Windows 11 Home 環境における Claude Code 利用完全ガイド

I. はじめに: Windows システムでの Claude Code 活用

A. Claude Code とは何か、どのように役立つのか？

Claude Code は、Anthropic 社によって開発された「エージェント型コーディングツール」であり、ターミナル内で動作し、利用者のコードベースを理解し、自然言語による指示を通じてコーディング作業を高速化するものです [1]。このツールは、追加のサーバーや複雑なセットアップを必要とせず、開発環境に直接統合されるように設計されています [1]。

Claude Code の主な機能には、コードベース全体のファイル編集やバグ修正、コードのアーキテクチャやロジックに関する質問への回答、テストの実行と修正、リンティング、Git 操作の管理（履歴検索、マージコンフリクトの解決、コミットやプルリクエストの作成など）、さらにはウェブ検索を通じたインターネット上のドキュメントやリソースの閲覧が含まれます [1]。プロジェクトの文脈を理解し、実際にアクションを実行することで、手動でファイルをコンテキストに追加する必要性をなくし、ワークフローを合理化することを目指しています [1]。

特筆すべきは、Claude Code が単なる高度な自動補完ツールを超えている点です。「エージェント型」という性質は、このツールが利用者の意図を理解し、計画を立て、コードベース内で複数のステップからなるタスクを実行できることを示唆しています。これは、従来のコーディングアシスタントからの大きな進歩と言えるでしょう。「実際にアクションを実行する」[1] 能力や、「コードベース全体のファイル編集やバグ修正」[2] といった複雑な操作を実行できる点は、この高度な能力を裏付けています。このような自律性の高さは、開発者がより複雑なタスクを委任できることを意味する一方で、ツールに提供するコンテキストや権限について、より注意深くなる必要があることも示しています。

また、Claude Code の設計思想の中心には、開発者の既存のワークフローへのシームレスな統合があります。「ターミナル内で動作する」[1]、「コードベースを理解する」[1]、そして「開発環境に直接統合される」[1] という特徴は、既存の開発習慣への影響を最小限に抑え、AI アシスタンスを自然な形で組み込むことを目指している証左です。多くの開発者は、確立されたワークフローを大幅に変更する必要があるツールに対して抵抗を感じる傾向があります。Claude Code は、ターミナルでの操作 [1] や IDE との統合 [3] を提供し、手動でのファイル指定なしに「必要に応じてコードベースを探索する」[1] ことで、この導入障壁を下げ、開発者の認知的負荷を軽減しようとしています。

B. Windows で Claude Code を利用するための Windows Subsystem for Linux (WSL) の不可欠な役割

Claude Code は、その動作に Linux 環境を明示的に要求します。Windows オペレーティングシステム上でこれを実現する手段が、Windows Subsystem for Linux (WSL) です [2]。重要な点として、Claude Code は Windows 上でネイティブには動作しません [5]。この前提条件を

明確に理解しておくことが、期待値を管理する上で不可欠です。

WSL の要件は、強力な機能を提供する一方で、これまで WSL を使用したことのない Windows 開発者にとっては、初期設定のレイヤーが追加されることになり、これが馴染みがなく複雑に感じられる可能性があります。本ガイドでは、この点を最大限明確に説明することを目指します。利用者が Windows 11 Home Edition を使用しているという事実は、この文脈で特に重要です。「Windows via WSL」[2] という要件や、「Windows 上でネイティブには動作しない」[6] という明確な記述は、セットアップの主要な課題が Claude Code 自体ではなく、その基盤となる環境の構築にあることを意味します。したがって、本ガイドのかなりの部分が、WSL のセットアップを可能な限りスムーズに行うための説明に割かれます。

さらに、Windows 上での Claude Code のパフォーマンスは、WSL 環境のパフォーマンスと設定に本質的に関連しています。推奨されるバージョンである WSL2 は、軽量なユーティリティ仮想マシン内で実際の Linux カーネルを実行します [7]。Windows と WSL 間のファイルシステム相互作用は、適切に管理されない場合、パフォーマンスのボトルネックになる可能性があります [9]。Claude Code はコードベースと集中的にやり取りするため [1]、最適なパフォーマンスを得るためには、プロジェクトファイルを WSL ファイルシステム内に配置することが極めて重要になります。この点は、後のトラブルシューティングやベストプラクティスのセクションで詳述します。

II. フェーズ 1: Windows 11 Home 環境の準備

A. Claude Code および WSL のシステム前提条件の確認

Claude Code と WSL を円滑に導入・利用するためには、まずシステムがこれらの前提条件を満たしているかを確認する必要があります。

WSL の基本的な要件として、Windows 10 バージョン 2004 (ビルド 19041) 以降、または Windows 11 が必要です [8]。利用者は Windows 11 Home Edition を使用しているため、この点は満たされています。

