



## PROXMOX VE 管理ガイド

リリース 9.0.8



2025年8月5日

Proxmox Server Solutions GmbH  
[www.proxmox.com](http://www.proxmox.com)

---

Copyright© 2025 Proxmox Server Solutions GmbH

この文書は、フリーソフトウェア財団が発行した 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 またはそれ以降のバージョンの条件に基づき、複製、頒布、および/または改変が許可されています。不变のセクション、表紙テキスト、裏表紙テキストは存在しません。

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

# 目次

<b>1</b>	<b>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	統合されたバックアップと復元	3
1.4	高可用性クラスタ	3
1.5	柔軟なネットワーク	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ストラクチャ	4
1.7.1	Proxmox VE によるハイパーコンバージドインフラストラクチャ (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	オープンソースを選ぶ理由	5
1.9	Proxmox VEのメリット	5
1.10	ヘルプ	6
1.10.1	Proxmox VE Wiki	6
1.10.2	コミュニティサポートフォーラム	6
1.10.3	メーリングリスト	6
1.10.4	商用サポート	6
1.10.5	バグトラッカー	6
1.11	プロジェクト履歴	6
1.12	Proxmox VE ドキュメントの改善	8
1.13	Proxmox VE の翻訳	8
1.13.1	git を使用した翻訳	8
1.13.2	gitなしで翻訳する	9
1.13.3	翻訳のテスト	9
1.13.4	翻訳の送信	9

<b>2 Proxmox VE のインストール</b>	<b>10</b>
<b>2.1 システム要件</b>	<b>10</b>
<b>2.1.1 評価版用最小要件</b>	<b>10</b>
<b>2.1.2 推奨システム要件</b>	<b>11</b>
<b>2.1.3 簡易性能概要</b>	<b>11</b>
<b>2.1.4 Webインターフェースへのアクセスに対応しているWebブラウザ</b>	<b>11</b>
<b>2.2 インストールメディアの準備</b>	<b>12</b>
<b>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</b>	<b>12</b>
<b>2.2.2 GNU/Linux の手順</b>	<b>12</b>
<b>2.2.3 macOS の手順</b>	<b>13</b>
<b>2.2.4 Windows の手順</b>	<b>14</b>
<b>2.3 Proxmox VE インストーラーの使用</b>	<b>14</b>
<b>2.3.1 インストール後の管理インターフェースへのアクセス</b>	<b>22</b>
<b>2.3.2 高度なLVM設定オプション</b>	<b>23</b>
<b>2.3.3 高度なZFS設定オプション</b>	<b>24</b>
<b>2.3.4 高度なBTRFS設定オプション</b>	<b>25</b>
<b>2.3.5 ZFSパフォーマンスのヒント</b>	<b>25</b>
<b>2.3.6 nomodesetカーネルパラメータの追加</b>	<b>25</b>
<b>2.4 無人インストール</b>	<b>25</b>
<b>2.5 DebianへのProxmox VEのインストール</b>	<b>26</b>
<b>3 ホストシステムの管理</b>	<b>27</b>
<b>3.1 パッケージリポジトリ</b>	<b>27</b>
<b>3.1.1 Proxmox VEのリポジトリ</b>	<b>27</b>
<b>3.1.2 Debianベースリポジトリ</b>	<b>29</b>
<b>3.1.3 Proxmox VE Enterpriseリポジトリ</b>	<b>29</b>
<b>3.1.4 Proxmox VE非サブスクリプションリポジトリ</b>	<b>30</b>
<b>3.1.5 Proxmox VEテストリポジトリ</b>	<b>30</b>
<b>3.1.6 Cephリポジトリ</b>	<b>30</b>
<b>3.1.7 Debianファームウェアリポジトリ</b>	<b>32</b>
<b>3.1.8 SecureApt</b>	<b>33</b>
<b>3.2 システムソフトウェアの更新</b>	<b>33</b>
<b>3.3 Firmware Updates</b>	<b>34</b>
<b>3.3.1 永続的ファームウェア</b>	<b>34</b>
<b>3.3.2 ランタイムファームウェアファイル</b>	<b>35</b>
<b>3.3.3 CPUマイクロコードの更新</b>	<b>35</b>

3.4 ネットワーク設定	37
3.4.1 ネットワーク変更を適用	38
3.4.2 命名規則	38
3.4.3 ネットワーク構成の選択	41
3.4.4 ブリッジを使用したデフォルト設定	42
3.4.5 ルーティング構成	43
3.4.6 iptables によるマスカレード (NAT)	44
3.4.7 Linux ボンディング	45
3.4.8 VLAN 802.1Q	48
3.4.9 ノードでの IPv6 の無効化	50
3.4.10 ブリッジでのMAC学習の無効化	51
3.5 時刻同期	51
3.5.1 カスタム NTP サーバーの使用	51
3.6 外部メトリックサーバー	53
3.6.1 Graphite サーバー設定	53
3.6.2 Influxdb プラグイン設定	54
3.7 ディスクの健全性監視	54
3.8 論理ボリュームマネージャ (LVM)	55
3.8.1 ハードウェア	56
3.8.2 ブートローダー	56
3.8.3 ボリュームグループの作成	56
3.8.4 /var/lib/vz 用に追加の論理ボリューム (LV) を作成する	57
3.8.5 シンプルのサイズ変更	57
3.8.6 LVM-thin プールを作成	58
3.9 Linux 上の ZFS	58
3.9.1 ハードウェア	59
3.9.2 ルートファイルシステムとしてのインストール	59
3.9.3 ZFS RAID レベルに関する考慮事項	60
3.9.4 ZFS dRAID	61
3.9.5 ブートローダー	62
3.9.6 ZFS 管理	62
3.9.7 Eメール通知の設定	67
3.9.8 ZFS メモリ使用量の制限	67
3.9.9 ZFS 上のスワップ	68
3.9.10 暗号化された ZFS データセット	68

---

3.9.11 ZFS における圧縮	70
3.9.12 ZFS 特殊デバイス	70
3.9.13 ZFS プールの機能	71
3.10 BTRFS	72
3.10.1 ルートファイルシステムとしてのインストール	73
3.10.2 BTRFS 管理	73
3.11 Proxmox ノード管理	75
3.11.1 Wake-on-LAN	75
3.11.2 タスク履歴	76
3.11.3 一括ゲスト電源管理	76
3.11.4 初回ゲスト起動遅延	77
3.11.5 バルクゲスト移行	77
3.11.6 バルーニングのRAM使用量目標	77
3.12 証明書管理	77
3.12.1 クラスタ内通信用証明書	77
3.12.2 API および Web GUI 用証明書	78
3.12.3 カスタム証明書のアップロード	78
3.12.4 Let's Encrypt (ACME) 経由の信頼済み証明書	79
3.12.5 ACME HTTP チャレンジプラグイン	80
3.12.6 ACME DNS API チャレンジプラグイン	81
3.12.7 ACME 証明書の自動更新	82
3.12.8 pvenode を使用した ACME の例	82
3.13 ホストブートローダー	85
3.13.1 インストーラが使用するパーティション分割方式	86
3.13.2 ESPの内容をproxmox-boot-toolと同期する	86
3.13.3 使用するブートローダーの決定	89
3.13.4 GRUB	90
3.13.5 systemd-boot	90
3.13.6 カーネルコマンドラインの編集	91
3.13.7 次回起動時のカーネルバージョンの上書き	91
3.13.8 セキュアブート	92
3.14 カーネル・セマページ・マージング (KSM)	95
3.14.1 KSMの影響	95
3.14.2 KSMの無効化	95

---

<b>4 グラフィカルユーザーインターフェース</b>	<b>96</b>
4.1 機能 .....	96
4.2 ログイン .....	97
4.3 GUI 概要 .....	97
4.3.1 ヘッダー .....	98
4.3.2 マイ設定 .....	98
4.3.3 リソースツリー .....	99
4.3.4 ログパネル .....	99
4.4 コンテンツパネル .....	100
4.4.1 データセンター .....	100
4.4.2 ノード .....	101
4.4.3 ゲスト .....	102
4.4.4 ストレージ .....	104
4.4.5 プール .....	105
4.5 タグ .....	106
4.5.1 スタイル設定 .....	107
4.5.2 権限 .....	107
4.6 同意バナー .....	108
<b>5 クラスタ・マネージャ</b>	<b>109</b>
5.1 要件 .....	109
5.2 ノードの準備 .....	110
5.3 クラスタの作成 .....	111
5.3.1 Web GUI経由での作成 .....	111
5.3.2 コマンドライン経由での作成 .....	111
5.3.3 同一ネットワーク内の複数クラスター .....	111
5.4 クラスタへのノードの追加 .....	112
5.4.1 GUI経由でのノードのクラスタへの参加 .....	112
5.4.2 コマンドライン経由でのノードのクラスタへの参加 .....	113
5.4.3 分離されたクラスターネットワークでのノード追加 .....	114
5.5 クラスタノードの削除 .....	114
5.5.1 再インストールせずにノードを分離する .....	117
5.6 クオーラム .....	118
5.7 クラスタネットワーク .....	118
5.7.1 ネットワーク要件 .....	118
5.7.2 ボンディング上のCorosync .....	119

---

5.7.3	クラスタ専用ネットワーク	120
5.7.4	Corosync アドレス	123
5.8	Corosyncの冗長性	124
5.8.1	既存クラスターへの冗長リンクの追加	124
5.9	Proxmox VE クラスターにおける SSH の役割	126
5.9.1	SSH 設定	126
5.9.2	.bashrcおよび関連ファイルの自動実行による落とし穴	126
5.10	Corosync 外部投票サポート	127
5.10.1	QDeviceの技術概要	127
5.10.2	サポート対象のセットアップ	127
5.10.3	QDevice-Net 設定	128
5.10.4	よくある質問	129
5.11	Corosync 設定	130
5.11.1	corosync.conf の編集	130
5.11.2	トラブルシューティング	131
5.11.3	Corosync 設定用語集	131
5.12	クラスタコールドスタート	131
5.13	ゲスト VMID 自動選択	132
5.14	ゲスト移行	132
5.14.1	移行タイプ	132
5.14.2	移行ネットワーク	133
<b>6</b>	<b>Proxmox クラスタファイルシステム (pmxcfs)</b>	<b>135</b>
6.1	POSIX互換性	135
6.2	ファイルアクセス権	136
6.3	テクノロジー	136
6.4	ファイルシステムのレイアウト	136
6.4.1	ファイル	136
6.4.2	シンボリックリンク	137
6.4.3	デバッグ用特別ステータスファイル (JSON)	137
6.4.4	デバッグの有効化/無効化	138
6.5	リカバリ	138
6.5.1	クラスタ構成の削除	138
6.5.2	障害発生ノードからのゲストの復旧/移動	138
<b>7</b>	<b>Proxmox VE ストレージ</b>	<b>140</b>
7.1	ストレージタイプ	140
7.1.1	シンプロビジョニング	141

---

---

7.2	ストレージ構成	142
7.2.1	ストレージプール	142
7.2.2	共通ストレージプロパティ	143
7.3	ボリューム	144
7.3.1	ボリュームの所有権	144
7.4	コマンドラインインターフェースの使用	144
7.4.1	例	145
7.5	ディレクトリバックエンド	146
7.5.1	設定	147
7.5.2	ファイル命名規則	147
7.5.3	ストレージ機能	148
7.5.4	例	148
7.6	NFS バックエンド	149
7.6.1	設定	149
7.6.2	ストレージ機能	150
7.6.3	例	150
7.7	CIFS バックエンド	150
7.7.1	設定	151
7.7.2	ストレージ機能	152
7.7.3	例	152
7.8	Proxmox バックアップサーバー	152
7.8.1	設定	153
7.8.2	ストレージ機能	154
7.8.3	暗号化	154
7.8.4	例: CLI 経由でのストレージ追加	155
7.9	ローカル ZFS プール バックエンド	155
7.9.1	設定	156
7.9.2	ファイル命名規則	156
7.9.3	ストレージ機能	157
7.9.4	例	157
7.10	LVM バックエンド	157
7.10.1	設定	157
7.10.2	ファイル命名規則	158
7.10.3	ストレージ機能	159
7.10.4	例	159
7.11	LVM スリムバックエンド	159

---

---

7.11.1 設定	160
7.11.2 ファイル命名規則	160
7.11.3 ストレージ機能	160
7.11.4 例	160
7.12 Open-iSCSI イニシエーター	161
7.12.1 設定	161
7.12.2 ファイル命名規則	161
7.12.3 ストレージ機能	162
7.12.4 例	162
7.13 ユーザーモード iSCSI バックエンド	162
7.13.1 設定	162
7.13.2 ストレージ機能	162
7.14 Ceph RADOS ブロックデバイス (RBD)	163
7.14.1 構成	163
7.14.2 認証	164
7.14.3 Cephクライアント設定 (オプション)	165
7.14.4 ストレージ機能	165
7.15 Ceph ファイルシステム (CephFS)	165
7.15.1 構成	166
7.15.2 認証	167
7.15.3 ストレージ機能	167
7.16 BTRFS バックエンド	168
7.16.1 設定	168
7.16.2 スナップショット	168
7.17 ZFS over ISCSI バックエンド	169
7.17.1 設定	169
7.17.2 ストレージ機能	171
<b>8 ハイパーコンバージド Ceph クラスターの展開</b>	<b>172</b>
8.1 はじめに	172
8.2 用語	173
8.3 健全なCephクラスターのための推奨事項	173
8.4 Ceph の初期インストールと設定	176
8.4.1 Webベースのウィザードの使用	176
8.4.2 CephパッケージのCLIインストール	178
8.4.3 CLIによるCephの初期設定	178
8.5 Ceph Monitor	179

---

---

8.5.1	モニターの作成	179
8.5.2	モニターの破棄	180
8.6	Ceph Manager	180
8.6.1	マネージャーを作成	180
8.6.2	マネージャーを破棄	180
8.7	Ceph OSD	181
8.7.1	OSDの作成	181
8.7.2	OSDの破棄	183
8.8	Ceph プール	184
8.8.1	プールの作成と編集	184
8.8.2	エラー訂正符号化プール	186
8.8.3	プールの破棄	188
8.8.4	PGオートスケーラー	188
8.9	Ceph CRUSH およびデバイスクラス	189
8.10	Ceph クライアント	191
8.11	CephFS	192
8.11.1	メタデータサーバー (MDS)	192
8.11.2	CephFSの作成	193
8.11.3	CephFS を破棄	194
8.12	Cephのメンテナンス	194
8.12.1	OSDの交換	194
8.12.2	トリム/破棄	195
8.12.3	スクラップ & ディープスクラップ	195
8.12.4	Proxmox VE+ Ceph HCI クラスターのシャットダウン	195
8.13	Cephの監視とトラブルシューティング	196
8.13.1	トラブルシューティング	196
<b>9</b>	<b>ストレージレプリケーション</b>	<b>198</b>
9.1	サポートされるストレージの種類	198
9.2	スケジュール形式	199
9.3	エラー処理	199
9.3.1	発生しうる問題	199
9.3.2	エラー発生時のゲスト移行	199
9.3.3	例	199
9.4	ジョブの管理	200
9.5	ネットワーク	201
9.6	コマンドラインインターフェースの例	201

---

<b>10 QEMU/KVM 仮想マシン</b>	<b>202</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	202
10.2 仮想マシンの設定	203
10.2.1 一般設定	203
10.2.2 OS設定	204
10.2.3 システム設定	204
10.2.4 ハードディスク	206
10.2.5 CPU	208
10.2.6 メモリ	213
10.2.7 メモリ暗号化	215
10.2.8 ネットワークデバイス	217
10.2.9 表示	219
10.2.10 USBバススルー	220
10.2.11 BIOS および UEFI	220
10.2.12 トラステッド・プラットフォーム・モジュール (TPM)	221
10.2.13 仮想マシン間共有メモリ	222
10.2.14 オーディオデバイス	222
10.2.15 VirtIO RNG	223
10.2.16 Virtiofs	223
10.2.17 デバイスの起動順序	225
10.2.18 仮想マシンの自動起動とシャットダウン	225
10.2.19 QEMU ゲストエージェント	226
10.2.20 SPICE の機能強化	228
10.3 移行	229
10.3.1 オンライン移行	229
10.3.2 オフライン移行	230
10.4 コピーとクローン	231
10.5 仮想マシンテンプレート	232
10.6 VM 生成 ID	232
10.7 仮想マシンのインポート	233
10.7.1 インポートウィザード	233
10.7.2 CLI 経由での OVF/OVA インポート	235
10.8 Cloud-Init のサポート	236
10.8.1 クラウドインイシャライズのテンプレート準備	237
10.8.2 Cloud-Init テンプレートのデプロイ	239
10.8.3 カスタム Cloud-Init 設定	239
10.8.4 Windows での Cloud-Init	240

---

10.8.5 Cloudbase-Init テンプレートの準備 .....	240
10.8.6 Cloudbase-Init と Sysprep .....	241
10.8.7 Cloud-Init固有のオプション .....	242
10.9 PCI(e) パススルー .....	243
10.9.1 一般的な要件 .....	244
10.9.2 ホストデバイスのパススルー .....	246
10.9.3 SR-IOV .....	249
10.9.4 仲介デバイス (vGPU、GVT-g) .....	249
10.9.5 クラスタでの使用 .....	250
10.9.6 vIOMMU (エミュレートされたIOMMU) .....	250
10.10 フックスクリプト .....	251
10.11 ハイバネーション .....	251
10.12 リソースマッピング .....	252
10.13 qmによる仮想マシンの管理 .....	254
10.13.1 CLI 使用例 .....	255
10.14 設定 .....	255
10.14.1 ファイル形式 .....	256
10.14.2 スナップショット .....	256
10.14.3 オプション .....	257
10.15 ロック .....	283
<b>11 Proxmox コンテナ ツールキット</b>	<b>284</b>
11.1 テクノロジー概要 .....	284
11.2 サポート対象ディストリビューション .....	285
11.2.1 Alpine Linux .....	285
11.2.2 Arch Linux .....	285
11.2.3 CentOS、Alma Linux、Rocky Linux .....	286
11.2.4 Debian .....	286
11.2.5 Devuan .....	287
11.2.6 Fedora .....	287
11.2.7 Gentoo .....	287
11.2.8 OpenSUSE .....	287
11.2.9 Ubuntu .....	287
11.3 コンテナイメージ .....	288
11.4 コンテナ設定 .....	289
11.4.1 一般設定 .....	289
11.4.2 CPU .....	290
11.4.3 メモリ .....	291

---

---

11.4.4 マウントポイント	292
11.4.5 ネットワーク	295
11.4.6 コンテナの自動起動とシャットダウン	296
11.4.7 フックスクリプト	297
11.5 セキュリティに関する考慮事項	297
11.5.1 AppArmor	297
11.5.2 制御グループ ( <i>cgroup</i> )	298
11.6 ゲストオペレーティングシステムの設定	299
11.7 コンテナストレージ	300
11.7.1 FUSE マウント	301
11.7.2 コンテナ内のクォータの使用	301
11.7.3 コンテナ内のACLの使用	302
11.7.4 コンテナのマウントポイントのバックアップ	302
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	302
11.8 バックアップと復元	302
11.8.1 コンテナのバックアップ	302
11.8.2 コンテナバックアップの復元	302
11.9 pct によるコンテナの管理	303
11.9.1 CLI 使用例	304
11.9.2 デバッグルогの取得	305
11.10 移行	305
11.11 設定	305
11.11.1 ファイル形式	306
11.11.2 スナップショット	306
11.11.3 オプション	307
11.12 ロック	313
<b>12 ソフトウェア定義ネットワーク</b>	<b>314</b>
12.1 はじめに	314
12.2 サポート状況	314
12.2.1 履歴	314
12.2.2 現在のステータス	315
12.3 インストール	315
12.3.1 SDN コア	315
12.3.2 DHCP IPAM	315
12.3.3 FRルーティング	316
12.4 設定の概要	316

---

12.5 テクノロジーと構成	316
12.6 ゾーン	317
12.6.1 共通オプション	317
12.6.2 シンプルゾーン	317
12.6.3 VLAN ゾーン	318
12.6.4 QinQ ゾーン	318
12.6.5 VXLAN ゾーン	318
12.6.6 EVPNゾーン	319
12.7 vNets	320
12.8 サブネット	321
12.9 コントローラ	322
12.9.1 EVPN コントローラ	322
12.9.2 BGP コントローラ	322
12.9.3 ISIS コントローラ	323
12.10 ファブリック	324
12.10.1 インストール	324
12.10.2 権限	324
12.10.3 設定	325
12.10.4 OpenFabric	326
12.10.5 OSPF	328
12.11 IPAM	330
12.11.1 PVE IPAM プラグイン	330
12.11.2 NetBox IPAM プラグイン	330
12.11.3 phplIPAM プラグイン	330
12.12 DNS	331
12.12.1 PowerDNS プラグイン	331
12.13 DHCP	331
12.13.1 設定	332
12.13.2 プラグイン	332
12.14 ファイアウォール統合	333
12.14.1 仮想ネットワークとサブネット	333
12.14.2 IPAM	335
12.15 例	335
12.15.1 シンプルなゾーンの例	335
12.15.2 ソースNATの例	336
12.15.3 VLAN設定例	336
12.15.4 QinQ 設定例	337

12.15.5 VXLAN 設定例	337
12.15.6 EVPN 設定例	338
12.16 注記	340
12.16.1 複数の EVPN 出口ノード	340
12.16.2 VXLAN IPSEC 暗号化	340
<b>13 Proxmox VE ファイアウォール</b>	<b>341</b>
13.1 方向とゾーン	341
13.1.1 方向	341
13.1.2 ゾーン	342
13.2 設定ファイル	342
13.2.1 クラスタ全体の設定	342
13.2.2 ホスト固有の設定	344
13.2.3 VM/コンテナ設定	346
13.2.4 vNet 構成	347
13.3 ファイアウォールルール	348
13.4 セキュリティグループ	349
13.5 IP エイリアス	350
13.5.1 標準 IP エイリアス local_network	350
13.6 IP セット	350
13.6.1 標準 IP セット管理	351
13.6.2 標準 IP セットブラックリスト	351
13.6.3 標準IPセット ipfilter-net*	351
13.7 サービスとコマンド	351
13.8 デフォルトのファイアウォールルール	352
13.8.1 データセンター着信/発信 DROP/REJECT	352
13.8.2 VM/CT 着信/発信 DROP/REJECT	353
13.9 ファイアウォールルールの記録	353
13.9.1 ユーザー定義ファイアウォールルールのログ記録	354
13.10 ヒントとコツ	354
13.10.1 FTPを許可する方法	354
13.10.2 Suricata IPS 統合	355
13.11 IPv6に関する注意事項	355
13.12 Proxmox VE が使用するポート	355
13.13 nftables	356
13.13.1 インストールと使用方法	356
13.13.2 使用方法	357

13.13.3 便利なコマンド	357
<b>14 ユーザー管理</b>	<b>360</b>
14.1 ユーザー	360
14.1.1 システム管理者	361
14.2 グループ	361
14.3 API トークン	361
14.4 リソースプール	361
14.5 認証レルム	362
14.5.1 Linux PAM 標準認証	362
14.5.2 Proxmox VE 認証サーバー	362
14.5.3 LDAP	363
14.5.4 Microsoft Active Directory (AD)	364
14.5.5 LDAP ベースのレルムの同期	364
14.5.6 OpenID Connect	367
14.6 二要素認証	369
14.6.1 利用可能な第二要素	369
14.6.2 Realm による強制的な二要素認証	370
14.6.3 二要素認証の制限とロックアウト	370
14.6.4 ユーザー設定のTOTP認証	370
14.6.5 TOTP	372
14.6.6 WebAuthn	372
14.6.7 リカバリキー	372
14.6.8 サーバーサイド Webauthn 設定	373
14.6.9 サーバーサイド U2F 設定	373
14.6.10 ユーザーとして U2F を有効化	374
14.7 権限管理	374
14.7.1 役割	374
14.7.2 特権	375
14.7.3 オブジェクトとパス	377
14.7.4 プール	378
14.7.5 必要な権限は？	378
14.8 コマンドラインツール	379
14.9 実例	380
14.9.1 管理者グループ	380
14.9.2 監査人	380
14.9.3 委任ユーザー管理	381

14.9.4 監視用限定 API トークン	381
14.9.5 リソースプール	381
<b>15 高可用性</b>	<b>383</b>
15.1 要件	384
15.2 リソース	385
15.3 管理タスク	385
15.4 仕組み	386
15.4.1 サービス状態	387
15.4.2 ローカルリソースマネージャー	388
15.4.3 クラスタリソースマネージャ	389
15.5 HAシミュレータ	390
15.6 構成	391
15.6.1 リソース	391
15.6.2 グループ	393
15.6.3 ルール	395
15.6.4 ルールの競合とエラー	399
15.7 フェンシング	400
15.7.1 Proxmox VE のフェンシング方法	400
15.7.2 ハードウェアウォッチドッグの設定	400
15.7.3 フェンスされたサービスの復旧	401
15.8 障害ポリシーを開始	401
15.9 エラー回復	401
15.10 パッケージの更新	402
15.11 ノードメンテナンス	402
15.11.1 メンテナンスマード	402
15.11.2 シャットダウンポリシー	403
15.12 クラスタリソーススケジューリング	405
15.12.1 基本スケジューラ	406
15.12.2 静的負荷スケジューラ	406
15.12.3 CRS スケジューリングポイント	406
<b>16 バックアップと復元</b>	<b>408</b>
16.1 バックアップモード	408
16.1.1 VMバックアップの擷取	410
16.1.2 CT 変更検出モード	410
16.2 バックアップファイル名	411
16.3 バックアップファイル圧縮	411
16.4 バックアップの暗号化	412

16.5 バックアップジョブ	412
16.6 バックアップの保持期間	414
16.6.1 シミュレータのブルーニング	415
16.6.2 保持設定の例	415
16.7 バックアップ保護	416
16.8 バックアップノート	416
16.9 復元	417
16.9.1 帯域幅制限	417
16.9.2 ライブ復元	418
16.9.3 単一ファイル復元	418
16.10 設定	419
16.11 フックスクリプト	422
16.12 ファイル除外	422
16.13 例	423
<b>17 通知</b>	<b>424</b>
17.1 概要	424
17.2 通知対象	424
17.2.1 Sendmail	424
17.2.2 SMTP	426
17.2.3 Gotify	427
17.2.4 Webhook	428
17.3 通知マッチャー	430
17.3.1 マッチャーオプション	431
17.3.2 カレンダーマッチングルール	431
17.3.3 フィールドマッチングルール	431
17.3.4 重大度マッチングルール	432
17.3.5 例	432
17.4 通知イベント	432
17.5 システムメール転送	433
17.6 権限	433
17.7 通知モード	433
17.8 通知テンプレートのオーバーライド	434
<b>18 重要なサービスデーモン</b>	<b>435</b>
18.1 pvedaemon - Proxmox VE API デーモン	435
18.1.1 労働者数	435
18.2 pveproxy - Proxmox VE API プロキシデーモン	435
18.2.1 ホストベースのアクセス制御	436

---

18.2.2 リスニング IP アドレス .....	436
18.2.3 SSL 暗号スイート .....	437
18.2.4 サポートされている TLS バージョン .....	437
18.2.5 Diffie-Hellman パラメータ .....	438
18.2.6 代替 HTTPS 証明書 .....	438
18.2.7 応答圧縮 .....	438
18.2.8 実際のクライアント IP の記録 .....	438
18.2.9 ワーカー数 .....	439
18.3 pvestatd - Proxmox VE ステータスデーモン .....	439
18.4 spiceproxy - SPICE プロキシサービス .....	439
18.4.1 ホストベースのアクセス制御 .....	439
18.5 pvescheduler - Proxmox VE スケジューラーデーモン .....	440
<b>19 便利なコマンドラインツール</b>	<b>441</b>
19.1 pvesubscription - サブスクリプション管理 .....	441
19.2 pveperf - Proxmox VE ベンチマークスクリプト .....	441
19.3 Proxmox VE API のシェルインターフェース .....	442
19.3.1 例 .....	442
<b>20 よくある質問</b>	<b>443</b>
<b>21 参考文献</b>	<b>446</b>
21.1 Proxmox VEに関する書籍 .....	446
21.2 関連技術に関する書籍 .....	446
21.3 関連トピックに関する書籍 .....	447
<b>A コマンドラインインターフェース</b>	<b>448</b>
A.1 一般 .....	448
A.2 出力形式オプション [FORMAT_OPTIONS] .....	448
A.3 pvesm - Proxmox VE ストレージマネージャ .....	449
A.4 pvesubscription - Proxmox VE サブスクリプションマネージャ .....	463
A.5 pveperf - Proxmox VE ベンチマークスクリプト .....	463
A.6 pveceph - Proxmox VE ノード上の CEPH サービスを管理する .....	464
A.7 pvenode - Proxmox VE ノード管理 .....	471
A.8 pvesh - Proxmox VE API 用のシェルインターフェース .....	479
A.9 qm - QEMU/KVM 仮想マシンマネージャ .....	481
A.10 qmrestore - QemuServer vzdump バックアップの復元 .....	526
A.11 pct - Proxmox コンテナツールキット .....	526

---

<b>A.12 pveam - Proxmox VE アプライアンスマネージャー</b>	551
<b>A.13 pvecm - Proxmox VE Cluster Manager</b>	552
<b>A.14 pvesr - Proxmox VE ストレージレプリケーション</b>	555
<b>A.15 pveum - Proxmox VE ユーザーマネージャー</b>	560
<b>A.16 vzdump - VMおよびコンテナ用バックアップユーティリティ</b>	576
<b>A.17 ha-manager - Proxmox VE HA マネージャー</b>	579
<b>B サービスデーモン</b>	587
<b>B.1 pve-firewall - Proxmox VE ファイアウォールデーモン</b>	587
<b>B.2 pvedaemon - Proxmox VE API デーモン</b>	588
<b>B.3 pveproxy - Proxmox VE API プロキシデーモン</b>	589
<b>B.4 pvestatd - Proxmox VE ステータスデーモン</b>	590
<b>B.5 spiceproxy - SPICE プロキシサービス</b>	590
<b>B.6 pmxcfs - Proxmox クラスタファイルシステム</b>	591
<b>B.7 pve-ha-crm - クラスタリソースマネージャデーモン</b>	591
<b>B.8 pve-ha-lrm - ローカルリソースマネージャデーモン</b>	592
<b>B.9 pvescheduler - Proxmox VE スケジューラデーモン</b>	593
<b>C 設定ファイル</b>	594
<b>C.1 一般</b>	594
<b>C.1.1 プロパティ名のキャッシング</b>	594
<b>C.2 データセンター構成</b>	594
<b>C.2.1 ファイル形式</b>	594
<b>C.2.2 オプション</b>	595
<b>D カレンダーイベント</b>	600
<b>D.1 スケジュール形式</b>	600
<b>D.2 詳細仕様</b>	600
<b>D.2.1 例:</b>	601
<b>E QEMU vCPU リスト</b>	603
<b>E.1 はじめに</b>	603
<b>E.2 Intel CPU タイプ</b>	603
<b>E.3 AMD CPUの種類</b>	604
<b>F ファイアウォールマクロ定義</b>	605
<b>G Markdown入門</b>	618
<b>G.1 Markdown の基本</b>	618

---

G.1.1	見出し	618
G.1.2	強調	618
G.1.3	リンク	619
G.1.4	リスト	619
G.1.5	表	620
G.1.6	ブロック引用	620
G.1.7	コードとスニペット	620

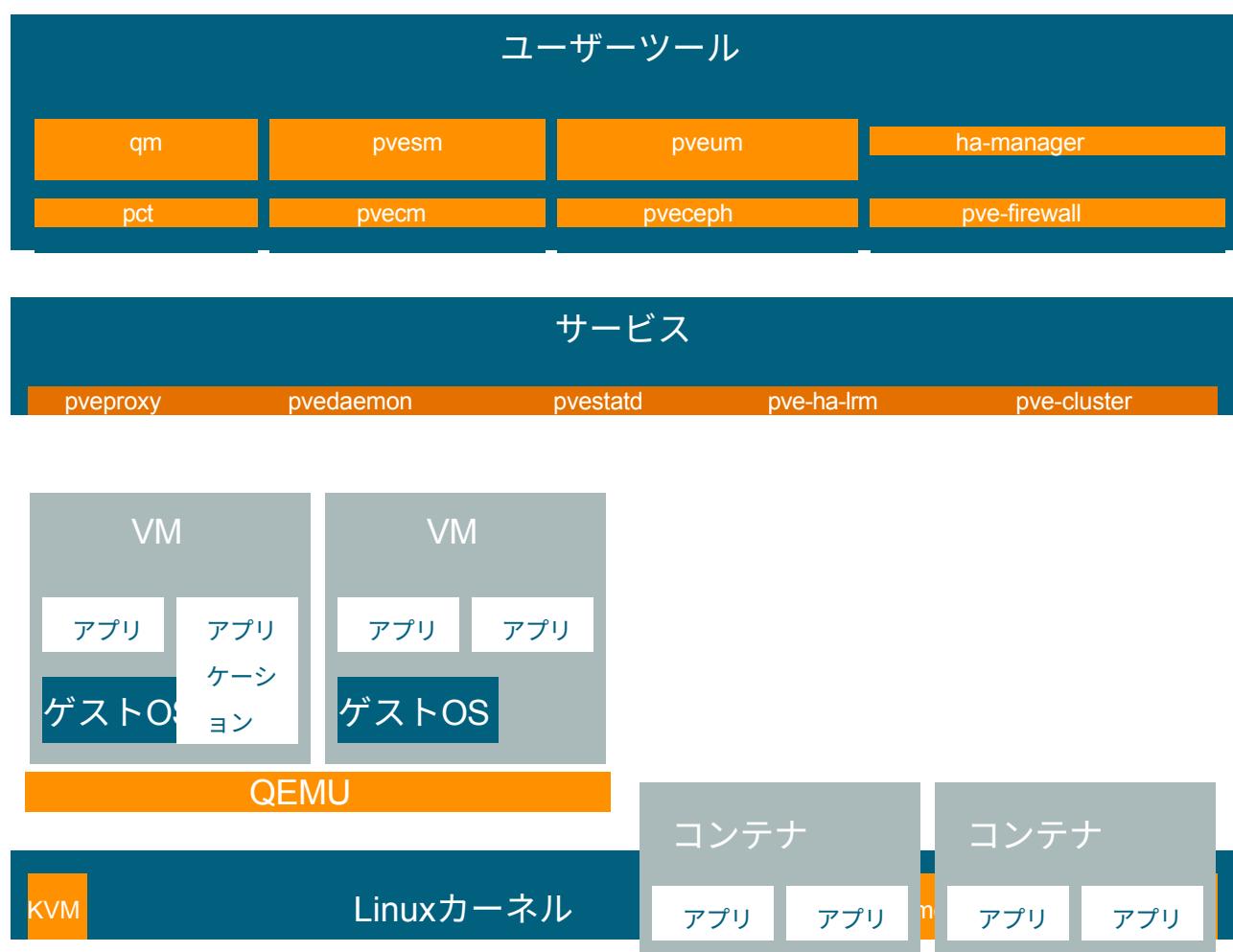
**H GNUフリー文書利用許諾契約書**

622

# 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 をベースとしており、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、カーネルベース仮想化 (KVM) とコンテナベース仮想化 (LXC) の 2 つの仮想化技術を実装しています。

主要な設計目標の一つは、管理を可能な限り容易にすることでした。Proxmox VE は単一ノードで利用することも、多数のノードからなるクラスターを構築することも可能です。すべての管理タスクはウェブベースの管理インターフェースを使用して実行でき、初心者ユーザーでも数分で Proxmox VE の設定とインストールを完了できます。



## 1.1 中央管理

多くの人は單一ノードから始めますが、Proxmox VE は大規模なクラスタノードセットまでスケールアウトできます。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提供されます。

### ユニークなマルチマスター設計

統合されたウェブベースの管理インターフェースにより、すべてのKVMゲスト、Linuxコンテナ、さらにはクラスタ全体を明確に把握できます。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。別途複雑で高価な管理サーバーを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、設定ファイルを保存するためのデータベース駆動型ファイルシステムである独自のProxmox Clusterファイルシステム (pmxcfs) を使用します。これにより、数千台の仮想マシンの設定を保存することが可能になります。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にリアルタイムで複製されます。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保存しますが、データのコピーはRAMにも存在し、最大30MBのストレージ容量を提供します。これは数千台のVMを十分に収容できる容量です。

Proxmox VEはこの独自のクラスターファイルシステムを採用する唯一の仮想化プラットフォームです。

### Webベース管理インターフェース

Proxmox VEは操作が簡単です。管理タスクは付属のWebベース管理インターフェースで実行可能であり、別途管理ツールや大規模データベースを必要とする追加管理ノードのインストールは不要です。マルチマスターツールにより、クラスター内の任意のノードから全体を管理できます。JavaScript フレームワーク (Ex-tJS) に基づく中央の Web ベースの管理により、GUI からすべての機能を制御し、各ノードの履歴やシステムログを一覧表示することができます。これには、バックアップや復元ジョブの実行、ライブマイグレーション、HA によってトリガーされるアクティビティなどが含まれます。

### コマンドライン

UnixシェルやWindows PowerShellの利便性に慣れた上級ユーザー向けに、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。このインターフェースはインテリジェントなタブ補完機能を備え、UNIXマニュアルページ形式の完全なドキュメントが付属しています。

### REST API

Proxmox VEはRESTful APIを採用しています。主要データ形式としてJSONを選択し、API全体はJSON Schemaを用い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などのサードパーティ管理ツールとの迅速かつ容易な統合が可能となります。

### ロールベース管理

ロールベースのユーザーおよび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）に対して詳細なアクセス権を定義できます。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可能になります。この概念はアクセス制御リスト（ACL）としても知られています。各権限は、特定のパスにおける主体（ユーザーまたはグループ）とロール（権限の集合）を指定します。

### 認証レベル

Proxmox VEは、Microsoft Active Directory、LDAP、Linux PAM標準認証、または組み込みのProxmox VE認証サーバーなど、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、あるいは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、必要な数のストレージ定義を設定可能です。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。

共有ストレージにVMを保存する主な利点は、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、稼働中のマシンをダウントIMEなしでライブマイグレーションできることです。

現在サポートしているネットワークストレージタイプは以下の通りです：

- 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共有
- CIFS共有
- Ceph RBD
- CephFS
- iSCSI LUN またはファイバー接続 SAN を直接使用します。

サポートされているローカルストレージの種類は次のとおりです:

- 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キングデバイス)
- ディレクトリ (既存ファイルシステム上のストレージ)
- ZFS

## 1.3 統合バックアップと復元

統合バックアップツール (`vzdump`) は、稼働中のコンテナおよびKVMゲストの一貫性のあるスナップショットを作成します。基本的に、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生成します。

さらに、[Proxmox Backup Serverとの統合](#)により、フルバックアップとデータ重複排除の組み合わせ、クライアント側暗号化の実行、テープまたはS3オブジェクトストレージへのバックアップ、他のオフサイトProxmox Backup Server設置環境との同期といった高度な機能が提供されます。

QEMU/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形式は、VMバックアップを高速かつ効率的に保存するために最適化されています（スペースファイル、順不同データ、最小限のI/O）。

## 1.4 高可用性クラスター

マルチノードのProxmox VE HA クラスタにより、高可用性仮想サーバーの定義が可能になります。Proxmox VE HA クラスタは、実績のあるLinux HA テクノロジーに基づいており、安定した信頼性の高いHAサービスを提供します。

## 1.5 柔軟なネットワーク構成

Proxmox VEはブリッジ型ネットワークモデルを採用しています。すべてのVMは、各ゲストの仮想ネットワークケーブルが同一スイッチに接続されているかのように、1つのブリッジを共有できます。VMを外部に接続するには、ブリッジを物理ネットワークカードに接続し、TCP/IP設定を割り当てます。

さらなる柔軟性のために、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。これにより、Linux ネットワークスタックの全機能を活用し、Proxmox VE ホスト向けに複雑で柔軟な仮想ネットワークを構築できます。

## 1.6 統合ファイアウォール

統合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できます。一般的な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

Proxmox VEは、コンピューティング、ストレージ、ネットワークリソースを緊密に統合し、高可用性クラスタ、バックアップ/復元、災害復旧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定義であり、相互に互換性があります。

そのため、集中管理型のWeb管理インターフェースを通じて単一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 (HCI) を展開・管理する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ストラクチャ (HCI) の利点

ハイパーコンバージドインフラストラクチャ (HCI) は、インフラ需要が高い一方で管理予算が限られている環境、リモートオフィスや支社環境などの分散型セットアップ、仮想プライベートクラウドやパブリッククラウドにおいて特に有用です。

HCIは以下の利点を提供します：

- スケーラビリティ：コンピューティング、ネットワーク、ストレージデバイスのシームレスな拡張（サーバーとストレージを迅速かつ独立してスケールアップ）。
- 低コスト：Proxmox VEはオープンソースであり、コンピューティング、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すべてのコンポーネントを統合しています。高価なコンピューティング/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。
- データ保護と効率性：バックアップや災害復旧などのサービスが統合されています。
- シンプルさ：容易な設定と集中管理。
- オープンソース：ベンダーロックインなし。

## 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の展開を緊密に統合サポートします。例えば以下の2つのストレージ技術をWebインターフェースのみで展開・管理可能です：

- **Ceph**: 自己修復機能と自己管理機能を兼ね備えた、共有型で信頼性が高く、高度にスケーラブルなストレージシステム。[Proxmox VEノード上でCephサービスを管理する方法](#)をご覧ください。
- **ZFS**: データ破損に対する広範な保護機能、多様なRAIDモード、高速かつ低コストのスナップショットなどを備えた、ファイルシステムと論理ボリュームマネージャを統合したシステム。[Proxmox VEノード上でZFSの力を活用する方法](#)をご覧ください。

上記に加え、Proxmox VEは幅広い追加ストレージ技術の統合をサポートしています。詳細は「[ストレージマネージャー](#)」の章でご確認ください。

## 1.8 オープンソースの理由

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基盤としています。Proxmox VEのソースコードは[GNU Affero General Public Licenseバージョン3](#)の下で公開されています。これは、いつでもソースコードを自由に確認したり、プロジェクトに自ら貢献したりできることを意味します。

Proxmoxでは可能な限りオープンソースソフトウェアを採用することを約束しています。オープンソースソフトウェアの使用は、すべての機能への完全なアクセス権、そして高いセキュリティと信頼性を保証します。私たちは、誰もがソフトウェアを実行し、それを基盤に構築し、変更をプロジェクトにフィードバックする権利を持つべきだと考えています。Proxmoxが製品が常にプロフェッショナルな品質基準を満たすことを保証する一方で、誰もが貢献することを奨励しています。

オープンソースソフトウェアはコスト削減にも寄与し、中核インフラを単一ベンダーへの依存から解放します。

## 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
- 迅速なインストールと使いやすい
- Webベースの管理インターフェース
- REST API
- 大規模な活発なコミュニティ
- 管理コストが低く、導入が簡単

## 1.10 ヘルプの入手

### 1.10.1 Proxmox VE Wiki

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提供コンテンツを統合しています。

### 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、ユーザーが知識について議論し、共有することを常に推奨しています。このフォーラムは Proxmox 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参加しています。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

メールを介して Proxmox VE コミュニティと迅速にコミュニケーションを取る方法です。

- 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は完全なオープンソースであり、貢献を歓迎します！開発者向けの主なコミュニケーションチャネルは以下の通りです：

- 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 商用サポート

Proxmox Server Solutions GmbHは、[Proxmox VEサブスクリプションサービスプラン](#)として提供される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お持ちの全ユーザーは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、プレミアムサブスクリプションでは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開発者による保証された応答時間でのヘルプとサポートが提供されます。

ボリュームディスカウントやその他の詳細情報については、[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は公開バグトラッカーを<https://bugzilla.proxmox.com>で運営しています。問題が発生した場合は、こちらで報告してください。問題とはバグだけでなく、新機能や機能強化のリクエストも含まれます。バグトラッカーは問題の追跡を支援し、解決時に通知を送信します。

## 1.11 プロジェクトの沿革

プロジェクトは2007年に開始され、2008年に最初の安定版がリリースされました。当時はコンテナにOpenVZ、仮想マシンにQEMUとKVMを使用していました。クラスタ機能は限定的で、ユーザーインターフェースもシンプル（サーバー生成のWebページ）でした。

しかしCorosyncクラスタスタックを活用した新機能を迅速に開発し、特に新開発のProxmoxクラスタファイルシステム（pmxefs）の導入は大きな飛躍となりました。これによりクラスタの複雑性がユーザーから完全に隠蔽され、16ノードのクラスタ管理が単一ノードと同等の簡便さを実現したのです。

JSON-Schemaで記述された完全な宣言的仕様を備えた新REST APIの導入により、他者がProxmox VEを自社のインフラに統合できるようになり、追加サービスの提供も容易になりました。

さらに、この新REST APIにより、従来のユーザーインターフェースをJavaScriptを用いたモダンなクライアントサイドのシングルページアプリケーションに置き換えることが可能になりました。また、旧式のJavaベースのVNCコンソールコードをnoVNCに置き換えたため、仮想マシンの管理にはウェブブラウザのみが必要となりました。

多様なストレージタイプのサポートも重要な課題です。特にProxmox VEは2014年、Linux上でZFSをデフォルト搭載した初のディストリビューションとなりました。もう一つの画期的な機能は、ハイパーバイザーノード上でCephストレージを実行・管理できる点です。このような構成は極めてコスト効率に優れています。

プロジェクト開始時、当社はKVMの商用サポートを提供する先駆的企業の一つでした。KVMプロジェクト自体も継続的に進化し、現在では広く利用されるハイパーバイザーとなっています。新機能は各リリースで追加され、当社が開発したKVMライブバックアップ機能により、あらゆるストレージタイプでのスナップショットバックアップ作成が可能になりました。

バージョン4.0における最も顕著な変更点は、OpenVZからLXCへの移行でした。コンテナは現在深く統合され、仮想マシンと同様のストレージおよびネットワーク機能を利用できます。同時に、高可用性（HA）設定の構成と管理を簡素化する、使いやすい高可用性（HA）マネージャーを導入しました。

Proxmox VE 5の開発過程では、[非同期ストレージレプリケーション](#)やACME/Let's Encryptを利用した自動証明書管理をはじめ、数多くの新機能が導入されました。

ソフトウェア定義ネットワーク（SDN）スタックはコミュニティとの協力により開発されました。バージョン6.2では実験的機能としてWebインターフェースに統合され、複雑なネットワーク構成の管理を簡素化しました。バージョン8.1以降、SDN統合は完全サポートされデフォルトでインストールされます。

2020年には、Rustプログラミング言語で記述された新プロジェクト「[Proxmox Backup Server](#)」がリリースされました。Proxmox Backup ServerはProxmox VEと深く統合され、増分バックアップや重複排除などの実装によりバックアップ機能を大幅に強化します。

2022年には新たなツール「[Proxmox Offline Mirror](#)」がリリースされ、パブリックインターネットに接続できないシステムでもサブスクリプションが可能になりました。

ウェブインターフェース向けに要望の多かったダークテーマが2023年に導入されました。同年後半にはバージョン8.0でCephエンタープライズリポジトリへのアクセスが統合されました。これにより、最も安定したCephリポジトリへのアクセスがProxmox VEのあらゆるサブスクリプションで利用可能になりました。

公式ISOインストーラーの自動化・無人インストール機能がバージョン8.2で導入され、Proxmox VEの大規模展開が大幅に簡素化されました。

同様にバージョン8.2で導入されたインポートウィザードにより、ユーザーはVMware ESXi<sup>1</sup>などの他のハイパーバイザーからゲストを直接、容易かつ効率的に移行できます。さらに、ウェブインターフェースからファイルベースストレージ内のOpen Virtualization Format（OVF/OVA）アーカイブを直接インポートできるようになりました。

<sup>1</sup> Migrate to Proxmox VE [https://pve.proxmox.com/wiki/Migrate\\_to\\_Proxmox\\_VE](https://pve.proxmox.com/wiki/Migrate_to_Proxmox_VE)

## 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善は常に歓迎します。貢献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点を見つけた場合は、修正を提案するために[Proxmoxバグトラッカー](#)でバグを報告してください。

新しいコンテンツを提案したい場合は、以下のいずれかの方法を選択してください：

- Wiki: 特定の設定方法、ハウツーガイド、チュートリアルについては、Wikiへの貢献が適切な選択肢です。
- 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コンテンツについては、リファレンスドキュメントへの貢献をご提案ください。これには、Proxmox VE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`形式で記述されています。  
ドキュメントを編集するには、`git://git.proxmox.com/git/pve-docs` でgitリポジトリをクローンし `README.adoc` ファイルの指示に従ってください。

---

### 注記

Proxmox VEのコードベースでの作業にご興味がある場合は、[開発者向けドキュメント](#)のWiki記事が開始の手引きとなります。

---

## 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、コミュニティの貢献により、他の言語への翻訳も利用可能です。新しい言語の追加、最新機能の翻訳、不完全または不一致な翻訳の改善など、あらゆる支援を歓迎します。

翻訳ファイルの管理には `gettext` を使用しています。Poeditなどのツールは翻訳ファイルを編集するための使いやすいユーザーインターフェースを提供しますが、お使いのエディタで編集することも可能です。翻訳にプログラミング知識は必要ありません。

### 1.13.1 gitを使用した翻訳

言語ファイルはgitリポジトリとして提供されています。gitに精通している場合は、[開発者向けドキュメント](#)に従って貢献してください。

以下の手順で新規翻訳を作成できます（&lt;LANG&gt;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-&lt;LANG&gt;.po
```

または、お好みのエディタで既存の翻訳を編集することもできます：

```
# poedit &lt;LANG&gt;.po
```

## 1.13.2 gitなしで翻訳する

gitに詳しくなくても、Proxmox VEの翻訳にご協力いただけます。まず、[こちらの言語ファイル](#)をダウンロードしてください。改善したい言語を選択し、その言語ファイルの「raw」リンクを右クリックして「リンク先を保存」を選択します。ファイルに変更を加えた後、最終的な翻訳を署名済みの貢献者ライセンス契約書と共に直接お送りください。

office(at)proxmox.com 宛に、署名済みの[貢献者ライセンス契約書](#)と共に直接お送りください。

## 1.13.3 翻訳のテスト

翻訳をProxmox VEで使用するには、まず.poファイルを.jsファイルに変換する必要があります。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変換できます：

```
# ./po2js.pl -t pve xx.po &gt;pve-lang-xx.js
```

生成されたファイル pve-lang-xx.js を、Proxmox サーバー上の /usr/share/pve-i18n ディレクトリにコピーすることでテストできます。

あるいは、リポジトリのルートディレクトリから次のコマンドを実行してdeb/パッケージを作成することもできます：

```
# make deb
```



### 重要

いずれの方法でも動作させるには、システムに以下のPerlパッケージが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

完成した翻訳ファイル（.poファイル）は、署名済みの貢献者ライセンス契約書と共に、[office\(at\)proxmox.com](mailto:office(at)proxmox.com) 宛てにProxmoxチームへ送信できます。あるいは、開発経験がある場合は、パッチとしてProxmox VE開発メーリングリストへ送信することも可能です。詳細は[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を基盤としています。そのため、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パッケージがすべて含まれています。

### ヒント

Proxmox VE リリースと Debian リリースの対応関係については、[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分割、基本システム設定（タイムゾーン、言語、ネットワークなど）の適用、必要なパッケージのインストールを実行します。このプロセスは数分以内に完了します。新規ユーザーおよび既存ユーザーには、提供されたISOを使用したインストールが推奨されます。

あるいは、既存のDebianシステム上にProxmox VEをインストールすることも可能です。このオプションは、Proxmox VEに関する詳細な知識が必要なため、上級ユーザーのみに推奨されます。

## 2.1 システム要件

本番環境で Proxmox VE を運用する際は、高品質なサーバーハードウェアの使用を推奨します。ホスト障害の影響をさらに軽減するには、高可用性（HA）仮想マシンおよびコンテナを備えたクラスター環境で Proxmox VE を実行できます。

Proxmox VE は、ローカルストレージ (DAS)、SAN、NAS、Ceph RBD などの分散ストレージを使用できます。詳細については、[ストレージの章](#)を参照してください。

### 2.1.1 評価用最小要件

これらの最小要件は評価目的のみであり、本番環境での使用は推奨されません。

- CPU: 64ビット (Intel 64 または AMD64)
- KVM完全仮想化サポートにはIntel VT/AMD-V対応CPU/マザーボードが必要
- RAM: 1 GB RAM、ゲスト用に追加のRAMが必要
- 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 64 または AMD64、Intel VT/AMD-V CPU フラグ付き。
- 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のメモリ。Ceph および ZFS の場合は、追加のメモリが必要です。使用ストレージ 1 TBあたり約 1 GB のメモリが必要です。
- 高速かつ冗長化されたストレージ。SSDを使用すると最良の結果が得られます。
- OSストレージ：バッテリー保護付き書き込みキャッシュ（BBU）を備えたハードウェアRAID、またはZFSを使用した非RAID（ZIL用にオプションのSSD）。
- VMストレージ：
  - ローカルストレージには、バッテリーバックアップ付き書き込みキャッシュ（BBU）を備えたハードウェアRAID、またはZFSおよびCeph用の非RAIDを使用してください。ZFSもCephもハードウェアRAIDコントローラーとは互換性がありません。
  - 共有ストレージおよび分散ストレージは可能です。
  - 良好的なパフォーマンスのためには、電源喪失保護（PLP）機能付きSSDの使用を推奨します。一般消費者向けSSDの使用は推奨されません。
- 冗長化された（マルチ）ギガビットNICを必須とし、採用するストレージ技術とクラスター構成に応じて追加NICを配置すること。
- PCI(e)バススルーには、CPUがVT-d/AMD-dフラグをサポートしている必要があります。

## 2.1.3 簡易性能概要

インストール済みのProxmox VEシステムにおけるCPUおよびハードディスクのパフォーマンス概要を取得するには、同梱のpveperfツールを実行します。

### 注意

これはあくまで簡易的な一般的なベンチマークです。特にシステムのI/O性能に関しては、より詳細なテストの実施をお勧めします。

## 2.1.4 Webインターフェースへのアクセスに対応しているWebブラウザ

Webベースのユーザーインターフェー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推奨します：

- Firefox、今年度のリリース、または最新の拡張サポートリリース
- Chrome、今年リリースされたバージョン
- Microsoftが現在サポートしているバージョンのEdge
- Safari、今年リリースされたバージョン

モバイルデバイスからアクセスした場合、Proxmox VEは軽量な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イメージは以下からダウンロードしてください：<https://www.proxmox.com/en/downloads/proxmox-virtual-environment/-iso>

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

- CDまたはDVDに書き込む準備が整った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）にコピー可能な生のセクター（IMG）イメージファイル。

Proxmox VEのインストールにはUSBフラッシュドライブの使用が推奨されます。これはより高速な方法だからです。

### 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準備する

フラッシュドライブには、少なくとも1GBの空き容量が必要です。

---

注意

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、重要なデータが含まれていないことを確認してください。

---

### 2.2.2 GNU/Linuxでの手順

Unix系オペレーティングシステムでは、ddコマンドを使用して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確認してください（下記参照）。その後、ddコマンドを実行します。

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

---

注意

/dev/XYZを正しいデバイス名に置き換え、入力ファイル名(/)のパスを適宜変更してください。

---



注意

非常に注意し、間違ったディスクを上書きしないでください！

---

### 正しいUSBデバイスの名前を確認する

USBフラッシュドライブの名前を確認する方法は2つあります。1つ目は、フラッシュドライブを接続する前後のdmesgコマンド出力の最終行を比較する方法です。2つ目は、lsblkコマンドの出力を比較する方法です。ターミナルを開き、以下を実行してください：

```
# lsblk
```

次にUSBフラッシュドライブを接続し、再度コマンドを実行します：

```
# lsblk
```

新しいデバイスが表示されます。これが使用したいデバイスです。万全を期すために、報告されたサイズがUSBフラッシュドライブと一致するか確認してください。

### 2.2.3 macOS の手順

ターミナルを開きます（Spotlightで「ターミナル」を検索）。

hdiutilのconvertオプションを使用して.isoファイルを.dmg形式に変換します。例：

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

---

#### ヒント

macOSは出力ファイル名に自動的に.dmgを追加する傾向があります。

---

現在のデバイス一覧を取得するには、次のコマンドを実行します：

```
# diskutil list
```

USBフラッシュドライブを挿入し、再度このコマンドを実行して、どのデバイスノードが割り当てられたかを確認します（例: /dev/diskX）。

```
# diskutil list
# diskutil unmountDisk /dev/diskX
```

---

#### 注記

Xを最後のコマンドで確認したディスク番号に置き換えてください。

---

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

---

#### 注

最後のコマンドでは diskX ではなく rdiskX を使用します。これにより書き込み速度が向上します。

---

## 2.2.4 Windowsでの手順

### Etcherの使用

Etcherはそのまま使用できます。<https://etcher.io>からEtcherをダウンロードしてください。ISOファイルとUSBフラッシュドライブの選択プロセスをガイドします。

### Rufusの使用

Rufusはより軽量な代替ツールですが、動作させるにはDDモードを使用する必要があります。<https://rufus.ie/>からRufusをダウンロードしてください。インストール版またはポータブル版を使用します。書き込み先ドライブとProxmox VE ISOファイルを選択します。



#### 重要

開始後、異なるバージョンのGRUBをダウンロードするかどうか尋ねるダイアログでは「いいえ」をクリックしてください。次のダイアログでDDモードを選択します。

## 2.3 Proxmox VEインストーラーの使用

インストーラーISOイメージには以下が含まれます：

- 完全なオペレーティングシステム (Debian Linux、64ビット)
- Proxmox VE インストーラーは、ローカルディスクを ext4、XFS、BTRFS（技術プレビュー）、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および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、クラスター、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
- Webベースの管理インターフェース

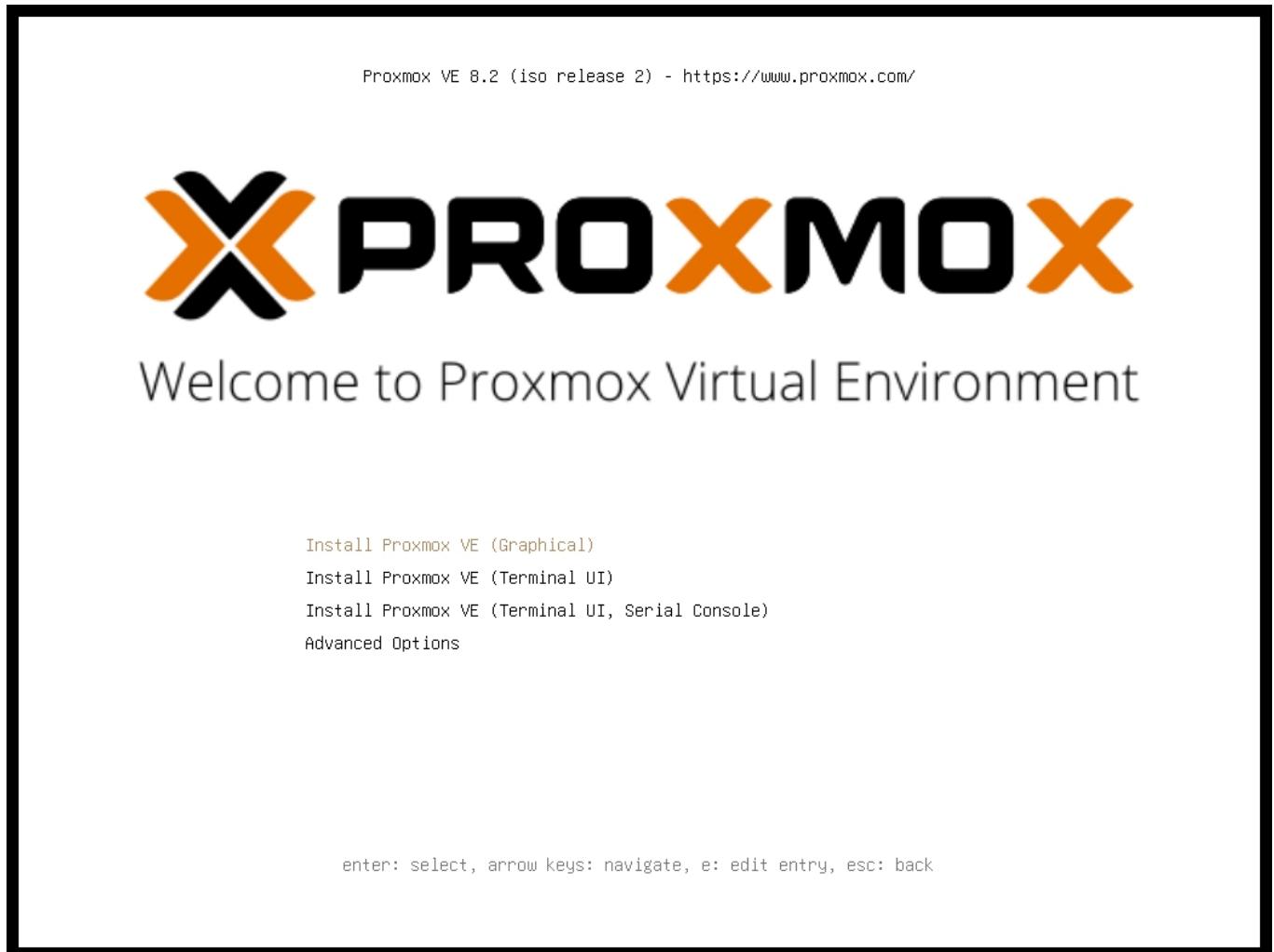
#### 注

選択したドライブ上の既存データはすべてインストールプロセス中に削除されます。インストーラーは他のオペレーティングシステム用のブートメニューEntriesを追加しません。

準備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。

#### ヒント

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（例：USB）からの起動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 以前のインストーラーを起動する場合は、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：USBから起動）を選択すると、Proxmox VEメニューが表示され、以下のいずれかのオプションを選択できます：

#### Proxmox VE のインストール（グラフィカル）

通常のインストールを開始します。

#### ヒント

インストールウィザードはキーボードのみでも操作可能です。各ボタンの下線付き文字とALTキーを同時に押すことでボタンをクリックできます。例：ALT + N で「次へ」ボタンを押します。

#### Proxmox VE のインストール（ターミナル UI）

ターミナルモードのインストールウィザードを開始します。グラフィカルインストーラとほぼ同じインストール体験を提供しますが、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開始し、さらにLinuxカーネルを設定してマシンの（最初の）シリアルポートを入出力に使用します。これは、マシンが完全にヘッダレスでシリアルコンソールのみが利用可能な場合に使用できます。



両モードとも、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ており、10年以上にわたるバグ修正の恩恵を受け、機能の互換性を確保しています。

#### ヒント

グラフィカルインストーラがドライバの問題などで正常に動作しない場合、ターミナルUIオプションを使用できます。[nomodeset](#)カーネルパラメータの追加も参照してください。

#### 詳細オプション: 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。インストール工程の複数箇所でコンソールが開きます。これにより、問題発生時の状況デバッグが可能になります。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利用可能なライブシステムを起動できます。例えば、[劣化したZFS rpoolの修復](#)や、既存のProxmox VE環境のブートローダー修正などに使用できます。

#### 詳細オプション: 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様ですが、代わりにターミナルベースのインストーラを実行する準備を行います。

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

ターミナルベースのデバッグモードと同様ですが、さらにLinuxカーネルを設定し、マシンの（最初の）シリアルポートを出力入出力に使用します。

#### 詳細オプション: Proxmox VE のインストール (自動化)

ISOが自動インストール用に適切に準備されていなくても、インストーラを無人モードで起動します。このオプションはハードウェアの詳細情報を収集したり、自動インストール設定のデバッグに役立つ場合があります。詳細は[無人インストール](#)を参照してください。

### 詳細オプション: レスキューーブート

このオプションを使用すると、既存のインストール環境を起動できます。接続された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環境が見つかった場合、ISO内のLinuxカーネルを使用してそのディスクから直接起動します。ブートローダー（GRUB/systemd-boot）に問題がある場合や、BIOS/UEFIがディスクのブートブロックを読み込めない場合に有用です。

### 詳細オプション: メモリテスト (memtest86+)

memtest86+を実行します。メモリが正常に機能しエラーがないかを確認するのに有用です。このオプションを実行するには、UEFIファームウェア設定ユーティリティでセキュアブートを無効にする必要があります。

通常は「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（エンドユーザー使用許諾契約書）を読むことです。その後、インストール先のハードディスクを選択できます。



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

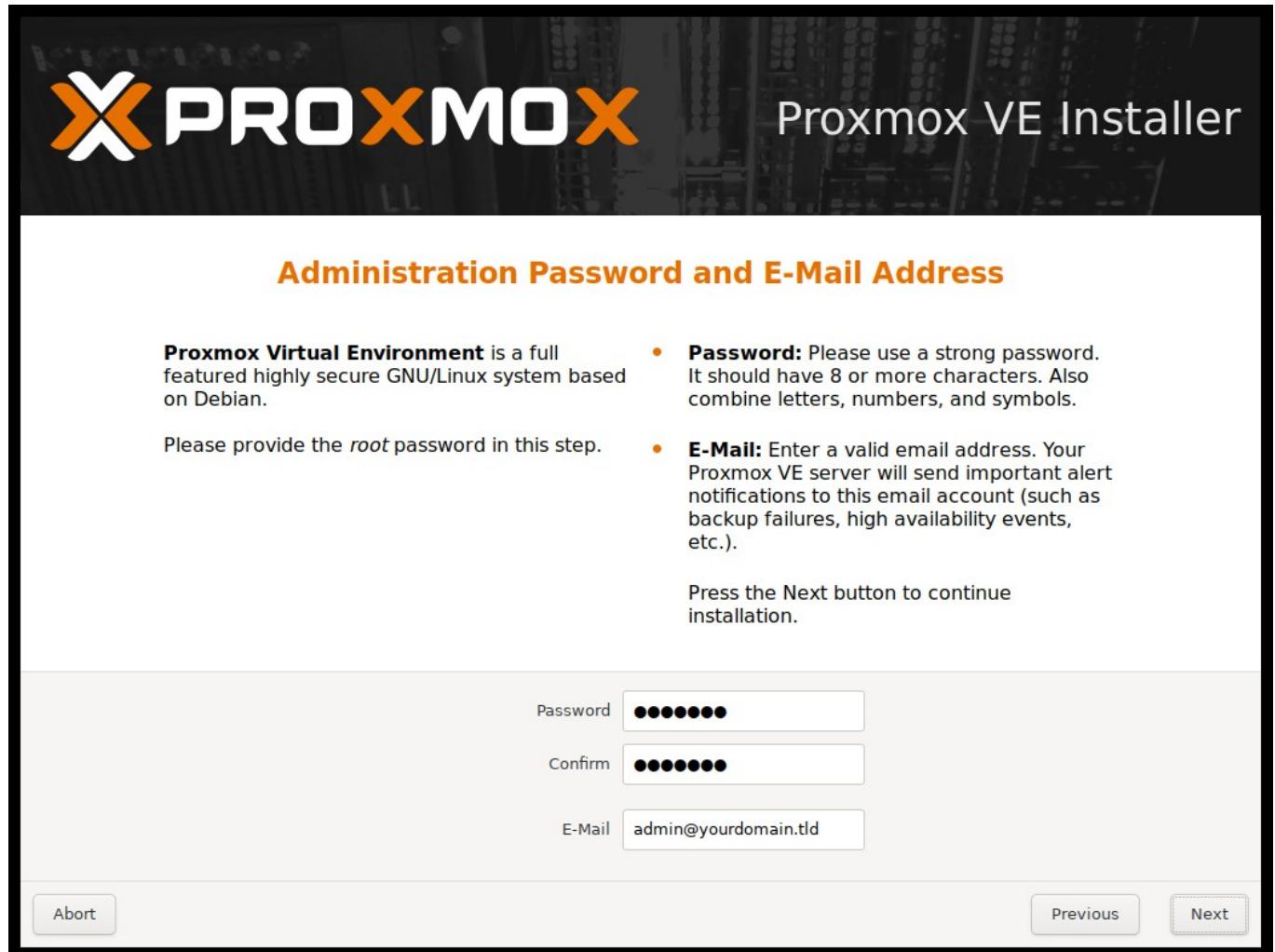
Proxmox VEはZFS上にもインストール可能です。ZFSは複数のソフトウェアRAIDレベルを提供するため、ハードウェアRAIDコントローラを持たないシステム向けの選択肢となります。対象ディスクはオプションダイアログで選択する必要があります。より詳細なZFS固有の設定は「[詳細オプション](#)」で変更可能です。

**警告**

ハードウェアRAID上のZFSはサポート対象外であり、データ損失の原因となる可能性があります。

The screenshot shows the Proxmox VE Installer interface. At the top left is the Proxmox logo. To its right, the text "Proxmox VE Installer" is displayed. Below this, the title "Location and Time Zone selection" is centered in orange. To the left of the title, there is a text block: "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." Below this text is a note: "Press the Next button to continue installation." To the right of the title, there is a bulleted list of three items: "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.", and "Keyboard Layout: Choose your keyboard layout." At the bottom of the screen, there are three input fields: "Country" set to "Austria", "Time zone" set to "Europe/Vienna", and "Keyboard Layout" set to "German". At the very bottom are three buttons: "Abort", "Previous", and "Next".

次のページでは、お住まいの地域、タイムゾーン、キーボードレイアウトなどの基本設定オプションを尋ねます。地域情報は、更新速度を向上させるために近隣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。インストーラーは通常これらの設定を自動検出できるため、自動検出が失敗した場合や、お住まい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など、ごく稀な状況でのみ変更が必要です。

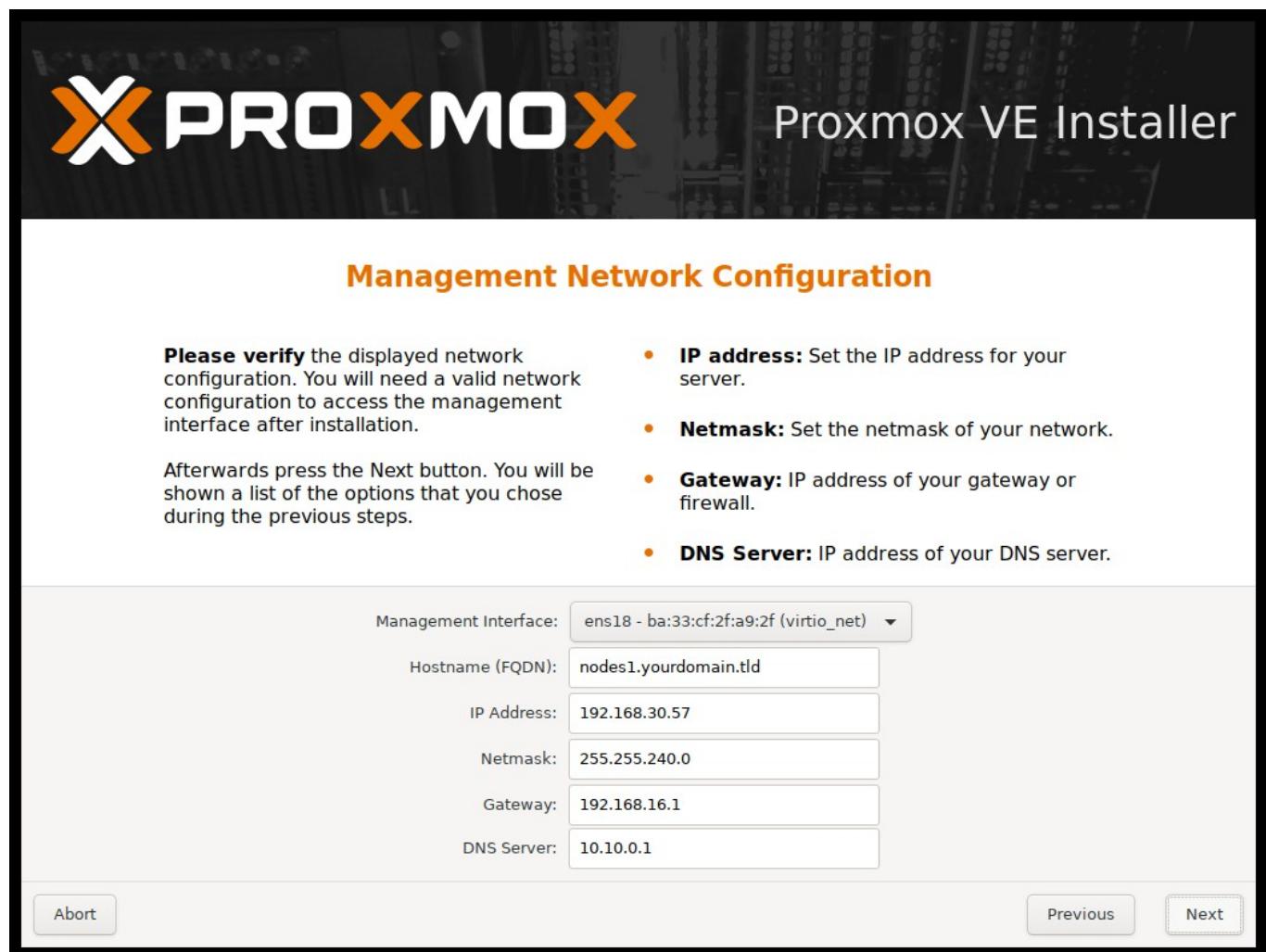


次に、スーパーユーザー (root) のパスワードとメールアドレスを指定する必要があります。パスワードは最低8文字以上で構成してください。より強力なパスワードの使用を強く推奨します。ガイドラインは以下の通りです：

- パスワードの長さは最低12文字以上にしてください。
- 小文字と大文字のアルファベット、数字、記号を含めてください。
- 文字の繰り返し、キーボードのパターン、一般的な辞書語、文字や数字の連続、ユーザー名、親族やペットの名前、恋愛関係（現在または過去）、個人を特定できる情報（例：ID番号、先祖の名前や日付）は避けてください。

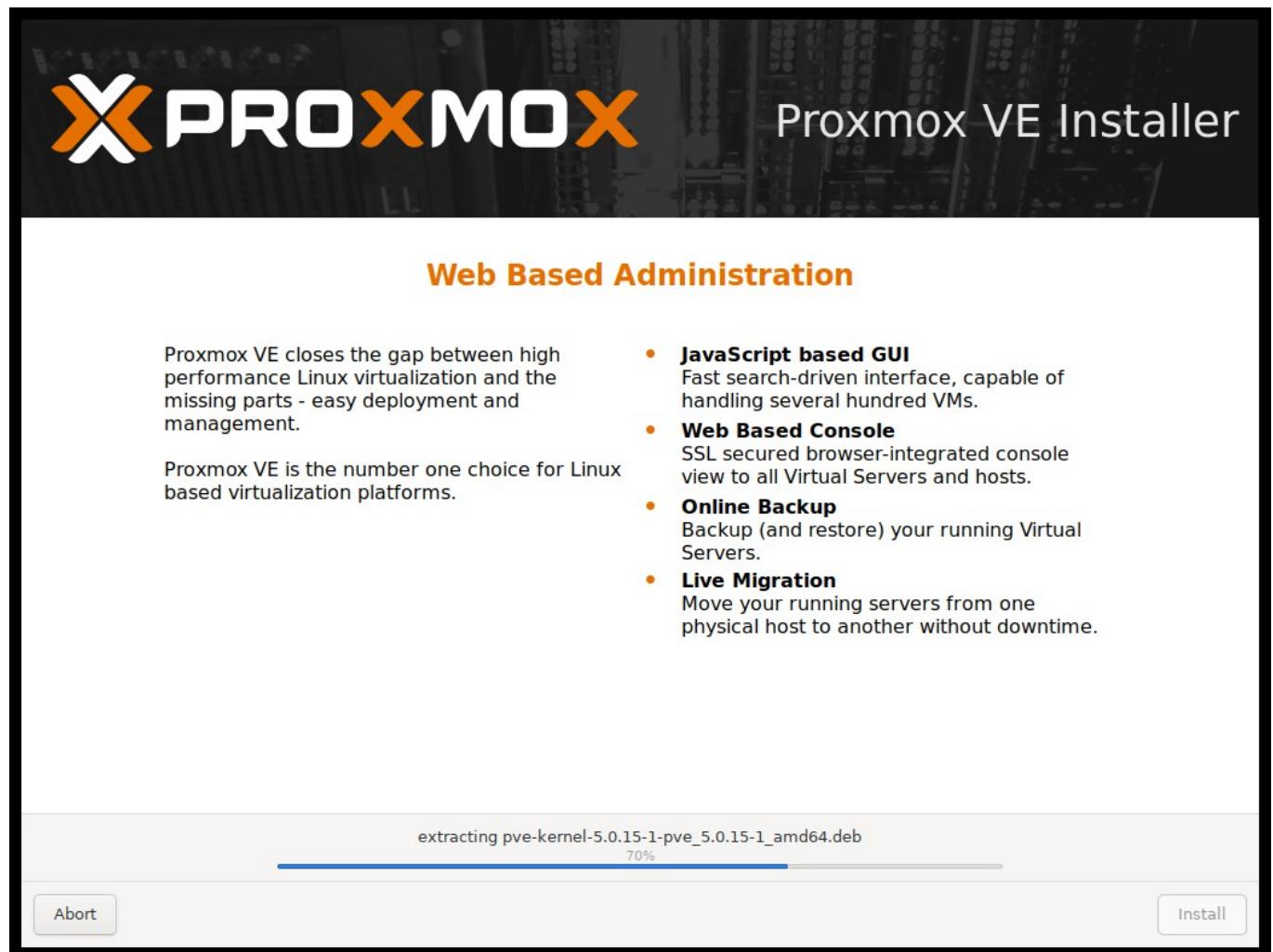
メールアドレスはシステム管理者への通知送信に使用されます。例：

- 利用可能なパッケージ更新に関する情報。
- 定期的なcronジョブからのエラーメッセージ。



これらの通知メールはすべて、指定されたメールアドレスに送信されます。

最後のステップはネットワーク設定です。UP状態のネットワークインターフェースは、ドロップダウンメニュー内で名前前に丸印が表示されます。インストール時にはIPv4またはIPv6アドレスのいずれかを指定可能ですが、両方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、インストール後に追加のIPアドレスを設定してください。



次のステップでは、これまで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、変更が必要な場合は「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、インストーラがディスクのフォーマットを開始し、パッケージをターゲットディスクにコピーします。このステップが完了するまでお待ちください。その後、インストールメディアを取り外し、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。所要時間は主に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設定が完了したら、サーバーを再起動できます。デフォルトでは数秒後に自動的に再起動されます。

#### インストール失敗

インストールが失敗した場合は、セカンドTTY (**CTRL + ALT + F2**) で具体的なエラーを確認し、システムが[最低要件](#)を満たしていることを確認してください。

それでもインストールが正常に動作しない場合は、「[ヘルプの入手方法](#)」の章を参照してください。

### 2.3.1 インストール後の管理インターフェースへのアクセス

The screenshot shows the 'Proxmox VE Login' window. It contains fields for 'User name' (set to 'root'), 'Password' (redacted), 'Realm' (set to 'Linux PAM standard authentication'), and 'Language' (set to 'English'). There is also a checkbox for 'Save User name' and a 'Login' button.

システムのインストールと再起動が正常に完了したら、Proxmox VE Web 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. ブラウザで、インストール中に指定された IP アドレスとポート 8006 を指定してください。例：<https://youripad>
2. `root`（レルム 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してください。
3. エンタープライズリポジトリへのアクセス権を得るには、サブスクリプションキーをアップロードしてください。そうしない場合、セキュリティ修正、バグ修正、新機能のアップデートを取得するには、公開されているテストが不十分なパッケージリポジトリのいずれかを設定する必要があります。
4. IP 設定とホスト名を確認してください。
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 設定オプション

インストーラーは `ext4` または `xfs` を使用する場合、`pve` というボリュームグループ (VG) と、`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以下を使用します：

#### **hdsize**

使用されるハードディスクの総容量を定義します。これにより、ハードディスク上に空き領域を予約し、追加のパーティション分割（例：同じハードディスク上に追加の物理ボリューム (PV) とボリュームグループ (VG) を作成し、LVMストレージとして使用するため）を行うことが可能です。

#### **swapsize**

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済みメモリのサイズで、最小4GB、最大8GBです。結果の値はハードドライブのサイズの8分の1 (`hdsize / 8`) を超えることはできません。

---

#### 注

0に設定した場合、スワップボリュームは作成されません。

---

#### **maxroot**

ルートボリュームの最大サイズを定義します。ルートボリュームにはオペレーティングシステムが格納されます。利用可能なストレージが48 GiBを超える場合、デフォルトはハードドライブ容量の4分の1 (`hdsize / 4`) で、最大96 GiBです。利用可能なストレージが48 GiB未満の場合、ルートボリュームサイズはハードドライブのサイズの少なくとも半分 (`hdsize / 2`) になります。

#### **maxvz**

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次の通りです：

`datasize= hdsize - rootsize - swapsize - minfree`

ここで、`datasize` は `maxvz` を超えることはできません。

---

**注**

LVM thinの場合、datasizeが4GBを超える場合にのみデータプールが作成されます。

---

**注**

0に設定した場合、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調整されます。

---

**minfree**

LVMボリュームグループpve内に確保すべき空き領域の量を定義します。利用可能なストレージが128GBを超える場合、デフォルトは16GBです。それ以外の場合hdszie/8が使用されます。  
128GB以上のストレージが利用可能な場合、デフォルトは16GBです。それ以外の場合はhdszie/8が使用されます。

---

**注**

LVM はスナップショット作成のために VG 内の空き領域を必要とします (lvmthin スナップショットは例外)。

---

### 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。スワップ領域は作成されませんが、インストールディスク上の未パーティション領域をスワップ用に予約できます。インストール後にスワップ用zvolを作成することも可能ですが、問題を引き起こす可能性があります ([ZFSスワップに関する注意事項](#)を参照)。

**ashift**

作成するプールに対するashift値を定義します。ashiftは少なくとも基盤ディスクのセクタサイズ (ashiftの2乗がセクタサイズとなる) 以上、あるいはプールに追加される可能性のあるディスク（例：故障ディスクの交換用）のサイズに設定する必要があります。  
サイズ (ashiftの2乗がセクタサイズ) 以上、またはプールに追加される可能性のあるディスク（例：故障ディスクの交換用）のセクタサイズ以上である必要があります。

**compress**

rpool に対して圧縮を有効にするかどうかを定義します。

**checksum**

rpool に使用するチェックサムアルゴリズムを定義します。

**copies**

rpool のコピー数を定義します。その意味や、これがディスクレベルの冗長性を置き換えない理由については、[zfs \(8\)](#) マニュアルページを参照してください。

**ARC max size**

ARCが成長できる最大サイズを定義し、それによってZFSが使用するメモリ量を制限します。詳細は、[ZFSメモリ使用量の制限方法](#)に関するセクションも参照してください。

**hdszie**

使用するハードディスクの総容量を定義します。これは、ハードディスク上の空き領域を保存し、さらなるパーティション分割（例：スワップパーティ

---

ションの作成）に活用するのに有用です。`hdsize`はブート可能なディスクに対してのみ有効です。つまり、最初のディスク、またはRAID0、RAID1、RAID10のミラー、およびRAID-Z[123]の全ディスクにのみ適用されます。

## 2.3.4 高度な BTRFS 設定オプション

BTRFSを使用する場合、スワップ領域は自動生成されませんが、インストールディスク上で未パーティション領域をスワップ用に確保できます。別パーティションの作成、BTRFSサブボリュームの作成、またはbtrfsファイルシステムのmkswapfileコマンドを用いたスワップファイルの作成のいずれかを選択可能です。

### compress

BTRFSサブボリュームの圧縮有効化を定義します。サポートされる圧縮アルゴリズムは次の通りです：*on* (*zlib*と同等)、*zlib*、*lzo*、*zstd*。デフォルトは無効です。

### hdszie

使用するハードディスクの総容量を定義します。これは、ハードディスク上の空き領域を保存し、さらなるパーティション分割（例：スワップパーティションの作成）に活用するのに便利です。

## 2.3.5 ZFS パフォーマンスのヒント

ZFSは大量のメモリで最高のパフォーマンスを発揮します。ZFSを使用する場合は、十分なRAMを確保してください。目安として、RAWディスク容量1TBあたり4GB+1GBのRAMが必要です。

ZFSは専用ドライブを書き込みキャッシュとして使用でき、これをZFS Intent Log（ZFS意図ログ、ZIL）と呼びます。高速ドライブ（SSD）を使用してください

```
# zpool add <pool-name> log </dev/path_to_fast_ssd>;  
。インストール後に以下のコマンドで追加できます：
```

## 2.3.6 nomodeset カーネルパラメータの追加

非常に古いハードウェアや新しいハードウェアでは、グラフィックドライバが原因で問題が発生する可能性があります。起動中にインストールが停止した場合、nomodesetパラメータを追加してみてください。これにより、Linuxカーネルがグラフィックドライバをロードせず、BIOS/UEFIが提供するフレームバッファの使用を強制します。

Proxmox VEブートローダーメニューで、[Install Proxmox VE (Terminal UI)]を選択し、eキーを押してエントリを編集します。矢印キーでlinuxで始まる行に移動し、カーソルをその行の末尾に移動させ、既存の最後のパラメータからスペースを空けてnomodesetパラメータを追加します。

次にCtrl-XまたはF10を押して設定を起動します。

## 2.4 無人インストール

自動インストール方式により、Proxmox VEを無人状態でインストールできます。これにより、ベアメタル環境で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などの自動化ツールを使用してインストール環境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指定する必要があります。このファイルでは、フィルタールールを使用して使用するディスクやネットワークカードを決定できます。

自動インストールを使用するには、まずアンサーファイルを取得するソースを選択し、その選択に基づいてインストールISOを準備する必要があります。

ISOを作成すると、初期ブートメニューに「Automated Installation」という新しいブートエントリが表示されます。

が表示され、10秒のタイムアウト後に自動的に選択されます。

無人インストールに関する詳細と情報は、[当社のWikiをご覧ください](#)。

## 2.5 DebianへのProxmox VEのインストール

Proxmox VEはDebianパッケージのセットとして提供され、標準的なDebianインストール環境上にインストールできます。リポジトリを設定後、以下のコマンド

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

を実行してください：

既存のDebianインストール環境への上書きインストールは一見簡単に見えますが、これはベースシステムが正しくインストールされていること、およびローカルストレージの設定方法と使用方法を理解していることを前提としています。またネットワーク設定も手動で行う必要があります。

一般的に、これは特にLVMやZFSを使用する場合、単純ではありません。詳細な手順は[Wiki](#)で確認

できます。

## 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と運用に関するProxmox VE固有の事項を説明します。

Proxmox VEはDebian GNU/Linuxをベースとしており、Proxmox VE関連パッケージを提供する追加リポジトリを備えています。これにより、セキュリティ更新やバグ修正を含むDebianパッケージの全範囲が利用可能です。Proxmox VEはUbuntuカーネルをベースとした独自の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化されており、ZFSや複数の追加ハードウェアドライバーが含まれています。

以下のセクションに含まれないその他のトピックについては、Debianのドキュメントを参照してください。[Debian管理者ハンドブック](#)はオンラインで入手可能であり、Debianオペレーティングシステムに関する包括的な入門を提供しています（[\[Hertzog13\]](#)を参照）。

## 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は毎日自動的にパッケージの更新を確認します。root@pamユーザーには利用可能な更新についてメールで通知されます。GUIでは、選択した更新の詳細を確認するために「変更履歴」ボタンを使用できます。

### 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のインストールに使用できるだけでなく、新しい更新入手するためにも重要です。

---

#### 注意

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およびProxmoxリポジトリが必要です。

---

APT リポジトリは、レガシーな單一行形式ではファイル /etc/apt/sources.list に、現代的な deb822 マルチライン形式では /etc/apt/sources.list.d/ に配置された sources ファイルに定義されます。詳細はリポジトリ形式を参照してください。

## リポジトリ管理

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

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。ノード概要パネルには高レベルのステータス概要が表示され、別個の「リポジトリ」パネルには詳細なステータスと設定済み全リポジトリのリストが表示されます。

リポジトリの有効化や無効化といった基本的な管理操作もサポートされています。

リポジトリから利用可能なパッケージは、`apt update` を実行することで取得されます。更新は `apt` を直接使用するか、GUI（ノード →→ 更新）経由でインストールできます。

## リポジトリ形式

パッケージリポジトリは、ソースリスト `/etc/apt/sources.list` および `/etc/apt/sources.list.d/` に含まれるファイルで設定できます。

サポートされている形式は2つあります：

### 單一行形式

單一行形式の `sources.list` ファイルでは、各行がパッケージリポジトリを定義します。空行は無視されます。行内のどこかに # 文字があると、その行の残りの部分がコメントとして扱われます。これはレガシー形式です。Debian 13 Trixie 以降、`apt` はこの形式の使用に対して警告を出します。`apt modernize-sources` コマンドを使用すれば、ほとんどのリポジトリを自動的に移行できます。

### deb822

複数行形式の `repo.sources` ファイルでは、各エントリはキーと値のペアが複数行にわたって記述されます。ファイルには、各グループを空白行で区切ることで複数のエントリを含めることができます。これが現代的な形式です。

## 利用可能なリポジトリ

Proxmox VEは、ベースとなるDebianリポジトリに加えて、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 Debian ベースリポジトリ

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

```
Types: deb deb-src
URIs: http://deb.debian.org/debian/ Suites: trixie
       trixie-updates Components: main non-free-firmware
      署名者: /usr/share/keyrings/debian-archive-keyring.gpg

タイプ: deb deb-src
URIs: http://security.debian.org/debian-security/Suites: trixie-security
       Components: main non-free-firmware
       署名者: /usr/share/keyrings/debian-archive-keyring.gpg
```

### 3.1.3 Proxmox VE Enterprise リポジトリ

これは推奨リポジトリであり、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を含み、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化されています:

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

```
Types: deb
URIs: https://enterprise.proxmox.com/debian/pve Suites: trixie
       Components: pve-enterprise
       署名者: /usr/share/keyrings/proxmox-archive-keyring.gpg
```

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。詳細については、<https://proxmox.com/en/proxmox-virtual-environment/pricing> でさまざまなサポートレベルをご確認ください。

---

#### 注意

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化できます。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止します。その場合は pve-no-subscription リポジトリを設定してください。

---

### 3.1.4 Proxmox VE サブスクリプション不要リポジトリ

名称が示す通り、このリポジトリへのアクセスにはサブスクリプションキーが不要です。テスト環境や非本番環境での使用に適しています。これらのパッケージは必ずしも十分にテスト・検証されていないため、本番サーバーでの使用は推奨されません。

このリポジトリは `/etc/apt/sources.list.d/proxmox.sources` に設定することを推奨します。

ファイル `/etc/apt/sources.list.d/proxmox.sources`

```
Types: deb
URIs: http://download.proxmox.com/debian/pveSuites: trixie
Components: pve-no-subscription
Signed-By: /usr/share/keyrings/proxmox-archive-keyring.gpg
```

---

#### 注意

Proxmox VE Proxmox リポジトリに加えて、常にベースの Debian リポジトリが必要であることを忘れないでください

---

### 3.1.5 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されます。設定するには、ファイル `/etc/apt/sources.list.d/proxmox.sources` に以下のスタンザを追加してください:

ファイル `/etc/apt/sources.list.d/proxmox.sources`

```
Types: deb
URIs: http://download.proxmox.com/debian/pveSuites: trixie
Components: pve-test
Signed-By: /usr/share/keyrings/proxmox-archive-keyring.gpg
```



#### 警告

`pve-test` リポジトリは（その名前が示す通り）新機能やバグ修正のテストにのみ使用してください。

---

### 3.1.6 Ceph リポジトリ

Ceph関連パッケージは、事前設定済みのCephエンタープライズリポジトリにより最新の状態に保たれます。プリインストールされたパッケージにより、外部Cephクラスターへの接続や、そのRBDまたはCephFSプールをストレージとして追加することができます。また、Proxmox VEクラスター上でCephを実行することで、ハイパーコンバージドインフラストラクチャ (HCI) を構築することもできます。

本章の情報は以下の場合に役立ちます：

### HCI構築のためのCephインストール

以下で説明するリポジトリと最新のCephリリースを決定し、[ウェブベースのウィザード](#)または[CLIツール](#)で選択します。

#### HCIを既に運用中で、Cephのメジャーリリースをアップグレードしたい場合

関連する[Cephアップグレードガイド](#)に従ってください。

#### 既にHCIを稼働中で、次のProxmox VEメジャーリリースへアップグレードしたい場合

HCI環境では、各Proxmox VEメジャーリリースに対応するCephの最小メジャーリリースが必要です。関連する[Proxmox VEアップグレードガイド](#)に従ってください。

#### HCIを運用しておらず、外部Cephクラスターを利用している場合

Ceph接続用の最新パッケージをインストールするには、利用可能なシステム更新を適用し、下記リストからリポジトリとCephリリースを選択、[リポジトリパネル](#)経由でノードに追加、新たに利用可能なシステム更新を適用、`ceph --version`の実行で結果を確認後、旧Cephリポジトリを無効化してください。

### Proxmox VE 9で利用可能な Ceph リリース

お使いの Proxmox VE リリースに関する最新版の管理ガイドをお読みになるには、すべてのシステムアップデートがインストールされていること、およびこのページが再読み込みされていることを確認してください。

	推定サポート終了時期	enterprise	no-subscription	n test
ceph-squid	2026-09 (v19.2)	推奨	利用可能	利用可能

### Proxmox VE 9向け Ceph リポジトリ

以下の `ceph.sources` ファイルの内容は参考としてご利用ください（Proxmox VE 9 以前では `ceph.list` ファイルが使用されていました）。変更を行う場合は、このサブチャプターの冒頭で説明されている、ご自身の状況に該当するケースに従ってください。Web UI でリポジトリを無効化し、かつリストから削除したい場合は、該当するエントリをファイルから手動で削除できます。

#### enterprise

本リポジトリは本番環境での使用が推奨され、最も安定したパッケージバージョンを含みます。Proxmox VEノードが有効なサブスクリプション（いずれのレベルでも可）を有する場合にアクセス可能です。詳細および含まれるカスタマーサポートレベルについては以下を参照：

<https://proxmox.com/en/proxmox-virtual-environment/pricing>

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

```
Types: deb
URIs: https://enterprise.proxmox.com/debian/ceph-squidSuites: trixie
コンポーネント: enterprise
署名者: /usr/share/keyrings/proxmox-archive-keyring.gpg
```

#### no-subscription

このリポジトリはテストおよび非本番環境での使用に適しています。自由にアクセス可能で、有効なサブスクリプションは不要です。しばらくすると、そのパッケージバージョンはエンタープライズリポジトリでも利用可能になります。

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

```
Types: deb
URI: http://download.proxmox.com/debian/ceph-squidスイート: trixie
コンポーネント: no-subscription
署名者: /usr/share/keyrings/proxmox-archive-keyring.gpg
```

#### test

このリポジトリには最新のパッケージバージョンが含まれており、主に開発者が新機能やバグ修正をテストするために使用されます。

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

```
Types: deb
URI: http://download.proxmox.com/debian/ceph-squidスイート: trixie
コンポーネント: test
署名者: /usr/share/keyrings/proxmox-archive-keyring.gpg
```



#### 警告

Ceph テストリポジトリは（その名前が示す通り）新機能やバグ修正のテストにのみ使用すべきです。

### 3.1.7 Debian フームウェアリポジトリ

Debian Bookworm (Proxmox VE 8) 以降、DFSG で定義される非フリーフームウェアは、新たに作成された Debian リポジトリコンポーネント non-free-firmware に移動されました。

Proxmox VE 9以降、新規インストールではこのリポジトリがデフォルトで有効化され、[早期OSマイクロコード更新](#)を取得できるようになっています。

また、プリインストール済みパッケージ pve-firmware に含まれていない追加の[ランタイムファームウェアファイル](#)を取得することも可能です。

このコンポーネントからパッケージをインストールするには、`editor /etc/apt/sources.list` を実行し、`non-free-firmware` を追加し、`apt update` を実行してください。

Proxmox VE 9 を以前のバージョンの Proxmox VE からアップグレードし、パッケージリポジトリを新しい deb822 スタイルに更新した場合は、代わりに `/etc/apt/sources.list.d/debian` を変更する必要があります。`editor /etc/apt/sources.list.d/debian.sources` を実行し、各スクリプトの Components: で始まる行に `non-free-firmware` を追加してください。

#### 注意

パッケージリポジトリの近代化を推奨します。そうしないと、Debian Trixie の apt がエラーを報告します。これを行うには `apt modernize-sources` を実行してください。

### 3.1.8 SecureApt

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、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検証するために使用します。

公式ISOイメージからProxmox VEをインストールする場合、検証用キーは既にインストールされています。

Debian 上に Proxmox VE をインストールする場合は、以下のコマンドで鍵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-archive-keyring-→  
trixie.gpg -O /usr/share/keyrings/proxmox-archive-keyring.gpg
```

---

#### 注記

上記の wget コマンドは、Debian Trixie ベースの Proxmox リリース用のキーリングを追加します。proxmox-archive-keyring パッケージがインストールされると、このファイルの管理はそちらに移行します。その時点で、新しい Proxmox リリースの鍵が追加または削除されるため、下記のハッシュ値がこのファイルのハッシュ値と一致しなくなる可能性があります。これは意図された動作であり、apt は信頼された鍵のみが使用されることを保証します。**proxmox-archive-keyring のインストール後は、このファイルの修正は推奨されません。**

---

変更後は sha512sum CLI ツールでチェックサムを確認してください:

```
# sha256sum /usr/share/keyrings/proxmox-archive-keyring.gpg  
136673be77aba35dcce385b28737689ad64fd785a797e57897589aed08db6e45 /usr/→  
share/keyrings/proxmox-archive-keyring.gpg
```

または md5sum CLI ツール:

```
# md5sum /usr/share/keyrings/proxmox-archive-keyring.gpg 77c8b1166d15ce8350102ab1bca2fcfb  
/usr/share/keyrings/proxmox-archive-→  
keyring.gpg
```

---

#### 注記

キーをインストールするパスが、リポジトリスタンザの *Signed-By:* 行と一致していることを確認してください。

---

## 3.2 システムソフトウェアの更新

Proxmoxはすべてのリポジトリに対して定期的に更新を提供します。更新をインストールするには、WebベースのGUIまたは以下のCLIコマンドを使用してください：

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

Cephを次のメジャーリリースにアップグレードする場合は、[Cephリポジトリ](#)を参照してください。

---

#### 注記

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ます。詳細はman apt-get、または[\[Hertzog 13\]](#)を参照してください。

---

### ヒント

最新のパッチやセキュリティ関連の修正を取得するには、定期的な更新が不可欠です。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 ファームウェアの更新

この章に記載されているファームウェア更新は、Proxmox VE をベアメタルサーバー上で実行している場合に適用してください。ゲスト内でのファームウェア更新の設定が適切かどうか（例：デバイスパススルーユ用時）は、設定環境に大きく依存するため、本項の範囲外です。

定期的なソフトウェア更新に加え、ファームウェアの更新も信頼性と安全な運用にとって重要です。

ファームウェア更新を入手して適用する際には、利用可能なオプションを組み合わせて、できるだけ早く、あるいは確実に更新を入手することをお勧めします。

ファームウェアという用語は、言語学的には通常、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）に分けられます。

### 3.3.1 永続的ファームウェア

このセクションは全てのデバイスに適用されます。通常BIOS/UEFIアップデートに含まれる更新済みマイクロコードはマザーボード上に保存される一方、他のファームウェアは各デバイス本体に保存されます。この永続的な方法は特にCPUにおいて重要であり、起動時に更新済み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します。



#### 注意

BIOS/UEFIやストレージコントローラなどの更新では、デバイス設定がリセットされる可能性があります。ベンダーの指示を厳守し、現在の設定を必ずバックアップしてください。

利用可能な更新方法についてはベンダーにご確認ください。

- サーバー向けの便利な更新方法としては、DellのライフサイクルマネージャーやHPEのサービスパックなどが挙げられます。
- Linux ユーティリティが利用可能な場合もあります。  
Broadcomネットワークカード向けの`bnxtnvm/niccli`などがあります。  
例としては、NVIDIA ConnectX 用の `mixup` や
- **ハードウェアベンダーとの協力関係があり、サポート対象ハードウェア**を使用している場合、LVFSも選択肢となります。技術的な要件として、システムが2014年以降に製造され、UEFI経由で起動されていることが必要です。

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化するため、独自のfwupdパッケージを提供します。このパッケージは、ハイパーバイザ環境での使用時に確認された問題のため、udisks2パッケージへの依存関係推奨を意図的に除外しています。つまり、`/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する必要があります。例：

#### ファイル /etc/fwupd/daemon.conf

```
# EFIシステムパーティション(ESP)パスの使用場所を上書きします。EspLocation=/boot/efi
```

#### ヒント

更新手順で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

この方法では、ファームウェアを Proxmox VE オペレーティングシステムに保存し、デバイスの[永続化ファームウェア](#)が最新でない場合にそれをデバイスに渡します。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は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永続化ファームウェアに依存す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既にインストールされています。したがって、通常の[システム更新（APT）](#)により、一般的なハードウェアの組み込み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 Firmware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。

追加の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が発生した場合、APTはインストールを中止します。その特定のファームウェアは別の方で入手できる可能性があります。

### 3.3.3 CPUマイクロコード更新

マイクロコード更新は、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ます。CPUパフォーマンスに影響が出る可能性はありますが、パッチ適用済みマイクロコードは通常、カーネル自体が緩和策を講じる必要がある末パッチ状態よりも高いパフォーマンスを発揮します。CPUタイプによっては、欠陥のある工場出荷状態のパフォーマンス結果を、安全でない状態を承知でCPUを動作させない限り再現できなくなる可能性があります。

現在のCPU脆弱性とその緩和策の概要を確認するには、lscpuを実行してください。現実世界で既知の脆弱性は、Proxmox VEホストが[最新状態](#)であり、そのバージョンが[サポート終了しておらず](#)、かつ前回のカーネル更新以降に少なくとも再起動されている場合にのみ検出されます。

[永続的なBIOS/UEFI更新](#)による推奨マイクロコード更新に加え、独立した手法として「[早期OSマイクロコード更新](#)」も利用可能です。マザーボードベンダーがBIOS/UEFI更新を提供しなくなった場合にも便利で有用です。いずれの方法を使用する場合でも、マイクロコード更新を適用するには再起動が常に必要です。

#### 早期OSマイクロコード更新の設定

Linuxカーネルが起動時に早期に適用するマイクロコード更新を設定するには、以下の操作が必要です:

1. [Debian ファームウェアリポジトリ](#)を有効にする
2. 利用可能な最新パッケージを取得するapt update (またはウェブインターフェースの「Node→Updates」セクションを使用)
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:

- Intel 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

今後のマイクロコード更新も、ロードするには再起動が必要です。

#### マイクロコードバージョン

比較やデバッグ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には：

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

マイクロコードパッケージには様々なCPU向けの更新が含まれています。しかし、特定のCPU向けの更新は頻繁に提供されない場合があります。そのため、パッケージの日付を確認するだけでは、メーカーが実際にあなたのCPU向けに更新をリリースした時期を特定できません。

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

```
# dmesg | grep microcode
[    0.000000] microcode: microcode updated early to revision 0xf0, date= 2021-11-12
[    0.896580] microcode: Microcode Update Driver: v2.2.          ←
[
```

#### トラブルシューティング

デバッグ目的で、システム起動時に定期的に適用される早期OSマイクロコード更新の設定を一時的に無効化するには、以下の手順を実行します:

1. ホストを[安全に](#)再起動できることを確認してください
2. ホストを再起動し、GRUBメニューを表示させる（非表示の場合はSHIFTキーを押す）
3. 目的のProxmox VEブートエントリでEキーを押す
4. `linux` で始まる行に移動し、スペース区切りで `dis_ucode_ldr` を追加する
5. CTRL-X を押すと、今回は早期OSマイクロコード更新なしで起動します

最近のマイクロコード更新に関連する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検討すべきです。そうしないと、より新しいマイクロコードが問題なく動作するにもかかわらず、古すぎるマイクロコードが[統的に読み込まれる可能性](#)があります。

Debianリポジトリに以前のマイクロコードパッケージバージョンが存在する場合、ダウングレードが可能です。例：

```
# apt list -a intel-microcode Listing... Done
intel-microcode/stable-security,now 3.20230808.1~deb12u1 amd64 [installed]intel-microcode/stable
3.20230512.1 amd64
```

```
# apt install intel-microcode=3.202305*
```

...

```
'intel-microcode' の選択バージョン '3.20230512.1' (Debian:12.1/stable [amd64]) ←→  
microcode'
```

...

```
dpkg: 警告: intel-microcode を 3.20230808.1~deb12u1 から ←→ にダウングレードします  
3.20230512.1
```

...

```
intel-microcode: マイクロコードは次回の起動時に更新されます
```

...

ホストを[安全に](#)再起動できることを（再度）確認してください。お使いのCPUタイプ向けのマイクロコードパッケージに含まれる可能性のあ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。

#### ヒント

ダウングレードしたパッケージをしばらく保留し、後日改めて新しいバージョンを試すのが合理的です。将来同じパッケージバージョンであっても、システム更新によって問題が修正されている可能性があります。

```
# apt-mark hold intel-microcode intel-  
microcode を保留状態に設定。
```

```
# apt-mark unhold intel-microcode # apt update  
# apt full-upgrade
```

## 3.4 ネットワーク設定

Proxmox VEはLinuxネットワークスタックを使用しています。これにより、Proxmox VEノード上のネットワーク設定方法に高い柔軟性が提供されます。設定はGUI経由で行うか、ネットワーク設定全体を含むファイル

/etc/network/interfaces（ネットワーク設定全体を記述）を手動編集する方法があります。インターフェース (5) マニュアルページには完全なフォーマット記述が記載されています。すべてのProxmox VEツールはユーザーによる直接編集を回避するよう努めていますが、エラー防止のためGUIの使用が推奨されます。

ゲストを基盤となる物理ネットワークに接続するには、Linuxブリッジインターフェース（一般にvmbrXと呼ばれる）が必要です。これは、ゲストと物理インターフェースが接続される仮想スイッチとることができます。このセクションでは、[ボンディング](#)による冗長化、[VLAN](#)、[ルーティング](#)およびNAT設定など、さまざまなユースケースに対応するためのネットワーク設定例をいくつか紹介します。

[ソフトウェア定義ネットワーク](#) (SDN) は、Proxmox VEクラスターにおけるより複雑な仮想ネットワークの選択肢です。

#### 警告

従来のDebianツールであるifupおよびifdownの使用は推奨されません。特にifdown vmbrXを実行するとゲストの全トラフィックが中断される一方、後で同じブリッジに対してifupを実行してもゲストが再接続されないといった落とし穴があるためです。

### 3.4.1 ネットワーク変更の適用

Proxmox VEは変更を直接/etc/network/interfacesに書き込まない。代わりに/etc/network/interfaces.newという一時ファイルに書き込む。これにより関連する変更をまとめて一度に行える。また、適用前に変更内容の正確性を確認できる。誤ったネットワーク設定はノードにアクセス不能になる可能性があるためである。

#### ifupdown2によるネットワークのライブ再読み込み

推奨される ifupdown2 パッケージ (Proxmox VE 7.0 以降の新しいインストールではデフォルト) を使用すると、再起動なしでネットワーク設定の変更を適用できます。GUI 経由でネットワーク設定を変更した場合、「設定を適用」ボタンをクリックできます。これにより、変更がステージング用の interfaces.new ファイルから /etc/network/interfaces に移動され、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`を実行して適用できます。

---

#### 注意

Debian 上に Proxmox VE をインストールした場合、または古い Proxmox VE インストールから 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: apt install ifupdown2

---

#### ノードの再起動による適用

新しいネットワーク設定を適用する別の方法は、ノードを再起動することです。この場合、systemd サービス pvenetcommit が、ネットワークサービスがその設定を適用する前に、ステージングされた interfaces.new ファイルをアクティブにします。

### 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。この命名規則はバージョン 5.0 以降の新しい Proxmox VE インストールで使用されます。
- イーサネットデバイス: eth[N] (0 ≤ N、eth0, eth1, ...)。この命名規則は、5.0 リリース以前にインストールされた Proxmox VE ホストで使用されます。5.0へのアップグレード時には、名前はそのまま維持されます。
- ブリッジ名: 一般的に vmbr[N] (0 ≤ N ≤ 4094 (vmbr0 - vmbr4094)) ですが、文字で始まり最大10文字の英数字文字列であれば任意の名称を使用できます。
- ボンディング: bond[N]。0 ≤ 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 (例: eno1.50, bond1.30)。これにより、デバイス名がデバイスタイプを示唆するため、ネットワーク問題のデバッグが容易になります。

## 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に対してバージョン管理された命名規則を定義しています。この規則では、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。続く文字列はデバイスドライバ、デバイスの位置、その他の属性によって異なります。可能なパターンの一部は以下の通りです：

- `o<index>[n<物理ポート名>;|d<デバイスポート>] — オンボードデバイス`
- `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ると：

- `eno1` — 最初のオンボードNIC
- `enp3s0f1` — PCIバス3のスロット0にあるNICの機能1

可能なデバイス名パターンの完全な一覧については、[`systemd.net-naming-scheme\(7\)` マニュアルページ](#)を参照してください。

systemd の新バージョンでは、ネットワークデバイス命名スキームの新バージョンが定義され、それがデフォルトで使用される場合があります。その結果、例えば Proxmox VE のメジャーアップグレード時など、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可能性があります。命名スキームの新バージョンによる名前変更を回避するには、特定の命名スキームバージョンを手動で固定することができます（[下記参照](#)）。

ただし、固定した命名スキームバージョンがあっても、カーネルやドライバの更新により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名を変更しないようにするには、リンクファイルを使用して手動で名前を上書きできます（[下記参照](#)）。

ネットワークインターフェース名に関する詳細は、「[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。

## 特定の命名スキームバージョンの固定

ネットワークデバイスの命名スキームの特定バージョンを固定するには、カーネルコマンドラインに `net.naming-scheme=<バージョン>` パラメータを追加します。命名スキームのバージョン一覧については、[`systemd.net-naming-scheme\(7\)` マニュアルページ](#)を参照してください。

例えば、新規のProxmox VE

8.0 インストール環境では、以下のカーネルコマンドラインパラメータを追加します：

```
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オーバーライド

### pve-network-interface-pinning ツールの使用

Proxmox VEでは、ネットワークデバイスの名前を上書きするための.linkファイルを自動生成するツールを提供しています。また、以下のファイル内で古いインターフェース名の出現箇所を自動的に置換します：

- /etc/network/interfaces
- /etc/pve/nodes/<nodename>/host.fw
- /etc/pve/sdn/controllers.cfg
- /etc/pve/sdn/fabrics.cfg

#### 注記

生成されたマッピングは、それが生成されたノードにローカルであるため、ファイアウォールデータセンター構成（/etc/pve/firewall/cluster.fw）に含まれるインターフェース名は自動的に更新されません。

生成されたリンクファイルは /usr/local/lib/systemd/network に保存されます。設定ファイルについては、同じ場所に .new サフィックス付きの新しいファイルが生成されます。これにより、diff（または好みの差分表示ツール）を使用して設定への変更点を確認できます：

```
diff -y /etc/network/interfaces /etc/network/interfaces.new
```

問題のある変更を確認した場合、または再起動前にピンニングツールによる変更を元に戻したい場合は、/usr/local/lib/systemd/network ディレクトリ内のすべての .new ファイルと対応するリンクファイルを削除してください。

以下のコマンドは、まだ.linkファイルを持たない全ての物理ネットワークインターフェースに対して.linkファイルを生成し、選択された Proxmox VE 設定ファイルを更新します（上記参照）。生成される名前はデフォルトの

.linkファイルが存在しない物理ネットワークインターフェース全てに対して.linkファイルを生成し、選択されたProxmox VE設定ファイルを更新します（上記参照）

```
pve-network-interface-pinning generate  
)。生成される名前にはデフォルトの接頭辞nicが使用されるため、結果として生成されるインターフェース名はnic1、nic2、...となります。
```

デフォルトのプレフィックスは --prefix フラグで上書きできます：

```
pve-network-interface-pinning generate --prefix myprefix
```

特定のインターフェースのみを固定することも可能です：

```
pve-network-interface-pinning generate --interface enp1s0
```

特定のインターフェースを固定する場合、インターフェースを固定する正確な名前を指定できます：

```
pve-network-interface-pinning generate --interface enp1s0 --target-name-->  
if42
```

pve-network-interface-pinning による変更をネットワーク設定に反映させるには、ノードの再起動が必要です。

## 手動による方法

特定のネットワークデバイスにカスタムの[systemd.link ファイル](#)を使用して手動で名前を割り当てることができます。これにより、最新のネットワークデバイス命名規則に基づいて割り当てられる名前を上書きします。この方法により、カーネル更新、ドライバ更新、または命名規則の新バージョンによる命名変更を回避できます。

カスタムリンクファイルは /etc/systemd/network/ に配置し、`<n>-<id>.link` という形式で命名します。ここで n は 99 未満の優先度、id は識別子です。リンクファイルは 2 つのセクションで構成されます：[Match] はファイルが適用されるインターフェースを決定し、[Link] はそれらのインターフェースの設定方法（命名を含む）を決定します。

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] セクション内でそのデバイスを一意かつ恒久的に識別する手段が必要です。MAC アドレスは変更される可能性が低いため、MACAddress オプションを使用してデバイスの MAC アドレスを照合する方法が考えられます。

[Match] セクションには、同じ MAC アドレスを持つブリッジ/ポンディング/VLANインターフェースではなく、期待される物理インターフェースのみに一致させるため、Type オプションも含める必要があります。ほとんどの設定では、イーサネットデバイスにのみ一致させるため Type を ether に設定しますが、他の選択肢が必要な設定もあります。詳細は [systemd.link\(5\) マニュアルページ](#) を参照してください。

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ます。

リンクファイルは initramfs にコピーされるため、リンクファイルの追加・変更・削除後は initramfs を更新することを推奨します：

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

例えば、MAC アドレスが aa:bb:cc:dd:ee:ff のイーサネットデバイスに enwan0 という名前を割り当てるには、以下の内容で

/etc/systemd/network/10-enwan0.link ファイルを作成します：

```
[Match]
MACAddress=aa:bb:cc:dd:ee:ff
Type=ether

[Link]
Name=enwan0
```

新しい名前を使用するよう /etc/network/interfaces を調整し、前述のように initramfs を更新することを忘れないでください。  
変更を有効にするにはノードの再起動が必要です。

### 注意

Proxmox VE がインターフェースを物理ネットワークデバイスとして認識し、GUI 経由で設定できるようにするために、名前を en または eth で始まるように割り当てる 것을 推奨합니다. また、将来的に他のインターフェース名と衝突しないようにする必要があります。一つの方法は、systemd がネットワークインターフェースに使用する名前パターン（[上記参照](#)）に一致しない名前を割り当てる 것입니다。例えば、上記の例にある enwan0 などです。

リンクファイルの詳細については、[systemd.link\(5\) マニュアルページ](#) 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とリソースに応じて、ブリッジ接続、ルーティング接続、またはマスカレード接続のいずれかのネットワーク設定を選択できます。

### プライベートLAN内のProxmox VEサーバーが外部ゲートウェイ経由でインターネットに接続する場合

このケースではブリッジモデルが最も理にかなっており、新規Proxmox VE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ースを持ちます。これは、ゲストのネットワークカードをLAN上の新しいスイッチに直接接続するのと効果的に同様であり、Proxmox VEホストがスイッチの役割を果たします。

### ホスティングプロバイダー上のProxmox VEサーバー、ゲスト用パブリックIP範囲付き

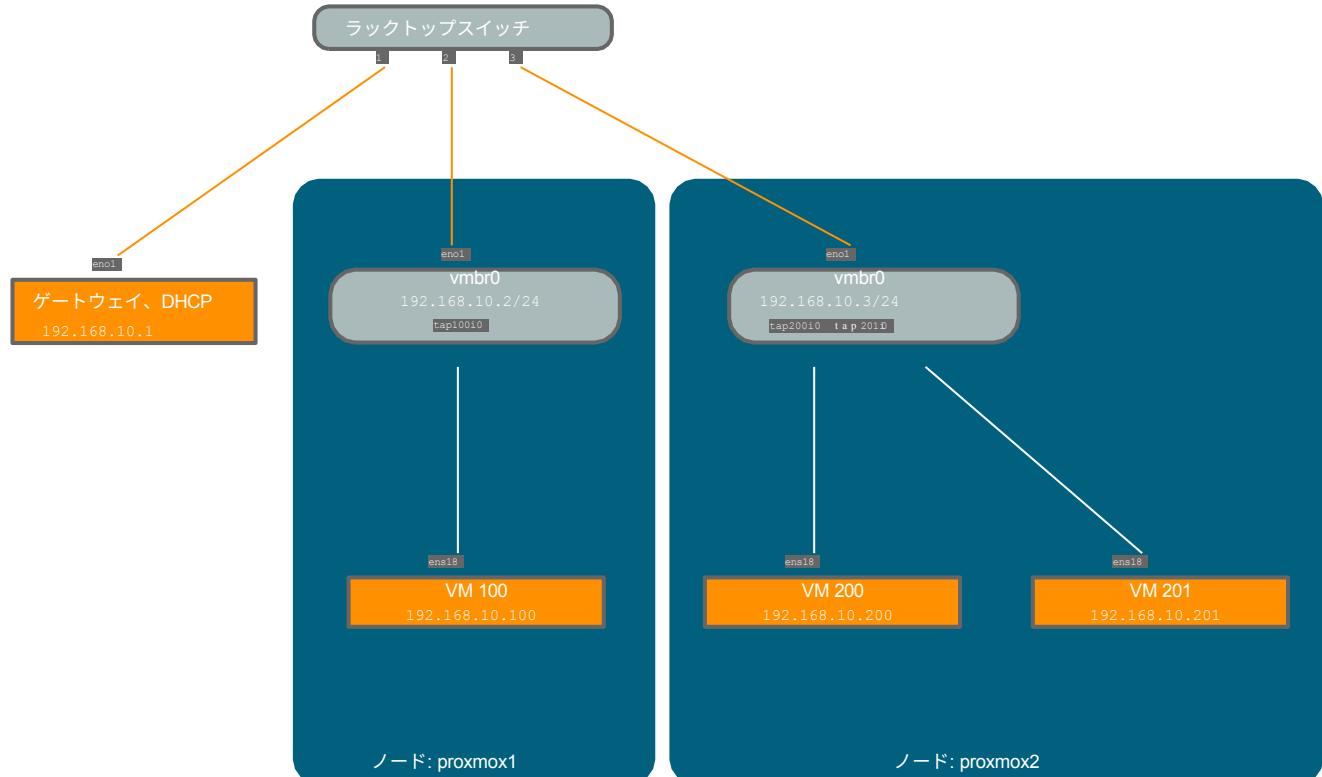
この設定では、プロバイダーが許可する内容に応じて、ブリッジ接続モデルまたはルーティング接続モデルのいずれかを使用できます。

### ホスティングプロバイダー上のProxmox VEサーバー（単一のパブリックIPアドレス）

この場合、ゲストシステムが外部ネットワークにアクセスする唯一の方法は、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の設定が必要です。

さらなる柔軟性のため、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」）を設定できます。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可能になります。

#### 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が単一のブリッジを共有することも、複数のブリッジを作成してネットワードメインを分離することもできます。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、最初のイーサネットカードに接続された vmbr0 という単一のブリッジを作成します。/etc/network/interfaces 内の対応する設定は次のようにになります：

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

auto vmbr0
iface vmbr0 inet static
    address 192.168.10.2/24
    gateway 192.168.10.1
    bridge-ports eno1
    bridge-stp off
    bridge-fd 0
```

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します。ネットワーク側では、これらのすべての仮想マシンを单一のネットワークケーブルで接続しているにもかかわらず、各仮想マシンが独自のMACアドレスを持っていると見なします。

### 3.4.5 ルーティング構成

ほとんどのホスティングプロバイダは上記の設定をサポートしていません。セキュリティ上の理由から、单一インターフェース上で複数のMACアドレスを検知すると、直ちにネットワーク機能を無効化します。

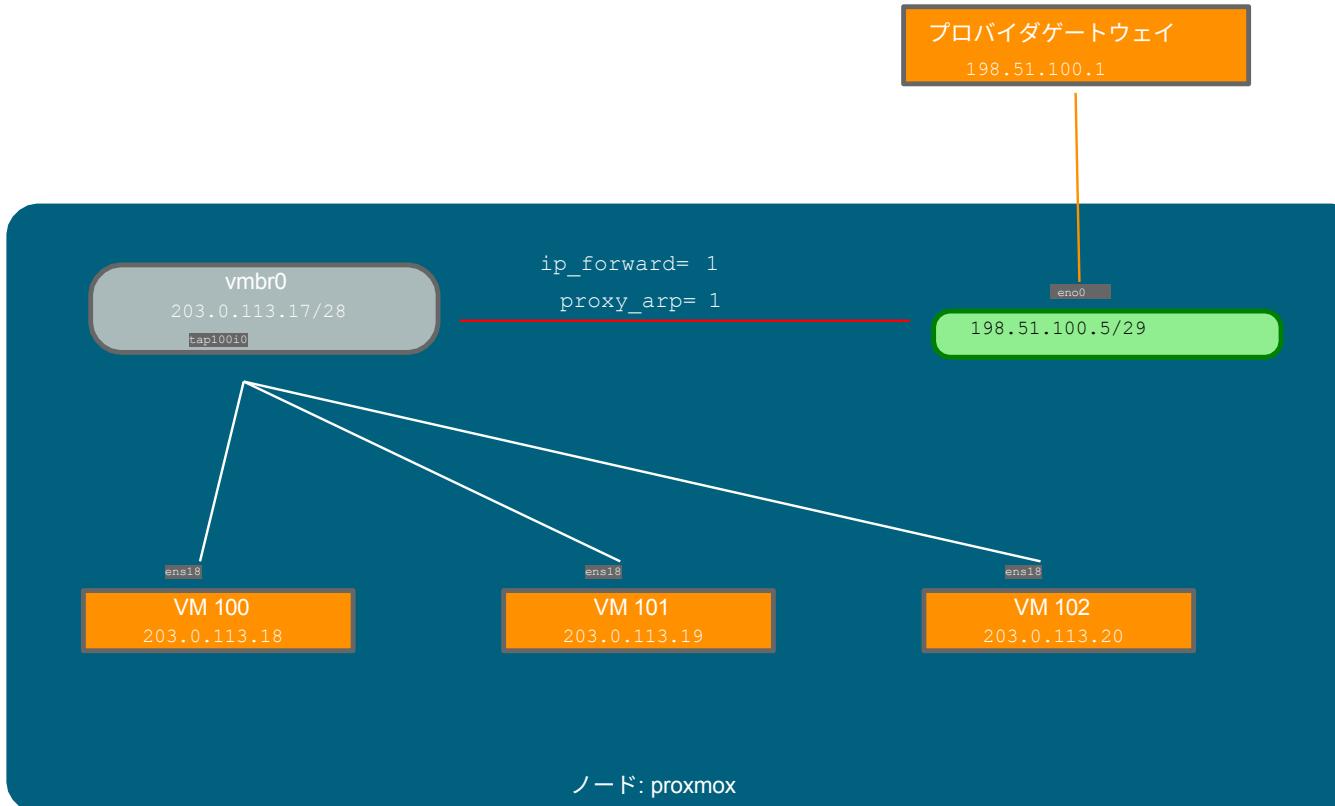
---

#### ヒント

一部のプロバイダーでは、管理インターフェースを通じて追加のMACアドレスを登録できます。これにより問題は回避できますが、各仮想マシンごとにMACアドレスを登録する必要があるため、設定が煩雑になる場合があります。

---

すべてのトラフィックを单一のインターフェース経由で「ルーティング」することで、この問題を回避できます。これにより、すべてのネットワークパケットが同一のMACアドレスを使用するようになります。



よくあるシナリオとして、パブリックIP（この例では198.51.100.5と仮定）と、VM用の追加IPブロック（203.0.113.16/28）がある場合が挙げられます。このような状況では、以下の設定を推奨します：

```
auto lo
iface lo inet loopback

auto eno0
iface eno0 inet static
    address 198.51.100.5/29 gateway
    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

auto vmbr0
iface vmbr0 inet static
    address 203.0.113.17/28 bridge-
    ports none bridge-stp off
    bridge-fd 0
```

### 3.4.6 iptablesによるマスカレード(NAT)

マスカレードにより、プライベートIPアドレスのみを持つゲストは、送信トラフィックにホストのIPアドレスを使用してネットワークにアクセスできます。各送信パケットはiptablesによって書き換えられ、ホストからの発信のように見せかけられ、応答も同様に書き換えられて元の送信者にルーティングされます。

```
auto lo
```

```
iface lo inet loopback

auto eno1
#実際のIPアドレスiface eno1 inet
static
    address 198.51.100.5/24 gateway
    198.51.100.1

auto vmbr0
#プライベートサブネットワークiface vmbr0
inet static
    address 10.10.10.1/24 bridge-
    ports nonebridge=stp off
    bridge-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 eno1 -j MASQUERADE
    post-down iptables -t nat -D POSTROUTING -s '10.10.10.0/24' -o eno1 -j MASQUERADE
```

### 注記

ファイアウォールを有効にした一部のマスカーレイド設定では、発信接続にconntrackゾーンが必要になる場合があります。そうしないと、発信接続がVMブリッジのPOSTROUTING (MASQUERADEではなく) を優先するため、ファイアウォールによってブロックされる可能性があります。

この問題を解決するには、`/etc/network/interfaces` に以下の行を追加します:

```
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
```

詳細については、以下のリンクを参照してください:

[Netfilter パケットフロー](#)

[netdev-listにおけるconntrackゾーン導入パッチ](#)

[rawテーブルのTRACEを使用した分かりやすい解説ブログ記事](#)

## 3.4.7 Linux Bond

ボンディング (NICチーム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を単一のネットワークデバイスに結合する技術です。ネットワークの耐障害性向上、パフォーマンス向上、あるいはその両方の達成が可能です。

ファイバーチャネルなどの高速ハードウェアや関連するスイッチングハードウェアは、非常に高価になる場合があります。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ースとして認識させ、速度を2倍にすることができます。これは、ほとんどのスイッチでサポートされている、Linuxカーネルのネイティブ機能です。ノードに複数のイーサネットポートがある場合、ネットワークケーブルを異なるスイッチに接続することで障害発生点を分散でき、ネットワーク障害時にはボンディング接続が一方のケーブルにフェイルオーバーします。

結合されたリンクは、ライブマイグレーションの遅延を改善し、Proxmox VEクラスタノード間のデータ複製速度を向上させます。

ボンディングには7つのモードがあります：

- **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ース (NIC) スレーブから最後のスレーブまで、ネットワークパケットを順番に送信します。このモードは負荷分散と耐障害性を提供します。
- **アクティブ・バックアップ (active-backup):** ボンディング内のNICスレーブは1つだけがアクティブとなる。アクティブなスレーブが障害を起こした場合に限り、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での歪みを避けるため、1つのNIC（ポート）上でのみ外部から見える。このモードはフォールトトレランスを提供する。
- **XOR（バランス-XOR）：**ネットワークパケットを[(送信元MACアドレスと宛先MACアドレスのXOR演算結果) ÷ NICスレーブ数]に基づいて送信します。これにより、宛先MACアドレスごとに同一のNICスレーブが選択されます。このモードは負荷分散と耐障害性を提供します。
- **ブロードキャスト (broadcast):** ネットワークパケットをすべてのスレーブネットワークインターフェースで送信します。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):** 同一の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仕様に基づき、アクティブなアグリゲーターグループ内の全スレーブネットワークインターフェースを利用します。
- **適応型送信負荷分散 (balance-tlb):** 特別なネットワークスイッチ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。送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イススレーブの現在の負荷（速度に対して相対的に計算）に応じて分散されます。受信トラフィックは、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イスで受信されます。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。
- **適応型負荷分散 (balance-alb):** balance-tlb に IPv4 トラフィック用の受信負荷分散 (rlb) を追加したもので、特別なネットワークスイッチサポートを必要としません。受信負荷分散はARPネゴシエーションにより実現されます。ボンディングドライバは、ローカルシステムから送信されるARP応答を送信途中で傍受し、送信元ハードウェアアドレスを単一の論理ボンディングインターフェース内のNICスレーブの一つ固有のハードウェアアドレスで上書きします。これにより、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アドレスを使用します。

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、対応するボンディングモード（802.3ad）の使用を推奨します。サポートしていない場合は、一般的にactive-backupモードを使用してください。

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推奨します。Corosyncはネットワーク冗長性のためにボンディングを必要としません。なぜなら、いずれかのネットワークが使用不能になった場合、自身でネットワークを切り替えることができるからです。一部のボンディングモードはCorosyncにとって問題となることが知られています。詳細は「[Corosync over Bonds](#)」を参照してください。

以下のボンディング構成は分散/共有ストレージネットワークとして使用可能です。これにより速度向上が得られ、ネットワークは耐障害性を備えます。

#### 例: 固定IPアドレスを使用したボンディング

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

```

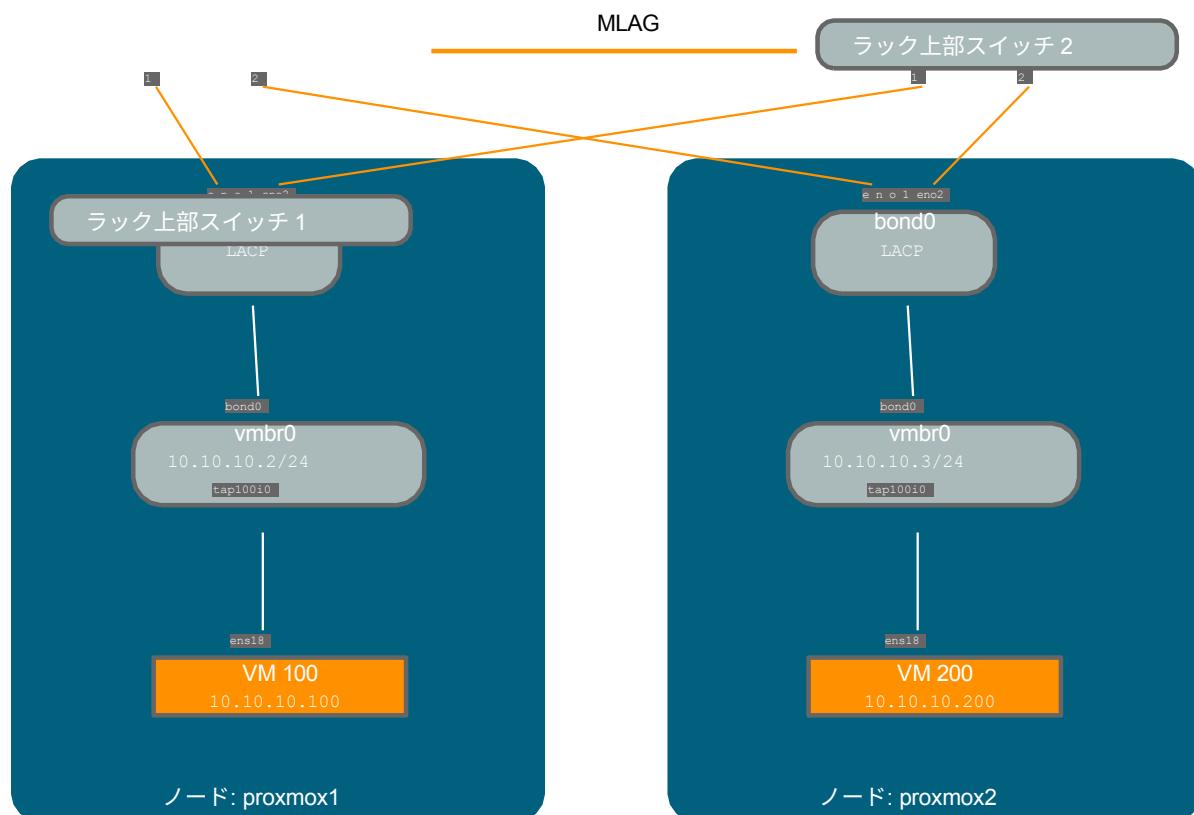
iface eno2 inet manual

iface eno3 inet manual auto

bond0
iface bond0 inet static bond-slaves
    eno1 eno2
    address 192.168.1.2/24 bond-
    miimon 100
    bond-mode 802.3ad
    bond-xmit-hash-policy layer2+3

auto vmbr0
iface vmbr0 inet static
    address 10.10.10.2/24 gateway
    10.10.10.1bridge-ports eno3
    bridge-stp off
    bridge-fd 0

```



別の方法として、ボンディングを直接ブリッジポートとして使用する方法があります。これによりゲストネットワークの耐障害性を実現できます。

#### 例: ボンディングをブリッジポートとして使用

```

auto lo
iface lo inet loopback

```

```
iface eno1 inet manual

iface eno2 inet manual auto

bond0
iface bond0 inet manual bond-slaves
    eno1 eno2 bond-miimon 100
    bond-mode 802.3ad
    bond-xmit-hash-policy layer2+3

auto vmbr0
iface vmbr0 inet static
    address 10.10.10.2/24 gateway
    10.10.10.1bridge-ports bond0
    bridge-stp off
    bridge-fd 0
```

### 3.4.8 VLAN 802.1Q

仮想LAN（VLAN）は、ネットワークのレイヤ2で分割・分離されたブロードキャストドメインです。これにより、物理ネットワーク内に複数のネットワーク（最大4096）を相互に独立して構築することが可能となります。

各VLANネットワークは、タグと呼ばれる番号で識別されます。ネットワークパケットには、どの仮想ネットワークに属するかを識別するためのタグが付けられます。

#### ゲストネットワーク向けVLAN

Proxmox VEは標準でこの設定をサポートしています。VM作成時にVLANタグを指定可能です。VLANタグはゲストネットワーク設定の一部です。ネットワーク層はブリッジ構成に応じて、VLAN実装のための異なるモードをサポートします：

- **LinuxブリッジにおけるVLAN認識:** この場合、各ゲストの仮想ネットワークカードはVLANタグに割り当てられ、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、その場合はゲスト側での設定が必要となります。
- **Linuxブリッジ上の「従来型」VLAN:** VLAN認識方式とは対照的に、この方式は透過的ではなく、各VLANごとにブリッジを伴うVLANデバイスを作成します。例えばVLAN 5上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生成され、再起動が行われるまで残存します。
- **Open vSwitch VLAN:** このモードではOVSの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、外部からの影響を受けません。利点は、単一の仮想NICで複数のVLANを使用できることです。

## ホスト上のVLAN

ホストが分離されたネットワークと通信できるようにするため。VLANタグはあらゆるネットワークデバイス（NIC、ボンディング、ブリッジ）に適用可能です。一般的には、物理NICとの間に抽象化レイヤーが最も少ないインターフェースでVLANを設定すべきです。

例：ホスト管理アドレスを別VLANに配置するデフォルト構成の場合

### 例: Proxmox VE管理IPにVLAN 5を使用（従来のLinuxブリッジ構成）

```
auto lo
インターフェース lo inet ループバックイン

インターフェース eno1 inet 手動インターフェース

ス eno1.5 inet 手動

auto vmbr0v5
インターフェイス vmbr0v5 インターネット静的
    address 10.10.10.2/24 gateway
    10.10.10.1 bridge-ports eno1.5
    bridge-stp off
    bridge-fd 0

auto vmbr0
インターフェイス vmbr0 inet 手動
    bridge-ports eno1
    bridge-stp off bridge-fd
    0
```

### 例: Proxmox VE 管理 IP に VLAN 5 を使用し、VLAN 対応の Linux ブリッジを設定

```
auto lo
iface lo inet loopbackiface eno1
inet manual

auto vmbr0.5
iface vmbr0.5 inet static
    address 10.10.10.2/24 gateway
    10.10.10.1

auto vmbr0
iface vmbr0 inet manual
    bridge-ports eno1
    bridge-stp off bridge-fd
    0
    bridge-vlan-aware yes
```

bridge-vids 2-4094

次の例は同じ設定ですが、このネットワークをフェイルセーフにするためにボンディングが使用されています。

例: Proxmox VE 管理用 IP アドレスには、従来の Linux ブリッジ方式で bond0 に VLAN 5 を使用する

```
auto lo
iface lo inet loopbackiface eno1

inet    manualiface    eno2    inet

manual

auto bond0
iface bond0 inet manual bond-slaves
    eno1 eno2 bond-miimon 100
    bond-mode 802.3ad
    bond-xmit-hash-policy layer2+3iface bond0.5

inet manual

auto vmbr0v5

インターフェイス vmbr0v5 inet static
    address 10.10.10.2/24 gateway
    10.10.10.1 bridge-ports bond0.5
    bridge-stp off
    bridge-fd 0

auto vmbr0
iface vmbr0 inet manual
    bridge-ports bond0 bridge-
    stp off bridge-fd 0
```

### 3.4.9 ノードでのIPv6無効化

Proxmox VE は、IPv6 が導入されているかどうかに関係なく、すべての環境で正しく動作します。すべての設定は、提供されているデフォルトのままにしておくことをお勧めします。

それでもノードで IPv6 のサポートを無効にする必要がある場合は、適切な `sysctl.conf`

(5) スニペットファイルを作成し、適切な `sysctl` を設定します。例えば、`/etc/sysctl.d/disable-ipv6.conf` に内容を追加します：

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

この方法は、**カーネルコマンドラインでIPv6モジュールの読み込みを無効化**するよりも推奨されます。

### 3.4.10 ブリッジでのMAC学習の無効化

デフォルトでは、仮想ゲストとそのネットワークの円滑な動作を確保するため、ブリッジ上で MAC 学習が有効になっています。

しかし、一部の環境ではこれが望ましくない場合があります。Proxmox VE 7.3以降では、`/etc/network/interfaces`内のブリッジに対して`bridge-disable-mac-learning 1`設定を適用することで、ブリッジ上のMAC学習を無効化できます。例：

```
# ...

auto vmbr0
iface vmbr0 inet static
    address 10.10.10.2/24 gateway
    10.10.10.1 bridge-ports ens18
    bridge-stp off
    bridge-fd 0
    bridge-disable-mac-learning 1
```

有効化すると、Proxmox VE は、ゲストがネットワークを引き続き使用できるように、VM およびコンテナから設定された MAC アドレスをブリッジの転送データベースに手動で追加します。ただし、実際の MAC アドレスを使用している場合に限りません。

## 3.5 時刻同期

Proxmox VE クラスタースタック自体は、すべてのノードの時刻が正確に同期されていることに大きく依存しています。Cephなどの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されていないと正常に動作しません。

ノード間の時刻同期は「ネットワーク時刻プロトコル」（NTP）を使用して実現できます。Proxmox VE 7 以降では `chrony` がデフォルトの NTP デーモンとして使用され、Proxmox VE 6 では `systemd-timesyncd` が使用されます。どちらも一連の公開サーバーを使用するよう事前設定されています。



#### 重要

システムをProxmox VE 7にアップグレードする場合、手動で

`chrony`、`ntp`、または`openntpd`のいずれかを手動でインストールすることを推奨します。

### 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 サーバーを使用することが望ましい場合があります。たとえば、制限の厳しいファイアウォールルールにより Proxmox VE ノードが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 サーバーを設定し、NTP デーモンにそれらを使用するよう指示する必要があります。

**chrony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:

```
server      ntp1.example.com      iburstserver
ntp2.example.com          iburstserver
ntp3.example.com  iburst
```

chronyを再起動します:

```
# systemctl restart chronyd
```

新たに設定したNTPサーバーが使用されていることを確認するため、ジャーナルを確認します:

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

```
...
Aug 26 13:00:09 node1 systemd[1]: chrony (NTPクライアント/サーバー) を起動しました。
Aug 26 13:00:15 node1 chronyd[4873]: ソース 10.0.0.1 (ntp1.example↔
.com)
8月26日 13:00:15 node1 chronyd[4873]: システムクロック TAI オフセットが 37 に設定されました↔
秒
...
```

**systemd-timesyncdを使用するシステムの場合:**

systemd-timesyncd が使用するサーバーを /etc/systemd/timesyncd.conf で指定:

```
[Time]
NTP=ntp1.example.com ntp2.example.com ntp3.example.com ntp4.example.com
```

次に、同期サービスを再起動します (systemctl restart systemd-timesyncd)。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には、ジャーナルを確認します (journalctl --since -1h -u systemd-timesyncd) で確認します:

```
...
Oct  07 14:58:36 node1  systemd[1]: Stopping Network Time Synchronization... Oct  07 14:58:36 node1
systemd[1]: Starting Network Time Synchronization... Oct 07 14:58:36 node1 systemd[1]: Started Network Time
Synchronization.
10月 07 14:58:36 node1 systemd-timesyncd[13514]: NTP サーバー 10.0.0.1:123 (ntp1.example.com) ↔
    を使用しています。
10月 07 14:58:36 node1 systemd-timesyncd[13514]: interval/delta/delay/jitter↔
...
...
```

## 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信する外部メトリックサーバーを定義できます。

現在サポートされているのは：

- Graphite (詳細は <https://graphiteapp.org> を参照)
- InfluxDB (参照: <https://www.influxdata.com/time-series-platform/influxdb/> )

外部メトリックサーバーの定義は `/etc/pve/status.cfg` に保存され、Web インターフェースから編集できます。

### 3.6.1 Graphiteサーバー設定

Create: Graphite

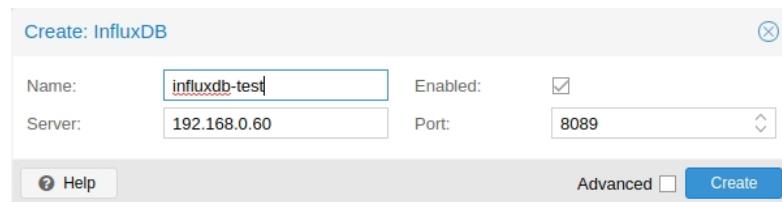
Name:	<input type="text" value="graphite-test"/>	Enabled:	<input checked="" type="checkbox"/>
Server:	<input type="text" value="192.168.0.50"/>	Port:	<input type="text" value="2003"/>
		Path:	<input type="text" value="proxmox"/>

デフォルトのポートは **2003** に設定され、デフォルトの graphite パスは **proxmox** です。

デフォルトでは、Proxmox VE は UDP 経由でデータを送信するため、Graphite 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。ここでは、標準の **1500 MTU** を使用していない環境向けに、最大伝送単位 (MTU) を設定できます。

また、プラグインを TCP を使用するように設定することもできます。重要な pvestatd 統計収集デーモンをブロックしないように、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プラグイン設定



Proxmox VEはUDP経由でデータを送信するため、InfluxDBサーバー側で対応設定が必要です。必要に応じてMTUもここで設定可能です。

以下はInfluxDBの設定例です（InfluxDBサーバー上で）：

```
[[udp]]
enabled= true
bind-address= "0.0.0.0:8089" database = "proxmox"
batch-size = 1000batch-
timeout= "1s"
```

この設定により、サーバーはすべてのIPアドレスのポート8089でリスンし、

**proxmox** データベースに書き込みます

または、プラグインをInfluxDB 2.xのhttp(s) APIを使用するように設定することもできます。InfluxDB 1.8.xには、このv2 API用の前方互換性のある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、設定に応じて *influxdbproto* を *http* または *https* に設定します。デフォルトでは、Proxmox VE は組織 *proxmox* およびバケット/データベース *proxmox* を使用します（それぞれ設定の *organization* および *bucket* で設定可能）。

InfluxDBのv2 APIは認証が必要であるため、適切なバケットへの書き込み権限を持つトークンを生成し設定する必要があります。

1.8.xのv2互換APIでは、トークンとして*user:password*を使用できます（必要な場合）。また、

組織名を省略できます（InfluxDB 1.xでは意味を持たないため）。

また、*timeout* 設定で HTTP タイムアウト（デフォルトは 1 秒）を設定したり、*max-body-size* 設定で最大バッチサイズ（デフォルトは 25000000 バイト）を設定したりすることもできます（これは、InfluxDB の同名の設定に対応しています）。

### 3.7 ディスクの健全性モニタリング

堅牢で冗長性のあるストレージの使用が推奨されますが、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 以降では、パッケージ smartmontools<sup>1</sup> がインストールされ、必須となります。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・制御するためのツール群です。

ディスクの状態は次のコマンドで取得できます:

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

ここで /dev/sdX はローカルディスクのパスです。出力に以下のように表示される場合:

```
SMART support is: Disabled
```

以下のコマンドで有効化できます:

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

smartctl の使用方法の詳細については、man smartctl を参照してください。

デフォルトでは、smartmontools デーモン smartd はアクティブかつ有効化されており、

- /dev/sd[a-z]
- /dev/sd[a-z][a-z]
- /dev/hd[a-t]
- または /dev/nvme[0-99]

に一致するデバイスを30分ごとにエラーと警告についてスキャンし、問題を検出すると root宛てに電子メールを送信します。

smartd の設定方法の詳細については、man smartd および man smartd.conf を参照してください。

ハードウェア RAID コントローラでハードディスクを使用している場合、RAID アレイ内のディスクおよびアレイ自体を監視するツールが用意されている可能性が高いです。詳細については、お使いの RAID コントローラ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 インストール CD はローカルディスク管理にいくつかのオプションを提供しており、現在のデフォルト設定では LVM を使用します。インストーラでは、そのような設定用に単一のディスクを選択でき、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、小さな 8GB ディスクを使用したテストインストールからのものです:

```
# pvs PV
/dev/sda3      VG      Fmt Attr PSize PFree
                pve    lvm2 a--     7.87GB 876.00MB

# vgs
VG            #PV #LV #SN 属性          VSize VFree
pve           1    3   0wz--n- 7.87g 876.00m
```

