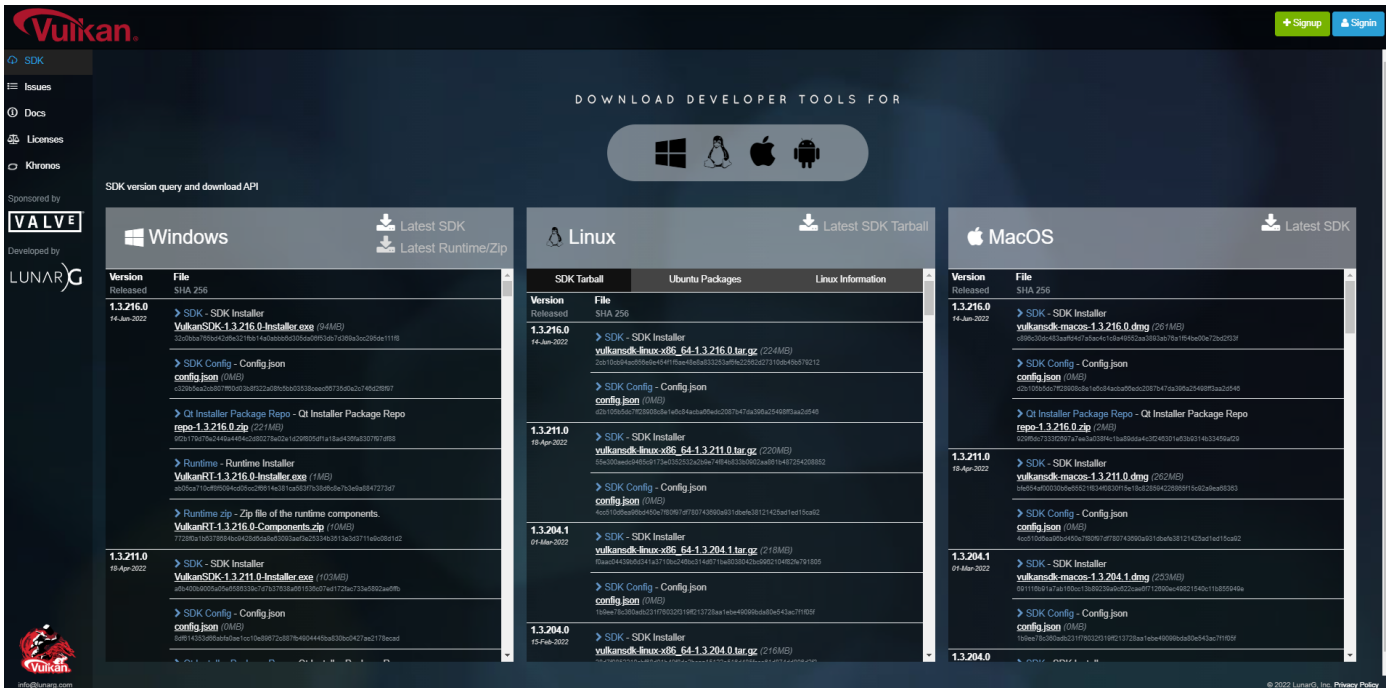
Benefits of Using the Vulkan SDK

Karen Ghavam, LunarG
July 2022

Introduction

The Vulkan SDK, which is available for free download at vulkan.lunarg.com, is a collection of developer tools for Vulkan application developers. There are versions of the Vulkan SDK for Windows 10, macOS, and Linux. For Linux, the Vulkan SDK is delivered as a “tarball” or as package for Ubuntu.



The SDK components come from open-source repositories. While it is feasible to develop Vulkan applications without using the Vulkan SDK, there are many benefits to installing it, such as access to:

- An installation process that is easy and fast
- Vetted and curated content to ensure compatibility and seamless integration
- Ready-to-use versions of the Vulkan Configurator
- An optional updated Vulkan Runtime for Windows environments
- SDK release notes and user documentation
- A License Registry that details all of the licenses included by the SDK components
- Complete package of available shader toolchain tools
- Most up-to-date set of Linux Vulkan components

- All necessary macOS binaries pre-built with an option for system-level installation of MoltenVK as an ICD

This paper explains these benefits.

