



Red Hat Enterprise Linux 8

RHEL 8 の cloud-init の設定と管理

Red Hat Enterprise Linux クラウドインスタンスの自動初期化

Red Hat Enterprise Linux 8 RHEL 8 の cloud-init の設定と管理

Red Hat Enterprise Linux クラウドインスタンスの自動初期化

法律上の通知

Copyright © 2024 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

cloud-init パッケージを使用すると、RHEL の複数のクラウドインスタンスを効率的に作成できます。これにより、さまざまなクラウドプラットフォームでの RHEL の一貫した繰り返しデプロイメントが可能になります。以下の章では、以下の情報を提供します。 cloud-init の仕組み cloud-init を使用してクラウドインスタンスを開始する方法 Red Hat がサポートする cloud-init の用途

目次

多様性を受け入れるオープンソースの強化	3
RED HAT ドキュメントへのフィードバック (英語のみ)	4
第1章 パブリッククラウドプラットフォームでの RHEL の導入	5
1.1. パブリッククラウドで RHEL を使用する利点	5
1.2. RHEL のパブリッククラウドのユースケース	6
1.3. パブリッククラウドへの移行時によくある懸念事項	7
1.4. パブリッククラウドデプロイメント用の RHEL の入手	8
1.5. RHEL クラウドインスタンスを作成する方法	8
第2章 CLOUD-INIT の概要	10
2.1. CLOUD-INIT 設定の概要	10
2.2. CLOUD-INIT はステージごとに動作する	11
2.3. CLOUD-INIT モジュールはフェーズごとに実行される	11
2.4. CLOUD-INIT は、ユーザーデータ、メタデータ、およびベンダーデータに対応する	12
2.5. CLOUD-INIT はクラウドプラットフォームを識別する	13
2.6. 関連情報	13
第3章 CLOUD-INIT の RED HAT サポート	14
3.1. CLOUD-INIT の重要なディレクトリーおよびファイル	14
3.2. CLOUD-INIT を使用する RED HAT 製品	15
3.3. RED HAT はこれらの CLOUD-INIT モジュールをサポートします。	15
3.4. デフォルトの CLOUD.CFG ファイル	18
3.5. CLOUD.CFG.D ディレクトリー	21
3.6. デフォルトの 05_LOGGING.CFG ファイル	21
3.7. CLOUD-INIT /VAR/LIB/CLOUD ディレクトリーのレイアウト	23
第4章 CLOUD-INIT の設定	25
4.1. NOCLOUD データソースの CLOUD-INIT を含む仮想マシンの作成	25
4.2. CLOUD-INIT を使用してクラウドユーザーパスワードを期限切れにする	27
4.3. CLOUD-INIT でのデフォルトユーザー名の変更	28
4.4. CLOUD-INIT を使用した ROOT パスワードの設定	28
4.5. CLOUD-INIT を使用した RED HAT サブスクリプションの管理	29
4.6. CLOUD-INIT を使用したユーザーおよびユーザーオプションの追加	30
4.7. CLOUD-INIT を使用した最初の起動コマンドの実行	31
4.8. CLOUD-INIT を使用した SUDOERS の追加	32
4.9. CLOUD-INIT を使用した静的ネットワーク設定	32
4.10. CLOUD-INIT を使用した ROOT ユーザーのみの設定	34
4.11. CLOUD-INIT で CONTAINER-STORAGE-SETUP を使用したストレージの設定	34
4.12. CLOUD-INIT を使用したシステムロケールの変更	35
4.13. CLOUD-INIT およびシェルスクリプト	35
4.14. CLOUD-INIT による設定ファイルの更新の阻止	36
4.15. CLOUD-INIT の実行後に KVM ゲストイメージから作成された仮想マシンの変更	36
4.16. CLOUD-INIT 実行後の特定データソースの仮想マシンの変更	37
4.17. CLOUD-INIT のトラブルシューティング	38

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、[Red Hat CTO である Chris Wright のメッセージ](#) を参照してください。

RED HAT ドキュメントへのフィードバック (英語のみ)

Red Hat ドキュメントに関するご意見やご感想をお寄せください。また、改善点があればお知らせください。

Jira からのフィードバック送信 (アカウントが必要)

1. [Jira](#) の Web サイトにログインします。
2. 上部のナビゲーションバーで **Create** をクリックします。
3. **Summary** フィールドにわかりやすいタイトルを入力します。
4. **Description** フィールドに、ドキュメントの改善に関するご意見を記入してください。ドキュメントの該当部分へのリンクも追加してください。
5. ダイアログの下部にある **Create** をクリックします。

第1章 パブリッククラウドプラットフォームでの RHEL の導入

パブリッククラウドプラットフォームは、コンピューティングリソースをサービスとして提供します。オンプレミスのハードウェアを使用する代わりに、Red Hat Enterprise Linux (RHEL) システムなどの IT ワークロードをパブリッククラウドインスタンスとして実行できます。

パブリッククラウドプラットフォーム上の RHEL の詳細は、以下を参照してください。

- [パブリッククラウドで RHEL を使用する利点](#)
- [RHEL のパブリッククラウドのユースケース](#)
- [パブリッククラウドへの移行時によくある懸念事項](#)
- [パブリッククラウドデプロイメント用の RHEL の入手](#)
- [RHEL クラウドインスタンスを作成する方法](#)

1.1. パブリッククラウドで RHEL を使用する利点

パブリッククラウドプラットフォーム上に配置されたクラウドインスタンスとしての RHEL には、RHEL オンプレミスの物理システムまたは仮想マシン (VM) に比べて次の利点があります。

- **リソースの柔軟性と詳細な割り当て**

RHEL のクラウドインスタンスは、クラウドプラットフォーム上の仮想マシンとして実行されます。この仮想マシンは通常、クラウドサービスのプロバイダーによって維持管理されるリモートサーバーのクラスターです。したがって、特定のタイプの CPU やストレージなどのハードウェアリソースのインスタンスへの割り当ては、ソフトウェアレベルで行われ、簡単にカスタマイズできます。

また、ローカルの RHEL システムと比較すると、物理ホストの機能によって制限されることがありません。むしろ、クラウドプロバイダーが提供する選択肢に基づいて、さまざまな機能から選択できます。

- **領域とコスト効率**

クラウドワークロードをホストするためにオンプレミスサーバーを所有する必要がありません。これにより、物理ハードウェアに関連するスペース、電力、メンテナンスの要件が回避されます。

代わりに、パブリッククラウドプラットフォームでは、クラウドインスタンスの使用料をクラウドプロバイダーに直接支払います。通常、コストはインスタンスに割り当てられたハードウェアとその使用時間に基づきます。したがって、要件に基づいてコストを最適化できます。

- **ソフトウェアで制御される設定**

クラウドインスタンスの設定全体がクラウドプラットフォーム上にデータとして保存され、ソフトウェアによって制御されます。したがって、インスタンスの作成、削除、クローン作成、または移行を簡単に行うことができます。また、クラウドインスタンスは、クラウドプロバイダーのコンソールでリモートで操作され、デフォルトでリモートストレージに接続されます。

さらに、クラウドインスタンスの現在の状態をいつでもスナップショットとしてバックアップできます。その後、スナップショットをロードしてインスタンスを保存した状態に復元できます。

- **ホストからの分離とソフトウェアの互換性**

ローカルの仮想マシンと同様に、クラウドインスタンス上の RHEL ゲストオペレーティングシステムは仮想化されたカーネル上で実行されます。このカーネルは、ホストオペレーティングシステムや、インスタンスへの接続に使用する **クライアント** システムとは別のものです。

したがって、任意のオペレーティングシステムをクラウドインスタンスにインストールできません。つまり、RHEL パブリッククラウドインスタンスでは、ローカルオペレーティングシステムでは使用できない RHEL 固有のアプリケーションを実行できます。

さらに、インスタンスのオペレーティングシステムが不安定になったり侵害されたりした場合でも、クライアントシステムには一切影響がありません。

関連情報

- [パブリッククラウドとは](#)
- [ハイパースケーラーとは](#)
- [クラウドコンピューティングの種類](#)
- [RHEL のパブリッククラウドのユースケース](#)
- [パブリッククラウドデプロイメント用の RHEL の入手](#)

1.2. RHEL のパブリッククラウドのユースケース

パブリッククラウドへのデプロイには多くの利点がありますが、すべてのシナリオにおいて最も効率的なソリューションであるとは限りません。RHEL デプロイメントをパブリッククラウドに移行するかどうかを評価している場合は、ユースケースがパブリッククラウドの利点を楽しむかどうかを検討してください。