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:

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

LV	VG	属性	LSize	プール起源	データ%	メタ%
データ	pve	twi-a-tz--	4.38g		0.00	0.63
ルート	pve	-wi-ao----	1.75g			
swap	pve	-wi-ao----	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が含まれています。

### swap

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ます。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、このタスクに最適です。

Proxmox VE バージョン 4.1 までは、インストーラが「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 以降では、論理ボリューム「data」は LVM-thin プールであり、ブロックベースのゲストイメージの保存に使用されます。また、`/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

このような設定には、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、ディスクの交換が容易になります（ホットプラグ対応）。

LVM自体には特別なハードウェアは必要なく、メモリ要件も非常に低い。

## 3.8.2 ブートローダー

デフォルトでは 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の GRUB ブートローダーが含まれます。2 番目のパーティションは EFI システムパーティション (ESP) であり、EFI システムでの起動と、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` があり、そこに「vmdata」というボリュームグループを作成するとします。



### 注意

以下のコマンドは `/dev/sdb` 上の既存データをすべて破壊します。

まずパーティションを作成します。

```
# 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 に設定します。  
  
# pvcreate --metadatasize 250k -y -ff /dev/sdb1  
  
/dev/sdb1 に「vmdata」というボリュームグループを作成します#  
vgcreate vmdata /dev/sdb1
```

### 3.8.4 /var/lib/vz用に追加のLVを作成

これは、新しいシンLVを作成することで簡単に実現できます。

```
# lvcreate -n &lt;名前&gt; -V &lt;サイズ[M,G,T]&gt; &lt;VG&gt;/&lt;LVThin_pool&gt;
```

実例:

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

次に、LV上にファイルシステムを作成する必要があります。

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

最後にマウントします。



#### 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空ではありません。

常にアクセス可能にするには、/etc/fstab に次の行を追加してください。

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

### 3.8.5 シンプルのサイズ変更

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します：

```
# lvresize --size +&lt;size[\M,G,T]&gt; --poolmetadatasize +&lt;size[\M,G]&gt;<-->  
VG&gt;/&lt;LVThin_pool&gt;
```

---

#### 注記

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ります。

---

### 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についてはセクションLVMを参照してください。

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

## 3.9 Linux上のZFS

ZFSはSun Microsystemsが設計したファイルシステムと論理ボリュームマネージャを統合したものです。Proxmox VE 3.4以降、ZFSファイルシステムのネイティブLinuxカーネル移植版がオプション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肢ともなりました。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ています。

ZFSを利用することで、低予算のハードウェアでエンタープライズレベルの機能を最大限に実現できるだけでなく、SSDキャッシュやSSDのみの構成を活用することで高性能システムも構築可能です。ZFSは、適度なCPUとメモリ負荷、そして容易な管理性を兼ね備え、高コストなハードウェアRAID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 GUIおよびCLIによる簡単な設定と管理。
- 信頼性
- データ破損に対する保護
- ファイルシステムレベルでのデータ圧縮
- スナップショット
- コピー・オン・ライト・クローン
- 各種RAIDレベル：RAID0、RAID1、RAID10、RAIDZ-1、RAIDZ-2、RAIDZ-3、dRAID、dRAID2、dRAID3
- SSDをキャッシュとして使用可能
- 自己修復
- 繙続的な整合性チェック
- 大容量ストレージ向けに設計
- 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ため、最低8GBは必要です。実際には、ハードウェアや予算の範囲内で可能な限り多くのメモリを使用してください。データ破損を防ぐため、高品質なECC RAMの使用を推奨します。

専用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を採用すべきです。これにより全体的なパフォーマンスが大幅に向上がります。

**重要**

独自のキャッシング管理機能を持つハードウェアRAIDコントローラ上で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アダプタや「IT」モードでフラッシュされたLSIコントローラなどがより適しています。

VM内のProxmox VEインストール（ネスト型仮想化）を実験する場合、ZFSがサポートしていないため、その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」とも呼ばれます。このボリュームの容量は、すべてのディスクの容量の合計となります。ただし、RAID0は冗長性を一切提供しないため、1台のドライブが故障するとボリューム全体が使用不能になります。
RAID1	「ミラーリング」とも呼ばれます。データは全てのディスクに同一の内容が書き込まれます。このモードには、同じサイズのディスクが最低2台必要です。結果として得られる容量は單一ディスクの容量となります。
RAID10	RAID0とRAID1の組み合わせ。最低4台のディスクが必要。RAIDZ-1 RAID-5の変種、シングルパリティ。最低3台のディスクが必要。RAIDZ-2 RAID-5のパリエーションで、二重パリティを採用。最低4台のディスクが必要。RAIDZ-3 RAID-5のパリエーションで、三重パリティを採用。最低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イメージを保存するための別のサブボリューム「`rpool/data`」が作成されます。Proxmox VEツールでこれを使用するために、インストーラーは

```
zfspool: local-zfs
        pool rpool/data sparse
        content images,rootdir
```

/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プールの状態を確認できます：

```
# zpool status pool:
  rpool
状态: ONLINE
  scan: none requested config:
```

NAME	状態	READ	書き込み	CKSUM
rpool	オンライン	0	0	0
mirror-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
mirror-1	オンライン	0	0	0
sdc	オンライン	0	0	0
sdd	ONLINE	0	0	0

エラー：既知のデータエラーはありません

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

# zfs list	NAME	使用済み	利用可能	REFER	MOUNTPOINT
	rpool	4.94G	7.68T	96K	/rpool
	rpool/ROOT	702M	7.68T	96K	/rpool/ROOT
	rpool/ROOT/pve-1	702M	7.68T	702M	/
	rpool/data	96K	7.68T	96K	/rpool/data
	rpool/swap	4.25G	7.69T	64K	-

### 3.9.3 ZFS RAID レベルの考慮事項

ZFS プールのレイアウトを選択する際には、いくつかの要素を考慮する必要があります。ZFS プールの基本構成要素は仮想デバイス (`vdev`) です。プール内のすべての `vdev` は均等に使用され、データはそれらにストライピングされます (RAID0)。`vdev` の詳細については、[zpoolconcepts\(7\)](#) マニュアルページを参照してください。

#### パフォーマンス

各 `vdev` タイプは異なるパフォーマンス特性を示します。注目すべき2つのパラメータは、IOPS (1秒あたりの入出力操作数) と、データの書き込みまたは読み取りが可能な帯域幅です。

ミラー `vdev` (RAID1) は、データ書き込み時において両パラメータの挙動が単一ディスクと同等となります。データ読み取り時の性能は、ミラー構成のディスク数に比例して直線的に向上します。

一般的な構成として4台のディスクを使用する場合、2つのミラー `vdev` (RAID10) として設定すると、プールは書き込み時において IOPS と帯域幅の両面で2台の単一ディスクと同等の特性を示します。読み取り操作では4台の単一ディスクに類似した挙動となります。

あらゆる冗長性レベルの RAIDZ は、IOPS に関しては単一ディスクと同等の動作を示し、帯域幅は大幅に増加します。帯域幅の増加量は、RAIDZ `vdev` のサイズと冗長性レベルに依存します。

dRAID プールは、同等の RAIDZ プールと同等のパフォーマンスを発揮するはずです。稼働中の VMにおいては、ほとんどの状況で IOPS がより重要な指標となります。

#### サイズ、使用容量、冗長性

ミラー `vdev` で構成されたプールは最高のパフォーマンス特性を示す一方、使用可能領域は利用可能なディスクの50%となります。ミラー `vdev` が2枚以上のディスクで構成される場合 (例: 3ウェイミラー)、使用可能領域はさらに減少します。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枚の健全なディスクが必要です。

N台のディスクで構成されるRAIDZタイプのvdevの有効使用可能領域は、おおむねN-Pとなります。ここでPはRAIDZレベルを表します。RAIDZレベルは、データを失わずに障害が発生できるディスクの数を示します。RAIDZ2を使用した4台ディスクのプールは特殊なケースです。この状況では、有効使用可能領域が同じになるため、パフォーマンス向上のために2つのミラーvdevを使用する方が通常は優れています。

いずれのRAIDZレベルを使用する場合でも、VMディスクに用いられるZVOLデータセットの挙動が重要な要素となる。各データブロックに対し、プールは少なくともプール定義のashift値で指定された最小ブロックサイズに相当するパリティデータを必要とする。ashiftが12の場合、プールのブロックサイズは4kとなる。ZVOLのデフォルトブロックサイズは8kである。したがって、RAIDZ2では8kブロックを書き込むごとに、追加で2つの4kパリティブロックが書き込まれます（ $8k + 4k + 4k = 16k$ ）。これは簡略化した説明であり、実際の状況ではメタデータや圧縮処理などが考慮されていないため、多少異なる場合があります。

この挙動は、ZVOLの以下のプロパティを確認することで観察できます：

- volsize
- refreservation (プールがシンプロビジョニングされていない場合)
- used (プールがシンプロビジョニングで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に提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想スペースを含むプール上の予約済み領域を示します。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。この挙動を確認する別の方法は、VM内の使用済みディスク領域とusedプロパティを比較することです。スナップショットは値を歪める可能性があることに注意してください。

増加したスペース使用に対処する方法はいくつかあります：

- volblocksize を増やしてデータとパリティの比率を改善する
- RAIDZの代わりにミラーvdevを使用する
- ashift=9 (ブロックサイズ512バイト) を使用する

volblocksizeプロパティはZVOL作成時のみ設定可能。デフォルト値はストレージ構成で変更できる。この場合、ゲスト側のチューニングが必要となり、ユースケースによっては書き込み増幅の問題がZFS層からゲスト側に移行するだけとなる。

プール作成時にashift=9を使用すると、基盤となるディスクによってはパフォーマンスが低下する可能性があり、後から変更することはできません。

ミラー vdev (RAID1、RAID10) は VM ワークロードに対して良好な動作特性を示します。環境が特定の要件や特性（RAIDZ の性能特性が許容される場合）を有していない限り、これらを使用してください。

### 3.9.4 ZFS dRAID

ZFS dRAID（分散型RAID）では、ホットスペアドライブがRAIDに参加します。これらのスペア容量は確保され、ドライブ障害発生時に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障害時のRAIDZと比較して高速な再構築が実現されます。詳細は公式OpenZFSドキュメントを参照してください。<sup>2</sup>

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

---

#### 注記

dRAIDは10~15台以上のディスク構成を想定しています。ディスク数が少ない場合、ほとんどのユースケースではRAIDZ構成の方が適しています。

---

#### 注

GUIは最小構成より1台多いディスクを必要とします（例：dRAID1には3台）。スペアディスクの追加も想定されています。

---

- dRAID1 または dRAID: 少なくとも2台のディスクが必要。1台が故障してもデータは失われない
- dRAID2: 少なくとも3台のディスクが必要。2台まで故障してもデータは失われない
- dRAID3: 少なくとも4台のディスクが必要、3台まで故障してもデータは失われない

詳細情報はマニュアルページを参照:

```
# man zpoolconcepts
```

#### スペアとデータ

スペアの数は、ディスク障害時にシステムが待機状態に保つべきディスクの数を示します。デフォルト値はスペア0です。スペアがない場合、再構築の速度向上効果は得られません。

data は冗長性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 - スペアの合計が 8 未満の場合を除き、より少ない数値が使用されます。一般的に、データデバイスの数が少ないほど IOPS が高くなり、圧縮率も向上し、リシルバリングも高速化されますが、データデバイス数を少なく定義するとプールの利用可能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はブートローダー設定の管理に[proxmox-boot-tool](#)を使用します。詳細は[Proxmox VEホストブートローダー](#)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いくつか紹介します。ZFS自体は非常に強力で、多くのオプションを提供します。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優れたマニュアルページが付属しており、以下のようにして読むことができます：

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

## 新しい zpool の作成

新しいプールを作成するには、少なくとも1台のディスクが必要です。ashift は、基盤となるディスクと同じセクタサイズ (ashift の2乗) 以上である必要があります。

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

### ヒント

プール名は以下の規則に従う必要があります：

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

圧縮を有効にするには ([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

最低1台のディスク

```
# zpool create -f -o ashift=12 <pool> <device1> <device2>;
```

### RAID-1で新しいプールを作成する

最低2台のディスク

```
# zpool create -f -o ashift=12 <pool> mirror <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台のディスク

```
# zpool create -f -o ashift=12 <pool> mirror <device1> <device2> mirror<↔> <device3> <device4>;
```

### RAIDZ-1で新しいプールを作成

最低3台のディスク

```
# zpool create -f -o ashift=12 <pool> raidz1 <device1> <device2> <device3>;
```

## RAIDZ-2で新しいプールを作成する

最低4台のディスク

```
# zpool create -f -o ashift=12 <pool> raidz2 <device1> <device2> <device3> <device4>;
```

プール設定前、特にRAID-Zモードを使用する場合、IOPSと帯域幅の期待値を概算するには、[ZFS RAIDレベル](#)に関する考慮事項のセクションを参照してください。

## RAIDZ-Nの拡張

### 注意

この機能はProxmox VE 9 (ZFS 2.3.3) 以降でのみ動作します。

既存の <pool> に RAIDZ-N <raidzN-M> vdev が存在する場合、以下の構文で新しい物理ディスク <device> を追加できます:

```
zpool attach <pool> <raidzN-M> <device>;
```

zpool status <pool> を実行すると一般的な成功を確認でき、zpool list <pool> -v を実行すると接続された全ディスクの詳細出力が得られます。プールの新たな容量を確認するには zfs list <pool> を実行してください。

## キャッシュ付きの新規プール作成 (L2ARC)

パフォーマンス向上のため、専用デバイスまたはパーティションをセカンドレベルキャッシュとして使用できます。このようなキャッシュデバイスは、主に静的なデータのランダム読み取りワークロードに特に効果的です。実際のストレージとメモリ内ARCの間に追加のキャッシュ層として機能するため、メモリ制約によりARCを縮小する必要がある場合にも役立ちます。

## ディスク上キャッシュ付きZFSプールを作成

```
# 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> のみを使用していますが、[RAID を使用した新規プール作成](#)で示されているように、複数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リングやRAIDモードが存在せず、すべて単純に蓄積されることに注意してください。

いずれかのキャッシュデバイスが読み取り時にエラーを発生させた場合、ZFSはその要求を透過的に基盤となるストレージ層へ転送します。

## ログ付き (ZIL) の新しいプールを作成する

ZFS インテントログ (ZIL) には、専用のドライブまたはパーティションを使用することができます。これは主に、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ます。そのため、データベースや、fsync 操作をより頻繁に発行するその他のプログラムなど、パフォーマンスが重要なパスでよく使用されます。

プールはデフォルトの ZIL 場所として使用され、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レイテンシを低減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を向上させることができます。

ログデバイスとして直接またはパーティション経由で使用するディスクについては、以下の設定を推奨します：

- 電源喪失保護機能付きの高速SSDを使用してください。これらはコミットレイテンシが大幅に小さくなります。
- 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を使用してください。ただし、インストール済みメモリの半分以上を使用しても実質的な利点は得られません。

#### 別個のログデバイスを持つZFSプールを作成

```
# 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>`を使用していますが、「[RAIDを使用した新規プール作成](#)」セクションで説明されているように、他のRAIDパリエーションと組み合わせることも可能です。

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可能です。これは主に、単一のログデバイスが故障した場合でもパフォーマンスが即座に低下しないようにするために有用です。

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。

#### 既存プールへのキャッシュとログの追加

キャッシュとログを持たないプールであっても、いつでも両方またはいずれか一方を追加できます。

例えば、電源喪失保護機能付きの優れたエンタープライズSSDを入手し、プール全体のパフォーマンス向上に活用したいと仮定しましょう。

ログデバイスの最大サイズはインストール済み物理メモリの約半分であるべきため、ZILがSSDの比較的小さな領域しか使用せず、残りのスペースをキャッシュとして利用できることを意味します。

まず、parted または gdisk を使用して SSD 上に 2 つの GPT パーティションを作成する必要があります。その後、

それらをプールに追加する準備が整います：

#### 既存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を追加

```
# zpool add -f <pool> log <device-part1> cache <device-part2>;<pool>、<device-part1>、<device-part2> をプール名と2つのデバイス名に置き換えてください  
パーティションへの /dev/disk/by-id/パス。ZILとキャッシュを  
個別に追加することも可能です。
```

#### 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 <pool> <old-device> <new-device>;
```

## 起動用デバイスの故障時の交換

Proxmox VEのインストール方法に応じて、`systemd-boot`または`proxmox-boot`経由のGRUBが使用されています。

<sup>3</sup> またはブートローダーとしてプレーンGRUBを使用（[ホストブートローダー](#)を参照）。以下のコマンドを実行して確認できます：

```
# 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パーティションの置換といった最初のステップは共通です。新しいディスクからシステムを起動可能にするには、使用中のブートローダーに応じて異なる手順が必要です。

```
# sgdisk <正常な起動可能デバイス> -R <新しいデバイス> # sgdisk -G  
&lt;新しいデバイス&gt;;  
# zpool replace -f <プール> <古いZFSパーティション> <新しいZFSパーティション>;
```

### 注記

新しいディスクのリシリバー処理の進捗状況を監視するには、`zpool status -v` コマンドを使用します。

### proxmox-boot-tool を使用する場合:

```
# proxmox-boot-tool format <新規ディスクのESP>;  
# proxmox-boot-tool init <新規ディスクのESP> [grub]
```

### 注記

ESPはEFIシステムパーティションの略称であり、Proxmox VEインストーラーバージョン5.4以降では、起動ディスク上でパーティション#2として設定されます。詳細は「[同期ESPとして使用する新規パーティションの設定](#)」を参照してください。

### プレーンGRUBの場合:

```
# grub-install <new disk>;
```

### 注意

プレーンGRUBは、Proxmox VE 6.3以前でインストールされたシステムでのみ使用され、まだ手動で`proxmox-boot-tool`への移行が行われていないものです。

<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

### 3.9.7 メール通知の設定

ZFSにはイベントデーモンZEDが付属しており、ZFSカーネルモジュールが生成するイベントを監視します。このデーモンはブルエラーなどのZFSイベント発生時にメールを送信することも可能です。新しいZFSパッケージでは、このデーモンは別個のzfs-zed/パッケージとして提供されており、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。

お好みのエディタで /etc/zfs/zed.d/zed.rc ファイルを介してデーモンを設定できます。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。

```
ZED_EMAIL_ADDR="root"
```

Proxmox VEはroot宛てのメールを、rootユーザー用に設定されたメールアドレスに転送します。

### 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% を適応型置換キャッシュ (ARC) に使用します。Proxmox VE 8.1 以降の新しいインストールでは、ARC 使用制限はインストール済み物理メモリの 10% に設定され、最大 16 GiB に制限されます。この値は /etc/modprobe.d/zfs.conf に書き込まれます。

IOパフォーマンスにはARCへの十分なメモリ割り当てが不可欠であるため、削減は慎重に行ってください。一般的な目安として、少なくとも2 GiBのベースメモリ+ストレージ容量1 TiBあたり1 GiBを割り当てます。例えば、利用可能ストレージ容量が8 TiBのプールでは、ARC用に10 GiBのメモリを使用すべきです。

ZFSは64 MiBの最小値も強制します。

現在のブートにおけるARC使用制限を変更するには（再起動するとこの変更はリセットされます）、

zfs\_arc\_max モジュールパラメータに直接書き込むことで変更できます:

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

ARC制限を恒久的に変更するには、/etc/modprobe.d/zfs.d/zfs\_modprobe.d/zfs\_modprobe.d/zfs\_modprobe.d/zfs\_modprobe.d/z  
options zfs zfs\_arc\_max=8589934592

この設定例では、使用量を 8 GiB ( $8 \times 2^{30}$ ) に制限します。

**重要**

zfs\_arc\_max の設定値が 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 を zfs\_arc\_max - 1 以下に設定しない限り、zfs\_arc\_max は無視されます。

```
echo "$[8 * 1024*1024(* 1024 - 1]" &ampgt /sys/module/zfs/parameters/zfs_arc_minecho "$[8 (* 1024(* 1024(* 1024]" &ampgt /sys/module/zfs/parameters/zfs_arc_max
```

この設定例は、合計メモリが 256 GiB を超えるシステムで、zfs\_arc\_max だけを設定しても機能しない場合に、使用量を 8 GiB ( $8 \times 2^{30}$ ) に（一時的に）制限します。

**重要**

ルートファイルシステムがZFSの場合、この値を変更するたびにinitramfsを更新する必要があります:

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

これらの変更を有効にするには再起動が必要です。

### 3.9.9 ZFS上のスワップ

zvol上に作成されたスワップ領域は、サーバーの停止や高いIO負荷の発生など、外部ストレージへのバックアップ開始時に頻繁に見られる問題を引き起こす可能性があります。

十分なメモリを使用し、通常はメモリ不足に陥らないことを強く推奨します。スワップを追加する必要がある場合、物理ディスク上にパーティションを作成し、それをスワップデバイスとして使用することを推奨します。インストーラーの詳細オプションで、この目的のために空き領域を確保できます。さらに、「swappiness」値を下げるることができます。サーバーに適した値は10です:

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

スワップ優先度を永続化するには、任意のエディタで `/etc/sysctl.conf` を開き、以下の行を追加します：

```
vm.swappiness= 10
```

表3.3: Linuxカーネルスワッピングパラメータの値

値	戦略
<code>vm.swappiness= 0</code>	カーネルはメモリ不足状態を回避するためだけにスワップを行う
<code>vm.swappiness= 1</code>	完全に無効化せずに最小限のスワッピング量を設定します。
<code>vm.swappiness= 10</code>	システムに十分なメモリがある場合に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ります システムに十分なメモリが存在する場合。
<code>vm.swappiness= 60</code>	デフォルト値。
<code>vm.swappiness= 100</code>	カーネルは積極的にスワップを行います。

### 3.9.10 暗号化された ZFS データセット

#### 警告



Proxmox VEにおけるネイティブZFS暗号化は実験段階です。既知の制限事項および問題点には、暗号化データセットを用いたレプリケーション<sup>a</sup>、ならびにスナップショットまたはZVOL使用時のチェックサムエラー<sup>b</sup>が含まれます。

<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>

Linux版ZFS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サポートが導入されました。以前のLinux版ZFSからのアップグレード後、暗号化機能はプール単位で有効化できます：

```
# zpool get feature@encryption tank
  NAME    プロパティ          値          ソース
  tank   feature@encryption  無効          ローカル

# zpool set feature@encryption=enabled

# zpool get feature@encryption tank
  NAME    プロパティ          VALUE        ソース
  tank   feature@encryption  enabled      local
```

**警告**

現在、GRUBを使用した暗号化データセットを含むプールからのブートはサポートされておらず、ブート時の暗号化データセットの自動ロック解除も限定的なサポートのみです。暗号化をサポートし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できません。

**注記**

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、起動時にロック解除に必要な鍵情報を `zfs load-key` に渡すカスタムユニットを作成することを推奨します。

**警告**

本番データの暗号化を有効化する前に、バックアップ手順を確立しテストしてください。関連する鍵素材/パスフレーズ/キーファイルが紛失した場合、暗号化されたデータへのアクセスは不可能になります。

暗号化はデータセット /zvol 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。例として、暗号化データセット `tank/encrypted_data` を作成し、Proxmox VE で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：

```
# zfs create -o encryption=on -o keyformat=passphrase tank/encrypted_data Enter passphrase:  
Re-enter passphrase:
```

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

このストレージ上に作成される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素材で暗号化されます。

ストレージを実際に使用するには、関連する鍵素材をロードし、データセットをマウントする必要があります。これは次の1つのステップで実行できます：

```
# zfs mount -l tank/encrypted_data  
'tank/encrypted_data' のパスフレーズを入力してください:
```

パスフレーズの入力を促す代わりに（ランダムな）キーファイルを使用することも可能です。これには、キーロケーション

およびキーフォーマットプロパティを設定することで、パスフレーズの入力を促す代わりに（ランダムな）キーファイルを使用することも可能です。これは作成時、または既存のデータセットに対して `zfs change-key` コマンドで設定できます：

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

```
# zfs change-key -o keyformat=raw -o keylocation=file:///path/to/keyfile -->  
tank/encrypted_data
```

**警告**

キーファイルを使用する際は、不正アクセスや偶発的な紛失から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不可能です！

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、その暗号化ルートプロパティが適切に設定されます。暗号化ルートごとにキー素材を一度ロードするだけで、その下にあるすべての暗号化データセットで利用可能になります。

詳細および高度な使用法については、`encryptionroot`、`encryption`、`keylocation`、`keyformat`、`keystatus` プロパティ、`zfs load-key`、`zfs unload-key`、`zfs change-key` コマンド、および `man zfs` の「暗号化」セクションを参照してください。

### 3.9.11 ZFSにおける圧縮

データセットで圧縮が有効化されている場合、ZFSは書き込み前に新規ブロックを圧縮し、読み取り時に復元します。既存データは遡及的に圧縮されません。

圧縮を有効にするには次のコマンドを実行します：

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

CPU負荷が非常に少ないため、lz4アルゴリズムの使用を推奨します。`lzjb` や `gzip-N` (`N` は 1 (最速) から 9 (最高圧縮率) までの整数) などの他のアルゴリズムも利用可能です。アルゴリズムとデータの圧縮率によっては、圧縮を有効にすることで I/O パフォーマンスが向上することもあります。

圧縮は無効化可能です：

```
# zfs set compression=off <dataset>;
```

繰り返しますが、この変更の影響を受けるのは新規ブロックのみです。

### 3.9.12 ZFS 特殊デバイス

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ます。プール内の特殊デバイスは、メタデータ、重複排除テーブル、およびオプションで小規模ファイルブロックの保存に使用されます。

メタデータの変更が頻繁な低速回転ハードディスクで構成されるプールでは、特殊デバイスにより速度向上が期待できます。例えば、大量のファイル作成・更新・削除を伴うワークロードでは、特殊デバイスの存在が効果を発揮します。ZFS データセットを構成し、小規模ファイル全体を特殊デバイスに保存させることで、さらなるパフォーマンス向上が可能です。特殊デバイスには高速 SSD を使用してください。



#### 重要

特別なデバイスの冗長性はプールと一致させる必要があります。特別なデバイスはプール全体の障害点となるためです。



#### 警告

プールへの特殊デバイスの追加は取り消せません！

**特殊デバイスとRAID-1でプールを作成:**

```
# zpool create -f -o ashift=12 <pool> mirror <device1> <device2> special<->mirror <device3> <device4>
```

**既存のプールに RAID-1 で特殊デバイスを追加:**

```
# 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ます。sizeは0（特殊デバイスへの小ファイルブロック保存を無効化）または512B～1Mの範囲の2の幂乗値を設定可能です。プロパティ設定後、size未満の新しいファイルブロックは特殊デバイス上に割り当てられます。

**重要**

special\_small\_blocks の値がレコードサイズ（デフォルト128K）以上の場合、すべてのデータが特殊デバイスに書き込まれるため、注意が必要です！

プールでspecial\_small\_blocksプロパティを設定すると、そのプロパティのデフォルト値がすべての子ZFSデータセットに適用されます（例：プール内のすべてのコンテナが小ファイルブロックを有効化）。

**プール全体で4Kブロック未満の全ファイルに適用:**

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

**単一データセットでの小ファイルブロックの有効化:**

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

**単一データセットで小ファイルブロックを無効化:**

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

### 3.9.13 ZFS プール機能

ZFSにおけるディスク上のフォーマットの変更は、メジャーバージョン変更時のみ行われ、機能を通じて指定されます。すべての機能および一般的なメカニズムは、zpool-features ( manページ)に詳細に記載されています。

新機能の有効化により、古いバージョンのZFSではプールをインポートできなくなる可能性があるため、管理者がプールに対してzpool upgradeを実行して明示的に行う必要があります (zpool-upgrade (8) マニュアルページを参照)。

新機能の1つを利用する必要がない限り、それらを有効化することにはメリットがありません。実際、新機能を有効化することにはいくつかのデメリットがあります:

- GRUBを使用して起動するシステムで、ルートパーティションがZFS上に存在する場合、rpool上で新機能が有効化されると、GRUBにおけるZFS実装の不整合により起動不能になります。
- 古いカーネル（旧ZFSモジュールを搭載）で起動した場合、アップグレードされたプールをインポートできません。
- 同様に、起動不能なシステムを修復するために古いProxmox VE ISOから起動することも機能しません。

**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。これによりシステムが起動不能になります。これにはProxmox VE 5.4以前にインストールされたシステム、およびレガシ BIOSブートで起動するシステムが含まれます（[ブートローダーの確認方法参照](#)）。

**ZFS プールの新機能有効化:**

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

## 3.10 BTRFS

**警告**

BTRFSの統合は現在、Proxmox VEにおける**技術プレビュー**段階です。

BTRFS は、スナップショット、組み込み RAID、データおよびメタデータのチェックサムによる自己修復などの機能を実装した、Linux カーネルがネイティブで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す。Proxmox VE 7.0 以降、BTRFS はルートファイルシステムのオプションとして導入されています。

**BTRFS の一般的な利点**

- 従来の ext4 ベースのセットアップとほぼ同じメインシステム設定
- スナップショット
- 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
- データ破損に対する保護
- 自己修復
- Linuxカーネルによってネイティブにサポートされています

**注意点**

- RAIDレベル5/6は実験的かつ危険です。[詳細はBTRFSステータスを参照](#)

### 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インストーラー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」とも呼ばれます。このボリュームの容量は、すべてのディスクの容量の合計となります。ただし、RAID0は冗長性を一切提供しないため、1台のドライブが故障するとボリューム全体が使用不能になります。
RAID1	「ミラーリング」とも呼ばれます。データは全てのディスクに同一の内容が書き込まれます。このモードには、同じサイズのディスクが最低2台必要です。結果として得られる容量は單一ディスクの容量となります。
RAID10	RAID0とRAID1の組み合わせ。最低4枚のディスクが必要。

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

Proxmox VE ツールで使用するために、インストーラーは次の設定エントリを

`/etc/pve/storage.cfg` に以下の設定エントリを作成します:

```
dir: local
    path /var/lib/vz
    content iso,vztmp,backup disable

btrfs: local-btrfs
    path /var/lib/pve/local-btrfs
    content iso,vztmp,backup,images,rootdir
```

これにより、追加サブボリューム上の BTRFS 専用ストレージエントリを優先して、デフォルトのローカルストレージが明示的に無効化されます。

btrfs コマンドは BTRFS ファイルシステムの構成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のサブボリュームをすべて一覧表示します:

```
# btrfs subvolume list /
ID 256 gen 6 top level 5 path var/lib/pve/local-btrfs
```

### 3.10.2 BTRFS 管理

このセクションでは、一般的なタスクの使用例をいくつか紹介します。

#### 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` が使用されます。`-d` および `-m` パラメータは、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ために使用されます。オプションの `-L` パラメータを使用すると、ラベルを設定できます。

一般的に以下のモードがサポートされています：`single`、`raid0`、`raid1`、`raid10`。単一ディスク `/dev/sdb` にラベル

`My-Storage` を持つ BTRFS ファイルシステムを作成します：

```
# mkfs.btrfs -m single -d single -L My-Storage /dev/sdb
```

または、2つのパーティション `/dev/sdb1` と `/dev/sdc1` で RAID1 を構築します:

```
# 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手動でマウントできます。例:

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

BTRFSは他のマウントポイントと同様に/etc/fstabに追加でき、起動時に自動マウントされます。特にBTRFS構成で複数のディスクを使用する場合、プロックデバイスパスではなくmkfs.btrfsコマンドが実行したUUID値を使用することが推奨されます。

例:

### ファイル /etc/fstab

```
# ... 簡略化のため他のマウントポイントは省略  
  
# mkfs.btrfs 出力の UUID を使用することを強く推奨します UUID=e2c0c3ff-2114-4f54-b767-3a203e49f6f3 /my-storage  
btrfs defaults 0 0
```

### ヒント

UUIDが利用できない場合は、blkidツールを使用してブロックデバイスの全プロパティを一覧表示できます。

その後、最初のマウントを実行するには以下を実行します:

```
mount /my-storage
```

次の再起動後は、システムが起動時に自動的に実行します。

## 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インターフェースまたはCLIを使用できます。例:

```
pvesm add btrfs my-storage --path /my-storage
```

### サブボリュームの作成

サブボリュームを作成すると、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します。

### 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異なり、サブボリュームはbtrfsコマンドで削除する際に空である必要があります。

```
# btrfs subvolume delete /some/path
```

## サブボリュームのスナップショット作成

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サブボリュームの任意のコピーを作成することを見なせます。慣例として、Proxmox VEはゲストディスクやサブボリュームのスナップショット作成時に読み取り専用フラグを使用しますが、このフラグは後から変更

```
# btrfs subvolume snapshot -r /some/path /a/new/path  
することも可能です。
```

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 に作成されます。/some/path に対する今後の変更は、変更前に変更されたデータがコピーされることを意味します。

読み取り専用 (-r) オプションを省略した場合、両方のサブボリュームは書き込み可能になります。

## 圧縮の有効化

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できます。既に書き込まれたデータは事後に圧縮されないことに注意してください。

デフォルトでは、ルートファイルシステムは /etc/fstab に以下のように記述されます：

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

デフォルト設定に compress=zstd、compress=lzo、または compress=zlib を追加するだけで済みます。

上記のように追加します：

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

この変更是再起動後に有効になります。

## 使用容量の確認

従来のdfツールは、一部のBTRFS設定で混乱を招く値を出力する場合があります。より正確な推定には

btrfs filesystem usage /PATH コマンドを使用してください。例:

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

## 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、ノード固有の設定やリソースを制御できます。

現在、pvenode ではノードの説明設定、ノード上のゲストに対する各種一括操作の実行、ノードのタスク履歴の表示、API および pveproxy 経由の Web GUI で使用されるノードの SSL 証明書の管理が可能です。

### 3.11.1 Wake-on-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上のスリープ状態のコンピュータを起動させます。少なくとも1つのNICがこの機能をサポートしている必要があります、対応するオプションがコンピュータのファームウェア (BIOS/UEFI) 設定で有効になっている必要があります。オプション名は「Enable Wake-on-Lan」から「Power On By PCIE Device」まで様々です。不明な場合はマザーボードベンダーのマニュアルを確認してください。ethtoolを使用して interface の WoL 設定を確認するには以下を実行します：

```
ethtool <インターフェース> | grep Wake-on
```

pvenode を使用すると、WoL 経由でクラスタの休止状態メンバーを起動できます。以下のコマンドを実行します:

```
pvenode wakeonlan <node>;
```

このコマンドは、wakeonlanプロパティから取得した<node>のMACアドレスを含むWoLマジックパケットをUDPポート9で送信します。ノード固有のwakeonlanプロパティは次のコマンドで設定できます:

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

WoLパケットの送信インターフェースはデフォルトルートから決定されます。以下のコマンドでバインドインターフェースを設定することで上書き可能です:

```
pvenode config set -wakeonlan XX:XX:XX:XX:XX:XX,bind-interface=<iface-name>;
```

WoLパケット送信時に使用するブロードキャストアドレス（デフォルト255.255.255.255）は、以下のコマンドで明示的に設定することで変更可能です:

```
pvenode config set -wakeonlan XX:XX:XX:XX:XX:XX,broadcast-address=<broadcast-address>;
```

### 3.11.2 タスク履歴

サーバーの問題（例：バックアップジョブの失敗）をトラブルシューティングする際、以前に実行され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`コマンドを使用して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、ノードの完了済み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、VM 100に関連するエラーで終了したタスクのリストを取得するには、次のコマンドを実行します:

```
pvenode task list --errors --vmid 100
```

タスクのログは、そのUPIDを使用して出力できます:

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

### 3.11.3 バルクゲスト電源管理

多数のVM/コンテナを管理する場合、ゲストの起動/停止はpvenodeのstartallおよびstopallサブコマンドによる一括操作で実行できます。デフォルトでは、pvenode startallは起動時に自動起動が設定された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](#)を参照）。ただし、--forceフラグでこの動作をオーバーライドできます。両コマンドには--vmsオプションもあり、停止/起動するゲストを指定したVMIDに限定できます。

例えば、onboot設定の有無に関わらずVM 100、101、102を起動するには以下のように実行します:

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

これらのゲスト（および実行中のその他のゲスト）を停止するには、次のコマンドを使用します:

```
pvenode stopall
```

#### 注記

stopall コマンドはまずクリーンなシャットダウンを試行し、その後、すべてのゲストが正常にシャットダウンするか、上書き可能なタイムアウト（デフォルトは 3 分）が経過するまで待機します。その時点で、force-stop パラメータが明示的に 0 (false) に設定されていない場合、まだ実行中のすべての仮想ゲストは強制的に停止されます。

### 3.11.4 最初のゲスト起動遅延

NFS サーバーなど、起動に時間がかかる外部リソースに依存する VM/コンテナを使用している場合、Proxmox VE の起動から、自動起動が設定されている最初の VM/コンテナが起動するまでの間に、ノードごとの遅延を設定することもできます（[仮想マシンの自動起動と自動シャットダウンを参照](#)）。

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

以下の設定で実現できます（10 は秒単位の遅延を表します）：

### 3.11.5 一括ゲスト移行

アップグレード作業でゲストを1つのノードから別のノードへ一括移行する必要がある場合、pvenodeには移行対象を指定する`migrateall`サブコマンドが用意されています。デフォルトではシステム上の全ゲストを移行先ノードに移行しますが、特定のゲストのみを移行するよう設定することも可能です。

例えば、VM 100、101、102 をノード pve2 に移行し、ローカルディスクのライブマイグレーションを有効にするには、以下のように実行します：

```
pvenode migrateall pve2 --vms 100,101,102 --with-local-disks
```

### 3.11.6 バルーニングのRAM使用率目標値

自動メモリ割り当ての目標パーセンテージはデフォルトで80%です。ballooning-targetプロパティを設定することで、ノードごとにこの目標値をカスタマイズできます。例えば、ホストメモリ使用率を90%に設定する場合：

```
pvenode config set --ballooning-target 90
```

## 3.12 証明書管理

### 3.12.1 クラスタ内通信用証明書

各Proxmox VEクラスターはデフォルトで独自の（自己署名型）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用の証明書を生成します。これらの証明書は、クラスターのpveproxyサービスとの暗号化通信、およびSPICEを使用する場合のShell/コンソール機能に使用されます。

CA証明書と鍵は[Proxmoxクラスターファイルシステム \(pmxcfs\)](#) に保存されます。

### 3.12.2 APIおよびWeb GUI用証明書

REST API および Web GUI は、各ノード上で動作する 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ついては、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ーCAによって署名されているため、ブラウザやオペレーティングシステムによって自動的に信頼されることはありません。
2. 外部提供の証明書を使用する（例：商用CAによる署名済み）。
3. ACME (Let's Encrypt) を使用して自動更新機能付きの信頼された証明書を取得する。これはProxmox VE APIおよびWebインターフェースにも統合されています。

オプション2および3では、ファイル/etc/pve/local/pveproxy-ssl.pem (および/etc/pve/local/pveproxy パスワードなしで設定されている必要があります) が使用されます。

#### 注意

/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留意してください。

証明書は、Proxmox VE ノード管理コマンド (pvenode (1) マニュアルページを参照) で管理されます。

#### 警告

! 使用してください。 は使用しないでください。 または を手動で置換しないでください。 を使用しないでください。 を変更しないでください。 は自動的に生成されます。 によって生成されます。 node certificate ファイル は /etc/pve/local/pve-ssl.pem および /etc/pve/local/pve-ssl.key、あるいはクラスタ CA ファイル /etc/pve/pve-root-ca.pem および /etc/pve/priv/pve-root-ca.key に保存されます。

### 3.12.3 カスタム証明書のアップロード

Proxmox VE ノードで使用する証明書が既に用意されている場合は、Web インターフェースからその証明書を簡単にアップロードできます。

Upload Custom Certificate

Private Key (Optional):

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAsmlirAywKeTsZis9rebE9TnVrhnyGR1jT138CADAacmy3QZD
195JybZRRGDTFU4tFivR/yVuOV7KvfiZ5GiIWwtislrqPN2y5LLKZu4H3M/ERkra
A7KtI4f5mHSTYV/CnIFFNnnnMaWtFPkDnFXn3nvzWrFu5Kd0FRfnVnCv7alk1nf1a
-----
```

From File

Certificate Chain:

```
-----BEGIN CERTIFICATE-----
MIIFVDCCBdyAwIBAgISA6NsXuXOXV6ey6TB+Zu1Vpr2MA0GCSqGSIb3DQEBCwUA
MEoxCzAjBgNVBAYTA1VTMRYwFAYDVQQKEw1MZXRQnycBFbmNyeXBOMSMwIjYDVQQD
FvnM7YOnnuRFhmNvaYRnIFF1dGhwmnlhSRVM7AeFm0vNTFwMDfymDI15MTlaFu0v
-----
```

From File

Upload

証明書キーファイルは、提供される場合、パスワード保護されてはいけません。

### 3.12.4 Let's Encrypt (ACME) 経由の信頼済み証明書

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 管理者は、Let's Encryptなどの 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 ブラウザでそのまま受け入れられ、信頼される TLS 証明書を簡単に設定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 \(LE\)](#) の本番環境とステージング環境です。当社の ACME クライアントは、組み込みの Web サーバーを使用した http-01 チャレンジの検証、および `acme.sh` がサポートするすべての 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'). At the bottom is a blue 'Register' button.

各クラスターごとに、使用するエンドポイントでACMEアカウントを登録する必要があります。そのアカウントで使用されるメールアドレスは、ACMEエンドポイントからの更新期限通知などの連絡窓口として機能します。

ACMEアカウントの登録および無効化は、Webインターフェース (Datacenter → ACME) またはpvenodeコマンドラインツールを使用して行えます。

```
pvenode acme account register account-name mail@example.com
```

#### ヒント

レート制限のため、実験時やACMEを初めて使用する場合はLEステージングを利用すべきです。

#### ACMEプラグイン

ACMEプラグインの役割は、あなた（およびあなたが管理するProxmox VEクラスター）がドメインの正当な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盤となる構成要素です。

ACMEプロトコルは様々なタイプのチャレンジを規定しています。例えばhttp-01では、ウェブサーバーが特定のコンテンツを含むファイルを提供し、ドメインを管理していることを証明します。技術的な制約や、レコードのアドレスがパブリックインターネットから到達できない場合など、これが不可能な場合もあります。そのようなケースではdns-01チャレンジが使用できます。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満たされます。

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. There are 'Add', 'Edit', and 'Remove' buttons for both sections.

Proxmox VEはこれらのチャレンジタイプを標準でサポートしています。プラグインの設定は、Webインターフェースの「Datacenter → ACME」から、または`pvenode acme plugin add`コマンドを使用して行えます。

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ー内の全ノードで利用可能です。

#### ノードドメイン

各ドメインはノード固有です。ノード -> 証明書の下で、または pvenode config コマンドを使用して、新しいドメインエントリの追加や既存エントリの管理が可能です。

The dialog box has fields for 'Challenge Type' (set to 'DNS'), 'Plugin' (set to 'pdns-example.com'), and 'Domain' (set to 'prod1.pve.example.com'). It includes 'Help' and 'Create' buttons.

ノードに対して必要なドメインを設定し、目的のACMEアカウントが選択されていることを確認した後、ウェブインターフェースから新規証明書の発行を依頼できます。成功すると、インターフェースは10秒後に再読み込みされます。

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 で起動する組み込み Web サーバーを介して http-01 チャレンジを検証するため、暗黙的に構成されたスタンダードアロンプラグインが常に存在します。

### 注

スタンダロンという名称は、サードパーティサービスなしで単独で検証を提供できることを意味します。したがって、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のACMEによる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。

- アカウント登録には Let's Encrypt の利用規約への同意が必要です。
- 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ります。
- ポート80には他のリスナーが存在してはなりません。
- 要求された（サブ）ドメインはノードのパブリックIPに解決される必要があります。

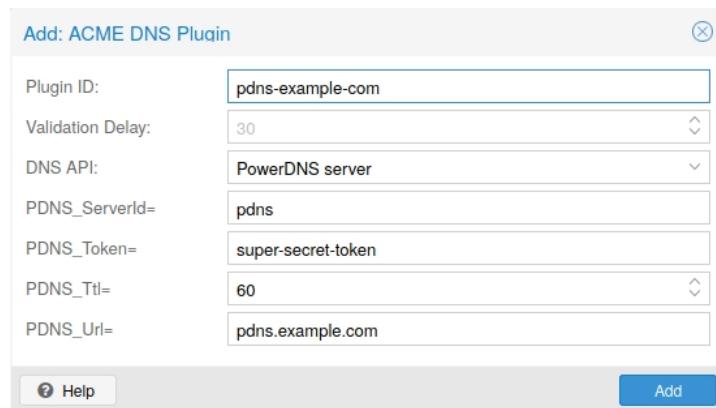
### 3.12.6 ACME DNS API チャレンジプラグイン

http-01 方式による検証のための外部アクセスが不可能または望ましくないシステムでは、dns-01 検証方式を使用することができます。この検証方式では、API 経由で TXT レコードのプロビジョニングを許可する 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<sup>4</sup> プロジェクト用に開発された DNS プラグインを再利用しています。特定の API の設定に関する詳細については、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」を選択します。その後、APIプロバイダーを選択し、そのAPI経由でアカウントにアクセスするための認証データを入力します。検証遅延は、DNSレコードの設定からACMEプロバイダーに検証を促すまでの時間を秒単位で決定します。プロバイダーは、そのインフラストラクチャ内でレコードを伝播させるためにしばしば時間を必要とするためです。

#### ヒント

プロバイダーのAPI認証情報を取得する方法の詳細については、acme.shの「DNS APIの使用方法」Wikiを参照してください。

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

DNSプロバイダーやAPIエンドポイントが多いため、Proxmox VEは一部のプロバイダー向けに認証情報の入力フォームを自動生成します。その他のプロバイダーでは大きなテキストエリアが表示されますので、認証情報のKEY=VALUEペアをすべてそこにコピーしてください。

#### CNAMEエイリアスによるDNS検証

プライマリ/実際の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、別のドメイン/DNSサーバーで検証を処理するために特別なエイリアスマードを使用できます。手動で永続的なCNAMEレコードを設定します：

\_acme-challenge.domain1.example を \_acme-challenge.domain2.example へ指す永続的な CNAME レコードを手動で設定し、対応する acmedomainX キーのエイリアスプロパティを Proxmox VE ノード設定ファイル内で domain2.example に設定してください。これにより、domain2.example の DNS サーバーが domain1.example の全チャレンジを検証できるようになります。

#### プラグインの組み合わせ

ノードが異なる要件／DNSプロビジョニング機能を持つ複数のドメイン経由で到達可能な場合、http-01とdns-01の検証を組み合わせることが可能です。また、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。

#### ヒント

同一サービスを複数ドメイン経由でアクセスすることは複雑性を増すため、可能な限り避けるべきです。

### 3.12.7 ACME証明書の自動更新

ノードが ACME 提供の証明書（pvenode または GUI 経由）で正常に構成されている場合、証明書は pve-daily-update.service によって自動的に更新されます。現在、更新は証明書が既に失効している場合、または今後 30 日以内に失効する場合に試行されます。

#### 注意

短寿命証明書を発行するカスタムディレクトリを使用している場合、再起動後の証明書更新を見逃さないよう、pve-daily-update.timer ユニットのランダム遅延を無効化することを推奨します。

### 3.12.8 pvenode を使用した ACME の例

例: Let's Encrypt 証明書を使用するための pvenode サンプル呼び出し

```
root@proxmox:~# pvenode acme account register defaultmail@example.invalid ディレクトリエンドポイント:  
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/↔  
directory)  
2) カスタム  
選択を入力してください: 1
```

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

上記の利用規約に同意しますか? [y|N] y
...
タスク完了
root@proxmox:~# pvenode config set --acme domains=example.invalidroot@proxmox:~# pvenode acme cert order
ACMEアカウントの詳細を読み込み中ACME注文を発行中
...
ステータスは '有効' です!

全ドメインの検証が完了しました!
...
証明書をダウンロード中
pveproxyの証明書と鍵を設定中pveproxyを再起動中
タスク完了
```

#### 例: ドメイン検証のためのOVH APIの設定

注

アカウント登録の手順は、使用するプラグインに関係なく同じです。ここでは繰り返しません。

注意

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。

まず、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。

```
root@proxmox:~# cat /path/to/api-
tokenOVH_AK=XXXXXXXXXXXXXXXXXXXXOVH_AS=YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
root@proxmox:~# source /path/to/api-token
root@proxmox:~# curl -XPOST -H "X-Ovh-Application: $OVH_AK" -H "Content-type: application/json" \ https://eu.api.ovh.com/1.0/auth/credential -d
'{
    "accessRules": [
        {"method": "GET", "path": "/auth/time"},
        {"method": "GET", "path": "/domain"},  

        {"method": "GET", "path": "/domain/zone/* "},  

        {"method": "GET", "path": "/domain/zone/* /record"},  

        {"method": "POST", "path": "/domain/zone/* /record"},  

        {"method": "POST", "path": "/domain/zone/* /refresh"},  

        {"method": "PUT", "path": "/domain/zone/* /record/"},  

        {"method": "DELETE", "path": "/domain/zone/* /record/* "}
    ]
}'
```

```
{"consumerKey": "Zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz", "state": "→  
pendingValidation", "validationUrl": "https://eu.api.ovh.com/auth/? → credentialToken= →  
AAAAAAAAAAAAAAAAAAAAAAA...AAAAAAA...AAAAAAA...AAAAAAA...AAAAAAA"}
```

(検証URLを開き、指示に従ってアプリケーションキーを →  
アカウント/コンシユーマーキーをリンクするには、検証URLを開き指示に従ってください)

```
root@proxmox:~# echo "OVH_CK=Zzzzzzzzzzzzzzzzzzzzzzzzzzzzz" >> /path/to/ →  
api-token
```

ACMEプラグインを設定できます:

```
root@proxmox:~# pvenode acme plugin add dns example_plugin --api ovh --data →  
/path/to/api_token  
root@proxmox:~# pvenode acme plugin config example_plugin  
  
&#x2502; key &#x2502; value &#x2502;  
  
&#x2502; api &#x2502; ovh &#x2502;  
  
&#x2502; ディレクトリ &#x2502; タイプ &#x2502; OVH_AK=XXXXXXXXXXXXXX  
&#x2502; ; &#x2502; &#x2502; OVH_AS=YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY  
&#x2502; &#x2502; OVH_CK=Zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz &#x2502;  
  
&#x2502; digest &#x2502; 867fcf556363calbea866863093fcab83edf47a1 &#x2502;  
  
&#x2502; plugin &#x2502; example_plugin &#x2502;  
  
&#x2502; type &#x2502; dns &#x2502;
```

ついに、証明書を取得したいドメインを設定し、その証明書を注文できるようになりました:

```
root@proxmox:~# pvenode config set -acmedomain0 example.proxmox.com,plugin= →  
example_plugin  
root@proxmox:~# pvenode acme cert order ACMEアカウントの詳細  
を読み込み中  
ACME注文を発行中  
注文URL: https://acme-staging-v02.api.letsencrypt.org/acme/order →  
/11111111/22222222  
  
'https://acme-staging-v02.api' から認証情報を取得します。 →  
letsencrypt.org/acme/authz-v3/33333333'  
example.proxmox.com の検証が保留中です！  
[2020年4月22日(水) 09:25:30 CEST] OVH エンドポイントを使用: ovh-eu [2020年4月22日(水)  
09:25:30 CEST] 認証を確認中 [2020年4月22日(水) 09:25:30 CEST] コンシユーマーキーは問題ありませ  
ん。  
[2020年4月22日(水) 09:25:31 CEST] レコードを追加中  
[2020年4月22日(水) 09:25:32 CEST] 追加、10 秒間スリープ。TXT レコードを追加: _acme-  
challenge.example.proxmox.com 検証をトリガー  
5秒間待機中
```

ステータスは 'valid' です！

[2020年4月22日(水) 09:25:48 CEST] OVH エンドポイントを使用: ovh-eu [2020年4月22日(水)

09:25:48 CEST] 認証を確認中 [2020年4月22日(水) 09:25:48 CEST] コンシューマーキーは問題ありません。

TXT レコードを削除: \_acme-challenge.example.proxmox.com すべてのドメインが検証されました！

CSR 作成中注文状況の確認

注文の準備が整いました。注文の確定が有効です！

証明書をダウンロード中

pveproxy 証明書とキーの設定 pveproxy を再起動

タスク OK

#### 例: ステージングから通常ACMEディレクトリへの切り替え

アカウントのACMEディレクトリ変更はサポートされていませんが、Proxmox VEは複数アカウントをサポートしているため、本番（信頼済み）ACMEディレクトリをエンドポイントとする新規アカウントを作成できます。ステージングアカウントを無効化して再作成することも可能です。

#### 例: pvenode を使用したデフォルト ACME アカウントのステージングからディレクトリへの変更

```
root@proxmox:~# pvenode acme account deactivate default
アカウントファイルのパスを '/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 ディレクトリエンドポイント:

- 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: 0

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

上記の利用規約に同意しますか? [y|N] y

...

タスク完了

## 3.13 ホストブートローダー

Proxmox VEは、インストーラで選択されたディスク構成に応じて、2種類のブートローダー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でない限り`systemd-boot`が使用されます。その他のすべての展開では標準のGRUBブートローダーが使用されます（これは通常、Debian上にインストールされたシステムにも適用されます）。

### 3.13.1 インストーラーが使用するパーティション構成

Proxmox VEインストーラーは、インストール対象として選択されたすべてのディスク上に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通りです：

- 1 MBのBIOSブートパーティション (gdiskタイプEF02)
- 512 MBのEFIシステムパーティション (ESP、gdiskタイプEF00)
- `hdspace` パラメータで設定された領域、または選択したストレージタイプに使用される残りの領域を跨ぐ第三のパーティション

ZFSをルートファイルシステムとして使用するシステムは、512 MB 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 BIOSシステムおよびセキュアブートが有効なEFIシステムではGRUBが使用され、セキュアブートなしのEFIシステムでは`systemd-boot`が使用されます。いずれもESPを指すようにインストールおよび設定されます。

GRUB (--target i386-pc)は、GRUB で起動する全システムにおいて、選択された全ディスクの 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` による ESP 内容の同期

`proxmox-boot-tool` は、EFIシステムパーティションの内容を適切に構成・同期させる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全てのESPにコピーし、対応するブートローダーを`vfat`フォーマットのESPから起動するよう設定します。ルートファイルシステムとしてZFSを使用する場合、これはGRUBのZFS実装で利用可能なサブセットではなく、ルートプール上のすべてのオプション機能を利用できることを意味します<sup>6</sup>。また、別途小さなブートプールを作成する必要もありません<sup>6</sup>。

冗長性のある構成では、インストーラによって全てのディスクにESPがパーティション分割されます。これにより、最初のブートデバイスが故障した場合や、BIOSが特定のディスクからのみ起動可能な場合でも、システムが確実に起動します。

通常動作中はESPをマウントしたままにしません。これによりシステムクラッシュ時の`vfat`フォーマットESPのファイルシステム破損を防ぎ、プライマリブートデバイス障害時の`/etc/fstab`手動修正が不要になります。

`proxmox-boot-tool` は以下のタスクを処理します：

- フォーマットと新規パーティションの設定
- リストされた全てのESPに新しいカーネルイメージと`initrd`イメージをコピーし設定する
- カーネルアップグレード時やその他のメンテナンスタスクにおける設定の同期
- 同期対象のカーネルバージョンリストの管理

<sup>5</sup> これらはすべて、ルートを`ext4`または`xfs`にインストールしたもの、および非EFIシステムでルートをZFSにインストールしたものです。

<sup>6</sup> ブート ZFS ZFS root で GRUB <https://openzfs.github.io/openzfs-docs/Getting%20Started/Debian-Debian%20Bookworm%20Root%20on%20ZFS.html>

- 特定のカーネルバージョンを起動するようブートローダーを設定する（ピンニング）現在設定されている

ESPとその状態は次のコマンドで確認できます:

```
# 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ESPとしてフォーマットおよび初期化するには、例えばrpool内の故障したvdevを交換した後や、同期メカニズム導入前の既存システムを変換する場合などに、proxmox-kernelのproxmox-boot-toolを使用できます。



##### 警告

format コマンドは <partition> をフォーマットします。正しいデバイス/パーティションを指定してください！

例: 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、以下を実行します：

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

既存のアンマウント済みESP (/dev/sda2) をProxmox VEのカーネル更新同期機構に組み込むには、以下を実行します：

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

または

```
# proxmox-boot-tool init /dev/sda2 grub
```

systemd-boot ではなく GRUB による初期化を強制するため（例：セキュアブート対応のため）。

その後、/etc/kernel/proxmox-boot-uuids には新たに追加されたパーティションの UUID が記載された行が追加されるはずです。この init コマンドは、設定済みのすべての ESP の自動更新も同時にトリガーします。

#### すべてのESPで設定を更新する

すべての起動可能なカーネルをコピーして設定し、/etc/kernel/proxmox-boot-uuids にリストされているすべての ESP を維持します。

同期するには、次のコマンドを実行するだけです:

```
# proxmox-boot-tool refresh
```

(ルートファイルシステムがext4またはxfsの場合、`update-grub`を実行するのと同等の処理です)。

カーネルコマンドラインを変更した場合、または全てのカーネルとinitrdを同期させたい場合に必要です。

---

#### 注

update-initramfs および apt (必要な場合) は、自動的にリフレッシュをトリガーします。

---

### proxmox-boot-tool が考慮するカーネルバージョン

デフォルトで設定されるカーネルバージョンは以下の通りです：

- 現在実行中のカーネル
- パッケージ更新時に新たにインストールされるバージョン
- 既にインストール済みの最新カーネル 2 つ
- 該当する場合、直近のカーネルシリーズの次バージョン（例：5.0、5.3）の最新バージョン
- 手動で選択されたカーネル

### 手動でカーネルを起動可能に維持する

特定のカーネルと initrd イメージを起動可能カーネルリストに追加する場合は、`proxmox-boot-tool kernel add` を使用してください。

例えば、ABI バージョン 5.0.15-1-pve のカーネルを、インストールされたまま全 ESP に同期されるカーネルリストに追加するには、以下を実行します：

```
# proxmox-boot-tool kernel add 5.0.15-1-pve
```

`proxmox-boot-tool kernel list` は現在ブート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 手動で選択された  
カーネル: 5.0.15-1-pve
```

自動選択カーネル: 5.0.12-1-pve

4.15.18-18-pve

手動で選択したカーネルのリストからカーネルを削除するには、`proxmox-boot-tool kernel remove` を実行します。例：

```
# proxmox-boot-tool カーネル 5.0.15-1-pve を削除
```

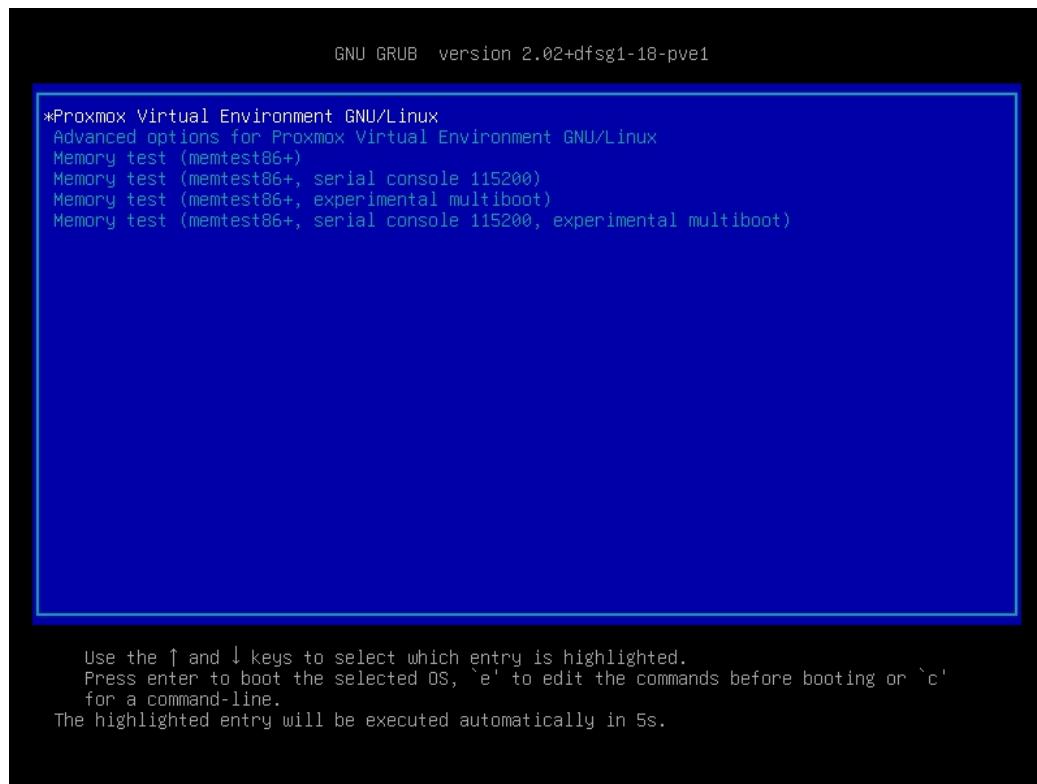
---

#### 注意

上記の手動でのカーネル追加または削除後、すべての EFI システムパーティション (ESP) を更新するには、`proxmox-boot-tool refresh` を実行する必要があります。

---

### 3.13.3 使用されているブートローダーの確認



使用されているブートローダーを確認する最も簡単で確実な方法は、Proxmox VE ノードの起動プロセスを観察することです。

GRUB の青いボックス、またはシンプルな白地に黒の `systemd-boot` が表示されます。



稼働中のシステムからブートローダーを特定することは100%正確ではない可能性があります。最も確実な方法は、以下のコマンドを実行することです：

```
# efibootmgr -v
```

EFI変数がサポートされていないというメッセージが返された場合、GRUBはBIOS/レガシーモードで使用されます。出力に以下のような行が含まれている場合、GRUBはUEFIモードで使用されます。

```
Boot0005* proxmox           [...] File(\EFI\proxmox\grubx64.efi)
```

出力に以下のような行が含まれている場合、`systemd-boot` が使用されています。

```
Boot0006* Linux Boot Manager      [...] File(\EFI\systemd\systemd-bootx64.efi)↔  
)
```

実行:

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

`proxmox-boot-tool` が設定されているかどうかを確認でき、システムがどのように起動しているかの良い指標となります。

### 3.13.4 GRUB

GRUBは長年にわたりLinuxシステムの起動における事実上の標準であり、非常に良く文書化されている  
<sup>7</sup>

#### 設定

GRUB設定の変更は、デフォルト設定ファイル `/etc/default/grub` または `/etc/default/grub.d` 内の設定スニペットで行います。設定変更後に設定ファイルを再生成するには、[以下を実行します](#)<sup>8</sup>:

```
# update-grub
```

### 3.13.5 Systemd-boot

`systemd-boot` は軽量な EFI ブートローダーです。インストール先の 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ロードする主な利点は、ストレージアクセス用ドライバを再実装する必要がないことです。Proxmox VE では、ESP 上の設定を同期するために [proxmox-boot-tool](#) が使用されます。

#### 設定

`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` ファイルを介して設定されます。詳細は `loader.conf(5)` マニュアルページを参照してください。

各ブートローダーエントリは、`loader/entries/` ディレクトリ内の個別のファイルに配置されます。

`entry.conf` の例は以下のようになります（/ は ESP のルートを指します）：

<sup>7</sup> GRUBマニュアル <https://www.gnu.org/software/grub/manual/grub/grub.html>

<sup>8</sup> `proxmox-boot-tool` を使用するシステムでは、`update-grub` 実行時に `proxmox-boot-tool refresh` が呼び出されます。

```
title      Proxmox version
5.0.15-1-pve
options    root=ZFS=rpool/ROOT/pve-1 boot=zfs
linux     /EFI/proxmox/5.0.15-1-pve/vmlinuz-5.0.15-1-pve initrd
```

### 3.13.6 カーネルコマンドラインの編集

使用しているブートローダーに応じて、以下の場所でカーネルコマンドラインを変更できます：

#### GRUB

カーネルコマンドラインは、ファイル

/etc/default/grub 内の変数 GRUB\_CMDLINE\_LINUX\_DEFAULT に設定する必要があります。update-grub を実行すると、その内容が /boot/grub/g

#### Systemd-boot

カーネルコマンドラインは、/etc/kernel/cmdline に 1 行で記述する必要があります。変更を適用するには、proxmox-boot-tool refresh を実行してください。これにより、loader/entries/proxmox-\* .conf 内のすべての設定ファイルのオプション行として設定されます。

カーネルパラメータの完全なリストは、<https://www.kernel.org/doc/html/v&lt;YOUR-KERNEL-VERSION&gt;/admin-guide/kernel-parameters.html> で確認できます。  
&lt;YOUR-KERNEL-VERSION&gt; をメジャー・マイナーバージョンに置き換えてください。例えば、バージョン 6.5 ベースのカーネルの場合、URL は <https://www.kernel.org/doc/html/-v6.5/admin-guide/kernel-parameters.html> となります。

カーネルバージョンは、ウェブインターフェース (*Node→Summary*) で確認するか、以下のコマンドを実行して確認できます。

```
# uname -r
```

出力の先頭にある最初の2つの数字を使用してください。

### 3.13.7 次回起動時のカーネルバージョンを上書きする

現在のデフォルトカーネル以外のカーネルを選択するには、次のいずれかの方法があります：

- ブートプロセスの開始時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、システムを特定のカーネルバージョンに一時的または永続的に固定する（固定設定が解除されるまで）。

これにより、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できるはずです。

#### 注意

最新のカーネルのセキュリティパッチをシステムに適用するため、このような固定設定は可能な限り早期に解除してください。

例: バージョン 5.15.30-1-pve を起動用に永続的に選択するには、以下を実行します:

```
# proxmox-boot-tool kernel pin 5.15.30-1-pve
```

#### ヒント

この固定機能は、ESPの内容を同期するためにproxmox-boot-toolを使用しているシステムだけでなく、すべてのProxmox VEシステムで動作します。システムが同期にproxmox-boot-toolを使用していない場合、最後のproxmox-boot-toolリフレッシュ呼び出しを省略することもできます。

カーネルバージョンを次回システム起動時にのみ設定することも可能です。これは、更新されたカーネルが当初バージョンを固定した原因となった問題を解決したかどうかをテストする場合などに有用です：

```
# proxmox-boot-tool カーネルピン 5.15.30-1-pve --next-boot
```

固定されたバージョンの設定を解除するには、unpinサブコマンドを使用します:

```
# proxmox-boot-tool kernel unpin
```

unpinコマンドにも--next-bootオプションがありますが、これは--next-bootで設定された固定バージョンをクリアするために使用されます。これは起動時に自動的に行われるため、手動で実行してもほとんど意味がありません。

固定バージョンの設定または解除後は、refreshサブコマンドを実行してESP上のコンテンツと設定を同期させる必要があります。

#### ヒント

プロキシモックスブートツールで管理されているシステムに対しては、対話的にツールを呼び出すと自動的に実行するよう促されます。

```
# proxmox-boot-tool refresh
```

### 3.13.8 セキュアブート

Proxmox VE 8.1以降、署名済みパッケージと

proxmox-boot-toolへの統合により、標準でサポートされています。

セキュアブートを機能させるには以下のパッケージが必要です。メタパッケージ'proxmox-secure-boot-support'を使用すれば一括インストールが可能です。

- shim-signed (Microsoft 署名付き shim ブートローダー)
- shim-helpers-amd64-signed (Proxmox 署名付きフォールバックブートローダーおよび MOKManager)
- grub-efi-amd64-signed (Proxmox 署名付き GRUB EFI ブートローダー)
- proxmox-kernel-6.X.Y-Z-pve-signed (Proxmox 署名付きカーネルイメージ)

GRUBのみが標準でサポートされるブートローダーです。他のブートローダーは現在、セキュアブートコード署名に対応していません。

Proxmox VEの新規インストール時には、上記のパッケージがすべて自動的に含まれます。

セキュアブートの仕組みや設定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

正しく行わないと、場合によっては起動不能になる可能性があります。ホストを再インストールすると、追加操作なしで、利用可能な場合は自動的にセキュアブートが設定されます。Proxmox VE ホストのバックアップが確実に動作し、十分にテストされていることを確認してください。

既存のUEFIインストール環境は、必要に応じてProxmox VEを最初から再インストールすることなく、セキュアブートに切り替えることが可能です。

まず、システム全体が最新の状態であることを確認してください。次に、`proxmox-secure-boot-support` をインストールします。GRUB はデフォルトの shim 経由で起動するために必要な EFI ブートエントリを自動的に作成します。

### systemd-boot

`systemd-boot` をブートローダーとして使用している場合（[使用中のブートローダーの確認を参照](#)）、追加設定が必要です。これは Proxmox VE を ZFS-on-root でインストールした場合にのみ該当します。

後者を確認するには、以下を実行してください：

```
# findmnt /
```

ホストが実際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`が含まれているはずです。

TARGET	SOURCE	FSTYPE	OPTIONS
/	rpool/ROOT/pve-1 zfs		<code>rw,relatime,xattr,noacl,casesensitive</code>

次に、適切な潜在的なESP（EFIシステムパーティション）を見つける必要があります。これは`lsblk`コマンドを使用して行うことができます。

コマンドで次のように実行できます：

```
# lsblk -o +FSTYPE
```

出力は以下のようになります：

NAME	MAJ:MIN	RM	サイズ	ロータイプ	マウントポイント	ファイルシステムタイプ
sda	8:0	0	32G	0	ディスク	
&#x251c;&#x2500;sda1	8:1	0	1007K	0	part	
&#x251c;&#x2500;sda2	8:2	0	512M	0	part	<code>vfat</code>
&#x2514;&#x2500;sda3	8:3	0	31.5G	0	part	<code>zfs_member</code>
sdb	8:16	0	32G	0	ディス	
&#x251c;&#x2500;sdb1	8:17	ク	1007K	0	part	
			0			
&#x251c;&#x2500;sdb2	8:18	0	512M	0	part	<code>vfat</code>
&#x2514;&#x2500;sdb3	8:19	0	31.5G	0	part	<code>zfs_member</code>

この場合、パーティション `sda2` と `sdb2` が対象です。これらはサイズが 512M であり、FSTYPE が `vfat` であることから識別できます（この例では ZFS RAID-1 構成上）。

これらのパーティションは、`proxmox-boot-tool` を使用して GRUB 経由でブートできるように適切に設定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 ごとに個別に実行する必要があります：

```
# proxmox-boot-tool init /dev/sda2 grub
```

その後、以下のコマンドを実行して設定の整合性を確認できます：

```
# efibootmgr -v
```

このリストには、以下のようなエントリが含まれているはずです:

```
[...]
Boot0009* proxmox
    shimx64.efi)
[...]
```

#### 注

従来の `systemd-boot` ブートローダーは維持されますが、GRUB が優先されます。これにより、何らかの理由でセキュアブートモードでの GRUB 起動が失敗した場合でも、セキュアブートを無効化した状態で `systemd-boot` を使用してシステムを起動することができます。

ホストを再起動し、UEFIファームウェア設定ユーティリティでセキュアブートを有効にできます。

再起動後、UEFIファームウェアのブートメニューに「`proxmox`」という新しいエントリが表示され、事前署名済みEFIシムを使用して起動できるようになります。

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は、ファームウェアがサポートしている場合に限り、`\EFI\proxmox\shimx64.efi` ファイルをカスタムブートエントリとして追加することで手動で追加を試みることができます。

#### 注意

一部の UEFI ファームウェアでは、再起動時に `proxmox` ブートオプションが削除されることが知られています。これは、`proxmox` 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る場合で、そのディスク自体がブートオプションとして設定されていない場合に発生する可能性があります。可能であれば、UEFI ファームウェア設定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` を再度実行してみてください。

#### ヒント

カスタムキーを登録するには、付属の[セキュアブートWikiページ](#)を参照してください。

### セキュアブート環境でのDKMS/サードパーティモジュールの使用

セキュアブートが有効なシステムでは、カーネルは信頼された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るデフォルトのモジュール群は、カーネルイメージに埋め込まれた一時的な鍵で署名されており、その特定のバージョンのカーネルイメージによって信頼されています。

DKMSでビルドされたモジュールや手動でビルドされたモジュールなど、他のモジュールをロードするには、Secure Bootスタックが信頼する鍵で署名されている必要があります。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所有者鍵（MOK）として登録することです。

`dkms`ツールは自動的に鍵ペアと証明書を生成し、`/var/lib/dkms/mok.key` および  
`/var/lib/dkms/mok.pub` に生成し、ビルド・インストールするカーネルモジュールの署名に使用します。証明書の内容は以下で確認できます

```
# openssl x509 -in /var/lib/dkms/mok.pub -noout -text
```

そして、次のコマンドを使用してシステムに登録します:

```
# mokutil --import /var/lib/dkms/mok.pub 入力パスワード:  
パスワードを再入力してください:
```

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一度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 バイナリが起動します。これにより、鍵/証明書を検証し、mokutil を使用して登録を開始した際に選択したパスワードを使用して登録を確認できます。その後、カーネルは DKMS でビルドされたモジュール（登録済みの MOK で署名されたもの）のロードを許可するはずです。必要に応じて、MOK を使用してカスタム EFI バイナリやカーネルイメージに署名することもできます。

DKMSで管理されていないカスタム/サードパーティ製モジュールに対しても同様の手順が適用可能ですが、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ります。

## 3.14 カーネル・セメページ・マージング (KSM)

カーネル・セメページ・マージング (KSM) 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あり、Proxmox VEではデフォルトで有効化されています。KSMは、物理メモリページの範囲をスキャンして同一の内容を検出し、それらにマップされている仮想ページを特定します。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再マップされ、古いページは解放されます。仮想ページは「コピー온라이트」としてマークされるため、それらへの書き込みはメモリの新しい領域に書き込まれ、共有されている物理ページはそのまま残ります。

### 3.14.1 KSMの意義

KSMは仮想化環境におけるメモリ使用を最適化できます。類似のOSやワークロードを実行する複数のVMが、多くの共通メモリページを共有する可能性があるためです。

ただし、KSMはメモリ使用量を削減できる一方で、サイドチャネル攻撃に対する脆弱性をVMに露呈させるセキュリティリスクも伴います。研究により、KSMの特定の特性を悪用することで、同一ホスト上の別のVMから稼働中のVMに関する情報を推測できる可能性が示されています。

したがって、Proxmox VEでホスティングサービスを提供している場合は、ユーザーに追加のセキュリティを提供するため、KSMを無効化することを検討すべきです。さらに、KSMの無効化が法的要件となる可能性があるため、自国の規制を確認する必要があります。

### 3.14.2 KSMの無効化

KSMが有効かどうかを確認するには、以下の出力を参照してください：

```
# systemctl status ksmtuned
```

有効な場合、次のコマンドで即時無効化できます：

```
# 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、以下を実行します：

```
# 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。別途管理ツールをインストールする必要はなく、すべてをウェブブラウザ（最新版のFirefoxまたはGoogle Chromeが推奨）から操作できます。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。代替手段としてSPICEも利用可能です。

Proxmoxクラスタファイルシステム (pmxcfs) を使用しているため、任意のノードに接続してクラスタ全体を管理できます。各ノードがクラスタ全体を管理可能で、専用の管理ノードは不要です。

ウェブベースの管理インターフェースは、最新のブラウザであればどれでも使用できます。Proxmox VEがモバイルデバイスからの接続を検知すると、よりシンプルなタッチ操作対応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ースは <https://youripaddress:8006> からアクセス可能です（デフォルトログイン：root、パスワードはインストールプロセス中に指定されます）。

## 4.1 機能

- Proxmox VEクラスター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およびコンテナへの安全なアクセス
- 高速検索駆動インターフェース（数百、おそらく数千の仮想マシンを処理可能）
- セキュアなHTML5コンソールまたはSPICE
- すべてのオブジェクト（仮想マシン、ストレージ、ノードなど）に対するロールベースの権限管理
- 複数の認証ソースのサポート（例：ローカル、MS ADS、LDAP、...）
- 二要素認証（OATH、Yubikey）
- ExtJS 7.x JavaScriptフレームワークベース

## 4.2 ログイン

Proxmox VE Login

User name: root

Password:

Realm: Linux PAM standard authentication

Language: English

Save User name:  Login

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 はさまざま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

下部にある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の入力が省けます。

## 4.3 GUI 概要

**Server View**

- Datacenter (prod-eu-central)
  - prod1
  - prod2
  - prod3
  - development

**Documentation** **Create VM** **Create CT** **admin@pve**

**Help**

**Health**

Status	Nodes	Ceph
✓ Online	3	✓
✗ Offline	0	✓

Cluster: prod-eu-central, Quorad: Yes  
HEALTH\_OK

**Guests**

Virtual Machines		LXC Container	
Running	1	Running	0
Stopped	2	Stopped	1

**Resources**

CPU	Memory	Storage
2%	22%	4%
of 12 CPU(s)	20.35 GiB of 94.13 GiB	256.60 GiB of 6.82 TiB

**Nodes**

**Tasks** Cluster log

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     | 上部。ステータス情報を表示し、主要な操作ボタンを含みます。リソースツリー 左側。特定のオブジェクトを選択できるナビゲーションツリーです。 |
| コンテンツパネル | 中央領域。選択されたオブジェクトの構成オプションとステータスが表示されます。                               |
| ログパネル    | 下部。直近のタスクのログエントリを表示します。ログエントリをダブルクリックすると詳細を確認したり、実行中のタスクを中止したりできます。  |

#### 注記

リソースツリーと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できます。小さな画面で作業する際、他のコンテンツを表示するスペースを確保するのに便利です。

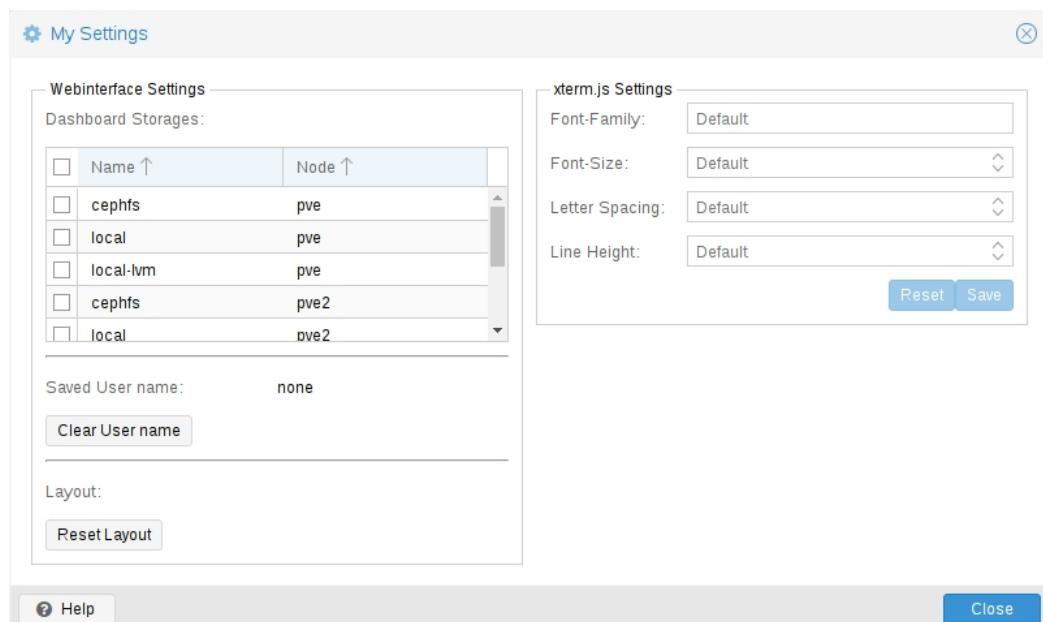
### 4.3.1 ヘッダー

左上には、まずProxmoxのロゴが表示されます。その隣には、現在実行中のProxmox VEのバージョンが表示されます。その近くの検索バーでは、特定のオブジェクト（VM、コンテナ、ノードなど）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早い場合があります。

ヘッダーの右側には4つのボタンがあります：

- |          |  |      |
|----------|--|------|
| 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開きます。仮想マシン作成                   | 仮想マシ |
| CT作成     | コンテナ作成ウィザードを開きます。  |      |
| ユーザーメニュー | 現在ログインしているユーザーの識別情報を表示し、クリックするとユーザー固有のオプションを含む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 [マイ設定] ダイアログがあります。その下には、TFA (二要素認証) および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、言語やカラースキームを変更するオプションもあります。最後に、メニューの下部には [ログアウト] オプションがあります。

### 4.3.2 マイ設定



マイ設定ウィンドウでは、ローカルに保存される設定を指定できます。これには「ダッシュボードストレージ」が含まれ、データセンター概要に表示される合計容量に特定ストレージを含めるか除外するかを選択できます。いずれのストレージもチェックされていない場合、合計は全ストレージの合計となり、すべてのストレージを有効にした場合と同じ結果になります。

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、およびGUIのすべての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` の設定があります。これには以下のオプションが含まれます：

Font-Family	<code>xterm.js</code> で使用するフォント（例：Arial）。Font-Size	使用する推奨フォントサイズ。
文字間隔	テキスト内の文字間の間隔を増減します。行の高さ	行の高さを絶対値で指定します。

### 4.3.3 リソースツリー

これはメインナビゲーションツリーです。ツリー上部では、下部のツリー構造を変更するいくつかの事前定義ビューを選択できます。デフォルトビューはサーバービューであり、以下のオブジェクトタイプを表示します：

データセンター	クラスタ全体の設定（全ノードに適用）を含みます。
ノード	クラスタ内のホストを表し、ゲストが実行されます。ゲスト 仮想マシン、コンテナ、およびテンプレート。
データセンター	データストレージ。
プール	管理を簡素化するため、プールを使用してゲストをグループ化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：

サーバービュー	ノードごとにグループ化された、あらゆる種類のオブジェクトを表示します
。 フォルダビュー	オブジェクトタイプ別にグループ化された全ての種類のオブジェクトを表示
します。 プールビュー	プールごとにグループ化された仮想マシンとコンテナを表示します。
タグビュー	タグごとにグループ化された仮想マシンとコンテナを表示します。

### 4.3.4 ログパネル

ログパネルの主な目的は、クラスター内で現在進行中の処理を表示することです。新しい仮想マシンの作成などの操作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。

タスクからの出力はすべて個別のログファイルに保存されます。タスクログエントリをダブルクリックするだけでそのログを確認できます。また、実行中のタスクをここで中止することも可能です。

なお、ここでは全クラスターノードから最新のタスクを表示します。これにより、他のユーザーが別のクラスターノードで作業している状況をリアルタイムで確認できます。

#### 注記

ログパネルのリストを簡潔に保つため、古いタスクや完了したタスクは削除されます。ただし、それらのタスクはノードパネルの「タスク履歴」内で引き続き確認可能です。

一部の短時間実行アクションは、単にログをすべてのクラスタメンバーに送信します。それらのメッセージはクラスタログパネルで確認できます。

## 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および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、この機能の概要を簡単に説明します。詳細については、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 6.0-4 interface with the 'Datacenter' content panel selected. The left sidebar lists clusters: prod1, prod2, prod3, and development. The main area displays a table of resources with columns: Type, Description, Disk usage..., Memory us..., CPU usage, and Uptime. The table includes entries for nodes (prod1, prod2, prod3), pools (development), and storage devices (qemu, ceph, cp, iso, local, local-lvm). Below the table is a log table titled 'Cluster log' showing tasks like VM creation and destruction.

Type	Description	Disk usage...	Memory us...	CPU usage	Uptime
lxn	510 (CT510)				-
node	prod1	6.8 %	20.4 %	3.7% of 4C...	11 days 00:0...
node	prod2	6.2 %	25.1 %	2.5% of 4C...	11 days 00:0...
node	prod3	6.1 %	19.3 %	1.0% of 4C...	11 days 00:3...
pool	development				-
qemu	101 (win10)				-
qemu	501 (VM 501)				-
qemu	100 (VM 100)		81.4 %	3.6% of 2C...	3 days 23:29...
storage	cephfs (prod1)	0.0 %			-
storage	cp (prod1)	6.2 %			-
storage	iso (prod1)	3.8 %			-
storage	local (prod1)	6.8 %			-
storage	local-lvm (prod1)	0.8 %			-
storage	cephfs (prod2)	0.0 %			-
storage	cp (prod2)	6.2 %			-
storage	iso (prod2)	3.8 %			-
storage	local (prod2)	6.2 %			-
storage	local-lvm (prod2)	0.0 %			-
storage	cephfs (prod3)	0.0 %			-
storage	cp (prod3)	6.2 %			-
storage	iso (prod3)	3.8 %			-
storage	local (prod3)	6.1 %			-
storage	local-lvm (prod3)	0.0 %			-

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や情報にアクセスできます。

- 検索:** ノード、仮想マシン、コンテナ、ストレージデバイス、プールを対象としたクラスタ全体の検索を実行します。
- 概要:** クラスタの健全性とリソース使用状況の概要を簡潔に説明します。
- クラスター:** クラスター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を表示および管理します。
- ストレージ:** クラスターストレージを管理するためのインターフェースを提供します。
- バックアップ:** バックアップジョブのスケジュール設定を行います。これはクラスター全体で動作するため、スケジュール設定時に仮想マシン/コンテナがクラスター内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を表示および管理します。
- **権限:** ユーザー、グループ、API トークンの権限、および LDAP、MS-AD、二要素認証を管理します。
- **HA:** Proxmox VE 高可用性を管理します。
- **ACME:** サーバーノード向けにACME (Let's Encrypt) 証明書を設定する。
- **ファイアウォール:** Proxmox ファイアウォールのクラスター全体設定とテンプレート作成。
- **メトリックサーバー:** Proxmox VE用の外部メトリックサーバーを定義する。
- **通知:** Proxmox VE の通知動作と通知先を設定します。
- **サポート:** サポートサブスクリプションに関する情報を表示します。

## 4.4.2 ノード

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

このレベルでは、クラスタ内のノードを個別に管理できます。

上部ヘッダーには、再起動、シャットダウン、シェル、一括操作、ヘルプなどの便利なボタンがあります。シェルにはnoVNC、SPICE、xterm.jsのオプションがあります。一括操作には一括起動、一括シャットダウン、一括移行のオプションがあります。

- 検索: ノード内の仮想マシン、コンテナ、ストレージデバイス、プールを検索します。
- 概要: ノードのリソース使用状況の概要を表示します。
- メモ: Markdown構文でカスタムコメントを記述します。
- シェル: ノードのシェルインターフェースにアクセスします。
- システム: ネットワーク、DNS、時刻設定を構成し、syslogにアクセスします。
- 更新: システムのアップグレードと利用可能な新規パッケージの確認。
- 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: 接続されたディスクの概要を取得し、使用方法を管理します。
- Ceph: ホストにCephサーバーをインストールしている場合のみ使用されます。この場合、Cephクラスターを管理し、その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を表示および管理します。
- タスク履歴: 過去のタスクの一覧を表示します。
- 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 top navigation bar includes links for Documentation, Create VM, Create CT, and Help, along with a user session for admin@pve.

The main content area is titled "Virtual Machine 99999 on node 'prod1'". On the left, a sidebar lists the "Server View" with a tree structure of Datacenter (prod-eu-central), prod1, prod2, prod3, and development. Under prod1, there are several virtual machines: 510 (CT510), 101 (win10), 501 (VM 501), and 99999 (selected). The 99999 entry has a submenu with options like cephfs, cp, iso, local, local-lvm, and others.

The central panel has tabs for "Summary", "Console", "Hardware", "Cloud-Init", "Options", "Task History", "Monitor", "Backup", "Replication", "Snapshots", "Firewall", and "Permissions". The "Summary" tab is active, displaying information for "DemoVM" including 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), and IPs (No Guest Agent configured).

Below the summary, there's a "CPU usage" graph showing utilization over time. At the bottom, there are tabs for "Tasks" and "Cluster log". The "Tasks" tab shows a table of recent operations:

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、いずれもテンプレートに変換可能です。1つはカーネルベース仮想マシン（KVM）、もう1つはLinuxコンテナ（LXC）です。操作体系はほぼ共通ですが、一部オプションが異なります。

各種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、移行、コンソールアクセスとタイプ、クローン、HA、ヘルプなどの項目に対するコマンドが含まれています。これらのボタンの中には、ドロップダウンメニューを含むものもあります。たとえば、「シャットダウン」には他の電源オプションも含まれており、「コンソール」には、SPICE、noVNC、xterm.jsなどのさまざまなコンソールタイプが含まれています。

右側のパネルには、左側のメニューから選択された項目に対応するインターフェースが表示されます。利用可能なインターフェースは以下の通りです

。

- **概要:** VMの動作概要を簡潔に表示し、[Markdown構文](#)によるコメント用の「メモ」フィールドを提供します。
- **コンソール:** VM/コンテナ用の対話型コンソールへのアクセス。
- **(KVM)ハードウェア:** KVM VM が利用可能なハードウェアを定義します。
- **(LXC)リソース:** LXCが利用可能なシステムリソースを定義します。
- **(LXC)ネットワーク:** コンテナのネットワーク設定を構成します。
- **(LXC)DNS:** コンテナのDNS設定を構成します。
- **オプション:** ゲスト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) モニター:** 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:** システムのバックアップを作成および復元します。
- **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復元。
- **ファイアウォール:** VM レベルでファイアウォールを設定します。
- **権限:** 選択したゲストの権限を管理します。

#### 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ースと同様に、ストレージのインターフェースは、左側に特定のストレージ要素のメニュー、右側にそれらの要素を管理するためのインターフェー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割されたビューが表示されます。左側にはストレージオプションが表示され、右側には選択したオプションの内容が表示されます。

- 概要**：ストレージのタイプ、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：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。
- 権限**：ストレージの権限を管理します。

#### 4.4.5 プール

The screenshot shows the Proxmox VE 6.0-4 interface. In the top navigation bar, there are links for Documentation, Create VM, Create CT, and a user account for admin@pve. The main window is titled "Resource Pool: development". On the left, a tree view shows the Datacenter (prod-eu-central) with nodes prod1, prod2, prod3, and a selected node "development". Node prod2 contains several storage volumes: 100 (VM 100), cephfs (prod2), cp (prod2), iso (prod2), local (prod2), and local-lvm (prod2). The right panel has tabs for Summary, Members, and Permissions. Under "Summary", there is a "Status" section with a comment "IT development pool". At the bottom, there are tabs for Tasks and Cluster log. The Cluster log table shows the following tasks:

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

プールビューは再び2つの領域で構成されます：左側のメニューと、右側の各メニュー項目に対応するインターフェースです。

- **概要:** プールの説明を表示します。
- **メンバー:** プールのメンバー（ゲストとストレージ）を表示および管理します。
- **権限:** プールの権限を管理します。

## 4.5 タグ

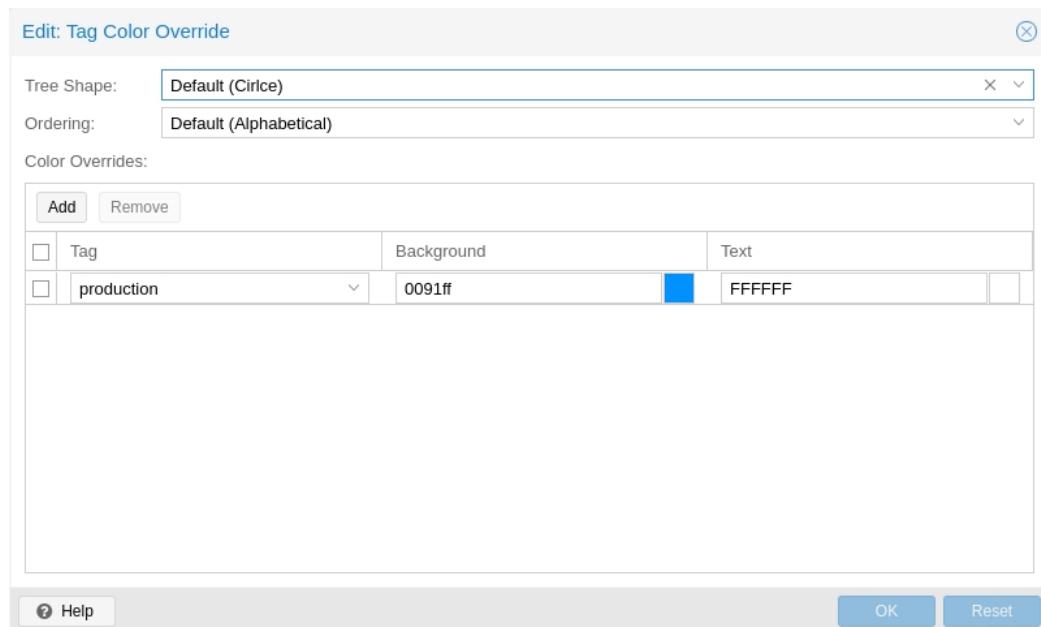
整理の目的で、ゲストにタグを設定できます。現在、これらはユーザーにとって情報提供のみを目的としています。タグはウェブインターフェースの2か所（リソースツリーとゲスト選択時のステータス行）に表示されます。

ゲストのステータスバーでは、鉛筆アイコンをクリックしてタグの追加・編集・削除が可能です。+ボタンで複数タグを追加し、-ボタンで削除できます。変更の保存/キャンセルにはそれぞれ✓ボタンと✗ボタンを使用します。

タグはCLIからも設定可能です。複数のタグはセミコロンで区切れます。例：

```
# qm set ID --tags 'myfirsttag;mysecondtag'
```

## 4.5.1 スタイル設定



デフォルトでは、タグの色はテキストから決定論的な方法で導出されます。色、リソースツリーでの形状、大文字小文字の区別、およびタグのソート方法はカスタマイズ可能です。これは、Webインターフェースの「データセンター」→「オプション」→「タグスタイルのオーバーライド」で設定できます。あるいは、CLI経由でも設定可能です。例：

```
# pvesh set /cluster/options --tag-style color-map=example:000000:FFFFFF
```

タグ example の背景色を黒 (#000000)、文字色を白 (#FFFFFF) に設定します。

## 4.5.2 権限

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), Tag: production
User Tag Access	Mode existing, Pre-defined: allowed
Registered Tags	backup

デフォルトでは、ゲスト (/vms / ID) に対して VM.Config.Options 権限を持つユーザーは、任意のタグを設定できます

任意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、適切な権限を以下に設定できます：データセンター→ オプション→ ユーザータグアクセス：

- free: ユーザーはタグ設定に制限なし（デフォルト）
- list: ユーザーは事前定義されたタグリストに基づいてタグを設定可能
- 既存: listと同様だが、既存タグも使用可能
- none: ユーザーはタグの使用が制限される

同様の設定はCLIからも行えます。

/ に対する Sys.Modify 権限を持つユーザーは、ここでの設定に関わらず、常に任意のタグを設定または削除できます。さらに、/ に対する Sys.Modify 権限を持つユーザーのみが追加・削除可能な登録済みタグのリストが設定可能です。登録済みタグのリストは、[データセンター]→[→ オプション]→[→ 登録済みタグ] または CLI から編集できます。

詳細なオプションとCLIでの呼び出し方法については、[データセンター設定](#)を参照してください。

## 4.6 同意バナー

ログイン前に承諾が必要なカスタム同意バナーは、データセンターの設定で構成できます。→ オプション

→ 同意テキスト。内容がない場合、同意バナーは表示されません。テキストは/etc/pve/datacenter.cfg設定ファイルにBase64文字列として保存されます。

## 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ー pvecm は、物理サーバー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は **クラスタ** と呼ばれます。信頼性の高いグループ通信には [Corosync クラスタエンジン](#) を使用します。クラスタ内のノード数に明示的な制限はありません。実際には、ホストとネットワークの性能によって実際の可能なノード数が制限される場合があります。現在（2021年）、ハイエンドのエンタープライズハードウェアを使用した50ノードを超えるクラスタが本番環境で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、ノードのクラスターへの参加、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タスクの実行に使用できます。Proxmox Cluster File System（「pmxcfs」）は、クラスター構成をすべてのクラスター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ことには以下の利点があります：

- ・ 集中管理型のWebベース管理
- ・ マルチマスタークラスター：各ノードが全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である pmxcfs を使用した設定ファイルの保存 (corosync により全ノードへリアルタイムに複製)
- ・ 物理ホスト間での仮想マシンおよびコンテナの容易な移行
- ・ 迅速なデプロイ
- ・ ファイアウォールや高可用性 (HA) などのクラスタ全体サービス

### 5.1 要件

- ・ すべてのノードは、Corosyncが機能するためにUDPポート5405-5412を介して相互に接続できる必要があります。
- ・ 日付と時刻は同期されている必要があります。
- ・ ノード間ではTCPポート22上のSSHトンネルが必要です。
- ・ 高可用性（High Availability）に関心がある場合は、信頼性の高いクオーラムを確保するために少なくとも3つのノードが必要です。すべてのノードは同じバージョンである必要があります。

---

#### 注記

小規模な2ノードクラスタの場合、[QDevice](#)を使用して3番目の投票権を提供できます。

---

- クラスタトラフィックには専用の物理NICを推奨します。

---

#### 注記

Proxmox VEクラスタ通信はCorosyncプロトコルを使用します。一貫した低遅延が必要ですが、大容量の帯域幅は不要です。専用1Gbit NICはほとんどの状況で十分です。これにより、他のサービスが利用可能な帯域幅を全て消費してしまう状況を回避できます。これが発生すると、Corosyncパケットの遅延が増加します。

---

- クラスタトラフィック用の追加リンクは、専用ネットワークがダウンした場合の冗長性を提供します。

---

#### 注

Corosyncは最大8つのリンクをサポートします。

---

---

#### 注記

信頼性の高いCorosync冗長性を確保するには、物理的に異なるネットワーク上に少なくとも1つの追加リンクを設置することが不可欠です。これにより、専用ネットワークがダウンした場合でも、Corosyncはクラスタ通信を維持できます。

---

---

#### 注

特定の障害シナリオでは、1つのリンクを1つのボンディングでバックアップすることは問題となる場合があります。詳細は[「Corosync Over Bonds」](#)を参照してください。

---

- ノード追加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移行は、ノードが同一ベンダーのCPUを搭載している場合にのみサポートされます。それ以外のケースでも動作する可能性はあります  
が、保証されることはありません。

## 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。各ノードは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。クラスタ一作成後のホスト名やIPの変更は不可能です。

/etc/hosts ファイルで全ノード名とIPを参照する（または他の方法で名前解決可能にする）ことは一般的ですが、クラスタ動作には必須ではありません。ただし、覚えやすいノード名でSSH接続が可能になるため有用です（[リンクアドレスの種類も](#)参照）。クラスタ設定では常にIPアドレスによるノード参照を推奨します。

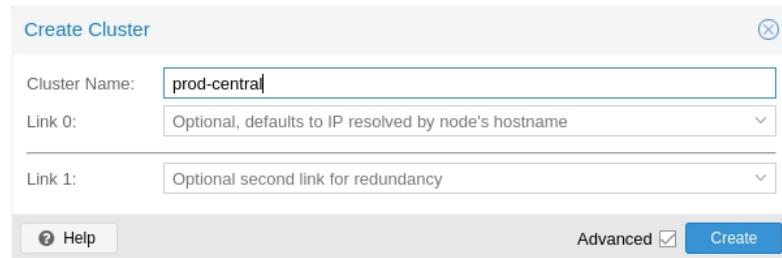
## 5.3 クラスターの作成

クラスターは、コンソール（SSH経由でログイン）で作成するか、Proxmox VE WebインターフェースのAPI（データセンター→ クラスタ）を使用して作成できます。

### 注記

クラスターには一意の名前を指定してください。この名前は後から変更できません。クラスター名はノード名と同じ規則に従います。

### 5.3.1 Web GUI経由での作成



[データセンター] → [→ クラスター] で、[クラスターの作成] をクリックします。クラスター名を入力し、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接続を選択します。デフォルトではノードのホスト名で解決されたIPが設定されます。

Proxmox VE 6.2以降、クラスターには最大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、[追加] ボタンをクリックし、各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、フォールバック用の2つ目のリンクを追加する場合、[詳細設定] 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](#)）を選択できます。

### 注意

クラスター通信用に選択したネットワークは、ネットワークストレージやライブマイグレーションなど、高トラフィック用途に使用しないでください。クラスタネットワーク自体は少量のデータしか生成しませんが、レイテンシに非常に敏感です。[クラスタネットワークの完全な要件](#)を確認してください。

### 5.3.2 コマンドライン経由での作成

最初のProxmox VEノードにSSHでログインし、次のコマンドを実行します:

```
hp1# pvecm create CLUSTERNAME
```

新しいクラスターの状態を確認するには以下を使用します:

```
hp1# pvecm status
```

### 5.3.3 同一ネットワーク内の複数クラスタ

同一の物理ネットワークまたは論理ネットワーク内に複数のクラスタを作成することができます。この場合、クラスタ間通信スタッフでの競合を避けるため、各クラスターには一意の名前を付ける必要があります。さらに、クラスターを明確に区別できるようにすることで、人為的な混乱を回避できます。

Corosyncクラスタの帯域幅要件は比較的低いものの、パケットの遅延とパケット毎秒（PPS）レートが制限要因となります。同一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競合するため、大規模クラスタでは物理ネットワークインフラを分離して使用することが依然として有効です。

## 5.4 クラスタへのノード追加

### 注意

クラスタ参加時には/etc/pve内の既存設定がすべて上書きされます。特に、参加ノードはゲストを保持できません（ゲストIDの競合が発生する可能性があるため）。また、ノードはクラスタのストレージ設定を継承します。既存ゲストを持つノードを参加させる場合、回避策として各ゲストのバックアップ（vzdumpを使用）を作成し、参加後に異なるIDで復元できます。ノードのストレージレイアウトが異なる場合、ノードのストレージを再追加し、各ストレージのノード制限を、ストレージが実際に利用可能なノードを反映するように適応させる必要があります。

### 5.4.1 GUI経由でノードをクラスターに追加

**Cluster Join Information**

Copy the Join Information here and use it on the node you want to add.

IP Address: 192.168.26.225

Fingerprint: 56:AB:A8:DD:4E:F4:85:FB:BD:C9:93:55:1C:C2:6B:D7:88:54:B6:05:B9:18:02:9A:25:0D:B5:39:D7:FE:7C:D6

Join Information: eyJpcEFkZHJlc3MiOiixtOTluMTY4LjI2LjlyNSIsImZpbmdlcnByaW50IjoiNTY6QUI6QTg6REQ6NEU6RjQ6ODU6RkI6QkQ6Qzk6OTM6NTU6MUM6QzI6NkI6RDc6ODg6NTQ6QjY6MDU6Qjk6MTg6MDI6OUe6MjU6MEQ6QjU6Mzk6RDc6RkU6N0M6RDYiLCJyaW5nX2FkZHliOlisiMTkyLjE2OC4yNi4yMjUiLG51bGxdLCJ0b3RlbSI6eyJjbHbHVzrdGVvX25hbWUiOljwcm9kLWV1LWNlbnRvYWwiLCJpbnRlcmbhY2UiOnsiMC16evJsaW5rbnVtYmVvlioiMCJ

**Copy Information**

既存クラスターノードのWebインターフェースにログインします。 [Datacenter→ Cluster] セクションの上部にある [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、 [Information] フィールドの文字列を手動でコピーします。

**Cluster Join**

Assisted join: Paste encoded cluster join information and enter password.

Information: eyJpcEFkZHJlc3MiOiixtOTluMTY4LjI2LjlyNSIsImZpbmdlcnByaW50IjoiNTY6QUI6QTg6REQ6NEU6RjQ6ODU6RkI6QkQ6Qzk6OTM6NTU6MUM6QzI6NkI6RDc6ODg6NTQ6QjY6MDU6Qjk6MTg6MDI6OUe6MjU6MEQ6QjU6Mzk6RDc6RkU6N0M6RDYiLCJyaW5nX2FkZHliOlisiMTkyLjE2OC4yNi4yMjUiLG51bGxdLCJ0b3RlbSI6eyJjbHbHVzrdGVvX25hbWUiOljwcm9kLWV1LWNlbnRvYWwiLCJpbnRlcmbhY2UiOnsiMC16evJsaW5rbnVtYmVvlioiMCJ

Peer Address: 192.168.26.225

Link 0: Default: IP resolved by node's hostname

Password: \*\*\*\*\*

Link 1:

Fingerprint: 56:AB:A8:DD:4E:F4:85:FB:BD:C9:93:55:1C:C2:6B:D7:88:54:B6:05:B9:18:02:9A:25:0D:B5:39:D7:FE:7C:D6

**Help** **Join**

次に、追加したいノードのWebインターフェースにログインします。「Datacenter」→「」→「Cluster」の下にある「Join Cluster」をクリックします。先ほどコピーした「Join Information」テキストを「Information」フィールドに入力します。クラスター参加に必要な設定の大半は自動入力されます。セキュリティ上の理由から、クラスター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、「アシスト付き参加」チェックボックスを無効にできます。

[Join]ボタンをクリックすると、クラスター参加プロセスが直ちに開始されます。ノードがクラスターに参加すると、現在のノード証明書はクラスター認証局（CA）から署名された証明書に置き換えられます。これにより、現在のセッションは数秒後に動作を停止します。その後、Webインターフェースを強制再読み込みし、クラスター認証情報で再度ログインする必要がある場合があります。

これで、ノードが「データセンター→ クラスター」下に表示されるはずです。

## 5.4.2 コマンドライン経由でのノードのクラスター参加

既存のクラスター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、既存クラスターノードのIPアドレスまたはホスト名を使用します。IPアドレスの使用が推奨されます（[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以下を使用します：

```
# pvecm status
```

### 4ノード追加後のクラスタ状態

```
# pvecm statusクラスタ情報
~~~~~
Name: prod-central 構成
バージョン: 3 トランSPORT: knet
セキュア認証: on

定足数情報
~~~~~
日付: 2021年9月14日(火) 11:06:47
クオーラムプロバイダー: corosync_votequorumノード: 4
ノードID: 0x00000001
リングID: 1.1a8
定足数: はい

投票定足数情報
~~~~~ 予想投票数:
4
最高予想: 4
総投票数: 4
定足数: 3
フラグ: 定足数あり

メンバーシップ情報
~~~~~ノードID 投票名
0x00000001 1 192.168.15.91
0x00000002 1 192.168.15.92 (ローカル)
```

```
0x00000003      1 192.168.15.93  
0x00000004      1 192.168.15.94
```

すべてのノードのリストのみが必要な場合は、以下を使用してください:

```
# pvecm nodes
```

#### クラスタ内のノードを一覧表示

```
# pvecm nodes
```

##### メンバーシップ情報

ノードID	投票	名前
1	1	hp1
2	1	hp2 (ローカル)
3	1	hp3
4	1	hp4

### 5.4.3 分離クラスター ネットワークでのノード追加

分離クラスタネットワークを持つクラスタにノードを追加する際は、link0パラメータを使用してそのネットワーク上のノードアドレスを設定する必要があります:

```
# pvecm add IP-ADDRESS-CLUSTER --link0 LOCAL-IP-ADDRESS-LINK0
```

Kronosnetトランスポート層の組み込み冗長性を利用する場合は、link1パラメータも使用してください。GUIを使用する場合、**クラスタ参加ダイアログ**の対応する

Link Xフィールドから正しいインターフェースを選択できます。

ダイアログで適切なインターフェースを選択できます。

## 5.5 クラスタノードの削除



#### 注意

手順を進める前に注意深くお読みください。ご希望や必要とする内容と異なる可能性があります。

対象ノードから全ての仮想マシンを移動させてください。保持したいローカルデータやバックアップは必ずコピーを取っておいてください。さらに、削除対象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は全て削除してください。

#### 注意

ノード削除前にレプリケーションジョブを解除しないと、当該ジョブが削除不能になります。特に注意：レプリケートされたVMが移行されるとき、レプリケーション方向は自動反転します。従って、削除対象ノードからレプリケートVMを移行すると、自動的に当該ノードへのレプリケーションジョブが設定されます。

削除対象ノードがCeph用に構成されている場合：

1. OSDが稼働中（アップ状態）のProxmox VEノードが十分に存在し続けることを確認してください。

#### 注意

デフォルトでは、Cephプールはサイズ/最小サイズが3/2であり、オブジェクトバランサーCRUSHにおいてフルノードが障害ドメインとなります。したがって、稼働中のOSDを持つノードがサイズ（3）未満でオンラインの場合、データの冗長性が低下します。最小サイズ未満のノードがオンラインの場合、プールのI/Oがブロックされ、影響を受けるゲストがクラッシュする可能性があります。

2. 十分な数のモニターノード、マネージャーノード、およびCephFSを使用している場合はメタデータサーバーが利用可能であることを確認してください。
3. データ冗長性を維持するため、OSDの破棄（特にノード上の最後のOSD）はデータ再バランスをトリガーします。従って、残存ノード上のOSDに十分な空き容量が残っていることを確認してください。
4. 削除対象ノードからCephを削除するには、まずそのノードのOSDを順次破壊してください。
5. CEPHステータスが再びHEALTH\_OKになったら、次の手順を実行してください：

1. CephのWebインターフェース（→ CephFS）からメタデータサーバーを破壊するか、以下のコマンドを実行します：

```
# pveceph mds destroy <ローカルホスト名>;
```

2. そのモニターの破壊
3. そのマネージャーを破壊する

6. 最後に、空になったバケット（削除対象のProxmox VEノード）をCRUSH階層から削除します。以下のコマンドを実行してください：

```
# ceph osd crush remove <ホスト名>;
```

以下の例では、クラスターからノード hp4 を削除します。

別のクラスターノード（hp4以外）にログインし、削除するノードIDを特定するために pvecm nodes コマンドを実行します：

```
hp1# pvecm nodes

メンバーシップ情報
~~~~~  ノードID      投票  名前
        1          1  hp1 (ローカル)
        2          1  hp2
        3          1  hp3
        4          1  hp4
```

この時点で、hp4の電源を切り、現在の設定では（ネットワーク上で）再び電源が入らないことを確認してください。

**重要**

前述の通り、ノードの撤去前に電源を切断し、現在の設定のまま（既存のクラスターネットワーク内で）再起動しないことを確実にすることが極めて重要です。ノードを現状のまま再起動すると、クラスターが破損する可能性があり、正常な状態への復旧が困難になる恐れがあります。

ノード hp4 の電源を切った後、クラスタから安全に削除できます。

```
hp1# pvecm delnode hp4 ノード4を削除中
```

**注記**

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。これはノード削除の実際の失敗ではなく、corosync がオフラインノードを強制終了しようとした際の失敗を示しています。したがって、安全に無視できます。

ノードリストを再度確認するには、`pvecm nodes` または `pvecm status` を使用してください。以下のようないい表示になるはずです：

```
hp1# pvecm status
```

...

**Votequorum 情報**

~~~~~ 予想投票数:

3

最高予想: 3

総投票数: 3

定足数: 2

フラグ: 定足数あり

**会員情報**

~~~~~

**ノードID****投票 名前**

0x00000001	1 192.168.15.90 (ローカル)
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度同じクラスターに追加したい場合は、以下の手順を実行する必要があります：

- Proxmox VEを新規インストールし、
- その後、前のセクションで説明した手順に従ってクラスターに再参加させる

削除されたノードの設定ファイルは `/etc/pve/nodes/hp4` に残ります。必要な設定を復元した後、このディレクトリを削除してください。

**注意**

ノード削除後も、そのSSHフィンガープリントは他のノードの`known_hosts`に残存します。同一IP/ホスト名でノードを再参加させた際にSSHエラーが発生する場合は、再追加したノードで`pvecm updatecerts`を一度実行し、クラスター全体でフィンガープリントを更新してください。

## 5.5.1 再インストールせずにノードを分離する



### 注意

推奨される方法ではありません。慎重に実行してください。確信が持てない場合は前述の方法を使用してください。

クラスタからノードを分離する際、再インストールから始める必要はありません。ただし、クラスタからノードを削除した後も、共有ストレージへのアクセス権は残ります。ノードの削除を開始する前に、この問題を解決する必要があります。Proxmox VEクラスタは、ストレージロックがクラスタ境界を越えて機能しないため、別のクラスタと同一のストレージを共有できません。さらに、VMIDの競合を引き起こす可能性もあります。

分離対象ノードのみがアクセス可能な新規ストレージの作成が推奨されます。例として、NFS上の新規エクスポートや新規Cephプールなどが挙げられます。重要なのは、同一路由が複数クラスターからアクセスされないようにすることです。このストレージ設定後、対象ノードから全データとVMを移行してください。その後、ノードをクラスターから分離する準備が整います。



### 警告

共有リソースは完全に分離されていることを必ず確認してください！ さもなければ競合や問題が発生します。

まず、ノード上でcorosyncおよびpve-clusterサービスを停止します：

```
systemctl stop pve-clustersystemctl stop  
corosync
```

クラスタファイルシステムをローカルモードで再起動してください：

```
pmxcfs -l
```

corosync設定ファイルを削除：

```
rm /etc/pve/corosync.conf rm -r  
/etc/corosync/*
```

ファイルシステムを通常のサービスとして再起動できます：

```
killall pmxcfs  
systemctl start pve-cluster
```

ノードはクラスターから分離されました。クラスター内の残りのノードから削除するには以下を実行してください：

```
pvecm delnode oldnode
```

残りのノードでクオーラムが失われたためにコマンドが失敗した場合、回避策として期待投票数を1に設定できます：

```
pvecm expected 1
```

その後、`pvecm delnode` コマンドを再実行してください。

次に分離されたノードに戻り、残存するクラスタ関連ファイルを全て削除します。これにより、当該ノードを別のクラスタに問題なく再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クラスタファイルシステムに残っている可能性があるため、それらもクリーンアップすることをお勧めします。正しいノード名を確実に確認した後、`/etc/pve/nodes/NODENAME` からディレクトリ全体を再帰的に削除できます。

#### 注意

ノードのSSHキーは`authorized_keys`ファイルに残ります。これにより、ノード間では公開鍵認証による接続が依然として可能です。これを修正するには、`/etc/pve/priv/authorized_keys`ファイルから該当するキーを削除する必要があります。

## 5.6 クオーラム

Proxmox VE は、クオーラムベースの手法を用いて、すべてのクラスタノード間で一貫した状態を提供します。

クオーラムとは、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必要とする最小限の投票数である。

— ウィキペディア「クオーラム (分散コンピューティング)」より

ネットワーク分割が発生した場合、状態変更には過半数のノードがオンラインであることが必要となる。クオーラムを失った場合、クラスターは読み取り専用モードに切り替わる。

#### 注記

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 クラスタネットワーク

クラスターネットワークはクラスターの中核です。このネットワーク経由で送信されるすべてのメッセージは、各ノードに信頼性をもって、それぞれの順序で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、高可用性を実現する開発ツールキットの実装であるcorosyncによって処理されます。これは分散型設定ファイルシステム (`pmxcfs`) を支えています。

### 5.7.1 ネットワーク要件

Proxmox VEクラスタスタックは、すべてのノード間で5ミリ秒未満の遅延 (LANパフォーマンス) を持つ信頼性の高いネットワークを必要とし、これにより安定した動作が保証されます。ノード数が少ない環境では、より高い遅延を持つネットワークでも動作する可能性がありますが、これは保証されず、ノード数が3つを超えると、遅延が約10ミリ秒を超えると、動作はかなり困難になります。

他のメンバーによるネットワークの過度な使用は避けるべきです。corosyncは帯域幅を多く消費しませんが、遅延の変動に敏感です。理想的には、corosyncは物理的に分離された専用ネットワーク上で動作させるべきです。特に、corosyncとストレージを共有ネットワークで使用しないでください（冗長構成における潜在的な低優先度フォールバックとして使用する場合を除く）。

クラスタ構築前に、ネットワークが目的用途に適しているか確認することが推奨されます。ノード間がクラスタネットワーク上で接続可能か確認するには、pingツールで相互接続性をテストできます。

Proxmox VEのファイアウォールが有効な場合、corosync用のACCEPTルールは自動的に生成されます。手動での操作は不要です。

#### 注記

Corosyncはバージョン3.0以前（Proxmox VE 6.0で導入）ではマルチキャストを使用していました。最新バージョンではクラスタ通信にKronosnetに依存しており、現時点では通常のUDPユニキャストのみをサポートしています。

#### 注意

`corosync.conf` でトランSPORT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依然として可能ですが、これによりすべての暗号化および冗長性サポートが無効になることに留意してください。したがって、これは推奨されません。

## 5.7.2 ボンディング経由のCorosync

### 推奨事項

プライマリ Corosync リンクには、少なくとも 1 つの専用物理 NIC を推奨します（要件を参照）。冗長性を高めるための追加リンクとしてボンディングを使用できます。Corosync トライフィックにボンディングを使用する場合、以下の注意点があります：

- ボンディングモード `active-backup` は特定の障害シナリオにおいて期待される冗長性を提供しない可能性があります（詳細は後述）。
- Corosyncトライフィックには、バランスモード（`balance-rr`、`balance-xor`、`balance-tlb`、`balance-alb`）の使用を推奨しません。これらは特定の障害シナリオで問題を引き起こすことが知られています（詳細は後述）。
- IEEE 802.3ad (LACP):** LACPボンディングをcorosyncトライフィックに使用する場合、**Proxmox VEノードとスイッチの両方でbond-lacp-rateをfastに設定すること**を強く推奨します！デフォルト設定の`bond-lacp-rate slow`では、特定の障害シナリオにおいて問題が発生することが知られています。詳細は以下を参照してください。

### 背景

ボンディングをCorosyncリンクとして使用すると、特定の障害シナリオで問題が発生する可能性があります。ボンディングされたインターフェースの1つが障害を起こしパケット送信を停止しても、そのリンク状態はアップのまま維持され、他のCorosyncリンクが利用できないという障害シナリオを考えてみてください。このシナリオでは、一部のボンディングモードが非対称接続状態を引き起こす可能性があります。これによりクラスタノードは他のノードの異なるサブセットとしか通信できなくなります。影響を受けるのは負荷分散を提供するボンディングモードです。これらのモードは障害発生インターフェース経由でパケットのサブセットを送信しようとするためです。非対称接続が発生した場合、Corosyncはクラスタ内で安定したクーラムを形成できなくなる可能性があります

この状態が持続しHAが有効な場合、ボンディングに問題のないノードでさえフェンスされる可能性があります。最悪の場合、クラスタ全体がフェンスされる可能性があります。

上記障害シナリオにおいて、ボンディングモード **active-backup** は非対称接続を引き起こしません。ただし、障害発生インターフェースのボンディングがバックアップリンクに切り替わらない可能性があります。これによりノードはクラスタ接続を失い、HA が有効な場合フェンス状態となる恐れがあります。

結合モードが **balance-rr**、**balance-xor**、**balance\_tlb**、または **balance-alb** の場合、上記の障害シナリオで非対称接続が発生する可能性があり、HA が有効な場合に予期せぬフェンシングを引き起こす恐れがあります。

ボンディングモード **IEEE 802.3ad (LACP)** は、上記の障害シナリオにおいて非対称接続を引き起こす可能性がありますが、ボンディングの両側（Proxmox VE ノードとスイッチ）は、ボンディングされたインターフェース上で3つのLACPDUを連続して受信しなかった場合、そのインターフェースの使用を停止できるため、この状態から回復できます。ただし、デフォルト設定ではLACPDUは30秒ごとにしか送信されないため、フェイルオーバー時間は90秒となります。これは長すぎます。HAリソースを持つノードは、安定したクォーラムが約1分間維持されない場合、既にフェンシングを開始するからです。corosyncトライフィックにLACPボンディングを使用する場合、**Proxmox VEノードとスイッチの両方**でbond-lacp-rate fastを設定することを推奨します！片側でこのオプションを設定すると、相手側に毎秒LACPDU送信を要求します。双方で設定すれば、前述のシナリオにおけるフェイルオーバー時間を3秒に短縮でき、フェンシングを防止できます。

### 5.7.3 クラスタ専用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クラスタネットワークは通常、WebインターフェースやVMのネットワークと共有されます。設定によっては、ストレージトライフィックまで同じネットワーク経由で送信される可能性があります。corosyncは時間依存のリアルタイムアプリケーションであるため、この構成を変更することを推奨します。

#### 新規ネットワークの設定

まず、物理的に分離されたネットワーク上に新しいネットワークインターフェースを設定する必要があります。ネットワークがクラスターネットワーク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、新規クラスタ作成に使用する *pvecm create* コマンドの *linkX* パラメータで実現できます。

10.10.10.1/25 の静的アドレスで追加 NIC を設定し、すべてのクラスタ通信をこのインターフェース経由で送受信したい場合、以下を実行します：

```
pvecm create test --link0 10.10.10.1
```

正常に動作しているか確認するには、以下を実行してください:

```
systemctl status corosync
```

その後、上記の手順に従って分離されたクラスターネットワークを持つノードを追加します。

#### クラスタ作成後の分離

クラスタを既に作成済みで、クラスタ全体を再構築せずに通信を別のネットワークに切り替える場合に使用できます。この変更により、ノードがcorosyncを再起動し、新しいネットワーク上で順次起動する必要があるため、クラスタ内で短時間のクォーラム喪失が発生す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。その後、ファイルを開くと以下のようない内容が表示されるはずです：

```
logging { debug:  
    off  
    to_syslog: yes  
}  
  
nodelist {  
  
    node {  
        name: due  
        nodeid: 2  
        quorum_votes: 1  
        ring0_addr: due  
    }  
  
    node {  
        name:  
        nodeid: 3  
        quorum_votes: 1  
        ring0_addr: tre  
    }  
  
    node {  
        name: uno  
        nodeid: 1  
        quorum_votes: 1  
        ring0_addr: uno  
    }  
}  
  
quorum {  
    プロバイダー: corosync_votequorum  
}  
  
totem {  
    クラスタ名: testcluster  
    設定バージョン:  
    3IPバージョン: ipv4-6secauth: on  
    version:  
    2interface {  
        linknumber: 0  
    }  
}
```

---

#### 注記

`ringX_addr`は実際には**corosyncリンクアドレス**を指定します。「ring」という名称は、下位互換性のために残された古いcorosyncバージョンの名残です。

---

まず最初に行うべきことは、ノードエントリに名前プロパティを追加することです（既に存在しない場合）。これらはノード名と一致している必要があります。

次に、全ノードの `ring0_addr` プロパティから取得したアドレスを、新しいアドレスで置き換えます。ここでは単純な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、全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](#)）。

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るため、各ノードの各ノードの `ring0_addr` をそれぞれ変更します。

#### 注

まったく同じ手順で他の `ringX_addr` 値も変更できます。ただし、問題発生時の復旧を容易にするため、リンクアドレスは一度に1つだけ変更することを推奨します。

`config_version` プロパティを増やした後、新しい設定ファイルは次のようにになります：

```
logging { debug: off
          to_syslog: yes
      }

nodelist { node

    {
        name:
        duenodeid: 2
        quorum_votes: 1
        リング0アドレス: 10.10.10.2
    }

    node {
        name:
        trenodeid: 3
        クオーラム投票数: 1
        リング0アドレス: 10.10.10.3
    }

    node {
        name:
        unonodeid: 1
        クオーラム投票数: 1
        リング0アドレス: 10.10.10.1
    }

}

クオーラム {
    プロバイダー: corosync_votequorum
}

totem {
    クラスタ名: testcluster
```

```
config_version:  
4ip_version: ipv4-  
6secauth: onversion:  
2interface {  
    linknumber: 0  
}  
}
```

その後、変更した情報がすべて正しいことを最終確認し、保存します。そして再度、[corosync.confファイルの編集セクション](#)に従って変更を有効化します。

変更はライブで適用されるため、corosyncの再起動は厳密には不要です。他の設定も変更した場合や、corosyncがエラーを報告していることに気付いた場合は、任意で再起動をトリガーできます。

シングルノード環境では以下を実行：

```
systemctl restart corosync
```

正常動作を確認します：

```
systemctl status corosync
```

corosyncが再び動作し始めたら、他のすべてのノードでも再起動してください。その後、新しいネットワーク上でクラスターメンバーシップに順次参加します。

## 5.7.4 Corosync アドレス

Corosyncリンクアドレス（下位互換性のためcorosync.confでは*ringX\_addr*で表記）は2通りの方法で指定できます：

- **IPv4/v6アドレス**を直接使用できます。これらは静的であり、通常は不用意に変更されないため推奨されます。
- **ホスト名**は `getaddrinfo` を使用して解決されます。これはデフォルトで、利用可能な場合 IPv6 アドレスが優先的に使用されることを意味します (`man gai.conf` も参照)。既存クラスタを IPv6 にアップグレードする際は特にこの点に留意してください。

### 注意

ホスト名は慎重に使用してください。ホスト名が解決されるアドレスは、corosyncやそれを実行するノードを変更せずに変更される可能性があるためです。これにより、corosyncへの影響を考慮せずにアドレスが変更される状況が生じる恐れがあります。

ホスト名を優先する場合、corosync専用に別の静的ホスト名を設定することを推奨します。また、クラスタ内の全ノードが全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、ホスト名は入力時に解決されます（サポート対象ですが）。設定に保存されるのは解決されたIPアドレスのみです。

以前のバージョンでクラスターに参加したノードは、corosync.conf内で未解決のホスト名を使用している可能性があります。前述のように、それらをIPアドレスまたは別のホスト名に置き換えることをお勧めします。

## 5.8 Corosync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クをサポートします（レガシーのudp/udpuトランスポートではサポートされません）。これを有効化するには、

pvecm の `--linkX` パラメータ、GUI での **Link 1**（クラスタ作成時または新規ノード追加時）、または `corosync.conf` での複数の `ringX_addr` の指定によって有効化できます。

### 注記

有用なフェイルオーバーを実現するには、各リンクは個別の物理ネットワーク接続上に配置する必要があります。

リンクは優先度設定に従って使用されます。この優先度は、`corosync.conf` の対応するインターフェースセクションで `knet_link_priority` を設定するか、推薦される方法として pvecm でクラスタを作成する際に `priority` パラメータを使用して設定できます：

```
# pvecm create CLUSTERNAME --link0 10.10.10.1,priority=15 --link1↔
10.20.20.1,priority=20
```

これにより、優先度が高いリンク1が最初に使用されます。

手動で優先度が設定されていない場合（または2つのリンクの優先度が同じ場合）、リンクは番号順に使用され、番号が小さいほど優先度が高くなります。

すべてのリンクが正常に動作していても、優先度が最も高いリンクのみがCorosyncトラフィックを処理します。リンクの優先度は混在させることができず、異なる優先度を持つリンクは相互通信できません。

優先度の低いリンクは、より高い優先度のリンクがすべて障害を起こさない限りトラフィックを受け取らないため、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低優先度リンクとして指定する戦略が有効です。最悪の事態に備え、遅延が大きい、あるいは輻輳した接続であっても、全く接続がないよりもましな場合があります。

### 5.8.1 既存クラスタへの冗長リンク追加

稼働中の構成に新規リンクを追加するには、まず[corosync.confファイルの編集方法](#)を確認してください。

次に、nodelistセクション内の全ノードに新しい`ringX_addr`を追加します。追加する全ノードで`X`が同一であり、かつ各ノードで一意であることを確認してください。

最後に、トムセクションに下記のように新しいインターフェースを追加し、`X`を上記で選択したリンク番号に置き換えてください。

リンク番号1を追加した場合、新しい設定ファイルは次のようにになります：

```
logging { debug: off
          to_syslog: yes
      }

nodelist {node {
    name: due
    nodeid: 2
    quorum_votes: 1
    リング0アドレス: 10.10.10.2
}}
```

```
リング1アドレス: 10.20.20.2
}

node {
    name:
    trenoodeid: 3
    quorum_votes: 1
    リング0アドレス: 10.10.10.3
    リング1アドレス: 10.20.20.3
}

ノード {
    name:
    unonodeid: 1
    quorum_votes: 1
    リング0アドレス: 10.10.10.1
    リング1アドレス: 10.20.20.1
}

}

quorum {
    プロバイダ: corosync_votequorum
}

totem {
    クラスタ名: testcluster設定バージョン:
    4IPバージョン: ipv4-6secauth: on
    version:
    2interface {
        linknumber: 0
    }
    interface { linknumber: 1
    }
}
```

最後のステップに従って[corosync.confファイルを編集すると](#)、新しいリンクが有効になります。再起動は不要です。corosyncが新しいリンクをロードしたかどうかは

```
journalctl -b -u corosync
、以下で確認できます:
```

新しいリンクをテストするには、一時的に1ノードで古いリンクを切断し、切断中でもそのステータスがオンライン状態を維持していることを確認するのが良いで

```
pvecm status
しようと:
```

クラスタ状態が正常であれば、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クラスターにおけるSSHの役割

Proxmox VEは様々な機能でSSHトンネルを利用します。

- コンソール/シェルセッションのプロキシ（ノードおよびゲスト）  
ノードAに接続した状態でノードBのシェルを使用する場合、ノードA上のターミナルプロキシに接続し、そのプロキシが非対話型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およびCTメモリとローカルストレージの移行。  
移行中、移行情報交換およびメモリ・ディスク内容転送のため、ソースノードとターゲットノード間に1つ以上のSSHトンネルが確立される。
- ストレージレプリケーション

### 5.9.1 SSH 設定

Proxmox VE システムでは、SSH 設定/セットアップに対して以下の変更が行われます。

- root ユーザーの SSH クライアント設定が ChaCha20 より AES を優先するように設定されます
- rootユーザーのauthorized\_keysファイルが/etc/pve/priv/authorized\_keysにリンクされ、クラスタ内の全認証鍵が統合される
- sshd は root ユーザーがパスワードでログインできるように設定されています

---

#### 注記

古いシステムでは、/etc/ssh/ssh\_known\_hosts がシンボリックリンクとして設定され、すべてのノードホストキーを統合したバージョンを含む。このシステムは pve-cluster <INSERT VERSION> で明示的なホストキーピンニングに置き換えられた。シンボリックリンクがまだ存在する場合、pvecm updatecerts --unmerge-known-hosts を実行して設定を解除できる。

---

### 5.9.2 .bashrcおよび関連ファイルの自動実行による落とし穴

カスタムの.bashrcや、設定されたシェルによってログイン時に実行される類似のファイルがある場合、SSHはセッションが正常に確立されると自動的にそれらを実行します。これにより、上記の操作のいずれかでroot権限でコマンドが実行される可能性があり、予期しない動作を引き起こすことがあります。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！

このような問題を回避するため、/root/.bashrc に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、その場合にのみ .bashrc コマンドを実行することを推奨します。

.bashrcファイルの先頭に以下のスニペットを追加できます：

```
# 副作用回避のため、非対話的に実行されている場合は早期終了! case $- in
  * i* ) ;;
  * ) return;;
esac
```

## 5.10 Corosync 外部投票サポート

このセクションでは、Proxmox VEクラスタに外部投票者（external voter）を導入する方法を説明します。設定が完了すると、クラスタ通信の安全性を損なうことなく、より多くのノード障害に耐えられるようになります。

これを機能させるには、2つのサービスが関与します：

- 各 Proxmox VE ノード上で動作する 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

結果として、小規模な構成（例：2+1ノード）でも高可用性を実現できます。

### 5.10.1 QDeviceの技術概要

Corosync クオーラムデバイス（QDevice）は、各クラスタノード上で動作するデーモンです。外部で動作するサードパーティ製アービトレータの決定に基づき、設定された数の投票権をクラスタのクオーラムサブシステムに提供します。その主な用途は、標準のクオーラムルールが許容するよりも多くのノード障害をクラスタが耐えられるようにすることです。これは外部デバイスが全ノードを監視できるため安全に実現可能です。つまり、投票対象となるノードセットを単一のみ選択できます。この選択は、当該ノードセットがサードパーティ投票を受け取った後で（再び）クオーラムを維持できる場合にのみ行われます。

現在、サードパーティ・アービトレータ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に、クラスタパーティションに投票を提供するデーモンです。QDevice Netは、クラスタの1つのパーティションにのみ投票を行います。複数のクラスタをサポートするよう設計されており、ほぼ設定や状態を必要としません。新しいクラスタは動的に処理され、QDeviceを実行するホスト上で設定ファイルは不要です。

外部ホストに対する唯一の要件は、クラスタへのネットワークアクセスが可能であること、およびcorosync-qnetdパッケージが利用可能であることです。Debianベースのホスト向けにパッケージを提供しており、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ーを通じてパッケージが入手可能であるはずです。

---

#### 注記

corosync本体とは異なり、QDeviceはTCP/IP経由でクラスタに接続します。デーモンはクラスタのLAN外でも実行可能であり、corosyncの低遅延要件に縛られません。

---

### 5.10.2 サポート対象の構成

QDeviceはノード数が偶数のクラスタでサポートされており、高可用性を必要とする2ノードクラスタでの使用を推奨します。奇数ノード数のクラスタでは、現時点ではQDeviceの使用を推奨しません。その理由は、クラスタタイプごとにQDeviceが提供する投票数の差異にあります。偶数ノードクラスタでは追加投票が1票のみ増加し、可用性は向上するものの、QDevice自体が故障した場合、QDevice未使用時と同等の状態に陥ります。

一方、奇数ノード数のクラスタでは、QDeviceは( $N-1$ )票を提供します（ $N$ はクラスタノード数に対応）。この代替動作は合理的です。追加票が1票のみの場合、クラスタはスプリットブレイン状態に陥る可能性があります。このアルゴリズムでは、QDevice自体を除く全ノードが1つまで（当然ながらQDevice自体を除く）障害を起こすことが許容されます。ただし、これには2つの欠点があります：

- QNetデーモン自体が障害を起こした場合、他のノードは一切障害を起こせません。さもなければクラスタは即座にクオーラムを失います。例えば15ノードのクラスタでは、7ノードが障害を起こしてもクオーラムは維持されます。しかしここにQDeviceが設定されており、それが障害を起こした場合、15ノードのうち唯一のノードすら障害を起こせません。この場合、QDeviceは事実上单一障害点として機能します。
- QDeviceを除く全ノードが障害可能という点は一見魅力的に思えますが、これによりHAサービスの一斉復旧が発生し、残存ノード1台に過負荷がかかる可能性があります。さらにCephサーバーは、オンラインノードが $((N-1)/2)$ 以下になるとサービス提供を停止します。

これらの欠点と影響を理解した上で、奇数ノード構成のクラスターでこの技術を採用するかどうかを判断してください。

### 5.10.3 QDevice-Net 設定

corosync-qdevice に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推奨します。Proxmox VE および Debian には、既にそのように設定済みのパッケージが提供されています。Proxmox VE における QDevice の安全かつ確実な統合を確保するため、デーモンとクラスター間の通信は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
```

次に、すべてのクラスターノードに corosync-qdevice パッケージをインストールします

```
pve# apt install corosync-qdevice
```

これを実行した後、クラスタ内でのすべてのノードがオンラインであることを確認してください。

Proxmox VEノードのいずれかで以下のコマンドを実行して、QDeviceを設定できます:

```
pve# pvecm qdevice setup &lt;QDEVICE-IP&gt;
```

クラスタのSSHキーは自動的にQDeviceにコピーされます。

#### 注意

外部サーバーのrootユーザーに対して鍵ベースのアクセスを設定するか、設定フェーズ中は一時的にパスワードによるrootログインを許可してください。この段階で「*Host key verification failed.*」などのエラーが発生した場合、pvecm updatecerts を実行すると問題が解決する可能性があります。

すべての手順が正常に完了すると「Done」と表示されます。QDeviceの設定が完了したことを以下のコマンドで確認できます：

```
pve# pvecm status  
...  
  
Votequorum 情報  
~~~~~ 予想投票数  
: 3  
最高予想: 3  
総投票数: 3
```

クオーラム: 2  
 フラグ: 定足数に達した Qdevice

#### メンバーシップ情報

ノード ID	投票	Qdevice 名前
0x00000001	1	A, V, NMW 192.168.22.180 (ローカル)
0x00000002	1	A, V, NMW 192.168.22.181
0x00000000	1	Qdevice

#### QDevice ステータスフラグ

上記のQDeviceの状態出力には通常、3つの列が含まれます:

- A / NA: Alive（稼働中）または Not Alive（非稼働）。外部 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がノードに対して投票を行うかどうか。ノード間のCorosync接続がダウンしているが、両ノードとも外部Corosync-qnetdデーモンと通信可能なスプリットブレイン状態では、投票権は1つのノードのみに付与される。
- MW / NMW: マスターが勝利する(MV)か否か(NMW)。デフォルトはNMW、[\(1\)を参照](#)。
- NR: QDeviceは登録されていない。

#### 注記

QDeviceがNot Alive（上記の出力ではNA）と表示される場合、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

### 同点の場合

同サイズのクラスタパーティションが互いに認識できないが、QDevice を認識できる場合、QDevice はそれらのパーティションからランダムに 1 つを選択し、それに投票します。

### 潜在的な悪影響

ノード数が偶数のクラスターでは、QDeviceを使用しても悪影響はありません。動作しない場合、QDeviceが存在しない場合と同じです。

### QDevice設定後のノード追加/削除

QDeviceが設定されたクラスターに新規ノードを追加、または既存ノードを削除する場合は、まずQDeviceを削除する必要があります。その後、通常通りノードの追加・削除を行えます。ノード数が再び偶数になった時点で、前述の手順に従いQDeviceを再設定できます。

<sup>1</sup> votequorum\_qdevice\_master\_wins マニュアル  
[votequorum\\_qdevice\\_master\\_wins.3.en.html](https://manpages.debian.org/stable/libvotequorum-dev/votequorum_qdevice_master_wins.3.en.html)

ページ [https://manpages.debian.org/stable/libvotequorum-dev/votequorum\\_qdevice\\_master\\_wins.3.en.html](https://manpages.debian.org/stable/libvotequorum-dev/votequorum_qdevice_master_wins.3.en.html)

## QDevice の削除

公式の pvecm ツールを使用して QDevice を追加した場合、以下のコマンドを実行して削除できます:

```
pve# pvecm qdevice remove
```

## 5.11 Corosync設定

/etc/pve/corosync.conf ファイルは Proxmox VE クラスターの中核的な役割を担います。クラスターのメンバーシップとネットワークを制御します。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
```

ノードのメンバーシップについては、常に Proxmox VE が提供する pvecm ツールを使用してください。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場合があります。その際のベストプラクティスをいくつか紹介します。

### 5.11.1 corosync.conf の編集

corosync.conf ファイルの編集は必ずしも簡単ではありません。各クラスタノードには2つのファイルが存在し、1つは /etc/pve/corosync.conf と /etc/corosync/corosync.conf にそれぞれ存在します。クラスタファイルシステム上のファイルを編集すると変更はローカルファイルに反映されますが、逆は適用されません。

ファイルが変更されると、設定は自動的に更新されます。これは、実行中の corosync に統合可能な変更が即座に有効になることを意味します。したがって、編集中にファイルを保存した際に意図しない変更を引き起こすのを避けるため、常にコピーを作成し、そのコピーを編集するようにしてください。

```
cp /etc/pve/corosync.conf /etc/pve/corosync.conf.new
```

次に、設定ファイルを nano や vim.tiny など、お好みのエディタで開きます。これらのエディタはすべての Proxmox VE ノードにプリインストールされています。

#### 注意

設定変更後は必ず config\_version の番号をインクリメントしてください。これを怠ると問題が発生する可能性があります。

必要な変更を加えた後、現在の作業用設定ファイルの別のコピーを作成します。これは、新しい設定の適用に失敗した場合や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。

次に、古い設定ファイルを新しいものと置き換えます:

```
mv /etc/pve/corosync.conf.new /etc/pve/corosync.conf
```

変更が自動的に適用されたかどうかは、以下のコマンドで確認できます:

```
systemctl status corosync journalctl -b -u corosync
```

変更を自動的に適用できなかった場合、corosyncサービスを再起動する必要がある可能性があります。以下のコマンドを実行してください:

```
systemctl restart corosync
```

エラーが発生した場合は、以下のトラブルシューティング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` の設定が必要です

corosync の起動に失敗し、システムログに以下のメッセージが表示される場合：

```
[...]
corosync[1647]:           [QUORUM] クオーラムプロバイダ: corosync_votequorum が失敗しました ←
    initialize.

corosync[1647]:           [SERV      ] サービスエンジン 'corosync_quorum' のロードに失敗しました          ←
    理由

    設定エラー: nodelist または quorum.expected_votes は ←
    設定してください！

[...]
```

これは、設定で corosync リングの `X_addr` に設定したホスト名が解決できなかったことを意味します。

### クオーラム未達成時の設定書き込み

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、その操作内容を理解している場合は、以下を使用してください:

```
pvecm expected 1
```

これにより、期待される投票数が1に設定され、クラスターがクオーラート状態になります。その後、設定を修正するか、最後に正常に動作したバックアップ状態に戻すことができます。

corosyncが起動しなくなった場合、これだけでは不十分です。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、corosyncが再起動できるようにするのが最善策です。スプリットブレイン状態を避けるため、全ノードでこの設定内容が同一であることを確認してください。

## 5.11.3 Corosync設定用語集

### ringX\_addr

これは、ノード間の Kronosnet 接続のさまざまなリンクアドレスに名前を付けます。

## 5.12 クラスタのコールドスタート

全ノードがオフラインの場合、クラスタがクオーラート状態にないことは明らかです。これは停電後の一般的なケースです。

### 注

特に高可用性 (HA) を実現したい場合には、この状態を回避するため、無停電電源装置 (UPS、別名「バッテリーバックアップ」) を使用することが常に推奨されます。

ノード起動時、`pve-guests` サービスが開始され、クオーラムを待機します。クオーラムが成立すると、`onboot` フラグが設定されている全てのゲストを起動します。

ノードの電源投入時や停電後の復電時には、一部のノードが他より早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起動が遅延される点にご留意ください。

## 5.13 ゲスト VMID 自動選択

新規ゲスト作成時、Webインターフェースは自動的にバックエンドに空きVMIDを要求します。デフォルトの検索範囲は100～1000000（スキーマで強制される最大許容VMID未満）です。

管理者は、例えば一時的なVMと手動でVMIDを選択するVMを簡単に分離するため、新しいVMIDを別の範囲で割り当てる場合があります。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100000などに設定することでより多くの余裕が生まれます。

このユースケースに対応するため、`datacenter.cfg` 設定ファイルで下限、上限、または両方の境界を設定できます。

設定ファイルで下限、上限、または両方の境界を設定できます。この設定ファイルは、Webインターフェースの「データセンター」→「→ オプション」で編集可能です。

---

### 注記

この範囲は`next-id` API呼び出しにのみ使用されるため、厳密な制限ではありません。

---

## 5.14 ゲストの移行

クラスタ環境において、仮想ゲストを他のノードへ移行する機能は有用です。このような移行の動作を制御する設定が存在します。これは設定ファイル `datacenter.cfg` 経由、または特定の移行に対しては API やコマンドラインパラメータ経由で実施可能です。

ゲストがオンラインかオフラインか、ローカルリソース（ローカルディスクなど）を所有しているかによって動作が異なります。仮想マシン

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](#)」の章を参照してください。

### 5.14.1 移行タイプ

移行タイプは、移行データを暗号化（セキュア）チャネル経由で送信するか、非暗号化（非セキュア）チャネル経由で送信するかを定義します。移行タイプを非セキュアに設定すると、仮想ゲストのRAM内容も非暗号化で転送されるため、ゲスト内部の機密データ（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、盗聴がないことを保証できない場合は、安全なチャネルの使用を強くお勧めします。

---

### 注

ストレージ移行はこの設定に従いません。現在、ストレージの内容は常にセキュアなチャネル経由で送信されます。

---

暗号化には膨大な計算能力が必要であるため、パフォーマンス向上のためにこの設定が安全でない状態に変更されることが多い。現代のシステムではAES暗号化をハードウェアで実装しているため、その影響は小さくなっている。特に高速ネットワーク（10Gbps以上の転送が可能）では、パフォーマンスへの影響が顕著に現れる。

## 5.14.2 移行ネットワーク

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移行トラフィックの送信に使用します。これは、機密性の高いクラスタ通信が妨げられる可能性があり、またこのネットワークがノード上で利用可能な最適な帯域幅を提供しない可能性があるため、最適とは言えません。

移行ネットワークパラメータを設定することで、すべての移行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ります。メモリに加え、オフライン移行時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表記を用いたネットワークとして設定されます。これにより、各ノードに個別のIPアドレスを設定する必要がなくなります。Proxmox VEは、CIDR形式で指定されたネットワークから宛先ノードの実際のアドレスを自動的に特定できます。これを有効にするには、各ノードが該当ネットワーク内で正確に1つのIPアドレス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

3ノード構成で、3つの独立したネットワークが存在すると仮定します。1つはインターネットとの公開通信用、1つはクラスタ通信用、そして移行専用の高速ネットワークとして使用したい非常に高速なネットワークです。

このような構成のネットワーク設定例は以下の通りです：

```
iface eno1 inet manual

# 公開ネットワーク auto 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 auto
eno2
iface eno2 inet static address
    10.1.1.1/24

# 高速ネットワーク auto
eno3
iface eno3 inet static address
    10.1.2.1/24
```

ここでは、移行ネットワークとして10.1.2.0/24ネットワークを使用します。単一の移行については、コマンドラインツールのmigration\_network/パラメータを使用して次のように実行できます：

```
# qm migrate 106 tre --online --migration_network 10.1.2.0/24
```

クラスタ内の全移行におけるデフォルトネットワークとして設定するには、

/etc/pve/datacenter.cfg ファイルの migration プロパティを設定します：

```
# 専用移行ネットワークを使用 migration:
secure,network=10.1.2.0/24
```

---

**注記**

移住　　移行　　type　　は　　常に　　設定　　設定　　設定　　設定　　ネットワーク　　ネットワーク　　は　　設定　　in  
/etc/pve/datacenter.cfg に設定されています。

