Red Hat build of Node.js 18

Node.js 18 のリリースノート

Node.js 18 LTS との使用

Last Updated: 2023-05-12

Red Hat build of Node.js 18 Node.js 18 のリリースノート

Node.js 18 LTS との使用

法律上の通知

Copyright © 2023 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

本リリースノートには、Node.js 18 LTS に関する重要な情報が含まれています。

目次

はじめに 3

第1章 必要なインフラストラクチャーコンポーネントのバージョン 4

第2章 機能 5
2.1. 新機能および変更された機能 5
2.2. 非推奨の機能 6
2.3. テクノロジープレビューの機能 7
2.4. サポートされているアーキテクチャー 9

第3章 リリースコンポーネント 10

第4章 修正された問題 11

第5章 既知の問題 12

第6章 必要なインフラストラクチャーコンポーネントに影響する既知の問題 13

第7章 本リリースに関連するアドバイザリー 14

はじめに

リリース日: 2023-01-31

第1章 必要なインフラストラクチャーコンポーネントのバージョン

Red Hat build of Node.js を使用する場合は、次のインフラストラクチャーコンポーネントが必要です。サポート対象として明示的に指定されているコンポーネントを除き、Red Hat はこれらのコンポーネントのサポートを提供していません。

コンポーネント名	Version
Nodeshift	2.1.1
npm 8	8.19.2
OpenShift Container Platform (OCP)[a]	3.11、4.5
git	2.0 以降
oc コマンドラインツール	3.11 以降[b]

[a] OCP は Red Hat によってサポートされます

[b] CLI ツール **oc** のバージョンは、使用している OCP のバージョンに対応する必要があります。

第2章 機能

このセクションには、Red Hat build of Node.js 18 リリースで導入された機能の変更に関する情報が含まれています。

2.1. 新機能および変更された機能

Node.js 18 LTS には、Red Hat build of Node.js がサポートする次の新機能と拡張機能があります。

Node.js 18 LTS の変更の詳細は、アップストリームのリリースノート および アップストリームのドキュメンテーション を参照してください。

2.1.1. v10.2 への V8 JavaScript エンジンのアップグレード

このリリースには、Chromium 101 の一部である v10.2 への V8 JavaScript エンジンのアップグレードが含まれています。

アップグレードされた V8 JavaScript エンジンには、以下の新機能および機能拡張が含まれています。

- **findLast()** および **findLastIndex()** 配列メソッド
- **Intl.Locale** API の改良
- **Intl.supportedValuesOf** 関数
- クラスフィールド と プライベートクラスメソッド のパフォーマンスが向上し、通常のプロパティーストアと同じ速度で初期化されるようになりました。

V8 JavaScript エンジンで利用可能な変更の詳細については、V8 ブログ を参照してください。

2.1.2. HTTP タイムアウトのデフォルト値

本リリースには、HTTP タイムアウトの以下の拡張機能が含まれます。

- パーサーが HTTP ヘッダー全体を受信するまで待機する時間を制限する **server.headersTimeout** プロパティーのデフォルト値が **60000** ミリ秒 (60 秒) になりました。
- サーバーがクライアントからの要求全体を受信するまで待機する時間を制限する **server.requestTimeout** プロパティーのデフォルト値が **300000** ミリ秒 (5 分) になりました。

これらのタイムアウトが期限切れになると、サーバーは **408** エラーで応答し、要求を要求リスナーに転送せずに接続を閉じます。

注記

リバースプロキシーがサーバーの前にデプロイされていない状況でサービス拒否攻撃を防止するには、これらのタイムアウト値をゼロ以外の値に設定してください。

2.1.3. グローバルスコープの Blob および BroadcastChannel API

次の API が完全にサポートされ、グローバルオブジェクトとして使用できるようになりました。

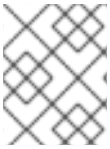
- **Blob** クラスは、複数のワーカースレッド間で安全に共有できる不変の生データをカプセル化します。**Blob** は、**Buffer** クラスのサブクラスです。

- **BroadcastChannel** クラスは、同じチャネル名にバインドされている他のすべての **BroadcastChannel** インスタンスとの非同期の 1 対多通信を有効にします。**BroadcastChannel** は **EventTarget** クラスを拡張します。