有益なユースケース

- パブリッククラウドインスタンスのデプロイは、デプロイメントのアクティブなコンピューティング能力を柔軟に増減する (**スケールアップ** および **スケールダウン** と呼ばれます) 場合に非常に効果的です。したがって、次のシナリオではパブリッククラウドで RHEL を使用することを推奨します。
 - ピーク時のワークロードが高く、一般的なパフォーマンス要件が低いクラスター。要求に応じてスケールアップおよびスケールダウンすることで、リソースコストの面で高い効率を得られる場合があります。
 - クラスターを迅速にセットアップまたは拡張できます。これにより、ローカルサーバーのセットアップにかかる高額な初期費用が回避されます。
- クラウドインスタンスは、ローカル環境で何が起っても影響を受けません。したがって、バックアップや障害復旧に使用できます。

問題が発生する可能性のあるユースケース

- 調整不可能な既存の環境を運用している場合。既存のデプロイメントの特定のニーズに合わせてクラウドインスタンスをカスタマイズすることは、現在のホストプラットフォームと比較して費用対効果が低い可能性があります。
- 厳しい予算制限の中で運用している場合。通常、ローカルデータセンターでデプロイメントを維持管理すると、パブリッククラウドよりも柔軟性は低くなりますが、最大リソースコストをより細かく制御できます。

次のステップ

- [パブリッククラウドデプロイメント用の RHEL の入手](#)

関連情報

- [アプリケーションをクラウドに移行すべきか?また、その決定方法](#)

1.3. パブリッククラウドへの移行時によくある懸念事項

RHEL ワークロードをローカル環境からパブリッククラウドプラットフォームに移行すると、それに伴う変更について懸念が生じる可能性があります。よくある質問は次のとおりです。

RHEL は、クラウドインスタンスとして動作する場合、ローカル仮想マシンとして動作する場合とは異なる動作になりますか?

パブリッククラウドプラットフォーム上の RHEL インスタンスは、ほとんどの点で、オンプレミスサーバーなどのローカルホスト上の RHEL 仮想マシンと同じように機能します。注目すべき例外には次のようなものがあります。

- パブリッククラウドインスタンスは、プライベートオーケストレーションインターフェイスの代わりに、プロバイダー固有のコンソールインターフェイスを使用してクラウドリソースを管理します。
- ネストされた仮想化などの特定の機能が正しく動作しない可能性があります。特定の機能がデプロイメントにとって重要な場合は、選択したパブリッククラウドプロバイダーとその機能の互換性を事前に確認してください。

ローカルサーバーと比べて、パブリッククラウドではデータは安全に保たれますか?

RHEL クラウドインスタンス内のデータの所有権はユーザーにあり、パブリッククラウドプロバイダーはデータにアクセスできません。さらに、主要なクラウドプロバイダーは転送中のデータ暗号化をサポートしているため、仮想マシンをパブリッククラウドに移行する際のデータのセキュリティが向上します。

RHEL パブリッククラウドインスタンスの一般的なセキュリティは次のように管理されます。

- パブリッククラウドプロバイダーは、クラウドハイパーバイザーのセキュリティを担当します。
- Red Hat は、RHEL ゲストオペレーティングシステムのセキュリティ機能をインスタンスに提供します。
- ユーザーは、クラウドインフラストラクチャーにおける特定のセキュリティ設定とプラクティスを管理します。

ユーザーの地理的リージョンは、RHEL パブリッククラウドインスタンスの機能にどのように影響しますか?

RHEL インスタンスは、地理的な場所に関係なく、パブリッククラウドプラットフォームで使用できます。したがって、オンプレミスサーバーと同じリージョンでインスタンスを実行できます。

ただし、物理的に離れたリージョンでインスタンスをホストすると、操作時に待ち時間が長くなる可能性があります。さらに、パブリッククラウドプロバイダーによっては、特定のリージョンで、追加機能が提供される場合や、より高いコスト効率を得られる場合があります。RHEL インスタンスを作成する

前に、選択したクラウドプロバイダーで利用可能なホスティングリージョンのプロパティーを確認してください。

1.4. パブリッククラウドデプロイメント用の RHEL の入手

RHEL システムをパブリッククラウド環境にデプロイするには、次の手順を実行します。

1. 要件と市場の現在のオファーに基づいて、ユースケースに最適なクラウドプロバイダーを選択します。
現在、RHEL インスタンスの実行が認定されているクラウドプロバイダーは次のとおりです。
 - [Amazon Web Services \(AWS\)](#)
 - 詳細は、[Deploying RHEL 9 on Amazon Web Services](#) を参照してください。
 - [Google Cloud Platform \(GCP\)](#)
 - 詳細は、[Deploying RHEL 9 on Google Cloud Platform](#) を参照してください。
 - [Microsoft Azure](#)
 - 詳細は、[Deploying RHEL 9 on Microsoft Azure](#) を参照してください。
2. 選択したクラウドプラットフォーム上に RHEL クラウドインスタンスを作成します。詳細は、[RHEL クラウドインスタンスを作成する方法](#) を参照してください。
3. RHEL デプロイメントを最新の状態に保つには、[Red Hat Update Infrastructure \(RHUI\)](#) を使用します。

関連情報

- [RHUI ドキュメント](#)
- [Red Hat Open Hybrid Cloud](#)

1.5. RHEL クラウドインスタンスを作成する方法

RHEL インスタンスをパブリッククラウドプラットフォームにデプロイするには、次のいずれかの方法を使用できます。

RHEL のシステムイメージを作成し、クラウドプラットフォームにインポートします。

- システムイメージを作成するには、[RHEL Image Builder](#) を使用するか、イメージを手動で構築します。
- これは既存の RHEL サブスクリプションを使用する方法で、**bring your own subscription (BYOS)** と呼ばれます。
- 年間サブスクリプションを前払いすると、Red Hat お客様割引を利用できます。
- カスタマーサービスは Red Hat によって提供されます。
- 複数のイメージを効率的に作成するには、**cloud-init** ツールを使用できます。

RHEL インスタンスをクラウドプロバイダーマーケットプレイスから直接購入します。

- サービスの利用に対して時間料金を後払いで支払います。したがって、この方法は **従量課金制** (PAYG) とも呼ばれます。
- カスタマーサービスはクラウドプラットフォームプロバイダーによって提供されます。

関連情報

- [ゴールデンイメージとは](#)

第2章 CLOUD-INIT の概要

cloud-init ユーティリティーは、システムの起動時にクラウドインスタンスの初期化を自動化します。**cloud-init** は、さまざまなタスクを実行するように設定できます。

- ホスト名の設定
- インスタンスへのパッケージのインストール
- スクリプトの実行
- デフォルトの仮想マシン (VM) 動作の抑制

前提条件

- [Red Hat カスタマーポータル](#) のアカウントに登録している。

cloud-init は、さまざまなタイプの RHEL イメージで利用できます。以下に例を示します。

- [Red Hat カスタマーポータル](#) から KVM ゲストイメージをダウンロードすると、イメージが **cloud-init** パッケージで事前にインストールされます。インスタンスを起動すると、**cloud-init** パッケージが有効になります。Red Hat カスタマーポータルの KVM ゲストイメージは、Red Hat Virtualization (RHV)、Red Hat OpenStack Platform (RHOSP)、および Red Hat OpenShift Virtualization で使用することを目的としています。
- Red Hat カスタマーポータルから RHEL ISO イメージをダウンロードして、カスタムゲストイメージを作成することもできます。この場合は、カスタマイズしたゲストイメージに **cloud-init** パッケージをインストールする必要があります。
- クラウドサービスプロバイダー(AWS または Azure など)からのイメージを使用する必要がある場合は、**RHEL イメージビルダーを使用してイメージ** を作成します。Image Builder イメージは、特定のクラウドプロバイダー向けにカスタマイズされます。次のイメージタイプには、**cloud-init** がすでにインストールされているものがあります。
 - Amazon Machine Image (AMI)
 - 仮想ハードディスク(VHD)
 - QEMU copy-on-write (qcow2)
RHEL Image Builder の詳細は [RHEL システムイメージのカスタマイズ](#) を参照してください。

ほとんどのクラウドプラットフォームは **cloud-init** をサポートしますが、設定手順とサポートされるオプションは異なります。また、NoCloud 環境向けに **cloud-init** を設定できます。

さらに、1つの仮想マシンで **cloud-init** を設定し、その仮想マシンをテンプレートとして使用し、追加の仮想マシンまたは仮想マシンクラスターを作成できます。

[Red Hat Virtualization](#) などの特定の Red Hat 製品では、これらの製品の **cloud-init** を設定する手順が文書化されています。

2.1. CLOUD-INIT 設定の概要

cloud-init ユーティリティーは、YAML 形式の設定ファイルを使用してユーザー定義のタスクをインスタンスに適用します。インスタンスが起動すると、**cloud-init** サービスが起動して、YAML ファイルからの指示を実行します。設定によっては、タスクは最初の起動時または仮想マシンの後続の起動時に完

