



# PROXMOX VE 管理ガイド

リリース8.2.2



2024年4月25

由 Proxmox Server Solutions

GmbH

[www.proxmox.com](http://www.proxmox.com)

著作権 © 2024 Proxmox Server Solutions GmbH

GNUフリー・ドキュメンテーション・ライセンス、バージョン1.3、またはフリーソフトウェアファウンデーションによって発行されたそれ以降のバージョンの条件の下で、この文書を複製、頒布、改変することを許可します。

ライセンスのコピーは、「GNU Free Documentation License」というセクションに含まれています。

# 内容

<b>1 はじめに</b>	<b>1</b>
1.1 中央管理	2
1.2 フレキシブル・ストレージ	3
1.3 統合バックアップとリストア	3
1.4 高可用性クラスタ	3
1.5 柔軟なネットワーキング	4
1.6 統合ファイアウォール	4
1.7 ハイパーコンバージドインフラ	4
1.7.1 Proxmox VEによるハイパーコンバージドインフラ（HCI）のメリット	4
1.7.2 ハイパーコンバージドインフラストラクチャストレージ	5
1.8 なぜオープンソースなのか	5
1.9 プロックスモックスVEのメリット	5
1.10 ヘルプを得る	6
1.10.1 プロックスモックス VE ウィキ	6
1.10.2 地域支援フォーラム	6
1.10.3 メーリングリスト	6
1.10.4 商業サポート	6
1.10.5 バグトラッカー	6
1.11 プロジェクトの歴史	6
1.12 Proxmox VE ドキュメントの改善	7
1.13 Proxmox VEの翻訳	7

1.13.1 gitを使った翻訳 .....	8
1.13.2 gitを使わない翻訳 .....	8
1.13.3 翻訳のテスト .....	8
1.13.4 翻訳の送信 .....	9

<b>2 Proxmox VEのインストール</b>	<b>10</b>
2.1 システム要件 .....	10
2.1.1 評価のための最低条件 .....	10
2.1.2 推奨動作環境 .....	11
2.1.3 シンプルなパフォーマンスの概要 .....	11
2.1.4 ウェブ・インターフェイスにアクセスするためにサポートされているウェブ・ブラウザ .....	11
2.2 インストールメディアの準備 .....	12
2.2.1 インストールメディアとしてUSBフラッシュドライブを準備する .....	12
2.2.2 GNU/Linux用説明書 .....	12
2.2.3 macOS用説明書 .....	13
2.2.4 Windows用説明書 .....	14
2.3 Proxmox VEインストーラの使用方法 .....	14
2.3.1 インストール後の管理インターフェイスへのアクセス .....	22
2.3.2 高度なLVM構成オプション .....	23
2.3.3 高度なZFS設定オプション .....	24
2.3.4 ZFSパフォーマンスのヒント .....	25
2.3.5 nomodesetカーネル・パラメータの追加 .....	25
2.4 無人設置 .....	25
2.5 DebianにProxmox VEをインストールする .....	25
<b>3 ホストシステム管理</b>	<b>26</b>
3.1 パッケージ・リポジトリ .....	26
3.1.1 Proxmox VEのリポジトリ .....	26
3.1.2 Proxmox VEエンタープライズリポジトリ .....	28
3.1.3 Proxmox VE ノーサブスクリプションリポジトリ .....	28
3.1.4 Proxmox VE テストリポジトリ .....	28
3.1.5 Ceph Reefエンタープライズリポジトリ .....	29
3.1.6 Ceph Reefのサブスクリプションなしリポジトリ .....	29

---

3.1.7 Ceph Reef テストリポジトリ .....	29
3.1.8 Ceph Quincy Enterprise リポジトリ .....	29
3.1.9 Ceph Quincy サブスクリプションなしリポジトリ .....	30
3.1.10 Ceph Quincy テストリポジトリ .....	30
3.1.11 古い Ceph リポジトリ .....	30
3.1.12 Debian ファームウェア リポジトリ .....	30
3.1.13 セキュア アパート .....	31
3.2 システムソフトウェアのアップデート .....	31

3.3	Firmware Updates .....	31
3.3.1	永続ファームウェア .....	32
3.3.2	ランタイムファームウェアファイル .....	33
3.3.	3CPU マイクロコードの更新 .....	33
3.4	ネットワーク構成	35
3.4.1	ネットワークの変更を 適用する	35
3.4.2	命名規則 .....	36
3.4.3	ネットワーク構成の 選択 .....	38
3.4.	4Default Configuration using a Bridge .....	39
3.4.	5Routed Configuration .....	40
3.4.6	iptablesによる マスカレード (NAT) .....	41
3.4.	7Linux Bond .....	42
3.4.	8vlan 802.1q .....	44
3.4.	9 ノードでIPv6を無効にする .....	46
3.4.10	Disabling MAC Learning on a Bridge .....	47
3.5	時間同期	47
3.5.	1 カスタム NTP サーバーの使用 .....	47
3.6	外部メトリック・サーバー	49
3.6.	1Graphite server configuration .....	49
3.6.	2Influxdb plugin configuration .....	50
3.7	Disk Health Monitoring .....	50
3.8	論理ボリュームマネージャ (LVM) .....	51
3.8.1	ハードウェア .....	52
3.8.2	ブートローダー .....	52
3.8.3	ボリュームグループの 作成 .....	52
3.8.4	/var/lib/vz用の追加LVの 作成 .....	53
3.8.5	薄いプールの サイズ変更 .....	53
3.8.	6Create a LVM-thin pool .....	53
3.9	Linux上のZFS .....	54
3.9.1	ハードウェア .....	54
3.9.2	ルートファイルシステムとしての インストール .....	55

---

3.9. 3ZFS RAID Level Considerations .....	56
3.9. 4ZFS dRAID .....	57
3.9.5 ブートローダー .....	58
3.9. 6ZFS Administration .....	58

3.9.7 電子メール通知の設定 .....	62
3.9.8 ZFSメモリ使用量の制限 .....	62
3.9.9 ZFSのSWAP .....	63
3.9.10 暗号化されたZFSデータセット .....	64
3.9.11 ZFSの圧縮 .....	65
3.9.12 ZFS専用デバイス .....	66
3.9.13 ZFSプールの特徴 .....	67
3.10 BTRFS .....	68
3.10.1 ルートファイルシステムとしてのインストール .....	68
3.10.2 BTRFSアドミニストレーション .....	69
3.11 Proxmoxノード管理 .....	71
3.11.1 ウェイクオンLAN .....	71
3.11.2 タスク履歴 .....	72
3.11.3 バルクゲスト電源管理 .....	72
3.11.4 ファースト・ゲスト・ブート・ディレイ .....	73
3.11.5 ゲストの一括移行 .....	73
3.12 証明書管理 .....	73
3.12.1 クラスタ内通信証明書 .....	73
3.12.2 APIおよびWeb GUIの証明書 .....	73
3.12.3 カスタム証明書のアップロード .....	74
3.12.4 Let's Encrypt (ACME) による信頼できる証明書 .....	74
3.12.5 ACME HTTPチャレンジプラグイン .....	76
3.12.6 ACME DNS API チャレンジプラグイン .....	77
3.12.7 ACME証明書の自動更新 .....	78
3.12.8 pvenodeを使用したACMEの例 .....	78
3.13 ホスト・ブートローダー .....	81
3.13.1 インストーラーが使用するパーティション方式 .....	81
3.13.2 proxmox-boot-toolでESPの内容を同期させる .....	82

3.13.3 どのブートローダを使用するかを決定する .....	84
3.13.4 GRUB.....	85
3.13.5 システムブート .....	86
3.13.6 カーネル・コマンドラインの編集 .....	86
3.13.7 次のブートのためにカーネルバージョンを上書きする .....	87
3.13.8 セキュアブート .....	88
3.14 カーネル・サムページ・マージング (KSM) .....	90
3.14.1 KSMの意味.....	90
3.14.2 KSMを無効にする.....	91
<b>4 グラフィカル・ユーザー・インターフェース</b>	<b>92</b>
4.1 特徴 .....	92
4.2 ログイン .....	93
4.3 GUIの概要 .....	93
4.3.1 ヘッダー .....	94
4.3.2 マイ・セッティング .....	95
4.3.3 リソースツリー .....	95
4.3.4 ログパネル .....	96
4.4 コンテンツパネル .....	96
4.4.1 データセンター .....	97
4.4.2 ノード .....	98
4.4.3 ゲスト .....	99
4.4.4 ストレージ .....	101
4.4.5 プール .....	102
4.5 タグ .....	103
4.5.1 スタイル構成 .....	104
4.5.2 アクセス許可 .....	104
<b>5 クラスター・マネージャー</b>	<b>106</b>

5.1 必要条件 .....	106
5.2 ノードの準備 .....	107
5.3 クラスタの作成 .....	107
5.3.1 ウェブGUIで作成 .....	108
5.3.2 コマンドラインによる作成 .....	108
5.3.3 同一ネットワーク内の複数のクラスタ .....	108
5.4 クラスタへのノードの追加 .....	109
5.4.1 GUIでノードをクラスタに参加させる .....	109
5.4.2 コマンドラインからクラスタにノードを参加させる .....	110
5.4.3 クラスタネットワークを分離してノードを追加する .....	111
5.5 クラスタノードの削除 .....	111
5.5.1 再インストールせずにノードを分離する .....	113
5.6 定足数 .....	114
5.7 クラスターネットワーク .....	115
5.7.1 ネットワーク要件 .....	115
5.7.2 クラスタ分離ネットワーク .....	115
5.7.3 コロシンク住所 .....	118
5.8 コロシンクの冗長性 .....	119
5.8.1 既存のクラスタに冗長リンクを追加する .....	120
5.9 Proxmox VEクラスタにおけるSSHの役割 .....	121
5.9.1 SSHセットアップ .....	121
5.9.2 .bashrcとその兄弟の自動実行による落とし穴 .....	122
5.10 コロシンク外部投票サポート .....	122
5.10.1 QDevice技術概要 .....	122
5.10.2 対応セットアップ .....	123
5.10.3 QDevice-Netのセットアップ .....	123
5.10.4 よくある質問 .....	124

5.11 コロシンクの構成 .....	125
5.11.1 corosync.confを編集する .....	125
5.11.2 トラブルシューティング .....	126
5.11.3 Corosync設定用語集 .....	126
5.12 クラスタークールドスタート .....	127
5.13 ゲストVMID自動選択 .....	127
5.14 ゲスト移住 .....	127
5.14.1 マイグレーション・タイプ .....	127
5.14.2 マイグレーション・ネットワーク .....	128
<b>6 Proxmox クラスタファイルシステム (pmxcfs)</b>	<b>130</b>
6.1 POSIX互換性 .....	130
6.2 ファイルアクセス権 .....	131
6.3 テクノロジー .....	131
6.4 ファイルシステムのレイアウト .....	131
6.4.1 ファイル .....	131
6.4.2 シンボリックリンク .....	132
6.4.3 デバッグ用の特別なステータス・ファイル (JSON) .....	132
6.4.4 デバッグの有効化／無効化 .....	133
6.5 回復 .....	133
6.5.1 クラスタ構成の削除 .....	133
6.5.2 故障したノードからのゲストの回復/移動 .....	133
<b>7 プロックスモックスVEストレージ</b>	<b>135</b>
7.1 ストレージの種類 .....	135
7.1.1 シン・プロビジョニング .....	136
7.2 ストレージ構成 .....	136
7.2.1 ストレージ・プール .....	137

7.2.2 共通ストレージ物件	137
7.3 卷数	139
7.3.1 数量所有権	139
7.4 コマンドラインインターフェイスの使用	139
7.4.1 例	140
7.5 ディレクトリバックエンド	141
7.5.1 構成	142
7.5.2 ファイルの命名規則	142
7.5.3 ストレージ機能	143
7.5.4 例	143
7.6 NFSバックエンド	144
7.6.1 構成	144
7.6.2 ストレージ機能	145
7.6.3 例	145
7.7 CIFSバックエンド	145
7.7.1 構成	146
7.7.2 ストレージ機能	147
7.7.3 例	147
7.8 Proxmoxバックアップサーバー	147
7.8.1 構成	148
7.8.2 ストレージ機能	149
7.8.3 暗号化	149
7.8.4 例CLIによるストレージの追加	150
7.9 GlusterFSバックエンド	150
7.9.1 構成	151
7.9.2 ファイルの命名規則	151
7.9.3 ストレージ機能	151

7.10 ローカルZFSプール・バックエンド .....	151
7.10.1 構成 .....	152
7.10.2 ファイルの命名規則 .....	152
7.10.3 ストレージ機能 .....	153
7.10.4 例 .....	153
7.11 LVMバックエンド .....	153
7.11.1 構成 .....	153
7.11.2 ファイルの命名規則 .....	154
7.11.3 ストレージ機能 .....	154
7.11.4 例 .....	155
7.12 LVMシン・バックエンド .....	155
7.12.1 構成 .....	155
7.12.2 ファイルの命名規則 .....	155
7.12.3 ストレージ機能 .....	156
7.12.4 例 .....	156
7.13 Open-iSCSIイニシエータ .....	156
7.13.1 構成 .....	156
7.13.2 ファイルの命名規則 .....	157
7.13.3 ストレージ機能 .....	157
7.13.4 例 .....	157
7.14 ユーザーモード iSCSI バックエンド .....	157
7.14.1 構成 .....	158
7.14.2 ストレージ機能 .....	158
7.15 Ceph RADOSブロックデバイス (RBD) .....	158
7.15.1 構成 .....	159
7.15.2 認証 .....	159
7.15.3 Cephクライアントの構成(オプション) .....	160

7.15.4 ストレージ機能 .....	160
7.16 Cephファイルシステム (CephFS) .....	161
7.16.1 構成 .....	161
7.16.2 認証 .....	162
7.16.3 ストレージ機能 .....	163
7.17 BTRFSバックエンド .....	163
7.17.1 構成 .....	163
7.17.2 スナップ写真 .....	164
7.18 ISCSIバックエンド上のZFS .....	164
7.18.1 構成 .....	164
7.18.2 ストレージ機能 .....	166
<b>8 ハイパーコンバージドCephクラスタの展開</b>	<b>167</b>
8.1 はじめに .....	167
8.2 用語解説 .....	168
8.3 健全なCephクラスタの推奨事項 .....	168
8.4 Cephの初期インストールと構成 .....	171
8.4.1 ウェブベースのウィザードの使用 .....	171
8.4.2 CephパッケージのCLIインストール .....	173
8.4.3 CLIによるCephの初期構成 .....	173
8.5 Cephモニター .....	174
8.5.1 モニター作成 .....	174
8.5.2 モニターを破壊する .....	175
8.6 Cephマネージャー .....	175
8.6.1 クリエイトマネージャー .....	175
8.6.2 デストロイ・マネージャー .....	175
8.7 Ceph OSD .....	176
8.7.1 OSDの作成 .....	176

8.7.2 OSDの破棄	178
8.8 セフ・プール	179
8.8.1 プールの作成と編集	179
8.8.2 消去符号化プール	181
8.8.3 デストロイ・プールズ	183
8.8.4 PGオートスケーラー	183
8.9 Ceph CRUSH & デバイスクラス	184
8.10 Cephクライアント	186
8.11 CephFS	187
8.11.1 メタデータ・サーバー (MDS)	187
8.11.2 CephFSの作成	188
8.11.3 CephFSの破棄	189
8.12 Cephメンテナンス	189
8.12.1 OSDの交換	189
8.12.2 トリム/廃棄	190
8.12.3 スクラップ&ディープスクラップ	190
8.13 Cephの監視とトラブルシューティング	190
<b>9 ストレージ・レプリケーション</b>	<b>192</b>
9.1 対応ストレージタイプ	192
9.2 スケジュール形式	193
9.3 エラー処理	193
9.3.1 考えられる問題	193
9.3.2 エラー時のゲストの移行	193
9.3.3 例	193
9.4 求人管理	194
9.5 コマンドライン・インターフェースの例	195
<b>10 QEMU/KVM 仮想マシン</b>	<b>196</b>

10.1 エミュレートされたデバイスと準仮想化されたデバイス .....	196
10.2 仮想マシンの設定 .....	197
10.2.1 一般設定 .....	197
10.2.2 OS設定 .....	198
10.2.3 システム設定 .....	198
10.2.4 ハードディスク .....	199
10.2.5 CPU .....	202
10.2.6 メモリー .....	207
10.2.7 ネットワーク機器 .....	209
10.2.8 表示 .....	210
10.2.9 USBバススルー .....	211
10.2.10 BIOSとUEFI .....	212
10.2.11 トラステッド・プラットフォーム・モジュール (TPM) .....	213
10.2.12 VM間共有メモリ .....	213
10.2.13 オーディオ機器 .....	214
10.2.14 ヴァーチオRNG .....	214
10.2.15 デバイスの起動順序 .....	215
10.2.16 仮想マシンの自動起動とシャットダウン .....	216
10.2.17 QEMUゲストエージェント .....	217
10.2.18 SPICEの強化 .....	218
10.3 マイグレーション .....	219
10.3.1 オンライン移行 .....	220
10.3.2 オフライン移行 .....	220
10.4 コピーとクローン .....	221
10.5 仮想マシンテンプレート .....	222
10.6 VMジェネレーションID .....	222
10.7 仮想マシンのインポート .....	223

10.7.1 インポートウィザード	223
10.7.2 CLIを使ったOVF/OVAのインポート	224
10.8 クラウド・イニト・サポート	226
10.8.1 Cloud-Initテンプレートの準備	226
10.8.2 Cloud-Initテンプレートのデプロイ	228
10.8.3 カスタムクラウドイニット構成	229
10.8.4 クラウドイット特有のオプション	229
10.9 PCI(e)バススルー	231
10.9.1 一般要件	231
10.9.2 ホスト・デバイス・バススルー	233
10.9.3 SR-IOV	236
10.9.4 媒介デバイス（vGPU、GVT-g）	236
10.9.5 クラスターでの使用	237
10.9.6 vIOMMU（エミュレートされたIOMMU）	237
10.10 フックスクリプト	238
10.11 冬眠	238
10.12 リソースマッピング	239
10.13 qmによる仮想マシンの管理	241
10.13.1 CLIの使用例	241
10.14 構成	242
10.14.1 ファイル形式	243
10.14.2 スナップ写真	243
10.14.3 オプション	243
10.15 ロック	271
<b>11 Proxmoxコンテナツールキット</b>	<b>272</b>
11.1 技術概要	272
11.2 支援分配金	273

11.2.1 アルパイン・リナックス .....	273
11.2.2 アーチリナックス .....	273
11.2.3 CentOS, Almalinux, Rocky Linux .....	274
11.2.4 デビアン .....	274
11.2.5 デヴアン .....	275
11.2.6 フェドラー .....	275
11.2.7 ジエンツー .....	275
11.2.8 オープンソース .....	275
11.2.9 ウブントウ .....	275
11.3 コンテナ画像 .....	276
11.4 コンテナ設定 .....	277
11.4.1 一般設定 .....	277
11.4.2 CPU .....	278
11.4.3 メモリー .....	279
11.4.4 マウント・ポイント .....	280
11.4.5 ネットワーク .....	283
11.4.6 コンテナの自動スタートとシャットダウン .....	284
11.4.7 フックスクリプト .....	285
11.5 セキュリティへの配慮 .....	285
11.5.1 AppArmor .....	286
11.5.2 対照群 (cgroup) .....	286
11.6 ゲストオペレーティングシステムの構成 .....	287
11.7 コンテナ保管 .....	289
11.7.1 ヒューズマウント .....	289
11.7.2 コンテナ内でクォータを使う .....	289
11.7.3 コンテナ内部で ACL を使う .....	290
11.7.4 コンテナマウントポイントのバックアップ .....	290
11.7.5 コンテナマウントポイントのレプリケーション .....	290

11.8 バックアップとリストア .....	291
11.8.1 コンテナのバックアップ .....	291
11.8.2 コンテナのバックアップの復元 .....	291
11.9 pctでコンテナを管理する .....	292
11.9.1 CLIの使用例 .....	292
11.9.2 デバッグログの取得 .....	293
11.10 マイグレーション .....	293
11.11 構成 .....	294
11.11.1 ファイル形式 .....	294
11.11.2 スナップショット .....	295
11.11.3 オプション .....	295
11.12 鍵 .....	301
<b>12 ソフトウェア定義ネットワーク</b>	<b>302</b>
12.1 はじめに .....	302
12.2 サポート状況 .....	302
12.2.1 沿革 .....	302
12.2.2 現在の状況 .....	303
12.3 インストール .....	303
12.3.1 SDNコア .....	303
12.3.2 DHCP IPAM .....	303
12.3.3 FRルーティング .....	304
12.4 コンフィギュレーションの概要 .....	304
12.5 技術と構成 .....	304
12.6 ゾーン .....	305
12.6.1 共通オプション .....	305
12.6.2 シンプルゾーン .....	305
12.6.3 VLANゾーン .....	306

12.6.4 秦Qゾーン .....	306
12.6.5 VXLANゾーン .....	306
12.6.6 EVPNゾーン .....	307
12.7 Vネット .....	308
12.8 サブネット .....	309
12.9 コントローラー .....	309
12.9.1 EVPNコントローラー .....	310
12.9.2 BGPコントローラ .....	310
12.9.3 ISISコントローラ .....	311
12.10 アイパム .....	311
12.10.1 PVE IPAM プラグイン .....	311
12.10.2 NetBox IPAM プラグイン .....	312
12.10.3 phpIPAM プラグイン .....	312
12.11 DNS .....	312
12.11.1 PowerDNS プラグイン .....	312
12.12 DHCP .....	313
12.12.1 構成 .....	313
12.12.2 プラグイン .....	314
12.13 例 .....	314
12.13.1 シンプルゾーンの例 .....	315
12.13.2 ソースNATの例 .....	315
12.13.3 VLAN の設定例 .....	316
12.13.4 QinQセットアップの例 .....	316
12.13.5 VXLAN セットアップの例 .....	317
12.13.6 EVPN セットアップの例 .....	318
12.14 備考 .....	319
12.14.1 複数のEVPN出口ノード .....	319
12.14.2 VXLAN IPSEC暗号化 .....	320

<b>13 Proxmox VE ファイアウォール</b>	<b>321</b>
13.1 ゾーン .....	321
13.2 設定ファイル .....	321
13.2.1 クラスターワイドセットアップ .....	322
13.2.2 ホスト固有の設定 .....	323
13.2.3 VM/コンテナの構成 .....	325
13.3 ファイアウォールのルール .....	326
13.4 セキュリティ・グループ .....	327
13.5 IPエイリアス .....	328
13.5.1 標準IPエイリアス local_network .....	328
13.6 IPセット .....	328
13.6.1 標準IPセット管理 .....	329
13.6.2 標準IPセットブラックリスト .....	329
13.6.3 標準IPセット ipfilter-net* .....	329
13.7 サービスとコマンド .....	329
13.8 デフォルトのファイアウォールルール .....	330
13.8.1 データセンター発着 DROP/REJECT .....	330
13.8.2 VM/CT着信/発信 DROP/REJECT .....	331
13.9 ファイアウォールルールのログ .....	331
13.9.1 ユーザー定義ファイアウォールルールのログ .....	332
13.10 ヒントとコツ .....	332
13.10.1 FTPを許可する方法 .....	332
13.10.2 スリカータIPS統合 .....	333
13.11 IPv6に関する注意事項 .....	333
13.12 Proxmox VEが使用するポート .....	333
13.13 nftables .....	334
13.13.1 インストールと使用方法 .....	334
13.13.2 使用方法 .....	335
13.13.3 役立つコマンド .....	335

<b>14 ユーザー管理</b>	<b>337</b>
14.1 ユーザー .....	337
14.1.1 システム管理者 .....	338
14.2 グループ .....	338
14.3 APIトークン .....	338
14.4 リソースプール .....	338
14.5 認証領域 .....	339
14.5.1 Linux PAM 標準認証 .....	339
14.5.2 Proxmox VE 認証サーバ .....	339
14.5.3 LDAP .....	340
14.5.4 マイクロソフト・アクティブ・ディレクトリ (AD) .....	341
14.5.5 LDAPベースのレルムの同期 .....	341
14.5.6 OpenID Connect .....	344
14.6 二要素認証 .....	345
14.6.1 利用可能なセカンドファクター .....	346
14.6.2 レルム強制二要素認証 .....	346
14.6.3 二要素認証の制限とロックアウト .....	347
14.6.4 ユーザーによるTOTP認証の設定 .....	347
14.6.5 TOTP .....	348
14.6.6 ウェブオート .....	348
14.6.7 リカバリーキー .....	349
14.6.8 サーバーサイドWebauthnの設定 .....	349
14.6.9 サーバーサイドU2Fの設定 .....	349
14.6.10 ユーザーとしてU2Fをアクティベートする .....	350
14.7 許可管理 .....	350
14.7.1 役割 .....	351
14.7.2 特典 .....	352
14.7.3 オブジェクトとパス .....	353
14.7.4 プール .....	354

14.7.5 どのパーミッションが必要か?	354
14.8 コマンドラインツール	355
14.9 実例	356
14.9.1 管理者グループ	356
14.9.2 監査役	356
14.9.3 ユーザー管理の委任	357
14.9.4 監視用限定APIトークン	357
14.9.5 リソースプール	357
<b>15 高い可用性</b>	<b>359</b>
15.1 必要条件	360
15.2 リソース	361
15.3 マネジメント・タスク	361
15.4 仕組み	362
15.4.1 サービス・ステート	363
15.4.2 ローカル・リソース・マネージャー	364
15.4.3 クラスター・リソース・マネージャー	365
15.5 HAシミュレーター	366
15.6 構成	367
15.6.1 リソース	367
15.6.2 グループ	369
15.7 フェンシング	371
15.7.1 プロックスモックスVEフェンス	371
15.7.2 ハードウェア・ウォッチドッグの設定	372
15.7.3 リカバー・フェンス・サービス	372
15.8 スタート失敗のポリシー	372
15.9 エラーリカバリー	373
15.10 パッケージ・アップデート	373

15.11 ノードのメンテナンス .....	374
15.11.1 メンテナンス・モード .....	374
15.11.2 シャットダウン・ポリシー .....	375
15.12 クラスタ・リソース・スケジューリング .....	376
15.12.1 基本スケジューラ .....	377
15.12.2 静的負荷スケジューラ .....	377
15.12.3 CRSスケジューリングポイント .....	378
<b>16 バックアップとリストア</b> .....	<b>379</b>
16.1 バックアップモード .....	379
16.1.1 VMバックアップフリッキング .....	381
16.2 バックアップファイル名 .....	381
16.3 バックアップファイルの圧縮 .....	381
16.4 バックアップの暗号化 .....	382
16.5 バックアップの仕事 .....	383
16.6 バックアップ保持 .....	385
16.6.1 ブルーンシミュレータ .....	386
16.6.2 保持設定例 .....	386
16.7 バックアップ保護 .....	387
16.8 バックアップノート .....	387
16.9 リストア .....	388
16.9.1 帯域幅の制限 .....	388
16.9.2 ライブ・レストア .....	389
16.9.3 単一ファイルの復元 .....	389
16.10 構成 .....	390
16.11 フックスクリプト .....	393
16.12 ファイルの除外 .....	393
16.13 例 .....	394

<b>17 お知らせ</b>	<b>395</b>
17.1 概要	395
17.2 通知対象	396
17.2.1 センドメール	396
17.2.2 SMTP	397
17.2.3 ゴティファイ	398
17.3 通知マッチャー	399
17.3.1 マッチャーオプション	399
17.3.2 カレンダー・マッチング・ルール	400
17.3.3 フィールド・マッチング・ルール	400
17.3.4 深刻度マッチングルール	400
17.3.5 例	400
17.4 通知イベント	401
17.5 システム・メール転送	401
17.6 アクセス許可	402
<b>18 重要なサービス・デーモン</b>	<b>403</b>
18.1 pvedaemon - Proxmox VE API デーモン	403
18.2 pveproxy - Proxmox VE API プロキシデーモン	403
18.2.1 ホストベースのアクセス制御	403
18.2.2 リスニングIPアドレス	404
18.2.3 SSL暗号スイート	405
18.2.4 対応TLSバージョン	405
18.2.5 ディフィー・ヘルマン パラメータ	405
18.2.6 代替HTTPS証明書	406
18.2.7 レスポンス・コンプレッション	406
18.3 pvestatd - Proxmox VE ステータスデーモン	406
18.4 spiceproxy - SPICE プロキシサービス	406

---

18.4.1 ホストベースのアクセス制御 .....	406
18.5 pvescheduler - Proxmox VE スケジューラーデーモン .....	407
<b>19 便利なコマンドラインツール</b>	<b>408</b>
19.1 pvesubscription - サブスクリプション管理 .....	408
19.2 pveperf - Proxmox VE ベンチマークスクリプト .....	408
19.3 Proxmox VE API 用シェルインターフェイス .....	409
19.3.1 使用例 .....	409
<b>20 よくある質問</b>	<b>410</b>
<b>21 書誌</b>	<b>413</b>
21.1 プロックスモックスVEに関する書籍 .....	413
21.2 関連技術に関する書籍 .....	413
21.3 関連書籍 .....	414
<b>A コマンドライン・インターフェース</b>	<b>415</b>
A.1 出力フォーマットのオプション [FORMAT_OPTIONS] .....	415
A.2 pvesm - Proxmox VE ストレージマネージャ .....	416
A.3 pvesubscription - Proxmox VE サブスクリプションマネージャ .....	430
A.4 pveperf - Proxmox VE ベンチマークスクリプト .....	431
A.5 pveceph - Proxmox VE ノードで CEPH サービスを管理する .....	431
A.6 pvenode - Proxmox VE ノード管理 .....	439
A.7 pvesh - Proxmox VE API 用シェルインターフェイス .....	447
A.8 qm - QEMU/KVM 仮想マシンマネージャ .....	448
A.9 qmrestore - QemuServer vzdump バックアップをリストアする .....	491
A.10 pct - Proxmoxコンテナツールキット .....	492
A.11 pveam - Proxmox VE アプライアンスマネージャ .....	516
A.12 pvecm - Proxmox VE クラスタマネージャ .....	517
A.13 pvesr - Proxmox VE ストレージレプリケーション .....	520

---

---

A.14 <b>pveum</b> - Proxmox VE ユーザーマネージャ	525
A.15 <b>vzdump</b> - VMとコンテナのバックアップユーティリティ	540
A.16 <b>ha-manager</b> - Proxmox VE HA マネージャ	543
<b>B サービス・デーモン</b>	<b>549</b>
B.1 <b>pve-firewall</b> - Proxmox VE ファイアウォールデーモン	549
B.2 <b>pvedaemon</b> - Proxmox VE API デーモン	550
B.3 <b>pveproxy</b> - Proxmox VE API プロキシデーモン	551
B.4 <b>pvestatd</b> - Proxmox VE ステータスデーモン	552
B.5 <b>spiceproxy</b> - SPICE プロキシサービス	552
B.6 <b>pmxcfs</b> - Proxmox クラスタファイルシステム	553
B.7 <b>pve-ha-crm</b> - クラスタリソーススマネージャデーモン	553
B.8 <b>pve-ha-lrm</b> - ローカルリソーススマネージャーデーモン	554
B.9 <b>pvescheduler</b> - Proxmox VE スケジューラデーモン	555
<b>C 設定ファイル</b>	<b>556</b>
C.1 データセンターの構成	556
C.1.1 ファイル形式	556
C.1.2 オプション	556
<b>D カレンダー イベント</b>	<b>562</b>
D.1 スケジュール形式	562
D.2 詳細仕様	562
D.2.1 例	563
<b>E QEMU vCPUリスト</b>	<b>565</b>
E.1 はじめに	565
E.2 インテル CPU タイプ	565
E.3 AMD CPU タイプ	566
<b>F ファイアウォールマクロの定義</b>	<b>567</b>

---

---

<b>G マークダウン入門</b>	<b>582</b>
G.1 マークダウンの基本 .....	582
G.1.1 見出し .....	582
G.1.2 強調 .....	582
G.1.3 リンク .....	583
G.1.4 リスト .....	583
G.1.5 テーブル .....	584
G.1.6 ブロック名言集 .....	584
G.1.7 コードとスニペット .....	584

**H GNU自由文書ライセンス****586**

# 第1章 はじめに

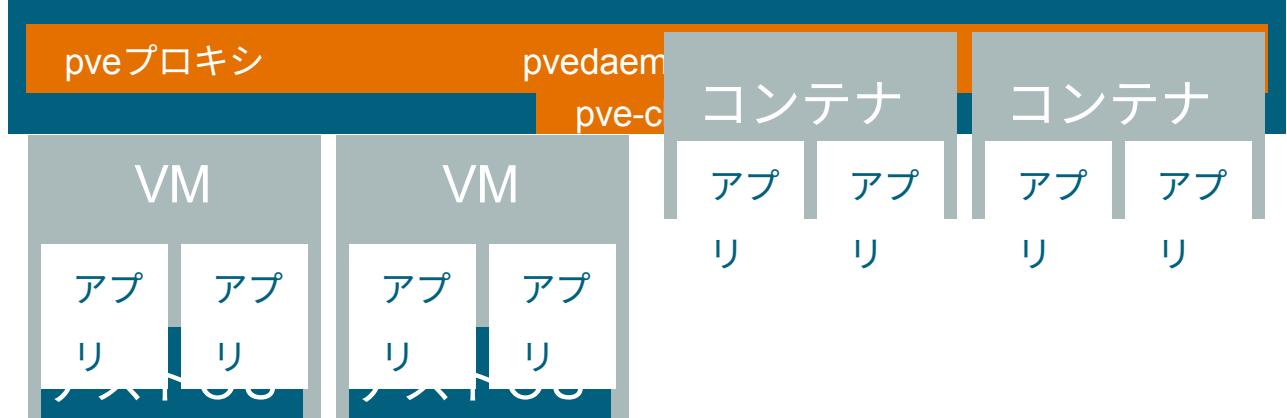
Proxmox VEは、仮想マシンとコンテナを実行するためのプラットフォームです。Debian Linuxベースで、完全にオープンソースです。最大限の柔軟性を実現するために、カーネルベースの仮想マシン（KVM）とコンテナベースの仮想化（LXC）という2つの仮想化技術を実装しました。

主な設計目標の1つは、管理ができるだけ簡単にすることでした。Proxmox VEは單一ノードで使用することも、多数のノードからなるクラスタを構築することもできます。すべての管理タスクはWebベースの管理インターフェイスを使用して行うことができ、初心者ユーザでもProxmox VEのセットアップとインストールを数分で行うことができます。

## ユーザーツール

qm	pvesm	はげ	管理人
パーティション	ピーブイシーエム	ヴェーセフ	PVEファイアウォール

## サービス



## QEMU

KVM

Linuxカーネル

AppArmor cグループ

## 1.1 中央管理

多くの人はシングルノードから始めますが、Proxmox VEは大規模なクラスタノードにスケールアウトすることができます。クラスタスタックは完全に統合されており、デフォルトのインストールで同梱されています。

### ユニークなマルチマスター・デザイン

統合されたウェブベースの管理インターフェイスにより、すべてのKVMゲストとLinuxコンテナ、さらにはクラスタ全体の概要が一目瞭然です。GUIからVMやコンテナ、ストレージ、クラスタを簡単に管理できます。複雑で高価な管理サーバを別途インストールする必要はありません。

### Proxmox クラスタファイルシステム (pmxcfs)

Proxmox VEは、独自のProxmox Clusterファイルシステム(pmxcfs)を使用しています。pmxcfsは、設定ファイルを格納するためのデータベース駆動型のファイルシステムです。これにより、何千もの仮想マシンの設定を保存できます。corosyncを使用することで、これらのファイルはすべてのクラスタノードでリアルタイムに複製されます。ファイルシステムは、すべてのデータをディスク上の永続データベース内に格納しますが、データのコピーはRAMに存在し、最大30MBのストレージサイズを提供します。

Proxmox VEは、このユニークなクラスタファイルシステムを使用する唯一の仮想化プラットフォームです。

### ウェブベースの管理インターフェース

Proxmox VEの使い方は簡単です。管理タスクは付属のWebベースの管理インターフェイスで実行できます。別の管理ツールをインストールしたり、巨大なデータベースを持つ管理ノードを追加したりする必要はありません。マルチマスターツールにより、クラスタのどのノードからでもクラスタ全体を管理できます。JavaScriptフレームワーク(Ex-tJS)をベースとしたWebベースの中央管理により、GUIからすべての機能を制御し、各ノードの履歴やシスログを確認することができます。これには、バッカアップやリストアジョブの実行、ライブマイグレーションやHAトリガーアクティビティが含まれます。

### コマンドライン

Unix シェルや Windows Powershell の使い心地に慣れた上級ユーザのために、Proxmox VE は仮想環

境の全コンポーネントを管理するコマンドラインインターフェースを提供します。このコマンドラインインターフェースは、インテリジェントなタブ補完とUNIX manページ形式の完全なドキュメントを備えています。

## REST API

Proxmox VEはRESTful APIを使用しています。主要なデータ形式としてJSONを選択し、API全体はJSONスキーマを使用して正式に定義されています。これにより、カスタムホスティング環境のようなサードパーティの管理ツールと迅速かつ容易に統合することができます。

## 役割ベースの管理

ロールベースのユーザ・権限管理を使用することで、すべてのオブジェクト（VM、ストレージ、ノードなど）に対するきめ細かなアクセスを定義できます。これにより、権限を定義し、オブジェクトへのアクセスを制御することができます。この概念はアクセス制御リストとしても知られている：各許可は、特定のパス上のサブジェクト（ユーザーまたはグループ）とロール（権限のセット）を指定します。

## 認証領域

Proxmox VEは、Microsoft Active Directory、LDAP、Linux PAM標準認証、または組み込みのProxmox VE認証サーバなど、複数の認証ソースをサポートしています。

## 1.2 柔軟なストレージ

Proxmox VEのストレージモデルは非常に柔軟です。仮想マシンイメージは、1つまたは複数のローカルストレージ、またはNFSやSANなどの共有ストレージに保存できます。制限はなく、好きなだけストレージ定義を設定できます。Debian Linuxで利用可能なすべてのストレージ技術を使用できます。

VMを共有ストレージに格納する主な利点の1つは、クラスタ内のすべてのノードがVMのディスクイメージに直接アクセスできるため、ダウンタイムなしに稼働中のマシンをライブマイグレーションできることだ。

現在、以下のネットワーク・ストレージ・タイプをサポートしています：

- LVMグループ (iSCSIターゲットによるネットワークバックアップ)
- iSCSIターゲット
- NFSシェア
- CIFS共有
- Ceph RBD
- iSCSI LUNを直接使用する
- GlusterFS

対応するローカル・ストレージ・タイプは以下の通り：

- LVMグループ (ブロック・デバイス、FCデバイス、DRBDなどのローカル・バックティング・デバイス)
- ディレクトリ (既存のファイルシステム上のストレージ)
- ゼットエフエス

## 1.3 統合されたバックアップとリストア

統合バックアップツール (`vzdump`) は、実行中のコンテナとKVMゲストの一貫したスナップショットを作成します。基本的には、VM/CT設定ファイルを含むVMまたはCTデータのアーカイブを作成します。

KVMライブバックアップは、NFS、CIFS、iSCSI LUN、Ceph RBD上のVMイメージを含むすべてのストレージタイプで動作します。新しいバックアップフォーマットは、VMバックアップを高速かつ効果的に保存する

ために最適化されています（疎なファイル、順序のずれたデータ、最小化されたI/O）。

## 1.4 高可用性クラスタ

マルチノードのProxmox VE HA Clusterは、可用性の高い仮想サーバの定義を可能にします。Proxmox VE HA Clusterは、実績のあるLinux HAテクノロジに基づいており、安定した信頼性の高いHAサービスを提供します。

## 1.5 柔軟なネットワーキング

Proxmox VEはブリッジネットワークモデルを採用しています。各ゲストからの仮想ネットワークケーブルがすべて同じスイッチに接続されているかのように、すべてのVMが1つのブリッジを共有できます。VMを外部に接続するために、ブリッジは物理ネットワークカードに接続され、TCP/IPコンフィギュレーションが割り当てられます。

さらに柔軟性を高めるために、VLAN (IEEE 802.1q)とネットワークボンディング/アグリゲーションが可能です。このようにして、Linuxネットワークスタックのフルパワーを活用して、Proxmox VEホスト用に複雑で柔軟な仮想ネットワークを構築することができます。

## 1.6 統合ファイアウォール

統合されたファイアウォールにより、任意のVMまたはコンテナ・インターフェース上のネットワーク・パケットをフィルタリングすることができます。共通のファイアウォールルールセットは、「セキュリティグループ」にグループ化することができます。

## 1.7 ハイパーコンバージド・インフラストラクチャー

Proxmox VEは、コンピュート、ストレージ、ネットワークリソースを緊密に統合し、可用性の高いクラスタ、バックアップ/リストア、ディザスタリカバリを管理する仮想化プラットフォームです。すべてのコンポーネントはソフトウェアで定義され、互いに互換性があります。

そのため、一元化されたウェブ管理インターフェイスを介して、単一のシステムのように管理することが可能です。これらの機能により、Proxmox VEはオープンソースのハイパーコンバージドインフラストラクチャの導入と管理に理想的な選択肢となります。

### 1.7.1 Proxmox VEによるハイパーコンバージドインフラ (HCI) のメリット

ハイパーコンバージドインフラストラクチャ (HCI) は、高いインフラ需要が低い管理予算に見合う展開、リモートオフィスやブランチオフィス環境のような分散セットアップ、または仮想プライベートクラウドやパブリッククラウドに特に有用である。

HCIには次のような利点がある：

- ・ 拡張性：コンピュート、ネットワーク、ストレージデバイスのシームレスな拡張（つまり、サーバーとストレージを互いに独立して迅速に拡張）。
- ・ 低コスト：Proxmox VEはオープンソースであり、コンピュート、ストレージ、ネットワーク、バックアップ、管理センターなど必要なコンポーネントをすべて統合しています。高価なコンピュート/ストレージインフラストラクチャを置き換えることができます。
- ・ データ保護と効率化：バックアップやディザスタリカバリなどのサービスが統合されている。
- ・ シンプルさ：簡単な設定と集中管理。
- ・ オープンソース：ベンダーロックインがない。

## 1.7.2 ハイパー・コンバージド・インフラストラクチャ・ストレージ

Proxmox VE は、ハイパー・コンバージド・ストレージ・インフラストラクチャを展開するためのサポートを緊密に統合しています。例えば、Web インタフェースのみを使用して、以下の 2 つのストレージ・テクノロジを開発および管理できます：

- **Ceph**: 自己修復と自己管理の両方を備えた、信頼性が高く拡張性の高い共有ストレージ・システムです。  
[Proxmox VE ノードで Ceph サービスを管理する方法](#)をご覧ください。
- **ZFS**: ファイルシステムと論理ボリュームマネージャを組み合わせたもので、データの破損に対する広範な保護、さまざまな RAID モード、高速で安価なスナップショットなどの機能を備えています。  
[Proxmox VE ノードで ZFS のパワーを活用する方法](#)をご覧ください。

上記以外にも、Proxmox VE は幅広い追加ストレージ・テクノロジの統合をサポートしています。これらについては[ストレージ・マネージャの章](#)を参照してください。

## 1.8 なぜオープンソースなのか

Proxmox VE は Linux カーネルを使用し、Debian GNU/Linux ディストリビューションをベースにしています。Proxmox VE のソースコードは、[GNU Affero General Public License, version 3](#) の下でリリースされています。つまり、いつでも自由にソースコードを閲覧したり、プロジェクトに貢献することができます。

Proxmox では、可能な限りオープンソース・ソフトウェアを使用することをお約束します。オープンソース・ソフトウェアを使用することで、すべての機能へのフルアクセスが保証され、高いセキュリティと信頼性が保証されます。私たちは、誰もがソフトウェアのソースコードにアクセスし、それを実行したり、ビルドしたり、プロジェクトに変更を提出する権利を持つべきだと考えています。Proxmox は、製品が常にプロフェッショナルな品質基準を満たしていることを保証する一方で、誰もが貢献することが奨励されています。

オープンソース・ソフトウェアはまた、コストを低く抑え、コアインフラを単一のベンダーから独立させるのに役立つ。

## 1.9 Proxmox VE のメリット

- オープンソースソフトウェア
- ベンダーロックインなし
- Linuxカーネル
- 迅速な設置と使いやすさ
- ウェブベースの管理インターフェース
- REST API
- 巨大で活発なコミュニティ
- 低い管理コストとシンプルな配備

## 1.10 ヘルプを得る

### 1.10.1 プロックスモックス VE ウィキ

主な情報源は[Proxmox VE Wiki](#)です。リファレンスドキュメントとユーザー投稿コンテンツが統合されています。

### 1.10.2 コミュニティ・サポート・フォーラム

Proxmox VE自体は完全にオープンソースであるため、[Proxmox VE コミュニティフォーラム](#)を使用して、ユーザが議論し、ノウハウを共有することを常に奨励しています。このフォーラムはProxmoxのサポートチームによって管理されており、世界中から多くのユーザーが集まっています。言うまでもなく、このような大規模なフォーラムは情報を得るのに最適な場所です。

### 1.10.3 メーリングリスト

Proxmox VEコミュニティとEメールで迅速に連絡を取ることができます。

- ユーザー向けメーリングリスト[Proxmox VE ユーザーリスト](#)

Proxmox VEは完全にオープンソースであり、貢献を歓迎します！開発者の主なコミュニケーションチャネルは次のとおりです：

- 開発者向けメーリングリスト[Proxmox VE 開発ディスカッション](#)

### 1.10.4 コマーシャル・サポート

Proxmox Server Solutions GmbHは、[Proxmox VEサブスクリプションサービスプランとして](#)エンタープライズサポートも提供しています。サブスクリプションを利用するすべてのユーザは、[Proxmox VE エンタープライズリポジトリ](#)にアクセスでき、ベーシック、スタンダード、またはプレミアムサブスクリプションを利用することで、Proxmoxカスタマーポータルにもアクセスできます。カスタマーポータルでは、Proxmox VE開発者による応答時間を保証したヘルプとサポートを提供します。

ボリュームディスカウント、または詳細については、[sales@proxmox.com](mailto:sales@proxmox.com)までお問い合わせください。

### 1.10.5 バグトラッカー

Proxmoxは<https://bugzilla.proxmox.com> で公開バグトラッカーを運営しています。問題が発生した場合は、そこに報告してください。問題はバグだけでなく、新機能や機能強化の要望でも構いません。バグトラッカーは問題の追跡に役立ち、問題が解決されると通知を送信します。

## 1.11 プロジェクトの歴史

プロジェクトは2007年に始まり、2008年に最初の安定版がリリースされた。当時、コンテナにはOpenVZ、仮想マシンにはKVMを使っていた。クラスタリング機能は限定的で、ユーザー・インターフェースはシンプル（サーバーが生成するウェブ・ページ）だった。

しかし、私たちはCorosyncクラスタスタックを使った新機能をすぐに開発しました。新しいProxmoxクラスタファイルシステム (pmxcfs) の導入は、クラスタの複雑さをユーザーから完全に隠すことができるため、大きな前進でした。16ノードのクラスタを管理するのは、1つのノードを管理するのと同じくらい簡単です。

また、JSON-Schemaで記述された完全な宣言的仕様の新しいREST APIも導入しました。これにより、他の人々がProxmox VEをインフラに統合し、追加サービスを簡単に提供できるようになりました。

また、新しいREST APIにより、JavaScriptを使用したモダンなHTML5アプリケーションでオリジナルのユーザーインターフェイスを置き換えることが可能になりました。また、古いJavaベースのVNCコンソールコードをnoVNCに置き換えた。そのため、VMを管理するのに必要なのはウェブブラウザだけです。

様々なストレージタイプのサポートも大きな課題だ。注目すべきは、Proxmox VEが2014年にLinux上でZFSをデフォルトで出荷した最初のディストリビューションであったことだ。もう一つのマイルストーンは、ハイパーバイザー・ノード上でCephストレージを実行・管理できるようになったことだ。このようなセットアップは非常に費用対効果が高い。

KVMの商用サポートを提供する最初の企業のひとつでした。KVMプロジェクト自体は継続的に進化し、今では広く使われているハイパーバイザーです。リリースのたびに新機能が追加されています。私たちはKVMライブ・バックアップ機能を開発し、あらゆるストレージ・タイプにスナップショット・バックアップを作成できるようにしました。

バージョン4.0で最も注目すべき変更は、OpenVZからLXCへの移行だ。コンテナは深く統合され、仮想マシンと同じストレージとネットワーク機能を使用できるようになった。

## 1.12 Proxmox VE ドキュメントの改善

Proxmox VE ドキュメントへの貢献や改良はいつでも歓迎します。貢献する方法はいくつかあります。

このドキュメントに誤りや改善の余地がある場合は、[Proxmoxのバグトラッカー](#)にバグを報告し、修正を提案してください。

新しいコンテンツを提案したい場合は、以下のオプションのいずれかを選択してください：

- ・ ウィキ特定のセットアップ、ハウツーガイド、チュートリアルについては、wikiが貢献するのに適したオプションです。

- リファレンス・ドキュメントすべてのユーザーに役立つ一般的な内容については、リファレンスドキュメンテーションに貢献してください。これにはProxmox VEの機能のインストール、設定、使用方法、トラブルシューティングに関するすべての情報が含まれます。リファレンスドキュメントはasciidoc形式で書かれています。リファレンス・ドキュメントはasciidoc形式で書かれています。  
ドキュメントを編集するには、`git://git.proxmox.com/git/pve-docs` にある git リポジトリをクローンする必要があります。  
その後、[README.adoc](#)文書に従ってください。

---

## 注

Proxmox VEのコードベースに興味がある方は、[Developer Documentation](#) wikiの記事から始められます。

---

## 1.13 Proxmox VEの翻訳

Proxmox VEのユーザーインターフェースはデフォルトで英語です。しかし、コミュニティーの貢献により、他の言語への翻訳も可能です。新しい言語の追加、最新機能の翻訳、不完全な翻訳や一貫性のない翻訳の改善などのサポートを歓迎します。

翻訳ファイルの管理にはgettextを使っています。Poeditのようなツールは、翻訳ファイルを編集するのに便利なユーザーインターフェイスを提供していますが、使い慣れたエディタでもかまいません。翻訳にプログラミングの知識は必要ありません。

### 1.13.1 gitを使った翻訳

言語ファイルはgitリポジトリとして公開されています。gitに詳しい方は、開発者向けドキュメントに従って貢献してください。

次のようにして、新しい翻訳を作成することができます（<LANG>を言語IDに置き換えてください）：

```
# git clone git://git.proxmox.com/git/proxmox-i18n.git # cd proxmox-i18n  
# make init-<LANG>.po
```

または、お好みのエディタを使用して、既存の翻訳を編集することができます：

```
# poedit <LANG>.po
```

### 1.13.2 gitを使わない翻訳

gitに精通していないなくても、Proxmox VEの翻訳を手伝うことができます。まず、ここから言語ファイルをダウンロードしてください。改善したい言語を見つけ、その言語ファイルの「raw」リンクを右クリックして「名前を付けてリンク先を保存」を選択します。ファイルに変更を加え、最終的な翻訳を直接送信してください。

をoffice(at)proxmox.comまで、署名済みのコントリビュータライセンス契約書とともにお送りください。

### 1.13.3 翻訳のテスト

Proxmox VEで翻訳を使用するには、まず.poファイルを.jsファイルに翻訳する必要があります。これは同じリポジトリにある以下のスクリプトを実行することで行えます：

```
# ./po2js.pl -t pve xx.po >pve-lang-xx.js
```

出来上がったpve-lang-xx.jsはproxmoxサーバーの/usr/share/pve-i18nディレクトリにコピーしてテストすることができます。

あるいは、リポジトリのルートから以下のコマンドを実行して、debパッケージをビルドすることもできる：

```
# make deb
```

---

**重要**

どちらの方法でも動作させるには、以下のperlパッケージがシステムにインストールされている必要があります。Debian/Ubuntuの場合：

---

```
# apt-get install perl liblocale-po-perl libjson-perl
```

#### 1.13.4 翻訳の送信

完成した翻訳(.poファイル)は、署名済みのコントリビュータライセンス契約書とともに、Proxmoxチームのアドレス office(at)proxmox.comに送ることができます。また、開発経験があれば、Proxmox VE開発マーリングリストにパッチとして送ることもできます。[開発者向けドキュメント](#)を参照してください。

## 第2章

# Proxmox VEのインストール

Proxmox VEはDebianをベースにしています。このため、Proxmoxが提供するインストールディスクイメージ（ISOファイル）には、完全なDebianシステムと必要なProxmox VEパッケージが含まれています。

### チップ

Proxmox VE のリリースと Debian のリリースの関係については [FAQ のサポート表](#) を参照してください。

インストーラーがセットアップをガイドし、ローカルディスクのパーティション設定、基本的なシステム設定（タイムゾーン、言語、ネットワークなど）、必要なパッケージのインストールを行います。このプロセスに数分以上かかることはありません。提供された ISO を使ってインストールするのが、新規および既存のユーザーに推奨される方法です。

また、Proxmox VE を既存の Debian システムの上にインストールすることもできます。Proxmox VEに関する詳細な知識が必要なため、このオプションは上級ユーザのみにお勧めします。

## 2.1 システム要件

Proxmox VEを本番環境で実行する場合は、高品質のサーバーハードウェアを使用することをお勧めします。障害が発生したホストの影響をさらに軽減するために、高可用性（HA）仮想マシンおよびコンテナを使用したクラスタでProxmox VEを実行できます。

Proxmox VEは、ローカルストレージ(DAS)、SAN、NAS、およびCeph RBDのような分散ストレージを使用できます。詳細は[ストレージの章](#)を参照してください。

## 2.1.1 評価のための最低条件

これらの最小要件は評価のみを目的としたものであり、本番では使用しないでください。

- CPU64ビット（インテルEMT64またはAMD64）
- KVM完全仮想化対応のインテルVT/AMD-V対応CPU/マザーボード
- RAM: 1 GB RAM、およびゲストに必要な追加RAM
- ハードドライブ
- ネットワークカード（NIC）1枚

## 2.1.2 推奨システム要件

- Intel VT/AMD-V CPUフラグを持つIntel EMT64またはAMD64。
- メモリOS および Proxmox VE サービス用に最低 2 GB、さらにゲスト用に指定メモリが必要です。CephとZFSについては、使用するストレージのTBごとに約1GBのメモリを追加する必要があります。
- 高速で冗長性のあるストレージは、SSDで最高の結果が得られます。
- OSストレージ：バッテリー保護ライトキャッシュ ("BBU") 付きハードウェアRAIDを使用するか、ZFSを使用した非RAIDを使用する (ZIL用オプションSSD)。
- VMストレージ：
  - ローカルストレージには、バッテリバックアップライトキャッシュ(BBU)付きのハードウェアRAIDを使用するか、ZFSとCephには非RAIDを使用します。ZFSもCephもハードウェアRAIDコントローラとは互換性がありません。
  - 共有・分散ストレージが可能。
  - 良好的なパフォーマンスを得るためにパワーロス保護 (PLP) 機能付きのSSDを推奨します。民生用SSDの使用は推奨されません。
- 冗長(マルチ)Gbit NIC。ご希望のストレージ技術やクラスタ設定に応じてNICを追加できます。
- PCI(e)バススルーのためにCPUがVT-d/AMD-dフラグをサポートする必要がある。

## 2.1.3 シンプルなパフォーマンス概要

インストールされたProxmox VEシステムのCPUとハードディスクのパフォーマンスの概要を知るには、付属のpveperfツールを実行します。

---

### 注

これは非常に迅速で一般的なベンチマークです。特にシステムのI/O性能については、より詳細なテストをお勧めします。

---

## 2.1.4 ウェブ・インターフェイスにアクセスするためにサポートされているウェブ・ブラウザ

ウェブベースのユーザーインターフェイスにアクセスするには、以下のブラウザのいずれかを使用することをお勧めします：

- Firefox、当年のリリース、または最新の拡張サポートリリース
- クローム、今年リリース
- マイクロソフトが現在サポートしているEdgeのバージョン
- サファリ、今年からのリリース

モバイルデバイスからアクセスすると、Proxmox VEは軽量でタッチベースのインターフェースを表示します。

## 2.2 インストールメディアの準備

インストーラーISOイメージのダウンロード: <https://www.proxmox.com/en/downloads/proxmox-virtual-environment-/iso>

Proxmox VEのインストールメディアはハイブリッドISOイメージです。これは2つの方法で動作します:

- CDやDVDに書き込むISOイメージファイル。
- USBフラッシュドライブ (USBスティック) にコピーするための生セクタ (IMG) イメージファイル。

Proxmox VEをインストールするには、USBフラッシュドライブを使用することをお勧めします。

### 2.2.1 インストールメディアとしてUSBフラッシュドライブを用意する

フラッシュドライブには、少なくとも1GBのストレージが必要です。

---

#### 注

UNetbootinは使用しないでください。Proxmox VEのインストールイメージでは動作しません。

---



#### 重要

USBフラッシュドライブがマウントされておらず、重要なデータが入っていないことを確認してください。

---

### 2.2.2 GNU/Linux用説明書

Unix系OSでは、ddコマンドを使ってISOイメージをUSBフラッシュドライブにコピーします。まず、USBフラッシュドライブの正しいデバイス名を見つけてください（下記参照）。次にddコマンドを実行します。

```
# dd bs=1M conv=fdatasync if=./proxmox-ve_*.iso of=/dev/XYZ
```

---

#### 注

/dev/XYZを正しいデバイス名に置き換えて、入力ファイル名（もしあれば）のパスを合わせる。

---



### 注意

間違ったディスクを上書きしないよう、十分注意してください！

---

## 正しいUSBデバイス名を見つける

USBフラッシュ・ドライブの名前を調べるには、2つの方法がある。1つ目は、フラッシュ・ドライブを差し込む前と後の`dmesg`コマンド出力の最終行を比較する方法。もう1つは、`lsblk`コマンドの出力を比較する

```
# lsblk
```

方法である。ターミナルを開いて実行する：

それからUSBフラッシュ・ドライブを接続し、もう一度コマンドを実行する：

```
# lsblk
```

新しいデバイスが表示されます。これが使用するデバイスです。念のため、報告されたサイズがお使いのUSBフラッシュ・ドライブと一致しているかどうかを確認してください。

### 2.2.3 macOS用説明書

ターミナルを開く（SpotlightでTerminalと問い合わせる）。

`hdiutil`の`convert`オプションなどを使って、`.iso`ファイルを`.dmg`フォーマットに変換する：

```
# hdiutil convert proxmox-ve_*.iso -format UDRW -o proxmox-ve_*.dmg
```

---

#### チップ

macOSは出力ファイル名に自動的に`.dmg`を付ける傾向がある。

---

現在のデバイスのリストを取得するには、次のコマンドを実行する：

```
# diskutil リスト
```

USBフラッシュ・ドライブを挿入し、このコマンドをもう一度実行して、どのデバイス・ノードが割り当てられて

```
# diskutil リスト  
# diskutil unmountDisk /dev/diskX
```

いるかを確認する。（例：`/dev/diskX`）。

---

#### 注

を最後のコマンドのディスク番号に置き換える。

---

```
# sudo dd if=proxmox-ve_*.dmg bs=1M of=/dev/rdiskX
```

---

### 注

最後のコマンドでは、diskXの代わりにrdiskXを指定する。これにより書き込み速度が向上する。

---

## 2.2.4 Windows用説明書

### エッチャーの使用

Etcherは箱から出してすぐに使えます。<https://etcher.io> から Etcher をダウンロードしてください。EtcherはISOとUSBフラッシュドライブの選択プロセスを案内します。

### ルーファスの使用

Rufusはより軽量な代替品だが、動作させるには**DDモード**を使用する必要がある。Rufusを<https://rufus.ie/>。インストールするか、ポータブル版を使用してください。インストール先ドライブと Proxmox VE ISO ファイルを選択します。



#### 重要

開始したら、異なるバージョンのGRUBをダウンロードするよう求めるダイアログで「いいえ」をクリックする必要があります。次のダイアログで**DDモード**を選択します。

## 2.3 Proxmox VEインストーラの使用

インストーラーのISOイメージには以下が含まれている：

- 完全なオペレーティングシステム (Debian Linux、64ビット)
- Proxmox VEインストーラは、ローカルディスクをext4、XFS、BTRFS（テクノロジープレビュー）、またはZFSでパーティション分割し、オペレーティングシステムをインストールします。
- KVMとLXCをサポートするProxmox VE Linuxカーネル
- 仮想マシン、コンテナ、ホストシステム、クラスタ、および必要なすべてのリソースを管理するための完全なツールセット。
- ウェブベースの管理インターフェース

---

#### 注

---

選択したドライブ上の既存のデータは、インストール中にすべて削除されます。インストーラは、他のオペレーティングシステムのブートメニューエントリを追加しません。

---

用意したインストールメディア（USBフラッシュドライブやCD-ROMなど）を挿入し、そこから起動してください。

---

### チップ

サーバーのファームウェア設定で、インストールメディア(USBなど)からのブートが有効になっていることを確認してください。Proxmox VE バージョン 8.1 より前のインストーラを起動する場合は、セキュアブートを無効にする必要があります。

---



正しい項目（例えば、*Boot from USB*）を選択すると、Proxmox VEメニューが表示され、以下のオプションのいずれかを選択できます：

#### Proxmox VE (グラフィカル)のインストール

通常のインストールを開始します。

---

#### チップ

キーボードのみでインストールウィザードを使用することができます。ボタンは、ALT キーと各ボタンの下線文字を組み合わせて押すことでクリックできます。例えば、ALT + NでNextボタンを押します。

---

#### Proxmox VE (ターミナルUI) のインストール

ターミナルモードのインストールウィザードを起動します。グラフィカルインストーラと同じ全体的

なインストールエクスペリエンスを提供しますが、非常に古いハードウェアや非常に新しいハードウェアとの互換性が一般的に優れています。

### Proxmox VEのインストール（ターミナルUI、シリアルコンソール）

ターミナル・モードのインストール・ウィザードを起動し、さらにLinuxカーネルがマシンの（最初の）シリアル・ポートを入出力に使用するように設定する。これは、マシンが完全にヘッドレスで、シリアルコンソールしか利用できない場合に使用できる。



どちらのモードも、実際のインストールプロセスには同じコードベースを使用し、10年以上にわたるバグ修正の恩恵を受け、機能の同等性を確保しています。

### チップ

ターミナル UI オプションは、ドライバの問題などでグラフィカルインストーラが正しく動作しない場合に使用できます。[nomodeset kernel パラメータ](#)の追加も参照してください。

## 高度なオプションProxmox VEのインストール（グラフィカル、デバッグモード）

デバッグモードでインストールを開始します。いくつかのインストールステップでコンソールが開きます。これは、何か問題が発生した場合のデバッグに役立ちます。デバッグコンソールを終了するには、**CTRL-D** を押してください。このオプションを使用すると、基本的なツールがすべて使用可能なライブシステムを起動できます。たとえば、[デグレードした ZFS rpool を修復](#)したり、既存の Proxmox VE セットアップのブートローダを修正したりする場合に使用できます。

## 高度なオプションProxmox VEのインストール（ターミナルUI、デバッグモード）

グラフィカルデバッグモードと同じだが、代わりにターミナルベースのインストーラを実行するようシステムを準備する。

## 高度なオプションProxmox VE（シリアルコンソールデバッグモード）のインストール

端末ベースのデバッグモードと同じだが、さらにLinuxカーネルがマシンの（最初の）シリアルポートを入出力に使うように設定する。

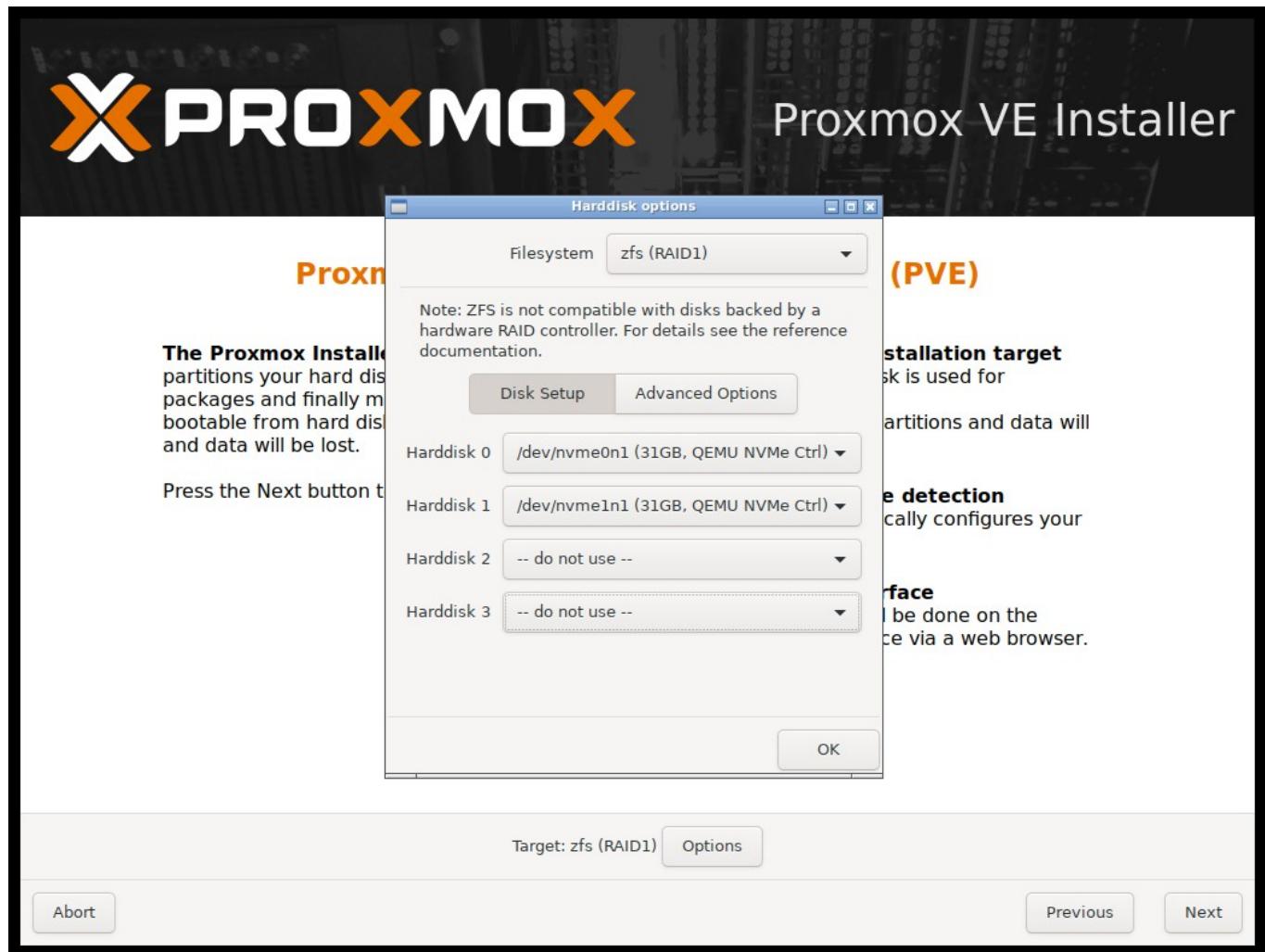
## 高度なオプションレスキューブート

このオプションを使用すると、既存のインストールをブートできます。接続されているすべてのハードディスクを検索します。既存のインストールが見つかると、ISO から Linux カーネルを使ってそのディスクに直接ブートします。これは、ブートローダ (GRUB/systemd-boot) に問題があったり、BIOS/UEFI がディスクからブートブロックを読み取れない場合に便利です。

## 高度なオプションメモリのテスト (memtest86+)

memtest86+を実行する。メモリが機能しているか、エラーがないかをチェックするのに便利です。このオプションを実行するには、UEFI ファームウェアセットアップユーティリティで Secure Boot をオフにする必要があります。

通常、Install Proxmox VE (Graphical) を選択してインストールを開始します。



最初のステップは、EULA（エンドユーザー ライセンス契約）を読むことです。続いて、インストール先のハードディスクを選択します。

### 注意

デフォルトでは、サーバー全体が使用され、既存のデータはすべて削除されます。インストールを続行する前に、サーバー上に重要なデータがないことを確認してください。

オプション] ボタンで、ターゲットファイルシステムを選択できます。デフォルトは ext4 です。ファイ

ルシステムとして ext4 または xfs を選択した場合、インストーラは LVM を使用し、LVM 領域を制限する追加オプションを提供します ([下記参照](#))。

Proxmox VEはZFSにもインストールできます。ZFSはいくつかのソフトウェアRAIDレベルを提供しているため、ハードウェアRAIDコントローラを搭載していないシステム向けのオプションです。オプションダイアログでターゲットディスクを選択する必要があります。ZFS固有の設定は[Advanced Options](#)で変更できます。

-



## 警告

ハードウェアRAID上のZFSはサポートされておらず、データ損失の原因となります。

The Proxmox Installer automatically makes location based optimizations, like choosing the nearest mirror to download files. Also make sure to select the right time zone and keyboard layout.

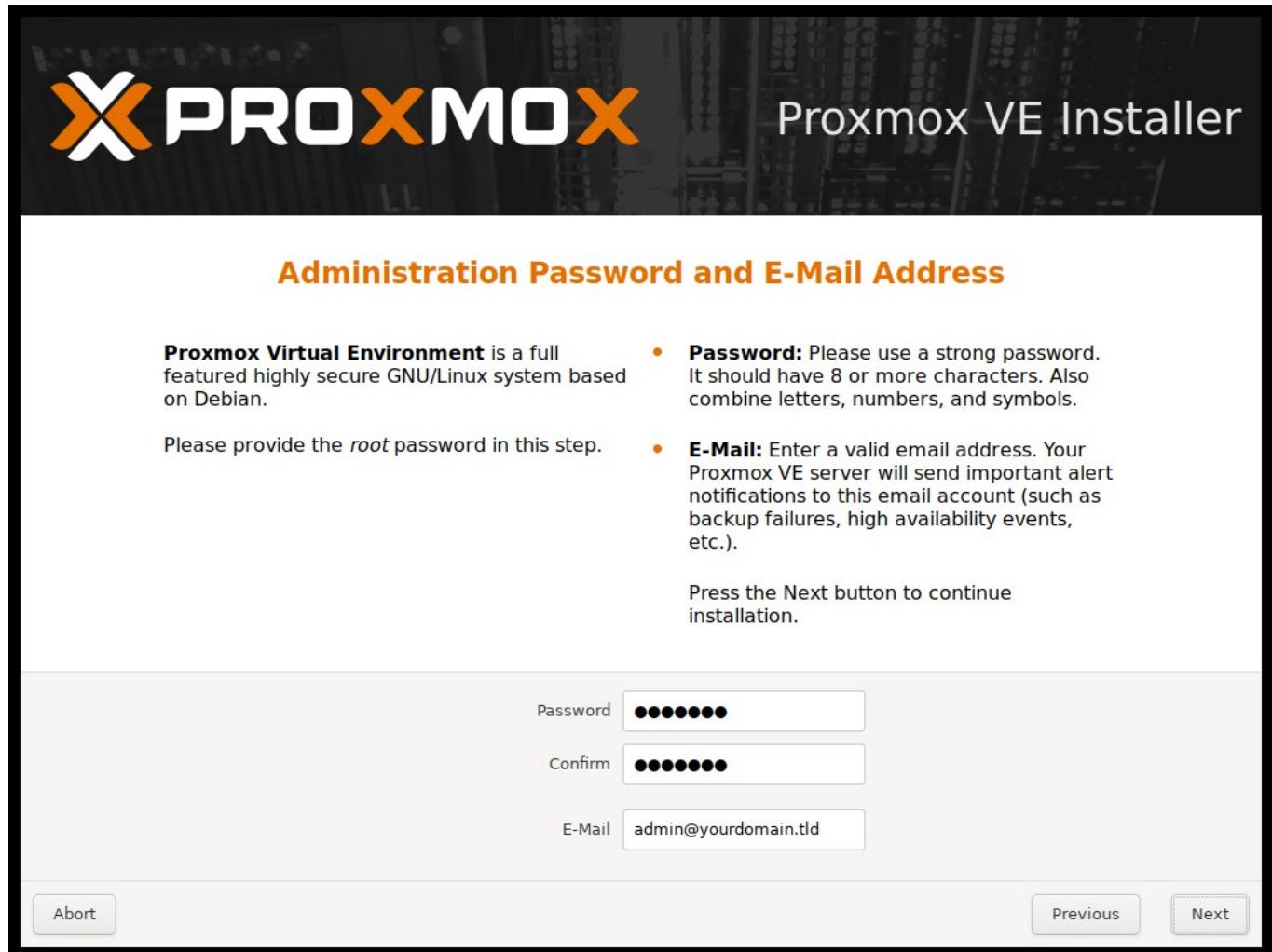
Press the Next button to continue installation.

- Country:** The selected country is used to choose nearby mirror servers. This will speedup downloads and make updates more reliable.
- Time Zone:** Automatically adjust daylight saving time.
- Keyboard Layout:** Choose your keyboard layout.

Country	Austria
Time zone	Europe/Vienna
Keyboard Layout	German

Abort      Previous      Next

次のページでは、お住まいの場所、タイムゾーン、キーボードレイアウトなどの基本的な設定オプションを尋ねられます。場所は、アップデートの速度を上げるために、近くのダウンロードサーバーを選択するために使用されます。インストーラは通常、これらの設定を自動検出できるので、自動検出に失敗した場合や、あなたの国で一般的に使用されていないキーボードレイアウトを使用したい場合にのみ、稀に設定を変更する必要があります。

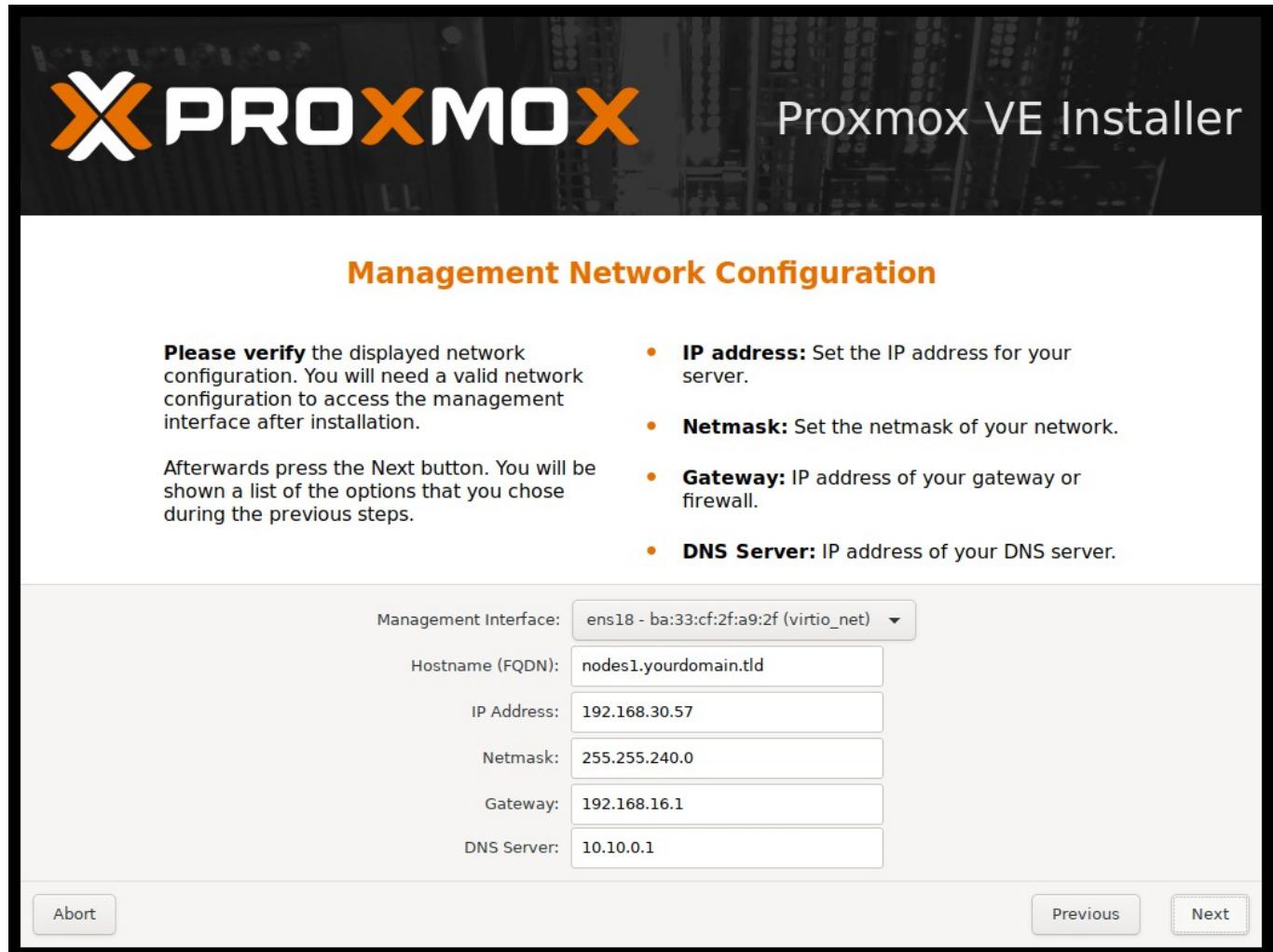


次に、スーパーユーザー（root）のパスワードとメールアドレスを指定する必要がある。パスワードは少なくとも5文字で構成されていなければならない。より強力なパスワードを使用することを強く推奨する。いくつかのガイドラインがあります：

- ・ パスワードの長さは最低12文字以上にしてください。
- ・ 小文字、大文字のアルファベット、数字、記号を含む。
- ・ 文字の繰り返し、キーボードのパターン、一般的な辞書の単語、文字や数字の並び、ユーザー名、親戚やペットの名前、恋愛関係（現在または過去）、伝記的な情報（ID番号、先祖の名前や日付など）は避けてください。

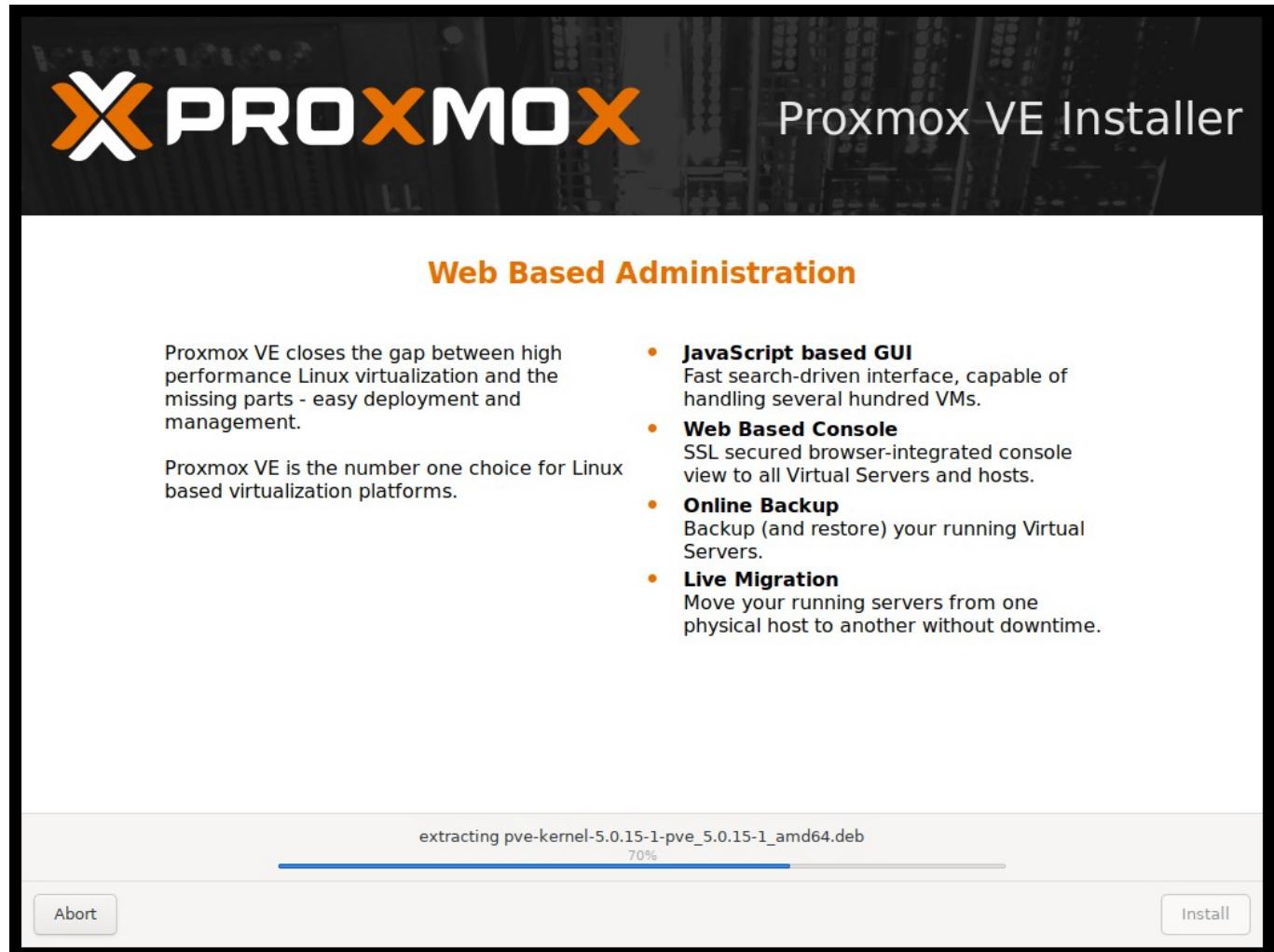
電子メールアドレスは、システム管理者に通知を送信するために使用されます。例えば

- ・ 利用可能なパッケージのアップデートに関する情報。
- ・ 定期的なcronジョブのエラーメッセージ。



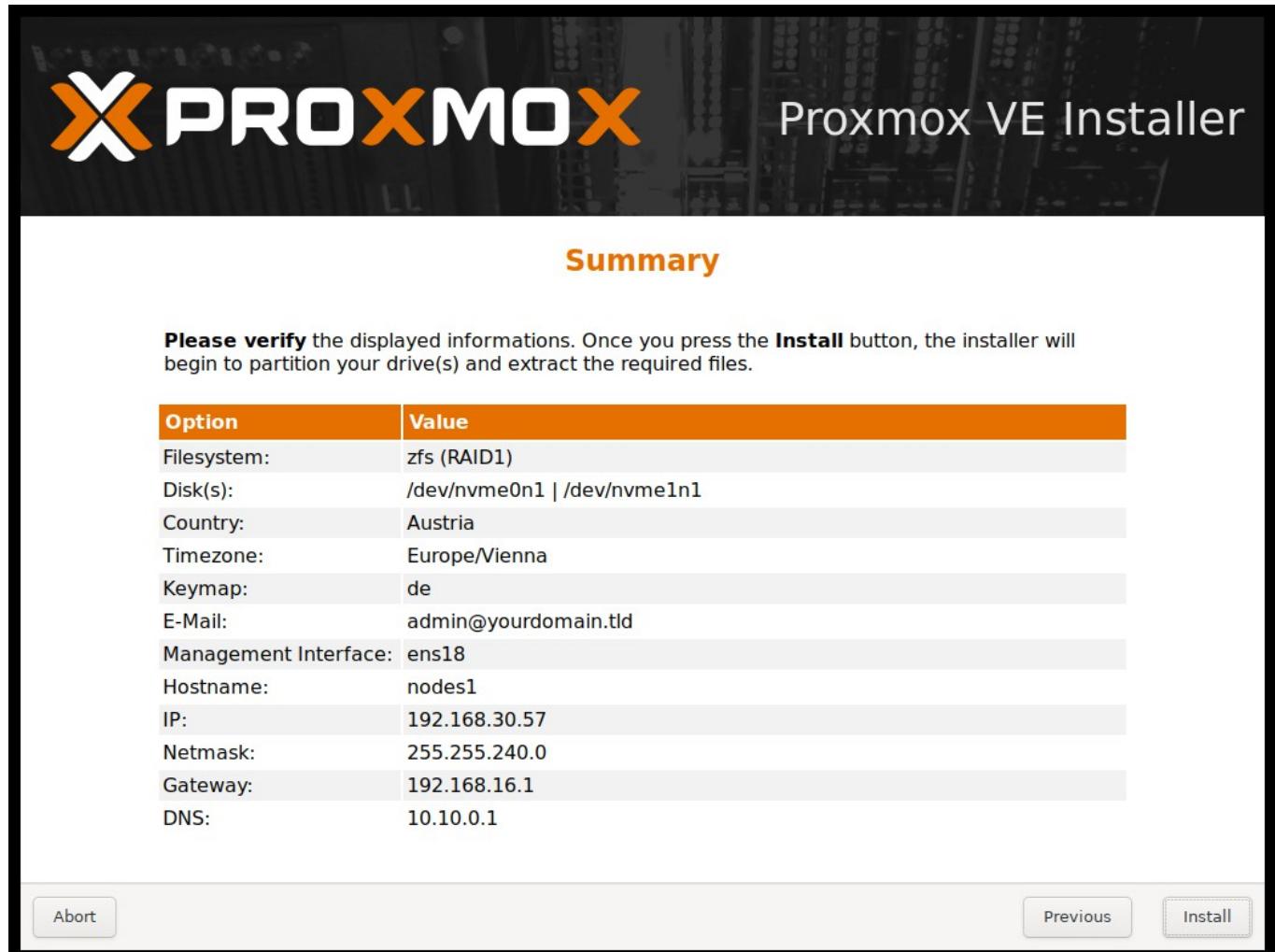
これらの通知メールはすべて、指定されたEメールアドレスに送信される。

最後のステップはネットワーク設定である。UPされているネットワークインターフェイスは、ドロップダウンメニューの名前の前に塗りつぶされた円が表示されます。インストール中はIPv4アドレスかIPv6アドレスのどちらかを指定できますが、両方を指定することはできません。デュアルスタックノードを設定するには、インストール後にIPアドレスを追加してください。



次のステップでは、以前に選択したオプションの概要が表示されます。すべての設定を再確認し、設定を変更する必要がある場合は「前へ」ボタンを使用してください。

インストールをクリックすると、インストーラーがディスクのフォーマットとターゲットディスクへのパッケージのコピーを開始します。このステップが終了するまで待ってから、インストールメディアを取り出し、システムを再起動してください。



パッケージのコピーには通常数分かかりますが、そのほとんどはインストールメディアの速度とターゲットディスクの性能に依存します。

パッケージのコピーとセットアップが終了したら、サーバーを再起動します。デフォルトでは数秒後に自動的に行われます。

### インストールの失敗

インストールに失敗した場合は、2台目のTTY (**CTRL + ALT + F2**) で特定のエラーをチェックし、システムが[最小要件を満たしている](#)ことを確認します。

それでもインストールがうまくいかない場合は、「[ヘルプの入手方法](#)」の章をご覧ください。

### 2.3.1 インストール後の管理インターフェイスへのアクセス

Proxmox VE Login

User name:	root
Password:	*****
Realm:	Linux PAM standard authentication
Language:	English

Save User name:

システムのインストールと再起動が完了したら、Proxmox VE ウェブインターフェースを使用して、さらに設定を行うことができます。

1. ブラウザを、インストール時に指定されたIPアドレスとポート8006（例：<https://youripad>）に合わせます。
2. `root` (realm PAM)ユーザー名と、インストール時に選択したパスワードを使用してログインする。
3. Enterprise リポジトリにアクセスするには、サブスクリプションキーをアップロードしてください。そうでない場合は、セキュリティフィックス、バグフィックス、新機能のアップデートを取得するために、あまりテストされていないパブリックパッケージリポジトリのいずれかをセットアップする必要があります。
4. IPコンフィグレーションとホスト名を確認する。
5. タイムゾーンを確認してください。
6. ファイアウォールの設定を確認してください。

### 2.3.2 高度なLVM構成オプション

インストーラーは、`pve` というボリュームグループ (VG) と、`ext4` または `xfs` を使用している場合は `root`、`data`、`swap` という追加の論理ボリューム (LV) を作成します。これらのボリュームのサイズを制御するには

#### ハードサイズ

使用するハードディスクの合計サイズを定義します。こうすることで、ハードディスク上に空き領域を確保し、さらにパーティション分割を行うことができます（例えば、同じハードディスク上にPVとVGを追加し、LVMストレージとして使用することができます）。

#### スワップサイズ

スワップボリュームのサイズを定義します。デフォルトはインストールされているメモリのサイズで、最小4GB、最大8GBです。結果の値は `hdsize/8` より大きくすることはできません。

---

#### 注

0に設定すると、スワップ・ボリュームは作成されない。

---

## マックスルート

オペレーション・システムを格納するルート・ボリュームの最大サイズを定義します。最大ルート・ボリューム・サイズの上限は`hdszie/4`です。

## マックスブズ

データボリュームの最大サイズを定義します。データボリュームの実際のサイズは

`datasize = hdszie - rootszie - swapsize - minfree`

ここで、`datasize`は`maxvz`より大きくできない。

---

### 注

LVM thinの場合、データプールはデータサイズが4GBより大きい場合のみ作成される。

---

---

## 注

0に設定すると、データボリュームは作成されず、ストレージ構成はそれに応じて適応される。

---

## ミニフリー

LVMボリューム・グループpveに残すべき空き容量を定義します。を超える128GBのストレージが使用可能で、デフォルトは16GB、それ以外はhdszie/8が使用される。

---

## 注

LVMでは、スナップショット作成にVG内の空き領域が必要です (lvmthinスナップショットでは必要ありません)。

---

### 2.3.3 高度なZFS設定オプション

ZFSを使用する場合、インストーラは ZFS プール rpool を作成します。swap領域は作成されませんが、インストールディスク上にswap用の未パーティション領域を確保できます。インストール後に swap zvol を作成することもできますが、これは問題につながる可能性があります ([ZFS swap notes を参照](#))。

#### アッシュシフト

作成されるプールのashift値を定義する。ashift は少なくとも sector- に設定する必要があります。ディスクのサイズ (2のashift乗がセクタサイズ)、またはプールに入れられる可能性のあるディスク (欠陥ディスクの交換など)。

#### コンプレス

rpoolで圧縮を有効にするかどうかを定義する。

#### チェックサム

rpoolにどのチェックサムアルゴリズムを使用するかを定義する。

#### コピー

rpool の copies パラメータを定義する。その意味と、なぜこれがディスクレベルの冗長性を置き換えないのかについては、[zfs\(8\)](#) のマニュアルページを確認してください。

#### ARC最大サイズ

ARCが成長できる最大サイズを定義し、ZFSが使用するメモリ量を制限します。詳細については、[ZFSのメモリ使用量を制限する方法](#)のセクションも参照してください。

---

## ハードサイズ

使用するハードディスクの総サイズを定義します。`hdszie` はブート可能なディスク、つまり RAID0、RAID1、RAID10 の最初のディスクかミラー、RAID-Z[123]の全てのディスクに対してのみ有効です。

### 2.3.4 ZFSパフォーマンスのヒント

ZFSは大容量のメモリで最もよく機能します。ZFSを使用する場合は、十分なRAMを用意してください。TBのRAWディスク容量ごとに4GBと1GBのRAMを追加するのが良い計算です。

ZFSは、ZFS Intent Log (ZIL) と呼ばれる専用ドライブを書き込みキャッシュとして使用できます。高速ド

```
# zpool add <pool-name> log </dev/path_to_fast_ss>.
```

ライブ (SSD) を使用してください。インストール後に以下のコマンドで追加できます：

### 2.3.5 nomodesetカーネル・パラメーターの追加

グラフィックドライバが原因で、非常に古いハードウェアや非常に新しいハードウェアで問題が発生することがあります。ブート中にインストールがハングする場合は、nomodeset パラメータを追加してみてください。これは Linux カーネルがグラフィックドライバをロードしないようにし、BIOS/UEFI が提供するフレームバッファを使い続けるようにします。

Proxmox VE ブートローダのメニューで、*Install Proxmox VE (Terminal UI)*に移動し、e キーを押してエントリを編集します。矢印キーを使用して linux で始まる行に移動し、カーソルをその行の最後に移動して、パラメータ nomodeset を既存の最後のパラメータからスペースで区切って追加します。

次に Ctrl-X または F10 を押して設定をブートする。

## 2.4 無人設置

Proxmox VEを無人で自動インストールすることができます。これにより、ベアメタル上のセットアッププロセスを完全に自動化できます。インストールが完了し、ホストが起動したら、Ansible のような自動化ツールを使用してインストールをさらに設定できます。

インストーラに必要なオプションは、アンサーファイルで提供されなければならない。このファイルでは、フィルタルールを使用して、使用するディスクとネットワークカードを決定することができます。

自動インストールを使用するには、まずインストール ISO を準備する必要があります。無人インストールの詳細と情報については、[ウィキをご覧ください](#)。

## 2.5 DebianにProxmox VEをインストールする

Proxmox VEはDebianパッケージのセットとして出荷され、標準的なDebianインストレーションの上にイン

```
# apt-get update  
# apt-get install proxmox-ve
```

ストールすることができます。[リポジトリを設定した後](#)、以下のコマンドを実行する必要があります：

既存の Debian インストールの上にインストールするのは簡単そうに見えますが、基本システムが正しくインストールされており、ローカルストレージをどのように設定し、使用したいかがわかっていることが前提です。また、ネットワークを手動で設定する必要があります。

一般的に、特にLVMやZFSを使用する場合、これは些細なことで

はない。詳細なステップバイステップのハウツーは[wiki](#)にあります

す。

## 第3章

# ホストシステム管理

以下のセクションでは、一般的な仮想化タスクに焦点を当て、ホストマシンの管理および運用に関する Proxmox VE の仕様について説明します。

Proxmox VE は [Debian GNU/Linux](#) をベースにしており、Proxmox VE 関連のパッケージを提供するためのリポジトリが追加されています。つまり、セキュリティアップデートやバグ修正を含む Debian パッケージの全範囲が利用可能です。Proxmox VE は、Ubuntu カーネルをベースにした独自の Linux カーネルを提供します。必要な仮想化機能とコンテナ機能がすべて有効になっており、ZFS といくつかの追加ハードウェアドライバが含まれています。

以下の節に含まれていないその他の話題については、Debian の文書を参照してください。[Debian 管理者ハンドブック \(Debian Administrator's Handbook\)](#) はオンラインで入手可能で、Debian オペレーティングシステムの包括的な入門書を提供しています ([\[Hertzog13\]](#) をご覧ください)。

### 3.1 パッケージ・リポジトリ

Proxmox VE は、他の Debian ベースのシステムと同様に、パッケージ管理ツールとして [APT](#) を使用します。

Proxmox VE はパッケージのアップデートを毎日自動的にチェックします。root@pam ユーザーには、利用可能なアップデートが電子メールで通知されます。GUI から *Changelog* ボタンを使用すると、選択したアップデートの詳細を見ることができます。

#### 3.1.1 Proxmox VE のリポジトリ

リポジトリはソフトウェア・パッケージの集合体であり、新しいソフトウェアをインストールするために使われるだけでなく、新しいアップデート入手するためにも重要である。

---

### 注

最新のセキュリティ更新、バグ修正、新機能入手するには、有効な Debian と Proxmox のリポジトリが必要です。

---

APTリポジトリは、/etc/apt/sources.listファイルと、/etc/apt/に置かれた.listファイルで定義される。

## リポジトリ管理

Enabled	Types	URIs	Suites	Components	Options	Origin	Comment
<input type="checkbox"/>	deb	http://deb.debian.org/debian/	bullseye	main contrib non-free			Debian
<input checked="" type="checkbox"/>	deb	http://security.debian.org/debian-security	bullseye-security	main contrib non-free			Debian
<input type="checkbox"/>	deb	https://enterprise.proxmox.com/debian/pve	bullseye	pve-enterprise			Proxmox

Proxmox VE 7 以降、Web インターフェースでリポジトリの状態を確認できます。ノードサマリーパネルは高レベルのステータス概要を表示し、独立したリポジトリパネルは詳細なステータスと設定されたすべてのリポジトリのリストを表示します。

基本的なリポジトリ管理、たとえばリポジトリのアクティブ化や非アクティブ化もサポートされている。

## 情報源リスト

`sources.list` ファイルでは、各行がパッケージ・リポジトリを定義する。優先するソースが最初に来なければならない。空行は無視されます。行のどこかに`#`を付けると、その行の残りはコメントとして扱われます。リポジトリから利用可能なパッケージは、`apt-get update`を実行して取得します。アップデートは `apt-get` を使って直接インストールするか、GUI (Node → Updates) を使ってインストールしてください。

### ファイル `/etc/apt/sources.list`

```
deb http://deb.debian.org/debian bookworm main contrib
deb http://deb.debian.org/debian bookworm-updates main contrib

# セキュリティ・アップデート
deb http://security.debian.org/debian-security bookworm-security main ←
    コントリビューション
```

Proxmox VEは3つの異なるパッケージリポジトリを提供します。

### 3.1.2 Proxmox VE エンタープライズリポジトリ

これは推奨リポジトリで、すべての Proxmox VE サブスクリプションユーザーが利用できます。最も安定したパッケージが含まれており、本番環境での使用に適しています。pve-enterprise リポジトリはデフォルトで有効になっています：

#### ファイル /etc/apt/sources.list.d/pve-enterprise.list

```
deb https://enterprise.proxmox.com/debian/pve 本の虫 pve-enterprise
```

pve-enterprise リポジトリにアクセスするには、有効なサブスクリプションキーが必要です。<https://proxmox.com/en/proxmox-virtual-environment/pricing> で詳細をご覧いただけます。

---

#### 注

上記の行を#（行頭）でコメントアウトすることで、このリポジトリを無効にすることができます。これにより、ホストにサブスクリプションキーがない場合のエラーメッセージを防ぐことができます。その場合は pve-no-subscription リポジトリを設定してください。

---

### 3.1.3 Proxmox VE ノーサブスクリプションリポジトリ

名前が示すように、このリポジトリにアクセスするためにサブスクリプションキーは必要ありません。テストや本番環境以外での使用に使用できます。本番サーバーで使用することはお勧めしません。

このリポジトリは /etc/apt/sources.list で設定することを推奨する。

#### ファイル /etc/apt/sources.list

```
deb http://ftp.debian.org/debian bookworm main contrib  
deb http://ftp.debian.org/debian bookworm-updates main contrib
```

```
# Proxmox VE pve-no-subscription リポジトリは proxmox.com によって提供され  
ています。 # 本番環境での使用は推奨されていません。
```

```
deb http://download.proxmox.com/debian/pve ブックワーム pve-no-サブスクリプション
```

```
# セキュリティ・アップデート
```

```
deb http://security.debian.org/debian-security bookworm-security main ←  
コントリビューション
```

### 3.1.4 Proxmox VE テストリポジトリ

このリポジトリには最新のパッケージが含まれており、主に開発者が新機能をテストするために使用します。設定するには、`/etc/apt/sources.list` に以下の行を追加します：

### pvetestのsources.listエントリ

```
deb http://download.proxmox.com/debian/pve bookworm pvetest
```



#### 警告

pvetestリポジトリは（その名の通り）新機能やバグ修正のテストにのみ使うべきです。

### 3.1.5 Ceph Reefエンタープライズリポジトリ

このリポジトリには、エンタープライズ向けProxmox VE Ceph 18.2 Reefパッケージが格納されています。本番環境に適しています。Proxmox VEでCephクライアントまたは完全なCephクラスタを実行する場合は、このリポジトリを使用します。

#### ファイル /etc/apt/sources.list.d/ceph.list

```
deb https://enterprise.proxmox.com/debian/ceph-reef ブックワーム・エンタープライズ
```

### 3.1.6 Ceph Reefサブスクリプションなしリポジトリ

このCephリポジトリには、エンタープライズリポジトリに移動する前のCeph 18.2 Reefパッケージと、テストリポジトリに移動した後のCeph 18.2 Reefパッケージが格納されています。

#### 注

本番マシンにはエンタープライズリポジトリを使用することを推奨する。

#### ファイル /etc/apt/sources.list.d/ceph.list

```
deb http://download.proxmox.com/debian/ceph-reef ブックワーム ノーサブスクリプション
```

### 3.1.7 Ceph Reefテストリポジトリ

このCephリポジトリには、メインリポジトリに移動する前のCeph 18.2 Reefパッケージが格納されています

。Proxmox VEでの新しいCephリリースのテストに使用されます。

#### ファイル /etc/apt/sources.list.d/ceph.list

```
deb http://download.proxmox.com/debian/ceph-reef 読書家テスト
```

### 3.1.8 Ceph Quincy Enterpriseリポジトリ

このリポジトリには、エンタープライズ向けProxmox VE Ceph Quincyパッケージが格納されています。本番環境に適しています。Proxmox VEでCephクライアントまたは完全なCephクラスタを実行する場合は、このリポジトリを使用します。

#### ファイル /etc/apt/sources.list.d/ceph.list

```
deb https://enterprise.proxmox.com/debian/ceph-quincy ブックワーム・エンタープライズ
```

### 3.1.9 Ceph Quincyサブスクリプションなしリポジトリ

このCephリポジトリには、エンタープライズリポジトリに移動する前のCeph Quincyパッケージと、テストリポジトリに移動した後のCeph Quincyパッケージが格納されます。

---

#### 注

本番マシンにはエンタープライズリポジトリを使用することを推奨する。

---

#### ファイル /etc/apt/sources.list.d/ceph.list

```
deb http://download.proxmox.com/debian/ceph-quincy ブックワーム ノーサブスクリプション
```

### 3.1.10 Ceph Quincyテストリポジトリ

このCephリポジトリには、メインリポジトリに移動する前のCeph Quincyパッケージが格納されています。Proxmox VEでの新しいCephリリースのテストに使用されます。

#### ファイル /etc/apt/sources.list.d/ceph.list

```
deb http://download.proxmox.com/debian/ceph-quincy 読書家テスト
```

### 3.1.11 古いCephリポジトリ

Proxmox VE 8は、Ceph Pacific、Ceph Octopus、またはハイパーコンバージドセットアップ用の古いリリースをサポートしていません。これらのリリースでは、Proxmox VE 8にアップグレードする前に、まずCephを新しいリリースにアップグレードする必要があります。

---

詳しくは各アップグレードガイドをご覧ください。

### 3.1.12 Debian ファームウェアリポジトリ

Debian Bookworm (Proxmox VE 8) から、(DFSG で定義された) non-free ファームウェアは新しく作成された Debian リポジトリの non-free-firmware コンポーネントに移動されました。

早期OSマイクロコードアップデートをセットアップする場合や、プリインストールパッケージpve-firmwareに含まれていないランタイムファームウェアファイルが必要な場合は、このリポジトリを有効にしてください。

このコンポーネントからパッケージをインストールできるようにするには、/etc/apt/sources.list を実行し、次のように追加します。

non-free-firmware を各 .debian.org リポジトリの行末に追加し、apt update を実行してください。

### 3.1.13 セキュアアパート

リポジトリ内の *Release* ファイルは GnuPG で署名されています。APTはこれらの署名を使用して、すべてのパッケージが信頼できるソースからのものであることを確認しています。

Proxmox VEを公式ISOイメージからインストールする場合、検証用のキーはすでにインストールされています。

Debianの上にProxmox VEをインストールする場合は、以下のコマンドでキーをダウンロードしてインストールしてください:

```
# wget https://enterprise.proxmox.com/debian/proxmox-release-bookworm.gpg -O /etc/apt/trusted.gpg.d/proxmox-release-bookworm.gpg
```

その後、`sha512sum` CLIツールでチェックサムを検証する:

```
# sha512sum /etc/apt/trusted.gpg.d/proxmox-release-bookworm.gpg  
7 ←  
da6fe34168adc6e479327ba517796d4702fa2f8b4f0a9833f5ea6e6b48f6507a6da403a274fe201  
/etc/apt/trusted.gpg.d/proxmox-release-bookworm.gpg
```

または`md5sum` CLIツール:

```
# md5sum /etc/apt/trusted.gpg.d/proxmox-release-bookworm.gpg  
41558dc019ef90bd0f6067644a51cf5b /etc/apt/trusted.gpg.d/proxmox-release- ←'  
bookworm.gpg
```

## 3.2 システムソフトウェアのアップデート

Proxmoxはすべてのリポジトリに対して定期的にアップデートを提供します。アップデートをインストール

```
# apt-get update  
# apt-get dist-upgrade
```

するには、WebベースのGUIまたは以下のCLIコマンドを使用します:

---

#### 注

APTパッケージ管理システムは非常に柔軟で、多くの機能を提供しています。詳細については、`man apt-get`や[\[Hertzog13\]](#)を参照してください。

---

#### チップ

定期的なアップデートは、最新のパッチとセキュリティ関連の修正を取得するために不可欠です。主要な

システムアップグレードはProxmox VEコミュニティフォーラムで発表されます。

---

### 3.3 ファームウェア・アップデート

本章のファームウェアアップデートは、ベアメタルサーバ上で Proxmox VE を実行する場合に適用してください。デバイス・パススルーを使用する場合など、ゲスト内でファームウェア更新を構成することが適切かどうかは、セットアップに大きく依存するため、ここでは触れません。

ソフトウェアの定期的なアップデートに加え、ファームウェアのアップデートも信頼性の高い安全な運用のために重要です。

ファームウェア・アップデートを入手し、適用する場合は、利用可能なオプションを組み合わせて、可能な限り早期に、またはまったく入手することを推奨する。

ファームウェアという用語は通常、言語的にはマイクロコード（CPU用）とファームウェア（その他のデバイス用）に分けられる。

### 3.3.1 永続的ファームウェア

このセクションはすべてのデバイスに適しています。通常 BIOS/UEFI アップデートに含まれる更新されたマイクロコードはマザーボードに保存され、その他のファームウェアはそれぞれのデバイスに保存されます。この永続的な方法は、ブート時に更新されたマイクロコードを可能な限り早期に定期的にロードすることを可能にするため、CPUにとって特に重要です。



#### 注意

BIOS/UEFIやストレージコントローラなどのアップデートにより、デバイスの設定がリセットされることがあります。ベンダーの指示に注意深く従い、現在の設定をバックアップしてください。

どの更新方法が利用可能か、ベンダーに確認してください。

- サーバーの便利な更新方法には、デルのライフサイクル・マネージャーやHPEのサービスパックがある。
- 例えは、NVIDIA ConnectX用のmlxupや、以下のような Linuxユーティリティがあります。  
*bnxtnvm/niccli* Broadcom ネットワークカード用。
- ハードウェアベンダーとの協力があり、サポートされているハードウェアが使用されている場合は、LVFS も選択肢となる。このための技術的条件は、システムが2014年以降に製造され、UEFI経由で起動することです。

Proxmox VE は、Proxmox 署名キーによるセキュアブートサポートを有効にするため、独自のバージョンの fwupd パッケージを出荷しています。このパッケージは、ハイパーバイザー上の `udisks2` package の使用に関する問題が確認されているため、依存関係の推奨を意識的に取りやめています。これは、例

えば /etc/fwupd/daemon.conf で EFI パーティションの正しいマウントポイントを明示的に設定しなければならないことを意味します:

#### ファイル /etc/fwupd/daemon.conf

```
# EFIシステムパーティション(ESP)のパスに使用される場所を上書きします。  
EspLocation=/boot/efi
```

---

#### チップ

更新の指示にホストの再起動が必要な場合は、安全に実行できることを確認してください。 [ノードのメンテナンスも参照してください。](#)

---

### 3.3.2 ランタイム・ファームウェア・ファイル

このメソッドは、Proxmox VE オペレーティングシステムにファームウェアを保存し、[パーシステッド ファームウェアが新しいものでない場合](#)、そのファームウェアをデバイスに渡す。ネットワークカードやグラフィックカードなどのデバイスでサポートされていますが、マザーボードやハードディスクなどのパーシステッドファームウェアに依存しているデバイスではサポートされていません。

Proxmox VEでは、`pve-firmware`パッケージがデフォルトでインストールされています。そのため、通常の[システムアップデート\(APT\)](#)により、一般的なハードウェアのファームウェアは自動的に最新の状態に保たれます。

追加の[Debian ファームウェアアリポジトリ](#)は存在しますが、デフォルトでは設定されていません。

追加ファームウェアパッケージをインストールしようとして競合した場合、APTはインストールを中断します。もしかしたら、そのファームウェアは別 の方法で入手できるかもしれません。

### 3.3.3 CPUマイクロコードアップデート

マイクロコードのアップデートは、発見されたセキュリティ脆弱性やその他の深刻なCPUバグを修正することを目的としている。CPUの性能は影響を受ける可能性がありますが、パッチを適用したマイクロコードは、通常、カーネル自身が緩和策を講じなければならないパッチを適用していないマイクロコードよりも、依然として高い性能を発揮します。CPUの種類によっては、故意に安全でない状態でCPUを動作させなければ、欠陥のある工場出荷状態の性能結果を達成できなくなる可能性がある。

現在のCPUの脆弱性とその緩和策の概要を知るには、`lscpu`を実行してください。Proxmox VEホストが[最新](#)であり、そのバージョンが[終了しておらず](#)、少なくとも最後のカーネル更新以降に再起動されている場合にのみ、現在の現実の既知の脆弱性が表示されます。

[永続的な BIOS/UEFI アップデート](#)による推奨マイクロコードアップデート以外に、[Early OS Microcode Updates](#)による独立した方法もあります。これは便利で、マザーボードベンダーがBIOS/UEFIアップデートを提供しなくなった場合にも役立ちます。いずれの方法を使用する場合でも、マイクロコードアップデートを適用するには必ず再起動が必要です。

#### 早期OSマイクロコード・アップデートの設定

Linuxカーネルによってブート時に早期に適用されるマイクロコード・アップデートを設定するには、次のようにする必要がある：

1. [Debianファームウェアリポジトリを有効にする](#)
2. 最新のパッケージを入手する `apt update` (またはウェブインターフェイスの Node → Updates を使う)
3. CPUベンダー固有のマイクロコードパッケージをインストールする：
  - インテルCPUの場合: `apt install intel-microcode`
  - AMD CPUの場合: `apt install amd64-microcode`
4. Proxmox VEホストを再起動します。

今後のマイクロコードアップデートも、ロードするために再起動が必要となる。

## マイクロコードバージョン

比較やデバッグの目的で、現在実行中のマイクロコードリビジョンを取得する：

```
# grep microcode /proc/cpuinfo |  
uniq microcode : 0xf0
```

マイクロコードパッケージには、さまざまなCPU用のアップデートがある。しかし、あなたのCPUに特化したアップデートはあまりないかもしれません。そのため、パッケージの日付を見ただけでは、その会社が実際にあなたの特定のCPU用のアップデートをいつリリースしたのかはわかりません。

新しいマイクロコードパッケージをインストールし、Proxmox VEホストを再起動した場合、この新しいマイ

```
# dmesg | grep マイクロコ  
[ド 0.000000] microcode: マイクロコードはリビジョン0xf0に早期更新、日付 = 2021- ←  
[ 11-12 0.896580] マイクロコード: Microcode Update Driver: v2.2。
```

クロコードがCPUに組み込まれたバージョンとマザーボードのファームウェアのバージョンの両方よりも新しい場合、システムログに "microcode updated early" というメッセージが表示されます。

## トラブルシューティング

デバッグの目的で、システム起動時に定期的に適用されるEarly OS Microcode Updateの設定を、以下のように一時的に無効にすることができます：

1. ホストが[安全にリブート](#)できることを確認する。
2. ホストを再起動してGRUBメニューを表示する（非表示の場合はSHIFTを押したままにする）
3. を押します。
4. linuxで始まる行に行き、スペースで区切って追加する **dis\_ucode\_ldr**
5. CTRL-Xキーを押して、Early OS Microcode Updateなしで起動します。

最近のマイクロコードのアップデートに関連した問題が疑われる場合、パッケージの削除 (`apt purge <intel-microcode|amd64-microcode>`) ではなく、パッケージのダウングレードを考慮すべきです。そうしないと、最新のマイクロコードなら問題なく実行できるにもかかわらず、古すぎる[永続化された](#)マイクロコードがロードされてしまうかもしれません。

この例で示すように、以前のマイクロコードパッケージのバージョンが Debian リポジトリで利用可能である。

```
# apt list -a intel-microcode リスト
...完了

intel-microcode/stable-security, now 3.20230808.1~deb12u1 amd64 [インストー
# apt install intel-microcode=3.202305*
...
intel-<--> にバージョン「3.20230512.1」 (Debian:12.1/stable [amd64]) を選択。
    マイクロコード
...
dpkg: 警告: intel-microcode を 3.20230808.1~deb12u1 から <'にダウングレードしています。
      3.20230512.1
```

れば、ダウングレードは可能です：

...

intel-microcode: マイクロコードは次のブート時に更新されます。

...

ホストが[安全に](#)リブートできることを（もう一度）確認してください。お使いのCPUタイプ用のマイクロコードパッケージに潜在的に含まれている古いマイクロコードを適用するには、今すぐ再起動してください。

## チップ

ダウングレードしたパッケージをしばらくの間保持し、後日より新しいバージョンを試してみることは理

```
# apt-mark hold intel-
microcode intel-microcodeを保留
# apt-mark unhold intel-microcode
# apt update
# apt upgrade
```

にかなっています。将来パッケージのバージョンが同じになったとしても、その間にシステムアップデートによって経験した問題が修正されているかもしれません。

## 3.4 ネットワーク構成

Proxmox VEはLinuxネットワークスタックを使用しています。このため、Proxmox VEノードのネットワーク設定方法に柔軟性があります。設定は、GUI または手動でファイル /etc/network/interfaces はネットワーク設定全体を含んでいます。interfaces (5) マニュアルページに、完全なフォーマットの説明があります。すべての Proxmox VE ツールは、ユーザが直接修正できるように努力していますが、GUI を使用する方がエラーから保護されるので、まだ望ましいです。

Linuxブリッジ・インターフェース（一般に `vmbrX` と呼ばれる）は、ゲストを基礎となる物理ネットワークに接続するために必要である。これは、ゲストと物理インターフェイスが接続される仮想スイッチと考えることができます。このセクションでは、[ボンドによる冗長化](#)、[VLAN](#)、ルーティングや[NAT](#) のセットアップなど、さまざまな使用ケースに対応するためにネットワークをどのようにセットアップできるか、いくつかの例を示します。

[Software Defined Network](#) は、Proxmox VE クラスタにより複雑な仮想ネットワークを構築するためのオプションで

す。

---

**警告**

不安であれば、Debianの伝統的なツールであるifupとifdownを使うのはお勧めしません。vmbtrX のifdownですべてのゲストのトラフィックが中断されますが、後で同じブリッジでifupを行ったときに、それらのゲストが再接続されないといった落とし穴があるからです。

---

### 3.4.1 ネットワークの変更を適用する

Proxmox VE は /etc/network/interfaces に直接変更を書き込みません。代わりに、/etc/network/interfaces.new という一時ファイルに書き込みます。こうすることで、関連する多くの変更を一度に行なうことができます。こうすることで、関連する多くの変更を一度に行なうことができます。また、間違ったネットワーク設定を行うとノードにアクセスできなくなる可能性があるため、適用する前に変更が正しいことを確認することができます。

## ifupdown2によるライブ・リロード・ネットワーク

推奨の *ifupdown2* パッケージ（Proxmox VE 7.0 以降の新規インストールのデフォルト）を使用すると、再起動せずにネットワーク構成の変更を適用できます。GUI 経由でネットワーク構成を変更した場合、[Apply Configuration] ボタンをクリックできます。これにより、ステージング `interfaces.new` ファイルから `/etc/network/interfaces` に変更が移動し、ライブで適用されます。

`etc/network/interfaces` ファイルに直接手動で変更を加えた場合は、`ifreload -a` を実行することで適用できる。

---

### 注

Debian 上に Proxmox VE をインストールした場合、または古い Proxmox VE から Proxmox VE 7.0 にアップグレードした場合は、*ifupdown2* がインストールされていることを確認してください。

---

## ノードを再起動して適用する

新しいネットワーク設定を適用するもう1つの方法は、ノードを再起動することです。この場合、`systemd` サービス `pvenetcommit` が `staging interfaces.new` ファイルをアクティブにしてから、ネットワークサービスがその設定を適用します。

### 3.4.2 命名規則

現在、デバイス名には以下の命名規則を使用しています：

- イーサネットデバイス: `en*`, `systemd` ネットワークインターフェース名。この命名法は、バージョン5.0 以降、Proxmox VE の新規インストールに使用されます。
- イーサネットデバイス: `eth[N]`,  $0 \leq N$  (`eth0, eth1, ...`) この命名法は、5.0リリース以前にインストールされたProxmox VEホストに使用されます。5.0にアップグレードする場合、名前はそのまま使用されます。
- ブリッジ名: 一般的には `vmb[r][N]` で、 $0 \leq N \leq 4094$  (`vmb[r]0 ~ vmb[r]4094`) ですが、文字で始まり、最大10文字までの英数字文字列なら何でも使えます。
- ボンド: `bond[N]`、ここで  $0 \leq N$  (`bond0, bond1, ...`)
- VLAN: デバイス名に VLAN 番号をピリオドで区切って追加するだけです (`eno1.50, bond1.30`)。デ

バイス名がデバイスのタイプを意味するため、ネットワークの問題をデバッグしやすくなります。

## Systemd ネットワークインターフェース名

Systemd はネットワーク・デバイス名のバージョン・ネーミング・スキームを定義しています。このスキームでは、イーサネットネットワークデバイスに2文字の接頭辞 `en` を使用します。次の文字はデバイスドライバやデバイスの場所、その他の属性に依存します。いくつかのパターンが考えられます：

- `o<index>[n<phys_port_name>|d<dev_port>]` - ボード上のデバイス
- `s<slot>[f<function>][n<phys_port_name>|d<dev_port>]` - ホットプラグIDによるデバイス
  -

- [P<domain>] p<bus>s<slot>[f<function>] [n<phys\_port\_name>|d<dev\_port>] - バスIDによるデバイス。
- x<MAC> - MACアドレスによるデバイス

最も一般的なパターンの例をいくつか挙げよう：

- eno1 - 最初のオンボードNIC
- enp3s0f1 - PCIバス3、スロット0上のNICのファンクション1です。

可能なデバイス名パターンの完全なリストについては、[systemd.net-naming-scheme\(7\) man](#) ページを参照してください。

systemd の新しいバージョンは、ネットワークデバイスの命名スキームの新しいバージョンを定義し、それをデフォルトで使用します。そのため、Proxmox VE のメジャーアップグレード時などに systemd を新しいバージョンに更新すると、ネットワークデバイスの名前が変更され、ネットワーク設定の調整が必要になることがあります。新しいバージョンの命名スキームによる名前の変更を避けるには、特定の命名スキームバージョンを手動で固定します（[下記参照](#)）。

ただし、名前付けを固定したバージョンでも、カーネルやドライバの更新によってネットワークデバイス名が変更される可能性があります。特定のネットワーク・デバイスの名前の変更を完全に避けるには、リンク・ファイルを使用して手動で名前を上書きします（[下記参照](#)）。

ネットワーク・インターフェース名の詳細については、[予測可能なネットワーク・インターフェース名](#)を参照してください。

### 特定のネーミングスキームのバージョンをピン留めする

カーネルコマンドラインに `net.naming-scheme=<` パラメータを追加することで、ネットワークデバイスのネーミングスキームの特定のバージョンを固定することができます。命名スキームのバージョンの一覧は、[systemd.net-naming-scheme\(7\) man](#) ページを参照してください。

例えば、新しいProxmox VEの最新のネーミングスキームのバージョンであるv252を固定するには、次のようにします。

8.0のインストールでは、以下のカーネル・コマンドライン・パラメーターを追加する：

```
net.naming-scheme=v252
```

カーネルコマンドラインの編集については、[このセクションも](#)参照してください。変更を有効にするには再起動が必要です。

## ネットワークデバイス名の上書き

カスタム [systemd.link ファイルを使って](#)、特定のネットワークデバイスに手動で名前を割り当てるすることができます。これは最新のネットワークデバイス命名スキームに従って割り当てられる名前を上書きします。こうすることで、カーネルのアップデートやドライバのアップデート、命名スキームの新しいバージョンによる名前の変更を避けることができます。

カスタムリンクファイルは `/etc/systemd/network/` に置き、`<n>-<id>.link` という名前を付けなければならない。リンクファイルには2つのセクションがあります： `[Match]` は、このファイルがどのインターフェースに適用されるかを決定し、`[Link]` は、これらのインターフェースがどのように設定されるべきかを決定する。

特定のネットワーク・デバイスに名前を割り当てるには、`[Match]` セクションでそのデバイスを一意かつ永続的に識別する方法が必要です。`MACAddress`] セクションを使用してデバイスのMACアドレスと一致させることもできます。

オプションを使ってください。次に、[Link]セクションの[Name]オプションを使用して名前を割り当てることができます。

例えば、MACアドレスがaa:bb:cc:dd:ee:ffのデバイスにenwan0という名前を割り当てるには、以

```
[マッチ] MACアドレス  
=aa:bb:cc:dd:ee:ff
```

```
[リンク] 名  
前=enwan0
```

下の内容で/etc/systemd/network/10-enwan0.linkファイルを作成する：

新しい名前を使用するように/etc/network/interfacesを調整するのを忘れないこと。変更を有効にするには、ノードを再起動する必要があります。

---

### 注

Proxmox VEがインターフェイスを物理的なネットワークデバイスとして認識し、GUIで設定できるように、enまたはethで始まる名前を割り当てることをお勧めします。また、この名前が将来的に他のインターフェイス名と衝突しないようにする必要があります。上の例ではenwan0のように、systemdがネットワークインターフェイスに使用する名前パターン（[上記参照](#)）にマッチしない名前を割り当てることもできます。

---

リンクファイルの詳細については、[systemd.link\(5\) man ページ](#)を参照してください。

### 3.4.3 ネットワーク構成の選択

現在のネットワーク構成やリソースに応じて、ブリッジ型、ルーティング型、マスカレード型のいずれかのネットワーク設定を選ぶことができる。

プライベートLAN内のProxmox VEサーバ、インターネットへのアクセスには外部ゲートウェイを使用

この場合、ブリッジモデルが最も理にかなっており、Proxmox VEの新規インストール時のデフォルトモードでもあります。各ゲストシステムはProxmox VEブリッジに接続された仮想インターフェイスを持ちます。これは、ゲストのネットワークカードがLAN上の新しいスイッチに直接接続され、Proxmox VEホストがスイ

---

ツの役割を果たすのと同じです。

### ホスティングプロバイダのProxmox VEサーバ、ゲスト用のパブリックIPレンジ付き

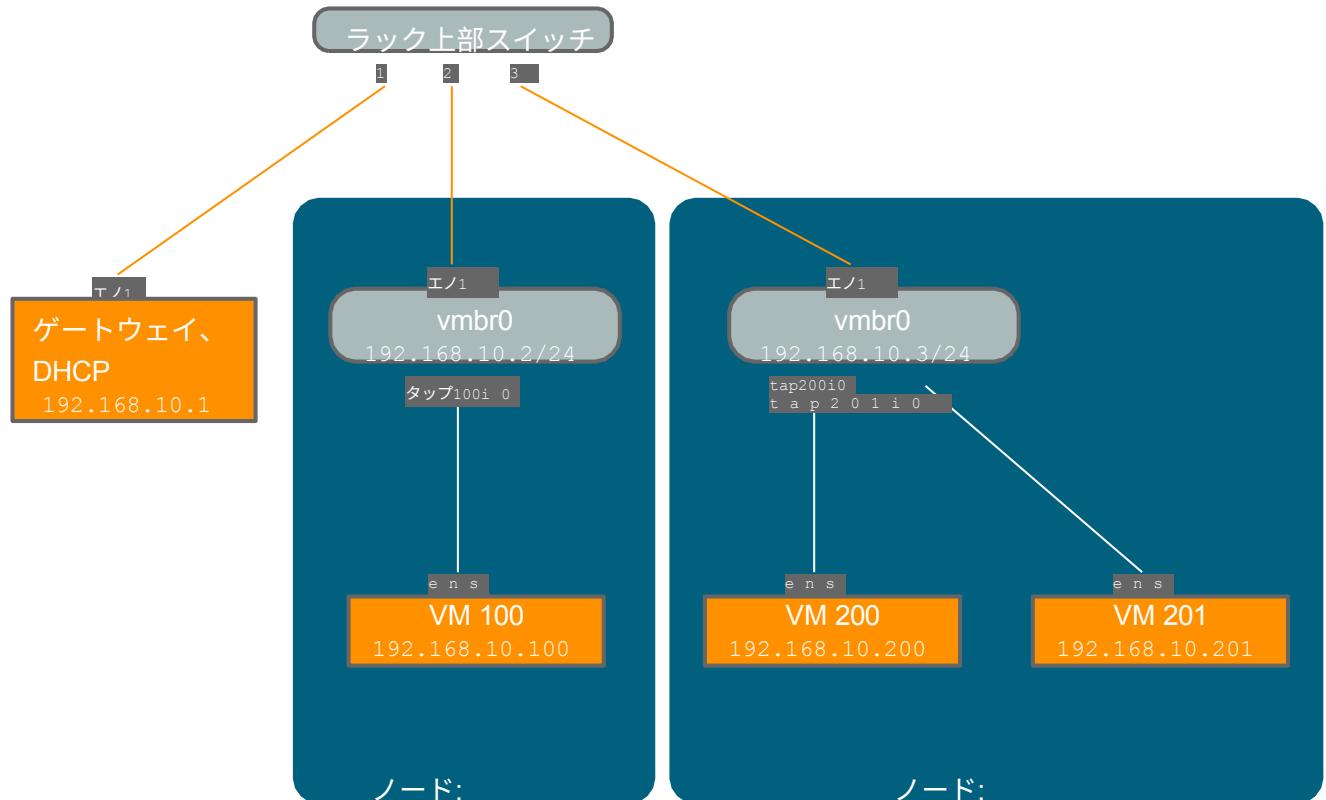
このセットアップでは、プロバイダーが許可している内容に応じて、ブリッジモデルまたはルーティングモデルのいずれかを使用できます。

### ホスティングプロバイダのProxmox VEサーバ、単一のパブリックIPアドレス付き

この場合、ゲストシステムの発信ネットワークアクセスを取得する唯一の方法は、**マスカレード**を使用することです。ゲストへの着信ネットワークアクセスには、**ポートフォワーディング**を設定する必要があります。

さらに柔軟性を高めるために、VLAN (IEEE 802.1q) やネットワークボンディング（別名「リンクアグリゲーション」）を設定することができる。これにより、複雑で柔軟な仮想ネットワークを構築することができます。

### 3.4.4 ブリッジを使用したデフォルト設定



ブリッジは、ソフトウェアで実装された物理的なネットワークスイッチのようなものです。すべての仮想ゲストは単一のブリッジを共有できますが、複数のブリッジを作成してネットワークドメインを分けることもできます。各ホストは最大4094個のブリッジを持つことができます。

インストール・プログラムは**vmbr0**という名前のブリッジを1つ作成し、最初のイーサネット・カードに接

```

オートロー
iface lo inet

loopback iface eno1

inet manual

自動 vmbr0
iface vmbr0 inet static
    アドレス 192.168.10.2/24

    ゲートウェイ
    192.168.10.1 bridge-
    ports eno1 bridge-stp
    off
    ブリッジ-FD 0
  
```

続します。etc/network/interfacesの対応するコンフィギュレーションは以下のようになる:

仮想マシンは、あたかも物理ネットワークに直接接続されているかのように動作する。ネットワークは、各仮想マシンをそれぞれ独自のMACを持っていると見なしますが、仮想マシンをネットワークに接続しているネットワークケーブルは1本しかありません。

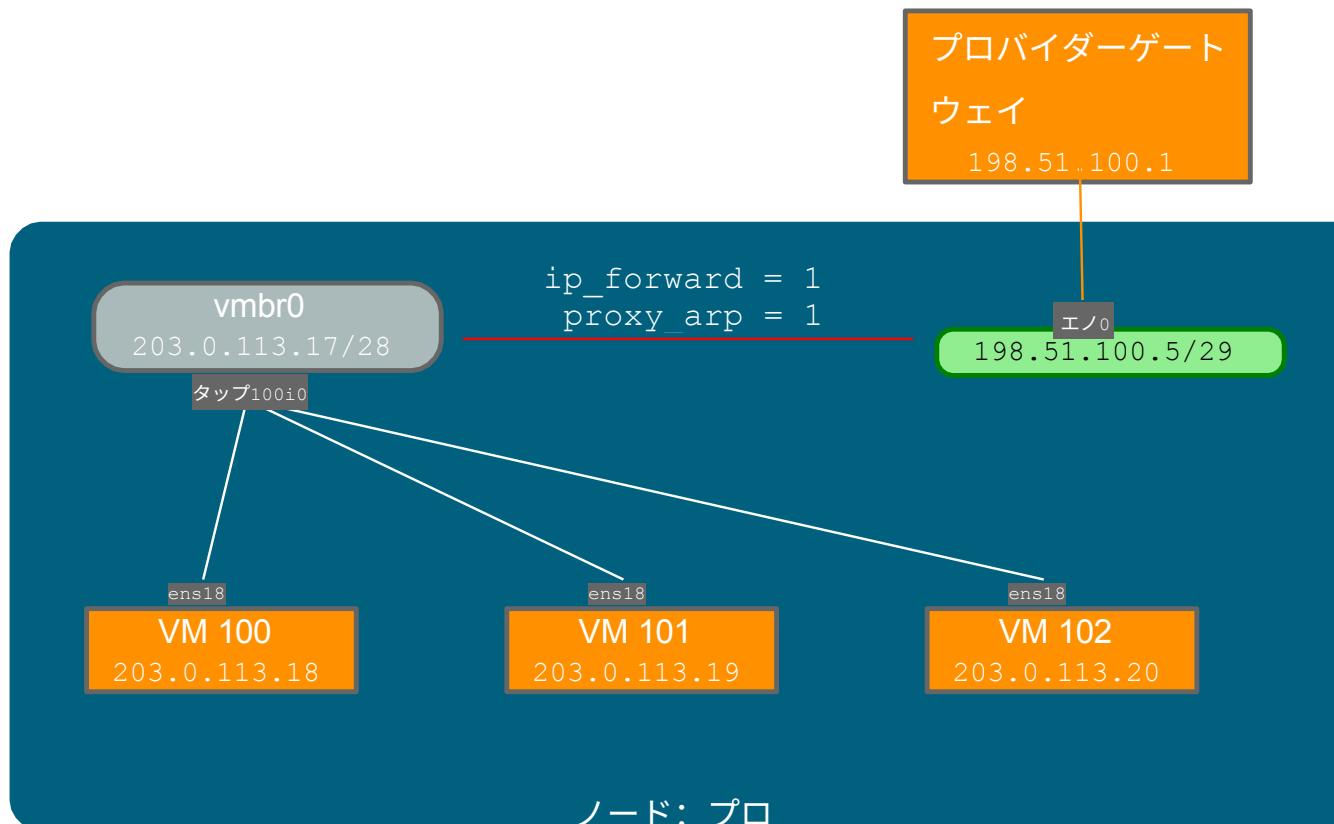
### 3.4.5 ルーティング・コンフィギュレーション

ほとんどのホスティング・プロバイダーは、上記の設定をサポートしていない。セキュリティ上の理由から、1つのインターフェイスで複数のMACアドレスを検出すると、すぐにネットワークを無効にする。

#### チップ

プロバイダーによっては、管理インターフェイスからMACを追加登録できるものもある。これにより問題は回避できるが、VMごとにMACを登録する必要があるため、設定が面倒になる可能性がある。

すべてのトラフィックを単一のインターフェイス経由で「ルーティング」することで、この問題を回避することができます。これにより、すべてのネットワークパケットが同じMACアドレスを使うようになります。



よくあるシナリオは、パブリックIP（この例では198.51.100.5とする）と、VM用の追加IPブロック（203.0.113.16/28）がある場合だ。このような場合、以下のようなセットアップをお勧めします：

**オートロー**

```
iface lo inet loopback
```

**オートエノ0**

```
iface eno0 inet static
```

アドレス 198.51.100.5/29 ゲート

ウェイ 198.51.100.1

```
post-up echo 1 > /proc/sys/net/ipv4/ip_forward
post-up echo 1 > /proc/sys/net/ipv4/conf/eno0/proxy_arp
```

**自動 vmbr0**

```
iface vmbr0 inet static
```

```
アドレス 203.0.113.17/28
bridge-ports none
bridge-stp off
ブリッジ-FD 0
```

### 3.4.6 iptablesによるマスカレード (NAT)

マスカレードは、プライベートIPアドレスしか持たないゲストが、送信トラフィックにホストIPアドレスを

```
オートロー
iface lo inet ループバック

オートエノ1
#実IPアドレス iface
en0 inet static
    アドレス 198.51.100.5/24 ゲート
    ウェイ 198.51.100.1

自動 vmbr0
#プライベート・サブ・ネットワ
ーク iface vmbr0 inet
static
    アドレス 10.10.10.1/24
    bridge-ports none
    bridge-stp off
    ブリッジ-FD 0

post-up echo 1 > /proc/sys/net/ipv4/ip_forward
post-up iptables -t nat -A POSTROUTING -s '10.10.10.0/24' -o
en0 -j MASQUERADE
ポストダウン iptables -t nat -D POSTROUTING -s '10.10.10.0/24' -o en0
-j MASQUERADE
```

使用することで、ネットワークにアクセスできるようにします。各送信パケットはiptablesによってホストから発信されているように書き換えられ、レスポンスもそれに応じて書き換えられ、元の送信者にルーティングされます。

## 注

ファイアウォールを有効にしたマスカレード・セットアップでは、発信接続にコントラック・ゾーンが必要になる場合がある。そうしないと、ファイアウォールは（MASQUERADEではなく）VMブリッジのPOSTROUTINGを優先するため、発信接続をブロックする可能性がある。

---

```
post-up iptables -t raw -I PREROUTING -i fwbr+ -j CT  
--zone 1 post-down iptables -t raw -D PREROUTING -i fwbr+ -j  
CT --zone 1
```

etc/network/interfacesに以下の行を追加すると、この問題を解決できる：

これについては、以下のリンクを参照してください： [ネットフ](#)

[ィルターのパケットフロー](#)

[conntrackゾーンを導入するnetdev-listのパッチ](#)

[生テーブルでTRACEを使用することで良い説明があるブログ記事](#)

### 3.4.7 Linuxボンド

ボンディング（NICチーミングまたはリンクアグリゲーションとも呼ばれる）は、複数のNICを1つのネットワークデバイスにバインドする技術です。ネットワークのフォールト・トレラント化、パフォーマンスの向上、またはその両方を同時に実現するなど、さまざまな目的を達成することができます。

ファイバー・チャネルのような高速ハードウェアと、それに関連するスイッチング・ハードウェアは、かなり高価になります。リンクアグリゲーションを行うことで、2つのNICを1つの論理インターフェイスとして表示することができ、結果として2倍の速度になります。これはLinuxカーネルのネイティブ機能で、ほとんどのスイッチでサポートされています。ノードに複数のイーサネットポートがある場合、異なるスイッチにネットワークケーブルを接続することで、障害点を分散させることができます。

集約リンクは、ライブマイグレーションの遅延を改善し、Proxmox VE Clusterノード間のデータのレプリケーション速度を向上させることができます。

ボンディングには7つのモードがある：

- **ラウンドロビン (balance-rr)**： 利用可能な最初のネットワークインターフェイス (NIC) スレーブから最後のスレーブまで、順次ネットワークパケットを送信します。このモードは負荷分散とフォールトトレランスを提供します。
- **アクティブ・バックアップ (active-backup)**： ボンド内の1つのNICスレーブだけがアクティブになる。アクティブスレーブに障害が発生した場合のみ、別のスレーブがアクティブになる。単一の論理ボンディング・インターフェースのMACアドレスは、ネットワーク・スイッチの歪みを避けるために、1つのNIC（ポート）のみで外部に表示されます。このモードはフォールトトレランスを提供します。
- **XOR (balance-xor)**： 送信元MACアドレスと宛先MACアドレスのXOR] モジュロNICスレーブ数]に基づいてネットワークパケットを送信する。これは、各宛先MACアドレスに対して同じNICスレーブを選択します。このモードは負荷分散とフォールトトレランスを提供する。
- **ブロードキャスト (放送)**： すべてのスレーブネットワークインターフェースにネットワークパケットを送信します。このモードはフォールトトレランスを提供します。
- **IEEE 802.3ad ダイナミックリンクアグリゲーション(802.3ad)(LACP)**： 同じ速度とデュプレックス設定を共有するアグリゲーショングループを作成します。802.3ad仕様に従って、アクティブアグリゲーターグループ内のすべてのスレーブネットワークインターフェースを使用します。
- **アダプティブ・トランスマッシュョン・ロードバランシング (balance-tlb)**： 特別なネットワークスイッチ

のサポートを必要としないLinuxボンディングドライバモード。発信ネットワーク・パケット・トラフィックは、各ネットワーク・インターフェース・スレーブの現在の負荷（速度に対して計算される）に従って分配される。着信トラフィックは、現在指定されている1つのスレーブ・ネットワーク・インターフェースで受信される。この受信スレーブが故障した場合、別のスレーブが故障した受信スレーブのMACアドレスを引き継ぎます。

- **適応型ロードバランシング (balance-alb)** : balance-tlbにIPV4トラフィックの受信負荷分散(rlb)を加えたもので、特別なネットワークスイッチのサポートは必要ありません。受信負荷分散はARPネゴシエーションによって達成される。ボンディングドライバーは、ローカルシステムから送信されるARPリプライをインターセプトし、異なるネットワークピアがネットワークパケットトラフィックに異なるMACアドレスを使用するように、単一の論理ボンディングインターフェース内のNICスレーブの1つのユニークなハードウェアアドレスでソースハードウェアアドレスを上書きします。

お使いのスイッチがLACP（IEEE 802.3ad）プロトコルをサポートしている場合は、対応するボンディングモード（802.3ad）を使用することをお勧めします。そうでない場合は、一般的にアクティブバックアップモードを使用する必要があります。

クラスタネットワーク（Corosync）については、複数のネットワークで構成することをお勧めします。

Corosyncは、1つのネットワークが使用できなくなった場合、それ自身でネットワークを切り替えることができるため、ネットワークの再切断のためのボンドは必要ありません。

以下のボンド構成は、分散/共有ストレージネットワークとして使用できる。利点は、より高速になり、ネットワークがフォールトレントになることである。

## 例固定IPアドレスでボンドを使用

### オートロー

```
iface      lo      inet
loopback  iface   eno1
inet      manual  iface
eno2      inet    manual
iface     eno3    inet
manual
```

### オートボンド0

```
iface bond0 inet static
  bond-slaves eno1
  eno2
  アドレス 192.168.1.2/24
  bond-miimon 100
  ボンドモード 802.3ad
  bond-xmit-hash-policyレイヤー2+3
```

### 自動 vmbr0

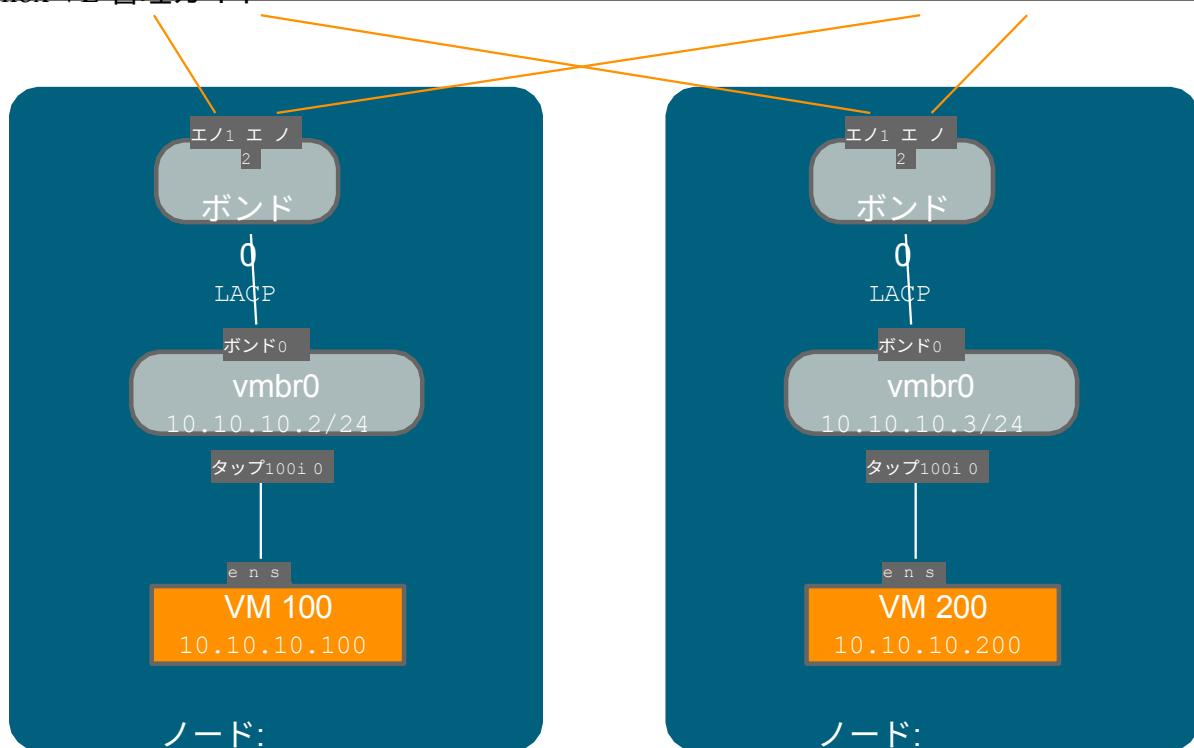
```
iface vmbr0 inet static
  アドレス 10.10.10.2/24
  ゲートウェイ 10.10.10.1
  bridge-ports eno3
  bridge-stp off
  ブリッジ-FD 0
```

MLAG

1 2

ラック上部スイッチ 2

1 2



もう一つの可能性は、ブリッジポートとしてボンドを直接使用することです。これはゲストネットワークをフォールトレラントにするために使えます。

ラック上部 スイッチ 1

## 例ブリッジポートとしてボンドを使用

オートロー

```
iface      lo      inet
loopback  iface   eno1
inet      manual  iface
eno2      inet manual
```

オートボンド0

```
iface bond0 inet manual
    bond-slaves eno1
    eno2 bond-miimon 100
    ボンドモード 802.3ad
    bond-xmit-hash-policyレイヤー2+3
```

自動 vmbr0

```
iface vmbr0 inet static
    アドレス 10.10.10.2/24
    ゲートウェイ 10.10.10.1
    bridge-ports bond0
    bridge-stp off
    ブリッジ-FD 0
```

### 3.4.8 VLAN 802.1Q

仮想LAN (VLAN) はブロードキャスト・ドメインであり、レイヤー2のネットワークで分割・分離される。そのため、物理ネットワーク内に複数のネットワーク (4096) を持ち、それぞれが他のネットワークから独立していることが可能です。

各VLANネットワークは、しばしばタグと呼ばれる番号で識別される。ネットワークパッケージは、どの仮想ネットワークに属するかを識別するためにタグ付けされます。

#### ゲストネットワーク用VLAN

Proxmox VEはこのセットアップをすぐにサポートします。VMの作成時にVLANタグを指定できます。VLANタグはゲストネットワーク設定の一部です。ネットワーキング・レイヤは、ブリッジ構成に応じて、VLANを実装するためのさまざまなモードをサポートします：

- **Linux ブリッジで VLAN を認識します:** この場合、各ゲストの仮想ネットワークカードにはVLANタグが割り当てられ、Linuxブリッジによって透過的にサポートされます。トランクモードも可能ですが、その場合はゲストでの設定が必要になります。
- **Linux ブリッジ上の "従来の" VLAN:** VLANを意識する方法とは対照的に、この方法は透過的ではなく、各VLANに関連するブリッジとVLANデバイスを作成します。つまり、例えば VLAN 5 にゲストを作成すると、eno1.5 と vmbr0v5 の2つのインターフェースが作成され、リブートが発生するまで残ります。
- **Open vSwitch VLAN:** このモードでは、OVS VLAN機能を使用します。
- **ゲスト設定VLAN:** VLANはゲスト内部で割り当てられます。この場合、設定は完全にゲスト内部で行われ、外部から影響を受けることはありません。利点は、1つの仮想NICで複数のVLANを使用できることです
  -

## ホスト上のVLAN

隔離されたネットワークとのホスト通信を可能にする。任意のネットワークデバイス（NIC、ボンド、ブリッジ）にVLANタグを適用することが可能です。一般的には、それ自身と物理NICとの間の抽象化レイヤーが最も少ないインターフェースにVLANを設定する必要があります。

例えば、デフォルトのコンフィギュレーションで、ホスト管理アドレスを別のVLANに置きたい場合。

### 例従来のLinuxブリッジでProxmox VEの管理IPにVLAN 5を使用する

#### オートロー

```
iface lo inet loopback  
  
iface eno1 inet manual  
  
iface eno1.5 inet manual
```

#### オートvmbr0v5

```
iface vmbr0v5 inet static  
    アドレス 10.10.10.2/24  
    ゲートウェイ 10.10.10.1  
    bridge-ports eno1.5  
    bridge-stp off  
    ブリッジ-FD 0
```

#### 自動 vmbr0

```
iface vmbr0 inet マニュアル  
    bridge-ports  
    eno1 bridge-stp  
    off bridge-fd 0
```

例VLAN を認識する Linux ブリッジで Proxmox VE 管理 IP に VLAN 5 を使用します。

**オートロー**

```
iface lo inet  
  
loopback iface eno1  
  
inet manual
```

**オートvmbr0.5**

```
iface vmbr0.5 inet static  
    アドレス 10.10.10.2/24  
    ゲートウェイ 10.10.10.1
```

**自動 vmbr0**

```
iface vmbr0 inet マニュアル  
    bridge-ports  
    eno1 bridge-stp  
    off bridge-fd 0  
    ブリッジ・ブラン・アウェア
```

## ブリッジビデオ 2-4094

次の例は同じセットアップだが、このネットワークをフェイルセーフにするためにボンドが使われている。

例従来の Linux ブリッジで、Proxmox VE 管理 IP に Bond0 を使用して VLAN 5 を使用します。

オートロー

```
iface      lo      inet
loopback  iface   eno1
inet      manual   iface
eno2     inet manual
```

オートボンド0

```
iface bond0 inet manual
    bond-slaves eno1
    eno2 bond-miimon 100
    ボンドモード 802.3ad
    bond-xmit-hash-policy layer2+3

iface bond0.5 inet manual
```

オートvmbr0v5

```
iface vmbr0v5 inet static
    アドレス 10.10.10.2/24
    ゲートウェイ 10.10.10.1
    bridge-ports bond0.5
    bridge-stp off
    ブリッジ-FD 0
```

自動 vmbr0

```
iface vmbr0 inet マニュアル
    bridge-ports
    bond0 bridge-stp
    off bridge-fd 0
```

### 3.4.9 ノードのIPv6を無効にする

Proxmox VEは、IPv6の導入の有無にかかわらず、すべての環境で正しく動作します。すべての設定をデフォルトのままにしておくことをお勧めします。

ノードのIPv6サポートを無効にする必要がある場合は、適切なsysctl.confを作成してください。

(5) スニペットファイルと適切なsysctlsを設定する。例えば、/etc/sysctl.d/disable-ipv6.confを追加する。

内容を含む：

```
net.ipv6.conf.all.disable_ipv6 = 1  
net.ipv6.conf.default.disable_ipv6 = 1
```

この方法は、**カーネルのコマンド**ラインでIPv6モジュールのロードを無効にするよりも好ましい。

### 3.4.10 ブリッジでMAC学習を無効にする

デフォルトでは、MAC学習は、仮想ゲストとそのネットワークでスムーズなエクスペリエンスを保証するために、ブリッジ上で有効になっています。

しかし、環境によってはこれが望ましくないこともある。Proxmox VE 7.3以降では、ブリッジの'bridge-

```
# ...

自動 vmbr0
iface vmbr0 inet static
    アドレス 10.10.10.2/24
    ゲートウェイ 10.10.10.1
    bridge-ports ens18
    bridge-stp off
    ブリッジ-FD 0
    bridge-disable-mac-learning 1
```

disable-mac-learning 1`設定を`/etc/network/interfaces`に設定することで、ブリッジのMAC学習を無効することができます：

有効化されると、Proxmox VEはVMとコンテナから設定されたMACアドレスをブリッジ転送データベースに手動で追加し、ゲストが実際のMACアドレステを使用している場合のみネットワークを使用できるようになります。

## 3.5 時間同期

Proxmox VEクラスタスタック自体は、すべてのノードの時刻が正確に同期していることに大きく依存しています。Cephのような他のコンポーネントも、すべてのノードのローカル時刻が同期していないと正しく動作しません。

ノード間の時刻同期は、"Network Time Protocol" (NTP) を使用して行います。Proxmox VE 7では、デフォルトのNTPデーモンとしてchronyが使用され、Proxmox VE 6ではsystemd-timesyncdが使用されます。どちらも、一連のパブリックサーバーを使用するように事前に設定されています。

---

### 重要

---

! システムをProxmox VE 7にアップグレードする場合は、以下のいずれかを手動でインストールすることをお勧めします。

chrony、ntpまたはopenntpd。

### 3.5.1 カスタムNTPサーバーの使用

場合によっては、デフォルト以外のNTPサーバを使用したいことがあります。たとえば、Proxmox VEノードが制限されたファイアウォールルールのためにパブリックインターネットにアクセスできない場合、ローカルNTPサーバをセットアップして、NTPデーモンにそれを使用するように指示する必要があります。

**クロニーを使用するシステムの場合:**

etc/chrony/chrony.confでchronyが使用するサーバーを指定します:

```
サーバ ntp1.example.com iburst サ  
ーバ ntp2.example.com iburst サー  
バ ntp3.example.com iburst
```

クロニーを再起動する:

```
# systemctl restart chronyd
```

ジャーナルをチェックして、新しく設定したNTPサーバーが使用されていることを確認する:

```
# journalctl --since -1h -u chrony
```

```
...  
Aug 26 13:00:09 node1 systemd[1]: NTP クライアント/サーバである chrony を開始した。  
Aug 26 13:00:15 node1 chronyd[4873]: 選択されたソース10.0.0.1 (ntp1.example ←  
.com)  
Aug 26 13:00:15 node1 chronyd[4873]: システムクロックTAIオフセットが37に設定された ←  
おわり  
...
```

**systemd-timesyncd を使用しているシステムの場合:**

etc/systemd/timesyncd.conf で systemd-timesyncd が使用するサーバーを指定します:

#### 時間

```
NTP=ntp1.example.com ntp2.example.com ntp3.example.com ntp4.example.com
```

その後、同期サービスを再起動し (systemctl restart systemd-timesyncd) 、ジャーナルをチェックして新しく設定したNTPサーバーが使用されていることを確認します (journalctl --since -1h

-u systemd-timesyncd) :

```
...  
Oct 07 14:58:36 node1 systemd[1]: ネットワーク時刻同期を停止しています...Oct 07  
14:58:36 node1 systemd[1]: ネットワーク時刻同期の開始...Oct 07 14:58:36 node1  
systemd[4]58ネットワーク時刻同期を開始しました。[13514]ドットネット時刻同期を開始しました。  
10.0.0.1:123 (ntp1.example.com)を使用しています。  
Oct 07 14:58:36 node1 systemd-timesyncd[13514]: interval/delta/delay/jitter ←  
.drift 64s/-0.002s/0.020s/0.000s/-31ppm  
...
```

## 3.6 外部メトリック・サーバー

Name	Type	Enabled	Server	Port
graphite-test	Graphite	Yes	192.168.0.50	2003
influxdb-test	InfluxDB	Yes	192.168.0.60	8089

Proxmox VEでは、ホスト、仮想ゲスト、ストレージに関する様々な統計情報を定期的に受け取る外部メトリックサーバを定義できます。

現在サポートされているのは

- グラフアイト (<https://graphiteapp.org> を参照)
- InfluxDB (<https://www.influxdata.com/time-series-platform/influxdb/> を参照)

外部メトリック・サーバー定義は`/etc/pve/status.cfg`に保存され、ウェブ・インターフェイスから編集できる。

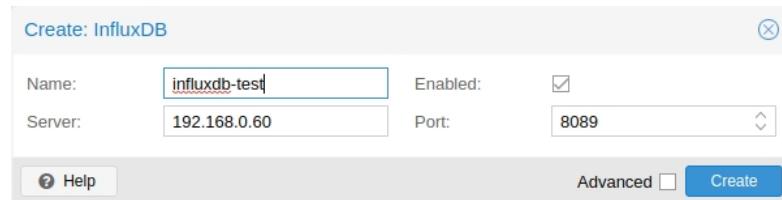
### 3.6.1 Graphiteサーバーの構成

デフォルトのポートは**2003**に設定され、デフォルトのグラファイトパスは**proxmox**である。

デフォルトでは、Proxmox VEはUDPでデータを送信するため、グラファイトサーバーはこれを受け入れるように設定する必要があります。標準の**1500MTU**を使用しない環境では、ここで最大伝送単位（MTU）を設定することができます。

TCPを使用するようにプラグインを設定することもできます。重要なpvestatd統計収集デーモンをプロックしないために、ネットワークの問題に対処するためのタイムアウトが必要です。

### 3.6.2 Influxdbプラグインの設定



Proxmox VEはUDPでデータを送信するので、influxdbサーバーはこのように設定する必要があります。必要に応じてMTUもここで設定できます。

以下は influxdb (influxdb サーバー) の設定例です：

```
[udp] である。  
有効 = true  
バインドアドレス = "0.0.0.0:8089" デ  
ータベース = "proxmox"  
バッチサイズ = 1000 バ  
ッヂタイムアウト =  
"1s"
```

この設定では、サーバーは8089番ポートですべてのIPアドレスをリッスンし、データをプロックスモックス・データベース

InfluxDB 1.8.x には、このv2 APIと互換性のあるAPIエンドポイントが含まれています。

これを使用するには、*influxdbproto* を *http* または *https* に設定します(設定に依存します)。デフォルトでは、Proxmox VEはorganization proxmoxとbucket/db proxmoxを使用します(それぞれorganizationとbucketの設定で設定できます)。

InfluxDBのv2 APIは認証がないと使えないで、正しいバケットに書き込めるトークンを生成して設定する

必要がある。

1.8.xのv2互換APIでは、トークンとして*user:password*を使用することができます（必要な場合）。

InfluxDB1.xでは意味がない。

また、*timeout*設定でHTTPタイムアウト（デフォルトは1秒）を、*max-body-size*設定で最大バッチサイズ（デフォルトは25000000バイト）を設定することができます（これは同名のInfluxDBの設定に対応します）。

### 3.7 ディスクヘルス・モニタリング

堅牢で冗長性のあるストレージを推奨しますが、ローカルディスクの健全性を監視することは非常に有用です。

Proxmox VE 4.3から、smartmontoolsパッケージ<sup>1</sup>をインストールする必要があります。これは、ローカルハードディスクのS.M.A.R.T.システムを監視および制御するためのツールセットです。

以下のコマンドを実行すれば、ディスクの状態を知ることができます：

```
# smartctl -a /dev/sdX
```

dev/sdXはローカルディスクのパスです。出力が

SMARTのサポートは無効

コマンドで有効にできる：

```
# smartctl -s on /dev/sdX
```

smartctlの使い方については、man smartctlを参照のこと。

デフォルトでは、smartmontoolsデーモンsmartdがアクティブで有効になっており、/dev/sdX以下のディスクをスキヤンし、/dev/sdX以下のディスクをスキヤンする。

/dev/hdXのエラーと警告を30分ごとにチェックし、問題があればrootにメールを送る。

smartdの設定方法の詳細については、man smartdおよびman smartd.confを参照のこと。

ハードディスクをハードウェアの RAIDコントローラーで使用している場合、 RAIDアレイ内のディスクやアレイ自体を監視するツールがある場合がほとんどです。これについての詳細は、 RAIDコントローラーのベンダーにお問い合わせください。

## 3.8 論理ボリュームマネージャ (LVM)

ほとんどの人はProxmox VEをローカルディスクに直接インストールします。Proxmox VEのインストールCDには、ローカルディスク管理のためのいくつかのオプションが用意されており、現在のデフォルトのセット

```
# PV
/dev/sda3  VG      Fmt Attr PSize PFree
             レ      lvm2 a--  7.87g 876.00m
             ベ
             ル

# vgs
VG      #PV #LV #SN Attr   VSサイズ
レ        1    3    0 WZ--N-  VF 876.00M
             ベ
             ル
```

アップではLVMが使用されています。インストーラでは、このようなセットアップに使用するディスクを1つ選択でき、そのディスクをボリュームグループ(VG) pve の物理ボリュームとして使用します。以下の出

力は、8GBの小型ディスクを使用したテスト・インストールによるものです：

インストーラーはこの VG 内に 3 つの論理ボリューム (LV) を割り当てます：

#	lv	属性	Origin	LSize	Pool	データ	メタ
データ	ル	twi-a-tz--		4.38g		0.00	0.63
ルート	ル	-wiアオ		1.75g			
スワッ	ル	-wiアオ		896.00m			

## ルート

ext4でフォーマットされ、オペレーティングシステムを含む。

---

<sup>1</sup>smartmontools ホームページ <https://www.smartmontools.org>

## スワップ

スワップ・パーティション

## データ

このボリュームはLVM-thinを使用し、VMイメージの保存に使用される。LVM-thinは、スナップショットとクローンを効率的にサポートするため、このタスクに適している。

4.1までのProxmox VEバージョンでは、インストーラは "data" という標準の論理ボリュームを作成し、`/var/lib/vz` にマウントします。

バージョン4.2から、論理ボリューム「data」はLVM-thinプールで、ブロックベースのゲストイメージを格納するために使用され、`/var/lib/vz` は単にルートファイルシステム上のディレクトリです。

### 3.8.1 ハードウェア

このようなセットアップには、ハードウェアRAIDコントローラ（BBU付き）の使用を強くお勧めします。これにより、パフォーマンスが向上し、冗長性が確保され、ディスク交換が容易になります（ホットプラグ対応）。

LVM自体は特別なハードウェアを必要とせず、メモリ要件も非常に低い。

### 3.8.2 ブートローダー

デフォルトで2つのブートローダーをインストールします。最初のパーティションには標準的なGRUBブートローダーが含まれています。2つ目のパーティションはEFIシステムパーティション（ESP）で、EFIシステムでのブートと、ユーザースペースからの[永続的なファームウェアアップデート](#)の適用を可能にします。

### 3.8.3 ボリュームグループの作成

空のディスク`/dev/sdb`があると仮定し、そこに「vmdata」という名前のボリューム・グループを作成する。



#### 注意

以下のコマンドは、`/dev/sdb` 上の既存のデータをすべて破壊することに注意してください。

---

まずパーティションを作成する。

```
# sgdisk -N 1 /dev/sdb
```

確認なしで物理ボリューム (PV) を作成し、メタデータサイズを250Kにする。

```
# pvcreate --metadatasize 250k -y -ff /dev/sdb1
```

「vmdata」という名前のボリュームグループを

```
/dev/sdb1に作成 # vgcreate vmdata
```

```
/dev/sdb1
```

### 3.8.4 var/lib/vz用の追加LVの作成

これは新しい薄いLVを作ることで簡単にできる。

```
# lvcreate -n <名前> -V <サイズ [M,G,T]> <VG>/<LVThin_pool>
```

実例を挙げよう：

```
# lvcreate -n vz -V 10G pve/data
```

ここで、LV上にファイルシステムを作成しなければならない。

```
# mkfs.ext4 /dev/pve/vz
```

ついに、これが搭載されることになった。



#### 警告

var/lib/vzが空であることを確認してください。デフォルトのインストールではそうなっていない。

常にアクセスできるようにするには、/etc/fstabに以下の行を追加する。

```
# echo '/dev/pve/vz /var/lib/vz ext4 defaults 0 2' >> /etc/fstab
```

### 3.8.5 シンプールのサイズ変更

以下のコマンドでLVとメタデータプールのサイズを変更する：

```
# lvresize --size +<size [M,G,T]> --poolmetadatasize +<size [M,G]> <-> VG/<LVThin_pool>。
```

---

#### 注

データプールを拡張する場合、メタデータプールも拡張する必要がある。

---

### 3.8.6 LVM-thinプールを作成する

シンプールは、ボリュームグループの上に作成する必要があります。ボリュームグループの作成方法は、セクションLVMを参照。

---

```
# lvcreate -L 80G -T -n vmstore vmdata
```

### 3.9 Linux上のZFS

ZFSは、Sun Microsystemsによって設計されたファイルシステムと論理ボリュームマネージャを組み合わせたものです。Proxmox VE 3.4から、ZFSファイルシステムのネイティブLinuxカーネルポートがオプションのファイルシステムとして導入され、ルートファイルシステムの追加選択もできるようになりました。ZFSモジュールを手動でコンパイルする必要はありません。

- すべてのパッケージが含まれる。

ZFSを使用することで、低予算のハードウェアで最大限のエンタープライズ機能を実現することが可能です。また、SSDキャッシングやSSDのみのセットアップを活用することで、高性能システムも実現できます。ZFSは、CPUとメモリへの負荷を軽減し、管理を容易にすることで、コストのかかるハードウェアレイドカードを置き換えることができます。

#### ZFSの一般的な利点

- Proxmox VEのGUIとCLIで簡単な設定と管理。
- 信頼できる
- データ破損からの保護
- ファイルシステムレベルでのデータ圧縮
- スナップ写真
- コピー・オン・ライト・クローン
- 様々なレイドレベルRAID0、RAID1、RAID10、RAIDZ-1、RAIDZ-2、RAIDZ-3、dRAID、dRAID2、dRAID3
- キャッシュにSSDを使用可能
- セルフ・ヒーリング
- 繙続的な完全性チェック
- 大容量収納に対応した設計
- ネットワーク経由の非同期レプリケーション
- オープンソース
- 暗号化
- ...