以前のリリースでは、これらの API はテクノロジープレビュー機能のみでした。

2.2. 非推奨の機能

以下の機能は、Red Hat build of Node.js 18 リリースで非推奨になりました。



注記

このリリースで廃止または削除された機能の詳細は、nodejs.org の Web サイト を参照してください。

2.2.1. ファイルの書き込みまたは追加のための fs メソッドでの文字列強制のランタイムの非推奨

このリリースには、次のいずれかのメソッドでパラメーターとして渡される独自の **toString** プロパティーがオブジェクトにある場合の暗黙的なオブジェクトから文字列への強制のランタイムの非推奨が含まれています。

- **fs.write()**
- **fs.writeFile()**
- **fs.appendFile()**
- **fs.writeFileSync()**
- **fs.appendFileSync()**

文字列強制の代わりに、オブジェクトをプリミティブ文字列に変換します。

2.2.2. dns.lookup および dnsPromises.lookup メソッドでのオプションタイプ強制

このリリースでは、**dns.lookup()** および **dnsPromises.lookup()** メソッドでパラメーターとして渡されるオプションタイプの強制が削除されます。次のオプションタイプと値の組み合わせのいずれかを使用すると、Node.js は **ERR_INVALID_ARG_TYPE** エラーを出力するようになりました。

- **family** オプションの null 以外の非整数値
- **hints** オプションの null 以外の非数値
- **all** オプションの null 以外の非ブール値
- **verbatim** オプションの null 以外の非ブール値

2.2.3. multipleResolves のランタイムの非推奨

このリリースには、**multipleResolves** プロセスイベントのランタイムの非推奨が含まれています。以下に例を示します。

```
process.on('multipleResolves', handler)
```

multipleResolves イベントは、V8 プロミスコンビネーターでは機能しませんでした。これは、**Promise** コンストラクターを使用する際の潜在的なエラーを追跡する際のこのイベントの有用性と信頼性に影響を与えました。たとえば、**Promise.race()** メソッドは、**multipleResolves** イベントをトリガーすることもできましたが、このイベントは、必ずしもエラーを示すわけではありませんでした。

2.2.4. thenable オブジェクトのサポート

このリリースでは、ストリーム実装メソッドで **thenable** オブジェクトを返す機能が削除されました。たとえば、ユーザーが関数をコールバックスタイルで実装し、**async** メソッドを使用した場合に **thenable** オブジェクトを使用すると、予期しない問題が発生する可能性があります。このタイプの実装では、プロミスとコールバックのセマンティクスが無効に混在する結果になりました。

以下に例を示します。

```
const w = new Writable({
  async final(callback) {
    await someOp();
    callback();
  },
});
```

thenable オブジェクトの代わりに、コールバックを使用し、ストリーム実装メソッドに **async** 関数を使用しないようにします。

2.2.5. tls.parseCertString()

このリリースでは、証明書のサブジェクトと発行者の文字列を解析するパーサーヘルパーであった **tls.parseCertString()** メソッドが削除されています。**tls.parseCertString()** メソッドは、複数値の相対識別名 (RDN) を正しく処理しなかったため、誤った表現やセキュリティーの問題が発生する可能性がありました。

さらに、**_tls_common.translatePeerCertificate** は、サブジェクトと発行者のプロパティーを変換しなくなりました。

注記

Node.js の以前のリリースでは、**tls.parseCertString()** の代わりに **querystring.parse()** メソッドを使用することが推奨されていました。ただし、**querystring.parse()** もすべての証明書サブジェクトを正しく処理できないため、このリリースでは代替手段として **querystring.parse()** を使用することは推奨されなくなりました。

2.3. テクノロジープレビューの機能

以下の機能は、Node.js 18 LTS リリースのテクノロジープレビュー機能として利用できます。

2.3.1. Fetch API

実験的な **fetch** API がグローバルオブジェクトとして利用できます。**fetch** API は、Node.js の HTTP/1.1 クライアントである Undici と、**Fetch** Web API の Node.js 実装を提供する **node-fetch** モジュールに基づいています。