Easy and Fast to Install

All of the developer tools included in the Vulkan SDK are pre-built, and the SDK installation process installs them into the correct locations on your system, ready for use. This process is the fastest way for Vulkan application developers to prepare their system for Vulkan application development. Critically important developer tools such as the Validation Layers and the Vulkan Configurator are installed.

Vetted with Integrated and Compatible Content

As part of the SDK release process, coordination is done across all of the repositories that comprise the Vulkan SDK. This coordination ensures that all repositories are updated to the same KhronosGroup/Vulkan-Header version (where applicable) and that versions included in the SDK are compatible and contain completed critical defect fixes or new functionality.

Many open source repositories contribute to the Vulkan SDK and are subject to this coordination process:

1. KhronosGroup/Vulkan-ValidationLayers
2. Shader toolchain
    a. Microsoft/DirectXShaderCompiler
    b. SPIRV-tools
    c. KhronosGroup/glslang
    d. google/shaderc
    e. SPIRV-cross
    f. SPIRV-reflect
3. KhronosGroup/Vulkan-Tools
4. KhronosGroup/Vulkan-Profiles
5. LunarG/VulkanTools
6. LunarG/gfxreconstruct
7. AMD Memory Allocator
8. VOLK

Although developers can go and build each of these repositories themselves, one needs to be very diligent in ensuring the same versions of dependent repositories are being used across all of them. The process of vetting and integrating compatible versions can save the Vulkan application developer significant time. For example, the validation layers have a dependency on glslang and spirv-tools. It is more desirable to ensure that the same version being used by the validation layers is also the version being delivered stand-alone from the spirv-tools and glslang repository. Using the SDK can ensure this compatibility.

Vulkan Configurator

Vulkan Configurator allows overriding of the configuration of the layers used by Vulkan applications at runtime and is a very useful developer tool enabling quick and easy configuration of enabled layers.

In addition to the Vulkan Configurator binary, the Vulkan Configurator expects many layers to be installed on the system. The user could build the Vulkan Configurator binary from the LunarG/VulkanTools repository. And in addition, each of the layers (Validation Layers, Profiles Layer, apidump layer, GFXreconstruct capture layer) can be built from their corresponding repositories. Once the Vulkan Configurator and all the layers are built, the user then needs to install the binaries in the correct locations on the system.

To bypass these steps, the user can install the Vulkan SDK, which has completed all of the builds and system installation of the Vulkan Configurator and layers to enable a ready-to-use solution.

The Vulkan Runtime

For Windows, the Vulkan Runtime (Vulkan Loader) is included with the driver packages provided by the independent hardware vendor (IHV) like NVIDIA, Intel, or AMD. However, there are times that you would like to install a newer version of the Vulkan loader sooner than waiting for a driver package update from the IHV. With each release of the SDK, the Windows Vulkan Runtime is available as a separate optional downloadable and installable package.

For the macOS and Linux SDKs, the Vulkan Loader is included and installed as part of the SDK installation.

The Complete Shader Toolchain

There are multiple workflows used for creating SPIR-V shaders to be used with Vulkan applications:

1. HLSL→SPIR-V
    a. The Microsoft DirectX Shader compiler
    b. Glslang (for HLSL versions of version shader model 5.1 or less)
    c. Google/shaderc
2. GLSL→SPIR-V
    a. Glslang
    b. Google/shaderc
3. HLSL/Metal/GLSL shaders → SPIR-V shaders
    a. SPIRV-cross

The Vulkan SDK includes the complete package of available shader toolchain tools:

1. KhronosGroup/glslang (glslangValidator)
2. KhronosGroup/SPIRV-Cross (SPIR-V Cross Compilation and Reflection)
3. SPIRV-Reflect
4. KhronosGroup/SPIRV-Tools (SPIR-V Optimizer, SPIR-V Disassembler, Assembler, and Validator, SPIR-V diff, SPIR-V remapper, SPIR-V Control Flow Visualization)
5. Microsoft/DirectXShaderCompiler
6. Google/shaderc (SPIR-V Compilation Wrapper)

If you use the static versions of the shader toolchain libraries when using Visual Studio to link a debug version of your application, you will need debug versions of these static libraries to link successfully. The debuggable versions of the static shader toolchain libraries are an optional download package with the Windows SDK.