### 3.9.1 ハードウェア

ZFSはメモリに大きく依存するので、最低でも8GBは必要だ。実際には、ハードウェア/予算に見合うだけの量を使用してください。データの破損を防ぐため、高品質のECC RAMの使用をお勧めします。

専用のキャッシュディスクやログディスクを使用する場合は、エンタープライズクラスのSSDを使用する必要があります。これにより、全体的なパフォーマンスを大幅に向上させることができる。

---

**重要**

独自のキャッシュ管理を持つハードウェアRAIDコントローラの上でZFSを使用しないでください。

ZFSはディスクと直接通信する必要があります。HBAアダプタか、"IT "モードでフラッシュされたLSIコントローラのようなものがより適切です。

---

VM（入れ子仮想化）内にProxmox VEをインストールして実験する場合、VMのディスクにvirtioを使用しないでください。代わりに IDE または SCSI を使用してください（virtio SCSI コントローラタイプでも動作します）。

### 3.9.2 ルートファイルシステムとしてのインストール

Proxmox VEインストーラを使用してインストールする場合、ルートファイルシステムにZFSを選択できます。インストール時にRAIDタイプを選択する必要があります：

RAID0	「ストライピング」とも呼ばれる。このようなボリュームの容量は、すべてのディスクの容量の合計です。しかし、RAID0 は冗長性を追加しないため、1台のドライブが故障するとボリュームは使用できなくなります。
RAID1	「ミラーリング」とも呼ばれる。データはすべてのディスクに同じように書き込まれる。このモードでは、同じサイズのディスクが少なくとも 2 台必要です。その結果、容量はディスク 1 台分となります。
RAID10	RAID0とRAID1を組み合わせたもの。少なくとも 4 台のディスクが必要。RAIDZ-1A RAID-5 のバリエーション、シングルパーティイ。
スクリプト	必要。RAIDZ-1A RAID-5 のバリエーション、シングルパーティイ。少なくとも 3 台のディスクが必要。RAIDZ-2A RAID-5 のバリエーション、ダブルパーティイ。少なくとも 4 台のディスクが必要。RAID-5 上のRAIDZ-3Aバリエーション、トリプルパーティイ。少なくとも 5 台のディスクが必要。

インストーラーは自動的にディスクをパーティション分割し、`rpool` という ZFS プールを作成し、ZFS サブボリューム `rpool/ROOT/pve-1` にルートファイルシステムをインストールします。

VM イメージを格納するために、rpool/data という別のサブボリュームが作成されます。これを

```
zfspool: local-zfs
  プール rpool/data
  スパース
  コンテンツ画像、ルートディレクトリ
```

Proxmox VEツールで使用するために、インストーラは/etc/pve/storage.cfgに以下の設定エントリを作成します：

インストール後、zpoolコマンドを使用してZFSプールのステータスを表示できます：

```
# zpool status
  pool: rpool
  状態オンライン
    scan: none リクエスト
```

された設定：

名前	状態	読む	WRITE	CKSUM
リップル	オンライン	0	0	0
ミラー0	オンライン	0	0	0
sda2	オンライン	0	0	0
sdb2	オンライン	0	0	0
ミラー1	オンライン	0	0	0
ディスク	オンライン	0	0	0

へん	オンラ イン	0	0	0
エラー既知のデ ちよ	タエラーはない			

`zfs` コマンドは、ZFS ファイルシステムの設定と管理に使用します。次のコマンドは、インストール後のすべてのファイルシステムを一覧表示します：

```
# zfs list
名前          中古   AVAIL  リファー  
              ー      マウントポイ  
リpool        4.94G  7.68T  96K  /rpool
rpool/ROOT    702M   7.68T  96K  /rpool/ROOT
rpool/ROOT/pve-1 702M   7.68T  702M /
rpool/データ   96K   7.68T  96K  /rpool/データ
```

### 3.9.3 ZFS RAIDレベルに関する考察

ZFS プールのレイアウトを選択する際に考慮すべき要素がいくつかあります。ZFS プールの基本的な構成要素は、仮想デバイス (`vdev`) です。プール内のすべての `vdev` は等しく使用され、データはそれらの間でストライプされます (RAID0)。`vdev` の詳細については、`zpoolconcepts(7)` man ページを確認してください。

#### パフォーマンス

各 `vdev` タイプはそれぞれ異なるパフォーマンス動作をする。注目すべき2つのパラメータは、IOPS (Input/Output Operations per Second) と、データの書き込みや読み出しが可能な帯域幅である。

ミラー `vdev` (RAID1) は、データを書き込む場合、両方のパラメーターに関して、ほぼシングルディスクのように動作します。データを読み込む場合、性能はミラー内のディスク数に比例します。

一般的な状況は、4つのディスクを持つことです。それを 2 つのミラー `vdev` (RAID10) としてセットアップすると、プールは IOPS と帯域幅に関して 2 つのシングルディスクとして書き込み特性を持つ。読み取り操作では、4 つの単一ディスクのようになります。

どの冗長レベルの RAIDZ でも、多くの帯域幅を持つ IOPS に関しては、ほぼシングルディスクと同じように動作します。どの程度の帯域幅になるかは、RAIDZ `vdev` のサイズと冗長レベルに依存します。

*d*RAID プールは同等の RAIDZ プールのパフォーマンスと同等であるべきで

ある。VM を実行する場合、ほとんどの状況では IOPS の方が重要な指標と

なる。

## サイズ、スペース使用量、冗長性

ミラー vdev で構成されるプールは最高のパフォーマンス特性を持つが、使用可能なスペースは使用可能なディスクの 50% となる。3 ウエイミラーなど、ミラー vdev が 2 台以上のディスクで構成されている場合は、それ以下になる。プールが機能し続けるためには、ミラーごとに少なくとも1つの健全なディスクが必要である。

N 台のディスクからなる *RAIDZ* タイプの vdev の使用可能領域はおおよそ N-P で、P は RAIDZ レベルです。RAIDZ- レベルは、データを失うことなく故障できる任意のディスクの数を示す。特殊なケースは、RAIDZ2 の 4 ディスクプールです。このような状況では、使用可能領域が同じになるため、通常は 2 つのミラー vdev を使用した方がパフォーマンスが向上します。

RAIDZ レベルを使用する際のもう1つの重要な要素は、VMディスクに使用されるZVOLデータセットがどのように動作するかである。各データ・ブロックに対して、プールは少なくとも最小ブロック・サイズのパーティ・データを必要とします。

プールのashift値で定義される。ashiftが12の場合、プールのブロック・サイズは4kである。ZVOLのデフォルトのブロックサイズは8kである。したがって、RAIDZ2では、8kのブロックが書き込まれるごとに、4kのパリティ・ブロックが2つ追加で書き込まれることになり、 $8k + 4k + 4k = 16k$ となります。もちろん、これは単純化したアプローチであり、メタデータや圧縮などはこの例では考慮されていないため、実際の状況は若干異なるでしょう。

この動作は、ZVOLの以下のプロパティをチェックすると確認できる：

- ボルサイズ
- リザベーション（プールがシンプロビジョニングされていない場合）
- 使用される（プールがシンプロビジョニングされ、スナップショットが存在しない場合）

```
# zfs get volsize,refreservation,used <pool>/vm-<vmid>-disk-X
```

volsize は、VM に表示されるディスクのサイズであり、refreservation は、パリティ・データに必要な予期されるスペースを含む、プール上の予約スペースを示す。プールがシン・プロビジョニングされている場合、refreservation は 0 に設定されます。動作を観察するもう1つの方法は、VM 内の使用済みディスク・スペースと使用済みプロパティを比較することです。スナップショットによって値が歪むことに注意してください。

スペースの増加に対抗するには、いくつかの選択肢がある：

- データ/パリティ比を改善するためにvolblocksizeを増やす。
- RAIDZの代わりにミラーvdevを使う
- ashift=9を使用（ブロックサイズは512バイト）

volblocksizeプロパティは、ZVOL作成時にのみ設定できる。デフォルト値はストレージ設定で変更できる。これを行う場合、ゲストはそれに応じて調整する必要があり、ユースケースによっては、書き込み増幅の問題がZFSレイヤーからゲストに移動するだけです。

プールの作成時に ashift=9 を使用すると、下のディスクによってはパフォーマンスが低下することがあり、後で変更することはできません。

ミラー vdev(RAID1、RAID10)は、VM ワークロードに有利な動作をします。ただし、RAIDZ のパフォーマンス特性が許容できるような特殊なニーズや特性を持つ環境を除きます。

### 3.9.4 ZFS dRAID

ZFS dRAID (declustered RAID) では、ホットスペアドライブがRAIDに参加します。その予備容量は予約され、1台のドライブが故障した際の再構築に使用されます。これにより、構成によっては、ドライブ故障時に RAIDZよりも高速な再構築が可能になります。詳細については、OpenZFSの公式ドキュメントを参照してください。<sup>2</sup>

---

#### 注

dRAID は 10-15 台以上のディスクを想定している。RAIDZ セットアップは、ほとんどのユースケースにおいて、より少ないディスク量に適しているはずである。

---

<sup>2</sup>OpenZFS dRAID <https://openzfs.github.io/openzfs-docs/Basic%20Concepts/dRAID%20Howto.html>

## 注

GUI は最小値より 1 台多いディスクを必要とする（つまり dRAID1 は 3 台必要）。スペアディスクも追加されることを期待する。

- dRAID1 または dRAID: 少なくとも2台のディスクが必要。
- dRAID2: 少なくとも3台のディスクが必要。
- dRAID3: 少なくとも4台のディスクが必要。

その他の情報はマニュアルのページをご覧ください:

```
# man zpoolconcepts
```

## スペアとデータ

スペアの数は、ディスク障害が発生した場合に備えて、システムにいくつのディスクを用意しておくかを指示します。デフォルト値は 0 スペアである。スペアがないと、再構築のスピードは上がらない。

データは、冗長グループ内のデバイス数を定義します。デフォルト値は 8 です。ディスク - パリティ - スペアが 8 より小さい場合を除き、小さい数が使用されます。一般に、データ・デバイスの数が少ないほど、IOPS が高くなり、圧縮率が向上し、再シルバーリングが速くなります。データ・デバイスの数を少なく定義すると、プールの利用可能なストレージ容量が減少します。

### 3.9.5 ブートローダー

Proxmox VE は [proxmox-boot-tool](#) を使ってブートローダの設定を管理します。詳細は [Proxmox VE ホストブートローダの章](#)を参照してください。

### 3.9.6 ZFS管理

このセクションでは、一般的なタスクの使用例を紹介します。ZFS自体は本当に強力で、多くのオプション

```
# man  
zpool #  
man zfs
```

を提供しています。ZFSを管理する主なコマンドはzfsとzpoolです。どちらのコマンドにも素晴らしいマニュアル・ページが付属しています：

### 新しいzpoolを作成する

新しいプールを作成するには、少なくとも1つのディスクが必要である。ashiftは、基礎となるディスク

```
zpool create -f -o ashift=12 <プール> <デバイス> # zpool create -f -o ashift=12  
<プール> <デバイス
```

と同じセクタサイズ (ashiftの2乗) 以上でなければならない。

## チップ

プール名は以下の規則に従わなければならない:

- 文字で始まる (a-zまたはA-Z)
- 英数字、-、\_、.、:、または`` (スペース) のみを含む。
- mirror、raidz、draid、spareのいずれかで始まつてはならない。
- ログであつてはならない

圧縮を有効にする (「[ZFSの圧縮](#)」セクションを参照) :

```
# zfs set compression=lz4 <pool>.
```

## RAID-0で新しいプールを作成する

ディスク1枚以上

```
# zpool create -f -o ashift=12 <プール> <デバイス1> <デバイス2>
```

## RAID-1で新しいプールを作成する

最低2ディスク

```
# zpool create -f -o ashift=12 <プール> ミラー <デバイス1> <デバイス2>
```

## RAID-10で新しいプールを作成する

最低4ディスク

```
# zpool create -f -o ashift=12 <pool> mirror <device1> <device2> mirror <'-'> <デバイス3> <デバイス4>
```

## RAIDZ-1で新しいプールを作成する

ディスク3枚以上

```
# zpool create -f -o ashift=12 <プール> raidz1 <デバイス1> <デバイス2> <デバイス3>
```

## RAIDZ-2で新しいプールを作成する

## 最低4ディスク

```
# zpool create -f -o ashift=12 <プール raidz2 <デバイス1 <デバイス2 <デバイス3> ->  
<デバイス4
```

特にRAID-Zモードを使用したい場合は、プールをセットアップする前に、IOPSと帯域幅の期待値の概算を知るために、[ZFS RAIDレベルの考慮事項](#)のセクションをお読みください。

## キャッシュ（L2ARC）付きプールを新規作成する。

セカンドレベルキャッシュとして専用デバイス（パーティション）を使用することで、パフォーマンスを向上させることができる。このようなキャッシュ・デバイスは、ほとんどが静的なデータのランダム・リード・ワークロードに特に役立ちます。これは、実際のストレージとインメモリARCの間の追加キャッシュ層として機能するため、メモリ制約のためにARCを削減しなければならない場合にも役立ちます。

### オンディスク・キャッシュを持つZFSプールを作成する

```
# zpool create -f -o ashift=12 <pool> <device> cache <cache-device>.
```

ここでは、単一の<device>と単一の<cache-device>のみが使用されていますが、[RAIDを使用した新しいプールの作成](#)で示されているように、より多くのデバイスを使用することも可能です。

キャッシュ・デバイスには、ミラー・モディもレイド・モディも存在しない。

キャッシュ・デバイスが読み込み時にエラーを発生した場合、ZFSは透過的にそのリクエストを基礎となるストレージ・レイヤーに迂回させる。

### ログ（ZIL）で新しいプールを作成する

ZFSインテント・ログ（ZIL）専用のドライブやパーティションを使用することも可能です。ZILは主に安全な同期トランザクションを提供するために使用されるため、データベースなどのパフォーマンスが重要なパスや、fsync操作を頻繁に発行するプログラムではよく使用されます。

プールはデフォルトの ZIL ロケーションとして使用され、ZIL IO 負荷を別のデバイスに迂回させることで、トランザクションの待ち時間を短縮すると同時にメイン・プールを解放し、全体的なパフォーマンス向上させることができる。

ディスクを直接、またはパーティションを通してログ・デバイスとして使用する場合は、以下を推奨する：

- 停電保護機能付きの高速SSDを使用すると、コミット・レイテンシがはるかに小さくなるからだ。
- パーティション（またはデバイス全体）には少なくとも数GBを使用するが、インストールされているメモリの半分以上を使用しても、実際の利点は得られない。

### 別のログデバイスを持つZFSプールを作成する

```
# zpool create -f -o ashift=12 <pool> <device> log <log-device>.
```

上記の例では、単一の<device>と単一の<log-device>が使用されていますが、[RAIDで新しいプールを作成する](#)セクションで説明されているように、これを他のRAIDバリエーションと組み合わせることもできます。

また、ログデバイスを複数のデバイスにミラーリングすることもできます。これは主に、1つのログデバイスに障害が発生した場合に、パフォーマンスが直ちに低下しないようにするために便利です。

すべてのログデバイスが故障した場合、ログデバイスが交換されるまで、ZFSメインプール自体が再び使用されます。

## 既存のプールにキャッシュとログを追加する

キャッシュとログがないプールがあっても、いつでもその両方、あるいは片方だけを追加することができる。

例えば、プールの全体的なパフォーマンスを向上させるために、電源損失保護機能を備えた優れたエンタープライズSSDを手に入れたとします。

ログ・デバイスの最大サイズは、搭載されている物理メモリの約半分であるべきなので、ZILはSSDの比較的小さな部分しか占有しない可能性が高く、残りのスペースはキャッシュとして使用できる。

まず、partedまたはgdiskを使ってSSD上にGPTパーティションを2つ作成する必要があります。それからプールに追加する：

### 既存のプールに、独立したログデバイスとセカンドレベルキャッシュの両方を追加する。

```
# zpool add -f <pool> log <device-part1> cache <device-part2>.
```

プール名と2つの/dev/diskを指定して、<pool>、<device-part1>、<device-part2>を再生するだけだ。

パーティションへのパス。

また、ZILとキャッシュを別々に追加することもできる。

### 既存のZFSプールにログデバイスを追加する

```
# zpool add <pool> log <log-device>
```

### 故障したデバイスの変更

```
# zpool replace -f <プール> <旧デバイス> <新デバイス
```

### 失敗したブータブルデバイスを変更する

Proxmox VE のインストール方法によって、systemd-boot を使用するか、proxmox-boot を介して GRUBを使用します。

<sup>3</sup> またはプレーンGRUBをブートローダとして使用します(ホストブートローダを参照)。実行することで確認できます：

```
# proxmox-boot-tool status
```

パーティションテーブルのコピー、GUIDの再発行、ZFSパーティションの交換という最初のステップは同じ

です。新しいディスクからシステムをブート可能にするには、使用するブートローダによって異なる手順が必要です。

```
# sgdisk <健康なブート可能デバイス> -R <新しいデバイス>
# sgdisk -G <新しいデバイス
# zpool replace -f <プール> <旧ZFSパーティション> <新ZFSパーティション>
```

---

### 注

`zpool status -v`コマンドを使用して、新しいディスクの再シルバー化プロセスがどの程度進んでいるかを監視する。

---

<sup>3</sup>Proxmox VE 6.4以降でインストールされたシステム、Proxmox VE 5.4以降でインストールされたEFIシステム

**proxmox-boot-toolを使用:**

```
# proxmox-boot-tool format <新しいディスクのESP>フォーマット  
# proxmox-boot-tool init <新しいディスクのESP> [grub]
```

**注**

ESPはEFIシステムパーティションの略で、バージョン5.4以降、Proxmox VEインストーラがセットアップするブータブルディスクのパーティション#2としてセットアップされます。詳細については、[同期ESPとして使用する新しいパーティションの設定](#)を参照してください。

**注**

proxmox-boot-toolのステータスが現在のディスクがGRUBを使用していることを示している場合、特にセキュアブートが有効になっている場合は、proxmox-boot-tool initにモードとしてgrubを渡すようにしてください！

**プレーンなGRUBで:**

```
# grub-install <新しいディスク
```

**注**

プレーンGRUBはProxmox VE 6.3以前でインストールされ、proxmox-boot-toolを使用して手動で移行されていないシステムでのみ使用されます。

### 3.9.7 電子メール通知の設定

ZFSには、ZFSカーネル・モジュールによって生成されたイベントを監視するイベント・デーモンZEDが付属しています。このデーモンは、プール・エラーのようなZFSイベントに関する電子メールを送信することもできます。新しいZFSパッケージでは、このデーモンは別の`zfs-zed`パッケージとして出荷されており、Proxmox VEではデフォルトで既にインストールされているはずです。

デーモンの設定は、`/etc/zfs/zed.d/zed.rc` ファイルを使用して、お好みのエディターで行うこと

```
ZED_EMAIL_ADDR="root"
```

ができます。電子メール通知に必要な設定は `ZED_EMAIL_ADDR` で、デフォルトでは `root` に設定されています。

Proxmox VE は `root` 宛のメールを `root` ユーザに設定されたメールアドレスに転送することに注意してください。

### 3.9.8 ZFSメモリ使用量の制限

ZFSは、デフォルトでホストメモリの 50 % を Adaptive Replacement Cache (ARC) に使用します。Proxmox

VE 8.1以降の新規インストールでは、ARCの使用制限はインストールされた物理メモリの 10% に設定され、

最大 16GiB までクランプされます。この値は `/etc/modprobe.d/zfs.conf` に書き込まれます。

ARCに十分なメモリを割り当てるることは、IOパフォーマンスにとって非常に重要であるため、慎重に減らしていくこと。一般的な経験則として、少なくとも 2GiB Base + 1GiB/TiB-Storage を割り当てます。たとえば、利用可能なストレージ容量が 8TiB のプールがある場合、ARCには 10GiB のメモリを使用する必要があります。

また、ZFSは最小値を 64 メガバイトに強制している。

に書き込むことで、現在のブートのARC使用制限を変更できます（再起動すると、この変更は再びリセットされます）。

`zfs_arc_max` モジュール・パラメーターを直接指定する：

```
echo "$[10 * 1024*1024*1024]" >/sys/module/zfs/parameters/zfs_arc_max
```

ARC制限を恒久的に変更するには、/etc/modprobeに以下の行を追加する（既にある場合は変更する）。

```
オプション zfs zfs_arc_max=8589934592
```

この設定例では、使用量を8GiB ( $8 * 2^{30}$ ) に制限している。

---

### 重要

希望するzfs\_arc\_max値がzfs\_arc\_min（デフォルトはシステム・メモリの1/32）以下である場合、zfs\_arc\_minも最大でzfs\_arc\_max - 1に設定しない限り、zfs\_arc\_maxは無視されます。

---

```
echo "[8* 1024* 1024* 1024 - 1]">/sys/module/zfs/parameters/zfs_arc_min echo "$[8* 1024* 1024* 1024]">/sys/module/zfs/parameters/zfs_arc_max
```

この設定例では（一時的に）、総メモリ量が256GiBを超えるシステムでの使用量を8GiB ( $8 * 2^{30}$ ) に制限しています。この場合、単にzfs\_arc\_maxを設定するだけでは機能しません。

---

### 重要

ルート・ファイル・システムがZFSの場合、この値が変更されるたびにinitramfsを更新しなければならない：

```
# update-initramfs -u -k all
```

これらの変更を有効にするには、**再起動する必要があります**。

---

## 3.9.9 ZFS上のSWAP

zvol上に作成されたスワップスペースは、サーバーをブロックしたり、外部ストレージへのバックアップを開始するときによく見られるような、高いIO負荷を発生させるなど、いくつかの問題を発生させる可能性があります。

メモリ不足に陥らないよう、十分なメモリを使用することを強くお勧めします。スワップを追加したい場合は、物理ディスク上にパーティションを作成し、スワップデバイスとして使用することをお勧めします。イ

```
# sysctl -w vm.swappiness=10
```

ンストーラの詳細オプションで、この目的のために空き領域を残すことができます。さらに、"swappiness" の値を下げるることもできます。サーバに適した値は 10 です：

スワップを持続させるには、/etc/sysctl.confをお好みのエディターで開き、以下の行を追加する：

```
vm.swappiness = 10
```

表 3.1: Linuxカーネルのスワッピネス・パラメーター値

価値	戦略
vm.swappiness = 0	カーネルがスワップを行うのは、メモリ不足の状態を回避するためだけである。

表3.1: (続き)

価値	戦略
vm.swappiness = 1	スワッピングを完全に無効にすることなく、最低限のスワッピングを行う。
vm.swappiness = 10	パフォーマンスを向上させるために、この値が推奨されることがある。 システムに十分なメモリがある場合。
vm.swappiness = 60	デフォルト値。
vm.swappiness = 100	カーネルは積極的にスワップを行う。

### 3.9.10 暗号化されたZFSデータセット

#### 警告



Proxmox VE のネイティブ ZFS 暗号化は実験的なものです。既知の制限と問題には、暗号化されたデータセットでのリプリケーション、スナップショットやZVOL使用時のチェックサムエラーがあります。 <sup>a</sup>スナップショットやZVOL使用時のチェックサムエラーなどがあります。 <sup>b</sup>

<sup>a</sup>[https://bugzilla.proxmox.com/show\\_bug.cgi?id=2350](https://bugzilla.proxmox.com/show_bug.cgi?id=2350)  
<sup>b</sup><https://github.com/openzfs/zfs/issues/11688>

ZFS on Linux バージョン 0.8.0 では、データセットのネイティブ暗号化がサポートされました。以前の ZFS on Linux バージョンからアップグレードすると、プールごとに暗号化機能を有効にできます：

```
# zpool get feature@暗号化タンク
名前 不動産          値値          ソース
タン 暗号化機能      使用不能      ローカル
ク

# zpool set feature@encryption=enabled

# zpool get feature@暗号化タンク
名前プロパティ          値値          ソース
```

#### 警告



暗号化されたデータセットを持つプールからGRUBを使ってブートすることは現在サポートされておらず、ブート時に暗号化されたデータセットのロックを自動的に解除するための限定的なサポートしかありません。暗号化をサポートしていない古いバージョンのZFSでは、保存されたデータを

復号化することはできません。

---

## 注

起動後に手動でストレージデータセットのロックを解除するか、`zfs load-key`に起動時のロック解除に必要なキー素材を渡すカスタムユニットを書くことを推奨します。

---

## 警告

 本番データの暗号化を有効にする前に、バックアップ手順を確立し、テストすること。関連付けられた鍵マテリアル／パスフレーズ／鍵ファイルを紛失した場合、暗号化されたデータへのアクセスは不可能となる。

---

暗号化はデータセット/zvols の作成時に設定する必要があり、デフォルトで子データセットに継承されます

```
# zfs create -o encryption=on -o keyformat=passphrase tank/encrypted_data
```

パスフレーズを入力してください:

パスフレーズを再入力する:

```
# pvesm add zfspool encrypted_zfs -pool tank/encrypted_data
```

。例えば、暗号化データセットtank/encrypted\_dataを作成し、Proxmox VEのストレージとして設定するには、以下のコマンドを実行します:

このストレージ上に作成されたすべてのゲストボリューム/ディスクは、親データセットの共有鍵マテリアルで暗号化される。

ストレージを実際に使用するには、関連するキーマテリアルをロードし、データセットをマウントする必要がある

```
# zfs mount -l tank/encrypted_data
```

「tank/encrypted\_data」にパスフレーズを入力する:

。これは

キーロケーションを設定することで、パスフレーズを入力する代わりに（ランダムな）キーファイルを使用することも可能です。

```
# dd if=/dev/urandom of=/path/to/keyfile bs=32 count=1
```

```
# zfs change-key -o keyformat=raw -o keylocation=file:///path/to/keyfile -o  
タンク/暗号化データ
```

およびkeyformatプロパティを、作成時または既存のデータセット上でzfs change-keyを使用して設定します:



### 警告

キーファイルを使用する場合は、不正アクセスや偶発的な紛失に対してキーファイルを保護するために特別な注意が必要です。キーファイルがなければ、平文データにアクセスすることはできない!

暗号化データセットの下に作成されたゲストボリュームは、それに応じてencryptionrootプロパティが設定されます。キー・マテリアルはencryptionrootごとに一度だけロードする必要があり、その下にあるすべての暗号化データセットで利用できるようになります。

詳細と高度な使用法については、`man zfs` の `encryptionroot`、`encryption`、`keylocation`、`keyformat`、および `keystatus proper-ties`、`zfs load-key`、`zfs unload-key`、および `zfs change-key` コマンド、および暗号化セクションを参照してください。

### 3.9.11 ZFSの圧縮

データセットで圧縮が有効になっている場合、ZFSは新しいブロックをすべて圧縮してから書き込み、読み込み時に解凍しようとする。既に存在するデータは遡って圧縮されることはありません。

で圧縮を有効にできる：

```
# zfs set compression=<algorithm> <dataset>.
```

`lz4` アルゴリズムを使うことを推奨する。`lzb` や `gzip-N` (`N` は 1 (最速) から 9 (最高の圧縮率) までの整数) のような他のアルゴリズムも利用可能である。アルゴリズムと圧縮可能なデータによっては、圧縮を有効にすることで I/O 性能が向上することもある。

でいつでも圧縮を無効にできる：

```
# zfs set compression=off <データセット
```

繰り返しますが、この変更の影響を受けるのは新しいブロックだけです。

### 3.9.12 ZFS専用デバイス

バージョン0.8.0以降、ZFSは特殊デバイスをサポートしている。プール内の特別なデバイスは、メタデータ、重複排除テーブル、およびオプションで小さなファイルブロックを格納するために使用される。

特殊デバイスは、メタデータの変更が多い、回転速度の遅いハードディスクで構成されるプールの速度向上させることができる。例えば、大量のファイルを作成、更新、または削除するようなワークロードでは、特別なデバイスがあると便利です。ZFSデータセットは、特別なデバイスに小さなファイル全体を保存するように設定することもでき、パフォーマンスをさらに向上させることができます。特殊デバイスには高速SSDを使用します。

#### 重要

特殊デバイスの冗長性は、プールの冗長性と一致すべきである。

デバイスはプール全体の障害点となる。

#### 警告

特別なデバイスをプールに追加した場合、元に戻すことはできません！

**特別なデバイスとRAID-1でプールを作成する：**

```
# zpool create -f -o ashift=12 <pool> mirror <device1> <device2> special <'ミラー <デバイス3> <デバイス4>'
```

**既存のRAID-1プールに特別なデバイスを追加する：**

```
# zpool add <pool> special mirror <device1> <device2>
```

ZFSデータセットは、`special_small_blocks=<size>`プロパティを公開しています。`size`には、小さなファイル・ブロックを特殊デバイスに保存しないようにする0か、512Bから1Mの範囲の2のべき乗を指

定します。このプロパティを設定すると、sizeより小さい新しいファイル・ブロックが特殊デバイス上に割り当てられる。

**重要**

special\_small\_blocksの値がrecordsize以上の場合（デフォルトはデータセットの128K）に書き込まれると、**すべてのデータが特別なデバイスに書き込まれるので注意が必要だ！**

プールでspecial\_small\_blocksプロパティを設定すると、すべての子ZFSデータセットでそのプロパティのデフォルト値が変更されます（たとえば、プール内のすべてのコンテナは、スマール・ファイル・ブロックを選択します）。

プール全体で4Kブロックより小さいファイルすべてにオプトインする:

```
# zfs set special_small_blocks=4K <pool>.
```

単一のデータセットに対して小さなファイルブロックを選択する:

```
# zfs set special_small_blocks=4K <pool>/<filesystem>.
```

単一のデータセットに対する小さなファイルブロックからオプトアウトする:

```
# zfs set special_small_blocks=0 <pool>/<filesystem>.
```

### 3.9.13 ZFSプールの特徴

ZFSのオンディスク・フォーマットへの変更は、メジャー・バージョンの変更の間にのみ行われ、**features**を通して指定されます。すべての機能と一般的な仕組みは、`zpool-features` (`manpage`に詳しく書かれています)。新しい機能を有効にすると、古いバージョンのZFSではプールがインポートできなくなる可能性があるため、管理者がプールで `zpool upgrade` を実行して、積極的に行う必要があります (`zpool-upgrade(8)` manページを参照)。

新機能のいずれかを使う必要がない限り、それらを有効にすることにプラス面はない。実際、新機能を有効にすることにはマイナス面もある:

- GRUBでZFSの互換性のない実装のため、GRUBを使用して起動するZFS上のrootを持つシステムは、`rpool`で新しい機能がアクティブになると起動できなくなります。
- 古いカーネルで起動した場合、システムはアップグレードされたプールをインポートすることができません。
- 起動しないシステムを修復するために古いProxmox VE ISOを起動しても、同様に機能しません。

**重要**

システムがまだ GRUB で起動している場合は、`rpool` をアップグレードしないでください。これには Proxmox VE 5.4 より前にインストールされたシステムや、レガシー BIOS ブートで起動しているシステムも含まれます ([ブートローダの判別方法を参照](#))。

ZFSプールの新機能を有効にする:

```
# zpool upgrade <pool>
```

## 3.10 BTRFS



### 警告

BTRFS の統合は現在 Proxmox VE のテクノロジープレビューです。

BTRFSは、Linuxカーネルがネイティブにサポートする最新のコピーオンライトファイルシステムで、スナップショット、ビルトインRAID、データとメタデータのチェックサムによるセルフヒーリングなどの機能を実装しています。Proxmox VE 7.0から、BTRFSはルートファイルシステムのオプション選択として導入されました。

### BTRFSの一般的な利点

- メインシステムのセットアップは、従来のext4ベースのセットアップとほぼ同じ
- スナップ写真
- ファイルシステムレベルでのデータ圧縮
- コピー・オン・ライト・クローン
- RAID0、RAID1、RAID10
- データ破損からの保護
- セルフ・ヒーリング
- Linuxカーネルがネイティブにサポート
- ...

### 注意事項

- RAIDレベル5/6は実験的で危険である。

### 3.10.1 ルートファイルシステムとしてのインストール

Proxmox VEインストーラを使用してインストールする場合、ルートファイルシステムにBTRFSを選択できます。インストール時にRAIDタイプを選択する必要があります：

RAID0

「ストライピング」とも呼ばれる。このようなボリュームの容量は、すべてのディ

スクの容量の合計です。しかし、RAID0 は冗長性を追加しないため、1 台のドライブが故障するとボリュームは使用できなくなります。

## RAID1

「ミラーリング」とも呼ばれる。データはすべてのディスクに同じように書き込まれる。このモードでは、同じサイズのディスクが少なくとも 2 台必要です。その結果、容量はディスク 1 台分となります。

RAID10 RAID0 と RAID1 を組み合わせたもの。最低 4 台のディスクが必要。

インストーラーは自動的にディスクをパーティション分割し、追加のサブボリュームを /var/lib/pve/local- に作成します。

Proxmox VE ツールでこれを使用するために、インストーラは以下の設定エントリを /etc/pve/storage.cfg:

```
dir:ローカル
  パス /var/lib/vz
  コンテンツ iso, vztmpl, バックアップ
  プ無効化

btrfs: ローカルbtrfs
  パス /var/lib/pve/local-btrfs
  コンテンツ iso, vztmpl, バックアップ, イメージ, ルートディレクトリ
```

これは、デフォルトのローカル・ストレージを明示的に無効にして、追加サブボリューム上のBTRFS固有のストレージ・エントリーを優先します。

btrfsコマンドは、BTRFSファイルシステムの設定と管理に使用されます。インストール後、以下のコマンドで追加サブボリュームをすべて一覧表示します：

```
# btrfs サブボリュームリスト /
ID 256 gen 6 トップレベル 5 パス var/lib/pve/local-btrfs
```

### 3.10.2 BTRFSアドミニストレーション

このセクションでは、一般的なタスクの使用例を紹介します。

#### BTRFSファイルシステムの作成

BTRFSファイルシステムを作成するには、`mkfs.btrfs`を使用する。`-d`パラメーターと`-m`パラメーターは、それぞれメタデータとデータのプロファイルを設定するのに使われる。オプションの`-L` パラメーターで、ラベルを設定できる。

一般に、以下のモードがサポートされている：シングル、`raid0`、`raid1`、`raid10`

。#シングルモードで/dev/sdb1にd My-Storageというラベルを付けてBTRFSファ

イルシステムを作成する：

あるいは、2つのパーティション`/dev/sdb1`と`/dev/sdc1`にRAID1を作成する：

```
# mkfs.btrfs -m raid1 -d raid1 -L My-Storage /dev/sdb1 /dev/sdc1
```

## BTRFSファイルシステムのマウント

新しいファイルシステムは、例えば手動でマウントすることができる：

```
# mkdir /my-storage  
# mount /dev/sdb /my-storage
```

BTRFSも他のマウントポイントと同様に/etc/fstabに追加し、ブート時に自動的にマウントすることができます。ブロック・デバイス・パスの使用は避け、mkfs.btrfs コマンドが出力する UUID 値を使用することをお勧めします。

例えば、こうだ：

## ファイル /etc/fstab

```
# ... 他のマウントポイントは簡潔にするため割愛した  
  
# UUID=e2c0c3ff-2114-4f54-b767-3a203e49f6f3 /my-storage btrfs defaults  
0 0
```

## チップ

UUIDが利用できなくなった場合は、blkidツールを使ってブロックデバイスのすべてのプロパティをリストアップすることができます。

その後、最初のマウントを実行することができる：

```
マウント /my-storage
```

次のリブート後は、ブート時にシステムによって自動的に実行されます。

## Proxmox VEへのBTRFSファイルシステムの追加

既存のBTRFSファイルシステムをProxmox VEに追加するには、Webインターフェイスを使用するか、CLIなどを

```
pvesm add btrfs my-storage --path /my-storage  
使用します：
```

## サブボリュームの作成

サブボリュームを作成すると、そのサブボリュームはBTRFSファイルシステム内のパスにリンクされ、通常のディレクトリとして表示されます。

```
# btrfs subvolume create /some/path
```

その後、/some/pathは通常のディレクトリのように動作する。

## サブボリュームの削除

rmdirで削除されるディレクトリとは逆に、サブボリュームはbtrfsコマンドで削除するために空である

```
# btrfs subvolume delete /some/path  
必要はありません。
```

## サブボリュームのスナップショットの作成

BTRFSは実際にはスナップショットと通常のサブボリュームを区別しないため、スナップショットの作成は

```
# btrfs subvolume snapshot -r /some/path /a/new/path
```

サブボリュームの任意のコピーの作成と見なすこともできます。慣例では、Proxmox VEはゲストディスクまたはサブボリュームのスナップショットを作成するときに読み取り専用フラグを使用しますが、このフラグは後で変更することもできます。

これにより、/some/path上のサブボリュームの読み取り専用「クローン」が/a/new/pathに作成される。今後/some/pathを変更すると、変更前のデータがコピーされる。

読み取り専用 (-r) オプションを省略すると、両方のサブボリュームが書き込み可能になる。

## 圧縮を有効にする

デフォルトでは、BTRFSはデータを圧縮しません。圧縮を有効にするには、compressマウント・オプションを追加します。すでに書き込まれたデータは、後から圧縮されないことに注意してください。

```
UUID=<ルート・ファイル・システムのUUID> / btrfsのデフォルト 0 1
```

デフォルトでは、rootfsは/etc/fstabに以下のようにリストされる:

デフォルトにcompress=zstd、compress=lzo、compress=zlibを追加するだけだ。

上記のように:

```
UUID=<ルートファイルシステムのUUID> / btrfs defaults,compress=zstd 0 1
```

この変更は再起動後に有効になります。

## スペースの使用状況の確認

古典的なdfツールは、BTRFSのセットアップによっては紛らわしい値を出力することがあります。より正確な見積もりには

```
# btrfs fi usage /my-storage
```

btrfsファイルシステムの使用法 /PATHコマンドなど:

## 3.11 Proxmoxノード管理

Proxmox VEノード管理ツール(pvenode)を使用すると、ノード固有の設定と再ソースを制御できます。

現在pvenodeでは、ノードの説明を設定したり、ノードのゲストに対して様々なバルク操作を実行したり、ノードのタスク履歴を表示したり、ノードのSSL証明書を管理したりすることができます。

### 3.11.1 ウェイクオンLAN

Wake-on-LAN (WoL)は、マジックパケットを送信することで、ネットワーク内でスリープしているコンピュータの電源を入れることができます。少なくとも1つのNICがこの機能をサポートし、コンピュータのファームウェア (BIOS/UEFI) 設定でそれぞれのオプションを有効にする必要があります。オプション名は *Enable*

```
etool <interface> | grep Wake-on
```

*Wake-on-Lan* から *Power On By PCIE Device* まで様々で、不明な場合はマザーボードのベンダーマニュアルを確認してください:

```
pvenode wakeonlan <ノード>
```

pvenodeでは、WoL経由でクラスタのスリープ・メンバーを起こすことができます:

これは、wakeonlan プロパティから取得した <node> の MAC アドレスを含む WoL magic パケットを UDP ポート 9 にブロードキャストします。ノード固有の wakeonlan プロパティは以下のコマンドで設定できます:

```
pvenode config set -wakeonlan XX:XX:XX:XX:XX:XX
```

WoLパケットを送信するインターフェースは、デフォルトルートから決定される。以下のコマンドでbind-

```
pvenode config set -wakeonlan XX:XX:XX:XX:XX:XX,bind-interface=<インターフェース名>を設定します。
```

interfaceを設定することで上書きできる:

WoLパケット送信時に使用されるブロードキャストアドレス（デフォルト255.255.255.255）は、以

```
pvenode config set -wakeonlan XX:XX:XX:XX:XX:XX,broadcast-address=<←!  
ブロードキャストアドレス
```

下のコマンドを使用して明示的にブロードキャストアドレスを設定することでさらに変更できる:

### 3.11.2 タスク履歴

バックアップジョブの失敗など、サーバーの問題をトラブルシューティングする際、以前に実行したタスクのログがあると役立つことがあります。Proxmox VEでは、`pvenode task`コマンドでノードのタスク履歴にアクセスできます。

`list`サブコマンドを使用すると、ノードの終了したタスクのフィルタリングされたリストを取得できます。例えば、エラーで終了したVM 100に関連するタスクのリストを取得するには、次のようにコマンドを実行します:

```
pvenodeタスクリスト --エラー --vmid 100
```

タスクのログは、そのUPIDを使って印刷できる:

```
pvenode task log UPID:pve1:00010D94:001CA6EA:6124E1B9:vzdump:100:root@pam:
```

### 3.11.3 一括ゲスト電源管理

多数のVM/コンテナがある場合、ゲストの起動と停止は`pvenode`の`startall`と`stopall`サブコマンドで一括操作できます。デフォルトでは、`pvenode startall`はブート時に自動的に起動するように設定されたVM/コンテナのみを起動します（[仮想マシンの自動起動とシャットダウン](#)参照）。両コマンドには`--vms`オプションもあり、停止/起動されるゲストを指定されたVMIDに制限します。

例えば、VM100、101、102を起動するには、オンブートが設定されているかどうかに関係なく、次のようにする:

```
pvenode startall --vms 100,101,102 --force
```

これらのゲスト（および実行中の他のゲスト）を停止するには、コマンドを使用する：

プロベノード・ストップオール

---

### 注

stopallコマンドはまずクリーンシャットダウンを試み、すべてのゲストが正常にシャットダウンされるか、オーバーライド可能なタイムアウト（デフォルトでは3分）が切れるまで待機します。これが発生し、force-stopパラメーターが明示的に0（false）に設定されないと、まだ実行中のすべての仮想ゲストがハード停止されます。

---

### 3.11.4 ファースト・ゲスト・ブート・ディレイ

VM/コンテナがNFSサーバなどの起動の遅い外部リソースに依存している場合、Proxmox VEが起動してから、自動起動に設定されている最初のVM/コンテナが起動するまでの遅延をノードごとに設定することもできます（[仮想マシンの自動起動とシャットダウンを参照](#)）。

これを実現するには、以下のように設定します（ここで10は秒単位の遅延を表します）：

```
pvenode config set --startall-onboot-delay 10
```

### 3.11.5 ゲストの一括移行

アップグレードの状況で、すべてのゲストをあるノードから別のノードに移行する必要がある場合、pvenodeは一括移行のためのmigrateallサブコマンドも提供します。デフォルトでは、このコマンドはシステム上のすべてのゲストをターゲットノードに移行します。しかし、ゲストのセットだけを移行するように設定することもできます。

例えば、ローカル・ディスクのライブ・マイグレーションを有効にして、VM 100、101、102をノードpve2

```
pvenode migrateall pve2 --vms 100,101,102 --ローカルディスクあり  
に移行するには、次のように実行します：
```

## 3.12 証明書管理

### 3.12.1 クラスタ内通信証明書

各Proxmox VEクラスタはデフォルトで独自の(自己署名)認証局(CA)を作成し、前述のCAによって署名される各ノードの証明書を生成します。これらの証明書はクラスタのpveproxyサービスおよびSPICEが使用されている場合のシェル/コンソール機能との暗号化通信に使用されます。

CA証明書と鍵は[Proxmox Cluster File System \(pmxcfs\)](#)に保存されます。

### 3.12.2 APIおよびWeb GUI用の証明書

REST API および Web GUI は、各ノードで実行される pveproxy サービスによって提供され

ます。pveproxy が使用する証明書には以下のオプションがあります：

1. デフォルトでは、/etc/pve/nodes/NODENAME/pve-ssl.pem にあるノード固有の証明書が使用されます。この証明書はクラスタ CA によって署名されているため、ブラウザや OS からは自動的に信頼されません。
2. 外部から提供された証明書（商用CAによって署名されたものなど）を使用する。
3. ACME（Let'sEncrypt）を使用して、自動更新付きの信頼できる証明書を取得してください。

オプション 2 と 3 では、/etc/pve/local/pveproxy-ssl.pem ファイル（および /etc/pve/local/pveproxy-ssl.pem ファイル）を使用します。  
パスワードは不要）が使われる。

## 注

`etc/pve/local` は `/etc/pve/nodes/NODENAME` へのノード固有のシンボリックリンクであることに注意してください。

証明書は Proxmox VE Node 管理コマンドで管理します (`pvenode(1)` man-page を参照)。

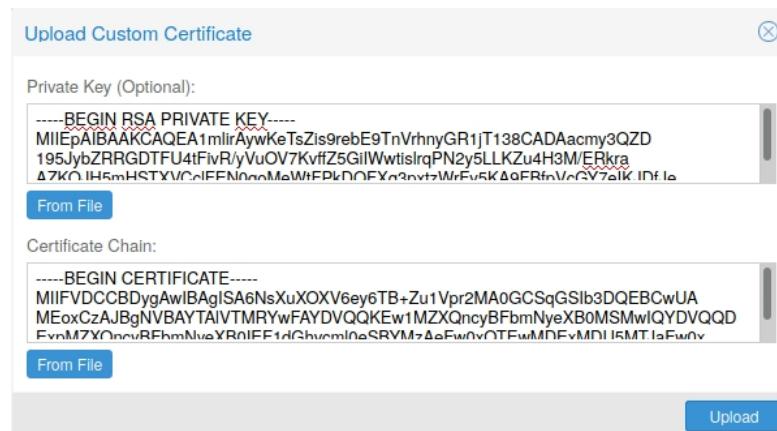
## 警告

の自動生成されたノード証明書ファイルを置き換えるか、手動で変更したりしないでください。

`/etc/pve/local/pve-ssl.pem` および `/etc/pve/local/pve-ssl.key`、または `/etc/pve/pve-root-ca.pem` および `/etc/pve/priv/pve-root-ca.key` にあるクルスタ CA ファイル。

### 3.12.3 カスタム証明書のアップロード

Proxmox VE ノードに使用する証明書をすでにお持ちの場合は、Web インターフェイスから簡単にアップロードできます。



証明書キー・ファイルが提供される場合、パスワードで保護されてはならないことに注意。

### 3.12.4 Let's Encrypt (ACME) による信頼できる証明書

Proxmox VE には、自動証明書管理環境 **ACME** プロトコルの実装が含まれており、Proxmox VE の管理者は、Let's Encrypt のような ACME プロバイダを使用して、最新のオペレーティングシステムや Web ブラウザで受け入れられ、信頼される TLS 証明書を簡単にセットアップすることができます。

現在、実装されている 2 つの ACME エンドポイントは、[Let's Encrypt \(LE\)](#) の本番環境とそのステージング環境です。私たちのACMEクライアントは、組み込みのウェブサーバーを使用したhttp-01チャレンジの検証と、[acme.sh](#)が行うすべてのDNS APIエンドポイントをサポートするDNSプラグインを使用したdns-01チャレンジの検証をサポートしています。

## ACMEアカウント

The screenshot shows a 'Register Account' dialog box. It has fields for Name (set to 'default'), ACME Directory (set to 'Let's Encrypt V2'), Terms of Service (link to 'https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf'), Accept TOS (checkbox checked), and E-Mail (set to 'admin@example.com'). A 'Register' button is at the bottom.

使用するエンドポイントには、クラスタごとに ACME アカウントを登録する必要があります。そのアカウントに使用される電子メールアドレスは、ACMEエンドポイントからの更新期限や同様の通知の連絡先として機能します。

ACME アカウントの登録と停止は、Web インターフェース Datacenter -> ACME または pvenode

```
pvenode acme アカウント登録 アカウント名 mail@example.com  
コマンドラインツールを使用して行うことができます。
```

### チップ

レート制限のため、実験や初めてACMEを使う場合はLEステージングを使うべきである。

## ACMEプラグイン

ACMEプラグインのタスクは、あなた、ひいてはあなたの運用下にあるProxmox VEクラスタがドメインの真の所有者であることを自動的に検証することです。これは、自動証明書管理の基礎となるビルディングブロックです。

例えば、`http-01`では、ウェブサーバーがドメインを管理していることを証明するために、特定のコンテンツを含むファイルを提供します。技術的な制限や、レコードのアドレスが公共のインターネットから到達できない場合など、これが不可能なこともある。`dns-01`チャレンジはこのような場合に使用できる。このチャレンジは、ドメインのゾーンに特定のDNSレコードを作成することで実行されます。

The screenshot shows the 'Accounts' section with two entries: 'default' and 'staging'. Below it is the 'Challenge Plugins' section, which lists a single entry: 'pdns-example.com' under the 'Plugin' column and 'pdns' under the 'API' column.

Proxmox VEは、これらのチャレンジタイプの両方をサポートしており、Webインターフェイスの Datacenter -> ACMEでプラグインを設定するか、`pvenode acme plugin add`コマンドを使用してプラグインを設定することができます。

ACMEプラグインの設定は`/etc/pve/priv/acme/plugins.cfg`に保存されます。プラグインはクラスタ内のすべてのノードで使用できます。

## ノード・ドメイン

各ドメインはノード固有です。新しいドメインエントリを追加したり、既存のドメインエントリを管理したりするには、ノード -> 証明書、または`pvenode config`コマンドを使用します。

The dialog box has the following fields:

- Challenge Type: DNS
- Plugin: pdns-example.com
- Domain: prod1.pve.example.com

At the bottom are 'Help' and 'Create' buttons.

ノードの希望するドメインを設定し、希望するACMEアカウントが選択されていることを確認した後、Webインターフェイス上で新しい証明書を注文することができます。成功すると、インターフェイスは10秒後にリロードされます。

更新は自動的に行われます。

### 3.12.5 ACME HTTPチャレンジ・プラグイン

ポート80で生成された組み込みのウェブサーバを経由してhttp-01チャレンジを検証するために、常に暗黙的に設定されたスタンドアロンプラグインがあります。

## 注

スタンダードアロンという名前は、サードパーティのサービスを使わずに、このプラグインだけでバリデーションを行うことができるという意味です。そのため、このプラグインはクラスタノードでも動作します。

Let's Encrypt ACMEでの証明書管理に使用するには、いくつかの前提条件があります。

- アカウントを登録するには、Let's EncryptのToSに同意する必要があります。
- ノードのポート80はインターネットから到達可能である必要がある。
- ポート80には他のリスナーがいてはならない。
- 要求された（サブ）ドメインはノードのパブリックIPに解決する必要がある。

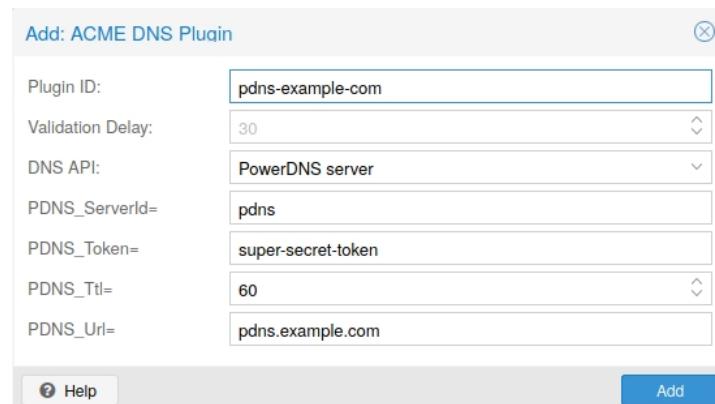
### 3.12.6 ACME DNS API チャレンジ・プラグイン

http-01メソッドによる検証のための外部アクセスが不可能であるか、または望まれないシステムでは、dns-01検証メソッドを使用することが可能である。この検証方法では、API経由でTXTレコードをプロビジョニングできるDNSサーバーが必要です。

#### 検証のためのACME DNS APIの設定

Proxmox VEは、acme.shプロジェクトで開発されたDNSプラグインを再利用しています。<sup>4</sup>プロジェクトのために開発されたDNSプラグインを使用します。特定のAPIの設定の詳細については、そのドキュメントを参照してください。

DNS APIを使用して新しいプラグインを設定する最も簡単な方法は、Webインターフェース（Datacenter -> ACME）を使用することです。



チャレンジ・タイプとしてDNSを選択する。次に、APIプロバイダーを選択し、そのAPIを介してアカウント

にアクセスするためのクレデンシャルデータを入力することができます。

---

## チップ

プロバイダのAPI認証情報の取得に関する詳細情報については、acme.sh [How to use DNS API](#) wikiを参照してください。

---

多くのDNSプロバイダとAPIエンドポイントがあるため、Proxmox VEはいくつかのプロバイダの認証情報用のフォームを自動的に生成します。その他のプロバイダについては、大きなテキストエリアが表示されますので、そこにすべての認証情報のKEY=VALUEペアをコピーしてください。

<sup>4</sup>acme.sh <https://github.com/acmesh-official/acme.sh>

## CNAMEエイリアスによるDNS検証

プライマリ/リアルDNSがAPI経由のプロビジョニングをサポートしていない場合、特別なエイリアスマードを使用して、別のドメイン/DNSサーバーで検証を処理することができます。の永続的なCNAMEレコードを手動で設定します。

`_acme-challenge.domain1.example`を`_acme-challenge.domain2.example`にポインティングし、Proxmox VEノード構成ファイルのエイリアス・プロパティを`domain2.example`に設定して、`domain2.example`のDNSサーバーが`domain1.example`のすべてのチャレンジを検証できるようにします。

## プラグインの組み合わせ

`http-01`と`dns-01`の検証を組み合わせることは、要件/DNSプロビジョニング機能が異なる複数のDNS経由でノードに到達可能な場合に可能です。ドメインごとに異なるプラグインインスタンスを指定することで、複数のプロバイダーやインスタンスのDNS APIを混在させることも可能です。

---

### チップ

同じサービスに複数のドメインでアクセスすることは複雑さを増すので、可能であれば避けるべきである。

---

### 3.12.7 ACME証明書の自動更新

ノードがACME提供の証明書で正常に構成されている場合（`pvenode`経由またはGUI経由）、証明書は`pve-daily-update.service`によって自動的に更新されます。現在のところ、証明書の有効期限が既に切れている場合、または今後30日以内に期限が切れる場合に更新が試みられます。

### 3.12.8 pvenodeを使用したACMEの例

例 Let's Encrypt証明書を使用する`pvenode`呼び出しのサンプル

```
root@proxmox:~# pvenode acme account register default mail@example.invalid
Directory エンドポイント:
0) Let's Encrypt v2 (https://acme-v02.api.letsencrypt.org/directory)
1) Let's Encrypt v2 ステージング (https://acme-staging-v02.api.letsencrypt.org/
   ←
   ディレクトリ)
2) カスタム

選択を入力します: 1

利用規約: https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf ←'
上記の条件に同意しますか? [y|N]y
...
タスクOK
root@proxmox:~# pvenode config set --acme domains=example.invalid
root@proxmox:~# pvenode acme cert order
ACMEアカウント詳細の読み込み ACME注
文の発注
...
ステータスは「有効」!
```

## すべてのドメインが有効!

• • •

## 証明書のダウンロード

## pveproxy 証明書と鍵の設定 pveproxy の再

起動

タスクOK

## 例ドメインを検証するためのOVH APIの設定

注

アカウント登録の手順は、どのプラグインを使用しても同じですので、ここでは繰り返しません。

注

OVH AKおよびOVH ASは、OVH APIドキュメントに従ってOVHから取得する必要があります。

ます、あなたとProxmox VEがAPIにアクセスできるように、すべての情報を取得する必要があります。

(バリデーションURLを開き、指示に従ってアプリケーションキーを←でリンクする。  
アカウント/コンシューマー・キー)

これでACMEプラグインをセットアップできます:

```
root@proxmox:~# pvenode acme plugin add dns example_plugin --api ovh --data='
    /パス/to/api_token
root@proxmox:~# pvenode acme plugin config example plugin

キー          値
アピ          ovh
              ; data OVH_AK=XXXXXXXXXXXXXX
              ; OVH_AS=YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
              ; OVH_CK=ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
              ; ダイジェスト 867fcf556363calbea866863093fcab83edf47a1
              ; plugin example plugin
              ;
タイプ        dns
              ;
```

最後に、証明書を取得したいドメインを設定し、そのドメインの証明書を注文することができる：

```
root@proxmox:~# pvenode config set -acmedomain0 example.proxmox.com,plugin='
    example_plugin
root@proxmox:~# pvenode acme cert order ACME
```

#### アカウント詳細のロード

#### ACMEの発注

注文URL: <https://acme-staging-v02.api.letsencrypt.org/acme/order> ←
 /11111111/22222222

<https://acme-staging-v02.api.letsencrypt.org/acme/authz-v3/333333>。

example.proxmox.comの検証は保留中です！

[Wed Apr 22 09:25:30 CEST 2020] OVH endpoint: ovh-eu の使用

[Wed Apr 22 09:25:30 CEST 2020] 認証の確認 [Wed Apr 22

09:25:30 CEST 2020] Consumer key は OK です。

[Wed Apr 22 09:25:31 CEST 2020] レコードの追加

[Wed Apr 22 09:25:32 CEST 2020] 追加、10秒スリープ。TXTレコードを追加: \_acme-challenge.example.proxmox.comトリガー検証

5秒間スリープ 状態は「有効」！

[Wed Apr 22 09:25:48 CEST 2020] OVH endpoint: ovh-eu の使用

[Wed Apr 22 09:25:48 CEST 2020] 認証の確認 [Wed Apr 22

09:25:48 CEST 2020] Consumer key は OK です。

TXTレコードを削除する: \_acme-challenge.example.proxmox.com

すべてのドメインが有効！

CSRの作成 注文状況の確認

注文の準備が整いました!

### 証明書のダウンロード

pveproxy 証明書と鍵の設定 pveproxy の再起動  
タスクOK

### 例ステージングディレクトリから通常のACMEディレクトリへの切り替え

アカウントのACMEディレクトリの変更はサポートされていませんが、Proxmox VEは複数のアカウントをサポートしているため、本番用(信頼済み)ACMEディレクトリをエンドポイントとして新しいアカウントを作成することができます。また、ステージングアカウントを非アクティブにして再作成することもできます。

#### 例pvenodeを使用して、デフォルトのACMEアカウントをステージングからディレクトリに変更する。

```
root@proxmox:~# pvenode acme account deactivate default  
アカウントファイルの名前を「/etc/pve/priv/acme/default」から「/etc/pve/priv/←」に変更。
```

```
    acme/_deactivated_default_4'
```

タスクOK

```
root@proxmox:~# pvenode acme account register default example@proxmox.com  
Directory エンドポイント:
```

- 0) Let's Encrypt V2 (<https://acme-v02.api.letsencrypt.org/directory>)
- 1) Let's Encrypt V2 ステージング (<https://acme-staging-v02.api.letsencrypt.org/>  
←  
ディレクトリ)
- 2) カスタム

選択を入力します: 0

利用規約: <https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf>

上記の条件に同意しますか? [y|N] y

...

タスクOK

## 3.13 ホスト・ブートローダー

Proxmox VEは現在、インストーラで選択されたディスクセットアップに応じて、2つのブートローダのいずれかを

使用します。

ZFS をルートファイルシステムとしてインストールした EFI システムでは、セキュアブートが有効になっていない限り、`systemd-boot` が使用されます。それ以外のすべてのデプロイメントでは、標準の GRUB ブートローダを使用します（これは通常、Debian 上にインストールされたシステムにも当てはまります）。

### 3.13.1 インストーラーが使用するパーティション方式

Proxmox VE のインストーラは、インストール用に選択されたすべてのディスクに

3つのパーティションを作成します。作成されるパーティションは以下のとおりで

す：

- 1 MB の BIOS ブートパーティション（`gdisk` タイプ EF02）

- 512 MB EFI システムパーティション (ESP、gdisk タイプ EF00)
- 設定されたhdsizeパラメータまたは選択されたストレージタイプで使用される残りのスペースにまたがる第3のパーティション

ルートファイルシステムとしてZFSを使用するシステムは、512MBのEFIシステムパーティションに保存されたカーネルとinitrdイメージで起動します。レガシーBIOSシステム、およびセキュアブートが有効なEFIシステムにはGRUBが使用され、セキュアブートのないEFIシステムにはsystemd-bootが使用される。どちらもインストールされ、ESPを指すように設定される。

BIOSモードのGRUB (--target i386-pc)は、GRUBで起動したすべてのシステムの選択したディスクのBIOSブートパーティションにインストールされます。<sup>5</sup>

### 3.13.2 proxmox-boot-toolでESPの内容を同期させる

proxmox-boot-toolは、EFIシステムパーティションの内容を適切に設定・同期しておくためのユーティリティです。特定のカーネルバージョンをすべてのESPにコピーし、vfatフォーマットされたESPからブートするようにそれぞれのブートローダを設定します。ルートファイルシステムとしてのZFSのコンテキストでは、これは、GRUBのZFS実装にも存在するサブセットや、別の小さなブートプールを作成する代わりに、ルートプールですべてのオプション機能を使用できることを意味します。<sup>6</sup>

冗長性のあるセットアップでは、インストーラによってすべてのディスクがESPでパーティション設定されます。これにより、最初のブートデバイスが故障したり、BIOSが特定のディスクからしか起動できない場合でも、システムが確実に起動します。

通常の操作では、ESPはマウントされたままにはならない。これにより、システムクラッシュ時にvfatフォーマットされたESPのファイルシステム破損を防ぐことができ、プライマリブートデバイスが故障した場合に/etc/fstabを手動で変更する必要がなくなる。

proxmox-boot-toolは以下のタスクを処理します：

- 新しいパーティションのフォーマットと設定
- 新しいカーネルイメージとinitrdイメージを、リストされたすべてのESPにコピーし、設定する。
- カーネル・アップグレードやその他のメンテナンス・タスクの際に、コンフィギュレーションを同期させる。
- 同期されているカーネルバージョンのリストを管理する
- 特定のカーネルバージョンをブートするようにブートローダを設定する（ピン留め）。

を実行すると、現在設定されているESPとその状態を見ることができる：

```
# proxmox-boot-tool status
```

### 同期ESPとして使用する新しいパーティションの設定

パーティションを同期されたESPとしてフォーマットして初期化するには、たとえばrpoolの故障したvdevを交換した後や、同期メカニズム以前の既存のシステムを接続する場合、`proxmox-kerne`の`proxmox-boot-tool`を使用できます。

<sup>5</sup>これらはすべて、ext4またはxfs上のrootによるインストールと、非EFIシステム上のZFS上のrootによるインストールです。

<sup>6</sup>GRUBでZFSをrootで起動する <https://github.com/zfsonlinux/zfs/wiki/Debian-Stretch-Root-on-ZFS>

**警告**

formatコマンドは<パーティション>をフォーマットするので、正しいデバイス/パーティションを渡してください！

例えば、空のパーティション/dev/sda2をESPとしてフォーマットするには、以下を実行する：

```
# proxmox-boot-tool format /dev/sda2
```

dev/sda2にある既存のマウントされていないESPをProxmox VEのカーネル更新同期メカニズムに含めるようにセットアップするには、以下を使用します：

```
# proxmox-boot-tool init /dev/sda2
```

または

```
# proxmox-boot-tool init /dev/sda2 grub
```

セキュアブートをサポートするためなど、systemd-bootの代わりにGRUBで初期化を強制する。

その後、/etc/kernel/proxmox-boot-uuidsに、新しく追加されたパーティションのUUIDの行が追加されるはずです。initコマンドは、設定されているすべてのESPのリフレッシュも自動的に行います。

### すべてのESPで設定を更新する

すべてのブータブルカーネルをコピーして設定し、すべてのESPを/etc/kernel/proxmox-boot-uuidsにリストしておくには、次のようにします。

同期が取れていれば、あとは走るだけだ：

```
# proxmox-boot-tool refresh
```

(rootでext4またはxfsのupdate-grubシステムを実行するのと同じことだ)。

これは、カーネルコマンドラインに変更を加えたり、すべてのカーネルとinitrdsを同期させたい場合に必要である。

### 注

update-initramfsとapt（必要な場合）の両方が、自動的にリフレッシュをトリガーする。

### proxmox-boot-toolが考慮するカーネルバージョン

以下のカーネル・バージョンがデフォルトで設定されている：

- 現在実行中のカーネル
- パッケージ更新時に新しくインストールされるバージョン
- すでにインストールされている2つの最新カーネル
- 該当する場合は、最後から2番目のカーネルシリーズの最新バージョン（例：5.0、5.3）。
- 手動で選択したカーネル

## 手動でカーネルをブート可能にする

特定のカーネルと initrd イメージをブート可能なカーネルのリストに追加したい場合は、 proxmox-boot-tool kernel add を使ってください。

たとえば、以下を実行して、ABIバージョン5.0.15-1-pveのカーネルを、すべてのESPにインストールし

```
# proxmox-boot-tool kernel add 5.0.15-1-pve
```

て同期しておくカーネルのリストに追加する：

proxmox-boot-tool kernel list は現在起動用に選択されている全てのカーネルバージョンを一覧表示します：

```
# proxmox-boot-tool kernel
list 手動で選択したカーネル:
5.0.15-1-pve
```

自動的に選択されたカーネル： 5.0.12-  
1-pve  
4.15.18-18-pve

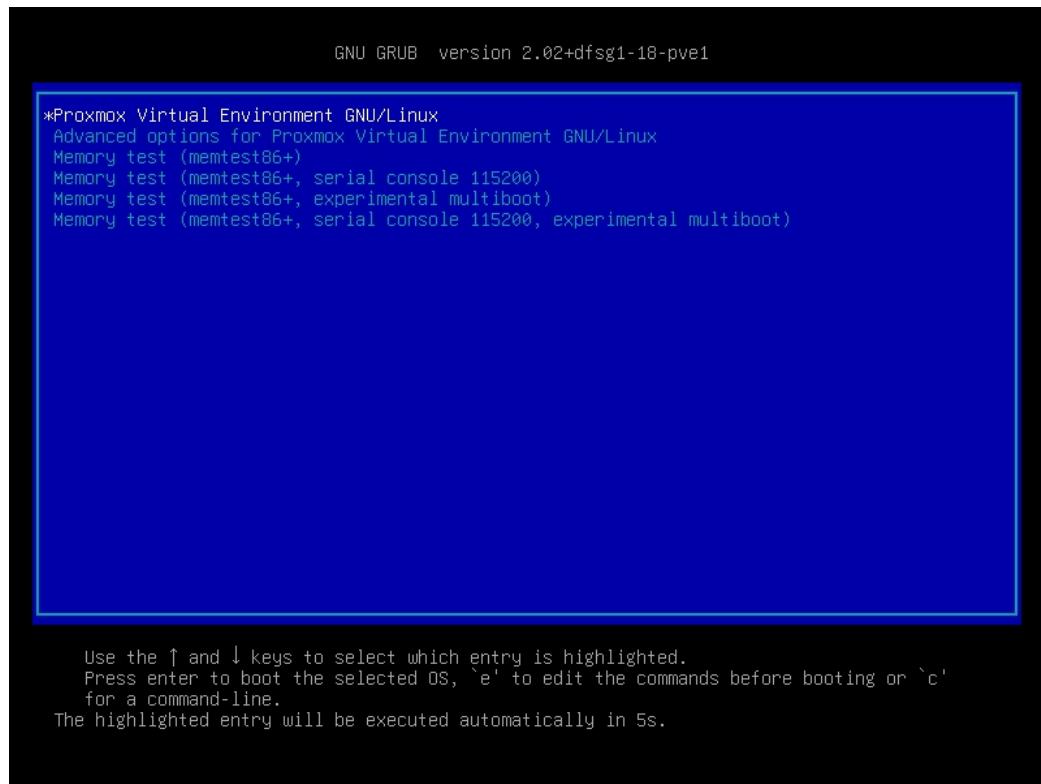
proxmox-boot-tool kernel remove を実行して、手動で選択したカーネルのリストからカーネルを削

```
# proxmox-boot-tool kernel remove 5.0.15-1-pve
除してください：
```

### 注

手動でカーネルを追加または削除した後、すべてのEFIシステムパーティション（ESP）を更新するため  
に proxmox-boot-tool refresh を実行する必要があります。

### 3.13.3 使用されるブートローダーの決定



どのブートローダが使用されているかを判断する最も簡単で確実な方法は、Proxmox VEノードのブートプロセスを観察することです。

GRUBの青いボックスか、白地に黒のシンプルなsystemd-bootが表示される。



実行中のシステムからブートローダーを判断するのは、100%正確ではないかもしれない。最も安全な方法は、以下のコマンドを実行することだ：

```
# efibootmgr -v
```

EFI変数がサポートされていないというメッセージを返した場合、GRUBはBIOS/Legacyモードで

使用されます。出力に以下のような行が含まれる場合、GRUBはUEFIモードで使用されます。  
Boot0005\* proxmox [...] File(\proxmox\grubx64.efi)

```
Boot0006* Linuxブートマネージ [...] File( / / / / / / / )  
ヤー  
)
```

出力に以下のような行がある場合、systemd-bootが使用される。

走ることによって：

```
# proxmox-boot-tool status
```

proxmox-boot-toolが設定されているかどうかを調べることができる。

### 3.13.4 GRUB

GRUBは、長年Linuxシステムを起動するための事実上の標準であり、非常によく文書化されている。

<sup>7</sup>.

---

<sup>7</sup>GRUB マニュアル <https://www.gnu.org/software/grub/manual/grub/grub.html>

## 構成

GRUB コンフィギュレーションの変更は、`/etc/default/grub` の `defaults` ファイルまたは `/etc/default/grub.d` の `config snip-pets` を介して行われる：<sup>8</sup>

```
# update-grub
```

## 3.13.5 システムブート

`systemd-boot` は軽量の EFI ブートローダです。インストールされている EFI サービスパーティション (ESP) からカーネルと `initrd` イメージを直接読み込みます。ESP から直接カーネルを読み込む主な利点は、ストレージにアクセスするためのドライバを再実装する必要がないことです。Proxmox VE [proxmox-bootツール](#) は、ESP 上の設定を同期させるために使用されます。

## 構成

`systemd-boot` は、EFI システムパーティション (ESP) のルートディレクトリにある `loader/loader.conf` ファイルで設定します。詳細は `loader.conf(5)` man ページを参照してください。

各ブートローダ・エントリは、`loader/entries` ディレクトリにあるそれ自身のファイルに置かれます。

`entry.conf` の例は以下のようになる（/は ESP のルートを指す）：

```
タイトル Proxmoxバージ  
ョン5.0.15-1-pve  
オプション root=ZFS=rpool/ROOT/pve-1 boot=zfs  
リナックス /EFI/proxmox/5.0.15-1-pve/vmlinuz-5.0.15-1-pve  
initrd /EFI/proxmox/5.0.15-1-pve/initrd.img-5.0.15-1-pve
```

## 3.13.6 カーネル・コマンドラインの編集

カーネルのコマンドラインは、使用するブートローダに応じて、以下の箇所で変更できます：

### GRUB

カーネルコマンドラインは、次のファイルの変数 `GRUB_CMDLINE_LINUX_DEFAULT` に置く必要がある。

`/etc/default/grub` に追加します。`update-grub` を実行すると、その内容が `/boot/grub/g` のすべて

## システムブート

カーネルコマンドラインは/etc/kernel/cmdlineに1行で配置する必要があります。変更を適用するには、`proxmox-boot-tool refresh`を実行して、`loader/entries/proxmox-* .conf`のすべての設定ファイルのオプション行として設定します。

カーネルパラメータの完全なリストは、<https://www.kernel.org/doc/html/v<YOUR-KERNEL-VERSION>/admin-guide/kernel-parameters.html> にあります。<YOUR-KERNEL-VERSION> を major.minor に置き換えてください。

---

<sup>8</sup>`proxmox-boot-tool` を使っているシステムは、`update-grub` 時に `proxmox-boot-tool refresh` を呼び出します。

例えば、バージョン6.5ベースのカーネルの場合、URLは次のようにになります:

<https://www.kernel.org/doc/html/-v6.5/admin-guide/kernel-parameters.html>

カーネルのバージョンは、ウェブインターフェイス (*Node → Summary*) で確認するか、次のコマンドを実行して

```
# uname -r
```

出力の最初の2つの数字を使用する。

### 3.13.7 次のブートのためにカーネルバージョンを上書きする。

現在デフォルトでないカーネルを選択するには、以下の方法がある:

- ブートプロセスの最初に表示されるブートローダーメニューを使用する。
- `proxmox-boot-tool`を使用して、システムをカーネルバージョンに一度だけ、または永久に（ピンガリセットされるまで）固定します。

これは、新しいカーネルバージョンとハードウェア間の非互換性を回避するのに役立つはずだ。

---

#### 注

このようなピンはできるだけ早く取り外し、最新のカーネルのセキュリティパッチがすべてシステムに適用されるようにすべきである。

---

```
# proxmox-boot-tool kernel pin 5.15.30-1-pve
```

例えばバージョン5.15.30-1-pveを恒久的に選択してブートするには、次のように実行する:

#### チップ

もしあなたのシステムが同期に `proxmox-boot-tool` を使用しないのであれば、最後に `proxmox-boot-tool` のリフレッシュコールをスキップすることもできます。

---

また、次のシステム起動時のみ起動するカーネルバージョンを設定することもできます。これは例えば、最初にバージョンを固定する原因となつた問題が更新されたカーネルによって解決されたかどうかをテストするのに便利です:

```
# proxmox-boot-tool kernel pin 5.15.30-1-pve --next-boot
```

---

ピン留めされたバージョン設定を削除するには、`unpin`サブコマンドを使用する：

```
# proxmox-boot-tool kernel unpin
```

`unpin`には`--next-boot`オプションもあるが、これは`--next-boot`で設定された`pin`されたバージョンをクリアするために使われる。これはブート時にすでに自動的に行われるため、手動で起動してもほとんど意味がない。

ピン留めされたバージョンを設定またはクリアした後は、`refresh`サブコマンドを実行してESPのコンテンツと設定を同期させる必要もある。

---

### チップ

`proxmox-boot-tool`で管理されているシステムでは、対話的にツールを呼び出すと、自動的にそうするようにプロンプトが表示されます。

```
# proxmox-boot-tool refresh
```

### 3.13.8 セキュアブート

Proxmox VE 8.1 以降、セキュアブートは、署名されたパッケージと以下の統合により、すぐにサポートされます。

proxmox-boot-tool。

セキュアブートの動作には以下のパッケージが必要です。proxmox-secure-boot-support' メタパッケージを使って一度にインストールできます。

- shim-signed (マイクロソフトによって署名されたshimブートローダ)
- shim-helpers-amd64-signed (ウォールバックブートローダおよびMOKManager、Proxmoxにより署名済み)
- grub-efi-amd64-signed (GRUB EFI ブートローダ、Proxmoxにより署名済み)
- proxmox-kernel-6.X.Y-Z-pve-signed (Proxmoxにより署名されたカーネルイメージ)

他のブートローダはセキュア・ブート・コード署名の対象外であるため、GRUBのみがブートローダとしてサポートされています。

Proxmox VEを新規インストールすると、自動的に上記のパッケージがすべて含まれます。

セキュアブートの仕組みやセットアップのカスタマイズ方法の詳細については、[wiki](#)をご覧ください。

#### 既存のインストールをセキュアブートに切り替える

##### 警告

これは、正しく行わないと、場合によっては起動不能なインストールにつながる可能性があります。ホストを再インストールすると、セキュアブートが利用可能であれば、余分な操作なしに自動的にセットアップが行われます。Proxmox VEホストのバックアップが動作し、十分にテストされていることを確認してください！

Proxmox VE をゼロから再インストールしなくても、必要に応じて既存の UEFI インストールをセキュアブートに切り替えることができます。

まず、システムがすべて最新であることを確認してください。次に proxmox-secure-boot-support をイ

ンストールする。GRUBは、デフォルトのシム経由で起動するために必要なEFIブートエントリーを自動的に作成します。

## システムドブート

systemd-bootをブートローダとして使用する場合（[使用するブートローダを決定する](#)を参照）、追加のセットアップが必要です。これはProxmox VEをZFS-on-rootでインストールした場合のみです。

後者をチェックするには

```
# findmnt /
```

ホストが本当にルート・ファイルシステムとしてZFSを使用している場合、FSTYPE列には`zfs`が含まれるはずである：

```
ターゲット・ソース          FSTYPEオプショ  
ース      /rpool/ROOT/pve-1  zfs    rw,relatime,xattr,noacl,ケースセンシティブ
```

次に、適切なESP（EFIシステムパーティション）を見つけなければならない。これは、`lsblk`コマンドを次のように実行する：

```
# lsblk -o +FSTYPE
```

出力は次のようになるはずだ:

名前	MAJ:MIN	RM	マウントポイント	fstype	
エス	8:0	0	32G ディスク0枚		
&#x251c;&#x2500;sda1	8:1	0	1007K 0 パート		
&#x251c;&#x2500;sda2	8:2	0	512M 0 パート		ブイファット
&#x2514;&#x2500;sda3	8:3	0	31.5G 0 パート		zfs_member
セロトニン	8:16	0	32G 0 ディスク	1007K 0 パート	
&#x251c;&#x2500;sdb1	8:17	0			
&#x251c;&#x2500;sdb2	8:18	0	512M 0 パート		ブイファット
&#x2514;&#x2500;sdb3	8:19	0	31.5G 0 パート		zfs_member

この場合、パーティションsda2とsdb2がターゲットである。パーティションのサイズは512Mで、FSTYPEはvfatです。

これらのパーティションは、`proxmox-boot-tool` を使って GRUB で起動するように適切にセットアップ

```
# proxmox-boot-tool init /dev/sda2 grub
```

する必要があります。このコマンド（例として sda2 を使用）は、個々の ESP に対して個別に実行する必要があります:

その後、以下のコマンドを実行して、セットアップをチェックすることができる:

```
# efibootmgr -v
```

このリストには、次のようなエントリーが含まれているはずだ:

```
[..]
Boot0009*           HD(2,GPT,...,0x800,0x100000)/File(\EFI\proxmox\<-->
    proxmox
    shimx64.efi)
[...]
```

## 注

古い `systemd-boot` ブートローダは維持されますが、GRUB が優先されます。こうすることで、セキュアブートモードで GRUB を使って起動しても何らかの理由でうまくいかない場合でも、セキュアブートをオフにした `systemd-boot` を使ってシステムを起動することができます。

これでホストを再起動し、UEFI ファームウェアセットアップユーティリティでセキュアブートを有効にすること

ができる。

再起動すると、`proxmox` という新しいエントリが UEFI ファームウェアブートメニューで選択できるようになっていて、署名済みの EFI shim を使って起動するようになっているはずです。

何らかの理由で UEFI ブートメニューに `proxmox` エントリが見つからない場合、(ファームウェアでサポートされている) カスタムブートエントリとしてファイル `\EFI\proxmox\shimx64.efi` を追加することで、手動で追加してみることができます。

---

### 注

いくつかの UEFI ファームウェアは再起動時に `proxmox` ブートオプションを落とすことが知られています。これは `proxmox` のブートエントリがディスク上の GRUB インストールを指していて、ディスク自体がブートオプションになっていない場合に起こります。可能であれば、UEFI ファームウェアセットアップユーティリティでディスクをブートオプションとして追加し、`proxmox-boot-tool` をもう一度実行してみてください。

---

## チップ

カスタム鍵を登録するには、付属のセキュアブートwikiページを参照のこと。

## セキュアブートでのDKMS/サードパーティーモジュールの使用

セキュアブートが有効になっているシステムでは、カーネルは信頼できる鍵で署名されていないモジュールのロードを拒否します。カーネル・パッケージに同梱されているデフォルトのモジュール・セットは、カーネル・イメージに埋め込まれたエフェメラル・キーで署名されており、特定のバージョンのカーネル・イメージで信頼されています。

DKMS または手動でビルドしたモジュールなど、他のモジュールをロードするには、セキュアブートスタイルに信頼された鍵で署名する必要があります。これを実現する最も簡単な方法は、`mokutil` を使用してそれらをマシンオーナーキー (MOK) として登録することです。

`dkms`ツールは自動的にキーペアと証明書を`/var/lib/dkms/mok.key`と`/var/lib/dkms/mok.pub`をビルドし、インストールするカーネルモジュールに署名する

```
# openssl x509 -in /var/lib/dkms/mok.pub -noout -text
```

そして、以下のコマンドを使ってシステムに登録する：

```
# mokutil --import /var/lib/dkms/mok.pub  
input password:  
もう一度パスワードを入力してください:
```

`mokutil`コマンドは(一時的な)パスワードを2回要求します。このパスワードは、プロセスの次のステップでもう1回入力する必要があります！システムを再起動すると、自動的にMOKManager EFIバイナリが起動し、`mokutil`を使用して登録を開始するときに選択したパスワードを使用して、キー/証明書を検証し、登録を確認することができます。その後、カーネルは（登録されたMOKで署名された）DKMSでビルドされたモジュールのロードを許可するはずです。必要であれば、MOKはカスタムEFIバイナリやカーネルイメージに署名するためにも使用できます。

同じ手順を、DKMSで管理されていないカスタム/サードパーティ・モジュールにも使用できるが、その場合は鍵/証明書の生成と署名の手順を手動で行う必要がある。

## 3.14 カーネル・サムページ・マージング (KSM)

KSM (Kernel Samepage Merging) はLinuxカーネルが提供するオプションのメモリ重複排除機能で、Proxmox VEではデフォルトで有効になっています。KSMは、物理メモリページの範囲をスキャンして同一のコンテンツを探し、それらにマップされている仮想ページを特定することで機能します。同一のページが見つかった場合、対応する仮想ページはすべて同じ物理ページを指すようにマッピングし直され、古いページは解放される。仮想ページは "コピー온ライト"としてマークされ、それへの書き込みはメモリの新しい領域に書き込まれ、共有された物理ページはそのまま残される。

### 3.14.1 KSMの意味

KSMは、仮想化環境におけるメモリ使用量を最適化することができる。類似のオペレーティング・システムやワークロードを実行する複数のVMは、多くの共通メモリ・ページを共有する可能性があるからだ。

しかし、KSMはメモリ使用量を削減できる一方で、VMをサイドチャネル攻撃にさらす可能性があるため、セキュリティ上のリスクも伴う。研究では、KSMの特定の特性を悪用することで、同じホスト上の2つ目のVMを介して、実行中のVMに関する情報を推測することが可能であることが示されている。

したがって、ホスティングサービスを提供するためにProxmox VEを使用している場合は、ユーザーに追加のセキュリティを提供するために、KSMを無効にすることを検討する必要があります。さらに、KSMを無効にすることが法律で義務付けられている場合がありますので、お住まいの国の規制を確認する必要があります。

### 3.14.2 KSMを無効にする

KSMがアクティブかどうかを確認するには、以下の出力をチェックすることができる：

```
# systemctl status ksmtuned
```

もしそうなら、すぐに無効化できる：

```
# systemctl disable --now ksmtuned
```

最後に、現在マージされているすべてのページをアンマージするには、以下を実行する：

```
# echo 2 > /sys/kernel/mm/ksm/run
```

## 第4章

# グラフィカル・ユーザー・インターフェース

Proxmox VEはシンプルです。別の管理ツールをインストールする必要はなく、すべてウェブブラウザ（最新のFirefoxまたはGoogle Chromeが望ましい）で行うことができます。ゲストコンソールへのアクセスには、組み込みのHTML5コンソールが使用されます。別 の方法として、SPICEを使用することもできます。

Proxmoxクラスタファイルシステム(pmxcfs)を使用しているため、どのノードに接続してもクラスタ全体を管理できます。各ノードはクラスタ全体を管理できます。専用のマネージャノードは必要ありません。

ウェブベースの管理インターフェイスは、最新のブラウザで使用できます。Proxmox VEがモバイルデバイスからの接続を検出すると、よりシンプルなタッチベースのユーザーインターフェースにリダイレクトされます。

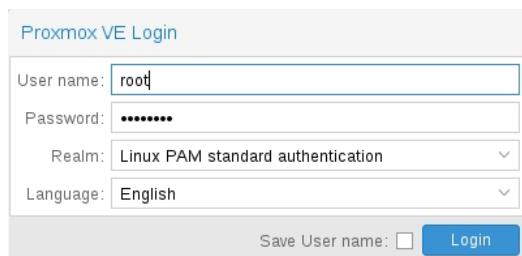
ウェブ・インターフェイスには、<https://youripaddress:8006>（デフォルト・ログインはrootで、パスワードはインストール・プロセス中に指定されます）を介してアクセスできます。

### 4.1 特徴

- Proxmox VEクラスタのシームレスな統合と管理
- リソースの動的更新のためのAJAX技術
- SSL暗号化(https)によるすべての仮想マシンとコンテナへのセキュアなアクセス
- 数百からおそらく数千のVMを処理できる、高速な検索駆動インターフェース
- 安全なHTML5コンソールまたはSPICE

- すべてのオブジェクト（VM、ストレージ、ノードなど）のロールベースの権限管理
- 複数の認証ソースのサポート（ローカル、MS ADS、LDAPなど）
- 二要素認証（OATH、Yubikey）
- ExtJS 7.x JavaScriptフレームワークがベース

## 4.2 ログイン



サーバーに接続すると、最初にログインウィンドウが表示されます。Proxmox VEは様々な認証バックエンド(Realm)をサポートしており、ここで言語を選択できます。GUIは20以上の言語に翻訳されています。

### 注

下部のチェックボックスを選択すると、クライアント側でユーザー名を保存できます。これにより、次回ログイン時に入力の手間が省けます。

## 4.3 GUIの概要

Start Time ↓	End Time	Node	User name	Description	Status
Jul 15 20:35:56	Jul 15 20:35:57	prod1	admin@pve	VM 9000 - Destroy	OK
Jul 15 20:35:53	Jul 15 20:35:53	prod1	admin@pve	VM 9000 - Create	OK
Jul 15 20:19:51	Jul 15 20:19:51	prod1	admin@pve	VM 99999 - Destroy	OK
Jul 15 20:19:45	Jul 15 20:19:45	prod1	admin@pve	VM 99999 - Create	OK
Jul 15 20:19:02	Jul 15 20:19:03	prod1	admin@pve	VM 9000 - Destroy	OK

Proxmox VEのユーザーインターフェースは、4つの領域で構成されています。

ヘッダー	一番上。ステータス情報を表示し、最も重要なアクションのボタンを含む。リソースツリー
Content	PanelCenter領域。選択されたオブジェクトは、ここに設定オプションとステータスを表示します。
ログパネル	下部。最近のタスクのログエントリが表示されます。これらのログエントリをダブルクリックして詳細を表示したり、実行中のタスクを中止したりできます。

## 注

リソースツリーやログパネルのサイズを縮小・拡大したり、ログパネルを完全に非表示にすることができます。これは、小さなディスプレイで作業していて、他のコンテンツを表示するためのスペースが欲しい場合に役立ちます。

### 4.3.1 ヘッダー

左上にはProxmoxのロゴがあります。その隣には現在稼働中のProxmox VEのバージョンが表示されます。近くの検索バーでは、特定のオブジェクト（VM、コンテナ、ノード、...）を検索できます。これはリソースツリーでオブジェクトを選択するよりも速い場合があります。

ヘッダーの右側には4つのボタンがある：

ドキュメント 参照ドキュメントを表示する新しいブラウザウィンドウをCreate VM

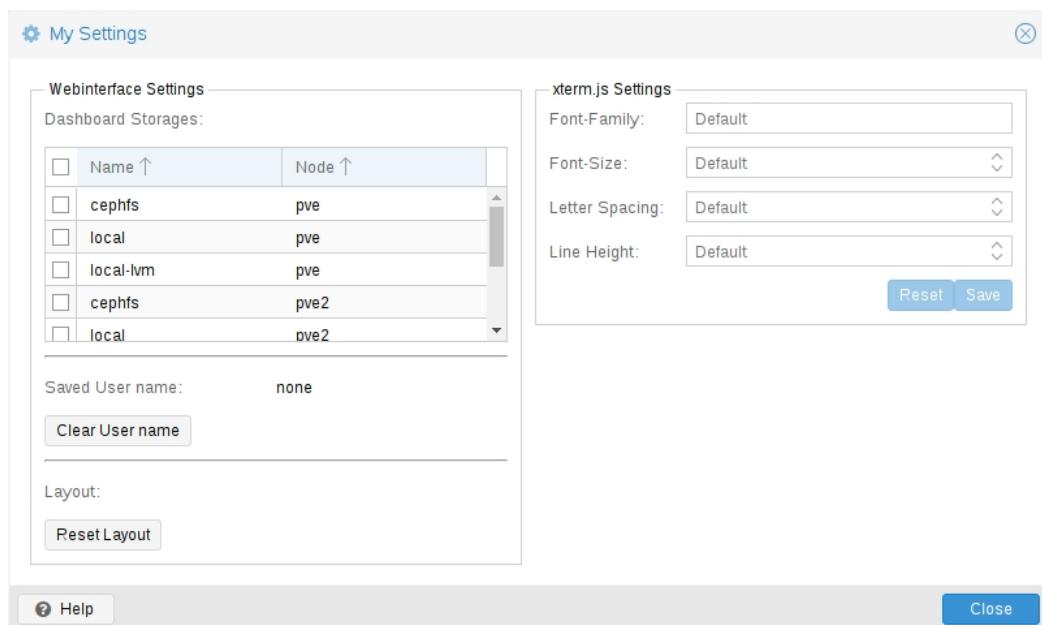
仮想マシンの作成ウィザードを

CT作成コンテナ作成ウィザードを開く。

ユーザーメニュー 現在ログインしているユーザーのIDが表示され、クリックするとユーザー固有のオプションメニューが開きます。

ユーザーメニューには、ローカルUI設定を提供する*My Settings*ダイアログがあります。その下には、TFA（二要素認証）とパスワードセルフサービスのショートカットがあります。また、言語とカラーテーマを変更するオプションもあります。最後に、メニューの一番下にログアウトオプションがあります。

### 4.3.2 マイ・セッティング



*My Settings*] ウィンドウでは、ローカルに保存された設定を行うことができます。これにはダッシュボード・ストレージが含まれ、データセンター・サマリーで表示される合計量にカウントする特定のストレージを有効または無効にすることができます。どのストレージにもチェックが入っていない場合は、すべてのストレージの合計になります。

ダッシュボード設定の下には、保存されたユーザー名とそれをクリアするボタン、GUIの各レイアウトをデフォルトにリセットするボタンがあります。

右側には*xterm.js Settings*があります。これらには以下のオプションがあります:

**Font-Family** *xterm.js*で使用するフォント（例: Arial）。

**Font-Size** 使用するフォントサイズ。

**Letter Spacing** テキストの文字間隔を増減

**Line Height** 行の高さの絶対

### 4.3.3 リソースツリー

これはメインのナビゲーションツリーです。ツリーの上部では、いくつかの定義済みビューを選択することができ、下のツリーの構造を変更することができます。デフォルトのビューはサーバービューで、以下のオ

プロジェクトタイプが表示されます：

Datacenterクラスタ全体の設定を含みます（すべてのノードに関連します）。

Node クラスタ内のホストを表し、ここでゲストが実行され

る。ゲストVM、コンテナ、テンプレートを表します。

ストレージ データ・ストレージ。

プールゲストをプールでグループ化し、管理を簡単にすることができます。

以下のビュータイプが利用可能です：

Server View あらゆる種類のオブジェクトをノード別にフォル

タビュー オブジェクトタイプ別にグループ化された、あら

ゆる種類のオブジェクトをPool View VM とコンテナをプール

#### 4.3.4 ログパネル

ログ・パネルの主な目的は、クラスタで現在何が行われているかを表示することです。新しいVMを作成するようなアクションはバックグラウンドで実行され、このようなバックグラウンドジョブをタスクと呼びます。

このようなタスクからの出力はすべて別のログファイルに保存される。タスクログエントリーをダブルクリックするだけで、そのログを見るることができます。そこで実行中のタスクを中止することもできる。

ここではすべてのクラスタノードから最新のタスクが表示されます。そのため、誰かが他のクラスタノードで作業しているのをリアルタイムで確認することができます。

---

##### 注

リストを短くするために、古いタスクと終了したタスクをログパネルから削除します。しかし、ノードパネル内のタスク履歴でそれらのタスクを見つけることができます。

---

短時間で実行されるアクションの中には、すべてのクラスタ・メンバーにログを送信するだけのものもあります。これらのメッセージは  
クラスターログパネル。

---

#### 4.4 コンテンツパネル

リソースツリーから項目を選択すると、対応するオブジェクトの設定とステータス情報がコンテンツパネルに表示されます。以下のセクションでは、この機能の概要を説明します。より詳細な情報については、リファレンス・ドキュメントの対応する章を参照してください。

## 4.4.1 データセンター

The screenshot shows the Proxmox VE Datacenter interface. On the left, there's a sidebar with navigation links like Server View, Datacenter (prod-eu-central), Summary, Cluster, Ceph, Options, Storage, Backup, Replication, Permissions, Users, Groups, Pools, Roles, Authentication, HA, Firewall, and Support. The main area has a search bar at the top. Below it is a table titled 'Search' with columns: Type, Description, Disk usage..., Memory us..., CPU usage, and Uptime. The table lists various resources: prod1 (lxc, 510 (CT510)), prod2 (node, prod1, prod2, prod3), prod3 (node, prod3), development (pool), qemu (101 (win10), 501 (VM 501), 100 (VM 100)), storage (cephfs, cp, iso, local, local-lvm), and HA (storage). One row for 'storage cp (prod3)' is highlighted with a blue background. At the bottom, there's a 'Tasks' section with a table showing cluster logs.

Start Time ↓	End Time	Node	User name	Description	Status
Jul 15 20:35:56	Jul 15 20:35:57	prod1	admin@pve	VM 9000 - Destroy	OK
Jul 15 20:35:53	Jul 15 20:35:53	prod1	admin@pve	VM 9000 - Create	OK
Jul 15 20:19:51	Jul 15 20:19:51	prod1	admin@pve	VM 99999 - Destroy	OK
Jul 15 20:19:45	Jul 15 20:19:45	prod1	admin@pve	VM 99999 - Create	OK
Jul 15 20:19:02	Jul 15 20:19:03	prod1	admin@pve	VM 9000 - Destroy	OK

データセンター・レベルでは、クラスタ全体の設定と情報にアクセスできます。

- 検索:** ノード、VM、コンテナ、ストレージデバイス、プールをクラスタ全体で検索します。
- Summary:** クラスタの健全性とリソースの使用状況の概要を示します。
- クラスタ:** クラスタの作成または参加に必要な機能と情報を提供します。
- オプション:** クラスタ全体のデフォルト設定の表示と管理。
- ストレージ:** クラスタ・ストレージを管理するためのインターフェースを提供する。
- バックアップ:** バックアップジョブをスケジュールします。これはクラスタ全体で動作するので、スケジューリング時にVM/コンテナがクラスタ上のどこにあるかは関係ありません。
- レプリケーション:** レプリケーションジョブの表示と管理。

- **パーミッション:** ユーザー、グループ、APIトークンのパーミッション、LDAP、MS-AD、Two-Factor認証を管理します。
- **HA:** Proxmox VE High Availability を管理します。

- **ACME:** サーバーノードにACME（Let's Encrypt）証明書をセットアップする。
- **ファイアウォール:** Proxmoxファイアウォールのクラスタ全体の設定とテンプレートの作成。
- **メトリックサーバー:** Proxmox VEの外部メトリックサーバーを定義します。
- **通知:** Proxmox VEの通知動作とターゲットを設定します。
- **サポート:** サポート契約に関する情報を表示します。

#### 4.4.2 ノード

The screenshot shows the Proxmox VE 6.0-4 interface. On the left, a sidebar lists the Datacenter (prod-eu-central) with nodes prod1, prod2, prod3, and development. The main panel is focused on node prod2, which has been up for 11 days and 7 minutes. It provides a summary of system health, including CPU usage (1.92% of 4 CPU(s)), load average (0.50.0.28.0.21), RAM usage (25.12% of 7.88 GiB of 31.38 GiB), HD space (6.20% of 2.49 GiB of 40.11 GiB), and swap usage (0.00% of 0 B of 8.00 GiB). Below this, it shows CPU(s) (4 x Intel(R) Xeon(R) CPU E3-1220 v3 @ 3.10GHz (1 Socket)), Kernel Version (Linux 5.0.15-1-pve #1 SMP PVE 5.0.15-1 (Wed, 03 Jul 2019 10:51:57 +0200)), and PVE Manager Version (pve-manager/6.0-4/79fa8d98). A detailed CPU usage graph is shown, with a sharp peak around 19:57:00. The bottom section displays a table of recent tasks, showing operations like VM creation and destruction on node prod1 by user admin@pve.

Start Time	End Time	Node	User name	Description	Status
Jul 15 20:35:56	Jul 15 20:35:57	prod1	admin@pve	VM 9000 - Destroy	OK
Jul 15 20:35:53	Jul 15 20:35:53	prod1	admin@pve	VM 9000 - Create	OK
Jul 15 20:19:51	Jul 15 20:19:51	prod1	admin@pve	VM 99999 - Destroy	OK
Jul 15 20:19:45	Jul 15 20:19:45	prod1	admin@pve	VM 99999 - Create	OK
Jul 15 20:19:02	Jul 15 20:19:03	prod1	admin@pve	VM 9000 - Destroy	OK

クラスタ内のノードはこのレベルで個別に管理できます。

トップヘッダには *Reboot*、*Shutdown*、*Shell*、*Bulk Actions*、*Help* といった便利なボタンがある。*Shell*には *noVNC*、*SPICE*、*xterm.js* のオプションがあります。*Bulk Actions*には *Bulk Start*、*Bulk Shutdown*、*Bulk Migrate* のオプションがあります。

- ・ **検索**: ノードのVM、コンテナ、ストレージデバイス、プールを検索する。
- ・ **サマリー**: ノードのリソース使用状況の概要を表示する。
- ・ **メモ**: [Markdown構文](#)でカスタムコメントを書く。

- ・ **シェル**: ノードのシェル・インターフェースへのアクセス。
- ・ **システム**: ネットワーク、DNS、時刻の設定、シスログへのアクセス。
- ・ **アップデート**: システムをアップグレードし、利用可能な新しいパッケージを表示します。
- ・ **ファイアウォール**: 特定のノードのProxmoxファイアウォールを管理します。
- ・ **ディスク**: 接続されているディスクの概要を把握し、その使用状況を管理する。
- ・ **Ceph**: は、ホストにCephサーバをインストールしている場合にのみ使用します。この場合、ここでCephクラスタを管理し、ステータスを確認できます。
- ・ **レプリケーション**: レプリケーションジョブの表示と管理。
- ・ **タスク履歴**: 過去のタスクのリストを見ることができます。
- ・ **サブスクリプション**: サブスクリプションキーをアップロードし、サポートケースで使用するためのシステムレポートを生成します。

#### 4.4.3 ゲスト

The screenshot shows the Proxmox VE 6.0-4 web interface. The main window displays the summary for a virtual machine named '99999' running on node 'prod1'. The summary panel shows the following details:

Category	Value
Status	stopped
HA State	none
Node	prod1
CPU usage	0.00% of 1 CPU(s)
Memory usage	0.00% (0 B of 1.00 GiB)
Bootdisk size	0 B
IPs	No Guest Agent configured

Below the summary is a chart titled 'CPU usage' showing usage over time.

The left sidebar shows the 'Server View' with the following structure:

- Datacenter (prod-eu-central)
  - prod1
    - 510 (CT510)
    - 101 (win10)
    - 501 (VM 501)
    - 99999
      - cephfs (prod1)
      - cp (prod1)
      - iso (prod1)
      - local (prod1)
      - local-lvm (prod1)
  - prod2
  - prod3
  - development

The bottom section shows a table of recent tasks:

Start Time	End Time	Node	User name	Description	Status
Jul 15 20:36:39	Jul 15 20:36:39	prod1	admin@pve	VM 99999 - Create	OK
Jul 15 20:35:56	Jul 15 20:35:57	prod1	admin@pve	VM 9000 - Destroy	OK
Jul 15 20:35:53	Jul 15 20:35:53	prod1	admin@pve	VM 9000 - Create	OK
Jul 15 20:19:51	Jul 15 20:19:51	prod1	admin@pve	VM 99999 - Destroy	OK
Jul 15 20:19:45	Jul 15 20:19:45	prod1	admin@pve	VM 99999 - Create	OK

ゲストには2種類あり、どちらもテンプレートに変換できる。1つはカーネルベースの仮想マシン（KVM）で

、もう1つはLinuxコンテナ（LXC）です。これらのナビゲーションはほとんど同じで、いくつかのオプションが異なるだけです。

さまざまなゲスト管理インターフェースにアクセスするには、左側のメニューからVMまたはコンテナを選択します。

ヘッダーには、電源管理、マイグレーション、コンソールへのアクセスとタイプ、クローン、HA、ヘルプなどの項目のコマンドが含まれている。これらのボタンのいくつかはドロップダウンメニューを含んでおり、例えば *Shutdown* には他の電源オプションも含まれており、*Console* には様々なコンソールタイプが含まれています：*Console*には、*SPICE*、*noVNC*、*xterm.js*といったコンソールの種類があります。

右側のパネルには、左側のメニューから選択された項目のインターフェイスが含まれる。利用可能なインターフェースは以下の通り。

- **Summary:** VMのアクティビティの簡単な概要と、[Markdown構文](#)のコメント用のNotesフィールドを提供します。
- **コンソール:** VM/コンテナの対話型コンソールにアクセスできる。
- **(KVM)Hardware:** KVM VMで使用可能なハードウェアを定義します。
- **(LXC)Resources:** LXCで利用可能なシステムリソースを定義します。
- **(LXC)Network:** コンテナのネットワーク設定を行う。
- **(LXC)DNS:** コンテナのDNS設定を行う。
- **オプション:** ゲストのオプションを管理します。
- **タスク履歴:** 選択したゲストに関連する過去のタスクをすべて表示します。
- **(KVM) モニタ:** KVMプロセスへの対話型通信インターフェース。
- **バックアップ:** システムバックアップの作成と復元。
- **レプリケーション:** 選択したゲストのレプリケーションジョブを表示および管理します。
- **スナップショット:** VMスナップショットの作成とリストア。
- **ファイアウォール:** VMレベルでファイアウォールを設定する。
- **パーミッション:** 選択したゲストのパーミッションを管理します。

#### 4.4.4 ストレージ

Start Time ↓	End Time	Node	User name	Description	Status
Jul 15 20:35:56	Jul 15 20:35:57	prod1	admin@pve	VM 9000 - Destroy	OK
Jul 15 20:35:53	Jul 15 20:35:53	prod1	admin@pve	VM 9000 - Create	OK
Jul 15 20:19:51	Jul 15 20:19:51	prod1	admin@pve	VM 99999 - Destroy	OK
Jul 15 20:19:45	Jul 15 20:19:45	prod1	admin@pve	VM 99999 - Create	OK
Jul 15 20:19:02	Jul 15 20:19:03	prod1	admin@pve	VM 9000 - Destroy	OK

ゲストインターフェイスと同様に、ストレージのインターフェイスは、左側にある特定のストレージエレメント用のメニューと、右側にあるこれらのエレメントを管理するためのインターフェイスで構成されています。このビューでは、2つのパーティションに分かれています。左側にはストレージのオプションがあり、右側には選択したオプションの内容が表示される。

- 概要:** ストレージの種類、使用状況、保存されているコンテンツなど、ストレージに関する重要な情報を表示します。
- コンテンツ:** ストレージが保存する各コンテンツタイプ（例えば、バックアップ、ISOイメージ、CTテンプレート）のメニュー項目。
- パーミッション:** ストレージのパーミッションを管理する。

#### 4.4.5 プール

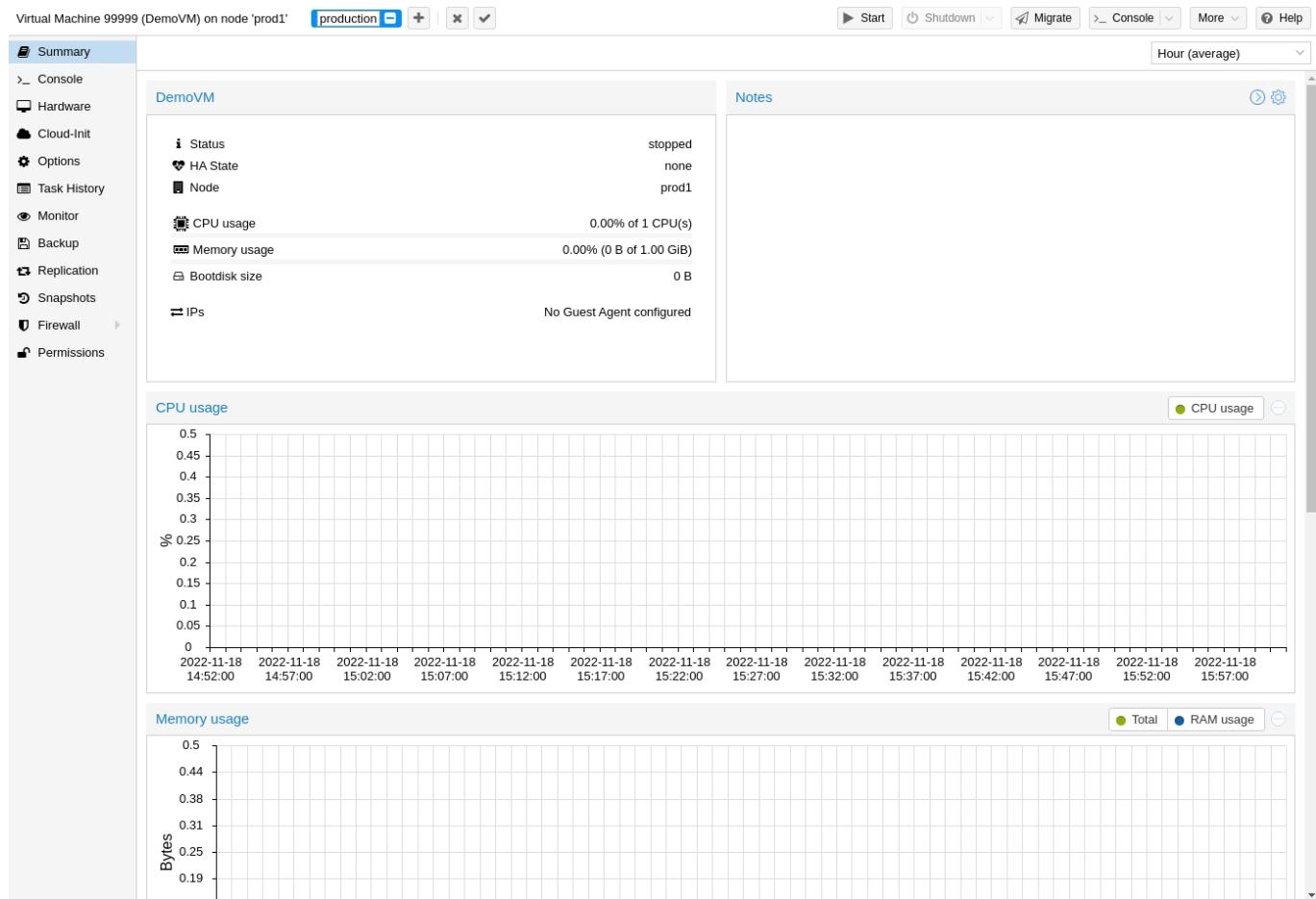
The screenshot shows the Proxmox VE management interface. On the left, a sidebar titled 'Server View' displays a hierarchical tree of datacenters and storage pools. A 'development' pool is selected. The main content area is titled 'Resource Pool: development' and contains three tabs: 'Summary' (selected), 'Members', and 'Permissions'. The 'Summary' tab includes a 'Status' section with a comment 'IT development pool'. At the bottom, there is a table titled 'Tasks' showing a log of recent operations:

Start Time ↓	End Time	Node	User name	Description	Status
Jul 15 20:35:56	Jul 15 20:35:57	prod1	admin@pve	VM 9000 - Destroy	OK
Jul 15 20:35:53	Jul 15 20:35:53	prod1	admin@pve	VM 9000 - Create	OK
Jul 15 20:19:51	Jul 15 20:19:51	prod1	admin@pve	VM 99999 - Destroy	OK
Jul 15 20:19:45	Jul 15 20:19:45	prod1	admin@pve	VM 99999 - Create	OK
Jul 15 20:19:02	Jul 15 20:19:03	prod1	admin@pve	VM 9000 - Destroy	OK

ここでも、プールビューは2つのパーティションで構成されている。左側にメニュー、右側に各メニュー項目に対応するインターフェースである。

- **概要:** プールの説明を示す。
- **Members:** プールメンバー（ゲストとストレージ）の表示と管理。
- **Permissions:** プールの権限を管理します。

## 4.5 タグ



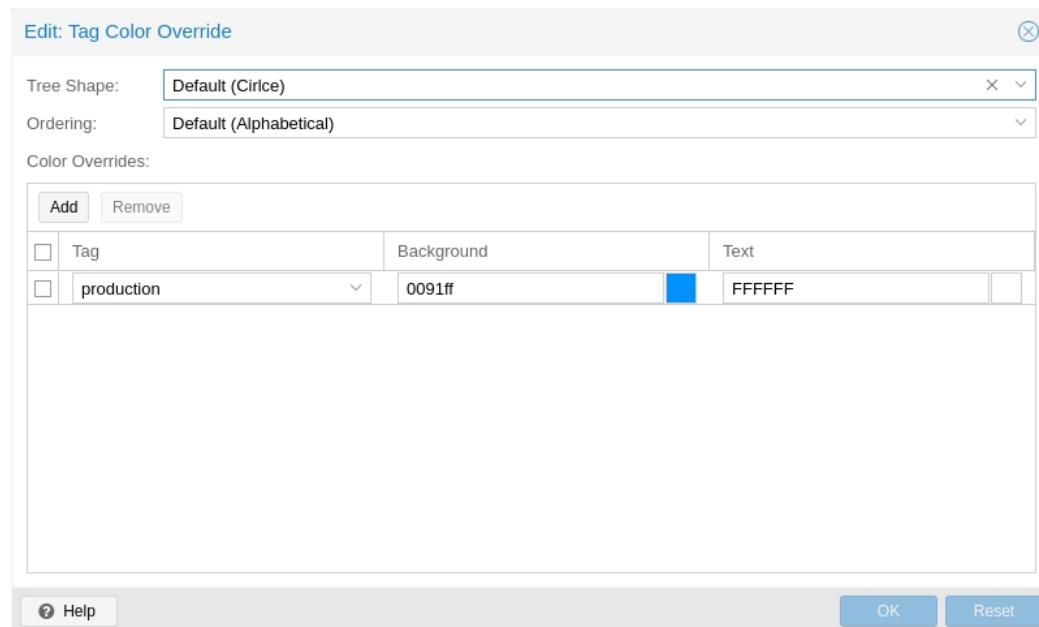
組織的な目的のために、ゲストにタグを設定することが可能です。現在のところ、これらはユーザーに情報価値を提供するだけです。タグは、リソースツリーと、ゲストが選択されたときのステータス行の2つの場所に表示されます。

タグは、鉛筆のアイコンをクリックすることで、ゲストのステータスラインに追加、編集、削除することができます。ボタンを押すと複数のタグを追加でき、-ボタンを押すと削除できます。変更を保存またはキャンセルするには、それぞれ✓ボタンと×ボタンを使用します。

タグはCLIでも設定でき、複数のタグはセミコロンで区切られる。例えば

```
# qm set ID --tags myfirsttag;mysecondtag
```

## 4.5.1 スタイル構成



デフォルトでは、タグの色はテキストから決定論的に導かれます。リソース

ツリー、大文字と小文字の区別、およびタグのソート方法をカスタマイズできます。これはウェブ・インターフェイスの *Datacenter → Options → Tag Style Override* で行うことができます。また、CLIを使用して行うこともできます。例えば

```
# pvesh set /cluster/options --tag-style color-map=example:000000:FFFFFF
```

タグ例の背景色を黒 (#000000) に、文字色を白 (#FFFFFF) に設定します。

## 4.5.2 アクセス許可

Keyboard Layout	German (de)
HTTP proxy	none
Console Viewer	Default (xterm.js)
Email from address	root@\$hostname
MAC address prefix	none
Migration Settings	network=10.3.0.64/24,type=insecure
HA Settings	shutdown_policy=migrate
U2F Settings	None
WebAuthn Settings	None
Bandwidth Limits	None
Maximal Workers/bulk-action	4
Next Free VMID Range	Default
Tag Style Override	Tree Shape: Default (Circle), Ordering: Default (Alphabetical), production
User Tag Access	Mode existing, Pre-defined: allowed
Registered Tags	backup

デフォルトでは、ゲスト (/vms/*ID*) の VM.Config.Options 権限を持つユーザーは、以下のタグを設定できます。

[権限管理を参照](#))。この動作を制限したい場合は、*Datacenter → Options → User Tag Access* で適切なパーミッションを設定できます：

- `free`: ユーザーはタグの設定を制限されない (デフォルト)
- リスト: ユーザーは事前に定義されたタグのリストに基づいてタグを設定できる
- 既存: リストのようなものだが、ユーザーは既存のタグを使うこともできる
- `none`: ユーザーはタグの使用を制限される

CLIでも同じことができる。

なお、/の`Sys.Modify`権限を持つユーザーは、ここでの設定に関係なく、常にタグの設定や削除が可能です。さらに、登録されたタグのリストが設定可能であり、これは/の`Sys.Modify`権限を持つユーザーのみが追加および削除できます。

*Datacenter → Options → Registered Tags* または CLI 経由。

正確なオプションとCLIでの呼び出し方法の詳細については、「[データセンター構成](#)」を参照してください。

## 第5章 クラスター

### マネージャー

Proxmox VE クラスタマネージャ pvecm は、物理サーバのグループを作成するためのツールです。このようなグループを **クラスターと呼びます**。信頼性の高いグループ通信には **Corosync Cluster Engine** を使用します。クラスター内のノード数に明確な制限はありません。実際には、実際に可能なノード数はホストとネットワークのパフォーマンスによって制限される可能性があります。現在（2021年）、50ノードを超えるクラスター（ハイエンドのエンタープライズ・ハードウェアを使用）が稼働しているという報告があります。

pvecm は、新しいクラスターの作成、クラスターへのノードの参加、クラスターからの離脱、ステータス情報の取得、その他様々なクラスター関連のタスクに使用できます。Proxmox クラスタファイルシステム ("pmxcfs") は、クラスター構成をすべてのクラスタノードに透過的に配布するために使用されます。

ノードをクラスターにグループ化すると、次のような利点がある：

- ウェブベースの集中管理
- マルチマスタークラスター：各ノードがすべての管理タスクを実行可能
- データベース駆動型ファイルシステム pmxcfs を設定ファイルの保存に使用し、corosync を使用して全ノードでリアルタイムにレプリケートされる。
- 物理ホスト間での仮想マシンやコンテナの移行が容易
- 迅速な展開
- ファイアウォールや HA のようなクラスタ全体のサービス

## 5.1 必要条件

- corosyncが動作するためには、すべてのノードがUDPポート5405-5412を介して互いに接続できる必要があります。
- 日付と時刻は同期していかなければならない。
- ノード間でTCPポート22のSSHトンネルが必要です。
- 高可用性に興味がある場合は、信頼できるクオーラムのために少なくとも3つのノードが必要です。すべてのノードが同じバージョンである必要があります。
- 特に共有ストレージを使用する場合は、クラスタ・トラフィック専用のNICを推奨します。

- ノードを追加するには、クラスタ・ノードのルート・パスワードが必要です。
- 仮想マシンのオンラインマイグレーションは、ノードが同じベンダーのCPUを搭載している場合にのみサポートされます。それ以外の場合は動作する可能性がありますが、保証はされません。

---

#### 注

Proxmox VE 3.x以前とProxmox VE 4.Xのクラスタノードを混在させることはできません。

---

---

#### 注

Proxmox VE 4.4とProxmox VE 5.0のノードを混在させることは可能ですが、本番構成としてはサポートされていません。

---

---

#### 注

Proxmox VE 6.x のクラスタを以前のバージョンで実行することはできません。Proxmox VE 6.xとそれ以前のバージョンのクラスタプロトコル(corosync)が根本的に変更されました。Proxmox VE 5.4用のcorosync 3/パッケージは、Proxmox VE 6.0へのアップグレード手順のみを対象としています。

---

## 5.2 ノードの準備

まず、すべてのノードにProxmox VEをインストールします。各ノードが最終的なホスト名とIP設定でインストールされていることを確認してください。クラスタ作成後にホスト名とIPを変更することはできません。

`etc/hosts`ですべてのノード名とそのIPを参照するのが一般的ですが（あるいは他の手段で名前を解決できるようにする）、これはクラスタを動作させるために必要ではありません。というのも、あるノードから別のノードにSSH経由で接続する場合、覚えやすいノード名を使うことができるからです（[リンクアドレスの種類も](#)参照してください）。クラスタ構成では常にIPアドレスでノードを参照することをお勧めします。

## 5.3 クラスタの作成

クラスタの作成は、コンソール上で行うか（sshでログイン）、Proxmox VEのWebインターフェース（*Datacenter → Cluster*）を使用してAPIから行うことができます。

---

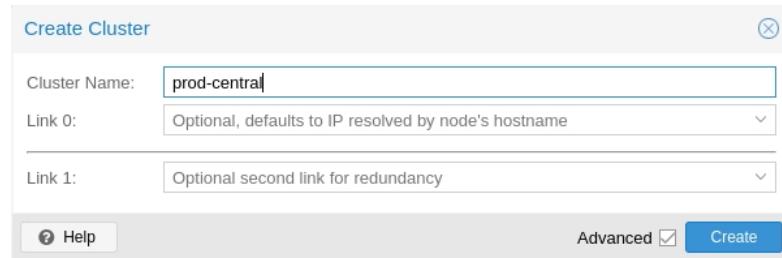
---

## 注

クラスタには一意の名前を付けます。この名前は後で変更できません。クラスタ名はノード名と同じ規則に従います。

---

### 5.3.1 ウェブGUIで作成



Datacenter → Cluster で、**Create Cluster**をクリックします。クラスタ名を入力し、ドロップダウン・リストからメイン・クラスタ・ネットワーク（リンク0）として使用するネットワーク・コネクションを選択します。デフォルトはノードのホスト名。

Proxmox VE 6.2では、クラスタに最大8つのフォールバックリンクを追加できます。冗長リンクを追加するには、Addボタンをクリックし、それぞれのフィールドからリンク番号とIPアドレスを選択します。Proxmox VE 6.2以前では、フォールバックとして2つ目のリンクを追加するには、Advancedチェックボックスを選択し、追加のネットワークインターフェース（リンク1、Corosync冗長性も参照）を選択します。

#### 注

クラスタ通信用に選択したネットワークが、ネットワークストレージやライブマイグレーションのような高トラフィックの目的に使用されていないことを確認してください。クラスタネットワーク自体は少量のデータを生成しますが、レイテンシに非常に敏感です。クラスタネットワーク要件の詳細を確認してください。

### 5.3.2 コマンドラインで作成

最初のProxmox VEノードにssh経由でログインし、以下のコマンドを実行します：

```
hp1# pvecm create CLUSTERNAME
```

新しいクラスタの状態を確認するには、次のようにする：

```
hp1# pvecmステータス
```

### 5.3.3 同一ネットワーク内の複数のクラスタ

同じ物理ネットワークまたは論理ネットワークに複数のクラスタを作成することは可能です。この場合、ク

ラスタ通信スタックでの衝突を避けるため、各クラスタに一意の名前を付ける必要があります。さらに、クラスタを明確に区別できるようにすることで、人間の混乱を避けることができます。

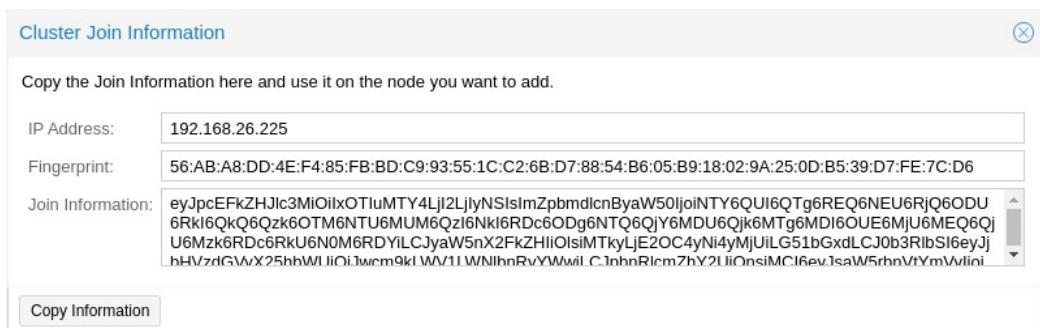
corosyncクラスタの帯域幅要件は比較的低いものの、パッケージのレイテンシと1秒あたりのパッケージ（PPS）レートが制限要因となります。同じネットワーク内の異なるクラスタは、これらのリソースをめぐつて互いに競合する可能性があるため、大規模なクラスタには別の物理ネットワークインフラを使用することが理にかなっている場合があります。

## 5.4 クラスタへのノードの追加

### 注意

**!** クラスタに参加すると、/etc/pveにある既存の設定はすべて上書きされます。特に、ゲストIDが衝突する可能性があるため、参加ノードはゲストを保持できず、ノードはクラスタのストレージ構成を継承します。既存のゲストを持つノードに参加するには、回避策として各ゲストのバックアップを作成し(vzdumpを使用)、参加後に別のIDでリストアします。ノードのストレージレイアウトが異なる場合は、ノードのストレージを再追加し、ストレージが実際に利用可能なノードに反映されるように各ストレージのノード制限を適応させる必要があります。

### 5.4.1 GUIでノードをクラスタに参加させる



既存のクラスタ・ノードでWebインターフェースにログインします。Datacenter → Clusterで、上部の**Join Information**ボタンをクリックします。次に、[Copy Information]ボタンをクリックします。または情報フィールドは手動で。



次に、追加するノードのWebインターフェースにログインします。Datacenter → Clusterで、**Join Cluster**をクリックします。情報フィールドに、先ほどコピーした[参加情報]テキストを入力します。ほとんどの設定が必要クラスタに参加するためのパスワードは自動的に入力されます。セキュリティ上の理由から、クラスタのパスワードは手動で入力する必要があります。

## 注

必要なデータをすべて手動で入力するには、*Assisted Join*チェックボックスを無効にします。

---

**Join**ボタンをクリックすると、クラスタ参加プロセスが直ちに開始されます。ノードがクラスタに参加すると、現在のノード証明書はクラスタ認証局（CA）から署名されたものに置き換えられます。これは、現在のセッションが数秒後に機能しなくなることを意味します。この場合、Webインターフェースを強制的にリロードし、クラスタ認証情報で再度ログインする必要があります。

これで、*Datacenter* → *Cluster*にノードが表示されるはずです。

## 5.4.2 コマンドラインからクラスタにノードを参加させる

既存のクラスタに参加させたいノードにsshでログインします。

```
# pvecm add IP-ADDRESS-CLUSTER
```

IP-ADDRESS-CLUSTERには、既存のクラスタ・ノードのIPまたはホスト名を使用します。IPアドレスが推奨されます([リンクアドレスの種類を参照](#))。

クラスタの状態を確認するには

```
# pvecmステータス
```

### 4ノード追加後のクラスタステータス

```
# クラスタ情報
```

```
~~~~~  
名前 prod-central  
Config Version: 3 トランスポー  
ト: ネット  
安全な認証 オン
```

#### 定足数情報

```
~~~~~  
日付: 9月14日 (火) 11:06:47 2021
```

#### クオーラムプロバイダ:

```
corosync_votequorum ノード数: 4  
ノードID 0x00000001  
リングID 1.1a8  
定足数 はい
```

#### 定足数情報

```
~~~~~  
~ 予想得票数: 4票  
予想最高4  
ノードID 投票 名称  
0x00000001 1 192.168.15.91  
0x00000002 1 192.168.15.92 (現地)  
0x00000003 1 192.168.15.93  
0x00000004 1 192.168.15.94
```

全員情報のリストだけが必要な場合は、これを使う:

```
# pvecmノード
```

## クラスタ内のノードをリストする

```
# pvecm ノード
```

### 会員情報

ノード	投票	名称
1	1	馬力
2	1	hp2 (ローカル )
3	1	馬力
4	1	馬力4

### 5.4.3 分離されたクラスタネットワークを持つノードの追加

クラスタ・ネットワークが分離されているクラスタにノードを追加する場合、*link0*パラメータを使用してその

```
# pvecm add IP-ADDRESS-CLUSTER --link0 LOCAL-IP-ADDRESS-LINK0
```

ネットワーク上のノードのアドレスを設定する必要があります：

クロノスネットのトランスポート・レイヤーに内蔵されている冗長性を使用したい場合は、*link1*パラメータも使用してください。GUIを使用して、**Cluster Join**の対応するLink Xフィールドから正しいインターフェイスを選択することができます。

ダイアログ

## 5.5 クラスタノードの削除



### 注意

手続きを進める前に注意深く読んでください。

ノードからすべての仮想マシンを移動します。保持したいローカルデータやバックアップのコピーが作成されていることを確認します。さらに、削除するノードへのスケジュールされたレプリケーション・ジョブをすべて削除してください。

---

### 注意



ノードを削除する前にそのノードへのレプリケーション・ジョブを削除しないと、レプリケショ

ン・ジョブが削除できなくなります。特に、レプリケートされたVMがマイグレーションされるとレプリケーションの方向が自動的に切り替わるため、削除するノードからレプリケートされたVMをマイグレーションすることで、そのノードへのレプリケーション・ジョブが自動的に設定されることに注意してください。

---

次の例では、クラスタからノードhp4を削除します。

**別のクラスタ・ノード (hp4ではない) にログインし、`pvecm nodes`コマンドを実行して削除するノードIDを特定します：**

hp1# pvecm ノード		
会員	ノード ID	投票 名称
~~~	1	1 hp1 (口一カル )
	2	1 馬力2
	3	1 馬力
	4	1 馬力4

この時点では hp4 の電源を切り、現在のコンフィギュレーションでは（ネットワーク内で）再び電源が入らないようにする必要があります。

### 重要



上述したように、ノードを取り外す前に電源を切り、現在の構成では（既存のクラスタネットワーク内で）再び電源が入らないことを確認することが重要です。そのままノードの電源を入れると、クラスタが壊れてしまい、機能する状態に戻すのが困難になる可能性があります。

ノード hp4 の電源を切ったら、安全にクラスタから取り除くことができる。

```
hp1# pvecm delnode hp4  
ノード4を殺す
```

### 注

この時点で、Could not kill node (error = CS\_ERR\_NOT\_EXIST) というエラー・メッセージが表示される可能性があります。これはノードの削除に失敗したのではなく、corosyncがオフライン・ノードを kill しようとしたことに失敗したことを意味します。したがって、安全に無視できます。

pvecm nodes または pvecm status を使ってノードのリストをもう一度確認してください。以下のようになるはずだ：

```
hp1# pvecm ステータス
```

...

### 定足数情報

```
~~~~~
```

~ 予想得票数: 3票

予想最高位: 3

総得票数 3

定足数 2

フラグ 定足数

### 会員情報

```
~~~~~
```

ノードID	投票 名称
0x00000001	1 192.168.15.90 (口一力 ル)
0x00000002	1 192.168.15.91
0x00000003	1 192.168.15.92

何らかの理由で、このサーバーを再び同じクラスターに参加させたい場合は、そうしなければならない：

- Proxmox VEを新規インストールする、
- そして、前のセクションで説明したように、それに加わる。

削除されたノードの設定ファイルはまだ `/etc/pve/nodes/hp4` に存在します。まだ必要な設定があれば、回復してからディレクトリを削除してください。

---

#### 注

ノードの削除後も、その SSH フィンガープリントは他のノードの `known_hosts` に存在します。同じ IP またはホスト名を持つノードに再参加した後に SSH エラーが発生した場合は、再参加したノードで `pvecm updatecerts` を 1 回実行してフィンガープリントをクラスタ全体で更新します。

---

### 5.5.1 再インストールせずにノードを分離する



#### 注意

これは推奨される方法ではありません。不安な場合は、前の方を使用してください。

---

ノードを最初から再インストールせずにクラスタから分離することもできます。ただし、クラスタからノードを削除した後も、そのノードは共有ストレージにアクセスできます。これは、クラスタからノードの削除を開始する前に解決しておく必要があります。Proxmox VEクラスタは他のクラスタと全く同じストレージを共有することはできません。ストレージのロックはクラスタの境界を越えて機能しないからです。さらに、VMIDの競合につながる可能性もあります。

分離したいノードだけがアクセスできる新しいストレージを作成することをお勧めします。例えば、NFS上の新しいエクスポートや新しいCephプールなどです。まったく同じストレージが複数のクラスタからアクセスされないようにすることが重要です。このストレージを設定したら、すべてのデータとVMをノードからのストレージに移動する。これでノードをクラスタから分離する準備が整った。



#### 警告

すべての共有リソースがきれいに分離されていることを確認してください！ そうしないと、コンフリクト

---

トや問題が発生します。

---

まず、ノードのcorosyncとpve-clusterサービスを停止します：

```
systemctl stop pve-cluster  
systemctl stop corosync
```

クラスタファイルシステムをローカルモードで再度起動します：

```
pmxcfs -l
```

corosync設定ファイルを削除する：

```
rm /etc/pve/corosync.conf  
rm -r /etc/corosync/*
```

これでファイルシステムを通常のサービスとして再び起動できる:

```
killall pmxcfs  
systemctl start pve-cluster
```

これでノードはクラスタから切り離されました。このノードはクラスタの残りのどのノードからも削除できます:

```
pvecm delnode oldnode
```

残りのノードのクオーラムが失われたためにコマンドが失敗した場合、回避策として予想得票数を1に設定す

pvecm期待度1

ることができます:

そして*pvecm delnode*コマンドを繰り返す。

ここで分離したノードに戻り、そのノードに残っているクラスタ・ファイルをすべて削除します。こうすることで、そのノードを再び別のクラスタに問題なく追加できるようになります。

```
rm /var/lib/corosync/*
```

他のノードの設定ファイルがまだクラスタ・ファイル・システムに残っているため、それらもクリーンアップすることをお勧めします。ノード名が正しいことを確認したら、*/etc/pve/nodes/NODENAME*からディレクトリ全体を再帰的に削除します。

---

#### 注意

ノードのSSH鍵は*authorized\_key*ファイルに残ります。これは、ノードがまだ公開鍵認証で互いに接続できることを意味します。*etc/pve/priv/authorized\_keys*ファイルからそれぞれの鍵を削除して、これを修正する必要があります。

---

## 5.6 定足数

Proxmox VEは、すべてのクラスタノード間で一貫した状態を提供するためにクオーラムベースの技術を使用しています。

クオーラム（定足数）とは、分散システムにおいて分散トランザクションが操作を実行するため

に最低限獲得しなければならない票数のことである。

- Wikipediaより クオーラム (分散コンピューティング)

ネットワーク・パーティショニングの場合、状態を変更するには過半数のノードがオンラインである必要がある。クオーラムを失うと、クラスタは読み取り専用モードに切り替わります。

---

### 注

Proxmox VEはデフォルトで各ノードに1票を割り当てます。

---

## 5.7 クラスターネットワーク

クラスタ・ネットワークはクラスタの中核である。これを介して送信されるすべてのメッセージは、それぞれの順序ですべてのノードに確実に配信されなければなりません。Proxmox VEでは、この部分は高性能、低オーバーヘッド、高可用性開発ツールキットの実装であるcorosyncによって行われます。corosyncは分散型設定ファイルシステム(pmxcfs)に対応しています。

### 5.7.1 ネットワーク要件

Proxmox VEクラスタスタックが安定して動作するには、全ノード間のレイテンシが5ミリ秒以下（LANのパフォーマンス）の信頼性の高いネットワークが必要です。ノード数が少ないセットアップでは、より高いレイテンシのネットワークでも動作する可能性がありますが、これは保証されたものではなく、ノード数が3つ以上、レイテンシが約10ミリ秒を超えると、むしろ可能性が低くなります。

corosyncはあまり帯域幅を使用しないが、レイテンシのジッターに敏感であるため、ネットワークは他のメンバーによって多用されるべきではない。理想的には、corosyncは物理的に分離された独自のネットワーク上で実行される。特に、corosyncとストレージのために共有ネットワークを使用しないこと（冗長構成における優先順位の低いフォールバックの可能性を除く）。

クラスタをセットアップする前に、ネットワークがその目的に適しているかどうかを確認することは良い習慣です。クラスタ・ネットワーク上でノードが互いに接続できることを確認するには、pingツールでノード間の接続性をテストします。

Proxmox VEファイアウォールが有効な場合、corosyncのACCEPTルールが自動的に生成されます。

#### 注

Corosyncは、バージョン3.0（Proxmox VE 6.0で導入）以前はマルチキャストを使用していました。最近のバージョンはクラスタ通信にKronosnetを利用していますが、今のところ通常のUDPユニキャストしかサポートしていません。

#### 注意

 corosync.confでトランスポートをudpまたはudpuに設定することで、マルチキャストまたはレガシーユニキャストを有効にすることはできますが、暗号化および冗長性サポートがすべて無効にな

ることに注意してください。したがって、これは推奨されません。

---

## 5.7.2 クラスタ・ネットワークの分離

パラメータなしでクラスタを作成する場合、corosyncクラスタのネットワークは通常、WebインターフェイスおよびVMのネットワークと共有されます。セットアップによっては、ストレージトラフィックも同じネットワークで送信されるかもしれません。corosyncはタイムクリティカルでリアルタイムなアプリケーションなので、これを変更することをお勧めします。

### 新しいネットワークの設定

まず、新しいネットワークインターフェイスをセットアップしなければならない。これは物理的に別のネットワーク上にある必要があります。ネットワークが[クラスタ・ネットワークの要件を満たしていることを確認](#)してください。

## クラスタ作成時の分離

これは、新しいクラスタの作成に使用される `pvecm create` コマンドの `linkX` パラメータによって可能です。

10.10.10.1/25にスタティック・アドレスを持つ追加のNICをセットアップし、すべてのクラスタ通信をこの

```
pvecm create test --link0 10.10.10.1
```

インターフェイスで送受信したい場合、次のように実行する：

すべてが正しく機能しているかどうかをチェックするには、以下を実行する：

```
systemctl status corosync
```

その後、上記の手順に従って、[分離したクラスタ・ネットワークでノードを追加します](#)。

## クラスタ作成後の分離

既にクラスタを作成していて、クラスタ全体を再構築することなく、その通信を別のネットワークに切り替える場合にこれを行うことができます。この変更により、ノードはcorosyncを再起動し、新しいネットワーク上で次々に立ち上がる必要があるため、クラスタのクオーラムが短時間失われる可能性があります。

まず、[corosync.conf](#) ファイルの編集方法を確認してください。そして、それを開くと、以下のようなファイルがあるはずだ：

```
logging {  
    debug: off  
    to_syslog: yes  
}
```

#### ノードリスト

```
name:  
due  
nodeid: 2  
quorum_votes: 1  
ring0_addr: デ  
ユー  
}
```

#### ノード

```
name:  
tre  
nodeid: 3  
quorum_votes: 1  
ring0_addr: tre  
}
```

#### ノード

```
name:  
uno  
nodeid: 1  
quorum_votes: 1  
ring0_addr: uno  
}
```

```
}
```

#### クオーラム

```
    プロバイダ: corosync_votequorum
}
```

## トーテム

```
    cluster_name: testcluster
    config_version: 3
    ip_version: ipv4-6
    secauth: on

    バージョン: 2
    リンク番号: 0
}

}
```

## 注

`ringX_addr`は実際にはcorosyncリンクアドレスを指定する。`ring`"という名前は、古いバージョンのcorosyncの名残であり、後方互換性のために残されている。

---

最初にすることは、もしまだ表示されていなければ、ノード・エントリーに**名前のプロパティ**を追加することです。これらはノード名と一致していなければなりません。

次に、すべてのノードの`ring0_addr`プロパティからすべてのアドレスを新しいアドレスに置き換える。ここでは、プレーンIPアドレスまたはホスト名を使用できます。ホスト名を使用する場合は、すべてのノードから解決可能であることを確認してください（[リンクアドレスの種類](#)も参照してください）。

この例では、クラスタ通信を10.10.10.0/25ネットワークに切り替えたいので、以下のように変更します。それぞれのノードの`ring0_addr`。

---

## 注

全く同じ手順で、他の`ringX_addr`値も変更できる。しかし、一度に1つのリンクアドレスだけを変更することを推奨する。

---

`config_version`プロパティを増やすと、新しいコンフィギュレーション・ファイルは次のようになる：

```
logging {  
    debug: off  
    to_syslog: yes  
}
```

## ノードリスト

```
name:  
due  
nodeid: 2  
定足数1  
ring0_addr: 10.10.10.2  
}
```

## ノード

```
name:  
tre  
nodeid: 3  
定足数1  
ring0_addr: 10.10.10.3  
}
```

## ノード

```
name:  
uno  
nodeid: 1  
定足数1  
ring0_addr: 10.10.10.1  
}
```

```
}
```

## クオーラム

```
プロバイダ: corosync_votequorum  
}
```

## トーテム

```
cluster_name: testcluster  
config_version: 4  
ip_version: ipv4-6  
secauth: on  
バージョン: 2  
interface {  
    リンク番号: 0  
}  
}
```

そして、変更した情報がすべて正しいことを最終チェックした後、保存し、もう一度、[corosync.conf](#)ファイルの編集セクションに従って、有効にする。

変更はライブで適用されるので、corosyncの再起動は厳密には必要ありません。他の設定も変更した場合や、corosyncが文句を言っているのに気づいた場合は、オプションで再起動をトリガーすることができます。

単一ノードで実行する：

```
systemctl restart corosync
```

今、すべてが問題ないかチェックする：

```
systemctl status corosync
```

corosyncが再び動き始めたら、他の全てのノードでもcorosyncを再起動してください。すると、新しいネットワーク上でクラスタ・メンバーシップに1つずつ加わることになります。

### 5.7.3 コロシンクのアドレス

corosyncリンクアドレス（後方互換性のため、`corosync.conf`では`ringX_addr`と表記）は、2つの方法で指定できる：

- IPv4/v6アドレスは直接使うことができる。これらは静的であり、通常は不用意に変更されることはないので、推奨される。
- ホスト名はgetaddrinfoを使って解決されます。つまり、デフォルトでは、利用可能であればIPv6アドレスが最初に使われます (man gai.confも参照してください)。特に既存のクラスタをIPv6にアップグレードする場合は、この点に注意してください。

---

#### 注意

 ホスト名の解決先アドレスは、corosyncやそれが動作しているノードに触れることなく変更することができます。そのため、注意して使用する必要があります。

---

ホスト名を優先する場合は、corosync専用の別個の静的ホスト名を推奨します。また、クラスタ内のすべてのノードがすべてのホスト名を正しく解決できることを確認してください。

Proxmox VE 5.1以降、サポートされている間は、ホスト名は入力時に解決されます。解決されたIPのみが設定に保存されます。

以前のバージョンでクラスタに参加したノードは、corosync.confで未解決のホスト名をまだ使用している可能性があります。上記のように、IPまたは別のホスト名で置き換えるのが良いかもしれません。

## 5.8 コロシンクの冗長性

Corosyncはデフォルトで、統合されたKronosnetレイヤーを介した冗長ネットワーキングをサポートしています（レガシーのudp/udpuトранSPORTではサポートされていません）。複数のリンクアドレスを指定することで、有効にすることができます。

--pvecmの-linkXパラメータ、GUIでリンク1（クラスタ作成時または新規ノード追加時）、またはcorosync.confで複数のringX\_addrを指定します。

---

#### 注

有用なフェイルオーバーを提供するためには、各リンクはそれ自身の物理的なネットワーク接続上にあるべきである。

---

リンクは優先度の設定に従って使用されます。この優先順位は、corosync.confの対応するインターフ

```
# pvecm create CLUSTERNAME --link0 10.10.10.1,priority=15 --link1='10.20.20.1,priority=20'
```

エイスクションで*knet\_link\_priority*を設定するか、できればpvecmでクラスタを作成するときに*priority*パラメータを使うことで設定できます：

この場合、優先順位が高いリンク1が最初に使われることになる。

優先順位が手動で設定されていない場合（または2つのリンクが同じ優先順位を持つ場合）、リンクは番号順に使用され、番号の小さい方が高い優先順位を持つ。

すべてのリンクが機能していても、最も優先度の高いリンクだけがコロシンク・トラフィックを見ることがある。リンクの優先度を混在させることはできません。つまり、優先度の異なるリンクは互いに通信することができません。

優先順位の低いリンクは、優先順位の高いリンクがすべて故障しない限りトラフィックが発生しないため、他のタスク（VM、ストレージなど）に使用するネットワークを優先順位の低いリンクに指定するのは有効な戦略となる。最悪の場合、待ち時間が長かったり混雑していたりする接続の方が、全く接続されないよりもかもしれません。

### 5.8.1 既存のクラスタに冗長リンクを追加する

実行中のコンフィギュレーションに新しいリンクを追加するには、まず[corosync.confファイルの編集](#)方法を確認する。

次に、`nodelist`セクションのすべてのノードに新しい`ringX_addr`を追加します。Xはどのノードに追加しても同じで、各ノードで一意であることを確認してください。

最後に、Xを上記で選んだリンク番号に置き換えて、下図のような新しいインターフェイスをトーテムセクションに追加する。

1番のリンクを追加したと仮定すると、新しいコンフィギュレーション・ファイルは次のようになる：

```
logging {  
    debug: off  
    to_syslog: yes  
}
```

#### ノodelリスト

##### ノード

```
name:  
due  
nodeid: 2  
定足数1  
ring0_addr: 10.10.10.2  
ring1_addr: 10.20.20.2  
}
```

##### ノード

```
name:  
tre  
nodeid: 3  
定足数1  
ring0_addr: 10.10.10.3  
ring1_addr: 10.20.20.3  
}
```

##### ノード

```
name:  
uno  
nodeid: 1  
定足数1  
ring0_addr: 10.10.10.1  
ring1_addr: 10.20.20.1  
}
```

```
}
```

## クオーラム

```
プロバイダ: corosync_votequorum
}
```

## トーム

```
cluster_name:
testcluster

config_version: 4
ip_version: ipv4-6
secauth: on
バージョン: 2
```

```
インターフェース {
    linknumber: 0
}
インターフェース {
    linknumber: 1
}
```

最後の手順に従って [corosync.conf ファイルを編集](#)すると、すぐに新しいリンクが有効になります。再起動は必

```
journalctl -b -u corosync
```

要ありません。corosyncが新しいリンクを読み込んだかどうかは、次のようにして確認できます：

あるノードで古いリンクを一時的に切断し、そのステータスが切断中もオンラインであることを確認して、新しいリンクをテストするのは良い考えかもしれない：

```
pvecmステータス
```

健全なクラスタ状態が表示された場合は、新しいリンクが使用されていることを意味します。

## 5.9 Proxmox VEクラスタにおけるSSHの役割

Proxmox VEは、様々な機能にSSHトンネルを利用しています。

- コンソール/シェルセッションのプロキシ（ノードとゲスト）

ノードAに接続している状態でノードBのシェルを使用する場合、ノードAのターミナル・プロキシに接続し、そのターミナル・プロキシは非インタラクティブSSHトンネルを介してノードBのログイン・シェルに接続する。

- セキュアモードでのVMとCTのメモリとローカルストレージのマイグレーション。

移行中、移行情報を交換し、メモリとディスクの内容を転送するために、1つ以上のSSHトンネルが移行元と移行先のノード間で確立される。

- ストレージ・レプリケーション

### 5.9.1 SSHセットアップ

Proxmox VEシステムでは、SSH設定/セットアップに以下の変更が行われます：

- ルート・ユーザーのSSHクライアントの設定が、ChaCha20よりもAESを優先するように設定されている。
- rootユーザーのauthorized\_keysファイルが/etc/pve/priv/authorized\_keysにリンクされ、クラスタ内のすべての認可キーがマージされる。
- sshdはパスワードを使ってrootとしてログインできるように設定されている。

---

### 注

古いシステムでは、/etc/ssh/ssh\_known\_hosts が次の場所を指すシンボリックリンクとして設定されていることもあります。

/このシステムは pve-cluster <> の明示的なホスト鍵固定に置き換えられました。このシステムは pve-cluster <> の明示的なホスト鍵の固定に置き換えられ、シンボリックリンクは pvecm updatecerts --unmerge-known-hosts を実行することで解除できます。

### 5.9.2 .bashrcとその兄弟の自動実行による落とし穴

カスタム.bashrcや、設定されたシェルがログイン時に実行する類似のファイルがある場合、セッションが正常に確立されると、sshは自動的にそれを実行します。これは予期せぬ動作を引き起こす可能性があります。これらのコマンドは、上記の操作のいずれに対してもroot権限で実行される可能性があるからです。これは、問題となる副作用を引き起こす可能性があります！

このような複雑さを避けるためには、/root/.bashrcでセッションが対話型であることを確認するチェックを追加し、その上で.bashrcコマンドを実行することをお勧めする。

```
# 副作用を避けるため、対話的に実行しない場合は早めに終了すること case $- in
  *i* ) ;;
  *) return;;
エサック
```

このスニペットを.bashrcファイルの先頭に追加してください：

## 5.10 コロシンク外部投票サポート

このセクションでは、Proxmox VEクラスタに外部ボータを配置する方法について説明します。このように構成すると、クラスタはクラスタ通信の安全特性に違反することなく、より多くのノード障害に耐えることができます。

これが機能するためには、2つのサービスが関係している：

- Proxmox VEの各ノードで動作するQDeviceデーモン
- 独立したサーバー上で動作する外部投票デーモン

その結果、小規模なセットアップ（例えば2+1ノード）でも高い可用性を実現できる。

### 5.10.1 QDevice技術概要

Corosync Quorum Device (QDevice)は、各クラスタノード上で動作するデーモンです。外部で実行されるカードパーティのアービトレーターの決定に基づいて、クラスタのクオーラムサブシステムに設定された票数を提供します。主な用途は、標準的なクオーラム規則が許容するよりも多くのノード障害にクラスタが耐えられるようにすることです。外部デバイスはすべてのノードを見るため、これは安全に行うこ

とができ、投票するノードのセットを1つだけ選択することができます。これは、サードパーティの投票を受けた後、そのノードのセットがクオーラムを（再び）持つことができる場合にのみ実行されます。

現在、サードパーティ・アービトレーターとしてサポートされているのは*QDevice Net*のみです。これは、ネットワーク経由でパーティションメンバーに到達できる場合、クラスターパーティションに票を提供するデーモンです。常にクラスタの1つのパーティションにしか票を提供しません。複数のクラスタをサポートするように設計されており、設定や状態はほとんど自由です。新しいクラスタは動的に処理され、*QDevice*を実行しているホストには設定ファイルは必要ありません。

外部ホストの唯一の要件は、クラスタへのネットワークアクセスが必要であることと、*corosync-qnetd*パッケージが利用可能であることです。私たちはDebianベースのホスト用にパッケージを提供していますが、他のLinuxディストリビューションでもそれぞれのパッケージマネージャから利用可能なパッケージがあるはずです。

---

### 注

*corosync*自体とは異なり、*QDevice*はTCP/IPでクラスタに接続する。デーモンはクラスタのLAN外でも実行でき、*corosync*の低レイテンシ要件に制限されない。

### 5.10.2 対応セットアップ

偶数ノード数のクラスタではQDevicesをサポートし、2ノードクラスタでは、より高い可用性を提供するのであればQDevicesを推奨しています。奇数ノード数のクラスタについては、現在のところQDeviceの使用を推奨していません。その理由は、QDeviceが各クラスタタイプに提供する票数に違いがあるためです。偶数番号のクラスタは追加の1票を得ますが、これは可用性を高めるだけで、QDevice自体に障害が発生した場合は、QDeviceをまったく使用しない場合と同じ状態になるからです。

一方、クラスタサイズが奇数の場合、QDeviceは $(N-1)$ 票を提供する-ここで $N$ はクラスタノード数に対応する。この代替動作は理にかなっています。もしQDeviceに1票しか追加の票がなかったら、クラスタは分裂状態に陥る可能性があります。このアルゴリズムでは、1つのノードを除くすべてのノード（そして当然QDevice自体も）が故障することが可能です。しかし、これには2つの欠点がある：

- QNetデーモン自体に障害が発生した場合、他のノードに障害が発生したり、クラスタが直ちにクオーラムを失うことはありません。たとえば、15ノードのクラスタでは、クラスタがクオーラムを失う前に7ノードが故障する可能性があります。しかし、ここでQDeviceが設定され、それ自体が故障した場合、15ノードのうち1ノードも故障することはできません。この場合、QDeviceはほぼ単一障害点として機能します。
- 1つのノードとQDeviceを除くすべてのノードが故障する可能性があるという事実は、最初は有望に聞こえますが、これはHAサービスの大量復旧につながる可能性があり、残りの1つのノードに過負荷がかかる可能性があります。さらに、 $((N-1)/2)$ ノード以下しかオンラインでない場合、Cephサーバはサービスの提供を停止します。

欠点とその意味を理解すれば、奇数クラスタのセットアップでこの技術を使うかどうかを自分で決めることができる。

### 5.10.3 QDevice-Netのセットアップ

corosync-qdeviceへの投票を提供するデーモンは、非特権ユーザーとして実行することをお勧めします。Proxmox VEとDebianは、そのように設定済みのパッケージを提供しています。Proxmox VEにQDeviceを安全かつセキュアに統合するには、デーモンとクラスタ間のトライフィックを暗号化する必要があります。

まず、外部サーバーにcorosync-qnetdパッケージをインストールします。

```
external# apt install corosync-qnetd
```

と `corosync-qdevice` パッケージを全クラスタノードにインストールします。

```
pve# apt install corosync-qdevice
```

これを行った後、クラスタ内のすべてのノードがオンラインであることを確認する。

Proxmox VE ノードの1つで以下のコマンドを実行して、QDeviceをセットアップできます：

```
pve# pvecm qdevice setup <QDEVICE-IP>.
```

クラスタのSSHキーが自動的にQDeviceにコピーされます。

---

### 注

外部サーバーのrootユーザーに鍵ベースのアクセスをセットアップするか、セットアップ段階で一時的にパスワードによるrootログインを許可するようにしてください。この段階で「*Host key verification failed.*」などのエラーが表示された場合は、`pvecm updatecerts` を実行すると問題が解決する可能性があります。

すべてのステップが正常に完了すると、「完了」と表示されます。でQDeviceがセットアップされたことを確認できます：

```
pve# pvecm ステータス
```

...

### 定足数情報

```
~~~~~
```

~ 予想得票数: 3票

予想最高位: 3

総得票数 3

定足数 2

フラグ: クオーレート Qdevice

ノードID	投票	Qデバイス 名称
会員 0x00000001	1	A、V、NMW 192.168.22.180 (口一力 ル)
~~~~~		
0x00000002	1	A、V、NMW 192.168.22.181
0x00000000	1	Qデバイス

### Qデバイス・ステータス・フラグ

上で見たように、QDeviceのステータス出力には通常3つの列が含まれる：

- A / NA: Alive または Not Alive。外部のcorosync-qnetdデーモンとの通信が機能しているかどうかを示します。
- V / NV: QDeviceがノードに投票する場合。ノード間のcorosync接続がダウンしているが、両方とも外部のcorosync-qnetdデーモンと通信できるスプリットブレインの状況では、1つのノードだけが票を獲得します。
- MW / NMW: マスターが勝つか (MV) 、勝たないか (NMW) 。デフォルトはNMW。<sup>1</sup>
- NR: QDeviceが登録されていません。

---

### 注

QDeviceがNot Alive (上記の出力でNA) と表示されている場合は、外部サーバーのポート5403 (qnetdサーバーのデフォルトポート) がTCP/IP経由で到達可能であることを確認してください!