Claude Code 固有の要件は以下の通りです。

- **オペレーティングシステム:** macOS 10.15 以降、Ubuntu 20.04+/Debian 10 以降、または Windows (WSL 経由) [2]。
- **ハードウェア:** 最低 4GB の RAM [2]。
- **ソフトウェア:** Node.js バージョン 18 以降 (WSL 内にインストール、これは非常に重要) [2]。オプションとして、git バージョン 2.23 以降、GitHub/GitLab CLI、ripgrep (rg) が挙げられます [2]。
- **ネットワーク:** 認証および AI 処理のためにインターネット接続が必要です [2]。

これらの前提条件を簡単に確認できるよう、以下のチェックリストを用意しました。インストール

作業を開始する前に、各項目を確認することで、後々の問題を未然に防ぐことができます。

表 1: システム要件チェックリスト

コンポーネント	要件	確認 (利用者記入)	備考
オペレーティングシステム	Windows 11 (Home Edition は利用者確認済み)	☐	
WSL 機能	有効化が必要	☐	
RAM	最低 4GB	☐	
Node.js	バージョン 18+ (WSL 内にインストール)	☐	Claude Code CLI にとって極めて重要
Git (オプション)	バージョン 2.23+ (WSL 内、推奨)	☐	Claude Code 内での Git 関連コマンドに利用
Ripgrep (オプション)	rg (WSL 内、ファイル検索強化のため)	☐	Claude Code のファイル検索能力を向上
インターネット	安定した接続	☐	インストール、認証、AI 処理に必須
Anthropic アカウント	作成が必要 (有効な支払い設定が必須)	☐	認証および API キー取得のため

Git や ripgrep といったツールは「オプション」として記載されていますが [2]、これらは開発者の一般的なタスク（例えば、「Git 履歴の検索」[1] や「強化されたファイル検索」[2]）において Claude Code の有用性を大幅に向上させます。これらのツールがない場合、宣伝されている一部の機能が低下したり利用できなくなったりする可能性があるため、WSL 内へのインストールを強く推奨します。

B. ステップバイステップ**: Windows 11 Home** での **WSL** の有効化と **Ubuntu** のインストール

WSL の有効化と Ubuntu のインストールは、管理者権限で実行する PowerShell または Windows コマンドプロンプトを通じて行います [6]。

主要なコマンドは wsl --install です [6]。このコマンドは、必要な Windows の機能（仮想マシン

プラットフォームなど）を有効にし、デフォルトで Ubuntu を Linux ディストリビューションとしてインストールします。この初期インストールの後、通常はシステムの再起動が求められます [7]。

もし wsl --install コマンドを実行した際にヘルプテキストが表示された場合（これは、WSL の一部コンポーネントが既に存在するか、古いシステムである可能性を示します）、wsl --list --online を実行して利用可能なディストリビューションの一覧を表示し、wsl --install -d <DistroName>（例: wsl --install -d Ubuntu-24.04）のようにして特定のバージョンをインストールできます [7]。Ubuntu は Claude Code でサポートされており、良いデフォルト選択肢です [2]。Microsoft Store アプリケーションを使用して WSL 上に Ubuntu をインストールすることも可能です [7]。

近年の wsl --install コマンドは、WSL のセットアッププロセスを大幅に簡素化しました [8]。これは、以前の手動で複数の Windows 機能を有効にする必要があった方法と比較して大きな改善点であり、利用者にとってはプロセスがより直接的になったことを意味します。

C. WSL 内 Ubuntu の初期設定

新しくインストールされた Linux ディストリビューション (Ubuntu) を初めて起動すると、コンソールウィンドウが開き、ファイルの展開と保存が完了するまで待機するよう求められます [8]。

その後、Ubuntu 環境用の UNIX ユーザー名とパスワードを作成する必要があります [6]。これらの認証情報は、Windows の認証情報と一致させる必要はありません。WSL Ubuntu 環境が Windows のログインとは独立した独自のユーザーアカウントを持っていることを理解することが重要です。これは Linux 環境内でのセキュリティと権限管理の基本であり、主に Windows の単一ログインに慣れている利用者にとっては混乱のポイントとなる可能性があるため、これらの認証情報が Linux 環境専用であることを明確にしておきます。

D. WSL Ubuntu 環境への Node.js (バージョン 18+) のインストール

Claude Code は Node.js バージョン 18 以降を要求します [2]。WSL Ubuntu 環境内に Node.js と npm をインストールするには、Node Version Manager (nvm) の使用を推奨します。これは、複数の Node.js バージョンをインストールし、必要に応じて簡単に切り替えるための最も一般的な方法です [12]。

WSL Ubuntu ターミナル内で nvm と Node.js をインストールする手順は以下の通りです [12]:

1. パッケージリストを更新します: sudo apt update (これは一般的な良い習慣であり、[13] からも示唆されます)。
2. cURL をインストールします: sudo apt-get install curl [12]。
3. nvm をインストールします: curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/master/install.sh | bash [12]。nvm の GitHub リポジトリで最新のバージョンとコマンドを確認することを推奨します。

4. シェル設定ファイルを読み込むか（例: source ~/.bashrc）、ターミナルを一度閉じて再度開くことで、nvm コマンドが利用可能になります。
5. nvm のインストールを確認します: command -v nvm [12]。
6. Node.js 18 以降をインストールします (例: 最新の LTS バージョン、または特定のバージョン): nvm install --lts (多くの場合 18 以上です) または nvm install 18 [12]。
7. Node.js と npm のインストールを確認します: node -v および npm -v。