---

## 第6章

# Proxmox クラスタファイルシステム (pmxcfs)

Proxmox クラスタファイルシステム（「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ファイルシステムであり、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複製されます。これを使用して、すべての Proxmox VE 関連の設定ファイルを保存します。

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保存しますが、データのコピーはRAMにも存在します。これにより最大サイズに制限が生じ、現在は128 MiBです。それでも数千台の仮想マシンの設定を保存するには十分な容量です。

このシステムは以下の利点を提供します：

- すべての設定を全ノードへリアルタイムでシームレスに複製
- 重複するVM IDを回避するための強力な整合性チェックを提供
- ノードがクオーラムを失った際の読み取り専用モード
- Corosyncクラスタ構成の全ノードへの自動更新
- 分散ロック機構を含む

## 6.1 POSIX互換性

ファイルシステムはFUSEを基盤としているため、動作はPOSIXに準拠しています。ただし、以下の機能は不要なため実装されていません：

- 通常のファイルやディレクトリは生成できますが、シンボリックリンクは生成できません。
- 空でないディレクトリの名前変更はできません（これによりVMIDの一意性を保証しやすくなります）。
- ファイルの権限を変更できません（権限はパスに基づいています）
- O\_EXCL による作成はアトミックではありません（旧式 NFS と同様）
- O\_TRUNC による作成はアトミックではありません（FUSEの制限）

## 6.2 ファイルアクセス権限

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` に属します。`root` のみが書き込み権限を持ちますが、グループ `www-data` はほとんどのファイルを読み取れます。以下のパス以下のファイルは `root` のみがアクセス可能です:

```
/etc/pve/priv/
/etc/pve/nodes/${NAME}/priv/
```

## 6.3 技術

クラスタ通信には[Corosyncクラスタエンジン](#)を、データベースファイルには[SQLite](#)を使用しています。ファイルシステムは[FUSE](#)を用いてユーザー空間で実装されています。

## 6.4 ファイルシステム構成

ファイルシステムは以下にマウントされます:

```
/etc/pve
```

### 6.4.1 ファイル

<code>authkey.pub</code>	チケットシステムで使用される公開鍵
<code>ceph.conf</code>	Ceph 設定ファイル (注: <code>/etc/ceph/ceph.conf</code> はシンボリックリンクです)
<code>corosync.conf</code>	Corosync クラスタ設定ファイル (Proxmox VE 4.x では、このファイルは <code>cluster.conf</code> と呼ばれていました)
<code>datacenter.cfg</code>	Proxmox VE データセンター全体の設定 (キーボードレイアウト、プロキシ、...)
<code>domains.cfg</code>	Proxmox VE 認証ドメイン
<code>firewall/cluster.fw</code>	ファイアウォール設定 (全ノードに適用)
<code>firewall/&lt;NAME&gt;.fw</code>	個々のノードに対するファイアウォール設定
<code>firewall/&lt;VMID&gt;.fw</code>	VM およびコンテナ用のファイアウォール設定
<code>ha/crm_commands</code>	現在実行中の HA 操作を表示 実行中の HA 操作を表示
<code>ha/manager_status</code>	クラスタ上の HA サービスに関する JSON 形式の情報 クラスタ上のサービスに関する
<code>ha/resources.cfg</code>	高可用性によって管理されるリソース、およびそれらの現在の状態
<code>ha/rules.cfg</code>	HAマネージャーの HAリソースのスケジューリングに関する制約ルール
<code>nodes/&lt;NAME&gt;/config</code>	ノード固有の設定
<code>nodes/&lt;NAME&gt;/lxc/&lt;VMID&gt;.conf</code>	LXCコンテナのVM構成データ

nodes/&lt;NAME&gt;/openvz/	Proxmox VE 4.0 以前では、コンテナ 設定データ用（非推奨、まもなく削除予定）
nodes/&lt;NAME&gt;/pve-ssl.key	pve-ssl.pem のプライベート SSL キー
nodes/&lt;NAME&gt;/pve-ssl.pem	Web サーバー用の公開 SSL 証明書（クラスタ CA によって署名済み） クラスタCAによって署名された）
nodes/&lt;NAME&gt;/pveproxy-ssl.key	pveproxy-ssl.pem の秘密 SSL キー (オプション)
nodes/&lt;NAME&gt;/pveproxy-ssl.pem	Web サーバー用パブリック SSL 証明書（チェーン） (pve-ssl.pem のオプション上書き)
nodes/&lt;NAME&gt;/qemu-server/&lt;VMID&gt;.co	KVM VM用のVfM 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	認証用のクラスタメンバーのSSH鍵
priv/ceph*	Ceph 認証キーおよび関連する 機能
priv/known_hosts	検証用クラスタメンバーのSSHキー
priv/lock/*	安全なクラスター全体の操作を保証するために様々なサービスが使用するロックファイル  クラスタ全体の操作を安全に実行するために各種サービスが使用するロックファイル
priv/pve-root-ca.key	クラスタCAの秘密鍵
priv/shadow.cfg	PVE Realm ユーザー用のシャドウパスワードファイル
priv/storage/&lt;STORAGE-ID&gt;.pw	ストレージのパスワードを平文で含む
priv/tfa.cfg	Base64エンコードされた二要素認証 設定
priv/token.cfg	すべてのトークンの API トークンシークレット
pve-root-ca.pem	クラスタ CA の公開証明書
pve-www.key	CSRF トークン生成に使用される秘密鍵
sdn/*	ソフトウェア定義ネットワーク（SDN）の共有設定ファイル Networking (SDN) 用共有設定ファイル
status.cfg	Proxmox VE 外部メトリクスサーバー設定
storage.cfg	Proxmox VE ストレージ設定
user.cfg	Proxmox VE アクセス制御設定 (users/groups/ ...)
virtual-guest/cpu-models.conf	カスタム CPU モデルを保存するため
vzdump.cron	クラスタ全体の vzdump バックアップジョブのスケジュール

## 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は、ノード自身の設定ファイルを指すためにシンボリックリンクを使用します。したがって、以下の表で参照されているファイルは、クラスタの各ノード上で異なるファイルを指します。

local	nodes/&lt;LOCAL_HOST_NAME&gt;
lxc	nodes/&lt;LOCAL_HOST_NAME&gt;/lxc/
openvz	nodes/&lt;LOCAL_HOST_NAME&gt;/openvz/ (非推奨、まもなく削除予定)
qemu-server	nodes/&lt;LOCAL_HOST_NAME&gt;/qemu-server

/

## 6.4.3 デバッグ用特殊ステータスファイル (JSON)

.version	ファイルバージョン（ファイル変更を検出するため）
.members	クラスタメンバーに関する情報
.vmlist	すべての仮想マシンのリスト
.clusterlog	クラスタログ（直近50エントリ）
.rrd	RRDデータ（最新のエントリ）

#### 6.4.4 デバッグの有効化/無効化

詳細なsyslogメッセージを有効にするには以下を実行します:

```
echo "1" >/etc/pve/.debug
```

詳細なsyslogメッセージを無効にするには:

```
echo "0" >/etc/pve/.debug
```

## 6.5 復旧

Proxmox VEホストにハードウェア障害などの重大な問題が発生した場合、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復旧に役立ちます。新しいホスト（何も実行されていない状態）では、`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（必要な権限は 0600）。その後、失われた Proxmox VE ホストに合わせて `/etc/hostname` および `/etc/hosts` を失われたProxmox VEホストに合わせて変更し、再起動して確認してください（VM/CTデータのバックアップを忘れないでください）。

### 6.5.1 クラスタ設定の削除

推奨される方法は、クラスタからノードを削除した後、再インストールすることです。これにより、すべての秘密のクラスタ/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すことを選択する場合があります。これについては「[再インストールせずにノードを分離する](#)」で説明されています。

### 6.5.2 障害発生ノードからのゲストの復旧/移動

ノード`/<NAME>/qemu-server/` (VM) およびノード`/<NAME>/lxc/` (コンテナ) 内のゲスト設定ファイルについて、Proxmox VE は、そのゲストを所有するノード`<NAME>` を認識します。この概念により、ゲスト設定の同時変更を防ぐために、高コストなクラスタ全体のロックではなく、ローカルロックを使用することが可能になります。

結果として、ゲストの所有ノードが障害発生（停電、フェンシングイベントなど）した場合、通常のマイグレーションは不可能となります（全ディスクが共有ストレージ上にある場合でも）。これは（オフライン状態の）所有ノード上のローカルロックを取得できないためです。これは HA 管理ゲストには問題になりません。Proxmox VE の高可用性スタックには、フェンスされたノードからのゲストの正確かつ自動的なリカバリを保証するために必要な（クラスタ全体の）ロックおよびウォッチドッグ機能が含まれているからです。

非HA管理ゲストが共有ディスクのみ（障害発生ノードにのみ存在するローカルリソースなし）を使用している場合、ゲスト設定ファイルを障害発生ノードから移動するだけで手動復旧が可能です。

/etc/pve/ 内のノードディレクトリをオンラインノードのディレクトリに変更します（これによりゲストの論理所有者または場所が変更されます）。

例えば、オフラインノード1から別のノード2へID 100のVMを復旧するには、クラスタ内の任意のメンバノードでroot権限で以下のコマンドを実行します：

```
mv /etc/pve/nodes/node1/qemu-server/100.conf /etc/pve/nodes/node2/leftrightarrow  
qemu-server/
```

---

**警告**

ゲストを手動でこのように復旧する前に、障害発生元のノードが確実に電源オフ/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、mvコマンドによってProxmox VEのロック原理が破られ、予期せぬ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その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。障害発生ノードがクラスターに再参加するのを待つか、バックアップから該当ゲストを復元してください。

---

## 第7章

# Proxmox 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に保存することも、NFSやiSCSI（NAS、SAN）などの共有ストレージに保存することもできます。制限はなく、好きなだけ多くのストレージプールを設定できます。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。

共有ストレージに仮想マシンを保存する主な利点は、クラスター内の全ノードが仮想マシンのディスクイメージに直接アクセスできるため、稼働中のマシンをダウントIMEなしでライブマイグレーションできることです。仮想マシンイメージデータをコピーする必要がないため、この場合ライブマイグレーションは非常に高速です。

ストレージライブラリ（パッケージ `libpve-storage-perl`）は、柔軟なプラグインシステムを用いて、すべてのストレージタイプに共通のインターフェースを提供します。これにより、将来的にさらなるストレージタイプを容易に追加できるようになっています。

## 7.1 ストレージタイプ

ストレージタイプは基本的に2つの異なるクラスに分類されます：

### ファイルレベルストレージ

ファイルレベルベースのストレージ技術は、完全な機能を備えた（POSIX準拠の）ファイルシステムへのアクセスを可能にします。これらは一般的に、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ます。ZFSはおそらく最も先進的なシステムであり、スナップショットとクローンを完全にサポートしています。

### ブロックレベルストレージ

大規模な生イメージの保存を可能にします。通常、このようなストレージタイプには他のファイル（ISO、バックアップなど）を保存することはできません。最新のブロックレベルストレージの実装のほとんどは、スナップショットとクローンをサポートしています。Ceph RADOSは分散システムであり、ストレージデータを異なるノードに複製し、RBD（RADOSブロックデバイス）としてアクセスできます。

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	スナップショット	安定
ZFS (ローカル)	<code>zfspool</code>	両方 <sup>1</sup>	いいえ	はい	はい
ディレクトリ	<code>dir</code>	ファイル	いいえ	はい <sup>2</sup>	はい
BTRFS	<code>btrfs</code>	ファイル	いいえ	はい	TP <sup>5</sup>

表 7.1: (続き)

説明	プラグインタイプ	レベル	共有	スナップショット	安定
NFS	nfs	ファイル	はい	はい <sup>2</sup>	はい
CIFS	cifs	ファイル	はい	はい <sup>2</sup>	はい
Proxmox Backup	pbs	両方	はい	該当なし	はい
CephFS	cephfs	file	はい	はい	はい
LVM	lvm	ブロック	いいえ <sup>3</sup>	はい <sup>4</sup>	はい
LVM-thin	lvmthin	ブロック	いいえ	はい	はい
iSCSI/カーネル	iscsi	ブロック	はい	いいえ	はい
iSCSI/libiscsi	iscsidirec	tblock	はい	いいえ	はい
Ceph/RBD	rbd	ブロック	はい	はい	はい
ZFS over iSCSI	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、ブロックデバイス機能を提供する ZFS ボリューム (zvol) データセットに保存されます。

<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。内部のスナップショット機能を使用するか、ボリュームチェーン<sup>4</sup>としてスナップショットを作成します。

<sup>3</sup>: iSCSIまたはFCベースのストレージ上にLVMを使用することが可能です。これにより共有LVMストレージを実現できます

<sup>4</sup>: Proxmox VE 9以降、VM向けにボリュームチェーン形式のスナップショットが利用可能になりました。これらのスナップショットはスナップショットデータ用に別個のボリュームを使用し、それらを階層化します。詳細はLVM設定セクションの「snapshot-as-volume-chain」の説明を参照してください。

<sup>5</sup> ベストエフォートサポートによる技術プレビュー。

## 7.1.1 シンプロビジョニング

多くのストレージおよびQEMUイメージ形式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のハードディスクを持つVMを作成し、ゲスト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場合、ゲストVMが32GBのハードドライブを認識していても、ストレージに書き込まれるのは3GBのみです。このように、シンプロビジョニングにより、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できます。VM用に大容量ディスクイメージを作成し、必要が生じた際にVMのファイルシステムを再サイズすることなくストレージにディスクを追加できます。

「スナップショット」機能を備えた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しています。

### 注意

ストレージが満杯になると、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でIOエラーが発生します。これによりファイルシステムの整合性が損なわれ、データが破損する可能性があります。したがって、ストレージリソースの過剰プロビジョニングを避けるか、空き容量を注意深く監視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関連のストレージ設定はすべて、`/etc/pve/storage.cfg` という単一のテキストファイル内に保存されます。このファイルは `/etc/pve/` ディレクトリ内にあるため、自動的にすべてのクラスターノードに配布されます。したがって、すべてのノードが同一のストレージ設定を共有します。

共有ストレージの場合、同じ「共有」ストレージが全ノードからアクセス可能であるため、設定を共有することは理にかなっています。しかしローカルストレージタイプにも有用です。この場合、そのようなローカルストレージは全ノードで利用可能ですが、物理的に異なり、全く異なる内容を持つ可能性があります。

### 7.2.1 ストレージプール

各ストレージプールには `<type>` が設定され、`<STORAGE_ID>` によって一意に識別されます。プール設定は次のようにになります：

```
<type>: <STORAGE_ID>
  <property>: <value>
  <property>: <value>
  <property>;
...
<type>: <STORAGE_ID>
```

行でプール定義が始まり、その後プロパティのリストが続きます。ほとんどのプロパティには値が必要です。合理的なデフォルト値が設定されているものもあり、その場合は値を省略できます。

具体的には、インストール後のデフォルトストレージ構成を確認してください。これには `local` という名前の特別なローカルストレージプールが含まれており、`/var/lib/vz` ディレクトリを指し、常に利用可能です。Proxmox VE インストーラーは、インストール時に選択され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: local
  path /var/lib/vz
  content iso,vztmpl,backup

# LVMベースのインストールにおけるデフォルトイメージストア lvmthin: local-lvm
  thinpool data vgname
  pve
  content rootdir,images

# ZFSベースのインストールにおけるデフォルトイメージストア zfspool: local-zfs
  pool rpool/data sparse
  content images,rootdir
```

**注意**

同一の基盤ストレージを指す複数のストレージ構成が存在することは問題です。このようなエイリアスされたストレージ構成は、同一のディスクイメージを指す2つの異なるボリュームID (*valid*) を生成する可能性があります。Proxmox VEは、イメージが指すボリュームIDが一意であることを前提としています。エイリアスされたストレージ構成で異なるコンテンツタイプを選択することは可能ですが、推奨されません。

## 7.2.2 共通ストレージプロパティ

いくつかのストレージプロパティは、異なるストレージタイプ間で共通しています。

**nodes**

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、ストレージアクセスを限定されたノードセットに制限できます。

**content**

ストレージは複数のコンテンツタイプをサポートします。例：仮想ディスクイメージ、CD-ROM 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。全てのストレージタイプが全てのコンテンツタイプをサポートするわけではありません。このプロパティを設定することで、当該ストレージの使用目的を選択できます。

**images**

QEMU/KVM VMイメージ。

**rootdir**

コンテナデータの保存を許可します。

**vztmp1**

コンテナテンプレート。

**backup**

バックアップファイル (`vzdump`)。

**iso**

ISOイメージ

**スニペット**

スニペットファイル、例えばゲストフックスクリプト

**共有**

これは単一のストレージであり、すべてのノード（または `nodes` オプションに列挙されたすべてのノード）で同じ内容を持つことを示します。ローカルストレージの内容を他のノードが自動的にアクセス可能にするものではなく、既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！

**無効化**

このフラグを使用すると、ストレージを完全に無効化できます。

**prune-backups**

バックアップの保持期間オプション。詳細は「[バックアップ保持](#)」を参照。

**format**

デフォルトのイメージ形式 (raw | qcow2 | vmdk)

**preallocation**

ファイルベースストレージ上のrawおよびqcow2イメージに対する事前割り当てモード (off | metadata | fallow | full)。デフォルトはmetadataで、rawイメージではoffと同様に扱われます。大規模なqcow2イメージとネットワークストレージを併用する場合、offを使用するとタイムアウト回避に有効です。

**警告**

異なるProxmox VEクラスタ間で同一のストレージプールを使用することは推奨されません。一部のストレージ操作では排他アクセスが必要となるため、適切なロック処理が求められます。これは同一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 ボリューム

ストレージデータへの参照には特別な表記法を使用します。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が返されます。ボリュームは <STORAGE\_ID> で識別され、その後に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が続きます。有効な <VOLUME\_ID> の形式は以下の通りです:

```
local:230/example-image.rawlocal:iso/debian-501-amd64-
```

```
netinst.iso
```

```
local:vztmp1/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、以下を使用します:

```
pvesm path <VOLUME_ID>
```

### 7.3.1 ボリュームの所有権

イメージタイプのボリュームには所有権関係が存在します。各ボリュームはVMまたはコンテナによって所有されます。例えば、ボリューム local:230/example-image.raw はVM 230によって所有されています。ほとんどのストレージバックエンドは、この所有権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、システムはそのVMまたはコンテナが所有する関連するすべてのボリュームも削除します。

## 7.4 コマンドラインインターフェースの使用

ストレージプールやボリューム識別子の概念を理解しておくことは推奨されますが、実際の運用ではコマンドラインでこのような低レベル操作を強制されることはありません。通常、ボリュームの割り当てや削除はVMおよびコンテナ管理ツールによって行われます。

しかしながら、pvesm（「Proxmox VE Storage Manager」）と呼ばれるコマンドラインツールが存在し、一般的なストレージ管理タスクを実行することが可能です。

## 7.4.1 例

ストレージプールの追加

```
pvesm add &lt;TYPE&gt; &lt;STORAGE_ID&gt; &lt;OPTIONS&gt;  
pvesm add dir &lt;STORAGE_ID&gt; --path &lt;PATH&gt;  
pvesm add nfs &lt;STORAGE_ID&gt; --path &lt;PATH&gt; --server &lt;SERVER&gt; --export -->  
     &lt;EXPORT&gt;  
pvesm add lvm &lt;STORAGE_ID&gt; --vgname &lt;VGNAME&gt;  
pvesm add iscsi &lt;STORAGE_ID&gt; --portal &lt;HOST[:PORT]&gt; --target &lt;TARGET -->  
     >
```

ストレージプールを無効化

```
pvesm set &lt;STORAGE_ID&gt; --disable 1
```

ストレージプールを有効化

```
pvesm set &lt;STORAGE_ID&gt; --disable 0
```

ストレージオプションの変更/設定

```
pvesm set &lt;STORAGE_ID&gt; &lt;OPTIONS&gt; pvesm  
set &lt;STORAGE_ID&gt; --shared 1 pvesm set local  
--format qcow2 pvesm set &lt;STORAGE_ID&gt; --  
content iso
```

ストレージプールを削除します。これによりデータは削除されず、接続解除やアンマウントも行われません。ストレージ設定のみが削除されます。

```
pvesm remove &lt;STORAGE_ID&gt;
```

ボリュームの割り当て

```
pvesm alloc &lt;STORAGE_ID&gt; &lt;VMID&gt; &lt;name&gt; &lt;size&gt; [--format &lt;raw|qcow2&gt;]
```

ローカルストレージに4Gのボリュームを割り当てます。&lt;name&gt;に空文字列を渡すと名前は自動生成されます。pvesm alloc local  
&lt;VMID&gt; '' 4G

ボリュームを解放します

```
pvesm free &lt;VOLUME_ID&gt;
```



### 警告

これにより、ボリュームのデータは完全に破壊されます。

ストレージの状態を一覧表示

```
pvesm status
```

ストレージの内容を一覧表示

```
pvesm list <STORAGE_ID> [--vmid <VMID>]
```

VMIDによって割り当てられたボリュームの一覧

```
pvesm list <STORAGE_ID> --vmid <VMID>;
```

ISOイメージの一覧表示

```
pvesm list <STORAGE_ID> --content iso
```

コンテナテンプレートのリスト表示

```
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

```
pvesm path <VOLUME_ID>;
```

ボリューム local:103/vm-103-disk-0.qcow2 をファイル target にエクスポートします。これは主に pvesm import 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。したがって、エクスポートされたファイルは単純に VM にアタッチできません。これは他の形式にも当てはまります。

```
pvesm export local:103/vm-103-disk-0.qcow2 qcow2+size target --with-↔  
スナップショット 1
```

## 7.5 ディレクトリバックエンド

ストレージプールタイプ: dir

Proxmox VEはストレージとしてローカルディレクトリまたはローカルマウントされた共有を使用できます。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、あらゆるコンテンツタイプを保存できます。

---

### 注

標準的な Linux の /etc/fstab 経由で追加ストレージをマウントし、そのマウントポイントに対してディレクトリストレージを定義できます。これにより、Linux がサポートするあらゆるファイルシステムを利用可能です。

---

このバックエンドは、基盤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ますが、それ以外の要件はありません。これは、ストレージレベルでのスナップショット作成ができないことを意味します。ただし、qcow2ファイル形式を使用するVMイメージについては、この形式が内部的にスナップショットをサポートしているため、回避策が存在します。

---

### ヒント

一部のストレージタイプはO\_DIRECTをサポートしていないため、そのようなストレージではキャッシングモードnoneを使用できません。代わりにキャッシングモードwritebackを使用してください。

---

異なるコンテンツタイプを別々のサブディレクトリに保存するため、事前定義されたディレクトリレイアウトを使用します。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採用されています。

表 7.2: ディレクトリレイアウト

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	template/iso/
コンテナテンプレート	template/cache/
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 設定

このバックエンドは、すべての一般的なストレージプロパティをサポートし、さらに2つのプロパティを追加します。 パスプロパティはディレクトリを指定するために使用されます。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` プロパティにより、デフォルトのレイアウトを変更できます。これは以下の形式の識別子をカンマ区切りで列挙したリストで構成されます:

```
vtype=path
```

`vtype` はストレージで許可されているコンテンツタイプの一つであり、`path` はストレージのマウントポイントに対する相対パスです。

### 設定例 (/etc/pve/storage.cfg)

```
dir: backup
    path /mnt/backup content
    backup
    prune-backups keep-last=7 max-
    protected-backups 3
    content-dirs backup=custom/backup/dir
```

上記の設定は「backup」という名前のストレージプールを定義します。このプールでは、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ます。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 ファイル命名規則

このバックエンドはVMイメージに対して明確な命名規則を採用しています：

```
vm-<VMID>-<NAME>.<FORMAT>
```

```
<VMID>;
```

これは所有者VMを指定します。

**&lt;NAME&gt;**

これは空白を含まない任意の名前（ASCII）です。 バックエンドはデフォルトで `disk-[N]` を使用します。  
デフォルトとして使用します。ここで [N] は一意な名前にするための整数に置換されます。

**&lt;FORMAT&gt;**

イメージ形式を指定します (`raw|qcow2|vmdk`)。

VMテンプレート作成時、全てのVMイメージは以下のように読み取り専用であることを示す名称に変更され、クローン用ベースイメージとして使用可能になります：

`base-&lt;VMID&gt;-&lt;NAME&gt;.&lt;FORMAT&gt;`

**注記**

このようなベースイメージはクローンイメージの生成に使用されます。そのため、これらのファイルが読み取り専用であり、決して変更されないことが重要です。ストレージが対応している場合、バックエンドはアクセスモードを 0444 に変更し、不变フラグ (`chattr +i`) を設定します。

### 7.5.3 ストレージ機能

前述の通り、ほとんどのファイルシステムは標準ではスナップショットをサポートしていません。この問題を回避するため、本バックエンドは `qcow2` の内部スナップショット機能を利用可能です。

クローンについても同様です。このバックエンドは `qcow2` のベースイメージ機能を利用してクローンを作成します。

表7.3: バックエンドdirのストレージ機能

コンテンツタイプ	イメージフォーマット	共有	スナップショット	クローン
イメージ	<code>raw qcow2</code>	なし	<code>qcow2</code>	<code>qcow2</code>
<code>rootdir</code>	<code>vmdk subvol</code>			
<code>vztmpl iso</code>				
<code>backup</code>				
スニペット				

### 7.5.4 例

```
ストレージ local に 4GB のイメージを割り当てるには、次のコマンドを使用してください。# pvesm alloc
local 100 vm-100-disk10.raw 4G
'/var/lib/vz/images/100/vm-100-disk10.raw' のフォーマット、fmt=raw サイズ-->
=4294967296
'local:100/vm-100-disk10.raw' の作成に成功しました
```

**注記**

画像名は上記の命名規則に準拠する必要があります。

実際のファイルシステムパスは以下のように表示されます:

```
# pvesm path local:100/vm-100-disk10.raw  
/var/lib/vz/images/100/vm-100-disk10.raw
```

イメージは以下で削除できます:

```
# 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基盤としているため、ほとんどのプロパティを共有します。ディレクトリ構成とファイル命名規則は同一です。

主な利点は、NFSサーバーのプロパティを直接設定できるため、バックエンドが共有を自動的にマウントできる点です。

/etc/fstab を変更する必要はありません。また、サーバーがオンラインであるかどうかのテストが可能で、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します。

### 7.6.1 設定

このバックエンドは、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ます。さらに、NFSサーバーを設定するために以下のプロパティが使用されます:

#### server

サーバーのIPアドレスまたは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ことが望ましいです。ただし、非常に信頼性の高いDNSサーバーを使用している場合、またはサーバーをローカルの/etc/hostsファイルに記述する場合を除きます。

#### export

NFSエクスポートパス (pvesm nfsscanで表示されるもの)。

NFS マウントオプションも設定できます:

#### path

ローカルマウントポイント (デフォルトは /mnt/pve/<STORAGE\_ID>/)。

#### content-dirs

デフォルトのディレクトリレイアウトを上書きします。オプションです。

#### options

NFS マウントオプション (man nfs を参照)。

**設定例 (/etc/pve/storage.cfg)**

```
nfs: iso-templates
    path /mnt/pve/iso-templates server
    10.0.0.10
    export /space/iso-templates options
    vers=3,soft content iso,vztmp
```

**ヒント**

NFSリクエストがタイムアウトした後、デフォルトではNFSリクエストが無限に再試行されます。これによりクライアント側で予期せぬハングが発生する可能性があります。読み取り専用コンテンツの場合、再試行回数を3回に制限するNFSソフトオプションの採用を検討する価値があります。

## 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ていませんが、バックエンドはqcow2機能を利用してスナップショットとクローンを実装しています。

表 7.4: バックエンド nfs のストレージ機能

コンテンツタイプ	イメージフォーマット	共有	スナップショット	クローン
イメージ	raw qcow2	はい	qcow2	qcow2
ルートディレクトリ	vmdk			
vztmp iso				
backup				
スニペット				

## 7.6.3 例

エクスポートされた NFS 共有の一覧は次のコマンドで取得できます:

```
# pvesm scan nfs <サーバー名>
```

## 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チェックやエクスポート済み共有の便利な選択など、当社のバックエンドの利点をすべて活用できます。

## 7.7.1 設定

このバックエンドは、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ます。さらに、以下のCIFS固有のプロパティが利用可能です：

### server

サーバーのIPアドレスまたはDNS名。必須。

### ヒント

DNSルックアップの遅延を避けるため、非常に信頼性の高いDNSサーバーを使用している場合や、サーバーをローカルの/etc/hostsファイルに記述している場合を除き、DNS名ではなくIPアドレスを使用することをお勧めします。

### share

使用する CIFS 共有（利用可能な共有は pvesm scan cifs <アドレス>；または Web UI で取得可能）。必須。

### username

CIFSストレージのユーザー名。オプション、デフォルトは「guest」。

### password

ユーザーパスワード。オプション。rootのみが読み取り可能なファイル（/etc/pve/priv/storage/

### domain

このストレージのユーザードメイン（ワークグループ）を設定します。オプション。

### smbversion

SMBプロトコルバージョン。オプション、デフォルトは3。セキュリティ上の問題によりSMB1はサポートされていません。

### path

ローカルのマウントポイント。オプション、デフォルトは /mnt/pve/<STORAGE\_ID>/。

### content-dirs

デフォルトのディレクトリレイアウトの上書き設定。オプション。

### options

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、ここで設定すべきではありません。Proxmox VE は常に soft オプションを設定します。設定に応じて、以下のオプションは自動的に設定されます: username、credentials、guest、domain、vers。

### subdir

マウントする共有のサブディレクトリ。オプションで、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: backup
    path /mnt/pve/backupserver
    10.0.0.11share VMDatacontent
    backup
    options noserverino,echo_interval=30 username 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ません。ただし、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 バッキングファイルを使用できます。

表 7.5: バックエンド cifs のストレージ機能

コンテンツタイプ	イメージフォーマット	共有	スナップショット	クローン
イメージ rootdir vztmp1 iso backup スニペット	raw qcow2 vmdk	はい	qcow2	qcow2

### 7.7.3 例

エクスポートされたCIFS共有のリストを取得するには、次のコマンドを実行します:

```
# pvesm scan cifs <サーバー名> [--username <ユーザー名>] [--password]
```

次に、これらの共有のいずれかを Proxmox VE クラスター全体のストレージとして追加するには、次のコマンドを実行します:

```
# pvesm add cifs <storagename> --server <server> --share <share> [--<--> username <username>] [--password]
```

## 7.8 Proxmox バックアップサーバー

ストレージプールタイプ: pbs

このバックエンドにより、[Proxmox Backup Server](#) を他のストレージと同様に Proxmox VE に直接統合できます。Proxmox Backup ストレージは、Proxmox VE API、CLI、または Web インターフェースを通じて直接追加できます。

## 7.8.1 設定

このバックエンドは、常に設定されてい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ます。さらに、Proxmox Backup Server 向けの以下の特別なプロパティが利用可能です:

**server**

サーバーのIPまたはDNS名。必須。

**port**

デフォルトのポート（8007）の代わりにこのポートを使用します。オプション。

**username**

Proxmox Backup Serverストレージのユーザー名。必須。

**ヒント**

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: root@pam または archiver@pbs。

**password**

ユーザーパスワード。値は/etc/pve/priv/storage/<STORAGE-ID>配下のファイルに保存され、ルートユーザーのみがアクセス可能です。  
必須。

に保存され、rootユーザーのみがアクセス可能です。必須。

**datastore**

使用する Proxmox Backup Server データストアの ID。必須。

**fingerprint**

Proxmox Backup Server API TLS証明書のフィンガープリント。サーバーダッシュボードまたはproxmox-backup-manager cert infoコマンドで取得可能。自己署名証明書、またはホストがサーバーのCAを信頼しない証明書の場合に必須。

**encryption-key**

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鍵。現在、パスワード保護されていないもの（鍵導出関数（kdf）なし）のみがサポートされています。  
./etc/pve/priv/storage/<STORAGE-ID>.master.pem 配下のファイルに保存され、rootユーザーのみがアクセス可能です。マジック値 autogen を使用して自動的に新しい鍵を生成します。

proxmox-backup-client key create --kdf none <path> で自動生成します。オプションです。

**master-pubkey**

バックアップタスクの一環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。  
/etc/pve/priv/storage/<STORAGE-ID>.master.pem 配下のファイルに保存され、rootユーザーのみがアクセス可能です。  
暗号化されたバックアップ暗号化キーのコピーは各バックアップに追加され、復旧目的でProxmox Backup Serverインスタンス上に保存されます。オプション、encryption-keyが必要です。

**設定例 (/etc/pve/storage.cfg)**

```

pbs: backup
    datastore main
    server enya.proxmox.com content
    backup
    fingerprint 09:54:ef:..snip..: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
    encryption-key a9:ee:c8:02:13:..snip..:2d:53:2c:98 master-pubkey 1

```

**7.8.2 ストレージ機能**

Proxmox Backup Server はバックアップのみをサポートしており、ブロックレベルまたはファイルレベルベースで実行できます。Proxmox VE は仮想マシンにはブロックレベル、コンテナには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

コンテンツタイプ	画像形式	共有	スナップショット	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**

オプションで、GCMモードのAES-256によるクライアント側暗号化を設定できます。暗号化は、ウェブインターフェース経由、またはCLIの`encryption-key`オプション（上記参照）で設定可能です。鍵は`/etc/pve/priv/storage/<STORAGE-ID>.enc`ファイルに保存され、rootユーザーのみがアクセス可能です。

**警告**

 鍵がないとバックアップにアクセスできなくなります。そのため、鍵は整理して保管し、バックアップ対象の内容とは別の場所に保管してください。例えば、システム全体のバックアップをそのシステム上の鍵で行った場合、何らかの理由でシステムにアクセスできなくなり復元が必要になっても、破損したシステムと共に暗号化鍵も失われるため復元は不可能です。

迅速な災害復旧のため、鍵は安全かつ容易にアクセス可能な場所に保管することを推奨します。このため、パスワードマネージャーへの保存が最適です。これにより即時復旧が可能です。バックアップとして、鍵をUSBフラッシュドライブに保存し、安全な場所に保管してください。これによりシステムから切り離されつつ、緊急時にも容易に復旧できます。最悪の事態に備え、鍵の紙媒体コピーを安全な場所に保管することも検討すべきです。paperkeyサブコマンドを使用すれば、鍵のQRコード化バージョンを作成できます。以下のコマンドはpaperkeyコマンドの出力をテキストファイルに送信し、印刷を容易にします。

```
# proxmox-backup-client key paperkey /etc/pve/priv/storage/<storage-id>.enc <->
--output-format text> qrkey.txt
```

さらに、キー復元目的で単一のRSAマスターキーペアを使用することも可能です：暗号化バックアップを行う全クライアントが単一の公開マスターキーを使用するよう設定すると、以降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。対応する秘密マスターキーがあれば、クライアントシステムが利用できなくなった場合でもAESキーを復元しバックアップを復号できます。

#### 警告

マスターキーペアの保管ルールは通常の暗号化キーと同様です。秘密鍵のコピーがなければ復元は不可能です！paperkeyコマンドは、物理的な安全な場所に保管するためのマスター秘密鍵の紙媒体コピー生成をサポートします。

暗号化はクライアント側で管理されるため、異なる鍵で暗号化されていても、サーバー上の同一データストアを暗号化バックアップと非暗号化バックアップの両方に使用できます。ただし、異なる鍵を使用したバックアップ間の重複排除は不可能なため、別々のデータストアを作成する方が望ましい場合が多いです。

#### 注意

暗号化による利点がない場合（例：信頼できるネットワーク内でサーバーをローカルに実行している場合）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は常に容易です。

### 7.8.4 例: CLI 経由でのストレージ追加

利用可能なProxmox Backup Serverデータストアのリストは以下で取得できます:

```
# pvesm scan pbs <server-name> <username> [<password>] [<fingerprint>]
```

次に、これらのデータストアのいずれかを Proxmox VE クラスター全体のストレージとして追加するには、次のコマンドを実行します:

```
# pvesm add pbs <id> --server <server> --datastore <datastore> --username <username> --fingerprint 00:B4:... --password
```

### 7.9 ローカル ZFS プール バックエンド

Storage pool type: zfspool

このバックエンドでは、ローカル ZFS プール（またはそのようなプール内の ZFS ファイルシステム）にアクセスできます。

## 7.9.1 設定

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 `content`、`nodes`、`disable`、および以下の ZFS 固有のプロパティをサポートしています。

### pool

ZFS プール/ファイルシステムを選択します。すべての割り当てはそのプール内で行われます。

### blocksize

ZFS のブロックサイズパラメータを設定します。

### スパース

ZFS のシンプロビジョニングを使用します。スパースボリュームとは、予約領域がボリュームサイズと等しくないボリュームです。

### 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` が認識す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: vmdatas
    pool tank/vmdatas content
    rootdir,images sparse
```

## 7.9.2 ファイル命名規則

バックエンドは VM イメージに対して以下の命名規則を使用します:

```
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)
```

### `<VMID>`

これは所有者 VM を指定します。

### `<NAME>`

これは空白を含まない任意の名前 (ASCII) です。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] は一意な名前とするため整数で置換されます。

### 7.9.3 ストレージ機能

ZFSはスナップショットとクローンに関して最も先進的なストレージタイプです。バックエンドはVMイメージ（raw形式）とコンテナデータ（subvol形式）の両方にZFSデータセットを使用します。ZFSプロパティは親データセットから継承されるため、親データセットでデフォルトを設定するだけで済みます。

表 7.7: バックエンド zfs のストレージ機能

コンテンツタイプ	イメージフォーマット	共有	スナップショット	クローン
images	rawサブボリューム	いいえ	はい	はい
rootdir				

### 7.9.4 例

VMイメージを保存するための追加のZFSファイルシステムを作成することを推奨します:

```
# zfs create tank/vmdata
```

新しく割り当てたファイルシステムで圧縮を有効にするには:

```
# zfs set compression=on tank/vmdata
```

利用可能なZFSファイルシステムのリストを取得するには:

```
# pvesm scan zfs
```

## 7.10 LVM バックエンド

ストレージプールタイプ: lvm

LVMは、ハードディスクやパーティションの上に構築される軽量なソフトウェア層です。利用可能なディスク領域を小さな論理ボリュームに分割するために使用できます。

別のユースケースとして、大規模なiSCSI LUN（論理ユニット番号）やファイバーチャネル経由で接続されたSAN（ストレージエリアネットワーク）の上にLVMを配置する方法があります。これにより、iSCSI LUN上のスペースを容易に管理できます。これは、iSCSI仕様がスペース割り当てのための管理インターフェースを定義していないため、通常は不可能なことです。

### 7.10.1 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に加え、以下のLVM固有のプロパティをサポートします：

#### vgname

LVMボリュームグループ名。既存のボリュームグループを指す必要があります。

**base**

ベースボリューム。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ボリュームです。主に、LVMボリュームグループがリモートiSCSIサーバー上に存在する場合に有用です。

**saferemove**

Web UIでは「削除済みボリュームのワイプ」と呼ばれます。LV削除時にデータをゼロクリアします。ボリューム削除時に、すべてのデータが確実に消去され、後で作成される他のLV（たまたま同じ物理エクステントが割り当てられる場合）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となる場合があります。

**saferemove\_throughput**

ワイプのスループット (cstream -t パラメータ値)。

**snapshot-as-volume-chain**

このフラグを設定すると、LVM上の仮想マシンに対して、ボリュームをバックするチェーンを持つスナップショットサポートが有効になります。この設定では、スナップショットを取得すると、現在の状態がスナップショット名の下に永続化され、そのスナップショットをバックする新しいボリュームが開始されます。

スナップショットに基づくボリュームは、その親スナップショットボリュームをバックингボリュームとして参照し、そのバックингボリュームに対する差分のみを記録します。スナップショットボリュームは現在、厚いプロビジョニングのLVM論理ボリュームですが、基盤となるブロックストレージは薄いプロビジョニングを提供する場合があります。

この設計により、ネイティブLVMスナップショットが抱える問題（大幅な入出力（I/O）ペナルティや、事前割り当て領域不足時の予期せぬ危険な動作など）を回避します。

スナップショットをボリュームチェーンとして扱うことで、ブロックストレージをサポートするあらゆるストレージシステム（iSCSIやファイバーチャネル接続のSANを含む）において、ベンダーに依存しないスナップショットサポートを実現します。

この機能はqcow2に依存していますが、qcow2のスナップショット機能ではなく、バックアップチェーン内で複数のボリュームを階層化する機能のみを利用している点に注意してください。スナップショット機能はPVEストレージシステムによって管理されます。

このフラグの有効化または無効化は、新規作成される仮想ディスクボリュームにのみ影響します。

**設定例 (/etc/pve/storage.cfg)**

```
lvm: myspace
    vgname myspace content
    rootdir,images
```

## 7.10.2 ファイル命名規則

バックエンドは基本的にZFS プールバックエンドと同じ命名規則を使用します。

```
vm-<VMID>-<NAME> // 通常の VM イメージ
```

### 7.10.3 ストレージ機能

LVMは典型的なブロックストレージシステムです。残念ながら、通常のスナップショットは非効率的です。スナップショットがアクティブな間、ボリュームグループ全体の書き込み操作すべてに干渉するため、I/O性能が大幅に低下するからです。これがLVMがリンクドクローンをサポートしない理由であり、Proxmox VEがスナップショットをボリュームチェーンとしてサポートするようになった理由です。この機能はストレージプラグインを通じてスナップショットボリュームを管理し、qcow2を使用して個別のボリュームをバックингチェーンとして階層化します。これによりゲストに公開される単一のディスク状態が作成されます。

LVMの利点は共有ストレージ（例：iSCSI LUN）との併用が可能な点です。構成でストレージが共有とマークされている場合、バックエンドは適切なクラスタ全体のロックを実装します。

#### ヒント

非共有のローカルストレージには LVM-thin バックエンドを使用できます。スナップショットとリンクドクローンをサポートします。

表 7.8: バックエンド lvm のストレージ機能

コンテンツタイプ	イメージ形式	共有	スナップショット	完全クローン
リンク クローン	イメージ rootdir	raw、qcow2	可能	はい <sup>1</sup>

<sup>1</sup>: Proxmox VE 9以降、VM向けにボリュームチェーンとしてスナップショットが利用可能になりました。詳細は[LVM設定セクション](#)を参照してください。

### 7.10.4 例

利用可能なLVMボリュームグループのリストは以下で取得できます:

```
# pvesm scan lvm
```

## 7.11 LVMシンバックエンド

ストレージプールタイプ: lvmthin

LVMは通常、ボリューム作成時にブロックを割り当てます。これに対しLVMシンプールは、ブロックが書き込まれる際に割り当てを行います。この動作はシンプルビジョニングと呼ばれ、物理的に利用可能な領域よりもはるかに大きなボリュームを作成できるためです。

通常のLVMコマンドラインツールを使用して、LVMシンプールの管理や作成が可能です（詳細はman lvmthinを参照）。既にpveというLVMボリュームグループが存在すると仮定した場合、以下のコマンドでdataという名前の新しいLVMシンプール（サイズ100G）を作成します：

```
lvcreate -L 100G -n data pvelvconvert --type thin-
pool pve/data
```

## 7.11.1 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、および以下のLVM固有のプロパティをサポートします：

### vgname

LVMボリュームグループ名。既存のボリュームグループを指す必要があります。

### thinpool

LVMシンプールの名前。

#### 設定例 (/etc/pve/storage.cfg)

```
lvmthin: local-lvm
        thinpool data vgname
        pve
        content rootdir,images
```

## 7.11.2 ファイル命名規則

バックエンドは基本的にZFS プールバックエンドと同じ命名規則を使用します。

```
vm-<VMID>-<NAME> // 通常の VM イメージ
```

## 7.11.3 ストレージ機能

LVM thin はブロックストレージですが、スナップショットとクローンを効率的に完全にサポートします。新しいボリュームは自動的にゼロで初期化されます。

LVM thin プールは複数ノード間で共有できないため、ローカルストレージとしてのみ使用可能である点に留意が必要です。

表 7.9: バックエンド lvmthin のストレージ機能

コンテンツタイプ	イメージフォーマット	共有	スナップショット	クローン
images	raw	いいえ	はい	はい
rootdir				

## 7.11.4 例

ボリュームグループ pve で利用可能な LVM スリムプールのリストを取得するには、次のコマンドを実行します:

```
# pvesm scan lvmthin pve
```

## 7.12 Open-iSCSI イニシエーター

ストレージプールタイプ: `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す。ほぼ全てのストレージベンダーがiSCSIをサポートしています。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利用可能です。

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。これは標準的なDebianパッケージですが、リソース節約のためデフォルトではインストールされません。

```
# apt-get install open-iscsi
```

低レベルなiSCSI管理タスクは`iscsiadm`ツールを使用して実行できます。

### 7.12.1 設定

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`content`、`nodes`、`disable`に加え、以下のiSCSI固有のプロパティをサポートします：

#### ポータル

iSCSI ポータル (IP または DNS 名、オプションでポート)。

#### ターゲット

iSCSIターゲット。

#### 設定例 (`/etc/pve/storage.cfg`)

```
iscsi: mynas
      portal 10.10.10.1
      ターゲット iqn.2006-01.openfiler.com:tsn.dcb5aaaddd コンテンツ none
```

#### ヒント

iSCSI上でLVMを使用したい場合は、`content none`を設定するのが理にかなっています。そうすることで、iSCSI LUNを直接使用してVMを作成することは不可能になります。

### 7.12.2 ファイル命名規則

iSCSIプロトコルはデータの割り当てや削除を行うインターフェースを定義していません。代わりに、それはターゲット側で実行される必要があり、ベンダー固有です。ターゲットは単にそれらを番号付きLUNとしてエクスポートします。したがって、Proxmox VEのiSCSIボリューム名は、Linuxカーネルが認識するLUNに関する情報をエンコードしたものです。

### 7.12.3 ストレージ機能

iSCSIはブロックレベルのストレージであり、管理インターフェースを提供しません。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上にLVMを設定するのが最善です。その後、LVMプラグインを使用してそのiSCSI LUN上のストレージを管理できます。

表 7.10: バックエンド `iscsi` のストレージ機能

コンテンツタイプ	イメージフォーマット	共有	スナップショット	クローン
イメージなし	raw	はい	いいえ	いいえ

### 7.12.4 例

リモートiSCSIポータルをスキャンし、可能なターゲットのリストを取得するには以下を実行します:

```
pvesm scan iscsi <HOST[:PORT]>;
```

## 7.13 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: `iscsidirect`

このバックエンドは基本的にOpen-iSCSIバックエンドと同様の機能を提供しますが、実装にはユーザーレベルライブラリを使用します。このバックエンドを使用するにはlibiscsi-binパッケージのインストールが必要です。

カーネルドライバが一切関与しないため、パフォーマンス最適化と見なせます。ただし、このiSCSI LUN上にLVMを使用できないという欠点があります。したがって、すべての領域割り当てはストレージサーバー側で管理する必要があります。

### 7.13.1 設定

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: faststore portal
    10.10.10.1
    target iqn.2006-01.openfiler.com:tsn.dcb5aaadd
```

### 7.13.2 ストレージ機能

#### 注記

このバックエンドは仮想マシン専用です。コンテナはこのドライバを使用できません。

表 7.11: バックエンド `iscsidirect` のストレージ機能

コンテンツタイプ	イメージ形式	共有	スナップショット	クローン
イメージ	raw	はい	いいえ	いいえ

## 7.14 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ため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現し、以下の利点があります:

- シンプロビジョンング
- リサイズ可能なボリューム
- 分散化および冗長化（複数のOSDにストライピング）
- 完全なスナップショットおよびクローン機能
- 自己修復機能
- 単一障害点なし
- エクサバイトレベルまで拡張可能
- カーネル空間およびユーザー空間実装が利用可能

### 注記

小規模な導入環境では、CephサービスをProxmox VEノード上で直接実行することも可能です。最近のハードウェアは十分なCPU能力とRAMを備えているため、ストレージサービスと仮想マシンを同一ノード上で稼働させることができます。

### 7.14.1 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 `nodes`、`disable`、`content`、および以下の `rbd`固有のプロパティをサポートします：

#### monhost

監視デーモン IP のリスト。オプション。Ceph が Proxmox VE クラスター上で動作していない場合にのみ必要。

#### pool

Ceph プール名。

**username**

RBD ユーザー ID。オプション。Proxmox VE クラスター上で Ceph が実行されていない場合にのみ必要です。ユーザー ID のみを使用する必要があ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。

**krbd**

krbd カーネルモジュールを介した Rados ブロックデバイスへのアクセスを強制します。オプションです。

**注記**

コンテナはオプション値に関わらず krbd を使用します。

**外部 Ceph クラスターの設定例 (/etc/pve/storage.cfg)**

```
rbd: ceph-external
    monhost 10.1.1.20 10.1.1.21 10.1.1.22
    pool ceph-external content
    images username admin
```

**ヒント**

rbd ユーティリティを使用して、低レベルの管理タスクを実行できます。

## 7.14.2 認証

**注**

Ceph が Proxmox VE クラスターにローカルで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の処理が自動的に行われます。

デフォルトで有効になっている ceph 認証を使用する場合は、外部 Ceph クラスターのキーリングを提供する必要があります。

CLI 経由でストレージを設定するには、まずキーリングを含むファイルを利用可能にする必要があります。一つの方法は、外部 Ceph クラスターからファイルを直接 Proxmox VE ノードの一つにコピーすることです。以下の例では、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 Ceph サーバ>:/etc/ceph/ceph.client.admin.keyring /root/rbd.→
    キーリング
```

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、コピーしたキーリングファイルへのパスを指定する必要があります。例：

```
# pvesm add rbd <name> --monhost "10.1.1.20 10.1.1.21 10.1.1.22" --content →
    images --keyring /root/rbd.keyring
```

GUI 経由で外部 RBD ストレージを設定する際、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。

キーリングは以下に保存されます

```
# /etc/pve/priv/ceph/<STORAGE_ID>.keyring
```

**ヒント**

外部クラスターに接続する際は、必要な権限のみを含むキーリングの作成が推奨されます。Cephユーザー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>

<sup>a</sup> Ceph User Management

### 7.14.3 Cephクライアント設定（オプション）

外部 Ceph ストレージへの接続では、外部クラスタの構成データベースでクライアント固有のオプションを設定できない場合があります。ストレージの Ceph クライアント設定を変更するには、Ceph キーリングの横に `ceph.conf` を追加できます。

`ceph.conf` はストレージと同じ名前である必要があります。

```
# /etc/pve/priv/ceph/<STORAGE_ID>.conf
```

設定可能な項目については` RBD設定リファレンス<sup>1</sup> を参照してください。

**注意**

これらの設定は軽率に変更しないでください。Proxmox VE は `<STORAGE_ID>.conf` をストレージ設定と統合します。

### 7.14.4 ストレージ機能

`rbd` バックエンドはブロックレベルストレージであり、完全なスナップショットおよびクローン機能を実装しています。

表 7.12: バックエンド `rbd` のストレージ機能

コンテンツタイプ	イメージフォーマット	共有	スナップショット	クローン
images	raw	はい	はい	はい
rootdir				

## 7.15 Ceph ファイルシステム (CephFS)

ストレージプールタイプ: `cephfs`

CephFSは、Cephストレージクラスターを使用してデータを保存するPOSIX準拠のファイルシステムを実装します。CephFSはCephを基盤としているため、その特性の大部分を共有します。これには冗長性、スケーラビリティ、自己修復機能、および高可用性が含まれます。

<sup>1</sup> RBD構成リファレンス <https://docs.ceph.com/en/squid/rbd/rbd-config-ref/>

---

### ヒント

Proxmox VEはCeph環境を管理できるため、CephFSストレージの設定が容易になります。現代のハードウェアは高い処理能力とRAMを備えているため、ストレージサービスとVMを同一ノード上で実行しても、パフォーマンスに大きな影響を与えることはありません。

---

CephFSストレージプラグインを使用するには、標準のDebian Cephクライアントを当社の[Cephリポジトリ](#)を追加して置き換える必要があります。追加後、最新のパッケージを取得するために`apt update`を実行し、続いて`apt dist-upgrade`を実行してください。

---



### 警告

他のCephリポジトリが設定されていないことを必ず確認してください。設定されている場合、インストールが失敗するか、ノード上でパッケージバージョンが混在し、予期しない動作を引き起こす可能性があります。

---

## 7.15.1 設定

このバックエンドは、一般的なストレージプロパティ `nodes`、`disable`、`content` に加え、以下の CephFS 固有のプロパティをサポートします：

#### **fs-name**

Ceph FS の名前。

#### **monhost**

監視デーモンアドレスのリスト。オプション。CephがProxmox VEクラスター上で実行されていない場合にのみ必要。

#### **path**

ローカルマウントポイント。オプション、デフォルトは `/mnt/pve/<STORAGE_ID>/`。

#### **username**

CephユーザーID。オプション。CephがProxmox VEクラスター上で実行されていない場合にのみ必要。デフォルトは`admin`。

#### **subdir**

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`/`。

#### **fuse**

カーネルクライアントではなくFUSE経由でCephFSにアクセスします。オプション、デフォルトは`0`。

#### 外部 Ceph クラスターの設定例 (`/etc/pve/storage.cfg`)

```
cephfs: cephfs-external
    monhost 10.1.1.20 10.1.1.21 10.1.1.22
    path /mnt/pve/cephfs-external
    content backup
    username admin
    fs-name cephfs
```

**注記**

cephxが無効化されていない場合、クライアントの秘密鍵ファイルの設定を忘れないでください。

## 7.15.2 認証

**注意**

Ceph が Proxmox VE クラスタにローカルで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の処理が自動的に行われます。

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 Ceph クラスタのシークレットを提供する必要があります。

CLI 経由でストレージを設定するには、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。方法の一つとして、外部 Ceph クラスターからファイルを Proxmox VE ノードの 1 つに直接コピーする方法があります。以下の例では、実行するノードの /root ディレクトリにコピーします：

```
# scp <external 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、コピーした秘密ファイルへのパスを指定する必要があります。例：

```
# pvesm add cephfs <name> --monhost "10.1.1.20 10.1.1.21 10.1.1.22" --<-->
    content backup --keyring /root/cephfs.secret
```

GUI 経由で外部 RBD ストレージを設定する際、シークレット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。

このシークレットは鍵そのもののみを指し、rbd バックエンドのように [client.userid] セクションを含むものではありません。セクションを含むのとは異なります。

シークレットは

```
# /etc/pve/priv/ceph/<storage_id>.secret
```

シークレットは、以下のコマンドを実行することで Ceph クラスター（Ceph 管理者として）から取得できます。ここで userid は、クラスターへのアクセス用に設定済みのクライアント ID です。Ceph ユーザー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
```

## 7.15.3 ストレージ機能

cephfs バックエンドは、Ceph クラスター上に構築された POSIX 準拠のファイルシステムです。

表 7.13: バックエンド cephfs のストレージ機能

コンテンツタイプ	イメージフォーマット	共有	スナップショット	クローン
vztmpl iso バックアップ スニペット	なし	はい	はい <sup>[1]</sup>	いいえ

<sup>[1]</sup> 既知のバグは存在しませんが、十分なテストが行われていないため、スナップショットの安定性はまだ保証されていません。

## 7.16 BTRFS バックエンド

ストレージプールタイプ: 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似ているため、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してください。

主な違いは、このストレージタイプでは、スナップショットの取得とスナップショットを保持したオフラインストレージ移行をサポートするために、フォーマット済みディスクがサブボリュームに配置される点です。

### 注

BTRFS はファイルを開く際に `O_DIRECT` フラグを尊重します。つまり、VM はキャッシュモードを使用すべきではありません `none` を指定しないでください。そうしないと、チェックサムエラーが発生します。

### 7.16.1 設定

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されます。ディレクトリを BTRFS ストレージとして追加する場合、そのディレクトリ自体がマウントポイントではない場合は、`is_mountpoint` オプションを使用して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。

たとえば、BTRFS ファイルシステムが `/mnt/data2` にマウントされており、その `pve-storage/` サブディレクトリ（スナップショットであることが推奨されます）を `data2` というストレージプールとして追加する場合、次のエントリを使用できます。

```
btrfs: data2
  パス /mnt/data2/pve-storage内容
  rootdir,imagesマウントポイント /mnt/data2
```

### 7.16.2 スナップショット

サブボリュームまたは生ファイルのスナップショットを撮ると、スナップショットは、同じパスに @ とスナップショットの名前が続く読み取り専用のサブボリュームとして作成されます。

## 7.17 ZFS over iSCSI バックエンド

ストレージプールタイプ: zfs

このバックエンドは、ZFS プールをストレージとして持ち、SSH 経由で iSCSI ターゲット実装を持つリモートマシンにアクセスします。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として使用されます。

以下の iSCSI ターゲット実装がサポートされています:

- LIO (Linux)
- IET (Linux)
- ISTGT (FreeBSD)
- Comstar (Solaris)

### 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を必要とします。通常のストレージアプライアンス/SAN 上で ZFS プールを作成するためには使用することはできません。

### 7.17.1 設定

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 接続を受け入れるよう設定する必要があります。

Proxmox VE は、ZVOL の作成と iSCSI 経由でのエクスポートのためにターゲットに接続します。認証は、

/etc/pve/priv/zfs/<ターゲット IP>/\_id\_rsa

以下の手順で SSH キーを作成し、IP アドレス 192.0.2.1 のストレージマシンに配布します:

```
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します:

#### pool

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、オプションでポート指定)。

#### ターゲット

iSCSI ターゲット。

#### iscsiprovider

リモートマシンで使用される iSCSI ターゲット実装

**comstar\_tg**

comstar 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 ターゲットのターゲットポータルグループ

**nowritecache**

ターゲットでの書き込みキャッシングを無効化

**blocksize**

ZFS ブロックサイズパラメータを設定します。

**sparse**

ZFS シンプロビジョンングを使用します。スパースボリュームとは、予約サイズがボリュームサイズと等しくないボリュームです。

**設定例 (/etc/pve/storage.cfg)**

```
zfs: lio
    ブロックサイズ
    4k
    iscsiprovider LIOプール
    tank
    ポータル 192.0.2.111
    ターゲット iqn.2003-01.org.linux-iscsi.lio.x8664:snxxxxxxxxxxxxx コンテンツ images
    lio_tpg tpg1 スペース 1

zfs: solaris ブロックサイズ 4k
    target iqn.2010-08.org.illumos:02:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx: -->
        tank1 プ
    ール tank
    iscsiprovider comstar portal
    192.0.2.112 コンテンツ images

zfs: freebsd ブロックサイズ 4k
    ターゲット iqn.2007-09.jp.ne.peach.istgt:tank1 プール tank
    iscsiprovider istgt ポータル
    192.0.2.113 コンテンツ イメー
        ジ

zfs: iet
    ブロックサイズ 4k
```

```
ターゲット iqn.2001-04.com.example:tank1 プール tank
iscsiprovider iet portal
192.0.2.114 content images
```

## 7.17.2 ストレージ機能

ZFS over iSCSIプラグインは、スナップショット機能を備えた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における単一障害点とならないよう注意が必要です。

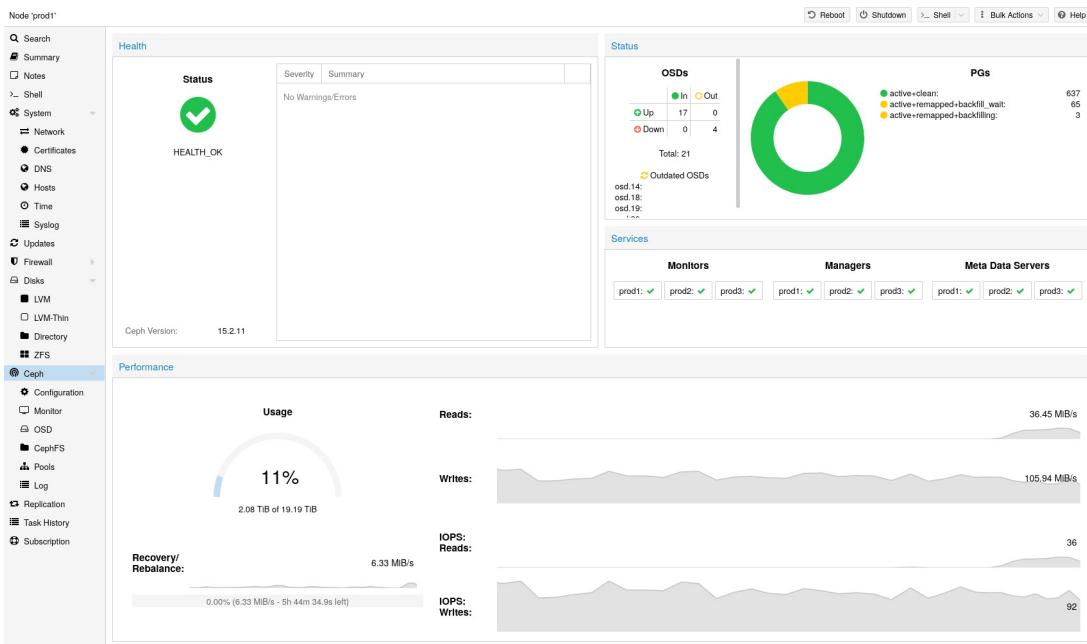
表 7.14: バックエンド iSCSI のストレージ機能

コンテンツタイプ	イメージ形式	共有	スナップショット	クローン
イメージ	raw	はい	はい	いいえ

## 第8章

# ハイパー・コンバージド Ceph クラスターの展開

## 8.1 はじめに



Proxmox VEは、コンピューティング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やコンテナの処理）と複製ストレージの両方に使用できます。従来のコンピューティングリソースとストレージリソースのサイロ化は、単一のハイパー・コンバージドアプライアンスに統合されます。ストレージネットワーク（SAN）やネットワーク接続ストレージ（NAS）を介した接続は不要になります。オープンソースのソフトウェア定義ストレージプラットフォームであるCephを統合することで、Proxmox VEはハイパー・バイザノード上で直接Cephストレージを実行・管理する機能を備えています。

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ため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。

PROXMOX VE上でCEPHを利用する主な利点は以下の通りです：

- CLIとGUIによる簡単な設定と管理
- シンプロビジョニング
- スナップショットサポート

- 自己修復機能
- 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
- 異なるパフォーマンスと冗長性特性を備えたプールの設定
- データは複製され、耐障害性を実現
- 汎用ハードウェア上で動作
- ハードウェアRAIDコントローラーが不要
- オープンソース

中小規模の展開では、Proxmox VEクラスターノード上に直接Cephサーバーをインストールし、RADOSブロックデバイス（RBD）またはCephFSを利用することができます（[Ceph RADOSブロックデバイス（RBD）を参照](#)）。最近のハードウェアはCPUパワーとRAMを豊富に備えているため、ストレージサービスと仮想ゲストを同一ノード上で実行することが可能です。

管理を簡素化するため、Proxmox VEはネイティブ統合を提供し、組み込みのWebインターフェースまたは

## 8.2 用語

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ます：

- Ceph Monitor (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オブジェクトストレージデーモン (ceph-osd、またはOSD)

### ヒント

Ceph<sup>a</sup>、そのアーキテクチャ<sup>b</sup>、および用語<sup>c</sup>をよく理解しておくことを強くお勧めします。

<sup>a</sup> Ceph イントロダクション <https://docs.ceph.com/en/squid/start/>

<sup>b</sup> Ceph アーキテクチャ <https://docs.ceph.com/en/squid/architecture/>

<sup>c</sup> Ceph用語集 <https://docs.ceph.com/en/squid/glossary>

## 8.3 健全なCephクラスターの推奨事項

ハイパーコンバージドProxmox+ Cephクラスターを構築するには、設定に少なくとも3台（できれば同一仕様）のサーバーを使用する必要があります。

[Ceph公式サイト](#)の推奨事項も参照してください。

### 注意

以下の推奨事項は、ハードウェア選択の概略的な指針として捉えてください。したがって、特定のニーズに合わせて調整することが依然として重要です。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ます：

- 高ベースクロック周波数とマルチコアの恩恵を受けた、高負荷なCPU使用。このカテゴリに属するものは以下の通り：
  - オブジェクトストレージデーモン (OSD) サービス
  - CephFSで使用されるメタデータサービス (MDS)
- 中程度のCPU使用量で、複数のCPUコアを必要としないもの。これらは以下の通りです：
  - Monitor (MON) サービス
  - Manager (MGR) サービス

大まかな目安として、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、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）を割り当てるべきです。

例えば、ノード上でCephモニター、Cephマネージャー、および6つのCeph OSDサービスを実行する場合、基本的に安定したパフォーマンスを目標とするなら、Ceph専用に8つのCPUコアを確保すべきです。

OSDのCPU使用率は主にディスク性能に依存することに注意してください。ディスクの可能なIOPS（1秒あたりの入出力操作数）が高いほど、OSDサービスがより多くのCPUを利用できます。NVMeのような最新のエンタープライズ SSD ディスクは、100,000 以上の高い IOPS 負荷を 1 ミリ秒未満のレイテンシで永続的に維持できるため、各 OSD は複数の CPU スレッドを使用できます。たとえば、非常に高性能なディスクでは、NVMe ベースの OSD あたり 4 から 6 の CPU スレッドが使用される可能性があります。

## メモリ

特にハイパーコンバージド環境では、メモリ消費量を慎重に計画し監視する必要があります。仮想マシンやコンテナの予測メモリ使用量に加え、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確保することも考慮しなければなりません。

経験則として、OSDはおよそ1TiBのデータに対して1GiBのメモリを使用します。通常時は使用量が少ない場合もありますが、リカバリ、リバランス、バックフィルなどの重要な操作時には最大使用量に達します。つまり、通常動作時に利用可能なメモリを上限まで使い切ることは避け、障害発生時に備えて余裕を持たせる必要があります。

OSDサービス自体も追加メモリを消費します。デーモンのCeph BlueStoreバックエンドはデフォルトで3~5 GiBのメモリを必要とします（調整可能）。

## ネットワーク

Cephトラフィック専用に、少なくとも10Gbps以上のネットワーク帯域幅を推奨します。10Gbps以上のスイッチが利用できない場合、3~5ノードクラスターではメッシュネットワーク構成<sup>1</sup>も選択肢となります。

### 重要

 トラフィック量（特に復旧時）は同一ネットワーク上の他サービスに影響を及ぼします。特に遅延に敏感なProxmox VEのcorosyncクラスタスタックが影響を受け、クラスタのクオーラム喪失を引き起こす可能性があります。Cephトラフィックを専用かつ物理的に分離されたネットワークに移行することで、corosyncだけでなく仮想ゲストが提供するネットワークサービスに対しても、このような干渉を回避できます。

必要な帯域幅を推定するには、ディスクの性能を考慮する必要があります。単一のHDDでは1Gbリンクを飽和させないかもしれません、ノードあたり複数のHDD OSD（Operating System Disk）があれば、10Gbpsリンクさえ飽和させる可能性があります。最新のNVMe接続SSDを使用する場合、単体で10Gbps以上の帯域幅を飽和せることもあります。このような高性能構成では最低25Gbpsを推奨し、基盤となるディスクの性能を最大限活用するには40Gbpsや100Gbps以上が必要となる場合もあります。

不明な場合は、高性能な設定において3つの（物理的に）分離されたネットワークの使用を推奨します：

- 1. Ceph（内部）クラスタトラフィック用の超高速帯域幅（25+ Gbps）ネットワーク。
- CephサーバーとCephクライアント間のストレージトラフィック（パブリック）用に、高帯域幅（10+ Gbps）ネットワークを1つ。必要に応じて、仮想ゲストトラフィックや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可能です。
- 遅延に敏感な Corosync クラスター通信専用の、中程度の帯域幅（1 Gbps）ネットワーク。

## ディスク

Cephクラスターの規模を計画する際には、復旧時間を考慮することが重要です。特に小規模クラスターでは復旧に時間がかかる可能性があります。復旧時間を短縮し、復旧中に次の障害が発生する可能性を最小限に抑えるため、小規模環境ではHDDではなくSSDの使用が推奨されます。

一般的に、SSDは回転ディスクよりも高いIOPSを提供します。この点を考慮すると、コストが高いという点を除けば、[クラスペースのプール分離](#)を実施する意味があるかもしれません。OSDの高速化には、ジャーナルやDB/WALデバイスとして高速ディスクを使用する方法もあります（[Ceph OSDの作成](#)を参照）。複数のOSDに高速ディスクを使用する場合、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。さもなければ、高速ディスクが関連する全OSDのボトルネックとなります。

ディスクの種類とは別に、Cephはノードごとに均等なサイズで均等に分散されたディスク数で最高のパフォーマンスを発揮します。例えば、各ノード内に500GBディスク4台を配置する方が、1TBディスク1台と250GBディスク3台を混在させる構成よりも優れています。

OSD数と単一OSD容量のバランスも重要です。容量を増やすとストレージ密度が向上しますが、一方で単一OSDの障害が発生した場合、Cephが一度に復旧すべきデータ量が増加します。

<sup>1</sup> Full Mesh Network for Ceph [https://pve.proxmox.com/wiki/Full\\_Mesh\\_Network\\_for\\_Ceph\\_Server](https://pve.proxmox.com/wiki/Full_Mesh_Network_for_Ceph_Server)

## RAIDの使用は避ける

Cephはデータオブジェクトの冗長性とディスク(OSD)への複数並列書き込みを独自に処理するため、RAIDコントローラの使用は通常、パフォーマンスや可用性を向上させません。むしろCephは、中間抽象化層なしでディスク全体を直接扱うよう設計されています。RAIDコントローラはCephのワークロード向けに設計されておらず、書き込み/キャッシュアルゴリズムがCephのアルゴリズムと干渉する可能性があるため、状況を複雑化させ、場合によってはパフォーマンスを低下させる恐れがあります。



### 警告

RAIDコントローラの使用は避けてください。代わりにホストバスアダプター（HBA）を使用してください。

## 8.4 Cephの初期インストールと設定

### 8.4.1 Webベースのウィザードを使用したインストール

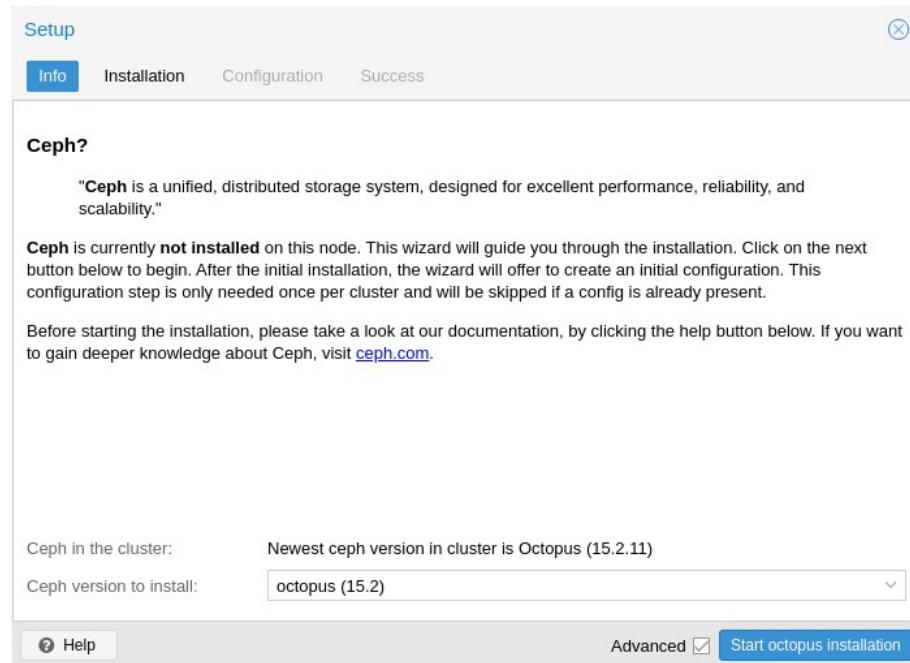
The screenshot shows the Proxmox VE web interface for managing a node named 'nina'. The left sidebar contains a navigation menu with various options like Search, Summary, Notes, Shell, System, Updates, Firewall, Disks, and Ceph. The 'Ceph' option is currently selected and highlighted in blue. The main content area is divided into several sections: 'Health' (Status: No Warnings/Errors), 'Ceph Version:' (status: N/A), 'Status' (OSDs: Total: 0, Up: 0, Down: 0), and 'Services' (Monitors, Managers, Meta Data Servers). A prominent callout box in the 'Status' section states: 'Ceph is not installed on this node. Would you like to install it now?' with a blue 'Install Ceph' button.

Proxmox VEでは、Ceph用の使いやすいインストールウィザードを利用できます。クラスタノードのいずれかをクリックし、メニューツリー内の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、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完了させる必要があります。

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。他のノードと同じバージョンを優先するか、これがCephをインストールする最初のノードの場合は最新版を選択してください。

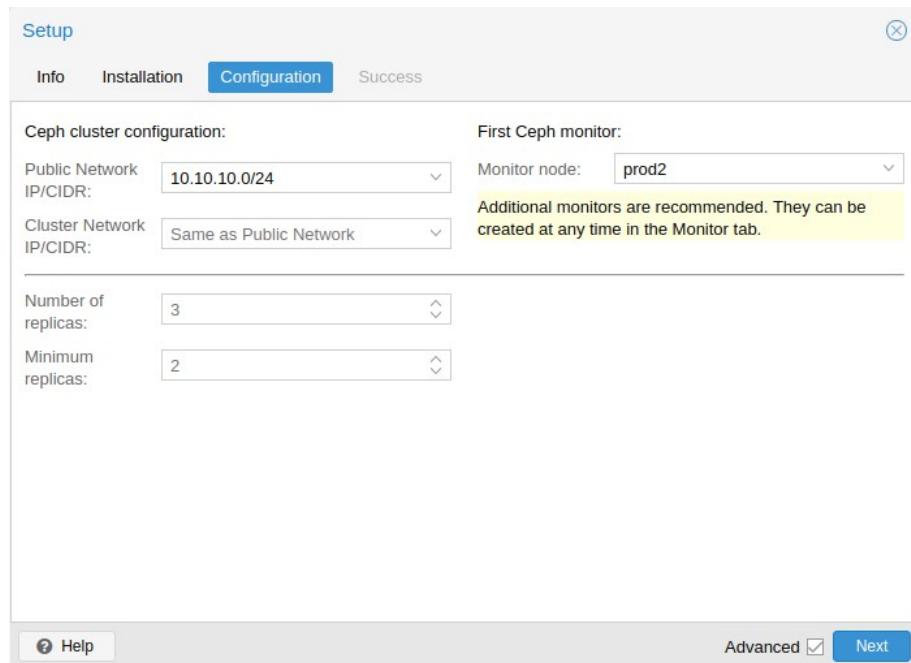
インストールを開始すると、ウィザードがProxmox VEのCephリポジトリから必要なパッケージをすべてダウンロードしてインストールします。



インストールステップが完了したら、設定を作成する必要があります。この設定はクラスタごとに一度だけ必要であり、Proxmox VEのクラスタ化された[設定ファイルシステム \(pmxcfs\)](#)を通じて、残りのクラスタメンバーすべてに自動的に配布されます。

設定ステップには以下の項目が含まれます：

- 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（例: Ceph RBD バックアップディスクを使用する仮想マシン、または CephFS マウント）および異なる 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 クラスター通信 (corosync) および仮想ゲストのフロントエンド（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 I/O トラフィックが他の低遅延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 レプリケーションとハートビート 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の使用が推奨されます。これにより、Ceph パブリックネットワークと仮想ゲストネットワークの負荷が軽減されると同時に、Ceph のパフォーマンスが大幅に向上します。  
Ceph クラスタネットワークは後から別の物理的に分離されたネットワークへ設定変更・移行可能です。



さらに 2 つのオプションがありますが、これらは高度な設定であるため、その内容を十分に理解している場合にのみ変更してください。

- **レプリカ数:** 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 が完了とマークされるために必要な最小レプリカ数を定義します。

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。この手順は必須です。

以上です。最終ステップとして成功ページが表示され、今後の手順に関する詳細な指示が確認できるはずです。これでシステムはCephの使用準備が整いました。開始するには、追加のモニター、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。

本章の残りの部分では、Proxmox VEベースのCeph環境を最大限に活用する方法について説明します。これには前述のヒントに加え、新規Cephクラスターに有用な追加機能であるCephFSなども含まれます。

## 8.4.2 CephパッケージのCLIインストール

ウェブ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、各ノードで以下のCLIコマンドを使用できます：

```
pveceph install
```

これにより、/etc/apt/sources.list.d/ceph.sources に apt パッケージリポジトリが設定され、必要なソフトウェアがインストールされます。

## 8.4.3 CLIによる初期 Ceph 設定

Proxmox VE Cephインストールウィザード（推奨）を使用するか、いずれか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、Ceph専用のネットワークを設定した初期構成が/etc/pve/ceph.confに作成されます。このファイルはpmxefs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。また、このコマンドは/etc/ceph/ceph.confにシンボリックリンクを作成し、そのファイルを指します。したがって、構成ファイルを指定する必要なく、Cephコマンドを単純に実行できます。

## 8.5 Ceph Monitor

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 Monitor (MON)<sup>2</sup>はクラスタマップのマスターコピーを管理します。高可用性を実現するには、少なくとも3台のモニターが必要です。インストールウィザードを使用した場合、1台のモニターは既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る限り、3台以上のモニターは必要ありません。これ以上のモニターが必要になるのは、非常に大規模なクラスタのみです。

### 8.5.1 モニターの作成

モニターを設置する各ノード（3台推奨）で、GUIのCeph→ モニタータブを使用するか、以下のコマンドを実行して作成します：

```
pveceph mon create
```

<sup>2</sup> Ceph Monitor <https://docs.ceph.com/en/squid/rados/configuration/mon-config-ref/>

## 8.5.2 モニターの削除

GUI経由でCephモニターを削除するには、まずツリービューでノードを選択し、**Ceph→ Monitor**パネルに移動します。MONを選択し、Destroyボタンをクリックします。パネルに移動します。MONを選択し、[Destroy]ボタンをクリックします。

CLI経由でCeph Monitor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。次に以下のコマンドを実行します：

```
pveceph mon destroy
```

### 注

クオーラムには最低3つのモニターが必要です。

## 8.6 Ceph Manager

Managerデーモンはモニターと並行して動作します。クラスターを監視するための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 Managerの作成

複数のマネージャーをインストールできますが、同時にアクティブなマネージャーは1つだけです。

```
pveceph mgr create
```

### 注

Ceph Managerは監視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ーをインストールしてください。

### 8.6.2 Manager の削除

GUI 経由で Ceph マネージャーを削除するには、まずツリービューでノードを選択し、**Ceph→ のモニター**に移動します。

パネル。マネージャーを選択し、[破棄]ボタンをクリックします。

CLI経由でCeph Monitorを削除するには、まずManagerが稼働しているノードに接続します。次に以下のコマンドを実行します：

```
pveceph mgr destroy
```

### 注記

マネージャーはハード依存関係ではありませんが、PG自動スケーリング、デバイス健全性監視、テレメトリなどの重要な機能を処理するため、Cephクラスターにとって不可欠です。

<sup>3</sup> Ceph Manager <https://docs.ceph.com/en/squid/mgr/>

## 8.7 Ceph OSD

Node 'prod2'									
<input type="button" value="Reload"/> <input type="button" value="Create: OSD"/> <input type="button" value="Manage Global Flags"/> <span style="float: right;">No OSD selected</span> <input type="button" value="Start"/> <input type="button" value="Stop"/> <input type="button" value="Restart"/> <input type="button" value="Out"/> <input type="button" value="In"/> <input type="button" value="More"/>									
Name	Class	OSD Type	Status	Version	weight	reweight	Used (%)	Total	Apply/Commit Latency (ms)
<b>default</b>									
<b>prod3</b>									
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
<b>prod2</b>									
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
<b>prod1</b>									
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 オブジェクトストレージデーモンは、ネットワーク経由で Ceph のオブジェクトを保存します。物理ディスクごとに 1 つの OSD を使用することをお勧めします。

### 8.7.1 OSDの作成

OSDは、Proxmox VEのWebインターフェースまたはpvecephを使用したCLIのいずれかで作成できます。例：

```
pveceph osd create /dev/sd[X]
```

#### ヒント

Cephクラスターは、少なくとも3ノードと12個のOSDを推奨します。OSDはノード間で均等に分散させてください。

ディスクが以前使用されていた場合（例：ZFSやOSDとして）、まずその使用痕跡をすべて消去する必要があります。パーティションテーブル、ブートセクタ、

```
ceph-volume lvm zap /dev/sd[X] --destroy
```

その他のOSDの残骸を削除するには、次のコマンドを使用できます：

**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します！

**Ceph Bluestore**

Ceph Kraken リリース以降、Bluestore<sup>4</sup> と呼ばれる新しい Ceph OSD ストレージタイプが導入されました。これは Ceph Luminous 以降、OSD 作成時のデフォルト設定です。

```
pveceph osd create /dev/sd[X]
```

**Block.db と block.wal**

OSD用に別個のDB/WALデバイスを使用したい場合、-db\_devおよび-wal\_devオプションで指定できます。WALは、別途指定しない限り DBと同じ場所に配置されます。

```
pveceph osd create /dev/sd[X] -db_dev /dev/sd[Y] -wal_dev /dev/sd[Z]
```

-db\_size および -wal\_size パラメータでそれぞれサイズを直接指定できます。指定しない場合、以下の値（順に）が使用されます：

- bluestore\_block\_{db,wal}\_size (Ceph設定より) ...
  - ...データベース、セクション *osd*
  - ...データベース、セクション *global*
  - ...ファイル、セクション *osd*
  - ...ファイル、セクション *global*
- OSDサイズの10% (DB)/1% (WAL)

**注記**

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（書き込み事前ログ）です。パフォーマンス向上のため、高速SSDまたはNVRAMの使用を推奨します。

**Ceph Filestore**

Ceph Luminous以前、FilestoreはCeph OSDのデフォルトストレージタイプとして使用されていました。Ceph Nautilus以降、Proxmox VE ではpvecephを用いたこのようなOSDの作成はサポートされなくなりました。Filestore OSDを作成したい場合は、ceph-volumeを直接使用してください。

```
ceph-volume lvm create --filestore --data /dev/sd[X] --journal /dev/sd[Y]
```

<sup>4</sup> Ceph Bluestore <https://ceph.com/community/new-luminous-bluestore/>

## 8.7.2 OSDの破棄

OSD またはそのディスクで問題が発生した場合は、まずトラブルシューティングを行って、[交換](#)が必要かどうかを判断してください。

OSD を破棄するには、`&lt;Node&gt; → Ceph → OSD` パネルに移動するか、OSD が存在するノードで上記の CLI コマンドを使用します。

1. クラスタがOSDの削除を処理するのに十分な空き容量を確保してください。Ceph の → OSD パネルで、破棄予定のOSDが稼働中で利用可能状態（AVAILが0以外）の場合、すべてのOSDの「使用率 (%)」がデフォルトのnearfull\_ratio (85%) を大幅に下回っていることを確認してください。これにより、今後のリバランスによるリスクを軽減できます。リバランスではOSDが満杯状態になり、Ceph プール上でI/Oがブロックされる可能性があります。

CLIで同様の情報を取得するには以下のコマンドを使用します:

```
ceph osd df tree
```

2. 破棄予定のOSDがまだアウト状態になっていない場合は、該当OSDを選択し「Out」をクリックしてください。これによりデータ分散から除外され、リバランスが開始されます。

以下のコマンドでも同様の処理が可能です：

```
ceph osd out &lt;id&gt;
```

3. 可能であれば、Cephがリバランスを完了するまで待機し、常に十分なレプリカを確保してください。OSDは空の状態になります。空になると、PG数が0と表示されます。
4. 「停止」をクリックしてください。停止が安全でない場合、警告が表示されますので「キャンセル」をクリックしてください。しばらくしてから再度お試しください。

OSDの停止が安全かどうかを確認し、停止するには以下のコマンドを使用できます：

```
ceph osd ok-to-stop &lt;id&gt; pveceph stop -service osd.&lt;id&gt;;
```

5. 最後に：

CephからOSDを削除し、ディスク上の全データを消去するには、まず「詳細」→「→」→「破棄」をクリックします。パーティションテーブルやその他の構造体をクリーンアップするオプションを有効にします。これにより、Proxmox VEでディスクを直ちに再利用可能になります。その後、「削除」をクリックします。

OSDを破壊するCLIコマンドは次の通りです:

```
pveceph osd destroy &lt;id&gt; [--cleanup]
```

## 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的なグループです。これは配置グループ (PG、pg\_num) として知られるオブジェクトの集合を保持します。pg\_num と呼ばれるオブジェクトの集合を保持します。

### 8.8.1 プールの作成と編集

プールは、コマンドラインまたは任意の Proxmox VE ホストの Web インターフェースから作成・編集できます。

Ceph → Pools

オプションが指定されていない場合、OSD の障害発生時にデータ損失が発生しないよう、デフォルトで 128 の PG、3 レプリカ、2 レプリカの min\_size を設定します。



#### 警告

min\_size を 1 に設定しないでください。min\_size が 1 のレプリケートプールでは、オブジェクトが 1 レプリカしか存在しない状態で I/O 操作を許可するため、データ損失、不完全な PG、またはオブジェクトが見つからない状態が発生する可能性があります。

PG-Autoscalerを有効にするか、環境に基づいてPG数を計算することを推奨します。計算式とPG計算ツール<sup>5</sup>はオンラインで確認できます。Ceph Nautilus以降では、設定後にPG数<sup>6</sup>を変更可能です。

<sup>5</sup> PG計算ツール <https://web.archive.org/web/20210301111112/http://ceph.com/pgcalc/>

<sup>6</sup> 配置グループ <https://docs.ceph.com/en/squid/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数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詳細パラメータを設定すると、PGオートスケーラーがより適切な判断を下すのに役立ちます。

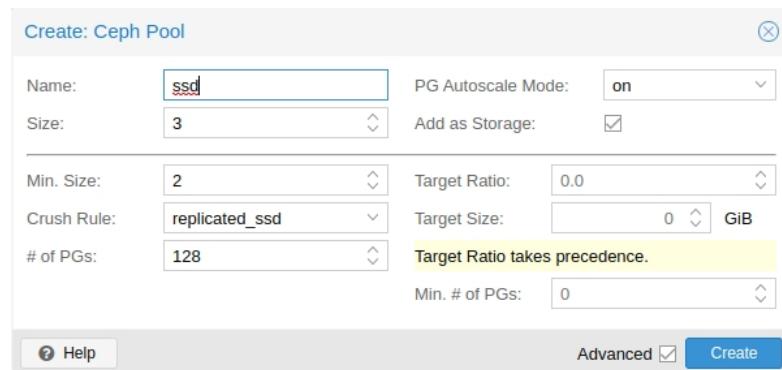
#### CLI経由でのプール作成例

```
pveceph pool create <pool-name> --add_storages
```

#### ヒント

プール用のストレージも自動的に定義したい場合は、Web インターフェースで「ストレージとして追加」チェックボックスをオンにしたままにするか、プール作成時にコマンドラインオプション `--add_storages` を使用してください。

#### プールオプション



以下のオプションは、プール作成時、および一部はプール編集時にも使用できます。

#### 名前

プールの名前。これは一意である必要があり、後から変更できません。

#### Size

オブジェクトごとのレプリカ数。Cephは常にこの数のオブジェクトコピーを保持しようとします。デフォルト: 3。

#### PG自動スケーリングモード

プールの自動PGスケーリングモード<sup>7</sup>。警告モードに設定すると、プールのPG数が最適でない場合に警告メッセージを出力します。デフォルト: 警告。

#### ストレージとして追加

新しいプールを使用してVMまたはコンテナのストレージを設定します。デフォルト: true (作成時のみ表示)。

#### 詳細オプション

#### 最小サイズ

オブジェクトごとの最小レプリカ数。PGのレプリカ数がこの数未満の場合、Cephは当該プールへの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/squid/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をマッピングする際に使用するルール。これらのルールは、データがクラスタ内にどのように配置されるかを定義します。デバイスベースのルールに関する情報は、[Ceph CRUSH およびデバイスクラス](#)を参照してください。

### PG数

プールが開始時に持つべき配置グループの数<sup>6</sup>。デフォルト：128。

### 目標比率

プールに期待されるデータの比率。PGオートスケーラーは他の比率セットとの相対的な比率を使用します。両方が設定されている場合、ターゲットサイズよりも優先されます。

### 目標サイズ

プールに存在すると予想されるデータの推定量。PGオートスケーラーはこのサイズを基に最適なPG数を推定します。

### 最小PG数

配置グループの最小数。この設定は、そのプールにおける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しきい値を下回るPGをマージしません。

Cephプール処理の詳細については、[Ceph プール操作<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「前方誤り訂正」コードの一種であり、一定量のデータ損失から回復することを可能にします。消去符号化プールは複製プールと比較してより多くの使用可能領域を提供できますが、その代償としてパフォーマンスが低下します。

比較のため：従来の複製プールではデータの複数レプリカが保存されます（サイズ）。一方、消去符号化プールではデータが $k$ 個のデータチャックに分割され、追加で $m$ 個の符号化（チェック）チャックが生成されます。データチャックが欠損した場合、これらの符号化チャックを用いてデータを再構築できます。

符号化チャックの数を $m$ とすると、データ損失なく失われるOSDの最大数が定義されます。格納されるオブジェクトの総数は $k + m$ となります。

### ECプールの作成

消去符号化（EC）プールはpveceph CLIツールで作成できます。ECプールの計画では、複製プールとは動作が異なる点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、ECプールのmin\_sizeは $k$ となります。 $m > 1$ の場合、min\_sizeは $k + 1$ となります。Cephのドキュメントでは保守的なmin\_sizeとして $k + 2$ <sup>9</sup>を推奨しています。

利用可能なOSDがmin\_size未満の場合、十分なOSDが再び利用可能になるまで、プールへのあらゆるI/Oはブロックされます。

<sup>8</sup> Ceph pool operation <https://docs.ceph.com/en/squid/rados/operations/pools/>

<sup>9</sup> Ceph Erasure Coded Pool Recovery <https://docs.ceph.com/en/squid/rados/operations/erasure-code/#erasure-coded-pool-recovery>

---

### 注記

消去符号化プールを計画する際は、利用可能なOSDの必要数を定義する`min_size`に注意してください。これを下回るとIOがブロックされます。

---

例えば、 $k = 2$ 、 $m = 1$  の EC プールは、**サイズ** = 3、**最小サイズ** = 2 となり、1つの OSD が障害を起こしても稼働を継続します。 $k = 2$ 、 $m = 2$  で構成されたプールは、**サイズ** = 4、**最小サイズ** = 3 となり、1つの OSD が失われても稼働を継続します。

新しいECプールを作成するには、次のコマンドを実行します：

```
pveceph pool create <pool-name> --erasure-coding k=2,m=1
```

オプションパラメータとして`failure-domain`と`device-class`が利用可能です。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規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のOMAPやその他のメタデータを格納するために必要なレプリケートされたプールが作成されます。最終的には、`<プール名> -data プールと <プール名> -metadata プール`が存在することになります。デフォルトでは、対応するストレージ構成も作成されます。この動作を無効にするには、`--add_storages 0` パラメータを指定してください。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。これにより初めて、EC プールがデータオブジェクトの保存に使用されます。例：

---

### 注記

オプションの`--size`、`--min_size`、`--crush_rule` パラメータは複製メタデータプールには適用されますが、エラー訂正符号化データプールには適用されません。データプールの`min_size` を変更する必要がある場合は、後から変更できます。サイズと`crush_rule` パラメータは、エラー訂正符号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ツールを使用して直接作成できます<sup>10</sup>。プロファイルパラメータで利用するプロファイルを指定します。例：

```
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

既存のECプールをProxmox VEのストレージとして追加できます。

RBD プールを追加する方法と同じですが、追加の`data-pool` オプションが必要です。

```
pvesm add rbd <storage-name> --pool <replicated-pool> --data-pool <ec-pool>;
```

---

### ヒント

ローカルの Proxmox VE クラスターによって管理されていない外部 Ceph クラスターについては、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/squid/rados/operations/erasure-code/#erasure-code-profiles>

### 8.8.3 プールの破棄

→ GUIでプールを削除するには、ツリービューでノードを選択し、Ceph管理パネルの「Pools」に移動します。削除するプールを選択し、「Destroy」ボタンをクリックします。プールの削除を確認するには、プール名を入力する必要があります。

プールを削除するには、以下のコマンドを実行します。関連するストレージも削除するには、*-remove\_storages* オプションを指定します。

```
pveceph pool destroy <name>;
```

#### 注記

プールの削除はバックグラウンドで実行され、時間がかかる場合があります。この処理中はクラスター内のデータ使用量が減少していくのが確認できます。

### 8.8.4 PG Autoscaler

PGオートスケーラーは、クラスターが各プールに保存されている（予想される）データ量を考慮し、適切なpg\_num値を自動的に選択することを可能にします。これは Ceph Nautilus以降で利用可能です。

調整が有効になる前に、PGオートスケーラーモジュールの有効化が必要になる場合があります。

```
ceph mgr module enable pg_autoscaler
```

オートスケーラーはプール単位で設定され、以下のモードがあります：

warn	提案された pg_num 値が現在の値から大きく異なる場合、ヘルス警告が発行されます。
on	pg_num は手動操作を必要とせず自動的に調整されます。off pg_num の自動調整は行われず、PG カウントが最適でない場合でも警告は発行されません。

スケーリング係数は、将来のデータ保存を容易にするために target\_size とともに調整できます。

スケーリング係数は、target\_size、target\_size\_rat

および pg\_num\_min オプションを使用して、将来のデータ保存を容易にするためにスケーリング係数を調整できます。



#### 警告

デフォルトでは、オートスケーラーは、PG カウントが 3 倍の差がある場合に、プールの PG カウントの調整を検討します。これにより、データの配置が大幅に変化し、クラスタに高い負荷がかかる可能性があります。

Cephのブログ「[Nautilusの新機能：PGマージと自動チューニング](#)」で、PGオートスケーラーの詳細な紹介をご覧いただけます。

## 8.9 Ceph CRUSH & デバイスクラス

The screenshot shows the Proxmox VE management interface. On the left, the navigation sidebar is expanded to show the Ceph section. In the main area, there are two tabs: 'Configuration' and 'Crush Map'. The 'Configuration' tab displays the Ceph configuration file with sections for [global], [client], [mds], and [mds.prod3]. The [global] section includes parameters like auth\_client\_required=cephx and auth\_cluster\_required=cephx. The [client] section specifies keyring paths. The [mds] section lists hosts and standby configurations. The [mds.prod3] section shows 'host = prod3' and 'mds standby for name = pve'. The 'Crush Map' tab shows the CRUSH map with various devices (osd.0 to osd.17) and their characteristics (class: hdd, ssd). It also lists types (host, chassis, rack, row, pdu, pod, room, datacenter, zone, region, root) and buckets (host prod1 with items osd.3, osd.4, osd.9, osd.10, osd.11, osd.12, osd.13, osd.15, osd.16, osd.17 and alg straw2, hash 0, item osd.0 weight 0.52).

<sup>11</sup> (Controlled Replication Under Scalable Hashing) アルゴリズムは Ceph の基盤となっています。

CRUSH は、データの保存先と取得先を計算します。これにより、中央のインデックスサービスが不要になるという利点があります。CRUSH は、OSD、バケット（デバイスの場所）、およびプール用のルールセット（データ複製）のマップを使用して機能します。

### 注

詳細については、Ceph ドキュメントの「CRUSH マップ<sup>a</sup>」のセクションをご覧ください。

<sup>a</sup> CRUSH map <https://docs.ceph.com/en/squid/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ます。オブジェクトのレプリカは分離可能（例：障害ドメイン）でありながら、望ましい分散を維持します。

一般的な構成として、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が挙げられます。このためCephは、ルールセット生成の簡便性を実現するため、luminousでデバイスクラスを導入しました。

デバイスクラス / `ceph osd tree` の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ており、以下のコマンドで確認可能です。

```
ceph osd crush tree --show-shadow
```

<sup>11</sup> <https://ceph.com/assets/pdfs/weil-crush-sc06.pdf>

上記コマンドの出力例:

ID	クラス	ウェイト	タイプ名
-16	nvme	2.18307	ルート デフォルト~nvme
-13	nvme	0.72769	ホスト sumil~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 sumil
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
14	nvme	0.72769	osd.14

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プールに指示するには、まずそのデバイスクラス用のルールセットを作成する必要があります:

```
ceph osd crush rule create-replicated <rule-name> <root> <failure-domain><-->
class>;
```

&lt;rule-name&gt;	プールに接続するためのルールの名前 (GUIおよびCLIで表示されます)
&lt;root&gt;	属するクラッシュルート (デフォルトはCephルート「default」)
&lt;failure-domain&gt;	オブジェクトを分散させる障害ドメイン (通常はホスト)
&lt;class&gt;	使用するOSD/バックングストアの種類 (例: nvme、ssd、hdd)

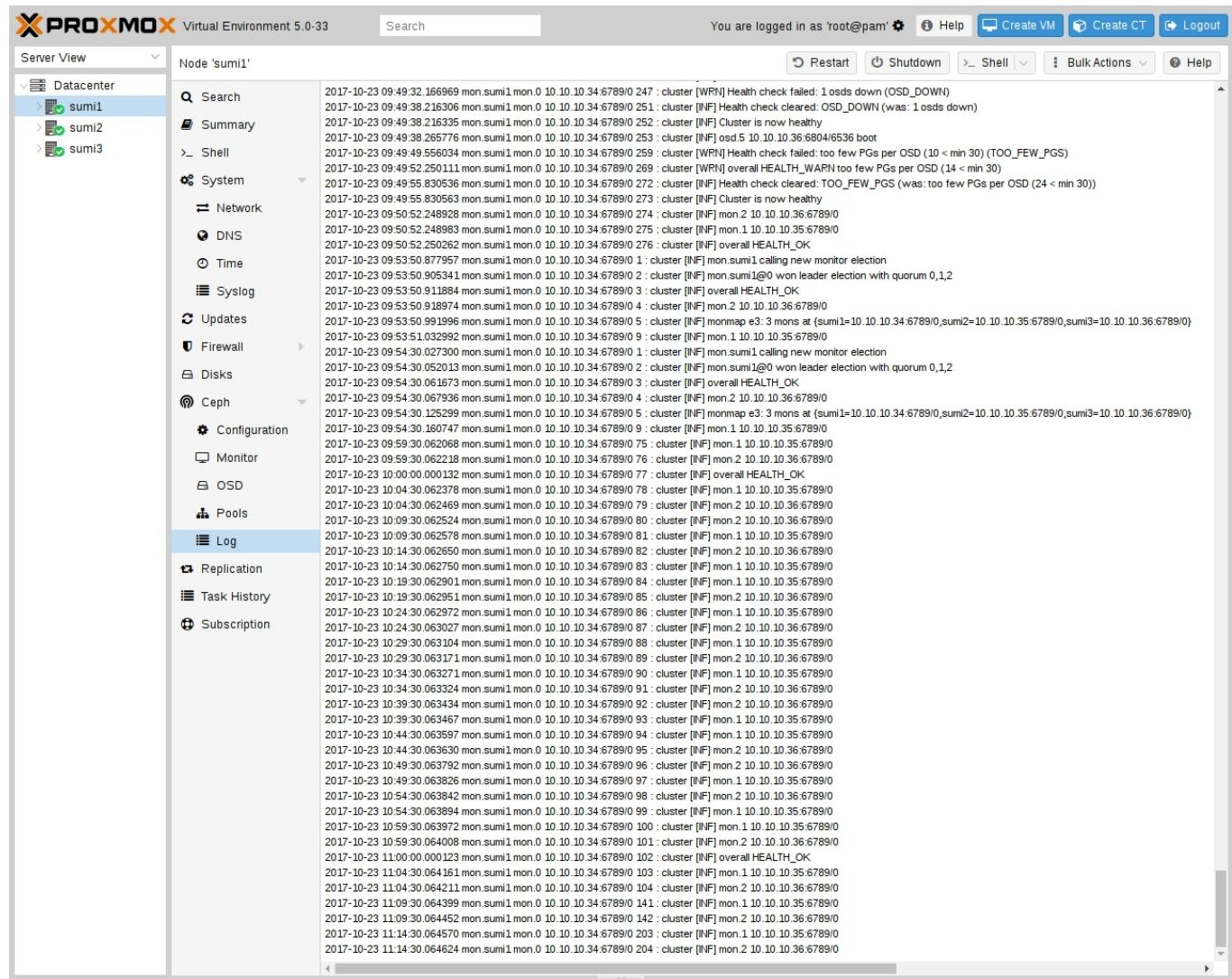
ルールがCRUSHマップに追加されたら、プールにルールセットの使用を指示できます。

```
ceph osd pool set <pool-name> crush_rule <rule-name>;
```

### ヒント

プールに既にオブジェクトが存在する場合、それらを適切に移動する必要があります。設定によっては、クラスターに大きなパフォーマンス影響が生じる可能性があります。代替手段として、新しいプールを作成し、ディスクを個別に移動する方法もあります。

## 8.10 Cephクライアント



前のセクションの設定に従い、Proxmox VE を構成して、VM およびコンテナイメージの保存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（セクション「[Ceph RADOS ブロックデバイス \(RBD\)](#)」を参照）。

また、外部 Ceph クラスター用にキーリングを事前定義された場所に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。

### 注記

ファイル名は `storage\_id` + `keyring` である必要があります。ここで `storage\_id` は `/etc/pve/storage.cfg` 内の `rbd` の後の表現です。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.←
キーリング
```

## 8.11 CephFS

Cephはファイルシステムも提供しており、これは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。メタデータサーバー（MDS）は、RADOSでバックアップされたオブジェクトをファイルやディレクトリにマッピングするために使用され、CephがPOSIX準拠の複製ファイルシステムを提供することを可能にします。これにより、クラスタ化された高可用性の共有ファイルシステムを簡単に設定できます。Cephのメタデータサーバーは、ファイルがCephクラスター全体に均等に分散されることを保証します。その結果、高負荷時でも単一ホストが過負荷になることはありません。これは、NFSなどの従来の共有ファイルシステムアプローチでは問題となる可能性があります。

Proxmox VEは、ハイパーコンバージドCephFSの作成と、既存のCephFSをバックアップ・ISOファイル・コンテナテンプレートの保存用ストレージとして利用する両方をサポートします。

### 8.11.1 メタデータサーバー (MDS)

CephFSが機能するためには、少なくとも1台のメタデータサーバーが設定され稼働している必要があります。MDSは、Proxmox VEのWeb GUIの「ノード」→「CephFS」パネルから、または以下のコマンドラインから作成できます：

```
pveceph mds create
```

クラスター内に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同時にアクティブにできるのは1台のみです。MDSまたはそのノードが応答しなくなった場合（またはクラッシュした場合）、別のスタンバイMDS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MDSとスタンバイMDS間の引き継ぎを高速化できます。既に作成済みの場合は、以下を設定/追加できます：

```
mds standby replay= true
```

これにより、指定されたMDSはアクティブMDSをポーリングするウォーム状態を維持し、問題発生時に迅速に引継ぎが可能になります。

#### 注意

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MDSに追加のパフォーマンス影響を与えます。

#### 複数のアクティブ MDS

Luminous (12.2.x) 以降では、複数のアクティブなメタデータサーバーを同時に実行できますが、これは通常、並列で動作するクライアントが大量にある場合にのみ有用です。そうでない場合、MDS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。<sup>12</sup>

### 8.11.2 CephFSの作成

Proxmox VEのCephFS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。これを機能させるにはいくつかの前提条件が必要です：

CephFS 設定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](#) - 以前にインストール済みの場合でも、最新のシステム環境で再実行し、CephFS関連パッケージが全てインストールされていることを確認してください。
- [モニターの設定](#)
- [OSD の設定](#)
- [少なくとも1台のMDSを設定](#)

これらが完了したら、Web GUIの「Node」→「CephFS」パネル、またはコマンドラインツールpvecephのいずれかを使用して、CephFSを作成できます。パネル、またはコマンドラインツール pveceph から簡単に作成できます。例：

```
pveceph fs create --pg_num 128 --add-storage
```

これにより、*cephfs* という名前の CephFS が作成されます。データ用プール *cephfs\_data* は 128 の配置グループを持ち、メタデータ用プール *cephfs\_metadata* はデータプールの配置グループの 4 分の 1(32) になります。適切な配置グループ数 (pg\_num) については、[Proxmox VE 管理 Ceph ブール章](#)を参照するか、Ceph 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 の作成成功後に Proxmox VE の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 MDS デーモンを設定する <https://docs.ceph.com/en/squid/cephfs/multimds/>

### 8.11.3 CephFS の破棄



警告

CephFS を破棄すると、その中のすべてのデータが使用不能になります。これは元に戻せません！

CephFSを完全に正常に削除するには、以下の手順が必要です：

- Proxmox VE以外のすべてのクライアントを切断してください（例：ゲスト内のCephFSをアンマウント）。
- 関連するすべてのCephFS Proxmox VEストレージエントリを無効化（自動マウントを防止するため）。
- 破壊対象のCephFS上にあるゲスト（例：ISO）から使用中のリソースをすべて削除してください。
- クラスタノード上の CephFS ストレージをすべて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 内の CephFS ストレージの名前です。

- 次に、そのCephFSに対してメタデータサーバー（MDS）が実行されていないことを確認してください。停止するか削除することで実現できます。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から実行可能です。後者の場合、以下のコマンドを発行します：

```
pveceph stop --service mds.NAME
```

停止する場合、または

```
pveceph mds destroy NAME
```

で削除します。

アクティブな MDS 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されることに注意してください。そのため、まずすべてのスタンバイサーバーを停止することをお勧めします。

- これで、以下のコマンドで CephFS を破棄できます。

```
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るとともに、ストレージがPVE設定から削除されます。

これらの手順を完了すると、CephFSは完全に削除されます。他のCephFSインスタンスがある場合、停止したメタデータサーバーを再起動してスタンバイとして機能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

以下の手順で、OSD のディスクを交換することができます。これは、Ceph で最も一般的なメンテナンス作業のひとつです。OSD に問題が発生しても、そのディスクは正常であると思われる場合は、まず [トラブルシューティング](#) のセクションをお読みください。

1. ディスクが故障した場合は、[推奨される](#)同じタイプおよびサイズの交換用ディスクを入手してください。
2. 問題のあるOSDを[破棄します](#)。
3. サーバーから古いディスクを取り外し、新しいディスクを取り付けます。
4. OSDを再[作成](#)します。
5. 自動リバランス後、クラスター状態はHEALTH\_OKに戻るはずです。まだリストされているクラッシュは、以下のコマンドを実行することで承認できます：

```
ceph crash archive-all
```

### 8.12.2 トリム/ディスカード

VMやコンテナでは定期的にfstrim (discard) を実行することが推奨されます。これによりファイルシステムが使用しなくなったデータブロックが解放され、データ使用量とリソース負荷が軽減されます。最新のOSの多くは定期的にディスクにdiscardコマンドを発行します。仮想マシンで[ディスクのdiscardオプションが有効](#)になっていることを確認するだけで十分です。

### 8.12.3 スクラップ&ディープスクラップ

Cephは配置グループのスクラップによりデータ整合性を確保します。CephはPG内の全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2種類あり、日次実行の低成本なメタデータチェックと週次実行のディープデータチェックです。週次ディープスクラップはオブジェクトを読み込み、チェックサムを用いてデータ整合性を確認します。実行中のスクラップが業務（パフォーマンス）要件に干渉する場合は、スクラップ<sup>13</sup>の実行時間を調整できます。

### 8.12.4 Proxmox VE+ Ceph HCI クラスターのシャットダウン

Proxmox VE + Cephクラスター全体をシャットダウンするには、まず全てのCephクライアントを停止します。これらは主にVMとコンテナです。Cephファイルシステムまたはインストール済みのRADOS GWにアクセスする可能性のある追加クライアントがある場合は、それらも停止してください。高可用性ゲストは、Proxmox VEツールで電源を切ると停止状態に切り替わります。

すべてのクライアント、VM、コンテナが停止しているか、Cephクラスターにアクセスしなくなったことを確認した後、Cephクラスターが健全な状態にあること

```
ceph -s
```

を検証してください。Web UIまたはCLIのいずれかを使用して実行します：

シャットダウン時および後の電源投入フェーズでリカバリが発生しないよう、noout OSDフラグを有効化してください。Cephの「→」OSDパネル内にある「Manage Global Flags」ボタンから、またはCLIで設定します：

```
ceph osd set noout
```

モニターノード (MON) を先にシャットダウンします。これらのノードが停止した後、モニターノードを含むノードのシャットダウンを続行します。

クラスターの電源投入時には、モニターノード (MON) から起動してください。全ノードの起動完了後、Ceph全サービスが稼働していることを確認します。最終的にCephで表示される警告は、nooutフラグが設定されたままであることのみです。Web UIまたはCLIで無効化できます：

<sup>13</sup> Ceph scrubbing <https://docs.ceph.com/en/squid/rados/configuration/osd-config-ref/#scrubbing>

```
ceph osd unset noout
```

ゲストの起動を開始できます。高可用性ゲストは電源投入時に状態が「started」に変わります。

## 8.13 Cephの監視とトラブルシューティング

Cephデプロイメントの健全性は、初期段階から継続的に監視することが重要です。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ことで監視できます。

以下のCephコマンドを使用すると、クラスターが正常状態（*HEALTH\_OK*）か、警告状態（*HEALTH\_WARN*）か、あるいはエラー状態（*HEALTH\_ERR*）かを確認できます。クラスターが正常でない状態の場合、下記のステータスコマンドは現在のイベントの概要と取るべきアクションも表示します。実行を停止するにはCTRL-Cを押してください。

クラスタ状態を継続的に監視:

```
watch ceph --status
```

クラスタ状態を一度だけ表示（更新されない）し、状態イベントの行を継続的に追加表示:

```
ceph --watch
```

### 8.13.1 トラブルシューティング

このセクションには、頻繁に使用されるトラブルシューティング情報が含まれています。詳細は、公式Cephウェブサイトのトラブルシューティング<sup>14</sup>を参照してください。

影響を受けたノードの関連ログ

- ディスクの健全性監視
- システム→システムログまたはCLI経由で、例えば過去2日間のログ:  

```
journalctl --since "2 days ago"
```
- IPMIおよびRAIDコントローラーのログ

Cephサービスのクラッシュは、以下のコマンドを実行することで一覧表示および詳細確認が可能です:

```
ceph crash ls  
ceph crash info &lt;crash_id&gt;
```

新規としてマークされたクラッシュは、以下のコマンドを実行して確認できます:

```
ceph クラッシュ アーカイブ-すべて
```

<sup>14</sup> Ceph トラブルシューティング <https://docs.ceph.com/en/squid/rados/troubleshooting/>

より詳細な情報を得るには、各Cephサービスが/var/log/ceph/下にログファイルを持っています。詳細が必要な場合は、ログレベルを調整できます<sup>15</sup>。

## CEPH 問題の一般的な原因

- 輻輳、スイッチの故障、シャットダウンしたインターフェース、ファイアウォールのブロックなどのネットワーク問題。すべての Proxmox VE ノードが、corosync クラスタネットワーク、Ceph パブリックネットワーク、クラスタネットワーク上で確実に到達可能かどうかを確認してください。
- ディスクまたは接続部品の以下の状態：
  - 不良
  - 堅牢にマウントされていない
  - 高負荷時のI/O性能不足（例：HDD、民生用ハードウェア、[推奨されないRAIDコントローラー](#)の使用時）
- 健全なCephクラスターの[推奨要件](#)を満たしていない。

## 一般的なCEPHの問題

### OSDがダウン/クラッシュした

障害のあるOSDはダウンとして報告され、ほとんどの場合10分後に（自動的に）除外されます。状況に応じて

原因によっては、自動的に再び稼働状態に戻ることがあります。ウェブインターフェース経由で手動起動を試すには、いずれかのノードで→ Ceph→ OSD にアクセスし、OSD を選択して「Start」 「In」 「Reload」をクリックしてください。シェルを使用する場合は、影響を受けたノードで次のコマンドを実行します：

```
ceph-volume lvm activate --all
```

障害発生OSDをアクティブ化するには、該当ノードの[安全な再起動](#)が必要となる場合があります。最終手段としてOSDの再作成または置換も検討してください。

<sup>15</sup> Cephのログとデバッグ <https://docs.ceph.com/en/squid/rados/troubleshooting/log-and-debug/>

## 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複製するため、共有ストレージを使用せずに全データを利用可能です。複製処理ではスナップショットを利用し、ネットワーク経由のトラフィックを最小限に抑えます。したがって、初期の完全同期後は新規データのみが増分送信されます。ノード障害時でも、複製ノード上でゲストデータは引き続き利用可能です。

レプリケーションは設定可能な間隔で自動的に実行されます。最小レプリケーション間隔は1分、最大間隔は1週間に1回です。これらの間隔を指定する形式はsystemdカレンダーイベントのサブセットです。詳細は「[スケジュール形式](#)」セクションを参照してください:

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、同一ターゲットノードへの二重レプリケーションは不可です。ストレージやサーバーの過負荷を避けるため、各レプリケーションの帯域幅を制限できます。

ゲストが既にレプリケーションされているノードへ移行する場合、前回のレプリケーション以降の変更部分（いわゆる差分）のみを転送すれば済みます。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対象ノードへ移行すると、レプリケーション方向は自動的に切り替わります。

例: VM100が現在ノードAにあり、ノードBにレプリケートされている場合、これをノードBに移行すると、自動的にノードBからノードAへ逆レプリケーションが開始されます。

ゲストがレプリケーションされていないノードへ移行する場合、ディスクデータ全体を転送する必要があります。移行後、レプリケーションジョブは設定されたノードへのゲストのレプリケーションを継続します。



### 重要

高可用性機能はストレージレプリケーションと併用可能ですが、最終同期時刻とノード障害発生時刻の間にデータ損失が発生する可能性があります。

## 9.1 サポートされているストレージタイプ

表 9.1: ストレージの種類

説明	プラグインタイプ	スナップショット	安定
ZFS (ローカル)	zfspool	はい	はい

## 9.2 スケジュール形式

レプリケーションは、スケジュール設定に[カレンダーイベント](#)を使用します。

## 9.3 エラー処理

レプリケーションジョブで問題が発生した場合、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が一時的に停止されます。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。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 想定される問題

最も一般的な問題の一部を以下に示します。設定によっては別の原因が考えられます。

- ネットワークが機能していない。
- レプリケーション対象ストレージに空き容量が残っていない。
- ターゲットノードに同じストレージIDを持つストレージが存在しない。

#### 注記

問題の原因を特定するには、常にレプリケーションログを利用できます。

### 9.3.2 エラー発生時のゲスト移行

重大なエラーが発生した場合、仮想ゲストが障害ノードで停止する可能性があります。その際は手動で正常なノードへ再移動させる必要があります。

### 9.3.3 例

ノードA上で動作しノードBにレプリケートされている2つのゲスト（VM 100とCT 200）があると仮定します。ノードAが障害を起こしオンライン復帰できません。この場合、ゲストを手動でノードBへ移行する必要があります。

- SSH経由でノードBに接続するか、Web UIからそのシェルを開く
- クラスタがクオーラム状態か確認する

```
# pvecm status
```

- クオーラムが成立していない場合、まずこの問題を修正しノードを再稼働させることを強く推奨します。それが現時点で不可能な場合に限り、以下のコマンドで現在のノードにクオーラムを強制適用できます:

```
# pvecm expected 1
```



## 警告

クオーラム設定中にクラスターに影響する変更（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。本コマンドは、重要なゲストの再起動やクオーラム問題自体の解決にのみ使用してください。

- ゲスト設定ファイルを両方とも元のノード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
```

- これでゲストを再起動できます:

```
# qm start 100 # pct
start 200
```

VMIDとノード名は、それぞれご自身の環境に合わせて置き換えてください。

## 9.4 ジョブの管理

Enabled	Guest ↑	Job ↑	Target	Status	Last Sync	Dur...	Next Sync

Web GUI を使用して、レプリケーションジョブを簡単に作成、変更、削除できます。さらに、コマンドラインインターフェイス (CLI) ツール `pvresr` を使用してこれを行うこともできます。

Web GUIでは、すべての階層（データセンター、ノード、仮想ゲスト）でレプリケーションパネルを確認できます。表示されるジョブは、全ジョブ、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には、ゲストが既に選択されていない場合、およびターゲットノードを指定する必要があります。デフォルトの15分間隔が望ましくない場合、[レプリケーションのスケジュール](#)を設定できます。レプリケーションジョブにレート制限を課すことが可能です。レート制限はストレージへの負荷を許容範囲内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で識別されます。このIDはVMIDとジョブ番号で構成されます。このIDはCLIツールを使用する場合にのみ手動で指定する必要があります。

## 9.5 ネットワーク

レプリケーショントラフィックは、ライブゲスト移行と同じネットワークを使用します。デフォルトでは、これは管理ネットワークです。移行に別のネットワークを使用するには、ウェブインターフェースの「データセンター」→「オプション」→「移行設定」または`datacenter.cfg`で移行ネットワークを設定してください。詳細は[移行ネットワーク](#)を参照してください。

## 9.6 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、5 分ごとに実行され、帯域幅を 10 Mbps（メガバイト毎秒）に制限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 pvel --schedule "* /5" --rate 10
```

アクティブなジョブID 100-0を無効化します。

```
# pvesr disable 100-0
```

ID 100-0 の無効化されたジョブを有効化します。

```
# pvesr enable 100-0
```

ジョブID 100-0 のスケジュール間隔を1時間ごとに変更します。

```
# pvesr update 100-0 --schedule '* /00'
```

## 第10章

### QEMU/KVM仮想マシン

QEMU (Quick Emulatorの略称) 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が動作するホストシステムの観点から見ると、QEMUはパーティション、ファイル、ネットワークカードなどのローカルリソースにアクセスできるユーザープログラムであり、これらのリソースはエミュレートされたコンピュータに渡され、あたかも実デバイスであるかのように認識されます。

エミュレートされたコンピュータ上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のパラメータとして渡すことで、エミュレートされたコンピュータ内で動作するOSは、CDドライブに挿入された实物のCD-ROMを認識します。

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が対応するのは32ビットおよび64ビットのPCクローンエミュレーションのみです。これはサーバーハードウェアの圧倒的多数を占めるためです。PCクローンのエミュレーションは、ホストアーキテクチャとエミュレート対象アーキテクチャが同一の場合、プロセッサ拡張機能の利用によりQEMUの速度が大幅に向上するため、最も高速なエミュレーションの一つでもあります。

---

#### 注記

KVM（カーネルベース仮想マシン）という用語を目にすることがあるかもしれません。これは、Linux KVMモジュールを介して仮想化プロセッサ拡張機能のサポートを得てQEMUが動作していることを意味します。Proxmox VEの文脈では、QEMUとKVMは互換的に使用できます。Proxmox VE内のQEMUは常にKVMモジュールのロードを試みるためです。

---

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、root 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されたデバイス

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およびSATAコントローラ、シリアルポート（完全なリストはkvm (1) マニュアルページで確認可能）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ます。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完全に同等のソフトウェアであり、ゲストで動作する OS が適切なドライバを持っている場合、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デバイスを使用します。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。

ただし、ハードウェアで動作するはずの機能をソフトウェアで実行するため、ホストCPUに多くの余分な処理負荷がかかるというパフォーマンス上の代償があります。これを軽減するため、QEMUはゲストOSに準仮想化デバイスを提供できます。準仮想化デバイスでは、ゲストOSはQEMU内で動作していることを認識し、ハイパーバイザーと連携します。

---

QEMUはvirtio仮想化標準に依存しているため、準仮想化されたvirtioデバイスを提供できます。これには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コントローラなどが含まれます。

### ヒント

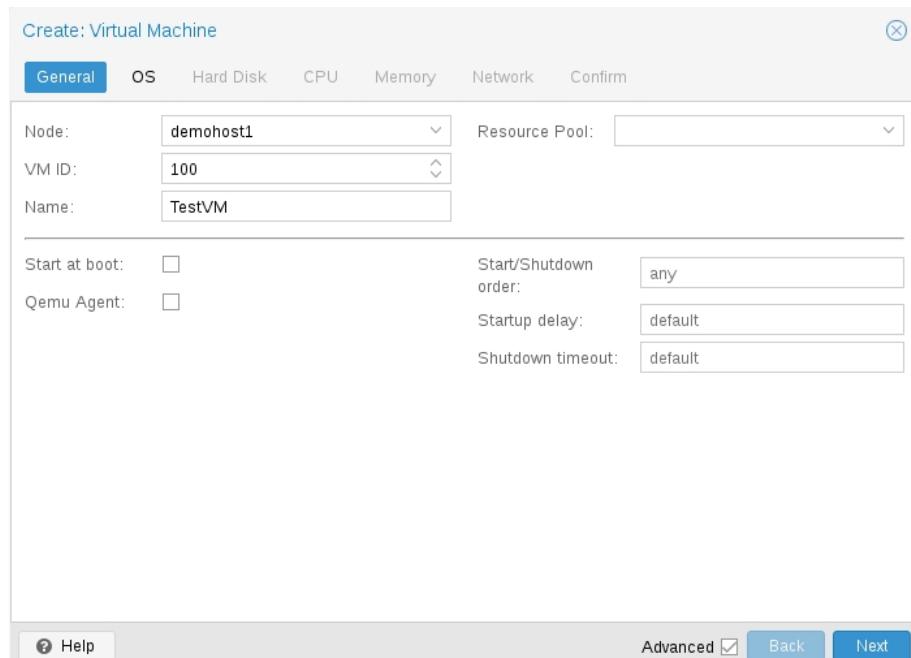
可能な限りvirtioデバイスを使用することを強く推奨します。virtioデバイスは大幅なパフォーマンス向上をもたらし、一般的にメンテナンス性にも優れています。emu-lated IDEコントローラと比較してvirtio汎用ディスクコントローラを使用すると、bonnie++ (8)による測定で順次書き込みスループットが2倍になります。virtioネットワークインターフェースを使用すると、iperf(1)で測定した場合、エミュレートされたIntel E1000ネットワークカードのスループットの最大3倍を実現できます。<sup>(a)</sup>

<sup>a</sup> KVM wikiのこのベンチマークを参照 [https://www.linux-kvm.org/page/Using\\_VirtIO\\_NIC](https://www.linux-kvm.org/page/Using_VirtIO_NIC)

## 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設定を選択しようとします。変更する設定の意味を理解していることを確認してください。パフォーマンスの低下やデータのリスクにつながる可能性があります。

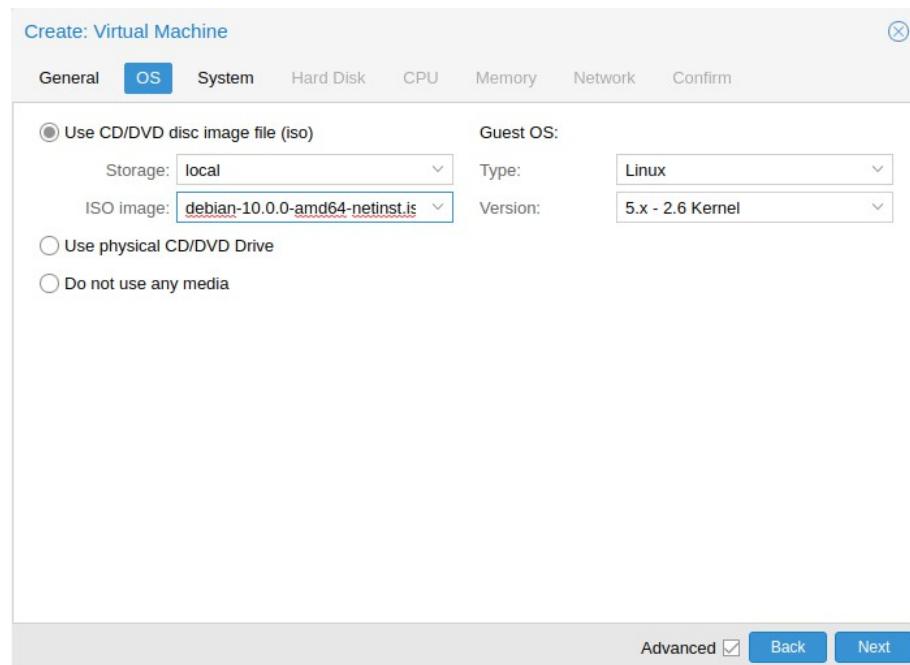
### 10.2.1 一般設定



VM の一般設定には以下が含まれます

- ノード: VMが実行される物理サーバー
- VM ID: このProxmox VEインストール環境内でVMを識別するための一意の番号
- 名前: VM を記述するために使用できる自由形式のテキスト文字列
- リソースプール: VMの論理的なグループ

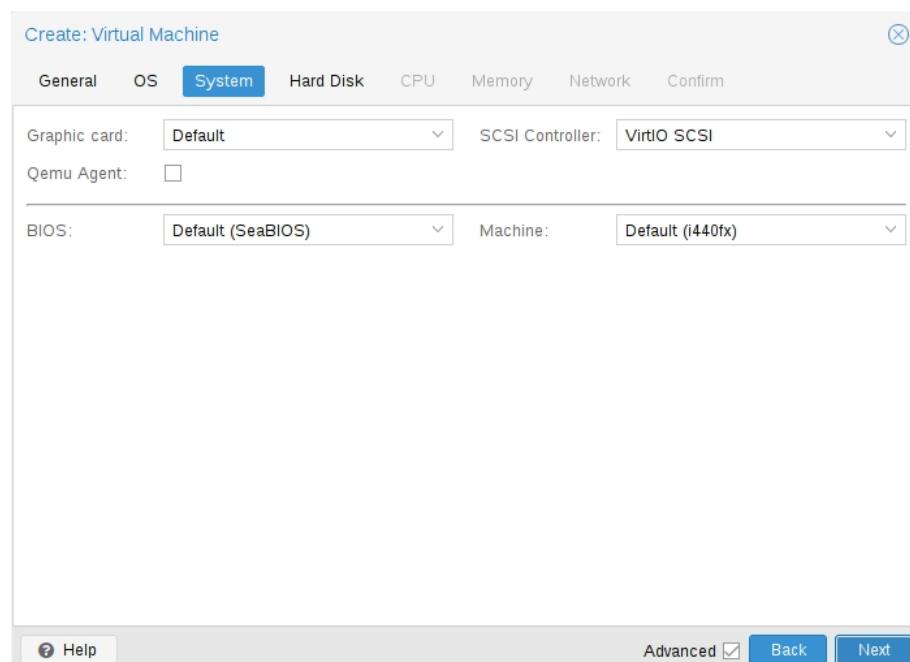
## 10.2.2 OS設定



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想定しているのに対し、UnixベースのOSはBIOSクロックがUTC時間を持つことを想定しています。

## 10.2.3 システム設定

VM作成時に、新規VMの基本的なシステムコンポーネントを変更できます。使用する[ディスプレイタイプ](#)を指定できます。



さらに、[SCSIコントローラー](#)を変更することも可能です。QEMUゲストエージェントをインストールする予定の場合、または選択したISOイメージが既にそれを自動的に提供・インストールする場合、[QEMUエージェント]チェックボックスをオンにすることをお勧めします。

これによりProxmox VEは、追加情報の表示やシャットダウン・スナップショットなどの操作をより効率的に実行できるようになります。

Proxmox VEでは、異なるファームウェアとマシンタイプ（[具体的にはSeaBIOSとOVMF](#)）で仮想マシンを起動できます。ほとんどの場合、[PCIeバススルー](#)を使用する予定がある場合にのみ、デフォルトのSeaBIOSからOVMFに切り替えることをお勧めします。

## マシンタイプ

マシンのタイプは、仮想マザーボードのハードウェア構成を定義します。デフォルトの[Intel 440FX](#)チップセットか、仮想PCIeバスを提供する[Q35](#)チップセットから選択できます。PCIeハードウェアのバススルーを必要とする場合、後者が適しています。さらに、[VIOMMU](#)の実装を選択することも可能です。

## マシンバージョン

各マシンタイプはQEMU内でバージョン管理されており、特定のQEMUバイナリは複数マシンバージョンをサポートします。新バージョンでは新機能のサポート、修正、全般的な改善が追加される場合があります。ただし、仮想ハードウェアのプロパティも変更されます。ゲスト視点での急な変更を回避し、VM状態・ライブマイグレーション・RAMスナップショットの互換性を確保するため、新しいQEMUインスタンスでは同じマシンバージョンが継続使用されます。

Windowsゲストの場合、マシンバージョンは作成時に固定されます。これはWindowsが仮想ハードウェアの変更に敏感であるためです（コールドブート間でも影響を受けます）。例えば、ネットワークデバイスの列挙順序がマシンバージョンによって異なる場合があります。Linuxなどの他のOSは通常、このような変更に問題なく対応できます。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。つまり、新規起動後はQEMUバイナリがサポートする最新のマシンバージョン（例：QEMU 8.1が各マシンタイプでサポートする最新バージョンは8.1）が適用されます。

マシンバージョンは、ハードウェアレイアウトを変更する新機能や修正を実装する際の安全策としても使用され、下位互換性を確保します。稼働中のVMに対する操作（ライブマイグレーションなど）では、稼働中のマシンバージョンが保存されます。これにより、QEMU仮想化の観点だけでなく、Proxmox VEがQEMU仮想マシンインスタンスを作成する方法においても、VMが正確に元の状態に復元されることが保証されます。

## PVE マシンリビジョン

Proxmox VEでは、新しいQEMUリリースの待ち時間なくハードウェアレイアウトの変更やオプションの修正が必要な場合があります。このため、Proxmox VEは+pveX形式の追加リビジョンを導入しました。これらのリビジョンでは、Xは各新しいQEMUマシンバージョンごとに0となり、この場合は省略されます。例えば、マシンバージョンpc-q35-9.2は、マシンバージョンpc-q35-9.2+pve0と同じになります。

Proxmox VEがハードウェアレイアウトやデフォルトオプションを変更する場合、リビジョンがインクリメントされ、新規作成されるゲストや、常に最新のマシンバージョンを使用するVMの再起動時に適用されます。

## QEMUマシンバージョンの廃止

QEMU 10.1以降、マシンバージョンは6年後にアップストリームQEMUから削除されます。Proxmox VEではメジャーリリースが約2年ごとに発生するため、メジャーProxmox VEリリースは過去2つのメジャーProxmox VEリリース程度のマシンバージョンをサポートします。

新しいメジャー Proxmox VE リリースにアップグレードする前に、次のメジャー Proxmox VE リリースで廃止されるすべてのマシンバージョンを回避するようVM構成を更新する必要があります。これにより、そのリリース期間中もゲストを引き続き使用できるようになります。[「新しいマシンバージョンへの更新」](#)セクションを参照してください。

Proxmox VE 8 では削除ポリシーはまだ有効になっていないため、サポート対象のマシンバージョンの基準は 2.4 です。Proxmox VE 9 向けにリリースされる最後のQEMUバイナリバージョンはQEMU 11.2 となる見込みです。このQEMUバイナリでは6.0より古いマシンバージョンのサポートが削除されるため、6.0がProxmox VE 9リリースライフサイクルのベースラインとなります。ベースラインは、Proxmox VE のメジャーリリースごとに 2 バージョンずつ増加する見込みです。例えば、Proxmox VE 10 では 8.0 となります。

#### 新しいマシンバージョンへの更新

非推奨の警告が表示された場合は、マシンバージョンを新しいものに変更してください。まず動作するバックアップを確保し、ゲストがハードウェアを認識する方法が変更される可能性に備えてください。状況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場合があります。また、これらのマシンバージョン（つまりrunningmachine設定エントリ）で作成されたRAMを含むスナップショットを確認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ため、スナップショットからデータを救出する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

QEMUは複数のストレージコントローラをエミュレートできます：

##### ヒント

パフォーマンス上の理由と、より適切なメンテナンスが行われていることから、**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を採用しています。このコントローラは近年の設計に取って代わられていますが、考えられるあらゆるOSがこれをサポートしているため、2003年以前にリリースされたOSを実行したい場合に最適な選択肢となります。このコントローラには最大4台のデバイスを接続できます。
  - SATA（シリアルATA）コントローラーは2003年に登場したより現代的な設計で、より高いスループットと多数のデバイス接続を可能にします。このコントローラーには最大6台のデバイスを接続できます。
  - 1985年に設計された**SCSI**コントローラーは、サーバーグレードのハードウェアに一般的に搭載されており、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コントローラー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、接続ディスクに対してIOスレッド設定を有効化することを推奨します。これはProxmox VE 7.3以降で新規作成されるLinux VMのデフォルト設定です。各ディスクは専用の*VirtIO SCSI*コントローラを持ち、QEMUが専用スレッドでディスクのI/O処理を管理します。Linuxディストリビューションでは2012年以降、FreeBSDでは2014年以降、このコントローラをサポートしています。Windows OSの場合、インストール時にドライバを含む追加のISOイメージを提供する必要があります。
- VirtIOブロックコントローラ**（しばしば単にVirtIOまたはvirtio-blkと呼ばれる）は、より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*コントローラに取って代わられています。

## イメージ形式

各コントローラには、設定済みのストレージ上に存在するファイルまたはブロックデバイスを裏付けとする、複数のエミュレートされたハードディスクを接続します。ストレージタイプの選択によって、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提供するストレージ (LVM、ZFS、Ceph) では生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、XFS、NFS、CIFSなど) では生ディスクイメージ形式かQEMUイメージ形式のいずれかを選択できます。

- **QEMUイメージ形式**はコピー온라이트方式であり、スナップショットの作成やディスクイメージのシンプロビジョニングが可能です。
- **生のディスクイメージ**は、ハードディスクのビット単位のイメージであり、Linuxでブロックデバイスに対してddコマンドを実行した際に得られるものと類似しています。この形式自体はシンプロビジョニングやスナップショットをサポートしておらず、これらのタスクにはストレージ層の協力が必要です。ただし、**QEMUイメージ形式よりも最大10%高速**である可能性があります。<sup>1</sup>
- **VMwareイメージ形式**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/エクスポートする意図がある場合にのみ意味を持ちます。

## キャッシングモード

ハードドライブのキャッシングモード設定は、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「キャッシングなし」は、ホストのページキャッシングを無視し、各ブロックが物理ストレージの書き込みキューに到達した時点で書き込み完了をゲストシステムに通知することを意味します。これは安全性と速度のバランスに優れています。

Proxmox VE バックアップマネージャーが VM のバックアップ時に特定のディスクをスキップするように設定したい場合、そのディスクに対してそのディスクに対してバックアップ対象外オプションを設定できます。

Proxmox VEのストレージレプリケーションメカニズムがレプリケーションジョブ開始時に特定のディスクをスキップするように設定するには、そのディスクに対して「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以降、レプリケーションにはディスクイメージが<sup>2</sup>zfspoolタイプのストレージ上に存在することが必須です。したがって、レプリケーションが設定されたVMのディスクイメージを他のストレージに追加する場合、そ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。

## Trim/Discard

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章を参照) 、ドライブでDiscardオプションを有効化できます。Discardを設定し、TRIM対応のゲストOS<sup>2</sup>を実行している場合、VMのファイルシステムがファイル削除後にブロックを未使用としてマークすると、コントローラーがこの情報をストレージに伝達し、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、ドライブでDiscardオプションを有効にする必要があります。一部のゲストOSではSSDエミュレーションフラグの設定も必要となる場合があります。VirtIO BlockドライブでのDiscardは、Linuxカーネル5.0以降を使用するゲストでのみサポートされている点に注意してください。

ドライブを回転式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提示したい場合、そのドライブでSSDエミュレーションオプションを設定できます。基盤となるストレージが実際にSSDで構成されている必要はなく、あらゆる物理メディアでこの機能を利用可能です。なお、VirtIOブロックドライブではSSDエミュレーションはサポートされていません。

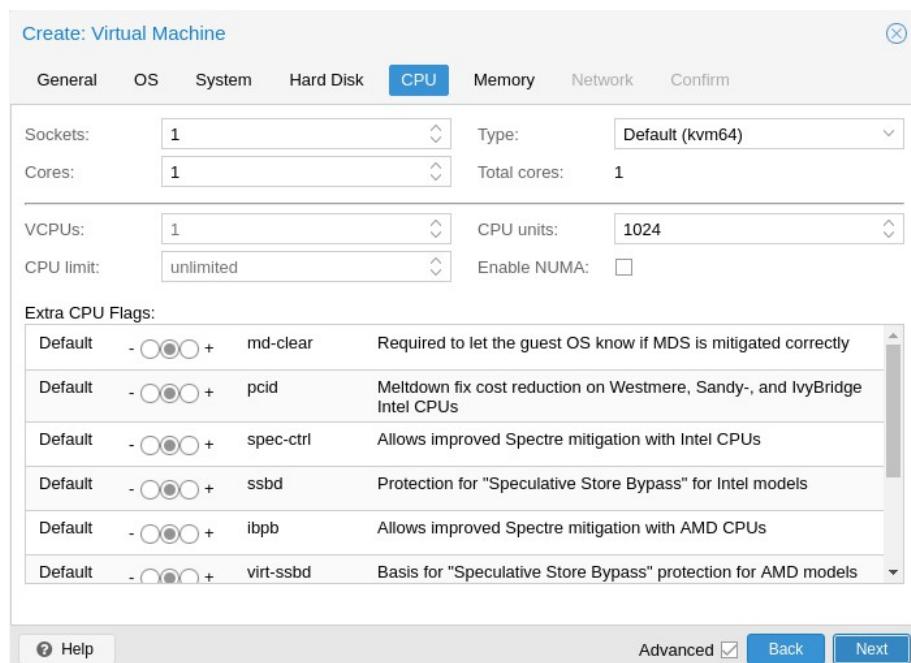
<sup>1</sup> 参照 この ベンチマーク 参照 詳細 [https://events.static.linuxfound.org/sites/events/files/slides-/CloudOpen2013\\_Khoa\\_Huynh\\_v3.pdf](https://events.static.linuxfound.org/sites/events/files/slides-/CloudOpen2013_Khoa_Huynh_v3.pdf)

<sup>2</sup> TRIM、UNMAP、およびdiscard [https://en.wikipedia.org/wiki/Trim\\_%28computing%29](https://en.wikipedia.org/wiki/Trim_%28computing%29)

## IO スレッド

IOスレッドオプションは、VirtIOコントローラを搭載したディスクを使用する場合、またはエミュレートされたコントローラタイプがVirtIO SCSIシングルであるSCSIコントローラを使用する場合にのみ有効です。IOスレッドを有効にすると、QEMUはメインイベントループやvCPUスレッドですべてのI/Oを処理する代わりに、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。これにより、基盤となるストレージの作業分散と利用効率が向上します。もう1つの利点は、I/O負荷が非常に高いホストワークロードにおいて、ゲスト側の遅延（ハング）が低減されることです。これは、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です。

## 10.2.5 CPU



CPUソケットとは、PCマザーボード上の物理的なスロットであり、ここにCPUを差し込むことができます。このCPUは1つまたは複数のコア（独立した処理ユニット）を含むことができます。4コアのCPUソケットが1つある場合と、2コアのCPUソケットが2つある場合とでは、パフォーマンスの観点からはほとんど関係ありません。ただし、一部のソフトウェアライセンスはマシンのソケット数に依存するため、その場合はライセンスが許可するソケット数に設定することが理にかなっています。

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は、仮想CPU数を増やすことで当然恩恵を受けます。追加する仮想CPUごとに、QEMUはホストシステム上で新たな実行スレッドを生成するためです。VMのワークロードが不明な場合、総コア数を2に設定するのが安全策です。

### 注記

すべての仮想マシンのコア総数がサーバーのコア数を超える場合（例：8コアのマシン上で4コアの仮想マシンを4台（合計16コア））でも完全に安全です。その場合、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、サーバーコア間でQEMU実行スレッドをバランス調整します。ただし、物理的に利用可能なコア数を超える仮想CPUコア数を持つVMの起動は、コンテキストスイッチのコストによりパフォーマンスが低下するため、Proxmox VEによって防止されます。

## リソース制限 cpulimit

仮想コア数に加え、VMが利用可能な「ホストCPU時間」の総量はcpulimitオプションで設定可能です。これはCPU時間をパーセントで表す浮動小数点値であり、1.0は100%。

2.5%から250%まで設定可能です。単一プロセスが1つのコアを完全に使用する場合、その使用率は100%となります。

4コアのVMが全コアを完全に利用する場合、理論上は400%を使用します。実際には、vCPUコア用以外にVM周辺機器用の追加スレッドを持つQEMUのため、使用率がさらに若干高くなる可能性があります。

この設定は、VMが複数のプロセスを並行実行するため複数のvCPUを必要とする場合、かつVM全体が同時に全vCPUを100%稼働させてはならない場合に有用です。

例えば、仮想マシンに8つの仮想CPUを割り当てることでパフォーマンスが向上すると仮定します。ただし、その仮想マシンが8コア全てをフル稼働状態で最大化することを望まない場合（サーバーに過負荷がかかり、他の仮想マシンやコンテナのCPU時間が不足する可能性があるため）は、cpulimitを4.0 (=400%)に設定することで解決できます。これは、VMが8つのプロセスを同時に実行して8つの仮想CPUを完全に利用した場合、各vCPUが物理コアから最大50%のCPU時間を割り当てられることを意味します。ただし、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、物理コアから最大100%のCPU時間を割り当てられ、合計で400%に達する可能性があります。

### 注記

VMは構成によっては、ネットワークやI/O操作、ライブマイグレーションなどの追加スレッドを使用する場合があります。そのため、VMが使用するCPU時間が割り当てられた仮想CPU時間を超えることがあります。VMが割り当てられたvCPU時間を絶対に使用しないようにするには、cpulimitをコア総数と同じ値に設定してください。

## cpuunits

cpuunitsオプション（現在ではCPUシェアやCPUウェイトと呼ばれることが多い）を使用すると、他の稼働中のVMと比較して、VMが取得するCPU時間を制御できます。これは相対的な重みであり、デフォルトは100（ホストがレガシー`cgroup v1`を使用している場合は1024）です。VMのこの値を増加させると、スケジューラによって、より低い重みを持つ他のVMと比較して優先されます。

例えば、VM 100 がデフォルトの 100 を設定し、VM 200 を 200 に変更した場合、後者の VM 200 は前者の VM 100 の 2 倍の CPU 帯域幅を受け取ります。

詳細は`man systemd.resource-control`を参照。ここでCPUQuotaはcpulimitに対応する。

CPUWeightをcpuunits設定に追加します。参照情報と実装の詳細については、そのNotesセクションをご覧ください。

## アフィニティ

アフィニティオプションを使用すると、VMのvCPUを実行する物理CPUコアを指定できます。I/O処理などの周辺VMプロセスはこの設定の影響を受けません。なお、CPUアフィニティはセキュリティ機能ではありません。

CPUアフィニティの強制は特定のケースでは有効ですが、複雑性と保守負担の増加を伴います。例えば、後でVMを追加する場合や、VMをCPUコア数の少ないノードに移行する場合などです。また、一部のCPUが完全に利用されている一方で他のCPUがほぼアイドル状態の場合、非同期状態に陥りやすく、結果としてシステムパフォーマンスが制限される可能性があります。

アフィニティは`taskset` CLIツールで設定します。このツールは`man cpuset`に記載のリスト形式でホストCPU番号（`lscpu`参照）を受け付けます。このASCII十進リストには単一番号だけでなく範囲指定も可能です。例えば、アフィニティ 0-1,8-11（展開すると 0, 1, 8, 9, 10, 11）と指定すると、VM はこれらの特定の 6 つのホストコア上でのみ実行可能になります。

## CPUタイプ

QEMUは486から最新のXeonプロセッサまで、様々な**CPUタイプ**をエミュレートできます。各新世代プロセッサは、ハードウェア支援3Dレンダリング、乱数生成、メモリ保護などの新機能を追加します。また現行世代は、バグ修正やセキュリティ修正を含む[マイクロコード更新](#)でアップグレード可能です。

通常、ホストシステムのCPUに可能な限り近いプロセッサタイプをVMに選択すべきです。これによりホストCPUの機能（CPUフラグとも呼ばれる）がVM内で利用可能になります。完全一致を望む場合はCPUタイプを「**host**」に設定すると、VMはホストシステムと全く同一のCPUフラグを持ちます。

ただし、これには欠点があります。異なるホスト間で仮想マシンのライブマイグレーションを実行する場合、仮想マシンが異なるCPUタイプや異なるマイクロコードバージョンの新しいシステムに移動する可能性があります。ゲストに渡されるCPUフラグが欠落している場合、QEMUプロセスは停止します。この問題を解決するため、QEMUには独自の仮想CPUタイプも存在し、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*であり、これは基本的に全てのx86\_64ホストCPUで動作します。一方、新規VM作成時のUIデフォルトは*x86-64-v2-AES*であり、これはIntelではWestmere以降のホストCPU、AMDでは少なくとも第4世代Opteronを必要とします。

要約すると：

ライブマイグレーションを重視しない場合、または全ノードが同一CPU・同一マイクロコードバージョンで構成される均質なクラスター環境では、CPUタイプをホストに設定してください。理論上、これによりゲストのパフォーマンスが最大化されます。

ライブマイグレーションとセキュリティを重視し、かつIntel CPUのみまたはAMD CPUのみの環境では、クラスター内で最も低世代のCPUモデルを選択してください。

セキュリティを重視せずライブマイグレーションのみを重視する場合、またはIntel/AMD混在クラスターの場合は、互換性のある仮想QEMU CPUタイプの中で最も低い世代を選択してください。

---

### 注意

Intel と AMD のホスト CPU 間のライブマイグレーションは動作が保証されません。

---

QEMU で定義されている AMD および Intel CPU タイプのリストも参照してください。

## QEMU CPU タイプ

QEMU は、Intel および AMD ホスト CPU の両方に互換性のある仮想 CPU タイプも提供しています。

---

### 注

仮想CPUタイプにおけるSpectre脆弱性を軽減するには、関連するCPUフラグを追加する必要があります。[詳細はMeltdown / Spectre関連のCPUフラグ](#)を参照してください。

---

従来、Proxmox VEは*kvm64* CPUモデルを採用し、Pentium 4レベルのCPUフラグが有効化されていたため、特定のワークロードではパフォーマンスが十分ではありませんでした。

2020年夏、AMD、Intel、Red Hat、SUSEは協力し、x86-64ベースラインの上に、最新のフラグを有効にした3つのx86-64マイクロアーキテクチャレベルを定義しました。詳細については、[x86-64-ABI仕様](#)を参照してください。

## 注記

CentOS 9などの新しいディストリビューションでは、最低要件としてx86-64-v2フラグが組み込まれています。

- **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 と比較して追加された CPU フラグ: +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 フラグが追加されました: +aes。
- **x86-64-v3**: Intel CPU>= Haswell、AMD CPU>= EPYC に対応。  
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 と比較して追加された CPU フラグ: +avx512f、+avx512bw、+avx512cd、+avx512dq、+avx512vl。

## カスタムCPUタイプ

設定可能な機能セットでカスタムCPUタイプを指定できます。これらはconfigura-で管理されます。

管理者による設定ファイル /etc/pve/virtual-guest/cpu-models.conf の編集。フォーマットの詳細については man cpu-models を参照してください。

フォーマットの詳細についてはman cpu-modelsを参照してください。

指定されたカスタムタイプは、/nodes 上で Sys.Audit 権限を持つユーザーが選択できます。CLI または API 経由で VM のカスタム CPU タイプを設定する場合、名前には custom- を接頭辞として付ける必要があります。

## Meltdown / Spectre 関連の CPU フラグ

Meltdown および Spectre の脆弱性に関連するいくつかの CPU フラグ<sup>3</sup>は、VM の選択した CPU タイプがデフォルトでそれらを有効にしていない限り、手動で設定する必要があります。

これらのCPUフラグを使用するには、以下の2つの要件を満たす必要があります：

- ホストCPUは当該機能をサポートし、ゲストの仮想CPUに伝播させる必要がある
- ゲストオペレーティングシステムは、攻撃を緩和しCPU機能を利用可能なバージョンに更新する必要があります

そうでない場合は、Web UIでCPUオプションを編集するか、VM設定ファイル内のcpuオプションのflagsプロパティを設定して、仮想CPUの希望するCPUフラグを設定する必要があります。

Spectre v1、v2、v4の修正については、CPUまたはシステムベンダーがCPU向けの「マイクロコード更新」を提供する必要があります（[ファームウェア更新の章](#)を参照）。影響を受けるすべてのCPUがspec-ctrlをサポートする更新を受けられるわけではない点に注意してください。

Proxmox VEホストの脆弱性を確認するには、root権限で以下のコマンドを実行してください：