---

## 5.10.4 よくある質問

### タイブレーク

2つの同じ大きさのクラスターパーティションが互いに見えないがQDeviceは見えるという同点の場合、QDeviceはそれらのパーティションの1つをランダムに選び、そのパーティションに票を提供する。

---

<sup>1</sup>votequorum\_qdevice\_master\_wins マニュアルページ [https://manpages.debian.org/bookworm/libvotequorum-dev/-/votequorum\\_qdevice\\_master\\_wins.3.ja.html](https://manpages.debian.org/bookworm/libvotequorum-dev/-/votequorum_qdevice_master_wins.3.ja.html)

## 否定的な影響の可能性

ノード数が偶数のクラスタでは、QDeviceを使用してもマイナスの影響はありません。QDeviceが機能しなければ、QDeviceがないのと同じです。

## QDeviceセットアップ後のノードの追加と削除

QDeviceがセットアップされたクラスタに新しいノードを追加したり、既存のノードを削除したりする場合は、まずQDeviceを削除する必要があります。その後、普通にノードを追加または削除できます。ノード数が均等なクラスタに戻ったら、前述の方法でQDeviceを再びセットアップできます。

## QDeviceの取り外し

```
pve# pvecm qdevice remove
```

公式のpvecmツールを使ってQDeviceを追加した場合は、それを削除することができます：

## 5.11 コロシンクの設定

etc/pve/corosync.confファイルはProxmox VEクラスタで中心的な役割を果たします。クラスタ・メンバーシップとそのネットワークを制御します。corosync.confの詳細については、corosync.confのマニュアルページを参照してください：

```
man corosync.conf
```

ノード・メンバーシップについては、Proxmox VEが提供するpvecmツールを常に使用する必要があります。その他の変更については、設定ファイルを手動で編集する必要があるかもしれません。以下はそのためのベストプラクティスのヒントです。

### 5.11.1 corosync.confを編集する

corosync.confファイルの編集は必ずしも簡単ではない。各クラスタノードに2つあり、1つは/etc/pve/corosync.confと、/etc/corosync/corosync.confにあります。クラスタ・ファイル・システムにあるものを編集すると、ローカルのものに変更が伝搬しますが、その逆はありません。ファイルが変更されるとすぐに、設定は自動的に更新されます。つまり、実行中のcorosyncに統合できる変

```
cp /etc/pve/corosync.conf /etc/pve/corosync.conf.new
```

更は即座に反映されます。従って、編集中にファイルを保存する際に意図しない変更を引き起こさないように、常にコピーを作成し、代わりにそれを編集する必要があります。

次に、すべてのProxmox VEノードにプリインストールされているnanoやvim.tinyなどのお気に入りのエディタで設定ファイルを開きます。

---

### 注

コンフィギュレーションを変更した後は、必ず*config\_version*番号をインクリメントする。

---

必要な変更を行った後、現在の作業用コンフィギュレーション・ファイルのコピーをもう1つ作成する。これは、新しいコンフィギュレーションの適用に失敗したり、その他の問題が発生した場合のバックアップとして機能します。

```
cp /etc/pve/corosync.conf /etc/pve/corosync.conf.bak
```

そして古い設定ファイルを新しいものに置き換える:

```
mv /etc/pve/corosync.conf.new /etc/pve/corosync.conf
```

変更が自動的に適用されるかどうかは、以下のコマンドで確認できる:

```
systemctl status  
corosync journalctl -b -  
u corosync
```

変更が自動的に適用されない場合は、corosyncサービスを再起動する必要があります:

```
systemctl restart corosync
```

エラーについては、以下のトラブルシューティングのセクションを確認してください。

## 5.11.2 トラブルシューティング

### 問題: *quorum.expected\_votes* の設定が必要

corosyncが失敗し始め、システムログに以下のメッセージが表示された場合:

```
[...]  
corosync[1647]: [QUORUM] 定数プロバイダ: corosync_votequorumが失敗しました。  
    初期化。  
corosync[1647]: [サー ブ] サービスエンジン 'corosync_quorum' のロードに失敗しま ←  
    :  
    '設定エラー: nodelistまたはquorum.expected_votesは←でなければならない'  
    何故なら構成されている  
[...]
```

これは、コンフィギュレーションでcorosync *ringX\_addr*に設定したホスト名が解決できなかったことを意味する。

### 非クオーレート時にコンフィギュレーションを書き込む

クオーラムのないノードで/etc/pve/corosync.confを変更する必要があり、自分が何をしているかを理解して

pvecm期待度1

いる場合は、これを使う:

これで予想される投票数が1に設定され、クラスタがquorateになります。その後、設定を修正したり、最後に動作したバックアップに戻したりすることができます。

corosyncが起動できなくなった場合は、これだけでは十分ではありません。その場合は、

`/etc/corosync/corosync.conf`にあるcorosync設定のローカルコピーを編集して、corosyncが再び起動できるようになるのがベストだ。スプリットブレインの状況を避けるため、すべてのノードでこのコンフィギュレーションが同じ内容であることを確認してください。

### 5.11.3 Corosync設定用語集

#### リンクXアドレス

これは、ノード間のクロノスネット接続のための異なるリンクアドレスに名前を付けます。

## 5.12 クラスター・コールドスタート

すべてのノードがオフラインの場合、クラスタがクオーレートでないことは明らかだ。これは停電後によくあるケースです。

---

### 注

無停電電源装置 ("UPS"、"バッテリー・バックアップ"とも呼ばれる) を使ってこのような状態にならないようにするのは、特にHAを望む場合には常に良い考えだ。

---

ノードの起動時に `pve-guests` サービスが開始され、クオーラムを待ちます。定足数が満たされると、`onboot` フラグが設定されているすべてのゲストが起動します。

ノードの電源を入れたとき、または停電後に電源が回復したとき、一部のノードが他のノードよりも速く起動する可能性があります。クオーラムに達するまでゲストの起動が遅れることに留意してください。

## 5.13 ゲストVMID自動選択

新しいゲストを作成する際、ウェブインターフェースは自動的にバックエンドに空いているVMIDを尋ねます。デフォルトの検索範囲は100から1000000です（スキーマによって強制される最大許容VMIDより低い）。

例えば、一時的なVMと手動でVMIDを選択するVMを簡単に分けたい場合などだ。また、安定した長さのVMIDを提供したい場合もあり、その場合は下限を例えば100000に設定すると、かなり余裕ができる。

このユースケースに対応するために、`datacenter.cfg`で下限、上限、または両方の境界を設定することができる。

設定ファイルは、ウェブ・インターフェイスの「*Datacenter → Options*」で編集できる。

---

### 注

この範囲はnext-id APIコールにのみ使用されるので、厳しい制限ではない。

---

## 5.14 ゲスト移住

仮想ゲストを他のノードに移行することは、クラスタにおいて便利な機能です。このようなマイグレーション

ンの動作を制御する設定があります。これは、設定ファイルdatacenter.cfgを介して、またはAPIまたはコマンドラインパラメータを介して特定のマイグレーションに対して行うことができます。

ゲストがオンラインかオフラインか、またはローカルリソース（ローカルディスクなど）を持っているかどうかに違いがあります。仮想マシンの移行の詳細については、[QEMU/KVM 移行の章](#)を参照してください。

コンテナ移行の詳細については、[コンテナ移行の章](#)を参照のこと。

### 5.14.1 マイグレーション・タイプ

マイグレーションタイプは、マイグレーションデータを暗号化された（安全な）チャネルで送信するか、暗号化されていない（安全でない）チャネルで送信するかを定義します。マイグレーション・タイプを安全でないものに設定することは、RAMのコンテンツである

仮想ゲストも暗号化されずに転送されるため、ゲスト内部の重要なデータ（パスワードや暗号化キーなど）の情報漏洩につながる可能性があります。

したがって、ネットワークを完全に制御できず、誰にも盗聴されていないことを保証できない場合は、安全なチャネルを使用することを強くお勧めします。

---

#### 注

ストレージの移行はこの設定に従わない。現在、ストレージコンテンツは常にセキュアチャネルで送信される。

---

暗号化には多くの計算パワーが必要なため、この設定はパフォーマンスを向上させるために安全でない設定に変更されることが多い。最近のシステムでは、ハードウェアでAES暗号化を実装しているため、影響は少なくなっています。パフォーマンスへの影響は、10Gbps以上の転送が可能な高速ネットワークで特に顕著です。

### 5.14.2 移住ネットワーク

デフォルトでは、Proxmox VEはクラスタ通信が行われるネットワークを使用して移行トラフィックを送信します。これは、機密性の高いクラスタトラフィックが中断される可能性があることと、このネットワークがノードで利用可能な最良の帯域幅を持たない可能性があるため、いずれも最適ではありません。

マイグレーションネットワークパラメータを設定すると、すべてのマイグレーショントラフィックに専用ネットワークを使用できるようになる。メモリに加えて、これはオフライン移行のストレージトラフィックにも影響します。

移行ネットワークは、CIDR表記を使用したネットワークとして設定されます。これには、各ノードに個別のIPアドレスを設定する必要がないという利点があります。Proxmox VEは、CIDR形式で指定されたネットワークから移行先ノードの実アドレスを決定できます。これを有効にするには、各ノードがそれぞれのネットワークで正確に1つのIPを持つようにネットワークを指定する必要があります。

#### 例

ここでは3つのノードのセットアップを想定し、3つの独立したネットワークを持つ。1つはインターネットとのパブリック通信用、もう1つはクラスタ通信用、そしてもう1つはマイグレーション用の専用ネットワー

---

クとして使いたい非常に高速なものだ。

このようなセットアップのネットワーク構成は以下のようになる：

```
iface eno1 inet manual  
  
# パブリックネットワー  
ク自動vmbr0  
iface vmbr0 inet static  
    address  
    192.X.Y.57/24  
    gateway 192.X.Y.1  
    bridge-ports eno1  
    bridge-stp off  
    bridge-fd 0  
  
# クラスタ・ネットワー  
ク自動eno2  
iface eno2 inet static
```

```
アドレス 10.1.1.1/24
```

```
# 高速ネットワー
```

```
ク自動エノ3
```

```
iface eno3 inet 静的アドレ
```

```
ス 10.1.2.1/24
```

ここでは、移行ネットワークとしてネットワーク10.1.2.0/24を使用します。1回のマイグレーションでは、コ

```
# qm migrate 106 tre --online --migration_network 10.1.2.0/24
```

マンドラインツールのmigration\_networkパラメータを使ってこれを行うことができます：

これをクラスタ内のすべてのマイグレーション用のデフォルト・ネットワークとして設定するには、次のように

/etc/pve/datacenter.cfgファイル:

```
# use dedicated migration network
migration: secure, network=10.1.2.0/24
```

---

## 注

その 移行 タイプ は 常に でなければならぬ。 設定されていなければなりません。  
でなければならぬ。 を設定しなければならぬ。 マイグレーション ネットワーク が 設定  
で

/etc/pve/datacenter.cfg。

---

## 第6章

# Proxmox クラスタファイルシステム (pmxcfs)

Proxmox Clusterファイルシステム("pmxcfs")は、設定ファイルを保存するためのデータベース駆動型のファイルシステムで、corosyncを使用してすべてのクラスタノードにリアルタイムでレプリケートされます。Proxmox VEに関連するすべての設定ファイルを保存するために使用します。

ファイルシステムはすべてのデータをディスク上の永続データベース内に格納するが、データのコピーはRAMに存在する。このため、最大サイズには制限があり、現在のところ128メガバイトとなっている。これでも数千台の仮想マシンの設定を保存するには十分だ。

このシステムには次のような利点がある：

- すべてのコンフィギュレーションをリアルタイムで全ノードにシームレスにレプリケーション
- VM IDの重複を避けるための強力な一貫性チェックを提供
- ノードがクオーラムを失った場合の読み取り専用
- 全ノードへのcorosyncクラスタ構成の自動更新
- 分散ロック機構付き

### 6.1 POSIX互換性

ファイルシステムはFUSEをベースにしているので、動作はPOSIXライクだ。しかし、いくつかの機能は単に実装されていない：

- 通常のファイルやディレクトリは生成できるが、シンボリックリンクは生成できない。
- 空でないディレクトリの名前を変更することはできない（この方がVMIDが一意であることを保証しやすくなるため）。
- ファイルのパーミッションを変更できない（パーミッションはパスに基づいている）。
- `O_EXCL`の作成はアトミックではなかった（古いNFSのように）。
- `O_TRUNC`の作成はアトミックではない（FUSEの制限）。

## 6.2 ファイルアクセス権

すべてのファイルとディレクトリはユーザーrootが所有し、グループwww-dataを持つ。rootのみが書き込み権限を持つが、グループwww-dataはほとんどのファイルを読むことができる。以下のパス以下のファイルはrootのみがアクセスできる：

```
/etc/pve/priv/  
/etc/pve/nodes/${NAME}/priv/
```

## 6.3 テクノロジー

クラスタ通信にはCorosync Cluster Engineを、データベースファイルにはSQLiteを使用している。ファイルシステムはFUSEを使ってユーザー空間に実装されている。

## 6.4 ファイルシステムのレイアウト

ファイルシステムは次の場所にマウントされている：

```
/etc/pve
```

### 6.4.1 ファイル

authkey.pub	チケットシステムで使用される公開鍵
ceph.conf	Ceph設定ファイル（注：/etc/ceph/ceph.confへのシンボリックリンク）
corosync.conf	Corosyncクラスタ設定ファイル（以前のものProxmox VE 4.xでは、このファイルはcluster.confと呼ばれていました）
datacenter.cfg	Proxmox VEのデータセンター全体の構成（キーボードレイアウト、プロキシ、……）
domains.cfg	Proxmox VE認証ドメイン
ファイアウォール/クラスタ.fw	ファイアウォールの設定を全ノードに適用
ファイアウォール/<NAME>.fw	各ノードのファイアウォール設定
ファイアウォール/<VMID>.fw	VMとコンテナのファイアウォール設定
ha/crm_commands	現在実行中のHAオペレーションを表示するCRMが実施

ha/manager_status	HAに関するJSON形式の情報 クラスタ上のサービス
ha/resources.cfg	高可用性によって管理されるリソースとその現在の状態
nodes/<NAME>/config	ノード固有の設定
nodes/<NAME>/lxc/<VMID>.conf	LXCコンテナのVM設定データ
nodes/<NAME>/openvz/	Proxmox VE 4.0以前はコンテナに使用 コンフィギュレーション・データ（非推奨。）
nodes/<NAME>/pve-ssl.key	pve-ssl.pemのSSL秘密鍵

nodes/<NAME>/pve-ssl.pem	ウェブサーバー用のパブリックSSL証明書（署名者クラスタCA）
nodes/<NAME>/pveproxy-ssl.key	pveproxy-ssl.pemのSSL秘密鍵（オプション）
nodes/<NAME>/pveproxy-ssl.pem	ウェブサーバの公開SSL証明書（チェーン（pve-ssl.pemのオプションのオーバーライド）
nodes/<NAME>/qemu-server/<VMID>.conf	KVM VM用のnVfM設定データ
priv/authkey.key	チケットシステムで使用される秘密鍵
priv/authorized_keys	認証用クラスタメンバーのSSHキー
プライベート/セフ*	Ceph認証キーと関連する能力
priv/known_hosts	検証用クラスタ・メンバーのSSH鍵
priv/lock/*	様々なサービスで使用されるファイルをロックし、安全性を確保するクラスタ全体の運営
priv/pve-root-ca.key	クラスタCAの秘密鍵
priv/shadow.cfg	PVEレルムユーザー用シャドウパスワードファイル
priv/storage/<ストレージID>.pw	ストレージのパスワードをプレーンテキストで含む
priv/tfa.cfg	Base64エンコードされた二要素認証構成
priv/token.cfg	すべてのAPIトークンの秘密
pve-root-ca.pem	クラスタ CA のパブリック証明書
pve-www.key	CSRFトークンの生成に使用する秘密鍵
sdn/*	ソフトウェア定義のための共有設定ファイルネットワーキング（SDN）
ステータス.cfg	Proxmox VEの外部メトリクス・サーバー構成
storage.cfg	Proxmox VEのストレージ構成
ユーザー設定ファイル	Proxmox VEのアクセス制御設定（ユーザー/グループ/...）
仮想ゲスト/cpu-models.conf	カスタムCPUモデルの保存用
vzdump.cron	クラスタ全体のvzdumpバックアップジョブスケジュール

## 6.4.2 シンボリックリンク

クラスタ・ファイル・システム内の特定のディレクトリでは、ノード独自の設定ファイルを指すためにシンボリック・リンクが使用されます。したがって、以下の表で指すファイルはクラスタの各ノードで異なる

ローカル	nodes/<LOCAL_HOST_NAME>
エカルを指します。	nodes/<LOCAL_HOST_NAME>/lxc/
オープンベツツ	nodes/<LOCAL_HOST_NAME>/openvz/。 (非推奨、近日削除)
qemuサーバー	nodes/<LOCAL_HOST_NAME>/qemu-serv

### 6.4.3 デバッグ用の特別なステータス・ファイル (JSON)

バージョン	ファイルのバージョン (ファイルの変更を検出する )
メンバー	クラスタ・メンバーに関する情報
.vmlist	全VMのリスト

.clusterlog	クラスターログ（直近50件）
.rrd	RRDデータ（最新エントリー）

#### 6.4.4 デバッグの有効化／無効化

冗長なsyslogメッセージを有効にするには、次のようにする：

```
echo "1" >/etc/pve/.debug
```

そして、冗長なsyslogメッセージを無効にする：

```
echo "0" >/etc/pve/.debug
```

### 6.5 リカバリー

Proxmox VEホストにハードウェアの問題など大きな問題がある場合、pmxcfsデータベースファイル /var/lib/pve-cluster/config.dbをコピーし、新しいProxmox VEホストに移動すると便利です。新しいホストでは（何も実行していない状態で）、pve-cluster サービスを停止し、config.db ファイルを置き換える必要があります（必要なパーミッションは 0600）。これに続いて、/etc/hostname と

失われたProxmox VEホストに従って /etc/hosts を設定し、再起動して確認してください（VM/CTデータも忘れないでください）。

#### 6.5.1 クラスタ構成の削除

推奨される方法は、ノードをクラスタから削除した後に再インストールすることです。これにより、すべての秘密クラスタ/sshキーと共有設定データが確実に破棄されます。

場合によっては、[再インストール](#)せずにノードをローカルモードに戻したいこともあります。

#### 6.5.2 故障したノードからのゲストの回復/移動

nodes/<NAME>/qemu-server/ (VM)およびnodes/<NAME>/lxc/ (コンテナ)内のゲスト設定ファイルについて、Proxmox VEは各ゲストの所有者として含まれるノード<NAME>を認識します。これにより、ゲスト設定の同時変更を防止するために、高価なクラスタ全体のロックの代わりにローカルロックを使用できるようになります。

その結果、ゲストの所有ノードに障害が発生した場合(たとえば停電、フェンシングイベントなど)、(オフラインの)所有ノードのローカルロックが取得できないため、(すべてのディスクが共有ストレージにある場合でも)通常のマイグレーションはできません。Proxmox VEの高可用性スタックには、フェンスで保護されたノードからのゲストの正しい自動回復を保証するために必要な（クラスタ全体の）ロックとウォッチドッグ機能が含まれているため、これはHAで管理されたゲストの問題ではありません。

HA管理されていないゲストが共有ディスクしか持っていない場合(そして、障害ノードでのみ利用可能な他のローカルリソースを持っていない場合)、/etc/pve/内の障害ノードのディレクトリからオンラインノードのディレクトリにゲスト設定ファイルを移動するだけで、手動復旧が可能です(これはゲストの論理的所有者または場所を変更します)。

例えば、ID 100のVMをオフラインのノード1から別のノードnode2にリカバリするには、クラスタのメンバー・ノードでrootとして以下のコマンドを実行する：

```
mv /etc/pve/nodes/node1/qemu-server/100.conf /etc/pve/nodes/node2/ ←'qemu-server/
```

**警告**

このようにゲストを手動でリカバリする前に、障害が発生したソースノードが本当に電源オフ/フェンスされていることを絶対に確認してください。そうしないと、Proxmox VEのロック原則がmvコマンドによって侵害され、予期しない結果を招く可能性があります。

**警告**

ローカルディスク（またはオフラインノードでのみ利用可能な他のローカルリソース）を持つゲストは、この方法では復旧できません。障害が発生したノードがクラスタに復帰するのを待つか、バックアップからそのようなゲストをリストアしてください。

## 第7章

# プロックスモックスVEストレージ

Proxmox VEのストレージモデルは非常に柔軟です。仮想マシンイメージは、1つまたは複数のローカルストレージ、またはNFSやiSCSI（NAS、SAN）などの共有ストレージに保存できます。制限はなく、好きなだけストレージプールを構成できます。Debian Linuxで利用可能なすべてのストレージ技術を使用できます。

VMを共有ストレージに格納する大きなメリットの1つは、クラスタ内の全ノードがVMディスクイメージに直接アクセスできるため、ダウンタイムなしに稼働中のマシンをライブマイグレーションできることだ。VMイメージ・データをコピーする必要がないため、ライブ・マイグレーションは非常に高速である。

ストレージ・ライブラリ（libpve-storage-perlパッケージ）は、柔軟なプラグイン・システムを使用し、すべてのストレージ・タイプに共通なインターフェースを提供する。これは、将来的にさらなるストレージタイプを追加するために簡単に採用することができます。

## 7.1 ストレージの種類

ストレージの種類には基本的に2つのクラスがある：

### ファイルレベルのストレージ

ファイル・レベル・ベースのストレージ技術では、完全な機能を備えた（POSIX）ファイル・システムにアクセスできる。一般的に、ブロック・レベル・ストレージ（下記参照）よりも柔軟性が高く、あらゆるタイプのコンテンツを保存できる。ZFSはおそらく最も先進的なシステムで、スナップショットとクローンを完全にサポートしている。

## ブロックレベルのストレージ

大きなRAW画像を保存できる。通常、このようなストレージタイプに他のファイル（ISO、バックアップなど）を保存することはできない。最近のブロックレベルのストレージ実装のほとんどは、スナップショットとクローンをサポートしている。RADOSとGlusterFSは分散システムで、ストレージデータを異なるノードに複製します。

表7.1: 利用可能なストレージ・タイプ

説明	プラグインタ イプ	レベル	共有	スナップ写真	安定
ZFS（ローカル）	ゼットスプ ール	両方 <sup>1</sup>	いいえ	はい	はい
ディレクトリ	監督	ファイル	いいえ	いいえ <sup>2</sup>	はい
BTRFS	btrfs	ファイル	いいえ	はい	テクノロジー プレビュー

表7.1: (続き)

説明	プラグインタ イプ	レベル	共有	スナップ写真	安定
ネットワークファイルシステム	エヌエフエス	ファイル	はい	いいえ <sup>2</sup>	はい
CIFS	cifs	ファイル	はい	いいえ <sup>2</sup>	はい
Proxmoxバックアップ	pbs	両方	はい	該当なし	はい
GlusterFS	グラスター フ	ファイル	はい	いいえ <sup>2</sup>	はい
セフエフエス	ケフフ	ファイル	はい	はい	はい
LVM	エルブイエ ム	ブロック	いいえ <sup>3</sup>	いいえ	はい
LVMシン	ラヴムティ ン	ブロック	いいえ	はい	はい
iSCSI/カーネル	イッシ	ブロック	はい	いいえ	はい
iSCSI/libiscsi	イシディレッ ク	ブロック	はい	いいえ	はい
Ceph/RBD	アールブイ	ブロック	はい	はい	はい
iSCSI上のZFS	zfs	ブロック	はい	はい	はい

<sup>1</sup>: VMのディスク・イメージはZFSボリューム (zvol) データセットに格納され、ブロック・デバイス機能を提供する。

<sup>2</sup>: ファイルベースのストレージでは、qcow2フォーマットでスナップショットが可能です。

<sup>3</sup>: iSCSIやFCベースのストレージの上でLVMを使うことは可能だ。そうすれば、共有LVMストレージを手に入れることができる。

### 7.1.1 シン・プロビジョニング

多くのストレージと QEMU イメージフォーマット qcow2 はシンプロビジョニングをサポートしています。シンプロビジョニングを有効にすると、ゲストシステムが実際に使用するブロックのみがストレージに書き込まれます。

例えば、32GBのハードディスクでVMを作成し、ゲストシステムOSをインストールした後、VMのルートファイルシステムに3GBのデータが含まれているとします。この場合、ゲストVMが32GBのハードディスクを見ていたとしても、ストレージには3GBしか書き込まれません。このようにシン・プロビジョニングでは、現在利用可能なストレージ・ブロックよりも大きなディスク・イメージを作成することができます。VM用に大容量のディスク・イメージを作成し、必要なときにVMのファイル・システムのサイズを変更することなく、ストレージにディスクを追加することができます。

「スナップショット」機能を持つすべてのストレージタイプは、シン・プロビジョニングもサポートする。

### 注意

ストレージがいっぱいになると、そのストレージ上のボリュームを使用しているすべてのゲストがIO エラーを受け取ります。これはファイルシステムの不整合を引き起こし、データを破損する可能性があります。そのため、ストレージリソースの過剰なプロビジョニングを避けるか、空き容量を注意深く観察して、このような状態を回避することをお勧めします。

## 7.2 ストレージ構成

Proxmox VEに関するすべてのストレージ構成は、`/etc/pve/storage.cfg`の1つのテキストファイルに保存されます。このファイルは `/etc/pve/` 内にあるため、すべてのクラスタノードに自動的に配布されます。そのため、すべてのノードで同じストレージ構成が共有されます。

共有ストレージの構成は、すべてのノードから同じ "共有"ストレージにアクセスできるため、共有ストレージでは完全に理にかなっている。しかし、ローカルストレージにも有効です。この場合、ローカルストレージはすべてのノードで利用可能だが、物理的に異なっており、コンテンツもまったく異なる可能性がある。

## 7.2.1 ストレージ・プール

各ストレージプールは<type>を持ち、<STORAGE\_ID>で一意に識別される。プール構成は次のようになる：

```
<タイプ>: <ストレージID>
  <プロパティ> <値>
  <プロパティ> <値>
  <プロパティ>
  ...
```

<type>: <STORAGE\_ID>行がプール定義を開始し、それにプロパティのリストが続く。ほとんどのプロパティは値を必要とする。いくつかのプロパティは、妥当なデフォルトを持ち、その場合、値を省略することができる。

より具体的には、インストール後のデフォルトのストレージ構成を見てみよう。`local` という名前の特別なローカル・ストレージ・プールが1つ含まれ、これは `/var/lib/vz` ディレクトリを参照し、常に使用可能です。Proxmox VEのインストーラは、インストール時に選択したストレージタイプに応じて、追加のストレージエントリを作成します。

### デフォルトのストレージ設定 (`/etc/pve/storage.cfg`)

```
dir:ローカル
  パス /var/lib/vz
  コンテンツ iso,vztmpl,バックアップ

# LVM ベースのインストールにおけるデフォルトのイメージスト
ア lvmthin: local-lvm
  thinpoolデータ
  vgname pve
  コンテンツ rootdir,images

# ZFS ベースのインストールにおけるデフォルトのイメージスト
ア zfspool: local-zfs
  プール rpool/data
  スペース
  コンテンツ画像、ルートディレクトリ
```

---

### 注意



全く同じストレージを指す複数のストレージ構成があるのは問題である。このようなエイリアスされたストレージ構成では、2つの異なるボリュームID (*volid*) がまったく同じディスク・イメージを指すことになります。Proxmox VEは、イメージのボリュームIDが一意であることを期待します。エイリアス・ストレージ構成に異なるコンテンツ・タイプを選択することは問題ありませんが、推奨されません。

---

## 7.2.2 一般的なストレージ特性

いくつかのストレージ特性は、異なるストレージタイプ間で共通である。

## ノード

このストレージが使用可能/アクセス可能なクラスタ・ノード名のリスト。このプロパティを使用して、ストレージへのアクセスを限られたノードに制限できます。

## 内容

例えば、仮想ディスクイメージ、cdrom iso イメージ、コンテナテンプレート、コンテナルートディレクトリなどである。すべてのストレージタイプがすべてのコンテンツタイプをサポートするわけではない。このプロパティを設定することで、このストレージが何に使用されるかを選択できる。

## イメージ

QEMU/KVM VM イメージ。

### ルートディレクトリ

コンテナデータの保存を許可する。

### vztmpl

コンテナのテンプレート。

## バックアップ

バックアップファイル (vzdump)。

## アイソ

ISO イメージ

## スニペット

スニペットファイル (ゲストフックスクリプトなど)

## シェアード

すべてのノード（またはnodesオプションにリストされているすべて）で同じ内容を持つ単一のストレージであることを示します。ローカルストレージの内容が自動的に他のノードからアクセスできるようになるわけではなく、すでに共有されているストレージをそのようにマークするだけです！

## 無効にする

このフラグを使えば、ストレージを完全に無効にすることができます。

## マックスファイル

非推奨。代わりに prune-backups を使用してください。VMごとのバックアップファイルの最大数。使用方法

0なら無制限。

## バックアップの削除

バックアップの保持オプション。詳細については、[バックアップ保持](#)を参照してください。

## フォーマット

デフォルトの画像フォーマット (raw | qcow2 | vmdk)

## プレアロケーション

ファイルベースのストレージ上の raw および qcow2 画像に対するプリアロケーションモード (off | metadata | fallow | full)。デフォルトはメタデータで、raw 画像では off と同様に扱われます。ネットワークストレージと大きな qcow2 イメージを組み合わせて使用する場合、off を使用するとタイムアウトを回避できます。

**警告**

異なるProxmox VEクラスタで同じストレージプールを使用することはお勧めできません。ストレージ操作によってはストレージへの排他的アクセスが必要なため、適切なロックが必要です。これはクラスタ内では実装されていますが、異なるクラスタ間では機能しません。

## 7.3 卷数

ストレージデータのアドレス指定には、特別な表記を使用する。ストレージ・プールからデータを割り当てるとき、このようなボリューム識別子を再変換します。ボリュームは、<STORAGE\_ID>と、それに続くコロンで区切られたストレージ・タイプ依存のボリューム名で識別されます。有効な<VOLUME\_ID>は次のようになる:

```
local:230/example-image.raw
```

```
local:iso/debian-501-amd64-netinst.iso
```

```
local:vztmpl/debian-5.0-joomla_1.5.9-1_i386.tar.gz
```

```
iscsi-storage:0.0.2.scsi-14<'  
f504e46494c4500494b5042546d2d646744372d31616d61
```

<VOLUME\_ID>のファイルシステム・パスを取得するには、次のようにする:

```
pvesmパス <VOLUME_ID
```

### 7.3.1 数量所有権

イメージ・タイプのボリュームには所有関係が存在する。このようなボリュームはそれぞれ、VMまたはコンテナによって所有される。例えば、ボリュームlocal:230/example-image.rawはVM 230が所有する。ほとんどのストレージバックエンドは、この所有者情報をボリューム名にエンコードします。

VMまたはコンテナを削除すると、そのVMまたはコンテナが所有する関連ボリュームもすべて削除されます。

## 7.4 コマンドラインインターフェイスの使用

ストレージ・プールやボリューム識別子の概念に慣れておくことをお勧めするが、実際の運用では、コマンド・ラインでそのような低レベルの操作を行うことはない。通常、ボリュームの割り当てと削除はVMとコンテナの管理ツールによって行われる。

とはいって、`pvesm`（「Proxmox VE Storage Manager」）と呼ばれるコマンドラインツールがあり、一般的なストレージ管理タスクを実行することができる。

## 7.4.1 例

ストレージプールの追加

```
pvesm add <TYPE> <STORAGE_ID> <OPTIONS>
pvesm add dir <STORAGE_ID> --path <PATH>
pvesm add nfs <STORAGE_ID> --path <PATH> --server <SERVER> --export <'>
    <エクスポート>
pvesm add lvm <STORAGE_ID> --vgname <VGNAME>
pvesm add iscsi <STORAGE_ID> --portal <HOST[:PORT]> --target <TARGET <'>
    >
```

ストレージプールを無効にする

```
pvesm set <STORAGE_ID> --disable 1
```

ストレージプールを有効にする

```
pvesm set <STORAGE_ID> --disable 0
```

ストレージオプションの変更/設定

```
pvesm set <STORAGE_ID> <OPTIONS>
pvesm set <STORAGE_ID> --shared 1
pvesm set local --format qcow2
pvesm set <STORAGE_ID> --content
iso
```

ストレージプールを削除する。これはいかなるデータも削除しないし、何かを切断したりアンマウントしたりもない。ストレージ構成を削除するだけです。

`pvesm remove <STORAGE_ID>` を削除する。

ボリュームの割り当て

```
pvesm alloc <STORAGE_ID> <VMID> <name> <size> [--format <raw|qcow2>].
```

ローカル・ストレージに 4G ボリュームを割り当てます。`<name> pvesm alloc local <VMID>`  
'' 4G として空の文字列を渡すと、名前が自動生成されます。

全巻無料

```
pvesm free <VOLUME_ID>
```

**警告**

これは本当にすべてのボリュームデータを破壊する。

---

ストレージの状態を一覧表示

`pvesm status`

## 収納内容リスト

```
pvesm list <STORAGE_ID> [--vmid <VMID>]
```

VMIDで割り当てられたボリュームを一覧表示

```
pvesm list <STORAGE_ID> --vmid <VMID>
```

## リストisoイメージ

```
pvesm list <STORAGE_ID> --content iso
```

## リスト・コンテナ・テンプレート

```
pvesm list <STORAGE_ID> --content vztmpl
```

ボリュームのファイルシステムパスを表示する

```
pvesmパス <VOLUME_ID>
```

ボリュームlocal:103/vm-103-disk-0.qcow2をファイルターゲットにエクスポートします。これは主にpvesmインポートで内部的に使用されます。ストリーム形式qcow2+sizeはqcow2形式とは異なります。そのため、エクスポートしたファイルを単純にVMにアタッチすることはできません。これは他の形式でも同様です。

```
pvesm export local:103/vm-103-disk-0.qcow2 qcow2+size target --with-'  
スナップ写真 1
```

## 7.5 ディレクトリ・バックエンド

ストレージ・プールのタイプ: dir

Proxmox VEは、ローカルディレクトリまたはローカルにマウントされた共有をストレージとして使用できます。ディレクトリはファイルレベルのストレージであるため、仮想ディスクイメージ、コンテナ、テンプレート、ISOイメージ、バックアップファイルなど、あらゆる種類のコンテンツを保存できます。

### 注

標準的なLinuxの/etc/fstabを使って追加ストレージをマウントし、そのマウントポイントにディレクトリストレージを定義することができる。こうすることで、Linuxでサポートされているあらゆるファイルシステムを使うことができる。

このバックエンドは、基礎となるディレクトリがPOSIX互換であることを前提としているが、それ以外は何も想定していない。これは、ストレージ・レベルでスナップショットを作成できないことを意味する。しかし、qcow2ファイル・フォーマットを使用しているVMイメージについては、このフォーマットが内部的にスナップショットをサポートしているため、回避策があります。

---

### チップ

いくつかのストレージ・タイプは`O_DIRECT`をサポートしていないので、そのようなストレージではキャッシュ・モード`none`は使えない。代わりにキャッシュ・モード・ライトバックを使用してください。

---

私たちは、異なるコンテンツタイプを異なるサブディレクトリに格納するために、事前定義されたディレクトリレイアウトを使用しています。このレイアウトは、すべてのファイルレベルストレージバックエンドで使用されます。

表7.2: ディレクトリのレイアウト

コンテンツタイプ	サブディレクトリ
VMイメージ	images/<VMID>/
ISOイメージ	テンプレート/iso/
コンテナ・テンプレート	テンプレート/キャッシュ
バックアップファイル	ダンプ
スニペット	スニペット

### 7.5.1 構成

このバックエンドは、一般的なストレージ・プロパティをすべてサポートし、さらに2つのプロパティを追加している。パス

プロパティを使用してディレクトリを指定します。これは絶対ファイル・システム・パスである必要があります。

オプションのcontent-dirsプロパティは、デフォルトのレイアウトを変更することができます。このプロパティは、コンマで区切られた以下の形式の識別子のリストで構成されます：

vtype=パス

vtypeはストレージに許可されるコンテンツタイプの1つで、pathはストレージのマウントポイントからの相対パスである。

#### 設定例 (/etc/pve/storage.cfg)

```
dir:バックアップ
  パス /mnt/backup
  内容 バックアップ
  バックアップの削除 keep-
  last=7 最大保護バックアップ
  数 3
  content-dirs backup=custom/backup/dir
```

上記の構成では、backupというストレージ・プールを定義している。このプールを使用して、VMごとに最大7つの通常バックアップ (keep-last=7) と3つの保護バックアップを保存できる。バックアップ・ファイルの実際のパスは

/mnt/backup/custom/backup/dir/....

## 7.5.2 ファイルの命名規則

このバックエンドは、VMイメージに対して明確に定義された命名スキームを使用する：

vm-<VMID>-<NAME>.<フォーマット

### <VMID>

これはオーナーVMを指定する。

### <名前>

これは空白のない任意の名前 (ascii) である。 バックエンドはdisk-[N]をデフォルトでは、[N]は名前を一意にするために整数で置き換えられる。

## <フォーマット

画像フォーマット (raw|qcow2|vmdk) を指定する。

VM テンプレートを作成すると、すべての VM イメージの名前が変更され、読み取り専用になり、クローン用のベースイメージとして使用できるようになります：

ベース<VMID>-<NAME>.<FORMAT>

### 注

このようなベース画像は、クローン画像を生成するために使用されます。そのため、これらのファイルは読み取り専用で、決して変更されないことが重要です。バックエンドはアクセスモードを0444に変更し、ストレージが対応していればimmutableフラグ (chattr +i) を設定します。

### 7.5.3 ストレージ機能

上述したように、ほとんどのファイルシステムは、そのままではスナップショットをサポートしていない。この問題を回避するために、このバックエンドはqcow2内部のスナップショット機能を使うことができます。

クローンも同様です。バックエンドはqcow2のベースイメージ機能を使ってクローンを作成します。

表7.3：バックエンド・ディアのストレージ機能

コンテンツの種類	画像フォーマット	共有	スナップ写真	クローン
イメージ ルートディレクトリ vztmp1 iso バックアップ スニペット	生のqcow2 vmdkサブボリューム	いいえ	qcow2	qcow2

### 7.5.4 例

以下のコマンドを使用して、ローカルストレージに4GBのイメージを割り当ててください： # pvesm alloc local 100 vm-100-disk10.raw  
4G

```
フォーマット '/var/lib/vz/images/100/vm-100-disk10.raw', fmt=raw size='
=4294967296
local:100/vm-100-disk10.raw」の作成に成功。
```

---

### 注

画像名は上記の命名規則に従わなければならない。

---

実際のファイルシステムのパスは次のように表示される：

```
# pvesm パス local:100/vm-100-disk10.raw  
/var/lib/vz/images/100/vm-100-disk10.raw
```

で画像を削除できる：

```
# pvesm free local:100/vm-100-disk10.raw
```

## 7.6 NFSバックエンド

ストレージ・プールのタイプ: nfs

NFSバックエンドはディレクトリバックエンドをベースにしているので、ほとんどのプロパティを共有している。ディレクトリのレイアウトやファイルの命名規則も同じです。主な利点は、NFSサーバーのプロパティを直接設定できるので、バックエンドが自動的に共有をマウントできることです。 を変更する必要はありません。

/etc/fstabを使用する。バックエンドはサーバーがオンラインかどうかをテス<sup>ト</sup>でき、エクスポートされた共有をサーバーに問い合わせる方法も提供する。

### 7.6.1 構成

バックエンドは、常に設定される共有フラグを除く、すべての一般的なストレージ・プロパティをサポートする。さらに、以下のプロパティがNFSサーバーの設定に使用される：

#### サーバー

サーバーIPまたはDNS名。DNSルックアップの遅延を避けるため、通常はDNS名ではなくIPアドレスを使用するのが望ましい。

/etc/hostsファイル。

#### 輸出

NFS エクスポート・パス (pvesm nfsscan によってリストされる)。

NFSマウント・オプションを設定することもできる：

#### パス

ローカルのマウントポイント（デフォルトは /mnt/pve/<STORAGE\_ID>/）。

### コンテンツ・ディレクトリ

デフォルトのディレクトリレイアウトのオーバーライド。オプション。

### オプション

NFSマウントオプション（man nfs を参照）。

### 設定例 (`/etc/pve/storage.cfg`)

```
nfs: iso-テンプレート
    パス /mnt/pve/iso-templates +
    一バー 10.0.0.10
    export /space/iso-templates
    options vers=3,ソフトコンテ
    ンツiso,vztmp1
```

#### チップ

NFSリクエストがタイムアウトした後、NFSリクエストはデフォルトで無期限に再試行される。これは、クライアント側で予期しないハングアップを引き起こす可能性がある。読み取り専用コンテンツについては、再試行回数を3回に制限するNFSソフトオプションを検討する価値がある。

### 7.6.2 ストレージ機能

NFSはスナップショットをサポートしないが、バックエンドはqcow2の機能を使ってスナップショットとクローンを実装する。

表7.4: バックエンドnfsのストレージ機能

コンテンツの種類	画像フォーマット	共有	スナップ写真	クローン
イメージ ルートディレクトリ vztmp1 iso バックアップ スニペット	生のqcow2 ブイエムディーケー	はい	qcow2	qcow2

### 7.6.3 例

でエクスポートされたNFS共有のリストを取得できます:

```
# pvesm nfsscan <server>
```

## 7.7 CIFSバックエンド

ストレージ・プール・タイプ: `cifs`

CIFSバックエンドはディレクトリバックエンドを拡張するため、CIFSマウントの手動設定は不要です。このようなストレージはProxmox VE APIまたはWeb UIから直接追加でき、サーバーのハートビート確認やエクスポートされた共有の快適な選択など、バックエンドの利点をすべて利用できます。

## 7.7.1 構成

バックエンドは、常に設定される共有フラグを除く、すべての一般的なストレージ・プロパティをサポートしている。さらに、以下のCIFS特殊プロパティが利用可能である：

### サーバー

サーバーIPまたはDNS名。必須。

---

### チップ

DNSルックアップの遅延を避けるため、通常はDNS名ではなくIPアドレスを使用するのが望ましい-よほど信頼できるDNSサーバーがあるか、ローカルの/etc/hostsファイルにサーバーをリストしている場合を除く。

---

### シェア

使用するCIFS共有 (pvesm scan cifs <address>またはWeb UIで利用可能なものを取得)。必要です。

### ユーザー名

CIFSストレージのユーザー名。オプション。デフォルトは「guest」。

### パスワード

ユーザーのパスワード。任意。rootだけが読めるファイルに保存されます (/etc/pve/priv/storage/)。

### ドメイン

このストレージのユーザー・ドメイン（ワークグループ）を設定する。オプション。

### スミバージョン

SMBプロトコルのバージョン。オプション、デフォルトは3。SMB1はセキュリティ上の問題からサポートされていない。

### パス

ローカルのマウントポイント。オプションで、デフォルトは /mnt/pve/<STORAGE\_ID>/です。

### コンテンツ・ディレクトリ

デフォルトのディレクトリレイアウトのオーバーライド。オプション。

---

## オプション

追加のCIFSマウント・オプション (`man mount.cifs`を参照)。一部のオプションは自動的に設定されるため、ここで設定する必要はありません。Proxmox VEは常にソフトオプションを設定します。設定に応じて、これらのオプションは自動的に設定されます: ユーザー名、資格情報、ゲスト、ドメイン、vers。

### サブディレクトリ

マウントする共有のサブディレクトリ。オプション。デフォルトは共有のルート・ディレクトリ。

### 設定例 (/etc/pve/storage.cfg)

```
cifs: バックアップ
    パス /mnt/pve/backup
    サーバー 10.0.0.11 共
    有 VMData コンテンツ
    バックアップ
    オプション noserverino,echo_interval=30 ユー
    ザー名 anna
    smbversion
    3 subdir
    /data
```

### 7.7.2 ストレージ機能

CIFSはストレージレベルでのスナップショットをサポートしていない。しかし、スナップショットやクローン機能を利用したい場合は、qcow2バックティングファイルを使用することができる。

表7.5: バックエンドcifsのストレージ機能

コンテンツの種類	画像フォーマット	共有	スナップ写真	クローン
イメージ ルートディレクトリ vztmpl iso バックアップ スニペット	生のqcow2 ブイエムディーケー	はい	qcow2	qcow2

### 7.7.3 例

でエクスポートされたCIFS共有のリストを取得できます:

```
# pvesm scan cifs <server> [--username <username>] [--password] # pvesm scan
cifs <server> [--username <username>] [--password]
```

そして、この共有をProxmox VEクラスタ全体のストレージとして追加します:

```
# pvesm add cifs <storagename> --server <server> --share <share> [--<
ユーザー名 <ユーザー名>] [パスワード]
```

## 7.8 Proxmoxバックアップサーバー

ストレージ・プールのタイプ: pbs

このバックエンドにより、他のストレージと同様に Proxmox Backup Server を Proxmox VE に直接統合できます。Proxmox BackupストレージはProxmox VEのAPI、CLI、またはWebインターフェイスから直接追加できます。

## 7.8.1 構成

バックエンドは、常に設定される共有フラグを除く、すべての一般的なストレージプロパティをサポートしています。さらに、Proxmox Backup Serverには以下の特別なプロパティがあります：

### サーバー

サーバーIPまたはDNS名。必須。

### ポート

デフォルトのポート（8007）ではなく、このポートを使用する。オプション。

### ユーザー名

Proxmox Backup Serverストレージのユーザー名。必須です。

---

### チップ

ユーザー名にレルムを追加することを忘れないでください。例えば、root@pamやarchiver@pbsのように。  
。

---

### パスワード

ユーザー・パスワード。この値は、/etc/pve/priv/storage/<STORAGE-ID> 以下のファイルに保存されます。

へのアクセスをrootユーザーに制限する。必須。

### データストア

使用するProxmox Backup ServerデータストアのID。必須。

### フィンガープリント

Proxmox Backup Server API TLS 証明書のフィンガープリント。Servers Dashboard または proxmox-backup-manager cert infoコマンドで取得できます。自己署名証明書またはホストがサーバーのCAを信頼していないその他の証明書に必要です。

### 暗号化キー

クライアント側からバックアップ・データを暗号化するためのキー。現在、非パスワード保護（キー派生関数（kdf）なし）のみがサポートされています。etc/pve/priv/storage/<STORAG>下のファイルに保存され、アクセスはrootユーザーに制限されます。マジックバリュー autogen を使って、自動的に

---

`proxmox-backup-client key create --kdf none <path>`を使って新しい鍵を作成します。オプション。

## マスターパブキー

バックアップタスクの一部としてバックアップ暗号化キーを暗号化するために使用される公開RSAキー。暗号化されたコピーはバックアップに追加され、リカバリ用にProxmox Backup Serverインスタンスに保存されます。オプションで、`encryption-key`が必要です。

### 設定例 (/etc/pve/storage.cfg)

```
pbs: バックアップ
    データストア・メイン
    サーバー
    enya.proxmox.com コンテ
        ンツバックアップ
        指紋09:54:ef:...スニップ...:88:af:47:fe:4c:3b:cf:8b:26:88:0b:4e:3 ←'
        c:b2
        prune-backups keep-all=1 ユー
        ザー名 archiver@pbs
```

### 7.8.2 ストレージ機能

Proxmox Backup Serverはブロックレベルまたはファイルレベルのバックアップのみをサポートします。

Proxmox VEは仮想マシンにブロックレベル、コンテナにファイルレベルを使用します。

表7.6: バックエンドpbsのストレージ機能

コンテンツの種類	画像フォーマット	共有	スナップ写真	クローン
バックアップ	該当なし	はい	該当なし	該当なし

### 7.8.3 暗号化



オプションとして、GCM モードで AES-256 によるクライアント側の暗号化を設定することができます。暗

号化は、Web インターフェース、または CLI で `encryption-key` オプション（上記参照）を使用して設定できます。キーは、root ユーザのみがアクセスできる `/etc/pve/priv/storage/<STORAGE-ID>.enc` ファイルに保存されます。

---

### 警告



キーがなければ、バックアップにアクセスできない。したがって、キーは順序立てて、バックアップするコンテンツとは別の場所に保管する必要がある。例えば、システム全体をバックアップし、そのシステムのキーを使用することがあります。その後、何らかの理由でシステムにアクセスできなくなり、復元する必要が生じた場合、暗号化キーは壊れたシステムとともに失われてしまうため、復元は不可能である。

迅速な災害復旧のために、鍵は安全に、しかし簡単にアクセスできる場所に保管することをお勧めします。そのため、すぐに復旧できるパスワード・マネージャー内に保管するのが最適です。このバックアップとして、キーをUSBフラッシュドライブに保存し、安全な場所に保管することもお勧めします。こうすることで、どのシステムからも切り離されるものの、万が一の場合に復旧が容易となる。最後に、最悪のシナリオに備えて、キーの紙コピーを安全な場所に保管しておくことも検討すべきである。`paperkey`サブコマンドを使えば、QRコード化された鍵のバージョンを作成することができる。次のコマンドは、`paperkey`コマンドの出力をテキスト・ファイルに送り、簡単に印刷できるようにしたものである。

```
# proxmox-backup-client key paperkey /etc/pve/priv/storage/<STORAGE-ID>.enc --output-format text > qrkey.txt
```

暗号化バックアップを行うすべてのクライアントが单一のパブリック・マスター・キーを使用するように設定すると、それ以降のすべての暗号化バックアップには、使用されたAES暗号化キーのRSA暗号化コピーが含まれます。対応するプライベート・マスター・キーにより、クライアント・システムが利用できなくなつた場合でも、AESキーを復元し、バックアップを復号化することができます。

---

#### 警告

マスターキー・ペアには、通常の暗号化キーと同じ保管ルールが適用される。秘密鍵のコピーがなければ、復元は不可能である！`paperkey`コマンドは、安全な物理的場所に保管するために、秘密鍵のマスター・キーの紙コピーの生成をサポートする。

暗号化はクライアント側で管理されるため、暗号化されていないバックアップと暗号化されたバックアップが異なるキーで暗号化されても、サーバー上の同じデータストアを使用することができます。しかし、異なるキーで暗号化されたバックアップ間の重複排除は不可能であるため、データストアは別々に作成した方がよい場合が多い。

---

#### 注

例えば、信頼できるネットワークでローカルにサーバーを運用している場合など、暗号化によるメリットがない場合は暗号化を使用しないでください。暗号化されていないバックアップからの復旧の方が常に簡単です。

---

### 7.8.4 例CLIによるストレージの追加

そして、この共有をProxmox VEクラスタ全体のストレージとして追加します：

```
# pvesm add pbs <id> --server <server> --datastore <datastore> --username <username> --fingerprint 00:B4:....-パスワード
```

## 7.9 GlusterFSバックエンド

ストレージプールの種類: glusterfs

GlusterFSはスケーラブルなネットワークファイルシステムである。このシステムはモジュラーデザインを採用し、コモディティハードウェア上で動作し、低コストで可用性の高いエンタープライズストレージを提供することができる。このシステムは数ペタバイトまで拡張可能で、数千のクライアントを扱うことができる。

---

### 注

ノードやブリックがクラッシュした後、GlusterFSはデータの一貫性を確保するために完全なrsyncを行う。これは大きなファイルだと非常に時間がかかるので、このバックエンドは大きなVMイメージの保存には適さない。

## 7.9.1 構成

バックエンドはすべての一般的なストレージ・プロパティをサポートし、以下のGlusterFS固有のオプションを追加する：

### サーバー

GlusterFS volfileサーバーのIPまたはDNS名。

### サーバー2

バックアップファイルサーバーのIPまたはDNS名。

### ボリューム

GlusterFSボリューム。

### 輸送

GlusterFSトランSPORT: tcp、 unixまたはrdma

### 設定例 (/etc/pve/storage.cfg)

```
glusterfs: クラスタ
    サーバー 10.2.3.4
    server2 10.2.3.5
    volume glustervol
    content
    images, iso
```

## 7.9.2 ファイルの命名規則

ディレクトリのレイアウトとファイルの命名規則はdirバックエンドから継承されます。

## 7.9.3 ストレージ機能

ストレージはファイルレベルのインターフェイスを提供するが、ネイティブのスナップショット/クローン実装はない。

表7.7: バックエンドglusterfsのストレージ機能

コンテンツの種類	画像フォーマット	共有	スナップ写真	クローン
イメージ vztmp1 iso バックアップ	生のqcow2 ブイエムディーケー	はい	qcow2	qcow2

スニペット				
-------	--	--	--	--

## 7.10 ローカルZFSプール・バックエンド

ストレージ・プール・タイプ: `zfspool`

このバックエンドを使うと、ローカルのZFSプール（またはプール内のZFSファイルシステム）にアクセスできる。

## 7.10.1 構成

バックエンドは、一般的なストレージ・プロパティであるcontent、nodes、disableと、以下のZFS固有のプロパティをサポートしています：

### プール

ZFSプール/ファイルシステムを選択する。すべての割り当てはそのプール内で行われる。

### ロックサイズ

ZFSロックサイズ・パラメータを設定する。

### まばら

ZFSシン・プロビジョニングを使用する。スパースボリュームとは、予約がボリュームサイズに等しくないボリュームのことです。

### マウントポイント

ZFSプール/ファイルシステムのマウントポイント。これを変更しても、`zfs`が見るデータセットのマウントポイント・プロパティには影響しません。デフォルトは `/<pool>` です。

### 設定例 (`/etc/pve/storage.cfg`)

```
zfspool: vmdata
    プール tank/vmdata コン
    テンツ rootdir,images
    sparse
```

## 7.10.2 ファイルの命名規則

バックエンドはVMイメージに以下のような命名スキームを使用する：

`vm-<VMID>-<NAME>/` // 通常のVMイメージ

`base-<VMID>-<NAME>/` // テンプレートVMイメージ（読み取り専用）

`subvol-<VMID>-<NAME>` // サブボリューム（コンテナ用ZFSファイルシステム）

これはオーナーVMを指定する。

#### <名前

これは空白のない任意の名前 (ascii) である。バックエンドはデフォルトで `disk[N]` を使用します、ここで [N] は、名前を一意にするために整数に置き換えられている。

### 7.10.3 ストレージ機能

ZFSは、スナップショットとクローニングに関して最も先進的なストレージタイプだろう。バックエンドは、VMイメージ（フォーマットraw）とコンテナ・データ（フォーマットsubvol）の両方にZFSデータセットを使用する。ZFSのプロパティは親データセットから継承されるため、親データセットにデフォルトを設定するだけでよい。

表7.8: バックエンド`zfs`のストレージ機能

コンテンツの種類	画像フォーマット	共有	スナップ写真	クローン
イメージ ルートディレクトリ	生サブヴォル	いいえ	はい	はい

### 7.10.4 例

VMイメージを保存するために、追加のZFSファイルシステムを作成することを推奨する：

```
# zfs create tank/vmdata
```

新しく割り当てられたファイルシステムで圧縮を有効にする：

```
# zfs set compression=on tank/vmdata
```

利用可能なZFSファイルシステムのリストは

```
# pvesm zfsscan
```

## 7.11 LVMバックエンド

ストレージ・プールのタイプ: lvm

LVMは、ハードディスクとパーティションの上にある軽いソフトウェアレイヤーである。利用可能なディスクスペースをより小さな論理ボリュームに分割するために使用できる。LVMはLinuxで広く使われており、ハードディスクの管理を容易にします。

もう一つの使用例は、大きなiSCSI LUNの上にLVMを置くことです。そうすれば、iSCSI LUN上のスペース

を簡単に管理することができます。iSCSI仕様では、スペース割り当てのための管理インターフェイスが定義されていないため、他の方法では不可能です。

### 7.11.1 構成

LVMバックエンドは、一般的なストレージ・プロパティであるcontent、nodes、disableと、以下のLVM固有のプロパティをサポートしている：

#### ブグネーム

LVMボリューム・グループ名。これは既存のボリューム・グループを指す必要があります。

## ベース

ベースボリューム。このボリュームは、ストレージにアクセスする前に自動的にアクティブ化される。これは主に、LVMボリュームグループがリモートのiSCSIサーバー上に存在する場合に便利です。

## セーフリムーブ

ウェブUIで "Wipe Removed Volumes" と呼ばれる。LVの削除時にデータをゼロにする。ボリュームを削除する際、すべてのデータが消去され、後から作成された他のLV（たまたま同じ物理エクステントが割り当てられている）からアクセスできないようにします。これはコストのかかる操作ですが、特定の環境ではセキュリティ対策として必要な場合があります。

## セーフリムーブ・スループット

ワイプスループット (cstream -t パラメータ値)。

### 設定例 (/etc/pve/storage.cfg)

```
lvm: マイスペース  
      vgname myspace コンテン  
      ツ rootdir,images
```

## 7.11.2 ファイルの命名規則

バックエンドは、基本的にZFSプールのバックエンドと同じ命名規則を使用する。

vm-<VMID>-<NAME> // 通常のVMイメージ

## 7.11.3 ストレージ機能

LVMは典型的なブロック・ストレージだが、このバックエンドはスナップショットとクローンをサポートしていない。残念なことに、通常のLVMスナップショットは、スナップショット時間中にボリュームグループ全体のすべての書き込みに干渉するため、かなり非効率的です。

大きな利点のひとつは、iSCSI LUNなどの共有ストレージの上で使えることだ。バックエンド自体が適切なクラスタ全体のロックを実装している。

新しいLVM-thinバックエンドはスナップショットとクローンを可能にするが、共有ストレージはサポートしない。

---

表7.9: バックエンド`lvm`のストレージ機能

コンテンツの種類	画像フォーマット	共有	スナップ写真	クローン
イメージ ルートディレクトリ	生	可能	いいえ	いいえ

### 7.11.4 例

利用可能なボリュームグループを一覧表示します:

```
# pvesm lvmscan
```

## 7.12 LVMシン・バックエンド

ストレージ・プールのタイプ: `lvmthin`

LVMは通常、ボリュームの作成時にブロックを割り当てます。LVMのシンプールは、代わりに書き込み時にブロックを割り当てます。ボリュームは物理的に利用可能なスペースよりもはるかに大きくなる可能性があるため、この動作はシン・プロビジョニングと呼ばれます。

通常のLVMコマンドラインツールを使用して、LVMシンプールを管理および作成することができます（詳細については、`man lvmthin`を参照してください）。`pve`というLVMボリュームグループが既にあると仮

```
lvcreate -L 100G -n data pve  
lvconvert --type thin-pool pve/data
```

定すると、以下のコマンドは`data`という新しいLVMシンプール(サイズ100G)を作成します:

### 7.12.1 構成

LVMシンバックエンドは、一般的なストレージプロパティである`content`、`nodes`、`disable`と、以下のLVM固有のプロパティをサポートします:

#### ブグネーム

LVMボリューム・グループ名。これは既存のボリューム・グループを指す必要があります。

#### シンプール

LVMシンプールの名前。

#### 設定例 (`/etc/pve/storage.cfg`)

```
lvmthin: ローカルlvm  
        thinpoolデータ  
        vgname pve  
        コンテンツ rootdir,images
```

### 7.12.2 ファイルの命名規則

バックエンドは、基本的にZFSプールのバックエンドと同じ命名規則を使用する。

vm-<VMID> >-<NAME> //通常のVMイメージ

### 7.12.3 ストレージ機能

LVM thinはブロックストレージだが、スナップショットやクローンを効率的にサポートする。新しいボリュームは自動的にゼロで初期化される。

LVMシン・プールは複数のノードで共有することができないため、ローカル・ストレージとしてのみ使用することができます。

表 7.10: バックエンド lvmthin のストレージ機能

コンテンツの種類	画像フォーマット	共有	スナップ写真	クローン
イメージ ルートディレクトリ	生	いいえ	はい	はい

### 7.12.4 例

ボリュームグループpveで利用可能なLVMシンプールをリストアップします:

```
# pvesm lvmthinscan pve
```

## 7.13 Open-iSCSIイニシエータ

ストレージプールのタイプ: iscsi

iSCSIは、ストレージ・サーバーへの接続に広く採用されている技術である。ほとんどすべてのストレージ・ベンダーがiSCSIをサポートしている。Debianベースの[OpenMediaVault](#)など、オープンソースのiSCSIターゲットソリューションもある。

このバックエンドを使用するには、[Open-iSCSI](#) (open-iscsi) パッケージをインストールする必要があります。これは Debian の標準パッケージですが、リソースを節約するためにデフォルトではインストールされません。

```
# apt-get install open-iscsi
```

低レベルのiscsi管理タスクはiscsiadmツールを使って行うことができる。

### 7.13.1 構成

バックエンドは、一般的なストレージ・プロパティであるcontent、nodes、disableと、以下のiSCSI固有のプロパティをサポートしている：

#### ポータル

iSCSIポータル（IPまたはDNS名とオプションのポート）。

#### ターゲット

iSCSI ターゲット。

### 設定例 (`/etc/pve/storage.cfg`)

```
iSCSI: マイナス
  ポータル 10.10.10.1
  ターゲット iqn.2006-01.openfiler.com:tsn.dcb5aaaddd 口
  コンテンツ none
```

#### チップ

iSCSIの上でLVMを使いたい場合は、`content none`を設定するのが理にかなっている。そうすることで、iSCSI LUNを直接使用してVMを作成することができなくなる。

### 7.13.2 ファイルの命名規則

iSCSIプロトコルは、データを割り当てたり削除したりするインターフェースを定義していない。代わりに、それはターゲット側で行われる必要があり、ベンダー固有である。ターゲットは単に番号付きLUNとしてエクスポートします。つまり、Proxmox VEのiSCSIボリューム名は、Linuxカーネルから見たLUNに関する情報をエンコードしているだけです。

### 7.13.3 ストレージ機能

iSCSIはブロック・レベル・タイプのストレージで、管理インターフェイスを提供しない。そのため、通常は1つの大きなLUNをエクスポートし、そのLUNの上にLVMをセットアップするのがベストだ。その後、LVMプラグインを使ってiSCSI LUN上のストレージを管理することができます。

表7.11: バックエンド`iscsi`のストレージ機能

コンテンツの種類	画像フォーマット	共有	スナップ写真	クローン
画像なし	生	はい	いいえ	いいえ

### 7.13.4 例

リモートのiSCSIポータルをスキャンし、可能なターゲットのリストを返す：

`pvesm scan iscsi <ホスト[:ポート]>`をスキャンする。

## 7.14 ユーザーモードiSCSIバックエンド

ストレージ・プール・タイプ: `iscidirect`

このバックエンドは基本的に Open-iSCSI バックエンドと同じ機能を提供しますが、実装にはユーザレベルのライブラリを使用します。このバックエンドを使うには `libiscsi-bin` パッケージをインストールする必要があります。

カーネル・ドライバが関与していないので、これはパフォーマンスの最適化とみなすことができる。しかしこれには、このようなiSCSI LUNの上でLVMを使えないという欠点がある。そのため、ストレージ・サーバー側ですべてのスペースの割り当てを管理する必要がある。

### 7.14.1 構成

ユーザー モード iSCSI バックエンドは、Open-iSCSI バックエンドと同じ設定オプションを使用します。

#### 設定例 (/etc/pve/storage.cfg)

```
iscsidirect: ファストス  
トア・ポータル  
10.10.10.1  
ターゲット iqn.2006-01.openfiler.com:tsn.dcb5aaadd
```

### 7.14.2 ストレージ機能

#### 注

このバックエンドはVMでのみ動作する。コンテナはこのドライバを使用できない。

表7.12: バックエンド iscsidirect のストレージ機能

コンテンツの種類	画像フォーマット	共有	スナップ写真	クローン
イメージ	生	はい	いいえ	いいえ

## 7.15 Ceph RADOSブロックデバイス (RBD)

ストレージ・プール・タイプ: rbd

Cephは、優れたパフォーマンス、信頼性、スケーラビリティを提供するように設計された分散オブジェクトストアおよびファイルシステムです。RADOSブロックデバイスは機能豊富なブロックレベルストレージを実装しており、以下のような利点があります：

- ・ シンプロビジョニング
- ・ サイズ変更可能なボリューム
- ・ 分散および冗長（複数のOSDにストライピングされる）
- ・ 完全なスナップショットとクローン機能

- 自己治癒
- 単一障害点なし
- エクサバイトレベルまで拡張可能
- カーネルおよびユーザー空間の実装が可能

---

### 注

小規模な導入では、Proxmox VE ノードで Ceph サービスを直接実行することも可能です。最近のハードウェアは CPU パワーと RAM に余裕があるため、ストレージサービスと VM を同じノードで実行できます。

---

### 7.15.1 構成

このバックエンドは、一般的なストレージ・プロパティであるノード、ディセーブル、コンテンツ、および以下をサポートしています。

RBD特有の特性:

#### モノホスト

モニターデーモンのIPのリスト。オプション。CephがProxmox VEクラスタで実行されていない場合にのみ必要です。

#### プール

Cephプール名。

#### ユーザー名

RBDユーザID。オプション。CephがProxmox VEクラスタで実行されていない場合にのみ必要です。

ユーザIDのみを使用することに注意してください。client. "タイプの接頭辞は省略する必要があります。

#### クルード

krbdカーネルモジュールを通してradosブロックデバイスへのアクセスを強制する。オプション。

---

#### 注

コンテナは、オプション値とは無関係にkrbdを使用する。

---

#### 外部Cephクラスタの構成例(/etc/pve/storage.cfg)

##### RBD: セフ外部

```
monhost 10.1.1.20 10.1.1.21 10.1.1.22
pool ceph-
external content
images ユーザー名
admin
```

---

#### チップ

rbdユーティリティを使って、低レベルの管理作業を行うことができます。

---

## 7.15.2 認証

---

### 注

Proxmox VEクラスタにCephがローカルにインストールされている場合、ストレージの追加時に以下が自動的に実行されます。

---

デフォルトで有効になっているcephx認証を使用する場合は、外部のCephクラスタからキーリングを提供する必要があります。

CLIでストレージを構成するには、まず、キーリングを含むファイルを利用できるようにする必要があります。1つの方法は、外部CephクラスタからProxmox VEノードの1つにファイルを直接コピーすることです。次の例では、実行するノードの/rootディレクトリにコピーします：

```
# scp <外部cephserver>:/etc/ceph/ceph.client.admin.keyring /root/rbd.keyring
```

次に、pvesm CLIツールを使って外部RBDストレージを設定します。--keyringパラメータには、コピーしたキーリング・ファイルへのパスを指定します。例えば

```
# pvesm add rbd <名前> --monhost "10.1.1.20 10.1.1.21 10.1.1.22" --content images --keyring /root/rbd.keyring
```

GUIで外部RBDストレージを設定する場合、キーリングをコピーして適切なフィールドに貼り付けることができます。

キーホルダーは

```
# /etc/pve/priv/ceph/<STORAGE_ID>.keyring
```

---

### チップ

外部クラスタに接続する場合は、必要な機能のみを持つキーリングを作成することをお勧めします。Cephのユーザ管理の詳細については、Cephのドキュメントを参照してください。<sup>a</sup>

---

<sup>a</sup> Cephユーザー管理

### 7.15.3 Cephクライアント構成(オプション)

外部Cephストレージに接続しても、外部クラスタのconfig DBでクライアント固有のオプションを設定できるとは限りません。Cephキーリングの横にceph.confを追加して、ストレージのCephクライアント設定を変更できます。

ceph.confはストレージと同じ名前にする必要がある。

```
# /etc/pve/priv/ceph/<STORAGE_ID>.conf
```

RBD設定リファレンス<sup>1</sup>を参照してください。

---

### 注

これらの設定を軽率に変更しないでください。Proxmox VEは<STORAGE\_ID>.confをストレージ設定にマージしています。

---

## 7.15.4 ストレージ機能

rbdバックエンドはブロックレベルのストレージで、完全なスナップショットとクローン機能を実装している。

---

<sup>1</sup>RBD 設定リファレンス <https://docs.ceph.com/en/quincy/rbd/rbd-config-ref/>

表7.13: バックエンド rbd のストレージ機能

コンテンツの種類	画像フォーマット	共有	スナップ写真	クローン
イメージ ルートディレクトリ	生	はい	はい	はい

## 7.16 Cephファイルシステム (CephFS)

ストレージプールのタイプ: cephfs

CephFSはPOSIX準拠のファイルシステムを実装し、[Ceph](#)ストレージクラスタを使用してデータを格納します。CephFSはCephをベースに構築されているため、Cephの特性のほとんどを共有します。これには、冗長性、スケーラビリティ、自己回復、高可用性が含まれます。

---

### チップ

Proxmox VEは[Cephのセットアップを管理](#)できるため、CephFSストレージの構成が容易になります。最新のハードウェアは多くの処理能力とRAMを備えているため、ストレージサービスとVMを同じノードで実行してもパフォーマンスに大きな影響はありません。

---

CephFSストレージプラグインを使用するには、Debian純正のCephクライアントを置き換えて、[Cephリポジトリ](#)を追加する必要があります。追加したら、`apt update`を実行し、続いて`apt dist-upgrade`を実行して、最新のパッケージを取得します。

---



### 警告

他のCephリポジトリが設定されていないことを確認してください。そうしないと、インストールに失敗するか、ノード上にパッケージのバージョンが混在して予期しない動作になります。

---

### 7.16.1 構成

このバックエンドは、一般的なストレージプロパティであるノード、`disable`、コンテンツと、以下の

cephfs固有のプロパティをサポートする：

### エフェスネーム

Ceph FSの名前。

### モノホスト

モニターモンタードレスのリスト。オプション。CephがProxmox VEクラスタで実行されていない場合にのみ必要です。

### パス

ローカルのマウントポイント。オプションで、デフォルトは /mnt/pve/<STORAGE\_ID>/です。

## ユーザー名

CephユーザID。オプション。CephがProxmox VEクラスタで実行されていない場合にのみ必要で、デフォルトはadminです。

## サブディレクトリ

マウントするCephFSサブディレクトリ。オプション、デフォルトは/です。

## ヒューズ

カーネルクライアントの代わりにFUSEを介してCephFSにアクセスします。オプション、デフォルトは0。

## 外部Cephクラスタの構成例(/etc/pve/storage.cfg)

```
cephfs: cephfs-external
    monhost 10.1.1.20 10.1.1.21 10.1.1.22
    パス /mnt/pve/cephfs-external
    コンテンツのバックアップ
    ユーザー名
    _____
    admin fs-名前
    cephfs
```

## 注

cephxが無効になっていない場合は、クライアントの秘密鍵ファイルを設定することを忘れないでください。

## 7.16.2 認証

### 注

Proxmox VEクラスタにCephがローカルにインストールされている場合、ストレージの追加時に以下が自動的に実行されます。

デフォルトで有効になっているcephx認証を使用する場合、外部のCephクラスタからシークレットを提供する必要があります。

CLIでストレージを構成するには、まず、シークレットを含むファイルを利用可能にする必要があります。1つの方法は、外部CephクラスタからProxmox VEノードの1つにファイルを直接コピーすることです。次の例では、実行するノードの/rootディレクトリにコピーします：

```
# scp <外部 cephserver>:/etc/ceph/cephfs.secret /root/cephfs.secret
```

次に、pvesm CLIツールを使用して外部RBDストレージを設定します。--keyringパラメータを使用し、

```
# pvesm add cephfs <name> --monhost "10.1.1.20 10.1.1.21 10.1.1.22" --<'  
    コンテンツバックアップ --キーリング /ルート/cephfs.secret
```

コピーしたシークレットファイルへのパスを指定します。例えば

GUIで外部RBDストレージを設定する場合は、適切なフィールドにシークレットをコピー&ペーストしてください。

rbdバックエンドには[client.userid]も含まれているのとは対照的に、シークレットはキーそのものだけである。

セクションを参照されたい。

秘密は

```
# /etc/pve/priv/ceph/<STORAGE_ID>.secret
```

useridはクラスタにアクセスするように設定されているクライアントIDです。Cephユーザ管理の詳細については、Cephドキュメントを参照してください。<sup>a</sup>

```
# ceph auth get-key client.userid > cephfs.secret
```

### 7.16.3 ストレージ機能

cephfsバックエンドは、Cephクラスタ上のPOSIX準拠ファイルシステムです。

表7.14: バックエンドcephfsのストレージ機能

コンテンツの種類	画像フォーマット	共有	スナップ写真	クローン
vztmp1 iso バックア ップスニ ペット	なし	はい	yes <sup>[1]</sup>	いいえ

<sup>[1]</sup> 既知のバグは存在しないが、スナップショットは十分なテストが行われていないため、安定性は保証されていない。

## 7.17 BTRFSバックエンド

ストレージプールのタイプ: btrfs

表面的には、このストレージ・タイプはディレクトリ・ストレージ・タイプと非常によく似ているので、一般的な概要についてはディレクトリ・バックエンドのセクションを参照のこと。

主な違いは、このストレージタイプでは、スナップショットを取り、スナップショットを保持したままオンラインストレージの移行をサポートするために、rawフォーマットされたディスクがサブボリュームに配置されることである。

### 注

BTRFSは、ファイルをオープンする際にO\_DIRECTフラグを尊重する。

そうでなければチェックサム・エラーが発生する。

## 7.17.1 構成

このバックエンドは、ディレクトリ・ストレージと同様に設定します。それ自身がマウントポイントでもないディレクトリを BTRFS ストレージとして追加する場合は、`is_mountpoint` オプションで実際のマウントポイントを指定することをお勧めします。

例えば、BTRFS ファイルシステムが /mnt/data2 にマウントされ、その pve-storage/ サブディレクトリ（スナップショットである可能性があり、これを推奨する）を data2 というストレージプールとして追加する場合、以下のエントリを使用することができます：

```
btrfs: データ2
  パス /mnt/data2/pve-storage ノ
  ンテンツ rootdir,images
  is_mountpoint /mnt/data2
```

### 7.17.2 スナップ写真

サブボリュームまたはRAWファイルのスナップショットを作成する場合、スナップショットは、同じパスに@とスナップショット名を付けた読み取り専用のサブボリュームとして作成されます。

## 7.18 ZFS over iSCSIバックエンド

ストレージ・プールのタイプ: zfs

このバックエンドは、ストレージとしてZFSプールを持ち、iSCSIターゲットを実装したリモートマシンにssh経由でアクセスします。各ゲストディスクに対してZVOLを作成し、iSCSI LUNとしてエクスポートします。このLUNはProxmox VEによってゲストディスクに使用されます。

以下のiSCSIターゲット実装がサポートされている:

- LIO (リナックス)
- IET (リナックス)
- ISTGT (FreeBSD)
- コムスター (ソラリス)

#### 注

このプラグインは、ZFS対応のリモートストレージアプライアンスが必要であり、通常のストレージアプライアンス/SAN上にZFSプールを作成するために使用することはできません。

### 7.18.1 構成

ZFS over iSCSI プラグインを使用するには、リモートマシン(ターゲット)が Proxmox VE ノードからのssh接続を受け付けるように設定する必要があります。Proxmox VEはターゲットに接続してZVOLを作成

し、iSCSI経由でエクスポートします。に保存されたsshキー（パスワード保護なし）を使用して認証します。

/etc/pve/priv/zfs/<target\_ip>\_id\_rsa

以下の手順でssh-keyを作成し、IP 192.0.2.1のストレージ・マシンに配布する：

```
mkdir /etc/pve/priv/zfs  
ssh-keygen -f /etc/pve/priv/zfs/192.0.2.1_id_rsa  
ssh-copy-id -i /etc/pve/priv/zfs/192.0.2.1_id_rsa.pub  
root@192.0.2.1 ssh -i /etc/pve/priv/zfs/192.0.2.1_id_rsa  
root@192.0.2.1
```

バックエンドは、一般的なストレージ・プロパティであるcontent、nodes、disable、および以下のZFS over ISCSI固有のプロパティをサポートする：

## プール

iSCSIターゲット上のZFSプール/ファイルシステム。すべての割り当てはそのプール内で行われる。

## ポータル

iSCSIポータル（IPまたはDNS名とオプションのポート）。

## ターゲット

iSCSI ターゲット。

## アイソサイプロバイダー

リモート・マシンで使用されているiSCSIターゲットの実装。

## コムスター

コムスター・ビューのターゲット・グループ。

## comstar\_hg

comstarビューのホストグループ。

## lio\_tpg

Linux LIOターゲット用ターゲット・ポータル・グループ

## ナウライトキャッシュ

ターゲットの書き込みキャッシングを無効にする

## ロックサイズ

ZFSロックサイズ・パラメータを設定する。

## まばら

ZFSシン・プロビジョニングを使用する。スペースボリュームとは、予約がボリュームサイズに等しくないボリュームのことです。

## 設定例 (/etc/pve/storage.cfg)

```
zfs: リオ
  ブロックサイズ 4k
  iscsiprovider
  LIOプールタンク
  ポータル 192.0.2.111
  ターゲット iqn.2003-01.org.linux-iscsi.lio.x8664:snxxxxxxxxx コン
  テンツイメージ
  lio_tpg tpg1 ス
  パース 1

zfs: ソラリスのブロック
  クサイズ 4k
  target iqn.2010-08.org.illumos:02:xxxxxxxx-xxxxxx-xxxxxxxxxxxx: ←'
    タンク1
  プールタンク
```

```
iscsiprovider
comstar portal
192.0.2.112 コンテンツ
イメージ

zfs: freebsd ブロック
クオタサイズ 4k
ターゲット iqn.2007-09.jp.ne.peach.istgt:tank1
プールタンク
iscsiprovider
istgt portal
192.0.2.113 コンテンツ
イメージ

zfs: iet
ブロッククオタサイズ 4k
ターゲット iqn.2001-04.com.example:tank1
プールタンク
iscsiprovider iet
ポータル
192.0.2.114 コンテンツ
イメージ
```

## 7.18.2 ストレージ機能

ZFS over iSCSIプラグインは、スナップショットが可能な共有ストレージを提供します。ZFS アプライアンスがデプロイメントの単一障害点にならないようにする必要があります。

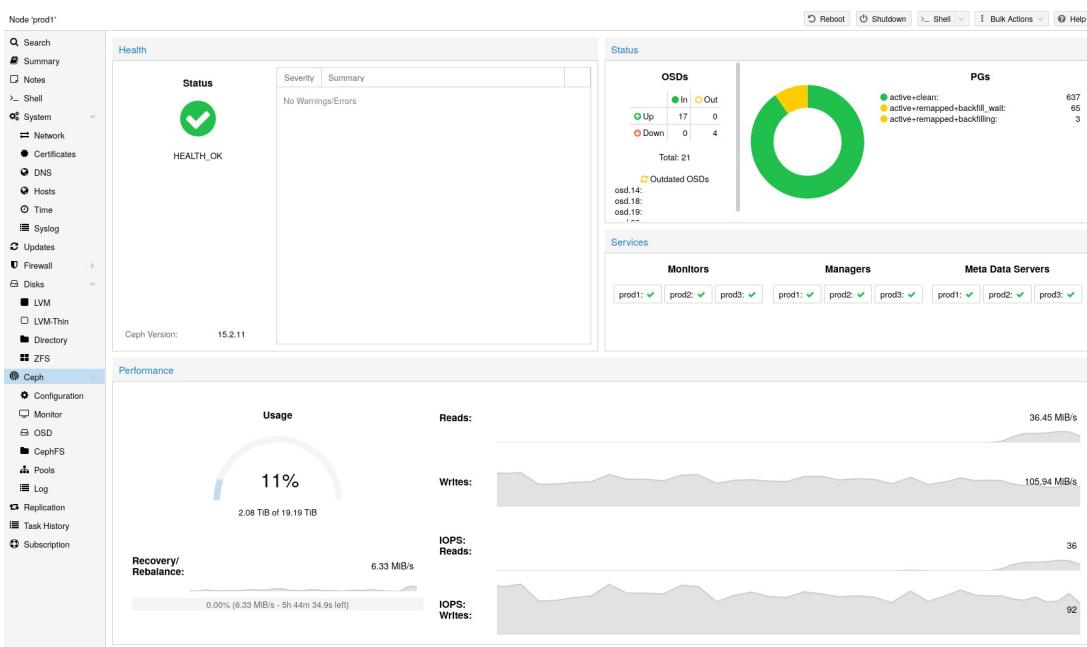
表7.15: バックエンド`iscsi`のストレージ機能

コンテンツの種類	画像フォーマット	共有	スナップ写真	クローン
イメージ	生	はい	はい	いいえ

## 第8章

# ハイパー・コンバージド Ceph クラスタの展開

## 8.1 はじめに



Proxmox VEは、コンピュートシステムとストレージシステムを統合します。つまり、クラスタ内の同じ物理ノードをコンピューティング（VMとコンテナの処理）とレプリケートされたストレージの両方に使用できます。コンピュート・リソースとストレージ・リソースの従来のサイロを、単一のハイパー・コンバージド・アプライアンスにまとめることができます。別々のストレージエイジネットワーク（SAN）やネットワークアタッチドストレージ（NAS）を介した接続は消滅する。オープンソースのSoftware-Defined StorageプラットフォームであるCephの統合により、Proxmox VEはハイパーバイザー・ノード上でCephストレージを直接実行・管理できる。

Cephは、優れたパフォーマンス、信頼性、スケーラビリティを提供するように設計された分散オブジェクトストアおよびファイルシステムである。

PROXMOX VE上のCEPHの利点は以下のとおりです：

- CLIとGUIによる簡単なセットアップと管理
- シン・プロビジョニング
- スナップショットのサポート

- セルフ・ヒーリング
- エクサバイト・レベルまで拡張可能
- ブロック、ファイルシステム、オブジェクトストレージを提供
- 異なる性能と冗長特性を持つプールを設定する
- データは複製されるため、フォールトトレラントである。
- コモディティ・ハードウェアで動作
- ハードウェアRAIDコントローラ不要
- オープンソース

小規模から中規模の導入では、RADOS Block Devices (RBD) またはCephFSを使用するCephサーバをProxmox VEクラスタノードに直接インストールできます([Ceph RADOS Block Devices \(RBD\)を参照](#))。最近のハードウェアはCPUパワーとRAMが大きいため、ストレージサービスと仮想ゲストを同じノードで実行できます。

管理を簡素化するため、Proxmox VEは、組み込みのWebインターフェース、または*pveceph*コマンドラインツールを使用して、Proxmox VEノードにCephサービスをインストールおよび管理するためのネイティブ統合を提供します。

## 8.2 用語解説

CEPHは複数のデーモンで構成され、RBDストレージとして使用される：

- Cephモニタ (ceph-mon、またはMON)
- Ceph Manager (ceph-mgr、またはMGS)
- Cephメタデータサービス (ceph-mds、またはMDS)
- Ceph Object Storage Daemon (ceph-osd、またはOSD)

---

### チップ

Cephに精通することを強くお勧めします。[a](#)とそのアーキテクチャ [b](#) および語彙 [c](#).

---

<sup>a</sup>Ceph intro <https://docs.ceph.com/en/quincy/start/intro/>

<sup>b</sup>Cephアーキテクチャ <https://docs.ceph.com/en/quincy/architecture/>

<sup>c</sup>Ceph用語集 <https://docs.ceph.com/en/quincy/glossary>

## 8.3 健全なCephクラスタの推奨事項

ハイパーコンバージドProxmox + Ceph Clusterを構築するには、セットアップに少なくとも3台の(できれば)同一のサーバーを使用する必要があります。

Cephのウェブサイトからの推奨事項も確認してほしい。

## 注

以下の推奨事項は、ハードウェアを選択する際のおおまかな指針として見ていただきたい。従って、あなたの特定のニーズに合わせることが不可欠であることに変わりはない。セットアップをテストし、健全性とパフォーマンスを継続的に監視する必要があります。

## CPU

Cephサービスは2つのカテゴリに分類できる： \* CPUを集中的に使用し、高いCPU基本周波数とマルチコアの恩恵を受ける。このカテゴリのメンバーは次のとおりです：**オブジェクトストレージデーモン (OSD) サービス** CephFSに使用されるメタデータサービス (MDS) \* 中程度のCPU使用率、複数のCPUコアを必要としない。これらは次のとおりです：**モニタ (MON) サービス マネージャ (MGR) サービス**

単純な経験則として、安定した耐久性のあるCephパフォーマンスに必要な最小限のリソースを提供するために、各Cephサービスに少なくとも1つのCPUコア（またはスレッド）を割り当てる必要があります。

たとえば、1つのノードでCephモニタ、Cephマネージャ、および6つのCeph OSDサービスを実行する予定であれば、基本的に安定したパフォーマンスを目標とする場合、8つのCPUコアをCeph専用に確保する必要があります。

OSDのCPU使用率は、ほとんどディスクの性能に依存することに注意してください。ディスクのIOPS (IO Operations per Second) が高ければ高いほど、OSDサービスで使用できるCPUは多くなる。ミリ秒以下のレイテンシで100,000を超える高いIOPS負荷を永続的に維持できるNVMeのような最新のエンタープライズSSDディスクの場合、各OSDは複数のCPUスレッドを使用することができます。

## メモリー

特にハイパーコンバージドセットアップでは、メモリ消費を注意深く計画し、監視する必要がある。仮想マシンとコンテナのメモリ使用量の予測に加えて、Cephが優れた安定したパフォーマンスを提供するために十分なメモリを利用できることも考慮する必要があります。

経験則として、およそ1TiBのデータに対して、1GiBのメモリがOSDによって使用される。通常の状態では使用量は少ないかもしれないが、リカバリー、再バランス、バックフィルなどのクリティカルなオペレーション時には最も多く使用することになる。つまり、通常の運用で使用可能なメモリーを最大にすることは避け、むしろ機能停止に対処できるよう、若干の余裕を残しておく必要がある。

OSDサービス自体はさらにメモリを使用します。デーモンのCeph BlueStoreバックエンドには、デフォルト

で3~5GiBのメモリが必要です（調整可能）。

## ネットワーク

少なくとも10 Gbps以上のネットワーク帯域幅をCephトラフィック専用に使用することを推奨します。メッシュ型ネットワーク設定<sup>1</sup>は、10Gbps以上のスイッチがない場合、3~5ノードクラスタのオプションにもなります。

### 重要



特にリカバリ時のトラフィック量は、同じネットワーク上の他のサービスに干渉し、特にレイテンシに敏感なProxmox VEのcorosyncクラスタスタックが影響を受け、クラスタクオーラムが失われる可能性があります。Cephトラフィックを専用の物理的に分離されたネットワークに移動することで、corosyncだけでなく、仮想ゲストによって提供されるネットワークサービスでも、このような干渉が回避されます。

---

<sup>1</sup>Ceph用フルメッシュネットワーク [https://pve.proxmox.com/wiki/Full\\_Mesh\\_Network\\_for\\_Ceph\\_Server](https://pve.proxmox.com/wiki/Full_Mesh_Network_for_Ceph_Server)

必要な帯域幅を見積るには、ディスクの性能を考慮する必要があります。1台のHDDでは1Gbのリンクを飽和させることはできないかもしれません、ノードあたり複数のHDD OSDがあれば、すでに10Gbpsも飽和させることができます。最新のNVMe接続SSDを使用すれば、1台で10Gbps以上の帯域幅を飽和させることができます。このような高性能セットアップの場合、少なくとも25Gpbsを推奨しますが、基盤となるディスクの潜在性能をフルに活用するには、40Gbpsまたは100Gbps以上が必要な場合もあります。

よくわからない場合は、高性能なセットアップには3つの(物理的な)個別のネットワークを使用することをお勧めします： \* Ceph(内部)クラスタトラフィック用の超高帯域幅(25Gbps以上)ネットワーク1つ。 \* CephサーバーとCephクライアントストレージトラフィック間のCeph (パブリック)トラフィック用の高帯域幅 (10+ Gpbs)ネットワーク1つ。ニーズに応じて、仮想ゲストトラフィックとVMライブマイグレーショントラフィックのホストにも使用できます。 \* レイテンシに敏感なcorosyncクラスタ通信専用の中帯域幅(1 Gbps)1つ。

## ディスク

Cephクラスタのサイズを計画する際は、リカバリ時間を考慮することが重要です。特に小さなクラスタでは、リカバリに時間がかかる可能性があります。リカバリ時間を短縮し、リカバリ中に後続の障害イベントが発生する可能性を最小限に抑えるため、小規模なセットアップではHDDの代わりにSSDを使用することをお勧めします。

一般に、SSDは回転ディスクよりも多くのIOPSを提供する。この点を考慮すると、コストが高くなることに加え、[クラスベースのプール分離](#)を実装する意味がある場合があります。OSDを高速化するもう1つの方法は、より高速なディスクをジャーナルまたはDB/Write-Ahead-Logデバイスとして使用することです。より高速なディスクを複数のOSDに使用する場合、OSDとWAL/DB(またはジャーナル)ディスクの適切なバランスを選択する必要があります。

ディスクの種類は別として、Cephはノードごとに均等なサイズで均等な量のディスクを使用するのが最適です。たとえば、1 TBディスク1台と250 GBディスク3台が混在するセットアップよりも、各ノード内に500 GBディスク4台がある方が優れています。

OSD数とシングルOSD容量のバランスも取る必要があります。容量を増やすとストレージ密度を高めることができますが、OSDが1つ故障するとCephは一度に多くのデータを復旧しなければならなくなります。

## RAIDを避ける

Cephはデータオブジェクトの冗長性とディスクへの複数の並列書き込み(OSD)を独自に処理するため、通常

、RAIDコントローラを使用してもパフォーマンスや可用性は向上しません。それどころか、Cephは、間に抽象化を挟むことなく、ディスク全体を単独で処理するように設計されています。RAIDコントローラはCephのワークロード用に設計されておらず、書き込みやキャッシュのアルゴリズムがCephのものと干渉する可能性があるため、状況が複雑になり、場合によってはパフォーマンスが低下することさえあります。



### 警告

RAIDコントローラは避けてください。代わりにホストバスアダプタ（HBA）を使用する。

## 8.4 Cephの初期インストールと構成

### 8.4.1 ウェブベースのウィザードの使用

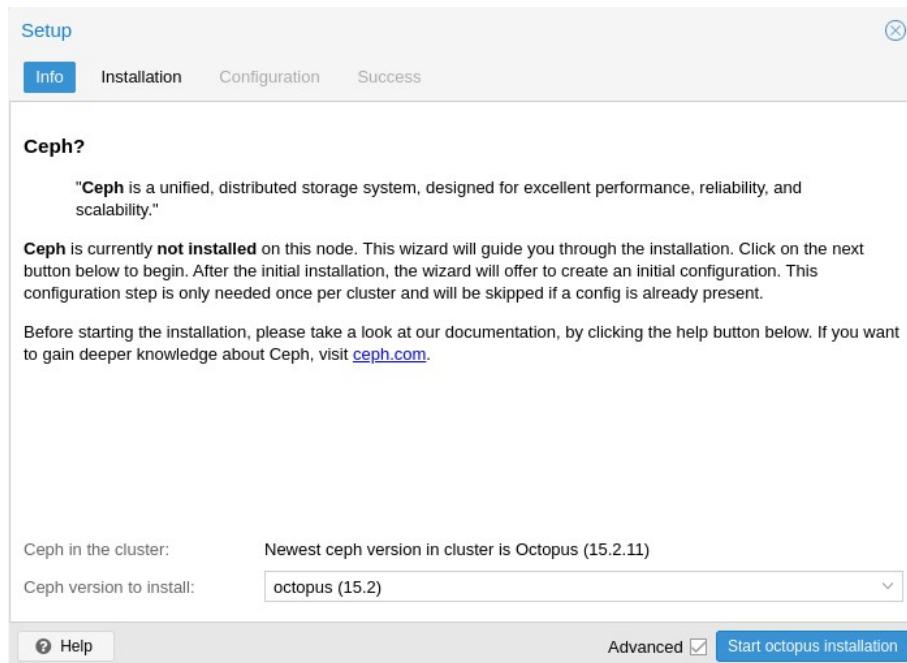
The screenshot shows the Proxmox VE web interface. The left sidebar has a 'Ceph' section selected, which includes options like Configuration, Monitor, OSD, CephFS, Pools, Log, Replication, Task History, and Subscription. The main content area has tabs for 'Health' and 'Status'. In the 'Status' tab, there's a 'Ceph Version:' section with a question mark icon. Below it is an 'OSDs' table with columns for In, Out, Up, and Down, all showing 0. To the right of the table is a message: 'Ceph is not installed on this node. Would you like to install it now?' with a blue 'Install Ceph' button. At the bottom of the main content area are sections for Monitors, Managers, and Meta Data Servers.

Proxmox VEには、Ceph用の使いやすいインストールウィザードが用意されています。クラスタノードの1つをクリックし、メニューツリーのCephセクションに移動します。Cephがまだインストールされていない場合は、インストールを促すプロンプトが表示されます。

ウィザードは複数のセクションに分かれており、Cephを使用するには各セクションを正常に終了する必要がある。

まず、インストールするCephのバージョンを選択する必要があります。他のノードのものを選ぶか、これがCephをインストールする最初のノードであれば最新のものを選びます。

インストールを開始すると、ウィザードがProxmox VEのCephリポジトリから必要なパッケージをすべてダウンロードしてインストールします。



インストール手順を終了したら、構成を作成する必要があります。この構成はProxmox VEのクラスタ構成ファイルシステム(pmxcfs)を通じて残りのすべてのクラスタメンバに自動的に配布されるため、このステップはクラスタごとに1回だけ必要です。

コンフィギュレーション・ステップには以下の設定が含まれる：

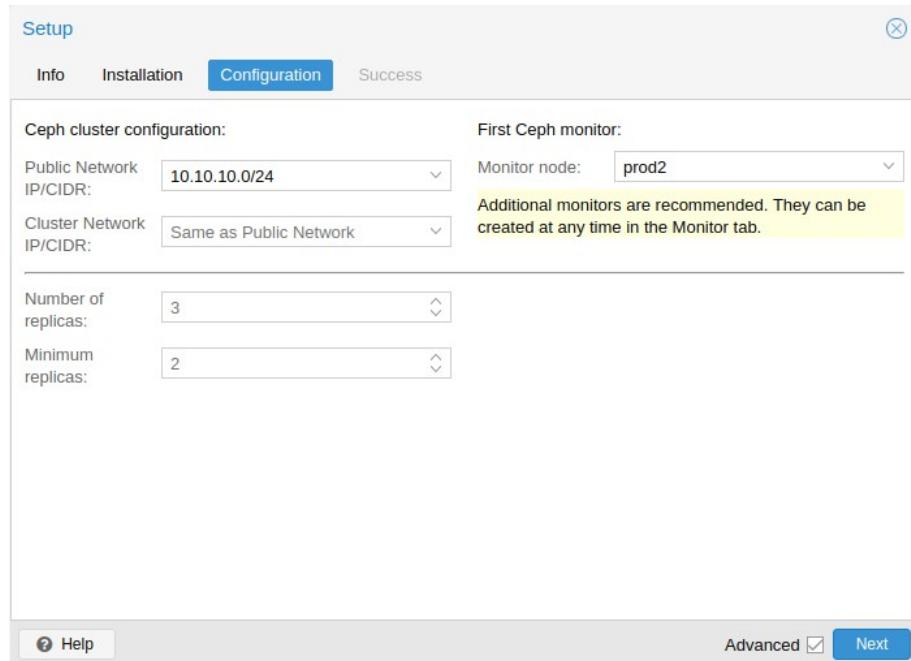
- **パブリックネットワーク：** このネットワークは、パブリックストレージ通信(Ceph RBDバックアップディスクやCephFSマウントを使用する仮想マシンなど)、およびさまざまなCephサービス間の通信に使用されます。この設定は必須です。

Ceph トラフィックをProxmox VEクラスタ通信(corosync)、および可能であれば仮想ゲストの前面(パブリック)ネットワークから分離することを強く推奨します。そうしないと、Cephの高帯域幅IOトラフィックが他の低レイテンシ依存サービスに干渉する可能性があります。

- **クラスタネットワーク：** OSDのレプリケーションとハートビートのトラフィックも分離するように指定します。この設定はオプションです。

物理的に分離されたネットワークを使用すると、Cephパブリックと仮想ゲストのネットワークが緩和され、Cephのパフォーマンスも大幅に向上するため、推奨される。

Cephクラスタネットワークを構成し、後で物理的に分離された別のネットワークに移動できます。



さらに2つのオプションがあるが、これは上級者向けなので、自分が何をしているのかわかっている場合のみ変更してほしい。

- **レプリカの数:** オブジェクトが複製される頻度を定義します。
- **最小レプリカ数:** I/Oを完了としてマークするために必要なレプリカの最小数を定義します。

さらに、最初のモニターノードを選択する必要があります。このステップは必須です。

これで完了です。最後のステップとして成功ページが表示され、さらに進む方法が表示されます。これで、システムはCephの使用を開始する準備ができました。開始するには、追加の[モニタ](#)、[OSD](#)、および少なくとも1つの[プール](#)を作成する必要があります。

この章の残りの部分では、Proxmox VEベースのCephセットアップを最大限に活用する方法を説明します。これには、前述のヒントのほか、新しいCephクラスタに追加すると便利な[CephFS](#)などが含まれます。

#### 8.4.2 CephパッケージのCLIインストール

Webインターフェースで利用可能な推奨のProxmox VE Cephインストールウィザードの代わりに、各ノードで

pvecephインストール

以下のCLIコマンドを使用できます：

これにより、/etc/apt/sources.list.d/ceph.listにaptパッケージリポジトリが設定され、必

必要なソフトウェアがインストールされます。

### 8.4.3 CLIによるCephの初期構成

Proxmox VE Cephインストールウィザードを使用するか(推奨)、1つのノードで以下のコマンドを実行します:

```
pveceph init --network 10.10.10.0/24
```

これにより、`/etc/pve/ceph.conf`に、Ceph専用のネットワークを持つ初期構成が作成されます。このファイルは、`pmxcfs`を使用して、すべてのProxmox VEノードに自動的に配布されます。このコマンドはまた、`/etc/ceph/ceph.conf`にシンボリックリンクを作成し、このファイルを指すようにします。したがって、設定ファイルを指定しなくても、Cephコマンドを単純に実行できます。

## 8.5 Cephモニター

Name	Host	Status	Address	Version	Quorum
mon.prod1	prod1	running	192.168.30.64:6789/0	15.2.11	Yes
mon.prod2	prod2	running	192.168.30.65:6789/0	15.2.11	Yes
mon.prod3	prod3	running	192.168.30.66:6789/0	15.2.11	Yes

Name	Host	Status	Address	Version
mgr.prod1	prod1	active	192.168.30.64	15.2.11
mgr.prod2	prod2	standby	192.168.30.65	15.2.11
mgr.prod3	prod3	standby	192.168.30.66	15.2.11

Cephモニタ(MON)<sup>2</sup>はクラスタマップのマスターコピーを維持します。高可用性を実現するには、少なくとも3つのモニタが必要です。インストールウィザードを使用した場合、1つのモニタがすでにインストールされています。クラスタが小規模から中規模であれば、3つ以上のモニタは必要ありません。これ以上必要なのは、本当に大規模なクラスタだけです。

### 8.5.1 モニターの作成

モニタを配置する各ノードで(3つのモニタを推奨)、GUIのCeph → Monitorタブを使用するか、runしてモニ

タを作成します：

<sup>2</sup>Ceph モニター <https://docs.ceph.com/en/quincy/start/intro/>

### 8.5.2 モニターを破壊する

GUIでCephモニタを削除するには、まずツリービューでノードを選択し、**Ceph → Monitor** パネルに表示されます。MONを選択し、**Destroy**ボタンをクリックします。

CLIでCephモニタを削除するには、まずMONが実行されているノードに接続します。次に、次のコマンドを

「`prvceph-mon`」

実行します：

---

#### 注

定足数には少なくとも3名のモニターが必要である。

---

## 8.6 Cephマネージャー

Managerデーモンはモニターと並行して実行される。クラスタを監視するインターフェースを提供します。

Ceph luminousのリリース以降、少なくとも1つの`ceph-mgr`<sup>3</sup> デーモンが必要です。

### 8.6.1 クリエイトマネージャー

複数のManagerをインストールすることができますが、常にアクティブなManagerは1つだけです。

「`privceph`」は作成する

---

#### 注

Ceph Managerはモニタノードにインストールすることを推奨します。高可用性を実現するには、複数のマネージャをインストールします。

---

### 8.6.2 デストロイ・マネージャー

GUIでCeph Managerを削除するには、まずツリービューでノードを選択し、**Ceph → Monitor** パネルに表示されます。Managerを選択し、**Destroy**ボタンをクリックする。

CLIでCephモニタを削除するには、まずManagerが実行されているノードに接続します。次に、次のコマンドを実

「`vecephmgr`」は破壊する

行します：

---

### 注

マネージャはハード依存ではないが、PG自動スケーリング、デバイスのヘルスモニタリング、テレメトリなどの重要な機能を処理するため、Cephクラスタにとって重要である。

---

<sup>3</sup>Ceph Manager <https://docs.ceph.com/en/quincy/mgr/>

## 8.7 Ceph OSD

Name	Class	OSD Type	Status	Version	weight	reweight	Used (%)	Total	Apply/Commit Latency (ms)
<b>No OSD selected</b>									
<b>default</b>									
prod3				15.2.11					
osd.15	ssd	bluestore	up  / in	15.2.11	0.9097	1.00	8.35	931.51 GiB	5 / 5
osd.12	ssd	bluestore	up  / in	15.2.11	0.45479	1.00	13.79	465.76 GiB	2 / 2
osd.11	ssd	bluestore	up  / in	15.2.11	0.46579	1.00	6.58	476.94 GiB	12 / 12
osd.8	hdd	bluestore	up  / in	15.2.11	3.63869	1.00	10.18	3.64 TiB	39 / 39
osd.5	hdd	bluestore	up  / in	15.2.11	0.45409	1.00	15.18	465.00 GiB	13 / 13
osd.3	hdd	bluestore	up  / in	15.2.11	0.45409	1.00	18.24	465.00 GiB	31 / 31
prod2				15.2.11					
osd.16	ssd	bluestore	up  / in	15.2.11	0.9097	1.00	10.82	931.51 GiB	5 / 5
osd.13	ssd	bluestore	up  / in	15.2.11	0.45479	1.00	13.53	465.76 GiB	1 / 1
osd.10	ssd	bluestore	up  / in	15.2.11	0.46579	1.00	10.92	476.94 GiB	20 / 20
osd.7	hdd	bluestore	up  / in	15.2.11	3.63869	1.00	10.80	3.64 TiB	15 / 15
osd.4	hdd	bluestore	up  / in	15.2.11	0.58199	1.00	8.67	596.00 GiB	11 / 11
osd.2	hdd	bluestore	up  / in	15.2.11	0.58199	1.00	6.64	596.00 GiB	12 / 12
prod1				15.2.11					
osd.17	ssd	bluestore	up  / in	15.2.11	0.9097	1.00	13.15	931.51 GiB	4 / 4
osd.9	ssd	bluestore	up  / in	15.2.11	0.46579	1.00	7.95	476.94 GiB	12 / 12
osd.6	hdd	bluestore	up  / in	15.2.11	3.63869	1.00	11.45	3.64 TiB	14 / 14
osd.1	hdd	bluestore	up  / in	15.2.11	0.58199	1.00	7.78	596.00 GiB	12 / 12
osd.0	hdd	bluestore	up  / in	15.2.11	0.58199	1.00	12.11	595.98 GiB	33 / 33

Ceph Object Storage Daemonは、ネットワーク経由でCephのオブジェクトを格納します。物理ディスクごとに1つのOSDを使用することを推奨します。

### 8.7.1 OSDの作成

OSD は Proxmox VE ウェブインターフェースまたは pveceph を使用した CLI で作成できます。例えば

```
pveceph osd create /dev/sd[X]
```

#### チップ

少なくとも3台のノードと、ノード間で均等に分散された少なくとも12台のOSDを備えたCephクラスタを推奨します。

ディスクが以前（たとえばZFSやOSDとして）使用されていた場合は、まずその使用痕跡をすべて消去する

```
ceph-volume lvm zap /dev/sd[X] --destroy
```

必要があります。パーティションテーブル、ブートセクタ、その他のOSDの残りを削除するには、以下のコマンドを使用します：



### 警告

上記のコマンドは、ディスク上のすべてのデータを破壊する！

## セフ・ブルーストア

Ceph Krakenリリースから、Bluestoreという新しいCeph OSDストレージタイプが導入されました。<sup>4</sup>これ

```
pveceph osd create /dev/sd[X]
```

は、Ceph Luminous以降のOSD作成時のデフォルトです。

### block.dbとblock.wal

OSDに別のDB/WALデバイスを使いたい場合は、`-db_dev`と

`-wal_dev`オプション。別途指定しない場合、WALはDBと一緒に置かれる。

```
pveceph osd create /dev/sd[X] -db_dev /dev/sd[Y] -wal_dev /dev/sd[Z].
```

それぞれ`-db_size`と`-wal_size`パラメーターで直接サイズを指定できる。それらが与えられない場合は、以下の値（順番に）が使われる：

- Ceph設定からのbluestore\_block\_{db,wal}\_size。 . .
  - データベース、セクションOSD
  - データベース、グローバルセクション
  - ファイル、セクションosd
  - ファイル、セクショングローバル
- OSDサイズの10%(DB)/1%(WAL)

### 注

DBはBlueStoreの内部メタデータを格納し、WALはBlueStoreの内部ジャーナルまたはライトアヘッド・ログです。パフォーマンスを向上させるために、高速なSSDまたはNVRAMを使用することをお勧めします。

## Cephファイルストア

Ceph Luminous以前は、Ceph OSDのデフォルトのストレージタイプとしてFilestoreが使用されていました

```
ceph-volume lvm create --filestore --data /dev/sd[X] --journal /dev/sd[Y].
```

。 Ceph Nautilus以降、Proxmox VEでは、*pveceph*を使用したこのようなOSDの作成がサポートされなくなりました。 ファイルストアOSDを作成する場合は、*ceph-volume*を直接使用してください。

<sup>4</sup>Ceph Bluestore <https://ceph.com/community/new-luminous-bluestore/>

## 8.7.2 OSDを破棄する

GUIでOSDを削除するには、まずツリービューでProxmox VEノードを選択し、**Ceph → OSD** パネルに表示されます。次に、破壊するOSDを選択し、**OUT**ボタンをクリックします。OSDのステータスがインからアウトに変わったら、**STOP**ボタンをクリックする。最後に、ステータスがアップからダウンに変わった後、**破棄**を選択します。

をドロップダウンメニューから選択する。

CLIでOSDを削除するには、以下のコマンドを実行する。

```
ceph osd out <ID>
systemctl stop ceph-osd@<ID>.service
```

---

### 注

最初のコマンドは、データ配布にOSDを含めないようにCephに指示します。2番目のコマンドは、OSDサービスを停止します。この時点まで、データは失われません。

---

次のコマンドはOSDを破壊する。さらにパーティションテーブルを破棄するには、*-cleanup*オプションを指定

```
pveceph osd destroy <ID>
```

します。



### 警告

上記のコマンドは、ディスク上のすべてのデータを破壊する！

---

## 8.8 Ceph プール

Name	Size/min	# of Placement Gr...	Optimal # of PGs	Autoscale Mode	CRUSH Rule (ID)	Used (%)
cp	3/2	256	256	on	replicated_rule (0)	937.64 GiB (5.85%)
cephfs_data	3/2	32	32	on	replicated_rule (0)	846.00 GiB (5.31%)
cephfs_metadata	3/2	32	16	warn	replicated_rule (0)	101.09 MiB (0.00%)
cp-krbd	3/2	256	256	on	replicated_rule (0)	320.31 GiB (2.08%)
device_health_metrics	3/2	1	1	on	replicated_rule (0)	147.90 MiB (0.00%)
ssd	3/2	128	128	on	replicated_ssd (1)	0 B (0.00%)

2.05 TiB

プールは、オブジェクトを格納するための論理グループである。これは、配置グループ（PG、pg\_num）。

### 8.8.1 プールの作成と編集

プールの作成と編集は、Proxmox VE ホストのコマンドラインまたは Web インタフェースから、以下の手順で実行できます。

#### Ceph → Pools.

オプションが指定されていない場合、OSDに障害が発生してもデータ損失が発生しないように、デフォルトのPG数を128、レプリカのサイズを3、min\_sizeを2に設定する。



#### 警告

**min\_sizeを1に設定しないでください。** min\_sizeを1に設定したレプリケート・プールでは、レプリカが1つしかないときにオブジェクトに対するI/Oが許可されるため、データ損失、不完全なPG、未発見のオブジェクトが発生する可能性があります。

PG-Autoscalerを有効にするか、セットアップに基づいてPG数を計算することをお勧めします。計算式とPG計算機は<sup>5</sup> を参照してください。Ceph Nautilus以降では、設定後にPG数を変更できます。<sup>6</sup> を変更できます。

<sup>5</sup> PG電卓 <https://web.archive.org/web/20210301111112/http://ceph.com/pgcalc/>

<sup>6</sup> プレイスマント・グループ <https://docs.ceph.com/en/quincy/rados/operations/placement-groups/>

PGオートスケーラー<sup>7</sup>は、バックグラウンドでプールの PG カウントを自動的にスケーリングできます。ターゲット・サイズまたはターゲット比率のアドバンス・パラメーターを設定することで、PGオートスケーラーがより適切な判断を下せるようになります。

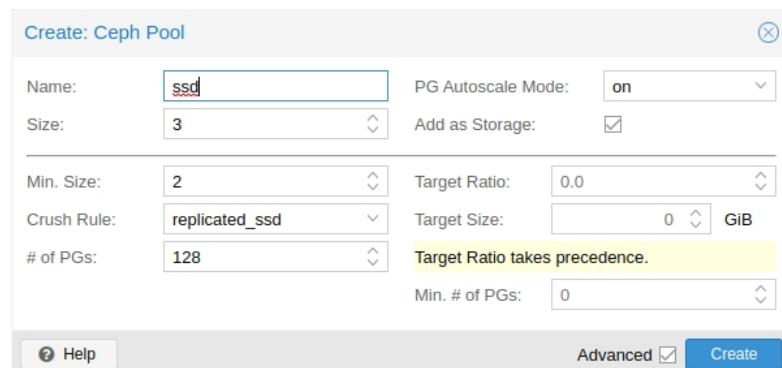
### CLIでプールを作成する例

```
pveceph pool create <プール名> --add_storages
```

### チップ

プールにストレージを自動的に定義したい場合は、Webインターフェースで'Add as Storage'チェックボックスをチェックしたままにするか、プール作成時にコマンドラインオプション--add\_storagesを使用する。

### プール・オプション



以下のオプションは、プール作成時に利用可能であり、プールの編集時にも部分的に利用可能である。

#### 名称

プールの名前。これは一意でなければならず、後から変更することはできない。

#### サイズ

オブジェクトあたりのレプリカ数。Cephは常にオブジェクトのコピーをこの数持つようにします。デフォルト:  
3.

#### PGオートスケールモード

プールの自動PGスケーリングモード<sup>7</sup> プールの自動 PG スケーリングモード。warn に設定すると、プールの PG 数が最適でない場合に警告メッセージを生成する。デフォルト: warn。

#### ストレージとして追加

新しいプールを使用してVMまたはコンテナ・ストレージを構成する。デフォルト: `true`（作成時にのみ表示される）。

## 高度なオプション

### 最小サイズ

オブジェクトあたりの最小レプリカ数。PGのレプリカ数がこの数未満の場合、Cephはプール上のI/Oを拒否します。デフォルト: 2。

---

<sup>7</sup> 自動スケーリング <https://docs.ceph.com/en/quincy/rados/operations/placement-groups/#automated-scaling>

## クラッシュ・ルール

クラスタ内のオブジェクト配置のマッピングに使用するルール。これらのルールは、クラスタ内でのデータの配置方法を定義します。デバイスベースのルールについては、[Ceph CRUSH & デバイスクラス](#)を参照してください。

## # PG数

配置グループの数<sup>6</sup> プールが最初に持つべき配置グループの数。デフォルト: 128。

## 目標比率

プールで予想されるデータの比率。PGオートスケーラは、他の比率セットに対する比率を使用します。両方が設定されている場合は、ターゲットサイズよりも優先されます。

## 目標サイズ

プールに予想されるデータ量。PGオートスケーラはこのサイズを使用して最適なPG数を推定する。

## 最低# PG数

配置グループの最小数。この設定は、そのプールのPG数の下限を微調整するために使用されます。PGオートスケーラーは、この閾値未満のPGをマージしません。

Cephプールの処理に関する詳細は、[Cephプール操作マニュアル<sup>8</sup>](#) マニュアルを参照してください。

### 8.8.2 消去符号化プール

消去符号化 (EC) は「順方向エラー訂正」符号の一種であり、ある程度のデータ損失から回復することができる。消去符号化されたプールは、複製されたプールと比較してより多くの使用可能領域を提供することができるが、それはパフォーマンスの代償となる。

一方、消去コード化プールでは、データは $k$ 個のデータチャンクに分割され、さらに $m$ 個のコーディング (チェック) チャンクが追加される。これらのコーディング・チャンクは、データ・チャンクが欠落した場合にデータを再作成するために使用される。

符号化チャンクの数 $m$ は、データを失うことなく、いくつのOSDを失うことができるかを定義する。保存されるオブジェクトの総量は $k + m$ である。

## ECプールの作成

消去符号化 (EC) プールは、`pveceph` CLIツールを使用して作成できます。ECプールを計画する際には、レプリケートされたプールとは動作が異なるという事実を考慮する必要があります。

ECプールのデフォルトの`min_size`は`m`パラメータに依存します。 $m > 1$ の場合、`min_size`は $k + 1$ になります。Cephのドキュメントでは、保守的な`min_size`として $k + 2$ を推奨しています。<sup>9</sup>

利用可能なOSDが`min_size`未満である場合、プールへのIOは、再び十分なOSDが利用可能になるまでブロックされる。

---

<sup>8</sup>Cephプール操作 <https://docs.ceph.com/en/quincy/rados/operations/pools/>

<sup>9</sup>Ceph Erasure Coded Pool Recovery <https://docs.ceph.com/en/quincy/rados/operations/erasure-code/#erasure-coded-pool-recovery>

## 注

消去コード化されたプールを計画する際には、`min_size`に注意してください。`min_size`は、利用可能なOSDの数を定義します。さもないと、IOがブロックされる。

たとえば、`k = 2, m = 1`のECプールは、`size = 3, min_size = 2`となり、OSDが1つ故障しても運用を継続する。プールが`k = 2, m = 2`で構成されている場合、`サイズ = 4, min_size = 3`となり、1つのOSDが失われても運用を継続する。

新しいECプールを作成するには、以下のコマンドを実行する：

```
pveceph pool create <pool-name> --erasure-coding k=2,m=1
```

オプションのパラメータは`failure-domain`と`device-class`である。プールで使用されるECプロファイル設定を変更する必要がある場合は、新しいプロファイルで新しいプールを作成する必要があります。これにより、新しいECプールと、RBDオマップとその他のメタデータを格納するために必要な複製プールが作成される。最終的に、`<プール名>-data`プールと`<プール名>-meta`プールができる。デフォルトの動作では、一致するストレージ構成も作成されます。この動作が不要な場合、`--add_storages 0`パラメータを指定することで、この動作を無効にすることができる。ストレージ構成を手動で設定する場合、`data-pool`パラメータを設定する必要があることに注意してください。その場合のみ、ECプールがデータ・オブジェクトの格納に使われる。例えば

## 注

オプションのパラメータ`-size`、`-min_size`、および`-crush_rule`は、複製されたメタデータ・プールに使用されますが、消去コード化されたデータ・プールには使用されません。データ・プールの`min_size`を変更する必要がある場合は、後で変更できます。`size`と`crush_rule`パラメータは、消去コード化プールでは変更できません。

ECプロファイルをさらにカスタマイズする必要がある場合は、Cephツールで直接作成できます。

<sup>10</sup>そして`profile`パラメータで使用するプロファイルを指定

ナビゲーション

```
pveceph pool create <pool-name> --erasure-coding profile=<profile-name>。
```

## ECプールをストレージとして追加する

既存のECプールをストレージとしてProxmox VEに追加することができます。これはRBDプールが必要だが、追加のデータプールオプションが必要。

```
pvesm add rbd <ストレージ名> --pool <複製プール> --data-pool <ec-pool
```

---

### チップ

ローカルのProxmox VEクラスタで管理されていない外部のCephクラスタ用に、**キーリング**と**monhost**オプションを追加することを忘れないでください。

---

<sup>10</sup>Ceph Erasure Code Profile <https://docs.ceph.com/en/quincy/rados/operations/erasure-code/#erasure-code-profiles>

### 8.8.3 プールを破壊する

GUIでプールを破棄するには、ツリービューでノードを選択し、**Ceph → Pools**パネルに移動します。破棄するプールを選択し、[Destroy]ボタンをクリックします。プールの破棄を確認するにはプール名。

以下のコマンドを実行してプールを破棄する。`remove_storages`を指定して、関連するストレージも削除す

```
pveceph プール破壊 <名前
```

る。

#### 注

プールの削除はバックグラウンドで実行されるため、時間がかかることがあります。このプロセスの間、クラスタのデータ使用量が減少していることに気づくでしょう。

### 8.8.4 PGオートスケーラー

PGオートスケーラにより、クラスタは各プールに保存される(予想される)データ量を考慮し、適切な`pg_num`値を自動的に選択できます。Ceph Nautilusから使用できます。

調整が有効になる前に、PGオートスケーラー・モジュールをアクティブにする必要があるかもしれません。

```
ceph mgr module enable pg_autoscaler
```

オートスケーラーはプールごとに設定され、以下のモードがある：

`warn` 推奨される`pg_num`の値が現在の値と大きく異なる場合、健全性に関する警告が発行されます。

`off` 自動的な`pg_num` の調整は行われず、以下の場合に警告は発せられない。  
PG数は最適ではない。

スケーリングファクターは、将来のデータ保存を容易にするために、`target_size`, `target_size_rate`と`pg_num_min`オプションがあります。

#### 警告

- デフォルトでは、オートスケーラは、プールのPGカウントが3倍ずれている場合にチューニングを検討します。これは、データ配置のかなりのシフトにつながり、クラスタに高負荷をもたらす可能性があります。
- 

PGオートスケーラの詳細については、Cephのブログ - [Nautilusの新機能](#)を参照してください: PGマージとオートチューニング。

## 8.9 Ceph CRUSH & デバイスクラス

The screenshot shows the Proxmox VE web interface with the Ceph configuration page selected. The left sidebar is collapsed. The main area displays the Ceph configuration file with sections for global, client, mds, and mds.prod3. The crush map tab on the right shows a detailed configuration map with various devices and buckets.

```

[global]
auth_client_required = cephx
auth_cluster_required = cephx
auth_service_required = cephx
cluster_network = 192.168.30.64/20
fsid = fc4181a6-56eb-4f68-b452-8ba1f381ca2a
mon_allow_pool_delete = true
mon_host = 192.168.30.64 192.168.30.65 192.168.30.66
osd_pool_default_min_size = 2
osd_pool_default_size = 3
public_network = 192.168.30.64/20
cluster_network = 10.3.0.0/24

[client]
keyring = /etc/pve/priv/$cluster.$name.keyring

[mds]
keyring = /var/lib/ceph/mds/ceph-$id/keyring

[mds.prod3]
host = prod3
mds standby for name = pve

[mds.prod1]
host = prod1

Configuration Database

WHO          OPTION           VALUE
mon          auth_allow_insecure_global_id_reclaim  false
  
```

```

# begin crush map
tunable choose_local_tries 0
tunable choose_local_fallback_tries 0
tunable choose_total_tries 50
tunable chooseleaf_descend_once 1
tunable chooseleaf_vary_r 1
tunable chooseleaf_stable 1
tunable straw_calc_version 1
tunable allowed_bucket_algs 54

# devices
device 0 osd.0 class hdd
device 1 osd.1 class hdd
device 2 osd.2 class hdd
device 3 osd.3 class hdd
device 4 osd.4 class hdd
device 5 osd.5 class hdd
device 6 osd.6 class hdd
device 7 osd.7 class hdd
device 8 osd.8 class hdd
device 9 osd.9 class ssd
device 10 osd.10 class ssd
device 11 osd.11 class ssd
device 12 osd.12 class ssd
device 13 osd.13 class ssd
device 15 osd.15 class ssd
device 16 osd.16 class ssd
device 17 osd.17 class ssd

# types
type 0 osd
type 1 host
type 2 chassis
type 3 rack
type 4 row
type 5 pdu
type 6 pod
type 7 room
type 8 datacenter
type 9 zone
type 10 region
type 11 root

# buckets
host prod1 {
    id -3      # do not change unnecessarily
    id -4 class hdd   # do not change unnecessarily
    id -9 class ssd   # do not change unnecessarily
    # weight 6.178
    alg straw2
    hash 0      # jenkins1
    item next 0 weight 0.50?
  }
  
```

この<sup>11</sup> (Controlled Replication Under Scalable Hashing)アルゴリズムはCephの基盤である。

CRUSHはどこにデータを保存し、どこから取得するかを計算する。これには、中央のインデックスサービスが必要ないという利点がある。CRUSHは、OSD、バケット（デバイスの位置）、プール用のルールセット（データの複製）のマップを使用して動作します。

### 注

詳細については、CephドキュメントのCRUSH map<sup>a</sup>セクションを参照してください。

<sup>a</sup>CRUSHマップ <https://docs.ceph.com/en/quincy/rados/operations/crush-map/>

このマップは、異なるレプリケーション階層を反映するように変更できる。オブジェクトのレプリカは、望ましい分散を維持しながら、分離することができる（例えば、障害ドメイン）。

一般的な構成は、異なるCephプールに異なるクラスのディスクを使用することである。このため、Cephはルミナスでデバイスクラスを導入し、ルールセットを簡単に生成できるようにした。

デバイスクラスは、*ceph osd*ツリー出力で確認できます。これらのクラスは独自のルートバケットを表し、

```
ceph osd crush tree --show-shadow
```

以下のコマンドで確認できます。

<sup>11</sup>CRUSH <https://ceph.com/wp-content/uploads/2016/08/weil-crush-sc06.pdf>

上記コマンドの出力例：

身分	クラス	重量	タイプ名
証明書			
-16	nvme	2.18307	ルート・デフォルト ~nvme
-13	nvme	0.72769	ホスト sumi1~nvme
12	nvme	0.72769	osd.12
-14	nvme	0.72769	ホスト sumi2~nvme
13	nvme	0.72769	osd.13
-15	nvme	0.72769	ホスト sumi3~nvme
14	nvme	0.72769	osd.14
-1		7.70544	ルートデフォルト
-3		2.56848	ホスト sumi1
12	nvme	0.72769	osd.12
-5		2.56848	ホスト sumi2
13	nvme	0.72769	osd.13
-7		2.56848	ホスト sumi3

特定のデバイスクラス上のオブジェクトのみを配布するようにプールに指示するには、まずデバイスクラス用

```
ceph osd crush rule create-replicated <ルール名> <ルート> <障害ドメイン> <←'クラス
```

のルールセットを作成する必要があります：

<ルール名>	プールと接続するためのルール名（GUIおよびCLIで表示される）
<ルート>	どのクラッシュルートに属するか(デフォルトのCephルート「default」)
<failure-domain>	オブジェクトを配布すべき障害ドメイン(通常はホスト)
<class>	どのタイプのOSDバックингストアを使用するか（例： nvme、 ssd、 hdd）

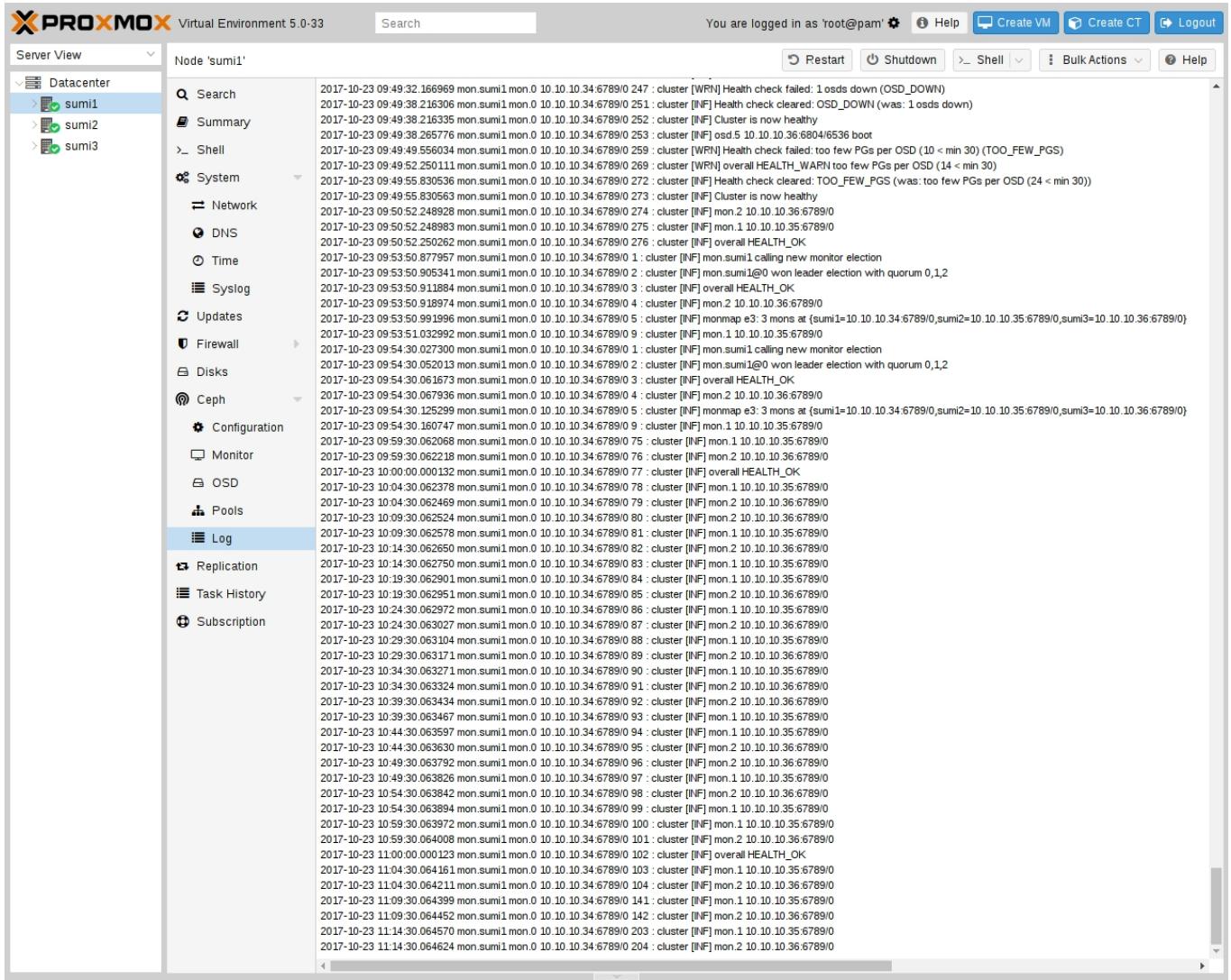
ルールがCRUSHマップにあれば、そのルールセットを使用するようプールに指示することができる。

```
ceph osd プールセット <プール名> crush_rule <ルール名>
```

## チップ

プールにすでにオブジェクトが含まれている場合は、これらを適宜移動する必要があります。セットアップによっては、これはクラスタに大きなパフォーマンスの影響を与える可能性があります。別の方法として、新しいプールを作成してディスクを個別に移動することができます。

## 8.10 Cephクライアント



前のセクションのセットアップに従って、Proxmox VEを設定して、VMとコンテナイメージを格納するためにこのようなプールを使用できます。GUIを使用して新しいRBDストレージを追加するだけです([Ceph RADOS Block Devices \(RBD\)](#)の項を参照)。

また、外部Cephクラスタ用の定義済みの場所にキーリングをコピーする必要があります。CephがProxmoxノード自体にインストールされている場合、これは自動的に実行されます。

### 注

ファイル名は `<storage_id> + ` .keyring` である必要があります。`<storage_id>` は `/etc/pve/storage.cfg` の `rbd:` の後に続く表現です。以下の例では、`my-ceph-storage` が `<storage_id>`:

```
mkdir /etc/pve/priv/ceph
cp /etc/ceph/ceph.client.admin.keyring /etc/pve/priv/ceph/my-ceph-storage.←
キホルダー
```

## 8.11 セフェフエス

Cephはファイルシステムも提供し、RADOSブロックデバイスと同じオブジェクトストレージ上で動作します。メタデータサーバ(MDS)を使用して、RADOSでバックアップされたオブジェクトをファイルとディレクトリにマッピングし、CephがPOSIX準拠の複製ファイルシステムを提供できるようにします。これにより、クラスタ化された可用性の高い共有ファイルシステムを簡単に構成できます。Cephのメタデータサーバは、ファイルがCephクラスタ全体に均等に分散されることを保証します。その結果、負荷が高い場合でも、NFSなどの従来の共有ファイルシステムアプローチで問題となる、単一のホストを圧倒することはありません。

Name	Host	Status	Address	Version
mds.prod1	prod1	up:standby	192.168.30.64.6805/3515028757	15.2.11
mds.prod2	prod2	up:active	192.168.30.65.6801/1445057454	15.2.11
mds.prod3	prod3	up:standby	192.168.30.66.6801/1826014728	15.2.11

Proxmox VEは、ハイパーコンバージドCephFSの作成と、バックアップ、ISOファイル、コンテナテンプレートを保存するストレージとしての既存のCephFSの使用の両方をサポートしている。

### 8.11.1 メタデータ・サーバー (MDS)

CephFSを機能させるには、少なくとも1つのメタデータサーバを構成して実行する必要があります。データシートは、Proxmox VE Web GUIのNode -> CephFSパネルまたはコマンドラインから作成できます：

mdsの作成

クラスタには複数のメタデータサーバーを作成できますが、デフォルト設定では一度にアクティブにできるのは1つのみです。データシートまたはそのノードが応答しなくなった（またはクラッシュした）場合、別のスタンバイ データシートがアクティブに昇格します。作成時に*hotstandby*パラメータオプションを使用することで、アクティブとスタンバイのデータシート間のハンドオーバーを高速化できます：

```
mdsスタンバイ再生 = true
```

を/etc/pve/ceph.confの各データシートセクションに追加します。この設定を有効にすると、指定されたデータシートはウォーム状態を維持し、アクティブなデータシートにポーリングします。

### 注

このアクティブなポーリングは、システムおよびアクティブなデータシートにさらなるパフォーマンスの影響を与えます。

### 複数のアクティブMDS

Luminous (12.2.x) 以降、複数のアクティブなメタデータサーバーを同時に実行することができますが、これは通常、大量のクライアントが並行して実行されている場合にのみ有用です。そうでない場合、データシートがシステムのボトルネックになることはほとんどありません。この設定を行う場合は、Cephのドキュメントを参照してください。[12](#)

## 8.11.2 CephFSの作成

Proxmox VEのCephFSの統合により、Webインターフェース、CLI、または外部APIインターフェースを使用して、簡単にCephFSを作成できます。これを動作させるには、いくつかの前提条件が必要です：

CephFSのセットアップを成功させるための前提条件：

- [Cephパッケージのインストール](#) - この作業を少し前に行った場合は、CephFS関連パッケージがすべてインストールされるように、最新のシステムで再実行するとよいでしょう。
- [モニターの設定](#)
- [OSDの設定](#)
- [少なくとも1つのデータシートをセットアップする](#)

これが完了したら、Web GUIのNode -> CephFSのいずれかでCephFSを作成します。

パネルやコマンドラインツールpvecephなどがある：

```
pveceph fs create --pg_num 128 --add-storage
```

これにより、cephfsという名前のCephFSが作成され、128の配置グループを持つcephfs\_dataという名前のデータ用プールと、データプールの配置グループの4分の1(32)を持つcephfs\_metadataという名前のメタデータ用プールが使用されます。セットアップに適した配置グループ番号(pg\_num)の詳細については、

Proxmox VE managed Ceph poolの章を参照するか、Cephのドキュメントを参照してください。<sup>6</sup>さらに--add-storageパラメータは、CephFSが正常に作成された後、Proxmox VEストレージ構成に追加します。

---

<sup>12</sup>複数のアクティブなデータシートデーモンを設定する <https://docs.ceph.com/en/quincy/cephfs/multimds/>

### 8.11.3 CephFSの破棄



#### 警告

CephFSを破棄すると、そのデータはすべて使用できなくなります。これは元に戻せません！

CephFSを完全かつ優雅に削除するには、以下の手順が必要です：

- すべての非Proxmox VEクライアントを切断します(たとえば、ゲストのCephFSをアンマウントします)。
- 関連するすべてのCephFS Proxmox VEストレージエントリを無効にします(自動的にマウントされないようにします)。
- 破棄するCephFS上のゲスト(ISOなど)から、使用済みのリソースをすべて削除します。
- を使用して、すべてのクラスタノードのCephFSストレージを手動でアンマウントします。

```
umount /mnt/pve/<STORAGE-NAME>
```

ここで、<STORAGE-NAME>はProxmox VEのCephFSストレージ名です。

- メタデータサーバ(MDS)を停止するか破棄して、そのCephFSでメタデータサーバ(MDS)が実行されていないことを確認します。これは、Webインターフェースまたはコマンドラインインターフェースで実行できます  
`pveceph stop --service mds.NAME`  
。後者の場合は、次のコマンドを実行します：

あるいは

```
pveceph mds destroy NAME
```

を破壊する。

アクティブなデータシートが停止または削除されると、スタンバイサーバーは自動的にアクティブに昇格することに注意してください。

- これで、CephFSを次のように破壊できます。

```
pveceph fs destroy NAME --remove-storages --remove-pools
```

これにより、基盤となるCephプールが自動的に破棄され、pve設定からストレージが削除されます。

これらの手順の後、CephFSを完全に削除し、他のCephFSインスタンスがある場合は、停止したメタデータサーバを再度起動してスタンバイとして動作させることができます。

## 8.12 Cephのメンテナンス

### 8.12.1 OSDの交換

Cephで最も一般的なメンテナンスタスクの1つは、OSDのディスクを交換することです。ディスクがすでに故障状態になっている場合は、[Destroy OSD](#)の手順を実行してください。Cephは、可能であれば残りのOSDにそれらのコピーを再作成します。このリバランシングは、OSD障害が検出されるか、OSDがアクティブに停止されるとすぐに開始されます。

## 注

プールのデフォルトのsize/min\_size (3/2)では、「size + 1」ノードが利用可能な場合にのみ復旧が開始される。この理由は、CephオブジェクトバランサCRUSHのデフォルトがフルノードを「障害ドメイン」としているためである。

GUIから機能しているディスクを交換するには、[Destroy OSDの手順](#)を実行します。唯一の追加は、OSDを停止して破壊する前に、クラスタが`HEALTH_OK`を表示するまで待つことです。

コマンドラインで以下のコマンドを使用する：

```
ceph osd out osd.<id>
```

OSDを安全に削除できるかどうかは、以下のコマンドで確認できます。

```
ceph osd セーフトウデストロイ osd.<id>
```

上記のチェックでOSDを取り外しても安全であることがわかったら、次のコマンドを続行できる：

```
systemctl stop ceph-osd@<id>.service  
pveceph osd destroy <id>
```

古いディスクを新しいディスクと交換し、[OSDの作成](#)で説明したのと同じ手順を使用する。

## 8.12.2 トリム/廃棄

VMやコンテナで`fstrim` (discard) を定期的に実行するのは良い習慣だ。これにより、ファイルシステムがもう使用していないデータ・ブロックが解放される。データ使用量とリソース負荷が軽減される。最近のオペレーティング・システムのほとんどは、ディスクに対して定期的にこのようなdiscardコマンドを発行している。仮想マシンが[ディスク廃棄オプション](#)を有効にしていることを確認するだけでよい。

## 8.12.3 スクラップ & ディープスクラップ

Cephは配置グループをスクラップすることで、データの整合性を確保します。CephはPG内のすべてのオブジェクトの健全性をチェックします。スクラップには、毎日の安価なメタデータチェックと毎週のディープデータチェックの2つの形式があります。毎週のディープスクラップはオブジェクトを読み取り、チェックサムを使用してデータの整合性を確保します。スクラップの実行がビジネス(パフォーマンス)の必要性を妨げる場合は、スクラップの実行時間を調整することができます。[13](#) を実行する時間を調整できます。

## 8.13 Cephの監視とトラブルシューティング

Cephツールを使用するか、Proxmox VE APIを介してステータスにアクセスするかして、Cephデプロイの健全性を最初から継続的に監視することが重要です。

以下のCephコマンドを使用して、クラスタが健全かどうか(*HEALTH\_OK*)、警告があるかどうか(*HEALTH\_WARN*)、またはエラーかどうか(*HEALTH\_ERR*)を確認できます。クラスタが不健全な状態にある場合、以下のステータスコマンドを使用すると、現在のイベントと取るべきアクションの概要もわかります。

<sup>13</sup>Ceph scrubbing <https://docs.ceph.com/en/quincy/rados/configuration/osd-config-ref/#scrubbing>

```
# 単一時間出力 pve#  
ceph -s  
# ステータスの変更を継続的に出力する（停止するにはCTRL+Cを押す）  
pve# ceph -w
```

より詳細に表示するには、すべてのCephサービスに /var/log/ceph/ 配下のログファイルがあります。 詳細が必要な場合は、ログレベルを調整できます。[14](#)。

Cephクラスタのトラブルシューティングの詳細については [15](#) を参照してください。

<sup>14</sup>Ceph ログとデバッグ <https://docs.ceph.com/en/quincy/rados/troubleshooting/log-and-debug/>

<sup>15</sup>Ceph トラブルシューティング <https://docs.ceph.com/en/quincy/rados/troubleshooting/>

## 第9章

# ストレージ・レプリケーション

`pvesr` コマンドラインツールは Proxmox VE ストレージレプリケーションフレームワークを管理します。ストレージレプリケーションは、ローカルストレージを使用するゲストに冗長性をもたらし、移行時間を短縮します。

ゲストボリュームを別のノードにレプリケートし、共有ストレージを使用せずにすべてのデータを利用できるようにします。レプリカはスナップショットを使用して、ネットワーク経由で送信されるトラフィックを最小限に抑えます。そのため、最初の完全同期の後、新しいデータは増分的にしか送信されません。ノードに障害が発生した場合でも、ゲストデータはレプリケートされたノードで利用可能です。

レプリケーションは設定可能な間隔で自動的に行われる。最小レプリケーション間隔は1分で、最大間隔は週に1回です。これらの間隔を指定するために使用されるフォーマットは、`systemd` カレンダーイベントのサブセットです：

ゲストを複数のターゲットノードにレプリケートすることは可能ですが、同じターゲットノードに2回レプリケートすることはできません。各レプリケーションの帯域幅は、ストレージやサー

バーの過負荷を避けるために制限することができます。

ゲストがすでにレプリケートされているノードに移行する場合は、最後のレプリケーション以降の変更（いわゆるデルタ）のみを転送する必要があります。これにより、所要時間が大幅に短縮されます。ゲストをレプリケーションターゲットノードに移行すると、レプリケーションの方向が自動的に切り替わります。

例えばVM100は現在nodeA上にあり、nodeBにレプリケートされている。VM100をnodeBに移行すると、自動的にnodeBからnodeAにレプリケートされます。

ゲストがレプリケートされていないノードに移行する場合は、ディスクデータ全体を送信する必要があります。

す。移行後、レプリケーションジョブはこのゲストを設定されたノードにレプリケートし続けます。

**重要**

高可用性はストレージ・レプリケーションとの組み合わせで許可されるが、最後に同期された時刻とノードが故障した時刻の間に何らかのデータ損失が発生する可能性がある。

## 9.1 対応ストレージタイプ

表9.1：ストレージの種類

説明	プラグインタイプ	スナップ写真	安定
ZFS (ローカル)	ゼットスプール	はい	はい

## 9.2 スケジュール形式

レプリケーションはカレンダーのイベントを使ってスケジュールを設定する。

## 9.3 エラー処理

レプリケーション・ジョブに問題が発生すると、エラー状態になります。この状態では、設定されたレプリケーション間隔は一時的に中断されます。失敗したレプリケーションは、30分間隔で繰り返し再試行されます。これが成功すると、元のスケジュールが再び有効になります。

### 9.3.1 考えられる問題

最も一般的な問題のいくつかは、以下のリストにあります。セットアップによっては別の原因があるかもしれません。

- ネットワークが機能していない。
- レプリケーション先のストレージに空き容量が残っていない。
- ターゲット・ノードで利用可能な同じストレージIDを持つストレージ

---

#### 注

レプリケーション・ログを使えば、いつでも問題の原因を突き止めることができる。

---

### 9.3.2 エラー時のゲストの移行

重大なエラーが発生した場合、仮想ゲストは障害が発生したノード上でスタッカする可能性があります。その場合、手動で再び動作するノードに移動させる必要があります。

### 9.3.3 例

ノードAで2つのゲスト（VM 100とCT 200）を実行し、ノードBにレプリケートしているとします。そこで、ゲストを手動でノードBに移行する必要があります。

---

- ・ノードBにsshで接続するか、ウェブUIでシェルを開きます。
- ・クラスタがクオーレートかどうかをチェックする

```
# pvecm モニタス
```

- ・クオーラムがない場合は、まずこれを修正し、ノードを再び操作できるようにすることを強くお勧めしま

```
# pvecm 期待値 1
```

す。現時点でこれが不可能な場合に限り、以下のコマンドを使用して現在のノードにクオーラムを適用することができます：

**警告**

予想される投票が設定されている場合、クラスタに影響を与える変更（ノード、ストレージ、仮想ゲストの追加/削除など）は絶対に避けてください。重要なゲストを再び稼働させるか、クオラムの問題そのものを解決する場合にのみ使用してください。

- 両方のゲスト設定ファイルを元のノードAからノードBに移動します:

```
# mv /etc/pve/nodes/A/qemu-server/100.conf /etc/pve/nodes/B/qemu-server-'100.conf  
# mv /etc/pve/nodes/A/lxc/200.conf /etc/pve/nodes/B/lxc/200.conf
```

- これでゲストを再びスタートさせることができる:

```
# qm start 100  
# pct start  
200
```

VMIDとノード名をそれぞれの値に置き換えることを忘れないでください。

## 9.4 求人管理

The screenshot shows the Proxmox VE Web GUI interface for managing virtual machines. On the left, there's a sidebar with various management options like Summary, Console, Hardware, Options, Task History, Monitor, Backup, Replication, Snapshots, Firewall, and Permissions. The 'Replication' option is currently selected. In the main area, a specific VM (VMID 99999) is selected. A 'Create: Replication Job' dialog box is open over the main content. The dialog has fields for 'CT/VM ID' (set to 99999), 'Target' (set to 'demohost2'), 'Schedule' (set to '\*15 - Every 15 minutes'), 'Rate (MB/s)' (set to 'unlimited'), and a 'Comment' field which is empty. There's also a checked checkbox for 'Enabled'. At the bottom of the dialog are 'Help' and 'Create' buttons.