WSL 内から Windows にインストールされた Node.js を使用すると、Claude Code で exec: node: not found エラーやプラットフォーム検出の問題が発生する可能性があるため、避けるべきです [2]。which npm および which node コマンドの実行結果は、/mnt/c/... のような Windows パスではなく、Linux パス (例: /usr/ や /home/user/.nvm/...) を指している必要があります [2]。

Node.js を WSL 環境 内部 に正しくインストールし、Claude Code がこのインストールを使用するようにすることが最も重要です。Windows にインストールされた Node.js を WSL から使用することは、よくある落とし穴です。[2] は、WSL が Windows の npm を使用した場合の「OS/プラットフォーム検出の問題」や、WSL 環境が Windows の Node.js インストールを使用した場合の「Node が見つからないエラー」について具体的に警告しています。これは、Node.js 環境が WSL 内で完全に内包され、管理される必要があることを明確に示しています。この目的のためには、nvm [12] が Linux 環境内でインストールとパス管理を正しく処理するため、ベストプラクティスとなります。

さらに、nvm の使用を推奨することは [12]、Node.js 18+ という当面の要件を満たすだけでなく、開発者が他のプロジェクトで異なる Node.js バージョンを管理できるようにするという点で、将来を見据えた対応でもあります。これは開発者にとって一般的な要件です。「バージョンは非常に速く変更されるため、バージョンマネージャーを使用することをお勧めします。作業しているさまざまなプロジェクトのニーズに基づいて、複数のバージョンの Node.js を切り替える必要があるでしょう」という記述は [12]、Claude Code の当面の要件を満たすだけでなく、nvm を堅牢なソリューションとして推奨することを直接的に支持しています。

III. フェーズ 2: Claude Code のインストールと認証

A. Anthropic アカウント、API キー、および支払い設定のセットアップ

Claude Code を利用するには、まず Anthropic のプラットフォームでアカウント設定と支払い準備を整える必要があります。

1. アカウント作成: Anthropic のコンソール (console.anthropic.com) でアカウントを作成します [2]。
2. **API キーの生成:**
   - Anthropic コンソールのサイドバーまたはダッシュボードで「API Keys」に移動します [14]。

    - 「Create New API Key」または同様のボタンをクリックし、キーに名前を付けて安全な場所にコピーします。**API** キーは作成時に一度しか表示されないため、必ず保管してください [14]。この事実は、API キーを即座に安全に保管する必要性を強調しています。紛失するとアクセスできなくなり、キーを再生成する必要が生じます。
3. 支払い設定:
    - Claude Code のデフォルトの認証方法では、console.anthropic.com での有効な支払い設定が必要です [1]。一部の無料枠が豊富な開発者向け CLI ツールとは異なり、Claude Code の主要な認証ルートは有効な支払い設定に紐づいていることを、このセクションの早い段階で明確にすることが非常に重要です。
    - API の利用料金は、前払いの「利用クレジット」を通じて請求されます [17]。API を利用する前にクレジットを購入する必要があり、これはユーザーが Claude Code をダウンロードして実行するだけでは不十分で、まず Anthropic の商用プラットフォームを利用する必要があることを意味します。これはユーザーの期待値を管理する上で極めて重要な情報です。
    - コンソールの「Billing」または「Plans & Billing」セクションで、支払い情報（会社情報、利用目的、支払い詳細など）を設定します [14]。
    - クレジットの自動リロードを設定することも可能です [16]。
4. **Claude Pro/Max** プランによる **Claude Code** アクセス: 「Pro」プラン（月額 $20 または年間契約で月額 $17）には、「ターミナルで Claude Code に直接アクセス」する権利が含まれています [1]。「Max」プランも同様です。無料ティアのユーザーは、[18] に基づくと、直接的なターミナル利用が制限されるか、アクセスできない可能性があります。新規 API ユーザーは、テスト用に少額の無料クレジットを受け取れる場合があります [16]。Claude Code を定期的に利用する予定がある場合、Pro プラン [18] がアクセスを確保するための最も簡単な方法であり、利用パターンによっては都度払いの API クレジットよりも価値が高い可能性があります。
5. コスト意識: 一般的な Claude Code の利用コストは、開発者1人あたり1日 $5〜$10 の範囲ですが、集中的な利用時には1時間あたり $100 を超えることもあります [2]。モデルの料金は異なり、例えば Claude Sonnet 4 は入力 $3/MTok、出力 $15/MTok ですが、Claude Opus 4 はより高価です [18]。

B. Claude Code コマンドラインインターフェース **(CLI)** のインストール

Anthropic アカウントと支払い設定が完了したら、WSL Ubuntu 環境に Claude Code CLI をインストールします。

1. WSL Ubuntu ターミナルを開きます。
2. この段階では必須ではありませんが、プロジェクトディレクトリに移動しておくことを推奨します: cd your-project-directory [2]。
3. インストールコマンドを実行します: npm install -g @anthropic-ai/claude-code [1]。
4. 極めて重要な注意点として、**npm install -g** コマンドに **sudo** を使用しないでください [2]。

