# Windows Subsystem for Linux (WSL) 2.0: A Review

Jatin Saroha

M.Tech. Student, CSE, UIET, M.D. University, Rohtak – 124001

## ABSTRACT

**The Windows Subsystem for Linux (WSL) 2.0, marked a significant leap forward in bridging gap between Windows along with Linux ecosystems. With the integration of a full Linux kernel directly into Windows, WSL 2.0 offers enhanced performance and compatibility for running Linux applications and tools within the Windows environment. This seamless integration allows users to leverage their favourite Linux command-line utilities and development workflows without the need for dual booting or virtualization. Moreover, WSL 2.0's native support for Docker containers enables developers to build, run, and manage Linux-based containers directly on Windows, further streamlining the development process. While occasional compatibility issues may arise, WSL 2.0's growing adoption and active community support underscores its potential to revolutionise cross-platform development and collaboration.**

**Keywords:** ***Linux command line, Docker, Cross platform, Memory consumption, WSL 2.0.***

## INTRODUCTION

Windows Subsystem for Linux (WSL) 2.0 is a feature in Microsoft Windows allows users to run a Linux kernel directly on top of Windows operating system. It enables developers along with system administrators to utilise Linux tools, utilities, and programming languages natively within a Windows environment, providing seamless integration of the two operating systems.WSL 2.0, which represents a significant evolution from its predecessor, WSL 1.0, by introducing a virtual-machine-based architecture that includes a full Linux kernel.

### Key Benefits

This architectural change results in several key benefits, including improved performance, enhanced compatibility, and broader support for Linux applications.

**Using WSL 2.0, users can :**

**1. Boost Performance:** By leveraging a virtual machine running alongside the Windows kernel, WSL 2.0 offers faster file system performance, reduced overhead for system calls, and improved networking capabilities compared with WSL 1.0.This resulted in a more responsive and efficient development environment.

**2. Access Full Linux Kernel:** Unlike WSL 1.0, which relies on a translation layer to map Linux system calls to Windows equivalents, WSL 2.0, which includes a genuine Linux kernel.This enables compatibility with a wider range of Linux software and ensures a more accurate behaviour for applications that depend on specific kernel features or modules.

**3. Seamless Docker Integration:** WSL 2.0 seamlessly integrates with Docker, allowing developers to run Linux containers directly in the WSL environment.This simplifies the development and testing of containerised applications by providing a consistent platform across the Windows and Linux environments.

**4. Enhance Development Workflows:** WSL 2.0 provides developers with a familiar Linux environment for building, testing, and debugging software, while retaining access to Windows-specific tools and applications. This enables a more integrated development workflow, and facilitates collaboration across diverse development teams.

**5. Improve System Administration:** System administrators can leverage WSL 2.0 to manage Linux servers and environments more efficiently from a Windows workstation. They can perform tasks such as scripting, monitoring, and troubleshooting using native Linux tools, thereby enhancing productivity and simplifying cross-platform management.
6. Educational and Learning Opportunities: WSL 2.0 serves as an educational platform for students and professionals to learn about Linux-based development and system administration skills.It provides a sandbox environment in which

users can experiment with Linux commands, programming languages, and system configurations without the need for dedicated hardware or virtual machines.

Overall, WSL 2.0 represents a significant advancement in Microsoft's effort to bridge gap between Windows and Linux environments. It offers improved performance, enhanced compatibility, and greater flexibility, empowering users to work more efficiently in diverse computing environments.

**Key Advancements**

WSL 2.0, or Windows Subsystem in case of Linux 2.0, represents a significant advancement in Microsoft's effort to bridge gap between Windows along with Linux environments. The following are some key advancements and implications:

**1. Performance Improvement:** One of the most significant enhancements in WSL 2.0 is the shift from a translation layer approach to a virtual machine-based architecture. This change provides a substantial boost in performance, particularly for file system operations and networking tasks, as it allows native Linux kernel functionality. Users can expect faster file I/O operations and improved overall system responsiveness.