```
for f in /sys/devices/system/cpu/vulnerabilities/* ; do echo "${f##* /} -" $(←
    cat "$f"); done
```

ホストが依然として脆弱であるかどうかを検出するためのコミュニティスクリプトも利用可能です。<sup>4</sup>

<sup>3</sup> Meltdown Attack <https://meltdownattack.com/>

<sup>4</sup> spectre-meltdown-checker <https://meltdown.ovh/>

## インテルプロセッサ

- *pcid*

これにより、カーネルページテーブル分離 (KPTI) と呼ばれるMeltdown (CVE-2017-5754) 対策のパフォーマンスへの影響が軽減されます。KPTIはカーネルメモリをユーザー空間から効果的に隠蔽します。PCIDなしでは、KPTIは非常にコストのかかるメカニズムとなります<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として次のコマンドを実行します:

```
# grep ' pcid ' /proc/cpuinfo
```

空の文字列が返されない場合、ホストのCPUは*PCID*をサポートしています。

- *spec-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。*retpolines*だけでは不十分な場合に適用されます。-IBRS サフィックスを持つ Intel CPU モデルではデフォルトで含まれています。-IBRS接尾辞を持たないIntel CPUモデルでは明示的に有効化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。

- *ssbd*

Spectre V4 (CVE-2018-3639) 対策の有効化に必須。Intel CPUモデルではデフォルトで含まれていません。全てのIntel CPUモデルで明示的に有効化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

- *ibpb*

*retpolines*では不十分な場合に、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。-IBPB サフィックスを持つ AMD CPU モデルにはデフォルトで含まれています。-IBPB サフィックスのない AMD CPU モデルでは明示的に有効にする必要があります。ゲスト CPU で使用するには、ホスト CPU 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。

- *virt-ssbd*

Spectre v4 (CVE-2018-3639) 対策の有効化に必須。デフォルトではどの AMD CPU モデルにも含まれていません。全ての AMD CPU モデルで明示的に有効化する必要があります。ゲスト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、ゲストに本機能を供給すべきです。物理CPUには存在しない仮想機能であるため、「host」CPUモデル使用時には明示的に有効化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) 対策の有効化に必須です。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モデルで明示的に有効にする必要があります。*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な限り常にゲストに公開すべきです。ただし、一部のカーネルは*virt-ssbd*しか認識しないため、ゲストの互換性を最大限に高めるために*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) の脆弱性に対して無防備である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は含まれていません。将来のCPUハードウェア世代はCVE-2018-3639の影響を受けないため、*amd-no-ssb*を公開することでゲストに緩和策を有効化しないよう指示すべきです。これは*virt-ssbd*および*amd-ssbd*と排他的です。

<sup>5</sup> PCIDは今やx86における重要な性能/セキュリティ機能です <https://groups.google.com/forum/m/#topic/mechanical-sympathy/L9mHTbeQLNU>

## NUMA

また、VM内でNUMA<sup>6</sup>アーキテクチャをエミュレートすることも可能です。NUMAの基本原理は、全コアが共有するグローバルメモリプールではなく、メモリが各ソケット付近のローカルバンクに分散配置される点にあります。これによりメモリバスがボトルネックとならなくなるため、速度向上が期待できます。システムがNUMAアーキテクチャ(<sup>7</sup>)を採用している場合、このオプションを有効にすることを推奨します。これにより、ホストシステム上でVMリソースが適切に分散されます。また、VM内でコアやRAMをホットプラグする場合にもこのオプションが必要です。

NUMAオプションを使用する場合、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します。

## vCPUのホットプラグ

現代のオペレーティングシステムは、稼働中のシステムにおいてCPUのホットプラグおよびある程度までのホットアンプラグ機能を導入しました。仮想化技術により、このようなシナリオで実ハードウェアが引き起こす可能性のある多くの（物理的な）問題を回避できます。とはいえ、これは比較的新しい複雑な機能であるため、その使用は絶対に必要な場合に限定すべきです。ほとんどの機能は、十分にテストされ複雑さの少ない他の機能で再現可能です（リソース制限を参照）。

Proxmox VEでは、接続可能なCPUの最大数は常にコア数×ソケット数です。この合計コア数未満のCPUでVMを起動するには、vcpus設定を使用します。これはVM起動時に接続すべきvCPUの数を示します。

現在この機能はLinuxでのみサポートされており、カーネル3.10以降が必要です。カーネル4.4以降が推奨されます。

4.7以降のカーネルが推奨されます。

ゲスト内で新しいCPUを自動的にオンライン状態に設定するには、以下のudevルールを使用できます：

```
SUBSYSTEM=="cpu", ACTION=="add", TEST=="online", ATTR{online}=="0", ATTR{online}=="1"
```

これを /etc/udev/rules.d/ 配下に .rules で終わるファイルとして保存します。

注記: CPUのホットリマップはマシン依存であり、ゲストOSの協力が必要です。削除コマンドはCPUの物理的リマップを保証するものではなく、通常はx86/amd64でのACPIなど、ターゲット依存のメカニズムを用いてゲストOSに要求を転送するものです。

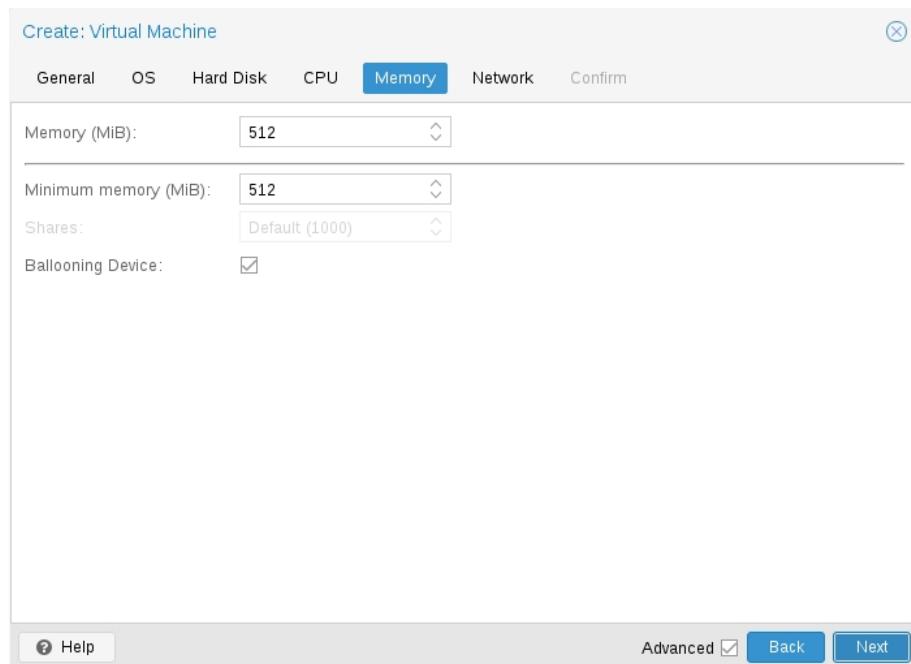
## 10.2.6 メモリ

各VMに対して、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Proxmox VEにメモリを動的に割り当てるかを選択できます。

<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 の結果が複数のノードを返す場合、ホストシステムはNUMAアーキテクチャを採用しています。

## 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した場合、Proxmox VEは指定した量をそのまま仮想マシンに割り当てます。

固定メモリサイズを使用する場合でも、バルーニングデバイスは仮想マシンに追加されます。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からです。一般的に**バルーニング**は有効のままにしておくべきですが、無効化したい場合（デバッグ目的など）は、単に**バルーニングデバイスのチェック**を外すか、

`balloon: 0`

を設定します。

## 自動メモリ割り当て

最小メモリをメモリ量より低く設定した場合、Proxmox VEは指定した最小量が常にVMに確保されるようにします。ホストのRAM使用率が特定の目標パーセンテージを下回ると、指定した最大メモリ量まで動的にゲストにメモリを追加します。目標パーセンテージはデフォルトで80%であり、[ノードオプション](#)で設定可能です。

ホストのRAMが不足すると、VMはメモリの一部をホストに解放します。必要に応じて実行中のプロセスをスワップし、最終手段としてoomキラーを起動します。ホストとゲスト間のメモリのやり取りは、ゲスト内で動作する特別なバルーニング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>

複数の仮想マシンが自動割り当て機能を使用する場合、各仮想マシンがホストの空きメモリから相対的に取得すべき量を示す「共有係数」を設定できます。例えば、4台の仮想マシンがあり、うち3台がHTTPサーバーを実行し、最後の1台がデータベースサーバーであるとします。ホストはRAM使用率を80%に設定されています。データベースサーバーのRAMにデータベースブロックをより多くキャッシュするため、空きRAMが利用可能な場合にデータベース仮想マシンを優先させたいとします。このため、データベースVMにシェア値3000を割り当て、他のVMはデフォルト値1000のままにします。ホストサーバーのRAM容量は32GBです。

<sup>8</sup> バルーンドライバの内部動作に関する優れた解説は[こちら](https://rwmj.wordpress.com/2010/07/17/-virtio-balloon/)で確認できます <https://rwmj.wordpress.com/2010/07/17/-virtio-balloon/>

RAM、現在16GBを使用中。これにより、VMに割り当て可能なRAMは $32 * 80/100 - 16 = 9\text{GB}$ となり、各VMの設定済み最小メモリ量に加えて利用可能。データベースVMは $9 * 3000 / (3000 + 1000 + 1000 + 1000) = 4.5\text{ GB}$ の追加RAMを、各HTTPサーバーは1.5 GBの追加RAMをそれぞれ利用できます。

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ドライバが組み込まれています。Windows OSではバルーンドライバを手動で追加する必要があり、ゲストの動作が遅くなる可能性があるため、重要なシステムでの使用は推奨しません。

仮想マシンにRAMを割り当てる際の目安として、常にホストに1GBのRAMを残しておくことを推奨します。

## 10.2.7 メモリ暗号化

### AMD SEV

SEV (Secure Encrypted Virtualization) は、AES-128暗号化とAMD Secure Processorを使用して、VMごとにメモリ暗号化を実現します。

SEV-ES (Secure Encrypted Virtualization - Encrypted State) はさらに、すべてのCPUレジスタの内容を暗号化し、ハイパーテーブルへの情報漏洩を防止します。

SEV-SNP (Secure Encrypted Virtualization - Secure Nested Paging) は、ソフトウェアベースの完全性攻撃の防止も試みます。詳細については、[AMD SEV SNPホワイトペーパー](#)を参照してください。

#### ホスト要件:

- AMD EPYC CPU
- SEV-ESはAMD EPYC 7002シリーズ以降のEPYC CPUでのみサポートされます
- SEV-SNPはAMD EPYC 7003シリーズ以降のEPYC CPUでのみサポートされます
- SEV-SNPはホストカーネルバージョン6.11以降が必要です。
- ホストマシンのBIOS設定でAMDメモリ暗号化を設定する
- デフォルトで有効になっていない場合、カーネルパラメータに「kvm\_amd.sev=1」を追加する
- ホスト上でメモリを暗号化したい場合（SME）、カーネルパラメータに「mem\_encrypt=on」を追加してください。詳細は<https://www.kernel.org/doc/Documentation/x86/amd-memory-encryption.txt>を参照
- 必要に応じてSWIOTLBを増やす 参照: <https://github.com/AMDESE/AMDSEV#faq-4>

ホストでSEVが有効か確認するには、dmesgでsevを検索し、kvm\_amdのSEVカーネルパラメータを出力:

```
# dmesg | grep -i sev
[...] ccp 0000:45:00.1: sev enabled
[...] ccp 0000:45:00.1: SEV API: <buildversion> [...] SEV
supported: <number> ASIDs
[...] SEV-ES サポート: <number> ASIDs # cat
/sys/module/kvm_amd/parameters/sev Y
```

#### ゲスト要件:

- edk2-OVMF
  - Q35の使用が推奨される
  - ゲストオペレーティングシステムはSEVサポートを含む必要があります。

### 制限事項:

- メモリが暗号化されているため、ホスト上のメモリ使用量は常に誤った値となります。
  - スナップショットやライブマイグレーションなど、メモリの保存や復元を伴う操作はまだ機能しないか、**攻撃を受けやすい**状態です。
  - PCIバスルートはサポートされていません。
  - SEV-ES および SEV-SNP は非常に実験的な段階です。
  - SEV-SNPではEFIディスクがサポートされていません。
  - SEV-SNPでは、VM内部のリブートコマンドは単にVMをシャットダウンします。

#### 設定例 (SEV):

```
# qm set <vmid> -amd-sev type=std,no-debug=1,no-key-sharing=1,kernel-hashes=1
```

**type** は暗号化技術を定義します（「`type=`」は不要）。利用可能なオプションは `std`、`es`、`snp` です。

QEMUのポリシーパラメータは、**no-debug**および**no-key-sharing**パラメータを用いて計算されます。これらのパラメータはポリシービット0および1に対応します。タイプが**es**の場合、ポリシービット2が1に設定され、SEV-ESが有効になります。ポリシービット3（nosend）は、マイグレーション攻撃を防ぐため常に1に設定されます。ポリシーの計算方法の詳細については、[AMD SEV API仕様書第3章](#)を参照してください。

カーネルハッシュオプションは、カーネル/initrdを測定しない古いOVMFイメージやゲストとの下位互換性のため、デフォルトで無効です。詳細は <https://lists.gnu.org/archive/html/qemu-devel/2021-11/msg02598.html> を参照してください

#### VM 内で SEV が機能しているか確認する

## 方法 1 - dmesq:

出力は以下のようになります。

```
# dmesg | grep -i sev  
AMDメモリ暗号化機能が有効: SEV
```

方法 2 - MSR 0xc0010131 (MSR AMD64 SEV):

出力は1であるべきです。

```
# apt install msr-tools #  
modprobe msr  
# rdmsr -a 0x00010131 1
```

#### 設定例 (SEV-SNP):

```
# gm set <:vmid:> -amd-sev type=snp,allow-smr=1,no-debug=1,kernel-hashes=1
```

allow-smt ポリシービットはデフォルトで設定されています。allow-smt を 0 に設定して無効化した場合、VM を実行するにはホスト上で SMT を無効化する必要があります。

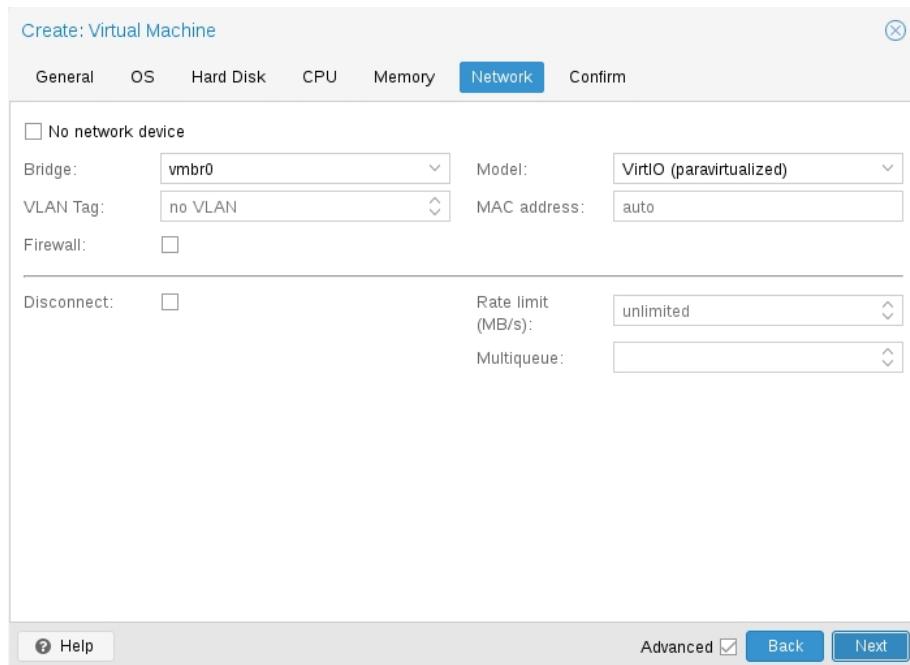
#### VM内でSEV-SNPが動作しているか確認

```
# dmesg | grep -i sns
メモリ暗号化機能の有効状態: AMD SEV SEV-ES SEV-SNPSEV: SNP CPUIDテーブルを使用中、29エントリ
存在。
SEV: SNP ゲストプラットフォームデバイスが初期化されました。
```

リンク:

- <https://developer.amd.com/sev/>
- <https://github.com/AMDESE/AMDSEV>
- <https://www.qemu.org/docs/master/system/i386/amd-memory-encryption.html>
- [https://www.amd.com/system/files/TechDocs/55766\\_SEV-KM\\_API\\_Specification.pdf](https://www.amd.com/system/files/TechDocs/55766_SEV-KM_API_Specification.pdf)
- <https://documentation.suse.com/sles/15-SP1/html/SLES-amd-sev/index.html>
- SEVセキュアネステッドペイジングファームウェアABI仕様

### 10.2.8 ネットワークデバイス



各仮想マシンは、4種類の異なるネットワークインターフェースコントローラ (NIC) を複数搭載可能です：

- Intel E1000 はデフォルトであり、Intel ギガビットネットワークカードをエミュレートします。
- 最高のパフォーマンスを求める場合は、VirtIO パラ仮想化 NIC を使用してください。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 100 MB/s ネットワークカードをエミュレートし、2002 年以前にリリースされた古いオペレーティングシステムをエミュレートする場合にのみ使用すべきです。

- **vmxnet3** は別の準仮想化デバイスであり、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べきです。

Proxmox VEは各NICに対してランダムな**MACアドレス**を生成するため、VMはイーサネットネットワーク上でアドレス指定可能となります。

仮想マシンに追加したNICは、以下の2つの異なるモデルのいずれかに従います：

- デフォルトの**ブリッジモード**では、各仮想NICはホスト上で**tap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）によって裏付けられます。このtapデバイスはブリッジ（Proxmox VEではデフォルトでvmbr0）に追加されます。このモードでは、VMはホストが存在するイーサネット LANに直接アクセスできます。
- 代替となる**NATモード**では、各仮想NICはQEMUユーザーネットワークスタックとのみ通信します。このスタックには組み込みルーターとDHCPサーバーが実装されており、ネットワークアクセスを提供します。この組み込みDHCPはプライベート範囲10.0.2.0/24のアドレスを割り当てます。NATモードはブリッジモードより大幅に低速であるため、テスト目的でのみ使用すべきです。このモードはCLIまたはAPI経由でのみ利用可能で、Web UIからは利用できません。

仮想マシン作成時に「ネットワークデバイスなし」を選択することで、ネットワークデバイスの追加をスキップすることも可能です。

各VMネットワークデバイスのMTU設定は手動で上書き可能です。フィールドを空欄のままにするかmtu=1を設定すると、基盤となるブリッジからMTUを継承します。このオプションは**VirtIO**ネットワークデバイスでのみ利用可能です。

## マルチキュー

VirtIOドライバを使用している場合、オプションでマルチキュー機能を有効化できます。この機能により、ゲストOSは複数の仮想CPUを使用してネットワーク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のネットワークキューはホストカーネルに渡され、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可能になります。

マルチキューを使用する際は、ゲストのvCPU数と同等の値に設定することを推奨します。vCPU数はソケット数にVMで設定されたコア数を乗じた値であることを留意してください。また、VM内の各VirtIO NICに対して、以下のethtoolコマンドでマルチバーパスチャネル数を設定する必要があります：

```
ethtool -L ens1 combined X
```

ここで X は VM の vCPU 数です。

Windowsゲストをマルチキュー用に設定するには、[Redhat VirtIO Ethernet Adapterドライバ](#)をインストールし、NICの設定を以下のように変更します。デバイスマネージャを開き、「ネットワークアダプタ」下のNICを右クリックして「プロパティ」を選択します。「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、リスト内の「RSSキューの最大数」に移動し、VMのvCPU数に設定します。設定が正しいことを確認したら、「OK」をクリックして確定します。

マルチキューパラメータを1より大きい値に設定すると、トラフィックの増加に伴いホストおよびゲストシステムの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、リバースプロキシ、またはロングポーリングを行う多忙なHTTPサーバーとして動作するなど、大量の着信接続を処理する必要がある場合にのみ設定することを推奨します。

## 10.2.9 表示

QEMUはいくつかの種類のVGAハードウェアを仮想化できます。例としては以下があります：

- **std** (デフォルト) は、Bochs VBE 拡張機能を備えたカードをエミュレートします。
- **cirrus**: かつてデフォルトだった設定。非常に古いハードウェアモジュールをエミュレートし、その問題点を全て再現します。この表示タイプは、Windows XP以前を使用する場合など、本当に必要な場合<sup>(9)fn</sup>のみ使用すべきです。
- **vmware**: VMWare SVGA-II互換アダプタです。
- **qxl**: 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向けにSPICE（リモートビューアプロトコル）も有効になります。
- **virtio-g1** (VirGLとも呼ばれる) は、VM内で使用するための仮想3D GPUです。特別な（高価な）モデルやドライバを必要とせず、ホストGPUを完全に占有することもなく、ワークロードをホストGPUにオフロードできるため、複数のゲスト間やホストでの再利用が可能です。

### 注意

VirGLサポートには追加ライブラリが必要です。これらはサイズが大きく、全てのGPUモデル/ベンダー向けにオープンソース化されていないため、デフォルトではインストールされません。ほとんどの環境では以下を実行するだけで十分です：`apt install libgl1 libegl1`

仮想GPUに割り当てるメモリ量は、`memory`オプションを設定することで編集可能です。これにより、特にSPICE/QXL環境において、VM内の高解像度化が可能になります。

メモリは表示デバイスごとに予約されるため、SPICEでマルチモニターモード（デュアルモニター用の`qxl2`など）を選択すると以下の影響が生じます：

- Windowsはモニタごとにデバイスを必要とするため、`ostype`がWindowsの何らかのバージョンである場合、Proxmox VEはVMにモニタごとに追加デバイスを割り当てます。各デバイスには指定された量のメモリが割り当てられます。
- Linux VMでは仮想モニターを常に追加できますが、マルチモニターモードを選択すると、デバイスに割り当てられるメモリがモニタ数に応じて増加します。

表示タイプとして`serialX`を選択すると、VGA出力が無効化され、Webコンソールが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、設定済みのディスプレイメモリ設定は無視されます。

### VNCクリップボード

```
# qm set <vmid> -vga <displaytype>,clipboard=vnc  
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できます。
```

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。その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-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すると、VNCクリップボード機能（例：noVNCコンソールパネル内）が利用可能になります。ただし、SPICE、virtio、またはvirglを使用している場合、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、クリップボードが`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。

<sup>9</sup> <https://www.kraxel.org/blog/2014/10/qemu-using-cirrus-considered-harmful/> qemu: cirrusの使用は有害とみなされる

**注意**

VNCクリップボードを有効化するため、QEMUはqemu-vdagentデバイスを使用するように設定されています。現在、qemu-vdagentデバイスはライブマイグレーションをサポートしていません。これは、VNCクリップボードが有効化されたVMは現時点でライブマイグレーションできないことを意味します。

## 10.2.10 USBバススルーデバイス

USBバススルーデバイスには2種類あります：

- ホストUSBバススルーデバイス
- SPICE USBバススルーデバイス

ホストUSBバススルーデバイスは、ホストのUSBデバイスを仮想マシンに割り当てることで機能します。これはベンダーIDとプロダクトID、またはホストバスとポートを介して行うことができます。

ベンダー/プロダクトIDは次のような形式です：**0123:abcd**。ここで**0123**はベンダーID、**abcd**はプロダクトIDを表し、同じUSBデバイスの2つの部品は同一のIDを持ちます。

バス/ポートは次のように表記されます：**1-2.3.4**。ここで**1**はバス、**2.3.4**はポートパスを表します。これはホストの物理ポート（USBコントローラの内部順序に依存）を示します。

仮想マシン起動時にVM構成内にデバイスが存在する場合、そのデバイスがホストに存在しなくてもVMは問題なく起動します。ホストでデバイス/ポートが利用可能になるとすぐに、それがバススルーデバイスになります。

**警告**

この種のUSBバススルーデバイスを使用すると、ハードウェアがVMが現在存在するホストでのみ利用可能であるため、オンライン状態でVMを別のホストに移動することはできません。

2つ目のバススルーデバイスはSPICE USBバススルーデバイスです。VMに1つ以上のSPICE USBポートを追加すると、SPICEクライアントからローカルUSBデバイスを動的にVMへバススルーデバイスになります。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有用です。

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す。これにより、HA環境での適切な使用、ハードウェア変更の検出、非rootユーザーによる設定が可能になります。詳細は[リソースマッピング](#)を参照してください。

## 10.2.11 BIOSとUEFI

コンピュータを適切にエミュレートするには、QEMUはファームウェアを使用する必要があります。一般的なPCではBIOSまたは(U)EFIとして知られるこのファームウェアは、VM起動時の最初のステップとして実行されます。基本的なハードウェア初期化を実行し、オペレーティングシステム向けにファームウェアおよび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います。デフォルトでは、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設定において適切な選択肢です。

一部のオペレーティングシステム（Windows 11など）では、UEFI互換の実装が必要となる場合があります。そのような場合は、代わりにオープンソースのUEFI実装であるOVMFを使用する必要があります。<sup>10</sup>

<sup>10</sup> OVMFプロジェクトを参照 <https://github.com/tianocore/tianocore.github.io/wiki/OVMF>

SeaBIOSが起動用ファームウェアとして最適でない他のシナリオも存在します。例えばVGA/パススルーを実行したい場合などです。<sup>11</sup>

OVMFを使用する場合、考慮すべき点がいくつかあります：

ブート順序などの設定を保存するには、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台のみ存在できます。

以下のコマンドで作成できます：

```
# qm set <vmid> -efidisk0 <storage>:1,format=<format>,efitype=4m,pre-enrolled-keys=1
```

ここで **<storage>** はディスクを配置するストレージ、**<format>** はストレージがサポートするフォーマットです。あるいは、VMのハードウェアセクションで 「→ EFIディスクの追加」を選択し、Webインターフェースからディスクを作成することもできます。

**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します。新規VMでは常に4mを選択してください。これはSecure Bootをサポートし、将来的開発を見据えた追加領域を確保しているためです（GUIではこれがデフォルト設定）。

**pre-enroll-keys** は、efidisk にディストリビューション固有および Microsoft 標準のセキュアブートキーを事前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 メニューで無効化することは可能です）。

---

#### 注記

既存のVM（2MB efidiskを使用中）でSecure Bootを有効化するには、efidiskを再作成する必要があります。具体的には、古いefidiskを削除（`qm set <vmid> -delete efidisk0`）し、上記の手順で新規追加してください。これによりOVMFメニューで設定したカスタム構成はリセットされます！

---

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（起動中にESCボタンを押すことでアクセス可能）でクライアント解像度を設定するか、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。

## 10.2.12 トラステッド・プラットフォーム・モジュール (TPM)

信頼できるプラットフォームモジュール (TPM) は、暗号化キーなどの機密データを安全に保存し、システム起動の検証に耐改ざん機能を提供するデバイスです。

特定のオペレーティングシステム（Windows 11など）では、物理マシンまたは仮想マシンにこのデバイスを接続する必要があります。

TPMはtpmstateボリュームを指定することで追加されます。これはefidiskと同様に、一度作成すると変更（削除のみ可能）できません。以下のコマンドで追加できます：

```
# qm set <vmid> -tpmstate0 <storage>:1,version=<version>;
```

**<storage>** は状態を保存するストレージ、**<version>** は v1.2 または v2.0 を指定します。また、VM のハードウェアセクションで 「→ TPM 状態の追加」を選択することで、Web インターフェース経由での追加も可能です。

v2.0 TPM仕様はより新しくサポートも優れているため、v1.2 TPMを必要とする特定の実装がない限り、優先的に選択すべきです。

<sup>11</sup> Alex Williamsonがこの件について優れたブログ記事を書いています <https://vfio.blogspot.co.at/2014/08/primary-graphics-assignment-without-vga.html>

---

#### 注記

物理TPMと比較して、エミュレートされたTPMは実質的なセキュリティ上の利点を提供しません。TPMの本質は、TPM仕様で定義されたコマンド以外ではデータを容易に変更できない点にあります。エミュレートデバイスではデータ保存が通常のボリューム上で行われるため、アクセス権を持つ者なら誰でも編集する可能性があります。

---

## 10.2.13 仮想マシン間共有メモリ

ホストとゲスト間、または複数のゲスト間でメモリを共有できる仮想マシン間共有メモリデバイス (`ivshmem`) を追加できます。

このデバイスを追加するには、`qm` コマンドを使用します:

```
# qm set <vmid> -ivshmem size=32,name=foo
```

サイズは MiB 単位です。ファイルは `/dev/shm/pve-shm-$name` (デフォルト名は `vmid`) に配置されます。

---

#### 注記

現在、このデバイスは、それを使用している仮想マシンがシャットダウンまたは停止されるとすぐに削除されます。開いている接続は維持されますが、まったく同じデバイスへの新規接続はできなくなります。

---

この種のデバイスの使用例としては、ホストとゲスト間の高性能、低遅延のディスプレイミラーリングを可能にする Looking Glass<sup>12</sup> プロジェクトがあります。

## 10.2.14 オーディオデバイス

オーディオデバイスを追加するには、次のコマンドを実行します:

```
qm set <vmid> -audio0 device=<device>;
```

サポートされているオーディオデバイスは以下の通りです:

- `ich9-intel-hda`: Intel HD Audio Controller、ICH9をエミュレート
- `intel-hda`: Intel HD Audio Controller、ICH6をエミュレート
- AC97: オーディオコーデック '97、Windows XPなどの古いOSに有用利用可能なバックエンドは2つあります:
- `spice`
- `none`

`spice` バックエンドは [SPICE](#) と組み合わせて使用でき、`none` バックエンドは仮想マシン内で特定のソフトウェア動作に必要なオーディオデバイスを提供する際に有用です。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パスルーを利用して下さい（[PCI](#)パスルーおよび[USB](#)パスルーを参照）。Microsoft RDPなどのリモートプロトコルには音声再生オプションがあります。

---

<sup>12</sup> Looking Glass: <https://looking-glass.io/>

## 10.2.15 VirtIO RNG

RNG（乱数生成器）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す。仮想ハードウェアRNGを使用すると、ホストシステムからゲストVMにエントロピーを供給できます。これにより、特にゲストの起動プロセス中に発生するエントロピー不足の問題（十分なエントロピーが得られず、システムの速度低下や問題発生を引き起こす状況）を回避できます。

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=<source>[,max_bytes=X,period=Y]
```

`source` はホスト上でエントロピーを読み込む場所を指定し、以下のいずれかでなければならない:

- `/dev/urandom`: 非ブロッキングカーネルエントロピープール（推奨）
- `/dev/random`: ブロッキングカーネルプール（推奨されません。ホストシステムでエントロピー枯渇を引き起こす可能性があります）
- `/dev/hwrng`: ホストに接続されたハードウェア乱数生成器（RNG）を透過的に使用（複数存在する場合、`/sys/devices/virtual/misc/hw_random/rng_current` で選択されたものが使用される）

`max_bytes` および `period` パラメータで制限を指定可能。これらは「期間（ミリ秒）あたりの最大バイト数」として解釈される。ただし線形関係ではない：`1024B/1000ms` は「1秒間隔で最大1KiBのデータが利用可能」を意味し、「1秒間に1KiBがゲストにストリーミングされる」ことを意味しない。したがって、期間を短縮することでゲストへのエントロピー注入速度を向上させられます。

デフォルトでは、制限は1000ミリ秒あたり1024バイト（1KiB/秒）に設定されています。ゲストがホストリソースを過剰に使用するのを防ぐため、常にリミッターを使用することを推奨します。必要に応じて、`max_bytes` に0を指定することで全ての制限を無効化できます。

## 10.2.16 Virtiofs

Virtiofsは仮想環境向けに設計された共有ファイルシステムです。ホスト上で利用可能なディレクトリツリーを仮想マシン内にマウントすることで共有を可能にします。ネットワークスタックを使用せず、ソースファイルシステムと同等のパフォーマンスとセマンティクスを提供することを目的としています。

`virtiofs` を使用するには、バックグラウンドで `virtiofsd` デーモンを実行する必要があります。Proxmox VE では、`virtiofs` マウントを使用して VM を起動すると自動的に実行されます。

カーネル 5.4 以降を搭載した Linux VM はデフォルトで `virtiofs` をサポートします（[virtiofs カーネルモジュール](#)）。ただし、一部の機能にはより新しいカーネルが必要です。

`virtiofs` を使用するには、Proxmox VE ホストに `virtiofsd` がインストールされていることを確認してください:

```
apt install virtiofsd
```

Windows VM で `virtiofs` を利用する方法については、ガイドが用意されています。

## 既知の制限事項

- virtiofsdがクラッシュした場合、そのマウントポイントはVMが完全に停止するまでVM内でハンギングした状態になります。
- virtiofsdが応答しない場合、到達不能な NFS と同様に、VM 内でマウントがハンギングする可能性があります。
- virtiofs とメモリホットプラグを併用すると機能せず（アクセスがハンギングする結果となります）。
- ライブマイグレーション、スナップショット、ハイバネーションなどのメモリ関連機能は、virtiofs デバイスでは利用できません。
- Windows は virtiofs のコンテキストでは ACL を理解できません。したがって、Windows VM に対して ACL を公開しないでください。公開すると、VM 内で virtiofs デバイスが表示されなくなります。

## 共有ディレクトリのマッピングを追加

共有ディレクトリのマッピングを追加するには、リソースマッピングセクションに記載されているように、`pvesh` で API を直接使用できます：

```
pvesh create /cluster/mapping/dir --id dir1 \
--map node=node1,path=/path/to/share1 \
--map node=node1,path=/path/to/share1 \
--map node=node2,path=/path/to/share2 \
```

## 仮想マシンにvirtiofsを追加する

virtiofsを使用してディレクトリを共有するには、VM設定にパラメータ `virtiofs< N>` (`N`は0から9までの任意の数字) を追加し、リソースマッピングで設定済みのディレクトリID (dirid) を使用します。さらに、要件に応じてキャッシュオプションをalways、never、metadata、またはauto（デフォルト: auto）のいずれかに設定できます。各キャッシュモードの動作については、[こちらの「Caching Modes」セクションを参照](#)してください。

`virtiofsd`はACLとxattrパスルーをサポートします（`expose-acl`および`expose-xattr`オプションで有効化可能）。これにより、ゲストは基盤となるホストファイルシステムがACLとxattrをサポートしている場合にそれらにアクセスできますが、ゲストファイルシステムとの互換性も必要（オプションで有効化可能）をサポートし、基盤となるホストファイルシステムが対応している場合、ゲストがACLや拡張属性にアクセスできるようにします。ただし、これらはゲストファイルシステムとも互換性がある必要があります（例：ほとんどのLinuxファイルシステムはACLをサポートしますが、Windowsファイルシステムはサポートしません）。

`expose-acl` オプションは自動的に `expose-xattr` を意味します。つまり、`expose-acl` が 1 に設定されている場合、`expose-xattr` を 0 に設定しても違いはありません。

virtiofsにO\_DIRECTフラグを尊重させたい場合、`direct-io`パラメータを1に設定できます（デフォルト: 0）。これによりパフォーマンスは低下しますが、アプリケーションが独自のキャッシュを行う場合に有用です。

```
qm set < vmid > -virtiofs0 dirid=< dirid >,cache=always,direct-io=1 qm set
&lt; vmid &gt; -virtiofs1 < dirid >,cache=never,expose-xattr=1
qm set < vmid &gt; -virtiofs2 < dirid >,expose-acl=1
```

ゲストVM内でvirtiofsを一時的にマウントするには、ゲスト内で以下のコマンドを実行します（Linuxカーネルvirtiofsドライバを使用）：

```
mount -t virtiofs < dirid > < マウントポイント >
```

永続的なvirtiofsマウントを実現するには、fstabエントリを作成します：

```
< dirid > < マウントポイント > virtiofs rw,relatime 0 0
```

現在のノード上のパスに関連付けられたdiridは、マウントタグ（ゲスト上でデバイスをマウントする際に使用される名前）としても使用されます。

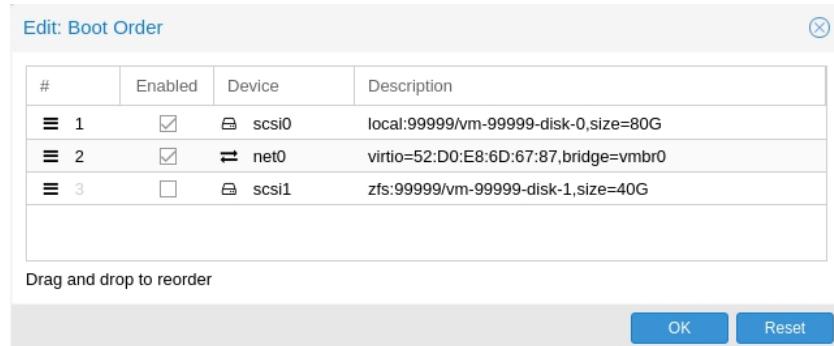
利用可能なvirtiofsdパラメータの詳細については、[GitLabのvirtiofsdプロジェクトページ](#)を参照してください。

## 10.2.17 デバイスの起動順序

QEMUはゲストに対して、どのデバイスから起動すべきか、またその順序を指示できます。これは設定ファイルのbootプロパティで指定可能です。例：

```
boot: order=scsi0;net0;hostpci0
```

この設定により、ゲストはまずディスク scsi0 から起動を試みます。これが失敗した場合、次に net0 からのネットワーク起動を試み、それでも失敗した場合



は最終的にバススルーされた PCIe デバイス (NVMe の場合はディスクとして認識され、それ以外の場合はオプション ROM への起動を試みます) からの起動を試みます。

GUIではドラッグ & ドロップエディタで起動順序を指定でき、チェックボックスで特定のデバイスの起動を有効／無効にできます。

### 注記

ゲストOSが複数のディスクを使用してOSを起動またはブートローダーをロードする場合、ゲストが起動できるようにするには、それらのディスク全てが起動可能としてマークされている必要があります (つまり、チェックボックスが有効になっているか、設定リストに表示されている必要があります)。これは、最近のSeaBIOSおよびOVMFバージョンでは、起動可能としてマークされたディスクのみが初期化されるためです。

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なデバイスであっても、ゲストのオペレーティングシステムが起動してそれらを初期化すれば、ゲストから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。

## 10.2.18 仮想マシンの自動起動とシャットダウン

仮想マシンを作成したら、ホストシステムの起動時に自動的に起動させたい場合が多いでしょう。これには、Webインターフェースで仮想マシンの[オプション]タブから[起動時に開始]オプションを選択するか、以下のコマンドで設定します:

```
# qm set <vmid> -onboot 1
```

## 起動順序とシャットダウン順序



場合によっては、VMの起動順序を微調整したいことがあります。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です。このためには、以下のパラメータを使用できます：

- **起動/シャットダウン順序:** 起動順序の優先度を定義します。例えば、VMを最初に起動させたい場合は1に設定します（シャットダウン時は逆の起動順序を使用するため、起動順序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に同じ順序が定義されている場合、VMIDの昇順で追加の順序付けが行われます。
- **起動遅延:** このVMの起動と後続VMの起動との間隔を定義します。例えば、他のVMの起動を240秒待機したい場合は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発行後、VMがオフラインになるまで待機する秒数を定義します。デフォルト値は180秒です。つまり、Proxmox VEは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秒待機します。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。

### 注記

HAスタックで管理されるVMは、現在「起動時の起動」および「起動順序」オプションに従いません。これらのVMは、HAマネージャー自体がVMの起動と停止を保証するため、起動/シャットダウンアルゴリズムによってスキップされます。

起動/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、パラメータが設定されているマシンよりも常に後で起動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強制でき、クラスター全体では適用されません。

ホスト起動と最初のVM起動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。

## 10.2.19 QEMUゲストエージェント

QEMUゲストエージェントは、仮想マシン（VM）内で動作するサービスであり、ホストとゲスト間の通信チャネルを提供します。情報の交換に使用され、ホストがゲストに対してコマンドを発行することを可能にします。

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。

バックアップを開始する際には、`fs-freeze`

および`fs-thaw`コマンドを介して未処理の書き込みを同期するようゲストに通知します。

ゲストエージェントを正常に動作させるには、以下の手順が必要です：

- ゲストにエージェントをインストールし、動作し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可能です。パッケージ名は通常qemu-guest-agentです。Windowsの場合は、[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通信の有効化

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化できます。変更を有効にするにはVMの再起動が必要です。

## QGAを使用した自動TRIM

「ゲストTRIMの実行」オプションを有効化できます。これを有効にすると、Proxmox VEは以下の操作後にゲストに対してTRIMコマンドを発行します（これらの操作はストレージへのゼロ書き込みを引き起こす可能性があります）：

- ディスクの別のストレージへの移動
- 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

シンブルピジョンングされたストレージでは、これにより未使用領域を解放できます。

### 注記

Linuxのext4には注意点があります。これは重複したTRIM要求を発行しないためのメモリ内最適化を使用しているためです。ゲストは基盤となるストレージの変更を認識しないため、最初のゲストTRIMのみが期待通りに実行されます。次の再起動までの後続の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凍結と解除

デフォルトでは、バックアップ実行時にfs-freeze QEMUゲストエージェントコマンドを介してゲストファイルシステムが同期され、一貫性が確保されます。

Windowsゲストでは、一部のアプリケーションがWindows VSS（ボリュームシャドウコピーサービス）レイヤーにフックすることで一貫性のあるバックアップを独自に処理する場合があり、fs-freezeの実行がこれに干渉する可能性があります。例えば、特定のSQL Serverでfs-freezeを呼び出すと、VSSがSQL Writer VSSモジュールを差分バックアップ用のSQL Server/バックアップチェーンを破壊するモードで呼び出されています。

このような状況に対処するには、2つの選択肢があります。

1. QEMUゲストエージェントを、他のVSSユーザーと競合しない別のVSSバリエントを使用するように設定します。詳細は[Proxmox VE Wiki](#)を参照してください。
2. あるいは、バックアップ時に凍結・解凍サイクルを発行しないようProxmox VEを設定する方法があります。これにはQGAオプション`freeze-fs-on-backup`を0に設定します。GUIから「ノックアップ時の一貫性確保のためゲストファイルシステムを凍結/解凍する」オプションで同様の設定が可能です。



### 重要

このオプションを無効化すると、ファイルシステムの一貫性が損なわれたバックアップが生成される可能性があります。そのため、ゲスト内のQEMUゲストエージェント設定を変更する方法が推奨されます。

## トラブルシューティング

### 仮想マシン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化されると、Proxmox VEはシャットダウンなどの電源コマンドをゲストエージェント経由で送信します。ゲストエージェントが動作していない場合、コマンドは正常に実行されず、シャットダウンコマンドはタイムアウトします。

## 10.2.20 SPICE 拡張機能

SPICE 機能強化は、リモートビューアの操作性を向上させるオプション機能です。

GUIで有効化するには、仮想マシンのオプションパネルに移動してください。CLIで有効化するには、次のコマンドを実行します：

```
qm set <vmid> -spice_enhancements folderssharing=1,videostreaming=all
```

### 注

これらの機能を利用するには、仮想マシンの[表示モード](#)をSPICE (qxl)に設定する必要があります。

### フォルダ共有

ゲストにローカルフォルダを共有します。ゲストにspice-webdavdデーモンをインストールする必要があります。これにより、共有フォルダが<http://localhost:9843>にあるローカルWebDAVサーバー経由で利用可能になります。

Windowsゲストの場合、Spice WebDAVデーモンのインストーラーは[公式SPICE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spice-webdavdパッケージが存在し、インストール可能です。

Virt-Viewer（リモートビューア）でフォルダを共有するには、[File] → [→] → [Preferences] に移動します。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。

### 注意

フォルダ共有は現在、Linux版Virt-Viewerでのみ動作します。



### 注意

実験的機能！現在、この機能は安定して動作しません。

### ビデオストリーミング

高速リフレッシュ領域は動画ストリームにエンコードされます。以下の2つのオプションが存在します：

- **all:** すべての高速更新領域をビデオストリームにエンコードします。

- **filter:** ビデオストリーミングの使用可否を判断する追加フィルターが適用されます（現在は小さなウィンドウサーフェス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るべきか、どのオプションを選択すべきかについての一般的な推奨事項は提供できません。具体的な状況によって結果が異なる場合があります。

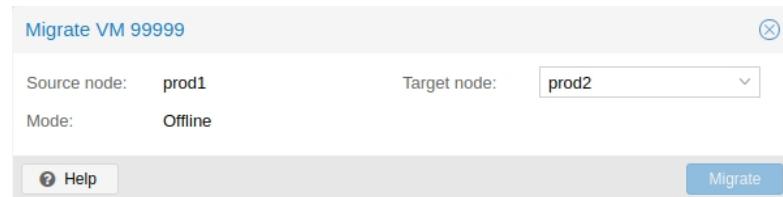
## トラブルシューティング

### 共有フォルダが表示されない

ゲスト側で WebDAV サービスが有効化され実行されていることを確認してください。Windows では Spice webdav proxy、Linux では spice-webdavd と呼ばれますが、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。

サービスが実行中の場合、ゲストのブラウザで <http://localhost:9843> を開き WebDAV サーバーを確認してください。SPICE セッションの再起動が有効な場合があります。

## 10.3 移行



クラスタをお持ちの場合、VMを別のホストに移行できます。

```
# qm migrate <vmid> <target>
```

これには一般的に2つのメカニズムがあります

- オンライン移行（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

仮想マシンが稼働中で、ローカルにバインドされたリソース（バススルーデバイスなど）が設定されていない場合、qm migration コマンドで --online フラグを指定してライブマイグレーションを開始できます。Web インターフェースでは、仮想マシンが稼働中の場合、デフォルトでライブマイグレーションが選択されます。

#### 動作原理

オンライン移行ではまず、ターゲットホスト上で *incoming* フラグ付きの新規QEMUプロセスを開始します。このプロセスはゲストvCPUを一時停止したまま基本初期化のみを実行し、その後ソース仮想マシンのゲストメモリおよびデバイス状態データストリームを待機します。ディスクなどのその他リソースは、VMの実行時状態移行開始前に共有済みか既に送信済みであるため、転送対象はメモリ内容とデバイス状態のみとなります。

接続が確立されると、ソースは非同期的にメモリ内容をターゲットへ送信を開始します。ソース側のゲストメモリが変更されると、該当セクションは変更済み（dirty）としてマークされ、再度ゲストメモリデータを送信する処理が行われます。このループは、実行中のソースVMと受信側ターゲットVM間のデータ差分が数ミリ秒で送信可能な程度に小さくなるまで繰り返されます。この状態になれば、ユーザーやプログラムが停止を感じることなくソースVMを完全に一時停止でき、残りのデータをターゲットに送信した後、ターゲットVMのCPUを1秒未満で再開して新たな実行VMとすることができます。

## 要件

ライブマイグレーションを機能させるには、以下の要件を満たす必要があります：

- VMに移行不可能なローカルリソースが存在しないこと。例えば、PCIデバイスやバススルーUSBデバイスは現在ライブマイグレーションを妨げます。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可能です。
- ホストが同一のProxmox VEクラスター内に存在すること。
- ホスト間には正常に機能する（かつ信頼性の高い）ネットワーク接続が確立されています。
- ターゲットホストは、Proxmox VEパッケージの同一バージョンまたはそれ以上のバージョンを必須とします。逆の構成でも動作する場合がありますが、保証はできません。
- ホストは、同じベンダーのCPUで、同等の性能を備えている必要があります。異なるベンダーのCPUでも、実際のモデルや設定されたVMのCPUタイプによっては動作する可能性がありますが、保証はできません。そのため、本番環境でこのような設定を展開する前にテストを行ってください。

## 接続状態の移行

### 注意

接続追跡状態の移行はベストエフォート方式であり、ネットワーク設定に大きく依存するため動作しない可能性があります。例えば、ホストで[送信元アドレス偽装（SNAT）](#)を使用している構成では、ほぼ確実に動作しません。

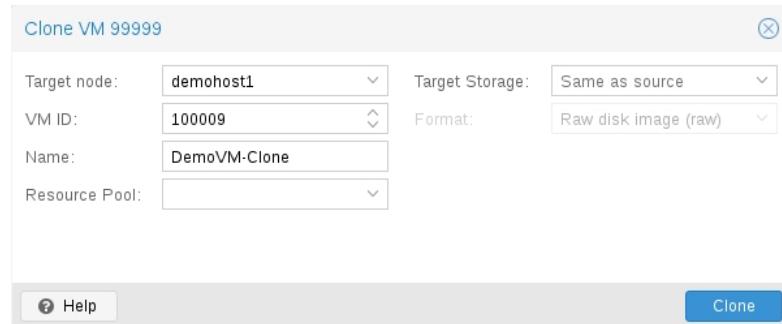
接続追跡（Conntrack）は、個々の接続を追跡することでステートフルファイアウォールを実現するLinuxカーネルの仕組みです。稼働中のVMをライブマイグレーションする場合、アクティブな着信/発信接続は、VMがターゲットホストで実行を開始するとすぐに中断される可能性があります。これは、新しいホストノードが同じ接続追跡エントリを持たないため、ファイアウォールがパケットをドロップする可能性があるためです。

接続追跡状態移行では、ライブ移行対象のVMに関するホスト上の全接続追跡エントリをターゲットノードにコピーした後、ソースノードの接続追跡テーブルから移行済みエントリをフラッシュします。

## 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できます。移行時には、オンライン移行と同様に、ディスクがネットワーク経由でターゲットホストにコピーされます。ハードウェアバススルー設定は、ターゲットホスト上のデバイス位置に合わせて調整が必要になる可能性がある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ベンダー提供のインストールメディア（CD-ROM）を使用して行われます。OSによっては時間がかかるため、この作業を避けたい場合もあるでしょう。

同じタイプの仮想マシンを多数展開する簡単な方法は、既存の仮想マシンをコピーすることです。このようなコピーをクローンと呼び、リンク済みクローンと完全クローンを区別します。

### 完全クローン

この複製の結果は独立した仮想マシンです。新しい仮想マシンは元の仮想マシンとストレージリソースを一切共有しません。

ターゲットストレージを選択できるため、VMを全く異なるストレージへ移行する際に利用可能です。また、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ます。

#### 注

完全クローンでは、すべてのVMイメージデータを読み取りコピーする必要があります。通常、リンククローンを作成するよりも大幅に時間がかかります。

一部のストレージタイプでは特定のスナップショットをコピーでき、デフォルトでは現在のVMデータが対象となります。これは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一切含まれないことを意味します。

### リンクド クローン

最新のストレージドライバは高速なリンクドクローンを生成する方式をサポートしています。この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一です。リンクドクローンの作成はほぼ瞬時に行われ、初期段階では追加のスペースを消費しません。

これらがリンクされていると呼ばれるのは、新しいイメージが依然として元のイメージを参照しているためです。変更されていないデータブロックは元のイメージ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（その後読み込まれます）。この技術はコピー・オン・ライトと呼ばれます。

この手法では元のボリュームが読み取り専用である必要があります。Proxmox VEでは任意のVMを読み取り専用テンプレートに変換できます。こうしたテンプレートは後でリンクドクローンを効率的に作成するために使用できます。

#### 注意

リンククローンが存在する間は、元のテンプレートを削除できません。

リンククローンはストレージ内部の機能であるため、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、別のノード上に新しい VM を作成することができます。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。

リソース競合を回避するため、すべてのネットワークインターフェースの 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

仮想マシンをテンプレートに変換できます。このようなテンプレートは読み取り専用であり、リンクドクローンを作成するために使用できます。

### 注意

テンプレートを起動することはできません。これはディスクイメージを変更するためです。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 世代 ID

Proxmox VE は仮想マシンの仮想マシン世代 ID (*vmgenid*)<sup>13</sup> をサポートしています。これはゲスト OS がバックアップの復元やスナップショットのロールバックなど、タイムシフトを引き起こす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、その設定ファイルに保存されます。

既存の VM に *vmgenid* を作成・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、*UUID*<sup>14</sup> を値として手動で設定します。例：

```
# qm set VMID -vmgenid 1
# qm set VMID -vmgenid 00000000-0000-0000-0000-000000000000
```

### 注記

既存の VM に *vmgenid* デバイスを初めて追加すると、スナップショットのロールバックやバックアップの復元などと同様の影響が生じる可能性があります。VM がこれを世代変更と解釈する可能性があるためです。

稀なケースとして、*vmgenid* メカニズムが不要な場合、VM 作成時にその値として「0」を指定するか、設定ファイルで後からプロパティを削除できます：

```
# qm set VMID -delete vmgenid
```

*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す。これらは、スナップショットのロールバック、バックアップの復元、または VM 全体のクローン操作時に、時間依存性のあるサービスや複製サービス（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使用します。

<sup>13</sup> 公式 *vmgenid* 仕様書 [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 生成ツール <http://guid.one/>

<sup>15</sup> <https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtualized-domain-controller-アーキテクチャ>

## 10.7 仮想マシンのインポート

外部ハイパーバイザーや他のProxmox VEクラスターから既存の仮想マシンをインポートするには、いくつかの方法があります。最も一般的な方法は以下の通りです：

- ネイティブインポートウィザードの使用。これはインポートコンテンツタイプ（例：ESXi特殊ストレージが提供する）を利用します。
- ソース側でバックアップを実行し、ターゲット側で復元する方法。この方法は、別のProxmox VEインスタンスからの移行に最適です。
- 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方法。

他のハイパーバイザーからProxmox VEへVMをインポートする場合は、[Proxmox VEの概念](#)を理解しておくことを推奨します。

### 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>		
<button>Import</button>			

Proxmox VE は、API および Web ベースのユーザーインターフェイスにネイティブに統合される、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ています。これを使用すると、VM を全体としてインポートでき、その設定の大半が Proxmox VE の設定モデルにマッピングされ、ダウンタイムが短縮されます。

#### 注

インポートウィザードはProxmox VE 8.2の開発サイクル中に追加され、技術レビュー段階にあります。すでに有望で安定した動作を示していますが、現在も活発に開発中です。

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の新規ストレージを設定する必要があります。ウェブインターフェースの「データセンター」→「→」→「ストレージ」→「→」→「追加」で設定可能です。

次に、リソースツリーで新しいストレージを選択し、[仮想ゲスト]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表示できます。

Import Guest - esxi-7.0:ha-datacenter/tom-nasi/deb-mediawiki/deb-mediawiki.vmx (X)

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

いずれかを選択し、[インポート]ボタン（またはダブルクリック）で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一部を変更し、インポートを開始できます。なお、インポート完了後にもより高度な変更が可能です。

#### ヒント

ESXiインポートウィザードはESXiバージョン6.5から8.0まででテスト済みです。vSANストレージを使用するゲストは直接インポートできず、まずディスクを別のストレージに移動する必要があります。vCenterをインポートソースとして使用することは可能ですが、パフォーマンスが大幅に低下します（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手順とヒントについては、[Proxmox VEへの移行に関するWiki記事](#)を参照してください。

#### OVA/OVFインポート

OVA/OVFファイルをインポートするには、まずインポートコンテンツタイプが設定されたファイルベースストレージが必要です。このストレージにはインポートフォルダが作成され、OVAファイルまたはOVFファイルを対応するイメージと共にフラット構造で配置できます。あるいはWeb UIからOVAファイルを直接アップロード/ダウンロードすることも可能です。その後Web UIで該当ファイルを選択し、インポートウィザードを使用してゲストをインポートします。

OVAファイルの場合、イメージを一時的に抽出するための追加スペースが必要です。これにはイメージコンテンツタイプが設定されたファイルベースストレージが必要です。デフォルトではソースストレージが選択されますが、実際のターゲットストレージにインポートする前にイメージを抽出する「インポート作業用ストレージ」を指定することも可能です。

#### 注記

OVA/OVFファイルの構造と内容は常に適切に維持または定義されているとは限らないため、ゲスト設定の一部を手動で調整する必要が生じる場合があります。例えば、SCSIコントローラタイプはOVA/OVFファイルで定義されることはありませんが、デフォルト設定ではOVMF (UEFI) で起動できません。そのため、このようなケースでは*Virtio SCSI*または*VMware PVSCSI*を選択する必要があります。

## 10.7.2 CLI経由でのOVF/OVAインポート

外部ハイパーバイザーからのVMエクスポートは通常、1つ以上のディスクイメージと、VMの設定 (RAM、コア数) を記述した設定ファイルで構成されます。

ディスクイメージは、VMwareやVirtualBox由来の場合はvmdk形式、KVMハイパーバイザー由来の場合はqcow2形式である可能性があります。VMエクスポートの最も一般的な構成フォーマットはOVF標準ですが、実際には相互運用性が制限されています。これは、多くの設定が標準自体に実装されておらず、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です。

フォーマットの問題に加え、エミュレートされるハードウェアがハイパーバイザー間で大きく異なる場合、他ハイパーバイザーからのディスクイメージインポートは失敗する可能性があります。特にWindows VMはこの問題に注意が必要です。OSがハードウェア変更に非常に敏感だからです。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タイプをIDEに選択することで解決できる場合があります。

最後に、仮想化ドライバーの問題があります。これはエミュレートシステムの速度を向上させ、ハイパーバイザー固有のものです。GNU/Linuxやその他のフリーUnix OSでは、必要なドライバーがすべてデフォルトでインストールされており、VMインポート直後に仮想化ドライバーに切り替えることができます。Windows VMの場合、Windows仮想化ドライバーを自身でインストールする必要があります。

GNU/Linuxおよびその他のフリーUnixは通常、問題なくインポートできます。上記の課題により、Windows VMのインポート/エクスポートが常に成功するとは保証できませんのでご注意ください。

### Windows OVFインポートの手順例

MicrosoftはWindows開発を始めるための[仮想マシン提供しています](#)。OVFインポート機能のデモにこれらの一つを使用します。

#### 仮想マシンのzipファイルをダウンロード

利用規約を確認後、VMwareプラットフォーム向けのWindows 10 Enterprise (評価版- ビルド) を選択し、zipファイルをダウンロードします。

#### zipからディスクイメージを抽出

お好みの解凍ユーティリティ (unzipなど) でzipファイルを解凍し、ovfファイルとvmdkファイルをssh/scp経由で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数、メモリ、VM名を使用して新しい仮想マシンが作成され、ディスクがlocal-lvmストレージにインポートされます。ネットワーク設定は手動で行う必要があります。

```
# qm importovf 999 WinDev1709Eval.ovf local-lvm
```

仮想マシンの起動準備が整いました。

## 仮想マシンへの外部ディスクイメージの追加

既存のディスクイメージを仮想マシンに追加することも可能です。これは外部ハイパーバイザーからのもの、または自分で作成したもののいずれでも構いません。

vmdebootstrapツールでDebian/Ubuntuディスクイメージを作成したと仮定します：

```
vmdebootstrap --verbose \--size 10GiB \--serial-console \
\--size 10GiB \--serial-console \
\--grub \--no-extlinux \
\--package openssh-server \
\--package avahi-daemon \
\--package qemu-guest-agent \
\--hostname vm600 \--enable-dhcp \
\--customize=./copy_pub_ssh.sh \
\--sparse \--image vm600.raw
```

新しいターゲットVMを作成し、イメージをストレージpvmdirにインポートしてVMのSCSIコントローラに接続できます：

```
# qm create 600 \--net0 virtio,bridge=vmbr0 \--name vm600 \--serial0 socket \--boot order=scsi0 \--scsihw virtio-scsi-pci \
\--ostype 126 \
\--boot order=scsi0 \--scsihw virtio-scsi-pci \--ostype 126 \
\--scsi0 pvmdir:0,import-from=/path/to/dir/vm600.raw
```

VMは起動準備が整いました。

## 10.8 Cloud-Init サポート

Cloud-Initは、仮想マシンインスタンスの初期化処理を扱う事実上のマルチディストリビューションパッケージです。Cloud-Initを利用することで、ハイパーバイザ一側でのネットワークデバイスやSSHキーの設定が可能になります。VMが初回起動時に、VM内部のCloud-Initソフトウェアがこれらの設定を適用します。

多くのLinuxディストリビューションでは、主にOpenStack向けに設計された、すぐに使えるCloud-Initイメージを提供しています。これらのイメージはProxmox VEでも動作します。このような既製のイメージ入手するのは便利に思えるかもしれません、通常はご自身でイメージを準備することをお勧めします。その利点は、インストールされた内容を正確に把握できることであり、これにより後で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ようになります。

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推奨します。VMテンプレートからリンクドクローンを迅速に作成できるため、新しいVMインスタンスを展開する効率的な方法となります。新しいVMを起動する前に、ネットワーク設定（および必要に応じてSSHキー）を構成するだけで済みます。

Cloud-InitでプロビジョニングされたVMへのログインには、SSHキーベースの認証の使用を推奨します。パスワードの設定も可能ですが、Proxmox VEがCloud-Init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ISOイメージを生成します。このため、すべてのCloud-Init VMにはCD-ROMドライブの割り当てが必要です。通常はシリアルコンソールを追加し、表示デバイスとして使用します。多くのCloud-Initイメージはこれに依存しており、OpenStackでは必須要件です。ただし、他のイメージではこの設定で問題が発生する可能性があります。シリアルコンソールが機能しない場合は、デフォルトのディスプレイ設定に戻してください。

## 10.8.1 Cloud-Initテンプレートの準備

最初のステップは仮想マシン（VM）の準備です。基本的にどのVMでも使用できます。準備したいVM内にCloud-Initパッケージをインストールするだけです。Debian/Ubuntuベースのシステムでは、以下のように単純に実行します：

Debian/Ubuntuベースのシステムでは、以下のように単純です：

```
apt-get install cloud-init
```



### 警告

このコマンドはProxmox VEホスト上で実行するものではなく、VM内部でのみ実行してください。

多くのディストリビューションでは、すぐに使用できるCloud-Initイメージ（.qcow2ファイルとして提供）が用意されています。そのため、代わりに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可能です。以下の例では、Ubuntuが<https://cloud-images.ubuntu.com>で提供しているクラウドイメージを使用します。

```
# イメージをダウンロード
wget https://cloud-images.ubuntu.com/bionic/current/bionic-server-cloudimg-amd64.img

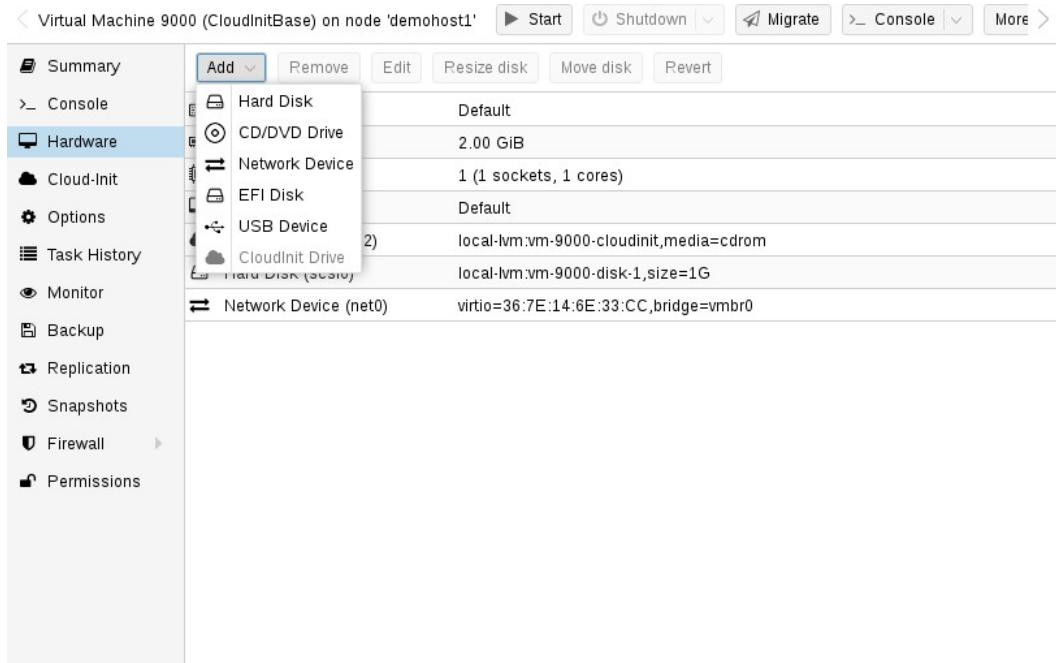
# VirtIO SCSI コントローラで新しい VM を作成
qm create 9000 --memory 2048 --net0 virtio,bridge=vmbr0 --scsihw virtio-scsi-pci

# 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し、SCSIドライブとして接続
qm set 9000 --scsi0 local-lvm:0,import-from=/path/to/bionic-server-cloudimg-amd64.img
```

### 注記

Ubuntu Cloud-Initイメージでは、SCSIドライブにvirtio-scsi-pciコントローラタイプが必要です。

### Cloud-Init CD-ROM ドライブの追加



次のステップは、Cloud-Init データを VM に渡すために使用する CD-ROM ドライブを設定することです。

```
qm set 9000 --ide2 local-lvm:cloudinit
```

クラウドインイニシャルイメージから直接起動できるようにするには、ブートパラメータをorder=scsi0に設定し、BIOSがこのディスクからのみ起動するよう制限します。これにより、VM BIOSが起動可能なCD-ROMのテストをスキップするため、起動が高速化されます。

```
qm set 9000 --boot order=scsi0
```

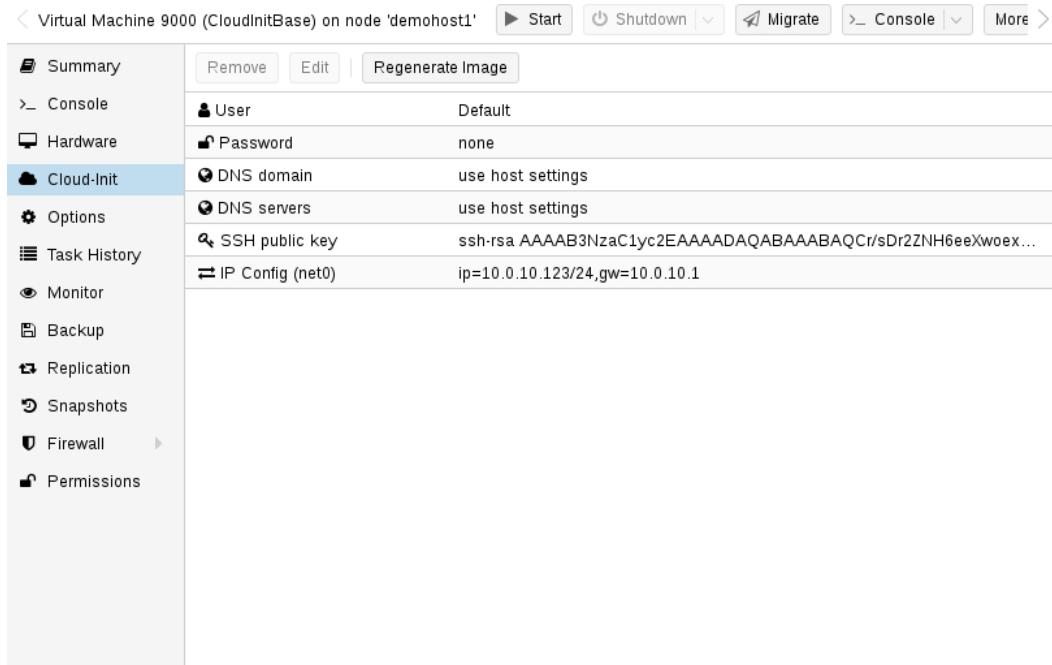
多くのCloud-Initイメージでは、シリアルコンソールを設定しディスプレイとして使用する必要があります。ただし、特定のイメージで設定が機能しない場合は、代わりに

```
qm set 9000 --serial0 socket --vga serial0
デフォルトのディスプレイに戻してください。
```

最後のステップとして、VMをテンプレートに変換すると便利です。このテンプレートから、リンクドクローンを素早く作成できます。VMテンプレートからのデ

```
qm template 9000
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です。
```

## 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デプロイできます:

```
qm clone 9000 123 --name ubuntu2
```

次に、認証に使用するSSH公開鍵を設定し、IP設定を構成します:

```
qm set 123 --sshkey ~/.ssh/id_rsa.pub
qm set 123 --ipconfig0 ip=10.0.10.123/24,gw=10.0.10.1
```

すべてのCloud-Initオプションは単一のコマンドで設定することも可能です。上記の例は単にコマンドを分割して行長を短縮したものです。また、ご自身の環境に合わせてIP設定を適宜変更してください。

## 10.8.3 カスタム Cloud-Init 設定

Cloud-Init 統合では、自動生成された設定ファイルの代わりにカスタム設定ファイルを使用することも可能です。これはコマンドラインの `cicustom` オプションを使用して行います:

```
qm set 9000 --cicustom "user=<ボリューム名>,network=<ボリューム名>,meta=<ボリューム名>"
```

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存在し、VMが移行されるすべてのノードで利用可能である必要があります。そうでない場合

```
qm set 9000 --cicustom "user=local:snippets/userconfig.yaml"
```

、VMは起動できません。例：

Cloud-Initには3種類の設定があります。1つ目は上記の例にあるユーザー設定です。2つ目はネットワーク設定、3つ目はメタ設定です。これらをまとめて指定することも、必要に応じて組み合わせて指定することも可能です。カスタム設定ファイルが指定されていない項目には、自動生成された設定が使用されます。

生成された設定はダンプしてカスタム設定のベースとして利用できます:

```
qm cloudinit dump 9000 user
```

ネットワークとメタにも同様のコマンドが存在します。

## 10.8.4 Windows上のCloud-Init

Windows向けにCloud-Initを再実装したcloudbase-initが利用可能です。Cloudbase-InitではCloud-Initの全機能が利用可能なわけではなく、一部の機能はCloud-Initとは異なります。

Cloudbase-Initでは、ostypeを任意のWindowsバージョンに設定するとともに、citypeをconfigdrive2に設定する必要があります。これは、Windows ostypeの場合のデフォルト設定です。

Windows用の既製のクラウドイメージは無料で利用できません。Cloudbase-Initを使用するには、Windowsゲストを手動でインストールおよび設定する必要があります。

## 10.8.5 Cloudbase-Initテンプレートの準備

最初のステップは、VMにWindowsをインストールすることです。ゲスト内でCloudbase-Initをダウンロードしてインストールします。ベータ版のインストールが必要になる場合があります。インストール終了時にSysprepを実行しないでください。代わりに、まずCloudbase-Initを設定してください。

設定する一般的なオプションとしては以下が挙げられます:

- *username*: 管理者のユーザー名を設定します
- *groups*: ユーザーをAdministratorsグループに追加します
- *inject\_user\_password*: trueに設定すると、VM設定でパスワードを設定可能になります
- *first\_logon\_behaviour*: ログイン時に新しいパスワードを要求しないようにするにはnoに設定
- *rename\_admin\_user*: trueに設定すると、デフォルトの管理者ユーザー名をusernameで指定したユーザー名に変更可能
- *metadata\_services*: Cloudbase-Initが最初にこのサービスを検査するようにするには、これを  
cloudbaseinit.metadata.services.configdrive.ConfigDriに設定してください。そうしないと、Cloudbase-Initが起動後にシステムを設定するのに数分かかる場合があります。

一部のプラグイン（例: SetHostnamePlugin）は再起動を必要とし、自動的に実行します。Cloudbase-Initによる自動再起動を無効化するには、allow\_rebootをfalseに設定してください。

設定オプションの完全な一覧は、[公式の cloudbase-init ドキュメント](#)で確認できます。

設定後にスナップショットを作成すると、設定の一部を調整する必要が生じた場合に便利です。Cloudbase-Initの設定後、テンプレートの作成を開始できます。Windowsゲストをシャットダウンし、Cloud-Initディスクを追加してテンプレート化します。

```
qm set 9000 --ide2 local-lvm:cloudinit qm template 9000
```

テンプレートを新しいVMにクローンします:

```
qm clone 9000 123 --name windows123
```

次にパスワード、ネットワーク設定、SSHキーを設定します:

```
qm set 123 --cipassword <password>
qm set 123 --ipconfig0 ip=10.0.10.123/24,gw=10.0.10.1qm set 123 --sshkey
~/.ssh/id_rsa.pub
```

パスワード設定前に、`ostype`が任意のWindowsバージョンに設定されていることを確認してください。さもないとパスワードが暗号化され、Cloudbase-Initが暗号化されたパスワードを平文パスワードとして使用します。

すべての設定が完了したら、クローンしたゲストを起動します。初回起動時にはログインが機能せず、変更されたホスト名のために自動的に再起動します。再起動後、新しいパスワードが設定され、ログインが機能するはずです。

## 10.8.6 Cloudbase-Init と Sysprep

SysprepはWindowsの設定をリセットし、新しいシステムを提供する機能です。Cloudbase-Initと組み合わせてクリーンなテンプレートを作成できます。

Sysprepを使用する際には、2つの構成ファイルを適応させる必要があります。1つ目は通常の構成ファイル、2つ目は`-unattend.conf`で終わるファイルです。

Cloudbase-Initは2段階の実行を行います。まず`-unattend.conf`を使用したSysprepステップを実行し、その後プライマリ設定ファイルを使用した通常ステップを実行します。

Windows Serverでは、提供されている`Unattend.xml`ファイルを使用してSysprepを実行すれば、そのまま動作するはずです。ただし、通常のWindowsバージョンでは追加の手順が必要です：

1. PowerShellインスタンスを開く

2. 管理者ユーザーを有効化：

```
net user Administrator /active:yes
```

3. 管理者ユーザーを使用してCloudbase-Initをインストールする

4. sysprep後の初回起動時に管理者ユーザーを有効化するコマンドを`Unattend.xml`に追加します：

```
&lt;RunSynchronousCommand wcm:action="add"&gt;  
  &lt;Path&gt;net user administrator /active:yes&lt;/Path&gt;  
  &lt;Order&gt;1&lt;/Order&gt;  
  &lt;Description&gt;管理者ユーザーを有効化&lt;/Description&gt;  
&lt;/RunSynchronousCommand&gt;
```

他の同期コマンドと`&lt;Order&gt;`が競合しないことを確認してください。このコマンドの後に実行するCloudbase-Initコマンドの`&lt;Order&gt;`を、数値を大きくして変更してください。

の値を大きい数値に変更して、このコマンドの後に実行されるようにしてください：

```
&lt;Order&gt;2&lt;/Order&gt;
```

5. (Windows 11のみ) 競合するMicrosoft.OneDriveSyncパッケージを削除します：

```
Get-AppxPackage -AllUsers Microsoft.OneDriveSync | Remove-AppxPackage ←  
-AllUsers
```

6. Cloudbase-Init設定ディレクトリに移動：

```
cd 'C:\Program Files\Cloudbase Solutions\Cloudbase-Init\conf'
```

7. (オプション) 設定ミスに備え、Sysprep前にVMのスナップショットを作成

8. Sysprepを実行：

```
C:\Windows\System32\Sysprep\sysprep.exe /generalize /oobe /unattend:→
Unattend.xml
```

上記の手順を実行すると、SysprepによりVMはシャットダウン状態になります。これでテンプレート化、クローン作成、必要に応じた設定が可能になります。

## 10.8.7 Cloud-Init固有のオプション

**cicustom:** [<メタ=>] [,ネットワーク=>] [,ユーザー=>] [,ベンダー=>]

起動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します。

**meta= <volume>;**

VMに"." cloud-init経由で渡される全メタデータを含むカスタムファイルを指定します。これはプロバイダー固有であり、configdrive2とnocloudでは異なります。

**network= <volume>;**

cloud-init経由でVMに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渡すために使用します。

**user= <volume>;**

cloud-init経由でユーザーデータを含むカスタムファイルをVMに渡す。

**ベンダー=>;**

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: <string>;**

ユーザーに割り当てるパスワード。通常は使用を推奨しません。代わりにSSHキーを使用してください。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点に注意してください。

**citype: <configdrive2| nocloud| opennebula>;**

cloud-init 設定フォーマットを指定します。デフォルトは設定されたオペレーティングシステムタイプ (ostype) に依存します。Linux では nocloud フォーマット、Windows では configdrive2 フォーマットを使用します。

**ciupgrade: <boolean>; (デフォルト= 1)**

初回起動後に自動パッケージアップグレードを実行します。

**ciuser: <string>;**

イメージに設定されたデフォルトユーザーではなく、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>]**

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してください。

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ですが、同じタイプのIPアドレスを指定する必要があります。

IPアドレスにDHCPを使用する場合は特別な文字列「`dhcp`」を使用でき、この場合明示的なゲートウェイは指定しないでください。IPv6では特別な文字列「`auto`」を使用してステートレス自動設定を利用できます。これにはcloud-init 19.4以降が必要です。

cloud-initが有効でIPv4/IPv6アドレスが指定されていない場合、デフォルトでIPv4のdhcpを使用します。

**gw= &lt;GatewayIPv4&gt;**  
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= &lt;GatewayIPv6&gt;**  
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= &lt;IPv4Format/CIDR&gt; (デフォルト= dhcp)**  
CIDR形式のIPv4アドレス。

**ip6= &lt;IPv6Format/CIDR&gt; (デフォルト= dhcp)**  
CIDR形式のIPv6アドレス。

**nameserver: &lt;string&gt;**

コンテナのDNSサーバーIPアドレスを設定します。searchdomainとnameserverの両方が設定されていない場合、作成時にはホストの設定が自動的に使用されます。  
。

**searchdomain: &lt;string&gt;**

コンテナのDNS検索ドメインを設定します。searchdomainとnameserverの両方が設定されていない場合、作成時にホストの設定が自動的に使用されます。

**sshkeys: &lt;string&gt;**

公開SSHキーを設定します（1行に1キー、OpenSSH形式）。

## 10.9 PCI(e)バススルー

PCI(e)バススルーは、仮想マシンがホストからPCIデバイスを制御できるようにする仕組みです。仮想化されたハードウェアを使用する場合と比較して、レイテンシの低減、パフォーマンスの向上、機能の拡張（例：オフロード）などの利点があります。

ただし、デバイスを仮想マシンにバススルーすると、そのデバイスをホストや他の仮想マシンで利用できなくなります。

なお、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

がPCI速度でしか動作しないわけではありません。デバイスをPCIeとしてバススルーすることは、ゲストに対して「非常に高速なレガシーPCIデバイス」ではなくPCIeデバイスであることを示す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 一般的な要件

バススルーは実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ります。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情報は、[PCI/バススルーの例](#)を参照してください。

### ハードウェア

お使いのハードウェアは、IOMMU(I/Oメモリ管理ユニット)による割り込み再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-Vを搭載したAMDシステムはこれをサポートしています。しかし、ハードウェア実装の不備やドライバーの欠如・低品質により、すべてがそのまま動作するとは限りません。

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向けハードウェアよりもサポートが充実していることが多いですが、それでも多くの現代システムでこの機能をサポートできます。

Linux環境における特定の構成での本機能サポート可否については、ハードウェアベンダーにお問い合わせください。

### PCIカードのアドレスの特定

最も簡単な方法は、VMのハードウェアタブで「ホストPCI」タイプのデバイスを追加するGUIを使用することです。あるいは、コマンドラインを使用することもできます。

以下のコマンドでカードの位置を確認できます。

```
lspci
```

### 設定

ハードウェアがバススルーをサポートしていることを確認したら、PCI(e)バススルーを有効にするために設定を行う必要があります。

### IOMMU

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と呼ばれます。正確なオプション名はマザーボードのマニュアルで確認してください。

AMD CPUではIOMMUがデフォルトで有効化されています。最近のカーネル(6.8以降)では、Intel CPUでも同様です。古いカーネルでは、Intel CPUでIOMMUを有

```
intel_iommu=on
```

効化するには[カーネルコマンドライン](#)に以下を追加する必要があります：

## IOMMUバススルーモード

ハードウェアがIOMMUバ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スが向上する可能性があります。これは、VMが（デフォルトの）ハイパーバイザによって通常実行される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

```
iommu=pt  
を有効にするには、以下を追加します：
```

カーネルコマンドラインに追加します。

## カーネルモジュール

以下のモジュールが確実にロードされていることを確認してください。これには、それらを『*/etc/modules*』に追加することで実現できます。

### 仲介デバイスバススルー

仲介デバイス（例：vGPU）をバススルーする場合、以下の設定は不要です。この場合、デバイスは適切なホストドライバによって直接所有されます。

```
vfio vfio_iommu_type1  
vfio_pci
```

モジュール関連の設定を変更した後は、*initramfs* を更新する必要があります。Proxmox VE では以下のコマンドを実行します：

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

モジュールが読み込まれているかどうかを確認するには、以下の出力を確認します。

```
# lsmod | grep vfio
```

上記の4つのモジュールが含まれていることを確認してください。

## 設定の完了

最後に再起動して変更を有効化し、実際に有効になっていることを確認します。

```
# dmesg | grep -e DMAR -e IOMMU -e AMD-Vi
```

IOMMU、Directed I/O、または割り込み再マッピングが有効になっていることを示すメッセージが表示されるはずです。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る場合があります。

IOMMUが意図した通りに動作しているかを確認またはトラブルシューティングする方法については、Wikiの「[IOMMUパラメータの確認](#)」セクションを参照してください。

バススルー対象デバイスが別のIOMMUグループに属していることも重要です。これはProxmox VE API呼び出しで確認できます：

```
# pvesh get /nodes/{nodename}/hardware/pci --pci-class-blacklist ""
```

デバイスが、その機能（例：HDMIオーディオデバイス付きのGPU）やルートポート、PCI(e)ブリッジと同じIOMMUグループに属していても問題ありません。

### PCI(e) スロット

一部のプラットフォームでは物理的なPCI(e)スロットの扱いが異なります。そのため、意図したIOMMUグループ分離が得られない場合、カードを別のPCI(e)スロットに挿し替えることで改善されることがあります。

### 安全でない割り込み

一部のプラットフォームでは、安全でない割り込みを許可する必要がある場合があります。その場合は、`/etc/modprobe.d/` ディレクトリ内の「.conf」で終わ

```
options vfio_iommu_type1 allow_unsafe_interrupts=1
```

るファイルに以下の行を追加してください：

このオプションはシステムの不安定化を招く可能性があることにご注意ください。

### GPUバススルーリーに関する注意事項

Proxmox VEのWebインターフェースでは、NoVNCやSPICEを介したGPUフレームバッファの表示は不可能です。

GPU または vGPU 全体をバススルーリーし、グラフィック出力を必要とする場合は、カードにモニターを物理的に接続するか、ゲスト内でリモートデスクトップソフトウェア（VNC や RDP など）を設定する必要があります。

OpenCL や CUDA を使用するプログラムなど、GPU をハードウェアアクセラレータとして使用したい場合は、これは必要ありません。

## 10.9.2 ホストデバイスのバススルーリー

PCI(e)/バススルーリーで最もよく使用されるバリエーションは、GPUやネットワークカードなどのPCI(e)カード全体をバススルーリーすることです。

### ホスト設定

Proxmox VEはホストがPCI(e)デバイスを利用できない状態にするよう自動的に試みます。ただし、これが機能しない場合、以下の2つの方法があります：

- `vfio-pci`モジュールのオプションにデバイスIDを渡すために、以下を追加します：

```
options vfio-pci ids=1234:5678,4321:8765
```

`/etc/modprobe.d/` 内の .conf ファイルに追加します。ここで 1234:5678 と 4321:8765 はベンダーIDとデバイスIDであり、以下のコマンドで取得できます：

```
# lspci -nn
```

- ホスト上でドライバを完全にブラックリスト登録し、バススルーリー用にバインド可能であることを保証します。

```
ブラックリスト DRIVERNAME
```

`/etc/modprobe.d/` ディレクトリ内の .conf ファイル

に記述します。ドライバ名を確認するには、以下の

コマンドを実行してください

```
# lspci -k
```

例：

```
# lspci -k | grep -A 3 "VGA"
```

以下のような出力が得られます

```
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: Micro-Star International Co., Ltd. [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>;
```

次に、.conf ファイルに記述することでドライバをブラックリスト登録できます:

```
echo "blacklist <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ります。

これが機能しない場合、vfio-pciをロードする前にGPUモジュールをロードするためのソフト依存関係を設定する必要があるかもしれません。これはsoftdepフラグを使用して実現できます。詳細はmodprobe.dのマニュアルページも参照してください。

例として、<some-module> という名前のドライバを使用している場合：

```
# echo "softdep <some-module> pre: vfio-pci">> /etc/modprobe.d/<some-<-->
module> .conf
```

## 設定の確認

変更が正常に適用されたか確認するには、次のコマンドを使用できます

```
# lspci -nnk
```

を実行し、デバイスエントリを確認してください。

```
Kernel driver in use: vfio-pci
```

または[使用中の行](#)が完全に欠落している場合、デバイスはバススルーに使用できる状態です。

## 仲介デバイス

仲介デバイスでは、この行は異なります。デバイスはvfio-pciではなく、ホストドライバが直接所有するためです。

## VM設定

GPUバススルー時、最高の互換性はマシンタイプとしてq35を使用し、SeaBIOSの代わりにOVMF (VM用UEFI) 、PCIの代わりにPCIeを使用することで達成されます。GPUバススルーにOVMFを使用する場合、GPUがUEFI対応ROMを搭載している必要があることに注意してください。そうでない場合は代わりにSeaBIOSを使用してください。ROMがUEFI対応かどうかを確認するには、[PCIバススルーリー例](#)を参照してください。

さらに、OVMFを使用することでVGAアビトリレーションを無効化できる可能性があり、これにより起動時に実行が必要なレガシーコードの量を削減できます。VGAアビトリレーションを無効化するには：

```
echo "options vfio-pci ids=<ベンダーID>;<デバイスID> disable_vga=1" > /etc/→
modprobe.d/vfio.conf
```

&lt;vendor-id&gt; および &lt;device-id&gt; を以下で取得した値に置き換える:

```
# lspci -nn
```

PCIデバイスは、VMのハードウェアセクションのWebインターフェースで追加できます。あるいは、コマンドラインを使用することもできます。VM設定で

```
# qm set VMID -hostpci0 00:02.0
```

hostpciXオプションを設定します。例えば、以下を実行します：

または、VM設定ファイルに次の行を追加します:

```
hostpci0: 00:02.0
```

デバイスに複数の機能（例: '00:02.0' と '00:02.1'）がある場合、短縮構文 ``00:02`` でまとめて通過させることができます。これはウェブインターフェースで ``All Functions`` チェックボックスをオンにするのと同等です。

デバイスやゲストOSによっては、以下のオプションが必要になる場合があります:

- **x-vga=on|off** は、PCI(e) デバイスを仮想マシンのプライマリ GPU として指定します。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 Proxmox VE に PCIe または PCI ポートの使用を指示します。一部のゲスト/デバイスの組み合わせでは PCI ではなく PCIe が必要です。PCIe は q35 マシンタイプでのみ利用可能です。
- **rombar=on|off** は、ファームウェア ROM をゲストから可視化します。デフォルトは **on** です。一部の PCI(e) デバイスではこれを無効にする必要があります。
- **romfile=&lt;path&gt;** は、デバイスが使用するROMファイルへのオプションパスです。これは **/usr/share/kvm/**

## 例

GPUをプライマリに設定したPCIe/バススルーリーの例:

```
# qm set VMID -hostpci0 02:00,pcie=on,x-vga=on
```

## PCI IDの上書き

ゲストが認識する PCI ベンダー ID、デバイス ID、サブシステム ID を上書きできます。これは、お使いのデバイスがゲストのドライバが認識しない ID を持つバリエーションである場合、それでもそれらのドライバを強制的にロードしたい場合に有用です（例: お使いのデバイスがサポートされているバリエーションと同じチップセットを共有しているとわかっている場合）。

利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できます。

例:

```
# qm set VMID -hostpci0 02:00,device-id=0x10f6,sub-vendor-id=0x0000
```

## 10.9.3 SR-IOV

PCI(e)デバイスをパススルーする別の方法として、利用可能な場合、デバイスのハードウェア仮想化機能を使用する方法があります。

### SR-IOVの有効化

SR-IOVを利用するには、プラットフォームのサポートが特に重要です。この機能を動作させるには、まずBIOS/UEFIで有効にする必要がある場合や、特定のPCI(e)ポートを使用する必要があります。不明な点がある場合は、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（シングルルート入出力仮想化）は、単一デバイスがシステムに複数のVF（仮想機能）を提供する機能です。各VFは異なるVMで使用可能で、完全なハードウェア機能に加え、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遅延を実現します。

現在、この技術で最も一般的なユースケースはSR-IOV対応NIC（ネットワークインターフェースカード）であり、物理ポートごとに複数のVFを提供できます。これにより、チェックサムオフロードなどの機能をVM内で使用可能となり、（ホスト）CPUのオーバーヘッドを削減できます。

### ホスト構成

デバイス上で仮想機能を有効にする方法は、一般的に2通りあります。

- 場合によっては、ドライバモジュールにオプションが存在します。例：一部のIntel ドライバ

```
max_vfs=4
```

この設定は、`/etc/modprobe.d/` 配下に `.conf` 拡張子を持つファイルとして記述できます。（設定後は `initramfs` の更新を忘れないでください）

正確なパラメータやオプションについては、お使いのドライバモジュールのドキュメントを参照してください。

- 2つ目の、より汎用的な方法はsysfsを使用することです。デバイスとドライバがこれをサポートしている場合、VFの数を動的に変更できます。例えば、デバイス`0000:01:00.0`に4つのVFを設定するには以下を実行します：

```
# echo 4 > /sys/bus/pci/devices/0000:01:00.0/sriov_numvfs
```

この変更を永続化するには、Debianパッケージ「sysfsutils」を使用できます。インストール後、`/etc/sysfs.conf` または `/etc/sysfs.d/` 内の「FILE.conf」で設定してください。

### VM設定

VFを作成後、`lspci` で出力すると個別のPCI(e)デバイスとして確認できるはずです。それらのIDを取得し、[通常のPCI\(e\)デバイスと同様に渡してください](#)。

## 10.9.4 仲介デバイス(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で再利用する別の方法です。IntelのGVT-gやNVIDIAのGRID技術で使用されるvGPUなど、仮想化GPU構成で最も一般的に見られます。

これにより、物理カードはSR-IOVと同様に仮想カードを生成できます。違いは、仲介デバイスがホスト上でPCI(e)デバイスとして認識されず、仮想マシンでの使用にのみ適している点です。

## ホスト構成

一般的に、お使いのカードのドライバーがその機能をサポートしている必要があります。サポートされていない場合、機能しません。互換性のあるドライバーと設定方法については、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世代、第6世代、第7世代インテルCoreプロセッサー、ならびにE3 v4、E3 v5、E3 v6 Xeonプロセッサーで動作するはずです。

Intelグラフィックスで有効化するには、[モジュールkvmtをロード](#)（例：/etc/modules経由）し、[カーネルコマンドラインで有効化](#)するとともに、以下のパラメータ

```
i915.enable_gvt=1  
を追加する必要があります：
```

その後、[initramfsを更新し](#)、ホストを再起動することを忘れないでください。

## VM設定

仲介デバイスを使用するには、hostpciX VM設定オプションで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一覧表示するには、単に以下を実行します：

```
# ls /sys/bus/pci/devices/0000:00:02.0/mdev_supported_types
```

各エントリはディレクトリであり、以下の重要なファイルを含みます：

- available\_instancesには、このタイプのまだ利用可能なインスタンス数が含まれます。VM内でmdevが使用されるたびにこの数は減少します。
- descriptionには、このタイプの機能に関する簡単な説明が含まれます
- createは、そ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。Proxmox VEは、hostpciXオプションが設定されている場合、これを自動的に行います。オプションが設定されている場合、Proxmox VEはこれを自動的に行います。

Intel GVT-g vGPU (Intel Skylake 6700k) を使用した設定例:

```
# qm set VMID -hostpci0 00:02.0,mdev=i915-GVTg_V5_4
```

この設定により、Proxmox VEはVM起動時に自動的にこのデバイスを作成し、VM停止時に再度クリーンアップします。

## 10.9.5 クラスタでの使用

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す。これにより、HA環境で適切に利用でき、ハードウェア変更が検出され、非rootユーザーでも設定が可能になります。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デバイス向けの高度なメモリアクセス制御とセキュリティを提供します。vIOMMUオプションを使用すると、[ネストド仮想化](#)を介してレベル1 VMからレベル2 VMへPCI(e)デバイスをバススルーすることも可能です。ホストからネストドVMへ物理PCI(e)デバイスをバススルーするには、PCI(e)バススルーの手順に従ってください。

現在、2つのvIOMMU実装が利用可能です：IntelとVirtIO。

### Intel vIOMMU

Intel vIOMMU固有のVM要件：

- ホストでIntelまたはAMD CPUを使用している場合でも、VMのカーネルパラメータで`intel\_iommu=on`を設定することが重要です。
- Intel 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る場合、PCI デバイスをバススルーできる VM の構成で、マシンパラメータに viommu=intel を追加できます。

```
# qm set VMID -machine q35,viommu=intel
```

### VT-d に関する QEMU ドキュメント

#### VirtIO vIOMMU

このvIOMMU実装はより新しく、Intel vIOMMUなどの制限はありませんが、現在では実稼働環境での使用頻度が低く、ドキュメントも少ないです。

VirtIO vIOMMUを使用する場合、カーネルパラメータを設定する必要はありません。また、マシンタイプとしてq35を使用する必要はありませんが、PCIeを利用した

```
# qm set VMID -machine q35,viommu=virtio
```

い場合には推奨されます。

Michael Zhaoによるvirtio-iommu解説ブログ記事

## 10.10 フックスクリプト

configプロパティhookscriptを使用して、VMにフックスクリプトを追加できます。

```
# qm set 100 --hookscript local:snippets/hookscript.pl
```

ゲストのライフサイクルの様々な段階で呼び出されます。例とドキュメントについては、/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 ハイバネーション

GUI オプションの「ハイバネート」または

```
# qm suspend ID --todisk
```

これにより、メモリの現在の内容がディスクに保存され、VMが停止します。次回起動時にはメモリ内容が読み込まれ、VMは中断した状態から再開できます。

### 状態保存先の選択

メモリの保存先を指定しない場合、以下の順序で自動的に選択されます：

1. VM設定からのストレージ状態。
2. いずれかのVMディスクからの最初の共有ストレージ。
3. いずれかのVMディスクから最初の非共有ストレージ。
4. フォールバックとしてローカルストレージ。

## 10.12 リソースマッピング

The screenshot shows the Proxmox VE web interface under the 'Datacenter' section. On the left, a sidebar lists various management options like Cluster, Ceph, Options, Storage, Backup, Replication, Permissions, and Resource Mappings (which is currently selected). The main area displays two tables: 'PCI Devices' and 'USB Devices'. Both tables show mappings between host resources and guest devices.

**PCI Devices**

ID/Node/Path ↑	Actions	Vendor/De...	Subsystem...	IOMMU gr...	Status	Comment
NIC	<span>Add</span>					
pve-ceph-01	<span>Edit</span>					
0000:06:12.0	<span>Edit</span>	1af4:1000	1af4:0001	9	<span>✓ Mapping matches host data</span>	
pve-ceph-02	<span>Edit</span>					
0000:06:12.0	<span>Edit</span>	1af4:1000	1af4:0001	9	<span>✓ Mapping matches host data</span>	

**USB Devices**

ID/Node/Vendor&Device ↑	Actions	Path	Status	Comment
STICK	<span>Add</span>			
pve-ceph-01	<span>Edit</span>			
0627:0001	<span>Edit</span>		<span>✓ Mapping matches host data</span>	
pve-ceph-02	<span>Edit</span>			
0627:0001	<span>Edit</span>	1-1	<span>✓ Mapping matches host data</span>	

ローカルリソース（例：PCIデバイスのアドレス）を使用または参照する場合、生のアドレスやIDを使用すると問題が生じることがあります。例えば：

- HAを使用する場合、ターゲットノード上に同一IDまたはパスを持つ異なるデバイスが存在することがあり、ゲストをHAグループに割り当てる際に注意を怠ると、誤ったデバイスが使用され、設定が破損する可能性があります。
- ハードウェアの変更によりIDやパスが変更される可能性があるため、割り当てられた全デバイスを確認し、パスやIDが依然として正しいかどうかを検証する必要があります。

この問題をより適切に処理するには、クラスタ全体のリソースマッピングを定義できます。これにより、リソースにはクラスタ内で一意のユーザー指定識別子が割り当てられ、異なるホスト上の異なるデバイスに対応させることができます。これにより、HAが誤ったデバイスでゲストを起動する事態を防ぎ、ハードウェアの変更を検知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「データセンター」内の「リソースマッピング」カテゴリの該当タブから、またはCLIで以下のように実行できます。

```
# pvsh create /cluster/mapping/<type> <options>;
```

ID ↑	IO...	Vendor	Device	Me...
0000:00:00.0	0	Intel Corporation	82G33/G31/P35/P31 Express DRAM Controller	No
0000:00:01.0	1			No
0000:00:1a	2	Intel Corporation	Pass through all functions as one device	No
0000:00:1a.0	2	Intel Corporation	82801I (ICH9 Family) USB UHCI Controller #4	No
0000:00:1a.1	2	Intel Corporation	82801I (ICH9 Family) USB UHCI Controller #5	No
0000:00:1a.2	2	Intel Corporation	82801I (ICH9 Family) USB UHCI Controller #6	No
0000:00:1a.7	2	Intel Corporation	82801I (ICH9 Family) USB2 EHCI Controller #2	No
0000:00:1b.0	3	Intel Corporation	82801I (ICH9 Family) HD Audio Controller	No
0000:00:1c	4	Red Hat, Inc.	Pass through all functions as one device	No
0000:00:1c.0	4	Red Hat, Inc.	QEMU PCIe Root port	No
0000:00:1c.1	5	Red Hat, Inc.	QEMU PCIe Root port	No

ここで `<type>` はハードウェアタイプ（現在は `pci`、`usb`、`dir` のいずれか）、`<options>` はデバイスマッピングやその他の設定パラメータです。

オプションには、ハードウェアの識別プロパティをすべて含む `map` プロパティを含める必要があります。これにより、ハードウェアが変更されていないこと、および正しいデバイスがバススルーされていることを検証できます。

例：デバイスID 0001、バス 0000:01:00.0 の PCI デバイスを `device1` として追加する場合

ノード1上のベンダーID 0002、ノード2上の0000:02:00.0は以下のように追加できます：

```
# pvsh create /cluster/mapping/pci --id device1 \
--map node=node1,path=0000:01:00.0,id=0002:0001 \
```

そのデバイスにマッピングを設定する各ノードに対して、`map` パラメータを繰り返し指定する必要があります（現在のところ、1つのマッピングにつきノードごとに1つのUSBデバイスのみをマッピングできる点に注意してください）。

GUIを使用すると、適切なプロパティが自動的に取得されAPIに送信されるため、この作業が大幅に簡略化されます。

また、PCIデバイスがノードごとに複数のデバイスを提供し、ノードごとに複数のマッププロパティを持つことも可能です。そのようなデバイスがゲストに割り当てられた場合、ゲスト起動時には最初に空き状態のデバイスが使用されます。パス指定の順序は試行順序にもなるため、任意の割り当てポリシーを実装できます。

これはSR-IOV対応デバイスで有用です。なぜなら、特定の仮想機能が通過するかどうかが重要でない場合があるためです。

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、または

```
# qm set <vmid> -usb0 <name>;
```

USBデバイスの場合。

ここで `<vmid>` はゲストID、`<name>` は作成するマッピングに付ける名前です。デバイスのパスルートに関する通常のオプション (`mdev`など) はすべて使用可能です。

マッピングを作成するには、``Mapping.Modify on/mapping/<type>/<name>;``が必要です（ここで `<type>` はデバイスタイプ、`<name>` はマッピング名です）。

これらのマッピングを使用するには、`/mapping/<type>/<name>;`に対する `Mapping.Use` が必要です（設定を編集するための通常のゲスト権限に加えて）。

#### 追加オプション

クラスタ全体のリソースマッピングを定義する際には追加オプションがあります。現在以下のオプションが利用可能です：

- `mdev (PCI)`: このオプションは、PCIデバイスが仲介デバイスを提供可能であることを示します。有効にすると、ゲスト上で設定時にタイプを選択できるようになります。マッピングに複数のPCIデバイスが選択されている場合、選択したタイプの利用可能なインスタンスが最初に存在するデバイス上で仲介デバイスが作成されます。
- `live-migration-capable (PCI)`: このオプションは、PCIデバイスがノード間でライブマイグレーション可能であることを示します。これにはドライバとハードウェアのサポートが必要です。最近のカーネルを搭載したNVIDIA GPUのみがこれをサポートすることが知られています。パスルートデバイスのライブマイグレーションは実験的な機能であり、動作しない場合や問題を引き起こす可能性があることに注意してください。

## 10.13 `qm` による仮想マシンの管理

`qm` は Proxmox VE 上で QEMU/KVM 仮想マシンを管理するためのツールです。仮想マシンの作成・削除、実行制御（起動/停止/一時停止/再開）が可能です。さらに、関連する設定ファイルのパラメータ設定にも `qm` を使用できます。仮想ディスクの作成・削除も可能です。

### 10.13.1 CLI使用例

ローカルストレージにアップロードしたISOファイルを使用し、local-lvmストレージ上に4GBのIDEディスクを持つVMを作成

```
# qm create 300 -ide0 local-lvm:4 -net0 e1000 -cdrom local:iso/proxmox-->
mailgateway_2.1.iso
```

新しい仮想マシンを開始

```
# qm start 300
```

シャットダウン要求を送信し、VMが停止するまで待機する。

```
# qm shutdown 300 && qm wait 300
```

上記と同様ですが、待機時間は40秒のみです。

```
# qm shutdown 300 && qm wait 300 -timeout 40
```

VMがシャットダウンしない場合、強制停止し、実行中のシャットダウンタスクを上書きします。VMの停止はデータ損失を引き起こす可能性があるため、注意し

```
# qm stop 300 -overrule-shutdown 1
```

て使用してください。

仮想マシンの破棄は常にアクセス制御リストからの削除を伴い、仮想マシンのファイアウォール設定を必ず削除します。レプリケーションジョブ、バックアップジョブ、および高可用性リソース構成から仮想マシンを追加で削除したい場合は、`--purge` を有効にする必要があります。

```
# qm destroy 300 --purge
```

ディスクイメージを別のストレージに移動します。

```
# qm move-disk 300 scsi0 other-storage
```

ディスクイメージを別のVMに再割り当てします。これにより、ソースVMからディスクscsi1が削除され、ターゲットVMにscsi3としてアタッチされます。バックグラウンドでは、ディスクイメージの名前が新しい所有者に一致するように変更されます。

```
# 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`でアクセス可能です

`/etc/pve/`内に保存されている他のファイルと同様に、これらは自動的に他のすべてのクラスターノードに複製されます。

#### 注

VMID<100は内部目的のために予約されており、VMIDはクラスタ全体で一意である必要があります。

## VM設定例

```
boot: order=virtio0;net0cores: 1
ソケット: 1
memory: 512name:
webmailostype: 126
net0: e1000=EE:D2:28:5F:B6:3E,bridge=vmbr0virtio0: 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、...）で編集できます。細かい修正を行う際に便利ですが、変更を反映させるにはVMの再起動が必要であることを覚えておいてください。

そのため、通常はqmコマンドを使用してこれらのファイルを生成・変更するか、GUIで全てを行う方が望ましいです。当社のツールキットは、稼働中のVMへの変更を瞬時に適用する十分な知能を備えています。この機能は「ホットプラグ」と呼ばれ、この場合VMの再起動は不要です。

## 10.14.1 ファイル形式

VM設定ファイルは、コロン区切りのキー/値形式というシンプルな形式を使用します。各行は次の形式です：

```
# これはコメントです OPTION: value
```

これらの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、同様に無視されます。

## 10.14.2 スナップショット

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点の構成を、同じ構成ファイル内の独立した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すると、構成ファイルは次のようにになります：

### スナップショット付きVM設定

```
memory: 512
swap: 512
親: testsnapshot
...
[testsnapshot] メモリ:
512
スワップ: 512
スナップタイム: 1457170803
...
```

スナップショットに関連するプロパティとして、親スナップショットやスナップショット作成時刻（スナップタイム）などがあります。親プロパティはスナップショット間の親子関係を保持するために使用されます。スナップタイムはスナップショットの作成時刻（Unixエポック）です。

実行中の仮想マシンのメモリは、オプションの `vmstate` を使用して任意で保存できます。仮想マシン状態の保存先ストレージの選択方法の詳細については、「[休止状態](#)」の章にある「[状態保存先の選択](#)」を参照してください。

### 10.14.3 オプション

**acpi: &lt;boolean&gt; (デフォルト= 1)**

ACPI を有効/無効にします。

**affinity: &lt;string&gt;**

ゲストプロセスを実行するために使用するホストコアのリスト。例: 0,5,8-11

**agent: [enabled=&lt;1|0&gt;] [,freeze-fs-on-backup=&lt;1|0&gt;] [,fstrim\_cloned\_disks=&lt;1|0&gt;] [,type=&lt;virtio|isa&gt;]**

QEMUゲストエージェントとの通信およびそのプロパティの有効化/無効化。

**有効化= &lt;boolean&gt; (デフォルト= 0)**

仮想マシン内で実行中の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= &lt;boolean&gt; (デフォルト= 1)**

バックアップ時にゲストファイルシステムを凍結/解凍して整合性を確保します。

**fstrim\_cloned\_disks= &lt;boolean&gt; (デフォルト= 0)**

ディスクの移動またはVMの移行後にfstrimを実行します。

**type= &lt;isa| virtio&gt; (デフォルト= virtio)**

エージェントタイプを選択

**amd-sev: [type=&lt;sev-type&gt;] [,allow-smt=&lt;1|0&gt;] [,kernel-hashes=&lt;1|0&gt;] [,no-debug=&lt;1|0&gt;] [,no-key-sharing=&lt;1|0&gt;]**

AMD CPUによるセキュア暗号化仮想化（SEV）機能

**allow-smt= &lt;boolean&gt; (デフォルト= 1)**

ポリシービットを設定し、同時マルチスレッディング（SMT）を許可する（SEV-SNP用でない限り無視される）

**kernel-hashes= &lt;boolean&gt; (デフォルト= 0)**

ゲストファームウェアにカーネルハッシュを追加し、Linuxカーネル起動を計測

**no-debug= &lt;boolean&gt; (デフォルト= 0)**

ゲストのデバッグを許可しないポリシービットを設定

**no-key-sharing= &lt;boolean&gt; (デフォルト= 0)**

他のゲストとのキー共有を許可しないポリシービットを設定します（SEV-SNPでは無視されます）

**type= &lt;sev-type&gt;**

`type=std` で標準 SEV を有効化、`es` オプションで実験的 SEV-ES を有効化、`snp` オプションで実験的 SEV-SNP を有効化。

**アーキテクチャ: &lt;aarch64| x86\_64&gt;**

仮想プロセッサアーキテクチャ。デフォルトはホスト。

**args: &lt;string&gt;**

kvmに渡される任意の引数。例: args: -no-reboot -smbios type=0, vendor=FOO

**注意**

このオプションは専門家のみが使用してください。

**audio0: device=&lt;ich9-intel-hda|intel-hda|AC97&gt;  
[,driver=&lt;spice|none&gt;]**

オーディオデバイスを設定します。QXL/Spiceと組み合わせて使用すると便利です。

**デバイス= &lt;AC97| ich9-intel-hda| intel-hda&gt;**

オーディオデバイスを設定します。

**driver= &lt;none| spice&gt; (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: &lt;boolean&gt; (デフォルト= 0)**

クラッシュ後の自動再起動(現在無視される)。

**balloon: &lt;integer&gt; (0 - N)**

VMのターゲットRAM容量(MiB単位)。ゼロを使用するとバルーンドライバが無効化されます。

**bios: &lt;ovmf| seabios&gt; (デフォルト= seabios)**

BIOS実装を選択します。

**boot: [[legacy]=[&lt;[acdn]{1,4}&gt;] [,order=&lt;device[:device...]&gt;]]**

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てください。キーなしまたはlegacy=の使用は非推奨です。

**レガシー= &lt;[acdn]{1,4}&gt; (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)から起動します。非推奨。代わりにorder=を使用してください。

**order= &lt;device[:device...]&gt;**

ゲストは、ここに表示される順序でデバイスからの起動を試みます。

ディスク、光学ドライブ、バススルーされたストレージUSBデバイスは直接起動されます。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(例: NVMe)、オプションROMをロードします(例: RAIDコントローラー、ハードウェアNIC)。

このリストにあるデバイスのみが起動可能としてマークされ、ゲストファームウェア(BIOS/UEFI)によって読み込まれることに注意してください。複数のディスクを起動に必要とする場合(例: ソフトウェアRAID)、それらをすべてここで指定する必要があります。

指定された場合、非推奨の legacy=[acdn]\* 値を上書きします。

**bootdisk: (ide|sata|scsi|virtio)\d+**

指定したディスクからの起動を有効化します。非推奨: 代わりに *boot: order=foo;bar* を使用してください。

**cdrom: <ボリューム>;**

オプション -ide2 の別名です

**cicustom: [meta=<ボリューム>;] [,network=<ボリューム>;] [,user=<ボリューム>;] [,vendor=<ボリューム>;]**

cloud-init: 起動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します。

**meta= <ボリューム>;**

カスタムファイルを指定し、"." cloud-init経由でVMに渡される全メタデータを格納します。これはプロバイダー固有であり、configdrive2とnocloudでは異なります。

**network= <volume>;**

cloud-init経由でネットワークデータを含むカスタムファイルをVMに渡すために使用します。

**user= <volume>;**

cloud-init経由でユーザーデータを含むカスタムファイルを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: <string>;**

cloud-init: ユーザーに割り当てるパスワード。通常は使用を推奨しません。代わりにSSHキーを使用してください。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点に注意してください。

**ciotype: <configdrive2| nocloud| opennebula>;**

cloud-initの設定形式を指定します。デフォルトは設定されたオペレーティングシステムタイプ (*ostype*) に依存します。Linuxではnocloud形式、Windowsではconfigdrive2形式を使用します。

**ciupgrade: <boolean>; (デフォルト= 1)**

cloud-init: 初回起動後に自動パッケージアップグレードを実行します。

**ciuser: <string>;**

cloud-init: イメージに設定されたデフォルトユーザーではなく、SSHキーとパスワードを変更するユーザー名。

**cores: <integer>; (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...]>;] [,hidden=<1|0>;] [,hv-vendor-id=<vendor-id>;] [,物理ビット=<8-64|ホスト>;] [,報告モデル=<列挙型>;]**

エミュレートされたCPUタイプ。

**cputype= &lt;string&gt; (デフォルト= kvm64)**

エミュレートされるCPUタイプ。デフォルトまたはカスタム名（カスタムモデル名はカスタム-で始まる必要がある）。

**flags= &lt;+FLAG[;-FLAG...]&gt;**

;で区切られた追加のCPUフラグのリスト。**+FLAG**で有効化、**-FLAG**で無効化します。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。セキュリティ上の理由から、VM固有のフラグは次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-tlflush, hv-evmcs, aes

**hidden= &lt;boolean&gt; (デフォルト= 0)**

KVM仮想マシンとして識別しない。

**hv-vendor-id= &lt;vendor-id&gt;**

Hyper-VベンダーID。Windowsゲスト内の特定のドライバやプログラムが特定のIDを必要とする場合があります。

**phys-bits= &lt;8-64|host&gt;**

ゲストOSに報告される物理メモリアドレスビット数。ホストの値以下である必要があります。ホストCPUの値を使用するにはhostに設定しますが、これにより他の値を持つCPUへのライブマイグレーションが機能しなくなることに注意してください。

**reported-model= &lt;486| Broadwell| Broadwell-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-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 | IvyBridge-IBRS | KnightsMill | Nehalem | Nehalem-IBRS | Opteron\_G1| Opteron\_G2| Opteron\_G3| Opteron\_G4| Opteron\_G5 | ペンリン | サンディブリッジ | サンディブリッジ-IBRS | サファイアラピッズ | サファイアラピッズ-v2 | スカイレイク-クライアント | スカイレイク-クライアント-IBRS | スカイレイク-クライアント-notSX-IBRS| スカイレイク-クライアント-v4| スカイレイク-サーバー | Skylake-Server-IBRS| Skylake-Server-notSX-IBRS | Skylake-Server-v4 | Skylake-Server-v5 | Westmere | Westmere-IBRS | athlon | core2duo | coreduo | host | kvm32 | kvm64| max| pentium| pentium2| pentium3| phenom| qemu32 | qemu64&gt; (デフォルト=kvm64)**

ゲストOS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モデル定義でのみ有効です。デフォルトモデルは常にゲストOSに自身を報告します。

**cpulimit: &lt;数値&gt; (0 - 128) (デフォルト= 0)**

CPU使用率の制限。

**注**

コンピュータに 2 つの CPU がある場合、合計 2 CPU 時間が割り当てられます。値 0 は CPU 制限なしを示します。

**cpuunits: <integer> (デフォルト= cgroup v1: 1024, cgroup v2: 100)**

VMのCPUウェイト。カーネルのフェアスケジューラで使用される引数。数値が大きいほど、このVMが割り当てられるCPU時間が増加します。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す。

**description: <string>;**

VMの説明。WebインターフェースのVM概要に表示されます。設定ファイル内にコメントとして保存されます。

**efidisk0: [file=<volume> [,efitype=<2m|4m>] [,format=<列挙型>]**

**[,pre-enrolled-keys=<1|0>] [,size=<ディスクサイズ>]**

EFI変数を保存するためのディスクを設定します。

**efitype= <2m| 4m> (デフォルト= 2m)**

OVMF EFI変数のサイズとタイプ。4mは新規で推奨され、セキュアブートに必須です。後方互換性のため、特に指定がない場合は2mが使用されます。arch=aarch64 (ARM) のVMでは無視されます。

**file= <volume>;**

ドライブのバックアップボリューム。

**format= <cloop| qcows| qcows2| 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、ディストリビューション固有およびMicrosoft標準キーが登録されたEFI変数テンプレートを使用します。これによりデフォルトでセキュアブートが有効化されますが、VM内部から無効化可能です。

**size= <DiskSize>;**

ディスクサイズ。これは情報提供のみであり、効果はありません。

**freeze: <boolean>;**

起動時にCPUを凍結する (c monitorコマンドを使用して実行を開始)。

**hookscript: <string>;**

VMのライフサイクルにおける様々な段階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>]]**

**[,サブデバイスID=<16進ID>] [,サブベンダーID=<16進ID>] [,ベンダー  
ID=<16進ID>] [,x-vga=<1|0>]**

ホストPCIデバイスをゲストにマッピングします。

---

**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許可します。そのため、このようなマシンの移行は不可能になります - 特別な注意を払って使用してください。

---

**注意**

実験的！このオプションに関する問題がユーザーから報告されています。

---

**device-id= &lt;16進ID&gt;**

ゲストから見えるPCIデバイスIDを上書き

**host= &lt;HOSTPCIID[,HOSTPCIID2...]&gt;**

ホストPCIデバイスのパススルー。ホストのPCIデバイスのID、またはホストのPCI仮想機能のリスト。HOSTPCIIDの構文は次の通り：

*bus:dev.func* (16進数)

既存のPCIデバイス一覧はlspciコマンドで取得可能。本設定またはマッピングキーのいずれかを

設定する必要がある。

**legacy-igd= &lt;boolean&gt; (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 内のプライマリかつ排他的なグラフィックスデバイスとします。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。

**mapping= &lt;mapping-id&gt;**

クラスタ全体のマッピングID。これがデフォルトキーホストのいずれかを設定する必要があります。

**mdev= &lt;string&gt;**

使用的する仲介デバイスのタイプ。このタイプのインスタンスはVM起動時に作成され、VM停止時にクリーンアップされます。

**pcie= &lt;boolean&gt; (デフォルト= 0)**

PCI-Express バスを選択します (q35 マシンモデルが必要です)。

**rombar= &lt;boolean&gt; (デフォルト= 1)**

デバイスのROMがゲストのメモリマップに表示されるかどうかを指定します。

**romfile= &lt;string&gt;**

カスタム PCI デバイス ROM ファイル名 (/usr/share/kvm/ に存在する必要があります)。

**sub-device-id= &lt;hex id&gt;**

ゲストに表示されるPCIサブシステムデバイスIDを上書き

**sub-vendor-id= &lt;hex id&gt;**

ゲストから見える PCI サブシステムベンダー ID を上書き

**ベンダーID= &lt;16進ID&gt;**

ゲストに表示される PCI ベンダー ID を上書き

---

**x-vga= <boolean> (デフォルト = 0)**

vfio-vga デバイスサポートを有効化。

**hotplug: <string> (デフォルト = network,disk,usb)**

ホットプラグ機能を選択的に有効化します。これはホットプラグ機能のカンマ区切りリストです：*network, disk, cpu, memory, usb, cloudinit*。0を指定するとホットプラグ機能を完全に無効化します。値として1を使用すると、デフォルトの*network, disk, usb*の別名となります。USBホットプラグはマシンバージョン7.1以上のゲストで利用可能であり、ostype l26またはWindows xml-ph-0000@deepl.internal 7.1以上が必要です。>= 7.1かつostype l26またはWindows> 7.

**hugepages: <1024| 2| any>**

巨大ページメモリの有効化/無効化。

**ide[n]: [file=<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=<秒>]  
[,bps_rd=<bps>][,bps_rd_max_length=<seconds>] [,bps_wr=<bps>]  
[,bps_wr_max_length=<seconds>] [,cache=<enum>] [,detect_zeroes=<1|0>] [,破棄=<無視|有効>] [,フォーマット=<列挙型>] [,I/O操作数=<I/O操作数>] [,最大I/O操作数=<I/O操作数>] [,最大I/O操作数持続時間=<秒>] [,読み取りI/O操作数=<I/O操作数>][,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>]  
[,serial=<serial>] [,shared=<1|0>][,size=<DiskSize>] [,snapshot=<1|0>]  
[,ssd=<1|0>] [,werror=<enum>] [,wwn=<wwn>]`  
ボリュームを IDE ハードディスクまたは CD-ROM として使用します(n は 0 から 3)。

**aio= <io\_uring| native| threads>**

使用するAIOタイプ。

**= のバックアップ <boolean>**

バックアップ作成時にドライブを含めるかどうか。

**bps= <bps>**

最大読み書き速度 (バイト/秒)。

**bps\_max\_length= <秒>**

I/O/バーストの最大長 (秒単位)。

**bps\_rd= <bps>**

最大読み取り速度 (バイト/秒)。

**bps\_rd\_max\_length= <秒>**

読み取りI/Oバーストの最大長 (秒単位)。

**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（秒単位）。

cache= <directsync| none| unsafe| writeback| writethrough>;
ドライブのキャッシングモード

detect_zeroes= <boolean>;
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します。

discard= <無視する| オン>;
ディスクカード/トリム要求を基盤ストレージに渡すかどうかを制御します。

file= <volume>;
ドライブのバックキングボリューム。

format= <cloop| qcow| qcown| qed| raw| vmdk>;
ドライブのバックキングファイルのデータ形式。

iops= <iops>;
最大読み書きI/O（秒あたりの操作数）。

iops_max= <iops>;
最大スロットリングなしの読み書きI/Oプール（秒あたりの操作数）。

iops_max_length= <seconds>;
I/Oバーストの最大長（秒単位）

iops_rd= <iops>;
最大読み取りI/O（秒あたりの操作数）。

iops_rd_max= <iops>;
スロットリングなしの最大読み取りI/Oプール（秒あたりの操作数）。

iops_rd_max_length= <seconds>;
読み取り I/O バーストの最大長（秒単位）。

iops_wr= <iops>;
最大書き込み I/O 操作数（秒あたり）。

iops_wr_max= <iops>;
最大スロットリングなし書き込みI/Oプール（秒あたりの操作数）。

iops_wr_max_length= <seconds>;
秒単位での書き込みI/Oバーストの最大長。

mbps= <mbps>;
最大読み書き速度（メガバイト毎秒）。

mbps_max= <mbps>;
最大スロットリングなし読み書きプール（メガバイト毎秒）。

mbps_rd= <mbps>;
最大読み取り速度（メガバイト/秒）。
```

**mbps\_rd\_max= &lt;mbps&gt;**  
最大スロットリングなし読み取りプール（メガバイト/秒）。

**mbps\_wr= &lt;mbps&gt;**  
最大書き込み速度（メガバイト/秒）。

**mbps\_wr\_max= &lt;mbps&gt;**  
スロットリングなしの最大書き込みプール（メガバイト/秒）。

**media= &lt;cdrom| disk&gt; (デフォルト= disk)**  
ドライブのメディアタイプ。

**model= &lt;model&gt;**  
ドライブの報告されたモデル名（URLエンコード済み、最大40バイト）。

**replicate= &lt;boolean&gt; (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= &lt;ignore| report| stop&gt;**  
読み取りエラー時の動作。

**シリアル= &lt;serial&gt;**  
ドライブが報告したシリアル番号（URLエンコード済み、最大20バイト）。

**共有= &lt;boolean&gt; (デフォルト= 0)**  
このローカル管理ボリュームを全ノードで利用可能とマークします。



#### 警告

このオプションはボリュームを自動的に共有するものではなく、既に共有されていることを前提としています。

**size= &lt;ディスクサイズ&gt;**  
ディスクサイズ。これは純粹に情報提供用であり、効果はありません。

**snapshot= &lt;boolean&gt;**  
qemuのスナップショットモード機能を制御します。有効にすると、ディスクへの変更は一時的なものとなり、VMのシャットダウン時に破棄されます。

**ssd= &lt;boolean&gt;**  
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。

**werror= &lt;enospcl| ignore| report| stop&gt;**  
書き込みエラー時の動作。

**wwn= &lt;wwn&gt;**  
ドライブのワールドワイド名。16 バイトの 16 進文字列としてエンコードされ、先頭に 0x が付加されます。

**ipconfig[n]: [gw=&lt;GatewayIPv4&gt;] [,gw6=&lt;GatewayIPv6&gt;] [,ip=&lt;IPv4Format/CIDR&gt;;]  
[,ip6=&lt;IPv6Format/CIDR&gt;]**  
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ですが、同じタイプのIPを指定する必要があります。

IPアドレスにDHCPを使用するには特殊文字列`dhcp`を使用でき、この場合明示的なゲートウェイは指定しない。IPv6では特殊文字列`auto`を使用するとステートレス自動設定が有効になる。これには`cloud-init` 19.4以降が必要である。

`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`を使用します。

**gw= &lt;GatewayIPv4&gt;**

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= &lt;GatewayIPv6&gt;**

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= &lt;IPv4Format/CIDR&gt; (デフォルト= dhcp)**

CIDR 形式の IPv4 アドレス。

**ip6= &lt;IPv6Format/CIDR&gt; (デフォルト= dhcp)**

CIDR形式のIPv6アドレス。

**ivshmem: size=&lt;integer&gt; [,name=&lt;string&gt;]**

仮想マシン間共有メモリ。仮想マシン間、またはホストへの直接通信に有用。

**name= &lt;string&gt;**

ファイル名。`pve-shm-` が接頭辞として付加されます。デフォルトは VMID です。VM が停止されると削除されます。

**size= &lt;integer&gt; (1 - N)**

ファイルサイズ (MB単位)。

**keephugepages: &lt;boolean&gt; (デフォルト= 0)**

`hugepages`と併用します。有効にすると、VMシャットダウン後も`hugepages`が削除されず、以降の起動で使用可能になります。

**キーボード: &lt;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&gt;**

VNCサーバーのキーボードレイアウト。このオプションは通常不要であり、ゲストOS内で設定する方が望ましい場合が多い。

**kvm:** <boolean> (デフォルト= 1)

KVMハードウェア仮想化を有効/無効にします。

**localtime:** <boolean>

リアルタイムクロック (RTC) を現地時間に設定します。 ostype が Microsoft Windows OS を示している場合、デフォルトで有効になります。

**lock:** <lock> backup | clone | create | migrate | rollback | snapshot | snapshot-delete | suspended | suspending &gt;

仮想マシンのロック/ロック解除を行います。

**マシン:** [[**type**=]<マシンタイプ>] [,enable-s3=<1|0>] [,enable-s4=<1|0>] [,viommu=<intel|virtio>]

QEMUマシンを指定します。

**enable-s3=** <boolean>

S3電源状態を有効化します。マシンタイプ9.2+pve1以降ではデフォルトでfalse、それ以前はtrueです。

**enable-s4=** <boolean>

S4電源状態を有効化します。マシンタイプ9.2+pve1以降ではデフォルトでfalse、それ以前ではtrueです。

**type=** <マシンタイプ>

QEMUマシンタイプを指定します。

**viommu=** <intel| virtio>

ゲスト vIOMMU バリARIANTを有効化および設定します (Intel vIOMMU にはマシンタイプとして q35 の設定が必要です)。

**memory:** [**current**=]<integer>

メモリのプロパティ。

**現在の=** <integer> (16 - N) (デフォルト= 512)

VMのオンラインRAMの現在の量 (MiB単位)。バルーンデバイス使用時の最大利用可能メモリです。

**migrate\_downtime:** <number> (0 - N) (デフォルト= 0.1)

移行時の最大許容ダウンタイム (秒単位) を設定します。移行の最終段階で、新たに汚れたRAMの転送量が多すぎて収束できない場合、移行が収束できるまで制限値が段階的に自動的に増加します。

**migrate\_speed:** <integer> (0 - N) (デフォルト= 0)

移行の最大速度 (MB/s) を設定します。値0は制限なしです。

**name:** <string>

VMの名前を設定します。設定Webインターフェースでのみ使用されます。

**nameserver:** <string>

cloud-init: コンテナのDNSサーバーIPアドレスを設定します。searchdomainとnameserverの両方が設定されていない場合、作成時にホストの設定が自動的に使用されます。

```
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...]>]
[,&lt;model>=<macaddr>]
```

ネットワークデバイスを指定します。

#### **ブリッジ= <bridge>;**

ネットワークデバイスを接続するブリッジ。Proxmox VEの標準ブリッジはvmbr0と呼ばれます。

ブリッジを指定しない場合、DHCPおよびDNSサービスを提供するkvmユーザー（NAT接続）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サーバーはゲストに10.0.2.15から始まるアドレスを割り当てます。

#### **ファイアウォール= <boolean>;**

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= <boolean>;**

このインターフェースを切断するかどうか（プラグを抜くような状態）。

#### **macaddr= <XX:XX:XX:XX:XX:XX>;**

I/G（個別/グループ）ビットが設定されていない共通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>;
```

ネットワークカードモデル。virtioモデルはCPUオーバーヘッドが非常に低く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を使用するのが最適です。

#### **mtu= <integer>; (1 - 65520)**

ネットワークデバイスの強制MTU（VirtIOのみ）。1または空に設定するとブリッジMTUを使用

#### **queues= <integer>; (0 - 64)**

デバイスで使用するパケットキューの数。

#### **rate= <number>; (0 - N)**

浮動小数点数による mbps（メガバイト毎秒）単位のレート制限。

#### **tag= <integer>; (1 - 4094)**

このインターフェース上のパケットに適用するVLANタグ。

#### **trunks= <vlanid[,vlanid...]>;**

このインターフェイスを通過するVLANトランク。

#### **numa: <boolean>; (デフォルト= 0)**

NUMAを有効/無効にします。

**numa[n]: cpus=<id[-id];...> [,hostnodes=<id[-id];...> [,memory=<number>] [,policy=<bind|interleave|preferred>]]**  
NUMA トポロジ。

**cpus= <id[-id];...>;**  
このNUMAノードにアクセスするCPU。

**hostnodes= <id[-id];...>;**  
使用するホストNUMAノード。

**メモリ= <number>;**  
このNUMAノードが提供するメモリ量。

**policy= <bind| interleave| preferred>;**  
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します。

**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向けの特別な最適化/機能を有効にするために使用されます:

その他	未指定のOS
wxp	Microsoft Windows XP
w2k	Microsoft Windows 2000
w2k3	Microsoft Windows 2003
w2k8	Microsoft Windows 2008
vvista	Microsoft Windows Vista
win7	Microsoft Windows 7
win8	Microsoft Windows 8/2012/2012r2
win10	Microsoft Windows 10/2016/2019
win11	Microsoft Windows 11/2022/2025
l24	Linux 2.4 カーネル
l26	Linux 2.6 - 6.X カーネル
Solaris	Solaris/OpenSolaris/OpenIndiana カーネル

**parallel[n]: /dev/parport\<d+>|/dev/usb/lp\<d+>**  
ホストのパラレルデバイスをマップします (n は 0 から 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許可します。そのため、このようなマシンの移行は不可能になります - 特別な注意を払って使用してください。

---



## 注意

実験的！このオプションに関するユーザー報告の問題があります。

**保護: &lt;boolean&ampgt (デフォルト= 0)**

仮想マシンの保護フラグを設定します。これにより、仮想マシンの削除およびディスクの削除操作が無効化されます。

**reboot: &lt;boolean&ampgt (デフォルト= 1)**

再起動を許可します。0に設定すると、再起動時にVMが終了します。

```
rng0: [source=&lt;/dev/urandom|/dev/random|/dev/hwrng&ampgt [,max_bytes=&lt;integer&ampgt] [,period=&lt;integer&ampgt]
```

VirtIOベースの乱数生成器を設定します。

**max\_bytes= &lt;integer&ampgt (デフォルト= 1024)**

ゲストに注入されるエントロピーの最大バイト数（毎周期ミリ秒ごと）。制限を無効化するには

制限を無効にするには0を使用（潜在的に危険！）。

**period= &lt;integer&ampgt (デフォルト= 1000)**

毎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、ゲストが別の max\_bytes のエントロピーを取得できるようになります。

**source= &lt;/dev/hwrng| /dev/random| /dev/urandom&ampgt;**

ホスト上でエントロピー収集元のファイルを指定します。urandomの使用は実質的なセキュリティ低下を招きません。実際のエントロピーからシードされ、提供されるバイトはゲスト側でも実際のエントロピーと混合される可能性が高いからです。/dev/hwrngを使用すると、ホストのハードウェア乱数生成器（RNG）をゲストに透過的に渡せます。

```
sata[n]: [file=&lt;ボリューム&ampgt [,aio=&lt;ネイティブ|スレッド|io_uring&ampgt] [,backup=&lt;1|0&ampgt] [,bps=&lt;bps&ampgt] [,bps_max_length=&lt;秒&ampgt]
```

```
[,bps_rd=&lt;bps&ampgt][,bps_rd_max_length=&lt;seconds&ampgt] [,bps_wr=&lt;bps&ampgt] [,bps_wr_max_length=&lt;seconds&ampgt] [,cache=&lt;enum&ampgt]
```

```
[,detect_zeroes=&lt;1|0&ampgt][,discard=&lt;無視|有効&ampgt] [,フォーマット=&lt;列挙型&ampgt]
```

```
[,iops=&lt;iops&ampgt] [,iops_max=&lt;iops&ampgt] [,iops_max_length=&lt;秒&ampgt]
```

```
[,iops_rd=&lt;iops&ampgt][,iops_rd_max=&lt;iops&ampgt] [,iops_rd_max_length=&lt;seconds&ampgt]
```

```
[,iops_wr=&lt;iops&ampgt] [,iops_wr_max=&lt;iops&ampgt][,iops_wr_max_length=&lt;seconds&ampgt]
```

```
[,mbps=&lt;mbps&ampgt] [,mbps_max=&lt;mbps&ampgt] [,mbps_rd=&lt;mbps&ampgt]
```

```
[,mbps_rd_max=&lt;mbps&ampgt] [,mbps_wr=&lt;mbps&ampgt][,書き込み最大Mbps=<Mbps>] [,メディア
```

```
&lt;CD-ROM|ディスク&ampgt] [,レプリケーション=<1|0&ampgt] [,エラー処理=<無視|報告|停止&ampgt] [,シリアル
```

```
&lt;シリアル番号&ampgt] [,共有=<1|0&ampgt][,size=&lt;DiskSize&ampgt] [,snapshot=&lt;1|0&ampgt]
```

```
[,ssd=&lt;1|0&ampgt] [,werror=&lt;enum&ampgt] [,wwn=&lt;wwn&ampgt]
```

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から5)。

**aio= &lt;io\_uring| native| threads&ampgt**

使用するAIOタイプ。

**backup= &lt;boolean&gt;**

バックアップ作成時にドライブを含めるかどうか。

**bps= &lt;bps&gt;**

最大読み書き速度（バイト/秒）。

**bps\_max\_length= &lt;秒&gt;**

I/Oバーストの最大長（秒単位）。

**bps\_rd= &lt;bps&gt;**

最大読み取り速度（バイト/秒）。

**bps\_rd\_max\_length= &lt;秒&gt;**

読み取りI/Oバーストの最大長（秒単位）。

**bps\_wr= &lt;bps&gt;**

最大書き込み速度（バイト/秒）。

**bps\_wr\_max\_length= &lt;秒&gt;**

書き込みI/Oバーストの最大長（秒単位）。

**cache= &lt;directsync| none| unsafe| writeback| writethrough&gt;**

ドライブのキャッシングモード

**detect\_zeroes= &lt;boolean&gt;**

ゼロ書き込みの検出と最適化の試行を制御します。

**discard= &lt;ignore| on&gt;**

基盤となるストレージに discard/trim リクエストを渡すかどうかを制御します。

**file= &lt;volume&gt;**

ドライブのキャッシングボリューム。

**format= &lt;cloop| qcow| qcow2| qed| raw| vmdk&gt;**

ドライブのキャッシングファイルのデータ形式。

**iops= &lt;iops&gt;**

最大読み書きI/O（秒あたりの操作数）。

**iops\_max= &lt;iops&gt;**

最大スロットリングなしの読み書きI/Oプール（秒あたりの操作数）。

**iops\_max\_length= &lt;seconds&gt;**

I/Oバーストの最大長（秒単位）。

**iops\_rd= &lt;iops&gt;**

最大読み取りI/O（秒あたりの操作数）。

**iops\_rd\_max= &lt;iops&gt;**

最大スロットリングなし読み取りI/Oプール（秒あたりの操作数）。

**iops\_rd\_max\_length= &lt;seconds&gt;**

読み取り I/O バーストの最大長（秒単位）。

**iops\_wr= &lt;iops&gt;**  
最大書き込み I/O 操作数（秒あたり）。

**iops\_wr\_max= &lt;iops&gt;**  
最大スロットリングなし書き込みI/Oプール（秒あたりの操作数）。

**iops\_wr\_max\_length= &lt;seconds&gt;**  
秒単位での書き込みI/Oバーストの最大長。

**mbps= &lt;mbps&gt;**  
最大読み書き速度（メガバイト毎秒）。

**mbps\_max= &lt;mbps&gt;**  
最大スロットリングなしの読み書きプール（メガバイト毎秒）。

**mbps\_rd= &lt;mbps&gt;**  
最大読み取り速度（メガバイト/秒）。

**mbps\_rd\_max= &lt;mbps&gt;**  
最大スロットリングなし読み取りプール（メガバイト/秒）。

**mbps\_wr= &lt;mbps&gt;**  
最大書き込み速度（メガバイト/秒）。

**mbps\_wr\_max= &lt;mbps&gt;**  
スロットリングなしの最大書き込みプール（メガバイト/秒）。

**media= &lt;cdrom| disk&gt; (デフォルト=disk)**  
ドライブのメディアタイプ。

**replicate= &lt;boolean&gt; (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= &lt;無視| 報告| 停止&gt;**  
読み取りエラー処理。

**シリアル= &lt;serial&gt;**  
ドライブが報告したシリアル番号（URLエンコード済み、最大20バイト）。

**shared= &lt;boolean&gt; (デフォルト= 0)**  
このローカル管理ボリュームを全ノードで利用可能とマークします。



#### 警告

このオプションはボリュームを自動的に共有するものではなく、既に共有されていることを前提としています。

**size= &lt;DiskSize&gt;**  
ディスクサイズ。これは純粋に情報提供用であり、効果はありません。

**snapshot= &lt;boolean&gt;**  
qemuのスナップショットモード機能を制御します。有効化すると、ディスクへの変更は一時的なものとなり、VMのシャットダウン時に破棄されます。

**ssd= &lt;boolean&gt;**

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。

**werror= &lt;enospcl| ignore| report| stop&gt;**

書き込みエラー時の動作。

**wwn= &lt;wwn&gt;**

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` が付加されます。

**scsi[n]:[file=&lt;ボリューム&gt; [,aio=&lt;ネイティブ|スレッド|io\_uring&gt;] [,backup=&lt;1|0&gt;] [,bps=&lt;bps&gt;] [,bps\_max\_length=&lt;秒&gt;] [,bps\_rd=&lt;bps&gt;][,bps\_rd\_max\_length=&lt;seconds&gt;] [,bps\_wr=&lt;bps&gt;] [,bps\_wr\_max\_length=&lt;seconds&gt;] [,cache=&lt;enum&gt;] [,detect\_zeroes=&lt;1|0&gt;] [,discard=&lt;ignore|on&gt;][,フォーマット=&lt;enum&gt;] [,iops=&lt;iops&gt;] [,iops\_max=&lt;iops&gt;] [,iops\_max\_length=&lt;seconds&gt;] [,iops\_rd=&lt;iops&gt;] [,iops\_rd\_max=&lt;iops&gt;][,iops\_rd\_max\_length=&lt;seconds&gt;] [,iops\_wr=&lt;iops&gt;] [,iops\_wr\_max=&lt;iops&gt;] [,iops\_wr\_max\_length=&lt;seconds&gt;] [,iothread=&lt;1|0&gt;][,mbps=&lt;mbps&gt;] [,mbps\_max=&lt;mbps&gt;] [,mbps\_rd=&lt;mbps&gt;] [,mbps\_rd\_max=&lt;mbps&gt;][,mbps\_wr=&lt;mbps&gt;] [,mbps\_wr\_max=&lt;mbps&gt;][,media=&lt;cdrom|disk&gt;] [,product=&lt;product&gt;] [,queues=&lt;integer&gt;] [,replicate=&lt;1|0&gt;] [,error=&lt;ignore|report|stop&gt;] [,ro=&lt;1|0&gt;][,scsiblock=&lt;1|0&gt;] [,serial=&lt;serial&gt;] [,shared=&lt;1|0&gt;] [,size=&lt;DiskSize&gt;] [,snapshot=&lt;1|0&gt;] [,ssd=&lt;1|0&gt;][,ベンダー=&lt;ベンダー&gt;] [,ワーエラー=&lt;列挙型&gt;] [,WWN=&lt;WWN&gt;]**

ボリュームをSCSIハードディスクまたはCD-ROMとして使用 (nは0から30)。

**aio= &lt;io\_uring| native| threads&gt;**

使用するAIOタイプ。

**= のバックアップ &lt;boolean&gt;**

バックアップ作成時にドライブを含めるかどうか。

**bps= &lt;bps&gt;**

最大読み書き速度 (バイト/秒)。

**bps\_max\_length= &lt;秒&gt;**

I/Oバーストの最大長 (秒単位)。

**bps\_rd= &lt;bps&gt;**

最大読み取り速度 (バイト/秒)。

**bps\_rd\_max\_length= &lt;秒&gt;**

読み取りI/Oバーストの最大長 (秒単位)。

**bps\_wr= &lt;bps&gt;**

最大書き込み速度 (バイト/秒)。

**bps\_wr\_max\_length= &lt;秒&gt;**

書き込みI/Oバーストの最大長 (秒単位)。