Web GUIを使用して、レプリケーション・ジョブの作成、変更、削除を簡単に行うことができます。さらに、コマンドライン・インターフェース（CLI）ツールのpvesrを使用することもできます。

レプリケーション・パネルは、Web GUIのすべてのレベル（データセンター、ノード、仮想ゲスト）で見つ

することができます。どのジョブが表示されるかは、すべてのジョブ、ノード固有のジョブ、ゲスト固有のジョブで異なります。

新しいジョブを追加する際、まだ選択されていない場合はゲストとターゲットノードを指定する必要があります。レプリケーション・スケジュールは、デフォルトの15分ではなく、設定することができます。レプリケーションジョブにレート制限を課すことができます。レート制限は、ストレージの負荷を許容範囲に保つのに役立ちます。

レプリケーション・ジョブはクラスタ全体で一意のIDによって識別されます。このIDは、ジョブ番号に加えてVMIDで構成されます。CLIツールを使用する場合のみ、このIDを手動で指定する必要があります。

## 9.5 コマンドライン・インターフェースの例

ID 100のゲストに対して、帯域幅を10Mbps（メガバイト/秒）に制限して5分ごとに実行するレプリケーションジョブを作成します。

```
# pvesr create-local-job 100-0 pve1 --schedule "* /5" --rate 10  
ジョブを作成します。
```

ID 100-0 のアクティブなジョブを無効にする。

```
# pvesr disable 100-0
```

IDが100-0の非活性化ジョブを有効にする。

```
# pvesr enable 100-0
```

ID100-0のジョブのスケジュール間隔を1時間に1回に変更する。

```
# pvesr update 100-0 --schedule '* /00'
```

## 第10章

### QEMU/KVM仮想マシン

QEMU（Quick Emulatorの略）は、物理コンピュータをエミュレートするオープンソースのハイパーバイザーです。QEMUが実行されているホストシステムから見ると、QEMUはユーザープログラムであり、パーティション、ファイル、ネットワークカードといった多数のローカルリソースにアクセスできる。

エミュレートされたコンピュータで動作するゲストオペレーティングシステムは、これらのデバイスにアクセスし、あたかも実際のハードウェア上で動作しているかのように動作します。例えば、ISO イメージを QEMU にパラメータとして渡すと、エミュレートされたコンピュータで実行されている OS には、CD ドライブに挿入された本物の CD-ROM が表示されます。

QEMUはARMからSparcまで多種多様なハードウェアをエミュレートできますが、Proxmox VEは32ビットおよび64ビットのPCクローンのエミュレーションにのみ対応しています。PCクローンのエミュレーションは、エミュレートするアーキテクチャがホストアーキテクチャと同じ場合にQEMUを大幅に高速化するプロセッサ拡張機能を利用できるため、最も高速なエミュレーションの1つでもあります。

---

#### 注

KVM（Kernel-based Virtual Machine）という用語に遭遇することがあります。これは、QEMU が Linux KVM モジュールを介して、仮想化プロセッサ拡張のサポートを受けながら実行されていることを意味します。Proxmox VEのQEMUは常にKVMモジュールをロードしようとするため、Proxmox VEのコンテキストではQEMUとKVMは同じ意味で使用できます。

---

Proxmox VE内のQEMUは、ブロックデバイスやPCIデバイスにアクセスするために必要なため、ルートプロセスとして実行されます。

---

## 10.1 エミュレートされたデバイスと準仮想化デバイス

QEMUによってエミュレートされるPCハードウェアには、マザーボード、ネットワークコントローラ、SCSI、IDE、SATAコントローラ、シリアルポート（完全なリストはkvm(1)のマニュアルページで見ることができる）が含まれ、これらはすべてソフトウェアでエミュレートされる。これらのデバイスはすべて、既存のハードウェアデバイスとまったく同等のソフトウェアデバイスであり、ゲストで実行されているOSに適切なドライバがあれば、実際のハードウェア上で実行されているかのようにデバイスを使用することができます。これにより、QEMUは変更されていないオペレーティングシステムを実行することができます。

ハードウェアで実行する予定だったものをソフトウェアで実行すると、ホストCPUに多くの余分な作業が発生するためです。これを軽減するために、QEMUはゲストOSに準仮想化デバイスを提示することができ、ゲストOSはQEMU内で実行されていることを認識し、ハイパーバイザーと協力します。

QEMUはvirtio仮想化標準に依存しているため、準仮想化された汎用ディスクコントローラ、準仮想化されたネットワークカード、準仮想化されたシリアルポート、準仮想化されたSCSIコントローラなど、準仮想化されたvirtioデバイスを表示することができます。

### チップ

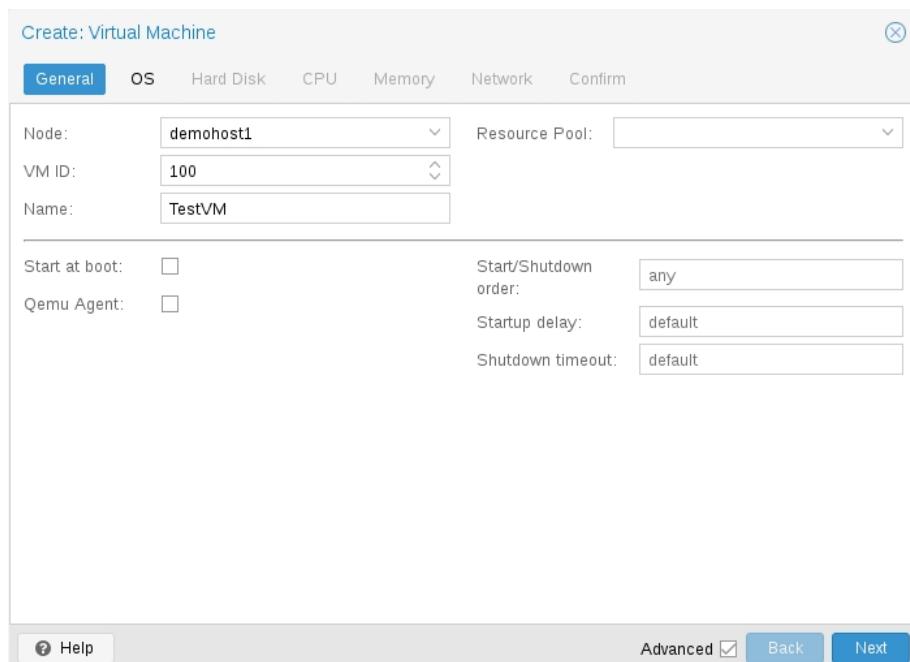
可能な限り、virtio デバイスを使用することを強く推奨します。virtio デバイスは大きな性能向上をもたらし、一般に保守が行き届いているからです。エミュレートされた IDE コントローラに対して virtio 汎用ディスクコントローラを使用すると、bonnie++(8) で測定されたシーケンシャル書き込みスループットが 2 倍になります。virtio ネットワークインターフェイスを使用すると、iperf(1) で計測されたように、エミュレートされた Intel E1000 ネットワークカードのスループットの最大 3 倍を提供することができます。<sup>a</sup>

<sup>a</sup>KVM wiki [https://www.linux-kvm.org/page/Using\\_VirtIO\\_NIC](https://www.linux-kvm.org/page/Using_VirtIO_NIC) のベンチマークを参照。

## 10.2 仮想マシンの設定

一般的にProxmox VEは、仮想マシン(VM)に対して適切なデフォルトを選択しようとします。パフォーマンスを低下させたり、データを危険にさらす可能性があるため、変更する設定の意味を理解してください。

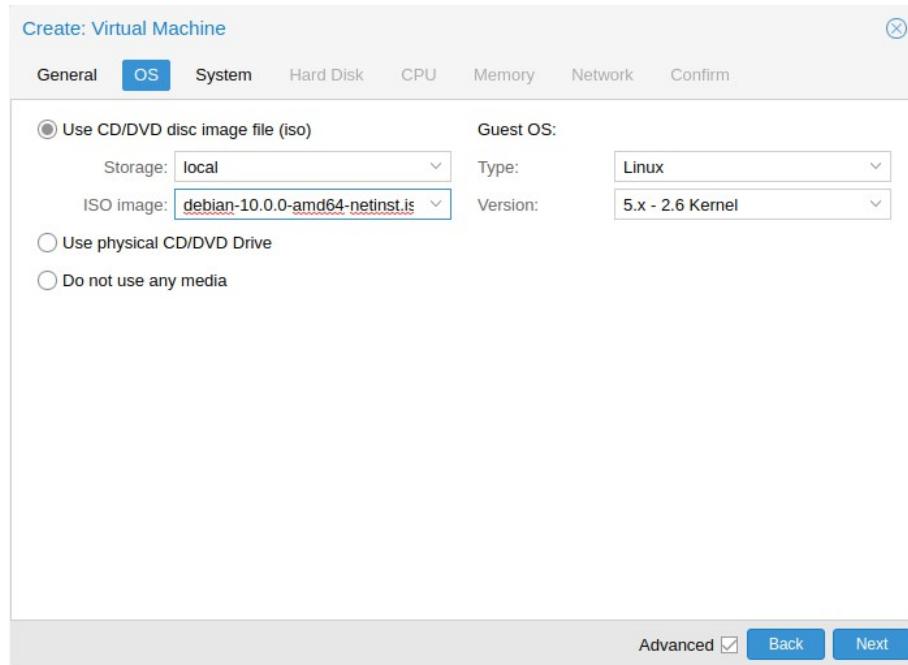
### 10.2.1 一般設定



VMの一般的な設定には以下が含まれる。

- ・ **ノード**: VMが実行される物理サーバー
- ・ **VM ID**: VMを識別するためのProxmox VEインストール内で一意の番号。
- ・ **Name**: VMの説明に使用する自由形式のテキスト文字列
- ・ **リソースプール**: VMの論理グループ

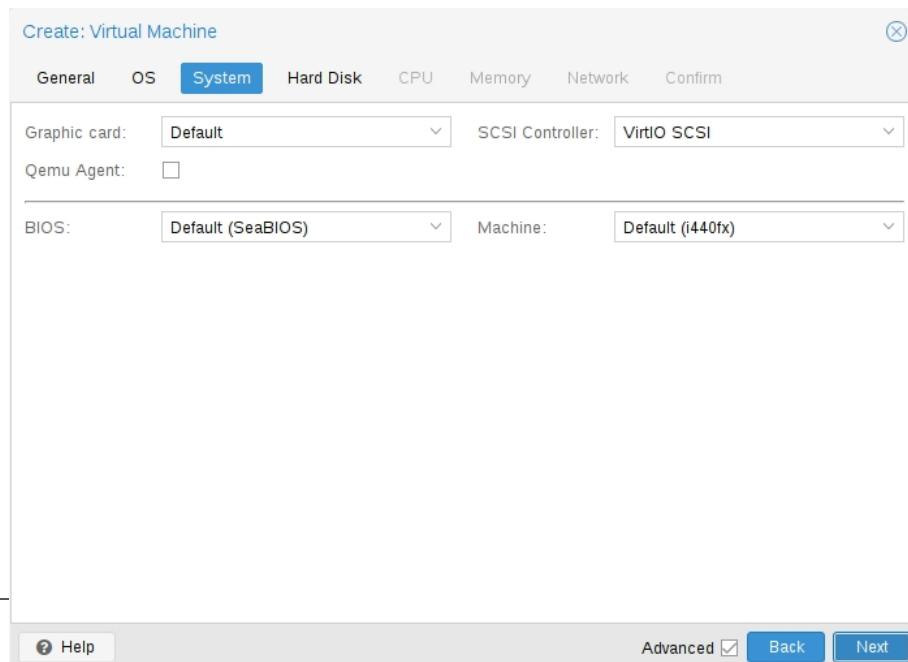
## 10.2.2 OS設定



仮想マシン(VM)を作成する際、適切なオペレーティングシステム(OS)を設定することで、Proxmox VEはいくつかの低レベルパラメータを最適化することができます。例えば、Windows OSは、BIOSクロックがローカル時間を使用することを期待しますが、UnixベースのOSは、BIOSクロックがUTC時間を使用することを期待します。

## 10.2.3 システム設定

VMの作成時に、新しいVMの基本的なシステム・コンポーネントをいくつか変更できる。[どのディスプレイ・タイプを使用するかを指定できる。](#)



さらに、[SCSI コントローラも変更できます。](#) QEMU ゲストエージェントをインストールする予定がある場合、または選択した ISO イメージがすでに出荷されており、自動的にインストールされる場合は、QEMU エージェントにチェックを入れるとよいでしょう、

これはProxmox VEに、より多くの情報を表示し、いくつかのアクション（シャットダウンやスナップショットなど）をよりインテリジェントに完了するために、その機能を使用できることを知らせます。

Proxmox VEでは、[SeaBIOSとOVMF](#)という異なるファームウェアとマシンタイプでVMをブートすることができます。ほとんどの場合、[PCIe パススルー](#)を使用する場合のみ、デフォルトの SeaBIOS から OVMF に切り替えたい。

## マシンタイプ

VMのマシン・タイプは、VMの仮想マザーボードのハードウェア・レイアウトを定義する。デフォルトの [Intel 440FX](#)か、仮想PCIeバスも提供する[Q35](#)チップセットのいずれかを選択できる。さらに、[vIOMMU](#)の実装も選択できる。

## マシン・バージョン

各マシンタイプはQEMUでバージョン管理されており、あるQEMUバイナリは多くのマシンバージョンをサポートしています。新しいバージョンは、新機能のサポート、修正、一般的な改良をもたらすかもしれません。ただし、仮想ハードウェアのプロパティも変更されます。ゲストの観点からの突然の変更を回避し、VM状態の互換性を確保するために、RAMを使用したライブマイグレーションとスナップショットは、新しいQEMUインスタンスで同じマシンバージョンを使用し続けます。

Windowsゲストの場合、マシンバージョンは作成時に固定される。Windowsは仮想ハードウェアの変更に敏感だからである。例えば、ネットワークデバイスの列挙は、異なるマシンバージョンでは異なるかもしれません。Linuxのような他のOSは、通常このような変更にうまく対応できる。これらのOSでは、デフォルトで最新のマシン・バージョンが使用されます。これは、再スタート後、QEMUバイナリがサポートする最新のマシンバージョンが使用されることを意味します（例えば、QEMU 8.1がサポートする最新のマシンバージョンは、各マシンタイプのバージョン8.1です）。

## 新しいマシン・バージョンへのアップデート

非常に古いマシンのバージョンは、QEMUで非推奨になる可能性があります。例えば i440fxマシンタイプでは1.4から1.7。これらのマシンバージョンのサポートは、ある時点で打ち切られることが予想されます。非推奨の警告が表示されたら、マシンのバージョンを新しいものに変更する必要があります。最初にバックアップを取り、ゲストがハードウェアを認識する方法の変更に備えるようにしてください。シナリオによっては、特定のドライバの再インストールが必要になるかもしれません。また、これらのマシンバージョンで取得された RAM のスナップショット（つまり `runningmachine` 設定エントリ）をチ

エックする必要があります。残念ながら、スナップショットのマシンバージョンを変更する方法はないので、スナップショットからデータを救い出すにはスナップショットをロードする必要があります。

#### 10.2.4 ハードディスク

##### バス/コントローラー

QEMUは多数のストレージコントローラをエミュレートできる：

---

##### チップ

パフォーマンス上の理由と、メンテナンスのしやすさから、**VirtIO SCSI** または **VirtIO Block** コントローラの使用を強く推奨します。

---

- **IDE**コントローラは、1984年のPC/ATディスクコントローラに遡る設計となっている。このコントローラが最近の設計に取って代わられたとしても、思いつく限りのすべてのOSがこのコントローラをサポートしているため、2003年以前にリリースされたOSを使いたい場合に最適な選択肢となる。このコントローラには最大4台のデバイスを接続できる。
  - SATA (Serial ATA) コントローラは2003年製で、より近代的な設計となっており、より高いスループットとより多くのデバイスを接続することができます。このコントローラには最大6台のデバイスを接続できます。
  - 1985年に設計された**SCSI**コントローラは、サーバグレードハードウェアによく見られ、最大14台のストレージデバイスを接続できます。Proxmox VEはデフォルトでLSI 53C895Aコントローラをエミュレートします。
- パフォーマンスを重視する場合は、*VirtIO SCSI*シングルタイプのSCSIコントローラを使用し、接続ディスクの**IOスレッド**設定を有効にすることを推奨します。Proxmox VE 7.3以降、新しく作成されるLinux VMのデフォルトはこれです。各ディスクは独自の*VirtIO SCSI*コントローラーを持ち、QEMUは専用スレッドでディスクのIOを処理します。Linuxディストリビューションは2012年から、FreeBSDは2014年からこのコントローラをサポートしている。Windows OSの場合は、インストール時にドライバを含む追加のISOを提供する必要があります。
- **VirtIO**またはvirtio-blkと呼ばれることが多い**VirtIO Block**コントローラは、古いタイプの準仮想化コントローラです。機能面では、*VirtIO SCSI*コントローラーに取って代わられました。

## 画像フォーマット

各コントローラには、設定されたストレージに存在するファイルまたはブロックデバイスによってバックされるエミュレートされたハードディスクをいくつか取り付けます。ストレージタイプの選択により、ハードディスクイメージのフォーマットが決まります。ブロックデバイスを使用するストレージ (LVM、ZFS、Ceph) では **raw ディスクイメージ形式**が必要ですが、ファイルベースのストレージ (Ext4、NFS、CIFS、GlusterFS) では **raw ディスクイメージ形式**または**QEMU イメージ形式**のいずれかを選択できます。

- **QEMUイメージフォーマット**は、スナップショットやディスクイメージのシンプロビジョニングを可能にするコピーオンライトフォーマットです。
- **rawディスクイメージ**は、ハードディスクのビット単位のイメージで、Linuxのブロックデバイスでddコマンドを実行したときに得られるものと似ている。このフォーマットは、それ自体ではシンプロビジョニ

グやスナップショットをサポートしないため、これらのタスクにはストレージレイヤーの協力が必要です。しかし、**QEMUイメージフォーマットよりも最大10%高速**になる可能性があります。<sup>1</sup>

- **VMwareイメージ・フォーマット**が意味を持つのは、ディスク・イメージを他のハイパーバイザーにインポート／エクスポートする場合だけです。

## キャッシュ・モード

ハードドライブの**キャッシュモード**を設定すると、ホストシステムがゲストシステムにブロック書き込み完了を通知する方法に影響します。デフォルトの **No cache** は、各ブロックが物理ストレージの書き込みキューニー到達したときにゲストシステムに書き込み完了が通知され、ホストのページキャッシングは無視されることを意味します。これにより、安全性と速度のバランスがとれます。

ProxmoxのVEバックアップマネージャがVMのバックアップを実行するときにディスクをスキップするように設定するには

そのディスクに**バックアップオプションはない。**

---

<sup>1</sup>参照 この ベンチマーク を参照してください。 詳細

[https://events.static.linuxfound.org/sites/events/files/slides/- CloudOpen2013\\_Khoa\\_Huynh\\_v3.pdf](https://events.static.linuxfound.org/sites/events/files/slides/- CloudOpen2013_Khoa_Huynh_v3.pdf)

Proxmox VE のストレージレプリケーションメカニズムでレプリケーションジョブの開始時にディスクをスキップさせたい場合は、そのディスクにレプリケーションをスキップするオプションを設定します。Proxmox VE 5.0 のレプリケーションでは、ディスクイメージは `zfspool` タイプのストレージ上にある必要があるため、VM にレプリケーションが設定されている場合にディスクイメージを他のストレージに追加すると、このディスクイメージのレプリケーションをスキップする必要があります。

## トリム/廃棄

ストレージがシンプロビジョニングをサポートしている場合（Proxmox VEガイドのストレージの章を参照）、ドライブで **Discard** オプションを有効にすることができます。**Discard** が設定され、**TRIM** が有効なゲスト OS の場合<sup>2</sup> VM のファイルシステムがファイルを削除した後にブロックを未使用とマークすると、コントローラはこの情報をストレージに伝え、ストレージはそれに応じてディスクイメージを縮小します。ゲストが **TRIM** コマンドを発行できるようにするには、ドライブの **Discard** オプションを有効にする必要があります。ゲストオペレーティングシステムによっては、**SSD Emulation** フラグを設定する必要もあります。**VirtIO Block** ドライブの **Discard** は、Linux Kernel 5.0 以降を使用しているゲストでのみサポートされていることに注意してください。

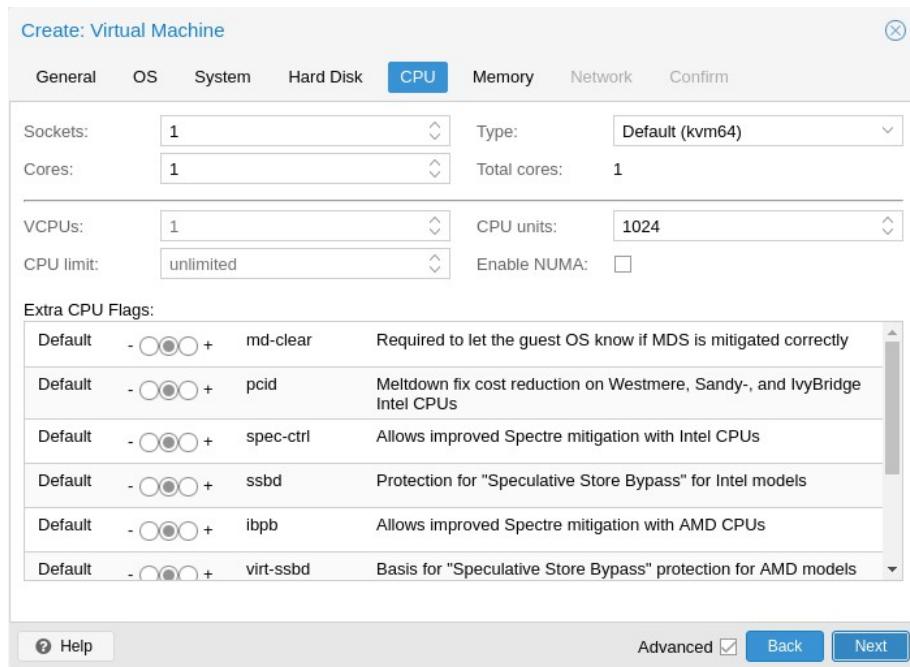
ドライブを回転ハードディスクではなくソリッドステートドライブとしてゲストに表示させたい場合は、そのドライブに **SSD エミュレーション** オプションを設定することができます。基盤となるストレージが実際に SSD でバックアップされている必要はなく、この機能はあらゆるタイプの物理メディアで使用できます。**SSD エミュレーション** は、**VirtIO Block** ドライブではサポートされていません。

## IOスレッド

**IO Thread** オプションは、**VirtIO** コントローラでディスクを使用する場合、またはエミュレートされるコントローラのタイプが **VirtIO SCSI シングル** である場合に **SCSI** コントローラでディスクを使用する場合にのみ使用できます。**IO Thread** を有効にすると、QEMU は、メインイベントループまたは vCPU スレッドですべての I/O を処理するのではなく、ストレージ コントローラごとに 1 つの I/O スレッドを作成します。利点の 1 つは、作業の分散と基礎となるストレージの利用が向上することです。もう 1 つの利点は、メインスレッドも vCPU スレッドもディスク I/O によってブロックされないため、非常に I/O 負荷の高いホストワークロードでゲストの待ち時間（ハング）が短縮されることです。

<sup>2</sup>TRIM、UNMAP、そして廃棄 [https://en.wikipedia.org/wiki/Trim\\_%28computing%29](https://en.wikipedia.org/wiki/Trim_%28computing%29)

## 10.2.5 CPU



CPUソケットは、PCマザーボード上の物理的なスロットで、CPUを差し込むことができる。このCPUには、独立した処理ユニットであるコアを1つ、または複数搭載することができます。CPUソケットが1つで4コアか、2つで2コアかは、性能の観点からはほとんど関係ありません。しかし、ソフトウェアのライセンスによっては、マシンのソケット数に依存するものもある。

仮想CPU（コアとソケット）の数を増やすと、通常はパフォーマンスが向上しますが、これはVMの使用状況に大きく依存します。マルチスレッド・アプリケーションでは、もちろん仮想CPUの数が多い方が有利です。仮想CPUを1つ追加ごとに、QEMUはホスト・システム上に新しい実行スレッドを作成するからです。VMのワークロードについて確信が持てない場合、通常は総コア数を2に設定するのが安全です。

### 注

すべてのVMの全体のコア数がサーバーのコア数よりも大きければ、まったく問題ありません（たとえば、8コアしかないマシンに、それぞれ4コア（=合計16コア）のVMを4台配置する場合など）。この場合、ホスト・システムは、標準的なマルチスレッド・アプリケーションを実行する場合と同様に、QEMUの実行スレッドをサーバー・コア間でバランスさせます。しかし、Proxmox VEは、物理的に利用可能なコア数よりも多くの仮想CPUコアでVMを起動することを防ぎます。

### リソース制限

#### cputlimit

仮想コアの数に加え、VMで使用可能な「ホストCPU時間」の合計も`cpulimit`オプションで設定できる。これはCPU時間をパーセントで表した浮動小数点値で、1.0は100%に相当します、2.5%から250%といった具合だ。1つのプロセスが1つのコアをフルに使えば、CPU時間は100%になる。使用率。4つのコアを持つVMがすべてのコアをフルに使用した場合、理論的には400%の使用率になります。QEMUはvCPUコア以外にVMペリフェラル用の追加スレッドを持つことができるため、実際には使用率はもう少し高くなる可能性があります。

この設定は、VMがいくつかのプロセスを並行して実行するために複数のvCPUを持つ必要があるが、VM全体としてはすべてのvCPUを同時に100%で実行することはできない場合に有効である。

例えば、8つの仮想CPUを持つことで恩恵を受けられる仮想マシンがあるとして、その仮想マシンが全負荷時に8つのコアを最大に使用できるようにしたくないとなります。これを解決するには、**cpulimit**を4.0 (=400%)に設定すればよい。これは、VMが8つのプロセスを同時に実行して8つすべての仮想CPUを完全に利用する場合、各vCPUは物理コアから最大50%のCPU時間を受け取ることを意味する。しかし、VMのワークロードが4つの仮想CPUしか完全に利用しない場合でも、物理コアから最大100%のCPU時間を受け取ることができます。

### 注

VMは、その構成によっては、ネットワーキングやIOオペレーション、ライブマイグレーションなど、追加のスレッドを使用することができる。そのため、VMは仮想CPUが使用できる以上のCPU時間を使用するように表示されることがある。VMが割り当てられたvCPU以上のCPU時間を使用しないようにするには、**cpulimit**を総コア数と同じ値に設定します。

### クプンティス

**cpuunits**オプション（最近ではCPU sharesまたはCPU weightと呼ばれることが多い）を使用すると、VMが他の実行中のVMと比較して取得するCPU時間を制御できます。これは相対的な重みで、デフォルトは100（またはホストがレガシー`cgroup v1`を使用している場合は1024）です。VMのウェイトを上げると、他のウェイトの低いVMと比較してスケジューラによって優先されます。

たとえば、VM 100がデフォルトを100に設定し、VM 200が200に変更された場合、後者のVM 200は最初のVM 100の2倍のCPU帯域幅を受け取ることになる。

より詳しい情報は `man systemd.resource-control` を参照してください。

とCPUWeightをcpuunits設定に変更する。参考文献や実装の詳細については、Notesセクションを参照してください。

### 親和性

アフィニティ・オプションを使用すると、VMのvCPUの実行に使用する物理CPUコアを指定できます。I/OなどのペリフェラルVMプロセスは、この設定の影響を受けません。**CPUアフィニティはセキュリティ機能ではないことに注意してください。**

**CPUアフィニティを強制することは、場合によっては意味のあることですが、複雑さとメンテナンスの手間の増加を伴います。** 例えば、後からVMを追加したり、VMをCPUコアの少ないノードに移行したりする場合です。また、一部のCPUがフルに利用され、他のCPUがほとんどアイドル状態である場合、非同期的でシス

テム・パフォーマンスが制限されやすくなります。

**アフィニティ**はtaskset CLIツールで設定する。man cpusetのList FormatにあるホストCPU番号 (lscpuを参照) を受け付ける。このASCII 10進数リストには、数字だけでなく数字の範囲も含めることができます。例えば、**アフィニティ**0-1,8-11（拡張0,1,8,9,10,11）を指定すると、VMはこれら6つの特定のホスト・コアだけで実行できるようになる。

## CPUタイプ

QEMUは、486から最新のXeonプロセッサーまで、さまざまな**タイプのCPU**をエミュレートできる。プロセッサの世代が新しくなるごとに、ハードウェア支援型3Dレンダリング、乱数生成、メモリ保護などの新機能が追加されます。また、現在の世代は、バグやセキュリティの修正を伴う[マイクロコードアップデート](#)によってアップグレードすることができます。

通常、ホスト・システムのCPUに近いプロセッサ・タイプをVMに選択する必要があります。これは、ホストのCPU機能（CPUフラグとも呼ばれます）がVMでも利用できることを意味します。完全に一致させたい場合は、CPUタイプをホストに設定すると、VMはホスト・システムと全く同じCPUフラグを持つことになります。

しかし、これには欠点もある。異なるホスト間でVMのライブマイグレーションを行いたい場合、VMは異なるCPUタイプや異なるマイクロコードバージョンを持つ新しいシステム上で終了するかもしれません。ゲストに渡されたCPUフラグが欠落していると、QEMUプロセスは停止してしまいます。これを解決するために、QEMUには独自の仮想CPUタイプがあり、Proxmox VEはデフォルトでこれを使用します。

バックエンドのデフォルトはkvm64で、基本的にすべてのx86\_64ホストCPUで動作し、新しいVMを作成する際のUIのデフォルトはx86-64-v2-AESで、インテルならウェストミアを始めとするホストCPU、AMDなら少なくとも第4世代のオプテロンが必要だ。

要するにだ：

ライブマイグレーションを気にしない場合や、すべてのノードが同じCPUと同じマイクロコードバージョンを持つ均質なクラスタの場合は、CPUタイプをホストに設定してください。

ライブマイグレーションとセキュリティを重視し、Intel CPUのみ、またはAMD CPUのみを使用している場合は、クラスタのCPUの世代が最も低いモデルを選択してください。

セキュリティなしのライブマイグレーションを重視する場合や、Intel/AMDクラスタが混在している場合は、互換性の最も低い仮想QEMU CPUタイプを選択してください。

---

## 注

インテルとAMDのホストCPU間のライブマイグレーションは、動作保証がない。

---

[QEMUで定義されているAMDとIntelのCPUタイプの一覧](#)も参照してください。

## QEMU CPUタイプ

QEMUは、IntelとAMDの両方のホストCPUと互換性のある仮想CPUタイプも提供している。

---

## 注

仮想CPUタイプのSpectre脆弱性を緩和するには、関連するCPUフラグを追加する必要がある。

---

歴史的に、Proxmox VEは*kvm64* CPUモデルで、Pentium 4レベルのCPUフラグを有効にしていたため、特定のワークロードではパフォーマンスが良くありませんでした。

2020年夏、AMD、Intel、Red Hat、SUSEは共同で、x86-64ベースラインの上に3つのx86-64マイクロアーキテクチャ・レベルを定義し、モダン・フラグを有効にした。詳細については、[x86-64-ABI仕様](#)を参照のこと。

---

## 注

CentOS 9のようないくつかの新しいディストリビューションは、*x86-64-v2*フラグを最低要件としてビルドされている。

---

- *kvm64 (x86-64-v1)*: Intel CPU >= Pentium 4、AMD CPU >= Phenomに対応。

- **x86-64-v2:** Intel CPU >= Nehalem, AMD CPU >= Opteron\_G3 に対応。x86-64-v1 と比較して CPU フラグを追加: +cx16, +lahf-lm, +popcnt, +pni, +sse4.1, +sse4.2, +ssse3.
- **x86-64-v2-AES:** Intel CPU >= Westmere, AMD CPU >= Opteron\_G4 に対応。x86-64-v2 と比較して CPU フラグを追加: +aes。
- **x86-64-v3:** Intel CPU >= Broadwell, AMD CPU >= EPYC に対応。と比較したCPUフラグを追加。  
**x86-64-v2-AES:** +avx、+avx2、+bmi1、+bmi2、+f16c、+fma、+movbe、+xsave。
- **x86-64-v4:** Intel CPU >= Skylake, AMD CPU >= EPYC v4 Genoa に対応。x86-64-v3 と比較してCPUフラグを追加: +avx512f, +avx512bw, +avx512cd, +avx512dq, +avx512vl.

## カスタムCPUタイプ

設定可能な機能を持つカスタムCPUタイプを指定することができる。これらは、コンフィギュレーション管理者は、/etc/pve/virtual-guest/cpu-models.conf ファイルを編集します。man cpu-models を参照してください。

フォーマットの詳細はこちら。

指定されたカスタムタイプは、/nodes の Sys.Audit 権限を持つユーザであれば誰でも選択できます。CLI または API を介して VM にカスタム CPU タイプを設定する場合は、名前の前に custom- を付ける必要があります。

## メルトダウン／スペクター関連CPUフラグ

MeltdownおよびSpectre脆弱性に関する CPU フラグがいくつかあります。<sup>3</sup> に関連するいくつかの CPU フラグがあり、選択した VM の CPU タイプがすでにデフォルトでこれらを有効にしている場合を除き、手動で設定する必要がある。

これらの CPU フラグを使用するためには、2つの条件を満たす必要がある：

- ホスト CPU がその機能をサポートし、ゲストの仮想 CPU にそれを伝搬させなければならない。
- ゲストオペレーティングシステムは、攻撃を軽減し、CPU 機能を利用できるバージョンにアップデートする必要があります。

それ以外の場合は、Web UI で CPU オプションを編集するか、VM コンフィギュレーション・ファイルで `cpu` オ

プロパティのflagsを設定して、仮想CPUの希望するCPUフラグを設定する必要がある。

Spectre v1,v2,v4の修正については、CPUまたはシステムベンダーも、いわゆる「マイクロコード・アップデート」を提供する必要があります。影響を受けるすべてのCPUがspec-ctrlをサポートするようにアップデートできるわけではないことに注意してください。

Proxmox VEホストが脆弱かどうかを確認するには、rootで以下のコマンドを実行する：

```
for f in /sys/devices/system/cpu/vulnerabilities/* ; do echo "${f##* /}.-" $(  
↳  
cat "$f"); done
```

ホストがまだ脆弱かどうかを検出するコミュニティ・スクリプトも用意されている。<sup>4</sup>

<sup>3</sup>メルトダウン攻撃 <https://meltdownattack.com/>

<sup>4</sup>spectre-meltdown-checker <https://meltdown.ovh/>

## インテル・プロセッサー

- ピシッド

これにより、ユーザー空間からカーネルメモリを効果的に隠すKPTI (*Kernel Page- Table Isolation*) と呼ばれるMeltdown (CVE-2017-5754) 緩和策のパフォーマンスへの影響が軽減される。PCIDがなければ、KPTIはかなり高価なメカニズムである。<sup>5</sup>

Proxmox VEホストがPCIDをサポートしているかどうかを確認するには、rootで以下のコマンドを実行します：

```
# grep ' pcid ' /proc/cpuinfo
```

これがemptyを返さない場合、ホストのCPUはpcidをサポートしている。

- スペックctrl

Spectre v1 (CVE-2017-5753) および Spectre v2 (CVE-2017-5715) の修正を有効にするために必要です。サフィックスが -IBRS のインテル CPU モデルにはデフォルトで含まれています。サフィックスが -IBRS でないインテル CPU モデルの場合は、明示的にオンにする必要があります。更新されたホストCPUマイクロコード (intel-microcode >= 20180425) が必要です。

- エスエスピーディー

Spectre V4 (CVE-2018-3639) 修正を有効にするために必要です。どのインテル CPU モデルにもデフォルトでは含まれていません。すべてのインテル CPU モデルで明示的にオンにする必要があります。更新されたホストCPUマイクロコード(intel-microcode >= 20180703)が必要です。

## AMDプロセッサー

- イブピブ

Spectre v1 (CVE-2017-5753) および Spectre v2 (CVE-2017-5715) の修正を有効にするために必要です。サフィックスが -IBPB の AMD CPU モデルにはデフォルトで含まれています。IBPBサフィックスがないAMD CPUモデルでは、明示的にオンにする必要があります。ゲスト CPU で使用する前に、ホスト CPU のマイクロコードがこの機能をサポートしている必要があります。

- ヴィルトエスピーディー

Spectre v4 (CVE-2018-3639) 修正を有効にするために必要です。どのAMD CPU モデルにもデフォルトでは含まれていません。すべての AMD CPU モデルで明示的にオンにする必要があります。ゲストの互換性を最大化するため、amd-ssbd も提供されている場合でも、ゲストに提供する必要があります。これは物理 CPU には存在しない仮想機能であるため、「ホスト」 CPU モデルを使用する場合は明示的に有効にする

必要があることに注意してください。

- *amd-ssbd*

Spectre v4 (CVE-2018-3639) 修正を有効にするために必要です。どのAMD CPUモデルにもデフォルトでは含まれていません。すべての AMD CPU モデルで明示的にオンにする必要があります。これは、virt-ssbd よりも高いパフォーマンスを提供するため、これをサポートするホストは、可能であれば常にこれをゲストに公開する必要があります。一部のカーネルは virt-ssbd についてしか知らないため、ゲストの互換性を最大化するために virt-ssbd も公開する必要があります。

- *amd-no-ssb*

ホストが Spectre V4 (CVE-2018-3639) に対して脆弱ではないことを示すために推奨されます。どの AMD CPU モデルにもデフォルトで含まれていません。将来のハードウェア世代の CPU は CVE-2018- 3639 に

---

に対して脆弱ではないため、*amd-no-ssb* を公開することで、その緩和策を有効にしないようにゲストに伝える必要があります。これは virt-ssbd と *amd-ssbd* とは相互に排他的です。

<sup>5</sup>PCIDは現在、x86のパフォーマンス／セキュリティ上重要な機能となっている

<https://groups.google.com/forum/m/#topic/mechanical-sympathy/L9mHTbeQLNU>

## NUMA

また、オプションで**NUMA**<sup>6</sup> アーキテクチャをエミュレートすることもできる。NUMAアーキテクチャの基本は、すべてのコアで利用可能なグローバル・メモリ・プールを持つ代わりに、メモリを各ソケットに近いローカル・バンクに分散させることを意味する。これにより、メモリバスがボトルネックにならなくなり、速度が向上します。システムがNUMAアーキテクチャの場合<sup>7</sup> これにより、ホスト・システム上でVMリソースを適切に分散できるようになるため、このオプションを有効にすることをお勧めします。このオプションは、VMのコアやRAMをホットプラグする場合にも必要です。

NUMAオプションを使用する場合は、ソケット数をホスト・システムのノード数に設定することを推奨する。

## vCPUホットプラグ

最近のオペレーティング・システムでは、実行中のシステムでCPUをホットプラグしたり、ある程度まではホットアンプラグしたりする機能が導入されている。仮想化によって、このようなシナリオで実際のハードウェアが引き起こす（物理的な）問題の多くを回避することができる。それでも、これはかなり新しく複雑な機能なので、その使用は絶対に必要な場合に限定すべきである。ほとんどの機能は、他の、よくテストされた、より複雑でない機能で再現することができます。

Proxmox VEでは、プラグインされたCPUの最大数は常にコア\* ソケットです。このコア数未満のCPUでVMを起動するには、**vcpus**設定を使用することができます。

現在、この機能はLinuxでのみサポートされており、3.10より新しいカーネルが必要で、4.7より新しいカーネルを推奨する。

以下のように udev ルールを使って、新しい CPU をゲストのオンラインに自動的に設定することができます：

```
SUBSYSTEM=="cpu", ACTION=="add", TEST=="online", ATTR{online}=="0", ATTR{online}="1"
```

これを/etc/udev/rules.d/の下に、.rulesで終わるファイルとして保存する。

注意: CPU ホットリムーブはマシン依存であり、ゲストの協力が必要です。削除コマンドはCPUの削除が実際に行われることを保証するものではなく、通常はx86/amd64のACPIなど、ターゲットに依存するメカニズムを使ってゲストOSに転送されるリクエストです。

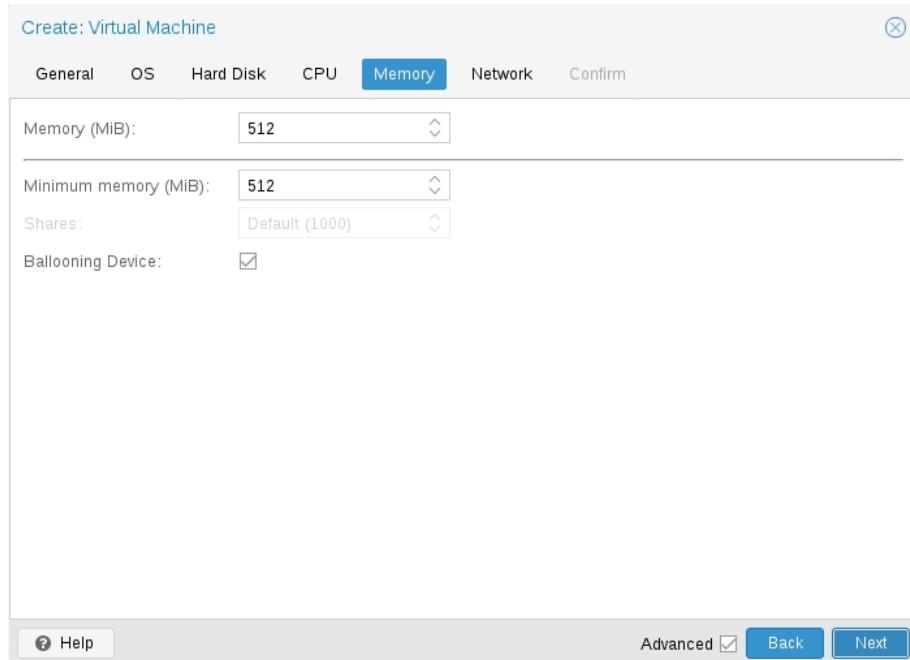
## 10.2.6 メモリー

各VMには、固定サイズのメモリを設定するか、ホストの現在のRAM使用量に基づいて動的にメモリを割り当てるようにProxmox VEに依頼するオプションがあります。

<sup>6</sup>[https://en.wikipedia.org/wiki/Non-uniform\\_memory\\_access](https://en.wikipedia.org/wiki/Non-uniform_memory_access)

<sup>7</sup>コマンド numactl --hardware | grep available が複数のノードを返す場合、ホスト・システムは NUMAアーキテクチャです。

## 固定メモリ割り当て



メモリと最小メモリを同じ量に設定すると、Proxmox VEは単純に指定した量をVMに割り当てます。

固定メモリ・サイズを使用している場合でも、バルーン・デバイスはVMに追加されます。これは、ゲストが実際に使用するメモリ量などの有用な情報を提供するためです。一般的には、**バルーンは有効のままにしておく必要がありますが、（デバッグ目的などで）無効にしたい場合は、単に Ballooning Device のチェックを外すか、または**

バルーン: 0

を設定する。

## 自動メモリ割り当て

最小メモリをメモリよりも低く設定すると、Proxmox VEは指定した最小量が常にVMで使用可能であることを確認し、ホスト上のRAM使用率が80%未満の場合は、指定した最大メモリまで動的にゲストにメモリを追加します。

ホストのRAMが残り少なくなると、VMはホストにメモリを解放し、必要に応じて実行中のプロセスをスワップし、最後の手段としてoomキラーを起動する。ホストとゲスト間のメモリの受け渡しは、ゲスト内部で動作する特別なバルーンカーネルドライバを介して行われ、ホストからメモリページを取得または解放します。<sup>8</sup>

複数のVMが自動割り当て機能を使用する場合、各VMが占有すべき空きホスト・メモリの相対量を示すシェア係数を設定することができる。例えば、4つのVMがあり、そのうち3つはHTTPサーバー、最後の1つはデ

ータベース・サーバーであるとします。より多くのデータベース・ブロックをデータベース・サーバーの RAM にキャッシュするため、予備の RAM が利用可能な場合、データベース VM を優先したいとします。このため、データベース VM に 3000 の Shares プロパティを割り当て、他の VM はデフォルト設定の 1000 にします。

ホスト・サーバには 32GB の RAM があり、現在 16GB を使用しています。

= 設定された最小メモリ量に加え、9GB の RAM が VM に割り当てられる。データベース

---

<sup>8</sup>バルーンドライバーの内部構造については、<https://rwmj.wordpress.com/2010/07/17/- virtio-balloon/> に詳しい説明がある。

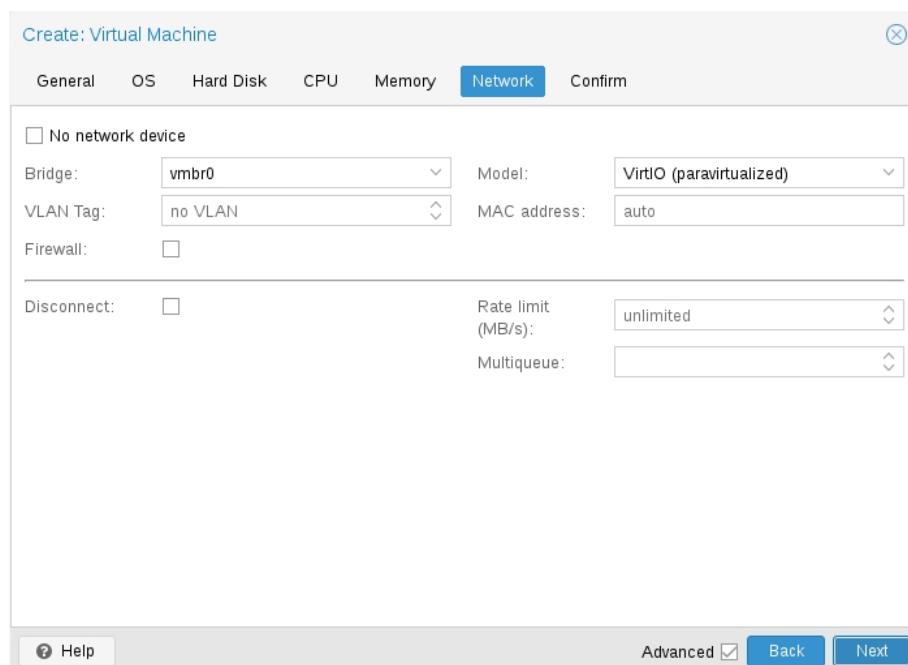
VMは $9 * 3000 / (3000)$ から利益を得る。

$1000 + 1000 + 1000 = 4.5$  GBの追加RAMと、各HTTPサーバーの1.5 GB。

2010年以降にリリースされたすべてのLinuxディストリビューションには、バルーン・カーネル・ドライバーが含まれています。Windows OSの場合、バルーンドライバを手動で追加する必要があり、ゲストの速度低下を引き起こす可能性があるため、重要なシステムでの使用はお勧めしません。

VMにRAMを割り当てる場合、経験則では常に1GBのRAMをホストに残しておくのがよい。

### 10.2.7 ネットワーク機器



各VMは、4つの異なるタイプの多数のネットワーク・インターフェイス・コントローラ (NIC) を持つことができる：

- インテルE1000がデフォルトで、インテルギガビットネットワークカードをエミュレートします。
- パフォーマンスを最大化するには、VirtIO 準仮想化 NIC を使用する必要があります。すべての VirtIO デバイスと同様に、ゲスト OS には適切なドライバがインストールされている必要があります。
- Realtek 8139は旧式の100MB/sネットワークカードをエミュレートしているため、古いオペレーティングシステム（2002年以前にリリースされたもの）をエミュレートする場合にのみ使用してください。
- vmxnet3も準仮想化デバイスで、他のハイパーバイザーからVMをインポートする場合にのみ使用する。

Proxmox VEは、各NICにランダムなMACアドレスを生成し、VMがイーサネット上でアドレス

指定できるようにします。

VMに追加したNICは、2つの異なるモデルのいずれかに従うことができる：

- デフォルトの**ブリッジモード**では、各仮想NICはタップデバイス(イーサネットNICをシミュレートするソフトウェアループバックデバイス)によってホスト上でバックアップされます。このタップデバイスは、Proxmox VEのデフォルトではvmbr0というブリッジに追加されます。このモードでは、VMはホストがあるイーサネットLANに直接アクセスできます。

- 代替NATモードでは、各仮想NICはQEMUユーザーネットワーキングスタックとのみ通信し、内蔵ルーターとDHCPサーバーがネットワークアクセスを提供します。この内蔵DHCPは、プライベート10.0.2.0/24範囲のアドレスを提供します。NATモードはブリッジモードよりはるかに遅いので、テストにのみ使うべきである。このモードはCLIまたはAPI経由でのみ利用可能で、ウェブUI経由では利用できません。

また、**ネットワーク・デバイスなし**を選択することで、VMの作成時にネットワーク・デバイスの追加を省略することもできます。

各VMネットワーク・デバイスの**MTU**設定を上書きすることができます。`mtu=1`オプションは、MTU値が基礎となるブリッジから継承される特殊なケースを表します。このオプションは、**VirtIO**ネットワークデバイスでのみ使用できます。

## マルチキュー

VirtIOドライバを使用している場合、オプションで**Multiqueue**オプションを有効にできます。このオプションにより、ゲストOSは複数の仮想CPUを使用してネットワーキング・パケットを処理できるようになり、転送されるパケットの総数が増加します。

Proxmox VEでVirtIOドライバを使用する場合、各NICネットワーク・キューはホスト・カーネルに渡され、キューはvhostドライバによって生成されたカーネル・スレッドによって処理されます。このオプションを有効にすると、各NICに対して複数のネットワーク・キューをホスト・カーネルに渡すことができます。

**Multiqueue**を使用する場合は、ゲストのvCPU数と同じ値に設定することをお勧めします。vCPU数は、ソケット数×VMに設定されたコア数であることを忘れないでください。また、このethtoolコマンドを使用して、VMの各VirtIO NICの多目的チャネル数を設定する必要があります：

```
ethtool -L ens1 結合 X
```

ここで、XはVMのvCPU数である。

WindowsゲストをMultiqueue用に設定するには、**Redhat VirtIO Ethernet Adapter**ドライバをインストールし、以下のようにNICの設定を変更します。デバイスマネージャーを開き、"Network adapters"の下でNICを右クリックし、"Properties"を選択します。次に「詳細設定」タブを開き、左側のリストから「受信側スケーリング」を選択します。有効に設定されていることを確認してください。次に、リストの「Maximum number of RSS Queues」に移動し、VMのvCPU数に設定する。設定が正しいことを確認したら、「OK」をクリックして確定する。

Multiqueue パラメータを 1 より大きな値に設定すると、トラフィックが増加するにつれてホストとゲスト・システムの CPU 負荷が増加することに注意してください。このオプションは、VMがルーターやリバースプロキシとして動作している場合や、長時間のポーリングを行うビジーなHTTPサーバーなど、VMが多数の着信接続を処理しなければならない場合にのみ設定することをお勧めします。

## 10.2.8 ディスプレイ

QEMUは、数種類のVGAハードウェアを仮想化できます。いくつか例を挙げます：

- デフォルトのstdは、Bochs VBE拡張を持つカードをエミュレートする。
- cirrus**、これはかつてデフォルトだったので、すべての問題を抱えた非常に古いハードウェアモジュールをエミュレートしている。このディスプレイタイプは、本当に必要な場合のみ使用してください。<sup>9</sup>例えれば、Windows XP以前を使用している場合など。
- vmwareは、VMWare SVGA-II互換アダプターです。

<sup>9</sup><https://www.kraxel.org/blog/2014/10/qemu-using-cirrus-considered-harmful/> qemu: Cirrusの使用は有害と考えられる

- qxlはQXL準仮想化グラフィックカードです。これを選択すると、VMのSPICE（リモート・ビューワ・プロトコル）も有効になります。
- **Virtio-g1**（しばしばVirGLと呼ばれる）は、VM内で使用するための仮想3D GPUであり、特別な（高価な）モデルやドライバを必要とせず、ホストGPUを完全にバインドすることなく、ワークロードをホストGPUにオフロードすることができます。

---

### 注

VirGLのサポートにはいくつかの追加ライブラリが必要ですが、比較的大きく、またすべてのGPUモデル/ベンダーでオープンソースとして利用できるわけではないため、デフォルトではインストールされません。ほとんどのセットアップでは、次のようにするだけです: `apt install libgl1 libegl1`

---

メモリオプションを設定することで、仮想GPUに与えるメモリ量を編集することができます。これにより、特にSPICE/QXLでは、VM内部でより高い解像度が可能になります。

メモリは表示デバイスによって予約されるため、SPICE のマルチモニタ・モード（デュアルモニタの場合はqxl2 など）を選択することには意味があります：

- Windowsは各モニタにデバイスを必要とするため、ostypeがWindowsの場合、Proxmox VEはモニタごとに追加のデバイスをVMに与えます。各デバイスは、指定された量のメモリを取得します。
- Linux VMでは、常に多くの仮想モニターを有効にすることができますが、マルチモニタモードを選択すると、モニター数に応じてデバイスに与えられるメモリーが倍増します。

ディスプレイタイプとしてserialXを選択すると、VGA出力が無効になり、Webコンソールは選択したシリアルポートにリダイレクトされます。この場合、ディスプレイメモリの設定は無視されます。

### VNCクリップボード

クリップボードをvncに設定することで、VNCクリップボードを有効にすることができます。

```
# qm set <vmid> -vga <displaytype>,clipboard=vnc
```

クリップボード機能を使用するには、まずSPICEゲストツールをインストールする必要があります。Debianベースのディストリビューションでは、spice-vdagent をインストールすることで実現できます。他のオペレーティングシステムでは、公式リポジトリで検索するか、<https://www.spice-vm.org/>

---

[space.org/download.html](http://space.org/download.html) を参照してください。

spiceゲストツールをインストールしたら、VNCクリップボード機能（noVNCコンソールパネルなど）を使うことができます。ただし、SPICE、virtio、virglを使用している場合は、使用するクリップボードを選択する必要があります。これは、`clipboard`が`vnc`に設定されている場合、デフォルトの**SPICE**クリップボードが**VNC**クリップボードに置き換えられるためです。

### 10.2.9 USBバススルー

USBバススルー・デバイスには2種類ある：

- ホストUSBバススルー
- SPICE USBバススルー

ホストUSBバススルーは、VMにホストのUSBデバイスを与えることで機能する。これは、ベンダーIDやプロダクトIDを経由するか、ホスト・バスやポートを経由するかのどちらかだ。

vendor/product-idは次のようになる: **0123:abcd**、ここで0123はベンダーのid、abcdは製品のidであり、同じusbデバイスの2つのピースが同じidを持つことを意味する。

バス/ポートは次のようになる: **1-2.3.4**で、1がバス、**2.3.4**がポートパスです。これはホストの物理ポートを表します（usbコントローラーの内部順序に依存します）。

VM起動時にVMコンフィギュレーションにデバイスが存在しても、そのデバイスがホストに存在しない場合、VMは問題なく起動できる。デバイス/ポートがホストで利用可能になると、すぐにそのデバイスが通過する。



### 警告

このようなUSBバススルーを使用することは、VMを別のホストにオンライン移動できないことを意味する。

バススルーの二番目のタイプは、SPICE USB バススルーです。VMに1つ以上のSPICE USB ポートを追加すると、SPICE クライアントから VM にローカルの USB デバイスを動的に渡すことができます。これは、入力デバイスやハードウェアドングルを一時的にリダイレクトするのに便利です。

クラスタレベルでデバイスをマッピングすることも可能で、HAで適切に使用でき、ハードウェアの変更が検出され、ルートユーザー以外でも設定できるようになる。詳細は[リソースマッピング](#)を参照してください。

## 10.2.10 BIOSとUEFI

コンピュータを適切にエミュレートするために、QEMUはファームウェアを使用する必要がある。一般的なPCではBIOSまたは(U)EFIとして知られるファームウェアは、VMの起動時に最初のステップとして実行される。BIOSは、基本的なハードウェアの初期化を行い、オペレーティング・システムにファームウェアとハードウェアへのインターフェースを提供する役割を担っている。デフォルトでQEMUは、オープンソースのx86 BIOS実装である[SeaBIOS](#)を使用します。SeaBIOSは、ほとんどの標準的なセットアップに適しています。

オペレーティングシステムによっては(Windows 11など)、UEFI互換の実装を使う必要があるかもしれません。そのような場合、代わりにオープンソースのUEFI実装である[OVMF](#)を使わなければなりません。<sup>10</sup>

例えば、VGAパススルーを行いたい場合など、SeaBIOSがブートするのに理想的なファームウェアではないシナリオもある。<sup>11</sup>

OVMFを使いたい場合、考慮すべき点がいくつかある：

ブート順などを保存するためには、EFIディスクが必要です。このディスクはバックアップやスナップショットに含まれ、1つしか存在できません。

このようなディスクは以下のコマンドで作成できる：

```
# qm set <vmid> -efidisk0 <storage>:1,format=<format>,efitype=4m,pre-'  
enrolled-keys=1
```

ここで、<storage>はディスクを置きたいストレージ、<format>はストレージがサポートするフォーマットである。あるいは、ウェブ・インターフェイスからVMのハードウェア・セクションにある*Add → EFI Disk*でこのようなディスクを作成することもできます。

<sup>10</sup>OVMFプロジェクト <https://github.com/tianocore/tianocore.github.io/wiki/OVMF> を参照。

<sup>11</sup>アレックス・ウィリアムソンがこの件に関して良いブログエントリーを書いている。  
<https://vfio.blogspot.co.at/2014/08/primary-graphics-assignment-without-vga.html>

**efitype** オプションは、使用する OVMF ファームウェアのバージョンを指定する。新しい VM の場合、これは常に *4m* であるべきである。これは、セキュア・ブートをサポートしており、将来の開発をサポートするために割り当てられる容量が多いからである（GUI ではこれがデフォルト）。

**pre-enroll-keys** は、efidisk にディストリビューション固有のセキュア・ブート・キーと Microsoft Standard セキュア・ブート・キーをプリロードするかどうかを指定します。また、デフォルトでセキュアブートを有効にします（ただし、VM 内の OVMF メニューで無効にすることもできます）。

---

### 注

既存の VM (*2m* の efidisk をまだ使用している) でセキュアブートの使用を開始したい場合は、efidisk を再作成する必要があります。そのためには、古いものを削除し (`qm set <vmid> -delete efidisk0`)、上記の説明に従って新しいものを追加します。これにより、OVMFメニューで行ったカスタム設定がリセットされる！

---

仮想ディスプレイ（VGAパススルーなし）でOVMFを使用する場合、OVMFメニューでクライアントの解像度を設定するか（ブート中にESCボタンを押すことで到達できます）、ディスプレイタイプとしてSPICEを選択する必要があります。

## 10.2.11 トラステッド・プラットフォーム・モジュール (TPM)

トラステッド・プラットフォーム・モジュールは、暗号化キーなどの秘密データを安全に保存し、システム起動を検証するための耐タンパー機能を提供するデバイスである。

特定のオペレーティング・システム（Windows11など）では、このようなデバイスをマシン（物理的であれ仮想的であれ）に接続する必要がある。

TPMは、**tpmstate**ボリュームを指定して追加する。これはefidiskに似た働きをするが、一度作成すると変更できない

```
# qm set <vmid> -tpmstate0 <storage>:1,version=<version>  
ない（削除のみ）。以下のコマンドで追加できる：
```

ここで、**<storage>**は状態を置きたいストレージで、**<version>**はv1.2かv2.0のどちらかだ。VMのハードウェア・セクションでAdd → TPM Stateを選択して、ウェブ・インターフェイスから追加することもできる。v2.0 TPM仕様の方が新しく、サポートも充実しているため、v1.2 TPMを必要とする特別な実装がない限り、そちらを優先すべきである。

---

## 注

物理的なTPMに比べ、エミュレートされたTPMは、セキュリティ上何のメリットもない。TPMのポイントは、TPM仕様の一部として指定されたコマンドを除いて、その上のデータを簡単に変更できることです。エミュレートされたデバイスの場合、データの保存は通常のボリューム上で行われるため、それにアクセスできる人なら誰でも編集できる可能性がある。

### 10.2.12 VM間共有メモリ

VM間共有メモリーデバイス (`ivshmem`) を追加することで、ホストとゲスト間、または複数のゲスト間でメモリーを共有することができます。

```
# qm set <vmid> -ivshmem size=32,name=foo
```

このようなデバイスを追加するには、`qm`:

サイズの単位はMiBです。ファイルは`/dev/shm/pve-shm-$name`の下に置かれます（デフォルトの名前は`vmid`です）。

### 注

現在、デバイスは、それを使用しているVMがシャットダウンされたり停止されたりすると、すぐに削除される。オープン・コネクションは維持されるが、全く同じデバイスへの新規接続はできなくなる。

このようなデバイスのユースケースとして、Looking Glassプロジェクトがある。[12](#) プロジェクトで、ホストとゲスト間で高性能、低遅延のディスプレイミラーリングを可能にする。

## 10.2.13 オーディオ機器

オーディオデバイスを追加するには、以下のコマンドを実行します：

```
qm set <vmid> -audio0 device=<デバイス>.
```

対応オーディオデバイスは以下の通り：

- `ich9-intel-hda`: Intel HD オーディオコントローラ、ICH9 をエミュレート
- `intel-hda`: インテル HD オーディオコントローラ、ICH6 をエミュレート
- `AC97`: Audio Codec '97。Windows XPのような古いOSに便利：
- スパイク
- なし

`spice`バックエンドは[SPICE](#)と組み合わせて使用することができ、`none`バックエンドはVM内のオーディオデバイスがソフトウェアの動作に必要な場合に便利です。ホストの物理オーディオ・デバイスを使用するには、デバイス・バススルーを使用する（[PCIバススルー](#)と[USBバススルー](#)を参照）。MicrosoftのRDPのようなリモート・プロトコルには、サウンドを再生するオプションがある。

## 10.2.14 VirtIO RNG

RNG (Random Number Generator) は、システムにエントロピー（ランダム性）を提供するデバイスである

。仮想ハードウェア RNG を使用して、ホストシステムからゲスト VM にエントロピーを提供することができます。これは、特にゲストのブートプロセス中に、ゲストにおけるエントロピーの飢餓問題（十分なエントロピーが利用できず、システムがスローダウンしたり問題が発生したりする状況）を回避するために役立ちます。

VirtIOベースのエミュレートされたRNGを追加するには、次のコマンドを実行します：

```
qm set <vmid> -rng0 source=<ソース>[,max_bytes=X,period=Y]。
```

sourceはエントロピーがホスト上のどこから読み込まれるかを指定し、以下のいずれかでなければならない：

<sup>12</sup> レッキンググラス：<https://looking-glass.io/>

- /dev/urandom: ノンブロッキングカーネルエントロピープール (推奨)
- /dev/random: カーネルプールをブロックする (推奨されません。ホストシステムでエントロピーが枯渇する可能性があります)
- /dev/hwrng: ホストに接続されたハードウェア RNG を通過させる (複数利用可能な場合は、/sys/devices/virtual/misc/hw\_random/rng\_current で選択されたものが使われる)

制限は、max\_bytesとperiodパラメータで指定することができ、これらはミリ秒単位で1周期あたりのmax\_bytesとして読み込まれる。しかし、これは線形関係を表すものではない：1024B/1000msは、1秒間に1KiBがゲストにストリーミングされるのではなく、1秒間のタイマーで最大1KiBのデータが利用可能になることを意味します。このように、周期を短くすることで、より速い速度でゲストにエントロピーを注入することができます。

デフォルトでは、リミットは1000msあたり1024バイト（1KiB/s）に設定されています。ゲストがホストのリソースを使いすぎないように、常にリミッターを使用することを推奨します。必要であれば、max\_bytesに0を指定することで、すべての制限を無効にすることができます。

### 10.2.15 デバイスの起動順序

QEMU は、どのデバイスからどの順序でブートすべきかをゲストに指示できます。これは、例えば boot

```
boot: order=scsi0;net0;hostpci0
```



プロパティ経由で config で指定できます：

この方法では、ゲストはまずディスク scsi0 からのブートを試み、それが失敗した場合は net0 からのネットワークブートを試み、それも失敗した場合は最後に通過した PCIe デバイス（NVMe の場合はディスクとみなされ、そうでない場合はオプション ROM への起動を試みます）からのブートを試みます。

GUI上では、ドラッグ & ドロップ・エディターを使ってブート順序を指定したり、チェックボックスを使っ

て特定のデバイスのブートを有効または無効にしたりすることができます。

---

## 注

ゲストが OS を起動したりブートローダをロードするために複数のディスクを使用する場合、ゲストが起動できるようにするためには、それら全てがブータブルとしてマークされていなければなりません（つまり、チェックボックスが有効になっているか、コンフィグ内のリストに表示されていなければなりません）。これは、最近の SeaBIOS と OVMF のバージョンは、ディスクがブータブルとマークされている場合にのみディスクを初期化するためです。

---

いずれにせよ、リストに表示されないデバイスやチェックマークが無効になっているデバイスでも、ゲストのオペレーティングシステムが起動して初期化されれば、利用可能になります。ブータブルフラグはゲスト BIOS とブートローダーにのみ影響します。

### 10.2.16 仮想マシンの自動起動とシャットダウン

VMを作成した後、ホスト・システムのブート時にVMを自動的に起動させたいと思うことでしょう。そのためには、ウェブ・インターフェイスのVMのオプション・タブから*Start at boot*オプションを選択するか、以下のコマンドで設定する必要がある：

```
# qm set <vmid> -onboot 1
```

めには、ウェブ・インターフェイスのVMのオプション・タブから*Start at boot*オプションを選択するか、以下のコマンドで設定する必要がある：

#### スタートとシャットダウンの順序



例えば、あるVMが他のゲスト・システムにファイアウォールやDHCPを提供している場合など、VMの起動順序を微調整したい場合があります。この場合、以下のパラメータを使用できます：

- スタート/シャットダウン順：**開始順序の優先順位を定義します。例えば、VMを最初に起動させたい場合は1に設定する。(シャットダウンには逆の起動順序を使用するため、起動順序が1のマシンは最後にシャットダウンされます)。ホスト上で複数のVMが同じ順序で定義されている場合、それらはさらにVMIDの昇順で並べられます。
- 起動遅延：**このVMが起動してから後続のVMが起動するまでの間隔を定義します。例えば、240秒待ってから他のVMを起動する場合は240に設定します。
- シャットダウンタイムアウト：**Proxmox VEがシャットダウンコマンドを発行した後、VMがオフラインになるまで待機する時間を秒単位で定義します。デフォルトでは、この値は180に設定されています。これは、Proxmox VEがシャットダウン要求を発行し、マシンがオフラインになるまで180秒待つことを意味します。タイムアウト後もマシンがオンラインの場合、強制的に停止されます。

---

#### 注

HAスタックによって管理されるVMは、現在のところ起動時開始と起動順序のオプションには従わない。これらのVMは、HAマネージャ自身がVMの起動と停止を確実に行うため、起動とシャットダウンのアルゴ

リズムによってスキップされる。

---

開始/シャットダウン順序パラメータが設定されていないマシンは、常にパラメータが設定されているマシンの後に開始することに注意してください。さらに、このパラメータは同じホスト上で実行されている仮想マシン間でのみ適用され、クラスタ全体では適用されません。

ホストのブートと最初のVMのブートの間に遅延が必要な場合は、[Proxmox VEノード管理のセクション](#)を参照してください。

## 10.2.17 QEMUゲストエージェント

QEMU ゲストエージェントは VM 内で実行されるサービスで、ホストとゲスト間の通信チャネルを提供します。情報を交換するために使用され、ホストがゲストにコマンドを発行できるようにします。

例えば、VM サマリーパネルの IP アドレスはゲストエージェント経由で取得されます。

あるいは、バックアップを開始する際、ゲストはゲストエージェントを介して、`fs-freeze`を介して未処理の書き込みを同期するように指示されます。

と`fs-thaw`コマンド。

ゲストエージェントが正しく機能するためには、以下の手順を踏む必要があります：

- ゲストにエージェントをインストールし、実行されていることを確認する。
- Proxmox VEでエージェント経由の通信を有効にする。

### ゲストエージェントのインストール

ほとんどのLinuxディストリビューションでは、ゲストエージェントが利用できる。パッケージ名は通常`qemu-guest-agent`です。Windows では、[Fedora VirtIO ドライバ ISO](#) からインストールできます。

### ゲストエージェントとの通信を有効にする

Proxmox VEからゲストエージェントへの通信は、VMのオプションパネルで有効にできます。変更を有効にするには、VMの再スタートが必要です。

### QGAによる自動TRIM

*Run guest-trim* オプションを有効にすることができます。これを有効にすると、Proxmox VE はストレージにゼロを書き出す可能性のある以下の操作の後に、ゲストにトリムコマンドを発行します：

- ディスクを別のストレージに移動する
- ローカルストレージを持つ別のノードへのVMのライブマイグレーション

シン・プロビジョニングされたストレージでは、これは未使用領域を解放するのに役立つ。

## 注

Linuxのext4では、重複したTRIMリクエストの発行を避けるためにメモリ内最適化を使用するため、注意点があります。ゲストは基礎となるストレージの変更について知らないので、最初のゲストトリムだけが期待通りに実行されます。次のリブートまで、それ以降のTRIMは、それ以降に変更されたファイルシステムの一部のみを考慮します。

---

## バックアップ時のファイルシステムのフリーズと解凍

デフォルトでは、ゲストファイルシステムは、一貫性を提供するために、バックアップが実行されるときに `fs-freeze` QEMUゲストエージェントコマンドを介して同期されます。

Windowsゲストでは、アプリケーションによってはWindows VSS（Volume Shadow Copy Service）レイヤーをフックして一貫性のあるバックアップを処理するものもあり、`fs-freeze`はそれを妨害する可能性があります。たとえば、一部のSQL Serverで`fs-freeze`を呼び出すと、VSSがSQL Writer VSSモジュールを呼び出す引き金となり、差分バックアップのSQL Serverバックアップチェーンが切断されることが確認されています。

このようなセットアップの場合、QGAオプションの`freeze-fs-on-backup`を0に設定することで、バックアップ時にフリーズと解凍のサイクルを発行しないようにProxmox VEを設定することができます。



### 重要

このオプションを無効にすると、一貫性のないファイルシステムでのバックアップにつながる可能性があるため、自分が何をしているかわかっている場合のみ無効にしてください。

## トラブルシューティング

### VMがシャットダウンしない

ゲストエージェントがインストールされ、実行されていることを確認してください。

ゲストエージェントが有効になると、Proxmox VEはゲストエージェント経由でシャットダウンなどの電源コマンドを送信します。ゲストエージェントが実行されていない場合、コマンドは正しく実行できず、シャットダウンコマンドはタイムアウトになります。

### 10.2.18 SPICEの強化

SPICE Enhancementsは、リモート・ビューワの操作性を向上させるオプション機能です。

GUIで有効にするには、仮想マシンの【オプション】パネルを開きます。CLIで有効にするには、次のコマン

```
qm set <vmid> -spice_enhancements folderssharing=1,videostreaming=すべて
```

ドを実行します：

## 注

これらの機能を使用するには、仮想マシンのDisplayをSPICE (qxl) に設定する必要があります。

---

## フォルダ共有

ローカルフォルダをゲストと共有します。spice-webdavd デーモンをゲストにインストールする必要があります。これは、<http://localhost:9843> にあるローカル WebDAV サーバーを通じて共有フォルダーを利用可能にします。

Windowsゲストの場合、Spice WebDAVデーモンのインストーラはSPICE公式ウェブサイトからダウンロードできます。

ほとんどのLinuxディストリビューションには、インストール可能なspice-webdavdというパッケージがある。

Virt-Viewer（リモートビューワー）でフォルダを共有するには、[ファイル]→[環境設定]に進みます。共有するフォルダを選択し、チェックボックスを有効にします。

---

## 注

フォルダの共有は、現在Linux版のVirt-Viewerでのみ機能します。

---



### 注意

実験的です！現在、この機能は安定して動作していません。

---

## ビデオストリーミング

高速リフレッシュエリアはビデオストリームにエンコードされる。2つのオプションがある：

- **すべて**：高速リフレッシュ領域はすべてビデオストリームにエンコードされる。
- **フィルターを使用します**：ビデオストリーミングを使用するかどうかを決定するために、追加のフィルターが使用されます（現在、小さなウィンドウ面のみがスキップされます）。

ビデオストリーミングを有効にすべきかどうか、またどのオプションを選ぶべきかについては、一般的な推奨はできません。具体的な状況によって異なります。

## トラブルシューティング

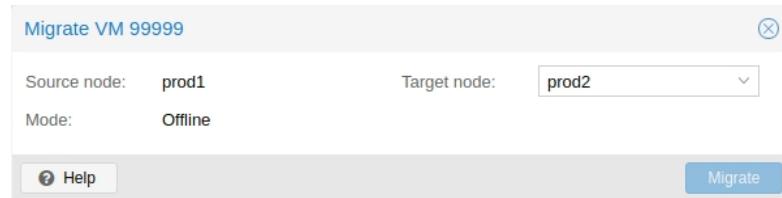
### 共有フォルダが表示されない

WebDAV サービスが有効になっていて、ゲストで実行されていることを確認してください。Windows では *Spice webdav proxy* と呼ばれます。Linux では *spice-webdavd* という名前ですが、ディストリビューションによって異なる場合があります。

サービスが実行されている場合は、ゲストでブラウザで <http://localhost:9843> を開いて WebDAV サーバを確認してください。SPICE セッションを再起動するのに役立ちます。

---

## 10.3 移住



クラスタがある場合、VMを別のホストに移行するには

```
# qm migrate <vmid> <target>
```

これには一般的に2つのメカニズムがある。

- オンラインマイグレーション（別名ライブマイグレーション）
- オフライン移行

### 10.3.1 オンライン移行

VMが稼動しており、ローカルにバインドされたリソース（パススルーされるデバイスなど）が設定されていない場合は、`qm migration`コマンドの`--online`フラグを使用してライブマイグレーションを開始できます。VMが稼動している場合、Webインターフェイスのデフォルトはライブマイグレーションになります。

#### 仕組み

オンラインマイグレーションはまず、ターゲットホスト上で着信フラグを持つ新しいQEMUプロセスを起動し、ゲストvCPUを一時停止したまま基本的な初期化のみを実行し、ソース仮想マシンのゲストメモリとデバイス状態のデータストリームを待ちます。ディスクなどの他のリソースはすべて、VMの実行時状態移行が始まる前に共有されるか、すでに送信されているため、転送が必要なのはメモリコンテンツとデバイス状態のみです。

この接続が確立されると、ソースは非同期でメモリコンテンツをターゲットに送信し始める。ソース上のゲスト・メモリーが変更されると、そのセクションはダーティとマークされ、ゲスト・メモリーのデータを送信するために別のパスが行われます。このループは、実行中のソースVMと受信中のターゲットVMのデータ差が数ミリ秒で送信できるほど小さくなるまで繰り返されます。その後、ソースVMはユーザーやプログラムに気づかれることなく完全に一時停止し、残りのデータをターゲットに送信できるようになります。

#### 必要条件

ライブ・マイグレーションが機能するためには、必要なことがいくつかある：

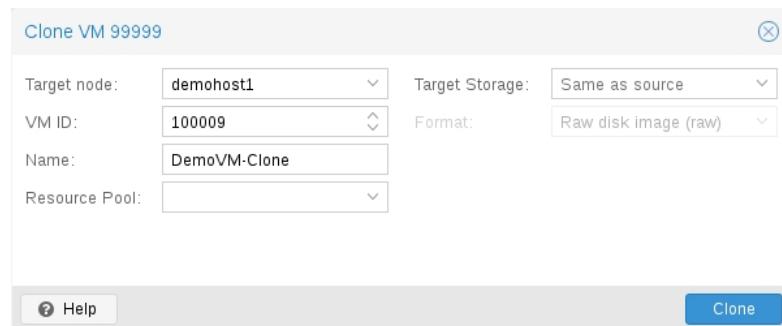
- VMには、マイグレーションできないローカルリソースがない。例えば、PCIデバイスやUSBデバイスは、現在ライブマイグレーションをブロックしている。一方、ローカルディスクは、ターゲットに送信することで問題なく移行できる。
- ホストは同じProxmox VEクラスタにあります。
- ホスト同士は正常に（信頼できる）ネットワーク接続されている。
- ターゲットホストには、Proxmox VEパッケージと同じかそれ以上のバージョンが必要です。逆に動作する場合もありますが、保証はできません。

- ホストは同じベンダーの同じようなCPUを使用しています。異なるベンダーであっても、設定されている実際のモデルとVMのCPUタイプによっては動作するかもしれません、保証はできません。

### 10.3.2 オフライン移行

ローカルリソースがある場合、すべてのディスクが両方のホストで定義されたストレージ上にある限り、VMをオフラインで移行することができます。マイグレーションは、オンラインマイグレーションと同様に、ネットワーク経由でターゲットホストにディスクをコピーします。ハードウェアパススルーの設定は、ターゲットホスト上のデバイスの場所に合わせる必要があるかもしれませんことに注意してください。

## 10.4 コピーとクローン



VMのインストールは通常、OSベンダーのインストール・メディア（CD-ROM）を使って行われる。OSにもよるが、これは避けたいほど時間のかかる作業だ。

同じタイプのVMを多数デプロイする簡単な方法は、既存のVMをコピーすることだ。このようなコピーにはクローンという用語を使い、リンククローンとフルクローンを区別する。

### フルクローン

このようなコピーの結果は、独立したVMとなる。新しいVMは元のVMとストレージ・リソースを共有しない。

ターゲット・ストレージを選択することができるので、これを使用してVMを全く別のストレージに移行することができる。ストレージドライバが複数のフォーマットをサポートしている場合、ディスクイメージのフォーマットを変更することもできる。

---

### 注

完全クローンは、すべてのVMイメージ・データを読み込んでコピーする必要がある。これは通常、リンクされたクローンを作成するよりもはるかに遅い。

---

ストレージの種類によっては、特定のスナップショットをコピーすることができる。これはまた、最終的なコピーに元のVMからの追加スナップショットが含まれないことを意味する。

### リンクド・クローン

最近のストレージ・ドライバは、リンクされたクローンを高速に生成する方法をサポートしている。このようなクローンは書き込み可能なコピーであり、初期内容は元のデータと同じである。リンクされたクローンの作成はほぼ瞬時に行え、最初は追加のスペースを消費しない。

---

新しい画像はまだ元の画像を参照しているため、リンクされていると呼ばれる。変更されていないデータ・ブロックは元の画像から読み込まれますが、変更は新しい場所から書き込まれます（その後、読み込まれます）。この手法はコピー・オン・ライトと呼ばれる。

これには、元のボリュームが読み取り専用であることが必要です。Proxmox VEを使用すると、任意のVMを読み取り専用の[テンプレートに](#)変換することができます）。このようなテンプレートは、後でリンクされたクローンを効率的に作成するために使用することができます。

---

## 注

リンクされたクローンが存在する間は、元のテンプレートを削除することはできません。

---

これはストレージ内部の機能であるため、リンクされたクローンのターゲットストレージを変更することはできません。

ターゲット・ノード・オプションを使用すると、新しいVMを別のノード上に作成できます。唯一の制限は、VMが共有ストレージ上にあり、そのストレージがターゲット・ノードでも利用可能であることです。

リソースの競合を避けるために、すべてのネットワーク・インターフェイスのMACアドレスはランダム化され、新しい

VM BIOS (smbios1) 設定の *UUID*。

## 10.5 仮想マシンテンプレート

VMをテンプレートに変換することができる。このようなテンプレートは読み取り専用で、リンクされたクローンを作成するために使用できる。

---

### 注

ディスクイメージを変更することになるため、テンプレートを起動することはできない。テンプレートを変更したい場合は、リンクされたクローンを作成し、それを変更してください。

---

## 10.6 VMジェネレーションID

Proxmox VEは仮想マシンのジェネレーションID (*vmgenid*) をサポートしています。[13](#) をサポートしています。これは、ゲストオペレーティングシステムが、バックアップやスナップショットのロールバックなどのタイムシフトイベントを検出するために使用できます。

新しいVMを作成すると、*vmgenid*が自動的に生成され、設定ファイルに保存される。

既存のVMに*vmgenid*を作成して追加するには、特別な値'1'を渡してProxmox VEに自動生成させるか、手動で*UUID*を設定します。[14](#) を手動で設定することもできます：

```
# qm set VMID -vmgenid 1
# qm set VMID -vmgenid 00000000-0000-0000-000000000000
```

---

### 注

既存のVMに*vmgenid*デバイスを最初に追加すると、スナップショットのロールバックやバックアップのリストアなどで、VMがこれを世代の変更と解釈するため、変更と同じ効果が生じる可能性がある。

---

*vmgenid*メカニズムが不要な場合は、VM作成時にその値に'0'を渡すか、またはコンフィギュレーションでそのプロ

```
# qm set VMID -delete vmgenid
```

パーティを遡及的に削除することができる：

*vmgenid*の最も顕著な使用例は、新しいMicrosoft Windowsオペレーティング・システムで、スナップショットのロールバックやバックアップのリストア、あるいはVM全体のクローン操作において、時間的制約のあるサービスやレプリケート・サービス（データベースやドメイン・コントローラーなど）の問題を回避するために*vmgenid*を使用している。<sup>15</sup>のような)スナップショット・ロールバック、バックアップ・リストア、またはVM全体のクローン操作における問題を回避するために使用される。

<sup>13</sup>公式 *vmgenid* 仕様 [https://docs.microsoft.com/en-us/windows/desktop/hyperv\\_v2/virtual-machine-generation- 識別子](https://docs.microsoft.com/en-us/windows/desktop/hyperv_v2/virtual-machine-generation- 識別子)

<sup>14</sup>オンラインGUIDジェネレーター <http://guid.one/>

<sup>15</sup><https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/virtualized-domain-controller-建築>

## 10.7 仮想マシンのインポート

海外のハイパーバイザーや他のProxmox VEクラスタから既存の仮想マシンをインポートするには、 さまざまな方法があります：

- ESXiの特殊ストレージが提供するようなインポート・コンテンツ・タイプを利用するネイティブ・インポート・ウィザードを使用する。
- ソースでバックアップを実行し、ターゲットでリストアする。この方法は、別のProxmox VEインスタンスから移行する場合に最適です。
- qm コマンドラインツールの OVF 専用インポートコマンドを使用する。

他のハイパーバイザーからProxmox VEにVMをインポートする場合は、[Proxmox VEの概念](#)に慣れることをお勧めします。

### 10.7.1 インポートウィザード

Import Guest - esxi-7.0:ha-datacenter/tom-nasi/deb-mediawiki/deb-mediawiki.vmx (X)

General   Advanced   Resulting Config

VM ID:	100	Name:	deb-mediawiki
Sockets:	1	CPU Type:	x86-64-v3
Cores:	2	Total cores:	2
Memory (MiB):	1024	OS Type:	Linux
		Version:	6.x - 2.6 Kernel
Default Storage:	cp	Default Bridge:	vnet1
Format:	Raw disk image (raw)		
Live Import:	<input type="checkbox"/>		
Warnings:	<ul style="list-style-type: none"><li>CD-ROM images cannot get imported, if required you can reconfigure the 'sata0' drive in the 'Advanced' tab.</li></ul>		
<span>Import</span>			

Proxmox VEは、APIとWebベースのユーザインターフェースにネイティブに統合するために、ストレージプラ

グインシステムを使用した統合VMインポーターを提供します。これを使用してVMを全体としてインポートし、そのコンフィグのほとんどをProxmox VEのコンフィグモデルにマッピングし、ダウントIMEを削減することができます。

---

### 注

インポートウィザードはProxmox VE 8.2の開発サイクルで追加され、技術プレビューの状態です。すでに有望で安定した動作をしていますが、将来的にはOVF/OVAファイルのような他のインポートソースを追加することに焦点を当て、まだ活発に開発中です。

---

インポートウィザードを使用するには、まずインポートソース用に新しいストレージをセットアップする必要があります。

次に、リソースツリーで新しいストレージを選択し、*Virtual Guests* コンテンツタブを使用して、インポート可能なすべてのゲストを確認できます。

Import Guest - esxi-7.0:ha-datacenter/tom-nasi/deb-mediawiki/deb-mediawiki.vmx

General Advanced Resulting Config

Disks:

Use	Disk ↑	Source	Size	Storage	Format
<input checked="" type="checkbox"/>	scsi0	deb-mediawiki...	32.00 GiB	From Default	Raw disk image

Prepare for VirtIO-SCSI      SCSI Controller: VirtIO SCSI single

CD/DVD Drives:

Use	Slot ↑	Storage	ISO
<input checked="" type="checkbox"/>	sata0	none	none

Network Interfaces:

Use	ID ↑	MAC address	Model	Bridge
<input checked="" type="checkbox"/>	net0	auto	VirtIO (paravirtualize)	From Default

Unique MAC addresses

Import

1つを選択し、インポートボタンを使って（またはダブルクリックして）インポートウィザードを開きます。ここで利用可能なオプションのサブセットを修正し、インポートを開始することができます。インポートが完了した後に、さらに詳細な修正を行うことができます。

## チップ

インポートウィザードは現在（2024-03）、ESXi 用に提供されており、ESXi バージョン 6.5 から 8.0 でテストされています。vSAN ストレージを使用しているゲストは、直接インポートできないことに注意してください。インポート元として vCenter を使用することは可能ですが、パフォーマンスは劇的に低下します（5~10 倍遅くなります）。

仮想ゲストを新しいハイパーバイザーに適応させる方法のステップバイステップのガイドとヒントについては、

Proxmox VE へのマイグレーションの wiki 記事を参照してください。

### 10.7.2 CLIによるOVF/OVAのインポート

海外のハイパーバイザーからのVMエクスポートは通常、VMの設定（RAM、コア数）を記述したコンフィギュレーション・ファイルとともに、1つまたは複数のディスク・イメージの形で行われる。

ディスクイメージは、VMwareやVirtualBoxのディスクの場合はvmdk形式、VMwareやVirtualBoxのディスクの場合はqcow2形式になります。

ディスクはKVMハイパーバイザから提供される。VMエクスポート用の最も一般的なコンフィギュレーション・フォーマットはOVF標準だが、多くの設定が標準自体で実装されておらず、ハイパーバイザーが非標準の拡張機能で補足情報をエクスポートするため、実際には相互運用が制限されている。

フォーマットの問題だけでなく、エミュレートされるハードウェアがハイパーバイザーごとに大きく変わると、他のハイパーバイザからのディスクイメージのインポートに失敗することがある。OSがハードウェアの変更に非常にうるさいため、Windows VMは特にこの問題を気にする。この問題は、エクスポート前にインターネットから入手可能なMergeIDE.zipユーティリティをインストールし、インポートしたWindows VMを起動する前にハードディスクのタイプを**IDEに選択**することで解決できる。

最後に、エミュレートされたシステムの速度を向上させ、ハイパーバイザーに特化した準仮想化ドライバーの問題がある。GNU/Linuxやその他のフリーのUnix OSには必要なドライバがデフォルトでインストールされており、VMをインポートした直後に準仮想化ドライバに切り替えることができる。Windows VMの場合は、Windows準仮想化ドライバを自分でインストールする必要がある。

GNU/Linuxやその他のフリーUnixは、通常、問題なくインポートできます。上記の問題により、Windows VMのインポート/エクスポートがすべてのケースで成功することは保証できません。

### Windows OVFインポートのステップバイステップの例

マイクロソフトは、Windows開発を始めるための**仮想マシンのダウンロード**を提供している。我々は、OVFインポート機能のデモを行うために、これらのうちの1つを使用するつもりだ。

#### 仮想マシンのzipファイルをダウンロードする

ユーザー同意書について説明を受けた後、VMwareプラットフォーム用の*Windows 10 Enterprise (Evaluation - Build)*を選択し、zipをダウンロードする。

#### zipからディスクイメージを取り出す

unzipユーティリティまたは任意のアーカイバを使用してzipを解凍し、ssh/scp経由でovfファイルとvmdkファイルをProxmox VEホストにコピーします。

#### 仮想マシンのインポート

これにより、OVFマニフェストから読み取ったコア、メモリ、VM名を使用して新しい仮想マシンが作成され

```
# qm importovf 999 WinDev1709Eval.ovf local-lvm
```

、ディスクがlocal-lvmストレージにインポートされる。ネットワークは手動で設定する必要がある。

VMを起動する準備ができた。

### 仮想マシンへの外部ディスクイメージの追加

また、既存のディスク・イメージをVMに追加することもできます。これは、他のハイパーバイザーからのものでも、自分で作成したものでもかまいません。

*vmdebootstrap*ツールでDebian/Ubuntuディスクイメージを作成したとする：

```
vmdebootstrap --verbose \
--サイズ 10GiB -シリアルコンソール
--grub --no-extlinux。
-パッケージ openssh-server
-パッケージ avahi-daemon
--package qemu-guest-agent \
--hostname vm600 --enable-dhcp \
--customize=./copy_pub_ssh.sh \
--sparse --image vm600.raw
```

これで新しいターゲットVMを作成し、イメージをストレージpvedirにインポートして、VMのSCSIコントローラにアタッチすることができる：

```
# qm create 600 --net0 virtio,bridge=vmbr0 --name vm600 --serial0 socket \
--boot order=scsi0 --scsihw virtio-scsi-pci --ostype 126  \\
--scsi0 pvedir:0,import-from=/path/to/dir/vm600.raw
```

VMを起動する準備ができた。

## 10.8 クラウド・イニト・サポート

Cloud-Initは、仮想マシンの早期初期化を行う、事実上の複数配布パッケージです。Cloud-Initを使用すると、ハイパーバイザー側でネットワークデバイスとsshキーの設定が可能になる。VMが初めて起動すると、VM内部のCloud-Initソフトウェアがこれらの設定を適用する。

多くのLinuxディストリビューションは、すぐに使えるCloud-Initイメージを提供しているが、そのほとんどはOpenStack用に設計されている。これらのイメージはProxmox VEでも動作する。このようなすぐに使えるイメージ入手するのは便利かもしれません、通常は自分でイメージを準備することをお勧めします。その利点は、何をインストールしたかを正確に知ることができ、後で自分のニーズに合わせてイメージを簡単にカスタマイズできることです。

このようなCloud-Initイメージを作成したら、それをVMテンプレートに変換することをお勧めします。VMテンプレートからリンクされたクローンを素早く作成できるので、これは新しいVMインスタンスをロールアウトする高速な方法だ。新しいVMを起動する前にネットワーク（と多分sshキー）を設定するだけです。

Cloud-InitでプロビジョニングされたVMにログインするには、SSHキーベースの認証を使用することをお勧めします。パスワードを設定することも可能ですが、Proxmox VEはCloud-Initのデータ内に暗号化されたパ

スワードを保存する必要があるため、SSHキーベースの認証ほど安全ではありません。

Proxmox VEはCloud-InitデータをVMに渡すためにISOイメージを生成します。そのためには、すべてのCloud-Init VMにCD-ROMドライブが割り当てられている必要があります。通常、シリアル・コンソールを追加してディスプレイとして使用する。多くのCloud-Initイメージはこれに依存しており、OpenStackの要件となっている。しかし、他のイメージではこの設定に問題があるかもしれません。シリアルコンソールを使ってうまくいかない場合は、デフォルトのディスプレイ構成に戻してください。

### 10.8.1 Cloud-Initテンプレートの準備

最初のステップはVMの準備だ。基本的にはどんなVMでも使える。Cloud-Initパッケージをインストールするだけだ。

を準備したいVMの内部にインストールします。Debian/Ubuntuベースのシステムでは、これは以下のように簡単だ：

```
apt-get install cloud-init
```



## 警告

このコマンドはProxmox VEホスト上では実行されず、VM内部でのみ実行されます。

すでに多くのディストリビューションがすぐに使えるCloud-Initイメージ（.qcow2ファイルとして提供）を提供しているので、そのようなイメージをダウンロードしてインポートすることもできます。以下の例では、Ubuntuが提供するクラウドイメージ (<https://cloud-images.ubuntu.com>) を使用します。

```
# 画像をダウンロードする
wget https://cloud-images.ubuntu.com/bionic/current/bionic-server-cloudimg-'amd64.img

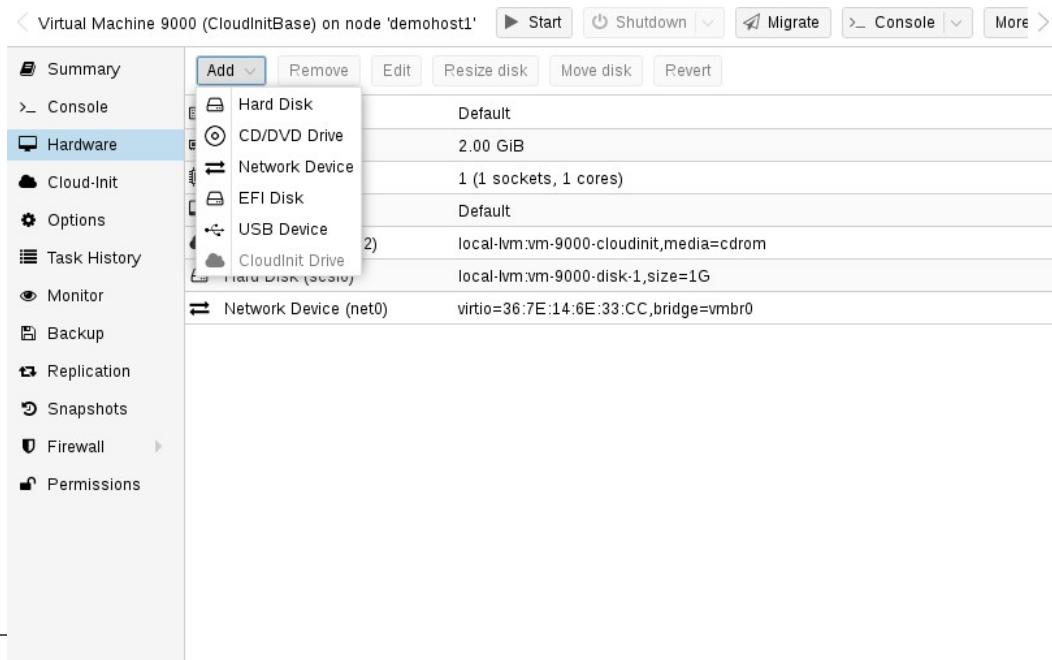
# VirtIO SCSI コントローラで新しい VM を作成します。
qm create 9000 --memory 2048 --net0 virtio,bridge=vmbr0 --scsihw virtio-'scsi-pci

# ダウンロードしたディスクをlocal-lvmストレージにインポートする。
SCSI ドライブ
qm set 9000 --scsi0 local-lvm:0,import-from=/path/to/bionic-server-cloudimg-'o
-amd64.img
```

## 注

Ubuntu Cloud-Init イメージには、SCSI ドライブ用に `virtio-scsi-pci` コントローラタイプが必要です。

## Cloud-InitのCD-ROM ドライブを追加する



次のステップはCD-ROMドライブを設定することである。CD-ROMドライブはCloud-InitデータをVMに渡すために使用される。

```
qm set 9000 --ide2 local-lvm:cloudinit
```

Cloud-Initイメージから直接ブートするには、ブートパラメータを`order=scsi0`に設定して、BIOSがこのディスクからのみブートするように制限します。これにより、VM BIOSがブート可能なCD-ROMのテストをスキップするため、ブートが高速化されます。

```
qm set 9000 --boot order=scsi0
```

多くのCloud-Initイメージでは、シリアルコンソールを設定してディスプレイとして使用する必要があります

```
qm set 9000 --serial0 socket --vga serial0
```

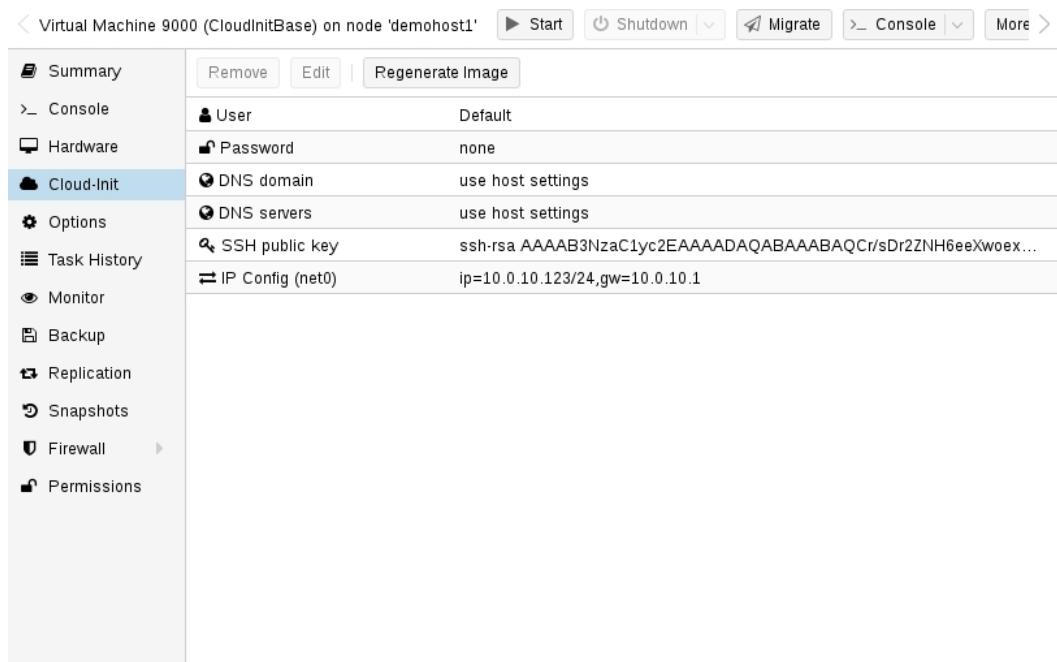
。しかし、その設定が特定のイメージで機能しない場合は、代わりにデフォルトのディスプレイに切り替えてください。

最後のステップでは、VMをテンプレートに変換するのが便利だ。このテンプレートからリンクされたクロ

#### QMテンプレート9000

ーンを素早く作成できる。VMテンプレートからのデプロイは、完全なクローン（コピー）を作成するよりもはるかに高速だ。

### 10.8.2 Cloud-Initテンプレートの展開



このようなテンプレートはクローンすることで簡単に配備できる：

```
qm clone 9000 123 --name ubuntu2
```

次に、認証に使用するSSH公開鍵を設定し、IP設定を行う：

```
qm set 123 --sshkey ~/.ssh/id_rsa.pub  
qm set 123 --ipconfig0 ip=10.0.10.123/24,gw=10.0.10.1
```

Cloud-Initのすべてのオプションを1つのコマンドだけで設定することもできます。上記の例では、行数を減らすためにコマンドを単純に分割しています。また、特定の環境にIPセットアップを採用するようにしてください。

### 10.8.3 カスタム・クラウド初期設定

Cloud-Initの統合では、自動生成されたコンフィグファイルの代わりにカスタムコンフィグファイルを使用することができます。

```
qm set 9000 --cicustom "user=<volume>,network=<volume>,meta=<volume>"
```

もできます。これはコマンドラインの `cicustom` オプションで行います：

カスタム設定ファイルはスニペットをサポートするストレージ上にあり、VMを移行するすべてのノードで

```
qm set 9000 --cicustom "user=local:snippets/userconfig.yaml"
```

利用可能でなければならない。そうでなければVMは起動できない。例えば

Cloud-Initには3種類のコンフィグがあります。1つ目は上の例にあるようにユーザー設定です。2つ目はネットワーク設定、3つ目はメタ設定です。これらはすべて一緒に指定することもできますし、必要に応じて組み合わせて指定することもできます。カスタムコンフィグファイルが指定されていない場合は、自動生成されたコンフィグが使われます。

生成されたコンフィグは、カスタムコンフィグのベースとしてダンプすることができる：

```
qm cloudinit dump 9000 ユーザー
```

ネットワークとメタにも同じコマンドがある。

### 10.8.4 クラウドイット特有のオプション

**カスタム:** [`meta=<volume>`] [, `network=<volume>`] [, `user=<volume>`]

[, `vendor=<volume>`]。

開始時に自動生成されたファイルを置き換えるカスタムファイルを指定する。

**meta=<ボリューム**

cloud-init経由でVMに渡されるすべてのメタデータを含むカスタムファイルを指定する。これはプロバイダー固有で、configdrive2とnocloudは異なる。

**ネットワーク=<ボリューム**

すべてのネットワークデータを含むカスタムファイルをcloud-init経由でVMに渡す。

**ユーザー=<ボリューム**

全てのユーザーデータを含むカスタムファイルをcloud-init経由でVMに渡す。

**ベンダー=<ボリューム**

すべてのベンダーデータを含むカスタムファイルをcloud-init経由でVMに渡す。

**cipassword: <文字列**

ユーザーに割り当てるパスワード。これを使用することは一般的に推奨されません。代わりにsshキーを使ってください。また、古いバージョンのcloud-initはハッシュ化されたパスワードをサポートしていないことに注意してください。

**citype: <configdrive2 | nocloud | opennebula>**

cloud-initコンフィギュレーション・フォーマットを指定します。デフォルトは、設定されているオペレーティング・システムの種類 (ostype.Linuxではnocloud形式、Windowsではconfigdrive2を使用します。

**ciupgrade: <boolean> (デフォルト = 1)**

最初の起動後にパッケージの自動アップグレードを行います。

**ciuser: <文字列>**

イメージに設定されているデフォルトユーザーの代わりに、sshキーとパスワードを変更するユーザー名。

**ipconfig[n]: [gw=<GatewayIPv4>] [, gw6=<GatewayIPv6>]**

[, ip=<IPv4Format/CIDR>] [, ip6=<IPv6Format/CIDR>] とする。

対応するインターフェイスのIPアドレスとゲートウェイを指定します。

IPアドレスはCIDR表記を使用し、ゲートウェイはオプションだが、同じタイプのIPを指定する必要がある。

特別な文字列*dhcp*は、DHCPを使用するIPアドレスに使用でき、その場合、明示的な ゲートウェイを指定する必要はない。IPv6の場合、ステートレス自動設定を使用するには、特別な文字列*auto*を使用できる。これにはcloud-init 19.4以降が必要です。

cloud-initが有効で、IPv4アドレスもIPv6アドレスも指定されていない場合、デフォルトでIPv4の*dhcp*を使用する。

**gw=<ゲートウェイIPv4>**

IPv4トラフィックのデフォルトゲートウェイ。

---

**注**

必要なオプション: *ip*

---

**gw6=<ゲートウェイIPv6>**

IPv6トラフィックのデフォルトゲートウェイ。

---

**注**

必要なオプション: *ip6*

---

**ip=<IPv4Format/CIDR> (デフォルト = dhcp)**

CIDR形式のIPv4アドレス。

**ip6=<IPv6Format/CIDR> (デフォルト = dhcp)**

CIDR形式のIPv6アドレス。

---

**ネームサーバー: <文字列>**

コンテナの DNS サーバー IP アドレスを設定する。searchdomainもnameserverも設定されていない場合、Createはホストからの設定を自動的に使用します。

**searchdomain: <文字列>**

コンテナのDNS検索ドメインを設定する。searchdomainもnameserverも設定されていない場合、Createはホストからの設定を自動的に使用します。

**sshkeys: <文字列>**

公開SSH鍵を設定する（1行に1つの鍵、OpenSSH形式）。

## 10.9 PCI(e)バススルー

PCIバススルーは、ホストから仮想マシンにPCIデバイスを制御させるメカニズムです。これにより、仮想化ハードウェアを使用する場合よりも、低レイテンシ、高パフォーマンス、より多くの機能（オフロードなど）などの利点があります。

しかし、デバイスを仮想マシンに渡すと、そのデバイスはホスト上でも他のVMでも使えなくなる。

PCIバススルーはi440fxおよびq35マシンで利用可能ですが、PCIeバススルーはq35マシンのみで利用可能です。これは、PCIデバイスとしてバススルーされたPCIe対応デバイスがPCI速度でのみ動作するという意味ではありません。PCIeとしてデバイスを通過させることは、デバイスが "本当に高速なレガシー PCI デバイス" ではなく PCIe デバイスであることをゲストに伝えるためのフラグを設定するだけです。一部のゲストアプリケーションはこの恩恵を受けます。

### 10.9.1 一般要件

バススルーは実際のハードウェア上で実行されるため、いくつかの要件を満たす必要がある。これらの要件の概要を以下に示します。特定のデバイスに関する詳細については、[PCIバススルーの例](#)を参照してください。

#### ハードウェア

ハードウェアがIOMMU（I/Oメモリー管理ユニット）割り込みリマッピングをサポートしている必要があります。これにはCPUとマザーボードが含まれます。

一般的に、VT-dを搭載したIntelシステムとAMD-Viを搭載したAMDシステムはこれをサポートしている。しかし、ハードウェアの実装が悪かったり、ドライバが不足していたり、品質が低かったりするため、箱から出してすぐにすべてが動作する保証はありません。

さらに、サーバーグレードのハードウェアは、コンシューマーグレードのハードウェアよりもサポートが優れていることが多いが、それでも多くの最新システムはこれをサポートできる。

この機能がLinuxでサポートされているかどうかは、ハードウェアベンダーにお問い合わせください。

#### PCIカードアドレスの決定

最も簡単な方法は、GUIを使用してVMのハードウェア・タブに「ホストPCI」タイプのデバイスを追加することです。また、コマンドラインを使用することもできます。

カードを探すには

エルエスピーシー

## 構成

ハードウェアがパススルーをサポートしていることを確認したら、PCI(e) パススルーを有効にするための設定を行う必要があります。

## IOMMU

まず、BIOS/UEFI で IOMMU サポートを有効にする必要があります。通常、対応する設定は IOMMU または VT-d ですが、正確なオプション名はマザーボードのマニュアルに記載されています

```
intel_iommu=on
```

要があります：

AMD CPU の場合は自動的に有効になるはずだ。

## IOMMU パススルーモード

ハードウェアが IOMMU パススルーモードをサポートしている場合、このモードを有効にするとパフォーマン

```
iommu=pt
```

スが向上する可能性があります。これは、VM が通常ハイパーバイザによって実行される（デフォルトの）DMA 変換をバイパスし、代わりに DMA 要求をハードウェア IOMMU に直接渡すためです。これらのオプションを有効にするには、以下を追加する：

をカーネルのコマンドラインに追加する。

## カーネルモジュール

以下のモジュールがロードされていることを確認する必要がある。これは、'etc/modules' に追加すること

```
vfio
vfio_iommu_type1
vfio_pci
vfio_virqfd #カーネル6.2以降では不要
```

で実現できます。6.2より新しいカーネル（Proxmox VE 8以降）では、`vfio_virqfd` モジュールは `vfio` モジュールの一部なので、Proxmox VE 8以降で `vfio_virqfd` をロードする必要はありません。

モジュール関連の何かを変更したら、`initramfs` をリフレッシュする必要があります。Proxmox VE では、これを実行することで行えます：

```
# update-initramfs -u -k all
```

モジュールがロードされているかどうかをチェックするには

```
# lsmod | grep vfio
```

上記の4つのモジュールが含まれていなければならない。

## 仕上げの構成

最後に再起動して変更を有効にし、実際に有効になっていることを確認する。

```
# dmesg | grep -e DMAR -e IOMMU -e AMD-Vi
```

は、IOMMU、Directed I/O、またはInterrupt Remappingが有効であることを表示するはずですが、ハードウェアやカーネルによって正確なメッセージは異なります。

IOMMUが意図したとおりに動作しているかどうかを確認する方法については、Wikiの「[IOMMU パラメータの検証](#)」セクションを参照してください。

また、通過させたいデバイスが別のIOMMUグループに属していることも重要です。これはProxmox VE APIを呼び出すことで確認できます：

```
# pvesh get /nodes/{nodename}/hardware/pci --pci-class-blacklist ""
```

デバイスが、その機能（例えば、HDMIオーディオデバイスを持つGPU）と一緒にIOMMUグループにあるか、そのルートポートまたはPCI(e)ブリッジと一緒にある場合は問題ありません。

### PCI(e)スロット

プラットフォームによっては、物理的なPCI(e)スロットの扱いが異なります。そのため、希望のIOMMUグループ分離が得られない場合、カードを別のPCI(e)スロットに置くことが助けになることもあります。

### 安全でない割り込み

プラットフォームによっては、安全でない割り込みを許可する必要があるかもしれません。この場合、

```
オプション vfio_iommu_type1 allow_unsafe_interrupts=1
```

/etc/modprobe.d/の'.conf'で終わるファイルに以下の行を追加する：

このオプションはシステムを不安定にする可能性がありますのでご注意ください。

### GPUパススルーの注意事項

Proxmox VEのウェブインターフェースで、NoVNCやSPICEを介してGPUのフレームバッファを表示することはできません。

GPU全体またはvGPUを通過し、グラフィック出力が必要な場合、物理的にモニターをカードに接続するか、ゲスト内でリモートデスクトップソフトウェア（例えばVNCやRDP）を設定する必要があります。

GPUをハードウェアアクセラレーターとして使用したい場合、たとえばOpenCLやCUDAを使用するプログラムでは、これは必要ありません。

## 10.9.2 ホスト・デバイス・パススルー

PCI(e)パススルーで最もよく使われるのは、PCI(e)カード全体（GPUやネットワークカードなど）をパススルーする方法である。

### ホスト設定

Proxmox VEは、自動的にPCI(e)デバイスをホストから利用できないようにしようとします。しかし、これがうまくいかない場合、できることが2つあります：

- を追加することで、デバイスIDを*vfio-pci*モジュールのオプションに渡します。

```
オプション vfio-pci ids=1234:5678,4321:8765
```

を/etc/modprobe.d/の.confファイルに追加する。ここで、1234:5678と4321:8765は、ベンダIDとデバイスIDである：

```
# lspci -nn
```

### ブラックリスト DRIVERNAME

- ホスト上のドライバを完全にブラックリストに入れ、パススルー用にバインドできるようにする。

を/etc/modprobe.d/の.confファイル

ルに追加する。ドライバ名を見つ

けるには

```
# lspci -k
```

例えば

```
# lspci -k | grep -A 3 "VGA"
```

のようなものが表示される。

```
01:00.0 VGA互換コントローラ: NVIDIA Corporation GP108 [GeForce GT ←' 1030] (rev a1)
    サブシステムマイクロスターインターナショナル株式会社 [msi] gp108 [←' GeForce GT 1030]
    使用中のカーネルドライバ: <some-module> カー
    ネルモジュール: <some-module>
```

これで、ドライバを.confファイルに記述してブラックリスト化することができる：

```
echo "ブラックリスト <some-module>" >> /etc/modprobe.d/blacklist.conf
```

どちらの方法でも、[initramfsを再度更新](#)し、その後再起動する必要がある。

これがうまくいかない場合は、`vfio-pci`をロードする前にgpuモジュールをロードするよう、ソフト依存を設定する必要があるかもしれません。これは`softdep`フラグで実行できます。詳しくは`modprobe.d`のマニュアルページも参照してください。

例えば、`<some-module>`という名前のドライバーを使っている場合：

```
# echo "softdep <some-module> pre: vfio-pci" >> /etc/modprobe.d/<some-<' モジュール>.conf
```

### 設定の確認

変更が成功したかどうかを確認するには

```
# lspci -nnk
```

をクリックし、デバイスのエントリーを確認してください。もし

使用中のカーネルドライバ: `vfio-pci`

または使用中の行がまったくない場合、デバイスはパススルーに使用する準備ができている。

## VMの構成

GPU をパススルーする場合、マシンタイプとして `q35`、SeaBIOS の代わりに `OVMF` (VM 用 `UEFI`)、PCI の代わりに PCIe を使うと最高の互換性が得られます。GPU パススルーに `OVMF` を使いたい場合、GPU が UEFI 対応の ROM を持っている必要があることに注意してください。ROM が UEFI に対応しているかチェックするには [PCI Passthrough Examples](#) wiki を見て下さい。

さらに、`OVMF`を使えば、`vga`アービトレーションを無効にすることができる、ブート中に実行する必要のあるレガシーコードの量を減らすことができる。`vga`アービトレーションを無効にするには

```
echo "options vfio-pci ids=<vendor-id>,<device-id> disable_vga=1" > /etc/modprobe.d/vfio.conf
```

で取得した<vendor-id>と<device-id>に置き換える:

```
# lspci -nn
```

PCIデバイスは、VMのハードウェア・セクションにあるウェブ・インターフェイスで追加できる。VMコン

```
# qm set VMID -hostpci0 00:02.0
```

フィギュレーションで**hostpciX**オプションを設定します:

またはVMコンフィギュレーション・ファイルに行を追加する:

```
ホストPCI0: 00:02.0
```

デバイスに複数のファンクション（例えば'00:02.0'と'00:02.1'）がある場合は、``00:02``という短縮構文でまとめて渡すことができます。これはウェブインターフェイスの`All Functions`チェックボックスをチェックするのと同じです。

デバイスとゲストOSによっては、必要なオプションがいくつかあります:

- **x-vga=on|off**は、PCI(e)デバイスをVMのプライマリGPUとしてマークします。これを有効にすると、**vga**設定オプションは無視されます。
- **pcie=on|off**はPCIeまたはPCIポートを使用するようProxmox VEに指示します。ゲスト/デバイスの組み合わせによってはPCIではなくPCIeを必要とします。PCIeはq35マシンタイプでのみ使用可能です。
- **rombar=on|off**は、ファームウェア ROM をゲストから見えるようにします。デフォルトは **on** です。いくつかの PCI(e) デバイスは、これを無効にする必要があります。
- **romfile=<path>**には、デバイスが使用するROMファイルへのパスをオプションで指定します。これは **/usr/share/kvm/**.

## 例

GPUをプライマリーに設定したPCIeバススルーの例:

```
# qm set VMID -hostpci0 02:00,pcie=on,x-vga=on.
```

## PCI IDオーバーライド

ゲストによって認識される PCI ベンダー ID、デバイス ID、サブシステム ID を上書きすることができます。これは、デバイスがゲストのドライバが認識しない ID を持つバリアントであるにもかかわらず、それらのドライバを強制的にロードしたい場合に便利です (例えば、デバイスがサポートされているバリアントと同じチップセットを共有していることが分かっている場合など)。

使用可能なオプションは、`vendor-id`、`device-id`、`sub-vendor-id`、`sub-device-id`です。デバイスのデフォルトIDを上書きするために、これらのいずれかまたはすべてを設定することができます。

例えば、こうだ:

```
# qm set VMID -hostpci0 02:00,device-id=0x10f6,sub-vendor-id=0x0000.
```

### 10.9.3 SR-IOV

PCI(e)デバイスを通過させるもう一つの方法は、もし利用可能であれば、デバイスのハードウェア仮想化機能を使うことです。

#### SR-IOVの実現

SR-IOVを使用するには、プラットフォームのサポートが特に重要です。最初に BIOS/UEFI でこの機能を有効にするか、特定の PCI(e)ポートを使用する必要があるかもしれません。不明な点は、プラットフォームのマニュアルを参照するか、ベンダーにお問い合わせください。

SR-IOV (Single-Root Input/Output Virtualization: シングルルート入出力仮想化) は、1つのデバイスが複数のVF (仮想機能) をシステムに提供することを可能にする。これらのVFはそれぞれ異なるVMで使用することができ、完全なハードウェア機能を備え、ソフトウェア仮想化デバイスよりも優れたパフォーマンスと低レイテンシーを実現します。

現在、最も一般的なユースケースは、SR-IOV をサポートする NIC (ネットワーク・インターフェース・カード) で、物理ポートごとに複数の VF を提供することができる。これにより、チェックサム・オフロードなどの機能をVM内部で使用することができ、(ホストの) CPUオーバーヘッドを削減することができる。

#### ホスト設定

一般的に、デバイス上で仮想機能を有効にするには2つの方法がある。

- ドライバー・モジュールにオプションがある場合もある。

```
max_vfs=4
```

これは、`/etc/modprobe.d/` の下に `.conf` で終わるファイルを置くことができます。(その後、initramfsを更新するのを忘れないでください)。

正確なパラメータとオプションについては、ドライバ・モジュールのマニュアルを参照してください。

- つ目の、より汎用的なアプローチは、sysfsを使うことだ。デバイスとドライバがこれをサポートしてい

```
# echo 4 > /sys/bus/pci/devices/0000:01:00.0/sriov_numvfs
```

れば、VFの数をその場で変更することができます。例えば、デバイス0000:01:00.0に4つのVFをセットアップするには、次のように実行します:

この変更を持続させるには、「sysfsutils」 Debian パッケージを使用する。インストール後、

`/etc/sysfs.conf` または `/etc/sysfs.d/`FILE.conf`` で設定する。

## VMの構成

VF を作成した後、それらを `lspci` で出力すると、別々の PCI(e) デバイスとして見えるはずです。それらの ID を取得し、[通常の PCI\(e\) デバイス](#) のように通過させる。

### 10.9.4 媒介デバイス（vGPU、GVT-g）

仲介デバイスは、物理ハードウェアの機能と性能を仮想化ハードウェアに再利用するもう1つの方法です。これらは、IntelのGVT-gやNVIDIAのGRIDテクノロジーで使用されているvGPUのような仮想化GPUセットアップで最もよく見られます。

これにより、物理カードはSR-IOVと同様に仮想カードを作成することができる。違いは、mediatedデバイスはホスト上ではPCI(e)デバイスとして表示されず、仮想マシンでの使用にのみ適していることです。

## ホスト設定

一般的に、カードのドライバがその機能をサポートしていなければなりません。そのため、互換性のあるドライバとその設定方法については、各ベンダーにお問い合わせください。

インテルのGVT-g用ドライバーはカーネルに統合されており、第5、第6、第7世代のインテル・コア・プロセッサー、およびE3 v4、E3 v5、E3 v6のXeonプロセッサーで動作するはずだ。

インテル・グラフィックスで有効にするには、モジュール*kvmgt*を（例えば/etc/modules経由で）ロードし、

```
i915.enable_gvt=1
```

カーネルのコマンドラインで有効にして、以下のパラメーターを追加する必要があります：

その後、[initramfs](#)を忘れずに更新し、ホストを再起動する。

## VMの構成

調停されたデバイスを使用するには、`hostpciX` VMコンフィギュレーション・オプションで`mdev`プロパティを指定するだけです。サポートされているデバイスは、`sysfs`経由で取得できます。例えば、デバイス`0000:00:02.0`のサポートされているタイプをリストアップするには、次のように実行します：

```
# ls /sys/bus/pci/devices/0000:00:02.0/mdev_supported_types
```

各エントリーは、以下の重要なファイルを含むディレクトリである：

- `available_instances`には、このタイプのまだ利用可能なインスタンスの量が含まれる。
- `description`には、その型の能力に関する短い説明が含まれます。
- `create`は、このようなデバイスを作成するためのエンドポイントです。

オプションが設定されている。

```
# qm set VMID -hostpci0 00:02.0,mdev=i915-GVTg_V5_4。
```

Intel GVT-g vGPU (Intel Skylake 6700k) を使用した構成例：

この設定により、Proxmox VEはVM起動時にこのようなデバイスを自動的に作成し、VM停止時に再びクリーンアップします。

## 10.9.5 クラスターでの使用

クラスタレベルでデバイスをマッピングすることも可能で、HAで適切に使用でき、ハードウェアの変更が検出され、ルートユーザー以外でも設定できるようになる。詳細は[リソースマッピング](#)を参照してください。

## 10.9.6 vIOMMU（エミュレートされたIOMMU）

vIOMMUは、仮想マシン内でハードウェアIOMMUをエミュレートし、仮想化I/Oデバイスのメモリアクセス制御とセキュリティを向上させます。vIOMMUオプションを使用すると、[Nested Virtualization](#)を介して、レベル1 VMのレベル2 VMにPCIデバイスを渡すこともできます。現在、2つのvIOMMU実装が利用可能です：IntelとVirtIOです。

ホストの条件

- CPUに応じてintel\_iommu=onまたはamd\_iommu=onをカーネルのコマンドラインに追加する。

## インテル vIOMMU

Intel vIOMMU固有のVM要件:

- ホストでインテルまたはAMDのどちらのCPUを使用している場合でも、VMのカーネル・パラメーターで `intel_iommu=on` を設定することが重要です。
- インテルvIOMMUを使用するには、マシンタイプとして **q35** を設定する必要があります。

すべての要件が満たされていれば、PCIデバイスを通過できるはずのVMのコンフィギュレーションのマシン

- パラメーターに `viommu=intel` を追加できる。

```
# qm set VMID -machine q35,viommu=intel
```

## VT-d 用 QEMU ドキュメント

### ヴィルティオ・ヴィオム

このvIOMMUの実装はより新しく、Intel vIOMMUほど多くの制限はないが、現在実運用ではあまり使われておらず、文書化もされていない。

VirtIO vIOMMUでは、カーネル・パラメータを設定する必要はない。また、マシンタイプとして **q35** を使用する

```
# qm set VMID -machine q35,viommu=virtio
```

必要はありませんが、PCIeを使用する場合は使用することをお勧めします。

[virtio-iommuを説明するマイケル・ザオのブログ記事](#)

## 10.10 フックスクリプト

設定プロパティ `hookscript` で、VMにフックスクリプトを追加できる。

```
# qm set 100 --hookscript local:snippets/hookscript.pl
```

このスクリプトはゲストが生きている間の様々な局面で呼び出されます。例とドキュメントは

`/usr/share/pve-docs/examples/guest-example-hookscript.pl` にあるサンプルスクリプトを参照してください。

## 10.11 冬眠

VMをディスクにサスペンドするには、GUIオプションのHibernateまたは

```
# qm suspend ID --todisk
```

つまり、メモリーの現在の内容がディスクに保存され、VMは停止する。次の起動時には、メモリの内容がロードされ、VMは中断したところから続行できる。

## ステート・ストレージの選択

メモリーのターゲット・ストレージが指定されていない場合は、自動的に選択される：

1. VMコンフィグからストレージvmstatestorage。
2. どのVMディスクからも最初の共有ストレージ。
3. どのVMディスクからも最初の非共有ストレージ。
4. フォールバックとしてのストレージ・ローカル。

## 10.12 リソースマッピング

The screenshot shows the Proxmox VE web interface under the 'Datacenter' tab. On the left, a sidebar lists various management sections: Search, Summary, Notes, Cluster, Ceph, Options, Storage, Backup, Replication, Permissions (with sub-options for Users, API Tokens, Two Factor, Groups, Pools, Roles, and Realms), HA, ACME, Firewall, Metric Server, and Resource Mappings (which is currently selected). The main content area is divided into two sections: 'PCI Devices' and 'USB Devices'. Both sections have an 'Add' button and a table with columns: ID/Node/Path, Actions, Vendor/De..., Subsystem..., IOMMU gr..., Status, and Comment.

**PCI Devices:**

ID/Node/Path	Actions	Vendor/De...	Subsystem...	IOMMU gr...	Status	Comment
-> NIC	⊕ ↖ ↖					
pve-ceph-01	⊕ ↖ ↖					
└ 0000:06:12.0	⊕ ↖ ↖	1af4:1000	1af4:0001	9	✓ Mapping matches host data	
pve-ceph-02	⊕ ↖ ↖					
└ 0000:06:12.0	⊕ ↖ ↖	1af4:1000	1af4:0001	9	✓ Mapping matches host data	

**USB Devices:**

ID/Node/Vendor&Device	Actions	Path	Status	Comment
-> STICK	⊕ ↖ ↖			
pve-ceph-01	⊕ ↖ ↖			
└ ↙ 0627:0001	⊕ ↖ ↖		✓ Mapping matches host data	
pve-ceph-02	⊕ ↖ ↖			
└ ↙ 0627:0001	⊕ ↖ ↖	1-1	✓ Mapping matches host data	

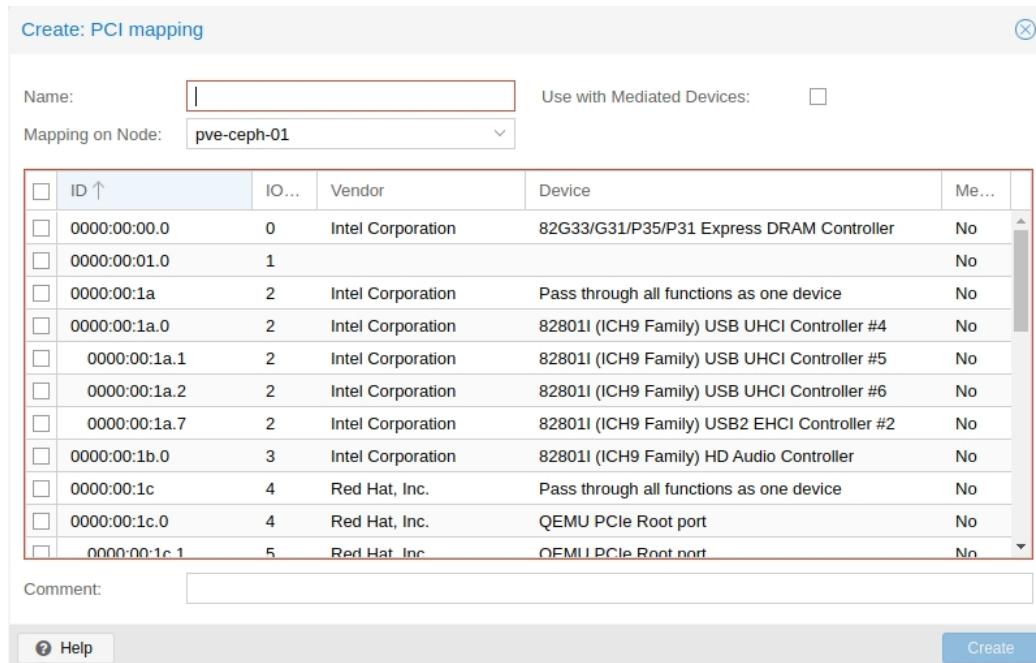
ローカル・リソース（例えばpciデバイスのアドレス）を使用したり参照したりする場合、生のアドレスやidを使用するのは問題がある場合がある：

- HAを使用する場合、ターゲットノード上に同じIDやパスを持つ別のデバイスが存在する可能性があり、そのようなゲストをHAグループに割り当てる際に注意しなければ、間違ったデバイスが使用され、設定が壊れる可能性がある。
- ハードウェアを変更するとIDやパスが変更される可能性があるため、割り当てられたすべてのデバイスをチェックし、パスやIDが正しいかどうかを確認する必要がある。

これをうまく処理するために、クラスタ全体のリソースマッピングを定義して、リソースがクラスタ固有の、ユーザが選択した識別子を持ち、異なるホスト上の異なるデバイスに対応できるようにすることができます。これにより、HAが間違ったデバイスでゲストを起動することはなくなり、ハードウェアの変更も検出できるようになります。

このようなマッピングの作成は、Proxmox VE の Web GUI でリソースマッピングカテゴリの関連タブに

```
# pvsh create /cluster/mapping/<type> <options>
```



あるデータセンターで行うか、または cli で次のように操作します。

ここで<type>はハードウェアのタイプ（現在はpciかusb）、<options>はデバイスのマッピングとその他の設定パラメーターである。

オプションには、ハードウェアが変更されておらず、正しいデバイスが通過していることを確認できるよう、そのハードウェアのすべての識別プロパティを含むマッププロパティが含まれていなければならないことに注意してください。

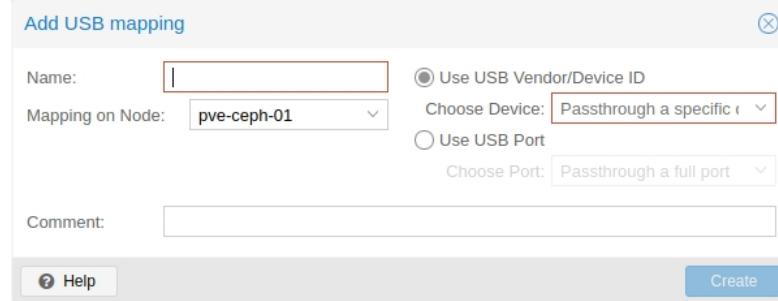
例えば、デバイスID 0001を持つパス0000:01:00.0を持つPCIデバイスをdevice1として追加するには、次のようにします。

で、ノードnode1ではベンダーID 0002、ノードnode2では0000:02:00.0で追加できる：

```
# pvsh create /cluster/mapping/pci --id device1 \
--node=node1,pass=0000:01:00.0,id=0002:0001 \
--map node=node2,path=0000:02:00.0,id=0002:0001
```

デバイスがマッピングされるべき各ノードに対して、mapパラメータを繰り返す必要があります（現在、1つのマッピングにつき1つのノードにつき1つのUSBデバイスしかマッピングできないことに注意してください）。

GUIを使えば、正しいプロパティが自動的にピックアップされ、APIに送信されるので、これはずっと簡単に



なる。

PCIデバイスでは、ノードごとに複数のマッププロパティを持つ複数のデバイスを提供することも可能です。このようなデバイスがゲストに割り当てられた場合、ゲストの起動時に最初に空いているものが使用されます。与えられたパスの順番は試行される順番でもあるため、任意の割り当てポリシーを実装できます。

これは、SR-IOVを持つデバイスにとって有用である。なぜなら、どの仮想関数を通過させるかは重要でない場合があるからだ。

このようなデバイスをゲストに割り当てるには、GUI または

```
# qm set ID -hostpci0 <name>
```

PCIデバイスの場合

```
# qm set <vmid> -usb0 <name>
```

USBデバイス用。

ここで、<vmid> はゲストID、<name> は作成されるマッピングの名前である。mdev のような、デバイスを通過するための通常のオプションはすべて許可されている。

マッピングを作成するには、/mapping/<type>/<name>に対するMapping.Modifyが必要である。（<type> はデバイスタイプ、<name> はマッピング名）。

これらのマッピングを使用するには、/mapping/<type>/<name>上のMapping.Useが必要です（設定を編集する通常のゲスト権限に加えて）。

## 10.13 qmによる仮想マシンの管理

qm は Proxmox VE 上の QEMU/KVM 仮想マシンを管理するツールです。仮想マシンの作成と破棄、実行の制御(スタート/ストップ/サスペンド/リジューム)が可能です。さらに、qm を使用して、関連する設定ファイルにパラメータを設定できます。仮想ディスクの作成と削除も可能です。

### 10.13.1 CLIの使用例

ローカル・ストレージにアップロードされたisoファイルを使って、4GBのIDEディスクを持つVMをlocal-lvmストレージに作成する。

```
# qm create 300 -ide0 local-lvm:4 -net0 e1000 -cdrom local:iso/proxmox-'  
  メールゲートウェイ_2.1.iso'
```

## 新しいVMを起動する

```
# qm start 300
```

シャットダウン要求を送信し、VMが停止するまで待つ。

```
# qm shutdown 300 && qm wait 300
```

上記と同じだが、40秒間だけ待つ。

```
# qm shutdown 300 && qm wait 300 -timeout 40
```

VMがシャットダウンしない場合は、VMを強制停止し、実行中のシャットダウン・タスクを上書きする。

VMを停止するとデータが失われる可能性があるため、使用には注意が必要である。

```
# qm stop 300 -overrule-shutdown 1
```

VMを破棄すると、そのVMは常にアクセス・コントロール・リストから削除され、そのVMのファイアウォー

```
# qm destroy 300 --purge
```

ル設定も常に削除される。レプリケーション・ジョブ、バックアップ・ジョブ、HAリソース・コンフィギュ  
レーションからもVMを削除したい場合は、*-purge* を有効にする必要がある。

ディスクイメージを別のストレージに移動する。

```
# qm move-disk 300 scsi0 other-storage
```

ディスクイメージを別のVMに再割り当てる。これにより、ディスク scsi1 がソース VM から削除され  
、scsi3 としてターゲット VM にアタッチされます。バックグラウンドでディスク・イメージは新しいオ  
ーナーと名前が一致するようにリネームされます。

```
# qm move-disk 300 scsi1 --target-vmid 400 --target-disk scsi3
```

## 10.14 構成

VM設定ファイルはProxmoxクラスタファイルシステム内に保存され、/etc/pve/qemuにアクセスできます。  
etc/pve/に格納されている他のファイルと同様に、他のすべてのクラスタノードに自動的に複製されます。

---

### 注

100未満のVMIDは内部用に予約されており、VMIDはクラスタ全体で一意である必要がある。

---

### VMの構成例

```
boot:  
order=virtio0;net0 コア  
: 1  
ソケット1  
メモリ512 name:  
webmail  
ostype: 126  
net0: e1000=EE:D2:28:5F:B6:3E,bridge=vmbr0  
virtio0: local:vm-100-disk-1,size=32G。
```

これらの設定ファイルは単純なテキストファイルであり、通常のテキストエディタ（vi、nano、...）。これは小さな修正を行うのに便利な場合もあるが、そのような変更を適用するにはVMを再起動する必要があることに留意してほしい。

そのため、通常はqmコマンドを使用してファイルを生成・変更するか、GUIを使用してすべてを行う方がよい。我々のツールキットは賢いので、実行中のVMにほとんどの変更を瞬時に適用することができる。この機能は「ホットプラグ」と呼ばれ、この場合VMを再起動する必要はない。

### 10.14.1 ファイル形式

VMコンフィギュレーション・ファイルは、コロンで区切られたシンプルなキー／バリュー・フォーマットを使用する。各行のフォーマットは以下の通り：

```
# これはコメントです。
```

これらのファイルの空白行は無視され、#文字で始まる行はコメントとして扱われ、これも無視される。

### 10.14.2 スナップ写真

スナップショットを作成すると、qmはスナップショット時の設定と同じ設定ファイル内の別のスナップショット秒に保存します。例えば、"testsnapshot" というスナップショットを作成した後、設定ファイルは次のようにになります：

#### スナップショットによるVM構成

```
メモリ: 512
スワップ: 512
親: テストナフォト
...
[testsnapshot]
メモリ: 512
スワップ: 512
スナップタイム: 1457170803
...
```

parentやsnaptimeのようなスナップショット関連のプロパティがいくつかあります。parentプロパティはスナップショット間の親子関係を保存するために使用されます。snaptimeはスナップショット作成のタイムスタンプ（Unixエポック）です。

オプションのvmstateを使用すると、実行中のVMのメモリを保存できます。VMの状態に対してターゲット・ストレージがどのように選択されるかの詳細については、「[休止状態](#)」の章の「[状態ストレージの選択](#)」を参照してください。

### 10.14.3 オプション

**acpi: <ブール値> (デフォルト = 1)**

ACPI を有効/無効にする。

**親和性: <文字列>**

ゲスト・プロセスの実行に使用されるホスト・コアのリスト（例: 0,5,8-11

**エージェント:** [`enabled=]<1|0> [,freeze-fs-on-backup=<1|0>]  
[,fstrim_cloned_disks=<1|0>] [,type=<virtio|isa>].`

QEMU ゲストエージェントとそのプロパティとの通信を有効/無効にします。

**enabled=<boolean>** (デフォルト = 0)

VM 内で実行されている QEMU ゲストエージェント (QGA) との通信を有効/無効にします。

**freeze-fs-on-backup=<boolean>** (デフォルト = 1)

一貫性を保つために、バックアップ時にゲストファイルシステムをフリーズ/解凍します。

**fstrim\_cloned\_disks=<boolean>** (デフォルト = 0)

ディスクの移動またはVMの移行後にfstrimを実行する。

**type=<isa | virtio>** (デフォルト=virtio)

エージェントタイプを選択

**arch: <aarch64 | x86\_64>.**

仮想プロセッサーのアーキテクチャ。デフォルトはホスト。

**args: <文字列>**

kvmに渡される任意の引数、例: args: -no-reboot -

smbios type=0, vendor=FOO

---

### 注

このオプションはエキスパート専用です。

---

**audio0: device=<ich9-intel-hda|intel-hda|AC97> [, driver=<spice|none>] .**

QXL/Spiceと組み合わせて使用すると便利です。

**device=<AC97 | ich9-intel-hda | intel-hda>.**

オーディオデバイスを設定する。

**driver=<none | spice>** (デフォルト = spice)

オーディオデバイスのドライババックエンド。

**autostart: <boolean>** (デフォルト = 0)

クラッシュ後の自動再起動（現在は無視）。

---

## バルーン: <整数> (0 - N)

VMのターゲットRAMの量 (MiB)。ゼロを使用すると、バロン・ドライバが無効になります。

## bios:<ovmf | seabios> (デフォルト = seabios)

BIOSの実装を選択します。

## を起動します: [[*legacy*=]<[*acdn*] {1,4}>] ブート: [,*order*=<デバイス[;デバイス...]>]]

◦

ゲストの起動順序を指定します。*order*= サブプロパティを使用します。*key* または *legacy*= を指定しない使用法は非推奨です。

**legacy=<[acdn]> {1,4}> (デフォルト = cdn)**

フロッピー(a)、ハードディスク(c)、CD-ROM(d)、ネットワーク(n)で起動する。非推奨。代わりに *order=* を使う。

**order=<デバイス[,デバイス...]>とする。**

ゲストはここに表示されている順番でデバイスからのブートを試みます。

ディスク、光学ドライブ、バススルー・ストレージのUSBデバイスは直接起動し、NICはPXEをロードし、PCIeデバイスはディスクのように動作するか (NVMeなど) 、オプションROMをロードする (RAIDコントローラー、ハードウェアNICなど) 。

このリストにあるデバイスだけがブート可能としてマークされ、ゲストファームウェア (BIOS/UEFI) によってロードされることに注意してください。ブートに複数のディスクが必要な場合 (例: ソフトウェアレイド)、ここで全てのディスクを指定する必要があります。

指定された場合、非推奨の *legacy=[acdn]\** 値を上書きする。

**bootdisk: (ide|sata|scsi|virtio) \d+.**

指定したディスクからの起動を有効にする。非推奨：代わりに *boot: order=foo;bar* を使う。

**CDROM: <ボリューム>**

これは -ide2 オプションのエイリアスである。

**カスタム: [meta=<volume>] [,network=<volume>] [,user=<volume>]****[,vendor=<volume>]。**

cloud-init: 開始時に自動生成されるファイルを置き換えるカスタムファイルを指定する。

**meta=<ボリューム>**

cloud-init経由でVMに渡されるすべてのメタデータを含むカスタムファイルを指定する。これはプロバイダー固有で、configdrive2とnocloudは異なる。

**ネットワーク=<ボリューム>**

すべてのネットワークデータを含むカスタムファイルをcloud-init経由でVMに渡す。

**ユーザー=<ボリューム>**

全てのユーザーデータを含むカスタムファイルをcloud-init経由でVMに渡す。

**ベンダー=<ボリューム>**

すべてのベンダーデータを含むカスタムファイルをcloud-init経由でVMに渡す。

**cipassword: <文字列>**

cloud-init: ユーザーに割り当てるパスワード。一般的に、これを使うことは推奨されない。代わりにsshキ

ーを使ってください。また、cloud-initの古いバージョンではハッシュ化されたパスワードをサポートしていないことに注意してください。

**citype: <configdrive2 | nocloud | opennebula>**

cloud-initコンフィギュレーション・フォーマットを指定します。デフォルトは、設定されているオペレーティング・システムの種類 (ostype.Linuxではnocloud形式、Windowsではconfigdrive2を使用します。

**ciupgrade: <boolean> (デフォルト = 1)**

cloud-init: 初回起動後にパッケージの自動アップグレードを行います。

**ciuser: <文字列>**

cloud-init: イメージに設定されているデフォルトユーザーの代わりに、sshキーとパスワードを変更するユーザー名。

**cores: <整数> (1 - N) (デフォルト = 1)**

ソケットあたりのコア数。

**cpu: [[cputype=<string>] [,flags=<+FLAG[ ;-FLAG... [,flags=<+FLAG[ ;-FLAG... ]]>]] を指定します。**

**[,hidden=<1|0> [,hv-vendor-id=<vendor-id>] である。**

**[phys-bits=<8-64|host> [,reported-model=<enum> ]。**

エミュレートされたCPUタイプ。

**cputype=<string> (デフォルト = kvm64)**

エミュレートされたCPUタイプ。デフォルトまたはカスタム名（カスタムモデル名の前に *custom-* を付ける必要がある）。

**flags=<+FLAG[ ;-FLAG... ]>とする。**

で区切られた追加CPUフラグのリスト。フラグを有効にするには +FLAG を、無効にするには -FLAG を使用します。カスタム CPU モデルは、QEMU/KVM がサポートする任意のフラグを指定できます。VM 固有のフラグは、セキュリティ上の理由から以下のセットから選択する必要があります: pcid、spec-ctrl、ibpb、ssbd、virt-ssbd、amd-ssbd、amd-no-ssb、pdpe1gb、md-clear、hv-tlbflush、hv-evmcs、aes

**hidden=<boolean> (デフォルト = 0)**

KVM仮想マシンとして識別しないでください。

**hv-vendor-id=<ベンダーID>。**

Hyper-VのベンダーID。Windows ゲスト内のお部のドライバまたはプログラムは、特定の ID を必要とします。

**phys-bits=<8-64|ホスト>。**

ゲストOSに報告される物理メモリアドレスビット。ホストの値より小さいか等しくなければなりません。ホスト CPU の値を使用するにはホストに設定しますが、そうすると他の値を持つ

CPUへのライブマイグレーションが壊れることに注意してください。

reported-model=<486 | ブロードウェル | ブロードウェル-IBRS | ブロードウェル-IBRS  
Broadwell-notSX | Broadwell-notSX-IBRS | Cascadelake-Server |  
Cascadelake-Server-notSX | Cascadelake-Server-v2 | Cascadelake-  
Server-v4 | Cascadelake-Server-v5 | Conroe | Cooperlake |  
Cooperlake-v2 | EPYC | EPYC-Genoa | EPYC-IBPB | EPYC-Milan | EPYC-  
Milan-v2 | EPYC-Rome | EPYC-Rome-v2  
EPYC-Rome-v3 | EPYC-Rome-v4 | EPYC-v3 | EPYC-v4 | GraniteRapids  
| Haswell | Haswell-IBRS | Haswell-notSX | Haswell-notSX-IBRS |  
Icelake-Client | Icelake-Client-notSX | Icelake-Server | Icelake-  
Server-notSX | Icelake-Server-v3 | Icelake-Server-v4 | Icelake-  
Server-v5 | Icelake-Server-v6 | IvyBridge | アイビーブリッジ  
IvyBridge-IBRS | KnightsMill | Nehalem | ネハalem-IBRS | Opteron\_G1 |  
Opteron\_G2 | Opteron\_G3 | Opteron\_G4 | Opteron\_G5  
| Penryn | SandyBridge | SandyBridge-IBRS | SapphireRapids |  
SapphireRapids-v2 | Skylake-Client | Skylake-Client-IBRS |  
Skylake-Client-notSX-IBRS | Skylake-Client-v4 | Skylake-Server  
| Skylake-Server-IBRS | Skylake-Server-notSX-IBRS |  
Skylake-Server-v4 | Skylake-Server-v5 | Westmere | ウエ  
スミア  
Westmere-IBRS | athlon | core2duo | coreduo | host | kvm32 |  
kvm64 | max | pentium | pentium2 | pentium3 | phenom | qemu32 |  
qemu64> (デフォルト = kvm64)

ゲストに報告する CPU モデルとベンダー。QEMU/KVM がサポートするモデルである必要があります。カスタム CPU モデル定義に対してのみ有効で、デフォルトモデルは常にゲスト OS に自分自身を報告します。

**cpulimit:<数値> (0 - 128) (デフォルト = 0)**

CPU使用量の上限。

---

## 注

コンピュータに2つのCPUがある場合、合計2つのCPU時間を持つ。値0はCPU制限なしを示す。

---

**cputunits: <整数> (1 - 262144) (デフォルト = cgroup v1: 1024, cgroup v2: 100)**

VMのCPUウェイト。引数はカーネルのフェア・スケジューラで使用される。この数値が大きいほど、このVMはより多くのCPU時間を得ることになる。数値は、他のすべての実行中のVMのウェイトに対する相対値である。

### 説明: <文字列>

VM の説明。WebインターフェイスのVMのサマリーに表示される。これはコンフィギュレーション・ファイルのコメントとして保存される。

**efidisk0: [file=]<volume> [,efitype=<2m|4m>]  
[,format=<enum>] [,pre-enrolled-keys=<1|0>]  
[,size=<DiskSize>].**

EFI/バーを保存するディスクを設定する。

**efitype=<2m | 4m> (デフォルト = 2m)**

OVMF EFI/バーのサイズとタイプ。4mが新しく、推奨される。

ブート。後方互換性のため、特に指定がない場合は `2m` を使用する。`arch=aarch64` (ARM) の VMでは無視される。

#### **ファイル=<ボリューム**

ドライブのバックアップ・ボリューム。

**format=<cloop | cow | qcow | qcow2 | qed | raw | vmdk>**。

ドライブのバックアップファイルのデータ形式。

**pre-enrolled-keys=<boolean>** (デフォルト=0)

`efitype=4m` で使用する場合は、ディストリビューション固有のキーと Microsoft Standard キーが登録された am EFI vars テンプレートを使用する。この場合、デフォルトでセキュアブートが有効になりますが、VM 内からオフにすることも可能です。

**size=<ディスクサイズ**

ディスクサイズ。これは単なる情報提供であり、何の効果もない。

**freeze: <ブール値**

起動時にCPUをフリーズさせる (`c monitor`コマンドで実行開始)。

#### **フックスクリプト: <文字列**

vmsのライフタイムの様々なステップで実行されるスクリプト。

**hostpci[n]: [[host=<HOSTPCIID[,HOSTPCIID2...]>] [,deviceid=<hex id>] [,legacy-igd=<1|0>] [,mapping=<mapping-id[,device-id=<hex id>] [,legacy-igd=<1|0>] [,mapping=<mapping-id>] [,mdev=<string>] [,pcie=<1|0>] [,rombar=<1|0>] [,romfile=<string>] [,sub-device-id=<hex id>] [,sub-vendor-id=<hex id>] [,vendor-id=<hex id>] [,x-vga=<1|0> ]。**

ホストのPCIデバイスをゲストにマップする。

---

#### 注

このオプションは、ホストハードウェアへの直接アクセスを可能にする。そのため、このようなマシンの移行はもはや不可能である。

---



注意

実験的! ユーザーからこのオプションに関する問題が報告されました。

**デバイスID=<16進数ID>**

ゲストから見えるPCIデバイスIDを上書きする

**ホスト=<HOSTPCIID [ ;HOSTPCIID2... ]>。**

ホストPCIデバイス・パススルー。ホストのPCIデバイスのPCI ID、またはホストのPCI仮想機能のリスト。HOSTPCIID の構文は次のとおり：

*bus:dev.func* (16進数)

*lspci*コマンドを使えば、既存のPCIデバイスをリストアップ

できる。これかマッピング・キーのどちらかが設定されて

いなければならない。

**legacy-igd=<boolean> (デフォルト = 0)**

このデバイスをレガシーIGDモードで渡し、VMのプライマリかつ排他的なグラフィックデバイスにする。pc-i440fxマシンタイプとVGAが*none*に設定されている必要があります。

**マッピング=<マッピングID>**

クラスタ全体のマッピングのID。このホストかdefault-keyホストのどちらかを設定する必要があります。

**mdev=<文字列>**

使用する媒介デバイスのタイプ。このタイプのインスタンスはVMの起動時に作成され、VMの停止時にクリーンアップされる。

**pcie=<boolean> (デフォルト = 0)**

PCI-expressバスを選択する (q35マシンのモデルが必要)。

**rombar=<boolean> (デフォルト = 1)**

デバイスの ROM をゲストのメモリーマップに表示するかどうかを指定します。

**romfile=<文字列>**

カスタム pci デバイス rom ファイル名 (/usr/share/kvm/ にあること)。

**sub-device-id=<ヘックスID>**

ゲストから見えるPCIサブシステムのデバイスIDを上書きする

**サブベンダーID=<16進数ID>**

ゲストから見えるPCIサブシステムのベンダーIDを上書きする

**ベンダーID=<16進数ID>**

ゲストに見えるPCIベンダーIDを上書きする

**x-vga=<boolean> (デフォルト = 0)**

vfio-vgaデバイスのサポートを有効にする。

**hotplug:<文字列> (デフォルト = ネットワーク,ディスク,USB)**

ホットプラグ機能を選択的に有効にします。ネットワーク、ディスク、CPU、メモリ、USB、*cloudinit*。ホットプラグを完全に無効にするには0を使用します。値として1を使用すると、デフォルトの*network,disk,usb*のエイリアスになります。USB ホットプラグは、マシンのバージョンが >= のゲストで可能です。

7.1とostype l26またはwindows > 7.

**hugepages: <1024 | 2 | any>.**

ヒュッゲページ・メモリの有効／無効。

**ide[n]: [ファイル=<ボリューム> [,aio=<native|threads|io\_uring>]  
[,backup=<1|0>] [,bps=<bps>] [,bps\_max\_length=<seconds>]  
[,bps\_rd=<bps>] [,bps\_rd\_max\_length=<seconds>] [,bps\_wr=<bps>]  
[,bps\_wr\_max\_length=<seconds>] [,cache=<enum>] [,cyls=<整数値>]  
[,detect\_zeroes=<1|0>] [,discard=<ignore|on>] [,format=<enum>]  
[,heads=<integer>] [,iops=<iops>] [,iops\_max=<iops>]  
[,iops\_max\_length=<seconds>] [,iops\_rd=<iops>]  
[,iops\_rd\_max=<iops>] [,iops\_rd\_max\_length=<seconds>]  
[,iops\_wr=<iops>] [,iops\_wr\_max=<iops>]  
[,iops\_wr\_max\_length=<seconds>] [,mbps=<mbps>] [,mbps\_max=<mbps>]  
[,mbps\_rd=<mbps>] [,mbps\_rd\_max=<mbps>] [,mbps\_wr=<mbps>]  
[,mbps\_wr\_max=<mbps>] [,media=<cdrom|disk>] [,model=<model>]  
[,replicate=<1|0>] [,rerror=<ignore|report|stop>]  
[,secs=<integer>] [,serial=<serial>] [,shared=<1|0> ]  
[,size=<Disk>] [,size=<DiskSize>] [,snapshot=<1|0>] [,ssd=<1|0>]  
[,trans=<none|lba|auto>] [,werror=<enum>] [,wwn=<wwn>]** とする。  
ボリュームをIDEハードディスクまたはCD-ROMとして使用する (nは0~3)。

**aio=<io\_uring | ネイティブ | スレッド>。**

使用するAIOタイプ。

**backup=<ブール値>**

バックアップを作成する際にドライブを含めるかどうか。

**bps=<bps>**

最大r/w速度をバイト/秒で示す。

**bps\_max\_length=<秒>**

I/Oバーストの最大長 (秒)。

**bps\_rd=<bps>**

最大読み取り速度 (バイト毎秒)。

**bps\_rd\_max\_length=<秒>。**

読み取りI/Oバーストの最大長 (秒)。

**bps\_wr=<bps>**

最大書き込み速度 (バイト/秒)。

**bps\_wr\_max\_length=<秒>**

書き込みI/Oバーストの最大長 (秒)。

**キャッシング=<directsync | none | unsafe | writeback | writethrough>**

ドライブのキャッシングモード

**cyls=<整数>**

ドライブの物理ジオメトリを特定のシリンドラ数に強制する。

**detect\_zeroes=<ブール値>**

ゼロの書き込みを検出し、最適化を試みるかどうかを制御する。

**discard=<無視 | オン**

破棄／トリム要求を基礎となるストレージに渡すかどうかを制御する。

**ファイル=<ボリューム**

ドライブのバックアップ・ボリューム。

**format=<cloop | cow | qcow | qcow2 | qed | raw | vmdk>**

ドライブのバックアップファイルのデータ形式。

**heads=<整数**

ドライブの物理ジオメトリを特定のヘッド数に強制する。

**iops=<iops>**

r/w I/O の最大値（1秒あたりの操作回数）。

**iops\_max=<iops>**

スロットルされていないR/W I/Oプールの最大値（オペレーション/秒）。

**iops\_max\_length=<seconds>**

I/Oバーストの最大長（秒）。

**iops\_rd=<iops>**

1秒あたりの最大読み取りI/O。

**iops\_rd\_max=<iops>**

スロットルなしの最大読み取りI/Oプール（1秒あたりの処理数）。

**iops\_rd\_max\_length=<seconds>**

読み取りI/Oバーストの最大長（秒）。

**iops\_wr=<iops>**

最大書き込みI/O（オペレーション毎秒）。

**iops\_wr\_max=<iops>**

スロットルなしの最大書き込みI/Oプール（1秒あたりの処理数）。

**iops\_wr\_max\_length=<秒>**

書き込みI/Oバーストの最大長（秒）。

**mbps=<mbps>**

最大R/W速度（メガバイト/秒）。

**mbps\_max=<mbps>**

最大スロットルなしR/Wプール（メガバイト/秒）。

**mbps\_rd=<mbps>**  
最大読み取り速度（メガバイト/秒）。

**mbps\_rd\_max=<mbps>**  
スロットルのない最大読み取りプール（メガバイト/秒）。

**mbps\_wr=<mbps>**  
最大書き込み速度（メガバイト/秒）。

**mbps\_wr\_max=<mbps>**とする。  
スロットルなしの最大書き込みプール（メガバイト/秒）。

**media=<cdrom | disk>** (デフォルト = disk)  
ドライブのメディア・タイプ。

**model=<モデル>**  
ドライブのモデル名（URLエンコード、最大40バイト）。

**replicate=<boolean>** (デフォルト = 1)  
ドライブをレプリケーション・ジョブの対象とするかどうか。

**rerror=<無視 | 報告 | 停止>**  
読み取りエラー。

**secs=<整数>**  
ドライブの物理ジオメトリを特定のセクタ数に強制する。

**シリアル=<シリアル>**  
ドライブのシリアル番号（URLエンコード、最大20バイト）。

**shared=<boolean>** (デフォルト = 0)  
このローカル管理ボリュームをすべてのノードで使用可能なボリュームとしてマークします。



### 警告

このオプションは、ボリュームを自動的に共有するのではなく、すでに共有されていると仮定します！

**size=<ディスクサイズ>**  
ディスクサイズ。これは単なる情報提供であり、何の効果もない。

**snapshot=<boolean>**  
qemu のスナップショットモード機能を制御します。有効な場合、ディスクに加えられた変更は一時的なものであり、VM がシャットダウンされると破棄されます。

**ssd=<ブール値>**  
このドライブを回転式ハードディスクではなくSSDとして公開するかどうか。

**trans=<自動 | lba | なし>**  
ディスク・ジオメトリ・バイオスのトランスレーション・モードを強制する。

**werror=<enospc | 無視 | 報告 | 停止>。**

書き込みエラー。

**wwn=<wwn>**

ドライブのワールドワイド名。16バイトの16進文字列でエンコードされ、先頭に0xが付く。

**ipconfig[n]: [gw=<GatewayIPv4>] [,gw6=<GatewayIPv6>]**

**[,ip=<IPv4Format/CIDR>] [,ip6=<IPv6Format/CIDR>]** とする。

cloud-init: 対応するインターフェースのIPアドレスとゲートウェイを指定する。

IPアドレスはCIDR表記を使用し、ゲートウェイはオプションだが、同じタイプのIPを指定する必要がある。

特別な文字列`dhcp`は、DHCPを使用するIPアドレスに使用でき、その場合、明示的な ゲートウェイを指定する必要はない。IPv6の場合、ステートレス自動設定を使用するには、特別な文字列`auto`を使用できる。これにはcloud-init 19.4以降が必要です。

cloud-initが有効で、IPv4アドレスもIPv6アドレスも指定されていない場合、デフォルトでIPv4の`dhcp`を使用する。

#### **gw=<ゲートウェイ IPv4**

IPv4トラフィックのデフォルトゲートウェイ。

---

##### **注**

必要なオプション: `ip`

---

#### **gw6=<ゲートウェイ IPv6**

IPv6トラフィックのデフォルトゲートウェイ。

---

##### **注**

必要なオプション: `ip6`

---

#### **ip=<IPv4Format/CIDR> (デフォルト = dhcp)**

CIDR形式のIPv4アドレス。

#### **ip6=<IPv6Format/CIDR> (デフォルト = dhcp)**

CIDR形式のIPv6アドレス。

#### **ivshmem: size=<整数> [,name=<文字列>]。**

VM間共有メモリ。VM間やホストとの直接通信に便利。

#### **name=<文字列**

ファイル名。先頭に `pve-shm-` が付く。デフォルトは VMID です。VM の停止時に削除されます

。

#### **size=<integer> (1 - N)**

---

MB単位のファイルサイズ。

**keephugepages: <boolean> (デフォルト = 0)**

hugepagesと併用する。有効にすると、VMシャットダウン後もhugepagesは削除されず、その後の起動に使用できます。

**キーボード: <da | de | de-ch | en-gb | en-us | es | fi | fr | fr-be | fr-ca | fr-ch | hu | is | it | ja | lt | mk | nl | no | pl | pt | pt-br | sl | sv | tr>.**

VNCサーバーのキーボードレイアウト。このオプションは一般的には必要なく、ゲストOS内で処理した方が良い場合が多い。

**kvm: <boolean> (デフォルト = 1)**

KVMハードウェア仮想化の有効/無効。

**localtime: <ブール値>**

リアルタイムクロック (RTC) をローカルタイムに設定する。これはデフォルトで有効である。

Microsoft Windows OS。

**lock: <バックアップ | クローン | 作成 | マイグレーション | ロールバック | スナップショット | スナップショット・デリート | サスペンド | サスPEND>。**

VMをロック/アンロックする。

**マシン[タイプ=]<マシンタイプ>]。 [viommu=<intel | virtio>]である。**

QEMUマシンを指定する。

**type=<マシン・タイプ>**

QEMUのマシンタイプを指定します。

**viommu=<インテル | ヴィルティオ>。**

ゲストのvIOMMUバリエントを有効にして設定する (Intel vIOMMUはマシンタイプとしてq35を設定する必要があります)。

**メモリを使用する: [カレント=]<整数>**

メモリ特性。

**current=<integer> (16 - N) (デフォルト = 512)**

VM の現在のオンライン RAM 量 (MiB 単位)。これは、バルーン デバイスを使用する際に使用可能な最大メモリです。

**migrate\_downtime: <数値> (0 - N) (デフォルト = 0.1)**

マイグレーションの最大許容ダウンタイム (秒) を設定します。

**migrate\_speed: <整数> (0 - N) (デフォルト = 0)**

マイグレーションの最大速度 (MB/s) を設定する。値0は制限なし。

**name: <文字列>**

VM の名前を設定します。コンフィギュレーション・ウェブ・インターフェイスでのみ使用されます。

## ネームサーバー: <文字列>

cloud-init: コンテナの DNS サーバー IP アドレスを設定する。searchdomain も nameserver も設定されていない場合、Create はホストからの設定を自動的に使用する。

```
net[n]: [model=<enum> [,bridge=<bridge>] [,firewall=<1|0>]  
[,link_down=<1|0>] [,macaddr=<XX:XX:XX:XX:XX: XX>] [,mtu=<integer>]  
[,queues=<integer>] [,rate=<number>] [,tag=<integer>]  
[,trunks=<vlanid[;vlanid...]>] [,<model>=<macaddr>] [,<model>=<macaddr>]  
,<model>=<macaddr>  
ネットワーク機器を指定する。
```

## ブリッジ

ネットワークデバイスを接続するブリッジ。Proxmox VE標準ブリッジはvmbr0と呼ばれます。

ブリッジを指定しない場合は、DHCPとDNSサービスを提供するkvmユーザー（NAT）ネットワークデバイスを作成します。以下のアドレスが使用されます：

- 10.0.2.2 ゲートウェイ
- 10.0.2.3 DNSサーバー
- 10.0.2.4 SMBサーバー

DHCPサーバーは10.0.2.15から始まるアドレスをゲストに割り当てる。

## ファイアウォール=<ブール>

このインターフェイスをファイアウォールで保護するかどうか。

## link\_down=<ブール値>

このインターフェイスを切断すべきかどうか（プラグを抜くように）。

## macaddr=<XX:XX:XX:XX:XX:XX>。

I/G (Individual/Group) ビットが設定されていない共通のMACアドレス。

model=<e1000 | e1000-82540em | e1000-82544gc | e1000-82545em |  
e1000e | i82551 | i82557b | i82559er | ne2k\_isa | ne2k\_pci | pcnet |  
rtl8139 | virtio | vmxnet3>となります。

ネットワークカードモデル。*virtio* モデルは、非常に低いCPUオーバーヘッドで最高のパフォーマンスを提供します。ゲストがこのドライバをサポートしていない場合、通常は*e1000*を使用するのがベストです。

## mtu=<整数> (1 - 65520)

MTUを強制します。ブリッジMTUを使用するには1を設定します

## queues=<整数> (0 - 64)

デバイスで使用するパケット・キューの数。

## rate=<数値> (0 - N)

mbps (メガバイト/秒) 単位の浮動小数点数でのレート制限。

## タグ=<整数> (1 - 4094)

このインターフェイスのパケットに適用するVLANタグ。

**trunks=<vlanid[;vlanid...]>とする。**  
このインターフェイスを通過するVLANトランク。

**numa: <ブール値> (デフォルト = 0)**

NUMAの有効/無効。

**numa[n]: cpus=<id[-id];...> [,hostnodes=<id[-id];...>]  
[,policy=<優先|バインド|インターリーブ>] [,policy=<優先|バイ  
ンド|インターリーブ[,memory=<number>]  
[,policy=<preferred|bind|interleave>] とする。**

NUMAトポロジー。

**cpus=<id[-id];...>とする。**

このNUMAノードにアクセスするCPU。

**hostnodes=<id[-id] ; ...>**とする。  
使用するホストNUMAノード。

### メモリ=<数字>

このNUMAノードが提供するメモリ量。

**policy=<バインド | インターリーブ | プリファード>**。

NUMA割り当てポリシー。

**onboot: <boolean>** (デフォルト = 0)

システム起動時にVMを起動するかどうかを指定する。

**ostype: <l24 | l26 | other | solaris | w2k | w2k3 | w2k8 | win10 | win11 | win7 | win8 | vvista | wxp>**.

ゲストオペレーティングシステムを指定します。これは、特定のオペレーティングシステムに対する特別な最適化/機能を有効にするために使用されます：

その他 不特定OS

wxpマイクロソフト・ウィンドウズXP

w2kマイクロソフト・ウィンドウズ2000

w 2k3マイクロソフト・ウィンドウズ2003

w 2k8マイクロソフト・ウィンドウズ2008

vvistaマイクロソフト・ウィンドウズ・ビスタ

win7マイクロソフト・ウィンドウズ7

win8Microsoft Windows 8/2012/2012r2

win10Microsoft Windows 10/2016/2019

win11マイクロソフト・ウィンドウズ 11/2022/2025

l24Linux 2.4 カーネル

l26Linux 2.6 - 6.X カーネル

ソラリス

Solaris/OpenSolaris/OpenIndianaカーネル

**parallel[n]: /dev/parportd+ | /dev/usb/lpd+**

ホストパラレルデバイスをマップする (nは0~2)。

---

## 注

このオプションは、ホストハードウェアへの直接アクセスを可能にする。そのため、このようなマシンの移行はもはや不可能である。

---



### 注意

実験的！ ユーザーからこのオプションに関する問題が報告されました。

---

#### **protection: <ブール値> (デフォルト = 0)**

VM の保護フラグを設定する。これにより、VMの削除とディスクの削除操作が無効になる。

#### **reboot: <boolean> (デフォルト = 1)**

再起動を許可する。0に設定すると、リブート時にVMが終了する。

**rng0: [ソース=]</dev/urandom|/dev/random|/dev/hwrng>**  
[,**max\_bytes=<integer>**] [,**period=<integer>**]  
VirtIO ベースの乱数ジェネレータを設定します。

#### **max\_bytes=<整数> (デフォルト = 1024)**

ミリ秒ごとにゲストに注入されるエントロピーの最大バイト数。ソースとして */dev/random* を使用する場合は低い値を推奨します。制限を無効にするには 0 を使用します（潜在的に危険です！）。

#### **period=<integer> (デフォルト=1000)**

ミリ秒ごとにエントロピー注入クォータがリセットされ、ゲストは他の最大バイト数のエントロピーを取得できるようになります。

#### **source=</dev/hwrng | /dev/random | /dev/urandom>。**

エントロピーを収集するホスト上のファイル。ほとんどの場合、*/dev/urandom* は */dev/random* を使用することで、ホスト上でのエントロピー枯渇の問題を避けることができます。*urandom*を使っても、実際のエントロピーのシードであることに変わりはなく、提供されるバイトはゲスト上でも実際のエントロピーと混合される可能性が高いので、意味のある方法でセキ

---

ユリティを低下させることはできません。`/dev/hwrng` はホストからハードウェアRNGを渡すために使うことができます。

```
sata[n]: [ファイル=<ボリューム> [,aio=<native|threads|io_uring>]
[,backup=<1|0>] [,bps=<bps>] [,bps_max_length=<seconds>]
[,bps_rd=<bps>] [,bps_rd_max_length=<seconds>] [,bps_wr=<bps>]
[,bps_wr_max_length=<seconds>] [,cache=<enum>] [,cyls=<整数値>]
[,detect_zeroes=<1|0>] [,discard=<ignore|on>] [,format=<enum>]
[,heads=<integer>] [,iops=<iops>] [,iops_max=<iops>]
[,iops_max_length=<seconds>] [,iops_rd=<iops>]
[,iops_rd_max=<iops>] [,iops_rd_max_length=<秒>] [,iops_wr=<iops>]
[,iops_wr_max=<iops>] [,iops_wr_max_length=<seconds>]
[,mbps=<mbps>] [,mbps_max=<mbps>] [,mbps_rd=<mbps>]
[,mbps_rd_max=<mbps>] [,mbps_wr=<mbps>] [,mbps_wr_max=<mbps>]
[,media=<cdrom|disk> ]。] [,replicate=<1|0>]
[,rerror=<ignore|report|stop>] [,secs=<integer>]
[,serial=<serial>] [,shared=<1|0>] [,size=<DiskSize>]
[,snapshot=<1|0>] [,ssd=<1|0>] [,trans=<none|lba|auto>]
[,werror=<enum>] [,wwn=<wwn>] とする。
```

ボリュームをSATAハードディスクまたはCD-ROMとして使用します（nは0～5）。

**aio=<io\_uring | ネイティブ | スレッド>**。

使用するAIOタイプ。

**backup=<ブール値>**

バックアップを作成する際にドライブを含めるかどうか。

**bps=<bps>**

最大r/w速度をバイト/秒で示す。

**bps\_max\_length=<秒>**

I/Oバーストの最大長（秒）。

**bps\_rd=<bps>**

最大読み取り速度（バイト毎秒）。

**bps\_rd\_max\_length=<秒>**。

読み取りI/Oバーストの最大長（秒）。

**bps\_wr=<bps>**

最大書き込み速度（バイト/秒）。

**bps\_wr\_max\_length=<秒>**

書き込みI/Oバーストの最大長（秒）。

**キャッシュ=<directsync | none | unsafe | writeback | writethrough>**

ドライブのキャッシュモード

**cyls=<整数>**

ドライブの物理ジオメトリを特定のシリンドラ数に強制する。

**detect\_zeroes=<ブール値>**

ゼロの書き込みを検出し、最適化を試みるかどうかを制御する。

**discard=<無視 | オン**

破棄／トリム要求を基礎となるストレージに渡すかどうかを制御する。

**ファイル=<ボリューム**

ドライブのバックアップ・ボリューム。

**format=<cloop | cow | qcow | qcow2 | qed | raw | vmdk>**

ドライブのバックアップファイルのデータ形式。

**heads=<整数**

ドライブの物理ジオメトリを特定のヘッド数に強制する。

**iops=<iops>**

r/w I/O の最大値（1秒あたりの操作回数）。

**iops\_max=<iops>**

スロットルされていないR/W I/Oプールの最大値（オペレーション/秒）。

**iops\_max\_length=<seconds>**

I/Oバーストの最大長（秒）。

**iops\_rd=<iops>**

1秒あたりの最大読み取りI/O。

**iops\_rd\_max=<iops>**

スロットルなしの最大読み取りI/Oプール（1秒あたりの処理数）。

**iops\_rd\_max\_length=<seconds>**

読み取りI/Oバーストの最大長（秒）。

**iops\_wr=<iops>**

最大書き込みI/O（オペレーション毎秒）。

**iops\_wr\_max=<iops>**

スロットルなしの最大書き込みI/Oプール（1秒あたりの処理数）。

**iops\_wr\_max\_length=<秒>**

書き込みI/Oバーストの最大長（秒）。

**mbps=<mbps>**

最大R/W速度（メガバイト/秒）。

**mbps\_max=<mbps>**

最大スロットルなしR/Wプール（メガバイト/秒）。

**mbps\_rd=<mbps>**

最大読み取り速度（メガバイト/秒）。

**mbps\_rd\_max=<mbps>**

スロットルのない最大読み取りプール（メガバイト/秒）。

**mbps\_wr=<mbps>**

最大書き込み速度（メガバイト/秒）。

**mbps wr max=<mbps>**とする。  
スロットルなしの最大書き込みプール（メガバイト/秒）。

**media=<cdrom | disk>** (デフォルト = disk)

ドライブのメディア・タイプ。

**replicate=<boolean>** (デフォルト = 1)

ドライブをレプリケーション・ジョブの対象とするかどうか。

**rerror=<無視 | 報告 | 停止>**。

読み取りエラー。

**secs=<整数>**

ドライブの物理ジオメトリを特定のセクタ数に強制する。

**シリアル=<シリアル>**

ドライブのシリアル番号（URLエンコード、最大20バイト）。

**shared=<boolean>** (デフォルト = 0)

このローカル管理ボリュームをすべてのノードで使用可能なボリュームとしてマークします。



### 警告

このオプションは、ボリュームを自動的に共有するのではなく、すでに共有されていると仮定します！

**size=<ディスクサイズ>**

ディスクサイズ。これは単なる情報提供であり、何の効果もない。

**snapshot=<boolean>**

qemu のスナップショットモード機能を制御します。有効な場合、ディスクに加えられた変更は一時的なものであり、VM がシャットダウンされると破棄されます。

**ssd=<ブール値>**

このドライブを回転式ハードディスクではなくSSDとして公開するかどうか。

**trans=<自動 | lba | なし>**。

ディスク・ジオメトリ・バイオスのトランスレーション・モードを強制する。

**werror=<enospc | 無視 | 報告 | 停止>**。

書き込みエラー。

**wwn=<wwn>**

ドライブのワールドワイド名。16バイトの16進文字列でエンコードされ、先頭に0xが付く。

```
scsi[n]: [ファイル=<ボリューム> [,aio=<ネイティブ|スレッド|io_uring>]
[,backup=<1|0>] [,bps=<bps>] [,bps_max_length=<seconds>]
[,bps_rd=<bps>] [,bps_rd_max_length=<seconds>] [,bps_wr=<bps>]
[,bps_wr_max_length=<seconds>] [,cache=<enum>] [,cyls=<integer>]
[,detect_zeroes=<1|0>] [,discard=<ignore|on>] [,format=<enum>]
[,heads=<integer>] [,iops=<iops>] [,iops_max=<iops>]
[,iops_max_length=<秒>] [,iops_rd=<iops>] [,iops_rd_max=<iops>]
[,iops_rd_max_length=<seconds>] [,iops_wr=<iops>]
[,iops_wr_max=<iops>] [,iops_wr_max_length=<seconds>]
[,iothread=<1|0>] [,mbps=<mbps>] [,mbps_max=<mbps>]
[,mbps_rd=<mbps>] [,mbps_rd_max=<mbps>] [,mbps_wr=<mbps>]
[,mbps_wr_max=<mbps>] [,media=<cdrom|disk>] [,product=<product>]
[,queues=<integer>] [,replicate=<1|0>]
[,rerror=<ignore|report|stop>] [,ro=<1|0>] [,scsiblock=<1|0>]
[,secs=<integer>] [,serial=<serial>] [,shared=<1|0>]
[,size=<DiskSize>] [,snapshot=<1|0>] [,ssd=<1|0>]
[,trans=<none|lba|auto>] [,vendor=<vendor>] [,werror=<enum>]
[,wwn=<wwn>] となる。
```

ボリュームをSCSIハードディスクまたはCD-ROMとして使用する (nは0~30)。

**aio=<io\_uring | ネイティブ | スレッド>。**

使用するAIOタイプ。

**backup=<ブール値>**

バックアップを作成する際にドライブを含めるかどうか。

**bps=<bps>**

最大r/w速度をバイト/秒で示す。

**bps\_max\_length=<秒>**

I/Oバーストの最大長 (秒)。

**bps\_rd=<bps>**

最大読み取り速度 (バイト毎秒)。

**bps\_rd\_max\_length=<秒>。**

読み取りI/Oバーストの最大長 (秒)。

**bps\_wr=<bps>**

最大書き込み速度 (バイト/秒)。

**bps\_wr\_max\_length=<秒>**

書き込みI/Oバーストの最大長（秒）。

**キャッシュ=<directsync | none | unsafe | writeback | writethrough>**。

ドライブのキャッシュモード

**cyls=<整数**

ドライブの物理ジオメトリを特定のシリンドラ数に強制する。

**detect\_zeroes=<ブール値>**

ゼロの書き込みを検出し、最適化を試みるかどうかを制御する。

**discard=<無視 | オン>**

破棄／トリム要求を基礎となるストレージに渡すかどうかを制御する。

**ファイル=<ボリューム>**

ドライブのバックアップ・ボリューム。

**format=<cloop | cow | qcow | qcown | qed | raw | vmdk>。**

ドライブのバックアップファイルのデータ形式。

**heads=<整数>**

ドライブの物理ジオメトリを特定のヘッド数に強制する。

**iops=<iops>**

r/w I/Oの最大値（1秒あたりの操作回数）。

**iops\_max=<iops>**

スロットルされていないR/W I/Oプールの最大値（オペレーション/秒）。

**iops\_max\_length=<seconds>**

I/Oバーストの最大長（秒）。

**iops\_rd=<iops>**

1秒あたりの最大読み取りI/O。

**iops\_rd\_max=<iops>**

スロットルなしの最大読み取りI/Oプール（1秒あたりの処理数）。

**iops\_rd\_max\_length=<seconds>**

読み取りI/Oバーストの最大長（秒）。

**iops\_wr=<iops>**

最大書き込みI/O（オペレーション毎秒）。

**iops\_wr\_max=<iops>。**

スロットルなしの最大書き込みI/Oプール（1秒あたりの処理数）。

**iops\_wr\_max\_length=<秒>**

書き込みI/Oバーストの最大長（秒）。

**iothread=<ブール値>**

このドライブにiothreadsを使用するかどうか

**mbps=<mbps>**

最大R/W速度（メガバイト/秒）。

**mbps\_max=<mbps>**

最大スロットルなしR/Wプール（メガバイト/秒）。

**mbps\_rd=<mbps>**

最大読み取り速度（メガバイト/秒）。

**mbps\_rd\_max=<mbps>**  
スロットルのない最大読み取りプール（メガバイト/秒）。

**mbps\_wr=<mbps>**  
最大書き込み速度（メガバイト/秒）。

**mbps\_wr\_max=<mbps>とする。**  
スロットルなしの最大書き込みプール（メガバイト/秒）。

**media=<cdrom | disk> (デフォルト=disk)**  
ドライブのメディア・タイプ。

**product=<商品名>**  
ドライブの製品名（最大16バイト）。

**キュー=<整数> (2 - N)**  
キューの数。

**replicate=<boolean> (デフォルト=1)**  
ドライブをレプリケーション・ジョブの対象とするかどうか。

**rerror=<無視 | 報告 | 停止>**  
読み取りエラー。

**ro=<ブール値>**  
ドライブが読み取り専用かどうか。

**scsiblock=<boolean> (デフォルト=0)**  
ホストブロックデバイスのフルパススルーにscsi-blockを使うかどうか



### 警告

ホストのメモリ不足やメモリの断片化が進むと、I/Oエラーが発生することがある。

**secs=<整数>**

ドライブの物理ジオメトリを特定のセクタ数に強制する。

**シリアル=<シリアル>**  
ドライブのシリアル番号（URLエンコード、最大20バイト）。

**shared=<boolean> (デフォルト=0)**

このローカル管理ボリュームをすべてのノードで使用可能なボリュームとしてマークします。

**警告**

このオプションは、ボリュームを自動的に共有するのではなく、すでに共有されていると仮定します！

---

**size=<ディスクサイズ>**

ディスクサイズ。これは単なる情報提供であり、何の効果もない。

**snapshot=<boolean>**

qemu のスナップショットモード機能を制御します。有効な場合、ディスクに加えられた変更は一時的なものであり、VM がシャットダウンされると破棄されます。

**ssd=<ブール値>**

このドライブを回転式ハードディスクではなくSSDとして公開するかどうか。

**trans=<自動 | lba | なし>。**

ディスク・ジオメトリ・バイオスのトランスレーション・モードを強制する。

**ベンダー**

ドライブのベンダー名（最大8バイト）。

**werror=<enospc | 無視 | 報告 | 停止>。**

書き込みエラー。

**wwn=<wwn>**

ドライブのワールドワイド名。16バイトの16進文字列でエンコードされ、先頭に0xが付く。

**scsihw:<lsi | lsi53c810 | megasas | pvscsi | virtio-scsi-pci  
| virtio-scsi-single> (デフォルト = lsi)**  
SCSIコントローラモデル

**searchdomain:<文字列>**

cloud-init: コンテナのDNS検索ドメインを設定する。searchdomainもnameserverも設定されていない場合、Createはホストからの設定を自動的に使用する。

**serial[n]: (/dev/.+|socket)**

VM内にシリアル・デバイスを作成し（nは0～3）、ホストのシリアル・デバイス（例：/dev/ttys0）を通すか、ホスト側にunixソケットを作成する（qm terminalを使ってターミナル接続を開く）。

---

**注**

ホストシリアルデバイスを経由する場合、そのようなマシンの移行はもはや不可能です。

---

**注意**

実験的！ ユーザーからこのオプションに関する問題が報告されました。

---

**shares: <整数> (0 - 50000) (デフォルト=1000)**

オート・バルーニングのメモリ・シェア量。この数値が大きいほど、このVMはより多くのメモリを獲得する。数値は、他のすべての実行中のVMの重みに対する相対値です。ゼロを使用すると、オート・バルーニングは無効になる。オート・バルーニングはpvestatdによって実行される。

**smbios1: [base64=<1|0>] [,family=<Base64 エンコードされた文字列>] [,manufacturer=<Base64 エンコードされた文字列>] [,product=<Base64 エンコードされた文字列>] [,serial=<Base64 エンコードされた文字列>] [,sku=<Base64 エンコードされた文字列>] [,uuid=<UUID>] [,version=<Base64 エンコードされた文字列>].**

SMBIOSタイプ1のフィールドを指定する。

**base64=<ブール値>**

SMBIOSの値がbase64エンコードされていることを示すフラグ

**family=<Base64エンコードされた文字列>。**

SMBIOS1 ファミリ文字列を設定する。

**メーカー=<Base64エンコード文字列>**

SMBIOS1 メーカーを設定する。

**product=<Base64でエンコードされた文字列>**

SMBIOS1のプロダクトIDを設定する。

**serial=<Base64でエンコードされた文字列>**

SMBIOS1のシリアル番号を設定する。

**sku=<Base64エンコードされた文字列>**

SMBIOS1 SKU文字列を設定する。

**uuid=<UUID>**

SMBIOS1のUUIDを設定する。

**バージョン=<Base64エンコード文字列>**

SMBIOS1のバージョンを設定する。

**smp: <整数> (1 - N) (デフォルト = 1)**

CPU数。代わりに -sockets オプションを使用してください。

**sockets: <整数> (1 - N) (デフォルト = 1)**

CPUソケットの数。

**spice\_enhancements: [foldersharing=<1|0>]**

[,videostreaming=<off|all|filter>]。

SPICE の追加機能拡張を設定する。

**foldersharing=<boolean> (デフォルト = 0)**

SPICE 経由でのフォルダ共有を有効にします。VMにSpice-WebDAVデーモンがインストールされている必要があります。

**videostreaming=<all | filter | off> (デフォルト = off)**

ビデオストリーミングを有効にします。検出されたビデオストリームに圧縮を使用する。

### **sshkeys: <文字列**

cloud-init: 公開SSH鍵を設定する（1行に1つの鍵、OpenSSH形式）。

**startdate: (now | YYYY-MM-DD | YYYY-MM-DDTHH:MM:SS) (デフォルト = now)**

リアルタイムクロックの初期日付を設定する。有効な日付の書式は、'now' または 2006-06-17T16:01:21 または  
2006-06-17.