了します。

特定のタスクを定義するには、`/etc/cloud/cloud.cfg` ファイルを設定し、`/etc/cloud/cloud.cfg.d/` ディレクトリーの下にディレクティブを追加します。

- **cloud.cfg** ファイルには、ユーザーアクセス、認証、システム情報など、さまざまなシステム設定のディレクティブが含まれます。
ファイルには、**cloud-init** のデフォルトおよびオプションのモジュールも含まれています。これらのモジュールは、のフェーズで順番に実行されます。**cloud-init** 初期化フェーズ ..設定フェーズ ..最終フェーズ。

+ **cloud.cfg** ファイルでは、3つのフェーズのモジュールが **cloud_init_modules**、**cloud_config_modules**、および **cloud_final_modules** の下にそれぞれ一覧表示されます。

- **cloud.cfg.d** ディレクトリーに、**cloud-init** のディレクティブを追加できます。**cloud.cfg.d** ディレクトリーにディレクティブを追加する場合は、それらを ***.cfg** という名前のカスタムファイルに追加し、ファイルの先頭に常に **#cloud-config** を含める必要があります。

2.2. CLOUD-INIT はステージごとに動作する

システムの起動時に、**cloud-init** ユーティリティーは5つの段階で動作し、その他のタスクの中で **cloud-init** が実行されるかどうか、データソースを見つける場所を決定します。ステージは次のとおりです。

1. **ジェネレーターステージ:** **systemd** サービスを使用することで、このフェーズは、起動時に **cloud-init** ユーティリティーを実行するかどうかを決定します。
2. **ローカルステージ:** **cloud-init** はローカルデータソースを検索し、DHCP ベースのフォールバックメカニズムを含むネットワーク設定を適用します。
3. **ネットワークステージ:** **cloud-init** は、`/etc/cloud/cloud.cfg` ファイルの **cloud_init_modules** の下にリストされているモジュールを実行してユーザーデータを処理します。**cloud_init_modules** セクションで、モジュールを追加、削除、有効化、または無効化できます。
4. **Config stage:** **cloud-init** は、`/etc/cloud/cloud.cfg` ファイルの **cloud_config_modules** セクションに一覧表示されているモジュールを実行します。**cloud_config_modules** セクションでモジュールを追加、削除、有効化、または無効にできます。
5. **最終ステージ:** **cloud-init** は、`/etc/cloud/cloud.cfg` ファイルの **cloud_final_modules** セクションに含まれるモジュールおよび設定を実行します。これには、特定のパッケージのインストールや、設定管理プラグインおよびユーザー定義のスクリプトのトリガーが含まれます。**cloud_final_modules** セクションでモジュールを追加、削除、有効化、または無効にできます。

関連情報

- [cloud-init のブートステージ](#)

2.3. CLOUD-INIT モジュールはフェーズごとに実行される

cloud-init を実行すると、**cloud.cfg** 内のモジュールが次の3つのフェーズで順番に実行されます。

1. ネットワークフェーズ (**cloud_init_modules**)

2. 設定フェーズ (`cloud_config_modules`)3. 最終フェーズ (`cloud_final_modules`)

仮想マシンで `cloud-init` が初めて実行されると、設定したすべてのモジュールがそれぞれのフェーズで実行されます。`cloud-init` の後続の実行では、モジュールがフェーズ内で実行されるかどうかは、個々のモジュールの **モジュール頻度** により異なります。`cloud-init` が実行されるたびに実行されるモジュールもあれば、インスタンス ID が変更された場合でも、`cloud-init` の初回実行時にしか実行されないモジュールもあります。



注記

インスタンス ID はインスタンスを一意に識別します。インスタンス ID が変更されると、`cloud-init` はそのインスタンスを新しいインスタンスとして処理します。

可能な **モジュール周波数** の値は次のとおりです。

- **Per instance** とは、モジュールがインスタンスの初回起動時に実行されることを意味します。たとえば、インスタンスのクローンを作成したり、保存したイメージから新しいインスタンスを作成したりすると、インスタンス別と指定されたモジュールは再度実行されます。
- **Per once** とは、モジュールが1回だけ実行されることを意味します。たとえば、インスタンスのクローンを作成したり、保存したイメージから新しいインスタンスを作成したりすると、1回と指定されたモジュールは、それらのインスタンスでは再度実行されません。
- **Per always** とは、モジュールが起動ごとに実行されることを意味します。



注記

モジュールの設定時またはコマンドラインを使用して、モジュールの頻度を上書きできます。

2.4. CLOUD-INIT は、ユーザーデータ、メタデータ、およびベンダーデータに対応する

`cloud-init` は、ユーザーデータ、メタデータ、ベンダーデータを消費し、これらに対応します。

- ユーザーデータには、`cloud.cfg` ファイルと `cloud.cfg.d` ディレクトリーで指定するディレクトリが含まれます。たとえば、ユーザーデータには、実行するファイル、インストールするパッケージ、およびシェルスクリプトを含むことができます。`cloud-init` が許可するユーザーデータのタイプに関する詳細は、`cloud-init` ドキュメントの [User-Data Formats](#) セクションを参照してください。
- メタデータには、特定のデータソースに関連付けられたデータが含まれます。たとえば、メタデータにはサーバー名とインスタンス ID を含むことができます。特定のクラウドプラットフォームを使用している場合、インスタンスがユーザーデータとメタデータを見つける場所をプラットフォームが決定します。プラットフォームでは、メタデータとユーザーデータの HTTP サービスへの追加が必要な場合があります。この場合、`cloud-init` を実行すると、HTTP サービスからメタデータとユーザーデータが消費されます。
- ベンダーデータは、組織(クラウドプロバイダーなど)がオプションで提供し、イメージが実行される環境に合わせてイメージをカスタマイズできる情報が含まれます。`cloud-init` は、メタデータを読み込んでシステムを初期化した後に、オプションのベンダーデータおよびユーザーデータに対応します。デフォルトでは、ベンダーデータは初回起動時に実行されます。ベンダーデータの実行を無効にすることができます。

メタデータの説明は **cloud-init** ドキュメントの [Instance Metadata](#) セクション、データソースのリストは [Datasources](#)、ベンダーデータの詳細は [Vendor Data](#) を参照してください。

2.5. CLOUD-INIT はクラウドプラットフォームを識別する

cloud-init は、**ds-identify** スクリプトを使用してクラウドプラットフォームの特定を試みます。スクリプトは、インスタンスの初回起動時に実行されます。

データソースディレクティブを追加すると、**cloud-init** の実行時に時間を節約できます。ディレクティブは、`/etc/cloud/cloud.cfg` ファイルまたは `/etc/cloud/cloud.cfg.d` ディレクトリーに追加します。以下に例を示します。

```
datasource_list:[Ec2]
```

クラウドプラットフォームのディレクティブを追加すること以外に、メタデータ URL などの追加の設定詳細を追加して **cloud-init** をさらに設定できます。

```
datasource_list: [Ec2]
datasource:
  Ec2:
    metadata_urls: ['http://169.254.169.254']
```

cloud-init の実行後に、プラットフォームに関する詳細情報を提供するログファイル (`run/cloud-init/ds-identify.log`) を表示できます。

関連情報

- [Datasources](#)
- [使用しているデータソースを特定する方法](#)
- [How can I debug my user data?](#)

2.6. 関連情報

- [cloud-init のアップストリームドキュメント](#)

第3章 CLOUD-INIT の RED HAT サポート

本章では、**cloud-init** の Red Hat サポートについて説明します。これには、**cloud-init** を使用する Red Hat 製品、Red Hat がサポートする **cloud-init** モジュール、デフォルトのディレクトリーおよびファイルに関する情報が含まれます。

3.1. CLOUD-INIT の重要なディレクトリーおよびファイル

以下の表には、重要なディレクトリーおよびファイルが記載されています。これらのディレクトリーおよびファイルを確認します。これらにより、以下のようなタスクを実行することができます。

- **cloud-init** の設定
- **cloud-init** 実行後の設定に関する情報の検索
- ログファイルの検証
- テンプレートの検索

シナリオおよびデータソースに応じて、お使いの設定にとって重要な追加のファイルとディレクトリーが存在する場合があります。

表3.1 cloud-init ディレクトリーおよびファイル

ディレクトリーまたはファイル	説明
<code>/etc/cloud/cloud.cfg</code>	cloud.cfg ファイルには基本的な cloud-init 設定が含まれ、各モジュールが実行するフェーズを確認できます。
<code>/etc/cloud/cloud.cfg.d</code>	cloud.cfg.d ディレクトリーでは、 cloud-init の追加ディレクティブを追加できます。
<code>/var/lib/cloud</code>	cloud-init を実行すると、 <code>/var/lib/cloud</code> の下にディレクトリーレイアウトが作成されます。このレイアウトには、インスタンス設定の詳細情報を提供するディレクトリーとファイルが含まれます。
<code>/usr/share/doc/cloud-init/examples</code>	examples ディレクトリーには、複数の例が含まれています。これらを使用すると、独自のディレクティブを作成できます。
<code>/etc/cloud/templates</code>	このディレクトリーには、特定のシナリオの cloud-init で有効にできるテンプレートが含まれています。テンプレートは、有効にするための指示を提供します。
<code>/var/log/cloud-init.log</code>	cloud-init.log ファイルは、デバッグに役立つログ情報を提供します。