以下に例を示します。

```
const res = await fetch('https://nodejs.org/api/documentation.json');
if (res.ok) {
  const data = await res.json();
  console.log(data);
}
```

この機能強化により、次のグローバルオブジェクトを使用できるようになりました。

- **fetch** (ブラウザー互換バージョンの **fetch()** 関数)
- **FormData** (**FormData** インターフェイスのブラウザー互換バージョン)
- **Headers** (**Headers** インターフェイスのブラウザー互換バージョン)
- **Request** (**Request** インターフェイスのブラウザー互換バージョン)
- **Response** (**Response** インターフェイスのブラウザー互換バージョン)

--no-experimental-fetch コマンドラインオプションを使用して、**fetch** API を無効にできます。

2.3.2. グローバルスコープの Web Streams API

Web Streams API は、Red Hat build of Node.js 16 リリースでテクノロジープレビュー機能として導入されました。このリリースでは、Node.js はグローバルスコープで Web Streams API を公開するようになりました。これは、**stream/web** コアモジュールを使用してのみ Web Streams API にアクセスできた以前のリリースの動作に優先されます。

この機能強化により、次の API をグローバルオブジェクトとして使用できるようになりました。

ReadableStream、**ReadableStreamDefaultReader**、**ReadableStreamBYOBReader**、**ReadableStreamBYOBRequest**、**ReadableByteStreamController**、**ReadableStreamDefaultController**、**TransformStream**、**TransformStreamDefaultController**、**WritableStream**、**WritableStreamDefaultWriter**、**WritableStreamDefaultController**、**ByteLengthQueuingStrategy**、**CountQueuingStrategy**、**TextEncoderStream**、**TextDecoderStream**、**CompressionStream**、**DecompressionStream**

2.3.3. ESM Loader Hooks API による複数のカスタムローダーのサポート

ESM ローダーフック API は、Red Hat build of Node.js 16 リリースでテクノロジープレビュー機能として導入されました。ESM Loader Hooks API は、複数のカスタムローダーをサポートするようになりました。

複数のローダーが連携できるようにするために、この機能は **チェーン** と呼ばれるプロセスを使用します。これはプロミスチェーンに似ています。 たとえば、**first-loader** は **second-loader** を呼び出し、**second-loader** は **third-loader** を呼び出します。

注記

カスタムローダーがチェーン内の次のローダーを意図的に呼び出さない場合、カスタムローダーは短絡を通知する必要があります。

詳細は、Node.js ECMAScript モジュールのドキュメント を参照してください。

2.3.4. ウォッチモード

node --watch オプションを使用して、ウォッチモードで Node.js アプリケーションを実行できるようになりました。

アプリケーションをウォッチモードで実行すると、インポートしたファイルを変更した場合は、プロセスが再起動されます。

2.4. サポートされているアーキテクチャー

Node.js ビルダーイメージおよび RPM パッケージが利用可能で、以下の CPU アーキテクチャーとの使用がサポートされます。

- AMD x86_64
- ARM64
- OpenShift 環境の IBM Z (s390x)
- OpenShift 環境の IBM Power System (ppc64le)

第3章 リリースコンポーネント

- Node.js 18 Builder Image for RHEL 8
- Node.js 18 Universal Base Image 8
- Node.js 18 Minimal Stand-alone Image for RHEL 8
- Node.js 18 Minimal Universal Base Image 8
- Node.js 18 Builder Image for RHEL 9
- Node.js 18 Universal Base Image 9
- Node.js 18 Minimal Stand-alone Image for RHEL 9
- Node.js 18 Minimal Universal Base Image 9

第4章 修正された問題

本リリースでは、Node.js 18 LTS のコミュニティーリリースのすべての修正された問題が含まれます。

第5章 既知の問題

本リリースに影響する既知の問題はありません。

第6章 必要なインフラストラクチャーコンポーネントに影響する既知の問題

本リリースで必要なインフラストラクチャーコンポーネントに影響を与える既知の問題はありません。

第7章 本リリースに関連するアドバイザリー

本リリースに含まれる拡張機能、バグ修正、および CVE 修正を文書化するために、以下のアドバイザリーが発行されています。

- RHSA-2022:8832
- RHSA-2022:8833
- RHBA-2022:7843
- RHBA-2022:8458