**2. Full Linux Kernel:** Unlike WSL 1.0, which translates Linux system calls intoWSL 2.0, which includes a full Linux kernel running alongside the Windows kernel. This enables compatibility with a broader range of Linux applications, including those that rely on specific kernel features or modules.

**3. Improved Docker Integration:** With WSL 2.0, Docker containers can operate seamlessly alongside other Linux applications, leveraging the same Linux kernel instance.This integration simplifies the development workflows for developers who work with containerised applications, as they can use Docker commands natively within the Linux environment.

**4. Enhanced Compatibility:** WSL 2.0 offers improved compatibility with a wider range of Linux software owing to its more accurate Linux kernel implementation. Developers and system administrators can leverage popular Linux tools, libraries, and programming languages within their Windows environment without significant modifications.

**5. Development Workflow Enhancement:** For developers, WSL 2.0 streamlines the process of building and testing software across different platforms. It enables developers to use familiar Linux development tools and workflows while retaining access to Windows-specific tools and applications, thereby providing a more integrated development experience.

**6. Cloud and DevOps Integration:** WSL 2.0 facilitates smoother integration with cloud platforms and DevOps workflows because developers can utilise Linux-based command-line tools and utilities directly within their Windows environment.This enables the seamless deployment, management, and automation of cloud resources and infrastructure.

**7. Implications for System Administrators:** System administrators can leverage WSL 2.0 to manage Linux servers and environments more efficiently from a Windows workstation. Tasks such as scripting, monitoring, and troubleshooting can be performed using native Linux tools to enhance productivity and reduce the need for context switching between different operating systems.

**8. Educational and Learning Opportunities:** WSL 2.0, which can serve as a valuable educational tool for students and professionals seeking to learn Linux-based development and system administration skills.It provides a sandbox environment in which users can experiment with Linux commands, programming languages, and system configurations without the need for dedicated hardware or virtual machines.

Overall, WSL 2.0 represents a significant step forward in Microsoft's efforts to create a more cohesive development and deployment ecosystem for Windows users while maintaining compatibility with the rich Linux software ecosystem. It offers improved performance, enhanced compatibility, and greater flexibility, empowering developers and system administrators to work more efficiently in diverse computing environments.

## LITERATURE REVIEW

### Evolution

Research related to (WSL) 2.0 primarily focuses on several key areas, including performance analysis, software compatibility, development workflows, system administration, and educational applications.Here is a brief overview of existing research and studies in these areas.

**1. Performance Analysis:** Researchers have conducted performance evaluations comparing WSL 2.0 with other virtualization solutions and previous versions of WSL.These studies typically measured factors such as file system performance, CPU utilisation, memory usage, and application responsiveness to assess the impact of WSL 2.0's architecture on system performance.

**2. Software Compatibility:** Studies have investigated the compatibility of various Linux applications and development tools using WSL 2.0.Researchers analysed the behaviour of applications running on WSL 2.0, compared to native Linux environments, to identify potential compatibility issues and assess the overall usability of the platform for software development and system administration.

**3. Development Workflows:** Research explores how WSL 2.0 affects development workflows for software developers working in heterogeneous environments. This includes studies on integrating WSL 2.0 with popular development tools and frameworks, evaluating the impact on build times and code deployment, and assessing developer productivity in mixed windows/Linux environments.

**4. System Administration:** Research examines the use of WSL 2.0, for system administration tasks, such as managing Linux servers and infrastructure from a Windows workstation.Studies may focus on scripting, automation, monitoring, and troubleshooting capabilities provided by WSL 2.0, and evaluate its effectiveness in streamlining administrative workflows.

**5. Educational Applications:** Some research has investigated the educational value of WSL 2.0, as a platform for teaching Linux-based development and system administration skills.Studies may explore how WSL 2.0 is used in educational settings, its effectiveness in facilitating hands-on learning experiences, and its impact on student and learning outcomes.