**を起動する: order=] [,up=] [,down=] `)**

スタートアップとシャットダウンの動作。Orderは、一般的な起動順序を定義する非負の数値である。

シャットダウンは逆の順序で行われます。さらに、次のVMが起動または停止するまでの待ち時間を秒単位で設定できます。

**tablet: <ブール値> (デフォルト = 1)**

USBタブレットデバイスを有効/無効にする。このデバイスは通常、VNCで絶対的なマウス位置決めを可能にするために必要である。そうしないと、マウスが通常のVNCクライアントと同期しなくなります。1つのホスト上でコンソールのみのゲストを多数実行している場合は、コンテキストスイッチの数を減らすためにこれを無効にすることを検討してもよいでしょう。spice (qm set <vmid> --vga qxl)を使用すると、これはデフォルトでオフになります。

**タグ:<string**

VMのタグ。これはメタ情報に過ぎない。

**tdf: <ブール値> (デフォルト = 0)**

タイムドリフト修正の有効／無効。

**テンプレート:<boolean> (デフォルト = 0)**

テンプレートの有効/無効。

**tpmstate0: [ファイル=<ボリューム> [, サイズ=<ディスクサイズ>] [, バージョン=<v1.2|v2.0>]。**

TPMの状態を保存するDiskを設定する。フォーマットはraw固定。

**ファイル=<ボリューム>**

ドライブのバックアップ・ボリューム。

**size=<ディスクサイズ>**

ディスクサイズ。これは単なる情報提供であり、何の効果もない。

**バージョン=<v1.2 | v2.0> (デフォルト=v2.0)**

TPMインターフェイスのバージョン。v2.0の方が新しいので、そちらを優先すべきである。これは後で変更できないことに注意。

**unused[n] である: [ファイル=<ボリューム**

未使用ボリュームへの参照。これは内部的に使用されるので、手動で変更してはならない。

**ファイル=<ボリューム**

ドライブのバックアップ・ボリューム。

**usb[n]: [[host=]<HOSTUSBDEVICE|spice>] [,mapping=<mappingid>]**

**[,usb3=<1|0[,mapping=<mapping-id>] [,usb3=<1|0>]** のようになります。

USBデバイスを設定する (nは0~4、マシンバージョン7.1以上、ostype l26またはwindows 7以上の場合、nは14まで)。

**host=<HOSTUSBDEVICE | spice>**

ホスト USB デバイスまたはポート、または値のスパイス。HOSTUSBDEVICE 構文は以下のとおり：

'bus-port(.port)\*' (10進数) または  
'vendor\_id:product\_id' (16進数) または'spice'

*lsusb -t*コマンドを使えば、既存のusbデバイスをリストアップできる。

---

### 注

このオプションは、ホストハードウェアへの直接アクセスを可能にする。そのため、このようなマシンの移行はもはや不可能である。

---

*spice*の値は、spice用のusbリダイレクトデバイスを追加するために使用できます。これがマッピングキーのどちらかを設定する必要があります。

### マッピング=<マッピングID

クラスタ全体のマッピングのID。このホストかdefault-keyホストのどちらかを設定する必要があります。

**usb3=<boolean> (デフォルト = 0)**

与えられたホストオプションが USB3 デバイスかポートかを指定します。最近のゲスト (マシンのバージョン >= 7.1 および ostype l26 と Windows > 7) では、このフラグは無関係です (すべてのデバイスは xhci コントローラに接続されています)。

**vcpus: <整数> (1 - N) (デフォルト = 0)**

ホットプラグされたVCPUの数。

**vga: [[type=<enum>] [[クリップボード=<vnc>] [[メモリ=<integer>]] ]。[クリップボード=<vnc>] [, メモリ=<整数>]**

VGAハードウェアを設定する。高解像度モード (>= 1280x1024x16) を使用したい場合は、VGAメモリオプションを増やす必要があるかもしれません。QEMU 2.9以降、デフォルトのVGAディスプレイタイプは、*cirrus*を使用する一部のWindowsバージョン (XPおよびそれ以前) 以外のすべてのOSタイプで標準となっています。*qxl*オプションはSPICEディスプレイサーバーを有効にします。Win\* OS では

、いくつの独立したディスプレイが欲しいかを選択できます。また、グラフィックカードなしで、シリアルデバイスをターミナルとして使用して実行することもできます。

### **クリップボード=<vnc>**

特定のクリップボードを有効にする。設定されていない場合、ディスプレイの種類に応じて SPICE のものが追加されます。VNCクリップボードとの移行はまだサポートされていません！

### **メモリ=<整数> (4 - 512)**

VGA メモリ (MiB) を設定します。シリアル表示には影響しません。

```
type=<cirrus | none | qxl | qxl2 | qxl3 | qxl4 | serial0  
| serial1 | serial2 | serial3 | std | virtio | virtio-gl  
| vmware> (デフォルト=std)
```

VGAタイプを選択します。

```
virtio[n]: [file=<volume> [,aio=<native|threads|io_uring>]
[,backup=<1|0>] [,bps=<bps>] [,bps_max_length=<seconds>]
[,bps_rd=<bps>] [,bps_rd_max_length=<seconds>] [,bps_wr=<bps>]
[,bps_wr_max_length=<seconds>] [,cache=<enum>] [,cyls=<整数値>]
[,detect_zeroes=<1|0>] [,discard=<ignore|on>] [,format=<enum>]
[,heads=<integer>] [,iops=<iops>] [,iops_max=<iops>]
[,iops_max_length=<seconds>] [,iops_rd=<iops>]
[,iops_rd_max=<iops>] [,iops_rd_max_length=<秒>]
[,iops_wr=<iops>] [,iops_wr_max=<iops>]
[,iops_wr_max_length=<seconds>] [,iothread=<1|0>] [,mbps=<mbps>]
[,mbps_max=<mbps>] [,mbps_rd=<mbps>] [,mbps_rd_max=<mbps>]
[,mbps_wr=<mbps>] [,mbps_wr_max=<mbps>]。]
[,media=<cdrom|disk>] [,replicate=<1|0>]
[,rerror=<ignore|report|stop>] [,ro=<1|0>] [,secs=<integer>]
[,serial=<serial>] [,shared=<1|0>] [,size=<DiskSize>]
[,snapshot=<1|0>] [,trans=<none|lba|auto>] [,werror=<enum>] とする。
```

ボリュームをVIRTIOハードディスクとして使用する（nは0～15）。

**aio=<io\_uring | ネイティブ | スレッド>。**

使用するAIOタイプ。

**backup=<ブール値>**

バックアップを作成する際にドライブを含めるかどうか。

**bps=<bps>**

r/wの最大速度（バイト/秒）。

**bps\_max\_length=<秒>**

I/Oバーストの最大長（秒）。

**bps\_rd=<bps>**

最大読み取り速度（バイト毎秒）。

**bps\_rd\_max\_length=<秒>。**

読み取りI/Oバーストの最大長（秒）。

**bps\_wr=<bps>**

最大書き込み速度（バイト/秒）。

**bps\_wr\_max\_length=<秒>**

書き込みI/Oバーストの最大長（秒）。

**キャッシュ=<directsync | none | unsafe | writeback | writethrough>**

ドライブのキャッシュモード

**cyls=<整数>**

ドライブの物理ジオメトリを特定のシリンドラ数に強制する。

**detect\_zeroes=<ブール値>**

ゼロの書き込みを検出し、最適化を試みるかどうかを制御する。

**discard=<無視 | オン**

破棄／トリム要求を基礎となるストレージに渡すかどうかを制御する。

**ファイル=<ボリューム**

ドライブのバックアップ・ボリューム。

**format=<cloop | cow | qcow | qcow2 | qed | raw | vmdk>**

ドライブのバックアップファイルのデータ形式。

**heads=<整数**

ドライブの物理ジオメトリを特定のヘッド数に強制する。

**iops=<iops>**

r/w I/O の最大値（1秒あたりの操作回数）。

**iops\_max=<iops>**

スロットルされていないR/W I/Oプールの最大値（オペレーション/秒）。

**iops\_max\_length=<seconds>**

I/Oバーストの最大長（秒）。

**iops\_rd=<iops>**

1秒あたりの最大読み取りI/O。

**iops\_rd\_max=<iops>**

スロットルなしの最大読み取りI/Oプール（1秒あたりの処理数）。

**iops\_rd\_max\_length=<seconds>**

読み取りI/Oバーストの最大長（秒）。

**iops\_wr=<iops>**

最大書き込みI/O（オペレーション毎秒）。

**iops\_wr\_max=<iops>**

スロットルなしの最大書き込みI/Oプール（1秒あたりの処理数）。

**iops\_wr\_max\_length=<秒>**

書き込みI/Oバーストの最大長（秒）。

**iothread=<ブール値>**

このドライブにiothreadsを使用するかどうか

**mbps=<mbps>**

最大R/W速度（メガバイト/秒）。

**mbps\_max=<mbps>**

最大スロットルなしR/Wプール（メガバイト/秒）。

**mbps\_rd=<mbps>**

最大読み取り速度（メガバイト/秒）。

**mbps\_rd\_max=<mbps>**

スロットルのない最大読み取りプール（メガバイト/秒）。

**mbps wr=<mbps>**  
最大書き込み速度（メガバイト/秒）。

**mbps wr\_max=<mbps>とする。**  
スロットルなしの最大書き込みプール（メガバイト/秒）。

**media=<cdrom | disk> (デフォルト=disk)**  
ドライブのメディア・タイプ。

**replicate=<boolean> (デフォルト=1)**  
ドライブをレプリケーション・ジョブの対象とするかどうか。

**rerror=<無視 | 報告 | 停止>。**  
読み取りエラー。

**ro=<ブール値**  
ドライブが読み取り専用かどうか。

**secs=<整数**  
ドライブの物理ジオメトリを特定のセクタ数に強制する。

**シリアル=<シリアル**  
ドライブのシリアル番号（URLエンコード、最大20バイト）。

**shared=<boolean> (デフォルト=0)**  
このローカル管理ボリュームをすべてのノードで使用可能なボリュームとしてマークします。



#### 警告

このオプションは、ボリュームを自動的に共有するのではなく、すでに共有されていると仮定します！

**size=<ディスクサイズ**  
ディスクサイズ。これは単なる情報提供であり、何の効果もない。

**snapshot=<boolean>**  
qemu のスナップショットモード機能を制御します。有効な場合、ディスクに加えられた変更は一時的なものであり、VM がシャットダウンされると破棄されます。

**trans=<自動 | lba | なし>。**  
ディスク・ジオメトリ・バイオスのトランスレーション・モードを強制する。

**werror=<enospc | 無視 | 報告 | 停止>。**

書き込みエラー。

#### **vmgenid: <UUID> (デフォルト = 1 (自動生成))**

VM世代ID (vmgenid) デバイスは、128ビットの整数値識別子をゲストOSに公開します。これにより、仮想マシンが異なる構成（スナップショット実行やテンプレートからの作成など）で実行された場合に、ゲストOSに通知することができます。ゲストOSはこの変更に気づき、分散データベースのコピーをダーティとしてマークしたり、乱数ジェネレーターを再初期化したりするなど、適切に対応することができます。自動作成はAPI/CLIのcreateまたはupdateメソッドで実行された場合にのみ機能し、設定ファイルを手動で編集した場合には機能しないことに注意してください。

**vmstatestorage: <ストレージID>**

VM状態のボリューム/ファイルのデフォルトストレージ。

**watchdog: [ [model=<i6300esb|ib700>] [アクション]**

仮想ハードウェアウォッチドッグデバイスを作成します。(ゲストのアクションによって)有効化されると、ウォッチドッグはゲスト内部のエージェントによって定期的にポーリングされる必要があります。

**action=<デバッグ|なし|一時停止|パワーオフ|リセット|シャットダウン>**

起動後、ゲストが時間内にウォッチドッグのポーリングに失敗した場合に実行するアクション。

**model=<i6300esb | ib700> (デフォルト=i6300esb)**

エミュレートするウォッチドッグ・タイプ。

## 10.15 錠前

オンライン・マイグレーション、スナップショット、およびバックアップ (vzdump) は、影響を受ける

```
# qm unlock <vmid>
```

VMに対する互換性のない同時アクションを防止するためにロックを設定します。時々、このようなロックを手動で削除する必要があります（電源障害後など）。

**注意**

ロックを設定したアクションがもう実行されていないことを確認している場合のみ、この操作を行ってください。

## 第11章

# Proxmoxコンテナツールキット

コンテナは、完全に仮想化されたマシン（VM）に代わる軽量なものだ。完全なオペレーティング・システム（OS）をエミュレートする代わりに、実行するホスト・システムのカーネルを使用する。つまり、コンテナはホストシステム上のリソースに直接アクセスできる。

コンテナのランタイム・コストは低く、通常は無視できる。しかし、考慮すべき欠点もある：

- Proxmoxコンテナで実行できるのはLinuxディストリビューションのみです。例えば、FreeBSDやMicrosoft Windowsのような他のオペレーティングシステムをコンテナ内で実行することはできません。
- セキュリティ上の理由から、ホスト・リソースへのアクセスを制限する必要がある。そのため、コンテナはそれぞれ別の名前空間で実行される。さらに、いくつかのシステムコール（Linuxカーネルに対するユーザー空間の要求）は、コンテナ内で許可されていない。

Proxmox VEは、[Linux Containers \(LXC\)](#)を基礎となるコンテナ技術として使用しています。Proxmox Container Toolkit（pct）は、複雑なタスクを抽象化するインターフェースを提供することで、LXCの使用と管理を簡素化します。

コンテナはProxmox VEと緊密に統合されています。つまり、コンテナはクラスタのセットアップを認識し、仮想マシンと同じネットワークおよびストレージリソースを使用できます。Proxmox VEのファイアウォールを使用したり、HAフレームワークを使用してコンテナを管理することもできます。

Proxmoxの主な目標は、VMを使用する利点を提供しながらも、オーバーヘッドを追加しない環境を提供することです。つまり、Proxmox Containersは "アプリケーションコンテナ"ではなく、"システムコンテナ"に分類されます。

## 注

アプリケーションコンテナ、例えば*Docker*イメージを実行する場合は、Proxmox QEMU VM内で実行することをお勧めします。これにより、アプリケーション・コンテナ化のすべての利点が得られるだけでなく、ホストからの強力な分離や、コンテナでは不可能なライブ・マイグレート機能など、VMが提供する利点も得られます。

---

## 11.1 技術概要

- LXC (<https://linuxcontainers.org/>)

- Proxmox VE グラフィカル・ウェブ・ユーザー・インターフェイス(GUI)に統合
- 使いやすいコマンドラインツール pct
- Proxmox VE REST API経由でのアクセス
- コンテナ化された /proc ファイルシステムを提供する lxcfs
- リソースの分離と制限のためのコントロールグループ (cgroups)
- AppArmorとseccompによるセキュリティの向上
- 最新のLinuxカーネル
- イメージベースのデプロイメント (テンプレート)
- Proxmox VEストレージライブラリを使用
- ホストからのコンテナ設定 (ネットワーク、DNS、ストレージなど)

## 11.2 対応ディストリビューション

公式にサポートされているディストリビューションの一覧は以下にある。

以下のディストリビューション用のテンプレートは、私たちのリポジトリから入手できます。pveam tool または Graphical User Interface を使ってダウンロードできます。

### 11.2.1 アルパイン・リナックス

Alpine Linuxは、musl libcとbusyboxをベースにしたセキュリティ重視の軽量Linuxディストリビューションです。

- <https://alpinelinux.org>

現在サポートされているリリースについては

<https://alpinelinux.org/releases/>

### 11.2.2 アーチリナックス

Arch Linux は軽量で柔軟な Linux® ディストリビューションで、Keep It Simple を目指しています。

Arch Linux はローリングリリースモデルを採用しています。詳しくは wiki を見て下さい:

[https://wiki.archlinux.org/title/Arch\\_Linux](https://wiki.archlinux.org/title/Arch_Linux)

### 11.2.3 CentOS、Almalinux、Rocky Linux

#### CentOS / CentOS Stream

CentOS Linuxディストリビューションは、Red Hat Enterprise Linux（RHEL）のソースから派生した、安定した、予測可能な、管理可能な、再現可能なプラットフォームです。

- <https://centos.org>

現在サポートされているリリースについては

[https://en.wikipedia.org/wiki/CentOS#End-of-support\\_schedule](https://en.wikipedia.org/wiki/CentOS#End-of-support_schedule) を

参照。

#### アルマリナックス

オープンソースで、コミュニティが所有し管理する、永久無料のエンタープライズLinuxディストリビューションで、長期的な安定性を重視し、堅牢なプロダクショングレードのプラットフォームを提供します。AlmaLinux OSは、RHEL®とプレストリームCentOSと1:1のバイナリ互換性があります。

- <https://almalinux.org>

現在サポートされているリリースについては

<https://en.wikipedia.org/wiki/AlmaLinux#Releases> を参照。

#### ロッキーリナックス

ロッキー・リナックスは、下流のパートナーが方向転換した現在、アメリカのトップ・エンタープライズLinuxディストリビューションと100%バグ互換となるように設計されたコミュニティ・エンタープライズ・オペレーティング・システムである。

- <https://rockylinux.org>

現在サポートされているリリースについては

[https://en.wikipedia.org/wiki/Rocky\\_Linux#Releases](https://en.wikipedia.org/wiki/Rocky_Linux#Releases) を参照。

### 11.2.4 デビアン

Debian はフリーなオペレーティングシステムであり、Debian プロジェクトによって開発・保守されています。フリーな Linux ディストリビューションで、ユーザのニーズを満たす数千のアプリケーションがあります。

- <https://www.debian.org/intro/index#software>

現在サポートされているリリースについては

<https://www.debian.org/releases/stable/releasenotes> を参照。

### 11.2.5 デヴアン

Devuan GNU+Linuxは、systemdを使わないDebianのフォークであり、不必要的絡みを避け、Init Freedomを確保することで、ユーザがシステムのコントロールを取り戻すことを可能にする。

- <https://www.devuan.org>

現在サポートされているリリースに

については

<https://www.devuan.org/os/releases>

を参照。

### 11.2.6 フェドラー

Fedoraproject.orgは、ハードウェア、クラウド、コンテナのための革新的で無償のオープンソースプラットフォームを作成し、ソフトウェア開発者とコミュニティメンバーがユーザーのためにカスタマイズされたソリューションを構築することを可能にします。

- <https://getfedora.org>

現在サポートされているリリースに

については

<https://fedoraproject.org/wiki/Releases>

を参照。

### 11.2.7 ジェンツー

柔軟性の高いソースベースのLinuxディストリビューション。

- <https://www.gentoo.org>

Gentooはローリング・リリース・モデルを使っている。

## 11.2.8 オープンソース

システム管理者、開発者、デスクトップ・ユーザーのためのメーカー選択です。

- <https://www.opensuse.org>

現在サポートされているリリースについては

<https://get.opensuse.org/leap/>

## 11.2.9 ウブントウ

Ubuntuは、エンタープライズ・サーバー、デスクトップ、クラウド、IoT向けのLinuxによる最新のオープンソース・オペレーティング・システムです。

- <https://ubuntu.com/>

現在サポートされているリリースに

については

<https://wiki.ubuntu.com/Releases> を参

照。

## 11.3 コンテナ画像

コンテナ・イメージは、「テンプレート」または「アプライアンス」とも呼ばれることがあります。コンテナを実行するためのすべてを含むtarアーカイブである。

Proxmox VE自体は、[最も一般的なLinuxディストリビューション](#)用のさまざまな基本テンプレートを提供しています。これらのテンプレートは、GUI または pveam (Proxmox VE Appliance Manager の略) コマンドラインユーティリティを使用してダウンロードできます。さらに、[TurnKey Linux](#) コンテナテンプレートもダウンロードできます。

利用可能なテンプレートのリストは、*pve-daily-update*タイマーによって毎日更新されます。手動で更新する

```
#pveam アップデート
```

こともできます：

利用可能な画像のリストを表示するには

```
#pveam available
```

例えば、基本システムなど、興味のあるセクションを指定することで、この大きなリストを制限することができます。

の画像をご覧いただきたい：

### 利用可能なシステムイメージのリスト

```
#pveam available --section system

システム          alpine-3.12-default_20200823_amd64.tar.xz
system          alpine-3.13-default_20210419_amd64.tar.xz
system          alpine-3.14-default_20210623_amd64.tar.xz
system          archlinux-base_20210420-1_amd64.tar.gz
system          centos-7-default_20190926_amd64.tar.xz
system          centos-8-default_20201210_amd64.tar.xz
system          debian-9.0-standard_9.7-1_amd64.tar.gz
system          debian-10-standard_10.7-1_amd64.tar.gz
system          devuan-3.0-standard_3.0_amd64.tar.gz
system          fedora-33-default_20201115_amd64.tar.xz
system          fedora-34-default_20210427_amd64.tar.xz

システム          gentoo-current-default_20200310_amd64.tar.xz
system          opensuse-15.2-default_20200824_amd64.tar.xz
system          ubuntu-16.04-standard_16.04.5-1_amd64.tar.gz
system          ubuntu-18.04-standard_18.04.1-1_amd64.tar.gz
system          ubuntu-20.04-standard_20.04-1_amd64.tar.gz
system          ubuntu-20.10-standard_20.10-1_amd64.tar.gz
system          ubuntu-21.04-standard_21.04-1_amd64.tar.gz
```

このようなテンプレートを使用する前に、いずれかのストレージにダウンロードする必要があります。どのストレージかわからない場合は、ローカルの名前付きストレージを使用することができます。クラスタ化されたインストールでは、すべてのノードがこれらのイメージにアクセスできるように、共有ストレージを使用することが推奨されます。

```
# pveam download local debian-10.0-standard_10.0-1_amd64.tar.gz
```

これで、そのイメージを使ってコンテナを作成する準備ができました。

```
# pveam list local
local:vztmpl/debian-10.0-standard_10.0-1_amd64.tar.gz 219.95MB
```

とのローカル:

## チップ

また、Proxmox VEウェブインターフェースGUIを使用して、コンテナテンプレートのダウンロード、一覧表示、削除を行うこともできます。

```
# pct create 999 local:vztmpl/debian-10.0-standard_10.0-1_amd64.tar.gz
```

pctは、例えば、新しいコンテナを作成するためにこれらを使用する:

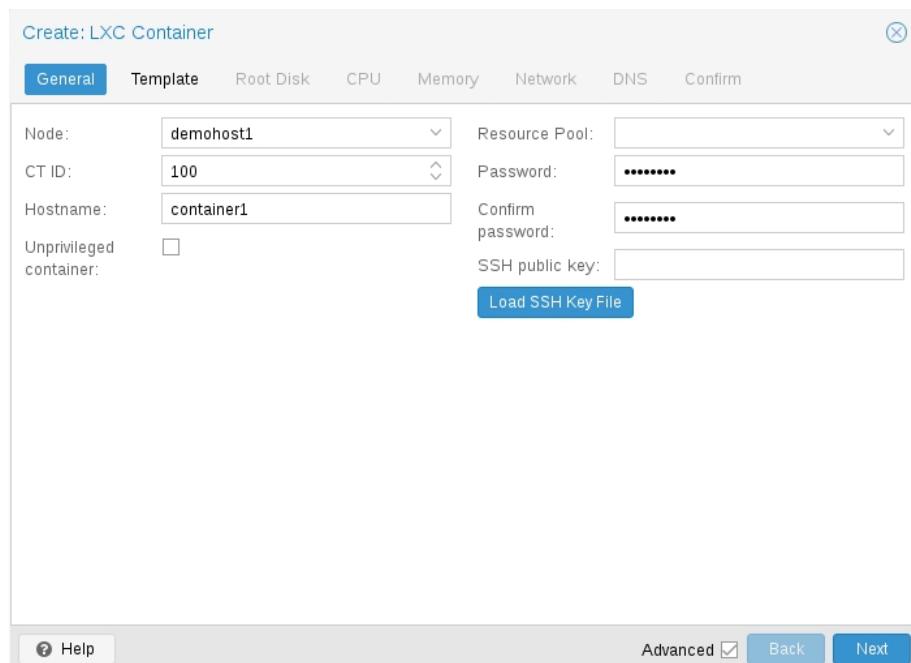
上記のコマンドは、完全なProxmox VEボリューム識別子を表示します。この識別子にはストレージ名が含

```
# pveam remove local:vztmpl/debian-10.0-standard_10.0-1_amd64.tar.gz
```

まれており、他のほとんどの Proxmox VE コマンドはこの識別子を使用できます。例えば、そのイメージを後で削除するには、次のようにします:

## 11.4 コンテナ設定

### 11.4.1 一般設定



コンテナの一般的な設定には以下が含まれる。

- ・ **ノード**: コンテナが実行される物理サーバー
- ・ **CT ID**: コンテナを識別するために使用されるProxmox VEのインストールで一意の番号です。

- **ホスト名:** コンテナのホスト名
- **リソースプール:** コンテナとVMの論理グループ
- **パスワード:** コンテナのルートパスワード
- **SSH公開鍵:** SSH経由でルート・アカウントに接続するための公開鍵
- **非特権コンテナ:** このオプションは、作成時に特権コンテナか非特権コンテナかを選択できる。

## 非特権コンテナ

非特権コンテナは、ユーザー・ネームスペースと呼ばれる新しいカーネル機能を使用する。コンテナ内のルートUID 0は、コンテナ外の非特権ユーザーにマッピングされる。つまり、これらのコンテナにおけるほとんどのセキュリティ問題（コンテナのエスケープ、リソースの乱用など）は、ランダムな非特権ユーザーに影響し、LXCの問題というよりは、一般的なカーネル・セキュリティのバグになる。LXCチームは、非特権コンテナは設計上安全だと考えている。

これは、新しいコンテナを作成するときのデフォルトのオプションである。

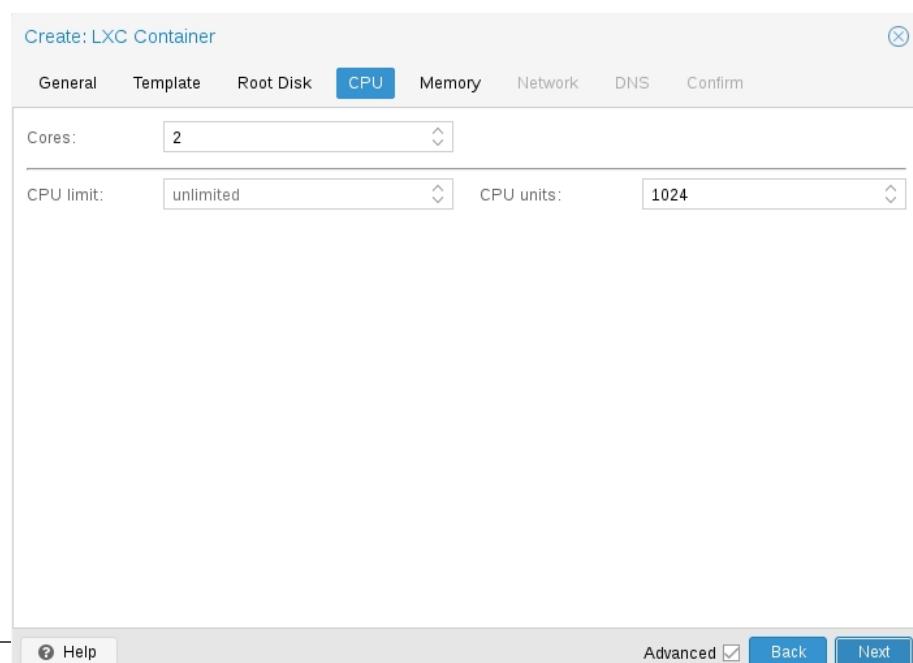
### 注

コンテナが init システムとして systemd を使用している場合、コンテナ内で実行されている systemd のバージョンは 220 以上でなければならない。

## 特権コンテナ

コンテナのセキュリティは、強制アクセス制御AppArmor制限、seccomp フィルター、Linuxカーネル名前空間を使用することで達成される。LXCチームは、この種のコンテナを安全でないものとみなしており、新しいコンテナ・エスケープ・エクスプロイトをCVEやクイック・フィックスに値するセキュリティ問題とはみなさない。だからこそ、特権コンテナは信頼できる環境でのみ使用すべきである。

### 11.4.2 CPU



`cores` オプションを使用して、コンテナ内の可視 CPU 数を制限できる。これはLinuxの*cpuset cgroup*（**制御グループ**）を使用して実装される。pvestatd 内部の特別なタスクが、実行中のコンテナを利用可能な CPU に定期的に分散しようとします。割り当てられたCPUを表示するには、以下のコマンドを実行する：

```
# pct cpusets
-----
102:       6  7
```

```
105:      2 3 4 5  
108:    0 1  
-----
```

コンテナはホスト・カーネルを直接使用する。コンテナ内のすべてのタスクは、ホストCPUスケジューラによって処理されます。Proxmox VEは、デフォルトでLinux CFS（Completely Fair Scheduler）スケジューラを使用し、これには帯域幅制御オプションが追加されています。

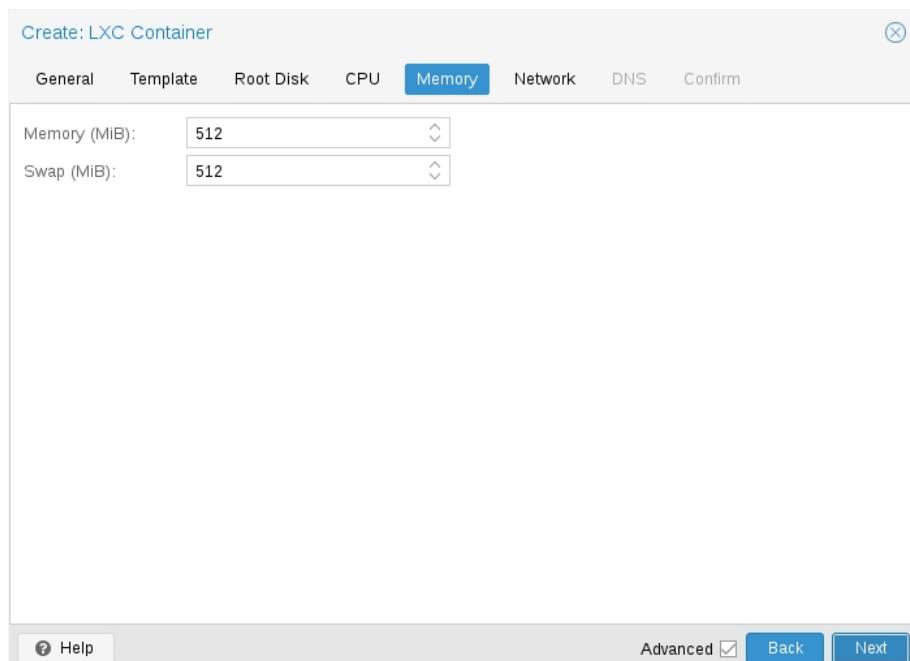
**cpulimit:** このオプションを使用すると、割り当てられたCPU時間をさらに制限することができます。これは浮動小数点数なので、コンテナに2つのコアを割り当ても、全体のCPU消費を半分のコアに制限することはまったく問題ありません。

コア数: 2

cpulimit: 0.5パーセント

**cpuunits:** これは、カーネル・スケジューラに渡される相対的な重みである。この数値が大きいほど、このコンテナのCPU時間は長くなる。数値は、実行中の他のすべてのコンテナの重みに対する相対値である。デフォルトは100（ホストがレガシーcgroupl v1を使用している場合は1024）。この設定を使用して、一部のコンテナに優先順位を付けることができる。

#### 11.4.3 メモリー



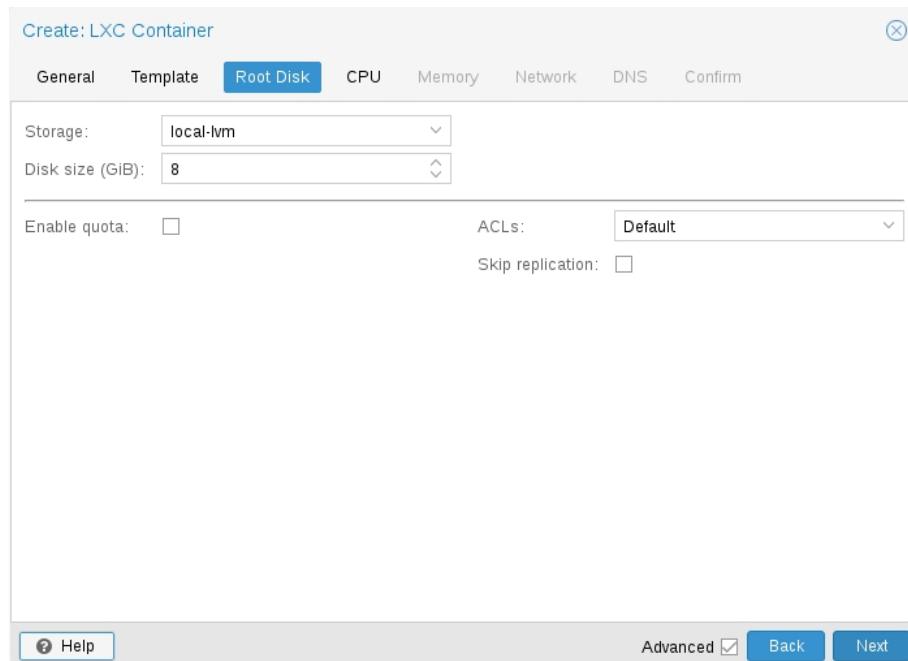
コンテナ・メモリは、cgroupメモリ・コントローラを使用して制御される。

memory:全体のメモリ使用量を 制限する。これは `memory.limit_in_bytes` に対応します。

cgroupの設定。

swap: コンテナがホストのスワップ領域から追加のスワップメモリを使用できるようにする。これは `memory.memsw.limit_in_bytes` cgroup 設定に 対応し、両方の値（メモリ + スワップ）の合計に設定される。

#### 11.4.4 マウントポイント



ルート・マウント・ポイントは `rootfs` プロパティで設定します。さらに最大256個のマウントポイントを設定できます。対応するオプションは `mp0` から `mp255` と呼ばれます。これらのオプションには、以下の設定を含めることができます:

```
rootfs: ボリューム=[<ボリューム> [,acl=<1|0>]
[,mountoptions=<opt[;opt...]>] [,quota=<1|0>] [,replicate=<1|0>]
[,ro=<1|0>] [,shared=<1|0>]
[,size=<1|0>[quota=<1|0>][,replicate=<1|0>][,ro=<1|0>][,shared=<
1|0>][,size=<DiskSize>]]。
```

ボリュームをコンテナ・ルートとして使用する。すべてのオプションの詳細については、以下を参照。

```
mp[n]: ボリューム=[<ボリューム> ,mp=<Path> [,acl=<1|0>] [,backup=<1|0>]
[,mountoptions=<opt[;opt...]>] [,quota=<1|0>] [,replicate=<1|0>]
[,ro=<1|0>] [,shared=<1|0>] [,size=<1|0>] [,マウントオプション
=<opt[;opt...[quota=<1|0>][,replicate=<1|0>][,ro=<1|0>][,shared=<1|0>]
```

**[, size=<DiskSize>]。**

ボリュームをコンテナ・マウント・ポイントとして使用する。新しいボリュームを割り当てるには、特別な構文 STORAGE\_ID:SIZE\_IN\_GiB を使用します。

**acl=<ブール値**

ACL サポートを明示的に有効または無効にする。

**backup=<ブール値**

マウント・ポイントをバックアップに含めるかどうか（ボリューム・マウント・ポイントにのみ使用）

。

**mountoptions=<opt[;opt...]>とする。**

rootfs/mps用の追加マウントオプション。

**mp=<パス**

コンテナ内部から見たマウントポイントへのパス。

**注**

セキュリティ上の理由から、シンボリックリンクを含んではならない。

**quota=<ブール値**

コンテナ内でユーザークォータを有効にする（zfsサブボリュームではサポートされていません）。

**replicate=<boolean>（デフォルト = 1）**

このボリュームをストレージ・レプリカ・ジョブに含めます。

**ro=<ブール値**

読み取り専用のマウントポイント

**shared=<boolean>（デフォルト = 0）**

この非ボリュームマウントポイントを全ノードで利用可能としてマークする。

**警告**

このオプションは、マウント・ポイントを自動的に共有するのではなく、共有されていると仮定します！

**size=<ディスクサイズ**

ボリュームサイズ（読み取り専用値）。

**ボリューム=<ボリューム**

コンテナにマウントするボリューム、デバイス、またはディレクトリ。

現在、マウント・ポイントには3つのタイプがある。ストレージ・バックされたマウント・ポイント、バインド・マウント、デバイス・マウントだ。

**典型的なコンテナrootfsの構成**

```
rootfs: thin1:base-100-disk-1,size=8G
```

**ストレージバックアップされたマウントポイント**

ストレージバックアップマウントポイントはProxmox VEストレージサブシステムによって管理され、3つの異なる種類があります：

- イメージベース： 単一のext4フォーマットされたファイルシステムを含むrawイメージです。
- ZFSサブボリューム： これらは技術的にはバインドマウントだが、管理されたストレージを持つため、サイズ変更とスナップショットが可能。
- ディレクトリ： size=0を渡すと、生イメージの代わりにディレクトリが作成される特殊なケースが発生します。

## 注

ストレージ・バックされたマウント・ポイント・ボリュームの特別なオプション構文

STORAGE\_ID:SIZE\_IN\_GBは、指定されたストレージ上に指定されたサイズのボリュームを自動的に割り当てます。例えば

```
pct set 100 -mp0 thin1:10,mp=/path/in/container
```

は、ストレージthin1上に10GBのボリュームを割り当て、ボリュームIDのプレースホルダー10を割り当てられたボリュームIDに置き換え、コンテナ内の/path/in/containerにマウントポイントをセットアップする。

## マウントポイントのバインド

バインドマウントを使用すると、Proxmox VEホストからコンテナ内の任意のディレクトリにアクセスできます。以下のような使用例が考えられます：

- ゲストでホームディレクトリにアクセスする
- ゲストでUSBデバイスのディレクトリにアクセスする
- ゲストでホストからNFSマウントにアクセスする

バインド・マウントはストレージ・サブシステムによって管理されていないとみなされるため、コンテナ内部からスナップショットを作成したりクォータを処理したりすることはできない。非特権コンテナでは、ユーザーマッピングに起因するパーミッションの問題に遭遇する可能性があり、ACLを使用できない。

## 注

vzdumpを使用した場合、バインドマウントポイントの内容はバックアップされません。

### 警告

セキュリティ上の理由から、バインドマウントは、この目的のために特別に確保されたソースディレクトリ（たとえば /mnt/bindmounts 以下のディレクトリ階層）を使ってのみ確立すべきである。マウント・システム・ディレクトリ（/、/var、/etc など）をコンテナにバインドしてはならない。

## 注

バインドマウントのソースパスにはシンボリックリンクを含んではならない。

---

たとえば、ID 100 のコンテナで /mnt/bindmounts/shared ディレクトリにアクセスできるようにするには、次のようにする。

```
mp0: /mnt/bindmounts/shared,mp=/shared
```

の下に、次のような設定行を追加する：

を /etc/pve/lxc/100.conf に

```
pct set 100 -mp0 /mnt//バインドマウント/共有,mp=/共有
```

う：

同じ結果を得るために。

## デバイスマウントポイント

デバイスマウントポイントを使用すると、ホストのブロックデバイスをコンテナに直接マウントできます。バインドマウントと同様に、デバイスマウントは Proxmox VE のストレージサブシステムによって管理されませんが、クォータと acl オプションは尊重されます。

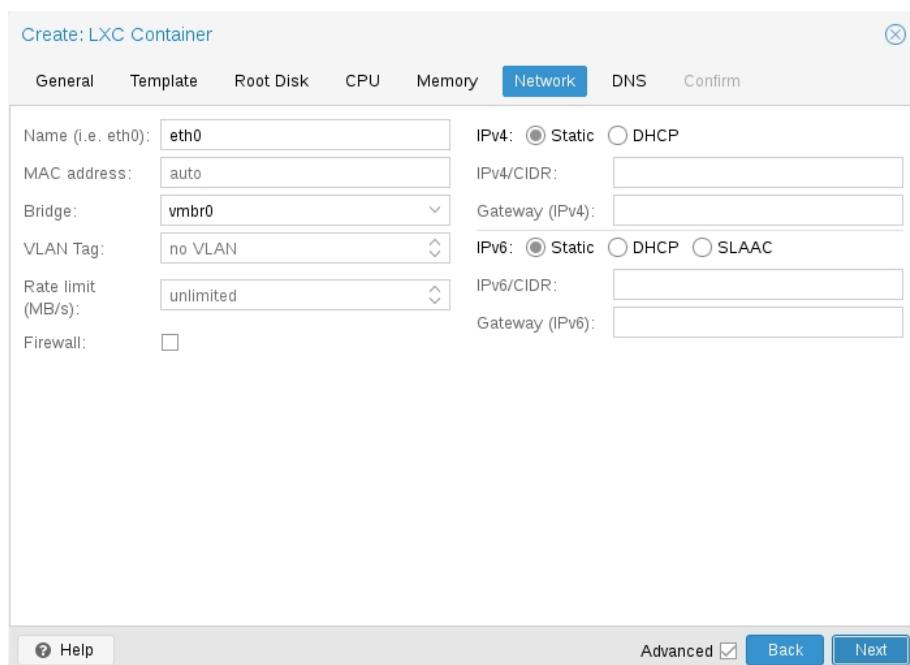
### 注

デバイスマウント・ポイントは、特別な状況下でのみ使用すべきである。ほとんどの場合、ストレージ・バックされたマウント・ポイントは、同じパフォーマンスとより多くの機能を提供します。

### 注

vzdumpを使用した場合、デバイスマウント・ポイントの内容はバックアップされません。

## 11.4.5 ネットワーク



1つのコンテナに最大10個のネットワーク・インターフェースを設定できる。対応するオプションは net0からnet9までで、以下の設定を含むことができる：

```
net[n]:name=<string> [,bridge=<bridge>] [,firewall=<1|0>]
[,gw=<GatewayIPv4>] [,gw6=<GatewayIPv6>] [,hwaddr=<XX:XX:XX:XX:XX:>
XX>] [,ip=<(IPv4/CIDR|dhcp|manual)>]
```

```
[,ip6=<(IPv6|CIDR|auto|dhcp|manual)>] [,link_down=<1|0>]  
[,mtu=<integer>] [,rate=<mbps>] [,tag=<integer>]  
[,trunks=<vlanid[;vlanid...]>] [,type=<veth>]
```

コンテナのネットワーク・インターフェースを指定する。

## ブリッジ

ネットワークデバイスを接続するブリッジ。

**ファイアウォール=<ブール**

このインターフェイスのファイアウォール・ルールを使用するかどうかを制御する。

**gw=<ゲートウェイIPv4**

IPv4 トライフィックのデフォルトゲートウェイ。

**gw6=<ゲートウェイIPv6**

IPv6 トライフィックのデフォルトゲートウェイ。

**hwaddr=<XX:XX:XX:XX:XX:XX>。**

I/G (Individual/Group) ビットが設定されていない共通のMACアドレス。

**ip=<(IPv4/CIDR|dhcp|manual)>とする。**

CIDR形式のIPv4アドレス。

**ip6=<(IPv6/CIDR|auto|dhcp|manual)>となります。**

CIDR形式のIPv6アドレス。

**link\_down=<ブール値**

このインターフェイスを切断すべきかどうか（プラグを抜くように）。

**mtu=<整数> (64 - 65535)**

インターフェースの最大転送単位。(lxc.network.mtu)

**name=<文字列**

コンテナ内部から見たネットワークデバイスの名前。(lxc.network.name)

**レート=<mbps**

インターフェイスにレート制限を適用する

**タグ=<整数> (1 - 4094)**

このインターフェースのVLANタグ。

**trunks=<vlanid[,vlanid...]>とする。**

インターフェイスを通過するVLAN ID

**type=<バス**

ネットワークインターフェースのタイプ。

### 11.4.6 コンテナの自動スタートとシャットダウン

ホスト・システムの起動時にコンテナを自動的に起動するには、*Options* で *Start at boot* オプションを選択する。パネルを開くか、以下のコマンドを実行する：

```
# pct set CTID -onboot 1
```

## スタートとシャットダウンの順序



コンテナの起動順序を微調整したい場合は、以下のパラメーターを使用できる：

- **スタート/シャットダウン順：**開始順序の優先順位を定義します。例えば、CTを最初に起動させたい場合は1に設定する。(シャットダウンには逆の起動順序を使用するため、起動順序が1のコンテナは最後にシャットダウンされます)
- **起動遅延：**このコンテナの起動と後続のコンテナの起動の間隔を定義する。たとえば、他のコンテナを起動する前に240秒待つ場合は240に設定します。
- **シャットダウンタイムアウト：**Proxmox VEがシャットダウンコマンドを発行した後、コンテナがオフラインになるまで待機する時間を秒単位で定義します。デフォルトでは、この値は60に設定されています。これは、Proxmox VEがシャットダウン要求を発行し、マシンがオフラインになるまで60秒待機し、60秒経過後もマシンがオンラインの場合は、シャットダウン動作が失敗したことを探知することを意味します。

Start/Shutdown順序パラメータが設定されていないコンテナは、常にパラメータが設定されているコンテナの後に開始することに注意してください。また、このパラメータは、ホスト上でローカルに実行されているマシン間でのみ意味を持ち、クラスタ全体では意味を持ちません。

ホストのブートと最初のコンテナのブートの間に遅延が必要な場合は、[Proxmox VE Node Management](#)のセクションを参照してください。

### 11.4.7 フックスクリプト

CTにフックスクリプトを追加するには、設定プロパティ `hookscript` を使用します。

```
# pct set 100 -hookscript local:snippets/hookscript.pl
```

このスクリプトはゲストが生きている間の様々な局面で呼び出されます。例とドキュメントは `/usr/share/pve-docs/examples/guest-example-hookscript.pl` にあるサンプルスク

リプトを参照してください。

## 11.5 セキュリティへの配慮

コンテナはホスト・システムのカーネルを使用する。そのため、悪意のあるユーザーにとって攻撃対象が露出することになる。一般的に、完全な仮想マシンの方がより優れた分離を提供する。コンテナを未知の人や信頼できない人に提供する場合は、この点を考慮する必要がある。

攻撃対象領域を減らすために、LXCはAppArmor、CGroups、kernel namespacesといった多くのセキュリティ機能を使用している。

### 11.5.1 AppArmor

AppArmorプロファイルは、危険な可能性のあるアクションへのアクセスを制限するために使用される。一部のシステムコール（マウントなど）は実行が禁止されている。

AppArmorのアクティビティをトレースするには、以下を使用する：

```
# dmesg | grep apparmor
```

推奨はされないが、AppArmorはコンテナに対して無効にすることができます。これにはセキュリティ・リスクが伴う。システムが誤って設定されていたり、LXCやLinuxカーネルに脆弱性があったりすると、コンテナ内でいくつかのシステムコールが実行され、特権の昇格につながる可能性がある。

コンテナのAppArmorを無効にするには、以下の行を、以下の場所にあるコンテナ構成ファイルに追加する。

/etc/pve/lxc/CTID.conf:

```
lxc.apparmor.profile=アンコンファインド
```



#### 警告

本番での使用は推奨されていませんのでご注意ください。

### 11.5.2 対照群 (cgroup)

cgroupは、プロセスを階層的に整理し、システムリソースを分配するために使用されるカーネルのメカニズムです。

cgroupsによって制御される主なリソースは、CPU時間、メモリとスワップの制限、デバイスノードへのアクセスである。cgroupsは、スナップショットを撮る前にコンテナを「フリーズ」させるためにも使われる。

cgroupsには現在、legacyとcgroupv2の2つのバージョンがある。

Proxmox VE 7.0 以降、デフォルトは純粋な cgroupv2 環境です。以前は「ハイブリッド」セットアップが使用され、リソース制御は主に cgroupv1 で行われ、cgroup\_no\_v1 カーネルコマンドラインパラメータを使用して一部のサブシステムを引き継ぐことができる cgroupv2 コントローラーが追加されました。（[詳細はカーネルパラメータのドキュメントを参照](#)）。

## CGroup バージョン互換性

Proxmox VE に関する純粋な *cgroupv2* と従来のハイブリッド環境の主な違いは、*cgroupv2* ではメモリとスワップが独立して制御されるようになったことです。以前はメモリの上限とメモリとスワップの合計の上限しか制限できませんでしたが、コンテナのメモリとスワップの設定はこれらの値に直接マッピングできます。

もう1つの重要な違いは、デバイスコントローラが全く異なる方法で設定されることである。このため、純粋な *cgroupv2* 環境では、ファイルシステムのクオータは現在サポートされていません。

純粋な *cgroupv2* 環境で実行するには、コンテナの OS による *cgroupv2* サポートが必要です。*systemd* バージョン 231 以降を実行しているコンテナは *cgroupv2* をサポートしています。<sup>1</sup>をサポートしています。<sup>2</sup>

<sup>1</sup> これには、Proxmox VEが出荷したコンテナテンプレートの最新メジャーバージョンすべてが含まれます。

<sup>2</sup> 例えばアルパイン・リナックス

## 注

CentOS 7とUbuntu 16.10は、*cgroupv2*環境で実行するには古すぎる*systemd*バージョンを持つ2つの著名なLinuxディストリビューションのリリースである。

- ディストリビューション全体を新しいリリースにアップグレードする。上記の例では、Ubuntu 18.04や20.04、CentOS 8（またはAlmaLinuxやRocky LinuxのようなRHEL/CentOSの派生版）などが考えられます。これには、最新のバグやセキュリティの修正、多くの場合新機能の追加、EOLの時期を未来にずらすという利点があります。
- Containers *systemd* のバージョンをアップグレードする。ディストリビューションが *backports* リポジトリを提供している場合、これは簡単で迅速な応急処置になります。
- コンテナやそのサービスを仮想マシンに移す。仮想マシンはホストとのインタラクションが非常に少ないで、何十年も前のOSバージョンでも問題なくインストールできる。
- レガシーの*cgroup*コントローラに戻す。これは有効な解決策ですが、永久的な解決策ではないことに注意してください。Proxmox VE 9.0以降、レガシーコントローラはサポートされなくなります。

## CGグループのバージョン変更

### チップ

ファイルシステムのクオータが必要なく、すべてのコンテナが*cgroupv2*をサポートしている場合は、新しいデフォルトにこだわることを推奨する。

以前のバージョンに戻すには、以下のカーネル・コマンドライン・パラメーターを使用する：

```
systemd.unified_cgroup_hierarchy=0
```

パラメータを追加する場所については、カーネルブートコマンドラインの編集に関する[このセクション](#)を参照のこと。

## 11.6 ゲストOSの構成

Proxmox VE はコンテナ内の Linux ディストリビューションの検出を試み、いくつかのファイルを変更する

。以下は、コンテナ起動時に実行されることの簡単なリストです：

**etc/ホスト名を設定する**

コンテナ名を設定する

**etc/hostsを修正する**

ローカルホスト名の検索を許可する

**ネットワーク設定**

完全なネットワーク・セットアップをコンテナに渡す

**DNSの設定**

DNSサーバーに関する情報を渡す

## initシステムを適応させる

例えば、gettyプロセスの生成数を修正する。

## rootパスワードを設定する

新規コンテナ作成時

## ssh\_host\_keys を書き換える

各コンテナが一意のキーを持つように

## crontabのランダム化

すべてのコンテナでcronが同時に起動しないようにする。

Proxmox VEによる変更はコメントマーカーで囲まれています：

```
# --- PVEを始める ---
<データ
--- pve 終了 ---
```

これらのマーカーは、ファイル内の適切な位置に挿入される。そのようなセクションがすでに存在する場合は、その場所に更新され、移動されることはない。

ファイルの変更は、.pve-ignore.ファイルを追加することで防ぐことができます。例えば /etc/.pve-ignore.hostsが存在する場合、/etc/hostsファイルは触られません。このファイルは単純な空のファイルでも構いません：

```
# touch /etc/.pve-ignore.hosts
```

ほとんどの修正はOSに依存するため、ディストリビューションやバージョンによって異なる。手動でostypeをunmanagedに設定することで、修正を完全に無効にすることができる。

OS タイプの検出は、コンテナ内の特定のファイルをテストすることで行われます。 Proxmox VE は最初に /etc/os-release ファイル<sup>3</sup> このファイルが存在しないか、明確に認識できるディストリビューション識別子が含まれていない場合、以下のディストリビューション固有のリリースファイルがチェックされます。

## ウブントゥ

etc/lsb-release を検査する (DISTRIB\_ID=Ubuntu)

## デビアン

test /etc/debian\_version

**フェドラ**

test /etc/fedora-release

**RedHatまたはCentOS**

test /etc/redhat-release

**アーキリナックス**

test /etc/arch-release

<sup>3</sup>etc/os-releaseは、ディストリビューションごとの多数のリリースファイルを置き換えます

<https://manpages.debian.org/stable/systemd/os-release.5.en.html>

### アルパイン

test /etc/alpine-release

### ジェンツー

test /etc/gentoo-release

### 注

設定された `ostype` が自動検出されたタイプと異なる場合、コンテナの起動に失敗します。

## 11.7 コンテナ保管

Proxmox VE LXC コンテナストレージモデルは、従来のコンテナストレージモデルよりも柔軟性が高い。コンテナは複数のマウントポイントを持つことができる。これにより、各アプリケーションに最適なストレージを使用できる。

例えば、コンテナのルートファイルシステムは低速で安価なストレージに置き、データベースは2つ目のマウントポイントを経由して高速で分散ストレージに置くことができる。詳細については、「[マウント・ポイント](#)」セクションを参照。

Proxmox VEストレージライブラリでサポートされているストレージタイプであれば、どれでも使用できます。つまり、コンテナはローカル (`lvm`、`zfs`、ディレクトリなど)、共有外部 (`iSCSI`、`NFS`など)、あるいはCephのような分散ストレージシステムにも保存できる。基礎となるストレージがサポートしていれば、スナップショットやクローンのような高度なストレージ機能も使用できる。`vzdump`バックアップツールは、スナップショットを使用して一貫性のあるコンテナバックアップを提供できる。

さらに、ローカルデバイスやローカルディレクトリは、バインドマウントを使って直接マウントできる。これにより、実質的にオーバーヘッドゼロでコンテナ内のローカルリソースにアクセスできる。バインドマウントは、コンテナ間でデータを共有する簡単な方法として使用できる。

### 11.7.1 ヒューズマウント

#### 警告

Linuxカーネルのフリーザー・サブシステムには既存の問題があるため、コンテナ内でFUSEマウントを使用することは強く推奨されません。コンテナは、サスPENDまたはスナップショット・モード

---

ドのバックアップのために凍結する必要があるからです。

---

FUSEマウントを他のマウント機構やストレージ技術で置き換えることができない場合は、Proxmoxホスト上でFUSEマウントを確立し、バインドマウントポイントを使用してコンテナ内でアクセスできるようにすることができます。

### 11.7.2 コンテナ内でクオータを使用する

クオータは、コンテナ内で各ユーザーが使用できるディスク容量の制限を設定することができる。

---

#### 注

これには現在、レガシーなcグループを使用する必要がある。

---

---

### 注

これはext4イメージベースのストレージタイプでのみ機能し、現在のところ特権コンテナでのみ機能する。

---

クオータ・オプションを有効にすると、マウント・ポイントに次のマウント・オプションが使用されます:  
usrjquota=aquo

これにより、他のシステムと同様にクオータを使用することができる。初期化された/aquota.userと

```
# quotacheck -cmug  
/ # quotaon /  
/aquota.groupファイルを実行する:
```

次に、edquota コマンドを使用してクオータを編集する。詳細については、コンテナ内で実行されているディストリビューションのドキュメントを参照すること。

---

### 注

上記のコマンドは、マウント・ポイントごとに実行する必要がある。その際、/の代わりにマウント・ポイントのパスを渡す。

---

## 11.7.3 コンテナ内でACLを使用する

標準のPosixアクセス・コントロール・リストもコンテナ内で利用できる。ACLでは、従来のユーザー/グループ/その他モデルよりも詳細なファイル所有権を設定できる。

## 11.7.4 コンテナマウントポイントのバックアップ

マウントポイントをバックアップに含めるには、コンテナ構成でそのマウントポイントのバックアップオプ

```
mp0: guest:subvol-100-disk-1,mp=/ルート/ファイル,size=8G
```

ションを有効にする。既存のマウントポイント mp0 の場合

backup=1を追加して有効にする。

```
mp0: guests:subvol-100-disk-1,mp=/root/files,size=8G,backup=1.
```

---

### 注

GUIで新しいマウント・ポイントを作成する場合、このオプションはデフォルトで有効になっています。

マウント・ポイントのバックアップを無効にするには、上記の方法で`backup=0`を追加するか、または  
**Backup=0**のチェックを外します。  
をチェックする。

### 11.7.5 コンテナのマウントポイントのレプリケーション

デフォルトでは、ルートディスクがレプリケートされると、追加のマウントポイントがレプリケートされます。Proxmox VEストレージレプリケーションメカニズムでマウントポイントをスキップしたい場合は、そのマウントポイントに [レプリケーションをスキップ] オプションを設定します。Proxmox VE 5.0では、レプリケーションには`zfspool`タイプのストレージが必要です。コンテナでレプリケーションが設定されているときに別のタイプのストレージにマウントポイントを追加するには、そのマウントポイントでレプリケーションをスキップするオプションを有効にする必要があります。

## 11.8 バックアップと復元

### 11.8.1 コンテナのバックアップ

コンテナのバックアップにvzdumpツールを使用することは可能である。詳細はvzdumpのマニュアルページを参照されたい。

### 11.8.2 コンテナ・バックアップの復元

vzdump で作成したコンテナ・バックアップの復元は、pct restore コマンドを使用して行うことができる。デフォルトでは、pct restore はバックアップされたコンテナ構成を可能な限り復元しようとする。コマンドラインでコンテナオプションを手動で設定することで、バックアップされた構成を上書きすることができる（詳細は pct のマニュアルページを参照）。

---

#### 注

pvesm extractconfig を使用すると、vzdump アーカイブに含まれるバックアップされた設定を表示できます。

---

基本的なリストアモードは2つあり、マウントポイントの扱いが異なるだけである：

#### 「シンプル」リストアモード

rootfsパラメータもオプションのmpXパラメータも明示的に設定されていない場合、バックアップされた設定ファイルのマウントポイント設定は、以下の手順で復元されます：

1. バックアップからマウントポイントとそのオプションを抽出する
  2. ストレージパラメータ（デフォルト：ローカル）で指定されたストレージ上に、ストレージバックされたマウントポイント用のボリュームを作成する。
  3. バックアップアーカイブからファイルを取り出す
  4. リストアされたコンフィギュレーションにバインドポイントとデバイスマウントポイントを追加（rootユーザー限定）
-

## 注

バインドとデバイスのマウントポイントは決してバックアップされないので、最後のステップではファイルはリストアされず、設定オプションだけがリストアされる。このようなマウントポイントは、別のメカニズム（たとえば、多くのコンテナにバインドマウントされたNFSスペース）でバックアップされているか、まったくバックアップされることを意図していないかのいずれかであることが前提です。

---

このシンプル・モードは、ウェブ・インターフェイスのコンテナ・リストア操作でも使用される。

## 「アドバンス」リストアモード

`rootfs` パラメータ (およびオプションで `mpX` パラメータの任意の組み合わせ) を設定することで、`pct restore` コマンドは自動的にアドバンストモードに切り替わります。このアドバンスマードは、バックアップアーカイブに含まれる `rootfs` と `mpX` の設定オプションを完全に無視し、代わりにパラメータとして明示的に提供されたオプションのみを使用します。

このモードでは、例えばリストア時にマウントポイントの設定を柔軟に設定できる：

- 各マウントポイントに対して、ターゲットストレージ、ボリュームサイズ、その他のオプションを個別に設定する。
- 新しいマウントポイントスキームに従ってバックアップされたファイルを再配布する。
- デバイスおよび/またはバインドマウントポイントへのリストア (rootユーザーに限定)

## 11.9 pctでコンテナを管理する

"Proxmox Container Toolkit" (`pct`) は Proxmox VE コンテナを管理するコマンドラインツールです。コンテナの作成や破棄、コンテナ実行の制御（開始、停止、再起動、マイグレーションなど）が可能です。また、コンテナの設定ファイルにパラメータを設定することも可能で、ネットワーク設定やメモリ制限などを設定できます。

### 11.9.1 CLIの使用例

Debian テンプレートをベースにコンテナを作成する (ウェブインターフェースでテンプレートをダウンロード

```
# pct create 100 /var/lib/vz/template/cache/debian-10.0-standard_10.0-1-'  
    amd64.tar.gz
```

済みであることが前提)

スタート・コンテナ 100

```
# pct start 100
```

getty経由でログインセッションを開始する

```
# pct コンソール 100
```

LXCネームスペースに入り、rootユーザーとしてシェルを実行する。

```
# pct enter 100
```

コンフィギュレーションを表示する

```
# pct config 100
```

ホストブリッジvmbr0にブリッジされたeth0というネットワークインターフェースを追加し、アドレスとゲートウェイを設定する。

```
# pct set 100 -net0 name=eth0,bridge=vmbr0,ip=192.168.15.147/24,gw=""  
=192.168.15.1
```

コンテナのメモリを512MBに減らす。

```
# pct set 100 -memory 512
```

コンテナを破棄すると、そのコンテナは常にアクセス制御リストから削除され、そのコンテナのファイアウ

```
# pct destroy 100 --purge
```

オール構成も常に削除される。レプリケーション・ジョブ、バックアップ・ジョブ、HAリソース構成からもコンテナを削除する場合は、*-purge* を有効にする必要がある。

マウントポイントボリュームを別のストレージに移動する。

```
# pct move-volume 100 mp0 other-storage
```

ボリュームを別のCTに再割り当てします。これにより、ソースCTからボリュームmp0が削除され、ターゲットCTにmp1としてアタッチされます。バックグラウンドでボリュームの名前が新しいオーナーと一致するように変更されます。

```
# pct move-volume 100 mp0 --target-vmid 200 --target-volume mp1
```

## 11.9.2 デバッグ・ログの取得

pct startが特定のコンテナを起動できない場合、*--debug* フラグ（CTIDをコンテナのCTIDに置き換える）を渡してデバッグ出力を収集すると役に立つかもしれない：

あるいは、以下のlxc-startコマンドを使用することもできる。このコマンドは、*-o output*オプションで指定

```
# lxc-start -n CTID -F -l DEBUG -o /tmp/lxc-CTID.log
```

されたファイルにデバッグ・ログを保存する：

このコマンドは、フォアグラウンド・モードでコンテナを起動しようとする。コンテナを停止するには、2番目のターミナルで pct shutdown CTID または pct stop CTID を実行する。

収集されたデバッグ・ログは /tmp/lxc-CTID.log に書き込まれる。

---

### 注

最後に pct start で起動を試みてからコンテナの構成を変更した場合は、少なくとも 1 回 pct start を実行して、lxc-start が使用する構成も更新する必要がある。

---

## 11.10 移住

クラスタがある場合、コンテナのマイグレーションは

```
# pct migrate <ctid> <target>
```

これはコンテナがオフラインである限り動作します。ローカルボリュームまたはマウントポイントが定義されている場合、同じストレージがターゲットホストに定義されていれば、移行はネットワーク経由でコンテナをターゲットホストにコピーします。

実行中のコンテナは、技術的な制限によりライブマイグレーションできない。再起動マイグレーションは可能で、シャットダウンしてコンテナを移動し、ターゲットノード上で再びコンテナを起動する。コンテナは非常に軽量であるため、通常は数百ミリ秒のダウンタイムで済む。

マイグレーションの再開は、ウェブ・インターフェイスを使うか、`pct migrate`コマンドで`--restart`フラグを使うことで行うことができる。

再起動マイグレーションはコンテナをシャットダウンし、指定されたタイムアウト（デフォルトは180秒）の後にコンテナを強制終了します。その後、オフラインマイグレーションのようにコンテナをマイグレートし、終了するとターゲットノード上でコンテナを起動します。

## 11.11 構成

`etc/pve/lxc/<CTID>.conf` ファイルにはコンテナ構成が格納されます。`<CTID>` は指定されたコンテナの数値 ID です。`etc/pve/` 内に格納されている他のすべてのファイルと同様に、他のすべてのクラスタノードに自動的に複製されます。

---

### 注

`CTID < 100`は内部用に予約されており、`CTID`はクラスタ全体で一意である必要がある。

---

### コンテナの構成例

```
ostype: debian
arch: amd64
hostname: www
memory: 512
スワップ512
net0: bridge=vmbr0, hwaddr=66:64:66:64:64:36, ip=dhcp, name=eth0, type=veth
rootfs: local:107/vm-107-disk-1.raw, size=7G
```

設定ファイルはシンプルなテキストファイルです。通常のテキストエディタ、たとえば `vi` や `nano` を使って編集できる。ちょっとした修正に便利な場合もあるが、そのような変更を適用するにはコンテナを再起動する必要があることを覚えておいてほしい。

そのため、通常は `pct` コマンドを使用してファイルを生成・変更するか、GUIを使用してすべてを行いう方がよい。私たちのツールキットは賢いので、実行中のコンテナに対してほとんどの変更を瞬時に適用できる。この機能は「ホットプラグ」と呼ばれ、この場合コンテナを再起動する必要はない。

---

変更をホットプラグできない場合は、保留中の変更として登録される（GUI では赤色で表示される）。これらはコンテナを再起動した後にのみ適用される。

### 11.11.1 ファイル形式

コンテナー・コンフィギュレーション・ファイルは、コロンで区切られた単純なキー/値形式を使用する。各

```
# これはコメントです。
```

行の書式は以下の通りである：

これらのファイルの空白行は無視され、#文字で始まる行はコメントとして扱われ、これも無視される。

例えば、低レベルのLXCスタイルのコンフィギュレーションを直接追加することも可能だ：

```
lxc.init_cmd: /sbin/my_own_init
```

または

```
lxc.init_cmd = /sbin/my_own_init
```

設定はLXCの低レベルツールに直接渡される。

## 11.11.2 スナップ写真

スナップショットを作成すると、`pct`はスナップショット時の設定と同じ設定ファイル内の別のスナップショット・セクションに保存します。例えば、"testsnapshot"というスナップショットを作成した後、設定ファイルは以下のようになります：

### スナップショットによるコンテナ構成

```
メモリ 512
スワップ 512
親: テストナフォト
...
[testsnapshot]
メモリ: 512
スワップ 512
スナップタイム: 1457170803
...
```

`parent`や`snaptime`のようなスナップショット関連のプロパティがいくつかあります。`parent`プロパティはスナップショット間の親子関係を保存するために使用されます。`snaptime`はスナップショット作成のタイムスタンプ（Unixエポック）です。

## 11.11.3 オプション

---

```
arch: <amd64 | arm64 | armhf | i386 | riscv32 | riscv64> (デフォルト = amd64)
```

OSのアーキテクチャタイプ。

#### cmode: <コンソール | シェル | tty> (デフォルト = tty)

コンソールモード。デフォルトでは、`console`コマンドは利用可能なttyデバイスの1つに接続を開こうとします。`cmode`を`console`に設定すると、代わりに`/dev/console`にアタッチしようとする。`cmode`を`shell`に設定すると、単にコンテナ内でシェルを起動する（ログインはしない）。

#### console: <論理値> (デフォルト = 1)

コンテナにコンソールデバイス（`/dev/console`）をアタッチする。

**コア:<整数> (1 - 8192)**

コンテナに割り当てられたコアの数。コンテナは、デフォルトで使用可能なすべてのコアを使用できる。

**cpulimit:<数値> (0 - 8192) (デフォルト = 0)**

CPU使用量の上限。

**注**

コンピュータに2つのCPUがある場合、合計2つのCPU時間を持つ。値0はCPU制限なしを示す。

**cpuunits:<整数> (0 - 500000) (デフォルト = cgroup v1: 1024, cgroup v2: 100)**

コンテナのCPUウェイト。引数は、カーネルのフェアスケジューラで使用される。数値が大きいほど、このコンテナのCPU時間は長くなる。数値は、他のすべての実行中のゲストの重みに対する相対値である。

**debug: <boolean> (デフォルト = 0)**

もっと冗長にしてみてください。今のところ、これは起動時にデバッグログレベルを有効にするだけです。

**説明:<文字列>**

コンテナの説明。ウェブインターフェイスCTのサマリーに表示されます。設定ファイルのコメントとして保存されます。

**dev[n]: [[パス=<パス>] [,gid=<整数>[gid=<整数>]] [,mode=<オクタルアクセスモード>] [,uid=<整数>]**

コンテナへの通過装置

**gid=<整数> (0 - N)**

デバイスノードに割り当てるグループID

**mode=<オクタル・アクセス・モード>**

デバイスノードに設定されるアクセスモード

**パス=<パス>**

コンテナに通すデバイスへのパス

**uid=<整数> (0 - N)**

デバイスノードに割り当てるユーザーID

の機能があります: [**force\_rw\_sys=<1|0>**] [, **fuse=<1|0>**] [, **keyctl=<1|0>**] [, **mknod=<1|0>**] [, **mount=<fstype;fstype;...>**] [, **nesting=<1|0>**] となります

。

コンテナが高度な機能にアクセスできるようにする。

**force\_rw\_sys=<boolean> (デフォルト = 0)**

非特権コンテナの /sys を mixed ではなく rw でマウントするように。これは、新しい (>= v245) systemd-network の使用下でネットワークを壊す可能性がある。

**fuse=<boolean> (デフォルト = 0)**

コンテナ内の fuse ファイルシステムの使用を許可する。fuse と freezer cgroup の相互作用により、I/O デッドロックが発生する可能性があることに注意。

**keyctl=<ブール値> (デフォルト = 0)**

非特権コンテナ専用：keyctl() システムコールの使用を許可する。これは、コンテナ内で docker を使用するためには必要です。デフォルトでは、非特権コンテナはこのシステムコールを存在しないと見なす。これは主に systemd-networkd のための回避策で、keyctl() の操作がパーミッション不足でカーネルに拒否された場合に致命的なエラーとして扱われるからです。基本的には、systemd-networkd を走らせるか docker を走らせるかを選ぶことができます。

**mknod=<boolean> (デフォルト = 0)**

非特権コンテナが mknod() を使用して特定のデバイスノードを追加できるようにした。これには、ユーザ空間への seccomp トラップがサポートされたカーネルが必要である (5.3 以降)。これは実験的なものである。

**mount=<fstype ; fstype ; ...>とする。**

特定のタイプのファイルシステムのマウントを許可する。これは、mount コマンドで使用するファイル・システム・タイプのリストでなければならない。これは、コンテナのセキュリティに悪影響を及ぼす可能性があることに注意。ループデバイスへのアクセスで、ファイルをマウントすると、デバイス cgroup の mknod 権限を回避できる。NFS ファイルシステムをマウントすると、ホストの I/O が完全にブロックされ、再起動できなくなる可能性がある。

**nesting=<boolean> (デフォルト = 0)**

ネストを許可する。id マッピングを追加した非特権コンテナで使用するのが最適です。ホストの procfs と sysfs の内容をゲストに公開することに注意してください。

**フックスクリプト:<文字列**

コンテナのライフタイムのさまざまなステップで実行されるスクリプト。

**ホスト名:<文字列**

コンテナのホスト名を設定する。

**lock:** <バックアップ | 作成 | 破棄 | ディスク | fstrim | マイグレーション | マウント  
| ロールバック | スナップショット | スナップショット削除>。

コンテナのロック/アンロック。

**メモリ:** <整数> (16 - N) (デフォルト = 512)

コンテナのRAM容量 (MB)。

**mp[n]: ボリューム=] <ボリューム> ,mp=<Path> [,acl=<1|0>] [,backup=<1|0>]  
[,mountoptions=<opt[;opt...]>] [,quota=<1|0>] [,replicate=<1|0>]  
[,ro=<1|0>] [,shared=<1|0>] [,size=<1|0>] [,マウントオプション  
=<opt[;opt...][quota=<1|0>][,replicate=<1|0>][,ro=<1|0>][,shared=<1|0>]  
[,size=<DiskSize>]。**

ボリュームをコンテナ・マウント・ポイントとして使用する。新しいボリュームを割り当てるには、  
特別な構文 STORAGE\_ID:SIZE\_IN\_GiB を使用します。

**acl=<ブール値**

ACL サポートを明示的に有効または無効にする。

**backup=<ブール値>**

マウント・ポイントをバックアップに含めるかどうか（ボリューム・マウント・ポイントにのみ使用）。

**mountoptions=<opt[ ; opt...]>とする。**

rootfs/mps用の追加マウントオプション。

**mp=</パス>**

コンテナ内部から見たマウントポイントへのパス。

**注**

セキュリティ上の理由から、シンボリックリンクを含んではならない。

**quota=<ブール値>**

コンテナ内でユーザークォータを有効にする（zfsサブボリュームではサポートされていません）。

**replicate=<boolean>（デフォルト = 1）**

このボリュームをストレージ・レプリカ・ジョブに含めます。

**ro=<ブール値>**

読み取り専用のマウントポイント

**shared=<boolean>（デフォルト = 0）**

この非ボリュームマウントポイントを全ノードで利用可能としてマークする。

**警告**

このオプションは、マウント・ポイントを自動的に共有するのではなく、共有されていると仮定します！

**size=<ディスクサイズ>**

ボリュームサイズ（読み取り専用値）。

**ボリューム=<ボリューム>**

コンテナにマウントするボリューム、デバイス、またはディレクトリ。

**ネームサーバー:<文字列>**

コンテナの DNS サーバー IP アドレスを設定する。searchdomainもnameserverも設定しなかった場合

、Createは自動的にホストの設定を使用します。

```
net[n]: name=<string> [,bridge=<bridge>] [,firewall=<1|0>]  
[,gw=<GatewayIPv4>] [,gw6=<GatewayIPv6>] [,hwaddr=<XX:XX:XX:XX:XX:  
XX>] [,ip=<(IPv4/CIDR|dhcp|manual)>]  
[,ip6=<(IPv6/CIDR|auto|dhcp|manual)>] [,link_down=<1|0>]  
[,mtu=<integer>] [,rate=<mbps>] [,tag=<integer>]  
[,trunks=<vlanid[;vlanid...]>] [,type=<veth>]
```

コンテナのネットワーク・インターフェースを指定する。

## ブリッジ

ネットワークデバイスを接続するブリッジ。

**ファイアウォール=<ブール**

このインターフェイスのファイアウォール・ルールを使用するかどうかを制御する。

**gw=<ゲートウェイIPv4**

IPv4 トライフィックのデフォルトゲートウェイ。

**gw6=<ゲートウェイIPv6**

IPv6 トライフィックのデフォルトゲートウェイ。

**hwaddr=<XX:XX:XX:XX:XX:XX>。**

I/G (Individual/Group) ビットが設定されていない共通のMACアドレス。

**ip=<(IPv4/CIDR|dhcp|manual)>とする。**

CIDR形式のIPv4アドレス。

**ip6=<(IPv6/CIDR|auto|dhcp|manual)>となります。**

CIDR形式のIPv6アドレス。

**link\_down=<ブール値**

このインターフェイスを切断すべきかどうか（プラグを抜くように）。

**mtu=<整数> (64 - 65535)**

インターフェースの最大転送単位。(lxc.network.mtu)

**name=<文字列**

コンテナ内部から見たネットワークデバイスの名前。(lxc.network.name)

**レート=<mbps**

インターフェイスにレート制限を適用する

**タグ=<整数> (1 - 4094)**

このインターフェースのVLANタグ。

**trunks=<vlanid[,vlanid...]>とする。**

インターフェイスを通過するVLAN ID

**type=<バス**

ネットワークインターフェースのタイプ。

**onboot: <boolean> (デフォルト = 0)**

システム起動時にコンテナを起動するかどうかを指定する。

**ostype: <alpine | archlinux | centos | debian | devuan | fedora | gentoo | nixos | opensuse | ubuntu | unmanaged>.**

OSタイプ。これは、コンテナ内部での設定に使用され、/usr/share/lxc/config/<ostype>.common.conf 内の lxc 設定スクリプトに対応する。値 *unmanaged* は、OS 固有のセットアップをスキップするため に使用できる。

**protection: <ブール値> (デフォルト = 0)**

コンテナの保護フラグを設定する。これにより、CTまたはCTのディスクの削除/更新操作を防ぐことができる。

**rootfs: ボリューム=] <ボリューム> [,acl=<1|0>]  
[,mountoptions=<opt[;opt...]>] [,quota=<1|0>] [,replicate=<1|0>]  
[,ro=<1|0>] [,shared=<1|0>]  
[,size=<1|0>[quota=<1|0>][,replicate=<1|0>][,ro=<1|0>][,shared=<1|0>][,size=<DiskSize>]。**

ボリュームをコンテナ・ルートとして使用する。

#### **acl=<ブール値>**

ACL サポートを明示的に有効または無効にする。

#### **mountoptions=<opt[;opt...]>とする。**

rootfs/mps用の追加マウントオプション。

#### **quota=<ブール値>**

コンテナ内でユーザークォータを有効にする (zfsサブボリュームではサポートされていません)。

#### **replicate=<boolean> (デフォルト = 1)**

このボリュームをストレージ・レプリカ・ジョブに含めます。

#### **ro=<ブール値>**

読み取り専用のマウントポイント

#### **shared=<boolean> (デフォルト = 0)**

この非ボリュームマウントポイントを全ノードで利用可能としてマークする。



#### **警告**

このオプションは、マウント・ポイントを自動的に共有するのではなく、共有されていると仮定します！

#### **size=<ディスクサイズ>**

ボリュームサイズ (読み取り専用値)。

#### **ボリューム=<ボリューム>**

コンテナにマウントするボリューム、デバイス、またはディレクトリ。

#### **searchdomain:<文字列>**

コンテナのDNS検索ドメインを設定します。searchdomainもnameserverも設定しない場合、Createはホストからの設定を自動的に使用します。

**を起動する: `order=J [,up=J] [,down=J ]`**

スタートアップとシャットダウンの動作。Orderは、一般的な起動順序を定義する非負の数値である。シャットダウンは逆の順序で行われます。さらに、次のVMが起動または停止するまでの待ち時間を秒単位で設定できます。

**swap: <整数> (0 - N) (デフォルト = 512)**

コンテナのSWAP量 (MB)。

**タグ:<string>**

コンテナのタグ。これはメタ情報に過ぎない。

**テンプレート: <boolean> (デフォルト = 0)**

テンプレートの有効/無効。

**timezone: <文字列>**

コンテナで使用するタイムゾーン。オプションが設定されていない場合は、何も行われない。ホストのタイムゾーンに合わせるために *host* を設定するか、*/usr/share/zoneinfo/zone.tab* から任意のタイムゾーンオプションを設定することができる。

**tty: <整数> (0 - 6) (デフォルト = 2)**

コンテナで利用可能なttyの数を指定する。

**unprivileged: <boolean> (デフォルト = 0)**

コンテナを非特権ユーザーとして実行させる。(手動で変更してはならない)。

**unused[n] である: [ボリューム=]<ボリューム>**

未使用ボリュームへの参照。これは内部的に使用されるので、手動で変更してはならない。

**ボリューム= <ボリューム>**

現在使用されていないボリューム。

## 11.12 錠前

コンテナの移行、スナップショット、およびバックアップ (*vzdump*) は、影響を受けるコンテナに対する互換性のない同時実行を防ぐためにロックを設定する。このようなロックを手動で削除する必要がある場合もある（停電後など）。

```
# pct アンロック <CTID>
```

**注意**

ロックを設定したアクションがもう実行されていないことを確認している場合のみ、この操作を行ってください。

## 第12章

# ソフトウェア定義ネットワーク

Proxmox VEのSDN（Software-Defined Network）機能は、仮想ゾーンとネットワーク（VNet）の作成を可能にします。この機能により、高度なネットワーキング設定とマルチテナント設定が簡素化されます。

## 12.1 はじめに

Proxmox VE SDN は柔軟でソフトウェア制御のコンフィギュレーションを使い、仮想ゲストネットワークの分離ときめ細かな制御を可能にする。

分離はゾーン、仮想ネットワーク（VNets）、サブネットを通じて管理される。ゾーンは仮想的に分離された独自のネットワーク領域である。VNetはゾーンに属する仮想ネットワークです。サブネットはVNet内のIP範囲です。

ゾーンのタイプによって、ネットワークは異なる挙動を示し、特定の機能、利点、制限を提供する。

SDN のユースケースは、個々のノード上の孤立したプライベートネットワークから、異なる場所にある複数のPVE クラスタにまたがる複雑なオーバーレイネットワークまで多岐にわたる。

クラスタ全体のデータセンタ SDN 管理インターフェイスで VNet を設定した後、各ノードのローカルで共通の Linux ブリッジとして利用でき、VM やコンテナに割り当てることができる。

## 12.2 サポート状況

### 12.2.1 歴史

Proxmox VE SDN スタックは 2019 年から実験的な機能として提供されており、多くの開発者とユーザによって継続的に改善とテストが行われてきた。Proxmox VE 6.2 のウェブインターフェースへの統合により、より広範な統合に向けた重要なマイルストーンが達成された。Proxmox VE 7 のリリースサイクルでは、数多くの改良と機能が追加されました。ユーザーからのフィードバックに基づき、基本的な設計の選択とその実装が非常に健全で安定していることが明らかになりました。従って、'実験的' というラベルを付けることは SDN スタックの状態を正当に評価するものではなかった。Proxmox VE 8 では、ネットワークとインターフェースの管理を Proxmox VE のアクセス制御スタックのコアコンポーネントに昇格させることで、SDN 機能の完全な統合のための基礎を築くという決定がなされた。Proxmox VE 8.1 では、二つの大きなマイルストーンが達成された。第一に、IP アドレス管理 (IPAM) 機能に DHCP の統合が追加された。

## 12.2.2 現在の状況

我々の SDN インストールの様々なレイヤーの現在のサポート状況は以下の通りだ:

- VNet 管理と Proxmox VE スタックとの統合を含むコア SDN は完全にサポートされている。
- 仮想ゲストのDHCP管理を含むIPAMは技術レビュー中です。
- FRRoutingによる複雑なルーティングとコントローラーの統合は技術レビュー中。

## 12.3 インストール

### 12.3.1 SDNコア

Proxmox VE 8.1 以降、Software-Defined Network (SDN) のコアパッケージがデフォルトでインストールされます。

古いバージョンからアップグレードする場合は、各ノードに libpve-network-perl パッケージをインストールする必要があります:

アptoアップデート

```
apt install libpve-network-perl
```

#### 注

Proxmox VEバージョン7.0以上では、デフォルトでifupdown2パッケージがインストールされています。それ以前のバージョンでシステムをインストールした場合は、明示的にifupdown2パッケージをインストールする必要があります。

インストール後、/etc/network/interfの最後に以下の行があることを確認する必要がある。

コンフィギュレーションファイルをすべてのノードに置き、SDN コンフィギュレーションが含まれ、アクティブになるようにする。

```
ソース /etc/network/interfaces.d/*
```

### 12.3.2 DHCP IPAM

PVE内蔵IPアドレス管理スタックへのDHCP統合は、現在DHCPリースを与えるのにdnsmasqを使っています。これは現在オプトインです。

この機能を使うには、各ノードにdnsmasqパッケージをインストールする必要がある：

#### アptoアップデート

```
apt インストール dnsmasq
# デフォルトのインスタンスを無効にする systemctl disable --now dnsmasq
```

### 12.3.3 FRルーティング

Proxmox VE SDN スタックは高度なセットアップのために [FRRouting](#) プロジェクトを使う。これは現在

アプトアップデート

```
apt install frr-pythontools
```

ージをインストールする必要があります:

## 12.4 コンフィギュレーションの概要

設定はデータセンター・レベルのウェブUIで行われ、以下のセクションに分かれている：

- **SDN**: ここで現在のアクティブな SDN 状態の概要が得られ、保留中のすべての変更をクラスタ全体に適用することができます。
- **ゾーン**仮想的に分離されたネットワークゾーンの作成と管理
- **VNets VNets**: 仮想ネットワークブリッジの作成とサブネットの管理

Options "カテゴリでは、SDNセットアップで使用する追加サービスを追加・管理できる。

- **コントローラー**: 複雑なセットアップにおけるレイヤー3ルーティングの制御用
- **DHCP**: IPAM内のゲストにIPを自動的に割り当て、DHCP経由でゲストにリースするDHCPサーバーをゾーンに定義する。
- **IPAM**: ゲストの外部IPアドレス管理を有効にする
- **DNS**: 仮想ゲストのホスト名とIPアドレスを登録するためのDNSサーバーの統合を定義します。

## 12.5 技術と構成

Proxmox VE Software-Defined Network の実装は可能な限り標準的な Linux ネットワークを使っている。最新の Linux ネットワーキングは SDN 実装に必要なほとんど全ての機能を提供し、外部依存の追加を避け、壊れる可能性のあるコンポーネントの総量を減らすことができるからだ。

Proxmox VE SDN 設定は `/etc/pve/sdn` にあり、Proxmox VE 設定ファイルシステムを通して他の全てのクラスタノードと共有されます。これらのコンフィギュレーションは基礎となるネットワークスタックを管理するツールのそれぞれのコンフィギュレーションフォーマット（例えば `ifupdown2` や `frr`）に変換されます。

新しい変更はすぐに適用されるのではなく、まず保留として記録されます。その後、ウェブインターフェイスのメインの SDN 概要パネルで一連の異なる変更を一度に適用することができる。このシステムは様々な変更を単一のアトミックなものとして展開することができる。

SDN は `/etc/pve/sdn` にある `.running-config` と `.version` ファイルを通してロールアウト状態を追跡する。

## 12.6 ゾーン

ゾーンは仮想的に分離されたネットワークを定義する。ゾーンは特定のノードに制限され、ユーザーを特定のゾーンとその中に含まれる VNets に制限するために、パーミッションが割り当てられる。

分離にはさまざまな技術を用いることができる：

- ・ シンプル：孤立ブリッジ。単純なレイヤー3ルーティングブリッジ (NAT)
- ・ VLAN：仮想LANは、LANを細分化する古典的な方法である。
- ・ QinQ：スタックドVLAN (正式名称: IEEE 802.1ad)
- ・ VXLAN：UDPトンネル経由のレイヤー2 VXLANネットワーク
- ・ evpn (bgp evpn)：レイヤ3ルーティングを確立するためのBGP付きVXLAN

### 12.6.1 共通オプション

以下のオプションはすべてのゾーンタイプで利用可能です：

#### ノード

ゾーンと関連するVNetsを配置するノード。

#### アイパム

ゾーン内のIPを管理するためにIPアドレス管理(IPAM)ツールを使用する。オプション、デフォルトはpve。

#### DNS

DNS APIサーバー。オプション。

#### リバースDNS

リバースDNS APIサーバー。オプション。

#### DNSゾーン

DNSドメイン名。<ホスト名>. <ドメイン名>のようなホスト名を登録するのに使われる。DNSゾーンはDNSサーバー上にすでに存在していなければならない。オプション。

### 12.6.2 シンプル・ゾーン

これは最もシンプルなプラグインです。孤立したVNetブリッジを作成します。このブリッジは物理インターフェイ

スにリンクされておらず、VM トライフィックは各ノードのローカルのみになります。NAT またはルーティングされたセットアップで使用できます。

### 12.6.3 VLANゾーン

VLAN プラグインは、既存のローカル Linux または OVS ブリッジを使用してノードの物理インターフェイスに接続します。VLAN プラグインは、VNet で定義された VLAN タギングを使用してネットワークセグメントを分離します。これにより、異なるノード間での VM の接続が可能になります。

VLANゾーン設定オプション：

#### ブリッジ

ノード間の接続を可能にするローカルブリッジまたはOVSスイッチは、各ノードにすでに設定されている。

### 12.6.4 QinQゾーン

QinQはVLANスタッキングとも呼ばれ、分離のためにVLANタグを複数層使用します。QinQゾーンは外側のVLANタグ（サービスVLAN）を定義し、内側のVLANタグはVNetによって定義されます。

---

#### 注

このコンフィギュレーションでは、物理ネットワークスイッチがスタックドVLANをサポートしている必要があります。

---

QinQゾーン設定オプション：

#### ブリッジ

各ローカルノードに設定済みのローカルVLAN対応ブリッジ

#### サービスVLAN

このゾーンのメインVLANタグ

#### サービスVLANプロトコル

802.1q（デフォルト）または 802.1ad サービス VLAN タイプを選択できます。

#### エムティーキュー

タグの二重スタックのため、QinQ VLAN にはさらに 4 バイトが必要です。たとえば、物理インターフェイスの MTU が 1500 の場合 MTU を 1496 に下げる必要があります。

### 12.6.5 VXLANゾーン

VXLANプラグインは、既存のネットワーク（アンダーレイ）の上にトンネル（オーバーレイ）を確立する。

これは、レイヤー2のイーサネットフレームを、デフォルトの宛先ポート4789を使用するレイヤー4のUDPデータグラム内にカプセル化する。

すべてのピア間でUDP接続を有効にするには、アンダーレイネットワークを自分で設定する必要がある。

例えば、パブリック・インターネットの上にVXLANオーバーレイ・ネットワークを作成し、あたかもVMが同じローカル・レイヤ2ネットワークを共有しているかのように見せることができる。

---

**警告**

 VXLAN単体では暗号化は行われません。VXLAN経由で複数のサイトに参加する場合は、サイト間VPNを使用するなどして、サイト間で安全な接続を確立してください。

---

VXLANゾーン設定オプション:

#### ピアアドレスリスト

VXLANゾーン内の各ノードのIPアドレスのリスト。このIPアドレスで到達可能な外部ノードでもよい。クラスタ内すべてのノードをここに記載する必要があります。

#### エム

ティ VXLANカプセル化は50バイトを使用するので、MTUは発信物理インターフェースより50バイト低くする必

一ユ 要がある。

—

## 12.6.6 EVPNゾーン

EVPNゾーンは、複数のクラスタにまたがるルーティング可能なレイヤ3ネットワークを構築します。これはVPNを確立し、ルーティングプロトコルとしてBGPを利用することで実現される。

EVPNのVNetはエニーキャストIPアドレスおよび/またはMACアドレスを持つことができます。ブリッジIPは各ノードで同じで、仮想ゲストはゲートウェイとしてこのアドレスを使用できます。

ルーティングはVRF（Virtual Routing and Forwarding）インターフェイスを通じて、異なるゾーンのVNet間で機能する。

EVPNゾーン設定オプション:

#### VRF VXLAN ID

VNet間の専用ルーティング相互接続に使用されるVXLAN-ID。VNetsのVXLAN-IDとは異なる必要があります。

#### コントローラー

このゾーンで使用するEVPNコントローラ。(コントローラのプラグインのセクションを参照)。

#### VNet MACアドレス

このゾーン内のすべてのVネットに割り当てられるエニーキャストMACアドレス。定義されていない場合は自動生成されます。

#### 出口ノード

EVPNネットワークから実ネットワークを経由する出口ゲートウェイとして設定されるノード。設定さ

れたノードはEVPNネットワークのデフォルトルートをアナウンスします。オプション。

## 一次出口ノード

複数の出口ノードを使用する場合は、すべてのノードでロードバランシングを行う代わりに、このプライマリ出口ノードを介してトラフィックを強制します。オプションですが、SNATを使用する場合や、アップストリーム・ルーターがECMPをサポートしていない場合は必要です。

## ローカル・ルーティング

これは、出口ノードからVM/CTサービスに到達する必要がある場合の特別なオプションです。(デフォルトでは、出口ノードはリアルネットワークとEVPNネットワーク間のトラフィック転送のみを許可します)。オプション。

### サブネットの広告

EVPNネットワーク内のフルサブネットをアナウンスします。サイレントVM/CTがある場合(例えば、複数のIPがあり、エニーキャストゲートウェイがこれらのIPからのトラフィックを見ない場合、IPアドレスはEVPNネットワーク内に到達できない)。オプション。

### ARP ND抑制を無効にする

ARPやND（Neighbor Discovery）パケットを抑制しないこと。これは、VMでフローティングIPを使用している場合に必要です（IPアドレスとMACアドレスがシステム間で移動している）。オプション。

### ルートターゲット・インポート

外部EVPNルートターゲットのリストをインポートできるようにします。クロスDCまたは異なるEVPNネットワークの相互接続に使用します。オプションです。

#### エム

ティ  
ーユ VXLAN カプセル化は 50 バイトを使用するため、MTU は発信物理インターフェイスの最大 MTU よりも 50 バイト小さい必要がある。オプション、デフォルトは 1450。

—

## 12.7 Vネット

SDN GUI で仮想ネットワーク（VNet）を作成すると、各ノードで同じ名前のローカルネットワークインターフェースが利用可能になります。ゲストを VNet に接続するには、ゲストにインターフェースを割り当て、それに応じて IP アドレスを設定します。

ゾーンによって、これらのオプションは異なる意味を持ち、本書の各ゾーンのセクションで説明されています。



#### 警告

現状では、一部のオプションは効果がなかったり、特定のゾーンでは機能しないことがある。

---

VNet設定オプション：

---

**身分証明書**

VNetを識別するための最大8文字のID。

**コメント**

より説明的な識別子。インターフェースのエイリアスとして割り当てられる。オプション

**ゾーン**

このVNetの関連ゾーン

**タグ**

固有のVLANまたはVXLAN ID

**VLAN対応**

インターフェイスの `vlan-aware` オプションを有効にし、ゲストでの設定を可能にします。

## 12.8 サブネット

サブネットは、CIDRネットワークアドレスで記述される特定のIP範囲を定義します。各VNetは1つ以上のサブネットを持つことができます。

サブネットは次のような用途に使われる：

- ・特定のVNetで定義できるIPアドレスを制限する
- ・レイヤー3ゾーンのVNetにルート/ゲートウェイを割り当てる
- ・レイヤー3ゾーンのVNetでSNATを有効にする
- ・IPAMプラグインによる仮想ゲスト(VMまたはCT)へのIP自動割り当て
- ・DNSプラグインによるDNS登録

サブネット・ゾーンにIPAMサーバーが関連付けられている場合、サブネット・プレフィックスは自動的にIPAMIに登録される。

サブネットの設定オプション：

### 身分証明書

CIDRネットワークアドレス（例：10.0.0.0/8

### ゲートウェイ

ネットワークのデフォルトゲートウェイのIPアドレス。レイヤー3ゾーン（Simple/EVPNプラグイン）では、VNet上に配置されます。

### SNAT

ソースNATを有効にすると、VNet内のVMがノードの送信インターフェースにパケットを転送することで、外部ネットワークに接続できるようになります。EVPNゾーンでは、転送はEVPNゲートウェイノードで行われます。オプション。

### DNSゾーンプレフィックス

<ホスト名>.prefix.<ドメイン名>のように、ドメイン登録にプレフィックスを追加する。

## 12.9 コントローラー

ゾーンによっては、制御プレーンとデータプレーンを分離して実装しているため、VNetの制御プレーンを管理する外部コントローラが必要になります。

現在、EVPNゾーンだけが外部コントローラを必要とする。

## 12.9.1 EVPNコントローラー

EVPNゾーンでは、コントロールプレーンを管理するために外部コントローラが必要です。EVPN コントローラープラグインは、Free Range Routing (frr) ルータを設定します。

EVPNコントローラを有効にするには、EVPNゾーンに参加するすべてのノードにfrrをインストールする必要があります。

```
apt install frr frr-pythontools
```

EVPN コントローラーの設定オプション:

### ASN #

一意のBGP ASN番号。プライベートASN番号（64512～65534、4200000000～4294967294）を使用することを強くお勧めします。そうしないと、間違ってグローバルルーティングを壊してしまう可能性があるからです。

### ピアーズ

EVPNゾーンの全ノードのIPリスト。(外部ノードまたはルートリフレクタサーバでもよい)

## 12.9.2 BGPコントローラ

BGPコントローラはゾーンによって直接使われることはありません。BGPピアを管理するためにFRRを設定するために使うことができます。

BGP-EVPNでは、ノードごとに異なるASNを定義してEBGPを行うことができます。また、外部のBGPピアにEVPNルートをエクスポートするためにも使用できます。

---

### 注

デフォルトでは、シンプルなフルメッシュEVPNの場合、BGPコントローラを定義する必要はありません。

---

BGPコントローラの設定オプション:

### ノード

このBGPコントローラのノード

### ASN #

一意のBGP ASN番号。(64512)の範囲のプライベートASN番号を使用することを強くお勧めします。

---

- 65534)または(4200000000 - 4294967294)でなければ、間違ってグローバルルーティングを壊してしまう可能性があるからです。

## ピア

基礎となるBGPネットワークを使用して通信したいピアIPアドレスのリスト。

## EBGP

ピアのリモートASが異なる場合、EBGPを有効にします。

## ループバックインターフェース

EVPNネットワークのソースとしてループバックまたはダミーインターフェースを使用する（マルチパスの場合）。

## エブグップ・ミューティホップ

ピアが直接接続されていない場合やループバックを使用している場合は、ピアに到達するためのホップ数を増やす。

### bgp-multipath-as-path-relax

ピアのASNが異なる場合、ECMPを許可する。

## 12.9.3 ISISコントローラー

ISISコントローラはゾーンで直接使用されない。ISIS ドメインに EVPN ルートをエクスポートするために FRR を設定するために使用できる。

ISIS コントローラ構成オプション:

### ノード

このISISコントローラのノード。

### ドメイン

ISIS独自のドメイン。

### ネットワーク・エンティティ名

このノードを識別する一意のISISネットワークアドレス。

### インターフェイス

ISIS が使用する物理インターフェースのリスト。

### ループバック

EVPNネットワークのソースとしてループバックまたはダミーアンターフェースを使用する（マルチパスの場合）。

## 12.10 アイパム

IPアドレス管理(IPAM)ツールはネットワーク上のクライアントのIPアドレスを管理する。Proxmox VE の SDN は IPAM を使って、例えば新しいゲストのために空いている IP アドレスを見つける。

1つのIPAMインスタンスは、1つまたは複数のゾーンに関連付けることができる。

## 12.10.1 PVE IPAM プラグイン

Proxmox VEクラスタのデフォルトの組み込みIPAMです。

PVE IPAM Plugin の現在のステータスは、データセンター設定の SDN セクションにある IPAM パネルで確認できます。この UI は IP マッピングの作成、更新、削除に使用できます。[DHCP](#)機能と併用すると特に便利です。

DHCP を使用している場合、IPAM パネルを使用して特定の VM に対してリースを作成または編集することができます。DHCPを使用しているVMのIPを編集する場合は、ゲストに新しいDHCPリースを取得させる必要があります。これは通常、ゲストのネットワークSTACKをリロードするか、ゲストを再起動することで行うことができます。

## 12.10.2 NetBox IPAM プラグイン

NetBoxは、オープンソースのIPアドレス管理（IPAM）およびデータセンター・インフラストラクチャ管理（DCIM）ツールです。

NetBox と Proxmox VE SDN を統合するには、以下の説明に従って NetBox で API トークンを作成します:  
<https://docs.netbox.d ja/stable/integrations/rest-api/#tokens>

NetBoxの設定プロパティは以下の通り：

### URL

NetBox REST API エンドポイント: `http://yournetbox.domain.com/api`

### トークン

APIアクセストークン

## 12.10.3 phplIPAM プラグイン

phplIPAM では、"アプリケーション" を作成し、アプリケーションに admin 権限を持つ API トークンを追加する必要があります。phplIPAMの設定プロパティは次のとおりです：

### URL

REST-API エンドポイント: `http://phpipam.domain.com/api/<appname>/`

### トークン

APIアクセストークン

### セクション

整数の ID。セクションは phplIPAM のサブネットのグループです。デフォルトのインストールでは `sectionid=1` を使用します。

お客様のために

## 12.11 DNS

Proxmox VE SDN の DNS プラグインは、ホスト名と IP アドレスを登録するための DNS API サーバを定義するために使用します。DNS 設定は 1 つ以上のゾーンに関連付けられ、ゾーンに設定された全てのサブネット IP に対して DNS 登録を提供します。

### 12.11.1 PowerDNSプラグイン

<https://doc.powerdns.com/authoritative/http-api/index.html>

PowerDNSの設定で、ウェブサーバーとAPIを有効にする必要があります：

```
api=はい
api-key=arandomgeneratedstring
webserver=yes
ウェブサーバー・ポート=8081
```

PowerDNSの設定オプションは以下の通り：

**url**

REST API エンドポイント : <http://yourpowerdnserver.domain.com:8081/api/v1/servers/localhost>

**キー**

APIアクセスキー

**トル**

レコードのデフォルトTTL

## 12.12 ディーエイチシーピー

Proxmox VE SDN の DHCP プラグインを使用すると、ゾーンに DHCP サーバを自動的に配置できます。DHCP 範囲が設定されているゾーン内のすべてのサブネットに DHCP を提供します。現在利用可能な DHCP 用のバックエンドプラグインは dnsmasq プラグインだけです。

DHCPプラグインは、VM/CTに新しいネットワークインターフェイスを追加する際に、ゾーンに設定されたIPAMプラグインでIPを割り当てることで機能します。IPAMの設定方法については、[ドキュメントの各セクション](#)を参照してください。

VMが起動すると、ゾーンのDHCPプラグインにMACアドレスとIPのマッピングが作成される。ネットワーク・インターフェイスが削除されるか、VM/CTが破棄されると、IPAMとDHCPサーバーのエントリも削除される。

---

**注**

現在、一部の機能（IPマッピングの追加/編集/削除）は、[PVE IPAMプラグイン](#)を使用した場合にのみ利用可能です。

---

### 12.12.1 構成

Web UIでゾーンの自動DHCPを有効にするには、[ゾーン]パネルを使用し、ゾーンの詳細オプションでDHCPを有効にします。

---

**注**

現在、自動DHCPをサポートしているのはシンプル・ゾーンのみです。

---

ゾーンで自動DHCPを有効にした後、ゾーン内のサブネットにDHCP範囲を設定する必要があります。そのためには、[Vnets]パネルに行き、DHCP範囲を設定したいサブネットを選択します。編集ダイアログでは、それぞれのタブでDHCP範囲を設定できます。または、以下のCLIコマンドを使用して、サブネットのDHCP範囲を設定することもできます：

```
pvsh set /cluster/sdn/vnets/<vnet>/subnets/<subnet>
-dhcp-range start-address=10.0.1.100,end-address=10.0.1.200
-dhcp-range start-address=10.0.2.100,end-address=10.0.2.200
```

また、サブネットにゲートウェイを設定する必要があります - そうしないと自動DHCPは機能しません。そうしないと、自動DHCPは機能しません。DHCPプラグインは、設定された範囲でのみIPAMにIPを割り当てます。

[dnsmasq DHCPプラグインのインストール手順](#)も忘れないでください。

## 12.12.2 プラグイン

### Dnsmasqプラグイン

現在、これは唯一のDHCPプラグインであり、したがってゾーンでDHCPを有効にするときに使用されるプラグインである。

#### インストール

インストールについては、[DHCP IPAMのセクション](#)を参照のこと。

#### 構成

プラグインは dnsmasq がデプロイされるゾーンごとに新しい systemd サービスを作成します。サービス名は dnsmasq@<zone> です。このサービスのライフサイクルは DHCP プラグインによって管理されます。

プラグインは、/etc/dnsmasq.d/<zone>フォルダに以下の設定ファイルを自動的に生成します：

##### 00-default.conf

dnsmasqインスタンスのデフォルトのグローバル構成が含まれます。

##### 10-<zone>-<subnet\_cidr>.conf

このファイルは、DHCP経由で設定されるべきDNSサーバーなど、サブネットの特定のオプションを設定する。

##### 10-<zone>-<subnet\_cidr>.ranges.conf

このファイルは、dnsmasqインスタンスのDHCP範囲を構成します。

#### エーテル

このファイルには、IPAMプラグインからのMACアドレスとIPマッピングが含まれています。これらのマッピングを上書きするには、dnsmasqプラグインによって上書きされるため、このファイルを編集するのではなく、それぞれのIPAMプラグインを使用してください。

上記のファイルはDHCPプラグインによって管理されるため、編集してはいけません。dnsmasqの設定をカスタマイズするには、設定フォルダに追加のファイル（90-custom.confなど）を作成します。

設定ファイルは順番に読み込まれるので、カスタム設定ファイルに適切な名前を付けることで、設定ディレクトリの順番を制御することができます。

DHCPリースは/var/lib/misc/dnsmasq.<zone>.leasesファイルに格納される。

PVE IPAMプラグインを使用すると、DHCPリースの更新、作成、削除ができます。詳細については、PVE IPAMプラグインのドキュメントを参照してください。他のIPAMプラグインでは、DHCPリースの変更は現在サポートされていません。

## 12.13 例

このセクションは一般的な SDN の使用例に合わせた複数の設定例を示す。利用可能な設定オプションの理解を深めるために追加的な詳細を提供し、具体的な実装を提供することを目的としている。

### 12.13.1 シンプルゾーンの例

シンプル・ゾーン・ネットワークは、単一のホスト上のゲストが相互に接続するための隔離されたネットワークを作成します。

---

#### チップ

ゲスト間の接続は、すべてのゲストが同じホストに存在するが、他のノードに到達できない場合に可能である。

---

- simpleという名前のシンプルなゾーンを作る。
- VNet名vnet1を追加します。
- ゲートウェイとSNATオプションを有効にしてサブネットを作成する。
- これでノード上にネットワークブリッジvnet1が作成されます。このブリッジをネットワークに参加するゲストに割り当て、IPアドレスを設定します。

2つのVMのネットワーク・インターフェイスのコンフィギュレーションは、10.0.1.0/24ネットワーク経由で通信で

```
allow-hotplug ens19
iface ens19 inet
static
    アドレス 10.0.1.14/24
allow-hotplug ens19
iface ens19 inet
static
    アドレス 10.0.1.15/24
```

きるよう、次のようになる。

### 12.13.2 ソースNATの例

シンプル・ネットワーク・ゾーンでゲストの発信接続を許可したい場合、シンプルゾーンにはソースNAT (SNAT) オプションがある。

上記の設定から、VNet vnet1にサブネットを追加し、ゲートウェイIPを設定し、SNATオプションを有効にします。

---

サブネット：

172.16.0.0/24 ゲートウェイ

エイ：172.16.0.1 SNAT

：チェック済み

ゲストでは、サブネットのIPレンジ内で静的IPアドレスを設定する。

ノード自身はゲートウェイIP 172.16.0.1でこのネットワークに参加し、サブネット範囲内のゲストのためのNATゲートウェイとして機能する。

### 12.13.3 VLAN設定例

異なるノード上のVMが分離されたネットワークを介して通信する必要がある場合、VLANゾーンはVLANタグを使用してネットワークレベルの分離を可能にします。

```
ID: myvlanzone  
ブリッジ: vmbr0
```

myVlanzoneという名前のVLANゾーンを作成します：

```
ID: myvnet1ゾーン  
: myvlanzoneタグ  
: 10
```

VLANタグ10と先に作成したmyvlanzoneでmyvnet1というVNetを作成します。

メイン SDN パネルから設定を適用し、各ノードにローカルに VNet を作成する。node1 上

に Debian ベースの仮想マシン (vm1) を作成し、myvnet1 上に vNIC を置く。

このVMには以下のネットワーク構成を使用する：

```
auto eth0  
iface eth0 inet static  
    アドレス 10.0.3.100/24
```

ノード2上に、vm1と同じVNet myvnet1上にvNICを持つ2台目の仮想マシン (vm2) を作成します。

```
auto eth0  
iface eth0 inet static  
    アドレス 10.0.3.101/24
```

これに従って、そのネットワークを使用して両方のVM間でpingを送れるようになるはずだ。

### 12.13.4 QinQセットアップの例

この例では、2つのQinQゾーンを設定し、各ゾーンに2つのVMを追加して、VLANタグのレイヤーを追加することで、より分離されたVLANを設定できることを示します。

この構成の典型的なユースケースは、ホスティング・プロバイダーがVM通信のために顧客に隔離されたネットワークを提供するが、VMを他の顧客から隔離する場合である。

ID: qinqzone1 ブリッジ

ジ: vmbr0 サービス

VLAN: 20

サービスVLAN 20でqinqzone1というQinQゾーンを作成します。

ID: qinqzone2 ブリッジ

ジ: vmbr0 サービス

VLAN: 30

サービスVLAN 30でqinqzone2という別のQinQゾーンを作成します。

先に作成したqinqzone1ゾーンに、VLAN-ID 100のmyVnet1というVNetを作成する。

```
ID: qinqvnet1  
Zone: qinqzone1  
Tag: 100
```

qinqzone2ゾーンにVLAN-ID 100のmyVnet2を作成します。

```
ID: qinqvnet2  
Zone: qinqzone2  
Tag: 100
```

メインの SDN ウェブインターフェースパネルで設定を適用して、各ノードにローカルに VNets を作成します。

4台のDebianベースの仮想マシン(vm1, vm2, vm3, vm4)を作成し、vm1とvm2にブリッジqinqvnet1で、vm3とvm4にブリッジqinqvnet2でネットワークインターフェイスを追加します。

VMの内部で、/etc/network/interfacesなどでインターフェースのIPアドレスを設定する：

```
auto eth0  
iface eth0 inet static  
    アドレス 10.0.3.101/24
```

4つのVMのIPアドレスが10.0.3.101から10.0.3.104の範囲になるように設定します。

これで、VM vm1とvm2の間、およびvm3とvm4の間でpingが送れるようになるはずだ。しかし、vm1もvm2もvm3やvm4は別のゾーンにあり、サービスVLANも異なるため、pingを打つことができない。

### 12.13.5 VXLANセットアップの例

この例では、ノードIPアドレスが192.168.0.1、192.168.0.2、192.168.0.3の3つのノードを持つクラスタを想定しています。

myvxlanzone という名前の VXLAN ゾーンを作成し、ノードからのすべての IP をピアアドレスリストに追加します。デフォルトのMTU 1450を使用するか、適宜設定します。

```
ID: myvxlanzone  
ピアのアドレスリスト192.168.0.1,192.168.0.2,192.168.0.3
```

先に作成したVXLANゾーンmyvxlanzoneを使用して、vxvnet1というVNetを作成します。

```
ID: vxvnet1  
Zone:  
myvxlanzone Tag:  
100000
```

メイン SDN ウェブインターフェースパネルで設定を適用し、各ノードにローカルに VNet を作成します

。node1 上に Debian ベースの仮想マシン (vm1) を作成し、vxvnet1 上に vNIC を作成します。

このVMには以下のネットワーク・コンフィギュレーションを使用する（MTUが低いことに注意）。

```
auto eth0
iface eth0 inet static
    アドレス 10.0.3.100/24
    mtu 1450
```

ノード3上に、vm1と同じVNet vxvnet1上にvNICを持つ2台目の仮想マシン（vm2）を作成します。

```
auto eth0
iface eth0 inet static
    アドレス 10.0.3.101/24
    mtu 1450
```

そうすると、vm1とvm2の間でpingが打てるようになるはずだ。

### 12.13.6 EVPNの設定例

この例では、IPアドレスが192.168.0.1、192.168.0.2、192.168.0.3の3つのノード（node1、node2、node3）を持つクラスタを想定しています。

プライベートASN番号と上記のノードアドレスをピアとして使用して、EVPNコントローラを作成します。

```
ID: myevpnctl
ASN#: 65000
ピア192.168.0.1,192.168.0.2,192.168.0.3
```

myevpnzoneという名前のEVPNゾーンを作成し、先に作成したEVPN-コントローラーを割り当て、次のように定義します。

node1とnode2を終了ノードとする。

```
ID: myevpnzone
VRF VXLAN タグ: 10000
コントローラ:
myevpnctl MTU: 1450
VNet MACアドレス: 32:f4:05:fe:6c:0a
終了ノード: ノード1, ノード2
```

EVPNゾーンmyevpnzoneを使用して、myvnet1という最初のVNetを作成します。

```
ID: myvnet1 ゾー
ン: myevpnzone
タグ: 11000
```

myvnet1にサブネットを作成する:

```
サブネット:
10.0.1.0/24 ゲートウ
エイ: 10.0.1.1
```

同じEVPNゾーンmyevpnzoneを使用して、myvnet2という2番目のVNetを作成する。

ID: myvnet2 ゾー

ン: myevpnzone

タグ: 12000

myvnet2に別のサブネットを作成する:

サブネット:

10.0.2.0/24 ゲートウ

エイ: 10.0.2.1

メイン SDN ウェブインターフェースパネルからコンフィグレーションを適用して、各ノードにローカルに VNets を作成し、FRR コンフィグレーションを生成します。

myvnet1にvNICを持つDebianベースの仮想マシン（vm1）をnode1上に作成しま

```
auto eth0
iface eth0 inet static
    アドレス 10.0.1.100/24
    ゲートウェイ 10.0.1.1
    ムチュ1450
```

もう1つのVNet myvnet2上にvNICを持つ2つ目の仮想マシン（vm2）をnode2上に作成しま

```
auto eth0
iface eth0 inet static
    アドレス 10.0.2.100/24
    ゲートウェイ 10.0.2.1
    ムチュ1450
```

これでvm1からvm2に、vm1からvm2にpingを送ることができるはずだ。

ゲートウェイでないnode3のvm2から外部IPにpingを打つと、パケットは設定されたmyvnet2ゲートウェイに行き、その後、出口ノード（node1またはnode2）にルーティングされ、そこからnode1またはnode2に設定されたデフォルトゲートウェイを経由してそれらのノードから離脱します。

---

## 注

外部ゲートウェイのnode1とnode2に10.0.1.0/24と10.0.2.0/24ネットワークのリバースルートを追加して、パブリックネットワークがリプライできるようにする必要があります。

---

外部BGPルーターを設定した場合、BGP-EVPNルート（この例では10.0.1.0/24と10.0.2.0/24）は動的にアナウンスされます。

## 12.14 備考

### 12.14.1 複数のEVPN出口ノード

---

複数のゲートウェイノードがある場合、`rp_filter` (Strict Reverse Path Filter)オプションを無効にしてください。

`etc/sysctl.conf`に以下を追加する:

```
net.ipv4.conf.default.rp_filter=0  
net.ipv4.conf.all.rp_filter=0
```

### 12.14.2 VXLAN IPSEC暗号化

VXLANの上にIPSEC暗号化を追加するために、この例ではstrongswanを使用する方法を示します。

暗号化に対応するためには、IPv4の場合は60バイト、IPv6の場合は80バイト、MTUを減らす必要がある。

つまり、デフォルトの実MTU 1500の場合、1370のMTUを使う必要がある ( $1370 + 80 \text{ (IPSEC)} + 50$ )

(VXLAN) == 1500) ホストにstrongswanをインストールする。  
apt install strongswan

`etc/ipsec.conf` に設定を追加します。VXLAN UDPポート4789からのトラフィックだけを暗号化する必要がある。

```
conn %default
    ike=aes256-sha1-modp1024!    # 最速だがそこそこ安全な暗号
    esp=aes256-sha1
!
leftfirewall=yes                # これはProxmox VEを使用する際に必要です。
.
。
コネクション出力ウォールルール
rightsubnet=%dynamic[udp/4789]
right=%any
type=transport
authby=psk
auto=route
```

## コネクション入力

```
leftsubnet=%dynamic[udp/4789]
type=transport
authby=psk
auto=ルート
```

で事前共有キーを生成する：

```
openssl rand -base64 128
```

そして、そのキーを/etc/ipsec.secretsに追加し、ファイルの内容が次のようになるようとする：

:PSK <generatedbase64key>

VXLANネットワークに参加しているすべてのノードにPSKとコンフィグレーションをコピーします。

## 第13章

# Proxmox VE ファイアウォール

Proxmox VE Firewallは、ITインフラストラクチャを保護する簡単な方法を提供します。クラスタ内のすべてのホストのファイアウォールルールを設定したり、仮想マシンやコンテナのルールを定義したりできます。ファイアウォールマクロ、セキュリティグループ、IPセット、エイリアスなどの機能が、このタスクを容易にします。

すべてのコンフィギュレーションはクラスタ・ファイル・システムに保存されますが、`iptables`ベースのファイアウォール・サービスは各クラスタ・ノード上で実行されるため、仮想マシン間で完全に分離されます。また、このシステムは分散型であるため、中央のファイアウォール・ソリューションよりもはるかに高い帯域幅を提供します。

ファイアウォールはIPv4とIPv6をフルサポートしています。IPv6のサポートは完全に透過的で、デフォルトで両方のプロトコルのトラフィックをフィルタリングします。そのため、IPv6用に別のルールセットを維持する必要はありません。

### 13.1 ゾーン

Proxmox VE ファイアウォールはネットワークを以下の論理ゾーンにグループ化します：

#### ホスト

クラスタノードから/へのトラフィック

#### VM

特定のVMから/へのトラフィック

各ゾーンに対して、受信および/または送信トラフィックのファイアウォールルールを定義することができます。

## 13.2 設定ファイル

ファイアウォール関連の設定はすべてproxmoxクラスタファイルシステムに保存されます。そのため、これらのファイルは自動的にすべてのクラスタノードに配布され、`pve-firewall`サービスは変更があったときに自動的に基礎となる`iptables`ルールを更新します。

GUI（データセンター → ファイアウォール、ノード → ファイアウォール）を使って何でも設定できますし、好みのエディタを使って設定ファイルを直接編集することもできます。

ファイアウォール設定ファイルには、キーと値のペアのセクションがあります。で始まる行と空白行はコメントとみなされます。セクションは [ と ] で囲まれたセクション名を含むヘッダー行で始まります。

### 13.2.1 クラスターワイドセットアップ

クラスタ全体のファイアウォール設定は、以下の場所に保存されます：

/etc/pve/firewall/cluster.fw

コンフィギュレーションには以下のセクションを含めることができます：

#### [オプション]

クラスタ全体のファイアウォールオプションを設定するために使用します。

**ebtables: <boolean> (デフォルト = 1)**

クラスタ全体でebtablesルールを有効にします。

**enable: <整数> (0 - N)**

ファイアウォールクラスター全体を有効または無効にします。

**log\_ratelimit: [enable=]<1|0> [,burst=<integer>] [,rate=<rate>].**

ログ・レイトリミットの設定

**burst=<整数> (0 - N) (デフォルト = 5)**

レートが適用される前に常に記録されるパッケージの初期バースト

**enable=<boolean> (デフォルト = 1)**

ログのレート制限を有効または無効にする

**rate=<レート> (デフォルト=1/秒)**

バーストバケツが補充される頻度

**policy\_in: <ACCEPT | DROP | REJECT>.**

インプットポリシー。

**policy\_out: <ACCEPT | DROP | REJECT>.**

アウトプットの方針。

#### [ルール]

このセクションには、全ノードのクラスタ全体のファイアウォールルールが含まれます。

**[IPSET <名前>]**。  
クラスタ全体のIPセット定義。

**[GROUP <名前>]**。  
クラスタ全体のセキュリティグループの定義。

**[ALIASES]**  
クラスタ全体のエイリアス定義。

## ファイアウォールの有効化

ファイアウォールはデフォルトでは完全に無効になっているので、ここで有効オプションを設定する必要がある：

### [オプション]

```
# enable firewall (クラスタ全体の設定、デフォルトは無効) enable: 1
```



### 重要

ファイアウォールを有効にすると、デフォルトですべてのホストへのトラフィックがブロックされます。唯一の例外は、ローカルネットワークからのWebGUI(8006)とssh(22)です。

リモートから Proxmox VE ホストを管理する場合は、リモート IP から Web GUI (ポート 8006) へのトラフィックを許可するルールを作成する必要があります。また、ssh(ポート22)やSPICE(ポート3128)も許可する必要があります。

### チップ

ファイアウォールを有効にする前に、Proxmox VE ホストの1つにSSH接続を開いてください。そうすれば、何か問題が発生した場合でもホストにアクセスできます。

この作業を簡略化するには、代わりに「management」というIPSセットを作成し、そこにすべてのリモートIPを追加します。これにより、リモートからGUIにアクセスするために必要なファイアウォールルールがすべて作成される。

### 13.2.2 ホスト固有の設定

ホスト関連のコンフィギュレーションを読み込む：

```
/etc/pve/nodes/<nodename>/host.fw
```

cluster.fw設定からルールを上書きしたい場合に便利です。また、ログの冗長度を上げたり、ネットフィルタ関連のオプションを設定することもできます。コンフィギュレーションには以下のセクションを含めることができます：

### [オプション]

ホスト関連のファイアウォールオプションを設定する。

**enable:<ブール値>**

ホストファイアウォールルールを有効にする。

**log\_level\_in:<alert | crit | debug | emerg | err | info | nolog | notice | warning>.**

受信トラフィックのログレベル。

**log\_level\_out:<alert | crit | debug | emerg | err | info | nolog | notice | warning>.**

発信トラフィックのログレベル。

**log\_nf\_conntrack: <boolean> (デフォルト = 0)**

conntrack情報のロギングを有効にする。

**ndp: <ブール値> (デフォルト = 0)**

NDP (Neighbor Discovery Protocol) を有効にする。

**nf\_conntrack\_allow\_invalid: <boolean> (デフォルト = 0)**

コネクション・トラッキングで無効なパケットを許可する。

**nf\_conntrack\_helpers: <string> (デフォルトは `` )**

特定のプロトコルの conntrack ヘルパーを有効にします。対応プロトコル: amanda、ftp、irc、netbios-ns、pptp、sane、sip、snmp、tftp

**nf\_conntrack\_max: <整数> (32768 - N) (デフォルト = 262144)**

追跡接続の最大数。

**nf\_conntrack\_tcp\_timeout\_established: <整数> (7875 - N) (デフォルト = 432000)**

CONNTRACK がタイムアウト。

**nf\_conntrack\_tcp\_timeout\_syn\_recv: <整数> (30 - 60) (デフォルト = 60)**

Conntrack syn recv タイムアウト。

**nftables: <ブール値> (デフォルト = 0)**

nftablesベースのファイアウォールを有効にする (技術プレビュー)

**nosmurfs: <ブール値>**

SMURFSフィルターを有効にする。

**protection\_synflood: <boolean> (デフォルト = 0)**

シン・フラッド・プロテクションを有効にする

**protection\_synflood\_burst: <整数> (デフォルト = 1000)**

ip srcによるシンフラッド・プロテクション・レート・バースト。

**protection\_synflood\_rate: <整数>** (デフォルト = 200)

ip srcによるシン・フラッド・プロテクション・レート syn/sec。

**smurf\_log\_level: <alert | crit | debug | emerg | err | info | nolog**

| お知らせ | 警告

SMURFSフィルターのログレベル。

**tcp\_flags\_log\_level: <alert | crit | debug | emerg | err | info | nolog | notice | warning>.**

不正なtcpフラグフィルターのログレベル。

**tcpflags: <boolean> (デフォルト = 0)**

TCPフラグの不正な組み合わせをフィルタリングする。

**[ルール]**

このセクションにはホスト固有のファイアウォールルールが含まれる。

### 13.2.3 VM/コンテナの構成

VMのファイアウォール設定を読み込む:

```
/etc/pve/firewall/<VMID>.fw
```

そして以下のデータを含んでいる:

**[オプション]**

VM/Container 関連のファイアウォールオプションを設定する。

**dhcp: <ブール値> (デフォルト = 0)**

DHCPを有効にする。

**enable: <ブール値> (デフォルト = 0)**

ファイアウォールルールを有効/無効にする。

**ipfilter: <ブール値>**

デフォルトのIPフィルタを有効にする。これは、すべてのインターフェイスに空の ipfilter-net<id> ipset を追加することと同じです。このようなipsetは、IPv6リンクのローカルアドレスをインターフェイスのMACアドレスに由来するものに制限するような、まともなデフォルトの制限を暗黙的に含んでいる。コンテナに対しては、設定されたIPアドレスが暗黙のうちに追加される。

**log\_level\_in: <alert | crit | debug | emerg | err | info | nolog | notice | warning>.**

受信トラフィックのログレベル。

**log\_level\_out: <alert | crit | debug | emerg | err | info | nolog | notice | warning>.**

発信トラフィックのログレベル。

**macfilter: <boolean> (デフォルト = 1)**

MACアドレスフィルターの有効/無効。

**ndp: <ブール値> (デフォルト = 0)**

NDP (Neighbor Discovery Protocol) を有効にする。

**policy\_in: <ACCEPT | DROP | REJECT>**

インプットポリシー。

**policy\_out: <ACCEPT | DROP | REJECT>。**

アウトプットの方針。

**radv: <ブール値**

ルーター広告の送信を許可する。

### [ルール]

このセクションには、VM/コンテナファイアウォールルールが含まれます。

**[IPSET <名前>]。**

IPセットの定義。

**[ALIASES]**

IPエイリアスの定義。

## VMとコンテナのファイアウォールを有効にする

各仮想ネットワーク・デバイスはそれぞれファイアウォール有効フラグを持っている。そのため、インターフェイスごとにファイアウォールを選択的に有効にすることができます。これは、一般的なファイアウォール有効オプションに加えて必要です。

### 13.3 ファイアウォールルール

ファイアウォール・ルールは、方向 (INまたはOUT) とアクション (ACCEPT、DENY、REJECT) で構成される。マクロ名を指定することもできます。マクロには定義済みのルールとオプションのセットが含まれます。ルールの前に | を付けると、ルールを無効にできます。

#### ファイアウォールルールの構文

[ルール]

ディレクション・アクション [オプション]

| DIRECTION ACTION [OPTIONS] # 無効ルール

DIRECTION MACRO(ACTION) [OPTIONS] # 定義済みのマクロを使う

以下のオプションを使用して、ルールのマッチを絞り込むことができる。

**--dest <文字列**

パケットの宛先アドレスを制限する。これは、単一のIPアドレス、IPセット(*+ipsetname*)、またはIPエイリアスの定義を参照することができます。また、*20.34.101.207-201.3.9.99*のようなアドレス範囲や、IPアドレスとネットワークのリストを指定することもできます（エントリーはカンマで区切られます）。このようなリストの中にIPv4アドレスとIPv6アドレスを混在させないでください。

**-d ポート <文字列**

TCP/UDP宛先ポートを制限する。*etc/services*で定義されているように、サービス名または単純な番号(0～65535)を使用できます。ポート範囲は、例えば*80:85*のように“*:~+:*”で指定することができます。

**--icmp-type <文字列>**

icmp-type を指定する。proto が *icmp* または *icmpv6/ipv6-icmp* の場合のみ有効。

**--iface <文字列>**

ネットワーク・インターフェース名。VMとコンテナには、ネットワーク・コンフィギュレーション・キー名を使用する必要がある (*netd+*)。ホスト関連のルールでは、任意の文字列を使用できる。

**--ログ <alert | crit | debug | emerg | err | info | nolog | notice | warning>.**

ファイアウォールルールのログレベル。

**-プロト <文字列>**

IPプロトコル。*etc/protocols*で定義されているように、プロトコル名 (*tcp/udp*) または単純な数字を使うことができる。

**--ソース <文字列>**

パケットの送信元アドレスを制限する。これは、単一のIPアドレス、IPセット(+*ipsetname*)、またはIPエイリアスの定義を参照することができます。また、20.34.101.207-201.3.9.99のようなアドレス範囲や、IPアドレスとネットワークのリストを指定することもできます（エントリーはカンマで区切れます）。このようなリストの中にIPv4アドレスとIPv6アドレスを混在させないでください。

**--スポーツ <文字列>**

TCP/UDPソース・ポートを制限する。*etc/services*で定義されているように、サービス名または単純な番号 (0~65535) を使用できます。ポート範囲は、例えば80:85のように" ":"+"で指定することができます。

いくつか例を挙げよう：

## [ルール]

```
IN SSH(ACCEPT) -i net0
IN SSH(ACCEPT) -i net0 # コメント
IN SSH(ACCEPT) -i net0 -source 192.168.2.192 # ←からのSSHだけを許可する。
192.168.2.192
IN SSH(ACCEPT) -i net0 -source 10.0.0.1-10.0.0.10 # IPレンジのSSHを受け入れる IN
SSH(ACCEPT) -i net0 -source 10.0.0.1,10.0.0.2,10.0.0.3 # ←'のSSHを受け入れる
IPリスト
IN SSH(ACCEPT) -i net0 -source +mynetgroup # ipsetのsshを受け入れる ←'
マイネットグループ
IN SSH(ACCEPT) -i net0 -source myserveralias # エイリアスのsshを受け入れる ←'
サーバーエイリアス

| IN SSH(ACCEPT) -i net0 # 無効ルール

IN DROP # すべての受信パッケージをドロップする
OUT ACCEPT # すべての送信パッケージを受け入れ
る
```

## 13.4 セキュリティ・グループ

セキュリティ・グループとは、クラスタ・レベルで定義されるルールの集合体であり、すべての VM のルールで使用することができる。例えば、*http* と *https* のポートを開くルールを持つ "webserver" という名前のグループを定義することができる。

```
# /etc/pve/firewall/cluster.fw

[グループ・ウェブサーバー]
IN ACCEPT -p tcp -dport 80
IN ACCEPT -p tcp -dport 443
```

次に、このグループをVMのファイアウォールに追加する。

```
# /etc/pve/firewall/<VMID>.fw

[ルール]
グループ・ウェブサーバー
```

## 13.5 IPエイリアス

IPエイリアスを使用すると、ネットワークのIPアドレスを名前に関連付けることができます。そして、その名前を参照することができます：

- ・インサイドIPセット定義
- ・ファイアウォールルールの送信元と送信先のプロパティ

### 13.5.1 標準IPエイリアス `local_network`

このエイリアスは自動的に定義されます。割り当てられた値を確認するには、以下のコマンドを使用してください：

```
# pve-firewall
localnet ローカルホスト名
: example
ローカルIPアドレス192.168.2.100 ネットワーク自動検出: 192.168.0.0/20
検出されたlocal_networkを使用する: 192.168.0.0/20
```

ファイアウォールは、このエイリアスを使用してクラスタ通信（corosync、API、SSH）に必要なすべてのものを許可するルールを自動的に設定する。

ユーザは`cluster.fw alias`セクションでこれらの値を上書きできます。パブリックネットワーク上で単一のホ

```
# /etc/pve/firewall/cluster.fw
[ALIASES].
local_network 1.2.3.4 # シングルIPアドレスを使う
```

ストを使用する場合は、明示的にローカルIPアドレス

## 13.6 IPセット

IP セットは、ネットワークとホストのグループを定義するために使用できます。ファイアウォールルールの source と dest プロパティで '+name` を使って参照できます。

以下の例では、管理IPセットからのHTTPトラフィックを許可している。

```
IN HTTP (ACCEPT) -ソース +管理
```

### 13.6.1 標準IPセット管理

このIPセットはホストファイアウォールのみに適用されます(VM ファイアウォールではありません)。これらのIPは通常の管理タスク(Proxmox VE GUI、VNC、SPICE、SSH)を実行できます。

ローカルクラスタネットワークは自動的にこのIPセット(alias cluster\_network)に追加され、ホストクラスタ間の通信が可能になります。(マルチキャスト、ssh、... ...)

```
# /etc/pve/firewall/cluster.fw

[IPSET管理]
192.168.2.10
192.168.2.10/24
```

### 13.6.2 標準IPセットブラックリスト

これらのIPからのトラフィックは、各ホストとVMのファイアウォールによってドロップされる。

```
# /etc/pve/firewall/cluster.fw

[IPSETブラックリスト]
77.240.159.182
213.87.123.0/24
```

### 13.6.3 標準IPセット ipfilter-net\*

これらのフィルターはVMのネットワーク・インターフェースに属し、主にIPスプーフィングを防ぐために使用される。このようなセットがインターフェイスに存在する場合、そのインターフェイスの対応するipfilterセットに一致しないソースIPを持つ発信トラフィックはすべてドロップされる。

IPアドレスが設定されているコンテナでは、これらのセットが存在する場合（またはVMのファイアウォールのオプション・タブにある一般的なIPフィルター・オプションで有効になっている場合）、関連するIPアドレスが暗黙的に含まれる。

仮想マシンとコンテナの両方について、近隣発見プロトコルを動作させるために、標準MAC由来のIPv6リンク

```
/etc/pve/firewall/<VMID>.fw
```

```
[IPSET ipfilter-net0] # net0 192.168.2.10上の指定されたIPのみを許可する。
```

クローカルアドレスも暗黙的に含んでいる。

## 13.7 サービスとコマンド

ファイアウォールは各ノードで2つのサービス・デーモンを実行する：

- pvefw-logger: NFLOG デーモン (ulogd の置き換え)。
- pve-firewall: iptablesルールを更新

pve-firewallというCLIコマンドもあり、ファイアウォール・サービスの開始と停止に使用できる：

```
# pve-firewall start  
# pve-firewall stop
```

ステータスを取得するには

```
# pve-firewall status
```

上記のコマンドは、すべてのファイアウォール・ルールを読み込んでコンパイルするので、ファイアウォール設定にエラーがある場合は警告が表示される。

生成されたiptablesルールを見たい場合は、次のようにする：

```
# iptables-save
```

## 13.8 デフォルトのファイアウォールルール

以下のトラフィックは、デフォルトのファイアウォール設定によってフィルターされる：

### 13.8.1 データセンター着信/発信 DROP/REJECT

ファイアウォールの入力または出力ポリシーがDROPまたはREJECTに設定されている場合でも、クラスタ内のすべてのProxmox VEホストで次のトラフィックが許可されます：

- ループバックインターフェース上のトラフィック
- すでに確立されたコネクション
- IGMPプロトコルを使用したトラフィック
- ウェブインターフェースへのアクセスを許可するため、管理ホストからのTCPトラフィックをポート8006に送る
- 管理ホストからのTCPトラフィックをポート範囲5900～5999に送り、VNCウェブコンソールのトラフィックを許可する
- 管理ホストからのTCP トラフィックは、SPICE プロキシへの接続用にポート 3128 に送られる。
- 管理ホストからのTCP トラフィックをポート22に送り、sshアクセスを許可する。
- クラスタネットワーク内のUDPトラフィックをポート5405-5412でcorosyncする。

- クラスタネットワークのUDPマルチキャストトラフィック
- ICMP トラフィック タイプ 3 (Destination Unreachable)、4 (congestion control)、または 11 (Time Exceeded) 以下のトラフィックはドロップされるが、ロギングを有効にしてもログには

記録されない：

- 無効な接続状態のTCPコネクション
- corosyncに関連しないブロードキャスト、マルチキャスト、エニーキャストのトラフィック、すなわちポート 5405-5412を経由しないトラフィック。

- ポート43へのTCPトラフィック
- ポート135と445へのUDPトラフィック
- UDPトラフィックのポートレンジは137から139。
- 送信元ポート137からポート範囲1024～65535へのUDPトラフィック
- ポート1900へのUDPトラフィック
- ポート135、139、445へのTCPトラフィック
- 送信元ポート53からのUDPトラフィック

残りのトラフィックはそれぞれドロップまたは拒否され、ログにも記録される。これは、NDP、SMURFS、TCPフラグフィルタリングなど、**ファイアウォール → オプション**で有効になっている追加オプションによって異なる場合があります。

の出力を調べてください。

```
# iptables-save
```

システム・コマンドを使用して、システムでアクティブなファイアウォール・チェーンとルールを確認できます。この出力はシステム・レポートにも含まれ、ウェブGUIのノードのサブスクリプション・タブ、またはpverereportコマンドライン・ツールからアクセスできます。

### 13.8.2 VM/CT着信/発信 DROP/REJECT

これは、設定されたコンフィギュレーションに応じて、DHCP、NDP、ルーター・アドバタイズド、MACおよびIPフィルタリングのためのいくつかの例外を除いて、VMへのすべてのトラフィックをドロップまたは拒否します。パケットをドロップ/リジェクトするルールはデータセンターから引き継がれ、ホストの送受信トラフィックの例外は適用されません。

繰り返しになるが、[iptables-save](#)（上記参照）を使って、適用されたすべてのルールとチェーンを検査することができる。

## 13.9 ファイアウォールルールのログ

デフォルトでは、ファイアウォールルールによってフィルタリングされたトラフィックのロギングはすべて無効になっています。ロギングを有効にするには、受信および/または送信トラフィックのログレベルをフ

ファイアウォール → オプションで設定する必要があります。これは、VM/CT ファイアウォールだけでなく、ホストに対しても個別に行うことができます。これにより、Proxmox VE の標準ファイアウォールルールのロギングが有効になり、[ファイアウォール → ログ](#)で出力を確認できます。さらに、標準ルール（[デフォルトのファイアウォールルールを参照](#)）では、一部のドロップまたは拒否されたパケットのみがログに記録されます。

loglevelは、フィルターされたトラフィックがどれだけログに記録されるかに影響しない。これは、フィルタリングと後処理を容易にするために、ログ出力に接頭辞として付加されるLOGIDを変更する。

loglevelは以下のフラグのいずれかである：

ログレベル	ロジッド
ノログ	-
エマージェンシー	0
アラート	1
クリティック	2

ログレベル	ロジッド
誤る	3
警告	4
お知らせ	5
インフォメーション	6
デバッグ	7

典型的なファイアウォールのログ出力は次のようになる：

```
vmid logid chain timestamp policy: PACKET_DETAILS
```

ホストファイアウォールの場合、VMIDは0に等しい。

### 13.9.1 ユーザー定義のファイアウォールルールのログ

ユーザー定義のファイアウォールルールによってフィルタリングされたパケットをログに記録するために、各ルールに対して個別にログレベルパラメータを設定することができます。これにより、定義されたログレベルとは無関係に、きめ細かくログを記録することができます。

**ファイアウォール → オプション.**

個々のルールのログレベルは、ルールの作成または変更時にウェブUIで簡単に定義または変更できますが、対応するpvesh APIコールでも設定できます。

さらに、`-log <loglevel>`を追加することで、ファイアウォール設定ファイルを通してログレベルを設定することもできる。

を選択したルールに適用する（[可能なログレベ](#)

```
IN REJECT -p icmp -log nolog
IN REJECT -p icmp
```

:

一方

```
IN REJECT -p icmp -log debug
```

は、デバッグ・レベルのフラグが付いたログ出力を生成する。

## 13.10 ヒントとコツ

### 13.10.1 FTPを許可する方法

FTPは、ポート21と他のいくつかの動的ポートを使用する古いスタイルのプロトコルである。そのため、ポート21を受け入れるルールが必要である。さらに、`ip_conntrack_ftp`モジュールをロードする必要がある。ですから、実行してください：

```
modprobe ip_conntrack_ftp
```

を追加し、`/etc/modules`に`ip_conntrack_ftp`を追加する（再起動後も動作するようにする）。

### 13.10.2 スリカータIPS統合

Suricata IPS (Intrusion Prevention System)を使用したい場合は可能です。パケッ

トはファイアウォールがACCEPTした後にのみIPSに転送されます。

リジェクト/ドロップされたファイアウォールのパケット

```
# apt-get install  
suricata # modprobe  
nfnetlink_queue
```

次のリブートのために、/etc/modulesにnfnetlink\_queueを追加するの

```
# /etc/pve/firewall/<VMID>.fw  
  
[オプション]  
ips: 1  
ips_queues: 0
```

ipsはこのVMに特定のCPUキューをバインドします。利用

```
# etc/default/suricata  
NFQUEUE=0
```

### 13.11 IPv6に関する注意事項

ファイアウォールにはIPv6特有のオプションがいくつかあります。IPv6はARPプロトコルを使わず、IPレベルで動作するNDP(Neighbor Discovery Protocol)を使うため、IPアドレスが必要です。このため、インターフェイスのMACアドレスに由来するリンクローカルアドレスが使用される。デフォルトでは、NDPオプションはホストレベルとVMレベルの両方で有効になっており、近隣探索(NDP)パケットの送受信が可能になっている。

ネイバーディスカバリーの他にも、NDPは自動コンフィギュレーションやルーターの広告など、いくつかの

ことに使われる。

デフォルトでは、VMはルーター勧誘メッセージの送信（ルーターへの問い合わせ）とルーター広告パケットの受信を許可されている。これにより、ステートレス自動コンフィギュレーションを使用できる。一方、「ルーター広告を許可する」 (radv: 1) オプションが設定されていない限り、VMは自分自身をルーターとして広告することはできない。

NDPに必要なリンク・ローカル・アドレスに関しては、"IP Filter" (ipfilter: 1) オプションもあり、これを有効にすると、対応するリンク・ローカル・アドレスを含むVMの各ネットワーク・インターフェースに ipfilter-net\* ipsetを追加したのと同じ効果が得られる。(詳細については、[標準IPセット ipfilter-net\\*](#) のセクションを参照してください)。

## 13.12 Proxmox VEが使用するポート

- ウェブインターフェース8006 (TCP、HTTP/1.1 over TLS)

- VNCウェブコンソール5900-5999 (TCP、WebSocket)
- SPICE プロキシ: 3128 (TCP)
- sshd (クラスタアクションに使用): 22 (TCP)
- rpcbind: 111 (UDP)
- sendmail: 25 (TCP、送信)
- corosyncクラスタのトラフィック: 5405-5412 UDP
- ライブマイグレーション (VMメモリとローカルディスクのデータ) : 60000-60050 (tcp)

## 13.13 エヌエフティーブルズ

pve-firewallの代替として、私たちはproxmox-firewallを提供しています。これはiptablesではなく、より新しいnftablesをベースにしたProxmox VEファイアウォールの実装です。

 **警告**

proxmox-firewallは現在技術プレビュー中です。バグやオリジナルのファイアウォールとの非互換性があるかもしれません。現在のところ本番環境での使用には適していません。

この実装では、同じ設定ファイルと設定フォーマットを使用するため、切り替え時に以前の設定を使用することができます。いくつかの例外を除いて、まったく同じ機能を提供します：

- REJECTは現在、ゲストトラフィックにはできない（代わりにトラフィックはドロップされる）。
- NDP、ルーター広告、またはDHCPオプションを使用すると、デフォルトのポリシーに関係なく、常にファイアウォールルールが作成されます。
- ゲスト用のファイアウォールルールは、conntrackテーブルエントリを持つ接続に対しても評価される。

### 13.13.1 インストールと使用方法

```
apt install proxmox-firewall
```

proxmox-firewallパッケージをインストールします：

nftables バックエンドを有効にするには、ホストの Web UI (Host > Firewall > Options > nftables) を使用するか、ホストの設定ファイル (`/etc/pve/nodes/<node_name>/host.fw`) で有効にします：

[オプション]

```
nftables: 1
```

---

注

`proxmox-firewall` を有効化/無効化した後、新旧のファイアウォールを正しく動作させるために、実行中のすべての VM とコンテナを再起動する必要がある。

---

nftablesコンフィギュレーションキーを設定すると、新しいproxmox-firewallサービスが引き継がれます。

```
systemctl status proxmox-firewall
```

れます。新しいサービスが動作しているかどうかは、proxmox-firewallのsystemctl statusをチェックすることで確認できます：

生成されたルールセットを調べることもできる。これについては「[役に立つコマンド](#)」のセクションに詳しい情報があります。また、pve-firewall が iptables ルールを生成しなくなったかどうかを確認する必要があります。

古いファイアウォールに戻すには、設定値を0/Noに戻すだけです。

### 13.13.2 使用方法

proxmox-firewallはproxmox-firewallサービスによって管理される2つのテーブル: proxmox-firewallとproxmox-firewall-guestsを作成します。Proxmox VE ファイアウォール設定の外側にあるカスタムルールを作成したい場合は、カスタムファイアウォールルールを管理するために独自のテーブルを作成することができます。

pve-firewallコマンドを使う代わりに、nftablesベースのファイアウォールはproxmox-firewall

```
systemctl start proxmox-firewall  
systemctl stop proxmox-firewall
```

を使う。これはsystemd サービスなので、systemctl で起動と停止ができます：

ファイアウォール・サービスを停止すると、生成されたルールはすべて削除される。

ファイアウォールのステータスを照会するには、systemctl サービスのステータスを照会します：

```
systemctl status proxmox-firewall
```

### 13.13.3 役立つコマンド

生成されたルールセットは、以下のコマンドで確認できる：

```
nftリスト・ルールセット
```

proxmox-firewallをデバッグしたい場合は、単純にRUST\_LOGでデーモンをフォアグラウンドで実行できます。

```
RUST_LOG=trace /usr/libexec/proxmox/proxmox-firewall
```

環境変数を `trace` に設定する。これで詳細なデバッグ出力が得られるはずだ：

ファイアウォールデーモンの詳細な出力を得たい場合は、`systemctl` サービスを編集することもできる：

```
systemctl proxmox-Firewallを編集する
```

次に、`RUST_LOG` 環境変数のオーバーライドを追加する必要があります：

[サービス]

```
Environment="RUST_LOG=trace"
```

これは大量のログを素早く生成するので、デバッグ目的でのみ使用すること。他の、より冗長でないログレベルは `info` と `debug` です。

フォアグラウンドで実行すると、ログ出力が STDERR に書き込まれるので、以下のコマンドでリダイレクトすることができる（コミュニティフォーラムにログを提出する場合など）：

```
RUST_LOG=trace /usr/libexec/proxmox/proxmox-firewall 2> firewall_log_$('  
ホスト名).txt
```

ファイアウォール・ルールをデバッグするために、異なるチェーンを経由するパケット・フローをトレース

```
#!/usr/sbin/nft -f  
table bridge  
tracebridge  
削除テーブル・ブリッジ・トレース  
  
table bridge tracebridge {  
    chain trace {  
        meta l4proto icmp meta nftrace set 1  
    }  
}
```

#### チェーン・プリルート

```
    type filter hook prerouting priority -350; policy accept;  
    jump trace  
}  
}
```

#### チェーン・ポストルーティング

```
    type filter hook postrouting priority -350; policy accept;  
    jump trace  
}  
}
```

することは有用である。これは、追跡したいパケットに `nftrace` を 1 に設定することで実現できる。すべてのパケットにこのフラグを設定しないことをお勧めします。

このファイルを保存し、実行可能にしてから一度実行すると、それぞれのトレースチェーンが作成されます。その後、Proxmox VE Web UI (Firewall > Log) または `nft monitor trace` 経由でトレース出力を検査できます。

上記の例では、すべてのブリッジのトラフィックをトレースしていますが、これは通常、ゲストのトラフィックが流れる場所です。ホストのトラフィックを調べたい場合は、ブリッジテーブルの代わりに `inet` テーブルにこれらのチェーンを作成します。

---

#### 注

これは大量のログスパムを生成し、ネットワークスタックのパフォーマンスを著しく低下させる可能性があることに注意してください。

---

以下のコマンドを実行することで、トレースルールを削除することができる：

## nft テーブル削除 ブリッジ・トレース

# 第14章

## ユーザー管理

Proxmox VEは、Linux PAM、統合Proxmox VE認証サーバ、LDAP、Microsoft Active Directory、OpenID Connectなど、複数の認証ソースをサポートしています。

すべてのオブジェクト（VM、ストレージ、ノードなど）に対してロールベースのユーザーと権限管理を使用することで、きめ細かなアクセスを定義できます。

### 14.1 ユーザー

Proxmox VEはユーザ属性を `/etc/pve/user.cfg` に保存します。パスワードはここには保存されません。代わりに、ユーザは後述する[認証レルム](#)に関連付けられます。そのため、ユーザは内部的には `<userid>@<realm>` という形でユーザ名とrealmで識別されることが多いです。

このファイルの各ユーザー・エントリには、以下の情報が含まれている：

- 名前
- 姓
- Eメールアドレス
- 団体会員
- 任意の有効期限
- このユーザーについてのコメント

- このユーザーが有効か無効か
- オプションの二要素認証キー

---

**注意**

 ユーザーを無効化または削除した場合、または設定された有効期限が過去の場合、このユーザーは新しいセッションにログインしたり、新しいタスクを開始したりできなくなります。このユーザーによってすでに開始されているすべてのタスク（ターミナル・セッションなど）は、このようなイベントによって自動的に終了することはありません。

---

### 14.1.1 システム管理者

システムのルート・ユーザーは、常にLinux PAM レベルでログインでき、制約のない管理者である。このユーザは削除できませんが、属性は変更できます。システムメールは、このユーザーに割り当てられたメールアドレスに送信されます。

## 14.2 グループ

各ユーザーは複数のグループのメンバーになることができる。グループは、アクセス・パーミッションを編成するのに適した方法です。個々のユーザーではなく、常にグループにアクセス許可を与えるべきです。そうすることで、より保守性の高いアクセス制御リストを得ることができます。

## 14.3 API トークン

API トークンを使用すると、別のシステム、ソフトウェア、またはAPIクライアントからREST APIのほとんどの部分にステータスでアクセスできます。トークンは個々のユーザーのために生成することができ、アクセスの範囲と期間を制限するために個別の権限と有効期限を与えることができる。API トークンが漏洩した場合、ユーザー自身を無効にすることなく、トークンを失効させることができます。

API トークンには基本的に2つのタイプがある：

- 権限の分離：トークンにはACLで明示的なアクセス権を与える必要がある。その有効権限は、ユーザー権限とトークン権限を交差させて計算されます。
- 完全な権限：トークンの権限は、関連するユーザーの権限と同じです。