ディレクトリまたはファイル	説明
/run/cloud-init	/run/cloud-init ディレクトリには、データソースおよび ds-identify スクリプトのログ記録情報が含まれます。

3.2. CLOUD-INIT を使用する RED HAT 製品

以下の Red Hat 製品で **cloud-init** を使用できます。

- **Red Hat Virtualization.cloud-init** を仮想マシンにインストールすると、テンプレートを作成し、そのテンプレートから作成したすべての仮想マシンに **cloud-init** 機能を利用できます。仮想マシンでの **cloud-init** の使用に関する詳細は、[Cloud-Init を使用した仮想マシンの設定の自動化](#) を参照してください。
- **Red Hat OpenStack Platform.cloud-init** を使用して、OpenStack のイメージの設定をサポートできます。詳細は、[インスタンスおよびイメージガイド](#) を参照してください。
- **Red Hat Satellite.cloud-init** を Red Hat Satellite で使用することができます。詳細は、[Preparing Cloud-init Images in Red Hat Virtualization](#) を参照してください。
- **Red Hat OpenShift.OpenShift** 用の仮想マシンを作成する際に **cloud-init** を使用できます。詳細は、[Creating virtual machines](#) を参照してください。

3.3. RED HAT はこれらの CLOUD-INIT モジュールをサポートします。

Red Hat は、ほとんどの **cloud-init** モジュールをサポートしています。個々のモジュールには、複数の設定オプションを含めることができます。以下の表は、Red Hat が現在サポートしているすべての **cloud-init** モジュールのリストで、簡単な説明とデフォルトのモジュール頻度を記載しています。これらのモジュールの完全な説明およびオプションは、cloud-init ドキュメントの [Modules](#) セクションを参照してください。

表3.2 サポートされている cloud-init モジュール

cloud-init モジュール	説明	デフォルトのモジュール頻度
bootcmd	ブートプロセスの初期段階でコマンドを実行します。	常時
ca_certs	CA 証明書を追加します。	インスタンス別
debug	デバッグを支援するために内部情報の出力を有効または無効にします。	インスタンス別
disable_ec2_metadata	AWS EC2 メタデータを有効または無効にします。	常時
disk_setup	シンプルなパーティションテーブルとファイルシステムを設定します。	インスタンス別

cloud-init モジュール	説明	デフォルトのモジュール頻度
final_message	cloud-init の完了後に出力メッセージを指定します。	常時
foo	モジュール構造の例 (モジュールは何もしません)。	インスタンス別
growpart	パーティションのサイズを変更して、利用可能なディスク領域を埋めます。	常時
keys_to_console	コンソールに書き込むことができるフィンガープリントとキーの制御を許可します。	インスタンス別
landscape	ランドスケープクライアントをインストールおよび設定します。	インスタンス別
locale	システムロケールを設定し、システム全体に適用します。	インスタンス別
mcollective	mcollective をインストール、設定、および起動します。	インスタンス別
migrator	cloud-init の古いバージョンを新しいバージョンへに移動します。	常時
mounts	マウントポイントとスワップファイルを設定します。	インスタンス別
phone_home	起動完了後にリモートホストにデータを投稿します。	インスタンス別
power_state_change	すべての設定モジュールの実行後にシャットダウンを完了し、再起動します。	インスタンス別
puppet	puppet をインストールおよび設定します。	インスタンス別
resizefs	ファイルシステムのサイズを変更し、パーティションで利用可能な領域をすべて使用します。	常時
resolv_conf	resolv.conf を設定します。	インスタンス別
rh_subscription	Red Hat Enterprise Linux システムを登録します。	インスタンス別

cloud-init モジュール	説明	デフォルトのモジュール頻度
rightscale_userdata	cloud-init に RightScale 設定フックのサポートを追加します。	インスタンス別
rsyslog	rsyslog を使用してリモートシステムロギングを設定します。	インスタンス別
runcmd	任意のコマンドを実行します。	インスタンス別
salt_minion	salt minion をインストール、設定、および開始します。	インスタンス別
scripts_per_boot	起動スクリプトごとに実行します。	常時
scripts_per_instance	インスタンススクリプトごとに実行します。	インスタンス別
scripts_per_once	スクリプトを1回実行します。	1回
scripts_user	ユーザースクリプトを実行します。	インスタンス別
scripts_vendor	ベンダースクリプトを実行します。	インスタンス別
seed_random	ランダムなシードデータを提供します。	インスタンス別
set_hostname	ホスト名および完全修飾ドメイン名 (FQDN) を設定します。	常時
set_passwords	ユーザーパスワードを設定し、SSH パスワード認証を有効または無効にします。	インスタンス別
ssh_authkey_fingerprints	ユーザーの SSH 鍵のフィンガープリントをログに記録します。	インスタンス別
ssh_import_id	SSH 鍵をインポートします。	インスタンス別
ssh	SSH、ホスト、および認可された SSH 鍵を設定します。	インスタンス別
timezone	システムのタイムゾーンを設定します。	インスタンス別
update_etc_hosts	/etc/hosts を更新します。	常時

cloud-init モジュール	説明	デフォルトのモジュール頻度
update_hostname	ホスト名および FQDN を更新します。	常時
users_groups	ユーザーおよびグループを設定します。	インスタンス別
write_files	任意のファイルを書き込みます。	インスタンス別
yum_add_repo	yum リポジトリ設定をシステムに追加します。	常時

以下の表は、現在 Red Hat がサポートしていないモジュールをリスト表示しています。

表3.3 モジュールはサポートされません。

モジュール
apt_configure
apt_pipeline
byobu
chef
emit_upstart
grub_dpkg
ubuntu_init_switch

3.4. デフォルトの CLOUD.CFG ファイル

`/etc/cloud/cloud.cfg` ファイルは、**cloud-init** の基本設定を設定するモジュールをリスト表示します。

ファイルのモジュールは、**cloud-init** のデフォルトのモジュールです。お使いの環境にモジュールを設定したり、不要なモジュールを削除したりすることができます。**cloud.cfg** に含まれるモジュールは、ファイルにリスト表示されているだけで、必ずしもなんでも実行する訳ではありません。**cloud-init** フェーズのいずれかでアクションを実行する必要がある場合は、それらを個別に設定する必要があります。

cloud.cfg ファイルは、個別のモジュールを実行するための時系列を提供します。Red Hat が追加するモジュールをサポートする限り、追加のモジュールを **cloud.cfg** に追加できます。

Red Hat Enterprise Linux (RHEL) のファイルのデフォルトコンテンツは、以下のとおりです。



注記

- モジュールは、**cloud.cfg** で指定された順序で実行されます。通常、この順序は変更しません。
- **cloud.cfg** ディレクティブは、ユーザーデータによって上書きされることができません。
- **cloud-init** を手動で実行する場合は、コマンドラインオプションで **cloud.cfg** を上書きできます。
- 各モジュールには独自の設定オプションが含まれており、この設定オプションに特定の情報を追加することができます。

users: **1**
- default

disable_root: 1 **2**

ssh_pwauth: 0 **3**

mount_default_fields: [~, ~, 'auto', 'defaults,nofail,x-systemd.requires=cloud-init.service', '0', '2'] **4**

ssh_deletekeys: 1 **5**

ssh_genkeytypes: ['rsa', 'ecdsa', 'ed25519'] **6**

syslog_fix_perms: ~ **7**

disable_vmware_customization: false **8**

cloud_init_modules: **9**

- disk_setup
- migrator
- bootcmd
- write-files
- growpart
- resizefs
- set_hostname
- update_hostname
- update_etc_hosts
- rsyslog
- users-groups
- ssh

cloud_config_modules: **10**

- mounts
- locale
- set-passwords
- rh_subscription
- yum-add-repo
- package-update-upgrade-install
- timezone
- puppet
- chef
- salt-minion
- mcollective
- disable-ec2-metadata
- runcmd

```

cloud_final_modules: 11
- rightscale_userdata
- scripts-per-once
- scripts-per-boot
- scripts-per-instance
- scripts-user
- ssh-authkey-fingerprints
- keys-to-console
- phone-home
- final-message
- power-state-change

system_info:
  default_user: 12
    name: cloud-user
    lock_passwd: true
    gecos: Cloud User
    groups: [adm, systemd-journal]
    sudo: ["ALL=(ALL) NOPASSWD:ALL"]
    shell: /bin/bash
  distro: rhel 13
  paths:
    cloud_dir: /var/lib/cloud 14
    templates_dir: /etc/cloud/templates 15
  ssh_svcname: sshd 16

# vim:syntax=yaml