```
cache= &lt;directsync| none| unsafe| writeback| writethrough&gt;
      ドライブのキャッシングモード

detect_zeroes= &lt;boolean&gt;
      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します。

discard= &lt;無視| オン&gt;
      ディスクカード/トリム要求を基盤ストレージに渡すかどうかを制御します。

file= &lt;volume&gt;
      ドライブのバックキングボリューム。

format= &lt;cloop| qcow| qcown| qed| raw| vmdk&gt;
      ドライブのバックキングファイルのデータ形式。

iops= &lt;iops&gt;
      最大読み書きI/O（秒あたりの操作数）。

iops_max= &lt;iops&gt;
      最大スロットリングなしの読み書きI/Oプール（秒あたりの操作数）。

iops_max_length= &lt;seconds&gt;
      I/Oバーストの最大長（秒単位）

iops_rd= &lt;iops&gt;
      最大読み取りI/O（秒あたりの操作数）。

iops_rd_max= &lt;iops&gt;
      スロットリングなしの最大読み取りI/Oプール（秒あたりの操作数）。

iops_rd_max_length= &lt;seconds&gt;
      読み取りI/Oバーストの最大長（秒単位）。

iops_wr= &lt;iops&gt;
      最大書き込みI/O操作数（秒あたり）。

iops_wr_max= &lt;iops&gt;
      最大スロットリングなし書き込みI/Oプール（秒あたりの操作数）。

iops_wr_max_length= &lt;seconds&gt;
      秒単位での書き込みI/Oバーストの最大長。

iothread= &lt;boolean&gt;
      このドライブでiothreadsを使用するかどうか

mbps= &lt;mbps&gt;
      最大読み書き速度（メガバイト毎秒）。

mbps_max= &lt;mbps&gt;
      最大スロットリングなし読み書きプール（メガバイト/秒）。

mbps_rd= &lt;mbps&gt;
      最大読み取り速度（メガバイト/秒）。
```

**mbps\_rd\_max= &lt;mbps&gt;**  
最大スロットリングなし読み取りプール（メガバイト/秒）。

**mbps\_wr= &lt;mbps&gt;**  
最大書き込み速度（メガバイト/秒）。

**mbps\_wr\_max= &lt;mbps&gt;**  
スロットリングなしの最大書き込みプール（メガバイト/秒）。

**media= &lt;cdrom| disk&gt; (デフォルト=disk)**  
ドライブのメディアタイプ。

**product= &lt;product&gt;**  
ドライブの製品名（最大16バイト）。

**キュー= &lt;integer&gt; (2 - N)**  
キューの数。

**replicate= &lt;boolean&gt; (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= &lt;無視 | report | 停止&gt;**  
読み取りエラー時の動作。

**ro= &lt;boolean&gt;**  
ドライブが読み取り専用かどうか。

**scsiblock= &lt;boolean&gt; (デフォルト= 0)**  
ホストブロックデバイスの完全パススルーにSCSIブロックを使用するかどうか



#### 警告

ホストのメモリ不足やメモリ断片化と組み合わさるとI/Oエラーを引き起こす可能性があります

**serial= &lt;serial&gt;**  
ドライブの報告されたシリアル番号、URLエンコード済み、最大20バイト。

**共有= &lt;boolean&gt; (デフォルト= 0)**  
このローカル管理ボリュームを、すべてのノードで利用可能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ものではなく、すでに共有されていることを前提としています。

**size= &lt;ディスクサイズ&gt;**  
ディスクサイズ。これは情報提供のみであり、効果はありません。

**snapshot= &lt;boolean&gt;**  
qemuのスナップショットモード機能を制御します。有効化すると、ディスクへの変更は一時的なものとなり、VMのシャットダウン時に破棄されます。

**ssd= &lt;boolean&gt;**

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。

**ベンダー= &lt;vendor&gt;**

ドライブのベンダー名（最大8バイト）。

**werror= &lt;enospcl| ignore| report| stop&gt;**

書き込みエラーの動作。

**wwn= &lt;wwn&gt;**

ドライブのワールドワイドネーム。16バイトの16進文字列としてエンコードされ、先頭に0xが付加されます。

**scsihw: &lt;lsi| lsi53c810| megasas| pvscsi| virtio-scsi-pci | virtio-scsi-single&gt; (デフォルト = lsi)**

SCSIコントローラモデル

**searchdomain: &lt;string&gt;**

cloud-init: コンテナのDNS検索ドメインを設定します。searchdomainもnameserverも設定されていない場合、作成時にはホストの設定が自動的に使用されます。

**serial[n]: (/dev/.+|socket)**

VM内にシリアルデバイスを作成（nは0～3）、ホスト側のシリアルデバイス（例：/dev/ptyS0）をバススルー、またはホスト側にUNIXソケットを作成（qmターミナルでターミナル接続を開く）。

---

**注意**

ホストのシリアルデバイスをバススルーする場合、そのようなマシンの移行は不可能になります。特別な注意を払って使用してください。

---

**注意**

実験的！このオプションに関する問題がユーザーから報告されています。

---

**shares: &lt;integer&gt; (0 - 50000) (デフォルト = 1000)**

自動バルーニング用のメモリシェア量。数値が大きいほど、このVMが割り当てられるメモリが増加します。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す。ゼロを使用すると自動バルーニングが無効になります。自動バルーニングはpvestatdによって実行されます。

**smbios1: [base64=&lt;1|0&gt;] [,family=&lt;Base64エンコード文字列&gt;] [,manufacturer=&lt;Base64エンコード文字列&gt;] [,product=&lt;Base64エンコード文字列&gt;] [,serial=&lt;Base64エンコード文字列&gt;] [,sku=&lt;Base64エンコード文字列&gt;] [,uuid=&lt;UUID&gt;] [,version=&lt;Base64エンコード文字列&gt;]**

SMBIOS タイプ1フィールドを指定します。

**base64= &lt;boolean&gt;**

SMBIOS値がBase64エンコードされていることを示すフラグ

---

= ファミリー &lt;Base64エンコードされた文字列&gt;

SMBIOS1ファミリ文字列を設定します。

**manufacturer= &lt;Base64 エンコードされた文字列&gt;**

SMBIOS1 メーカーを設定します。

**製品= &lt;Base64エンコード文字列&gt;**

SMBIOS1 プロダクト ID を設定します。

**シリアル= &lt;Base64エンコードされた文字列&gt;**

SMBIOS1 シリアル番号を設定します。

**sku= &lt;Base64 エンコードされた文字列&gt;**

SMBIOS1 SKU 文字列を設定します。

**uuid= &lt;UUID&gt;**

SMBIOS1 UUIDを設定します。

**version= &lt;Base64 エンコードされた文字列&gt;**

SMBIOS1のバージョンを設定します。

**smp: &lt;integer&gt; (1 - N) (デフォルト= 1)**

CPUの数。代わりにオプション -sockets を使用してください。

**ソケット数: &lt;integer&gt; (1 - N) (デフォルト= 1)**

CPUソケットの数。

**spice\_enhancements: [foldersharing=&lt;1|0&gt;]  
[,videostreaming=&lt;off|all|filter&gt;]**

SPICE の追加機能設定。

**foldersharing= &lt;boolean&gt; (デフォルト= 0)**

SPICE経由のフォルダ共有を有効化します。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= &lt;all| filter| off&gt; (デフォルト= off)**

ビデオストリーミングを有効化。検出されたビデオストリームに対して圧縮を使用。

**sshkeys: &lt;string&gt;**

cloud-init: 公開SSHキーの設定 (1行に1キー、OpenSSH形式)。

**startdate: (現在 | YYYY-MM-DD | YYYY-MM-DDTHH:MM:SS) (デフォルト= 現在)**

リアルタイムクロックの初期日付を設定します。日付の有効な形式は次の通りです：'now' または 2006-06-17T16:01:21 または

2006-06-17。

**起動: `[[order=]d+] [.up=d+] [.down=d+]`**

起動およびシャットダウン時の動作を定義します。Orderは起動順序を定義する非負の数値です。シャットダウン時は逆順で実行されます。さらに、アップダウン遅延 (秒単位) を設定可能で、次のVM起動/停止までの待機時間を指定します。

**tablet: <boolean> (デフォルト= 1)**

USBタブレットデバイスの有効化/無効化。このデバイスは通常、VNCでマウスの絶対位置指定を可能にするために必要です。無効化すると、通常VNCクライアントではマウスが同期しなくなります。1台のホストで多数のコンソール専用ゲストを実行している場合、コンテキストスイッチを節約するために無効化を検討できます。spiceを使用している場合 (`qm set <vmid> --vga qxl`) 、デフォルトで無効化されます。

**タグ: <string>**

VMのタグ。これはメタ情報のみです。

**tdf: <boolean> (デフォルト= 0)**

タイムドリフト修正の有効/無効設定。

**template: <boolean> (デフォルト= 0)**

テンプレートの有効/無効設定。

**tpmstate0: [<file>] <volume> [,size=<disksize>] [,version=<v1.2|v2.0>]**

TPM状態を保存するためのディスクを設定します。フォーマットは固定でRAWです。

**file= <volume>**

ドライブのバックアップボリューム。

**size= <disksize>**

ディスクサイズ。これは情報提供のみであり、効果はありません。

**version= <v1.2| v2.0> (デフォルト= v1.2)**

TPMインターフェースのバージョン。v2.0が最新であり推奨される。後から変更できない点に注意。

**unused[n]: [<file>] <volume>**

未使用ボリュームへの参照。内部で使用されるため、手動で変更しないでください。

**ファイル= <volume>**

ドライブのバックアップボリューム。

**usb[n]: [[<host>] <HOSTUSBDEVICE|spice>] [,mapping=<mapping-id>] [,usb3=<1|0>]**

USB デバイスを設定します (n は 0 から 4 です。マシンバージョンが 7.1 以上で ostype が l26 または Windows 7 以上の場合、n は 14 まで可能です)。

**host= <HOSTUSBDEVICE|spice>**

ホストUSBデバイスまたはポート、または値spice。HOSTUSBDEVICEの構文は次の通り：

'bus-port(.port)\*' (10進数) または 'ベンダーID:製品ID' (16進数) または  
'spice'

既存のUSBデバイス一覧は `lsusb -t` コマンドで確認できます。

#### 注意

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許可します。そのため、このようなマシンの移行は不可能になります。特別な注意を払って使用してください。

`spice` 値を使用すると、`spice` 用の USB リダイレクトデバイスを追加できます。これかマッピングキーのいずれかを設定する必要があります。

**mapping= &lt;mapping-id&gt;**

クラスタ全体のマッピングIDです。これかデフォルトキー`host`のいずれかを設定する必要があります。

**usb3= &lt;boolean&gt; (デフォルト= 0)**

指定されたホストオプションがUSB3デバイスまたはポートであるかを指定します。最新のゲスト（マシンバージョン≥7.1かつostype l26かつ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ます）。

**vcpus: &lt;integer&gt; (1 - N) (デフォルト= 0)**

ホットプラグされた vcpus の数。

**vga: [[type=] &lt;enum&gt;] [,clipboard=&lt;vnc&gt;] [,memory=&lt;integer&gt;]**

VGAハードウェアを設定します。高解像度モード（1280x1024x16以上）を使用する場合は、vgaメモリオプションの値を増やす必要がある場合があります。QEMU 2.9以降、デフォルトのVGA表示タイプは、一部のWindowsバージョン（XPおよびそれ以前）が`cirrus`を使用する以外、すべてのOSタイプで`std`です。qxlオプションはSPICEディスプレイサーバーを有効化します。win\* OSでは独立したディスプレイ数を指定可能で、Linuxゲストは自身でディスプレイを追加できます。グラフィックカードなしでシリアルデバイスを端末として使用することも可能です。

**clipboard= &lt;vnc&gt;**

特定のクリップボードを有効化します。設定しない場合、ディスプレイタイプに応じてSPICEクリップボードが追加されます。VNCクリップボードの移行はまだサポートされていません！

**memory= &lt;integer&gt; (4 - 512)**

VGAメモリ（MiB単位）を設定します。シリアル表示では効果ありません。

**= &lt;cirrus| none| qxl1| qxl2| qxl3| qxl4| serial0 | serial1 | serial2 | serial3 | std | virtio | virtio-gl | vmware&gt; (デフォルト= std)**

VGAタイプを選択してください。タイプ`cirrus`の使用は推奨されません。

```
virtio[n]: [file=<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=<秒>;]  
[,bps_rd=<bps>;][,bps_rd_max_length=<seconds>;] [,bps_wr=<bps>;]  
[,bps_wr_max_length=<seconds>;] [,cache=<enum>;] [,detect_zeroes=<1|0>;]  
[,discard=<ignore|on>;][,format=<enum>;] [,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>;][,書き込み_最大_mbps=<mbps>;] [,メディア  
=<cdrom|disk>;] [,レプリケーション=<1|0>;] [,エラー処理=<無視|報告|停止>;] [,読み取り専  
用=<1|0>;][,シリアル=<シリアル>;] [,共有=<1|0>;] [,サイズ=<ディスクサイズ>;] [,ス  
ナップショット=<1|0>;] [,書き込みエラー=<列挙型>;]
```

ボリュームをVIRTIOハードディスクとして使用 (nは0から15)。

**aio= <io\_uring| native| threads>;**  
使用するAIOタイプ。

**backup= <boolean>;**  
バックアップ作成時にドライブを含めるかどうか。

**bps= <bps>;**  
最大読み書き速度 (バイト/秒)。

**bps\_max\_length= <秒>;**  
I/Oバーストの最大長 (秒単位)。

**bps\_rd= <bps>;**  
最大読み取り速度 (バイト/秒)。

**bps\_rd\_max\_length= <秒>;**  
読み取りI/Oバーストの最大長 (秒単位)。

**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 (秒単位)。

**cache= <directsync| none| unsafe| writeback| writethrough>;**  
ドライブのキャッシングモード

**detect\_zeroes= <boolean>;**  
ゼロ書き込みの検出と最適化の試行を制御します。

**discard= <ignore| on>;**  
基盤となるストレージに discard/trim リクエストを渡すかどうかを制御します。

```
file= &lt;volume&gt;
      ドライブのバックингボリューム。

format= &lt;cloop| qcow| qcow2| qed| raw| vmdk&gt;
      ドライブのバックングファイルのデータ形式。

iops= &lt;iops&gt;
      最大読み書きI/O (秒あたりの操作数)。

iops_max= &lt;iops&gt;
      最大スロットリングなしの読み書きI/Oプール (秒あたりの操作数)。

iops_max_length= &lt;seconds&gt;
      I/Oバーストの最大長 (秒単位)。

iops_rd= &lt;iops&gt;
      最大読み取りI/O (秒あたりの操作数)。

iops_rd_max= &lt;iops&gt;
      最大スロットリングなし読み取りI/Oプール (秒あたりの操作数)。

iops_rd_max_length= &lt;seconds&gt;
      読み取り I/O バーストの最大長 (秒単位)。

iops_wr= &lt;iops&gt;
      最大書き込み I/O 操作数 (秒あたり)。

iops_wr_max= &lt;iops&gt;
      最大スロットリングなし書き込みI/Oプール (秒あたりの操作数)。

iops_wr_max_length= &lt;seconds&gt;
      秒単位での書き込みI/Oバーストの最大長。

iothread= &lt;boolean&gt;
      このドライブでiothreadsを使用するかどうか

mbps= &lt;mbps&gt;
      最大読み書き速度 (メガバイト/秒単位)。

mbps_max= &lt;mbps&gt;
      最大スロットリングなし読み書きプール (メガバイト毎秒)。

mbps_rd= &lt;mbps&gt;
      最大読み取り速度 (メガバイト/秒)。

mbps_rd_max= &lt;mbps&gt;
      最大スロットリングなし読み取りプール (メガバイト/秒)。

mbps_wr= &lt;mbps&gt;
      最大書き込み速度 (メガバイト/秒)。

mbps_wr_max= &lt;mbps&gt;
      スロットリングなしの最大書き込みプール (メガバイト/秒)。
```

**media= &lt;cdrom| disk&gt; (デフォルト= disk)**

ドライブのメディアタイプ。

**replicate= &lt;boolean&gt; (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= &lt;ignore| report| stop&gt;**

読み取りエラー時の動作。

#### =読み取りエラーの動作。

ドライブが読み取り専用かどうか。

**serial= &lt;serial&gt;**

ドライブの報告されたシリアル番号 (URLエンコード済み、最大20バイト)。

**shared= &lt;boolean&gt; (デフォルト= 0)**

このローカル管理ボリュームを全ノードで利用可能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ものではなく、既に共有されていることを前提としています。

**size= &lt;ディスクサイズ&gt;**

ディスクサイズ。これは純粋に情報提供用であり、効果はありません。

**snapshot= &lt;boolean&gt;**

qemuのスナップショットモード機能を制御します。有効化されている場合、ディスクへの変更は一時的なものとなり、VMのシャットダウン時に破棄されます。

**werror= &lt;enospcl| ignore| report| stop&gt;**

書き込みエラー時の動作。

**virtiofs[n]: [dirid=] &lt;mapping-id&gt; [,cache=&lt;enum&gt;] [,direct-io=&lt;1|0&gt;] [,expose-acl=&lt;1|0&gt;] [,expose-xattr=&lt;1|0&gt;]**

ホストとゲスト間でディレクトリを共有するためのVirtio-fs設定。

**cache= &lt;always| auto| metadata| never&gt; (デフォルト= auto)**

ファイルシステムが使用するキャッシングポリシー (auto、always、metadata、never)。

**direct-io= &lt;boolean&gt; (デフォルト= 0)**

ゲストアプリケーションから渡されたO\_DIRECTフラグを尊重する。

**dirid= &lt;mapping-id&gt;**

ゲストと共有するディレクトリマッピングの識別子。VM内部のマウントタグとしても使用されます。

**expose-acl= &lt;boolean&gt; (デフォルト= 0)**

このマウントに対するPOSIX ACLのサポートを有効化します (有効なACLはxattrを意味します)。

```
expose-xattr= &lt;boolean&ampgt (デフォルト= 0)
```

このマウントに対する拡張属性のサポートを有効にします。

```
vmgenid: &lt;UUID&ampgt (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（スナップショットの実行やテンプレートからの作成など）で実行された際に、ゲスト OS に通知することが可能になります。ゲストOSは変更を検知し、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、乱数生成器を再初期化したりするなど、適切な対応を取ることができます。自動生成はAPI/CLIのcreateまたはupdateメソッド経由でのみ機能し、設定ファイルの手動編集時には機能しない点に注意してください。

```
vmstatestorage: &lt;ストレージID&ampgt;
```

VM状態ボリューム/ファイルのデフォルトストレージ。

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。ゲストアクションにより有効化されると、ゲスト内部のエージェントが定期的にウォッチドッグをポーリングする必要があります。さもないと、ウォッチドッグはゲストをリセット（または指定されたアクションを実行）します。

```
action= &lt;debug| none| pause| poweroff| reset| shutdown&ampgt;
```

起動後、ゲストがウォッチドッグを時間内にポーリングできなかった場合に実行するアクション。

```
model= &lt;i6300esb| ib700&ampgt (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 ロック

オンライン移行、スナップショット、バックアップ (vzdump) は、影響を受ける仮想マシン上で互換性のない同時操作を防ぐためにロックを設定します。場合によっては、このようなロックを手動で解除する必要がある場合があります（例：停電後）。

```
# qm unlock &lt;vmid&ampgt;
```



### 注意

ロックを設定した操作が実行されなくなったことを確認した場合にのみ、そ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、完全仮想化マシン（VM）に代わる軽量な選択肢です。コンテナは、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、実行されるホストシステムのカーネルを利用します。これにより、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ます。

コンテナの実行コストは低く、通常は無視できる程度です。ただし、考慮すべきいくつかの欠点があります：

- 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。FreeBSDやMicrosoft Windowsなどの他のOS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は制限する必要があります。そのため、コンテナは独自の分離されたネームスペースで実行されます。さらに、コンテナ内では一部のシステムコール（ユーザー空間からLinuxカーネルへの要求）が許可されません。

Proxmox VEは基盤となるコンテナ技術としてLinuxコンテナ（LXC）を採用しています。「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、クラスタ設定を認識し、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利用できます。Proxmox VEファイアウォールの使用や、HAフレームワークを用いたコンテナ管理も可能です。

私たちの主な目標は、VMを使用する利点を提供しつつ、追加のオーバーヘッドを伴わない環境を提供することです。これは、Proxmoxコンテナが「アプリケーションコンテナ」ではなく「システムコンテナ」に分類されることを意味します。

### 注記

アプリケーションコンテナ（例：Dockerイメージ）を実行したい場合は、Proxmox QEMU VM内で実行することを推奨します。これにより、アプリケーションコンテナ化の利点をすべて享受しつつ、ホストからの強力な分離やライブマイグレーションといったVMが提供するメリットも得られます。これらはコンテナ単独では実現不可能です。

## 11.1 技術概要

- LXC (<https://linuxcontainers.org/>)

- Proxmox VEのグラフィカルWebユーザーインターフェース（GUI）に統合済み
- 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*
- リソース分離と制限のための制御グループ（*cgroups*）
- セキュリティ強化のための*AppArmor*と*seccomp*
- 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の使用
- 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 サポート対象ディストリビューション

公式サポート対象ディストリビューションの一覧は以下をご覧ください。

以下のディストリビューション向けテンプレートは、当社のリポジトリから入手可能です。[pveam](#) ツールまたはグラフィカルユーザーインターフェースを使用してダウンロードできます。

### 11.2.1 Alpine Linux

Alpine Linuxは、musl libcとbusyboxを基盤とした、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以下を参照してください:

<https://alpinelinux.org/releases/>

### 11.2.2 Arch Linux

Arch Linuxは、シンプルさを追求する軽量で柔軟なLinux®ディストリビューションです。

— <https://archlinux.org/>

Arch Linuxはローリングリリースモデルを採用しています。詳細はWikiを参照してください: [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、安定性・予測可能性・管理性・再現性に優れた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以下を参照してください：

[https://en.wikipedia.org/wiki/CentOS#End-of-support\\_schedule](https://en.wikipedia.org/wiki/CentOS#End-of-support_schedule)

#### AlmaLinux

オープンソースでコミュニティ所有・運営の、永続的に無料のエンタープライズ向けLinuxディストリビューション。長期的な安定性に重点を置き、堅牢な本番環境向けプラットフォームを提供します。AlmaLinux OSはRHEL®およびStream以前のCentOSと1:1でバイナリ互換です。

— <https://almalinux.org>

現在サポートされているリリースについては以下を参照:

<https://en.wikipedia.org/wiki/AlmaLinux#Releases>

#### Rocky Linux

Rocky Linuxは、ダウンストリームパートナーが方向転換した今、米国を代表するエンタープライズ向けLinuxディストリビューションと100%バグ単位で互換性を保つよう設計されたコミュニティ主導のエンタープライズOSです。

— <https://rockylinux.org>

現在サポートされているリリースについては以下を参照: [https://en.wikipedia.org/wiki/Rocky\\_Linux#Releases](https://en.wikipedia.org/wiki/Rocky_Linux#Releases)

### 11.2.4 Debian

Debianは、Debianプロジェクトによって開発・保守されているフリーのオペレーティングシステムです。ユーザーのニーズに応える数千ものアプリケーションを備えたフリーのLinuxディストリビューションです。

— <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以下を参照: <https://www.debian.org/releases/stable/releasenotes>

## 11.2.5 Devuan

Devuan GNU+Linux は systemd を排除した Debian のフォークであり、不要な依存関係を回避し Init Freedom を保証することで、ユーザーがシステム制御を取り戻すことを可能にします。

— <https://www.devuan.org>

現在サポートされているリリースについては以下を参照:

<https://www.devuan.org/os/releases>

## 11.2.6 Fedora

Fedora は、ハードウェア、クラウド、コンテナ向けの革新的で自由なオープンソースプラットフォームを構築し、ソフトウェア開発者やコミュニティメンバーがユーザー向けにカスタマイズされたソリューションを構築できるようにします。

— <https://getfedora.org>

現在サポートされているリリースについては、

<https://fedoraproject.org/wiki/Releases> を参照してください。

## 11.2.7 Gentoo

高度に柔軟なソースベースの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採用しています。

## 11.2.8 OpenSUSE

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推奨ディストリビューション。

— <https://www.opensuse.org>

現在サポートされているリリースについては以下を参照:

<https://get.opensuse.org/leap/>

## 11.2.9 Ubuntu

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向けのモダンなオープンソースLinux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ついては以下を参照:

<https://wiki.ubuntu.com/Releases>

## 11.3 コンテナイメージ

コンテナイメージ（別名「テンプレート」または「アプライアンス」）は、コンテナを実行するために必要なすべての要素を含むtarアーカイブです。

Proxmox VE 自体は、最も一般的な Linux ディストリビューション向けに様々な基本テンプレートを提供しています。これらは GUI または pveam (Proxmox VE Appliance Manager の略称) 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可能です。

利用可能なテンプレートのリストは、pve-daily-update タイマーを通じて毎日更新されます。手動で更新をトリガーするには、以下を実行します：

```
# pveam update
```

利用可能なイメージの一覧を表示するには以下を実行します：

```
# pveam available
```

この大規模なリストは、関心のあるセクションを指定することで絞り込みます。例えば基本システムなど

イメージ:

### 利用可能なシステムイメージの一覧

```
# pveam available --section system
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 システ
ム          debian-10-standard_10.7-1_amd64.tar.gz system    devuan-3.0-
standard_3.0_amd64.tar.gz システム      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 システム
                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 systemubuntu-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、それらをストレージのいずれかにダウンロードする必要があります。どのストレージにダウンロードすべきか不明な場合は、その目的のためにローカルの命名済みストレージを使用できます。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用したコンテナの作成準備が整いました。ストレージ

local にあるダウンロード済みイメージは以下で一覧表示できます:

```
# pveam list local
```

### ヒント

Proxmox VEのWeb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 はこれらを使用して新しいコンテナを作成します。例:

```
# pct create 999 local:vztmp1/debian-10.0-standard_10.0-1_amd64.tar.gz
```

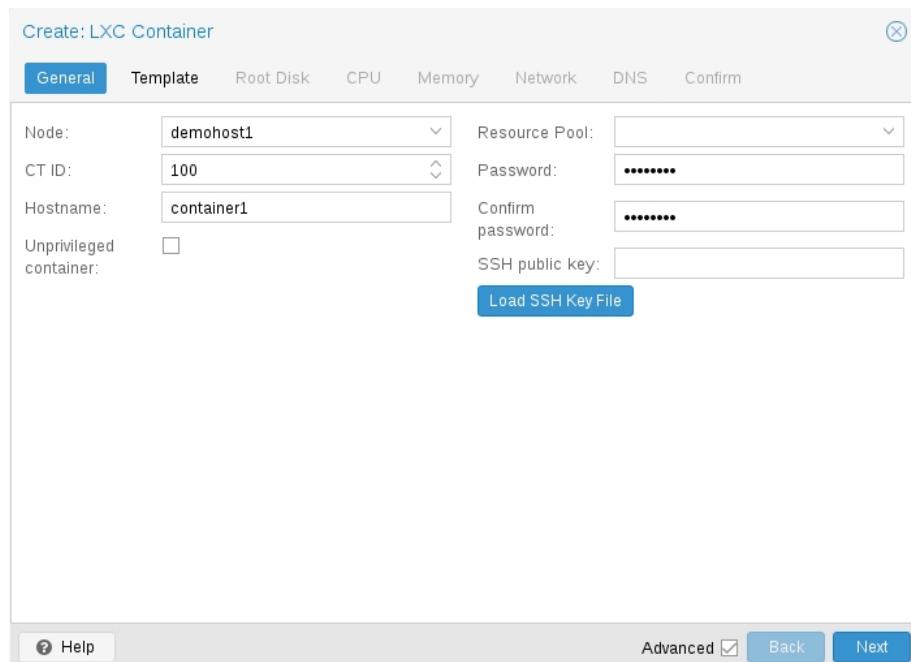
上記のコマンドは完全な Proxmox VE ボリューム識別子を表示します。これにはストレージ名が含まれており、他のほとんどの Proxmox VE コマンドで使用可能

```
# pveam remove local:vztmp1/debian-10.0-standard_10.0-1_amd64.tar.gz
```

です。例えば後でそのイメージを削除するには:

## 11.4 コンテナ設定

### 11.4.1 一般設定



コンテナの一般設定には以下が含まれます

- **ノード**: コンテナが実行される物理サーバー
- **CT ID**: このProxmox VEインストール環境内でコンテナを識別するための一意の番号
- **ホスト名**: コンテナのホスト名
- **リソースプール**: コンテナと仮想マシンを論理的にグループ化したもの
- **パスワード**: コンテナの root パスワード
- **SSH公開鍵**: SSH経由でrootアカウントに接続するための公開鍵
- **非特権コンテナ**: 作成時に特権コンテナと非特権コンテナのどちらを作成するかを選択できます。

## 非特権コンテナ

非特権コンテナは、ユーザー名前空間と呼ばれる新しいカーネル機能を利用します。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ます。これは、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（コンテナからの脱出、リソースの悪用など）が、ランダムな非特権ユーザーに影響を与え、LXCの問題ではなく一般的なカーネルのセキュリティバグとなることを意味します。LXCチームは、非特権コンテナは設計上安全であると考えています。

これは新規コンテナ作成時のデフォルトオプションです。

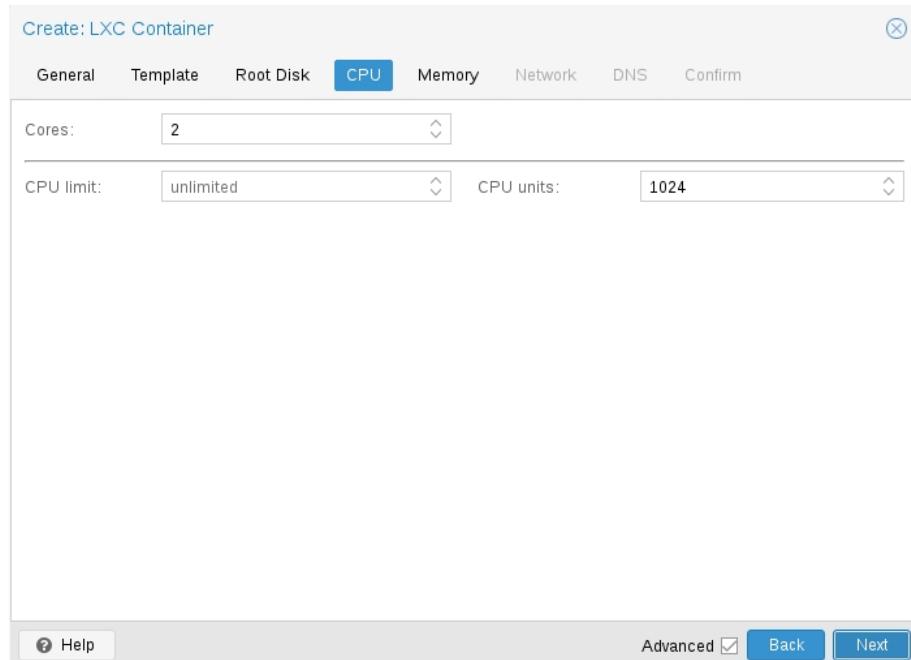
### 注意

コンテナが `systemd` を init システムとして使用する場合、コンテナ内で実行される `systemd` のバージョンは 220 以上である必要があることに注意してください。

## 特権付きコンテナ

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、および Linux カーネルネームスペースを使用して実現されています。LXC チームは、この種のコンテナは安全ではないと考えており、新しいコンテナエスケープエクスプロイトを CVE や迅速な修正に値するセキュリティ問題とは見なしていません。そのため、特権コンテナは信頼できる環境でのみ使用すべきです。

### 11.4.2 CPU



コンテナ内で可視化されるCPUの数を制限するには、coresオプションを使用します。これはLinuxのcpuset cgroup（制御グループ）を用いて実装されています。pvestatd内の特別なタスクが、稼働中のコンテナを利用可能なCPUに定期的に分散させようと試みます。割り当てられたCPUを確認するには、次のコマンドを実行します:

```
# pct cpusets
-----
102:          6 7
```

```
105:          2 3 4 5
108:          0 1
-----
```

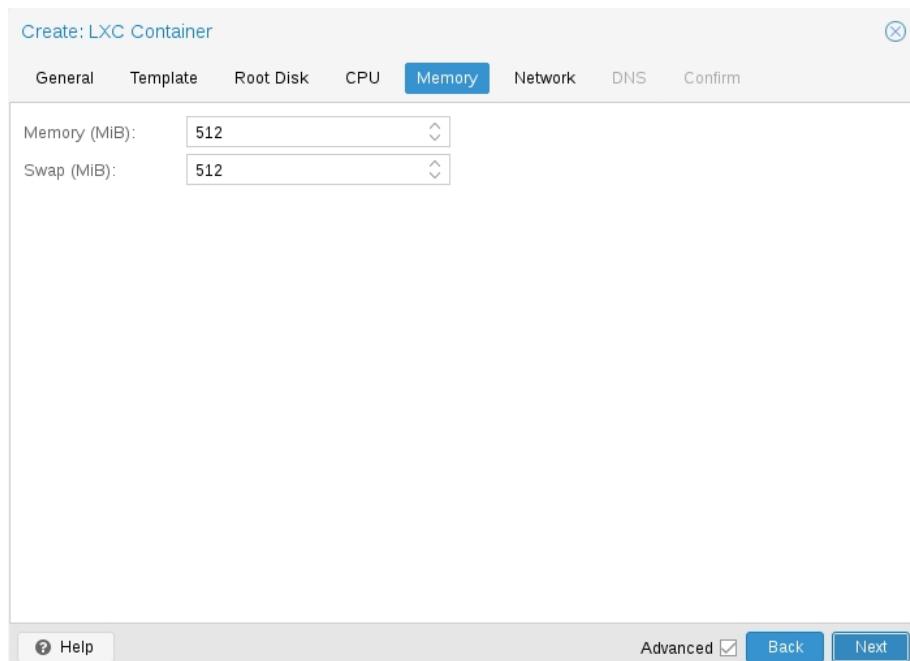
コンテナはホストカーネルを直接使用します。コンテナ内のすべてのタスクはホストCPUスケジューラによって処理されます。Proxmox VEはデフォルトでLinux CFS（完全公平スケジューラ）スケジューラを使用しており、追加の帯域幅制御オプションを備えています。

**cpulimit:** このオプションを使用すると、割り当てられたCPU時間をさらに制限できます。これは浮動小数点数であるため、コンテナに2つのコアを割り当てつつ、全体のCPU消費を0.5コアに制限することも完全に有効です。

cores: 2  
cpulimit: 0.5

**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す。数値が大きいほど、このコンテナが割り当てられるCPU時間が増加します。数値は他のすべての実行中のコンテナの重みに対する相対値です。デフォルトは100（ホストがレガシーCGROUP v1を使用している場合は1024）です。この設定を使用して特定のコンテナを優先させることができます。

### 11.4.3 メモリ

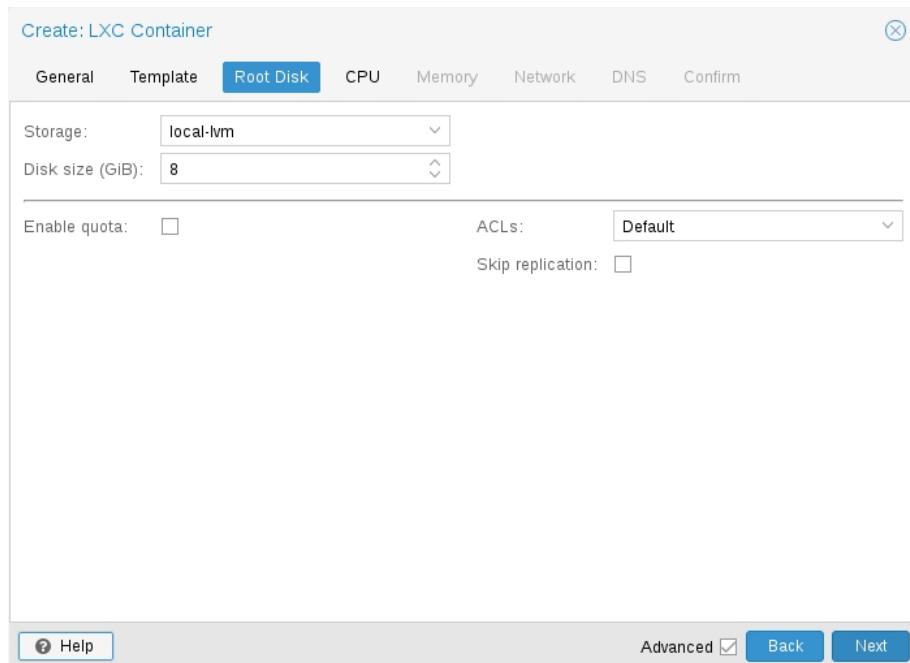


コンテナのメモリはcgroupメモリコントローラを使用して制御されます。

**メモリ:** メモリ使用量の全体的な制限を設定します。これはmemory.limit\_in\_bytes cgroup設定に相当します。

**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します。これはmemory.memsw.limit\_in\_bytes cgroup設定に対応し、両値（メモリ＋スワップ）の合計に設定されます。

## 11.4.4 マウントポイント



ルートマウントポイントは rootfs プロパティで設定されます。最大 256 個の追加マウントポイントを設定可能です。対応するオプションは mp0 から mp255 と呼ばれます。以下の設定を含めることができます:

```
rootfs: [volume=<ボリューム>; ,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=<DiskSize>]
```

ボリュームをコンテナのルートとして使用します。すべてのオプションの詳細な説明は以下を参照してください。

```
mp[n]: [volume=<ボリューム名>; ,mp=<パス>] [,acl=<1|0>] [,backup=<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=<DiskSize>]
```

ボリュームをコンテナのマウントポイントとして使用します。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= <boolean>;**  
ACLサポートを明示的に有効化または無効化します。

**backup= <boolean>;**  
バックアップにマウントポイントを含めるかどうか（ボリュームのマウントポイントのみに使用されます）。

**mountoptions= <opt[;opt...]>;**  
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めてはいけません。

**quota= &lt;boolean&gt;**  
コンテナ内でユーザークォータを有効化 (zfsサブボリュームではサポートされません)

**replicate= &lt;boolean&gt; (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= &lt;boolean&gt;**  
読み取り専用マウントポイント

**共有= &lt;boolean&gt; (デフォルト= 0)**  
このボリューム以外のマウントポイントを、すべてのノードで利用可能としてマークします。



#### 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ものではなく、既に共有されていることを前提としています。

**size= &lt;DiskSize&gt;**  
ボリュームサイズ (読み取り専用値)。

**ボリューム= &lt;ボリューム&gt;**  
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種類あります：ストレージバックアップマウントポイント、バインドマウント、デバイスマウントです。

#### 典型的なコンテナのルートファイルシステム構成

```
rootfs: thin1:base-100-disk-1,size=8G
```

#### ストレージバックアップマウントポイント

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、以下の3種類があります:

- イメージベース: 単一の ext4 フォーマットファイルシステムを含む生のイメージです。
- ZFSサブボリューム: 技術的にはバインドマウントですが、管理対象ストレージを使用するため、サイズ変更やスナップショットが可能です。
- ディレクトリ: size=0 を指定すると、特別なケースとして、生のイメージの代わりにディレクトリが作成されます。

#### 注記

ストレージバックアップマウントポイントボリューム用の特殊オプション構文 `STORAGE_ID:SIZE_IN_GB` は、指定されたストレージ上に指定サイズで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
```

ストレージ thin1 上に 10GB のボリュームを割り当てる、ボリューム ID プレスホールダー 10 を割り当てられたボリューム ID で置き換え、コンテナ内のマウントポイントを /path/in/container に設定します。

## バインドマウントポイント

バインドマウントにより、Proxmox VEホスト上の任意のディレクトリをコンテナ内で利用できます。主な使用例は以下の通りです：

- ゲスト内のホームディレクトリへのアクセス
- ゲスト内のUSBデバイスディレクトリへのアクセス
- ゲスト内でホストのNFSマウントにアクセス

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見なされるため、コンテナ内部からスナップショットの作成やクォータの管理を行うことはできません。非特権コンテナでは、ユーザーマッピングに起因する権限の問題が発生する可能性があり、ACLを使用できません。

### 注意

vzdumpを使用する場合、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予約されたソースディレクトリ（例：/mnt/bindmounts 以下のディレクトリ階層）を使用してのみ確立してください。/、/var、/etcなどのシステムディレクトリをコンテナにバインドマウントすることは絶対に避けてください。これは重大なセキュリティリスクをもたらします。

### 注意

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めてはいけません。

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  
/shared パスでアクセス可能にするには、以下のような設定行を追加します：

```
mp0: /mnt/bindmounts/shared,mp=/shared
```

を /etc/pve/lxc/100.conf に追加します。

あるいは、pct ツールを使用して

```
pct set 100 -mp0 /mnt/bindmounts/shared,mp=/shared
```

で同じ結果を得られます。

## 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 オプションは適用されます。

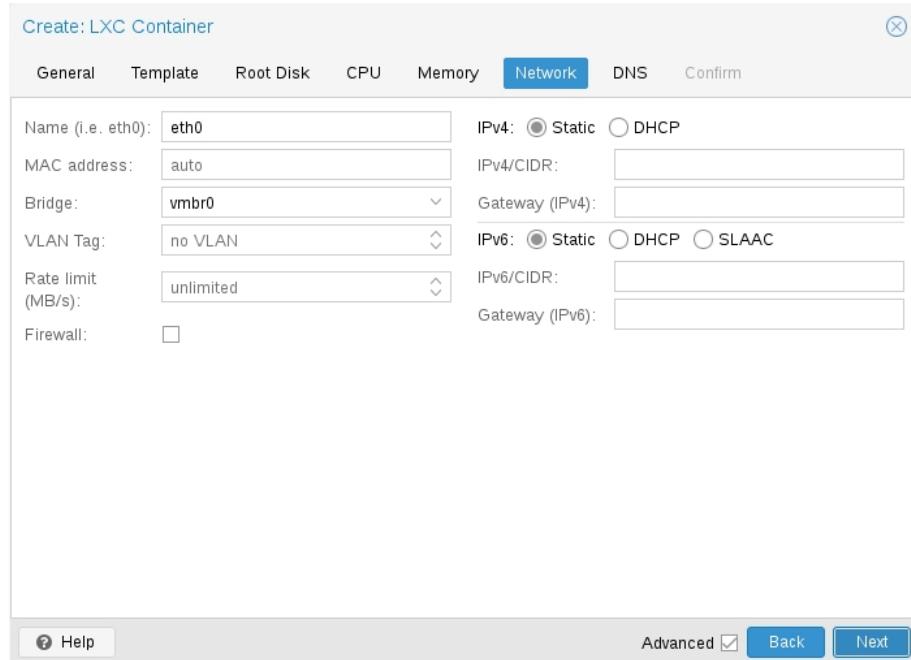
### 注意

デバイスマウントポイントは特別な状況でのみ使用してください。ほとんどの場合、ストレージベースのマウントポイントが同等のパフォーマンスとより多くの機能を提供します。

### 注意

vzdumpを使用する場合、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



单一のコンテナに対して最大10個のネットワークインターフェースを設定できます。対応するオプションは

net0 から net9 まで、以下の設定を含めることができます:

```
net[n]:name=&lt;string&gt; [,bridge=&lt;bridge&gt;] [,firewall=&lt;1|0&gt;]
[,gw=&lt;GatewayIPv4&gt;] [,gw6=&lt;GatewayIPv6&gt;] [,hwaddr=&lt;XX:XX:XX:XX:XX:XX&gt;]
[,ip=&lt;(IPv4/CIDR|dhcp|manual)&gt;] [,ip6=&lt;(IPv6/CIDR|auto|dhcp|manual)&gt;]
[,link_down=&lt;1|0&gt;] [,mtu=&lt;整数&gt;] [,rate=&lt;Mbps&gt;] [,tag=&lt;整数&gt;]
[,trunks=&lt;vlanid[,vlanid...]&gt;] [,type=&lt;veth&gt;]
```

コンテナのネットワークインターフェースを指定します。

**bridge= &lt;bridge&gt;**

ネットワークデバイスを接続するブリッジ。

**firewall= &lt;boolean&gt;**

このインターフェー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します。

**gw= &lt;GatewayIPv4&gt;**

IPv4トラフィックのデフォルトゲートウェイ。

**gw6= &lt;GatewayIPv6&gt;**

IPv6トラフィックのデフォルトゲートウェイ。

**hwaddr= &lt;XX:XX:XX:XX:XX:XX&gt;**

I/G (個別/グループ) ビットが設定されていない共通のMACアドレス。

**ip= &lt;(IPv4/CIDR|dhcp|manual)&gt;**

CIDR形式のIPv4アドレス。

```

ip6= &lt;(IPv6/CIDR|auto|dhcp|manual)&gt;
    CIDR形式のIPv6アドレス。

link_down= &lt;boolean&gt;
    このインターフェースを切断するかどうか（プラグを抜くような状態）。

mtu= &lt;integer&gt; (64 - 65535)
    インターフェースの最大転送単位。（lxc.network.mtu）

name= &lt;string&gt;
    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）

rate= &lt;mbps&gt;
    インターフェースにレート制限を適用

tag= &lt;integer&gt; (1 - 4094)
    このインターフェースのVLANタグ。

trunks= &lt;vlanid[,vlanid...]&gt;
    インターフェースを通過させるVLAN ID

type= &lt;veth&gt;
    ネットワークインターフェースタイプ。

```

#### 11.4.6 コンテナの自動起動とシャットダウン

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、Webインターフェースのコンテナのオプションパネルで「起動時に開始」オプションを選択するか、以下のコマンドを実行します：

パネルで「起動時に開始」オプションを選択するか、次のコマンドを実行します：

```
# pct set CTID -onboot 1
```

##### 起動およびシャットダウン順序



コンテナの起動順序を微調整したい場合は、以下のパラメータを使用できます：

- 起動/シャットダウン順序:** 起動順序の優先度を定義します。例えば、CTを最初に起動させたい場合は1に設定します。（シャットダウン時は逆の起動順序を使用するため、起動順序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:** このコンテナ起動と後続コンテナ起動の間隔を定義します。例えば、他のコンテナ起動前に240秒待機したい場合は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発行後、コンテナがオフラインになるまで待機する秒数を定義します。デフォルト値は60秒です。つまりProxmox VEは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秒待機します。60秒経過後もマシンがオンラインの場合、シャットダウン操作の失敗を通知します。

注意: 起動/シャットダウン順序パラメータが設定されていないコンテナは、常にこのパラメータが設定されたコンテナの後に起動します。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有効であり、クラスター全体では意味を持ちません。

ホスト起動と最初のコンテナ起動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。

## 11.4.7 フックスクリプト

```
# pct set 100 -hookscript local:snippets/hookscript.pl
```

CTにフックスクリプトを追加するには、configプロパティのhookscriptを使用します。

ゲストのライフサイクルの様々な段階で呼び出されます。例とドキュメントについては、[/usr/share/pve-docs/examples/guest-example-hookscript.pl](#) のサンプルスクリプトを参照してください。

## 11.5 セキュリティ上の考慮事項

コンテナはホストシステムのカーネルを使用します。これにより悪意のあるユーザーに対する攻撃対象領域が生じます。一般的に、完全な仮想マシンの方が優れた分離性を提供します。コンテナを未知または信頼できないユーザーに提供する場合には、この点を考慮すべきです。

攻撃対象領域を縮小するため、LXCはAppArmor、CGroups、カーネルネームスペースなどの多くのセキュリティ機能を利用しています。

### 11.5.1 AppArmor

AppArmorプロファイルは、潜在的に危険な操作へのアクセスを制限するために使用されます。マウントなどの一部のシステムコールは実行が禁止されます。

AppArmorの動作を追跡するには以下を使用します:

```
# dmesg | grep apparmor
```

推奨はされませんが、コンテナに対してAppArmorを無効化することは可能です。これによりセキュリティリスクが生じます。システム設定が誤っている場合や、LXCまたはLinuxカーネルの脆弱性が存在する場合には、コンテナ内で実行された一部のシステムコールが権限昇格につながる可能性があります。

コンテナのAppArmorを無効化するには、コンテナ設定ファイル（

```
lxc.apparmor.profile= unconfined
```

/etc/pve/lxc/CTID.conf に以下の行を追加します:



警告

これは本番環境での使用には推奨されませんのでご注意ください。

## 11.5.2 制御グループ (*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メカニズムです。

*cgroups* を介して制御される主なリソースは、CPU時間、メモリおよびスワップ制限、デバイスノードへのアクセスです。*cgroups* はまた、スナップショット取得前にコンテナを「凍結」するためにも使用されます。

現在利用可能な*cgroups*には、[レガシー](#)と[cgroupv2](#)の2つのバージョンがあります。

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」構成が採用されており、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介して一部のサブシステムを引き継ぐ追加の*cgroupv2*コントローラが使用されていました（詳細は[カーネルパラメータのドキュメント](#)を参照）。

### cgroupバージョンの互換性

Proxmox VEにおける純粋な*cgroupv2*と旧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点です。コンテナのメモリとスワップ設定はこれらの値に直接マッピングできますが、以前はメモリ制限とメモリ+スワップの合計制限のみが設定可能でした。

もう1つの重要な相違点は、デバイスコントローラーの設定方法が完全に異なることです。このため、純粋な*cgroupv2*環境では現在ファイルシステムクオータがサポートされていません。

純粋な*cgroupv2*環境で動作させるには、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以降を使用するコンテナ<sup>(1)</sup>は*cgroupv2*をサポートしており、*systemd*をinitシステムとして使用しないコンテナ<sup>(2)</sup>も同様です。

### 注記

CentOS 7 および Ubuntu 16.10 は、*cgroupv2* 環境で実行するには *systemd* のバージョンが古すぎる代表的な Linux ディストリビューションです。以下のいずれかの方法を選択できます：

- 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します。上記の例では、Ubuntu 18.04 または 20.04、CentOS 8（または AlmaLinux や Rocky Linux などの RHEL/CentOS 派生ディストリビューション）が該当します。これにより最新のバグ修正やセキュリティ修正、多くの場合新機能も得られ、サポート終了日（EOL）を先延ばしできる利点があります。
- コンテナの *systemd* 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バックポートリポジトリを提供している場合、これは簡単で迅速な暫定措置となり得る。
- コンテナまたはそのサービスを仮想マシンに移行する。仮想マシンはホストとの相互作用が大幅に少ないため、数十年前のOSバージョンでも問題なくインストールできます。
- 従来の*cgroup*コントローラーに戻す。有効な解決策ではあるが、恒久的な解決策ではないことに注意。Proxmox VE 9.0以降、従来のコントローラーはサポートされなくなる。

<sup>1</sup> Proxmox VEが提供するコンテナテンプレートの最新メジャーバージョン全てを含む

<sup>2</sup> 例：Alpine Linux

## Cgroupバージョンの変更

### ヒント

ファイルシステムのクオータが不要で、すべてのコンテナが`cgroupv2`をサポートしている場合、新しいデフォルト設定を維持することを推奨します。

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を使用できます:

```
systemd.unified_cgroup_hierarchy=0
```

このパラメータを追加する場所については、カーネル起動コマンドラインの編集に関する[このセクション](#)を参照してください。

## 11.6 ゲストオペレーティングシステムの設定

Proxmox VEはコンテナ内のLinuxディストリビューションを検出し、いくつかのファイルを変更します。コンテナ起動時に実行される処理の簡単なリストは以下の通りです：

### /etc/hostname を設定

コンテナ名を設定するため

### /etc/hosts の変更

ローカルホスト名の検索を可能にするため

### ネットワーク設定

コンテナに完全なネットワーク設定を渡す

### DNSを設定する

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：生成されるgettyプロセスの数を修正する

### rootパスワードを設定する

新しいコンテナ作成時

### 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をランダム化

すべてのコンテナでcronが同時に起動しないように

Proxmox VEによる変更はコメントマーカーで囲まれています:

```
# --- BEGIN PVE ---
<data>
# --- END PVE ---
```

それらのマーカーはファイル内の適切な位置に挿入されます。そのようなセクションが既に存在する場合、その場で更新され、移動されることはありません。

ファイルの変更は、そのファイル用の `.pve-ignore` ファイルを追加することで防止できます。例えば、

`/etc/.pve-ignore.hosts` が存在する場合、`/etc/hosts` ファイルは変更されません。これは以下のように作成する空のファイルで十分です：

```
# touch /etc/.pve-ignore.hosts
```

変更の大半はOS依存であるため、ディストリビューションやバージョンによって異なります。`ostype`を手動で `unmanaged` に設定することで、変更を完全に無効化できます。

OSタイプの検出は、コンテナ内の特定のファイルを検査することで行われます。 Proxmox VE はまず

`/etc/os-release` ファイル [を確認します<sup>\(3\)</sup>](#)。このファイルが存在しない場合、または明確に認識可能なディストリビューション識別子を含まない場合、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。

#### Ubuntu

```
inspect /etc/lsb-release (DISTRO_ID=Ubuntu)
```

#### Debian

```
/etc/debian_version をテスト
```

#### Fedora

```
test /etc/fedora-release
```

#### RedHat または CentOS

```
test /etc/redhat-release
```

#### ArchLinux

```
test /etc/arch-release
```

#### Alpine

```
test /etc/alpine-release
```

#### Gentoo

```
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、コンテナの起動は失敗します。

## 11.7 コンテナストレージ

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。1つのコンテナが複数のマウントポイントを持つことが可能である。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。

<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に取って代わります <https://manpages.debian.org/stable/systemd/os-release.5.en.html>

例えば、コンテナのルートファイルシステムを低速で安価なストレージに配置しつつ、データベースを高速で分散型のストレージに配置（第二のマウントポイント経由）できます。詳細は「マウントポイント」セクションを参照してください。

Proxmox VEストレージライブラリがサポートするあらゆるストレージタイプを利用可能です。つまり、コンテナはローカルストレージ（例：LVM、ZFS、ディレクトリ）、共有外部ストレージ（例：iSCSI、NFS）、さらにはCephのような分散ストレージシステム上に保存できます。基盤となるストレージが対応していれば、スナップショットやクローンといった高度なストレージ機能も利用可能です。vzdumpバックアップツールはスナップショットを活用し、一貫性のあるコンテナバックアップを提供します。

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用して直接マウントできます。これにより、実質的にオーバーヘッドなしでコンテナ内からローカルリソースにアクセスできます。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利用できます。

## 11.7.1 FUSEマウント

### 警告

Linuxカーネルのfreezerサブシステムに既存の問題があるため、コンテナ内のFUSEマウントの使用は強く推奨されません。コンテナはサスペンドやスナップショットモードのバックアップのために凍結する必要があるためです。

FUSE マウントを他のマウント機構やストレージ技術で代替できない場合、Proxmox ホスト上に 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からアクセス可能にすることができます。

## 11.7.2 コンテナ内のクォータの使用

クォータを使用すると、コンテナ内で各ユーザーが使用できるディスク容量に制限を設定できます。

### 注記

現在、レガシー`cgroups`の使用が必要です。

### 注意

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、現在は特権コンテナでのみ動作します。

クォータオプションを有効にすると、マウントポイントに対して次のマウントオプションが使用されます：`usrquota=aquo`

これにより、他のシステムと同様にクォータを使用できます。以下のコマンドを実行することで、`/aquota.user` および `/aquota.group` ファイルを初期化するには、以下を実行します：

```
# quotacheck -cmug / #
quotao /
```

その後、`edquota`コマンドを使用してクォータを編集します。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してください。

### 注意

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ります。その際、`/` ではなくマウントポイントのパスを指定してください。

### 11.7.3 コンテナ内のACLの使用

標準のPosixアクセス制御リスト（ACL）はコンテナ内でも利用可能です。ACLを使用すると、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ます。

### 11.7.4 コンテナのマウントポイントのバックアップ

バックアップにマウントポイントを含めるには、コンテナ設定でそのバックアップオプションを有効にしてください。既存のマウントポイント mp0 の場合

```
mp0: guests:subvol-100-disk-1,mp=/root/files,size=8G  
backup=1 を追加して有効化してください。
```

```
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、上記の手順で backup=0 を追加するか、GUI の [バックアップ] チェックボックスをオフにしてください。チェックボックスをオフにしてください。

### 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ションされる際に追加のマウントポイントもレプリケーションされます。Proxmox VEのストレージレプリケーション機構でマウントポイントをスキップしたい場合、そのマウントポイントに対して「レプリケーションをスキップ」オプションを設定できます。Proxmox VE 5.0 以降、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。コンテナにレプリケーションが設定されている状態で、異なるタイプのストレージにマウントポイントを追加する場合、そのマウントポイントに対して「レプリケーションをスキップ」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は vzdump ツールを使用できます。詳細は vzdump のマニュアルページを参照してください。

### 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は、pct restore コマンドを使用して復元できます。デフォルトでは、pct restore はバックアップされたコンテナ設定を可能な限り復元しようとします。コマンドラインでコンテナオプションを手動設定することで、バックアップされた設定を上書きすることも可能です（ 詳細は pct マニュアルページを参照）。

---

**注**

vzdump アーカイブに含まれるバックアップ構成は、`pvesm extractconfig` を使用して表示できます。

---

復元モードには基本的に2種類あり、マウントポイントの処理方法のみが異なります：

**「簡易」復元モード**

`rootfs` パラメータも任意の `mpX` パラメータも明示的に設定されていない場合、バックアップ設定ファイルからのマウントポイント設定は次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で指定されたストレージ（デフォルト: `local`）上に、保存用マウントポイント用のボリュームを作成します。
  3. バックアップアーカイブからファイルを抽出する
  4. 復元された構成にバインドおよびデバイスマウントポイントを追加（rootユーザーに限定）
- 

**注**

バインドおよびデバイスマウントポイントはバックアップ対象外であるため、最終ステップではファイルは復元されず、設定オプションのみが復元されます。これらのマウントポイントは、別のメカニズム（例：多数のコンテナにバインドマウントされるNFSスペース）でバックアップされているか、あるいはバックアップ対象外と想定されていることが前提です。

---

この簡易モードは、Webインターフェースのコンテナ復元操作でも使用されます。

**「詳細」復元モード**

`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高度モードに切り替わります。この高度モードでは、バックアップアーカイブに含まれる `rootfs` および `mpX` 構成オプションは完全に無視され、代わりにパラメータとして明示的に指定されたオプションのみが使用されます。

このモードでは、復元時にマウントポイント設定を柔軟に構成できます。例：

- 各マウントポイントごとに、対象ストレージ、ボリュームサイズ、その他のオプションを個別に設定
- 新しいマウントポイント構成に従ってバックアップ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復元（rootユーザーに限定）

## 11.9 pctによるコンテナ管理

「Proxmox Container Toolkit」（`pct`）は、Proxmox VEコンテナを管理するためのコマンドラインツールです。コンテナの作成や削除、コンテナの実行制御（起動、停止、再起動、移行など）が可能です。コンテナの設定ファイル（`config` ファイル）内のパラメータ設定（例：ネットワーク構成やメモリ制限）にも使用できます。

---

## 11.9.1 CLI使用例

Debianテンプレートに基づくコンテナの作成（Webインターフェース経由でテンプレートを事前にダウンロード済みであることが前提）

```
# pct create 100 /var/lib/vz/template/cache/debian-10.0-standard_10.0-1-->
    _amd64.tar.gz
```

コンテナ 100 を起動

```
# pct start 100
```

getty経由でログインセッションを開始

```
# pct console 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

```
# pct enter 100
```

設定を表示する

```
# pct config 100
```

eth0 という名前のネットワークインターフェースを追加し、ホストブリッジ vmbr0 にブリッジ接続する。稼働中に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に削減

```
# pct set 100 -memory 512
```

コンテナを破棄すると、常にアクセス制御リストから削除され、コンテナのファイアウォール設定が削除されます。レプリケーションジョブ、バックアップジョブ、および高可用性リソース構成からコンテナを追加で削除したい場合は、`--purge` を有効にする必要があります。

```
# pct destroy 100 --purge
```

マウントポイントボリュームを別のストレージに移動します。

```
# 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、ボリュームmp0がソースCTから削除され、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は、ボリューム名が新しい所有者に一致するよう名前が変更されます。

```
# 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 で特定のコンテナを起動できない場合、--debug フラグを指定してデバッグ出力を収集すると役立つ場合があります（CTID をコンテナの CTID に置き換えてください）：

```
# pct start CTID --debug
```

あるいは、以下の lxc-start コマンドを使用することもできます。これにより、-o 出力オプションで指定されたファイルにデバッグログが保存されます：

```
# lxc-start -n CTID -F -l DEBUG -o /tmp/lxc-CTID.log
```

このコマンドはコンテナをフォアグラウンドモードで起動しようとします。コンテナを停止するには、別のターミナルで pct shutdown CTID または pct stop CTID を実行してください。

収集されたデバッグログは /tmp/lxc-CTID.log に書き込まれます。

### 注

pct start による前回の起動試行以降にコンテナの設定を変更した場合は、lxc-start が使用する設定も更新するために、少なくとも一度は pct start を実行する必要があります。

## 11.10 移行

クラスタをお持ちの場合は、以下のコマンドでコンテナを移行できます。

```
# pct migrate <ctid> <target>;
```

コンテナがオフラインである限り、この操作は機能します。ローカルボリュームやマウントポイントが定義されている場合、移行処理では同じストレージがターゲットホストに定義されている場合に限り、ネットワーク経由でコンテンツをコピーします。

技術的な制約により、稼働中のコンテナはライブ移行できません。代わりに再起動移行を実行できます。これはコンテナをシャットダウンし、移動した後、ターゲットノードで再起動するものです。コンテナは非常に軽量であるため、通常は数100ミリ秒程度のダウンタイムしか発生しません。

再起動マイグレーションは、Webインターフェースから、または pct migrate コマンドに--restart フラグを指定して実行できます。

再起動マイグレーションでは、コンテナをシャットダウンし、指定されたタイムアウト時間（デフォルトは180秒）後に強制終了します。その後、オフラインマイグレーションと同様にコンテナを移行し、完了後にターゲットノードでコンテナを起動します。

## 11.11 設定

/etc/pve/lxc/<CTID>.conf ファイルにはコンテナ設定が保存されます。<CTID> は対象コンテナの数値IDです。/etc/pve/内に保存される他のファイルと同様に、これらは自動的にクラスタ内の全ノードに複製されます。

### 注記

CTID<100 は内部用途のために予約されており、CTID はクラスタ全体で一意である必要があります。

### コンテナ設定の例

```
ostype: debian arch:  
amd64 hostname: www  
memory: 512  
swap: 512  
net0: bridge=vmbr0,hwaddr=66:64:66:64:64:36,ip=dhcp,name=eth0,type=vethrootfs: local:107/vm-107-disk-  
1.raw,size=7G
```

設定ファイルは単純なテキストファイルです。viやnanoなどの通常のテキストエディタで編集できます。小規模な修正を行う際に便利ですが、変更を反映させるにはコンテナの再起動が必要であることを覚えておいてください。

そのため、通常はpctコマンドを使用してこれらのファイルを生成・修正するか、GUIで全てを操作する方が望ましいです。当ツールキットは、稼働中のコンテナへの変更を即座に適用できる高度な機能を備えています。この機能は「ホットプラグ」と呼ばれ、この場合コンテナの再起動は不要です。

変更がホットプラグできない場合、それは保留中の変更として登録されます（GUIでは赤色で表示）。これらはコンテナの再起動後にのみ適用されます。

### 11.11.1 ファイル形式

コンテナ設定ファイルは、コロン区切りのシンプルなキー/値形式を採用しています。各行は次の形式です：

```
# これはコメントです OPTION: value
```

これらの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、同様に無視されます。

LXCスタイルの低レベル設定を直接追加することが可能です。例：

```
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ます。

### 11.11.2 スナップショット

スナップショットを作成すると、pct はスナップショット作成時の設定を、同じ設定ファイル内の独立した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、設定ファイルは次のようにになります：

#### スナップショット付きコンテナ設定

```
memory: 512
swap: 512
parent: testsnapshot
...
[testsnapshot] メモリ:
512
スワップ: 512
スナップタイム: 1457170803
...
```

スナップショットに関連するプロパティとして、`parent`や`snaptime`などがあります。`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の作成時刻（Unixエポック）です。

### 11.11.3 オプション

**アーキテクチャ: &lt;amd64| arm64| armhf| i386| riscv32| riscv64&gt;** (デフォルト = `amd64`)

OS アーキテクチャタイプ。

**cmode: &lt;console| shell| tty&gt;** (default = `tty`)

コンソールモード。デフォルトでは、コンソールコマンドは利用可能なttyデバイスへの接続を試みます。`cmod`を`console`に設定すると、代わりに`/dev/console`へのアタッチを試みます。`cmod`を`shell`に設定すると、コンテナ内でシェルを起動します（ログインなし）。

**console: &lt;boolean&gt;** (デフォルト = 1)

コンソールデバイス（`/dev/console`）をコンテナに接続します。

**cores: &lt;integer&gt;** (1 - 8192)

コンテナに割り当てるコア数。デフォルトではコンテナは利用可能な全コアを使用できます。

**cpulimit: &lt;number&gt;** (0 - 8192) (default = 0)

CPU 使用率の制限。

---

#### 注

コンピュータが2つのCPUを搭載している場合、合計で2のCPU時間があります。値0/無CPU制限なしを示します。

---

**cpuunits: &lt;integer&gt;** (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ます。数値が大きいほど、このコンテナが割り当たされるCPU時間が増加します。数値は他のすべての実行中のゲストのウェイトに対する相対値です。

**debug: &lt;boolean&gt; (デフォルト= 0)**

より詳細な出力を試みます。現時点では起動時のデバッグルогレベルのみ有効化します。

**description: &lt;string&gt;**

コンテナの説明。WebインターフェースのCTサマリーに表示されます。設定ファイル内にコメントとして保存されます。

**dev[n]: [[path=&lt;パス&gt;] [,deny-write=&lt;1|0&gt;] [,gid=&lt;整数&gt;] [,mode=&lt;8進アクセスモード&gt;] [,uid=&lt;整数&gt;]]**

コンテナに渡すデバイス

**deny-write= &lt;boolean&gt; (デフォルト= 0)**

コンテナがデバイスへの書き込みを拒否する

**gid= &lt;integer&gt; (0 - N)**

デバイスノードに割り当てるグループID

**mode= &lt;8進アクセスモード&gt;**

デバイスノードに設定するアクセスモード

**path= &lt;パス&gt;**

コンテナに渡すデバイスのパス

**uid= &lt;整数&gt; (0 - N)**

デバイスノードに割り当てるユーザーID

**機能: [force\_rw\_sys=&lt;1|0&gt;] [,fuse=&lt;1|0&gt;] [,keyctl=&lt;1|0&gt;] [,mknod=&lt;1|0&gt;] [,mount=&lt;fstype;fstype;...&gt;] [,nesting=&lt;1|0&gt;]**

コンテナが高度な機能にアクセスできるようにします。

**force\_rw\_sys= &lt;boolean&gt; (デフォルト= 0)**

非特権コンテナで/sysをmixedモードではなくrwモードでマウントします。これにより、新しい(v245以上)systemd-networkの使用下でネットワーク機能が破損する可能性があります。

**fuse= &lt;boolean&gt; (デフォルト= 0)**

コンテナ内で fuse ファイルシステムの使用を許可します。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してください。

**keyctl= &lt;boolean&gt; (デフォルト= 0)**

非特権コンテナのみ: keyctl() システムコールの使用を許可します。コンテナ内で docker を使用するにはこれが必要です。デフォルトでは、非特権コンテナはこのシステムコールが存在しないものと見なします。これは主に systemd-networkd に対する回避策であり、権限不足によりカーネルによって一部の keyctl() 操作が拒否された場合、systemd-networkd はこれを致命的なエラーとして扱うためです。本質的には、systemd-networkd を実行するか、docker を実行するかを選択できます。

**mknod= &lt;boolean&gt; (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することを許可します。これには seccomp トラップをユーザー空間でサポートするカーネル (5.3 以降) が必要です。これは実験的な機能です。

**ip6= &lt;(IPv6|CIDR|auto|dhcp|manual)&gt;**

CIDR形式のIPv6アドレス。

**link\_down= &lt;boolean&gt;**

このインターフェースを切断するかどうか（プラグを抜くような状態）。

**mtu= &lt;integer&gt; (64 - 65535)**

インターフェースの最大転送単位。（lxc.network.mtu）

**name= &lt;string&gt;**

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）

**rate= &lt;mbps&gt;**

インターフェースにレート制限を適用

**tag= &lt;integer&gt; (1 - 4094)**

このインターフェースのVLANタグ。

**trunks= &lt;vlanid[,vlanid...]&gt;**

インターフェースを通過させるVLAN ID

**type= &lt;veth&gt;**

ネットワークインターフェースタイプ。

**onboot: &lt;boolean&gt; (デフォルト= 0)**

コンテナがシステム起動時に起動されるかどうかを指定します。

**ostype: &lt;alpine| archlinux| centos| debian| devuan| fedora | gentoo | nixos | opensuse | ubuntu | unmanaged&gt;**

OSタイプ。コンテナ内の設定を構成するために使用され、/usr/share/lxc/config/&lt;ostype&gt;.common.conf 内の lxc 設定スクリプトに対応します。値 *unmanaged* を使用すると、OS固有の設定をスキップできます。

**protection: &lt;boolean&gt; (デフォルト= 0)**

コンテナの保護フラグを設定します。これにより、CTまたはCTのディスクの削除/更新操作が防止されます。

**rootfs: [volume=] &lt;ボリューム&gt; [,acl=&lt;1|0&gt;] [,mountoptions=&lt;オプション[,オプション...]&gt;] [,quota=&lt;1|0&gt;][,replicate=&lt;1|0&gt;] [,ro=&lt;1|0&gt;] [,shared=&lt;1|0&gt;] [,size=&lt;DiskSize&gt;]**

ボリュームをコンテナのルートとして使用します。

**acl= &lt;boolean&gt;**

明示的にACLサポートを有効または無効にします。

**mountoptions= &lt;opt[,opt...]&gt;**

rootfs/mps 用の追加マウントオプション。

**quota= &lt;boolean&gt;**

コンテナ内でユーザークォータを有効化（ZFSサブボリュームではサポートされていません）

**replicate= &lt;boolean&ampgt (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= &lt;boolean&ampgt**

読み取り専用マウントポイント

**shared= &lt;boolean&ampgt (デフォルト= 0)**

この非ボリュームマウントポイントを、すべてのノードで利用可能としてマークします。



#### 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ものではなく、既に共有されていることを前提としています。

**size= &lt;DiskSize&ampgt**

ボリュームサイズ（読み取り専用値）。

**ボリューム= &lt;ボリューム&ampgt**

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: &lt;string&ampgt**

コンテナのDNS検索ドメインを設定します。searchdomainとnameserverの両方を設定しない場合、作成時にはホストの設定が自動的に使用されます。

**startup: '[[order]=\d+] [,up=\d+] [,down=\d+]'**

起動およびシャットダウンの動作。Order は起動順序を定義する非負の数値です。シャットダウンは逆順序で行われます。さらに、次の VM の起動または停止前に待機する遅延時間を秒単位で設定できます。

**swap: &lt;integer&ampgt (0 - N) (default = 512)**

コンテナのSWAP容量（MB単位）。

**tags: &lt;string&ampgt**

コンテナのタグ。これはメタ情報のみです。

**template: &lt;boolean&ampgt (デフォルト= 0)**

テンプレートの有効/無効設定。

**タイムゾーン: &lt;string&ampgt**

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、何も実行されません。ホストのタイムゾーンに合わせるために「host」に設定するか、/usr/share/zoneinfo/zone.tab にある任意のタイムゾーンオプションを設定できます。

**tty: &lt;integer&ampgt (0 - 6) (デフォルト= 2)**

コンテナが使用できるttyの番号を指定します

**unprivileged: &lt;boolean&ampgt (デフォルト= 0)**

コンテナを非特権ユーザーとして実行します（手動で変更しないでください）。

```
unused[n]: [volume=] <volume>;
```

未使用ボリュームへの参照。内部で使用されるため、手動で変更しないでください。

```
volume= <volume>;
```

現在使用されていないボリューム。

## 11.12 ロック

コンテナ移行、スナップショット、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操作を防ぐためにロックを設定します。場合によっては、このようなロックを手動で解除する必要がある場合があります（例：停電後）。

```
# pct unlock <CTID>;
```



### 注意

ロックを設定した操作が確実に終了している場合にのみ実行してください。

## 第12章

# ソフトウェア定義ネットワーク

Proxmox VE のソフトウェア定義ネットワーク (SDN) 機能により、仮想ゾーンおよびネットワーク (VNets) を作成できます。この機能により、高度なネットワーク構成や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、柔軟なソフトウェア制御による構成を用いて、仮想ゲストネットワークの分離ときめ細かい制御を可能にします。

分離はゾーン、仮想ネットワーク（VNet）、サブネットを通じて管理されます。ゾーンはそれ自体が仮想的に分離されたネットワーク領域です。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。

ゾーンの種類によって、ネットワークの挙動は異なり、特定の機能、利点、制限を提供します。

SDN のユースケースは、個々のノード上の独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ります。

クラスタ全体のデータセンター SDN 管理インターフェイスで VNet を設定すると、各ノードのローカルで共通の Linux ブリッジとして利用可能になり、VM およびコンテナに割り当てられます。

## 12.2 サポート状況

### 12.2.1 沿革

Proxmox VE SDN スタックは2019年から実験的機能として提供され、多くの開発者やユーザーによって継続的に改善・テストされてきました。Proxmox VE 6.2でのWeb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ました。Proxmox VE 7リリースサイクルでは、数多くの改善と機能が追加されました。ユーザーフィードバックに基づき、基本的な設計選択とその実装が非常に堅牢で安定していることが明らかになりました。その結果、「実験的」というラベルはSDNスタックの現状を適切に反映していませんでした。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アクセス制御スタックの中核コンポーネントに昇格させることで、SDN機能の完全統合に向けた基盤を構築する決定がなされました。Proxmox VE 8.1では、2つの主要なマイルストーンが達成されました。第一に、IPアドレス管理（IPAM）機能へのDHCP統合が追加され、第二に、SDN統合がデフォルトでインストールされるようになりました。

## 12.2.2 現状

当社のSDN導入における各レイヤーの現在のサポート状況は以下の通りです：

- コアSDN（VNet管理およびProxmox VEスタックとの統合を含む）は完全にサポートされています。
- 仮想ゲスト向けDHCP管理を含むIPAMは、技術プレビュー段階です。
- FRRoutingを介した複雑なルーティングおよびコントローラー統合は、技術プレビュー段階です。

## 12.3 インストール

### 12.3.1 SDNコア

Proxmox VE 8.1 以降、コアのソフトウェア定義ネットワーク (SDN) パッケージは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、すべてのノードに `libpve-network-perl` パッケージをインストールする必要があります:

```
apt update  
apt install libpve-network-perl
```

#### 注記

Proxmox VE バージョン 7.0 以降では、`ifupdown2` パッケージがデフォルトでインストールされています。古いバージョンでシステムをインストールした場合、`ifupdown2` パッケージを明示的にインストールする必要があります。

インストール後、SDN 設定がすべてノードに含められ有効化されるよう、

設定ファイルの末尾に以下の行が存在することを確認してください。これによりSDN設定が読み込まれ有効化されます。

```
source /etc/network/interfaces.d/*
```

### 12.3.2 DHCP IPAM

組み込みのPVE IPアドレス管理スタックへのDHCP統合は、現在DHCPリースを割り当てるために`dnsmasq`を使用しています。これは現在オプトイン方式です。

この機能を利用するには、各ノードに`dnsmasq`パッケージをインストールする必要があります：

```
apt update  
apt install dnsmasq  
# デフォルトインスタンスを無効化  
systemctl disable  
--now dnsmasq
```

### 12.3.3 FRRouting

Proxmox VE SDNスタックは、高度な設定にFRRoutingプロジェクトを利用します。これは現在オプションです。SDNルーティング統合を使用するには、すべてのノードにfrr-pythontoolsパッケージをインストールする必要があります:

```
apt update  
apt install frr-pythontools
```

次に、すべてのノードでfrrサービスを有効にします:

```
systemctl enable frr.service
```

## 12.4 設定の概要

設定はデータセンターレベルでWeb UIから行い、以下のセクションに分かれています:

- SDN: 現在のアクティブなSDN状態の概要を確認でき、保留中の変更をクラスター全体に適用できます。
- ゾーン: 仮想的に分離されたネットワークゾーンを作成・管理します
- VNets: 仮想ネットワークブリッジを作成し、サブネットを管理します

オプションカテゴリでは、SDN設定で使用する追加サービスの追加と管理が可能です。

- コントローラー: 複雑な設定におけるレイヤー3ルーティングの制御用
- DHCP: ゾーン用のDHCPサーバーを定義し、IPAM内のゲストに自動的にIPを割り当て、DHCP経由でリースします。
- IPAM: ゲストのIPアドレス管理を外部で可能にします
- 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サーバー統合を定義します

## 12.5 テクノロジーと構成

Proxmox VEのソフトウェア定義ネットワーク実装は、可能な限り標準的なLinuxネットワーキングを利用します。その理由は、現代のLinuxネットワーキングが機能豊富なSDN実装に必要なほぼすべての要件を満たし、外部依存関係の追加を回避し、障害発生の可能性があるコンポーネントの総量を削減できるためです。

Proxmox VE SDN設定は/etc/pve/sdnに配置され、Proxmox VE[設定ファイルシステム](#)を通じて他のクラスターノードと共有されます。これらの設定は、基盤となるネットワークスタックを管理するツール（例：ifupdown2やfrr）のそれぞれの設定形式に変換されます。

新規変更は即時適用されず、まず保留状態として記録されます。その後、WebインターフェースのメインSDN概要パネルで複数の変更を一括適用可能です。この仕組みにより、様々な変更を単一の原子的な操作として展開できます。

SDNは、/etc/pve/sdnにある.running-configおよび.versionファイルを通じて、ロールアウトされた状態を追跡します。

## 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します。特定のゾーンとその内部の仮想ネットワーク（VNets）にユーザーを制限するため、ゾーンは特定のノードと割り当てられた権限に制限されます。

分離には異なる技術が使用可能です：

- シンプル: 隔離されたブリッジ。単純なレイヤ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手法
- QinQ: スタックVLAN (正式名称 IEEE 802.1ad)
- VXLAN: UDPトンネル経由のレイヤ2 VXLANネットワーク
- EVPN (BGP EVPN): BGPを伴うVXLANによるレイヤ3ルーティングの確立

### 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

ゾーンおよび関連する仮想ネットワーク（VNets）を展開するノード。

#### IPAM

ゾーン内のIPアドレスを管理するためにIPアドレス管理（IPAM）ツールを使用します。オプション、デフォルトはpveです。

#### DNS

DNS APIサーバー。オプション。

#### リバースDNS

逆引きDNS APIサーバー。オプション。

#### DNSZone

DNSドメイン名。`<ホスト名> . <ドメイン>`などのホスト名を登録するために使用されます。DNSゾーンはDNSサーバー上に既に存在している必要があります。オプション。

### 12.6.2 シンプルゾーン

最もシンプルなプラグインです。分離された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上で個別にのみ処理されます。NATまたはルーティング環境で使用可能です。

## 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。VNetで定義されたVLANタグ付け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、異なるノード間のVM間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

各ノードに既に設定済みのローカルブリッジまたはOVSスイッチ。ノード間接続を可能にします。

## 12.6.4 QinQ ゾーン

QinQ（VLANスタッキングとも呼ばれる）は、複数のVLANタグ層を用いて分離を実現します。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。

### 注記

この構成では、物理ネットワークスイッチがスタック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既に設定済みの、VLAN対応ローカルブリッジ

### 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 タイプから選択できます。

### MTU

タグの二重スタック化により、QinQ VLANではさらに4バイトが必要です。例えば、物理インターフェースのMTUが1500の場合、MTUを1496に削減する必要があります。

## 12.6.5 VXLANゾーン

VXLANプラグインは既存ネットワーク（アンダーレイ）上にトンネル（オーバーレイ）を構築します。これにより、デフォルト宛先ポート4789を使用したレイヤ4 UDPデータグラム内にレイヤ2イーサネットフレームがカプセル化されます。

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ります。

たとえば、パブリックインターネット上にVXLAN オーバーレイネットワークを構築し、VMに対して、それら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ます。

### 警告

VXLAN自体は暗号化機能を提供しません。複数のサイトをVXLANで接続する場合は、サイト間VPNなどを使用して、サイト間の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を含めることができます。クラスタ内のすべてのノードをここに記述する必要があります。

#### SDNファブリック

すべてのピアを手動で定義する代わりに、ピアリストを自動生成する[ファブリック](#)を使用してください。

#### MTU

VXLAN カプセル化には 50 バイトを使用するため、MTU は送信元の物理インターフェースよりも 50 バイト小さくする必要があります。

## 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ます。

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あるため、仮想ゲストは、このアドレスをゲートウェイとして使用できます。

VRF（仮想ルーティングおよび転送）インターフェースを介して、異なるゾーンのVNet間でルーティングを機能させることができます。

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。VNetのVXLAN-IDとは異なる必要があります。

#### コントローラ

このゾーンで使用する EVPN コントローラ。（コントローラ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VNetに割り当てられるAnycast 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経由で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ネットワーク内でデフォルトルートをアドバタイズします。オプション。

#### プライマリ出口ノード

複数の出口ノードを使用する場合、全ノードでの負荷分散ではなく、このプライマリ出口ノード経由でトラフィックを強制します。オプションですが、SNATを使用する場合や上流ルーターがECMPをサポートしていない場合には必須です。

#### 出口ノードのローカルルーティング

出口ノードからVM/CTサービスにアクセス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、出口ノードは実ネットワークとEVPNネットワーク間のトラフィック転送のみ許可します）。オプション。

### サブネットのアドバタイズ

EVPNネットワーク内でサブネット全体をアドバタイズします。サイレントVM/CTが存在する場合（例：複数のIPアドレスがあり、アニキヤストゲートウェイがこれらのIPからのトラフィックを検知しない場合、EVPNネットワーク内から該当IPアドレスに到達できなくなります）。オプション。

### ARP ND抑制を無効化

ARPまたはND（ネイバーディスカバリ）パケットを抑制しない。VMでフローティングIPを使用する場合（IPアドレスとMACアドレスがシステム間で移動される）に必要です。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ます。データセンター間または異なるEVPNネットワーク間の相互接続に使用されます。オプション。

### MTU

VXLANカプセル化には50バイトを使用するため、MTUは送信先物理インターフェースの最大MTUより50バイト小さくする必要があります。オプション、デフォルトは1450です。

## 12.7 VNets

SDN GUIで仮想ネットワーク（VNet）を作成すると、各ノードに同名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、このインターフェースをゲストに割り当て、対応する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の意味は異なり、このドキュメントのそれぞれのゾーンセクションで説明されています。



### 警告

現在の状態では、一部のオプションは効果がないか、特定のゾーンでは機能しない場合があります。

VNet設定オプション:

### ID

仮想ネットワーク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。インターフェース上で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に関連付けられたゾーン

### タグ

一意のVLANまたはVXLAN ID

### VLAN対応

インターフェイスで vlan-aware オプションを有効にし、ゲスト内の設定を可能にします。

### ポートの分離

このインターフェースのすべてのゲストポートに対して分離フラグを設定しますが、インターフェース自体には設定しません。これにより、ゲストは分離されていないブリッジポート（ブリッジ自体）へのみトライフィックを送信できます。この設定を有効にするには、影響を受けるゲストを再起動する必要があります。

---

### 注

ポート分離は各ホストにローカルに適用されます。ノード間でVNET内のトライフィックをさらに分離するには、[VNETファイアウォール](#)を使用してください。例えば、デフォルトでDROPし、特定のIPサブネットからゲートウェイへのトライフィック、およびその逆方向のトライフィックのみを許可します。

---

## 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。

サブネットは次の目的で使用できます：

- 特定のVNet上で定義可能なIPアドレスを制限する
- レイヤー3ゾーン内のVNetにルート/ゲートウェイを割り当てる
- レイヤー3ゾーン内のVNetでSNATを有効化
- IPAMプラグインを介した仮想ゲスト（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IPAMに自動的に登録されます。

サブネット構成オプション：

#### ID

CIDR形式の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上に展開されます。

#### SNAT

ソースNATを有効化します。これにより、VNet内のVMがパケットをノードのアウトゴーイングインターフェースに転送することで外部ネットワークに接続できます。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

ドメイン登録にプレフィックスを追加します（例：`<ホスト名>.プレフィックス.<ドメイン>`）。オプション。

## 12.9 コントローラー

一部のゾーンでは、制御プレーンとデータプレーンが分離されており、VNetの制御プレーンを管理するために外部コントローラーが必要です。

現在、外部コントローラーが必要なのは EVPN ゾーンのみです。

### 12.9.1 EVPN コントローラー

EVPN ゾーンでは、制御プレーンを管理するために外部コントローラーが必要です。EVPN コントローラー プラグインは、フリーレンジルーティング (frr) ルーターを設定します。

EVPN コントローラーを有効化するには、すべてのノードで FRR を有効にする必要があります。詳細は [install FRRouting](#) を参照してください。

EVPN コントローラー設定オプション:

#### ASN #

一意のBGP ASN番号。プライベートASN番号（64512～65534、4200000000～4294967294）の使用を強く推奨します。そうしないと、誤ってグローバルルーティングを破壊する可能性があります。

#### SDNファブリック

EVPNゾーンに属するすべてのノードを含むファブリック。アンダーレイネットワークとして使用されます。

#### ピア

EVPNゾーンに属する全てのノードのIPリスト。（外部ノードやルートリフレクターサーバーも含む可能性がある）

### 12.9.2 BGP コントローラ

BGPコントローラはゾーンから直接使用されるものではありません。BGPピアを管理するFRRの設定に使用できます。

BGP-EVPNでは、ノードごとに異なるASNを定義してEBGPを実行するために使用できます。また、EVPNルートを外部BGPピアにエクスポートするためにも使用できます。

---

#### 注記

デフォルトでは、単純なフルメッシュ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。プライベートASN番号（64512～65534）または（4200000000 - 4294967294）の範囲のプライベートASN番号を使用することを強く推奨します。そうしないと、誤ってグローバルルーティングを破壊する可能性があります。

#### ピア

基盤となる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、これによりEBGPが有効になります。

**ループバックインターフェース**

EVPN ネットワークの送信元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します（マルチパス用）。

**ebgp-multipath**

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に備え、ピア到達までのホップ数を増加させます。

**bgp-multipath-as-path-relax**

ピアが異なるASNを持つ場合にECMPを許可します。

## 12.9.3 ISIS コントローラ

ISISコントローラはゾーンから直接使用されるものではありません。ISISドメインにEVPNルートをエクスポートするFRRの設定に使用できます。

ISIS コントローラの設定オプション:

**ノード**

このISISコントローラのノード。

**ドメイン**

一意のISISドメイン。

**ネットワークエンティティタイトル**

このノードを識別する一意のISISネットワークアドレス。

**インターフェース**

ISIS が使用する物理インターフェースの一覧。

**ループバック**

EVPNネットワーク（マルチパス用）の送信元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します。

## 12.10 ファブリック

The screenshot shows the Proxmox VE management interface. On the left is a sidebar with various management options like Summary, Notes, Cluster, Ceph, Options, Storage, Backup, Replication, Permissions, Users, API Tokens, Two Factor, Groups, Pools, Roles, Realms, HA, SDN, Zones, VNets, Options, IPAM, VNet Firewall, and Fabrics. The 'Fabrics' option is selected and highlighted in blue. The main content area has tabs for 'Add Fabric', 'Add Node', and 'Reload'. A table lists two fabrics: 'test1' (OpenFabric) and 'test2' (OSPF). Each fabric has multiple nodes (pve0, pve1, pve2) with their respective IP ranges (e.g., 192.0.2.1, 192.0.2.2, 192.0.2.3 for test1; 198.51.100.1, 198.51.100.2, 198.51.100.3 for test2). The table includes columns for Name, Protocol, IPv4, IPv6, Interfaces, Action, and State.

Name	Protocol	IPv4	IPv6	Interfaces	Action	State
test1	OpenFabric	192.0.2.0/24		pve0, pve1, pve2		new
		192.0.2.1		ens25, ens26		new
		192.0.2.2		ens27, ens28		new
		192.0.2.3		ens29, ens30		new
test2	OSPF	198.51.100.0/24		pve0, pve1, pve2		new
		198.51.100.1		ens19, ens20		new
		198.51.100.2		ens21, ens22		new
		198.51.100.3		ens23, ens24		new

Proxmox VE SDN のファブリックは、クラスタ内のノード間で自動ルーティングを提供します。ノード間のアンダーレイネットワーク設定を簡素化し、SDN 導入の基盤を形成します。

物理ネットワークインターフェース上でルーティングプロトコルを自動設定し、クラスタ内のノード間接続を確立します。これにより、ネットワークトポジの変化に適応する、耐障害性のある自動設定ネットワークファブリックが構築されます。これらのファブリックは、Ceph用のフルメッシュネットワークとして、またはEVPNコントローラーとVXLANゾーン内で使用できます。

### 12.10.1 インストール

OpenFabricおよびOSPFのFRR実装が使用されるため、まずfrrおよびfrr-pythontoolsパッケージがインストールされていることを確認してください。  
パッケージがインストールされていることを確認してください:

```
apt update
apt install frr frr-pythontools
```

### 12.10.2 権限

SDNファブリックの構成を表示するには、ユーザーはSDN.AuditまたはSDN.Allocate権限が必要です。ファブリック構成を作成または変更するには、ユーザーはSDN.Allocate権限が必要です。ノードの構成を表示するには、

には、Sys.Audit または Sys.Modify 権限が必要です。ファブリック内でノードを追加または更新する場合、この操作にはノードの /etc/network/interfaces ファイルへの書き込みを伴うため、特定のノードに対する追加の Sys.Modify 権限が必要です。

## 12.10.3 構成

ファブリックを作成するには、データセンター→SDN→ファブリックに移動し、「ファブリックの追加」をクリックします。希望のプロトコルを選択すると、ファブリックが作成されます。「+」ボタンでファブリックに追加したいノードを選択でき、他のノードとの通信に使用するインターフェースも選択する必要があります。

### ループバックプレフィックス

ファブリックのループバックプレフィックスとして、CIDRネットワーク範囲（例：192.0.2.0/24）を指定できます。設定すると、システムは自動的にすべてのルータIDがこのプレフィックス内に含まれていることを検証します。これにより、アドレス指定スキームの一貫性が確保され、アドレス指定の競合やエラーを防止するのに役立ちます。

### ルータIDの選択

ファブリック内の各ノードには一意のルータIDが必要です。これはドット付き10進表記のIPv4アドレス（例：192.0.2.1）です。OpenFabricでは、コロン区切りの標準的な16進表記によるIPv6アドレス（例：2001:db8::1428:57ab）も使用可能です。ルータIDをアドレスとするダミーインターフェースが自動的に作成され、ファブリックのループバックインターフェースとして機能します（デフォルトではパッシブです）。

### RouteMaps

すべてのファブリックに対して、アクセリストとルートマップが自動的に作成されます。これらはルータを構成し、送信パケットの送信元アドレスを書き換えます。別のノードと通信するとき（例：pingを実行するとき）、これによりトライフィックが物理インターフェースではなくローカルダミーインターフェースのIPアドレスから発信されることが保証されます。これにより、ファブリック全体で一貫したルーティング動作と適切な送信元アドレス選択が実現されます。

### IPv6に関する注意事項

IPv6は現在、OpenFabricファブリックでのみ使用可能です。これらのIPv6ファブリックでは、ファブリック内の全ノードでグローバルIPv6転送を有効にする必要があります。IPv6転送が有効でない場合、非フルメッシュファブリックは機能しません。これは中継ノードが外部ノードへパケットを転送しないためです。現在、IPv4のようにインターフェイス単位でIPv6転送を有効化する簡単な方法はないため、グローバルに有効化する必要があります。これには以下の行を追加することで実現できます：

```
sysctl -w net.ipv6.conf.all.forwarding=1 を設定
```

/etc/network/interfaces ファイル内のファブリックインターフェースに追加します。これにより、そのインターフェースが起動した際に IPv6 転送がグローバルに有効になります。これは、インターフェースが自動 IPv6 設定 (SLAAC)、ネイバー広告、ルータ要求、およびルータ広告を処理する方法に影響を与えることに注意してください。詳細はこちら：

<https://www.kernel.org/doc/Documentation/networking/ip-sysctl.txt> の net.ipv6.conf.all.forwarding セクションを参照

## 12.10.4 OpenFabric

OpenFabricはデータセンターファブリック向けに設計されたルーティングプロトコルです。IS-ISを基盤とし、データセンターで一般的なスパイン・リーフトポロジー向けに最適化されています。

Create: OpenFabric

Name: test1

IPv4 Prefix: 192.0.2.0/24

IPv6 Prefix:

Hello Interval: 1

CSNP Interval:

Help Create

設定オプション:

### ファブリック上

#### 名前

これはOpenFabricファブリックの名前であり、最大8文字までです。

#### IPv4 プレフィックス

ファブリック内のすべてのルータ ID がこのプレフィックス内に含まれていることを確認するために使用される IPv4 CIDR ネットワーク範囲（例：192.0.2.0/24）。

#### IPv6 プレフィックス

ファブリック内のすべてのルータ ID がこのプレフィックス内に含まれていることを確認するために使用される IPv6 CIDR ネットワーク範囲（例：2001:db8::/64）。



#### 警告

IPv6ファブリックを動作させるには、全ノードでグローバル転送を有効にする必要があります。設定方法と追加情報については「[IPv6に関する注意事項](#)」を参照してください。

#### Hello 間隔

隣接ノードとの接続を確立・維持するために送信されるハロー・パケットの頻度（秒単位）を制御します。値を小さくすると障害を早く検出できますが、ネットワークトラフィックが増加します。このオプションはファブリック全体で適用され、ファブリック内の全ノードの全インターフェースがこのハロー間隔プロパティを継承します。デフォルト値は3秒です。

#### CSNP間隔

ノードが隣接ノードとルーティングデータベースを同期する頻度（秒単位）を設定します。値を小さくするとネットワークトポジ情報により迅速に同期できますが、ネットワークトラフィックが増加します。この設定はファブリック全体で有効であり、このファブリック内の全ノードの全インターフェースがこのプロパティを継承します。デフォルト値は10秒です。

## ノード上で

Create: Node

Node:	pve			
IPv4:	192.0.2.1			
IPv6:				
S	Name ↑	Type	IP	IPv6
<input type="checkbox"/>	ens1	eth		
<input type="checkbox"/>	ens18	eth		
<input checked="" type="checkbox"/>	ens19	eth		
<input type="checkbox"/>	ens2	eth		
<input checked="" type="checkbox"/>	ens20	eth		
<input type="checkbox"/>	ens21	eth		
<input type="checkbox"/>	ens22	eth		
<input type="checkbox"/>	ens23	eth		
<input type="checkbox"/>	ens3	eth		
<input type="checkbox"/>	vmbro	bridge	172.16.0.18/24	

**Create** **Create another**

ファブリックを構成するすべてのノードで利用可能なオプション:

## ノード

ファブリックに追加するノードを選択します。現在クラスター内にあるノードのみが表示されます。

## IPv4

OpenFabricネットワークエンティティタイトル(NET)を生成するために使用される一意のIPv4アドレス。同じファブリック内の各ノードは異なるルータIDを持つ必要がありますが、単一のノードはすべてのファブリックで同じNETアドレスを使用する必要があります（指定しない場合、Proxmox VEが自動的に選択し、設定が有効であることを保証します）。

## IPv6

OpenFabric Network Entity Title (NET) の生成に使用される一意の IPv6 アドレス。同じファブリック内の各ノードは異なる Router-ID を持つ必要がありますが、単一のノードはすべてのファブリックで同じ NET アドレスを使用する必要があります。IPv4 と IPv6 アドレスの両方が設定されている場合、NET の導出には IPv4 アドレスが使用されます。



### 警告

IPv6アドレスを使用する場合、最後の3セグメントがNETの生成に使用されます。ノード間でこれらのセグメントが異なることを確認してください。

## インターフェース

他のOpenFabricノードとのピアリング接続を確立するために使用するインターフェースを指定します。事前に割り当てられたIPアドレスのないインターフェースを選択し、必要に応じてIPv4/IPv6列でアドレスを設定することを推奨します。ルーターIDを持つダミーの「ループバック」インターフェースが自動的に作成されます。

## インターフェース上で

追加列を有効化する際に、インターフェースごとに以下のオプションパラメータを設定できます:

### IP

このインターフェースで自動設定されるべきIPv4アドレス。ネットマスク（例: /31）を含める必要があります

### IPv6

このインターフェースに自動設定されるべきIPv6アドレス。ネットマスク（例: /127）を含める必要があります。

### Hello Multiplier

接続失敗と判断されるまでのHelloパケットの未受信数を定義します。値が大きいほどパケット損失に対する耐性は高まりますが、障害検出が遅くなります。デフォルト値は10です。



### 警告

/etc/network/interfaces にエントリがあるインターフェースを削除する場合、SDN設定を適用してもIPアドレスは削除されません。

## 12.10.5 OSPF

OSPF (Open Shortest Path First) は、IP ネットワークを介したトライフィックのルーティングに最適な最短経路を効率的に計算する、広く使用されているリンク状態ルーティングプロトコルです。

Create: OSPF

Name:	test2
IPv4 Prefix:	198.51.100.0/24
Area:	0

[Help](#) [Create](#)

設定オプション:

### ファブリック上

### IPv4プレフィックス

ファブリック内でのすべてのルータIDがこのプレフィックス内に含まれていることを確認するために使用されるIPv4 CIDRネットワーク範囲（例：192.0.2.0/24）。

### エリア

OSPFエリア識別子を指定します。32ビット符号付き整数またはIPアドレスのいずれかです。エリアはOSPFネットワークを階層的に組織化・構造化する手段であり、エリア0（または0.0.0.0）がバックボーンエリアとして機能します。

## ノード上

**Create: Node**

Node:	pve																																																							
IPv4:	198.51.100.1																																																							
<table border="1"> <thead> <tr> <th></th> <th>S</th> <th>Name ↑</th> <th>Type</th> <th>IP</th> </tr> </thead> <tbody> <tr><td><input type="checkbox"/></td><td></td><td>ens1</td><td>eth</td><td></td></tr> <tr><td><input type="checkbox"/></td><td></td><td>ens18</td><td>eth</td><td></td></tr> <tr><td><input checked="" type="checkbox"/></td><td></td><td>ens19</td><td>eth</td><td></td></tr> <tr><td><input type="checkbox"/></td><td></td><td>ens2</td><td>eth</td><td></td></tr> <tr><td><input checked="" type="checkbox"/></td><td></td><td>ens20</td><td>eth</td><td></td></tr> <tr><td><input type="checkbox"/></td><td></td><td>ens21</td><td>eth</td><td></td></tr> <tr><td><input type="checkbox"/></td><td></td><td>ens22</td><td>eth</td><td></td></tr> <tr><td><input type="checkbox"/></td><td></td><td>ens23</td><td>eth</td><td></td></tr> <tr><td><input type="checkbox"/></td><td></td><td>ens3</td><td>eth</td><td></td></tr> <tr><td><input type="checkbox"/></td><td>⚠</td><td>vmbr0</td><td>bridge</td><td>172.16.0.18/24</td></tr> </tbody> </table>			S	Name ↑	Type	IP	<input type="checkbox"/>		ens1	eth		<input type="checkbox"/>		ens18	eth		<input checked="" type="checkbox"/>		ens19	eth		<input type="checkbox"/>		ens2	eth		<input checked="" type="checkbox"/>		ens20	eth		<input type="checkbox"/>		ens21	eth		<input type="checkbox"/>		ens22	eth		<input type="checkbox"/>		ens23	eth		<input type="checkbox"/>		ens3	eth		<input type="checkbox"/>	⚠	vmbr0	bridge	172.16.0.18/24
	S	Name ↑	Type	IP																																																				
<input type="checkbox"/>		ens1	eth																																																					
<input type="checkbox"/>		ens18	eth																																																					
<input checked="" type="checkbox"/>		ens19	eth																																																					
<input type="checkbox"/>		ens2	eth																																																					
<input checked="" type="checkbox"/>		ens20	eth																																																					
<input type="checkbox"/>		ens21	eth																																																					
<input type="checkbox"/>		ens22	eth																																																					
<input type="checkbox"/>		ens23	eth																																																					
<input type="checkbox"/>		ens3	eth																																																					
<input type="checkbox"/>	⚠	vmbr0	bridge	172.16.0.18/24																																																				
<input type="button" value="Create"/> <input type="button" value="Create another"/>																																																								

ファブリックを構成する各ノードで利用可能なオプション:

## ノード

ファブリックに追加するノードを選択します。現在クラスター内にあるノードのみが表示されます。

## IPv4

OSPFネットワーク内でこのルータを識別するための一意のルータID。同一ファブリック内の各ノードは異なるルータIDを持つ必要があります。

## インターフェース

他のOSPFノードとのピアリング接続を確立するために使用するインターフェースを指定します。事前に割り当てられたIPアドレスを持たないインターフェースを選択し、必要に応じてIPv4列でアドレスを設定することを推奨します。ルータIDを持つダミーの「ループバック」インターフェースが自動的に作成されます。

## インターフェース上で

以下のオプションパラメータはインターフェースごとに設定可能です：

## IP

このインターフェースに自動設定されるべきIPv4アドレス。ネットマスク（例: /31）を含める必要があります



### 警告

/etc/network/interfaces にエントリがあるインターフェースを削除する場合、SDN設定を適用してもIPアドレスは削除されません。

## 注意

ダミーインターフェースは自動的にパッシブとして設定されます。IPアドレスが設定されていないすべてのインターフェースは、ポイントツーポイントリンクとして扱われます。

## 12.11 IPAM

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 アドレスを管理します。Proxmox VE の SDN は、たとえば新しいゲスト用の空き IP アドレスを見つけるために IPAM を使用します。

単一の IPAM インスタンスは、1 つ以上のゾーンに関連付けることができます。

### 12.11.1 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。

データセンター設定のSDNセクションにあるIPAMパネルから、PVE IPAMプラグインの現在の状態を確認できます。このUIを使用して、IPマッピングの作成、更新、削除が可能です。特にDHCP機能と組み合わせて使用すると便利です。

DHCPを使用している場合、IPAMパネルを使用して特定のVMのリースを作成または編集でき、これによりDHCP経由で割り当てられたIPを変更できます。DHCPを使用しているVMのIPを編集する際は、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スタックを再読み込みするか、ゲストを再起動することで実現できます。

### 12.11.2 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管理（DCIM）ツールです。

NetBoxをProxmox VE SDNと統合するには、こちらに記載されている手順に従いNetBoxでAPIトークンを作成してください：<https://docs.netbox.dev/stable/integrations/rest-api/#tokens>

NetBoxの設定プロパティは以下の通りです：

#### URL

NetBox REST API エンドポイント:`http://yournetbox.domain.com/api`

#### Token

APIアクセストークン

#### フィンガープリント

NetBox APIのSHA-256フィンガープリント。デフォルト証明書使用時は`openssl x509 -in /etc/ssl/certs/ca-certificates.crt -noout -fingerprint -sha256`で取得可能  
-noout -fingerprint -sha256 で取得可能（デフォルト証明書使用時）。

### 12.11.3 phplIPAM プラグイン

phplIPAMでは、「アプリケーション」を作成し、そのアプリケーションに管理者権限を持つ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以下の通りです：

#### URL

REST-API エンドポイント:`http://phplipam.domain.com/api/<appname>/`

**Token**

APIアクセストークン

**セクション**

整数ID。セクションはphpIPAMにおける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が使用されます  
を使用します。

## 12.12 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 サーバーを定義するために使用されます。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すべてのサブネット IP の DNS 登録を提供します。

### 12.12.1 PowerDNS プラグイン

<https://doc.powerdns.com/authoritative/http-api/index.html>

PowerDNS設定でWebサーバーとAPIを有効にする必要があります:

```
api=yes
api-key=arandomgeneratedstringwebserver=yes
webserver-port=8081
```

PowerDNSの設定オプションは以下の通りです:

**url**

REST API エンドポイント: <http://yourpowerdnserver.domain.com:8081/api/v1/servers/localhost>

**key**

API アクセスキー

**ttl**

レコードのデフォルトTTL

## 12.13 DHCP

Proxmox VE SDN の DHCP プラグインは、ゾーン向けに DHCP サーバーを自動展開するために使用できます。ゾーン内で DHCP 範囲が設定されているすべてのサブネットに対して DHCP を提供します。現在、DHCP 用の利用可能なバックエンドプラグインは dnsmasq プラグインのみです。

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ースを追加する際に、ゾーンで設定されたIPAM プラグインからIPを割り当てることで動作します。IPAMの設定方法の詳細については、[ドキュメントの該当セクション](#)を参照してください。

VM起動時、MACアドレスとIPの対応関係がゾーンのDHCP プラグイン内に作成されます。ネットワークインターフェースが削除されるか、VM/CTが破棄されると、IPAMおよびDHCPサーバー内の該当エントリも削除されます。

#### 注記

一部の機能（IPマッピングの追加/編集/削除）は、現在[PVE IPAMプラグイン](#)使用時のみ利用可能です。

## 12.13.1 設定

Web UI の [ゾーン] パネルでゾーンの高度なオプションから DHCP を有効にすることで、ゾーンの自動 DHCP を有効にできます。

#### 注記

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を有効化した後、ゾーン内のサブネットに対してDHCP範囲を設定する必要があります。設定するには、Vnetsパネルに移動し、DHCP範囲を設定するサブネットを選択します。編集ダイアログの該当タブでDHCP範囲を設定できます。または、以下のCLIコマンドでサブネットのDHCP範囲を設定することも可能です：

```
pvsh set /cluster/sdn/vnets/<vnet>/subnets/<subnet>;
```

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。DHCPプラグインは設定された範囲内でのみIPAMからIPを割り当てます。

[dnsmasq DHCPプラグインのインストール](#)手順も忘れずに行ってください。

## 12.13.2 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、ゾーンで DHCP を有効にしたときに使用されるプラグインです。

#### 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してください。

#### 設定

このプラグインは、dnsmasqが展開される各ゾーンに対して新しいsystemdサービスを作成します。サービスの名前は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。

プラグインは自動的に以下の設定ファイルを /etc/dnsmasq.d/<zone> フォルダに生成します：

#### 00-default.conf

これはdnsmasqインスタンスのデフォルトのグローバル設定を含みます。

**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します。

**10-<zone>-<subnet\_cidr>.ranges.conf**

このファイルは、dnsmasq インスタンスの DHCP 範囲を設定します。

**ethers**

このファイルには、IPAM プラグインからの MAC アドレスと IP の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、このファイルを編集するのではなく、対応する IPAM プラグインを使用してください。このファイルは dnsmasq プラグインによって上書きされます。

上記のファイルは DHCP プラグインによって管理されているため、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、設定フォルダ内に追加ファイル（例：90-custom.conf）を作成してください。これらは dnsmasq DHCP プラグインによって変更されません。

設定ファイルは読み込まれた順序で処理されるため、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ティブの実行順序を制御できます。

DHCP リースは、ファイル /var/lib/misc/dnsmasq.<zone>.leases に保存されます。

PVE IPAM プラグインを使用する場合、DHCP リースを更新、作成、削除することができます。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、現在 DHCP リースの変更はサポートされていません。

## 12.14 ファイアウォール統合

SDN は、ファイアウォールルールの送信元/宛先フィールドで参照できる IPSet を自動的に生成することで、Proxmox VE ファイアウォールと統合します。これは、VNet および IPAM エントリに対して自動的に行われます。

### 12.14.1 VNets およびサブネット

ファイアウォールは、SDN スコープ内で各 VNet に対して以下の IP Sets を自動生成します：

**vnet-all**

VNet 内の全サブネットの CIDR を含む

**vnet-gateway**

VNet 内の全サブネットのゲートウェイの IP アドレスを含む

**vnet-no-gateway**

VNet 内のすべてのサブネットの CIDR を含みますが、ゲートウェイは除外します

**vnet-dhcp**

VNet 内のサブネットで構成されたすべての DHCP 範囲を含みます

構成を変更すると、IPSet は自動的に更新されるため、サブネットの構成を変更する際にファイアウォール ルールを更新する必要はありません。

## 簡易ゾーンの例

以下のVNetとそのサブネットの構成を想定します:

```
# /etc/pve/sdn/vnets.cfg

vnet: vnet0
    zone simple

# /etc/pve/sdn/subnets.cfg: subnet: simple-
192.0.2.0/24
    vnet vnet0
    dhcp-range start-address=192.0.2.100,end-address=192.0.2.199 gateway 192.0.2.1

subnet: simple-2001:db8::/64 vnet vnet0
    dhcp-range start-address=2001:db8::1000,end-address=2001:db8::1999 gateway 2001:db8::1
```

この例では、VNet vnet0 に IPv4 サブネットを設定しました。IP 範囲は 192.0.2.0/24。

ゲートウェイは192.0.2.1、DHCP範囲は192.0.2.100～192.0.2.199 です。

さらに、IPv6サブネットを構成し、IP範囲を2001:db8::/64、ゲートウェイを2001:db8::1、DHCP範囲を2001:db8::1000～2001:db8::1999と設定しました。

vnet0 に対して自動生成される IP セットには、以下の要素が含まれます:

### vnet0-all

- 192.0.2.0/24
- 2001:db8::/64

### vnet0-gateway

- 192.0.2.1
- 2001:db8::1

### vnet0-no-gateway

- 192.0.2.0/24
- 2001:db8::/64
- !192.0.2.1
- !2001:db8::1

### vnet0-dhcp

- 192.0.2.100 ~ 192.0.2.199
- 2001:db8::1000 - 2001:db8::1999

## 12.14.2 IPAM

組み込みのPVE IPAMを使用している場合、ファイアウォールはIPAMにエントリを持つすべてのゲストに対して自動的にIPセットを生成します。ID 100のゲストに対応するIPセットはguest-ipam-100となります。このIPセットには、すべてのIPAMエントリからのIPアドレスが含まれます。したがって、ゲスト100が複数のVNetのメンバーである場合、IPセットにはすべてのVNetからのIPが含まれます。

エントリが追加/更新/削除されると、対応するIPセットもそれに応じて更新されます。



### 警告

ゲストの全エントリを削除する際、自動生成されたIPセットを参照するファイアウォールルールが残っている場合、ファイアウォールは存在しないIPセットを参照するためルールセットの更新に失敗します。

## 12.15 例

このセクションでは、一般的な SDN ユースケースに合わせた複数の構成例を紹介します。利用可能な構成オプションの理解を深めるための追加情報を提供し、具体的な実装例を示すことを目的としています。

### 12.15.1 シンプルなゾーンの例

シンプルなゾーンネットワークは、単一ホスト上のゲストが相互に接続するための分離ネットワークを構築します。

#### ヒント

ゲスト間の接続は、すべてのゲストが同一ホスト上に存在する場合には可能ですが、他のノード上では到達できません。

- simple という名前のシンプルなゾーンを作成します。
- vnet1 という名前の VNet を追加します。
- ゲートウェイと SNAT オプションを有効にしたサブネットを作成します。
- これにより、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させるゲストに割り当て、IP アドレスを設定します。

2台のVMにおけるネットワークインターフェース設定は以下のように見える可能性があります。これにより、10.0.1.0/24ネットワーク経由での通信が可能になります。

```
allow-hotplug ens19iface  
ens19inet static  
    address 10.0.1.14/24
```

```
allow-hotplug ens19iface ens19  
inet staticaddress 10.0.1.15/24  
    アドレス 10.0.1.15/24
```

## 12.15.2 ソースNATの例

シンプルネットワークゾーンのゲストに対して発信接続を許可したい場合、シンプルゾーンはソースNAT（SNAT）オプションを提供します。

上記の設定から開始し、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: 172.16.0.1 SNAT: チェック済み
```

ゲストマシンでは、サブネットのIP範囲内で静的IPアドレスを設定します。

このノード自体はゲートウェイ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向けNATゲートウェイとして機能します。

## 12.15.3 VLAN設定例

異なるノード上の仮想マシン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 タグを使用したネットワークレベルの分離を可能にします。

myvlanzone という名前の VLAN ゾーンを作成します:

```
ID: myvlanzoneBridge:  
vmbr0
```

VLAN タグ 10 と、以前に作成した myvlanzone を使用して、myvnet1 という名前の VNet を作成します。

```
ID: myvnet1 ゾーン:  
myvlanzone タグ: 10
```

メインの SDN パネルから設定を適用し、各ノードにローカルで VNet を作成します。node1 に Debian ベースの仮想マシン (vm1) を作成し、myvnet1 に vNIC を設定します。

この仮想マシンには以下のネットワーク設定を使用してください:

```
auto eth0  
iface eth0 inet static  
    address 10.0.3.100/24
```

ノード2上に2台目の仮想マシン(vm2)を作成し、vm1と同じVNet myvnet1上にvNICを配置する。このVMには以下のネットワーク設定を使用する:

```
auto eth0  
iface eth0 inet static  
    address 10.0.3.101/24
```

この設定後、そのネットワークを使用して両方の仮想マシン間でpingを実行できるようになります。

## 12.15.4 QinQ設定例

この例では、2つのQinQゾーンを設定し、各ゾーンに2台の仮想マシンを追加することで、より分離されたVLANの設定を可能にする追加のVLANタグ層を実証します。

この設定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顧客にVM通信用の分離ネットワークを提供しつつ、他顧客のVMから隔離する場合です。

サービスVLAN 20を持つqinqzone1という名前のQinQゾーンを作成します

```
ID: qinqzone1 ブリッジ:  
vmbr0 サービスVLAN: 20
```

サービスVLAN 30を持つ別のQinQゾーンqinqzone2を作成

```
ID: qinqzone2 ブリッジ:  
vmbr0 サービスVLAN: 30
```

以前に作成した qinqzone1 ゾーンに、VLAN-ID 100 の myvnet1 という VNet を作成します。

```
ID: qinqvnet1 ゾーン:  
qinqzone1 タグ: 100
```

qinqzone2 ゾーンに VLAN-ID 100 の myvnet2 を作成します。

```
ID: qinqvnet2 ゾーン:  
qinqzone2 タグ: 100
```

メインの SDN Web インターフェイスパネルで設定を適用し、各ノードにローカルで VNet を作成します。

4つのDebianベースの仮想マシン(vm1、vm2、vm3、vm4)を作成し、vm1 および vm2 にはブリッジ qinqvnet1 を、vm3 および vm4 にはブリッジ qinqvnet2 をネットワークインターフェースとして追加します。

VM内部で、インターフェースのIPアドレスを設定します(例: /etc/network/interfaces 経由)。

```
auto eth0  
iface eth0 inet static  
    address 10.0.3.101/24
```

4台のVMすべてに、10.0.3.101から10.0.3.104の範囲のIPアドレスを設定してください。

これで、VM vm1 と vm2 間、および vm3 と vm4 間で ping が可能になるはずです。ただし、vm1 と vm2 は異なるゾーン(異なるサービスVLAN)にあるため、vm3 や vm4 を ping することはできません。

## 12.15.5 VXLAN設定例

この例では、3ノードのクラスタを想定し、ノードIPアドレスは192.168.0.1、192.168.0.2、192.168.0.3とします。

myvxlanzone という名前の VXLAN ゾーンを作成し、ノードの全 IP アドレス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、必要に応じて設定してください。

```
ID: myvxlanzone
```

```
ピアアドレスリスト: 192.168.0.1,192.168.0.2,192.168.0.3
```

以前に作成した VXLAN ゾーン myvxlanzone を使用して、vxvnet1 という名前の VNet を作成します。

```
ID: vxvnet1 ゾーン:
```

```
myvxlanzone タグ: 100000
```

メインの SDN Web インターフェイスパネルで設定を適用し、各ノードにローカルで VNet を作成します。node1 に Debian ベースの仮想マシン (vm1) を作成し、vxvnet1 に vNIC を設定します。

この VM には、以下のネットワーク設定を使用します (MTU が低いことに注意してください)。

```
auto eth0
iface eth0 inet static
    address 10.0.3.100/24 mtu 1450
```

ノード3上に2台目の仮想マシン(vm2)を作成し、vm1と同じVNet vxvnet1上にvNICを配置する。このVMには以下のネットワーク設定を使用する:

```
auto eth0
iface eth0 inet static
    address 10.0.3.101/24 mtu 1450
```

その後、vm1とvm2の間でpingが通るはずです。

## 12.15.6 EVPN設定例

この例は、3つのノード (node1、node2、node3) から構成されるクラスタを想定しています。ノードの IP アドレスは、それぞれ 192.168.0.1、192.168.0.2、192.168.0.3 です。

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、EVPN コントローラを作成します。

```
ID: myevpnctl ASN#:
65000
```

myevpnzone という名前の EVPN ゾーンを作成し、以前に作成した EVPN コントローラを割り当て、以下を定義します。

ノード1とノード2を出口ノードとして。

```
ID: myevpnzone
```

```
VRF VXLAN タグ: 10000
```

```
コントローラ: myevpnctlMTU:
1450
```

```
VNet MAC アドレス: 32:F4:05:FE:6C:0A
```

```
出口ノード: node1,node2
```

EVPN ゾーン myevpnzone を使用して、myvnet1 という名前の最初の VNet を作成します。

```
ID: myvnet1 ゾーン:  
myevpnzone タグ: 11000
```

myvnet1上にサブネットを作成:

```
サブネット: 10.0.1.0/24 ゲート  
ウェイ: 10.0.1.1
```

同じ EVPN ゾーン myevpnzone を使用して、myvnet2 という名前の 2 つ目の VNet を作成します。

```
ID: myvnet2 ゾーン:  
myevpnzone タグ: 12000
```

`myvnet2` 上に別のサブネットを作成します:

```
サブネット: 10.0.2.0/24 ゲート  
ウェイ: 10.0.2.1
```

メインのSDNウェブインターフェースパネルから設定を適用し、各ノードにローカルでVNetを作成し、FRR設定を生成します。

node1上にDebianベースの仮想マシン(vm1)を作成し、myvnet1上にvNICを配置する。vm1には以下のネットワーク設定を

使用する:

```
auto eth0  
iface eth0 inet static  
    address 10.0.1.100/24 gateway  
    10.0.1.1  
    mtu 1450
```

ノード2に2台目の仮想マシン(vm2)を作成し、別のVNet myvnet2上にvNICを配置します。vm2には以下のネットワーク設定を使用し

ます:

```
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  
    mtu 1450
```

これでvm1からvm2へ、vm2からvm1へpingが通るはずです。

ゲートウェイではないノード3上のvm2から外部IPにpingを送信すると、パケットは設定済みのmyvnet2ゲートウェイを経由し、出口ノード (node1またはnode2) ヘルーティングされます。その後、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ノードを離れます。

#### 注意

外部ゲートウェイ上のノード1およびノード2に、10.0.1.0/24 および 10.0.2.0/24 ネットワークへの逆ルートを追加する必要があります。これにより、パブリックネットワークからの返信が可能になります。

外部BGPルータを設定している場合、BGP-EVPNルート（この例では10.0.1.0/24および10.0.2.0/24）は動的にアドバンスされます。

## 12.16 注記

### 12.16.1 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、パケットは1つのノードに到着しても別のノードから送信される可能性があるため、`rp_filter`（厳格な逆パスフィルタ）オプションを無効にする必要があります。

/etc/sysctl.conf に以下を追加します:

```
net.ipv4.conf.default.rp_filter=0 net.ipv4.conf.all.rp_filter=0
```

### 12.16.2 VXLAN IPSEC 暗号化

VXLAN上にIPSEC暗号化を追加する場合、この例ではstrongswanの使用方法を示します。

暗号化処理のため、IPv4では追加で60バイト、IPv6では80バイトのMTU削減が必要です。

したがって、デフォルトの物理MTU 1500の場合、1370のMTUを使用する必要があります ( $1370 + 80 \text{ (IPSEC)} + 50 \text{ (VXLAN)} = 1500$ )。ホストにstrongswanをインストールします。

```
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のみを暗号化する必要があります。

```
conn %default
    ike=aes256-sha1-modp1024! (最新ハードウ # 最速でありながら合理的なセキュリティを備えた暗号化方式
        ェアの場合)
    esp=aes256-sha1
    leftfirewall=yes # Proxmox VEを使用する場合に必要です
        ファイアウォールルール

conn 出力
    rightsubnet=%dynamic[udp/4789] right=%any
    type=transport
    authby=psk
    auto=route

conn input
    leftsubnet=%dynamic[udp/4789] type=transport
    認証方式=PSK
    auto=route
```

事前共有鍵を生成するには:

```
openssl rand -base64 128
```

生成した鍵を /etc/ipsec.secrets に追加し、ファイルの内容を以下のようにします:

```
PSK <生成済みbase64キー>;
```

VXLANネットワークに参加するすべてのノードにPSKと設定をコピーします。

## 第13章

# Proxmox VE ファイアウォール

Proxmox VE ファイアウォールは、IT インフラ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に対して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向け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により、この作業が容易になります。

すべての設定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、仮想マシン間の完全な分離を実現します。この分散型システムの性質により、中央集権型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を完全にサポートします。IPv6サポートは完全に透過的であり、デフォルトで両プロトコルのトラフィックをフィルタリングします。したがって、IPv6用に別個のルールセットを維持する必要はありません。

## 13.1 方向とゾーン

Proxmox VE ファイアウォールはネットワークを複数の論理ゾーンにグループ化します。各ゾーンごとに独立したルールを定義可能です。ゾーンに応じて、着信トラフィック、発信トラフィック、転送トラフィックのルールを設定できます。

### 13.1.1 方向

ゾーンのルール定義時に選択可能な方向は3種類です：

#### In

ゾーンに到達するトラフィック。

#### アウト

ゾーンから発信されるトラフィック。

#### 転送

ゾーンを通過するトラフィック。ホストゾーンでは、これはルーティングされたトラフィック（ホストがゲートウェイとして機能している場合やNATを実行している場合）となる可能性があります。VNetレベルでは、これはVNetを通過するすべてのトラフィックに影響し、ブリッジされたネットワークインターフェイスからのトラフィックやそれへのトラフィックも含まれます。

**重要**

転送トラフィック用のルール作成は、現在新しいnftablesベースのproxmox-firewallを使用している場合のみ可能です。転送ルールは標準のpve-firewallでは無視され、効果はありません！

## 13.1.2 ゾーン

ファイアウォールルールを定義できるゾーンは3種類あります：

**ホスト**

ホストからの/ホストへのトラフィック、またはホストによって転送されるトラフィック。このゾーンのルールはデータセンターレベルまたはホストレベルで定義できます。ホストレベルのルールはデータセンターレベルのルールよりも優先されます。

**VM**

VMまたはCTへの/からのトラフィック。転送トラフィックにはルールを定義できず、受信/送信トラフィックのみに適用されます。

**VNet**

SDN VNetを通過するトラフィック（ゲスト間、またはホストとゲスト間の双方向）。このトラフィックは常に転送トラフィックであるため、転送方向のみのルール作成が可能です。

**重要**

VNetレベルでのルール作成は、現在新しいnftablesベースのproxmox-firewallを使用する場合のみ可能です。VNetレベルのルールは標準のpve-firewallでは無視され、効果を発揮しません！

## 13.2 設定ファイル

ファイアウォール関連の全設定はProxmoxクラスタファイルシステムに保存されます。これらのファイルは自動的に全クラスタノードに配布され、pve-firewallサービスは変更時に基盤となるiptablesルールを自動更新します。

GUI（例：データセンター→Firewall、またはノード単位→Firewall）で設定を行うか、お好みのエディタで設定ファイルを直接編集できます。

ファイアウォールの設定ファイルには、キーと値のペアで構成されるセクションが含まれます。#で始まる行と空白行はコメントと見なされます。セクションは、セクション名を角括弧[ ]で囲んだヘッダー行で始まります。

### 13.2.1 クラスタ全体の設定

クラスタ全体のファイアウォール設定は以下に保存されます：

/etc/pve/firewall/cluster.fw

設定には以下のセクションを含めることができます:

#### [OPTIONS]

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されます。

##### **ebtables: &lt;boolean&gt; (デフォルト= 1)**

クラスタ全体で ebtables ルールを有効化します。

##### **enable: &lt;integer&gt; (0 - N)**

クラスタ全体のファイアウォールを有効化または無効化します。

##### **log\_ratelimit: [enable=&lt;1|0&gt; [,burst=&lt;integer&gt;] [,rate=&lt;rate&gt;]]**

ログのレート制限設定

##### **burst= &lt;integer&gt; (0 - N) (デフォルト= 5)**

レートが適用される前に常にログに記録されるパケットの初期バースト

##### **有効化= &lt;boolean&gt; (デフォルト= 1)**

ログレート制限の有効/無効設定

##### **rate= &lt;rate&gt; (デフォルト= 1/秒)**

バーストパケットが補充される頻度

##### **policy\_forward: &lt;ACCEPT| DROP&gt;**

転送ポリシー。

##### **policy\_in: &lt;ACCEPT| DROP| REJECT&gt;**

入力ポリシー。

##### **policy\_out: &lt;ACCEPT| DROP| REJECT&gt;**

出力ポリシー。

#### [RULES]

このセクションには、すべてのノードに適用される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 &lt;name&gt;]

クラスタ全体のIPセット定義。

#### [GROUP &lt;name&gt;]

クラスタ全体のセキュリティグループ定義。

#### [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完全に無効化されているため、ここで有効化オプションを設定する必要があります:

```
[OPTIONS]
# ファイアウォールを有効化（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。例外は、ローカルネットワークからの WebGUI(8006)とssh(22)のみです。

Proxmox VEホストをリモートから管理する場合は、それらのリモート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）の許可も検討してください。

### ヒント

ファイアウォールを有効化する前に、Proxmox VEホストのいずれかに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「管理」という名前のIPSetを作成し、そこにすべてのリモートIPを追加できます。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すべてのファイアウォールルールが作成されます。

## 13.2.2 ホスト固有の設定

ホスト関連の設定は以下から読み込まれます：

```
/etc/pve/nodes/<nodename>/host.fw
```

これは、cluster.fw設定のルールを上書きしたい場合に有用です。ログの詳細レベルを上げたり、netfilter関連のオプションを設定することもできます。設定には以下のセクションを含めることができます：

### [OPTIONS]

ホスト関連のファイアウォールオプションを設定します。

```
enable: <boolean>
      ホストファイアウォールルールを有効化します。
```

```
log_level_forward: <alert| crit| debug| emerg| err| info | nolog | notice | warning>;
      転送トラフィックのログレベル。
```

```
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>;**  
送信トラフィックのログレベル。

**log\_nf\_conntrack: <boolean>; (デフォルト= 0)**  
接続追跡情報のログ記録を有効化します。

**ndp: <boolean>; (デフォルト= 0)**  
NDP（ネイバー検出プロトコル）を有効にする。

**nf\_conntrack\_allow\_invalid: <boolean>; (デフォルト= 0)**  
接続追跡で無効なパケットを許可する。

**nf\_conntrack\_helpers: <string>; (デフォルト= `')**  
特定のプロトコルに対して接続追跡ヘルパーを有効にします。サポートされているプロトコル：amanda、ftp、irc、netbios-ns、pptp、sane、sip、snmp  
、tftp

**nf\_conntrack\_max: <integer>; (32768 - N) (デフォルト= 262144)**  
追跡される接続の最大数。

**nf\_conntrack\_tcp\_timeout\_established: <integer>; (7875 - N) (デフォルト= 432000)**  
確立済み接続のタイムアウト。  
接続追跡 SYN 受信タイムアウト。

**nf\_conntrack\_tcp\_timeout\_syn\_recv: <integer>; (30 - 60) (デフォルト= 60)**  
接続追跡 SYN 受信タイムアウト。

**nftables: <boolean>; (デフォルト= 0)**  
nftablesベースのファイアウォールを有効化 (技術プレビュー)

**nosmurfs: <boolean>;**  
SMURFS フィルタを有効化します。

**protection\_synflood: <boolean>; (デフォルト= 0)**  
synflood 保護を有効化

**protection\_synflood\_burst: <integer>; (デフォルト= 1000)**  
IP 送信元ごとのシンフロッド保護レートバースト。

**protection\_synflood\_rate: <integer>; (デフォルト= 200)**  
IP 送信元ごとの Synflood 保護レート syn/秒。

**smurf\_log\_level: <alert| crit| debug| emerg| err| info| nolog | notice | warning>;**  
SMURFS フィルタのログレベル。

**tcp\_flags\_log\_level:** <alert| crit| debug| emerg| err| info | nolog | notice | warning>;

不正なTCPフラグフィルタのログレベル。

**tcpflags:** <boolean>; (デフォルト= 0)

不正なTCPフラグの組み合わせをフィルタリングします。

#### [RULES]

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ます。

### 13.2.3 VM/コンテナ設定

VMファイアウォール設定は以下から読み込まれます:

/etc/pve/firewall/<VMID>.fw

以下のデータを含みます:

#### [OPTIONS]

これはVM/コンテナ関連のファイアウォールオプションを設定するために使用されます。

**dhcp:** <boolean>; (デフォルト= 0)

DHCPを有効化します。

**有効化:** <boolean>; (デフォルト= 0)

ファイアウォールルールの有効化/無効化。

**ipfilter:** <boolean>;

デフォルトのIPフィルタを有効化します。これは、すべてのインターフェースに対して空のipfilter-net<id>.ipsetを追加するのと同じです。このようなipsetsは、インターフェースのMACアドレスから派生したIPv6リンクローカルアドレスを制限するなど、適切なデフォルト制限を暗黙的に含みます。コンテナの場合、設定されたIPアドレスが暗黙的に追加されます。

**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>;

送信トラフィックのログレベル。

**macfilter:** <boolean>; (デフォルト= 1)

MACアドレスフィルタの有効/無効設定。

**ndp:** <boolean>; (デフォルト= 0)

NDP (Neighbor Discovery Protocol) を有効化します。

```
policy_in: <ACCEPT| DROP| REJECT>;
```

入力ポリシー。

```
policy_out: <ACCEPT| DROP| REJECT>;
```

出力ポリシー。

```
radv: <boolean>;
```

ルータ広告の送信を許可します。

#### [RULES]

このセクションにはVM/コンテナのファイアウォールルールが含まれます。

```
[IPSET <name>;]
```

IPセット定義。

#### [ALIASES]

IPエイリアスの定義。

#### VMおよびコンテナのファイアウォール有効化

各仮想ネットワークデバイスには独自のファイアウォール有効化フラグがあります。これにより、各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化できます。これは、一般的なファイアウォール有効化オプションに加えて必要です。

### 13.2.4 VNet構成

VNet 関連の設定は以下から読み込まれます:

```
/etc/pve/sdn/firewall/<vnet_name>.fw
```

これは、VNet内の各VMごとに個別にファイアウォールルールを設定する必要なく、VNetレベルでファイアウォール設定をグローバルに設定するために使用できます。着信トラフィックや発信トラフィックの概念がないため、FORWARD方向のルールのみを含めることができます。これは、ホストインターフェイスを含む、あるブリッジポートから別のブリッジポートへ移動するすべてのトラフィックに影響します。



#### 警告

この機能は現在、新しいnftablesベースのproxmox-firewallでのみ利用可能です

FORWARDチェーンを通過するトラフィックは双方向であるため、双方向の通信を許可するには両方向のルールを作成する必要があります。例えば特定のホストへのHTTPトラフィックを許可する場合、以下のルールを作成します：

```
FORWARD ACCEPT -dest 10.0.0.1 -dport 80  
FORWARD ACCEPT -source 10.0.0.1 -sport 80
```

### [オプション]

これはVNet関連のファイアウォールオプションを設定するために使用されます。

#### 有効化: &lt;boolean&ampgt (デフォルト= 0)

ファイアウォールルールの有効化/無効化。

```
log_level_forward: &lt;alert| crit| debug| emerg| err| info | nolog | notice | warning&ampgt;  
転送トラフィックのログレベル。
```

```
policy_forward: &lt;ACCEPT| DROP&ampgt;
```

転送ポリシー。

### [RULES]

このセクションには、VNet固有のファイアウォールルールが含まれます。

## 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、OUT、またはFORWARD) とアクション (ACCEPT、DENY、REJECT) で構成されます。マクロ名を指定することも可能です。マクロには事前定義されたルールとオプションのセットが含まれます。ルールは、| を接頭辞として付けることで無効化できます。

#### ファイアウォールルールの構文

```
[RULES]  
  
DIRECTION ACTION [OPTIONS]  
| DIRECTION ACTION [OPTIONS] # 無効化されたルール  
  
DIRECTION MACRO(ACTION) [OPTIONS] # 定義済みマクロを使用
```

以下のオプションを使用してルールの一致を調整できます。

#### --dest &lt;文字列&ampgt;

パケット宛先アドレスを制限します。単一のIPアドレス、IPセット (+ipsetname)、またはIPエイリアス定義を指定できます。20.34.101.207-201.3.9.99のようなアドレス範囲、またはIPアドレスとネットワークのリスト（コンマ区切り）も指定可能です。リスト内でIPv4とIPv6アドレスを混在させないでください。

#### --dport &lt;string&ampgt;

TCP/UDP宛先ポートを制限します。/etc/services で定義されたサービス名または単純な数値 (0-65535) を使用できます。ポート範囲はid+:id+で指定可能（例: 80:85）。複数のポートや範囲をコンマ区切りリストで指定できます。

#### --icmp-type &lt;string&ampgt;

ICMPタイプを指定します。protoがicmpまたはicmpv6/ipv6-icmpの場合にのみ有効です。

```
--iface &lt;string&gt;
ネットワークインターフェース名。VMおよびコンテナにはネットワーク設定キー名 (net<id>) を使用する必要があります。ホスト関連のルールでは任意の文字列を使用できます。
```

```
--log &lt;alert| crit| debug| emerg| err| info| nolog| notice| warning&gt;
ファイアウォールルールのログレベル。
```

```
--proto &lt;string&gt;
IPプロトコル。/etc/protocolsで定義されたプロトコル名 (tcp/udp) または単純な番号を使用できます。
```

```
--source &lt;string&gt;
パケット送信元アドレスを制限します。単一のIPアドレス、IPセット (+ipsetname) 、またはIPエイリアス定義を指定できます。20.34.101.207-201.3.9.99のようなアドレス範囲、またはIPアドレスとネットワークのリスト（コンマ区切り）も指定可能です。リスト内でIPv4とIPv6アドレスを混在させないでください。
```

```
--sport &lt;string&gt;
TCP/UDP送信元ポートを制限します。/etc/servicesで定義されたサービス名または単純な数値（0-65535）を使用できます。ポート範囲は\d+:\d+で指定可能（例：80:85）。複数のポートや範囲を一致させるにはカンマ区切りリストを使用できます。
```

以下に例を示します:

```
[RULES]
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のみ許可
192.168.2.192
IN SSH(ACCEPT) -i net0 -source 10.0.0.1-10.0.0.10 # IP範囲からの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 #← からのSSHを受け入れる ,
IPリスト
IN SSH(ACCEPT) -i net0 -source +mynetgroup # ipset← からのSSHを受け入れる ,
mynetgroup
IN SSH(ACCEPT) -i net0 -source myserveralias #エイリアス← からの SSH を受け入れる ,
myserveralias

| IN SSH(ACCEPT) -i net0 # 無効化されたルール

IN DROP # 全ての着信パケットをドロップ OUT ACCEPT # 全ての発信パケッ
トを許可
```

## 13.4 セキュリティグループ

セキュリティグループは、クラスタレベルで定義されるルールの集合であり、すべての仮想マシンのルールで使用できます。たとえば、「webserver」という名前のグループを定義し、httpおよびhttpsポートを開くルールを設定できます。

```
# /etc/pve/firewall/cluster.fw

[group webserver]
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できます

```
# /etc/pve/firewall/<VMID>.fw

[RULES]
GROUP webserver
```

## 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。その後、それらの名前を参照できます:

- IPセット定義内
- ファイアウォールルールの送信元および宛先プロパティ内

### 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、次のコマンドを使用してください:

```
# pve-firewall localnet local
hostname: 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: 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、クラスタ通信に必要なすべて (corosync、API、SSH) を許可するルールを自動的に設定します。

ユーザーは `cluster.fw` の `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の単一ホストを使用する場合、ローカルIPアドレスを明示的に割り当てる方が望ましいです

```
# /etc/pve/firewall/cluster.fw [ALIASES]
local_network 1.2.3.4 # 単一IPアドレスを使用
```

## 13.6 IPセット

IPセットは、ネットワークや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送信元および宛先プロパティで「+名前」として参照できます。

次の例は、管理 IP セットからの HTTP トラフィックを許可します。

```
IN HTTP(ACCEPT) -source +management
```

### 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（VMファイアウォールではない）にのみ適用されます。これらのIPアドレスは通常の管理タスク（Proxmox VE GUI、VNC、SPICE、SSH）を実行することが許可されています。

ローカルクラスタネットワークは自動的にこのIPセットに追加されます（別名 `cluster_network`）。これによりホスト間クラスタ通信が可能になります（マルチキャスト、SSHなど）。

```
# /etc/pve/firewall/cluster.fw

[IPSET management]
192.168.2.10
192.168.2.10/24
```

### 13.6.2 標準 IP セットのブラックリスト

これらのIPからのトラフィックは、すべてのホストおよびVMのファイアウォールによって破棄されます。

```
# /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\*

これらのフィルタは仮想マシンのネットワークインターフェースに属し、主にIPスプーフィングの防止に使用されます。インターフェースに対してこのようなセットが存在する場合、送信元IPがそのインターフェースに対応するipfilterセットと一致しないすべての送信トラフィックは破棄されます。

設定済みIPアドレスを持つコンテナの場合、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オプションタブの「一般IPフィルタ」オプションで有効化されている場合）、関連付けられたIPアドレスが暗黙的に含まれます。

仮想マシンとコンテナの両方において、これらのセットにはネイバーディスカバリプロトコルを機能させるため、標準的なMACアドレスから派生したIPv6リンク

```
/etc/pve/firewall/<VMID>.fw

[IPSET ipfilter-net0] # net0 では指定された IP のみ許可 192.168.2.10
ローカルアドレスも暗黙的に含まれます。
```

## 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します：

- pvefw-logger: NFLOGデーモン（ulogdの代替）。
- pve-firewall: iptables ルールを更新

また、ファイアウォールサービスの起動/停止に使用できるpve-firewallというCLIコマンドも存在します：

```
# pve-firewall start # pve-
firewall stop
```

状態を確認するには以下を使用します：

```
# 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みコンパイルするため、ファイアウォール設定にエラーが含まれている場合は警告が表示されます。

生成されたiptablesルールを確認したい場合は以下を使用できます：

```
# iptables-save
```

## 13.8 デフォルトのファイアウォールルール

デフォルトのファイアウォール設定では以下のトラフィックがフィルタリングされます：

### 13.8.1 データセンターへの/からのトラフィック 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ホストに対して以下のトラフィックは許可されます：

- ループバックインターフェース経由のトラフィック
- 既に確立された接続
- IGMPプロトコルを使用したトラフィック
- 管理ホストからポート8006へのTCPトラフィック（Webインターフェースへのアクセスを許可するため）
- 管理ホストからポート範囲5900～5999へのTCPトラフィック（VNCウェブコンソール用トラフィックを許可）
- 管理ホストからポート3128へのTCPトラフィック（SPICEプロキシへの接続用）
- 管理ホストからポート22へのTCPトラフィック（SSHアクセスを許可するため）
- クラスタネットワーク内のUDPトラフィック（ポート5405-5412宛て）はcorosync用
- クラスタネットワーク内の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（宛先到達不能）、4（輻輳制御）、または11（タイムアウト）以下のトラフィックは、ログ記録が有効な場合でも記録されず破棄されます：

- 無効な接続状態のTCP接続
- corosyncに関連しないブロードキャスト、マルチキャスト、エニキャストトラフィック（ポート5405-5412経由でないもの）

- ポート43へのTCPトラフィック
- ポート135および445へのUDPトラフィック
- ポート範囲 137 から 139 への UDP トラフィック
- 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

それ以外のトラフィックは、それぞれ破棄または拒否され、ログにも記録されます。これは、**ファイアウォール→オプション**で有効化された**追加オプション**（NDP、SMURFS、TCPフラグフィルタリングなど）によって異なる場合があります。

以下の出力内容を確認してください

```
# iptables-save
```

システムコマンドの出力を確認し、システム上で有効なファイアウォールチェーンとルールを確認してください。この出力は、Web GUIのノードサブスクリプションタブからアクセス可能なシステムレポート、またはpverereportコマンドラインツールを通じて入手可能です。

### 13.8.2 VM/CT 着信/発信 DROP/REJECT

設定に応じてDHCP、NDP、ルータ広告、MACおよびIPフィルタリングに対する例外を除き、VMへの全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/拒否するルールはデータセンターから継承されますが、ホストの許可された受信/送信トラフィックに対する例外は適用されません。

再度、適用されている全ルールとチェーンを確認するには、[iptables-save](#)（上記参照）を使用できます。

### 13.9 ファイアウォールルールのログ記録

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グ記録はすべて無効になっています。ログ記録を有効にするには、**ファイアウォール→オプション**で、受信トラフィックおよび/または送信トラフィックのログレベルを設定する必要があります。これはホスト単位でも、VM/CT ファイアウォール単位でも個別に設定可能です。これにより、Proxmox VEの標準ファイアウォールルールのログ記録が有効化され、出力は「**ファイアウォール→ログ**」で確認できます。さらに、標準ルールでは一部のドロップまたは拒否されたパケットのみが記録されます（[デフォルトのファイアウォールルール](#)を参照）。

`loglevel` はフィルタリングされたトラフィックのログ記録量に影響しません。ログ出力に接頭辞として付加される LOGID を変更し、フィルタリングや後処理を容易にするものです。

`loglevel` は以下のフラグのいずれかです:

loglevel	LOGID
nolog	—
emerg	0
alert	1
crit	2

ログレベル	ログID
err	3
警告	4
通知	5
情報	6
デバッグ	7

典型的なファイアウォールログの出力は次のようにになります:

```
VMID LOGID CHAIN TIMESTAMP POLICY: PACKET_DETAILS
```

ホストファイアウォールの場合は、VMIDは0に等しい。

### 13.9.1 ユーザー定義ファイアウォールルールのログ記録

ユーザー定義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には、各ルールごとに個別にログレベルパラメータを設定できます。これにより、[ファイアウォールオプション](#)で定義された標準ルールのログレベルとは独立して、きめ細かくログを記録することができます。

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 Web UI で簡単に定義または変更できますが、対応する `pvsh` API 呼び出しでも設定できます。

さらに、ファイアウォール設定ファイルに `-log <loglevel>` を追加することで設定可能ですが（[設定可能なログレベル](#)を参照）。

例えば、以下の2つは同等です：

```
IN REJECT -p icmp -log nolog IN REJECT -p
icmp
```

一方

```
IN REJECT -p icmp -log debug
```

デバッグレベルでフラグ付けされたログ出力を生成します。

## 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と複数の動的ポートを使用する旧式プロトコルです。ポート21を受け入れるルールが必要です。さらに、`ip_conntrack_ftp`モジュールをロードする必要があります。以下のコマンドを実行してください:

```
modprobe ip_conntrack_ftp
```

また、再起動後も有効にするため、`/etc/modules` に `ip_conntrack_ftp` を追加してください。

## 13.10.2 Suricata IPS統合

Suricata IPS（侵入防止システム）を使用したい場合、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。

ファイアウォールで拒否/破棄されたパケットはIPSに送信されません。Proxmoxホスト

にSuricataをインストール:

```
# apt-get install suricata # modprobe  
nfnetlink_queue
```

次回再起動時に備え、/etc/modulesにnfnetlink\_queueを追加することを忘れないでください。その後、特定のVMに対してIPSを有効化します:

```
# /etc/pve/firewall/<VMID>.fw  
  
[オプション]  
ips: 1  
ips_queues: 0
```

ips\_queuesはこのVMに特定のCPUキューをバインドします。利用可能なキュー

は

```
# /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いくつかのIPv6固有のオプションが含まれています。注意すべき点として、IPv6はもはやARPプロトコルを使用せず、代わりにIPレベルで動作するNDP（Neighbor Discovery Protocol）を使用します。そのため、NDPの成功にはIPアドレスが必要です。この目的のために、インターフェースのMACアドレスから派生したリンクローカルアドレスが使用されます。デフォルトでは、ホストレベルとVMレベルの両方でNDPオプションが有効化されており、ネイバーディスカバリー（NDP）パケットの送受信を許可しています。

ネイバーディスカバリー以外にも、NDPは自動設定やルータのアドバタイズなど、いくつかの用途に使用されます。

デフォルトでは、仮想マシンはルータ要求メッセージ（ルータの問い合わせ）の送信と、ルータ広告パケットの受信が許可されています。これによりステートレス自動設定が利用可能です。一方、仮想マシンが自身をルータとして広告するには、「ルータ広告を許可」（radv: 1）オプションの設定が必要です。

NDPに必要なリンクローカルアドレスについては、「IPフィルタ」（ipfilter: 1）オプションを有効化できます。これは、各VMのネットワークインターフェースに対応するリンクローカルアドレスを含むipfilter-net\*ipsetを追加するのと同じ効果を持ちます（詳細は「[標準IPセット ipfilter-net\\*](#)」セクションを参照）。

## 13.12 Proxmox VEが使用するポート

- Webインターフェース: 8006 (TCP, HTTP/1.1 over TLS)

- VNC Webコンソール: 5900-5999 (TCP, WebSocket)
- SPICEプロキシ: 3128 (TCP)
- sshd (クラスタ操作に使用) : 22 (TCP)
- rpcbind: 111 (UDP)
- sendmail: 25 (TCP, 送信)
- corosyncクラスタトラフィック: 5405-5412 UDP
- ライブマイグレーション (VMメモリおよびローカルディスクデータ) : 60000-60050 (TCP)

## 13.13 nftables

pve-firewall の代替として、iptables ではなく新しい **nftables** に基づく Proxmox VE ファイアウォールの実装である proxmox-firewall を提供しています。



### 警告

proxmox-firewall は現在テクニカルレビュー段階です。バグや元のファイアウォールとの互換性問題が発生する可能性があります。現時点では本番環境での使用には適していません。

この実装は同じ設定ファイルと設定形式を使用するため、切り替え時に既存の設定を流用できます。以下の例外を除き、完全に同一の機能を提供します：

- Linuxブリッジを使用する場合、追加のファイアウォールブリッジ(fwbrX)は作成されません。OVSブリッジを使用するゲストインターフェースには、引き続きファイアウォールブリッジが存在します。
- ゲストトラフィックに対してREJECTは現在使用できません（代わりにトラフィックはドロップされます）。
- NDP、ルータアドバタイズメント、またはDHCPオプションを使用すると、デフォルト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、接続追跡テーブルエントリを持つ接続に対しても評価されます。

### 13.13.1 インストールと使用方法

proxmox-firewallパッケージをインストールします:

```
apt install proxmox-firewall
```

ホストのWeb UI (ホスト> ファイアウォール> オプション> nftables) またはホストの設定ファイル (`/etc/pve/nodes/<node_name>/host.fw`) で

```
[OPTIONS]
```

```
nftables: 1
```

nftablesバックエンドを有効化します：

#### 注意

`proxmox-firewall` の有効化/無効化後、古い/新しいファイアウォールが正常に動作するには、実行中のすべての VM とコンテナを再起動する必要があります。

nftables設定キーを設定後、新しい`proxmox-firewall`サービスが引き継ぎます。新しいサービスが動作しているかどうかは、`proxmox-firewall`の`systemctl`

```
systemctl status proxmox-firewall  
statusを確認することで確認できます:
```

生成されたルールセットを確認することもできます。詳細は「[便利なコマンド](#)」セクションを参照してください。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。関連するコマンドは「[サービスとコマンド](#)」セクションに記載されています。

古いファイアウォールに戻すには、設定値を0 / Noに戻すだけで済みます。

### 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 つのテーブルを作成します。`proxmox-firewall` と `proxmox-firewall-guests` です。Proxmox VE ファイアウォール設定の外部でカスタムルールを作成したい場合は、独自のテーブルを作成してカスタムファイアウォールルールを管理できます。`proxmox-firewall` は自身が生成したテーブルのみを操作するため、独自のテーブルを追加することで `proxmox-firewall` の動作を容易に拡張・変更できます。

nftablesベースのファイアウォールでは、`pve-firewall`コマンドの代わりに`proxmox-firewall`を使用します。これは`systemd`サービスであるため、

```
systemctl start proxmox-firewall  
systemctl stop proxmox-firewall  
systemctl経由で起動・停止できます:
```

ファイアウォールサービスを停止すると、生成されたすべてのルールが削除されます。

ファイアウォールの状態を照会するには、`systemctl` サービスの状態を照会できます:

```
systemctl status proxmox-firewall
```

### 13.13.3 便利なコマンド

生成されたルールセットは次のコマンドで確認できます:

```
nft list ruleset
```

`proxmox-firewall` をデバッグしたい場合、`compile` サブコマンドでファイアウォールが生成するコマンドをダンプできます。さらに、`PVE_LOG` 環境変数を設定するとログ出力が `STDERR` に出力され、`nftables` ルールセット生成時の問題のデバッグに役立ちます:

```
PVE_LOG=trace /usr/libexec/proxmox/proxmox-firewall compile> firewall.json
```

nftablesルールセットは、`proxmox-firewall`バイナリに含まれるスケルトンルールセットと、ファイアウォール設定から生成されたルールで構成されます。基本ルールセットは`skeleton`サブコマンドで取得できます:

```
/usr/libexec/proxmox/proxmox-firewall skeleton
```

両コマンドの出力をnft実行ファイルに直接パイプできます。以下のコマンドはnftablesルールセット全体を最初から再作成します:

```
/usr/libexec/proxmox/proxmox-firewall skeleton| nft -f -
/usr/libexec/proxmox/proxmox-firewall compile| nft -j -f -
```