#### 注意

トークンの値は、トークンが生成されたときに一度だけ表示/返されます。後日、APIを介して再度取得することはできません！

API トークンを使用するには、HTTPヘッダーのAuthorizationを、PVEAPIToken=USER@の形式で表示される値に設定します。

APIリクエストを行う際には、API クライアントのドキュメントを参照すること。

## 14.4 リソースプール



リソースプールは、仮想マシン、コンテナ、ストレージデバイスの集合である。リソースごとに管理する必要がなく、単一のパーミッションを一連の要素に適用できるため、特定のユーザーが特定のリソース・セットへのアクセス権を管理する場合に便利です。リソース・プールはしばしばグループと同時に使用され、グループのメンバーが一連のマシンとストレージに対するパーミッションを持つようにします。

## 14.5 認証領域

Proxmox VE ユーザは、外部レルムに存在するユーザと対になるため、`/etc/pve/domains.cfg` でレルムを設定する必要があります。以下のレルム(認証方法)が利用可能です：

### Linux PAM標準認証

Linux PAMは、システム全体のユーザー認証のためのフレームワークである。これらのユーザは、ホストシステム上で `adduser` などのコマンドを使用して作成します。Proxmox VE ホストシステムに PAM ユーザーが存在する場合、対応するエントリを Proxmox VE に追加して、これらのユーザーがシステムのユーザー名とパスワードでログインできるようにすることができます。

### Proxmox VE 認証サーバー

これはUnixライクなパスワードストアで、`/etc/pve/priv/shadow.cfg` にハッシュ化されたパスワードを保存します。パスワードはSHA-256ハッシュアルゴリズムを使ってハッシュされます。これは、ユーザがProxmox VEの外部にアクセスする必要がない、小規模（または中規模）のインストールに最も便利な領域です。この場合、ユーザーは Proxmox VE で完全に管理され、GUI を使用して自分のパスワードを変更できます。

### ライトウェイトディレクトリアクセスプロトコル

LDAP (Lightweight Directory Access Protocol) は、ディレクトリサービスを使った認証のための、オープンでクロスプラットフォームなプロトコルである。OpenLDAPは、LDAPプロトコルの一般的なオープンソース実装です。

### マイクロソフト・アクティブ・ディレクトリ (AD)

Microsoft Active Directory (AD)はWindows ドメインネットワーク用のディレクトリサービスで、Proxmox VEの認証レルムとしてサポートされています。認証プロトコルとしてLDAPをサポートしています。

### OpenID Connect

OpenID Connectは、OATH 2.0プロトコルの上にIDレイヤーとして実装されている。これによりクラウドアントは、外部の認証サーバーによって実行された認証に基づいて、ユーザーの身元を確認することができます。

### 14.5.1 Linux PAM標準認証

Linux PAMはホスト・システム・ユーザーに対応するため、ユーザーがログインを許可される各ノードにシステム・ユーザーが存在しなければならない。ユーザーは通常のシステムパスワードで認証する。このレルムはデフォルトで追加され、削除することはできない。設定可能性という点では、管理者はレルムからのログインで2要素認証を要求したり、レルムをデフォルトの認証レルムとして設定したりすることができる。

### 14.5.2 Proxmox VE 認証サーバー

Proxmox VE認証サーバのレルムは、単純なUnixライクなパスワードストアである。レルムはデフォルトで作成され、Linux PAMと同様に、利用可能な設定項目は、レルムのユーザに2要素認証を要求する機能と、ログイン用のデフォルトのレルムとして設定する機能だけです。

他のProxmox VEレルムタイプとは異なり、ユーザは他のシステムに対して認証するのではなく、Proxmox VEを通じて作成および認証されます。したがって、このタイプのユーザーは作成時にパスワードを設定する必要があります。

### 14.5.3 ライトウェイトディレクトリーアクセスプロトコル

ユーザ認証に外部のLDAPサーバを使用することもできます(例えばOpenLDAP)。このレルム・タイプでは、ユーザはユーザ属性名(user\_attr)フィールドで指定されたユーザ名属性を使用して、ベース・ドメイン名(base\_dn)の下で検索されます。

サーバーとオプションのフォールバック・サーバーを設定し、SSLで接続を暗号化することができる。さらに、ディレクトリとグループに対してフィルタを設定することができる。フィルタによって、レルムの範囲をさらに限定することができる。

例えば、あるユーザーが次のようなLDIFデータセットで表現されているとする:

```
# ldap-test.comのPeopleのuser1
dn: uid=user1,ou=People,dc=ldap-test,dc=com
objectClass: top
オブジェクトクラス: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
uid: user1
cn: テストユーザー1
sn: テスター
```

の説明を参照してください: これは最初のテストユーザーです。

ベース・ドメイン名はou=People,dc=ldap-test,dc=comとなり、ユーザー属性はuidとなる。

Proxmox VE がユーザを照会および認証する前に LDAP サーバに認証(バインド)する必要がある場合は、/etc/pve/domains.cfg のbind\_dn プロパティでバインド・ドメイン名を設定できます。そのパスワードは、/etc/pve/priv/ldap/<realmname>.pw に保存する必要があります。このファイルには、生のパスワードを1行だけ記述する。

証明書を検証するには、capathを設定する必要がある。LDAPサーバーのCA証明書を直接設定するか、すべての信頼されたCA証明書を含むシステム・パス (/etc/ssl/certs) を設定することができる。さらに、verifyオプションを設定する必要があるが、これはウェブ・インターフェース上でも行うことができる。

LDAPサーバー・レルムの主な設定オプションは以下の通りである:

- レルム(realm): Proxmox VEユーザのレルム識別子
- ベース・ドメイン名 (base\_dn) : ユーザーを検索するディレクトリ

- ユーザー属性名 (user\_attr) : ユーザーがログインするユーザー名を含むLDAP属性
- サーバー(server1): LDAPディレクトリをホストするサーバー
- 予備サーバー (server2) : プライマリ・サーバーが到達不能な場合に備えて、オプションでフルバック・サーバーのアドレスを指定します。
- Port (ポート) : LDAPサーバーがリッスンするポート

---

### 注

特定のユーザに LDAP サーバを使用した認証を許可するには、Proxmox VE サーバからそのレルムのユーザとして追加する必要があります。これは[同期によって](#)自動的に実行できます。

---

#### 14.5.4 マイクロソフト・アクティブ・ディレクトリ (AD)

Microsoft ADをレルムとして設定するには、サーバーアドレスと認証ドメインを指定する必要がある。Active Directoryは、オプションのフォールバックサーバ、ポート、SSL暗号化など、LDAPと同じプロパティのほとんどをサポートしています。さらに、ユーザは設定後、同期操作によって自動的にProxmox VEに追加することができます。

LDAPと同様に、Proxmox VEがADサーバーにバインドする前に認証が必要な場合は、以下のように設定する必要があります。

バインド・ユーザー (bind\_dn) プロパティ。このプロパティは通常、Microsoft ADではデフォルトで必要です。Microsoft Active Directoryの主な構成設定は以下のとおりです：

- レルム(realm): Proxmox VEユーザのレルム識別子
- ドメイン (domain) : サーバーのADドメイン
- サーバー (server1) : サーバーのFQDNまたはIPアドレス
- 予備サーバー (server2) : プライマリ・サーバーが到達不能な場合に備えて、オプションでフォールバック・サーバーのアドレスを指定します。
- Port (ポート) : Microsoft ADサーバーがリッスンするポート。

---

#### 注

Microsoft ADは通常、大文字と小文字を区別せずにユーザー名のような値をチェックします。Proxmox VEを同じようにするには、Web UIでrealmを編集するか、CLIを使用してデフォルトの大文字小文字を区別するオプションを無効にします(realm IDでIDを変更します): `pveum realm modify ID --case-sensitive 0`

---

#### 14.5.5 LDAPベースのレルムの同期

Add: LDAP Server

General Sync Options

Realm:	office-dc	Server:	dc1.example.proxmox.com
Base Domain Name:	ou=Users,dc=example,dc=rf	Fallback Server:	dc2.example.proxmox.com
User Attribute Name:	uid	Port:	Default
Default:	<input type="checkbox"/>	SSL:	<input checked="" type="checkbox"/>
Comment:			
<a href="#">Help</a>		<a href="#">Add</a>	

LDAP ベースのレルム (LDAP & Microsoft Active Directory) のユーザーとグループを Proxmox VE に手動で追加するのではなく、自動的に同期することができます。同期オプションは、Web インターフェイスの認証パネルの Add/Edit ウィンドウ、または pveum realm add/modify コマンドからアクセスできます。その後、GUI の Authentication パネルから、または以下のコマンドを使用して同期操作を実行できます：

#### pveum レルム同期<レルム

ユーザとグループはクラスタ全体の構成ファイル /etc/pve/user.cfg に同期されます。

## 属性からプロパティへ

同期レスポンスにユーザー属性が含まれている場合、その属性は

`user.cfg`で指定します。例: `firstname`または`lastname`。

属性の名前がProxmox VEのプロパティと一致しない場合は、`sync_attributes`オプションを使用して、コンフィグでカスタムフィールド間マップを設定できます。

このようなプロパティが消滅した場合にどのように処理されるかは、後述の同期オプションで制御できる。

## 同期設定

LDAPベースのレルムを同期するための設定オプションは、Add/EditウィンドウのSync Optionsタブにあります。

設定オプションは以下の通り：

- バインド・ユーザー (`bind_dn`)：ユーザーとグループの問い合わせに使用する LDAP アカウントを指す。このアカウントはすべてのエントリーにアクセスできる必要があります。これが設定されている場合、検索はバインド経由で行われます。そうでない場合、検索は匿名で行われます。ユーザーは完全な LDAP形式の識別名(DN)でなければなりません。例えば、`cn=admin,dc=example,dc=com`です。
- グループ名attr (`group_name_attr`)：ユーザーのグループを表す。`user.cfg`の通常の文字制限に従ったエントリのみが同期される。グループは、名前の衝突を避けるために、名前に`-$realm`が付加された状態で同期されます。同期によって手動で作成したグループが上書きされないようにしてください。
- ユーザー・クラス (`user_classes`)：ユーザーに関連するオブジェクト・クラス。
- グループ・クラス (`group_classes`)：グループに関連するオブジェクト・クラス。
- Eメール属性：LDAPベースのサーバーでユーザーの電子メールアドレスが指定されている場合、ここで関連する属性を設定することで、これらの電子メールアドレスも同期に含めることができます。コマンドラインから `--sync_attributes` パラメータを指定することで実現できます。
- ユーザー・フィルター (`filter`)：特定のユーザーをターゲットとする、さらなるフィルタオプション。
- グループ・フィルター (`group_filter`)：特定のグループをターゲットとするフィルタオプション。

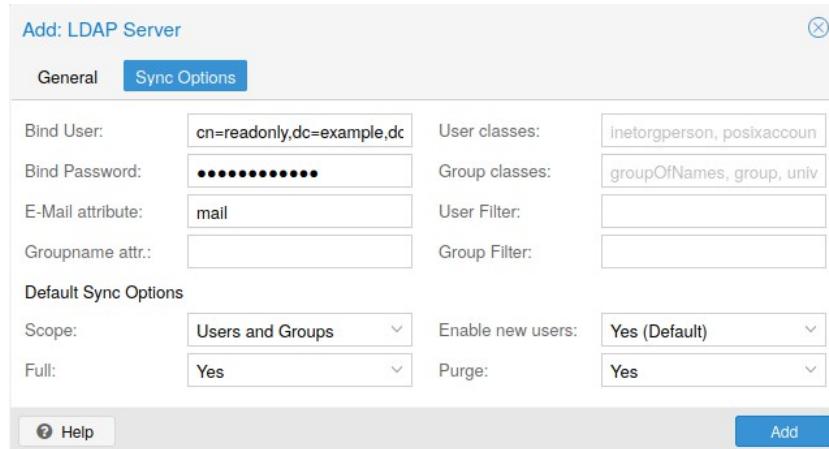
---

### 注

フィルタを使用すると、追加の一致基準を作成して、同期の範囲を絞り込むことができます。利用可能な LDAP フィルタの種類とその使い方については、[ldap.com](#) を参照してください。

---

## 同期オプション



前のセクションで指定したオプションに加えて、シンク操作の動作を定義するさらなるオプションを設定することもできます。

これらのオプションは、同期の前にパラメータとして設定されるか、レルムオプションsync-defaults-によってデフォルトとして設定される。

同期の主なオプションは以下の通り：

- **スコープ (範囲)**：同期するものの範囲。ユーザー、グループ、またはその両方を指定できます。
- **新規を有効にする (enable-new)**：設定すると、新しく同期されたユーザが有効になり、ログインできるようになる。デフォルトは  
本当だ。
- **バニッシュされたものを削除する (remove-vanished)**：これはオプションのリストで、有効にすると、同期レスポンスから返されなかったときに削除されるかどうかを決定します。オプションは次のとおりです：
  - **ACL (acl)**：同期応答で返されなかったユーザーとグループの ACL を削除する。  
これはエントリーとともに意味を持つことが多い。
  - **エントリー (entry)**：エントリ (entry) : 同期応答で返されなかったエントリ（ユーザーやグループなど）を削除します。
  - **プロパティ (properties)**：プロパティ (properties) : シンク応答のユーザーがそれらの属性を含んでいないエントリのプロパティを削除します。これにはすべてのプロパティが含まれ、シンクによって一度も設定されなかったプロパティも含まれます。例外はトークンと有効フラグで、これらはこのオプションを有効にしても保持されます。

- プレビュー（ドライラン）：データはコンフィグに書き込まれない。どのユーザーとグループが `user.cfg` に同期されるかを確認したい場合に便利です。

## 予約文字

ある種の文字は予約されており（[RFC2253](#)参照）、適切にエスケープされなければ、DNの属性値で簡単に使うことはできない。

以下の文字はエスケープが必要です：

- 先頭または末尾にスペース（）を入れる
- 先頭に数字記号 (#)
- カンマ(,)

- プラス記号 (+)
- ダブルクオート (")
- スラッシュ (/)
- 角括弧 (<>)
- セミコロン(;)
- 等号 (=)

このような文字をDNで使用するには、属性値を二重引用符で囲む。例えば、ユーザー CN(コモンネーム) Example, User で、CN="Example, User", OU=people, DC=example を使用する。  
を bind\_dn の値とする。

これは、base\_dn、bind\_dn、group\_dn 属性に適用される。

---

## 注

コロンとスラッシュはユーザー名の予約文字であるため、同期できません。

---

### 14.5.6 OpenID Connect

OpenID Connectの主な設定オプションは以下の通り：

- 発行者URL (issuer-url) : 認証サーバのURLです。ProxmoxはOpenID Connect Discoveryプロトコルを使用して、さらに詳細を自動的に設定します。  
暗号化されていないhttp:// URLを使用することも可能ですが、暗号化されたhttps:// コネクション。
- レルム(realm): Proxmox VEユーザのレルム識別子
- クライアントID (client-id) : OpenIDクライアントID。
- クライアント・キー (client-key): オプションのOpenIDクライアント・キー。
- ユーザーの自動作成 (autocreate) : ユーザが存在しない場合、自動的にユーザを作成します。認証はOpenIDサーバで行われますが、Proxmox VEのユーザ設定にはすべてのユーザのエントリが必要です

- 手動でユーザを追加するか、`autocreate` オプションを使用して新しいユーザを自動的に追加します。
- ユーザー名クレーム (`username-claim`)：一意なユーザー名を生成するために使用されるOpenIDクレーム(`subject`、  
ユーザー名または電子メール)。

## ユーザー名のマッピング

OpenID Connectの仕様では、`subject`という一意な属性（OpenID用語では`claim`）が定義されています。デフォルトでは、この属性の値を使ってProxmox VEのユーザー名を生成します。

残念ながら、ほとんどのOpenIDサーバーはDGH76OKH34BNG3245SBのようなランダムな文字列を件名に使っているため、典型的なユーザー名はDGH76OKH34BNG3245SB@yourrealmのようになります。ユニークではありますが、人間がこのようなランダムな文字列を覚えておくことは難しく、実際のユーザーをこの文字列と関連付けることはかなり不可能です。

`username-claim`設定では、ユーザー名のマッピングに他の属性を使用することができます。これを`username`は、OpenID Connect サーバがその属性を提供し、一意性が保証されている場合に使用されます。もう1つのオプションは`email`を使用することで、こちらも人間が読めるユーザー名が得られます。この場合も、サーバーがこの属性の一意性を保証している場合にのみ、この設定を使用してください。

### 例

以下は、Googleを使ってOpenIDレルムを作成する例です。`client-id`と`--client-key`には、Google OpenIDの設定の値を指定します。

```
pveum realm add myrealm1 --type openid --issuer-url https://accounts.google.com --client-id XXXX --client-key YYYY --username-claim email
```

上記のコマンドでは`--username-claim email`を使用しているため、Proxmox VE側のユーザー名は`example.user@google.com@myrealm1`のようになります。

Keycloak (<https://www.keycloak.org/>) は、OpenID ConnectをサポートするオープンソースのIDおよびアクセス管

```
pveum realm add myrealm2 --type openid --issuer-url https://your.server:8080/realm/your-realm --client-id XXX --username-claim ユーザー名
```

理ツールとして人気がある。以下の例では、`--issuer-url`と`--client-id`をあなたの情報で置き換える必要がある：

`--username-claim username`を使用すると、Proxmox VE側で`example.u`のようなシンプルなユーザー名を使用できます。



### 警告

ユーザーが（Keycloakサーバー上で）ユーザー名の設定自分で編集できないようにする必要がありま

す。

---

## 14.6 二要素認証

二要素認証を使うには2つの方法がある：

これは、*TOTP*（Time-based One-Time Password）または*YubiKey OTP*のいずれかの認証レルムによって要求される。この場合、新しく作成されたユーザは、すぐに鍵を追加する必要があります。*TOTP*の場合、最初にログインできれば、後から*TOTP*を変更することも可能です。

あるいは、レルムが2ファクタ認証を強制していなくても、ユーザは後で2ファクタ認証にオプトインすることを選択できる。

### 14.6.1 利用可能なセカンドファクター

スマートフォンやセキュリティ・キーを紛失してアカウントから永久にロックされてしまう事態を避けるため、複数のセカンド・ファクターを設定することができる。

レルムで強制される TOTP および Yu- biKey OTP に加えて、以下の 2 要素認証方法が利用可能です：

- ユーザーが設定するTOTP（[時間ベースのワンタイムパスワード](#)）。共有シークレットと現在時刻から導き出される短いコードで、30秒ごとに変更される。
- WebAuthn ([Web Authentication](#))。認証のための一般的な規格。コンピュータやスマートフォンのハードウェア・キーやTPM（Trusted Platform Module）など、さまざまなセキュリティ・デバイスによって実装される。
- シングルユースのリカバリーキー。プリントアウトして安全な場所に施錠するか、電子保管庫にデジタル保存する必要があるキーのリスト。各キーは一度しか使用できません。他のすべてのセカンドファクターが紛失または破損した場合でも、ロックアウトされないようにするのに最適なキーです。

WebAuthnがサポートされる前は、U2Fはユーザーが設定することができました。既存のU2Fファクターはまだ使用できますが、サーバー上で設定されたら、WebAuthnに切り替えることをお勧めします。

### 14.6.2 レルム強制二要素認証

これは、認証レルムを追加または編集する際に、TFAドロップダウン・ボックスから利用可能な方法の1つを選択することで行うことができます。レルムがTFAを有効にすると、それが必須条件となり、TFAを設定したユーザのみがログインできるようになります。

現在、利用可能な方法は2つある：

#### タイムベースOATH (TOTP)

これは標準的なHMAC-SHA1アルゴリズムを使用し、現在時刻はユーザーが設定した鍵でハッシュ化される。時間ステップとパスワードの長さのパラメータは設定可能である。

ユーザーは複数のキーを設定でき（スペースで区切る）、キーはBase32（RFC3548）または16進数表記で指定できる。

Proxmox VEは、oathtoolコマンドラインツールやAndroid Google Authenticator、FreeOTP、andOTPなどの様々なOTPツールで直接使用できるBase32記法のランダムキーを出力するキー生成ツール(oathkeygen)を提供しています。

### **YubiKey OTP**

YubiKeyを使った認証には、Yubico API ID、API KEY、認証サーバーのURLが必要です。YubiKeyからキーIDを取得するには、YubiKeyをUSBで接続した後、一度YubiKeyを起動し、入力されたパスワードの最初の12文字をユーザーのキーIDsフィールドにコピーします。

[YubiCloud](#)を使用する方法、または[独自の認証サーバをホスト](#)する方法については、[YubiKey OTP のドキュメント](#)を参照してください。

### 14.6.3 二要素認証の制限とロックアウト

第二の要因は、パスワードが何らかの形で漏れたり推測されたりした場合に、ユーザーを保護するためのものである。しかし、ブルートフォースによって破られる可能性もあります。このため、第2要素によるログインに何度も失敗すると、ユーザーはロックアウトされます。

TOTPの場合、8回失敗すると、ユーザーのTOTPファクターは無効になります。回復キーでログインすると解除されます。TOTPが唯一の利用可能な要素であった場合、管理者の介入が必要であり、ユーザに直ちにパスワードの変更を要求することが強く推奨される。

FIDO2/Webauthnとリカバリ・キーはブルート・フォース・アタックの影響を受けにくいので、そちらの制限は高くなる（100回）が、それを超えるとすべてのセカンド・ファクターが1時間ブロックされる。

管理者は、UIのユーザーリストまたはコマンドラインから、いつでもユーザーの二要素認証を解除すること

```
pveum user tfa ロック解除 joe@pve
```

ができます：

### 14.6.4 ユーザーが設定したTOTP認証

ユーザは、ログイン時に *TOTP* または *WebAuthn* を第 2 要素として有効にするかどうかを、ユーザー一覧の *TFA* ボタンから選択できます（レルムが *YubiKey OTP* を強制している場合を除く）。

ユーザーはいつでも1回限りのリカバリーキーを追加して使用することができます。

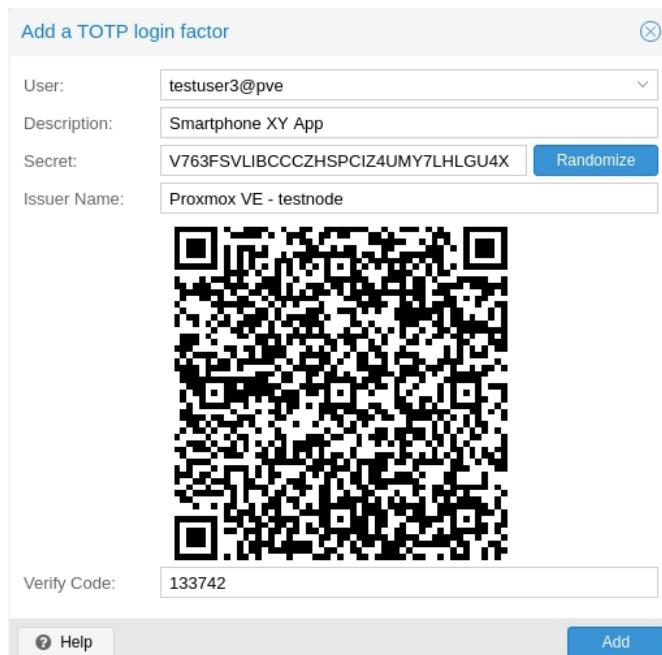
The screenshot shows the Proxmox VE management interface. On the left, there is a sidebar with various navigation options: Datacenter, Search, Summary, Notes, Cluster, Ceph, Options, Storage, Backup, Replication, Permissions (with sub-options: Users, API Tokens, Two Factor), Groups, Pools, Roles, Realms, HA, ACME, Firewall, Metric Server, and Support. The 'Two Factor' option is currently selected and highlighted with a blue background. At the top right, there is a 'Help' button. In the main content area, there is a table with columns: User, Enabled, TFA Type, Created, and Description. A single row is visible: testuser3@pve, Yes, recovery, 2021-11-15 12:20:38.

TFA ウィンドウを開くと、TOTP 認証を設定するためのダイアログが表示される。シークレットフィールドには鍵が含まれます。鍵はRandomizeボタンでランダムに生成することができます。オプションの発行者名

を追加することで、TOTP アプリに、その鍵が何のものであるかの情報を提供することができる。ほとんどの TOTP アプリは、対応する OTP 値とともに発行者名を表示する。ユーザ名は TOTP アプリの QR コードにも含まれる。

キーを生成すると、FreeOTP などのほとんどの OTP アプリで使用できる QR コードが表示されます。その後、ユーザーは現在のユーザー・パスワード（root としてログインしている場合を除く）と、TOTP キーを正しく使用できることを、現在の OTP 値を検証コード・フィールドに入力し、適用ボタンを押すことで確認する必要があります。

## 14.6.5 TOTP



サーバーの設定は不要です。スマートフォンにTOTPアプリ（例えばFreeOTP）をインストールし、Proxmox Backup Serverのウェブインターフェースを使用してTOTPファクターを追加するだけです。

## 14.6.6 ウェブオート

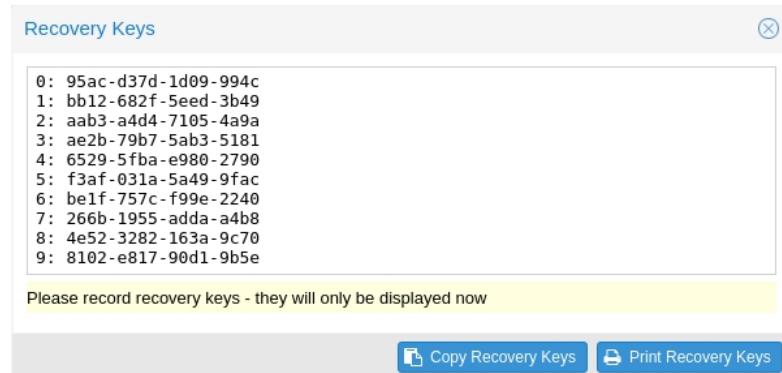
WebAuthnが機能するためには、2つのものが必要です：

- 信頼できる HTTPS 証明書 (Let's Encrypt など)。信頼されていない証明書でもおそらく動作しますが、信頼されていない場合、ブラウザによっては警告が表示されたり、WebAuthn の操作が拒否されたりすることがあります。
- WebAuthn 設定を設定します (Proxmox VE ウェブインターフェースの Datacenter → Options →

**WebAuthn Settings を参照)。** これはほとんどのセットアップで自動入力できます。

これらの両方の要件を満たすと、WebAuthn 設定を **Two Factor** パネルで 「**Datacenter → Permissions → Two Factor**」 を選択する。

## 14.6.7 リカバリーキー

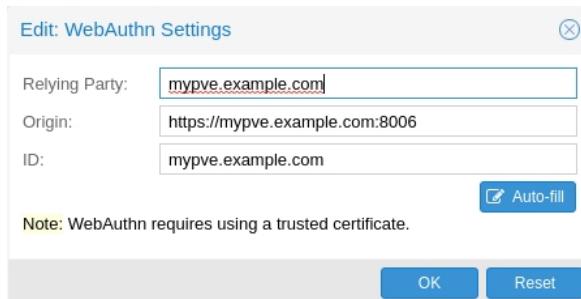


データセンター → 権限 → Two FactorのTwo Factorパネルで回復キーのセットを作成するだけです。

### 注

1ユーザーにつき1セットしか使用できない。

## 14.6.8 サーバー側のWebauthn設定



ユーザーがWebAuthn認証を使用できるようにするには、有効なSSL証明書と有効なドメインを使用する必要があります。

### 注

WebAuthn の設定を変更すると、既存のすべての WebAuthn 登録が使用できなくなる可能性があります！

これは/etc/pve/datacenter.cfgで行います。例えば

```
webauthn: rp=mvpve.example.com,origin=https://mvpve.example.com:8006,id='-'  
mvpve.example.com
```

## 14.6.9 サーバー側のU2F設定

---

### 注

代わりにWebAuthnを使用することをお勧めします。

---

ユーザーがU2F認証を使用できるようにするには、有効なSSL証明書と有効なドメインを使用する必要がある場合があります。そうでない場合、ブラウザによっては警告が表示されたり、U2Fの使用を完全に拒否したりする場合があります。最初に、*AppId*<sup>1</sup> を設定する必要がある。

---

### 注

*AppId*を変更すると、既存のすべてのU2F登録が使用できなくなる！

---

これは/etc/pve/datacenter.cfgで行います。例えば

```
u2f: appid=https://mypve.example.com:8006
```

1つのノードの場合、*AppId*は単純にウェブインターフェースのアドレスとことができ、ブラウザで使用されているものと全く同じである。ブラウザによっては、*AppId*をマッチングする際に他のブラウザよりも厳密な場合があることに注意してください。

複数のノードを使用する場合は、appid.json ファイルを提供する別の https サーバーを用意するのが最善です。<sup>2</sup>ファイルを提供する別のhttpsサーバーを用意するのが最善である。すべてのノードが同じトップレベルドメインのサブドメインを使用している場合、*AppId*としてTLDを使用すれば十分かもしれない。しかし、ブラウザによってはこれを受け入れない場合もあることに注意すべきである。

---

### 注

不正な*AppId*を使用すると通常はエラーが発生するが、特にChromiumでサブドメイン経由でアクセスされるノードにトップレベルドメインの*AppId*を使用した場合など、エラーが発生しないことがある。このため、後で*AppId*を変更すると既存のU2F登録が使用できなくなるため、複数のブラウザで設定をテストすることを推奨する。

---

## 14.6.10 ユーザーとしてU2Fをアクティベートする

U2F認証を有効にするには、TFAウィンドウのU2Fタブを開き、現在のパスワードを入力し（rootでログインしていない場合）、Registerボタンを押す。サーバーが正しくセットアップされ、ブラウザがサーバーから提供された*AppId*を受け入れると、U2Fデバイスのボタンを押すよう促すメッセージが表示されます（YubiKeyの場合、ボタンのランプが1秒間に2回程度、点灯と消灯を繰り返すはずです）。

Firefoxユーザーは、U2Fトークンを使用する前に、about:config経由でsecurity.webauth.u2fを有効にする必要があるかもしれません。

---

## 14.7 許可管理

ユーザーがアクション（VMのコンフィギュレーションの一部を一覧表示、変更、削除など）を実行するには、適切なパーミッションが必要です。

Proxmox VEは、ロールおよびパスベースの権限管理システムを使用しています。パーミッションテーブルのエントリは、オブジェクトやパスにアクセスする際に、ユーザー、グループ、トークンが特定の役割を担うことを許可します。このようなアクセスルールは、(パス、ユーザー、ロール)、(パス、グループ、ロール)、(パス、トークン、ロール)のトリプルとして表現され、ロールは許可されたアクションのセットを含み、パスはこれらのアクションのターゲットを表します。

<sup>1</sup>AppId [https://developers.yubico.com/U2F/App\\_ID.html](https://developers.yubico.com/U2F/App_ID.html)

<sup>2</sup>多面的なアプリ： [https://developers.yubico.com/U2F/App\\_ID.html](https://developers.yubico.com/U2F/App_ID.html)

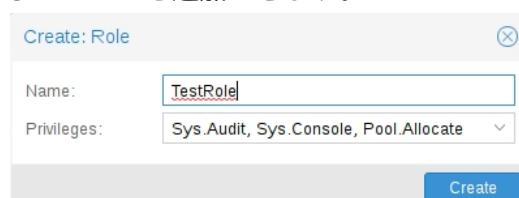
### 14.7.1 役割

ロールは単に権限のリストです。Proxmox VEには定義済みのロールが多数用意されており、ほとんどの要件を満たすことができます。

- **管理者:** 全権限を持つ
- **NoAccess:** 権限を持たない（アクセスを禁止するために使用される）
- **PVEAdmin:** ほとんどのタスクを実行できるが、システム設定（`Sys.PowerMgmt`、`Sys.Modify Realm.Allocate`）やパーミッション（`Permissions.Modify`）を変更する権限はない。
- **PVEAuditor:** 読み取り専用
- **PVEDatastoreAdmin:** バックアップ領域とテンプレートの作成と割り当て
- **PVEDatastoreUser:** バックアップ領域の割り当てとストレージの表示
- **PVEMappingAdmin:** リソースマッピングの管理
- **PVEMappingUser:** リソースマッピングの表示と使用
- **PVEPoolAdmin:** プールを割り当てる
- **PVEPoolUser:** プールを見る
- **PVESDNAdmin:** SDN 設定を管理する
- **PVESDNUser:** ブリッジ/ネットへのアクセス
- **PVESysAdmin:** 監査、システムコンソール、システムログ
- **PVETemplateUser:** テンプレートの表示とクローン
- **PVEUserAdmin:** ユーザーの管理
- **PVEVMAdmin:** VMを完全に管理する
- **PVEVMUser:** 表示、バックアップ、CD-ROMの設定、VMコンソール、VM電源管理

GUIでは、定義済みのロール一式を見ることができる。

GUIまたはコマンドラインから新しいロールを追加できます。



GUIから*Datacenter*の*Permissions* → *Roles*タブに移動し、*Create*ボタンをクリックします。そこでロール名を設定し、*Privileges*ドロップダウンメニューから必要な権限を選択できます。

```
pveum role add VM_Power-only --privs "VM.PowerMgmt VM.Console"  
pveum role add Sys_Power-only --privs "Sys.PowerMgmt Sys.Console"
```

---

コマンドラインからロールを追加するには、例えばツールを使うことができる：

### 注

PVEで始まるロールは常にビルトインであり、カスタムロールはこの予約接頭辞を使用できません。

---

## 14.7.2 特典

権限とは、特定のアクションを実行する権利のことです。管理を簡単にするために、権限のリストはロールにグループ化され、権限テーブルで使用することができます。特権はロールの一部でなければ、ユーザやパスに直接割り当てることができないことに注意してください。

現在、以下の特権をサポートしています：

### ノード / システム関連権限

- Group.Allocate: グループの作成/変更/削除
- Mapping.Audit: リソースマッピングの表示
- Mapping.Modify: リソースマッピングの管理
- Mapping.Use: リソースマッピングを使用する
- Permissions.Modify: アクセス許可の変更
- Pool.Allocate: プールの作成/変更/削除
- Pool.Audit: プールを見る
- Realm.AllocateUser: ユーザーをレルムに割り当てる
- Realm.Allocate: 認証レルムの作成/変更/削除
- SDN.Allocate: SDN コンフィギュレーションを管理する
- SDN.Audit: SDN コンフィギュレーションを見る
- Sys.Audit: ノードステータス/設定、Corosyncクラスタ設定、HA設定の表示
- Sys.Console: ノードへのコンソールアクセス
- Sys.Incoming: 他のクラスタからの受信データストリームを許可（実験的）
- Sys.Modify: ノード・ネットワーク・パラメータの作成/変更/削除
- Sys.PowerMgmt: ノードの電源管理（スタート、ストップ、リセット、シャットダウン、...）
- Sys.Syslog: syslogを見る
- User.Modify: ユーザーアクセスと詳細の作成/変更/削除。

### 仮想マシン関連権限

- SDN.Use: SDN のネットとローカルネットワークのブリッジにアクセスする。
- VM.Allocate: サーバー上にVMを作成/削除する

- VM.Audit: VMの設定を見る
- VM.Backup: VMのバックアップ/リストア
- VM.Clone: VMのクローン/コピー
- VM.Config.CDROM: CD-ROMの取り出し/交換
- VM.Config.CPU: CPU設定の変更
- VM.Config.Cloudinit: クラウドイットパラメータを変更する
- VM.Config.Disk: ディスクの追加/変更/削除
- VM.Config.HWType: エミュレートされたハードウェアタイプを変更する

- VM.Config.Memory: メモリ設定の変更
- VM.Config.Network: ネットワークデバイスの追加/変更/削除
- VM.Config.Options: その他のVM設定を変更する
- VM.Console: VMへのコンソールアクセス
- VM.Migrate: VMをクラスタ上の代替サーバーに移行する
- VM.Monitor: VMモニター (kvm) へのアクセス
- VM.PowerMgmt: 電源管理 (スタート、ストップ、リセット、シャットダウン、...)
- VM.Snapshot.Rollback: VMをスナップショットの1つにロールバックする
- VM.Snapshot: VMスナップショットの作成/削除

### ストレージ関連の権限

- Datastore.Allocate: データストアの作成/変更/削除とボリュームの削除
- Datastore.AllocateSpace: データストアに領域を割り当てる
- Datastore.AllocateTemplate: テンプレートとISOイメージの割り当て/アップロード
- Datastore.Audit: データストアの表示/閲覧



#### 警告

Permissions.ModifyとSys.Modifyはどちらも、危険であったり機密であったりするシステムおよびそのコンフィギュレーションの側面の変更を許可するものであるため、取り扱いには注意が必要です。



#### 警告

以下の継承に関するセクションを注意深く読み、割り当てられたロール（およびその権限）がどのようにACLツリーに沿って伝搬されるかを理解してください。

### 14.7.3 オブジェクトとパス

アクセス許可は、仮想マシン、ストレージ、リソースプールなどのオブジェクトに割り当てられます。これらのオブジェクトのアドレスには、ファイルシステムのようなパスを使用します。これらのパスは自然なツリーを形成し、より高い レベル（より短いパス）のパーミッションは、オプションでこの階層内に伝搬することができます。

パスはテンプレート化できる。API呼び出しがテンプレート化されたパスのパーミッションを必要とする場合、パスには API 呼び出しのパラメータへの参照が含まれることがあります。これらの参照は中かっこで指定される。いくつかのパラメータは、API コールの URI から暗黙的に取得される。例えば、`/nodes/mynode/status` を呼び出す際のパーミッションパス `/nodes/{node}` は `/nodes/mynode` のパーミッションを要求し、`/access/acl` への PUT リクエストのパス `{path}` はメソッドのパスパラメータを参照します。

いくつか例を挙げよう：

- `/ノード/{ノード}`: Proxmox VE サーバマシンへのアクセス
- `/vms`: すべてのVMをカバー
- `/vms/{vmid}`: 特定のVMへのアクセス

- /storage/{storeid}: 特定のストレージへのアクセス
- /pool/{プール名}: 特定のプールに含まれるリソースへのアクセス
- /アクセス/グループグループ管理
- /access/realms/{realmid}: レルムへの管理者アクセス

### 継承

前述したように、オブジェクト・パスはファイル・システムのようなツリーを形成し、パーミッションはそのツリーの下のオブジェクトに継承される（デフォルトではpropagateフラグが設定されている）。以下の継承ルールを使用します：

- 個人ユーザーに対するパーミッションは、常にグループパーミッションに取って代わります。
- グループのパーミッションは、ユーザーがそのグループのメンバーである場合に適用されます。
- より深いレベルのパーミッションは、上位レベルから継承されたものを置き換える。
- NoAccessは、指定されたパス上の他のすべてのロールをキャンセルする。

さらに、特権分離トークンは、関連するユーザーが持っていないパーミッションを、任意のパス上で持つことはできない。

## 14.7.4 プール

プールは、仮想マシンとデータストアのセットをグループ化するために使用できます。そして、プール（/pool/{poolid}）にパーミッションを設定するだけで、そのパーミッションはすべてのプール・メンバーに継承されます。これは、アクセス制御を簡素化する素晴らしい方法です。

## 14.7.5 どのパーミッションが必要ですか？

必要なAPIパーミッションは個々のメソッドごとに文書化されており、<https://pve.proxmox. pve-docs/api-viewer/>にあります。

パーミッションはリストとして指定され、ロジックとアクセスチェック関数のツリーとして解釈できる：

[ "and"、<subtests>...] と [ "or"、<subtests>...]。

現在のリストの各要素(and)またはそれ以上の要素(any(or))は真でなければならぬ。

[ "perm"，<path>，[ <privileges>... ]，<options>... ]。

パスはテンプレート化されたパラメータである（「オブジェクトとパス」参照）。列挙された特権のすべて（または any オプションが用いられている場合はいずれか）が、指定されたパス上で許されている必要があります。require-param オプションが指定されているときは、その API 呼び出しのスキーマがそれ以外ではオプショナルとしてリストしているときでも、その指定されたパラメタは必須です。

[userid-group"，[ <privileges>... ]，<options>... ]。

呼び出し元は、/access/groupsにリストされた特権のいずれかを持っていなければならない。さらに、groups\_paramオプションが設定されているかどうかによって、2つのチェックが可能である：

- `groups_param` が設定されている: APIコールに非オプションのgroupsパラメータがあり、呼び出し元がリストされたすべてのグループに対してリストされた特権のいずれかを持っている必要があります。
- `groups_param`が設定されていない: `groups_param`が設定されていない場合: `userid`パラメータで渡されたユーザは、呼び出し元がリストされた特権のいずれかを持つグループに存在し、そのグループの一部でなければならぬ(/access/groups/<group>パスを介して)。

#### [`userid-param`" , "self"]。

APIコールの`userid`パラメータに提供される値は、アクションを実行するユーザーを参照しなければならない（通常、昇格権限を持っていなくても、ユーザーが自分自身に対してアクションを実行できるようにするために、`or`と組み合わせて使用される）。

#### [`userid-param`" , "Realm.AllocateUser"]。

ユーザは /access/realm/<realm> への `Realm.AllocateUser` アクセスが必要で、<realm>は`userid`パラメータで渡されたユーザのrealmを指します。ユーザ ID は <ユーザー名>@<レルム>.

#### [`perm-modify`"、<path>"]。

パスはテンプレート化されたパラメータである ([オブジェクトとパスを参照](#))。ユーザは `Permissions.Modify`

特権か、パスによっては以下の特権で代用できる:

- /storage/...: 'Datastore.Allocate` が必要です。
- /vms/...: 'VM.Allocate` が必要です。
- /pool/...: 'Pool.Allocate` が必要です。

パスが空の場合は、/accessの`Permissions.Modify`が必要です。

ユーザーが`Permissions.Modify`権限を持っていない場合、指定されたパス上の自分の権限のサブセットのみを委譲することができます（例えば、PVEVMAAdminを持つユーザーは PVEVMUserを委譲することはできますが、PVEAdminを委譲することはできません）。

## 14.8 コマンドラインツール

ほとんどのユーザーは GUI を使ってユーザーを管理します。しかし、`pveum` ("Proxmox VE User Manager" の略) と呼ばれるフル機能のコマンドラインツールもあります。Proxmox VEのコマンドラインツールはす

べてAPIのラッパーなので、REST APIからこれらの機能にアクセスすることもできます。

以下は簡単な使用例である。ヘルプを表示するには

```
はげ
```

または（特定のコマンドに関する詳細なヘルプを表示するには）

```
pveumヘルプユーザー追加
```

新規ユーザーを作成する：

```
pveum user add testuser@pve -comment "ただのテスト"
```

パスワードを設定または変更する（すべてのレームが対応しているわけではありません）：

```
pveum passwd testuser@pve
```

ユーザーを無効にします:

```
pveum user modify testuser@pve --enable 0
```

新しいグループを作る:

```
グループ追加テストグループ
```

新しいロールを作成する:

```
pveum role add PVE_Power-only -privs "VM.PowerMgmt VM.Console"
```

## 14.9 実例

### 14.9.1 管理者グループ

管理者が、(rootアカウントを使わずに) 完全な管理者権限を持つユーザーグループを作りたいと思うことはあり得る。

そのためには、まずグループを定義する:

```
pveum group add admin -comment "システム管理者"
```

それから役割を割り当てる:

```
pveum acl modify / -group admin -role 管理者
```

最後に、新しい管理者グループにユーザーを追加することができます:

```
pveum user modify testuser@pve -group admin
```

### 14.9.2 監査役

PVEAuditorロールをユーザーまたはグループに割り当てることで、ユーザーに読み取り専用アクセ

```
-----  
pveum acl modify / -user joe@pve -role PVEAuditor
```

例2: ユーザjoe@pveにすべての仮想マシンを表示することを許可する

```
pveum acl modify /vms -user joe@pve -role PVEAuditor
```

### 14.9.3 ユーザー管理の委任

```
pveum acl modify /access -user joe@pve -role PVEUserAdmin
```

ユーザー管理をユーザーjoe@pveに委譲したい場合は、次のようにする：

ユーザーjoe@pveは、ユーザの追加と削除、およびパスワードなどの他のユーザ属性を変更できるようにな

```
pveum acl modify /access/realm/pve -user joe@pve -role PVEUserAdmin  
pveum acl modify /access/groups/customers -user joe@pve -role PVEUserAdmin
```

りました。これは非常に強力なロールであり、選択されたレルムとグループに制限したい場合がほとんどで  
しょう。以下の例では、joe@pveはレルムpve内のユーザを変更することができます：

---

#### 注

そのユーザーは他のユーザーを追加することができますが、そのユーザーがグループcustomersのメン  
バーであり、レルムpve内にいる場合に限られます。

---

### 14.9.4 モニタリング用限定APIトークン

APIトークンの権限は、常に対応するユーザーの権限のサブセットである。つまり、APIトークンを使用して  
、バックユーザーが権限を持たないタスクを実行することはできない。このセクションでは、トークン所有  
者の権限をさらに制限するために、別の権限を持つAPIトークンを使用する方法を示します。

```
pveum acl modify /vms -user joe@pve -role PVEVMAadmin
```

すべてのVMで、ユーザーjoe@pveにロールPVEVMAadminを与えます：

新しいAPIトークンを別の権限で追加し、VM情報の閲覧のみを許可する（監視目的など）：

```
pveum user token add joe@pve モニタリング -privsep 1  
pveum acl modify /vms -token 'joe@pve!monitoring' -role PVEAuditor
```

ユーザーとトークンの権限を確認する：

```
pveum ユーザー権限 joe@pve
```

```
pveum ユーザー・トークン権限 joe@pve モニタリング
```

### 14.9.5 リソースプール

通常、企業はいくつかの小さな部門に分かれており、それぞれの部門にリソースを割り当てたり、管理

```
pveum group add developers -コメント "Our software developers"
```

タスクを委任したりするのが一般的です。ここでは、ソフトウェア開発部門のためのプールを設定したいとします。まず、グループを作成します：

次に、そのグループのメンバーである新しいユーザーを作成する：

```
pveum user add developer1@pve -group developers -password
```

---

## 注

password "パラメーターはパスワードの入力を要求します。

---

そして、開発部門が使用するリソース・プールを作成する：

```
pveum pool add dev-pool --comment "IT開発プール"
```

最後に、そのプールに権限を割り当てることができる：

```
pveum acl modify /pool/dev-pool/ -group developers -role PVEAdmin
```

ソフトウェア開発者は、そのプールに割り当てられたリソースを管理できるようになった。

## 第15章

# 高可用性

私たちの現代社会は、ネットワークを通じてコンピューターから提供される情報に大きく依存している。モバイル機器は、人々がいつでもどこからでもネットワークにアクセスできるため、その依存度を高めている。このようなサービスを提供する場合、常に利用可能であることが非常に重要である。

可用性を数学的に定義すると、(A)、ある間隔の間にサービスが使用可能な合計時間と、(B)、間隔の長さの比率となる。これは通常、ある年の稼働時間のパーセンテージで表されます。

表15.1：可用性 - 年間ダウントIME

空室率	年間ダウントIME
99	3.65日
99.9	8.76時間
99.99	52分56秒
99.999	5.26分
99.9999	31.5秒
99.99999	3.15秒

可用性を高める方法はいくつかある。最もエレガントな解決策は、ソフトウェアを書き換えて、複数のホストで同時に実行できるようにすることだ。ソフトウェア自体がエラーを検出し、フェイルオーバーを行う方法が必要です。読み取り専用のウェブページを提供するだけなら、これは比較的簡単です。しかし、これは一般的に複雑で、ソフトウェアを自分で修正することができないため、不可能な場合もあります。以下の解決策は、ソフトウェアを修正することなく動作します：

- 信頼性の高い「サーバー」コンポーネントを使用する

---

## 注

同じ機能を持つコンピュータ・コンポーネントでも、コンポーネントの品質によって信頼性の数値は異なる。ほとんどのベンダーは、より信頼性の高いコンポーネントを「サーバー」コンポーネントとして販売しており、通常はより高価格で販売している。

---

- 単一障害点の排除（冗長コンポーネント）

- 無停電電源装置 (UPS) を使用する
  - サーバーに冗長電源を使用する
  - ECC-RAMを使う
  - 冗長ネットワークハードウェアを使用する
  - ローカル・ストレージにRAIDを使用
  - VMデータに分散冗長ストレージを使用する
- ダウンタイムの削減
    - 迅速にアクセス可能な管理者 (24時間365日)
    - スペアパーツの入手可能性 (Proxmox VEクラスタの他のノード)
    - 自動エラー検出 (`ha-manager`によって提供される)
    - 自動フェイルオーバー (`ha-manager`によって提供される)

Proxmox VEのような仮想化環境では、「ハードウェア」への依存がなくなるため、高可用性の実現が非常に容易になります。また、冗長ストレージやネットワークデバイスのセットアップと使用もサポートされているため、1台のホストに障害が発生しても、クラスタ内の別のホストでそれらのサービスを開始するだけで済みます。

さらに良いことに、Proxmox VEは`ha-manager`と呼ばれるソフトウェアスタックを提供しています。これは自動的にエラーを検出し、自動フェイルオーバーを行うことができます。

Proxmox VE `ha-manager`は、"自動化された"管理者のように動作します。まず、どのリソース (VM、コンテナ、...) を管理するかを設定します。次に、`ha-manager`は正しい機能を監視し、エラーが発生した場合に別のノードへのサービスフェイルオーバーを処理します。

しかし、高い可用性には代償が伴う。高品質の部品は高価であり、それを冗長化することで少なくともコストは倍増する。スペアパーツを追加すれば、コストはさらに増加する。そのため、利点を慎重に計算し、追加コストと比較する必要がある。

---

## チップ

可用性を99%から99.9%に高めるのは比較的簡単です。しかし、可用性を99.9999%から99.99999%に高めるのは非常に難しく、コストがかかる。`ha-manager`の典型的なエラー検出とフェイルオーバーの時間は約2分なので、99.999%以上の可用性を得ることはできない。

---

## 15.1 必要条件

HAを始める前に、以下の条件を満たしていかなければならない：

- 少なくとも3つのクラスタノード（信頼できるクオーラムを得るため）
- VMとコンテナ用の共有ストレージ
- ハードウェアの冗長性
- 信頼性の高い「サーバー」コンポーネントを使用する
- ハードウェア・ウォッチドッグ - 利用できない場合は、Linuxカーネル・ソフトウェア・ウォッチドッグ（ソフトドッグ）にフォールバックします。
- オプションのハードウェア・フェンス装置

## 15.2 リソース

ha-managerが扱う主要な管理単位をリソースと呼ぶ。リソース ("service"とも呼ばれる) はサービスID (SID) によって一意に識別される。SIDはリソースのタイプとタイプ固有のIDから構成され、例えば vm:100 のようになる。この例では、vm (仮想マシン) タイプのリソースで、IDは100です。

仮想マシンとコンテナだ。ここでの基本的な考え方のひとつは、関連するソフトウェアをこのようなVMやコンテナにバンドルすることができるということだ。したがって、rgmanagerで行われたように、他のサービスからひとつの大きなサービスを構成する必要はない。一般的に、HAが管理するリソースは他のリソースに依存すべきではない。

## 15.3 マネジメント・タスク

このセクションでは、一般的な管理タスクの簡単な概要を説明する。最初のステップは、リソースのHAを有

```
# ha-manager add vm:100
```

効にすることである。これはリソースをHAリソース設定に追加することで行う。これはGUIを使っても、コマンドラインツールなどを使っても行うことができる：

HAスタックはリソースを起動し、実行し続けようとする。要求された」リソースの状態を設定できることに注意してください。例えば、HAスタックにリソースを停止させたい場合がある：

```
# ha-manager set vm:100 --state stopped
```

そしてまた後で始める：

```
# ha-manager set vm:100 --state started
```

通常のVMやコンテナ管理コマンドを使うこともできる。コマンドは自動的にHAスタックに転送される。

```
# qm start 100
```

は単に要求された状態をstartedに設定する。qm stopも同様で、要求された状態をstoppedに設定する。

### 注

HAスタックは完全に非同期で動作し、他のクラスタメンバーと通信する必要がある。そのため、このようなアクションの結果が表示されるまでには数秒かかります。

現在のHAリソース設定を表示するには

```
# ha-manager config  
vm:100  
    状態停止
```

そして、実際のHAマネージャーとリソースの状態を見ることができる：

```
# ha-manager status  
quorum OK  
master node1 (active, Wed Nov 23 11:07:23 2016)  
lrm elsa (active, Wed Nov 23 11:07:19 2016)  
service vm:100 (node1, started)
```

他のノードへのリソース移行を開始することもできます:

```
# ha-manager migrate vm:100 node2
```

これはオンライン・マイグレーションを使用し、VMを実行し続けようとする。オンライン・マイグレーション

```
# ha-manager relocate vm:100 node2
```

ンでは使用済みメモリをすべてネットワーク経由で転送する必要があるため、VMを停止してから新しいノードで再起動した方が速い場合がある。これはrelocateコマンドで実行できる:

最後に、以下のコマンドを使用して、HA構成からリソースを削除することができる:

```
# ha-manager remove vm:100
```

---

### 注

これはリソースを開始したり停止したりするものではない。

---

しかし、HA関連のタスクはすべてGUIで行えるので、コマンドラインを使う必要はまったくない。

## 15.4 仕組み

このセクションでは、Proxmox VE HAマネージャの内部について詳しく説明します。関係するすべてのデーモンと、それらがどのように連携して動作するかについて説明します。HAを提供するために、2つのデーモンが各ノード上で実行されます:

### プベ・ハ・ルム

ローカル・リソース・マネージャ (LRM) は、ローカル・ノード上で動作するサービスを制御する。

現在のマネージャー・ステータス・ファイルからサービスの要求状態を読み取り、それぞれのコマンドを実行します。

### PVE-HA-CRM

クラスタ全体の意思決定を行うクラスタ・リソース・マネージャ (CRM)。LRMにコマンドを送信し、結果を処理し、何か障害が発生した場合はリソースを他のノードに移動します。CRMはノードのフェンシングも行います。

---

---

## 注

ロックは分散設定ファイルシステム(pmxcfs)によって提供される。ロックは、各LRMが一度だけアクティブになって動作することを保証するために使用される。LRMはロックを保持しているときのみアクションを実行するため、ロックを取得できれば、障害が発生したノードをフェンスされたノードとしてマークすることができる。これにより、障害が発生したHAサービスを、未知の障害ノードからの干渉を受けずに安全に復旧させることができる。これはすべて、現在マネージャーマスターLOCKを保持しているCRMによって監視される。

---

### 15.4.1 サービス提供国

CRM はサービス状態の列挙を使用して現在のサービス状態を記録します。この状態はGUIに表示され、ha-

```
# ha-manager status
quorum OK
エルザ様 (アクティブ, 月 11 21 07:23:29 2016)
lrm elsa (active, Mon Nov 21 07:23:22 2016)
service ct:100 (elsa, stopped)
service ct:102 (elsa,
started) service vm:501
(elsa, started)
```

managerコマンドラインツールを使用して照会できます：

以下は可能な州のリストである：

#### 停止

サービスが停止している（LRMで確認）。LRMは停止したサービスがまだ実行中であることを検出した場合、そのサービスを再度停止します。

#### リクエスト・ストップ

サービスを停止すべきである。CRM は LRM からの確認を待つ。

#### 停止

停止要求の保留。しかし、CRMは今のところリクエストを受け取っていない。

#### 開始

サービスがアクティブな場合、LRM は、まだ実行中でなければ早急にサービスを開始する必要があります。サービスが失敗し、実行されていないことが検出された場合、LRMはサービスを再起動します（「[開始失敗ポリシー](#)」を参照）。

#### スタート

開始要求を保留しています。しかし、CRMはLRMからサービスが実行されていることを確認していない。

#### フェンス

サービスノードがクオーレートクラスターパーティション内にないため、ノードのフェンシングを待ちます（[フェンシングを参照](#)）。ノードのフェンシングが成功すると、サービスはリカバリ状態になります。

## 回復

サービスの復旧を待つ。HAマネージャは、サービスが実行できる新しいノードを探そうとする。この探索は、オンラインノードと定常ノードのリストだけでなく、サービスがグループメンバーであるかどうか、そのようなグループがどのように制限されているかにも依存する。新しい利用可能なノードが見つかり次第、サービスはそこに移動され、最初は停止状態になります。実行するように設定されている場合は、新しいノードが実行します。

## フリーズ

サービス状態には触れないでください。この状態はノードのリブート時やLRM демонの再起動時に使用します（[パッケージの更新を参照](#)）。

## 無視

あたかもサービスがHAによって管理されていないかのように振る舞う。サービスをHA構成から外すことなく、一時的に完全に管理したい場合に便利である。

## マイグレート

他のノードにサービス（ライブ）を移行する。

## エラー

LRMエラーによりサービスが無効になっている。手動による処置が必要（[エラーリカバリーを参照](#)）。

## キューに入れられた

サービスは新しく追加されたもので、CRMは今のところ見ていない。

## 使用不能

サービスが停止し、無効とマークされる

### 15.4.2 ローカル・リソース・マネージャー

ローカルリソースマネージャ（pve-ha-lrm）は起動時にデーモンとして起動され、HAクラスタがクオーレートされ、クラスタ全体のロックが機能するまで待機します。

3つの状態がある：

#### エージェントロックを待つ

LRMは我々の排他的ロックを待つ。これは、サービスが設定されていない場合、アイドル状態としても使用される。

#### アクティブ

LRMは排他的ロックを保持し、サービスが設定されている。

#### エージェントロックの紛失

LRMがロックを失った。これは障害が発生し、クオーラムが失われたことを意味する。

LRMはアクティブ状態になった後、/etc/pve/ha/manager\_statusにあるマネージャー・ステータス・ファイルを読み、所有しているサービスに対して実行しなければならないコマンドを決定する。各コマンドに対してワーカーが開始され、これらのワーカーは並列に実行され、デフォルトでは最大4つまでに制限され

ています。このデフォルト設定は、データセンター設定キー `max_worker` で変更できます。終了するとワーカープロセスは収集され、その結果はCRMのために保存されます。

---

## 注

デフォルト値の最大同時ワーカーは、特定のセットアップには適さないかもしれません。例えば、4つのライブマイグレーションが同時に発生する可能性があり、低速なネットワークや（メモリ的に）大きなサービスでネットワークが混雑する可能性があります。また、最悪の場合、`max_worker` の値を下げても、輻輳が最小になるようにしてください。逆に、特にパワフルでハイエンドなセットアップを使用している場合は、`max_worker` の値を増やした方がよいでしょう。

---

CRM が要求する各コマンドは UID で一意に識別できます。ワーカーが終了すると、その結果は処理され、LRM ステータスファイル /etc/pve/nodes/<nodename>/lrm\_status に書き込まれます。そこで CRM はそれを収集し、コマンドの出力に応じたステートマシンを動作させることができます。

通常、CRM と LRM 間の各サービスのアクションは常に同期されます。これは、CRM が UID で一意にマークされた状態を要求し、LRM がそのアクションを 1 回実行し、同じ UID で識別可能な結果を書き戻すことを意味します。これは、LRM が古いコマンドを実行しないために必要です。この動作の唯一の例外は、停止コマンドとエラーコマンドである。この2つは、生成される結果に依存せず、停止状態の場合は常に実行され、エラー状態の場合は1回実行される。

---

### 注

HA Stack はすべての動作をログに記録する。これはクラスタ内で何が、そしてなぜ何かが起こるのかを理解するのに役立つ。ここでは、LRM と CRM の両方のデーモンが何をしたかを確認することが重要です。サービスが存在するノードで journalctl -u pve-ha-lrm を、現在のマスターであるノードで pve-ha-crm に対して同じコマンドを使用することができます。

---

## 15.4.3 クラスター・リソース・マネージャー

クラスタリソースマネージャ(pve-ha-crm)は各ノード上で起動し、一度に1つのノードのみが保持できるマネージャロックを待ちます。マネージャロックの取得に成功したノードはCRMマスターに昇格します。

3つの状態がある：

### エージェントロックを待つ

CRM は我々の排他的ロックを待つ。サービスが設定されていない場合、これはアイドル状態としても使用されます。

### アクティブ

CRM は排他的ロックを保持し、サービスが設定されている。

### エージェントロックの紛失

CRM がロックを失った。これは障害が発生し、クオーラムが失われたことを意味する。

その主な仕事は、高可用性に設定されたサービスを管理し、常に要求された状態を強制しようとすることです。

---

ある。例えば、要求された状態が開始されたサービスは、まだ実行されていなければ開始される。クラッシュした場合は自動的に再スタートする。このように、CRMはLRMが実行する必要のあるアクションを指示する。

ノードがクラスタ・クオーラムから離脱すると、そのノードの状態はunknownに変わります。その後、現在のCRMが故障したノードのロックを確保できれば、サービスは盗まれて別のノードで再起動されます。

クラスタ・メンバがクラスタ・クオーラムから外れたと判断すると、LRMは新しいクオーラムが形成されるのを待ちます。クオーラムがない限り、ノードはウォッチドッグをリセットできません。これにより、ウォッチドッグがタイムアウトした後に再起動がトリガされます（これは60秒後に発生します）。

## 15.5 HAシミュレーター

Type	Status
quorum	OK
master	demohost2 (active, Fri Jun 30 09:41:54 2017)
lrm	demohost1 (active, Fri Jun 30 09:41:47 2017)
lrm	demohost2 (idle, Fri Jun 30 09:41:53 2017)

Resources						
Add	Edit	Remove	ID	State	Node	Max. Restart
			vm:501	stopped	demohost1	1
			ct:510	queued	demohost1	1
					prefer_node1	
					mygroup1	
Description						

HAシミュレータを使用することで、Proxmox VE HAソリューションのすべての機能をテストし、学習することができます。

デフォルトでは、シミュレータは6つのVMを持つ実際の3ノードクラスタの動作を監視し、テストすることができます。また、VMやコンテナを追加・削除することもできます。

実際のクラスタをセットアップしたり設定したりする必要はなく、HAシミュレータは箱

```
apt install pve-ha-simulator
```

他のProxmox VEパッケージがないDebianベースのシステムにもインストールできます。その場合、パッケージをダウンロードし、インストールしたいシステムにコピーする必要があります。ローカルファイルシステムからaptでパッケージをインストールすると、必要な依存関係も解決してくれます。

リモートマシンでシミュレータを起動するには、現在のシステムに X11 リダイレクトを設定する必要

```
ssh root@<IPofPVE> -Y
```

Windowsではmobaxtermで動作する。

シミュレータがインストールされた既存のProxmox VEに接続するか、ローカルのDebianベースのシステムに手動でインストールした後、以下の手順で試すことができます。

まず、シミュレータが現在の状態を保存し、デフォルトの設定を書き込む作業ディレクトリを作成する必要

`mkdir` 作業中

があります：

作成したディレクトリを `pve-ha-simulator` にパラメータとして渡すだけです：

## pve-ha-シミュレーター

その後、シミュレーションしたHAサービスを開始、停止、移行したり、ノード障害時に何が起こるかをチェックすることもできる。

## 15.6 構成

HAスタックはProxmox VE APIにうまく統合されています。例えば、`ha-manager` コマンドラインインターフェースや Proxmox VE ウェブインターフェースで HA を設定することができます。自動化ツールはAPIを直接使用することができます。

すべてのHA設定ファイルは`/etc/pve/ha/`内にあるので、クラスタノードに自動的に配布され、すべてのノードが同じHA設定を共有します。

### 15.6.1 リソース

The screenshot shows the Proxmox VE web interface with the sidebar expanded. The 'HA' section is selected, indicated by a blue background. The main content area displays two tabs: 'Status' and 'Resources'. The 'Status' tab shows a table with columns 'Type' and 'Status'. It lists a 'quorum' entry with 'OK' status and three 'master' entries for 'demohost2' (active), 'demohost1' (active), and 'demohost2' (idle). The 'Resources' tab shows a table with columns 'ID', 'State', 'Node', 'Max. Restart', 'Max. Reio...', 'Group', and 'Description'. It lists two resources: 'vm:501' (stopped) and 'ct:510' (queued), both associated with 'demohost1' and group 'prefer\_node1'.

リソース設定ファイル`/etc/pve/ha/resources.cfg`は、`ha-manager`が管理するリソースのリストを格納します。そのリスト内のリソース設定は以下のようになります：

```
<タイプ>: <名前  
      <プロパティ> <値  
      ...
```

リソースの種類から始まり、コロンで区切られたリソース固有の名前が続く。これはすべての ha-manager コマンドでリソースを一意に識別するために使用される（例: `vm:100` または `ct:101`）。次の行には追加のプロパティが含まれる：

## コメント:<文字列

説明

## group: <文字列

HAグループの識別子。

## max\_relocate: <整数> (0 - N) (デフォルト = 1)

サービスの起動に失敗した場合のサービス再配置の最大試行回数。

## max\_restart: <整数> (0 - N) (デフォルト = 1)

起動に失敗したノードでサービスを再起動する最大回数。

## state: <無効 | 有効 | 無視 | 開始 | 停止> (デフォルト =)

### を開始した)

要求されたリソースの状態。CRMはこの状態を読み取り、それに従って動作します。**有効**な  
は単にstartedの別名である。

### 開始

CRMはリソースの起動を試みます。起動に成功すると、サービス状態はstartedに設定されます。  
オン

ノードに障害が発生した場合、または起動に失敗した場合、リソースの回復を試みる。すべてが  
失敗した場合、サービスの状態はエラーに設定される。

### 停止

CRMはリソースを停止状態に保とうとするが、ノード障害時にはリソースの再配置を試みる。

### 使用不能

CRMはリソースを停止状態にしようとするが、リソースを再配置しようとはしない。

ノードの障害時に発生する。このステートの主な目的はエラー回復である。なぜなら、エラー  
状態からリソースを移動させる唯一の方法だからである。

### 無視

リソースはマネージャーステータスから削除され、CRMとLRMはリソースに触れなくなります。  
このリソースに影響する全ての{pve}APIコールが実行されます。APIコールが実行され、HAスタ  
ックを直接バイパスします。ソースがこの状態にある間、CRMコマンドは破棄されます。リソー  
スはノード障害時に再配置されません。

以下は、1つのVMと1つのコンテナを使った実例である。ご覧のように、これらのファイルの構文は実にシンプルなので、お気に入りのエディターを使ってファイルを読んだり編集したりすることも可能だ：

#### 設定例 (`/etc/pve/ha/resources.cfg`)

```
vm: 501
    状態開始 最大再配置数
    2

CT: 102
    # 注意：すべてデフォルト設定を使用すること
```

Add: Resource: Container/Virtual Machine

CT/VM ID:	510	Group:	prefer_node1
Max. Restart:	1	Request State:	started
Max. Relocate:	1		
Comment:			
<a href="#">Help</a>		<a href="#">Add</a>	

上記のコンフィグはha-managerコマンドラインツールを使って生成された:

```
# ha-manager add vm:501 --state started --max_relocate 2
# ha-manager add ct:102
```

## 15.6.2 グループ

Datacenter

Create   Edit   Remove				
Group ↑	restricted	nofailback	Nodes	Comment
mygroup1	No	No	node3:1,node4,node2:1,node1:2	complex group
mygroup2	Yes	No	node1,node2	simple restricted group
prefer_node1	No	No	node1	prefer node1

**Groups** is selected in the sidebar.

クラスタノードのグループを定義するには、HAグループ設定ファイル/etc/pve/ha/groups.cfgを使用します。リソースの実行を制限することができます。グループ設定は以下のようになります:

```
group: <グループ>
      ノード <ノードリスト>
      <プロパティ> <値>
      ...
```

**コメント:<文字列>**  
説明

ノード:<ノード>[:<プライ>]{,<ノード>[:<プライ>]}\*。

クラスタ・ノード・メンバーのリストで、各ノードに優先順位を与えることができる。リソースは

グループは、最も優先順位の高い利用可能なノード上で実行される。最も優先度の高いクラスのノードが多ければ、サービスはそれらのノードに分散される。優先順位は相対的な意味しか持ちません。

#### **nofailback: <boolean> (デフォルト = 0)**

CRM は最も優先度の高いノードでサービスを実行しようとします。より優先度の高いノードがオンラインになると、CRM はサービスをそのノードに移行します。nofailback を有効にすると、この動作が防止されます。

#### **restricted: <ブール値> (デフォルト = 0)**

制限付きグループにバインドされたリソースは、グループによって定義されたノード上でのみ実行できます。グループ・ノード・メンバーがオンラインでない場合、リソースは停止状態になります。非制限グループ上のリソースは、グループ・メンバがすべてオフラインの場合、どのクラスタ・ノードでも実行できますが、グループ・メンバがオンラインになるとすぐに元に戻ります。メンバが1人だけの無制限グループを使用して、優先ノードの動作を実装できます。

Node ↑	Memory usage %	CPU usage	Priority
demohost1	65.6 %	5.7% of 2CPUs	▼
demohost2	70.6 %	1.2% of 2CPUs	▼

一般的な要件として、リソースは特定のノードで実行する必要があります。通常、リソースは他のノードでも実行できるので、単一のメンバーで制限のないグループを定義できます：

```
# ha-manager groupadd prefer_node1 --nodes node1
```

大規模なクラスタでは、より詳細なフェイルオーバー動作を定義することが理にかなっています。例えば、可能であればnode1で一連のサービスを実行したいとします。node1が利用できない場合は、node2と

```
# ha-manager groupadd mygroup1 -nodes "node1:2,node2:1,node3:1,node4"
```

node3で均等にサービスを実行したい。これらのノードにも障害が発生した場合は、node4でサービスを実行する。これを実現するには、ノードリストを次のように設定します：

もう1つのユースケースは、リソースが特定のノード（例えばnode1とnode2）でのみ利用可能な他のリソースを使用する場合です。HAマネージャーが他のノードを使用しないようにする必要があるため、当該ノードで制限グループを作成する必要がある：

```
# ha-manager groupadd mygroup2 -nodes "node1,node2" -restricted
```

上記のコマンドは、以下のグループ設定ファイルを作成した：

### 設定例 (`/etc/pve/ha/groups.cfg`)

```
グループ: prefer_node1
    ノード node1

グループ: mygroup1
    ノード2:1, ノード4, ノード1:2, ノード3:1

グループ: mygroup2
    ノード
        node2, node1
        restricted 1
```

`nofailback`オプションは、管理タスク中に不要なリソースの移動を回避するのに便利です。例えば、あるサービスをグループ内で最も高い優先度を持たないノードに移行する必要がある場合、`nofailback`オプションを設定することで、このサービスを即座に戻さないようにHAマネージャに指示する必要があります。

もう1つのシナリオは、あるサービスがフェンスにかけられ、別のノードに復旧した場合だ。管理者はフェンスで保護されたノードを修復して再びオンラインにし、障害の原因を調査して、再び安定して動作するかどうかを確認しようとします。`nofailback`フラグを設定すると、復旧したサービスがそのままフェンスで保護されたノードに戻るのを防ぐことができます。

## 15.7 フェンシング

ノードの障害時には、フェンシングによって、誤ったノードがオフラインであることが保証される。これは、リソースが別のノードで復旧したときに2度実行されないようにするために必要です。これがなければ、別のノードでリソースを回復することができないため、これは本当に重要なタスクです。

もしノードがフェンスされなければ、共有リソースにアクセスできるかもしれない未知の状態になる。これは本当に危険だ！ストレージ以外の全てのネットワークが壊れたとしよう。パブリック・ネットワークからはアクセスできないが、VMはまだ稼働しており、共有ストレージに書き込んでいる。

このVMを別のノードで起動すると、両方のノードから書き込みを行うため、危険なレースコンディションが発生する。このような状態は、全てのVMデータを破壊し、VM全体を使用不能にする可能性がある。また、ストレージが複数マウントから保護されている場合、リカバリに失敗する可能性もある。

### 15.7.1 プロックスモックスVEフェンス

例えば、ノードからの電力を遮断したり、通信を完全に無効にするフェンス・デバイスなどだ。これらは非常に高価であることが多く、システムにさらに重要なコンポーネントを追加することになる。

そこで、外部ハードウェアを追加する必要のない、よりシンプルなフェンシング方法を統合したいと考えた。これはウォッチドッグ・タイマーを使って実現できる。

#### 可能なフェンスの方法

- 外部電源スイッチ
- スイッチ上のネットワーク・トラフィックを完全に無効にすることで、ノードを隔離する。
- ウォッチドッグタイマーを使ったセルフフェンシング

ウォッチドッグ・タイマーは、マイクロ・コントローラーが登場して以来、重要で信頼性の高いシステムで広く使われてきた。ウォッチドッグ・タイマーは多くの場合、独立したシンプルな集積回路であり、コンピュータの誤動作を検出し、回復するために使用される。

通常動作中、`ha-manager`は定期的にウォッチドッグタイマーをリセットし、タイマーが経過しないようにします。ハードウェア障害またはプログラムエラーにより、コンピューターがウォッチドッグのリセットに失敗した場合、タイマーが経過し、サーバー全体のリセット（再起動）がトリガーされます。

最近のサーバー・マザーボードには、このようなハードウェア・ウォッチドッグが搭載されていることが多いが、これらを設定する必要がある。ウォッチドッグが利用できない、あるいは設定されていない場合、Linuxカーネルのソフトドッグに頼ることになる。それでも信頼性はありますが、サーバーのハードウェアから独立していないため、ハードウェア・ウォッチドッグよりも信頼性は低くなります。

### 15.7.2 ハードウェア・ウォッチドッグの設定

デフォルトでは、セキュリティ上の理由から、すべてのハードウェア・ウォッチドッグ・モジュールはブロックされている。正しく初期化されなければ、装填された銃のようなものである。ハードウェア・ウォッチドッグを有効にするには、ロードするモジュールを

例えば、`/etc/default/pve-ha-manager`:

```
# ウォッチドッグモジュールを選択 (デフォルトはソフトドッグ)
WATCHDOG_MODULE=iTCO_wdt
```

このコンフィギュレーションは`watchdog-mux`サービスによって読み込まれ、起動時に指定されたモジュールがロードされる。

### 15.7.3 リカバー・フェンス・サービス

ノードに障害が発生し、そのフェンシングが成功した後、CRMは障害が発生したノードからオンライン状態のノードにサービスを移動しようとします。

これらのサービスが回復するノードの選択は、リソースグループによって影響される。

設定、現在アクティブなノードのリスト、およびそれぞれのアクティブなサービス数。

CRMはまず、（グループ設定から）ユーザーが選択したノードと利用可能なノードとの交点から集合を構築します。次に、最も優先順位の高いノードのサブセットを選択し、最後に最もアクティブなサービス数が少ないノードを選択します。これにより、ノードが過負荷になる可能性を最小限に抑えます。

**注意**

ノード障害が発生すると、CRMは残りのノードにサービスを分散します。これにより、これらのノードのサービス数が増加し、特に小規模なクラスタでは高負荷になる可能性があります。このような最悪のケースに対応できるようにクラスタを設計してください。

## 15.8 スタート失敗ポリシー

開始失敗ポリシーは、ノード上でサービスが1回以上開始できなかった場合に有効になります。このポリシーを使用して、同じノードで再起動をトリガする頻度や、サービスを別のノードで開始できるように再配置する頻度を設定できます。このポリシーの目的は、特定のノードで共有リソースが一時的に利用できなくなるのを回避することです。例えば、共有ストレージがネットワークの問題などでクオリテート・ノードで利用できなくなったが、他のノードではまだ利用可能な場合、リロケート・ポリシーによってサービスを開始することができます。

各リソースに固有の設定を行うことができる2つのサービス開始回復ポリシー設定があります。

**max\_restart**

実際のノードで障害が発生したサービスの再起動を試行する最大回数。デフォルトは1回です。

**max\_relocate**

サービスを別のノードに再配置する最大試行回数。実際のノードでmax\_restart値を超えた場合にのみ再配置が行われます。デフォルトは1です。

**注**

再配置カウントの状態は、サービスの起動が少なくとも1回成功した場合にのみゼロにリセットされます。つまり、エラーが修正されずにサービスが再スタートした場合、再スタートポリシーだけが繰り返されます。

## 15.9 エラーリカバリー

試行錯誤の末、サービスの状態を回復できなかった場合、エラー状態になる。この状態では、サービスはも

```
# ha-manager set vm:100 --state disabled
```

うHAスタックに触れられない。唯一の方法は、サービスを無効にすることである：

これはウェブ・インターフェイスでも可能です。

エラー状態から回復するには、次のようにする：

- リソースを安全で一貫性のある状態に戻す（例：サービスを停止できなかった場合、そのプロセスを強制終了する）。
- リソースを無効にしてエラーフラグを取り除く
- 失敗の原因となったエラーを修正する
- すべてのエラーを修正した後、サービスの再開をリクエストすることができます。

## 15.10 パッケージ・アップデート

ha-managerをアップデートする際は、様々な理由から、一度に全部をアップデートするのではなく、1つの

ノードを次々にアップデートする必要があります。第一に、私たちはソフトウェアを徹底的にテストしていますが、あなたの特定のセットアップに影響するバグを完全に排除することはできません。1ノードずつアップデートを行い、アップデート終了後に各ノードの機能をチェックすることは、最終的な問題からの回復に役立ちますが、一度にすべてをアップデートするとクラスタが壊れてしまう可能性があり、一般的には良いやり方ではありません。

また、Proxmox VE HAスタックは、クラスタとローカルリソースマネージャの間でアクションを実行するためにリクエストアクノリッジプロトコルを使用します。再起動のために、LRMはCRMにすべてのサービスをフリーズする要求を行います。これにより、LRMが再起動する短時間の間、クラスタがこれらのサービスに触れることを防ぎます。その後、LRMは再起動中にウォッチドッグを安全に閉じることができます。このような再起動は通常パッケージの更新中に起こり、すでに述べたように、LRMからの要求を承認するためにアクティブなマスター CRM が必要です。そうでない場合、更新処理に時間がかかりすぎ、最悪の場合、ウォッチドッグによってリセットされる可能性があります。

## 15.11 ノードのメンテナンス

ハードウェアの交換や新しいカーネルイメージのインストールなど、ノードのメンテナンスが必要になることがある。これはHAスタック使用中にも適用される。

HAスタックは、主に2種類のメンテナンスをサポートすることができる：

- 一般的なシャットダウンやリブートについては、[シャットダウンポリシー](#)を参照してください。
- シャットダウンや再起動を必要としないメンテナンスの場合、または1回の再起動だけで自動的にオフにならないメンテナンスの場合、手動メンテナンスマードを有効にすることができます。

### 15.11.1 メンテナンスマード

手動メンテナンスマードを使用すると、ノードをHA動作不可としてマークし、HAによって管理されるすべてのサービスが他のノードに移行するように促すことができます。

これらの移行のターゲットとなるノードは、現在利用可能な他のノードから選択され、HAグループの構成と設定されているクラスタリソーススケジューラ（CRS）モードによって決定されます。各移行の間、元のノードはHAマネージャの状態に記録されるため、メンテナンスマードが無効になってノードがオンラインに戻ると、サービスは再び自動的に移行されます。

現在、ha-manager CLIツールを使用して、メンテナンスマードを有効または無効にすることができます。

#### ノードのメンテナンスマードの有効化

```
# ha-manager crm-command node-maintenance enable NODENAME
```

これはCRMコマンドをキューに入れ、マネージャがこのコマンドを処理すると、マネージャのステータスにメンテナンスマードのリクエストが記録されます。これにより、メンテナンスマードにしたいノードだけでなく、どのノードでもコマンドを送信することができます。

各ノードのLRMがコマンドを拾うと、それ自身を利用不可とマークするが、すべてのマイグレーションコマンドを処理する。つまり、LRMのセルフフェンシング・ウォッチドッグは、すべてのアクティブなサービスが移動し、すべての実行中のワーカーが終了するまでアクティブなままでです。

LRMのステータスは、全てのサービスが移動した時だけでなく、LRMが要求されたステータスをピックアップ

した時点でメンテナンスモードと表示されることに注意してください、このユーザーエクスペリエンスは将来的に改善される予定です。今のところ、ノード上にアクティブな HA サービスが残っていないか確認したり、次のようなログ行を見たりすることで、ノードがいつメンテナンスモードへの移行を終了したかを知ることができます: pve-ha-lrm[PID]: watchdog closed (disabled) .

---

### 注

手動メンテナンスモードは、ノードのリブート時には自動的に削除されませんが、ha-manager CLIを使用して手動で解除するか、manager-statusを手動でクリアした場合にのみ削除されます。

---

### ノードのメンテナンスモードを無効にする

```
# ha-manager crm-command node-maintenance disable NODENAME
```

手動メンテナンスモードを無効にするプロセスは、有効にするプロセスと同様です。上記のha-manager CLIコマンドを使用すると、CRMコマンドがキューに入れられ、それが処理されると、それぞれのLRMノードが再び使用可能にマークされます。

メンテナンスモードを解除すると、メンテナンスモードがアクティブになったときにノード上にあったサービスはすべて元に戻ります。

### 15.11.2 シャットダウン・ポリシー

以下に、ノードのシャットダウンに関するさまざまなHAポリシーの説明を示します。現在のところ、後方互換性のために条件付きがデフォルトとなっています。ユーザーによっては、*Migrate*の方がより期待通りの動作をすると感じるかもしれません。

シャットダウン・ポリシーは、Web UI (Datacenter → Options → HA Settings) で設定することも、datacenter.cfgで直接設定することもできる：

```
ha: shutdown_policy=<値>。
```

#### マイグレート

ローカルリソースマネージャ(LRM)がシャットダウンリクエストを受け取り、このポリシーが有効になると、現在のHAマネージャにとって利用できないものとしてマークされる。これは、現在このノードにあるすべてのHAサービスの移行をトリガーする。LRMは、実行中のサービスがすべて移動されるまで、シャットダウン処理を遅らせようとする。しかし、これは実行中のサービスが別のノードに移行できることを想定している。言い換えると、ハードウェア・パススルーを使用するなどして、サービスがローカルにバインドされていない必要があります。グループメンバーでないノードは、グループメンバーが利用できない場合、実行可能なターゲットとみなされるため、一部のノードだけを選択してHAグループを利用する場合でも、このポリシーを使用することができる。しかし、グループを制限ノードとしてマークすることで、HAマネージャに、選択されたノードセット以外ではサービスを実行できないことを伝えることができる。これらのノードがすべて利用できない場合、手動で介入するまでシャットダウンはハングします。シャットダウンされたノードが再びオンラインに戻ると、その間にまだ手動で移行されていなければ、以前に移動されたサービスは元に戻される。

---

#### 注

---

ウォッチャドッグはシャットダウンの移行プロセス中もアクティブである。ノードがクオーラムを失った場合、それはフェンスされ、サービスは回復されます。

---

現在メンテナンス中のノードで（以前に停止した）サービスを開始する場合、サービスを移動して別の利用可能なノードで開始できるように、そのノードをフェンスで囲む必要があります。

### フェイルオーバー

このモードでは、現在のノードがすぐにオンラインにならない場合、すべてのサービスが停止され、復旧される。一度に多くのノードがパワーオフされるとVMのライブマイグレーションが不可能になる可能性があるが、それでもHAサービスを確実にリカバリしてできるだけ早く再スタートさせたい場合に、クラスタ規模でメンテナンスを行う場合に便利だ。

### フリーズ

このモードでは、すべてのサービスが停止してフリーズし、現在のノードが再びオンラインになるまで復旧しない。

## 条件付き

条件付きシャットダウンポリシーは、シャットダウンまたは再起動が要求されたかどうかを自動的に検出し、それに応じて動作を変更します。

### シャットダウン

シャットダウン（電源オフ）は通常、ノードがしばらくの間停止する予定がある場合に実行されます。この場合、LRMはすべての管理サービスを停止します。これは、その後、他のノードがこれらのサービスを引き継ぐことを意味します。

---

#### 注

最近のハードウェアは大量のメモリ（RAM）を搭載している。そのため、すべてのリソースを停止してから再起動し、すべてのRAMのオンラインマイグレーションを回避します。オンライン・マイグレーションを使用したい場合は、ノードをシャットダウンする前に手動で起動する必要があります。

---

### リブート

ノードの再起動は*reboot*コマンドで行う。これは通常、新しいカーネルをインストールした後に行われます。これは「シャットダウン」とは異なることに注意してください。

LRM は CRM に再起動を指示し、CRM がすべてのリソースをフリーズ状態にするまで待ちます（[Package Updates](#) と同じメカニズム）。これにより、これらのリソースが他のノードに移動することを防ぎます。代わりに、CRM は同じノード上で再起動後にリソースを起動します。

### 手動リソース移動

最後になるが、ノードをシャットダウンまたは再起動する前に、手動でリソースを他のノードに移動することもできる。その利点は、完全な制御が可能で、オンラインマイグレーションを使用するかどうかを決定できることです。

---

#### 注

`pve-ha-crm`、`pve-ha-lrm`、`watchdog-mux`のようなサービスは殺さないでください。これらは

---

ウォッチドッグを管理・使用しているため、ノードのリブートやリセットを引き起こす可能性があります。

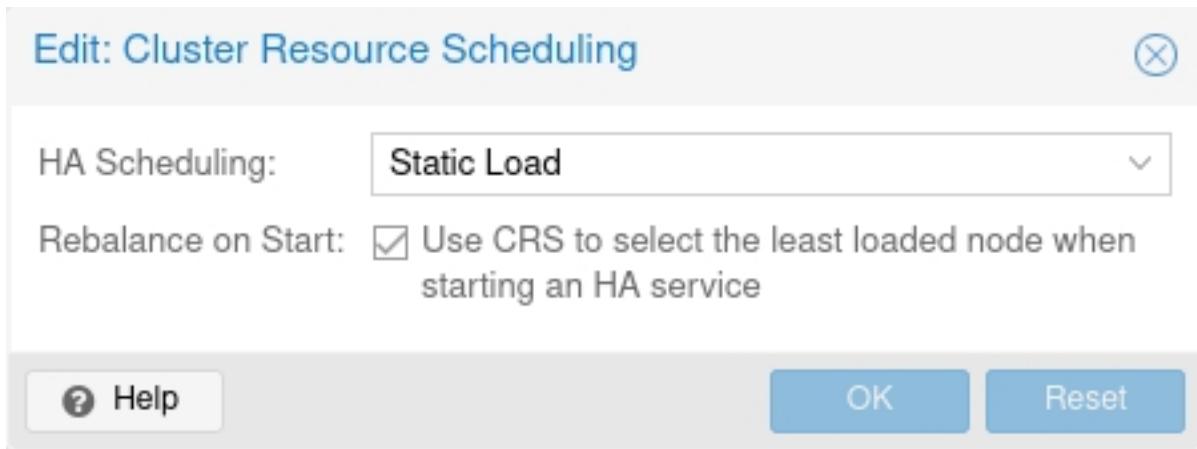
## 15.12 クラスタ・リソース・スケジューリング

クラスタリソーススケジューラ(CRS)モードは、HAがサービスを復旧させるためのノードを以下のように選択する方法を制御する。

また、シャットダウンポリシーによってトリガーされる移行にも使用できます。デフォルトのモードはベー

```
crs: ha=静的
```

シックで、Web UI (Datacenter → Options) で変更するか、datacenter.cfgで直接変更できます：



変更は次の監督ラウンドから有効になる（数秒後）。

復旧や移行が必要な各サービスに対して、スケジューラは、そのサービスのグループ内で最も優先順位の高いノードの中から最適なノードを繰り返し選択する。

#### 注

将来的には、（静的および動的な）負荷分散のモードを追加する計画もある。

### 15.12.1 基本スケジューラー

各ノードのアクティブなHAサービスの数は、復旧ノードを選択するために使用される。HA管理されていないサービスは現在カウントされていません。

### 15.12.2 静的負荷スケジューラ



#### 重要

スタティック・モードはまだ技術レビューだ。

各ノードのHAサービスからの静的な利用情報は、回復ノードを選択するために使用される。HAで管理されていないサービスの利用は現在のところ考慮されていない。

この選択のために、各ノードは、関連するゲスト構成からのCPUとメモリ使用量を使用して、サービスがそのノード上ですでに実行されているかのように順番に検討されます。次に、このような各選択肢について、すべてのノードのCPUとメモリの使用量が考慮され、メモリは本当に限られたリソースであるため、より重

み付けされます。CPUとメモリの両方について、ノード間で最も高い使用率（理想的にはどのノードもオーバーコミットされるべきではないため、より重み付けされます）と、すべてのノードの平均使用率（より高いコミット率のノードがすでにある場合に区別できるようにするため）が考慮されます。

**重要**

サービス数が多ければ多いほど、可能な組み合わせも増えるので、数千ものHAマネージドサービスを利用している場合は、今のところ利用を推奨していない。

### 15.12.3 CRSのスケジューリングポイント

CRS アルゴリズムは毎ラウンドすべてのサービスに適用されるわけではありません。なぜなら、これは多数の定的なマイグレーションを意味するからです。ワークロードによっては、常時バランスするよりもクラスタに負担がかかる可能性があります。Proxmox VE HAマネージャがサービスを現在のノードに維持することを好むのはそのためです。

CRSは現在、以下のスケジューリングポイントで使用されている：

- サービス復旧（常にアクティブ）。アクティブなHAサービスを持つノードが故障した場合、そのノードの全サービスを他のノードに復旧させる必要がある。CRSアルゴリズムは、残りのノードにバランスよく復旧させるために使用される。
- HAグループ設定の変更（常にアクティブ）あるノードがグループから削除された場合、またはそのノードの優先度が低下した場合、HAスタックはCRSアルゴリズムを使用して、適応された優先度制約に一致する、そのグループ内のHAサービスのための新しいターゲットノードを見つける。
- HAサービス停止→移動開始（オプトイン）。停止しているサービスを開始するように要求することは、CRSアルゴリズムに従って最適なノードをチェックする良い機会です。

特にディスクボリュームが共有ストレージ上にある場合は、それらを移動し始めるよりも安上がりです。あなたは

を有効にするには、データセンター設定で **ha-rebalance-on-start** CRS オプションを設定します。このオプションは Web UI の Datacenter → Options → Cluster Resource Scheduling でも変更できます。

## 第16章

# バックアップと復元

Proxmox VEは、各ストレージおよび各ゲストシステムタイプの機能を使用し、完全に統合されたソリューションを提供します。Proxmox VEは、各ストレージおよび各ゲストシステムのタイプの機能を使用し、完全に統合されたソリューションを提供します。これにより、システム管理者は、バックアップの一貫性とゲストシステムのダウンタイムをモードオプションで微調整することができます。

Proxmox VEのバックアップは、常にVM/CTのコンフィギュレーションとすべてのデータを含むフルバックアップです。バックアップはGUIまたはvzdumpコマンドラインツールから開始できます。

### バックアップ・ストレージ

バックアップを実行する前に、バックアップ・ストレージを定義する必要があります。ストレージを追加する方法については、[ストレージのドキュメント](#)を参照してください。Proxmox Backup Serverストレージの場合、バックアップは複製解除されたチャンクとメタデータとして保存され、ファイルレベルのストレージの場合、バックアップは通常のファイルとして保存されます。Proxmox Backup Serverは高度な機能を備えているため、専用ホストで使用することをお勧めします。NFSサーバを使用するのも良い方法です。どちらの場合も、バックアップをテープドライブに保存し、オフサイトアーカイブにすることができます。

### 定期バックアップ

バックアップジョブは、選択可能なノードとゲストシステムに対して、特定の日時に自動的に実行されるようにスケジュールすることができます。詳細はバックアップジョブセクションを参照してください。

## 16.1 バックアップ・モード

一貫性（オプションモード）を提供するには、ゲストのタイプによっていくつかの方法がある。

VMのバックアップモード：

### 停止モード

このモードでは、VMのオペレーションが短時間停止する代わりに、バックアップの一貫性が最も高くなります。このモードは、VMの整然としたシャットダウンを実行し、バックグラウンドでQEMUプロセスを実行してVMデータをバックアップします。バックアップが開始されると、VMはフル稼働モードに移行する。ライブ・バックアップ機能を使用することで、一貫性が保証される。

## サスPENDモード

このモードは互換性のために提供されており、スナップショット・モードを呼び出す前にVMを一時停止します。VMをサスPENDするとダウンタイムが長くなり、データの一貫性が向上するとは限らないため、代わりにスナップショット・モードの使用を推奨します。

## スナップショットモード

このモードは、わずかな不整合リスクを伴いますが、最小の運用停止時間为您提供します。このモードはProxmox VEのライブバックアップを実行することで機能し、VMの実行中にデータブロックがコピーされます。ゲストエージェントが有効（エージェント：1）で実行中の場合、`guest-fsfreeze-freeze`と`guest-fsfreeze-thaw`を呼び出して整合性を向上させます。

QemuServer用Proxmox VEライブバックアップの技術概要は、[こちらをご覧ください](#)。

### 注

Proxmox VEライブバックアップは、あらゆるストレージタイプでスナップショットのようなセマンティクスを提供します。基礎となるストレージがスナップショットをサポートしている必要はありません。また、バックアップはバックグラウンドのQEMUプロセスで実行されるため、VMディスクがQEMUによって読み込まれている間、停止しているVMは短時間実行されているように見えます。しかし、VM自体は起動せず、ディスクだけが読み込まれます。

## コンテナのバックアップモード：

### 停止モード

バックアップの間、コンテナを停止する。このため、ダウンタイムが非常に長くなる可能性がある。

### サスPENDモード

このモードでは、`rsync`を使用してコンテナのデータを一時的な場所にコピーする（オプション`--tmpdir`を参照）。その後、コンテナが一時停止され、2回目の`rsync`で変更されたファイルがコピーされる。その後、コンテナが再び起動（再開）される。この結果、ダウンタイムは最小限に抑えられるが、コンテナのコピーを保持するための追加の領域が必要になる。

コンテナがローカル・ファイル・システム上にあり、バックアップのターゲット・ストレージがNFS/CIFSサーバーである場合、`-tmpdir`もローカル・ファイル・システム上に存在するように設

定すべきである。バックアップ・ストレージがNFSサーバーの場合、サスPEND・モードでACLを使用してローカル・コンテナをバックアップする場合にも、ローカルの`tmpdir`を使用する必要がある。

◦

## スナップショットモード

このモードでは、基礎となるストレージのスナップショット機能を使用する。まず、データの一貫性を確保するためにコンテナをサスPENDする。コンテナのボリュームの一時スナップショットが作成され、スナップショットの内容はtarファイルにアーカイブされる。最後に、一時スナップショットは再び削除される。

---

### 注

スナップショット・モードでは、すべてのバックアップ・ボリュームがスナップショットをサポートするストレージ上にある必要があります。`backup=no`マウントポイントオプションを使用すると、個々のボリュームをバックアップから除外することができます（したがって、この要件も除外されます）。

---

## 注

デフォルトでは、ルート・ディスク・マウント・ポイント以外の追加マウント・ポイントはバックアップに含まれません。ボリューム マウント ポイントについては、**バックアップ オプション**を設定して、マウント ポイントをバックアップに含めることができます。デバイスマウントとバインドマウントは、コンテンツが Proxmox VE ストレージ ライブラリの外部で管理されるため、バックアップされることはありません。

### 16.1.1 VMバックアップフリッキング

VMのバックアップが開始されると、QEMUはブロック層に「copy-before-write」フィルターをインストールします。このフィルタは、新しいゲストの書き込み時に、バックアップにまだ必要な古いデータが最初にバックアップターゲットに送信されることを保証します。ゲストの書き込みはこの操作が終了するまでブロックされるため、まだバックアップされていないセクタへのゲストのIOはバックアップターゲットの速度によって制限されます。

バックアップフリーシングでは、そのような古いデータはバックアップターゲットに直接送られるのではなく、フリーシングイメージにキャッシュされます。これはゲストのIOパフォーマンスを助け、特定のシナリオではハングアップを防ぐこともできますが、その代償としてより多くのストレージ容量を必要とします。例えば `vzdump 123 --fleecing enabled=1,storage=local-lvm` を使ってバックアップフリーシングを有効にし、ストレージ `local-lvm` 上にフリーシングイメージを作成します。

フリークするストレージは、シンプロビジョニングとディスカードサポートを備えた高速なローカルストレージでなければならない。例えば、LVM-thin、RBD、ストレージ構成にスベース<sup>1</sup>を持つZFS、多くのファイルベースのストレージなどである。理想的には、fleecingストレージは専用ストレージで、フル稼働しても他のゲストには影響せず、バックアップに失敗するだけです。バックアップされたfleecingイメージの一部は、スペース使用量を低く抑えるために破棄されます。

廃棄をサポートしないファイルベースのストレージ（例えば、バージョン4.2以前のNFS）では、ストレージ設定で `preallocation` を `off` に設定すべきである。`qcow2`（自動的にfleec-のフォーマットとして使用される）と組み合わせて使用する。

ストレージがサポートしている場合）、この方法には、すでに割り当てられている画像の一部を後で再利用できるという利点があり、これでもかなりの容量を節約することができる。

## 警告



シンプリー・プロビジョニングされていないストレージ、例えばLVMやスパース・オプションのないZFSでは、オリジナルディスクのフルサイズを、前もってフリークイメージ用に予約しておく必要があります。シンプリー・プロビジョニングされたストレージでは、バックアップが別のディスクでビギーになっている間に、ゲストがディスク全体を再書き込みした場合にのみ、フリージングイメージはオリジナルイメージと同じサイズまで成長できます。

## 16.2 バックアップファイル名

vzdump の新しいバージョンでは、ゲストの種類とバックアップ時間がファイル名にエンコードされます。

vzdump-1xc-105-2009\_10\_09-11\_04\_43.tar

これにより、複数のバックアップを同じディレクトリに保存することができます。様々な保持オプションを使用して、保持するバックアップの数を制限できます。

## 16.3 バックアップファイルの圧縮

バックアップファイルは次のいずれかのアルゴリズムで圧縮できます。<sup>1</sup>[gzip](#)<sup>2</sup>または[zstd](#)<sup>3</sup>.

<sup>1</sup>Lempel-Ziv-Oberhumer 可逆データ圧縮アルゴリズム <https://en.wikipedia.org/wiki/Lempel-Ziv-Oberhumer>

<sup>2</sup>gzip - DEFLATE アルゴリズムに基づく <https://en.wikipedia.org/wiki/Gzip>

<sup>3</sup>Zstandard 可逆データ圧縮アルゴリズム <https://en.wikipedia.org/wiki/Zstandard>

現在、Zstandard (zstd)はこれら3つのアルゴリズムの中で最速である。マルチスレッドは lzo と gzip に対する zstd のもう一つの利点です。Lzo と gzip はより広く使われており、しばしばデフォルトでインストールされます。

pigz<sup>4</sup> をインストールすることができる。pigz と zstd では、スレッド数/コア数を調整できます。以下の[設定オプション](#)を参照してください。

バックアップファイル名の拡張子から、どの圧縮アルゴリズムでバックアップが作成されたかを判断することができます。

.zst	Zstandard (zstd) 圧縮
.gz または .tgz	gzip 圧縮
.lzo	lzo 圧縮

バックアップファイル名が上記のファイル拡張子のいずれかで終わっていない場合、vzdumpによって圧縮されていない。

## 16.4 バックアップの暗号化

Proxmox Backup Serverストレージでは、オプションでバックアップのクライアント側暗号化を設定できます。

<sup>4</sup>pigz - gzip の並列実装 <https://zlib.net/pigz/>

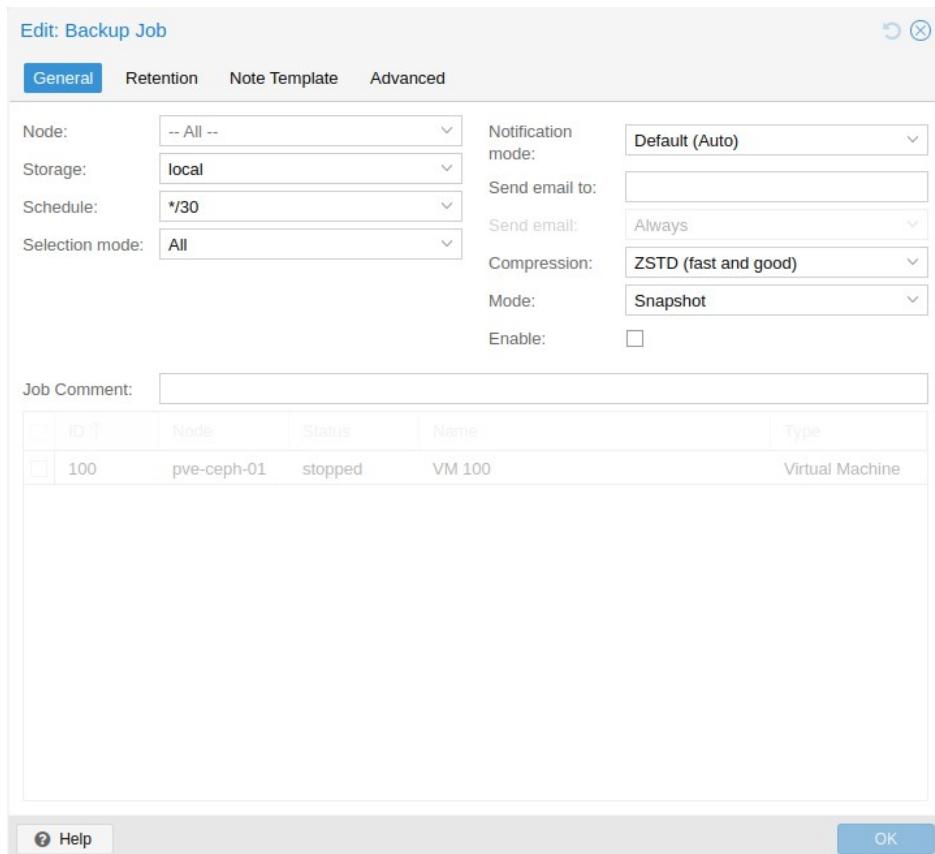
## 16.5 バックアップの仕事

The screenshot shows the Proxmox VE Datacenter interface. On the left is a sidebar with various management sections: Datacenter, Search, Summary, Notes, Cluster, Ceph, Options, Storage, Backup (which is selected and highlighted in blue), Replication, Permissions, Users, API Tokens, Two Factor, Groups, Pools, Roles, Realms, HA, SDN, Zones, VNets, Options, IPAM, ACME, Firewall, Metric Server, Resource Mappings, Notifications, and Support. At the top right are buttons for Help, Add, Remove, Edit, Job Detail, Run now, and Schedule Simulator. Below these buttons is a table with columns: Enabled, Node, Schedule, Next Run, Storage, Comm..., Retention, and Selection. The table shows one job entry: Enabled is checked, Node is set to 'All', Schedule is set to '\*/30', Next Run is '2024-04-23 11:00:00', Storage is 'local', Comm... is 'Fallback from storag...', and Selection is 'All'.

手動でバックアップをトリガーする以外に、すべての仮想ゲストまたは選択した仮想ゲストをストレージにバックアップする定期的なジョブを設定することもできます。 ジョブの管理は、UI の *Datacenter* → *Backup* または

`/cluster/backup` API エンドポイントを使用します。どちらも `/etc/pve/jobs.cfg` にジョブ・エントリを生成します。

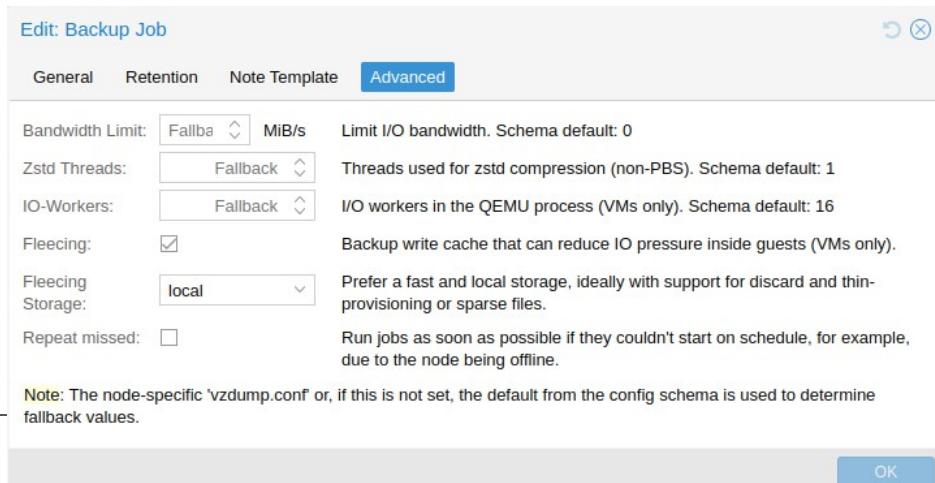
`pvescheduler` デーモンによって解析され、実行される。



ジョブは全てのクラスタノードまたは特定のノードに対して設定され、指定されたスケジュールに従って実行されます。スケジュールのフォーマットは `systemd` のカレンダーイベントによく似ています。詳細は [カレンダーイベントのセクション](#)を参照してください。UI の `Schedule` フィールドは自由に編集することができ、そのドロップダウンリストには出発点として使えるいくつかの例が含まれています。

ストレージやノードの設定より優先されるジョブ固有の [保持](#)オプションや、バックアップと一緒に保存される追加情報の [メモ用](#)テンプレートを設定できます。

スケジュールされたバックアップは、スケジュールされた時間中にホストがオフラインであったり、`pvescheduler`が無効であったりすると実行されないので、それをキャッチアップするための動作を設定することができます。Repeat missedオプション (UIのAdvancedタブ、configのrepeat-missed) を有効にすると、見逃したジョブができるだけ早く実行するようスケジューラに指示できます。

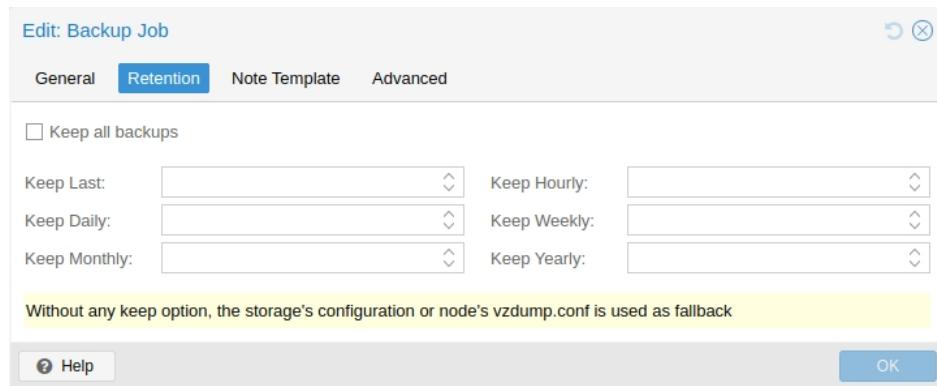


バックアップのパフォーマンスを調整するための設定がいくつかある（そのうちのいくつかはUIの*Advanced*タブで公開されている）。最も注目すべきは、IO帯域幅を制限するためのbwlimitである。com-pressorに使用されるスレッドの量は、それぞれpigz（gzipの代わり）、zstd設定で制御できる。さらに

は ionice (BFQスケジューラ使用時)、パフォーマンス設定の一部として max-workers (VMバックアップにのみ影響)、pbs-entries-max (コンテナバックアップにのみ影響) である。詳細は[設定オプションを参照](#)。

## 16.6 バックアップの保持

prune-backupsオプションを使用すると、柔軟な方法で保持するバックアップを指定することができます。



以下の保持オプションがある：

### **keep-all <ブール値>**

すべてのバックアップを保持する。これが真の場合、他のオプションは設定できません。

### **キープラスト <N>**

最後の<N>個のバックアップを保持する。

### **キープ・アワー<N>**

直近<N>時間のバックアップを保持する。1時間に複数のバックアップがある場合は、最新のものだけが保持されます。

### **キープデイリー<N>**

過去<N>日間のバックアップを保持する。1日に複数のバックアップがある場合は、最新のものだけが保持されます。

### **キープ・ウィークリー<N>**

過去<N>週間分のバックアップを保持する。1週間に複数のバックアップがある場合は、最新のものだけが保存されます。

---

### 注

週は月曜日に始まり、日曜日に終わります。このソフトウェアはISOの週日付システムを使用し、年末の週を

---

正しく処理します。

---

### キープ・マンスリー<N>

過去<N>ヶ月分のバックアップを保持する。1ヶ月に複数のバックアップがある場合は、最新のものが保存されます。

## 年間 <N>

過去<N>年間のバックアップを保持する。1年間に複数のバックアップがある場合は、最新のものだけが保存されます。

保持オプションは上記の順序で処理されます。各オプションは、その期間内のバックアップのみを対象とします。次のオプションでは、すでにカバーされているバックアップは処理されません。古いバックアップのみを考慮します。

使用したい保持オプションをカンマ区切りのリストで指定する：

```
# vzdump 777 --剪定バックアップ keep-last=3,keep-daily=13,keep-yearly=9
```

prune-backupsを直接vzdumpに渡すこともできますが、ストレージレベルで設定した方が賢明な場合が多く、ウェブインターフェースを介して行うことができます。

---

### 注

古い maxfiles オプションは非推奨であり、keep-last か、もしくは maxfiles は、keep-all で無制限に保持するために 0 にした。

---

## 16.6.1 プルーン・シミュレーター

Proxmox Backup Serverドキュメントの剪定シミュレータを使用すると、さまざまなバックアップスケジュールでさまざまな保持オプションの効果を調べることができます。

## 16.6.2 保持設定例

バックアップの頻度や古いバックアップの保持は、データの変更頻度や、特定の作業負荷における古い状態の重要性によって異なります。バックアップが企業の文書アーカイブとして機能する場合、バックアップの保存期間に関する法的要件がある場合もあります。

この例では、バックアップを毎日行い、保存期間を10年とし、保存されるバックアップの間隔は徐々に長くなっていくと仮定します。

keep-last=3 - たとえ毎日バックアップを取るとしても、管理者は大きなアップグレードの直前や直後に追加のバックアップを取りたいと思うかもしれません。keep-lastを設定することで、これが確実になります。

---

す。

`keep-hourly` が設定されていない - 毎日のバックアップには関係ありません。`keep-last`を使用することで、手動バックアップを増やすことができます。

`keep-daily=13` - 少なくとも1日をカバーする`keep-last`と合わせて、少なくとも2週間分のバックアップを確保します。

`keep-weekly=8` - 少なくとも2ヶ月分の週次バックアップを確保します。

`keep-monthly=11` - 前回の`keep`設定と合わせて、少なくとも1年分の月次バックアップを確保します。

`keep-yearly=9` - これは長期アーカイブ用である。前のオプションで現在の年をカバーしたので、残りの年についてはこれを9に設定し、合計で少なくとも10年間をカバーすることになります。

保存期間が不必要に長いと感じたら、いつでも減らすことができますが、一度削除されたバックアップを作成することはできません。

## 16.7 バックアップ保護

バックアップを保護マークして削除できないようにすることができます。Proxmox VE の UI、CLI、または API を使用して保護されたバックアップを削除しようとすると、失敗します。ただし、これは Proxmox VE によって強制されるものであり、ファイルシステムによって強制されるものではありません。つまり、バックアップファイル自体を手動で削除することは、基礎となるバックアップストレージへの書き込みアクセス権を持っている人であれば誰でも可能です。

### 注

保護されたバックアップはプルーニングによって無視され、保持設定にカウントされません。

ファイルシステムベースのストレージの場合、保護はセンチネルファイル`<backup-name>.protected`を介して実装されます。Proxmox Backup Server の場合は、サーバ側で処理されます（Proxmox Backup Server バージョン 2.1）。

ストレージオプションの`max-protected-backups`を使用して、ゲストごとにストレージで許可される保護バックアップの数を制御します。無制限には`-1`を使用します。デフォルトは、`Datastore.Allocate` 権限を持つユーザは無制限、その他のユーザは5です。

## 16.8 バックアップノート

UI の [Edit Notes (ノートの編集)] ボタンを使用するか、ストレージコンテンツAPIを介してバックアップにノートを追加できます。



バックアップ・ジョブや手動バックアップのノートを動的に生成するためのテンプレートを指定することも

可能です。テンプレート文字列には、2つの中括弧で囲まれた変数を含めることができます、バックアップ実行時に対応する値に置き換えられます。

現在サポートされているのは

- {cluster} } クラスタ名（もしあれば）
- {ゲスト名} } 仮想ゲストに割り当てられた名前
- {バックアップが作成されるノードのホスト名。}
- {{vmid}} ゲストの数値 VMID

APIやCLIで指定する場合は、1行にする必要があり、改行とバックスラッシュはそれぞれリテラル //としてエスケープする必要がある。

## 16.9 リストア

バックアップアーカイブは、Proxmox VEのWeb GUIまたは以下のCLIツールでリストアできます：

### pct リストア

コンテナ復元ユーティリティ

### qmrestore

仮想マシン復元ユーティリティ

詳細については、対応するマニュアルのページを参照してください。

### 16.9.1 帯域幅の制限

1つまたは複数の大きなバックアップをリストアする場合、多くのリソース、特にバックアップ・ストレージからの読み込みとターゲット・ストレージへの書き込みの両方にストレージ帯域幅が必要になることがあります。ストレージへのアクセスが混雑するため、他の仮想ゲストに悪影響を及ぼす可能性があります。

これを回避するには、バックアップジョブに帯域幅制限を設定します。Proxmox VEはリストアとアーカイブの2種類の制限を実装しています：

- リストアごとの制限：バックアップアーカイブからの読み出しに使用する帯域幅の最大値を示します。
- ストレージごとの書き込み制限：特定のストレージへの書き込みに使用される最大帯域幅を示す。

読み取り制限は書き込み制限に間接的に影響する。ジョブごとの上限が小さいと、ストレージごとの上限が大きくても上書きされます。より大きなジョブごとの制限は、影響を受けるストレージに'Data.Allocate'パミッシュがある場合にのみ、ストレージごとの制限を上書きします。

リストアCLIコマンドで '--bwlimit <integer>' オプションを使用すると、リストアジョブ固有の帯域幅制限を設定できます。KiB/sが制限の単位として使用されます。つまり、「10240」を渡すと、バックアップの読み取り速度が10MiB/sに制限され、実行中の仮想ゲストが残りのストレージ帯域幅を使用できるようになります。バックアップが仮想ゲストの操作に影響を与えないようになります。

---

### 注

---

`bwlimit` パラメーターに '0' を使用すると、特定のリストアジョブのすべての制限を無効にできます。これは非常に重要な仮想ゲストを可能な限り速くリストアする必要がある場合に役立ちます。(ストレージの 'Data.Allocate' パーミッションが必要です。)

---

ほとんどの場合、ストレージの一般的に利用可能な帯域幅は、時間が経過しても同じままです。そのため、設定し

```
# pvesm set STORAGEID --bwlimit restore=KIBs
```

たストレージごとにデフォルトの帯域幅制限を設定できるようにしました：

## 16.9.2 ライブリストア

大きなバックアップをリストアすると、ゲストが利用できないまま長い時間がかかることがあります。

Proxmox Backup Server上に保存されたVMバックアップの場合、ライブリストアオプションを使用することで、この待ち時間を短縮することができます。

GUIのチェックボックスまたはqmrestoreの-live-restore引数のいずれかを使用してライブ・リストアを有効にすると、リストアが開始されるとすぐにVMが起動します。データはバックグラウンドでコピーされ、VMがアクティブにアクセスしているチャンクが優先されます。

なお、これには2つの注意点がある：

- ライブ・リストア中は、データをバックアップ・サーバーからロードする必要があるため、VMは制限されたディスク・リード速度で動作することになる（ただし、一度ロードされれば、宛先ストレージですぐに利用できるため、データに2回アクセスしてもペナルティが発生するのは最初の1回のみ）。書き込み速度はほとんど影響を受けない。
- つまり、すべてのデータがバックアップからコピーされていない可能性があり、失敗したリストア操作中に書き込まれたデータを保持できない可能性が高い。

この動作モードは、ウェブ・サーバーなど、初期動作に必要なデータ量が少ない大規模なVMに特に有効で、OSと必要なサービスが起動すればVMは稼働し、バックグラウンド・タスクはあまり使用されないデータをコピーし続ける。

## 16.9.3 単一ファイルの復元

ストレージGUIのバックアップタブにあるファイルリストアボタンを使用すると、バックアップに含まれるデータ上で直接ファイルブラウザを開くことができます。この機能は Proxmox Backup Server 上のバックアップでのみ使用できます。

コンテナの場合、ファイルツリーの最初のレイヤには、含まれるすべてのpxarアーカイブが表示され、自由に開いたり参照したりできる。VMの場合、最初のレイヤには含まれるドライブイメージが表示され、これを開くと、ドライブ上でサポートされているストレージ技術のリストが表示される。最も基本的なケースでは、これはパーティションテーブルを表すpartと呼ばれるエントリで、ドライブにある各パーティションのエントリを含んでいます。VMの場合、すべてのデータにアクセスできるわけではないことに注意してください

(サポートされていないゲストファイルシステム、ストレージテクノロジーなど)。

ファイルやディレクトリは*Download*ボタンでダウンロードでき、後者はその場でzipアーカイブに圧縮される。

信頼できないデータを含む可能性のあるVMイメージへの安全なアクセスを可能にするため、一時的なVM（ゲストとして見えない）が起動される。これは、このようなアーカイブからダウンロードされたデータが本質的に安全であることを意味するものではないが、ハイパーバイザー・システムを危険にさらすことを避けることができる。VMはタイムアウト後に停止する。このプロセス全体は、ユーザーから見て透過的に行われる。

---

### 注

について                    トラブルシューティング                    を目的としています、                    各 一時的な VM  
インスタンス    は                    a                    ログ                    ファイル                    を生成する。  
/var/log/proxmox-backup/file-restore/。ログファイルには、バックアップアーカイブに含まれる個々のファイルのリストアやファイルシステムへのアクセスが失敗した場合の追加情報が含まれている可能性があります。

---

## 16.10 構成

グローバル設定は/etc/vzdump.confに保存される。このファイルでは、コロンで区切られた単純なキー/値形式を使用します。各行の書式は以下の通りである：

オプション： 値

ファイル内の空白行は無視され、#文字で始まる行はコメントとして扱われ、これも無視される。このファイルの値はデフォルトとして使用され、コマンドラインで上書きすることができる。

現在、以下のオプションをサポートしています：  
す：

**bwlimit: <整数> (0 - N) (デフォルト = 0)**

I/O バンド幅を制限する（単位：KiB/s）。

**compress: <0 | 1 | gzip | lzo | zstd> (デフォルト = 0)**

ダンプファイルを圧縮する。

**dumpdir: <文字列>**

結果のファイルを指定したディレクトリに保存する。

**exclude-path: <配列>**

特定のファイル/ディレクトリ（シェル・グロブ）を除外する。で始まるパスはコンテナのルートに固定され、その他のパスは各サブディレクトリに相対的に一致する。

**fleecing: [[enabled=<1|0>], ,ストレージ=<ストレージID>]]。**

バックアップフリートのオプション（VMのみ）。

**enabled=<boolean> (デフォルト = 0)**

バックアップフリーを有効にする。新しいゲストの書き込みがバックアップターゲットに直接コピーする代わりに、指定されたストレージで発生したブロックからバックアップデータをキャッシュします。これにより、ゲストのI/Oパフォーマンスを向上させ、ハングを防止することができます。

## ストレージ=<ストレージID>

このストレージは、ストレージフリーキングイメージに使用する。スペースを効率的に使うには、廃棄をサポートし、シンプロビジョニングかスパースファイルをサポートするローカルストレージを使うのがベストだ。

### **ionice: <整数> (0 - 8) (デフォルト = 7)**

BFQスケジューラ使用時にIOプライオリティを設定する。VMのスナップショットとサスPENDモードのバックアップでは、これはコンプレッサにのみ影響します。値が8の場合はアイドルプライオリティが使用され、それ以外の場合は指定した値でベストエフォートプライオリティが使用されます。

### **lockwait: <整数> (0 - N) (デフォルト = 180)**

グローバルロックの最大待機時間（分）。

### **mailnotification: <always | failure> (デフォルト = always)**

非推奨: 代わりに通知ターゲット/マッチャーを使用してください。通知メールを送るタイミングを指定する

**mailto: <文字列>**

非推奨: 代わりに notification targets/matchers を使用してください。メール通知を受け取るメールアドレスまたはユーザのカンマ区切りリスト。

**maxfiles: <整数> (1 - N)**

非推奨: 代わりに *prune-backups* を使用してください。ゲストシステムごとのバックアップファイルの最大数。

**mode: <スナップショット | 停止 | サスPEND> (デフォルト = スナップショット)**

バックアップモード。

**notes-template: <文字列>**

バックアップのメモを生成するためのテンプレート文字列。その値で置き換えられる変数を含むことができる。現在サポートされているのは {{cluster}}, {{guestname}}, {{node}}, {{vmid}} であるが、今後追加されるかもしれない。改行とバックスラッシュは、それぞれ ``n と ``n としてエスケープする必要がある。

---

**注**

必要なオプション: ストレージ

---

**notification-mode: <auto | legacy-sendmail | notification-system>.****(デフォルト = 自動)**

使用する通知システムを決定する。*legacy-sendmail* に設定すると、vzdump は mailto/mailnotification パラメータを考慮し、*sendmail* コマンドで指定したアドレスにメールを送信します。*notification-system* に設定すると、PVE の通知システム経由で通知が送信され、mailto と mailnotification は無視されます。*auto* (デフォルト設定) に設定すると、mailto が設定されていればメールが送信され、設定されていなければ通知システムが使用されます。

**notification-policy: <always | failure | never> (デフォルト = always)**

非推奨: を使用しないでください。

**通知先: <文字列>**

非推奨: を使用しないでください。

---

**パフォーマンスを向上させます: [max-workers=<integer>] [,pbs-entries-max=<integer>]。**

その他のパフォーマンスに関する設定。

**max-workers=<integer> (1 - 256) (デフォルト = 16)**

VMに適用される。同時に最大これだけのIOワーカーを許可する。

**pbs-entries-max=<integer> (1 - N) (デフォルト = 1048576)**

PBS に送信されるコンテナバックアップに適用される。意図しない OOM 状況を回避するために、ある時点でメモリ上に許可されるエントリ数を制限する。これを増やすと、大量のファイルを含むコンテナのバックアップが可能になります。

**pigz: <整数> (デフォルト = 0)**

N>0の場合はgzipの代わりにpigzを使う。N=1 はコアの半分を使い、N>1 は N をスレッド数として使う。

**pool: <文字列>**

指定したプールに含まれるすべての既知のゲストシステムをバックアップする。

**protected: <ブール値>**

trueの場合、バックアップを保護マークします。

---

**注**

必要なオプション: ストレージ

---

**バックアップを削除する:** [`keep-all=<1|0>`] [`,keep-daily=<N>`] [`,keep-hourly=<N>`] [`,keep-last=<N>`] [`,keep-monthly=<N>`] [`,keep-weekly=<N>`] .  
[`,keep-yearly=<N>`] (デフォルト = `keep-all=1`)

これらの保持オプションは、ストレージ構成のオプションの代わりに使用する。

**keep-all=<ブール値>**

すべてのバックアップを保持する。trueの場合、他のオプションと競合します。

**キープデイリー=<N>**

過去<N>日のバックアップを保持します。1日に複数のバックアップがある場合は、最新のものだけが保持されます。

**キープ・アワー=<N>**

直近の<N>異なる時間のバックアップを保持します。1時間に複数のバックアップがある場合は、最新のものだけが保持されます。

**keep-last=<N>**

最後の<N>個のバックアップを保持する。

**keep-monthly=<N>**

過去<N>の異なる月のバックアップを保持します。1つの月に複数のバックアップがある場合は、最新のものだけが保持されます。

**keep-weekly=<N>**

過去<N>週間分のバックアップを保持します。1週間に複数のバックアップがある場合は、最新のものだけが保持されます。

---

**keep-yearly=<N> (毎年)**

過去<N>年分のバックアップを保持する。一つの年に複数のバックアップがある場合、最新のものだけが保存されます。

**remove: <論理値> (デフォルト = 1)**

*prune-backups*に従って古いバックアップを削除する。

**スクリプト: <文字列>**

指定されたフックスクリプトを使用する。

**stdexcludes: <論理値> (デフォルト = 1)**

一時ファイルとログを除く。

**stopwait: <整数> (0 - N) (デフォルト = 10)**

ゲストシステムが停止するまでの最大待機時間（分）。

**ストレージ: <ストレージID>**

出来上がったファイルをこのストレージに保存する。

**tmpdir: <文字列>**

指定したディレクトリに一時ファイルを保存する。

**zstd: <整数> (デフォルト = 1)**

Zstdのスレッド数。N=0は利用可能なコアの半分を使用し、Nが0より大きな値に設定された場合、Nはスレッド数として使用される。

#### vzdump.confの設定例

```
tmpdir: /mnt/fast_local_disk
storage: my_backup_storage
mode: snapshot
bwlimit: 10000
```

## 16.11 フックスクリプト

オプション --script でフック・スクリプトを指定できます。このスクリプトはバックアップ処理の様々な段階で呼び出され、パラメータが適宜設定されます。ドキュメンテーションディレクトリに例があります(vzdump-hook-script.pl)。

## 16.12 ファイルの除外

---

### 注

このオプションはコンテナ・バックアップにのみ使用できる。

---

vzdumpはデフォルトで以下のファイルをスキップします (--stdexcludes 0オプションで無効にできます)

---

)。

```
/tmp/?*  
/var/tmp/?*  
/* pid
```

手動で（追加の）除外パスを指定することもできる：

```
# vzdump 777 --exclude-path /tmp/ --exclude-path '/var/foo*''
```

は、/tmp/ディレクトリと、/var/foo、/var/foobarといった名前のファイルやディレクトリを除外する。

で始まらないパスは、コンテナのルートにアンカーされない。たとえば

```
# vzdump 777 --exclude-path bar
```

は、/bar、/var/bar、/var/foo/barなどのファイルやディレクトリを除外するが、/bar2は除外しない。設定ファイルもバックアップアーカイブ内（./etc/vzdump/内）に保存され、正しくリストアされます。

## 16.13 例

単にゲスト777をダンプします - スナップショットはありません。ゲストのプライベート領域と設定ファイルをデフォルトのダンプディレクトリ(通常は/var/lib/vz/dump/)にアーカイブするだけです。

```
# vzdump 777
```

rsyncとサスPEND/レジュームを使ってスナップショットを作成する（ダントンタイムは最小限）。

```
# vzdump 777 --mode suspend
```

すべてのゲストシステムをバックアップし、rootとadminに通知メールを送信します。mailtoが設定され、notification-modeがデフォルトでautoに設定されているため、通知メールは通知システムではなく、システムのsendmailコマンドで送信されます。

```
# vzdump --all --mode suspend --mailto root --mailto admin
```

スナップショットモード（ダントンタイムなし）とデフォルト以外のダンプディレクトリを使用する。

```
# vzdump 777 --dumpdir /mnt/backup --mode snapshot
```

複数のゲストを（選択的に）バックアップする

```
# vzdump 101 102 103 --mailto root
```

101と102を除く全ゲストをバックアップ

```
# vzdump --mode suspend --exclude 101,102
```

コンテナを新しいCT 600にリストアする

```
# pct restore 600 /mnt/backup/vzdump-lxc-777.tar
```

QemuServer VMをVM 601にリストアする

```
# qmrestore /mnt/backup/vzdump-qemu-888.vma 601
```

既存のコンテナ101を、パイプを使用して、4GBのルートファイルシステムを持つ新しいコンテナ300にクローンする。

```
# vzdump 101 --stdout | pct restore --rootfs 4 300 -.
```

# 第17章 通知

## 17.1 概要

Enable	Target Name ↑	Type	Comment	Origin
✓	mail-to-root	sendmail	Send mails to root@pam's email address	Built-In

Enable	Matcher Name ↑	Comment	Origin
✓	default-matcher	Route all notifications to mail-to-root	Built-In
✓	example-matcher		Custom

Proxmox VEは、システム内で注目すべきイベントが発生した場合、通知を送信します。

さまざまな[通知イベント](#)があり、それぞれが通知マッチャーで使用できるメタデータフィールドのセットを持っている。

通知マッチャーは、どの通知をどこに送るかを決定します。マッチャーは、特定の通知プロパティ(例えば、タ

イムスタンプ、重大度、メタデータフィールド)でマッチするために使用できるマッチルールを持っています。  
もし

matcher が通知とマッチすると、その通知は、設定可能な通知ターゲットのセットにルーティングされます。

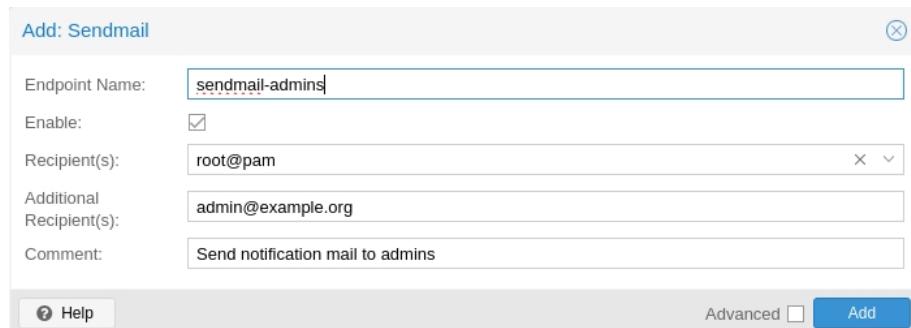
通知ターゲットは、通知が送られるべきデスティネーションの抽象化です-例えば、Gotifyサーバーインスタンス、または電子メールアドレスのセットです。通知ターゲットには、システムのsendmailコマンドを使って電子メールを送るsendmailや、Gotifyインスタンスに通知を送るgotifyなど、複数のタイプがあります。

通知システムは、GUI の [データセンター] → [通知] で構成できます。構成は、/etc/pve/notifications.cfg および /etc/pve/priv/notifications.cfg に保存されます。

をターゲットにしている。

## 17.2 通知対象

### 17.2.1 センドメール



sendmailバイナリは、電子メールメッセージの送信を処理する、Unixライクなオペレーティングシステムで一般的に見られるプログラムです。コマンドラインユーティリティであり、ユーザやアプリケーションがコマンドラインやスクリプト内から直接電子メールを送信することができます。

sendmail通知ターゲットは、sendmailバイナリを使用して電子メールを送信する。

#### 注

Proxmox VEの標準インストールでは、sendmailバイナリはPostfixによって提供されます。このタイプのターゲットが正しく動作するためには、Postfixの設定を変更して、メールを正しく配信できるようにする必要があるかもしれません。クラスタ・セットアップの場合、クラスタ・ノードごとにPostfixの設定が必要です。

センドメールターゲットプラグインの設定には以下のオプションがあります：

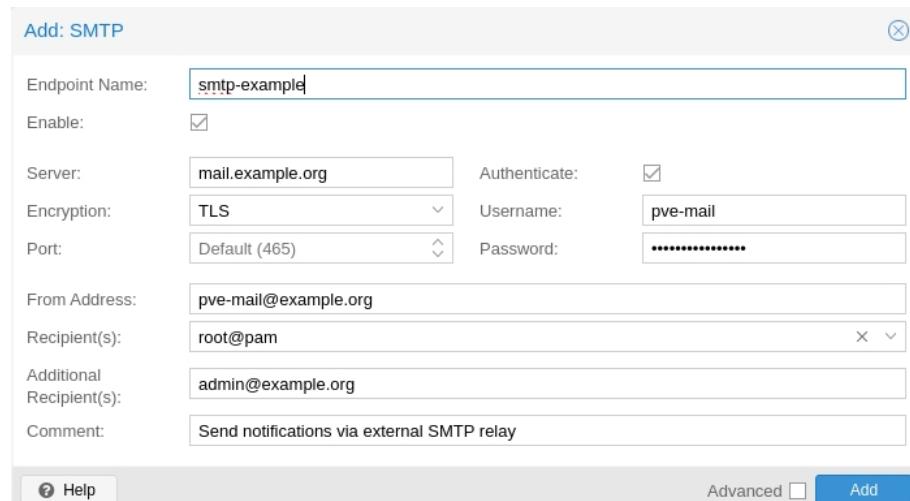
- `mailto`: 通知を送信するメールアドレス。複数の受信者に対応するために複数回設定することができます
  -
- `mailto-user`: メールを送信するユーザー。ユーザーのメールアドレスは `users.cfg`。複数の受信者に対応するため、複数回設定することができます。
- `author`: メールの作成者を設定します。デフォルトはProxmox VEです。
- `from-address`: メールの送信元アドレスを設定します。このパラメータが設定されていない場合、プラグインは`datacenter.cfg`の`email_from`設定にフォールバックします。このパラメータも設定されていない場合、プラグインのデフォルトは`root@$hostname` (`$hostname`はノードのホスト名) です。

- コメントこのターゲットに対するコメント メールのFromヘッダは\$author <\$from-addressに設定される。

設定例 (/etc/pve/notifications.cfg):

```
sendmail: 例
  mailto-user root@pam
  mailto-user admin@pve
  mailto
  max@example.com
  フォームアドレス pve1@example.com
  コメント 複数のユーザー/アドレスに送信する
```

## 17.2.2 SMTP



SMTP通知ターゲットはSMTPメールリレーに直接メールを送信することができます。

SMTPターゲットプラグインの設定には以下のオプションがあります。

ます:

- mailto: 通知を送信するメールアドレス。複数の受信者に対応するために複数回設定することができます
  - mailto-user: メールの送信先ユーザー。ユーザーのメールアドレスはusers.cfg。複数の受信者に対応するため、複数回設定することができます。
- author: メールの作成者を設定します。デフォルトはProxmox VEです。
- From-address: メールのFromアドレスを設定します。SMTPリレーでは、なりすましを避けるため

に、このアドレスがユーザのものであることを要求する場合があります。メールのFromヘッダは\$authorに設定されます。  
<\$from-address>.

- username: 認証時に使用するユーザー名。ユーザー名が設定されていない場合、認証は行われません。PLAIN および LOGIN 認証方式をサポートしています。
- password: 認証時に使用するパスワード。
- モード: 暗号化モード (insecure、starttls、tls) を設定する。デフォルトはtls。
- サーバーを指定します: SMTPリレーのアドレス/IP

- ポート: 接続先ポート。設定されていない場合、使用されるポートのデフォルトは、mode の値に応じて 25(安全でない)、465(tls)、または 587(starttls)になります。
- コメントこのターゲットに対するコメント

smtp: 例

```
mailto-user root@pam
mailto-user admin@pve
mailto
max@example.com

From: pve1@example.com
User: pve1
Server:
mail.example.com モード
starttls
```

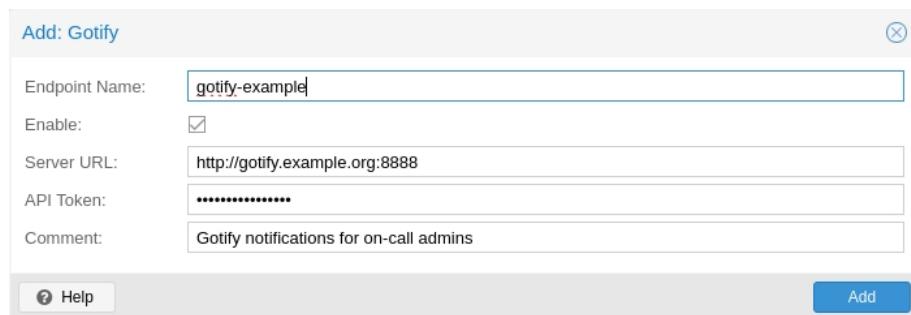
設定例(/etc/pve/notifications.cfg):

etc/pve/priv/notifications.cfgの一致するエントリで、シークレットトークンが含まれています:

smtp: 例

```
password
```

### 17.2.3 ゴティファイ



Gotifyは、様々なデバイスやアプリケーションにプッシュ通知を送受信できるオープンソースのセルフホスト型通知サーバーです。シンプルなAPIとウェブインターフェースを提供し、様々なプラットフォームやサービスと簡単に統合することができます。

Gotifyターゲットプラグインの設定には以下のオプションがあります:

- サーバーを指定します: GotifyサーバーのベースURL、例: http://<ip>:8888

- ・ トークン：認証トークン。トークンはGotifyのWebインターフェイス内で生成することができます。
- ・ コメントこのターゲットに対するコメント

---

#### 注

Gotify ターゲットプラグインは、[データセンター設定からの HTTP プロキシ設定](#)を尊重します。

---

#### 例

```
サーバー http://gotify.example.com:8888 コメ  
ント 複数のユーザー/アドレスに送信する
```

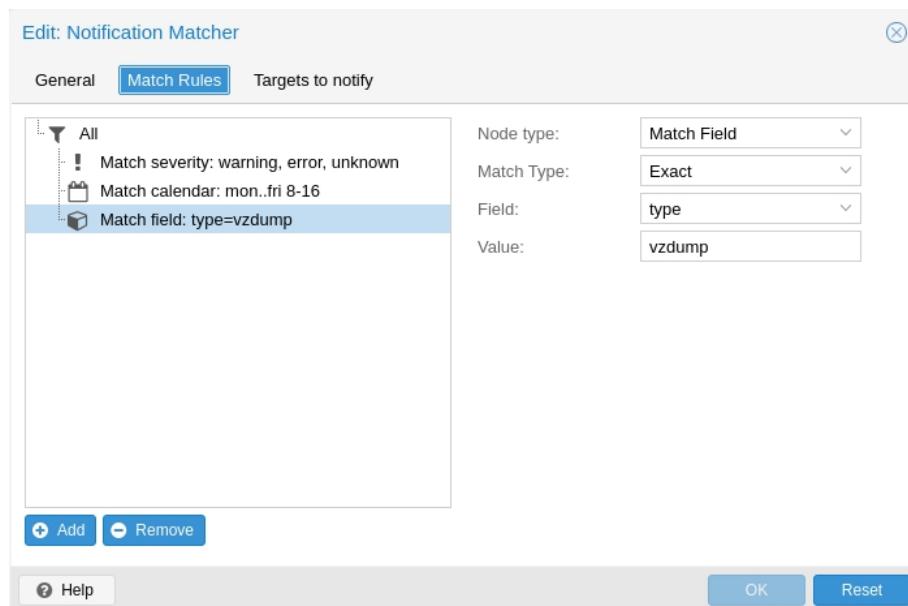
設定例 (/etc/pve/notifications.cfg)：

etc/pve/priv/notifications.cfgの一致するエントリで、シークレットトークンを含む:

例

トークン

## 17.3 通知マッチャー



通知マッチャーは、マッチングルールに基づいて通知を通知ターゲットにルーティングします。これらのルールは、タイムスタンプ (match-calendar)、通知の重大度 (match-severity)、メタデータフィールド (match-field) など、通知の特定のプロパティにマッチすることができます。通知がマッチャーによってマッチされると、マッチャー用に構成されたすべてのターゲットが通知を受信します。

任意の数のマッチャーを作成することができ、それが独自のマッチングルールと通知するターゲットを持つ。ターゲットが複数のマッチャーで使用されている場合でも、各ターゲットは通知のたびに最大1回通知されます。

マッチング・ルールのないマッチャーは常に真であり、設定されたターゲットは常に通知される。

matcher: 常にマッチする

ターゲット管理者

コメント このマッチャーは常に

### 17.3.1 マッチャーのオプション

- ・ ターゲット： マッチャーがマッチした場合、どのターゲットに通知するかを決定します。複数のターゲットに通知するために複数回使用することができます。
- ・ invert-match: マッチャー全体の結果を反転する。
- ・ モード： マッチャー全体の結果を計算するために、個々のマッチ・ルールがどのように評価されるかを決定します。 `all` に設定すると、すべてのマッチ・ルールがマッチしなければならない。`any` に設定すると、少なくとも 1 つのルールがマッチする必要があります。デフォルトは `all` です。
- ・ match-calendar: 通知のタイムスタンプをスケジュールと照合する。

- マッチフィールド: 通知のメタデータ・フィールドにマッチする
- match-severity: 通知の重大度に一致させる
- コメントこのマッチャーのコメント

### 17.3.2 カレンダー・マッチング・ルール

カレンダーマッチャーは、設定可能なスケジュールに対して通知が送信される時刻をマッチングします。

- マッチカレンダー 8-12
- 試合カレンダー 8:00-15:30
- 試合カレンダー 月～金 9:00～17:00
- 試合カレンダー 日・火・水・金 9-17

### 17.3.3 フィールド・マッチング・ルール

通知には、一致させることができるメタデータ・フィールドの選択がある。

- match-field exact:type=vzdump バックアップに関する通知にのみマッチします。
- ノードのホスト名にマッチする。

マッチしたメタデータフィールドが存在しない場合、その通知はマッチしません。例えば、match-field regex:hostname=.\* ディレクティブは、任意のホスト名メタデータフィールドを持つ通知だけにマッチしますが、フィールドが存在しない場合はマッチしません。

### 17.3.4 深刻度マッチングルール

通知には、一致させることができると関連する重大度があります。

- マッチ・セバリティ・エラー: マッチエラーのみ
- match-severity 警告、エラー: マッチの警告とエラー

次の深刻度が使用されている: info、notice、warning、error、unknown。

### 17.3.5 例

matcher: 平日

マッチカレンダー 月～金 9-17 目標

管理者

コメント 就業時間内に管理者に通知

マッチャー: 夜と週末

マッチカレンダー 月-金 9-17 反

転マッチ true

オンコール管理者対象

コメント 非労働時間については別途目標を設定

```

matcher: バックアップの失敗
  match-field exact:type=vzdump
    match-severity エラー
    バックアップ管理者
    コメント バックアップ失敗の通知を1つのグループに送る←'
      管理者

matcher: クラスタの失敗
  match-field exact:type=レプリケーシ
    ョン match-field exact:type=フェン
    シング・モード any
    対象のクラスタ管理者
    コメント クラスタ関連の通知を他の管理者グループに送る

```

最後のマッチャーは、正規表現によるフィールドマッチャーを使って書き換えることもできる：

```

matcher: クラスタ失敗
  match-field regex:type=^(replication|fencing)$
  target cluster-admins
  コメント クラスタ関連の通知を他の管理者グループに送る

```

## 17.4 通知イベント

イベント	タイプ	重大性	メタデータ・フィールド タイプの追加)
システム・アップデート 利用可能	パッケージ更新	インフォメーション	ホスト名
クラスターノード	フェンシング	エラー	ホスト名
ストレージ・レプリケー ション 失敗した	レプリケーション	エラー	-
バックアップ終了	ブイエスタンプ	情報 (失敗した場合は エラー)	ホスト名
ルートへのメール	システムメール	不明	-

フィールド名	説明
タイプ	通知の種類
ホスト名	ドメインを含むホスト名 (例 pvel.example.com)

## 17.5 システム・メール転送

smartdなどの特定のローカル・システム・デーモンは、最初にローカル・ルート・ユーザーに向けられた通知メールを生成します。Proxmox VEは、これらのメールを、タイプ*system-mail*、深刻度*unknown*の通知として通知システムに送ります。

転送処理が電子メールベースのターゲット（sendmailやsmtpなど）を含む場合、電子メールは受信したとおりに転送され、元のメールヘッダーはすべてそのまま残ります。それ以外のターゲットの場合、システムはメールのコンテンツから件名と本文の両方を抽出しようとします。電子メールがHTMLコンテンツのみで構成されている場合、このプロセスでプレーンテキスト形式に変換される。

## 17.6 アクセス許可

通知ターゲットのコンフィギュレーションを変更/表示するには、`Mapping.Modify`/`Mapping.Audit` パーミッションが必要です。

ターゲットをテストするには、`/mapping` の `Mapping.Use`、`Mapping.Audit`、または `Mapping.Modify` のいずれかのパーミッションが必要です。

## 第18章

# 重要なサービス・デーモン

### 18.1 pvedaemon - Proxmox VE API デーモン

このデーモンは、Proxmox VE API全体を127.0.0.1:85に公開します。このデーモンは `root` として実行され、すべての特権操作を実行する権限を持っています。

---

#### 注

デーモンはローカルアドレスのみをリッスンするので、外部からアクセスすることはできません。

`pveproxy`

デーモンはAPIを外部に公開する。

---

### 18.2 pveproxy - Proxmox VE API プロキシデーモン

このデーモンは、HTTPSを使用してTCPポート8006でProxmox VE API全体を公開します。ユーザー`www-data`として実行されます。

であり、非常に限定されたパーミッションを持つ。より多くのパーミッションを必要とする操作は、ローカルの`pvedaemon`に転送されます。

他のノードをターゲットにしたリクエストは、自動的にそれらのノードに転送されます。つまり、1つのProxmox VEノードに接続するだけで、クラスタ全体を管理できます。

---

#### 18.2.1 ホストベースのアクセス制御

apache2」ライクなアクセス制御リストを設定することができる。値は /etc/default/pvepro ファイルから読み込まれます。

例えば、こうだ：

```
ALLOW_FROM="10.0.0.1-10.0.0.5,192.168.0.0/22"
DENY_FROM="all"
POLICY="許可"
```

IPアドレスは、Net::IPが理解できるあらゆる構文を使って指定できる。allという名前は  
0/0と::/0（すべてのIPv4とIPv6アドレスを意味する）

。デフォルトのポリシーは許可です。

試合	POLICY=拒否	POLICY=許可
マッチのみ許可	認める	認める
マッチ拒否のみ	拒否する	拒否する
該当なし	拒否する	認める
許可と拒否の両方に対応	拒否する	認める

## 18.2.2 リスニングIPアドレス

デフォルトでは、pveproxyとspiceproxyデーモンはワイルドカード・アドレスをリッスンし、IPv4とIPv6の両方のクライアントからの接続を受け付ける。

etc/default/pveproxy で LISTEN\_IP を設定することで、pveproxy がどの IP アドレスに接続されるかを制御できます。

と spiceproxy デーモンがバインドする。IPアドレスはシステム上で設定する必要がある。

sysctl net.ipv6.bindv6onlyをデフォルトでない1に設定すると、デーモンはIPv6クライアントからの接続しか受け付けなくなるが、通常は他の多くの問題も引き起こす。この設定を行った場合は、sysctl設定を削除するか、LISTEN\_IPを0.0.0.0（IPv4クライアントのみを許可する）に設定することを推奨する。

LISTEN\_IPは、例えば、ソケットを内部インターフェイスに限定し、公衆インターネットへの露出を少なくするためだけに使うことができる：

```
LISTEN_IP="192.0.2.1"
```

同様に、IPv6アドレスを設定することもできる：

```
LISTEN_IP="2001:db8:85a3::1"
```

リンクローカルIPv6アドレスを指定したい場合は、インターフェース名そのものを指定する必要があること

```
LISTEN_IP="fe80::c463:8cff:feb9:6a4e%vmbr0"
```

に注意。例えば

### 警告

クラスタ内のノードは通信のために pveproxy にアクセスする必要があります。クラスタ化されたシステムで LISTEN\_IP を設定することはお勧めしません。

変更を適用するには、ノードを再起動するか、pveproxyとspiceproxyを完全に再起動する必要があります。

サービスを提供する:

```
systemctl restart pveproxy.service spiceproxy.service
```

---

### 注

`reload` とは異なり、`pveproxy` サービスの再起動は、仮想ゲストから実行中のコンソールやシェルなど、いくつかの長時間実行中のワーカープロセスを中断する可能性があります。そのため、この変更を有効にするにはメンテナンスウィンドウを使用してください。

---

### 18.2.3 SSL暗号スイート

`etc/default/pveproxy` で CIPHERS (TLS <= 1.2) と CIPHERSUITS (TLS >= 1.3) キー。例えば

```
CIPHERS="ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:←
ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE- ←
ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA- ←
AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES128-SHA256:←
ECDHE-RSA-AES128- ←
SHA256" CIPHERSUITS="TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:←
TLS_AES_128_GCM_SHA256"
```

上記はデフォルトである。利用可能なオプションの一覧は、`openssl` パッケージの `ciphers(1)` man ページを参照のこと。

さらに、クライアントが`/etc/default/pveproxy` で使用する暗号を選択するように設定することもできます（デフォルトは、クライアントと`pveproxy` の両方で使用可能なリストの最初の暗号です）：

```
honor_cipher_order=0
```

### 18.2.4 対応TLSバージョン

安全でない SSL バージョン 2 と 3 は `pveproxy` では無条件に無効化されます。最近の OpenSSL バージョンでは、1.1 未満の TLS バージョンはデフォルトで無効化され、`pveproxy` はこれに従います（`/etc/ssl/openssl.cnf` を参照）。

TLS バージョン 1.2 または 1.3 を無効にするには、`/etc/default/pveproxy` で以下を設定する：

```
disable_tls_1_2=1
```

あるいは、それぞれ

```
disable_tls_1_3=1
```

---

#### 注

特別な理由がない限り、サポートされる TLS バージョンを手動で調整することは推奨されません。

---

### 18.2.5 ディフィー・ヘルマン・パラメーター

etc/default/pveproxyでDHPARAMSを設定することで、使用するDiffie-Hellmanパラメータを定義できます。

をPEM形式のDHパラメータを含むファイルのパスに変換する。

```
DHPARAMS="/path/to/dhparams.pem"
```

このオプションが設定されていない場合、組み込みの skip2048 パラメータが使用される。

---

## 注

DHパラメータは、DH鍵交換アルゴリズムを利用する暗号スイートがネゴシエートされる場合にのみ使用される

。

---

## 18.2.6 代替HTTPS証明書

使用する証明書は、外部証明書または ACME 経由で取得した証明書に変更できます。

pveproxyは/etc/pve/local/pveproxy-ssl.pemと/etc/pve/local/pveproxy-ssl.keyを使用します。

が存在する場合は、/etc/pve/local/pve-ssl.pem と /etc/pve/local/pve-ssl.key にフルバックします。秘密鍵はパスフレーズを使用することはできません。

証明書の秘密鍵/etc/pve/local/pveproxy-ssl.keyの場所を上書きすることも可能です。

例えば、/etc/default/pveproxyでTLS\_KEY\_FILEを設定する：

```
TLS_KEY_FILE="/secrets/pveproxy.key"
```

---

### 注

付属のACMEインテグレーションはこの設定に対応していません。

---

詳細については、マニュアルの「ホストシステム管理」の章を参照してください。

## 18.2.7 レスポンス・コンプレッション

デフォルトでは、pveproxyはクライアントがサポートしていれば、圧縮可能なコンテンツに対してgzip HTTPレベル圧縮を使用します。これは /etc/default/pveproxy で無効にできます。

```
COMPRESSION=0
```

## 18.3 pvestatd - Proxmox VE ステータステーモン

このデーモンはVM、ストレージ、コンテナのステータスを定期的にクエリする。結果はクラスタ内のすべてのノードに送信される。

## 18.4 spiceproxy - SPICE プロキシサービス

SPICE (Simple Protocol for Independent Computing Environments) はオープンなリモート・コンピューテ

ィング・ソリューションで、リモート・ディスプレイやデバイス（キーボード、マウス、オーディオなど）へのクライアント・アクセスを提供する。主なユースケースは、仮想マシンやコンテナへのリモートアクセスです。

このデーモンは TCP ポート 3128 をリッスンし、SPICE クライアントからの CONNECT リクエストを正しい Proxmox VE VM に転送する HTTP プロキシを実装しています。このデーモンはユーザ www-data として実行され、非常に制限された権限を持っています。

#### 18.4.1 ホストベースのアクセス制御

apache2 "をアクセス制御リストのように設定することができる。値は /etc/default/pveproxy ファイルから読み込まれます。

詳細は pveproxy のドキュメントを参照。

## 18.5 pvescheduler - Proxmox VE スケジューラーモン

この deamon は、レプリケーションや vzdump ジョブなど、スケジュールに従ってジョブを開始します。

vzdumpジョブについては、/etc/pve/jobs.cfgファイルから設定を取得します。

# 第19章

## 便利なコマンドラインツール

### 19.1 pvesubscription - サブスクリプション管理

このツールはProxmox VEのサブスクリプションを処理するために使用されます。

### 19.2 pveperf - Proxmox VE ベンチマークスクリプト

PATH（デフォルトは/）にマウントされたハードディスク上のCPU/ハードディスクのパフォーマンスデータの収集を試みる：

#### CPUボゴミップス

全CPUのボゴミップス合計

#### REGEX/SECOND

1秒あたりの正規表現数（perlパフォーマンステスト）は300000以上でなければならない。

#### HDサイズ

ハードディスクサイズ

#### バファードリード

簡単なHD読み取りテスト。最近のHDは少なくとも40MB/秒に達するはずです。

#### 平均シーク時間

は平均シーク時間をテストする。高速SCSI HDは8ミリ秒未満に達する。一般的なIDE/SATAディスクの値は15~20ミリ秒です。

#### FSyncs/SECOND

値は200より大きくなければならない（RAIDコントローラーのライトバックキャッシュモード

を有効にする必要がある - バッテリーバックアップキャッシュ (BBWC) が必要）。

**DNS EXT**

外部DNS名の平均解決時間

**DNS INT**

ローカルDNS名の平均解決時間

## 19.3 Proxmox VE API 用シェルインターフェース

Proxmox VE管理ツール(pvsh)を使用すると、REST/HTTPSサーバを使用せずにAPI関数を直接呼び出すことができます。

---

### 注

それができるのはルートだけだ。

---

### 19.3.1 使用例

クラスタ内のノードのリストを取得する

```
# pvsh get /nodes
```

データセンターで利用可能なオプションのリストを取得する

```
# pvsh usage cluster/options -v
```

HTML5 NoVNCコンソールをデータセンターのデフォルトコンソールとして設定する。

```
# pvsh set cluster/options -console html5
```

## 第20章

### よくある質問

---

#### 注

新しいFAQはこのセクションの一番下に追加されている。

---

1. *Proxmox VE*はどのようなディストリビューションに基づいていますか?

Proxmox VEはDebian GNU/Linuxをベースにしています。

2. *Proxmox VE*プロジェクトはどのライセンスを使用していますか?

Proxmox VEコードのライセンスはGNU Affero General Public License, version 3です。

3. *Proxmox VE*は32ビットプロセッサで動作しますか?

Proxmox VEは64ビットCPU (AMDまたはIntel) でのみ動作します。32ビットの予定はありません。

---

#### 注

VMとコンテナは32ビットと64ビットの両方に対応する。

---

4. 私のCPUは仮想化をサポートしていますか?