これはパーミッションの問題やセキュリティリスクを引き起こす可能性があります。パーミッションエラーが発生した場合は、npm のプレフィックス設定やパーミッションに関する公式のトラブルシューティング情報を参照してください。[2] で sudo npm install -g の使用を避けるよう強く警告されているのは非常に重要です。Linux や npm に不慣れな多くのユーザーは、パーミッションエラーを解決するために直感的に sudo を使用しがちですが、これが後々より大きな問題を引き起こす可能性があります。このガイドでは、この点を強調し、必要に応じて正しい npm パーミッション設定を案内します。

C. Anthropic アカウントによる Claude Code の認証

CLI のインストール後、Claude Code を Anthropic アカウントと連携させて認証を行います。

1. まだ移動していない場合は、プロジェクトディレクトリに移動します。
2. WSL ターミナルで claude と入力して Claude Code を起動します [2]。
3. Anthropic Console アカウントを使用した1回限りの OAuth プロセスに従います [1]。これには通常、ブラウザウィンドウが開き、ログインしてアクセスを承認する手順が含まれます。OAuth の利用 [2] は、初期設定時に CLI で直接 API キーを手動設定するのに比べて認証プロセスを簡素化します。これは一般的に、ユーザーの主要な認証情報を直接扱うことなくアプリケーション (Claude Code) が限定的なアクセス権を取得できるため、より安全でユーザーフレンドリーなメカニズムです。
4. 前述の通り、このデフォルトの方法では console.anthropic.com での有効な支払い設定、またはその認証ルートを選択した場合は有効な Claude Pro/Max プランが必要です [1]。

IV. フェーズ 3: Visual Studio Code とのシームレスな連携 (開発者推奨)

Claude Code はターミナルファーストのツールですが、多くの開発者にとって Visual Studio Code (VS Code) は主要な統合開発環境 (IDE) です。VS Code の強力な WSL 連携機能は、Claude Code との組み合わせに自然に適合します。このセクションでは、VS Code との連携手順を説明し、これを生産的なワークフローのための非常に推奨される、あるいは不可欠なセットアップの一部として位置づけます。

A. Windows への Visual Studio Code のインストール

まだインストールしていない場合は、公式ウェブサイト (code.visualstudio.com) から VS Code for Windows をダウンロードしてインストールします [12]。

インストール中に「追加タスクの選択」画面が表示されたら、「PATH に追加」オプションが選択されていることを確認してください。これにより、コマンドライン (WSL を含む) から code. コマンドを使用して VS Code を起動できるようになります [21]。

B. 必須ツール: VS Code への「Remote - WSL」拡張機能のインストール

「Remote - WSL」拡張機能は、Windows の UI と Linux のバックエンドをシームレスに繋ぐ、円滑な開発体験の鍵となります。この UI (Windows 上) と開発ツール (WSL 上) の関心の分離は強力です。これは単なる利便性だけでなく、WSL 環境で VS Code を使用した効果的な開発には不可欠な要素です。

1. Windows 上で VS Code を開きます。
2. 拡張機能ビュー (Ctrl+Shift+X) を開きます。
3. Microsoft 製の「WSL」(ms-vscode-remote.remote-wsl) を検索し、インストールします[12]。
4. この拡張機能により、VS Code は Windows 上で UI を実行しつつ、WSL 内で実行されている開発環境 (ツール、コンパイラ、拡張機能など) に接続できます[20]。

C. VS Code と WSL Ubuntu 環境の接続

WSL の利点を最大限に活用し、パフォーマンスの問題を回避するためには、プロジェクトファイルが WSL ファイルシステム内 (例: ~your_project_directory_in_wsl または /home/username/project) に存在し、そのコンテキストから VS Code で開かれることが重要です。Windows ファイルシステムパス (例: /mnt/c/...) を WSL からプライマリ開発用にアクセスすると、パフォーマンスが低下します[9]。ファイル集約型の Claude Code は、この恩恵を大きく受けます。VS Code の Remote - WSL 拡張機能は、プロジェクトルートが WSL 内にある場合に最適に動作します。

接続方法は主に2つあります。

1. **WSL ターミナルから:**
   - WSL Ubuntu ターミナルを開き、プロジェクトフォルダに移動します (cd /path/to/your/project)。
   - code. と入力します[20]。これにより、プロジェクトが VS Code で開かれ、自動的に WSL インスタンスに接続されます。
2. **VS Code から:**
   - VS Code を開きます。
   - 左下の緑色のリモートステータスバーアイテムをクリックするか、F1 キーを押して「WSL: Connect to WSL」または「WSL: Connect to WSL using Distro...」と入力します[20]。
   - 使用する Ubuntu ディストリビューションを選択します。
   - その後、ファイル > フォルダを開く を使用して、WSL ファイルシステム内にあるプロジェクトディレクトリ (例: /home/your_username/your_project) を開きます。