```

- 1 システムのデフォルトユーザーを指定します。詳細は、[Users and Groups](#) を参照してください。
- 2 root ログインを有効または無効にします。詳細は、[Authorized Keys](#) を参照してください。
- 3 **ssh** がパスワード認証を受け入れるよう設定されているかどうかを指定します。詳細は、[Set Passwords](#) を参照してください。
- 4 マウントポイントを設定します。6つの値を含むリストでなければなりません。詳細は、[Mounts](#) を参照してください。
- 5 デフォルトのホスト SSH 鍵を削除するかどうかを指定します。詳細は、[Host Keys](#) を参照してください。
- 6 生成する鍵のタイプを指定します。詳細は、[Host Keys](#) を参照してください。RHEL 8.4 以前の場合、この行のデフォルト値は ~ であることに注意してください。
- 7 **cloud-init** は、起動時に複数のステージで実行されます。**cloud-init** が、すべてのステージをログファイルにログ記録できるように、このオプションを設定します。このオプションの詳細は、[usr/share/doc/cloud-init/examples](#) ディレクトリーの **cloud-config.txt** ファイルを参照してください。
- 8 VMware vSphere のカスタマイズを有効または無効にします。
- 9 本セクションのモジュールは、起動プロセスの初期段階における **cloud-init** サービスの起動時に実行されるサービスです。
- 10 これらのモジュールは、初回起動後の **cloud-init** 設定時に実行されます。

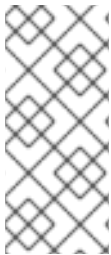
- 11 これらのモジュールは、設定完了後に **cloud-init** の最終フェーズで実行されます。
- 12 デフォルトユーザーの詳細を指定します。詳細は、[Users and Groups](#) を参照してください。
- 13 ディストリビューションを指定します。
- 14 **cloud-init** 固有のサブディレクトリーが含まれるメインディレクトリーを指定します。詳細は、[Directory layout](#) を参照してください。
- 15 テンプレートの場所を指定します。
- 16 SSH サービスの名前

関連情報

- [モジュール](#)

3.5. CLOUD.CFG.D ディレクトリー

cloud-init は、ユーザーが提供および設定するディレクティブに対応します。通常、これらのディレクティブは **cloud.cfg.d** ディレクトリーに含まれています。



注記

cloud.cfg ファイル内でユーザーデータディレクティブを追加することでモジュールを設定できますが、**cloud.cfg** を未変更のままにすること (ベストプラクティス) をご検討ください。ディレクティブを **/etc/cloud/cloud.cfg.d** ディレクトリーに追加します。このディレクトリーにディレクティブを追加することで、今後の変更およびアップグレードを容易にすることができます。

ディレクティブを追加する方法は複数あります。ディレクティブは、見出し **#cloud-config** を含む ***.cfg** という名前のファイルに追加できます。通常、ディレクトリーには複数の ***.cfg** ファイルが含まれます。ディレクティブを追加するオプションは他にもあります。たとえば、ユーザーデータスクリプトを追加できます。詳細は、[User-Data Formats](#) を参照してください。

関連情報

- [Cloud config examples](#)

3.6. デフォルトの 05_LOGGING.CFG ファイル

05_logging.cfg ファイルは、**cloud-init** のログ情報を設定します。**/etc/cloud/cloud.cfg.d** ディレクトリーには、追加する他の **cloud-init** ディレクティブと共にこのファイルが含まれます。

cloud-init は、デフォルトで **05_logging.cfg** のロギング設定を使用します。Red Hat Enterprise Linux (RHEL) のファイルのデフォルトコンテンツは、以下のとおりです。

```
## This yaml formatted config file handles setting
## logger information. The values that are necessary to be set
## are seen at the bottom. The top '_log' are only used to remove
## redundancy in a syslog and fallback-to-file case.
##
## The 'log_cfgs' entry defines a list of logger configs
```

```
## Each entry in the list is tried, and the first one that
## works is used. If a log_cfg list entry is an array, it will
## be joined with '\n'.
_log:
- &log_base |
  [loggers]
  keys=root,cloudinit

  [handlers]
  keys=consoleHandler,cloudLogHandler

  [formatters]
  keys=simpleFormatter,arg0Formatter

  [logger_root]
  level=DEBUG
  handlers=consoleHandler,cloudLogHandler

  [logger_cloudinit]
  level=DEBUG
  qualname=cloudinit
  handlers=
  propagate=1

  [handler_consoleHandler]
  class=StreamHandler
  level=WARNING
  formatter=arg0Formatter
  args=(sys.stderr,)

  [formatter_arg0Formatter]
  format=%(asctime)s - %(filename)s[%(levelname)s]: %(message)s

  [formatter_simpleFormatter]
  format=[CLOUDINIT] %(filename)s[%(levelname)s]: %(message)s
- &log_file |
  [handler_cloudLogHandler]
  class=FileHandler
  level=DEBUG
  formatter=arg0Formatter
  args=('/var/log/cloud-init.log',)
- &log_syslog |
  [handler_cloudLogHandler]
  class=handlers.SysLogHandler
  level=DEBUG
  formatter=simpleFormatter
  args=("/dev/log", handlers.SysLogHandler.LOG_USER)

log_cfgs:
# Array entries in this list will be joined into a string
# that defines the configuration.
#
# If you want logs to go to syslog, uncomment the following line.
# - [ *log_base, *log_syslog ]
#
# The default behavior is to just log to a file.
```

```
# This mechanism that does not depend on a system service to operate.
- [ *log_base, *log_file ]
# A file path can also be used.
# - /etc/log.conf

# This tells cloud-init to redirect its stdout and stderr to
# 'tee -a /var/log/cloud-init-output.log' so the user can see output
# there without needing to look on the console.
output: {all: '| tee -a /var/log/cloud-init-output.log'}
```

関連情報

- [Logging](#)

3.7. CLOUD-INIT /VAR/LIB/CLOUD ディレクトリーのレイアウト

cloud-init を最初に実行すると、インスタンスおよび **cloud-init** 設定に関する情報が含まれるディレクトリーレイアウトが作成されます。

ディレクトリーには、**/scripts/vendor** などのオプションのディレクトリーを追加できます。

以下は、**cloud-init** のサンプルディレクトリーレイアウトです。

```
/var/lib/cloud/
- data/
  - instance-id
  - previous-instance-id
  - previous-datasource
  - previous-hostname
  - result.json
  - set-hostname
  - status.json
- handlers/
- instance
  - boot-finished
  - cloud-config.txt
  - datasource
  - handlers/
  - obj.pkl
  - scripts/
  - sem/
  - user-data.txt
  - user-data.txt.i
  - vendor-data.txt
  - vendor-data.txt.i
- instances/
  f111ee00-0a4a-4eea-9c17-3fa164739c55/
    - boot-finished
    - cloud-config.txt
    - datasource
    - handlers/
    - obj.pkl
    - scripts/
    - sem/
    - user-data.txt
```

- user-data.txt.i
- vendor-data.txt
- vendor-data.txt.i
- scripts/
 - per-boot/
 - per-instance/
 - per-once/
 - vendor/
- seed/
- sem/
 - config_scripts_per_once.once

関連情報

- [Directory layout](#)

第4章 CLOUD-INIT の設定

本章では、**cloud-init** で最も一般的な設定タスクの例を紹介します。

cloud-init 設定では、**cloud.cfg** ファイルおよび **cloud.cfg.d** ディレクトリーへのディレクティブの追加を必要とすることがあります。あるいは、特定のデータソースでは、ユーザーデータファイルやメタデータファイルなどのファイルにディレクティブを追加する必要がある場合があります。データソースでは、ディレクティブの HTTP サーバーへのアップロードが必要な場合があります。データソースの要件を確認し、それに応じてディレクティブを追加します。

4.1. NOCLOUD データソースの CLOUD-INIT を含む仮想マシンの作成

cloud-init を含む新しい仮想マシン (VM) を作成するには、次の手順を参照してください。この手順では、**meta-data** ファイルと **user-data** ファイルを作成します。

- **meta-data** ファイルには、インスタンスの詳細が含まれます。
- **user-data** ファイルには、ユーザーを作成し、アクセスを付与するための情報が含まれます。

これらのファイルを新しい ISO イメージに追加し、KVM ゲストイメージから作成された新しい仮想マシンに ISO ファイルをアタッチします。このシナリオでは、データソースは NoCloud です。

手順

1. **cloudinitiso** という名前のディレクトリーを作成し、作業ディレクトリーとして設定します。

```
$ mkdir cloudinitiso
$ cd cloudinitiso
```

2. **meta-data** ファイルを作成して、以下の情報を追加します。

```
instance-id: citest
local-hostname: citest-1
```