CPUが仮想化対応かどうかを確認するには、このコマンドの出力にvmxまたはsvmタグがあるかどうか

```
egrep '(vmx|svm)' /proc/cpuinfo  
をチェックする:
```

5. 対応インテルCPU

インテル® バーチャライゼーション・テクノロジー (インテル® VT-x) をサポートする64ビット・プ

---

ロセッサー。(インテルVTおよび64ビット対応プロセッサー一覧)

## 6. 対応AMD CPU

AMD仮想化テクノロジー (AMD-V) をサポートする64ビットプロセッサー。

## 7. コンテナ／仮想環境 (VE) ／仮想専用サーバー (VPS) とは?

コンテナの文脈では、これらの用語はすべてオペレーティング・システム・レベルの仮想化という概念を指す。オペレーティング・システム・レベルの仮想化とは、オペレーティング・システムのカーネルが、カーネルを共有する複数の分離されたインスタンスを可能にする仮想化の手法である。LXCでは、このようなインスタンスをコンテナと呼ぶ。コンテナは完全なオペレーティング・システムをエミュレートするのではなく、ホストのカーネルを使用するため、オーバーヘッドは少なくて済むが、Linuxゲストに限られる。

## 8. QEMU/KVMゲスト（またはVM）とは何ですか？

QEMU/KVM ゲスト（または VM）は、QEMU と Linux KVM カーネルモジュールを使用して Proxmox VE で仮想化されたゲストシステムです。

## 9. QEMUとは？

QEMUは、汎用的なオープンソースのマシンエミュレータおよび仮想化ツールです。QEMUはLinux KVMカーネルモジュールを使用し、ホストCPU上でゲストコードを直接実行することで、ネイティブに近いパフォーマンスを実現する。Linuxゲストに限らず、任意のオペレーティング・システムを実行できる。

## 10. Proxmox VEのサポート期間は？

Proxmox VEのバージョンは、少なくとも対応するDebianバージョンが`oldstable`である限りサポートされます。Proxmox VE はローリングリリースモデルを採用しており、常に最新の安定版を使用することを推奨します。

プロックスモックスVE バージョン	Debianバージョン	ファースト・リリース	Debian EOL	プロックスモックスEOL
プロックスモックスVE 8	Debian 12 (本の虫)	2023-06	tba	tba
プロックスモックスVE 7	Debian 11 (ブルズアイ)	2021-07	2024-07	2024-07
プロックスモックスVE 6	デビアン10 (バスター)	2019-07	2022-09	2022-09
プロックスモックスVE 5	Debian 9 (ストレッチ)	2017-07	2020-07	2020-07
プロックスモックスVE 4	Debian 8 (ジェシー)	2015-10	2018-06	2018-06
プロックスモックスVE 3	Debian 7 (ウィージー)	2013-05	2016-04	2017-02
プロックスモックスVE 2	Debian 6 (スクリーズ)	2012-04	2014-05	2014-05
プロックスモックスVE 1	Debian 5 (レニー)	2008-10	2012-03	2013-01

## 11. Proxmox VEを次のポイントリリースにアップグレードするには？

マイナーバージョンアップ、例えばバージョン7.1のProxmox VEから7.2や7.3へのアップグレードは、通常のアップデートと同様に行うことができます。ただし、[リリースノート](#)で注意すべき点、または変更点を確認してください。

アップデート自体には、Web UI *Node → Updates* パネル、またはCLIを使用します：

アプトアップデート

```
apt full-upgrade
```

---

### 注

[パッケージ・リポジトリ](#)を正しくセットアップしていることを常に確認し、`apt update`でエラーが出なかった場合のみ、実際のアップグレードを続行してください。

---

12. Proxmox VEを次のメジャーリリースにアップグレードするには?

Proxmox VE 4.4から5.0へのメジャーバージョンアップもサポートされています。アップグレードは慎重に計画し、テストする必要があります。

アップグレードの具体的な手順はそれぞれのセットアップによって異なりますが、アップグレードの一般的な手順とアドバイスを提供します：

- [Proxmox VE 7から8へのアップグレード](#)
- [Proxmox VE 6から7へのアップグレード](#)
- [Proxmox VE 5から6へのアップグレード](#)
- [Proxmox VE 4から5へのアップグレード](#)
- [Proxmox VE 3から4へのアップグレード](#)

### 13. LXC vs LXD vs Proxmox Containers vs Docker

LXCは、Linuxカーネルのコンテナ機能のユーザー空間インターフェイスです。強力なAPIとシンプルなツールにより、Linuxユーザーはシステム・コンテナを簡単に作成・管理できる。LXCは、以前のOpenVZと同様に、**システムの仮想化**を目的としている。そのため、コンテナ内で完全なOSを実行し、sshを使ってログインしたり、ユーザーを追加したり、apacheを実行したりすることができる。

LXDはLXCの上に構築され、より優れた新しいユーザー体験を提供する。LXDは、liblxcとそのGoバインディングを通じてLXCを使用し、コンテナの作成と管理を行う。基本的には、LXCのツールやディストリビューション・テンプレート・システムに代わるもので、ネットワーク経由で制御可能な機能が追加されている。

Proxmox Containersは、Proxmox Container Toolkit (pct)を使用して作成および管理されるコンテナを指す。また、**システム仮想化**をターゲットとし、コンテナ提供の基盤としてLXCを使用します。

Proxmox Container Toolkit (pct) は Proxmox VE と緊密に連携しています。つまり、クラスタ・セットアップを認識し、QEMU仮想マシン (VM) と同じネットワークとストレージ・リソースを使用できる。Proxmox VEのファイアウォールの使用、バックアップの作成と復元、HAフレームワークを使用したコンテナの管理も可能です。Proxmox VE APIを使用して、すべてをネットワーク経由で制御できます。

Dockerは、隔離された自己完結型の環境で**単一の**アプリケーションを実行することを目的としている。これらは一般的に「システムコンテナ」ではなく「アプリケーションコンテナ」と呼ばれる。Dockerインスタンスは、Docker Engineコマンドラインインターフェイスを使ってホストから管理する。Proxmox VEホスト上で直接Dockerを実行することは推奨されません。

---

### 注

アプリケーションコンテナ、例えば*Docker*イメージを実行したい場合は、Proxmox QEMU VM内で実行するのが最適です。

---

## 第21章 書誌

### 21.1 プロックスモックスVEに関する書籍

- [1] [Ahmed16] ワシム・アーメド Mastering Proxmox - Third Edition.Packt Publishing, 2017.ISBN 978-1788397605
- [2] [Ahmed15] Wasim Ahmed.Proxmox Cookbook.Packt Publishing, 2015.ISBN 978-1783980901
- [3] [Cheng14] Simon M.C. Cheng.Proxmox High Availability.Packt Publishing, 2014.ISBN 978-1783980888
- [4] [Goldman16] Rik Goldman.Learning Proxmox VE.Packt Publishing, 2016.ISBN 978-1783981786
- [5] [サーバー16] Lee R. Surber.仮想化大全：Business Basic Edition.Linux Solutions (LRS-TEK), 2016.asin b01bbvqzt6

### 21.2 関連技術に関する書籍

- [6] [Hertzog13] Raphaël Hertzog, Roland Mas., Freexian SARL **Debian 管理者ハンドブック：Debian 管理者ハンドブック: Debian Bullseye の発見から習得まで**, Freexian, 2021.ISBN 979-10-91414- 20-3
- [7] [Bir96] Kenneth P. Birman.Building Secure and Reliable Network Applications.Manning Publications Co, 1996.ISBN 978-1884777295
- [8] [Walsh10] Norman Walsh.DocBook 5: The Definitive Guide.O'Reilly & Associates, 2010. ISBN 978-0596805029
- [9] [Richardson07] Leonard Richardson & Sam Ruby.RESTful Web Services.O'Reilly Media, 2007.ISBN 978-0596529260
- [10] [Singh15] Karan Singh. Learning Ceph.Packt Publishing, 2015.ISBN 978-1783985623

[11] [Singh16] Karan Singh.Ceph Cookbook Packt Publishing, 2016.ISBN 978-1784393502

- [12] [Mauerer08] Wolfgang Mauerer. Professional Linux Kernel Architecture. John Wiley & Sons, 2008. ISBN 978-0470343432
- [13] [Loshin03] Pete Loshin, IPv6: Theory, Protocol, and Practice, 2nd Edition. Morgan Kaufmann, 2003. ISBN 978-1558608108
- [14] [Loeliger12] Jon Loeliger & Matthew McCullough. Gitによるバージョン管理: 共同ソフトウェア開発のための強力なツールとテクニック. O'Reilly and Associates, 2012. ISBN 978-1449316389
- [15] [Kreibich10] Jay A. Kreibich. Using SQLite, O'Reilly and Associates, 2010. ISBN 978-0596521189

## 21.3 関連書籍

- [16] [Bessen09] James Bessen & Michael J. Meurer, Patent Failure: How Judges, Bureaucrats, and Lawyers Put Innovators at Risk. Princeton Univ Press, 2009. ISBN 978-0691143217

## 付録A

# コマンドライン・インターフェース

## A.1 出力フォーマットのオプション [**FORMAT\_OPTIONS**]。

出力形式は--output-formatパラメータで指定できる。デフォルトのフォーマット・テキストは、ASCII-artを使用して表の周りにきれいな枠線を描きます。さらに、いくつかの値を人間が読めるようなテキストに変換します：

- UnixのエポックはISO 8601の日付文字列として表示される。
- 期間は週/日/時/分/秒のカウントで表示される。
- バイトサイズの値には単位 (B、KiB、MiB、GiB、TiB、PiB) が含まれる。
- つまり、1.0は100%と表示される。

オプション--quietを使えば、出力を完全に抑制することもできる。

**--human-readable <boolean>** (デフォルト = 1)

出力レンダリング関数を呼び出して、人間が読めるテキストを生成する。

**--noborder <boolean>** (デフォルト = 0)

ボーダーを描かない (テキスト形式の場合)。

**--noheader <boolean>** (デフォルト = 0)

カラムヘッダを表示しない (テキスト形式の場合)。

**--output-format <json | json-pretty | text | yaml> (default = text)**  
出力フォーマット。

**--quiet <ブール値**  
印刷結果を抑制する。

## A.2 pvesm - Proxmox VE Storage Manager

**pvesm** <COMMAND> [ARGS] [OPTIONS].

**pvesm add** <タイプ> <ストレージ> [オプション].

新しいストレージを作成する。

<type>: <btrfs | cephfs | cifs | dir | esxi | glusterfs | iscsi |  
iscsidirect | lvm | lvmthin | nfs | pbs | rbd | zfs | zfspool>.

収納タイプ。

<ストレージ>: <ストレージID

ストレージ識別子。

--authsupported <文字列

認証済み。

--ベース <文字列

基本音量。このボリュームは自動的にアクティブになります。

--ブロックサイズ <文字列

ブロックサイズ

--bwlimit [clone=<LIMIT>] [,default=<LIMIT>] [,migration=<LIMIT>]

[,move=<LIMIT>] [,restore=<LIMIT>] とする。

さまざまな操作のI/O帯域幅制限を設定する（単位: KiB/s）。

--コムスター\_hg <文字列

コムスター・ビューのホスト・グループ

--comstar\_tg <文字列

コムスター・ビューのターゲット・グループ

--コンテンツ <文字列

許可されるコンテンツタイプ

---

### 注

コンテナには*rootdir*が、VMには*images*が使用される。

---

**--コンテンツ・ディレクトリ <文字列>**

デフォルトのコンテンツ・タイプ・ディレクトリのオーバーライド。

**--ベースパス作成 <ブール値> (デフォルト = yes)**

ベース・ディレクトリが存在しない場合は作成する。

**--サブディレクトリの作成 <論理値> (デフォルト = yes)**

ディレクトリにデフォルトの構造を入力する。

**--データプール <文字列>**

データプール（消去符号化のみ）

**--データストア <文字列>**

Proxmox Backup Serverのデータストア名。

**--無効 <ブール値>**

ストレージを無効にするフラグ。

**--ドメイン <文字列>**

CIFSドメイン。

**--暗号化キーを含むファイル、または特別な値 "autogen"**

暗号化キー。パスフレーズなしで自動生成するには`autogen`を使用します。

**--エクスポート <文字列>**

NFSエクスポートパス。

**--fingerprint ([A-Fa-f0-9]{2}[:]{31}[A-Fa-f0-9]{2})**

証明書の SHA 256 フィンガープリント。

**--format <文字列>**

デフォルトの画像フォーマット。

**-fs-name<文字列>**

Cephファイルシステム名。

**--ヒューズ <ブール>**

FUSEを使用してCephFSをマウントします。

**--is\_mountpoint <string> (デフォルト = no)**

指定されたパスが外部で管理されているマウントポイントであると仮定し、マウントされていない場合はストレージをオフラインとみなす。ブーリアン（yes/no）値を使用すると、このフィールドでターゲ

ット・パスを使用するショートカットになります。

**--iscsiprovider <文字列**

ISCSIプロバイダ

**--cephクラスタで認証するためのキーリングを含むkeyringファイル**

クライアントのキーリングの内容（外部クラスタ用）。

**--krbd <ブール値**

常にkrbdカーネルモジュールを通してrbdにアクセスする。

**--lio\_tpg <文字列>**

Linux LIOターゲット用ターゲット・ポータル・グループ

**-master-pubkey** PEM形式のマスター公開鍵を含むファイル Base64エンコードされたPEM形式のRSA公開鍵。暗号化された各バックアップに追加される暗号化キーのコピーを暗号化するために使用されます。

**--max-protected-backups <integer> (-1 - N) (デフォルト = Datastore.Allocate権限を持つユーザは無制限、他のユーザは5)**  
ゲストごとの保護バックアップの最大数。無制限には-1を使用します。

**-マックスファイル <整数> (0 - N)**

非推奨：代わりに*prune-backups*を使用してください。VMごとのバックアップファイルの最大数。無制限の場合は0を使用します。

**--mkdir <ブール値> (デフォルト = yes)**

ディレクトリが存在しない場合は作成し、デフォルトのサブディレクトリを設定します。注意：非推奨。*create-base-path* と *create-subdirs* オプションを使用する。

**--モノホスト <文字列>**

モニターのIPアドレス（外部クラスタの場合）。

**--マウントポイント <文字列>**  
マウントポイント

**--名前空間 <文字列>**  
名前空間。

**--nocow <ブール値> (デフォルト = 0)**

ファイルにNOCOWフラグを設定する。データのチェックサムを無効にし、直接I/Oを許可しながら、データエラーを回復できなくします。このフラグを使用するのは、基礎となるレイドシステムがない単一の ext4 フォーマットディスク上よりもデータを安全にする必要がない場合だけです。

**--ノード <文字列>**

ストレージ構成が適用されるノードのリスト。

**--nowritecache <ブール値**  
ターゲットの書き込みキャッシングを無効にする

**--オプション <文字列**  
NFS/CIFS マウントオプション (*man nfs* または *man mount.cifs* を参照)

**--パスワード <password**  
共有/データストアにアクセスするためのパスワード。

**--パス <文字列**  
ファイルシステムのパス。

**--プール <文字列**

プールだ。

**--ポート <整数> (1 - 65535)**

デフォルト以外のポートの場合。

**--portal <文字列**

iSCSIポータル (IPまたはDNS名とオプションのポート)。

**--preallocation <falloc | full | metadata | off> (デフォルト = metadata) raw**

および qcow2 画像に対するプリアロケーションモード。raw 画像でメタデータを使うと preallocation=off になります。

**--剪定バックアップ [keep-all=<1|0>] [,keep-daily=<N>]****[,keep-hourly=<N>] [,keep-last=<N>] [,keep-monthly=<N>] [,keep-weekly=<N>] [,keep-yearly=<N>]**

間隔が短い保持オプションが最初に処理され、--keep-lastが一番最初に処理される。

各オプションは特定の期間をカバーします。この期間内のバックアップはこのオプションの対象となります。次のオプションでは、すでにカバーされているバックアップは考慮されず、古いバックアップのみが考慮されます。

**--セーフリムーブ <ブール値**

LVの削除時にデータをゼロにする。

**-saferemove\_throughput<文字列**

ワイスループット (cstream -tパラメータ値)。

**--サーバー <文字列**

サーバーIPまたはDNS名。

**--サーバー2 <文字列**

バックアップファイルサーバーのIPまたはDNS名。

---

**注**

必要なオプション: サーバー

---

## --共有 <文字列>

CIFS共有。

## --共有 <ブール値>

すべてのノード（または`nodes`オプションにリストされているすべて）で同じ内容を持つ単一のストレージであることを示します。ローカルストレージの内容が自動的に他のノードからアクセスできるようになるわけではなく、すでに共有されているストレージをそのようにマークするだけです！

## `-skip-cert-verification <boolean> (デフォルト = false)`

TLS証明書の検証を無効にし、完全に信頼できるネットワークでのみ有効にする！

**--smbversion <2.0 | 2.1 | 3 | 3.0 | 3.11 | default> (default = デフォルト)**

SMBプロトコルのバージョン。デフォルトでは、設定されていない場合、クライアントとサーバーの両方でサポートされている最高のSMB2+バージョンをネゴシエートする。

**-スパース <ブール値**

スパースボリュームを使う

**--サブディレクトリ <文字列**

マウントするサブディレクトリ。

**--タグ付きのみ <論理値**

*pve-vm-ID* でタグ付けされた論理ボリュームのみを使用します。

**--ターゲット <文字列**

iSCSI ターゲット。

**--シンプール <文字列**

LVMシンプールLV名。

**--トランスポート <rdma | tcp | unix>**

グラスター・トランスポート: tcpまたはrdma

**--ユーザー名 <文字列**

RBD Id.

**--vgname <文字列**

ボリュームグループ名。

**--ボリューム <文字列**

Glusterfsボリューム。

**pvesm alloc <ストレージ> <vmid> <ファイル名> <サイズ> [オプション]。**

ディスクイメージを割り当てる。

**<ストレージ>: <ストレージID**

ストレージ識別子。

**<vmid>**: <整数> (100 - 999999999)

オーナーVMの指定

**<ファイル名>**: <文字列>

作成するファイル名。

**<サイズ>**: \d+[MG]?

キロバイト (1024バイト) 単位のサイズ。オプションの接尾辞M (メガバイト、1024K) およびG (ギガバイト、1024M)。

--format <qcow2 | raw | subvol>.

記述なし

---

## 注

必要なオプション: サイズ

---

### pvesm apiinfo

APIVERとAPIAGEを返す。

### pvesm cifsscan

pvesm scan cifs のエイリアス。

**pvesm export** <ボリューム> <フォーマット> <ファイル名> [オプション].

ボリュームのエクスポートに内部的に使用される。

#### <ボリューム>: <文字列

ボリューム識別子

<format>: <btrfs | qcow2+size | raw+size | tar+size | vmdk+size | zfs>.

ストリーム形式のエクスポート

#### <ファイル名>: <文字列

保存先ファイル名

--ベース (?^i:[a-zA-Z0-9\_-]{1,40})

からインクリメンタルストリームを開始するスナップショット。

--snapshot (?^i:[a-zA-Z0-9\_\-]{1,40})

エクスポートするスナップショット

#### --スナップショット・リスト <文字列

転送するスナップショットの順序付きリスト

--with-snapshots <boolean> (デフォルト = 0)

ストリームに中間スナップショットを含めるかどうか

---

**pvesm extractconfig <ボリューム>。**

vzdumpバックアップアーカイブから設定を抽出する。

**<ボリューム>: <文字列**

ボリューム識別子

**pvesm free** <ボリューム> [オプション].

ボリュームの削除

<ボリューム>: <文字列>

ボリューム識別子

--ディレイ <整数> (1 - 30)

タスクが終了するまでの待ち時間。その時間内にタスクが終了した場合は*null*を返す。

--ストレージID

ストレージ識別子。

**pvesm glusterfsscan**

*pvesm scan glusterfs* のエイリアス。

**pvesm help** [OPTIONS]

指定したコマンドに関するヘルプを表示する。

--extra-args <array>

特定のコマンドのヘルプを表示する

--verbose <ブール値>

冗長な出力形式。

**pvesm import** <volume> <format> <filename> [OPTIONS].

ボリュームのインポートに内部的に使用される。

<ボリューム>: <文字列>

ボリューム識別子

<format>: <btrfs | qcow2+size | raw+size | tar+size | vmdk+size | zfs>.

インポート・ストリーム・フォーマット

<ファイル名>: <文字列>

ソースファイル名。stdinを使用する場合、tcp://<IP-or-CIDR>フォーマットはTCPコネクションを、unix://PATH-TO-SOCKETフォーマットはUNIXソケットを入力として使用することができる。

**--allow-rename <boolean> (デフォルト = 0)**

要求されたボリュームIDがすでに存在する場合は、エラーをスローする代わりに新しいボリュームIDを選択します。

**--ベース (?^i:[a-zA-Z0-9\_-]{1,40})**

増分ストリームの基本スナップショット

**--delete-snapshot** (?^i:[a-zA-Z0-9\_\\-]{1,80})  
成功時に削除するスナップショット

**--snapshot** (?^i:[a-zA-Z0-9\_\\-]{1,40})  
ストリームがスナップショットを含む場合、現在の状態のスナップショット

**--with-snapshots** <boolean> (デフォルト = 0)  
ストリームが中間スナップショットを含むかどうか

### **pvesm iscsiscan**

*pvesm scan iscsi* のエイリアス。

### **pvesm list** <storage> [OPTIONS] .

ストレージの内容をリストアップする。

<ストレージ>: <ストレージID

ストレージ識別子。

**--コンテンツ** <文字列

このタイプのコンテンツのみをリストアップする。

**--vmid** <整数> (100 - 999999999)

このVMの画像のみをリストアップ

### **pvesm lvmscan**

*pvesm scan lvm* のエイリアス。

### **pvesm lvmthinscan**

*pvesm scan lvmthin* のエイリアス。

### **pvesm nfsscan**

*pvesm scan nfs* のエイリアス。

### **pvesm パス** <ボリューム>

指定されたボリュームのファイルシステムのパスを取得する

<ボリューム>: <文字列

## ボリューム識別子

**pvesm prune-backups <storage> [OPTIONS] .**

バックアップの刈り込み。標準的な命名スキームを使用しているものだけが考慮されます。keepオプションが指定されていない場合は、ストレージ構成のものが使用されます。

<ストレージ>: <ストレージID>

ストレージ識別子。

**- ドライ・ラン <ブール値>**

刈り込まれるものだけを表示し、何も削除しない。

**--keep-all <ブール値>**

すべてのバックアップを保持する。trueの場合、他のオプションと競合します。

**--キープ・デイリー <N>**

過去<N>日分のバックアップを保持します。1日に複数のバックアップがある場合は、最新のものだけが保持されます。

**--キープ・アワー<N>**

直近の<N>異なる時間のバックアップを保持します。1時間に複数のバックアップがある場合は、最新のものだけが保持されます。

**--キープ・ラスト <N>**

最後の<N>個のバックアップを保持する。

**--キープ・マンスリー<N>**

過去<N>の異なる月のバックアップを保持します。1つの月に複数のバックアップがある場合は、最新のものだけが保持されます。

**--キープ・ウィークリー<N>**

過去<N>週間分のバックアップを保持します。1週間に複数のバックアップがある場合は、最新のものだけが保持されます。

**--キープ・イヤー・リー <N>**

過去<N>年分のバックアップを保持する。一つの年に複数のバックアップがある場合、最新のものが保存されます。

**--タイプ <lxc | qemu>**

qemuかlxcのどちらか。このタイプのゲストのバックアップのみを考慮してください。

**--vmid <整数> (100 - 999999999)**

このゲストのバックアップのみを考慮する。

**pvesm remove <ストレージ>**

ストレージ構成を削除する。

**<ストレージ>: <ストレージID>**

ストレージ識別子。

**pvesm scan cifs <server> [OPTIONS] .**

リモートCIFSサーバーをスキャンします。

**<サーバー>: <文字列>**

サーバーのアドレス（名前またはIP）。

**--ドメイン <文字列>**

SMB ドメイン (ワークグループ)。

**--パスワード <password>**

ユーザー パスワード。

**--ユーザー名 <文字列>**

ユーザー名

**pvesm scan glusterfs <サーバー>**

リモートの GlusterFS サーバをスキャンする。

**<サーバー>: <文字列>**

サーバーのアドレス (名前または IP)。

**pvesm scan iscsi <ポータル>**

リモート iSCSI サーバをスキャンする。

**<ポータル>: <文字列>**

iSCSI ポータル (IP または DNS 名とオプションのポート)。

**pvesm scan lvm**

ローカル LVM ボリューム グル

ープを一覧表示します。

**pvesm scan lvmthin <vg>**

ローカル LVM シンプールを一

覧表示します。

**<vg>: [a-zA-Z0-9\.\+\\_\\_][a-zA-Z0-9\.\+\\_\-\\_]+\\_**  
記述なし

**pvesm scan nfs <server>**

リモート NFS サーバをスキャンします。

## <サーバー>: <文字列

サーバーのアドレス（名前またはIP）。

```
pvesm scan pbs <server> <username> --password <string> [OPTIONS]  
[FORMAT_OPTIONS]
```

リモートのProxmox Backup Serverをスキャンします。

## <サーバー>: <文字列

サーバーのアドレス（名前またはIP）。

**<ユーザー名>: <文字列**

ユーザー名またはAPIトークンID。

```
--fingerprint ([A-Fa-f0-9]{2}:){31}[A-Fa-f0-9]{2}
```

証明書の SHA 256 フィンガープリント。

**--パスワード <文字列**

ユーザーパスワードまたはAPIトークンの秘密。

```
--ポート <整数> (1 - 65535) (デフォルト = 8007)
```

オプションのポート。

**pvesm scan zfs**

ローカルノードのzfsプールリストをスキャンします。

**pvesm set <storage> [OPTIONS] .**

ストレージ構成を更新する。

**<ストレージ>: <ストレージID**

ストレージ識別子。

```
--ブロックサイズ <文字列
```

ブロックサイズ

```
--bwlimit [clone=<LIMIT>] [,default=<LIMIT>] [,migration=<LIMIT>]
```

```
[,move=<LIMIT>] [,restore=<LIMIT>] とする。
```

さまざまな操作のI/O帯域幅制限を設定する（単位：KiB/s）。

```
--コムスター_hg <文字列
```

コムスター・ビューのホスト・グループ

```
--comstar_tg <文字列
```

コムスター・ビューのターゲット・グループ

```
--コンテンツ <文字列
```

許可されるコンテンツタイプ

---

**注**

コンテナには`rootdir`が、VMには`images`が使用される。

---

**--コンテンツ・ディレクトリ <文字列>**

デフォルトのコンテンツ・タイプ・ディレクトリのオーバーライド。

**--ベースパス作成 <ブール値> (デフォルト = yes)**

ベース・ディレクトリが存在しない場合は作成する。

**--サブディレクトリの作成 <論理値> (デフォルト = yes)**

ディレクトリにデフォルトの構造を入力する。

**--データプール <文字列>**

データプール（消去符号化のみ）

**--削除 <文字列>**

削除したい設定のリスト。

**-ダイジェスト <文字列>**

現在のコンフィギュレーション・ファイルのダイジェストが異なる場合、変更を防ぐ。これは、現在のコンフィギュレーション・ファイルの変更を防ぐために使用できます。

**--無効 <ブール値>**

ストレージを無効にするフラグ。

**--ドメイン <文字列>**

CIFS ドメイン。

**--暗号化キーを含むファイル、または特別な値 "autogen"**

暗号化キー。パスフレーズなしで自動生成するには *autogen* を使用します。

**--fingerprint ([A-Fa-f0-9]{2}:){31}[A-Fa-f0-9]{2}**

証明書の SHA 256 フィンガープリント。

**--format <文字列>**

デフォルトの画像フォーマット。

**-fs-name<文字列>**

Ceph ファイルシステム名。

**--ヒューズ <ブール>**

FUSE を使用して CephFS をマウントします。

**--is\_mountpoint <string> (デフォルト = no)**

指定されたパスが外部で管理されているマウントポイントであると仮定し、マウントされていない場合はストレージをオフラインとみなす。ブーリアン (yes/no) 値を使用すると、このフィールドでターゲ

ット・パスを使用するショートカットになります。

**--ceph** クラスタで認証するためのキーリングを含む **keyring** ファイル  
クライアントのキーリングの内容（外部クラスタ用）。

**--krbd <ブール値**  
常にkrbdカーネルモジュールを通してrbdにアクセスする。

**--lio\_tpg <文字列**  
Linux LIOターゲット用ターゲット・ポータル・グループ

**-master-pubkey-PEM** フォーマットのマスター公開鍵を含むファイル Base64エンコードされた PEMフォーマットのRSA公開鍵。暗号化された各バックアップに追加される暗号化キーのコピーを暗号化するために使用されます。

**--max-protected-backups <integer>** (-1 - N) (デフォルト = Datastore.Allocate権限を持つユーザは無制限、他のユーザは5)  
ゲストごとの保護バックアップの最大数。無制限には-1を使用します。

#### **-マックスファイル <整数>** (0 - N)

非推奨：代わりに*prune-backups*を使用してください。VMごとのバックアップファイルの最大数。無制限の場合は0を使用します。

#### **--mkdir <ブール値>** (デフォルト = yes)

ディレクトリが存在しない場合は作成し、デフォルトのサブディレクトリを設定します。注：非推奨。*create-base-path* と *create-subdirs* オプションを使用する。

#### **--モノホスト <文字列>**

モニターのIPアドレス（外部クラスタの場合）。

#### **--マウントポイント <文字列>** マウントポイント

#### **--名前空間 <文字列>** 名前空間。

#### **--nocow <ブール値>** (デフォルト = 0)

ファイルにNOCOWフラグを設定する。データのチェックサムを無効にし、直接I/Oを許可しながら、データエラーを回復できなくします。このフラグを使用するのは、基礎となるレイドシステムがない単一の ext4 フォーマットディスク上よりもデータを安全にする必要がない場合だけです。

#### **--ノード <文字列>**

ストレージ構成が適用されるノードのリスト。

#### **--nowritecache <ブール値>** ターゲットの書き込みキャッシングを無効にする

**--オプション <文字列>**

NFS/CIFS マウントオプション (*man nfs* または *man mount.cifs* を参照)

**--パスワード <password>**

共有/データストアにアクセスするためのパスワード。

**--プール <文字列>**

プールだ。

**--ポート <整数> (1 - 65535)**

デフォルト以外のポートの場合。

**--preallocation <fallback | full | metadata | off>** (デフォルト = metadata) raw

および qcow2 画像に対するプリアロケーションモード。raw 画像でメタデータを使うと preallocation=off になります。

**--剪定バックアップ [keep-all=<1|0>] [,keep-daily=<N>]**

[,keep-hourly=<N>] [,keep-last=<N>] [,keep-monthly=<N>] [,keep-weekly=<N>] [,keep-yearly=<N>]

間隔が短い保持オプションが最初に処理され、--keep-lastが一番最初に処理される。

各オプションは特定の期間をカバーします。この期間内のバックアップはこのオプションの対象となります。次のオプションでは、すでにカバーされているバックアップは考慮されず、古いバックアップのみが考慮されます。

**--セーフリムーブ <ブール値>**

LVの削除時にデータをゼロにする。

**-saferemove\_throughput<文字列>**

ワイプスループット (cstream -tパラメータ値)。

**--サーバー <文字列>**

サーバーIPまたはDNS名。

**--サーバー2 <文字列>**

バックアップファイルサーバーのIPまたはDNS名。

---

### 注

必要なオプション: サーバー

---

**--共有 <ブール値>**

すべてのノード（またはnodesオプションにリストされているすべて）で同じ内容を持つ単一のストレージであることを示します。ローカルストレージの内容が自動的に他のノードからアクセスできるようになるわけではなく、すでに共有されているストレージをそのようにマークするだけです！

**-skip-cert-verification <boolean>** (デフォルト = false)

TLS証明書の検証を無効にし、完全に信頼できるネットワークでのみ有効にする！

---

**--smbversion <2.0 | 2.1 | 3 | 3.0 | 3.11 | default> (default = デフォルト)**

SMBプロトコルのバージョン。デフォルトでは、設定されていない場合、クライアントとサーバーの両方でサポートされている最高のSMB2+バージョンをネゴシエートする。

**-スパース <ブール値**  
スパースボリュームを使う

**--サブディレクトリ <文字列**

マウントするサブディレクトリ。

**--タグ付きのみ <論理値>**

*pve-vm-ID* でタグ付けされた論理ボリュームのみを使用します。

**--トランスポート <rdma | tcp | unix>**

グラスター・トランスポート: tcpまたはrdma

**--ユーザー名 <文字列>**

RBD Id.

**pvesm status [OPTIONS]**

すべてのデータストアのステータスを取得する。

**--コンテンツ <文字列>**

このコンテンツタイプをサポートする店舗のみをリストアップします。

**--有効 <ブール値> (デフォルト = 0)**

有効になっている（コンフィグで無効になっていない）店舗のみをリストアップする。

**--format <boolean> (デフォルト = 0)**

フォーマットに関する情報を含む

**--ストレージID**

指定されたストレージのステータスのみをリストアップ

**--ターゲット <文字列>**

ターゲットがノードと異なる場合は、このノードと指定されたターゲットノードでアクセス可能な共有ストレージのみをリストアップします。

**pvesm zfsscan**

*pvesm scan zfs* のエイリアス。

## A.3 pvesubscription - Proxmox VE サブスクリプションマネージャ

**pvesubscription <COMMAND> [ARGS] [OPTIONS].**

## **pvesubscription削除**

このノードのサブスクリプションキーを削除する。

### **サブスクリプション取得**

**pvesubscription help [OPTIONS]**

指定したコマンドに関するヘルプを

表示します。

**--extra-args <array>**

特定のコマンドのヘルプを表示する

**--verbose <ブール値>**

冗長な出力形式。

**pvesubscriptionセット<キー>**

サブスクリプション・キーを設定する。

**<key>: \s\*pve([1248])([cbsp])- [0-9a-f]{10}\s\***  
Proxmox VEサブスクリプション・キー

**pvesubscription set-offline-key <data> (pvesubscriptionセットオフラインキー <データ>)**

内部使用のみです！ オフラインキーを設定するには、代わりに proxmox-offline-mirror-helper パッケージを使ってください。

**<データ>: <文字列>**

署名された購読情報の塊

**pvesubscription update [OPTIONS] [オプション]。**

購読情報を更新する。

**--force <ブール値> (デフォルト = 0)**

ローカルのキャッシュがまだ有効であっても、常にサーバーに接続する。

## A.4 pveperf - Proxmox VE ベンチマークスクリプト

**pveperf [PATH]**

## A.5 pveceph - Proxmox VE ノードで CEPH サービスを管理する

**pveceph <COMMAND> [ARGS] [OPTIONS] .****pvecephのcreatemgr**

*pveceph mgr create* のエイリアス。

### ヴェセフ・クリエイトモン

*pveceph mon create* のエイリアス。

### プベセフ・クリエイト・オスド

*pveceph osd create* のエイリアス。

### プール作成

*pveceph pool create* のエイリアス。

## プベセフ・デストロイムグ

*pveceph mgr destroy* のエイリアス。

## プヴェセフ・デストロイモン

*pveceph mon destroy* の別名。

## スティーブ・デストロイド

*pveceph osd destroy* のエイリアス。

## プベセフ・デストロイプール

*pveceph pool destroy* のエイリアス

ス 。 **pveceph fs create**

[OPTIONS] Cephファイルシステム

テムを作成します。

**--add-storage <boolean>** (デフォルト = 0)

作成されたCephFSをこのクラスタのストレージとして構成します。

**--name <string>** (デフォルト = cephfs)

Cephファイルシステム名。

**--pg\_num <整数>** (8 - 32768) (デフォルト = 128)

バッキング・データ・プールの配置グループ数。メタデータ・プールはこの4分の1を使用する。

**pveceph fs destroy <name> [OPTIONS] [オプション]。**

Cephファイルシステムを破棄する

<名前>: <文字列

Cephファイルシステム名。

**--remove-pools <boolean>** (デフォルト = 0)

このファイル用に設定されたデータとメタデータのプールを削除する。

**--remove-storages <boolean>** (デフォルト = 0)

この fs に設定されている pveceph 管理下のストレージをすべて削除する。

**pveceph help [OPTIONS]** (ヘルプ)

指定したコマンドに関するヘルプを表示する。

**--extra-args <array>**

特定のコマンドのヘルプを表示する

**--verbose <ブール値**

冗長な出力形式。

**pveceph init** [OPTIONS] [OPTIONS]。

cephの初期デフォルト構成を作成し、シンボリックリンクを設定します。

#### --cluster-network <文字列>

独立したクラスタネットワークを宣言し、OSDはハートビート、オブジェクトレプリケーション、リカバリのトラフィックをルーティングします。

---

#### 注

必要なオプション: ネットワーク

---

#### --disable\_cephx <boolean> (デフォルト = 0)

cephx認証を無効にします。



#### 警告

cephxは中間者攻撃から保護するセキュリティ機能です。ネットワークがプライベートな場合にのみ、cephxの解除を検討してください!

---

#### -最小サイズ <整数> (1 - 7) (デフォルト = 2)

I/Oを許可するために、オブジェクトごとに利用可能なレプリカの最小数

#### --ネットワーク <文字列>

すべてのceph関連トラフィックに特定のネットワークを使用する。

#### --pg\_bits <整数> (6 - 14) (デフォルト = 6)

配置グループのデフォルト数を指定するために使用される配置グループビット

。非推奨。この設定は、最近のCephバージョンで非推奨になりました。

#### --サイズ <整数> (1 - 7) (デフォルト = 3)

オブジェクトあたりの目標レプリカ数

---

**pveceph install** [OPTIONS] (オプション)

ceph関連パッケージをインストールします。

**--allow-experimental <boolean> (デフォルト = 0)**

実験的バージョンを許可する。注意して使用してください!

**--リポジトリ <エンタープライズ | サブスクリプションなし | テスト> (デフォルト = 企業)**

使用するCephリポジトリ。

**--バージョン <quincy | reef> (デフォルト = quincy)**

インストールするCephのバージョン。

## 鞭打ち症

**pveceph pool ls** のエイリアス。

**pveceph mds create** [OPTIONS]

Ceph メタデータサーバ(MDS)を作成

します。

--hotstandby <boolean> (デフォルト = 0)

ceph-mds デーモンがアクティブなデータシートのログをポーリングして再生するかどうかを決定します。データシートの障害時の切り替えが速くなります、アイドルリソースがより多く必要になります。

--name [a-zA-Z0-9] ([a-zA-Z0-9-]\*[a-zA-Z0-9])? (*default = nodename*)

省略時はノード名と同じ。

**pveceph mds destroy** <名前>.

Ceph メタデータサーバの破棄

<name>: [a-zA-Z0-9] ([a-zA-Z0-9\-\-]\*[a-zA-Z0-9])?  
mds の名前 (ID)

**pveceph mgr create** [OPTIONS] [オプション]。

Ceph Manager の作成

--id [a-zA-Z0-9] ([a-zA-Z0-9\-\-]\*[a-zA-Z0-9])?  
省略時はノード名と同じ。

**pveceph mgr** <id> を破壊する。

Ceph Manager を破棄する。

<id>: [a-zA-Z0-9] ([a-zA-Z0-9\-\-]\*[a-zA-Z0-9])?  
マネージャーの ID

**pveceph mon create** [OPTIONS] [オプション] ...

Ceph モニタとマネージャの作成

## -モン・アドレス <文字列>

自動検出されたモニタIPアドレスを上書きします。Cephのパブリックネットワーク内にある必要があります。

--monid [a-zA-Z0-9]([a-zA-Z0-9\-\-]\*[a-zA-Z0-9])?

省略時はノード名と同じ。

## pveceph モンを破壊する <monid>

Ceph MonitorとManagerを破棄する。

<monid>: [a-zA-Z0-9] ([a-zA-Z0-9\-\-]\*[a-zA-Z0-9])?  
モニターID

**pveceph osd create** <dev> [OPTIONS] [オプション]。

OSDの作成

<dev>: <文字列  
ブロックデバイス名。

#### -クラッシュ・デバイス・クラス <文字列

OSDのデバイスクラスをクラッシュで設定する。

#### --db\_dev <文字列

block.dbのブロックデバイス名。

--db\_dev\_size <number> (1 - N) (デフォルト=bluestore\_block\_db\_size または OSD サイズの 10%)  
block.dbのGiB単位のサイズ。

---

#### 注

必要なオプション: db\_dev

---

#### --暗号化 <ブール値> (デフォルト = 0)

OSDの暗号化を有効にする。

#### --osds-per-device <整数> (1 - N)

物理デバイスごとのOSDサービス。高速な NVMe デバイスにのみ有効である」。

#### --wal\_dev <文字列

block.walのブロックデバイス名。

--wal\_dev\_size <number> (0.5 - N) (デフォルト = bluestore\_block\_wal\_size または OSD サイズの 1%)  
block.walのGiBサイズ。

---

---

## 注

必要なオプション: wal\_dev

---

**pveceph osd destroy** <osdid> [OPTIONS] [オプション]。

OSDを破壊する

**<osdid>: <整数>**

OSD ID

**--クリーンアップ <ブール値> (デフォルト = 0)**

もしセットされていれば、パーティションテーブルのエントリーを削除する。

**pveceph osd details <osdid> [OPTIONS] [FORMAT\_OPTIONS].**

OSDの詳細を取得する。

**<osdid>: <文字列>**

OSDのID

**--verbose <boolean> (デフォルト = 0)**

json-prettyの出力形式と同じ。

**pveceph pool create <name> [OPTIONS] [オプション]。**

Cephプールを作成する

**<名前>: <文字列>**

プールの名前。一意でなければならない。

**--add\_storages <boolean> (デフォルト = 0; データ消去プールの場合: 1)**

新しいプールを使用してVMとCTストレージを構成する。

**--アプリケーション <cephfs | rbd | rgw> (デフォルト = rbd)**

プールのアプリケーション。

**--クラッシュ・ルール <文字列>**

クラスタ内のオブジェクト配置のマッピングに使用するルール。

**--erasure-coding k=<integer>,m=<integer> [,device-class=<class>] [,failure-domain=<domain>] [,profile=<profile>]**

◦

RBD用に消去コード化プールを作成し、メタデータ・ストレージ用にレプリケート・プールを付随させる。

ECでは、共通のcephオプションsize、min\_size、crush\_ruleパラメータがメタデータプールに適用されます

◦

**-最小サイズ <整数> (1 - 7) (デフォルト = 2)**

オブジェクトあたりの最小レプリカ数

**--pg\_autoscale\_mode <off | on | warn> (デフォルト = warn)**

プールの自動PGスケーリングモード。

**--pg\_num <整数> (1 - 32768) (デフォルト = 128)**

配置グループの数。

--pg\_num\_min <整数> (-N - 32768)

配置グループの最小数。

--サイズ <整数> (1 - 7) (デフォルト = 3)

オブジェクトごとのレプリカ数

--target\_size ^(M|G|T|)<数> ([K|M|G|T])?

PGオートスケーラーのプールの推定目標サイズ。

-ターゲット・サイズ比率 <数値>

PGオートスケーラーのプールの推定目標比率。

**pveceph pool destroy** <name> [OPTIONS] [オプション]。

プールを破壊する

<名前>: <文字列>

プールの名前。一意でなければならない。

--force <ブール値> (デフォルト = 0)

trueの場合、使用中であってもプールを破棄する。

--remove\_ecprofile <boolean> (デフォルト = 1)

消去コード・プロファイルを削除します。該当する場合、デフォルトはtrueです。

--remove\_storages <boolean> (デフォルト = 0)

このプールに設定されているすべての pveceph 管理ストレージを削除します。

**pveceph pool get** <name> [OPTIONS] [FORMAT\_OPTIONS].

現在のプールの状態を表示する。

<名前>: <文字列>

プールの名前。一意でなければならない。

**--verbose <boolean>** (デフォルト = 0)

有効にすると、追加データ（統計など）が表示されます。

**pveceph プール ls** [FORMAT\_OPTIONS]。

すべてのプールとその設定 (POST/PUT エンドポイントで設定可能) を一覧表示します。

**pveceph pool set <name> [OPTIONS] [オプション]**。

POOL 設定の変更

**<名前>: <文字列>**

プールの名前。一意でなければならない。

--application <cephfs | rbd | rgw>.

プールのアプリケーション。

--クラッシュ・ルール <文字列

クラスタ内のオブジェクト配置のマッピングに使用するルール。

-最小サイズ <整数> (1 - 7)

オブジェクトあたりの最小レプリカ数

-pg\_autoscale\_mode (オートスケールモード) <オフ | オン | 警告

プールの自動PGスケーリングモード。

--pg\_num <整数> (1 - 32768)

配置グループの数。

--pg\_num\_min <整数> (-N - 32768)

配置グループの最小数。

--サイズ <整数> (1 - 7)

オブジェクトごとのレプリカ数

--target\_size ^(<数> [<単位>])?

PGオートスケーラーのプールの推定目標サイズ。

-ターゲット・サイズ比率 <数値

PGオートスケーラーのプールの推定目標比率。

**pveceph purge [OPTIONS] (ページ・オプション)**

ceph関連のデータと構成ファイルを破棄する。

--クラッシュ <ブール

さらに、/var/lib/ceph/crashのCephクラッシュログをページする。

--ログ <ブール値

さらに、/var/log/cephのCephログをページします。

**pveceph start** [OPTIONS] [オプション].

cephサービスを開始します。

#### --サービス

(ceph|mon|mds|osd|mgr) (\.[a-zA-Z0-9]([a-zA-Z0-9\-\-]\* [a-zA-Z0-9])?)?  
(デフォルト = ceph.target)

Cephサービス名。

## pvecephステータス

Cephのステータスを取得する。

**pveceph stop** [OPTIONS] [オプション]。

cephサービスを停止します。

### --サービス

(ceph|mon|mds|osd|mgr)(\.[a-zA-Z0-9]([a-zA-Z0-9\-\-]\* [a-zA-Z0-9])?)?  
(デフォルト = ceph.target)

Cephサービス名。

## A.6 pvenode - Proxmox VE ノード管理

**pvenode** <COMMAND> [ARGS] [OPTIONS].

**pvenode acme account deactivate** [<name>] (アクメ・アカウントの無効化)

CAで既存のACMEアカウントを停止する。

<name>: <名前> (デフォルト = デフォルト)

ACMEアカウント設定ファイル名。

**pvenode acme account info** [<name>] [FORMAT\_OPTIONS].

既存のACMEアカウント情報を返します。

<name>: <名前> (デフォルト = デフォルト)

ACMEアカウント設定ファイル名。

**pvenode** アクメ・アカウント・リスト

ACMEEAccount インデックス。

**pvenode acme account register** [<name>] {<contact>} [OPTIONS].

互換性のある CA に新しい ACME アカウントを登録する。

**<name>: <名前> (デフォルト = デフォルト)**

ACMEアカウント設定ファイル名。

**<連絡先>: <文字列**

連絡先メールアドレス

**-ディレクトリ `http://.*`**

ACME CA ディレクトリエンドポイントの URL。

**pvenode acme account update** [<name>] [OPTIONS] .

既存の ACME アカウント情報を CA で更新する。注意：新しいアカウント情報を指定しないと、更新がトリガれます。

<name>: <名前> (デフォルト = デフォルト)

ACMEアカウント設定ファイル名。

--連絡先 <文字列

連絡先メールアドレス

**pvenode acme cert order** [OPTIONS] (アクメ証明書注文) .

ACME 互換の CA に新しい証明書を注文する。

--force <ブール値> (デフォルト = 0)

既存のカスタム証明書を上書きする。

**pvenode acme cert renew** [OPTIONS] (アクメ証明書更新オプション

CA から既存の証明書を更新する。

--force <ブール値> (デフォルト = 0)

有効期限が30日以上先であっても、強制的に更新する。

**pvenode acme 証明書失効**

CA から既存の証明書を失効させる。

**pvenode acme plugin add** <type> <id> [OPTIONS] .

ACMEプラグインの設定を追加します。

<タイプ>: <dns | standalone>.

ACMEのチャレンジタイプ。

<id>: <文字列

ACMEプラグインID名

---

```
--api <1984hosting | acmedns | acmeproxy | active24 | ad | ali |
anx | artfiles | arvan | aurora | autodns | aws | azion | azure |
bookmyname | bunny | cf | clouddns | cloudns | cn | conoha |
constellix | cpanel | curanet | cyon | da | ddns | desec | df |
dgon | dnsexit | dnshome | dnimple | dnsservices | do | doapi |
domeneshop | dp | dpi | dpi | dnshome | dnservices | dodesec | df |
| dgon | dnseexit | dnshome | dnsimple | dnsservices | do | doapi |
domeneshop | dp | dpi | dreamhost | duckdns | durabledns | dyn |
dynu | dynv6 | easydns | edgedns | euserv | exoscale | fornex |
freedns | gandi_livedns | gcloud | gcore | gd | geoscaling |
googledomains | he | hetzner | hexonet | hostingde | huaweicloud |
infoblox | infomaniak | internetbs | inwx | ionos | ipv64 |
ispconfig | jd | joker | kappernet | kas | kinghost | knot | la |
leaseweb | lexicon | linode | linode_v4 | loopia | lua | maradns |
me | miab | misaka | myapi | mydevil | mydnsjp | mythic_beasts |
namecheap | namecom | namesilo | nanelo | nederhost | neodigit |
netcup | netlify | nic | njalla | nm | nsd | nsone | nsupdate | nw |
oci | one | online | openprovider | openstack | opnsense | ovh |
pdns | pleskxml | pointhq | porkbun | rackcorp | rackspace | rage4
| rcode0 | regru | scaleway | schlundtech | selectel | selfhost | servercow
| simply | tele3 | tencent | transip | udr | ultra | unoEuro | variomedia |
veesp | vercel | vscale | vultr | websupport
| world4you | yandex | yc | zilore | zone | zonomi>。
```

APIプラグイン名

--data 1行に1つのキーと値のペアを持つファイルで、プラグイン設定に保存するため  
にbase64urlエンコードされます。

DNSプラグインのデータ。(base64エンコード)

--無効 <ブール値

設定を無効にするフラグ。

--ノード <文字列

クラスタ・ノード名のリスト。

--validation-delay <integer> (0 - 172800) (デフォルト = 30)

検証を要求する前に待つ、秒単位の追加遅延。DNS レコードの TTL が長い場合に対応できるようにする。

**pvenode acme plugin config <id> [FORMAT\_OPTIONS].**

ACMEプラグインの設定を取得します。

**<id>: <文字列**

ACMEプラグインインスタンスの一意な識別子。

**pvenode acme プラグイン一覧 [OPTIONS] [FORMAT\_OPTIONS]**

ACMEプラグインインデックス

**--タイプ <dns | standalone>**

特定のタイプのACMEプラグインのみをリストアップする

**pvenode acme プラグイン削除 <id>**

ACMEプラグインの設定を削除します。

**<id>: <文字列**

ACMEプラグインインスタンスの一意な識別子。

**pvenode acme plugin set <id> [OPTIONS] (アクメ・プラグイン・セット <id> [オプション])**

ACMEプラグインの設定を更新します。

**<id>: <文字列**

ACMEプラグインID名

--api <1984hosting | acmedns | acmeproxy | active24 | ad | ali | anx | artfiles | arvan | aurora | autodns | aws | azion | azure | bookmyname | bunny | cf | clouddns | cloudns | cn | conoha | constellix | cpanel | curanet | cyon | da | ddns | desec | df | dgon | dnsexit | dnshome | dnimple | dnsservices | do | doapi | domeneshop | dp | dpi | dpi | dnshome | dnservices | dodesec | df | dgon | dnsexit | dnshome | dnsimple | dnsservices | do | doapi | domeneshop | dp | dpi | dreamhost | duckdns | durabledns | dyn | dynu | dynv6 | easydns | edgedns | euserv | exoscale | fornex | freedns | gandi\_livedns | gcloud | gcore | gd | geoscaling | googledomains | he | hetzner | hexonet | hostingde | huaweicloud | infoblox | infomaniak | internetbs | inwx | ionos | ipv64 | ispconfig | jd | joker | kappernet | kas | kinghost | knot | la | leaseweb | lexicon | linode | linode\_v4 | loopia | lua | maradns | me | miab | misaka | myapi | mydevil | mydnsjp | mythic\_beasts | namecheap | namecom | namesilo | nanelo | nederhost | neodigit | netcup | netlify | nic | njalla | nm | nsd | nsone | nsupdate | nw | oci | one | online | openprovider | openstack | opnsense | ovh | pdns | pleskxml | pointhq | porkbun | rackcorp | rackspace | rage4 | rcode0 | regru | scaleway | schlundtech | selectel | selfhost | servercow | simply | tele3 | tencent | transip | udr | ultra | uno euro | variomedia | veeesp | vercel | vscale | vultr | websupport | world4you | yandex | yc | zilore | zone | zonomi>。

API プラグイン名

--data 1 行に 1 つのキーと値のペアを持つファイルで、プラグイン設定に保存するため  
に base64url エンコードされます。

DNS プラグインのデータ。(base64 エンコード)

--削除 <文字列>

削除したい設定のリスト。

**-digest <文字列>**

現在のコンフィギュレーション・ファイルのダイジェストが異なる場合、変更を防ぐ。これは、現在のコンフィギュレーション・ファイルが変更されないようにするために使用できます。

**--no <ブール値>**

設定を無効にするフラグ。

**--node <文字列>**

クラスタ・ノード名のリスト。

**--validation-delay <integer> (0 - 172800) (デフォルト = 30)**

検証を要求する前に待つ、秒単位の追加遅延。DNS レコードの TTL が長い場合に対応できるようにする。

**pvenode cert delete [<restart>] (pvenode証明書の削除)**

カスタム証明書チェーンとキーを削除する。

**<restart>: <ブール値> (デフォルト = 0)**

pveproxy を再起動します。

**pvenode証明書情報 [FORMAT\_OPTIONS]。**

ノードの証明書に関する情報を取得する。

**pvenode cert set <certificates> [<key>] [OPTIONS] [FORMAT\_OPTIONS].**

カスタム証明書チェーンとキーをアップロードまたは更新します。

**<証明書>: <文字列>**

PEM エンコードされた証明書（チェーン）。

**<キー>: <文字列>**

PEM エンコードされた秘密鍵。

**--force <ブール値> (デフォルト = 0)**

既存のカスタムまたは ACME 証明書ファイルを上書きする。

**--restart <boolean> (デフォルト = 0)**

pveproxy を再起動します。

### **pvenodeコンフィグ取得 [OPTIONS]**

ノードの設定オプションを取得します。

```
--プロパティ <acme | acmedomain0 | acmedomain1 | acmedomain2  
| acmedomain3 | acmedomain4 | acmedomain5 | description |  
startall-onboot-delay | wakeonlan> (デフォルト = all)
```

ノード構成から特定のプロパティのみを返します。

**pvenodeコンフィグセット [OPTIONS]。**

ノード設定オプションを設定します。

**--acme [アカウント=<名前>] [,ドメイン=<ドメイン[;ドメイン;...]>]**]。  
ノード固有のACME設定。

**--acmedomain[n] [domain=<domain> [,alias=<domain>] [,plugin=<プラグイン設定名>]]**。  
ACMEドメインとバリデーション・プラグイン

**--削除 <文字列**

削除したい設定のリスト。

**--説明 <文字列**

ノードの説明。ウェブインターフェースのノードノートパネルに表示されます。これは設定ファイル内にコメントとして保存されます。

**-ダイジェスト <文字列**

現在のコンフィギュレーションファイルのSHA1ダイジェストが異なる場合、変更を防止する。これは同時変更を防ぐために使用できます。

**--startall-onboot-delay <integer> (0 - 300) (デフォルト = 0)**

オンブートを有効にしたすべての仮想ゲストを起動するまでの初期遅延時間（秒）。

**--wakeonlan [mac=<MACアドレス> [,bind-interface=<バインドインターフェイス>] [,broadcast-address=<IPv4プロードキャストアドレス>]]**。  
ノード固有のウェイクオンLAN設定。

**pvenodeヘルプ [オプション]**

指定したコマンドに関するヘルプを表示する。

**--extra-args <array>**

特定のコマンドのヘルプを表示する

**--verbose <ブール値**

冗長な出力形式。

**pvenode migrateall <ターゲット> [オプション]。**

すべてのVMとコンテナを移行する。

**<ターゲット>: <文字列**

ターゲット・ノード

---

**-最大労働者数 <整数> (1 - N)**

並列マイグレーションジョブの最大数。設定されていない場合は、datacenter.cfgの'max\_workers'を使用します。両方を設定する必要があります！

**--vms <文字列>**

これらのIDを持つゲストのみを対象とする。

**--with-local-disks <ブール値>**

ローカルディスクのライブストレージ移行を有効にする

**pvenode startall [OPTIONS]。**

このノードにあるすべてのVMとコンテナ（デフォルトではonboot=1のもののみ）を起動する。

**--force <ブール値> (デフォルト = オフ)**

仮想ゲストのonbootが設定されていない、またはoffに設定されている場合でも、startコマンドを発行します。

**--vms <文字列>**

コンマで区切られたVMIDリストのゲストのみを考慮する。

**pvenode stopall [OPTIONS] (ペノード・ストップオール)**

すべてのVMとコンテナを停止する。

**-強制停止 <ブール値> (デフォルト = 1)**

タイムアウト後にハードストップを強いる。

**--タイムアウト <整数> (0 - 7200) (デフォルト = 180)**

各ゲストシャットダウンタスクのタイムアウト。強制停止に応じて、シャットダウンは単に中断されるか、ハード停止が強制されます。

**--vms <文字列>**

これらのIDを持つゲストのみを対象とする。

**pvenodeタスクリスト [OPTIONS] [FORMAT\_OPTIONS]**

1つのノードのタスクリスト（終了したタスク）を読む。

**--エラー <ブール値> (デフォルト = 0)**

ステータスが ERROR のタスクのみをリストアップする。

**--リミット <整数> (0 - N) (デフォルト = 50)**

この量のタスクだけをリストアップする。

**--以降 <整数>**

このUNIXエポック以降のタスクだけをリストアップする。

**--ソース <アクティブ | すべて | アーカイブ> (デフォルト=アーカイブ)**

アーカイブされたタスク、アクティブなタスク、すべてのタスクをリストアップします。

**--start <integer> (0 - N) (デフォルト=0)**

このオフセットから始まるタスクをリストアップする。

**--ステータスフィルター <文字列>**

返されるべきタスク状態のリスト。

**--タイプフィルター <文字列>**

このタイプのタスク（例：vzstart、vzdump）のみをリストアップする。

**<integer> まで**

このUNIXエポックまでのタスクだけをリストアップする。

**--ユーザーフィルター <文字列>**

このユーザーのタスクのみをリストアップします。

**--vmid <整数> (100 - 999999999)**

このVMのタスクだけをリストアップする。

**pvenode task log <upid> [OPTIONS] (ペノード・タスク・ログ<upid> [オプション])。**

タスクログを読む。

**<upid>: <文字列>**

タスクのユニークID。

**--ダウンロード <ブール値>**

タスクログファイルをダウンロードするかどうか。このパラメータは他のパラメータと併用することはできない。

**--start <integer> (0 - N) (デフォルト=0)**

タスクログを読むときはこの行から始める

**pvenode タスク・ステータス <upid> [FORMAT\_OPTIONS].**

タスクのステータスを読む。

**<upid>: <文字列**

タスクのユニークID。

**pvenode wakeonlan <ノード>**

wake on LANネットワークパケットでノードをウェイクアップさせる。

**<ノード>: <文字列**

ウェイクオンLANパケットのターゲットノード

## A.7 pvesh - Proxmox VE API 用シェルインターフェイス

**pvesh** <COMMAND> [ARGS] [OPTIONS].

**pvesh create** <api\_path> [OPTIONS] [FORMAT\_OPTIONS].

<api\_path>でAPI POSTを呼び出す。

**<api\_path>**: <文字列  
APIパス。

**--noproxy** <ブール値  
自動プロキシを無効にする。

**pvesh delete** <api\_path> [OPTIONS] [FORMAT\_OPTIONS].

<api\_path>でAPI DELETEを呼び出す。

**<api\_path>**: <文字列  
APIパス。

**--noproxy** <ブール値  
自動プロキシを無効にする。

**pvesh get** <api\_path> [OPTIONS] [FORMAT\_OPTIONS].

<api\_path>でAPI GETを呼び出す。

**<api\_path>**: <文字列  
APIパス。

**--noproxy** <ブール値  
自動プロキシを無効にする。

**pvesh help** [OPTIONS]

指定したコマンドに関するヘルプを表示する。

**--extra-args** <array>  
特定のコマンドのヘルプを表示する

**--verbose** <ブール値  
冗長な出力形式。

**pvesh ls** <api\_path> [OPTIONS] [FORMAT\_OPTIONS].

<api\_path>上の子オブジェクトをリストアップする。

**<api\_path>**: <文字列  
APIパス。

**--noproxy <ブール値**  
自動プロキシを無効にする。

**pvesh set <api\_path> [OPTIONS] [FORMAT\_OPTIONS].**

<api\_path>にAPI PUTを呼び出す。

**<api\_path>**: <文字列  
APIパス。

**--noproxy <ブール値**  
自動プロキシを無効にする。

**pvesh usage <api\_path> [OPTIONS] [オプション]。**

<api\_path> のAPI使用情報を表示する。

**<api\_path>**: <文字列  
APIパス。

**--コマンド <create | delete | get | set>.**  
APIコマンド。

**<boolean>を返す**  
返されたデータのスキーマを含む。

**--verbose <ブール値**  
冗長な出力形式。

## A.8 qm - QEMU/KVM 仮想マシンマネージャー

**qm <COMMAND> [ARGS] [OPTIONS].**

### qmエージェント

qmゲストcmdのエイリアス。

**qm cleanup <vmid> <clean-shutdown> <guest-requested>.**

タップデバイスやvgpusなどのリソースをクリーンアップする。vm のシャットダウンやクラッシュなどの後に呼び

出されます。

<vmid>: <整数> (100 - 99999999)

VMの（ユニークな）ID。

**<クリーン・シャットダウン>: <ブール値**  
qemuがクリーンにシャットダウンしたかどうかを示す。

**<guest-requested>: <ブール値**  
シャットダウンがゲストによって要求されたのか、qmpを介して要求されたのかを示す。

**qm clone <vmid> <newid> [オプション]。**

仮想マシン/テンプレートのコピーを作成する。

**<vmid>: <整数> (100 - 99999999)**

VMの（ユニークな）ID。

**<newid>: <整数> (100 - 99999999)**

クローンのVMID。

**--bwlimit <integer> (0 - N) (デフォルト=データセンターまたはストレージ設定からのクローン制限)**

I/Oバンド幅の制限をオーバーライドする（単位：KiB/s）。

**--説明 <文字列**

新しいVMの説明。

**--format <qcow2 | raw | vmdk>.**

ファイルストレージのターゲットフォーマット。フルクローンの場合のみ有効です。

**--full <ブール値**

すべてのディスクの完全コピーを作成する。これは通常のVMをクローンするときに必ず行われる。VMテンプレートでは、デフォルトでリンクされたクローンを作成しようとする。

**--名前 <文字列**

新しいVMの名前を設定する。

**--プール <文字列**

新しいVMを指定されたプールに追加する。

**--スナップネーム <文字列>**

スナップショットの名前。

**--ストレージID**

フルクローンの対象ストレージ。

**--ターゲット <文字列>**

ターゲット・ノード。元のVMが共有ストレージ上にある場合にのみ許可される。

**qm cloudinit dump <vmid> <type>**

自動生成されたcloudinitの設定を取得します。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**<type>: <meta | network | user>。**

コンフィグタイプ。

**qm cloudinit pending <vmid>**

cloudinitコンフィギュレーションを現在の値と保留中の値の両方で取得します。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**qm cloudinit update <vmid>**

cloudinit設定ドライブの再生成と変更。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**qm config <vmid> [OPTIONS] .**

保留中の構成変更を適用した仮想マシン構成を取得します。現在のコンフィギュレーションを取得するには、*current* パラメータを設定します。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**--カレント <ブール値> (デフォルト = 0)**

保留値ではなく）現在の値を取得する。

**--スナップショット <文字列>**

指定されたスナップショットから設定値を取得します。

**qm create <vmid> [OPTIONS] [オプション]。**

仮想マシンを作成またはリストアする。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**--acpi <ブール値> (デフォルト = 1)**

ACPI を有効/無効にする。

**--アフィニティ <文字列>**

ゲスト・プロセスの実行に使用されるホスト・コアのリスト（例： 0,5,8-11

--agent [enabled=<1|0> [,freeze-fs-on-backup=<1|0>] [,fstrim\_cloned\_disks=<1|0>] [,type=<virtio|isa> ]。

QEMU ゲストエージェントとそのプロパティとの通信を有効/無効にします。

--arch <aarch64 | x86\_64>.

仮想プロセッサーのアーキテクチャ。デフォルトはホスト。

--アーカイブ <文字列>

バックアップアーカイブ。.tarまたは.vmaファイルへのファイルシステムのパス（標準入力からデータをパイプする場合は-を使用）、またはproxmoxストレージのバックアップボリューム識別子。

--args <文字列>

kvmに渡される任意の引数。

--audio0 device=<ich9-intel-hda|intel-hda|AC97> [,driver=<spice|none>].

QXL/Spiceと組み合わせて使用すると便利です。

--autostart <boolean> (デフォルト = 0)

クラッシュ後の自動再起動（現在は無視）。

--バルーン <整数> (0 - N)

VMのターゲットRAMの量 (MiB)。ゼロを使用すると、バロン・ドライバが無効になります。

--bios <ovmf | seabios> (デフォルト = seabios)

BIOSの実装を選択します。

--boot [[legacy=]<[acdn]{1,4}>] [order=<デバイス[;デバイス...]>]]。

ゲストの起動順序を指定します。order= サブプロパティを使用します。key または legacy= を指定しない使用法は非推奨です。

--ブートディスク (ide|sata|scsi|virtio)◆d+.

指定したディスクからの起動を有効にする。非推奨：代わりに boot: order=foo;bar を使う。

--bwlimit <integer> (0 - N) (デフォルト = データセンターまたはストレージ設定から

## 制限を復元)

I/O バンド幅の制限をオーバーライドする（単位：KiB/s）。

### --CDROM <ボリューム>

これは -ide2 オプションのエイリアスである。

### --cicustom [meta=<volume>] [,network=<volume>] [,user=<volume>] [,vendor=<volume>].

cloud-init: 開始時に自動生成されるファイルを置き換えるカスタムファイルを指定する。

### --パスワード

cloud-init: ユーザーに割り当てるパスワード。一般的に、これを使うことは推奨されない。代わりに sshキーを使ってください。また、古いバージョンの cloud-init はハッシュ化されたパスワードをサポートしていないことに注意してください。

**--シティープ <configdrive2 | nocloud | opennebula>。**

cloud-initコンフィギュレーション・フォーマットを指定します。デフォルトは、設定されているオペレーティング・システムの種類（ostype.Linux では nocloud 形式、Windows では configdrive2 を使用します。

**-ciupgrade (アップグレード) <ブール値> (デフォルト = 1)**

cloud-init: 初回起動後にパッケージの自動アップグレードを行います。

**-ユーザー <文字列>**

cloud-init: イメージに設定されているデフォルトユーザーの代わりに、sshキーとパスワードを変更するユーザー名。

**--コア数 <整数> (1 - N) (デフォルト = 1)**

ソケットあたりのコア数。

--cpu [[cputype=]<string>] [, フラグ=<+FLAG[-FLAG...]>] を指定します。[,hidden=<1|0>] [,hv-vendor-id=<vendor-id>] を指定する。

[phys-bits=<8-64|host>] [,reported-model=<enum>]。  
エミュレートされたCPUタイプ。

**--cpulimit <number> (0 - 128) (デフォルト = 0)**

CPU使用量の上限。

--cpuunits <integer> (1 - 262144) (デフォルト = cgroup v1: 1024, cgroup v2: 100)  
VM の CPU ウェイトは、cgroup v2 では [1, 10000] にクランプされる。

**--説明 <文字列>**

VM の説明。WebインターフェイスのVMのサマリーに表示される。これはコンフィギュレーション・ファイルのコメントとして保存される。

--efidisk0 [file=]<volume> [,efitype=<2m|4m>]  
[,format=<enum>] [,import-from=<ソースボリューム>] [,pre-

**enrolled-keys=<1|0> [,size=<DiskSize>] .**

EFIバーを格納するディスクを設定する。特別な構文 STORAGE\_ID:SIZE\_IN\_GiB を使用して、以下のようにディスクを割り当てます。

を新しいボリュームにコピーする。SIZE\_IN\_GiBはここでは無視され、代わりにデフォルトのEFIバーがボリュームにコピーされることに注意。既存のボリュームからインポートするには、STORAGE\_ID:0と*import-from*パラメータを使用します。

**--force <ブール値**

既存のVMの上書きを許可する。

---

**注**

必要なオプション: archive

---

**--フリーズ <ブール値**

起動時にCPUをフリーズさせる (c monitorコマンドで実行開始) 。

**--hookscript <文字列>**

vmsのライフタイムのさまざまなステップで実行されるスクリプト。

```
--hostpci[n]  [[host=<HOSTPCIID[,HOSTPCIID2...]>][,device-id=<hex id>][,legacy-igd=<1|0>][,mapping=<mapping-id>][,mdev=<string>][,pcie=<1|0>][,rombar=<1|0>][,romfile=<string>][,sub-device-id=<hex id>][,sub-vendor-id=<hex id>][,vendor-id=<hex id>][,x-vga=<1|0>]]。
```

ホストのPCIデバイスをゲストにマップする。

**--hotplug <string> (デフォルト=ネットワーク,ディスク,USB)**

ホットプラグ機能を選択的に有効にします。ネットワーク、ディスク、CPU、メモリ、USB、*cloudinit*。ホットプラグを完全に無効にするには0を使用します。値として1を使用すると、デフォルトのnetwork,disk,usbのエイリアスになります。USB ホットプラグは、マシンのバージョンが>=のゲストで可能です。

7.1とostype l26またはwindows > 7.

**--hugepages <1024 | 2 | any>.**

ヒュッゲページ・メモリの有効／無効。

```
--ide[n] [file=<volume>[,aio=<native|threads|io_uring>][,backup=<1|0>][,bps=<bps>][,bps_max_length=<seconds>][,bps_rd=<bps>][,bps_rd_max_length=<seconds>][,bps_wr=<bps>][,bps_wr_max_length=<seconds>][,cache=<enum>][,cyls=<integer>][,detect_zeroes=<1|0>][,discard=<ignore|on>][,format=<enum>][,heads=<integer>][,import-from=<ソースボリューム>][,iops=<iops>][,iops_max=<iops>][,iops_max_length=<seconds>][,iops_rd=<iops>][,iops_rd_max=<iops>][,iops_rd_max_length=<seconds>][,iops_wr=<iops>][,iops_wr_max=<iops>][,iops_wr_max_length=<seconds>][,mbps=<mbps>][,mbps_max=<mbps>][,mbps_rd=<mbps>][,mbps_rd_max=<mbps>][,mbps_wr=<mbps>][,mbps_wr_max=<mbps>][,media=<CDROM|DISK>][,model=<model>][,replicate=<1|0>][,rerror=<ignore|report|stop>][,secs=<integer>][,serial=<serial>][,shared=<1|0>][,size=<DiskSize>][,snapshot=<1|0>][,ssd=<1|0>][,trans=<none|lba|auto>][,werror=<enum>][,wwn=<wwn>]
```

となる。

ボリュームをIDEハードディスクまたはCD-ROMとして使用する(nは0~3)。新しいボリュームを割り当てるには、特別な構文 STORAGE\_ID:SIZE\_IN\_GiB を使用する。既存のボリュームからインポートするに

は、STORAGE\_ID:0と*import-from*パラメータを使用する。

```
--ipconfig[n] [gw=<GatewayIPv4>]
[,gw6=<GatewayIPv6>] [,ip=<IPv4Format/CIDR>]
[,ip6=<IPv6Format/CIDR>].
```

cloud-init: 対応するインターフェースのIPアドレスとゲートウェイを指定する。

IPアドレスはCIDR表記を使用し、ゲートウェイはオプションだが、同じタイプのIPを指定する必要がある。

特別な文字列*dhcp*は、DHCPを使用するIPアドレスに使用でき、その場合、明示的な ゲートウェイを指定する必要はない。IPv6の場合、ステートレス自動設定を使用するには、特別な文字列*auto*を使用できる。これにはcloud-init 19.4以降が必要です。

cloud-initが有効で、IPv4アドレスもIPv6アドレスも指定されていない場合、デフォルトでIPv4のdhcpを使用する。

--ivshmem size=<整数> [,name=<文字列>]。

VM間共有メモリ。VM間やホストとの直接通信に便利。

--keephugepages <boolean> (デフォルト = 0)

hugepagesと併用する。有効にすると、VMシャットダウン後もhugepagesは削除されず、その後の起動に使用できます。

--キーボード <da | de | de-ch | en-gb | en-us | es | fi | fr | fr-be | fr-ca | fr-ch | hu | is | it | ja | lt | mk | nl | no | pl | pt | pt-br | sl | sv | tr>

VNCサーバーのキーボードレイアウト。このオプションは一般的には必要なく、ゲストOS内で処理した方が良い場合が多い。

--kvm <論理値> (デフォルト = 1)

KVMハードウェア仮想化の有効/無効。

--ライブ・リストア <ブール値>

バックグラウンドでインポートやリストアを行っている間、すぐにVMを起動する。

--ローカルタイム <ブール値>

リアルタイムクロック (RTC) をローカルタイムに設定する。これはデフォルトで有効である。

Microsoft Windows OS。

--lock <バックアップ | クローン | 作成 | マイグレーション | ロールバック | スナップショット | スナップショット削除 | サスPEND | 一時停止>。

VMをロック/アンロックする。

--マシン [[タイプ=<マシンタイプ>]] [,viommu=<intel|virtio>] ]。

[viommu=<intel|virtio>] である。

QEMUマシンを指定する。

--メモリ [current=<整数>]

メモリ特性。

**--migrate\_downtime <number> (0 - N) (デフォルト = 0.1)**

マイグレーションの最大許容ダウンタイム（秒）を設定します。

**--migrate\_speed <integer> (0 - N) (デフォルト = 0)**

マイグレーションの最大速度（MB/s）を設定する。値0は制限なし。

## **--名前 <文字列>**

VM の名前を設定します。コンフィギュレーション・ウェブ・インターフェイスでのみ使用されます。

## **--ネームサーバー <文字列>**

cloud-init: コンテナの DNS サーバー IP アドレスを設定する。searchdomain も nameserver も設定されていない場合、Create はホストからの設定を自動的に使用する。

```
--net[n] [model=<enum> [,bridge=<bridge>] [,firewall=<1|0>]
[,link_down=<1|0>] [,macaddr=<XX:XX:XX:XX:XX: XX>] [,mtu=<integer>]
[,queues=<integer>] [,rate=<number>] [,tag=<integer>]
[,trunks=<vlanid[;vlanid...]>] [,<model>=<macaddr>] [,<model>=<macaddr>]
[,<model>=<macaddr>
ネットワーク機器を指定する。
```

```
--numa <boolean> (デフォルト = 0)
NUMAの有効/無効。
```

```
--numa[n] cpus=<id[-id];...> [,hostnodes=<id[-id];...>]
を指定します。 [,memory=<number>]
[,policy=<preferred|bind|interleave>] である。
NUMAトポロジー。
```

```
--onboot <boolean> (デフォルト = 0)
システム起動時にVMを起動するかどうかを指定する。
```

```
--ostype <l24 | 126 | other | solaris | w2k | w2k3 | w2k8 | win10 |
win11 | win7 | win8 | vvista | wxp>.
ゲストOSを指定する。
```

```
--parallel[n] /dev/parportd+|/dev/usb/lpd+
ホストパラレルデバイスをマップする (nは0~2) 。
```

```
--プール <文字列>
VMを指定したプールに追加する。
```

```
--プロテクション <ブール値> (デフォルト = 0)
VM の保護フラグを設定する。これにより、VMの削除とディスクの削除操作が無効になる。
```

```
--reboot <boolean> (デフォルト = 1)
再起動を許可する。0に設定すると、リブート時にVMが終了する。
```

```
--rng0 [source=</dev/urandom|/dev/random|/dev/hwrng>
[,max_bytes=<integer>] [,period=<integer>].
VirtIOベースの乱数発生器を設定します。
```

```
--sata[n] [file=<volume> [,aio=<native|threads|io_uring>]
[,backup=<1|0>] [,bps=<bps>] [,bps_max_length=<seconds>]
[,bps_rd=<bps>] [,bps_rd_max_length=<seconds>] [,bps_wr=<bps>]
[,bps_wr_max_length=<seconds>] [,cache=<enum>] [,cyls=<integer>]
[,detect_zeroes=<1|0>] [,discard=<ignore|on>] [,format=<enum>]
[,heads=<integer>] [,import-from=<ソースボリューム>] [,iops=<iops>]
[,iops_max=<iops>] [,iops_max_length=<seconds>] [,iops_rd=<iops>]
[,iops_rd_max=<iops>] [,iops_rd_max_length=<seconds>]
[,iops_wr=<iops>] [,iops_wr_max=<iops>]
[,iops_wr_max_length=<seconds>] [,mbps=<mbps>] [,mbps_max=<mbps>]
[,mbps_rd=<mbps>] [,mbps_rd_max=<mbps>] [,mbps_wr=<mbps>]
[,mbps_wr_max=<mbps>] [,media=<cdrom|disk>] [,replicate=<1|0>]
[,rerror=<ignore|report|stop>] [,secs=<integer>] [,serial=<serial>]
[,shared=<1|0> ]。 [,size=<DiskSize>] [,snapshot=<1|0>]
[,ssd=<1|0>] [,trans=<none|lba|auto>] [,werror=<enum>] [,wwn=<wwn>]
```

とする。

ボリュームをSATAハードディスクまたはCD-ROMとして使用する (nは0～5)。新しいボリュームを割り当てるには、特別な構文 STORAGE\_ID:SIZE\_IN\_GiB を使用する。既存のボリュームからインポートするには、STORAGE\_ID:0と*import-from*パラメータを使用します。

```
--scsi[n] [[file=<volume> [,aio=<native|threads|io_uring>>]
[,backup=<1|0>] [,bps=<bps>] [,bps_max_length=<seconds>]
[,bps_rd=<bps>] [,bps_rd_max_length=<seconds>] [,bps_wr=<bps>]
[,bps_wr_max_length=<seconds>] [,cache=<enum>] [,cyls=<integer>]
[,detect_zeroes=<1|0>] [,discard=<ignore|on>] [,format=<enum>]
[,heads=<integer>] [,import-from=<ソースボリューム>] [,iops=<iops>]
[,iops_max=<iops>] [,iops_max_length=<seconds>] [,iops_rd=<iops>]
[,iops_rd_max=<iops>] [,iops_rd_max_length=<seconds>]
[,iops_wr=<iops>] [,iops_wr_max=<iops>]
[,iops_wr_max_length=<seconds>] [,iothread=<1|0>] [,mbps=<mbps>]
[,mbps_max=<mbps>] [,mbps_rd=<mbps>] [,mbps_rd_max=<mbps>]
[,mbps_wr=<mbps>] [,mbps_wr_max=<mbps>] [,media=<cdrom|disk>]
[,product=<product>] [,queues=<integer>] [,replicate=<1|0>]
[,rerror=<ignore|report|stop>] [,ro=<1|0>] [,scsiblock=<1|0>]
[,secs=<integer>] [,serial=<serial>] [,shared=<1|0> ]。
[,size=<DiskSize>] [,snapshot=<1|0>] [,ssd=<1|0>]
[,trans=<none|lba|auto>] [,vendor=<vendor>] [,werror=<enum>]
[,wwn=<wwn> ]。
```

ボリュームをSCSIハードディスクまたはCD-ROMとして使用する (nは0から30)。新しいボリュームを割り当てるには、特別な構文 STORAGE\_ID:SIZE\_IN\_GiB を使用する。新しいボリュームを割り当てるには、STORAGE\_ID:0と*import-from*パラメータを使用する。

を設定する。

```
--scsihw <lsi | lsi53c810 | megasas | pvscsi | virtio-scsi-pci  
| virtio-scsi-single> (デフォルト=lsi)
```

SCSIコントローラモデル

#### --検索ドメイン <文字列>

cloud-init: コンテナのDNS検索ドメインを設定する。searchdomainもnameserverも設定されていない場合、Createはホストからの設定を自動的に使用する。

**-シリアル[n] (/dev/.+|socket)**

VM内にシリアル・デバイスを作成する (nは0~3)

**--shares <integer> (0 - 50000) (デフォルト = 1000)**

オート・バルーニングのメモリ・シェア量。この数値が大きいほど、このVMはより多くのメモリを獲得する。数値は、他のすべての実行中のVMの重みに対する相対値です。ゼロを使用すると、オート・バルーニングは無効になる。オート・バルーニングはpvestatdによって実行される。

**--smbios1 [base64=<1|0>] [,family=<Base64エンコードされた文字列>]**

[,manufacturer=<Base64エンコードされた文字列>] [,product=<Base64エンコードされた文字列>] [,serial=<Base64エンコードされた文字列>] [,sku=<Base64エンコードされた文字列>] [,uuid=<UUID>] [,version=<Base64エンコードされた文字列>] となります。

SMBIOSタイプ1のフィールドを指定する。

**--smp <整数> (1 - N) (デフォルト = 1)**

CPU数。代わりに -sockets オプションを使用してください。

**--ソケット <整数> (1 - N) (デフォルト = 1)**

CPUソケットの数。

**--spice\_enhancements [foldersharing=<1|0>]**

[,videostreaming=<off|all|filter>]。

SPICE の追加機能拡張を設定する。

**--sshkeys <ファイルパス**

cloud-init: 公開SSH鍵を設定する (1行に1つの鍵、OpenSSH形式)。

**--start <boolean> (デフォルト = 0)**

VMが正常に作成された後、VMを起動する。

**--startdate (now | YYYY-MM-DD | YYYY-MM-DDTHH:MM:SS) (default = now)**

リアルタイムクロックの初期日付を設定する。有効な日付の書式は、'now' または 2006-06-17T16:01:21 または

2006-06-17.

**--startup`[[order=]\d+] [,up=d+] [,down=d+]`.**

スタートアップとシャットダウンの動作。Orderは、一般的な起動順序を定義する非負の数値である。

シャットダウンは逆の順序で行われます。さらに、次のVMが起動または停止するまでの待ち時間を秒単位で設定できます。

**--ストレージID**

デフォルトのストレージ。

**--タブレット <ブール値> (デフォルト = 1)**

USBタブレットデバイスを有効/無効にします。

**--タグ <文字列**

VMのタグ。これはメタ情報に過ぎない。

--tdf <ブール値> (デフォルト = 0)

タイムドリフト修正の有効／無効。

--テンプレート <ブール値> (デフォルト = 0)

テンプレートの有効/無効。

--tpmstate0 [ファイル=<ボリューム> [,インポート元=<ソースボリューム>] [,サイズ=<ディスクサイズ>] [,バージョン=<v1.2|v2.0>]。

TPMの状態を保存するDiskを設定する。フォーマットはrawに固定される。新しいボリュームを割り当てるには、特別な構文STOR-AGE\_ID:SIZE\_IN\_GiBを使用する。SIZE\_IN\_GiBはここでは無視され、代わりに4MiBが使用されることに注意。既存のボリュームからインポートするには、STORAGE\_ID:0とimport-fromパラメータを使用します。

--unique <ブール値>

一意のランダムなイーサネットアドレスを割り当てる。

---

## 注

必要なオプション: archive

---

--unused[n] [file=<volume>]

未使用ボリュームへの参照。これは内部的に使用されるので、手動で変更してはならない。

--usb[n] [[host=<HOSTUSBDEVICE|spice>] [,mapping=<mappingid>]

[,usb3=<1|0[,マッピング=<マッピングID>] [,usb3=<1|0]

USBデバイスを設定する (nは0~4、マシンバージョン7.1以上、ostype l26またはwindows 7以上の場合、nは14まで)。

--vcpus <整数> (1 - N) (デフォルト = 0)

ホットプラグされたVCPUの数。

--vga [[type=<enum>] [,クリップボード=<vnc[,clipboard=<vnc>]

[,memory=<integer>]] とする。

VGAハードウェアを設定する。

---

```
--virtio[n] [file=<volume> [,aio=<native|threads|io_uring>]
[,backup=<1|0>] [,bps=<bps>] [,bps_max_length=<seconds>
[,bps_rd=<bps>] [,bps_rd_max_length=<seconds>] [,bps_wr=<bps>]
[,bps_wr_max_length=<seconds>] [,cache=<enum>] [,cyls=<integer>]
[,detect_zeroes=<1|0>] [,discard=<ignore|on>] [,format=<enum>]
[,heads=<integer>] [,import-from=<ソースボリューム>] [,iops=<iops>]
[,iops_max=<iops>] [,iops_max_length=<seconds>] [,iops_rd=<iops>]
[,iops_rd_max=<iops>] [,iops_rd_max_length=<seconds>]
[,iops_wr=<iops>] [,iops_wr_max=<iops>]
[,iops_wr_max_length=<seconds>] [,iothread=<1|0>] [,mbps=<mbps>]
[,mbps_max=<mbps>] [,mbps_rd=<mbps>] [,mbps_rd_max=<mbps>]
[,mbps_wr=<mbps>] [,mbps_wr_max=<mbps>] [,media=<cdrom|disk>]
[,replicate=<1|0>] [,rerror=<ignore|report|stop>] [,ro=<1|0>]
[,secs=<integer> ]。] [,serial=<serial>] [,shared=<1|0>]
[,size=<DiskSize>] [,snapshot=<1|0>] [,trans=<none|lba|auto>]
[,werror=<enum> ]。
```

ボリュームをVIRTIOハードディスクとして使用する (nは0~15)。新しいボリュームを割り当てるには、特別な構文 STORAGE\_ID:SIZE\_IN\_GiB を使用する。既存のボリュームからインポートするには、STORAGE\_ID:0と*import-from*パラメータを使用します。

#### --vmgenid <UUID> (デフォルト = 1 (自動生成))

VM生成IDを設定する。作成または更新時に自動生成する場合は 1 を使用し、明示的に無効にする場合は 0 を渡します。

#### --vmstatestorage <ストレージID>.

VM状態のボリューム/ファイルのデフォルトストレージ。

#### --watchdog [[model=<i6300esb|ib700>] [アクション]

仮想ハードウェア・ウォッチドッグ・デバイスを作成する。

**qm delsnapshot** <vmid> <snapname> [OPTIONS]。

VMスナップショットを削除する。

#### <vmid>: <整数> (100 - 999999999)

VMの（ユニークな）ID。

#### <スナップ名>: <文字列

スナップショットの名前。

**--force <ブール値>**

ディスクスナップショットの削除に失敗しても、設定ファイルから削除できる。

**qm destroy <vmid> [OPTIONS]。**

VM とすべての使用済み/所有ボリュームを破棄します。VM固有のパーミッションとファイアウォールルールを削除します。

**<vmid>: <整数> (100 - 99999999)**

VMの（ユニークな）ID。

**--destroy-unreferenced-disks <boolean> (デフォルト = 0)**

設定されている場合、すべての有効なストレージから、設定に参照されていないが、VMIDが一致するすべてのディスクを追加で破壊する。

**--ページ <ブール値**

バックアップ&レプリケーションジョブやHAなどの構成からVMIDを削除する。

**--スキップロック <ブール値**

Ignore locks - rootのみがこのオプションを使用できる。

**qm disk import <vmid> <source> <storage> [OPTIONS].**

VM の未使用ディスクとして外部ディスクリメージをインポート。イメージフォーマットは qemu-img(1) でサポートされていなければならない。

**<vmid>: <整数> (100 - 999999999)**

VM の (ユニークな) ID。

**<ソース>: <文字列**

インポートするディスクリメージへのパス

**<ストレージ>: <ストレージID**

対象ストレージID

**--format <qcow2 | raw | vmdk>.**

対象フォーマット

**qm disk move <vmid> <disk> [<storage>] [OPTIONS].**

ボリュームを別のストレージまたは別のVMに移動する。

**<vmid>: <整数> (100 - 999999999)**

VM の (ユニークな) ID。



<ディスク>: <efidisk0 | ide0 | ide1 | ide2 | ide3 | sata0 | sata1 |  
sata2 | sata3 | sata4 | sata5 | scsi0 | scsi1 | scsi10 | scsi11 |  
scsi12 | scsi13 | scsi14 | scsi15 | SCSI16 | SCSI17 | SCSI18 | SCSI19 |  
SCSI2 | SCSI20 | SCSI21 | SCSI22 | SCSI23 | SCSI24 | SCSI25 | SCSI26 | SCSI27  
| SCSI28 | SCSI29 | SCSI3 | SCSI30 | SCSI4  
| SCSI5 | SCSI6 | SCSI7 | SCSI8 | SCSI9 | TPMSTATE0 | UNUSED0 | UNUSED1 |  
UNUSED10 | UNUSED100 | UNUSED101 | UNUSED102 | UNUSED103  
| 未使用104 | 未使用105 | 未使用106 | 未使用107 | 未使用108 | 未使用109 | 未使用11 | 未  
使用110 | 未使用111 | 未使用112 | 未使用113 | 未使用114 | 未使用115 | 未使用116 | 未使用  
117 | 未使用118 | 未使用119 | 未使用12 | 未使用120 | 未使用121 | 未使用122 | 未使用123 |  
未使用124 | 未使用125 | 未使用126 | 未使用127 | 未使用未使用128 | 未使用129 | 未使用  
13 | 未使用130 | 未使用131 | 未使用132 | 未使用133 | 未使用134 | 未使用135 | 未使用  
136 | 未使用137 | 未使用138 | 未使用139 | 未使用14 | 未使用140 | 未使用141 | 未使用  
142 | 未使用143 | 未使用144 | 未使用145 | 未使用146 | 未使用147 | 未使用148 | 未使  
用149 | 未使用15 | 未使用150 | 未使用151 | unused152 | unused153 |  
unused154 | unused155 | unused156 | unused157 | unused158 |  
unused159 | unused16 | unused160 | unused161 | unused162 |  
unused163 | unused164 | unused165 | unused166 | unused167 |  
unused168 | unused169 | unused17 | unused170 | unused171 |  
unused172 | unused173 | unused174 | unused175 | 未使用176 | 未使用177 |  
未使用178 | 未使用179 | 未使用18 | 未使用180 | 未使用181 | 未使用182 | 未使用183 |  
未使用184 | 未使用185 | 未使用186 | 未使用187 | 未使用188 | 未使用189 | 未使用19 |  
未使用190 | 未使用191 | 未使用192 | 未使用193 | 未使用194 | 未使用195 | 未使用196  
| 未使用197 | 未使用198 | 未使用199 | 未使用2  
| 未使用20 | 未使用200 | 未使用201 | 未使用202 | 未使用203 | 未使用204 | 未使用  
205 | 未使用206 | 未使用207 | 未使用208 | 未使用209 | 未使用21 | 未使用210 | 未使用  
211 | 未使用212 | 未使用213 | 未使用214 | 未使用215 | 未使用216 | 未使用217 | 未使  
用218 | 未使用219 | 未使用22 | 未使用220 | 未使用221 | 未使用222 | 未使用223未使用  
224 | 未使用225 | 未使用226 | 未使用227 | 未使用228 | 未使用229 | 未使用23 | 未使用  
230 | 未使用231 | 未使用232 | 未使用233 | 未使用234 | 未使用235 | 未使用236 | 未使  
用237 | 未使用238 | 未使用239 | 未使用241 | 未使用242 | 未使用243 | 未使用244 | 未  
使用245 | 未使用246 | 未使用247 | 未使用248 | 未使用249 | 未使用25 | 未使用250 | 未  
使用251 | 未使用252 | 未使用253 | 未使用254 | 未使用255 | 未使用26 | 未使用27 | 未使用

28|未使用29|未使用3|未使用30|未使用31|未使用32|未使用33|未使用34|未使用35|  
未使用36|未使用37|未使用38|未使用39|未使用4|未使用40|未使用41|未使用42|未使  
用43|未使用44|未使用45|未使用未使用46|未使用47|未使用48|未使用49|未使用5|未  
使用50|未使用51|未使用52|未使用53|未使用54|未使用55|未使用56|未使用57|未使  
用58|未使用59|未使用6|未使用60|未使用61|未使用62|未使用63|未使用64|未使用  
65|未使用66|未使用67|未使用68|未使用69|未使用7|未使用70|未使用71|未使用72|  
未使用

未使用73|未使用74|未使用75|未使用76|未使用77|未使用78|

---

~~未使用99|未使用0|未使用90|未使用91|未使用92|未使用93|未使用84|未使用85|未使用  
86|未使用87|未使用88|未使用89|未使用~~

移動したいディスク。

**<ストレージ>: <ストレージID**  
ターゲット・ストレージ。

**--bwlimit <integer> (0 - N) (デフォルト = データセンターまたはストレージ設定**

**から制限を移動)**

I/O バンド幅の制限をオーバーライドする (単位: KiB/s)。

**--delete <boolean> (デフォルト = 0)**

コピー成功後、元のディスクを削除する。デフォルトでは、元のディスクは未使用ディスクとして保持されます。

**-ダイジェスト <文字列>**

現在のコンフィギュレーション・ファイルのSHA1" . "ダイジェストが異なる場合、変更を防ぐ。これは、同時変更を防ぐために使用できます。

**--format <qcow2 | raw | vmdk>.**

ターゲット・フォーマット

**-ターゲット・ダイジェスト <文字列>**

ターゲットVMの現在のコンフィグファイルのSHA1ダイジェストが" . "異なる場合、変更を防止する。

これは同時変更を検出するために使用できる。





28 | 未使用 29 | 未使用 30 | 未使用 31 | 未使用 32 | 未使用 33 | 未使用 34 | 未使用 35 |  
未使用 36 | 未使用 37 | 未使用 38 | 未使用 39 | 未使用 4 | 未使用 40 | 未使用 41 | 未使用 42 | 未使  
用 43 | 未使用 44 | 未使用 45 | 未使用 未使用 46 | 未使用 47 | 未使用 48 | 未使用 49 | 未使用 5 | 未  
使用 50 | 未使用 51 | 未使用 52 | 未使用 53 | 未使用 54 | 未使用 55 | 未使用 56 | 未使用 57 | 未使  
用 58 | 未使用 59 | 未使用 6 | 未使用 60 | 未使用 61 | 未使用 62 | 未使用 63 | 未使用 64 | 未使用  
65 | 未使用 66 | 未使用 67 | 未使用 68 | 未使用 69 | 未使用 7 | 未使用 70 | 未使用 71 | 未使用 72 |  
未使用

未使用 73 | 未使用 74 | 未使用 75 | 未使用 76 | 未使用 77 | 未使用 78 |

---

~~未使用 99 | 未使用 9 | 未使用 90 | 未使用 91 | 未使用 92 | 未使用 93 | 未使用 84 | 未使用 85 | 未使用  
86 | 未使用 87 | 未使用 88 | 未使用 89 | 未使用~~

ターゲット VM 上でディスクが移動されるコンフィグ・キー (例えば ide0 や scsi1 など)。デフォルトはソース・ディスク・キーです。

**--ターゲット-vmid <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**qm disk rescan [OPTIONS] (ディスク再スキャン)**

すべてのストレージを再スキャンし、ディスクサイズと未使用ディスクイメージを更新する。

**--ドライラン <ブール値> (デフォルト = 0)**

実際にVMコンフィグに変更を書き出さない。

**--vmid <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**qm disk resize <vmid> <disk> <size> [OPTIONS].**

ボリュームサイズを拡張する。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**<ディスク>: <efidisk0 | ide0 | ide1 | ide2 | ide3 | sata0 | sata1 | sata2 | sata3 | sata4 | sata5 | scsi0 | scsi1 | scsi10 | scsi11 | scsi12 | scsi13 | scsi14 | scsi15 | SCSI16 | SCSI17 | SCSI18 | SCSI19 | SCSI2 | SCSI20 | SCSI21 | SCSI22 | SCSI23 | SCSI24 | SCSI25 | SCSI26 | SCSI27 | SCSI28 | SCSI29 | SCSI3 | SCSI30 | SCSI4 | scsi5 | scsi6 | scsi7 | scsi8 | scsi9 | tpmstate0 | virtio0 | virtio1 | virtio10 | virtio11 | virtio12 | virtio13 | virtio14 | virtio15 | virtio2 | virtio3 | virtio4 | virtio5 | virtio6 | virtio7 | virtio8 | virtio9>。**

リサイズしたいディスク。

**<サイズ>です: \+?\d+(\.\d+)? [KMGT] ?**

新しいサイズ。記号を付けると、その値はボリュームの実際のサイズに加算され、付けない場合は絶対値となります。ディスク・サイズの縮小はサポートされていません。

### -ダイジェスト <文字列>

現在のコンフィギュレーションファイルのSHA1ダイジェストが異なる場合、変更を防止する。これは同時変更を防ぐために使用できます。

### --スキップロック <ブール値>

Ignore locks - rootのみがこのオプションを使用できる。

**qm disk unlink** <vmid> --idlist <string> [OPTIONS].

ディスクイメージのリンク解除/削除。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**--force <ブール値>**

物理的な削除を強制する。これがないと、単純に設定ファイルからディスクを削除し、ボリュームIDを含む`unused[n]`という追加の設定エントリを作成します。`unused[n]`のリンク解除は、常に物理的な削除を引き起こします。

**--idlist <文字列>**

削除したいディスクIDのリスト。

**qm guest cmd <vmid> <command>**

QEMU Guest Agentのコマンドを実行します。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**<コマンド>: <fsfreeze-freeze | fsfreeze-status | fsfreeze-thaw | fstrim | get-fsinfo | get-host-name | get-memory-block-info | get-memory-blocks | get-osinfo | get-time | get-timezone | get-users | get-vcpus | info | network-get-interfaces | ping | shutdown | suspend-disk | suspend-hybrid | suspend-ram>。**

QGAコマンド。

**qm guest exec <vmid> [<extra-args>] [OPTIONS].**

ゲストエージェント経由で指定されたコマンドを実行する

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**<extra-args>: <配列>。**

配列としての追加引数

**--pass-stdin <boolean> (デフォルト = 0)**

設定されると、EOFまでSTDINを読み込み、*input-data*経由でゲストエージェントに転送します(通常、ゲストエージェントが起動するプロセスへのSTDINとして扱われます)。最大1MiBを許可します。

◦

#### --同期 <ブール値> (デフォルト = 1)

offに設定すると、コマンドの終了やタイムアウトを待たずに、即座にpidを返す。

#### --タイムアウト <整数> (0 - N) (デフォルト = 30)

コマンドが終了するまで同期的に待つ最大時間。到達するとpidが返される。無効にするには0を設定する。

**qm guest exec-status <vmid> <pid>.**

ゲストエージェントによって開始された指定された pid のステータスを取得する

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**<pid>: <整数>**

問い合わせるPID

**qm guest passwd <vmid> <username> [OPTIONS].**

指定されたユーザーのパスワードを指定されたパスワードに設定する

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**<ユーザー名>: <文字列>**

パスワードを設定するユーザー。

**--暗号化 <ブール値> (デフォルト = 0)**

パスワードがすでに crypt() を通過している場合は 1 を設定する。

**qm help [OPTIONS]**

指定したコマンドに関するヘルプを表示する。

**--extra-args <array>**

特定のコマンドのヘルプを表示する

**--verbose <ブール値>**

冗長な出力形式。

**qm import <vmid> <source> --storage <string> [OPTIONS].**

ESXi ストレージなど、サポートされているインポートソースから外部仮想ゲストをインポートします。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**<ソース>: <文字列**

インポート元のボリュームID。

**--acpi <ブール値> (デフォルト = 1)**

ACPI を有効/無効にする。

**--アフィニティ <文字列**

ゲスト・プロセスの実行に使用されるホスト・コアのリスト（例：0,5,8-11

--agent [enabled=<1|0> [,freeze-fs-on-backup=<1|0>] [,fstrim\_cloned\_disks=<1|0>]  
[,type=<virtio|isa> ]。

QEMU ゲストエージェントとそのプロパティとの通信を有効/無効にします。

--arch <aarch64 | x86\_64>.

仮想プロセッサーのアーキテクチャ。デフォルトはホスト。

--args <文字列

kvmに渡される任意の引数。

--audio0 device=<ich9-intel-hda|intel-hda|AC97>  
[,driver=<spice|none>].

QXL/Spiceと組み合わせて使用すると便利です。

--autostart <boolean> (デフォルト = 0)

クラッシュ後の自動再起動（現在は無視）。

--バルーン <整数> (0 - N)

VMのターゲットRAMの量 (MiB)。ゼロを使用すると、バロン・ドライバが無効になります。

--bios <ovmf | seabios> (デフォルト = seabios)

BIOSの実装を選択します。

--boot [[legacy=]<[acdn]{1,4}>] [order=<デバイス[,デバイス...]>]]を指定します。

ゲストの起動順序を指定します。order= サブプロパティを使用します。key または legacy= を指定しない使用法は非推奨です。

--ブートディスク (ide|sata|scsi|virtio)◆d+.

指定したディスクからの起動を有効にする。非推奨：代わりに boot: order=foo;bar を使う。

--CDROM <ボリューム

これはオプション -ide2 のエイリアスである。

--cicustom [meta=<volume>] [,network=<volume>] [,user=<volume>]  
[,vendor=<volume>].

cloud-init: 開始時に自動生成されるファイルを置き換えるカスタムファイルを指定する。

-パスワード <文字列

cloud-init: ユーザーに割り当てるパスワード。一般的に、これを使うことは推奨されない。代わりにsshキーを使ってください。また、cloud-initの古いバージョンではハッシュ化されたパスワードをサポートしていないことに注意してください。

**--シティープ <configdrive2 | nocloud | opennebula>。**

cloud-initコンフィギュレーション・フォーマットを指定します。デフォルトは、設定されているオペレーティング・システムの種類（ostype.Linuxではnocloud形式、Windowsではconfigdrive2を使用します。

**-ciupgrade (アップグレード) <ブール値> (デフォルト = 1)**

cloud-init: 初回起動後にパッケージの自動アップグレードを行います。

## -ユーザー <文字列>

cloud-init: イメージに設定されているデフォルトユーザーの代わりに、sshキーとパスワードを変更するユーザー名。

## --コア数 <整数> (1 - N) (デフォルト = 1)

ソケットあたりのコア数。

--cpu [[cputype=<string>] [, フラグ=<+FLAG|;-FLAG...>] ]を指定します。[,hidden=<1|0>] [,hv-vendor-id=<vendor-id>] を指定する。

[phys-bits=<8-64|host>] [,reported-model=<enum>]。

エミュレートされたCPUタイプ。

## --cpulimit <number> (0 - 128) (デフォルト = 0)

CPU使用量の上限。

--cpuunits <integer> (1 - 262144) (デフォルト = cgroup v1: 1024, cgroup v2: 100)

VM の CPU ウェイトは、cgroup v2 では [1, 10000] にクランプされる。

## --削除 <文字列>

削除したい設定のリスト。

## --説明 <文字列>

VM の説明。WebインターフェイスのVMのサマリーに表示される。これはコンフィギュレーション・ファイルのコメントとして保存される。

## --ドライラン <ブール値> (デフォルト = 0)

createコマンドを表示し、何もせずに終了する。

--efidisk0 [file=<volume> [,efitype=<2m|4m>] [,format=<enum>] [,pre-enrolled-keys=<1|0>] [,size=<DiskSize>].

EFIバーを保存するディスクを設定する。

--format <qcow2 | raw | vmdk>.

## 対象フォーマット

### --フリーズ <ブール値>

起動時にCPUをフリーズさせる (c monitorコマンドで実行開始)。

### --hookscript <文字列>

vmsのライフタイムのさまざまなステップで実行されるスクリプト。

```
--hostpci[n] [[host=<HOSTPCIID[;HOSTPCIID2...]>][,device-id=<hex id>][,legacy-igd=<1|0>][,mapping=<mapping-id>][,mdev=<string>][,pcie=<1|0>][,rombar=<1|0>][,romfile=<string>][,sub-device-id=<hex id>][,sub-vendor-id=<hex id>][,vendor-id=<hex id>][,x-vga=<1|0>]。
```

ホストのPCIデバイスをゲストにマップする。

--hotplug <string> (デフォルト=ネットワーク、ディスク、USB)

ホットプラグ機能を選択的に有効にします。ネットワーク、ディスク、CPU、メモリ、USB、*cloudinit*。ホットプラグを完全に無効にするには0を使用します。値として1を使用すると、デフォルトのnetwork,disk,usbのエイリアスになります。USB ホットプラグは、マシンのバージョンが>=のゲストで可能です。

7.1とostype l26またはwindows > 7.

--hugepages <1024 | 2 | any>.

ヒュッゲページ・メモリの有効／無効。

--ide[n] [file=<volume> [,aio=<native|threads|io\_uring>]  
[,backup=<1|0>] [,bps=<bps>] [,bps\_max\_length=<seconds>]  
[,bps\_rd=<bps>] [,bps\_rd\_max\_length=<seconds>] [,bps\_wr=<bps>]  
[,bps\_wr\_max\_length=<seconds>] [,cache=<enum>] [,cyls=<integer>]  
[,detect\_zeroes=<1|0>] [,discard=<ignore|on>] [,format=<enum>]  
[,heads=<integer>] [,iops=<iops>] [,iops\_max=<iops>]  
[,iops\_max\_length=<seconds>] [,iops\_rd=<iops>]  
[,iops\_rd\_max=<iops>] [,iops\_rd\_max\_length=<秒>] [,iops\_wr=<iops>]  
[,iops\_wr\_max=<iops>] [,iops\_wr\_max\_length=<seconds>]  
[,mbps=<mbps>] [,mbps\_max=<mbps>] [,mbps\_rd=<mbps>]  
[,mbps\_rd\_max=<mbps>] [,mbps\_wr=<mbps>] [,mbps\_wr\_max=<mbps>]  
[,media=<cdrom|disk>] [,model=<model>] [,replicate=<1|0>]  
[,rerror=<ignore|report|stop>] [,secs=<integer>]  
[,serial=<serial>] [,shared=<1|0>] [,size=<DiskSize>]  
[,size=<DiskSize>] [,snapshot=<1|0>] [,ssd=<1|0>]  
[,trans=<none|lba|auto>] [,werror=<enum>] [,wwn=<wwn>] とする。

ボリュームをIDEハードディスクまたはCD-ROMとして使用する (nは0~3)。

--ipconfig[n] [gw=<GatewayIPv4>]  
[,gw6=<GatewayIPv6>] [,ip=<IPv4Format/CIDR>]  
[,ip6=<IPv6Format/CIDR>].

cloud-init: 対応するインターフェースのIPアドレスとゲートウェイを指定する。

IPアドレスはCIDR表記を使用し、ゲートウェイはオプションだが、同じタイプのIPを指定する必要がある。

特別な文字列*dhcp*は、DHCPを使用するIPアドレスに使用でき、その場合、明示的な ゲートウェイを指定する必要はない。IPv6の場合、ステートレス自動設定を使用するには、特別な文字列*auto*を使用できる。これにはcloud-init 19.4以降が必要です。

cloud-initが有効で、IPv4アドレスもIPv6アドレスも指定されていない場合、デフォルトでIPv4の*dhcp*を使用する。

**--ivshmem size=<整数> [,name=<文字列>]。**

VM間共有メモリ。VM間やホストとの直接通信に便利。

**--keephugepages <boolean> (デフォルト = 0)**

hugepagesと併用する。有効にすると、VMシャットダウン後もhugepagesは削除されず、その後の起動に使用できます。

**--キーボード <da | de | de-ch | en-gb | en-us | es | fi | fr | fr-be | フランス語 | スペイン語 | フランス語 | フランス語 | 中国語 | 胡語 | is | it | ja | lt | mk | nl | no | pl | pt | pt-br | sl | sv | tr>。**

VNCサーバーのキーボードレイアウト。このオプションは一般的には必要なく、ゲストOS内で処理した方が良い場合が多い。

#### --kvm <論理値> (デフォルト = 1)

KVMハードウェア仮想化の有効/無効。

#### --ライブ・インポート <ブール値> (デフォルト = 0)

すぐにVMを起動し、バックグラウンドでデータをコピーする。

#### --ローカルタイム <ブール値>

リアルタイムクロック (RTC) をローカルタイムに設定する。これはデフォルトで有効である。

Microsoft Windows OS。

#### --lock <バックアップ | クローン | 作成 | マイグレーション | ロールバック | スナップショット | スナップショット削除 | サスPEND | 一時停止>。

VMをロック/アンロックする。

#### --マシン [[タイプ=]<マシンタイプ>] [,viommu=<intel|virtio>] ]。

[viommu=<intel|virtio>] である。

QEMUマシンを指定する。

#### --メモリ [current=]<整数>

メモリ特性。

#### --migrate\_downtime <number> (0 - N) (デフォルト = 0.1)

マイグレーションの最大許容ダウンタイム (秒) を設定します。

#### --migrate\_speed <integer> (0 - N) (デフォルト = 0)

マイグレーションの最大速度 (MB/s) を設定する。値0は制限なし。

#### --名前 <文字列>

VMの名前を設定します。コンフィギュレーション・ウェブ・インターフェイスでのみ使用されます。

#### --ネームサーバー <文字列>

cloud-init: コンテナの DNS サーバー IP アドレスを設定する。searchdomain も nameserver も設定されていない場合、Create はホストからの設定を自動的に使用する。

```
--net[n] [model=<enum> [,bridge=<bridge>] [,firewall=<1|0>]  
[,link_down=<1|0>] [,macaddr=<XX:XX:XX:XX:XX: XX>] [,mtu=<integer>]  
[,queues=<integer>] [,rate=<number>] [,tag=<integer>]  
[,trunks=<vlanid[;vlanid...]>] [,<model>=<macaddr>] [,<model>=<macaddr>]  
[,<model>=<macaddr>
```

ネットワーク機器を指定する。

```
--numa <boolean> (デフォルト = 0)
```

NUMAの有効/無効。

```
--numa[n] cpus=<id[-id];...> [,hostnodes=<id[-id];...>]
```

を指定します。 [,memory=<number>]

```
[,policy=<preferred|bind|interleave>] である。
```

NUMA トポロジー。

--onboot <boolean> (デフォルト = 0)

システム起動時にVMを起動するかどうかを指定する。

--ostype <l24 | 126 | other | solaris | w2k | w2k3 | w2k8 | win10 | win11 | win7 | win8 | vvista | wxp>.

ゲストOSを指定する。

--parallel[n] /dev/parportd+|/dev/usb/lpd+

ホストパラレルデバイスをマップする (nは0~2)。

--プロテクション <ブール値> (デフォルト = 0)

VMの保護フラグを設定する。これにより、VMの削除とディスクの削除操作が無効になる。

--reboot <boolean> (デフォルト = 1)

再起動を許可する。0に設定すると、リブート時にVMが終了する。

--rng0 [source=</dev/urandom|/dev/random|/dev/hwrng>

[,max\_bytes=<integer>] [,period=<integer>].

VirtIO ベースの乱数ジェネレータを設定します。

--sata[n] [file=<volume> [,aio=<native|threads|io\_uring>>]  
[,backup=<1|0>] [,bps=<bps>] [,bps\_max\_length=<seconds>  
[,bps\_rd=<bps>] [,bps\_rd\_max\_length=<seconds>] [,bps\_wr=<bps>]  
[,bps\_wr\_max\_length=<seconds>] [,cache=<enum>] [,cyls=<integer>]  
[,detect\_zeroes=<1|0>] [,discard=<ignore|on>] [,format=<enum>]  
[,heads=<integer>] [,iops=<iops>] [,iops\_max=<iops>]  
[,iops\_max\_length=<seconds>] [,iops\_rd=<iops>]  
[,iops\_rd\_max=<iops>] [,iops\_rd\_max\_length=<秒>] [,iops\_wr=<iops>]  
[,iops\_wr\_max=<iops>] [,iops\_wr\_max\_length=<seconds>]  
[,mbps=<mbps>] [,mbps\_max=<mbps>] [,mbps\_rd=<mbps>]  
[,mbps\_rd\_max=<mbps>] [,mbps\_wr=<mbps>] [,mbps\_wr\_max=<mbps>]  
[,media=<cdrom|disk> ]。] [,replicate=<1|0>]  
[,rerror=<ignore|report|stop>] [,secs=<integer>]  
[,serial=<serial>] [,shared=<1|0>] [,size=<DiskSize>]  
[,snapshot=<1|0>] [,ssd=<1|0>] [,trans=<none|lba|auto>]  
[,werror=<enum>] [,wwn=<wwn>] とする。

ボリュームをSATAハードディスクまたはCD-ROMとして使用します (nは0~5)。

```
--scsi[n] [file=<volume> [,aio=<native|threads|io_uring>]
[,backup=<1|0>] [,bps=<bps>] [,bps_max_length=<seconds>]
[,bps_rd=<bps>] [,bps_rd_max_length=<seconds>] [,bps_wr=<bps>]
[,bps_wr_max_length=<seconds>] [,cache=<enum>] [,cyls=<integer>
]。] [,detect_zeroes=<1|0>] [,discard=<ignore|on>]
[,format=<enum>] [,heads=<integer>] [,iops=<iops>]
[,iops_max=<iops>] [,iops_max_length=<seconds>] [,iops_rd=<iops>]
[,iops_rd_max=<iops>] [,iops_rd_max_length=<seconds>]
[,iops_wr=<iops>] [,iops_wr_max=<iops> ]。]
[,iops_wr_max_length=<seconds>] [,iothread=<1|0>] [,mbps=<mbps>]
[,mbps_max=<mbps>] [,mbps_rd=<mbps>] [,mbps_rd_max=<mbps> ]。]
[,mbps_wr=<mbps>] [,mbps_wr_max=<mbps>] [,media=<cdrom|disk>]
[,product=<product>] [,queues=<integer>] [,replicate=<1|0>]
[,rerror=<ignore|report|stop>] [,ro=<1|0>] [,scsiblock=<1|0>]
[,secs=<integer>] [,serial=<serial>] [,shared=<1|0>]
[,size=<DiskSize>] [,snapshot=<1|0>] [,ssd=<1|0>]
[,trans=<none|lba|auto>] [,vendor=<vendor>] [,werror=<enum>]
[,wwn=<wwn>] となる。
```

ボリュームをSCSIハードディスクまたはCD-ROMとして使用する (nは0~30)。

```
--scsihw <lsi | lsi53c810 | megasas | pvscsi | virtio-scsi-pci
| virtio-scsi-single> (デフォルト=lsi)
SCSIコントローラモデル
```

## --検索ドメイン <文字列>

cloud-init: コンテナのDNS検索ドメインを設定する。searchdomainもnameserverも設定されていない場合、Createはホストからの設定を自動的に使用する。

## -シリアル[n] (/dev/.+|socket)

VM内にシリアル・デバイスを作成する (nは0~3)

## --shares <integer> (0 - 50000) (デフォルト=1000)

オート・バルーニングのメモリ・シェア量。この数値が大きいほど、このVMはより多くのメモリを獲得する。数値は、他のすべての実行中のVMの重みに対する相対値です。ゼロを使用すると、オート・バルーニングは無効になる。オート・バルーニングはpvfstatdによって実行される。

## --smbios1 [base64=<1|0>] [,family=<Base64エンコードされた文字列>]

```
[,manufacturer=<Base64エンコードされた文字列>] [,product=<Base64エンコード
```

**された文字列】** [,serial=<Base64エンコードされた文字列>] [,sku=<Base64エンコードされた文字列>] [,uuid=<UUID>] [,version=<Base64エンコードされた文字列>] となります。

SMBIOSタイプ1のフィールドを指定する。

**--smp <整数> (1 - N) (デフォルト = 1)**

CPU数。代わりに -sockets オプションを使用してください。

**--ソケット <整数> (1 - N) (デフォルト = 1)**

CPUソケットの数。

--spice\_enhancements [folderssharing=<1|0>]  
[,videostreaming=<off|all|filter>]。

SPICE の追加機能拡張を設定する。

#### --sshkeys <文字列

cloud-init: 公開SSH鍵を設定する（1行に1つの鍵、OpenSSH形式）。

#### --startdate (now | YYYY-MM-DD | YYYY-MM-DDTHH:MM:SS) (default = now)

リアルタイムクロックの初期日付を設定する。有効な日付の書式は、'now' または 2006-06-17T16:01:21 または

2006-06-17.

#### --startup `[[order=]\d+] [,up=d+] [,down=d+]`.

スタートアップとシャットダウンの動作。Orderは、一般的な起動順序を定義する非負の数値である。  
シャットダウンは逆の順序で行われます。さらに、次のVMが起動または停止するまでの待ち時間を秒  
単位で設定できます。

#### --ストレージID

デフォルトのストレージ。

#### --タブレット <布尔值> (デフォルト = 1)

USBタブレットデバイスを有効/無効にします。

#### --タグ <文字列

VMのタグ。これはメタ情報に過ぎない。

#### --tdf <布尔值> (デフォルト = 0)

タイムドリフト修正の有効／無効。

#### --テンプレート <布尔值> (デフォルト = 0)

テンプレートの有効/無効。

#### --tpmstate0 [file=<volume> [,size=<DiskSize>] [,version=<v1.2|v2.0>].

TPMの状態を保存するDiskを設定する。フォーマットはraw固定。

--unused [n] [file=]<volume>

未使用ボリュームへの参照。これは内部的に使用されるので、手動で変更してはならない。

--usb [n] [[host=]<HOSTUSBDEVICE | spice>] [,mapping=<mappingid>]

[,usb3=<1|0[,マッピング=<マッピングID>] [,usb3=<1|0]

USBデバイスを設定する (nは0~4、マシンバージョン7.1以上、ostype l26またはwindows 7以上の場合、nは14まで)。

--vcpus <整数> (1 - N) (デフォルト = 0)

ホットプラグされたVCPUの数。

--vga [[type=]<enum>] [,クリップボード=<vnc[,clipboard=<vnc>>]

[,memory=<integer>] とする。

VGAハードウェアを設定する。

--virtio[n] [file=<volume> [,aio=<native|threads|io\_uring>]  
[,backup=<1|0>] [,bps=<bps>] [,bps\_max\_length=<seconds>]  
[,bps\_rd=<bps>] [,bps\_rd\_max\_length=<seconds>] [,bps\_wr=<bps>]  
[,bps\_wr\_max\_length=<seconds>] [,cache=<enum>] [,cyls=<integer>]  
[,detect\_zeroes=<1|0>] [,discard=<ignore|on>] [,format=<enum>]  
[,heads=<integer>] [,iops=<iops>] [,iops\_max=<iops>]  
[,iops\_max\_length=<seconds>] [,iops\_rd=<iops>]  
[,iops\_rd\_max=<iops>] [,iops\_rd\_max\_length=<秒>]  
[,iops\_wr=<iops>] [,iops\_wr\_max=<iops>]  
[,iops\_wr\_max\_length=<seconds>] [,iothread=<1|0>] [,mbps=<mbps>]  
[,mbps\_max=<mbps>] [,mbps\_rd=<mbps>] [,mbps\_rd\_max=<mbps>]  
[,mbps\_wr=<mbps>] [,mbps\_wr\_max=<mbps>]。]  
[,media=<cdrom|disk>] [,replicate=<1|0>]  
[,rerror=<ignore|report|stop>] [,ro=<1|0>] [,secs=<integer>]  
[,serial=<serial>] [,shared=<1|0>] [,size=<DiskSize>]  
[,snapshot=<1|0>] [,trans=<none|lba|auto>] [,werror=<enum>] とす  
る。

ボリュームをVIRTIOハードディスクとして使用する (nは0~15)。

--vmgenid <UUID> (デフォルト = 1 (自動生成))

VM生成IDを設定する。作成または更新時に自動生成する場合は 1 を使用し、明示的に無効にする場合は 0 を渡します。

--vmstatestorage <ストレージID>.

VM状態のボリューム/ファイルのデフォルトストレージ。

--watchdog [[model=]<i6300esb|ib700>][アクション]

仮想ハードウェア・ウォッチドッグ・デバイスを作成する。

## qmインポートディスク

qm disk import のエイリアス。

**qm importovf** <vmid> <manifest> <storage> [OPTIONS].

OVFマニフェストから読み込んだパラメータを使用して新しいVMを作成する

**<vmid>**: <整数> (100 - 999999999)

VMの（ユニークな）ID。

**<マニフェスト>**: <文字列

ovfファイルへのパス

**<ストレージ>**: <ストレージID

対象ストレージID

--ドライラン <ブール値

抽出されたOVFパラメータを解析して表示するが、VMは作成しない。

--format <qcow2 | raw | vmdk>.

対象フォーマット

### qmリスト [オプション]

仮想マシンのインデックス（ノードごと）。

--full <ブール値

アクティブなVMの完全なステータスを決定する。

**qm listsnapshot <vmid>**

すべてのスナップショットをリストアップする。

<vmid>: <整数> (100 - 999999999)

VMの（ユニークな）ID。

**qm migrate <vmid> <target> [OPTIONS]**

仮想マシンを移行します。新しい移行タスクを作成します。

<vmid>: <整数> (100 - 999999999)

VMの（ユニークな）ID。

<ターゲット>: <文字列

ターゲット・ノード

--bwlimit <integer> (0 - N) (デフォルト=データセンターまたはストレージ設定から

のマイグレーション制限)

I/Oバンド幅の制限をオーバーライドする（単位：KiB/s）。

--force <ブール値

ローカルデバイスを使用するVMのマイグレーションを許可する。このオプションを使用できるのはrootのみです。

--マイグレーション・ネットワーク <文字列

移行に使用する（サブ）ネットワークのCIDR。

--マイグレーション・タイプ <insecure | secure>

マイグレーショントラフィックはデフォルトでSSHトンネルを使用して暗号化されます。安全で完全にプライベートなネットワークでは、パフォーマンスを上げるためにこれを無効にすることができます。

#### --オンライン <ブール値>

VMが稼動している場合、オンライン/ライブマイグレーションを使用する。VMが停止している場合は無視されます。

#### --ターゲットストレージ <文字列>

ソースストレージからターゲットストレージへのマッピング。単一のストレージIDのみを指定すると、すべてのソースストレージがそのストレージにマッピングされる。特別な値 1 を指定すると、各ソース・ストレージはそれ自体にマッピングされます。

**--with-local-disks <ブール値>**  
ローカルディスクのライブストレージ移行を有効にする

**qmモニター <vmid>**

QEMU Monitorインターフェイスに入ります。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**qmムーブディスク**

*qm disk move* の別名。

**qm move\_disk**

*qm disk move* の別名。

**qm mtunnel**

*qmigrate* が使用する。

**qm nbdstop <vmid>**

組み込み nbd サーバーを停止します。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**qmペンドィング <vmid>**

仮想マシンのコンフィギュレーションを、現在の値と保留中の値の両方で取得します。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**qm reboot <vmid> [オプション]。**

VMをシャットダウンして再起動する。保留中の変更を適用する。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

#### --タイムアウト <整数> (0 - N)

シャットダウンまで最大タイムアウト秒数待つ。

```
qm remote-migrate <vmid> [<target-vmid>] <target-endpoint> --target-bridge <string>  
--target-storage <string> [OPTIONS].
```

仮想マシンをリモートクラスタに移行します。新しい移行タスクを作成します。EXPERIMENTAL 機能!

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**<ターゲット-vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**<ターゲットエンドポイント>: apitoken=<PVEAPIToken=user@realm!token=SECRET>。**

**ホスト=<ADDRESS> [, フィンガープリント=<FINGERPRINT>] [, ポート=<PORT>]。**

リモート・ターゲット・エンドポイント

**--bwlimit <integer> (0 - N) (デフォルト = データセンターまたはストレージ設定からのマイグレーション制限)**

I/O バンド幅の制限をオーバーライドする（単位：KiB/s）。

**--delete <boolean> (デフォルト = 0)**

移行が成功したら、元の VM と関連データを削除します。デフォルトでは、元の VM は停止状態でソース・クラスタ上に保持される。

**--オンライン <ブール値>**

VM が実行中の場合、オンライン/ライブマイグレーションを使用する。VM が停止している場合は無視されます。

**--ターゲット・ブリッジ <文字列>**

ソースブリッジからターゲットブリッジへのマッピング。単一のブリッジ ID だけを指定すると、すべてのソース・ブリッジがそのブリッジにマッピングされます。特別な値 1 を指定すると、各ソース・ブリッジはそれ自体にマッピングされます。

**-ターゲット・ストレージ <文字列>**

ソースストレージからターゲットストレージへのマッピング。単一のストレージ ID のみを指定すると、すべてのソースストレージがそのストレージにマッピングされる。特別な値 1 を指定すると、各ソース・ストレージはそれ自体にマッピングされます。

## qm再スキャン

*qm disk rescan* の別名。

**qm reset <vmid> [OPTIONS] (qmリセット <vmid> [オプション])**

仮想マシンをリセットします。

**<vmid>: <整数> (100 - 99999999)**

VMの（ユニークな）ID。

**--スキップロック <ブール値**

Ignore locks - rootのみがこのオプションを使用できる。

**qmリサイズ**

*qm disk resize* の別名。

**qm resume** <vmid> [OPTIONS] [オプション]。

仮想マシンを再開する。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**--nocheck <布尔值**

記述なし

**--スキップロック <布尔值**

Ignore locks - rootのみがこのオプションを使用できる。

**qm rollback <vmid> <スナップ名> [オプション]。**

VMの状態を指定したスナップショットにロールバックする。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**<スナップ名>: <文字列**

スナップショットの名前。

**--start <boolean> (デフォルト = 0)**

ロールバック成功後にVMを起動するかどうか。(注意: スナップショットにRAMが含まれている場合、

VMは自動的に起動します)

**qm sendkey <vmid> <key> [オプション]。**

仮想マシンにキーイベントを送信する。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**<キー>: <文字列**

キー (qemuモニターエンコーディング)。

**--スキップロック <布尔值**

Ignore locks - rootのみがこのオプションを使用できる。

**qm set <vmid> [OPTIONS] [オプション] .**

仮想マシン・オプションの設定 (synchrounous API) - ホットプラグやストレージ割り当てを含む操作には、代わりに POST メソッドの使用を検討する必要があります。

**<vmid>: <整数> (100 - 99999999)**

VMの（ユニークな）ID。

**--acpi <ブール値> (デフォルト = 1)**

ACPI を有効/無効にする。

**--アフィニティ <文字列>**

ゲスト・プロセスの実行に使用されるホスト・コアのリスト（例：0,5,8-11

```
--agent [enabled=<1|0> [,freeze-fs-on-backup=<1|0>] [,fstrim_cloned_disks=<1|0>] [,type=<virtio|isa> ]。
```

QEMU ゲストエージェントとそのプロパティとの通信を有効/無効にします。

```
--arch <aarch64 | x86_64>.
```

仮想プロセッサーのアーキテクチャ。デフォルトはホスト。

**--args <文字列>**

kvmに渡される任意の引数。

```
--audio0 device=<ich9-intel-hda|intel-hda|AC97> [,driver=<spice|none>].
```

QXL/Spiceと組み合わせて使用すると便利です。

```
--autostart <boolean> (デフォルト = 0)
```

クラッシュ後の自動再起動（現在は無視）。

**--バルーン <整数> (0 - N)**

VMのターゲットRAMの量（MiB）。ゼロを使用すると、バロン・ドライバが無効になります。

```
--bios <ovmf | seabios> (デフォルト = seabios)
```

BIOSの実装を選択します。

```
--boot [[legacy=]<[acdn]{1,4}>] [order=<デバイス[,デバイス...]>]]。
```

ゲストの起動順序を指定します。order= サブプロパティを使用します。key または legacy= を指定しない使用法は非推奨です。

```
--ブートディスク (ide|sata|scsi|virtio)◆d+.
```

指定したディスクからの起動を有効にする。非推奨：代わりに boot: order=foo;bar を使う。

```
--CDROM <ボリューム>
```

これは -ide2 オプションのエイリアスである。

```
--cicustom [meta=<volume>] [,network=<volume>] [,user=<volume>]  
[,vendor=<volume>].
```

cloud-init: 開始時に自動生成されるファイルを置き換えるカスタムファイルを指定する。

## --パスワード

cloud-init: ユーザーに割り当てるパスワード。一般的に、これを使うことは推奨されない。代わりにsshキーを使ってください。また、古いバージョンの cloud-init はハッシュ化されたパスワードをサポートしていないことに注意してください。

## --シティープ <configdrive2 | nocloud | opennebula>。

cloud-initコンフィギュレーション・フォーマットを指定します。デフォルトは、設定されているオペレーティング・システムの種類 (ostype.Linuxではnocloud形式、Windowsではconfigdrive2を使用します。

**-ciupgrade (アップグレード) <ブール値> (デフォルト = 1)**

cloud-init: 初回起動後にパッケージの自動アップグレードを行います。

**-ユーザー <文字列>**

cloud-init: イメージに設定されているデフォルトユーザーの代わりに、sshキーとパスワードを変更するユーザー名。

**--コア数 <整数> (1 - N) (デフォルト = 1)**

ソケットあたりのコア数。

**--cpu [[cputype=<string>] [, フラグ=<+FLAG[;-FLAG...]>] ]を指定します。 [,hidden=<1|0>] [,hv-vendor-id=<vendor-id>] を指定する。**

[phys-bits=<8-64|host>] [,reported-model=<enum> ]。

エミュレートされたCPUタイプ。

**--cpulimit <number> (0 - 128) (デフォルト = 0)**

CPU使用量の上限。

**--cpuunits <integer> (1 - 262144) (デフォルト = cgroup v1: 1024, cgroup v2: 100)**

VM の CPU ウェイトは、cgroup v2 では [1, 10000] にクランプされる。

**--削除 <文字列>**

削除したい設定のリスト。

**--説明 <文字列>**

VM の説明。WebインターフェイスのVMのサマリーに表示される。これはコンフィギュレーション・ファイルのコメントとして保存される。

**-ダイジェスト <文字列>**

現在のコンフィギュレーションファイルのSHA1ダイジェストが異なる場合、変更を防止する。これは同時変更を防ぐために使用できます。

```
--efidisk0 [file=<volume> [,efitype=<2m|4m>]  
[,format=<enum>] [,import-from=<ソースボリューム>] [,pre-  
enrolled-keys=<1|0>] [,size=<DiskSize>].
```

EFIバーを格納するディスクを設定する。特別な構文 STORAGE\_ID:SIZE\_IN\_GiB を使用して、以下のようにディスクを割り当てます。

を新しいボリュームにコピーする。SIZE\_IN\_GiBはここでは無視され、代わりにデフォルトのEFIバーがボリュームにコピーされることに注意。既存のボリュームからインポートするには、STORAGE\_ID:0と*import-from*パラメータを使用します。

#### --force <ブール値

物理的な削除を強制する。これがないと、単純に設定ファイルからディスクを削除し、ボリュームIDを含む*unused[n]*という追加の設定エントリを作成します。*unused[n]*のリンク解除は、常に物理的な削除を引き起こします。

---

## 注

必要なオプション: 削除

---

### --freeze <ブール値>

起動時にCPUをフリーズさせる (c monitorコマンドで実行開始)。

### --hookscript <文字列>

vmsのライフタイムの様々なステップで実行されるスクリプト。

```
--hostpci [n] [[host=<HOSTPCIID[;HOSTPCIID2...]>][,device-id=<hex id>][,legacy-igd=<1|0>][,mapping=<mapping-id>][,mdev=<string>][,pcie=<1|0>][,rombar=<1|0>][,romfile=<string>][,sub-device-id=<hex id>][,sub-vendor-id=<hex id>][,vendor-id=<hex id>][,x-vga=<1|0>]。
```

ホストのPCIデバイスをゲストにマップする。

### --hotplug <string> (デフォルト = ネットワーク,ディスク,USB)

ホットプラグ機能を選択的に有効にします。ネットワーク、ディスク、CPU、メモリ、USB、*cloudinit*。ホットプラグを完全に無効にするには0を使用します。値として1を使用すると、デフォルトのnetwork,disk,usbのエイリアスになります。USB ホットプラグは、マシンのバージョンが >= のゲストで可能です。

7.1とostype l26またはwindows > 7.

### --hugepages <1024 | 2 | any>.

ヒュッゲページ・メモリの有効／無効。

```
--ide [n] [file=<volume>[,aio=<native|threads|io_uring>][,backup=<1|0>][,bps=<bps>][,bps_max_length=<seconds>][,bps_rd=<bps>][,bps_rd_max_length=<seconds>][,bps_wr=<bps>][,bps_wr_max_length=<seconds>][,cache=<enum>][,cyls=<integer>][,detect_zeroes=<1|0>][,discard=<ignore|on>][,format=<enum>][,heads=<integer>][,import-from=<ソースボリューム>][,iops=<iops>][,iops_max=<iops>][,iops_max_length=<seconds>][,iops_rd=<iops>][,iops_rd_max=<iops>][,iops_rd_max_length=<seconds>][,iops_wr=<iops>][,iops_wr_max=<iops>][,iops_wr_max_length=<seconds>][,mbps=<mbps>][,mbps_max=<mbps>][,mbps_rd=<mbps>][,mbps_rd_max=<mbps>][,mbps_wr=<mbps>][,mbps_wr_max=<mbps>][,media=<CDROM|DISK>][,model=<model>][,replicate=<1|0>][,rerror=<ignore|report|stop>][,secs=<integer>][,serial=<serial>][,shared=<1|0>][,size=<DiskSize>]
```

---

```
[,snapshot=<1|0>] [,ssd=<1|0>] [,trans=<none|lba|auto>]  
[,werror=<enum>] [,wwn=<wwn>] となる。
```

ボリュームをIDEハードディスクまたはCD-ROMとして使用する（nは0～3）。新しいボリュームを割り当てるには、特別な構文 `STORAGE_ID:SIZE_IN_GiB` を使用する。既存のボリュームからインポートするには、`STORAGE_ID:0`と*import-from*パラメータを使用する。

```
--ipconfig[n] [gw=<GatewayIPv4>]  
[,gw6=<GatewayIPv6>] [,ip=<IPv4Format/CIDR>]  
[,ip6=<IPv6Format/CIDR>].
```

cloud-init: 対応するインターフェースのIPアドレスとゲートウェイを指定する。

IPアドレスはCIDR表記を使用し、ゲートウェイはオプションだが、同じタイプのIPを指定する必要がある。

特別な文字列`dhcp`は、DHCPを使用するIPアドレスに使用でき、その場合、明示的な ゲートウェイを指定する必要はない。IPv6の場合、ステートレス自動設定を使用するには、特別な文字列`auto`を使用できる。これにはcloud-init 19.4以降が必要です。

cloud-initが有効で、IPv4アドレスもIPv6アドレスも指定されていない場合、デフォルトでIPv4の`dhcp`を使用する。

**--ivshmem size=<整数> [,name=<文字列>]**。

VM間共有メモリ。VM間やホストとの直接通信に便利。

**--keephugepages <boolean> (デフォルト = 0)**

`hugepages`と併用する。有効にすると、VMシャットダウン後も`hugepages`は削除されず、その後の起動に使用できます。

**--キーボード <da | de | de-ch | en-gb | en-us | es | fi | fr | fr-be | フランス語 | スペイン語 | フランス語 | フランス語 | 中国語 | 胡語 | is | it | ja | lt | mk | nl | no | pl | pt | pt-br | sl | sv | tr>**。

VNCサーバーのキーボードレイアウト。このオプションは一般的には必要なく、ゲストOS内で処理した方が良い場合が多い。

**--kvm <論理値> (デフォルト = 1)**

KVMハードウェア仮想化の有効/無効。

**--ローカルタイム <ブール値>**

リアルタイムクロック (RTC) をローカルタイムに設定する。これはデフォルトで有効である。  
Microsoft Windows OS。

**--lock <バックアップ | クローン | 作成 |マイグレーション | ロールバック | スナップショット | スナップショット削除 | サスPEND | 一時停止>**。

VMをロック/アンロックする。

**--マシン [[タイプ=]<マシンタイプ>] [,viommu=<intel|virtio>] ]。**

[`viommu=<intel|virtio>`] である。

QEMUマシンを指定する。

--メモリ [current=]<整数>

メモリ特性。

--migrate\_downtime <number> (0 - N) (デフォルト = 0.1)

マイグレーションの最大許容ダウンタイム（秒）を設定します。

--migrate\_speed <integer> (0 - N) (デフォルト = 0)

マイグレーションの最大速度（MB/s）を設定する。値0は制限なし。

--名前 <文字列>

VM の名前を設定します。コンフィギュレーション・ウェブ・インターフェイスでのみ使用されます。

## --ネームサーバー <文字列>

cloud-init: コンテナの DNS サーバー IP アドレスを設定する。searchdomain も nameserver も設定されていない場合、Create はホストからの設定を自動的に使用する。

--net[n] [model=<enum> [,bridge=<bridge>] [,firewall=<1|0>]  
[,link\_down=<1|0>] [,macaddr=<XX:XX:XX:XX:XX: XX>] [,mtu=<integer>]  
[,queues=<integer>] [,rate=<number>] [,tag=<integer>]  
[,trunks=<vlanid[;vlanid...]>] [,<model>=<macaddr>] [,<model>=<macaddr>]  
,<model>=<macaddr>  
ネットワーク機器を指定する。

--numa <boolean> (デフォルト = 0)

NUMAの有効/無効。

--numa[n] cpus=<id[-id];...> [,hostnodes=<id[-id];...>]

を指定します。[,memory=<number>]

[,policy=<preferred|bind|interleave>] である。

NUMA トポロジー。

--onboot <boolean> (デフォルト = 0)

システム起動時にVMを起動するかどうかを指定する。

--ostype <l24 | 126 | other | solaris | w2k | w2k3 | w2k8 | win10 |  
win11 | win7 | win8 | wvista | wxp>.

ゲストOSを指定する。

--parallel[n] /dev/parportd+|/dev/usb/lpd+

ホストパラレルデバイスをマップする (nは0~2) 。

--プロテクション <ブール値> (デフォルト = 0)

VM の保護フラグを設定する。これにより、VMの削除とディスクの削除操作が無効になる。

--reboot <boolean> (デフォルト = 1)

再起動を許可する。0に設定すると、リブート時にVMが終了する。

--復帰 <文字列>

保留中の変更を取り消す。

```
--rng0 [source=</dev/urandom|/dev/random|/dev/hwrng>
[,max_bytes=<integer>] [,period=<integer>].
```

VirtIO ベースの乱数ジェネレータを設定します。

```
--sata[n] [file=<volume> [,aio=<native|threads|io_uring>]
[,backup=<1|0>] [,bps=<bps>] [,bps_max_length=<seconds>]
[,bps_rd=<bps>] [,bps_rd_max_length=<seconds>] [,bps_wr=<bps>]
[,bps_wr_max_length=<seconds>] [,cache=<enum>] [,cyls=<integer>]
[,detect_zeroes=<1|0>] [,discard=<ignore|on>] [,format=<enum>]
[,heads=<integer>] [,import-from=<ソースボリューム>] [,iops=<iops>]
[,iops_max=<iops>] [,iops_max_length=<seconds>] [,iops_rd=<iops>]
[,iops_rd_max=<iops>] [,iops_rd_max_length=<seconds>]
[,iops_wr=<iops>] [,iops_wr_max=<iops>]
[,iops_wr_max_length=<seconds>] [,mbps=<mbps>] [,mbps_max=<mbps>]
[,mbps_rd=<mbps>] [,mbps_rd_max=<mbps>] [,mbps_wr=<mbps>]
[,mbps_wr_max=<mbps>] [,media=<cdrom|disk>] [,replicate=<1|0>]
[,rerror=<ignore|report|stop>] [,secs=<integer>] [,serial=<serial>]
[,shared=<1|0> ]。 [,size=<DiskSize>] [,snapshot=<1|0>]
[,ssd=<1|0>] [,trans=<none|lba|auto>] [,werror=<enum>] [,wwn=<wwn>]
```

とする。

ボリュームをSATAハードディスクまたはCD-ROMとして使用する (nは0~5)。新しいボリュームを割り当てるには、特別な構文 STORAGE\_ID:SIZE\_IN\_GiB を使用する。既存のボリュームからインポートするには、STORAGE\_ID:0と*import-from*パラメータを使用します。

```
--scsi[n] [[file=<volume> [,aio=<native|threads|io_uring>>]
[,backup=<1|0>] [,bps=<bps>] [,bps_max_length=<seconds>]
[,bps_rd=<bps>] [,bps_rd_max_length=<seconds>] [,bps_wr=<bps>]
[,bps_wr_max_length=<seconds>] [,cache=<enum>] [,cyls=<integer>]
[,detect_zeroes=<1|0>] [,discard=<ignore|on>] [,format=<enum>]
[,heads=<integer>] [,import-from=<ソースボリューム>] [,iops=<iops>]
[,iops_max=<iops>] [,iops_max_length=<seconds>] [,iops_rd=<iops>]
[,iops_rd_max=<iops>] [,iops_rd_max_length=<seconds>]
[,iops_wr=<iops>] [,iops_wr_max=<iops>]
[,iops_wr_max_length=<seconds>] [,iothread=<1|0>] [,mbps=<mbps>]
[,mbps_max=<mbps>] [,mbps_rd=<mbps>] [,mbps_rd_max=<mbps>]
[,mbps_wr=<mbps>] [,mbps_wr_max=<mbps>] [,media=<cdrom|disk>]
[,product=<product>] [,queues=<integer>] [,replicate=<1|0>]
[,rerror=<ignore|report|stop>] [,ro=<1|0>] [,scsiblock=<1|0>]
[,secs=<integer>] [,serial=<serial>] [,shared=<1|0> ]。
[,size=<DiskSize>] [,snapshot=<1|0>] [,ssd=<1|0>]
[,trans=<none|lba|auto>] [,vendor=<vendor>] [,werror=<enum>]
[,wwn=<wwn> ]。
```

ボリュームをSCSIハードディスクまたはCD-ROMとして使用する (nは0から30)。新しいボリュームを割り当てるには、特別な構文 STORAGE\_ID:SIZE\_IN\_GiB を使用する。新しいボリュームを割り当てるには、STORAGE\_ID:0と*import-from*パラメータを使用する。

を設定する。

```
--scsihw <lsi | lsi53c810 | megasas | pvscsi | virtio-scsi-pci  
| virtio-scsi-single> (デフォルト=lsi)
```

SCSIコントローラモデル

#### --検索ドメイン <文字列>

cloud-init: コンテナのDNS検索ドメインを設定する。searchdomainもnameserverも設定されていない場合、Createはホストからの設定を自動的に使用する。

**-シリアル[n] (/dev/.+|socket)**

VM内にシリアル・デバイスを作成する (nは0~3)

**--shares <integer> (0 - 50000) (デフォルト = 1000)**

オート・バルーニングのメモリ・シェア量。この数値が大きいほど、このVMはより多くのメモリを獲得する。数値は、他のすべての実行中のVMの重みに対する相対値です。ゼロを使用すると、オート・バルーニングは無効になる。オート・バルーニングはpvestatdによって実行される。

**--スキップロック <ブール値>**

Ignore locks - rootのみがこのオプションを使用できる。

**--smbios1 [base64=<1|0>] [,family=<Base64エンコードされた文字列>]**

[,manufacturer=<Base64エンコードされた文字列>] [,product=<Base64エンコードされた文字列>] [,serial=<Base64エンコードされた文字列>] [,sku=<Base64エンコードされた文字列>] [,uuid=<UUID>] [,version=<Base64エンコードされた文字列>] となります。

SMBIOSタイプ1のフィールドを指定する。

**--smp <整数> (1 - N) (デフォルト = 1)**

CPU数。代わりに -sockets オプションを使用してください。

**--ソケット <整数> (1 - N) (デフォルト = 1)**

CPUソケットの数。

**--spice\_enhancements [foldersharing=<1|0>]**

[,videostreaming=<off|all|filter>]。

SPICE の追加機能拡張を設定する。

**--sshkeys <ファイルパス>**

cloud-init: 公開SSH鍵を設定する (1行に1つの鍵、OpenSSH形式)。

**--startdate (now | YYYY-MM-DD | YYYY-MM-DDTHH:MM:SS) (default = now)**

リアルタイムクロックの初期日付を設定する。有効な日付の書式は、'now' または 2006-06-17T16:01:21 または

2006-06-17.

--startup `[[order=\d+] [,up=d+] [,down=d+]]`.