ファイアウォールデーモンが実行中の詳細出力を表示したい場合は、systemctl サービスを編集することもできます:

```
systemctl edit proxmox-firewall
```

次に、PVE\_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:

```
[Service] Environment="PVE_LOG=trace"
```

これにより非常に短時間で大量のログが生成されるため、デバッグ目的でのみ使用してください。他の、より簡潔なログレベルとしてはinfoとdebugがあります。

ファイアウォールルールのデバッグには、異なるチェーンを通るパケットの流れを追跡することが有用です。追跡したいパケットに対してnftraceを1に設定することで実現できます。すべてのパケットにこのフラグを設定しないことを推奨します。以下の例ではICMPパケットのみを調査します。

```
#!/usr/sbin/nft -f table bridge
tracebridge
delete table bridge tracebridge

table bridge tracebridge { chain trace {
    meta l4proto icmp meta nftrace set 1
}

chain prerouting {
    type filter hook prerouting priority -350; policy accept; jump trace
}

chain postrouting {
    type filter hook postrouting priority -350; policy accept; jump trace
}
}
```

このファイルを保存し、実行可能にして一度実行すると、対応するトレースチェーンが作成されます。その後、Proxmox VE Web UI（ファイアウォール > ログ）またはnft monitor trace コマンドでトレース出力を確認できます。

上記の例は、ゲストトラフィックが通常通過するすべてのブリッジ上のトラフィックをトレースします。ホストトラフィックを調査したい場合は、bridgeテーブルではなくinetテーブルにそれらのチェーンを作成してください。

#### 注意

これにより大量のログスパムが生成され、ネットワークスタックのパフォーマンスが大幅に低下する可能性があることに注意してください。

トレースルールは次のコマンドで削除できます:

```
nft delete table bridge tracebridge
```

## 第14章

### ユーザー管理

Proxmox VEは複数の認証ソースをサポートしています。例えば、Linux PAM、統合されたProxmox VE認証サーバー、LDAP、Microsoft Active Directory、OpenID Connectなどです。

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および権限管理を使用することで、きめ細かいアクセスを定義できます。

#### 14.1 ユーザー

Proxmox VE はユーザー属性を `/etc/pve/user.cfg` に保存します。パスワードはここに保存されず、代わりにユーザーは後述の[認証領域](#)に関連付けられます。したがって、ユーザーは内部では `<userid>@<realm>` の形式でユーザー名と領域によって識別されることがよくあります。

このファイル内の各ユーザーエントリには以下の情報が含まれます：

- 名
- 姓
- メールアドレス
- グループ所属
- 任意の有効期限
- このユーザーに関するコメントまたはメモ
- このユーザーを有効にするか無効にするか
- 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、あるいは設定された有効期限が過去の日付である場合、当該ユーザーは新規セッションへのログインや新規タスクの開始ができなくなります。ただし、当該ユーザーによって既に開始されたタスク（例：ターミナルセッション）は、これらの操作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レルム経由でログイン可能であり、制限のない管理者権限を持ちます。このユーザーは削除できませんが、属性の変更は可能です。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宛に送信されます。

## 14.2 グループ

各ユーザーは複数のグループに所属できます。グループはアクセス権限を整理する推奨方法です。個々のユーザーではなく、常にグループに権限を付与してください。これにより、はるかに保守性の高いアクセス制御リストが得られます。

## 14.3 APIトークン

APIトークンは、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の大部分へステータスにアクセスすることを可能にします。トークンは個々のユーザー向けに生成でき、アクセス範囲と期間を制限するために個別の権限と有効期限を設定できます。APIトークンが不正利用された場合、ユーザー自体を無効化することなくトークンを無効化できます。

APIトークンには基本的に2種類あります：

- 分離された権限: トークンにはACLによる明示的なアクセス権限の付与が必要です。有効な権限はユーザー権限とトークン権限の交差によって計算されます。
- フル権限: トークンの権限は関連付けられたユーザーのものと同一です。



### 注意

トークン値は生成時に一度だけ表示/返されます。後からAPI経由で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 *Authorization* を *PVEAPIToken=USER@* の形式で表示された値に設定するか、API リクエスト時に API クライアントのドキュメントを参照してください。

の形式で設定するか、APIクライアントのドキュメントを参照してください。

## 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、およびストレージデバイスからなる集合体です。特定のユーザーが特定のリソースセットへのアクセスを制限される必要がある場合、権限処理に有用です。これは、リソースごとに管理する必要がなく、単一の権限を要素の集合体に適用できるためです。リソースプールはグループと組み合わせて使用されることが多く、グループのメンバーがマシンとストレージの集合体に対する権限を持つようにします。

## 14.5 認証レルム

Proxmox VEのユーザーは外部レルムに存在するユーザーの対応物であるため、レルムは/etc/pve/domains.cfgで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す。これらのユーザーはadduserなどのコマンドでホストシステム上に作成されます。Proxmox VEホストシステム上にPAMユーザーが存在する場合、対応するエントリをProxmox VEに追加することで、これらのユーザーが自身のシステムユーザー名とパスワードでログインできるようになります。

### 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あり、ハッシュ化されたパスワードを/etc/pve/priv/shadow.cfgに保存します。パスワードはSHA-256ハッシュアルゴリズムでハッシュ化されます。Proxmox VE外部へのアクセスを必要としないユーザー環境において、小規模（あるいは中規模）のインストールに最も適した領域です。この場合、ユーザーは完全にProxmox VEによって管理され、GUI経由で自身のパスワードを変更できます。

### LDAP

LDAP (Lightweight Directory Access Protocol) は、ディレクトリサービスを使用した認証のための、オープンなクロスプラットフォームプロトコルです。OpenLDAP は、LDAP プロトコルの人気のあるオープンソース実装です。

### Microsoft Active Directory (AD)

Microsoft Active Directory (AD) は Windows ドメインネットワーク向けのディレクトリサービスであり、Proxmox VE の認証領域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OAuth 2.0プロトコル上に実装されたIDレイヤーです。外部認証サーバーによる認証に基づいて、クライアントがユーザーの身元を確認することを可能にします。

### 14.5.1 Linux PAM標準認証

Linux PAMはホストシステムの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ている各ノード上にシステムユーザーが存在する必要があります。ユーザーは通常のシステムパスワードで認証します。この領域はデフォルトで追加され、削除できません。

GUIまたは/access/password APIエンドポイント経由でのパスワード変更は、ローカルノードにのみ適用され、クラスター全体には適用されません。Proxmox VEはマルチマスター設計を採用していますが、ノードごとに異なるパスワードを使用することは、依然としてセキュリティ上の利点を提供します。

設定面では、管理者はこのレルムからのログインに二要素認証を要求するよう選択でき、またこのレルムをデフォルト認証レルムとして設定できます。

### 14.5.2 Proxmox VE 認証サーバー

Proxmox VE 認証サーバー領域は、シンプルな Unix ライクなパスワードストアです。この領域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、領域のユーザーに二要素認証を要求する機能と、ログインのデフォルト領域として設定する機能のみです。

他のProxmox VE レルムタイプとは異なり、ユーザーは別のシステムに対して認証するのではなく、完全に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を作成する際にはパスワードの設定が必須となります。

### 14.5.3 LDAP

ユーザー認証には外部LDAPサーバー（例：OpenLDAP）も利用可能です。このレルムタイプでは、`ユーザー属性名 (user_attr)` フィールドで指定されたユーザー名属性を使用し、`ベースドメイン名 (base_dn)` 下でユーザーを検索します。

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でき、接続はSSL経由で暗号化可能です。さらに、ディレクトリやグループに対するフィルタを設定できます。フィルタを使用すると、レルムの適用範囲をさらに制限できます。

例えば、ユーザーが以下のLDIFデータセットで表現される場合：

```
# ldap-test.com の People にある user1
dn: uid=user1,ou=People,dc=ldap-test,dc=com objectClass: top
objectClass: person
objectClass: organizationalPerson objectClass: inetOrgPerson
uid: user1
cn: Test User lsn:
Testers
description: これは最初のテストユーザーです。
```

基底ドメイン名は`ou=People,dc=ldap-test,dc=com`となり、ユーザー属性は`uid`となります。

Proxmox VE がユーザーを照会および認証する前に LDAP サーバーへの認証（バインド）が必要な場合、`/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/realm/<realmname>.pw`（例：`/etc/pve`）に保存する必要があります。  
このファイルには、パスワードを1行で記述してください。

証明書を検証するには、`capath` を設定する必要があります。これは、LDAP サーバーの CA 証明書に直接設定するか、信頼されたすべての CA 証明書を含むシステムパス（`/etc/ssl/certs`）に設定することができます。さらに、`verify` オプションも設定する必要がありますが、これは Web インターフェースからも行うことができます。

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す：

- `Realm (realm)`: Proxmox VE ユーザー用のレルム識別子
- `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に使用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
- `フォールバックサーバー (server2)`: プライマリサーバーに接続できない場合の代替サーバーアドレス（オプション）
- `ポート (port)`: LDAPサーバーが待機するポート

**注記**

特定のユーザーがLDAPサーバーを使用して認証できるようにするには、Proxmox VEサーバーからそのユーザーを当該レルムのユーザーとして追加する必要があります。これは[同期により](#)自動的に実行できます。

### 14.5.4 Microsoft Active Directory (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ります。Active Directoryは、オプションのフォールバックサーバー、ポート、SSL暗号化など、LDAPと同様のほとんどのプロパティをサポートしています。さらに、設定後には[同期](#)操作を通じてユーザーをProxmox VEに自動的に追加できます。

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、

`bind_dn` プロパティを設定する必要があります。このプロパティはMicrosoft ADでは通常デフォルトで必要です。

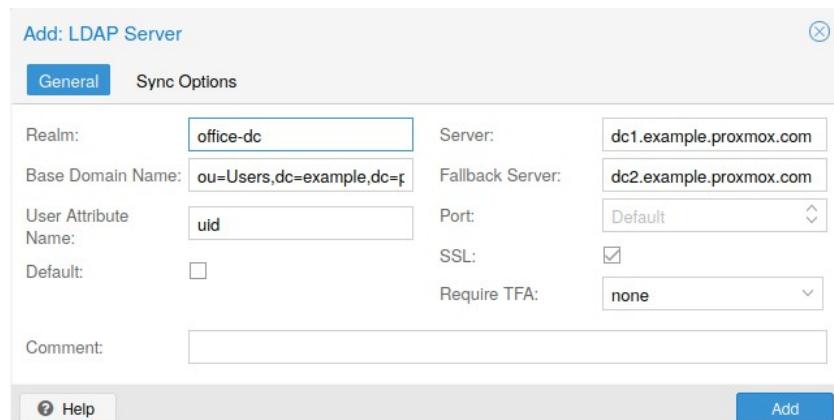
Microsoft Active Directoryの主な設定項目は以下の通りです：

- レルム (realm): Proxmox VE ユーザー向けのレルム識別子
- ドメイン (domain): サーバーの AD ドメイン
- サーバー (server1): サーバーの完全修飾ドメイン名 (FQDN) または IP アドレス
- フォールバックサーバー (server2): プライマリサーバーに到達できない場合のオプションのフォールバックサーバーアドレス
- ポート (port): Microsoft AD サーバーが待機するポート

**注**

Microsoft ADは通常、ユーザー名などの値を大文字小文字を区別せずにチェックします。Proxmox VEでも同様に動作させるには、Web UIでレルムを編集するか、CLIを使用してデフォルトの大文字小文字区別オプションを無効にできます (IDをレルムIDに置き換えてください) :  
`pveum realm modify ID --case-sensitive 0`

### 14.5.5 LDAPベースのレルムの同期



LDAPベースのレルム (LDAPおよびMicrosoft Active Directory) では、ユーザーとグループを手動でProxmox VEに追加する代わりに、自動同期が可能です。同期オプションは、Webインターフェースの認証パネルにある追加/編集ウィンドウ、またはpveumのrealm追加/変更コマンドからアクセスできます。

コマンドからアクセスできます。その後、GUIの認証パネルから、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 realm sync &lt;realm&gt;
```

ユーザーとグループはクラスター全体の設定ファイル `/etc/pve/user.cfg` に同期されます。

#### 属性からプロパティへ

同期応答にユーザー属性が含まれる場合、それらは

例: `firstname` や `lastname`。

属性の名前が Proxmox VE のプロパティと一致しない場合、config 内で `sync_attributes` オプションを使用してカスタムのフィールド間マッピングを設定できます。

プロパティが消失した場合の処理方法は、以下の同期オプションで制御できます。

#### 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、[追加/編集]ウィンドウの[同期オプション]タブにあります。

設定オプションは以下の通りです：

- **バインドユーザー** (`bind_dn`): ユーザーとグループをクエリするために使用される LDAP アカウントを指します。このアカウントは、必要なすべてのエントリへのアクセス権が必要です。設定されている場合、検索はバインドを介して実行されます。設定されていない場合、検索は匿名で実行されます。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: `cn=admin,dc=example,dc=com`。
- **グループ名属性** (`group_name_attr`): ユーザーの所属グループを表します。`user.cfg` の通常の文字数制限に準拠したエントリのみ同期されます。グループ名は命名衝突を避けるため、名前に「`-$realm`」を付加して同期されます。手動で作成したグループが同期によって上書きされないようご注意ください。
- **ユーザークラス** (`user_classes`): ユーザーに関連付けられたオブジェクトクラス。
- **グループクラス** (`group_classes`): グループに関連付けられたオブジェクトクラス。
- **E-Mail 属性**: LDAPベースのサーバーがユーザーのメールアドレスを指定している場合、関連する属性をここで設定することで同期に含めることができます。コマンドラインからは`--sync_attributes` パラメータで実現可能です。
- **ユーザーフィルター** (`filter`): 特定のユーザーを対象とする追加のフィルターオプション。
- **グループフィルター** (`group_filter`): 特定のグループを対象とする追加のフィルターオプション。

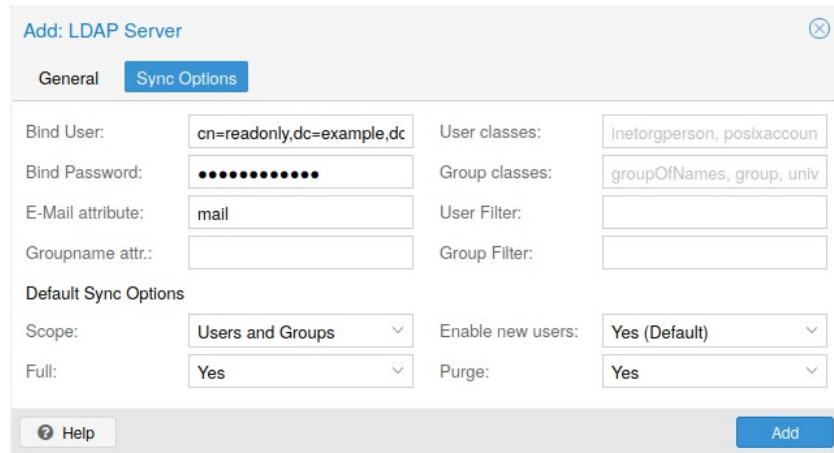
---

#### 注記

フィルターを使用すると、同期の範囲を絞り込むための追加の一致条件セットを作成できます。利用可能なLDAPフィルタータイプとその使用方法については、[ldap.com](#)で確認できます。

---

## 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、同期操作の動作を記述するさらなるオプションも設定できます。

これらのオプションは、同期前にパラメータとして設定するか、realm オプション sync-defaults- を通じてデフォルト値として設定します。

同期の主なオプションは以下の通りです：

- **範囲 (scope)**: 同期対象の範囲。ユーザー、グループ、またはその両方を選択できます。
- **新規有効化 (enable-new)** : 設定すると、新規同期されたユーザーは有効化されログイン可能になります。デフォルトは true。
- **Remove Vanished (remove-vanished)**: アクティビ化すると、同期応答から返されなかった場合に削除するかどうかを決定するオプションのリストです。オプションは以下の通りです：
  - **ACL (acl)**: 同期レスポンスで返されなかったユーザーおよびグループのアクセス制御リスト (ACL) を削除します。  
通常はEntryと併用します。
  - **エントリ (entry)**: 同期応答で返されなかったエントリ (ユーザーおよびグループ) を削除します。
  - **プロパティ (properties)**: 同期応答内のユーザーがそれらの属性を含んでいなかった場合、エントリのプロパティを削除します。これには、同期によって一度も設定されたことがないプロパティも含みます。例外はトークンと有効化フラグであり、このオプションが有効な場合でもこれらは保持されます。
- **プレビュー (dry-run)**: 設定ファイルへの書き込みは行われません。ユーザーとグループが user.cfg に同期されるかどうかを確認したい場合に便利です。

## 予約文字

特定の文字は予約済み ([RFC2253](#)参照) であり、DN内の属性値で使用するには適切にエスケープ処理が必要です。

以下の文字はエスケープが必要です：

- 先頭または末尾のスペース ( )
- 先頭のシャープ記号 (#)
- コンマ (,)

- プラス記号 (+)
- 二重引用符 ("")
- スラッシュ (/)
- 山括弧 (<>)
- セミコロン (;)
- 等号 (=)

DN内でこのような文字を使用するには、属性値を二重引用符で囲みます。例えば、CN（共通名）が「Example, User」のユーザーとバインドするには、CN（共通名）がExample, Userのユーザーとバインドするには、bind\_dnの値としてCN="Example, User",OU=people,DC=exampleをbind\_dnの値として指定します。

これはbase\_dn、bind\_dn、group\_dn属性に適用されます。

#### 注意

ユーザー名ではコロンとスラッシュは予約文字であるため、これを含むユーザーは同期できません。

## 14.5.6 OpenID Connect

主なOpenID Connect設定オプションは以下の通りです：

- 発行者URL (issuer-url): これは認証サーバーのURLです。Proxmox VEはOpenID Connect Discoveryプロトコルを使用して、詳細設定を自動的に行います。  
暗号化されていないhttp:// URLの使用も可能ですが、暗号化された  
https://接続の使用を強く推奨します。
- レルム (realm): Proxmox VEユーザー用のレルム識別子
- 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。
- ユーザー自動作成 (autocreate): ユーザーが存在しない場合に自動的に作成します。認証は OpenID サーバーで行われますが、すべてのユーザーは Proxmox VE ユーザー設定にエントリが必要です。手動で追加するか、自動作成オプションを使用して新規ユーザーを自動的に追加できます。
- ユーザー名クレーム (username-claim): 一意のユーザー名（サブジェクト、  
ユーザー名またはメールアドレス）を生成するために使用されるOpenIDクレーム。
- グループの自動作成 (groups-autocreate): 既存のPVEグループを使用せず、クレーム内の全グループを作成する（デフォルト動作）。
- グループクレーム (groups-claim): IDトークンまたはuserinfoエンドポイントからグループを取得するために使用されるOpenIDクレーム。
- グループの書き換え (groups-overwrite): 既存のグループに追加する代わりに、ユーザーに割り当てられたすべてのグループを上書きします（デフォルトの動作）。

## ユーザー名マッピング

OpenID Connect仕様では、`subject`という単一の一意の属性（OpenID用語ではクレーム）が定義されています。デフォルトでは、この属性の値に@とrealm名を追加するだけでProxmox VEユーザー名を生成します： `${subject}@${realm}`。

残念ながら、ほとんどのOpenIDサーバーは件名にDGH76OKH34BNG3245SBのようなランダムな文字列を使用するため、典型的なユーザー名はDGH76OKH34BNG3245SB@yourrealmのようになります。一意ではありますが、人間がこのようなランダムな文字列を記憶するのは困難であり、実際のユーザーとこれを関連付けることはほぼ不可能です。

ユーザー名クレーム設定により、ユーザー名マッピングに他の属性を使用できます。OpenID Connectサーバーがその属性を提供し一意性を保証している場合、OpenID Connectサーバーがその属性を提供し、一意性を保証している場合は、`username`を優先的に使用することをお勧めします。

別の選択肢として電子メールを使用する方法があり、これも人間が読み取り可能なユーザー名を生成します。ただし、この設定もサーバーがこの属性の唯一性を保証する場合にのみ使用してください。

## グループマッピング

OpenID設定で`groups-claim`設定を指定すると、グループマッピング機能が有効になります。`groups-claim`で提供されるデータは、ユーザーがProxmox VEで所属すべきグループに対応する文字列のリストである必要があります。衝突を防止するため、OpenIDクレームのグループ名には「-<realm名>」が接尾辞として付加されます（例：realm「oidc」内のOpenIDグループ名「my-openid-group」の場合、Proxmox VE内のグループ名は「my-openid-group-oidc」となります）。

OpenIDプロバイダーから報告されたグループで、Proxmox VEに存在しないものはデフォルトで無視されます。OpenIDプロバイダーから報告されたすべてのグループをProxmox VEに存在させる必要がある場合、`groups-autocreate`オプションを使用してユーザーイン時にこれらのグループを自動作成できます。

デフォルトでは、グループはユーザーの既存グループに追加されます。ユーザーが既にProxmox VEでメンバーとなっているグループを、OpenIDプロバイダーからのグループで上書きしたい場合があります。`groups-overwrite`設定を有効にすると、OpenIDプロバイダーから報告されたグループを追加する前に、Proxmox VEからユーザーのグループをすべて削除します。

場合によっては、OpenIDサーバーがProxmox VEのグループIDとして無効な文字を含むグループクレームを送信することがあります。Proxmox VEのグループ名で許可されていない文字を含むグループはすべて除外され、警告がログに記録されます。

## 詳細設定

- ユーザー情報エンドポイントのクエリ (`query userinfo`): このオプションを有効にすると、OpenID Connect認証システムがクレーム値を取得するために「userinfo」エンドポイントをクエリします。このオプションを無効にすると、「userinfo」エンドポイントをサポートしない一部のIDプロバイダー（例: ADFS）で有用です。

## 例

Googleを使用してOpenID レルムを作成する例を以下に示します。`--client-id`および`--client-key`は、Google OpenID 設定の値に置き換えてください。

```
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/>) は人気のオープンソースID管理ツールで、OpenID Connectをサポートしています。以下の例では、`--issuer-url` と `--client-id` を自身の情報に置き換える必要があります:

```
pveum realm add myrealm2 --type openid --issuer-url https://your.server:8080/realms/your-realm --client-id XXX --username-claim username  
--username-claim ユーザー名を使用すると、Proxmox VE 側で example.u のようなシンプルなユーザー名を有効にできます。
```



#### 警告

ユーザーが（Keycloakサーバー上で）自分でユーザー名設定を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

二要素認証の利用方法は2通りあります：

認証領域で必須設定とし、TOTP（時間ベースのワンタイムパスワード）またはYubiKey OTPのいずれかを使用する方法。この場合、新規作成ユーザーは即座に認証キーを追加する必要があります。第二要素なしではログインできないためです。TOTPの場合、ユーザーは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、レルムが強制しなくとも、ユーザーが後から二要素認証を任意で選択することも可能です。

### 14.6.1 利用可能な二次認証方法

スマートフォンやセキュリティキーを紛失した場合にアカウントから永久に締め出される事態を避けるため、複数の第二要素を設定できます。

領域で強制されるTOTPおよびYubiKey OTPに加え、以下の二要素認証方式が利用可能です：

- ユーザー設定のTOTP（時間ベースのワンタイムパスワード）。共有秘密鍵と現在時刻から生成される短いコードで、30秒ごとに更新されます。
- WebAuthn（[Web認証](#)）。認証のための一般的な標準規格です。コンピューターやスマートフォンのハードウェアキーやTPM（信頼できるプラットフォームモジュール）など、様々なセキュリティデバイスによって実装されます。
- 単回使用リカバリキー。印刷して安全な場所に保管するか、電子保管庫にデジタル保存すべきキーのリストです。各キーは一度しか使用できません。他の二次認証手段を全て紛失または破損した場合でも、アカウントから締め出されないようにするのに最適です。

WebAuthnがサポートされる前は、ユーザーがU2Fを設定できました。既存のU2F要素は引き続き使用可能ですが、サーバーでWebAuthnが設定されたら、WebAuthnへの切り替えが推奨されます。

## 14.6.2 領域強制型二要素認証

認証領域を追加または編集する際、TFA ドロップダウンボックスから利用可能な方法のいずれかを選択することで設定できます。領域で TFA が有効化されると、これは必須条件となり、設定済みの TFA を持つユーザーのみがログイン可能となります。

現在利用可能な方法は2つあります：

### 時間ベースのOATH (TOTP)

これは標準の HMAC-SHA1 アルゴリズムを使用し、現在の時刻をユーザーの設定済みキーでハッシュ化します。時間ステップとパスワード長パラメータは設定可能です。

ユーザーは複数のキー（スペース区切り）を設定可能で、キーは Base32 (RFC3548) または 16進表記で指定できます。

Proxmox VE は鍵生成ツール (`oathkeygen`) を提供しており、Base32 表記でランダムな鍵を出力します。この鍵は、`oathtool` コマンドラインツールや、Android の Google Authenticator、FreeOTP、andOTP などの類似アプリケーションで直接使用できます。

### YubiKey OTP

YubiKey による認証には、Yubico API ID、API KEY、検証サーバー URL の設定が必要です。また、ユーザーは YubiKey を所持している必要があります。

YubiKey からキー ID を取得するには、USB 接続後に YubiKey を一度トリガーし、入力されたパスワードの最初の 12 文字をユーザーの `/Key IDs` フィールドにコピーします。

[YubiCloud](#) の使用方法や検証サーバーのホスティングについては、[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

第二要素は、パスワードが何らかの形で漏洩または推測された場合にユーザーを保護することを目的としています。ただし、一部の要素は依然としてブルートフォース攻撃によって破られる可能性があります。このため、第二要素によるログイン試行が何度も失敗すると、ユーザーはロックアウトされます。

TOTP の場合、8 回の失敗でユーザーの TOTP 要素が無効化されます。復旧キーでのログイン時にロックが解除されます。TOTP が唯一の要素だった場合、管理者の介入が必要となり、ユーザーにパスワードの即時変更を強く推奨します。

FIDO2/Webauthn と復旧キーは総当たり攻撃の影響を受けにくいため、制限回数はより多く（100 回）設定されていますが、超過時には 1 時間すべての第二要素がブロックされます。

管理者は、ユーザーリスト (UI) 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できます:

```
pveum user tfa unlock joe@pve
```

## 14.6.4 ユーザー設定のTOTP認証

ユーザーは、ログイン時の第二要素として TOTP または WebAuthn を有効化できます。ユーザーリスト内の TFA ボタンから設定可能です（ただし、レルムが YubiKey OTP を強制している場合は除く）。

ユーザーはいつでもワンタイム リカバリーキーを追加・使用できます。

The screenshot shows the Proxmox VE web interface with the sidebar expanded. The 'Two Factor' section is selected, indicated by a blue background. The main content area displays a table with one row of data:

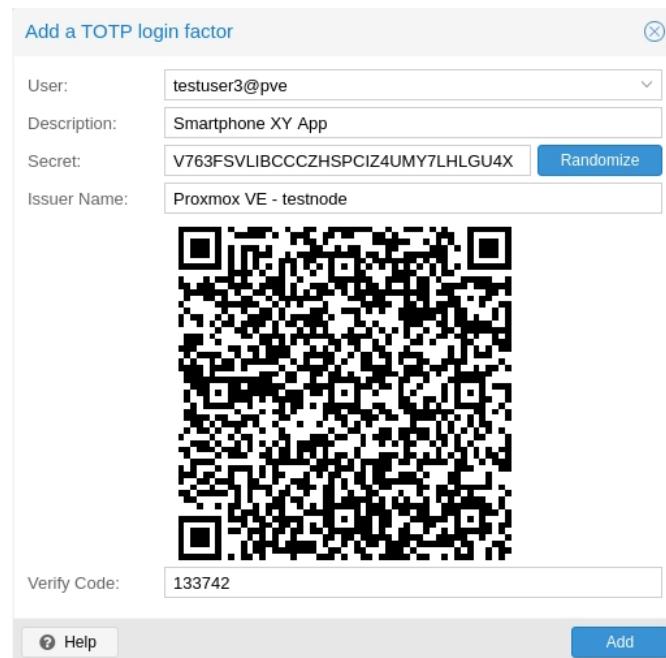
User	Enabled	TFA Type	Created	Description
testuser3@pve	Yes	recovery	2021-11-15 12:20:38	

At the top right of the main content area, there are 'Add', 'Edit', and 'Remove' buttons.

TFAウィンドウを開くと、ユーザーにはTOTP認証を設定するダイアログが表示されます。シークレット欄には鍵が表示され、ランダム化ボタンでランダムに生成できます。オプションの発行者名を追加すると、鍵がどのサービスに属するかをTOTPアプリに通知できます。ほとんどのTOTPアプリは、対応するOTP値と共に発行者名を表示します。ユーザー名もTOTPアプリのQRコードに含まれます。

キー生成後、FreeOTPなどの大半のOTPアプリで使用可能なQRコードが表示されます。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）を確認し、さらに検証コード欄に現在のOTP値を入力して適用ボタンを押すことで、TOTPキーの正しい使用能力を証明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：[FreeOTP](#)）をインストールし、Proxmox VEのWebインターフェースでTOTP要素を追加するだけです。

## 14.6.6 WebAuthn

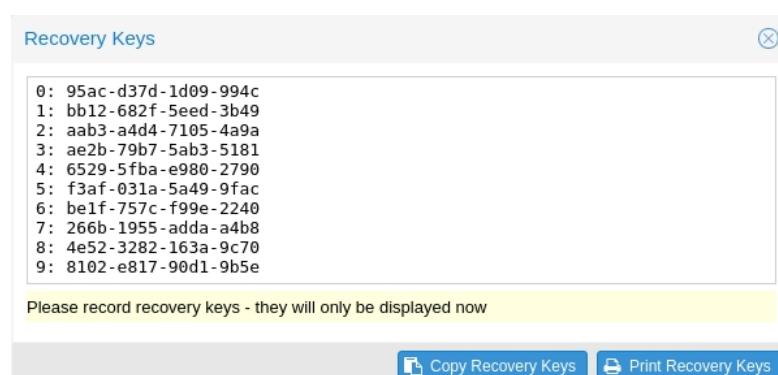
WebAuthnを機能させるには、次の2つが必要です：

- 信頼されたHTTPS証明書（例：[Let's Encryptを使用](#)）。信頼されていない証明書でも動作する可能性がありますが、一部のブラウザでは信頼されていない場合、WebAuthn操作に対して警告を表示したり拒否したりする可能性があります。
- WebAuthn設定の構成（Proxmox VEウェブインターフェースの「データセンター」→「→ オプション」→「→」→「WebAuthn設定」を参照）。ほとんどの設定では自動入力が可能です。

これらの要件を両方満たしたら、データセンターの「二要素認証」パネルでWebAuthn設定を追加できます。

パネルの「データセンター」→「→ 権限」→「→」→「二要素認証」でWebAuthn設定を追加できます。

## 14.6.7 リカバリキー



リカバリキーコードは事前準備が不要です。データセンター→Permissions→Two Factor の「二要素認証」パネルでリカバリキーのセットを作成するだけです。

#### 注記

ユーザーごとに、一度に有効な使い捨て復旧キーは1セットのみです。

### 14.6.8 サーバーサイド WebAuthn 設定



ユーザーが *WebAuthn* 認証を利用するようにするには、有効な SSL 証明書を持つ有効なドメインを使用する必要があります。そうしないと、一部のブラウザが警告を表示したり、認証を完全に拒否したりする可能性があります。

#### 注意

*WebAuthn* 設定を変更すると、既存のすべての*WebAuthn*登録が無効になる可能性があります！

設定は `/etc/pve/datacenter.cfg` で行います。例：

```
webauthn: rp=mypve.example.com,origin=https://mypve.example.com:8006,id=<-->
          mypve.example.com
```

### 14.6.9 サーバーサイド U2F 設定

#### 注記

代わりに*WebAuthn*の使用が推奨されます。

ユーザーが *U2F* 認証を利用するようにするには、有効な SSL 証明書を持つ有効なドメインを使用する必要がある場合があります。そうしないと、一部のブラウザが警告を表示したり、*U2F*の使用を完全に拒否したりする可能性があります。最初に、*AppId*<sup>1</sup> を設定する必要があります。

#### 注意

*AppId*を変更すると、既存のすべての*U2F*登録が無効になります！

これは `/etc/pve/datacenter.cfg` で行います。例：

```
u2f: appId=https://mypve.example.com:8006
```

<sup>1</sup> *AppId* [https://developers.yubico.com/U2F/App\\_ID.html](https://developers.yubico.com/U2F/App_ID.html)

単一ノードの場合、*AppId* はブラウザで使用されるのと同じ形式で、*https://*とポートを含む Web インターフェースのアドレスをそのまま指定できます（上記参照）。なお、*AppId* の照合において、ブラウザによってはより厳格な動作をする場合がありますのでご注意ください。

複数のノードを使用する場合、ほとんどのブラウザと互換性があると思われるため、*appid.json*<sup>2</sup> ファイルを提供する独立した HTTPS サーバーを用意するのが最適です。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、TLD を *AppId* として使用すれば十分かもしれません。ただし、一部のブラウザではこれが受け入れられない可能性があることに注意してください。

#### 注意

不正な*AppId* は通常エラーを生じますが、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、エラーが発生しない事例が確認されています。このため、複数ブラウザでの設定テストを推奨します。後から*AppId*を変更すると既存のU2F登録が無効化されるためです。

### 14.6.10 ユーザーとしてU2Fを有効化する

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（rootとしてログインしている場合を除く）し、[登録ボタン]を押します。サーバーが正しく設定され、ブラウザがサーバー提供の*AppId*を受け入れる場合、ユーザーに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、ボタンライトが約1秒に2回のペースで点滅します）。

Firefoxユーザーは、U2Fトークンを使用する前に、*about:config* で *security.webauth.u2f* を有効にする必要がある場合があります。

## 14.7 権限管理

ユーザーがアクション（VM 構成の一部の一覧表示、変更、削除など）を実行するには、適切な権限が必要です。

Proxmox VE は、ロールとパスに基づく権限管理システムを使用しています。権限テーブルのエントリにより、ユーザー、グループ、またはトークンは、オブジェクトまたはパスにアクセスする際に特定のロールを引き受けることができます。つまり、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）、または（パス、トークン、ロール）の 3 つ組として表現でき、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。

### 14.7.1 ロール

ロールは単に権限のリストです。Proxmox VEには多数の事前定義ロールが付属しており、ほとんどの要件を満たします。

- **管理者:** 完全な特権を持つ
- **NoAccess:** 特権なし（アクセス禁止に使用）
- **PVEAdmin:** ほとんどのタスクを実行可能ですが、システム設定（*Sys.PowerMgmt*、*Sys.Modify Realm.Allocate*）や権限（*Permissions.Modify*）を変更する権限はありません
- **PVEAuditor:** 読み取り専用アクセス権限

<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)

- PVEDatastoreAdmin: バックアップ領域とテンプレートの作成・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を管理
- PVEMappingUser: リソースマッピングの閲覧と使用
- PVEPoolAdmin: プールの割り当て
- PVEPoolUser: プールを表示する
- PVESDNAdmin: SDN構成の管理
- PVESDNUser: ブリッジ/仮想ネットワークへのアクセス
- PVESysAdmin: 監査、システムコンソール、システムログ
- PVETemplateUser: テンプレートの閲覧と複製
- PVEUserAdmin: ユーザーの管理
- PVEVMAAdmin: 仮想マシンの完全な管理
- PVEVMUser: 表示、バックアップ、CD-ROM設定、VMコンソール、VM電源管理

GUIでは事前定義された全ロールを確認できます。GUIまたはコマンドラインから新規

ロールを追加可能です。



GUIでは、データセンターから[権限] > [ロール]タブに移動し、[作成]ボタンをクリックします。そこでロール名を設定し、[権限]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。

コマンドラインからロールを追加するには、*pveum* CLI ツールを使用できます。例：

```
pveum role add VM_Power-only --privs "VM.PowerMgmt VM.Console" pveum role add Sys_Power-only --privs "Sys.PowerMgmt Sys.Console"
```

#### 注

PVE で始まるロールは常に組み込みであり、カスタムロールがこの予約済みプレフィックスを使用することは許可されていません。

## 14.7.2 権限

特権とは特定の操作を実行する権利です。管理を簡素化するため、特権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できます。特権はロールの一部として含まれない限り、ユーザーやバスに直接割り当てるることはできないことに注意してください。

現在、以下の権限をサポートしています：

## ノード/システム関連の権限

- 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閲覧
- 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の使用
- Permissions.Modify: アクセス権限の変更
- プール割り当て: プールの作成/変更/削除
- Pool.Audit: プールを表示する
- 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構成を管理する
- SDN.Audit: SDN構成の表示
- Sys.Audit: ノード状態/構成、Corosyncクラスタ構成、およびHA構成を表示
- Sys.Console: ノードへのコンソールアクセス
- Sys.Incoming: 他のクラスターからのデータストリーム受信を許可（実験的）
- Sys.Modify: ノードの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（起動、停止、リセット、シャットダウン、...）
- Sys.Syslog: システムログの表示
- User.Modify: ユーザーアクセスと詳細の作成/変更/削除

## 仮想マシン関連の権限

- SDN.Use: SDN仮想ネットワークおよびローカルネットワークブリッジへのアクセス
- VM.Allocate: サーバー上の仮想マシンの作成/削除
- VM.Audit: VM設定の閲覧
- VM.Backup: VMのバックアップ/復元
- VM.Clone: VMのクローン作成/コピー
- VM.Config.CDROM: CD-ROMの取り出し/交換
- VM.Config.CPU: CPU設定を変更する
- VM.Config.Cloudinit: Cloud-initパラメータを変更
- VM.Config.Disk: ディスクの追加/変更/削除
- VM.Config.HWType: エミュレートされるハードウェアの種類を変更
- VM.Config.Memory: メモリ設定を変更
- 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
- VM.Console: VMへのコンソールアクセス
- VM.GuestAgent.Audit: QEMUゲストエージェントの情報コマンドを発行
- VM.GuestAgent.FileRead: QEMUゲストエージェント経由でゲストからファイルを読み取る

- VM.GuestAgent.FileSystemMgmt: QEMUゲストエージェント経由でファイルシステムの凍結/解凍/トリミングを実行
- VM.GuestAgent.FileWrite: QEMUゲストエージェント経由でゲスト内のファイルを書き込み
- VM.GuestAgent.Unrestricted: QEMUゲストエージェントへの任意のコマンド発行
- VM.Migrate: クラスタ内の代替サーバーへVMを移行する
- VM.PowerMgmt: 電源管理（起動、停止、リセット、シャットダウンなど）
- VM.Replicate: ゲストトレプリケーションの設定と実行
- VM.Snapshot.Rollback: VMをスナップショットのいずれかにロールバック
- 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/削除およびボリュームの削除
- Datastore.AllocateSpace: データストア上のスペースを割り当てる
- Datastore.AllocateTemplate: テンプレートおよび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 および Sys.Modify は、危険または機密性の高いシステムおよびその構成の側面を変更できるため、慎重に扱う必要があります。



#### 警告

割り当てられたロール（およびその特権）が ACL ツリーに沿ってどのように伝播されるかを理解するために、以下の継承に関するセクションを注意深くお読みください。

### 14.7.3 オブジェクトとパス

アクセス権限は、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にはファイルシステムのようなパスを使用して参照します。これらのパスは自然なツリー構造を形成し、上位の レベル（より短いパス）の権限は、この階層内で任意に下位へ伝播させることができます。

パスはテンプレート化できます。API呼び出しがテンプレート化されたパスに対する権限を必要とする場合、そのパスには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。これらの参照は中括弧で指定されます。一部のパラメータはAPI呼び出しのURIから暗黙的に取得されます。例えば、`/nodes/mynode/status`を呼び出す際の権限パス `/nodes/{node}` は `/nodes/mynode` への権限を要求し、`/access/acl` へのPUTリクエスト内のパス `{path}` はメソッドの `path` パラメータを参照します。

例をいくつか挙げます：

- `/nodes/{node}`: Proxmox VE サーバーマシンへのアクセス
- `/vms`: すべての仮想マシンをカバーします
- `/vms/{vmid}`: 特定の仮想マシンへのアクセス

- /storage/{storeid}: 特定のストレージへのアクセス
- /pool/{poolname}: 特定のプールに含まれるリソースへのアクセス
- /access/groups: グループ管理
- /access/realms/{realmid}: レルムへの管理アクセス

#### 継承

前述の通り、オブジェクトパスはツリー状のファイルシステムを形成し、そのツリーアクセスへの下位のオブジェクトは権限を継承できます（プロパゲートフラグはデフォルトで設定されています）。以下の継承ルールを適用します：

- 個々のユーザーに対する権限は常にグループ権限を上書きします。
- ユーザーがそのグループのメンバーである場合、グループの権限が適用されます。
- より深いレベルの権限は、上位レベルから継承された権限を上書きします。
- NoAccess は、指定されたパス上の他のすべてのロールを無効にします。

さらに、特権分離トークンは、関連付けられたユーザーが持っていない特定のパスに対する権限を一切持つことができません。

### 14.7.4 プール

プールは、一連の仮想マシンやデータストアをグループ化するために使用できます。その後、プール (/pool/{poolid}) に対して権限を設定するだけで、その権限はプールメンバーすべてに継承されます。これは、アクセス制御を簡略化する優れた方法です。

### 14.7.5 必要な権限は？

必要な API 権限は、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/> で確認できます。

権限はリスト形式で指定され、論理演算とアクセスチェック関数のツリーとして解釈されます：

`["and", <サブテスト>...]` と `["or", <サブテスト>...]`  
現在のリスト内の各要素（および）またはいずれかの要素が真でなければならない。

`["perm", <path>, [<privileges>...], <options>...]`  
パスはテンプレート化されたパラメータです（[オブジェクトとパスを参照](#)）。リストされた特権のすべて（または、anyオプションが使用されている場合はいずれか）が、指定されたパスで許可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場合、API呼び出しのスキーマでそれ以外の場合はオプションとしてリストされている場合でも、その指定されたパラメータは必須となります。

`["userid-group", [<privileges>...], <options>...]`  
呼び出し元は /access/groups 上でリストされた特権のいずれかを持つ必要があります。さらに、groups\_param オプションの設定状況に応じて、以下の 2 種類のチェックが行われます：

- `groups_param` が設定されている場合 : API呼び出しに必須の`groups`パラメータが存在し、呼び出し元はリストされた全グループに対して記載された権限のいずれかを保有している必要があります。
- `groups_param` が設定されていません: `userid` パラメータで渡されたユーザーは存在し、かつ呼び出し元がリストされた権限のいずれかを持つグループに所属している必要があります (`/access/groups/<group>`; パス経由)。

#### `["userid-param", "self"]`

API呼び出しの`userid`パラメータに指定された値は、アクションを実行するユーザーを参照する必要があります（通常は`or`と組み合わせて、昇格された権限を持たないユーザーでも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）。

#### `["userid-param", "Realm.AllocateUser"]`

ユーザーは `/access/realm/<realm>` に対する `Realm.AllocateUser` アクセス権を必要とします。`<realm>` は `userid` パラメータで渡されるユーザーのレルムを指します。 ユーザーがレルムに関連付けられるために、そのユーザーが実際に存在している必要はありません。ユーザー ID は `<username>@<realm>`;

#### `["perm-modify", "<path>"]`

パスはテンプレート化されたパラメータです（[オブジェクトとパスを参照](#)）。ユーザーは `Permissions.Modify` 特権、またはパスに応じて以下の特権のいずれかを代替として必要とします：

権限、またはパスに応じて以下の権限のいずれかを代替として必要とします：

- `/storage/...: 'Datastore.Allocate'` が必要
- `/vms/...: 'VM.Allocate'` が必要
- `/pool/...: 'Pool.Allocate'` が必要

パスが空の場合、`/access` に対する `Permissions.Modify` 権限が必要です。

ユーザーが `Permissions.Modify` 権限を持たない場合、指定されたパス上で自身の権限の一部のみを委任できます（例：`PVEVMAdmin` 権限を持つユーザーは `PVEVMUser` を割り当て可能ですが、`PVEAdmin` は不可）。

## 14.8 コマンドラインツール

ほとんどのユーザーはユーザー管理にGUIを使用します。ただし、`pveum`（「Proxmox VE User Manager」の略称）と呼ばれる完全な機能を備えたコマンドラインツールも存在します。すべてのProxmox VEコマンドラインツールはAPIをラップしたものであるため、これらの機能はREST API経由でもアクセス可能です。

簡単な使用例を以下に示します。ヘルプを表示するには、以下を入力してください:

```
pveum
```

または（特定のコマンドに関する詳細ヘルプを表示する場合）

```
pveum help user add
```

新規ユーザーを作成:

```
pveum user add testuser@pve -comment "Just a test"
```

パスワードの設定または変更（すべての領域でサポートされているわけではありません）:

```
pveum passwd testuser@pve
```

ユーザーを無効化:

```
pveum user modify testuser@pve --enable 0
```

新しいグループを作成:

```
pveum group add testgroup
```

新しいロールを作成:

```
pveum role add PVE_Power-only -privs "VM.PowerMgmt VM.Console"
```

## 14.9 実世界の例

### 14.9.1 管理者グループ

管理者は、rootアカウントを使用せずに完全な管理者権限を持つユーザーグループを作成したい場合があります。

これを行うには、まずグループを定義します：

```
pveum group add admin -comment "システム管理者"
```

次にロールを割り当てます:

```
pveum acl modify / -group admin -role Administrator
```

最後に、新しい管理者グループにユーザーを追加できます:

```
pveum user modify testuser@pve -group admin
```

### 14.9.2 監査者

ユーザーまたはグループにPVEAuditorロールを割り当てることで、読み取り専用アクセス権を付与できます。例1: ユーザーjoe@pveにすべての

アクセスを許可

```
pveum acl modify / -user joe@pve -role PVEAuditor
```

例2: ユーザーjoe@pveにすべての仮想マシンを表示させる

```
pveum acl modify /vms -user joe@pve -role PVEAuditor
```

### 14.9.3 ユーザー管理の委任

ユーザー joe@pve にユーザー管理を委任したい場合は、以下のように実行できます:

```
pveum acl modify /access -user joe@pve -role PVEUserAdmin
```

ユーザー joe@pve は、ユーザーを追加・削除したり、パスワードなどの他のユーザー属性を変更したりできるようになりました。これは非常に強力な役割であるため、特定のルームやグループに限定することをお勧めします。次の例では、joe@pve がルーム pve 内のユーザーを、グループ customers のメンバーである場合に限り変更できるように設定しています:

```
pveum acl modify /access/realm/pve -user joe@pve -role PVEUserAdmin  
pveum acl modify /access/groups/customers -user joe@pve -role PVEUserAdmin
```

---

#### 注記

ユーザーは他のユーザーを追加できますが、そのユーザーがグループ「customers」のメンバーであり、かつルーム「pve」内にいる場合に限ります。

---

### 14.9.4 監視用限定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す。つまり、API トークンは、そのトークンを発行したユーザーが許可されていないタスクを実行するために使用することはできません。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、別個の権限を持つAPI トークンを使用する方法を示します。

ユーザー joe@pve に全 VM に対する PVEVMAdmin ロールを付与:

```
pveum acl modify /vms -user joe@pve -role PVEVMAdmin
```

VM情報の閲覧のみを許可する（例：監視目的）新しいAPI トークンを追加します：

```
pveum user token add joe@pve monitoring -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します:

```
pveum user permissions joe@pve  
pveum user token permissions joe@pve monitoring
```

### 14.9.5 リソースプール

企業は通常、いくつかの小規模な部門に構造化されており、各部門にリソースを割り当て、管理タスクを委任することが一般的です。ソフトウェア開発部門用のプールを設定したい場合を考えてみましょう。まず、グループを作成します：

```
pveum group add developers -comment "当社のソフトウェア開発者"
```

次に、そのグループのメンバーとなる新しいユーザーを作成します:

```
pveum user add developer1@pve -group developers -password
```

---

**注記**

"-password" パラメータを指定するとパスワードの入力を求められます

---

次に、開発部門が使用するリソースプールを作成します:

```
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ます:

```
pveum acl modify /pool/dev-pool/ -group developers -role PVEAdmin
```

当社の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りました。

## 第15章

### 高可用性

現代社会はネットワークを介したコンピュータからの情報提供に大きく依存しています。モバイルデバイスは、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、その依存度をさらに高めました。このようなサービスを提供する場合は、ほとんどの時間利用可能であることが非常に重要です。

可用性は、数学的に (A) 特定の期間中にサービスが利用可能な総時間と (B) その期間の長さの比率として定義できます。通常、特定の年における稼働時間の割合としてパーセンテージで表されます。

表15.1: 可用性 - 年間ダウンタイム

可用性 %	年間ダウンタイム
99	3.65日
99.9	8.76時間
99.99	52.56分
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ります。最も洗練された解決策は、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ことです。ソフトウェア自体にエラーを検出しフェイルオーバーを行う仕組みが必要です。読み取り専用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、ソフトウェアを自分で変更できない場合、これは一般的に複雑で、時には不可能です。以下の解決策はソフトウェアを変更せずに機能します：

- 信頼性の高い「サーバー」コンポーネントを使用する

#### 注記

同一機能のコンピュータコンポーネントでも、品質により信頼性数値は異なります。多くのベンダーは高信頼性コンポーネントを「サーバー向け」として販売しており、通常は高価格です。

- 単一障害点の排除（冗長化コンポーネント）

- 無停電電源装置 (UPS) を使用する
  - サーバーに冗長電源を採用する
  - ECC-RAMを使用する
  - 冗長ネットワークハードウェアの使用
  - ローカルストレージにRAIDを使用
  - VMデータ用に分散型冗長ストレージを使用する
- ダウンタイムを削減する
    - 迅速に対応可能な管理者 (24時間365日対応)
    - スペアパーツの可用性 (Proxmox VEクラスタ内他のノード)
    - 自動エラー検出 (`ha-manager`による提供)
    - 自動フェイルオーバー (`ha-manager`による提供)

Proxmox VEのような仮想化環境は、「ハードウェア」依存性を排除するため、高可用性の実現を大幅に容易にします。また、冗長化されたストレージやネットワークデバイスの設定・使用をサポートしているため、1台のホストが障害を起こした場合でも、クラスター内の別のホストでそれらのサービスを単純に起動できます。

さらに優れた点は、Proxmox VEが`ha-manager`と呼ばれるソフトウェアスタックを提供し、これを自動的に実行できることです。エラーを自動検出し、自動フェイルオーバーを実行します。

Proxmox VEの`ha-manager`は「自動化された管理者」のように機能します。まず、管理対象リソース (VM、コンテナなど) を設定します。その後、`ha-manager`は正常な動作を監視し、エラー発生時にはサービスを別のノードへフェイルオーバーします。`ha-manager`は、サービスの起動・停止・再配置・移行といった通常のユーザー要求も処理可能です。

ただし、高可用性には代償が伴います。高品質なコンポーネントは高価であり、冗長化によりコストは少なくとも倍増します。予備部品の追加によりさらにコストが増加します。したがって、メリットを慎重に計算し、追加コストと比較検討する必要があります。

#### ヒント

可用性を99%から99.9%に高めるのは比較的容易です。しかし99.9999%から99.99999%への向上は非常に困難でコストがかかります。`ha-manager`の典型的な障害検出・フェイルオーバー時間は約2分であるため、99.999%以上の可用性は達成できません。

## 15.1 要件

HAを導入する前に、以下の要件を満たす必要があります：

- 少なくとも3つのクラスターノード (信頼性の高いクオーラムを確保するため)
- VMおよびコンテナ用の共有ストレージ
- ハードウェア冗長性 (あらゆる場所で)
- 信頼性の高い「サーバー」コンポーネントの使用

オプションでハードウェアウォッチドッグを使用可能 - 利用不可の場合はLinuxカーネルソフトウェアウォッチドッグ (`softdog`) にフォールバック。

## 15.2 リソース

ha-managerが管理する主要な管理単位をリソースと呼びます。リソース（「サービス」とも呼ばれる）は、サービスID（SID）によって一意に識別されます。SIDはリソースタイプとタイプ固有のIDで構成され、例えばvm:100となります。この例は、ID 100を持つvm（仮想マシン）タイプのリソースです。

現時点では、仮想マシンとコンテナという2つの重要なリソースタイプがあります。ここでの基本的な考え方は、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できるため、rgmanagerで行われていたように他のサービスから1つの大きなサービスを構成する必要がないということです。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ありません。

## 15.3 管理タスク

このセクションでは、一般的な管理タスクの概要を簡潔に説明します。最初のステップは、リソースに対して HA を有効にすることです。これは、リソースを HA リソース構成に追加することで実現します。GUI を使用するか、コマンドラインツールを直接使用できます。例：

```
# ha-manager add vm:100
```

HAスタックは現在、リソースの起動を試行し、稼働状態を維持します。なお、「要求された」リソースの状態は設定可能です。例えば、HAスタックにリソースを停止させたい場合：

```
# ha-manager set vm:100 --state stopped
```

後で再起動させる場合：

```
# ha-manager set vm:100 --state started
```

通常のVMおよびコンテナ管理コマンドも使用できます。これらは自動的にコマンドをHAスタックに転送するため、

```
# qm start 100
```

単に要求された状態を「開始済み」に設定します。qm stop も同様で、要求された状態を「停止済み」に設定します。

### 注記

HAスタックは完全に非同期で動作し、他のクラスターメンバーとの通信が必要です。そのため、このような操作の結果が反映されるまで数秒かかります。

現在の HA リソース構成を確認するには以下を使用します：

```
# ha-manager config vm:100
    状態: 停止中
```

実際の HA マネージャーとリソースの状態は以下で確認できます：

```
# ha-manager status クオーラム 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も開始できます:

```
# ha-manager migrate vm:100 node2
```

これはオンライン移行を使用し、VMの稼働を維持しようとします。オンライン移行では使用中の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する方が速い場合があります。これはrelocateコマンドを使用して実行できます:

```
# ha-manager relocate vm:100 node2
```

最後に、以下のコマンドでリソースをHA構成から削除できます:

```
# ha-manager remove vm:100
```

---

#### 注記

これはリソースの起動や停止を行いません。

---

ただし、すべての HA 関連タスクは GUI で実行できるため、コマンドラインを使用する必要はありません。

## 15.4 動作原理

このセクションでは、Proxmox VE HA マネージャーの内部動作について詳細に説明します。関連するすべてのデーモンとその連携方法について解説します。HAを実現するため、各ノード上で以下の2つのデーモンが実行されます：

---

#### pve-ha-lrm

ローカルリソースマネージャー（LRM）。ローカルノード上で実行中のサービスを制御します。現在のマネージャー状態ファイルからサービスの要求状態を読み取り、対応するコマンドを実行します。

---

#### pve-ha-crm

クラスタリソースマネージャ（CRM）は、クラスタ全体の決定を行う。LRMにコマンドを送信し、結果を処理し、障害発生時にはリソースを他のノードに移動させる。CRMはノードフェンシングも処理する。

---

#### 注記

ロックは分散構成ファイルシステム（pmxcfs）によって提供されます。各LRMが一度だけアクティブになり動作することを保証するために使用されます。LRMは自身のロックを保持している場合に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発生ノードをフェンシング済みとマークできます。これにより、現在未知の障害発生ノードからの干渉なしに、障害発生したHAサービスを安全に回復できます。これらはすべて、現在マネージャーマスターLOCKを保持しているCRMによって監視されます。

---

## 15.4.1 サービス状態

CRMはサービス状態列挙型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-managerコマンドラインツールで照会可能です：

```
# ha-manager status quorum OK
master elsa (active, Mon Nov 21 07:23:29 2016)
lrm elsa (active, Mon Nov 21 07:23:22 2016) service ct:100 (elsa,
stopped)
サービス ct:102 (elsa, 起動済み) サービス vm:501
(elsa, 起動済み)
```

可能な状態の一覧は以下の通りです：

### 停止中

サービスは停止中（LRMにより確認済み）。停止中と判定されたサービスがまだ実行中の場合、再度停止します。

### 停止要求中

サービスを停止すべき状態。CRMはLRMからの確認を待機中。

### stopping

停止リクエストが保留中。ただし、CRMは現時点でリクエストを受信していません。

### started

サービスはアクティブです。LRMは、まだ実行されていない場合、できるだけ早く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はそれを再起動します（[開始失敗ポリシー](#)を参照）。

### 起動中

開始リクエスト待ち。ただし、CRMはサービスが稼働しているというLRMからの確認をまだ受けていない。

### フェンス

サービスノードがクォーレートクラスタパーティション内に存在しないため、ノードフェンシングを待機中（[フェンシング](#)参照）。ノードのフェンシングが成功次第、サービスはリカバリ状態に移行します。

### recovery

サービスの回復待ち。HAマネージャーはサービスが実行可能な新規ノードを検索します。この検索はオンラインかつクォーレート状態のノードリスト、および該当する親和性ルール（存在する場合）に依存します。新規利用可能ノードが見つかり次第、サービスは当該ノードへ移動され、初期状態として停止状態となります。実行設定が有効な場合、新規ノードでサービスが起動されます。

### freeze

サービス状態を変更しない。ノードの再起動時やLRMデーモンの再起動時（[パッケージ更新](#)参照）に使用する状態。

### ignored

サービスがHAによって管理されていないかのように動作します。HA構成から削除せずに、一時的にサービスに対する完全な制御を望む場合に有用です。

**migrate**

サービスを（稼働中に）他のノードへ移行します。

**エラー**

サービスはLRMエラーのため無効化されています。手動での介入が必要です（[エラー回復](#)を参照）。

**キューに追加済み**

サービスが新たに追加され、CRMがまだ認識していません。

**無効**

サービスは停止され、無効化としてマークされています

## 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し、HA クラスタがクオーレート状態になり、クラスタ全体のロックが機能するまで待機します。

以下の3つの状態があります：

**エージェントロック待機**

LRMは排他ロックを待機します。サービスが設定されていない場合のアイドル状態としても使用されます。

**アクティブ**

LRMは排他ロックを保持し、サービスが設定されています。

**エージェントロック喪失**

LRMがロックを失った。これは障害発生によりクオーラムが失われたことを意味する。

LRMがアクティブ状態になると、/etc/pve/ha/manager\_status内のマネージャー状態ファイルを読み取り、自身が管理するサービスに対して実行すべきコマンドを決定します。各コマンドに対してワーカーが起動され、これらのワーカーは並列で実行されます（デフォルトでは最大4つまで）。このデフォルト設定はデータセンター設定キーmax\_workerで変更可能です。完了後、ワーカープロセスは回収され、その結果がCRM用に保存されます。

---

**注記**

デフォルト値である最大4つの同時ワーカーは、特定の環境設定には不適切な場合があります。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と、低速なネットワークや（メモリ消費量の）大きなサービスにおいてネットワーク輻輳を引き起こす可能性があります。また、最悪の場合でも輻輳を最小限に抑えるよう、たとえmax\_worker値を下げるこになんしても確実に実施してください。逆に、特に高性能なハイエンド環境では、この値を増加させることも検討してください。

---

CRMが要求する各コマンドは一意の識別子（UID）で特定されます。ワーカーが処理を完了すると、その結果はLRMステータスファイル（/etc/pve/nodes/<nodename>/lrm\_status）に書き込まれます。CRMはこの情報を収集し、コマンド出力に応じたステートマシンを動作させて対応します。

CRMとLRM間の各サービスに対するアクションは通常常に同期されます。これは、CRMがUIDで一意にマークされた状態を要求すると、LRMがそのアクションを一度実行し、結果を書き戻すことを意味します。

---

結果も同様のUIDで識別可能である。これはLRMが古いコマンドを実行しないようにするために必要である。この動作の唯一の例外は停止 (stop) とエラー (error) コマンドである。これら2つは生成された結果に依存せず、停止状態では常に、エラー状態では一度だけ実行される。

#### 注記

HAスタックは実行するすべてのアクションをログに記録します。これにより、クラスター内で何が、またなぜ発生したかを理解するのに役立ちます。ここでは、LRMとCRMの両方のデーモンが何を行っ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は journalctl -u pve-ha-lrm を、現在のマスターノードでは同じコマンドで pve-ha-crm を実行できます。

### 15.4.3 クラスタリソースマネージャー

クラスタリソースマネージャー (pve-ha-crm) は各ノードで起動し、マネージャーロックを待機します。このロックは一度に1つのノードのみが保持できます。マネージャー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。

以下の3つの状態があります：

#### エージェントロック待機

CRMが排他ロックを待機する状態。サービスが設定されていない場合のアイドル状態としても機能します

#### アクティブ

CRMが排他ロックを保持し、サービスが設定されている状態

#### エージェントロック喪失

CRMがロックを失いました。これは障害が発生し、クオーラムが失われたことを意味します。

その主な役割は、高可用性が設定されたサービスを管理し、要求された状態を常に強制することです。例えば、要求された状態が「起動済み」であるサービスは、まだ実行されていない場合に起動されます。クラッシュした場合、自動的に再起動されます。したがって、CRMはLRMが実行すべきアクションを指示します。

ノードがクラスタクオーラムから離脱すると、その状態は「不明」に変化します。その後、現在のCRMが障害発生ノードのロックを確保できれば、サービスは奪取され別のノードで再起動されます。

クラスタメンバーがクオーラムから外れたと判断した場合、LRMは新たなクオーラムが形成されるまで待機します。クオーラムが再形成されるまで、ノードはウォッチドッグをリセットできません。ノード上でアクティブなサービスが存在する場合、またはLRM/CRMプロセスがスケジューリングされない/強制終了された場合、ウォッチドッグタイムアウト (60秒後) 後に再起動がトリガーされます。

なお、ノードにアクティブなCRMが存在してもLRMがアイドル状態の場合、クオーラム喪失は自己フェンスリセットをトリガーしません。その理由は、CRMがアクセスするすべての状態ファイルと設定は、クオーラム喪失時に読み取り専用となる[クラスタ化された設定ファイルシステム](#)によってバックアップされているためです。これは、CRMが自身のプロセスが長時間スケジュールされることからの保護のみを必要とする意味です。その場合、別のCRMが状況を認識せずに引き継ぐ可能性があり、HA状態の破損を引き起こす恐れがあります。オープンウォッチドッグは、これが発生しないことを保証します。

15分以上サービスが設定されていない場合、CRMは自動的にアイドル状態に戻り、ウォッチドッグを完全に終了します。

## 15.5 HAシミュレータ

The screenshot shows the Proxmox VE management interface. On the left, there's a sidebar with various management options like Search, Summary, Options, Storage, Backup, Replication, Permissions, Users, Groups, Pools, Roles, Authentication, HA, Groups, Fencing, Firewall, and Support. The 'HA' option is currently selected and highlighted in blue.

The main content area has two tables:

- Status** table:
 

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** table:
 

ID	State	Node	Max. Restart	Max. Relo...	Group	Description
vm:501	stopped	demohost1	1	1	prefer_node1	
ct:510	queued	demohost1	1	1	mygroup1	

HAシミュレーターを使用することで、Proxmox VE HAソリューションの全機能をテストおよび学習できます。

デフォルトでは、シミュレーターは6台の仮想マシン（VM）を備えた実世界の3ノードクラスターの動作を監視・テストできます。追加のVMやコンテナの追加・削除も可能です。

実際のクラスターをセットアップまたは構成する必要はなく、HAシミュレーターはすぐに使用できます。aptでインストール:

```
apt install pve-ha-simulator
```

他のProxmox VEパッケージがインストールされていないDebianベースのシステムでも、パッケージを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、必要な依存関係も自動的に解決されます。

リモートマシンでシミュレータを起動するには、現在のシステムへのX11リダイレクトが必要です。Linuxマシンを使用している場合は以下を実行してください:

ださい:

```
ssh root@<PVEのIPアドレス> -Y
```

WindowsではmobaXtermで動作します。

既存のProxmox VEにシミュレータがインストール済みの状態で接続するか、ローカルのDebianベース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、デフォルト設定を書き込む作業ディレクトリを作成する必要があります:

```
mkdir working
```

次に、作成したディレクトリをパラメータとして *pve-ha-simulator* に渡します:

```
pve-ha-simulator working/
```

これでシミュレートされたHAサービスの起動、停止、移行が可能になり、ノード障害時の動作確認もできます。

## 15.6 設定

HAスタックはProxmox VE APIに完全に統合されています。例えば、HAは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から設定可能です。どちらのインターフェースもHA管理を容易に行えます。自動化ツールはAPIを直接利用できます。

すべてのHA設定ファイルは/etc/pve/ha/内に存在するため、クラスタノードへ自動的に配布され、全ノードが同一の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 'Datacenter' menu selected. On the left, there is a collapsed sidebar with various management options like Summary, Options, Storage, Backup, Replication, Permissions, Groups, Pools, Roles, Authentication, HA, Groups, Fencing, Firewall, and Support. The 'HA' section is currently active and expanded. In the main content area, there are two tabs: 'Status' and 'Resources'. The 'Status' tab displays a table with columns 'Type' and 'Status'. It shows a 'quorum' entry with 'OK' status and three 'lrm' entries for 'demohost1' and 'demohost2' with their respective states (active or idle) and timestamps. The 'Resources' tab displays a table with columns 'ID', 'State', 'Node', 'Max. Restart', 'Max. Relo...', 'Group', and 'Description'. It lists two resources: 'vm:501' (stopped, node demohost1, max restart 1, max rel 1, group prefer\_node1) and 'ct:510' (queued, node demohost1, max restart 1, max rel 1, group mygroup1). There are 'Add', 'Edit', and 'Remove' buttons above the resource table.

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次のようにになります:

```
&lt;type&gt;; &lt;name&gt;
    &lt;プロパティ &gt; &lt;値&gt;
    ...

```

リソースタイプとリソース固有の名前がコロンで区切られて記述されます。これらが組み合わさってHAリソースIDを形成し、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ます（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ます:

**comment: &lt;string&gt;**

説明。

**fallback: &lt;boolean&gt; (デフォルト= 1)**

現在のノードよりも優先度の高いノードがオンラインになった場合、ノードアフィニティルールに従い、HAリソースを自動的に優先度の高いノードへ移行します。

**group: &lt;string&gt;**

HAグループの識別子。

**max\_relocate: &lt;integer&gt; (0 - N) (デフォルト= 1)**

サービス起動に失敗した場合の最大再配置試行回数。

**max\_restart: &lt;integer&gt; (0 - N) (デフォルト= 1)**

ノードでのサービス起動失敗後、サービスを再起動する最大試行回数。

**状態: &lt;無効 | 有効 | 無視 | 起動 | 停止&gt; (デフォルト= started)**

要求されたリソースの状態。CRMはこの状態を読み取り、それに応じて動作します。注意：enabled は started の別名に過ぎません。

#### **started**

CRMはリソースの起動を試みます。起動成功後、サービス状態はstartedに設定されます。

ノード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ます。すべてが失敗した場合、サービス状態はエラーに設定されます。

#### **停止**

CRMはリソースを停止状態に保とうとしますが、ノード障害時にはリソースの再配置を試行します。

#### **無効**

CRMはリソースを停止状態に置こうとしますが、ノード障害時にはリソースの再配置を試みません。

。この状態の主な目的はエラー回復であり、リソースをエラー状態から移行させる唯一の手段である。

#### **無視**

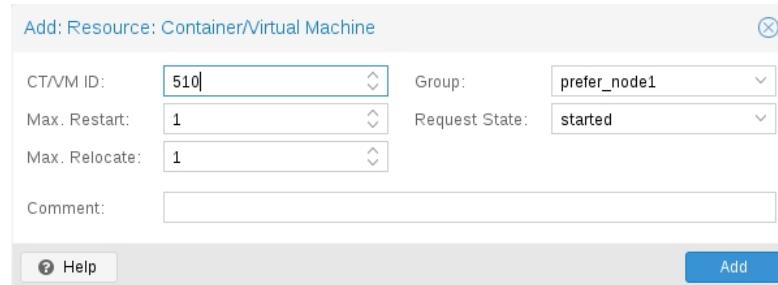
リソースはマネージャーの状態から削除されるため、CRMおよびLRMはリソースに一切触れません。このリソースに影響するすべての{pve} API呼び出しが、HAスタックを直接バイパスして実行されます。リソースがこの状態にある間、CRMコマンドは破棄されます。ノード障害時にもリソースは再配置されません。

以下は、1台のVMと1つのコンテナを用いた実例です。ご覧の通り、これらのファイルの構文は非常にシンプルなので、お好みのエディタで読み書きすることも可能です。

**設定例 (/etc/pve/ha/resources.cfg)**

```
vm: 501
    state started max_relocate
    2

ct: 102
    # 注: すべての設定はデフォルト値を使用
```



上記の設定は ha-manager コマンドラインツールを使用して生成されました:

```
# ha-manager add vm:501 --state started --max_relocate 2 # ha-manager add ct:102
```

## 15.6.2 グループ

### 注記

HAグループは非推奨となり、Proxmox VE 9.0以降ではHAノードアフィニティルールに移行されました。

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

HAグループ設定ファイル `/etc/pve/ha/groups.cfg` は、クラスタノードのグループを定義するために使用されます。リソースは、そのようなグループのメンバー上でのみ実行されるように制限できます。グループ設定は次のようにになります:

```
group: &lt;group&gt;
    nodes &lt;node_list&gt;
        &lt;プロパティ&gt; &lt;値&gt;
    ...

```

**comment: &lt;string&gt;**

説明。

**nodes: &lt;node&gt;[&lt;pri&gt;]{, &lt;node&gt;[&lt;pri&gt;]}\***

クラスターノードメンバーのリスト。各ノードに優先度を付与可能。グループにバインドされたリソースは、利用可能なノードの中で最優先度のノード上で実行される。最優先度クラスにノードが複数存在する場合はグループにバインドされたリソースは、利用可能なノードの中で優先度が最も高いノード上で実行されます。最高優先度クラスに複数のノードが存在する場合、サービスはそれらのノードに分散されます。優先度は相対的な意味しか持ちません。数値が高いほど優先度が高くなります。

**nofailback: &lt;boolean&gt; (デフォルト= 0)**

CRMは優先度の最も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はそのノードへサービスを移行します。nofailbackを有効にするとこの動作を防止します。

**restricted: &lt;boolean&gt; (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、グループで定義されたノードでのみ実行可能です。グループノードメンバーがすべてオフラインの場合、リソースは停止状態になります。制限なしグループのリソースは、グループメンバーがすべてオフラインの場合、任意のクラスターノードで実行可能ですが、グループメンバーがオンラインになるとすぐに移行が戻ります。メンバーが1つのみである制限なしグループを使用して、優先ノードの動作を実装できます。

Create: HA Group

ID:	<input type="text" value="mygroup"/>	restricted:	<input type="checkbox"/>
Comment: <input type="text"/>			
<input type="checkbox"/> Node ↑	Memory usage %	CPU usage	Priority
<input type="checkbox"/> demohost1	65.6 %	5.7% of 2CPUs	▼
<input type="checkbox"/> demohost2	70.6 %	1.2% of 2CPUs	▼

リソースを特定のノードで実行させるという要件は一般的です。通常、リソースは他のノードでも実行可能であるため、メンバーが1つの制限なしグループを定義できます：

```
# ha-manager groupadd prefer_node1 --nodes node1
```

大規模なクラスターでは、より詳細なフェイルオーバー動作を定義することが合理的です。例えば、可能な限りノード1で一連のサービスを実行したい場合があります。ノード1が利用できない場合、ノード2とノード3で均等に分割して実行したいでしょう。それらのノードも障害が発生した場合、サービスはノード4で実行されるべきです。これを実現するには、ノードリストを次のように設定できます:

```
# ha-manager groupadd mygroup1 -nodes "node1:2,node2:1,node3:1,node4"
```

別のユースケースとして、特定ノード（例：node1とnode2）でのみ利用可能なリソースを他のリソースが使用する場合があります。HAマネージャーが他のノードを使用しないようにするために、指定したノードを含む制限付きグループを作成する必要があります：

```
# ha-manager groupadd mygroup2 -nodes "node1,node2" -restricted
```

上記のコマンドにより、以下のグループ設定ファイルが作成されました:

#### 設定例 (/etc/pve/ha/groups.cfg)

```
group: prefer_node1
    nodes node1

group: mygroup1
    nodes node2:1,node4,node1:2,node3:1

group: mygroup2
    nodes node2,node1
    restricted 1
```

nofailbackオプションは、管理タスク中の不要なリソース移動を回避するのに主に有用です。例えば、グループ内で最優先ではないノードへサービスを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、HAマネージャーにこのサービスを即座に戻さないよう指示できます。

別のシナリオとして、フェンスされたサービスが別のノードに復旧した場合が挙げられます。管理者はフェンスされたノードを修復し、障害原因の調査や安定稼働の確認のために再度オンラインにします。nofailbackフラグを設定することで、復旧したサービスがフェンスされたノードに直ちに移動するのを防ぎます。

### 15.6.3 ルール

HAルールは、HA管理リソースに特定の制約を課すために使用され、HAルール設定ファイル /etc/pve/ha/rules.cfg で定義されます。

```
&lt;type&gt;: &lt;rule&gt;
    resources &lt;resources_list&gt;
        &lt;property&gt; &lt;value&gt;
        ...
comment: &lt;string&gt;
    HAルールの説明。
```

**無効化: &lt; ブール値&gt;**

HAルールが無効化されているかどうか。

```
resources: &lt;type&gt;:&lt;name&gt;{, &lt;type&gt;:&lt;name&gt;}*
```

HAリソースIDのリスト。リソースタイプとリソース固有名のリストで構成され、

(例: vm:100,ct:101)。

表 15.2: 利用可能な HA ルールタイプ

HAルールタイプ	説明
node-affinity	1つ以上のHAリソースから1つ以上のノードへのアフィニティを配置します。
リソースアフィニティ	複数のHAリソース間のアフィニティを設定します。アフィニティポジティブはHAリソースを別々のノードに配置することを指定し、アフィニティセパレートはHAリソースを同じノードに配置することを指定します。

**ノードアフィニティルール**

デフォルトでは、HAリソースはクラスタ内の任意のノードで実行可能ですが、特定のノードでの実行を要求するケースが一般的です。これを実現するには、HAノードアフィニティルールを定義し、HAリソースvm:100がnode1:100を優先するように設定します。

```
# ha-manager rules add node-affinity ha-rule-vm100 --resources vm:100 --<-->
nodes node1
```

デフォルトでは、ノードアフィニティルールは厳密ではありません。つまり、指定されたノードが利用できない場合、HAリソースは他のノードに移動することも可能です。一方、HAリソースを指定されたノードに限定する必要がある場合は、ノードアフィニティルールを厳密に設定する必要があります。

前の例では、リソースvm:100をnode1のみに制限するようノードアフィニティルールを変更できます:

```
# ha-manager rules set node-affinity ha-rule-vm100 --strict 1
```

大規模なクラスターや特定のユースケースでは、より詳細なフェイルオーバー動作を定義することが合理的です。例えば、リソースvm:200とct:300はnode1で実行されるべきです。node1が利用不可になった場合、リソースはnode2とnode3に分散されるべきです。node2とnode3も利用不可の場合、リソースはnode4で実行されるべきです。

この動作をノードアフィニティルールで実装するには、ノードを優先順位とペアにしてノードの優先度を順序付けます。2つ以上のノードが同じ優先度を持つ場合、リソースはそれらのいずれでも実行可能です。上記の例では、node1が最優先、node2とnode3が同優先度、最後にnode4が最低優先度となります（省略時はデフォルトで0となります）。

```
# ha-manager rules add node-affinity priority-cascade \--resources vm:200,ct:300--nodes "node1:2,node
--resources vm:200,ct:300 --nodes "node1:2,node2:1,node3:1,node4"
```

上記のコマンドは、ルール設定ファイルに以下のルールを作成します:

### ノードアフィニティルール設定例 (/etc/pve/ha/rules.cfg)

```
node-affinity: ha-rule-vm100 resources
    vm:100 nodes node1
    strict 1

node-affinity: priority-cascade resources
    vm:200,ct:300
    nodes node1:2,node2:1,node3:1,node4
```

### ノードアフィニティルールプロパティ

#### **nodes: <node>[:<pri>] {, <node>[:<pri>]}\***

クラスタノードメンバーのリスト。各ノードに優先度を指定可能。グループにバインドされたリソースはグループにバインドされたリソースは、利用可能なノードの中で優先度が最も高いノード上で実行されます。最高優先度クラスに複数のノードが存在する場合、サービスはそれらのノードに分散されます。優先度は相対的な意味しか持ちません。数値が高いほど優先度が高くなります。

#### **resources: <type>:<name>{, <type>:<name>}\***

HAリソースIDのリスト。リソースタイプのリストにリソース固有の名前をコロンで区切って続きます  
(例: vm:100,ct:101)。

#### **strict: <boolean> (デフォルト = 0)**

ノードアフィニティルールが厳密か非厳密かを記述します。

非厳密なノードアフィニティルールでは、リソースは定義されたノード上で実行されることを優先します。定義されたノードがすべて利用不可の場合、リソースは他の任意のノード上で実行される可能性があります。

厳格なノードアフィニティルールでは、リソースは定義されたノードに制限されます。定義されたノードが利用できない場合、リソースは停止されます。

### リソースアフィニティルール

もう1つの一般的な要件は、2つ以上のHAリソースが同じノード上で実行されるか、別々のノードに分散されることです。これらは一般的に「アフィニティ/アンチアフィニティ制約」とも呼ばれます。

例えば、HAリソースvm:100とvm:200（例：Webサーバーとデータベースサーバー）間で大量の通信トラフィックが発生する場合を考えます。これらのHAリソースが別々のノードにあると、潜在的に高いレイテンシと不要なネットワーク負荷が生じる可能性があります。アフィニティポジティブを伴うリソースアフィニティルールは、HAリソースを同じノードに保持する制約を実装します：

```
# ha-manager rules add resource-affinity keep-together \--affinity positive --resources vm:100,vm:200
  --affinity positive --resources vm:100,vm:200
```

### 注記

共通のHAリソースを持つ複数の正の資源親和性ルールが存在する場合、これらは単一の正の資源親和性ルールとして扱われます。たとえば、HAリソース vm:100 と vm:101、および HAリソース vm:101 と vm:102 がそれぞれ肯定的なリソースアフィニティルールに含まれている場合、vm:100、vm:101、vm:102 が単一の肯定的なリソースアフィニティルールに含まれている場合と同じになります。

### 注

肯定的リソースアフィニティルールのHAリソースが現在異なるノードで実行されている場合、CRSはそれらのHAリソースを、既に大半が実行されているノードへ移動させます。HAリソース数が同数の場合、アルファベット順で名前が先に表示されるノードが選択されます。

ただし、例えばシャーディングされたデータベースインスタンスのように、計算負荷が高く分散処理を必要とするプログラムがHAリソース vm:200 と ct:300 で実行されている場合、同一ノード上で実行するとハードウェアリソースに負荷がかかり、これらのHAリソースの動作が遅くなる可能性があります。アフィニティネガティブを指定したリソースアフィニティルールは、HAリソースを別々のノードに分散させる制約を実装します：

```
# ha-manager rules add resource-affinity keep-separate \
    --affinity negative --resources vm:200,ct:300
```

ノードアフィニティルールとは異なり、リソースアフィニティルールはデフォルトで厳格です。つまり、リソースアフィニティルールによる制約がHAリソースに対して満たせない場合、HAマネージャーはフェイルオーバー時にはHAリソースをリカバリ状態に、その他の状況ではエラー状態に置きます。

上記のコマンドにより、ルール設定ファイルに以下のルールが作成されました:

#### リソースアフィニティルール設定例 (/etc/pve/ha/rules.cfg)

```
resource-affinity: keep-together resources
    vm:100,vm:200 affinity positive

resource-affinity: keep-separate resources
    vm:200,ct:300 affinity negative
```

#### 正と負のリソース親和性ルールの相互作用

正の資源親和性ルールにHAリソースが含まれ、かつそれらのリソースが負の資源親和性ルールにも含まれる場合、正の資源親和性ルール内の他のすべてのHAリソースは、これらの負の資源親和性ルール内のHAリソースとも負の親和性を持つ。

例えば、HAリソース vm:100、vm:101、vm:102 が正のリソースアフィニティルールに含まれ、vm:100 が HAリソース ct:200 との負のリソースアフィニティルールに含まれる場合、vm:101 と vm:102 もそれぞれ ct:200 との負のリソースアフィニティを持つことになります。

複数のHAリソースが肯定的および否定的リソースアフィニティルールの両方に含まれる場合、それらは競合を引き起こすため無効化されることに注意してください。複数のHAリソースを同時に同一ノードに保持しつつ異なるノードに分離することはできません。これらのケースの詳細については、以下の[ルール競合とエラー](#)に関するセクションを参照してください。

### ノード親和性ルールと正のリソース親和性ルールの相互作用

ノード親和性ルールにHAリソースが含まれ、かつそのリソースが正の親和性ルールにも属する場合、正の親和性ルール内の他のすべてのHAリソースはノード親和性ルールも継承します。

例えば、 HAリソースvm:100、vm:101、vm:102が正のリソースアフィニティルールに含まれ、vm:102がノードアフィニティルール（vm:102をノード3にのみ配置する制限）に含まれる場合、vm:100とvm:101もノード3にのみ配置されるようになります。

注意：2つ以上のHAリソースが肯定的リソースアフィニティルールと異なるノードアフィニティルールに同時に含まれる場合、これらのルールはエラーを引き起こし無効化されます。これは現在サポートされていないためです。これらのケースの詳細については、以下の[ルール競合とエラー](#)に関するセクションを参照してください。

### リソースアフィニティルールのプロパティ

#### **affinity: <negative| positive>;**

HAリソースを同一ノードに保持する（ポジティブ）か、別々のノードに保持する（ネガティブ）かを記述します。

#### **resources: <type>:<name>{,<type>:<name>}\***

HAリソースIDのリスト。リソースタイプとリソース固有名のリストで構成されます。

コロンで区切られる（例：vm:100,ct:101）。

## 15.6.4 ルール競合とエラー

HAルールはHAリソースに対してかなり複雑な制約を課す可能性があります。新規または変更されたHAルールがHAスタックのCRSスケジューラに不確実性をもたらさないよう、HAルールは適用前に実現可能性テストを受けます。テストに失敗したルールは、競合やエラーが解決されるまで無効化されます。

現在、HAルールは次の実現可能性テストでチェックされます：

- HAリソースは単一のHAノードアフィニティルールにのみ属することができます。
- HAリソースアフィニティルールには、少なくとも2つのHAリソースが含まれている必要があります。
- 否定的な HA リソースアフィニティルールは、クラスタ内のノード数よりも多くの HA リソースを指定できません。そうしないと、HA リソースを分離するのに十分なノードが確保されません。
- 肯定的なHAリソースアフィニティルールは、否定的なHAリソースアフィニティルールと同じ2つ以上のHAリソースを指定できません。つまり、2つ以上のHAリソースを同時に「一緒に保持」かつ「分離」することはできません。
- HAリソースは、HAノードアフィニティルールが単一の優先度クラスを持つ場合に限り、HAノードアフィニティルールとHAリソースアフィニティルールの両方の一部となることができます。
- 肯定的なHAリソースアフィニティルールのHAリソースは、最大でも単一のHAノードアフィニティルールにのみ属することができます。
- 否定的な HA リソースアフィニティルールの HA リソースは、そのノードアフィニティルールによって HA リソースよりも少ないノードに制限することはできません。そうしないと、HA リソースを分離するのに十分なノードがなくなります。

## 15.7 フェンシング

ノード障害時、フェンシングは問題のあるノードが確実にオフラインになることを保証します。これは、リソースが別のノードで回復された際に二重に実行されないようにするために必要です。これは非常に重要なタスクであり、これがなければ別のノードでリソースを回復することは不可能です。

ノードがフェンシングされなかった場合、共有リソースへのアクセス権を保持したまま未知の状態に陥ります。これは非常に危険です！例えば、ストレージ以外の全ネットワークが切断された状況を想像してください。パブリックネットワークからは到達不能でも、VMは動作を継続し共有ストレージへの書き込みを続けます。

このVMを別のノードで単純に起動すると、両ノードから書き込みが行われるため危険な競合状態が発生します。このような状態ではVMデータが破壊され、VM全体が使用不能になる可能性があります。ストレージが複数マウントを保護している場合、復旧も失敗する可能性があります。

### 15.7.1 Proxmox VEのフェンス処理

ノードをフェンシングする方法は複数存在します。例えば、ノードへの電源を遮断するフェンシングデバイスや、通信を完全に遮断する手法などです。これらは通常非常に高価であり、システムに追加の重要コンポーネントを導入することになります。なぜなら、これらのデバイスが故障した場合、いかなるサービスも復旧できなくなるからです。

そこで我々は、追加の外部ハードウェアを必要としない、よりシンプルなフェンシング手法を統合したいと考えました。これはウォッチドッグタイマーを使用して実現できます。

#### 可能なフェンシング手法

- 外部電源スイッチ
- スイッチ上でネットワークトラフィックを完全に無効化しノードを隔離
- ウォッチドッグタイマーを用いた自己フェンシング

ウォッチドッグタイマーは、マイクロコントローラーの登場以来、重要かつ信頼性の高いシステムで広く使用されてきた。これらは通常、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、シンプルで独立した集積回路である。

通常動作中、ha-managerはウォッチドッグタイマーが満了しないよう定期的にリセットする。ハードウェア故障やプログラムエラーによりウォッチドッグのリセットに失敗した場合、タイマーが満了するとサーバー全体のリセット（再起動）がトリガーされる。

最近のサーバーマザーボードには、こうした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、これらは設定が必要である。ウォッチドッグが利用できない、または設定されていない場合、Linuxカーネルのソフトドッグにフォールバックする。信頼性は保たれるものの、サーバーのハードウェアに依存するため、ハードウェアウォッチドッグよりも信頼性は低くなる。

### 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全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ます。正しく初期化されていない場合、これらは装填された銃のような危険性を有します。 ハードウェアウォッチドッグを有効化するには、

例:

```
# ウォッチドッグモジュールを選択 (デフォルトはsoftdog) WATCHDOG_MODULE=ITCO_wdt
```

この設定は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をロードします。

### 15.7.3 フェンシングされたサービスの復旧

ノードが障害を起こし、そのフェンシングが成功した後、CRMは障害を起こしたノードから、まだオンラインであるノードへサービスを移動しようと試みます。

回復対象ノードの選択には、現在アクティブなノードのリスト、使用されているスケジューラに応じた各ノードの負荷、および該当サービスが属するアフィニティルール（存在する場合）が影響します。

まず、CRMはサービスが利用可能なノードの集合を構築します。サービスがノードアフィニティルールの一部である場合、この集合はノードアフィニティルール内の最優先ノードに絞り込まれます。サービスがリソースアフィニティルールの一部である場合、その制約を満たすようさらに集合が絞り込まれます。この制約は、サービスを他のサービスと同じノードに保持するか、他のサービスとは異なるノードに保持するかのいずれかです。最後に、CRMは使用中のスケジューラに基づき負荷が最も低いノードを選択し、過負荷ノードの可能性を最小限に抑えます。

**注意**

ノード障害発生時、CRMはサービスを残存ノードに分散します。これにより対象ノードのサービス数が増加し、特に小規模クラスタでは高負荷状態を引き起こす可能性があります。このような最悪のシナリオに対処できるクラスタ設計を行ってください。

## 15.8 起動失敗ポリシー

ノード上でサービスが1回以上起動に失敗した場合、起動失敗ポリシーが発動します。これにより、同一ノードでの再起動頻度や、別のノードで起動を試みるためのサービス再配置頻度を設定できます。このポリシーの目的は、特定のノードにおける共有リソースの一時的な利用不可を回避することです。例えば、ネットワーク問題などによりクオーラートノードで共有ストレージが利用不可になっても、他のノードでは利用可能な場合、リロケートポリシーによりサービスは起動を継続できます。

各リソースごとに個別に設定可能な、2つのサービス起動回復ポリシー設定があります。

**max\_restart**

現在のノード上で失敗したサービスを再起動する試行の最大回数。デフォルトは1回に設定されています。

**max\_relocate**

サービスを別のノードに移転する試行の最大回数。移転は、実際のノードで max\_restart の値を超えた後にのみ発生します。デフォルトは 1 に設定されています。

**注記**

再配置カウント状態は、サービスが少なくとも1回正常に起動した場合にのみゼロにリセットされます。つまり、エラーを修正せずにサービスを再起動すると、再起動ポリシーのみが繰り返されることになります。

## 15.9 エラー回復

すべての試行後もサービス状態が回復できない場合、エラー状態に移行します。この状態では、サービスはHAスタックによって操作されなくなります。唯一の解決策はサービスを無効化することです：

```
# ha-manager set vm:100 --state disabled
```

これはウェブインターフェースでも実行できます。

エラー状態から回復するには、以下の手順を実行してください：

- リソースを安全かつ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はそのプロセスを強制終了する）
- エラーフラグを解除するためリソースを無効化する
- この障害を引き起こしたエラーを修正する
- すべてのエラーを修正した後、サービスの再起動を要求できます

## 15.10 パッケージの更新

ha-managerの更新は、複数の理由から一度にまとめて行うのではなく、ノードを1つずつ順番に更新する必要があります。第一に、ソフトウェアは徹底的にテストしていますが、特定の環境設定に影響するバグが完全に排除できるわけではありません。ノードを1つずつ更新し、更新完了後に各ノードの機能を確認することで、万が一の問題発生時の復旧が容易になります。一方、一括更新はクラスタの破損を招く可能性があり、一般的に推奨される手法ではありません。

また、Proxmox VE HAスタックは、クラスターとローカルリソースマネージャー（LRM）間の操作実行に要求確認プロトコルを使用します。再起動時には、LRMがCRMに対し全サービスを凍結するよう要求します。これにより、LRM再起動の短時間中、クラスターによるサービスへの干渉が防止されます。その後、再起動中にLRMはウォッチドッグを安全に終了できます。このような再起動は通常パッケージ更新時に発生し、前述の通りLRMからのリクエストを承認するにはアクティブなマスターCRMが必要です。これが満たされない場合、更新プロセスが過度に長引く可能性があり、最悪の場合ウォッチドッグによるリセットがトリガーされる恐れがあります。

## 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場合があります。これはHAスタックが稼働中である場合にも適用されます。

HAスタックは主に2種類のメンテナンスをサポートします：

- 一般的なシャットダウンや再起動については、動作を構成できます（[シャットダウンポリシー](#)を参照）。
- シャットダウンや再起動を必要としないメンテナンス、または1回の再起動後に自動的に電源を切断すべきでないメンテナンスについては、手動メンテナンスマードを有効にできます。

### 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操作不可としてマークでき、HAが管理するすべてのサービスを他のノードへ移行させます。

移行先のノードは、現在利用可能な他のノードから選択され、HAルール設定と設定済みのクラスターリソーススケジューラ（CRS）モードによって決定されます。各移行時、元のノードはHAマネージャーの状態に記録されるため、メンテナンスマードを無効化しノードがオンラインに戻った際に、サービスを自動的に元のノードへ戻すことが可能です。

現在、メンテナンスマードは ha-manager CLI ツールを使用して有効化または無効化できます。

#### ノードのメンテナンスマードを有効にする

```
# ha-manager crm-command node-maintenance enable NODENAME
```

これによりCRMコマンドがキューに追加され、マネージャーがこのコマンドを処理すると、メンテナンスマードへの移行要求がマネージャーステータスに記録されます。これにより、メンテナンスマードの開始/終了対象ノードだけでなく、任意のノードからコマンドを送信できます。

対象ノードのLRMがこのコマンドを処理すると、自身を「利用不可」状態に設定しますが、移行コマンドは引き続き処理します。これにより、すべてのアクティブサービスが移動され、稼働中のワーカーがすべて終了するまで、LRMの自己フェンシング監視機能は有効なまま維持されます。

なお、LRMが要求された状態を認識した時点（全サービス移行完了時ではなく）LRMステータスは「メンテナンスマード」と表示されます。このユーザー体験は将来的に改善予定です。現時点では、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するか、または「pve-ha-lrm[PID]: watchdog closed (disabled)」のようなログ行を監視することで、ノードがメンテナンスマードへの移行を完了したタイミングを把握できます。

---

#### 注記

手動メンテナンスマードはノード再起動時に自動解除されません。ha-manager CLIで手動無効化するか、manager-statusを手動でクリアした場合にのみ解除されます。

---

#### ノードのメンテナンスマードを無効化

```
# ha-manager crm-command node-maintenance disable NODENAME
```

手動メンテナンスマードを無効化する手順は有効化時と同様です。上記の ha-manager CLI コマンドを実行すると CRM コマンドがキューに追加され、処理完了後に該当する LRM ノードが再び利用可能状態としてマークされます。

メンテナンスマードを無効化すると、モード有効時に当該ノード上にあった全サービスが元に戻されます。

## 15.11.2 シャットダウンポリシー

以下に、ノードシャットダウンに関する各種HAポリシーの説明を示します。現在、下位互換性のためConditionalがデフォルトです。一部のユーザーはMigrateの方が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（データセンター→オプション→HA設定）で設定するか、datacenter.cfg ファイルに直接記述できます：

```
ha: shutdown_policy=<value>;
```

## 移行

ローカルリソースマネージャー (LRM) がシャットダウン要求を受信し、このポリシーが有効になっている場合、LRMは現在のHAマネージャーに対して自身を利用不可としてマークします。これにより、現在このノード上に存在するすべてのHAサービスの移行がトリガーされます。LRMは、実行中のサービスがすべて移行されるまでシャットダウンプロセスを遅延させようと試みます。ただし、これは実行中のサービスが別のノードに移行可能であることを前提としています。言い換えれば、サービスはハードウェアパススルーの使用などによってローカルに拘束されてはなりません。例えば、厳格なノードアフィニティルールは、選択されたノード群の外ではサービスが実行できないことをHAマネージャーに伝えます。これらのノードがすべて利用不可の場合、シャットダウンは手動介入があるまで停止します。シャットダウンしたノードが再びオンラインに戻ると、その間に手動で移行されていない限り、以前に移動されたサービスは元の位置に戻されます。

---

### 注記

シャットダウン時の移行プロセス中もウォッチドッグは有効です。ノードがクローラムを失った場合、フェンス処理が行われサービスが復旧されます。

---

メンテナンス中のノードで（以前に停止した）サービスを起動する場合、そのサービスを別の利用可能なノードに移動して起動できるようにするために、当該ノードをフェンスする必要があります。

## フェイルオーバー

このモードでは、すべてのサービスが停止されますが、現在のノードがすぐにオンラインに戻らない場合でも、それらのサービスが確実に復旧されます。クラスタ規模のメンテナンスを行う際に有用です。一度に多くのノードの電源をオフにすると、VM のライブマイグレーションが不可能になる場合がありますが、それでも HA サービスをできるだけ早く復旧して再起動させたい場合に適しています。

## 凍結

このモードでは、すべてのサービスが停止および凍結され、現在のノードが再びオンラインになるまで復旧されません。

## 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のリクエストを自動的に検出し、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、ノードが一定期間停止状態を維持する計画がある場合に通常実行されます。この場合、LRMは管理対象サービスをすべて停止します。これにより、他のノードがその後これらのサービスを継承します。

---

### 注記

最近のハードウェアは大量のメモリ（RAM）を搭載しています。そのため、すべてのリソースを停止し、再起動することで、そのRAM全体のオンライン移行を回避します。オンライン移行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それを呼び出す必要があります。

---

## 再起動

ノードの再起動/rebootコマンドで開始されます。これは通常、新しいカーネルのインストール後に実行されます。「shutdown」とは異なり、ノードは直ちに再起動することに注意してください。

LRMはCRMに再起動を要求し、CRMが全リソースを凍結状態にするまで待機します（[パッケージ更新時](#)と同じ仕組み）。これにより、それらのリソースが他のノードに移動されるのを防ぎます。代わりに、CRMは再起動後、同じノード上でリソースを起動します。

## 手動リソース移動

最後に重要な点として、ノードのシャットダウンや再起動前に、リソースを手動で他のノードに移動することも可能です。この方法の利点は、完全な制御が可能であり、オンライン移行を使用するかどうかを判断できることです。

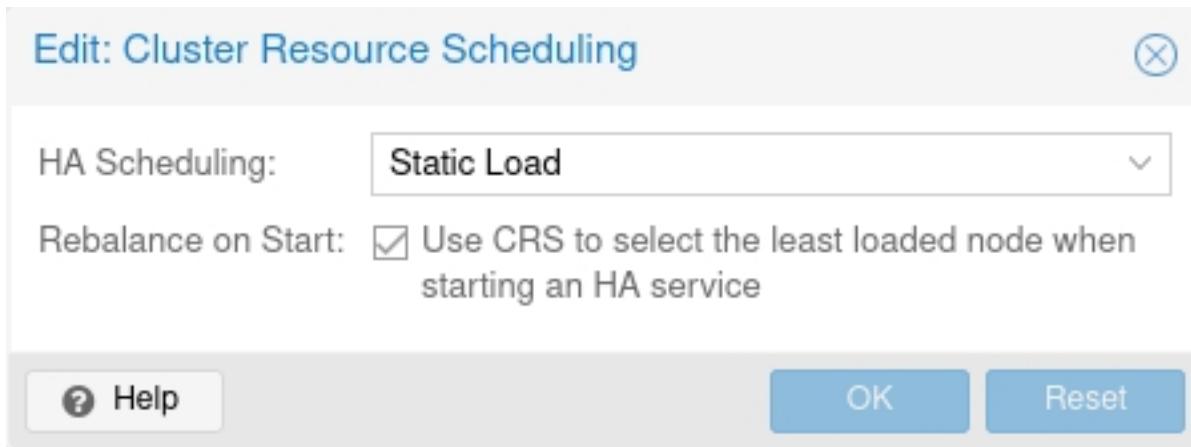
### 注意

pve-ha-crm、pve-ha-lrm、watchdog-muxなどのサービスを強制終了しないでください。これらはウォッチドッグを管理・使用しているため、ノードの即時再起動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がサービスの回復およびシャットダウンポリシーによってトリガーされる移行のためにノードを選択する方法を制御します。デフォルトモードはbasicです。Web UI(データセンター→オプション)またはdatacenter.cfgで直接変更できます:

```
crs: ha=static
```



変更は次のマネージャーラウンド(数秒後)から有効になります。

各サービスのリカバリまたは移行が必要となる場合、スケジューラはHAルール(存在する場合)に基づき、サービスが利用可能なノードの中から最適なノードを反復的に選択します。

### 注記

将来的には(静的および動的な)負荷分散モードを追加する計画があります。

### 15.12.1 基本スケジューラ

各ノード上のアクティブなHAサービスの数が、リカバリノードの選択に使用されます。非HA管理サービスは現在カウントされません。

### 15.12.2 静的負荷スケジューラ

**重要**

静的モードは現在も技術レビュー段階です。

各ノード上のHAサービスからの静的使用状況情報が、リカバリノードの選択に使用されます。非HA管理サービスは現在考慮されません。

この選択では、各ノードを順番に、サービスが既にそのノード上で実行されているかのように扱い、関連するゲスト構成のCPU使用率とメモリ使用率を使用します。次に、そのような代替案ごとに、すべてのノードのCPUとメモリ使用量が考慮されます。メモリは真に制限されたリソースであるため、はるかに重み付けされます。CPUとメモリの両方について、ノード間の最高使用量（より重み付けされる、理想的にはどのノードも過剰コミットされるべきではないため）と、すべてのノードの平均使用量（より高いコミット率のノードが既に存在する場合でも区別できるようにするため）が考慮されます。

**重要**

サービス数が増えるほど組み合わせの可能性も増えるため、数千ものHA管理サービスがある場合は、現時点では使用を推奨しません。

### 15.12.3 CRSスケジューリングのポイント

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。そうすると頻繁なノード移動が発生し、ワークロードによってはクラスターに過剰な負荷がかかる可能性があるためです。このため、Proxmox VE HAマネージャーはサービスを現在のノードに保持することを優先します。

CRSは現在、以下のスケジューリングポイントで使用されます：

- サービス回復（常時有効）。アクティブなHAサービスを持つノードが障害を起こした場合、そのノードの全サービスを他のノードに回復させる必要があります。この回復処理は、残存ノード間で負荷分散するためCRSアルゴリズムが使用されます。
- HAグループ構成変更（常時有効）。ノードがグループから削除された場合、またはそ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合致する、そのグループ内のHAサービスに対する新たなターゲットノードを特定します。
- HAルール構成変更（常時有効）。ルールがHAリソースに異なる制約を課す場合、HAスタックは新規ルールの種類に応じて、これらのルールに影響を受けるHAリソースの新たなターゲットノードをCRSアルゴリズムで探索します：

- ノードアフィニティルール：ノードアフィニティルールが作成されるか、既存のルールにHAリソース/ノードが追加されると、HAスタックはCRSアルゴリズムを使用して、これらのHAリソースがノードおよび優先度制約に従って割り当てられることを保証します。
  - 肯定的リソースアフィニティルール：肯定的リソースアフィニティルールが作成されるか、既存の肯定的ルールにHAリソースが追加されると、HAスタックはCRSアルゴリズムを使用してこれらのHAリソースが共通ノードに移動されることを保証します。
  - ネガティブリソースアフィニティルール：ネガティブリソースアフィニティルールが作成されるか、既存のネガティブリソースアフィニティルールにHAリソースが追加されると、HAスタックはCRSアルゴリズムを使用して、これらのHAリソースが別々のノードに移動されることを保証します。
- HAサービス **停止** 起動移行（オプトイン）。停止中のサービスの起動要求は、CRSアルゴリズムに基づく最適なノードの確認に適した機会です。特にディスクボリュームが共有ストレージ上に存在する場合は、起動中のサービスを移動するよりも停止中のサービスを移動する方がコスト効率が良いからです。この機能は、データセンター設定で **ha-rebalance-on-start** CRS オプションを設定することで有効化できます。このオプションは Web UI の「データセンター」→「→ オプション」→「クラスタリソーススケジューリング」からも変更可能です。

## 第16章

# バックアップと復元

バックアップは、あらゆる賢明な IT 導入に必須の要件であり、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利用した、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の間で、モードオプションを使用して微調整を行うことができます。

Proxmox VE のバックアップは常に完全バックアップであり、VM/CT 構成とすべてのデータが含まれます。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。ストレージの追加方法については、[ストレージのドキュメント](#)を参照してください。ストレージは、バックアップが重複排除されたチャンクとメタデータとして保存されるProxmox Backup Serverストレージ、またはバックアップが通常のファイルとして保存されるファイルレベルストレージのいずれかです。高度な機能を備えているため、専用ホスト上でProxmox Backup Serverを使用することを推奨します。NFSサーバーの使用も有効な代替手段です。いずれの場合も、オフサイトアーカイブ用に後でテープドライブにバックアップを保存したい場合があります。

### スケジュールされたバックアップ

バックアップジョブは、特定の曜日と時間に自動的に実行されるようスケジュール設定できます。対象となるノードやゲストシステムを選択可能です。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

ゲストの種類に応じて、整合性（オプションモード）を確保する方法はいくつかあります。

VMのバックアップモード：

### 停止モード

このモードは、VM 操作の短いダウンタイムを代償として、バックアップの最高の一貫性を提供します。VM を順序立てシャットダウンし、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。バックアップ開始後、VM が以前に実行されていた場合は完全な動作モードに戻ります。一貫性はライブバックアップ機能を使用して保証されます。

## サスPENDモード

互換性のために提供されるモードで、スナップショットモードを呼び出す前にVMをサスPENDします。VMのサスPENDはより長いダウンタイムを発生させ、データの整合性を必ずしも向上させないため、代わりにスナップショットモードの使用が推奨されます。

## 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代償として、最も低い稼働停止時間を提供します。Proxmox VEのライブバックアップを実行することで機能し、VMが稼働中にデータブロックをコピーします。ゲストエージェントが有効化 (agent: 1) され稼働している場合、整合性を向上させるために guest-fsfreeze-freeze および guest-fsfreeze-thaw を呼び出します。

### 注記

Windowsゲストでは、ゲスト内で別のバックアップソフトウェアを使用する場合、ゲストエージェントの設定が必要です。詳細はゲストエージェントセクションの [「Freeze & Thaw」](#) を参照してください。

QemuServer向けProxmox VEライブバックアップの技術概要は、[こちらでオンライン閲覧可能](#)です。

### 注意

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動作を実現します。基盤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プロセス経由で実行されるため、停止中のVMはQEMUがVMディスクを読み取る間、短時間「実行中」と表示されることにご注意ください。ただしVM自体は起動しておらず、ディスクのみが読み取られています。

コンテナのバックアップモード:

## 停止モード

バックアップ中はコンテナを停止します。これにより、非常に長いダウンタイムが発生する可能性があります。

## サスPENDモード

このモードではrsyncを使用してコンテナデータを一時的な場所 (--tmpdirオプション参照) にコピーします。その後コンテナをサスPENDし、変更されたファイルのみを再度rsyncでコピーします。その後コンテナを再開(再開)します。これによりダウンタイムは最小限に抑えられますが、コンテナのコピーを保持するための追加スペースが必要です。

コンテナがローカルファイルシステム上にあり、バックアップの保存先がNFS/CIFSサーバーの場合、--tmpdirもローカルファイルシステム上に設定してください。これによりパフォーマンスが大幅に向かいます。また、バックアップ保存先がNFSサーバーで、サスPENDモードでACLを使用したローカルコンテナをバックアップする場合も、ローカルのtmpdirの使用が必須です。

## スナップショットモード

このモードでは、基盤となるストレージのスナップショット機能を利用します。まず、データの整合性を確保するためコンテナを一時停止します。コンテナのボリュームに対する一時的なスナップショットを作成し、その内容をtarファイルにアーカイブします。最後に、一時スナップショットを再度削除します。

### 注記

スナップショットモードでは、バックアップ対象の全ボリュームがスナップショット対応ストレージ上に存在する必要があります。`backup=no` マウントポイントオプションを使用することで、個々のボリュームをバックアップ対象から除外できます（この要件も適用除外となります）。

### 注

デフォルトでは、ルートディスクの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することで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。デバイスおよびバインドマウントは、その内容が Proxmox VE ストレージライブラリ外で管理されるため、バックアップ対象外です。

## 16.1.1 VMバックアップのコピー処理

VMのバックアップが開始されると、QEMUはブロック層に「コピー・ビフォア・ライト」フィルターを適用します。このフィルターにより、ゲストが新規書き込みを行う際、バックアップに必要な古いデータがまずバックアップ対象に送信されます。この操作が完了するまでゲストの書き込みはブロックされるため、バックアップ対象の速度によって、まだバックアップされていないセクターへのゲストIOが制限されます。

バックアップ・フリーシングでは、このような古いデータはバックアップ対象に直接送信される代わりに、フリーシングイメージにキャッシュされます。これによりゲストのIOパフォーマンスが向上し、特定のシナリオではハングを防止することも可能ですが、その代償としてより多くのストレージ容量が必要となります。

ストレージ local-lvm に作成された fleecing イメージを使用して VM 123 のバックアップを手動で開始するには、以下を実行します。

```
vzdump 123 --fleecing enabled=1,storage=local-lvm
```

特定のバックアップジョブに対してオプションを設定することも、[設定オプション](#)を介してノード全体のフォールバックとして設定することも可能です。UIでは、バックアップジョブ編集時の詳細タブで fleecing を設定できます。

fleecing ストレージは高速なローカルストレージである必要があり、シンプロビジョンングと廃棄サポートを備えている必要があります。例としては、LVM-thin、RBD、ストレージ構成でスベース1を指定したZFS、多くのファイルベースストレージが挙げられます。理想的には、fleecingストレージは専用ストレージであるため、容量が満杯になっても他のゲストに影響せず、バックアップが失敗するだけです。バックアップ済みの fleecing イメージの一部は、使用容量を抑えるために破棄されます。

ディスク破棄をサポートしないファイルベースのストレージ（例：バージョン4.2以前のNFS）では、ストレージ設定で事前割り当てを無効にする必要があります。qcow2（ストレージがサポートする場合、fleecing イメージのフォーマットとして自動的に使用される）と組み合わせることで、既に割り当てられたイメージ領域を後で再利用できる利点があり、これによりかなりの容量節約が可能です。

### 警告

スベースオプションなしのLVMやZFSなど、シンプロビジョンングされていないストレージでは、fleecing イメージ用に元のディスクのフルサイズを事前に確保する必要があります。シンプロビジョン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処理中の際にゲストがディスク全体を上書きする場合に限り、fleecing イメージは元のイメージと同じサイズまで成長できます。

## 16.1.2 CT 変更検出モード

変更検出モードの設定は、pxarアーカイブのエンコーディング形式と、Proxmox Backup Serverをターゲットとするコンテナバックアップにおける変更済みファイルおよび変更なしファイルの処理方法を定義します。

変更検出モードオプションは、ジョブ編集時の詳細タブで個々のバックアップジョブごとに設定可能です。さらに、[設定オプション](#)経由でノード全体のフォールバックとして設定することもできます。

利用可能な変更検出モードは3種類です：

モード	説明
デフォルト	すべてのファイルを読み込み、pxar形式バージョン1を使用して単一のアーカイブにエンコードします。
データ	すべてのファイルを読み込みエンコードするが、データとメタデータを別々のストリームに分割する。 pxarフォーマットバージョン2を使用する。
メタデータ	ストリームを分割し、データと同様にアーカイブ形式バージョン2を使用しますが、変更されていないファイルを検出するために（存在する場合）以前のスナップショットのメタデータアーカイブを使用し、ディスクからファイルの内容を読み取ることなく、可能な限りそれらのデータチャンクを再利用します。 可能な限り。

変更検出モードメタデータを使用したバックアップを実行するには、以下を実行できます。

```
vzdump 123 --storage pbs-storage --pbs-change-detection-mode=metadata
```

#### 注記

Proxmox Backup Server以外のストレージバックエンドへのVMのバックアップは、この設定の影響を受けません。

## 16.2 バックアップファイル名

新しいバージョンの `vzdump` は、ゲストタイプとバックアップ時間を作成するファイル名にエンコードします。例えば

```
vzdump-lxc-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可能になります。さまざまな保存期間オプションを使用して、保存するバックアップの数を制限することができます。詳細は、以下の[「バックアップの保存期間」](#)セクションをご覧ください。

## 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、以下のアルゴリズムのいずれかで圧縮できます：`lzo`<sup>1</sup>、`gzip`<sup>2</sup>、または`zstd`<sup>3</sup>。

現在、`Zstandard`(`zstd`)はこれら3つのアルゴリズムの中で最速です。マルチスレッド処理も、`lzo`や`gzip`に対する`zstd`の利点です。`lzo`と`gzip`はより広く使用されており、デフォルトでインストールされていることがよくあります。

マルチスレッドによるパフォーマンス向上のため、`gzip`の代替として`pigz`<sup>4</sup>をインストールできます。`pigz`および`zstd`では、スレッド数/コア数を調整可能です。詳細は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が使用されたかを判別できる場合が多い。

<sup>1</sup> Lempel-Ziv-Oberhumer (LZO) : 無損失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 a lossless data compression algorithm <https://en.wikipedia.org/wiki/Zstandard>

<sup>4</sup> pigz - gzip の並列実装 <https://zlib.net/pigz/>

.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拡張子で終わっていない場合、vzdumpによって圧縮されていません。

## 16.4 バックアップの暗号化

Proxmox Backup Server ストレージでは、バックアップのクライアント側暗号化をオプションで設定できます。[対応するセクション](#)を参照してください。

## 16.5 バックアップジョブ

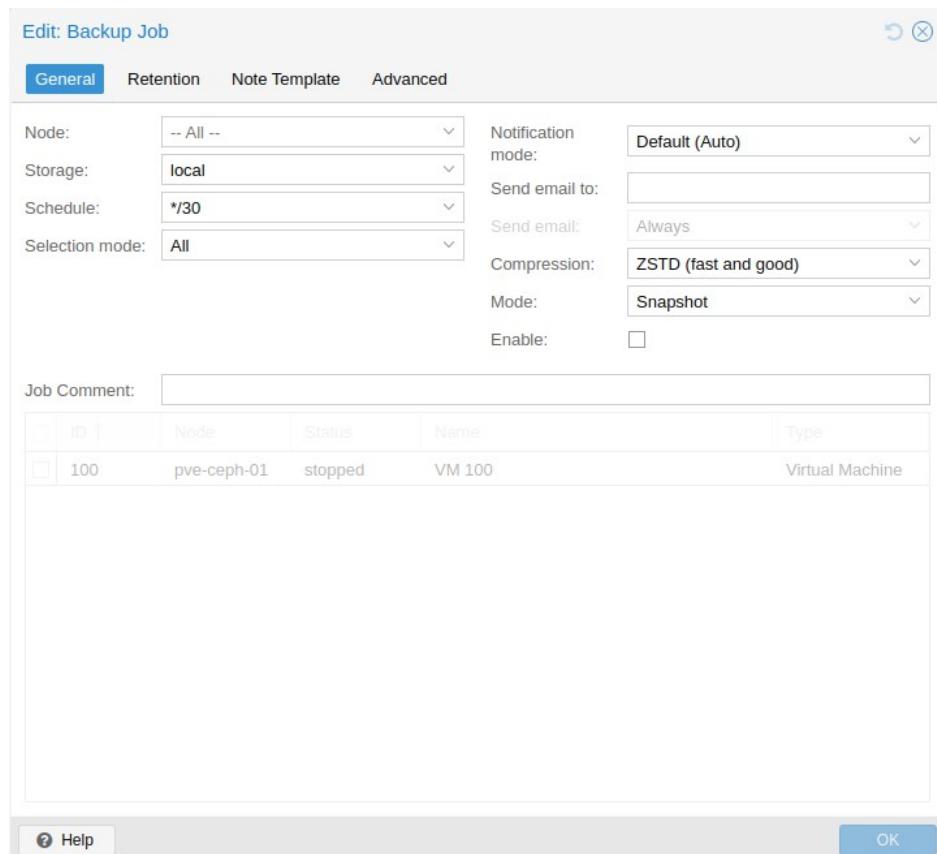
The screenshot shows the Proxmox VE Datacenter interface. The left sidebar is titled 'Datacenter' and contains a tree view of management sections: Search, Summary, Notes, Cluster, Ceph, Options, Storage, and Backup (which is selected). Under Backup, there are sub-options: Replication, Permissions (with sub-options for Users, API Tokens, Two Factor, Groups, Pools, Roles, and Realms), HA, SDN (with sub-options for Zones and VNets), Options, IPAM, ACME, Firewall, Metric Server, Resource Mappings, Notifications, and Support.

The main content area displays a table of scheduled backup jobs:

Enabled	Node	Schedule	Next Run	Storage	Comm...	Retention	Selection
-- All --	*30	2024-04-23 11:00:00	local	Fallback from storag...	-- 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ジョブを設定することもできます。ジョブはUIの「データセンター」→ *Backup* または

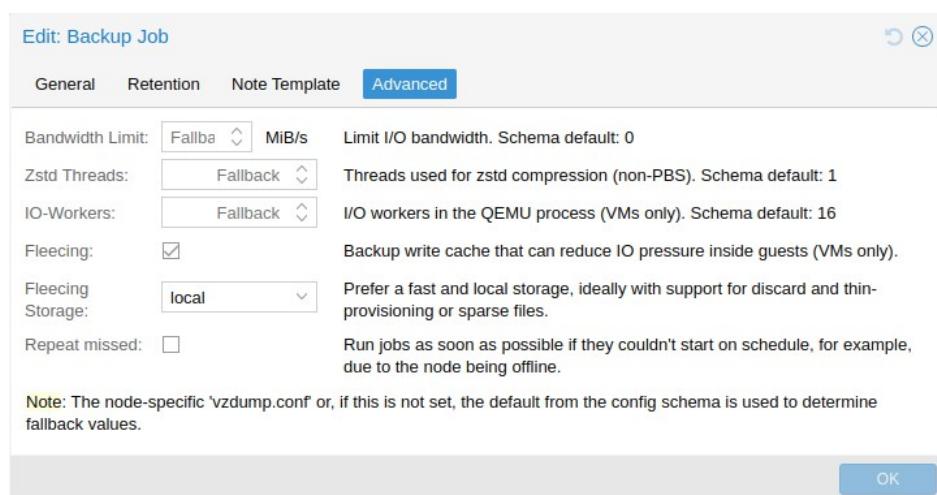
/cluster/backup API エンドポイントからも管理できます。いずれの方法でも、/etc/pve/jobs.cfg にジョブエントリが生成され、pvescheduler デーモンによって解析・実行されます。



ジョブはクラスタ全ノードまたは特定ノード向けに設定され、指定されたスケジュールに従って実行されます。スケジュールの形式はsystemdカレンダーイベントと非常に類似しており、詳細は[カレンダーアイベントセクション](#)を参照してください。UIの「スケジュール」フィールドは自由に編集可能で、ドロップダウンリストには開始点として使用できる複数の例が含まれています。

ストレージやノード設定の保持オプションを上書きするジョブ固有の保持オプションや、バックアップと共に保存される追加情報用の[メモテンプレート](#)を設定できます。

スケジュールされたバックアップは、ホストがオフラインだった場合や、スケジュールされた時間に pvescheduler が無効になっていた場合に実行を逃します。そのため、遅延したジョブを処理する動作を設定することができます。「遅延ジョブを再実行」オプション (UI の [詳細設定] タブ、設定ファイルでは repeat-missed) を有効にすることで、スケジューラに遅延したジョブを可能な限り早く実行するよう指示できます。

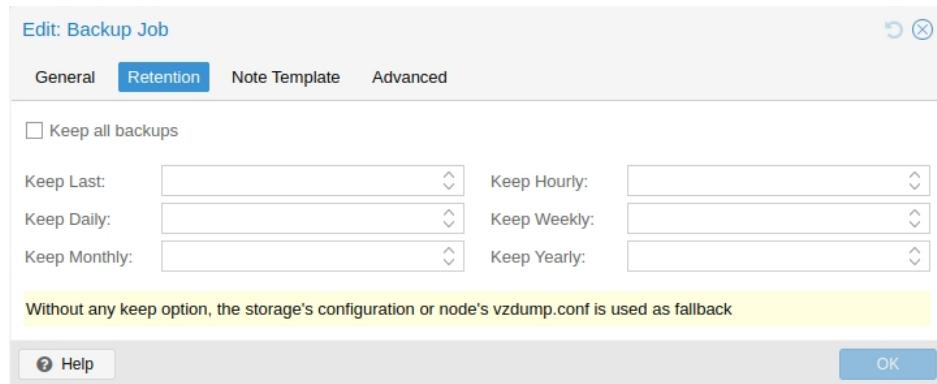


バックアップパフォーマンスを調整するための設定がいくつかあります（一部はUIの[詳細設定]タブで公開されています）。最も注目すべきは、IO帯域幅を制限するbwlimitです。圧縮に使用するスレッド数は、それぞれpigz (gzipの代替) およびzstd設定で制御できます。さらに、

ionice (BFQスケジューラ使用時)、パフォーマンス設定の一部であるmax-workers (VMバックアップのみに影響)、pbs-entries-max (コンテナバックアップのみに影響) があります。詳細は設定オプションを参照してください。

## 16.6 バックアップの保持期間

prune-backups オプションを使用すると、柔軟に保持するバックアップを指定できます。



以下の保持オプションが利用可能です：

**keep-all <boolean>**

すべてのバックアップを保持します。これが true の場合、他のオプションは設定できません。

**keep-last <N>**

最後の <N> 個のバックアップを保持します。

**keep-hourly <N>**

過去 <N> 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、最新のバックアップのみが保持されます。

**keep-daily <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、最新の1つだけが保持されます。

**keep-weekly <N>**

過去 <N> 週間のバックアップを保持します。1週間にに対して複数のバックアップがある場合、最新のバックアップのみを保持します。

---

### 注記

週は月曜日に始まり日曜日に終わります。本ソフトウェアはISO週日付システムを採用し、年末の週を正しく処理します。

---

**keep-monthly <N>**

過去 <N> ヶ月分のバックアップを保持します。同一月に複数のバックアップが存在する場合、最新のバックアップのみを保持します。

---

**keep-yearly &lt;N&gt;**

過去 &lt;N&gt; 年分のバックアップを保持します。1年につき複数のバックアップがある場合、最新のバックアップのみを保持します。

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。次のオプションは既に処理済みのバックアップには影響せず、より古いバックアップのみを考慮します。

使用する保持オプションをカンマ区切りリストで指定します。例:

```
# vzdump 777 --prune-backups keep-last=3,keep-daily=13,keep-yearly=9
```

prune-backups を vzdump に直接渡すことも可能ですが、多くの場合、Web インターフェース経由でストレージレベルで設定を構成する方が合理的です。

## 16.6.1 削除シミュレーター

Proxmox Backup Server のドキュメントにある削除シミュレーターを使用すると、様々なバックアップスケジュールにおける異なる保持オプションの効果を確認できます。

## 16.6.2 保存設定の例

バックアップの頻度と古いバックアップの保存期間は、特定のワークロードにおいて、データの変更頻度や古い状態の重要度によって異なる場合があります。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も存在する可能性があります。

この例では、毎日バックアップを実行し、保持期間を10年とし、保存されるバックアップ間の期間が徐々に長くなることを想定しています。

`keep-last=3` - たとえ日次バックアップのみを取っている場合でも、管理者は大規模なアップグレード直前または直後に追加のバックアップを作成したい場合があります。`keep-last`を設定することでこれを保証します。

`keep-hourly` は設定されていません - 日次バックアップではこれは関係ありません。追加の手動バックアップは `keep-last` で既にカバーされています。

`keep-daily=13` - 少なくとも1日をカバーする`keep-last`と組み合わせることで、最低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か月分の週次バックアップを確実に保持します。

`keep-monthly=11` - これまでの`keep`設定と組み合わせることで、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` - 長期アーカイブ用です。前述の設定で当年度をカバーしているため、残りの年数を9年分設定することで、合計で少なくとも10年分のバックアップを保持できます。

環境で最低限要求される保持期間よりも長い期間を設定することを推奨します。必要以上に長いと判断した場合はいつでも短縮できますが、一度削除したバックアップを再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対象としてマークすると、その削除が防止されます。Proxmox VEのUI、CLI、APIを介した保護済みバックアップの削除試行は失敗します。ただし、これはProxmox VEによって強制されるものでファイルシステム自体による制限ではないため、バックアップストレージへの書き込み権限を持つユーザーは、バックアップファイル自体を手動で削除することが依然として可能です。

### 注記

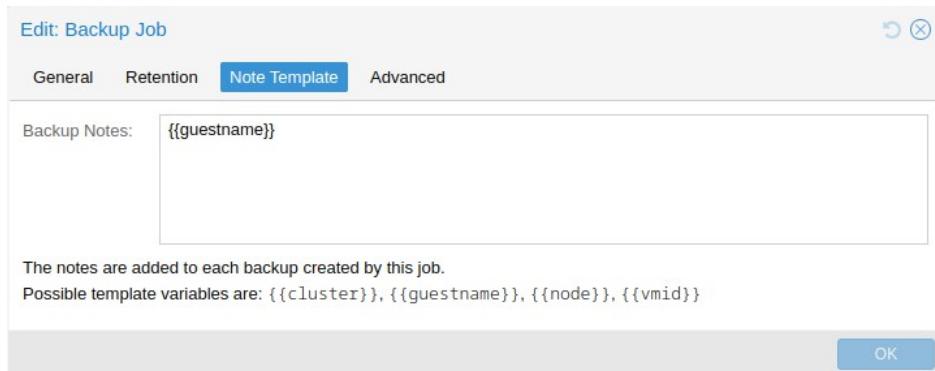
保護されたバックアップは、プルーニングによって無視され、保持設定の対象にはなりません。

ファイルシステムベースのストレージでは、保護は監視ファイル `<backup-name>.protected` を通じて実装されます。Proxmox Backup Server では、サーバー側で処理されます（Proxmox Backup Server バージョン 2.1 以降で利用可能）。

ストレージオプション `max-protected-backups` を使用して、ストレージ上でゲストごとに許可される保護バックアップの数を制御します。無制限の場合は `-1` を使用します。デフォルトは、`Datastore.Allocate` 権限を持つユーザーには無制限、その他のユーザーには 5 です。

## 16.8 バックアップに関する注意事項

バックアップにメモを追加するには、UIの「メモを編集」ボタンを使用するか、ストレージコンテンツAPI経由で行えます。



バックアップジョブおよび手動バックアップ用に、動的にメモを生成する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、中括弧で囲まれた変数を記述でき、バックアップ実行時に該当する値に置換されます。

現在サポートされている変数は以下の通りです：

- `{{cluster}}` クラスタ名（存在する場合）
- `{{guestname}}` 仮想ゲストに割り当てられた名前
- `{{node}}` バックアップが作成されているノードのホスト名
- `{{vmid}}` ゲストの数値VMID

APIまたはCLI経由で指定する場合、改行とバックスラッシュはそれぞれリテラルな`\n`と`\\"`としてエスケープされた單一行である必要があります。

## 16.9 復元

バックアップアーカイブは、Proxmox VE Web GUI または以下の CLI ツールを使用して復元できます：

**pct restore**  
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

詳細は対応するマニュアルページを参照してください。

### 16.9.1 帯域幅制限

1つ以上の大規模なバックアップを復元するには、特にバックアップストレージからの読み取りとターゲットストレージへの書き込みの両方におけるストレージ帯域幅など、多くのリソースが必要になる場合があります。これにより、ストレージへのアクセスが混雑し、他の仮想ゲストに悪影響を及ぼす可能性があります。

これを回避するため、バックアップジョブに帯域幅制限を設定できます。Proxmox VEでは復元とアーカイブ用に2種類の制限を実装しています：

- 復元ごとの制限：バックアップアーカイブからの読み取りに割り当てられる最大帯域幅
- 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します

読み取り制限は書き込み制限に間接的に影響します。読み取り量を超える書き込みはできないためです。ジョブごとの制限値が小さい場合、ストレージごとの制限値を上書きします。ジョブごとの制限値が大きい場合、影響を受けるストレージに対して「Data.Allocate」権限を持っている場合にのみ、ストレージごとの制限値を上書きします。

復元ジョブ固有の帯域幅制限を設定するには、復元CLIコマンドの「--bwlimit <integer>」オプションを使用できます。制限の単位はKiB/sです。つまり`10240`を指定すると、バックアップの読み取り速度が10MiB/sに制限され、残りのストレージ帯域幅が稼働中の仮想ゲストに確保されるため、バックアップがゲストの動作に影響を与えません。

#### 注記

特定の復元ジョブで全ての制限を無効化するには、bwlimitパラメータに「0」を指定できます。非常に重要な仮想ゲストを可能な限り迅速に復元する必要がある場合に有用です（ストレージに対する`Data.Allocate`権限が必要です）。

多くの場合、ストレージの一般的に利用可能な帯域幅は時間経過で変化しません。そのため、設定済みストレージごとにデフォルトの帯域幅制限を設定する機能を実装しました。設定方法は以下の通りです：

```
# pvesm set STORAGEID --bwlimit restore=KIBs
```

## 16.9.2 ライブ復元

大規模なバックアップの復元には長時間かかり、その間ゲストは利用不可状態が続きます。Proxmox Backup Serverに保存されたVMバックアップの場合、ライブ復元オプションを使用することでこの待機時間を軽減できます。

GUIのチェックボックスまたはqmrestoreの--live-restore引数でライブ復元を有効にすると、復元開始と同時に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を優先します。

ただし、以下の2点に留意してください：

- ライブリストア中は、データがバックアップサーバーから読み込まれる必要があるため、VMのディスク読み取り速度は制限されます（ただし読み込み完了後は直ちに宛先ストレージで利用可能となるため、データへの二重アクセスによるペナルティは初回のみ発生します）。書き込み速度はほぼ影響を受けません。
- 何らかの理由でライブリストアが失敗した場合、VMは未定義の状態（バックアップからのデータが完全に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の保持はほぼ不可能）となります。

この動作モードは、初期動作に必要なデータ量が少ない大規模VM（例：Webサーバー）に特に有用です。OSと必須サービスが起動されればVMは稼働状態となり、バックグラウンドタスクが使用頻度の低いデータのコピーを継続します。

## 16.9.3 単一ファイル復元

ストレージGUIの「バックアップ」タブにある「ファイル復元」ボタンを使用すると、バックアップ内のデータに対して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利用可能です。

コンテナの場合、ファイルツリーの最上位層には含まれるすべてのpxarアーカイブが表示され、自由に開いて閲覧できます。VMの場合、最上位層には含まれるドライブイメージが表示され、開くとドライブ上で検出されたサポート対象ストレージ技術のリストが表示されます。最も基本的なケースでは、これは「part」と呼ばれるエントリとなり、パーティションテーブルを表します。このテーブルにはドライブ上で検出された各パーティションのエントリが含まれます。VMの場合、すべてのデータにアクセスできるとは限りません（非対応のゲストファイルシステム、ストレージ技術など）。

ファイルとディレクトリはダウンロードボタンで取得可能で、ディレクトリはオンザフライでzipアーカイブに圧縮されます。

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（ゲストとして表示されない）が起動されます。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ありませんが、ハイパーテナシスシステムを危険に晒すことを回避します。VMはタイムアウト後に自動的に停止します。このプロセス全体は、ユーザー視点では透過的に行われます。

### 注記

トラブルシューティングについては トラブルシューティング 目的で 各 一時的な VM インスタンス 生成する  
生成する ファイル ファイル /var/log/proxmox-backup

/var/log/proxmox-backup/file-restore/ に生成されます。バックアップアーカイブ内の個々のファイルの復元やファイルシステムへのアクセスが失敗した場合、ログファイルに追加情報が記録されることがあります。

## 16.10 設定

グローバル設定は `/etc/vzdump.conf` に保存されます。このファイルはコロン区切りのキー/値形式を使用します。各行は次の形式です:

OPTION: value

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同様に無視されます。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可能です。

現在サポートされているオプションは以下の通りです:

**bwlimit: <整数> (0 - N) (デフォルト= 0)**

I/O 帯域幅を制限します (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します。

**dumpdir: <string>**

結果ファイルを指定ディレクトリに保存します。

**exclude-path: <array>**

特定のファイル/ディレクトリを除外します (シェルグロブ)。./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を基準に相対的に一致します。

**fleecing: [[enabled=<1|0>] [,storage=<storage ID>]]**

バックアップ・フリーシングのオプション (VMのみ)。

**有効化= <boolean> (デフォルト= 0)**

バックアップのフリッピングを有効化します。新規ゲスト書き込みが発生したブロックのバックアップデータを、バックアップ先へ直接コピーする代わりに指定ストレージにキャッシュします。これによりゲストのIOパフォーマンスが向上し、ハング防止にも寄与しますが、より多くのストレージ容量を必要とする点が欠点です。

**storage= <storage ID>**

バックアップイメージの保存に使用するストレージ。効率的なスペース利用のため、ディスク破棄をサポートし、シンプロビジョンングまたはスペースファイルのいずれかをサポートするローカルストレージの使用が最適です。

**ionice: <整数> (0 - 8) (デフォルト= 7)**

BFQスケジューラ使用時のIO優先度を設定します。VMのスナップショットおよびサスペンドモードバックアップでは、これは圧縮モジュールにのみ影響します。値8はアイドル優先度を使用することを意味し、それ以外の場合は指定値でベストエフォート優先度が使用されます。

**lockwait: <integer> (0 - N) (デフォルト= 180)**

グローバルロック待機の上限時間 (分)。

**mailnotification: <常に| 失敗時> (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。通知メールを送信するタイミングを指定します

**mailto: <string>;**

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。メール通知を受信すべきメールアドレスまたはユーザーのカンマ区切りリスト。

**maxfiles: <integer>; (1 - N)**

非推奨: *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: <snapshot| stop| suspend>; (デフォルト= snapshot)**

バックアップモード。

**notes-template: <string>;**

バックアップのノートを生成するためのテンプレート文字列。変数を含めることができます、それらは値に置換されます。現在サポートされているのは `\{cluster\}`、`\{guestname\}`、`\{node\}`、および`\{vmid\}`ですが、将来追加される可能性があります。1行で記述する必要があり、改行とバックスラッシュはそれぞれ\nと\\でエスケープする必要があります。

---

**注記**

必要なオプション: storage

---

**notification-mode: <auto| legacy-sendmail| notification-system>; (デフォルト= auto)**

使用する通知システムを決定します。*legacy-sendmail* に設定した場合、vzdump は mailto-/mailnotification パラメータを考慮し、sendmail コマンド経由で指定されたアドレスにメールを送信します。*notification-system* に設定した場合、PVE の通知システム経由で通知が送信され、mailto および mailnotification は無視されます。*auto* (デフォルト設定) に設定した場合、mailtoが設定されている場合はメールが送信され、設定されていない場合は通知システムが使用されます。

**pbs-change-detection-mode: <data| legacy| metadata>;**

ファイル変更の検出およびコンテナバックアップのエンコーディング形式切り替えに使用するPBSモード。

**performance: [max-workers=<integer>;] [,pbs-entries-max=<integer>;]**

その他のパフォーマンス関連設定。

**max-workers= <integer>; (1 - 256) (デフォルト= 16)**

仮想マシンに適用されます。同時に許可される最大IOワーカー数を指定します。

**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ます。意図しないOOM（メモリ不足）状態を回避するため、メモリ内に同時に保持できるエントリ数を制限します。大量のファイルを含むコンテナのバックアップを有効にするには、この値を増やしてください。

**pigz: <integer>; (デフォルト= 0)**

N>0の場合、gzipの代わりにpigzを使用します。N=1ではコア数の半分を使用し、N>1ではNをスレッド数として使用します。

**pool: <string>;**

指定されたプールに含まれる既知のゲストシステム全てをバックアップします。

---

**protected: &lt;boolean&gt;**

trueの場合、バックアップを保護済みとしてマークします。

**注記**

必要なオプション: storage

```
prune-backups: [keep-all=&lt;1|0&gt;] [,keep-daily=&lt;N&gt;] [,keep-hourly=&lt;N&gt;] [,keep-last=&lt;N&gt;] [,keep-monthly=&lt;N&gt;] [,keep-weekly=&lt;N&gt;]  
[,keep-yearly=&lt;N&gt;] (デフォルト= keep-all=1)
```

ストレージ設定の保持オプションの代わりに、これらの保持オプションを使用してください。

**keep-all= &lt;boolean&gt;**

すべてのバックアップを保持します。trueの場合、他のオプションと競合します。

**keep-daily= &lt;N&gt;**

過去 &lt;N&gt; 日分の異なる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、最新の1つだけが保持されます。

**keep-hourly= &lt;N&gt;**

過去 &lt;N&gt; 時間の異なる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、最新の1つだけが保持されます。

**keep-last= &lt;N&gt;**

直近の &lt;N&gt; 個のバックアップを保持します。

**keep-monthly= &lt;N&gt;**

過去 &lt;N&gt; ヶ月分の異なるバックアップを保持します。単一月に複数のバックアップがある場合、最新の1つだけが保持されます。

**keep-weekly= &lt;N&gt;**

過去 &lt;N&gt; 週間分の異なるバックアップを保持する。同一週に複数のバックアップが存在する場合、最新のバックアップのみを保持する。

**keep-yearly= &lt;N&gt;**

過去 &lt;N&gt; 年間の異なるバックアップを保持します。1年間に複数のバックアップがある場合、最新のバックアップのみを保持します。

```
remove: &lt;boolean&gt; (デフォルト= 1)
```

prune-backups 設定に従い古いバックアップを削除します。

```
script: &lt;string&gt;
```

指定されたフックスクリプトを使用します。

```
stdexcludes: &lt;boolean&gt; (デフォルト= 1)
```

一時ファイルとログを除外します。

```
stopwait: &lt;integer&gt; (0 - N) (デフォルト= 10)
```

ゲストシステムの停止まで待機する最大時間(分単位)。

**storage: &lt;ストレージID&gt;**

結果ファイルをこのストレージに保存します。

**tmpdir: &lt;string&gt;**

一時ファイルを指定されたディレクトリに保存します。

**zstd: &lt;integer&gt; (デフォルト= 1)**

Zstdスレッド数。N=0の場合、利用可能なコアの半数を使用します。Nが0より大きい値に設定された場合、Nがスレッド数として使用されます。

**vzdump.conf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プロセスの各段階で呼び出され、対応するパラメータが設定されます。ドキュメントディレクトリに例があります(vzdump-hook-script.pl)。

## 16.12 ファイル除外

**注意**

このオプションはコンテナバックアップでのみ利用可能です。

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 オプションで無効化可能)

```
/tmp/* /var/tmp/* /var/tmp/  
/var/tmp/*  
/var/run/* pid
```

手動で（追加の）除外パスを指定することも可能です。例：

```
# vzdump 777 --exclude-path /tmp/ --exclude-path '/var/foo*' '
```

/tmp/ディレクトリおよび/var/foo、/var/foobarなどの名前のファイルやディレクトリを除外します。

**警告**

 Proxmox Backup Server (PBS)へのバックアップおよびサスペンドモードバックアップでは、末尾にスラッシュが付いたパターンはディレクトリには一致しますが、ファイルには一致しません。一方、PBS以外のスナップショットモードおよびストップモードバックアップでは、末尾にスラッシュが付いたパターンは現在まったく一致しません。これは tar コマンドがそれをサポートしていないためです。

/ で始まらないパスはコンテナのルートに固定されず、任意のサブディレクトリを基準に相対的に一致します。例:

```
# vzdum 777 --exclude-path bar
```

/bar、/var/bar、/var/foo/barなどのファイルやディレクトリは除外されますが、/bar2は除外されません。

設定ファイルもバックアップアーカイブ内 (./etc/vzdum/) に保存され、正しく復元されます。

## 16.13 使用例

ゲストの単純なダンプ777 - スナップショットは作成せず、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 /var/lib/vz/dump/）にアーカイブします。

```
# vzdum 777
```

rsyncとサスペンド/レジュームを使用してスナップショットを作成する（最小限のダウンタイム）。

```
# vzdum 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する。デフォルトでmailtoが設定され、notification-modeが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コマンド経由で送信される。

```
# vzdum --all --mode suspend --mailto root --mailto admin
```

スナップショットモード（ダウンタイムなし）とデフォルト以外のダンプディレクトリを使用する。

```
# vzdum 777 --dumpdir /mnt/backup --mode snapshot
```

複数のゲストを選択的にバックアップ

```
# vzdum 101 102 103 --mailto root
```

101と102を除く全てのゲストをバックアップ

```
# vzdum --mode suspend --exclude 101,102
```

コンテナを新しいCT 600に復元

```
# pct restore 600 /mnt/backup/vzdum-lxc-777.tar
```

QemuServer VM を VM 601 に復元

```
# qmrestore /mnt/backup/vzdum-qemu-888.vma 601
```

既存コンテナ 101 を 4GB ルートファイルシステムで新規コンテナ 300 にクローン（パイプ使用）

```
# vzdum 101 --stdout| pct restore --rootfs 4 300 -
```

## 第17章 通知

### 17.1 概要

- Proxmox VEは、ストレージレプリケーションの失敗、ノードフェンシング、バックアップの完了/失敗、その他のイベントが発生した場合に[通知イベント](#)を発行します。これらのイベントは、グローバル通知設定に基づいて処理されます。各通知イベントには、タイムスタンプ、重大度レベル、タイプ、およびイベント固有の追加フィールドなどのメタデータが含まれます。
- [通知マッチャー](#)は、通知イベントを1つ以上の通知ターゲットにルーティングします。マッチャーは、通知イベントのメタデータに基づいて選択的にルーティングするためのマッチルールを持つことができます。
- [通知ターゲット](#)は、マッチャーによって通知イベントがルーティングされる宛先です。ターゲットにはメールベース（SendmailおよびSMTP）とGotifyの複数タイプがあります。

グローバル通知設定は、GUIの「Datacenter」>「→」>「Notifications」で構成できます。設定は/etc/pve/notifications.cfgおよび/etc/pve/priv/notifications.cfgに保存されます  
- 後者は通知ターゲット用のパスワードや認証トークンなどの機密設定オプションを含み  
対象の認証トークンなどの機密設定オプションが含まれており、rootのみが読み取り可能です。

### 17.2 通知ターゲット

Proxmox VE は複数の通知ターゲットタイプを提供します。

#### 17.2.1 Sendmail

The screenshot shows the 'Add: Sendmail' dialog box. It contains the following fields:

- Endpoint Name: sendmail-admins
- Enable:
- Recipient(s): root@pam
- Additional Recipient(s): admin@example.org
- Comment: Send notification mail to admins

At the bottom right are 'Help' and 'Advanced' buttons, and a large blue 'Add' button.

sendmail バイナリは、Unix 系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あり、電子メールメッセージの送信を処理します。これは、ユーザーやアプリケーションがコマンドラインから、またはスクリプト内から直接電子メールを送信できるようにするコマンドラインユーティリティです。

sendmail通知ターゲットは、sendmailバイナリを使用して、設定済みのユーザーまたはメールアドレスのリストにメールを送信します。ユーザーが受信者として選択された場合、ユーザー設定で設定されたメールアドレスが使用されます。root@pamユーザーの場合、これはインストール時に入力されたメールアドレスです。ユーザーのメールアドレスは、Datacenter→ Permissions→ Users で設定できます。ユーザーに関連付けられたメールアドレスがない場合、メールは送信されません。

---

#### 注記

標準的な Proxmox VE インストール環境では、sendmail バイナリは Postfix によって提供されます。メールを正しく配信できるように Postfix を設定する必要がある場合があります（例：外部メール中継サーバー（スマートホスト）の設定）。配信に失敗した場合は、Postfix デーモンによって記録されたメッセージをシステムログで確認してください。

---

Sendmail 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複数回設定可能。
- **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` から検索されます。複数の受信者に対応するため複数回設定可能です。
- **author:** メールの送信者を設定します。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。それも設定されていない場合、プラグインは `root@$hostname` をデフォルトとします。ここで `$hostname` はノードのホスト名です。メールのFromヘッダーは `$author <$from-address>` に設定されます。
- **comment:** このターゲットに関するコメント

設定例 (`/etc/pve/notifications.cfg`):

```
sendmail: example
    mailto-user root@pammmailto-user
    admin@pvemailto max@example.com
    from-address pve1@example.com
    コメント 複数のユーザー/アドレスに送信
```

## 17.2.2 SMTP

Add: SMTP

Endpoint Name:	smtp-example		
Enable:	<input checked="" type="checkbox"/>		
Server:	mail.example.org	Authenticate:	<input checked="" type="checkbox"/>
Encryption:	TLS	Username:	pve-mail
Port:	Default (465)	Password:	*****
From Address:	pve-mail@example.org		
Recipient(s):	root@pam		
Additional Recipient(s):	admin@example.org		
Comment:	Send notifications via external SMTP relay		

[Help](#) [Advanced](#) [Add](#)

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できます。このターゲットはシステムのMTAを使用せずメールを配信します。sendmailターゲットと同様に、ユーザーが受信者として選択された場合、そのユーザーの設定済みメールアドレスが使用されます。

### 注

sendmailターゲットとは異なり、SMTPターゲットはメール配信失敗時のキューイング/再試行メカニズムを備えていません。

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:

- `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複数回設定可能。
- `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` から検索されます。複数の受信者に対応するため複数回設定可能です。
- `author`: メールの差出人を設定します。デフォルトは Proxmox VE です。
- `from-address`: メールの差出人アドレスを設定します。SMTPリレーでは、なりすましを防止するため、このアドレスがユーザー自身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ヘッダーは `$author <$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 認証方式がサポートされています。
- `password`: 認証時に使用するパスワード。
- `mode`: 暗号化モードを設定します (`insecure`、`starttls`、または `tls`)。デフォルトは `tls` です。
- `server`: SMTPリレーのアドレス/IP。
- `port`: 接続するポート番号。設定しない場合、デフォルトのポートが使用されます。モードの値に応じて、デフォルトは 25 (非暗号化)、465 (TLS)、または 587 (STARTTLS) です。
- `comment`: このターゲットのコメント

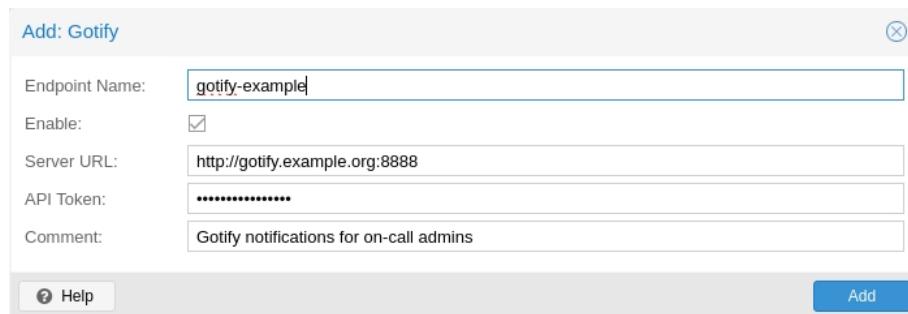
設定例 (/etc/pve/notifications.cfg):

```
smtp: example
    mailto-user root@pve
    mailto-user admin@pve
    max@example.com
    from-address pve1@example.com
    username pve1
    server mail.example.com mode
    starttls
```

/etc/pve/priv/notifications.cfg 内の対応するエントリ（シークレットトークンを含む）：

```
smtp: example
    password somepassword
```

### 17.2.3 Gotify



Gotifyはオープンソースのセルフホスト型通知サーバーであり、様々なデバイスやアプリケーションへのプッシュ通知の送受信を可能にします。シンプルなAPIとウェブインターフェースを提供し、異なるプラットフォームやサービスとの統合を容易にします。

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：

- server: GotifyサーバーのベースURL（例: http://<ip>:8888）
- token: 認証トークン。トークンはGotifyのWebインターフェース内で生成できます。
- comment: このターゲットへのコメント

#### 注記

Gotifyターゲットプラグインは、[データセンター設定](#)のHTTPプロキシ設定を尊重します

設定例 (/etc/pve/notifications.cfg):