3. **user-data** ファイルを作成して、以下の情報を追加します。

```
#cloud-config
password: cilogon
chpasswd: {expire: False}
ssh_pwauth: True
ssh_authorized_keys:
  - ssh-rsa AAA...fhHQ== sample@redhat.com
```



注記

user-data ファイルの最後の行は、SSH 公開鍵を参照します。~/**.ssh/id_rsa.pub** で SSH 公開鍵を検索します。このサンプル手順を行う場合は、行を変更して公開鍵の1つを含めます。

4. **genisoimage** コマンドを使用して、**user-data** および **meta-data** を含む ISO イメージを作成します。

```
# genisoimage -output ciiso.iso -volid cidata -joliet -rock user-data meta-data
```

```
l: -input-charset not specified, using utf-8 (detected in locale settings)
Total translation table size: 0
Total rockridge attributes bytes: 331
Total directory bytes: 0
Path table size(bytes): 10
Max brk space used 0
183 extents written (0 MB)
```

- Red Hat カスタマーポータルから、`/var/lib/libvirt/images` ディレクトリーに KVM ゲストイメージをダウンロードします。
- virt-install** ユーティリティーを使用して KVM ゲストイメージから新しい仮想マシンを作成し、ダウンロードしたイメージを既存のイメージにアタッチします。

```
# virt-install \
  --memory 4096 \
  --vcpus 4 \
  --name mytestcvm \
  --disk /var/lib/libvirt/images/rhel-8.1-x86_64-
kvm.qcow2,device=disk,bus=virtio,format=qcow2 \
  --disk /home/sample/cloudinitiso/ciiso.iso,device=cdrom \
  --os-type Linux \
  --os-variant rhel8.0 \
  --virt-type kvm \
  --graphics none \
  --import
```

- ユーザー名 **cloud-user** およびパスワード **cilogon** でイメージにログオンします。

```
citest-1 login: cloud-user
Password:
[cloud-user@citest-1 ~]$
```

検証

- cloud-init** ステータスを確認して、ユーティリティーが定義されたタスクを完了していることを確認します。

```
[cloud-user@citest-1 instance]$ cloud-init status
status: done
```

- cloud-init** ユーティリティーは、実行時に `/var/lib/cloud` の下に **cloud-init** ディレクトリーレイアウトを作成し、指定したディレクティブに基づいて特定のディレクトリーコンテンツを更新または変更します。
たとえば、データソースファイルをチェックして、データソースが **NoCloud** であることを確認できます。

```
$ cd /var/lib/cloud/instance
$ cat datasource
DataSourceNoCloud: DataSourceNoCloud [seed=/dev/sr0][dsmode=net]
```

- cloud-init** copies user-data into `/var/lib/cloud/instance/user-data.txt`

```
$ cat user-data.txt
#cloud-config
password: cilogon
chpasswd: {expire: False}
ssh_pwauth: True
ssh_authorized_keys:
- ssh-rsa AAA...fhHQ== sample@redhat.com
```



注記

OpenStack の場合、[インスタンスの作成と管理](#) には、**cloud-init** を使用してインスタンスを設定するための情報が含まれています。特定の手順は、Creating a customized instance を参照してください。

関連情報

- [Upstream documentation for the NoCloud data source](#)

4.2. CLOUD-INIT を使用してクラウドユーザーパスワードを期限切れにする

初回ログイン時に **cloud-user** パスワードを変更するよう **cloud-user** に強制することができます。パスワードを失効させるには、以下の手順を実行します。

手順

1. データソースの要件に応じて、**user-data** ファイルを編集するか、以下のディレクティブを **cloud.cfg.d** ディレクトリーに追加します。



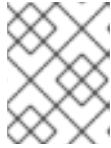
注記

cloud-init が、ユーザーディレクティブを含むファイルを認識できるように、すべてのユーザーディレクティブはファイルの最上部に **#cloud-config** が含まれます。**cloud.cfg.d** ディレクトリーにディレクティブを含める場合は、ファイル名を ***.cfg** とし、ファイルの最上部に常に **#cloud-config** を含めます。

2. **chpasswd: {expire: False}** の行を **chpasswd: {expire: True}** に変更します。

```
#cloud-config
password: mypassword
chpasswd: {expire: True}
ssh_pwauth: True
ssh_authorized_keys:
- ssh-rsa AAA...SDvz user1@yourdomain.com
- ssh-rsa AAB...QTuo user2@yourdomain.com
```

これはパスワードを失効させます。なぜなら、**password** と **chpasswd** は特に指定がない限り、デフォルトのユーザーで動作するからです。



注記

これはグローバル設定です。**chpasswd** を **True** に設定すると、作成するすべてのユーザーが、ログイン時にパスワードを変更する必要があります。

4.3. CLOUD-INIT でのデフォルトユーザー名の変更

デフォルトのユーザー名は、**cloud-user** 以外のものに変更できます。

手順

1. データソースの要件に応じて、**user-data** ファイルを編集するか、以下のディレクティブを **cloud.cfg.d** ディレクトリーに追加します。



注記

cloud-init が、ユーザーディレクティブを含むファイルを認識できるように、すべてのユーザーディレクティブはファイルの最上部に **#cloud-config** が含まれます。**cloud.cfg.d** ディレクトリーにディレクティブを含める場合は、ファイル名を ***.cfg** とし、ファイルの最上部に常に **#cloud-config** を含めます。

2. **user: <username>** の行を追加します。<username> は新しいデフォルトのユーザー名に置き換えます。

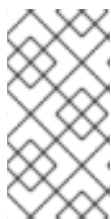
```
#cloud-config
user: username
password: mypassword
chpasswd: {expire: False}
ssh_pwauth: True
ssh_authorized_keys:
- ssh-rsa AAA...SDvz user1@yourdomain.com
- ssh-rsa AAB...QTuo user2@yourdomain.com
```

4.4. CLOUD-INIT を使用した ROOT パスワードの設定

root パスワードを設定するには、ユーザーのリストを作成します。

手順

1. データソースの要件に応じて、**user-data** ファイルを編集するか、以下のディレクティブを **cloud.cfg.d** ディレクトリーに追加します。



注記

cloud-init が、ユーザーディレクティブを含むファイルを認識できるように、すべてのユーザーディレクティブはファイルの最上部に **#cloud-config** が含まれます。**cloud.cfg.d** ディレクトリーにディレクティブを含める場合は、ファイル名を ***.cfg** とし、ファイルの最上部に常に **#cloud-config** を含めます。

2. ファイルの **chpasswd** セクションで、ユーザー一覧を作成します。



注記

空白は重要です。ユーザーリストのコロンの前後に空白を含めないでください。空白が含まれている場合、パスワードは空白を入れた設定となります。

```
#cloud-config
ssh_pwauth: True
ssh_authorized_keys:
  - ssh-rsa AAA...SDvz user1@yourdomain.com
  - ssh-rsa AAB...QTuo user2@yourdomain.com
chpasswd:
  list: |
    root:myrootpassword
    cloud-user:mypassword
  expire: False
```



注記

この方法を使用してユーザーパスワードを設定する場合は、本セクションのすべてのパスワードを設定する必要があります。

4.5. CLOUD-INIT を使用した RED HAT サブスクリプションの管理

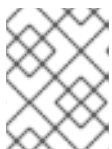
rh_subscription ディレクティブを使用してシステムを登録できます。サブスクリプションごとに、ユーザーデータを編集する必要があります。以下の手順では、**rh_subscription** ディレクティブを使用して加えることのできる変更の例をいくつか示します。

手順

- **自動アタッチ オプション**と **サービスレベル オプション**を使用するには、次の手順を実行します。

rh_subscription の下に **username** と **password** を追加して、**auto-attach** を **True** に設定し、**service-level** を **self-support** に設定します。

```
rh_subscription:
  username: sample@redhat.com
  password: 'mypassword'
  auto-attach: True
  service-level: self-support
```



注記

service-level オプションでは、**auto-attach** オプションを使用する必要があります。

- **activation-key** および **org** オプションを使用するには、以下の手順を実行します。**rh_subscription** の下に **activation key** と **org** の番号を追加し、**auto-attach** を **True** に設定します。

```
rh_subscription:
  activation-key: example_key
  org: 12345
```

```
auto-attach: True
```

- サブスクリプションプールを追加するには、以下の手順を実行します。
rh_subscription の下に、**username**、**password**、およびプール番号を追加します。

```
rh_subscription:
  username: sample@redhat.com
  password: 'password'
  add-pool: XYZ01234567
```



注記

このサンプルは、**subscription-manager attach --pool=XYZ01234567** コマンドに相当します。

- /etc/rhsm/rhsm.conf** ファイルでサーバーのホスト名を設定するには、以下の手順を実行します。
rh_subscription の下に **username**、**password**、**server-hostname** を追加し、**auto-attach** を **True** に設定します。

```
rh_subscription:
  username: sample@redhat.com
  password: 'password'
  server-hostname: test.example.com
  auto-attach: True
```