D. Claude Code VS Code 拡張機能の自動インストール

VS Code が WSL 環境に接続され、Claude Code CLI が WSL 内にインストールおよび認証されると、公式の VS Code 拡張機能のインストールが簡素化されます。統合ターミナルで

claude を実行した際の公式 VS Code 拡張機能の自動インストール [4] は、セットアップを簡略化するために設計されたユーザーフレンドリーな機能です。これにより、ユーザーの手動ステップ数が削減されます。

1. VS Code で統合ターミナル (Ctrl+`) を開きます。このターミナルは WSL Ubuntu ターミナルとして機能します。
2. この統合ターミナルで claude を実行します。公式の Claude Code VS Code 拡張機能が自動的にインストールされるはずです [4]。
3. 自動インストールされない場合は、[4] のトラブルシューティング情報を確認してください (例: WSL 内の PATH に code コマンドが含まれているか、VS Code が拡張機能をインストールする権限を持っているかなど)。
4. AnandTyagi 氏による代替のコミュニティ開発「Claude Code Interface」拡張機能 [23] も存在しますが、利用可能であれば公式の自動インストール [4] が推奨されます。AnandTyagi 氏の拡張機能は、まず Claude Code CLI がインストールされ認証されている必要があり、チャット形式のインターフェースを提供します。人気のあるツールでは、公式の拡張機能が登場する前や並行してコミュニティによる拡張機能が存在することは一般的です。このガイドでは、公式の連携パスを優先します。公式のものは、Claude Code の進化する機能とより良く、より深く統合されている可能性が高いためです。

V. フェーズ 4: Claude Code の第一歩

環境設定と VS Code との連携が完了したら、いよいよ Claude Code を実際に使ってみましょう。このセクションでは、基本的な起動方法から、具体的なコマンド例までを紹介します。

A. ターミナルでの Claude Code の起動と対話

Claude Code の効果は、現在のプロジェクトを理解する能力に大きく左右されます。そのため、プロジェクトのルートディレクトリから起動することが非常に重要です。

1. WSL Ubuntu ターミナル (または VS Code の WSL に接続された統合ターミナル) を開きます。
2. プロジェクトのルートディレクトリに移動します: cd /path/to/your/project。Claude Code はプロジェクトのコンテキストを理解することで最適に動作します [2]。
3. 対話型セッションを開始するには、claude と入力します [2]。
4. 単発のコマンドを実行する場合は、claude -p "your query" のように -p フラグを使用します [24]。
5. ファイルの内容をパイプで渡すことも可能です: cat somefile.txt | claude -p "summarize this" [24]。

B. コアコマンドと機能の概要

Claude Code には、CLI コマンドと対話型セッション内で利用できるスラッシュコマンドの2種類のコマンド体系があります。スラッシュコマンドは、セッションの管理、設定の構成、特定の機

能へのアクセスを、会話の流れを中断することなく行うための構造化された方法を提供します。これらは単なるショートカット以上のものであり、「エージェント型」ツールを効果的に使用するための鍵となります（例：コンテキスト管理のための /clear、AI モデルを切り替えるための /model、設定のための /config）。

以下に、新規ユーザーがすぐに利用を開始し、機能を試すのに役立つ主要なコマンドとフラグをまとめた表を示します。これは、[24] に記載されている広範なリストから、特に有用なものを抜粋したものです。

表 **2:** 主要な **Claude Code** コマンドとフラグ

タイプ	コマンド/フラグ	説明	出典
CLI	claude	対話型 REPL セッションを開始します。	[5]
CLI	claude "query"	初期クエリを指定して REPL を開始します。	[24]
CLI	claude -p "query"	単一のクエリを実行して終了します (非対話型)。	[24]
CLI	claude -c	直近の会話を続行します。	[24]
フラグ	--output-format	-p モードの出力形式を指定します (text, json, stream-json)。	[24]
フラグ	--model	セッションの AI モデルを設定します (例: sonnet, opus)。	[24]
スラッシュ	/help	ヘルプと利用可能なスラッシュコマンドを表示します。	[24]
スラッシュ	/clear	会話履歴とコンテキストをクリアします。	[24]

スラッシュ	/cost	現在のセッションのトークン使用量とコストを表示します。	[24]
スラッシュ	/init	CLAUDE.md プロジェクトガイドを初期化します。	[2]
スラッシュ	/config	Claude Code の設定を表示または変更します。	[24]
スラッシュ	/review	コードレビューを要求します (例: PR に対して、GitHub 連携で多用)。	[24]
スラッシュ	/project:optimize	カスタムプロジェクトコマンドの例 (.claude/commands に optimize.md が存在する場合)	[26]