```
gotify: example
    server http://gotify.example.com:8888 comment 複数のユーザー/ア
    ドレスに送信
```

/etc/pve/priv/notifications.cfg 内の対応するエントリ（シークレットトークンを含む）：

```
gotify: example
    token somesecrettoken
```

## 17.2.4 Webhook

Webhook通知ターゲットは、設定可能なURLに対してHTTPリクエストを実行します。以下の設定オプションが利用可能です：

- `url`: HTTPリクエストを実行するURL。メッセージ内容、メタデータ、シークレットを挿入するためのテンプレートをサポートします。
- `method`: 使用するHTTPメソッド (POST/PUT/GET)
- ヘッダー: リクエストに設定すべきHTTPヘッダーの配列。メッセージ内容、メタデータ、シークレットを挿入するためのテンプレートをサポートします。
- `body`: 送信すべきHTTPボディ。メッセージ内容、メタデータ、シークレットを挿入するためのテンプレートをサポート。
- `secret`: シークレットのキーと値のペアの配列。これらはrootのみが読み取り可能な保護された設定ファイルに保存されます。シークレットは、`body/header/URL`テンプレート内で`secrets`名前空間を介してアクセスできます。
- `comment`: このターゲットに対するコメント。

テンプレート化をサポートする設定オプションでは、Handlebars構文を使用して以下のプロパティにアクセスできます：

- `{{ title }}`: 表示された通知のタイトル
- `{{ message }}`: 通知本文の表示内容
- `{{ severity }}`: 通知の重大度 (info, notice, warning, error, unknown)
- `{{ timestamp }}`: UNIXエポック（秒単位）での通知のタイムスタンプ
- `{{ fields.<name> }}`: 通知のメタデータフィールド用のサブネームスペース。例:  
`fields.type` には通知タイプが含まれます - 利用可能な全フィールドについては[通知イベント](#)を参照してください。
- `{{ secrets.<name> }}`: シークレットのサブネームスペース。例えば、`token` という名前のシークレットは `secrets.token` 経由でアクセス可能。

利便性のため、以下のヘルパーが利用可能です：

- `{{ url-encode <value/property> }}`: プロパティ/リテラルをURLエンコードします。
- `{{ escape <value/property> }}`: JSON文字列として安全に表現できない制御文字をエスケープします。
- `{{ json <value/property> }}`: 値をJSONとしてレンダリングします。これは、サブネームスペース全体 (例: `fields`) をJSONペイロードの一部として渡す場合に便利です (例: `{{ json fields }}`)。

## 例

### ntfy.sh

- メソッド: POST
- URL: `https://ntfy.sh/{{ secrets.channel }}`
- ヘッダー:
  - Markdown: はい
- 本文:

```
...  
{{ message }}  
...
```

- シークレット:
  - channel: <あなたの ntfy.sh チャンネル>;

### Discord

- メソッド: POST
- URL: `https://discord.com/api/webhooks/{{ secrets.token }}`
- ヘッダー:
  - Content-Type: application/json
- Body:

```
{  
    "content": "... {{ escape message }} ..."  
}
```

- シークレット:
  - token: <token>;

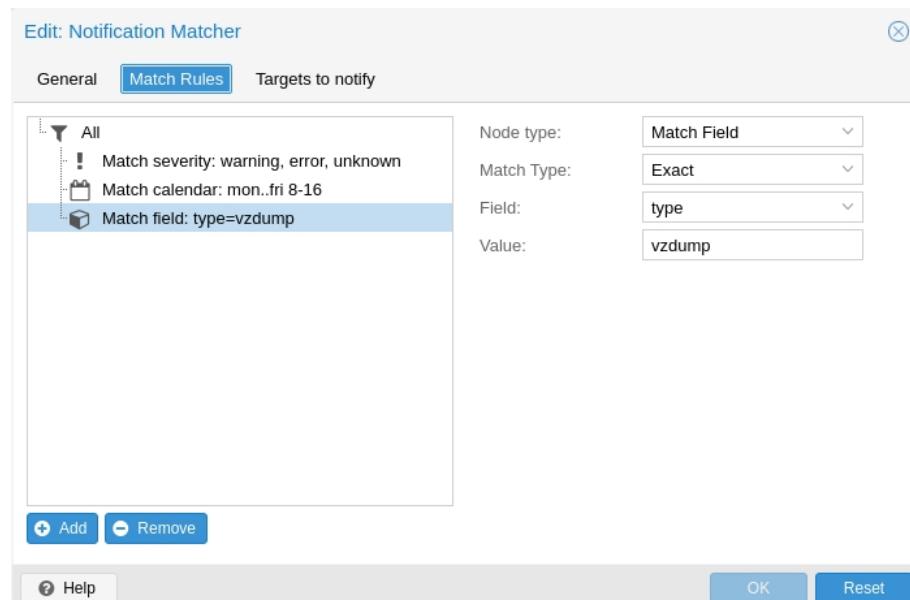
## Slack

- メソッド: POST
- URL: <https://hooks.slack.com/services/{{ secrets.token }}>
- ヘッダー:
  - Content-Type: application/json
- Body:

```
{
  "text": "... {{escape message}} ...",
  "type": "mrkdwn"
}
```

- シークレット:
  - token: <token>

## 17.3 通知マッチャー



通知マッチャーは、その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、通知の特定のプロパティ（例：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)）に一致させることができます。通知がマッチャーに一致した場合、そのマッチャーに設定されたすべてのターゲットが通知を受け取ります。

任意の数のマッチャーを作成でき、それぞれ独自のマッチングルールと通知先ターゲットを持ちます。各ターゲットは、複数のマッチャーで使用されても、通知ごとに最大1回のみ通知されます。

マッチングルールを一切持たないマッチャーは常に真 (true) となり、設定されたターゲットには常に通知が行われます。

```
matcher: 常に一致する
        ターゲット管理者
        コメント このマッチャーは常に一致します
```

### 17.3.1 マッチャーオプション

- `target`: マッチャーがマッチした場合に通知すべき対象を決定します。複数の対象に通知するためには複数回使用できます。
- `invert-match`: マッチャー全体の結果を反転させます
- `mode`: 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を評価する方法を決定します。`all` に設定すると、すべてのマッチルールが一致する必要があります。`any` に設定すると、少なくとも 1 つのルールが一致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します。
- `match-field`: 通知のメタデータフィールドを照合します。
- `match-severity`: 通知の重大度を一致させる。
- `comment`: このマッチャーに対するコメント。

### 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、通知が送信される時刻を、設定可能なスケジュールと照合します。

- `match-calendar 8-12`
- `match-calendar 8:00-15:30`
- `match-calendar 月～金 9:00-17:00`
- `match-calendar 日,火-水,金 9-17`

### 17.3.3 フィールド一致ルール

通知には、照合可能なメタデータフィールドが複数用意されています。照合モードとして `exact` を使用する場合、区切り文字として `,` を使用できます。この照合ルールは、メタデータフィールドが指定された値のいずれかを含む場合に一致します。

- `match-field exact:type=vzdump バックアップに関する通知のみをマッチさせる。`
- `match-field exact:type=replication,fencing レプリケーションとフェンシングの通知をマッチさせる。`
- `match-field regex:hostname=^.+\example\.com$ ノードのホスト名を一致させる。`

一致したメタデータフィールドが存在しない場合、通知は一致しません。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のみに一致しますが、フィールドが存在しない場合は一致しません。

### 17.3.4 重大度マッチングルール

通知には関連付けられた重大度があり、これを照合できます。

- match-severity error: エラーのみにマッチ
- match-severity warning,error: 警告とエラーに一致

以下の重大度が使用されます: info、notice、warning、error、unknown。

### 17.3.5 例

```
matcher: workday
    試合カレンダー 月～金 9-17 担当者管理
    コメント 勤務時間中に管理者に通知

matcher: 夜間と週末
    match-calendar 月-金 9-17 invert-match
    true
    対象: オンコール管理者
    コメント 勤務時間外用の別ターゲット
```

```
matcher: バックアップ障害
    match-field exact:type=vzdump match-
    severity error
    ターゲット: バックアップ管理者
    コメント バックアップ失敗に関する通知を 1 つのグループに送信します。←
        admins

matcher: cluster-failures
    match-field exact:type=replication,fencingtarget 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信する
```

## 17.4 通知イベント

イベント	タイプ	重大度	メタデータフィールド（ タイプに加えて）
システム更新 利用可能	パッケージ更新	info	ホスト名
クラスタノードフェンス化	フェンシング	エラー	hostname
フェンシング 失敗	レプリケーション	エラー	ホスト名、ジョブ ID
バックアップは成功しました	vzdump	info	ホスト名、ジョブ ID (バックアップジョブ専用)

イベント	タイプ	重大度	メタデータフィールド（ タイプに加えて）
バックアップ失敗	vzdump	エラー	ホスト名、ジョブID (バックアップジョブのみ)
root宛メール	システムメール	不明	ホスト名

フィールド名	説明
タイプ	通知の種類
ホスト名	ホスト名（ドメインなし、例: pve1）
job-id	ジョブ ID

#### 注記

バックアップジョブの通知には、スケジュールに基づいて自動的に実行された場合のみジョブIDが設定されます。UIの「今すぐ実行」ボタンによる手動トリガーの場合は設定されません。

## 17.5 システムメール転送

特定のローカルシステムデーモン（例：smartd）は、通知メールを生成し、初期状態ではローカルのrootユーザー宛てに送信されます。これらのメールは、タイプがsystem-mail、重大度がunknownの通知イベントに変換され、グローバル通知設定に基づいて処理されます。

メールがsendmailターゲットに転送される場合、メールの内容とヘッダーはそのまま転送されます。その他のすべてのターゲットに対しては、システムはメール内容から件名と本文の両方を抽出しようと試みます。メールがHTMLコンテンツのみで構成されている場合、この処理中にプレーンテキスト形式に変換されます。

## 17.6 権限

通知ターゲットの設定を変更/閲覧するには、/mapping/notifications ACLノードに対するMapping.Modify/Mapping.Audit権限が必要です。ターゲットのテストには、/mappingに対するMapping.Use、Mapping.Audit または Mapping.Modify 権限が必要です。

## 17.7 通知モード

バックアップジョブでは、バックアップ関連の通知送信に2つのモードから選択できます。これは[バックアップジョブ設定](#)のnotification-modeオプションで制御されます。

- グローバル通知設定 (notification-system) に基づく通知の送信。
- システムのsendmailコマンドを介して、バックアップジョブで設定されたメールアドレスに通知メールを送信します (legacy-sendmail)。このモードでは、メールを常に送信するか、バックアップ失敗時のみ送信するかを選択できます。グローバル通知設定のターゲットやマッチャーは無視されます。このモードは、Proxmox VE 8.1以前のバージョンの通知動作に相当します。Proxmox VEの今後のリリースで廃止される可能性があります。

## 17.8 通知テンプレートのオーバーライド

Proxmox VE は通知のレンダリングに Handlebars テンプレートを使用します。Proxmox VE が提供する元のテンプレートは /usr/share/pve-manager/templates/default/ に保存されています。

通知テンプレートは、オーバーライドディレクトリ

/etc/pve/notification-templates/default/ にカスタムテンプレートファイルを配置することで上書きできます。特定のタイプの通知をレンダリングする際、Proxmox VE はまずオーバーライドディレクトリからテンプレートを読み込むうとします。このテンプレートが存在しないか、レンダリングに失敗した場合、元のテンプレートが使用されます。

テンプレートファイルの命名規則は <type>-<body|subject>. <html|txt>.hbs です。例えば、ファイル vzdump-body.html.hbs はバックアップ通知の HTML 版をレンダリングするためのテンプレートを含み、package-updates-subject.txt.hbs は利用可能なパッケージ更新の通知の件行をレンダリングするために使用されます。

メールベースの通知ターゲット (sendmail や smtp など) は、常に HTML と ブラーンテキスト のマルチパートメッセージを送信します。その結果、メールメッセージのレンダリング時には、<type>-body.html.hbs テンプレートと <type>-body.txt テンプレートの両方が使用されます。その他の通知ターゲットタイプはすべて

<type>-body.txt.hbs テンプレートのみを使用します。

## 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、127.0.0.1:85 で Proxmox VE API 全体を公開します。root として実行され、特権操作をすべて実行する権限を持っています。

---

### 注意

このデーモンはローカルアドレスのみをリッスンするため、外部からはアクセスできません。 [pveproxy](#) デーモンは、API を外部に公開します。

---

### 18.1.1 ワーカー数

pvedaemonは着信リクエストの処理をワーカープロセスに委譲します。デフォルトでは、pvedaemonは3つのワーカープロセスを生成します。これはほとんどのワークロードで十分です。大量のAPIリクエストを発行し、リクエスト処理が遅延したりタイムアウトが発生したりする自動化が中心のワークロードでは、/etc/default/pvedaemon内のMAX\_WORKERSを設定することでワーカープロセスの数を増やすことができます。例：

```
MAX_WORKERS=5
```

ワーカープロセスの数を増やすとCPU使用率が上昇する可能性がある点に注意してください。ワーカープロセスの数は0より大きく128未満である必要があります。

[pveproxy](#)にも同様の設定が存在します。

## 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 上で 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。権限が非常に制限されています。より多くの権限を必要とする操作はローカルの pvedaemon に転送されます。

他のノードを宛先とするリクエストは自動的に該当ノードへ転送されます。これにより、単一の Proxmox VE ノードに接続するだけでクラスター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」のようなアクセス制御リストの設定が可能です。値はファイル /etc/default/pveproxy から読み込まれます。

例:

```
POLICY="allow"
```

IPアドレスはNet::IPが理解する任意の構文で指定できます。名前allは  
0/0 および ::/0 (すべての IPv4 および IPv6 アドレスを意味) の別名です。デフォルトのポリシーは許可です。

Match	POLICY=deny	POLICY=allow
Match 許可のみ	許可	許可
マッチ 拒否のみ	拒否	拒否
一致なし	拒否	許可
許可と拒否の両方に一致	拒否	許可

## 18.2.2 リスニング IP アドレス

デフォルトでは、pveproxy および spiceproxy デーモンはワイルドカードアドレスでリスニングし、IPv4 および IPv6 クライアントの両方からの接続を受け入れます。

/etc/default/pveproxy で LISTEN\_IP を設定することで、pveproxy がどの IP アドレスをリッスンするかを制御できます。  
および spiceproxy デーモンがバインドされます。システム上で IP アドレスを設定する必要があります。

sysctl net.ipv6.bindv6only をデフォルト以外の値 1 に設定すると、デーモンは IPv6 クライアントからの接続のみを受け付けるようになります。ただし、通常は他の多くの問題も引き起こします。この設定を行う場合は、sysctl 設定を削除するか、LISTEN\_IP を 0.0.0.0 に設定することを推奨します（これにより IPv4 クライアントのみが許可されます）。

LISTEN\_IP はソケットを内部インターフェースに限定し、パブリックインターネットへの露出を減らすために使用できます。例：

```
LISTEN_IP="192.0.2.1"
```

同様に、IPv6 アドレスも設定可能です：

```
LISTEN_IP="2001:db8:85a3::1"
```

リンクローカル IPv6 アドレスを指定する場合は、インターフェース名 자체を指定する必要があります。例:

```
LISTEN_IP="fe80::c463:8cff:feb9:6a4e%vmbr0"
```



### 警告

クラスタ内のノードは通信のために pveproxy へのアクセスが必要であり、異なるサブネット上にある可能性があります。クラスタ化されたシステムで LISTEN\_IP を設定することは 推奨されません。

変更を適用するには、ノードの再起動、または pveproxy と spiceproxy の完全な再起動が必要です

```
:  
systemctl restart pveproxy.service spiceproxy.service
```

---

#### 注意

pveproxyサービスの再起動は、リロードとは異なり、仮想ゲストからの実行中のコンソールやシェルなど、一部の長期実行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際はメンテナンスウィンドウをご利用ください。

---

### 18.2.3 SSL暗号スイート

暗号リストは、/etc/default/pveproxy 内の CIPHERS (TLS<= 1.3) および CIPHERSUIT (TLS>= 1.3) キーで暗号リストを定義できます。例:

```
←→      , AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES128-SHA256, ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-CIPHERS="ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-SHA256"  
← TLS_AES_128_GCM_SHA256"
```

上記はデフォルト設定です。利用可能な全オプションの一覧については、opensslパッケージのciphers(1)マニュアルページを参照してください。

さらに、クライアントで使用する暗号スイートを /etc/default/pveproxy で選択設定できます（デフォルトはクライアントと pveproxy の両方が利用可能なリストの最初の暗号スイートです）。

```
HONOR_CIPHER_ORDER=0
```

### 18.2.4 サポートされる TLS バージョン

安全でないSSLバージョン2および3は、pveproxyにおいて無条件に無効化されています。TLSバージョン1.1未満は、最近のOpenSSLバージョンではデフォルトで無効化されており、pveproxyもこれを順守します（/etc/ssl/openssl.cnfを参照）。

TLSバージョン1.2または1.3を無効化するには、/etc/default/pveproxyに以下を設定してください:

```
DISABLE_TLS_1_2=1
```

または、それぞれ:

```
DISABLE_TLS_1_3=1
```

---

#### 注記

特に理由がない限り、サポートする 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

/etc/default/pveproxy で DHPARAMS を設定することで、使用するディフィー・ヘルマン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設定することで定義できます。例：

```
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ます。

---

### 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た場合にのみ使用されます。

---

## 18.2.6 代替 HTTPS 証明書

使用する証明書を外部証明書または ACME 経由で取得した証明書に変更することができます。

pveproxy は /etc/pve/local/pveproxy-ssl.pem および /etc/pve/local/pveproxy-ssl.key を使用しますが存在する場合に使用し、存在しない場合は /etc/pve/local/pve-ssl.pem および /etc/pve/local/pve-ssl.key をフォールバックとして使用します。秘密鍵はパスフレーズを使用しない場合があります。

証明書秘密鍵の保存場所 /etc/pve/local/pveproxy-ssl.key は、  
/etc/default/pveproxy で TLS\_KEY\_FILE を設定することで可能です。例：

```
TLS_KEY_FILE="/secrets/pveproxy.key"
```

---

### 注意

付属のACME統合はこの設定を尊重しません。

---

詳細は、ドキュメントの「ホストシステム管理」の章を参照してください。

## 18.2.7 レスポンス圧縮

デフォルトでは、pveproxy はクライアントがサポートしている場合、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 で無効化できます。

```
COMPRESSION=0
```

## 18.2.8 実際のクライアント IP のログ記録

デフォルトでは、pveproxy はリクエストを送信したクライアントの IP アドレスをログに記録します。pveproxy の前にプロキシサーバーが存在する場合、プロキシの IP ではなくリクエストを発行したクライアントの IP をログに記録することが望ましい場合があります。

プロキシによって設定されたHTTPヘッダーをログ記録目的で処理するには、クライアントIPを取得するヘッダー名にPROXY\_REAL\_IP\_HEADERを設定します。例：

を、クライアントIPを取得するヘッダー名に設定します。例:

```
PROXY_REAL_IP_HEADER="X-Forwarded-For"
```

このヘッダーに渡された無効な値は無視されます。

デフォルトでは、すべての受信リクエストにおいてこのヘッダーの値をログに記録します。上記のHTTPヘッダーの設定を許可する信頼できるプロキシサーバーのリストを定義するには、`PROXY_REAL_IP_ALLOW_FROM`を設定します。例：

```
PROXY_REAL_IP_ALLOW_FROM="192.168.0.2"
```

`PROXY_REAL_IP_ALLOW_FROM`設定は、`ALLOW_FROM`および`DENY_FROM`設定と同様の値をサポートします。

IPアドレスはNet::IPが理解する任意の構文で指定できます。名前`all`は`0/0`および`::/0`（すべてのIPv4およびIPv6アドレスを意味）の別名です。

## 18.2.9 ワーカー数

`pveproxy`は着信リクエストの処理をワーカープロセスに委譲します。デフォルトでは3つのワーカープロセスを生成し、これはほとんどのワークロードで十分です。大量のAPIリクエストを発行する自動化中心のワークロードでリクエスト処理が遅延またはタイムアウトが発生する場合、`/etc/default/pveproxy`内の`MAX_WORKERS`を設定することでワーカープロセス数を増加できます。例：

```
MAX_WORKERS=5
```

ワーカープロセスの数を増やすとCPU使用率が上昇する可能性があることに注意してください。ワーカープロセスの数は0より大きく128未満である必要があります。

`pvedaemon`にも同様の設定が存在します。

## 18.3 pvestatd - Proxmox VE ステータステーモン

このデーモンは、一定間隔でVM、ストレージ、コンテナのステータスを問い合わせます。結果はクラスター内の全ノードに送信されます。

## 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あり、クライアントがリモートディスプレイやデバイス（例：キーボード、マウス、オーディオ）にアクセスできるようにします。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でリスンし、SPICEクライアントからのCONNECTリクエストを適切なProxmox VE VMに転送するHTTPプロキシを実装します。`www-data`ユーザーとして実行され、権限は非常に制限されています。

### 18.4.1 ホストベースのアクセス制御

`apache2`のようなアクセス制御リストの設定が可能です。設定値はファイル`/etc/default/pveprox`から読み込まれます。詳細は`pveproxy` のドキュメントを参照してください。

## 18.5 pvescheduler - Proxmox VE スケジューラーデーモン

このデーモンは、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する役割を担います。vzdump ジョブについては、その設定はファイル /etc/pve/jobs.cfg から取得します。

## 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 サブスクリプションの処理に使用されます。

## 19.2 pveperf - Proxmox VE ベンチマークスクリプト

PATH にマウントされたハードディスク（デフォルトは /）の CPU/ハードディスク性能データを収集します：

### CPU BOGOMIPS

全CPUのbogomips合計

### REGEX/SECOND

1秒あたりの正規表現処理数（Perlパフォーマンステスト）、300000以上が望ましい

### HD SIZE

ハードディスクサイズ

### バッファ読み取り

単純なHD読み取りテスト。現代のHDは少なくとも40MB/秒に達すべき

### 平均シーク時間

平均シーク時間をテストします。高速SCSIハードディスクは8ミリ秒未満の値を達成します。一般的なIDE/SATAディスクは15~20ミリ秒の値を示します。

### FSyncs/秒

値は200以上であるべきです（RAIDコントローラで書き込みキャッシングモードを有効にする必要があります - バッテリーバックアップ付きキャッシング（BBWC）が必要です）。

### DNS EXT

外部DNS名解決の平均所要時間

### DNS INT

ローカル DNS 名の解決にかかる平均時間

## 19.3 Proxmox VE API 用の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 サーバーを使用せずに API 機能を直接呼び出すことができます。

---

### 注

*root* のみがこれを実行できます。

---

### 19.3.1 例

クラスタ内のノード一覧を取得する

```
# pvsh get /nodes
```

データセンターの利用可能なオプション一覧を取得

```
# pvsh usage cluster/options -v
```

データセンターのデフォルトコンソールとしてHTML5 NoVNCコンソールを設定

```
# pvsh set cluster/options -console html5
```

## 第20章

### よくある質問

---

#### 注記

新しいFAQはこのセクションの末尾に追加されます。

---

#### 1. Proxmox VE はどのディストリビューションをベースにしていますか？

Proxmox VEはDebian GNU/Linuxをベースとしています

#### 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のコードは、GNU Affero General Public Licenseバージョン3の下でライセンスされています。

#### 3. Proxmox VEは32ビットプロセッサ上で動作しますか？

Proxmox VEは64ビットCPU (AMDまたはIntel) でのみ動作します。このプラットフォーム向けの32ビット版は計画されていません。

---

#### 注記

仮想マシンとコンテナは、32ビットと64ビットの両方で動作します。

---

#### 4. お使いの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、以下のコマンド出力でvmxまたはsvmタグがあるか確認してください:

```
egrep '(vmx|svm)' /proc/cpuinfo
```

#### 5. 対応インテルCPU

Intel Virtualization Technology (Intel VT-x) 対応の64ビットプロセッサ。 ([Intel VTおよび64ビット対応プロセッサー一覧](#))

#### 6. サポート対象のAMD CPU

AMD Virtualization Technology (AMD-V) をサポートする 64 ビットプロセッサ。

#### 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仮想化の概念を指します。オペレーティングシステムレベル仮想化とは、オペレーティングシステムのカーネルが複数の分離されたインスタンスを許可し、それらすべてがカーネルを共有する仮想化手法です。LXCに関しては、このようなインスタンスをコンテナと呼びます。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、オーバーヘッドが少ないですが、Linuxゲストに限定されます。

---

## 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）とは、QEMUとLinux KVMカーネルモジュールを使用してProxmox VE上で仮想化されるゲストシステムです。

## 9. QEMUとは何ですか？

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ソフトウェアです。QEMUはLinux KVMカーネルモジュールを利用し、ゲストコードをホストCPU上で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します。Linuxゲストに限定されず、任意のオペレーティングシステムの実行が可能です。

## 10. Proxmox VEのバージョンはどのくらいサポートされますか？

Proxmox VEのバージョンは、対応するDebianバージョンが~~oldstable~~である限り、少なくともサポートされます。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が推奨されます。

Proxmox VE バージョン	Debian バージョン	初回リリース	Debian EOL	Proxmox サポート終了
Proxmox VE 9	Debian 13 (Trixie)	2025-07?	未定	未定
Proxmox VE 8	Debian 12 (Bookworm)	2023-06	2026-08	2026-08
Proxmox VE 7	Debian 11 (Bullseye)	2021-07	2024年7月	2024-07
Proxmox VE 6	Debian 10 (buster)	2019-07	2022-09	2022-09
Proxmox VE 5	Debian 9 (Stretch)	2017-07	2020-07	2020年7月
Proxmox VE 4	Debian 8 (Jessie)	2015-10	2018-06	2018-06
Proxmox VE 3	Debian 7 (Wheezy)	2013-05	2016-04	2017年2月
Proxmox VE 2	Debian 6 (Squeeze)	2012-04	2014-05	2014-05
Proxmox VE 1	Debian 5 (Lenny)	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どうすればよいですか？

マイナーバージョンのアップグレード（例：Proxmox VE バージョン8.1から8.2または8.3へのアップグレード）は、通常の更新と同様に行えます。ただし、関連する重要な変更点や互換性を損なう変更がないか、[リリースノート](#)を確認する必要があります。

更新自体は、Web UI Node→ の 「Updates」 パネルを使用するか、CLI で以下のコマンドを実行します：

```
apt update
apt full-upgrade
```

### 注意

パッケージリポジトリを正しく設定していることを常に確認し、apt update でエラーが発生しなかった場合にのみ、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どうすればよいですか？

メジャーバージョンアップグレード（例：Proxmox VE 8.4 から 9.0 への移行）もサポートされています。これらは慎重に計画・テストが必要であり、最新のバックアップが準備されていない状態で開始してはいけません。

具体的なアップグレード手順は環境設定によって異なりますが、一般的な実施方法とアドバイスを提供します：

- [Proxmox VE 8 から 9 へのアップグレード](#)
- [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に対するユーザースペースインターフェースです。強力なAPIとシンプルなツールを通じて、Linuxユーザーがシステムコンテナを簡単に作成・管理できるようにします。LXCは、以前のOpenVZと同様に、**システム仮想化**を目的としています。したがって、コンテナ内で完全なOSを実行でき、sshでログインしたり、ユーザーを追加したり、apacheを実行したりすることができます。

LXDはLXCの上に構築され、新たな優れたユーザー体験を提供します。内部では、LXDはliblxcとそのGoバインディングを通じてLXCを利用し、コンテナの作成と管理を行います。これは基本的にLXCのツールと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あり、ネットワーク経由で制御可能であることから得られる追加機能を備えています。

Proxmoxコンテナとは、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します。これらは**システム仮想化**を目的としており、コンテナ基盤としてLXCを採用しています。Proxmox Container Toolkit (pct) はProxmox VEと緊密に連携しています。これは、クラスター構成を認識し、QEMU仮想マシン (VM) と同じネットワークおよびストレージリソースを利用できることを意味します。Proxmox VEファイアウォールの使用、バックアップの作成と復元、HAフレームワークを用いたコンテナ管理も可能です。Proxmox VE APIを介してネットワーク経由で全てを制御できます。

Dockerは、**単一のアプリケーションを分離された自己完結型環境で実行すること**を目的としています。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」と呼ばれます。Dockerインスタンスは、Docker Engineコマンドラインインターフェースを使用してホストから管理します。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。

---

### 注記

アプリケーションコンテナ（例：Dockerイメージ）を実行する場合は、Proxmox QEMU VM内で実行するのが最適です。

---

## 第21章 参考文献

### 21.1 Proxmox VEに関する書籍

- [1] [Ahmed16] Wasim Ahmed. 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] リック・ゴールドマン. 『Learning Proxmox VE』. パックト出版, 2016. ISBN 978-1783981786
- [5] [Surber16] Lee R. Surber. 『Virtualization Complete: 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 Bullseye入門から上級まで』, Freexian, 2021. ISBN 979-10-91414-20-3
- [7] [Bir96] Kenneth P. Birman. 『安全で信頼性の高いネットワークアプリケーションの構築』. 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] ヴォルフガング・マウラー. 『プロフェッショナル Linux カーネルアーキテクチャ』. ジョン・ワイリー・アンド・サンズ, 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. Version Control with Git: Powerful tools and techniques for collaborative software development. O'Reilly and Associates, 2012. ISBN 978-1449316389
- [15] [Kreibich10] Jay A. Kreibich. 『SQLiteの活用』, O'Reilly and Associates, 2010. ISBN 978-0596521189

## 21.3 関連トピックに関する書籍

- [16] [Bessen09] James Bessen & Michael J. Meurer, 『特許制度の失敗：裁判官・官僚・弁護士が革新者を危険に晒す仕組み』 プリンストン大学出版局, 2009年. ISBN 978-0691143217

## 付録A

# コマンドラインインターフェース

## A.1 概要

オプションの大文字小文字の表記スタイルが歴史的に統一されていない点については、[設定ファイルに関する関連セクション](#)を参照してください。

## A.2 出力形式オプション [`FORMAT_OPTIONS`]

`--output-format` パラメータを使用して出力形式を指定できます。デフォルトの形式テキストは、表の周囲に美しい枠線を描くために ASCII アートを使用します。さらに、一部の値を人間が読みやすいテキストに変換します。例：

- UnixエポックはISO 8601日付文字列として表示されます。
- 期間 (duration) は週/日/時間/分/秒のカウントで表示されます (例: 1d 5h)。
- バイトサイズの値には単位 (B、KiB、MiB、GiB、TiB、PiB) が含まれます。
- 小数部分はパーセンテージで表示されます。例えば 1.0 は 100% と表示されます。

オプション `--quiet` を使用すると出力を完全に抑制できます。

`--human-readable <boolean>` (デフォルト= 1)

出力レンダリング関数を呼び出し、人間が読みやすいテキストを生成します。

`--noborder <boolean>` (デフォルト= 0)

境界線を描画しない (テキスト形式の場合)。

`--noheader <boolean>` (デフォルト= 0)

列ヘッダーを表示しない (テキスト形式の場合)。

`--output-format <json| json-pretty| text| yaml>` (デフォルト= text)

出力形式。

`--quiet <boolean>`

結果の表示を抑制します。

## A.3 pvesm - Proxmox VE Storage Manager

**pvesm** <コマンド> [引数] [オプション]  
**pvesm add** <type> <storage> [OPTIONS]

新しいストレージを作成します。

<type>; <btrfs| cephfs| cifs| dir| esxi| iscsi| iscsidirect | lvm | lvmthin | nfs | pbs | rbd | zfs | zfspool>;  
ストレージタイプ。

<storage>; <storage ID>;  
ストレージ識別子。

--authsupported <string>;  
認証サポート。

--base <string>;  
ベースボリューム。このボリュームは自動的にアクティブになります。

--blocksize <文字列>;  
ブロックサイズ

--bwlimit [<clone>=LIMIT] [<default>=LIMIT] [<migration>=LIMIT] [<move>=LIMIT]  
[,<restore>=LIMIT]  
各種操作に対するI/O帯域幅制限を設定します（単位：KiB/s）。

--comstar\_hg <文字列>;  
comstarビュー用のホストグループ

--comstar\_tg <string>;  
comstar ビューのターゲットグループ

--content <string>;  
許可されるコンテンツタイプ。

---

### 注

コンテナには値 *rootdir* が、VM には値 *images* が使用されます。

---

--content-dirs <string>;  
デフォルトのコンテンツタイプディレクトリを上書きします。

--create-base-path <boolean> (デフォルト= は有効)  
ベースディレクトリが存在しない場合は作成します。

---

```
--create-subdirs <boolean> (デフォルト= yes)
ディレクトリにデフォルト構造を生成します。

--data-pool <string>;
データプール（エラー訂正符号化専用）

--datastore <string>;
Proxmox Backup Serverのデータストア名。

--disable <boolean>;
ストレージを無効化するフラグ。

--domain <string>;
CIFS ドメイン。

--encryption-key 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。autogenを使用するとパスフレーズなしで自動的に生成されます。

--export <string>;
NFSエクスポートパス。

証明書のSHA-256フィンガープリント。

--format <qcow2| raw| subvol| vmdk>;
デフォルトのイメージ形式。

--fs-name <string>;
Cephファイルシステム名。

--fuse <boolean>;
FUSE経由で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管理のマウントポイントであると見なし、マウントされていない場合、ストレージをオフラインと見なします。ブール値 (yes/no) を使用すると、このフィールドでターゲットパスを指定するショートカットとして機能します。

--iscsiprovider <string>;
iscsiプロバイダー

--keyring-file (Cephクラスターで認証するためのキーリングを含む)
クライアントキーリングの内容（外部クラスター用）。

--krbd <boolean> (デフォルト= 0)
krbdカーネルモジュールを介して常にrbdにアクセスする。
```

--lio\_tpg <string>  
Linux LIOターゲットの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形式の公開RSA鍵。暗号化キーのコピーを暗号化するために使用され、各暗号化バックアップに追加されます。

--max-protected-backups <integer> (-1 - N) (デフォルト= Datastore.Allocate権限を持つユーザーは無制限、他のユーザーは5)  
ゲストごとの保護対象バックアップの最大数。無制限の場合は-1を指定。

--mkdir <boolean> (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で埋めます。注記: 非推奨です。代わりに代わりに *create-base-path* および *create-subdirs* オプションを使用してください。

--monhost <string>  
モニターのIPアドレス (外部クラスター用)。

--mountpoint <string>  
マウントポイント

--namespace <string>  
ネームスペース。

--nocow <boolean> (デフォルト= 0)  
ファイルにNOCOWフラグを設定します。データチェックサムを無効化し、データエラーを回復不能にします。直接I/Oを許可します。基盤となるRAIDシステムのない単一のext4フォーマットディスクと同等の安全性で十分である場合にのみ使用してください。

--nodes <string>  
ストレージ構成が適用されるノードのリスト。

--nowritecache <boolean>  
対象ノードでの書き込みキャッシングを無効化

--options <string>  
NFS/CIFS マウントオプション (*man nfs* または *man mount.cifs* を参照)

--password <パスワード>  
共有/データストアへのアクセス用パスワード。

--path <文字列>  
ファイルシステムのパス。

--pool <文字列>  
プール。

```
--port &lt;integer&gt; (1 - 65535)
デフォルト以外のポートでストレージに接続する場合に使用します（例：PBSやESXiの場合）。NFSおよびCIFSでは、オプションオプションを使用してマウントオプション経由でポートを設定してください。
```

```
--portal &lt;string&gt;
iSCSIポータル（IPまたはDNS名、オプションでポートを指定）。
```

```
--preallocation &lt;fallback | full | metadata | off&gt; (デフォルト=metadata) raw および qcow2 イメージの事前割り当てモード。raw イメージで metadata を使用すると、preallocation=off となります。
```

```
--prune-backups [keep-all=&lt;1|0&gt;] [,keep-daily=&lt;N&gt;] [,keep-hourly=&lt;N&gt;] [,keep-last=&lt;N&gt;] [,keep-monthly=&lt;N&gt;] [,keep-weekly=&lt;N&gt;] [,keep-yearly=&lt;N&gt;]
保持期間が短いオプションが優先的に処理され、--keep-last が最優先となります。
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当該オプションの対象となります。次のオプションは既にカバーされたバックアップを扱わず、より古いバックアップのみを考慮します。
```

```
--saferemove &lt;boolean&gt;
LV削除時にデータをゼロクリアします。
```

```
--saferemove_throughput &lt;文字列&gt;
消去スループット（cstream -t パラメータ値）。
```

```
--server &lt;string&gt;
サーバーのIPアドレスまたはDNS名。
```

```
--share &lt;string&gt;
CIFS共有。
```

```
--shared &lt;boolean&gt;
すべてのノード（または nodes オプションで指定されたすべてのノード）で同じ内容を持つ単一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を他のノードに自動的にアクセス可能にするものではなく、既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 &lt;boolean&gt; (デフォルト= false)
TLS証明書検証を無効化します。完全に信頼できるネットワークでのみ有効にしてください！
```

```
--smbversion &lt;2.0| 2.1| 3| 3.0| 3.11| default&gt; (デフォルト= default)
SMBプロトコルバージョン。設定されていない場合、クライアントとサーバーの両方がサポートする最高レベルのSMB2+バージョンをネゴシエートします。
```

```
--snapshot-as-volume-chain &lt;boolean&gt; (デフォルト= 0)
ボリュームのバックアップチェーンを通じて、ストレージベンダーに依存しないスナップショットの作成をサポートします。
```

```
--sparse <boolean>;
    スパースボリュームを使用する

--subdir <string>;
    マウントするサブディレクトリ。

--tagged_only <boolean>;
    pve-vm-IDタグが付与された論理ボリュームのみを使用する。

--target <string>;
    iSCSIターゲット。

--thinpool <string>;
    LVMシンプルのLV名。

--username <string>;
    RBD ID。

--vgname <string>;
    ボリュームグループ名。

--zfs-base-path <string>;
    作成されたZFSブロックデバイスを検索するベースパス。指定しない場合、作成時に自動的に設定されます。通常は/dev/zvol です。

pvesm alloc <storage> <vmid> <filename> <size> [オプション]
ディスクイメージを割り当てます。

<storage>::<storage ID>;
    ストレージ識別子。

<vmid>::<整数> (100 - 999999999)
    所有者 VM を指定

<filename>::<string>;
    作成するファイルの名前。

<size>::\d+[MG]?
    キロバイト単位のサイズ (1024バイト)。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| vmdk>;
    イメージのフォーマット。
```

---

#### 注記

必要なオプション: size

---

**pvesm apiinfo**

APIVER および APIAGE を返します。

**pvesm cifsscan**

*pvesm scan cifs* の別名。

**pvesm export <ボリューム> <フォーマット> <ファイル名> [オプション]**

ボリュームをエクスポートするために内部で使用されます。

<volume>: <string>;

ボリューム識別子

<format>: <btrfs| qcow2+size| raw+size| tar+size| vmdk+size| zfs>;

エクスポートストリーム形式

<filename>: <string>;

出力ファイル名

--base (?^i:[a-zA-Z0-9\_\\-]{1,40})

増分ストリームを開始するスナップショット

エクスポート用スナップショット

--snapshot-list <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構成を抽出します。

<volume>: <string>;

ボリューム識別子

**pvesm free <volume> [オプション]**

ボリュームを削除します

<volume>: <string>;

ボリューム識別子

--delay <integer> (1 - 30)

タスク完了までの待機時間。指定時間内に完了した場合、*null*を返します。

--storage <ストレージID>;

ストレージ識別子。

**pvesm help** [オプション]

指定され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します

--verbose <boolean>;

詳細出力形式。

**pvesm import** <ボリューム>; <フォーマット>; <ファイル名>; [オプション]

ボリュームをインポートするために内部で使用されます。

<volume>; <string>;

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形式

<filename>; <string>;

ソースファイル名。- 標準入力を使用する場合、tcp://<IP-or-CIDR> 形式は TCP 接続を、unix://PATH-TO-SOCKET 形式は UNIX ソケットを入力として使用可能。それ以外の場合、ファイルは通常のファイルとして扱われる。

--allow-rename <boolean>; (デフォルト= 0)

要求されたボリュームIDが既に存在する場合、エラーを発生させる代わりに新しいボリュームIDを選択してください。

--base (?^i:[a-zA-Z0-9\_\-]{1,40})

増分ストリームのベーススナップショット

成功時に削除するスナップショット

ストリームにスナップショットが含まれる場合の現在の状態のスナップショット

--with-snapshots <boolean>; (デフォルト= 0)

ストリームに中間スナップショットを含めるかどうか

**pvesm iscsiscan**

*pvesm scan iscsi* の別名。

**pvesm list** <storage>; [OPTIONS]

ストレージの内容を一覧表示します。

```
&lt;storage&gt;:: &lt;storage ID&gt;  
ストレージ識別子。  
  
--content &lt;string&gt;  
このタイプのコンテンツのみをリストします。  
  
--vmid &lt;integer&gt; (100 - 999999999)  
このVMの画像のみをリスト表示
```

**pvesm lvmscan**

*pvesm scan lvm* の別名。

**pvesm lvmthinscan**

*pvesm scan lvmthin* の別名。

**pvesm nfsscan**

*pvesm scan nfs* の別名。

**pvesm path &lt;ボリューム&gt;**

指定されたボリュームのファイルシステムパスを取得します

```
&lt;volume&gt;:: &lt;string&gt;  
ボリューム識別子
```

**pvesm prune-backups &lt;storage&gt; [オプション]**

バックアップを整理します。標準命名規則を使用しているもののみが対象となります。保持オプションが指定されていない場合、ストレージ設定のオプションが使用されます。

```
&lt;storage&gt;:: &lt;storage ID&gt;  
ストレージ識別子。
```

```
--dry-run &lt;boolean&gt;  
削除対象のみを表示し、実際の削除は行わない。
```

```
--keep-all &lt;boolean&gt;  
すべてのバックアップを保持します。trueの場合、他のオプションと競合します。
```

```
--keep-daily &lt;N&gt;  
過去 &lt;N&gt; 日分の異なる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、最新の1つだけが保持されます。
```

```
--keep-hourly &lt;N&gt;  
過去 &lt;N&gt; 時間の異なる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、最新の1つだけが保持されます。
```

```
--keep-last &lt;N&gt;  
最後の &lt;N&gt; 個のバックアップを保持する。
```

```
--keep-monthly <N>;
過去 &lt;N&gt; ヶ月分の異なるバックアップを保持します。同一月に複数のバックアップがある場合、最新の1つだけが保持されます。
```

```
--keep-weekly <N>;
過去 &lt;N&gt; 週間の異なるバックアップを保持します。1週間にに対して複数のバックアップがある場合、最新の1つだけが保持されます。
```

```
--keep-yearly <N>;
過去 &lt;N&gt; 年分の異なるバックアップを保持します。1年につき複数のバックアップがある場合、最新の1つだけが保持されます。
```

```
--type <lxc| qemu>;
qemu または lxc のいずれか。このタイプのゲストのバックアップのみを考慮します。
```

```
--vmid <整数>; (100 - 99999999)
このゲストのバックアップのみを考慮します。
```

```
pvesm remove <storage>;
ストレージ構成を削除します。
```

```
<storage>:: <storage ID>;
ストレージ識別子。
```

```
pvesm scan cifs <サーバー>; [オプション]
リモート CIFS サーバーをスキャンします。
```

```
<server>:: <string>;
サーバーのアドレス（名前またはIP）。
```

```
--domain <string>;
SMB ドメイン（ワークグループ）。
```

```
--password <password>;
ユーザーpassword。
```

```
--username <string>;
ユーザー名。
```

```
pvesm scan iscsi <ポータル>;
リモート iSCSI サーバーをスキャンします。
```

```
<portal>:: <string>;
iSCSIポータル（IPまたはDNS名、オプションでポート）。
```

**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**

&lt;vg&gt; ローカル LVM シンプールを一

覧表示します。

説明はありません

**pvesm scan nfs &lt;server&gt;**

リモート NFS サーバーをスキャンします。

&lt;server&gt;; &lt;string&gt;

サーバーアドレス（名前またはIP）。

**pvesm scan pbs &lt;サーバー&gt; &lt;ユーザー名&gt; --password &lt;文字列&gt; [オプション] [フォーマットオプション]**

リモート Proxmox バックアップサーバーをスキャンします。

&lt;サーバー&gt;; &lt;文字列&gt;

サーバーのアドレス（名前または IP）。

&lt;username&gt;; &lt;string&gt;

ユーザー名または API トークン ID。

証明書のSHA256フィンガープリント。

--password &lt;文字列&gt;

ユーザーパスワードまたはAPIトークンシークレット。

--port &lt;整数&gt; (1 - 65535) (デフォルト= 8007)

オプションのポート。

**pvesm scan zfs**

ローカルノード上のzfsプールリストをスキャンします。

**pvesm set &lt;storage&gt; [OPTIONS]**

ストレージ構成を更新します。

&lt;storage&gt;; &lt;storage\_ID&gt;

ストレージ識別子。

--blocksize &lt;string&gt;

ブロックサイズ

--bwlimit [clone=<LIMIT>] [,default=<LIMIT>] [,migration=<LIMIT>] [,move=<LIMIT>]  
[,restore=<LIMIT>]

各種操作に対するI/O帯域幅制限を設定します（単位: KiB/s）。

--comstar\_hg <文字列>;

comstarビュー用のホストグループ

--comstar\_tg <string>;

comstar ビューのターゲットグループ

--content <string>;

許可されるコンテンツタイプ。

---

#### 注

コンテナには値 *rootdir* が、VM には値 *images* が使用されます。

---

--content-dirs <string>;

デフォルトのコンテンツタイプディレクトリを上書きします。

--create-base-path <boolean> (デフォルト= はい)

ベースディレクトリが存在しない場合は作成します。

--create-subdirs <boolean> (デフォルト= yes)

ディレクトリにデフォルト構造を生成します。

--data-pool <string>;

データプール（エラー訂正符号化専用）

--delete <string>;

削除したい設定のリスト。

--digest <string>;

現在の設定ファイルのダイジェストが異なる場合、変更を防止します。これにより同時変更を防止できます。

--disable <boolean>;

ストレージを無効化するフラグ。

--domain <string>;

CIFS ドメイン。

--encryption-key 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。*autogen*を使用すると、パスフレーズなしで自動的に生成されます。

---

証明書のSHA-256フィンガープリント。

--format &lt; qcow2 | raw | subvol | vmdk&gt;

デフォルトのイメージ形式。

--fs-name &lt; string&gt;

Cephファイルシステム名。

--fuse &lt; boolean&gt;

FUSE経由でCephFSをマウントします。

--is\_mountpoint &lt; string&gt; (デフォルト= no)

指定されたパスが外部管理の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がオフラインであると見なします。布尔値 ( yes/no) を使用することで、このフィールドでターゲットパスを指定するショートカットとして機能します。

--keyring Cephクラスタで認証するためのキーリングを含むファイル

クライアントキーリングの内容 (外部クラスター用)。

--krbd &lt; ブール値&gt; (デフォルト= 0)

krbdカーネルモジュールを介して常にrbdにアクセスする。

--lio\_tpg &lt; string&gt;

Linux LIOターゲットの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形式の公開RSA鍵。暗号化キーのコピーを暗号化す

るために使用され、各暗号化バックアップに追加されます。

--max-protected-backups &lt; integer&gt; (-1 - N) (デフォルト= 。Datastore.Allocate権限を持つユ

ーザーは無制限、その他のユーザーは5)

ゲストごとの保護対象バックアップの最大数。無制限の場合は-1を指定。

--mkdir &lt; boolean&gt; (デフォルト= yes)

ディレクトリが存在しない場合に作成し、デフォルトのサブディレクトリを生成します。注：非推奨。代わりに

create-base-path および create-subdirs オプションを使用してください。

--monhost &lt; string&gt;

監視対象のIPアドレス (外部クラスター用)。

--mountpoint &lt; string&gt;

マウントポイント

--namespace &lt; string&gt;

ネームスペース。

--nocow <boolean> (デフォルト= 0)

ファイルにNOCOWフラグを設定します。データチェックサムを無効化し、データエラーを回復不能になると同時に、直接I/Oを許可します。この設定は、基盤となるRAIDシステムのない単一のext4フォーマットディスクと同等の安全性で十分な場合にのみ使用してください。

--nodes <string>

ストレージ構成が適用されるノードのリスト。

--nowritecache <boolean>

対象ノードでの書き込みキャッシングを無効化

--options <string>

NFS/CIFS マウントオプション (*man nfs* または *man mount.cifs* を参照)

--password <password>

共有/データストアへのアクセス用パスワード

--pool <string>

プール。

--port <integer> (1 - 65535)

デフォルト以外のポートでストレージに接続する場合に使用します（例：PBSやESXi）。NFSおよびCIFSでは、オプションオプションを使用してマウントオプション経由でポートを設定します。

--preallocation <fallback | full | metadata | off> (デフォルト=metadata) raw および qcow2 イメージの事前割り当てモード。raw イメージで *metadata* を使用すると、*preallocation=off* となります。

--prune-backups [keep-all=<1|0>] [,keep-daily=<N>] [,keep-hourly=<N>] [,keep-last=<N>] [,keep-monthly=<N>] [,keep-weekly=<N>] [,keep-yearly=<N>]

保持期間が短いオプションが優先的に処理され、--keep-last が最優先となります。

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当該オプションの対象となります。次のオプションは既にカバーされたバックアップを扱わず、より古いバックアップのみを考慮します。

--saferemove <boolean>

LV削除時にデータをゼロクリアします。

--saferemove\_throughput <文字列>

消去スループット (cstream -t パラメータ値)。

--server <string>

サーバーのIPアドレスまたはDNS名。

--shared <boolean>

すべてのノード（または *nodes* オプションで列挙されたノードすべて）に同一の内容を持つ単一ストレージであることを示す。

オプションで列挙されたノードすべて) に同じ内容を持つ単一ストレージであることを示します。これにより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可能になるわけではなく、既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証明書検証を無効化してください。完全に信頼できるネットワークでのみ有効にしてください！

**--smbversion <2.0| 2.1| 3| 3.0| 3.11| default> (デフォルト= default)**

SMBプロトコルバージョン。設定されていない場合、クライアントとサーバーの両方がサポートする最高レベルのSMB2+バージョンをネゴシエートします。

**--snapshot-as-volume-chain <boolean> (デフォルト= 0)**

ボリュームのバックアップチェーンを通じて、ストレージベンダーに依存しないスナップショットの作成をサポートします。

**--sparse <boolean>**

スパースボリュームを使用する

**--subdir <文字列>**

マウントするサブディレクトリ。

**--tagged\_only <boolean>**

pve-vm-IDタグが付与された論理ボリュームのみを使用する。

**--username <string>**

RBD ID。

**--zfs-base-path <string>**

作成されたZFSブロックデバイスを検索するベースパス。指定しない場合、作成時に自動的に設定されます。通常は/dev/zvol です。

## pvesm status [オプション]

すべてのデータストアのステータスを取得します。

**--content <string>**

このコンテンツタイプをサポートするストアのみを一覧表示します。

**--enabled <boolean> (デフォルト= 0)**

設定で無効化されていない有効なストアのみを一覧表示します。

**--format <boolean> (デフォルト= 0)**

フォーマットに関する情報を含める

**--storage <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一覧表示

**--target <文字列>**

ターゲットがノードと異なる場合、このノードと指定されたターゲットノードの両方でコンテンツにアクセス可能な共有ストレージのみを一覧表示します。

**pvesm zfsscan**

*pvesm scan zfs* の別名。

## A.4 pvesubscription - Proxmox VE サブスクリプションマネージャー

**pvesubscription** <コマンド> [引数] [オプション]

**pvesubscription delete**

このノードのサブスクリプションキーを削除します。

**pvesubscription get**

サブスクリプション情報を読み取ります。

**pvesubscription help** [オプション] 指定したコマンド

に関するヘルプを取得します。

**--extra-args** <array>;

特定のコマンドのヘルプを表示します

**--verbose** <boolean>;

詳細出力形式。

**pvesubscription set** <key>;

サブスクリプションキーを設定します。

<key>:\s\* pve([1248])([nbsp])-([0-9a-f]{10})\s\*

Proxmox VE サブスクリプションキー

**pvesubscription set-offline-key** <data>;

内部使用のみ！オフラインキーを設定するには、代わりにパッケージ proxmox-offline-mirror-helper を使用してください。

<data>:<string>;

署名付きサブスクリプション情報プロト

**pvesubscription update** [オプション]

サブスクリプション情報を更新します。

**--force** <boolean> (デフォルト= 0)

ローカルキャッシュが有効な場合でも常にサーバーに接続する。

## A.5 pveperf - Proxmox VE ベンチマークスクリプト

**pveperf** [PATH]

## A.6 pveceph - Proxmox VE ノード上の CEPH サービスを管理

**pveceph** <コマンド> [引数] [オプション]

**pveceph createmgr**

*pveceph mgr create* の別名。

**pveceph createmon**

*pveceph mon create* の別名。

**pveceph createosd**

*pveceph osd create* の別名。

**pveceph createpool**

*pveceph pool create* の別名。

**pveceph destroymgr**

*pveceph mgr destroy* の別名。

**pveceph destroymon**

*pveceph mon destroy* の別名。

**pveceph destroyosd**

*pveceph osd destroy* の別名。

**pveceph destroypool**

*pveceph pool destroy* の別名。 **pveceph fs create**

[OPTIONS] Ceph ファイルシステムを作成しま

す。

**--add-storage** <boolean> (デフォルト= 0)

作成した CephFS をこのクラスタのストレージとして設定します。

**--name** (?^:|^:[^\s]+\$) (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用の配置グループ数。メタデータプールはこの4分の1を使用します。

**pveceph fs destroy** <name> [OPTIONS]

Ceph ファイルシステムを破棄する

<name>; <string>;

Ceph ファイルシステム名。

**--remove-pools** <boolean> (デフォルト= 0)

このファイルシステム用に構成されたデータプールとメタデータプールを削除します。

--remove-storages <boolean> (デフォルト= 0)

このファイルシステム用に設定されたすべてのpveceph管理ストレージを削除します。

#### pveceph help [オプション]

指定されたコマンドに関するヘルプを取得します。

--extra-args <array>;

特定のコマンドのヘルプを表示します

--verbose <boolean>;

詳細出力形式。

#### pveceph init [オプション]

初期の Ceph デフォルト設定を作成し、シンボリックリンクを設定します。

--cluster-network <string>;

別のクラスタネットワークを宣言します。OSDはハートビート、オブジェクトレプリケーション、リカバリトラフィックをこのネットワーク経由でルーティングします。

---

##### 注記

オプションが必要です: network

---

--disable\_cephx <boolean> (デフォルト= 0)

cephx認証を無効化します。



##### 警告

cephxは中間者攻撃を防ぐ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のみ、cephxの無効化を検討してください！

---

--min\_size <integer> (1 - 7) (デフォルト= 2)

オブジェクトごとにI/Oを許可するための利用可能なレプリカの最小数

--network <string>;

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

--pg\_bits <integer> (6 - 14) (デフォルト= 6)

配置グループビット。デフォルトの配置グループ数を指定するために使用されます。非推奨。この設定は最近のCephバージョン

で非推奨となりました。

--size <integer> (1 - 7) (デフォルト= 3)

オブジェクトごとの目標レプリカ数

---

**pveceph install [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 <布尔值> (デフォルト= 0)

実験的バージョンのインストールを許可します。注意して使用してください！

--リポジトリ <enterprise| no-subscription| test> (デフォルト= エンタープライズ)

使用するCephリポジトリ。

--version <squid> (デフォルト= squid)

インストールするCephのバージョン。

**pveceph ls pools**

pveceph pool ls の別名。pveceph mds create [オプ

ション] Ceph メタデータサーバー (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 MDS のログをポーリングおよびリプレイするかどうかを決定します。MDS 障害時の切り替えは高速化されますが、より多くのアイドルリソースが必要となります。

--name [<a-zA-Z0-9> ([<a-zA-Z0-9> -]\* [<a-zA-Z0-9>])? (デフォルト= ノード名)

MDSのID。省略時はノード名と同じ

pveceph mds destroy <name>;

Cephメタデータサーバーを破壊する

<name>; [<a-zA-Z0-9> ([<a-zA-Z0-9> -]\* [<a-zA-Z0-9>])?]

mds の名前 (ID)

**pveceph mgr create [オプション]**

Ceph マネージャーの作成

--id [<a-zA-Z0-9> ([<a-zA-Z0-9> -]\* [<a-zA-Z0-9>])?]

マネージャーのID。省略した場合、ノード名と同じ

pveceph mgr destroy <id>;

Ceph Manager を破棄します。

<id>; [<a-zA-Z0-9> ([<a-zA-Z0-9> -]\* [<a-zA-Z0-9>])?]

マネージャーの ID

**pveceph mon create** [オプション]

Ceph Monitor と Manager を作成

**--mon-address &lt;string&gt;**

自動検出されたモニター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である必要があります。

**--monid [a-zA-Z0-9] ([a-zA-Z0-9\-\-]\* [a-zA-Z0-9])?**

モニターのID。省略時はノード名と同じ

**pveceph mon destroy** &lt;monid&gt;

Cephモニターおよびマネージャーを破棄します。

**&lt;monid&gt;: [a-zA-Z0-9] ([a-zA-Z0-9\-\-]\* [a-zA-Z0-9])?**

Monitor ID

**pveceph osd create** &lt;dev&gt; [OPTIONS]

OSDを作成

**&lt;dev&gt;: &lt;string&gt;**

ブロックデバイス名。

**--crush-device-class &lt;文字列&gt;**

crushにおけるOSDのデバイスクラスを設定します。

**--db\_dev &lt;string&gt;**

block.dbのブロックデバイス名。

**--db\_dev\_size &lt;number&gt; (1 - N) (デフォルト= bluestore\_block\_db\_size または OSD サイズの 10%)**

block.dbのサイズ (GiB単位)。

---

#### 注記

必要なオプション: db\_dev

---

**--encrypted &lt;boolean&gt; (デフォルト= 0)**

OSDの暗号化を有効にします。

**--osds-per-device &lt;integer&gt; (1 - N)**

物理デバイスごとのOSDサービス数。高速NVMeデバイスでのみ有用であり、その性能をより効果的に活用します。

**--wal\_dev &lt;string&gt;**

block.wal用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のサイズ (GiB単位)。
```

---

**注記**

必要なオプション: wal\_dev

---

**pveceph osd destroy** <osdid> [オプション]

OSD を破棄します

```
<osdid>; <integer>;  
OSD ID
```

```
--cleanup <boolean>; (デフォルト= 0)
```

設定すると、パーティションテーブルエントリを削除します。

**pveceph osd details** <osdid> [OPTIONS] [FORMAT\_OPTIONS]

OSDの詳細を取得します。

```
<osdid>; <string>;  
OSDのID
```

```
--verbose <boolean>; (デフォルト= 0)
```

詳細情報を出力します。json-pretty出力形式と同様です。

**pveceph pool create** <name> [OPTIONS]

Cephプールを作成

```
<name>; (?^:^[\^:/\s]+$)
```

プールの名前。一意である必要があります。

```
--add_storages <boolean>; (デフォルト= 0; 誤差訂正コード化プールの場合: 1)
```

新しいプールを使用してVMとCTストレージを設定します。

```
--application <cephfs| rbd| rgw>; (デフォルト= rbd)
```

プールの適用先アプリケーション。

```
--crush_rule <string>;
```

クラスタ内のオブジェクト配置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>]
```

メタデータ保存用のレプリケートプールを伴う、RBD用のエラー訂正符号化プールを作成します。

ECを使用する場合、メタデータプールには共通のcephオプションであるsize、min\_size、crush\_rule/パラメータが適用されます。

```
--min_size <integer> (1 - 7) (デフォルト= 2)
```

オブジェクトごとの最小レプリカ数

```
--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 <integer> (1 - 32768) (デフォルト= 128)
```

配置グループの数。

```
--pg_num_min <integer> (-N - 32768)
```

最小配置グループ数。

```
--size <integer> (1 - 7) (デフォルト= 3)
```

オブジェクトごとのレプリカ数

```
--target_size ^(\d+(\.\d+)?)([KMG])?
```

PGオースケーラーのプールに対する推定ターゲットサイズ。

```
--target_size_ratio <number>;
```

PGオースケーラーのプールに対する推定目標比率。

```
pveceph pool destroy <name> [OPTIONS]
```

プールを削除します

```
<name>::<string>;
```

プールの名前。一意である必要があります。

```
--force <boolean> (デフォルト= 0)
```

trueの場合、使用中であってもプールを破壊します

```
--remove_ecprofile <boolean> (デフォルト= 0)
```

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たすべての pveceph 管理ストレージを削除します

```
pveceph pool get <name> [OPTIONS] [FORMAT_OPTIONS]
```

現在のプール状態を表示します。

**&lt;name&gt;:: &lt;string&gt;**

プールの名前。一意である必要があります。

**--verbose &lt;boolean&gt; (デフォルト= 0)**

有効にすると、追加データ（例：統計情報）を表示します。

**pveceph pool ls [ フォーマットオプション ]**

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 &lt;name&gt; [ OPTIONS ]**

POOLの設定を変更します

**&lt;name&gt;:: (?^:^[^:/\s]+\$)**

プールの名前。一意である必要があります。

**--application &lt;cephfs| rbd| rgw&gt;**

プールのアプリケーション。

**--crush\_rule &lt;string&gt;**

クラスタ内のオブジェクト配置をマッピングする際に使用するルール。

**--min\_size &lt;integer&gt; (1 - 7)**

オブジェクトごとの最小レプリカ数

**--pg\_autoscale\_mode &lt;off| on| warn&gt;**

プールの自動PGスケーリングモード。

**--pg\_num &lt;整数&gt; (1 - 32768)**

配置グループの数。

**--pg\_num\_min &lt;整数&gt; (-N ~ 32768)**

配置グループの最小数。

**--size &lt;integer&gt; (1 - 7)**

オブジェクトごとのレプリカ数

**--target\_size ^(\d+(\.\d+)? ([KMGT])?)?**

PGオートスケーラーのプールに対する推定ターゲットサイズ。

**--target\_size\_ratio &lt;number&gt;**

PGオートスケーラーのプールに対する推定ターゲット比率。

**pveceph purge [ オプション ]**

CEPH関連のデータおよび設定ファイルを削除します。

```
--crash &lt;boolean&gt;  
追加でCephクラッシュログ (/var/lib/ceph/crash) を削除します。
```

```
--logs &lt;boolean&gt;  
Cephログ (/var/log/ceph) を追加で削除します。
```

#### **pveceph start [オプション]**

Cephサービスを起動します。

```
--service
```

```
(デフォルト= ceph.target)  
Ceph サービス名。
```

#### **pveceph status**

Ceph のステータスを取得します。

#### **pveceph stop [オプション]**

Cephサービスを停止します。

```
--service
```

```
(デフォルト= ceph.target)  
Ceph サービス名。
```

## A.7 pvenode - Proxmox VE ノード管理

### **pvenode &lt;コマンド&gt; [引数] [オプション]**

```
pvenode acme account deactivate [&lt;name&gt;]
```

CA で既存の ACME アカウントを無効化します。

```
&lt;name&gt;; &lt;name&gt; (デフォルト=)  
ACMEアカウント設定ファイル名。
```

```
pvenode acme account info [&lt;name&gt;] [FORMAT_OPTIONS]
```

既存の ACME アカウント情報を返します。

```
&lt;name&gt;; &lt;name&gt; (デフォルト= default)  
ACMEアカウント設定ファイル名。
```

**pvenode acme アカウント一覧**

ACMEアカウントインデックス。

**pvenode acme account register [<name>] [<contact>] [OPTIONS]**

互換性のあるCAに新しいACMEアカウントを登録します。

**<name>; <name> (デフォルト= デフォルト)**

ACMEアカウント設定ファイル名。

**<contact>; <string>**

連絡先メールアドレス。

**--directory ^https?://.\***

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に更新します。注記: 新しいアカウント情報を指定しない場合、リフレッシュがトリガーされます。

**<name>; <name> (デフォルト= デフォルト)**

ACMEアカウント設定ファイル名。

**--contact <string>**

連絡先メールアドレス。

**pvenode acme cert order [オプション]**

ACME互換CAから新規証明書を注文します。

**--force <boolean> (デフォルト= 0)**

既存のカスタム証明書を上書きします。

**pvenode acme cert renew [オプション]**

CAから既存の証明書を更新します。

**--force <boolean> (デフォルト= 0)**

有効期限まで30日以上ある場合でも強制的に更新します。

**pvenode acme cert revoke**

CAから既存の証明書を失効します。

**pvenode acme plugin add <type> <id> [オプション]**

ACMEプラグインの設定を追加します。

```
&lt;type&gt;: &lt;dns| standalone&gt;
ACMEチャレンジタイプ。

&lt;id&gt;: &lt;string&gt;
ACMEプラグインID名

--api &lt;1984hosting | acmedns | acmeproxy | active24 | ad | ali | alviy | anx | artfiles |
arvan | aurora | autodns | aws | azion | azure | beget | bookmyname | bunny | cf | clouddns |
cloudns | cn | conoha | constellix | cpanel | curanet | cyon | da | ddns | desec
| df | dgon | dnsexit | dnshome | dnsimple | dnsservices | doapi | domeneshop | dp | dpi |
dreamhost | duckdns | durabledns | dyn | dynu | dynv6 | easydns | edgecenter | edgedns | euserv |
exoscale | fornex | freedns | freemyip | gandi_livedns | gcloud | gコア | gd | geoscaling |
googledomains | he | he_ddns | hetzner | hexonet | hostingde | huaweicloud | infoblox |
infomaniak | internetbs | inwx
| ionos | ionos_cloud | ipv64 | ispconfig | jdl | joker | kappernet
| kas | kinghost | knot | la | leaseweb | lexicon | limacity | linode | linode_v4 | loopia | lua |
maradns | me | miab | mijnhost
| misaka | myapi | mydevil | mydnsjp | mythic_beasts | namecheap | namecom | namesilo | nanelo |
nederhost | neodigit | netcup | netlify | nic | njalla | nm | nsd | nsone | nsupdate | nw |
oci | omglol | one | online | openprovider | openstack | opnsense | ovh | pdns | pleskxml | pointhq |
porkbun | rackcorp | rackspace | rage4
| rcode0 | regru | scaleway | schlundtech | selectel | selfhost | servercow | simply |
technitium | tele3 | tencent | timeweb | transip | udr | ultra | unoEuro | variomedia | veeesp |
vercel | vscale | vultr | websupport | west_cn | world4you | yandex360 | yc
| zilore | zone | zoneedit | zonomi&gt;

APIプラグイン名

--data 1行に1つのキーと値のペアを含むファイル。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プラグインデータ (base64エンコード済み)

--disable &lt;boolean&gt;
設定を無効化するフラグ。

--nodes &lt;string&gt;
クラスタノード名のリスト。

--validation-delay &lt;integer&gt; (0 - 172800) (デフォルト= 30)
検証要求前に待機する追加遅延 (秒単位)。DNSレコードの長いTTLに対応可能。

pvenode acme plugin config &lt;id&gt; [FORMAT_OPTIONS]
```

&lt;id&gt;: &lt;string&gt;  
ACMEプラグインインスタンスの一意の識別子。

**pvenode acme plugin list [OPTIONS] [FORMAT\_OPTIONS]**

ACMEプラグインインデックス。

--type &lt;dns| standalone&gt;  
特定のタイプのACMEプラグインのみを一覧表示

**pvenode acme plugin remove &lt;id&gt;**

ACMEプラグインの設定を削除します。

&lt;id&gt;: &lt;string&gt;  
ACMEプラグインインスタンスの一意の識別子。

**pvenode acme plugin set &lt;id&gt; [オプション]**

ACMEプラグインの設定を更新します。

&lt;id&gt;: &lt;string&gt;  
ACMEプラグインID名

--api &lt;1984hosting | acmedns | acmeproxy | active24 | ad | ali | alviy | anx | artfiles | arvan | aurora | autodns | aws | azion | azure | beget | bookmyname | bunny | cf | clouddns | cloudns | cn | conoha | constellix | cpanel | curanet | cyon | da | ddns | desec | df | dgon | dnsexit | dnshome | dnsimple | dnsservices | doapi | domeneshop | dp | dpi | dreamhost | duckdns | durabledns | dyn | dynu | dynv6 | easydns | edgecenter | edgedns | euserv | exoscale | fornex | freedns | freemyip | gandi\_livedns | gcloud | gコア | gd | geoscaling | googledomains | he | he\_ddns | hetzner | hexonet | hostingde | huaweicloud | infoblox | infomaniak | internetbs | inwx | ionos | ionos\_cloud | ipv64 | ispconfig | jd | joker | kappernet | kas | kinghost | knot | la | leaseweb | lexicon | limacity | linode | linode\_v4 | loopia | lua | maradns | me | miab | mijnhost | misaka | myapi | mydevil | mydnsjp | mythic\_beasts | namecheap | namecom | namesilo | nanelo | nederhost | neodigit | netcup | netlify | nic | njalla | nm | nsd | nsone | nsupdate | nw | oci | omglol | one | online | openprovider | openstack | opnsense | ovh | pdns | pleskxml | pointhq | porkbun | rackcorp | rackspace | rage4 | rcode0 | regru | scaleway | schlundtech | selectel | selfhost | servercow | simply | technitium | tele3 | tencent | timeweb | transip | udr | ultra | uno euro | variomedia | veesp | vercel | vscale | vultr | websupport | west\_cn | world4you | yandex360 | yc | zilore | zone | zoneedit | zonomi&gt;  
APIプラグイン名

--data 1行に1つのキーと値のペアを含むファイル。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プラグインデータ (base64エンコード済み)

--delete &lt;文字列&gt;

削除したい設定のリスト。

--digest &lt;string&gt;

現在の設定ファイルのダイジェストが異なる場合、変更を防止します。これにより同時編集を防止できます。

--disable &lt;boolean&gt;

設定を無効化するフラグ。

--nodes &lt;string&gt;

クラスタノード名のリスト。

--validation-delay &lt;integer&gt; (0 - 172800) (デフォルト= 30)

検証要求前に待機する追加遅延時間（秒単位）。DNSレコードの長いTTLに対応可能。

**pvenode cert delete [&lt;restart&gt;]**

カスタム証明書チェーンと鍵を削除します。

&lt;restart&gt;:: &lt;boolean&gt; (デフォルト= 0)

pveproxyを再起動します。

**pvenode cert info [ フォーマットオプション ]**

ノードの証明書に関する情報を取得します。

**pvenode cert set [&lt;certificates&gt; [&lt;key&gt;] [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。

&lt;certificates&gt;: &lt;string&gt;

PEMエンコードされた証明書（チェーン）。

&lt;key&gt;: &lt;string&gt;

PEMエンコードされた秘密鍵。

--force &lt;boolean&gt; (デフォルト= 0)

既存のカスタム証明書またはACME証明書ファイルを上書きします。

--restart &lt;boolean&gt; (デフォルト= 0)

pveproxyを再起動します。

**pvenode config get [オプション]**

ノード設定オプションを取得します。

```
--property &lt;acme | acmedomain0 | acmedomain1 | acmedomain2 | acmedomain3| acmedomain4|  
acmedomain5| ballooning-target | description | startall-onboot-delay | wakeonlan&gt; (デフォルト=all)
```

ノード設定から特定のプロパティのみを返す。

**pvenode config set [オプション]**

ノード設定オプションを設定します。

```
--acme [account=&lt;name&gt;] [,domains=&lt;domain[,domain;...]&gt;]  
ノード固有のACME設定。
```

```
--acmedomain[n] [domain=] &lt;domain&gt; [,alias=&lt;domain&gt;] [,plugin=&lt;plugin&gt;]  
ACMEドメインと検証プラグイン
```

```
--ballooning-target &lt;integer&gt; (0 - 100) (デフォルト= 80)  
バルーニングのRAM使用量目標（総メモリのパーセント）
```

```
--delete &lt;string&gt;  
削除したい設定のリスト。
```

```
--description &lt;string&gt;  
ノードの説明。Web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。設定ファイル内にコメントとして保存されます。
```

```
--digest &lt;string&gt;  
現在の設定ファイルのSHA1ダイジェストが異なる場合、変更を防止します。同時編集を防ぐために使用できます。
```

```
--startall-onboot-delay &lt;integer&gt; (0 - 300) (デフォルト= 0)  
起動時に有効なすべての仮想ゲストを起動するまでの初期遅延時間（秒単位）。
```

```
--wakeonlan [mac=] &lt;MACアドレス&gt; [,bind-interface=&lt;バインドインターフェース&gt;]  
[,broadcast-address=&lt;IPv4ブロードキャストアドレス&gt;]  
ノード固有のWake on LAN設定。
```

**pvenode help [オプション]**

指定したコマンドに関するヘルプを取得します。

```
--extra-args &lt;array&gt;  
特定のコマンドのヘルプを表示します
```

--verbose <boolean>

詳細出力形式。

**pvenode migrateall** <target> [OPTIONS]

すべての仮想マシンとコンテナを移行します。

<target>; <string>;

移行先のノード。

--maxworkers <integer> (1 - N)

並列移行ジョブの最大数。設定しない場合、datacenter.cfg の 'max\_workers' を使用します。いずれか一方を設定する必要があります！

--vms <string>;

これらのIDを持つゲストのみを考慮します。

--with-local-disks <boolean>;

ローカルディスクに対するライブストレージ移行を有効化

**pvenode startall** [オプション]

このノード上にあるすべての仮想マシンとコンテナを起動します（デフォルトではonboot=1のもののみ）。

--force <boolean> (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起動コマンドを発行します。

--vms <string>;

このカンマ区切りのVMIDリストに含まれるゲストのみを対象とする。

**pvenode stopall** [オプション]

すべての仮想マシンとコンテナを停止します。

--force-stop <boolean> (デフォルト= 1)

タイムアウト後に強制的にハードストップを実行します。

--timeout <integer> (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。強制停止の有無に応じて、シャットダウンは単純に中止されるか、ハードストップが強制されます

。

--vms <string>;

指定されたIDを持つゲストのみを考慮します。

**pvenode タスクリスト** [オプション] [フォーマットオプション]

1ノードのタスクリスト（完了済みタスク）を読み込みます。

```
--errors <boolean> (デフォルト= 0)
ERRORステータスのタスクのみをリスト表示します。

--limit <整数> (0 - N) (デフォルト= 50)
この数のタスクのみをリスト表示します。

--since <integer>;
このUNIXエポック以降のタスクのみを表示します。

--source <active| all| archive> (デフォルト=archive)
アーカイブ済み、アクティブ、または全タスクを一覧表示します。

--start <整数> (0 - N) (デフォルト= 0)
指定したオフセットから始まるタスクを一覧表示します。

--statusfilter <string>;
返されるべきタスク状態のリスト。

--typefilter <文字列>;
このタイプのタスクのみをリストします（例：vzstart、vzdump）。

--until <integer>;
このUNIXエポックまでのタスクのみをリスト表示します。

--userfilter <string>;
このユーザーからのタスクのみを一覧表示します。

--vmid <整数> (100 - 999999999)
このVMのタスクのみを一覧表示します。
```

**pvenode タスクログ <upid> [オプション]**

タスクログを読み取ります。

<upid>; <string>;
タスクの一意のID。

--download <boolean>;
タスクリ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できません

--start <integer> (0 - N) (デフォルト= 0)
タスクリグを読み込む際の開始行

**pvenode タスクステータス <upid> [フォーマットオプション]**

タスクのステータスを読み取る。

```
&lt;upid&ampgt:: &lt;string&ampgt;  
タスクの一意のID。
```

```
pvenode wakeonlan &lt;node&ampgt;  
Wake on LAN ネットワークパケットでノードを起動を試みます。
```

```
&lt;node&ampgt:: &lt;string&ampgt;  
Wake on LAN パケットのターゲットノード
```

## A.8 pvesh - Proxmox VE API 用のシェルインターフェース

```
pvesh &lt;コマンド&ampgt [引数] [オプション]  
pvesh create &lt;api_path&ampgt [OPTIONS] [FORMAT_OPTIONS]  
&lt;api_path&ampgt で API POST を呼び出します。
```

```
&lt;api_path&ampgt:: &lt;string&ampgt;  
API/パス。
```

```
--noproxy &lt;boolean&ampgt;  
自動プロキシ機能を無効化します。
```

```
pvesh delete &lt;api_path&ampgt [OPTIONS] [FORMAT_OPTIONS]  
&lt;api_path&ampgt で API DELETE を呼び出します。
```

```
&lt;api_path&ampgt:: &lt;string&ampgt;  
API/パス。
```

```
--noproxy &lt;boolean&ampgt;  
自動プロキシ機能を無効化します。
```

```
pvesh get &lt;api_path&ampgt [OPTIONS] [FORMAT_OPTIONS]  
&lt;api_path&ampgt で API GET を呼び出します。
```

```
&lt;api_path&ampgt:: &lt;string&ampgt;  
API/パス。
```

```
--noproxy &lt;boolean&ampgt;  
自動プロキシ機能を無効化します。
```

```
pvesh help [オプション]  
指定され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します

--verbose <boolean>  
詳細出力形式。

**pvesh ls** <api\_path> [オプション] [フォーマットオプション]  
<api\_path> 上の子オブジェクトを一覧表示します。

<api\_path>: <string>  
APIパス。

--noproxy <boolean>  
自動プロキシ機能を無効化します。

**pvesh set** <api\_path> [OPTIONS] [FORMAT\_OPTIONS]  
<api\_path> で API PUT を呼び出します。

<api\_path>: <string>  
APIパス。

--noproxy <boolean>  
自動プロキシ機能を無効化します。

**pvesh usage** <api\_path> [OPTIONS]  
<api\_path> の API 使用方法を表示します。

<api\_path>: <string>  
API パス。

--command <create| delete| get| set>  
API コマンド。

--boolean型を返す  
返されるデータのスキーマを含む。

--verbose <boolean>  
詳細出力形式。

## A.9 qm - QEMU/KVM 仮想マシン マネージャー

**qm** <コマンド> [引数] [オプション]

**qm agent**

*qm guest* コマンドの別名。

**qm cleanup** <vmid>: <clean-shutdown>: <guest-requested>;

tapデバイス、vgpuなどのリソースをクリーンアップします。VMのシャットダウン後やクラッシュ後に呼び出されます。

<vmid>:: <integer> (100 - 999999999)

VMの（一意の）ID。

<clean-shutdown>: <boolean>;

qemuが正常にシャットダウンしたかどうかを示す。

<guest-requested>: <boolean>;

シャットダウンがゲストによって要求されたか、qmp経由で要求されたかを示します。

**qm clone** <vmid> <newid> [OPTIONS]

仮想マシン/テンプレートのコピーを作成します。

<vmid>:: <integer> (100 - 999999999)

仮想マシンの（一意の）ID。

<newid>: <integer> (100 - 999999999)

クローン用の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 帯域幅制限 (KiB/s単位) を上書きします。

--description <string>;

新しい仮想マシン (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。完全クローンでのみ有効。

--full <boolean>;

すべてのディスクの完全なコピーを作成します。通常の VM をクローンする場合、常にこの操作が行われます。VM テンプレートの場合、デフォルトではリンククローンを作成しようとします。

--name <string>;

新しい VM の名前を設定します。

```
--pool &lt;string&gt;
新しい VM を指定されたプールに追加します。

--snapname &lt;string&gt;
スナップショットの名前。

--storage &lt;storage ID&gt;
完全クローン用のターゲットストレージ。

--target &lt;string&gt;
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ます。

qm cloudinit dump &lt;vmid&gt; &lt;type&gt;;
自動生成されたcloudinit設定を取得します。

&lt;vmid&gt;; &lt;integer&gt; (100 - 999999999)
VMの（一意の）ID。

&lt;type&gt;; &lt;meta| network| user&gt;;
設定タイプ。

qm cloudinit pending &lt;vmid&gt;;
現在の値と保留中の値の両方を含む cloudinit 設定を取得します。

&lt;vmid&gt;; &lt;integer&gt; (100 - 999999999)
VMの（一意の）ID。

qm cloudinit update &lt;vmid&gt;;
cloudinit設定ドライブを再生成および変更します。

&lt;vmid&gt;; &lt;integer&gt; (100 - 999999999)
VMの（一意の）ID。

qm config &lt;vmid&gt; [オプション]
保留中の設定変更を適用した仮想マシンの設定を取得します。代わりに現在の設定を取得するには、現在のパラメータを設定してください。

&lt;vmid&gt;; &lt;integer&gt; (100 - 999999999)
仮想マシンの（一意の）ID。

--current &lt;boolean&gt; (デフォルト= 0)
保留中の値ではなく現在の値を取得します。
```

**--snapshot <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]**

仮想マシンを作成または復元します。

**<vmid>: <integer> (100 - 999999999)**

仮想マシンの（一意の）ID。

**--acpi <boolean> (デフォルト= 1)**

ACPIを有効/無効にします。

**--affinity <文字列>;**

ゲストプロセスを実行するために使用す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の通信およびそのプロパティの有効化/無効化。

**--amd-sev [type=<sev-type> [,allow-smt=<1|0>]]**

AMD CPUによるセキュア暗号化仮想化 (SEV) 機能

**--arch <aarch64| x86\_64>;**

仮想プロセッサーアーキテクチャ。デフォルトはホスト。

**--archive <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パス (- を使用して標準入力からデータをパイプ) または Proxmox ストレージ バックアップボリューム識別子。

**--args <string>;**

KVMに渡す任意の引数。

**--audio0 device=<ich9-intel-hda|intel-hda|AC97> [,driver=<spice|none>]**

オーディオデバイスを設定します。QXL/Spiceとの組み合わせで有用です。

**--autostart <boolean> (デフォルト= 0)**

クラッシュ後の自動再起動（現在は無視される）。

**--balloon <integer> (0 - N)**

仮想マシンが使用するターゲットRAMの量 (MiB単位)。0を指定するとバルーンドライバが無効化される。

**--bios <ovmf| seabios> (デフォルト= seabios)**

BIOS実装を選択してください。

ゲストブート順序を指定します。order=サブプロパティを使用してください。キーなしありはlegacy=の使用は非推奨です。

--bootdisk (`ide|sata|scsi|virtio`)\d+

指定したディスクからの起動を有効化。非推奨: 代わりに `boot: order=foo;bar` を使用してください。

--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復元制限)

I/O 帯域幅制限 (KiB/s単位) を上書きします。

--cdrom <ボリューム>;

オプション `-ide2` の別名です

--cicustom [`meta=<ボリューム>;`] [`,network=<ボリューム>;`] [`,user=<ボリューム>;`] [`,vendor=<ボリューム>;`]

cloud-init: 起動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します。

--cipassword <password>;

cloud-init: ユーザーに割り当てるパスワード。通常は使用を推奨しません。代わりにSSHキーを使用してください。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点に注意してください。

--ciptype <configdrive2| nocloud| opennebula>;

cloud-init設定フォーマットを指定します。デフォルトは設定されたOSタイプ (`ostype`) に依存します。Linuxでは`nocloud`フォーマット、Windowsでは`configdrive2`フォーマットを使用します。

--ciupgrade <boolean>; (デフォルト= 1)

cloud-init: 初回起動後に自動パッケージアップグレードを実行します。

--ciuser <文字列>;

cloud-init: イメージで設定されたデフォルトユーザーではなく、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...]>;`]  
[`,hidden=<1|0>;`] [`,hv-vendor-id=<vendor-id>;`]

[`,物理ビット数=<8-64>;` ホスト] [`,報告モデル=<列挙型>;`]

エミュレートされるCPUタイプ。

--cpulimit <number>; (0 - 128) (デフォルト= 0)

CPU使用率の制限。

--cpuunits <integer>; (1 - 262144) (デフォルト= cgroup v1: 1024, cgroup v2: 100)

VM の CPU ウエイト。cgroup v2 では [1, 10000] に制限されます。

**--description &lt;string&gt;**

VMの説明。WebインターフェースのVM概要に表示されます。設定ファイル内にコメントとして保存されます。

**--efidisk0 [file=&lt;ボリューム&gt; [,efitype=&lt;2m|4m&gt;] [,format=&lt;enum&gt;]****[,import-from=&lt;ソースボリューム&gt;] [,pre-enrolled-keys=&lt;1|0&gt;] [,size=&lt;ディスクサイズ&gt;]**

EFI変数を保存するディスクを設定します。特別な構文 STORAGE\_ID:SIZE\_IN\_GiB を使用して

新しいボリュームを割り当てます。SIZE\_IN\_GiBはここでは無視され、代わりにデフォルトのEFI変数がボリュームにコピーされることに注意してください。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 &lt;boolean&gt;**

既存のVMを上書きすることを許可します。

---

**注記**

必要なオプション: archive

---

**--freeze &lt;boolean&gt;**

起動時にCPUを凍結 (c monitorコマンドで実行を開始)。

**--hookscript &lt;文字列&gt;**

VMSのライフサイクルにおける様々な段階で実行されるスクリプト。

**--hostpci[n] [[host=&lt;HOSTPCIID[,HOSTPCIID2...]&gt;] [,device-id=&lt;hex id&gt;] [,legacy-igd=&lt;1|0&gt;] [,マッピング=&lt;マッピング ID&gt;] [,mddev=&lt;文字列&gt;] [,pcie=&lt;1|0&gt;]****[,rombar=&lt;1|0&gt;] [,romfile=&lt;文字列&gt;]****[,サブデバイスID=&lt;16進ID&gt;] [,サブベンダーID=&lt;16進ID&gt;] [,ベンダー****ID=&lt;16進ID&gt;] [,x-vga=&lt;1|0&gt;]**

ホストPCIデバイスをゲストにマッピングします。

**--hotplug &lt;string&gt; (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化します。これはホットプラグ機能のカンマ区切りリストです: *network, disk, cpu, memory, usb, cloudinit*。0を指定するとホットプラグを完全に無効化します。値として1を使用すると、デフォルトの*network, disk, usb*の別名となります。USBホットプラグは、マシンバージョン >= 7.1かつostype l26 または Windows> 7.

**--hugepages &lt;1024| 2| any&gt;**

巨大ページメモリの有効化/無効化。

```
--ide[n] [file=<ボリューム>; ,aio=<ネイティブ|スレッド|io_uring>;]
[,backup=<1>; ,bps=<bps>; [,bps_max_length=<秒>;]
[,bps_rd=<bps>;][,bps_rd_max_length=<seconds>; [,bps_wr=<bps>;]
[,bps_wr_max_length=<seconds>; [,cache=<enum>; [,detect_zeroes=<1>; [,破
棄=<無視|有効>; [,フォーマット=<列挙型>; [,インポート元=<ソースボリューム>; [,I/O操
作数=<I/O操作数>; [,最大I/O操作数=<I/O操作数>; [,最大I/O操作数持続時間=<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>; [,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>; [,rerror=<ignore|report|stop>;]
[,serial=<serial>;][,共有=<1>; [,サイズ=<DiskSize>; [,スナップショット
=<1>; [,SSD=<1>; [,書き込みエラー=<enum>; [,WWN=<wwn>;]

ボリュームをIDEハードディスクまたはCD-ROMとして使用（nは0から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。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。
```

```
--import-working-storage <storage ID>;
イメージコンテンツタイプが有効化されたファイルベースストレージ。インポート時の中間抽出ストレージとして使用される。デフォルトはソースストレージ。
```

```
--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ですが、同じタイプのIPアドレスを指定する必要があります。

IPアドレスにDHCPを使用するには特殊文字列「dhcp」を使用でき、この場合明示的なゲートウェイは指定しないこと。IPv6では特殊文字列「auto」を
使用してステートレス自動設定を利用可能。これにはcloud-init 19.4以降が必要。

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を使用します。
```

```
--ivshmem size=<integer>; [,name=<string>;]
仮想マシン間共有メモリ。仮想マシン間、またはホストとの直接通信に有用です。
```

```
--keephugepages <boolean>; (デフォルト= 0)
hugepagesと併用します。有効にすると、VMシャットダウン後もhugepagesが削除されず、以降の起動で使用可能になります。
```

```
--keyboard <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。このオプションは通常不要であり、ゲストOS内で設定する方が望ましい場合が多い。
```

```
--kvm <boolean> (デフォルト= 1)
KVMハードウェア仮想化を有効/無効にします。

--live-restore <boolean>;
インポートまたは復元をバックグラウンドで実行しながら、VMを直ちに起動します。

--localtime <布尔值>;
リアルタイムクロック (RTC) を現地時間に設定します。 ostypeがMicrosoft Windows OSを示す場合、これはデフォルトで有効です。
Microsoft Windows OS を示す場合、デフォルトで有効です。

--lock <backup|clone|create|migrate|rollback|snapshot | snapshot-delete |
suspended | suspending>;
仮想マシンのロック/ロック解除。

--machine [<type=>] [,enable-s3=<1|0>] [,enable-
s4=<1|0>] [,viommu=<intel|virtio>]
QEMU マシンを指定します。

--memory [<current=>] <integer>;
メモリプロパティ。

--migrate_downtime <number> (0 - N) (デフォルト= 0.1)
移行時の最大許容ダウンタイム (秒単位) を設定します。移行終了間際に、新たに変更されたRAMの転送量が多すぎて収束できない場合、移行が収束で
きるまで制限値が段階的に自動的に増加します。

--migrate_speed <integer> (0 - N) (デフォルト= 0)
移行の最大速度 (MB/s単位) を設定します。値0は制限なしを意味します。

--name <string>;
仮想マシンに名前を設定します。設定Webインターフェースでのみ使用されます。

--nameserver <string>;
cloud-init: コンテナのDNSサーバーIPアドレスを設定します。searchdomainもnameserverも設定されていない場合、作成時にはホストの設定が自動的に
使用されます。

--net[n] [<model=>] [,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>]
ネットワークデバイスを指定します。

--numa <boolean> (デフォルト= 0)
NUMAを有効/無効にします。
```

```
--numa[n] cpus=<id[-id];...> [,hostnodes=<id[-id];...>]
[,memory=<number>] [,policy=<preferred|bind|interleave>]
NUMAトポロジー。
```

```
--onboot <boolean> (デフォルト= 0)
システム起動時に仮想マシンを起動するかどうかを指定します。
```

```
--ostype <124| 126| 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。
```

```
--parallel[n] /dev/parport\d+|/dev/usb/lp\d+
ホストのパラレルデバイスをマッピングします (n は 0 から 2) 。
```

```
--pool <文字列>;
指定されたプールに仮想マシンを追加します。
```

```
--protection <boolean> (デフォルト= 0)
VMの保護フラグを設定します。これにより、VMの削除およびディスク削除操作が無効化されます。
```

```
--reboot <boolean> (デフォルト= 0)
再起動を許可します。0に設定すると、再起動時にVMが終了します。
```

```
--rng0 [source=</dev/urandom|/dev/random|/dev/hwrng> [,max_bytes=<integer>]
[,period=<integer>]
VirtIOベースの乱数生成器を設定します。
```

```
--sata[n] [file=<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=<秒>]
[,bps_rd=<bps>][,bps_rd_max_length=<seconds>] [,bps_wr=<bps>]
[,bps_wr_max_length=<seconds>] [,cache=<enum>] [,detect_zeroes=<1|0>][,破棄
=<無視|有効>] [,フォーマット=<列挙型>] [,インポート元=<ソースボリューム>] [,I/O操作
数=<I/O操作数>] [,最大I/O操作数=<I/O操作数>] [,最大I/O操作数持続時間=<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>] [,mbps=<mbps>]
[,mbps_max=<mbps>] [,mbps_rd=<mbps>] [,mbps_rd_max=<mbps>]
[,mbps_wr=<mbps>][,書き込み最大Mb/秒<mb/s>] [,メディア=<CD-ROM|ディスク>] [,レブ
リケーション=<1|0>] [,エラー処理=<無視|報告|停止>] [,シリアル=<シリアル>] [,共有
=<1|0>][,size=<DiskSize>] [,snapshot=<1|0>] [,ssd=<1|0>]
[,werror=<enum>] [,wwn=<wwn>]
ボリュームをSATA/ハードディスクまたはCD-ROMとして使用 (nは0から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。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。
```

```
--scsi[n] [file=<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=<秒>;]
[,bps_rd=<bps>; [,bps_rd_max_length=<seconds>; [,bps_wr=<bps>;]
[,bps_wr_max_length=<seconds>; [,cache=<enum>; [,detect_zeroes=<1|0>; [,破棄
=<無視|有効>; [,フォーマット=<拡張型>; [,インポート元=<ソースボリューム>;]
[,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>; [,書き込みMbps=<Mbps>; [,書き込みMbps上限=<Mbps>; [,メディア
<CD-ROM|ディスク>; [,製品=<製品名>; [,キュー数=<整数>; [,レプリケーション
=<1|0>; [,rerror=<無視|報告|停止>; [,ro=<1|0>; [,scsiblock=<1|0>;]
[,serial=<シリアル番号>; [,shared=<1|0>; [,size=<DiskSize>;]
[,snapshot=<1|0>; [,ssd=<1|0>; [,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 (nは0~30)。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
。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
既存ボリュームからインポートするには、STORAGE_ID:0とimport-
```

```
--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 <string>;
cloud-init: コンテナのDNS検索ドメインを設定します。searchdomainもnameserverも設定されていない場合、Createはホストの設定を自動的に使用しま
す。
```

```
--serial[n] (</dev/.+|socket>)
VM内にシリアルデバイスを作成する (nは0から3)
```

```
--shares <整数>; (0 - 50000) (デフォルト= 1000)
自動バルーニング用のメモリシェア量。数値が大きいほど、このVMが割り当てられるメモリが増加します。数値は他のすべての実行中のVMのウェイト
に対する相対値です。ゼロを使用すると自動バルーニングが無効になります。自動バルーニングはpvestatdによって実行されます。
```

```
--smbios1 [base64=<1|0>; [,family=<Base64エンコード文字列>; [,manufacturer=<Base64エン
コード文字列>; [,product=<Base64エンコード文字列>; [,serial=<Base64エンコード文字列>;]
[,sku=<Base64エンコード文字列>; [,uuid=<UUID>; [,version=<Base64エンコード文字列>;]
SMBIOS タイプ1フィールドを指定します。
```

```
--smp <整数>; (1 - N) (デフォルト= 1)
CPUの数。代わりにオプション -sockets 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 [folderssharing=<1|0>] [,videostreaming=<off|all|filter>]  
SPICE の追加機能設定を行います。

--sshkeys <ファイルパス>;  
cloud-init: 公開SSHキーの設定 (1行に1キー、OpenSSH形式)。

--start <boolean> (デフォルト= 0)  
VMの作成成功後に起動する。

--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します。日付の有効な形式は次の通りです:'now' または 2006-06-17T16:01:21 または  
2006-06-17。

--startup `[[order]=\d+] [\,up=\d+] [\,down=\d+]`  
起動およびシャットダウンの動作。順序は起動時の一般的な順序を定義する非負の数値です。シャットダウンは逆順で行われます。さらに、アップまたは  
ダウンの遅延を秒単位で設定でき、次のVMが起動または停止されるまでの待機時間を指定します。

--storage <ストレージID>;  
デフォルトのストレージ。

--tablet <boolean> (デフォルト= 1)  
USBタブレットデバイスの有効化/無効化。

--tags <string>;  
仮想マシンのタグ。これはメタ情報のみです。

--tdf <boolean> (デフォルト= 0)  
タイムドリフト補正の有効/無効を切り替えます。

--template <boolean> (デフォルト= 0)  
テンプレートの有効化/無効化。

--tpmstate0 [file=<ボリューム>] [,import-from=<ソースボリューム>] [,size=<ディスクサイズ>]  
[,version=<v1.2|v2.0>]  
TPM 状態を保存するディスクを設定します。フォーマットは固定でRAW です。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。ここでSIZE\_IN\_GiBは無視され、代わりに4 MiBが使用されることに注意してください。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 <boolean>;  
一意のランダムなイーサネットアドレスを割り当てます。

---

#### 注記

必要なオプション: archive

---

--unused[n] [file=<ボリューム>]

未使用ボリュームへの参照。これは内部で使用されるため、手動で変更しないでください。

--usb[n] [[host=]<HOSTUSBDEVICE|spice>[,mapping=<mapping-id>[,usb3=<1|0>]]

USBデバイスの設定 (nは0~4。マシンバージョン7.1以上かつostype I26、またはWindows 7以上の場合、nは最大14まで設定可能)。

--vcpus <整数> (1 - N) (デフォルト = 0)

ホットプラグ可能な仮想CPUの数。

--vga [[type=<enum>[,clipboard=<vnc>[,memory=<integer>]]

VGAハードウェアを設定します。

--virtio[n] [file=<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=<秒>[,

,bps\_rd=<bps>[,bps\_rd\_max\_length=<seconds>[,bps\_wr=<bps>[,

,bps\_wr\_max\_length=<seconds>[,cache=<enum>[,detect\_zeroes=<1|0>[,破棄=<無視|有効>[,フォーマット=<列挙型>[,インポート元=<ソースボリューム>[,

,I/O操作数=<I/O操作数>[,最大I/O操作数=<I/O操作数>[,最大I/O操作数持続時間=<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>[,書き込みmbps=<mbps>[,書き込みmbps上限=<mbps>[,メデ

ィア<CD-ROM|ディスク>[,レプリケーション=<1|0>[,エラー処理=<無視|報告|停止>[,

読み取り専用=<1|0>[,シリアル=<シリアル>[,共有=<1|0>[,サイズ=<ディスクサイズ>[,

,スナップショット=<1|0>[,書き込みエラー=<列挙型>[,

ボリュームをVIRTIOハードディスクとして使用 (nは0から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。

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。

--virtiofs[n] [dirid=<mapping-id>[,cache=<enum>]]

Virtio-fsを使用してホストとゲスト間でディレクトリを共有するための設定。

--vmgenid <UUID> (デフォルト = 1 (自動生成))

VM 世代 ID を設定します。作成時または更新時に自動生成するには 1 を指定し、明示的に無効化するには 0 を渡します。

--vmstatestorage <ストレージID>;

VM状態ボリューム/ファイルのデフォルトストレージ。

仮想ハードウェアオッチャドッグデバイスを作成します。

```
qm delsnapshot <vmid> <snapname> [オプション]
```

VMのスナップショットを削除します。

```
<vmid>; <integer> (100 - 999999999)
```

VM の（一意の）ID。

```
<snapname>; <string>
```

スナップショットの名前。

```
--force <boolean>
```

設定ファイルからの削除は、ディスクスナップショットの削除が失敗した場合でも実行されます。

```
qm destroy <vmid> [オプション]
```

VMおよび使用中/所有中の全ボリュームを破棄します。VM固有の権限およびファイアウォールルールを削除します

```
<vmid>; <integer> (100 - 999999999)
```

仮想マシンの（一意の）ID。

```
--destroy-unreferenced-disks <boolean> (デフォルト= 0)
```

設定すると、有効なすべてのストレージから、設定で参照されていないが一致するVMIDを持つディスクを追加で破棄します。

```
--purge <boolean>
```

構成からVMIDを削除します。例：バックアップ＆レプリケーションジョブやHAなど。

```
--skiplock <boolean>
```

ロックを無視する - このオプションはrootのみが使用可能。

```
qm disk import <vmid> <source> <storage> [OPTIONS]
```

外部ディスクイメージをVM内の未使用ディスクとしてインポートします。イメージ形式はqemu-img(1)でサポートされている必要があります。

```
<vmid>; <integer> (100 - 999999999)
```

仮想マシンの（一意の）ID。

```
<source>; <string>
```

インポートするディスクイメージのパス

```
<storage>; <storage ID>
```

ターゲットストレージID

```
--format <qcow2| raw| vmdk>
```

ターゲットフォーマット



```
--target-disk &lt;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| 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| 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24| 未使用240| 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73| 未使用74| 未使用75| 未使用76| 未使用77| 未使用78|  
未使用79| 未使用8| 未使用80| 未使用81| 未使用82| 未使用83| 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90| 未使用91| 未使用92| 未使用93| 未使用94|
```

ボリュームがインポートされるディスク名（例: scsi1）。

**qm disk move** <vmid> <disk> [<storage>] [オプション]

ボリュームを別のストレージまたは別の VM に移動します。

<vmid>: <整数> (100 ~ 99999999)

VMの（一意の）ID。

&lt;disk&gt;;&lt;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 | 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 | 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 | unused197 | unused198 | unused199 | unused2 | 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24 | 未使用240 | 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73 | 未使用74 | 未使用75 | 未使用76 | 未使用77 | 未使用78 | 未使用79 | 未使用8 | 未使用80 | 未使用81 | 未使用82 | 未使用83 | 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90 | 未使用91 | 未使用92 | 未使用93 | 未使用94 |

移動したいディスク。

**&lt;storage&gt;:: &lt;storage ID&gt;**  
移動先のストレージ。

**--bwlimit &lt;integer&gt; (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移動制限)**  
I/O 帯域幅制限 (KiB/s単位) を上書きします。

**--delete &lt;boolean&gt; (デフォルト= 0)**  
コピー成功後に元のディスクを削除します。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。

**--digest &lt;string&gt;**  
現在の設定ファイルのSHA1ハッシュが異なる場合、変更を防止します。これにより同時編集を防止できます。

**--format &lt;qcow2| raw| vmdk&gt;**  
ターゲットフォーマット。

**--target-digest &lt;string&gt;**  
ターゲットVMの現在の設定ファイルのSHA1ダイジェストが異なる場合、変更を防止します。これにより同時編集を検出できます。

```
--target-disk &lt;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| 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| 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24| 未使用240| 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73| 未使用74| 未使用75| 未使用76| 未使用77| 未使用78|  
未使用79| 未使用8| 未使用80| 未使用81| 未使用82| 未使用83| 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90| 未使用91| 未使用92| 未使用93| 未使用94|
```

ターゲットVM上でディスクが移動される構成キー（例：ide0またはscsi1）。デフォルトはソースディスクキー。

**--target-vmid <整数> (100 - 999999999)**

VMの（一意の）ID。

**qm disk rescan [オプション]**

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します。

**--dryrun <ブール値> (デフォルト= 0)**

変更を実際にVM構成ファイルに書き出さない。

**--vmid <integer> (100 - 999999999)**

仮想マシンの（一意の）ID。

**qm disk resize <vmid> <disk> <size> [オプション]**

ボリュームサイズを拡張します。

**<vmid>:<integer> (100 - 999999999)**

仮想マシンの（一意の）ID。

**<disk>:<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 | virtio16 | virtio2 | virtio3 | virtio4 | virtio5 | virtio6 | virtio7 | virtio8 | virtio9>**

サイズを変更したいディスク。

新しいサイズ。+記号がある場合、値はボリュームの実際のサイズに加算され、ない場合は絶対値として扱われます。ディスクサイズの縮小はサポートされていません。

**--digest <string>**

現在の設定ファイルのSHA1ダイジェストが異なる場合、変更を防止します。同時編集の防止に使用できます。

**--skiplock <boolean>**

ロックを無視します - このオプションはrootのみが使用できます。

**qm disk unlink <vmid> --idlist <string> [オプション]**

ディスクイメージのリンク解除/削除を行います。

**&lt;vmid&ampgt; &lt;整数&ampgt** (100 - 999999999)

仮想マシンの（一意の）ID。

**--force &lt;boolean&ampgt**

物理的な削除を強制します。これを指定しない場合、設定ファイルからディスクを削除するだけで、未使用[n]という追加の設定エントリを作成します。

このエントリにはボリュームIDが含まれます。未使用[n]のアンリンクは常に物理的な削除を引き起こします。

**--idlist &lt;string&ampgt**

削除したいディスクIDのリスト。

**qm guest cmd &lt;vmid&ampgt &lt;command&ampgt**

QEMUゲストエージェントコマンドを実行します。

**&lt;vmid&ampgt; &lt;integer&ampgt** (100 - 999999999)

VMの（一意の）ID。

**&lt;command&ampgt; &lt;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&ampgt**

QGAコマンド。

**qm guest exec &lt;vmid&ampgt [&lt;追加引数&ampgt] [オプション]**

ゲストエージェント経由で指定されたコマンドを実行します

**&lt;vmid&ampgt; &lt;integer&ampgt** (100 - 999999999)

VMの（一意の）ID。

**&lt;extra-args&ampgt; &lt;array&ampgt**

追加引数を配列として指定

**--pass-stdin &lt;boolean&ampgt** (デフォルト= 0)

設定すると、EOFまでSTDINを読み取り、*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したプロセスへのSTDINとして扱われます）。最大1MiBまで許可されます。

**--synchronous &lt;boolean&ampgt** (デフォルト= 0)

オフに設定すると、コマンド終了またはタイムアウトを待たずに直ちにpidを返します。

**--timeout &lt;integer&ampgt** (0 - N) (デフォルト= 30)

コマンド完了を同期的に待機する最大時間。到達するとpidが返される。無効化するには0を設定

**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します

**<vmid>: <integer> (100 - 999999999)**  
VMの（一意の）ID。

**<pid>: <integer>**  
問い合わせ対象のPID

**qm guest passwd <vmid> <username> [オプション]**

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します

**<vmid>: <integer> (100 - 999999999)**  
仮想マシンの（一意の）ID。

**<username>: <string>**  
パスワードを設定するユーザー。

**--encrypted <boolean> (デフォルト= 0)**  
パスワードが既にcrypt()を通過している場合に1を設定

**qm help [オプション]**

指定されたコマンドの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します

**--verbose <boolean>**  
詳細出力形式。

**qm import <vmid> <source> --storage <string> [OPTIONS]**

サポートされているインポートソース（ESXiストレージなど）から外部仮想ゲストをインポートします。

**<vmid>: <integer> (100 - 999999999)**  
仮想マシンの（一意の）ID。

**<source>: <string>**  
インポート元のボリュームID。

**--acpi <boolean> (デフォルト= 1)**  
ACPIの有効化/無効化。

**--affinity <string>**  
ゲストプロセスを実行するために使用す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の通信およびそのプロパティの有効化/無効化。

```
--amd-sev [type=<sev-type>; [,allow-smt=<1|0>;]
```

AMD CPUによるセキュア暗号化仮想化（SEV）機能

```
--arch <aarch64| x86_64>;
```

仮想プロセッサーアーキテクチャ。デフォルトはホスト。

```
--args <文字列>;
```

KVMに渡す任意の引数。

```
--audio0 device=<ich9-intel-hda|intel-hda|AC97>; [,driver=<spice|none>;]
```

オーディオデバイスを設定します。QXL/Spiceとの組み合わせで有用です。

```
--autostart <boolean>; (デフォルト= 0)
```

クラッシュ後の自動再起動（現在は無視される）。

```
--balloon <integer>; (0 - N)
```

仮想マシン（VM）のターゲットRAM容量（MiB単位）。0を指定するとバルーンドライバが無効化される。

```
--bios <ovmf| seabios>; (デフォルト= seabios)
```

BIOS実装の選択。

ゲストブート順序を指定します。order=サブプロパティの使用方法（キーなしままたはlegacy=）は非推奨です。

```
--bootdisk (ide|sata|scsi|virtio)\d+
```

指定したディスクからの起動を有効にします。非推奨: 代わりに boot: order=foo;bar を使用してください。

```
--cdrom <ボリューム>;
```

これはオプション -ide2 の別名です

```
--cicustom [meta=<ボリューム>;] [,network=<ボリューム>;] [,user=<ボリューム>;] [,vendor=<ボリューム>;]
```

cloud-init: 起動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します。

```
--cipassword <文字列>;
```

cloud-init: ユーザーに割り当てるパスワード。通常は使用を推奨しません。代わりにSSHキーを使用してください。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点に注意してください。

```
--citype <configdrive2| nocloud| opennebula>;
```

cloud-init設定フォーマットを指定します。デフォルトは設定済みOSタイプ（ostype）に依存します。Linuxではnocloudフォーマット、Windowsではconfigdrive2フォーマットを使用します。

```
--ciupgrade <boolean> (デフォルト= 1)
cloud-init: 初回起動後に自動パッケージアップグレードを実行する。

--ciuser <string>
cloud-init: イメージの設定済みデフォルトユーザーではなく、SSHキーとパスワードを変更するユーザー名。

--cores <integer> (1 - N) (default = 1)
ソケットあたりのコア数。

--cpu [[cpuplatform=<string>] [,flags=<+FLAG|-FLAG...>]
[,hidden=<1|0>] [,hv-vendor-id=<vendor-id>]
[,物理ビット数=<8-64|ホスト>] [,報告モデル=<列挙型>]
エミュレートされるCPUタイプ。

--cpulimit <number> (0 - 128) (デフォルト= 0)
CPU使用率の制限。

--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において[1, 10000]の範囲に制限されます。

--delete <string>
削除したい設定のリスト。

--description <string>
VMの説明。WebインターフェースのVM概要に表示されます。設定ファイル内にコメントとして保存されます。

--dryrun <boolean> (デフォルト= 0)
作成コマンドを表示し、何も実行せずに終了します。

--efidisk0 [file=<ボリューム>] [,efitype=<2m|4m>] [,format=<enum>]
[,pre-enrolled-keys=<1|0>] [,size=<ディスクサイズ>]
EFI変数を保存するためのディスクを設定します。

--format <qcow2| raw| vmdk>
ターゲットフォーマット

--freeze <boolean>
起動時にCPUを凍結（c monitorコマンドで実行を開始）。

--hookscript <string>
VMのライフサイクルにおける様々な段階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>]  
,サブデバイスID=<16進ID>[,サブベンダーID=<16進ID>][,ベンダー-  
ID=<16進ID>][,x-vga=<1|0>]
```

ホストPCIデバイスをゲストにマッピングします。

```
--hotplug <string> (デフォルト= network,disk,usb)
```

ホットプラグ機能を選択的に有効化します。これはホットプラグ機能のカンマ区切りリストです：*network, disk, cpu, memory, usb, cloudinit*。0を指定するとホットプラグを完全に無効化します。値として1を使用すると、デフォルトの*network, disk, usb*の別名となります。USBホットプラグは、マシンバージョン >=

7.1かつostype l26 または Windows> 7 のゲストでUSB ホットプラグが可能。

```
--hugepages <1024| 2| any>;
```

巨大ページメモリの有効化/無効化。

```
--ide[n] [file=<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=<秒>]  
,bps_rd=<bps>[,bps_rd_max_length=<seconds>][,bps_wr=<bps>]  
,bps_wr_max_length=<seconds>[,cache=<enum>][,detect_zeroes=<1|0>][,破棄  
=<1|無視|有効>][,フォーマット=<列挙型>][,I/O操作数=<I/O操作数>][,最大I/O操作数  
=<I/O操作数>][,最大I/O操作数持続時間=<秒>][,読み取りI/O操作数=<I/O操作数  
>][,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>]  
,serial=<serial>][,shared=<1|0>][,size=<DiskSize>][,snapshot=<1|0>]  
,ssd=<1|0>][,werror=<enum>][,wwn=<wwn>]
```

ボリュームをIDEハードディスクまたはCD-ROMとして使用 (nは0から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ですが、同じタイプのIPを指定する必要があります。

IPアドレスに特殊文字列「*dhcp*」を使用するとDHCPを利用でき、この場合明示的なゲートウェイは指定しないでください。IPv6では特殊文字列「*auto*」を使用するとステートレス自動設定を利用できます。これにはcloud-init 19.4以降が必要です。

クラウドインイシャライザが有効で、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を使用します。

```
--ivshmem size=<integer>[,name=<string>]
```

仮想マシン間共有メモリ。仮想マシン間、またはホストとの直接通信に有用です。

--keephugepages <boolean> (デフォルト= 0)

hugepagesと併用します。有効にすると、VMシャットダウン後もhugepagesが削除されず、以降の起動で使用可能になります。

--keyboard <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。このオプションは通常不要であり、ゲストOS内で処理する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を有効/無効にします。

--live-import <boolean> (デフォルト= 0)

VMを直ちに起動し、データをバックグラウンドでコピーします。

--localtime <boolean>

リアルタイムクロック (RTC) を現地時間に設定します。ostype が

Microsoft Windows OS。

--lock <backup| clone| create| migrate| rollback| snapshot | snapshot-delete | suspended | suspending>

仮想マシンのロック/ロック解除。

--machine [<type=>] [,enable-s3=<1|0>] [,enable-s4=<1|0>] [,viommu=<intel|virtio>]

QEMU マシンを指定します。

--memory [<current=>] <integer>

メモリプロパティ。

--migrate\_downtime <number> (0 - N) (デフォルト= 0.1)

移行時の最大許容ダウンタイム（秒単位）を設定します。移行終了間際に、新たに変更されたRAMの転送量が多すぎて収束できない場合、移行が収束するまで制限値が段階的に自動的に増加します。

--migrate\_speed <整数> (0 - N) (デフォルト= 0)

移行の最大速度（MB/s単位）を設定します。値0は制限なしを意味します。

--name <string>

仮想マシンに名前を設定します。設定Webインターフェースでのみ使用されます。

--nameserver <string>

cloud-init: コンテナのDNSサーバーIPアドレスを設定します。searchdomainとnameserverの両方が設定されていない場合、作成時にはホストの設定が自動的に使用されます。

```
--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...]>]
[,&lt;model>=<macaddr>]
ネットワークデバイスを指定します。
```

```
--numa <boolean> (デフォルト= 0)
NUMAを有効/無効にします。
```

```
--numa[n] cpus=<id[-id];...> [,hostnodes=<id[-id];...>]
[,memory=<number>] [,policy=<preferred|bind|interleave>]
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します。
```

```
--ostype <124| 126| 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てください。
```

```
--parallel[n] /dev/parport\<d+>/dev/usb/lp\<d+
ホストのパラレルデバイスをマッピングします (n は 0 から 2)。
```

```
--protection <boolean> (デフォルト= 0)
仮想マシンの保護フラグを設定します。これにより、仮想マシンの削除およびディスク削除操作が無効化されます。
```

```
--reboot <boolean> (デフォルト= 0)
再起動を許可します。0に設定すると、再起動時にVMが終了します。
```

```
--rng0 [source=</dev/urandom|/dev/random|/dev/hwrng>] [,max_bytes=<integer>]
[,period=<integer>]
VirtIOベースの乱数生成器を設定します。
```

```
--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>] [,detect_zeroes=<1|0>][,破棄
=<無視|有効>] [,フォーマット=<列挙型>] [,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>] [,mbps=<mbps>]
[,mbps_max=<mbps>] [,mbps_rd=<mbps>] [,mbps_rd_max=<mbps>]
[,mbps_wr=<mbps>][,書き込み最大Mb/秒<mb/s>] [,メディア=<CD-ROM|ディスク>] [,レプリ
ケーション=<1|0>] [,エラー処理=<無視|報告|停止>] [,シリアル=<シリアル番号>] [,共有
=<1|0>][,size=<DiskSize>] [,snapshot=<1|0>] [,ssd=<1|0>]
[,werror=<enum>] [,wwn=<wwn>]
```

ボリュームをSATA/ハードディスクまたはCD-ROMとして使用 (nは0から5)。

```
--scsi[n] [file=<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=<秒>;]  
[,bps_rd=<bps>;][,bps_rd_max_length=<秒数>;] [,bps_wr=<bps>;]  
[,bps_wr_max_length=<秒数>;] [,cache=<列挙型>;] [,detect_zeroes=<1|0>;]  
[,discard=<無視|有効>;][,format=<enum>;] [,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>;] [,error=<ignore|report|stop>;]  
[,ro=<1|0>;][,scsiblock=<1|0>;] [,serial=<serial>;] [,shared=<1|0>;]  
[,size=<DiskSize>;] [,snapshot=<1|0>;] [,ssd=<1|0>;][,ベンダー=<ベンダー>;]  
[,ワーエラー=<列挙型>;] [,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 (nは0から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

```
--searchdomain <string>;
```

cloud-init: コンテナのDNS検索ドメインを設定します。searchdomainもnameserverも設定されていない場合、Createは自動的にホストの設定を使用します。

```
--serial[n] (</dev/.+>|socket)
```

VM内にシリアルデバイスを作成します (nは0から3)。

```
--shares <integer>; (0 - 50000) (デフォルト= 1000)
```

自動バルーニング用のメモリシェア量。数値が大きいほど、このVMが割り当てられるメモリが増加します。数値は他のすべての実行中VMのウェイトに対する相対値です。0を使用すると自動バルーニングが無効化されます。自動バルーニングはpvestatdによって実行されます。

```
--smbios1 [base64=<1|0>;] [,family=<Base64エンコード文字列>;] [,manufacturer=<Base64エン  
コード文字列>;][,product=<Base64エンコード文字列>;] [,serial=<Base64エンコード文字列>;]  
[,sku=<Base64エンコード文字列>;] [,uuid=<UUID>;] [,version=<Base64エンコード文字列>;]  
SMBIOS タイプ1 フィールドを指定します。
```

```
--smp <整数>; (1 - N) (デフォルト= 1)
```

CPUの数。代わりにオプション -sockets 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 の追加機能設定。

--sshkeys <文字列>;  
cloud-init: パブリックSSHキーの設定 (1行に1キー、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します。日付の有効な形式は次の通りです:'now' または 2006-06-17T16:01:21 または  
2006-06-17。

--startup [[order=\d+] [,up=\d+] [,down=\d+]]  
起動およびシャットダウンの動作。順序は起動時の一般的な順序を定義する非負の数値です。シャットダウンは逆順で行われます。さらに、アップまたは  
ダウングレードの遅延を秒単位で設定でき、これは次のVMが起動または停止されるまでの待機時間を指定します。

--storage <ストレージID>;  
デフォルトのストレージ。

--tablet <ブール値> (デフォルト= 1)  
USBタブレットデバイスの有効化/無効化。

--tags <文字列>;  
仮想マシンのタグ。これは単なるメタ情報です。

--tdf <boolean> (デフォルト= 0)  
タイムドリフト補正の有効/無効設定。

--template <boolean> (デフォルト= 0)  
テンプレートの有効化/無効化。

--tpmstate0 [file=<ボリューム>] [,size=<ディスクサイズ>]  
,version=<v1.2|v2.0>]  
TPM状態を保存するディスクを設定します。フォーマットはraw/固定されます。

--unused[n] [file=<ボリューム>]  
未使用ボリュームへの参照。内部で使用されるため、手動で変更しないでください。

--usb[n] [[host=<HOSTUSBDEVICE|spice>] [,mapping=<mapping-id>]  
,usb3=<1|0>]  
USBデバイスの設定 (nは0~4。マシンバージョン7.1以上かつostype I26、またはWindows 7以上の場合、nは最大14まで設定可能)。

--vcpus <整数> (1 - N) (デフォルト= 0)  
ホットプラグ可能な仮想CPUの数。

```
--vga [[type=<enum>] [,clipboard=<vnc>] [,memory=<integer>]]  
VGAハードウェアを設定します。
```

```
--virtio[n] [file=<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=<秒>]  
,bps_rd=<bps> [,bps_rd_max_length=<seconds>] [,bps_wr=<bps>]  
,bps_wr_max_length=<seconds> [,cache=<enum>] [,detect_zeroes=<1|0>]  
,discard=<ignore|on> [,format=<enum>] [,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>][,書き込み_最大  
_mbps=<mbps>] [,メディア=<cdrom|disk>] [,レプリケーション=<1|0>] [,エラー処理  
=<無視|報告|停止>] [,読み取り専用=<1|0>] [,シリアル=<シリアル>] [,共有  
=<1|0>] [,サイズ=<ディスクサイズ>] [,スナップショット=<1|0>] [,書き込みエラー  
=<列挙型>]  
ボリュームをVIRTIOハードディスクとして使用 (nは0から15)。
```

```
--virtiofs[n] [dirid=<mapping-id>] [,cache=<enum>]
```

Virtio-fs を使用してホストとゲスト間でディレクトリを共有するための設定。

```
--vmgenid <UUID> (デフォルト= 1 (自動生成))
```

VM 世代 ID を設定します。作成時または更新時に自動生成するには 1 を指定し、明示的に無効化するには 0 を渡します。

```
--vmstatestorage <ストレージID>;
```

VM状態ボリューム/ファイルのデフォルトストレージ。

仮想ハードウェアオッチャドッグデバイスを作成します。

```
qm importdisk
```

*qm disk import* の別名。

```
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 を作成します

```
<vmid>; <integer> (100 - 99999999)
```

仮想マシンの（一意の）ID。

```
<manifest>; <string>;
```

OVFファイルへのパス

```
<storage>; <storage ID>;
```

対象ストレージID

--dryrun &lt;boolean&ampgt  
抽出されたOVF/パラメータの解析結果を表示するが、VMは作成しない

--format &lt;qcow2| raw| vmdk&ampgt  
ターゲットフォーマット

#### qm list [オプション]

仮想マシンインデックス（ノードごと）。

--full &lt;boolean&ampgt  
アクティブな仮想マシンのフルステータスを判定します。

#### qm listsnapshot &lt;vmid&ampgt

すべてのスナップショットを一覧表示します。

&lt;vmid&ampgt; &lt;integer&ampgt (100 - 999999999)  
VMの（一意の）ID。

#### qm migrate &lt;vmid&ampgt &lt;target&ampgt [オプション]

仮想マシンの移行。新しい移行タスクを作成します。

&lt;vmid&ampgt; &lt;integer&ampgt (100 - 999999999)  
仮想マシンの（一意の）ID。

&lt;target&ampgt; &lt;string&ampgt  
移行先のノード。

--bwlimit &lt;integer&ampgt (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移行制限)  
I/O帯域幅制限（KiB/s単位）を上書きします。

--force &lt;boolean&ampgt  
ローカルデバイスを使用するVMの移行を許可します。このオプションはrootのみが使用できます。

--migration\_network &lt;文字列&ampgt  
移行に使用する（サブ）ネットワークのCIDR。

--migration\_type &lt;insecure| secure&ampgt  
デフォルトでは、移行トラフィックは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、パフォーマンス向上のためにこれを無効化できます。

--online &lt;boolean&ampgt  
VMが実行中の場合にオンライン/ライブ移行を使用します。VMが停止中の場合は無視されます。

**--targetstorage <string>**

ソース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ます。特別な値1を指定すると、各ソースストレージがそれ自身にマッピングされます。

**--with-contrack-state <boolean> (デフォルト= 0)**

実行中のVMの接続状態エントリを移行するかどうか。

**--with-local-disks <boolean>**

ローカルディスクに対するライブストレージ移行を有効化

**qm monitor <vmid>**

QEMU Monitor インターフェースに入る。

**<vmid>; <integer> (100 - 999999999)**

仮想マシンの（一意の）ID。

**qm move-disk**

*qm disk move* の別名。

**qm move\_disk**

*qm disk move* の別名。

**qm mtunnel**

*qmigrate* によって使用されます - 手動では使用しないでください。

**qm nbdstop <vmid>**

組み込みの NBD サーバーを停止します。

**<vmid>; <integer> (100 - 999999999)**

仮想マシンの（一意の）ID。

**qm pending <vmid>**

仮想マシンの設定を取得します。現在の値と保留中の値の両方が含まれます。

**<vmid>; <integer> (100 - 999999999)**

仮想マシンの（一意の）ID。

**qm reboot <vmid> [オプション]**

VMをシャットダウンして再起動します。保留中の変更を適用します。

**<vmid>; <整数> (100 - 999999999)**

仮想マシンの（一意の）ID。

--timeout <整数> (0 - N)

シャットダウンに最大タイムアウト秒待機します。

**qm remote-migrate** <vmid> [<target-vmid>] <target-endpoint> --target-bridge <str>  
--target-bridge <str>

仮想マシンをリモートクラスターに移行します。新しい移行タスクを作成します。実験的機能です！

<vmid>: <integer> (100 - 999999999)

仮想マシンの（一意の）ID。

<target-vmid>: <integer> (100 - 999999999)

仮想マシンの（一意の）ID。

<target-endpoint>: apitoken=<PVEAPIToken=user@realm!token=SECRET>;  
, host=<ADDRESS> [,fingerprint=<FINGERPRINT>] [,port=<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移行制限)

I/O帯域幅制限を上書き（単位：KiB/s）。

--delete <boolean> (デフォルト= 0)

移行成功後に元の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ーに残ります。

--online <boolean>;

VMが実行中の場合はオンライン/ライブ移行を使用します。VMが停止中の場合は無視されます。

--target-bridge <string>;

ソースからターゲットへのブリッジのマッピング。単一のブリッジIDのみを指定すると、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、各ソースブリッジがそれ自身にマッピングされます。

--target-storage <string>;

ソースからターゲットへのストレージのマッピング。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ます。特別な値1を指定すると、各ソースストレージがそれ自身にマッピングされます。

**qm rescan**

*qm disk rescan* の別名。

**qm reset** <vmid> [オプション]

仮想マシンをリセットします。

<vmid>: <integer> (100 - 999999999)

仮想マシンの（一意の）ID。

--skiplock <boolean>;

ロックを無視する - このオプションの使用はrootのみに許可される。

**qm resize**

*qm disk resize* の別名。

**qm resume** <vmid> [オプション]

仮想マシンの再開。

<vmid>::<integer> (100 - 999999999)

仮想マシンの（一意の）ID。

--nocheck <ブール値>;

説明なし

--スキップロック <ブール値>;

ロックを無視する - このオプションの使用は root のみに許可される。

**qm rollback** <vmid> <snapname> [オプション]

VMの状態を指定されたスナップショットにロールバックします。

<vmid>::<整数> (100 - 999999999)

仮想マシンの（一意の）ID。

<snapname>::<string>;

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: スナップショットにRAMが含まれる場合、VMは自動的に起動されます。)

**qm sendkey** <vmid> <key> [オプション]

仮想マシンにキーイベントを送信します。

<vmid>::<integer> (100 - 999999999)

仮想マシンの（一意の）ID。

<key>::<string>;

キー (qemu monitor エンコーディング)。

--skiplock <boolean>;

ロックを無視する - このオプションはrootのみが使用できます。

**qm set <vmid> [オプション]**

仮想マシンオプションの設定（同期API） - ホットプラグやストレージ割り当てに関する操作には、代わりにPOSTメソッドの使用を検討してください。

**<vmid>; <integer> (100 - 999999999)**

仮想マシンの（一意の）ID。

**--acpi <boolean> (デフォルト= 1)**

ACPIの有効化/無効化。

**--affinity <文字列>**

ゲストプロセスを実行するために使用す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の通信を有効/無効にし、そのプロパティを設定します。

**--amd-sev [type=<sev-type> [,allow-smt=<1|0>]]**

AMD CPUによるセキュア暗号化仮想化（SEV）機能

**--arch <aarch64| x86\_64>;**

仮想プロセッサーアーキテクチャ。デフォルトはホスト。

**--args <文字列>**

KVMに渡す任意の引数。

**--audio0 device=<ich9-intel-hda|intel-hda|AC97> [,driver=<spice|none>]**

オーディオデバイスを設定します。QXL/Spiceとの組み合わせで有用です。

**--autostart <boolean> (デフォルト= 0)**

クラッシュ後の自動再起動（現在は無視される）。

**--balloon <integer> (0 - N)**

VMのターゲットRAM容量（MiB単位）。ゼロを使用するとバルーンドライバが無効化されます。

**--bios <ovmf| seabios> (デフォルト= seabios)**

BIOS実装を選択します。

ゲストブート順序を指定します。order=サブプロパティの使用法（キーなしまたは`/legacy=`）は非推奨です。

**--bootdisk (ide|sata|scsi|virtio)\d+**

指定したディスクからの起動を有効化。非推奨: 代わりに `boot: order=foo;bar` を使用してください。

--cdrom <ボリューム>;  
オプション -ide2 の別名です

--cicustom [meta=<ボリューム>] [,network=<ボリューム>] [,user=<ボリューム>] [,vendor=<ボリューム>]  
cloud-init: 起動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します。

--cipassword <password>;  
cloud-init: ユーザーに割り当てるパスワード。通常は使用を推奨しません。代わりにSSHキーを使用してください。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点に注意してください。

--citype <configdrive2| nocloud| opennebula>;  
cloud-init設定フォーマットを指定します。デフォルトは設定済みOSタイプ (ostype) に依存します。Linuxではnocloudフォーマット、Windowsではconfigdrive2フォーマットを使用します。

--ciupgrade <boolean> (デフォルト= 1)  
cloud-init: 初回起動後に自動パッケージアップグレードを実行します。

--ciuser <文字列>;  
cloud-init: イメージに設定されたデフォルトユーザーではなく、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...]>] [,hidden=<1|0>] [,hv-vendor-id=<vendor-id>] [,物理ビット数=<8-64>] [,報告モデル=<列挙型>]  
エミュレートされるCPUタイプ。

--cpulimit <number> (0 - 128) (デフォルト= 0)  
CPU使用率の制限。

--cpuunits <integer> (1 - 262144) (デフォルト= 0)  
VM の CPU ウェイト。cgroup v2 では [1, 10000] に制限されます。

--delete <string>;  
削除したい設定のリスト。

--description <string>;  
VMの説明。WebインターフェースのVM概要に表示されます。設定ファイル内にコメントとして保存されます。

--digest <string>;  
現在の設定ファイルのSHA1ダイジェストが異なる場合、変更を防止します。これにより同時編集を防止できます。

```
--efidisk0 [file=<ボリューム> [,efitype=<2m|4m>] [,format=<列挙型>] [,import-from=<ソースボリューム>] [,pre-enrolled-keys=<1|0>] [,size=<ディスクサイズ>]
```

EFI変数を保存するディスクを設定します。特別な構文 `STORAGE_ID:SIZE_IN_GiB` を使用して新しいボリュームを割り当てます。`SIZE_IN_GiB` はここで無視され、代わりにデフォルトの EFI 変数がボリュームにコピーされることに注意してください。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 <boolean>
```

物理的な削除を強制します。これを指定しない場合、ディスクを構成ファイルから削除するだけで、未使用[n]という追加の構成エントリが作成されます。このエントリにはボリュームIDが含まれます。未使用[n]のアンリンクは常に物理的な削除を引き起こします。

---

#### 注記

必要なオプション: `delete`

---

```
--freeze <boolean>
```

起動時にCPUを凍結する（`c monitor`コマンドを使用して実行を開始）。

```
--hookscript <string>
```

VMのライフサイクルにおける様々な段階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>] [,サブデバイスID=<16進ID>] [,サブベンダーID=<16進ID>] [,ベンダーID=<16進ID>] [,x-vga=<1|0>]
```

ホストPCIデバイスをゲストにマッピングします。

```
--hotplug <string> (デフォルト= network,disk,usb)
```

ホットプラグ機能を選択的に有効化します。これはホットプラグ機能のカンマ区切りリストです：ネットワーク、ディスク、CPU、メモリ、USB、`cloudinit`。0を指定するとホットプラグを完全に無効化します。値として1を使用すると、デフォルトのネットワーク、ディスク、USBの別名となります。USBホットプラグはマシンバージョンが>=7.1 および ostype l26 または windows>7.

```
--hugepages <1024|2|any>
```

巨大ページメモリの有効化/無効化。

```
--ide[n] [file=<ボリューム>; ,aio=<ネイティブ|スレッド|io_uring>;]
[,backup=<1>; [,bps=<bps>; [,bps_max_length=<秒>;]
[,bps_rd=<bps>;][,bps_rd_max_length=<seconds>; [,bps_wr=<bps>;]
[,bps_wr_max_length=<seconds>; [,cache=<enum>; [,detect_zeroes=<1>; [,破
棄=<無視|有効>; [,フォーマット=<列挙型>; [,インポート元=<ソースボリューム>;]
[,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>; [,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>; [,rerror=<ignore|report|stop>;]
[,serial=<serial>;][,共有=<1>; [,サイズ=<ディスクサイズ>; [,スナップショット
=<1>; [,SSD=<1>; [,書き込みエラー=<列挙型>; [,WWN=<WWN>;]
ボリュームをIDEハードディスクまたはCD-ROMとして使用（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
。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。
```

```
--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ですが、同じタイプのIPアドレスを指定する必要があります。
IPアドレスにDHCPを使用するには特殊文字列「dhcp」を使用でき、この場合明示的なゲートウェイは指定しない。IPv6では特殊文字列「auto」を使用
してステートレス自動設定を利用可能。これにはcloud-init 19.4以降が必要。
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を使用します。
```

```
--ivshmem size=<integer>; [,name=<string>;]
仮想マシン間共有メモリ。仮想マシン間、またはホストとの直接通信に有用です。
```

```
--keephugepages <boolean>; (デフォルト= 0)
hugepagesと併用します。有効にすると、VMシャットダウン後もhugepagesが削除されず、以降の起動で使用可能になります。
```

```
--keyboard <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。このオプションは通常不要であり、ゲストOS内で設定する方が望ましい場合が多い。
```

```
--kvm <boolean>; (デフォルト= 1)
KVMハードウェア仮想化を有効/無効にします。
```

```
--localtime &lt;boolean&ampgt
リアルタイムクロック (RTC) を現地時間に設定します。 ostype が
Microsoft Windows OS。
```

```
--lock &lt;backup| clone| create| migrate| rollback| snapshot | snapshot-delete |
suspended | suspending&ampgt;
仮想マシンのロック/ロック解除。
```

```
--machine [[type=<マシンタイプ>] [,enable-s3=<1|0>] [,enable-
s4=<1|0>] [,viommu=<intel|virtio>]
QEMU マシンを指定します。
```

```
--memory [current=<integer>];
メモリプロパティ。
```

```
--migrate_downtime <数値> (0 - N) (デフォルト= 0.1)
移行時の最大許容ダウンタイム (秒単位) を設定します。移行の最終段階で、新たに変更されたRAMの転送量が多すぎて収束できない場合、移行が収束
できるまで制限値が段階的に自動的に増加します。
```

```
--migrate_speed <整数> (0 - N) (デフォルト= 0)
移行の最大速度 (MB/s単位) を設定します。値0は制限なしを意味します。
```

```
--name <string>;
仮想マシンに名前を設定します。設定Webインターフェースでのみ使用されます。
```

```
--nameserver <string>;
cloud-init: コンテナのDNSサーバーIPアドレスを設定します。searchdomainとnameserverの両方が設定されていない場合、作成時にはホストの設定が自
動的に使用されます。
```

```
--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>]
ネットワークデバイスを指定します。
```

```
--numa <boolean> (デフォルト= 0)
NUMAを有効/無効にします。
```

```
--numa[n] cpus=<id[-id];...> [,hostnodes=<id[-id];...>]
[,memory=<number>] [,policy=<preferred|bind|interleave>]
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します。
```

```
--ostype <124| 126| 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てください。

```
--parallel[n] /dev/parport\d+|/dev/usb/lp\d+
```

ホストのパラレルデバイスをマッピングします (n は 0 から 2)。

```
--protection <boolean> (デフォルト= 0)
```

仮想マシンの保護フラグを設定します。これにより、仮想マシンの削除およびディスク削除操作が無効化されます。

```
--reboot <boolean> (デフォルト= 0)
```

再起動を許可します。0に設定すると、再起動時にVMが終了します。

```
--revert <string>;
```

保留中の変更を元に戻します。

```
--rng0 [source=</dev/urandom|/dev/random|/dev/hwrng>; [,max_bytes=<integer>;] [,period=<integer>;]
```

VirtIOベースの乱数生成器を設定します。

```
--sata[n] [file=<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=<秒>;]
```

```
[,bps_rd=<bps>;][,bps_rd_max_length=<seconds>;] [,bps_wr=<bps>;]
```

```
[,bps_wr_max_length=<seconds>;] [,cache=<enum>;] [,detect_zeroes=<1|0>;][,破棄
```

```
=<無視|有効>;] [,フォーマット=<拡張型>;] [,インポート元=<ソースボリューム>;]
```

```
[,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>;] [,mbps=<mbps>;]
```

```
[,mbps_max=<mbps>;] [,mbps_rd=<mbps>;] [,mbps_rd_max=<mbps>;]
```

```
[,mbps_wr=<mbps>;][,書き込み最大Mbps=<mbps>;] [,メディア=<CD-ROM|ディスク>;] [,レプリ
```

```
ケーション=<1|0>;] [,エラー処理=<無視|報告|停止>;] [,シリアル=<シリアル>;] [,共有
```

```
<1|0>;][,size=<DiskSize>;] [,snapshot=<1|0>;] [,ssd=<1|0>;]
```

```
[,werror=<enum>;] [,wwn=<wwn>;]
```

ボリュームをSATA/ハードディスクまたはCD-ROMとして使用 (nは0~5)。新しいボリュームを割り当てるには特殊構文STORAGE\_ID:SIZE\_IN\_GiBを使用。既

存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。

```
--scsi[n] [file=<ボリューム>; ,aio=<ネイティブ|スレッド|io_uring>;]
[,backup=<1> [,bps=<bps>; [,bps_max_length=<秒>;]
[,bps_rd=<bps>; [,bps_rd_max_length=<seconds>; [,bps_wr=<bps>;]
[,bps_wr_max_length=<seconds>; [,cache=<enum>;]
[,detect_zeroes=<1>[,discard=<無視|有効>; [,format=<列挙型>; [,import-
from=<ソースボリューム>; [,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=<秒数>; [,iothread=<1>]
[,mbps=<mbps>; [,mbps_max=<mbps>; [,mbps_rd=<mbps>;]
[,mbps_rd_max=<mbps>; [,書き込みMbps=<Mbps>; [,書き込みMbps上限=<Mbps>; [,メディア
<CD-ROM|ディスク>; [,製品=<製品名>; [,キュー数=<整数>; [,レプリケーション
<1>[,rerror=<無視|報告|停止>; [,ro=<1> [,scsiblock=<1>]
[,serial=<シリアル番号>; [,shared=<1> [,size=<DiskSize>;]
[,snapshot=<1> [,ssd=<1> [,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（nは0～30）。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
。既存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。
既存ボリュームからインポートします。
```

```
--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 <string>;
cloud-init: コンテナのDNS検索ドメインを設定します。searchdomainもnameserverも設定されていない場合、Createはホストの設定を自動的に使用しま
す。
```

```
--serial[n] (</dev/.+>|socket)
VM内にシリアルデバイスを作成します（nは0から3）。
```

```
--shares <整数>; (0 - 50000) (デフォルト = 1000)
自動バルーニング用のメモリシェア量。数値が大きいほど、このVMが割り当てられるメモリ量が増加します。数値は他の全稼働VMのウェイトに対する
相対値です。0を指定すると自動バルーニングが無効化されます。自動バルーニングはpvestatdによって実行されます。
```

```
--スキップロック <布尔値>;
ロックを無視する - このオプションはrootのみが使用可能。
```

```
--smbios1 [base64=<1> [,family=<Base64エンコード文字列>; [,manufacturer=<Base64エン
コード文字列>; [,product=<Base64エンコード文字列>; [,serial=<Base64エンコード文字列>;]
[,sku=<Base64エンコード文字列>; [,uuid=<UUID>; [,version=<Base64エンコード文字列>;]
SMBIOS タイプ1 フィールドを指定します。
```

```
--smp <整数>; (1 - N) (デフォルト = 1)
CPUの数。代わりにオプション -sockets 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 [<folderssharing=>1|0] [,videostreaming=<off|all|filter>]
SPICEの追加機能設定

--sshkeys <ファイルパス>;
cloud-init: 公開SSHキーの設定 (1行に1キー、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します。日付の有効な形式は次の通りです:'now' または 2006-06-17T16:01:21 または
2006-06-17。

--startup `[[order]=<d+> [<up=>d+] [<down=>d+]` 
起動およびシャットダウン時の動作。Orderは起動順序を定義する非負の整数です。シャットダウン時は逆順で実行されます。さらに、アップダウン遅延 (秒単位) を設定可能で、次のVM起動/停止までの待機時間を指定します。

--tablet <boolean> (デフォルト= 1)
USBタブレットデバイスの有効化/無効化。

--tags <string>;
仮想マシンのタグ。これは単なるメタ情報です。

--tdf <boolean> (デフォルト= 0)
タイムドリフト補正の有効/無効設定。

--template <boolean> (デフォルト= 0)
テンプレートの有効化/無効化。

--tpmstate0 [<file=>ボリューム] [,import-from=<ソースボリューム>] [,size=<ディスクサイズ>]
[,version=<v1.2|v2.0>]
TPM状態を保存するディスクを設定します。フォーマットは固定でrawです。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。ここでSIZE_IN_GiBは無視され、代わりに4 MiBが使用されることに注意してください。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=>ボリューム];
未使用ボリュームへの参照。内部で使用されるため、手動で変更しないでください。

--usb[n] [[host=>HOSTUSBDEVICE|spice] [,mapping=<mapping-id>]
[,usb3=<1|0>]
USBデバイスを設定します (nは0から4。マシンバージョン7.1以上かつostype l26またはWindows7以上の場合、nは最大14まで設定可能)。
```

--vcpus <整数> (1 - N) (デフォルト= 0)

ホットプラグ可能な仮想CPUの数。

--vga [[type=<enum>] [,clipboard=<vnc>] [,memory=<integer>]]  
VGAハードウェアを設定します。

--virtio[n] [file=<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=<秒>]  
[,bps\_rd=<bps>] [,bps\_rd\_max\_length=<seconds>] [,bps\_wr=<bps>]  
[,bps\_wr\_max\_length=<seconds>] [,cache=<enum>]  
[,detect\_zeroes=<1|0>] [,discard=<無視|有効>] [,format=<列挙型>] [,import-  
from=<ソースボリューム>] [,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>] [,書き込みMbps=<Mbps>] [,書き込みMbps上限=<Mbps>] [,メデ  
ィア<CD-ROM|ディスク>] [,レプリケーション=<1|0>] [,エラー処理=<無視|報告|停止>] [,  
読み取り専用=<1|0>] [,シリアル=<シリアル番号>] [,共有=<1|0>] [,サイズ=<ディスク  
サイズ>] [,スナップショット=<1|0>] [,書き込みエラー=<列挙型>]  
ボリュームをVIRTIOハードディスクとして使用 (nは0から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。  
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。

--virtiofs[n] [dirid=<mapping-id>] [,cache=<enum>]

ホストとゲスト間でディレクトリを共有するためのVirtio-fs設定。

--vmgenid <UUID> (デフォルト= 1 (自動生成))

VM Generation IDを設定します。作成時または更新時に自動生成するには1を、明示的に無効化するには0を指定します。

--vmstatestorage <ストレージID>

VM状態ボリューム/ファイルのデフォルトストレージ。

仮想ハードウェアオッチャドッグデバイスを作成します。

qm showcmd <vmid> [オプション]

VM起動に使用されるコマンドラインを表示します (デバッグ情報)。

<vmid>; <integer> (100 - 99999999)

仮想マシンの (一意の) ID。

--pretty <boolean> (デフォルト= 0)

各オプションを改行して表示し、可読性を向上させます

--snapshot <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します。物理マシンの電源ボタンを押すのと同様です。ゲストOSにACPIイベントを送信し、クリーンなシャットダウンを実行します。

<vmid>; <integer> (100 - 99999999)

仮想マシンの（一意の）ID。

--forceStop <boolean> (デフォルト= 0)

VMが確実に停止します。

--keepActive <ブール値> (デフォルト= 0)

ストレージボリュームを非アクティブ化しない。

--skiplock <boolean>;

ロックを無視する - このオプションはrootのみが使用可能。

--timeout <整数> (0 ~ N)

最大タイムアウト秒待機する。

**qm snapshot** <vmid> <snapname> [オプション]

VMのスナップショットを作成します。

<vmid>; <integer> (100 - 99999999)

VMの（一意の）ID。

<snapname>; <string>;

スナップショットの名前。

--description <文字列>;

テキストによる説明またはコメント。

--vmstate <boolean>;

vmstate を保存

**qm start** <vmid> [オプション]

仮想マシンを起動します。

<vmid>; <integer> (100 - 99999999)

仮想マシンの（一意の）ID。

--force-cpu <文字列>;

QEMUの-cpu引数を指定された文字列で上書きします。

--machine [[type=]<マシンタイプ>] [,enable-s3=<1|0>] [,enable-s4=<1|0>] [,viommu=<intel|virtio>]

QEMUマシンを指定します。

--migratedfrom <string>;

クラスタノード名。

--migration\_network <string>;

移行に使用する（サブ）ネットワークのCIDR。

--migration\_type <insecure| secure>;

デフォルトでは、移行トラフィックは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、パフォーマンス向上のためにこれを無効化できます。

--skiplock <boolean>;

ロックを無視する - このオプションはrootのみが使用できます。

--stateuri <string>;

一部のコマンドはこの場所から状態を保存/復元します。

--targetstorage <string>;

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ます。特別な値1を指定すると、各ソースストレージが自身にマッピングされます。

--timeout <integer> (0 - N) (デフォルト= 最大(30, vmメモリ in GiB))

最大タイムアウト秒待機します。

--with-contrack-state <boolean> (デフォルト= 0)

実行中の VM の接続状態エントリを移行するかどうか。

**qm status** <vmid> [オプション]

VMの状態を表示します。

<vmid>: <integer> (100 - 999999999)

仮想マシンの（一意の）ID。

--verbose <boolean>;

詳細出力形式

**qm stop** <vmid> [オプション]

仮想マシンを停止します。qemuプロセスは直ちに終了します。これは動作中のコンピュータの電源プラグを抜くことに相当し、VMデータが破損する可能性があります。

**&lt;vmid&gt;; &lt;integer&gt; (100 - 999999999)**  
仮想マシンの（一意の）ID。

**--keepActive &lt;boolean&gt; (デフォルト= 0)**  
ストレージボリュームを非アクティブ化しない。

**--migratedfrom &lt;string&gt;**  
クラスタノード名。

**--overrule-shutdown &lt;boolean&gt; (デフォルト= 0)**  
停止前にアクティブなqmshutdownタスクを中止しようとする。

**--スキップロック &lt;ブール値&gt;**  
ロックを無視する - このオプションの使用はrootのみに許可される。

**--timeout &lt;integer&gt; (0 - N)**  
最大タイムアウト秒待機します。

**qm suspend &lt;vmid&gt; [オプション]**

仮想マシンをサスPENDします。

**&lt;vmid&gt;; &lt;integer&gt; (100 - 999999999)**  
仮想マシンの（一意の）ID。

**--スキップロック &lt;ブール値&gt;**  
ロックを無視する - このオプションは root のみが使用可能。

**--statestorage &lt;ストレージID&gt;**  
VMの状態を保存するストレージ

---

#### 注記

必要なオプション: todisk

---

**--todisk &lt;boolean&gt; (デフォルト= 0)**  
設定すると、VMをディスクにサスPENDします。次回VM起動時に再開されます。

**qm template &lt;vmid&gt; [オプション]**

テンプレートを作成します。

**&lt;vmid&gt;; &lt;integer&gt; (100 - 999999999)**  
仮想マシンの（一意の）ID。

---