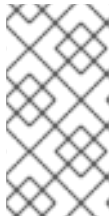
4.6. CLOUD-INIT を使用したユーザーおよびユーザーオプションの追加

users セクションでユーザーを作成し、説明します。セクションを変更して初期システム設定にユーザーをさらに追加でき、追加のユーザーオプションを設定できます。

users セクションを追加する場合、本セクションのデフォルトのユーザーオプションを設定する必要もあります。

手順

- データソースの要件に応じて、**user-data** ファイルを編集するか、以下のディレクティブを **cloud.cfg.d** ディレクトリーに追加します。



注記

cloud-init が、ユーザーディレクティブを含むファイルを認識できるように、すべてのユーザーディレクティブはファイルの最上部に **#cloud-config** が含まれます。**cloud.cfg.d** ディレクトリーにディレクティブを含める場合は、ファイル名を ***.cfg** とし、ファイルの最上部に常に **#cloud-config** を含めます。

- users** セクションを追加または変更し、ユーザーを追加します。
 - cloud-user** を、指定する他のユーザーと共に作成したデフォルトユーザーにするには、セクションの最初のエントリーとして **default** を追加することを確認してください。これが最初のエントリーでない場合は、**cloud-user** は作成されません。

- デフォルトでは、**selinux-user** 値がない場合、ユーザーは **unconfined_u** とラベル付けされます。

```
#cloud-config
users:
  - default
  - name: user2
    gecos: User N. Ame
    selinux-user: staff_u
    groups: users,wheel
    ssh_pwauth: True
    ssh_authorized_keys:
      - ssh-rsa AA..vz user@domain.com
chpasswd:
  list: |
    root:password
    cloud-user:mypassword
    user2:mypassword2
  expire: False
```



注記

- この例では、ユーザー **user2** を2つのグループ (**users** と **wheel**) に配置します。

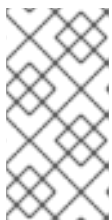
4.7. CLOUD-INIT を使用した最初の起動コマンドの実行

runcmd セクションおよび **bootcmd** セクションを使用して、起動および初期化中にコマンドを実行できます。

bootcmd セクションは、初期化プロセスの早い段階で実行され、デフォルトでは起動ごとに実行されます。**runcmd** セクションは、プロセスの最後の方で実行され、初回起動時および初期化中にのみ実行されます。

手順

1. データソースの要件に応じて、**user-data** ファイルを編集するか、以下のディレクティブを **cloud.cfg.d** ディレクトリーに追加します。



注記

cloud-init が、ユーザーディレクティブを含むファイルを認識できるように、すべてのユーザーディレクティブはファイルの最上部に **#cloud-config** が含まれます。**cloud.cfg.d** ディレクトリーにディレクティブを含める場合は、ファイル名を ***.cfg** とし、ファイルの最上部に常に **#cloud-config** を含めます。

2. **bootcmd** および **runcmd** のセクションを追加します。**cloud-init** が実行するコマンドを含めません。

```
#cloud-config
users:
  - default
  - name: user2
    gecos: User N. Ame
```

```

groups: users
chpasswd:
  list: |
    root:password
    fedora:myfedpassword
    user2:mypassword2
  expire: False
bootcmd:
  - echo New MOTD >> /etc/motd
runcmd:
  - echo New MOTD2 >> /etc/motd

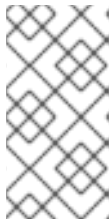
```

4.8. CLOUD-INIT を使用した SUDOERS の追加

sudo および **groups** エントリーを **users** セクションに追加することで、ユーザーを sudoer として設定できます。

手順

1. データソースの要件に応じて、**user-data** ファイルを編集するか、以下のディレクティブを **cloud.cfg.d** ディレクトリーに追加します。



注記

cloud-init が、ユーザーディレクティブを含むファイルを認識できるように、すべてのユーザーディレクティブはファイルの最上部に **#cloud-config** が含まれます。**cloud.cfg.d** ディレクトリーにディレクティブを含める場合は、ファイル名を ***.cfg** とし、ファイルの最上部に常に **#cloud-config** を含めます。

2. **sudo** エントリーを追加し、ユーザーアクセスを指定します。たとえば、**sudo: ALL=(ALL) NOPASSWD:ALL** は、ユーザーに無制限のユーザーアクセスを許可します。
3. **groups** エントリーを追加し、ユーザーを含むグループを指定します。

```

#cloud-config
users:
  - default
  - name: user2
    gecos: User D. Two
    sudo: ["ALL=(ALL) NOPASSWD:ALL"]
    groups: wheel,adm,systemd-journal
    ssh_pwauth: True
    ssh_authorized_keys:
      - ssh-rsa AA...vz user@domain.com
chpasswd:
  list: |
    root:password
    cloud-user:mypassword
    user2:mypassword2
  expire: False

```

4.9. CLOUD-INIT を使用した静的ネットワーク設定

メタデータに **network-interfaces** セクションを追加することで、**cloud-init** を使用したネットワーク設定を設定できます。

Red Hat Enterprise Linux は、動的ネットワークを制御および設定するデーモンである **NetworkManager** を介してデフォルトのネットワークサービスを提供します。これにより、ネットワークデバイスと接続が利用可能な場合に起動してアクティブな状態を維持します。

データソースがネットワーク設定を提供する場合があります。詳細は、**cloud-init** の [Network Configuration Sources](#) セクションを参照してください。

cloud-init のネットワーク設定を指定せずに、ネットワーク設定を無効にしていない場合、**cloud-init** は割り当てられているデバイスに接続があるかどうかを判別しようとします。接続されたデバイスを見つければ、インターフェイスで DHCP 要求を発行するネットワーク設定が生成されます。詳細は、**cloud-init** ドキュメントの [Fallback Network Configuration](#) セクションを参照してください。

手順

以下の例では、静的ネットワーク設定を追加します。

1. データソースの要件に応じて、**user-data** ファイルを編集するか、以下のディレクティブを **cloud.cfg.d** ディレクトリーに追加します。



注記

cloud-init が、ユーザーディレクティブを含むファイルを認識できるように、すべてのユーザーディレクティブはファイルの最上部に **#cloud-config** が含まれます。**cloud.cfg.d** ディレクトリーにディレクティブを含める場合は、ファイル名を ***.cfg** とし、ファイルの最上部に常に **#cloud-config** を含めます。

2. **network-interfaces** セクションを追加します。

```
network:
  version: 1
  config:
    - type: physical
      name: eth0
      subnets:
        - type: static
          address: 192.0.2.1/24
          gateway: 192.0.2.254
```



注記

メタデータに以下の情報を追加することで、ネットワーク設定を無効にすることができます。

```
network:
  config: disabled
```

関連情報

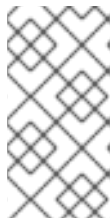
- [Network Configuration](#)
- [NoCloud](#)

4.10. CLOUD-INIT を使用した ROOT ユーザーのみの設定

root ユーザーがあり、他のユーザーはないようにユーザーデータを設定することができます。

手順

1. データソースの要件に応じて、**user-data** ファイルを編集するか、以下のディレクティブを **cloud.cfg.d** ディレクトリーに追加します。



注記

cloud-init が、ユーザーディレクティブを含むファイルを認識できるように、すべてのユーザーディレクティブはファイルの最上部に **#cloud-config** が含まれます。**cloud.cfg.d** ディレクトリーにディレクティブを含める場合は、ファイル名を ***.cfg** とし、ファイルの最上部に常に **#cloud-config** を含めます。

2. **users** セクションに、ユーザー **root** のエントリーを作成します。
以下の簡単な例には、**name** オプションのみを持つ **users** セクションが含まれます。

```
users:
  - name: root
chpasswd:
  list: |
    root:password
  expire: False
```

3. オプションで、root ユーザーの SSH 鍵を設定します。

```
users:
  - name: root
    ssh_pwauth: True
    ssh_authorized_keys:
      - ssh-rsa AA..vz user@domain.com
```

4.11. CLOUD-INIT で CONTAINER-STORAGE-SETUP を使用したストレージの設定

write_files モジュール内で、**container-storage-setup** ユーティリティーを参照してストレージを設定できます。

手順

1. データソースの要件に応じて、**user-data** ファイルを編集するか、以下のディレクティブを **cloud.cfg.d** ディレクトリーに追加します。



注記

cloud-init が、ユーザーディレクティブを含むファイルを認識できるように、すべてのユーザーディレクティブはファイルの最上部に **#cloud-config** が含まれます。**cloud.cfg.d** ディレクトリーにディレクティブを含める場合は、ファイル名を ***.cfg** とし、ファイルの最上部に常に **#cloud-config** を含めます。