The Linux SDK

The Vulkan SDK for Linux supplies Ubuntu packages and a Linux tarball for developers.

Newer Content in Linux SDK

Although many Linux distributions include some of the Vulkan SDK components as part of their distribution (the Validation Layers and Vulkan Loader are the most common), the Linux SDK is updated much more frequently (about every 8 weeks), whereas the versions included in Linux distributions can get older than a year or more depending upon the update frequency for the Linux distribution.

Ubuntu Distributions

LunarG creates Ubuntu packages for all of the SDK elements for the two latest versions of Ubuntu. These packages install the SDK components, pre-built, in the system directories, ready for use.

Support for Non-Ubuntu Distributions

Not all application developers are using the two most recent versions of Ubuntu or may be developing on another Linux distribution. For these environments, we deliver a Linux tarball that has all of the SDK binaries pre-compiled for the user. It is possible that the precompiled binaries won't work on your Linux distribution (the most common error is incompatible system runtime libraries). To help support these other Linux distribution environments, the Linux tarball SDK comes with a "vulkansdk" script that will rebuild all of the SDK binaries, resulting in compatible binaries with the runtime environment.

Support for Multiple SDK Versions

When using the Linux tarball SDK, you can install as many versions as desired. You also can easily switch between the different versions with the following environment variables (the VULKAN_SDK environment variable is set to the platform-specific directory of your SDK installation):

1. export VULKAN_SDK=~/vulkan/1.x.yy.z/x86_64
2. export PATH=$VULKAN_SDK/bin:$PATH
3. Export LD_LIBRARY_PATH=$VULKAN_SDK/lib${LD_LIBRARY_PATH:+:$LD_LIBRARY_PATH}
4. export VK_LAYER_PATH=$VULKAN_SDK/etc/vulkan/explicit_layer.d

The macOS SDK

The Vulkan SDK is especially useful for macOS and iOS developers. On platforms such as Windows and Linux, Vulkan support is provided as a system-wide driver. The Vulkan loader is typically distributed by IHVs along with their native Vulkan ICDs (Installable Client Drivers). On macOS and iOS devices, Vulkan is implemented a bit differently. There is no system-wide Vulkan driver per se, as Apple does not natively support Vulkan. Instead, a translation library, *MoltenVK*, is used to map the Vulkan API onto Apple's native *Metal* low-level 3D rendering API.

MoltenVK can be used as a static Vulkan implementation, or the Vulkan Loader can load a dynamic version of MoltenVK as if it were an ICD. The loader you will need is provided pre-built by the Vulkan SDK. Typically developers will either link statically to MoltenVK or bundle both the loader and MoltenVK within their application bundle. Using the latter approach, developers can also make use of the Vulkan validation and utility layers, also bundled with the Vulkan SDK.

