessram' to lower memory usage)
- Disk: 10+ GiB (20+ GiB for running `integration tests`)
- Internet connection

Packages:

Debian/Ubuntu:

```
apt install bash sudo podman crun make coreutils gnupg git qemu-system-x86 qemu-s
```

CentOS/RedHat (>=8):

CFSSL requires `GLIBC 2.28`. Therefore, we recommand to build on systems running CentOS/RedHat 8 or later.

```
# Install needed packages
yum install bash sudo podman crun make gnupg git qemu-kvm qemu-img coreutils
```

macOS (>=12):

Build support on `macOS` (>=12) supports `Intel` (AMD64) and `Apple Silicon` (ARM64/AARCH64) architectures. Building on macOS requires the GNU versions of multiple tools that can be installed in different ways like Brew, MacPorts or self compiled. Self compiled GNU packages must be located in `/opt/local/bin/`. However, the following build instructions only cover the recommended `Brew` way.

Furthermore, building on macOS requires to fulfill further build requirements:

- Command Line Tools (CLT) for Xcode
- Homebrew (Optionally: MacPorts https://macports.org)
- Docker

```
# Install needed packages
brew install coreutils bash gnu-getopt gnu-sed gawk podman

# Change to bash (Default: ZSH)
$> bash

# Export Docker as Container Runtime Environment for Garden Linux
$> export GARDENLINUX_BUILD_CRE=docker
```

Adjust Repository:

*Note: This is **not** needed on macOS.*

Add `docker.io` to `unqualified-search-registries` in your registries.conf. On freshly installed `Podman` systems this can be done by executing:

```
echo 'unqualified-search-registries = ["docker.io"]' | sudo tee -a /etc/container
```

If `Podman` was already present please add the repository yourself to `unqualified-search-registries` in `/etc/containers/registries.conf`.

Kernel Modules:

- ext4
- loop
- squashfs
- vfat
- vsock *(image builds and extended virtualized tests)*

Build Options

Option	Description
--features	Comma separated list of features activated (see features/) (default:base)
--disable-features	Comma separated list of features to deactivate (see features/)
--lessram	Build will be no longer in memory (default: off)
--debug	Activates basically `set -x` everywhere (default: off)
--manual	Built will stop in build environment and activate manual mode (default:off)
--arch	Builds for a specific architecture (default: architecture the build runs on)
--suite	Specifies the debian suite to build for e.g. bullseye, potatoe (default: testing)
--skip-tests	Deactivating tests (default: off)
--tests	Test suite to use, available tests are unittests, kvm, chroot (default: unittests)
--skip-build	Do not create the build container

Building

To build all supported images you may just run the following command:

```
make all
```

However, to save time you may also build just a platform specific image by running one of the following commands. Related dev images can be created by appending the '-dev' suffix (e.g. "make aws-dev").

```
make aws
make gcp
make azure
make ali
make vmware
make openstack
make kvm
make metal
```

Artifacts are located in the `.build/` folder of the project's build directory.

Customizing

Building Garden Linux is based on a feature system.

Feature Type	Includes
Platforms	`ali` , `aws` , `azure` , `gcp` , `kvm` , `metal` , ...
Features	`container host` , `vitual host` , ...
Modifiers	`_slim` , `_readonly` , `_pxe` , `_iso` , ...
Element	`cis` , `fedramp` , `gardener`

if you want to build manually choose:

```
build.sh  --features <Platform>,[<feature1>],[<featureX>],[_modifier1],[_modifier
```

Additionally, please find some `build.sh` example calls in the `Makefile` .

Example:

```
build.sh  --features server,cloud,cis,vmware .build/
```

This builds a server image, cloud-like, with `CIS` feature for the VMware platform. The build result can be found in `.build/` . Also look into our Version scheme since adding a date or a Version targets the whole build for a specific date.

Deploying

Deploying on common cloud platforms requires additional packages. The following overview gives a short quick start to run cloud platform deployments. Currently, all modules are based on `Python` . Therefore, please make sure to have Python installed.

Platform	Module
Alicloud	Aliyun CLI
AWS:	AWS CLI
Azure	Azure CLI
GCP:	Cloud SDK, gsutil
OpenStack	OpenStackCLI

Release

Garden Linux frequently publishes snapshot releases. These are available as machine images in most major cloud providers as well as file-system images for manual import. See the releases page for more info.

Documentation

Garden Linux provides a great documentation for build options, customizing, configuration, tests and pipeline integrations. The documentation can be found within the project's `docs/` path or by clicking here. Next to this, you may find a corresponding `README.md` in each directory explaning some more details. Below, you may find some important documentations for continous integration and integration tests.

Continous Integration

Garden Linux can build in an automated way for continous integration. See ci/README.md for details.

Integration Tests

While it may be confusing for beginners we highlight this chapter for `integration tests` here. Garden Linux supports integration testing on all major cloud platforms (Alicloud, AWS, Azur, GCP). To allow testing even without access to any cloud platform we created an universal `kvm` platform that may run locally and is accessed in the same way via a `ssh client object` as any other cloud platform. Therefore, you do not need to adjust tests to perform local integration tests. Just to mention here that there is another platform called `chroot`. This platform is used to perform `unit tests` and will run as a local `integration test`. More details can be found within the documentation in tests/README.md.

Contributing

Feel free to add further documentation, to adjust already existing one or to contribute with code. Please take care about our style guide and naming conventions. More information are available in in CONTRIBUTING.md and our `docs/` .

Community

Garden Linux has a large grown community. If you need further asstiance, have any issues or just want to get in touch with other Garden Linux users feel free to join our public chat room on Gitter.

Link: https://gitter.im/gardenlinux/community