**6. Security and Privacy Implications:** Researchers may also examine the security and privacy implications of using WSL 2.0, including potential vulnerabilities introduced by the integration of a Linux kernel into the Windows operating system and the impact on system isolation and access control mechanisms.

Overall, research related to WSL 2.0 contributes to a deeper understanding of its capabilities, limitations, and implications for various applications, including software development, system administration, and education.It provides valuable insights into how WSL 2.0 can be effectively utilised and integrated into existing workflows and environments, as well as areas for further improvement and development.

**Existing research**

An integral part of Microsoft's Windows 10 operating system, Windows Subsystem for Linux (WSL) was first introduced with the Anniversary Update. In its original form, it allowed the host operating system to run native Linux applications more easily. Volatility and other existing memory forensic frameworks are designed to enable a single operating system type per analysis task, such as the execution of a single framework plugin. Problems arise for these frameworks when Windows environments add Linux executable support. Since WSL finds Linux forensic artifacts, such ELF executables, in a Windows computer's physical memory sample, it disrupts this analytical paradigm. Further, Windows Workspace Language (WSL) integrates data structures unique to Linux with data structures that are already present in Windows. Process information and userland runtime data are both stored in these structures. All of the existing analysis plugins provide contradictory results when comparing native Windows processes to WSL processes because of this integration. The lack of documentation for a substantial chunk of the WSL subsystem's internals further adds complexity to this situation. An investigation was conducted to determine whether current volatility plugins are affected by WSL and which updates are necessary to fully allow WSL's memory forensics. The current shortcomings of the WSL analysis were addressed in this way. These actions are detailed in this paper. Our research into the data architectures of WSL-relevant OSes and the development of novel volatility monitoring plugins are examples of these endeavors [1].

Microsoft has little documentation for the WSL architecture's core components, and these components are not open source. The data structures and algorithms that WSL would use are not described in Microsoft's MSDN or Windows Internals 7th Edition (Yosifovich et al., 2017), however these sources do include documentation of high-level design concepts and exported APIs. Also, the WSL subsystem does not come with full Visual Studio debugging files (PDB files) from Microsoft.

Alex Ionescu is only person who has conducted significant memory analysis research on WSL, and his findings were published in Blackhat 2016 (Ionescu, 2016a). A repository on Github contains a code that is connected to this endeavour and is open to the general public.The code is in form of WinDbg scripts (Ionescu 2016b).

Michael Ligh, who is part of the volatility development team, contributed a patch set that allowed us to properly report the identities of WSL processes in conjunction with our study (Ligh, 2017). With these updates, WSL process names may be reported accurately.

**Cygwin is a Windows environment that runs Linux.**
With the introduction of Windows Server Language (WSL), it became possible to run Linux programs on Windows laptops. One such software project is Cygwin, which allows users to execute Linux programs in Windows settings. Using the Cygwin terminal, which provides a shell environment, users may access a virtual file system, execute supported applications, and make POSIX system calls (Cygwin, 2017). Cygwin and WSL were both developed with the goal of bringing Linux environments to Windows PCs using lightweight virtualization. Here we can see one comparison between the two designs. Several various approaches, each with its own unique features, provide this possibility. When you compile Linux source code using Cygwin, the resulting executables will use the standard PE format. The next step is to link these executables with a library that provides POSIX compatibility by translating system calls between Unix and Windows. It should be noted that Cygwin does not need any kernel components to finish its activities in user space and does not contain ELF files in Windows. In contrast, WSL incorporates userland and kernel space components, is more tightly integrated, and can execute ELF files.

An approach for lightweight virtualization known as application sandboxing was the primary focus of the research efforts of the Drawbridge project team at Microsoft. Incorporating a library OS model into a commercial Windows version was the project's stated goal (Baumann et al., 2016). This architecture relocates the OS requirements of the sandboxed applications to their own process address spaces. According to research by Porter et al. (2011), Drawbridge was the pioneer in developing a library OS prototype of Windows 7.