C. 実用的な例

Claude Code の真価は、実際の開発タスクに適用することで発揮されます。以下にいくつかの具体的な使用例を示します。

1. **/init** と要約によるプロジェクト理解
   - /init コマンドを使用すると、CLAUDE.md というプロジェクトガイドファイルが生成されます [2]。このファイルは、Claude がプロジェクトの特性を理解するのを助け、カスタマイズすることも可能です [27]。CLAUDE.md ファイル [2] は、Claude に永続的でプロジェクト固有のコンテキストと指示を提供し、その振る舞いを導き、アクションの関連性を向上させる強力なメカニズムです。これは単なるドキュメントではなく、特定のプロジェクト内での AI エージェントの構成レイヤーとして機能します。
   - プロジェクトの概要を尋ねることもできます: claude "summarize this project" または claude -p "summarize this project" [1]。
2. コードベースへの問い合わせ
   - アーキテクチャやロジックについて自然言語で質問します: claude "how does our authentication system work?" [1]。
   - 関連ファイルを見つけます: claude "find the files that handle user authentication" [26]。
   - プロセスの流れを追跡します: claude "trace the login process from front-end to database" [26]。

3. バグ修正とコード編集の支援
    - エラーメッセージを共有します: claude "I'm seeing this error when I run npm test: <エラーメッセージを貼り付け>" [26]。
    - 修正案を尋ねます: claude "suggest a few ways to fix the @ts-ignore in user.ts" [26]。
    - Claude に修正の適用を指示します: claude "update user.ts to add the null check you suggested" [26]。
    - より広範な編集コマンドも可能です: claude "fix the type errors in the auth module" [1]、claude "add input validation to the signup form" [5]。
    - 画像を利用したデバッグも有効です。UI のバグやダイアグラムのスクリーンショットを貼り付けて Claude に分析させることができます [26]。効果的な利用は、多くの場合、会話的で反復的なプロセスを伴います。つまり、質問し、レビューし、改良し、再度質問するという流れです [29]。常に一発で完璧な解決策が得られるわけではありません。Claude Code を共同作業のパートナーとして扱い、望ましい結果に到達するために対話を行うことが重要です。
4. **Git** 操作の効率化
    - コミットを作成します: claude commit [1]、またはより具体的に claude "commit the generated CLAUDE.md file to your repository" [2]。
    - Git 履歴の検索、マージコンフリクトの解決、プルリクエストの作成といった機能も備えています [1]。
    - テスト駆動開発 (TDD) ワークフローもサポートします。Claude にテストを作成させ、それらが失敗することを確認し、テストをパスするコードを作成させ、その後コミットするという流れです [29]。
    - GitHub Actions との連携により、プルリクエストや ཀྱཏ་ で @claude とメンションするだけで、コードレビュー、バグ修正、機能実装などを自動化できます [3]。これには、Claude GitHub App のセットアップと ANTHROPIC_API_KEY シークレットの設定が必要です。この GitHub 連携は、Claude Code を個人のアシスタントから、コードレビューや自動修正といったタスクのためのチームレベルの自動化ツールへと変貌させ、その影響範囲を個々の開発者のターミナルをはるかに超えて広げます。

VI. フェーズ 5: Windows (WSL) 環境における一般的な問題のトラブルシューティング

Claude Code を Windows (WSL) 環境で使用する際には、特有の問題が発生することがあります。このセクションでは、よくある問題とその解決策について説明します。多くの一般的な問題は、WSL 環境が意図せず Windows ベースのツール（Node.js や npm など）を使用しようとしたり、Windows のファイルパスから操作しようとしたりすることに起因します。Claude Code の操作を WSL 内の Linux 環境およびファイルシステムコンテキスト内で厳密に維持す

ることが、これらの問題を回避する鍵となります。

A. Claude Code の WSL 固有のインストールおよび実行時問題への対処

- **OS/プラットフォーム検出の問題 (npm install 時):** WSL が Windows の npm を使用している場合にエラーが発生することがあります。
  - 解決策 **1:** インストール前に npm config set os linux を実行します [2]。
  - 解決策 **2:** npm install -g @anthropic-ai/claude-code --force --no-os-check を実行します (sudo は使用しないでください) [2]。
- **exec: node: not found エラー (claude 実行時):** WSL 環境が Windows の Node.js インストールを使用している可能性があります。
  - 確認方法**:** which npm および which node の実行結果が、Windows パス (/mnt/c/...) ではなく、Linux パス (例: /usr/..., ~/.nvm/...) を示していることを確認します [2]。
  - 解決策**:** WSL 内に、そのパッケージマネージャ (例: apt) を使用するか、より推奨される nvm を介して Node.js をインストールします (フェーズ 1D で詳述) [2]。
- 「**Claude Code not supported on Windows**」エラー**:** 利用者が WSL Ubuntu ターミナルではなく、Windows のターミナル (PowerShell, CMD) を使用している可能性が高いです [6]。
  - 解決策**:** ターミナルのプロンプトが Linux のユーザー名を示していることを確認します。uname -a を実行し (「Linux」と表示されるはずです)、スタートメニューから Ubuntu を開くか、PowerShell/CMD で wsl と入力して切り替えます [6]。
- **ENOENT エラー (ファイルが見つからない):** WSL 内の Linux ファイルシステム (~/...) ではなく、Windows ファイルシステムのコンテキスト (/mnt/c/...) からコマンドが実行された場合にしばしば発生します [6]。
  - 解決策**:** claude を実行する前に、Linux ファイルシステム内のプロジェクトディレクトリに移動します (cd ~) [6]。
- **npm パーミッション拒否 (グローバルインストール時の EACCES エラー、sudo なしの場合):**
  - 解決策**:** ユーザーが所有する別のディレクトリをグローバルインストールに使用するように npm を設定します。(これは標準的な npm のトラブルシューティングであり、パーミッションエラーが発生した場合の「推奨される解決策のために Claude Code を設定する」[5] からも示唆されます)。