2. **write_files** モジュールを追加または変更して、**container-storage-setup** ユーティリティーへのパスを追加します。
以下の例では、**root** 論理ボリュームのサイズを、デフォルトの 3 GB ではなく 6GB に設定します。

```
write_files:
- path: /etc/sysconfig/docker-storage-setup
  permissions: 0644
  owner: root
  content: |
    ROOT_SIZE=6G
```



注記

RHEL 7.4 より前のバージョンでは、**container-storage-setup** は **docker-storage-setup** と呼ばれていました。RHEL 7.4 の時点で、ストレージに OverlayFS を使用している場合は、SELinux でのこのタイプのファイルシステムを Enforcing モードで使用できるようになりました。

4.12. CLOUD-INIT を使用したシステムロケールの変更

locale モジュールを使用して、システムの場所を設定できます。

手順

1. データソースの要件に応じて、**meta-data** ファイルを編集します。以下のディレクティブを **cloud.cfg** ファイルまたは **cloud.cfg.d** ディレクトリーに追加することもできます。
2. 場所を指定して **locale** ディレクティブを追加します。以下の例では、**UTF-8** エンコーディングで **locale** を **ja_JP** (日本) に設定しています。

```
#cloud-config
locale: ja_JP.UTF-8
```

関連情報

- [システムロケールの設定](#)

4.13. CLOUD-INIT およびシェルスクリプト

bootcmd または **runcmd** に、リスト値または文字列の値を追加できます。また、ユーザーデータ内にシェルスクリプトを指定することもできます。

- **bootcmd** または **runcmd** のリスト値を使用すると、各リスト項目は **execve** を使用して順番に実行されます。
- 文字列の値を使用する場合、文字列全体がシェルスクリプトとして実行されます。
- **cloud-init** を使用してシェルスクリプトを実行する場合は、**cloud-init** に **.yaml** ファイルを指定する代わりに、(シバン (#!)) 機能を備えたシェルスクリプトを指定できます。

シェルスクリプトを **bootcmd** および **runcmd** に配置する方法の例は、[Run commands on first boot](#) を参照してください。

4.14. CLOUD-INIT による設定ファイルの更新の阻止

バックアップイメージからインスタンスを作成または復元すると、インスタンス ID が変更されます。インスタンス ID の変更により、**cloud-init** ユーティリティーは設定ファイルを更新します。

バックアップから作成または復元する際に、**cloud-init** が特定の設定ファイルを更新しないようにするには、以下の手順を実行します。

手順

1. `/etc/cloud/cloud.cfg` ファイルを編集します。次に例を示します。

```
# vi /etc/cloud/cloud.cfg
```

2. インスタンスの復元時に、**cloud-init** が更新しない設定をコメントアウトまたは削除します。たとえば、SSH 鍵ファイルの更新を回避するには、**cloud_init_modules** セクションから **-ssh** を削除します。

```
cloud_init_modules:
- disk_setup
- migrator
- bootcmd
- write-files
- growpart
- resizefs
- set_hostname
- update_hostname
- update_etc_hosts
- rsyslog
- users-groups
# - ssh
```

検証

- **cloud-init** によって更新された設定ファイルを確認するには、`/var/log/cloud/cloud-init.log` ファイルを調べます。更新されたファイルは、インスタンスの起動時に **Writing to** で始まるメッセージでログに記録されます。以下に例を示します。

```
2019-09-03 00:16:07,XXX - util.py[DEBUG]: Writing to /root/.ssh/authorized_keys - wb: [XXX]
554 bytes
2019-09-03 00:16:08,XXX - util.py[DEBUG]: Writing to /etc/ssh/sshd_config - wb: [XXX] 3905
bytes
```

4.15. CLOUD-INIT の実行後に KVM ゲストイメージから作成された仮想マシンの変更

cloud-init ユーティリティーを再実行する前に、**cloud-init** 設定を変更するには、次の手順を使用します。**cloud-init** パッケージがインストールされ、有効にして仮想マシンを起動すると、**cloud-init** は仮想マシンの初回起動時にデフォルト状態で実行されます。

手順

1. 仮想マシンにログインします。

2. ディレクティブを追加または変更します。たとえば、`/etc/cloud` ディレクトリーの `cloud.cfg` ファイルを変更するか、ディレクティブを `/etc/cloud/cloud.cfg.d` ディレクトリーに追加します。
3. `cloud-init clean` コマンドを実行してディレクトリーをクリーンアップし、`cloud-init` が再実行できるようにします。root で以下のコマンドを実行して、仮想マシンをクリーンアップすることもできます。

```
rm -Rf /var/lib/cloud/instances/
rm -Rf /var/lib/cloud/instance
rm -Rf /var/lib/cloud/data/
```



注記

クリーニングしたイメージを新しいイメージとして保存し、そのイメージを複数の仮想マシンに使用できます。新しい仮想マシンは、更新された `cloud-init` 設定を使用して `cloud-init` を実行します。

4. `cloud-init` を再実行するか、仮想マシンを再起動します。`cloud-init` が再実行し、変更した設定を実装します。

4.16. CLOUD-INIT 実行後の特定データソースの仮想マシンの変更

`cloud-init` を再実行する前に、`cloud-init` 設定を変更するには、次の手順を参照してください。この手順では、例として OpenStack を使用します。実行する必要がある正確な手順は、データソースによって異なることに注意してください。

手順

1. OpenStack Platform のインスタンスを作成して起動します。OpenStack のインスタンスの作成は、[インスタンスの作成](#) を参照してください。この例では、仮想マシン (VM) には `cloud-init` が含まれており、これは仮想マシンの起動時に実行されます。
2. ディレクティブを追加または変更します。たとえば、OpenStack HTTP サーバー上に保管されている `user-data.file` ファイルを変更します。
3. 仮想マシンをクリーンアップします。以下のコマンドを root 権限で実行します。

```
# rm -rf /etc/resolv.conf /run/cloud-init
# userdel -rf cloud-user
# hostnamectl set-hostname localhost.localdomain
# rm /etc/NetworkManager/conf.d/99-cloud-init.conf
```



注記

クリーニングしたイメージを新しいイメージとして保存し、そのイメージを複数の仮想マシンに使用できます。新しい仮想マシンは、更新された `cloud-init` 設定を使用して、`cloud-init` を実行します。

4. `cloud-init` を再実行するか、仮想マシンを再起動します。`cloud-init` が再実行し、設定変更を実装します。

4.17. CLOUD-INIT のトラブルシューティング

cloud-init ユーティリティーの実行後に、設定ファイルとログファイルを確認してインスタンスをトラブルシューティングできます。問題を特定したら、インスタンスで **cloud-init** を再実行します。コマンドラインから **cloud-init** を実行できます。詳細は、**cloud-init --help** コマンドを実行してください。

以下の手順では、**cloud-init** の問題を特定する手順と、プログラムを再実行するためのサンプルを説明します。

手順

1. **cloud-init** 設定ファイルを確認します。
 - a. `/etc/cloud/cloud.cfg` 設定ファイルを検証します。**cloud_init_modules**、**cloud_config_modules**、および **cloud_final_modules** に含まれるモジュールを確認します。
 - b. `/etc/cloud/cloud.cfg.d` ディレクトリーで、ディレクティブ (`*.cfg` files) を確認します。
2. 特定の問題の詳細については、`/var/log/cloud-init.log` ファイルおよび `/var/log/cloud-init-output.log` ファイルを確認してください。たとえば、`root` パーティションが自動的に拡張されなかった場合は、**growpart** ユーティリティーのログメッセージを確認します。ファイルシステムが拡張されなかった場合は、ログメッセージで **resizefs** を確認します。以下に例を示します。

```
# grep resizefs /var/log/cloud-init.log
```



注記

growpart は LVM をサポートしません。`root` パーティションが LVM をベースとしている場合は、`root` パーティションは初回起動時に自動的に拡張されません。

3. `root` で **cloud-init** コマンドを再実行します。
 - a. `init` モジュールのみで **cloud-init** を再実行します。


```
# /usr/bin/cloud-init -d init
```
 - b. 設定内のすべてのモジュールで **cloud-init** を再実行します。


```
# /usr/bin/cloud-init -d modules
```
 - c. **cloud-init** キャッシュを削除し、起動後に **cloud-init** を強制的に実行します。


```
# rm -rf /var/lib/cloud/ && /usr/bin/cloud-init -d init
```
 - d. ディレクトリーをクリーンなし、クリーンなインスタンスをシミュレートします。

```
# rm -rf /var/lib/cloud/instances/
# rm -rf /var/lib/cloud/instance
# rm -rf /var/lib/cloud/data/
# reboot
```

e. **cloud-init** ユーティリティーを再実行します。

```
# cloud-init init --local  
# cloud-init init
```

関連情報

- [cloud-init cli コマンド](#)