```
--disk &lt;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&gt;
```

1つのディスクのみをベースイメージに変換したい場合。

#### **qm terminal &lt;vmid&gt; [オプション]**

シリアルデバイスを使用してターミナルを開く（VMにシリアルデバイスが設定されている必要があります。例：`serial0: socket`）

```
&lt;vmid&gt;:&lt;整数&gt; (100 - 99999999)
```

仮想マシンの（一意の）ID。

```
--escape &lt;文字列&gt; (デフォルト= ^O)
```

エスケープ文字。

```
--iface &lt;serial0| serial1| serial2| serial3&gt;
```

シリアルデバイスを選択します。デフォルトでは、最初に利用可能なデバイスを使用します。

#### **qm unlink**

`qm disk unlink` の別名。`qm unlock`

&lt;vmid&gt;; VM のロックを解除し

ます。

```
&lt;vmid&gt;:&lt;integer&gt; (100 - 99999999)
```

仮想マシンの（一意の）ID。

#### **qm vncproxy &lt;vmid&gt;**

VNCトラフィックを標準入力/標準出力にプロキシ

```
&lt;vmid&gt;:&lt;integer&gt; (100 - 99999999)
```

VMの（一意の）ID。

#### **qm wait &lt;vmid&gt; [オプション]**

VMが停止するまで待機します。

```
&lt;vmid&gt;:&lt;integer&gt; (100 - 99999999)
```

仮想マシンの（一意の）ID。

```
--timeout &lt;整数&gt; (1 - N)
```

タイムアウト（秒単位）。デフォルトは無期限待機。

## A.10 qmrestore - QemuServer vzdump バックアップの復元

### qmrestore ヘルプ

```
qmrestore <アーカイブ> <vmid> [オプション]
```

QemuServer vzdump バックアップを復元します。

```
<archive>:<string>;
```

バックアップファイル。標準入力から読み込むには - を指定します。

```
<vmid>:<整数> (100 ~ 999999999)
```

仮想マシンの（一意の）ID。

```
--bwlimit <数値> (0 - N)
```

I/O帯域幅制限 (KiB/s単位) を上書きします。

```
--force <boolean>;
```

既存のVMを上書きすることを許可します。

```
--live-restore <boolean>;
```

バックアップから直ちにVMを起動し、復元をバックグラウンドで実行。PBS専用。

```
--pool <文字列>;
```

指定されたプールにVMを追加します。

```
--storage <storage ID>;
```

デフォルトのストレージ。

```
--unique <boolean>;
```

一意のランダムなイーサネットアドレスを割り当てます。

## A.11 pct - Proxmox Container Toolkit

```
pct <コマンド> [引数] [オプション]
```

```
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>:<integer> (100 - 999999999)
```

VMの（一意の）ID。

```
<newid>:<integer> (100 - 999999999)
```

クローン用の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帯域幅制限 (KiB/s単位) を上書きします。

--description <string>;
新規CTの説明。

--full <boolean>;
すべてのディスクの完全コピーを作成します。通常のCTをクローンする場合、常に実行されます。CTテンプレートでは、デフォルトでリンククローンを作成しようとします。

--hostname <string>;
新しいCTのホスト名を設定します。

--pool <string>;
新しいCTを指定されたプールに追加します。

--snapname <string>;
スナップショットの名前。

--storage <storage ID>;
フルクローン用のターゲットストレージ。

--target <string>;
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ます。

pct config <vmid>; [オプション]
コンテナ構成を取得します。

<vmid>; <integer>; (100 - 999999999)
VMの（一意の）ID。

--current <boolean> (デフォルト= 0)
現在の値を取得する（保留中の値ではなく）。

--snapshot <string>;
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 console <vmid>; [オプション]
指定されたコンテナのコンソールを起動します。

<vmid>; <integer>; (100 - 999999999)
VMの（一意の）ID。
```

--escape \^? [a-z] (デフォルト= ^a)

エスケープシーケンスのプレフィックス。例：&lt;Ctrl+b q&gt; をエスケープシーケンスとして使用するには ^bを指定。

#### pct cpusets

割り当てられたCPUセットのリストを出力します。

**pct create** &lt;vmid&gt; &lt;ostemplate&gt; [オプション]

コンテナを作成または復元します。

&lt;vmid&gt;: &lt;integer&gt; (100 - 999999999)

仮想マシンの（一意の）ID。

&lt;ostemplate&gt;: &lt;string&gt;

OSテンプレートまたはバックアップファイル。

--arch &lt;amd64| arm64| armhf| i386| riscv32| riscv64&gt; (デフォルト= amd64)

OS アーキテクチャタイプ。

--bwlimit &lt;number&gt; (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 帯域幅制限を上書き (KiB/s 単位)。

--cmode &lt;console| shell| tty&gt; (デフォルト= tty)

コンソールモード。デフォルトでは、コンソールコマンドは利用可能なttyデバイスの一つへの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へのアタッチを試みます。cmodeをshellに設定すると、コンテナ内で単純にシェルを起動します（ログインなし）。

--console &lt;boolean&gt; (デフォルト= 1)

コンソールデバイス (/dev/console) をコンテナにアタッチします。

--cores &lt;integer&gt; (1 - 8192)

コンテナに割り当てるコア数。デフォルトではコンテナは利用可能な全コアを使用できます。

--cpulimit &lt;数値&gt; (0 - 8192) (デフォルト= 0)

CPU 使用率の制限。

---

#### 注記

コンピュータに2つのCPUがある場合、合計2のCPU時間を持つ。値0/はCPU制限なしを示す。

---

--cpuunits &lt;integer&gt; (0 - 500000) (デフォルト= cgroup v1: 1024, cgroup v2: 100)

コンテナのCPUウェイト。cgroup v2では[1, 10000]に制限されます。

---

```
--debug &lt;boolean&ampgt (デフォルト= 0)
    より詳細な出力を試みます。現時点では起動時のデバッグログレベルのみ有効化します。

--description &lt;string&ampgt
    コンテナの説明。WebインターフェースのCT概要に表示されます。設定ファイル内にコメントとして保存されます。

--dev[n] [[path=&lt;パス&gt;] [,deny-write=&lt;1|0&gt;] [,gid=&lt;整数&gt;] [,mode=&lt;8進
アクセスモード&gt;] [,uid=&lt;整数&gt;]
    コンテナに渡すデバイス

--features [force_rw_sys=&lt;1|0&gt;] [,fuse=&lt;1|0&gt;][,keyctl=&lt;1|0&gt;]
[,mknod=&lt;1|0&gt;] [,mount=&lt;fstype;fstype;...&gt;] [,nesting=&lt;1|0&gt;]
    コンテナが高度な機能にアクセスすることを許可します。

--force &lt;boolean&ampgt
    既存のコンテナを上書きすることを許可します。

--hookscript &lt;文字列&gt;
    コンテナのライフサイクルの様々な段階で実行されるスクリプト。

--hostname &lt;string&ampgt
    コンテナのホスト名を設定します。

--ignore-unpack-errors &lt;boolean&ampgt
    テンプレートの展開時にエラーを無視します。

--lock &lt;backup| create| destroyed| disk| fstrim| migrate | mounted | rollback |
snapshot | snapshot-delete&gt;
    コンテナのロック/アンロック。

--memory &lt;integer&ampgt (16 - N) (デフォルト= 512)
    コンテナのRAM使用量 (MB単位) 。
```

```
--mp[n] [volume=&lt;ボリューム名&gt; ,mp=&lt;バス名&gt; [,acl=&lt;1|0&gt;]
[,backup=&lt;1|0&gt;] [,mountoptions=&lt;オプション名;オプション名...&gt;]
[,quota=&lt;1|0&gt;][,replicate=&lt;1|0&gt;] [,ro=&lt;1|0&gt;] [,shared=&lt;1|0&gt;]
[,size=&lt;DiskSize&gt;]
    ボリュームをコンテナのマウントポイントとして使用します。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。
```

```
--nameserver &lt;string&gt;
    コンテナのDNSサーバーIPアドレスを設定します。searchdomainもnameserverも設定しない場合、作成時にホストの設定が自動的に使用されます。
```

```
--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=<整数>] [,rate=<Mbps>] [,tag=<整数>]
[,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します。
```

```
--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します。
```

```
--ostype <alpine| archlinux| centos| debian| devuan| fedora | gentoo | nixos | opensuse | ubuntu | unmanaged>;

OSタイプ。コンテナ内の設定を構築するために使用され、/usr/share/lxc/config/<ostype>.common.conf 内の lxc 設定スクリプトに対応します。値 unmanaged を使用すると、OS固有の設定をスキップできます。
```

```
--password <password>

コンテナ内のrootパスワードを設定します。
```

```
--pool <文字列>

指定されたプールにVMを追加します。
```

```
--protection <boolean> (デフォルト= 0)

コンテナの保護フラグを設定します。これにより、CTまたはCTのディスクの削除/更新操作が防止されます。
```

```
--restore <boolean>

これを復元タスクとしてマークします。
```

```
--rootfs [volume=<ボリューム名>] [,acl=<1|0>] [,mountoptions=<オプション名[,オプション名...]>] [,quota=<1|0>][,replicate=<1|0>] [,ro=<1|0>]
[,shared=<1|0>] [,size=<DiskSize>]

ボリュームをコンテナのルートとして使用します。
```

```
--searchdomain <string>

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、作成時にはホストの設定が自動的に使用されます。
```

```
--ssh-public-keys <ファイルパス>

公開SSHキーを設定します（1行に1キー、OpenSSH形式）。
```

```
--start <boolean> (デフォルト= 0)

コンテナの作成が正常に完了した後、CTを起動します。
```

```
--startup `[[order=\d+][,up=\d+][,down=\d+]]`
```

起動およびシャットダウンの動作。Orderは非負の整数で、一般的な起動順序を定義します。シャットダウンは逆順で行われます。さらに、アップまたはダウンの遅延を秒単位で設定でき、次のVMが起動または停止されるまでの待機時間を指定します。

```
--storage <ストレージID> (デフォルト= local)
デフォルトのストレージ。

--swap <整数> (0 ~ N) (デフォルト = 512)
コンテナのSWAP容量 (MB単位)。

--tags <string>;
コンテナのタグ。これはメタ情報のみです。

--template <boolean> (デフォルト= 0)
テンプレートの有効/無効設定。

--timezone <string>;
コンテナで使用するタイムゾーン。設定しない場合は何も行われません。ホストのタイムゾーンに合わせるために「host」に設定するか、/usr/share/zoneinfo/zone.tabからの任意のタイムゾーンオプションを設定できます。

--tty <integer> (0 - 6) (デフォルト = 2)
コンテナが使用できるttyの数を指定します

--unique <boolean>;
一意のランダムなイーサネットアドレスを割り当てます。
```

---

#### 注記

必要なオプション: restore

---

```
--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します。 (手動で変更しないでください。)
```

```
--unused[n] [volume=]<ボリューム>;
未使用ボリュームへの参照。内部で使用されるため、手動で変更しないでください。
```

```
pct delsnapshot <vmid> [<snapname>] [OPTIONS]
```

LXCスナップショットを削除します。

```
<vmid>; <integer> (100 - 999999999)
VMの（一意の）ID。
```

```
<snapname>; <string>;
スナップショットの名前。
```

```
--force <boolean>;
ディスクスナップショットの削除に失敗した場合でも、設定ファイルから削除する。
```

**pct destroy <vmid> [オプション]**

コンテナを破棄します（使用中のファイルもすべて削除します）。

<vmid>; <integer> (100 - 999999999)

VMの（一意の）ID。

**--destroy-unreferenced-disks <boolean>**

設定すると、有効なすべてのストレージから、設定で参照されていないVMIDを持つディスクを追加で破棄します。

**--force <boolean> (デフォルト= 0)**

実行中であっても強制的に破壊を実行します。

**--purge <boolean> (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します。例：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エントリは常に削除されます。

**pct df <vmid>**

コンテナの現在のディスク使用量を取得します。

<vmid>; <integer> (100 - 999999999)

VMの（一意の）ID。

**pct enter <vmid> [オプション]**

指定されたコンテナのシェルを起動します。

<vmid>; <integer> (100 - 999999999)

VMの（一意の）ID。

**--keep-env <布尔値> (デフォルト= 1)**

現在の環境を維持します。このオプションはPVE 9ではデフォルトで無効化されます。事前設定済みの環境に依存している場合は、将来の互換性を確保するためこのオプションを使用してください。

**pct exec <vmid> [<extra-args>] [OPTIONS]**

指定されたコンテナ内でコマンドを実行します。

<vmid>; <整数> (100 - 999999999)

VMの（一意の）ID。

<extra-args>; <array>

追加引数を配列として指定

**--keep-env <boolean> (デフォルト= 1)**

現在の環境を保持します。このオプションはPVE 9以降でデフォルト無効化されます。事前設定環境を利用する場合は、将来の互換性確保のため本オプションを使用してください。

**pct fsck <vmid> [オプション]**

コンテナボリュームに対してファイルシステムチェック (fsck) を実行します。

**<vmid>; 整数 (100 - 999999999)**

VMの（一意の）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 | 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 | 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&gt;
```

ファイルシステムチェックを実行するボリューム

**--force <boolean> (デフォルト= 0)**

ファイルシステムが正常に見える場合でも強制的にチェックを実行

**pct fstrim <vmid> [オプション]**

選択したCTとそのマウントポイント（bindまたはread-onlyマウントポイントを除く）に対してfstrimを実行します。

**&lt;vmid&gt;; &lt;integer&gt; (100 - 999999999)**  
仮想マシンの（一意の）ID。

**--ignore-mountpoints &lt;boolean&gt;**  
すべてのマウントポイントをスキップし、コンテナのルートディレクトリのみに対してfstrimを実行します。

**pct help [オプション]**

指定されたコマンドに関するヘルプを取得します。

**--extra-args &lt;array&gt;**  
特定のコマンドのヘルプを表示します

**--verbose &lt;boolean&gt;**  
詳細出力形式。

**pct list**

LXC コンテナインデックス（ノードごと）。

**pct list snapshot &lt;vmid&gt; すべての**

スナップショットを一覧表示します。

**&lt;vmid&gt;; &lt;integer&gt; (100 - 999999999)**  
VM の（一意の）ID。

**pct migrate &lt;vmid&gt; &lt;target&gt; [OPTIONS]**

コンテナを別のノードに移行します。新しい移行タスクを作成します。

**&lt;vmid&gt;; &lt;整数&gt; (100 - 999999999)**  
仮想マシンの（一意の）ID。

**&lt;target&gt;; &lt;string&gt;**  
ターゲットノード。

**--bwlimit &lt;number&gt; (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移行制限)**  
I/O帯域幅制限（KiB/s単位）を上書きします。

**--online &lt;boolean&gt;**  
オンライン/ライブ移行を使用する。

**--restart &lt;boolean&gt;**  
再起動マイグレーションを使用

**--target-storage <string>**

ソース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ます。特別な値1を指定すると、各ソースストレージがそれ自身にマッピングされます。

**--timeout <integer> (デフォルト値:= 値:180)**

再起動移行のためのシャットダウンタイムアウト (秒)

**pct mount <vmid>;**

コンテナのファイルシステムをホストにマウントします。これによりコンテナにロックがかかり、起動と停止以外の操作が禁止されるため、緊急メンテナンス時のみに使用してください。

**<vmid>: <integer> (100 - 999999999)**

VMの（一意の）ID。

**pct move-volume <vmid> <volume> [<storage>] [<target-vmid>] [<target-volume>] [OPTIONS]**

ルートファイルシステム (/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します。

**<vmid>: <integer> (100 - 999999999)**

仮想マシンの（一意の）ID。

&lt;volume&gt;; &lt;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 | 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 | 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 | 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

移動されるボリューム。

<storage>:<storageID>;

ターゲットストレージ。

<target-vmid>:<integer> (100 - 999999999)

VMの（一意の）ID。

&lt;target-volume&gt;: &lt;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| 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| unused11| unused110| unused111| unused112| unused113| 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| 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

ボリュームが移動される構成キー。デフォルトはソースボリュームキーです。

**--bwlimit <数値> (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帯域幅制限を上書き（単位：KiB/s）。

**--delete <boolean> (デフォルト= 0)**

コピー成功後に元のボリュームを削除します。デフォルトでは元のボリュームは未使用のボリュームエントリとして保持されます。

**--digest <string>**

現在の設定ファイルのSHA1ダイジェストが異なる場合、変更を防止します。これにより同時編集を防止できます。

**--target-digest <string>**

対象の「.」コンテナの現在の設定ファイルが異なるSHA1ダイジェストを持つ場合、変更を防止します。これにより「.」同時編集を防止できます。

#### pct move\_volume

*pct move-volume* の別名。

**pct pending <vmid>**

保留中の変更を含むコンテナ設定を取得します。

**<vmid>: <integer> (100 - 999999999)**

VMの（一意の）ID。

**pct pull <vmid> <path> <destination> [OPTIONS]**

コンテナからローカルシステムへファイルをコピーします。

**<vmid>: <integer> (100 - 999999999)**

仮想マシンの（一意の）ID。

**<path>: <string>**

コンテナ内のブル対象ファイルへのパス。

**<destination>: <string>**

宛先

**--group <string>**

所有者グループ名またはID。

**--perms <string>**

使用するファイル権限（デフォルトは8進数、16進数の場合は0xを接頭辞として付記）。

```
--user <string>;
所有者のユーザー名またはID。
```

```
pct push <vmid>; <file>; <destination>; [OPTIONS]
ローカルファイルをコンテナにコピーします。
```

```
<vmid>; <integer>; (100 - 999999999)
VMの（一意の）ID。
```

```
<file>; <string>;
ローカルファイルへのパス。
```

```
<destination>; <string>;
コンテナ内の書き込み先。
```

```
--group <string>;
所有者グループ名またはID。名前を使用する場合、コンテナ内に存在している必要があります。
```

```
--perms <string>;
使用するファイル権限（デフォルトは8進数、16進数の場合は0xを接頭辞として指定）。
```

```
--user <string>;
所有者ユーザー名またはID。名前を使用する場合、コンテナ内に存在している必要があります。
```

```
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します。保留中の変更を適用します。
```

```
<vmid>; <integer>; (100 - 999999999)
仮想マシンの（一意の）ID。
```

```
--timeout <integer>; (0 - N)
シャットダウンを最大タイムアウト秒間待機します。
```

```
pct remote-migrate <vmid>; [<target-vmid>] <target-endpoint>; --target-bridge <str>
--target-bridge <str>
コンテナをリモートクラスターに移行します。新しい移行タスクを作成します。実験的機能です！
```

```
<vmid>; <integer>; (100 - 999999999)
VMの（一意の）ID。
```

```
<target-vmid>; <integer>; (100 - 999999999)
仮想マシンの（一意の）ID。
```

```
&lt;target-endpoint&ampgt:: apitoken=&lt;PVEAPIToken=user@realm!token=SECRET&ampgt;
,host=&lt;ADDRESS&ampgt [,fingerprint=&lt;FINGERPRINT&ampgt] [,port=&lt;PORT&ampgt]
リモートターゲットエンドポイント

--bwlimit &lt;integer&ampgt (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移行制限)
I/O帯域幅制限を上書き (単位: KiB/s) 。

--delete &lt;boolean&ampgt (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は停止状態でソースクラスター上に保持されます。

--online &lt;boolean&ampgt
オンライン/ライブ移行を使用します。

--restart &lt;boolean&ampgt
再起動移行を使用する

--target-bridge &lt;string&ampgt;
ソースからターゲットへのブリッジのマッピング。単一のブリッジIDのみを指定すると、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自身にマッピングされます。

--target-storage &lt;string&ampgt;
ソース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
れます。特別な値1を指定すると、各ソースストレージがそれ自身にマッピングされます。

--timeout &lt;integer&ampgt (デフォルト= 180)
シャットダウン/再起動移行のタイムアウト (秒単位)

pct rescan [オプション]
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します。

--dryrun &lt;布尔値&ampgt (デフォルト= 180)
設定への変更を実際に書き込まないでください。

--vmid &lt;整数&ampgt (100 - 999999999)
仮想マシンの (一意の) ID。

pct resize &lt;vmid&ampgt &lt;disk&ampgt &lt;size&ampgt [OPTIONS]
コンテナのマウントポイントをリサイズします。

&lt;vmid&ampgt:&lt;integer&ampgt (100 - 999999999)
VMの (一意の) ID。
```

```
&lt;disk&ampgt:&lt;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 | 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&gt;
```

サイズ変更したいディスク。

新しいサイズ。+記号がある場合、値はボリュームの実際のサイズに加算され、ない場合は絶対値として扱われます。ディスクサイズの縮小はサポートされていません。

--digest &lt;string&gt;

現在の設定ファイルのSHA1ダイジェストが異なる場合、変更を防止します。並行編集を防ぐために使用できます。

**pct restore &lt;vmid&gt; &lt;ostemplate&gt; [オプション]**

コンテナの作成または復元を行います。

&lt;vmid&gt;: &lt;integer&gt; (100 - 999999999)

仮想マシンの（一意の）ID。

**&lt;osTemplate&gt;: &lt;string&gt;**

OSテンプレートまたはバックアップファイル。

**--arch &lt;amd64| arm64| armhf| i386| riscv32| riscv64&gt; (デフォルト= amd64)**

OS アーキテクチャタイプ。

**--bwlimit &lt;number&gt; (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 帯域幅制限を上書き (KiB/s 単位)。

**--cmode &lt;console| shell| tty&gt; (デフォルト= tty)**

コンソールモード。デフォルトでは、コンソールコマンドは利用可能なttyデバイスへの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へのアタッチを試みます。cmodeをshellに設定すると、コンテナ内で単純にシェルを起動します（ログインなし）。

**--console &lt;boolean&gt; (デフォルト= 1)**

コンソールデバイス (/dev/console) をコンテナにアタッチします。

**--cores &lt;integer&gt; (1 - 8192)**

コンテナに割り当てるコア数。デフォルトでは利用可能な全コアを使用可能。

**--cpulimit &lt;数値&gt; (0 - 8192) (デフォルト= 0)**

CPU 使用率の制限。

---

#### 注記

コンピュータに2つのCPUがある場合、合計2のCPU時間を持つ。値0はCPU制限なしを示す。

---

**--cpuunits &lt;integer&gt; (0 - 500000) (デフォルト= cgroup v1: 1024, cgroup v2: 100)**

コンテナのCPUウェイト。cgroup v2では[1, 10000]に制限されます。

**--debug &lt;boolean&gt; (デフォルト= 0)**

より詳細な出力を試みます。現時点では起動時のデバッグログレベルのみ有効化します。

**--description &lt;string&gt;**

コンテナの説明。WebインターフェースのCTサマリーに表示されます。設定ファイル内にコメントとして保存されます。

**--dev[n] [[path=&lt;パス&gt;] [,deny-write=&lt;1|0&gt;] [,gid=&lt;整数&gt;] [,mode=&lt;8進  
アクセスモード&gt;] [,uid=&lt;整数&gt;]**

コンテナに渡すデバイス

**--features [force\_rw\_sys=&lt;1|0&gt;] [,fuse=&lt;1|0&gt;] [,keyctl=&lt;1|0&gt;]  
[,mknod=&lt;1|0&gt;] [,mount=&lt;fstype;fstype;...&gt;] [,nesting=&lt;1|0&gt;]**

コンテナが高度な機能にアクセスすることを許可します。

---

```
--force &lt;boolean&gt;
既存のコンテナを上書きすることを許可します。

--hookscript &lt;文字列&gt;
コンテナのライフサイクルの様々な段階で実行されるスクリプト。

--hostname &lt;string&gt;
コンテナのホスト名を設定します。

--ignore-unpack-errors &lt;boolean&gt;
テンプレート展開時のエラーを無視します。

--lock &lt;backup| create| destroyed| disk| fstrim| migrate | mounted | rollback | snapshot | snapshot-delete&gt;
コンテナのロック/アンロック。

--memory &lt;integer&gt; (デフォルト= 512)
コンテナのRAM容量 (MB単位) 。

--mp[n] [volume=&lt;ボリューム&gt; ,mp=&lt;パス&gt; [,acl=&lt;1|0&gt;]
[,backup=&lt;1|0&gt;] [,mountoptions=&lt;オプション [&lt;オプション ...]&gt;]
[,quota=&lt;1|0&gt;][,replicate=&lt;1|0&gt;] [,ro=&lt;1|0&gt;] [,shared=&lt;1|0&gt;]
[,size=&lt;DiskSize&gt;]
ボリュームをコンテナのマウントポイントとして使用します。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。

--nameserver &lt;string&gt;
コンテナのDNSサーバーIPアドレスを設定します。searchdomainもnameserverも設定しない場合、作成時にホストの設定が自動的に使用されます。

--net[n] name=&lt;string&gt; [,bridge=&lt;bridge&gt;] [,firewall=&lt;1|0&gt;]
[,gw=&lt;GatewayIPv4&gt;] [,gw6=&lt;GatewayIPv6&gt;] [,hwaddr=&lt;XX:XX:XX:XX:XX:XX&gt;]
[,ip=&lt;(IPv4/CIDR|dhcp|manual)&gt;] [,ip6=&lt;(IPv6/CIDR|auto|dhcp|manual)&gt;]
[,link_down=&lt;1|0&gt;][,mtu=&lt;整数&gt;] [,rate=&lt;Mbps&gt;] [,tag=&lt;整数&gt;]
[,trunks=&lt;vlanid[:vlanid...]&gt;] [,type=&lt;veth&gt;]
コンテナのネットワークインターフェースを指定します。

--onboot &lt;boolean&gt; (デフォルト= 0)
システムの起動時にコンテナを起動するかどうかを指定します。

--ostype &lt;alpine| archlinux| centos| debian| devuan| fedora | gentoo | nixos | opensuse | ubuntu | unmanaged&gt;
OSタイプ。これはコンテナ内の設定をセットアップするために使用され、/usr/share/lxc/config/&lt;ostype&gt;.common.conf にある lxc セットアップスクリプトに対応します。値 unmanaged を使用すると、OS固有のセットアップをスキップできます。
```

```
--password <password>;
コンテナ内の root パスワードを設定します。

--pool <文字列>;
指定されたプールにVMを追加します。

--protection <boolean> (デフォルト= 0)
コンテナの保護フラグを設定します。これにより、コンテナまたはコンテナのディスクの削除/更新操作が防止されます。

--rootfs [volume=<ボリューム>; [,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=<DiskSize>]
ボリュームをコンテナのルートとして使用します。

--searchdomain <string>;
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、作成時はホストの設定が自動的に使用されます。

--ssh-public-keys <ファイルパス>;
公開SSH鍵の設定（1行に1鍵、OpenSSH形式）。

--start <boolean> (デフォルト= 0)
CTの作成が正常に完了した後、CTを起動します。

--startup `[[order=\d+] [,up=\d+] [,down=\d+]` 
起動およびシャットダウンの動作。Orderは非負の整数で、一般的な起動順序を定義します。シャットダウンは逆順で行われます。さらに、アップまたはダウンの遅延を秒単位で設定でき、次のVMが起動または停止されるまでの待機時間を指定します。

--storage <ストレージID> (デフォルト= local)
デフォルトストレージ。

--swap <整数> (0 - N) (デフォルト= 512)
コンテナのSWAP容量（MB単位）。

--tags <文字列>;
コンテナのタグ。これはメタ情報のみです。

--template <boolean> (デフォルト= 0)
テンプレートの有効化/無効化。

--timezone <string>;
コンテナで使用するタイムゾーン。設定されていない場合、何も実行されません。ホストのタイムゾーンに合わせるために '/host' に設定するか、/usr/share/zoneinfo/zone.tab からの任意のタイムゾーンオプションを設定できます。

--tty <整数> (0 - 6) (デフォルト= 2)
コンテナで使用可能なttyの数を指定します
```

--unique <boolean>  
一意のランダムなイーサネットアドレスを割り当てます。

---

注

必要なオプション: restore

---

--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します。 (手動で変更しないでください。)

--unused[n] [volume=]<ボリューム名>  
未使用ボリュームへの参照。これは内部で使用されるものであり、手動で変更すべきではありません。

pct resume <vmid>;

コンテナを再開します。

<vmid>::<integer> (100 - 999999999)  
VMの（一意の）ID。

pct rollback <vmid>; <snapname> [オプション]

LXC 状態を指定されたスナップショットにロールバックします。

<vmid>::<integer> (100 - 999999999)  
VMの（一意の）ID。

<snapname>::<string>  
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べきかどうか

pct set <vmid>; [オプション]

コンテナオプションを設定します。

<vmid>::<integer> (100 - 999999999)  
VMの（一意の）ID。

--arch <amd64| arm64| armhf| i386| riscv32| riscv64> (デフォルト/ amd64)=  
amd64  
OS アーキテクチャタイプ。

--cmode <console| shell| tty> (デフォルト= tty)

コンソールモード。デフォルトでは、コンソールコマンドは利用可能なttyデバイスへの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へのアタッチを試みます。cmodeをshellに設定すると、コンテナ内で単純にシェルを起動します（ログインなし）。

---

```
--console &lt;boolean&ampgt (デフォルト= 1)
コンソールデバイス (/dev/console) をコンテナにアタッチします。

--cores &lt;integer&ampgt (1 - 8192)
コンテナに割り当てるコア数。デフォルトではコンテナは利用可能な全コアを使用できます。

--cpulimit &lt;number&ampgt (0 - 8192) (デフォルト= 0)
CPU 使用率の制限。
```

---

#### 注記

コンピュータに2つのCPUがある場合、合計2のCPU時間を持つ。値0はCPU制限なしを示す。

---

```
--cpuunits &lt;integer&ampgt (0 - 500000) (デフォルト= cgroup v1: 1024, cgroup v2: 100)
コンテナのCPUウェイト。cgroup v2では[1, 10000]に制限されます。

--debug &lt;boolean&ampgt (デフォルト= 0)
より詳細なログを出力します。現時点では起動時のデバッグログレベルのみ有効化します。
```

```
--delete &lt;文字列&ampgt;
削除したい設定の一覧。
```

```
--description &lt;string&ampgt;
コンテナの説明。WebインターフェースのCT概要に表示されます。設定ファイル内にコメントとして保存されます。
```

```
--dev[n] [[path=&lt;パス&gt;] [,deny-write=&lt;1|0&gt;] [,gid=&lt;整数&gt;] [,mode=&lt;8進
アクセスモード&gt;] [,uid=&lt;整数&gt;]
コンテナに渡すデバイス
```

```
--digest &lt;string&ampgt;
現在の設定ファイルのSHA1ダイジェストが異なる場合、変更を防止します。これにより同時変更を防止できます。
```

```
--features [force_rw_sys=&lt;1|0&gt;] [,fuse=&lt;1|0&gt;][,keyctl=&lt;1|0&gt;]
[,mknod=&lt;1|0&gt;] [,mount=&lt;fstype;fstype;...&gt;] [,nesting=&lt;1|0&gt;]
コンテナが高度な機能にアクセスできるようにします。
```

```
--hookscript &lt;string&ampgt;
コンテナのライフサイクルにおける様々な段階で実行されるスクリプト。
```

```
--hostname &lt;string&ampgt;
コンテナのホスト名を設定します。
```

```
--lock <backup| create| destroyed| disk| fstrim| migrate | mounted | rollback | snapshot | snapshot-delete>;
コンテナのロック/アンロック。
```

```
--memory <integer> (デフォルト= 512)
コンテナのRAM使用量 (MB単位)。
```

```
--mp[n] [volume=<ボリューム>, mp=<パス>, [acl=<1|0>]
[,backup=<1|0>] [,mountoptions=<オプション [&lt;オプション...]>]
[,quota=<1|0>][,replicate=<1|0>] [,ro=<1|0>] [,shared=<1|0>]
[,size=<DiskSize>]
ボリュームをコンテナのマウントポイントとして使用します。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。
```

```
--nameserver <string>
コンテナのDNSサーバーIPアドレスを設定します。searchdomainもnameserverも設定しない場合、作成時にホストの設定が自動的に使用されます。
```

```
--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=<整数>] [,rate=<Mbps>] [,tag=<整数>]
[,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します。
```

```
--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します。
```

```
--ostype <alpine| archlinux| centos| debian| devuan| fedora | gentoo | nixos | opensuse | ubuntu
| unmanaged>;
OSタイプ。コンテナ内の設定を構築するために使用され、/usr/share/lxc/config/<ostype>.common.conf 内の lxc 設定スクリプトに対応します。値
unmanaged を使用すると、OS固有の設定をスキップできます。
```

```
--protection <boolean> (デフォルト= 0)
コンテナの保護フラグを設定します。これにより、CTまたはCTのディスクの削除/更新操作が防止されます。
```

```
--revert <文字列>;
保留中の変更を元に戻します。
```

```
--rootfs [volume=<ボリューム> [,acl=<1|0>] [,mountoptions=<オプション [&lt;オプション...]>]
[,quota=<1|0>][,replicate=<1|0>] [,ro=<1|0>]
[,shared=<1|0>][,size=<DiskSize>]
ボリュームをコンテナのルートとして使用します。
```

**--searchdomain &lt;string&gt;**

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、作成時にはホストの設定が自動的に使用されます。

**--startup [[order=]\d+] [up=\d+] [down=\d+]**

起動およびシャットダウン時の動作を定義します。orderは起動順序を定義する非負の数値です。シャットダウン時は逆順で処理されます。さらに、次のVMの起動または停止前に待機する遅延時間を秒単位で設定できます（upまたはdown遅延）。

**--swap &lt;integer&gt; (0 - N) (デフォルト = 512)**

コンテナのSWAP容量（MB単位）。

**--tags &lt;string&gt;**

コンテナのタグ。これはメタ情報のみです。

**--template &lt;boolean&gt; (デフォルト = 0)**

テンプレートの有効/無効設定。

**--timezone &lt;string&gt;**

コンテナで使用するタイムゾーン。設定しない場合は何も行われません。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からの任意のタイムゾーンオプションを設定できます。

**--tty &lt;整数&gt; (0 - 6) (デフォルト = 2)**

コンテナが使用できるttyの数を指定します

**--unprivileged &lt;boolean&gt; (デフォルト = 0)**

コンテナを非特権ユーザーとして実行します。（手動で変更すべきではありません。）

**--unused[n] [volume=] &lt;volume&gt;**

未使用ボリュームへの参照。内部で使用されるため、手動で変更しないでください。

**pct shutdown &lt;vmid&gt; [オプション]**

コンテナをシャットダウンします。コンテナのクリーンなシャットダウンをトリガーします。詳細はlxc-stop(1)を参照してください。

**&lt;vmid&gt;; &lt;integer&gt; (100 - 999999999)**

仮想マシンの（一意の）ID。

**--forceStop &lt;boolean&gt; (デフォルト = 0)**

コンテナを確実に停止させます。

**--timeout &lt;integer&gt; (0 - N) (デフォルト = 60)**

最大タイムアウト秒間待機します。

**pct snapshot &lt;vmid&gt; &lt;snapname&gt; [オプション]**

コンテナのスナップショットを作成します。

```
&lt;vmid&ampgt; &lt;integer&ampgt (100 - 999999999)  
VMの（一意の）ID。
```

```
&lt;snapname&ampgt; &lt;string&ampgt  
スナップショットの名前。
```

```
--description &lt;string&ampgt  
テキストによる説明またはコメント。
```

```
pct start &lt;vmid&ampgt [オプション]
```

コンテナを起動します。

```
&lt;vmid&ampgt; &lt;整数&ampgt (100 - 999999999)  
仮想マシンの（一意の）ID。
```

```
--debug &lt;boolean&ampgt (デフォルト= 0)  
設定すると、起動時に非常に詳細なデバッグログレベルを有効にします。
```

```
--skiplock &lt;boolean&ampgt  
ロックを無視する - このオプションはrootのみが使用可能。
```

```
pct status &lt;vmid&ampgt [オプション]
```

CTステータスを表示します。

```
&lt;vmid&ampgt; &lt;integer&ampgt (100 - 999999999)  
VMの（一意の）ID。
```

```
--verbose &lt;boolean&ampgt  
詳細出力形式
```

```
pct stop &lt;vmid&ampgt [オプション]
```

コンテナを停止します。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します。

```
&lt;vmid&ampgt; &lt;整数&ampgt (100 - 999999999)  
VMの（一意の）ID。
```

```
--overrule-shutdown &lt;boolean&ampgt (デフォルト= 0)  
停止前にアクティブなvzshutdownタスクを中止しようと試みます。
```

```
--skiplock &lt;boolean&ampgt  
ロックを無視する - rootのみがこのオプションを使用できる。
```

```
pct suspend &lt;vmid&ampgt
```

コンテナをサスペンドします。これは実験的な機能です。

```
&lt;vmid&ampgt; &lt;integer&ampgt (100 - 999999999)  
VMの（一意の）ID。
```

```
pct template &lt;vmid&ampgt;
```

テンプレートを作成します。

```
&lt;vmid&ampgt; &lt;integer&ampgt (100 - 999999999)  
仮想マシンの（一意の）ID。
```

```
pct unlock &lt;vmid&ampgt;
```

仮想マシンをアンロックします。

```
&lt;vmid&ampgt; &lt;integer&ampgt (100 - 999999999)  
仮想マシンの（一意の）ID。
```

```
pct unmount &lt;vmid&ampgt;
```

コンテナのファイルシステムをアンマウントします。

```
&lt;vmid&ampgt; &lt;integer&ampgt (100 - 999999999)  
仮想マシンの（一意の）ID。
```

## A.12 pveam - Proxmox VE アプライアンス マネージャー

```
pveam &lt;コマンド&ampgt [引数] [オプション]
```

```
pveam available [オプション]
```

利用可能なテンプレートを一覧表示します。

```
--section &lt;mail| system| turnkeylinux&ampgt;  
リストを指定セクションに制限する。
```

```
pveam download &lt;storage&ampgt; &lt;template&ampgt;
```

アプライアンステンプレートをダウンロードします。

```
&lt;storage&ampgt; &lt;storage ID&ampgt;  
テンプレートが保存されるストレージ
```

```
&lt;template&ampgt; &lt;string&ampgt;  
ダウンロードするテンプレート
```

**pveam help [オプション]**

指定されたコマンドの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します

**--verbose <boolean>**

詳細出力形式。

**pveam list <storage>;**

ストレージ上の全テンプレートのリストを取得

**<storage>; <storage ID>;**

指定されたストレージ上のテンプレートのみを一覧表示

**pveam remove <テンプレートパス>;**

テンプレートを削除します。

**<template\_path>; <string>;**

削除するテンプレート。

**pveam update**

コンテナテンプレートデータベースを更新します。

## A.13 pvecm - Proxmox VE Cluster Manager

**pvecm <コマンド> [<引数>] [<オプション>]**

**pvecm add <ホスト名> [<オプション>]**

現在のノードを既存のクラスターに追加します。

**<ホスト名>; <文字列>;**

既存クラスターメンバーのホスト名（またはIPアドレス）。

証明書のSHA256フィンガープリント。

**--force <boolean>;**

ノードが既に存在する場合でもエラーを発生させない。

**--link[n] [<address=>] <IP> [,priority=<integer>];**

単一のCorosyncリンクのアドレスと優先度情報（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 <boolean>;

常にSSHを使用して参加する（相手側がAPI経由で参加する場合でも）。

--votes <整数> (0 - N)

このノードの投票数

**pvecm addnode** <node> [OPTIONS]

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。

<node>: <string>;

クラスタノード名。

--apiversion <integer>;

新規ノードの JOIN\_API\_VERSION。

--force <boolean>;

ノードが既に存在する場合でもエラーを発生させない。

--link[n] [address=<IP> [,priority=<integer>]]

単一のcorosyncリンクのアドレスと優先度情報。（最大8リンクまでサポート；link0..link7）

--new\_node\_ip <string>;

追加するノードのIPアドレス。リンクが指定されていない場合のフォールバックとして使用されます。

--nodeid <integer> (1 - N)

このノードのノードID。

--votes <integer> (0 - N)

このノードの投票数

**pvecm apiver**

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** <クラスタ名> [オプション]

新しいクラスター構成を生成します。リンクが指定されていない場合、link0としてローカルIPアドレスがデフォルトで設定されます。

<clusternname>: <string>;

クラスタの名前。

--link[n] [address=<IP> [,priority=<integer>]]

単一の Corosync リンクのアドレスと優先度情報。（最大 8 リンクまでサポート；link0..link7）

--nodeid <integer> (1 - N)

このノードのノードID。

--votes <integer> (1 - N)

このノードへの投票数。

**pvecm delnode** <node>;

クラスタ構成からノードを削除します。

<node>: <string>;

クラスタノード名。

**pvecm expected** <expected>;

corosync に期待票数の新しい値を通知します。

<expected>: <integer> (1 - N)

期待票数

**pvecm help** [オプション]

指定されたコマンドのヘルプを取得します。

--extra-args <array>;

特定のコマンドのヘルプを表示します

--verbose <boolean>;

詳細出力形式。

**pvecm keygen** <filename>;

corosync用の新しい暗号鍵を生成します。

<filename>: <string>;

出力ファイル名

**pvecm mtunnel** [<追加引数>] [オプション]

VM/CT移行で使用されます - 手動では使用しないでください。

<extra-args>: <array>;

配列形式の追加引数

--get\_migration\_ip <boolean> (デフォルト= 0)

設定されている場合、移行用IPアドレスを返す

--migration\_network <string>;

ローカル移行IPを検出するために使用する移行ネットワーク

--run-command <boolean>;

TCPソケットを標準入力としてコマンドを実行します。IPアドレスとポート番号は、このコマンドの標準出力を経由してコマンドの標準出力で、それぞれ別々の行に最初に表示される。

#### 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

設定済みのQDeviceを削除します

##### pvecm qdevice setup <アドレス> [オプション]

QDeviceの使用を設定します

<address>: <string>;

外部 Corosync QDevice のネットワークアドレスを指定します

--force <boolean>;

危険な操作の可能性があってもエラーをスローしない。

--network <string>;

外部 qdevice に接続するために使用するネットワーク

#### pvecm status

クラスタ状態のローカルビューを表示します。

##### pvecm updatecerts [オプション]

ノード証明書を更新する（および必要なすべてのファイル/ディレクトリを生成する）。

--force <boolean>;

新しいSSL証明書の生成を強制する。

--silent <boolean>;

エラーを無視する（例：クラスタにクォーラムがない場合）。

--unmerge-known-hosts <boolean> (デフォルト= 0)

レガシSSH既知ホストの統合を解除します。

## A.14 pvesr - Proxmox VE ストレージレプリケーション

#### pvesr <コマンド> [引数] [オプション]

##### pvesr create-local-job <id> <target> [OPTIONS]

新しいレプリケーションジョブを作成します

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。例：  
&lt;guest>-&lt;jobnum>。

**&lt;target&gt;**: &lt;string&gt;

ターゲットノード。

**--comment &lt;string&gt;**

説明。

**--disable &lt;boolean&gt;**

エントリを無効化/非アクティブ化するフラグ。

**--rate &lt;number&gt; (1 - N)**

浮動小数点数で指定するbps単位のレート制限（メガバイト毎秒）。

**--remove\_job &lt;完全ローカル&gt;**

レプリケーションジョブを削除対象としてマークします。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。フルに設定した場合、ターゲット上のレプリケートされたボリュームの削除も試みます。その後、ジョブは設定ファイルから自身を削除します。

**--schedule &lt;string&gt; (デフォルト= \*/15)**

ストレージレプリケーションスケジュール。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。

**--source &lt;string&gt;**

内部使用。ゲストが盗まれたかどうかを検出するため。

**pvesr delete &lt;id&gt; [オプション]**

レプリケーションジョブを削除対象としてマークします。

レプリケーションジョブID。IDはゲストIDとジョブ番号をハイフンで区切った形式です。例：  
&lt;ゲストID>-&lt;ジョブ番号>。

**--force &lt;boolean&gt; (デフォルト= 0)**

ジョブ設定エントリを削除しますが、クリーンアップは行いません。

**--keep &lt;boolean&gt; (デフォルト= 0)**

ターゲットに複製データを保持（削除しない）。

**pvesr disable &lt;id&gt;**

レプリケーションジョブを無効化します。

レプリケーションジョブID。IDはゲストIDとジョブ番号をハイフンで区切った形式です。例：  
&lt;guest>-&lt;jobnum>。

```
pvesr enable <id>;
```

レプリケーションジョブを有効化します。

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。例：

```
<guest>-<jobnum>.
```

```
pvesr finalize-local-job <id> [<extra-args>] [OPTIONS]
```

レプリケーションジョブを終了します。これにより、`<last_sync>`以外の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。

レプリケーションジョブID。IDはゲストIDとジョブ番号をハイフンで区切ったもので構成されます。例：

```
<guest>-<jobnum>.
```

```
<extra-args>: <array>;
```

考慮すべきボリュームIDのリスト。

```
--last_sync <整数> (0 - N)
```

最後の正常な同期の時刻（UNIXエポック）。指定しない場合、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る。

```
pvesr help [オプション]
```

指定されたコマンドに関するヘルプを取得します。

```
--extra-args <配列>;
```

特定のコマンドのヘルプを表示します

```
--verbose <boolean>;
```

詳細出力形式。

```
pvesr list
```

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を行います。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使用であり、標準出力にJSONオブジェクトを返します。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します。その後、メソッドはスナップショット用の全ボリュームIDをスキャンし、`<last_sync>`と異なるタイムスタンプを持つレプリケーションスナップショットを全て削除します。未使用的ボリュームも同様に削除します。既存のレプリケーションスナップショットを持つ全ボリュームの布尔値マーカーを含むハッシュを返します。

レプリケーションジョブID。IDはゲストIDとジョブ番号をハイフンで区切った形式です。例：

```
<guest>-<jobnum>.
```

```
<extra-args>: <array>;
```

考慮すべきボリュームIDのリスト。

--force <boolean> (デフォルト= 0)

既存のボリュームをすべて削除することを許可（空のボリュームリスト）。

--last\_sync <integer> (0 - N)

最後の正常な同期の時刻（UNIXエポック）。指定しない場合、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る。

--parent\_snapname <string>;

スナップショットの名前。

--scan <文字列>;

古いボリュームをスキャンするストレージIDのリスト。

**pvesr read** <id>;

レプリケーションジョブの設定を読み込みます。

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。例：

&lt;GUEST&gt;-&lt;JOBNUM&gt;。

**pvesr run** [オプション]

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、ハイフンで区切られたゲスト ID とジョブ番号で構成されます。つまり、

&lt;GUEST&gt;-&lt;JOBNUM&gt;。

--mail <boolean> (デフォルト= 0)

障害発生時にメール通知を送信する。

--verbose <boolean> (デフォルト= 0)

より詳細なログを標準出力に表示します。

**pvesr schedule-now** <id>;

レプリケーションジョブを可能な限り早く開始するようにスケジュールします。

レプリケーションジョブID。IDはゲストIDとジョブ番号をハイフンで区切った形式です。例:

&lt;GUEST&gt;-&lt;JOBNUM&gt;。

**pvesr set-state** <vmid>; <state>;

移行時のジョブレプリケーション状態を設定します。この呼び出しあは内部使用です。ジョブ状態をJSONオブジェクトとして受け付けます。

**&lt;vmid&gt;; &lt;integer&gt;** (100 - 999999999)

仮想マシンの（一意の）ID。

**&lt;state&gt;; &lt;string&gt;**

JSONデコードされた文字列としてのジョブ状態。

### pvesr status [オプション]

このノード上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。

**--guest &lt;integer&gt;** (100 - 999999999)

このゲストのレプリケーションジョブのみを一覧表示します。

### pvesr update &lt;id&gt; [オプション]

レプリケーションジョブの構成を更新します。

レプリケーションジョブID。IDはゲストIDとジョブ番号をハイフンで区切ったもので構成されます。例:

*&lt;GUEST&gt;-&lt;JOBNUM&gt;.*

**--comment &lt;string&gt;**

説明。

**--delete &lt;string&gt;**

削除したい設定の一覧。

**--digest &lt;string&gt;**

現在の設定ファイルのダイジェストが異なる場合、変更を防止します。これにより同時編集を防止できます。

**--disable &lt;boolean&gt;**

エントリを無効化/停止するフラグ。

**--rate &lt;数値&gt;** (1 ~ N)

浮動小数点数で指定するbps単位のレート制限（メガバイト毎秒）。

**--remove\_job &lt;完全|ローカル&gt;**

レプリケーションジョブを削除対象としてマークします。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。フルに設定された場合、ターゲット上のレプリケートされたボリュームの削除も試みます。その後、ジョブは設定ファイルから自身を削除します。

**--schedule &lt;文字列&gt;** (デフォルト=\* /15)

ストレージレプリケーションスケジュール。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。

**--source &lt;string&gt;**

内部使用のため、ゲストが盗まれたかどうかを検出します。

## A.15 pveum - Proxmox VE ユーザーマネージャー

**pveum &lt;コマンド&gt; [引数] [オプション]**

**pveum acl delete &lt;パス&gt; --roles &lt;文字列&gt; [オプション]**

アクセス制御リストを更新（権限の追加または削除）。

**&lt;path&gt;: &lt;string&gt;**

アクセス制御対象パス

**--groups &lt;string&gt;**

グループのリスト。

**--propagate &lt;boolean&gt; (デフォルト= 1)**

権限の伝播（継承）を許可します。

**--roles &lt;string&gt;**

ロールのリスト。

**--tokens &lt;string&gt;**

APIトークンのリスト。

**--users &lt;string&gt;**

ユーザーの一覧。

**pveum acl list [ フォーマットオプション]**

アクセス制御リスト（ACL）を取得します。

**pveum acl modify &lt;path&gt; --roles &lt;string&gt; [OPTIONS]**

アクセス制御リストを更新（権限の追加または削除）。

**&lt;path&gt;: &lt;string&gt;**

アクセス制御対象パス

**--groups &lt;string&gt;**

グループのリスト。

**--propagate &lt;boolean&gt; (デフォルト= 1)**

権限の伝播（継承）を許可します。

**--roles &lt;string&gt;**

ロールのリスト。

**--tokens &lt;string&gt;**

APIトークンのリスト。

--users <string>;

ユーザーのリスト。

**pveum acldel**

*pveum acl delete* の別名。

**pveum aclmod**

*pveum acl modify* の別名。

**pveum group add** <groupid> [オプション]

新しいグループを作成します。

<groupid>: <string>;

説明なし

--comment <string>;

説明は利用できません

**pveum group delete** <groupid>;

グループを削除します。

<groupid>: <string>;

説明はありません

**pveum group list** [ フォーマットオプション ]

グループインデックス。

**pveum group modify** <groupid> [OPTIONS]

グループデータを更新します。

<groupid>: <string>;

説明なし

--comment <string>;

説明はありません

**pveum groupadd**

*pveum group add* の別名。

**pveum groupdel**

*pveum group delete* の別名。

**pveum groupmod**

*pveum group modify* の別名。

**pveum help** [オプション]

指定されたコマンドに関するヘルプを表示します。

--extra-args <配列>;  
特定のコマンドのヘルプを表示します

--verbose <boolean>;  
詳細出力形式。

**pveum passwd** <userid> [OPTIONS]

ユーザー パスワードを変更します。

<userid>; <string>;  
完全なユーザーID (name@realm 形式)。

--confirmation-password <文字列>;  
変更を行うユーザーの現在のパスワード。

**pveum pool add** <poolid> [オプション]

新しいプールを作成します。

<poolid>; <string>;  
説明なし

--comment <string>;  
説明は利用できません

**pveum** プール削除 <poolid>;

プールを削除します。

<poolid>; <string>;  
説明はありません

**pveum pool list** [オプション] [フォーマットオプション]

プールの一覧表示またはプール設定の取得。

--poolid <string>;  
説明はありません

--type <lxc|qemu|storage>;  
説明なし

---

#### 注記

必要なオプション: poolid

---

```
pveum pool modify <poolid> [OPTIONS]
```

プールを更新します。

```
<poolid>: <string>;
```

説明なし

```
--allow-move <boolean> (デフォルト= 0)
```

他のプールに既に存在するゲストの追加を許可します。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。

```
--comment <string>;
```

説明なし

```
--delete <boolean> (デフォルト= 0)
```

渡されたVMIDおよび/またはストレージIDを追加する代わりに削除してください。

```
--storage <string>;
```

このプールに追加または削除するストレージIDのリスト。

```
--vms <string>;
```

このプールに追加または削除するゲストVMIDのリスト。

```
pveum realm add <realm> --type <string> [OPTIONS]
```

認証サーバーを追加します。

```
<realm>: <string>;
```

認証ドメインID

```
--acr-values ^[^\x00-\x1F\x7F &gt;#"]*$
```

認証コンテキストクラス参照値を指定します。これは、認証サーバーが認証リクエストに使用するよう要求される値です。

認証要求に使用するよう要求されている認証コンテキストクラス参照値を指定します。

```
--autocreate <boolean> (デフォルト= 0)
```

ユーザーが存在しない場合に自動的に作成する。

```
--base_dn <string>;
```

LDAP ベースドメイン名

```
--bind_dn <string>;
```

LDAP/バインドドメイン名

```
--capath <string> (デフォルト= /etc/ssl/certs)
```

CA証明書ストアへのパス

```
--case-sensitive <boolean> (デフォルト= 1)
ユーザー名は大文字小文字を区別する

--cert <string>
クライアント証明書のパス

--certkey <string>
クライアント証明書キーのパス

--check-connection <boolean> (デフォルト= 0)
サーバーへのバインド接続を確認する。

--client-id <string>
OpenID クライアント ID

--client-key <string>
OpenID クライアントキー

--comment <string>
説明。

--default <boolean>
これをデフォルトのレルムとして使用

--domain \S+
AD ドメイン名

--filter <string>
ユーザー同期用のLDAPフィルター

--group_classes <string> (デフォルト= groupOfNames, group, univentionGroup,
ipausergroup)
グループ用のオブジェクトクラス。

--group_dn <string>
グループ同期用のLDAPベースドメイン名。設定しない場合、base_dnが使用されます。

--group_filter <string>
グループ同期用のLDAPフィルター。

--group_name_attr <string>
LDAP属性でグループ名を表す。設定されていないか見つからない場合、DNの最初の値が名前として使用される。

--groups-autocreate <boolean> (デフォルト= 0)
グループが存在しない場合に自動的に作成します。
```

グループを取得するために使用するOpenIDクレーム。

**--groups-overwrite &lt;boolean&gt;** (デフォルト= 0)

ログイン時にユーザーの全グループを上書きします。

**--issuer-url &lt;string&gt;**

OpenID発行元URL

**--mode &lt;ldap| ldap+starttls| ldaps&gt;** (デフォルト= ldap)

LDAPプロトコルモード。

**--password &lt;文字列&gt;**

LDAP/ BINDパスワード。/etc/pve/priv/realm/&lt;REALM&gt;.pw に保存されます。

**--port &lt;integer&gt;** (1 - 65535)

サーバーポート。

**--prompt (?::none|login|consent|select\_account|\S+)**

認証サーバーがエンドユーザーに再認証と同意を求めるかどうかを指定します。

**--query userinfo &lt;boolean&gt;** (デフォルト= 1)

クレーム値を取得するためにuserinfoエンドポイントへのクエリを有効化します。

**--scopes &lt;string&gt;** (デフォルト= email profile)

承認および返却されるべきスコープ（ユーザー詳細）を指定します。例：email または profile。

**--secure &lt;boolean&gt;**

安全なLDAPSプロトコルを使用してください。非推奨: 代わりにモードを使用してください。

**--server1 &lt;string&gt;**

サーバーのIPアドレス（またはDNS名）

**--server2 &lt;string&gt;**

フォールバックサーバーのIPアドレス（またはDNS名）

**--sslversion &lt;tlsv1| tlsv1\_1| tlsv1\_2| tlsv1\_3&gt;**

LDAPS TLS/SSL バージョン。1.2より古いバージョンの使用は推奨されません！

**--sync-defaults-options [enable-new=&lt;1|0&gt;] [,full=&lt;1|0&gt;] [,purge=&lt;1|0&gt;] [,remove-vanished=([acl];[properties];[entry])|none] [,scope=&lt;users|groups|both&gt;]**

同期動作のデフォルトオプション。

```
--sync_attributes \w+=[^,]+(,\s* \w+=[^,]+)*
LDAP属性とPVEユーザーのマッピングを指定するためのキー=値ペアのカンマ区切りリスト
フィールド。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属性で表現されます。
```

```
--tfa type=<TFATYPE> [,digits=<COUNT>] [,id=<ID>] [,key=<KEY>]
[,step=<SECONDS>] [,url=<URL>]
二要素認証を使用します。
```

```
--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> (デフォルト= inetorgperson, posixaccount, person, user)
ユーザーに対するオブジェクトクラス
```

```
--username-claim <string>;
一意のユーザー名生成に使用するOpenIDクレーム。
```

```
--verify <boolean> (デフォルト= 0)
サーバーのSSL証明書を検証します
```

```
pveum realm delete <realm>;
認証サーバーを削除します。
```

```
<realm>; <string>;
認証ドメインID
```

```
pveum レルム一覧 [ フォーマットオプション]
認証ドメインインデックス。
```

```
pveum realm modify <realm> [OPTIONS]
認証サーバー設定を更新します。
```

```
<realm>; <string>;
認証ドメインID
```

```
--acr-values ^[\x00-\x1F\x7F <gt;#"]*$
認証コンテキストクラス参照値を指定します。これは認証サーバーが
認証要求に使用するよう要求されている認証コンテキストクラス参照値を指定します。
```

```
--autocreate <boolean> (デフォルト= 0)
ユーザーが存在しない場合に自動的に作成します。
```

```
--base_dn <string>;
LDAPベースドメイン名

--bind_dn <string>;
LDAPバインドドメイン名

--capath <string> (デフォルト= /etc/ssl/certs)
CA証明書ストアへのパス

--case-sensitive <boolean> (デフォルト= 1)
ユーザー名は大文字小文字を区別する

--cert <string>;
クライアント証明書のパス

--certkey <string>;
クライアント証明書キーのパス

--check-connection <boolean> (デフォルト= 0)
サーバーへのバインド接続を確認する。

--client-id <string>;
OpenIDクライアントID

--client-key <string>;
OpenIDクライアントキー

--comment <string>;
説明。

--default <boolean>;
これをデフォルトのレルムとして使用

--delete <string>;
削除したい設定のリスト。

--digest <string>;
現在の設定ファイルのダイジェストが異なる場合、変更を防止します。これにより同時変更を防止できます。

--domain \S+
AD ドメイン名

--filter <string>;
ユーザー同期用のLDAPフィルター。
```

```
--group_classes &lt;string&gt; (デフォルト= groupOfNames, group, univentionGroup, ipausergroup)
```

グループ用のオブジェクトクラス。

```
--group_dn &lt;string&gt;
```

グループ同期用のLDAPベースドメイン名。設定しない場合、base\_dnが使用されます。

```
--group_filter &lt;string&gt;
```

グループ同期用のLDAPフィルター。

```
--group_name_attr &lt;string&gt;
```

LDAP属性でグループ名を表す。設定されていないか見つからない場合、DNの最初の値が名前として使用される。

```
--groups-autocreate &lt;boolean&gt; (デフォルト= 0)
```

グループが存在しない場合に自動的に作成します。

グループを取得するために使用するOpenIDクレーム。

```
--groups-overwrite &lt;boolean&gt; (デフォルト= 0)
```

ログイン時にユーザーの全グループを上書きします。

```
--issuer-url &lt;string&gt;
```

OpenID発行元URL

```
--mode &lt;ldap| ldap+starttls| ldaps&gt; (デフォルト= ldap)
```

LDAPプロトコルモード。

```
--password &lt;文字列&gt;
```

LDAPバインドパスワード。/etc/pve/priv/realm/&lt;REALM&gt;.pw に保存されます。

```
--port &lt;integer&gt; (1 - 65535)
```

サーバーポート。

```
--prompt (?::none|login|consent|select_account|\S+)
```

認証サーバーがエンドユーザーに再認証と同意を求めるかどうかを指定します。

```
--query-userinfo &lt;boolean&gt; (デフォルト= 1)
```

クレーム値を取得するためにuserinfoエンドポイントへのクエリを有効化します。

```
--scopes &lt;string&gt; (デフォルト= email profile)
```

承認および返却すべきスコープ（ユーザー詳細）を指定します。例：email または profile。

```
--secure &lt;boolean&gt;
```

安全なLDAPSプロトコルを使用してください。非推奨：代わりにモードを使用してください。

```
--server1 &lt;文字列&gt;
サーバーのIPアドレス（またはDNS名）

--server2 &lt;string&gt;
フォールバックサーバーのIPアドレス（またはDNS名）

--sslversion &lt;tls1| tls1_1| tls1_2| tls1_3&gt;
LDAPS TLS/SSL バージョン。1.2 より古いバージョンの使用は推奨されません！

--sync-defaults-options [enable-new=&lt;1|0&gt;] [,full=&lt;1|0&gt;] [,purge=&lt;1|0&gt;]
[,remove-vanished=([acl];[properties];[entry])|none] [,scope=&lt;users|groups|both&gt;]
同期動作のデフォルトオプション。

--sync_attributes \w+=[^,]+(,\s*\ \w+=[^,]+)*
LDAP属性とPVEユーザーのマッピングを指定するためのキー=値ペアのカンマ区切りリスト
フィールド。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属性で表現されます。

--tfa type=&lt;TFATYPE&gt; [,digits=&lt;COUNT&gt;] [,id=&lt;ID&gt;] [,key=&lt;KEY&gt;]
[,step=&lt;SECONDS&gt;] [,url=&lt;URL&gt;]
二要素認証を使用します。

--user_attr \S{2,}
LDAPユーザー属性名

--user_classes &lt;string&gt; (デフォルト= inetorgperson, posixaccount, person, user)
ユーザーのオブジェクトクラス。

--verify &lt;boolean&gt; (デフォルト= 0)
サーバーのSSL証明書を検証

pveum realm sync &lt;realm&gt; [OPTIONS]
設定済みのLDAPからユーザーおよび/またはグループをuser.cfgに同期します。注記: 同期されたグループの名前は
name-$realm、上書きを防ぐため、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。
```

```
&lt;realm&gt;; &lt;string&gt;;
認証ドメインID

--dry-run &lt;boolean&gt; (デフォルト= 0)
設定すると、何も書き込まない。

--enable-new &lt;boolean&gt; (デフォルト= 0)
新規同期ユーザーを即時有効化します。
```

**--full <布尔值>;**

非推奨: 代わりに `remove-vanished` を使用してください。設定すると、LDAPディレクトリを真実のソースとして使用し、同期から返されなかったユーザー やグループを削除し、同期されたユーザーのローカルで変更されたプロパティをすべて削除します。設定しない場合、同期されたデータに存在する情報のみを同期し、他のものは削除または変更しません。

**--purge <boolean>;**

非推奨: 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ーまたはグループの ACL を削除します。

**--remove-vanished ([acl] ; [properties] ; [entry]) | none (デフォルト = none)**

同期中にユーザーまたは対象が消失した場合に削除する項目のセミコロン区切りリスト。以下の値が指定可能です：`entry` は同期から返されなかったユーザー/グループを削除します。`properties` はソースに存在しない既存ユーザー/グループの設定プロパティ（カスタムプロパティも含む）を削除します。`acl` は同期から返されなかったユーザー/グループのアクセス制御リスト（ACL）を削除します。リストの代わりに `none`（デフォルト）を指定することも可能です。

**--scope <both| groups| users>;**

同期対象を選択します。

**pveum role add <roleid> [OPTIONS]**

新しいロールを作成します。

**<roleid>: <string>;**

説明なし

**--privs <string>;**

説明なし

**pveum role delete <roleid>;**

ロールを削除します。

**<roleid>: <string>;**

説明なし

**pveum role list [ フォーマットオプション ]**

ロールインデックス。

**pveum role modify <roleid> [OPTIONS]**

既存のロールを更新します。

**<roleid>: <string>;**

説明なし

```
--append &lt;boolean&gt;  
説明はありません
```

---

#### 注記

必要なオプション: privs

---

```
--privs &lt;文字列&gt;  
説明はありません
```

**pveum roleadd**

*pveum role add* の別名。

**pveum roledel**

*pveum role delete* の別名。

**pveum rolemod**

*pveum role modify* の別名。

**pveum ticket &lt;username&gt; [OPTIONS]**

認証チケットの作成または検証を行います。

```
&lt;username&gt;: &lt;string&gt;  
ユーザー名
```

**--new-format &lt;boolean&gt; (デフォルト= 1)**

このパラメータは現在無視され、1とみなされます。

**--otp &lt;string&gt;**

二要素認証用のワンタイムパスワード。

**--path &lt;文字列&gt;**

チケットを検証し、ユーザーがパスに対するアクセス権限を持っているか確認する

---

#### 注記

必要なオプション: privs

---

```
--privs &lt;文字列&gt;  
チケットを検証し、ユーザーがパスに対するアクセス権限を持っているかどうかを確認します
```

---

#### 注

必要なオプション: path

---

```
--realm <string>;
このパラメータを使用して、オプションでレルムを渡すことができます。通常、レルムは単にユーザー名 <username>@<realm> に追加されます。
```

```
--tfa-challenge <string>;
ユーザーが応答する署名付きTFAチャレンジ文字列。
```

```
pveum user add <userid> [OPTIONS]
```

新規ユーザーを作成します。

```
<userid>; <string>;
完全なユーザーID（名前@領域形式）。
```

```
--comment <string>;
説明はありません
```

```
--email <string>;
説明なし
```

```
--enable <boolean> (デフォルト= 1)
アカウントを有効化（デフォルト）。0に設定するとアカウントを無効化
```

```
--expire <integer> (0 - N)
アカウント有効期限（エポックからの秒数）。0/期限なしを意味します。
```

```
--firstname <string>;
説明なし
```

```
--groups <string>;
説明なし
```

```
--keys [0-9a-zA-Z!=]{0,4096}
二要素認証用キー (yubico)。
```

```
--lastname <string>;
説明なし
```

```
--password <string>;
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>; <string>;
完全なユーザーID（名前@レルム形式）。
```

**pveum user list** [オプション] [フォーマットオプション]

ユーザーインデックス。

**--enabled <boolean>**  
enable プロパティのオプションフィルター。

**--full <boolean> (デフォルト= 0)**  
グループおよびトークン情報を含める。

**pveum user modify** <userid> [オプション]

ユーザー設定を更新します。

**<userid>: <string>**  
完全なユーザーID (名前@レルム形式)。

**--append <boolean>**  
説明なし

---

#### 注記

必要なオプション: groups

---

**--comment <文字列>**  
説明なし

**--email <string>**  
説明がありません

**--enable <boolean> (デフォルト= 1)**  
アカウントを有効化 (デフォルト)。0に設定するとアカウントを無効化

**--expire <integer> (0 - N)**  
アカウント有効期限 (エポックからの秒数)。0は期限なしを意味します。

**--firstname <string>**  
説明なし

**--groups <string>**  
説明なし

**--keys [0-9a-zA-Z!=]{0,4096}**  
二要素認証用キー (yubico)。

**--lastname <string>**  
説明なし

---

**pveum ユーザー権限 [<userid>] [OPTIONS] [FORMAT\_OPTIONS]**

指定されたユーザー/トークンの有効な権限を取得します。

<userid>:

ユーザーIDまたは完全なAPIトークンID

--path <string>;

この特定のパスのみをダンプし、ツリー全体はダンプしない。

**pveum user tfa delete <userid> [オプション]**

ユーザーのTFAエントリを削除します。

<userid>: <string>;

完全なユーザーID (name@realm形式)。

--id <string>;

TFA ID。指定しない場合、すべてのTFAエントリが削除されます。

**pveum user tfa list [<userid>]**

TFAエントリの一覧を表示します。

<userid>: <string>;

完全なユーザーID (名前@realm形式)。

**pveum user tfa unlock <userid>;**

ユーザーのTFA認証を解除する。

<userid>: <string>;

完全なユーザーID (name@realm形式)。

**pveum user token add <userid> <tokenid> [OPTIONS] [FORMAT\_OPTIONS]**

特定のユーザー向けに新しいAPIトークンを生成します。注意: APIトークン値を返します。後で取得できないため、必ず保存してください！

<userid>: <string>;

完全なユーザーID (name@realm形式)。

ユーザー固有のトークン識別子。

```
--comment &lt;string&gt;
```

説明なし

```
--expire &lt;integer&gt; (0 - N) (デフォルト= ユーザーと同じ)
```

APIトークンの有効期限（エポックからの秒数）。0は有効期限なしを意味します。

```
--privsep &lt;boolean&gt; (デフォルト= 1)
```

APIトークンの権限を個別のACLで制限（デフォルト）、または対応するユーザーの全権限を付与。

```
pveum user token delete &lt;userid&gt; &lt;tokenid&gt; [FORMAT_OPTIONS]
```

特定のユーザーのAPIトークンを削除します。

```
&lt;userid&gt;: &lt;string&gt;
```

フルユーザーID（名前@realm形式）。

ユーザー固有のトークン識別子。

```
pveum user token list &lt;userid&gt; [FORMAT_OPTIONS]
```

ユーザー API トークンを取得します。

```
&lt;userid&gt;: &lt;string&gt;
```

完全なユーザーID（名前@realm形式）。

```
pveum user token modify &lt;userid&gt; &lt;tokenid&gt; [OPTIONS] [FORMAT_OPTIONS]
```

特定のユーザーに対するAPIトークンを更新します。

```
&lt;userid&gt;: &lt;string&gt;
```

完全なユーザーID（name@realm形式）。

ユーザー固有のトークン識別子。

```
--comment &lt;string&gt;
```

説明なし

```
--expire &lt;integer&gt; (0 - N) (デフォルト= ユーザーと同じ)
```

APIトークンの有効期限（エポックからの秒数）。0は期限なしを意味します。

```
--privsep &lt;boolean&gt; (デフォルト= 1)
```

APIトークンの権限を個別のACLで制限する（デフォルト）、または対応するユーザーに完全な権限を付与する。

```
pveum user token permissions &lt;userid&gt; &lt;tokenid&gt; [OPTIONS] [FORMAT_OPTIONS]
```

指定されたトークンの有効な権限を取得します。

&lt;userid&gt;; &lt;string&gt;  
フルユーザーID (名前@レルム形式)。

ユーザー固有のトークン識別子。

--path &lt;string&gt;  
この特定のパスのみをダンプし、ツリー全体はダンプしません。

**pveum user token remove**

*pveum user token delete* の別名。

**pveum useradd**

*pveum user add* の別名。

**pveum userdel**

*pveum user delete* の別名。

**pveum usermod**

*pveum user modify* の別名。

## A.16 vzdump - VM およびコンテナ用バックアップユーティリティ

**vzdump ヘルプ**

**vzdump {&lt;vmid&gt;} [オプション]**

バックアップを作成します。

&lt;vmid&gt;; &lt;string&gt;  
バックアップ対象のゲストシステムのID。

--all &lt;boolean&gt; (デフォルト= 0)  
このホスト上の既知のゲストシステムをすべてバックアップします。

--bwlimit &lt;integer&gt; (0 - N) (デフォルト= 0)  
I/O 帯域幅を制限します (単位: KiB/s)。

--compress &lt;0| 1| gzip| lzo| zstd&gt; (デフォルト= 0)  
ダンプファイルを圧縮します。

--dumpdir &lt;string&gt;  
生成されたファイルを指定ディレクトリに保存します。

--exclude &lt;文字列&gt;  
指定されたゲストシステムを除外する (-all を前提とする)

**--exclude-path <array>**

特定のファイル/ディレクトリを除外（シェルグロブ）。/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を基準に相対的に一致する。

**--fleecing [[enabled=<1|0>] [,storage=<storage ID>]]**

バックアップ処理のオプション（VMのみ）。

**--ionice <整数> (0 - 8) (デフォルト= 7)**

BFQスケジューラ使用時のIO優先度を設定します。VMのスナップショットおよびサスペンドモードバックアップでは、これは圧縮ツールにのみ影響します。値8はアイドル優先度を使用することを意味し、それ以外の場合は指定値でベストエフォート優先度が使用されます。

**--job-id \S+**

バックアップジョブのID。設定すると、バックアップ通知の`backup-job`メタデータフィールドにこの値が設定されます。`root@pam`のみがこのパラメータを設定できます。

**--lockwait <integer> (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。通知メールを送信するタイミングを指定します

**--mailto <string>;**

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。メール通知を受信すべきメールアドレスまたはユーザーのカンマ区切りリスト。

**--maxfiles <integer> (1 - N)**

非推奨: `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 <snapshot| stop| suspend> (デフォルト= snapshot)**

バックアップモード。

**--node <string>;**

このノード上で実行された場合のみ実行する。

**--notes-template <string>;**

バックアップのノートを生成するためのテンプレート文字列。変数を含めることができます。現在サポートされているのは`\{cluster\}`、`\{guestname\}`、`\{node\}`、および`\{vmid\}`ですが、将来追加される可能性があります。1行で記述する必要があり、改行とバックスラッシュはそれぞれ\nと\でエスケープする必要があります。

---

**注記**

必要なオプション: storage

---

--notification-mode <auto| legacy-sendmail| notification-system>;  
(デフォルト= auto)

使用する通知システムを決定します。*legacy-sendmail* に設定すると、vzdump は mailto-/mailnotification パラメータを考慮し、sendmail コマンド経由で指定され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 および mailnotification は無視されます。*auto* (デフォルト設定) に設定した場合、mailtoが設定されている場合はメールが送信され、設定されていない場合は通知システムが使用されます。

--pbs-change-detection-mode <データ| レガシー| メタデータ>;

コンテナバックアップのファイル変更検出およびエンコーディング形式切り替えに使用するPBSモード。

--performance [max-workers=<integer>] [,pbs-entries-max=<integer>]  
その他のパフォーマンス関連の設定。

--pigz <整数>; (デフォルト= 0)

N>0の場合、gzipの代わりにpigzを使用します。N=1ではコア数の半分を使用し、N>1ではNをスレッド数として使用します。

--pool <文字列>;

指定されたプールに含まれる既知のゲストシステム全てをバックアップします。

--protected <boolean>;

trueの場合、バックアップを保護済みとしてマークします。

---

#### 注記

必要なオプション: storage

---

--prune-backups [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)

ストレージ設定の保持オプションの代わりに、これらの保持オプションを使用します。

--quiet <boolean>; (デフォルト= 0)

静かにする。

--remove <boolean>; (デフォルト= 1)

*prune-backups*に基づいて古いバックアップを削除する。

--script <string>;

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外します。

--stdout <boolean>;

ファイルではなく標準出力にtarを出力します。

---

--stop &lt;boolean&ampgt (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 &lt;integer&ampgt (0 - N) (デフォルト= 10)

ゲストシステムが停止するまで待機する最大時間 (分)。

--storage &lt;ストレージID&ampgt

結果ファイルをこのストレージに保存します。

--tmpdir &lt;string&ampgt

一時ファイルを指定ディレクトリに保存します。

--zstd &lt;整数&ampgt (デフォルト= 1)

Zstdスレッド数。N=0の場合、利用可能なコアの半数を使用します。Nが0より大きい値に設定された場合、Nがスレッド数として使用されます。

## A.17 ha-manager - Proxmox VE HA Manager

ha-manager &lt;コマンド&ampgt [引数] [オプション]

ha-manager add &lt;sid&ampgt [オプション]

新しい HA リソースを作成します。

&lt;sid&ampgt: &lt;type&ampgt: &lt;name&ampgt

HAリソースID。リソースタイプとリソース固有の名前をコロンで区切った形式です（例: vm:100 / ct:100）。仮想マシンやコンテナの場合は、VM IDやCT IDをショートカットとして使用できます（例: 100）。

--comment &lt;string&ampgt

説明。

--failback &lt;boolean&ampgt (デフォルト= 1)

現在のノードよりも優先度の高いノードがオンラインになった場合、ノードアフィニティルールに従い、HAリソースを自動的に優先度の高いノードへ移行します。

--group &lt;string&ampgt

HAグループ識別子。

--max\_relocate &lt;integer&ampgt (0 - N) (デフォルト= 1)

サービス起動に失敗した際の最大再配置試行回数。

--max\_restart &lt;integer&ampgt (0 - N) (デフォルト= 1)

ノード上でサービスの起動に失敗した後の、サービスの再起動試行の最大回数。

```
--state <disabled| enabled| ignored| started| stopped>; (デフォルト=started)
```

要求されたリソースの状態。

```
--type <ct| vm>;
```

リソースタイプ。

#### ha-manager config [OPTIONS]

HAリソースを一覧表示します。

```
--type <ct| vm>;
```

特定のタイプのリソースのみを一覧表示

#### ha-manager crm-command migrate <sid>; <node>;

別のノードへのリソース移行（オンライン）を要求します。

```
<sid>; <type>; <name>;
```

HAリソースID。リソースタイプとリソース固有名からなり、コロンで区切られます（例: vm:100 / ct:100）。仮想マシンとコンテナの場合、VM IDまたはCT IDをショートカットとして使用できます（例: 100）。

```
<node>; <string>;
```

対象ノード。

#### ha-manager crm-command node-maintenance disable <node>;

ノードメンテナンス要求の状態を変更します。

```
<node>; <string>;
```

クラスタノード名。

#### ha-manager crm-command node-maintenance enable <node>;

node-maintenance リクエストの状態を変更します。

```
<node>; <string>;
```

クラスタノード名。

#### ha-manager crm-command relocate <sid>; <node>;

リソースを別のノードへ再配置します。これにより、元のノード上のサービスが停止され、ターゲットノード上で再起動されます。

**&lt;sid&gt;::&lt;type&gt;::&lt;name&gt;**

HAリソースID。リソースタイプとリソース固有の名前をコロンで区切った形式です（例: vm:100 / ct:100）。仮想マシンやコンテナの場合は、VM IDやCT IDを短縮形として使用できます（例: 100）。

**&lt;node&gt;::&lt;string&gt;**

対象ノード。

**ha-manager crm-command stop &lt;sid&gt; [&lt;timeout&gt;]**

サービスの停止を要求します。

**&lt;sid&gt;::&lt;type&gt;::&lt;name&gt;**

HAリソースID。リソースタイプとリソース固有名からなり、コロンで区切られます（例: vm:100 / ct:100）。仮想マシンやコンテナの場合は、VM IDやCT IDをショートカットとして使用できます（例: 100）。

**&lt;timeout&gt;::&lt;integer&gt; (0 - N)**

タイムアウト（秒単位）。0に設定すると強制停止が実行されます。

**ha-manager groupadd &lt;group&gt; --nodes &lt;string&gt; [OPTIONS]**

新しい HA グループを作成します（HA ルールに取って代わられ非推奨）。

**&lt;group&gt;::&lt;string&gt;**

HAグループの識別子。

**--comment &lt;文字列&gt;**

説明。

**--nodes &lt;node&gt;[:&lt;pri&gt;]{,&lt;node&gt;[:&lt;pri&gt;]}\***

クラスタノード名のリスト（優先度はオプション）。

**--nofailback &lt;boolean&gt; (デフォルト= 0)**

CRMは優先度が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はそのノードにサービスを移行します。nofailbackを有効にすると、この動作を防止します。

**--restricted &lt;boolean&gt; (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、グループで定義されたノードでのみ実行できます。

**--type &lt;group&gt;**

グループタイプ。

**ha-manager groupconfig**

HAグループを取得します。（HAルールに取って代わられ非推奨）

**ha-manager groupremove &lt;group&gt;**

HAグループ設定を削除します。（HAルールに取って代わられ非推奨）

&lt;group&gt;; &lt;string&gt;

HAグループの識別子。

**ha-manager groupset** &lt;group&gt; [OPTIONS]

HAグループ設定を更新します。 (HAルールに取って代わられ非推奨)

&lt;group&gt;; &lt;string&gt;

HAグループの識別子。

--comment &lt;string&gt;

説明。

--delete &lt;string&gt;

削除したい設定のリスト。

--digest &lt;string&gt;

現在の設定ファイルのダイジェストが異なる場合、変更を防止します。これにより同時編集を防止できます。

--nodes &lt;node&gt;[:&lt;pri&gt;]{,&lt;node&gt;[:&lt;pri&gt;]}\*

クラスタノード名のリスト（オプションで優先度付き）。

--nofailback &lt;boolean&gt; (デフォルト= 0)

CRMは優先度が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はそのノードへサービスを移行します。nofailbackを有効にするとこの動作を防止します。

--restricted &lt;boolean&gt; (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、グループで定義されたノードでのみ実行できます。

**ha-manager help** [オプション]

指定されたコマンドに関するヘルプを取得します。

--extra-args &lt;array&gt;

特定のコマンドのヘルプを表示します

--verbose &lt;boolean&gt;

詳細出力形式。

**ha-manager migrate**

*ha-manager* *crm-command* *migrate* の別名。

**ha-manager relocate**

*ha-manager* *crm-command* *relocate* の別名。

**ha-manager remove** &lt;sid&gt;

リソース構成を削除します。

**&lt;sid&gt;::&lt;type&gt;::&lt;name&gt;**

HAリソースID。リソースタイプとリソース固有の名前をコロンで区切ったものです（例: vm:100 / ct:100）。仮想マシンやコンテナの場合は、VM IDやCT IDをショートカットとして使用できます（例: 100）。

```
ha-manager rules add &lt;type&gt; &lt;rule&gt; --resources &lt;string&gt; [OPTIONS]
```

HAルールを作成します。

**&lt;type&gt;::&lt;node-affinity| resource-affinity&gt;**

HAルールタイプ。

**&lt;rule&gt;::&lt;string&gt;**

HAルールの識別子。

**--comment &lt;string&gt;**

HAルールの説明。

**--disable &lt;boolean&gt;**

HAルールが無効化されているかどうか。

**--resources &lt;type&gt;::&lt;name&gt;{,&lt;type&gt;::&lt;name&gt;}\***

HAリソースIDのリスト。リソースタイプとリソース固有名のリストで構成され、

（例: vm:100,ct:101）。

条件付きオプション：

**[type=node-affinity]**

```
--nodes &lt;node&gt;[:&lt;pri&gt;]{,&lt;node&gt;[:&lt;pri&gt;]}*
```

クラスタノード名のリスト（優先度はオプション）。

**--strict &lt;boolean&gt; (デフォルト= 0)**

ノードアフィニティルールが厳密か非厳密かを記述します。

**[type=resource-affinity]****--affinity &lt;負| 正&gt;**

HAリソースを同一ノードに保持する（正）か、別々のノードに保持する（負）かを記述します。

```
ha-manager rules config [OPTIONS] [FORMAT_OPTIONS]
```

HAルールを取得します。

**--resource &lt;string&gt;**

返されるリストを、指定されたリソースに影響するルールに限定します。

```
--type <node-affinity| resource-affinity>;
```

返されるリストを指定されたルールタイプに制限します。

```
ha-manager rules list [オプション] [フォーマットオプション]
```

HAルールを取得します。

```
--リソース <文字列>;
```

返されるリストを、指定されたリソースに影響するルールに限定します。

```
--type <node-affinity| resource-affinity>;
```

返されるリストを指定されたルールタイプに制限します。

```
ha-manager rules remove <rule>;
```

HAルールを削除します。

```
<rule>::<string>;
```

HAルールの識別子。

```
ha-manager rules set <type> <rule> [OPTIONS]
```

HAルールを更新します。

```
<type>:<node-affinity| resource-affinity>;
```

HAルールのタイプ。

```
<rule>:<string>;
```

HAルールの識別子。

```
--comment <string>;
```

HAルールの説明。

```
--delete <string>;
```

削除したい設定のリスト。

```
--digest <string>;
```

現在の設定ファイルのダイジェストが異なる場合、変更を防止します。これにより同時編集を防止できます。

```
--disable <boolean>;
```

HAルールが無効化されるかどうか。

```
--resources <type>:<name>{,<type>:<name>}*
```

HAリソースIDのリスト。リソースタイプとリソース固有名のリストで構成され、

(例: vm:100,ct:101)。

条件付きオプション:

[**type=node-affinity**]

--nodes <node>[:<pri>]{,<node>[:<pri>]}\*  
クラスタノード名のリスト（オプションで優先度付き）。

--strict <boolean> (デフォルト= 0)  
ノードアフィニティールールが厳密か非厳密かを記述します。

[**type=resource-affinity**]

--affinity <正|負>  
HAリソースを同一ノードに保持する（正）か、別々のノードに保持する（負）かを記述します。

**ha-manager set** <sid> [OPTIONS]

リソース構成を更新します。

<sid>:<type>:<name>;

HAリソースID。リソースタイプとリソース固有の名前をコロンで区切った形式（例: vm:100 / ct:100）。仮想マシンやコンテナの場合は、VM IDやCT IDを短縮形として使用可能（例: 100）。

--comment <string>;

説明。

--delete <string>;

削除したい設定のリスト。

--digest <string>;

現在の設定ファイルのダイジェストが異なる場合、変更を防止します。これにより同時編集を防止できます。

--failback <boolean> (デフォルト= 1)

現在のノードより優先度の高いノードがオンラインになった場合、ノードアフィニティールールに基づき、自動的にHAリソースを優先度の高いノードへ移行します。

--group <string>;

HAグループ識別子。

--max\_relocate <integer> (0 - N) (デフォルト= 1)

サービス起動に失敗した場合の最大再配置試行回数。

--max\_restart <integer> (0 - N) (デフォルト= 1)

ノード上でサービスの起動に失敗した場合の、再起動試行の最大回数。

**--state <disabled| enabled| ignored| started| stopped>** (デフォルト=started)

要求されたリソースの状態。

#### **ha-manager status [オプション]**

HA マネージャーのステータスを表示します。

**--verbose <boolean>** (デフォルト= 0)

詳細出力。完全なCRMおよびLRMステータス (JSON) を含める。

## 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** <コマンド> [引数] [オプション]

**pve-firewall compile**

ファイアウォールルールをコンパイルして表示します。テストに便利です。

**pve-firewall help** [オプション]

指定されたコマンドに関するヘルプを取得します。

**--extra-args** <array>;

特定のコマンドのヘルプを表示します

**--verbose** <boolean>;

詳細出力形式。

**pve-firewall localnet**

ローカルネットワークに関する情報を表示します。

**pve-firewall restart**

Proxmox VE ファイアウォールサービスを再起動します。

**pve-firewall simulate** [オプション]

ファイアウォールルールをシミュレートします。これはカーネルルーティングテーブルをシミュレートするものではなく、単にソースゾーンから宛先ゾーンへのルーティングが可能であると仮定します。

**--dest** <string>;

宛先IPアドレス。

**--dport** <integer>;

宛先ポート。

**(デフォルト= 外部)**

送信元ゾーン。

**--protocol (tcp|udp) (デフォルト= tcp)**

プロトコル。

**--source &lt;string&gt;**

送信元 IP アドレス。

**--sport &lt;integer&gt;**

送信元ポート。

**(デフォルト= ホスト)**

宛先ゾーン。

**--verbose &lt;boolean&gt; (デフォルト= 0)**

詳細出力。

**pve-firewall start [オプション]**

Proxmox VE ファイアウォールサービスを起動します。

**--debug &lt;boolean&gt; (デフォルト= 0)**

デバッグモード - フォアグラウンドに留まる

**pve-firewall status** ファイア

ウォールの状態を取得します

**。pve-firewall stop** ファイア

ウォールを停止します。

Proxmox VE ファイアウォールサービスを停止します。注意：停止すると、すべての Proxmox VE 関連の iptables ルールが削除され、ホストが保護されない状態になる可能性があります。

## B.2 pvedaemon - Proxmox VE API デーモン

**pvedaemon &lt;コマンド&gt; [引数] [オプション]****pvedaemon help [オプション]**

指定したコマンドのヘルプを取得します。

**--extra-args &lt;array&gt;**

特定のコマンドのヘルプを表示します

**--verbose <boolean>**

詳細出力形式。

**pvedaemon restart**

デーモンを再起動（または実行されていない場合は起動）。

**pvedaemon start [オプション]**

デーモンを起動します。

**--debug <boolean> (デフォルト= 0)**

デバッグモード - フォアグラウンドで実行

**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 <コマンド> [引数] [オプション]****pveproxy help [オプション]**

指定されたコマンドに関するヘルプを取得します。

**--extra-args <array>**

特定のコマンドのヘルプを表示します

**--verbose <boolean>**

詳細出力形式。

**pveproxy restart**

デーモンを再起動（実行中でない場合は起動）。

**pveproxy start [オプション]**

デーモンを起動します。

**--debug <boolean> (デフォルト= 0)**

デバッグモード - フォアグラウンドで実行

**pveproxy status** デーモンの

ステータスを取得します。

**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** <コマンド> [引数] [オプション]

**pvestatd help** [オプション]

指定したコマンドに関するヘルプを取得します。

**--extra-args** <array>

特定のコマンドのヘルプを表示します

**--verbose** <boolean>

詳細出力形式。

**pvestatd restart**

デーモンを再起動（実行中でない場合は起動）。

**pvestatd start** [オプション]

デーモンを起動します。

**--debug** <boolean> (デフォルト= 0)

デバッグモード - フォアグラウンドで実行

**pvestatd status** デーモンのス

テータスを取得します。

**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** <コマンド> [引数] [オプション]

**spiceproxy help** [オプション]

指定されたコマンドに関するヘルプを取得します。

**--extra-args** <array>

特定のコマンドのヘルプを表示します

**--verbose** <boolean>

詳細出力形式。

**spiceproxy restart**

デーモンを再起動します（実行されていない場合は起動します）。

**spiceproxy start** [オプション]

デーモンを起動します。

```
--debug <boolean> (デフォルト= 0)
デバッグモード - フォアグラウンドで実行
```

**spiceproxy status** デーモンの

ステータスを取得します。

**spiceproxy stop** デーモンを

停止します。

## B.6 pmxcfs - Proxmox クラスターファイルシステム

**pmxcfs** [オプション]

ヘルプ オプション:

```
-h, --help
ヘルプオプションを表示 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有効にする
```

```
-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ツールセットを使用して起動および管理されます。サービスの名前は*pve-cluster*です。

```
systemctl start pve-clustersystemctl stop
```

```
pve-clustersystemctl status pve-cluster
```

## B.7 pve-ha-crm - クラスターリソースマネージャーデーモン

**pve-ha-crm** <コマンド> [引数] [オプション]

**pve-ha-crm help** [OPTIONS]

指定されたコマンドに関するヘルプを取得します。

--extra-args <array>  
特定のコマンドのヘルプを表示します

--verbose <boolean>  
詳細出力形式。

**pve-ha-crm start** [オプション]

デーモンを起動します。

--debug <boolean> (デフォルト= 0)  
デバッグモード - フォアグラウンドで実行

**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** <コマンド> [引数] [オプション]

**pve-ha-lrm help** [オプション]

指定したコマンドに関するヘルプを取得します。

--extra-args <array>  
特定のコマンドのヘルプを表示します

--verbose <boolean>  
詳細出力形式。

**pve-ha-lrm start** [オプション]

デーモンを起動します。

--debug <boolean> (デフォルト= 0)  
デバッグモード - フォアグラウンドで実行

**pve-ha-lrm status** デーモンの

ステータスを取得します。

**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** <コマンド> [引数] [オプション]

**pvescheduler help** [オプション]

指定されたコマンドに関するヘルプを取得します。

**--extra-args** <array>;

特定のコマンドのヘルプを表示します

**--verbose** <boolean>;

詳細出力形式。

**pvescheduler restart**

デーモンを再起動します（実行中でない場合は起動します）。

**pvescheduler start** [オプション]

デーモンを起動します。

**--debug** <boolean>; (デフォルト= 0)

デバッグモード - フォアグラウンドで実行

**pvescheduler status** デーモンの

ステータスを取得します。

**pvescheduler stop** デーモンを停

止します。

## 付録 C 設定ファイル

### C.1 概要

Proxmox VE の設定ファイルのほとんどは、`/etc/pve` にマウントされた[共有クラスタファイルシステム](#)上にあります。`/etc/vzdump.conf` にあるバックアップ用のノード固有の設定ファイルなど、例外もあります。

通常、設定ファイルのプロパティは、関連する API エンドポイントにも使用される JSON スキーマから派生しています。

#### C.1.1 プロパティ名の大文字小文字

従来、長いプロパティ（およびサブプロパティ）にはスネークケースが使用されるか、または1語で記述されることが多かった。これは主に、Proxmox VE スタックがプログラミング言語Perlで開発されたことに起因すると考えられる。Perlではケバブケースのプロパティアクセスに追加の引用符が必要であり、また初期開発段階ではスタイルの強制が弱かったため、開発者によって異なる規約が使用されていた。

新規プロパティではケバブケースが推奨され、既存のスネークケースプロパティにはエイリアスを導入する計画です。長期的には、設定復元時の後方互換性を維持しつつ、API、CLI、および使用中の設定ファイルをケバブケースに移行する予定です。

### C.2 データセンター設定

ファイル `/etc/pve/datacenter.cfg` は Proxmox VE の設定ファイルです。クラスタ全体のデフォルト値が含まれており、すべてのノードで使用されます。

#### C.2.1 ファイル形式

このファイルは、コロンで区切られたキー/値形式のシンプルな形式を使用しています。各行は次の形式です：

OPTION: value

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、同様に無視されます。

## C.2.2 オプション

**bwlimit: [clone=<LIMIT>] [,default=<LIMIT>] [,migration=<LIMIT>] [,move=<LIMIT>]  
[,restore=<LIMIT>]**

各種操作の I/O 帯域幅制限を (KiB/s 単位) 設定します。

**clone= <LIMIT>;**

ディスククローン作成時の帯域幅制限 (KiB/s単位)

**デフォルト= <LIMIT>;**

デフォルトの帯域幅制限 (KiB/s単位)

**migration= <LIMIT>;**

ゲストの移行 (ローカルディスクの移動を含む) における帯域幅制限 (KiB/s単位)

**移動= <LIMIT>;**

ディスク移動時の帯域幅制限 (KiB/s単位)

### = の復元

バックアップからのゲスト復元における帯域幅制限 (KiB/s単位)

**consent-text: <string>;**

ログイン前に表示される同意テキスト

**console: <applet| html5| vv| xtermjs>;**

デフォルトのコンソールビューアを選択します。組み込みの Java アプレット (VNC; 非推奨で HTML5 にマッピング)、外部 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) 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: VM で SPICE が有効化されていない)、フォールバックは noVNC になります。

**crs: [ha=<basic|static>] [,ha-rebalance-on-start=<1|0>]**

クラスタリソーススケジューリング設定。

**ha= <basic| static>; (デフォルト= basic)**

HA マネージャーがサービスの起動または回復に使用するノードの選択方法を設定します。basic ではサービスの数のみが使用され、static ではサービスの静的な CPU およびメモリ構成が考慮されます。

**ha-rebalance-on-start= <boolean>; (デフォルト= 0)**

HAサービスの状態が停止から起動に変化した際、適切なノード選択にCRSを使用するよう設定します。

**description: <string>;**

データセンターの説明。Web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ます。設定ファイル内にコメントとして保存されます。

**email\_from: <string>;**

通知送信元のメールアドレスを指定 (デフォルトは root@\$hostname)

**フェンシング: &lt;both| hardware| watchdog&gt; (デフォルト= watchdog)**

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。bothを選択すると両モードが使用されます。

**警告**

*hardware* および *both* は実験的かつ開発中

**ha: shutdown\_policy=&lt;enum&gt;**

クラスタ全体の HA 設定。

**shutdown\_policy= &lt;条件付き| failover| freeze| migrate&gt; (デフォルト= conditional)**

ノードの電源オフまたは再起動時のHAサービス処理方針を説明します。Freezeはシャットダウン時に当該ノード上に残存するサービスを常に凍結し、それらのサービスはHAマネージャによって復旧されません。フェイルオーバーはサービスを凍結状態とマークしないため、シャットダウンしたノードが短時間（1分未満）で再起動しない場合、サービスは他のノードに復旧される。条件付きはシャットダウンの種類に応じて自動的に選択する。つまり、再起動時にはサービスは凍結されるが、電源オフ時にはサービスは現状維持され、約2分後に復旧される。移行は、再起動またはシャットダウンがトリガーされた際に、実行中の全サービスを別のノードへ移動しようと試みます。電源オフ処理は、ノード上に実行中のサービスが一切存在しなくなった時点で初めて継続されます。ノードが再起動した場合、少なくとも他の移行・再配置・復旧が発生していない限り、サービスは以前に電源が切られたノードへ戻されます。

**http\_proxy: http://.\***

ダウンロードに使用する外部HTTPプロキシを指定します（例: *http://username:password@host:port/*）

**キーボード: &lt;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&gt;**

VNCサーバーのデフォルトキーボードレイアウト。

**言語: &lt;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&gt;**  
デフォルトのGUI言語。**mac\_prefix: &lt;string&gt; (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、IEEE が Proxmox Server Solutions GmbH に割り当てた MAC アドレスブロックラージ (MA-L) の組織固有識別子 (OUI) です。このプレフィックスは、パブリックから直接アクセスできないローカルネットワーク（例：LANやNAT/マスカレード環境）でのみ使用可能です。

複数のクラスターを運用し、仮想ゲストのネットワークを（部分的に）共有する場合、MAC衝突の可能性を低減するため、デフォルトのMACプレフィックスを拡張するか、カスタム（有効な）プレフィックスを生成することを強く推奨します。

例として、各クラスターごとにProxmox OUIに別個の16進数を追加します（例：1つ目はBC:24:11:0、2つ目はBC:24:11:1）。あるいは、VLANなどを用いてゲストのネットワークを論理的に分離する方法もあります。

+ 公開アクセス可能なゲストについては、[IEEE](#)登録から独自のOUIを取得するか、自社またはホスティングプロバイダのネットワーク管理者との調整をお勧めします。

**max\_workers: <integer> (1 - N)**

ha-managerからのstopall VMsやタスクなどのアクションで起動されるワーカー（ノードあたり）の最大数を定義します。

**migration: [type=] <secure|insecure> [,network=<CIDR>]**

クラスタ全体の移行設定用。

**network= <CIDR>**

移行に使用される（サブ）ネットワークのCIDR。レプリケーションネットワーク設定が未設定の場合、レプリケーションジョブのフォールバックとして使用されます。

**type= <secure|secure> (デフォルト = secure)**

デフォルトでは、移行トラフィックは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、パフォーマンス向上のためにこれを無効化できます。

**migration\_unsecure: <boolean>**

デフォルトではSSHトンネルを使用した移行は安全です。安全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無効化できます。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=<integer>] [,upper=<integer>]**

自動選択される空きVMIDのプール範囲を制御します。

**lower= <integer> (デフォルト = 100)**

next-id API の利用可能範囲の下限（含む）。

**upper= <integer> (デフォルト = 1000000)**

フリー-next-id API範囲の上限（除外）。

**通知: [フェンシング=<常に|無効>] [,パッケージ更新=<常に|無効>]****自動|常に|無効|**

[,replication=<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= <always|never>**

未使用 - 代わりにデータセンター通知設定を使用してください。

**package-updates= <always|auto|never> (デフォルト = auto)**

非推奨: 代わりにデータセンターの通知設定を使用してください。日々の更新ジョブが通知を送信する頻度を制御します:

- `auto daily` 有効なサブスクリプションを持つシステム向け。本番環境と想定されるため、保留中の更新を把握すべきです。
- 有効なサブスクリプションを持つシステムの場合、毎日自動で通知します。これらは本番環境準備完了とみなされ、保留中の更新について知る必要がある
- `never` 新規保留更新がある場合でも通知を送信しない。

`replication= <always| never>;`

UNUSED - 代わりにデータセンター通知設定を使用します。

`target-fencing= <TARGET>;`

UNUSED - 代わりにデータセンターの通知設定を使用します。

`target-package-updates= <TARGET>;`

UNUSED - 代わりにデータセンターの通知設定を使用します。

`target-replication= <TARGET>;`

未使用 - 代わりにデータセンターの通知設定を使用してください。

#### 登録済みタグ: `<タグ>[;<タグ>...]`

/に対する `Sys.Modify` が必要となる設定および削除対象のタグ一覧。ここに設定され、かつ `user-tag-access` にも含まれる場合、`Sys.Modify` が必要です。

`replication: [type=]<secure|insecure> [,network=<CIDR>];`

クラスタ全体のレプリケーション設定用。

`network= <CIDR>;`

レプリケーションジョブに使用される（サブ）ネットワークのCIDR。

`type= <insecure| secure>; (デフォルト= secure)`

レプリケ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。安全な完全プライベートネットワークでは、パフォーマンス向上のために無効化できます。

#### タグスタイル: [大文字小文字を区別する=<1|0>:]

`[,color-map=<tag>:<hex-color>[:<hex-color-for-text>][;<tag>=...]]`  
`[,ordering=<config|alphabetical>] [,shape=<enum>];`

タグスタイルのオプション。

`大文字小文字を区別するかどうかを制御します。= <boolean> (デフォルト= 0)`

更新時に一意のタグをフィルタリングする際に大文字小文字を区別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= <アルファベット順| config>; (デフォルト= alphabetical)`

WebインターフェースおよびAPI更新におけるタグのソート順を制御します。

`形状= <circle| dense| full| none>; (デフォルト= circle)`

Web UIツリー用タグ形状。`full`はタグ全体を描画。`circle`は背景色のみの円を描画。`dense`は小さな矩形のみを描画（各ゲストに多数のタグが割り当てられている場合に有用）。`none`はタグ表示を無効化。

**u2f: [appid=<APPID>] [,origin=<URL>]**  
u2f

**appid= <APPID>**  
U2F AppId URLの上書き。デフォルトはオリジン。

**origin= <URL>**  
U2F オリジン上書き。単一ノードで単一URLを使用する場合に主に有用。

**user-tag-access: [user-allow=<enum>] [,user-allow-list=<tag>[,<tag>,...]]**  
ユーザー設定可能なタグの権限オプション

**user-allow= <既存| free| list| none> (デフォルト= free)**

ユーザーが制御するリソース（ゲストなど）に対して設定または削除可能なタグを制御します。/ に対する Sys.Modify 特権を持つユーザーは常に制限なしです。

- *none* タグは使用不可。
- *list user-allow-list* に記載されたタグが使用可能。
- 既存のタグも使用可能（既存リストと同様）。
- *free* タグの制限なし。

**user-allow-list= <タグ>[,<タグ>,...]**

ユーザーが設定および削除を許可されているタグのリスト（セミコロン区切り）  
および既存の値

**webauthn: [allow-subdomains=<1|0>] [,id=<DOMAINNAME>] [,origin=<URL>]  
[,rp=<RELYING\_PARTY>]**  
webauthn 設定

**allow-subdomains= <boolean> (デフォルト= 1)**

オリジンを完全なURLではなくサブドメインとして許可するかどうか。

**id= <DOMAINNAME>;**

信頼当事者ID。プロトコル、ポート、場所を含まないドメイン名である必要があります。これを変更すると  
既存の認証情報を無効にします。

**origin= <URL>;**

サイトオリジン。https:// URL（または http://localhost）でなければなりません。ユーザーがブラウザに入力してWebインター  
フェースにアクセスするアドレスを含める必要があります。これを変更すると既存の認証情報が無効になる可能性があります。

**=信頼当事者名。任意のテキスト識別子。変更すると既存の認証情報が無効になる可能性があります。**

信頼当事者名。任意のテキスト識別子。変更すると既存の認証情報が無効になる可能性があります。

## 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備えています。これは `systemd time` カレンダーイベント形式に基づいています。<sup>1</sup> カレンダーイベントは、単一の式で1つ以上の時点を参照するために使用できます。

このようなカレンダーイベントは次の形式を使用します：

```
[WEEKDAY] [[YEARS-]MONTHS-DAYS] [HOURS:MINUTES[:SECONDS]]
```

この形式では、ジョブを実行する日数を設定できます。また、1つ以上の開始時刻を設定することも可能です。これによりレプリケーションスケジューラはジョブ開始のタイミングを認識します。この情報をもとに、平日22時に実行されるジョブを作成できます：'mon,tue,wed,thu,fri 22'（略記：'mon..fri 22'）。この形式なら、ほとんどの合理的なスケジュールを直感的に記述できます。

---

#### 注記

時間は24時間形式で指定します。

---

設定を簡便かつ短くするため、ゲストごとに1つ以上の繰り返し時間を設定できます。これにより、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でレプリケーションが実行されます。例えば、レプリケーションを午前8時に開始し、午前9時まで15分間隔で繰り返す場合は「'8:00/15'」を使用します。

ここで、時間区切り記号（:）を使用しない場合、値は分として解釈されます。区切り記号を使用する場合、左側の値が時間、右側の値が分を表します。さらに、すべての可能な値に一致させるには\* を使用できます。

追加のアイデアを得るには、[以下の例](#)を参照してください。

### D.2 詳細仕様

---

<sup>1</sup> 詳細情報は `man 7 systemd.time` を参照

**weekdays**

曜日は英語の略称で指定します: sun、mon、tue、wed、thu、fri、sat。複数の曜日はカンマ区切りでリストとして使用できます。また「..」で区切った開始日と終了日を指定することで、範囲の日を設定できます（例: mon..fri）。これらの形式は混在可能です。省略された場合は「\*」が想定されます。

**時間形式**

時刻形式は時間と分の間隔リストで構成されます。時間と分は「:」で区切れます。時間と分は両方とも、日付と同じ形式で値のリストや範囲を指定できます。最初に時間が、

次に分が続きます。不要な場合は時間を省略できます。この場合、時間の値として「\*」が仮定されます。有効な値の範囲は、時間の場合0-23、分の場合0-59です。

**注記**

`systemd-analyze calendar` を使用すると、指定されたカレンダーイベントの仕様が有効かどうか、および次にいつトリガーされるかを確認できます。`--iterations=<N>` フラグを渡すことで、カレンダーイベントが次に <N> 回トリガーされるタイミングを出力させることもできます（<N> を有効な整数で置き換える必要があります）。

**D.2.1 例:**

特定の意味を持つ特殊値がいくつか存在します:

表 D.1: 特殊値

値	構文
minutely	* -* -** :* :00
毎時	* -* -** :00:00
毎日	* -* -* 00:00:00
週単位	月曜日 * -* -* 00:00:00
月次	* -* -01 00:00:00
年次または毎年	* -01-01 00:00:00
四半期	* -01,04,07,10-01 00:00:00
半年に一度	* -01,07-01 00:00:00

表 D.2: スケジュール例

スケジュール文字列	代替	意味
月、火、水、木、金	月～金	平日毎日 0:00
土、日	土..日	週末のみ 0:00
月・水・金	—	月曜、水曜 と金曜日の0:00のみ
12:05	12:05	毎日 午後12時05分
*5	0/5	毎5分

表 D.2: (続き)

スケジュール文字列	代替	意味
月～水 30/10	月・火・水 30/10	月曜日、火曜日、水曜日 30分後、40分後、50分後 毎正時
月～金 8..17,22:0/15	—	平日 8:00～18:00 及び 22:00～23:00 の間、15分ごとに 午後
金曜日 12..13:5/20	金曜日 12時、13時 5分/20分	金曜日 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まで、2時間おき
*	*/1	毎分（最小間隔）
*-05	—	毎月5日
土曜日 *-1..7 15:00	—	毎月第1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、2007年まで遡るQEMUで定義されたAMDおよびIntel x86-64/AMD64 CPUタイプのリストです。

### E.2 Intel CPU タイプ

#### インテルプロセッサ

- *Nehalem* : 第1世代インテル Core プロセッサー
- ナヘレム-IBRS (v2) : Spectre v1 保護機能追加 (+spec-ctrl)
- *Westmere* : 第1世代インテル Core プロセッサー (Xeon E7-)
- *Westmere-IBRS* (v2) : Spectre v1 保護機能追加 (+spec-ctrl)
- *SandyBridge* : 第2世代インテル® Core™ プロセッサー
- *SandyBridge-IBRS* (v2) : Spectre v1 保護機能追加 (+spec-ctrl)
- *IvyBridge* : 第3世代インテル Core プロセッサ
- *IvyBridge-IBRS* (v2) : Spectre v1 保護機能追加 (+spec-ctrl)
- *Haswell* : 第4世代インテル Core プロセッサー
- *Haswell-noTSX* (v2) : TSXを無効化 (-hle, -rtm)
- *Haswell-IBRS* (v3) : TSXを再追加、Spectre v1保護を追加 (+hle, +rtm, +spec-ctrl)
- *Haswell-noTSX-IBRS* (v4) : TSX を無効化 (-hle, -rtm)
- *Broadwell* : 第5世代インテル® Core™ プロセッサー
- *Skylake* : 第1世代Xeon Scalableサーバープロセッサ
- *Skylake-IBRS* (v2) : Spectre v1 保護を追加、CLFLUSHOPT を無効化 (+spec-ctrl, -clflushopt)

- *Skylake-noTSX-IBRS (v3)* : TSXを無効化 (-hle, -rtm)
- *Skylake-v4*: EPT切り替え機能を追加 (+vmx-eptp-switching)
- カスケードレイク: 第2世代Xeon Scalableプロセッサ
- *Cascadelake-v2* : arch\_capabilities MSRを追加 (+arch-capabilities, +rdctl-no, +ibrs-all, +skip-l1dfl-vmentry, +mds-no)
- *Cascadelake-v3*: TSX を無効化 (-hle, -rtm)
- *Cascadelake-v4* : EPT スイッチングを追加 (+vmx-eptp-switching)
- *Cascadelake-v5* : XSAVES を追加 (+xsaves, +vmx-xsaves)
- *Cooperlake* : 4 & 8 ソケットサーバー向け第 3 世代 Xeon Scalable プロセッサ
- *Cooperlake-v2* : XSAVESを追加 (+xsaves, +vmx-xsaves)
- *Icelake*: 第3世代Xeon Scalableサーバープロセッサ
- *Icelake-v2* : TSX を無効化 (-hle, -rtm)
- *Icelake-v3* : arch\_capabilities MSR を追加 (+arch-capabilities, +rdctl-no, +ibrs-all, +skip-l1dfl-vmentry, +mds-no, +pschange-mc-no, +taa-no)
- *Icelake-v4* : 不足しているフラグを追加 (+sha-ni, +avx512ifma, +rdpid, +fsrm, +vmx-rdseed-exit, +vmx-pml, +vmx-eptp-switching)
- *Icelake-v5* : XSAVES を追加 (+xsaves, +vmx-xsaves)
- *Icelake-v6* : 「5-level EPT」を追加 (+vmx-page-walk-5)
- *Sapphire Rapids* : 第4世代Xeon Scalableサーバープロセッサー

## E.3 AMD CPUの種類

### AMD プロセッサ

- *Opteron\_G3* : K10
- *Opteron\_G4* : Bulldozer
- *Opteron\_G5* : Piledriver
- *EPYC* : 第1世代Zenプロセッサ
- *EPYC-IBPB (v2)* : Spectre v1 保護機能追加 (+ibpb)
- *EPYC-v3* : 不足しているフラグを追加 (+perfctr-core、+clzero、+xsaveerptr、+xsaves)
- *EPYC-Rome* : 第2世代Zenプロセッサ
- *EPYC-Rome-v2* : Spectre v2、v4 保護を追加 (+ibrs、+amd-ssbd)
- *EPYC-Milan* : 第3世代Zenプロセッサ
- *EPYC-Milan-v2* : 不足しているフラグを追加 (+vaes, +vpclmulqdq, +stibp-always-on, +amd-psfd, +no-nested-data-bp, +fence-always-serializing, +null-sel-clr-base)

## 付録 F

### ファイアウォールマクロ定義

*Amanda*

Amanda バックアップ

アクション	プロト	dport	sport
PARAM	udp	10080	
PARAM	tcp	10080	

*認証*

認証 (identd) トラフィック

アクション	プロトコル	宛先ポート	sport
PARAM	tcp	113	

*BGP*

ボーダーゲートウェイプロトコルトラフィック

アクション	プロトコル	dport	sport
PARAM	tcp	179	

*BitTorrent*

BitTorrent 3.1 およびそれ以前のバージョンの BitTorrent トラフィック

アクション	プロト	dport	sport
PARAM	tcp	6881:6889	
PARAM	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	dport	sport
PARAM	tcp	6881:6999	
PARAM	udp	6881	

**CVS**

Concurrent Versions System pserver トラフィック

アクション	プロト	dport	sport
PARAM	tcp	2401	

**Ceph**

Ceph ストレージクラスタのトラフィック (Ceph モニター、OSD および MDS デーモン)

アクション	プロトコル	dport	sport
PARAM	tcp	6789	
PARAM	tcp	3300	
PARAM	tcp	6800:7300	

**Citrix**

Citrix/ICA トラフィック (ICA、ICA ブラウザ、CGP)

アクション	プロト	dport	sport
PARAM	tcp	1494	
PARAM	udp	1604	
PARAM	tcp	2598	

**DAAP**

デジタルオーディオアクセスプロトコルトラフィック (iTunes、Rhythmboxデーモン)

アクション	プロト	dport	sport
PARAM	tcp	3689	
PARAM	udp	3689	

**DCC**

分散チェックサムクリアリングハウススマイルタリングメカニズム

アクション	プロト	dport	sport
PARAM	tcp	6277	

**DHCP転送**

転送されたDHCP トラフィック

アクション	プロトコル	宛先ポート	sport
PARAM	udp	67:68	67:68

**DHCPv6**

DHCPv6 トラフィック

アクション	プロトコル	宛先ポート	sport
PARAM	udp	546:547	546:547

*DNS* ドメインネームシステムトラフィック (upd および tcp)

アクション	プロト	dport	sport
PARAM	udp	53	
PARAM	tcp	53	

*Distcc* 分散コンパイラサービス

アクション	プロト	dport	sport
PARAM	tcp	3632	

*FTP* ファイル転送プロトコル

アクション	プロト	dport	sport
PARAM	tcp	21	

*Finger* フィンガープロトコル (RFC 742)

アクション	プロト	dport	sport
PARAM	tcp	79	

*GNUnet* GNUnet セキュア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	dport	sport
PARAM	tcp	2086	
PARAM	udp	2086	
PARAM	tcp	1080	
PARAM	udp	1080	

*GRE* 汎用ルーティングカプセル化トンネリングプロトコル

アクション	プロト	dport	sport
PARAM	47		

*Git* Git分散リビジョン管理トラフィック

アクション	プロト	dport	sport
PARAM	tcp	9418	

*HKP* OpenPGP HTTP キーサーバープロトコルトラフィック

アクション	プロト	dport	sport
PARAM	tcp	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	dport	sport
PARAM	tcp	80	

*HTTPS* SSL 上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	dport	sport
PARAM	tcp	443	

*ICPV2* インターネットキャッシュプロトコル V2 (Squid) トラフィック

アクション	プロト	dport	sport
PARAM	udp	3130	

*ICQ* AOL インスタントメッセンジャートラフィック

アクション	プロト	dport	sport
PARAM	tcp	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	dport	sport
PARAM	tcp	143	

*IMAPS* SSL 上の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	dport	sport
PARAM	tcp	993	

*IPIP*

IPIPカプセル化トラフィック

アクション	プロト	dport	sport
PARAM	94		

*IPsec*

IPsec トライフィック

アクション	プロト	宛先ポート	sport
PARAM	udp	500	500
PARAM	50		

*IPsecah*

IPsec 認証 (AH) トライフィック

アクション	プロト	宛先ポート	sport
PARAM	udp	500	500
PARAM	51		

*IPseccnat*

IPsec トライフィックとNAT トランザクション

アクション	プロト	宛先ポート	sport
PARAM	udp	500	
PARAM	udp	4500	
PARAM	50		

*IRC*

インターネットリレーチャットトライフィック

アクション	プロト	dport	sport
PARAM	tcp	6667	

*Jetdirect*

HP Jetdirect 印刷

アクション	プロト	dport	sport
PARAM	tcp	9100	

*L2TP*

レイヤ2トンネリングプロトコルトライフィック

アクション	プロト	dport	sport
PARAM	udp	1701	

*LDAP* 軽量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	dport	sport
PARAM	tcp	389	

*LDAPS* 軽量ディレクトリアクセスプロトコル (LDAP) のトラフィックを保護

アクション	プロト	dport	sport
PARAM	tcp	636	

*MDNS* マルチキャスト DNS

アクション	プロトコル	dport	sport
PARAM	udp	5353	

*MSNP* Microsoft Notification Protocol

アクション	プロト	dport	sport
PARAM	tcp	1863	

*MSSQL* Microsoft SQL Server

アクション	プロト	dport	sport
PARAM	tcp	1433	

*メール* メールトラフィック (SMTP、SMTPS、Submission)

アクション	プロト	dport	sport
PARAM	tcp	25	
PARAM	tcp	465	
PARAM	tcp	587	

*Munin* Munin ネットワークリソース監視トラフィック

アクション	プロト	dport	sport
PARAM	tcp	4949	

*MySQL*

MySQL サーバー

アクション	プロト	dport	sport
PARAM	tcp	3306	

*NNTP*

NNTP トラフィック (Usenet)。

アクション	プロト	dport	sport
PARAM	tcp	119	

*NNTPS*

暗号化された NNTP トラフィック (Usenet)

アクション	プロト	dport	sport
PARAM	tcp	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	dport	sport
PARAM	udp	123	

NeighborDiscoveryPv6 ネイバーソリシテーション、ネイバーおよびルータアドバタイズメント

アクション	プロト	dport	sport
PARAM	icmpv6	router-solicitation	
PARAM	icmpv6	ルータ広告	
PARAM	icmpv6	neighbor-solicitation	
PARAM	icmpv6	neighbor-広告	

*OSPF*

OSPF マルチキャストトラフィック

アクション	プロトコル	宛先ポート	sport
PARAM	89		

*OpenVPN*

OpenVPN トラフィック

アクション	プロト	dport	sport
PARAM	udp	1194	

*PCA* シマンテック PCAnywhere (tm)

アクション	プロト	dport	sport
PARAM	udp	5632	
PARAM	tcp	5631	

*PMG* Proxmox メールゲートウェイ Web インターフェース

アクション	プロト	dport	sport
PARAM	tcp	8006	

*POP3* POP3 トラフィック

アクション	プロト	dport	sport
PARAM	tcp	110	

*POP3S* 暗号化された POP3 トラフィック

アクション	プロト	dport	sport
PARAM	tcp	995	

*PPtP* ポイントツーポイントトンネリングプロトコル

アクション	プロト	dport	sport
PARAM	47		
PARAM	tcp	1723	

*Ping* ICMP エコー要求

アクション	プロト	宛先ポート	送信元ポート
PARAM	icmp	echo-request	

*PostgreSQL* PostgreSQL サーバー

アクション	プロト	dport	sport
PARAM	tcp	5432	

プリンタ

ラインプリンタプロトコル印刷

アクション	プロト	dport	sport
PARAM	tcp	515	

RDP

Microsoft 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	dport	sport
PARAM	tcp	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	dport	sport
PARAM	udp	520	

RNDC

BIND リモート管理プロトコル

アクション	プロト	dport	sport
PARAM	tcp	953	

Razor

Razor 迷惑メール対策システム

アクション	プロト	dport	sport
PARAM	tcp	2703	

Rdate

リモート時間取得 (rdate)

アクション	プロト	dport	sport
PARAM	tcp	37	

Rsync

Rsync サーバー

アクション	プロト	dport	sport
PARAM	tcp	873	

**SANE**

SANE ネットワークスキャン

アクション	プロト	dport	sport
PARAM	tcp	6566	

**SMB**

Microsoft SMB トラフィック

アクション	プロト	dport	sport
PARAM	udp	135,445	
PARAM	udp	137:139	
PARAM	udp	1024:65535	137
PARAM	tcp	135,139,445	

**SMBswat**

Samba Web 管理ツール

アクション	プロト	dport	sport
PARAM	tcp	901	

**SMTP**

簡易メール転送プロトコル

アクション	プロト	dport	sport
PARAM	tcp	25	

**SMTPTS**

暗号化簡易メール転送プロトコル

アクション	プロト	dport	sport
PARAM	tcp	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	dport	sport
PARAM	udp	161:162	
PARAM	tcp	161	

**SPAMD**

Spam Assassin SPAMD トラフィック

アクション	プロト	dport	sport
PARAM	tcp	783	

*SPICEproxy* Proxmox VE SPICE ディスプレイプロキシトラフィック

アクション	プロトコル	dport	sport
PARAM	tcp	3128	

*SSH* セキュアシェルトラフィック

アクション	プロト	dport	スポーツ
PARAM	tcp	22	

*SVN* Subversion サーバー (svnserve)

アクション	プロト	dport	sport
PARAM	tcp	3690	

*SixXS* SixXS IPv6 導入およびトンネルプローカー

アクション	プロト	dport	sport
PARAM	tcp	3874	
PARAM	udp	3740	
PARAM	41		
PARAM	udp	5072,8374	

*Squid* Squid Web プロキシトラフィック

アクション	プロト	dport	sport
PARAM	tcp	3128	

*送信* メールメッセージ送信トラフィック

アクション	プロトコル	送信ポート	sport
PARAM	tcp	587	

*Syslog* Syslog プロトコル (RFC 5424) トラフィック

アクション	プロト	dport	sport
PARAM	udp	514	
PARAM	tcp	514	

*TFTP* Trivial File Transfer Protocol トラフィック

アクション	プロト	dport	sport
PARAM	udp	69	

*Telnet* Telnet トラフィック

アクション	プロトコル	dport	送信ポート
PARAM	tcp	23	

*Telnet* SSL経由のTelnet

アクション	プロト	dport	sport
PARAM	tcp	992	

*時間* RFC 868 時刻プロトコル

アクション	プロト	dport	sport
PARAM	tcp	37	

*Trcrt* Traceroute (最大 30 ホップ) トラフィック

アクション	プロト	dport	sport
PARAM	udp	33434:33524	
PARAM	icmp	echo-request	

*VNC* VNC ディスプレイ 0 ~ 99 の VNC トラフィック

アクション	プロト	dport	sport
PARAM	tcp	5900:5999	

*VNC* Vncservers から Vncviewers への VNC トラフィック (リスンモード)

アクション	プロト	dport	sport
PARAM	tcp	5500	

*Web*

WWW トライフィック (HTTP および HTTPS)

アクション	プロト	dport	sport
PARAM	tcp	80	
PARAM	tcp	443	

*Web キャッシュ*

Web キャッシュ/プロキシ トライフィック (ポート 8080)

アクション	プロト	dport	sport
PARAM	tcp	8080	

*Webmin*

Webmin トライフィック

アクション	プロト	dport	sport
PARAM	tcp	10000	

*Whois*

Whois (nickname, RFC 3912) トライフィック

アクション	プロト	dport	sport
PARAM	tcp	43	

## 付録 G マークダウン入門

Markdown は、ウェブライター向けのテキストから HTML への変換ツールです。Markdown を使用すると、読みやすく書きやすいプレーンテキスト形式で記述し、構造的に有効な XHTML（または HTML）に変換できます。

— John Gruber <https://daringfireball.net/projects/markdown/>

Proxmox VE のウェブインターフェースは、ノードおよび仮想ゲストのノートにおいて、リッチテキストフォーマットをレンダリングするための Markdown の使用をサポートしています。

Proxmox VE は CommonMark をサポートし、テーブルやタスクリストなど GFM (GitHub Flavoured Markdown) のほとんどの拡張機能に対応しています。

### G.1 Markdownの基本

ここでは基本のみを説明します。より詳細なリソースについては、例えば <https://www.markdownguide.org/> などでウェブ検索してください。

#### G.1.1 見出し

```
# これは見出し h1## これは見出し h2  
##### これは見出し h5 です
```

#### G.1.2 強調

強調には \* text\* または \_text\_ を使用してください。

強調には \*\* text\*\* または \_\_text\_\_ を太字の強調表示に使用します。組み合  
わせも可能です。例：

```
_You ** can** combine them_
```

## G.1.3 リンク

リンクの自動検出機能を利用できます。例えば、<https://forum.proxmox.com/> はクリック可能なリンクに変換されます。

リンクテキストを制御することも可能です。例：

[角括弧内の部分がリンクテキストになります] (<https://forum.proxmox.com/>)

/

## G.1.4 リスト

### 番号なしリスト

無順序リストには \* または – を使用します。例：

- \* 項目 1
- \* 項目 2
- \* 項目 2a
- \* 項目 2b

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できます。

### 番号付きリスト

1. 項目 1
1. 項目 2
1. 項目 3
  1. 項目 3a
  1. 項目 3b

---

### 注記

順序付きリストの整数部分は正確である必要はなく、自動的に番号が振られます。

---

### タスクリスト

タスクリストでは、未完了のタスクには空のボックス [ ] を、完了したタスクにはXの入ったボックスを使用します。例：

- [X] 最初のタスクは完了済み！
- [X] 二つ目も完了
- [ ] これはまだ未完了
- [ ] これも未完了

## G.1.5 表

テーブルでは、パイプ記号 | で列を区切り、-でヘッダーと本文を区切れます。この区切り部分でテキストの配置（左揃え・中央揃え・右揃え）を設定できます。

左側の列も	右カラム	の追加	cols.	中央揃え機能↔
	----- ↔			
左 foo	right foo	最初	行	ここ   &gt;center&lt;↔
左 左	右 バー	セカンド	行	こちら   12345↔
左 baz	右 baz	第三	行	ここ   テスト↔
左 zab	right zab	Fourth  Row  Here  ↔	&#x2601;&#xe0f;&#x2601;&#fe0f;&#x2601;&#fe0f;&#x2601;&#xe0f;	
左 rab	right rab	そして	最後  こちら   終わり	↔

列を空白で綺麗に揃える必要はありませんが、そうするとテーブルの編集が容易になります。

## G.1.6 ブロック引用

ブロック引用は、行の先頭に> を付けることで入力できます。プレーンテキストメールと同様の方法です。

```
> 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↔
構文を持ち、
> 2004年にJohn GruberとAaron Swartzによって開発されました。
>
>>Markdownは、↔
オンラインディスカッションフォーラムでのメッセージ作成、
>>プレーンテキストエディタでリッチテキストを作成するために使用されます。
```

## G.1.7 コードとスニペット

バッククオート (`) を使用すると、数語または数段落の処理を回避できます。これにより、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防ぐのに役立ちます。

### インラインコード

単一バッククオートで囲んだ行の一部は、コードをインラインで記述できます。例：

```
This hosts IP address is .
```

### コードブロック全体

複数行にわたるコードブロックには、三重バックティック (```) でブロックの開始と終了を指定できます。例：

```
...
# ここに覚えておきたいネットワーク設定です auto vmbr2
iface vmbr2 inet static
    address 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 Free Documentation License

バージョン 1.3、2008年11月3日

Copyright (C) 2000, 2001, 2002, 2007, 2008 Free Software Foundation,  
Inc.  
<<http://fsf.org/>>;

本ライセンス文書の完全な複製物の複製および配布は誰でも許可されていますが、変更は許可されていません。

#### 0. 前文

本ライセンスの目的は、マニュアル、教科書、その他の機能的で有用な文書を「自由」なものとすることである。すなわち、商業的・非商業的を問わず、改変の有無にかかわらず、誰もがそれを複製・再配布する実効的な自由を保証することである。副次的目的として、本ライセンスは、著作者および出版者が自らの著作に対して評価を得る方法を維持しつつ、他者による改変について責任を負わないようにするものである。

本ライセンスは一種の「コピーレフト」であり、この文書に基づく派生作品も同様に自由でなければならないことを意味します。これはフリーソフトウェア向けに設計されたコピーレフトライセンスであるGNU一般公衆利用許諾契約書を補完するものです。

本ライセンスは、フリーソフトウェアのマニュアルに使用することを意図して設計されました。なぜなら、フリーソフトウェアにはフリーな文書が必要だからです。すなわち、フリープログラムには、ソフトウェア自体と同じ自由を提供するマニュアルが付属すべきです。ただし、本ライセンスはソフトウェアマニュアルに限定されません。主題や印刷書籍として出版されるかどうかに問わらず、あらゆるテキスト作品に使用できます。主に、指導や参照を目的とする作品に対して本ライセンスの使用を推奨します。

#### 1. 適用範囲と定義

本ライセンスは、著作権者により「本ライセンスの条件下で配布可能」との通知が記載された、あらゆる媒体のマニュアルその他の著作物に適用されます。当該通知は、本契約に定める条件下で当該著作物を利用するための、期間無制限・全世界対象・ロイヤリティフリーのライセンスを付与します。以下における「文書」とは、かかるマニュアルまたは著作物を指す。公衆のいかなる者もライセンサーであり、「あなた」として扱われる。著作権法上許可を必要とする方法で著作物を複製、改変、または頒布する場合、あなたはこのライセンスを受け入れるものとする。

「改変版」とは、文書またはその一部を、そのまま複製したもの、あるいは改変を加えたもの、および／または他の言語に翻訳したものと包含するあらゆる著作物を意味する。

「**二次的セクション**」とは、当該文書の付録または前書きとして明示されたセクションであり、当該文書の発行者または著者と、その文書の全体的な主題（または関連事項）との関係のみを扱い、その全体的な主題に直接該当する内容を含まないものである。（したがって、文書が数学の教科書の一部である場合、二次的セクションは数学を説明してはならない。）この関係は、主題または関連事項との歴史的関連性、あるいはそれらに関する法的・商業的・哲学的・倫理的・政治的立場に関する事項である可能性がある。

「**不变セクション**」とは、本ライセンスに基づき公開される旨の通知において、不变セクションとして指定されたタイトルを持つ特定の二次的セクションを指す。セクションが上記の二次的セクションの定義に該当しない場合、不变セクションとして指定することは許されない。文書は不变セクションを全く含んでいない場合もある。文書が不变セクションを特定していない場合、不变セクションは存在しない。

「**表紙テキスト**」とは、文書が本ライセンスの下で公開されていることを示す通知において、表紙テキストまたは裏表紙テキストとして記載される特定の短い文章である。表紙テキストは最大5語、裏表紙テキストは最大25語とする。

「**透明な**」文書の複製とは、一般公衆が入手可能な仕様で表される機械可読の複製を意味し、汎用テキストエディタ（またはピクセルで構成される画像の場合は汎用ペイントプログラム、図面の場合は広く利用可能な図面エディタ）を用いて文書を直接修正するのに適し、テキストフォーマッタへの入力、またはテキストフォーマッタへの入力に適した様々な形式への自動変換に適したものとする。透明なファイル形式で作成されたコピーであっても、そのマークアップ（またはマークアップの欠如）が読者の後続の改変を妨げる、あるいは抑制するように調整されている場合は透明ではない。画像形式は、相当量のテキストに使用される場合、透明ではない。「透明」でないコピーは「不透明」と呼ばれる。

透明なコピーに適した形式の例としては、マークアップのないブレーンASCII、Texinfo入力形式、LaTeX入力形式、公開されているDTDを使用したSGMLまたはXML、および人間による修正を目的とした標準準拠の簡易HTML、PostScript、PDFが挙げられる。透明な画像フォーマットの例としては、PNG、XCF、JPGが挙げられる。不透明なフォーマットには、専有ワードプロセッサでのみ読み書き可能な専有フォーマット、DTDや処理ツールが一般に入手できないSGMLやXML、および一部のワードプロセッサが出力専用に生成する機械生成のHTML、PostScript、PDFが含まれる。

「**タイトルページ**」とは、印刷書籍において、タイトルページ自体に加え、本ライセンスがタイトルページへの記載を要求する事項を可読性をもって保持するために必要な後続のページを意味する。タイトルページそのものを持たない形式の著作物においては、「**タイトルページ**」とは、本文の開始前に位置し、著作物のタイトルが最も顕著に表示される箇所の近傍にあるテキストを意味する。

「**発行者**」とは、文書を公衆に配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、文書の命名されたサブユニットであり、そのタイトルが正確にXYZであるか、XYZを別の言語で訳したテキストの後に括弧付けてXYZを含むものである。（ここでXYZは、以下に記載する特定のセクション名、「謝辞」、「献辞」、「推薦文」、「沿革」などを表す。）文書を改変する際に、そのようなセクションの「**タイトルを保持する**」とは、この定義に従って、それが「**タイトル XYZ**」のセクションのままであることを意味する。

本ライセンスが文書に適用される旨の通知に併せて、文書に保証免責事項が含まれる場合があります。これらの保証免責事項は、保証の否認に関してのみ、参照により本ライセンスに含まれるものとみなされます。これらの保証免責事項が持つその他の含意はすべて無効であり、本ライセンスの意味に影響を及ぼしません。

## 2. 完全複製

本ライセンス、著作権表示、および本ライセンスが文書に適用される旨のライセンス通知を全ての複製物に再現し、かつ本ライセンスの条件に一切の追加条件を加えないことを条件として、いかなる媒体においても、商業的または非商業的に、文書を複製および頒布することができます。作成または頒布する複製物の閲覧またはさらなる複製を妨害または制御する技術的措置を使用してはなりません。ただし、複製物と引き換えに報酬を受け取ることができます。十分な数の複製物を配布する場合、第3項の条件も遵守しなければなりません。

上記と同じ条件のもとで、複製物を貸与することも、複製物を公に展示することもできます。

## 3. 大量複製

文書を印刷物（または通常印刷された表紙を持つ媒体）で100部以上出版する場合、かつ文書のライセンス通知が表紙テキストを要求する場合、これらの表紙テキストをすべて明瞭かつ読みやすく記載した表紙で複製物を包まなければなりません。表紙テキストは表紙に、裏表紙テキストは裏表紙に記載します。両表紙には、これらの複製物の発行者としてあなたを明確かつ読みやすく識別できる表示が必要です。表紙には完全なタイトルを、タイトルのすべての単語が同等に目立ち見えるように提示しなければなりません。表紙には追加の素材を加えることもできます。表紙への変更が限定され、かつ文書のタイトルを保持しこれらの条件を満たす場合、その他の点では逐語的複製として扱われます。

いずれかの表紙に必要な記載事項が過多で可読性を損なう場合、最初に列挙された事項（合理的に収まる範囲で）を実際の表紙に記載し、残りは隣接ページに継続して記載すること。

文書番号100を超える不透明なコピーを出版または配布する場合、各不透明なコピーに機械可読の透明なコピーを添付するか、各不透明なコピーに、一般的なネットワーク利用者が公的な標準ネットワークプロトコルを用いて追加素材を含まない文書の完全な透明なコピー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記載または添付しなければなりません。後者の選択肢を採用する場合、不透明なコピーを大量に配布し始める際には、当該透明なコピーが、その版の不透明なコピーを（直接または代理店・小売業者を通じて）最後に一般に配布してから少なくとも1年間は、指定された場所でアクセス可能な状態を維持するよう、合理的に慎重な措置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、文書著者に連絡し、更新版を提供する機会を与えることが望ましいが、必須ではない。

## 4. 変更

文書を改変した版（以下「改変版」）を、上記第2項および第3項の条件に従って複製および頒布することができます。ただし、改変版を本ライセンスと全く同じ条件で公開し、改変版が文書の役割を果たすようにしなければなりません。これにより、改変版の複製を所持する者すべてに対して、改変版の頒布および改変を許可することになります。さらに、改変版では以下のことを行わなければなりません：

- A. 表紙（および表紙がある場合はカバー）には、文書および過去のバージョン（存在する場合、文書の「履歴」セクションに記載されるべきもの）のタイトルとは異なるタイトルを使用すること。過去のバージョンの元の出版者が許可した場合に限り、そのバージョンと同じタイトルを使用できる。
- B. 改変版における改変の著作者として、改変版の著作者である一人以上の個人または団体を、原著作物の主要な著作者少なくとも5名（5名未満の場合は全員）と共に、タイトルページに記載すること。ただし、彼らがこの要件を免除する場合を除く。

- C. タイトルページには、改変版の発行者として、その発行者の名称を記載すること。
- D. 文書のすべての著作権表示を保持すること。
- E. 他の著作権表示に隣接して、あなたの変更に対する適切な著作権表示を追加すること。
- F. 著作権表示の直後に、以下の付録に示す形式で、本ライセンスの条件下で改変版を使用することを公衆に許可するライセンス表示を含めること。
- G. そのライセンス通知には、文書内のライセンス通知に記載されている不变セクションおよび必須カバーテキストの完全なリストをそのまま記載すること。
  - 。
- H. 本ライセンスの改変されていないコピーを含めること。
- I. 「歴史」と題されたセクションを保持し、そのタイトルを保持し、改変版の見出しページに記載されている改変版のタイトル、年、新たな著者、出版社の少なくともこれらを記載した項目を追加すること。文書に「歴史」と題されたセクションが存在しない場合、そのタイトルページに記載されている文書のタイトル、年、著者、出版社を記載したセクションを作成し、前文で述べた改変版を説明する項目を追加すること。
- J. 文書内で公開アクセス可能な透明なコピーのネットワーク場所が記載されている場合はこれを保持し、同様に、その文書が基にした以前のバージョンのネットワーク場所も保持すること。これらは「歴史」セクションに記載してもよい。ただし、当該文書より少なくとも4年以上前に公開された作品、または参照されるバージョンの原出版者が許可を与えた場合は、ネットワーク場所の記載を省略してもよい。
- K. 「謝辞」または「献辞」と題されたセクションについては、セクションのタイトルを保持し、そのセクション内で述べられている各寄稿者への謝辞および／または献辞の内容とトーンをすべて保持すること。
- L. 文書の不变セクションはすべて、そのテキストとタイトルを変更せずに保持すること。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. 「推薦文」と題されたセクションは削除すること。この種のセクションは改変版に含めることはできない。
- N. 既存のセクションを「推奨」と改題したり、不变セクションとタイトルが競合する形で改題したりしてはならない。
- O. 保証免責事項はすべて保持すること。

改変版に二次的のセクションに該当し、かつ文書から複製された内容を含まない新たな前書きセクションや付録が含まれる場合、任意でこれらのセクションの一部または全てを不变セクションとして指定できる。これを行うには、改変版のライセンス通知にある不变セクションのリストにそれらのタイトルを追加すること。これらのタイトルは他のセクションタイトルと区別されなければならない。

「推薦文」と題するセクションを追加することができます。ただし、このセクションには、様々な団体による改変版への推薦文のみを含めること。例えば、査読の声明や、テキストが標準の権威ある定義として組織によって承認された旨の声明など。

改変版における表紙テキスト一覧の末尾に、最大5語の「表紙テキスト」と最大25語の「裏表紙テキスト」を追加できます。同一の主体（またはその主体による手配）によって追加できる表紙テキストは、表表紙用と裏表紙用それぞれ1箇所のみとする。文書に既に同一の表紙用テキスト（あなた自身、またはあなたが代理を務める同一の主体による手配で以前に追加されたもの）が含まれている場合、別のものを追加することはできない。ただし、古いものを置き換えることは可能であり、その場合は古いものを追加した前の出版者の明示的な許可を得なければならない。

追加されたカバーテキストが既に存在する場合、別のものを追加することはできません。ただし、以前のカバーテキストを追加した出版者の明示的な許可を得て、古いものを置き換えることは可能です。

本ライセンスにより、文書の著者および出版者は、いかなる改変版についても、宣伝目的でその名称を使用すること、または改変版を推奨することを明示または暗示することを許可するものではありません。

## 5. 文書の結合

あなたは、この文書を、本ライセンスに基づいて公開されている他の文書と、上記のセクション 4 で定義されている変更版に関する条件に従って組み合わせることができます。ただし、その組み合わせには、すべての元の文書の不变セクションを、変更せずにすべて含め、それらをすべて、組み合わせた作品のライセンス通知において不变セクションとしてリストし、それらの保証の免責事項をすべて保持しなければなりません。

結合された著作物には、本ライセンスの写しを1部のみ含めればよく、複数の同一の不变セクションは1つの写しで置き換えてもよい。同じ名称だが内容が異なる不变セクションが複数ある場合、各セクションのタイトルを固有のものとするため、その末尾に括弧で囲み、そのセクションの元の著者または出版者の名前（判明している場合）または固有の番号を追加すること。結合作品のライセンス通知に記載される不变セクション一覧のセクションタイトルについても同様の調整を行ってください。

結合時には、各原著作物内の「歴史」と題されたセクションを全て統合し、「歴史」と題された单一のセクションを形成すること。同様に、「謝辞」と題されたセクション、および「献辞」と題されたセクションも統合すること。「推薦文」と題されたセクションは全て削除すること。

## 6. 文書の集合体

本ライセンスに基づき公開された文書と他の文書からなるコレクションを作成し、各文書内の個別の本ライセンスのコピーを、コレクション内に含まれる単一のコピーに置き換えることができます。ただし、その他の点において各文書の逐語的複製に関する本ライセンスの規則を遵守することを条件とします。

そのようなコレクションから単一の文書を抽出し、本ライセンスに基づき個別に配布することができます。ただし、抽出した文書に本ライセンスの写しを挿入し、その文書の逐語的複製に関する本ライセンスのその他の規定をすべて遵守しなければなりません。

## 7. 独立した著作物との集合体

文書またはその派生作品を、他の独立した文書や作品と、記憶媒体または流通媒体の1巻内に、またはその上に編集したものは、その編集によって生じる著作権が、個々の作品が許容する範囲を超えて編集物の利用者の法的権利を制限するために使用されない場合、「集合体」と呼ばれる。文書が集合体に含まれる場合、このライセンスは、それ自体が文書の派生作品ではない集合体内の他の作品には適用されない。

第3条のカバーテキスト要件が当該文書の複製物に適用される場合、文書が集合体全体の半分未満であるときは、文書のカバーテキストは集合体内で文書を囲むカバー、または文書が電子形式である場合はカバーに相当する電子的表示に配置できる。それ以外の場合は、集合体全体を囲む印刷されたカバーに表示されなければならない。

## 8. 翻訳

翻訳は一種の改変と見なされるため、第4節の条件に基づき文書の翻訳を配布することができます。不变セクションを翻訳で置き換えるには、それらの著作権者からの特別な許可が必要です

著作権者からの特別な許可が必要ですが、不变セクションの原文に加えて、その一部または全部の翻訳を含めることは可能です。本ライセンスの翻訳版、文書内の全てのライセンス表示、および免責事項の翻訳を含めることができます。ただし、本ライセンスの原文（英語版）およびそれらの表示・免責事項の原文も併せて含めることが条件です。翻訳版と原文（本ライセンス、表示、免責事項）との間に矛盾がある場合、原文が優先されます。

文書内のセクションが「謝辞」、「献辞」、または「沿革」と題されている場合、そのタイトル（第1条）を保持する要件（第4条）は、通常、実際のタイトルを変更することを要求します。

## 9. 終了

本ライセンスで明示的に認められている場合を除き、文書を複製、改変、サプライセンス、または配布することはできません。それ以外の方法で複製、改変、サプライセンス、または配布しようとする試みは無効であり、本ライセンスに基づくあなたの権利は自動的に終了します。

ただし、本ライセンス違反を全て中止した場合、特定の著作権者からのライセンスは (a) 著作権者が明示的かつ最終的にライセンスを終了させない限り暫定的に、(b) 著作権者が中止後60日以内に合理的な手段で違反通知を行わなかった場合には恒久的に、回復される。

さらに、特定の著作権者からのあなたのライセンスは、その著作権者が合理的な手段で違反を通知し、かつ、その著作権者から（いかなる著作物についても）本ライセンス違反の通知を初めて受けた場合、かつ、通知受領後30日以内に違反を是正した場合、恒久的に回復されます。

本項に基づくあなたの権利の終了は、本ライセンスに基づきあなたから複製物または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、かつ恒久的に回復されていない場合、同一の素材の一部または全部の複製物を受領しても、それを使用する権利は一切付与されない。

## 10. 本ライセンスの将来の改訂

フリーソフトウェア財団は隨時、GNUフリードキュメンテーションライセンスの新規改訂版を公開することができます。そのような新バージョンは、精神的には現行版と同様ですが、新たな問題や懸念に対処するため、細部が異なる場合があります。詳細は <http://www.gnu.org/copyleft/> をご覧ください。

本ライセンスの各バージョンには識別用のバージョン番号が付与されます。文書が特定の番号付きバージョンまたは「それ以降のバージョン」を適用対象と指定している場合、指定されたバージョンの条件に従うか、フリーソフトウェア財団によって（草案としてではなく）公開されたそれ以降のバージョンのいずれかの条件に従うかを選択できます。文書が本ライセンスのバージョン番号を指定していない場合、フリーソフトウェア財団が公表した（草案ではない）いずれのバージョンでも選択できます。文書が将来のライセンスバージョン使用の決定権を代理人に委任すると規定している場合、その代理人が公表したバージョン受諾声明により、当該バージョンを文書に適用する権限が恒久的に付与されます。

## 11. 再ライセンス

「大規模多作者共同編集サイト」（または「MMCサイト」）とは、著作権対象作品を公開し、かつ誰でもそれらの作品を編集できる顕著な機能を提供するワールドワイドウェブサーバーを意味する。誰でも編集可能な公開ウィキはそのようなサーバーの一例である。サイトに含まれる「大規模多作者共同編集」（または「MMC」）とは、MMCサイトにこのように公開された著作権対象作品の集合を意味する。

「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クリエイティブ・コモンズ・コーポレーションが公表したクリエイティブ・コモンズ 表示-継承 3.0 ライセンス、および同組織が公表する将来の同ライセンスのコピーレフト版を意味する。

「組み込む」とは、文書の一部または全部を、別の文書の一部として公開または再公開することを意味する。

MMCが「再ライセンス対象」となるのは、本ライセンスに基づきライセンスされ、かつ本MMC以外の場所で本ライセンスに基づき最初に公開され、その後MMCに全体または一部が組み込まれた全ての著作物が、(1) カバーテキストや不变セクションを持たず、(2) したがって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、当該MMCが再ライセンス対象である場合に限り、2009年8月1日以前に、サイト内に含まれるMMCを同一サイトでCC-BY-SAの下で再公開することができる。