スタートアップとシャットダウンの動作。Orderは、一般的な起動順序を定義する非負の数値である。シャットダウンは逆の順序で行われます。さらに、次のVMが起動または停止するまでの待ち時間を秒単位で設定できます。

--タブレット <ブール値> (デフォルト = 1)

USBタブレットデバイスを有効/無効にします。

--タグ <文字列>

VMのタグ。これはメタ情報に過ぎない。

--tdf <ブール値> (デフォルト = 0)

タイムドリフト修正の有効／無効。

--テンプレート <ブール値> (デフォルト = 0)

テンプレートの有効/無効。

--tpmstate0 [ファイル=<ボリューム> [,インポート元=<ソースボリューム>] [,サイズ=<ディスクサイズ>] [,バージョン=<v1.2|v2.0>]。

TPMの状態を保存するDiskを設定する。フォーマットはrawで固定される。新しいボリュームを割り当てるには、特別な構文STOR-AGE\_ID:SIZE\_IN\_GiBを使用する。SIZE\_IN\_GiBはここでは無視され、代わりに4MiBが使用されることに注意。既存のボリュームからインポートするには、STORAGE\_ID:0とimport-fromパラメータを使用します。

--unused[n] [file=<volume>

未使用ボリュームへの参照。これは内部的に使用されるので、手動で変更してはならない。

--usb[n] [[host=<HOSTUSBDEVICE|spice>] [,mapping=<mappingid>] [,usb3=<1|0>[,マッピング=<マッピングID>] [,usb3=<1|0>]

USBデバイスを設定する (nは0~4、マシンバージョン7.1以上、ostype l26またはwindows 7以上の場合、nは14まで)。

--vcpus <整数> (1 - N) (デフォルト = 0)

ホットプラグされたVCPUの数。

--vga [[type=<enum>] [,クリップボード=<vnc[,clipboard=<vnc>] [,memory=<integer>]] とする。

VGAハードウェアを設定する。

--virtio[n] [file=<volume> [,aio=<native|threads|io\_uring>] [,backup=<1|0>] [,bps=<bps>] [,bps\_max\_length=<seconds>] [,bps\_rd=<bps>] [,bps\_rd\_max\_length=<seconds>] [,bps\_wr=<bps>] [,bps\_wr\_max\_length=<seconds>] [,cache=<enum>] [,cyls=<integer>] [,detect\_zeroes=<1|0>] [,discard=<ignore|on>] [,format=<enum>] [,heads=<integer>] [,import-from=<ソースボリューム>] [,iops=<iops>] [,iops\_max=<iops>] [,iops\_max\_length=<seconds>] [,iops\_rd=<iops>] [,iops\_rd\_max=<iops>] [,iops\_rd\_max\_length=<seconds>] [,iops\_wr=<iops>] [,iops\_wr\_max=<iops>] [,iops\_wr\_max\_length=<seconds>] [,iothread=<1|0>] [,mbps=<mbps>] [,mbps\_max=<mbps>] [,mbps\_rd=<mbps>] [,mbps\_rd\_max=<mbps>] [,mbps\_wr=<mbps>] [,mbps\_wr\_max=<mbps>] [,media=<cdrom|disk>]

```
[,replicate=<1|0>] [,rerror=<ignore|report|stop>] [,ro=<1|0>]  
[,secs=<integer> ]。 [,serial=<serial>] [,shared=<1|0>]  
[,size=<DiskSize>] [,snapshot=<1|0>] [,trans=<none|lba|auto>]  
[,werror=<enum> ]。
```

ボリュームをVIRTIOハードディスクとして使用する (nは0~15)。新しいボリュームを割り当てるには、特別な構文 STORAGE\_ID:SIZE\_IN\_GiB を使用する。既存のボリュームからインポートするには、STORAGE\_ID:0と*import-from*パラメータを使用します。

#### --vmgenid <UUID> (デフォルト = 1 (自動生成))

VM生成IDを設定する。作成または更新時に自動生成する場合は 1 を使用し、明示的に無効にする場合は 0 を渡します。

**--vmstatestorage <ストレージID>**

VM状態のボリューム/ファイルのデフォルトストレージ。

**--watchdog [[model=]<i6300esb|ib700>] [アクション]**

仮想ハードウェア・ウォッチドッグ・デバイスを作成する。

**qm showcmd <vmid> [OPTIONS]**

VMの起動に使用されるコマンドラインを表示する（デバッグ情報）。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**--pretty <ブール値> (デフォルト = 0)**

各オプションを改行し、可読性を高める。

**--スナップショット <文字列>**

指定されたスナップショットから設定値を取得します。

**qm shutdown <vmid> [オプション]**

仮想マシンをシャットダウンする。これは物理マシンの電源ボタンを押すのと似ています。これはゲストOSにACPIイベントを送信し、クリーンシャットダウンに進むはずです。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**--forceStop <boolean> (デフォルト = 0)**

VMが停止していることを確認する。

**--keepActive <boolean> (デフォルト = 0)**

ストレージボリュームを非アクティブにしないでください。

**--スキップロック <ブール値>**

Ignore locks - rootのみがこのオプションを使用できる。

--タイムアウト <整数> (0 - N)

最大タイムアウト秒待つ。

**qm snapshot** <vmid> <snapname> [OPTIONS]。

VMをスナップショットする。

<vmid>: <整数> (100 - 999999999)

VMの（ユニークな）ID。

**<スナップ名>: <文字列**

スナップショットの名前。

**--説明 <文字列**

テキストによる説明やコメント。

**--vmstate <ブール値**

vmstateを保存する

**qm start <vmid> [OPTIONS] .**

仮想マシンを起動します。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**-強制CPU <文字列**

QEMUの-cpu引数を与えられた文字列でオーバーライドする。

**--マシン [[タイプ=]<マシンタイプ>] [,viommu=<intel|virtio>] 。**

[viommu=<intel|virtio>] である。

QEMUマシンを指定する。

**-migratedfrom <文字列>**

クラスタ・ノード名。

**--マイグレーション・ネットワーク <文字列**

移行に使用する（サブ）ネットワークのCIDR。

**--マイグレーション・タイプ <insecure | secure>**

マイグレーショントラフィックはデフォルトでSSHトンネルを使用して暗号化されます。安全で完全にプライベートなネットワークでは、パフォーマンスを上げるためにこれを無効にすることができます。

**--スキップロック <ブール値**

Ignore locks - rootのみがこのオプションを使用できる。

**--stateuri <文字列**

いくつかのコマンドは、この場所から状態をセーブ／リストアする。

### --ターゲットストレージ <文字列>

ソースストレージからターゲットストレージへのマッピング。単一のストレージIDのみを指定すると、すべてのソースストレージがそのストレージにマッピングされる。特別な値 1 を指定すると、各ソース・ストレージはそれ自体にマッピングされます。

### --タイムアウト <integer> (0 - N) (デフォルト = max(30, vmのメモリはGiB))

最大タイムアウト秒待つ。

**qm status** <vmid> [OPTIONS] .

VMのステータスを表示する。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**--verbose <布尔值>**

冗長出力フォーマット

**qm stop <vmid> [OPTIONS].**

仮想マシンを停止します。qemuプロセスは直ちに終了する。これは実行中のコンピュータの電源プラグを抜くようなもので、仮想マシンのデータを損傷する可能性がある。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**--keepActive <boolean> (デフォルト = 0)**

ストレージボリュームを非アクティブにしないでください。

**-migratedfrom <文字列>**

クラスタ・ノード名。

**--overrule-shutdown <boolean> (デフォルト = 0)**

停止する前に、アクティブなqmshutdownタスクを中止するようにしてください。

**--スキップロック <布尔值>**

Ignore locks - rootのみがこのオプションを使用できる。

**--タイムアウト <整数> (0 - N)**

最大タイムアウト秒待つ。

**qm suspend <vmid> [OPTIONS].**

仮想マシンをサスペンドします。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**--スキップロック <布尔值>**

Ignore locks - rootのみがこのオプションを使用できる。

## --ストレージID

VMの状態を保存するストレージ

---

### 注

必要なオプション: `toDisk`

---

**-todisk<ブール値> (デフォルト=0)**

設定すると、VMをディスクにサスペンドする。次回のVM起動時に再開される。

**qm テンプレート <vmid> [OPTIONS] .**

テンプレートを作成する。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

--ディスク <efidisk0 | ide0 | ide1 | ide2 | ide3 | sata0 | sata1 |  
sata2 | sata3 | sata4 | sata5 | scsi0 | scsi1 | scsi10 | scsi11 |  
scsi12 | scsi13 | scsi14 | scsi15 | SCSI16 | SCSI17 | SCSI18 | SCSI19 |  
SCSI2 | SCSI20 | SCSI21 | SCSI22 | SCSI23 | SCSI24 | SCSI25 | SCSI26 | SCSI27  
| SCSI28 | SCSI29 | SCSI3 | SCSI30 | SCSI4  
| scsi5 | scsi6 | scsi7 | scsi8 | scsi9 | tpmstate0 | virtio0 | virtio1 | virtio1  
0 | virtio11 | virtio12 | virtio13 | virtio14 | virtio15 | virtio2 | virtio3 | v  
irtio4 | virtio5 | virtio6 | virtio7 | virtio8 | virtio9>。

1つのディスクだけをベースイメージに変換したい場合。

**qm terminal <vmid> [OPTIONS] .**

シリアル・デバイスを使用してターミナルを開く（VMにはシリアル・デバイスが設定されている必要があります。）

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**--escape <文字列> (デフォルト = ^O)**

エスケープ文字。

**--iface <シリアル0 | シリアル1 | シリアル2 | シリアル3**

シリアル・デバイスを選択します。デフォルトでは、単に最初の適切なデバイスを使用します。

**qmアンリンク**

**qm disk unlink** のエイリアス。

ス。 **qm unlock** <vmid>

VM のロックを解除する。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**qm vncproxy** <vmid>

VMのVNC トラフィックを標準入出力にプロキシする

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**qm wait <vmid> [OPTIONS] .**

VMが停止するまで待つ。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**--タイムアウト <整数> (1 - N)**

秒単位のタイムアウト。デフォルトは永久に待つ。

## A.9 qmrestore - QemuServer vzdump バックアップをリストアする

**qmrestore**ヘルプ

**qmrestore <アーカイブ> <vmid> [オプション] .**

QemuServer vzdump/バックアップのリストア。

**<アーカイブ>: <文字列**

バックアップ・ファイル。標準入力から読み込む場合は - を渡す。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**-bwlimit<数値> (0 - N)**

I/Oバンド幅の制限をオーバーライドする（単位：KiB/s）。

**--force <ブール値>**

既存のVMの上書きを許可する。

**--ライブ・リストア <ブール値>**

バックアップから直ちにVMを起動し、バックグラウンドでリストアする。PBSのみ。

**--プール <文字列>**

VMを指定したプールに追加する。

#### --ストレージID

デフォルトのストレージ。

#### --unique <ブール値>

一意のランダムなイーサネットアドレスを割り当てる。

## A.10 pct - Proxmoxコンテナツールキット

**pct** <COMMAND> [ARGS] [OPTIONS].

**pct clone** <vmid> <newid> [オプション]。

コンテナのクローン/コピーを作成する

<vmid>: <整数> (100 - 999999999)

VMの（ユニークな）ID。

<newid>: <整数> (100 - 999999999)

クローンのVMID。

--bwlimit <number> (0 - N) (デフォルト = データセンターまたはストレージ設定からのクローン制限)

I/Oバンド幅の制限をオーバーライドする（単位：KiB/s）。

--説明 <文字列

新しいCTの説明

--full <ブール値

すべてのディスクの完全コピーを作成する。これは通常のCTをクローンするときに必ず行われます。

CTテンプレートでは、デフォルトでリンクされたクローンを作成しようとします。

--ホスト名 <文字列

新しいCTのホスト名を設定します。

--プール <文字列

新しいCTを指定されたプールに追加する。

--スナップネーム <文字列

スナップショットの名前。

--ストレージID

フルクローンの対象ストレージ。

## --ターゲット <文字列>

ターゲット・ノード。元のVMが共有ストレージ上にある場合にのみ許可される。

**pct config** <vmid> [OPTIONS] [オプション]。

コンテナ構成を取得する。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**--カレント <ブール値> (デフォルト = 0)**

保留値ではなく) 現在の値を取得する。

**--スナップショット <文字列>**

指定されたスナップショットから設定値を取得します。

**pct コンソール <vmid> [オプション] .**

指定したコンテナのコンソールを起動する。

**<vmid>: <整数> (100 - 999999999)**

VMの(ユニークな)ID。

**--escape ``^w`` (デフォルト = ^a)**

エスケープシーケンスの接頭辞。例えば、<Ctrl+b q>をエスケープシーケンスとして使用するには、`bを渡す。

**pct cpusets**

割り当てられたCPUセットのリストを印刷する。

**pct create <vmid> <ostemplate> [OPTIONS] [オプション]。**

コンテナを作成または復元する。

**<vmid>: <整数> (100 - 999999999)**

VMの(ユニークな)ID。

**<ostemplate>: <文字列>**

OSのテンプレートまたはバックアップファイル。

**--arch <amd64 | arm64 | armhf | i386 | riscv32 | riscv64> (デフォルト = amd64)**

OSのアーキテクチャタイプ。

**--bwlimit <number> (0 - N) (デフォルト = データセンターまたはストレージ設定から制限を復元)**

I/O バンド幅の制限をオーバーライドする（単位：KiB/s）。

**--cmode <コンソール | シェル | tty> (デフォルト = tty)**

コンソールモード。デフォルトでは、consoleコマンドは利用可能なttyデバイスの1つに接続を開こうとします。cmode を *console* に設定すると、代わりに /dev/console にアタッチしようとする。cmode を *shell* に設定すると、単にコンテナ内でシェルを起動する（ログインはしない）。

**--コンソール <ブール値> (デフォルト = 1)**

コンテナにコンソールデバイス（/dev/console）をアタッチする。

**--コア <整数> (1 - 8192)**

コンテナに割り当てられたコアの数。コンテナは、デフォルトで使用可能なすべてのコアを使用できる。

--cpulimit <number> (0 - 8192) (デフォルト = 0)

CPU使用量の上限。

---

### 注

コンピュータに2つのCPUがある場合、合計2つのCPU時間を持つ。値0はCPU制限なしを示す。

---

--cpuunits <integer> (0 - 500000) (デフォルト = cgroup v1: 1024, cgroup v2: 100)

コンテナのCPU重量は、cgroup v2 では [1, 10000] にクランプされる。

--debug <ブール値> (デフォルト = 0)

もっと冗長にしてみてください。今のところ、これは起動時にデバッグログレベルを有効にするだけです。

--説明 <文字列>

コンテナの説明。ウェブインターフェイスCTのサマリーに表示されます。設定ファイルのコメントとして保存されます。

--dev[n] [[path=] <パス> [gid=<integer>] [,mode=<オクタルのアクセスモード>] [,uid=<integer>]。]

コンテナへの通過装置

--features [force\_rw\_sys=<1|0>] [,fuse=<1|0>] [,keyctl=<1|0>] [,mknod=<1|0>] [,mount=<fstype;fstype;...>] [,nesting=<1|0>] とする。

コンテナが高度な機能にアクセスできるようにする。

--force <ブール値>

既存のコンテナの上書きを許可する。

--hookscript <文字列>

コンテナのライフタイムのさまざまな段階で実行されるスクリプト。

--ホスト名 <文字列>

コンテナのホスト名を設定する。

--ignore-unpack-errors <ブール値>

テンプレート抽出時のエラーを無視する。

---

--lock <バックアップ | 作成 | 破壊 | ディスク | fstrim | マイグレーション | マウント | ロールバック | スナップショット | スナップショット・削除>。

コンテナのロック/アンロック。

--メモリ <整数> (16 - N) (デフォルト = 512)

コンテナのRAM容量 (MB) 。

```
--mp[n]  [volume=<ボリューム>,mp=<パス>[,acl=<1|0>]
[,backup=<1|0>][,mountoptions=<opt[;opt...]>][,quota=<1|0>]
[,replicate=<1|0>][,ro=<1|0>][,shared=<1|0>]
[,size=<DiskS[quota=<1|0>][,replicate=<1|0>][,ro=<1|0>][,shared=<1|0>][,size=<DiskSize>]。
```

ボリュームをコンテナ・マウント・ポイントとして使用する。新しいボリュームを割り当てるには、特別な構文 `STORAGE_ID:SIZE_IN_GiB` を使用します。

## --ネームサーバー <文字列>

コンテナの DNS サーバー IP アドレスを設定する。`searchdomain` も `nameserver` も設定しなかった場合、`Create` は自動的にホストの設定を使用します。

```
--net[n] name=<string> [,bridge=<bridge>][,firewall=<1|0>]
[,gw=<GatewayIPv4>][,gw6=<GatewayIPv6>][,hwaddr=<XX:XX:XX:XX:XX:XX>]
[,ip=<(IPv4/CIDR|dhcp|manual)>]
[,ip6=<(IPv6/CIDR|auto|dhcp|manual)>][,link_down=<1|0>]
[,mtu=<integer>][,rate=<mbps>][,tag=<integer>]
[,trunks=<vlanid[;vlanid...]>][,type=<veth>]
```

コンテナのネットワーク・インターフェースを指定する。

## --onboot <boolean> (デフォルト = 0)

システム起動時にコンテナを起動するかどうかを指定する。

```
--ostype <alpine | archlinux | centos | debian | devuan | fedora |
gentoo | nixos | opensuse | ubuntu | unmanaged>.
```

OS タイプ。これは、コンテナ内部での設定に使用され、`/usr/share/lxc/config/<ostype>.common.conf` 内の `lxc` 設定スクリプトに対応する。値 `unmanaged` は、OS 固有のセットアップをスキップするため使用できる。

## --パスワード <password>

コンテナ内の root パスワードを設定する。

## --プール <文字列>

VM を指定したプールに追加する。

## --プロテクション <ブール値> (デフォルト = 0)

コンテナの保護フラグを設定する。これにより、CTまたはCTのディスクの削除/更新操作を防ぐことができる。

#### --restore <ブール値>

これを復元タスクとしてマークする。

```
--rootfs [<ボリューム名>] [<ボリューム名> [,acl=<1|0>]
[,mountoptions=<オプション[;オプション...]>] [,quota=<1|0>] [,replicate=<1|0>]
[,ro=<1|0>] [,shared=<1|0>]
[,size=<ディスクサイズ[quota=<1|0>]> [,replicate=<1|0>] [,ro=<1|0>] [,shared=<1|0>] [,size=<ディスクサイズ>]]。
```

ボリュームをコンテナ・ルートとして使用する。

#### --検索ドメイン <文字列>

コンテナのDNS検索ドメインを設定します。searchdomainもnameserverも設定しない場合、Createはホストからの設定を自動的に使用します。

**-ssh-公開鍵 <ファイルパス>**

公開SSH鍵を設定する（1行に1つの鍵、OpenSSH形式）。

**--start <boolean> (デフォルト = 0)**

CTの作成が正常に終了したら、CTを開始する。

**--startup `[[order=\d+] [,up=d+] [,down=d+]]` :**

スタートアップとシャットダウンの動作。Orderは、一般的な起動順序を定義する非負の数値である。

シャットダウンは逆の順序で行われます。さらに、次のVMが起動または停止するまでの待ち時間を秒単位で設定できます。

**--storage <ストレージID> (デフォルト = ローカル)**

デフォルトのストレージ。

**--スワップ <整数> (0 - N) (デフォルト = 512)**

コンテナのSWAP量（MB）。

**--タグ <文字列>**

コンテナのタグ。これはメタ情報に過ぎない。

**--テンプレート <ブール値> (デフォルト = 0)**

テンプレートの有効/無効。

**--タイムゾーン <文字列>**

コンテナで使用するタイムゾーン。オプションが設定されていない場合は、何も行われない。ホストのタイムゾーンに合わせるために host を設定するか、/usr/share/zoneinfo/zone.tab から任意のタイムゾーンオプションを設定することができる。

**--tty <整数> (0 - 6) (デフォルト = 2)**

コンテナで利用可能なttyの数を指定する。

**--unique <ブール値>**

一意のランダムなイーサネットアドレスを割り当てる。

## 注

必要なオプション: リストア

---

**--unprivileged <boolean>** (デフォルト = 0)

コンテナを非特権ユーザーとして実行させる。(手動で変更してはならない)。

**--未使用 [n] [ボリューム=]<ボリューム>**

未使用ボリュームへの参照。これは内部的に使用されるので、手動で変更してはならない。

**pct delsnapshot <vmid> <snapname> [OPTIONS]**。

LXCスナップショットを削除する。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**<スナップ名>: <文字列>**

スナップショットの名前。

**--force <ブール値>**

ディスクスナップショットの削除に失敗しても、設定ファイルから削除できる。

**pct destroy <vmid> [OPTIONS] [オプション]。**

コンテナを破棄する（使用中のファイルもすべて削除する）。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**-destroy-unreferenced-disks<boolean> (参照されないディスクを破棄する)。**

設定されている場合、設定内で参照されていないすべての有効なストレージから、VMIDを持つすべてのディスクを追加で破壊します。

**--force <ブール値> (デフォルト = 0)**

走っていても強制的に破壊する。

**--ページ <ブール値> (デフォルト = 0)**

関連するすべての構成からコンテナを削除する。例えば、バックアップジョブ、レプリケーションジョブ、HAなど。関連するACLとファイアウォールのエントリは常に削除される。

**pct df <vmid>**

コンテナの現在のディスク使用量を取得する。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**pct enter <vmid> [オプション] .**

指定したコンテナのシェルを起動する。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**--keep-env <boolean> (デフォルト = 1)**

現在の環境を維持する。このオプションはPVE 9ではデフォルトで無効になります。事前に提供された環境に依存している場合は、このオプションを使用して将来に備えてください。

**pct exec <vmid> [<extra-args>] [OPTIONS]**.

指定したコンテナ内でコマンドを起動する。

<vmid>: <整数> (100 - 999999999)

VMの（ユニークな）ID。

<extra-args>: <配列>。

配列としての追加引数

--keep-env <boolean> (デフォルト = 1)

現在の環境を維持する。このオプションはPVE 9ではデフォルトで無効になります。事前に提供された環境に依存している場合は、このオプションを使用して将来に備えてください。

**pct fsck** <vmid> [OPTIONS] .

コンテナ・ボリュームでファイルシステム・チェック  
(fsck) を実行する。

<vmid>: <整数> (100 - 999999999)

VMの（ユニークな）ID。

--device <mp0 | mp1 | mp10 | mp100 | mp101 | mp102 | mp103 | mp104 | mp105 | mp106 | mp107 | mp108 | mp109 | mp11 | mp110 | mp111 | mp112 | mp113 | mp114 | mp115 | mp116 | mp117 | mp118 | mp119 | mp12 | mp120 | mp121 | mp122 | mp123 | mp124 | mp125 | mp126 | mp127 | mp128 | mp129 | mp13 | mp130 | mp131 | mp132 | mp133 | mp134 | mp135 | mp136 | mp137 | mp138 | mp139 | mp14 | mp140 | mp141 | mp142 | mp143 | mp144 | mp145 | mp146 | mp147 | mp148 | mp149 | mp15 | mp150 | mp151 | mp152 | mp153 | mp154 | mp155 | mp156 | mp157 | mp158 | mp159 | mp16 | mp160 | mp161 | mp162 | mp163 | mp164 | mp165 | mp166 | mp167 | mp168 | mp169 | mp17 | mp170 | mp171 | mp172 | mp173 | mp174 | mp175 | mp176 | mp177 | mp178 | mp179 | mp173 | mp174 | mp175 | mp175 | mp175 | mp175 | mp175 | mp176 | mp177 | mp178 | mp179 | mp18 | mp180 | mp181 | mp182 | mp183 | mp184 | mp185 | mp186 | mp187 | mp188 | mp189 | mp19 | mp190 | mp191 | mp192 | mp193 | mp194 | mp195 | mp196 | mp197 | mp198 | mp199 | mp2 | mp20 | mp200 | mp201 | mp202 | mp203 | mp204 | mp205 | mp206 | mp207 | mp208 | mp209 | mp21 | mp210 | mp211 | mp212 | mp213 | mp214 | mp215 | mp216 | mp217 | mp218 | mp219 | mp22 | mp220 | mp221 | mp222 | mp223 | mp224 | mp225 | mp226 | mp227 | mp228 | MP229 | MP23 | MP230 | MP231 | MP232 | MP233 | MP234 | MP235 | MP236 | MP237 | MP238 | MP239 | MP24 | MP240 | MP241 | MP242 | MP243 | MP244 | MP242 | mp243 | mp244 | mp245 | mp246 | mp247 | mp248 | mp249 | mp25 | mp250 | mp251 | mp252 | mp253 | mp254 | mp255 | mp26 | mp27 | mp28

```
| mp29 | mp3 | mp30 | mp31 | mp32 | mp33 | mp34 | mp35 | mp36 | mp37 | mp38 |
| mp39 | mp4 | mp40 | mp41 | mp42 | mp43 | mp44 | mp45
| mp46 | mp47 | mp48 | mp49 | mp5 | mp50 | mp51 | mp52 | mp53 | mp54 | mp55 |
mp56 | mp57 | mp58 | mp59 | mp6 | mp60 | mp61 | mp62
| mp63 | mp64 | mp65 | mp66 | mp67 | mp68 | mp69 | mp7 | mp70 |
mp71 | mp72 | mp73 | mp74 | mp75 | mp76 | mp77 | mp78 | mp79 | mp8
| mp80 | mp81 | mp82 | mp83 | mp84 | mp85 | mp86 | mp87 | mp88 | mp89 | mp9 |
mp90 | mp91 | mp92 | mp93 | mp94 | mp95 | mp96 | mp97
| mp98 | mp99 | rootfs>
```

ファイルシステム・チェックを実行するボリューム。

**--force <ブール値> (デフォルト = 0)**

ファイルシステムがクリーンであるように見えても、強制的にチェックする。

**pct fstrim <vmid> [OPTIONS] [オプション]。**

選択したCTとそのマウントポイント（バインドまたは読み取り専用マウントポイントを除く）に対してfstrimを実行します。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**--ignore-mountpoints <ブール値>**

すべてのマウントポイントをスキップし、コンテナルート上のfstrimのみを実行する。

**pct help [OPTIONS]**

指定したコマンドに関するヘルプを表示する。

**--extra-args <array>**

特定のコマンドのヘルプを表示する

**--verbose <ブール値>**

冗長な出力形式。

## パーセントリスト

**pct listsnapshot <vmid> す**

べてのスナップショットを一覧

表示します。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**pct migrate <vmid> <target> [OPTIONS] (マイグレート <vmid> <target> [オプション])**

コンテナを別のノードにマイグレーションする。新しいマイグレーション・タスクを作成する。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**<ターゲット>: <文字列**  
ターゲット・ノード

**--bwlimit <number> (0 - N) (デフォルト = データセンターまたはストレージ設定からの  
マイグレーション制限)**

I/Oバンド幅の制限をオーバーライドする（単位：KiB/s）。

**--オンライン <ブール値>**

オンライン／ライブマイグレーションを利用する。

**--restart <ブール値>**

リスタート・マイグレーションを使用する

**-ターゲット・ストレージ <文字列>**

ソースストレージからターゲットストレージへのマッピング。単一のストレージIDのみを指定すると、すべてのソースストレージがそのストレージにマッピングされる。特別な値 1 を指定すると、各ソース・ストレージはそれ自体にマッピングされます。

**--タイムアウト <整数> (デフォルト = 180)**

再移行のためのシャットダウンのタイムアウト時間 (秒)

**pct マウント <vmid>**

コンテナのファイルシステムをホストにマウントする。これはコンテナに対するロックを保持し、コンテナに対する起動と停止以外の操作を防止するため、緊急時の保守のみを目的としている。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**pct move-volume <vmid> <volume> [<storage>] [<target-vmid>] [<target-volume>] [OPTIONS].**

rootfs-/mp-ボリュームを別のストレージまたは別のコンテナに移動する。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。



<ボリューム>: <mp0 | mp1 | mp10 | mp100 | mp101 | mp102 | mp103 | mp104 | mp105 | mp106 | mp107 | mp108 | mp109 | mp11 | mp110 | mp111 | mp112 | mp113 | mp114 | mp115 | mp116 | mp117 | mp118 | mp119 | mp12 | mp120 | mp121 | mp122 | mp123 | mp124 | mp125 | mp126 | mp127 | mp128 | mp129 | mp13 | mp130 | mp131 | mp132 | mp133 | mp134 | mp135 | mp136 | mp137 | mp138 | mp139 | mp14 | mp140 | mp141 | mp142 | mp143 | mp144 | mp145 | mp146 | mp147 | mp148 | mp149 | mp15 | mp150 | mp151 | mp152 | mp153 | mp154 | mp155 | mp156 | mp157 | mp158 | mp159 | mp16 | mp160 | mp161 | mp162 | mp163 | mp164 | mp165 | mp166 | mp167 | mp168 | mp169 | mp17 | mp170 | mp171 | mp172 | mp173 | mp174 | mp175 | mp176 | mp177 | mp178 | mp179 | mp18 | mp180 | mp181 | mp182 | mp183 | mp184 | mp185 | mp186 | mp187 | mp188 | mp189 | mp19 | mp190 | mp191 | mp192 | mp193 | mp194 | mp195 | mp196 | mp197 | mp198 | mp199 | mp2 | mp20 | mp200 | mp201 | mp202 | mp203 | mp204 | mp205 | mp206 | mp207 | mp208 | mp209 | mp21 | mp210 | mp211 | mp212 | mp213 | mp214 | mp215 | mp216 | mp217 | mp218 | mp219 | mp22 | mp220 | mp221 | mp222 | mp223 | mp224 | mp225 | mp226 | mp227 | mp228 | MP229 | MP23 | MP230 | MP231 | MP232 | MP233 | MP234 | MP235 | MP236 | MP237 | MP238 | MP239 | MP24 | MP240 | MP241 | MP242 | MP243 | MP244 | MP242 | mp243 | mp244 | mp245 | mp246 | mp247 | mp248 | mp249 | mp25 | mp250 | mp251 | mp252 | mp253 | mp254 | mp255 | mp26 | mp27 | mp28 | mp29 | mp3 | mp30 | mp31 | mp32 | mp33 | mp34 | mp35 | mp36 | mp37 | mp38 | mp39 | mp4 | mp40 | mp41 | mp42 | mp43 | mp44 | mp45 | mp46 | mp47 | mp48 | mp49 | mp5 | mp50 | mp51 | mp52 | mp53 | mp54 | mp55 | mp56 | mp57 | mp58 | mp59 | mp6 | mp60 | mp61 | mp62 | mp63 | mp64 | mp65 | mp66 | mp67 | mp68 | mp69 | mp7 | mp70 | mp71 | mp72 | mp73 | mp74 | mp75 | mp76 | mp77 | mp78 | mp79 | mp8 | mp80 | mp81 | mp82 | mp83 | mp84 | mp85 | mp86 | mp87 | mp88 | mp89 | mp9 | mp90 | mp91 | mp92 | mp93 | mp94 | mp95 | mp96 | mp97 | mp98 | mp99 | rootfs | unused0 | unused1 | unused10 | unused100 | unused101 | unused102 | unused103 | unused104 | unused105 | unused106 | unused107 | unused108 | unused109 | unused111 | unused110 | unused111 | unused112 | unused113 | unused114 | unused115 | unused116 | unused117 | unused118 | 未使用119 | 未使用12 | 未使用120 | 未使用121 | 未使用122 | 未使用123 | 未使用124 | 未使用125 | 未使用126 | 未使用127 | 未使用128 | 未使用129 | 未使用13 | 未使用130 | 未使用131 | 未使用132 | 未使用134 | 未使用135 | 未使用136 | 未使用137 | 未使用138 | 未使用139 | 未使用14 | 未使用140 | 未使用141 | 未使用142 | 未使用143

| 未使用144 | 未使用145 | 未使用146 | 未使用147 | 未使用148 | 未使用149 | 未使用15  
| 未使用150 | 未使用151 | 未使用152 | 未使用153 | 未使用154 | 未使用155 | 未使用  
156 | 未使用157 | 未使用158 | 未使用159 | 未使用16 | 未使用160 | 未使用161 | 未使用  
162 | 未使用163 | 未使用164 | 未使用145 | 未使用146 | 未使用147 | 未使用148 | 未使  
用149未使用161枚 | 未使用162枚 | 未使用163枚 | 未使用164枚 | 未使用165枚 | 未使用  
166枚 | 未使用167枚 | 未使用168枚 | 未使用169枚 | 未使用170枚 | 未使用171枚 | 未使  
用172枚 | 未使用173枚 | 未使用174枚 | 未使用175枚 | 未使用176枚 | 未使用177枚 | 未  
使用178枚 | 未使用179枚 | 未使用18枚 | 未使用180枚 | 未使用181枚 | 未使用182枚  
未使用183|未使用184|未使用185|未使用186|未使用187|

---

未使用188|未使用189|未使用19|未使用190|未使用191|未使用  
未使用192|未使用193|未使用194|未使用195|未使用196|未使用  
未使用197|未使用198|未使用199|未使用2|未使用20|未使用200

移動するボリューム。

**<ストレージ>: <ストレージID**

ターゲット・ストレージ。

**<ターゲット-vmid>: <整数> (100 - 99999999)**

VMの（ユニークな）ID。



<ターゲットボリューム>: <mp0 | mp1 | mp10 | mp100 | mp101 | mp102 | mp103 | mp104 | mp105 | mp106 | mp107 | mp108 | mp109 | mp11 | mp110MP111 | MP112 | MP113 | MP114 | MP115 | MP116 | MP117 | MP118 | MP119 | MP12 | MP120 | MP121 | MP122 | MP123 | MP124 | MP125 | MP126 | MP126 | MP127 | MP118 | MP119 | MP12 | MP121 | MP121 | MP122 | MP123 | MP124 | MP125 | MP126mp125 | mp126 | mp127 | mp128 | mp129 | mp13 | mp130 | mp131 | mp132 | mp133 | mp134 | mp135 | mp136 | mp137 | mp138 | mp139 | mp14 | mp140 | mp141 | mp142 | mp143 | mp144 | mp145 | mp146 | mp147 | mp148 | mp149 | mp15 | mp150 | mp151 | mp152 | mp153 | mp151 | mp152 | mp153 | mp153mp152 | mp153 | mp154 | mp155 | mp156 | mp157 | mp158 | mp159 | mp16 | mp160 | mp161 | mp162 | mp163 | mp164 | mp165 | mp166 | mp167 | mp168 | mp169 | mp17 | mp170 | mp171 | mp172 | mp173 | mp174 | mp175 | mp176 | mp177 | mp178 | mp179 | mp18 | mp180 | mp181 | mp182 | mp183 | mp184 | mp185 | mp186 | mp187 | mp188 | mp189 | mp19 | mp190 | mp191 | mp192 | mp193 | mp194 | mp195 | mp196 | mp197 | mp198 | mp199 | mp2 | mp20 | mp200 | mp201 | mp202 | mp203 | mp204 | mp205 | mp206 | mp207 | mp208 | mp209 | mp21 | mp210 | mp211 | mp212 | mp213 | mp214 | mp215 | mp216 | mp217 | mp218 | mp219 | mp22 | mp220 | mp221 | mp222 | mp223 | mp224 | mp225 | mp226 | mp227 | mp228 | mp229 | mp23 | mp230MP231 | MP232 | MP233 | MP234 | MP235 | MP236 | MP237 | MP238 | MP239 | MP24 | MP240 | MP241 | MP242 | MP243 | MP244 | MP245 | MP245 | MP225 | MP226 | MP227 | MP228 | MP229 | MP23 | MP230 | MP230mp244 | mp245 | mp246 | mp247 | mp248 | mp249 | mp25 | mp250 | mp251 | mp252 | mp253 | mp254 | mp255 | mp26 | mp27 | mp28 | mp29 | mp3 | mp30 | mp31 | mp32 | mp33 | mp34 | mp35 | mp36 | mp37 | mp38 | mp39 | mp4 | mp40 | mp41 | mp42 | mp43 | mp44 | mp45 | mp46 | mp47 | mp48 | mp49 | mp5 | mp50 | mp51 | mp52 | mp53 | mp54 | mp55 | mp56 | mp57 | mp58 | mp59 | mp6 | mp60 | mp61 | mp62 | mp63 | mp64 | mp65 | mp66 | mp67 | mp68 | mp69 | mp7 | mp70 | mp71 | mp72 | mp73 | mp74 | mp75 | mp76 | mp77 | mp78 | mp79 | mp8 | mp80 | mp81 | mp82 | mp83 | mp84 | mp85 | mp86 | mp87 | mp88 | mp89 | mp9 | mp90 | mp91 | mp92 | mp93 | mp94 | mp95 | mp96 | 未使用0 | 未使用1 | 未使用10 | 未使用100 | 未使用101 | 未使用102 | 未使用103 | 未使用104 | 未使用105 | 未使用106 | 未使用107 | 未使用108 | 未使用109 | 未使用11 | 未使用110 | 未使用111 | 未使用112 | 未使用113 | 未使用114 | 未使用115 | 未使用116 | 未使用117 | 未使用118 | 未使用119 | 未使用121 | 未使用122 | 未使用123 | 未使用124 |

未使用125|未使用126|未使用127|未使用128|未使用129|未使用13|未使用130|未使用131|未使用132|未使用133|未使用134|未使用135|未使用136|未使用137|未使用138|未使用未使用139|未使用14|未使用140|未使用141|未使用142|未使用143|未使用144|未使用145|未使用146|未使用147|未使用148|未使用149|未使用15|未使用150|未使用151|未使用152|未使用153|未使用154|未使用155|未使用156|未使用157|未使用158|未使用159|未使用16|未使用未使用160|未使用161|未使用162|未使用163|未使用164|未使用165|未使用166|未使用167|未使用168|未使用169|未使用17|未使用170|未使用171|未使用172|未使用173|未使用174|未使用175|未使用176|未使用177|未使用178|未使用179|未使用18|未使用180|未使用181|未使用

#### 未使用182|未使用183|未使用184|未使用185|未使用186|

---

未使用187|未使用188|未使用189|未使用19|未使用190|未使用  
未使用191|未使用192|未使用193|未使用194|未使用195|未使用  
未使用196|未使用197|未使用198|未使用199|未使用2|未使用20

ボリュームを移動させるコンフィグキー。デフォルトは移動元のボリュームキーです。

**--bwlimit <number> (0 - N) (デフォルト = データセンターまたはストレージ設定からのクローン制限)**

I/O バンド幅の制限をオーバーライドする (単位: KiB/s)。

**--delete <boolean> (デフォルト = 0)**

コピー成功後、元のボリュームを削除します。デフォルトでは、オリジナルは未使用のボリューム・エントリとして保持されます。

**-ダイジェスト <文字列>**

現在の設定ファイルの SHA1 " ." ダイジェストを使用する。これは同時変更を防ぐために使用できる。

**-ターゲット・ダイジェスト <文字列>**

ターゲット " ." コンテナの現在の設定ファイルが異なる SHA1 ダイジェストを持っている場合に変更を防ぐ。これは、" ." 同時変更を防ぐことができる。

**pct 移動量**

*pct move-volume* のエイリアス。

**pct 保留中 <vmid>**

保留中の変更を含むコンテナ構成を取得する。

**<vmid>: <整数> (100 - 999999999)**

VM の (ユニークな) ID。

**pct pull <vmid> <path> <destination> [OPTIONS] .**

コンテナからローカルシステムにファイルをコピーする。

**<vmid>: <整数> (100 - 999999999)**

VM の (ユニークな) ID。

**<path>: <文字列>**

プルするコンテナ内のファイルへのパス。

**<目的地>: <文字列**

目的地

**--グループ <文字列**

所有者のグループ名またはID。

**--perms <文字列**

使用するファイルパーミッション（デフォルトは8進数、16進数の場合は先頭に0xを付ける）。

## --ユーザー <文字列>

所有者のユーザー名またはID。

**pct push** <vmid> <file> <destination> [OPTIONS] [オプション] .

ローカルファイルをコンテナにコ  
ピーする。

### <vmid>: <整数> (100 - 999999999)

VMの（ユニークな）ID。

## <ファイル>: <文字列>

ローカルファイルへのパス。

### <目的地>: <文字列>

コンテナ内の書き込み先。

## --グループ <文字列>

オーナー・グループ名またはID。名前を使用する場合は、コンテナ内に存在する必要があります。

## --perms <文字列>

使用するファイルパーミッション（デフォルトは8進数、16進数の場合は先頭に0xを付ける）。

## --ユーザー <文字列>

所有者のユーザー名またはID。名前を使用する場合は、コンテナ内に存在する必要があります。

**pct reboot** <vmid> [オプション]。

コンテナをシャットダウンして再起動する。保留中の変更を適用する。

### <vmid>: <整数> (100 - 999999999)

VMの（ユニークな）ID。

## --タイムアウト <整数> (0 - N)

シャットダウンまで最大タイムアウト秒数待つ。

**pct remote-migrate** <vmid> [<target-vmid>] <target-endpoint> --target-bridge <string> --target-storage <string> [OPTIONS].

コンテナをリモートクラスタに移行します。新しいマイグレーションタスクを作成します。EXPERIMENTAL 機能！

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**<ターゲット-vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

<ターゲットエンドポイント>: apitoken=<PVEAPIToken=user@realm! token=SECRET>。  
ホスト=<ADDRESS> [, フィンガープリント=<FINGERPRINT>] [, ポート=<PORT>]。  
リモート・ターゲット・エンドポイント

--bwlimit <integer> (0 - N) (デフォルト = データセンターまたはストレージ設定から  
のマイグレーション制限)

I/O バンド幅の制限をオーバーライドする (単位: KiB/s)。

--delete <boolean> (デフォルト = 0)

移行成功後、元の CT と関連データを削除します。デフォルトでは、元の CT は停止状態で移行元クラス  
タに保持されます。

--オンライン <ブール値

オンライン／ライブマイグレーションを利用する。

--restart <ブール値

リスタート・マイグレーションを使用する

--ターゲット・ブリッジ <文字列>

ソースブリッジからターゲットブリッジへのマッピング。単一のブリッジ ID だけを指定すると、すべて  
のソース・ブリッジがそのブリッジにマッピングされます。特別な値 1 を指定すると、各ソース・ブリ  
ッジはそれ自体にマッピングされます。

-ターゲット・ストレージ <文字列>

ソースストレージからターゲットストレージへのマッピング。単一のストレージ ID のみを指定すると  
、すべてのソースストレージがそのストレージにマッピングされる。特別な値 1 を指定すると、各ソ  
ース・ストレージはそれ自体にマッピングされます。

--タイムアウト <整数> (デフォルト = 180)

再移行のためのシャットダウンのタイムアウト時間 (秒)

pct rescan [OPTIONS]

すべてのストレージを再スキャンし、ディスクサイズと未使用ディスクイメージを更新する。

--ドライラン <ブール値> (デフォルト = 0)

実際にconfigに変更を書き出さないこと。

**--vmid <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**pct resize <vmid> <disk> <size> [OPTIONS] .**

コンテナマウントポイントのサイズを変更する。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

<ディスク>: <mp0 | mp1 | mp10 | mp100 | mp101 | mp102 | mp103 | mp104 | mp105 | mp106 | mp107 | mp108 | mp109 | mp11 | mp110 | mp111 | mp112 | mp113 | mp114 | mp115 | mp116 | mp117 | mp118 | mp119 | mp12 | mp120 | mp121 | mp122 | mp123 | mp124 | mp125 | mp126 | mp127 | mp128 | mp129 | mp13 | mp130 | mp131 | mp132 | mp123 | mp124 | mp125 | mp126 | mp127 | mp128 | mp129 | mp13 | mp130 | mp131 | mp132 | mp133 | mp134 | mp135 | mp136 | mp137 | mp138 | mp139 | mp14 | mp140 | mp141 | mp142 | mp143 | mp144 | mp145 | mp146 | mp147 | mp148 | mp149 | mp15 | mp150 | mp151 | mp152 | mp153 | mp154 | mp155 | mp156 | mp157 | mp158 | mp159 | mp16 | mp160 | | mp160MP161 | MP162 | MP163 | MP164 | MP165 | MP166 | MP167 | MP168 | MP169 | MP17 | MP170 | MP171 | MP172 | MP173 | MP174 | MP175 | MP175 | MP175 | MP170 | MP171 | MP172 | MP173 | MP174 | MP175 | MP175 | MP175MP174 | MP175 | MP176 | MP177 | MP178 | MP179 | MP18 | MP180 | MP181 | MP182 | MP183 | MP184 | MP185 | MP186 | MP187 | MP188 | MP186 | MP186mp187 | mp188 | mp189 | mp19 | mp190 | mp191 | mp192 | mp193 | mp194 | mp195 | mp196 | mp197 | mp198 | mp199 | mp2 | | mp20 | mp200 | mp201 | mp202 | mp203 | mp204 | mp205 | mp206 | mp207 | mp208 | mp209 | mp21 | mp210 | mp211 | mp212 | mp213 | mp214 | mp215 | mp216 | mp217 | mp218 | mp219 | mp22 | mp220 | mp221 | mp222 | mp223 | mp224 | mp225 | mp226 | mp227 | mp228 | MP229 | MP23 | MP230 | MP231 | MP232 | MP233 | MP234 | MP235 | MP236 | MP237 | MP238 | MP239 | MP24 | MP240 | MP241 | MP242 | MP243 | MP244 | MP242mp243 | mp244 | mp245 | mp246 | mp247 | mp248 | mp249 | mp25 | mp250 | mp251 | mp252 | mp253 | mp254 | mp255 | mp26 | mp27 | mp28 | | mp29 | mp3 | mp30 | mp31 | mp32 | mp33 | mp34 | mp35 | mp36 | mp37 | mp38 | mp39 | mp4 | mp40 | mp41 | mp42 | mp43 | mp44 | mp45 | | mp46 | mp47 | mp48 | mp49 | mp5 | mp50 | mp51 | mp52 | mp53 | mp54 | mp55 | mp56 | mp57 | mp58 | mp59 | mp6 | mp60 | mp61 | mp62 | | mp63 | mp64 | mp65 | mp66 | mp67 | mp68 | mp69 | mp7 | mp70 | mp71 | mp72 | mp73 | mp74 | mp75 | mp76 | mp77 | mp78 | mp79 | mp8 | | mp80 | mp81 | mp82 | mp83 | mp84 | mp85 | mp86 | mp87 | mp88 | mp89 | mp9 | mp90 | mp91 | mp92 | mp93 | mp94 | mp95 | mp96 | mp97 | | mp98 | mp99 | rootfs>

リサイズしたいディスク。

<サイズ>です: \+?\d+(\.\d+)? [KMGT]?

新しいサイズ。記号を付けると、その値はボリュームの実際のサイズに加算され、付けない場合は絶対値となります。ディスク・サイズの縮小はサポートされていません。

### -ダイジェスト <文字列>

現在のコンフィギュレーションファイルのSHA1ダイジェストが異なる場合、変更を防止する。これは同時変更を防ぐために使用できます。

**pct restore** <vmid> <ostemplate> [OPTIONS] [オプション]。

コンテナを作成または復元する。

<vmid>: <整数> (100 - 999999999)

VMの（ユニークな）ID。

**<osTemplate>: <文字列>**

OSのテンプレートまたはバックアップファイル。

**--arch <amd64 | arm64 | armhf | i386 | riscv32 | riscv64> (デフォルト = amd64)**

OSのアーキテクチャタイプ。

**--bwlimit <number> (0 - N) (デフォルト = データセンターまたはストレージ設定から制限を復元)**

I/Oバンド幅の制限をオーバーライドする（単位：KiB/s）。

**--cmode <コンソール | シェル | tty> (デフォルト = tty)**

コンソールモード。デフォルトでは、consoleコマンドは利用可能なttyデバイスの1つに接続を開こうとします。cmodeをconsoleに設定すると、代わりに/dev/consoleにアタッチしようとする。cmodeをshellに設定すると、単にコンテナ内でシェルを起動する（ログインはしない）。

**--コンソール <ブール値> (デフォルト = 1)**

コンテナにコンソールデバイス（/dev/console）をアタッチする。

**--コア <整数> (1 - 8192)**

コンテナに割り当てられたコアの数。コンテナは、デフォルトで使用可能なすべてのコアを使用できる。

**--cpulimit <number> (0 - 8192) (デフォルト = 0)**

CPU使用量の上限。

---

**注**

コンピュータに2つのCPUがある場合、合計2つのCPU時間を持つ。値0はCPU制限なしを示す。

---

**--cpuunits <integer> (0 - 500000) (デフォルト = cgroup v1: 1024, cgroup v2: 100)**

コンテナのCPU重量は、cgroup v2では[1, 10000]にクランプされる。

**--debug <ブール値> (デフォルト = 0)**

もっと冗長にしてみてください。今のところ、これは起動時にデバッグログレベルを有効にするだけです。

---

## --説明 <文字列>

コンテナの説明。ウェブインターフェイスCTのサマリーに表示されます。設定ファイルのコメントとして保存されます。

--dev[n] [[path=] <パス> [gid=<integer>] [,mode=<オクタルのアクセス

モード>] [,uid=<integer>]。

コンテナへの通過装置

--features [force\_rw\_sys=<1|0>] [,fuse=<1|0>] [,keyctl=<1|0>]

[,mknod=<1|0>] [,mount=<fstype;fstype;...>] [,nesting=<1|0>] とする。

コンテナが高度な機能にアクセスできるようにする。

**--force <ブール値>**

既存のコンテナの上書きを許可する。

**--hookscript <文字列>**

コンテナのライフタイムのさまざまな段階で実行されるスクリプト。

**--ホスト名 <文字列>**

コンテナのホスト名を設定する。

**--ignore-unpack-errors <ブール値>**

テンプレート抽出時のエラーを無視する。

**--lock <バックアップ | 作成 | 破棄 | ディスク | fstrim | マイグレーション | マウント | ロールバック | スナップショット | スナップショット・削除>。**

コンテナのロック/アンロック。

**--メモリ <整数> (16 - N) (デフォルト = 512)**

コンテナのRAM容量 (MB)。

**--mp[n] [volume=<ボリューム>, mp=<パス> [,acl=<1|0>] [,backup=<1|0>] [,mountoptions=<opt[;opt...]>] [,quota=<1|0>] [,replicate=<1|0>] [,ro=<1|0>] [,shared=<1|0>] [,size=<DiskS[quota=<1|0>][,replicate=<1|0>][,ro=<1|0>][,shared=<1|0>][,size=<DiskSize>]]。**

ボリュームをコンテナ・マウント・ポイントとして使用する。新しいボリュームを割り当てるには、特別な構文 STORAGE\_ID:SIZE\_IN\_GiB を使用します。

**--ネームサーバー <文字列>**

コンテナの DNS サーバー IP アドレスを設定する。searchdomainもnameserverも設定しなかった場合、Createは自動的にホストの設定を使用します。

**--net[n] name=<string> [,bridge=<bridge>] [,firewall=<1|0>] [,gw=<GatewayIPv4>] [,gw6=<GatewayIPv6>] [,hwaddr=<XX:XX:XX:XX:XX:XX>] [,ip=<(IPv4/CIDR|dhcp|manual)>] [,ip6=<(IPv6/CIDR|auto|dhcp|manual)>] [,link\_down=<1|0>] [,mtu=<integer>] [,rate=<mbps>] [,tag=<integer>] [,trunks=<vlanid[;vlanid...]>] [,type=<veth>]**

コンテナのネットワーク・インターフェースを指定する。

**--onboot <boolean> (デフォルト = 0)**

システム起動時にコンテナを起動するかどうかを指定する。

**--ostype <alpine | archlinux | centos | debian | devuan | fedora | gentoo | nixos | opensuse | ubuntu | unmanaged>.**

OSタイプ。これは、コンテナ内部での設定に使用され、`/usr/share/lxc/config/<ostype>.common.conf` 内の lxc 設定スクリプトに対応する。値 `unmanaged` は、OS 固有のセットアップをスキップするために使用できる。

**--パスワード <password>**

コンテナ内の root パスワードを設定する。

**--プール <文字列>**

VMを指定したプールに追加する。

**--プロテクション <プール値> (デフォルト = 0)**

コンテナの保護フラグを設定する。これにより、CTまたはCTのディスクの削除/更新操作を防ぐことができる。

```
--rootfs [ボリューム=]<ボリューム> [,acl=<1|0>]
[,mountoptions=<opt[;opt...]>] [,quota=<1|0>] [,replicate=<1|0>]
[,ro=<1|0>] [,shared=<1|0>]
[,size=<DiskS[quota=<1|0>][,replicate=<1|0>][,ro=<1|0>][,shared=<1|0>][,size=<DiskSize>]。
```

ボリュームをコンテナ・ルートとして使用する。

**--検索ドメイン <文字列>**

コンテナのDNS検索ドメインを設定する。searchdomainもnameserverも設定しない場合、Createはホストからの設定を自動的に使用します。

**-ssh-公開鍵 <ファイルパス>**

公開SSH鍵を設定する（1行に1つの鍵、OpenSSH形式）。

**--start <boolean> (デフォルト = 0)**

CTの作成が正常に終了したら、CTを開始する。

**--startup `[[order=]\d+] [,up=d+] [,down=d+]`.**

スタートアップとシャットダウンの動作。Orderは、一般的な起動順序を定義する非負の数値である。

シャットダウンは逆の順序で行われます。さらに、次のVMが起動または停止するまでの待ち時間を秒単位で設定できます。

**--storage <ストレージID> (デフォルト = ローカル)**

デフォルトのストレージ。

**--スワップ <整数> (0 - N) (デフォルト = 512)**

コンテナのSWAP量 (MB)。

**--タグ <文字列>**

コンテナのタグ。これはメタ情報に過ぎない。

**--テンプレート <ブール値> (デフォルト = 0)**

テンプレートの有効/無効。

**--タイムゾーン <文字列>**

コンテナで使用するタイムゾーン。オプションが設定されていない場合は、何も行われない。ホストのタイムゾーンに合わせるために *host* を設定するか、/usr/share/zoneinfo/zone.tab から任意のタイムゾーンオプションを設定することができる。

**--tty <整数> (0 - 6) (デフォルト = 2)**

コンテナで利用可能なttyの数を指定する。

**--unique <ブール値>**

一意のランダムなイーサネットアドレスを割り当てる。

---

**注**

必要なオプション: リストア

---

**--unprivileged <boolean> (デフォルト = 0)**

コンテナを非特権ユーザーとして実行させる。(手動で変更してはならない)。

**--未使用 [n] [ボリューム=] <ボリューム>**

未使用ボリュームへの参照。これは内部的に使用されるので、手動で変更してはならない。

**pct レジューム <vmid>**

コンテナを再開する。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**pct rollback <vmid> <スナップ名> [オプション]。**

LXCの状態を指定したスナップショットにロールバックします。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**<スナップ名>: <文字列>**

スナップショットの名前。

**--start <boolean> (デフォルト = 0)**

ロールバック成功後にコンテナを起動するかどうか

**pct set <vmid> [OPTIONS] [オプション] .**

コンテナのオプションを設定する。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**--arch <amd64 | arm64 | armhf | i386 | riscv32 | riscv64> (デフォルト = amd64)**

OSのアーキテクチャタイプ。

**--cmode <コンソール | シェル | tty> (デフォルト = tty)**

コンソールモード。デフォルトでは、consoleコマンドは利用可能なttyデバイスの1つに接続を開こうとします。cmodeをconsoleに設定すると、代わりに /dev/consoleにアタッチしようとする。cmodeをshellに設定すると、単にコンテナ内でシェルを起動する（ログインはしない）。

**--コンソール <ブール値> (デフォルト = 1)**

コンテナにコンソールデバイス (/dev/console) をアタッチする。

**--コア <整数> (1 - 8192)**

コンテナに割り当てられたコアの数。コンテナは、デフォルトで使用可能なすべてのコアを使用できる。

**--cpulimit <number> (0 - 8192) (デフォルト = 0)**

CPU使用量の上限。

---

**注**

コンピュータに2つのCPUがある場合、合計2つのCPU時間を持つ。値0はCPU制限なしを示す。

---

**--cpuunits <integer> (0 - 500000) (デフォルト = cgroup v1: 1024, cgroup v2: 100)**

コンテナのCPU重量は、cgroup v2では[1, 10000]にクランプされる。

**--debug <ブール値> (デフォルト = 0)**

もっと冗長にしてみてください。今のところ、これは起動時にデバッグログレベルを有効にするだけです。

**--削除 <文字列>**

削除したい設定のリスト。

**--説明 <文字列>**

コンテナの説明。ウェブインターフェイスCTのサマリーに表示されます。設定ファイルのコメントとして保存されます。

**--dev[n] [[path=] <パス> [gid=<integer>] [,mode=<オクタルのアクセスモード>] [,uid=<integer>]]。**

コンテナへの通過装置

**-ダイジェスト <文字列>**

現在のコンフィギュレーションファイルのSHA1ダイジェストが異なる場合、変更を防止する。これは同時変更を防ぐために使用できます。

---

**--features [force\_rw\_sys=<1|0>] [,fuse=<1|0>] [,keyctl=<1|0>]  
[,mknod=<1|0>] [,mount=<fstype;fstype;...>] [,nesting=<1|0>]** とする。

コンテナが高度な機能にアクセスできるようにする。

**--hookscript <文字列>**

コンテナのライフタイムのさまざまなステップで実行されるスクリプト。

**--ホスト名 <文字列>**

コンテナのホスト名を設定する。

--lock <バックアップ | 作成 | 破壊 | ディスク | fstrim | マイグレーション | マウント | ロールバック | スナップショット | スナップショット・削除>。  
コンテナのロック/アンロック。

--メモリ <整数> (16 - N) (デフォルト = 512)

コンテナのRAM容量 (MB)。

--mp[n] [ボリューム =]<ボリューム> [,mp=<Path> [,acl=<1|0>]  
[,backup=<1|0>] [,mountoptions=<opt[:opt...]>] [,quota=<1|0>]  
[,replicate=<1|0>] [,ro=<1|0>] [,shared=<1|0>]  
[,size=<DiskS[quota=<1|0>][,replicate=<1|0>][,ro=<1|0>][,shared=<1|0>][,size=<DiskSize>]>]。

ボリュームをコンテナ・マウント・ポイントとして使用する。新しいボリュームを割り当てるには、特別な構文 STORAGE\_ID:SIZE\_IN\_GiB を使用します。

--ネームサーバー <文字列>

コンテナの DNS サーバー IP アドレスを設定する。searchdomainもnameserverも設定しなかった場合、Createは自動的にホストの設定を使用します。

--net[n] name=<string> [,bridge=<bridge>] [,firewall=<1|0>]  
[,gw=<GatewayIPv4>] [,gw6=<GatewayIPv6>] [,hwaddr=<XX:XX:XX:XX:XX:  
XX>] [,ip=<(IPv4/CIDR|dhcp|manual)>]  
[,ip6=<(IPv6/CIDR|auto|dhcp|manual)>] [,link\_down=<1|0>]  
[,mtu=<integer>] [,rate=<mbps>] [,tag=<integer>]  
[,trunks=<vlanid[:vlanid...]>] [,type=<veth>]  
コンテナのネットワーク・インターフェースを指定する。

--onboot <boolean> (デフォルト = 0)

システム起動時にコンテナを起動するかどうかを指定する。

--ostype <alpine | archlinux | centos | debian | devuan | fedora | gentoo | nixos | opensuse | ubuntu | unmanaged>.

OSタイプ。これは、コンテナ内部での設定に使用され、/usr/share/lxc/config/<ostype>.common.conf 内の lxc 設定スクリプトに対応する。値 *unmanaged* は、OS 固有のセットアップをスキップするため使用できる。

**--プロテクション <ブール値> (デフォルト = 0)**

コンテナの保護フラグを設定する。これにより、CTまたはCTのディスクの削除/更新操作を防ぐことができる。

**--復帰 <文字列>**

保留中の変更を取り消す。

```
--rootfs [ボリューム=]<ボリューム> [,acl=<1|0>]
[,mountoptions=<opt[;opt...]>] [,quota=<1|0>] [,replicate=<1|0>]
[,ro=<1|0>] [,shared=<1|0>]
[,size=<DiskS[quota=<1|0>][,replicate=<1|0>][,ro=<1|0>][,shared=<1|0>][,size=<DiskSize>]。
```

ボリュームをコンテナ・ルートとして使用する。

## --検索ドメイン <文字列>

コンテナのDNS検索ドメインを設定します。searchdomainもnameserverも設定しない場合、Createはホストからの設定を自動的に使用します。

## --startup `[[order=\d+] [,up=d+] [,down=d+]]`.

スタートアップとシャットダウンの動作。Orderは、一般的な起動順序を定義する非負の数値である。シャットダウンは逆の順序で行われます。さらに、次のVMが起動または停止するまでの待ち時間を秒単位で設定できます。

## --スワップ <整数> (0 - N) (デフォルト = 512)

コンテナのSWAP量 (MB)。

## --タグ <文字列>

コンテナのタグ。これはメタ情報に過ぎない。

## --テンプレート <ブール値> (デフォルト = 0)

テンプレートの有効/無効。

## --タイムゾーン <文字列>

コンテナで使用するタイムゾーン。オプションが設定されていない場合は、何も行われない。ホストのタイムゾーンに合わせるために host を設定するか、/usr/share/zoneinfo/zone.tab から任意のタイムゾーンオプションを設定することができる。

## --tty <整数> (0 - 6) (デフォルト = 2)

コンテナで利用可能なttyの数を指定する。

## --unprivileged <boolean> (デフォルト = 0)

コンテナを非特権ユーザーとして実行させる。(手動で変更してはならない)。

## --未使用 [n] [ボリューム=]<ボリューム>

未使用ボリュームへの参照。これは内部的に使用されるので、手動で変更してはならない。

**pct shutdown <vmid> [オプション]。**

コンテナをシャットダウンする。詳細は lxc-stop(1) を参照のこと。

**<vmid>**: <整数> (100 - 999999999)

VMの（ユニークな）ID。

**--forceStop** <boolean> (デフォルト = 0)

コンテナが停止していることを確認する。

**--タイムアウト** <整数> (0 - N) (デフォルト = 60)

最大タイムアウト秒待つ。

**pct snapshot** <vmid> <snapname> [OPTIONS]。

コンテナをスナップショットする。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**<スナップ名>: <文字列>**

スナップショットの名前。

**--説明 <文字列>**

テキストによる説明やコメント。

**pct start <vmid> [OPTIONS] [オプション]。**

コンテナをスタートさせる。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**--debug <ブール値> (デフォルト = 0)**

設定すると、起動時に非常に冗長なデバッグ・ログ・レベルを有効にする。

**--スキップロック <ブール値>**

Ignore locks - rootのみがこのオプションを使用できる。

**pct status <vmid> [OPTIONS] [オプション]。**

CTステータスを表示する。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**--verbose <ブール値>**

冗長出力フォーマット

**pct stop <vmid> [OPTIONS]。**

コンテナを停止する。これにより、コンテナ内で実行中のすべてのプロセスが突然停止する。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**--overrule-shutdown <boolean> (デフォルト = 0)**

停止する前に、アクティブなvzshutdownタスクを中止する。

**--スキップロック <ブール値>**

Ignore locks - rootのみがこのオプションを使用できる。

**pct サスPEND <vmid>**

コンテナを一時停止する。これは実験的なものである。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**pct テンプレート <vmid>**

テンプレートを作成する。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**pct アンロック <vmid>**

VMのロックを解除する。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

**pct アンマウント <vmid>**

コンテナのファイルシステムをアンマウントする。

**<vmid>: <整数> (100 - 999999999)**

VMの（ユニークな）ID。

## A.11 pveam - Proxmox VE アプライアンスマネージャ

**pveam <COMMAND> [ARGS] [OPTIONS]**.

**pveamあり [OPTIONS]**

利用可能なテンプレートを一覧表示します。

**--section <メール | システム | turnkeylinux>**

リストを指定したセクションに制限する。

**pveam ダウンロード <ストレージ> <テンプレート**

アプライアンス・テンプレートのダウンロード

**<ストレージ>: <ストレージID**  
テンプレートが保存されるストレージ

**<テンプレート>: <文字列**  
ダウンロードされるテンプレート

**pveam help [OPTIONS]**

指定したコマンドに関するヘルプを表示する。

**--extra-args <array>**

特定のコマンドのヘルプを表示する

**--verbose <ブール値>**

冗長な出力形式。

**pveamリスト <ストレージ>**

ストレージ上の全テンプレートのリストを取得する

**<ストレージ>: <ストレージID>**

指定したストレージ上のテンプレートのみをリストアップする

**pveam remove <template\_path>**

テンプレートを削除します。

**<template\_path>: <文字列>**

削除するテンプレート。

**pveamアップデート**

コンテナテンプレートデータベースを更新する。

## A.12 pvecm - Proxmox VE クラスタマネージャ

**pvecm <COMMAND> [ARGS] [OPTIONS].**

**pvecm add <ホスト名> [OPTIONS].**

現在のノードを既存のクラスタに追加します。

**<ホスト名>: <文字列>**

既存のクラスタ・メンバーのホスト名（またはIP）。

**--fingerprint ([A-Fa-f0-9]{2}:){31}[A-Fa-f0-9]{2}**

証明書の SHA 256 フィンガープリント。

**--force <ブール値>**

ノードが既に存在する場合はエラーをスローしない。

**-リンク [n] [アドレス=><IP> [,優先度=<整数>]]。**

1つのコロシンク・リンクのアドレスとプライオリティ情報。(最大8リンクまでサポート; link0..link7)

**--nodeid <整数> (1 - N)**

このノードのノード ID。

**--use\_ssh <ブール値**

たとえ相手がAPI経由で行うとしても、参加するには常にSSHを使用すること。

**--votes <整数> (0 - N)**

このノードの投票数

**pvecm addnode <ノード> [オプション].**

クラスタ構成にノードを追加します。この呼び出しは内部使用です。

**<ノード>: <文字列**

クラスタ・ノード名。

**--apiversion <整数**

新しいノードのJOIN\_API\_VERSION。

**--force <ブール値**

ノードが既に存在する場合はエラーをスローしない。

**-リンク [n] [アドレス=]<IP> [,優先度=<整数>]。**

1つのコロシンク・リンクのアドレスとプライオリティ情報。(最大8リンクまでサポート; link0..link7)

**--new\_node\_ip <文字列**

追加するノードのIPアドレス。リンクが指定されなかった場合のフォールバックとして使用されます。

**--nodeid <整数> (1 - N)**

このノードのノード ID。

**--votes <整数> (0 - N)**

このノードの投票数

**pvecm アピバー**

このノードで利用可能なクラスタ参加APIのバージョンを返します。

**pvecm create <クラスター名> [OPTIONS] [オプション].**

新しいクラスタ構成を生成します。リンクが指定されていない場合は、デフォルトでローカルIPアドレスがlink0になります。

#### <クラスター名>: <文字列>

クラスタの名前。

#### -リンク [n] [アドレス=<IP> [, 優先度=<整数>].

1つのコロシンク・リンクのアドレスとプライオリティ情報。(最大8リンクまでサポート; link0..link7)

**--nodeid <整数> (1 - N)**

このノードのノード ID。

**--投票 <整数> (1 - N)**

このノードの投票数。

**pvecm delnode <ノード>**

クラスタ構成からノードを削除します。

**<ノード>: <文字列**

クラスタ・ノード名。

**pvecm 予想 <予想**

corosyncに期待される投票数の新しい値を伝える。

**<予想>: <整数> (1 - N)**

予想得票数

**pvecm help [OPTIONS]**

指定したコマンドに関するヘルプを表示する。

**--extra-args <array>**

特定のコマンドのヘルプを表示する

**--verbose <ブール値**

冗長な出力形式。

**pvecm keygen <ファイル名>**

corosync用の新しい暗号鍵を生成する。

**<ファイル名>: <文字列**

出力ファイル名

**pvecm mtunnel [<extra-args>] [OPTIONS] .**

VM/CTマイグレーションで使用する。

**<extra-args>**: <配列>  
配列としての追加引数

**--get\_migration\_ip** <boolean> (デフォルト = 0)

設定されている場合、マイグレーションIPを返す

**--マイグレーション・ネットワーク <文字列>**

ローカルマイグレーションIPの検出に使用されるマイグレーションネットワーク

**--run-command <ブール値>**

tcpソケットを標準入力としてコマンドを実行する。IPアドレスとポートが次のように表示される。

コマンドの標準出力を、それぞれ別個の行で最初に出力する。

**pvecmノード**

クラスタノードのローカルビューを表示します。

**pvecm qdevice remove**

設定されたQDeviceを削除する

**pvecm qdevice setup <アドレス> [オプション] .**

QDeviceの使用設定

**<アドレス>: <文字列>**

外部corosync QDeviceのネットワークアドレスを指定します。

**--force <ブール値>**

可能性のある危険な操作に対してエラーを投げないこと。

**--ネットワーク <文字列>**

外部qdeviceへの接続に使用するネットワーク。

**pvecmステータス**

クラスタステータスのローカルビューを表示します。

**pvecm updatecerts [OPTIONS] [オプション] .**

ノード証明書を更新する（必要なファイル/ディレクトリをすべて生成する）。

**--force <ブール値>**

新しいSSL証明書を強制的に生成する。

**--サイレント <ブール値>**

エラーを無視する（クラスタに定足数がない場合など）。

**--unmerge-known-hosts <boolean> (デフォルト = 0)**

レガシーSSHの既知のホストをアンマージする。

## A.13 pvesr - Proxmox VE ストレージレプリケーション

**pvesr** <COMMAND> [ARGS] [OPTIONS].

**pvesr create-local-job** <id> <target> [OPTIONS].

新しいレプリケーション・ジョブを作成する

**<id>: [1-9][0-9]{2,8}-\d{1,9}**

レプリケーション・ジョブID。IDはゲストIDとジョブ番号で構成され、ハイフンで区切られます。

<ゲスト>-<ジョブ名>.

**<ターゲット>: <文字列**

ターゲット・ノード

**--コメント <文字列**

説明

**--無効 <ブール値**

エントリーを無効化/非活性化するフラグ。

**--レート <数値> (1 - N)**

mbps (メガバイト/秒) 単位の浮動小数点数でのレート制限。

**--remove\_job <full | local>.**

レプリケーション・ジョブを削除するようにマークします。このジョブはすべてのローカルレプリケーションスナップショットを削除します。フルに設定すると、ターゲット上のレプリケートされたボリュームも削除しようとなります。その後、ジョブは設定ファイルから自身を削除します。

**--スケジュール <文字列> (デフォルト =\*/15)**

ストレージのレプリケーションスケジュール。このフォーマットは systemd カレンダーイベントのサブセットです。

**--ソース <文字列**

ゲストが盗まれたかどうかを検出するための内部使用。

**pvesr delete <id> [OPTIONS] .**

レプリケーションジョブに削除マークを付けます。

**<id>: [1-9][0-9]{2,8}-\d{1,9}**

レプリケーション・ジョブID。IDはゲストIDとジョブ番号で構成され、ハイフンで区切られます。

<ゲスト>-<ジョブ名>.

**--force <ブール値> (デフォルト = 0)**

jobconfigエントリーは削除されるが、クリーンアップはされない。

**--keep <boolean> (デフォルト = 0)**

レプリケートされたデータをターゲットに残す (削除しない)。

**pvesr disable <id>**

レプリケーション・ジョブを無効にします。

**<id>: [1-9][0-9]{2,8}-\d{1,9}**

レプリケーション・ジョブID。IDはゲストIDとジョブ番号で構成され、ハイフンで区切られます。

<ゲスト>-<ジョブ名>.

**pvesr enable <id>**

レプリケーション・ジョブを有効にする。

<id>: [1-9][0-9]{2,8}-\d{1,9}

レプリケーション・ジョブID。IDはゲストIDとジョブ番号で構成され、ハイフンで区切られます。

<ゲスト>-<ジョブ名>.

**pvesr finalize-local-job <id> [<extra-args>] [OPTIONS].**

レプリケーション・ジョブを確定します。これにより、<last\_sync>と異なるタイムスタンプを持つすべてのレプリケーション・スナップショットが削除されます。

<id>: [1-9][0-9]{2,8}-\d{1,9}

レプリケーション・ジョブID。IDはゲストIDとジョブ番号で構成され、ハイフンで区切られます。

<ゲスト>-<ジョブ名>.

<extra-args>: <配列>。

検討するボリュームIDのリスト。

--last\_sync <整数> (0 - N)

最後に同期が成功した時刻（UNIXエポック）。指定しない場合は、すべてのレプリケーション・スナップショットが削除されます。

**pvesr help [OPTIONS]**

指定したコマンドに関するヘルプを表示する。

--extra-args <array>

特定のコマンドのヘルプを表示する

--verbose <ブール値>

冗長な出力形式。

**pvesrリスト**

レプリケーション・ジョブを一覧表示します。

**pvesr prepare-local-job <id> [<extra-args>] [OPTIONS].**

レプリケーション・ジョブの開始準備。これはレプリケーション開始前にターゲットノードで呼び出されます。この呼び出しは内部用で、標準出力にJSONオブジェクトを返します。このメソッドはまず、VM

<vmid>がローカルノードに存在するかどうかをテストします。存在する場合は直ちに停止する。その後、このメソッドはスナップショットについてすべてのボリュームIDをスキャンし、<last\_sync>と異なるタイムスタンプを持つすべてのレプリケーション・スナップショットを削除します。また、未使用のボリュームも削除する。既存のレプリケーション・スナップショットを持つすべてのボリュームのブーリアン・マーカーを持つハッシュを返します。

**<id>:** [1-9] [0-9]{2,8}-\d{1,9}

レプリケーション・ジョブID。IDはゲストIDとジョブ番号で構成され、ハイフンで区切られます。

<ゲスト>-<ジョブ名>.

**<extra-args>:** <配列>。

検討するボリュームIDのリスト。

**--force <ブール値> (デフォルト = 0)**

存在するすべてのボリューム（空のボリュームリスト）の削除を許可する。

**--last\_sync <整数> (0 - N)**

最後に同期が成功した時刻（UNIXエポック）。指定しない場合は、すべてのレプリケーション・スナップショットが削除されます。

**-親スナップ名 <文字列>**

スナップショットの名前。

**--スキャン <文字列>**

古くなったボリュームをスキャンするストレージIDのリスト。

**pvesr リード <id>**

レプリケーション・ジョブの設定を読み込む。

**<id>: [1-9][0-9]{2,8}-\d{1,9}**

レプリケーション・ジョブID。IDはゲストIDとジョブ番号で構成され、ハイフンで区切られます。

<ゲスト>-<ジョブ名>.

**pvesr run [OPTIONS]**

このメソッドはsystemd-timerによって呼び出され、すべての（または特定の）同期ジョブを実行する。

**--id [1-9][0-9]{2,8}-\d{1,9}**

レプリケーション・ジョブID。IDはゲストIDとジョブ番号で構成され、ハイフンで区切られます。

<ゲスト>-<ジョブ名>.

**--メール <ブール値> (デフォルト = 0)**

障害が発生した場合、電子メールによる通知を送信します。

**--verbose <boolean> (デフォルト = 0)**

標準出力に詳細なログを出力する。

**pvesr schedule-now <id>**

レプリケーション・ジョブをできるだけ早く開始するようにスケジュールする。

**<id>: [1-9][0-9]{2,8}-\d{1,9}**

レプリケーション・ジョブID。IDはゲストIDとジョブ番号で構成され、ハイフンで区切られます。

<ゲスト>-<ジョブ名>.

**pvesr set-state** <vmid> <state>

マイグレーション時にジョブのレプリケーション状態を設定します。この呼び出しは内部用です。これはJSON objとしてジョブ状態を受け取ります。

**<vmid>: <整数> (100 - 99999999)**

VMの（ユニークな）ID。

**<状態>: <文字列**

JSONデコードされた文字列としてのジョブ状態。

**pvesr status [OPTIONS]**

このノードのすべてのレプリケーション・ジョブのステータスを一覧表示します。

**--ゲスト <整数> (100 - 99999999)**

このゲストのレプリケーション・ジョブのみをリストします。

**pvesr update <id> [OPTIONS] .**

レプリケーション・ジョブの設定を更新します。

**<id>: [1-9][0-9]{2,8}-\d{1,9}**

レプリケーション・ジョブID。IDはゲストIDとジョブ番号で構成され、ハイフンで区切られます。

<ゲスト>-<ジョブ名>.

**--コメント <文字列**

説明

**--削除 <文字列**

削除したい設定のリスト。

**-ダイジェスト <文字列**

現在のコンフィギュレーション・ファイルのダイジェストが異なる場合、変更を防ぐ。これは、現在のコンフィギュレーション・ファイルの変更を防ぐために使用できます。

**--無効 <ブール値**

エントリーを無効化/非活性化するフラグ。

**--レート <数値> (1 - N)**

Mbps (メガバイト/秒) 単位の浮動小数点数でのレート制限。

**--remove\_job <full | local>.**

レプリケーション・ジョブを削除するようにマークします。このジョブはすべてのローカルレプリケーションスナップショットを削除します。フルに設定すると、ターゲット上のレプリケートされたボリュームも削除しようとします。その後、ジョブは設定ファイルから自身を削除します。

**--スケジュール <文字列> (デフォルト =\*/15)**

ストレージのレプリケーションスケジュール。このフォーマットは `systemd` カレンダーイベントのサブセットです。

**--ソース <文字列**

ゲストが盗まれたかどうかを検出するための内部使用。

## A.14 pveum - Proxmox VE ユーザーマネージャ

**pveum** <COMMAND> [ARGS] [OPTIONS].

**pveum acl delete** <path> --roles <string> [OPTIONS].

アクセスコントロールリストの更新（パーミッションの追加または削除）。

**<path>: <文字列**

アクセス・コントロール・パス

**--グループ <文字列**

グループ一覧。

**--プロパゲート <ブール値> (デフォルト = 1)**

パーミッションの伝搬（継承）を許可する。

**--roles <文字列**

役割のリスト

**--トークン <文字列**

APIトークンのリスト。

**--users <文字列**

ユーザー一覧。

**pveumのaclリスト** [FORMAT\_OPTIONS]。

アクセス制御リスト（ACL）を取得する。

**pveum acl modify** <path> --roles <string> [OPTIONS].

アクセスコントロールリストの更新（パーミッションの追加または削除）。

**<path>: <文字列**

アクセス・コントロール・パス

**--グループ <文字列**

グループ一覧。

--プロパゲート <ブール値> (デフォルト = 1)

パーティションの伝搬（継承）を許可する。

--roles <文字列>

役割のリスト

--トークン <文字列>

APIトークンのリスト。

**--users <文字列>**

ユーザー一覧。

**プベウム・アクデル**

*pveum acl delete* のエイリアス。

**pveum aclmod**

*pveum acl modify* のエイリアス。

**pveum group add <groupid> [OPTIONS] [オプション] .**

新しいグループを作る。

**<groupid>: <文字列>**

記述なし

**--コメント <文字列>**

記述なし

**pveum グループ削除 <グループID>**

グループを削除する。

**<groupid>: <文字列>**

記述なし

**pveumグループリスト [FORMAT\_OPTIONS] .**

グループのインデックス。

**pveum group modify <groupid> [OPTIONS] [オプション] .**

グループデータを更新する。

**<groupid>: <文字列>**

記述なし

**--コメント <文字列>**

記述なし

**プベウム・グループ追加**

*pveum group add* のエイリアス。

## プベウム・グループデル

*pveum group delete* のエイリアス。

## プベウム・グループモッド

*pveum group modify* のエイリアス。

**pveum help [OPTIONS]**

指定したコマンドに関するヘルプを表示する。

**--extra-args <array>**

特定のコマンドのヘルプを表示する

**--verbose <ブール値>**

冗長な出力形式。

**pveum passwd <ユーザーID> [オプション]**.

ユーザーpasswordを変更する。

**<userid>: <文字列>**

name@realm形式の完全なユーザーID。

**--確認パスワード <文字列>**

変更を実行するユーザーの現在のパスワード。

**pveum pool add <poolid> [オプション]**

新しいプールを作る。

**<プール>: <文字列>**

記述なし

**--コメント <文字列>**

記述なし

**プール削除 <プール>**

プールを削除する。

**<プール>: <文字列>**

記述なし

**pveum プールリスト [OPTIONS] [FORMAT\_OPTIONS].**

プールの一覧表示またはプール設定の取得。

**--プール <文字列>**

記述なし

--タイプ <lx~~c~~ | qemu | storage>

記述なし

---

### 注

必要なオプション: poolid

---

**pveum pool modify** <poolid> [OPTIONS] [オプション].

プールを更新。

<プール>: <文字列

記述なし

--allow-move <boolean> (デフォルト = 0)

すでに別のプールにいる場合でも、ゲストを追加できるようにします。ゲストは現在のプールから削除され、このプールに追加されます。

--コメント <文字列

記述なし

--delete <boolean> (デフォルト = 0)

渡されたVMIDやストレージIDを追加する代わりに削除する。

--ストレージ <文字列

このプールから追加または削除するストレージIDのリスト。

--vms <文字列

このプールから追加または削除するゲストVMIDのリスト。

**pveum realm add** <realm> --type <string> [OPTIONS]

認証サーバを追加する。

<realm>: <文字列

認証ドメインID

--acr-values ^[^\x00-\xxx1F <>#]\*\$.

認証サーバが使用する認証コンテキスト・クラス参照値を指定する。

は認証リクエストに使用するようリクエストされた。

--autocreate <boolean> (デフォルト = 0)

ユーザーが存在しない場合、自動的にユーザーを作成します。

--base\_dn <文字列

LDAPベースドメイン名

--bind\_dn <文字列>  
LDAPバインドドメイン名

--capath <文字列> (デフォルト = /etc/ssl/certs)  
CA 証明書ストアへのパス

**--大文字小文字を区別する <論理値> (デフォルト = 1)**

ユーザー名は大文字と小文字を区別する

**--証明書 <文字列>**

クライアント証明書へのパス

**--認証キー <文字列>**

クライアント証明書キーへのパス

**--チェックコネクション <ブール値> (デフォルト = 0)**

サーバーへのバインド接続を確認する。

**--クライアントID <文字列>**

OpenIDクライアントID

**--client-key <文字列>**

OpenIDクライアント・キー

**--コメント <文字列>**

説明

**--default <ブール値>**

デフォルトのレルムとして使用する

**--ドメイン**

AD ドメイン名

**--フィルター <文字列>**

ユーザー同期用のLDAPフィルター。

**--group\_classes <string> (default = groupOfNames, group, univentionGroup, ipausergroup)**

グループのオブジェクトクラス。

**--group\_dn <文字列>**

グループ同期用のLDAPベース・ドメイン名。設定されていない場合は、base\_dnが使用されます。

**--グループ・フィルター <文字列>**

グループ同期用のLDAPフィルター。

**--group\_name\_attr <文字列>**

グループ名を表すLDAP属性。設定されていない場合、または見つかった場合は、DN の最初の値が name として使用されます。

**--issuer-url <文字列>**

OpenID発行者URL

--モード <ldap | ldap+starttls | ldaps> (デフォルト = ldap)

LDAPプロトコルモード。

--パスワード <文字列>

LDAP バインドパスワード。etc/pve/priv/realm/<REALM>.pw に格納されます。

--ポート <整数> (1 - 65535)

サーバーポート

--プロンプト (?::none|login|consent|select\_account|S+)

認証サーバがエンドユーザに再認証と同意を求めるかどうかを指定します。

--スコープ <文字列> (デフォルト = 電子メールプロファイル)

認可され、返されるべきスコープ(ユーザの詳細)を指定します(例えば、電子メールやプロフィールなど)。

--secure <ブール値>

安全な LDAPS プロトコルを使用する。廃止: 代わりに mode を使用してください。

--サーバー1 <文字列>

サーバーIPアドレス (またはDNS名)

--サーバー2 <文字列>

フォールバックサーバーのIPアドレス (またはDNS名)

-sslバージョン <tls v1 | tls v1\_1 | tls v1\_2 | tls v1\_3>.

LDAPS TLS/SSLのバージョン。1.2より古いバージョンを使用することは推奨されません!

-sync-defaults-options [enable-new=<1|0>] [,full=<1|0>]  
[,purge=<1|0>] [,remove-vanished=([acl];[properties];[entry])|none]  
[,scope=<users|groups|both>] [,scope=<users|groups|both>] ]。

同期の動作に関するデフォルトのオプション。

--sync\_attributes \w+=[^,]+(\s\*\w+=[^,]+)\*

どの LDAP 属性がどの PVE ユーザーに対応するかを指定するための key=value ペアのカンマ区切りリスト  
。

フィールドを指定します。例えば、LDAP 属性 mail を PVE の email にマッピングするには

`email=mail` と記述します。デフォルトでは、各 PVE ユーザーフィールドは同じ名前の LDAP 属性で表されます。

**--tfa type=<TFATYPE> [,digits=<COUNT>] [,id=<ID>] [,key=<KEY>]  
[,step=<SECONDS>] [,url=<URL>].**  
二要素認証を使用する。

**--type <ad | ldap | openid | pam | pve>.**  
レルムタイプ。

**--user\_attr &S{2,}.**  
LDAPユーザー属性名

**--user\_classes <string> (default = inetorgperson, posixaccount, person, user)**

ユーザー用のオブジェクトクラス。

**--username-claim <string>**

一意のユーザー名を生成するために使用されるOpenIDクレーム。

**--verify <ブール値> (デフォルト = 0)**

サーバーのSSL証明書を検証する

**pveum realm delete <realm>**

認証サーバーを削除します。

**<realm>: <文字列**

認証ドメインID

**pveum レルムリスト [FORMAT\_OPTIONS]。**

認証ドメインのインデックス。

**pveum realm modify <realm> [OPTIONS] [オプション]。**

認証サーバーの設定を更新します。

**<realm>: <文字列**

認証ドメインID

**--acr-values [^x00-xxx1F <>#"]\*\$.**

認証サーバが使用する認証コンテキスト・クラス参照値を指定する。

は認証リクエストに使用するようリクエストされた。

**--autocreate <boolean> (デフォルト = 0)**

ユーザーが存在しない場合、自動的にユーザーを作成します。

**--base\_dn <文字列**

LDAPベースドメイン名

**--bind\_dn <文字列**

LDAPバインドドメイン名

**--capath <文字列> (デフォルト = /etc/ssl/certs)**

CA 証明書ストアへのパス

--大文字小文字を区別する <論理値> (デフォルト = 1)

ユーザー名は大文字と小文字を区別する

--証明書 <文字列

クライアント証明書へのパス

**--認証キー <文字列>**

クライアント証明書キーへのパス

**--チェックコネクション <ブール値> (デフォルト = 0)**

サーバーへのバインド接続を確認する。

**--クライアントID <文字列>**

OpenIDクライアントID

**--client-key <文字列>**

OpenIDクライアント・キー

**--コメント <文字列>**

説明

**--default <ブール値>**

デフォルトのレルムとして使用する

**--削除 <文字列>**

削除したい設定のリスト。

**-ダイジェスト <文字列>**

現在のコンフィギュレーション・ファイルのダイジェストが異なる場合、変更を防ぐ。これは、現在のコンフィギュレーション・ファイルが変更されないようにするために使用できます。

**--ドメイン**

AD ドメイン名

**--フィルター <文字列>**

ユーザー同期用のLDAPフィルター。

**--group\_classes <string> (default = groupOfNames, group, univentionGroup, ipausergroup)**

グループのオブジェクトクラス。

**--group\_dn <文字列>**

グループ同期用のLDAPベース・ドメイン名。設定されていない場合は、base\_dnが使用されます。

**--グループ・フィルター <文字列>**

グループ同期用のLDAPフィルター。

**--group\_name\_attr <文字列>**

グループ名を表すLDAP属性。設定されていない場合、または見つかった場合は、DN の最初の値が name として使用されます。

**--issuer-url <文字列>**

OpenID発行者URL

--モード <ldap | ldap+starttls | ldaps> (デフォルト = ldap)

LDAPプロトコルモード。

--パスワード <文字列>

LDAP バインドパスワード。etc/pve/priv/realm/<REALM>.pw に格納されます。

--ポート <整数> (1 - 65535)

サーバーポート

--プロンプト (? : none | login | consent | select\_account | S +)

認証サーバがエンドユーザに再認証と同意を求めるかどうかを指定します。

--スコープ <文字列> (デフォルト = 電子メールプロファイル)

認可され、返されるべきスコープ(ユーザの詳細)を指定します(例えば、電子メールやプロフィールなど)。

--secure <ブール値>

安全な LDAPS プロトコルを使用する。廃止: 代わりに mode を使用してください。

--サーバー1 <文字列>

サーバーIPアドレス (またはDNS名)

--サーバー2 <文字列>

フォールバックサーバーのIPアドレス (またはDNS名)

-sslバージョン <tls v1 | tls v1\_1 | tls v1\_2 | tls v1\_3>.

LDAPS TLS/SSLのバージョン。1.2より古いバージョンを使用することは推奨されません!

-sync-defaults-options [enable-new=<1|0>] [,full=<1|0>]  
[,purge=<1|0>] [,remove-vanished=([acl];[properties];[entry])|none]

[,scope=<users|groups|both>] [,scope=<users|groups|both>] ]。

同期の動作に関するデフォルトのオプション。

--sync\_attributes \w+=[^,]+(,\s\*\w+=[^,]+)\*

どの LDAP 属性がどの PVE ユーザーに対応するかを指定するための key=value ペアのカンマ区切りリスト。

フィールドを指定します。例えば、LDAP 属性 mail を PVE の email にマッピングするには

`email=mail` と記述します。デフォルトでは、各 PVE ユーザーフィールドは同じ名前の LDAP 属性で表されます。

`--tfa type=<TFATYPE> [,digits=<COUNT>] [,id=<ID>] [,key=<KEY>]  
[,step=<SECONDS>] [,url=<URL>].`  
二要素認証を使用する。

`--user_attr &S{2,}.`  
LDAPユーザー属性名

`--user_classes <string> (default = inetorgperson, posixaccount, person,  
user)`  
ユーザー用のオブジェクトクラス。

**--verify <ブール値> (デフォルト = 0)**

サーバーのSSL証明書を検証する

**pveum realm sync <realm> [OPTIONS] [オプション]。**

設定されたLDAPからuser.cfgにユーザおよび/またはグループを同期します。注：同期されたグループは *name-\$realm* であるため、上書きを防ぐためにこれらのグループが存在しないことを確認してください。

**<realm>: <文字列>**

認証ドメインID

**--ドライラン <ブール値> (デフォルト = 0)**

もしセットされていれば、何も書き込まない。

**--enable-new <boolean> (デフォルト = 1)**

新しく同期したユーザーをすぐに有効にする。

**--full <ブール値>**

廃止: 代わりに *remove-vanished* を使用してください。設定されている場合、真実のソースとして LDAPディレクトリを使用し、同期から返されなかったユーザまたはグループを削除し、同期されたユーザのローカルで変更されたすべてのプロパティを削除します。設定されていない場合、同期されたデータに存在する情報のみが同期され、それ以外のものは削除または変更されません。

**--ページ <ブール値>**

廃止: 代わりに *remove-vanished* を使用してください。同期中に設定から削除されたユーザまたはグループの ACL を削除します。

**--remove-vanished ([acl]; [properties]; [entry]) | none (default = none)**

セミコロンで区切られたリストで、同期中にユーザーやグループが消えたときに削除されます。 *acl* は ユーザー/グループが同期から返されないときに *acl* を削除します。リストの代わりに *none* (デフォルト) を指定することもできます。

**--スコープ <両方 | グループ | ユーザー>**

同期するものを選択します。

**pveum role add <roleid> [OPTIONS] [オプション]。**

新しいロールを作成する。

**<roleid>: <文字列**

記述なし

**--privs <文字列**

記述なし

**pveumロール削除<ロールID>**

役割を削除する。

**<roleid>: <文字列>**

記述なし

**pveumロールリスト [FORMAT\_OPTIONS]。**

役割指標。

**pveum role modify <roleid> [OPTIONS] [オプション]。**

既存のロールを更新します。

**<roleid>: <文字列>**

記述なし

**--append <ブール値>**

記述なし

---

**備考**

必要なオプション: privs

---

**--privs <文字列>**

記述なし

**プベウム・ロールアド**

*pveum role add*のエイリアス。

**プベウム・ローデル**

*pveum role delete*のエイリアス。

**pveumロールモッド**

*pveum role modify*のエイリアス。

**pveum ticket <ユーザー名> [オプション] .**

認証チケットを作成または検証する。

---

<ユーザー名>: <文字列

ユーザー名

--new-format <boolean> (デフォルト = 1)

このパラメータは無視され、1とみなされる。

--otp <文字列

二要素認証用のワンタイムパスワード。

## --パス <文字列>

チケットを確認し、ユーザがパスにアクセス権限を持っているかチェックする。

---

### 備考

必要なオプション: privs

---

## --privs <文字列>

チケットを確認し、ユーザがパスにアクセス権限を持っているかチェックする。

---

### 注

必要なオプション: path

---

## --realm <文字列>

オプションで、このパラメータを使ってレルムを渡すことができます。通常、レルムは単にユーザ名 <username>@<realm> に追加されます。

## -tfa-チャレンジ <文字列>

ユーザーが応答したい署名済みTFAチャレンジ文字列。

**pveum user add <userid> [OPTIONS]。**

新しいユーザーを作成する。

### <userid>: <文字列>

name@realm形式の完全なユーザーID。

## --コメント <文字列>

記述なし

## --email <文字列>

記述なし

## --有効 <ブール値> (デフォルト = 1)

アカウントを有効にする（デフォルト）。アカウントを無効にするには、これを0に設定します。

## --expire <整数> (0 - N)

---

アカウントの有効期限（エポックからの秒数）。0は有効期限がないことを意味します。

--名 <文字列

記述なし

--グループ <文字列

記述なし

--keys [0-9a-zA-Z!=] {0,4096}。

二要素認証 (yubico) 用の鍵。

--ラストネーム <文字列

記述なし

--パスワード <文字列

初期パスワード

**pveum ユーザー削除** <ユーザーID>

ユーザーを削除します。

<userid>: <文字列

name@realm形式の完全なユーザーID。

**pveum ユーザーリスト** [OPTIONS] [FORMAT\_OPTIONS]

ユーザーインデックス

--有效 <ブール値>

enableプロパティのオプションフィルタ。

--full <ブール値> (デフォルト = 0)

グループとトークンの情報を含める。

**pveum user modify** <userid> [OPTIONS].

ユーザー設定を更新します。

<userid>: <文字列

name@realm形式の完全なユーザーID。

--append <ブール値>

記述なし

---

注

必要なオプション: グループ

---

--コメント <文字列

記述なし

--email <文字列

記述なし

**--有効 <ブール値> (デフォルト = 1)**

アカウントを有効にする（デフォルト）。アカウントを無効にするには、これを0に設定します。

**--expire <整数> (0 - N)**

アカウントの有効期限（エポックからの秒数）。0は有効期限がないことを意味します。

**--名 <文字列>**

記述なし

**--グループ <文字列>**

記述なし

**--keys [0-9a-zA-Z!=] {0,4096}。**

二要素認証（yubico）用の鍵。

**--ラストネーム <文字列>**

記述なし

**pveum user permissions [<userid>] [OPTIONS] [FORMAT\_OPTIONS].**

指定されたユーザー/トークンの有効なパーミッションを取得します。

**<userid>:**

(?^:^(?^:[^\s:/]+)\@(?^:[A-Za-z][A-Za-z0-9\.\-\_]+)(?:! (?^:[A-Za-z][A-Za-z0-

ユーザーIDまたは完全なAPIトークンID

**--パス <文字列>**

ツリー全体ではなく、特定のパスだけをダンプする。

**pveum user tfa delete <userid> [OPTIONS] [オプション]。**

ユーザーからTFAエントリーを削除する。

**<userid>:<文字列>**

name@realm形式の完全なユーザーID。

**--id <文字列>**

TFA ID。指定がない場合、すべてのTFAエントリーが削除される。

**pveumユーザーtfaリスト [<ユーザーID>]。**

TFAのエントリーをリストアップする。

**<userid>: <文字列**

name@realm形式の完全なユーザーID。

**pveumユーザーtfaロック解除<ユーザーID**

ユーザーのTFA認証を解除する。

**<userid>: <文字列**

name@realm形式の完全なユーザーID。

**pveum user token add <userid> <tokenid> [OPTIONS] [FORMAT\_OPTIONS].**

特定のユーザー用に新しいAPIトークンを生成する。注: APIトークンの値を返します。この値は後で取得できないため、保存しておく必要があります!

**<userid>: <文字列**

name@realm形式の完全なユーザーID。

**<tokenid>: (?^ : [A-Za-z] [A-Za-z0-9\.\-\_]+)**

ユーザー固有のトークン識別子。

**--コメント <文字列**

記述なし

**--expire <integer> (0 - N) (デフォルト = user と同じ)**

APIトークンの有効期限（エポックからの秒数）。0は有効期限がないことを意味します。

**--privsep <ブール値> (デフォルト = 1)**

個別のACLでAPIトークンの権限を制限するか（デフォルト）、対応するユーザーの全権限を与える。

**pveum ユーザー・トークン・リスト <ユーザーID> [FORMAT\_OPTIONS].**

ユーザー API トークンを取得します。

**<userid>: <文字列**

name@realm形式の完全なユーザーID。

**pveum user token modify <userid> <tokenid> [OPTIONS] [FORMAT\_OPTIONS].**

特定のユーザーのAPIトークンを更新します。

**<userid>: <文字列**

name@realm形式の完全なユーザーID。

<tokenid>: (?^:[A-Za-z][A-Za-z0-9\.\_]+)

ユーザー固有のトークン識別子。

--コメント <文字列

記述なし

**--expire <integer> (0 - N) (デフォルト = user と同じ)**

API トークンの有効期限（エポックからの秒数）。0 は有効期限がないことを意味します。

**--privsep <ブール値> (デフォルト = 1)**

個別の ACL で API トークンの権限を制限するか（デフォルト）、対応するユーザーの全権限を与える。

**pveum user token permissions <userid> <tokenid> [OPTIONS] [FORMAT\_OPTIONS].**

与えられたトークンの有効なパーミッションを取得します。

**<userid>: <文字列**

name@realm 形式の完全なユーザー ID。

**<tokenid>: (?^ : [A-Za-z] [A-Za-z0-9\.\-\_]+)**

ユーザー固有のトークン識別子。

**--パス <文字列**

ツリー全体ではなく、特定のパスだけをダンプする。

**pveum user token remove <userid> <tokenid> [FORMAT\_OPTIONS]。**

特定のユーザーの API トークンを削除します。

**<userid>: <文字列**

name@realm 形式の完全なユーザー ID。

**<tokenid>: (?^ : [A-Za-z] [A-Za-z0-9\.\-\_]+)**

ユーザー固有のトークン識別子。

## ユーザー追加

*pveum user add* のエイリアス。

## ユーザー削除

*pveum user delete* のエイリアス。

## pveum usermod

*pveum user modify* のエイリアス。

## A.15 **vzdump** - VMとコンテナのバックアップユーティリティ

### **vzdump**ヘルプ

```
vzdump {<vmid>} [OPTIONS].
```

バックアップを作成する。

**<vmid>: <文字列>**

バックアップしたいゲストシステムのID。

**--all <boolean> (デフォルト = 0)**

このホスト上のすべての既知のゲストシステムをバックアップする。

**--bwlimit <整数> (0 - N) (デフォルト = 0)**

I/Oバンド幅を制限する（単位：KiB/s）。

**--圧縮 <0 | 1 | gzip | lzo | zstd> (デフォルト = 0)**

ダンプファイルを圧縮する。

**--dumpdir <文字列>**

結果のファイルを指定したディレクトリに保存する。

**--除外 <文字列>**

指定したゲストシステムを除外する（--all を想定）

**--除外パス <配列>**

特定のファイル/ディレクトリ（シェル・グロブ）を除外する。で始まるパスはコンテナのルートに固定され、その他のパスは各サブディレクトリに相対的に一致する。

**--フリーク [[enabled=<1|0>] [[ストレージ=<ストレージID>]] [,ストレージ=<ストレージID>]]。**

バックアップフリートのオプション（VMのみ）。

**--ionice <整数> (0 - 8) (デフォルト = 7)**

BFQスケジューラ使用時にIOプライオリティを設定する。VMのスナップショットとサスペンドモードのバックアップでは、これはコンプレッサにのみ影響します。値が8の場合はアイドルプライオリティが使用され、それ以外の場合は指定した値でベストエフォートプライオリティが使用されます。

**--ロックウェイト <整数> (0 - N) (デフォルト = 180)**

グローバルロックの最大待機時間（分）。

**--メール通知 <常に | 失敗> (デフォルト = 常に)**

非推奨: 代わりに通知ターゲット/マッチャーを使用してください。通知メールを送るタイミングを指定する

**--電子メール <文字列**

非推奨: 代わりにnotification targets/matchersを使用してください。メール通知を受け取るメールアドレスまたはユーザのカンマ区切りリスト。

**-マックスファイル <整数> (1 - N)**

非推奨: 代わりに *prune-backups* を使用してください。ゲストシステムごとのバックアップファイルの最大数。

**--モード <スナップショット | 停止 | サスペンド> (デフォルト = スナップショット)**

バックアップモード。

## --ノード <文字列>

このノードで実行された場合のみ実行される。

## --notes-template <文字列>

バックアップのメモを生成するためのテンプレート文字列。値で置き換えられる変数を含むことができる。現在サポートされているのは {{cluster}}, {{guestname}}, {{node}}, {{vmid}} であるが、今後追加されるかもしれない。改行とバックスラッシュは、それぞれ ``n と ``n としてエスケープする必要がある。

---

### 注

必要なオプション: ストレージ

---

## --notification-mode <auto | legacy-sendmail | notification-system>. (デフォルト = 自動)

使用する通知システムを決定する。*legacy-sendmail* に設定すると、vzdump は mailto/mailnotification パラメータを考慮し、*sendmail* コマンドで指定したアドレスにメールを送信します。*notification-system* に設定すると、PVE の通知システム経由で通知が送信され、mailto と mailnotification は無視されます。*auto* (デフォルト設定) に設定すると、mailto が設定されていればメールが送信され、設定されていなければ通知システムが使用されます。

## --notification-policy <always | failure | never> (デフォルト = always)

非推奨: を使用しないでください。

## --通知先 <文字列>

非推奨: を使用しないでください。

## --performance [max-workers=<integer>] [,pbs-entries-max=<integer>]。

その他のパフォーマンスに関する設定。

## --pigz <整数> (デフォルト = 0)

N>0 の場合は gzip の代わりに pigz を使う。N=1 はコアの半分を使い、N>1 は N をスレッド数として使う。

## --プール <文字列>

---

指定したプールに含まれるすべての既知のゲストシステムをバックアップする。

## --プロジェクト <プール>

trueの場合、バックアップを保護マークします。

---

### 注

必要なオプション: ストレージ

---

```
--prune-backups [keep-all=<1|0>] [,keep-daily=<N>]  
[,keep-hourly=<N>] [,keep-last=<N>] [,keep-  
monthly=<N>] [,keep-weekly=<N>] [,keep-yearly=<N>]  
(default = keep-all=1)
```

これらの保持オプションは、ストレージ構成のオプションの代わりに使用する。

**--quiet <ブール値>** (デフォルト = 0)

静かにしなさい。

**--remove <boolean>** (デフォルト = 1)

*prune-backups* に従って古いバックアップを削除する。

**--スクリプト <文字列>**

指定されたフックスクリプトを使用する。

**--stdexcludes <boolean>** (デフォルト = 1)

一時ファイルとログを除く。

**--stdout <ブール値>**

*tar* をファイルではなく標準出力に書き出す。

**--stop <boolean>** (デフォルト = 0)

このホストでのバックアップジョブの実行を停止します。

**--stopwait <整数>** (0 - N) (デフォルト = 10)

ゲストシステムが停止するまでの最大待機時間 (分)。

**--ストレージID**

出来上がったファイルをこのストレージに保存する。

**--tmpdir <文字列>**

指定したディレクトリに一時ファイルを保存する。

**--zstd <整数>** (デフォルト = 1)

Zstdのスレッド数。N=0は利用可能なコアの半分を使用し、Nが0より大きな値に設定された場合、Nはスレッド数として使用される。

## A.16 ha-manager - Proxmox VE HA マネージャ

**ha-manager <COMMAND> [ARGS] [OPTIONS] .**

**ha-manager add <sid> [OPTIONS]** を追加する。

新しいHAリソースを作成する。

**<sid>: <type>: <name>**。

HAリソースID。これは、リソースの種類と、コロンで区切られたリソース固有の名前で構成される（例：vm:100 / ct:100）。仮想マシンとコンテナの場合は、VMまたはCT IDをショートカットとして使用できます（例：100）。

**--コメント <文字列**

説明

**--グループ <文字列>**

HAグループの識別子。

**--max\_relocate <integer> (0 - N) (デフォルト = 1)**

サービスの起動に失敗した場合のサービス再配置の最大試行回数。

**--max\_restart <整数> (0 - N) (デフォルト = 1)**

起動に失敗したノードでサービスを再起動する最大回数。

**--state <無効 | 有効 | 無視 | 開始 | 停止> (デフォルト =)**

**を開始した)**

要求されたリソースの状態。

**--タイプ <ct | vm>**

リソースの種類

**ha-manager config [OPTIONS]**

HAのリソースをリストアップする。

**--タイプ <ct | vm>**

特定のタイプのリソースのみをリストアップ

**ha-manager crm-command migrate <sid> <node>**

他のノードへのリソース移行（オンライン）を要求する。

**<sid>: <type>: <name>。**

HAリソースID。これは、リソースの種類と、コロンで区切られたリソース固有の名前で構成される（例：vm:100 / ct:100）。仮想マシンとコンテナの場合は、VMまたはCT IDをショートカットとして使用できます（例：100）。

**<ノード>: <文字列>**

ターゲット・ノード

**ha-manager crm-command ノードメンテナンス無効化 <ノード>**

ノード保守要求の状態を変更する。

**<ノード>: <文字列>**

クラスタ・ノード名。

**ha-manager crm-command ノードメンテナンスイネーブル <ノード>**

ノード保守要求の状態を変更する。

**<ノード>: <文字列>**

クラスタ・ノード名。

**ha-manager crm-コマンド relocate <sid> <node>**

別のノードへのリソース再配置を要求する。これにより、古いノードでサービスが停止し、ターゲット・ノードでサービスが再起動します。

**<sid>: <type>: <name>。**

HAリソースID。これは、リソースの種類と、コロンで区切られたリソース固有の名前で構成される（例：vm:100 / ct:100）。仮想マシンとコンテナの場合は、VMまたはCT IDをショートカットとして使用できます（例：100）。

**<ノード>: <文字列>**

ターゲット・ノード

**ha-manager crm-コマンドストップ <sid> <timeout>**

サービスの停止を要求する。

**<sid>: <type>: <name>。**

HAリソースID。これは、リソースの種類と、コロンで区切られたリソース固有の名前で構成される（例：vm:100 / ct:100）。仮想マシンとコンテナの場合は、VMまたはCT IDをショートカットとして使用できます（例：100）。

**<タイムアウト>: <整数> (0 - N)**

秒単位のタイムアウト。0に設定するとハードストップが実行される。

**ha-manager groupadd <group> --nodes <string> [OPTIONS] .**

新しいHAグループを作成する。

**<group>: <文字列>**

HAグループの識別子。

**--コメント <文字列>**

説明

**--nodes <node>[:<pri>]{,<node>[:<pri>]}\***.

クラスタ・ノード名のリスト。優先順位は任意。

**--nofailback <boolean> (デフォルト = 0)**

CRM は最も優先度の高いノードでサービスを実行しようとします。より優先度の高いノードがオンラインになると、CRM はサービスをそのノードに移行します。nofailback を有効にすると、この動作が防止されます。

**--restricted <boolean> (デフォルト = 0)**

制限付きグループにバインドされたリソースは、グループによって定義されたノード上でのみ実行できる。

**--タイプ <グループ>**

グループタイプ。

**ha-managerグループ設定**

HAグループを獲得する。

**ha-manager groupremove <グループ削除>**

haグループの設定を削除します。

**<group>: <文字列>**

HAグループの識別子。

**ha-manager groupset <group> [OPTIONS] .**

haグループの設定を更新する。

**<group>: <文字列>**

HAグループの識別子。

**--コメント <文字列>**

説明

**--削除 <文字列>**

削除したい設定のリスト。

**-ダイジェスト <文字列>**

現在のコンフィギュレーション・ファイルのダイジェストが異なる場合、変更を防ぐ。これは、現在のコンフィギュレーション・ファイルの変更を防ぐために使用できます。

**--nodes <node>[:<pri>]{,<node>[:<pri>]}\*.**

クラスタ・ノード名のリスト。優先順位は任意。

**--nofailback <boolean> (デフォルト = 0)**

CRM は最も優先度の高いノードでサービスを実行しようとします。より優先度の高いノードがオンラインになると、CRM はサービスをそのノードに移行します。nofailback を有効にすると、この動作が

防止されます。

**--restricted <boolean> (デフォルト = 0)**

制限付きグループにバインドされたリソースは、グループによって定義されたノード上でのみ実行できる。

**ha-manager help [OPTIONS]**

指定したコマンドに関するヘルプを表示する。

**--extra-args <array>**

特定のコマンドのヘルプを表示する

**--verbose <ブール値**  
冗長な出力形式。

### **ha-manager migrate**

*ha-manager crm-command* の *migrate* のエイリアス。

### HAマネージャー移転

*ha-manager crm-command relocate* のエイリアス。

### **ha-manager remove <sid>**

リソース構成を削除します。

**<sid>: <type>: <name>**

HAリソースID。これは、リソースの種類と、コロンで区切られたリソース固有の名前で構成される（  
例：vm:100 / ct:100）。仮想マシンとコンテナの場合は、VMまたはCT IDをショートカットとして使  
用できます（例：100）。

### **ha-manager set <sid> [OPTIONS] [オプション].**

リソース構成を更新する。

**<sid>: <type>: <name>**

HAリソースID。これは、リソースの種類と、コロンで区切られたリソース固有の名前で構成される（  
例：vm:100 / ct:100）。仮想マシンとコンテナの場合は、VMまたはCT IDをショートカットとして使  
用できます（例：100）。

**--コメント <文字列**  
説明

**--削除 <文字列**

削除したい設定のリスト。

**-ダイジェスト <文字列**

現在のコンフィギュレーション・ファイルのダイジェストが異なる場合、変更を防ぐ。これは、現在  
のコンフィギュレーション・ファイルが変更されないようにするために使用できます。

**--グループ <文字列**

HAグループの識別子。

**--max\_relocate <integer> (0 - N) (デフォルト = 1)**

サービスの起動に失敗した場合のサービス再配置の最大試行回数。

**--max\_restart <整数> (0 - N) (デフォルト = 1)**

起動に失敗したノードでサービスを再起動する最大回数。

**--state <無効 | 有効 | 無視 | 開始 | 停止> (デフォルト =)**

**を開始した)**

要求されたリソースの状態。

**ha-manager status [OPTIONS]**

HA 管理者のステータスを表示する。

**--verbose <boolean> (デフォルト = 0)**

詳細出力。完全なCRMとLRMステータス (JSON) を含む。

## 付録B サービス・

### デーモン

#### B.1 pve-firewall - Proxmox VE ファイアウォールデーモン

**pve-firewall** <COMMAND> [ARGS] [OPTIONS].

**pve-firewall** コンパイル

ファイアウォールのルールをコンパイルして印刷する。これはテストに便利です。

**pve-firewall help** [OPTIONS] [オプション]。

指定したコマンドに関するヘルプを表示する。

**--extra-args <array>**

特定のコマンドのヘルプを表示する

**--verbose <ブール値>**

冗長な出力形式。

**pve-ファイアウォール**

ローカルネットワークの情報を印刷する。

**pve-ファイアウォール再起動**

Proxmox VE ファイアウォールサービスを再起動します。

**pve-firewall simulate** [OPTIONS] をシミュレートする。

ファイアウォールルールをシミュレートする。これはカーネルのルーティングテーブルをシミュレートするのではなく、単にソースゾーンからデスティネーションゾーンへのルーティングが可能であると仮定する。

**--dest <文字列**

宛先IPアドレス。

**--ポート <整数**

宛先ポート。

--from (host|outside|vm\d+|ct\d+|([a-zA-Z][a-zA-Z0-9]{0,9})/(\s+))

(デフォルト = 外側)

ソースゾーン

--プロトコル (tcp|udp) (デフォルト = tcp)

プロトコル

--ソース <文字列>

ソースIPアドレス。

--スポーツ <整数>

ソースポート

--to (host|outside|vm\d+|ct\d+|([a-zA-Z][a-zA-Z0-9]{0,9})/(\s+))

(デフォルト = ホスト)

デスティネーションゾーン

--verbose <boolean> (デフォルト = 0)

冗長出力。

**pve-firewall start** [OPTIONS] [オプション]。

Proxmox VEファイアウォールサービスを開始します。

--debug <ブール値> (デフォルト = 0)

デバッグモード - フォアグラウンドにとどまる

**pve-firewall status**

ファイアウォール

の状態を取得しま

す。

Proxmox VEファイアウォールサービスを停止します。停止すると、Proxmox VE関連のiptableルールがすべてアクティブに削除され、ホストが保護されなくなる可能性があることに注意してください。

## B.2 pvedaemon - Proxmox VE API デーモン

**pvedaemon** <COMMAND> [ARGS] [OPTIONS].

**pvedaemonヘルプ** [OPTIONS]。

指定したコマンドに関するヘルプを表示する。

**--extra-args <array>**

特定のコマンドのヘルプを表示する

**--verbose <ブール値**

冗長な出力形式。

**pvedaemon再起動**

デーモンを再起動する（起動していない場合は起動する）。

**pvedaemon start** [OPTIONS] [オプション]。

デーモンを起動する。

**--debug <ブール値>** (デフォルト = 0)

デバッグモード - フォアグラウンドにとどまる

**pvedaemon status**

デーモンの状態を取

得します。

**pvedaemon stop** デ

ーモンを停止します

。

### B.3 pveproxy - Proxmox VE API プロキシデーモン

**pveproxy** <COMMAND> [ARGS] [OPTIONS].

**pveproxy help** [OPTIONS] [オプション]。

指定したコマンドに関するヘルプを表示する。

**--extra-args <array>**

特定のコマンドのヘルプを表示する

**--verbose <ブール値>**

冗長な出力形式。

**pveproxyの再起動**

デーモンを再起動する（起動していない場合は起動する）。

**pveproxy start** [OPTIONS] [オプション]。

デーモンを起動する。

--debug <ブール値> (デフォルト = 0)

デバッグモード - フォアグラウンドにとどまる

**pveproxy status**

デーモンの状態を

取得します。

**pveproxy stop** デ

ーモンを停止しま

す。

## B.4 pvestatd - Proxmox VE ステータスデーモン

**pvestatd** <COMMAND> [ARGS] [OPTIONS].

**pvestatdヘルプ** [オプション]

指定したコマンドに関するヘルプを表示する。

**--extra-args <array>**

特定のコマンドのヘルプを表示する

**--verbose <ブール値**

冗長な出力形式。

**pvestatd再起動**

デーモンを再起動する（起動していない場合は起動する）。

**pvestatd start** [OPTIONS] [OPTIONS]。

デーモンを起動する。

**--debug <ブール値>** (デフォルト = 0)

デバッグモード - フォアグラウンドにとどまる

**pvestatd stop** デー

モンを停止します

。

## B.5 spiceproxy - SPICE プロキシサービス

**spiceproxy** <COMMAND> [ARGS] [OPTIONS].

**spiceproxy help** [OPTIONS] [オプション].

指定したコマンドに関するヘルプを表示する。

**--extra-args <array>**

特定のコマンドのヘルプを表示する

**--verbose <ブール値**

冗長な出力形式。

**spiceproxy 再起動**

デーモンを再起動する（起動していない場合は起動する）。

**spiceproxy start** [OPTIONS] [オプション]。

デーモンを起動する。

**--debug <ブール値> (デフォルト = 0)**

デバッグモード - フォアグラウンドにとどまる

**spiceproxy status**

デーモンの状態を

取得します。

**spiceproxy stop デ**

ーモンを停止しま

す。

## B.6 pmxcfs - Proxmox クラスタファイルシステム

**pmxcfs [OPTIONS]**

ヘルプオプション:

**-h, --help**

ヘルプオプションを表

示する アプリケーションオ

プション:

**-d, --debug**

デバッグメッセージをオンにする

**-f, --foreground**

サーバーをデーモン化しない

**-l, --local**

ローカルモードを強制する (corosync.confを無視し、クオーラムを強制する)

このサービスは通常systemdツールセットを使って起動・管理される。このサービスは *pve-cluster* と呼ばれます。

```
systemctl start pve-cluster
```

```
systemctl stop pve-cluster
```

```
systemctl status pve-cluster
```

## B.7 pve-ha-crm - クラスタリソーススマネージャーデーモン

**pve-ha-crm** <COMMAND> [ARGS] [OPTIONS].

**pve-ha-crmヘルプ** [オプション] (**pve-ha-crm help**)

指定したコマンドに関するヘルプを表示する。

**--extra-args <array>**

特定のコマンドのヘルプを表示する

**--verbose <ブール値**

冗長な出力形式。

**pve-ha-crm start** [OPTIONS] [オプション]。

デーモンを起動する。

**--debug <ブール値> (デフォルト = 0)**

デバッグモード - フォアグラウンドにとどまる

**pve-ha-crm status**

デーモンの状態を

取得します。 **pve-**

**ha-crm stop** デー

モンを停止します

。

## B.8 pve-ha-lrm - ローカルリソースマネージャーデーモン

**pve-ha-lrm** <COMMAND> [ARGS] [OPTIONS].

**pve-ha-lrmヘルプ** [オプション] (**pve-ha-lrm help** [OPTIONS])

指定したコマンドに関するヘルプを表示する。

**--extra-args <array>**

特定のコマンドのヘルプを表示する

**--verbose <ブール値**

冗長な出力形式。

**pve-ha-lrm start** [OPTIONS] [オプション]。

デーモンを起動する。

**--debug <ブール値> (デフォルト = 0)**

デバッグモード - フォアグラウンドにとどまる

**pve-ha-lrm status**

デーモンの状態を

取得します。 **pve-**

**ha-lrm stop** デーモ

ンを停止します。

## B.9 pvescheduler - Proxmox VE スケジューラーデーモン

**pvescheduler** <COMMAND> [ARGS] [OPTIONS].

**pvescheduler help** [OPTIONS] [オプション]。

指定したコマンドに関するヘルプを表示する。

**--extra-args <array>**

特定のコマンドのヘルプを表示する

**--verbose <ブール値>**

冗長な出力形式。

### pveschedulerの再起動

デーモンを再起動する（起動していない場合は起動する）。

**pvescheduler start** [OPTIONS] (プロセスケジューラ・スタート) [OPTIONS] (オプション)

デーモンを起動する。

**--debug <ブール値> (デフォルト = 0)**

デバッグモード - フォアグラウンドにとどまる

### pvescheduler status

デーモンの状態を取

得します。

### pvescheduler stop

デーモンを停止しま

す。

# 付録C 設定ファイル

## C.1 データセンターの構成

etc/pve/datacenter.cfgファイルはProxmox VEの設定ファイルです。このファイルには、すべてのノードで使用されるクラスタ全体のデフォルト値が含まれています。

### C.1.1 ファイル形式

このファイルでは、コロンで区切られたシンプルなキー／バリュー・フォーマットを使用する。各行のフォーマットは以下の通り：

オプション：値

ファイル内の空白行は無視され、#文字で始まる行はコメントとして扱われ、これも無視される。

### C.1.2 オプション

**bwlimit:**

[clone=<LIMIT>] [,default=<LIMIT>] [,migration=<LIMIT>] [,move=<LIMIT>] [,restore=<LIMIT>]。

さまざまな操作のI/O帯域幅制限を設定する（単位：KiB/s）。

**クローン=<リミット**

クローン作成ディスクの帯域制限（KiB/s単位）

**デフォルト=<LIMIT**

デフォルトの帯域幅制限（単位：KiB/s）

## マイグレーション=<リミット

ゲストの移行（ローカルディスクの移動を含む）の帯域幅制限（KiB/s単位）

### **move=<リミット**

移動ディスクの帯域幅制限（KiB/s単位）

**リストア=<リミット>**

バックアップからゲストをリストアする際の帯域幅の制限（KiB/s単位）

**コンソール:<アプレット | html5 | vv | xtermjs>.**

デフォルトの Console ビューアを選択します。ビルトインの Java アプレット (VNC。非推奨で html5 にマップされます)、外部の virt-viewer 互換アプリケーション (SPICE)、HTML5 ベースの vnc ビューア (noVNC)、または HTML5 ベースのコンソールクライアント (xtermjs) のいずれかを使用できます。選択したビューアが利用できない場合 (例えば、SPICE が VM に対して有効化されていない場合)、フォールバックは noVNC になります。

**crs: [ha=<基本 | 静的>] [,ha-rebalance-on-start=<1 | 0>]。**

クラスタリソースのスケジューリング設定。

**ha=<basic | static> (デフォルト=basic)**

HAマネージャがサービスを開始または回復するノードをどのように選択するかを設定する。

*basic* では、サービスの数のみが使用され、*static* では、サービスのCPUとメモリ構成が考慮される。

**ha-rebalance-on-start=<boolean> (デフォルト=0)**

HAサービスの要求状態が停止から開始へ変更された場合に、CRSを使用して対象ノードを選択するように設定。

**説明:<文字列>**

データセンターの説明。ウェブ・インターフェースのデータセンター・ノート・パネルに表示される。これは設定ファイル内にコメントとして保存されます。

**email\_from:<文字列>**

通知を送信するメールアドレスを指定する（デフォルトはroot@\$hostname）

**フェンシング: <両方 | ハードウェア | ウオッヂドッグ> (デフォルト=ウォッヂドッグ)**

HAクラスタのフェンシングモードを設定します。ハードウェアモードでは、/etc/pve/ha/fence.cfgにフェンスデバイスの有効な設定が必要です。両方の場合、2つのモードがすべて使用されます。

**警告**

どちらも実験的なものであり、WIP である。

---

**ha: shutdown\_policy=<enum>**

クラスタ全体のHA設定。

**shutdown\_policy=<conditional | failover | freeze | migrate>。**

(デフォルト = 条件付き)

ノードの電源オフや再起動時にHAサービスを処理するポリシーを記述します。Freezeは、シャットダウン時にノードに残っているサービスを常に凍結し、これらのサービスはHAマネージャによって回復されません。フェイルオーバーはサービスをフリーズとしてマークせず、そのため

条件はシャットダウンの種類に応じて自動的に選択されます。つまり、リブート時にはサービスはフリーズしますが、パワーオフ時にはサービスはそのまま維持されるため、約2分後に復旧します。リブートまたはシャットダウンがトリガーされると、Migrateは実行中のサービスをすべて別のノードに移動しようとします。ノード上に実行中のサービスがなくなると、パワーオフプロセスが続行されます。ノードが再び起動した場合、少なくとも他のマイグレーション、リロードアクション、リカバリーが行われていなければ、サービスはパワーオフされたノードに戻されます。

**http\_proxy: http://.\***

ダウンロードに使用する外部httpプロキシを指定する（例：`http://username:password@host:port/`）

**キーボード:<da | de | de-ch | en-gb | en-us | es | fi | fr | fr-be | fr-ca | fr-ch | hu | is | it | ja | lt | mk | nl | no | pl | pt | pt-br | sl | sv | tr>.**

vncサーバのデフォルトキーボードレイアウト。

**language:<ar | ca | da | de | en | es | eu | fa | fr | he | hr | it | ja | ka | kr | nb | nl | nn | pl | pt\_BR | ru | sl | sv | tr | ukr | zh\_CN | zh\_TW>。**

デフォルトのGUI言語。

**mac\_prefix: <文字列> (デフォルト = BC:24:11)**

仮想ゲストの自動生成 MAC アドレスのプレフィックス。デフォルトのBC:24:11は、MAC Address Block Large (MA-L) のためにIEEEからProxmox Server Solutions GmbHに割り当てられたOrganizationally Unique Identifier (OUI) です。これはローカルネットワーク、つまり一般ユーザーから直接アクセスできないネットワーク (LANやNAT/マスカレードなど) で使用できます。

仮想ゲストのネットワークを(部分的に)共有する複数のクラスタを実行する場合は、MACの衝突の可能性を減らすために、デフォルトのMACプレフィックスを拡張するか、カスタムの(有効な)プレフィックスを生成することを強くお勧めします。たとえば、各クラスタのProxmox OUIに、1番目はBC:24:11:0、2番目はBC:24:11:1というように、個別の16進数を追加します。また、VLANを使用するなどして、ゲストのネットワークを論理的に分離することもできます。

+ 一般にアクセス可能なゲストの場合は、IEEEに登録されているOUIを取得するか、ホスティング・プロバ

イダーのネットワーク管理者と調整することをお勧めします。

**max\_workers: <整数> (1 - N)**

ha-managerからのstopall VMやタスクのようなアクションで、（ノードあたり）最大何人のワーカーが起動するかを定義します。

マイグレーションで使用されます: [type=]<secure|insecure>[,network=<CIDR>]。

クラスタ全体のマイグレーション設定

**ネットワーク=<CIDR>**

移行に使用する（サブ）ネットワークのCIDR。

**type=<insecure | secure> (デフォルト = secure)**

マイグレーショントラフィックはデフォルトでSSHトンネルを使用して暗号化されます。安全で完全にプライベートなネットワークでは、パフォーマンスを上げるためにこれを無効にすることができます。

**migration\_unsecure: <論理値>**

移行はデフォルトでSSHトンネルを使用して安全に行われます。セキュアなプライベートネットワークでは、移行を高速化するためにこれを無効にすることができます。非推奨です。代わりに*migration*プロパティを使用してください！

**next-id: [lower=<整数>] [,upper=<整数>]。**

フリーVMID自動選択プールの範囲を制御する。

**lower=<整数> (デフォルト = 100)**

自由な next-id API 範囲の下限、包括的な境界。

**upper=<整数> (デフォルト = 1000000)**

自由な next-id API 範囲の上限、排他的境界。

**を通知する: [fencing=<always|never>]  
[,package-updates=<auto|always|never>]  
]。**

[レプリケーション=<always|never> [,target-fencing=<TARGET>]  
[,target-package-updates=<TARGET>] [,target-replication=<TARGET>]  
]。

クラスタ全体の通知設定。

## フェンシング

UNUSED - 代わりにデータセンター通知設定を使用する。

**package-updates=<always | auto | never> (デフォルト = auto)**

廃止: 代わりにデータセンター通知設定を使用してください。日次更新ジョブが通知を送信する頻度を制御します:

- 有効なサブスクリプションを持つシステムは、プロダクションレディであると仮定されるため

- 、保留中のアップデートについて知っているはずだからである。
- 新しい保留中のアップデートがある場合は、常にアップデートのたびに。
- 保留中の新しいアップデートの通知を送信することはありません。

### レプリケーション=<常に | 決して

UNUSED - 代わりにデータセンター通知設定を使用する。

### ターゲット・フェンシング=<ターゲット>

UNUSED - 代わりにデータセンター通知設定を使用する。

### target-package-updates=<TARGET>。

UNUSED - 代わりにデータセンター通知設定を使用する。

### target-replication=<TARGET>。

UNUSED - 代わりにデータセンター通知設定を使用する。

**登録タグ: <タグ>[ ;<タグ>... ]。**

タグの設定と削除に Sys.Modify on /が必要なタグのリスト。ここで設定されたタグは user-tag-accessもSys.Modify.Sysが必要です。

**タグのスタイルを指定します: [大文字・小文字を区別する=<1|0>]。**

[ ,color-map=<tag>:<hex-color>[:<hex-color-for-text>] [;<tag>=...]] [ ,ordering=<設定|アルファベット順>] [ ,shape=<enum>] .

タグスタイルのオプション。

**case-sensitive=<boolean> (デフォルト = 0)**

更新時に一意なタグをフィルタリングする際に、大文字と小文字を区別してチェックするかどうかを制御します。

**color-map=<tag>:<hex-color>[:<hex-color-for-text>] [;<tag>=...].**

タグの手動カラーマッピング（セミコロン区切り）。

**ordering=<alphabetical | config> (デフォルト=アルファベット順)**

ウェブインターフェイスとAPIアップデートにおけるタグのソートを制御します。

**shape=<circle | dense | full | none> (デフォルト=circle)**

fullは完全なタグを描画します。circleは背景色の円だけを描画します。denseは小さな長方形だけを描画します(多くのタグがそれぞれのゲストに割り当てられている場合に便利です).noneはタグの表示を無効にします。

**u2f: [appid=<APPID>] [ ,origin=<URL>]。**

u2f

**appid=<APPID>**

U2F AppId URLのオーバーライド。デフォルトはオリジン。

**オリジン=<URL>**

U2F Originのオーバーライド。ほとんどの場合、単一のURLを持つ単一のノードに便利です。

**user-tag-access: [user-  
allow=<enum>] [ ,user-allow-list=<タ  
グ>[;<タグ>...]]。**

**user-allow=<existing | free | list | none> (デフォルト = free)**

ユーザが制御するリソース（ゲストなど）に対して、どのタグを設定または削除できるかを制御します。`SYS.Modify`権限を持つユーザは、常に無制限です。

- タグは使用できません。
- *user-allow-list*のリストタグは使用可能です。
- のような既存のリストも使えるが、すでにあるリソースのタグも使える。
- タグ制限なし。

**user-allow-list=<タグ>[ ;<タグ>... ]。**

ユーザーが設定・削除できるタグのリスト（セミコロン区切り）。

そして既存の。

**webauthn: [allow-subdomains=<1|0>] [,id=<DOMAINNAME>]  
[,origin=<URL>] [,rp=<RELYING\_PARTY>].**  
webauthnの設定

**allow-subdomains=<boolean> (デフォルト = 1)**

正確なURLではなく、サブドメインをオリジンとすることを許可するかどうか。

**id=<DOMAINNAME>**

依拠当事者ID。プロトコル、ポート、場所を除いたドメイン名でなければならない。これを変更すると既存のクレデンシャルを破る。

**オリジン=<URL**

サイトのオリジン。`https:// URL`（または`http://localhost`）でなければならない。

ユーザーがウェブ・インターフェイスにアクセスするためにブラウザに入力するアドレスが含まれていなければならない。これを変更すると、既存の認証情報が壊れる**可能性があります**。

**rp=<RELYING\_PARTY> です。**

依拠当事者名。任意のテキスト識別子。これを変更すると、既存のクレデンシャルが破壊される**可能性がある**。

# 付録D カレンダー

## ・イベント

### D.1 スケジュール形式

Proxmox VEは非常に柔軟なスケジュール設定を持っています。これは systemd time calendar event format に基づいています。<sup>1</sup> カレンダーイベントは、1つの式で1つ以上の時点を参照するために使うことができます。

このようなカレンダーイベントは、次のような書式を使用する：

平日] [[年-]月-日] [時:分[:秒]] [年-]月-日

このフォーマットでは、ジョブを実行する日を設定できます。また、1つ以上の開始時間を設定することもできます。これはレプリケーション・スケジューラーに、ジョブが開始されるべき時刻を指示します。この情報を使って、毎日午後10時に実行するジョブを作成することができます：'月,火,水,木,金,22'と省略できます：ほとんどの合理的なスケジュールはこのように直感的に書くことができる。

---

#### 注

時間は24時間表示。

---

便利で短い設定を可能にするために、ゲストごとに1つ以上の繰り返し時間を設定することができます。これらは、開始時刻そのものと、開始時刻に繰り返し値の倍数を加えた時刻にレプリケーションが行われることを示します。午前8時にレプリケーションを開始し、午前9時まで15分ごとにレプリケーションを繰り返す場合は、次のようにになります：'8:00/15'

---

ここでは、時区切り(:)が使用されていない場合、値は分として解釈されることがわかります。このような区切りが使用された場合、左側の値は「時」を表し、右側の値は「分」を表します。さらに、\* を使用すると、すべての可能な値にマッチさせることができます。

さらにアイデアを得るために、[以下の例をもっと見る](#)。

## D.2 詳細仕様

---

<sup>1</sup> 詳細は man 7 systemd.time を参照。

## 平日

曜日は、sun、mon、tue、wed、thu、fri、satの略語で指定する。複数の日をカンマ区切りで指定することもできる。また、開始日と終了日を". ."で区切って、例えばmon..fri.のように範囲指定することもできる。これらの書式は混在させることができる。省略した場合は、'\*'とみなされます。

## タイムフォーマット

時刻の書式は、時・分のインターバル・リストで構成される。時間と分は': 'で区切られる。時」も「分」も、「日」と同じ書式で、リストや値の範囲を指定することができます。最初は時間です、次に分。時間は必要なければ省略できる。この場合、時間の値は'\*'とみなされる。値の有効範囲は、時が0～23、分が0～59である。

### D.2.1 例を挙げよう：

特定の意味を持つ特別な値がいくつかある：

表D.1： 特別な値

値	構文
ここまかに	*** ** : --:00
毎時	*** * --:00:00
デイリー	*** --00:00:00
ウィークリー	MON - -*** 00:00:00
毎月	** -01 --00:00:00
毎年または毎年	*-01-01 00:00:00
クオータリー	*-01,04,07,10-01 00:00:00
半年ごとまたは半年ごと	*-01,07-01 00:00:00

表D.2： スケジュールの例

スケジュール文字列	オルタナティブ	意味
月、火、水、木、金	月・金	毎営業日0:00
土、日	土・日	週末のみ0:00
月、水、金	-	月曜日、水曜日のみ と金曜日の0:00
12:05	12:05	毎日午後12時5分
*/5	0/5	5分ごと

月...30/10水	月、火、水 30/10	月曜日、火曜日、水曜日 30分後 、40分後、50分後 毎時
月...金8...17,22:0/15	-	毎営業日、午前8時から午後6時 までと午後10時から11時までは 15分ごと PM

表D.2: (続き)

スケジュール文字列	オルタナティブ	意味
12日 (金) 13:5/20	12,13日(金):5/20	金曜日の12:05、12:25、12:45、 13:05、13:25、13:45
12,14,16,18,20,22:5	12/2:5	毎日12:05から 22:05、2時間ごと
*	*/1	毎分（最小間隔）
*-05	-	毎月5日
土*-1.7 15:00	-	毎月第一土曜日 15:00
2015-10-21	-	2015年10月21日 00:00

## 付録E QEMU

### vCPUリスト

#### E.1 はじめに

これは、QEMUで定義されているAMDとIntelのx86-64/amd64 CPUタイプのリストで、2007年までさかのぼります。

#### E.2 インテルCPUの種類

##### インテル・プロセッサー

- ナハレム: 第1世代インテル・コア・プロセッサー
- *Nahalem-IBRS (v2)* : Spectre v1 保護機能を追加 (+spec-ctrl )
- *Westmere* : 第1世代インテル・コア・プロセッサー (Xeon E7-)
- *Westmere-IBRS (v2)* : Spectre v1 保護機能を追加 (+spec-ctrl )
- *SandyBridge* : 第2世代インテル・コア・プロセッサー
- *SandyBridge-IBRS (v2)* : Spectre v1 プロテクションの追加 (+spec-ctrl )
- *IvyBridge* : 第3世代インテル・コア・プロセッサー
- *IvyBridge-IBRS (v2)*: Spectre v1 保護を追加 (+spec-ctrl )

- *Haswell* : 第4世代インテル・コア・プロセッサー
- *Haswell-noTSX (v2)* : TSX (-hle, -rtm) を無効にする。
- *Haswell-IBRS (v3)* : TSX の再追加、Spectre v1 保護の追加 (+hle, +rtm, +spec-ctrl )
- *Haswell-noTSX-IBRS (v4)* : TSX (-hle, -rtm) を無効にする。
- *Broadwell* : 第5世代インテル・コア・プロセッサー
- *Skylake* : 第1世代Xeonスケーラブル・サーバー・プロセッサー
- *Skylake-IBRS (v2)* : Spectre v1 保護の追加、CLFLUSHOPT の無効化 (+spec-ctrl, -clflushopt )

- *Skylake-noTSX-IBRS (v3)* : TSX を無効にする (-hle, -rtm)
- *Skylake-v4* : EPTスイッチングを追加 (+vmx-eptp-switching)
- カスカデレイク: 第2世代Xeonスケーラブル・プロセッサ
- *Cascadelake-v2* : add arch\_capabilities msr (+arch-capabilities, +rdctl-no, +ibrs-all, +skip-l1dfl-vmentry、+mds-no)。
- *Cascadelake-v3* : TSX (-hle, -rtm) を無効にする。
- *Cascadelake-v4* : EPTスイッチングを追加 (+vmx-eptp-switching)
- *Cascadelake-v5* : XSAVESの追加 (+xsaves, +vmx-xsaves)
- *Cooperlake*: 4ソケットおよび8ソケットサーバー用第3世代Xeonスケーラブル・プロセッサー
- *Cooperlake-v2* : XSAVESの追加 (+xsaves, +vmx-xsaves)
- アイスレイク第3世代Xeonスケーラブル・サーバ・プロセッサ
- *Icelake-v2* : TSX (-hle, -rtm) を無効にする。
- *Icelake-v3* : arch\_capabilities msr (+arch-capabilities, +rdctl-no, +ibrs-all, +skip-l1dfl-vmentry, +mds-no, +pschange-mc-no, +taa-no) を追加する。
- *Icelake-v4* : 不足していたフラグを追加 (+sha-ni, +avx512ifma, +rdpid, +fsrm, +vmx-rdseed-exit, +vmx-pml, +vmx-eptp-switching)
- *Icelake-v5* : XSAVESの追加 (+xsaves, +vmx-xsaves)
- *Icelake-v6* : 「5レベルEPT」を追加 (+vmx-page-walk-5)
- *SapphireRapids* : 第4世代Xeonスケーラブル・サーバ・プロセッサ

## E.3 AMD CPUの種類

### AMDプロセッサー

- *Opteron\_G3* : K10
- *Opteron\_G4* : ブルドーザー
- *Opteron\_G5* : パイルドライバー
- *EPYC* : Zenプロセッサーの第1世代

- *EPYC-IBPB (v2)* : Spectre v1プロテクションを追加 (+*ibpb*)
- *EPYC-v3* : 不足していたフラグを追加 (+*perfctr-core*, +*clzero*, +*xsaveerptr*, +*xsaves*)
- *EPYC-Rome*: 第2世代Zenプロセッサ
- *EPYC-Rome-v2* : Spectre v2, v4プロテクションの追加 (+*ibr*s, +*amd-ssbd* )
- *EPYC-Milan*: 第3世代Zenプロセッサ
- *EPYC-Milan-v2* : 不足していたフラグを追加 (+*vAES*, +*vpclmulqdq*, +*stibp-always-on*, +*amd-psfd*, +*no-nested-data-bp*, +*lfence-always-serializing*, +*null-sel-clr-base*)

## 付録F

### ファイアウォール・マクロの定義

アマンダ アマンダ・バックアップ

アクション	プロト	ポート	スポーツ
パラム	udp	10080	
パラム	ティーシーピー	10080	

認証 認証 (identd) トラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	113	

BGP ボーダー・ゲートウェイ・プロトコルのトラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	179	

BitTorrent BitTorrent 3.1以前のBitTorrentトラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	6881:6889	

パラム	udp	6881	
-----	-----	------	--

*BitTorrent32* BitTorrent 3.2 以降用の BitTorrent トラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	6881:6999	
パラム	udp	6881	

CVS

同時バージョン管理システム pserver トラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	2401	

Ceph

Cephストレージクラスタ トラフィック (Cephモニタ、OSD、MDSデーモン)

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	6789	
パラム	ティーシーピー	3300	
パラム	ティーシーピー	6800:7300	

シトリックス

Citrix/ICA トラフィック (ICA、ICAブラウザ、CGP)

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	1494	
パラム	udp	1604	
パラム	ティーシーピー	2598	

DAAP

Digital Audio Access Protocol トラフィック (iTunes、Rhythmboxデーモン)

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	3689	
パラム	udp	3689	

DCC

分散型チェックサム・クリアリングハウス・スパム・フィルタリング機構

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	6277	

*DHCPfwd*

転送されたDHCPトラフィック

アクション	プロト	ポート	スポーツ
パラム	udp	67:68	67:68

*DHCPv6* DHCPv6 トラフィック

アクション	プロト	ポート	スポーツ
パラム	udp	546:547	546:547

*DNS* ドメイン名システムのトラフィック (updおよびtcp)

アクション	プロト	ポート	スポーツ
パラム	udp	53	
パラム	ティーシーピー	53	

*ディストック* 分散コンパイラサービス

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	3632	

*FTP* ファイル転送プロトコル

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	21	

*フィンガー* フィンガープロトコル (RFC 742)

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	79	

*GNUnet* GNUnet 安全なピアツーピアネットワーキングトラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	2086	
パラム	udp	2086	
パラム	ティーシーピー	1080	

アクション	プロト	ポート	スポーツ
パラム	udp	1080	

**GRE** Generic Routing Encapsulation トンネリングプロトコル。

アクション	プロト	ポート	スポーツ
パラム	47		

**ギット** Git分散リビジョン管理トラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	9418	

**HKP** OpenPGP HTTP鍵サーバプロトコルトラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	11371	

**HTTP** ハイパーテキスト転送プロトコル (WWW)

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	80	

**HTTPS** SSL経由のハイパーテキスト転送プロトコル (WWW)

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	443	

**ICPV2** インターネット・キャッシュ・プロトコルV2 (Squid) のトラフィック

アクション	プロト	ポート	スポーツ
パラム	udp	3130	

*ICQ*

AOLインスタント・メッセンジャーのトラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	5190	

*IMAP*

インターネット・メッセージ・アクセス・プロトコル

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	143	

*IMAPS*

インターネット・メッセージ・アクセス・プロトコル・オーバーSSL

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	993	

*IP/IP*

IPIPカプセレーション・トラフィック

アクション	プロト	ポート	スポーツ
パラム	94		

*IPsec*

IPsec トラフィック

アクション	プロト	ポート	スポーツ
パラム	udp	500	500
パラム	50		

*IPsecah*

IPsec 認証 (AH) トラフィック

アクション	プロト	ポート	スポーツ
パラム	udp	500	500
パラム	51		

*IPsecnat*

IPsecトラフィックとNat-Traversal

アクション	プロト	ポート	スポーツ
パラム	udp	500	
パラム	udp	4500	
パラム	50		

*IRC*

インターネット・リレー・チャットのトラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	6667	

*ジェットダイレクト*

HP Jetdirectプリント

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	9100	

*L2TP*

レイヤー2トンネリングプロトコルのトラフィック

アクション	プロト	ポート	スポーツ
パラム	udp	1701	

*LDAP*

ライトウェイト・ディレクトリ・アクセス・プロトコル・トラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	389	

*LDAPS*

セキュア・ライトウェイト・ディレクトリ・アクセス・プロトコル・トラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	636	

*MDNS* マルチキャストDNS

アクション	プロト	ポート	スポーツ
パラム	udp	5353	

*MSNP* マイクロソフト通知プロトコル

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	1863	

*MSSQL* マイクロソフトSQLサーバー

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	1433	

メール メールトラフィック (SMTP、SMTPE、送信)

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	25	
パラム	ティーシーピー	465	
パラム	ティーシーピー	587	

ムニン ムニン・ネットワーク・リソース・モニタリング・トラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	4949	

*MySQL* MySQLサーバー

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	3306	

**NNTP** NNTPトラフィック（ユーズネット）。

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	119	

**NNTPS** 暗号化されたNNTPトラフィック（ユースネット）

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	563	

**NTP** ネットワークタイムプロトコル(ntpd)

アクション	プロト	ポート	スポーツ
パラム	udp	123	

**NeighborDiscoveryPv6** ネイバー勧誘、ネイバー広告、ルーター広告

アクション	プロト	ポート	スポーツ
パラム	アイシーエムピーブイシックス	ルーター募集	
パラム	アイシーエムピーブイシックス	ルーター広告	
パラム	アイシーエムピーブイシックス	隣人募集	
パラム	アイシーエムピーブイシックス	隣人 広告	

**OSPF** OSPFマルチキャストトラフィック

アクション	プロト	ポート	スポーツ
パラム	89		

[OpenVPN](#)[OpenVPN トラフィック](#)

アクション	プロト	ポート	スポーツ
パラム	udp	1194	

*PCA*

シマンテックPCAnywhere (tm)

アクション	プロト	ポート	スポーツ
パラム	udp	5632	
パラム	ティーシーピー	5631	

*PMG*

Proxmox Mail Gatewayウェブインターフェース

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	8006	

*POP3*

POP3トラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	110	

*POP3S*

暗号化されたPOP3トラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	995	

*PPtP*

ポイント・ツー・ポイント・トンネル・プロトコル

アクション	プロト	ポート	スポーツ
パラム	47		
パラム	ティーシーピー	1723	

*ピング*

ICMPエコーリクエスト

アクション	プロト	ポート	スポーツ
パラム	アイシーエムピー	エコー要求	

*PostgreSQL* PostgreSQLサーバー

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	5432	

*プリンタ* ラインプリンタープロトコル印刷

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	515	

*RDP* マイクロソフト・リモート・デスクトップ・プロトコル・トラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	3389	

*RIP* ルーティング情報プロトコル（双方向）

アクション	プロト	ポート	スポーツ
パラム	udp	520	

*RNDC* BINDリモート管理プロトコル

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	953	

*レイザー* アンチスパムシステム

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	2703	

*Rdate*

リモート時刻検索 (rdate)

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	37	

*Rsync* Rsyncサーバー

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	873	

*SANE* SANEネットワークスキャン

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	6566	

*SMB* マイクロソフトSMBトラフィック

アクション	プロト	ポート	スポーツ
パラム	udp	135,445	
パラム	udp	137:139	
パラム	udp	1024:65535	137
パラム	ティーシーピー	135,139,445	

*SMBswat* Sambaウェブ管理ツール

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	901	

*SMTP* 簡易メール転送プロトコル

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	25	

*SMTPS* 暗号化簡易メール転送プロトコル

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	465	

**SNMP** 簡易ネットワーク管理プロトコル

アクション	プロト	ポート	スポーツ
パラム	udp	161:162	
パラム	ティーシーピー	161	

**SPAMD** スパム・アサシン SPAMD トラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	783	

**SPICEproxy** Proxmox VE SPICEディスプレイのプロキシトラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	3128	

**SSH** 安全なシェルトラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	22	

**SVN** Subversion サーバー (svnserve)

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	3690	

**SixXS** SixXS IPv6導入とトンネルブローカー

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	3874	
パラム	udp	3740	
パラム	41		

アクション	プロト	ポート	スポーツ
パラム	udp	5072,8374	

イカ SquidのWebプロキシトラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	3128	

投稿 メール送信トラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	587	

シスログ Syslogプロトコル（RFC 5424）のトラフィック

アクション	プロト	ポート	スポーツ
パラム	udp	514	
パラム	ティーシーピー	514	

TFTP Trivial File Transfer Protocol トラフィック

アクション	プロト	ポート	スポーツ
パラム	udp	69	

テルネット Telnetトラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	23	

テルネット Telnet over SSL

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	992	

**時間** RFC 868 時間プロトコル

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	37	

*Trcrt* トレースルート（最大30ホップ）トラフィック

アクション	プロト	ポート	スポーツ
パラム	udp	33434:33524	
パラム	アイシーエムピー	エコー要求	

**VNC** VNCディスプレイのVNCトラフィック 0 - 99

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	5900:5999	

*VNCL* VncsサーバーからVncビューアーへのVNCトラフィック（リッスンモード

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	5500	

**ウェブ** WWWトラフィック（HTTPおよびHTTPS）

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	80	
パラム	ティーシーピー	443	

ウェブキャッシュウェブキャッシュ/プロキシトラフィック（ポート8080）

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	8080	

ウェブミン Webmin トラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	10000	

Whois Whois (nickname、RFC 3912) トラフィック

アクション	プロト	ポート	スポーツ
パラム	ティーシーピー	43	

# 付録G マークダウン入門

## 門

MarkdownはウェブライターのためのテキストからHTMLへの変換ツールです。Markdownを使えば、読みやすく、書きやすいプレーンテキスト形式を使って文章を書き、それを構造的に妥当な XHTML（またはHTML）に変換することができます。

- ジヨン・グルーバー <https://daringfireball.net/projects/markdown/>

Proxmox VEのウェブインターフェイスは、ノードとバーチャルゲストのノートにリッチテキストフォーマットをレンダリングするMarkdownの使用をサポートしています。

Proxmox VEは、テーブルやタスクリストのようなGFM（GitHub Flavoured Markdown）のほとんどの拡張機能でCommonMarkをサポートしています。

## G.1 マークダウンの基本

なお、ここでは基本的なことを説明するだけなので、例えば <https://www.markdownguide.org/> など、より広範なリソースをウェブで検索してほしい。

### G.1.1 見出し

```
# This is 見出し h1 ##  
This is 見出し h2  
##### これは見出します。
```

## G.1.2 強調

\* `text*` または `_text_` を使って強調する。

\*\* `テキスト**` または `_____テキスト_`を使います。組

み合わせも可能です：

\*\* `_あなたは**` `__それらを組み合わせることができます。`

### G.1.3 リンク

リンクの自動検出を使うこともできる。例えば、<https://forum.proxmox.com>、クリック可能なリンクに変換する。

例えば、リンクテキストをコントロールすることもできる：

さて、【カッコ内がリンクテキストになる】 (<https://forum.proxmox.com>←'  
/).

### G.1.4 リスト

#### 順序なしリスト

- \* 項目 1
- \* 項目 2
- \* 項目 2a
- \* 項目 2b

順序なしリストの場合は、\* または – を使用する：

インデントを追加することで、入れ子になったリストを作成することができる。

#### 注文リスト

1. 項目 1
1. 項目 2
1. 項目 3
  1. 項目 3a
  1. 項目 3b

---

#### 注

順番に並べられたリストの整数は正しくする必要はなく、自動的に番号が振られる。

---

#### タスクリスト

---

タスクリストでは、未完了のタスクには空白のボックス [ ] を、完了したタスクにはXの

- [X] 最初の仕事はすでに終わった!
- [X] 2本目も
- [これはまだやる
- [これもそうだ

## G.1.5 テーブル

テーブルは、パイプ記号 | で列を区切り、-で表のヘッダーと本文を区切る。

左カラムも	右カラム	一部   その他   コラム   センタリングワークス ←'	
----- ←'			
左 フー	右 フー	最初   列   ここ   >センター< ←'	
左 バー	右 バー	セカン ド	列   ここ   12345 ←'
左 バズ	右 バズ	サード	列   ここ   テスト に
左サフ	石サフ	4列目   ここ   ←'	
&#x2601; &#xfe0f; &#x2601; &#xfe0f; &#x2601; &#xfe0f;			
左ラビ	右ラビ	そして   ラスト   ここで   終 ←'	
わり			

列を空白できれいに揃える必要はないが、その方が表の編集が楽になることに注意。

## G.1.6 ブロックの名言

- > Markdownは、プレーンテキスト形式の軽量マークアップ言語です。  
シンタックスである、
- > 2004年、ジョン・グルーバーがアーロン・スワーツとともに創設。
- >
- >> Markdownは、`readme`ファイルのフォーマットや、←のメッセージの記述によく使われる。  
オンライン・ディスカッション・フォーラム
- >> プレーンテキストエディターを使ってリッチテキストを作成する。

ブロック引用符は、プレーンテキストの電子メールと同様に、行の先頭に > を付けることで入力できます。

## G.1.7 コードとスニペット

バックスティックを使って、いくつかの単語や段落の処理を避けることができる。これは、コードや設定の塊が誤ってマークダウンとして解釈されるのを避けるのに便利です。

### インラインコード

行の一部をシングル・バックスティックで囲むと、例えばインラインでコードを書くことができる：

```
このホストのIPアドレスは...』
```

## コードブロック全体

複数の行にまたがるコード・ブロックでは、例えば、トリプル・バックスティックを使ってそのようなブロックを開始したり終了したりすることができる：

```
# これはここで覚えておきたいネットワーク設定です。
```

```
iface vmbr2 inet static
    アドレス 10.0.0.1/24
    bridge-ports ens20
    bridge-stp off
    bridge-fd 0
    bridge-vlan-aware
    yes bridge-vids 2-
    4094
```

## 付録H

# GNU自由文書ライセンス

バージョン1.3、2008年11月3日

Copyright (C) 2000, 2001, 2002, 2007, 2008 Free Software Foundation, ←'.  
株式会社  
<<http://fsf.org/>>

誰もがこのライセンス文書の逐語的なコピーをコピーし配布することは許可され  
ていますが、変更することは許可されていません。

### 0. 前文

本許諾書の目的は、マニュアルや教科書、あるいはその他の機能的で有用な 文書を、自由という意味で「自由」にすることです。つまり、商業的であれ非商業的であ ろうと、改变の有無に関わらず、それを複製したり再配布したりする効果的な自由 を誰にでも保証することです。第二に、本許諾書は作者や出版者に、他者による改变に対して責任を負わされることなく、自分たちの作品に対する信用を得る方法を保持します。

このライセンスは一種の「コピーレフト」であり、この文書の派生物も同じ意味で自由でなければならないことを意味します。これはGNU一般公衆ライセンス(GNU General Public License)を補完するものであり、自由ソフトウェアのために設計されたコピーレフトのライセンスです。

なぜなら、自由ソフトウェアには自由な文書が必要だからです: 自由なプログラムには、ソフトウェアと同じ自由を提供するマニュアルが付属しているべきです。しかし、本許諾書はソフトウェアのマニュアルに限定されるものではありません。題材や印刷された書籍として出版されるかどうかに関わらず、あらゆるテキスト作品に利用することができます。本許諾書は、主に指導や参照を目的とした作品に推奨します。

## 1. 適用と定義

本許諾書は、著作権者によって本許諾書の条項の下で頒布可能である旨の告知が含まれた、あらゆる媒体のマニュアルやその他の作品に適用される。そのような告知は、本許諾書に記載された条件下でその作品を使用するための、世界を対象とした、期間無制限の、ロイヤリティフリーのライセンスを付与するものです。以下の「文書」とは、そのようなマニュアルや作品を指す。一般利用者はライセンサーであり、「あなた」として扱われます。著作権法上の許可を必要とする方法で作品を複製、変更、頒布する場合、あなたはこのライセンスに同意するものとします。

本件文書の「修正版」とは、本件文書またはその一部をそのまま、あるいは修正を加えて、および/または他の言語に翻訳して複製した著作物を意味する。

「二次的セクション(Secondary Section)」とは、その文書の出版者または著者と、その文書全体の主題との関係（または関連事項）のみを扱い、その文書全体の主題に直接含まれる可能性のあるものを含まない、名称の付録または文書のフロントマター・セクションを指す。（したがって、『文書』の一部が数学の教科書である場合、二次セクションでは数学を一切説明しないことができる）。その関係は、その主題や関連事項との歴史的な関係、あるいはそれらに関する法的、商業的、哲学的、倫理的、政治的な立場の問題となりうる。

「変更不可セクション」とは、『文書』が本許諾書の下でリリースされたことを示す通知の中で、変更不可セクションであるとしてタイトルが指定されている、ある特定のセカンダリ・セクションのことである。セクションが上記の「二次的セクション」の定義に当てはまらない場合、そのセクションを「変更不可セクション」として指定することは許可されない。文書には『変更不可セクション』をゼロ個だけ含めることもできる。もし『文書』がいかなるInvariantセクションも特定しないのであれば、その時点でInvariantセクションは存在しないことになる。

「表紙文(Cover Texts)」とは、『文書』が本許諾書のもとでリリースされることを示す告知に、表紙文(Front-Cover Texts)または裏表紙文(Back-Cover Texts)として記載される、ある短い文章のことである。表表紙テキストは最大で5語、裏表紙テキストは最大で25語である。

「透明な(Transparent)」『文書』のコピーとは、その仕様が一般公衆に利用可能な形式で表現された、機械可読なコピーであって、汎用のテキストエディタ、(ピクセルで構成される画像については)汎用のペイントプログラム、または(図面については)広く利用可能な何らかの描画エディタを用いて文書を直截的に修正するのに適しており、かつテキストフォーマットへの入力、またはテキストフォーマットへの入力に適した様々な形式への自動翻訳に適しているものを指す。そうでなければTransparentなファイルフォーマットで作成されたコピーで、マークアップがある、あるいはマークアップがない場合、読者によるその後の改変を妨げたり、阻止したりするように配置されているものは、Transparentではありません。画像フォーマットは、かなりの量のテキストに使われる場合、「透過的」ではありません。Transparent "でないコピーは "Opaque "と呼ばれます。

透過コピーに適したフォーマットの例としては、マークアップのないプレーンなASCII、Texinfo入力フォーマット、LaTeX入力フォーマット、一般に公開されているDTDを使ったSGMLやXML、人間が修正できるよう設計された標準準拠のシンプルなHTML、PostScript、PDFなどがあります。透過画像フォーマットの例としては、PNG、XCF、JPGがあります。不透明なフォーマットには、プロプライエタリなワードプロセッサーでしか読めず編集できないプロプライエタリなフォーマット、DTDや処理ツールが一般に利用できないSGMLやXML、出力のみを目的として一部のワードプロセッサーが生成する機械生成のHTML、PostScript、

PDFなどがあります。

「タイトルページ」とは、印刷された書籍の場合、タイトルページそれ自体と、本許諾書がタイトルページに掲載することを要求する資料を読みやすく掲載するために必要な次のページを意味する。そのようなタイトルページを持たない形式の著作物については、「タイトルページ」とは、本文の冒頭に先立つ、著作物のタイトルが最も目立つ位置にあるテキストを意味する。

「発行者」とは、『文書』の複製物を公衆に頒布する個人または団体を意味する。

XYZと題された "セクションとは、その題名が正確にXYZであるか、またはXYZを他の言語で翻訳したテキストの後に括弧付きでXYZを含む、『文書』の名前付き小部分を意味する（ここでXYZは、『謝辞』、『献辞』、『推薦』、『沿革』など、後述の特定のセクション名を表す）。（ここでXYZは、『謝辞』、『献辞』、『裏書』、『歴史』など、後述の特定のセクション名を表す）このようなセクションの "タイトルを保持する"ということは、あなたが文書を変更したときに、そのセクションがこの定義に従って "XYZと題された" セクションであり続けることを意味する。

『文書』には、本許諾書が『文書』に適用されることを示す注意書きの隣に、保証の否認が含まれている場合がある。これらの保証の否認は本許諾書に参考として含まれるものとみなされるが、それは保証の否認に関してのみである。

## 2. 逐語コピー

あなたは、本許諾書、著作権表示、および本許諾書が『文書』に適用される旨の使用許諾表示をすべての複製物に複製し、本許諾書の条件に他のいかなる条件も付加しないことを条件に、営利・非営利を問わず、いかなる媒体においても『文書』を複製し頒布することができる。あなたは、あなたが作成または頒布した複製物の閲覧や更なる複製を妨害または制御する技術的手段を用いてはならない。ただし、あなたはコピーの対価として報酬を受け取ることができる。あなたが十分な数の複製物を頒布する場合、あなたは第3項の条件にも従わなければならない。

また、上記と同じ条件でコピーを貸与し、コピーを公に展示することもできます。

## 3. 数量コピー

あなたが『文書』の印刷物(または一般的に印刷された表紙を持つ媒体の複製物)を100部以上発行し、『文書』のライセンス告知でカバーテキストが要求されている場合、あなたはその複製物を、以下のカバーテキストをすべて明瞭かつ読みやすく記載した表紙に封入しなければならない：表表紙には「表紙本文」、裏表紙には「裏表紙本文」。また、両表紙とも、あなたがこれらのコピーの発行者であることを明確かつ判読しやすいように明記しなければなりません。表表紙は、タイトルの全単語を等しく目立たせ、見えるようにしなければならない。さらに、表紙に他の資料を追加することもできます。『文書』の題名を保持し、これらの条件を満たす限り、表紙に限定して変更を加えた複製は、その他の点において逐語的複製として扱うことができる。

どちらかの表紙に必要な文章が多すぎて読みやすく収まらない場合は、最初に挙げたもの（無理なく収まる数）を実際の表紙に掲載し、残りは隣のページに続けて掲載する。

あなたが『文書』の『不透明な複製物』を100部以上発行または頒布する場合、あなたは各『不透明な複製物』に機械可読の『透明な複製物』を同梱するか、または各『不透明な複製物』に、一般のネットワーク利用者が公衆標準ネットワーク・プロトコルを用いて『文書』の完全な『透明な複製物』(追加資料のないものをダウンロードできるコンピュータ・ネットワーク上の場所を明記しなければならない。あなたが後者の選択肢を用いる場合、あなたは『不透明コピー』の配布を数量において開始する際に、この『透明コピー』が、あなたがその版の『不透明コピー』を(直接に、あるいはあなたの代理人や小売店を通じて)公衆に配布した最後の時点から少なくとも1年経過するまでは、記載された場所においてこのようにアクセス可能であり続けるよう、合理的に慎重な手段を講じなければならない。

大量のコピーを再配布する前に、『文書』の作成者によく連絡し、『文書』の最新版を提供する機会を与え

ることが要求されるが、その必要はない。

#### 4. 変更点

あなたは、上記第2項および第3項の条件の下で、『文書』の改変された版を複製し頒布することができる。ただし、改変された版をまさに本許諾書の下でリリースし、改変された版が『文書』の役割を果たし、改変された版の頒布と改変が、そのコピーを所持する誰に対しても許諾されることを条件とする。加えて、あなたは『改変されたバージョン』において以下のことを行わなければならない：

- A. タイトルページ（および表紙がある場合は表紙）には、『文書』のタイトルおよび旧版のタイトル（旧版がある場合は、『文書』の「履歴」の項に記載されているはずである）とは異なるタイトルを使用する。旧版の発行元が許可している場合は、旧版と同じタイトルを使用してもよい。
- B. タイトルページには、修正版の改変の著者として、その文書の主要な著者のうち少なくとも5名（主要な著者の数が5名未満の場合はその全員）とともに責任を負う1名以上の個人または団体を記載する（ただし、その個人または団体がこの要件からあなたを免除する場合を除く）。

- C. タイトルページには、発行者として修正版の発行者名を明記する。
- D. 文書のすべての著作権表示を保存する。
- E. 他の著作権表示に隣接して、あなたの改変に対する適切な著作権表示を追加してください。
- F. 著作権表示の直後に、以下の補遺に示す形式で、本許諾書の条項の下で改変された バージョンの利用を公衆に許可する利用許諾表示を含めること。
- G. そのライセンス告知には、『文書』のライセンス告知に記載されている、変更不可 セクションの完全なリストと必要なカバーテキストを保存すること。
- H. 本許諾書の改変されていないコピーを同封すること。
  - I. "History"と題されたセクションを温存し、そのタイトルを温存し、少なくともタイトルページに記載されている修正版のタイトル、年、新しい著者、出版社を記載した項目をそのセクションに追加する。もし『文書』に「歴史」と題されたセクションがない場合、タイトルページに記載されているように、『文書』のタイトル、年、著者、発行者を記載したセクションを作成し、前文にあるように、修正版について記述した項目を追加する。
  - J. 文書の透過的なコピーを公開するために、その文書で指定されている ネットワークの場所(もしあれば)を保存し、同様に、その文書の基になった旧版の 文書で指定されているネットワークの場所も保存する。これらは「履歴」セクションに置くことができる。あなたは、『文書』それ自体よりも少なくとも4年前に出版された著作物、あるいはその著作物が参照する版の原版発行者が許可を与えた場合には、その著作物のネットワーク上の位置を省略することができる。
- K. 「謝辞」または「献辞」と題されたセクションについては、そのセクションのタイトルを保持し、そこに記載された各寄稿者の謝辞および／または献辞の内容および論調をすべて保持すること。
- L. 文書のすべての不变セクションを、本文もタイトルも変更せずに保存する。セクション番号またはそれに相当するものは、セクションタイトルの一部とはみなされない。
- M. 「裏書」と題されたセクションを削除すること。このようなセクションは、修正版に含めることはできません。
- N. 既存のセクションのタイトルを「エンドースメント」に変更したり、不变のセクションのタイトルと矛盾させたりしないこと。

## O. 保証の免責事項を守ること。

改変された『バージョン』が、『文書』から複製されたものを一切含まない、セカンダリ・セクションとして適格な新しいフロントマター・セクションまたは付録を含む場合、あなたは任意でこれらのセクションの一部または全部を変更不可セクションとして指定することができる。これを行うには、改変された版のライセンス告知にある「変更不可のセクション」の一覧に、それらのタイトルを追加してください。これらのタイトルは、他のいかなるセクションのタイトルとも区別されなければならない。

エンドースメント(Endorsements)と題するセクションを追加してもよい。ただし、そのセクションには、さまざまな当事者によるあなたの修正版に対するエンドースメント、例えば、査読の記述や、ある標準の権威ある定義としてある組織によって承認されたことを示す記述以外を含めることはできない。

あなたは、5語までの文章を表紙テキストとして、25語までの文章を裏表紙テキストとして、修正版の表紙テキストリストの最後に追加することができます。表表紙テキストの一節と裏表紙テキストの一節のみを、一団体によって(または一団体による取り決めによって)追加することができる。もし『文書』に既に同じ表紙のカバー・テキストが含まれており、それが以前にあなたによって追加されたものである場合、あるいはあなたの手配によって追加されたものである場合

しかし、あなたは、古いものを追加した以前の発行者から明確な許可を得た上で、古いものを置き換えることができます。

本許諾書によって、『文書』の著者および発行者は、その名前を宣伝のために使用すること、あるいは改変された『文書』の支持を主張したり示唆したりすることを許可しない。

## 5. 文書結合

あなたは、本許諾書の下でリリースされた他の文書と、改変されたバージョンについて上記第4項で定義された条件の下で、『文書』を結合することができる。ただし、その結合の中に、改変されていないすべてのオリジナル文書の変更不可部分をすべて含み、それらをすべてあなたの結合著作物の変更不可部分としてその許諾告知に記載し、それらのすべての保証の否認を保持することを条件とする。

結合された著作物には本許諾書が一部含まれていればよく、複数の同一の変更不可部分が一つのコピーで置き換えられても構わない。同じ名前で異なる内容の複数の「変更不可部分」が存在する場合、それぞれの「変更不可部分」のタイトルの末尾に、その「変更不可部分」の原著作者または発行者の名前(既知であれば)、あるいは一意の番号を括弧書きで追加して、そのような「変更不可部分」のタイトルを一意にする。結合著作物のライセンス告知にある、変更不可のセクションの一覧にあるセクションのタイトルにも、同じ調整を行う。

同様に、「謝辞」と題されたセクションと「献辞」と題されたセクションもすべて組み合わせてください。  
裏書」と題されたセクションはすべて削除しなければならない。

## 6. 文書集

あなたは、『文書』および本許諾書に基づいてリリースされた他の文書から成る文書集を作成し、様々な文書に含まれる本許諾書の個々のコピーを、その文書集に含まれる単一のコピーに置き換えることができる。ただし、他のすべての点において、各文書の逐語的コピーに関する本許諾書の規則に従うことを条件とする。

あなたは、そのような文書集から一つの文書を抜き出し、本許諾書の下で個別に頒布することができる。ただし、抜き出した文書に本許諾書のコピーを挿入し、その文書の逐語的複製に関する他のすべての点において本許諾書に従うことを条件とする。

## 7. 独立した作品とのアグレゲーション

文書』またはその派生物と、他の別個独立の文書や著作物とを、記憶媒体 や頒布媒体の一巻の中に、あるいは一巻の上に編集したものを「総合体」と呼ぶが、その編集物の結果として生じる著作権が、個々の著作物が許容する範囲を超えてその編集物の利用者の法的権利を制限するために用いられることがない場合には、その編集物は「総体」と呼ばれる。文書』が総集編に含まれる場合、本許諾書は総集編に含まれる他の著作物であって、それ自体が『文書』の二次的著作物でないものには適用されない。

第3項のカバーテキストの要件が当該文書のコピーに適用される場合、当該文書が総体全体の2分の1未満であれば、当該文書のカバーテキストは、総体内の当該文書を囲むカバー、または当該文書が電子形式であればカバーに相当する電子表紙に記載することができる。それ以外の場合は、総体全体を囲む印刷された表紙に掲載しなければならない。

## 8. 翻訳

翻訳は一種の改変とみなされるため、あなたは第4項の条件に従って『文書』の翻訳を頒布することができる。変更不可セクションを翻訳に置き換えるには、そのセクションの特別な許可が必要です。

ただし、あなたはこれらの変更不可部分の原版に加えて、一部または全部の変更不可部分の翻訳を含めることができる。あなたは、本許諾書の翻訳、および『文書』中の全てのライセンス表示、そして保証の否認を含めることができる。ただし、本許諾書の英語原文、およびそれらの通知や否認の原文も含めることを条件とする。翻訳と本許諾書の原版、あるいは通知や免責事項の原版との間に不一致がある場合、原版が優先される。

文書のセクションのタイトルが「謝辞」、「献辞」、「歴史」である場合、そのタイトル（セクション1）を保持するための要件（セクション4）は、通常、実際のタイトルを変更することを必要とする。

## 9. 終了

あなたは、本許諾書の下で明示的に規定されている場合を除き、『文書』を複製、変更、サブライセンス、または頒布することはできない。それ以外の方法で複製、変更、サブライセンス、または頒布しようとする試みは無効であり、本使用許諾に基づくあなたの権利は自動的に消滅します。

ただし、あなたが本許諾書に対する違反をすべて止めた場合、特定の著作権者からのあなたのライセンスは、(a)著作権者が明示的かつ最終的にあなたのライセンスを解除しない限り、また解除するまでは暫定的に、(b)著作権者が違反の停止後60日以前に何らかの合理的な手段であなたに違反を通知しなかった場合には、永続的に復活する。

さらに、著作権者が何らかの合理的な手段であなたに違反を通知し、あなたがその著作権者から本許諾書に対する違反の通知(いかなる作品についても)を初めて受け取り、あなたがその通知を受け取ってから30日以前に違反を是正した場合、特定の著作権者からのあなたのライセンスは永久に復活します。

本節に基づくあなたの権利の終了は、本許諾書に基づいてあなたから複製物や権利を受領した当事者のライセンスを終了させるものではない。あなたの権利が終了し、恒久的に復活しない場合、同じ資料の一部または全部のコピーを受け取ったとしても、それを使用する権利はあなたに与えられない。

## 10. 本ライセンスの将来の改訂

フリーソフトウェアファンデーションは、GNU自由文書利用許諾書の新しい改訂版を隨時公表することができる。そのような新バージョンは、現在のバージョンと精神的には類似しているが、新たな問題や懸念に対処するために詳細が異なるかもしれない。<http://www.gnu.org/copyleft/>をご覧ください。

本許諾書の各バージョンには、識別可能なバージョン番号が付されている。文書』において、本許諾書の特

定のバージョン番号「またはそれ以降のバージョン」が適用されると指定されている場合、あなたはその指定されたバージョンか、フリーソフトウェア財団によって(草案としてではなく)発行されたそれ以降のバージョンのいずれかの条項と条件に従うことができる。文書』に本許諾書のバージョン番号が明記されていない場合、あなたはフリーソフトウェア財団によって(草案としてではなく)公表されたどのバージョンでも選ぶことができる。文書』において、代理人が本許諾書の将来のどの版を使用できるかを決定できると指定されている場合、その代理人がある版を受諾すると公言することで、あなたは『文書』においてその版を選択することが永久に許可される。

## 11. リライセンシング

「大規模マルチ・オーラー・コラボレーション・サイト」(Massive Multiauthor Collaboration Site)(または「MMCサイト」)とは、著作権で保護される作品を公開し、また、誰もがそれらの作品を編集できるように著名な設備を提供するワールド・ワイド・ウェブ・サーバを意味します。誰でも編集できる公開ウィキは、そのようなサーバーの一例です。MMCサイトに含まれる「大規模複数著者の共同作業」(Massive Multiauthor Collaboration)（または「MMC」）とは、このようにしてMMCサイトで公表される、著作権で保護される著作物のあらゆる集合を意味する。

「CC-BY-SA」とは、カリフォルニア州サンフランシスコに主たる事業所を置く非営利法人であるCreative Commons Corporationが発行するCreative Commons Attribution-Share Alike 3.0ライセンス、および同組織が発行する同ライセンスの将来のコピーレフト版を意味します。

「組込」とは、文書の全部または一部を他の文書の一部として発行または再発行することを意味する。

MMCが「再ライセンスの対象となる」のは、それが本許諾書の下でライセンスされ、本許諾書の下で本 MMC以外のどこかで最初に公表され、その後そのMMCに全部または一部が組み込まれたすべての作品が、(1)カバーテキストや不变部分がなく、(2)こうして2008年11月1日より前に組み込まれた場合である。

MMCサイトの運営者は、2009年8月1日以前であればいつでも、同サイトに含まれるMMCをCC-BY-SAの下で再公開することができる。