B. VS Code 連携の不具合解決

VS Code の統合ターミナルは、WSL に正しく接続されている場合、Claude Code 拡張機能の自動インストールなどの機能にとって重要な橋渡し役となります。ここでの問題は、多くの場合 PATH や WSL 内の VS Code 自体のセットアップに関連しています。

- **Claude Code VS Code 拡張機能が自動インストールされない:**

- claude が VS Code の WSL に接続された統合ターミナルから実行されていることを確認します [4]。
- code コマンド (Cursor IDE の場合は cursor など) が WSL の PATH で利用可能であることを確認します。利用可能でない場合は、VS Code のコマンドパレット (Ctrl+Shift+P) を使用して「Shell Command: Install 'code' command in PATH」を実行します [4]。
- VS Code が拡張機能をインストールする権限を持っていることを確認します [4]。

C. よりスムーズな体験のための **WSL** パフォーマンス最適化

開発者が WSL を使用する上で最も影響の大きいパフォーマンス最適化は、プロジェクトファイルと開発活動を Windows/WSL の境界を越えずに、WSL Linux ファイルシステム 内 で行うことを保証することです。

- プロジェクトファイルを **WSL** ファイルシステムに保存する**:** Linux ファイルシステム (例: /home/username/project) 内のファイルへのアクセスは、WSL 内から /mnt/c/ 経由で Windows ファイルシステム上のファイルにアクセスするよりも大幅に高速です [9]。これは、Claude Code が実行する可能性のある I/O 集約的な操作にとって極めて重要です。
- **WSL** への十分な **RAM** 割り当て**:** Claude Code 自体は 4GB の RAM を必要としますが [2]、WSL もリソースを消費します。Windows 全体で十分な RAM があることを確認してください。WSL2 はデフォルトでシステム RAM のかなりの部分を使用する可能性があります。問題が発生しない限り Home Edition では不要かもしれませんが、必要に応じて .wslconfig ファイルで制限を設定することを検討してください (これは高度な設定です)。
- 定期的なクリーンアップ**:**
  - Ubuntu のパッケージマネージャキャッシュをクリアします: sudo apt clean, sudo apt autoremove [9]。
  - WSL2 で Docker を使用している場合は、未使用のコンテナとボリュームを整理します: docker system prune, docker volume prune [9]。
- **WSL1 vs. WSL2:** wsl --install のデフォルトであり、一般的に推奨されるのは WSL2 です。WSL1 は Windows ファイルシステム上の一部の I/O ヘビーなタスクでより良いパフォーマンスを提供するかもしれませんが、完全なシステムコール互換性や Docker 統合などの制限があり、完全な Linux 環境を期待する Claude Codeのようなツールには一般的に推奨されません [10]。本ガイドでは WSL2 を前提としています。

VII. まとめ**:** コーディングワークフローの強化

本ガイドでは、Windows 11 Home Edition 環境で Claude Code を利用するための詳細な手順を解説しました。WSL の設定から始まり、Claude Code のインストールと認証、Visual Studio Code との連携、そして基本的な使い方とトラブルシューティングに至るまで、段階的に説明を進めてきました。

A. 総括: Windows 開発環境における Claude Code の価値

WSL を介してセットアップされた Claude Code は、特に VS Code と組み合わせることで、Windows ベースの開発者のワークフローに統合され、生産性を向上させ、複雑なタスクを支援する強力なツールとなり得ます。自然言語による指示でコードを生成・編集したり、コードベースに関する深い洞察を得たり、Git 操作を効率化したりする能力は、日々の開発作業を大きく変革する可能性を秘めています。

B. さらなる探求: 公式リソースと高度な機能

Claude Code は進化し続けるツールであり [3]、その全機能を活用するためには、公式ドキュメントの参照が不可欠です。

- Anthropic の公式 Claude Code ドキュメントには、最新情報、チュートリアル、ベストプラクティス、高度な設定などが網羅されています [1]。
- カスタムコマンドの作成 [26]、CLAUDE.md ファイルの活用 [27]、画像分析機能 [26]、そしてカスタムエージェント構築のための Claude Code SDK [3] など、探求すべき高度な機能が多数存在します。
- コマンドリスト全体 [24] を確認し、Claude Code の可能性を最大限に引き出してください。
- claude update コマンド [24] を使用して、Claude Code を常に最新の状態に保つことを忘れないでください。