In addition to supporting Microsoft's legacy NT processes, Drawbridge introduces two new process types: minimal and pico. The critical window components that directly link NT processes to the kernel are absent from minimal processes, in contrast to NT processes. These parts are shown in Figure 1. The absence of any kind of kernel administration and the presence of empty userland memory are two hallmarks of minimal processes. Along with being the smallest process type, pico processes also need a corresponding kernel driver. A pico process's kernel driver is the program that manages the process's userland memory, threads, scheduling, file handles, and sockets (Hron, 2017; Hammons, 2016). The term "pico provider" is often used to describe this specific driver.

Windows Server Language (WSL), the most prominent use of pico processes in Windows, was released in 2017 with the 64-bit version of Windows 10 Fall Creator Update after over a year of beta testing (Turner, 2017). Direct execution of userland Linux apps on Windows 10 is made possible by detecting each Linux application running in a pico process. This way, users may execute ELF binaries directly from the source code, eliminating the requirement for a virtual machine and other intermediate programs. Also, as of recently, Microsoft Store has apps available for all five Linux variants (Cooley et al., 2017). Linux Mint, Ubuntu, Debian, openSUSE Leap42, SUSE Linux Enterprise Server12, and Kali Linux are the distributions in question.

It will launch the WSL NT services and an/initpico process for the user if their Linux instance does not already have them. The lxss service may become a pico provider in Windows by registering with the kernel using the PsRegisterPicoProvider system function. The kernel is authorized to allow lxss to manage system calls, exceptions, and resources for the WSL pico processes by means of this instruction (Hammons, 2016a). A graphical user interface for Linux shell is created whenever wsl.exe is used, whether from inside cmd.exe or from the Windows GUI. There is also an other way to achieve the same outcomes by launching wsl.exe with the -C argument, which will run an ELF binary file and immediately return the caller process (Cooley, 2017).

## PROBLEM STATEMENT

While the (WSL) 2.0 brings significant advancements and implications for users, developers, and system administrators, several challenges accompany its adoption and implementation.The following are some of the key challenges.
**1. Performance Overhead:** Despite improvements over WSL 1.0, WSL 2.0 still incurs performance overhead compared to running Linux natively on hardware or within a traditional virtual machine environment.This overhead may affect resource utilisation, application responsiveness, and overall system performance, particularly for computationally intensive workloads.

**2. Compatibility Issues:** Although WSL 2.0 aims to provide better compatibility with Linux applications, certain software may still exhibit unexpected behaviour or compatibility issues within the WSL environment.This can be due to differences in system libraries, kernel modules, or other dependencies between the Windows and Linux environments.

**3. File System Interoperability:** While WSL 2.0 offers improved file system performance, interoperability between Windows and Linux file systems can still pose challenges. File permissions, symbolic links, and file metadata may behave differently across operating systems, leading to potential data inconsistencies or compatibility issues, especially in multiplatform development workflows.

**4. Networking Limitations:** WSL 2.0, which introduces enhanced networking capabilities compared to WSL 1.0, but certain network configurations and protocols may still be challenging to implement or may not work as expected within the WSL environment.This can affect networking-dependent applications such as web servers, databases, and distributed systems.

**5. Resource Consumption:** Running a full Linux kernel alongside the Windows kernel consumes additional system resources, including CPU, memory, and disk space. This resource consumption may be a concern, particularly for users with limited hardware resources or for those running multiple virtualised environments concurrently.

**6. Security Risks:** Integrating a Linux kernel into the Windows operating system introduces potential security risks and attack vectors. Researchers and adversaries may exploit vulnerabilities in the Linux subsystem to gain unauthorised access to system resources, escalate privileges, or execute malicious codes, highlighting the importance of maintaining robust security measures and regular updates.