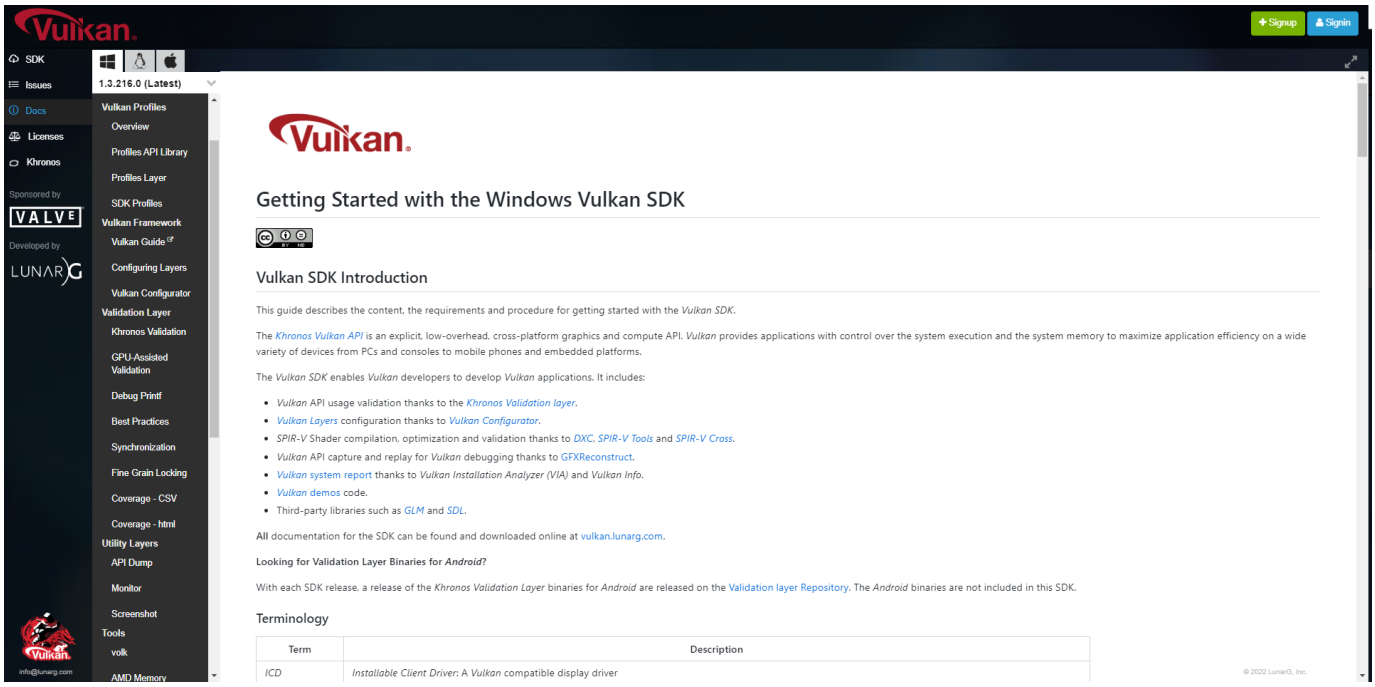
Moreover, the MoltenVK ICD can optionally be installed as a system-wide Vulkan driver by the SDK on macOS, making application development easier (no need to bundle the files during development, command-line Vulkan utilities will work on macOS, etc), and enabling the use of system-wide Vulkan layers. A system-wide Vulkan implementation is not expected on end-user systems, so it is important to remember that you should ship your applications with MoltenVK and the Vulkan Loader in the application bundle, or link statically to MoltenVK for final distribution.

The Vulkan SDK for macOS from LunarG provides all of these binaries prebuilt and validated against the latest set of Vulkan headers. Building the loader, MoltenVK, the layers, and the associated shader tools against the latest Vulkan headers, and testing and validating their operation is no small task (trust us, we do it several times a year!). Using the Vulkan SDK will save you days, if not weeks, of work whenever there is a significant update to the Vulkan headers or expanded MoltenVK functionality. The binaries included with the SDK are also built to support both Intel and Apple Silicon processors and are tested on the current and most recent versions of macOS.

For more information about developing Vulkan applications on Apple devices, see the white paper “The State of Vulkan on Apple Devices.

User Documentation

With each SDK release, the release notes clarify what new functionality and Vulkan extensions are delivered with the SDK. In addition, for each of the developer tools, user documentation is easily found without searching within the open-source repositories. The documentation is available online and is navigated from the “docs panel” at vulkan.lunarg.com:



License Registry

The Vulkan SDK is composed of 100% open-source components. The majority of the materials have MIT or Apache 2.0 licenses, but there are many other open-source licenses as well. The Vulkan SDK license registry discloses all components in the SDK and their corresponding open-source license and copyrights.

It is not uncommon that corporate environments are concerned about the licenses of products installed by their employees. The License Registry details ALL of the open-source licenses that are found in the Vulkan SDK to enable corporate review.

For those who need to see all the details about licenses and copyrights for elements in the SDK, a CSV and a TXT file are available for download.

For more information about the License Registry, see the white paper, The Vulkan SDK License Registry.