このガイドが、Windows 環境で Claude Code を活用するための一助となれば幸いです。

引用文献

1. Claude Code overview - Anthropic API, 6月 7, 2025にアクセス、https://docs.anthropic.com/en/docs/claude-code/overview
2. Claude Code overview - Anthropic API, 6月 7, 2025にアクセス、https://docs.anthropic.com/s/claude-code-security
3. Introducing Claude 4 - Anthropic, 6月 7, 2025にアクセス、https://www.anthropic.com/news/claude-4
4. IDE integrations - Anthropic, 6月 7, 2025にアクセス、https://docs.anthropic.com/en/docs/claude-code/ide-integrations
5. Claude Code overview - Anthropic API, 6月 7, 2025にアクセス、https://docs.anthropic.com/en/docs/agents/claude-code/introduction
6. How to Install Claude Code on Windows: Complete 2025 Guide - iTecs, 6月 7, 2025にアクセス、https://itecsonline.com/post/how-to-install-claude-code-on-windows
7. Install Ubuntu on WSL2 - Ubuntu on WSL documentation, 6月 7, 2025にアクセス、https://documentation.ubuntu.com/wsl/latest/howto/install-ubuntu-wsl2/
8. Install WSL | Microsoft Learn, 6月 7, 2025にアクセス、https://learn.microsoft.com/en-us/windows/wsl/install
9. WSL2 File System Management: Optimize Storage and Performance - Ceos3c, 6

月 7, 2025にアクセス、https://www.ceos3c.com/linux/wsl2-file-system-management-optimize-storage-and-performance/
10. Solving Windows 11 Windows Subsystem for Linux (WSL) Performance Optimization, 6月 7, 2025にアクセス、https://itfix.org.uk/solving-windows-11-windows-subsystem-for-linux-wsl-performance-optimization/
11. WSL のインストール | Microsoft Learn, 6月 7, 2025にアクセス、https://learn.microsoft.com/ja-jp/windows/wsl/install
12. Install Node.js on Windows Subsystem for Linux (WSL2) - Learn Microsoft, 6月 7, 2025にアクセス、https://learn.microsoft.com/en-us/windows/dev-environment/javascript/nodejs-on-wsl
13. How to Install Node.js on Ubuntu (Step-by-Step Guide) | DigitalOcean, 6月 7, 2025にアクセス、https://www.digitalocean.com/community/tutorials/how-to-install-node-js-on-ubuntu-22-04
14. How to set up Anthropic Claude + Billing - Team-GPT Knowledge Base, 6月 7, 2025にアクセス、https://help.team-gpt.com/how-to-set-up-anthropic-claude-billing
15. www.nightfall.ai, 6月 7, 2025にアクセス、https://www.nightfall.ai/ai-security-101/anthropic-claude-api-key#:~:text=biased%2C%20or%20unfair.-,How%20to%20Obtain%20an%20Anthropic%20Claude%20API%20Key,generate%20a%20new%20API%20key.
16. How to get your Anthropic API key (3 steps) - Merge.dev, 6月 7, 2025にアクセス、https://www.merge.dev/blog/anthropic-api-key
17. How do I pay for my API usage? | Anthropic Help Center, 6月 7, 2025にアクセス、https://support.anthropic.com/en/articles/8977456-how-do-i-pay-for-my-api-usage
18. Pricing \ Anthropic, 6月 7, 2025にアクセス、https://www.anthropic.com/pricing
19. Pricing - Anthropic API, 6月 7, 2025にアクセス、https://docs.anthropic.com/en/docs/about-claude/pricing
20. Remote development in WSL - Visual Studio Code, 6月 7, 2025にアクセス、https://code.visualstudio.com/docs/remote/wsl-tutorial
21. Developing in WSL - Visual Studio Code, 6月 7, 2025にアクセス、https://code.visualstudio.com/docs/remote/wsl
22. WSL - Visual Studio Marketplace, 6月 7, 2025にアクセス、https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-wsl
23. Claude Code Editor - Visual Studio Marketplace, 6月 7, 2025にアクセス、https://marketplace.visualstudio.com/items?itemName=AnandTyagi.claude-code-editor
24. CLI usage and controls - Anthropic API, 6月 7, 2025にアクセス、https://docs.anthropic.com/en/docs/claude-code/cli-usage
25. Claude Code: A Guide With Practical Examples - DataCamp, 6月 7, 2025にアクセ

ス、https://www.datacamp.com/tutorial/claude-code
26. Tutorials - Anthropic API, 6月 7, 2025にアクセス、https://docs.anthropic.com/en/docs/claude-code/tutorials
27. Anthropic's Guide to Claude Code: Best Practices for Agentic Coding : r/ClaudeAI - Reddit, 6月 7, 2025にアクセス、https://www.reddit.com/r/ClaudeAI/comments/1k5slll/anthropics_guide_to_claude_code_best_practices/
28. GitHub Actions - Anthropic API, 6月 7, 2025にアクセス、https://docs.anthropic.com/en/docs/claude-code/github-actions
29. Claude Code: Best practices for agentic coding - Anthropic, 6月 7, 2025にアクセス、https://www.anthropic.com/engineering/claude-code-best-practices
30. How to Use Claude Code with GitHub Actions - Apidog, 6月 7, 2025にアクセス、https://apidog.com/blog/claude-code-github-actions/
31. anthropics/claude-code-action - GitHub, 6月 7, 2025にアクセス、https://github.com/anthropics/claude-code-action