**7. Development and Debugging Tools:** While WSL 2.0 facilitates development workflows by providing a Linux-like environment on Windows, developers may encounter challenges in configuring and debugging applications within the WSL environment. Differences in debugging tools, environmental variables, and system libraries between Windows and Linux can complicate the development process and hinder productivity.

**8. Training and Support:** Adopting WSL 2.0 may require users, developers, and system administrators to acquire new skills, knowledge, and best practices to effectively manage and troubleshoot Linux-based environments within a Windows ecosystem.Training and support resources may be limited or fragmented, making it challenging for users to address technical issues or optimise their workflows.

Addressing these challenges requires ongoing collaboration among Microsoft, the open-source community, and industry stakeholders to improve the performance, compatibility, security, and usability of WSL 2.0, while providing comprehensive documentation, training, and support for users transitioning to this hybrid computing environment.

## SCOPE OF WORK

The future scope of work for (WSL) 2.0 is promising, with several potential areas of development and improvement. Some future directions and possibilities for WSL 2.0:

**1. Enhanced performance optimisation:** Continued optimisation of performance to further reduce overhead and improve responsiveness, particularly for CPU- and memory-intensive workloads. Fine-tuning file system performance and reducing latency for file I/O operations.

**2. Improved Integration with Windows Ecosystem:** Deeper integration with Windows development tools and frameworks, enabling seamless collaboration and interoperability between Windows and Linux environments. Enhanced support for graphical applications and desktop environments, making it easier to run Linux GUI applications on Windows.

**3. Expansion of Supported Linux Distributions:** Increasing the selection of Linux distributions available through Microsoft Store to provide users with more choices and flexibility in selecting their preferred distribution.Ensuring compatibility with a wider range of Linux software and libraries, including specialised distributions tailored for specific use cases.

**4. Security Enhancements:** Strengthening security measures to mitigate potential vulnerabilities and threats associated with running a Linux kernel within the Windows environment and implementing additional security features such as enhanced access controls, sandboxing mechanisms, and secure boot support.

**5. Containerisation and Cloud Integration:** Further integration with containerisation technologies such as Docker and Kubernetes, enabling seamless deployment and management of containerised applications within WSL 2.0, facilitating integration with cloud platforms and services, and allowing users to leverage WSL 2.0, for cloud-native development and deployment workflows.

**6. Development Workflow Improvements:** Enhancing development workflows by providing better support for debugging, profiling, and testing within the WSL environment and streamlining the setup and configuration of development tools and frameworks, making it easier for users to begin with WSL 2.0.

**7. Enterprise Adoption and Management:** Addressing the needs of enterprise users by providing features for centralised management, deployment, and monitoring of WSL 2.0 instances across large-scale deployments, and enhancing compatibility with enterprise security policies, compliance requirements, and identity management systems.

**8. Community Collaboration and Contribution:** Encouraging community involvement and contributions to the development and improvement of WSL 2.0 through open-source collaboration, feedback channels, developer engagement initiatives, and leveraging community-driven innovation to identify emerging use cases, address user needs, and prioritise feature development.

Overall, the future of WSL 2.0 is characterized by ongoing innovation, collaboration, and improvement, as Microsoft continues to invest in enhancing the capabilities, performance, and usability of the platform to meet the evolving needs of developers, system administrators, and users in diverse computing environments.

## REFERENCES

[1]. Lewis, N., Case, A., Ali-Gombe, A., & Richard III, G. G. (2018). Memory forensics and the windows subsystem for linux. Digital Investigation, 26, S3-S11.
[2]. Yosifovich, P., Russinovich, M. E., Ionescu, A., & Solomon, D. A. (2017). Windows Internals: System architecture, processes, threads, memory management, and more, Part 1. Microsoft Press.
[3]. Ionescu, A. (2016). The Linux kernel hidden inside Windows 10.
[4]. Ligh, M. (2017). Patches to Volatility to Correctly Parse Pico Process Names.
[5]. https://cygwin.com
[6]. Baumann, Z., Zill, B., Galen, H., Lorch, J., &Olinsky, R. (2016). Drawbridge.
[7]. Porter, D. E., Boyd-Wickizer, S., Howell, J., Olinsky, R., & Hunt, G. C. (2011). Rethinking the library OS from the top down. In Proceedings of the sixteenth international conference on Architectural support for programming languages and operating systems (pp. 291-304).
[8]. Hammons, J. (2016). Pico Process Overview. Windows Subsystem for Linux.
[9]. Turner, R. (2017). Windows Subsystem for Linux Out of Beta! Windows Command Line Tools for Developers.
[10]. S. Cooley, S. Hanselman, A. McBee, R. Kottmann, A. Nikoli, "Windows 10 Installation Guide" , 2017
[11]. Laviola, S., &Levizzani, V. (2011). The 183-WSL fast rain rate retrieval algorithm: Part I: Retrieval design. Atmospheric Research, 99(3-4), 443-461.
[12]. Hegg, C., McArdell, B. W., &Badoux, A. (2006). One hundred years of mountain hydrology in Switzerland by the WSL. Hydrological Processes: An International Journal, 20(2), 371-376.
[13]. Laviola, S., Levizzani, V., Cattani, E., & Kidd, C. (2013). The 183-WSL fast rain rate retrieval algorithm. Part II: Validation using ground radar measurements. Atmospheric research, 134, 77-86.
[14]. Bohn, T. J., Melton, J. R., Ito, A., Kleinen, T., Spahni, R., Stocker, B. D., ...& Kaplan, J. O. (2015). WETCHIMP-WSL: intercomparison of wetland methane emissions models over West Siberia. Biogeosciences, 12(11), 3321-3349.
[15]. Singh, P., & Singh, P. (2020). Linux development on WSL. Learn Windows Subsystem for Linux: A Practical Guide for Developers and IT Professionals, 131-168.
[16]. Enescu, I. I., Fraefel, M., Plattner, G. K., Espona-Pernas, L., Haas-Artho, D., Lehning, M., & Steffen, K. (2018). Fostering Open Science at WSL with the EnviDat Environmental Data Portal (No. e27211v1). PeerJ Preprints.
[17]. Zunin, V. V., Stempkovsky, A. L., &Solovyev, R. A. (2024, March). CAD Architecture for Expansion of WSL-Based Combinational Circuits Dataset. In 2024 International Russian Smart Industry Conference (SmartIndustryCon) (pp. 294-298). IEEE.
[18]. Blanco, A. F., Bergel, A., &Alcocer, J. P. S. (2022). Software visualizations to analyze memory consumption: A literature review. ACM Computing Surveys (CSUR), 55(1), 1-34.
[19]. Barthe, G., Pavlova, M., & Schneider, G. (2005, September). Precise analysis of memory consumption using program logics. In Third IEEE International Conference on Software Engineering and Formal Methods (SEFM'05) (pp. 86-95). IEEE.
[20]. Thomas, O. (2020). Windows server 2019 inside out. Microsoft Press.
[21]. Gao, Y., Liu, Y., Zhang, H., Li, Z., Zhu, Y., Lin, H., & Yang, M. (2020, November). Estimating GPU memory consumption of deep learning models. In Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 1342-1352).
[22]. Sreelatha, K., & Krishna Reddy, V. (2021). Integrity and memory consumption aware electronic health record handling in cloud. Concurrent Engineering, 29(3), 258-265.
[23]. [23] Chu, D. H., Jaffar, J., &Maghareh, R. (2016, June). Symbolic execution for memory consumption analysis. In Proceedings of the 17th ACM SIGPLAN/SIGBED Conference on Languages, Compilers, Tools, and Theory for Embedded Systems (pp. 62-71).